

UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Evolving a Tool Set for Biological Research and Education

Permalink

<https://escholarship.org/uc/item/9fp7j60d>

Author

Hill, Douglas Andrew

Publication Date

2016

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Evolving a Tool Set for Biological Research and Education

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Bioengineering

by

Douglas A. Hill

December 2016

Dissertation Committee:

Dr. William H. Grover, Chairperson
Dr. Victor G. J. Rodgers
Dr. Dimitrios Morikis

Copyright by
Douglas A. Hill
2016

The Dissertation of Douglas A. Hill is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I have had tremendous support on my journey toward earning a PHD, and would like to acknowledge all those who supported me. My appreciation begins with Dr. Jiayu Liao, who welcomed me into his lab and started my education into bioengineering. I joined the IGERT program, which continued my education. I would like to thank the professors and staff who supported me in the program, which included: Dr. Victor Rodgers, Dr. Jiayu Liao, Dr. David Carter, Dr. Prudence Talbot, Dr. Philip Brisk, Dr. Dimitrios Morikis and Dr. Bir Bhanu at UC Riverside. I had many enriching discussions with them, as well as many of the IGERT students, which included: Ben Sanders, Ben Guan, Nolan Ung, Ason Tambo, Jo-Hao Weng and Devin McBride. For a brief time I was Dr. Victor Rodgers PHD candidate, where he continued to give me his sage advice and support. My sincere thanks. I then became Dr. William Grovers PHD student, where, because of the environment he created for me, I was able to flourish. This environment included procuring a lab for me, which he allows me to call my own. Continuously, guiding interns to the TEC center (my lab), Dr. Grover, again, allows me to call them my interns. I take great ownership and pride in work we have created together and am forever thankful for the support and advice he has given me. He is an amazing person, and I cannot thank him enough. I also wanted to thank the students in his lab. I want to thank my son, Casey Hill, and his finance, Lindsey Anderson, for taking a year off to help me with my program. They were some of my first interns and were instrumental to the success of the program through grant writing, intern recruiting, and component development. I want to thank my wife and Dr. Grover for endless editing. Finally, I want to thank all my interns for all their hard

work, and enriching my life. They include: Aaron Tan, Adam Veres, Afshin Mostaghim, Allen Dalisay, Allen Partono, Andrew Gen Kuong, Ayeshan Senadeera, Calvin Bui, Calvin Deng, Casey Hill, Chandler Morehouse, Chip Pearson, Chirawat Chriss Sanpakit, Daniel Laurence Chang, Devin Finkel, Hashim Chaudry, Hayden Karich, Humberto Scott Sanchez, Ivan Sotelo, Joey Milia, John Pham, Josh Ice, Kevin Pham, Krishna Addanki, Lindsey Anderson, Madeline Mullen, Manoel Tamraz, Matthew Smith, Michael Uy, Nathan Berue, Norman Li, Palak Parwal, Paul Ehlinger, Raymond Lu, Raymond Wan, Razn Abu Qamar, Ryan Milton, Ryota Saito, Stefany Cruz, Timothy Williams, Tyler Hunting, Vishak Kumar, Xiqiao Zhang, and Zareh Hairapetian. This list does not include many of my Outreach students, whose efforts I also appreciated.

I also appreciate all the financial support that the TEC Center and I have received which include: the Instrument Development for Biological Research (IDBR) program of the National Science Foundation under award DBI-1353974; Prof. Jerome Schultz and the Center for Bioengineering Research at UC Riverside; the Video Bioinformatics IGERT Grant DGE 0903667 from the National Science Foundation; the Innovative Use of Information Technology program of the UC Riverside Faculty Senate; the Sophomore STEM Success program of the UC Riverside Office of Undergraduate Education; the HSI STEM Pathway Summer Bridge to Research Internship Program at UC Riverside; and personal support from Dr. Grover.

To my loving Parents for all the support over the years, and my wonderful wife for
putting up with all my crazy dreams.

ABSTRACT OF THE DISSERTATION

Evolving a Tool Set for Biological Research and Education

by

Douglas A. Hill

Doctor of Philosophy, Graduate Program in Bioengineering
University of California, Riverside, December 2016
Dr. William H. Grover, Chairperson

I returned to school in hopes of being able to develop an instrument that could monitor a large number of biomarkers that, over the course of time, would do early disease detection. Unlike electronics, I found bioengineering lacked a standardized tool set from which to work from. Thus, I began developing a new tool set. Here, I present the foundation of this set of components, a system called Multifluidic Evolutionary Components (MECs). “Multifluidic” conveys the wide range of fluid volumes MECs operate upon (from nanoliters to milliliters). “Multi” also reflects the multi-disciplinary nature of the system (not only fluidics but also electronics, optics, and mechanics). “Evolutionary” refers to the design principles that guided the component development, enabling the library of MEC parts to easily grow and adapt to a wide range of applications. Many of the MEC “building blocks” perform fundamental functions that are commonly found in biological or chemical instruments; functions such as: valving, pumping, mixing, controlling, and sensing. Along with the physical hardware, an equally powerful and flexible piece of software was developed for the control of these instruments. The development of this multidisciplinary tool

set was challenging, so I recruited interns from all areas of engineering, opening the TEC (Technology Evolution & Component) Center. The TEC Center evolved into a multidisciplinary “learn by doing” environment, where real world problems were addressed. The Center is product based. Students enter the program lacking many necessary skills, so training became an integral part of our Center.

For each MEC component a unique symbol is linked to a physical definition. When developing an instrument, a student begins by creating a MEC multidisciplinary schematic, and then uses this to efficiently fabricate a system. As proof-of-concept, we use MEC methodology to build a variety of multidisciplinary schematics, components and instruments, which are discussed in this work. One example; the development of an analytical bioreactor, could be considered a model for implementing STEM education. We look not just at the final instrument but how it evolved, since it is the product development/*Evolution* of these instruments where the critical lessons were to be learned.

Contents

List of Figures	xiv
1 Introduction	1
2 MECs: “Building Blocks” for Creating Biological and Chemical Instruments	23
2.1 Abstract	23
2.2 Introduction	25
2.3 Methods	28
2.3.1 “Evolutionary” design principles	28
2.3.2 Open standard for MEC interfaces	30
2.3.3 Fabrication of MECs	32
2.3.4 MacroMECs	33
2.3.5 MicroMECs	38
2.3.6 Integrated components.	39
2.4 Results	39
2.4.1 Fluidic router	41
2.4.2 MEC-based titration instrument	44
2.4.3 MEC-based cell culture instrument	46
2.5 Discussion	47
2.6 Acknowledgments	50
2.7 Supporting Information	50
3 MEC Characteristics	52
3.1 Characteristics of evolutionary components	52
3.1.1 Multi-scale/Hierarchical packaging	54
3.1.2 Symbolic/code representation guiding an efficient process	54
3.1.3 Standardized packaging	56
3.1.4 Standardized interface spacing	57
3.1.5 Fundamental elements	58
3.1.6 Interdisciplinary characteristics	59
3.1.7 Mechanical 3D structure critical to function	60

3.1.8	Standardized three-dimensional interfaces	61
3.1.9	Component-to-component compatibility	62
4	Early MEC	64
5	Developing an early MEC scripting language and quantifying fluorescence with a microscope	74
5.1	Summary of experimental method	77
5.2	Experimental results	79
5.3	Methods	82
5.3.1	Fluorescent Pin	82
5.4	Experimental Scripting Language	90
5.5	Conclusion for MEC scripting language, and quantifying fluorescence	94
6	How to fabricate a microfluidic device starting with symbolic representation	96
6.1	Start by creating a symbolic table, and from that a schematic	98
6.2	Translating a schematic to layout	102
6.3	Translating layout to Gerber files	105
6.4	Translating Gerber files to mold plate	105
7	Identifying substances toxic to stem cell colonies by measuring multiple factors	110
7.1	Growth (change in area) is insufficient for defining health state of a colony.	113
7.2	Using growth (change in area), halo (apoptosis), and health ratio (fingers) to study the health state of the two controls	118
7.3	Smart edge detection	122
7.4	Conclusion of identifying substances toxic to stem cell colonies by measuring multiple factors	124
8	3D visualization of biomarkers	126
8.1	A first data set: my blood tests	128
8.2	Parsing Data	129
8.3	Graphing medical data to look for trends	130
8.4	Developing a 3D visualization method for simultaneous visualization of a wide range of biomarkers	132
9	MEC Compressor	138
9.1	C++ to Python	141
9.2	Operation of the Compressor	141
9.3	Discussion of the Compressor and its components	143
9.4	Developing the one-way valve	147
9.5	The next generation compressor	149
9.6	The compressor could be a “Smart Instrument”	149

10 Electrical Valve	151
10.1 An electrically controlled valve is a needed fundamental element in the MEC system	151
10.1.1 Existing electrically-controlled microfluidic valves	155
10.1.2 Designs of MEC electrically-controlled microfluidic valves	158
10.2 Early electrically controlled valve device	160
10.3 Solenoid Valve, H-Bridge and Testing	163
11 Solenoid-valve-operated MEC Pump	182
11.1 MEC schematic defines the system and its assembly	184
11.2 Operation of MEC pump	185
12 Coulter System	187
12.1 Coulter Device	188
12.2 Coulter 1 system	200
12.3 Coulter 2 system	207
13 Bioreactor Development	216
13.1 Early Bioreactor	216
13.2 Bioreactor System	221
13.2.1 Bioreactor Computer	227
13.2.2 Bioreactor Media Dispensing	228
13.2.3 Serial Dispensing	230
13.2.4 Heating system test board	233
14 Accelerometer Weight Scale	236
15 Density Meter	241
16 I2C bus device test board	245
17 Software	249
17.1 Computer side structure (Python)	251
17.1.1 Configuration	251
17.1.2 Menu	252
17.1.3 Macro	253
17.1.4 User main thread	253
17.2 The code structure in C++ developed for the microcontrollers of the instrument.	254
17.2.1 Initialization	255
17.2.2 Service	255
17.2.3 Access	256
17.2.4 Additional critical functions	256
17.3 Automatic code generation	257
17.3.1 Instance of a system	258
17.3.2 Only the code the user needs is generated	258

17.3.3	User main thread	259
17.3.4	Walking through the steps in automatic code generation to create software for the Test Board.	260
17.3.5	Creating an instance of software for the Test Board	265
18	Intern Program	277
18.1	Douglas Hill	278
18.2	Adam Veres	280
18.3	Chandler Morehouse	281
18.4	Hashim Chaudhry	282
18.5	Hayden Karich	283
18.6	Krishna Addanki	285
18.7	Manoel Tamraz	286
18.8	Matthew Smith	287
18.9	Raymond Iu	288
18.10	Raymond Wan	289
18.11	Razn Abu Qamar	290
18.12	Ryan Milton	292
18.13	Stefany Cruz	293
18.14	Timothy Williams	294
18.15	Vishak Kumar	295
18.16	Zareh Hairapetian	297
19	Intern Program Alumni	299
19.1	Afshin Mostaghim	300
19.2	Casey Hill	301
19.3	Nels “Chip” Pearson	303
19.4	Daniel Chang	304
19.5	Humberto Scott Sanchez	305
19.6	Ivan Soletto	307
19.7	Kevin Pham	308
19.8	Norman Li	310
19.9	Ryota Saito	311
20	The TEC Center	313
20.1	The problems the TEC Center is addressing, and the partners that help define them	313
20.2	The Multifluidic Evolutionary Component (MEC) system is uniquely qualified to address biological problems.	315
20.3	Real world problems engage students and can create focus for a Makerspace	316
20.4	Grant funding	317
20.5	Fully engaged students have no limit to what they can accomplish	318
20.6	Students’ time is a key problem	320
20.7	Teacher training	321
20.8	Experts	322

20.9 The Multifluidic Evolutionary Component (MEC) Inventory system	324
20.10 Students must have a basic set of tools and equipment with which to work	325
20.11 Makerspace environment and costs	325
20.12 Website housing the knowledge	328
20.12.1 Real-world problems	328
20.12.2 Tutorials	328
20.12.3 People	329
20.12.4 Results	329
20.12.5 MEC System	330
20.12.6 WebLab	331
20.12.7 Library of Inventions	332
21 Outreach	333
21.1 Outreach teaching leadership	335
21.2 Students teaching students was a constant theme in the Outreach program.	336
21.3 The design will talk, if you listen	336
21.4 Marble Machine inventory	338
21.5 Persistence is a key factor in a student's success.	340
22 Practioners	346
22.1 Silt damaging the ocean environment	348
22.2 Weather conditions endangering lives	349
22.3 Chagas disease is killing people and lowering the quality of living	350
22.4 Potential extinction of the Panamanian golden frog	352
22.5 Health problems due to smoke inhalation from wood-burning stoves in homes	353
22.6 Poor health due to poor nutrition	353
22.7 Sickness due to poor water quality	354
22.8 Better local food source	355
22.9 Better education	356
22.10 Indigenous population needs deed to land	357
22.11 Retention of local population and culture	357
22.12 The ill effects of jungle deforestation on plants, animals, and ocean life. . .	358
22.13 Better housing conditions	359
22.14 Conclusion to Panama problems	360
23 Conclusions	361
Bibliography	364

List of Figures

1.1	A system I developed for the Navy, demonstrating electronic packaging and grey scaling	3
1.2	Systems built with my Snap Circuit blocks.	4
2.1	Overview of designing and building a custom instrument using the Multifluidic Evolutionary Component (MEC) system. The instrument is designed by arranging MEC symbols on a schematic, then the schematic is used to guide the assembly of MEC “building blocks” into the finished instrument. For many applications involving manipulating fluids on the milliliter scale, the “off the shelf” macroMEC components (like the manual valve shown) are adequate for building complete instruments. For instruments that must manipulate fluids on the microliter or nanoliter scale, microMECs can be arranged into a micro-schematic (like the mixer chip shown), fabricated using conventional microfabrication techniques, packaged in a MEC shell, and plugged into the rest of the instrument.	27
2.2	Design standard for the MEC system. Standardized sizing and spacing allows MECs to be connected together in all three spatial dimensions, and interface points (circles) can be designed to allow fluid, electricity, or light to flow between MECs.	29
2.3	Selected Multifluidic Evolutionary Components (MECs), including macroMECs that can be “clicked together” (A–D) and microMECs that are fabricated into microfluidic integrated components (E).	34
2.4	Two <i>integrated components</i> , microfluidic chips containing several microMECs combined together and fabricated using soft lithography of PDMS (A) and wet etching of glass (B). The integrated components are packaged in a MEC-standard shell that plugs into the fluidic socket macroMECs (SA1-10 or SB1-20 in Fig 2.3A). By packaging a microfluidic chip in a MEC shell in this manner, virtually any microfluidic chip can be integrated into the MEC system.	40

2.5	Three sample instruments built using the MEC system. (A) A MEC-based multiscale fluidic router designed to mix fluids from nine different fluid storage MECs in volumes spanning five orders of magnitude (from 400 nanoliters to 10 milliliters). (B) A MEC-based acid-base titration instrument suitable for educational applications in classrooms. By toggling the manual valves SU1, SU2, and SV1, different ratios of water : sodium hydroxide solution : phenolphthalein are delivered to a microcentrifuge vial. The observed range of phenolphthalein colors (from clear in pure water to pink in 4.0×10^{-5} mass concentration NaOH) confirms the successful operation of the instrument. (C) A MEC-based bioreactor capable of culturing cells. The optical density macroMEC (OD1 in Fig 2.3B) measures the concentration (via optical density) of the cells as they grow and react to stimuli. (D) A growth curve obtained by using the bioreactor in C to culture <i>Saccharomyces cerevisiae</i> yeast cells. The bioreactor is sensitive enough to detect important metabolic checkpoints in the growing yeast, including the diauxic shift (when the yeast cells switch from glycolysis to the aerobic oxidation of ethanol) and the entry into G_0 (when the yeast cells exhaust all nutrients and enter stationary phase).	42
3.1	A key inspiration for the packaging methodology of our multifluidic tool set is the integrated circuit. The silicon microchip is surrounded by a shell. The shell holds the standardized spaced pins that facilitate connection of this part to other components	56
3.2	Spatial design standard for the MEC system. Standardized spacing allows MECs to be connected together in all three spatial dimensions, and interface points (circles) allow fluid, electricity, light, etc. to flow between MECs. . .	61
4.1	Photograph of the front side (A) of Snap mock-up of a MEC mixing system, with storage, valves, and mixing chamber. Backside (B) of the same mock-up, which shows the microcontroller-based control interface.	64
4.2	a) Air chamber made from a pill bottle. b) The PDMS casting mold that was used to make it mechanically compatible to the Snap MEC standard. .	65
4.3	a) Air chamber test board. The test board includes a mechanical screw valve, syringe air pump, mechanical baseboard, and air chamber. b) Simple bubble test of the air chamber. Bubbles emerging indicate even the smallest of leaks.	66
4.4	First functioning MEC mechanical screw valve, built from Snap blocks . . .	68
4.5	The power plug for interfacing electrical power from wall adapter to the MEC system. It includes a switch, adapter plug, and “power on” LED indicator .	69
4.6	(a) Layers of a future MEC baseboard, which includes fluidic and electrical layers. It also includes vacuum and pressure chamber planes. (b) Cross section closeup of baseboard, showing component interfaces.	70
4.7	Fluidic schematic for mock-up MEC mixing system	72
5.1	Data from the Flexstation comparing concentration of UBC9-YPET to intensity in Relative Fluorescent Units (RFU) at 530nm wavelength.	79

5.2	Data from the Nikon Eclipse Inverted Microscope comparing concentration of UBC9-YPET to inverted exposure time (1/ms) using the FITC filter. . .	80
5.3	Graphical comparison between inverse exposure from the Nikon and Intensity from Flexstation.	81
5.4	Image of fluorescent pin shown at top. Fluorescent pins are placed in upper left corner and lower right corner of wells of interest.	82
5.5	This shows the translation of the two-point XML file to the 18-point XML file which corresponds to the wells of interest.	85
5.6	Basic set up of the Nikon inverted microscope. The excitation filter determines range of light to hit the sample. The emission filter determines the range of light measured from the sample.	86
5.7	Specifications for four filter cubes are shown. “425/26” means the excitation wavelength ranges from 412 to 438 nm. The diagram was supplied by Semrock and shows the optical density for excitation (FF01-425/26-25) and emission (FF01-475/28-25).	87
5.8	The methodology for finding the effective exposure at constant intensity of 125. For Well A1 the $E_{125} = 5.1$ ms.	88
5.9	The Imaging and Flexstation configuration scripts. These scripts define the layout of the wells. In addition, they defines the filters or wavelengths that will be used. For imaging, the script also defines the exposures.	90
5.10	The first function of the script is to translate the raw data from the device to the data array. Above is shown the two data types the scripting language has been designed to work with, which is images and text.	92
5.11	Plot Script was used to translate data from the data array to a graph. First the script defines how the data will be filter, and produces a plot array. . .	93
6.1	Set of custom symbols made with Eagle software for producing a microfluidic device. The fluidic symbols are directly linked to physical characteristics that are then produced on the artwork. <i>Table Components:</i> IO Fluid is an input and output connection to the outside world. They are also being used as pressure inputs. The IO controls are vacuum connections that are designed for controlling valves. The Mixer is designed for mixing material. It is made of two chambers separated by a membrane. Fluids are added into the upper chamber. A pulsed vacuum or pressure is applied to the lower chamber, causing the membrane to flex and mix the material. The Bleed is a component that is designed to bleed off the vacuum once it is turned off, allowing the valve to close. The Storage is a component designed to store a set amount of a liquid compound prior to mixing. The valve is a component designed for turning off and on a fluid channel.	98
6.2	Eagle schematic showing the microfluidic mixing system. Custom sysmbols were developed for microfluidic components	99
6.3	Desktop printed circuit board (PCB) prototype machine. The machine can generate a PCB from Gerber files which have been translated to CAM files.	100
6.4	The microfluidic microcontroller was made from a tabletop milling machine	101
6.5	“Rat’s nest” physical layout is generated from the symbolic schematic. . .	102

6.6	Zoomed-in view of rat's nest layout shown above	104
6.7	Image of Gerber files displayed in ViewMate. They were generated by Eagle from the layout, which was created from the schematic. The drill master shows where inner layer holes and fixture holes are located.	105
6.8	Printed circuit board (PCB) PDMS-cast mold. The traces form the channels. The vias are made with stainless steel pins passing through drilled holes in the PCB.	107
6.9	Artwork PDMS-cast mold. The thickness of the ink determines the height of the microfluidic channel (a). Leaky cast mold after it has been sealed with tape, with extra PDMS well on top (b).	108
7.1	The health index concept: learn what is the normal state for several biomarkers, and when a person's biomarkers moves out of normal range, flag an error.	111
7.2	Not actual test image, only showing placement of colonies. Toxicity testing of human induced pluripotent stem cells (Riv33). Row 1 used artificial cinnamon e-juice, and Row 3 used cinnamon e-juice. Each row had two controls and used 0.01% and 0.1% mixture of e-juice and media. E-juice is the liquid used in electronic cigarettes.	112
7.3	Images from the first and last frames of two different IPSC colonies. The reduction in colony area is insufficient to show which colony had a higher effect from toxins. The starting health state of the 0.1% colony was lower, showing we needed a better way of determining the start state of a colony.	113
7.4	Micrograph of a dying induced pluripotent stem cell colony. Notice the region of higher white pixels where cells are dying. I interpreted a lack of fingers as a sign of an unhealthy colony.	115
7.5	Parameters I measured with my MATLAB software by the end of IGERT class included colony growth, halo ratio (relates to light reflection as cell dies), health ratio (relates to number of fingers), mobility, growth rate, maximum distance rate, and maximum distance traveled	117
7.6	Twelve frames acquired of IPSC control 1. The first frame was acquired at 5 hours, and then a frame was acquired every four hours, up to twelve frames. Also shown are three graphs from the analysis of these images: growth (change in area), halo (apoptosis), and health ratio (fingers).	118
7.7	Twelve frames of IPSC control 2. First frame was acquired at 5 hours, and then a frame was acquired every four hours, up to twelve frames. Also shown are three graphs from the analysis of these images: growth (change in area), halo (apoptosis), and health ratio (fingers).	120
7.8	(a) Colony extension that the boundary algorithm missed. (b) Bay in colony that boundary algorithm missed. (c) Loose edge that boundary algorithm formed. (d) Tightening of edge with smart boundary algorithm.	122
7.9	Smart boundary detection	123
8.1	This is an example of a section of blood test that was provided in PDF form	129
8.2	Command line menu, showing the current biomarkers that are supported by this software	130

8.3	Example of a graph that was generated by this software, showing red blood cell and white blood cell counts over time. The left axis is a normalized range from 0 to 1. The zero represents the minimum value for the red or white blood cell count. The actual values for these minimums are shown in the legend. The value 1 represents the maximum values for the red and white blood cell count, and again the legend shows the actual values	132
8.4	Early 3D visualization map of biomarkers	135
9.1	This MEC multidisciplinary schematic symbolically defines the compressor and vacuum system. The blue lines are the air lines, the red lines are the electrical connections, and the black lines are the mechanical components. .	140
9.2	This photograph shows the top view (a) and bottom view (b) of the compressor and vacuum system. On the bottom are air lines held down by Snap blocks.	143
9.3	Included in the compressor and vacuum system are a daughter card (a), pressure and vacuum gauge (b), quad air joint (c), electrical pressure sensor (d), air T joint (e), Power connector and switch (f), one way valve (g), air chamber (h), motor, cam, and piston (i).	145
9.4	For every part at the TEC Center that works, there is often a junkyard of parts that did not. Most of these one-way valves failed because we could not get a good back seal.	147
10.1	a) Example of a Medusa-head lab-on-a-chip microfluidic device [63]. The device has a high level of integration. b) Simple colorimetric indicators for paper-based microfluidic devices [21].	151
10.2	Standard microfluidic cross valve is small. Small solenoid valve required for turning off and on the microfluidic valve is relatively large (i.e. at a different size scale).	153
10.3	An existing electrically-controlled microfluidic valve. FC40 and FC70 are temperature-sensitive fluids can be used as actuation fluids in the design of a thermal valve [22].	155
10.4	An electro-active polymer sheet, with electrodes plated on both sides. The sheet flexes when 1 to 10 volts is applied [29].	156

10.5	A hydrogel linkage collapses at a thermal threshold, opening the valve. FC70 has a high thermal expansion coefficient, closing the valve when heated (A). An electro-active polymer flexes when a voltage is placed across the membrane. I designed a family of valves with this material in mind. The first EAP valve (B) uses positive feedback, so when closed, pressure builds up on the inlet, further holding the valve closed. The EAP valve can be designed to be normally open (C), and normally closed (D). Valve design needs to be adjusted whether one is working with a line under pressure or vacuum. I designed a vacuum valve (E), with positive feedback. When the valve is open, the feedback helps hold it open. The high speed valve does not have an opening and closing valve. Instead, it uses osmotic flow to adjust the flow of fluid down a channel, so that a particle flowing down a channel will either go left or right.	158
10.6	An early MEC component concept design. In this component, multiple electrically-controlled valves integrate into one device. There is an on-board microcontroller which receives its instruction through serial communications, and an optical window through which samples can be analyzed.	160
10.7	The printed circuit board electrical layers for the 16 well storage module. Surface-mount thermal resistors are on layer 1. Surface-mount transistors are on layer 4. PC layer 3 is a ground plane and not shown. The lower printed circuit board and its layers are not shown.	161
10.8	Microfluidic layer designs for the 16-well module . The hydrogel requires a fluid inlet line to allow for hydrogel swelling, which closes the channel. . . .	162
10.9	Electromagnet made from a nail, a coil of wire, and a 3D-printed base. It was intended to apply a magnetic force to another rod which was to close a valve. The rod it pulled on was external to the electromagnet. This electromagnet's magnetic force was far too low for valve actuation.	163
10.10	Solenoid valve test station. We greatly increased the number of coils in the solenoid valve. The metal rod pulled by the electromagnet is internal to the coil, and we added a permanent magnet. The magnet is able to contribute its force to opening the valve (required to overcome spring force), reducing the force needed from the coil. Once open, no coil force is required to remain open, so this assembly has a latch state. Reversing the direction of current flow through the coil will release the rod and close the valve.	165
10.11	Test jig to determine spring force required to close Tygon tubing channel .	167
10.12	Solenoid valve test station. This station includes a paper H-bridge to drive current through the coil. This is also a great educational tool for showing how an H-bridge functions.	168
10.13	The assembly of the MEC solenoid valve (g) is made up of several components: a permanent magnet (a), a spring (d), a 3D-printed spacer (e), a pinch rod (c), and connector pins (h). The two wires connected to the connector pins can be seen in the assembled solenoid valve (b). From another view one can see the 22 gauge wire that connects to the outside (f).	169

10.14	Complete solenoid valve test jig. To test the solenoid valve, we first connect the solenoid to the bubble bottle and the air chamber through a mechanical switch. Next, we use a syringe to increase pressure in the air chamber up to 20 PSI, which will be indicated on the gauge. We then turn on the mechanical switch. If no bubbles appear, we pass our first leak test. Next, we use the paper H-bridge to apply current to the coil, opening the valve. The valve latches on, and air flow is evident from the stream of bubbles. Finally, we reverse the H-bridge, closing the valve.	173
10.15	As the solenoid valves evolved so did the test jigs. The electrical H-bridge breadboard used to test the solenoid valves (a). We used this design to create a printed circuit board H-bridge version that we used to test solenoid valves (b). We also developed a pulsing solenoid valve (non latching). With this valve, an H-bridge is not required. A single field effect transistor (FET) can be used. FET-based pulse solenoid valve test jig (c).	175
10.16	A MEC schematic showing the configuration of a solenoid test board that would allow for repeated testing of the solenoid with proper coding of the microcontroller.	178
11.1	Frontside of electronically controlled pump (a). Backside of electronically controlled pump (b).	182
11.2	MEC schematic of an electrically-controlled dual-pump system. The pump is designed to be controlled by a laptop through an Arduino interface. It has two on-board chambers for pressure and vacuum. Each chamber has a manual gauge and electrical monitoring of pressure and vacuum, respectively. Both chambers have electrically-controlled external feeds, so that outside laboratory air pressure and vacuum supplies can be used for maintaining chambers. Pressure and vacuum have electrically controlled sources to the pump piston, drawing fluid in and out of the pump chamber. The flow direction is set by the one-way valves.	184
12.1	First Coulter sensor with 3D-printed base, PDMS-cast channels, and 20-gauge stainless steel IO pins (a). The sensing aperture is shown magnified in figure (b). The dark parts are 20 gauge stainless steel pins with a diameter of 0.036 in. The clear tube is the PDMS channel. Figure (c) and (d) show the top and side view of the 3D-printed body and cavity. 3D printed with the UC Riverside Bioengineering Department ABS printer.	188
12.2	As a small object passes through the aperture, the Coulter sensor is able to measure the size of the object by measuring the change in voltage across the stainless steel pins.	189
12.3	3D printed Coulter sensors with different aperture sizes. The apertures sizes are: 1/1000 in, 2/1000 in, 4/1000 in, 6/1000 in, 8/1000 in, 10/1000 in, and 23/1000 in. We were able to get useful apertures with sizes greater than four thousands of an inch.	190

12.4	Measured flow rates confirm changes in different aperture sizes. These tests were done on the MEC scaffolding system. We used the apertures shown in 12.3 and recorded the amount of time while fluid passed through aperture, then weighed the fluid that passed through the aperture.	192
12.5	The MEC scaffolding system required some new components to be developed. a) shows the 10 milliliter syringe holders. Some of the syringes have been adapted to allow pressurized air to be applied at the top. The screws shown in b) and c) are used to lock the scaffolding into place. b) shows how platforms can be locked in place with the scaffolding system. d) shows the base support for a scaffolding rod.	192
12.6	MEC platform used for testing Coulter sensors	193
12.7	Setup for passing bubbles through the Coulter sensor, which is used to test its operation. Pressure is applied to water in an external bottle, driving the fluid first through a T joint and then through the Coulter sensor (not shown). Air pressure is then applied to a dual switch. When the switch is pushed forward, air passes to a small loop in front of the switch. When the switch is pulled back, the air feed line is turned off and a small pulse of air is sent to the T joint. This will then pass through the Coulter sensor, resulting in an electrical resistance change while the pulse is passing through the sensor.	195
12.8	Length of aperture path and diameter of aperture are critical to the sensitivity of the Coulter sensor. Components of the next MEC generation of Coulter sensors include: 3D printed base, two O-rings, clear film with small aperture hole, copper foil on both sides of film, two stainless steel pins, conductive rubber, and two small wire connectors (a). Assembled Coulter sensor (b). This sensor was never tested.	197
12.9	Unit fluid pulse component. Two sets of lines are used for loading the pulse. The other two lines are used to send the unit pulse down the line through the sensor with air.	198
12.10	MEC Coulter 1 schematic. Blue lines represent fluid lines. Red lines represent electrical connections. C1 and C2 are the Coulter sensors. The signal line is connected between the two Coulter sensors to an analog-to-digital pin on the Arduino.	200
12.11	Frontside and backside of Coulter system. Alligator clips are used to make electrical connections to the stainless steel pins on the Coulter sensor. A Snap quad-rocker switch is used to send a four-bit digital signal to the daughter card to set the pulse width time of the sample. The pulse width time is used to determine how long the solenoid valve is open. On the backside one can see two air tanks and an electrical pressure sensor.	203
12.12	Some of the key components in the Coulter 1 system include: daughter card (a), MEC solenoid made with a Dimension 3D printer (b), and Snap quad rocker electrical switch.	204
12.13	Signal read by the Arduino without (a) and with (b) bubbles in the sample pulse of fluid.	205

12.14	Early electrical measurements of fluid samples with NaCl in solution (a and b).	206
12.15	The MEC-based Coulter 2 Instrument	207
12.16	Bottom view of Coulter 2	208
12.17	Closeup of the on-board mini-computer, with daughter card plugged in over the Arduino.	209
12.18	a) a 2x2 MEC Coulter sensor with half-grid fluidic connection. b) a 3x2 MEC Coulter sensor with on-grid fluidic connection.	210
12.19	a) Breadboard of a dual FET pulse solenoid driver. b) Press piston for flow regulator. c) Vertical bead storage unit. d) Female T joints. e) Electrical pressure sensor. f) Electrical latch, for latch digital signals.	211
12.20	Two design revisions of the air chamber.	214
13.1	First model of bioreactor utilized a pump for fluid flow	217
13.2	Improving the bioreactor joint MEC by switching from the Bioengineering Department's FDM printer (a) to an external 3D printing house (b).	218
13.3	The first MEC bioreactor.	219
13.4	A growth curve obtained by using the bioreactor to culture <i>Saccharomyces cerevisiae</i> yeast cells. The bioreactor is sensitive enough to detect important metabolic checkpoints in the growing yeast, including the diauxic shift (when the yeast cells switch from glycolysis to the aerobic oxidation of ethanol) and the entry into G ₀ (when the yeast cells exhaust all nutrients and enter the stationary phase).	220
13.5	MEC Visio drawing of Bioreactor. It contains two blow-ups, one of the Serial Fluid Dispenser and one of the H-Bridge Driver	221
13.6	Front view of bioreactor.	223
13.7	Back view of bioreactor.	224
13.8	Bottom view of bioreactor	226
13.9	The bioreactor computer's sixteen-button key pad and two-line, sixteen-character display.	227
13.10	Bottle hold magnet in a Snap base (A). Metal strip on bottle (B). 250ml Media bottle magnetically attached to the bioreactor. Feed line from top of bottle runs to connector on back of bioreactor. Pressure line on top of bottle runs to pressure source under bioreactor (C). Two lines go into bottle. Feed line connects to long tube that goes to bottom of bottle. Pressure line goes to top of bottle cap and is pressuring the air over the fluid, driving the fluid out the feed line.	228
13.11	Schematic of a MEC serial dispenser, including eight solenoid valves, four sample dispenser tanks, one flush water tank, a trace optical sensor, I2C bus in red, and four H-bridge modules.	230
13.12	Frontside of serial dispenser (A). Backside of serial dispenser (B).	232
13.13	Front-side of heating test board (A). Back-side of heating test board (B). Channel thermal resistor component (C). Heating coil component. The heating coil component plugs in the end of the L channel joint (D).	233

14.1	Bottom of accelerometer weight scale instrument, showing the accelerometer, vibrator, FET, decoupling capacitor and ATtiny84 microcontroller (A). Top of accelerometer weight scale showing bottle, bottle holder with embedded accelerometer, Arduino, and power interface (B).	237
14.2	Graphs of Fourier-transformed X axis data for the accelerometer. From left to right, the graphs correspond to bottles that are contain 0, 50,100, 150, and 200 mL of water.	238
15.1	MEC multidisciplinary schematic of a MEC density meter. Red are the electrical connections. Blue are the fluidic channels. Black are the mechanical aspects of the design (A). 3D printed U tube holder block (B). Density meter optical sensor holder block (C).	241
15.2	Density Meter. Shown in the figure is the electromagnet (coil) for bending the stainless steel U channel; Clamp with 2-56 screws for clamping 20 gauge U tube; Optical sensor for determining position of U-tube and then sending signal to turn off and on the coil ; Electronics on electronic breadboard for receiving signal from optical sensor and then turning on and off the coil.; Mechanical MEC baseboard for mounting components.; Spring terminal for ease of wiring of electronics	243
16.1	I2C bus device test board. Devices for which software has been written, and have been made I2C compatible, include: light emitting diode (LED), electrical switch, dome button, 4x4 key pad, 2x16 character display, solenoid valve, and potentiometer.	245
17.1	MEC device test board containing two ATtiny 84 microcontrollers (AVR1 and AVR2). The devices on AVR 1 include a button, two LEDs, and a potentiometer. The devices on AVR2 include a marble sensor, a green LED, and a muscle-wire valve.	260
17.2	Netlist for AVR1 (A) and AVR2 (B). A netlist defines the pin-out interconnection between devices. AVR 1 has devices defined in figure 17.1. AVR 2 has an IR emitter, IR receiver, green LED, and output.	261
17.3	Double left click “Start_Create_or_Update.py” to start Auto generation software (A). Select 1 to create a new system (B). Type the name of the new system (C). Confirm that name is correct and we will not be using a pre-existing system (D). Input the number of AVRs in the system (E). A Button is selected as the first device in AVR1 (F).	262
17.4	Menu options for selecting devices. Next device in AVR 1 is an LED, and we put it on pin 8 (A). Next device is an Analog to Digital Converter (ADC), connected on pin 10 (B). Finally , we place a green LED on pin 13 (C). We then use option 0 to quit this menu (D).	263

17.5	Set of menu options used for selecting devices for AVR 2. First, we switch to AVR 2 (A). Option 3 is for “adding device” (B). We select LED and place it on pin 2 (C). We select another LED and place it on pin 5 (D). We then select an ADC and place it on pin 6 (E). We are now ready to quit device selection so we select option 0 (F).	264
17.6	Set of menu options for selecting features for AVR 2. Option 5 is used for selecting features (A). The first feature we select is an “Output” (B). We put the Output on pin 8 (C). We are done adding features so we quit the menu (D). We are done adding devices and features to both AVRs so we quit this menu (E). We are done creating new systems, so we select one final quit (F).	265
17.7	Key tabs, windows, and entry boxes used when programming ATtiny84 microcontroller with Atmel Studio. The user enters the project name and selects “GCC C Executable Project” (A). This is the window used for adding pre-existing code to project (B). Automatically-generated code is in the “Test Board” folder under the “mec_System” folder. Auto generated code for “Test Board” AVR1 (D). The window for building one’s solution or compiling one’s code (E). The window for “Device Programming” (F).	266
17.8	Code folders for Test_ Board (A). Code and script folders under Python, also including the Start_Run.py file, which is the Test Board instance start program (B). Command line starting menu, which contains Convert and Run which are used to convert and run auto-generated script and code (C). Printout that lets the user know if Python code was able to find MEC Arduino on USB port (D) . User defined Start Menu (E). Menu for toggling LED_2. Entry 1 toggles LED (F). Menu for toggling LED 2, entry 2 exits menu (G). Entry 1 in Start menu selects Button_1 test menu.	268
17.9	Selecting option 1 in button test menu (A). This option reads the state of the button. A True is returned if the button is not pushed (B). We now press on the button and select option 1 again. This time we return a False (C), indicating the button has been pushed. Once finished testing the button, we select option 2 to exit (D).	270
17.10	Selecting option 3 on start menu (A). This option is used to test the potentiometer on the test board. Select option 5 on ADC menu (B). This option is used to read the selected ADC pin on the microcontroller. Here, it is showing that it obtained an ADC reading of 386 (C). Turn the potentiometer and reread the ADC, we got 897 as a reading (D). Option 3 is used to change the threshold (E). Setting threshold to 600 (F).	271
17.11	ADC menu with threshold set to 600. We select option 4 to test whether the analog signal is above or below the threshold (A). Error message indicating it cannot find the device macro specified (B). We select 1 to begin logging data to a text file (C). We then select 2 to start the plot. Several plots may have been taken so a menu will come up, where the user can select which plot he wants to display (D). Moving the potentiometer back and forth, one can shift the value from 0 to 1024. This information is time-stamped and stored in a log file (E). Plot of data from log file (F).	273

18.1 Douglas Hill	278
18.2 Adam Veres	280
18.3 Chandler Morehouse	281
18.4 Hashim Chaudhry	282
18.5 Hayden Karich	283
18.6 Krishna Addanki	285
18.7 Manoel Tamraz	286
18.8 Matthew Smith	287
18.9 Raymond Iu	288
18.10 Raymond Wan	289
18.11 Razn Abu Qamar	290
18.12 Ryan Milton	292
18.13 Stefany Cruz	293
18.14 Timothy Williams	294
18.15 Vishak Kumar	295
18.16 Zareh Hairapetian	297
19.1 Afshin Mostaghim	300
19.2 Casey Hill	301
19.3 Nels “Chip” Pearson	303
19.4 Daniel Chang	304
19.5 Humberto Scott Sanchez	305
19.6 Ivan Soletto	307
19.7 Kevin Pham	308
19.8 Norman Li	310
19.9 Ryota Saito	311
20.1 Analytical bioreactor and MEC schematic	316
20.2 Interns engaged in evolving the MEC system	320
20.3 TEC Center website	327
21.1 SolidWorks layout showing the five groups and the three teams per group. This was one of the first SolidWorks designs done by an Outreach student.	335
21.2 The part drawn in the first Outreach SolidWorks training class.	336
21.3 MEC inventory system under development. This cupboard is dedicated to all the marble track components that are being developed.	338
21.4 Short tutorial on how to make an involute gear in Solidworks, by Manoel Tamraz	339
21.5 Marble transport parts Hayden Karich was evolving (A). Marble transport parts Manoel Tamraz was evolving (B).	340
21.6 MEC schematic of Marble Elevator system driven by differential pressure, drawn by Vishak Kumar. This is the Marble Machine system Hayden’s team is developing.	341
21.7 Marble elevator lift system (A). Marble chain lift system (B). Marble gear lift system (C).	342

21.8	First working marble machine developed in the Outreach program.	343
22.1	Website of Floating Doctors, an organization with which the TEC Center is working.	347

Chapter 1

Introduction

Since I was young I have always loved learning and creating, whether it was building a robotic costume to writing a story. My higher education began with an electrical engineering B.S degree from University of California, Irvine in June 1979. My first job was with Texas Instruments, where I was given the honor of being the only Junior Engineer listed as a “Key Contributor.” At Texas Instruments I began my electronic packaging career. My job was to shrink the size of a power supply by 10 times. I did this by redesigning the magnetics and developing my own family of surface mount components. I still remember when they first turned on my miniature power inductor, which was smaller and far more efficient than anything that had been done before, watching the insulation melt off the load wires as my part stayed relatively cool. After a year, I left TI to start my own business in California, with a short break to sail an 18 foot catamaran from Trinidad to Florida. My system design house was known as Advanced Electronic Packaging Corporation. Its mission was to solve electronic packaging problems, which means fitting a large quantity of

electronics into a small space. I planned on doing this through the use of surface mount components, which at the time were ceramic. Ceramic parts were expensive, leading to very few early customers, but then Texas Instruments came out with the first plastic surface mount component, the 64K X1 DRAM. Being one of few houses that knew how to work with surface mount components, I landed a large contract with Alpha Micro, a computer manufacturer. My business helped lead the transition from “through-hole” technology to surface mounted technology. In the end, I designed for most of the major corporations across the United States such as IBM, Ford, Motorola, Northrop, JPL, Gilbarco, Navy, Woods Hole Oceanographic Institute, and AT&T. My business spanned over 15 years. I was one of the early inventors of memory sticks that can be seen in computers today. As surface mount became more popular, my business transitioned from modules to systems, such as the memory system I developed for the Navy, shown in figure 1.1.

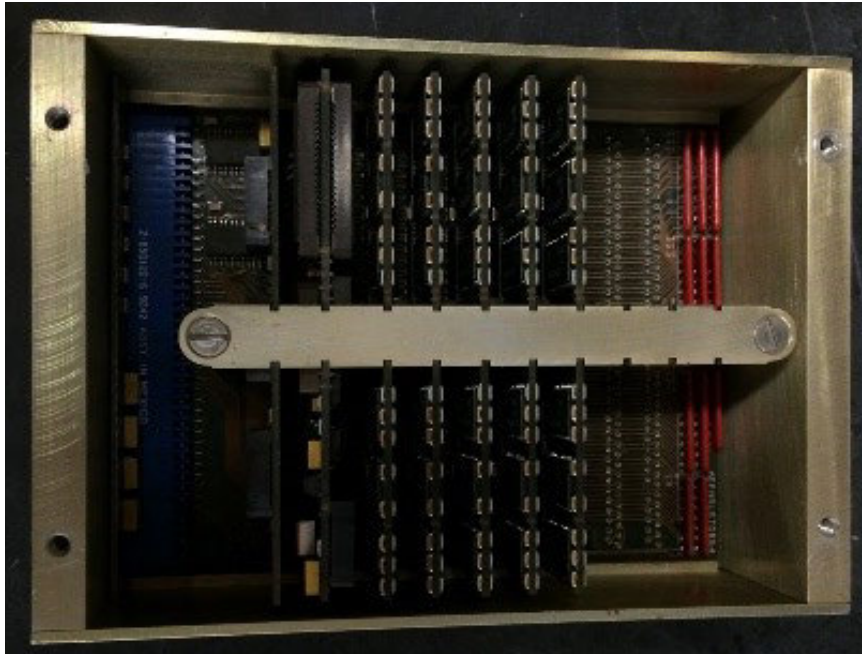


Figure 1.1: A system I developed for the Navy, demonstrating electronic packaging and grey scaling

Even though I was able to be creative through solving my customers' problems, I was not able to choose the problems. I set out to select a problem that I wanted to solve. The challenge I ended up with was the lack of creative problem solving in the educational system. I've never thought small. Since my area was electronic packaging, it seemed only natural to build a set of building blocks for making electronic circuits, one where a student's imagination was their strongest tool.

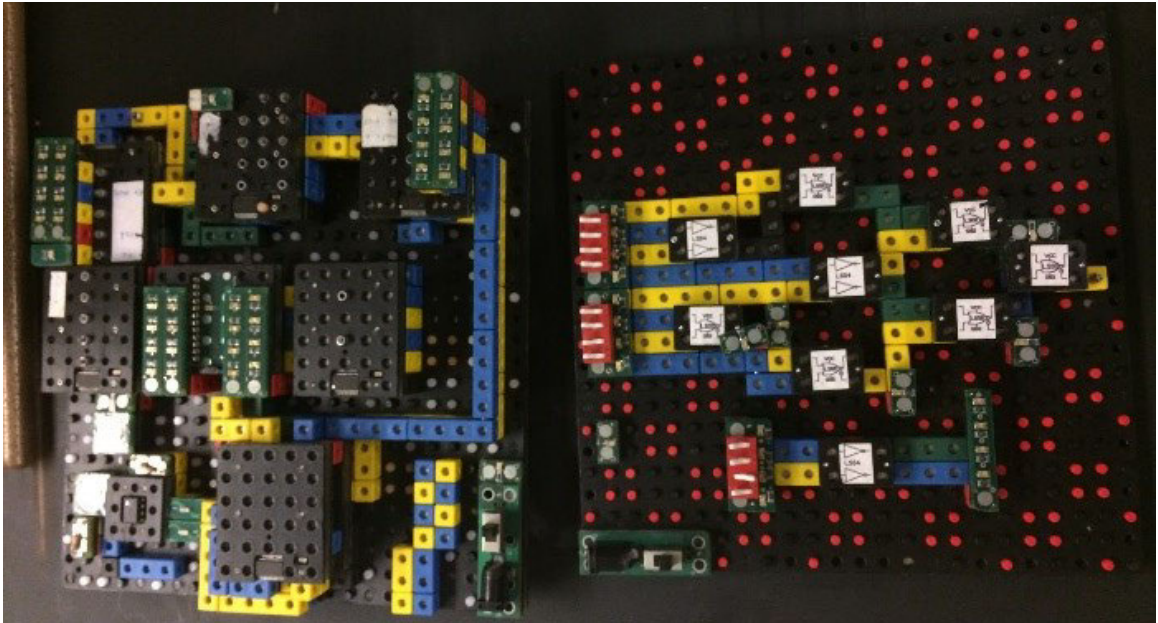


Figure 1.2: Systems built with my Snap Circuit blocks.

Shown in figure 1.2 are examples of systems built with my building blocks, which I named Snap Circuits. I developed several kits for classrooms, renting and selling them. I ran summer programs. Once, in Compton, where I was using Snap Circuits in a summer program, I remember talking to one of the teachers about a young girl in my class who was obviously gifted, only to find out that she had been previously kicked out of school, which was why she was in my program. I remember helping another student build a simple light tower. As I walked away, I watched him tear it down and rebuild it his own way. He was already taking ownership over his design. I had also worked with some college students who, in contrast, would ask “where are the instructions?” The Compton kids were used to being on their own and solving the problems themselves. Here is where I developed my philosophy on why kids from poorer communities have problems. It’s the vision they have.

They do not envision themselves going to college, so the school problems they are given seem like garbage. To the student with vision, the problems may still look like garbage, but they realize this is what is required for them to get to where they want to go. Now, I work to bring real world problems into the classroom (or lab). This provides focus and understanding for the student, and when you have the full energy of the student, he/she will reach the top of their potential.

In my business I was accustomed to a world where I provided engineers with solutions and in return, they provided me with contracts. The educational community was very different; with low budgets and strict guidelines, they often did not have the time or money required for creative problem solving. I reached my goal of bringing creative problem solving into the classroom, but had lost my business focus, which ultimately caused me to close my business.

On one of my last projects, the one shown in figure 1.1, the Navy had needed a very high reliability system. It was to be launched remotely under the ocean. I Developed a method for predicting a failure before it occurred, which I called grey scaling. The system would test and record system parameters every time it was turned on, and over time it learned what was ‘normal’. If it saw a shift out of normal, it alerted the user. When this system was first built it warned us that it had too large a number of flash memory writes, so even though it was returning 100% valid data, it informed us that there was still a problem.

If I could keep a piece of electronics healthy, I felt I could apply the same methodology to humans. One of my ideas was the design of a diagnostic toilet that could determine human health through urine profiling. My idea was that early disease detection for an indi-

vidual could be accomplished through the repeated measuring of a large array of biomarkers and other health factors over time; comparing these values with one's personal norms and the norms from other people within one's health group community. A person's health group community is defined by people who have in common health state backgrounds such as similar age, sex, and ethnicity. As diseases are identified by health professionals, trends are looked for in their biomarkers and other health factors. By measuring biomarkers over time, and during a healthy period, we can learn what a normal health state is for an individual. Therefore when you shift out of your normal health state, we can provide an earlier health alert. We would want to measure as many biomarkers and health factors as economically possible. These biomarkers and health factors will come from a wide range of places. First, we would want to include the tests that a person currently is having done. These might include urine and blood tests. Next, we would want to include the growing number of wearables, which can measure sleep patterns and many athletic activities. Other common devices include digital scales and devices utilized to measure body mass index. Imaging systems are being developed to image food and provide estimates of nutrient and caloric uptake. We also would want to monitor medicine uptake with a digital pill and cream dispenser. Genetic profiling is providing a rich body of predictive information to add to our other biomarkers. Finally, we want to develop new instrumentation for improving individualized body fluid profiling. Body fluids we would be addressing would include urine, blood, sweat, saliva, bone marrow, fecal matter, etc. A critical factor in data collection is the ease in which it can be done. A good example of this would be the diagnostic toilet.

The problem was that I had no background in biology, so I came back to school to learn and develop biosensors.

Entering the graduate bioengineering program at the University of California, Riverside, I found bioengineering and microfluidics fascinating. Since I am a system designer, I looked at not only biosensors but also at the components to support them. What I found surprised me: as large as bioengineering is, it does not have a tool set to work from. It does not have the equivalent of a schematic to develop designs from, as we do with electronics. Thus, I began developing what I call the MEC system (Multifluidic Evolutionary Components). It became a natural extension of the Snap Circuit System I had previously developed in electronics, but included many fluidic components and their symbolic representations.

I began studying the characteristics the MEC system would need to have to achieve my goals. I called these characteristics “evolutionary”. The components would need to be able to easily adapt to a wide range of new applications and seamlessly accommodate new ones. To accomplish this, I incorporated design principles from two existing systems that are already “evolutionary” by my definition: *biological cells* and *electronic components*. The specific qualities of cells and electronics that I attempted to replicate in the MEC system include:

1. multi-scale hierarchical
2. symbolic/code representation
3. symbolic/code guiding efficient processes

4. standardized packaging-shell
5. standardized interface spacing
6. fundamental elements
7. interdisciplinary component set
8. mechanical 3D structure critical to function
9. component-to-component compatibility

The MEC system would need to incorporate multi-scale interdisciplinary components to be able to adapt to a wide range of applications. The one characteristic I thought was critical was standardized spacing, which exists with integrated circuits. However, after studying cells (which do not have standardized spacing) I realized that if one has an efficient method to adapt to different spacing, this may not be as critical as I thought. 3D printing may offer the flexibility that is needed for non standardized IO spacing, at least in the prototype stage. The characteristics of the MEC system are described in better detail in sections 2.3.1 and 3.1.

Microfluidics was one of the first areas I addressed with the component-based MEC system. At the time, I was in Dr. Jiayu Liao's lab. One of his interests is the sumoylation pathway. He was using a Molecular Devices Flexstation to quantify fluorescence of FRET, which he was utilizing to study protein interactions. At the time he was searching for a small molecule that would block the sumoylation process. His lab was looking at tens of thousands of small molecules. It was a very labor intensive process. As with most biological instruments, the Flexstation is a closed system. I envisioned being able to covert a

proven labor-intensive process, like what Dr. Liao's team had developed, into an automated process, with the MEC system. At the time, my funding came from IGERT, which is a program that pertains to the processing of video images. I combined my three areas, and proceeded to build a MEC microfluidic device that could combine proteins and image FRET interactions. I developed an imaging process that could replace the quantification of fluorescence by the Flexstation in order for the process to be integrated into the MEC system. This work is detailed in chapter 5.

The MEC microfluidic device for combining proteins was fabricated using a novel casting process. The process was used for building both the micro and macro MEC components. For the microfluidic MEC components the casting process includes stainless steel pins, which formed the channels for the IO pins and represented an improvement over the more common method of stamping out the IO connection. With printed circuit board design software (Eagle), I created a physical fluidic component layout, which were linked to symbols. The fluidic channels were formed by traces. After translation, the Eagle design formed artwork which was utilized for making a casting mold. This demonstrated the evolutionary characteristic of a symbolic design guiding an efficient process of fabrication. One of the key challenges faced by these components was proper sealing. These early MEC components are described in greater detail in section 4, and the casting process in chapter 6.

Another part of the IGERT program was a class I took with Dr. Prudence Talbot. In this class I was given the opportunity to test my method of early disease detection by measuring a large number of biomarkers and looking for normal patterns. Instead of early

detection of a disease in a person, I measured the effect of a toxin on a stem cell colony. I was able to collect a large number of biomarkers through video imaging analysis. More detailed exploration of this can be found in section 7.

The development of the MEC system moved forward after I joined Dr. Grover's lab. I began recruiting interns to help with the development of the MEC system, and I became aware of the Bioengineering Department's 3D printer. The fluidic routing instrument (Fig. 2.1) was developed to demonstrate the multiscale ability of the MEC system. It incorporated a MEC fluidic socket, allowing the integration of micro and macro MECs. The system included an assortment of macro-components (Fig. 2.3). It was assembled on a mechanical baseboard from the original Snap Circuit system. For fluid control, we moved to 3/32 inch Tygon tubing, lengths of which were interconnected in a breadboard fashion. The mechanical valve, air chamber and baseboard have proven to be valuable components, and they continue to evolve with our current systems. The fluidic router instrument and its components are discussed in sections 2.2, 2.3.4, and 2.4.1.

One of the projects outside the MEC system that my intern, Devin Finkel, and I worked on was how to visualize a large number of biomarkers. What we developed was a 3D visualization system. Each biomarker formed a rib of the 3 dimensional structure. Height and color represented the range from normal to abnormal, with green being in the normal range and deep red being the farthest extreme outside the normal range. Ordering of biomarkers or ribs would be determined by their correlation with each other. For data we utilized my personal blood tests, collected over time from the UCR health center. We were able to graph over thirty biomarkers. What we are hoping is for different diseases to

have distinct topographical shapes and color patterns. For a better understanding of this methodology see chapter 8.

The MEC compressor was my first instrument to integrate electronics with fluidics. We selected it to solve the problem of the very short operation times provided by pressure and vacuum chambers. We needed a continuous source of vacuum and pressure to serve as our fluidic energy source. The system is controlled from a laptop through the USB port connected to an Arduino microprocessor. The Arduino has a daughter card with additional support circuitry. In this case, the daughter card has two FETs which drive the DC motor. The motor drives a wheel, which in turn drives a slider, which drives a piston. The last critical part of this system is the one-way valve, which allows the piston to either pull air to form a vacuum or press air to build up pressure. This system further demonstrates our mechanical valves. The compressor was able to reach pressures of close to 20 PSI, and could achieve a sufficient vacuum level to support our pumps. The one-way valve was one of the more challenging components to develop for the compressor. One of my interns found a vendor which sold the internal part of a one-way valve, allowing us to design a shell around the valve to MEC standards and still achieve the reliability we needed. We are continuing to evolve this system today. It was exciting for me to watch a mechanical engineering student calculate torque and load modeling for proper specification of the motor. The compressor is detailed in chapter 9, and the one-way valve in section 9.4.

Density is a fundamental property of a fluid. In our lab we strive to measure fundamental properties, since knowing these properties is a first step in the characterization and analysis of a material. To construct an instrument for measuring fluid density, we bend

a 20 gauge stainless steel tubing into a “U.” The “U” is mounted into a 3D-printed base, and a magnet is epoxied on the end of the “U” tube. A magnetic coil and the “U” base are mounted on a MEC baseboard. They are mounted such that the coil can pull at the magnet. A photodetector is mounted and used to detect movement of the stainless steel tubing. When the tubing is at rest it blocks the detector signal. A simple feedback circuit is constructed, so that when the photodetector’s light is blocked it turns on the coil, pulling at the magnet. As the “U” tube bends, the signal is no longer blocked, and the coil is turned off. The natural frequency at which this “U” tube vibrates is dependent on the mass of the “U” tube and the fluid within. We can therefore determine the density of the fluid because it is a fixed volume system. We used a National Instruments data acquisition card to measure a range of different fluid densities. The instrument was so sensitive that a light tap on the table could be seen in the readings. We are replacing the National Instruments card with analog-to-digital readings through an AVR microcontroller. For more information on the density meter see chapter 15.

Automation will be central to the success of the MEC system. The key fundamental component that was missing from the MEC system, at this point, was the electrically controlled valve. More effort has been put into the evolution of the electrically controlled valve than any other component. After several iterations, the first functional latching solenoid (Fig. 10.12) utilized a 1/4 inch iron core rod, a spring, and a hand-wrapped coil with hundreds of winds. It utilized plexiglass tubing and 3D printed parts. I developed a paper H-bridge for testing, which would also serve as an excellent classroom demonstration. In conjunction with this we were also evolving an electrical H-bridge. We eventually de-

veloped several types of this valve, which included normally-closed pulsing, normally-open pulsing, and normally-closed latching. The 3D-printed part needed to be fabricated by an outside company due to the tolerances that were required. Several test boards have also been developed for the evaluation of this component. Today, we are investigating a “muscle wire” (shape-memory alloy) electrically controlled valve, which I designed for ease of fabrication and increased reliability. Reliability is essential in the MEC system development. Even this is not the final version of the electrically controlled valve; one needs to look at the fabrication process of the transistor to understand the requirements of a truly powerful fundamental component. For an in-depth look at the evolution and design of the electrical valve see chapter 10.

The pump system was the first design to utilize an electronically controlled solenoid valve. It is designed to be controlled by a laptop through an Arduino interface. It has two on-board chambers for sources of pressure and vacuum. Each chamber has a dial and electrical gauges for monitoring of pressure and vacuum levels. Both chambers have been designed with electrically controlled external feeds so that outside lab air pressure and vacuum can be used for maintaining levels. Pressure and vacuum are electrically controlled and applied to the pump piston, which draws fluid in and out of the pump chamber. The pump uses one way valves so fluid can be transferred in only one direction. The design and operation are further discussed in chapter 11.

Here, I’ll pause for a moment to reflect on the theme that is emerging. We want to solve biological problems, yet we lack a component tool set from which to work from. To that end, we are building interdisciplinary components that work together, and from those

components we are building instruments. The devices we are developing are some of the first one might find in a wet lab. Pumps and mixing systems are good examples of common instruments used in a wet lab. Many labs (including my current lab) actually come with compressed air plumbed in, since it is so fundamental for lab operation. However, we don't want the MEC system to only work in well-equipped labs; we want it to work anywhere. As one studies the compressor and pump, one begins to see the duplication of components, an essential aspect of the MEC system. Unlike almost every other research project, we do not strive to be first. In fact, we search for technologies that have proven themselves and look to replicate their function with the MEC system. Our job is translation, education, and then evolution. People see the MEC system as being excellent in resource-limited settings, which I agree with, but I believe many of our colleges and high schools qualify as resource-limited settings. Having students help engineer and evolve these fundamental instruments is, in itself, invaluable, for they are learning how to design and develop products. The compressor, pump and density meter are examples of applying the MEC tool set.

It is not surprising that the first medical instrument we developed with the MEC system is the Coulter counter. It was one of the founding instruments of the medical equipment industry, and it helped change how medical diagnostics was done. The standard commercial Coulter is able to give accurate red and white blood cell counts, and more.

Being an early instrument, it was founded on a very simple principle. The story goes:

When we started we didn't have much money, so we made an aperture by making a small hole with a hot needle in a piece of cellophane from a cigarette package. It didn't hold up long, but we were able to count some cells. [1].

If one measures the resistance across an aperture it changes as a particle passes through it. From that change the size of the particle or cell can be determined. We started

by developing several apertures of various sizes, and then evolved it into an instrument. Bioengineering undergraduate, Madeline Mullen, was the lead intern on this project and was able to measure the passing of small bubbles. Our Coulter counter was the first instrument to use the pulsing solenoid valve, and it proved that the valve functions over repeated operations. Several different aperture designs are in development. We have built two Coulter instruments. The second, with its complex parallel structure, was one of the driving forces for shifting to the I2C serial data structure we use today. A Python command line menu structure was developed to control the solenoid valve and record information from the analog-to-digital converters on the Arduino for the first Coulter instrument. The software for the second Coulter was never completed because of its inherent custom nature. Details of the particle sensor device and then the two instruments can be found in the chapter 12 on the Coulter.

One of the most advanced systems I worked on, and one that really demonstrates the ability of the MEC system, is the analytical bioreactor. It can be utilized for applications ranging from producing biofuels to curing diseases. One application I want to develop is a personalized bioreactor, in which one trains a patient's T-cells to attack the cancer that person is fighting. Control of the cellular environment is needed for this type of application, and this is something our bioreactor is well equipped to do. Our analytical bioreactor is structured in a loop, with thermal and media flow control in that loop. The bioreactor has the ability to inject air or gases such as CO₂ into the media. It has a media supply and waste disposal system. It also has a serial dispensing system with precise control over injection volumes of whatever biomaterials, small molecules or nutrients are needed. Cell

density can be maintained and controlled through an MEC optical density sensor. It has an onboard minicomputer with a display and keypad for basic operations. This bioreactor could also be used in classroom environments enabling high school or college students to grow cells and be energized with the challenges of engineering real world solutions. Chapter 13 has been devoted to the bioreactor and its subsystems.

An instrument that was developed the summer of 2016 was the weight scale accelerometer. It was designed to weigh a bottle by identifying accelerometer patterns at different volumes of fluids. A bottle which can be filled with different volumes of water is placed on a spring platform to isolate it from the table. A small, high-speed vibrator shakes the bottle, and an accelerometer measures the bottle's acceleration in the X, Y, and Z directions. As we place different volumes of fluid in the bottle, we measure different complex acceleration patterns over time. Even for the same setup these acceleration patterns can differ, sometimes by an infinitesimal amount and sometimes significantly. We are developing automated software to read and analyze these patterns. The setup is simple, so this offers us the ability to collect large amounts of data, which is precisely what is needed for developing predictive software. The first part of the data analysis was to perform a Fourier transform of the raw data. We can then start determining statistics on these graphs, such as minimum, maximum, mean, median, and standard deviation. We are also studying frequency peaks and amplitudes. We hope the methodology of determining weight through the studying of vibrational patterns, and their characteristics across a large data set, will offer insights into the software for disease prediction through monitoring of a range of diverse biomarkers over time. The weight scale accelerometer is discussed further in chapter 14.

As the projects increased in complexity, as can be seen in the Coulter 2 instrument, the parallel electronic bus structure also increased in complexity, creating custom designs in both hardware and software. To overcome this problem, we shifted to a serial bus structure. This allowed us to simplify both the hardware and software. We developed an I2C test board for testing devices. The Arduino takes a much less prominent role in this serial I2C system. It acts as a translator between serial USB connections from a computer to the serial I2C of the MEC system. Our I2C bus requires four wires: 5 V power, GND (ground), SDA (Serial Data), and SCL (Serial Clock). We added one additional wire, if needed, which was 9 Volt or 12 Volt power. This was used for powering solenoids, motors, and any other device that required more power. This shift enabled essential changes in hardware, and equally important (if not more), changes in software. What it effectively did was allow for standardization and simplification of electrically controlled components for evolving new systems. This simplification on the software side is so straightforward that it allows for automatic code generation. Our early I2C test board was able to test many MEC components, including LEDs, buttons, potentiometers, LCD displays, 4x4 keypads, and solenoid valves. Many of these devices were originally tested with the parallel structure but are now able to be controlled through the I2C bus. For more details on the I2C test board see chapter 16.

There have been two major recent efforts in MEC component development. First, was the development of a large family of electronic MEC components (with a large set of I2C-based devices), and second, was auto-generated software. The auto code generation is the result of an accumulation of work by Adam Veres, Chip Pearson, and myself. Adam Veres

worked on the Python or computer side, Chip Pearson worked on the AVR microcontroller side, and I worked on developing the integration of the two with auto code generation. The result is an incredibly powerful piece of software. The auto code generation software allows for the initial automated construction of instrument software, which I call an “instance”. The MEC mechanical system allows for the easy construction of complex instruments such as the bioreactor, yet without an equivalently powerful software system, the control and operation of these systems would be severely limited. What made this software system possible was the serial I2C device bus structure that I envisioned. The way our system functions is the computer (often being directed by the user from a menu) sends a command set to the Arduino. The Arduino translates this to the I2C bus. The first part of the command set is the AVR microcontroller address, which is sent down the serial bus, with the selected AVR then acknowledging. Then, the device ID is sent down the I2C serial bus, along with a command. The command could be anything from turning on an LED to reading a button. LED and button are assigned to microcontrollers, and given device IDs. Devices are developed with a standard set of commands. The menus that are needed for control of these devices are auto-generated from a script. The bottom line is that complex systems can be constructed, and the core of software that is required can be auto generated on both the computer and microcontroller side. The auto generated software has the ability to test each of its components. The details of the auto code generation can be found in chapter 17.

In this introduction I may mention something in a paragraph or a few words that may have taken my interns and myself months or years to develop. I wanted to

acknowledge the interns who have helped evolve the MEC system and other projects. In addition to the projects mentioned above, which I have devoted a chapter to, there are several other important MEC projects that my interns and I have evolved which I wish to list: phage display, 3D printer evaluation, coil winder, and loop mixer. Outside the MEC system my interns and I have worked on web development, grants, tutorials, MRI monitored cell absorption and microfarming. Their work has been greatly appreciated. I've included a chapter with a picture and a statement from many of my interns, both current and alumni. These statements were collected for our website. Working with my interns, I hope, has been as valuable to them as it has been to me. Sometimes, it is frustrating to me how slowly we move forward, but when one observes who we have touched and the body of work we have produced as a whole, we are making progress. My interns can feel good about the advancements they added in the MEC system, and I feel good about the engineering skills they have acquired or improved upon, which includes: Solidworks, Python, C++ microcontroller programming, Visio, 3D printing, web design, proposal writing, video design, tutorial writing, reading specification sheets, soldering, CNC machining, jig saw cutting, drilling, Microsoft Excel, Adobe Illustrator, MATLAB, system design, leadership, problem solving and data analysis. The current intern statements can be found in chapter 18, and the alumni intern statements can be found in chapter 19

As undergrads joined the TEC center to work on these projects, Dr. Grover's lab started to overflow, so he found us a spot in Pierce Hall. I called the new space the TEC Center, for Technology Evolutionary Components Center, where I was to continue building this tool kit of parts to solve real-world problems. With the new space I could handle more

interns, so an e-mail was sent out by Jun Wang in search of around five new undergraduates to assist in the MEC system development. When over 50 undergraduates showed up, I was excited and perhaps a bit overwhelmed. The room was packed and they were all eager to help. Here is where I learned that there was a large deficiency of good undergraduate intern positions for students at UC Riverside, and probably at most universities. After reading their resumes, I found that most of the new undergrad recruits had little to no technical experience. I needed experienced Solidworks, Python and AVR developers. I could have done interviewing and quickly narrowed it down to five qualified students. The problem was that they all needed more practical experience. I had mechanical engineers who had never used a drill press or jig saw. I had electrical engineers who had never used an oscilloscope, and I had computer science students who had never coded a microcontroller. So I put a special program under the TEC center called “Outreach”. The students work in teams to develop a “Rube Goldberg” marble machine to move a marble around a track. Outreach was designed to teach a large number of students engineering skills and principles, preparing them for rigorous real-world challenges that the TEC Center works to solve, challenges that they would face in the workforce. I built multidisciplinary teams of undergrads. With this many students it was easy to find students who had some experience in an area or had just learned a skill, and they would then be selected to assist those who needed help. The marble machine proved to be well suited for advancing the MEC system, software, and interns. The TEC center has turned into a model for teaching STEM education and the creation of a biomedical Makerspace, the first of its kind. Information on Outreach and intern projects of the marble machine can be found in chapter 21.

I began with a desire to address the application of early disease detection, and for that effort I developed a tool kit of components for building instruments (MEC system). The MEC system, after intense effort by myself and my interns, is just reaching the point that it is ready to address real world applications. So, I have begun engaging with practitioners. I've been talking with professors, looking for where the MEC system could be applied. I hope to reach out to practitioners at the School of Medicine at UCR, for problems they face. One of my more recent connections is with Noah Haas from Floating Doctors. He works with the poor rural communities of Panama. We have discussed problems ranging from environmental issues of the ocean, to health issues of the people. From these discussions I have listed the challenges and have begun to look at potential solutions. Some of these solutions my interns have started developing, such as the bug zapper. Connecting with practitioners is what can bring focus to my work and inspire my students. A list of problems and the start of solutions for the Panamanian people can be found in chapter 22.

It is my hope that the reader of this dissertation will walk away with a belief that component-based design is essential for biomedical research, and, in fact, for any research where fluidic instrumentation is required. I hope that one would start to have an understanding of what that tool set might look like, and the characteristics it must have. All my instrument examples simply show what can be done with such a tool set. Next, I want the reader to walk away with the dream that I have; that if such a tool set existed it would allow for a more rapid evolution of medical science, which, in turn, saves lives. Finally, I hope the reader has a realization that this tool set is already changing lives, because it lowers the barriers to students engaging in real-world problem solving. The TEC Center

has provided an environment where students learn professional skills. This is my greatest accomplishment, to date, at UC Riverside. The TEC center can become a model for how STEM education can be accomplished in a Makerspace environment centered around the MEC system.

Chapter 2

MECs: “Building Blocks” for Creating Biological and Chemical Instruments

Reprinted with permission from Douglas A. Hill, Lindsey E. Anderson, Casey J. Hill, Afshin Mostaghim, Victor G. J. Rodgers, and William H. Grover, “MECs: ‘Building blocks’ for creating biological and chemical instruments.” *PLOS One* **11** (7), e0158706.

2.1 Abstract

The development of new biological and chemical instruments for research and diagnostic applications is often slowed by the cost, specialization, and custom nature of these instruments. New instruments are built from components that are drawn from a host of different disciplines and not designed to integrate together, and once built, an instrument

typically performs a limited number of tasks and cannot be easily adapted for new applications. Consequently, the process of inventing new instruments is very inefficient, especially for researchers or clinicians in resource-limited settings. To improve this situation, we propose that a family of *standardized multidisciplinary components* is needed, a set of “building blocks” that perform a wide array of different tasks and are designed to integrate together. Using these components, scientists, engineers, and clinicians would be able to build custom instruments for their own unique needs quickly and easily. In this work we present the foundation of this set of components, a system we call Multifluidic Evolutionary Components (MECs). “Multifluidic” conveys the wide range of fluid volumes MECs operate upon (from nanoliters to milliliters and beyond); “multi” also reflects the multiple disciplines supported by the system (not only fluidics but also electronics, optics, and mechanics). “Evolutionary” refers to the design principles that enable the library of MEC parts to easily grow and adapt to new applications. Each MEC “building block” performs a fundamental function that is commonly found in biological or chemical instruments, functions like valving, pumping, mixing, controlling, and sensing. Each MEC also has a unique symbol linked to a physical definition, which enables instruments to be designed rapidly and efficiently using schematics. As a proof-of-concept, we use MECs to build a variety of instruments, including a fluidic routing and mixing system capable of manipulating fluid volumes over five orders of magnitude, an acid-base titration instrument suitable for use in schools, and a bioreactor suitable for maintaining and analyzing cell cultures in research and diagnostic applications. These are the first of many instruments that can be built by researchers, clinicians, and students using the MEC system.

2.2 Introduction

The importance of instrumentation in research, industry, and healthcare is difficult to overstate. From a simple incubator in a biology lab to a sophisticated genetic analyzer in a hospital, instruments provide essential automation and quantification. Consequently, new instruments accelerate research and open doors to potentially lifesaving diagnostics and treatments.

New instruments that are purely electronic can be designed and built relatively easily because they are comprised of electronic components (resistors, capacitors, integrated circuits, etc.) that were designed for easy interconnectivity. However, many instruments (especially those in the chemical, biological, and medical fields) are not only electronic but also include fluidic, mechanical, and optical elements. With no standardized components designed for interconnectivity, new instruments that combine fluidic, mechanical, optical, and electrical components are relatively slow and inefficient to build. As a result, scientists and clinicians may recognize the need for a new instrument in their work but be unable to obtain an instrument or build their own because of the specialized training, equipment, and time needed to design and build new instruments.

In this work, we introduce “Multifluidic Evolutionary Components” (MECs), a system of “building blocks” that can be used to create complete, functional instruments quickly and easily. The term “multifluidic” recognizes the fact that real-world chemical and biological instruments are often *multiscale*, operating on fluid volumes ranging from nanoliters to milliliters and beyond. While microfluidic “building blocks” have been demonstrated that allow for custom microfluidic instruments [2, 3, 4, 5], these systems are purely microfluidic

and cannot also operate on the milliliter-and-larger volumes commonly encountered in real-world chemical, biological, and medical instruments. Our term “multifluidic” also reflects the fact that real-world biological and chemical instruments are often *multidisciplinary*, containing not only fluidic but also mechanical, optical, and electronic parts. Most existing microfluidic “building blocks” are purely fluidic and do not provide mechanical, optical, or electronic components. Each MEC “building block” performs a fundamental function that is commonly found in biological or chemical instruments, functions like valving, pumping, mixing, controlling, and sensing. By combining MECs together, researchers can build custom biological or chemical instruments quickly and easily.

Fig 2.1 provides an overview of the process used to create an instrument using MECs (in this case, a fluidic router capable of pumping and mixing various fluids). Instrument design begins with a user arranging MEC symbols into schematics. Each symbol is linked to a physical definition of the component. Two types of MECs are used: *macroMECs* (milliliter-scale off-the-shelf components which can be “snapped together” for many applications) and *microMECs* (nanoliter- or microliter-scale components which are combined in a micro-schematic and made using any conventional microfluidic technique like etching, embossing, or soft lithography). After the design of the instrument is finalized, the schematic guides the user in assembling macroMECs and microMECs into a complete and functional instrument.

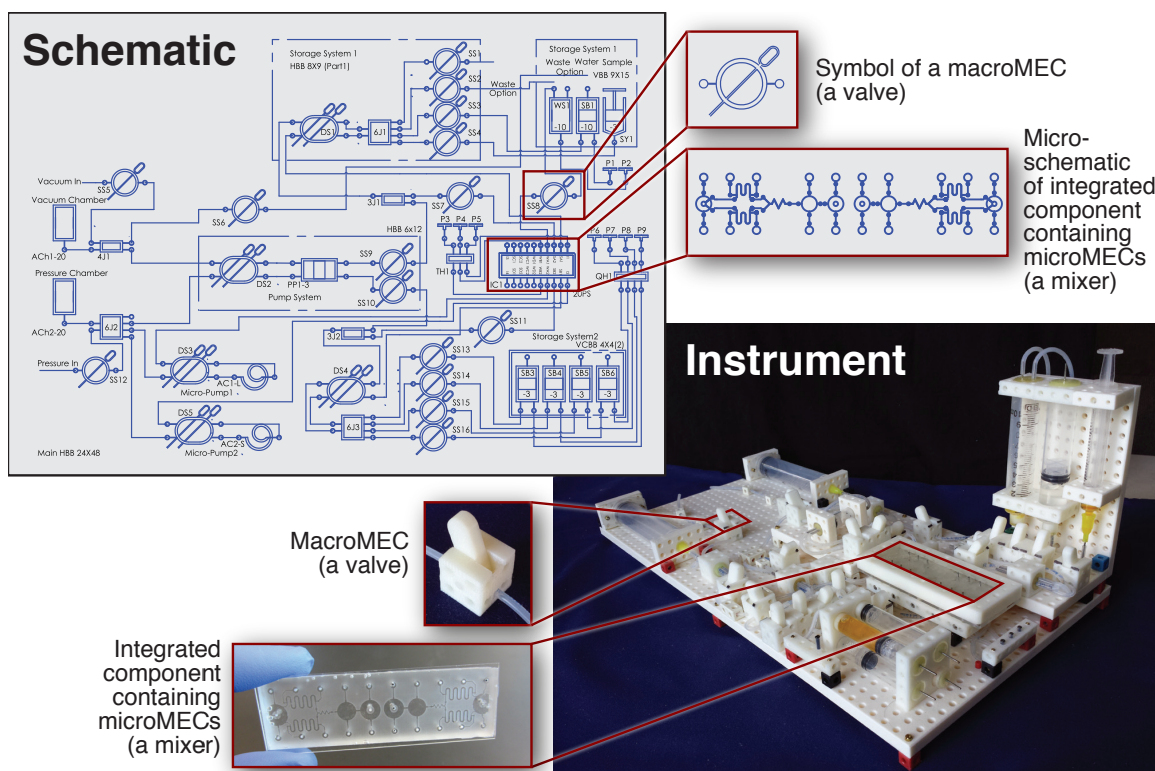


Figure 2.1: Overview of designing and building a custom instrument using the Multifluidic Evolutionary Component (MEC) system. The instrument is designed by arranging MEC symbols on a schematic, then the schematic is used to guide the assembly of MEC “building blocks” into the finished instrument. For many applications involving manipulating fluids on the milliliter scale, the “off the shelf” macroMEC components (like the manual valve shown) are adequate for building complete instruments. For instruments that must manipulate fluids on the microliter or nanoliter scale, microMECs can be arranged into a micro-schematic (like the mixer chip shown), fabricated using conventional microfabrication techniques, packaged in a MEC shell, and plugged into the rest of the instrument.

2.3 Methods

2.3.1 “Evolutionary” design principles

The MEC system was designed to be “evolutionary,” to easily adapt to a wide range of new applications and easily accommodate new components. To accomplish this, we incorporated design principles from two existing systems that are already “evolutionary” by our definition: *biological cells* and *electronic components*. The specific qualities of cells and electronics that we attempted to replicate in the MEC system include:

Packaging. Biological cells are packaged in a cell membrane that controls mass transport into and out of the cell, and electronic components like integrated circuits are typically packaged in cases with metal pins for input/output connectivity. In both cases, the package’s contents and function may vary widely in different cell or chip types, but the package itself remains relatively constant. By wrapping contents in standardized packages, components can be interconnected easily and efficiently. Similarly, in the MEC system a structured package design (Fig 2.2) enables a variety of different fluidic, electronic, optical, and mechanical components to be interconnected with each other.

Symbolic design. DNA provides a set of symbols that define the design and function of a cell, and schematics serve the same purpose in electronics. Likewise, in the MEC system design begins at the symbolic level, where each component has a symbol linked to a physical definition. Several MEC symbols are combined to generate a schematic, and the final instrument is assembled by following the schematic like a set of instructions. Schematics facilitate the design of complex MEC-based instruments, enable multiple designers to easily

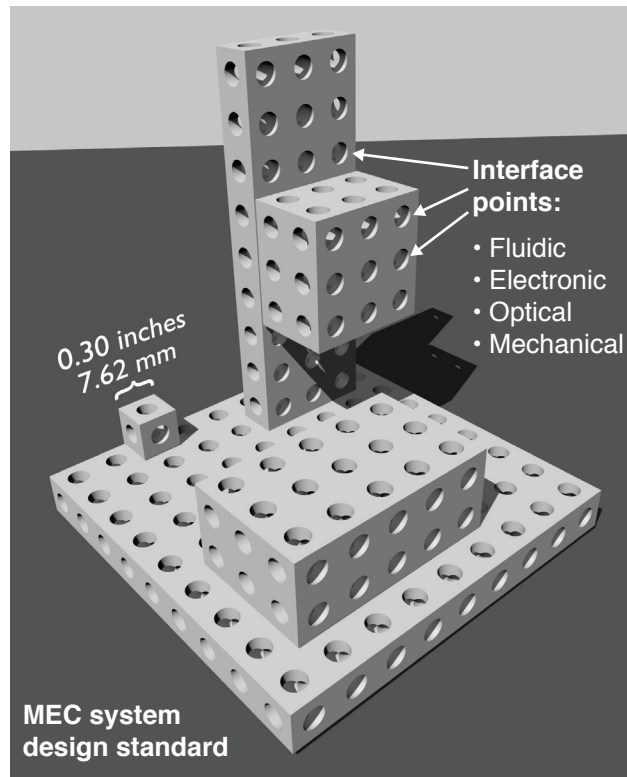


Figure 2.2: Design standard for the MEC system. Standardized sizing and spacing allows MECs to be connected together in all three spatial dimensions, and interface points (circles) can be designed to allow fluid, electricity, or light to flow between MECs.

communicate and collaborate with each other, encourage the sharing of MEC instrument designs, and allow new MEC instruments to build on the designs of older MEC instruments.

Interdisciplinary components. The functions of biological cells span many different disciplines, including optics (rod and cone cells in the eye), electronics (nerve cells), mechanics (bone cells), and so on. Similarly, many electronic components are also optical (LEDs and photodiodes), mechanical (solenoids and strain gauges), and so on. Integrating multiple fields or disciplines into a set of components greatly expands the range of applications for those components. For this reason, the MEC system includes components with fluidic, electronic, mechanical, and optical functionalities, packaged in a way that facilitates interconnection.

Standardized three-dimensional interfaces. Biological cells can communicate and exchange matter with their neighbors in all three spatial dimensions. Interestingly, electronic circuits are largely planar and communicate mainly in two dimensions, a limitation that electrical engineers have struggled with for the last half century [6]. To avoid the limitations of two-dimensional interconnectivity, the MEC system uses a three-dimensional interface standard shown in Fig 2.2. Consistently-sized and regularly-spaced interface points (the circles in Fig 2.2) can contain mechanical, electrical, fluidic, or optical interfaces. Fluid, light, or electric current flow from one MEC to another through these interfaces.

2.3.2 Open standard for MEC interfaces

To make it easier for other researchers to develop their own components that can interface with the MEC system, we are maintaining an open standard for MEC interfaces.

The current version of the standard is summarized below, and the latest version is available for reference on our website [7].

Mechanical interface: As depicted in Fig 2, MEC blocks are designed in 0.3 inch increments (a block’s nominal size in each dimension can be 0.3 inches, 0.6 inches, 0.9 inches, and so on). Each dimension of a MEC is actually designed to be 0.005 inches smaller than a 0.3-inch multiple to allow space for two MECs to be connected side-by-side (for example, the 0.3 inch thick mechanical baseboard MEC is actually 0.295 inches thick). Surfaces of MECs contain rectilinear arrays of holes spaced on 0.3 inch increments. The diameter of each hole changes along the depth of the hole: the topmost (and bottommost) 0.0865 inches of the hole has a diameter of 0.125 inches, and the middle of the hole has a diameter of 0.092 inches. The larger 0.125-inch diameter portion of the hole is designed to receive a 0.125 inch diameter and 0.175 inch long pin. Using these pins, two MECs can be connected together using a simple and reversible friction-fit assembly. Alternately, a standard #2-56 screw can fit through the 0.092-inch-diameter portion of a hole and self-thread into a 0.076-inch-diameter hole on a MEC part. This provides a mechanically stronger alternative to the pins for connecting two MECs together.

Fluidic interface: The holes on 0.3-inch-spaced increments can be sized to accommodate flexible tubing (e.g., Tygon brand) with 3/32 inch outer diameter and a 1/32 inch inner diameter. A short length of tubing inserted into a hole can then serve as a fluidic socket. A 20-gauge rigid tube (e.g., stainless steel) can then be inserted into the tubing-based fluidic socket. This creates a simple and reversible friction-fit fluidic connection between two MECs. Alternately, two MECs whose faces do not touch (and therefore

cannot be easily connected by the rigid tube) may be connected using a length of flexible tubing. In this case, a 20-gauge rigid tube is permanently inserted into each MEC (extending at least 0.29 inches out of the MEC surface), and flexible tubing is slipped onto these tubes to form a reversible fluidic connection between the two MECs. For applications that must withstand higher fluid pressures, a retaining ring (0.125 inches outer diameter, 0.086 inches inner diameter, and 0.175 inches long) can be slipped on the outside of the flexible tubing at each MEC to serve as a compression fitting.

Electrical interface: The 0.3-inch-spaced holes described above can also accommodate metal pins, conductive rubber pins, metal springs, or metal screws to provide electrical connectivity between MECs.

Optical interface: Optical elements like light emitting diodes, phototransistors, and fiber optics can be designed into the 0.3-inch-spaced holes and used to provide optical interfaces between MECs.

2.3.3 Fabrication of MECs

As long as MECs are designed and packaged to interface together, virtually any material or fabrication technique can be employed to make MECs. In this proof-of-concept demonstration, we mostly used rapid prototyping (3D printing) and CNC machining to fabricate the larger-scale macroMECs. Two 3D printers were used: a fused deposition modeling (FDM) printer (Dimension Elite by Stratasys, Eden Prairie, MN) which uses melted acrylonitrile butadiene styrene (ABS) to form a part, and a stereolithography (SLA) printer (Form 1+ by Formlabs, Cambridge, MA) which uses ultraviolet light to cure liquid methacrylate resins into a solid part. To fabricate microfluidic integrated components that

contain microMECs, we used a custom casting process we developed based on soft lithography [8] for polydimethylsiloxane (PDMS) components, and conventional photolithography and wet etching [9] for borosilicate glass components.

2.3.4 MacroMECs

Each of the macro-scale Multifluidic Evolutionary Components (macroMECs) shown in Fig 2.3 performs a fundamental function that is commonly found in real-world instruments. They include fluidic macroMECs for storing, pumping, and controlling milliliter-scale volumes of fluids (Fig 2.3A), electronic/fluidic and electronic/optical macroMECs that control and measure fluids and light (Fig 2.3B), electronic macroMECs that perform logical and sensing operations (Fig 2.3C), and mechanical macroMECs that form the mechanical framework or skeleton for MEC-based instruments (Fig 2.3D).

Fluid storage. A variety of fluid storage MECs contain fluid samples, reagents, and waste in MEC-based instruments. Some of these MECs (like SB1-3 in Fig 2.3A) are modeled after syringes and have moveable plungers that separate the fluidic and pneumatic sides of the component. Air pressure or vacuum applied to the pneumatic side empties or fills the fluidic side of the storage MEC, thereby creating a storage reservoir that doubles as a simple pump. In addition, a family of unit storage MECs (like SU1 in Fig 2.3A) can contain variable volumes, with the precise volume determined by the length of an internal spacing pin. Other storage MECs interface to more conventional fluid reservoirs like microcentrifuge tubes (VH1 in Fig 2.3A), syringes (SS1), and milliliter-scale containers (SB1-10 and WS1).

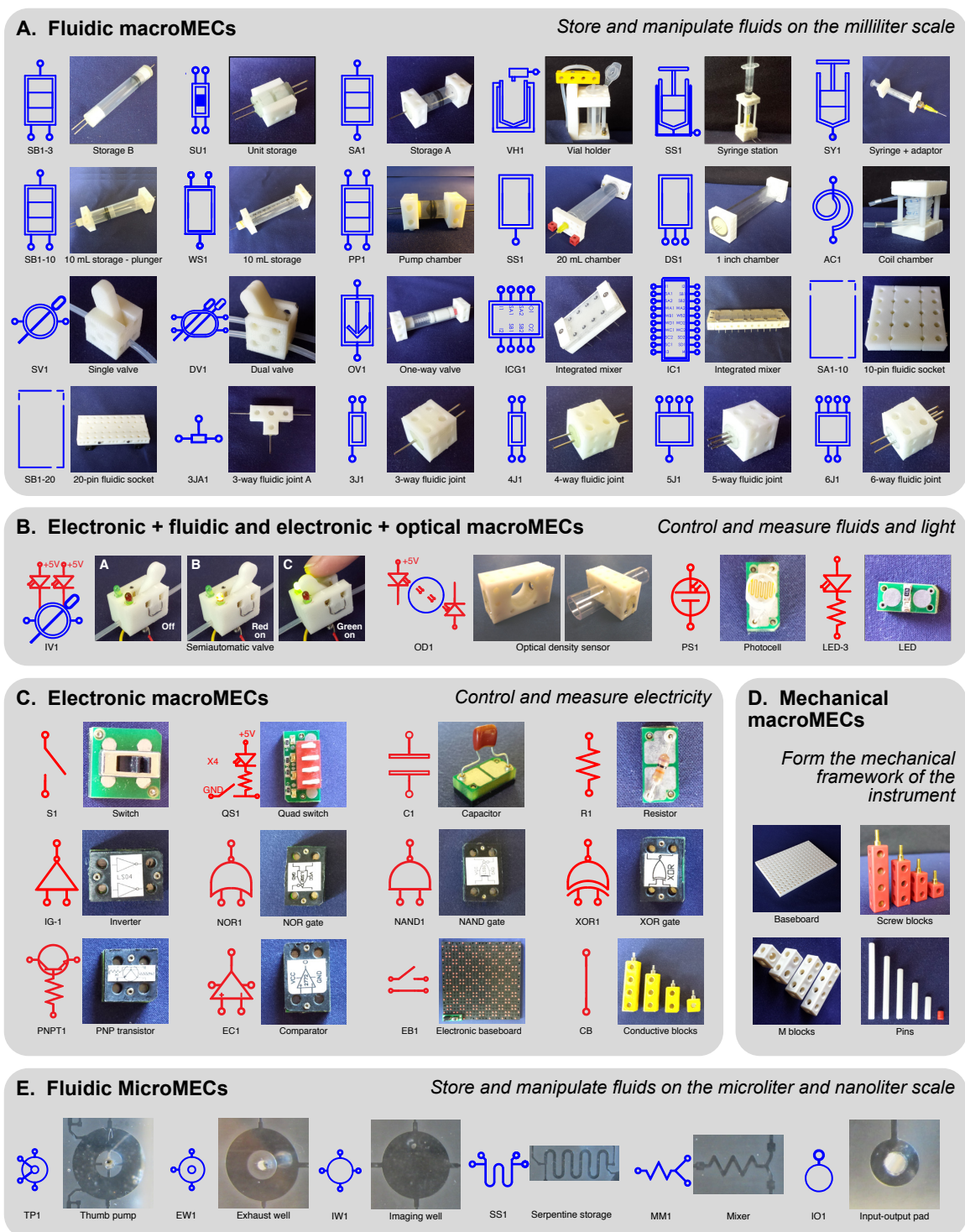


Figure 2.3: Selected Multifluidic Evolutionary Components (MECs), including macroMECs that can be “clicked together” (A–D) and microMECs that are fabricated into microfluidic integrated components (E).

Vacuum and pressure chambers. Portability is important for many instruments intended for point-of-care or field applications. Portable MEC-based instruments are powered by an onboard pneumatic vacuum and pressure chamber MECs (DS1 in Fig 2.3A). These chamber MECs are analogous to batteries in electronics; they provide a pressure differential to pull or push fluid through other components.

Manual valves. Valve macroMECs control the flow of fluid in a MEC-based instrument. As the operator toggles the lever on the mechanical pinch valve MEC (SV1 in Fig 2.3A), the lower part of the lever either pinches the internal tube closed (stopping the flow of fluid through the valve) or releases the tube (allowing fluid flow to resume). A dual pinch valve MEC (DV1 in Fig 2.3A) has four fluid connections and three states. In the first state (switch up), fluid flows between connections 1 and 2. In the second state (switch down), fluid flows between connections 3 and 4. In the third state (switch in the middle), all fluid flow is stopped. This behavior (called “break before make” in electronic switches) is particularly useful in applications requiring switching between two sources (say, pneumatic pressure and vacuum) that should never be connected together during valve operation.

Pumps. The ability to pump fluid is a fundamental requirement in many biological and chemical instruments. The MEC system provides several different pumping methods. First, as described above, some fluid storage MECs (like SB1-3 in Fig 2.3A) are equipped with internal plungers and function as simple pumps; they are well-suited for operations such as rinsing or waste removal that require the entire contents of a storage MEC to be delivered. Second, by combining a pump chamber MEC (PP1 in Fig 2.3A) with a dual valve (DV1) and two single valves (SV1), a milliliter-scale “unit volume” pump is created

that is useful for pumping known volumes of fluids between MECs. The volume dispensed from the pump with each actuation is determined by the volume of the pump chamber. Third, a coil chamber MEC (AC1 in Fig 2.3A) combined with a dual valve (DV1) creates a microliter-scale unit volume pump. The volume of air pumped per actuation ranges from a few to hundreds of microliters and is determined by the volume of compressed air held in the coil chamber MEC.

Fluidic sockets. Integrated components (microfluidic chips containing microMECs, described below) connect to the rest of the MEC system through fluidic socket macroMECs (SA1-10 or SB1-20 in Fig 2.3A). These sockets provide an array of holes containing elastic tubing that receives stainless steel fluidic pins on the integrated components. Fluid passes between the microfluidic integrated component and the rest of the MEC-based instrument through these sockets.

Semiautomatic valves. Complex instruments for biological and chemical applications can require a large number of valves. If these valves are actuated by hand, like the mechanical pinch valve MECs, actuation steps may be missed or improperly timed. To reduce this risk while still keeping the overall component cost low, the “semiautomatic” valve (IV1 in in Fig 2.3B) combines computer control with manual actuation. This valve contains LEDs that interface to an Arduino-based microprocessor MEC. The red light indicates to the operator that a particular valve needs to be toggled. Once the valve is switched, the red light turns off and the green light comes on, indicating to the operator that the valve was properly switched. The green light signal can also be sent back to the microcontroller to confirm that the valve’s state is now correct; the microcontroller could then turn off

the light. This combination of manual operation with computer-guided control reduces component cost while still supporting complex operations and reducing error rates.

Electronic MECs. A variety of electronic macroMECs support electrical and logical functions in MEC-based instruments (Fig 2.3C). Discrete components like resistors (R1) and capacitors (C1) plug into a two-pin macroMEC adaptor. This allows a large array of standard electronic components to be packaged as macroMECs and incorporated into MEC-based instruments. Switches (S1 and QS1) and transistor (PNPT1) MECs control the flow of electric current between two or more MECs. A family of logic macroMECS (IG-1, NOR1, NAND1, and XOR1) can perform all the standard Boolean logic functions. Using logic MECs, the operation of an instrument can be defined and controlled. Finally, the the power and ground baseboard (EB1 in Fig 2.3C) contains an embedded printed circuit board. This baseboard contains an array of power and ground connections arranged in a checkerboard pattern, and a jack for connecting a standard 5 V wall-mounted power supply. This baseboard provides many convenient locations for connecting and powering electronic macroMECs.

Sensor and measurement MECs. A variety of sensors can be connected to a macroMEC adaptor and used in the MEC system; for example, a photocell MEC (PS1 in Fig 2.3C) enables MEC-based instruments to measure light intensity. A more specialized sensing MEC, the optical density MEC (OD1 in Fig 2.3B), includes a light source (an LED) and light sensor (a photodiode) oriented directly across from each other on a transparent tube. By measuring how much light from the LED is detected at the photodiode, the optical density MEC can measure many different properties of the fluid flowing through the tube,

including solute concentration (using the Beer-Lambert law), turbidity or optical density (a measure of how many particles or cells are in the fluid), fluid level sensing (using the different indices of refraction for gas and liquid), and so on.

Mechanical MECs. A set of structural macroMECs (Fig 2.3D) provides the mechanical framework for MEC-based instruments. Hollow blocks can contain plastic pins (for connecting MECs together mechanically), metal rods (for electrical MEC connections), tubing (for fluidic MEC connections), etc. The mechanical baseboard MECs provide a foundation for MEC assembly and can be oriented horizontally or vertically (see Fig 2.1 for examples of both orientations).

2.3.5 MicroMECs

Like the macroMECs described above, micro-scale Multifluidic Evolutionary Components (microMECs) perform fundamental functions like pumping, mixing, and storage, only on a smaller scale (microliters and nanoliters). Each of the microMECs has a symbol associated with it (Fig 2.3E). A few of the microMECs are detailed below.

Micropumps. The thumb pump microMEC (TP1 in Fig 2.3E) contains a microliter-scale volume of liquid or gas. When pressed by a finger in a manner described previously [10], the pump is designed to push fluid downstream to other microMECs. An exhaust well microMEC (EW1 in Fig 2.3E) vents displaced liquid or gas from the chip.

Fluid storage and metering. The serpentine storage microMEC (SS1 in Fig 2.3E) holds a fixed fluid volume of 400 nL. It is designed with two sets of inlets and outlets. Other nanoliter- or microliter-scale storage microMECs can be designed using differently-

sized channels. Additionally, the imaging well microMEC (IW1 in Fig 2.3E) has a transparent glass top for imaging its contents.

Mixing. A simple mixing microMEC (MM1 in Fig 2.3E) is designed to combine the contents of two channels.

Fluidic input/output. The circular input/output pad (IO1 in Fig 2.3E) provides a fluid interface between other microMECs in an integrated component and a fluidic pin (a stainless steel tube). This pin then serves as the fluidic interface to macroMECs as described above.

2.3.6 Integrated components.

MicroMECs are combined into *integrated components*, chips that can be fabricated using conventional microfabrication techniques, packaged inside a MEC-standard mechanical shell, and plugged into the fluidic socket macroMEC described above. Fig 2.4 shows two examples of integrated components, one fabricated in PDMS using soft lithography (Fig 2.4A) and one fabricated in glass using wet chemical etching (Fig 2.4B).

2.4 Results

To evaluate the performance of the Multifluidic Evolutionary Component system, we used MECs to build several test instruments. Here we present three MEC-based chemical and biological instruments: a fluidic router capable of sending fluids between several storage MECs, a prototype acid-base titration instrument for educational use, and a research-grade bioreactor capable of measuring the growth rate of cells in culture.

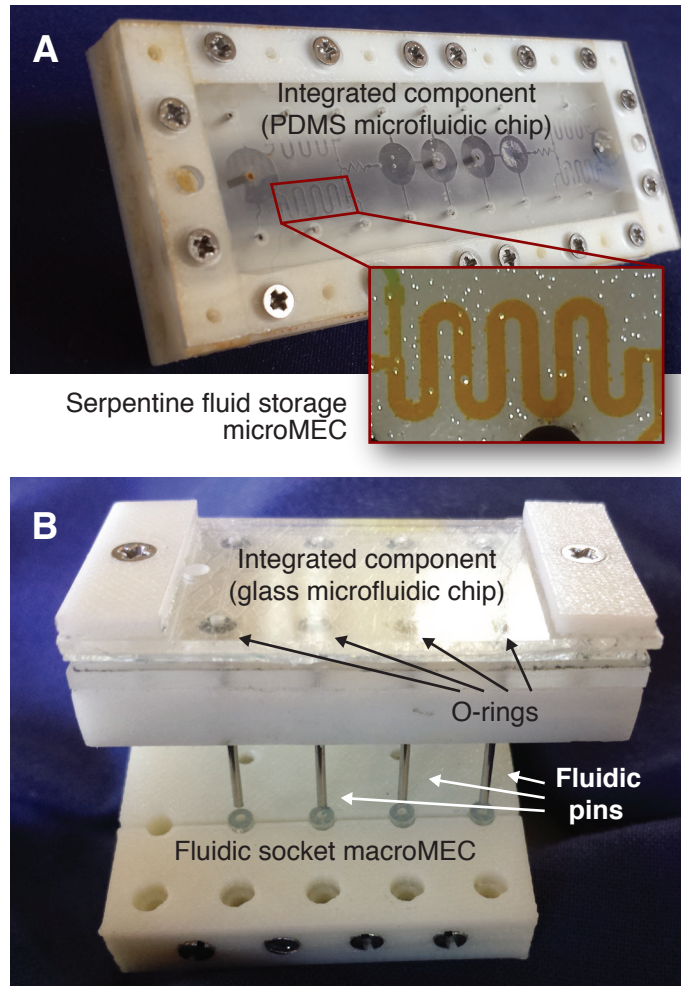


Figure 2.4: Two *integrated components*, microfluidic chips containing several microMECs combined together and fabricated using soft lithography of PDMS (A) and wet etching of glass (B). The integrated components are packaged in a MEC-standard shell that plugs into the fluidic socket macroMECs (SA1-10 or SB1-20 in Fig 2.3A). By packaging a microfluidic chip in a MEC shell in this manner, virtually any microfluidic chip can be integrated into the MEC system.

2.4.1 Fluidic router

To demonstrate the complexity of instruments that can be built using MECs and the multiscale capability of the MEC system, we developed the fluidic router shown in Fig 2.5A. It was built from our current set of standard MEC components and combined both macroMECs and microMECs in the same instrument. The instrument is designed to route fluids between nine fluid storage MECs in several different ways, in volumes spanning five orders of magnitude (from nanoliters to milliliters). The fluidic router also demonstrates a serial fluidic bus design in which several fluid storage MECs are connected in series by a single channel. This design accommodates additional storage MECs with ease and reduces the risk of cross-contamination by supporting a rinse operation between fluid transfer operations. During bus rinsing, the contents of the serial bus channel are flushed to a waste storage MEC to remove residues from the previous fluidic operation and prepare for the next operation. A larger storage MEC (WS1 in Fig 2.3) was added to the MEC system to provide rinse fluid for this purpose. The router instrument also demonstrates the ability to interface between microMECs and macroMECs using the fluidic socket.

During operation, the router instrument’s pressure tanks are first pressurized as described earlier. The pressure tanks shown on the top of the fluid router were selected from our original standard component set (SS1 in Fig 3A). Unfortunately, these tanks provided only enough air for a few pump actuations, so a larger air chamber (DS1 in Fig 3A) was developed and was mounted on the bottom of the fluidic router in Fig 5A. This new air chamber became our standard, and was fabricated in different lengths for different air capacities. The user loads a sample into the instrument via a syringe. The central pump

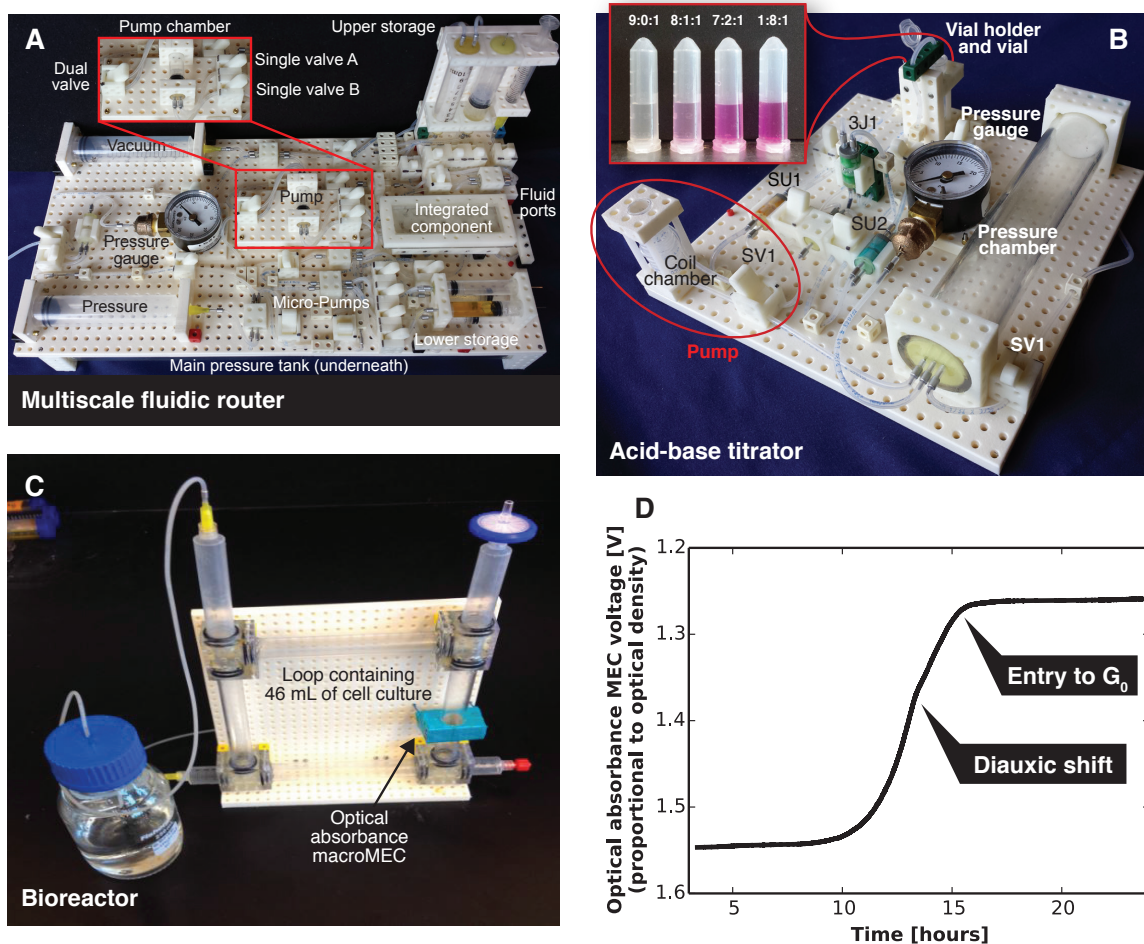


Figure 2.5: Three sample instruments built using the MEC system. (A) A MEC-based multiscale fluidic router designed to mix fluids from nine different fluid storage MECs in volumes spanning five orders of magnitude (from 400 nanoliters to 10 milliliters). (B) A MEC-based acid-base titration instrument suitable for educational applications in classrooms. By toggling the manual valves SU1, SU2, and SV1, different ratios of water : sodium hydroxide solution : phenolphthalein are delivered to a microcentrifuge vial. The observed range of phenolphthalein colors (from clear in pure water to pink in 4.0×10^{-5} mass concentration NaOH) confirms the successful operation of the instrument. (C) A MEC-based bioreactor capable of culturing cells. The optical density macroMEC (OD1 in Fig 2.3B) measures the concentration (via optical density) of the cells as they grow and react to stimuli. (D) A growth curve obtained by using the bioreactor in C to culture *Saccharomyces cerevisiae* yeast cells. The bioreactor is sensitive enough to detect important metabolic checkpoints in the growing yeast, including the diauxic shift (when the yeast cells switch from glycolysis to the aerobic oxidation of ethanol) and the entry into G_0 (when the yeast cells exhaust all nutrients and enter stationary phase).

can be used to transfer fluid to the lower reservoirs, or fluid can be directly loaded from the side fluid ports on the right in Fig 2.5A. To transfer fluid using the central pump, first single valve A and two other valves are opened to allow passage of fluid from the syringe or storage MEC to the pump chamber. The dual valve on the pump is toggled to allow vacuum to be applied to the pump chamber, pulling back the plunger. As the pump plunger pulls back, it draws fluid from the storage MECs into the pump chamber, with the precise volume determined by the size of the chamber. Single valve A is then turned off. One of the four lower storage MECs is then selected by toggling the appropriate single valve. Single valve B is then turned on. By switching the dual valve on the pump, pneumatic pressure is applied to the plunger in the pump chamber, pushing the plunger forward and forcing fluid out of the pump chamber into the destination lower storage MEC. In this manner, fluid can be moved between any of the storage MECs in either direction.

To mix fluids on the milliliter scale, the pump is used to route fluid from two or more storage MECs into a third storage MEC. To mix fluids on the microliter to nanoliter scale, the dual valve is switched to divert fluids through the serpentine storage microMECs on the microfabricated integrated component. Once the 400 nL serpentine storage microMECs are loaded, the coil pump microMEC is used to drive the fluid out of the serpentine storage microMECs, through the mixer, and into the imaging well. The coil pump is made from the standard dual valve and a small length of tubing. It is noteworthy that this pumping application was not envisioned when the dual valve was developed; this is an example of how standard MECs can be used in novel and unexpected ways. Fig 2.5A shows a milliliter-scale volume of fluid (dyed yellow) that has been delivered from a syringe

(or one of the other storage units) to one of the lower storage MECs by the instrument, and Fig 2.4A shows a serpentine fluid storage microMEC filled with 400 nL of fluid. The PDMS integrated component shown in the fluidic router was fabricated using an experimental casting process that caused some bonding problems and leakage between the PDMS and glass surfaces, but the socketing and interconnection of the integrated component still functioned as intended.

2.4.2 MEC-based titration instrument

To demonstrate the potential of the MEC system in educational applications, we used our standard MEC components to construct the acid-base titration instrument shown in Fig 2.5B. This system mechanizes the steps involved in performing an acid-base titration, a classic chemistry lab experiment used to measure the equilibrium constant of a weak acid or base. The MEC-based titration instrument in Fig 2.5B delivers unit volumes (100 μL) of either water, sodium hydroxide solution (5.0×10^{-5} mass concentration), or phenolphthalein solution (0.1% in 50% ethanol) to a vial. The titration instrument also utilizes the new pressure chamber macroMEC described above. This pressure chamber makes the MEC-based titration instrument portable and suitable for use in the field, in classrooms, or at the point of care. The reuse of this new pressure chamber MEC in multiple instruments demonstrates the natural evolution of MEC components: the original pressure chamber MEC had less utility than the new pressure chamber MEC, so the original MEC was naturally replaced.

During operation, a syringe is used to load pressurized air into the pressure chamber (DS1) until the chamber's contents reach the desired pressure (measured by the gauge

in Fig 2.5B). This gauge MEC is an example of how an standard industry component like a gauge can be adapted to the MEC system (in this case, by attaching the gauge to a MEC mechanical baseboard and fitting the gauge with a MEC standard tubing interface). Fluid is then loaded by syringe into each of the unit chambers (SU1 and SU2). The unit chambers can be loaded one at a time or in parallel. Next, the dual valve (SV1) is used to load the coil chamber (AC1) with pressurized air. The volume of pressurized air in the pump chamber is determined by the length of tubing in the coil. The dual valve is then switched back to apply pressure from the coil chamber to the unit chambers, forcing their fluid contents into the vial. With each valve actuation, a unit volume from one or both storage units of fluid is delivered to the vial. This pump is the same as the coil pump used in the fluidic router, except with a longer coil of tubing. The coil holder was developed for this instrument and is now a standard component (Fig 3A; AC1). The vial holder was also developed for this system, and is another example of how an existing component (in this case, a microcentrifuge vial) can be integrated into the MEC system.

To fill Vial 1 in Fig 2.5B, the titration instrument combined nine unit volumes of water, zero unit volumes of sodium hydroxide solution, and one unit volume of phenolphthalein solution (a mixing ratio of 9:0:1; final NaOH mass concentration = 0). Vial 2 received unit volumes in a ratio of 8:1:1 (NaOH mass concentration = 0.5×10^{-5}), Vial 3 received a ratio of 7:2:1 (NaOH mass concentration = 1.0×10^{-5}), and Vial 4 received a ratio of 1:8:1 (NaOH mass concentration = 4.0×10^{-5}). Each vial received a single unit volume of phenolphthalein solution (a pH indicator that turns from clear to pink at pH > 8.2). The observed colors of the solutions (ranging from clear in Vial 1 to pink in Vial 4 in

Fig 2.5B) confirm the increasing pH from Vial 1 to Vial 4 and the successful operation of the MEC-based titration instrument.

2.4.3 MEC-based cell culture instrument

Finally, to demonstrate that the MEC system can be used to create research-grade biological instruments, a MEC-based bioreactor was built. The bioreactor (Fig 5C) contains a square-shaped loop of MECs that contains 46 mL of cell culture media. This loop was constructed from a new family of MEC components that interconnect lengths of 0.5-inch diameter rigid plastic tubing. These interconnect MECs are fabricated to the MEC mechanical standards in both size and interface spacing. This allows them to easily mount to our standard baseboard as shown in Fig 5C. The media is recirculated (and cells are kept in suspension) using a stream of air bubbles, although pump MECs could also be used. The bioreactor uses the optical density MEC (OD1 in Fig 3B) to measure the optical density of the cell culture in the media loop.

After sterilizing the MEC bioreactor with a 70% ethanol solution, the bioreactor was filled with yeast extract peptone dextrose (YEPD) media, then seeded with a small amount of *Saccharomyces cerevisiae* yeast culture. The bioreactor was then placed in an incubator at 30 °C for 24 hours. An inexpensive data acquisition card (NI USB-6008, National Instruments, Austin, TX) was used to record the voltage output of the optical density MEC, though an Arduino-based microcontroller MEC (under development) could also be used. Data from the optical density MEC is available in online Supporting Information S1 File; this file includes the Python program used to analyze the data.

The output of the optical density MEC is plotted versus time in Fig 5D. The slight decrease in growth rate observed at 13 hours likely represents the diauxic shift, the point at which the glucose in the culture media is depleted and the yeast cells must switch their metabolism from glycolysis to the less-favored aerobic oxidation of ethanol. Additionally, the halt in growth observed after 15 hours indicates the depletion of ethanol in the media and the entry of the yeast cells into stationary phase (G_0). These metabolic transitions are extremely important in yeast biology and even provide insights into human conditions like cancer and aging [11]. The fact that our prototype MEC-based bioreactor is capable of observing these biologically and clinically meaningful metabolic transitions confirms that research-grade instruments can be built using the MEC system.

2.5 Discussion

In this work we presented a roadmap for developing a set of components useful in building biological and chemical instruments. Central to this roadmap are characteristics—packaging, symbolic design, interdisciplinary components, and standardized 3D interfaces—that we feel enable a set of components to adapt to a wide range of applications.

We then used our roadmap to guide the creation of a prototype set of building blocks we call Multifluidic Evolutionary Components. Because the MEC system controls fluid on multiple scales (from nanoliters to milliliters) and integrates multiple fields (fluidics, electronics, mechanics, and optics), MEC blocks can be used to build instruments for many different fields, and we demonstrated this by using MECs to create functional chemical and biological instruments. In building these instruments with our standard MEC components,

we demonstrated the utility of the component set and also showed how this component set naturally evolves.

A system of evolutionary components would allow users from a wide range of backgrounds to develop instruments faster and more efficiently. For example, users with limited technical experience and no access to fabrication resources can combine off-the-shelf macroMECs into a working custom instrument. For applications that require the manipulation of very small fluid volumes, the integrated components of microMECs can be used to perform standard microfluidic functions. Users with access to microfabrication resources can combine parts from the library of microMECs into custom microfluidic integrated components that interface easily with the rest of the MEC system. By packaging microfluidic chips in MEC shells, chips from different researchers can be easily connected together to support new applications, thereby encouraging collaboration in the microfluidics community. Finally, as the library of MECs grows and becomes more widely available, industrial users can use MECs to accelerate the development and production of new instruments for the chemical and life sciences.

Our own experience with undergraduate researchers suggests that even individuals with no prior research experience can quickly learn about the MEC system, use MECs to build functional instruments, and even design and fabricate new MECs. Thus far over 50 undergraduates from the University of California, Riverside have worked with the MEC system, and these undergraduates have been able to use MECs to build functional instruments that would not have been practical to build using conventional tools.

Evolutionary components could also prove valuable in resource-limited settings. While first-world hospitals and research universities commonly have rooms full of instruments (each of which typically performs only one or a few different tasks), hospitals and universities in developing parts of the world often do not have these resources. The MEC system could provide researchers and clinicians in resource-limited settings with the “building blocks” they need to construct their own instruments that are custom tailored to their own unique research or diagnostic needs. Having a library of MECs would give these individuals access to a virtually limitless variety of custom instruments. Realizing this vision will require the expansion of the MEC library and the mass-production of MEC parts at low per-unit costs using techniques like injection molding.

A system of standardized components like the MEC system could also help fix the “irreproducible research crisis” facing science today. Recent studies indicate that a significant fraction of published research contains results that can not be reproduced by others [12]. One practical barrier to reproducing another researcher’s results is simply not having access to the equipment needed to reproduce the results. Another barrier to confirming research is the lack of adequate published experimental details about the tools used to perform an experiment. These and other barriers to reproducible research could be eliminated by using standardized components like the MEC system. If a researcher builds their experimental tools using MECs and publishes the schematic, anyone with access to a library of MECs could replicate the researcher’s tools and confirm their results. A set of standardized components for research could have a significant positive impact on the reproducibility of scientific research.

Finally, in addition to research applications, evolutionary components can also be used in a variety of educational applications. For example, the MEC fluidic router in Fig 2.5A can demonstrate the different types of flow (laminar vs. turbulent) at different flow rates and scales, the MEC titration system shown in Fig 2.5B could be used to study acid-base equilibrium in a chemistry lab course, and the MEC bioreactor in Fig 2.5C illustrates concepts of cell growth and metabolism. The National Research Council of the National Academies recently recommended that students from kindergarten through high school supplement their science curriculum by (1) defining and delimiting engineering problems, (2) developing possible solutions, and (3) optimizing the design solution [13]. These engineering activities are a key part of the Next Generation Science Standards that eighteen states are currently adopting. The MEC system supports all three of these engineering activities and can help teachers incorporate engineering practices throughout the science curriculum.

2.6 Acknowledgments

This work was supported by the Instrument Development for Biological Research (IDBR) program of the National Science Foundation under award DBI-1353974. The authors gratefully acknowledge funding from Prof. Jerome Schultz and the Center for Bioengineering Research at the University of California, Riverside.

2.7 Supporting Information

S1 File. Data from Optical Density MEC during yeast growth curve measurement. ZIP file containing the raw data from the growth curve plot in Fig 5D,

obtained from the Optical Density Multifluidic Evolutionary Component (MEC) in the bioreactor instrument shown in Fig 5C. The file also contains the Python program used to convert the raw data into the growth curve plot shown in Fig 5D.

Chapter 3

MEC Characteristics

3.1 Characteristics of evolutionary components

As we evolved the MEC system, we examined what characteristics were necessary to make this system of components adaptable to a wide range of applications (evolutionary). This adaptability is accomplished through a number of synergistic characteristics. Evolutionary components have a *hierarchical packaging structure*. A component's packaging refers to how components are interconnected together. The package also refers to the shell that surrounds the interior parts. *Hierarchical* refers to the multi-scale nature of evolutionary components. Small parts are integrated together to form larger components which perform higher-level functions (*i.e.* macro- and micro-scale components are interfaced together). Also, evolutionary components are designed through *symbolic or code representation*. A well-defined process is used to translate an symbolic abstraction to a completed component or system. Finally, these components have *interdisciplinary characteristics*. Evolutionary

Table 3.1: Characteristics of evolutionary components, and whether three sample components (biological cells, integrated circuits, and MECs) have those characteristics. The biological cell and integrated circuit act as a guide for the design our multfluidic components. Acting together these characteristic make a system adaptable to a wide range of applications.

Evolutionary characteristic	Biological cells	Integrated circuits	MECs
Multi-scale hierarchal	Yes	Yes	Yes
Symbolic/code representation	Yes	Yes	Yes
Symbolic/code guiding efficient process	Yes	Yes	No
Standardized packaging-shell	Yes	Yes	Yes
Standardized interface spacing	No	Yes	Yes
Fundamental elements	Yes	Yes	No
Interdisciplinary component set	Yes	Some	Yes
Mechanical 3D structure critical to function	Yes	No	Yes
Component-to-component compatible	Yes	No	Yes

components integrate functions from several different fields, which allows them to adapt to a wide range of applications.

Two good examples of existing evolutionary components are the *biological cell* and the *electronic integrated circuit*. By studying the characteristics of these components, we are guided in the development of our MEC components. Shown in Table 3.1 is a non-exhaustive list of evolutionary characteristics. The table shows whether or not biological cells, integrated circuits, and MECs have these particular evolutionary characteristics.

Cells and integrated circuits get their evolutionary classification because of their adaptability. These characteristics provide the fundamental criteria for defining an evolu-

tionary component, but it is the function or adaptability that the characteristics provide, not the characteristics themselves, that determine a component's evolutionary nature. In the following parallels, we do not intend to claim that our MEC components are on the same level of sophistication as cells or integrated circuits. Rather, we seek to show that our components share certain characteristics with cells and integrated circuits, characteristics that enhance the adaptability of these components and help guide us in their development.

3.1.1 Multi-scale/Hierarchal packaging

The *multi-scale/hierarchal* packaging of components contributes to their complexity and ability to evolve higher-level functions. Animals are highly complex systems. They are constructed from a hierarchal system: cells form tissue, tissues form organs, and organs form organ systems. The organization and interconnection of cells and tissues determine their function. Electronics also has a multi-scale packaging system. Thousands of transistors are combined to form integrated circuits, integrated circuits are combined to perform tasks on a card, and those cards are combined to build whole instruments or machines such as a computer. The ease at which these hierarchal systems can be constructed is what lends itself to the adaptability in application and rise in complexity.

3.1.2 Symbolic/code representation guiding an efficient process

A system of evolutionary components produces components with many different functions. The process that produces this wide range of component functions must be well defined and guided by a formal set of instructions. In the biological cell these instructions

are provided by DNA, and in electronics they are provided by the schematic. Both of these examples illustrate design that occurs at the *symbolic* or *code* level of abstraction.

In our MEC system, much like electronics, design starts at the symbolic level where symbols are linked to a physical definition. We combine those symbols to generate a schematic, and from that schematic we assemble an instrument. The enhanced flexibility this process provides is evident in the integrated circuit, which can contain thousands of symbols in a schematic, but a designer needs to change only a small fraction of the symbols to create a new component for a new application. Schematics facilitate the design of complex MEC-based instruments, encourage sharing of MEC instrument designs, and allow new MEC instruments to build on the designs of older MEC instruments.

In electronics, symbolic design happens on two levels. On the first level, a micro-schematic defines an integrated circuit. On the second level, a macro-schematic defines a combination of integrated circuits and other macro-scale components to perform a higher-level function. We applied this methodology in the development of our macro- and micro-MEC libraries and schematics. Also as in electronics, we have developed a process to directly translate the micro-schematic to fabricated components. The ability of designers to modify schematics easily for different functions allows new work to build on old. Compared to electronics or the cell, we still have an inefficient process. The process that is used for the fabrication of MEC components still has many inefficiencies. The 3D-printing process itself is fairly efficient, but we still require several secondary operations. The original casting process was even more labor intensive which is why few cast components evolved.

3.1.3 Standardized packaging

The outer standardized packaging of evolutionary components contributes to their function, flexibility, adaptability, and producibility. In the cell the surrounding package is the membrane, and in the integrated circuit the package is the plastic case and metal pins that surround the internal chip (Figure 3.1). While the package that surrounds the internal

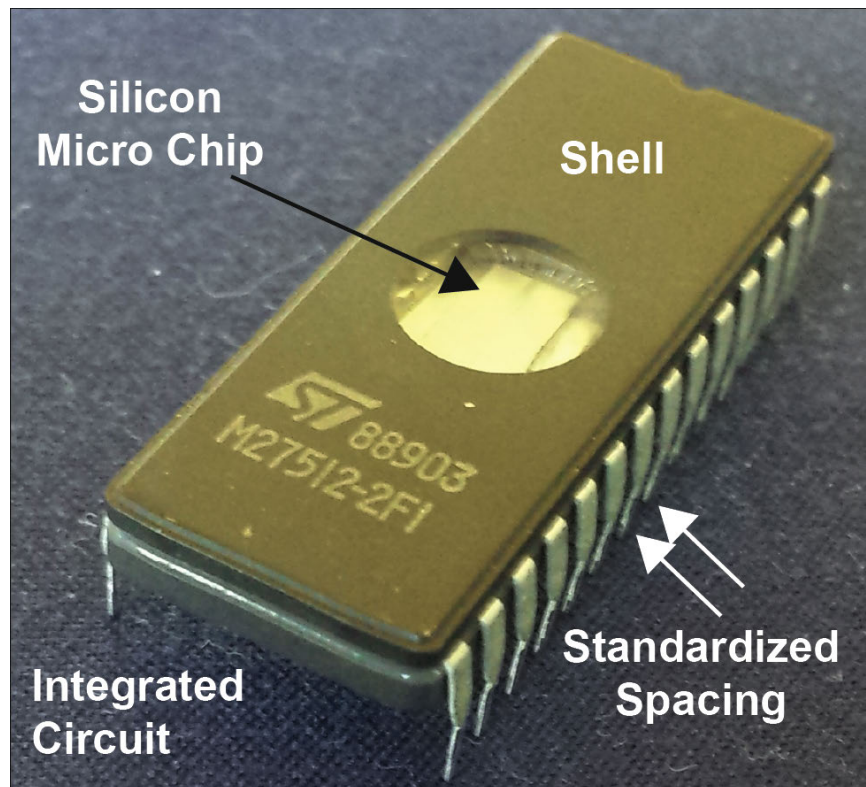


Figure 3.1: A key inspiration for the packaging methodology of our multifluidic tool set is the integrated circuit. The silicon microchip is surrounded by a shell. The shell holds the standardized spaced pins that facilitate connection of this part to other components

components often remains relatively constant and is designed for interfacing to other parts, a component's function can still be uniquely defined to fulfill a specific need. The shell is an important feature of the integrated circuit; it provides standardized input/output

(IO) interconnections, protects the internal parts, and enables flexibility in design and fabrication. Additionally, the shell is fabricated separately from the internal integrated circuit. Due to this separation of processes, parts such as the shell and standardized socket do not have to be re-fabricated with every revision of the component. Fabrication is thereby streamlined and costs are reduced. Figure 3.1 shows the shell that surrounds the integrated circuit. The socket that interfaces the integrated component to other macro-components can be standardized. In this manner, hierarchal packaging allows our MEC system to operate on multiple fluid scales (nanoliters to milliliters) with standardized interfaces for component-to-component compatibility.

Biological cells are packaged in a cell membrane that controls mass transport into and out of the cell, and electronic components like integrated circuits are typically packaged in cases with metal pins for input/output connectivity. In both cases, the contents of the package may vary widely in different cell or chip types, but the package itself remains relatively constant. By wrapping custom components in a standard package, components can be interconnected easily and efficiently. Similarly, in the MEC system a constant package design (Fig. 3.2) enables a limitless variety of different fluidic, electronic, optical, and mechanical components to be interconnected with each other.

3.1.4 Standardized interface spacing

In Table 3.1 the only evolutionary characteristic that one that the cell does not exhibit is standardized interface spacing. This brings into question whether standardized interface spacing really is an evolutionary characteristic. We define an evolutionary characteristic as one that allows a component to easily adapt to a wide range of applications

and easily evolve in a cost-effective manner. The cell can evolve to perform a wide range of functions, and with its non-standard interface, it can adapt very efficiently. The cost of adaptation is very low for the cell; it doesn't affect the cell's reproducibility. With the integrated circuit this is not the case, so a standard interface is important for reproducibility. Standardized spacing was crucial for the integrated circuit in making it a evolutionary component, because without standardized spacing, the reproducibility costs for new adaptations would have been too high. The limitations that standards induced were not sufficient to cause a problem. In contrast, cells are usually not handled and tested on an individual bases, so standards would have only posed a limitation. To me the cell's methodology is obviously better, but due to our fabrication, testing, processing costs, design costs, and documentation requirements, standards are still an important consideration for the MEC system. That being said, 3D printing opens the door to the possibility of more non-standard MEC components.

3.1.5 Fundamental elements

Both the integrated circuit and biological cell have fundamental elements. *Fundamental elements* are the most basic building blocks in a system; they cannot be broken down any further without a loss of function. In electronics a key fundamental element is the transistor, and in a biological system it is the cell. These basic parts are combined to form higher-level functions, which are then combined to form even higher functions. For example, cells combine to form tissue, then organs, and finally organisms. Ultimately, a wide array of applications (organisms) comes from this fundamental element. With electronics,

transistors are combined to form integrated circuits, integrated circuits are combined to form subsystems, and subsystems are combined to form instruments such as the computer.

The components within our micro-MEC library are the beginning of the definition of our fundamental elements. The level of success of our MEC system will depend on the correct selection and development of fundamental elements. One of the reasons the field of microfluidics has not evolved more quickly is the incorrect selection of its fundamental elements, valves. Microfluidic valves require interfaces to solenoid valves which are on a drastically different size scale than the microfluidic valve and require a separate fabrication process. This therefore limits the scalability of microfluidic valves. Instead, the field of microfluidics needs an electronically-controlled valve fabricated with a scalable process.

3.1.6 Interdisciplinary characteristics

Interdisciplinary characteristics are central to an evolutionary component's ability to solve a range of applications across multiple fields. For example, cells have the ability to adapt their characteristics to perform different functions within the human body. As a result, rod and cone cells perform optical sensing, muscle cells perform mechanics, and nerve cells control electrical signals. The diversity of electronic components further illustrates the value of interdisciplinary flexibility. Applications for electronic components range from optics (charge-coupled devices) to mechanics (solenoids) and acoustics (speakers and microphones). In other words, as a component set has more fields integrated into its design, its range of applications increases. Conversely, components that support fewer fields have limited applications: an integrated circuit board that lacks fluidic characteristics is barred from fluidic applications. Our evolutionary components incorporate electronics, mechanics,

optics and fluidics into a single family of components. These interdisciplinary characteristics can be integrated in a single device or can be incorporated within multiple devices (*i.e.* one device can incorporate multiple fields, or multiple compatible components can each incorporate a different field).

Scientific and medical instruments are rarely purely fluidic, purely mechanical, purely optical or purely electronic. Instead, they typically integrate several of these fields. By creating a family of integrated components we seek to create instruments with real world applications across a wide range of fields. For this reason, the MEC system includes components with fluidic, electronic, mechanical, and optical functionalities.

3.1.7 Mechanical 3D structure critical to function

As we developed our MEC tool set we recognized the importance of three-dimensional mechanical structure and interfacing. These insights were motivated by the importance of structures and interfaces in biological cells. We also had a clear understanding of the limitations that the lack of a 3D structure imposed on integrated circuits with its planar methodology. To obtain higher density of integration in an integrated circuit, finer and finer resolution is required. This trend continued until ICs were developed with traces that are only a few atoms across. The problem was not properly utilizing the third dimension. Single inline modules using surface-mount components were an attempt to alleviate this problem by better utilizing the third dimension, and I was a pioneer in this field. We also saw the utility of the cell having a mechanical structure adding to its functionality. In addition, unlike electronics, cells interface on all sides, increasing possible number of in-

terconnections. We combined all these motivations with an understanding of standardized spacing in integrated circuit packaging to produce the concept shown in Figure 3.2.

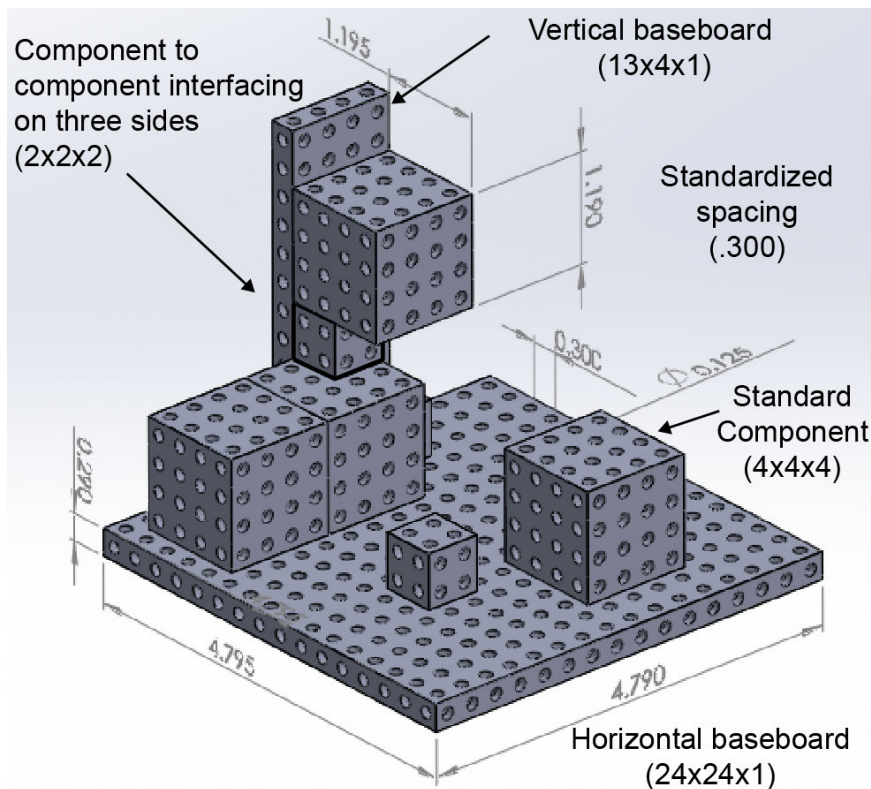


Figure 3.2: Spatial design standard for the MEC system. Standardized spacing allows MECs to be connected together in all three spatial dimensions, and interface points (circles) allow fluid, electricity, light, etc. to flow between MECs.

3.1.8 Standardized three-dimensional interfaces

Biological cells can communicate and exchange matter with their neighbors in all three spatial dimensions. Interestingly, electronic circuits are largely planar and communicate mainly in two dimensions, a limitation that electrical engineers have struggled with for the last half century [6]. To avoid the limitations of two-dimensional interconnectivity, the MEC system uses a three-dimensional interface standard shown in Figure 3.2. Consistently-

sized and regularly-spaced interface points (the circles in Fig. 3.2) can contain mechanical, electrical, fluidic, or optical interfaces. Fluid, light, and electric current flow from one MEC to another through these interfaces.

As shown in Fig. 3.2, the model MEC components have standardized interfaces on all six sides. The components have standardized sizing and spacing in all three dimensions. This simplifies packaging of these components and interfacing them together. With the MEC system only the mechanical interface has maintained the 0.3-inch Snap interface standard. In electronics, standards were easier to maintain because there was only one interface type, electrical.

Strict standards may not be the optimum methodology when dealing with several interface types. We are still early in the development of the MEC system, and the fact that we are starting with a more breadboard-like approach has reduced the need for standards on all interface types. It is interesting to see that MEC interface types are evolving. This was also seen in both electronics and cells. In electronics, for example, inline pin and surface-mount pin evolved for electrical connectivity. In the cell, examples of receptors would include receptor tyrosine kinases (RTKs) and G-protein-coupled receptors (GPCRs). With the MEC system we have a standard mechanical interface type with a small 0.125-inch-diameter pin. We let the design dictate how the standards evolve.

3.1.9 Component-to-component compatibility

Component-to-component compatibility means components can plug directly into or attach to other components. Cells provide for this; electronics almost always require an intermediate printed circuit board. The ability for component-to-component interfacing

will increase packaging density and provide for a higher number of interconnections per unit volume, which can result in much higher functionality. In the human body, cell grouping types evolved such as bone and muscle. These grouping types remained highly compatible. We are starting to see this with the MEC mechanical components and fluidic components. Because we maintain a mechanical standard, they remain highly compatible.

Chapter 4

Early MEC

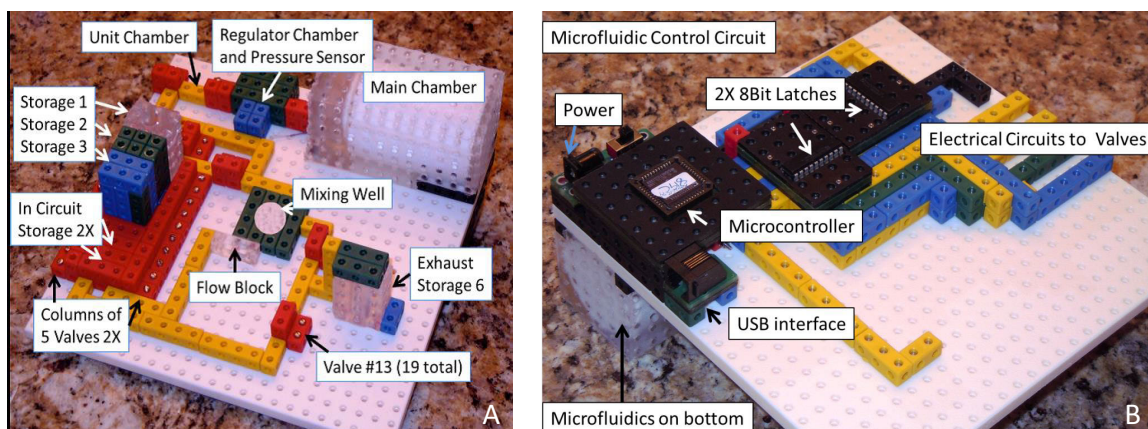


Figure 4.1: Photograph of the front side (A) of Snap mock-up of a MEC mixing system, with storage, valves, and mixing chamber. Backside (B) of the same mock-up, which shows the microcontroller-based control interface.

The MEC system is intended to be a group of components that work easily together and can adapt to a wide range of applications. Figure 4.1 is an early mock-up of what a MEC component system might look like for an automated mixing system. The mock-up was made from both working and non-working components. As the MEC system evolves we are striving to understand the evolution of the components, and what makes a component

successful in our MEC system. A successful component is one that is perdurable and is not replaced. It is important to realize that unsuccessful components may still provide critical information for the evolution of our system. For the system shown in Figure 4.1, very few of the components still exist, but many of their functions and features still exist. The mechanical baseboard is still being widely used. The mechanical snap blocks are still being used. The mechanical female-component-to-female-component interface, with a male pin interface, is still the primary mechanical interface. The standard 0.3-inch spacing is still the primary factor used in sizing components. A microcontroller is still used for controlling devices and interfacing with the personal computer via USB (although we have moved to a distributed microcontroller structure with a serial inter-component interface, which will be discussed later). The 2-56 screw is still being used as the primary method for firmly securing devices. The wall mount 5 V regulator is still one of the main power sources, although a 9 V supply is also typically used.

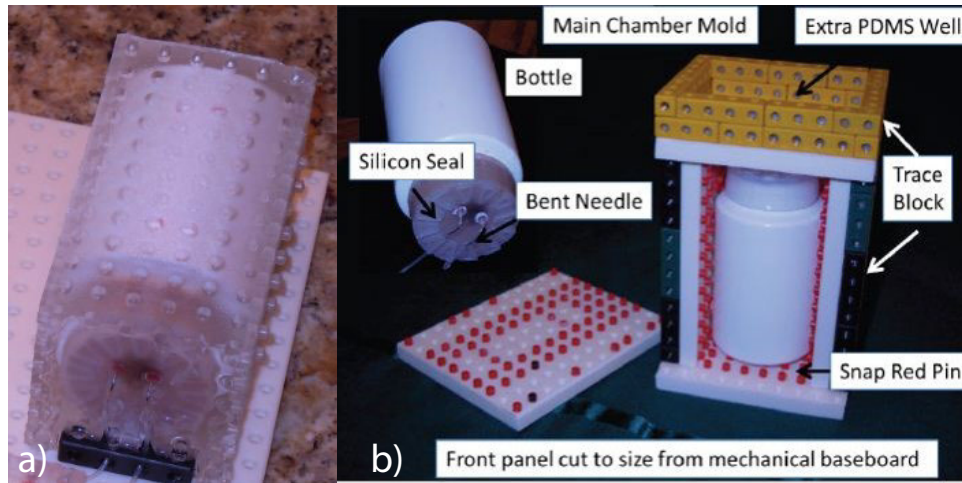


Figure 4.2: a) Air chamber made from a pill bottle. b) The PDMS casting mold that was used to make it mechanically compatible to the Snap MEC standard.

The air chamber (Fig. 4.2a) shown was a working device, made using the casting mold in Figure 4.2b. The air chamber was constructed using a bottle with bent needles epoxied into the lid. The bottle was then cast in PDMS to create the correct mechanical form factor for MEC components. A casting mold was made from Snap mechanical baseboard, red pins, and blocks. PDMS was used to fill the mold. Duck tape was used to seal the mold. A pool was designed above the mold to hold excess PDMS, so if material leaks it is replaced during the curing process. The red pins are used to form Snap holes in the air chamber so it can be mechanically mounted on the Snap baseboard. The overall sizing of the component is to Snap dimensions. Very small tubing attached to a stainless steel needle is used to connect between the air chamber and other devices. A similar process was also used to make the a working storage chamber. The casting process is labor intensive, but results in a part that is functional and fits in the MEC system.

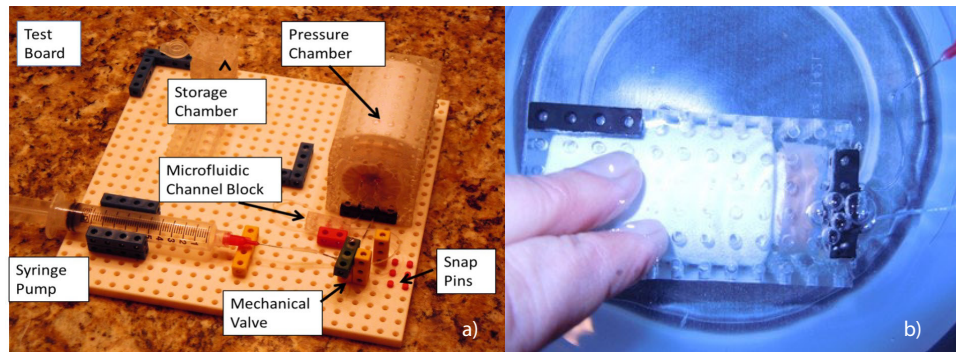


Figure 4.3: a) Air chamber test board. The test board includes a mechanical screw valve, syringe air pump, mechanical baseboard, and air chamber. b) Simple bubble test of the air chamber. Bubbles emerging indicate even the smallest of leaks.

Shown in Figure 4.3 are two test methods that were used for evaluating the air chamber. *Developing component testing procedures is an critical part of the MEC system development.* In the first method, air is pumped with a syringe through a mechanical valve

(Figure 4.4) into the air chamber. The valve is closed and the syringe is then reloaded with air, and the process is repeated. The valve is finally closed, trapping pressurized air in the chamber. This allows multiple volumes of air from the syringe pump to be loaded into the air chamber. The chamber was left overnight and then the valve was opened, releasing the air back into the syringe. When the syringe was full, the valve was turned off. The syringe was then emptied, and the process repeated. By comparing the number of syringe volumes of air that went into the air chamber to those that came out, we evaluated the ability of the air chamber to hold pressurized air. Because the plunger on the syringe has friction, not all of the pressurized air in the chamber is released, but the small fraction that is not released can be calculated. One problem that was identified with this system was the ease of clogging and the unreliability of the small tubing. *Evolution: this led us to a larger, more-elastic Tygon tubing (3/32 inch OD and 1/32 inch ID) and the 20-gauge stainless steel tubing interface.* The second (less quantitative) method for testing the air chamber is to place the chamber under water and pump air in with a syringe to look for air bubbles. Even the smallest of leaks becomes evident using this process. This is the testing method we use currently.

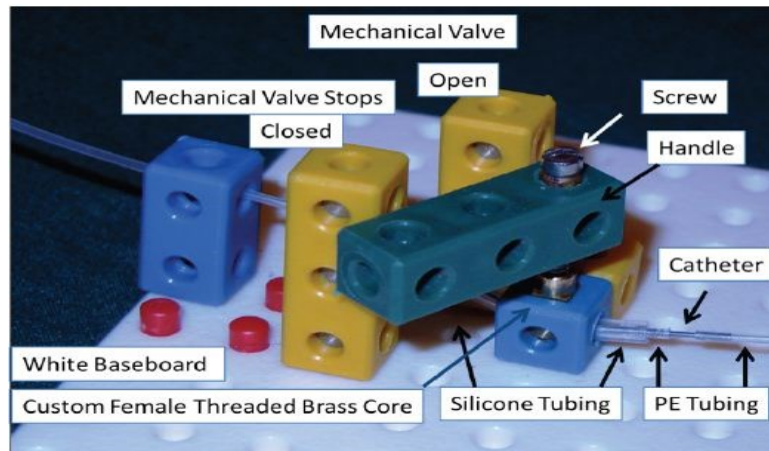


Figure 4.4: First functioning MEC mechanical screw valve, built from Snap blocks

The screw valve (Fig. 4.4) was made from Snap Blocks, a screw, a cover, and a nut. The screw pinched the tubing closed. The cover was used to protect the tubing from the sharp edge of the screw. It was an effective valve, but care was needed to not over-compress the tubing. A mechanical Snap block was used as a stop, but the stop was poorly secured and could fall off. The valve, although easy to build, was still time consuming. Therefore, a more integrated solution was needed. *Evolution: The tube pinch concept was effective way to create a valve, and offered isolation between the valve and the fluid it controlled.*

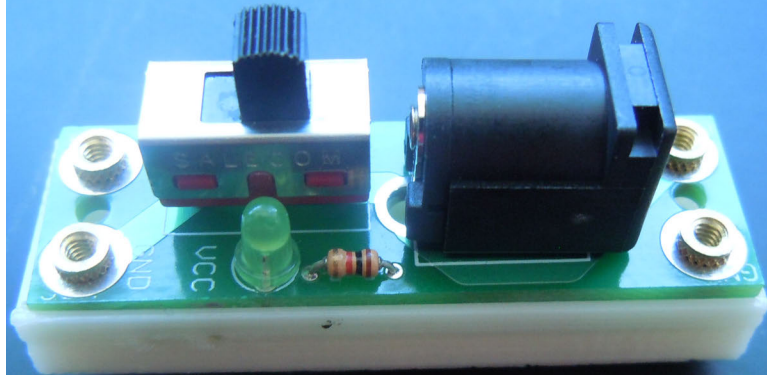


Figure 4.5: The power plug for interfacing electrical power from wall adapter to the MEC system. It includes a switch, adapter plug, and “power on” LED indicator

One of the few components in the mock-up to have a close relative in the current MEC system is the power plug. Shown in Figure 4.5 is what evolved from the early power plug used in this mock-up. Both plugs transfer power from the wall outlet to the system. They both use the same switch and power socket. Both have the ability to be screwed down using 2-56 screws for a secure mount. Two key differences between these power plug MECs are how the electrical interface to the system is made, and the process of fabricating the base the printed circuit board is attached to. The original Snap power plug utilized conductive pins to provide a direct component-to-component electrical interface with standardized spacing. Power to devices is then transferred through a power and ground plane. In our current power plug, power is transferred through wires that are either screwed down or plugged into side sockets. This represents a more breadboard-like approach where standardized electrical spacing is not required due to the flexibility of the wire. While the conductive baseboards were more complex and costly to fabricate, our current mechanical baseboards can be easily cut to whatever size is needed. This provides for rapid prototype development. So, in our current design requirements, which utilize large baseboards, our

mechanical breadboard methodology is preferable, and more cost effective. The original methodology removed the need for routing power and ground, which most electronic components need. It had a standardized power interface. It allowed for ease of component to component power interface. Both methods served a purpose, so which is better depends on the application. The power plug shown in figure 4.5 is a look in the future in regards to the chronological order of this dissertation. The printed circuit power adapter is integrated with a 3D printed base to provide it with the MEC standard mechanical interface. This allows it to easily mount to the MEC mechanical baseboard. This will prove to be an effective methodology in the future. The 3D-printed power plug base has gone through several iterations. *Evolution: 3D printed power plug base. Printed circuit board snap into place in the base. Wires attach to power plug, instead of conductive pins, to make the electrical connection*

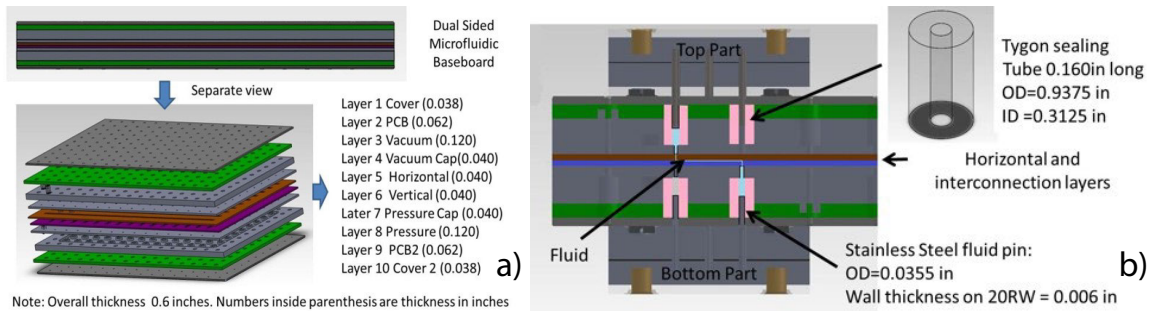


Figure 4.6: (a) Layers of a future MEC baseboard, which includes fluidic and electrical layers. It also includes vacuum and pressure chamber planes. (b) Cross section closeup of baseboard, showing component interfaces.

As a designer, sometimes you are allowed to see the future, or at least imagine it. A key element in electronics is the printed circuit board. Shown in Figure 4.6 is the equivalent for the printed circuit board for the MEC system. It expands from just interconnecting

electronic element to including fluidics. The concept for this evolves by first examining how the two current baseboard systems are fabricated, and thinking about how they can be improved. The current conductive baseboard is fabricated by sandwiching a punched laminate around a printed circuit board. The current mechanical baseboard is fabricated by machining plastic with a custom drill bit that creates the Snap hole. The printed circuit board provides power and ground planes for components and a method for interfacing a large number of electrical components. The concepts evolved into Figure 4.6. This MEC baseboard has two routing planes (also acting as power and ground planes) and a vacuum and pressure chamber plane. The pressure and vacuum planes act much as the power and ground planes in printed circuit boards. They provide the energy for driving fluids between components. Today, this could be fabricated by laminating plastic and printed circuit board layers together. The utilization of a separate fabrication process for the fluidic and electronic parts and the subsequent lamination would make this an expensive process. 3D printing in the future will hold the key to this fabrication problem. It would allow for the fabrication of this MEC baseboard in a single process. 3D printing currently has the ability to dispense different materials, but this level of sophistication is only just now beginning to become commercially available. In addition to simply routing electrical and fluidic channels made with hard material, 3D printing would also have an elastic material that it could print. This would provide the ability to fabricate fluidic sockets directly in the baseboard and valves, as shown.

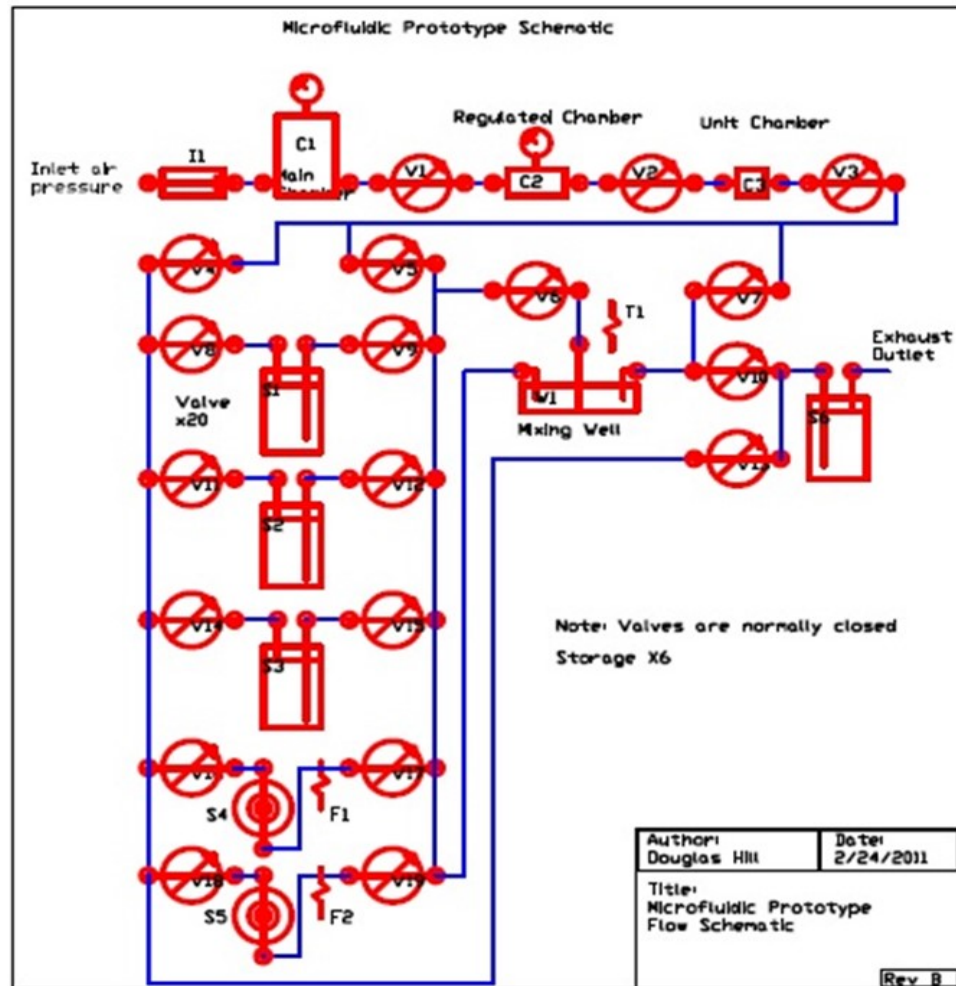


Figure 4.7: Fluidic schematic for mock-up MEC mixing system

Shown in Figure 4.7 is a schematic for the mock MEC mixing system. Symbolic design is an important aspect of component design. It allows for definition of a design for communication, with easy translation and modification. Adobe Illustrator was used for many of our early schematics. This was a purely fluidic schematic, and the electrical control lines that are needed for turning off and on the mock-up of the electrically controlled valves are not shown. The mock design assumes a family of fluid-carrying Snap blocks will be

developed, just as was done with electrical Snap blocks. The schematic shows several air chambers, storage units, and valves. The symbols directly relate to physical components. The storage unit and air chamber were working components, but the electrically controlled valve was not. All components were of the MEC family and could easily plug together for both fluidic and mechanical connections. The schematic can also act as a type of “to do list.” Here, it shows that the mechanical fluidic valve needs to be developed. Drawing traces over other traces with Adobe Illustrator was problematic, so further development was needed (described in later chapters).

Chapter 5

Developing an early MEC scripting language and quantifying fluorescence with a microscope

In this chapter we present an imaging method for quantifying fluorescence by using exposure time and a microscope. In addition to satisfying the practical need for quantifying fluorescence in research, this experimental method also serves as a model system for demonstrating an evolving experimental MEC scripting language for processing and analyzing data. The processing and analysis of large amounts of data, and its seamless flow into the cloud, will be critical for MEC system development. The scripting language described here allows for the processing of large quantities of images, and text-based data by Matlab, without the need for the user to understand the underlying programming language [14]. In addition to guiding the processing of data, the scripting language also defines and

produces the plots that are required. The images that are being processed are of individual wells in a 384-well plate. The wells contain proteins at various concentrations that have been tagged with the fluorescent protein YPet. In addition to automating the image processing, we are also automating the method of taking images. This was done by the development of fluorescent pins which mark the boundaries of the plate's rows and columns. For this experiment we are using a Nikon inverted microscope with a programmable XY stage. By determining the position of the two fluorescent pins, we are able to extrapolate the location of all the wells of interest and automatically program them into the Nikon microscope. Additionally, we are using a multi-well plate reader (Molecular Devices Flexstation) to take fluorescent images of the 384-well plate and verify our methodology. Furthermore, the scripting language was able to read and process the text data from the Flexstation.

Evolution: Two fluorescent pins allow for the automated programming of the imaging well locations of a 384-well plate.

Imaging is fundamental for the diagnosis of disease, and as video imaging progresses, it will become an increasingly powerful tool in the health care industry. As a bioengineering tool, imaging and video imaging can be used for the characterization of protein interactions. Proteins of interest can be tagged with YPet and CYPet and their interaction monitored by changes in the Förster Resonance Energy Transfer (FRET), both *in vivo* and *in vitro*. This method begins with the quantification of protein concentration by fluorescence. A standard method of characterizing protein dissociation constants with fluorescence has been demonstrated with a multi-well plate reader (Molecular Devices Flexstation) [15]. The problem was that this is a closed system that requires an expensive piece of

equipment. In addition, with the standard method one cannot dynamically measure protein interaction during the mixing process. As I developed the MEC system, I looked for ways of translating existing methods into methods that can be adapted to the MEC component-based system. Fluorescence quantification by Flexstation is one of those methods that is incompatible with the MEC system. To overcome this I developed an alternative method with microscope imaging.

When utilizing imaging for quantification of fluorescence one faces many challenges. The first challenge that my method overcomes in using fluorescent imaging is the limited range of concentration that can be measured at any specific exposure time. Typically, imaging intensity is used for measuring concentration at a set exposure time. The problem was that at low concentration the noise floor of imaging was reached and at high concentration the CCD pixels are saturated. What I recognized was that when not at extremes, the intensity is directly related to the exposure time. By using a range of exposure times and mathematically holding intensity constant at approximately the midpoint (i.e. 125 for 255 range of a JPG image), one may measure concentration over a wide range. The next challenge that my method of using a range of exposures exacerbates was the number of images that are required to be processed. The number of images will only increase as we move to dynamic monitoring. In addition, if we wish our method to remain flexible, repeatable and accessible, MATLAB presents a barrier. The learning curve for its use in image processing can be prohibitive. Dedicated software can be used, but this would quickly limit a researcher's range of investigation. What I have developed is a scripting language that can both handle the processing of a large quantity of images, but also insulate the

user from the need to learn MATLAB [14]. The scripting language is being designed to handle different data and imaging types. The structure of the MEC scripting language will facilitate a broad range of applications and a natural increase in its functionality. Scripting languages are a way to lower the programming barrier for the user, while still maintaining flexibility, and are a critical part of the MEC system development.

There are many challenges in utilizing fluorescence to quantify protein concentration that are not covered in this chapter. First, quantifying fluorescence was challenging because controlling all the parameters of the equipment that affect the fluorescence reading was challenging [16]. This was why, on equipment such as the Molecular Devices Flexstation, they use Relative Fluorescent Units (RFUs). Another problem was that the Beer-Lambert method can be used to find the overall concentration of a sample, but that concentration may not accurately reflect the concentration of active protein due to denaturing or impurities. Finally, human error presents additional unknowns in experimentation. It should be noted that our automated method of well location helps reduce human error in finding the center of the wells, and therefore could affect well intensity readings. The scripting language cannot eliminate these problems, but it provides a platform for addressing these problems and quantifying their significance.

5.1 Summary of experimental method

The objective of this experiment was to measure the concentration of UBC9 tagged with YPet ranging from 0.125 μM to 8 μM . Row A in the well plate contains nine column concentrations ranging from 0.125 μM to 8 μM , with a Tris buffer control well. 30 μl was

used in each well. Row B mirrors row A. Concentration of UBC9-YPet was originally measured using the Beer-Lambert method (absorbance). Fluorescence, then is correlated with concentration, and was measured by two methods; first, by readings from a multi-well plate reader (Molecular Devices Flexstation). This was used to verify our imaging results. Second, fluorescent intensity was measured at a range of exposures using an inverted microscope with a programmable XY stage. The experimentation scripting language handles the information from raw data through analysis, and finally, to the display of results (typically in graph form). The text based scripting language was translated by our interpreter, which was written in MATLAB. These two different data types will demonstrate the flexibility of the scripting language.

During experiment 1, the automated well location method had not been developed. The automated well location method works by the use of a fluorescent pin that fits into a well. For example, in an experiment calling for a 10 x 10 array of wells, instead of programming all 100 wells, a fluorescent pin would be placed in the upper left and lower right corner. One would then manually find the center of these two pins with the motorized Nikon XY table and save the XML file. This file would then be interpreted by MATLAB, generating a new XML file with the one hundred points.

5.2 Experimental results

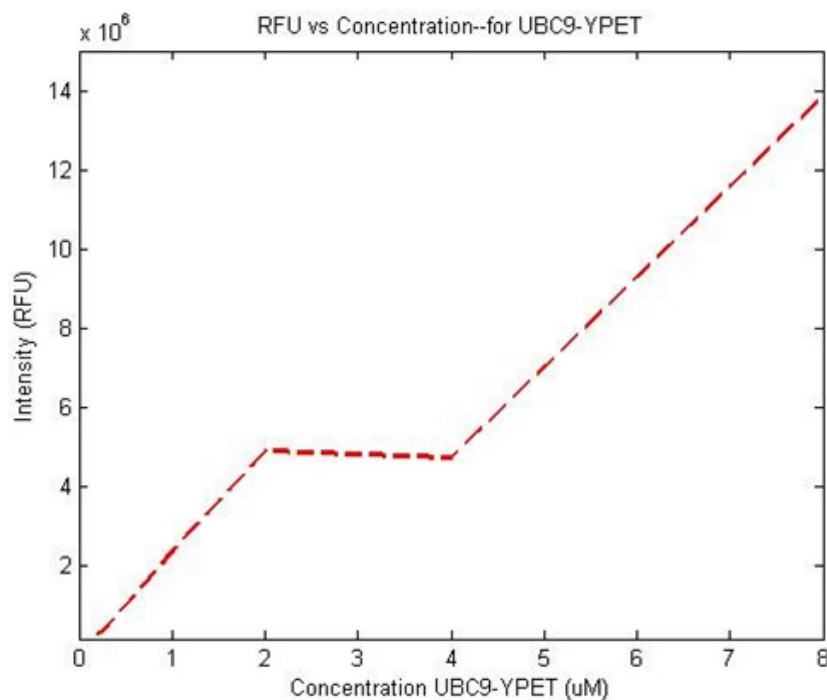


Figure 5.1: Data from the Flexstation comparing concentration of UBC9-YPET to intensity in Relative Fluorescent Units (RFU) at 530nm wavelength.

Data was first collected using the Flexstation. This data, at 530 nm, was used by scripting language to produce Fig 5.1. This gives a comparison of intensity in RFU units to UBC9-YPET concentration, which was previously determined using the Beer-Lambert method.

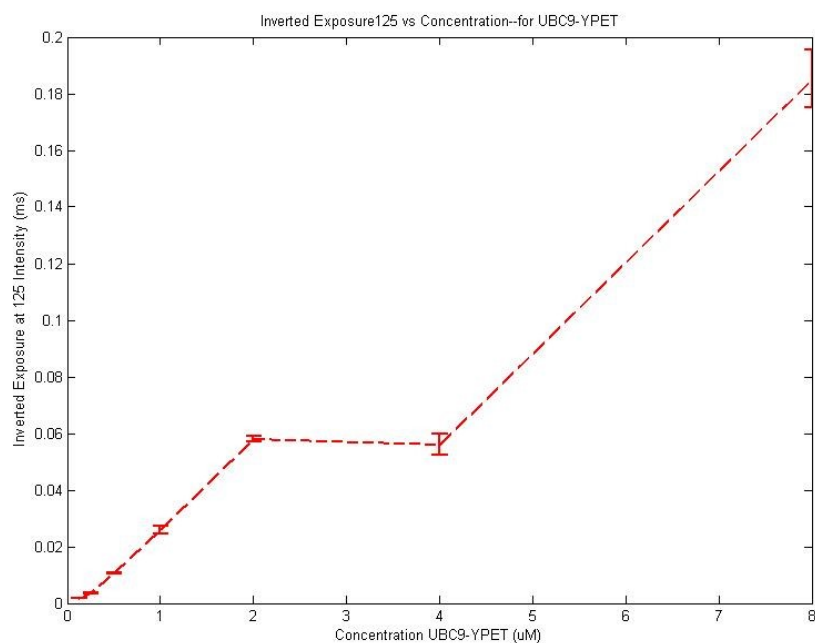


Figure 5.2: Data from the Nikon Eclipse Inverted Microscope comparing concentration of UBC9-YPET to inverted exposure time (1/ms) using the FITC filter.

Next, data was collected by fluorescent imaging; 183 images in all were taken in this set of experiments using the Nikon Eclipse Inverted Microscope. The data, which was taken using the FITC filter, was translated by the scripting language to produce Fig. 5.2. This gives a comparison of inverse exposure time to UBC9-YPET concentration (previously determined). Visually, the two graphs are very similar. We surmise that the flat line in the graphs shown in Fig. 5.1 and Fig. 5.2 was from human error.

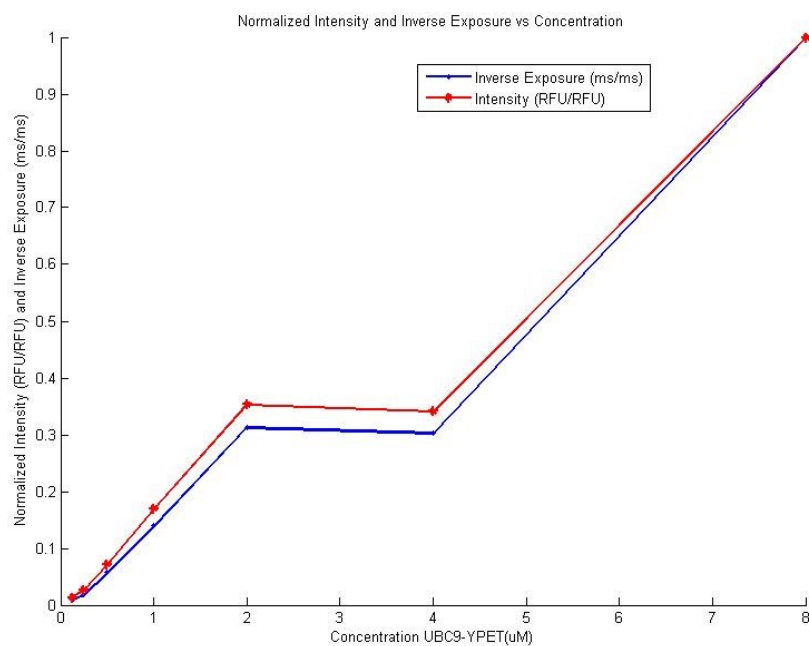


Figure 5.3: Graphical comparison between inverse exposure from the Nikon and Intensity from Flexstation.

Fig. 5.3 was a comparison of the two methods. It validates the inverse exposure method of measuring concentration. This figure shows that a wide range of concentration can be read without pixel saturation. We recognize the difference between the two methods; one being measured by imaging with a wider range of excitation and emission recording; the other being with a photomultiplier tube. The correlation between the two graphs was exciting. One additional experiment was run in which we were able to run 2268 images. This was achieved because of our automated method of finding XY locations.

5.3 Methods

5.3.1 Fluorescent Pin

Fluorescent Pin

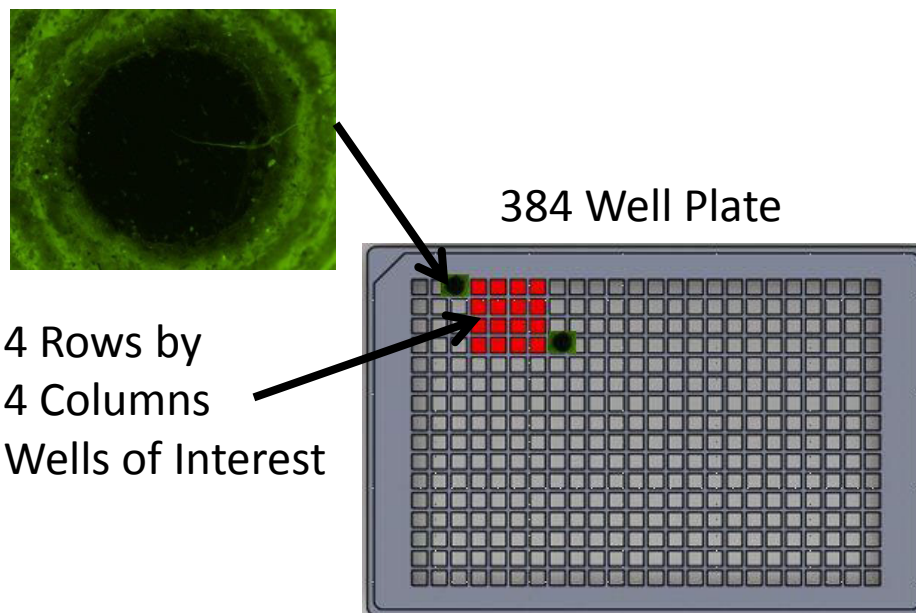


Figure 5.4: Image of fluorescent pin shown at top. Fluorescent pins are placed in upper left corner and lower right corner of wells of interest.

The fluorescent pin is an invention to automate the determination of the XY locations of the wells or locations of interest. The fluorescent pin prototype was made by wrapping paper and double-sided tape around a small nail. Then, the tip was ground flat. The paper was unwound until it just fits into a single well in the 384-well plate. As seen in the top of Fig. 5.4 the paper fluoresces and the nail provides a dark alignment target in the middle. A small string was added to the back, making it easy to remove and reuse. After the wells have been filled with the material(s) of interest (Experiment 1: $30\mu\text{l}$ of UBC9-YPet at several concentrations), the XY determination was done in four steps:

1. Pins are placed on the upper left corner and the lower right corner of the rows, as shown in Fig. 5.4.
2. The motorized stage and snapshots of the Nikon are used to find the location of the center of the pin. This was repeated for both pins and the locations are saved in XML format. I align with the eyepiece, but use snapshots to insure we are at the center of the pin. The focusing of pins also sets the Z plane to the bottom of the plate. Consistent control of the Z plane was also critical to this type of experiment.
3. Next, 2 point XML code shown at the top of Fig. 5.5 was loaded into a translation folder. Nikon calls the file “multipoints.xml” and “Exp1” was added to name.
4. Special Matlab code: XYREADnWrite.m reads XML format and then generates XML coded with all the rows and columns defined, as also shown in Fig. 5.5. The name generated for the file will be XMLExp1.xml corresponding to the specific experiment. This file was loaded back into the folder where “multipoints.xml” file had been placed and the points uploaded back into the Nikon Eclipse microscope. The experiment was now ready to run. The microscope XY stage will now go to all desired locations on the 384 well plate.

Prior to this method, individual points had to be programmed into the Nikon. As with most manual operations, this was not only time-consuming, but also could introduce alignment errors if the edge of the wells were in the image. Even better alignment can be made if MATLAB code was written to determine the error in the alignment snapshots taken in step two and could automatically adjust for the error. I have not found this necessary for

these experiments, but as this methodology was moved over to microfluidics and a location map was derived from the CAD software, tighter tolerances may be required.

```

<?xml version="1.0" encoding="utf-8" ?>
- <variant version="1.0">
- <no_name runtime="CLxListVariant">
  <bIncludeZ runtime="bool" value="true" />
  <bPFSEnabled runtime="bool" value="false" />
  - <Point00000 runtime="NDSetupMultipointListItem">
    <bChecked runtime="bool" value="true" />
    <strName runtime="CLxStringW" value="#1" />
    <dXPosition runtime="double" value="47238.659999999996000" />
    <dYPosition runtime="double" value="33606.199999999997000" />
    <dZPosition runtime="double" value="8515.164999999999100" />
    <dPFsoffset runtime="double" value="-1.0000000000000000" />
  </Point00000>
  - <Point00001 runtime="NDSetupMultipointListItem">
    <bChecked runtime="bool" value="true" />
    <strName runtime="CLxStringW" value="#2" />
    <dXPosition runtime="double" value="42727.119999999995000" />
    <dYPosition runtime="double" value="33606.199999999997000" />
    <dZPosition runtime="double" value="8515.164999999999100" />
    <dPFsoffset runtime="double" value="-1.0000000000000000" />
  </Point00001>
</no_name>
</variant>

```

↓

```

<?xml version="1.0" encoding="UTF-16" ?>
- <variant version="1.0">
- <no_name runtime="CLxListVariant">
  <bIncludeZ runtime="bool" value="true" />
  <bPFSEnabled runtime="bool" value="false" />
  - <Point00000 runtime="NDSetupMultipointListItem">
    <bChecked runtime="bool" value="true" />
    <strName runtime="CLxStringW" value="#1" />
    <dXPosition runtime="double" value="372079.0999999999980000" />
    <dYPosition runtime="double" value="219843.1000000000010000" />
    <dZPosition runtime="double" value="210885.3999999999990000" />
    <dPFsoffset runtime="double" value="-1.0000000000000000" />
  </Point00000>
  - <Point00001 runtime="NDSetupMultipointListItem">
    <bChecked runtime="bool" value="true" />
    <strName runtime="CLxStringW" value="#2" />
    <dXPosition runtime="double" value="683869.6999999999950000" />
    <dYPosition runtime="double" value="903264.3000000000050000" />
    <dZPosition runtime="double" value="210885.3999999999990000" />
    <dPFsoffset runtime="double" value="-1.0000000000000000" />
  </Point00001>
</no_name>
</variant>

```

↓

```

- <Point00016 runtime="NDSetupMultipointListItem">
  <bChecked runtime="bool" value="true" />
  <strName runtime="CLxStringW" value="#17" />
  <dXPosition runtime="double" value="15657.8800000000005000" />
  <dYPosition runtime="double" value="29127.7000000000001000" />
  <dZPosition runtime="double" value="8515.164999999999100" />
  <dPFsoffset runtime="double" value="-1.0000000000000000" />
</Point00016>
- <Point00017 runtime="NDSetupMultipointListItem">
  <bChecked runtime="bool" value="true" />
  <strName runtime="CLxStringW" value="#18" />
  <dXPosition runtime="double" value="11146.3400000000004000" />
  <dYPosition runtime="double" value="29127.7000000000001000" />
  <dZPosition runtime="double" value="8515.164999999999100" />
  <dPFsoffset runtime="double" value="-1.0000000000000000" />
</Point00017>
</no_name>
</variant>

```

Figure 5.5: This shows the translation of the two-point XML file to the 18-point XML file which corresponds to the wells of interest.

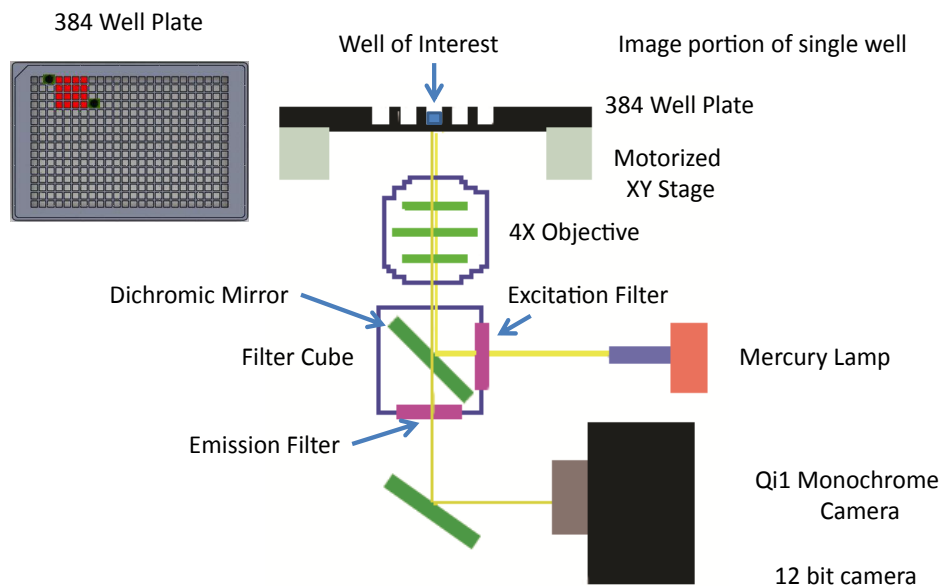


Figure 5.6: Basic set up of the Nikon inverted microscope. The excitation filter determines range of light to hit the sample. The emission filter determines the range of light measured from the sample.

The Nikon Eclipse inverted microscope was being used to determine the fluorescent intensity in individual wells of a 384-well plate. Its basic operation was shown in Fig. 5.6. The Qi1 monochrome 12-bit camera was being used to record the fluorescent intensity. Excitation wavelengths are adjusted by changing the filter cube, and therefore, the excitation wavelength.

FRET 530 NTE Cube Set :
 Excitation Filter=FF01-425/26-25
 Emitter Filter= FF01-529/28-25
 Dichromic Mirror= FF458-Di-25/36

FRET 475 NTE Cube Set :
 Excitation Filter=FF01-425/26-25
 Emitter Filter= FF01-475/28-25 Range (461-489)
 Dichromic Mirror= FF458-Di-25/36

FITC NTE Cube Set:
 Semrock: FITC-3540C-NTE-Zero
 Excitation Filter: 482/35 Range (464-500)
 Emitter Filter: 536/40 Range (516-556)

DAPI NTE Cube Set:
 Semrock: DAPI-5060C-NTE-Zero
 Excitation Filter: 377/50 Range(352- 402)
 Emitter Filter: 447/60 Range(417- 477)



Figure 5.7: Specifications for four filter cubes are shown. “425/26” means the excitation wavelength ranges from 412 to 438 nm. The diagram was supplied by Semrock and shows the optical density for excitation (FF01-425/26-25) and emission (FF01-475/28-25).

Fig. 5.7 lists four of the filter cubes that are available for these experiments. The originating light source for this system was a mercury lamp. For experiment 1 the FITC filter was used with an excitation range of 464 nm to 500 nm. This was in sharp contrast to the Flexstation, where very narrow excitation wavelength was possible at 475nm. Our results demonstrate that this wider excitation range was not a problem. We used an objective of 4x to provide the largest signal in this system, and it still fit easily into a single well. The emission filters control the wavelength of light that was recorded from the sample.

For Experiment 1 we utilized the FITC filter cube, which was well suited for the YPet fluorescent protein we are using. YPet protein has an excitation centered at 475 nm (fitting well into FITC range of 464 to 500 nm of the cube) and an emission center of 530

nm (fitting well into the emitter filter range of 516nm to 556nm). Optical density (OD) is related to transmitted light (T) by equation 5.1:

$$OD = -\log_{10}(T) \quad (5.1)$$

Ideally, it was desirable to have an optical density greater than 5 in areas where the light was being blocked.



Figure 5.8: The methodology for finding the effective exposure at constant intensity of 125. For Well A1 the E₁₂₅ = 5.1 ms.

Fig. 5.8 Shows samples of images taken of the same well (A1) with the FITC filter at different exposure times. For this experiment we are only measuring intensity, and there was no feature information. Typically, when quantifying fluorescence, intensity is used.

$$\frac{I_1}{E_1} = \frac{I_2}{E_2} \quad (5.2)$$

Equation 5.2 gives the relationship between intensity and exposure as long as one is in the linear range. If we make I_2 constant at 125 (Format JPG range 255), then at an exposure of 3 ms we predict $E_{125} = 5.0$ ms.

This can be repeated for each of the 4 images. By selecting the two images with intensities closest to or bounding 125, a value can be predicted. This method eliminates the saturation problem, as long as one does not exceed the limits of the equipment, at a cost of more images.

5.4 Experimental Scripting Language

Imaging Script:

```

SETVALUE:EXPID=1
%Don't put quotes around strings
SETVALUE:StartRow=A
SETVALUE:EndRow=B
SETVALUE:StartColumn=1
SETVALUE:EndColumn=9
SETVALUE:FilterTypes=FITC
%SETVALUE:FilterTypes=FITC,DAPI,FRET530

SETVALUE:TotalExp=54
EXP:1 2 3 4 5 6 7 8 9
EXP:10 20 30 40 50 60 70 80 90
EXP:100 150 200 250 300 350 400 450 500
EXP:550 600 650 700 750 800 850 900 950
EXP:1000 1500 2000 2500 3000 3500 4000
EXP:4500 5000 5500 6000 6500 7000 7500
EXP:8000 8500 9000 9500

%UBC9-YPET 6.8mg/ml Stored -80 Degree C
%um= microMolar Unit Must come before
UNIT:uM
WELL:A1=UBC9-YPET(8)
WELL:A2=UBC9-YPET(4)
WELL:A3=UBC9-YPET(2)
WELL:A4=UBC9-YPET(1)
WELL:A5=UBC9-YPET(.5)
WELL:A6=UBC9-YPET(.25)
WELL:A7=UBC9-YPET(.125)
WELL:B1=UBC9-YPET(8)
WELL:B2=UBC9-YPET(4)
WELL:B3=UBC9-YPET(2)
WELL:B4=UBC9-YPET(1)
WELL:B5=UBC9-YPET(.5)
WELL:B6=UBC9-YPET(.25)
WELL:B7=UBC9-YPET(.125)
%mM= milli Molar
UNIT:mM
WELL:A8=TRIS(5)
WELL:A9=TRIS(5)
WELL:B8=H2O
WELL:B9=H2O

```

Flexstation Script:

```

SETVALUE:EXPID=1
%Don't put quotes around string
SETVALUE:StartRow=A
SETVALUE:EndRow=B
SETVALUE:StartColumn=1
SETVALUE:EndColumn=8
SETVALUE:Startwave=495
SETVALUE:Endwave=555
SETVALUE:Step=5
SETVALUE:EXRepeat=2
SETVALUE:Excitation=475

%UBC9-YPET 6.8mg/ml Stored -80
%um= microMolar Unit Must come
UNIT:uM
WELL:A1=UBC9-YPET(8)
WELL:A2=UBC9-YPET(4)
WELL:A3=UBC9-YPET(2)
WELL:A4=UBC9-YPET(1)
WELL:A5=UBC9-YPET(.5)
WELL:A6=UBC9-YPET(.25)
WELL:A7=UBC9-YPET(.125)
WELL:B1=UBC9-YPET(8)
WELL:B2=UBC9-YPET(4)
WELL:B3=UBC9-YPET(2)
WELL:B4=UBC9-YPET(1)
WELL:B5=UBC9-YPET(.5)
WELL:B6=UBC9-YPET(.25)
WELL:B7=UBC9-YPET(.125)
%mM= milli Molar
UNIT:mM
WELL:A8=TRIS(5)
WELL:A9=TRIS(5)
WELL:B8=H2O
WELL:B9=H2O

```

Figure 5.9: The Imaging and Flexstation configuration scripts. These scripts define the layout of the wells. In addition, they defines the filters or wavelengths that will be used. For imaging, the script also defines the exposures.

We are developing a dedicated experimental scripting language. *Evolution: MEC scripting language for the processing and analysis of data, allowing for an increased throughput, analysis and organization of information.* The first task of the scripting language was to translate raw data into a data file. This was one of the more critical operations, for it forms the bridge between equipment and the processing Matlab software. Special commands will often need to be developed for each new raw data type. With the 384-well plate,

the software needed to understand rows and columns. With imaging, it has to know the filter cube type. With the Flexstation, the software needed to know the wavelength and steps between them. In addition, with our new methodology of using exposure times to quantify concentration, the set of exposure times was needed in the imaging script, but not in the Flexstation script. All of these items can be found in the configuration scripts shown in Fig. 5.9. What Matlab does with the text script is parse it into fields. An example is given for the “Well:” command “Well:A2=UBC9-YPET(4)”:

1. “Well” Command
2. “:” Defines Command Field
3. “A” Row of Well
4. “2” Column of Well
5. “=” Define Row Field
6. “UBC9-YPET” Material
7. “(” Define Material
8. “4” Concentration
9. “)” Define Concentration Field

Note, each character and the order of characters is critical. In this way, the scripting language is very unforgiving. If one were to change a letter in material from upper to lower case, the script would think that you had a new material.

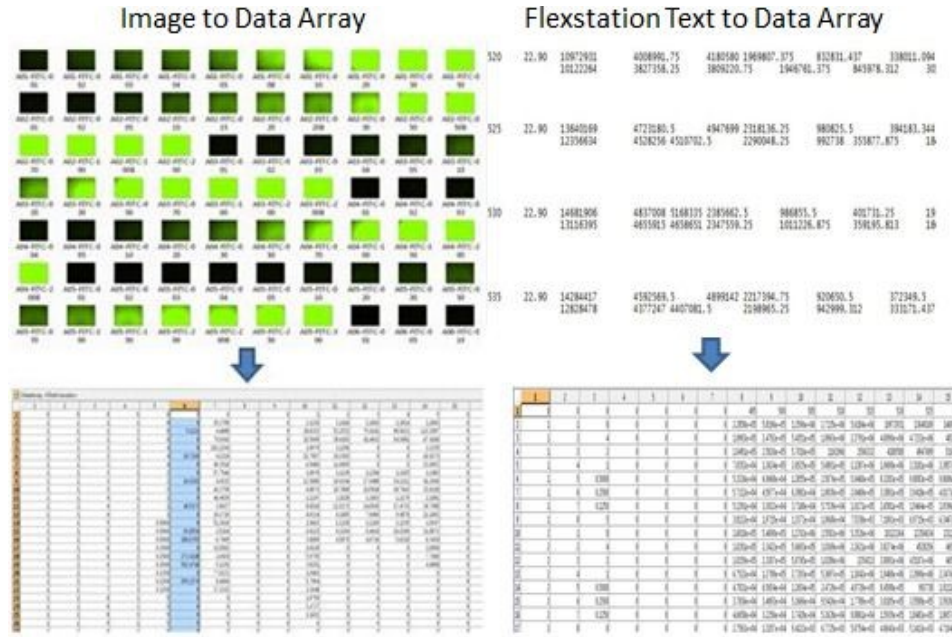


Figure 5.10: The first function of the script is to translate the raw data from the device to the data array. Above is shown the two data types the scripting language has been designed to work with, which is images and text.

The MATLAB interpreter reads the script and translates the information into a data array, as shown in Fig. 5.10. For the image array, some additional processing is required to get the average intensity per pixel for each image. Once all the intensity values at each exposure are in the data array, some post processing is also required to determine the E125 value, as described earlier. These are placed in the column in blue. There was a constant which was currently set to 10%, determining how close the E125 needs to be to its boundaries before it was rejected.

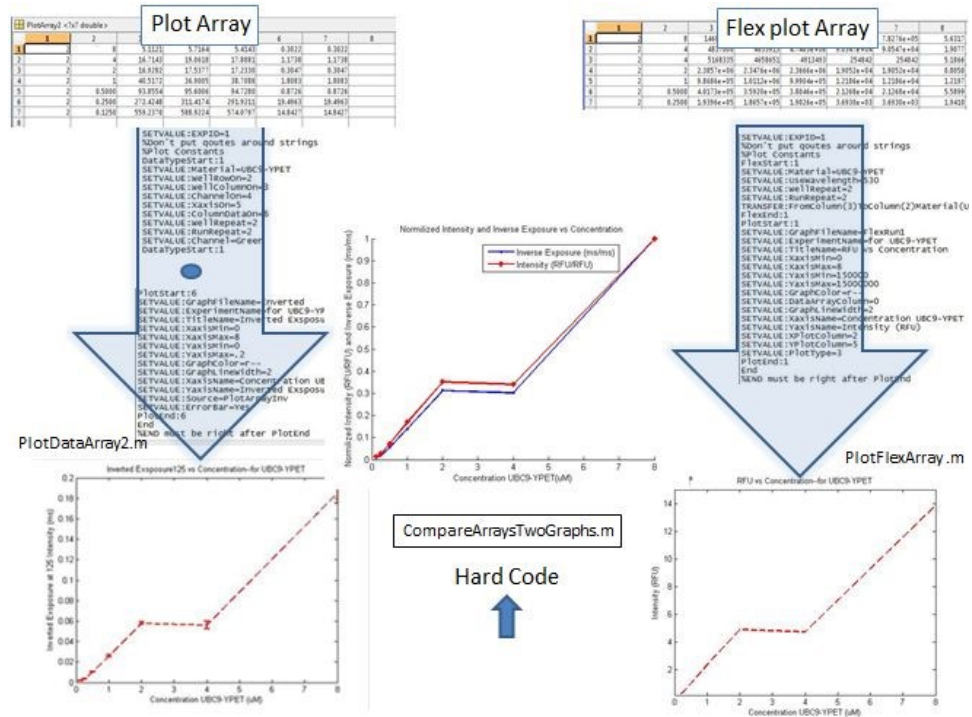


Figure 5.11: Plot Script was used to translate data from the data array to a graph. First the script defines how the data will be filter, and produces a plot array.

After the data arrays are generated, they are ready for post processing. For experiment 1, little post processing is required beyond what is mentioned above. A plot script language has also been developed and is shown over the arrows in Fig. 5.11. The first task of the script was to define the data that was desired from the data array. For example, for the graphs shown in Fig. 5.11 we filtered by:

1. SETVALUE:Channel=Green % Green Channel
2. SETVALUE:Material=UBC9-YPET %Only
3. SETVALUE:WellRepeat=2 %Row A=B
4. SETVALUE:UseWavelength=530 %Flex only

There are three color channels, but we are only using data from the green channel. There are three materials used for experiment 1; this script says to only use UBC9-YPET in data processing.

5.5 Conclusion for MEC scripting language, and quantifying fluorescence

The fluorescent imaging methodology demonstrated here will be an important tool in the MEC platform. It allows for dynamic mixing and characterization of protein-protein interactions [17]. Protein-protein interaction is the language of the body. Förster Resonance Energy Transfer (FRET) is a powerful method for characterizing these interactions. Closed systems like the Flexstation, although powerful, still limit FRET from reaching its full potential. Protein interactions are highly dynamic, and a closed system like the Flexstation has difficulty capturing those dynamics. Therefore, we began by developing a method for quantifying fluorescence with images. Moving to imaging opens up many possibilities, yet it also presents some significant hurdles. The rate of progress is determined by the tools. We have therefore aimed to make our fluorescent quantification method highly automated, both in the image acquisition and analysis. Scripting languages have proven to be invaluable in industry, with common languages such as Maya (Autodesk) and Python used every day. Yet specialized languages such as BioStream1 [18], used in microfluidics, are still in their infancy. Even if we ignore the potential for our experimental scripting language outside of our lab, or the fact that it helps to lower the barrier to image analysis to non-MATLAB uses, it was still essential to our research because of the time it saves. In our fourth imaging

experiment we took 2,268 images. This was only a small fraction of what we feel was necessary to properly characterize just the interaction of Sumo1 and UBC9. The scripting language automates the image processing. From experiment 1 to experiment 4 we achieved an order-of-magnitude increase in the number of images taken. Automation came in two parts; first, in the method of collecting data, which starts by automatically determining the XY location of the wells; and second, by the development of the scripting language for rapid processing of data. We also showed that the use of an effective exposure time, at a constant intensity, was a viable alternative to measuring fluorescent levels. It greatly reduces the saturation problem and was only limited by the physical limitations of the machine. *Evolution: Quantifying fluorescents by use of exposure time, with a microscope.*

Chapter 6

How to fabricate a microfluidic device starting with symbolic representation

Much of my early work focused on microfluidics. Most labs draw up their microfluidic chips using AutoCAD or other CAD programs and then fabricate them using glass etching or PDMS casting. Here again, I went back to my printed circuit board background and felt that the microfluidic design process should start with symbolic design. I wanted to make a simple system that mixes two compounds, and then image the mixing well to measure a fluorescence resonance energy transfer (FRET) signal. I designed the PDMS-cast microfluidic module to be sealed to a glass slide using a plasma treatment. I saw the possibility of using PCB design software to make a microfluidic casting mold. Numerous types of PCB design layout software exist. I decided to use Eagle (Easily Applicable

Graphical Layout Editor) for its ease of use, free download, and full online tutorials. The software package includes schematic capture, board layout, design rule check and even an auto router. It has the ability to work at the micrometer scale. The artwork produced from Gerber files (made by Eagle software) would be equivalent to what was typically developed in AutoCAD for SU8 silicon fabrication. In our case, the Gerber file can be used to generate artwork or CAM files. *Evolution: Creation of microfluidic symbols and use of schematic to directly design microfluidic chips. This was a needed advancement over the current industry methods.*

6.1 Start by creating a symbolic table, and from that a schematic

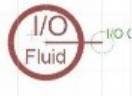
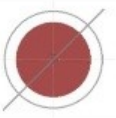
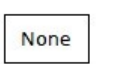
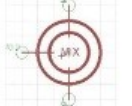
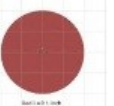
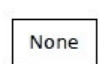
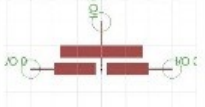
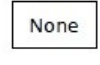
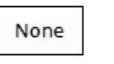
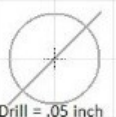
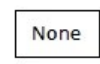
Name	Symbol	All Layers	Top	Bottom	Hole
IO Fluid					
IO Control					
Mixer					
Valve					
Bleed					
Storage					

Figure 6.1: Set of custom symbols made with Eagle software for producing a microfluidic device. The fluidic symbols are directly linked to physical characteristics that are then produced on the artwork. *Table Components:* IO Fluid is an input and output connection to the outside world. They are also being used as pressure inputs. The IO controls are vacuum connections that are designed for controlling valves. The Mixer is designed for mixing material. It is made of two chambers separated by a membrane. Fluids are added into the upper chamber. A pulsed vacuum or pressure is applied to the lower chamber, causing the membrane to flex and mix the material. The Bleed is a component that is designed to bleed off the vacuum once it is turned off, allowing the valve to close. The Storage is a component designed to store a set amount of a liquid compound prior to mixing. The valve is a component designed for turning off and on a fluid channel.

I started my microfluidic design of a simple mixing system by creating a set of symbols in Eagle as shown in figure 6.1. The table shows the link between the schematic

symbol and a physical layer definition, which is entered into the Eagle software in the creation of the symbolic libraries. Unlike standard printed circuit board fabrication (where the pads are used for the mounting of electrical components), these pads will form molds. Each layer forms an individual mold, and these layer molds will be combined to form a system of components. Pins will be placed in the holes, which ultimately form the layer-to-layer channels and external interconnections. This is in contrast to the conventional method in microfluidics which individually punches each channel hole. In the future, a design tool like Eagle will need to be developed specifically for microfluidic devices. *Evolution: Stainless steel pins used in casting mold to form IO and layer to layer interconnections.*

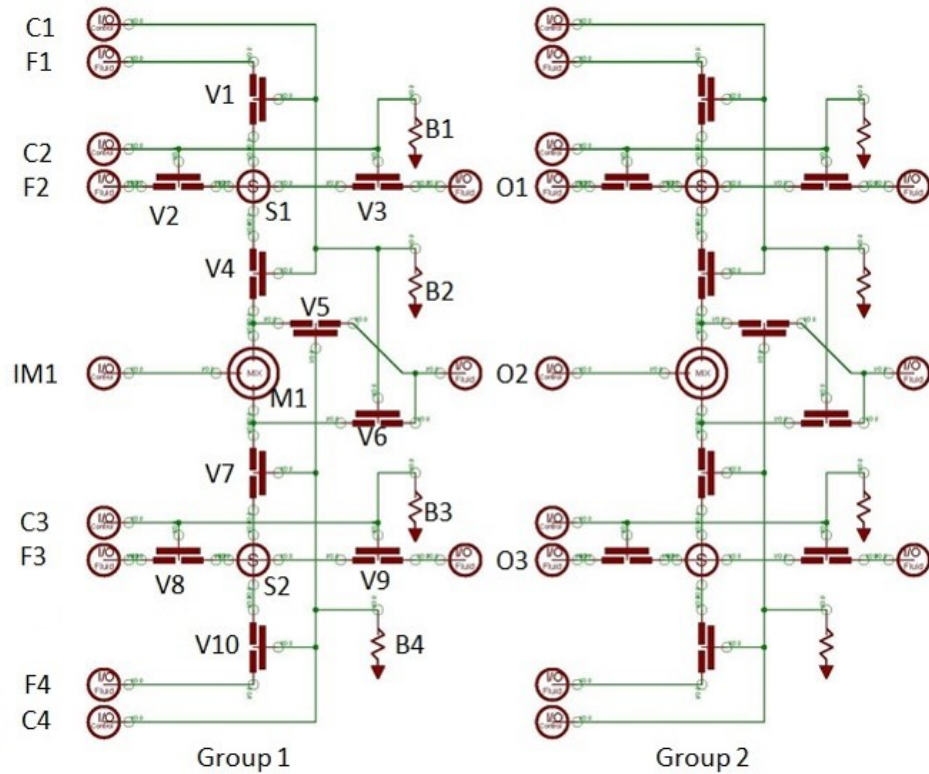


Figure 6.2: Eagle schematic showing the microfluidic mixing system. Custom symbols were developed for microfluidic components

Next, the symbols described in figure 6.1 are interconnected to form a schematic in Eagle software, as shown in figure 6.2. This schematic defines the microfluidic mixing system. The schematic defines how each of the fluidic components of the mixing system will be interconnected. The schematic interconnection will physically become the fluidic channels, as apposed to an electrical schematic, where they become the electrical interconnections. Schematics in both electronics and microfluidics have been used for simulation of a design. In the future, our microfluidic schematic could be utilized for modeling the performance of a design, in addition to defining it for fabrication. *Future Evolution: Fluidic Schematic used for both fabrication and simulation.*



Figure 6.3: Desktop printed circuit board (PCB) prototype machine. The machine can generate a PCB from Gerber files which have been translated to CAM files.

There are several methods that we looked at for fabricating a microfluidic device from a schematic. First, Gerber files can be generated from a schematic, and from these files, CAM files can be generated. These CAM files can be used to fabricate the PCB with a desktop milling machine (Fig. 6.3). There is a key difference between how the PCB is used in the electronics industry and how we are utilizing it. Instead of utilizing the PCB for mounting components onto and being part of an electronic device, our PCB is being used as a mold for casting the microfluidic device. The mold plate shown in figure 6.4 was produced by a desktop milling machine. This plate was then used to cast a microfluidic device. The microfluidic device did have leakage problems.

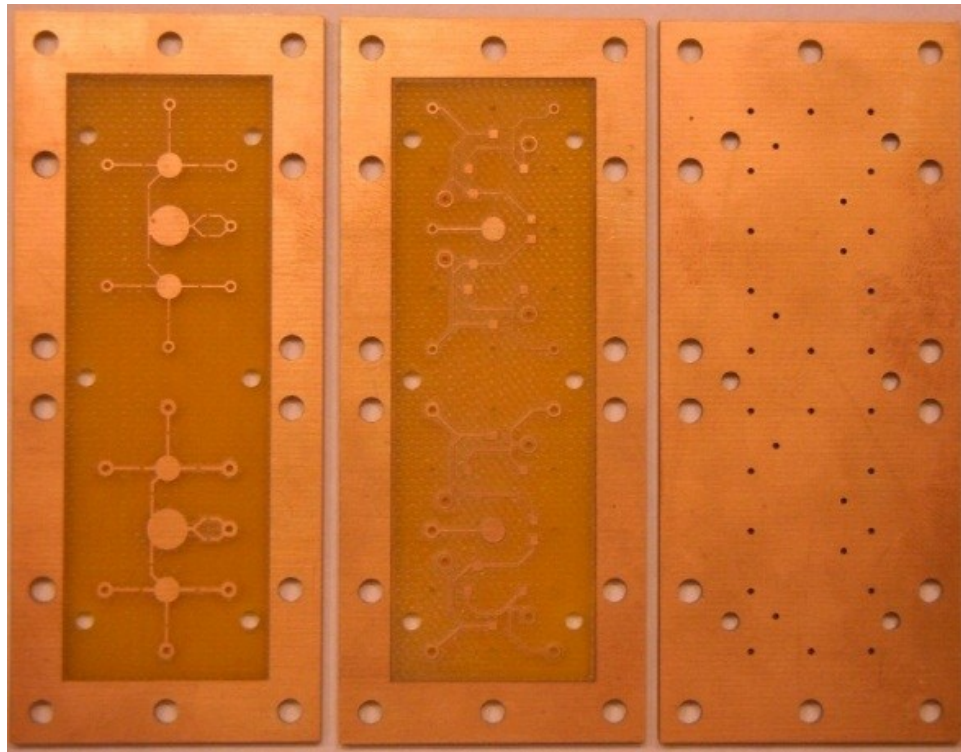


Figure 6.4: The microfluidic microcontroller was made from a tabletop milling machine

6.2 Translating a schematic to layout

After producing the schematic, the next operation is producing the layout. The layout is used to produce the printed circuit board. Recall from figure 6.1 that each component in the schematic is related to physical features. After all the components have been entered and connected in a schematic (figure 6.2), one can switch to the layout screen in the Eagle software, as shown in figure 6.5.

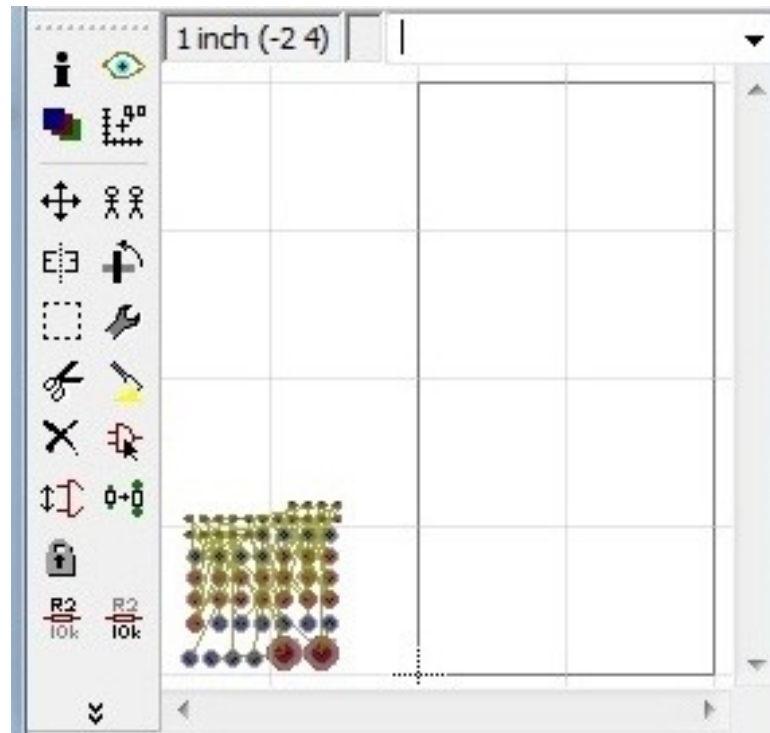


Figure 6.5: “Rat’s nest” physical layout is generated from the symbolic schematic.

Notice that all the components from the schematic are at the lower left. The components are no longer in symbolic form, but are represented by physical features on each layer of the layout, features that are defined by the symbolic table. Also, note that there is a

“rat’s nest” interconnecting the components. The rat’s nest comes from the interconnections defined by the schematic. These show where channels need to be created. In standard printed circuit boards they show where the electrical interconnections are needed and will be produced. Notice that the position of the components in the layout has not been defined. Instead, the components are simply shown clustered together on the left, waiting to be positioned. First, one can lay out the board on graph paper, recording the location of each component in an Excel spreadsheet. By right clicking on the component, one can enter its location, and the component mold feature will move to that location. The field of microfluidics could look at standardization of both IO type and location, as is done in electronics. Here, both IO Fluid and IO Control have the same hole size and are of the same physical IO interconnection type. Because of prior prototyping standardization, a 300 mil inch grid was selected for the IO interconnections to lie on. A device can have interconnections on any of the six sides, and in future work we will show this. However, for this device, all the interconnections will be on the lower face. Once the position of each of the components has been entered and the components are in their correct position, it is possible to zoom in and start laying out the channels. The default channel width can be set.

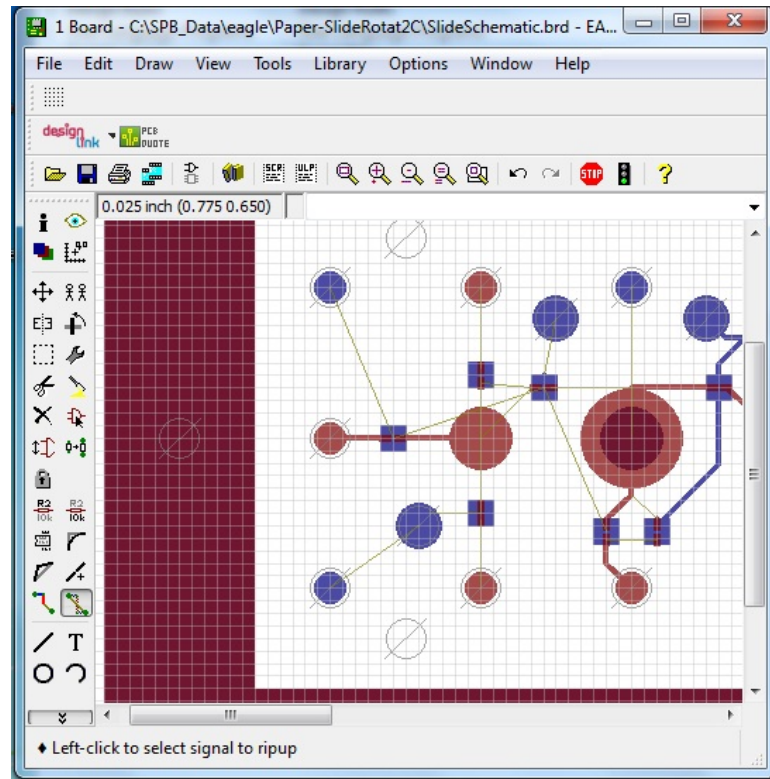


Figure 6.6: Zoomed-in view of rat's nest layout shown above

Figure 6.6 is zoomed in on a part of the board layout. The components are all in the correct position, or at least a starting position. These positions do tend to shift for non-IO components as the layout is developed. The IO components are fixed because they are set by the 0.30 Snap standard. The thin rat's nest lines on the layout show where the channel interconnection are needed. The blue and red lines show where channels have been made. The colors represent the different physical layers the channel will be on. Components can have interconnecting pads that can be on any or all of the layers.

6.3 Translating layout to Gerber files

Once the layout is complete, Gerber files are generated as shown in figure 6.7.

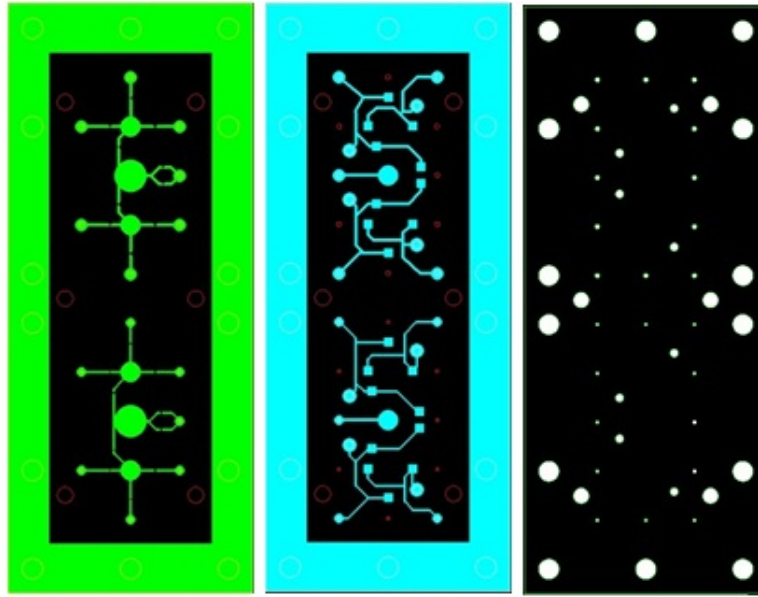


Figure 6.7: Image of Gerber files displayed in ViewMate. They were generated by Eagle from the layout, which was created from the schematic. The drill master shows where inner layer holes and fixture holes are located.

6.4 Translating Gerber files to mold plate

Gerber files can be used for generating artwork or translated to CAM software for operation of a desktop PC prototyping machine (figure 6.3) to produce a printed circuit board mold plate (figure 6.4). Printed circuit board mold plates can also be fabricated from the artwork (or Gerber files) and drill files at a PC fabrication house.

CAM software is used to translate Gerber files to Cam Files. A 0.062 inch single-sided 1/2 oz. FR4 material was used in fabrication of the layer mold plates shown in Figure 6.4. The 1/2 oz. represents the thickness of the copper, which is 0.0007 inch or 18 μm .

In this method, copper thickness determines the thickness of the channel, although the machine can be set to over or under mill. The FR4 is standard material used when making printed circuit boards, but the machine can mill a wide range of materials, including plastic. This method offers an important advantage over the standard silicon-SU8 mold method. It allows the ability to fabricate features at different depths. By using 1/4 inch plastic or 6.35 mm, it is possible to fabricate storage units up to that thickness. Another advantage we were hoping to achieve with the milling method was time savings. The time required to fabricate the three boards was 1.5 hours with the milling machine. Standard silicon-SU8 fabrication requires artwork. From placing an order to receiving the artwork, it can take a week. In addition, there is considerable cost savings in lab time and equipment. In the standard SU-8 method, clean room equipment is needed.[19],[20] This includes a spin coater, UV curing station, controlled hot plate, and washing baths as compared with the PCB desktop milling machine. The mold plate shown in Figure 6.4 is designed to be inserted between two aluminum plates containing a cavity in which the PDMS is injected into. We are testing two channel widths, 10 mil ($254\ \mu\text{m}$) or 16 mil ($406\mu\text{m}$). The sealing, channel width, thickness, and tolerance are the key drawbacks to this method (versus standard silicon-SU8 processing). In the standard silicon-SU8 process, 100 micrometers is the common channel width vs 250 micrometers. The problem we had with the PCB milled mold plate was the surface finish was too rough for PDMS-to-PDMS bonding using standard plasma etch methods.

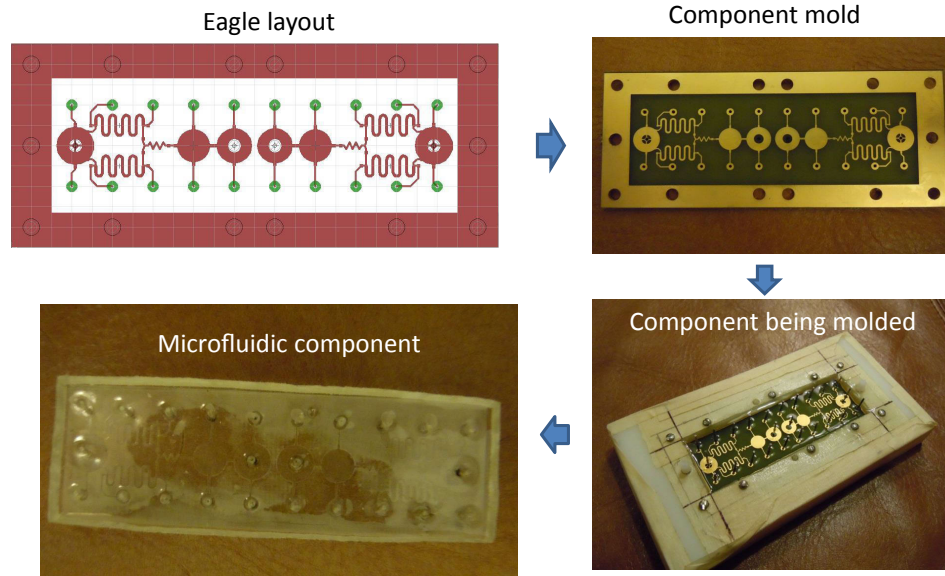


Figure 6.8: Printed circuit board (PCB) PDMS-cast mold. The traces form the channels. The vias are made with stainless steel pins passing through drilled holes in the PCB.

Next, we tried printed circuit board mold plates fabricated from an outside PCB house. They had a better surface finish. The PCB type mold offers another important improvement over the standard silicon-SU8 method, which is the PCB hole. The holes can act as guides for pins as shown in figure 6.8b. These pins will become the layer-to-layer fluid channels. These channels are typically made by a hand-aligned punching tool in standard microfluidic fabrication. Using our technique, the layer holes are made as part of the casting process. Unfortunately, the surface finish was still not sufficient to get a quality seal.

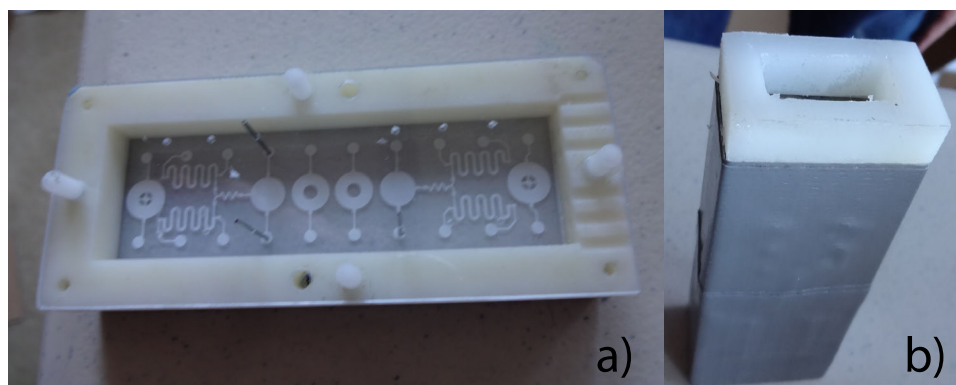


Figure 6.9: Artwork PDMS-cast mold. The thickness of the ink determines the height of the microfluidic channel (a). Leaky cast mold after it has been sealed with tape, with extra PDMS well on top (b).

Next, we tried having the artwork produced by an outside house. The outside house was able to print inks in different thicknesses. These thicknesses would become the channel thickness. Here, we laminated the artwork to Plexiglass and CNC drilled the holes to make the equivalent of the PCB mold plate. We tried 3D printing the entire mold, except for the mold plate, but due to flatness this again presented a sealing problem. So, the outside pieces were made in aluminum. Shown in figure 6.9a is the PDMS casting mold (open to make the insides visible). The frame was still 3D printed. Shown are two stainless steel pins which become fluid vias, and the artwork mold plate. The artwork did provide a flat enough surface to get a good seal, but the ink used by the outside house outgassed materials that caused curing problems with the PDMS. This problem seemed to improve over time, probably thanks to better curing of inks. We did try just using artwork from a normal copy machine, which has been shown to work in this type of process. Here, the channel thickness is determined by the thickness of copy ink. Under vacuum we had channels collapse, but our adhesion was improved. Shown in figure 6.9 b) is the mold after

it has been sealed with tape. This mold is made of several layers, and then sealed with tape. I call it a leaky mold because I know PDMS will lead out between the layers. I created a well on the top, so as material leaks out, new material can flow in. This method worked very well for casting patterns with few air pockets. The via method also worked extremely well, although we learned not to drill holes in the bottom plate. The stainless steel pins just needed to press up against the plate. Holes in the bottom created a spot for PDMS to leak into, and this PDMS would again cause a barrier to sealing.

Chapter 7

Identifying substances toxic to stem cell colonies by measuring multiple factors

As mentioned in the introduction, I developed for the Navy (Fig. 1.1) a data-logger to be deployed under the ocean. It needed high reliability, so I implemented a hardware health state methodology I called *grey scaling*. With this methodology the user measures as many parameters as reasonably possible and learns what is normal, and when a system's parameters move outside of the normal range, the system alerts the user. I came to UC Riverside with the concept that early disease identification can be achieved through regular monitoring of changes in a person's biomarkers. To do this, one needs to study multiple parameters simultaneously, beginning hopefully while the individual was healthy, and then continuing after the individual experienced health problems.

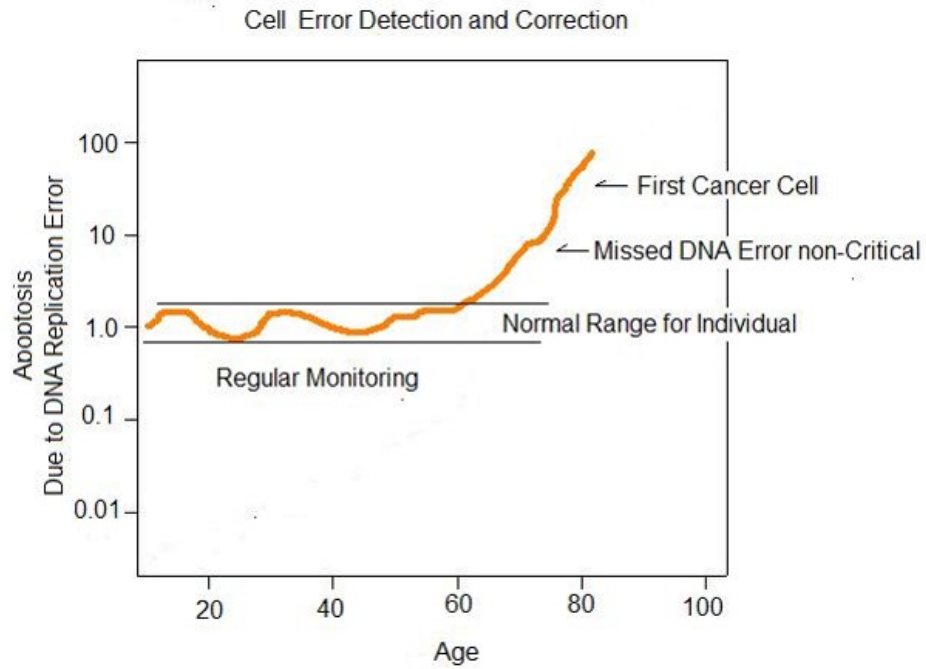


Figure 7.1: The health index concept: learn what is the normal state for several biomarkers, and when a person's biomarkers moves out of normal range, flag an error.

With this methodology, I believe that it is possible to identify a potential health failure before it occurs (Fig.7.1). For example, body environments can change; certain environments can be more conducive to cancer than others. Thus, if one can identify a shift toward an unhealthy environment and reverse it, one could potentially stop cancer before it occurred. This was one of my initial concepts when I returned to school.



Figure 7.2: Not actual test image, only showing placement of colonies. Toxicity testing of human induced pluripotent stem cells (Riv33). Row 1 used artificial cinnamon e-juice, and Row 3 used cinnamon e-juice. Each row had two controls and used 0.01% and 0.1% mixture of e-juice and media. E-juice is the liquid used in electronic cigarettes.

I was given the opportunity to implement this methodology in Dr. Prudence Talbot's IGERT class at UC Riverside, where I was given raw data on the growth of stem cell colonies under different toxic conditions. The other students in the class used the software CL-Quant to measure area and change in area of the colony to determine the health of the colony. I wanted to apply the concept I had come to the university with, and measure numerous parameters simultaneously to determine the health state of the colony. CL-Quant did not offer me the control over the raw data that I needed. I wanted to use MATLAB to look at multiple parameters simultaneously to better determine the state of

a stem cell colony. I began by trying to determine what perimeters I could find through image analysis of the cell colony video images. The images were taken on a Nikon Biostation imaging system in the Stem Cell Core at UC Riverside. We utilized 2 rows of a 12-well plate (Fig. 7.2). *Evolution: Measuring a large number of parameters from video images for the determination of the health state of a cell colony for toxicity determination.*

7.1 Growth (change in area) is insufficient for defining health state of a colony.

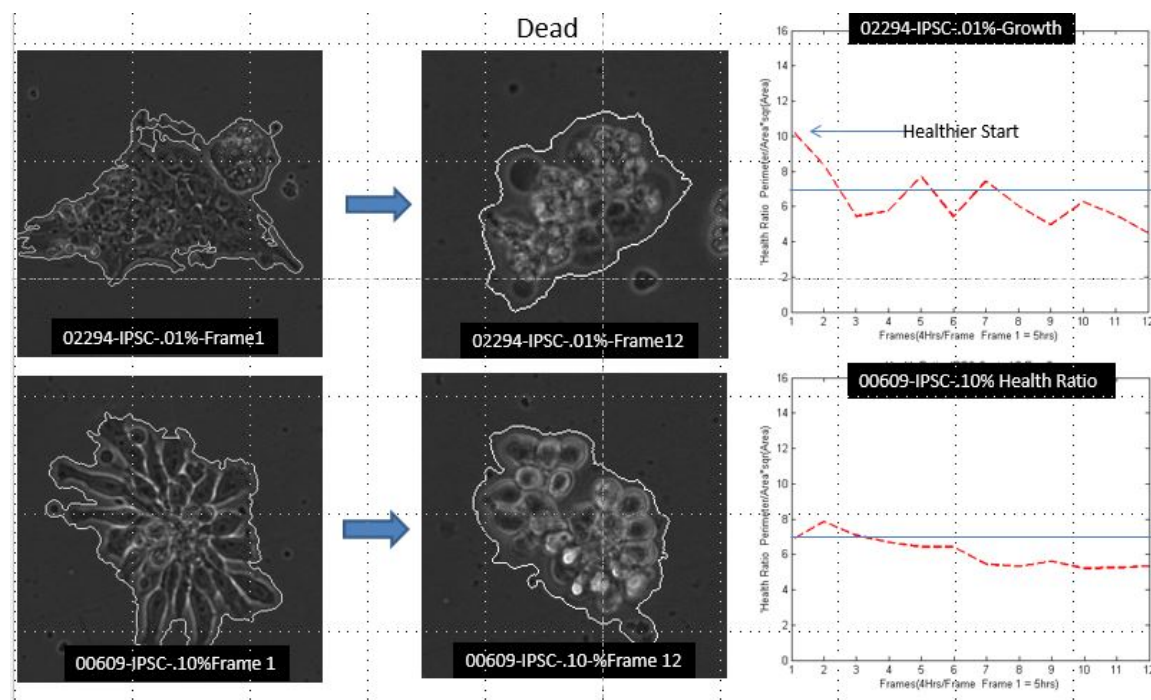


Figure 7.3: Images from the first and last frames of two different IPSC colonies. The reduction in colony area is insufficient to show which colony had a higher effect from toxins. The starting health state of the 0.1% colony was lower, showing we needed a better way of determining the start state of a colony.

One might assume that a decrease in colony area was worse than the area staying constant for identifying an unhealthy colony. For most cases this is true, but when the colony is already mostly dead, its area would stay relatively unchanged. Therefore, unchanged area measurements could occur because one has a dead colony (Fig. 7.3). I also noted another problem with the methods used in class. The imaging was started a certain time after seeding, and this was then considered the equal starting point for each of the colonies. The problem is that seeding of the colonies is not exact, and the colonies could already be in very different states at what was being considered the start of the experiment (Fig. 7.3). I felt we needed a better way to determine the state of the colonies from start to finish.

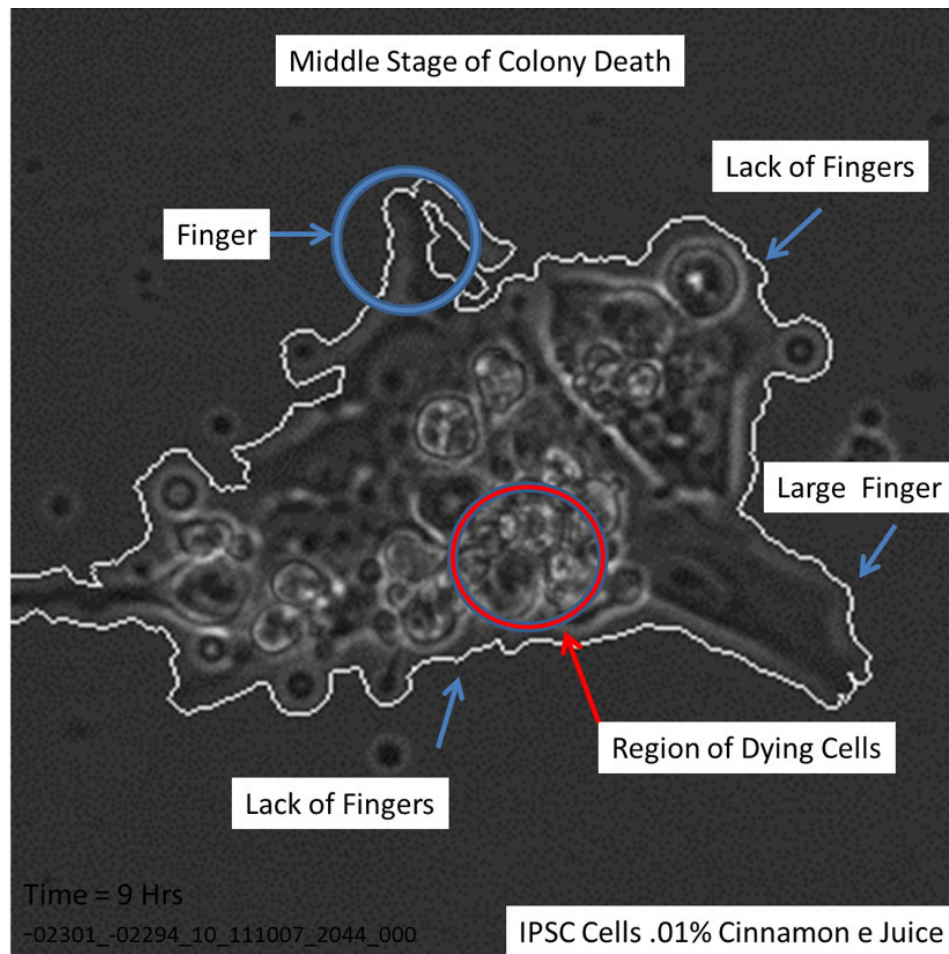


Figure 7.4: Micrograph of a dying induced pluripotent stem cell colony. Notice the region of higher white pixels where cells are dying. I interpreted a lack of fingers as a sign of an unhealthy colony.

I spent hours studying the cell colony images, searching for patterns. I noticed several things: first, when cells begin to die, they pull off of the tray and start to round. This rounding causes more reflection, or a halo. The final result is more white pixels (Fig. 7.4, “Region of Dying Cells”). Thus, I measured the ratio of white pixels in the colony to the total number of pixels in the colony. This is an important biomarker for determining dying cells. *Evolution: The use of halo or light reflection as an indicator of dying cells in a cell*

colony, and combining this with other parameters to determine the state of the colony. Next, I noticed that more fingers growing out of the colonies were related to what I considered to be healthier colonies. The dead or dying colonies did not have as many of these fingers (Fig. 7.4, “Lack of Fingers”). I wanted a simple mathematical way of determining the number of fingers. What I came up with was to begin by taking the ratio of the perimeter to the area of the colony. The more fingers, the larger the perimeter will be in comparison to the area. I felt this was an important biomarker for a healthy colony. *Evolution: The use of fingers stretching out of a colony to determine the health of the colony, and combining this with other parameters in determining the state of the colony.* Next, I saw that the colonies that were dying did not move as much. Therefore, I also added movement to my list of parameters to determine the health of a colony. *Evolution: The use of movement as a parameter in measuring health, and combining this with other parameters in determining the state of the colony.* The concept I was determined to demonstrate was that measuring more parameters, such as light reflection and fingers, gives one better determination of the health of the colony. I also showed that imaging was an excellent method of collecting several parameters simultaneously. Eventually, I wanted to combine all these parameters into one index, which I would call the health index.

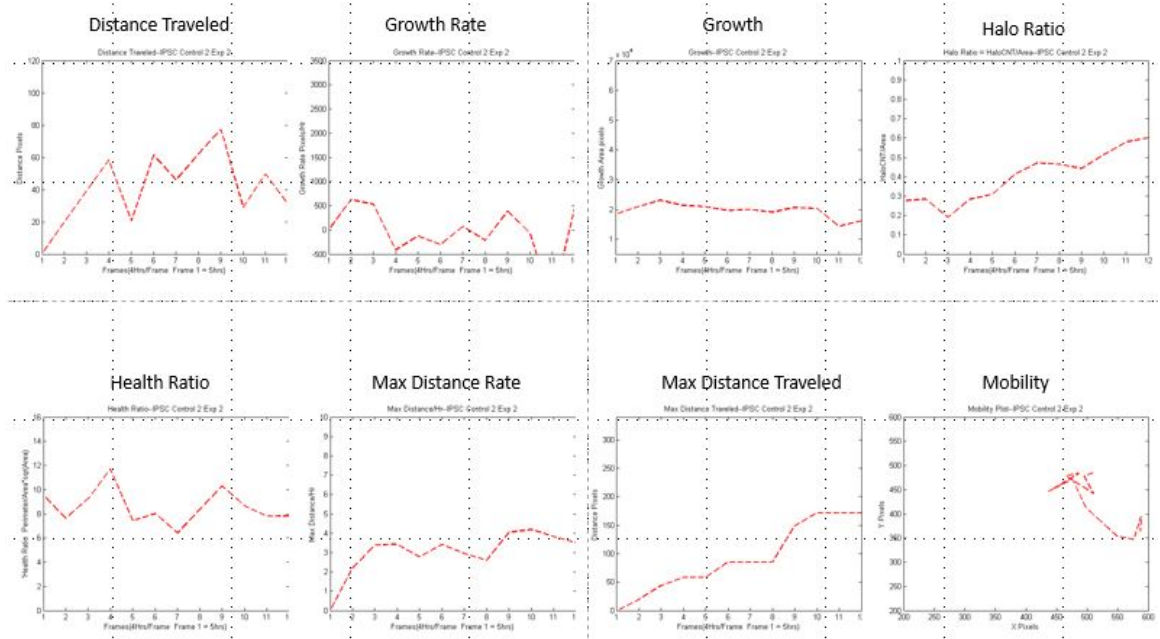


Figure 7.5: Parameters I measured with my MATLAB software by the end of IGERT class included colony growth, halo ratio (relates to light reflection as cell dies), health ratio (relates to number of fingers), mobility, growth rate, maximum distance rate, and maximum distance traveled

Once I had studied the colony images, I wrote the MATLAB code to perform the analysis. I felt that colony growth, halo ratio (relates to extra light reflection as cell dyes), and health ratio (relates to number of fingers) were the key parameters in determining the health state of the colony. The methodology and software also included distance traveled, maximum distance rate, maximum distance traveled, mobility, and growth rate (Fig. 7.5. As the user runs through many images, and experts define the final state of a large set of colonies, statistics should be able to identify the most critical factors.

7.2 Using growth (change in area), halo (apoptosis), and health ratio (fingers) to study the health state of the two controls

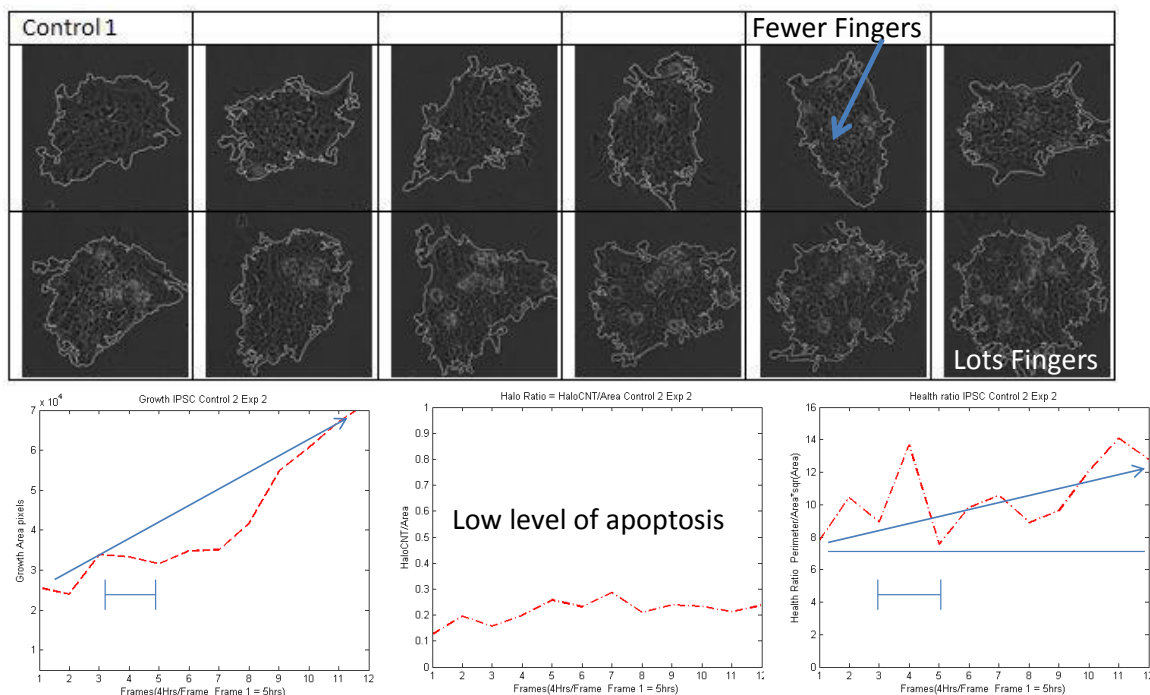


Figure 7.6: Twelve frames acquired of IPSC control 1. The first frame was acquired at 5 hours, and then a frame was acquired every four hours, up to twelve frames. Also shown are three graphs from the analysis of these images: growth (change in area), halo (apoptosis), and health ratio (fingers).

I began by looking at apoptosis (halo), number of fingers (health ratio), and growth (change in area). Looking at all three graphs simultaneously with the most healthy colony (control 1), I studied what graphical patterns could be determined. Ultimately, a researcher would want to run the control several times to develop a normal pattern over time for all three parameters. Examining Figure 7.6 we see that the area almost continuously rises from

frame one to twelve, indicating the cells were dividing and growing. This is what would be expected for a healthy colony control. Apoptosis stayed below a low ratio of 0.3 (ratio of bright pixels to total pixels of colony). The rise in apoptosis indicates to me that as colonies grow larger, it is normal for a slightly higher level of apoptosis to occur. It could be because cells are older, or it could be that the cells are in a more resource-limited environment as nutrients are consumed. Between frames 3 and 5 apoptosis had its most continuous rise. The health ratio remained above 7, even during the sharp changes. Since this was the control, this told me that above 7 for the health ratio indicated a healthy colony. Overall, the health ratio continued to rise as the colony grew, indicating a rise in protrusions (fingers) for the larger colonies. The health ratio had an interesting feature. It had a sharp rise from frame 3 to 4, followed by a sharp decline from 4 to 5. This means a decline can occur in a healthy colony but is not maintained. All three graphs indicate something was happening between frames 3 and 5, with the health ratio being the most dramatic. Growth showed its only decline in this region, and apoptosis showed a slight increase. These types of anomalies are interesting, and could become important for toxicity testing. One potential problem is that these anomalies could result a false positive. The key result here is that anomalies can occur, but the best way to determine the health state of a colony is through the automated interrogation of several parameters, with a good understanding of what is normal behaviors for those parameters, by running several controls.

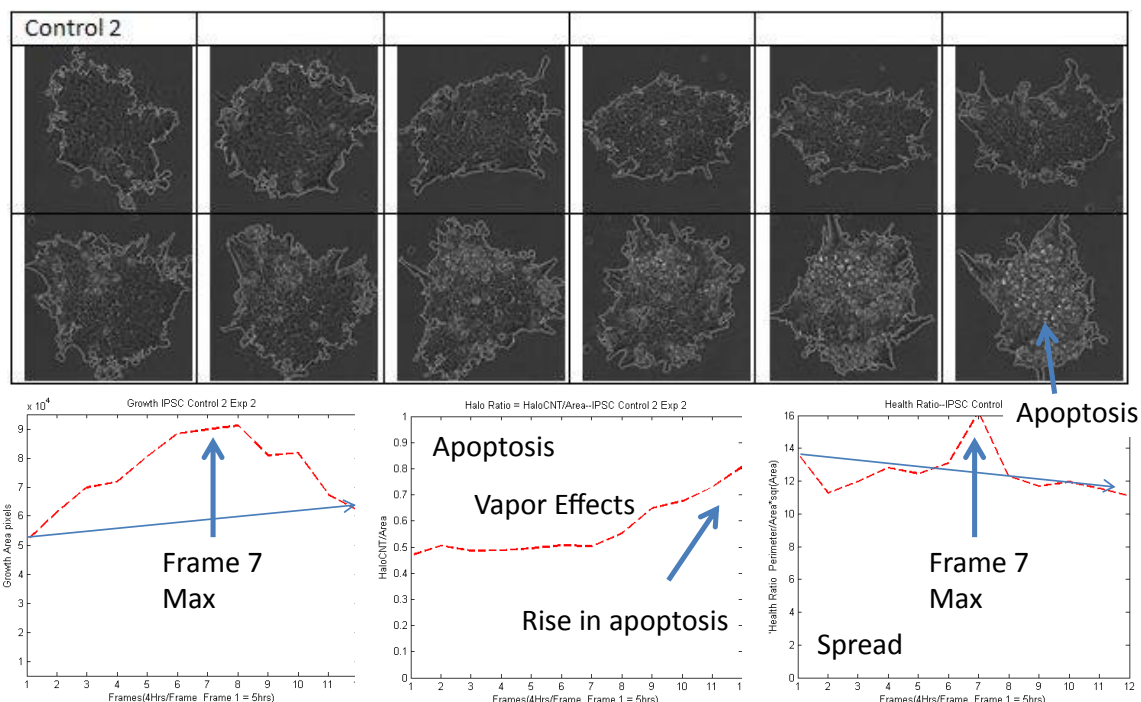


Figure 7.7: Twelve frames of IPSC control 2. First frame was acquired at 5 hours, and then a frame was acquired every four hours, up to twelve frames. Also shown are three graphs from the analysis of these images: growth (change in area), halo (apoptosis), and health ratio (fingers).

Control 2 was the most interesting of the colonies (Fig. 7.7). The experimental setup was almost identical to control 1, except its well was physically closer to the well containing 0.1% e-juice. All three graph types indicate the colony health started declining around frame 7, with the health ratio having the sharpest feature. Images of Frame 1 of both controls were acquired relatively at the same time. Again, this shows the importance of understanding the state of the colony at the start of experiment (frame 1). In this case, control 2 is almost double the size of the colony in control 1, so control 2 is therefore at a different point of its growth curve. What caused the decline in health of control 2? The graph of the pixel area shows a rise from frame 1 to 8 (from 5.2×10^4 to 9×10^4 pixels),

and then a decline from frame 8 to frame 12 (from 9×10^4 to 6×10^4 pixels) (units of area are the number of pixels in the region). During the growth period (rise in area) the cells are multiplying at a rate faster than cells are dying. During the reduction period (area shrinkage) cells are dying and leaving the colony at a rate faster than cells are multiplying. The apoptosis graph stays relatively flat from frame 1 to 7, with a value of around 0.5, and then it rises from frames 7 to 12 to 0.8. Halo is the unitless ratio of bright pixels to total pixel area of colony. This supports what we saw in the area graph, but also tells us that apoptosis was not increasing during the growth phase, and only started rising after frame 7. Because of the marked drop in area, I also think this indicates a slowdown in cell multiplication. The health ratio graph was above 10 though all the frames overall rose from frame 1 to 7, spiked up at 7, and then dropped after that. Larger colonies tend to have a higher health index. I originally believed the drop was due to vapor build up that came from the nearby well containing 0.1% e-juice. When one observes control 2 during the same size range as control 1 it only exhibits a rise in health. Therefore, the decline could have been due to large colony size. Looking at multiple parameters simultaneously when doing toxicity testing provides better insight. This work would be improved with repeated testing under similar conditions.

7.3 Smart edge detection

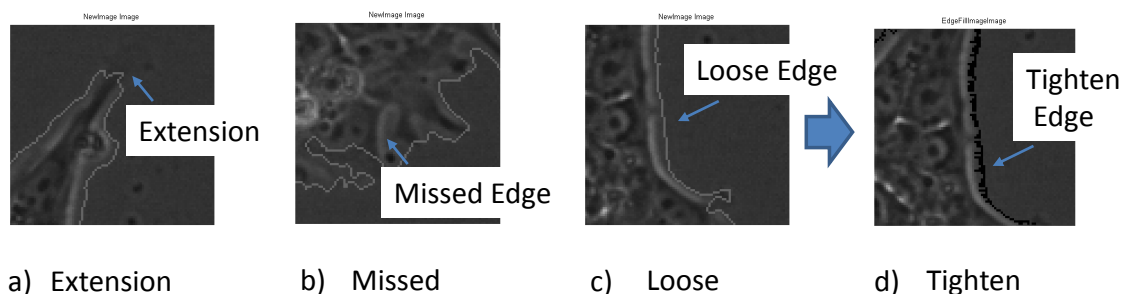


Figure 7.8: (a) Colony extension that the boundary algorithm missed. (b) Bay in colony that boundary algorithm missed. (c) Loose edge that boundary algorithm formed. (d) Tightening of edge with smart boundary algorithm.

Our ability to measure toxicity is limited by our capability to define the health state of a colony. Therefore, the accuracy in which we can measure parameters or track morphological changes will dictate our ability to determine the health state of a colony. Defining the boundary of a colony is one such critical parameter. Looking at figure 7.8 we see several places where the boundary algorithm was insufficient. The white edge boundary is the first boundary created by my software. In Fig.7.8 a) one can see that we are missing the tip of the extension. In Fig.7.8 b) one can observe that we are missing some of the inner bays of the colony. In Fig.7.8 c) one can see that the boundary is loosely following the actual colony boundary and could be tightened up. Knowing what features were being missed, I set out to write a smart boundary algorithm.

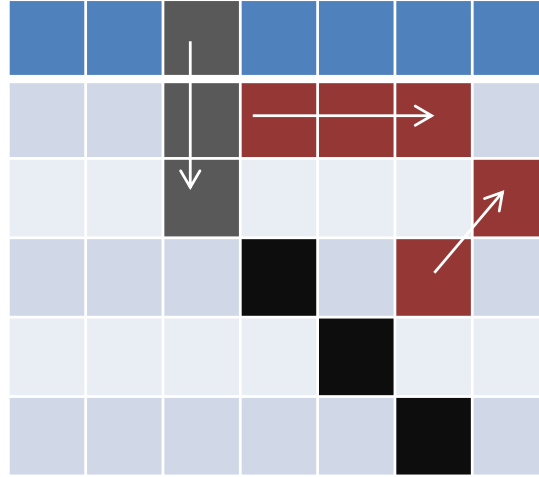


Figure 7.9: Smart boundary detection

As shown in Figure 7.9, my smart boundary detection searched perpendicular from the current boundary (shown in gray and black) for the correct edge. These searches become an array of gray-scale vectors. We know that the boundary of the actual colony is continuous. When searching, these vectors are created from the original gray-scaled image. With the perpendicular gray-scale vectors we can determine a better estimate of the edge. We can teach the software to look for features such as extensions and bays as we demonstrated in Fig.7.8. *Evolution: Smart edge detection is critical for more accurately determining the state of the colony.*

7.4 Conclusion of identifying substances toxic to stem cell colonies by measuring multiple factors

Here, I have developed a MATLAB-based video analysis methodology to track morphological and other changes in stem cell colonies under different toxic conditions. Standardized growth patterns for several parameters can be established to compare colonies under different toxic conditions. A colony health index could be developed incorporating changes in cell colony motility, attachment, fingers, apoptosis and growth. We hypothesize that changes in this index will correlate with changes in gene expression and cellular pathways, and that these changes can be used to predict the health risk of chemicals. In addition, imaging has shown itself to be an excellent tool for capturing several cell colony parameters simultaneously and dynamically. However, I believe what imaging is detecting can go far beyond what we can see. This methodology has the ability to determine changes in state at multiple scales, not just the size scale of the cell colony. These cells could have been taken from a patient, and morphological changes of the cells could relate to a health issue of that patient (macro-scale identification). On the other extreme, imagine a healthy cell wanting to move or begin cell differentiation. Due to micro-environment changes, the cell starts a sequence of DNA processes that lead to the remodeling of the actin filaments. This, in turn, results in the formation of fingers. So, a finger (micrometer-scale) is the result of a large number of DNA events (nanometer-scale). With humans it is important to recognize that a toxin could be something that changes or disrupts normal cellular pathways. The result of these changes can be very subtle, and most often results in a small change, not death. Thus, by studying changes in normal morphology of a stem cell colony, one can detect changes

in pathways. Studying many parameters and determining their normal patterns will give us the best sensitivity to changes resulting from a toxin or a wide range of problem that we wish to detect. The changes we want to detect, may not always be cell death; they could be uncontrolled growth, improper differentiation, etc. The more parameters we can accurately measure, and the better we understand the normal patterns of these parameters, the earlier we can detect shifts. The multi-scale nature of this method can be incorporated into the health index I wish to develop for people. Therefore, I believe that monitoring of cellular processes can be utilized to identify overall health problems, such as obesity. I recognize that this methodology works best with control tests, where the health of the cell colony can be expertly determined. Then, one can mathematically determine the advantage of adding more parameters and determine which ones are more critical. The IGERT class was too short, and I had insufficient data to fully prove my methodology. I believe that I was the first to test the toxicity of E-cigarette on a IPSC (induced pluripotent stem cells) by better determining the state of the colony, through image analysis which incorporated the measuring of halo, fingers, growth, and movement. My Matlab software was able to determine all of these factors simultaneously, thus providing a better understanding of the state of the colony at any time, over looking at only one or two of these factors.

Chapter 8

3D visualization of biomarkers

One of my interests is early disease detection. A key to accomplishing this is through the repeated measuring of a large number of biomarkers, across large populations, over time. Diseases would be initially identified by health professionals. Computers would then be used to analyze the collected data and look for patterns that then could be associated with these diseases. Over time, computers would be able to determine what is a normal, healthy state for an individual, and a health group community. One's health group community is defined by people who have a similar health background, such as a similar age, sex, and ethnicity. Once we have measured a patient's biomarkers over time during a healthy period, we would then learn what a normal, healthy state is. Therefore, when someone shifts out of their normal health state, we can provide an earlier health alert. One would want to measure as many biomarkers and health factors as economically possible. These bio-markers and health factors will come from a wide range of places. First, we wish to include the tests that a person currently is having done. This might include urine and

blood tests. Next, we wish to include the growing number of wearables, which can measure sleep patterns and many athletic activities. Other common devices include digital scales and products to measure body mass index. Imaging systems are being developed to image food and provide estimates on one's nutrient and caloric uptake. We could design an imaging system in a microwave, which could identify a person's food and nutrient intake. We also wish to monitor one's medicine uptake with a digital pill and cream dispenser. Genetic profiling is providing a rich body of predictive information to add to our other biomarkers. Finally, we wish to develop new instrumentation for improving individualized body fluid profiling. Body fluids we would be addressing would include: urine, blood, sweat, saliva, bone marrow, fecal matter, etc. Ease of collection will be a critical factor in data collection. A diagnostic toilet could be a critical first step in the ease of personalized data collection. In third world countries, a diagnostic outhouse could be developed to allow an individual's health factors to be uploaded onto the web, and analyzed by the best technology of the time, illuminating needs before there is a crisis.

Early disease detection can be achieved through studying a wide variety of biomarkers, but the first data that should be utilized in this effort is the data that already exists. The current standard is that if a patient is having a problem, the doctor looks at several markers, and searches for things that are outside the "normal" range. There are several problems with this. First "normal" is defined by a much larger population set than it should be. Ideally, "normal" would be determined for an individual. For example my red blood cell count is high. Doctors suspect that is because I live in the mountains where the air is thinner. So, if my red blood cell count ever moved to the bottom of the normal range

it could indicate a problem. A doctor looking at a single test would not see that problem. Only by looking at trends of health over time would it be seen. People should be more continuously monitored, even during healthy periods, and a very wide range of biomarkers looked at, so early disease identification and prevention could be achieved.

8.1 A first data set: my blood tests

With this in mind I went to the UC Riverside health office and asked for my medical records. They turned over 169 PDF files (Fig. 8.1). I then asked one of my interns, Devin Finkel, to analyze this data. Ultimately, I wanted to be able to visualize the progression of a disease (I do not have a disease, but the process of analyzing the data is the same). If we could analyze my data, the software we developed could be used for anyone. We made some assumptions; first, that the progression of a disease could be subtle and a combination of several factors. Next, when the data is being analyzed, we may not know the disease we are looking for. This is why looking at numerous biomarkers is so important. Different diseases will effect a different set of biomarkers. This defined our objective.

The first task was to convert the PDF files to a text file. We accomplished this. The one thing that was missing from our conversion was the carriage return at the end of each line. This carriage return would be extremely useful in data parsing.

Test Name	Result	Flag(s)	Reference Range	Reported Date	Footnote
WBC	7.2		3.5-10.0 K/uL	10/8/2012 10:34 AM	LAB
RBC	5.50		3.80-5.80 M/uL	10/8/2012 10:34 AM	LAB
HGB	17.8	H	11.0-16.5 g/dL	10/8/2012 10:34 AM	LAB
HCT	51.5	H	35.0-50.0 g/dL	10/8/2012 10:34 AM	LAB
MCV	94		80-97 fL	10/8/2012 10:34 AM	LAB
MCH	32.3		26.5-33.5 pg	10/8/2012 10:34 AM	LAB
MCHC	34.5		31.5-35.0 g/dL	10/8/2012 10:34 AM	LAB
RDW	13.2		10.0-15.0 %	10/8/2012 10:34 AM	LAB
PLT	168		150-390 K/uL	10/8/2012 10:34 AM	LAB
GRAN%	65		43-76 %	10/8/2012 10:34 AM	LAB
LYM%	30		17-48 %	10/8/2012 10:34 AM	LAB
MON%	5		3-7 %	10/8/2012 10:34 AM	LAB
GRAN#	4.8		1.2-6.8 K/uL	10/8/2012 10:34 AM	LAB
LYM#	2.10		1.20-3.20 K/uL	10/8/2012 10:34 AM	LAB
MON#	0.3		0.1-0.4 K/uL	10/8/2012 10:34 AM	LAB

Figure 8.1: This is an example of a section of blood test that was provided in PDF form

8.2 Parsing Data

Once we had text files, Devin then developed a parsing program in Python to move the data into arrays for each biomarker. He was then able to parse out over 50 biomarkers from the PDF files. Shown in figure 8.2 is the command line menu used for selecting which biomarker one wants to look at. The user is able to select as many biomarkers at a time as they wish to review. The biomarkers are selected one at a time, and when finished one simply enter “-1”.

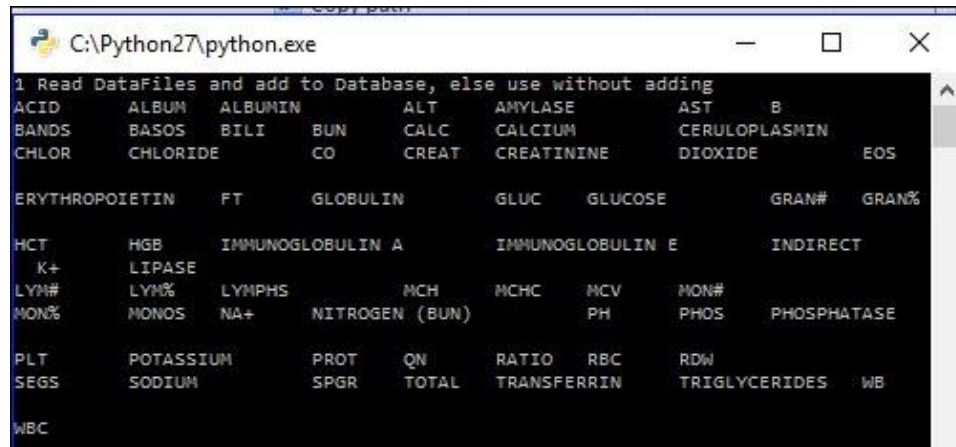


Figure 8.2: Command line menu, showing the current biomarkers that are supported by this software

8.3 Graphing medical data to look for trends

A graph is then generated as shown in figure 8.3. Again, this is a step above what is currently being done at our health center, where a doctor looks at single or multiple sheets of paper (or on a computer screen) with blood tests on them to look for problems. Here, it is much easier to see trends and how biomarkers track together. The software needs further refinement, but this is a good proof of concept.

One problem we are having is gaining access to data. This is needed to better develop the software. The software could be downloaded to a patient's computer, allowing them to analyze their records in complete privacy. Software could be developed to aid the doctor in looking for trends, and uploaded on the health center's computers. So, if the program identifies a pattern it could alert the doctor to look closer in an area. Reading this, you might be thinking that people already know this, and large organizations are already working on this, and this would be true. But how many people die everyday of

diseases that could have been detected early by these methods? Even as these methods start showing up in larger hospitals, how fast will they spread? Will they reach third world countries as fast as they could if a larger effort was made? How good are these algorithms at detecting disease, and which diseases are being looked for? We could compare our student-developed algorithms to theirs. We could target diseases that are not being looked at. Another important factor is, will our students be learning while others solve this problem? Instead of figuring out why this might not bear fruit, we should be figuring out how to bring this type of data to our students, so they can join in the effort. What I have seen is how problems they understand inspires and drives students. *Evolution: Translation of standard lab results to a data base, so they can be analyzed and graphed over time.*

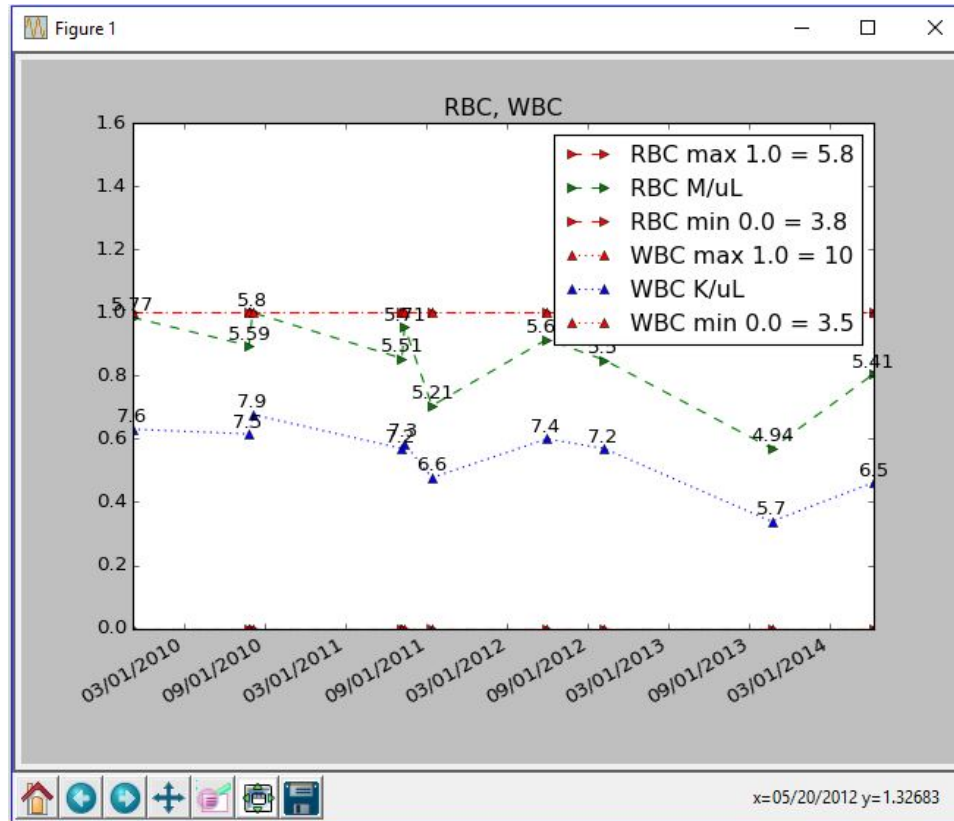


Figure 8.3: Example of a graph that was generated by this software, showing red blood cell and white blood cell counts over time. The left axis is a normalized range from 0 to 1. The zero represents the minimum value for the red or white blood cell count. The actual values for these minimums are shown in the legend. The value 1 represents the maximum values for the red and white blood cell count, and again the legend shows the actual values

8.4 Developing a 3D visualization method for simultaneous visualization of a wide range of biomarkers

In long conversations with Devin, we imagined what this software might look like. He came up with a spine concept, where each spine was a different biomarker over time; these would create a 3D visualization structure of a person's health. Together, we evolved this concept. This would require a large body of cloud-based data. From that data, one

would be able to determine what was normal and healthy, and what percentile for each biomarker a person fit into. We then envisioned a 3D topographical color map where deep green was healthy and normal, and deep red was the few percentile in the unhealthy range. Orange would be where you were transitioning from healthy to unhealthy. One would be able to look at hundreds of biomarkers simultaneously. Our hope would be that different diseases would have distinct topographical shapes and color patterns. We know that the primary biomarkers for a disease would show a known trend, but what about less-studied biomarkers? We believe this methodology could highlight unknown patterns. When looking at these 3D visualization maps over time for the same disease, are there consistent patterns, and if so, how early could they be detected? Because one is looking at multiple cases of the same disease, one can see what is consistent and what is different. The user may only be able to find consistency within a subgroup of that disease, but within that subgroup you can see early signs. By studying raw data this way, a student begins to develop instincts about a problem, and with those instincts they will better be able to guide the algorithm being developed to identify a problem. It is understood that these topographical visualization maps of a person's health are for people. They might allow a trained physician to identify a problem, but more likely they help guide the development of an algorithm that would alert the physician. Ordering of the data in these topographical maps would be critical. It would need to be the same across all the patients. I believe that one would place biomarkers together that had a high correlation with each other. If one correlated strongly with the inverse of another, they would still be placed next to each other, but one biomarker's values would need to be inverted. A physical example of this is whether one measures

the area furniture takes up in a room or floor space that remains; the information these numbers provide is the same. The software would start by minimizing change between data spines. Also, unfortunately, optimum biomarker grouping could depend on the disease. So, even though one would start out with the same biomarker ordering when looking for a disease, when a specific disease was identified one might select a different data grouping, and include additional biomarkers. We thought about both flat and cylindrical 3D maps. For a cylindrical map one might have two spines for the same biomarker. One spine could be for when the biomarker was above max healthy value, and one spine for when it was below the biomarker minimum. Missing data would be one of the real world challenges that this type of data mapping would be facing. The software could linearize a guess for data from other data of the same biomarker that one did have, or look to correlated data for an estimate. We then started figuring out how to create a 3D surface. We began to look at mesh algorithms, which often use triangles. Because our data comes in sets, a rectangle mesh may make more sense. Two triangles can still form a rectangle. Shown in figure 8.4 is a early visualization map. It begins to show the direction in topographical maps that we are working toward.

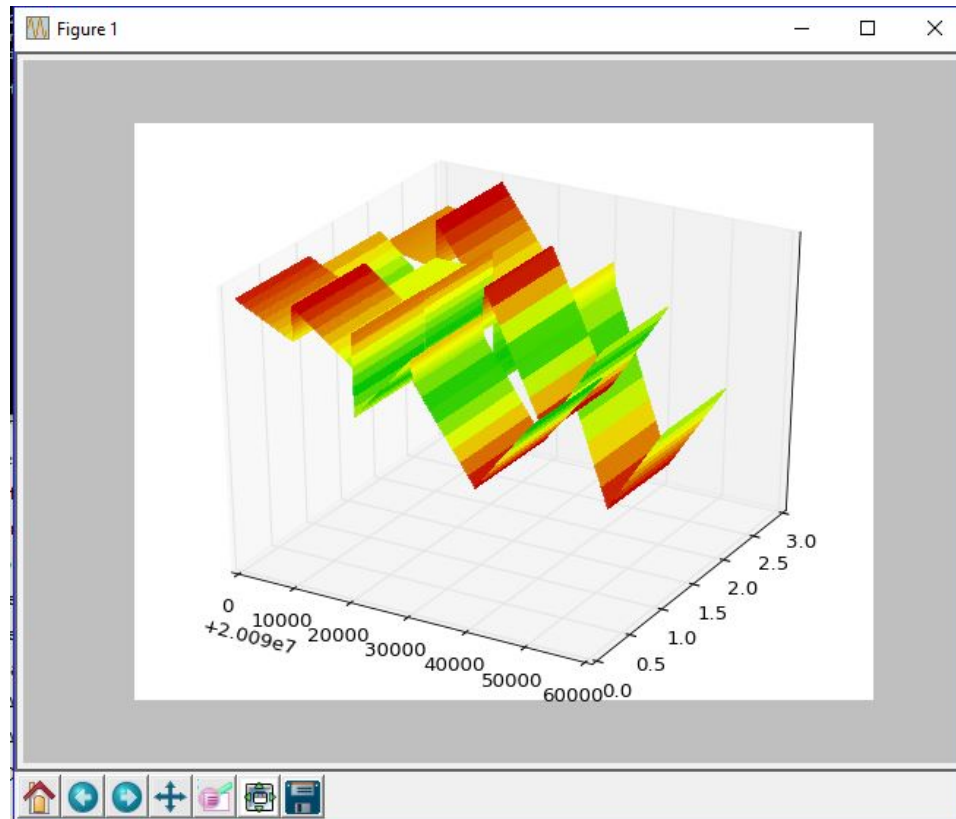


Figure 8.4: Early 3D visualization map of biomarkers

For this current method (figure 8.4) we would need to fill in rectangles between ribbons. One would need to be able to adjust space between ribbons and the width of ribbon. We might decide to make the ribbon width zero, so it's simply the surface between ribbons, or we might decide to make the space between ribbons and the ribbon width equal. Also, we may want to take a step back and look at what other surface algorithms already exist. MATLAB may have an algorithm that can create a surface from a set of points. So, here I would want to call a MATLAB program from Python. I had intern Krishna Addanki work on just that, and he was able to call an executable file from Python, and then after it

was finished, the program continued to run. It may be important to recognize, when looking from above, that the colors provide all the information needed. We may want to look at a 2D image with a color map. The 2D color map may be the easier software to develop for data visualization, although a 3D color topographical is cooler. It also would be important in the software to be able to click on a region of the topographical map, and have it identify which blood test (or other test) was used to create that region. This would be important to let a doctor know what specific blood test or tests were showing the problem; it would also be a way for us to confirm the functionality of our software. It might also be useful to select a blood test and see which region of the color map it was associated with. Finally, we may want to ultimately normalize what are normal patterns. So, if a person was 100% normal you would have a flat plane or a green color map, and colors or mountains and valleys are indication, shifts in normal. Here, you would want to tighten up what you consider normal so you could see small changes early. This, again, can be improved upon by learning what is normal for an individual. *Evolution: Disease identification and visualization through color topographical maps of a person's biomarkers over time.*

One of the most powerful search engines I know of is Google. By entering a word or phrase, it skillfully determines what website the user is most likely to be interested in. Imagine using that search engine to look at biomarkers and skillfully determine what disease you most probably have or don't have. I believe Devin said the Google search engine was open source, or at least there are methods to tap into it. Thus, students could start applying its power to this problem. One final note, much of this section came from an email I sent

to Devin, giving him thoughts I had on this system. We can see the future. It is only a question of what part we play in it, and if our part can make it happen sooner.

Chapter 9

MEC Compressor

I was in a room full of potential interns, helping with the development of the MEC system. I had just put an electrical schematic up on the chalk board to describe a MEC system. At that time, MEC schematics were either electrical or fluidic. Realizing that I could not fully describe the system with the schematic from only one discipline, the idea for the multidisciplinary schematic was born. With these new students, I put three of them on drawing up the schematics for three different systems which were already being developed: the MEC Compressor, MEC Coil winder, and MEC Pump. I wanted these schematics to reflect the interdisciplinary nature of the devices. Sketching out the schematics for the devices with new symbols, I defined a grid standard on how to make the symbols. At the time, I had been using Adobe Illustrator to draw up the schematics like the one shown in figure 2.1. This proved very frustrating for the students. There was no fast or easy way to draw the interconnections between symbols. The father of one of the students (Andrew Gen-Kuong) suggested using Microsoft Visio instead. Custom symbols could be easily made, and

Visio is made for the interconnection of symbols. The interdisciplinary schematic shown in 9.1 was drawn up with Visio for the compressor by one of those early interns. Visio is what we use to draw our multidisciplinary schematics today. To me, watching a MEC schematic evolve was very exciting. One thing that is still needed is a better part numbering system, because our system is evolving so rapidly. It is not sufficient to refer to something as a “solenoid valve.” The valve’s part number is needed to refer back to which revision of the valve is being used. Now, when I have my students start a system, I first have them define the system with a multidisciplinary schematic. *Evolution: MEC multidisciplinary schematic done with Visio provides a full description of a system.*

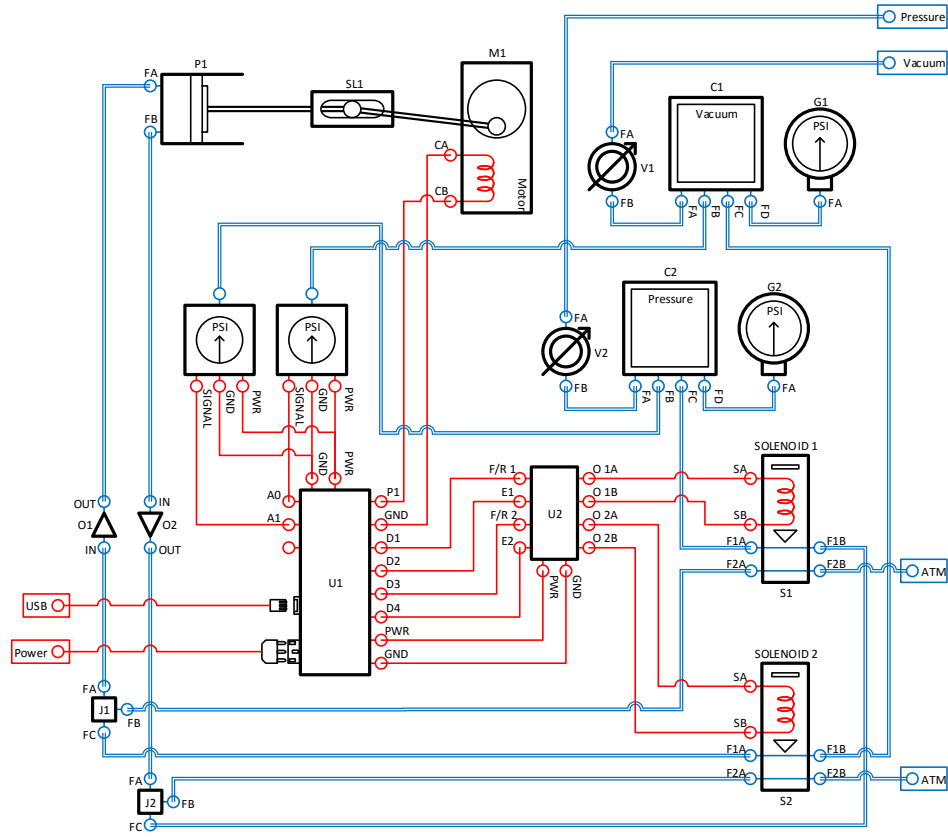


Figure 9.1: This MEC multidisciplinary schematic symbolically defines the compressor and vacuum system. The blue lines are the air lines, the red lines are the electrical connections, and the black lines are the mechanical components.

I think that it is important to note that a *compressor* was the first instrument to really integrate electronics with fluidics. It was selected because of the limitations found on the fluidic router described earlier. With the fluidic router the instrument only had a very short operation time depending on the pneumatic “charge” stored in the vacuum and pressure chambers. After that, the pump would no longer function. We needed a continuous source of vacuum and pressure, our fluidic energy source. We developed the compressor to meet this need.

9.1 C++ to Python

Control for the compressor system is through the USB connection between a laptop and U1 (Arduino) on the schematic 9.1. Originally, one of my interns (Josh Ice) had written the control code in C++. We were developing a general MEC scripting language for controlling devices. The USB interface in C++ proved to have some issues. Dr. Grover suggested we switch to Python, a change I was initially quite opposed to because of all the work we had already put in; plus I felt C++ was more standard in industry. The problem was that our scripting language was going to need loops, conditions, etc. Utilizing C++, we were going to have to reinvent all these commands. With Python we could simply use the Python command itself, because it is, itself, a scripting language. We switched to Python and saw an immediate improvement. It is a much more straightforward language to work with, especially for our application. I learn the most from the areas I disagree with people, so I always appreciate getting alternative suggestions. Adam Veres took over the writing of the Python code, and has written most of the Python code used in the MEC system today.

Evolution: MEC scripting language switches from C++ to Python.

9.2 Operation of the Compressor

The compressor schematic (figure 9.1) has several sections. *Control* of the compressor is performed by commands coming from a laptop, through the USB connector, defining the pressure or vacuum the user wants and when to turn on and off the system. *Electrical power* comes from a wall plug, through an adapter, to the system. U1 is an Arduino and daughter card. Kevin Pham was one of my early interns. I directed him to the same on-line

tutorials that I had used to learn Eagle. He taught himself Eagle, and created the daughter card from my design. The daughter card is designed to plug into an Arduino, and provides additional support circuitry. In this case, the daughter card had two FETs (figure 9.3a), which could drive the DC motor. Thus, the Arduino would send a signal to the FET, and the FET could turn on the DC motor of the compressor. U2 is an H-bridge module. The H-bridge can drive current in either direction through the dual latching solenoid. One direction latches the solenoid on, the other off. The compressor has two *storage chambers*, a vacuum chamber C1, and a pressure chamber C2. The pressure and vacuum in these tanks are monitored in two ways: first, with mechanical pressure and vacuum gauges (figure 9.3b), and second, with electrical pressure and vacuum sensors, which put out an analog voltage that the Arduino can read. The method used for *generating pressure and vacuum* (figure 9.3i) is using a piston and a geared motor. The motor drives a wheel (M1). The wheel has a rod that drives a slider (SL1). The slider drives a piston (P1). The last critical parts of this section are the one-way valves (figure 9.3g), which allow air to only flow in one direction through them.

The compressor operates through a sequence of events. First, a command specifying the desired pressure is sent to the Arduino. After the command reaches the Arduino, the Arduino reads its pressure sensor voltage to determine its current status. If the value is too low, the Arduino sets the air control lines to allow pressure from the piston to be driven into the pressure chamber and drawn from outside air. It then turns on the motor. As the motor turns a wheel, a piston is drawn back. As the piston pulls back, air is pulled from the outside through a one-way valve into the piston chamber. Then, as the piston

drives forward, air is pressed through another one-way valve into the pressure tank. The Arduino continuously monitors its pressure level. It repeats this operation until it reaches the appropriate pressure. A similar procedure is used for charging the vacuum chamber, except instead of drawing air from the outside, it draws air from the vacuum chamber and then releases it to the outside.

9.3 Discussion of the Compressor and its components

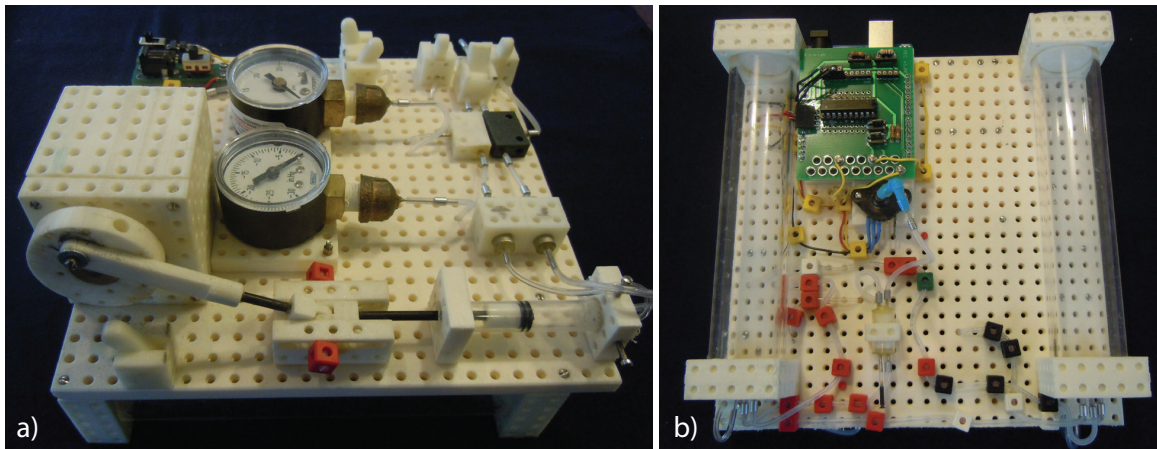


Figure 9.2: This photograph shows the top view (a) and bottom view (b) of the compressor and vacuum system. On the bottom are air lines held down by Snap blocks.

Shown in figure 9.2 is the front a) and back b) of our first compressor. This was a fully operational compressor, and was software controllable. It was able to reach pressures of close to 20 PSI and also support a vacuum operation. Because I actually wanted to use the compressor, one of the first software changes I made was to have Josh Ice write code to allow the Arduino to operate the compressor independent of the laptop on power up. A switch was used to tell the Arduino to turn on the compressor. With the Arduino we added a top level software loop so that when the switch was turned on, the compressor had a

default pressure it would go to. This way, it did not have to be connected to a laptop to be used. *Evolution: Start up code, independent of laptop control, that performs basic functions on a MEC system.* If a command did come in, the control code would jump out of the top-level loop and perform the command as usual. Another unique aspect of this device is that it has three different T joints. A T joint is simply where air comes in one connection and is split out to two connections. This compressor has three different T joints because they were evolving. Black and white T valves are visible in figure 9.2 a) top right and figure 9.3 e). The white one was 3D-printed with ABS plastic in an additive process. An additive process deposits small amounts of molten plastic to build a part. The problem with this process is that the resulting material is porous. To overcome this I ran acetone through the part, which would sometimes seal it. However, “sometimes” means that it was not a consistent process. The black part was a sample I had made by EnvisionTEC (Gardena, CA) on a 3D printer that uses projection curing technology. The resulting part is solid material, so does not have leakage problems. The key disadvantage to using EnvisionTEC’s printer was that their parts would have to be purchased.

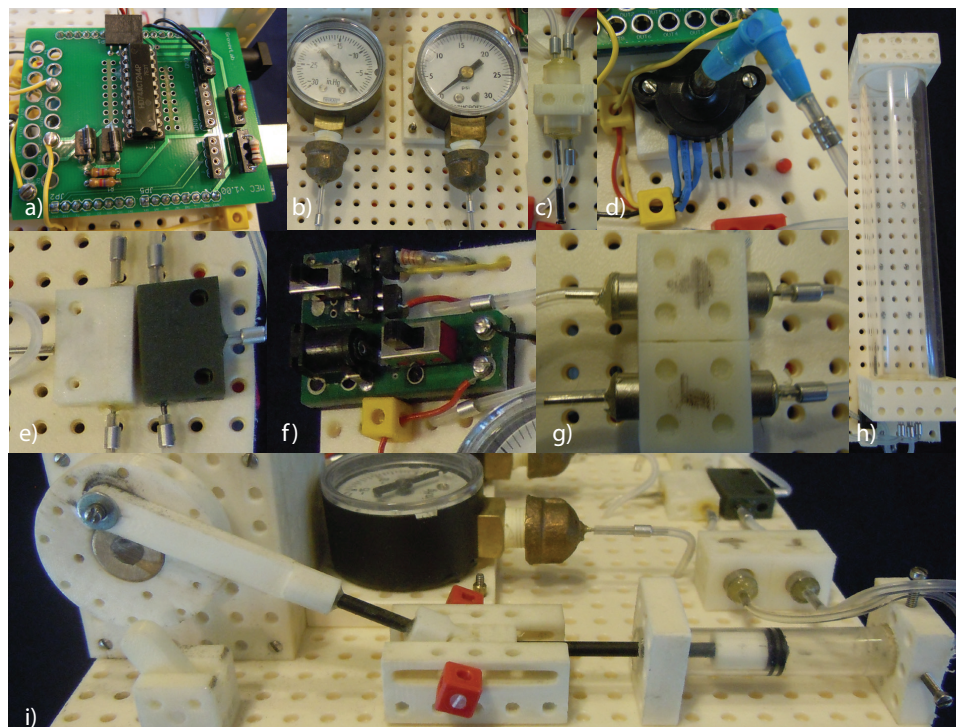


Figure 9.3: Included in the compressor and vacuum system are a daughter card (a), pressure and vacuum gauge (b), quad air joint (c), electrical pressure sensor (d), air T joint (e), Power connector and switch (f), one way valve (g), air chamber (h), motor, cam, and piston (i).

The T joint on the middle bottom of figure 9.2 b) or figure 9.3 c) is made using the same process we developed for making storage units out of syringes: 3D-printed end-caps with I/O pins epoxied in place and mounted on the ends of cut syringes. This part is labor-intensive to make. The positioning of the I/O pins is not on standard spacing. This design can accommodate a large number of interconnection joints. One can have up to 4 connections per side. We later found that some of the epoxy end caps sometimes had leakage problems. This was also the first system that required a calibration program. *Evolution: Calibration function is integrated into the MEC software.* The user would calibrate the pressure sensor (figure 9.3 d) so that at any sensor voltage the user would know what

pressure was associated with it. On the back, the daughter card (figure 9.3 a) is visible over the Arduino. It was designed to also accommodate a dongle for wireless communication. When comparing with the schematic, you should note that this system was made with manual switches as opposed to solenoid valves. This system worked well until we burned out the on-board FET, probably due to the voltage spike a motor is known to put out. We had determined that the elastic Tygon tubing had leakage problems with the 20 gauge tubing, and had started using small aluminum collars. *Evolution: Small Aluminum ring collar to stop air leakage at higher pressures (20 PSI).* The switch and power adapter were original Snap components. This system used the same type of air chamber (figure 9.3 h) and mechanical switches that were developed for the fluidic router. Finally, the pressure and vacuum gauges (figure 9.3 b) were double-sided and taped down onto a small standard Snap baseboard.

9.4 Developing the one-way valve

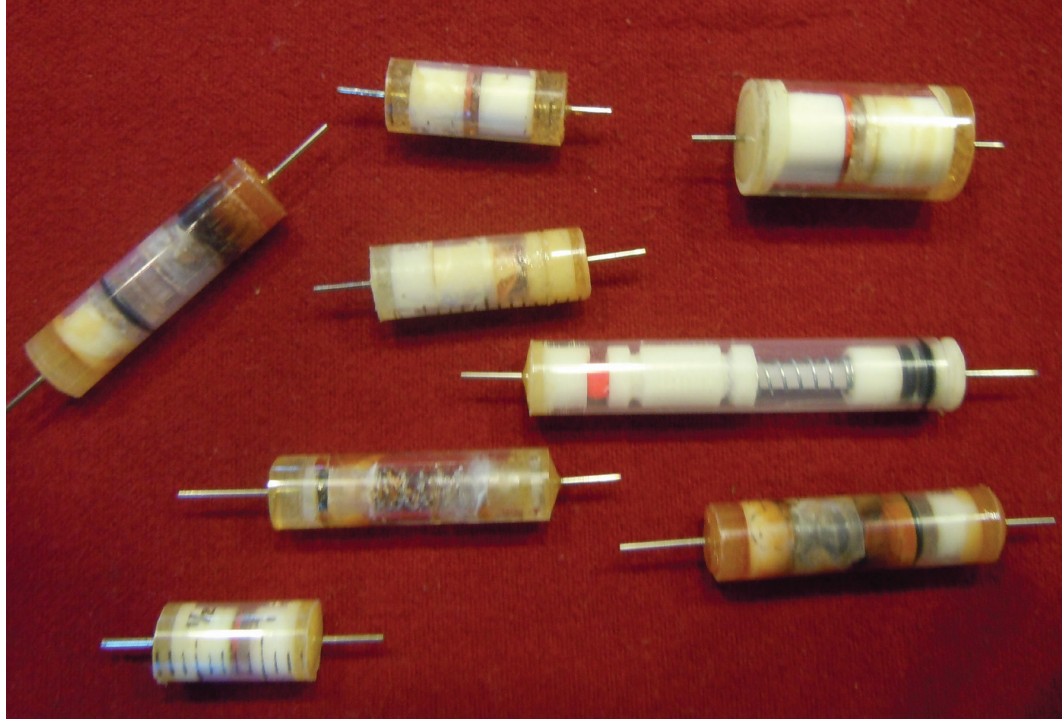


Figure 9.4: For every part at the TEC Center that works, there is often a junkyard of parts that did not. Most of these one-way valves failed because we could not get a good back seal.

Shown in figure 9.4 is a graveyard of one-way valves that either did not work or only partly worked. They were fabricated by pressing I/O pins into 3D-printed end-caps and epoxying them into place, a process that mostly worked. One of the methods tried here was drilling a hole into Plexiglass and then using a rubber flap. Thus, if air comes in one way, the rubber flap is pressed up against the hole and the pressure builds up, holding the rubber flap against the hole. If air comes in the other way, the rubber flap is pushed open. The problem was that we could not get the rubber flap to seal perfectly on the Plexiglass, so air would leak around the rubber flap. We tried using a spring to

pushing a piece of Plexiglass up against an O-ring. If air comes in one way, it helps hold the spring closed; air coming in the other way would build up pressure and eventually push the spring open. If this design worked as intended one could adjust the spring tension to adjust what pressure the spring opened the valve at. In the process of developing these valves, we learned that zinc plated steel springs rust. The deep brown visible on some of these one way valves is rust. The one-way valves were intended for air and liquid. The first application we wanted to use this on was the compressor. This is a strict application, because if there is any back leakage the compressor will not be able to hold pressure. I did not want to buy commercial one-way valves, due to their cost, and because I like to have full control over the interconnection method and form factor. Here, another intern (Casey Hill) found the solution: a company, miniValve (Oldenzaal, Netherlands), sells the internal part for a one-way valve. It was called a duckbill valve. We still fabricated the shell that the one-way valve fit into and therefore set the valve's form factor. The last problem we had to overcome was that the duckbill valve was very sensitive to the diameter of the tube it was in. After some trial and error, a stainless steel tube was found that gave a perfect seal. We still used 3D printed end-caps. The working version can be seen in figure 9.3 g) or figure 9.2 a). The arrow indicates the direction of flow. Notice the 3D printed holder for the stainless steel tube, which allows the tube to be mechanically screwed down to a Snap baseboard. The component was pressure-tested up to 20 PSI with no leakage in a bubble test. Real product development is at the core of the TEC Center, and students get to watch, participate, and be a viable part of that development. *Evolution: Purchased component (Duckbill valve insert) used for creation of the MEC one-way valve.*

9.5 The next generation compressor

Undergraduate intern Joey Milia worked on the next generation compressor. The current compressor had reliability problems so he redesigned the compressor's cam components. Utilizing mechanical analysis, he selected a more-appropriate motor. Joey will be using four solenoids valves so that pressure and vacuum can be charged simultaneously. The daughter card will be removed.

9.6 The compressor could be a “Smart Instrument”

A smart instrument is one that characterizes its own components and functions. It then uses this information to determine the instrument's health, thus being able to predict an instrument problem before it occurs. Its characterization information could be stored locally, or in the cloud. Cloud information could be compared to other instruments of the same type, providing information for normal operation. Since many of the components of the instrument have their own AVR microcontroller, characterization can be done down to the component level. With instruments that are fabricated in a quality-controlled environment, this level of diagnostics is usually not required. However, in our case, both our components and instruments are evolving in a research environment, so this type of diagnostics could be critical. It could provide unskilled users immediate feedback on their instrument, possibly informing them at assembly when components are improperly connected. Our software will have a diagnostic section. For example, with the compressor it could track the pressure curve as it comes up to pressure, then monitor how effectively it holds pressure. A normal curve range could then be determined. Over time, with cloud information, it would learn

what a normal curve is, and what an abnormal curve looks like. Abnormal operation could be caused by a leak, a motor with burned brushes, etc. If the compressor had a problem holding pressure, that would give a higher likelihood to a leakage problem. These are examples of functional characterization. The microcontroller has spare pins that can read A/D signals, providing a great source for diagnostic information. We could use these pins to measure voltage across the solenoid coil. Again, we would determine what was a normal characterization curve for these voltages. This characterization would be done in detail, and different failures would have different characterization curves associated with them, allowing early and detailed diagnostics of problems. *Future Evolution: Smart instruments that regularly do diagnostics at the component and system level, and share that information on the web, for early detection of problems.*

Chapter 10

Electrical Valve

10.1 An electrically controlled valve is a needed fundamental element in the MEC system

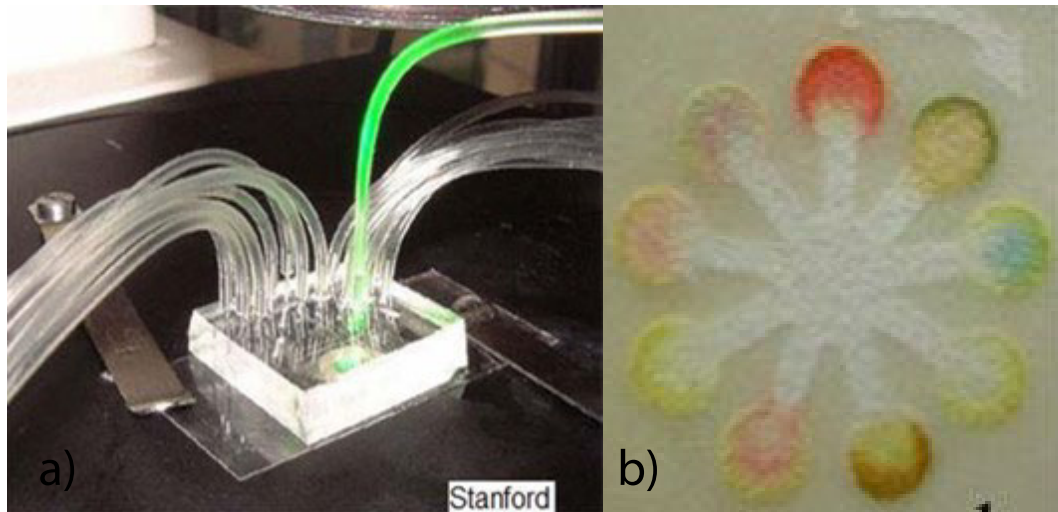


Figure 10.1: a) Example of a Medusa-head lab-on-a-chip microfluidic device [63]. The device has a high level of integration. b) Simple colorimetric indicators for paper-based microfluidic devices [21].

A fundamental need in the development of biological instruments is the control of fluids on multiple scales. The field of microfluidics has tried to solve this problem for the micro-scale. Different approaches to solving this challenge have emerged. The first approach is characterized by high interface device complexity: for example, integrating hundreds or even thousands of microfluidic valves and pumps into a single chip (figure 10.1a). These complex microfluidic devices can perform incredibly sophisticated operations on fluids and cells, and their solenoid-valve-driven microfluidic valves are easily controlled by computer. However, the custom designs of these chips and the amount of infrastructure needed to fabricate them are significant barriers to researchers interested in producing them. In addition, the expensive external hardware required to control these chips precludes their use in resource-limited settings, or in the TEC Center.

The second approach to microfluidics is characterized by low interface device complexity: for example, flowing fluid down simple strips of paper and detecting binding via a simple color change (figure 10.1b) [21]. As they require little infrastructure to make and are relatively inexpensive, these simple microfluidic devices can be easily developed and deployed in resource-limited settings. Yet simple microfluidics inevitably support far fewer on-chip operations than their complex counterparts, thereby limiting their utility. We are looking for a general, low cost valving solution that can solve a wide range of applications.

It is striking how these two approaches in microfluidics—complex vs. simple—differ in how they solve the control and user interface problem. These two methods result in useful instruments, but they still lack the broad range appeal to solving biological problems, due to either cost or flexibility. The complex interfaces come from the essential need to

connect microfluidic devices to computers. Computers are well-evolved, offering a powerful tool for both control and analysis. Figure 10.1a) is such a device with a complex interface. The lines coming off the device are primarily for control. These lines connect to solenoid valves which, in turn, are controlled by driver cards from the computer. Software is required for controlling the driver card. All this forms a powerful, complex, and unfortunately custom interface to microfluidics.

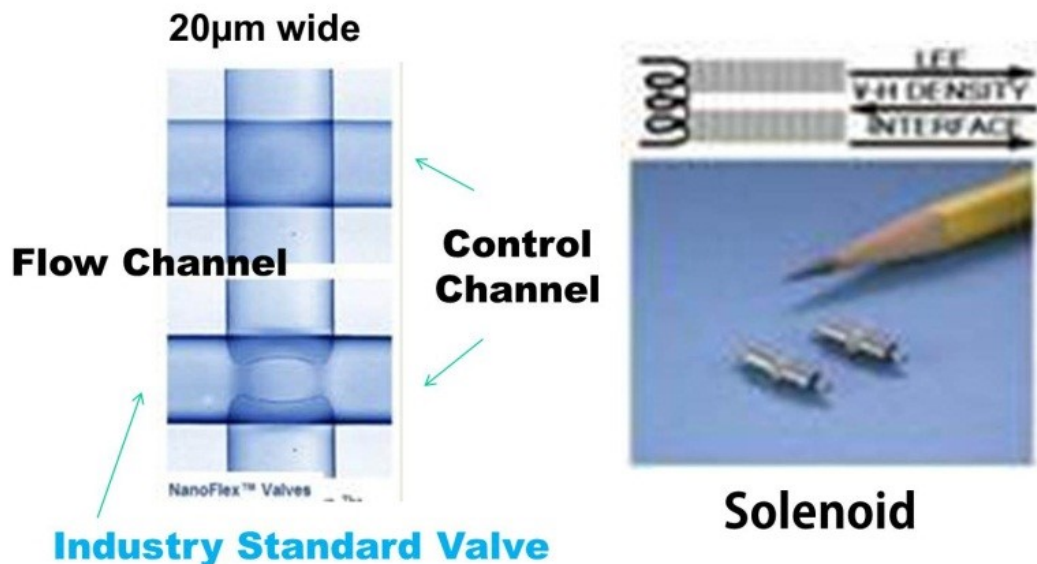


Figure 10.2: Standard microfluidic cross valve is small. Small solenoid valve required for turning off and on the microfluidic valve is relatively large (i.e. at a different size scale).

The interface and fabrication problem is complicated by the size mismatch between microfluidics (micron-scale) and off-chip control hardware (millimeter- or centimeter-scale). Figure 10.2 provides a comparison between the smallest of the microfluidic cross valve and the smallest solenoid valves. The device shown in figure a) may have a thousand pneumatic valves and over a dozen solenoids. The fact that these devices must work together, yet are at different scales, with separate fabrication methods, forms an obvious barrier to component-

based design. And yet, pneumatic valves are the primary type of valve used in microfluidics today. I believe that pneumatic valves linked to solenoids are at the heart of the barrier to component-based design [22] [23] in microfluidics.

Several methods have been proposed to reduce the number of control lines coming off of a microfluidic device, such as the creation of on-chip valve-based multiplexers [23]. For example, a multiplexer could allow three solenoid lines to turn on and off up to eight pneumatic valves, or four solenoid lines to control up to sixteen valves, etc. A second method to reduce the number of external lines is the creation of an on-chip valve-based state machine. Here, a set of valves can be configured to become a clock [24] which, when combined with valve logic, can turn off a set of valves in a controllable set sequence. All of this has parallels with electronics. In fact, this type of state machine can be thought of as an early computer. These approaches can reduce or eliminate the cost associated with off-chip control hardware. However, all of these extra valves utilized to create multiplexers and a state machine reduce flexibility of the device and increase microfluidic valve overhead. Multiplexers and state machines reduce the functionality of the device, since all combinations of open and closed valves are no longer possible.

An electrically controlled valve will allow us to use the full power of the computer for device control. I believe we need an electrically controlled microfluidic valve that can be fabricated in a single process, much like the transistor in electronics.

10.1.1 Existing electrically-controlled microfluidic valves

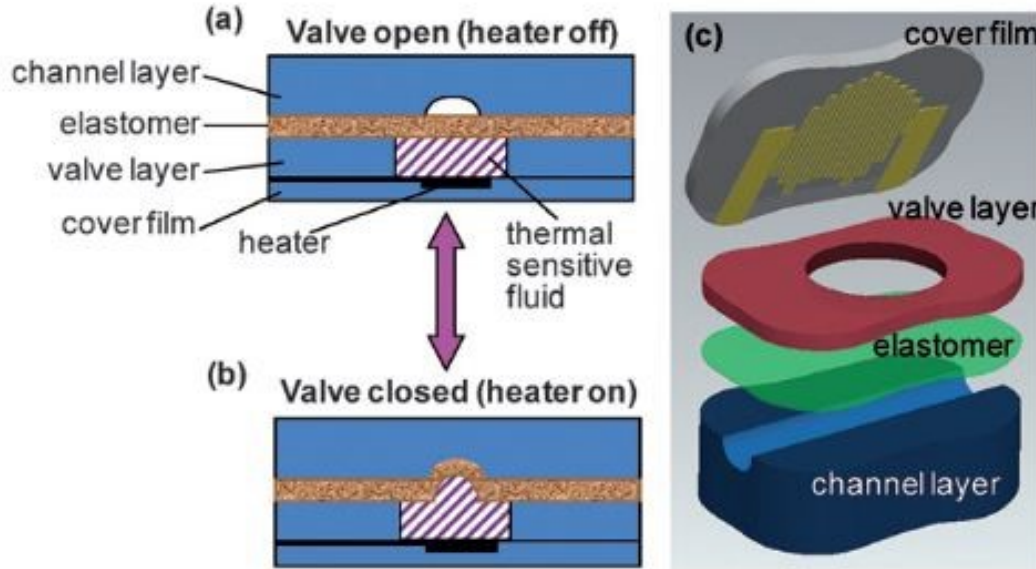


Figure 10.3: An existing electrically-controlled microfluidic valve. FC40 and FC70 are temperature-sensitive fluids can be used as actuation fluids in the design of a thermal valve [22].

To remove the solenoid and move to component-based design, I started developing an integrated, electrically-controlled valve for use in the MEC system. Some work has already been done on electrically-controlled microfluidic valves, but no accepted solution has emerged. The purpose of a valve is to control fluid movement. One solution does not require an actuator. This method utilizes electro-osmotic flow, but only applies to specialized applications [23]. We are focusing on more general solutions, which would use an electrically controlled actuator. Several materials can be used for the actuator, including thermal expansion or contraction materials[22] [25], thermally viscous hydrogels [26], pH-sensitive hydrogels, magnetoresistive material, and electro-active polymers (EAP) [27] [28] [29]. Microfluidic valves using paraffin wax, fluorocarbons (FC40) [22], and thermal hydrogels[25]

are examples of thermal materials that have been demonstrated. Their primary problems are slow actuation times and processing complexity. The first thermal contraction material I considered is a thermal hydrogel, N-isopropylacrylamide (NIPAAm). An example of its use in a design is shown in figure 10.3. This can be electrically actuated, however actuation speeds are slow. Further development is needed to improve cost, speed, reliability testing, reproducibility, and ease of fabrication. Actuation times are reported at 1 to 10 minutes [23]. Each method will require overcoming ease of fabrication and reproducibility. This thermal hydrogel has the additional requirement that arises from the need for a water reservoir.

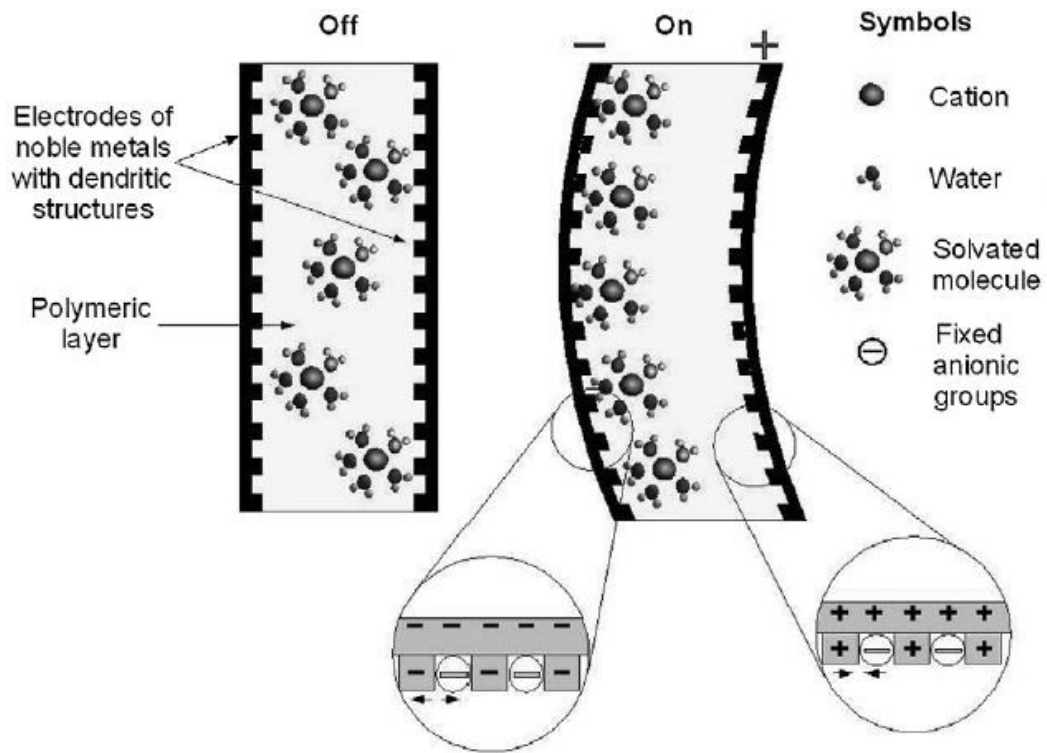


Figure 10.4: An electro-active polymer sheet, with electrodes plated on both sides. The sheet flexes when 1 to 10 volts is applied [29].

Electro-active polymers (EAP) are being developed for artificial muscles [29]. These polymers change shape with the application of an electric field. There are several types of electro-active polymers [30]. I focused on electro-active ionomeric polymer-metal composites (IPMCs). This is due to their high deformation induced from low DC voltages. This polymer comes in sheet or bulk form. The electro-active polymer sheets are approximately 200 μm thick. In figure 10.4, electrodes have been plated onto both sides of the sheet. When a low voltage (1 to 10 volts) is applied to a sheet, the material flexes. This flexing occurs for two reasons: First, the solvated ions are drawn to the negative electrode, causing water swelling on negative electrodes and water depletion on positive electrodes; and second, negative ions on negative electrodes are pushed apart [29] [31].

10.1.2 Designs of MEC electrically-controlled microfluidic valves

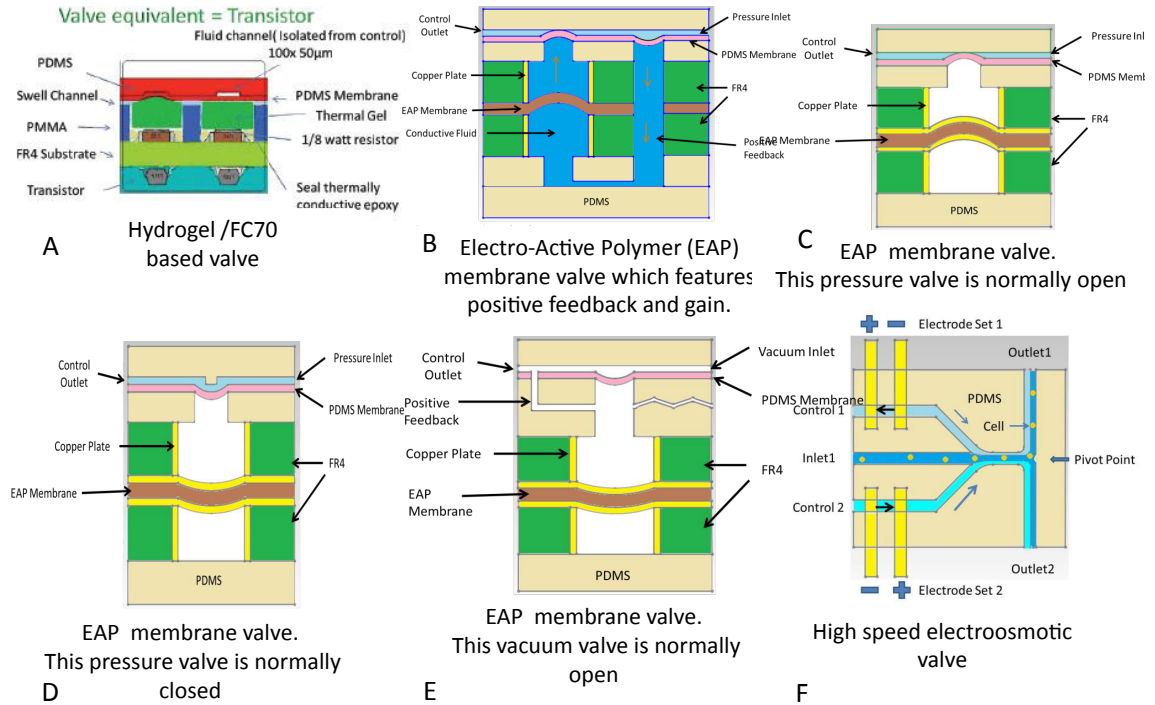


Figure 10.5: A hydrogel linkage collapses at a thermal threshold, opening the valve. FC70 has a high thermal expansion coefficient, closing the valve when heated (A). An electro-active polymer flexes when a voltage is placed across the membrane. I designed a family of valves with this material in mind. The first EAP valve (B) uses positive feedback, so when closed, pressure builds up on the inlet, further holding the valve closed. The EAP valve can be designed to be normally open (C), and normally closed (D). Valve design needs to be adjusted whether one is working with a line under pressure or vacuum. I designed a vacuum valve (E), with positive feedback. When the valve is open, the feedback helps hold it open. The high speed valve does not have an opening and closing valve. Instead, it uses osmotic flow to adjust the flow of fluid down a channel, so that a particle flowing down a channel will either go left or right.

The first valve shown in figure 10.5 a) is a valve actuation method using a thermally expansive or contracting material, and a heating element. In addition to thermal Hydrogel (NIPAAm), I would consider FC70 or FC40 fluorinated carbons, which have a high thermal expansion coefficient. They have the advantage that no water reservoir is needed. The

materials are readily available. The channel that it closes is similar to the standard cross valve used in microfluidics. Draw backs include ease of fabrication, and a slow actuation speed. The configuration shown in figure 10.5 a) is a normally open valve, which is counter to our preferences. This expansion presses on an elastomer membrane and closes a channel. Looking at the article on this valve, the heater power ranges from 26mW to 80mW [22]. A three to one ratio was found between the calculated temperature change required of 8.7 °C to close the channel and the actual temperature required of 27 °C. For power greater than 50mW, the actuation time was 7 to 80 seconds [22].

As we examine different valve designs (10.5), the requirements would include ease of fabrication, ability to be made to meet MEC system mechanical and fluidic requirements, non-contamination of samples, and reliability. The next four valves in figure 10.5 b,c,d, and e utilize an electro-active polymer.

Some of the valves have incorporated positive feedback to improve reliability. This feedback methodology is related to that which is used in the common sprinkler solenoid. Some of the functional differences of these valves include whether they are normally open or closed. Typically, I prefer a normally closed valve for most applications. Some of the other differences include whether the valve was designed for a control vacuum line, pressurized air line, or fluid line. Ultimately, we will need a range of valves, with different functions, and abilities to control different scales of volume. *Future Evolution: Electrically controlled valve for microfluidic scaled device, that can be fabricated in the same process as the microfluidic device.*

10.2 Early electrically controlled valve device

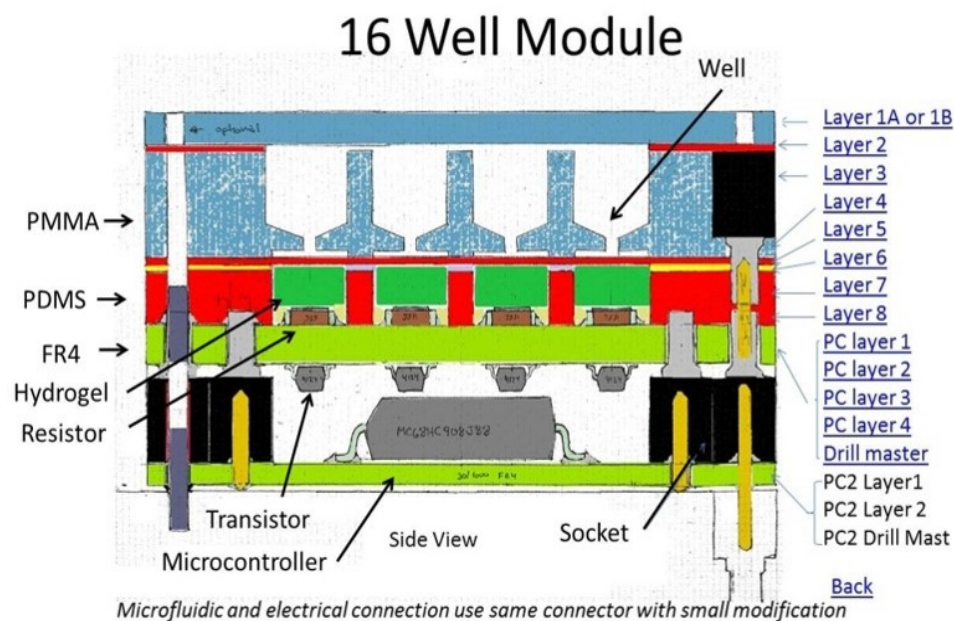


Figure 10.6: An early MEC component concept design. In this component, multiple electrically-controlled valves integrate into one device. There is an on-board microcontroller which receives its instruction through serial communications, and an optical window through which samples can be analyzed.

Shown in figure 10.6 is a design for a 16-well storage and imaging module with electrically-controlled valves. This module would not require off-module solenoids and thus realizes the tremendous value of electrically-controlled valves in component-based design. The module was designed with standard 0.3-inch spacing for both size and interface spacing of both fluidic and electrical connections. Its operation would start with serial commands to the on-board microcontroller. The microcontroller would turn on a transistor, which in turn would apply power to a resistor, which would heat a hydrogel, drying it out and causing it to contract. This in turn opens up the lower valve, allowing fluid in or out of the storage wells.

These valves design are normally closed, which is optimum for this type of device. Integrated connectors were used for both fluid and electrical connections. The module was designed for component-to-component compatibility. This means that a component of a similar type could be plugged in above this device, thereby doubling the number of storage wells. The structure of this device would come from the lamination of four different materials: glass (top), PMMA (storage wells), PDMS (valves), and FR4 (control). Even though most of these materials have been shown to be able to be laminated, this lamination is the real drawback to this design, especially since the design needs to have a complete seal between all layers to stop leakage. This module would require the integration of components from several different processes. If one can achieve a single process, scalability, as in electronics, becomes possible. Shown on the right side of the figure are the layers that would be required for the complete fabrication of this device.

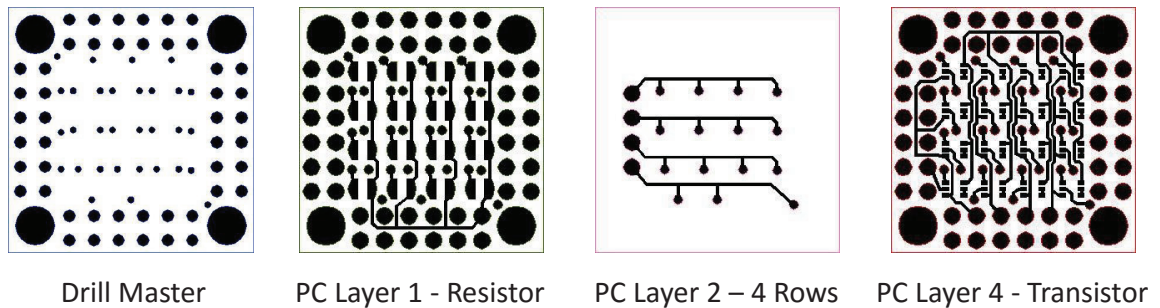


Figure 10.7: The printed circuit board electrical layers for the 16 well storage module. Surface-mount thermal resistors are on layer 1. Surface-mount transistors are on layer 4. PC layer 3 is a ground plane and not shown. The lower printed circuit board and its layers are not shown.

There are two printed circuit boards in the 16-well module shown in figure 10.6. The top printed circuit board has four layers. Internal layers are for ground and traces. The top layer is for mounting the thermal resistors. The bottom layer is for mounting

surface mount transistors (figure 10.7). The second printed circuit board would house the on board microcontroller. Sockets would be used for electrical connections outside. The PC was designed with no smaller than 10 mil lines and 10 mil spaces.

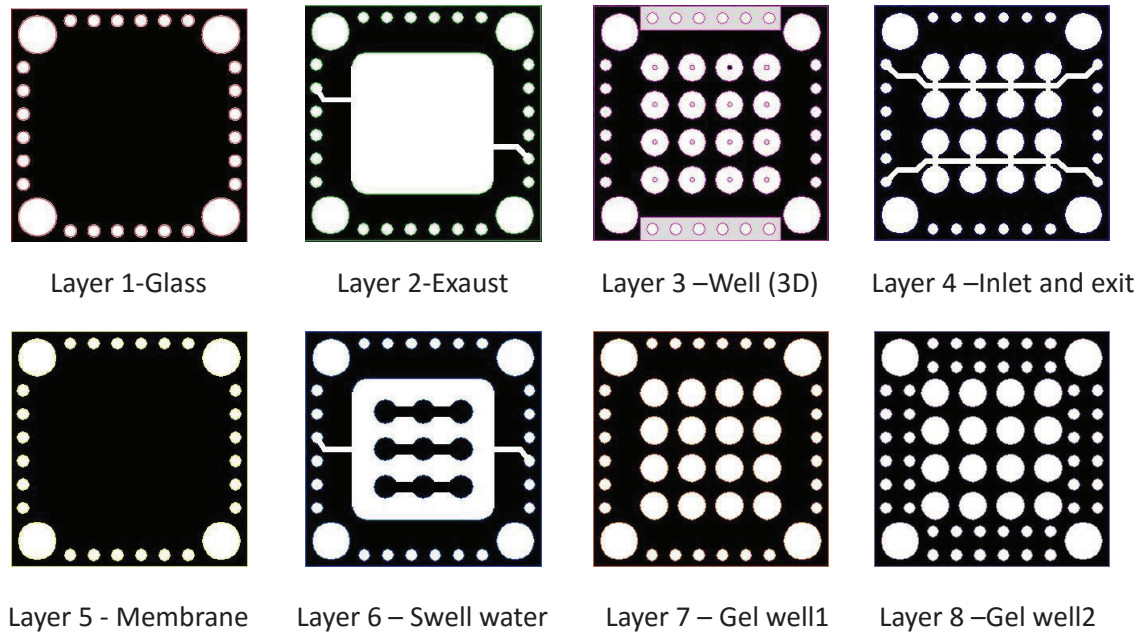


Figure 10.8: Microfluidic layer designs for the 16-well module . The hydrogel requires a fluid inlet line to allow for hydrogel swelling, which closes the channel.

There are eight fluidic layers shown in figure 10.8. Layer 1 is made out of glass, so that optical imaging of wells is possible. Layer 3 is a three-dimensional layer made from PMMA for the storage of materials. The bottom of layer 3 goes to a small opening which can be shut off by the layer 5 membrane. Layer 4 is a thin layer which has channels for material to go in and out of the storage wells. Layer 5 is an elastic membrane, which stretches across layer 4 to seal the opening in layer 3. Layer 4 is designed to supply water to the hydrogel, allowing it to swell again after cooling. Layer 7 is designed for housing the hydrogel.

This module was one of the first integrated electronic and fluidic components that we designed. The one characteristic that would need to be addressed for its continued development would be the complexity of the fabrication process. Sample storage is an essential component in the broad range of biological applications. Without the need for external components, these storage units can be assembled in parallel to whatever level the application requires. This component provided us a vision for what a multidisciplinary fluidic component might look like. It contains many of characteristics we are striving for in component-based design.

10.3 Solenoid Valve, H-Bridge and Testing

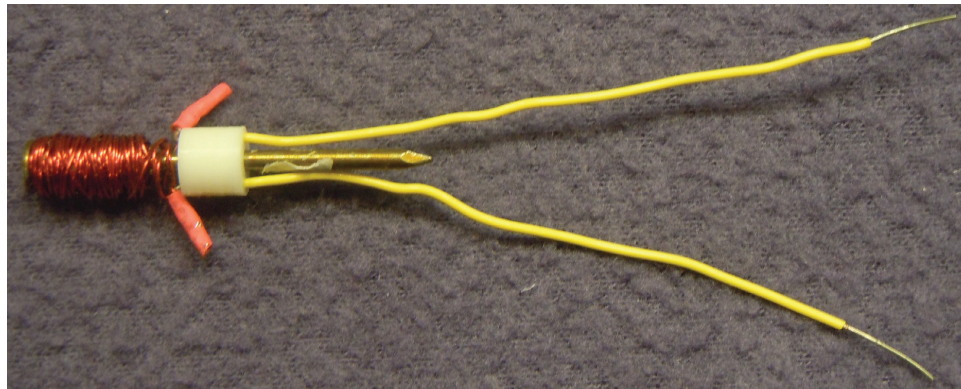


Figure 10.9: Electromagnet made from a nail, a coil of wire, and a 3D-printed base. It was intended to apply a magnetic force to another rod which was to close a valve. The rod it pulled on was external to the electromagnet. This electromagnet's magnetic force was far too low for valve actuation.

My very first valve design used an electromagnet that pulls back a rod. This rod was held forward with a spring, opening up a channel. What I was surprised to discover was that a simple electromagnet such as the one shown in figure 10.9 generates a very small

force, only a fraction of the force required. So, how does a solenoid valve in a sprinkler system work? I went online and began to study electromagnets. A couple of things that I learned are: First, once a rod is in direct contact with an electromagnet, the attractive force is substantially increased. For example, a small bucket of ball bearings can be held up with an electromagnet. Second, if one can get very close to the electromagnet core, the magnetic strength is also much higher. Third, an armature surrounding the electromagnet inside and out can also help focus the field, thereby strengthening it. Finally, more coils can also help strengthen the field if one has a limited current supply.

Adding more coils to the electromagnet may not have as strong of an effect as one might guess due to the increase in resistance that the coils add. In our case we are using a 5 V wall plug with current rating of 1 A , so if the coil has a resistance of less than 5 Ω , then current through the coil would be limited by the 5 V power supply. In this case, adding more coils will increase the magnetic field strength.

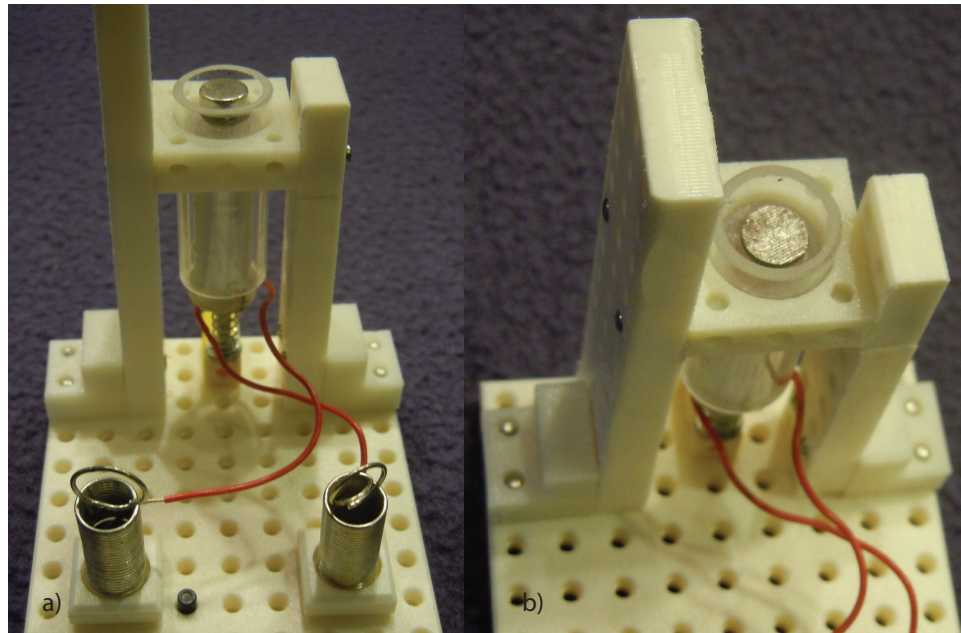


Figure 10.10: Solenoid valve test station. We greatly increased the number of coils in the solenoid valve. The metal rod pulled by the electromagnet is internal to the coil, and we added a permanent magnet. The magnet is able to contribute its force to opening the valve (required to overcome spring force), reducing the force needed from the coil. Once open, no coil force is required to remain open, so this assembly has a latch state. Reversing the direction of current flow through the coil will release the rod and close the valve.

In my next design I incorporated what I learned from earlier versions. First, I added significantly more coils to the solenoid valve. Second, I used two cut nails facing each other. One nail was fixed in the coil, and the other could slide up and down and was held down by a spring. Critically, the gap between the two nails had to be very small. Our tubing had a $1/32$ inch inner diameter (0.0312 inches or 0.794 mm), so that would be the maximum gap needed. Finally, I built the valve to fit in the Snap system, as shown in figure 10.10. At first, it still did not work, but I happened to have some small neodymium magnets, and I placed one on the top nail. The nail rod was capable of being lifted when the force from the magnet was added to the force caused by the current running through

the coil. One very important benefit of adding the magnet was that when the bottom nail came in contact with the top nail with the magnet, a very strong contact force formed. One could even turn the current off and it would still hold open. Thus, I had developed a latching solenoid valve. The one additional requirement was that to release the solenoid valve, the direction of current flow had to be reversed. The latching solenoid valve is very important: if we had a system that needed, say, 10 solenoid valves on at a time, one would need 10 amps of constant current, which very quickly becomes restrictive. Using latching valves, current is used only when the valve state is changed. *Evolution: Adding a magnet aids in adding force in opening a solenoid valve, and once open, can hold it open even when power is removed. Thus, this creates a latching solenoid valve.*

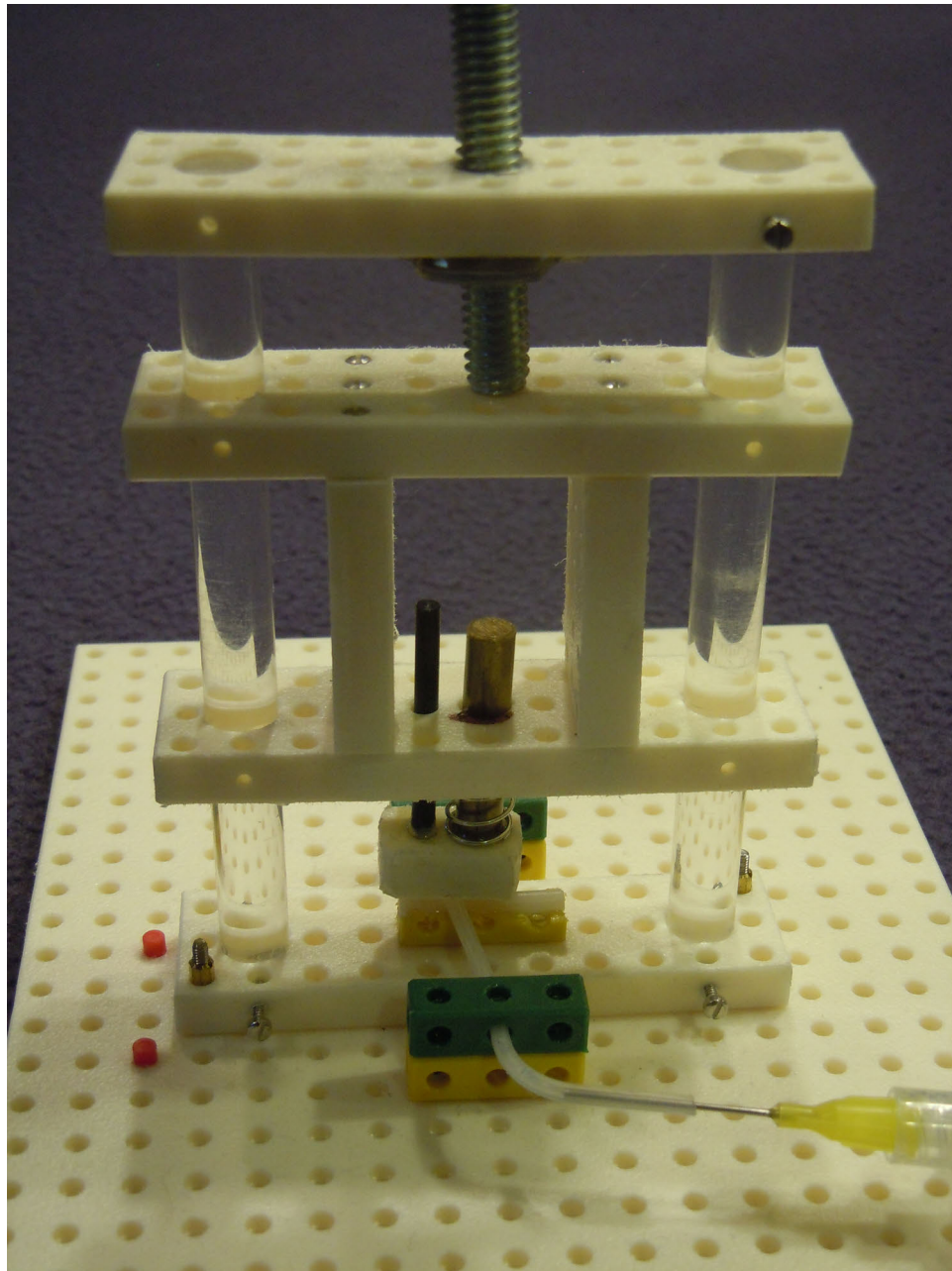


Figure 10.11: Test jig to determine spring force required to close Tygon tubing channel

Next, I switched to a larger core and a different spring. I wanted to determine the force that was required to pinch Tygon tubing closed, so I developed the test jig shown in

figure 10.11. As one tightens the screw at the top, it compresses the spring. At a certain point, liquid or air will no longer flow through the Tygon tube line. One then can measure how compressed the spring is in order to appropriately design one's solenoid valve. As it turned out, there must have been more friction in the final valve design, because more compression force was needed than what was measured using this rig.

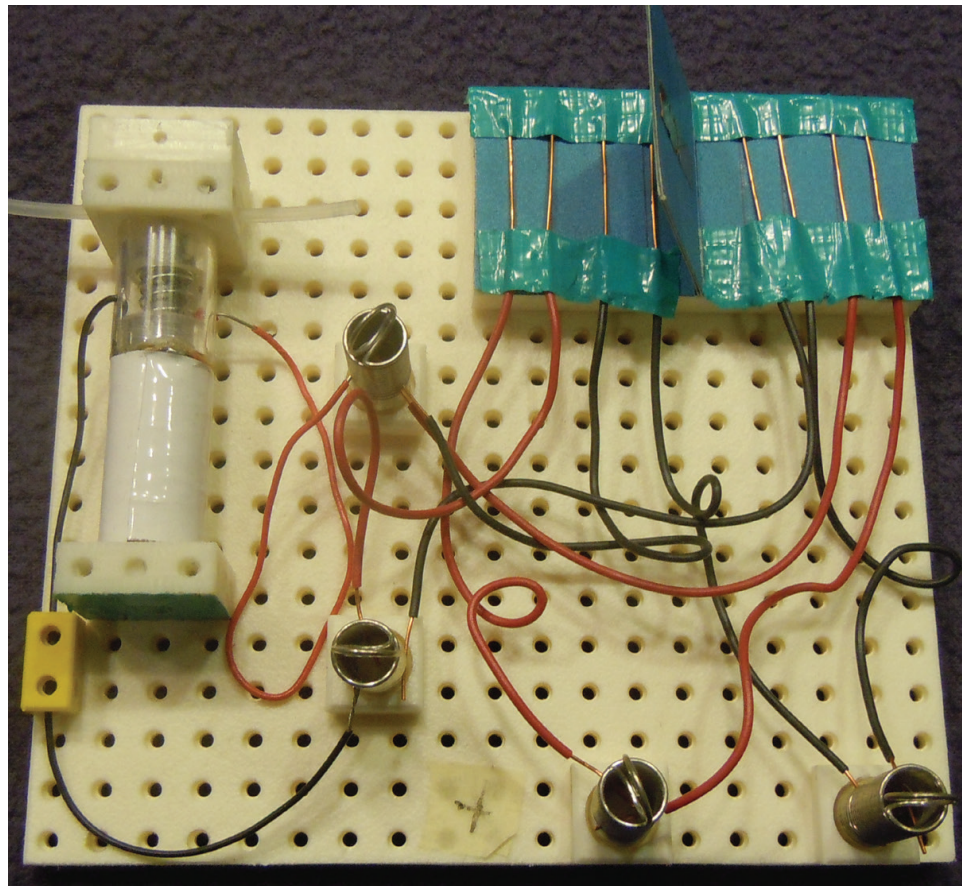


Figure 10.12: Solenoid valve test station. This station includes a paper H-bridge to drive current through the coil. This is also a great educational tool for showing how an H-bridge functions.

Figure 10.12 is one of the first functional latching solenoid valves. It utilized 1/4 inch iron core rod instead of a cut nail. It used a spring from a spring kit from Harbor

Tool and Freight. It also utilized Plexiglas tubing and 3D-printed parts. I also developed the paper H-Bridge shown. As the paper H-bridge is pressed to one side it runs the current through the solenoid valve in one direction. This pulls the solenoid valve piston up to the latch-up position. In the latch-up position the permanent magnet holds the rod. When the paper H-bridge is flipped to the other side, current flows in the opposite direction, releasing the piston and pinching off the Tygon tubing. 5 V from a 1 amp 5 V regulator is applied to the two springs on the bottom right of the figure. A voltmeter can be placed across the other two springs, measuring the voltage across the coil. We were able to demonstrate a fully functional solenoid valve, but there was some instability to the device.

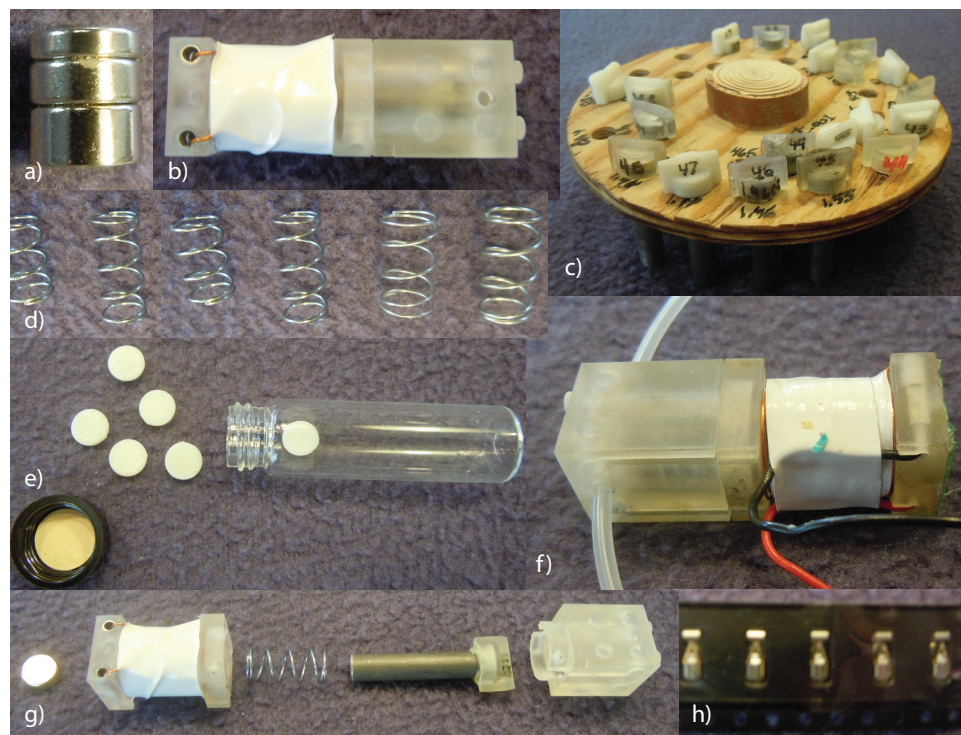


Figure 10.13: The assembly of the MEC solenoid valve (g) is made up of several components: a permanent magnet (a), a spring (d), a 3D-printed spacer (e), a pinch rod (c), and connector pins (h). The two wires connected to the connector pins can be seen in the assembled solenoid valve (b). From another view one can see the 22 gauge wire that connects to the outside (f).

We then evolved, and continue to evolve, through several iterations of the solenoid valve. The solenoid valve shown in figure 10.13 b) and f) is close to the version we now have at the beginning of 2016. Some notable improvements over the solenoid valve shown in figure 10.12: First, it is fully 3D printed with no Plexiglas component. Second, the original solenoid valves had wires that came from the coil, soldered to larger wires. The wires had a fixed length. Here, we run the wires from the coil to small sockets. Wire can then be plugged into the sockets connecting it up to the rest of a system. The small red magnet wires from the coil are shown going into holes (figure 10.13 b) where they solder to the small sockets (figure 10.13 h). Third, base dimensions are 2x2 versus 3x3 Snap units. Finally, the top coil and base twist together.

The coil is made from 32-gauge magnet wire with 6 layers at 30 wraps per layer, and has an overall resistance of approximately $1.5\ \Omega$. As we developed the solenoid valve we primarily varied three things: First, the spring, as shown in figure 10.13 d) was altered to vary the spring tension. Second, we varied the small neodymium magnet size (figure 10.13 a). The magnet fit on the end of coil, so as different size magnets were used, we developed different 3D printed spacers (figure 10.13 e). Third, we varied the rod length. I created an array of different length rods, which varied by approximately 1/1000 inch.

We now consider how the solenoid valve operates to see how these design factors interplay. In this design, when the solenoid valve is up, the rod is in direct contact with the magnet (maximum static holding force). At the same time, the spring is at its maximum compression (maximum static pushing force). In the up state the tube that runs through

the solenoid valve is no longer pinched, allowing fluid to flow. When I use the term static it means the coil is not energized either up or down.

- Condition 1: For the solenoid valve to stay latched, the maximum static holding force created by the magnet must exceed the max static pushing force created by the spring.
- Condition 2: When current flows in the correct direction the iron core rod (figure 10.13 g) begins to reduce the pull of the magnet, and when the pull of the magnet is less than the force of the spring, the rod releases from the magnet, going down to the pinch state.
- Condition 3: Here, in the static pinch state, the force of the spring must be strong enough to overcome the pull of the magnet and the push back from the elastic tubing (including its internal pressure) to completely stop flow in the tubing.
- Condition 4: When the rod is in the pinch state and current flows in the opposite direction, the rod is pulled back toward the magnet. When the pull of the magnet plus the force of the electromagnet is stronger than the spring force, the rod is pulled back to the up state.

Thus, as the rod length is lengthened it gets closer to the magnet, making it easier to pull it back into the up state. If the rod gets too close, no current may be required and the magnet may be sufficient to pull the rod back. In this case the rod will remain in the up-latched position; a fail condition. Notice that this design can allow for much lower coil forces because the rod length can be such that it is almost ready to be pulled back, and only a small force is required to actually pull it back. One test we run to see if condition 2

is met is to remove the coil from the base and push the rod up into the latch state. If the magnet is too strong or the spring too weak, one will not be able to release the rod from the magnet. As we worked with these different configurations, several disparate functioning solenoid valves emerged. If the spacer is placed between the rod and the magnet, the rod can no longer come in contact with the magnet, thus making it so that when the coil is no longer pulling the rod up, the spring will push it back down to the pinch position. This configuration has several advantages. First, the valve can get shorter “on” pulses, because the solenoid valve no longer has to go to a latch state. Second, the valve no longer needs an H-bridge since one only needs to energize the coil to pull the rod up. The spring pulls it back down. We call this a pulsing solenoid valve, as opposed to a latching solenoid valve. The obvious disadvantage is that if one needs the solenoid valve on for any length of time, say over a half a second, this would require a relatively large amount of current and the solenoid valve might start to heat up. This pulsing solenoid is normally off, so this is a normally-off pulsing solenoid valve. We also found that if we utilized a slightly longer rod, at a certain length the magnet was strong enough to pull the rod to the up state, even when the current was off. Here, we found a position where, when the coil was energized to release the rod, one could get a temporary pinch condition. This, we called a normally-on pulsing solenoid valve. *Evolution: Spacer allows us to create a pulsing solenoid valve, either normally on or normally off.*

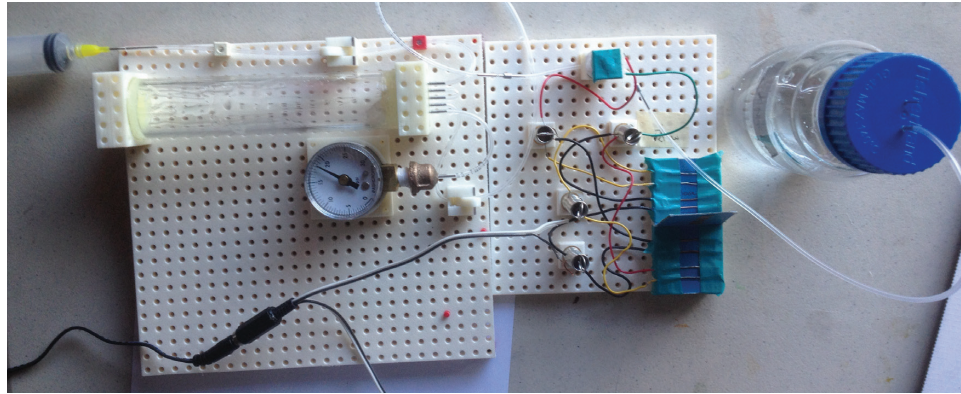


Figure 10.14: Complete solenoid valve test jig. To test the solenoid valve, we first connect the solenoid to the bubble bottle and the air chamber through a mechanical switch. Next, we use a syringe to increase pressure in the air chamber up to 20 PSI, which will be indicated on the gauge. We then turn on the mechanical switch. If no bubbles appear, we pass our first leak test. Next, we use the paper H-bridge to apply current to the coil, opening the valve. The valve latches on, and air flow is evident from the stream of bubbles. Finally, we reverse the H-bridge, closing the valve.

Shown in figure 10.14 is a complete solenoid valve test set-up with the solenoid valve described in figure 10.13. To operate this test jig, the user opens the top mechanical valve, pushes air into the chamber using a syringe, closes the mechanical valve, and repeats this process. This builds up pressure in the air chamber. I usually build up pressure to 20 PSI. Some solenoid valves work below 10 PSI, but will start showing leakage at 20 PSI. After the pressure chamber is charged, the user then turns on the second mechanical valve, releasing air to the solenoid valve. Assuming that the solenoid valve is in the down pinch state, no air should pass through. The line from the solenoid valve runs to a stainless steel pin on top of the bubble bottle. Inside the bottle, the same stainless steel pin is attached to a tube which continues to the bottom of the bottle. These are regular lab bottles that have had two 20 gauge stainless steel pins epoxied into their lids. If any air does pass through the solenoid valve, one will see bubbles in the bottom of the bottle. This

is a very effective method for testing the solenoid's valve ability to shut off an air line. One then flips the paper H-bridge. Current will flow through the coil, pulling the solenoid valve to the up state. For a latching solenoid valve, even after one moves the H-bridge back to the neutral position, the solenoid will stay open, and bubbles will still be observed. Then, flipping the paper H-Bridge in the other direction energizes the coil in the opposite direction, and the bubbles stop. *Evolution: Testing is an important part of any component development program. Notice the use of standard MEC components in the creation of the test instrument.*

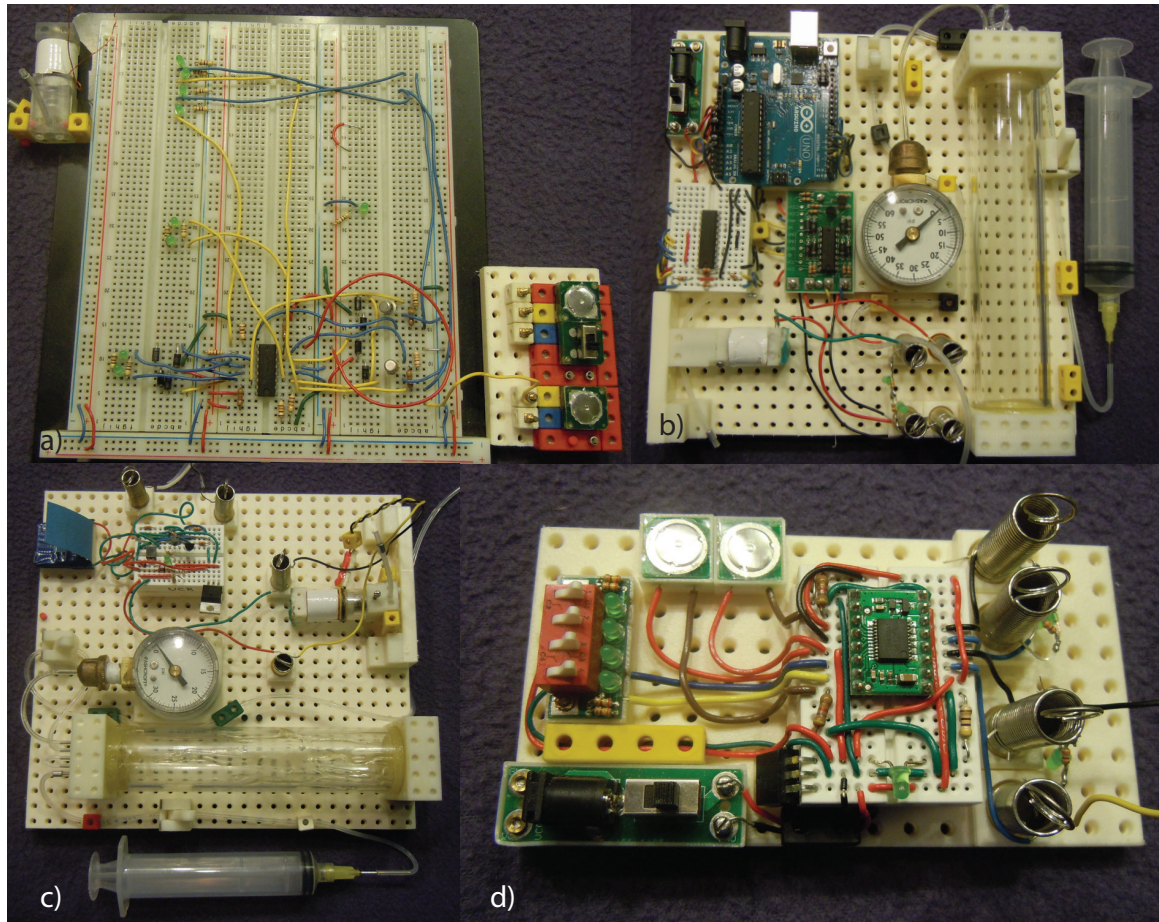


Figure 10.15: As the solenoid valves evolved so did the test jigs. The electrical H-bridge breadboard used to test the solenoid valves (a). We used this design to create a printed circuit board H-bridge version that we used to test solenoid valves (b). We also developed a pulsing solenoid valve (non latching). With this valve, an H-bridge is not required. A single field effect transistor (FET) can be used. FET-based pulse solenoid valve test jig (c).

The paper H-bridge is simple, excellent for teaching the principles of an H-bridge, and works well for testing the functionality of the solenoid valve, but solenoid valves are much more useful when they can be electrically controlled by a microcontroller. Therefore, we next looked at electrical H-bridges. There are several available, and the first one we selected was the SN754410. It is shown wired up in figure 10.15 a) on a breadboard. Instead of a microcontroller, we initially controlled the H-bridge using a button and a switch. The

switch sets the direction of the H-Bridge, and the button allows us to apply a pulse of current into the solenoid coil. We were successfully able to test the solenoid valve when it used a light spring and small magnet. We fabricated the circuit in a printed circuit board, and built up the test instrument shown in figure 10.15 b). This is the same test set up as in figure 10.14 except with electrical control. An Arduino provides the interface between the laptop and the H-bridge. We utilized Python as the control code in the laptop, and were able to successfully control the solenoid valve.

One of our first applications was the Coulter system. Unfortunately, the Coulter system needed a short pulse of fluid. Because one needs to turn a latching solenoid on and then turn it off, the latching solenoid valve is not well suited for a short pulse. The pulse solenoid valve is much better suited for producing a pulse of fluid. In addition, the pulse valve no longer needs an H-bridge. Instead, a field effect transistor (FET) can be used for turning on the current, and the spring brings the rod back down to the pinch state when the current is turned off. I used an IRF D014 for our FET. We then built up the test circuit shown in figure 10.15 c) with the FET circuit. Instead of a microcontroller, we used a paper switch to apply 5 V to the FET circuit. As we tested different springs, magnets, and solenoid designs, we found that a 12 V drive current worked over a wider range of parts. Also, for stiffer springs we found that 5 V driving the FET gate was not fully turning on the FET (the FET is limiting the current). We added some transistors so that 5 V turned on a transistor, which applied 12 V to the gate of the FET, providing more current to the solenoid. The basic operation of figure 10.15 c) is the same as b). Later, I taught Ryan Milton, an undergraduate intern, about FETs and how to read an electronic specification

sheet. He also spent his own time studying FETs. We wanted to select a FET that had low on-resistance when turned on with 4 V (we used 4 V to allow a design margin). He went through hundreds of FET specifications online to find the right device. At the time, we were developing a mixing system for which Ryan was designing a printed circuit board of a FET driver board for our new muscle wire valve. *Evolution: In the process of developing a mixing system, Ryan Milton learned: printed circuit board design, how to read a specification sheet, the function of a FET, and Visio software. Example of need-based learning*

We also found that the paper H-bridge (figure 10.14) worked over a wider range of parts than the electrical H-bridge. This is because the electrical H-bridge does do some current limiting. One of my interns, Chip Pearson, had worked with H-bridges with robotics, and he suggested we use the Pololu TB6612FNG dual motor driver module. We built up the H-bridge test circuit shown in figure 10.15 d) with this part. *Evolution: Pololu TB6612FNG dual motor driver module has better drive capability.* This electrical H-bridge did work over the wider range of parts we were looking for. Moving to a stiffer spring allowed us the ability to develop a dual solenoid valve, which could turn off two flow lines at a time. Also, a stiffer spring seemed to make the valve more reliable because the valve was less sensitive to changes in friction on its moving parts. We often had to sand the solenoid valve parts because of the inaccuracy of 3D printing, resulting in changes in friction between solenoid valves. Also shown in figure 10.15 d) are three new parts: The MEC quad-rocker switch, power plug, and dome button. All three are similar to the Snap Circuit versions of these components, but utilize a new small socket instead of the Snap pin interconnection method. This small socket allows wires to be plugged into the components as shown in the figure.

be designed to record solenoid operations over hundreds or even thousands of operations. Its operation would be designed as follows: First, Python code running on a laptop would verify the Adriano and AVR microcontrollers are present and running the correct code. Code used would be recorded in test results. The operator would then select the test mode. Next, the operator would enter the serial number of the solenoid valve under test. The serial number would be associated with a number of solenoid valve parameters: Solidworks STL file names for all three solenoid valve components, spring (length), magnet (latch force if available), rod length, 3D printer used for fabrication, solenoid configuration (latch or pulse), and any post-printing fabrication. The operator would enter these values if they are missing. When new solenoids are entered, the prior solenoid valves would serve as default values. Two solenoids can be tested at a time, although additional solenoid valve test stations could easily be added. After entering all the data, the user would then start the test. Lab air would then be released into the pressure chamber. Each test mode would have preset values. These can be modified by the user and would be recorded on the test results. The test results not only record the solenoid valve's performance, but also all relevant conditions it was under while during testing, such as software and hardware used, and any change in preset values. This should allow the test to be easily duplicated. In test mode A, the pressure is set to 20PSI, and we are testing a latching solenoid. After solenoid valve 1 is turned on, the user records the rise in air pressure (Fig. 10.16). Prior to filling the tank, the pressure should be at zero. If not, solenoid valve 2 can be used to release any pressure. How fast the air pressure rises will be an indication of whether the solenoid is fully open. Once the system reaches 20 PSI, the user could try and hold that pressure

for some time. If solenoid valve 2 was to leak, air bubbles would be evident, as would a drop in pressure. Interestingly, the elastic Tygon tubing itself shows some leakage through its sidewalls at 20 PSI when subjected to a bubble test. Thus, we need to learn what is normal leakage for a system. Because normal leakage can be measured, we can take steps to reduce or remove it from the test system. Once pressure is held for a period of time, the user will release the pressure into the bubble reservoir using solenoid valve 2. Again, the release time will help determine if the solenoid valve fully opens. This process can be repeated over and over to test the basic function of the solenoid valve. In test mode B, the instrument might release the air in different pulse widths, latch the pressurized air line, wait for 100 ms, and then turn the latch off. By measuring how much the pressure drops and comparing this with other solenoid valves of the same type, we would learn what is normal for this type of solenoid valve operation. This can be repeated for several pulse widths, and the minimum pulse width can be determined. We can measure the pulse width required to get the solenoid valve to latch on and the pulse width required to get the solenoid valve to pulse off, thereby determining at what pulse width the solenoid valve fails. By setting the pulse width well above the failure point, we get a more reliable solenoid valve. The results of this test may vary depending on the pressure in the tank, so we could repeat the test at 10 PSI. Being able to try so many different variations becomes possible because we have an automated system. We have dedicated microcontrollers that can be wired to measure the differential voltage across a coil, providing the user with details on how the coil is functioning. All of these details become characteristics of the solenoid valve, and can be compared to solenoids of the same type. This type of methodology of component diagnos-

tics is what I refer to as smart components. This same type of software to identify problems with the solenoid valves can be implemented in systems using the solenoid valve. Again, the methodology measures many characteristics, determine what is normal, and looks for shifts outside normal. Through detailed failure analysis, we can determine what characteristic parameter patterns are associated with what failures. We hopefully will find that a weak spring, for example, will result in a certain type of shifts in characteristics; likewise for a weak magnet, an overly-strong magnet, or another type of shift. So ultimately, the software can direct the user to where the problem is. Fabricators can address these problems and get feedback on the effectiveness of their solution. This is the same methodology that we hope to use in early disease detection.

Chapter 11

Solenoid-valve-operated MEC

Pump

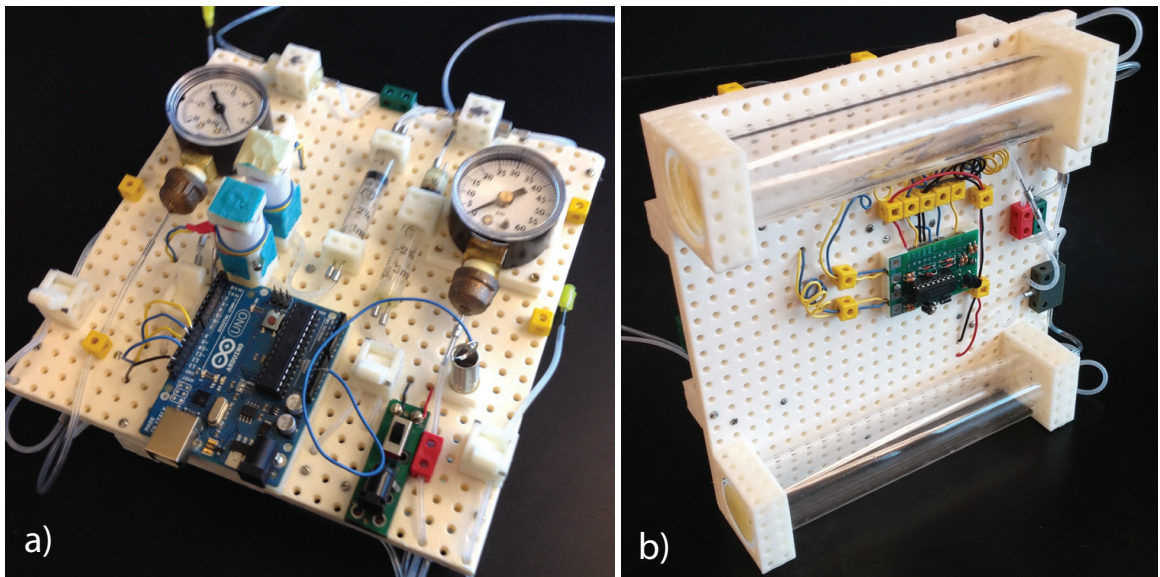


Figure 11.1: Frontside of electronically controlled pump (a). Backside of electronically controlled pump (b).

Shown in figure 11.1 is the original MEC pump system. Its function is based on the central pump found on the fluidic router. Both pumps used a modified syringe chamber with a plunger. In the case of the fluidic router, it used mechanical valves to apply pressure and vacuum to the pump piston. In contrast, the pump system shown in figure 11.1 uses electrically-controlled solenoid valves to apply pressure and vacuum to the pump piston. The pump on the fluidic router had the ability to transfer fluid between any of its macro storage units, in either direction. The pump shown in the figure 11.1 used one-way valves, so fluid could be transferred in only one direction, which is not a problem for many applications.

The schematic shown in figure 11.2 was drawn by an intern. It is the next generation to the pump system that was built and tested in figure 11.1. The schematic shows that the system has two electrically controlled pump systems. In contrast, the pump system built in figure 11.1 had one electrical and one mechanical pump.

11.1 MEC schematic defines the system and its assembly

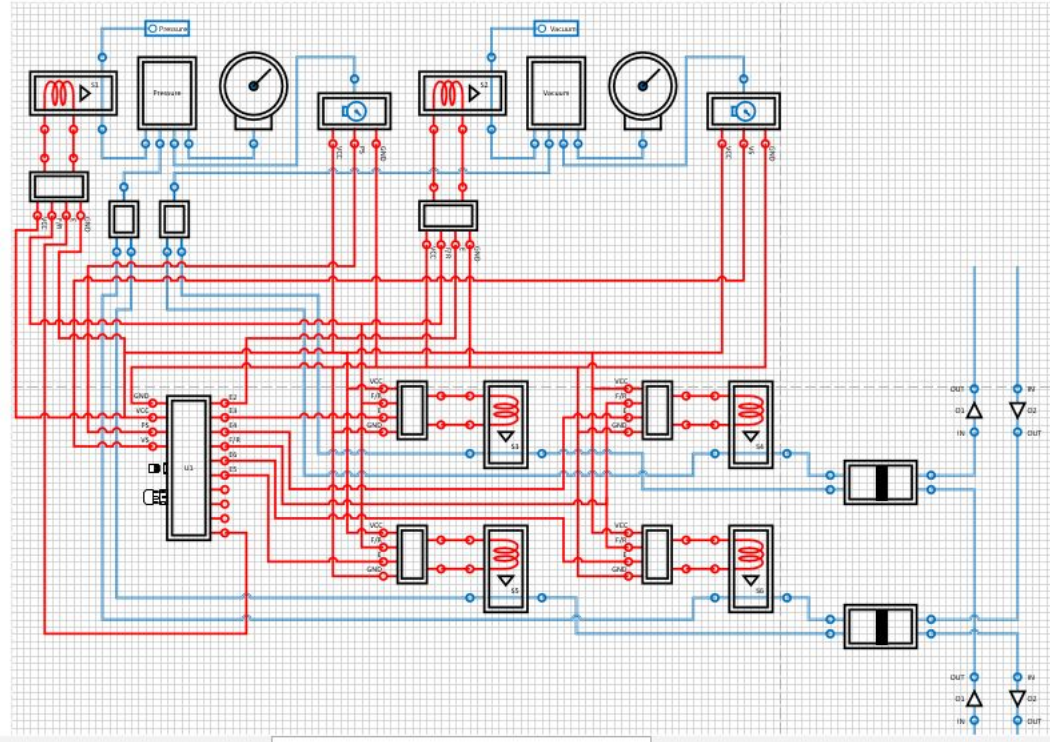


Figure 11.2: MEC schematic of an electrically-controlled dual-pump system. The pump is designed to be controlled by a laptop through an Arduino interface. It has two on-board chambers for pressure and vacuum. Each chamber has a manual gauge and electrical monitoring of pressure and vacuum, respectively. Both chambers have electrically-controlled external feeds, so that outside laboratory air pressure and vacuum supplies can be used for maintaining chambers. Pressure and vacuum have electrically controlled sources to the pump piston, drawing fluid in and out of the pump chamber. The flow direction is set by the one-way valves.

Another difference between the built system and the schematic is the built pump does not have electrical pressure sensors. Therefore, the microcontroller does not know the pressure in the chambers, which does effect the operation of the pump. The lower the pressure, the slower the piston moves. The MEC schematic (11.2) offers us the opportunity to discuss a design in detail, before it is fabricated, and because it is a multidisciplinary

schematic the user can fully examine the pump’s function. Looking at the schematic, even now, I can see the design changes that I would implement. The solenoid valve drivers are I2C controlled, but they are singleton drivers. The current I2C-controlled driver that we use is a dual driver. The signals from the pressure and vacuum sensors go to the Arduino. We now prefer to use separate AVR microcontrollers because of their structure and ease of integration with our current software system. Since this schematic was drawn, we developed a peristaltic pump. It could replace this system in many applications; this highlights the large overhead in components that this system requires.

11.2 Operation of MEC pump

Here we step through the basic operation of the pump system defined in figure 11.2. A laptop sends a command to the Arduino, which transfers it across the I2C bus to the target microcontroller and ultimately to the device specified in the command. The command could signal to turn on the H-bridge that controls the vacuum supply to the pump, putting the latching solenoid in an “on state.” Vacuum would then flow to the pump piston, pulling it back and drawing fluid in. Currently, there is no feedback to know when the piston is fully back, so a time delay is used. Then, a signal is sent to turn this solenoid off, and the solenoid that controls pressure would then be turned on. Again, a time delay is used which is set to allow the piston to travel all the way back, expelling a set volume of fluid out which is dependent on the chamber size. If too large a time delay is used, it is not a problem because the piston reaches a hard stop. The length of both of these time delays depend on the pressure and the vacuum levels. Lower levels will require

longer time delays. Optimally, a system would have some type of feedback. One method I am currently studying would involve weighing the source fluid reservoirs. With this type of feedback, precise partial pump actuation would be possible. Fluid would be drawn into the pump chamber and stopped when the desired amount of fluid was drawn. Then, that amount of fluid would be expelled. One could also place an IR emitter- and receiver-paired sensor across the pump chamber. This would allow the user to calibrate the time for the piston to move across the chamber as a function of the pressure and vacuum. Again, this time could be used to determine the volume of fluid that would be drawn into the pump with different delay times.

Chapter 12

Coulter System

The Coulter counter revolutionized medical diagnostics. It automated the characterization of blood samples by being able to differentiate and count the different blood cell types. It did this using a very simple principle. Electrical probes are placed on two sides of a very small aperture, and a small electric current is passed through the aperture. As a cell passes through the aperture, the electrical probes will measure an increase in the voltage. This is due to the increase in resistance of the aperture as the cell passes through. The passage of a cell is therefore recorded as a pulse in a plot of voltage vs. time. The area of this pulse is proportional to the volume or size of the cell. Different cell types have different sizes. Many times in academia we work to be on the cutting edge of research, but I believe there is much to be learned from what has already been done, from leveraging concepts from existing systems. Older instruments are often designed with more basic principles than their evolved counterparts, which make older instruments easier to duplicate. So, for one of the first MEC instruments I wanted to utilize the principles from the Coulter counter.

I thought this was fitting because it could be considered the start of medical diagnostic instrumentation.

12.1 Coulter Device

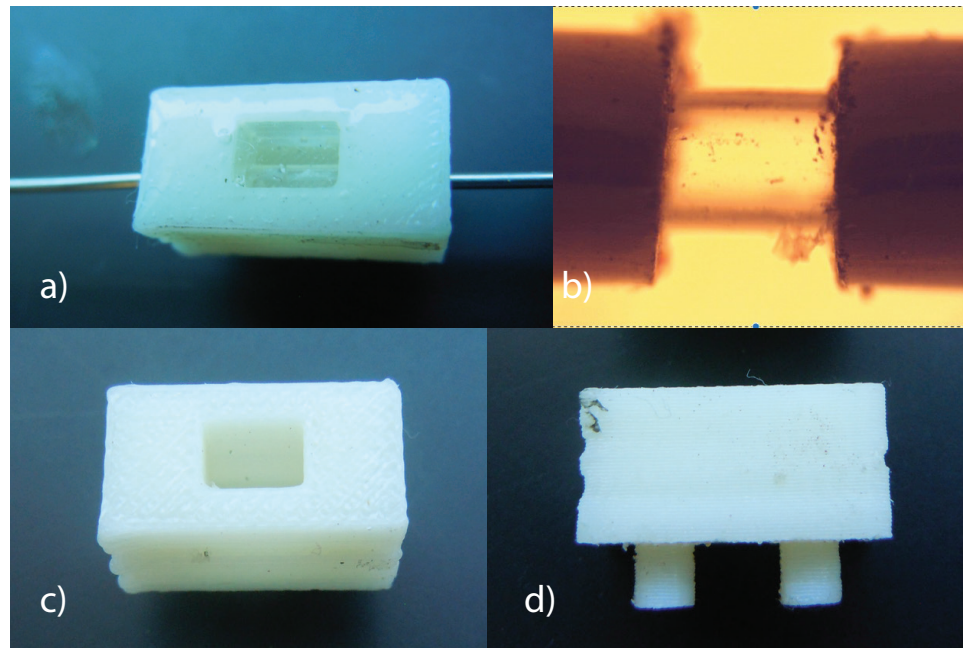


Figure 12.1: First Coulter sensor with 3D-printed base, PDMS-cast channels, and 20-gauge stainless steel IO pins (a). The sensing aperture is shown magnified in figure (b). The dark parts are 20 gauge stainless steel pins with a diameter of 0.036 in. The clear tube is the PDMS channel. Figure (c) and (d) show the top and side view of the 3D-printed body and cavity. 3D printed with the UC Riverside Bioengineering Department ABS printer.

We began the development of the MEC Coulter instrument with the development of the sensor. Madeline Mullen, an intern (at the time) in the program, was central to much of the work presented in this chapter. Figure 12.1 shows images of one of the first Coulter sensors we developed. The fabrication of the sensor is as follows: First, we 3D printed a small base, as shown in figure 12.1 c) (top view), and d) (side view). Notice the white

3D-printed pins coming out the bottom in figure d). These white pins allow the sensor to be inserted into a Snap baseboard. Next, at a trade show I found a range of very thin fibers. We used these fibers in the casting of the sensor. Once the user has the 3D-printed base, the user presses stainless steel pins into it, leaving a very small gap which can be seen in figure 12.1 b). The thin fiber was then passed through the stainless steel pins, and PDMS was cast into the small cavity. The PDMS completely encapsulates the thin fiber and the ends of the stainless steel pins. The thin fiber was then removed, leaving a cast channel in the PDMS. We tried two different sizes of stainless steel pins. First, we used 20-gauge stainless steel pins and permanently epoxied them into place. Second, we used 22-gauge pins. They were removed after casting and then we epoxied the 20-gauge pins in place. The smaller 22-gauge pin provides a press fit when the 20 gauge pin is inserted. The second method offered more flexibility for the user, in case one encountered problems in the fabrication.

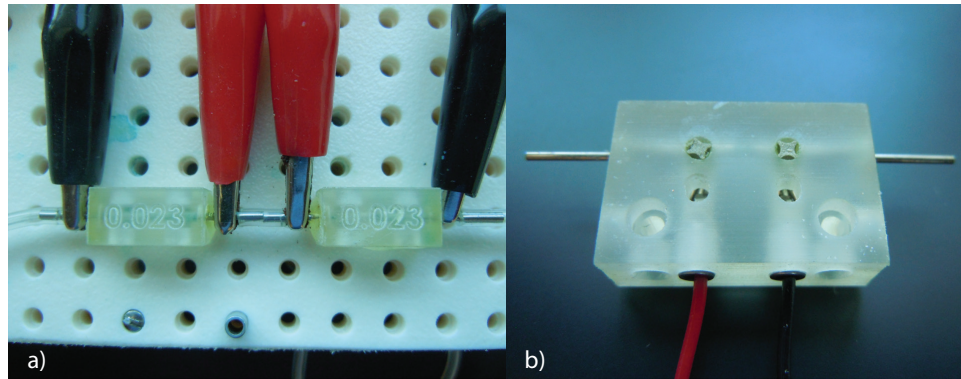


Figure 12.2: As a small object passes through the aperture, the Coulter sensor is able to measure the size of the object by measuring the change in voltage across the stainless steel pins.

To test the sensor we wanted to pass fluid with microbeads through the Coulter sensor and measure the voltage across the stainless steel pins. The sensor is integrated

into a simple resistor divider network so that the voltage across the pins is dependent on the resistance across them. We used alligator clips as shown in figure 12.2 to make the electrical connection to the stainless steel pins. Figure a) shows two 3D-printed sensors in series. Having two sensors in series gives us better noise removal and makes signal averaging possible. We wired the sensors so that current flows in opposite directions in the two sensors. This will result in a positive and negative pulse measured when an object flows through the sensors. The signal can drift toward the positive or negative threshold over time. Having both pulses will aid in reducing the problems associated with this drift.

Shown in figure 12.2 b) is the next-generation Coulter aperture sensor. Here, the alligator clips have been replaced by small sockets. The sockets are integrated into the 3D-printed sensor. Conductive rubber is used for connecting the sockets to the stainless steel pins. Screw holes have been added so that the sensor can be securely mounted down onto a MEC baseboard. *Evolution: 3D printing the complete Coulter sensor, removing the need for PDMS casting. The use of embedded sockets removed need for alligator clips*

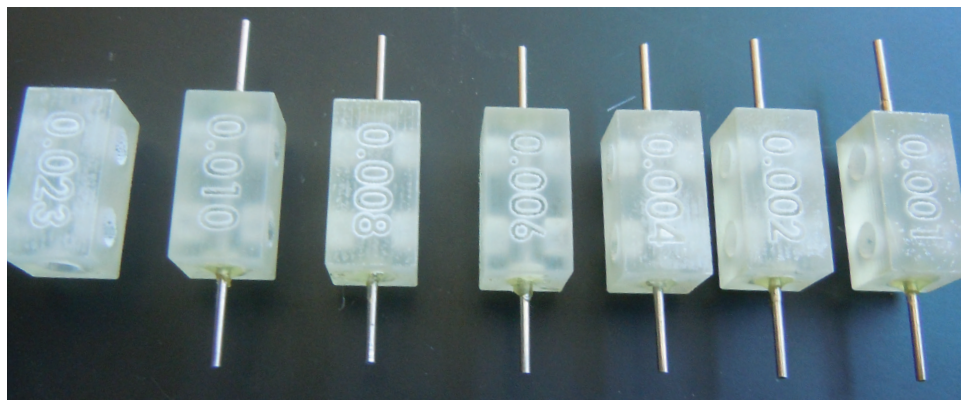


Figure 12.3: 3D printed Coulter sensors with different aperture sizes. The aperture sizes are: 1/1000 in, 2/1000 in, 4/1000 in, 6/1000 in, 8/1000 in, 10/1000 in, and 23/1000 in. We were able to get useful apertures with sizes greater than four thousands of an inch.

The sensors shown in figure 12.3 were 3D printed with a Stratasys printer. The Stratasys 3D printer uses a wax support system. Part of the development of these apertures was to determine what hole sizes were possible with the Stratasys 3D printer. It is our hope in the future to use 3D printing to make fluidic channels in more complicated fluidic systems. These Coulter sensors are a first step in understanding the limitations of 3D printing. The 1/1000 inch and 2/1000 inch apertures were clogged, and we were unable to flow fluid through them. In our first tests we were able to get flow through 6/1000 inch to 23/1000 inch apertures. The results are shown in figure 12.3. These first results were obtained by Madeline Mullen. The 4/1000 inch aperture was originally clogged with wax. Stainless steel pins were epoxied onto the 4/1000 inch aperture. We then used a syringe and tubing to create a vacuum connected to the sensor. Next, we placed the sensor under a vacuum in the oven. Once the wax heated up, the setup automatically sucked out the wax. We were then able to get pressurized flow through the 4/1000 inch at 5 PSI, and a flow rate of 0.056 mL/min. These are very simple flow channels. This shows that as more complex flow channels are developed, we will be fighting a wax clogging problem with parts from the Stratasys 3D printer.

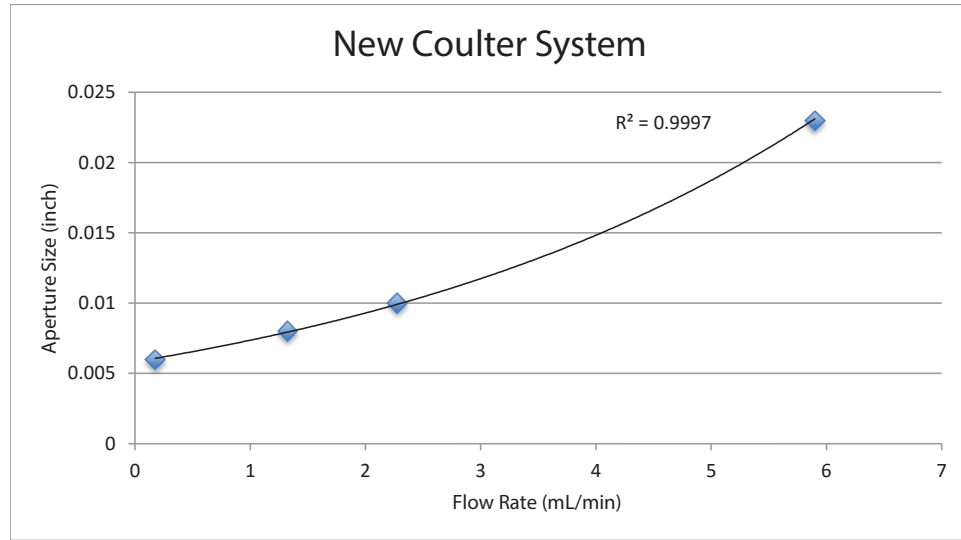


Figure 12.4: Measured flow rates confirm changes in different aperture sizes. These tests were done on the MEC scaffolding system. We used the apertures shown in 12.3 and recorded the amount of time while fluid passed through aperture, then weighed the fluid that passed through the aperture.

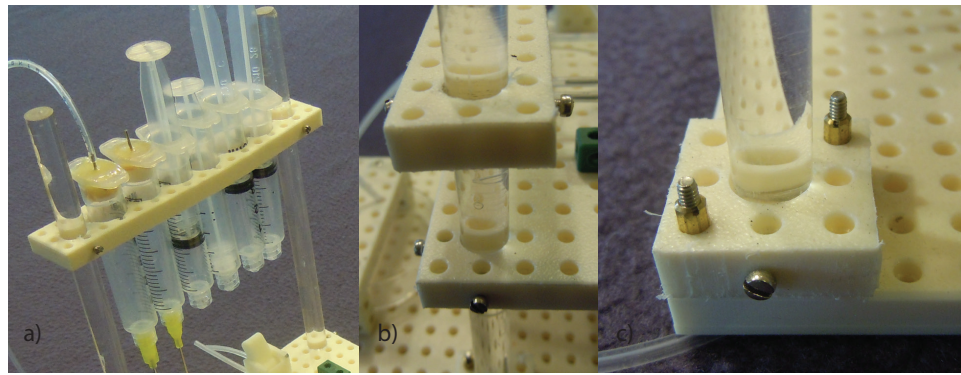


Figure 12.5: The MEC scaffolding system required some new components to be developed. a) shows the 10 milliliter syringe holders. Some of the syringes have been adapted to allow pressurized air to be applied at the top. The screws shown in b) and c) are used to lock the scaffolding into place. b) shows how platforms can be locked in place with the scaffolding system. d) shows the base support for a scaffolding rod.

To provide gravity-driven and pressurized fluid feed to the sensors, we decided to modify a scaffolding platform that was originally developed to test dual mechanical valves.

Shown in figure 12.5 are some of the parts that were added to the MEC system as the

scaffolding system was developed. These parts were all machined from Snap baseboard material with a drill press and jig saw. 3D-printed parts could also be developed.

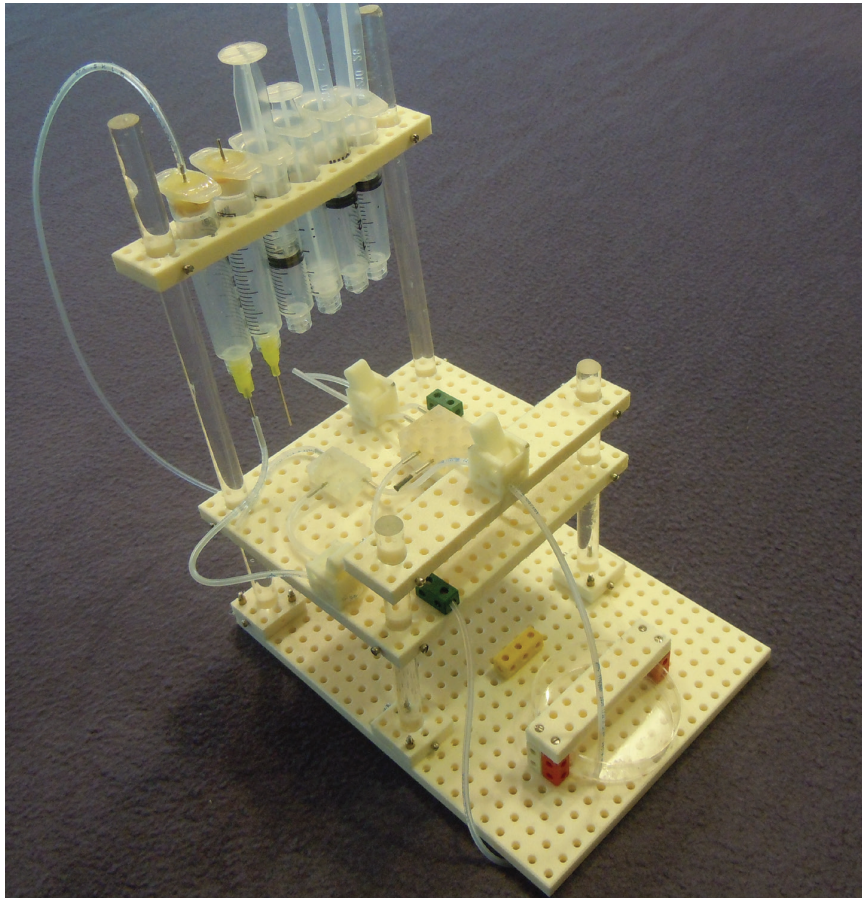


Figure 12.6: MEC platform used for testing Coulter sensors

Shown in figure 12.6 is the scaffolding platform that was adapted for the gravity and pressure testing of the Coulter aperture sensors. A mechanical valve can be used to shut off the flow line used for testing an aperture or Coulter sensor. By timing the flow and weighing the fluid in the petri dish, flow rates can be determined. The scaffolding system allows 3D structures to be rapidly built. Tubing resistance did play a role in the aperture

flow rates, and this platform would be well suited for testing tubing resistance in the future.

Evolution: Development of MEC scaffolding system. The scaffolding system further evolved with the marble machine, which used 1/4 inch diameter rod.

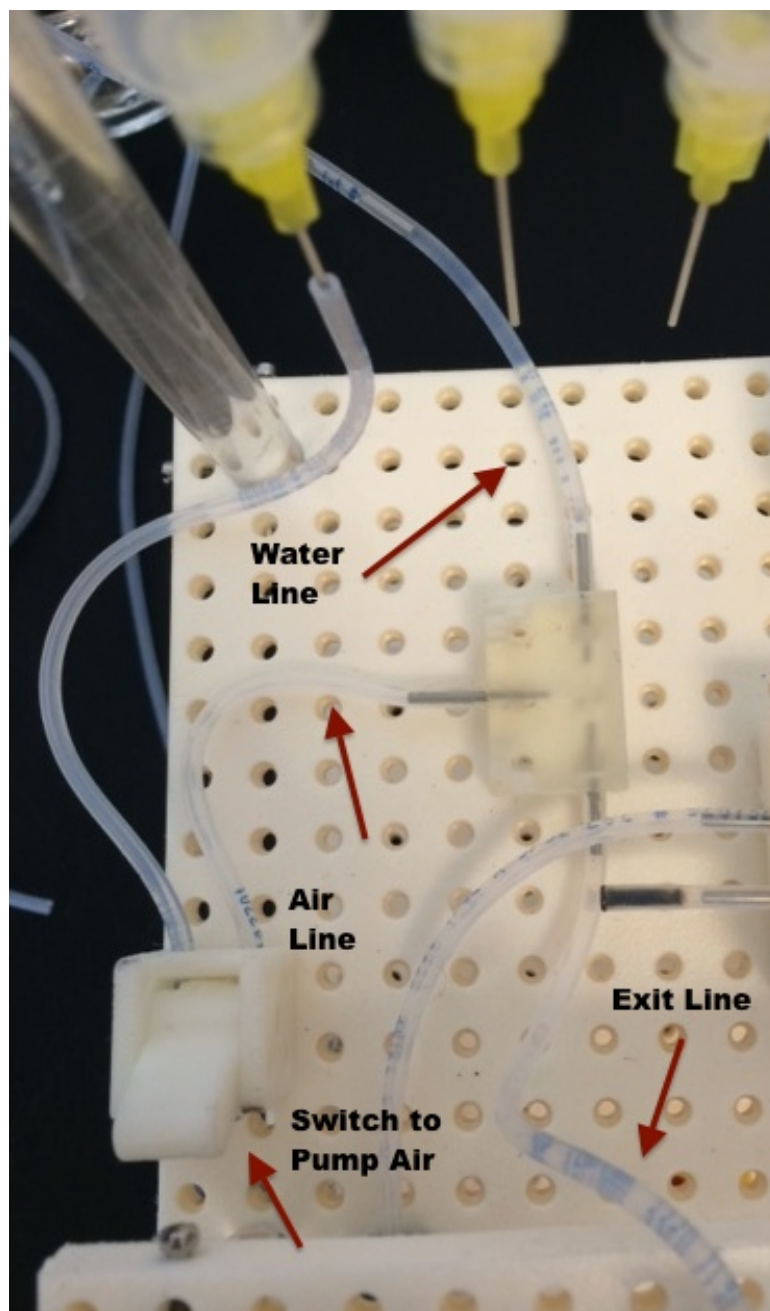


Figure 12.7: Setup for passing bubbles through the Coulter sensor, which is used to test its operation. Pressure is applied to water in an external bottle, driving the fluid first through a T joint and then through the Coulter sensor (not shown). Air pressure is then applied to a dual switch. When the switch is pushed forward, air passes to a small loop in front of the switch. When the switch is pulled back, the air feed line is turned off and a small pulse of air is sent to the T joint. This will then pass through the Coulter sensor, resulting in an electrical resistance change while the pulse is passing through the sensor.

After flow testing of the aperture sensors, Madeline Mullen prepared the platform to test voltage across the pins using alligator clips as described earlier. She utilized a National Instruments data acquisition card and LabVIEW for recording the test results.

Figure 12.7 shows the configuration of the platform for the air pulse test of the aperture sensor. In this test, one can use pressure or gravity to drive flow through the water line, T joint, and aperture sensor. A simple resistor divider network is utilized, and the voltage is measured with the National Instruments data acquisition card across the stainless steel pins on the aperture sensor. Pressurized air is run to the air pump (dual valve). As the valve is pushed forward to the load position, pressurized air goes into the small air line loop on the pump. As the valve is turned back, air from the loop is released to the line running to the T joint. A pressurized air pulse then travels to the T joint and creates a small bubble. The bubble size is dependent on the size of the small air pump loop and the pressure in the line feeding it. The bubble travels down the fluid line and through the aperture sensor. The resistance in the aperture sensor changes over time as the bubble passes through it. The size of the bubble theoretically could be determined by the area of the voltage pulse.

A number of observations were made during this early test. First, one cannot use pure water for this test due to its very high resistance. Instead, different concentrations of salt water were used. We kept the concentrations around that which is found in blood. For very small apertures the flow resistance could be so high that it would stop the fluid flow. Here, pressurized flow was preferable. As a drop formed on the end of the tube the resistance to flow would increase due to surface tension, so having the end of the tubing

immersed in fluid reduced this resistance. Diameters of objects measured with this sensor must be smaller than the sensor diameter, but should be the same order of magnitude. We were able to get positive results when testing air bubbles passing through the sensors of different aperture sizes in the scaffolding system. This is due to the relatively large size of the air bubbles compared to the aperture hole size. When passing beads through the same aperture sensor, the results were much less conclusive. There were several problems: First, due to the high resistance through the aperture, the system was noisy. Second, the beads produced a signal that was not significantly larger than the noise. Third, when we began, we were only utilizing one sensor, so there was no way to confirm something as signal versus noise. Finally, we were using continuous flow, so the resulting data set is large and has no set way to determine where to look for the signal.

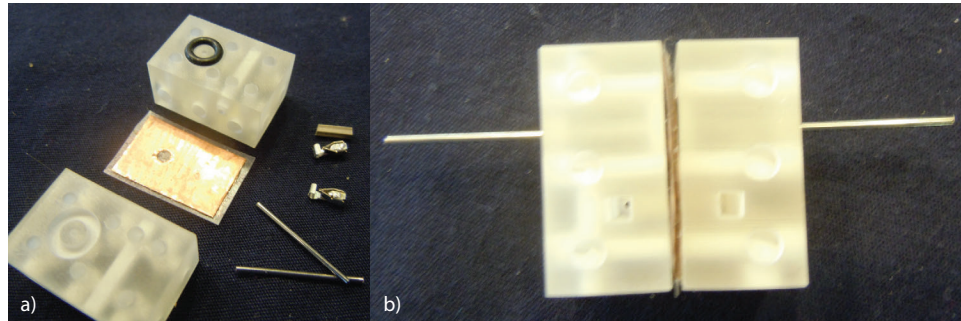


Figure 12.8: Length of aperture path and diameter of aperture are critical to the sensitivity of the Coulter sensor. Components of the next MEC generation of Coulter sensors include: 3D printed base, two O-rings, clear film with small aperture hole, copper foil on both sides of film, two stainless steel pins, conductive rubber, and two small wire connectors (a). Assembled Coulter sensor (b). This sensor was never tested.

To receive the best readings from a sensor, one wants the aperture to be only slightly larger than the object. Here, the user might have a sequence of different-size apertures going from larger to smaller in the sensing system. Also, the change in resistance

is also dependent on the path length in the aperture. For maximum readings, one wants a path length that is short. Shown in figure 12.8 are the parts for a Coulter sensor which take these limitations under consideration. Here, the path length is determined by the thickness of the film and adhesives between the two copper foils. Aperture size is determined by the size of a hole in the foil. O-rings are used to create a seal between the foil and the 3D-printed sensor housing. 20-gauge pins were epoxied into place in the housing. Again, small sockets were utilized for making electrical connections to the sensor. Conductive rubber was used to make the connection between the socket and the foil. This sensor was not tested.

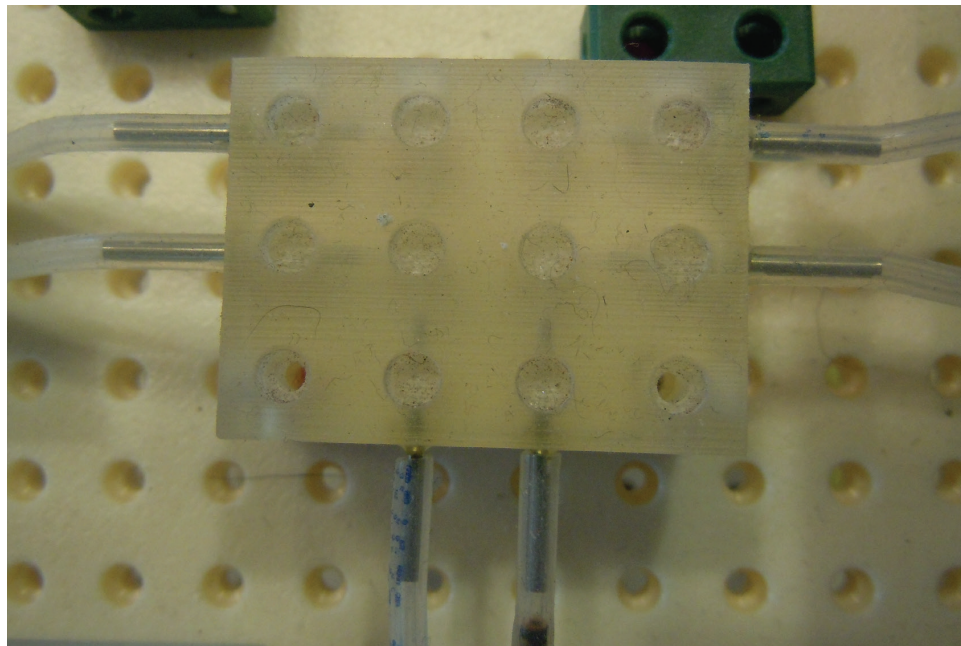


Figure 12.9: Unit fluid pulse component. Two sets of lines are used for loading the pulse. The other two lines are used to send the unit pulse down the line through the sensor with air.

One modification we investigated involved putting a pulse of fluid into a flow of air instead of a pulse of air into a flow of liquid. The fluid sample would contain the cells or beads of interest. In this way, we would know where in the data to look for the signal. We

designed a component for this purpose, shown in figure 12.9. This component uses the same principle as that serpentine storage system in the router system described earlier, except that this component operates at the macro-scale instead of the micro-scale. It uses two sets of pins. The first set of pins are for loading the storage unit. The overflow fluid goes out the second pin. The volume is determined by the loop and the volume in the device. In this way, different volumes can be achieved by changing the loop length. Once loaded, air is slowly run through the main line, pushing the stored sample of set volume out through the sensor. This design works best if lines are shut off when not in use and are as close to the device as possible. In this way, fluid is less likely to flow down dead-end lines. Even when a line is shut off, some fluid could flow since air is compressible. *Evolution: Sample pulse of fluid is used in air line, so that the critical area of recording can be determined.* Another untried method would be to use pure water in the non sample region (replacing air). *Future Evolution: Sample pulse of fluid is used in the pure water line, so that the critical area of recording can be determined.* This may have the additional advantage that objects such as beads tend to stick to the leading or exiting surface of the fluid pulse. This is because of the surface tension of the fluid at the air-to-fluid interface.

12.2 Coulter 1 system

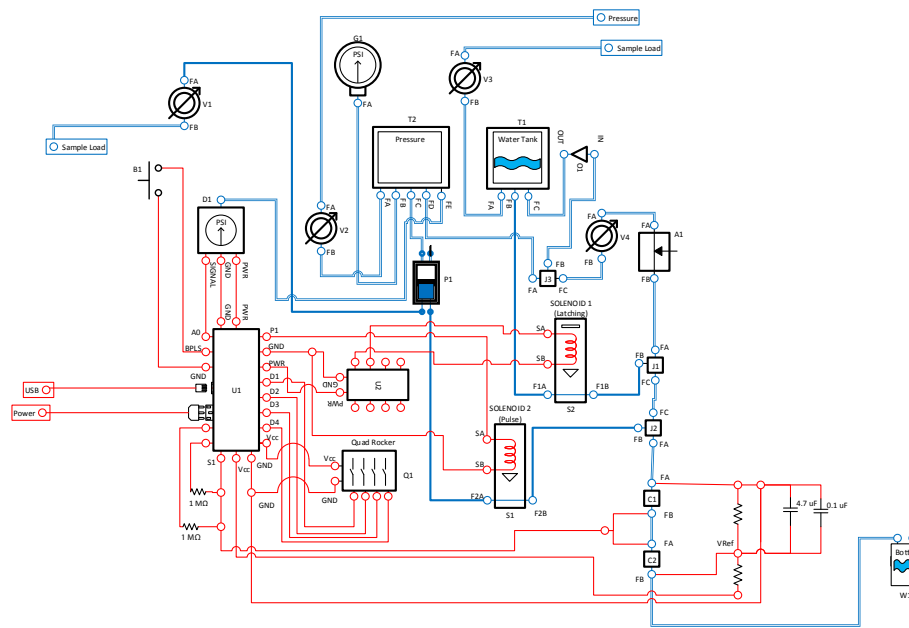


Figure 12.10: MEC Coulter 1 schematic. Blue lines represent fluid lines. Red lines represent electrical connections. C1 and C2 are the Coulter sensors. The signal line is connected between the two Coulter sensors to an analog-to-digital pin on the Arduino.

After finishing our preliminary tests of the Coulter sensor with the scaffolding platform, we utilized that knowledge to design our first Coulter instrument. Shown in Figure 12.10 is the schematic that was used in the development of the instrument. Since we only wanted a pulse of fluid, we utilized a pulsing solenoid instead of a latching solenoid. This instrument highlighted the need for a pulsing solenoid, for applications where the user wants only a short pulse of fluid to travel through the two Coulter sensors. This also shows a strength with our methodology of designing, not purchasing, components. Utilizing in-house design, one can modify devices to more closely fit one's needs. Along with the hardware, I

had another intern, Adam Veres, develop the Python and Arduino software that is required for this instrument.

The operation of this instrument is as follows: begin with a command-line menu on a computer. Next, identify the instrument one wishes to operate. The software shows a menu where the user can select the pulse length desired. The fluid volume of the pulse is dependent on the pulse length. It is also dependent on other factors, such as the pressure that is driving the pulse. The schematic shows both a mechanical and an electrical pressure sensor. This allows the software to adjust the pulse length, providing a more precise volume of fluid. Another factor in determining the pulse volume is the air flow rate. This flow rate is determined by a number of factors. Component A1 in figure 12.10 is a mechanical flow control. Here, a 2-56 screw is utilized to press down on the elastic tubing. By adjusting the screw one can adjust the flow rate. The flow rate is also dependent on the pressure. The pressure is known, but it is not constant. The multidisciplinary schematic provides us the ability to discuss the design. There is the usual tradeoff: better functionality with an increase in complexity. If the manual valve V1 was replaced with a solenoid valve and the external pressure line ran to another tank or a lab pressure source, we would be able to have a controllable constant pressure running to our system. This is better in many ways, but requiring an external pressure source does add to the system's complexity. Our plan was to build a simple system first, and then add features as time allowed. It is important to remember that we were just creating our solenoid valves as this system was being developed. The latching solenoid valve S2, shown in the schematic, which was for a flushing operation, had problems and was not implemented in the original instrument.

The flushing operation is another example of why preliminary testing is so critical in facilitating instrument evolution. The Coulter sensors have a high resistance, above 500 k Ω . However, as salt solution pulses flow through the sensors, a residue develops on the sensors. This residue can shift the central signal voltage level. Say that both sensors have 500 k Ω resistance, and V_{ref} is set at 2 V. Then the central signal voltage level is 1 V. As a salt solution passes through the first sensor it pulses the central signal voltage level toward ground. Then, as it passes through the next sensor it pulls the signal toward V_{ref} . After the pulse leaves the voltage should go back to 1 V. Instead, we see some drift. If we assume the resistance in the first sensor drops to 250 k Ω due to the residue, then the central signal voltage level will drop to 0.67 V. In this manner, the signal is clipped. Instead of getting clear positive and negative pulses as a pulse of fluid passes, one voltage hits a threshold and is flattened. To overcome this problem, we added in two features. First, a flushing operation was added using deionized water. Second, we utilized two 1 M Ω resistors tied to the central signal voltage level. The other side of these resistors go to an Arduino digital output, which can be high (close to 5 V), low (close to ground), or left floating. These resistors allowed us to pull the central signal voltage level toward either threshold.

What I'm hoping is that the reader will see that students are being exposed to the design process from start to finish, and all the messiness that involves. School often provides highly complex paper problems, and students learn how to work through them. Instrument development can use those skills, but it needs something more. It requires practical, hands on problem solving skills, and those are best learned through experience. Who would have guessed we would have a voltage drift problem? Our one clue might have been that we

were dealing with very high impedance, which experience has taught me makes the system susceptible to noise.

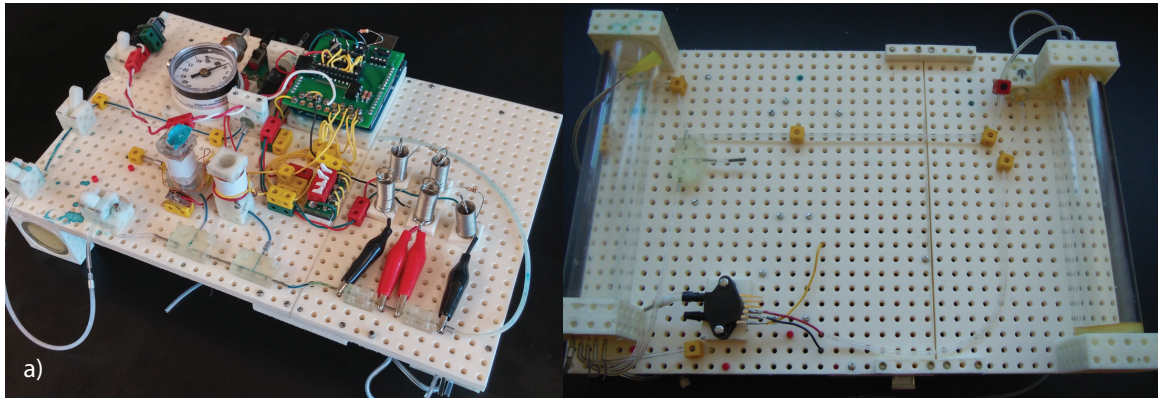


Figure 12.11: Frontside and backside of Coulter system. Alligator clips are used to make electrical connections to the stainless steel pins on the Coulter sensor. A Snap quad-rocker switch is used to send a four-bit digital signal to the daughter card to set the pulse width time of the sample. The pulse width time is used to determine how long the solenoid valve is open. On the backside one can see two air tanks and an electrical pressure sensor.

Shown in figure 12.11 is the original instrument we developed. One aspect that is exciting is to see the number of repeated MEC components that have also been used in earlier systems. These components include the mechanical valve, air chamber, pressure gauge, Arduino, T joints, fluid storage, and MEC baseboard. This demonstrates the flexibility of the MEC system. The original baseboard was not large enough. In the bottom view one can see some two 1x1x7 Snap baseboard blocks which were used to attach to a second baseboard. On the bottom one can also see yellow Snap mechanical blocks being used to help run fluid lines. Also visible is an air chamber, which is different from what is shown on the schematic. Here, we were setting up a sacrificial tank which was designed to feed the other tank so we could maintain a constant pressure. On the top view one sees electrical spring terminals. These are from the original Snap system and are a simple way of making

electrical connections. One can also see a quad-rocker switch. This switch could be used to directly change the pulse width without going back to the laptop software.

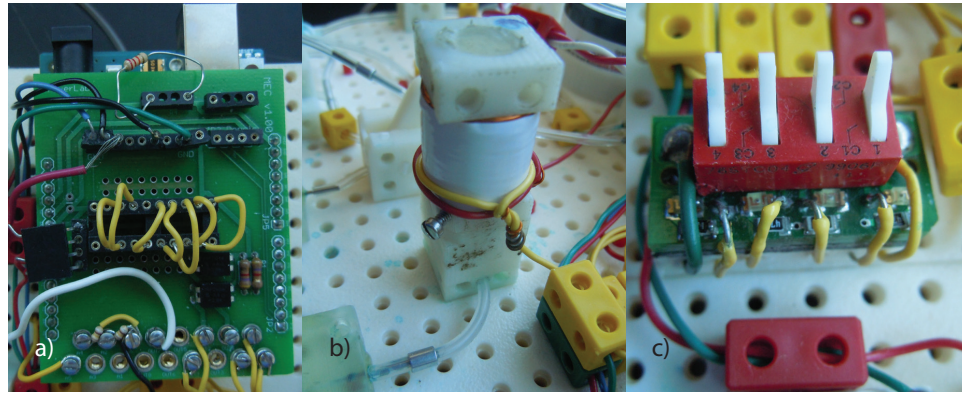


Figure 12.12: Some of the key components in the Coulter 1 system include: daughter card (a), MEC solenoid made with a Dimension 3D printer (b), and Snap quad rocker electrical switch.

Shown in figure 12.12 are some of the electrical components of the system. In contrast to our fluidic router (which was focused on developing a basic fluidic component set), the Coulter 1 system combines fluidic and electrical components. Figure 12.12 a) shows an early daughter card which fits over the Arduino. This daughter PC was drawn up by Kevin Pham, an intern, in Eagle software. The concept was to put most of the needed electrical components of the system on this daughter card. Two of the key components are the FETs, which can be used for driving the pulsing solenoids. We press-fit terminal nuts into the PCB, shown at bottom of figure 12.12 a). This allowed us to screw down the wires. This made connection and changing of wires both easy and reliable, but terminal nuts seemed relatively expensive and time-consuming to implement. Shown in figure 12.12 b) is one of our first working solenoid valves. This pulsing solenoid valve worked over many repeated operations and proved the functionality of the solenoid valve design. This solenoid

valve was made prior to our current socketing methodology. We wrapped 22-gauge yellow and red wire around the solenoid, twisting a small, exposed section to hold the wire in place. We then took the thin magnet wire that was used for making the coil and soldered it to the exposed section. As with the compressor, we wanted to be able to turn on the Coulter 1 system and have it be able to function at a basic level without computer control. Here, we wanted to be able to change the pulse width of fluid that travels through the sensors, without going through the laptop. A quad-rocker switch worked well here. It can provide a four bit binary number to the Arduino, allowing the user 16 different pulse widths. The MEC system did not have a quad-rocker switch, so we modified a Snap circuit quad-rocker, as shown in figure 12.12 c)

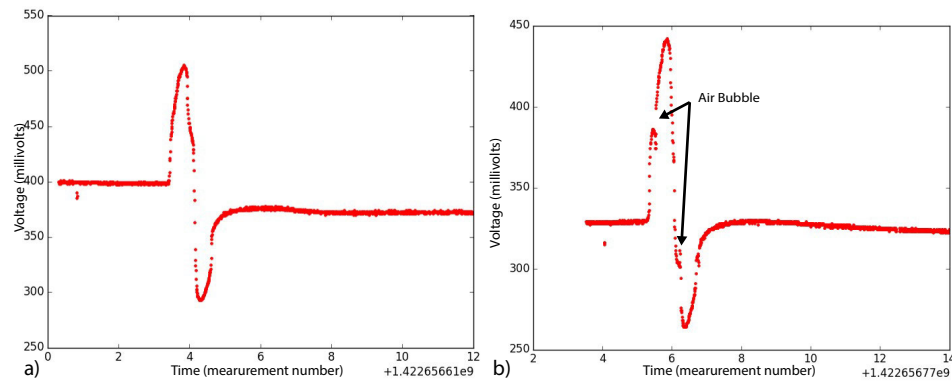


Figure 12.13: Signal read by the Arduino without (a) and with (b) bubbles in the sample pulse of fluid.

Shown in figure 12.13 are some of the data logging results obtained by the instrument shown in figure 12.11. Figure 12.13 a) shows the signal without an air bubble, and figure b) shows it with an air bubble. We can tell the air bubble is not noise because it shows up in both the positive and negative pulses, as seen in figure 12.13 b). Another observation with figure 12.13 a) is the drift of the central voltage, from before and after the

pulse travels through the sensor. We believe that this is due to the buildup of residue in the sensor.

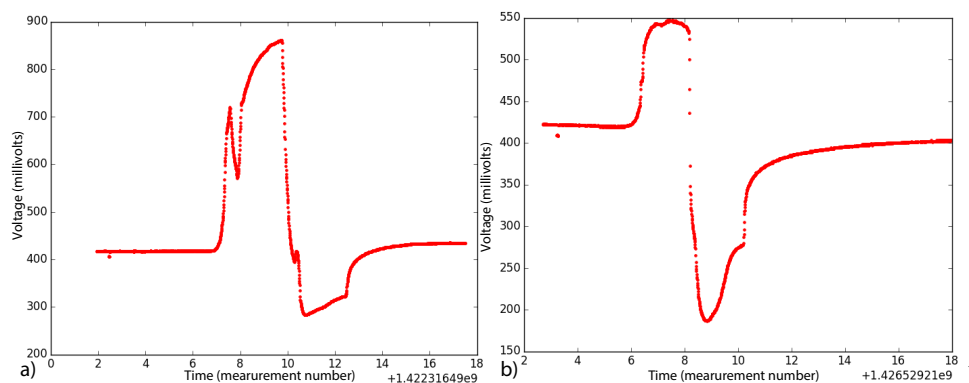


Figure 12.14: Early electrical measurements of fluid samples with NaCl in solution (a and b).

Shown in figure 12.14 are a couple of earlier results obtained as the fluid pulse moved across the Coulter sensor. These are included to show the range of pulse shapes we recorded. We obtained pulses with flatter tops with some of our earlier testing, which was consistent with expectations. If one imagines a small volume of homogeneous salt solution crossing a small conductive gap over time, whether the gap is at the beginning of the pulse or the end, the resistance should remain the same. The fact that we did not observe pulses with flat tops indicates to me that the resistance is changing during the period of time the pulse is moving across the gap. I would suspect a surface ion residue is building up in the sensor, but more testing would be required to determine the reason for the observed pulse shape. From an object detection standpoint, as long as we can get consistent wave shapes over a short period of time, which we were getting, we can normalize to that consistent wave shape and identify any changes. One can think of it as changing our consistent pulse shape to a flat line so that the only thing one sees are changes from baseline. Thus, a

small object in fluid would show up as a change. I was working with another intern, Devin Finkel, on a Fourier transform coefficient matrix to model the consistent wave shape. Once completed, it would be used to identify the change in wave shape. This would allow us to identify objects as they pass through an aperture.

12.3 Coulter 2 system

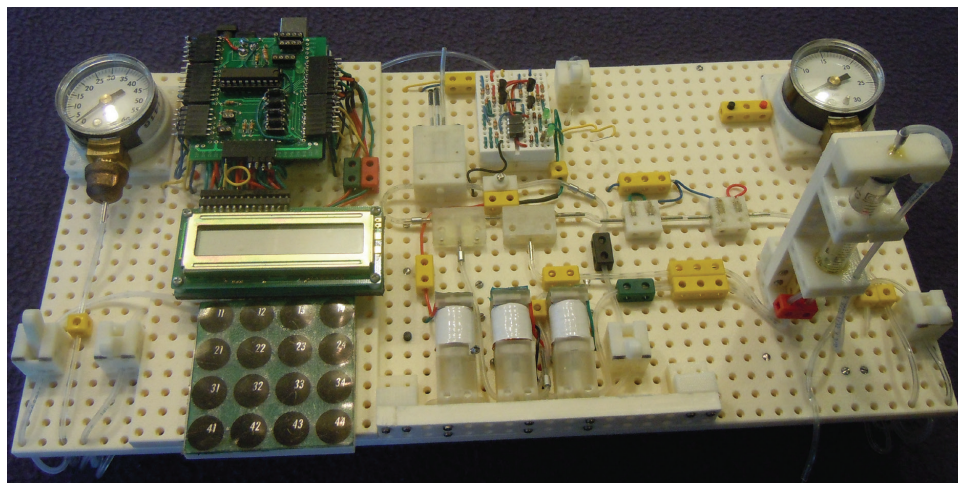


Figure 12.15: The MEC-based Coulter 2 Instrument

Shown in figure 12.15 is the next evolution of the Coulter 1 system. Each generation we took what we learned from the prior design and added it to the next. Some improvements include: an on board mini-computer with keypad and display, vertical sample dispenser, electronic flow control, socket Coulter sensor, and regulated pressure. We can also see improvements in our daughter card design, solenoids, power plug, and T joints. To keep the front of the design looking clean, most of the electrical routing is on the back. This design used a parallel electrical routing methodology, which I will discuss later. Some

of the reused components include baseboard, yellow routing blocks, mechanical valve, and mechanical pressure gauges.

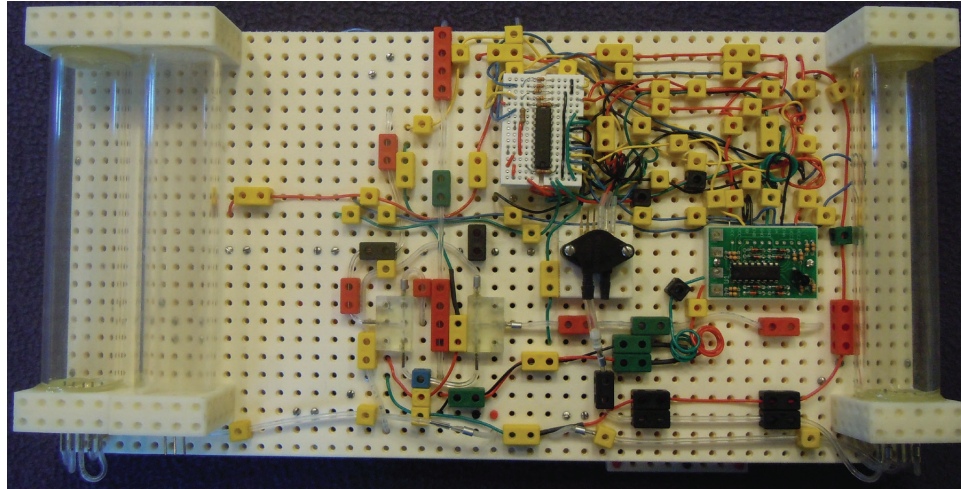


Figure 12.16: Bottom view of Coulter 2

Figure 12.16 is the bottom view of Coulter 2. The reader has seen most of these components before. They include air chambers, electrical pressure sensor, male T joints, multi-color routing blocks (both fluidic and electrical), and H-bridge. The system uses an electrical breadboard MEC block. This time the breadboard is used to assemble a electrical latch. Clearly visible is the complexity of the electrical routing, which is a result of our use of latches in a parallel routing methodology.

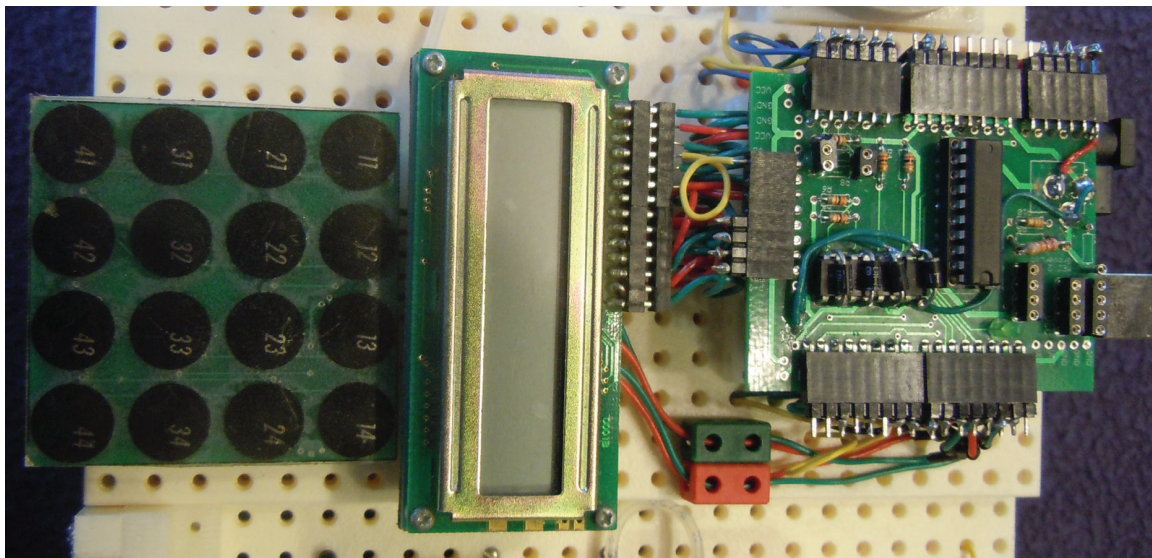


Figure 12.17: Closeup of the on-board mini-computer, with daughter card plugged in over the Arduino.

Shown in figure 12.17 is what I like to call a mini-computer. The three primary components are an Arduino, keypad, and display. This is a natural extension of our philosophy of wanting to be able to perform basic functions with our instrument, even when not plugged into a computer. With the keypad and display, we can develop menu operations for the operation of the instrument. This can allow the user to set up and control all the parameters of the instrument. The keypad was made with a custom PCB, dome buttons, and a transparency sheet. The keypad transparency can be custom labeled with whatever button names are required. The keypad works with 4 rows and 4 columns of buttons. When a button is pushed it connects the selected row to the selected column. Software scans the columns as it sequences through setting each row high. The keypad requires eight I/O lines and the display requires nine. We utilized the display in nibble mode, so the data was multiplexed on four lines instead of eight. Each solenoid valve requires two lines. To

accommodate these large I/O requirements, we placed a latch on the daughter card, and a second latch on the main board. A latch allows one to take eight I/O lines and latch the data by pulsing a latch pin. Once the data is latched, the I/O lines are free to change and perform other functions, or prepare to send data to the next latch.

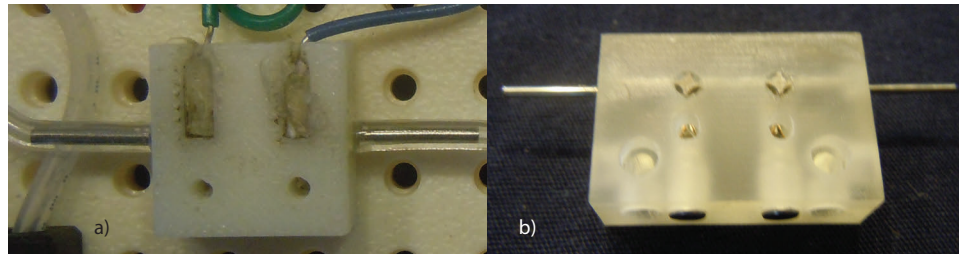


Figure 12.18: a) a 2x2 MEC Coulter sensor with half-grid fluidic connection. b) a 3x2 MEC Coulter sensor with on-grid fluidic connection.

Shown in figure 12.18 a) is the Coulter sensor. This was the first MEC Coulter sensor to use small wire sockets. The sockets were epoxied into place. The MEC size is a 2x2x2. The wires are plugged in at the top. In figure 12.18 b) one can see some small socket access holes. These are used for removing wires from the socket. Because the device shown in figure 12.18 a) is missing these holes, it is difficult to remove wires without damaging the socket. In figure 12.18 b) the sockets are pressed up against the stainless steel pins with conductive rubber. Aluminum rings are pressed into place to hold the sockets. Wires can then be fed through the rings. On the bottom of figure 12.18 b) we have male pins and 2-56 screw holes on grid. This allows it to be easily mounted on a Snap baseboard.

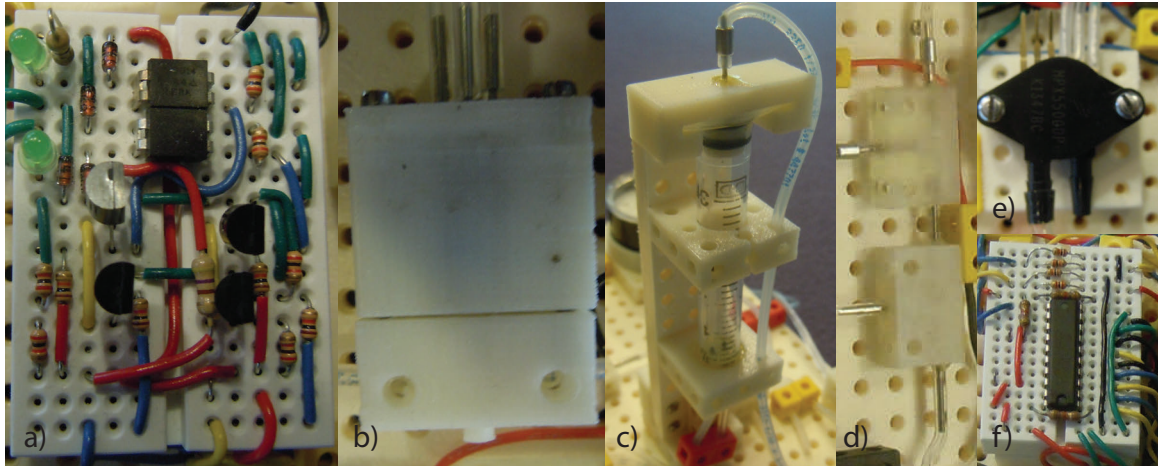


Figure 12.19: a) Breadboard of a dual FET pulse solenoid driver. b) Press piston for flow regulator. c) Vertical bead storage unit. d) Female T joints. e) Electrical pressure sensor. f) Electrical latch, for latch digital signals.

Shown in figure 12.19 are components from the Coulter 2 System. Two of the solenoids in the Coulter 2 system were pulsing solenoids, so they only required a FET driver as opposed to an H-bridge. This FET driver is shown in figure 12.19 a). When the FET drive is on, the LED turns on to let the user know that the system is working. A transistor is used to drive the FET input gate to a higher voltage, which, in turn, lowers the “on” resistance of the FET. Higher coil currents were required in some solenoid, spring, magnet, and rod combinations. Two transistors were used to keep the same software logic as the design that used an Arduino output to directly drive the FET input. The flow regulator for the Coulter 2 system is shown in figure 12.19 b). In the Coulter 1 system we utilized a screw for adjusting the flow rate. Here, we wanted to have an electrically-controlled flow rate. We modified the solenoid design so that the piston pushes down, pinching the tubing. Instead of having the piston driven with a coil, we have a pressure-driven piston. Adjusting the pressure adjusts the force pressing on the tubing. One of our concerns with this design is

the static frictional resistance of the piston rings. The applied pressure has to also overcome this frictional resistance, which means there is not a one-to-one correlation between pressure and flow rate.

Usually, when there is something I do not like about a design, the design will change. Shown in figure 12.19 c) is a vertical bead storage unit. If the reader remembers from Coulter 1, the storage unit was horizontal. Pressure was applied to the back side of the syringe piston, and it would drive fluid out. With this type of system, beads could settle to the bottom, away from the fluid outlet. With the Coulter 1 system we would manually inject beads into the fluidic line to avoid the settling problem in the storage unit. The vertical bead dispenser in Coulter 2 has two fluid connections on the bottom and one on the top. The top fluidic connection passes through a mechanical valve, and is typically closed. The top of the bead dispenser with the fluidic connection can be pressed on and then twisted to lock into place as shown. O-rings in the top piece form an air tight seal. I conceived the idea for this part of the design from a solder dispenser I owned in my company. To release fluid and beads out, a pulse of air is applied into the lower connection of the bead dispenser. The pulse first helps mix the beads, then the increase in pressure drives the fluid out of the second connection. For ease of transport, the bead dispenser was built on a hinge so the whole system can be laid down flat. This is the first demonstration of the use of a hinge. As we work to make a large tool kit for engineers to work from, we often design components like a hinge, which have general utility, before a specific application has been determined. Shown in figure 12.19 d) are female T joints. These look very similar to male T joints. Here, instead of epoxying male pins in place, Tygon tubing is inserted into holes,

creating a female socket. Stainless steel pins can then be inserted into the Tygon tubing. If we implement this in all three holes, the result is functionally the same as the male T joint. There is only one Snap space between the T joints shown in figure 12.19 d). This would not be have been possible with the two male T joints, which typically require three Snap spaces between them. With the female T joints we have one Snap space between joints (and could have zero). This is possible because one stainless steel pin can be used to connect between the two. In some respects, this is the same kind of connection we had with the electrical Snap circuit blocks. One has two female joints, butted up against each other with a male pin joining them. This is the first fluidic example of this connection, and it is an incredibly important connection type. Since it allows for direct component-to-component connection with no gap between, this connection be implemented on any of the six sides of a MEC fluidic component. *Evolution: MEC female fluid socket, allowing for component to component fluid connections.*

Shown in figure 12.19 e) is an electrical differential pressure sensor. Because it is a differential sensor, it can be used for detecting pressure or vacuum. Shown in figure 12.19 f) is an electrical latch connected to an electrical breadboard. The electrical latch latches digital signals from the Arduino. Notice that the breadboard is the same as that used in figure 12.19 a). The bottom of the breadboard is adhesively attached to a Snap mechanical baseboard, making the electrical breadboard MEC-compatible. This allows the user to easily mount the breadboard on a MEC baseboard.

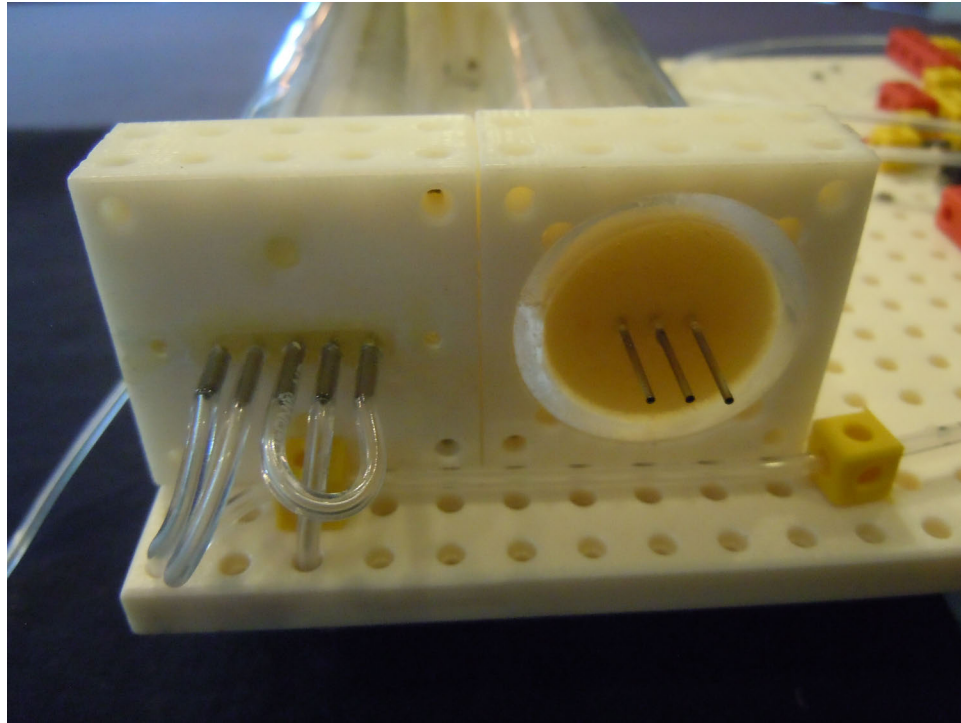


Figure 12.20: Two design revisions of the air chamber.

The air chamber in figure 12.20 on the right was the original design, and was fabricated with two 3D-printed pieces per end cap. The air chamber on the left is the next evolution of the design, and fabricated with one piece per end cap. The end cap is epoxied into place. These air chambers are tested in a bubble test up to 20 PSI. The original design of the air chamber sometimes exhibited air leakage over time. This was due to the poor epoxy adhesion between the 3D-printed part and the Plexiglas or polycarbonate tubing. When pressure builds up inside it can separate the tubing from the 3D-printed part. To overcome this leakage we designed the end cap as a single piece. This added some complexity to fabrication, because originally the outside holder just slid on. In the original design, in order to obtain Snap-compliant dimensions one simply slid both end cap holders into place.

When one epoxies the two one piece end-caps with the 1.0 inch polycarbonate tubing, it is necessary for the end caps to be kept on Snap dimensions during the epoxying process in order for them to be screwed down onto a Snap baseboard.

We used a Dimension 3D printer to print these caps. This printer uses an additive process that results in the 3D-printed material being porous. We used an epoxy well to form an airtight seal on the inside of the tube. From a safety standpoint, with each end caps being one piece and typically screwed down, when pressure builds up on the inside there is less chance of the end caps becoming a projectile. Currently, one of the air lines typically runs to an electrical pressure sensor (figure 12.19 e) and this pressure sensor connects to an analog to digital pin on the AVR. This allows us to read the pressure with an I2C query. In the future, we would want to integrate the electrical pressure sensor with an I2C interface into these air chambers. Here again is a good example of component evolution.

One additional aspect to ponder is whether to use male or female fluidic joints. My feeling is that when we are in a more breadboard-like methodology the male fluidic pins are easier to use, but as we develop more integrated systems the female fluidic pins will be preferable. Ultimately, I would expect that the female interconnection will be the preferable interface type.

Intern Hayden Karich developed the next-generation large end-cap, which press fit on the end of a cylindrical tube instead of using epoxy. It was developed for his shuttle system. The end-caps are removable so the shuttle can be loaded in the tube.

Chapter 13

Bioreactor Development

13.1 Early Bioreactor

Our very first model of the MEC bioreactor had a motor-driven pump. The bioreactor started a whole new family of MEC components based on the 0.5 inch inner diameter rigid polycarbonate tubing. An L-joint component was developed for connecting together pieces of this tubing; it has four O-ring seals, two for each tubing connection. The parts were fabricated using the UC Riverside Bioengineering Department's Stratasys Dimension 3D printer. Since the fluid contents of the bioreactor are configured in a loop, the bioreactor can replicate aspects of the human circulatory system. For example, if one were trying to test a cell surface receptor binding to a ligand on a glass slide, one could place the slide in the bioreactor loop and adjust the flow rate to match physiologically-relevant levels.

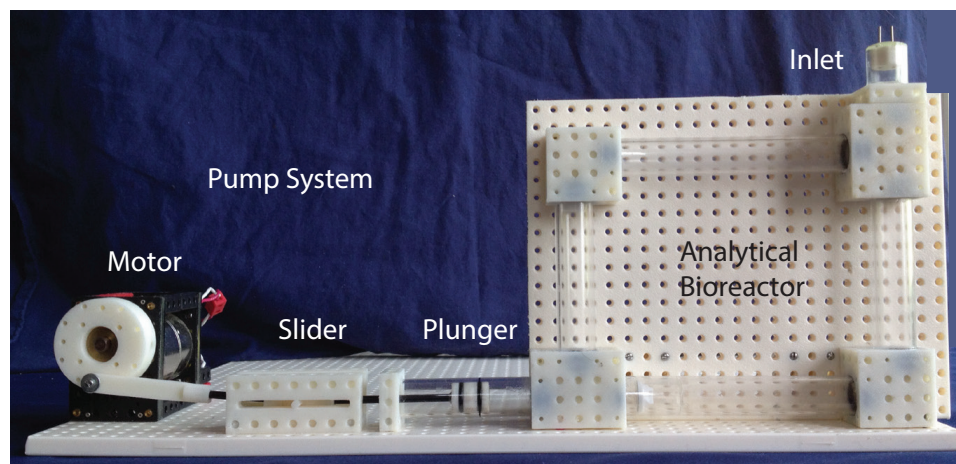


Figure 13.1: First model of bioreactor utilized a pump for fluid flow

The Stratasys Dimension 3D printer produces porous parts, so our corner joints leaked, as expected. We were able to develop an acetone sealing method for these parts, but the process proved to be inconsistent in its application. We transitioned to fabricating these parts using an outside 3D printing house and began exploring the purchase of our own 3D printer. Figure 13.2 shows the transition to the external 3D printer, and one can more easily see the O-rings inside the transparent part. The outside 3D printer also provided better tolerances, which is critical when utilizing an O-ring seal.

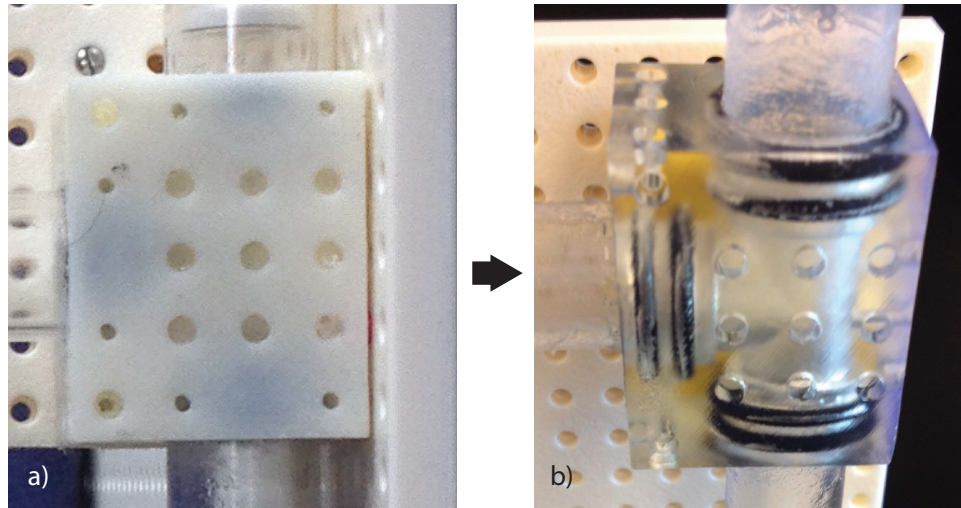


Figure 13.2: Improving the bioreactor joint MEC by switching from the Bioengineering Department's FDM printer (a) to an external 3D printing house (b).

Shown in figure 13.3 is our first working bioreactor. Here, we utilized the new 3D printed joints. The motor has been removed, and an air inlet has been added. The air inlet serves two purposes. First, it allows for gas to be bubbled through and dissolved into the media, and second, it acts like a pump. As the bubbles rise it pulls the media through the circular loop, another advantage of the loop design. 10 mL syringes were cut to form inlet connections. Also, an optical density sensor and a National Instruments data acquisition card were added to measure the optical density of cells in the bioreactor. A LabVIEW program was used for reading and saving data from the optical density MEC. The optical density sensor was made with a photodiode and a phototransistor housed in a MEC 3D-printed casing.

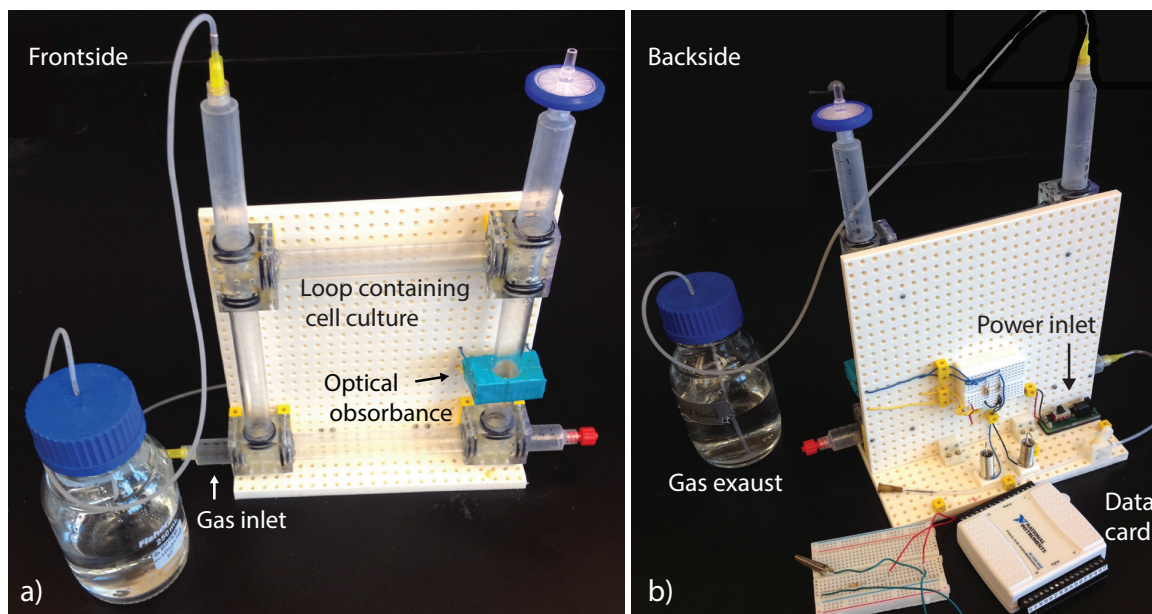


Figure 13.3: The first MEC bioreactor.

The optical density sensor was our first electrical MEC component in the new family of 0.5 tubing. As with almost all of our MEC components, we develop test methods for evaluating the components individually before putting them into a system. Here we had just a single 0.5 tube capped on one end, with the optical density sensor slid around the tube. Different concentrations of dyes were poured into the open tube to test the functionality of our optical density sensor. Afshin Mostaghim was the intern performing the testing of the optical density sensor. This work earned him coauthorship of our paper in *PLoS ONE* on the MEC system. At first, he was receiving inconsistent readings with the sensor, but when I would take the sensor home it would work correctly. What we discovered was that if one spilled water on the exposed electrical connections of the optical detector one would get false readings. This is due to the high resistance used in this circuit.

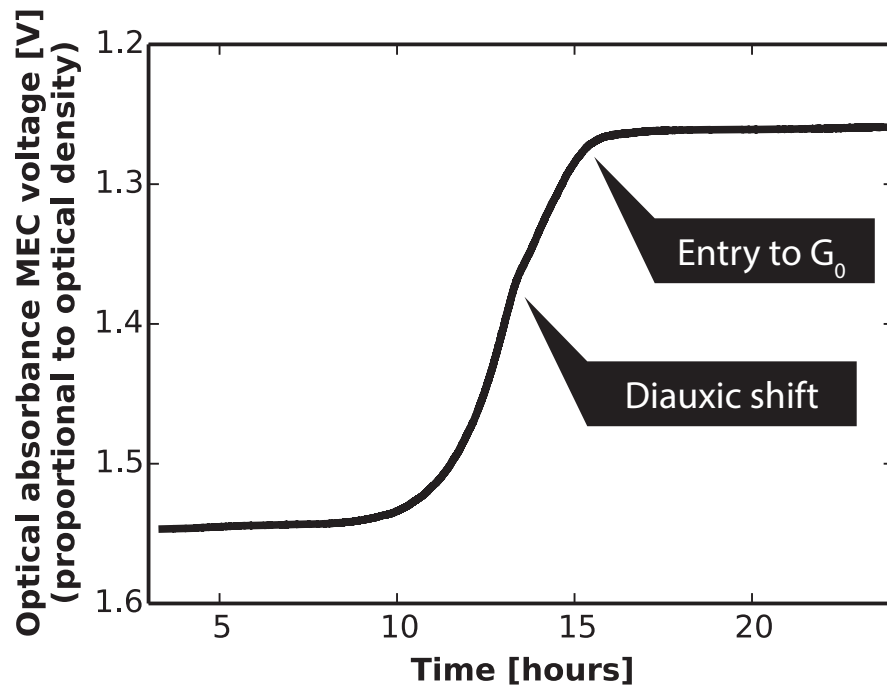


Figure 13.4: A growth curve obtained by using the bioreactor to culture *Saccharomyces cerevisiae* yeast cells. The bioreactor is sensitive enough to detect important metabolic checkpoints in the growing yeast, including the diauxic shift (when the yeast cells switch from glycolysis to the aerobic oxidation of ethanol) and the entry into G_0 (when the yeast cells exhaust all nutrients and enter the stationary phase).

When we grew yeast cells in this bioreactor to collect the data shown in figure 13.4, we placed the bioreactor in an oven to maintain an optimal growth temperature. In future bioreactors we are planning to incorporate a heater and a thermal sensor. Even in these early systems we were accurately able to track cell growth with our optical density sensor. Here, we used a National Instruments data acquisition card, but in the future we will use a microcontroller for data acquisition.

13.2 Bioreactor System

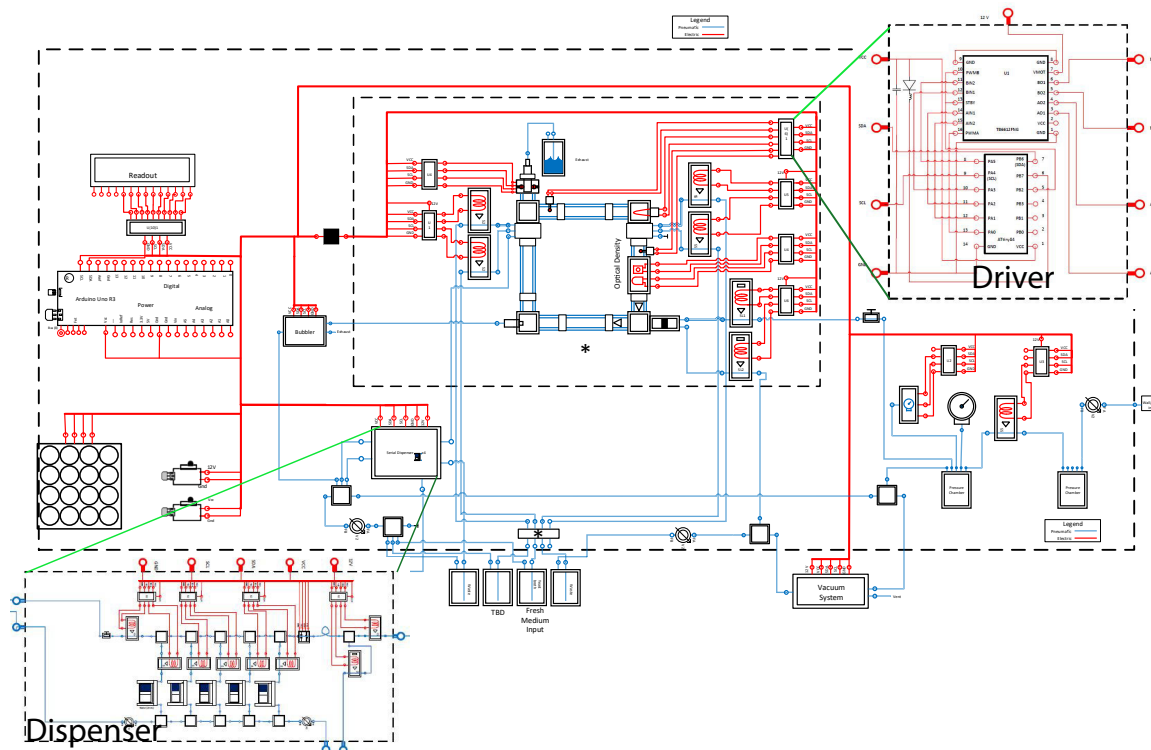


Figure 13.5: MEC Visio drawing of Bioreactor. It contains two blow-ups, one of the Serial Fluid Dispenser and one of the H-Bridge Driver

Symbolic design is an important part of the MEC engineering program. We developed the MEC schematic for bioengineering systems such as the bioreactor. Figure 13.5 shows is a schematic of the bioreactor system, evolved over time by several students. Nathan Berue was one of those dedicated students who worked on the bioreactor Visio design. On his last day in the lab, even after he had secured a job outside, he still came in to help. On that day he trained Daniel Lawrence Chang on how to develop a MEC Visio schematic, and then Daniel continued work on the bioreactor design. Students training students is an important part of the TEC Center. The schematic provides a rapid way of communicating

with my students. It provides a road map on what they will be developing. With the multidisciplinary MEC schematic the user can fully describe the functional operation of a system. As a system evolves, groups of components will be combined to form modules. The use of modules can reduce the clutter of a drawing, and proven modules become more like components. The analytical bioreactor we are developing is a powerful tool in the field of bioengineering, and it is described in greater detail in subsequent sections.

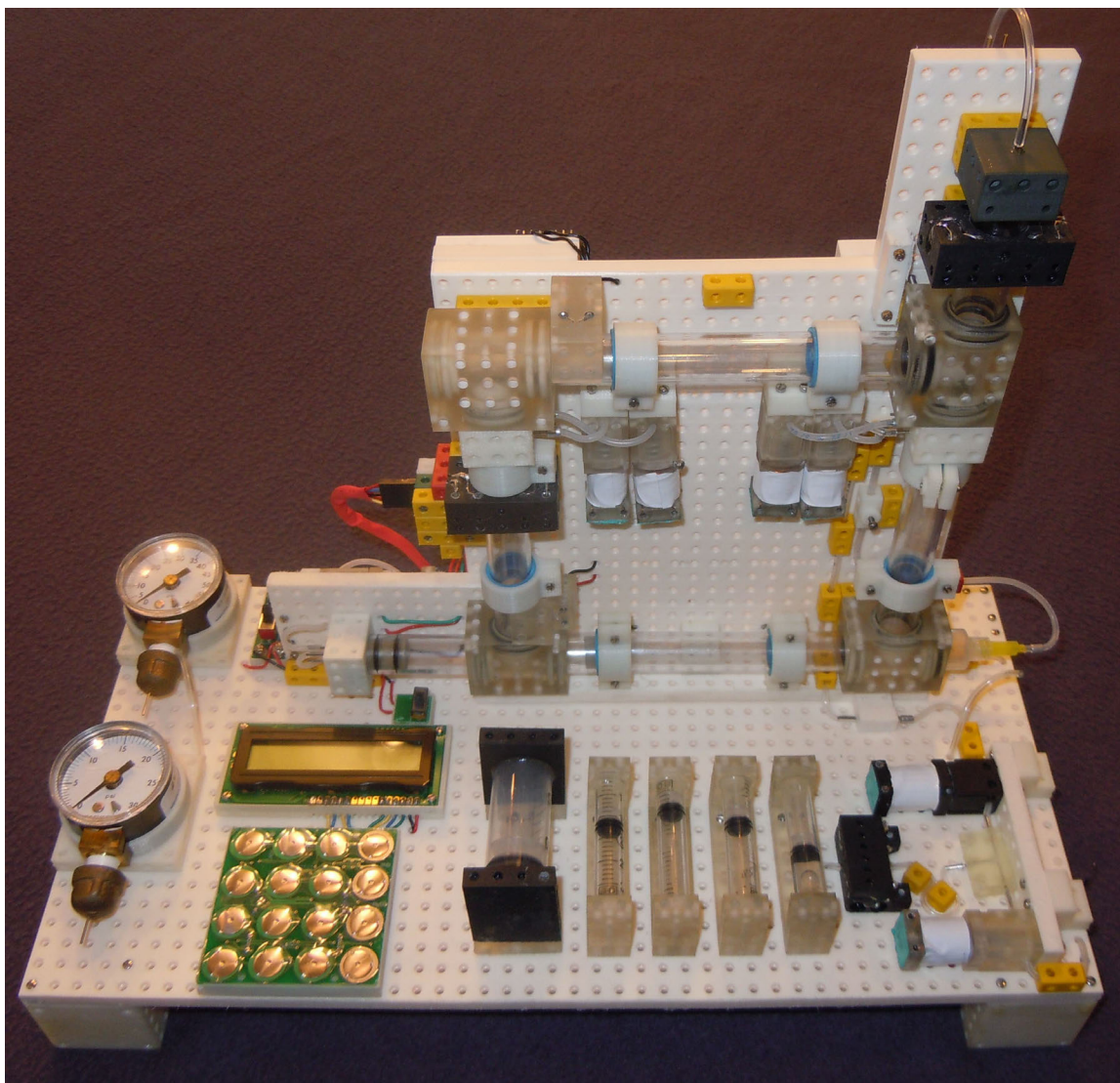


Figure 13.6: Front view of bioreactor.

Shown in figure 13.6 is the front view of the bioreactor. The reader can compare this photograph to the schematic shown in figure 13.5. The fluidic loop in the middle of the schematic is the loop shown in the front view of the bioreactor. Other items from the schematic that the reader might recognize are the keypad and display. Because the MEC

system is a true three-dimensional system, the user needs to examine the bioreactor from all sides to locate all of its components.

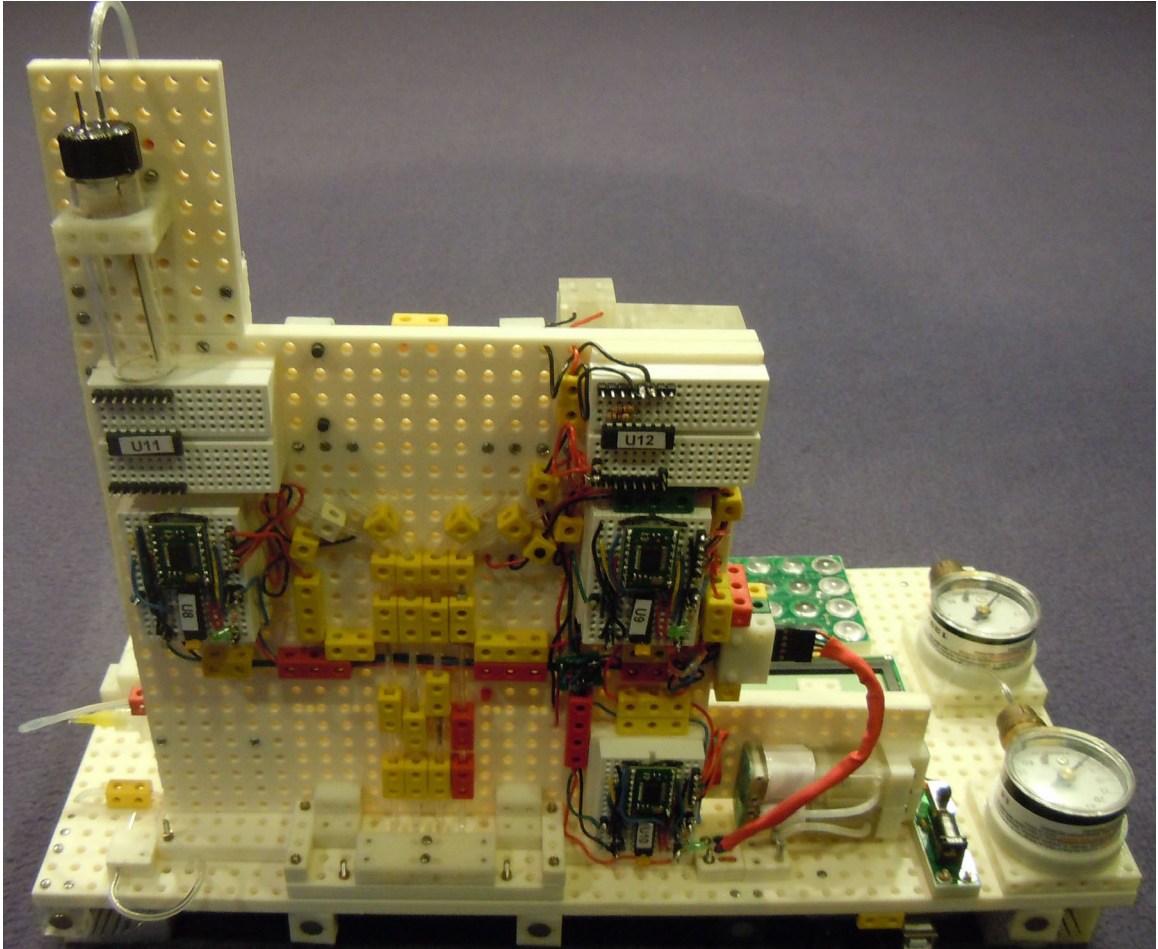


Figure 13.7: Back view of bioreactor.

The back view of the bioreactor shows several components, including the bubble bottle on top which performs like a one-way valve, allowing air to escape but still isolating the air in the bioreactor from the outside air, which can be critical for avoiding contamination. The two solenoid valves on the bottom right are used to allow pressurized air and vacuum to push and pull the piston shown on the front side. This performs like a pump,

allowing the user to control the flow rate in the bioreactor loop. A connector joins the horizontal baseboard to the vertical baseboard. This is the I2C bus coming from the Arduino to the vertical baseboard. The five-wire I2C bus interconnects the microcontrollers on the backside of the instrument. Again, this serial wiring bus structure is a critical improvement over a parallel structure. Power and USB connectors can also be seen connecting on the back of the bioreactor. A fluidic connector with four lines can be seen on the backside. The connector is used to connect to four 250 mL bottles, which could contain media, water, waste, etc. The bottles have metal plates and stick to the magnets shown on the back.

Evolution: Fluidic connectors and I2C bus connectors are evolving with the MEC system.

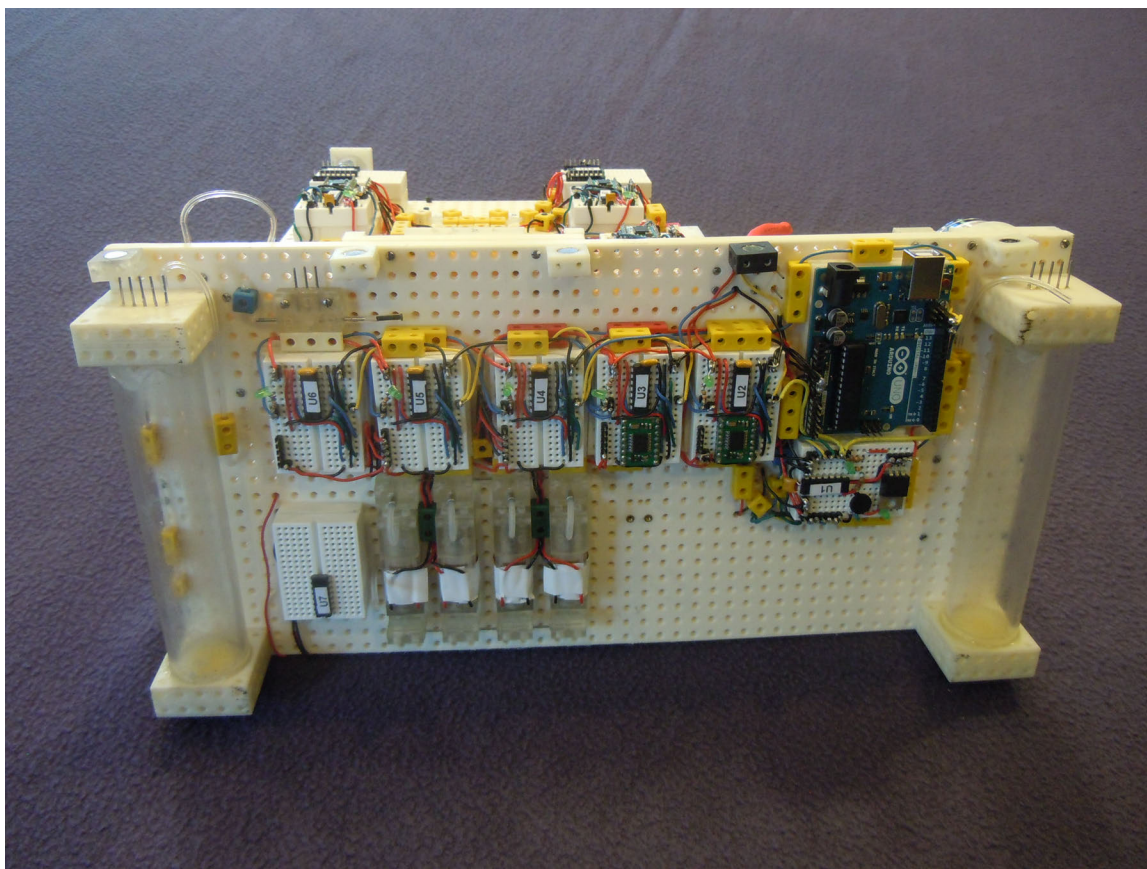


Figure 13.8: Bottom view of bioreactor

The bottom view shown in figure 13.8 has many solenoid valve drivers. They are on the small white prototype boards. The driver section blow-up shown in the left corner of the schematic of the bioreactor (Fig. 13.8) tells the user how these prototype boards are wired. When one sees several prototype boards wired the same way, it becomes an obvious student project. A printed circuit board is needed to replace the white prototype boards. Besides the solenoid valve driver, one can see the pressure and vacuum air tanks. Notice how the Snap blocks are used to hold down both the electrical wiring and the fluidic routing.

13.2.1 Bioreactor Computer

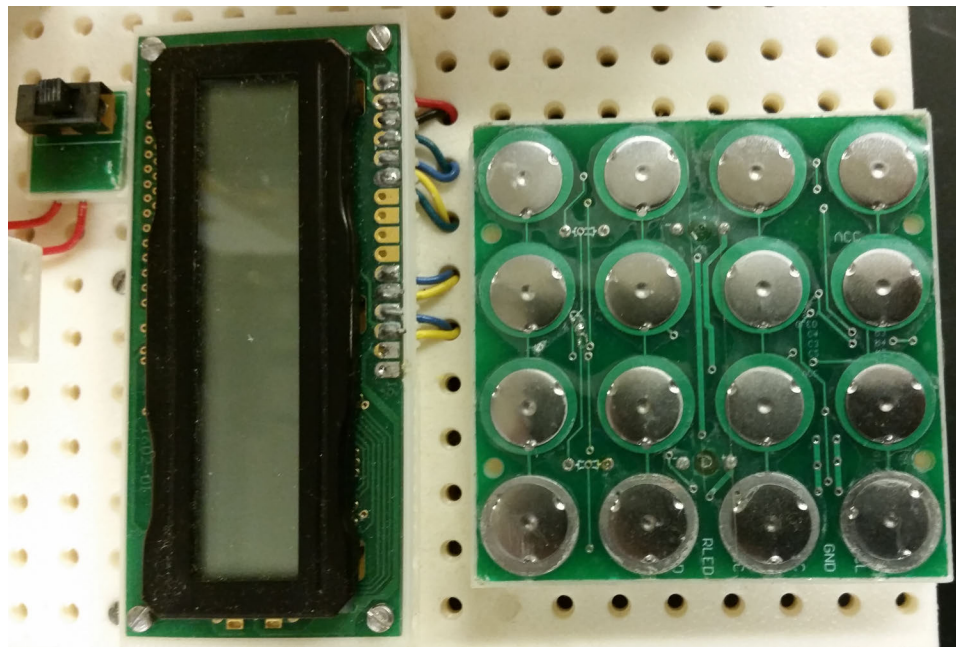


Figure 13.9: The bioreactor computer's sixteen-button key pad and two-line, sixteen-character display.

Our bioreactor is made to be fully controlled and monitored through the USB port connected to a laptop running the MEC software. In addition to this, we wished to provide the user with basic control of the bioreactor through the keypad and display MECs shown in figure 13.9. The keys or combination of keys can be custom defined. The keypad could control basic functions such as pump rate. One button could be used to increase pump rate and another could reduce it. The display could show the current pump rate. Keys could be programmed to define an entire predetermined bioreactor setup. This setup could include flow rate, media feed rate, sample feed rates, temperature, optical density level, gas input rate, and a host of other possibilities. Once a setup is selected, the user could use the keypad to scroll through the predefined settings and make adjustments as needed. If a

number of these bioreactors were in a lab, this on-board computer would reduce the need for each bioreactor to be continuously connected to a laptop. In the future, the bioreactors could be wirelessly connected to a laptop, again reducing the need for each one to have its own computer system. The software for the keypad and display has been fully tested and is operational. The keypad and display both have custom printed circuit boards developed by students. I typically sketch out an electrical design in my lab notebook, and a student then translates it into a printed circuit board. Intern Kevin Pham translated my sketches into the original printed circuit boards, making both these devices I2C-compatible.

13.2.2 Bioreactor Media Dispensing

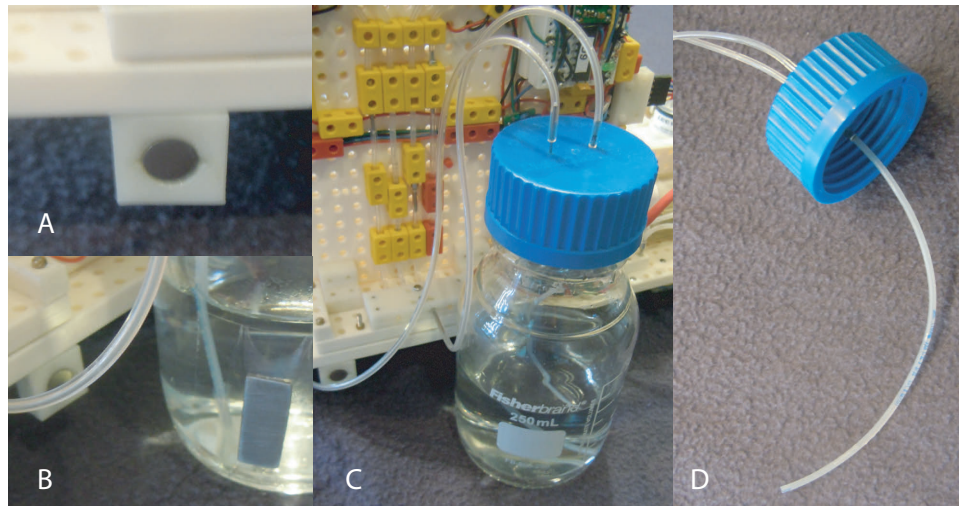


Figure 13.10: Bottle hold magnet in a Snap base (A). Metal strip on bottle (B). 250ml Media bottle magnetically attached to the bioreactor. Feed line from top of bottle runs to connector on back of bioreactor. Pressure line on top of bottle runs to pressure source under bioreactor (C). Two lines go into bottle. Feed line connects to long tube that goes to bottom of bottle. Pressure line goes to top of bottle cap and is pressuring the air over the fluid, driving the fluid out the feed line.

The Media bottle shown here is a 250 milliter bottle (Fig 13.10(C)). Media bottles come in a range of sizes, up to 1000ml, that all use the same caps. These media bottles have been modified with the addition of two 20 gauge stainless steel pins in the cap. This allows connection to the 3/32 Tygon tubing. The stainless steel pins are cut to size, and two 0.036 holes are drilled in top of the lid. Epoxy on the bottom side of lid is used to seal the pins into place. The current epoxy we have selected has proven to be a poor chose, because mold has shown up on the epoxy. Students need to develop a sealing material or method that does not allow the growth of bacteria or mold. The current system is sufficient to test the media dispensing methodology. The operation is as follows: first, media is placed in a bottle. Pressure (10PSI) is then released into the top of the bottle through one of the stainless steel pins. The fluid now wants to be driven out of the bottle. The other stainless steel pin has Tygon tubing attached to it that runs to the bottom of the bottle (Fig. 13.10(D)). The same pin comes out of the top of the bottle and another Tygon tube connects it to the connector on the backside of the bioreactor. A Tygon tube line then takes it from the connector through a MEC solenoid to the main channel bioreactor connector. Notice the yellow Snap blocks used to route the Tygon tube along the backside of the bioreactor (Fig. 13.10(C)). The media bottle is physically held to the bioreactor with a magnet in a Snap base (A), and a metal plate which is attached to bottle (B).

13.2.3 Serial Dispensing

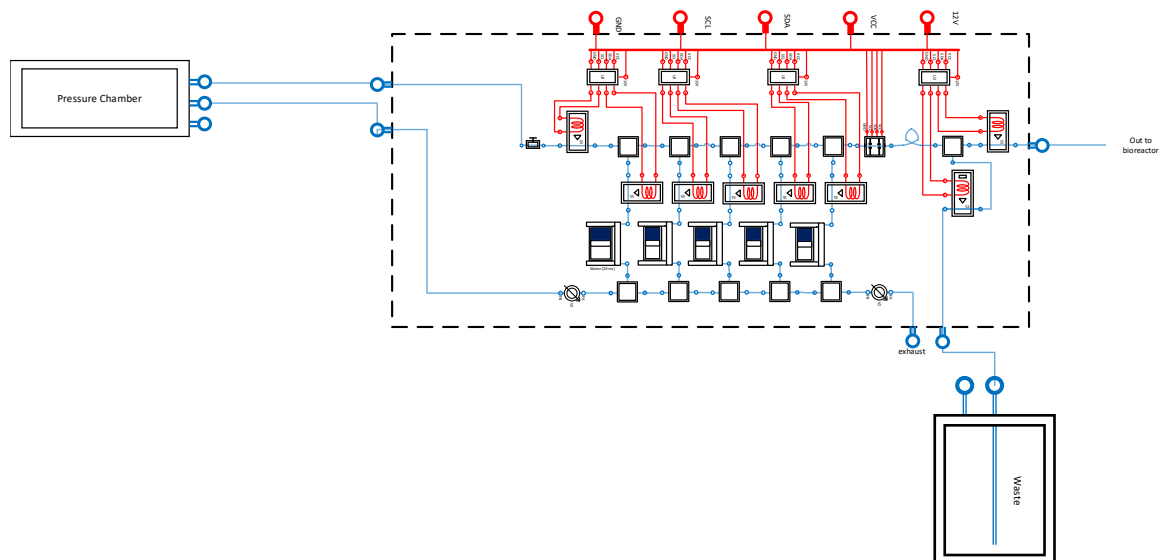


Figure 13.11: Schematic of a MEC serial dispenser, including eight solenoid valves, four sample dispenser tanks, one flush water tank, a trace optical sensor, I2C bus in red, and four H-bridge modules.

Shown in figure 13.11 is the MEC schematic for a serial dispenser. The MEC multidisciplinary schematic provides the user with a visual aid for understanding the operation of the serial dispenser. As more samples or materials need to be provided to the analytical bioreactor, serial dispensing provides a reduction in necessary interconnection overhead as compared to a parallel methodology. Whether one has four or one hundred sample dispensers, there is still only one final line that feeds into the bioreactor. Potential problems with serial dispensing is contamination between sample deliveries. This is why a water flush tank is included (Fig. 13.12 (A) Flush Water) allows the user to flush the serial line between operations.

Let us walk through the operation of the serial dispenser while looking at the schematic: On a computer the dispenser software is started. A mechanical valve is utilized to release pressure to the backside of the four sample vials and flush water. This connection to the backside of the sample vials is made through five T joints which were specially constructed for this application. The sample vials have an internal piston, which is driven forward, but is stopped because there is nowhere for the sample fluid to exit to yet. The piston provides isolation from the driving air and the sample fluid. Through the command-line interface, a command is sent through the USB to the Arduino. The Arduino then sends an I2C command out to the I2C lines (SDA, SCL) selecting the ATtiny microcontroller and the solenoid valve of interest. After the command has been clocked into the ATtiny84 on the driver module, the ATtiny84 instructs the solenoid valve to turn on, wait, and then turn off (i.e., to pulse). With this, a pulse of fluid is released into the serial line at another special T joint which connects to the main line. The pulsing sample solenoid valve is only on for a brief time. There is a manual air velocity control for the main line, which has to be set prior to operation of the serial dispenser. The main line entry solenoid and waste exit solenoid are now turned on, allowing air to travel down the main line. The sample pulse of fluid now travels down the main line, past the dual optical sensor, and the solenoid valves are turned off, stopping the fluid pulse at the serial line wait loop. The optical sensor was used to determine the exact size of the sample pulse as it passed. If the sample pulse size is correct, the main line entry solenoid valve and the bioreactor entry solenoid valve are turned on. Pressurized air then moves the pulse of fluid into the bioreactor. After an appropriate delay, the solenoid valves are turned off, completing the operation. Next, we

flush the line. We do this by performing the same operation that was just described for the sample pulse, except using a much larger pulse of fluid from the water flush tank. A pulse of flush water is sent down the main line to clean the line. Media or other fluids may be used instead of flush water if necessary. With the optical sensor we can record the amount of flush fluid that is sent into the bioreactor. If the sample pulse size is the wrong size, then it can be diverted into the waste container. *Evolution: Serial dispensing is an efficient way to deliver a large number of sample materials to a system.*

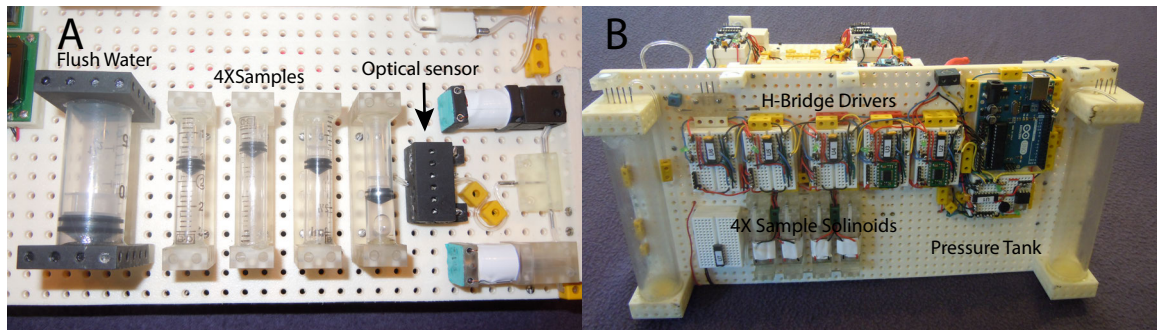


Figure 13.12: Frontside of serial dispenser (A). Backside of serial dispenser (B).

Many novel concepts are evolving in this serial dispenser. We have developed a sample vial that isolates the sample from the pressurized air that is being used to expel it (Fig. 13.12 (A) 4x Sample). This is important for preventing contamination. The sample vial plugs into a socket, so a rack of sample material could exist. Replacing material would then be as simple as unplugging and replugging in the new material. Controlling the precise volume of the pulses of the sample material can be challenging with the MEC solenoid (Fig. 13.12(B) 4x Sample Solenoid). To overcome this deficiency, we have developed the optical sensor, which will allow us to measure the precise volume of fluid dispensed (Fig. 13.12(A)

Optical sensor). *Evolution: The MEC sample vial has two important features: It is socket-able, and the piston provides isolation from the driving air, which can dispense fluid in an automated fashion.*

13.2.4 Heating system test board

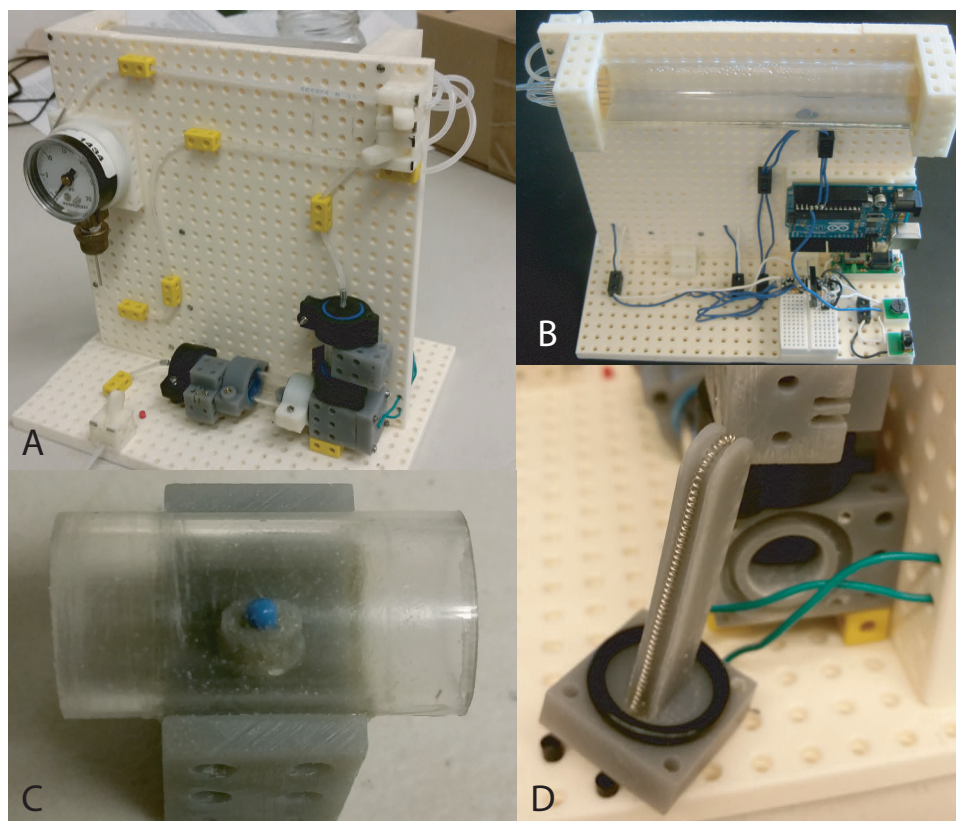


Figure 13.13: Front-side of heating test board (A). Back-side of heating test board (B). Channel thermal resistor component (C). Heating coil component. The heating coil component plugs in the end of the L channel joint (D).

The heating system for the bioreactor has undergone several evolutions. An important aspect of developing components is developing test systems to evaluate them. Shown in figure 13.13(A) and (B) is the latest test system for testing the heating unit of the biore-

actor. The heating system was first constructed and tested as an in-line heater. It was one of the first components for which we developed calibration software. The calibration routine related the actual temperature to a voltage, which came from a resistor divider network (including a thermal resistor like the one shown in figure 13.13 (C)). The in-line heater suffered from the need for wires and epoxy to be inside the main channel. At the time, we were placing the thermal coil inside a glass tube. Intern Humberto Scott Sanchez took over the project and built a test board similar to that shown in figure 13.13 (A), and (B). He wanted the thermal coil to plug in to the end of a T joint, similar to what is shown in figure 13.13 (D). At the time, we placed the coils in a glass tube and then bent the glass tube into a U. The glass tube was unsupported, and it was very fragile on insertion and extraction. Humberto did some mathematical modeling of the heating system. This type of modeling can provide important insights when developing a system, and is an excellent opportunity for a student to learn. This system and the bioreactor have two thermal resistors with the heating element in between. We plan to measure the absolute voltage and differential voltage between the two thermal resistors, and in addition to monitoring the temperature, also be able to determine flow rate. Intern Razn Qamar took over the project next. We redesigned the T joint and L joints so that the 1/2 inch tubing no longer pressed into the joints, as can be seen in earlier designs. Instead, we 3D printed the tubing coming out of the joint. If one studies the bioreactor design, one will see a short tube has been pressed into each corner joint. These can be potential sources of leaks. This new method reduces component count and the possibility of leaks in this area. It also looks to the future. The O-rings slots that were in the original design were going to be extremely difficult to inject

tion mold. *Evolution: L and T joints for 1/2 inch polycarbonate tubing that do not require internal O-rings.* 3D printing provides the capability to make designs that do not always lend themselves to mass production. Razn also drew up the design shown in figure 13.13 (D), which allows the heater to be easily plugged in. This was a great improvement over the prior method. *Evolution: Heater that plugs and then screws into a L joint, with O-ring seal.* Students need to learn the rules that apply to each type of manufacturing method that they will be using. Students need to not only need to come up with solutions, but also understand the best practices in the areas in which they are working. In 3D printing this could be the minimum thickness of a wall. For the design of a printed circuit board, it could be the minimum width of a trace.

Chapter 14

Accelerometer Weight Scale

The weight scale accelerometer is a MEC instrument designed to weigh a bottle by identifying the change in accelerometer patterns measured while different volumes of fluid are vibrated. A bottle containing liquid is mounted onto a spring platform to isolate it from the table. Embedded under the platform (Fig. 14.2(A)) is a circuit board which contains a small vibrator and an accelerometer (which measures acceleration in the X, Y, and Z directions). I began this design in my design notes from an idea I had from discussions with Dr.Grover. Intern Timothy William translated the design into a SolidWorks design, which was then 3D printed as shown in figure 14.2. Intern Krishna Addanki wrote the original accelerometer software for collecting the data. We then translated the software into the MEC Create software, which allows for automatic code generation. Timothy then proceeded to undertake data collection and MATLAB analysis. Together, we discussed different ways of processing the data.

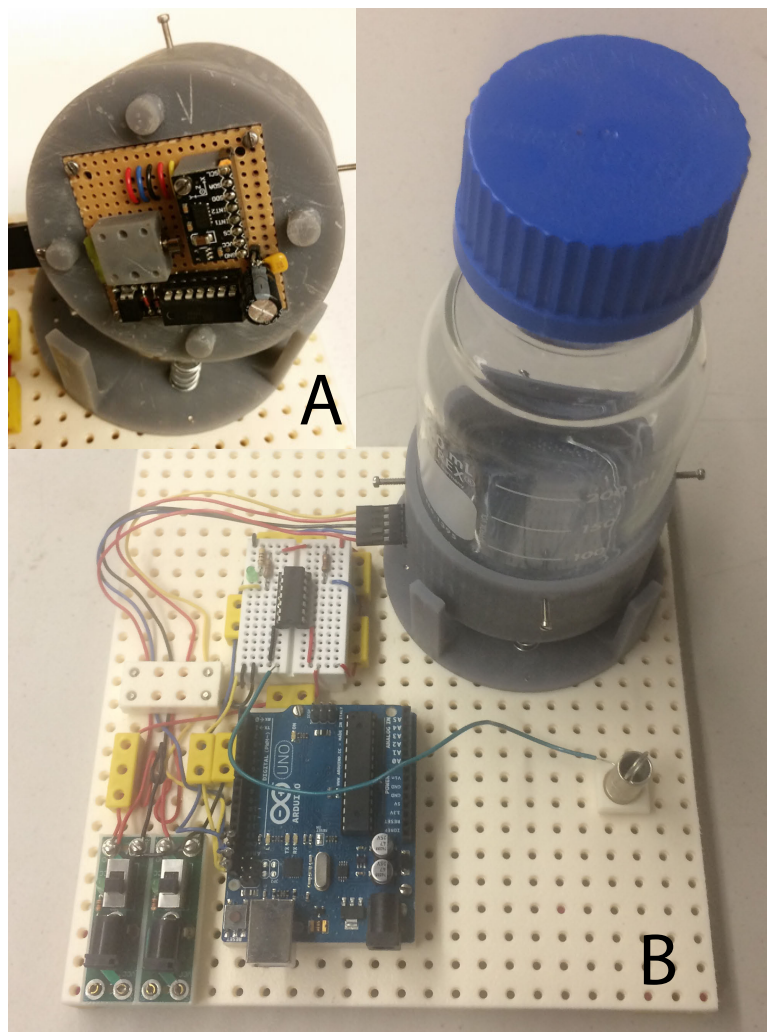


Figure 14.1: Bottom of accelerometer weight scale instrument, showing the accelerometer, vibrator, FET, decoupling capacitor and ATtiny84 microcontroller (A). Top of accelerometer weight scale showing bottle, bottle holder with embedded accelerometer, Arduino, and power interface (B).

As we placed different volumes of fluids into the bottles, we obtained different complex acceleration patterns over time. Even for the same experimental setup these acceleration patterns were different, sometimes by a slight amount and sometimes by a significant amount. We are developing automated software to read and analyze these patterns. The setup is simple and offers us the ability to collect a large amount of data, which is precisely

what is needed for developing predictive software. The first part of the data analysis was to perform a Fourier transform on the raw data. We can then statistically analyze the data in the frequency domain, determining values for minimum, maximum, mean, median, and standard deviation. One could take the area under the curve of the raw data and/or the frequency domain data. Each one of these parameters may, or may not, help us to predict the weight or volume of fluid in the bottle. Statistics for a given data set should reveal the value of each parameter in making these predictions.

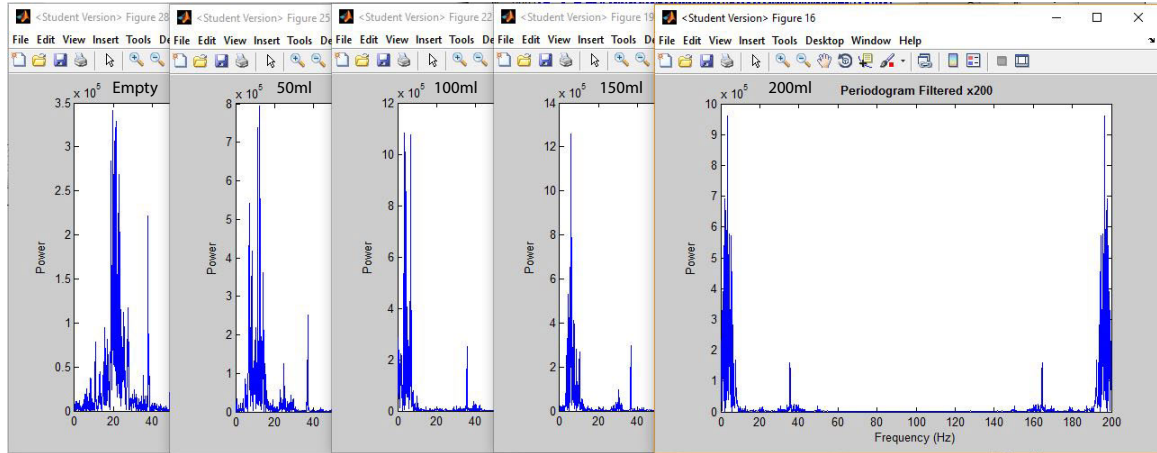


Figure 14.2: Graphs of Fourier-transformed X axis data for the accelerometer. From left to right, the graphs correspond to bottles that contain 0, 50, 100, 150, and 200 mL of water.

We commenced our testing using volumes of 0, 50, 100, 150, and 200 mL of water (Fig. 14.2). We then plan to plot the X axis with the fluid volume, and the Y axis with the measured parameter. We are repeating the tests at each volume, so the parameters have a range of values for each volume. If a parameter was found whose values were unique for a given volume, it would allow one to predict that volume if the value was determined (Fig. 14.2). For example, we are showing the frequency centered on only 20 Hz for the

empty container. For this data the empty bottle has a unique pattern. The experiment would need to be repeated several times to confirm this pattern. Even if a parameter value were not unique to a single volume, but to a subset of volumes, this could still be valuable. Here, for example, 100 mL and 200 mL have frequencies centered on 5 Hz. One would then be able to narrow it to a smaller subset of volumes that the parameter would indicate. This would enable the user to say that if the frequency center is at 5 Hz, the volume was probably 100 mL or 200 mL. We would then look to other parameters to distinguish between these two volumes. Another parameter could be the amplitude of the second-most-predominant frequency. I chose this somewhat randomly. The dominant frequency will probably come from the drive frequency. What we would like to determine is the natural resonance frequency of the system, which would relate to the mass of the system. Parameters can come from any of the different axes. There is a large number of potential parameters to investigate and as we combine more parameters with more tests, we hope to be able to make better predictions. Thus, along with our predictions, we will need to be able to determine the accuracy of that prediction. We would start by trying to predict between 50 mL intervals, and then move to increasingly finer resolutions.

In addition to this application, I have been working on the methodology of practicing early disease prediction through monitoring of several biomarkers, learning what is “normal” and then, when a shift is observed to depart from “normal,” to identifying a problem. Some of the aspects that will be needed for disease prediction may also be found in the software we are developing for bottle weight prediction. One can imagine, with our pump system, pumping fluid in and out of the system, maintaining a volume between 30

and 50 milliliters. We would be recording data and determining several factors such as key frequencies, mean values, etc. With the software we would identify patterns for several factors as normal when they were in these volume ranges. With this system we can generate copious amounts of data, and the software can be developed to teach itself what “normal” is. Then, we would shift the volume outside these ranges, observing how quickly the software could identify a shift out of normal and assessing the accuracy of its prediction. The system allows for the rapid accumulation and analysis of a large body of data, which is precisely what is needed for software development of early disease prediction. In addition to volume, we could also adjust the vibrator motor speed with pulse width modulation. This would allow us to vary two factors in the normal range, then shift one of the factors out of normal range, thus giving several ways of creating a failure condition. With the human body there are several factors that could shift one out of a healthy state, and we would want to see the effect of adjusting multiple factors on our ability to predict a problem.

Chapter 15

Density Meter

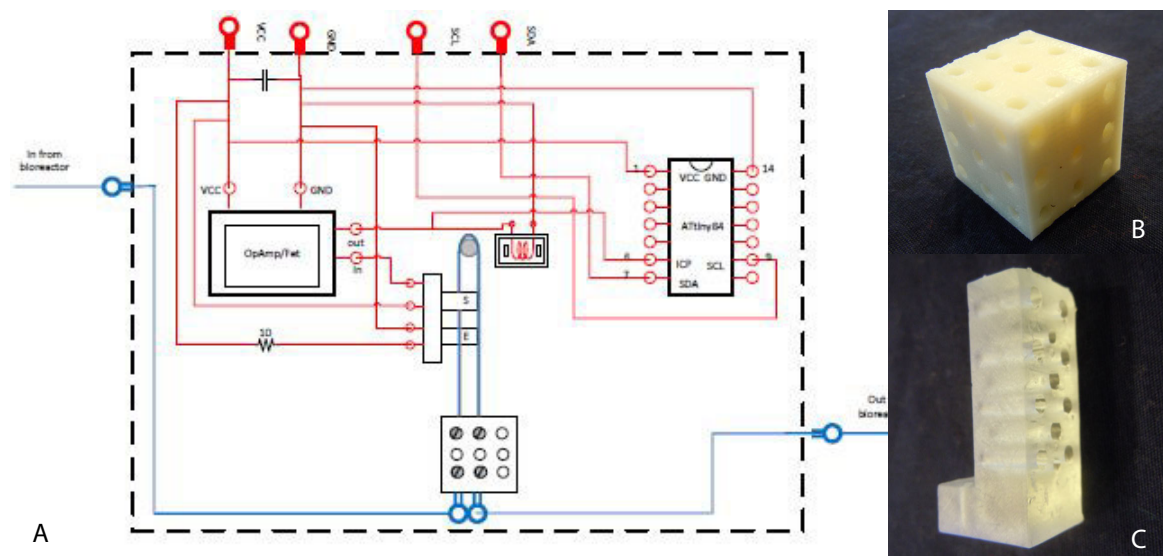


Figure 15.1: MEC multidisciplinary schematic of a MEC density meter. Red are the electrical connections. Blue are the fluidic channels. Black are the mechanical aspects of the design (A). 3D printed U tube holder block (B). Density meter optical sensor holder block (C).

The density meter is designed to measure the density of fluid passing through a stainless steel U-shaped tube. The operation of the density meter is as follows: fluid whose

density is to be tested is loaded into the U-shaped tube. The density meter is turned on. At rest, the stainless steel tube is blocking the optical sensor signal. When the signal is blocked, it causes a comparator, whose thresholds have been previously set, to turn on a field-effect transistor (FET). When the FET is on, it allows current to flow through the coil, creating an electromagnet. The electromagnet attracts a magnet mounted on the end of the U tube, causing the U tube to bend. When the U tube bends, it no longer is blocking the signal of the optical sensor, so the comparator turns off the FET. This, in turn, turns off the electromagnet. The U tube will then return back to blocking the optical sensor and the process is repeated. This system has two modes of operation. In mode 1, there is limited movement. The U tube just returns back to blocking the sensor and before it has a chance to overshoot the sensor it is pulled back toward the electromagnet. In mode 2, it overshoots the optical sensor, thus creating two pulses for every cycle. The natural resonance frequency at which this U-tube vibrates is dependent on the mass of the tube and the density of the fluid within the tube. We can therefore determine the density of the fluid because the tube is a fixed volume system. The schematic aids in communicating the operation of the density meter and becomes a “do list” on what needs to be done. In this case, we needed to develop a U-tube vibrator, which requires a block to hold the stainless steel vibrator tube (Fig. 15.1(C)). We also needed an optical sensor, which needed to be held in place and adapted to Snap spacing (Fig. 15.1 (B)). Both of these parts can be seen in the schematic in figure 15.1 (A).

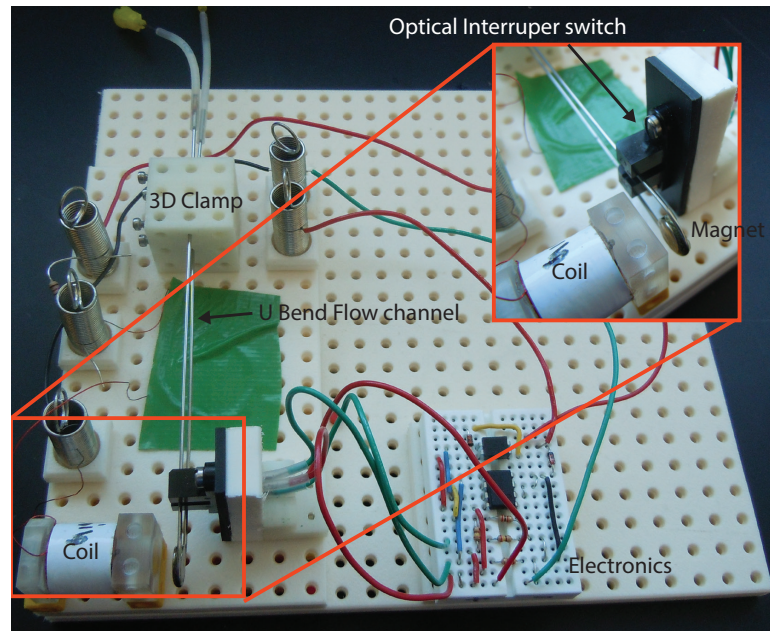


Figure 15.2: Density Meter. Shown in the figure is the electromagnet (coil) for bending the stainless steel U channel; Clamp with 2-56 screws for clamping 20 gauge U tube; Optical sensor for determining position of U-tube and then sending signal to turn off and on the coil ; Electronics on electronic breadboard for receiving signal from optical sensor and then turning on and off the coil.; Mechanical MEC baseboard for mounting components.; Spring terminal for ease of wiring of electronics

Density is a fundamental property of a fluid. In our lab we strive to measure fundamental properties, since that is a first step in the characterization and analysis of a material. To construct the instrument shown in figure 15.1, we bent a 20-gauge stainless steel tubing into a “U.” We used 20-gauge stainless steel tubing because the 3/32 Tygon tubing which it fits is standard with the MEC system. The “U” is mounted into a 3D printed base which was designed to conform to the MEC standards. It uses 2-56 screws to secure the U-tube and securely mount the component to the baseboard. A magnet is epoxied on the end of the U-tube. A magnetic coil, which was previously developed, is mounted on a MEC baseboard with 2-56 screws. They are mounted so the coil can pull

at the magnet (Fig. 15.2). A photodetector is mounted and used to detect movement of the stainless steel tubing. The simple feedback circuit is constructed on a breadboard which has been adhered to a small MEC(Snap) baseboard, making it easily mounted onto a standard MEC baseboard. As mentioned before, when the U tube bends, the signal is no longer blocked and the coils is turned off (Fig. 15.2) . We have used a National Instrument data acquisition card to measure a range of different density fluids. The instrument was so sensitive that a light tap on the table could be seen in the readings. We are replacing the National Instrument card with digital readings and recording through an AVR micro-controller. Several students have worked on the progress of this project. As a group they have progressed their hard skills in Solidworks, Visio, Eagle design and micro-controller C++ code development. I have walked several students through how the circuit functions, which allowed them to build and debugged their own systems.

Chapter 16

I2C bus device test board

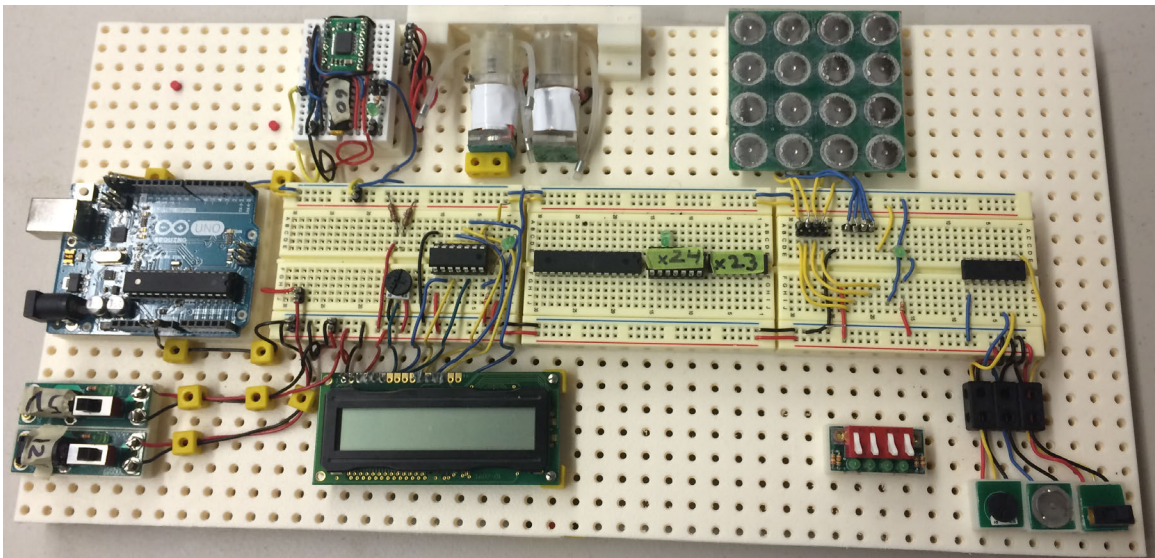


Figure 16.1: I2C bus device test board. Devices for which software has been written, and have been made I2C compatible, include: light emitting diode (LED), electrical switch, dome button, 4x4 key pad, 2x16 character display, solenoid valve, and potentiometer.

The I2C bus test board shown in figure 16.1 was used in developing our early I2C device software, primarily by intern Adam Veres and Chip Pearson. After the complex parallel electronic structure of the Coulter 2, I perceived it essential that we move to a serial

electronic bus structure. The methodology of MEC systems must be straightforward in its construction from MEC schematics, both on the hardware and software side. The parallel structure, although simpler in some respects (i.e. does not need I2C software), did not lend itself to straight forward expansion. The parallel structure would grow in complexity and become custom in both hardware and software as more complex systems were needed. I outlined the command structure and how I wanted the serial device structure to function. I built the board shown in figure 16.1, and then Adam and Chip wrote the code to make it function. *Evolution: Serial bus structure makes the addition of components into a MEC system straight forward in both hardware and software.*

The I2C serial bus and device structure were a paradigm shift over the parallel structure that we had previously been using. Previously, daughter cards with digital pin latching were placed on top of Arduino boards. The Arduino in the serial bus structure takes a much less prominent role. It acts as a translator between serial USB connections from a computer to the serial I2C of the system. I2C bus requires four wires: 5 V power, ground, SDA (Serial Data), and SCL (Serial Clock). We added one additional wire, if needed, which was 9 or 12 V power. This is used for powering solenoid valves, motors, and any other component that requires more power. This shift offered essential changes in hardware and equally important, if not more, in software. What it effectively did was allow for standardization and simplification of evolving new systems. At the beginning we had split the responsibilities for software development, with Adam being responsible for getting the user-defined signal to the I2C bus and Chip being responsible for the AVR devices receiving the signals from the I2C bus and performing the needed operations. Adam's side

included the Computer Python code and Arduino micro-controller code. Chip's focus was primarily the AVR microcontroller code. Adam got each of the components shown in figure 16.1 working in the new serial bus structure system that I defined; no small task. It is telling to see the quad-rocker switch in the lower-right of figure 16.1 not wired up. It is a component waiting for software to be developed. As new components are added, it can change some of the basic software structure. A quad-rocker switch could be thought of as four switches, but currently switches require a timer to alert software to see if the switch has changed states. So, four switches means four timers. Timers in a microcontroller are a limited resource. This is inefficient. The same timer needs to be shared between switches. I have had this discussion with Adam and it is simply a matter of changing how that part of the software functions. Currently, Adam is working on a GUI menu interface to replace the current command line interface. This is a higher level software challenge and therefore a better use of his time. More experienced software interns are needed. Adam wrote a tutorial for the MEC interns on how to write software for adding and using devices with this serial bus structure. Even with Adam's tutorials guiding them, our interns have limited software experience and were having trouble developing the code for the system. I saw this as enough of a problem that I taught myself Python and proceeded with successfully building our automatic code generation software. This software currently can generate the core of the code needed for a new system and can test each of a new system's components. This conforms to my desire to make the work reusable.

To use the I2C test board in figure 16.1, a computer (often being directed by the user from a menu) sends a command set to the Arduino. The Arduino translates

this command set to the I2C bus. The command set starts with an AVR micro-controller address being sent down the I2C serial bus to select the AVR microcontroller, which then acknowledges its selection. Then the device ID is sent down the I2C serial bus, along with a command to the selected microcontroller. The command could be as simple as turning on a light emitting diode (LED) or reading the state of a button. Looking at the I2C test board, one can observe some of the first I2C devices that we developed, which included an LED, button, potentiometer, LCD display, 4x4 keypad, and solenoid valve (Fig 16.1). Many of these devices were originally tested with the parallel electronic structure.

Chapter 17

Software

We have been evolving the software for the MEC system along with the hardware. The software's first job is the control of the MEC instrument. Just as the MEC hardware makes creating new instruments more easy and flexible, creating the software for these instrument must also be easy and flexible. Instrumentation software has two sides: the computer or laptop side, and the microcontroller instrument side. On the computer side we started with C++ for a base language but quickly moved to Python. I wanted to create a MEC scripting language geared to allowing the user, not skilled in programming, to program the operation of a MEC instrument. We created a translator that could translate MEC script to Python. The translated Python could then be interpreted to run the operation of the instrument. We wanted regular Python code to be combined with the MEC script, providing a large command base to start from. Adam Veres has been the key student writing Python code at the TEC Center. On the instrument side, we started with the Arduino as the interface to the computer and the primary control of the system. Prior to the serial

bus structure we used a daughter card with latches on top of the Arduino to enhance its operation. The Arduino controls the instrument through its ports. To add more devices or functionality to the Arduino, we added latches which would be addressed through a parallel bus. This methodology quickly led to semi-custom hardware and software, which is counter to the MEC philosophy (a standard building-block approach to instrument development). I wanted to move to a serial bus structure based on the I2C protocol. Moving to a serial bus structure simplified both the hardware and software control. As new devices are added, one can simply add a microcontroller onto the already existing I2C bus. *Evolution: Serial bus structure is an important step forward.*

This transition opened the door to automatic code generation because the process of adding devices to a system became so straightforward. We were very lucky to get the aid of Chip Pearson, an experienced programmer, in the development of the I2C C++ microcontroller code. He wrote most of the original microcontroller code and set up the original code structure which I will be discussing here. I defined what I wanted but he did the actual original code writing. The USB on the computer side is connected to an Arduino on the instrument side, for serial communication. This is where commands are transferred from the computer to the instrument and where data flows back from the instrument to the computer. I organized the user definable Python (computer side structure) code into four folders: Configuration, Menu, Macros, and User Main. The Python code is defined through MEC scripts.

17.1 Computer side structure (Python)

We eased the burden to the programmer or user on the computer side through the use of the MEC scripting language, which included scripts for Configuration, Menu, Macro, and User Main. All these scripts are then translated into Python code. Other MEC scripts that have been developed but not discussed in this section are the MEC scripts for data processing.

17.1.1 Configuration

The configuration script is utilized to define the variables that the user desires to use in the instrument system. Along with the variables and their data types, one also defines their default values and the variable ranges. In the menu code that is generated, if the user tries to set a value outside the range, they will be directed to try again with a value inside the range. This can be particularly useful on high-current devices which can be damaged if turned on for too long. In addition, in the configuration script the user defines the devices that will be used in the system. Part of defining a device is giving it a unique device number and assigning it to an AVR microcontroller. In the future, this section will also include pin definitions and any other specifications that are needed to define the device. Currently, this information is held in a dictionary variable. This method is not optimum when creating a new system from an existing system, and should be part of the configuration text file. *Evolution: The MEC configuration script makes it easy for a user to define the devices and variables needed in a system. Translation from MEC configuration*

script to Python variables and code is an important achievement. Future Evolution: The MEC configuration script will be generated from the MEC schematic.

17.1.2 Menu

The menu script is another example of simplifying what the user needs to input. The menu script lines are primarily of four types: first, they are used to allow the user to jump to another menu. For example, from a main menu one could go to a sub-menu on the LED (specific device menu). Second, a device like an LED has functions. These functions will be defined or called by macros. Therefore, the next menu line type is used to call those macros. For the LED the macro functions are: turn on, turn off or toggle. The third menu script line type is used to SET variable values. For example, the user could set the pulse time on how long to leave an LED on. The fourth line type is “done.” This will exit the menu and take the operator to the next higher level menu.

When the menu is being defined by the user, “*” is placed in front of the menu and these will be converted to the appropriate number by the Python translator. A “#” can be used at the end of any menu line to add comments. These comments show up to the user if they select, for example, “i2”, which would give him the information or comments for menu option 2. *Evolution: MEC menu script simplifies the user’s skills and time required in menu creation for the instrument interface. Translation from MEC menu script to Python code is an important achievement.*

17.1.3 Macro

The two most common Python commands are to obtain a variable value from the user-defined dictionary of variables and to perform the function of a device (object). Macros can range from very simple to very complex. The macro translator, when translating from user macro script to Python, adds in all necessary imports. The macro translator also leaves the door open for MEC specific script. *Evolution: MEC Macros allow user to develop reusable complex software functions.*

17.1.4 User main thread

When the user runs the Python instrument code there are two threads that are created. The first thread runs the Menu or Terminal thread and the second runs the User Main thread. The User Main thread is where the device or system input and output interconnections are defined. It is where the user defines the operation of the instrument. Currently, the user has to write Python code to define these interconnections. This is simple code, yet it could be the most complex code required for the user to write. We plan on creating a MEC script to define these interconnections. The folder for this section is where the User Main script would be held. *Evolution: User Main Script folder becomes a logical place to define a systems operation, which often takes the form of a state machine.*

17.2 The code structure in C++ developed for the micro-controllers of the instrument.

Microcontroller programming is facilitated by the straightforward device structure that has been developed. Our I2C command structure consists of a minimum of five bytes:

- Byte one: eight bits define the microcontroller to which the user wishes to talk.
- Byte two: eight bits define the device one wants to select on the micro-controller.
- Byte three: eight bits define the command.
- Byte four: eight bits define the extension (the additional bytes that are needed to perform the command).
- Byte five: eight bits define the number of bytes one will be returning.

After this would come any additional bytes that are needed, which would be defined by the extension. I wanted to go with this structure so that if, at some point, we wanted to create a list of commands that did not need the extension, or to return bytes, we could have the software identify these commands, allowing the additional bytes to be left off. If the number of bytes in the command would have been first, then removing the extension and return bytes would not be possible. The main microcontroller we use is an Atmel ATTiny 84. Code is created for each device and a microcontroller can contain multiple devices. The user defined code for each device connected to a microcontroller is broken down into three sections: Initialization, Service and Access. There are a number of additional critical functions that fall outside these sections, which are the same for all the microcontrollers

in the MEC system. *Evolution: Starting with a professional AVR C++ software structure eases the challenges in the codes growth.*

17.2.1 Initialization

As the name suggests, this section performs initialization. It does a small amount of initialization for the microcontroller, such as making sure that there is an internal pull-up on the reset line. Most of the initialization code comes from each of the devices that are selected for this microcontroller. The code automatically cycles through the initialization that is defined for each device. After initialization the software goes into a “while loop” which calls two functions; Service and Access.

17.2.2 Service

Often, we want to have multiple functions running simultaneously. This was handled with our Python code on the computer using threading. With our microcontrollers we handle this with a function we call Service. The routine has timers which are set, for example, every millisecond. If a time is set, it clears the timer and performs the appropriate service routines. This could be for periodically putting a pulse on an output line, for example. Or, it could be used for checking periodically as to whether a button has been pressed. *Evolution: Service routine is an important software construct. It provides for the effective software threading of several activities.*

17.2.3 Access

This is where the commands are received and executed, such as “turn on the LED.” Again, each device has its own access commands and the software will first identify the device the user wishes to communicate to, executing the appropriate command for that device (defined in the serial byte sequence).

17.2.4 Additional critical functions

There are several functions or codes which are common for all the AVR devices.

These include:

```
access.c
access.h
base_command_table.c
base_initialize.h
command_tables.h
eeprom_util.c
flash_tables.h
flash_table.s
initialize.c
main.c
service.c
service.h
sysdef.h
sys timer.c
sys timer.h
usiTwISlave.c
usiTwISlave.h
```

Chip Pearson wrote most of these. The I2C structure requires some sophisticated software.

The usiTwISlave has 616 lines of code by itself. It is the code primarily responsible for the I2C communication. In sum, there is some upfront work in implementing I2C, yet it is well worth it to create a simplified hardware and software structure. This straightforward structure makes it easier to integrate in new code.

17.3 Automatic code generation

Once this software was formed we began creating the software for individual devices and instruments. Adam Veres created excellent tutorials on how to create and use the devices and how to use the overall software. Outreach became our first large test case with several software engineers. I gave talks on how the software was structured. In the end, even though the user did not need to worry about most of the code, it was still too daunting for a new programmer to work within such a large code structure. My son is a computer science programmer at UC San Diego. He had a summer internship at Amazon where he was required to write code into an already massive code system. He was very successful. My point is that being able to work within an already existing system is important and something our computer science students need better training in.

The MEC system is being created to lower the barrier to instrument creation. At this point, code software creation became a limiting factor. We had tried to make code creation straightforward, but it still was not straightforward enough. I took three steps to remedy the problem. First, I restructured Adam's working code to make it more straightforward to step through. Second, I wrote code to automatically generate the entire system. Automatic code generation is proving to be a highly successful approach. It is successful for several reasons that go beyond the simplicity of creating a new system. Third, I created a user main thread.

17.3.1 Instance of a system

The MEC automatic code generation software is called the Create software. The Create software generates all the standalone new code for a system, which I call an Instance. The first step in making the code less daunting for the user was the creation of these instances of a system. Each instance has three different code sections that are auto-generated: Python (or Script from which Python will be created), AVR C++ code, and Arduino code. The instance of a system can be turned over to a user to run a system and can be modified in whatever way the user needs, without the risk of causing problems to the Create code. This had proved to be a problem with the original method. When we were working on a new system, we might inadvertently effect the function of another system (breaking the code for the older system). Instances have also proved powerful in generating new devices. Once again, an instance can be generated and modified to create a new device. The user then tests the new device. Once the device is functional, one can then integrate it back into the Create code, so that it can be auto-generated. The user knows precisely what they want the automatic code generator to create since they already have a working instance.

Evolution: Automatic code generation allows users who are unskilled in programming to develop complex systems. Evolution: The instance of a system allows the user complete control without the fear of damaging original code.

17.3.2 Only the code the user needs is generated

In the prior coding method, every function that was needed for any system was part of the software. Thus, if one wanted a system to turn on an LED, one still would have

the software for data logging an analog port (it would not be used, but it would be there). For a new user, the amount of code that was present was overwhelming. With automatic code generation, an instance only has to include what is needed for the desired application, simplifying the code for the user.

17.3.3 User main thread

We now have a system which is very good at creating controllable devices, such as an analog-to-digital port for reading pressure sensor gauges, or a digital port for turning on and off an LED. We are automatically provided menu options for testing the devices we create. We can create variables, yet what we are missing is a straightforward way of integrating these devices. This had been done with the macro software. For a simple system this works, yet for more complex systems a state machine configuration is better. Therefore, I created a user main thread. I designed it so that the main thread would not run until it was turned on from the menu. This is the next place in the code that I need to update. Currently, the user main thread code is written in Python, making it the most complex code the user needs to generate. In the future, I will generate MEC script that defines these interconnections. I will also utilize menu options for defining what script is needed. Even at this stage I was able to have one of my students (Raymond Wan, who was unfamiliar with Python) successfully modify the Python code. He did this while trying to get the marble machine operational.

17.3.4 Walking through the steps in automatic code generation to create software for the Test Board.

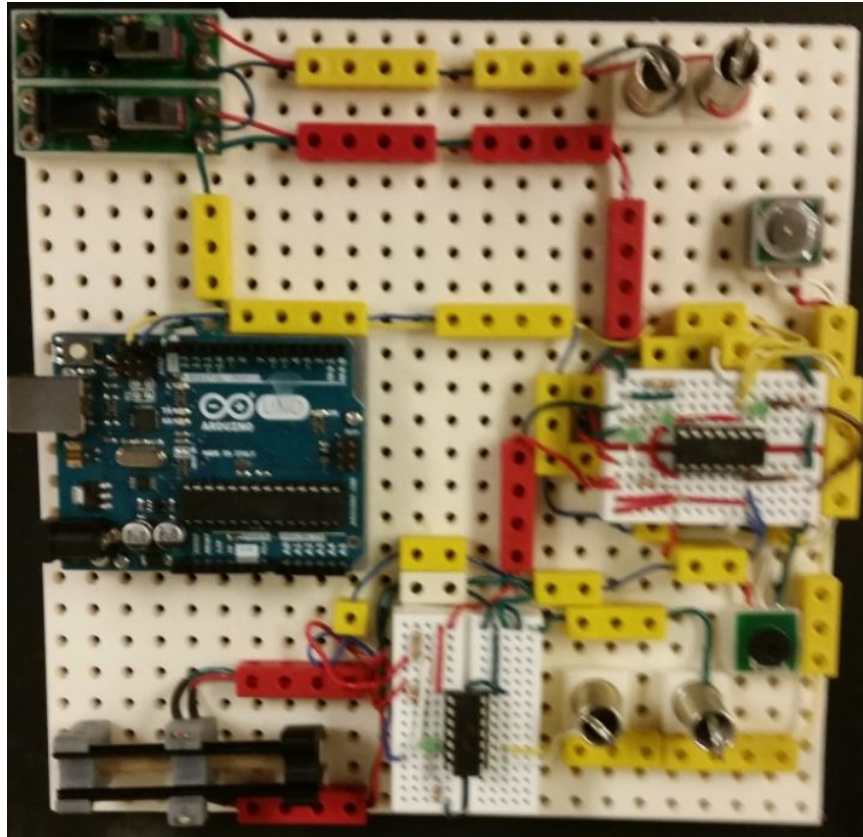


Figure 17.1: MEC device test board containing two ATtiny 84 microcontrollers (AVR1 and AVR2). The devices on AVR 1 include a button, two LEDs, and a potentiometer. The devices on AVR2 include a marble sensor, a green LED, and a muscle-wire valve.

Many of the successful systems we have built have been for the testing of components, such as the one shown in figure 17.1. Testing the reliability of components is a key step in their development. The test board shown here can test several devices on two separate AVRs. This test board was primarily developed to test devices whose software was automatically generated. Here, we will walk through the steps for generating the software for this test board.

Device	Pin	Device	Pin	Device	Pin	Device	Pin
ATTiny84	1	Power Plug	(red) VCC	ATTiny84	1	Power Plug	(red) VCC
ATTiny84	2	NC		ATTiny84	2	R3 (390 ohm)	1
ATTiny84	3	NC		R3 (390 ohm)	2	IR Emitter LED	Anode
ATTiny84	4 (reset)	NC		IR Emitter LED	Cathode	Power Plug	(green) GND
ATTiny84	5	NC		ATTiny84	3	NC	
ATTiny84	6	R1(3.9K ohm)	1	ATTiny84	4 (reset)	NC	
R1 (3.9k)	2	Power Plug	(green) GND	ATTiny84	5	Green LED	Anode
ATTiny84	6	Button	1	Green LED	Cathode	R1 (390 ohm)	1
Button	2	Power Plug	(red) VCC	R1 (390 ohm)	2	Power Plug	(green) GND
ATTiny84	7(SDA)	Arduino	(Blue) SDA	ATTiny84	6	R2(10K ohm)	1
ATTiny84	8	Green LED	Anode	R2 (10k)	2	Power Plug	(red) VCC
Green LED	Cathode	R2 (390 ohm)	1	ATTiny84	6	IR Photo Transistor	(green) GND
R2 (390 ohm)	2	Power Plug	(black) GND	IR Photo Transistor	green	Power Plug	(red) VCC
ATTiny84	9 (SCL)	Arduino	(Yellow) SCL	ATTiny84	7(SDA)	Arduino	(Blue) SDA
ATTiny84	9 (SCL)	R3(3.9K ohm)	1	ATTiny84	8	Output 1 spring	High side
R3 (3.9k)	2	Power Plug	(red) VCC	Output 2 spring	Low side	Power Plug	(green) GND
ATTiny84	10	Potentiometer	2 (center tap)	ATTiny84	9 (SCL)	Arduino	(Yellow) SCL
Potentiometer	1(top tap)	Power Plug	(red) VCC	ATTiny84	10	NC	
Potentiometer	3(bottom tap)	Power Plug	(green) GND	ATTiny84	11	NC	
ATTiny84	11	NC		ATTiny84	12	NC	
ATTiny84	12	NC		ATTiny84	13	NC	
ATTiny84	13	Green LED	Anode	ATTiny84	14	Power Plug	(green) GND
Green LED	Cathode	R2 (390 ohm)	1				
R2 (390 ohm)	2	Power Plug	(green) GND				
ATTiny84	14	Power Plug	(green) GND				

AVR1_B6L8ADC10L13	AVR2_L2L5ADC6O8
B6 -Button	L2 -IR <u>Emmitter</u> LED
L8-Green LED	L5-Green LED
ADC10-Potentiometer	ADC6-IR <u>Reciever</u>
L13-Green LED	O8-Output

Figure 17.2: Netlist for AVR1 (A) and AVR2 (B). A netlist defines the pin-out interconnection between devices. AVR 1 has devices defined in figure 17.1. AVR 2 has an IR emitter, IR receiver, green LED, and output.

There are two methods we use to define a system: one is with a symbolically defined schematic, and the other is with a netlist. A schematic takes longer to create yet gives a better visualization of the system. For automatic code generation the netlist is sufficient for providing the information needed. If one looks at the bottom of the netlist for AVR 1 (Fig. 17.2(A)), one will see “AVR1 B6L8ADC10L3”. This is often how I name a system. Here, I chose “Test Board,” which is much less descriptive. The B6 in the sequence lets me know there is a button on pin 6; looking above into the netlist will confirm this (Fig. 17.2). The 8th line in the netlist indicates that pin 6 of the device Attiny84 is connected to pin 1 of the button. L8 means there is an LED on pin 8. ADC10 means we have a potentiometer on pin 10. L13 lets us know we have another LED on pin 13. Looking at the bottom of AVR 2 netlist (Fig. 17.2(B)) one will see “AVR 2 L2L5ADC6O8”. A marble sensor is made up of two basic devices. In the future, once the marble sensor device is

defined by the two basic devices, one would simply need to say that they have a marble sensor device. For now, one needs to call out the individual components. In this case the devices are L2 for the IR emitter LED and ADC 6 for the IR receiver. Also, note that for the muscle-wire valve we utilized a feature called an Output. There does not need to be a difference from a software perspective for an LED device or a digital Output feature. If I state that I have an LED device, it brings the correct image to mind. If I'm using an Output of AVR 2 to control something, such as a muscle wire, it also brings the correct image to mind. In the future, I will allow the user to change the name of the Output.

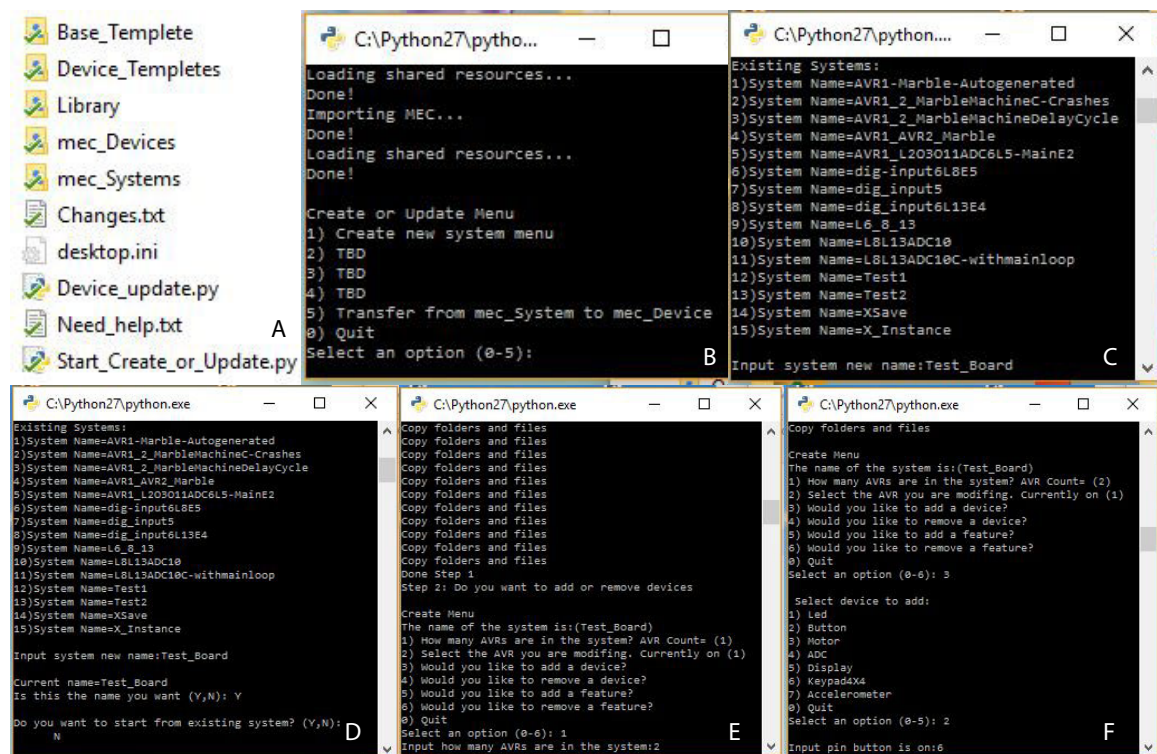


Figure 17.3: Double left click “Start_Create_or_Update.py” to start Auto generation software (A). Select 1 to create a new system (B). Type the name of the new system (C). Confirm that name is correct and we will not be using a pre-existing system (D). Input the number of AVRs in the system (E). A Button is selected as the first device in AVR1 (F).

The auto code generator will generate the Python, AVR and Arduino code. The procedure starts with a simple double-click as shown in figure 17.3 (A). The name that the user specifies here will be the instance of the code for this system, showing up in the system's folder at the end. In the future, we will allow the user to start from a prior system as a base. Selecting how many AVRs are in a system is as simple as selecting option 1 and then entering the number of AVRs (Fig. 17.3 (E)), in this case 2. Option 3 is used when one wants to start. Notice on the line of option 2 that one will see a 1 at the end. This lets the user know that they are currently modifying devices in AVR 1.

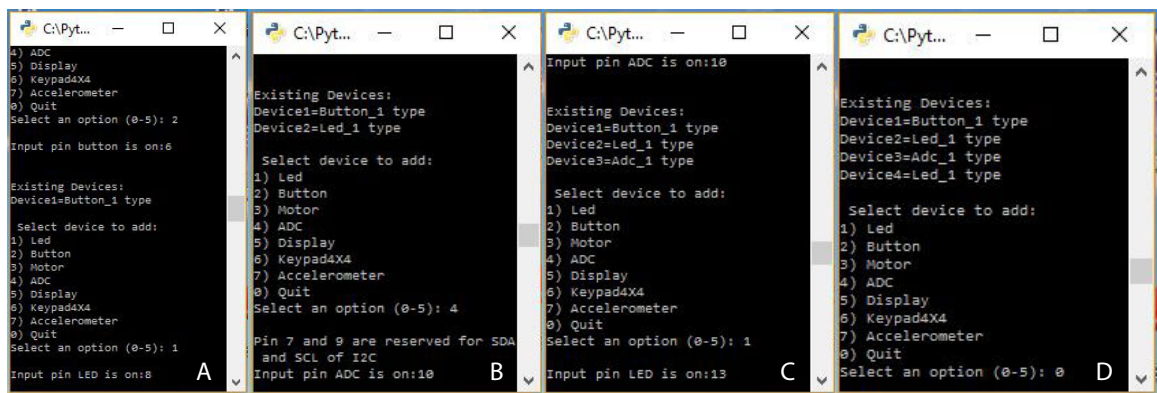


Figure 17.4: Menu options for selecting devices. Next device in AVR 1 is an LED, and we put it on pin 8 (A). Next device is an Analog to Digital Converter (ADC), connected on pin 10 (B). Finally , we place a green LED on pin 13 (C). We then use option 0 to quit this menu (D).

Looking at figure (A), the user can see under “Existing Devices:” Device 1 Button_1 type. This lets the user know a button device was the first device added to this AVR. In this menu sequence we add an LED, analog-to-digital converter (ADC), and another LED. With each device we assign the pin it will be using. A device is not required to use just one pin, but these do. Currently, 5 volts is the default reference voltage we are using. In the future we will provide the user with other options for the reference voltage. Also, we

are using a 10 millisecond sampling rate for the ADC. In the future we will allow for other sampling rates. Each of these become menu options once the ADC device is selected. If the user does not select those options the default values will be used as before.

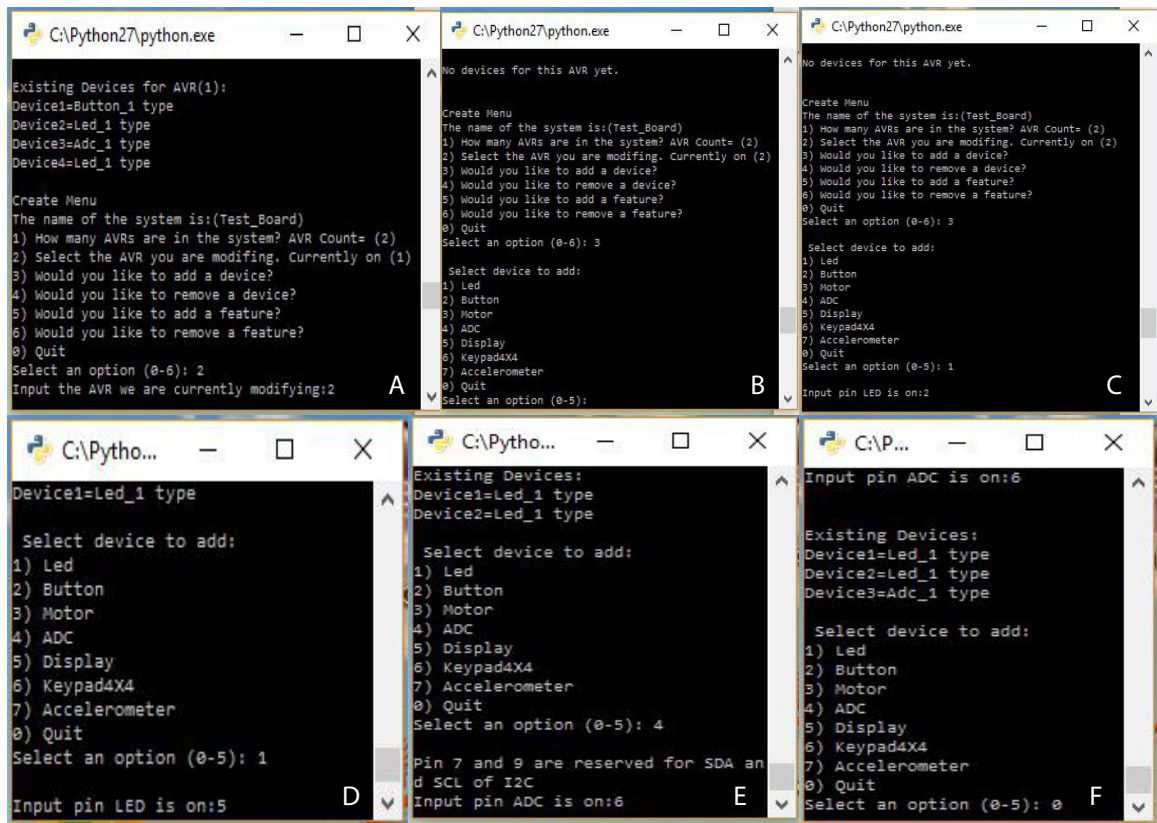


Figure 17.5: Set of menu options used for selecting devices for AVR 2. First, we switch to AVR 2 (A). Option 3 is for “adding device” (B), We select LED and place it on pin 2 (C). We select another LED and place it on pin 5 (D). We then select an ADC and place it on pin 6 (E). We are now ready to quit device selection so we select option 0 (F).

Because of the serial I2C structure, adding and wiring additional AVR micro-controllers and the devices they control is very straightforward. The software is equally straightforward for adding devices. Here, we switch over to AVR 2 to add devices to it (17.5 (A)). This simplicity was not an accident, but something I saw a real need for as the MEC system grew.

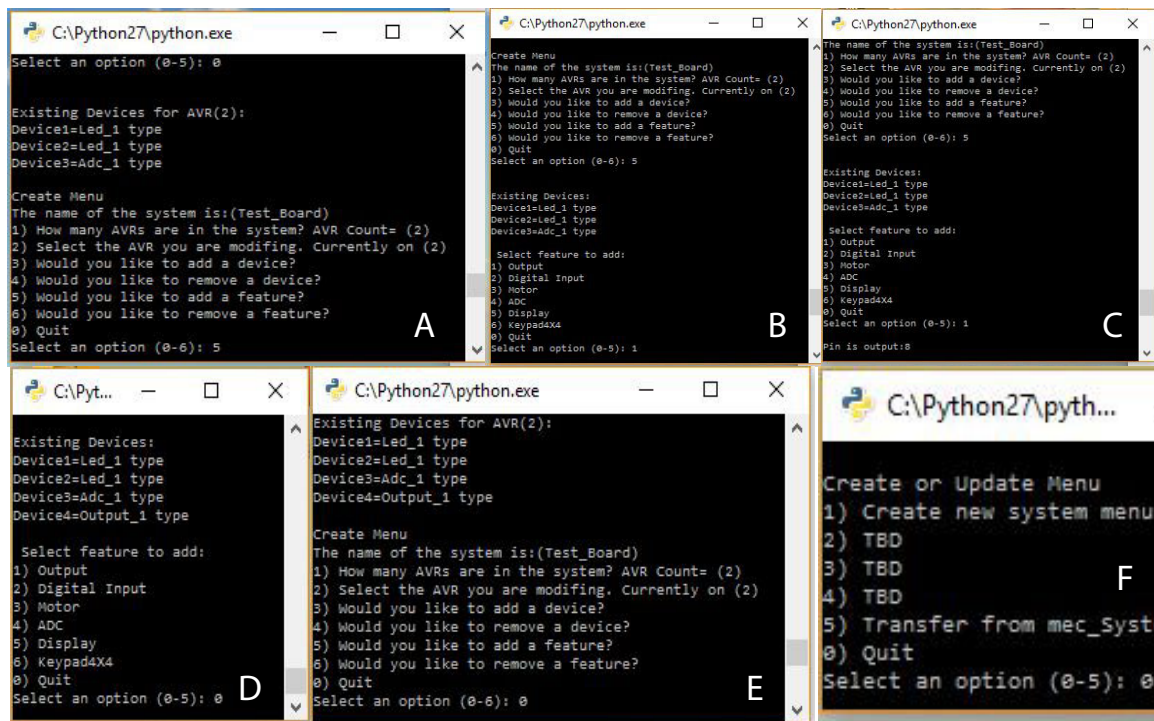


Figure 17.6: Set of menu options for selecting features for AVR 2. Option 5 is used for selecting features (A). The first feature we select is an “Output” (B). We put the Output on pin 8 (C). We are done adding features so we quit the menu (D). We are done adding devices and features to both AVR’s so we quit this menu (E). We are done creating new systems, so we select one final quit (F).

An AVR has several features and we are still coding many of them. The menu options for features shown in figure 17.7 are not yet all functional.

17.3.5 Creating an instance of software for the Test Board

We now go into the `mec_Systems` folder and find the `Test_Board` folder. This is the instance of the `Test_Board` software. Again, the user does not need to worry if they modifies this code; it does not affect the main code. The folder contains four sub-folders: `Arduino`, `AVR`, `Objects` and `Python`. The `Arduino` folder contains the Arduino code, the `AVR` folder contains the AVR code, and the `Python` folder contains the MEC scripts for

generating the Python code. The objects folder contains information that the user inputs through the menu options, such as what pin the devices are on, or the number of AVR's in the system. In the future, much of this information will be moved over to the configuration file, allowing the user to see and modify it as needed. Also, having the information in the configuration file will make working from an existing system more straightforward.

Programming the ATtiny84 microcontrollers

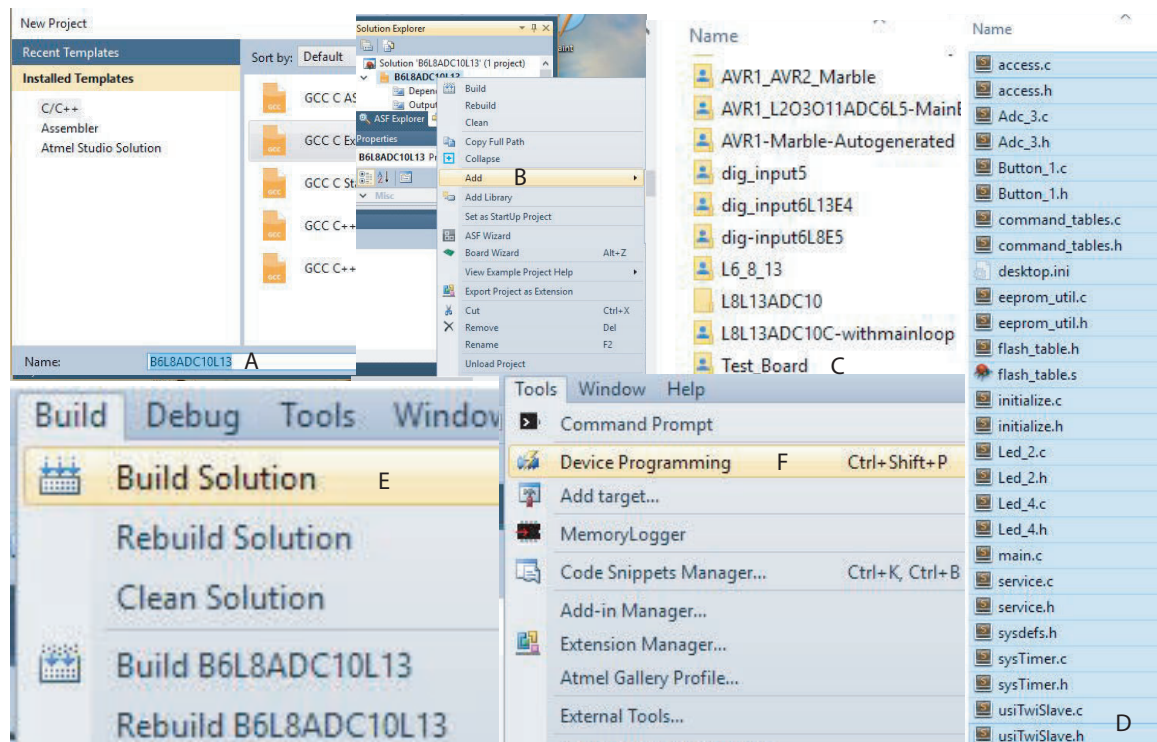


Figure 17.7: Key tabs, windows, and entry boxes used when programming ATtiny84 microcontroller with Atmel Studio. The user enters the project name and selects “GCC C Executable Project” (A). This is the window used for adding preexisting code to project (B). Automatically-generated code is in the “Test Board” folder under the “mec.System” folder. Auto generated code for “Test Board” AVR1 (D). The window for building one’s solution or compiling one’s code (E). The window for “Device Programming” (F).

Now that the code has been automatically generated by the MEC Create software, we are ready to program our two ATtiny84 microcontrollers. I will walk the reader through the steps shown in figure 17.7. If one has not already downloaded Atmel studio onto their laptop or computer, download it, or find a computer that Atmel Studio is on. Open the Atmel Studio software. Under “File” open a new project. Enter the name of the project and select “GCC C Executable Project” (A). Next, select ATtiny84 or ATtiny84A device, starting by entering the name. Then, the device should come up and one selects it. Remove the Atmel-Studio-generated main on right side of the screen. It has the name of the project, with “.c” at the end. We will be auto generating a new main file. Right click on the project folder on the right side of the screen and select Add (B). Browse to Test Board folder under “mec_System” folder (C). Go into the code folder under the AVR folder and select all the auto-generated code (D). After selecting, click Add to add this code into the project folder. On top tap go to build, then click Build Solution (E). Looking at the bottom one will see a “Build Succeeded” message if it was a successful build. Use a printer cable to plug the USB from one’s computer into the AVRISP mkII programmer. The other side of the programmer is attached to a small custom adapter card which has a fourteen-pin socket. Plug the ATtiny84 microcontroller into this socket. I hand wired and soldered one of these adapter cards and then a student designed up a printed circuit card from this design, allowing for more of my students to have a programmer. Next, one will program the ATtiny84 device. Go to the top taps, and select Tools, and then Device Programming (F). A “AVRISP mkII Device Programming” window will appear. Press apply. Under fuses make sure Div/8 is unchecked and then program fuses in that configuration. The user

only has to do this the first time an ATtiny84 is used. Afterward, the chip will remember. Finally, program the memory. One should get a verification OK after programming. In this manner, the user has programmed the first AVR. This process is repeated for the second AVR.

Running the MEC created software to operate the Test Board

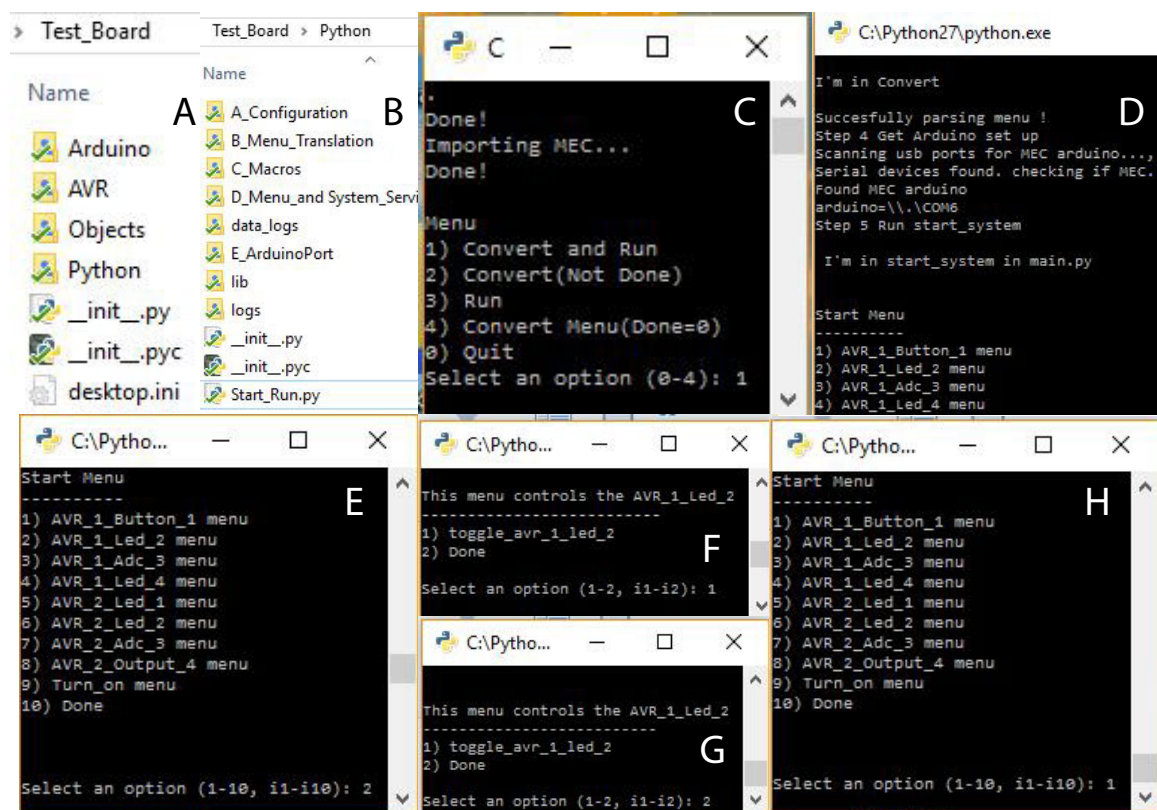


Figure 17.8: Code folders for Test_ Board (A). Code and script folders under Python, also including the Start_Run.py file, which is the Test Board instance start program (B). Command line starting menu, which contains Convert and Run which are used to convert and run auto-generated script and code (C). Printout that lets the user know if Python code was able to find MEC Arduino on USB port (D) . User defined Start Menu (E). Menu for toggling LED_2. Entry 1 toggles LED (F). Menu for toggling LED 2, entry 2 exits menu (G). Entry 1 in Start menu selects Button.1 test menu.

Here we briefly discuss an instance of “Test_Board.” The code and folders for this instance were auto-generated by the MEC Create software. The top level folder under Test_Board folder are: Arduino, AVR, Objects, and Python 17.8(A). Under Arduino folder is the Arduino code that is required in the Arduino microcontroller on the Test Board. This code on the Arduino is used to interface between the laptop and the ATtiny84 microcontrollers or any I2C device. The Arduino code used is currently the same for each system. Under the AVR folder is the auto-generated code for each of the microcontrollers, typically an ATtiny84. The microcontroller code is under the folders labeled AVR1 and AVR2. Under the Objects folder one has device and system objects that are used to store user input information. Under the Python folder is all folders that contain user definable script and the Start_Run.py. Start_Run.py is the start program for the Test Board instance. The user-definable scripts are contained in: Configuration, Menu_Translation, Macros, and Menu_and System_Service. The last folder am going to rename User_Main (B).

Running the Test board instance

To start the MEC instance of Test_Board code one double-clicks Start_Run.py. The Menu that comes up has five options: Convert and Run, Convert, Run, Convert Menu, and Quit. Convert and Run converts the user definable script contained in the four folders (Fig. 17.8 (B)) into Python code, objects, and variables. It then runs this newly generated Python code. The Convert option does same thing but does not run, and the Run option runs Python code assuming it has been converted. The user only needs to Convert the first time, then the user can just Run. The Convert Menu option will convert the four script types separately. This option is used when one is not auto-generating script and

wants to check if there are any errors in the user-created script when converted. For this example we will assume we use option 1 (Convert and Run). The software goes through three conversions (Configuration, Menu Translation, and Macros). After conversions, it searches the USB port for an Arduino that has the MEC Arduino software properly loaded (D). If an Arduino is found, the Start Menu appears. The Start Menu contains a sub-menu for each device (E). Here, we have selected option 2 to test the LED. It is labeled “AVR.1.Led.2” The “1” tells us that this device is in AVR 1. The “2” tells us it was the second device loaded. Each device has an auto-generated unique ID. The LED test menu comes up. Notice the unique name at the top (F). We select option 2, toggling the LED state. If we repeat this option, we will toggle the LED on and off. Once finished, we select option 2 to exit the menu (G). This takes us back to the start menu. This time we select option 2 to test the button (H).

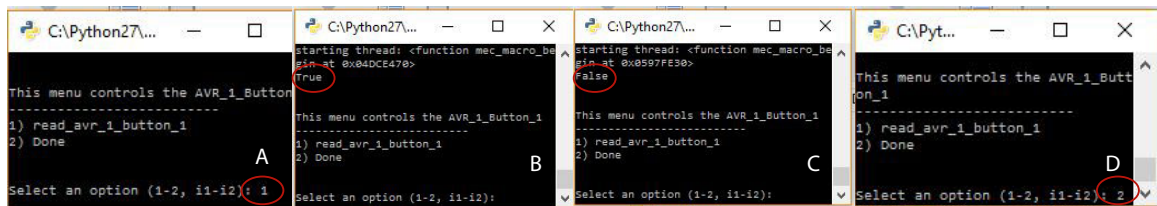


Figure 17.9: Selecting option 1 in button test menu (A). This option reads the state of the button. A True is returned if the button is not pushed (B). We now press on the button and select option 1 again. This time we return a False (C), indicating the button has been pushed. Once finished testing the button, we select option 2 to exit (D).

The LED verified we were able to send a digital signal out of the microcontroller, which can be used to control any number of devices. Next, we want to confirm that we can obtain a digital signal back. A simple button will allow us to test this. We wire the button so that it has a pull-up resistor on the line going to the microcontroller, going high when

open (True). When button is pushed, it connects the line to ground (False). From the main menu we had selected option 1 to test the button. The button test menu then comes up. We select option 1 which reads the button state (Fig. 17.9 (A)). Notice it returns a True (B), indicating the button is not pushed. We now push the button and repeat the test. This time we get a False (C). Once finished, we select option 2 to exit (D).

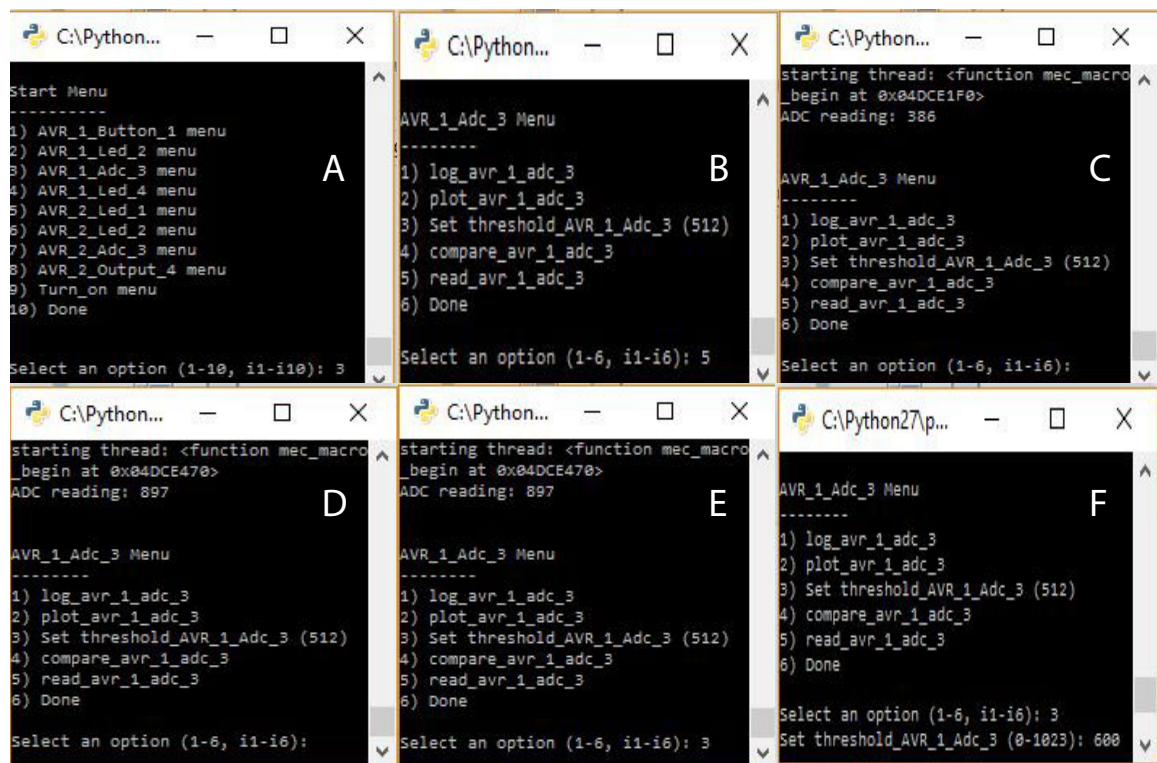


Figure 17.10: Selecting option 3 on start menu (A). This option is used to test the potentiometer on the test board. Select option 5 on ADC menu (B). This option is used to read the selected ADC pin on the microcontroller. Here, it is showing that it obtained an ADC reading of 386 (C). Turn the potentiometer and reread the ADC, we got 897 as a reading (D). Option 3 is used to change the threshold (E). Setting threshold to 600 (F).

After we are finished testing the button in AVR 1, we return back to the start menu. This time, we select option 3. This will take us to the analog-to-digital converter (ADC) pin test menu (Fig. 17.10 (A)). On our test board a potentiometer is used to generate

an analog signal from zero to five volts. The microcontroller has a 10-bit analog-to-digital converter, which give us a range of 0 to 1023. The analog-to-digital converter menu has several options. The first option turns on a text log file. The data read from the ADC is recorded in this file and time stamped. The second option is used to plot the above recorded data. The plot comes up live, so the data will continue to be recorded and plotted. The third option is used to set a threshold. The fourth option compares the analog signal from the pin to the current threshold. If higher than the threshold it sets the threshold state high. If lower than the threshold, it sets the threshold state low. This can be used for sensors such as the marble sensor to indicate the presence of a marble. The fifth option is used to read and return the value of the selected ADC pin. The sixth option is used to exit the menu. In this sequence we select option 5 (B), reading the selected ADC pin. The value read from the ADC pin is 386 (C). We change the potentiometer and do a second reading. This time we get an ADC reading of 897 (D). Next, we want to test out the threshold function. We start by selecting option 3, to change the threshold (E). We set the threshold to 600 (F).

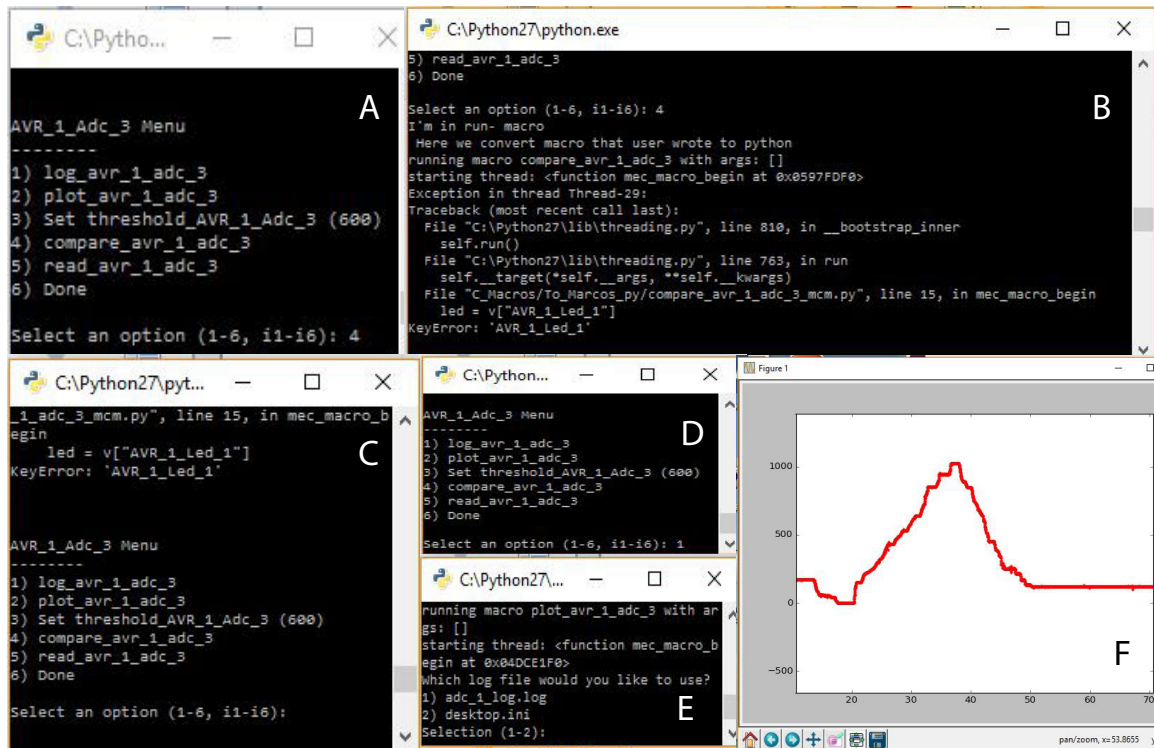


Figure 17.11: ADC menu with threshold set to 600. We select option 4 to test whether the analog signal is above or below the threshold (A). Error message indicating it cannot find the device macro specified (B). We select 1 to begin logging data to a text file (C). We then select 2 to start the plot. Several plots may have been taken so a menu will come up, where the user can select which plot he wants to display (D). Moving the potentiometer back and forth, one can shift the value from 0 to 1024. This information is time-stamped and stored in a log file (E). Plot of data from log file (F).

Observing the end of option 3, the user can see the threshold is now set to 600. We now select option 4. At this point we want to see if the ADC value being read is above or below the threshold (Fig. 17.11(A)). Unfortunately, we get an error. This is due to the Python translator, which cannot find the macro specified (B). After the error, we return back to the ADC menu. This time, we select option 1 to start the data logging. The data has a time stamp placed on it. When studying the data the user will see the time stamps are not consistent (E). This is because we are in a loop in the Python code asking for data.

When it receives the data, the PC places a time stamp with it and records the data to a text file. The Python software in the PC is running in several threads. A thread is a sequence of code that runs independently from another thread. The computer controls the transitions between threads and it is not consistent. Therefore, the time stamps are not consistent. Once the data logger is initiated, one can select option 2 to plot the data (D). The plot is a live plot. By moving the potentiometer up and down one can see the value change on the plot (F). The icons at the bottom of the plot allow the user to pan out and zoom in.

When undergrad students are given the opportunity to participate in real product development, as the MEC system provides, they learn countless lessons. Not the least of these lessons is persistence. Persistence is needed to overcome many engineering challenges, yet it is something difficult to teach in a class environment. Adam Veres did most of the Python code for the data logger shown here, evolving the code and his skills over time. When I look at what has been done, I am excited by what we have achieved and the learning that has taken place. However, I also see the challenges ahead and the great learning opportunities to come. Let me give some examples: A unique name is generated for each new device the user inputs. That name needs to be edited into the actual code. Default values are set, usually based on the instance in which the device was created, so sometimes auto-generating code works because the device is placed in the same spot it was when code was created. Therefore, the default value is correct and no editing is needed. When the device is moved, as it was here, the user will receive an error. We can go in and change the name and the code will work for this instance. That is the quick fix; what really

needs to be done is to fix the auto generating code. This can be a good teaching moment. A student can see the error and be asked to fix it. Or they can be walked through the process of fixing the problem. Both methods provide valuable lessons and often a combination are used. The user, or one of the user's experienced students, begin walking the student through the problem until they understand enough to proceed on their own. Another good example of a great learning opportunity is the inconsistent, slow, and somewhat inaccurate time stamps (Fig. 17.11 (F)). Here, one can discuss with the student the problem. In this case, I suggested two different solutions for two different students because their backgrounds were different. One student (Timothy Williams) was performing data analysis with MATLAB and was implementing a transform to the frequency domain. The transform assumed that the data had a consistent time stamp. Here, I suggested taking the average time stamp and then effectively sampling data at that rate. To obtain the approximate values at those sampling points, one would do a linear approximation from the actual sampled data. We overlapped the two graphs and at first found a transformation error. After the error was fixed, the two graphs matched. This MATLAB project continues to help Timothy evolve his skills. As the TEC Center grows I hope to have a set of experts in areas such as MATLAB, so, as students create solutions, we can compare their solutions with what I will call "best practices." A much better way of solving this problem is fixing the way the data is collected in the first place. Here, a student will need experience in C microcontroller coding. The solution I have discussed with my students is creating a buffer in the microcontroller. We can look at the time stamped data and observe the max time it takes between samples. As one of my students pointed out, this time may vary between different machines. From this

we can determine the size needed for the buffer. At the microcontroller level the data can be stored in the buffer at a constant rate. The upload rate, on average, needs to be the same or faster than the microcontroller rate. The buffer simply needs to handle the variations in the upload rate. If the buffer ever overflows, which it should not, it can set a flag to inform the user that the data has an error. If there is no error we will have a consistent rate at which the data is taken.

Another example of a learning opportunity came with the auto code generator. It allowed us to put together increasingly complex systems such as the marble machine and the Test Board. As multiple devices are combined together, the service routine on the microcontroller grows. The Arduino, through the I2C, sends commands to the slave microcontrollers. This might be a command that one wants to read data. The Arduino would then proceed to read the data. If the slave is in its service routine then it might not be ready to send the data. This creates an error and when I first plotted the data shown in figure 17.11 (F), I had just received a stubby red line before the data logger locked up. Adam had been working on this problem and had placed a time delay between the write and read on the Arduino. We discussed the problem. The slave microcontroller is supposed to hold the clock low on the I2C bus, when it is not ready to send back data. This is not happening. I like to say to my students that “this problem is above my pay grade,” which simply means the student’s skill level in this area is higher than mine and they can solve this problem faster than me. I think this gives students like Adam great satisfaction and motivation, knowing they are working on problems that are above my pay grade.

Chapter 18

Intern Program

I started the intern program about three years ago. This has turned into what I consider a “best practice education model” on how to implement STEM education, utilizing the MEC system. I did not set out to create a model of how to optimally build skills; it simply came from my need to have better trained interns. It is a project-based or need-based learning model. The MEC system adds to the standard model by letting undergrad students attack higher-level real-world problems than was previously possible, with a special focus on biological problems. The TEC Center utilizes students from all the engineering disciplines and recently has also included business students. The real testament to the program comes from my students. Every day that I work with the students I could write another story showing the MEC educational model working. The testaments of the students can be found on the Center’s website and I have included some of them here. I hope they make the reader feel as good about the success of the program as they make me feel. I also included my paragraph from the website.

18.1 Douglas Hill



Figure 18.1: Douglas Hill

Project Leader, TEC Center. My name is Douglas Hill, and I manage the TEC Center. My mission is to “Save the world one step component at a time,” and in the process train the next generation of scientists and engineers. The TEC Center is a multi-disciplinary learn-by-doing environment where we tackle real world problems. We do this first through the development of MEC components, then by combining these components for the rapid develop of instruments (see projects). I graduated from the University of California, Irvine with a B.S. in Electrical Engineering, in June 1979. My first job was with Texas Instruments, where I was given the honor of being the only junior engineer awarded

“key contributor.” I started my own system design house, Advanced Electronic Packaging Corp., solving electronic packaging problems for major corporations across the United States for over 15 years. I’m currently a Bioengineering graduate student at the University of California, Riverside. I came back to school to implement methods I had developed in electronics, to do early disease detection. Unlike electronics, I found bioengineering lacked a standardized tool set from which to work from. So the MEC system was born. The development of this multidisciplinary tool set is a daunting task, so I brought in interns from all areas of engineering, and the TEC Center was born. The students lacked many of the skills necessary to attack these real world problems, so training became an integral part of our program. The “E” stands for Evolutionary, and the Center, its people and tools keep evolving in amazing ways. I invite you to come to the center, see for yourself, and possible share in our adventure.

18.2 Adam Veres



Figure 18.2: Adam Veres

Undergraduate Researcher, TEC Center. My name is Adam, and I am a fourth year Computer Science Major at University of California, Riverside. My work in the lab is to develop software to create and control MEC devices. This entails programming and/or debugging AVR for major MEC components, programming python interface for MEC components, and teaching others how to use this software. As such, I have created appropriate tutorials for programming these different devices. During the 2016-2017 school year, I would like to assist in integrating GUI with MEC software, develop AVR components for outreach or TEC as needed, and assemble computers for TEC.

18.3 Chandler Morehouse



Figure 18.3: Chandler Morehouse

Undergraduate Researcher, TEC Center. My name is Chandler Morehouse, and I am an undergraduate at University of California, Riverside. My objective for this program is to be able to properly implement a three stage communication system between a personal computer, Arduino Uno, and various ATtiny84 microcontrollers. The project will consist of writing code that will use I2C interfacing to communicate between all devices and to design a system of devices that control mechanical systems and multiple sensors. The purpose of this system is to be able to write high level programs in Python which is translated through the Arduino into I2C serial data protocol. This data then communicates with address specific AVR controllers to execute specific commands created in the original Python code.

My objective for participating in this project is to further my knowledge in the use of AVR microcontrollers and programming. I wish to learn how to use I2C communication and to get a better understanding on how a system of AVR controllers work together to accomplish particular goal. From this project I would also like to get more experience with working with a team of engineers and more specifically learn how working with engineers who are not from the same field of engineering.

18.4 Hashim Chaudhry



Figure 18.4: Hashim Chaudhry

Undergraduate Researcher, TEC Center. My name is Hashim Chaudhry, and I am an Undergraduate in the University of California, Riverside. My main objective in this lab is to create an image processing system capable of distinguishing the number of dead and alive cells from a given sample using a camera-and-lens system with image recognition software (most likely using Python OPEN-GL). Visio was used to draft up a camera and pump system schematic. OPEN-GL allows for direct programming for software. Solidworks was utilized to create various parts such as the pump, test tube holders, and motor mounts.

18.5 Hayden Karich



Figure 18.5: Hayden Karich

Undergraduate Researcher, TEC Center. My name is Hayden, and I am a third year Mechanical Engineering major at University of California, Riverside. As a mechanical

engineer, I want to develop my design abilities and learn proper machinery operation safety in order to fabricate my designs. I hope to gain a professional level of ability with SolidWorks and gain a better understanding of programming. I would also learn to use as many different fabrication machines as I can. I am designing a material transport system using pressure and vacuums. Since entering with minimal experience, my first task will be to transport a marble with the pressure system so I can develop the skills I need to pursue a larger scale project. Eventually during the 2016-2017 school year, I would translate the same method of transportation to horizontally transport samples for experiments into different stages of the experiment like in/out of light or in/out of different temperatures.

18.6 Krishna Addanki



Figure 18.6: Krishna Addanki

Undergraduate Researcher, TEC Center. My name is Krishna Addanki, and I am a fourth year Electrical Engineering major at University of California, Riverside. I am interested in working on creating a basic system that allows for hardware integration with the main computer or master device. Currently, I am working on a software I2C to integrate an external I2C device. Utilizing an ATtiny84 to control an accelerometer, the lab would be able to use a customized I2C interface for most of the lab devices. Microcontrollers connected to this interface allows for pre-made I2C device additions when needed. During

the 2016-2017 school year, I would like to be able to finish working on software I2C creation as well as begin on a system etching in-house PCB through either manual etching (using Eagle and printed PCB) or computer-guided etching (using laser printer).

18.7 Manoel Tamraz



Figure 18.7: Manoel Tamraz

Undergraduate Researcher, TEC Center. My name is Manoel, and I am a second year Bioengineering major at University of California, Riverside. I joined this lab as wanted to know if I can meet expectations in a working environment and if I can exceed those expectations through technical and working skills learned from lab. I am interested to find out if I can handle responsibilities placed upon me. Furthermore, I want to have more valuable experience that I can show on a resume which will help me in the future as I search

for jobs within my field. My project objective in lab is to create a working track system and all related components, create a lift system to keep the system running in a loop, and develop a system whereby I can use black and white balls to send a binary message down the track and trigger a response. I will meet these objectives by learning SolidWorks, Visio, and any other required skills. Furthermore In my lab work during the 2016-2017 academic year, I will utilize this base knowledge to streamline the solenoids used for the bioreactor which previously held a 95% success rate.

18.8 Matthew Smith



Figure 18.8: Matthew Smith

Undergraduate Researcher, TEC Center. My name is Matthew Smith, and I am a third year Mechanical Engineering major in University of California, Riverside. My goal in

this lab is to further increase my understanding and ability in Solidworks. Furthermore, I would like to learn how to 3D print and be able to design a product fully through SolidWorks into the 3D product. My work in the lab involves developing a flipping mechanism involving muscle wire to swap the marble ball between the two connected tracks.

18.9 Raymond Iu



Figure 18.9: Raymond Iu

Undergraduate Researcher, TEC Center. My name is Raymond, and I am a third year Bioengineering major at the University of California, Riverside. My interest focus currently leans toward quality control and sales. As such, my work in the lab has been towards building a quality control standards as well as grant writing for the lab. Continual build-up of proficiency in SolidWorks, Visio, C++, technical writing, and 3D printing has

allowed me a solid technical foundation. In my lab work during the 2016-2017 academic year, I will utilize this base knowledge to write a quality control process for 3D printed components within the lab. I will then test component performance using C++ to gather data from part usage. Finally, I will analyze the data to provoke improvements to use Solidworks in creating better parts. These parts will set a lab standard of consistency to continue top level research in the lab.

18.10 Raymond Wan



Figure 18.10: Raymond Wan

Undergraduate Researcher, TEC Center. My name is Raymond Wan, and I am a third year Bioengineering major at the University of California, Riverside. My first project in this lab is the automation of a marble machine. Furthermore, I am helping to set up the inventory system in lab to organize the 3D printed parts manufactured in the TEC center. As my interests lie in prototyping, programming, microcontrollers, and fabrication, these projects will help me to further develop these skills. During the 2016-2017 school year, I plan to build a new system from the ground up to learn how the process works.

18.11 Razn Abu Qamar



Figure 18.11: Razn Abu Qamar

Undergraduate Researcher, TEC Center. It was Hugh Herr that proclaimed “A person can never be broken. Our built environment, our technologies, are broken.” My goal

is to dedicate my work to ensuring people have the technology to overcome any disability or disease, to always have the option to live whatever life they choose. In this pursuit, I have joined the TEC lab. This lab has already garnered a multitude of skills. Amongst the most prominent of which has been experience with SolidWorks, AVR coding, and systems design/analysis. I hope to further learn skills in management and leadership, as well as other practical skills through the experience I gained here. I work on the heating component of the bioreactor, which regulates heating of fluids and cells whilst analyzing and moderating flow. The bioreactor regulates the growth of cells within its system whilst boasting modularity and cost efficiency. My work has taught me skills in calibration, hardware design, and design intuition. I have created modular parts that are incorporated throughout the bioreactor allowing for flexibility in design and construction. Furthermore, my work around lab circulates towards training new recruits in lab procedures, SolidWorks, and system design. The work I conduct at the lab has been invaluable. This work has helped propel my education and my knowledge as an engineer, teaching me important skills that will come to use throughout my lifetime. My name is Razn Abu Qamar, I am a senior bioengineer, and I push to save the world one small component at a time.

18.12 Ryan Milton



Figure 18.12: Ryan Milton

Undergraduate Researcher, TEC Center. My name is Ryan, and I am a 3rd Year Bioengineering major. I want to learn how to utilize Bioengineering design software in order to learn how to design and build complex systems from sets of components. I began with a solidworks project in the lab. Currently, I am working on Eagle PCB design to create an Eagle board to gain knowledge of circuitry and schematics.

18.13 Stefany Cruz



Figure 18.13: Stefany Cruz

Undergraduate Researcher, TEC Center. My name is Stefany, and I am a third year Electrical Engineering major at the University of California, Riverside. My interests lie within the realms of medical robotics, bio-mechatronics, and creating surgical/medical devices that would help enhance the human experience. I know the basics of electronics from what I have learned in Electricity and Magnetism physics class, Quantum Physics class, and an Intro to Electronics class. But, I desire to learn more about electronics and the computer programming aspects that it takes to build such cyber physical systems such as MEC and other related projects. I would especially like to gain hands-on experience in the lab with microcontrollers, programming languages, and other forms of circuitry. I want to learn to not only read and follow the schematics of electrical components, but design them

myself as well. With this gained knowledge, I believe that I would be able to apply them to future projects and integrate them with other subfields of medical robotics. Ultimately, I see this TEC Center opportunity as one of the first stepping stones that I need to reach my goals.

18.14 Timothy Williams



Figure 18.14: Timothy Williams

Undergraduate Researcher, TEC Center. My name is Timothy Williams and I am an undergraduate in the University of California, Riverside. In this lab, I want to learn as much as I can from everyone. I am particularly interested in improving my knowledge of Fluid Dynamics, 3D modeling, and Electrical Engineering. Currently, I am developing

an inexpensive, reproducible weight sensor. My work thus far has consisted of Solidworks design to create the motor holder, bottle holder, and the baseplate of the device.

18.15 Vishak Kumar



Figure 18.15: Vishak Kumar

Undergraduate Researcher, TEC Center. My name is Vishak Kumar, and I am an undergraduate at University of California, Riverside. The reason for me joining the Outreach program at the TEC Center was because I wanted to gain insight into the real life scenarios of engineers and the jobs they do. I also wanted to understand exactly what is required of a person, both mentally and physically, in order to successfully contribute to and work in a field such as engineering. Beyond this, I definitely wanted a personal gain in knowledge, which can only be done by acquiring new experiences and by being challenged.

The TEC Center seems to me like a perfect opportunity to do just this. In regards to the lab itself, I want to do research that is worth something, be it now or in the future. The TEC Center stands for the evolution of technology, which is exactly what I intend to research and develop during my time in the lab. On top of evolving technologies, I also aim to create new technological components as well, as this contributes to the growth of the field that is technologies of engineering. My work in the lab so far has been creating Visio schematics for the vacuum elevator.

18.16 Zareh Hairapetian



Figure 18.16: Zareh Hairapetian

Undergraduate Researcher, TEC Center. My name is Zareh Hairapetian, and I am a third year material science engineering major in the University of California, Riverside. I hope to acquire the basic skills necessary in starting off in the engineering career path such as the proper methods of making schematics, Solidworks, and 3D printer operation. I also want to understand learning in a work/research setting rather than a classroom and to incorporate said basic skills and proper methods to reach a final product. I hope to get out of this program a wide variety of research experiences in different engineering disciplines, a

mentor figure that will assist me in finding the correct career path for myself, friends and colleagues with similar aspirations, and research lab experience. Currently in the lab, I am working with a fellow undergraduate on a standard 3D part catalog while designing a part for a pump system.

Chapter 19

Intern Program Alumni

Several of our interns have graduated and gone on to jobs, graduate school, and medical school. Statements from a few of our alumni are included in this chapter.

19.1 Afshin Mostaghim

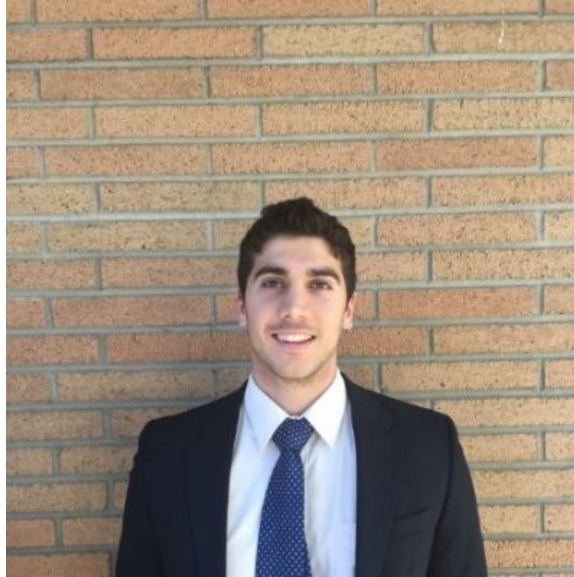


Figure 19.1: Afshin Mostaghim

My name is Afshin Mostaghim and I currently work as a Quality Engineer in one of the largest medical device manufacturers, Boston Scientific. I graduated from UCR in 2015 with a background in Bioengineering. I was given the opportunity to join the team and be one of the individual contributors for the Bioreactor project. This was truly an eye-opening and humbling experience to be able to see a system come to life from scratch. Each component and every bolt in this system worked hand in hand in order to make it functional. This is what I learned most while being on the team with Doug: Teamwork is key. I was fortunate enough to be apart of an enthusiastic and exciting team that worked hand in hand to get the job done. We even got to publish a paper out of our hard work! In addition to teamwork, I learned how to use a 3D printer, Solidworks, and various machine

shop skills. These skills have given me an extra edge in my resume and allowed me to be a competitive candidate in the job market. This opportunity has opened many doors for me including an internship I had with a start up medical device company and then numerous job interviews. I could not thank Doug and Dr. Grover enough for the opportunity, I will always be thankful to be apart of their team.

19.2 Casey Hill



Figure 19.2: Casey Hill

My name is Casey Hill and I currently work at an all-in-one business automation company in Santa Barbara called ONTRAPORT. To give a full scope of the impact of working with Doug and the lab in one paragraph will be challenging but let me start

here. Ideas are cheap. Execution is the impetus that drives real and sustaining change. Working with Doug was about bridging that gap between ideation of paradigm shifting technologies and implementation of them. When I worked in the lab, it was in a multifaceted capacity. I did grant writing, design work, outreach, recruiting and a range of other tasks as needed. My passion lies in creating innovative ideas that have the capacity to create real world change and working with the TEC program I could see without a shadow of a doubt that what we were building had that potential. There were many hard skills I learned in the lab. Some of the skills I learned include Solidworks design, Adobe Illustrator, G Codes for CNC, navigating around grant platforms like Pivot, etc.. But what was most impactful was iterating and working in a dynamic atmosphere where my opinions were heard and integrated into our designs and approaches. I have no doubt that the work that was compiled during my time there, and continues to be developed, will turn into transformative technology. The mark of a truly great project or company is one that can answer the question, "How is this going to change the world?" This does just that.

19.3 Nels “Chip” Pearson



Figure 19.3: Nels “Chip” Pearson

My name is Nels “Chip” Pearson. I have always been involved in technology with a focus on small scale embedded microcontroller systems, sensors, and actuators. I enjoy combining hardware and software designs for the best results. From March 2015 to October 2015, I was primarily involved in implementing Doug’s concepts of an I2C network of components that could support multiple devices within them. The architecture would also support auto-generated code processes to simplify development of new components. I also provided suggestions for the design of the website. Working in the lab gave me the incentive to advance local network concepts that I had been investigating for use in small scale robot platforms development. I learned a little of what Bioengineering was all about

and the current state of some bio-tech investigations. I have been retired since 2013 but I still build small robots, work on various embedded microcontroller projects, and design websites.

19.4 Daniel Chang



Figure 19.4: Daniel Chang

My name is Daniel Chang, and I graduated from the University of California, Riverside in 2016 with a Bachelors degree in Bioengineering. I joined this team with no experience, eager to learn anything and everything I could. Starting with Microsoft Visio, I would draw schematics of solenoids and circuit boards. Schematics of circuit boards would eventually be translated to printed circuit boards with the help of EAGLE. The design process allowed me to see the details that went into electronics. Eventually I was introduced to SOLIDWORKS in order to design 3D models for the future MEC system.

Being able to actually print my designs and use them as part of a larger system provided me with a sense of accomplishment. Before joining, I had zero ideas for what I wanted to do in the future. But working at the TEC Center allowed me to find my passion in CAD design and pursue a career in designing components and medical devices. I have no doubt this work will end up making the world a better place!

19.5 Humberto Scott Sanchez



Figure 19.5: Humberto Scott Sanchez

Hello there! My name is Humberto Scott Sanchez, I graduated with a BS in Chemical Engineering in the Spring of 2015. Before I started my time in the laboratory, I was interested in the field of bio-process engineering. My past research experiences were spent studying and engineering protein-DNA nano-structures, refining cell transformation

processes, and optimizing the separation of cellular components from culture media. My Senior Design Project was based on the incorporation of a Microbial Fuel Cell into Organic Wastewater Treatment. All of these experiences led me to helping Doug Hill with his analytical bioreactor, I was only able to assist the lab for a short time, but that time helped me a lot. In that project I helped redesign the heating unit and temperature control system, I designed a testing rig for that system, and fabricated prototypes of those components. Along with the hands-on work I developed many design equations for those systems. I learned a lot about how I can translate my background of chemical engineering into other fields of engineering. I learned how to solder, design milli-fluidic systems, and handle 3D printed components. Most importantly I gained confidence that I can convey most physical phenomena to others using design equations. The lab was a big inspiration for me to pursue instrumentation and microfluidic research for when I go back to get my doctorate. I am currently a support technician for the Amgen Pilot Plant in Thousand Oaks, CA. The lab has helped me be a more well rounded engineer who is able to communicate with others in different fields of engineering. I plan to go back to back to graduate school in the future and implement what I learned in the lab.

19.6 Ivan Soleto



Figure 19.6: Ivan Soleto

My name is Ivan Soleto, and I am a Bioengineering alumni of University of California, Riverside. My Objective to participate in this program, was to learn and understand better design techniques related to 3D Printing, such as the use of CAD softwares, as well as evaluating the printed pieces and printers themselves. I also learned the overall process of multi-disciplined involved projects, and how related fields come together to see the big

picture of the project. Such projects involved were the creation of bioreactors, lab on a chip projects, etc. Thus, my participation in lab was 3D printer evaluation, printed piece testing (especially with cell filters and quality control), and bioreactor design with the inline filter. These skills built upon one another and culminated in my mastery of ImageJ, AMScope, SolidWorks, and Excel. These skills will help me greatly to pursue my 3D printed prosthetic designing aspirations.

19.7 Kevin Pham

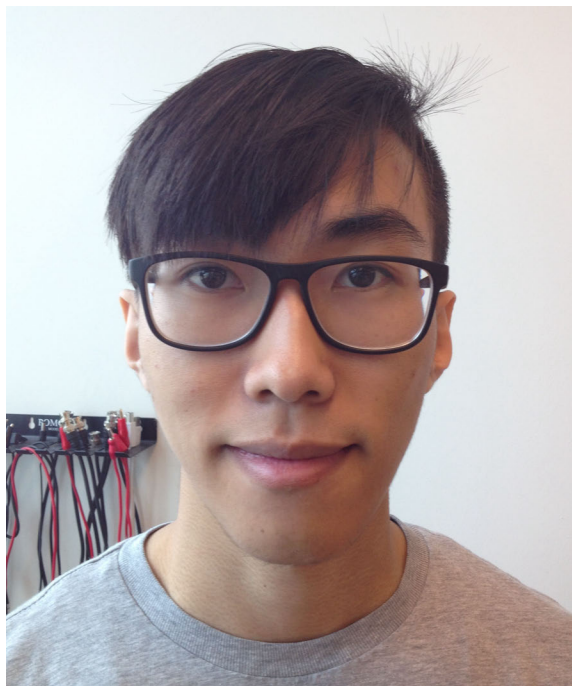


Figure 19.7: Kevin Pham

My name is Kevin Pham, and I graduated from University of California, Riverside with a Masters degree in Bioengineering. I joined this lab hoping to learn more about

sensors and automation, and so my work in MEC was tailored for these goals. I started with circuit board layout to create boards that would link our MEC systems to a micro-controller for automation. Designing and creating my own circuit boards provided insight into how electronics are assembled and operated. Eventually, I began building 3D models to prototype, test, and improve new 3D printed components for the MEC systems, parts including t-joints, valves, solenoids, optical density sensors, edge detectors, and many more. This lab has taught me how to solve problems through engineering design. Working in a team environment also promotes communication skills and collaboration. We each have our own tasks, but it is important we understand one another's plans and needs to create a functioning system. It has provided a hands-on experience and valuable skills that can carry over into industry.

19.8 Norman Li

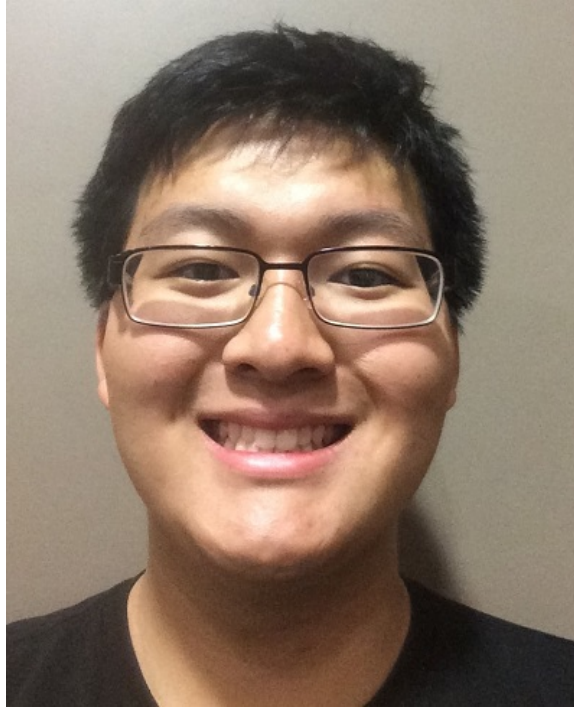


Figure 19.8: Norman Li

My name is Norman Li, and I have worked on the MEC project for two summers. I worked on mostly mechanical aspects of the project, as well as refining fabrication techniques and aiding in the design process. My first summer was mostly spent creating the Coulter system through board design and testing. My second summer was spent aiding in Solidworks design of parts, designing the heater part and helping to refine parts of other interns. I also helped to manage other interns and fabricate the parts and electronics required for the Solidworks parts. Another important aspect of my involvement was testing the solenoids. I helped to test and build many early designs of solenoids and helped test new standardized

components. During my time as an intern I learned important fabrication skills including: soldering, using a jig saw, using a drill press, and printing parts on two different 3D printers. The rapid design and revision process allowed by the 3D printers was key in putting together these systems, and knowing that these many of these parts were easily created in a few steps and couple hours made logistics more simple. I feel that my time working as an intern with Doug helped me learn marketable skills and begin to understand the design process integral to a large project such as MEC.

19.9 Ryota Saito



Figure 19.9: Ryota Saito

My name is Ryota Saito, and I am a recent graduate (Class of 2016) of University of California, Riverside. I began by assisting fellow undergraduates in the lab with

programming. Later on, my work in the lab has been with an Arduino and an ATiny84 mounted onto a I2C bus. This meant interpreting the AVR spec sheet, translating to code, then working on servo control for pulse-width modulation. After weeks of studying and learning this subject, I was able to successfully teach another undergraduate my work in order to carry on the project. This allowed me to solidify my I2C bus knowledge for the future.

Chapter 20

The TEC Center

Makerspaces are becoming a vital part of STEM education. A Makerspace should be more than just a space where students play with 3D printers. A Makerspace can be a place where students engage in real world problem solving. For that to happen we have to acknowledge the interdisciplinary nature of real world problems in the 21st century, providing students with the tools, skills and materials necessary to attack these problems. This is precisely what I propose doing, and in many respects what I'm currently doing at the TEC Center at UC Riverside. In this chapter, I present some of the important characteristics that help form the TEC Center.

20.1 The problems the TEC Center is addressing, and the partners that help define them

My students are taught to start at the top, the problem, and then to break that problem down into what is needed to solve it. The partnerships we form at the TEC

Center will partially determine the problems we focus on. For example, if we form a strong partnership with the School of Medicine at UCR, then the doctors will be an important part in determining the problems we face. If we partner with industry, then they should help define the problems. We are beginning to form partnerships with other labs at UCR and they have collaborated with us in defining the problems that they are facing. There are many wet labs at UCR which are hampered by the extreme labor intensity of their processes. The MEC system could automate these processes, allowing the students to focus on the problem, not the labor, of the process. We could serve as a resource at UCR for the automation of processes and for the storage of the reproducible results, which our equipment will provide.

Across the UC campuses there are some amazing technologies which are evolving, such as CRISPR gene editing. The TEC Center could be a vehicle for the translation of those technologies to UCR and other campuses. I believe that various types of cancer have been cured in research labs across the United States, but researchers still need to test and refine these existing methods. By partnering with the leading universities on these cutting-edge methods, the TEC Center can help automate those existing processes, allowing for the rapid transfer and refinement of techniques that are needed for their transition into clinical settings. I've also begun a partnership with Floating Doctors, and they are helping us understand the problems that the rural communities in Panama are facing.

Working with practitioners is an important step in making sure the problems we tackle are relevant. In whatever problem we chose, we need to form a timeline for reaching Patient One (applied research benefiting the patient, etc). Preference should be given to

problems with a shorter timeline and largest impact. As schools work to solve real-world problems they need to be linked to practitioners, individuals who are the end users of the technology being developed. Partnerships formed because of a specific grant can be weak and temporary. I seek relationships that are mutually beneficial, with or without external funding. Throughout this discussion it is critical to keep in mind that the TEC Center students are our first partners. Whatever challenges we choose will optimally be problems which they are also eager to solve, since it is their energy which fuels the TEC Center. Some students come to the TEC Center with their own problems which they wish to address by utilizing their engineering skills, and those issues need to be considered. For me, the problem that I came to UCR to address was early disease detection through body fluid profiling. I quickly found bioengineering lacked a tool set from which to work from so I began developing the MEC system, but early disease detection is still important to me.

20.2 The Multifluidic Evolutionary Component (MEC) system is uniquely qualified to address biological problems.

The MEC system is a multidisciplinary component set that was developed for biological instrument advancement. What sets the TEC Center apart from other Makerspaces is the MEC tool set. The MEC components are structured in a way that allows them to be easily interconnected. Their multidisciplinary nature grants them the ability to perform a wide range of very different functions (fluidic, electronic, optical, and mechanical). MECs were developed to solve fluidic problems that can occur at multiple scales. Control and sensing is primarily through electrical and computer interfaces. The MEC system is backed

by a software/microcontroller system that allows the instrument to be controlled through a computer or simple electronic interface. Users can design and build their own unique instruments or select from an ever-growing library of existing symbolic interdisciplinary instrument designs. The need for this tool set is one of the first problems my interns have been addressing at the TEC center. Another equally important need that our interns have been addressing is the issue of better-skilled workers. From the start, many of our interns lacked the skills required to address the problems that they were being challenged with. Working on real world issues provided the motivation for them to evolve those skills.

20.3 Real world problems engage students and can create focus for a Makerspace

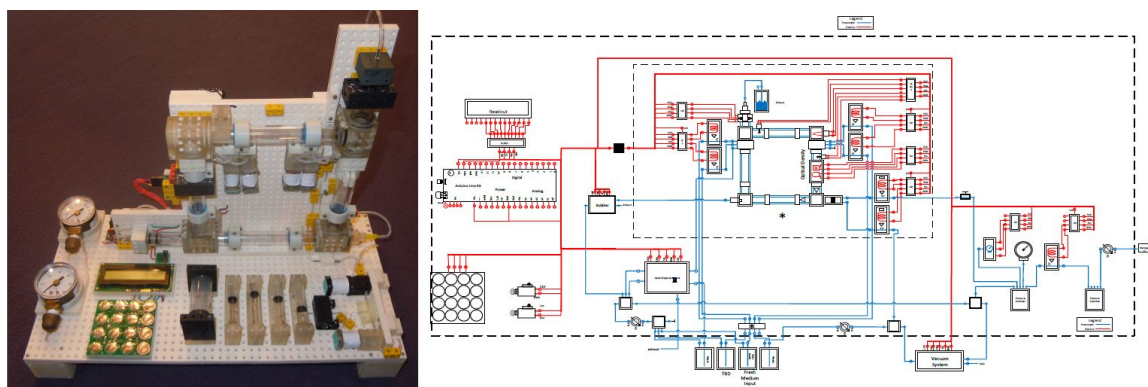


Figure 20.1: Analytical bioreactor and MEC schematic

The analytical bioreactor (Fig. 20.1) is a good example of a tool that we are developing which is well suited to address global problems such as cancer, hunger, and climate change. The MEC analytical bioreactor is a powerful starting platform. The bioreactor is

component-based and can take many forms. As students customize it to a particular application they will be learning valuable engineering skills. Once one fully engages students they will learn the skills necessary to accomplish their goals. This is accelerated in a team environment with students helping each other. The bioreactor, which has been discussed in detail earlier in this dissertation, is a good example of a problem that must be broken down in a team environment. Each element needs to become its own challenge for the student.

20.4 Grant funding

My biggest challenge is how slowly instruments and reliable components are developed at the TEC Center in comparison to industry. My instruments and components are being developed by undergrads, and their volunteer time is limited. Each student tries to maintain a five-hour commitment every week, but even that can be challenging. Classes have to take priority. However, if a small amount of funds are utilized to pay some interns it does accelerate the program, since they become more reliable and consistent. There are also repetitive jobs that need to be done, jobs that a volunteer may hesitate to do but a paid intern has no problem completing. Small amounts of funding used in this way also help pay students' tuition. I was also wondering what a small amount of funding for students in disadvantaged high schools would do. I would assume that it would have the same positive effects. One reason a student may not be able to participate in a summer program is that they need to work to help their family. Here, I believe that industry can also help, since industry would receive workers with more refined skill sets tailored to their needs. Additionally, the problems that are being addressed may be directed partially by industry.

Grant funding should be organized from the bottom up, so that the bulk of the funding goes to the work force at the bottom and for the equipment and materials that are necessary. Grant funding will play an integral part in the TEC Center growth.

20.5 Fully engaged students have no limit to what they can accomplish

In school, especially in disadvantaged areas, students often may have no idea why they are learning higher-level math or science. Projects that tackle real-world problems engage the students, since the students can see the reason for what they are doing. To accomplish these real-world goals, students will need to learn new skills, and that reason drives them to accomplish that learning. Real-world problems are so multidisciplinary that one can find areas that can engage almost any student. My interns are at the college level. I work to match a student's interests with the portion of the problem that will most engage them and require the skill building they desire. For high school students, multidisciplinary project based learning may be problematic in a regular classroom setting.

At the TEC Center we are developing a Makerspace model that shows how this can be done. Students can achieve and learn in their area of interest. They are free to learn and perform at whatever level they are capable of performing. This is possible because the problems they are tasked to solve can be resolved at different levels, or different degrees of completeness. For a large project there may be hundreds of parts that need to be developed. One student may only make one part, while another may make dozens. One may lead the integration of the parts and another may assist. One student may design a simple part and

another may teach himself how a gear is designed, utilizing complex math in the component construction. A student may only learn one skill, such as Solidworks, while another may learn SolidWorks, Visio, and printed circuit board design. Writing these examples is easy, for I can give examples of students for each of these cases. Whether it is complex or a simple problem, each of these students are helping the program move forward. Schools and universities need to be inclusive and not just for the gifted. Many labs only take what they perceive are the brightest interns and miss out on the gift of seeing someone grow. The only way the TEC Center survives this broad range of abilities is with students teaching students. The MEC system has the potential of accelerating what a student can accomplish and increasing their level of achievement, because the MEC system itself builds on previously-engineered MEC components.



Figure 20.2: Interns engaged in evolving the MEC system

20.6 Students' time is a key problem

An instructor's curriculum is often so defined that it leaves little time for creative problem solving. I was once told by a teacher who was attempting to integrate projects into his classroom that creative problem solving was not on his list of teaching requirements in the course, and therefore, little if any time could be allocated for it. I home-schooled one of my children in math, and with the workbook I was given, I too had little time to teach creative problem solving. A change is needed, and I think the new STEM education movement helps addresses this. By recognizing that real-world problems are multidisciplinary, time could be spent across multiple classroom periods, such as math, physics, biology, computer

science and English. We are developing a Makerspace model showing how this could be done. Cross-disciplinary teaching, although challenging for the teachers, can be quite an enriching experience for the students.

Having adequate time is one of the biggest challenges my interns face in solving problems. This is due to other class obligations that undergraduates have (school must come first). I hope to create a MEC Makerspace class at UCR for engineers of all disciplines; this would help with time allocation. There are two phases in a student's progression in the TEC Center. The first phase involves developing skills, and the second involves using those skills to help us address a problem. It is this second phase where I hope to utilize funding to allow for paid internships, which would also allow students to devote more of their time. Often, students need to find outside jobs to pay for the high cost of education. As one would expect, paid interns tend to be more consistent over time. The type of problems we are solving take time, and so the one consistent trait a student has who remains at the TEC Center is persistence. During the summer the students who make the most progress are the ones who spend their free time working on their MEC project.

20.7 Teacher training

Once we have established a working makerspace model developed at the TEC Center, teacher training will be a natural next step at the Center. A Makerspace instructor's primary focus should be to help students learn how to break down a problem and find the necessary resources to solve it themselves. Instructors are facilitators and guides. A large percentage of the resources needed by the students are cloud based. We are developing a

website to help lead students to the appropriate resource. Students will be going through a tutorial on how to use a specific piece of software or perform a process. My students (interns) at UC Riverside have learned software like SolidWorks and Visio, printed circuit board design, glass etching, CNC machining, lathe machining, Python programming, Arduino programming, C++ programming, web design, YouTube video production, 3D printing, drilling, sawing, and many more skills. They learn about important biochemical processes like ELISA, and the physics of how a motor works. Each student does not learn all these skills, but they learn the skills that are needed to get their portion of the job done, hopefully in their area of interest. Since I deal in real-world problems, I've needed my students to work in all of these areas, yet it is important to note that I still do not personally know how to do many of these skills, and I have been working for many years. To teach someone how to solve a problem, it is not necessary to have all the skills oneself. An instructor needs to be able to teach a student how to recognize what skills are necessary and how to teach themselves those skills. An instructor should be an expert in finding the necessary resources. Students will also need assistance understanding what is possible with the resources that are available. Often, a student lacks the experience to understand the scale of a problem. Here, though, it is also critical that the instructor not underestimate what a student can do. It is during the journey that the learning occurs.

20.8 Experts

In my vision of a Makerspace, I (the instructor) will be asking students to solve problems that I do not know how to solve directly. This is a scary prospect for anyone. To

aid in this problem, I want to develop a network of expert partners at UCR, in industry, and with other university professionals. Finding time will always be a challenge with these professionals. I believe that, through common interests and a range of communication methods, a powerful network can be developed. Industrial partners are not only experts in the skill sets we need, but also in the problem-solving process. I have also found that retired professionals can be a powerful resource. I may only spend a few minutes providing students with insights into solving a problem that they may spend the next day or week solving. I gain great satisfaction from this, and I believe our experts will find the same satisfaction. A small amount of time can make a huge difference.

As an instructor, the most valuable experts I have found are the students themselves: students teaching students. They have told me that they find helping other students rewarding. When I have a student challenged with a problem in MATLAB, I need an expert who can help show them the professional or standard way to approach the problem. This often should occur after the student has toiled and developed his/her own approach. For a student like Ramond Lu, who wants to be a QA engineer and is developing a process quality control document on how to fabricate a muscle valve, it would be optimal if he could have an actual QA engineer from industry look at the document he is producing. Whether it is a software program or a biological process, students need to be linked to people who can help them understand how to use a tool or run a process. These experts can be local or across the world. They can come from industry or academia.

20.9 The Multifluidic Evolutionary Component (MEC) Inventory system

We are working to establish both a physical and digital inventory of components. These components would be both MEC and from outside vendors. The physical inventory of MEC parts already contains hundreds of components. We link a part number and picture to a physical location. In addition to MEC components, we also have an inventory of electrical and mechanical parts from outside vendors. Many of these components and raw materials are used in the construction of the MEC components. In conjunction with the MEC inventory system we have developed a part numbering system. This part numbering system allows for components to be rapidly found in our digital library. The digital library or inventory maintains components in four forms. The first digital form is an image of the component. The name of the image is the part number. The first part of the part number is a unique ID. This ID is linked to a Solidworks file name (second digital form). The SolidWorks file was used for designing the part. The next digital form is an STL file, which can be generated from the SolidWorks file. The final digital form is Formlabs “.form” file which is created from the STL file and is used for 3D printing the part. This library accelerates a student’s ability to solve problems by allowing them to use pre-engineered components. As the MEC system evolves, so does the complexity of our components. This might include items such as a peristaltic pump or a density meter. This ever growing library of parts, beyond the student, is the most valuable resource the TEC Center has. In the future, we will be working to move this into an SQL database.

20.10 Students must have a basic set of tools and equipment with which to work

We have to strive to provide access to students for the tools and equipment they need to achieve their goals. The basic tools should be in the Makerspace. Additional tools can be located in surrounding schools, labs or businesses. The starting equipment we have at the TEC Center is a drill press, scroll saw, belt sander, small press, electrical multi-meter, oscilloscope, soldering iron, and a 3D printer. There is a large range of 3D printers on the market today. We are currently using a \$4,000 Form 2 printer from Formlabs (Cambridge, MA). For the money, this printer does a good job. Makerspaces today often expect more from a 3D printer than the printer is capable of producing. This is due to limitations in both material selection and the low tolerances 3D printers provide. 3D printed materials can be toxic to organisms [32]. They can often require secondary machining operations. Students should dream big, yet need to understand the limitations of their equipment. Additional equipment could include: a tabletop CNC machine, lathe, microscope with camera, laser scribe, and oven. At the TEC Center (our Makerspace) we are very cost sensitive, and therefore utilize the MEC system itself for developing a wide range of powerful tools.

20.11 Makerspace environment and costs

A junkyard can be idealized as a great Makerspace. Kids can look at numerous unrelated things and come up with their own unique creative ideas on how they can be combined and utilized. The cost of material is low (they are junk), and wonderful creations

can evolve. Some of the creativity comes from the fact that the parts were not meant to work together. From an educational standpoint, I find few problems with this. The problem only arrives when one wants to make actual progress in solving what I call “real world problems,” beyond education. I find the same problem with many undergraduate Senior Design projects. The student often has to start from scratch. What if work was designed to build on prior work and the work produced became a stepping stone for future work? What if the work was partially self-documenting? These are the kinds of visions I have for the TEC Center. It would have many MEC components, and those components would be combined to form more complex components. The components are multidisciplinary and are meant to work together. The vast library of components can exist on the computer, waiting to be 3D printed, combined with raw material, and assembled, keeping costs low. The students are in an environment where we regularly talk about real-world problems and current approaches to solving those problems. They are in an environment where they can see and understand the vast number of components at their disposal to solve these problems. When a component doesn’t exist, they have the skills necessary to create one. This is what we are striving for at the TEC Center. We might utilize a task management system, which helps break down the problems and automatically sets tasks, letting the students know what is needed next. With a philosophy of always trying to work from raw low-cost materials and 3D-printed parts, and students assembling the parts, we keep the cost to a minimum. Making components from raw materials takes time, yet it also opens up the possibility for more replication of work from student to student or Makerspace to Makerspace. It also allows students to design, build, and understand instruments down to

their basic components. A Makerspace filled with physical material is best, but a virtual library of components that can be readily fabricated with a 3D printer and assembled is lower cost, and still opens up a world of possibilities for the student.

As students create tools and more are brought into the Makerspace, we need to make sure proper safety training is maintained to keep it a safe environment. We have an environment where students teach students, but there will always be a need for online training. A website is being developed that guides students and instructors to relevant videos, tutorials, material, experts, etc. This website is currently under development at <http://evolutionarycomponents.org>.

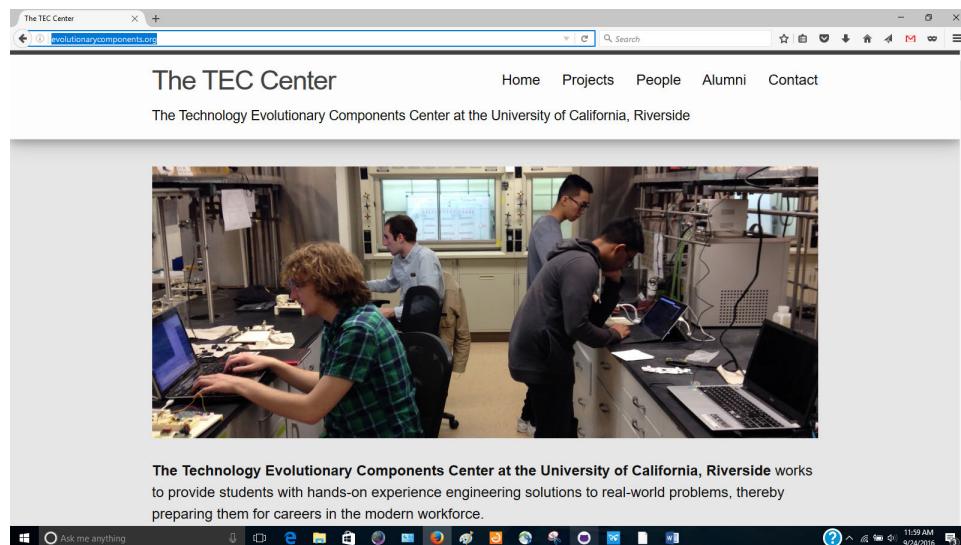


Figure 20.3: TEC Center website

20.12 Website housing the knowledge

When a student is starting a project, one of the first things they must do is define the problem they are trying to solve. Next, they must determine the *knowledge and materials* that are needed to complete the project. This website will help act as a guide to finding answers to these questions.

20.12.1 Real-world problems

Our website will maintain a library of real world problems. With any real-world problem one should start by researching the issue, understanding it and examining the solutions that are being developed or in use. The internet is a powerful resource for this. As students study a problem, we hope to capture their work on this website. We would provide a breakdown of the real-world problems that we are addressing. The website would provide links to websites that have already helped students understand the problem and the research that is currently underway. As students develop PowerPoint presentations summarizing the problems and their solutions, those, too, can be included on the website.

20.12.2 Tutorials

Since the Makerspace instructor cannot be an expert in all the interdisciplinary areas that a student may explore in solving problems, the web becomes a vital resource. The <http://evolutionarycomponents.org> website will help guide students to the appropriate tutorials. Many videos on YouTube and other tutorials online have already been developed by others to teach the skills we need. We will direct our web user to those sites, but we

are also developing our own tutorials that are tailored to teaching the information needed to solve problems utilizing the MEC system. This website and additional more directed tutorials allow students to be more efficient with their time, permitting them to solve problems more quickly.

20.12.3 People

Our website already maintains a list of students who are currently working on the MEC system and those that are Alumni.

20.12.4 Results

The internet-based “cloud” will inevitably change medical diagnostics and instrument development, but today we sit well below what is possible. In fact, some say researchers are in a crisis due to our inability to duplicate experimental results. This topic may be outside what is needed to be discussed on Makerspaces and student projects, but I believe that the sooner undergrad students can be made a real part of the solution, the more engaged and recognized their work will be. Students’ experimental results need to be stored on the cloud, and these results should include a description of how these results were obtained. This includes the equipment used by the students and its setup and operation. Standards are needed on how raw data needs to be stored. This standard would include the information necessary to allow for automated processing. Researchers would be able to automatically comb databases and process this information. The MEC system has taken important steps which will aid this effort. First, we have developed an interdisciplinary schematic for the MEC components that allows the user to fully define (and share) the

hardware they use. Second, our MEC software can record not only data, but also the hardware, software, and operations used to obtain that data. All this information would be stored on the “cloud.” Third, the cost of instrumentation used in an experiment is a barrier to the ability to duplicate that experiment. The MEC system lowers that barrier. Fourth, the MEC system can automate processes, remove human error, and allow for more repetition in testing. Fifth, we have developed scripts for the automated processing of data.

20.12.5 MEC System

Our website will have a section devoted to the MEC system describing how the MEC system functions and its standards. The sections on the website under MEC Systems will include Projects, Components, and Equipment.

Projects

Our website will have a list of MEC projects that are in progress and that are complete. This list of projects will show the variety of systems that can be built and are being built with the MEC system. Seeing what other students have accomplished can be a good source of inspiration for students. Students may chose a new project or extend an existing project. The web project list will have a picture and a brief summary of each project. When one clicks on the project name one will be given a more complete description of the project and its operation. There will be a MEC schematic defining the project, and a complete part list breaking down the components in the project. Results of the data analysis will be given, along with links to the raw data. If the project was presented, the PowerPoint presentation will be provided. If the projects are properly documented they

can also be a good source of information for students. We need to strive to build on each other's work instead of starting over with each student project. The MEC system can help facilitate this with its component-based approach and multidisciplinary schematics. A library of well-documented projects is needed and will be housed on our website.

Components and Equipment

Components, material, and equipment are a critical part of any Makerspace. Students need to be able to find components and understand what is available. The website should contain both the MEC components and raw materials from outside vendors. The website should contain a list of vendors and the types of components they carry. Over time, we have found a good deal of repetition regarding where materials come from, so we will start with the vendors we are currently using. As students locate new vendors and provide links, they will be added to the website, thereby developing a database of resources. As new projects evolve, this database should be able to organically grow from new sources that students find. At the TEC Center one of the most critical materials is the MEC library of components themselves. In addition to components, the website will include databases of equipment available for student use, how to operate this equipment, and any relevant safety information. This includes equipment inside the Makerspace and outside the Makerspace (in area businesses, colleges, universities, etc.).

20.12.6 WebLab

The MEC system provides the ability to automate lab experimental processes. If we combine this capacity with a webcam and a little bit of software, we have what I call

a WebLab. A WebLab-based experiment can be controlled and viewed through the Web. These experiments could be built in cloud lab stations that, like docks for experiments on the Space Shuttle, can accommodate a variety of different WebLab-based experiments. Additionally, it would include a cloud interface with data storage, instrument control, and data visualization. In this way classes provided with the specifications of these cloud web stations could develop their own WebLab experiments. These WebLabs could become a powerful resource for teaching engineering in resource-limited settings. It is important to recognize that it is the MEC system which would allow WebLabs to be constructed in a cost-effective manner.

20.12.7 Library of Inventions

Our website will have a library of inventions. I've always loved science museums, and most importantly, studying how things are made and function. From a simple chain to a Coulter counter, their influence can be seen in my designs. If I taught a class, I would challenge students to study machines or inventions and break down how they function. What simple principles do these machines use? Each student would be responsible for a small piece of the cloud invention museum. Not all things are simple: a student may decide to study how the heart functions. There is much to be learned from existing systems or inventions.

Chapter 21

Outreach

My intern program was outgrowing Dr. Grover's lab when he was able to locate additional lab space for me. I was therefore able to handle a few more interns. We asked Jun Wang, the Director of Student Development and International Initiatives in the Bourns College of Engineering at UCR, to send out an email out informing the engineering undergraduates that there were openings. I was looking for up to five more interns for the TEC Center. When over fifty showed up, it was a little overwhelming. Looking at their resumes, I could see that there was a real need for an intern program that taught basic engineering skills. Instead of turning anyone away, I decided to see if I could come up with a way to include everyone with a real desire to learn. The problem was that the TEC Center was focused on solving some challenging problems, and the skill sets of these undergraduates, for the most part, was insufficient. Therefore, I decided to open a new program. I called it Outreach; its goal was to teach a large number of students the skill sets that industry

needs, and I needed too. My aim was also to solve another problem, which was quality STEM education in the secondary grades, with a focus on engineering and technology.

I presented my students in Outreach with the problem of building a massive “Marble Machine.” Each team’s section of the Marble Machine would teach a fundamental math, physics, chemistry or science principle. It would be demonstrating these principles in the process of moving a marble around a track. The track system would be required to maintain the same standard 0.3-inch grid system the MEC system was using. When completed, it would be brought into the classrooms of lower grades, or students could come see it in the TEC Center, to inspire lower grade students on what engineers were able to accomplish. With mentoring from my students, I was hoping to open up the same engineering challenge of the Great Marble Machine to them. My students also understood that in the process of developing this Marble Machine they would be developing the very skills that I needed at the TEC center, and that industry needs. These very same skills would be what I wanted them to pass on to the students of the lower grades. *Evolution: Outreach a new TEC Center program for teaching engineering skills. Outreach project is based on the development of the MEC Marble Machine.*

21.1 Outreach teaching leadership

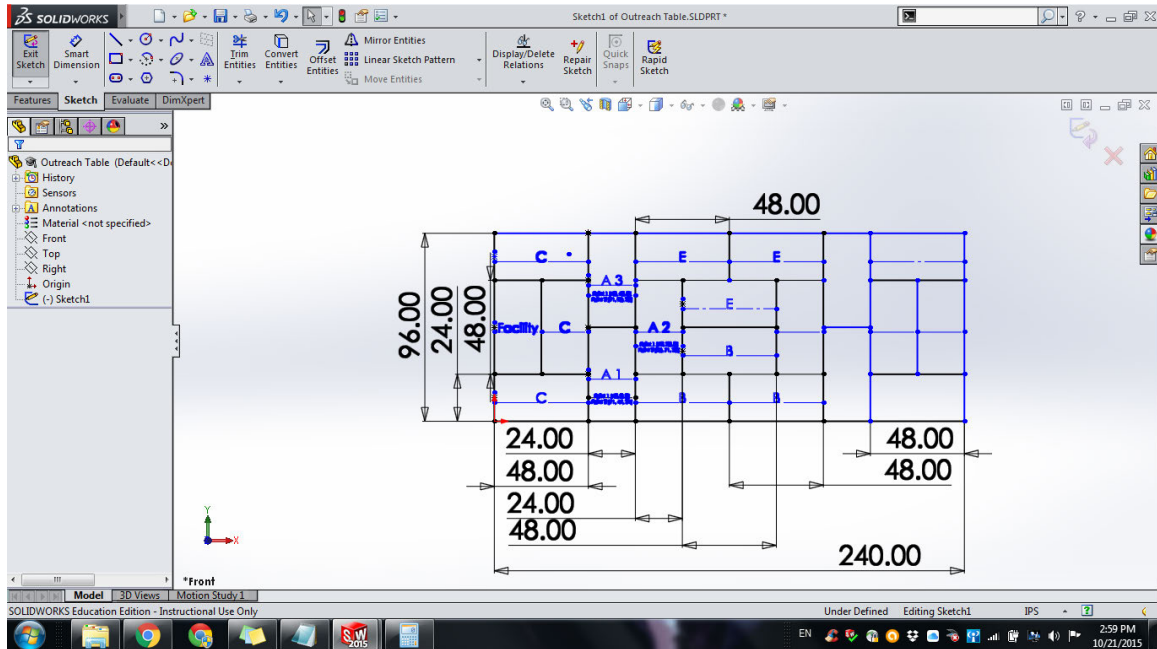


Figure 21.1: SolidWorks layout showing the five groups and the three teams per group. This was one of the first SolidWorks designs done by an Outreach student.

We began by breaking into five groups. Each group was made of three teams, and each team was made of two or three students. I assigned a team leader to each group. Coordinating group meetings was one of the first challenges the team leaders faced. We had interns from electrical engineering, computer science, material science, bioengineering, and chemistry. I worked at separating the various skills among the different groups. Each team would have a 24x48 Snap baseboard to build from. (1 Snap unit equals 0.3 inches). Shown in figure 21.1 is the positioning of each group and team.

21.2 Students teaching students was a constant theme in the Outreach program.

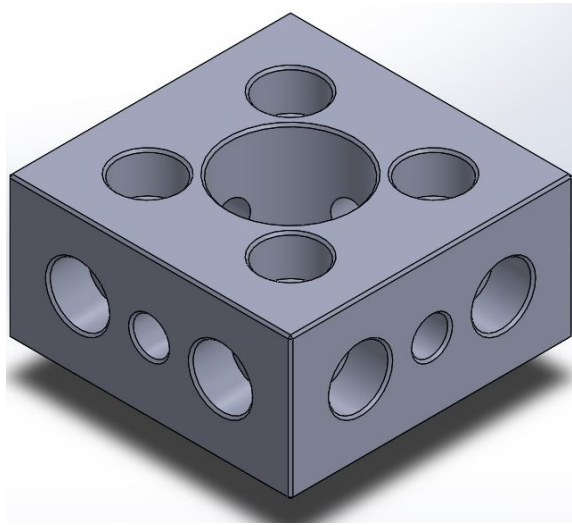


Figure 21.2: The part drawn in the first Outreach SolidWorks training class.

Razn Abu Qamar, one of the team leaders, offered to hold a SolidWorks class to help bring everyone up to speed on designing in SolidWorks. During the class, I was happy to see one of my TEC Center interns joining in. In addition to teaching SolidWorks, they also learned about some of the MEC mechanical standards. Shown in figure 21.2 is the part that was drawn by one of the Outreach students during the class, a rod support. The rods hold up the track rails in the Marble Machine.

21.3 The design will talk, if you listen

Notice the rod is centered between Snap holes. It could have been centered over a Snap hole. This Marble track system had *not* been envisioned before Outreach started.

What makes this a special learning experience is that the Outreach students are getting to observe and participate as the Marble Machine evolves for the first time. When I first defined the marble track system, I had told them that the marble would pass through a Snap grid point. I had selected a 7/16 inch marble or ball bearing. Early one morning, about 4:00 a.m., the design began talking to me. It was telling me I could get a better track pack density if we moved to centering in the middle of four Snap holes, instead of on a Snap hole. This can be understood by this simple part (Fig. 21.2). If the rod came from the Snap grid point one would need an additional Snap grid set of points outside the rod. This would make the support a 3x3 Snap block as opposed to the 2x2 Snap grid block as shown in figure 21.2. This will set the spacing between tracks. As a designer, I like to be creative, yet as this demonstrated, one often learns a set of rules. Apply those rules and constraints, and if one listens, the design evolves itself. I went in the following day and shared these thoughts with my students. As I walked them through the logic, they saw a system evolving. In that same conversation we also discussed the spacing of the two 0.125-inch rods that support the marble. We wanted the Snap blocks to be 2 Snaps or 0.6 inch wide. That would hold the rods. Ideally, one wants the rods to be as far apart as possible so the marble will not fall off, but one also does not want the 3D-printed material to get too thin. Here, one has two conflicting constraints, which often occurs in design. We decided on 0.4 inch spacing. Doing the math, $0.6 - 0.125$ (1/2 of both rods) $- 0.4 = 0.075$, divided by two equals 0.0375 inch wall thickness for 3D-printed material. I would have preferred 0.05 inch, so this was thinner than I wanted. Today, we often oversize our parts to keep the 3D printed walls from becoming too thin. This would suggest that we should have moved the tracks closer

together. We also sometimes have the marbles fly off on turns, suggesting just the opposite.

Sometimes, in design, there are no perfect answers.

21.4 Marble Machine inventory



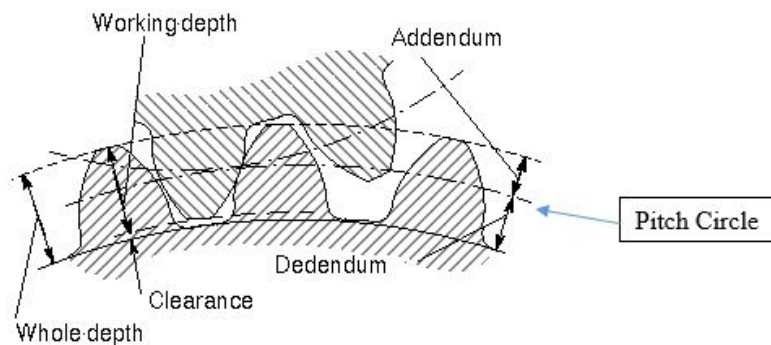
Figure 21.3: MEC inventory system under development. This cupboard is dedicated to all the marble track components that are being developed.

I can count seven different students' work shown on the Marble Machine cupboard (Fig. 21.3). This was a great opportunity for many of them to learn SolidWorks. Since this is what I call a linear problem, there is no limit on what a student can do. Outreach student

Manoel Tamraz took this to heart and was the creator of a large number of the SolidWorks parts shown. What makes this more impressive is that he did not know SolidWorks before he started the Outreach program.

How to Make an Involute Gear on Solidworks

1. Access the Excel spreadsheet that includes the equations for involute gears.
2. Changing B-2 will adjust the pitch of the gear teeth. We use 20 pitch for our gears in the lab so you will most likely leave this value alone.
3. Changing B-3 will adjust the number of teeth on the gear. Adjusting this value will change the size of the gear. The snap size of the gear can be found at C-8. This is called the Pitch Circle Diameter. It is the circle where the teeth from one gear will meet the teeth of another gear.



4. Changing B-4 will adjust the pressure angle. You probably won't need to adjust this

Figure 21.4: Short tutorial on how to make an involute gear in Solidworks, by Manoel Tamraz

Manoel progressed to developing gears, which required integrating math equations into his SolidWorks designs, something I've never done. He put together a tutorial on how to build an involute gear in SolidWorks so that others like myself can learn the process (Fig. 21.4). *Evolution: MEC system added involute gears to its list of parts.*

21.5 Persistence is a key factor in a student's success.

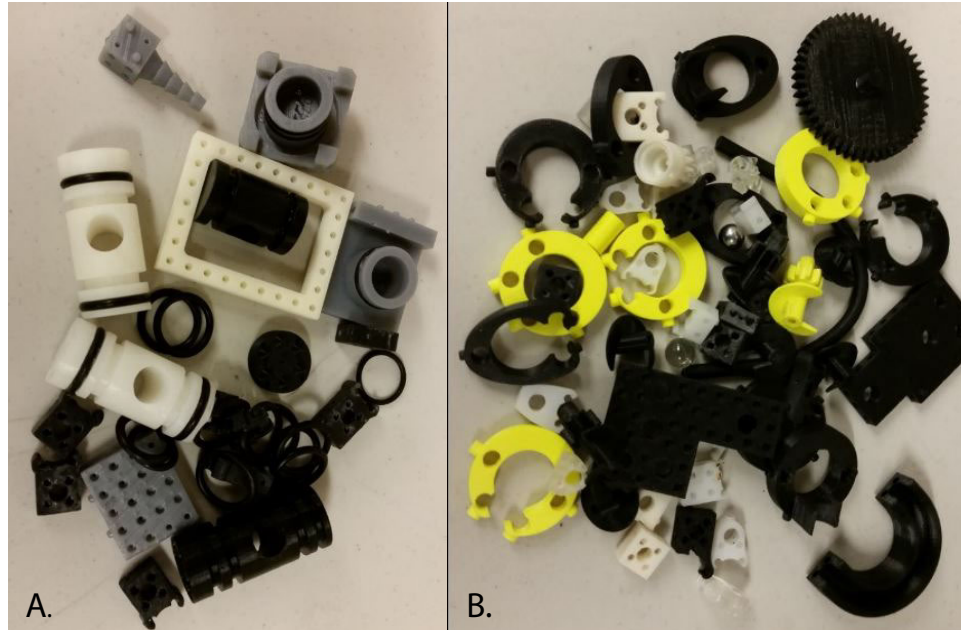


Figure 21.5: Marble transport parts Hayden Karich was evolving (A). Marble transport parts Manoel Tamraz was evolving (B).

Manoel was working on using a gear system for raising a marble from a lower track level to a higher track level. Shown in figure 21.5 (B) are some of the parts he developed in an effort to build this system. Designs often have problems, and often require several iterations to make function. He is now working on a printed circuit board design for the motor driver that his system needs. Again, what surprised me was how adapt he was at grasping the material when he had never worked with electronics before, yet he had always wanted to.

Hayden Karich led one of the most successful Outreach groups. Shown in figure 21.5 (A) are parts that Hayden's team had developed. Often, Outreach develops parts and systems that find applications at the TEC Center. In figure 21.5 (A) the reader can

remained active at the TEC Center. Matthew Smith has been developing a marble track joint that utilizes a servo. This would allow the user the option of two tracks to divert a marble down. Lately, he has started working on a multi-vial sampling system for the bioreactor. Krishna Addanki has worked on several of TEC Center projects. He developed the original software for the accelerometer weight scale system and is currently working on a bug zapper for the Panama project. Vishak Kumar made the original Visio designs shown in figure 21.6. When regarding the Marble Machine schematic design and comparing it with the bioreactor, one will quickly realize that the skills the students are developing here can directly be used at the TEC Center. This is confirmed by the TEC Center projects the Outreach students have moved into.

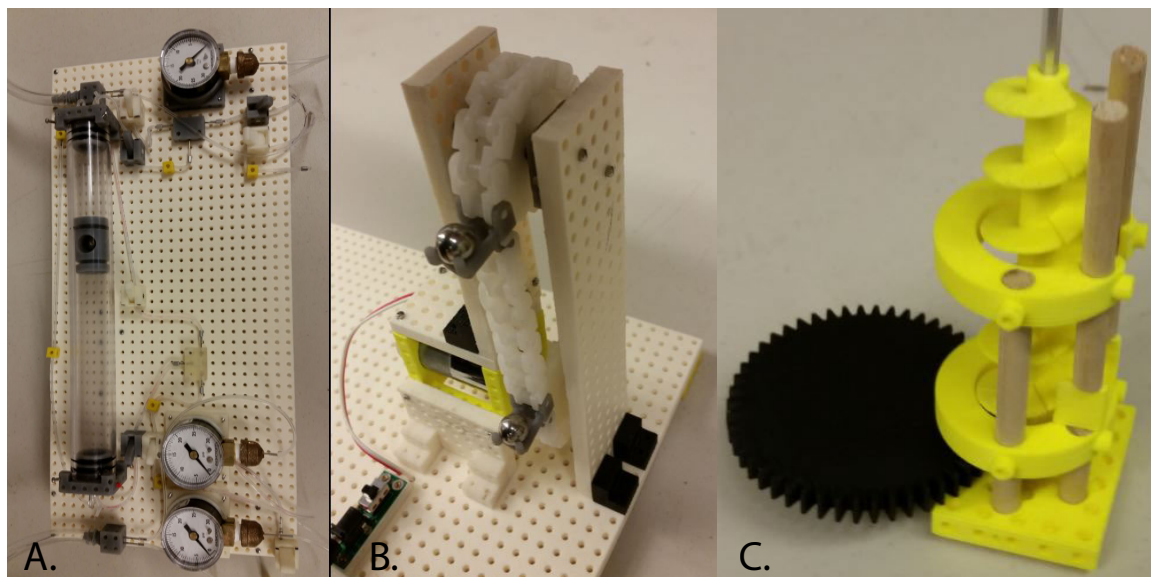


Figure 21.7: Marble elevator lift system (A). Marble chain lift system (B). Marble gear lift system (C).

The Marble Machine system has an unlimited number of ways that it can be accomplished. Shown in figure 21.7 are three of the methods that are still being developed

for lifting the marble from one track level to another. The developing of these systems has been invaluable in teaching engineering and SolidWork skills. Shown in 21.7(A) is the elevator marble transport being developed by Hayden's group (although mainly Hayden now). It has been laid over on its side for testing. We have also looked at this system for rapid transport of a vial between experiments. Shown in figure 21.7(B) is the chain lift system. The chain and sprocket were drawn up by Allen Partono. Xiqiao Zhang, a new intern, has been working on the part where the marble enters the bottom of the chain lift system, waiting to be lifted up. Mathew Smith is working with the chain system to transport vials. Shown in figure 21.7 (C) is the marble gear lift system being developed by Manoel Tamraz. As the tall screw gear turns it will lift a marble up from a lower track level to an upper track level.

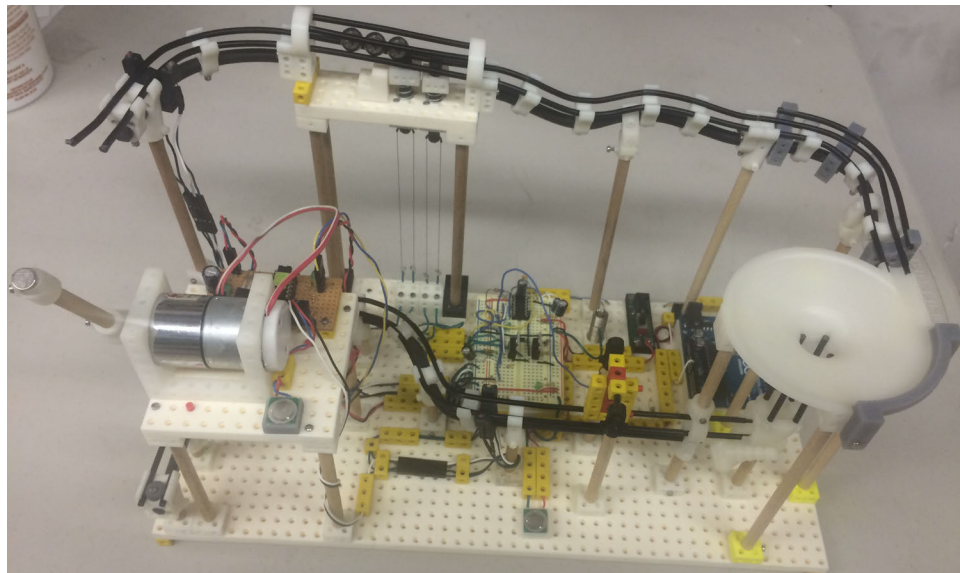


Figure 21.8: First working marble machine developed in the Outreach program.

I designed the original construction of the marble machine shown in figure 21.8. Many of the components I used were created in SolidWorks by students. A key element of any Marble Machine is lifting the marble. I accomplished this with a motor and a magnet on the end of a single spoke. As the motor turns the spoke with a magnet on the end, it lifts the metal marble between the rails. Next, I wanted a way to release the marbles in a timed fashion. Here, I used a dual gate system actuated with muscle wire. As current is run through muscle wire it heats and contracts. The contraction pulls a pin down, releasing a single marble to a lower gate. The pin then goes back up, holding back any additional marbles from following. Then the system heats the next muscle wire, pulling the next pin down, and a single marble is released. *Evolution: Muscle wire is a good actuation material.* The marble flows around a turn and into a funnel and then drops down to a lower rail, around a few more turns and stops at a spot waiting to be lifted. The marble is then picked up by the magnet and raised back to the upper rail, and placed back on the track, where it rolls back into the dispenser. The process can then be repeated. Along the way, two photo detectors track the marble's movements. Two embedded microcontrollers are used for control and feeding information back to the laptop. The lower photo detector is used to tell the motor when to turn on and does so when a marble is detected. When the upper photo detector detects a marble, it tells the system to release a marble from the dispenser. The last section of software being developed for the MEC system is the integration of devices. The Marble Machine has also shown itself to be well suited for the advancement of this section of the MEC system software. *Evolution: Development of*

state machine structure in User Main code for the Marble Machine. Example of integrating devices for device code that was automatically generated.

In summary, Outreach is designed to teach a large number of students engineering skills and principles, preparing the students for rigorous real-world challenges. Much of what we learn and develop in Outreach is being translated to assist in problems the TEC Center is facing.

Chapter 22

Practitioners

The focus of the TEC Center is problem solving. It began with the development of the MEC system for easier development of biological instruments, and in the process, became a tool for training students. As our students' skills improve along with the MEC system we will be reaching out more and more to practitioners. Practitioners are people out in the world, working to help people or the environment in a real and direct way. The practitioners who see our technology as a way to help become the customers of the TEC Center. Practitioners are essential in helping define the problems.



Figure 22.1: Website of Floating Doctors, an organization with which the TEC Center is working.

An example of one such practitioner is Noah Haas, one of the founders of Floating Doctors (Fig. 22.1). The mission of Floating Doctors is “to reduce the present and future burden of disease in the developing world, and to promote improvements in health care delivery worldwide” [33]. His group has been helping indigenous people in Panama, and he sees technology as an important tool in that effort. As one of his many ways of helping, he assists in getting solar panel arrays installed, bringing power for the first time to communities, and with it, internet connectivity. Working directly with the indigenous people of Panama gives him an important perspective on the problems that they face. Some of the first problems he helped define for me were environmental. We have touched on several problems for the Panamanian people, which I will outline here:

22.1 Silt damaging the ocean environment

Problem: Panama experiences heavy rains. With these rains, silt is brought down to the ocean from the rivers and streams. The concern Noah Haas has is that the silt may be damaging the coral reefs and affecting the ocean environment. The ocean is an important local food source. Coral reefs take hundreds of years to form and are irreplaceable. The coral reefs are also a tourist attraction, which is an important revenue source. The silt runoff is exacerbated by deforestation. Changes in the ocean environment are also occurring due to global warming.

Solution: The development of an ocean monitoring station. If the TEC Center were to develop such a station, it would have a modular platform, allowing for various oceanic sensors to be added. Silt monitoring using optical density will be one of the first sensors to be incorporated. It would deploy a small, tethered buoy to uplink information. This buoy would also be used to determine tide levels. The buoy could be pulled back down to the station or left on the surface. Some of the water measurements we would be monitoring would include salinity, particulate matter (silt), oxygen level, temperature, tidal levels, and pH. A camera combined with image analysis can also be a powerful monitoring tool. Environmental monitoring has a broader importance in keeping our planet healthy. When developing this type of device we would need to be in contact with organizations such as Woods Hole Oceanographic Institution. Bringing power to the station would be one of the first challenges we would face. I would prefer not to use batteries due to cost and the need for replacement. I would look at tidal movement as a way to bring energy to the station.

22.2 Weather conditions endangering lives

Problem: Certain diseases may have some correlation with the weather. For example, rain may carry the feces of the triatomine (an insect) down from thatched roofs onto humans below, or into their water supply. This is not thought to be a common method of transmission, but it still would be interesting to see if this or other diseases have any correlation with weather. Various weather conditions may be far more conducive to the growth of mosquitoes, so there may be correlations with malaria and weather conditions. Weather can have a more-direct danger to human life. The indigenous population in Panama may travel between islands in open canoes. Sudden bad weather can result in drowning. Floating Doctors, as their name suggests, travel by boat, and they, too, are affected by adverse weather conditions. With a large number of volunteers coming each year, it is important for them to be able to travel as safely as possible. Certain weather conditions may cause strong undertows at the beach, which have taken local lives. Inclement weather may affect crops, leaving the local population with less food. In addition, weather monitoring stations could be important in tracking the change in the local environment during global warming.

Solution: Placement of local weather stations. Here, it would be interesting to correlate local weather conditions with health. Can certain diseases be correlated with extreme weather conditions, or a specific weather pattern? If weather is tracked in this local region, what type of shifts are seen over time? Do we see the effect of global warming on the region? Again, environmental monitoring can help determine if human activity is affecting our planet, and if so, in what way. Even though this is a remote area, it still might

be fruitful to study air pollution that is carried to this region. One of the first steps in building a weather stations is investigating what is already available.

22.3 Chagas disease is killing people and lowering the quality of living

Problem: Chagas is a deadly disease that can affect the heart and digestive system. The disease, which comes from a parasite, is transmitted to humans from triatomine feces. This insect also bites and draws blood at night. The insect is found in the thatched roofs of the local population. One method of transmission is that after the insect bites, it defecates. The person then scratches and carries the insect's feces into the open sore, where the person becomes infected. It also has been transmitted through mucosal tissue in eyes and mouth. The disease can take a decade to show symptoms. We also know that the triatomine is attracted to light and many odors.

Solution 1: I could convert the bioreactor into a bug farm that could collect fecal matter from insects. We then could develop tests on the insect, and tests on the parasites in the fecal matter. At UCR, for safety, I would first use a safe beetle to monitor growth and develop processes. When a person becomes infected, we would want to develop an early test that is easy to perform and detect the disease. The first step would be to look at what is already being done and seeing if there is anything we could add.

Solution 2: Pyrethroids (a class of insecticides) could be used to treat thatch. Insecticides have been very effective in Brazil.

Solution 3: Insecticide lamps—cloth lamps which have been treated with insecticide. A cloth lamp could be inexpensive and compact for shipping. It could be attractive. Many insecticides degrade with light, and the ones that don't may be found to be more of an irritant to humans. Selection of insecticide is critical. We could develop a spray to boost the cloth lamp's insect killing properties. The lamp could have a solar power link so additional power would not be required. I would place a dead bug catcher at the bottom of the lamp. We could also develop a phone app. The user of the insecticide lamp would dump the contents of the dead bug bucket onto a white cloth and spread the dead bugs out. One would then use the app to take a photo of the bugs. The app would then count and categorize the bugs. Different odors could be placed on the lamp to increase its attraction. We could place two lamps in a home with different treatments. Then, over time, we could learn which treatments are most effective. If this method worked it could provide an early detection method for triatomines. In a village, one person could have the job of cleaning out the dead bug buckets and taking pictures, so only one phone would be required.

Solution 4: Replacing thatched roofs with corrugated iron. This is already being done. The question is whether this is the most cost effective solution, and does this affect the temperature in the home? A temperature sensor in the home could answer the latter of those questions.

Solution 5: Zapper lamp—electrically zap the bug. One could use a bucket, as described above, and image the contents, or a funnel drop could be attached onto a trap door when one detects a bug victim (detect discharge). One would image the dead bug and then drop the bug into a bucket. All the bug traps would be networked and then the

information uploaded onto the internet. Here, one would be able to collect a large amount of data, and very quickly determine where there is an infestation problem. It should be noted there is already work going on at UCR in this area.

22.4 Potential extinction of the Panamanian golden frog

Problem: The rainforests hold a vast variety of the Earth's biodiversity. This biodiversity cannot be replaced and must be protected. The Panamanian golden frog is at the point of the extinction.

Solution: One idea that has been used for monitoring the environment is a network of sensory stations. What I would plan to add to this is the ability of the station to identify the Panamanian golden frog. In this way we could help track its population. The same technique could be applied to the howler monkey and other indigenous animals. We could start by recording sounds from the zoo for various animals and then learn how to identify them from their sounds. This could include humans. We would translate the sounds to the frequency domain and apply pattern recognition to identify which animal is making the sound. I envision a network of sound recording stations in the forest. Each node of the network would transmit this information to nodes around it. The information would be passed from node to node until it reached an uplink node. This node would allow the information to be uplinked to the web. The sensory station would monitor temperature and record sound. They would use a solar cell for power, which could provide information on sunlight. Additional information that could be recorded would be humidity. One could

install IR sensors to record IR levels over time. As ozone levels change, how does that effect IR levels? Noah Haas is aware of a reserve where this could be implemented in Panama.

22.5 Health problems due to smoke inhalation from wood-burning stoves in homes

Problem: In small villages, wood burning-stoves are common in the household. Smoke from these stoves can present a health risk.

Solution 1: As power becomes available, install a small fan.

Solution 2: Villagers could use non-wood-burning stoves that have been developed for this purpose. They could use small electric stoves (if power was available).

Solution 3: A fan that uses heat from the stove to produce electricity, and that electricity utilized to drive a fan.

Solution 4: We could monitor the smoke levels in the house. Here, I would recommend a smoke detector sensor as the base sensor. I would also monitor temperature. We could look at monitoring light in the house. This could be useful in conjunction with the insecticide lamp and for understanding the differences between homes. These sensors would be networked together from home to home like the forest network.

22.6 Poor health due to poor nutrition

Problem: Proper eating and having the availability of balanced nutritional foods, needs to be studied.

Solution 1: Nutrient bars are already being used to help address this problem.

Solution 2: See section below on “Better local food source.”

Problem: Because of a lack of access to modern medical facilities and doctors, patients may not get properly diagnosed in a timely fashion, if at all.

Solution 1: More low cost diagnostic equipment and testing. If one can reduce the cost of equipment or tests, one reduces the barrier to accessibility. Better patient diagnosis and treatment is needed (i.e. better health care).

Solution 2: Improved awareness of the problem. The more people are aware of the problems, the more likely organizations like Floating Doctors will get volunteers and donations.

22.7 Sickness due to poor water quality

Problem: Fresh water quality that is not properly maintained. This problem needs to be looked at separately for each community because they do not use a central water system. For those communities that are using rain runoff, we need to study if fecal matter from birds, animals and insects can be contaminating the water. If water is stored in a tank, is there opportunity for bacteria to grow? What tests and treatment are currently being done?

Note: As with most of these problems, we need to do research first. Here, we need to learn how water quality is maintained. What tests are run to insure the water meets standards, both in the US and Panama?

Solution 1: I imagine having a separate, smaller tank which filters water from the primary tank. Here, we need to determine what contaminants can be filtered out and what

cannot. How do we make sure to maintain a clean filter? One thing we can do is monitor the pressure differential across the filter. When the pressure gets too high, the filter needs to be cleaned or replaced. A pump or gravity system can be used to drive fluid to the smaller tank. Water levels can be monitored in both tanks.

Solution 2: Part of the solution involves going to Panama and studying how the water quality is maintained, including bringing water quality test kits.

Solution 3: We need to develop methods of studying water quality remotely. Here, the MEC system is well suited for developing a remote water quality test station. It will be advantageous to develop a portable water testing station also.

22.8 Better local food source

Problem: Not having enough food. Not having enough variety or quality of food. Not having enough extra food to sell or exchange. Volunteers come to work for Floating Doctors, so a food source for them is also needed to reduce the need to pull food away from the indigenous population. The food source for these people is a smaller challenge, and may be a good place to start. Panama has heavy rains which can affect crop yield.

Solution 1: Improve what the indigenous population is currently doing. Here, we need to understand what they are doing and comparing it with best practices.

Solution 2: The plants that the locals are currently growing can be genetically evolved to have better yields in the environment they are in. They can be made more resistant to mold or bug infestation. This can be done through rapid mutation under controlled

environmental conditions. Again, the obvious first place to start is communicating with plant biologists at UCR.

Solution 3: Better farm equipment. On the web there is a group developing open plans on how to build farm equipment, such as a tractor [34]. These methods need to be studied.

Solution 4: Best practices in green houses may be a way to control the quality of food and increase variety.

Solution 5: Vertical farming is another way to control food production.

Solution 6: Pests can cause enormous damage to crops. Early detection of a pest problem can be critical to reducing the damage. Also, with proper monitoring, one may be able to reduce the amount of pesticides needed. The monitoring could be done with sensor networks. This network could capture and identify pests, and then uplink this information to the web, as discussed earlier.

22.9 Better education

Problem: It is not always ideal for us to go into these types of communities and start solving problems. We are often only there temporarily and our solutions may be incomplete. Solutions often take time to evolve; a simple broken part may make a piece of technology useless. Not understanding the locals' current methods may make them not accept our solutions, often with good cause.

Solution 1: It is better to train people who have a long term commitment to the community (often local people) on how to implement solutions and ways of approaching the

problem. The more they understand and we listen, the better the chances the solution we develop together will work. If we developed a technology training center, I could envision people from these communities coming here to learn. They could be hosted by local families. I grew up with foreign students living in our home my whole childhood and it was a rewarding experience.

Solution 2: Agriculture is an excellent area to start education. This could be a garden or green house. Here, it is ideal to study the best practices in the agriculture that is currently being grown, and look how additional varieties can be added. This is a potentially fruitful area to begin, since the ability to improve yield and health of a crop is of immediate value to the community. Modern methods of soil and nutrient testing can be taught. We would also include methods to reduce damage to crops from pests, fungus and weather damage.

22.10 Indigenous population needs deed to land

Problem: Indigenous people in Panama often do not have the deed to the land they live on. This can lead to the land being taken from them with no recourse. It has also been shown that indigenous people are often the best custodians of the land they live on.

Solution: Get the indigenous people in Panama the deed to the land they live on.

22.11 Retention of local population and culture

Problem: As we hope to help a community, we bring in things like the Internet, a tool that almost every child in America takes for granted. In so doing, however, we

open up the possibilities for the youth to see what exists outside their community. This can (and most probably will) lead to youth leaving the community, creating a drain on a community's labor and intellectual development. Communities that work together are becoming harder to find in the US. These are communities where neighbors know their neighbors, where neighbors work together to form a strong community and have a long history in its development. These communities provide support and a strong feeling of belonging. So, the loss of this is something we must try to avoid. On the other hand, people need to be allowed to follow their dreams and passions.

Solutions: Work to remove the reasons a person would want to leave home and their community. This can be because of poor health-care and living conditions. It can also be from a lack of opportunities to grow and better oneself within their community.

22.12 The ill effects of jungle deforestation on plants, animals, and ocean life.

Problem: Deforestation kills plants, animals, and insects, sometimes to extinction. This includes some of the plants and animals in the ocean.

Solution 1: Higher crop yields can reduce acreage needed.

Solution 2: Farming is one of the reasons for deforestation. Vertical farming can reduce space required by an order of magnitude. We can also provide early crop disease detection and identification.

Problem: Often diseases are not detected in their early stages. This is especially true for diseases like Chagas. Without early identification, the treatments can be more severe and costly.

Solution 1: The “lab in a latrine” (or outhouse) is a concept I have been envisioning for some time. Here, one would develop methods of processing a person’s urine and fecal matter in a latrine-like structure for biomedical information. This information could then be unlinked on the web for data analysis.

22.13 Better housing conditions

Problem: Poor quality housing can lead to illness. This can be due to lack of control due to weather conditions. It can also be difficult to control bacteria, fungal growth, germs, insects, and other potential health hazards in poorly sealed homes. Limited housing space can also reduce one’s ability to solve problems.

Solution 1: Online there were instructions on how to fabricate a brick maker and plans for a simple home [34].

Solution 2: One might also look at growing better building material, such as bamboo.

Solution 3: Procure donations and volunteers to construct better housing, looking to organizations like Habitat for Humanity.

Problem: For hundreds of years the indigenous medicine men have developed methods for treating their people. As modern medicine and doctors come to help, some

of those old methods will be lost. Some of those traditional methods may hold answers to problems still faced by modern medicine.

Solution: Record old methods. Analyze the plant materials used and how they affect the human body (i.e. one's cells).

22.14 Conclusion to Panama problems

The problems and solutions listed here are simply a point to begin a discussion. Until research is done, the problems and solutions will be poorly defined. The point of including this is to show how much rich information can come from discussions with practitioners. Also, this provides me and my students a feeling that the work that we produce has the real potential to touch lives. When someone points out where solutions presented here are off track, I am one step closer to the right solution.

Chapter 23

Conclusions

Bioengineering has an exciting future. It will transform the medical industry from reactive to predictive. Translation of existing methods for the prevention and prediction of disease, from research to practice, will be an important step in this transformation. The MEC system that is evolving at the TEC Center at UCR is uniquely qualified to facilitate that transition for several reasons. First, it will fill a void in the biomedical industry for the rapid creation of instruments, which are needed for many of the processes that are being developed. The MEC system has developed symbolic design to aid in the rapid development of these instruments. MEC instruments can be designed to automate existing processes, automatically analyze data, document methods, and feed the results onto the Web. The MEC system has developed a powerful software system with automatic code generation, which is essential in instrument development for automating processes. Large data sets will form, ever improving our predictive and preventive methods. Once these processes are

automated in this fashion, with automatic documentation, translation to other institutions for verification and replication will be straightforward.

I hope to expand the TEC Center and make it a leader in the translation of state-of-the-art methods in the biomedical industry, such as the CRISPR/Cas 9 process. I speak of biomedical instruments, but bioengineering (and the MEC system) goes far beyond that, delving into agriculture, water quality and environmental monitoring. The mission of the TEC Center is to “save the world one step component at a time, and in the process train the next generation of scientists and engineers.” Some people say my mission is too broad, but for each individual the words “save the world” mean something different. For one student it might mean micro-farming to help prevent famine. For another it might be preventing the Chagas disease to save lives in Panama, and to another it could be saving an endangered poisonous frog. The point is, if you can tap into a student’s passion, there are few limits on what they can accomplish. As they work, they will not be alone, for energy, effort, and progress draw other students in. The one constant among all my students is that they are good people, and they all want to help. Students teaching other students new skills with the MEC system is at the core of the TEC Center’s success. Thus, the second reason the MEC system is uniquely qualified is its ability to facilitate STEM education. The “Hole in the Wall” experiment by Sugata Mitra [35] showed the self-growth kids can achieve when given access to the right intrinsically motivating technology and purpose. The ability of a student to evolve solutions to solve complex real-world problems, which they or their peers did not think were solvable, is intrinsically motivating. This is what the MEC system does. Therefore, if someone was to ask, “can a model be provided for the best practice for teaching

STEM education?” I’d respond that I believe that the TEC Center is such a model. Its student testimonials confirm that. It uses science, technology and math in the engineering processes. The MEC system allows for the rapid design and building of instruments, with the student evolving the process. MEC software allows for close-to-immediate functionality.

I use the term “STEM education” because that seems to be in favor today, yet I believe we should be looking at all the skills that are needed in the 21st century. That would include English and business. The TEC Center is like running a business inside the University, with all the excitement of a startup. We sometimes call ourselves a Makerspace, but I believe a business is a better description. The TEC Center is product-driven, with the evolving MEC system at the core of product development. To aid in this, we are putting QA documents together on how to fabricate products. The students are helping develop an inventory and part numbering system. We develop as many instruments to test components as instruments themselves. The Center has a website, evolutionarycomponents.org, to market our achievements in an effort to get funding. Thus, in addition to training the next generation of scientists and engineers, I believe we also help to teach skills needed for business and entrepreneurship. I am eager to evolve to the next level of MEC development, which is the use of the MEC system to work with practitioners, such as the UCR School of Medicine, industry, Floating Doctors, or other labs, to solve problems.

Bibliography

- [1] Marshall Don Graham. The coulter principle: Imaginary origins. *Cytometry Part A*, 83(12):1057–1061, 2013.
- [2] C Gonzalez, SD Collins, and RL Smith. Fluidic interconnects for modular assembly of chemical microsystems. *Sensors and Actuators B: Chemical*, 49(1):40–45, 1998.
- [3] Minsoung Rhee and Mark A Burns. Microfluidic assembly blocks. *Lab Chip*, 8(8):1365–73, Aug 2008.
- [4] Po Ki Yuen, Jody T. Bliss, Christopher C. Thompson, and Richard C. Peterson. Multidimensional modular microfluidic system. *Lab Chip*, 9:3303–3305, 2009.
- [5] Krisna C Bhargava, Bryant Thompson, and Noah Malmstadt. Discrete elements for 3D microfluidics. *Proc Natl Acad Sci U S A*, 111(42):15013–8, Oct 2014.
- [6] Gordon E. Moore. Cramming more components ont integrated circuits. *Electronics*, pages 114–117, 1965.
- [7] Open standard for mec interfaces, <http://evolutionarycomponents.org/standard>.
- [8] Younan Xia and George M. Whitesides. Soft lithography. *Angewandte Chemie International Edition*, 37(5):550–575, 1998.
- [9] Peter C. Simpson, Adam T. Woolley, and Richard A. Mathies. Microfabrication technology for the production of capillary array electrophoresis chips. *Biomedical Microdevices*, 1(1):7–26, 1998.
- [10] Wentao Li, Tao Chen, Zitian Chen, Peng Fei, Zhilong Yu, Yuhong Pang, and Yanyi Huang. Squeeze-chip: a finger-controlled microfluidic flow network device and its application to biochemical assays. *Lab Chip*, 12(9):1587–90, May 2012.
- [11] Luciano Galdieri, Swati Mehrotra, Sean Yu, and Ales Vancura. Transcriptional regulation in yeast during diauxic shift and stationary phase. *OMICS*, 14(6):629–38, Dec 2010.
- [12] Francis S Collins and Lawrence A Tabak. Policy: Nih plans to enhance reproducibility. *Nature*, 505(7485):612–3, Jan 2014.

- [13] National Research Council. *A framework for K-12 science education: practices, cross-cutting concepts, and core ideas*. The National Academies Press, Washington, D.C., 2012.
- [14] S. Patil Santini, F. and IceLuva K. Scheffler. A scripting framework for mr image reconstruction based on free software, concepts in magnetic resonance part b: Magnetic resonance engineering. 39B:1–10, 2011.
- [15] V. Madahar Song, Y. and J. Liao. Development of fret assay into quantitative and high-throughput screening technology platforms for protein-protein interactions. *Annals of Biomedical Engineering*, 39(4):1224–34, 2011.
- [16] J Pawley. The 39 steps: a cautionary tale of quantitative 3-d fluorescence microscopy. *Biotechniques*, 28(5):884–6, 888, 2000.
- [17] et al. Pfeil, S.H. A microfluidic mixing system for single-molecule measurements. *Rev Sci Instrum*, 80(5):055105, 2009.
- [18] Jennifer Healey Leonidas Kontothanassis Beth Logan Alex Nelson JM Van Thong Amir Bar-Or, David Goddeau. A system architecture for real-time processing of physiological signals. *BioStream*, 2004.
- [19] Arjun P Sudarsan and Victor M Ugaz. Printed circuit technology for fabrication of plastic-based microfluidic devices. *Analytical chemistry*, 76(11):3229–3235, 2004.
- [20] George M. Whitesides. What comes next? *Lab Chip*, 11:191–193, 2011.
- [21] W. Dungchai, O. Chailapakul, and C. S. Henry. Use of multiple colorimetric indicators for paper-based microfluidic devices. *Anal Chim Acta*, 674(2):227–33, 2010. Dungchai, Wijitar Chailapakul, Orawon Henry, Charles S Evaluation Studies Research Support, Non-U.S. Gov’t Netherlands Analytica chimica acta *Anal Chim Acta*. 2010 Aug 3;674(2):227-33. Epub 2010 Jun 25.
- [22] Karthik Pitchaimani, Brian C Sapp, Adam Winter, Austin Gispanski, Toshikazu Nishida, and Z Hugh Fan. Manufacturable plastic microfluidic valves using thermal actuation. *Lab on a Chip*, 9(21):3082–3087, 2009.
- [23] Anthony K Au, Hoyin Lai, Ben R Utela, and Albert Folch. Microvalves and micropumps for biomems. *Micromachines*, 2(2):179–220, 2011.
- [24] Todd Thorsen, Sebastian J Maerkl, and Stephen R Quake. Microfluidic large-scale integration. *Science*, 298(5593):580–584, 2002.
- [25] Marianne E Harmon, Mary Tang, and Curtis W Frank. A microfluidic actuator based on thermoresponsive hydrogels. *Polymer*, 44(16):4547–4556, 2003.
- [26] Boris Stoeber, Zhihao Yang, Dorian Liepmann, and Susan J Muller. Flow control in microdevices using thermally responsive triblock copolymers. *Journal of microelectromechanical systems*, 14(2):207–213, 2005.

- [27] David T Eddington and David J Beebe. Flow control with hydrogels. *Advanced drug delivery reviews*, 56(2):199–210, 2004.
- [28] Michael J Bassetti, Aveek N Chatterjee, NR Aluru, and David J Beebe. Development and modeling of electrically triggered hydrogels for microfluidic applications. *Journal of microelectromechanical systems*, 14(5):1198–1207, 2005.
- [29] Laos A Hirano, Marcia T Escote, Luiz S Martins-Filho, Gerson L Mantovani, and Carlos H Scuracchio. Development of artificial muscles based on electroactive ionomeric polymer-metal composites. *Artificial organs*, 35(5):478–483, 2011.
- [30] Yoseph Bar-Cohen. Electroactive polymers: current capabilities and challenges. In *SPIE’s 9th Annual International Symposium on Smart Structures and Materials*, pages 1–7. International Society for Optics and Photonics, 2002.
- [31] Sia Nemat-Nasser and Yongxian Wu. Comparative experimental study of ionic polymer-metal composites with different backbone ionomers and in various cation forms. *Journal of Applied Physics*, 93(9):5255–5267, 2003.
- [32] Shirin Mesbah Oskui, Graciél Diamante, Chunyang Liao, Wei Shi, Jay Gan, Daniel Schlenk, and William H. Grover.
- [33] Floating doctors, <http://floatingdoctors.com>.
- [34] Open source ecology, <http://opensourceecology.org>.
- [35] The hole-in-the-wall experiment, <http://www.hole-in-the-wall.com/beginnings.html>.
- [36] B.L. Gray, S.D. Collins, and R.L. Smith. Interlocking mechanical and fluidic interconnections for microfluidic circuit boards. *Sensors and Actuators A: Physical*, 112(1):18 – 24, 2004.
- [37] BL Gray, D Jaeggi, NJ Mourlas, BP Van Drienuizen, KR Williams, NI Maluf, and GTA Kovacs. Novel interconnection technologies for integrated microfluidic systems. *Sensors and Actuators A: Physical*, 77(1):57–65, 1999.
- [38] George M. Whitesides. Solving problems. *Lab Chip*, 10:2317–2318, 2010.
- [39] Philip N Duncan, Transon V Nguyen, and Elliot E Hui. Pneumatic oscillator circuits for timing and control of integrated microfluidics. *Proceedings of the National Academy of Sciences*, 110(45):18104–18109, 2013.
- [40] Erik C Jensen, William H Grover, and Richard A Mathies. Micropneumatic digital logic structures for integrated microdevice computation and control. *Journal of Microelectromechanical Systems*, 16(6):1378–1385, 2007.
- [41] William H Grover, Robin H C Ivester, Erik C Jensen, and Richard A Mathies. Development and multiplexed control of latching pneumatic valves using microfluidic logical structures. *Lab Chip*, 6(5):623–31, May 2006.

- [42] Carl K Fredrickson and Z Hugh Fan. Macro-to-micro interfaces for microfluidic devices. *Lab on a Chip*, 4(6):526–533, 2004.
- [43] Gerardo Perozziello, Frederik Bundgaard, and Oliver Geschke. Fluidic interconnections for microfluidic systems: A new integrated fluidic interconnection allowing plug&play functionality. *Sensors and Actuators B: Chemical*, 130(2):947 – 953, 2008.