UNIVERSITY OF CALIFORNIA, SAN DIEGO

"Gizmo: Mobile Mesh Network and Sensor Platform"

A thesis submitted in partial satisfaction of the requirements of degree Master of Science

in

Electrical Engineering (Electronic Circuits and Systems)

by

Javier Rodriguez Molina

Committee in charge:

Professor Lawrence E. Larson, Chair
Professor Pankaj Das
Professor William S. Hodgkiss

2008

The Thesis of Javier Rodriguez Molina is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2008

Dedication

After exactly twenty years of long, exciting, scary, hard, sad and fun times in elementary, middle, and high school, community college, university and what not, I am taking this opportunity to thank the two people that made all this possible. Thank you Mom and thank you Dad for allowing me to have an amazing education.

Epigraph

“Tutto e’ piu semplice di quanto sembri.
Dipende dal tuo punto di vista.”

David Rodriguez Molina

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Acc: ……. Accelerometer

ADC: …… Analog to Digital Converter

AVR: …… Alf (Egil Bogen) and Vegard (Wollan) 's Risc processor

BOM: ……... Bill of Materials

BT: ……... Bluetooth

Calit2: …….. California Institute of Communication and Information Technology

COM: ……... Communication

DAC: …… Digital to Analog Converter

DRC: …… Design Rule Check

ESC: ……. Electric Speed Control

GCC: …… Gnu Compiler Collection

GND: ……... Ground

GPS: ……. Global Positioning System

GROBI: …… Gizmo Remote Operated Bluetooth Interface

GUI: ……. Graphical User Interface

IP: …… Internet Protocol

LCD: …… Liquid Crystal Display

MAC: ……... Media Access Control

MPH: ……... Miles per Hour

PCB: …… Printed Circuit Board

PCI: ……. Peripheral Component Interconnect

PWM: …….. Pulse Width Modulation

PyS60: ……. Python for S60

RC: …….. Radio Controlled

RP-TNC: ……. Reverse Polarity Threaded Neill Concelman

SBC: …… Single Board Computer

SIP: …….. Session Initiation Protocol

SSH: …… Secure Shell

SSID: ……... Service Set Identifier

TCP: ……. Transmission Control Protocol

UCSD: ……. University of California, San Diego

UDP: …… User Datagram Protocol

USART: ……... Universal Synchronous Asynchronous Receiver Transmitter

USB: …… Universal Serial Bus

VB: …….. Visual Basic

VGA: ……... Video Graphics Array

VoIP: …… Voice over Internet Protocol

Acknowledgements

I would like to acknowledge Professor Lawrence Larson for his support as a chair of my committee and Donald Kimball (principle engineer at UCSD Calit2) for mentoring me throughout the entire project. After many prototypes and hard work, the Gizmo project has finally got the attention from several media (both press and television) around the world and from quite a few industry representatives which have started (or about to start) collaboration with UCSD academia and Calit2.

I would also like to thank the Calit2 staff, particularly Daniel Johnson, Jeff Cuenco, and Anthony Nwokafor for helping and collaborating on developing the mechanical and network side of the Gizmo platform. Thank you guys I learned a lot from working with you.

Finally, I would like to thank the person that gave me the energy to sit down and put this thesis together, thank you Andrea. I made it thanks to you.

ABSTRACT OF THE THESIS

"Gizmo: Mobile Mesh Network and Sensor Platform"

by

Javier Rodriguez Molina

Master of Science in Electrical Engineering (Electronic Circuits & Systems)

University of California, San Diego, 2008

Professor Lawrence E. Larson, Chair

Deploying wireless networks can be a really complicated and time consuming procedure. The choice of the access point locations, the selection of a frequency band with minimal interference, and the operation of multiple sensors using a single platform are few of the key factors which will determine the efficiency of the deployment. The Gizmo truck invention proposes an effective and robust solution to the problem. Gizmo has taken the Calmesh Network project developed at Calit2 (California Institute of Telecommunication and Information Technology) together with the *"always best*

connected approach" in mind and joined them in a mobile platform which is built using solely off-the-shelf components.

The Gizmo project has led to a versatile mobile platform capable to create instantaneous mesh networks using a variety of different radios and technology (i.e. 802.11 a, b or g, 900Mhz radios, Zigbee, …) or act as a client to any preexisting network, stream audio and video feeds, collect important data using an array of different sensors which can be swapped depending on the needs, and navigate through areas using GPS way points and distance sensors for object avoidance. The main characteristic of Gizmo is its high versatility and re-programmability. Gizmo presents a great platform for research and development. The applications in which Gizmo can be used are numerous (medical, military, educational, outreach, self guided tours, surveillance …), all it is needed to do is to swap the new sensors and download the desired scripts on the single board computer (SBC) and/or microcontroller board.

Introduction

During emergency or disaster situation, the presence of a reliable and stable wireless network which covers the entire area of interest could help the first responders in resolving and reestablishing the normal conditions in an easier, faster and more successful way. It is often the case that several different government or public agencies are invoked to respond to an immediate situation. The problem arises when these different entities do not use the same communication systems and therefore they cannot communicate or collaborate between themselves. This unfortunate lack of collaboration will hurt and slow down the process of resolving the current events.

At the University of California, San Diego (UCSD), a new research institute has arisen in the last decade which is putting a lot of time and effort in coming up with practical and easily deployable solutions that could help during these states of emergency. Calit2 (California Institute of Telecommunications and Information Technology) is a two campuses institution, present in both UC Irvine and UCSD. Larry Smarr, director of Calit2, once said: “Calit2 represents a new mechanism to address large-scale societal issues by bringing together multidisciplinary teams of the best minds”. Calit2 is a multidisciplinary institution which includes artists, visual arts, mechanical engineers, biologist, archeologist, historians, electrical engineers, nanotechnology engineers, and many more. All of these widely different disciplines working together and complementing each other with a unique scope; create, invent, and provide solutions to current social issues.

In this kind of environment, in particular in the Circuit Lab (http://circuitslab.calit2.net), is where I started the Gizmo project under the direction of Donald Kimball. The first Gizmo idea came across as an easier and autonomous way to deploy the instantaneous and self configuring ad-hoc wireless Calmesh network (http://calmesh.calit2.net/) developed also at Calit2. Once the first prototype was created, a tremendous number of ideas to try and directions to pursue opened up. Cameras, microphones, pollution sensors, controls systems, microcontroller boards, VoIP, GPS, sonar and laser distance sensors, network sniffing algorithms, cell phone communication technology, Bluetooth, Wi-Fi 802.11 a-b-g, Zigbee, 900MHz and 700 MHz radios are only a few of the possibilities that the Gizmo platform could be set up to use.

Now, starting the project was a relatively easy task, but putting an end and finishing the project is a more complicated job to do. The Gizmo project does not have a clear stop or end point. The project can keep going on and on implementing new and diverse technologies, adopting and incorporating new hardware, and being deployed for novel applications that I might not even be aware of. Gizmo presents a great and unique research platform that can be used by anyone for their own research using their own applications. In this Master Thesis, I will give a detailed overview of evolution of the Gizmo project through its four prototypes. I will explain the hardware and software of each of the models, the interfaces that have been developed up to the end of the spring quarter 2008, the pros and cons of each system, and finally the limitations which motivated the design process and construction of each of the next generations of Gizmo. Also, a small section about outreach programs such as ECE 191 project proposals and

Preuss high school senior internships will be included together with a future work and goals for this project.

Chapter 1: I-Hope-It-Works Prototype

The first Gizmo prototype was created with a particular scope in mind, deploy wireless Calmesh nodes in the field during drills and be able to remotely move them if necessary. From experience, the Calit2 Network team has noticed that fire fighters trucks are excellent signal destroyers. If a truck parks between the line of sight of two nodes, or it partially shadows one access point, the connection is lost or seriously degraded. This can cause two main problems, the network could get split it in two separate networks or all the network traffic could be redirected through a unique node causing overload of data (increasing delay and package lost) since one of the paths has been blocked by the truck.

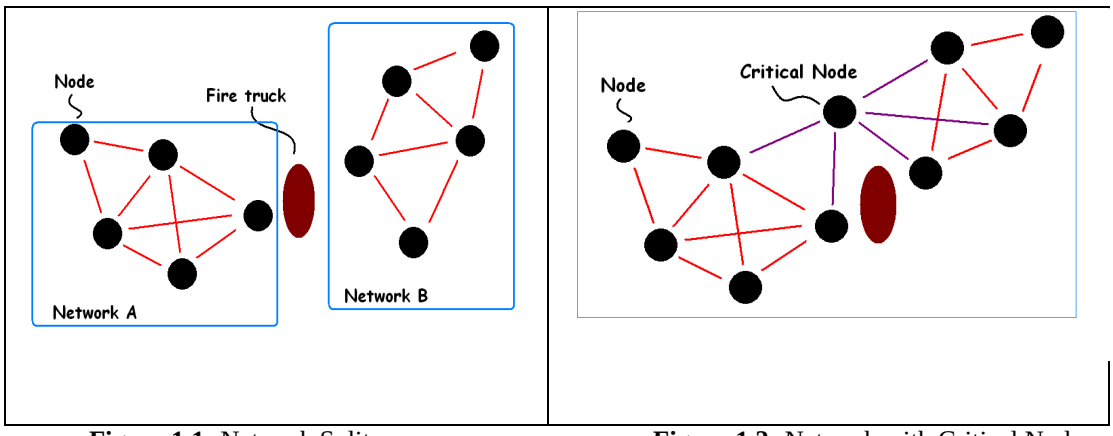

Figure 1.1: Network Split

Figure 1.2: Network with Critical Node

The goal with the first prototype was to create a Calmesh node on wheels. If I firefighter truck parked in the way, we could have it relocated the node without having to actually be physically next to the node. Nevertheless, Gizmo v.1 came to be more than a node on wheels, it became the beginning of a great research platform.

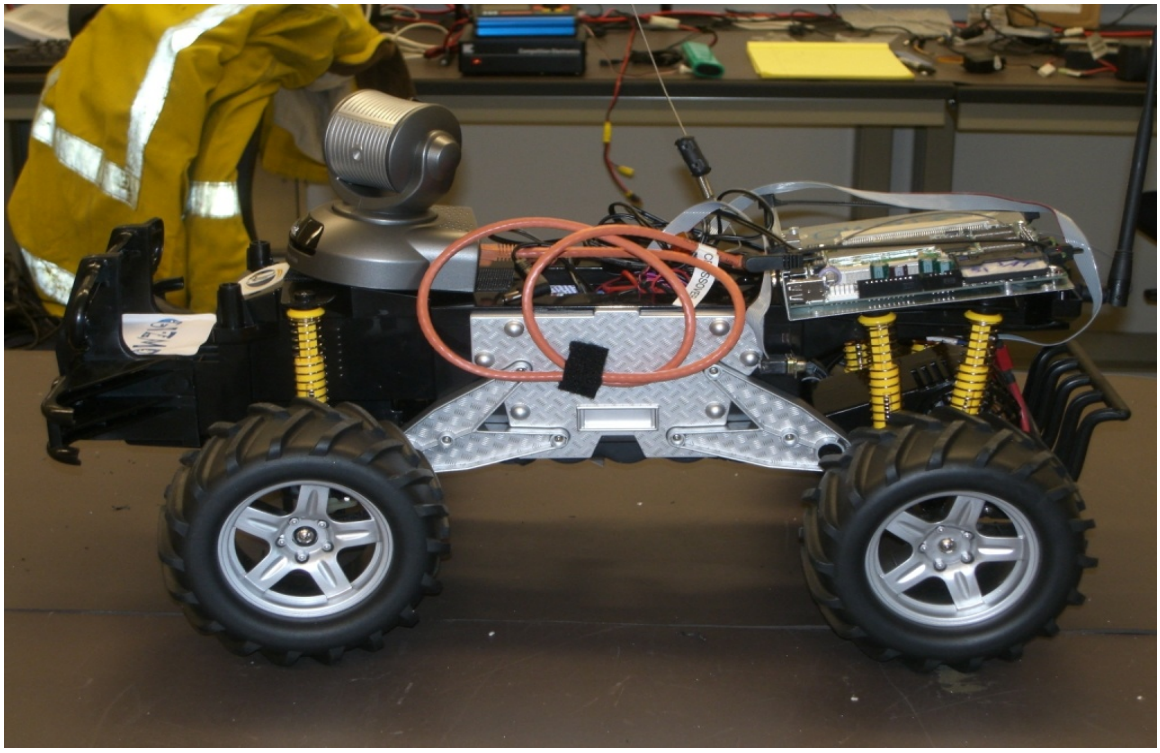

Figure 1.3: Gizmo v.1 Platform

1.1 Hardware

This first prototype was built on top of a cheap RC vehicle. It used a 3 channel 75MHz transmitter, one servo for steering, a +12V electrical speed controller (ESC), one motor, one +12V 10Ah battery, one Ethernet camera, a single board computer (SBC) with Calmesh network software running as a client, a mini-PCI Wi-Fi card, a mini serial servo controller, a custom made override circuitry and finally an 5 amp RC switch to select between the 75MHz controls or the webpage controls. A block diagram and a bill of materials are shown below:

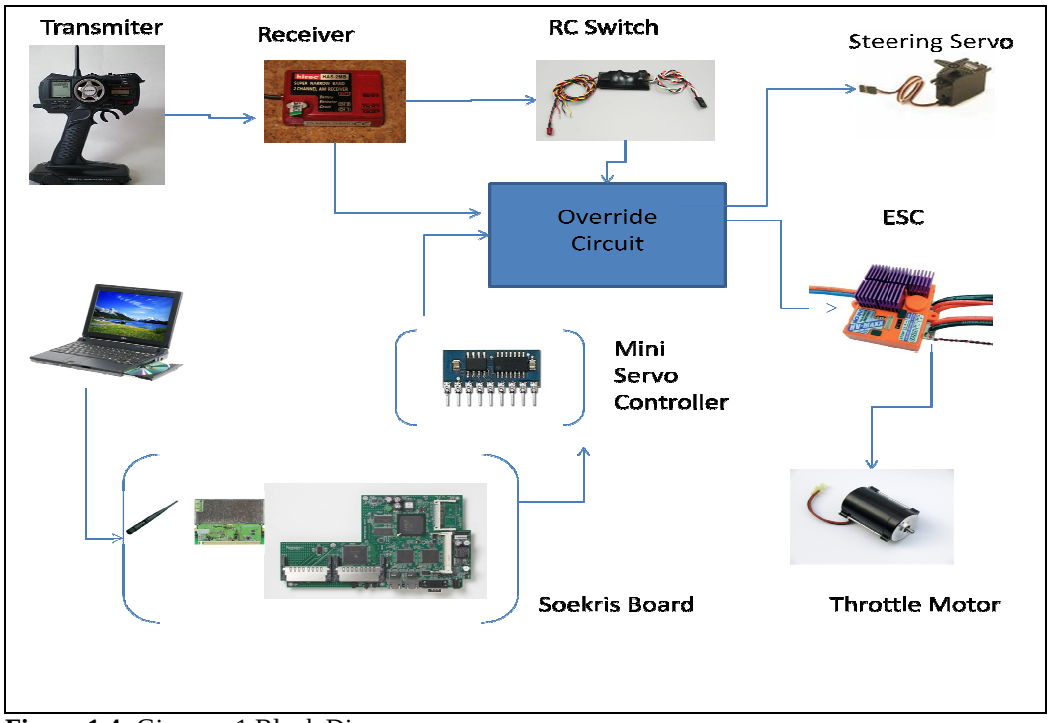

Figure 1.4: Gizmo v.1 Block Diagram

Table 1.1: Gizmo v.1 List of Materials

Component	Description
Chassis	Hot Wheels RC - Ford F-150 Off Road RC Truck from allaboardtoys.com
Receiver/Transmitter	Included with the chassis package
ESC	Novak High voltage brushless ESC from teamnovak.com
SBC	Soekris Engineering net4521 from soekris.com
Compact Flash	Scan 500Mb
RC Switch	Ram Simple R/C Switch 5A from towerhobbies.com
Mini Serial Servo Controller	Micro Serial Servo Controller from parallax.com
Wi-Fi Card	2511 MP PLUS 100mW: 802.11b mini PCI Card from netgate.com
Antenna	2.4 / 4.9 / 5.8 GHz Tri Band Rubber Duck RP-TNC from netgate.com
IP camera	DCS-6620G from dlink.com
Motor	Included in the chassis package
Battery	12V 10000mAh NiMh, batteryspace.com

The override circuitry was built using an *IC 8-CH MUX/DEMUX 16-DIP* Texas Instrument (TI) chip (link http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=296-3828-5-ND)

Table 1.2: Mux Truth Table

FUNCTION TABLE

INPUTS				ON CHANNEL
INH	C	B	A	
L	L	L	L	Y0
L	L	L	H	Y1
L	L	H	L	Y2
L	L	H	H	Y3
L	H	L	L	Y4
L	H	L	H	Y5
L	H	H	L	Y6
L	H	H	H	Y7
H	X	X	X	None

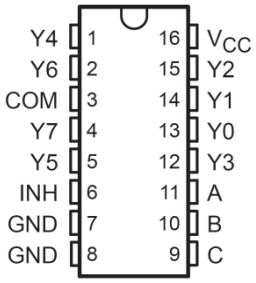

Figure 1.5: Mux Pin Diagram

The third channel of the transmitter/receiver system was used as the override signal. The PWM coming from the receiver would be fed to the RC switch which will transform the incoming square wave (20 ms period and 1 to 2 ms duty cycle) to a high or low signal based on the length of the pulse. The output of the switch will finally be connected to the selection bit of the de-multiplexer.

Table 1.3: PWM Table

Duty Cycle	RC Switch Output
1 ms < x < 1.5 ms	LOW
2 ms > x > 1.5 ms	HIGH

Here below is the complete schematic of the system and a bill of materials (BOM):

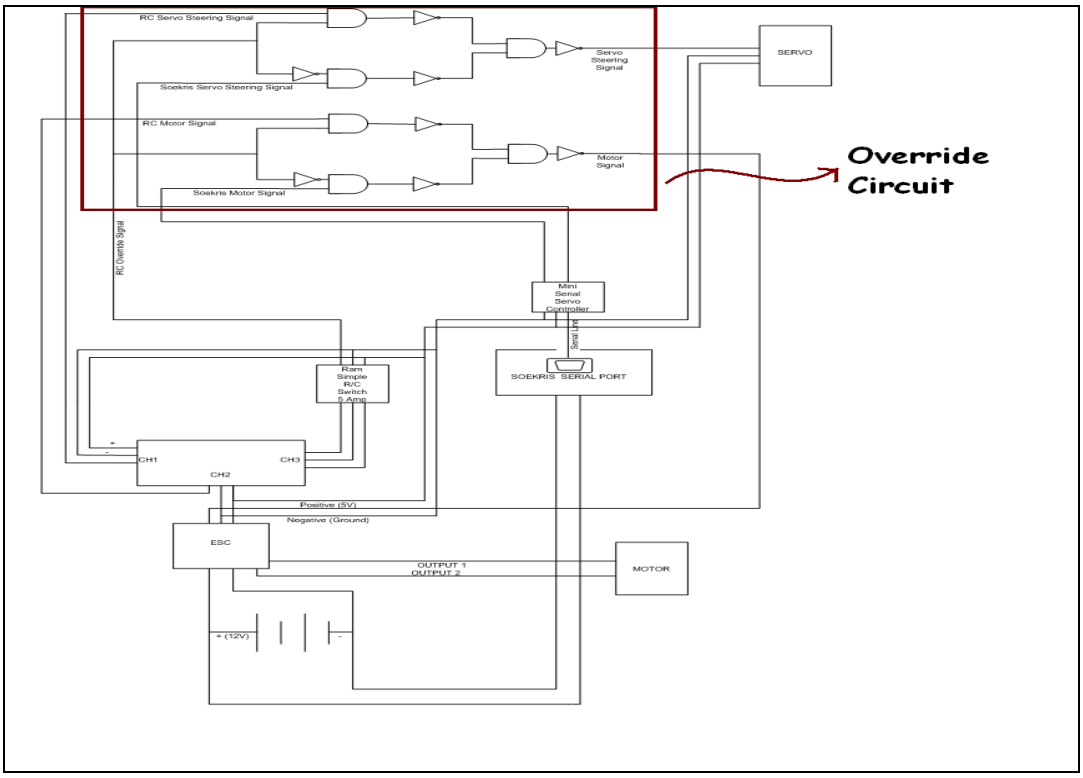

Figure 1.6: Gizmo v.1 Schematic

1.2 Software

In this first version of Gizmo the software side was not too involved. The platform was running the Calmesh spanning tree software (refer to the Calmesh project website for more information) which allowed it to connect as a client to the UCSD open wireless network or to the Calmesh ad-hoc Network. It is important to remember that if the network is a third party network (i.e the network is not managed by you) then it is necessary to register the MAC address of the mini PCI card with your network provider in order for the robot to be able to connect to the network.

Once the Gizmo platform gets an IP address assigned, port forwarding is used to link the video feed coming from the web camera to port 81 on that IP. With the help of

Anthony Nwokafor, a command webpage was created and stored in the gizmo platform. Through this webpage we were able to type commands to the Basic Stamp which would control the steering and the throttle. This typed data would be grabbed by a *.php* program running on the background and saved on a text file. Another script stored on Gizmo would check continuously for new data on the text file and output the commands to the serial port which is connected to the Basic Stamp. From here the commands would go to the mini servo controller which output the right Pulse Width Modulation (PWM) signal to the servo and ESC. Shown below is the flow chart of the software system:

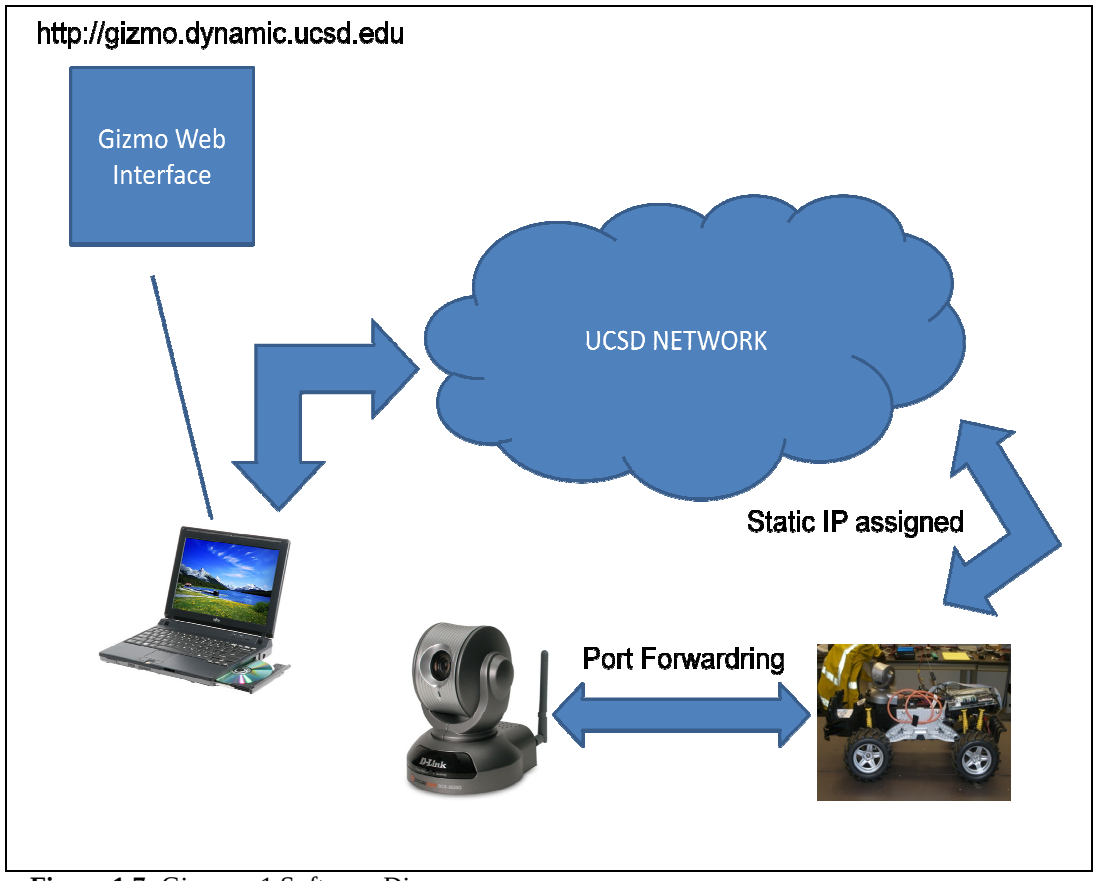

Figure 1.7: Gizmo v.1 Software Diagram

1.3 Platform Evaluation

This first Gizmo platform has overall fewer pros that cons. From one side, it was a really cheap platform, it was easy to debug since it had little components and lines of code running in the background, and provided video feeds and test data that helped improving the Calmesh Network protocol. From the other side, Gizmo v.1 was really limited in its functionality and it presented a lot of problems at both the mechanical and software level. The override system for instance, was really noisy and therefore it never worked properly. The source of the noise was introduced in the circuit by the receiver. If the transmitter was not in a 5 meter range from the receiver, the noise in the receiver line would have reached the RC switch causing the override signal channel to go High and Low randomly. From a software point of view, the system was really slow and impractical to control over the web interface. The 133MHz processor present on the SBC was not fast enough to transmit a fluent video feed. The resulting transmission feed was less than 10 frames per second with random losses in connectivity. Lastly, the webpage command interface was impractical for two main reasons. First, the continuous association and disassociation of the Gizmo client to the network was causing a delay between consecutive commands (i.e. after a “Go” signal the robot would have hit the wall before the “Stop” signal arrived). Second, the protocol to transmit commands was too slow and included too many steps. First we typed the command, and then the *.php* code stored the command on a text file, a secondary script checked for new commands in the text file and finally it transmitted the command to the servo controller.

In conclusion, despite the communication and controls problems, the Gizmo v.1 platform achieved two extremely important goals. First, it proved the concept that it was possible to have mobile clients on a network transmitting video feeds to a static IP address. Second, the Gizmo platform attracted a lot of interest from fellow researchers who saw the potential of this new platform that was created in only two weeks with a modest budget of $400. Researchers provided a funding stream to continue on with the project allowing me to develop and designed the second version of Gizmo Platform.

Chapter 2: Cell Phone Enabled Platform

The second version of Gizmo was a complete different and improved platform with respect to the first version. The new design mainly focused on improving three aspects; the chassis, the user interface and the network capabilities.

The first component to be changed on the new prototype was the framework. The first version was build on top of a $40 RC truck. This platform presented several hardware limitations; it didn't have enough torque to carry the weight of the battery and electronics at a decent speed and its steering angle was really small (it was only able to make wide turns). Therefore, I decided to integrate the second version on the TRAXXAS E-MAXX RC truck platform which was faster, more robust, and with greater steering radius (link http://www.traxxas.com/products/electric/emaxx3905/trx_emaxx3905.htm). These trucks cost about $360-$400 and can go up to a speed of 30 mph. The transmitter/receiver system was a three channel device working at a frequency of 25MHz.

The second component to modify was the user interface. Typing commands on a webpage was definitely not the right way to go and having three programs running to complete the command transaction was too slow to be practical. We needed a different wireless communication protocol to transfer small data packages to the robot and a more intuitive and user friendly device that could create commands faster that we can type. After doing some research and sharing ideas with other researchers, the wireless protocol that best fitted this project was Bluetooth since it is an inexpensive solution which allows communication between mobile platforms (for more basic information on Bluetooth protocol please refer to the following link http://en.wikipedia.org/wiki/Bluetooth). For the

end user device, two different pieces of hardware where suited for this task; joysticks and cell phones. The gaming and cell phone industry are in continuous evolution and expansion. In our current society, it is really common to own a cell phone or game console. New generations are growing up surrounded by these new technologies and using a joystick or a phone is becoming really intuitive. In this version of gizmo we were using the Nokia 6600 cellphone and the USB version of the PlayStation joystick.

Finally, the third aspect that needed improvement was the video streaming over the network. In order to achieve this improvement a Linksys IP camera was selected to replace the DLink one. The Linksys camera came with a free surveillance utility which allowed acquiring the video feed without having to use the web-browser. Also, a new SBC with a faster processor was used for this Gizmo v.2 prototype.

2.1 Hardware

The hardware for this prototype was essentially an upgraded version of the first Gizmo platform with the addition of an accelerometer, a custom microcontroller board, and Bluetooth (BT) module. The number of motors driving the truck increased to two. The processor speed on the new SBC board doubled to 233 MHz. This WRAP 2E board also came with two mini-PCI slots instead of one which allowed one Wi-Fi card to be used for networking purposes and one to be use for signal strength detection. The IP camera was also upgraded in order to overcome some software limitations.

The microcontroller board was design during the ECE 191 design class in which researcher Paul Blair and I co-mentored the GROBI project. More information on the educational outreach projects can be found in chapter six. The Atmel AVR family provided the perfect solution for our needs. Some of the features offered by the microcontroller Atmega164p are two serial ports (used to communicate with the BT module and the SBC board), eight 8-bit ADC, and six PWM. In this case only three ADCs and two PWMs channels were used to query the accelerometer and to control the truck. The Atmega164p provided a lot of room for expansion. More information on the chip can be found at http://www.atmel.com/dyn/Products/product_card.asp?part_id=3887

The custom board was designed using free PCB layout software called Eagle (a zip file can be download from the internet together with a tutorial at http://www.cadsoft.de/download.htm). The environment comes with basic components libraries installed and more specific ones that can be downloaded at their site.

The manufacturer used to print the board was PCB Express (link http://www.pcbexpress.com). This company supports multilayer designs and fast shipping but it is a little on the pricey side. In the PCB Express website you can download a library specifically for the Eagle environment which needs to be run on your design in order to create the files necessary for the manufacturing of your board. In Eagle there are two main files that the user will produce; a schematic file .sch and a board layout file .brd. The library contains scripts which will extract information from the .brd file such as via locations, drill sizes, copper layout, solder masks, and so on. A good

tutorial on how to prepare the files for manufacturer can be found here: http://www.pcbexpress.com/downloads/EAGLE%20Convert-Sunstone%20Protos.pdf

There are a couple of mistakes in the manual that could create some confusion. On page two of the tutorial, the files extensions listed do not match with the files created by the library. The .sol, .cmp, and .dri actually correspond to the files *.top*, *.bot*, and *.drl*. A cheaper alternative to PCB Express would be BatchPCB (link http://www.batchpcb.com). This company is supported by the same people that run Sparkfun (link http://www.sparkfun.com). One feature to keep in mind about this manufacturer is that the minimum distance between traces is fixed to eight mils. This factor could start to become a problem when designing 2-layer board with chips with a higher pin density while performing the DRC (i.e Atmega2560). Also, it can take up to a month and a half to receive your board back.

Here below are the updated block diagram and new BOM for Gizmo v.2 and the schematic and board layout for the microcontroller:

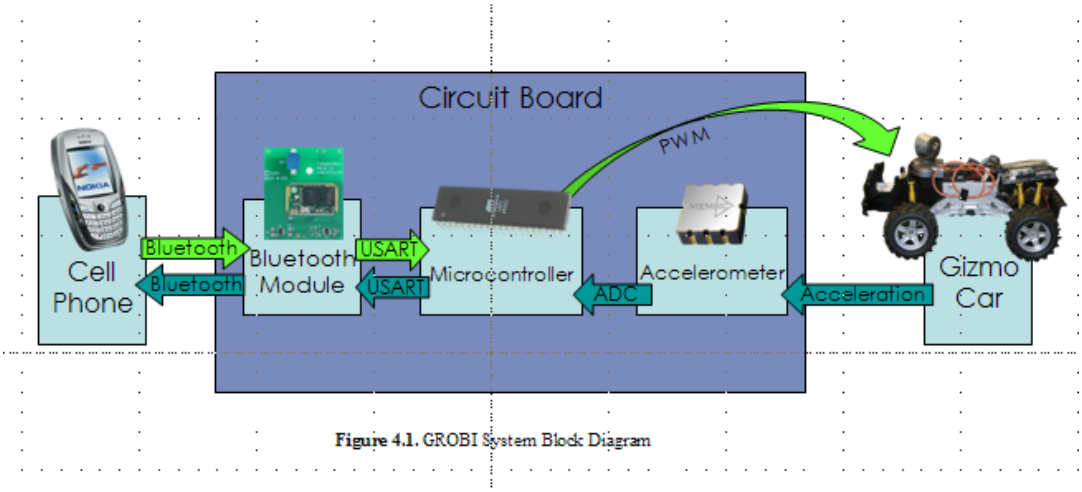

Figure 2.1: Gizmo v.2 Block Diagram, courtesy of GROBI project

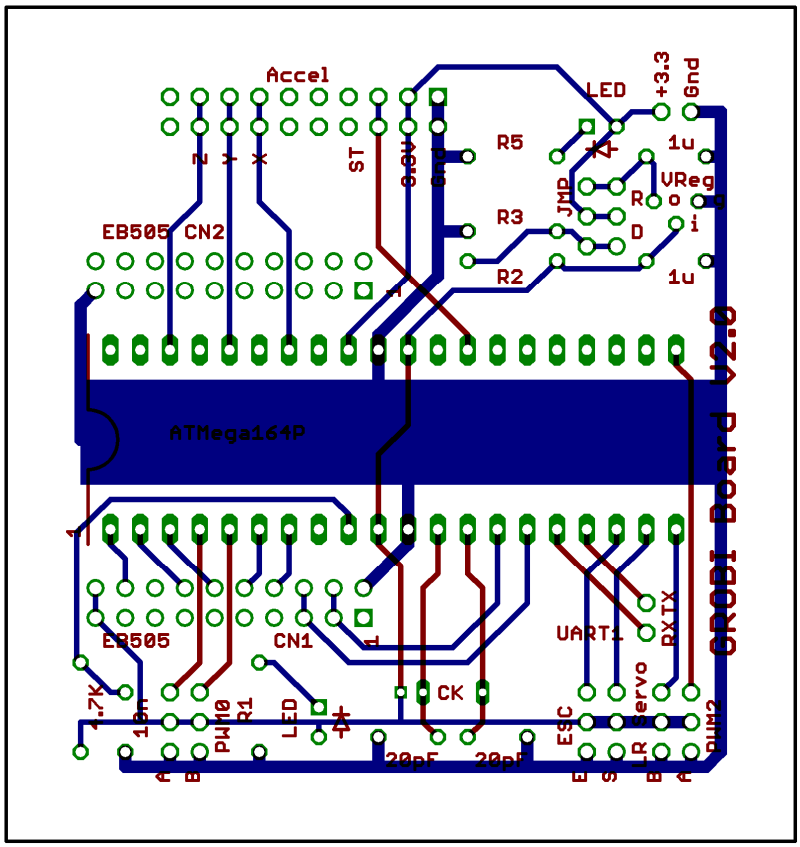

Figure 2.2: Microcontroller Circuit Board Layout, courtesy of GROBI project

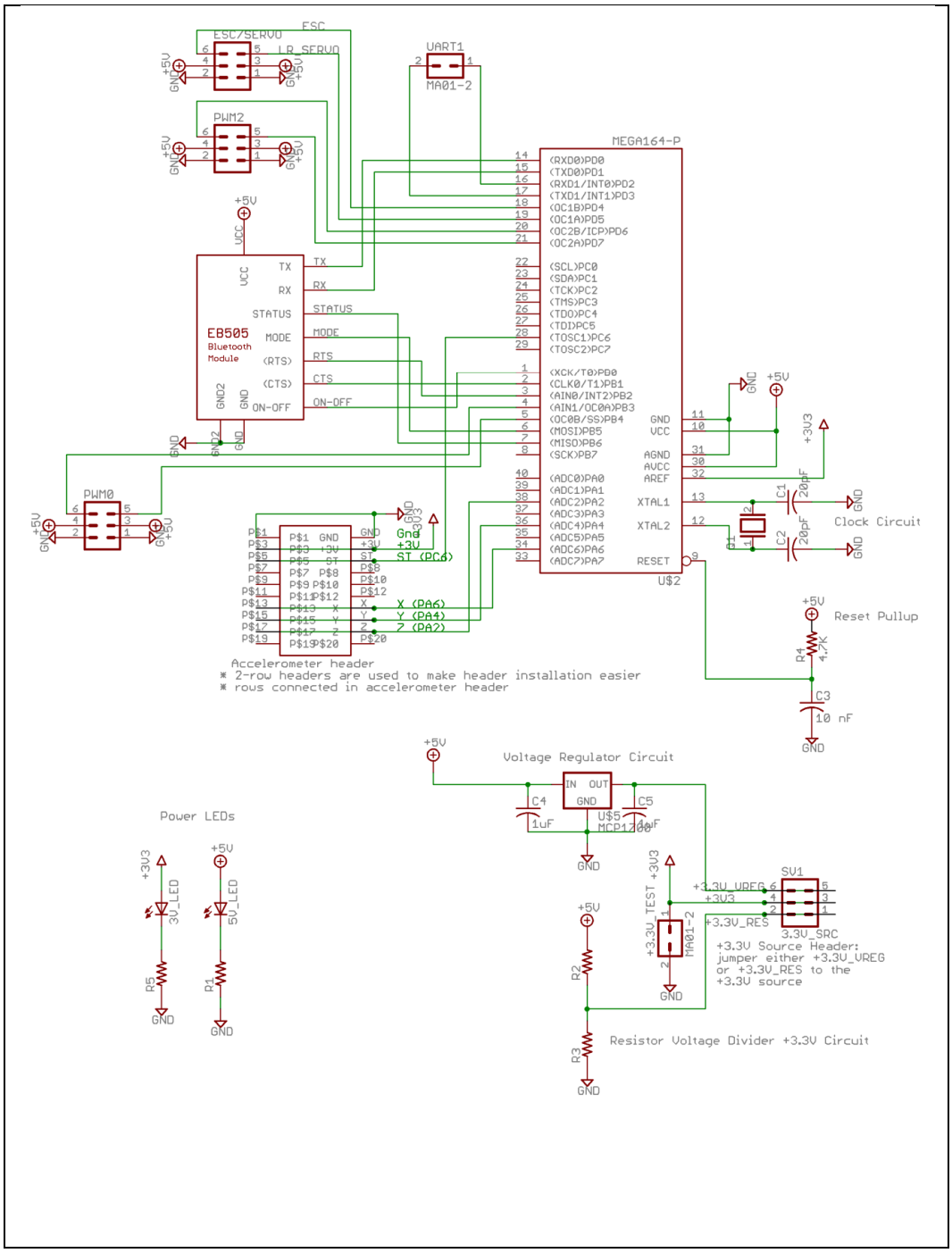

Figure 2.3: Microcontroller Board Schematic, courtesy of GROBI project.

Table 2.1: Gizmo v.2 List of Materials

Component	Description
Chassis	E-MAXX TRAXXAS from traxxas.com
Receiver/Transmitter	Included with the chassis package
ESC	Novak model 3014, included in package
SBC	WRAP 2E from mini-box.com
Compact Flash	Scan 500Mb
BT Module	eb501 from ieng7.com
Cell phone	Nokia 6600
Wi-Fi Card	2511 MP PLUS 100mW: 802.11b mini PCI Card from netgate.com
Antenna	2.4 / 4.9 / 5.8 GHz Tri Band Rubber Duck RP-TNC from netgate.com
IP camera	Linksys WVC200 from linksys.com
Motor	Both included in the chassis package
Microcontroller	Atmega164p from atmel.com
Accelerometer	Two axis accelerometer by memsic.com
Battery	2 x 7.2V 4200mAh NiMh in series, Batteryspace.com

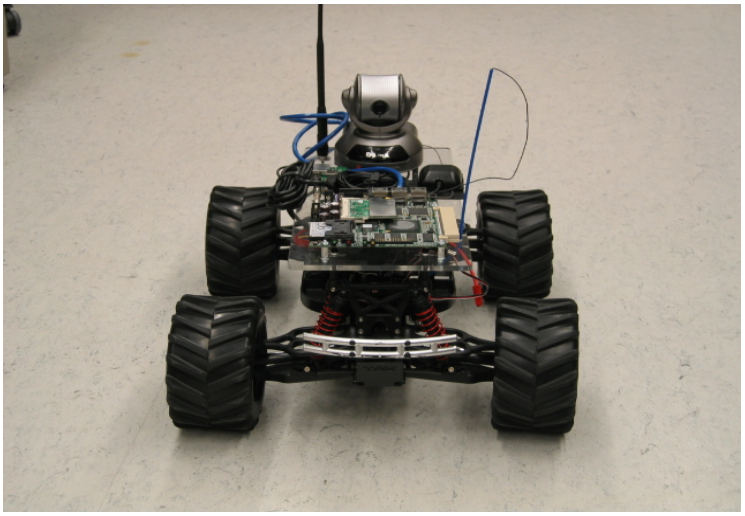

Figure 2.4: Gizmo v.2 Platform

2.2 Software

2.2.1 AVR

The Atmel AVR microcontroller family is programmed in C language. The Atmel STK500 development board (link http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=2735) was used for testing, debugging, and loading the software onto the atmega164p microcontroller before installing the chip on the board mention earlier in this chapter. There are several compilers available for the Atmel series that can be downloaded online. In this project, the WinAVR open source software development tools were used. It includes the GCC compiler for C. The latest version of WinAVR can be found at http://winavr.sourceforge.net/download.html. Programming Atmel chips with the WinAVR environment is really easy. For each Atmel chip there is a correspondent header file with all the pointers to every single one of the registers. These header files can be found on the installation path in the folder …/avr/includes/avr/.

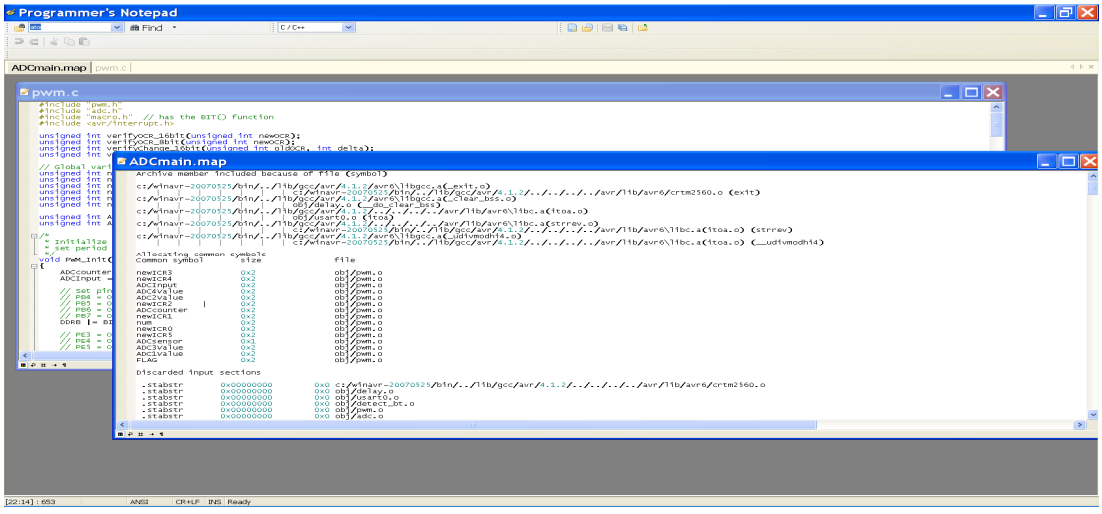

Figure 2.5: WinAVR Software

In order to load the *.hex* file generated by the WinAVR compiler into the microcontroller the AVR Studio 4 is used. This application also allows you to control and set all the parameters and fuses of the STK500. More information on the AVR Studio and a link to the download can be found at http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725.

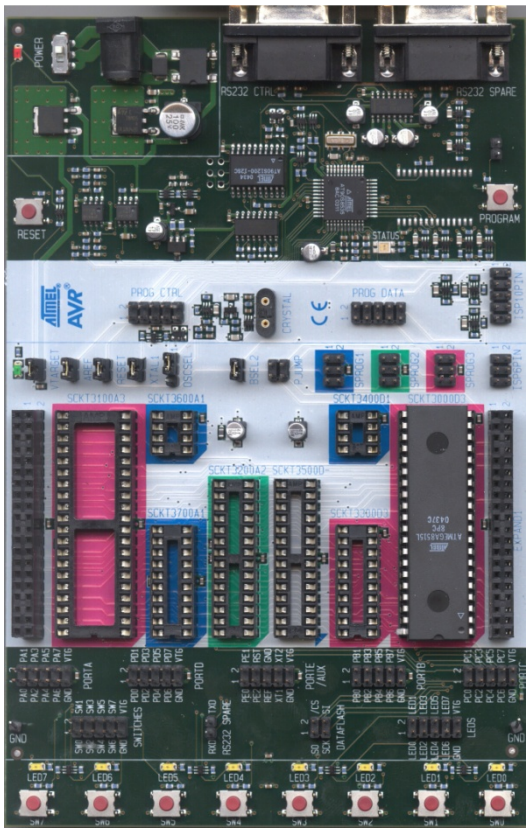

Figure 2.6: STK500 Board

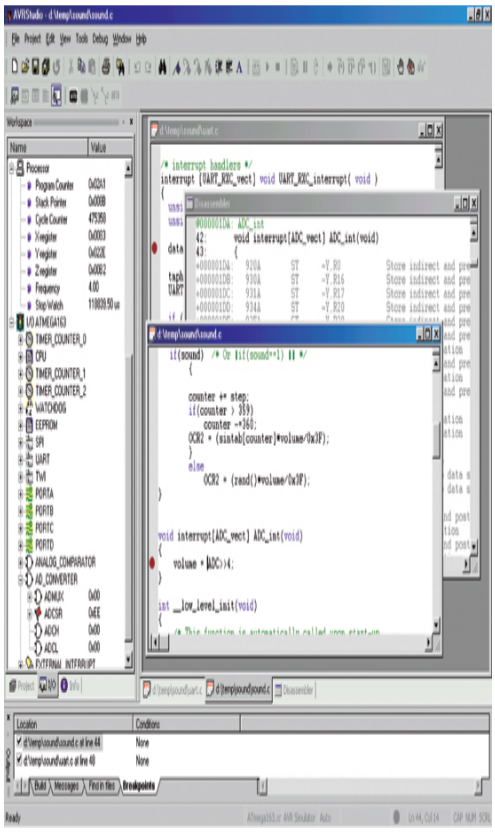

Figure 2.7: Studio4 Software

2.2.2 PyS60

The Python S60 software allows the use of the open source Python programming language for Symbian O.S. cell phones with S60 platform installed. The S60 platform includes many libraries which allow the use of all cell phones functionalities. Text messaging, taking a picture, browses the internet, connect to a Bluetooth device and so on, can be easily done by importing a simple module into your program. In this second version of Gizmo the BT, keyboard and graphical interface module were used to connect and controlled the truck. Commands were created by pushing buttons on the key pad and transmitted over Bluetooth.

Table 2.2: PyS60 Commands

COMMAND	ACTION
Button 2 (or Joystick Up)	Forward (or Speed up)
Button 6 (or Joystick Right)	Steering Right
Button 8 (or Joystick Down)	Reverse (or Slow down)
Button 4 (or Joystick Left)	Steering Left
Button 5 (or Joystick Middle)	STOP

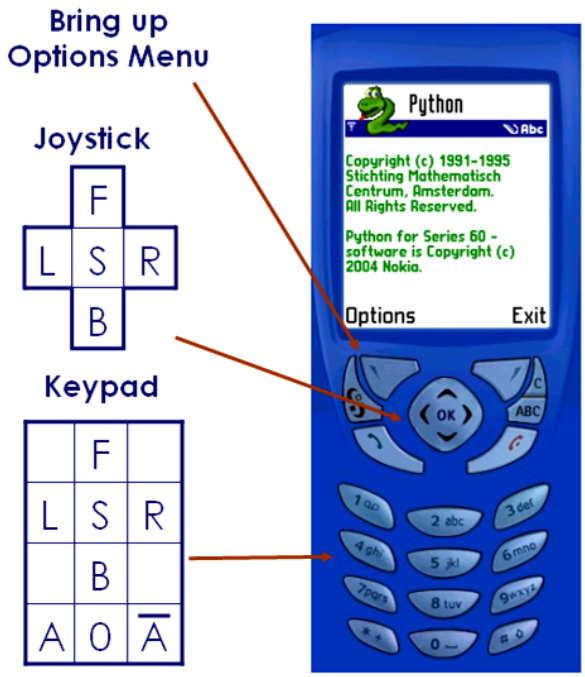

Figure 2.8: Keypad Commands, Courtesy of GROBI project

For more information on how to set up pyS60 on a Nokia phone and how to write a pyS60 program refer to the GROBI project report. Also, a great source of tutorials and examples on pyS60 can be found online at http://www.mobilenin.com/pys60/menu.htm.

2.2.3 Visual Basic

Obviously the BT capabilities are not limited to cell phone devices. Almost every laptop has integrated BT modules. Bluetooth ports are considered wireless serial port and will appear as an available COM port in your computer.

A secondary user interface has been developed in parallel with the cell phone one. Visual Basic (VB) in Visual Studio environment was used as the programming language. VB is an extremely easy language to learn and to use. All the user interface objects are provided as drag and drop modules that can be configured and integrated with no trouble into your project. The VB code had two main modules; a communication module and an input module. The first module was responsible to open a BT port and to establish a serial connection to send data. The second module instead, was in charge of periodically pulling data from a USB joystick and to create the corresponding command to be transmitted. A serial port object can be created by dragging and dropping the existing serial object or by directly instantiating it in your code:

```
////////////////////////////////////////////////////////////////

//Declare

Dim WithEvents serialPort As New IO.Ports.SerialPort

//Configure
Try
        With serialPort
          .PortName = SerialPortComboBox.Text
          .BaudRate = SerialSpeedComboBox.Text
          .DataBits = DatabitComboBox.Text

          If ParityComboBox.Text.Equals("none") Then
            .Parity = IO.Ports.Parity.None
          ElseIf ParityComboBox.Text.Equals("odd") Then
            .Parity = IO.Ports.Parity.Odd
          ElseIf ParityComboBox.Text.Equals("even") Then
            .Parity = IO.Ports.Parity.Even
          ElseIf ParityComboBox.Text.Equals("mark") Then
            .Parity = IO.Ports.Parity.Mark
          ElseIf ParityComboBox.Text.Equals("space") Then
            .Parity = IO.Ports.Parity.Space
          End If
          If StopBitComboBox.Text.Equals("1") Then
            .StopBits = IO.Ports.StopBits.One
          ElseIf StopBitComboBox.Text.Equals("1.5") Then
            .StopBits = IO.Ports.StopBits.OnePointFive
          ElseIf StopBitComboBox.Text.Equals("2") Then
            .StopBits = IO.Ports.StopBits.Two
          ElseIf StopBitComboBox.Text.Equals("0") Then
            .StopBits = IO.Ports.StopBits.None
          End If
        End With
        serialPort.Open()
```

```
            'SerialStatusTextBox.Text = serialPort.PortName & " connected"
            SerialConnectButton.Text = "DISCONNECT"
        Catch ex As Exception
            SerialConnectButton.Text = "CONNECT"
        End Try

//USE

serialPort.Write(message)

////////////////////////////////////////////////////////////////////
```

The joystick module used was the JOYNFOEX structure which imports the windows drivers for USB joysticks. More information about this module can be found at http://msdn.microsoft.com/en-us/library/ms709358(VS.85).aspx.

Table 2.3: Visual Basic Code

```
Option Strict Off
 Option Explicit On
 Module modInput

    'UPGRADE_WARNING: Structure JOYINFOEX may
require marshalling attributes to be passed as an argument in
this Declare statement. Click for more: 'ms-
help://MS.VSCC.v80/dv_commoner/local/redirect.htm?keyw
ord="C429C3A5-5D47-4CD9-8F51-74A1616405DC"'
    Public Declare Function joyGetPosEx Lib "winmm.dll"
(ByVal uJoyID As Integer, ByRef pji As JOYINFOEX) As
Integer
    'UPGRADE_WARNING: Structure JOYCAPS may
require marshalling attributes to be passed as an argument in
this Declare statement. Click for more: 'ms-
help://MS.VSCC.v80/dv_commoner/local/redirect.htm?keyw
ord="C429C3A5-5D47-4CD9-8F51-74A1616405DC"'
    Public Declare Function joyGetDevCapsA Lib
"winmm.dll" (ByVal uJoyID As Integer, ByRef pjc As
JOYCAPS, ByVal cjc As Integer) As Integer

    Public Structure JOYCAPS
        Dim wMid As Short
        Dim wPid As Short
        'UPGRADE_WARNING: Fixed-length string size must
fit in the buffer. Click for more: 'ms-
help://MS.VSCC.v80/dv_commoner/local/redirect.htm?keyw
ord="3C1E4426-0B80-443E-B943-0627CD55D48B"'
        <VBFixedString(32),
System.Runtime.InteropServices.MarshalAs(System.Runtim
e.InteropServices.UnmanagedType.ByValArray,
SizeConst:=32)> Public szPname() As Char
        Dim wXmin As Integer
        Dim wXmax As Integer
        Dim wYmin As Integer
        Dim wYmax As Integer
        Dim wZmin As Integer
        Dim wZmax As Integer
```

```
        Dim dwButtonNumber As Integer   'NUMBER OF BUTTONS
PRESSED
        Dim dwPOV As Integer
        Dim dwReserved1 As Integer
        Dim dwReserved2 As Integer
    End Structure

    Public JoyNum As Integer
    Public MYJOYEX As JOYINFOEX
    Public MYJOYCAPS As JOYCAPS
    Public CenterX As Integer
    Public CenterY As Integer
    Public JoyButtons(16) As Boolean
    Public JoyButtonsChanged(16) As Boolean

    Public AnalogLeftX As Integer
    Public AnalogLeftY As Integer

    Public AnalogRightX As Integer
    Public AnalogRightY As Integer


    Public Function StartJoystick(Optional ByVal
JoystickNumber As Integer = 0) As Boolean
        JoyNum = JoystickNumber
        If joyGetDevCapsA(JoyNum, MYJOYCAPS, 404) <> 0
Then 'Get joystick info
            StartJoystick = False
        Else
            MYJOYEX.dwSize = 64
            MYJOYEX.dwFlags = 255
            Call joyGetPosEx(JoyNum, MYJOYEX)
            CenterX = MYJOYEX.dwXpos
            CenterY = MYJOYEX.dwYpos
            StartJoystick = True
        End If
    End Function
```

Table 2.3: Continued

```
        Dim wNumButtons As Integer
        Dim wPeriodMin As Integer
        Dim wPeriodMax As Integer
        Dim wRmin As Integer
        Dim wRmax As Integer
        Dim wUmin As Integer
        Dim wUmax As Integer
        Dim wVmin As Integer
        Dim wVmax As Integer
        Dim wCaps As Integer
        Dim wMaxAxes As Integer
        Dim wNumAxes As Integer
        Dim wMaxButtons As Integer
        'UPGRADE_WARNING: Fixed-length string size must
fit in the buffer. Click for more: 'ms-
help://MS.VSCC.v80/dv_commoner/local/redirect.htm?keyw
ord="3C1E4426-0B80-443E-B943-0627CD55D48B"'
        <VBFixedString(32),
System.Runtime.InteropServices.MarshalAs(System.Runtime.
InteropServices.UnmanagedType.ByValArray,
SizeConst:=32)> Public szRegKey() As Char
        'UPGRADE_WARNING: Fixed-length string size must
fit in the buffer. Click for more: 'ms-
help://MS.VSCC.v80/dv_commoner/local/redirect.htm?keyw
ord="3C1E4426-0B80-443E-B943-0627CD55D48B"'
        <VBFixedString(260),
System.Runtime.InteropServices.MarshalAs(System.Runtime.
InteropServices.UnmanagedType.ByValArray,
SizeConst:=260)> Public szOEMVxD() As Char
    End Structure

    Public Structure JOYINFOEX
        Dim dwSize As Integer
        Dim dwFlags As Integer
        Dim dwXpos As Integer
        Dim dwYpos As Integer
        Dim dwZpos As Integer
        Dim dwRpos As Integer
        Dim dwUpos As Integer
        Dim dwVpos As Integer
        Dim dwButtons As Integer        '2^(button
number)+2^(next button)+...
```

```
Public Sub PollJoystick()
        Dim i As Integer
        Dim t As Integer
        Dim state As Boolean
        Dim changed As Boolean

        MYJOYEX.dwSize = 64
        MYJOYEX.dwFlags = 255

        ' Get the joystick information
        Call joyGetPosEx(JoyNum, MYJOYEX)

        t = MYJOYEX.dwButtons
        For i = 15 To 0 Step -1
            state = False
            'See if button i is being pressed
            If (2 ^ i) <= t Then
                t = t - (2 ^ i)
                state = True
            End If

            'If the button state has changed or stayed the same
            changed = (state <> JoyButtons(i + 1))

            JoyButtonsChanged(i + 1) = changed
            JoyButtons(i+1) = state

        Next i

        AnalogLeftX = MYJOYEX.dwXpos
        AnalogLeftY = MYJOYEX.dwYpos

        AnalogRightX = MYJOYEX.dwZpos
        AnalogRightY = MYJOYEX.dwRpos

    End Sub
End Module
```

2.2.4 Kismet Software

The last software upgrade made on Gizmo was the insertion of the Kismet open source software on the SBC. Kismet is an application able to record the signal strength of the access point present in the area, their MAC address and SSID. Furthermore, Kismet provides a GPS extension which permits the correlation of the data with its location and the further integration of the data into Google Earth map.

This project was also used as an ECE 191 senior design project which I co-mentored together with Donald Kimball (Principle Engineer at Calit2). More information about this project and its results can be found in chapter 6.

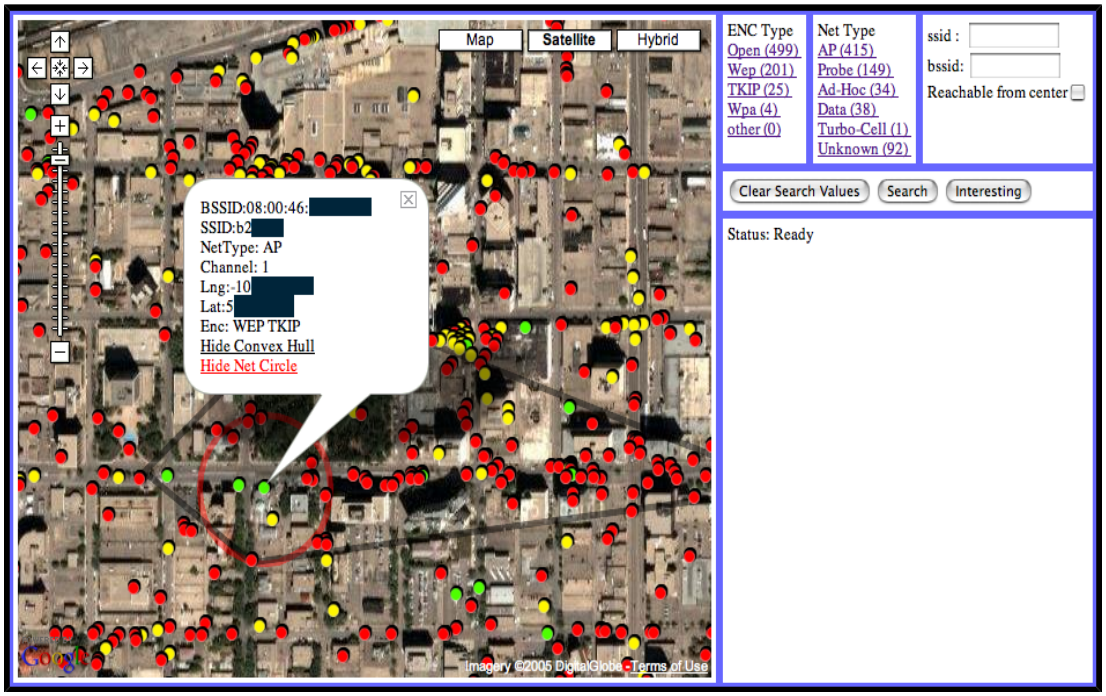

Figure 2.9: Kismet User Interface, courtesy of the Wi-Fi sniffer project.

2.3 Platform Evaluation

The Cell phone enabled Gizmo was a more robust and user friendly platform with respect to its previous version. Gizmo was able to be controlled by three different interfaces; a laptop using VB, a cell phone using python, and a RC controller. Its steering and throttle systems were highly improved which allowed a better control while driving and more torque power to go over obstacles such as curbs. Also, with the integration of a microcontroller board the possibilities of expansion were numerous. Unfortunately there

were still a few things to consider. Having two motors in the platform was a little heavy on the battery life. Furthermore, an expensive gear box was needed to reduce the maximum speed of the truck in order to make it safe to maneuver it with the different interfaces in small spaces (i.e. a laboratory). Even if we were able to control the truck with three different interfaces, only the BT solutions allowed the possibility to switch instantaneously between each other. In order to use the RC controller we had to physically change the servo connections from the microcontroller board to the receiver. In addition, the problem of short range connectivity was still there. In order to control the robot we needed to stay in a 5-10 meters radius with line of sight. Lastly, navigation wise, front steering only was a limited solution. For autonomous navigation purpuses, whether it is based on computer vision or distance sensors, it is extremely important that the robot is able to turn around on a small radius or even on the spot.

Chapter 3: Wi-Fi Enabled, 2-Way Audio/Video

As the project was evolving, new features were being added through software which allowed a better control over the truck. For example, a “check for command data flow” algorithm was inserted in order to stop the truck if no commands were present in the input buffer. This approach would prevent losing control over the vehicle in case the connection with the users drops.

The biggest limitation we had to overcome in the third version of Gizmo was the small range restraint. The Bluetooth chipset in the Nokia 6600 and the Dell D830 laptop were efficient only on a small range (approximately 5 meters for the cell phone and 15 meters for the laptop). In this version of Gizmo new hardware has been introduced to improve the Bluetooth range. A Nokia N90 cell phone was donated by fellow researcher Shannon Spanhake. Based on Spanhake’s project and results, these phones had a stronger BT radio than the Nokia 6600, which allowed a greater transmission range. Also, a Iogear BT USB dongle was purchased as a substitute to the internal BT radio present in laptop. The specification on the external BT dongle claimed a 330 feet (100 meters) transmission range (http://www.iogear.com/product/GBU321/?PHPSESSID=1059883988fe0edeec54d901ea26be1e).

Even though the BT side was looking promising, we needed to start thinking on a bigger scale. BT will only allow us to do a peer-to-peer communication; therefore the distance that we can cover is, by the BT nature, limited. A mesh network solution on the Wi-Fi 802.11 protocol was definitely a larger-scale approach. Allegedly, once the Gizmo

robot associates to a network and gets assigned a static IP address, any computer in the world with access to that network and knowing the correspondent IP address, could communicate to the robot. Now, how to deliver commands or retrieve data from that IP address was a little trickier. Three main approaches were attempted; serving a webpage, opening a socket, and using chat software.

3.1 Hardware

Gizmo v.3 was the first fully functional platform. This prototype is considered a final product and no hardware modifications have been made to it since it has been built. All upcoming Gizmo generations and prototypes have been built from the ground up without using parts from the previous versions.

The chassis for the third Gizmo platform was the first one to be custom designed. Daniel Johnson, mechanical engineer, designed an independent dual front and back steering platform around the electronic components which made up Gizmo. The body of the vehicle consisted of two front ends of the TRAXXAS E-MAXX RC trucks connected back-to-back through two plates of polycarbonate machined with a laser cam:

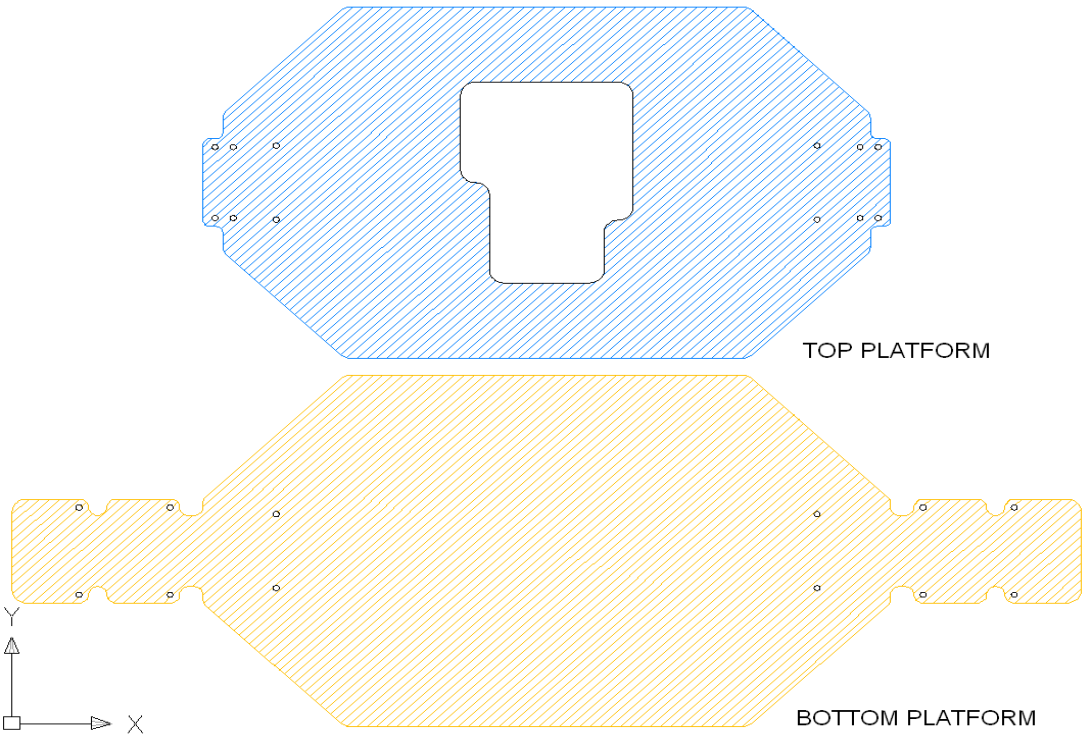

Figure 3.1: Main Body

This new hardware allowed sharper turns, Crab-Drive style and carrying a higher load. The drive system used only one motor with a better gear set in order to get more torque. Between Gizmo v.2 to Gizmo v.3 models there have been several hardware upgrades and few more add-ons. The following table summarizes the major differences between the two versions:

Table 3.1: Version Comparison

Hardware Component	**Gizmo .2**	**Gizmo v.3**
SBC	Wrap2E3 board, 233MHz processor	ALIX3C2 Board, 500MHz processor
Custom AVR board	Atmega164p, 2 PWM, 1 ADC	Atmega2560, 15 PWM, 6ADC, BT, Gas Sensor, 5V 4A voltage regulator, Sonar distance sensors, Digital Compass
Override Circuitry	NONE	YES, able to switch between Wi-Fi users and RC controller
Battery	10Ah 12V NiMh	2 x 14.8V 12Ah LiPo
Voltage Display	None	Yes. Small LCD indicating the battery voltage
Sound System	None	Mini-PCI audio card, Speaker and Microphone
Chassis	E-MAXX front steering	Custom platform, front and rear steering
ESC	Novak model 3014	Novak model 3221 HV Pro High-Voltage Brushless
Rx/Tx	25MHz, 3 channel	75MHz Futaba T7CAP, 7 channels

The integration of the gas sensor was thanks to the collaboration offered by researcher Spanhake with her Airbud project. More information on the pollution monitoring project can be found at http://www.calit2.net/newsroom/article.php?id=1080.

The new microcontroller board was designed around the 100 pin chip atmega2560. Not all of its functionalities were exposed but merely the most common ones such as 16 PWM, 4 serial ports, and a few ADCs.

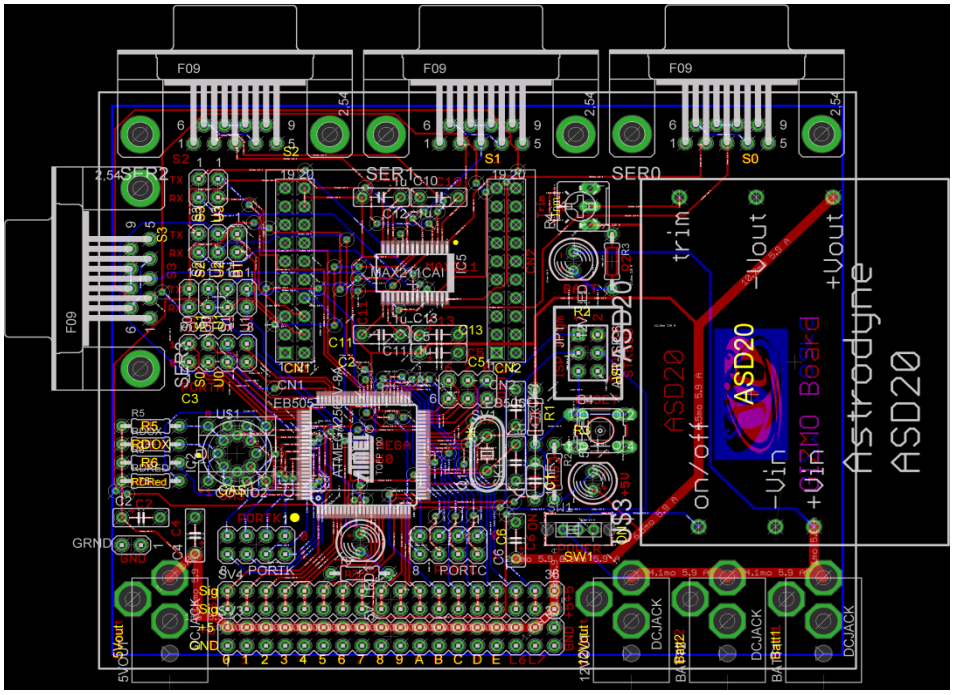

Figure 3.2: Gizmo v.3 Board Layout

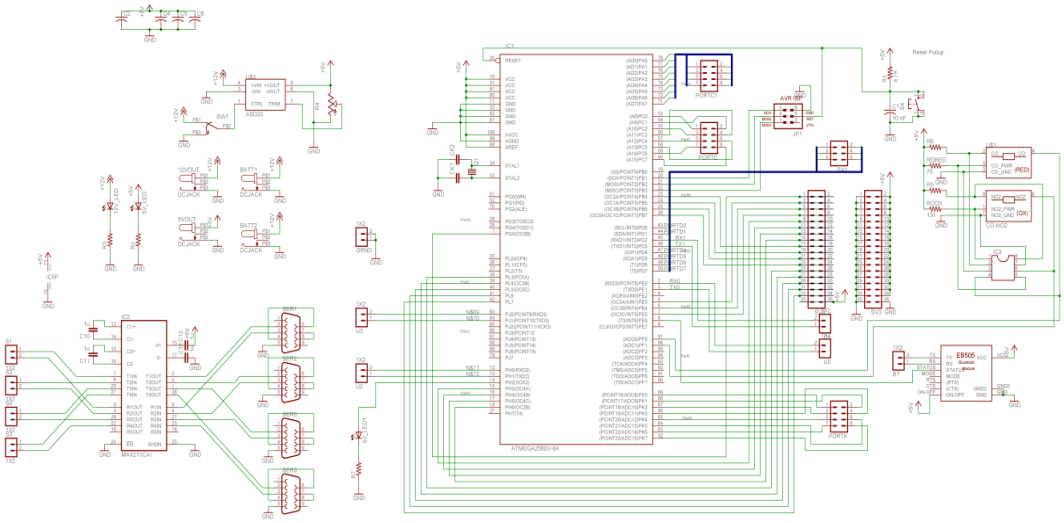

Figure 3.3: Schematic

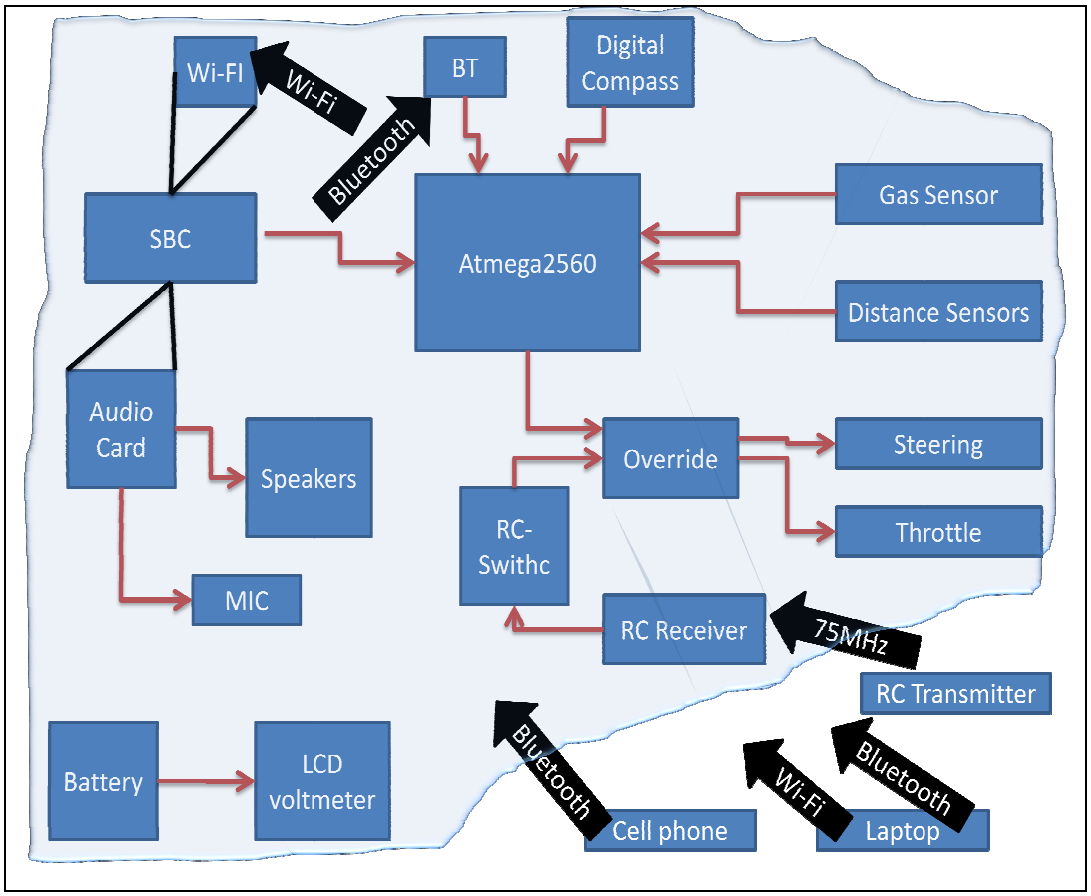

Figure 3.4: Block Diagram

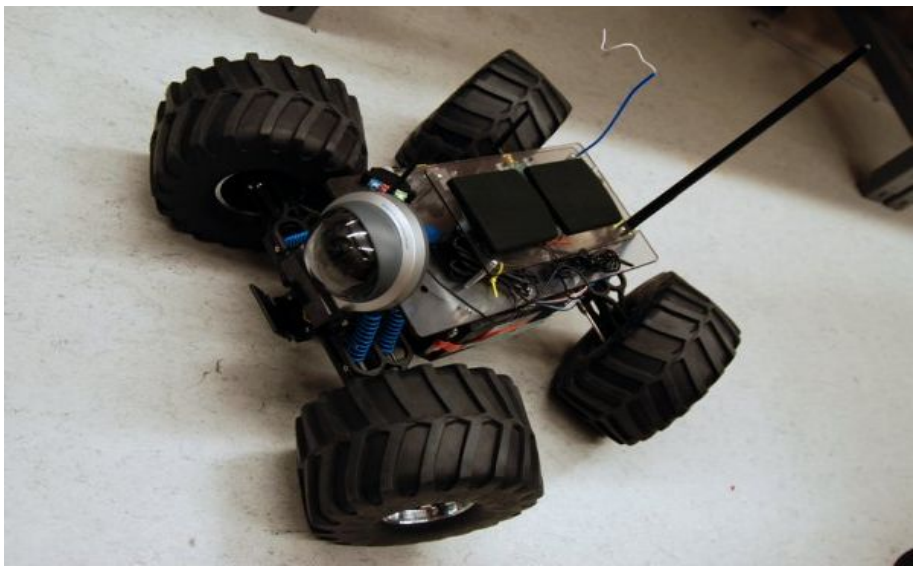

Figure 3.5: Gizmo v.3

Table 3.2: List of Materials

Component	Description
Chassis	2 x E-MAXX TRAXXAS front ends, and custom chassis
Receiver/Transmitter	75MHz Futaba T7CAP, 7 channels
ESC	Novak model 3221 HV Pro High-Voltage Brushless
SBC	Alix3c2 board, 500MHz processor
Compact Flash	Scan 1Gb
BT Module	eb501 from ieng7.com
Cell phone	N90
Wi-Fi Card	2511 MP PLUS 100mW: 802.11b mini PCI Card from netgate.com
Antenna	2.4 / 4.9 / 5.8 GHz Tri Band Rubber Duck RP-TNC from netgate.com
IP camera	Linksys WVC200
Motor	45x1 integy motor
Microcontroller	Atmega2560
Digital Compass	OS5000-S (RS232) from Ocean Server
Battery	2 x 14.4 12000mAh LiPo in parallel, Batteryspace.com
Voltage Display	Mini Digital Panel Meter C102b
Speakers	HP multimedia (USB) from frys.com
Microphone	Logitech

3.2 Software

Beside the great improvements achieved on the hardware side, important advances were made on the software area as well. With the new cell phone N90 and the new python code, a greater control range was obtained and the ease of driving the truck was also enhanced. The previously used open source application, Kismet, ended up being

too unreliable and with few bugs here and there. Therefore, a Linux command line approach was adopted. The command line *"iwlist eth1 scan"* with the new Wi-Fi cards, not only returned the available access points information present in the area, but also the clients. Finally, the major step forward that has been done in this prototype was on the Visual basic program.

3.2.1 Socket Connection

As mentioned earlier in the chapter, it was extremely important to achieve a communication range, between the user and the robot, larger than what the Bluetooth could offer. Configuring the robot to be able to communicate over 802.11 would extend the range anywhere from the coverage of the private network to anywhere in the world (i.e. using a global IP address or a private network with a satellite or cellular backhaul).

Based on the experience with Gizmo v.1, serving a webpage as the communication channel was not a fast approach, therefore this option was discarded immediately.

The chat system approach was successful and straight forward to implement. Two instances of the command line Linux chat were used, one installed on Gizmo and one on a Linux laptop (link http://linux.about.com/od/commands/l/blcmdl8_chat.htm). The user would start a chat communication with the robot which would received the command and transmit it out of the SBC serial port straight to the microcontroller board. From there, the command would get interpreted and executed. This solution was definitely faster than

serving a webpage, but the delay obtained was still visible to the human eye. The gaps recorded in the communication were between 300ms to almost 1 sec.

Finally, our last approach produced the results we were looking for. Digging in a little more into the available libraries in Visual Basic, a “socket” connection approach was implemented:

> An Internet socket (or commonly, a network socket or socket), is an end-point of a bidirectional process-to-process communication flow across an IP based network, such as the Internet. Each socket is mapped to an application process or thread. A socket is an interface between an application process or thread and the TCP/IP protocol stack provided by the operating system. An Internet socket is identified by the operating system as a unique combination of the following: Protocol (TCP, UDP or raw IP), Local IP address, or Local port number. Wikipedia

Two modules were created in Visual Basic, TCP and UDP module, which allowed opening a socket communication with the Gizmo IP address. The modules could accept either the IP address or resolve the hostname. In our case we went with resolving the Host name since it was easier to memorize. No need to mention that the delay decreased drastically and the Gizmo response, to the upcoming commands, was instantaneous to the naked eye.

Table 3.3: UDP Socket Module

```
Option Strict Off
Option Explicit On

Imports System.IO
Imports System.Threading

Module UDPModule
    Dim UDPClientObj As System.Net.Sockets.UdpClient
    Private receiveDone As New ManualResetEvent(False)
    Private udpEndpoint As System.Net.IPEndPoint
    Public buffer


    'Usage Example:
    ' ConnectUDP("gizmo2.dynamic.ucsd.edu", 10001)
    'Return true if the connection was successful
    'Return false if the connection failed
    Public Function ConnectUDP(ByVal address As String, ByVal port As Integer) As Boolean
        Dim ip As String

        ip = GetIPAddress(address)  'resolve the IP address
        Try
            UDPClientObj = New System.Net.Sockets.UdpClient(port)
            UDPClientObj.Connect(address, port)
            udpEndpoint = New System.Net.IPEndPoint(System.Net.IPAddress.Parse(ip), port)
        Catch ex As Exception
            MsgBox("UDP connect to " + address + " (" + ip + ") on port " + port.ToString() + " failed" +
vbCrLf + _
                "Make sure UDP server is running on " + address & vbCrLf & ex.ToString)
            Return False
        End Try
        Return True
    End Function

    Public Sub CloseUDP()
        Try
            UDPClientObj.Close()
```

Table 3.3: Continued

```
Catch ex As Exception

        End Try
    End Sub

    Public Function GetIPAddress(ByVal Address As String) As String
        Dim IPAddr() As Net.IPAddress
        Try
            IPAddr = System.Net.Dns.GetHostAddresses(Address)
            Return IPAddr(0).ToString
        Catch ex As Exception
            MsgBox("Could not resolve IP address.  Check internet connection.")
            Return "Error Resolving IP by DNS"
        End Try
    End Function

    Public Sub SendMessage(ByVal str As String)
        Dim bytes As [Byte]() = System.Text.Encoding.ASCII.GetBytes(str)
        Try
            'UDPClientObj.GetStream().Write(bytes, 0, bytes.Length)
            UDPClientObj.Send(bytes, bytes.Length)
        Catch ex As Exception

        End Try
    End Sub

    Public Function RecvMessage() As String

        Try
            'If UDPClientObj.GetStream().DataAvailable() Then
            '    Dim sr As New System.IO.StreamReader(UDPClientObj.GetStream())
            '    Return sr.ReadLine()
            'End If
            Return Str(UDPClientObj.Receive(udpEndpoint))

        Catch ex As Exception

        End Try
        Return ""
    End Function
End Module
```

Several tests were performed to verify this communication delay. All experiments were performed on both the UCSD network and on our private Calmesh network. First,

we wanted to check the delay obtained on a single hop situation (both Gizmo and user connected to the same access point). Second, a multi hop situation was studied in which Gizmo and the user were two or three nodes apart (on the Calmesh network) or in two different points in the UCSD campus. UCSD has practically a unique wireless hop between Gizmo and an access point and then is wired to the other nodes. To conclude, a hand over scenario was analyzed in order to measure the time of re-association of the Gizmo client from one node to the next one. Few conclusions can be extrapolated from the results. When using Gizmo on the UCSD network there is an inevitable delay associated with resolving the host name which maps to the IP address. In fact UCSD set up a dynamic IP address rather that a static one. The network does not handle very well static IPs on mobile platform which hop around between the different access-points (see roaming delay on table 3.2). Nevertheless, after the initial delay of establishing the connection by opening the socket, if Gizmo stays on the same access point, independent of where the user is located on the network, the delay achieved is better on the UCSD network than it is on the Calmesh. This is due the fact that UCSD has only one wireless hop. From the hand over test we can see that the delay does not depend on the network itself but it caused by the 802.11 protocol. A client needs to completely lose communication with an access point before it can re-associate to the next one; this causes delay. In the next generation of Gizmo a new Calmesh network protocol was released which improved the hand over process, and allowed Gizmo to be not only and client but an actual mobile access point. More information on the new Calmesh can be found at http://mesh.calit2.net/whzhao/thesis.pdf.

Table 3.4: Socket Communication Delay

EXPERIMENT	UCSD	CALMESH
Single Hop	Delay ~ practically instantaneous	Delay < 5ms
Different location on Network / Multi-Hop	Delay < 1/2 sec	10 ms < Delay < 2 sec Depending on the network load
Roaming	Delay ~12 sec	Delay ~12 sec

3.2.2 Visual Basic

With the control over Wi-Fi taken care of, all efforts were directed on the Audio and Video communication. So far, Gizmo didn't have any audio streaming capabilities, and in order to see the video from the Gizmo IP camera proprietary Linksys software was used or a webpage was accessed. The goal was to retrieve the video stream coming from the IP camera and be able to manipulate it and display it as we pleased. It was then that Jeff Cuenco and I came up with a solution for this problem. Cuenco was one of the students I mentored on an ECE 191 project who then got hired by Calit2 on the Network team. Cuenco and I set up this project and co-mentored it on an ECE191 senior design project named *"Gizmo TALK"*.

This project consisted in using two VideoLan instances, one embedded on the user end in the visual basic program, and one on the Gizmo SBC platform. VideoLan would capture

the audio coming from both microphones, the one on Gizmo and the laptop's one, and stream it to designated IP ports where it would be retrieved and sent to the speakers installed on the robot and the laptop. The video component worked on the same way. Unfortunately the Gizmo SBC didn't have any VGA output and therefore no screen was installed to display the image coming from the user end. Even if the two-way video was feasible, only one direction video streaming was implemented. For more information on VideoLan, please refer to http://www.videolan.org/.

The students assigned to this project were exceptional. All the goals required to finish this project were met. Even extra features were added. The achievements made by this team were highly recognized with the winning of the "Best Senior Design Project" award for the winter quarter 2008. For further information about this project, please refer to chapter 6.

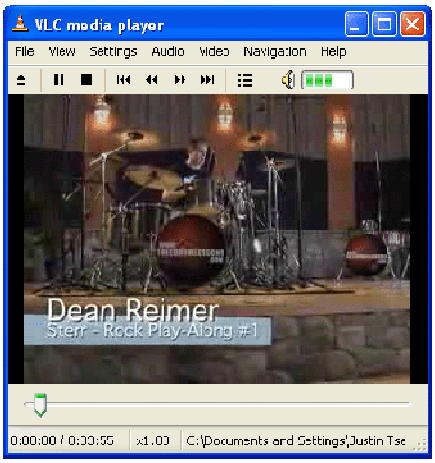

Figure 3.6: VideoLan, Courtesy of "Gizmo Talk"

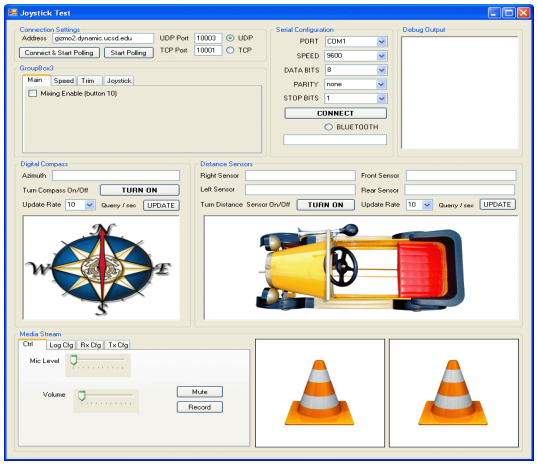

Figure 3.7: GUI, Courtesy of "Gizmo Talk"

3.3 Platform Evaluation

Gizmo v.3 was a great success. Dual steering, high torque drive, two-way audio and video communication, possible controls over short or long range protocols such as Blueetooth, 802.11, and 75MHz transmitter, and multiple user remote controls (cell phones, joystick, laptops,…) are only a few of the great capabilities of this platform. This research robot received a great deal of attention after a short article published by Jim Gogek, Director of National Media at UCSD (link http://www.calit2.net/newsroom/article.php?id=1205). From there to a month, Gizmo was all over the world's web; homeland security news, technology magazines, and so on. It also made a couple of appearance in the local news (link http://digitalcorrespondent.com/?p=434) and is currently used in the UCSD welcoming video which is showed to about 48,000 prospective students each year.

Gizmo v.3 was also used in several emergency response drills in the state of California. The latest video feeds from retrieve by the robot can be found at http://gizmoserver.calit2.net/traumgeber/gizmo_vids/ (keep in mind that the server will be there until the project is over, after that, please feel free to contact me if necessary at jar001@soe.ucsd.edu).

The prototype was a success, but that didn't mean that it could not be improved. In fact, the IP camera was not the optimal solution. The video feeds have being getting smoother throughout the different prototypes, but there was still space for improvement. A USB web cam could give a better performance, since we did not have to rely on any firmware and we control the video feed coming from the camera; anything from

encoding, frame rate, and quality. One action item for Gizmo v.4 was to find a USB camera that could be easily integrated on the SBC without being too heavy on the processor.

Some other limitations that were addressed for the next generation of Gizmo were: lack of autonomous navigation algorithm such as obstacle avoidance, unable to turn on the spot, going upstairs, two-way audio and video from cell phone, and increase Gizmo visual outputs by adding small LCD and a touch screen.

Chapter 4: Gizmo Tank

As the title for this chapter infers, the design for Gizmo v.4 has taken a slightly different approach. Gizmo Tank uses a four motor driving system with a dual differential speed controller to regulate the movement. The new hardware permitted the truck to skid steer which also allows the operation of the trucks in tighter spaces. In skid steering the ESC uses differential signals to drive the motors at different speeds, or in opposite direction, based on the action that needs to be performed. If the truck is stopped, in order to make a right turn, or left turn, the pair of wheels on one side of the Gizmo will spin in the opposite direction of the tires on the other side. If the truck is in movement, the tires opposite to the direction of the turn will spin faster than the other ones. For simple straight movement, all tires will spin at the same velocity. The generation of all this different signals is controlled by the ESC, no extra knowledge or skills are required by the user.

Due to time constraints, Gizmo v.4 has not being developed to its full potential. Unfortunately, few “action items” have been removed from this prototype and added to the “future work bucket”. Nonetheless, the new generation presents few major changes that drastically improved the Gizmo capabilities both on the hardware and software aspects.

4.1 Hardware

4.1.1 Skid Steering

From a hardware point of view, the adoption of skid steering drive was the main improvement for this prototype. Four independent motors controlled by a dual differential speed controller were used to build this system. Currently, the motors are connected in pairs by a parallel connection. The possibility to use a secondary ESC in order to have 4 independent signals was also considered. Having two ESC would allow the user to create four independent signals to control individually each motor. This might be necessary while driving in extremely rough and uneven terrains. Due to the amount of coding and tests needed to develop the controls for such a system, the dual ESC approach has been putted on hold for future work.

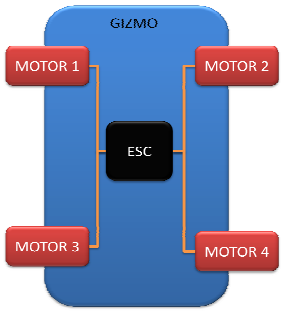

Figure 4.1: Dual ESC

To conclude, the use of a skid steering capable chassis would also allow the transition between tires and tank tracks. Due to the longer and wider surface, tank tracks could be the answer to the “going upstairs” problem. Daniel Johnson designed a tank track system that could be installed instead of the tires. Regrettably, tank tracks of that small size are really hard to find. The only ones we found were made of a material which

did not offer enough grip with the ground. Therefore, Gizmo with tank tracks was not successful and the tires were installed as a permanent solution for this platform.

4.1.2 Miscellaneous HW Add-Ons

As we have seen so far, a new upgraded Gizmo platform comes with new upgraded hardware. Some of the hardware changes include upgrading existing components; ESC, SBC, USB webcam, and microcontroller board. Others modifications consisted on adding new hardware; camera pan-tilt system, serial LDC screen, three axis accelerometer, and 900MHz radio.

The new ESC, the Alix.3c3, uses the same 500MHz processor as its previous version, but it comes with a built-in audio/video card with microphone input, speaker output, and VGA video output. This latest integration made the mini-PCI audio card from the Gizmo v.3 obsolete. It also allowed two future improvements; installing a VGA-input touch screen on Gizmo v.4 and use a 700MHz mini-PCI card for communication. As mentioned earlier, a dual differential ESC has taken the place of the Traxxas/Novak single channel ESC. Lastly, the troublesome IP camera has been finally replaced with a USB camera allowing more control over the image stream.

With the hardware innovations instead, we obtained complete control over the camera’s view angle with the use of a two servo pan-tilt system. While controlling the truck remotely, it was extremely helpful being able to look before actually make a turn. A serial LCD was used to display messages to people around the robot, debug, and report

internal hardware status. The three axis accelerometer integration was not used to its full potential in this prototype. In fact, we merely displayed its data; no action was taken after that. In future prototypes with more extensive navigation system, it will indispensable to have an accelerometer in order to have a robust algorithm. As a last add-on, a GPS was installed to support the first autonomous navigation algorithm.

4.1.3 Microcontroller Board

The main purpose of the microcontroller was the easier integration of the GPS, LCD, accelerometer, digital compass, camera pan-tilt system, and navigation sensors into the new platform. All components footprint were inserted in a small and compact PCB design.

The atmega2560 was still chosen as the microcontroller for the board. A new 100 pin socket was used on the first revision. The reason for attempting a Through Hole Push Down connector was to alleviate the soldering process. Soldering a highly dense chip straight to the board can be quite a task.

Figure 4.2: Atmega2560

Figure 4.3: 100 Pin QTFP Connector

For the second revision of the microcontroller board, I switch back to soldering the chip straight on the board. The connector footprint on the board was not suitable for a two layer board. Routing traces to the middle pins was really hard and sometime impossible. I suggest this connector to be used primarily on multi layer boards. Also, soluble flux makes soldering a 100 pin chip an easy task. Solder all pins down; don't worry about shorting pins together. Once the chip is all soldered down, remove the solder bridging between pins with the solder wick. Finally, wash any possible residual away, dry the board out, and there you go, a 100 pin chip soldered in less than a minute.

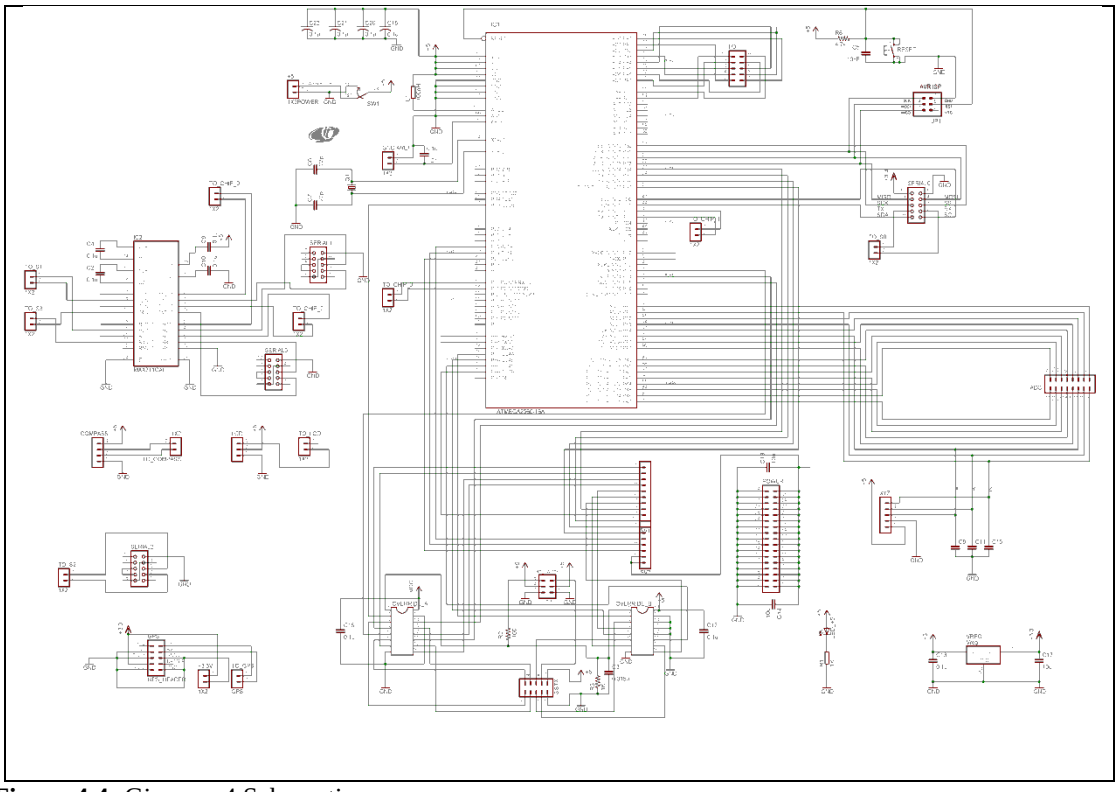

Figure 4.4: Gizmo v.4 Schematic

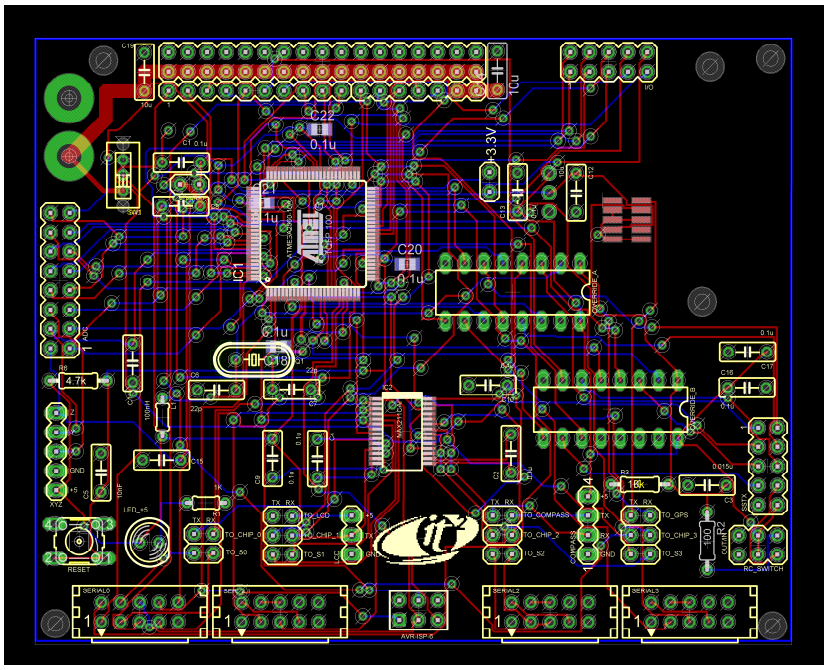

Figure 4.5: Gizmo v.4 Board Layout

Figure 4.6: Gizmo v.4

4.2 Software

4.2.1 Cell Phone Wi-Fi & SIP Server

The most challenging and interesting part of the entire Gizmo project was the Cell phone interface. Cell Phone technology is advancing tremendously integrating more and more components in such small factor that can fit in your pocket. My train of thought in this project was: *"if a laptop can do it, a cell phone can too"*. In the previous Gizmo versions, it has been proven that communication between a mobile device and the robot can be established with basic data transfer. My final objective is to be able to transmit data, audio, and video both ways and replace the use of the laptop interface.

The Nokia phone N95 was utilized for this Gizmo model. The phone has many features few of which are python, Bluetooth, infrared, accelerometers, and most important, Wi-Fi capabilities. This new communication protocol freed the cell phone from the short Bluetooth range limitation and turns it into a long range control device since it can connect to any wireless network. Even more exiting, it opened the door to Voice over IP (VoIP).

Establishing WI-FI connection with phone was not complicated thanks to the available python libraries. Here below is a piece of python code which would allow the connection to occur:

Table 4.1: PyS60 Code

```
#-------------------------------------------------------------
def connect_TCP(gizmoHost, gizmoPort):
   global gizmo
   HOST = gizmoHost    # The remote host
   PORT = gizmoPort    # The same port as used by the server
   print "define socket"
   gizmo = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   print "trying to connect to socket"
   gizmo.connect((HOST, PORT))
   print "connected"
#-------------------------------------------------------------
# Function name: send_TCP
# Description: send commands to a IP port
# Arguements: command
# Output: none
#-------------------------------------------------------------
def send_TCP(commmand):
   try:
      gizmo.send(command)
   except:
      connect_TCP
      gizmo.send(command)
#-------------------------------------------------------------
connect_TCP('10.1.100.101', 10001)   #connect_UDP('10.1.100.101', 10001)
```

The first application developed for this phone was based on the 3-axis accelerometer sample code wrote by Jurgen Scheible (link http://www.mobilenin.com/pys60/N95_accelerometer.php). The program allowed controlling the basic steering and throttle movements of the Gizmo robot by simply tilting the device in the direction we wanted it to go.

In order to establish a VoIP between the phone and the robot, the Session Initiation Protocol (SIP) was used:

> The Session Initiation Protocol (SIP) is a signaling protocol, widely used for setting up and tearing down multimedia communication sessions such as voice and video calls over the Internet. Other feasible application examples include video conferencing, streaming multimedia distribution, instant messaging, presence information and online games. In November 2000, SIP was accepted as a 3GPP signaling protocol and permanent element of the IMS architecture for IP based streaming multimedia services in cellular systems. Wikipedia

The idea of using SIP came to me after participating in the Jacobs School of Engineering Expo. The ***Best Poster Award*** was granted to Yuvraj Agarwal with the project "Cell2notify: energy management for VoIP over Wi-Fi Smartphone". Agarwal directed me to the same SIP server that he was using. The Asterisk SIP server is an open source and Linux command based software which worked great for our application. There are only 2 files that need to be modified in order to create the server and registered the clients; *extentions.conf* and *sip.conf*. Here below are the files used:

Table 4.2: SIP Files

extensions.conf	**sip.conf**
[internalExtensions] exten => _XXXX,1,Answer exten => _XXXX,2,Dial(SIP/${EXTEN},30) exten => _XXXX,3,Hangup [testingExtensions] ;exten => 600,1,agi,startDesktop.agi exten => 600,1,Answer exten => 600,2,Playback(vm-goodbye) exten => 600,3,HangUp	[general] port = 5060 ; Port to bind to bindaddr = 0.0.0.0 ; Address to bind to context = incoming ; Default for incoming calls of not registered phones pedantic = no allowguest = no ;register => 3000:password@128.54.61.239:7000/3000 ;re instantiate code below and create more users ;change 1000 with another number, ie 2000 [1000] type=friend

Table 4.2: Continued

[sip] include => internalExtensions include => testingExtensions exten => s,1,Answer exten => s,2,Playback(vm-goodbye) exten => t,3,HangUp	host=dynamic username=1000 nat=yes canreinvite=no qualify=yes secret=password context=sip disallow=all allow=ulaw allow=gsm ;port = 5060 ;

The server was installed on a Linux laptop that connected as a client to the Wi-Fi network. The decision of installing the Asterisk server on a PC rather than on a Gizmo was made because we wanted to reduce the processing load on the trucks' SBCs and to make the SIP network undependable of the truck location (the SIP network would crash if a truck with Asterisk installed loses connectivity to the network). Finally, with the simple command line *"sudo asterisk -c"* the Asterisk server would boot up and it would be ready to handle VoIP calls between cell phones, Gizmo robots, and laptops. For more information about Asterisk please refer to http://www.asterisk.org/.

The Nokia N95 carries SIP client software which can be configured to connect over Wi-Fi to any SIP server. At the other end, on the Gizmo robot side, an instance of the PJSUA SIP client was installed. PJSUA is a particularly easy to use Linux program.

Most important, it is command line based which made it easy to integrate and configure on the scripts running on the SBC. With a simple command line we are able to configure a client in auto response mode which will answer any incoming calls. In the command we need to specify the user, the domain, auto answer mode and other functions. Here below is the script which would initialize the SIP Client:

Table 4.3: SIP Script

```
#!/bin/bash

IP_ADDR=$1

if [-z $1]

then

        echo "usage: start_pjsua<ip_address/hostname of SIP server>"

else

        alsactl restore

        cd /home/javi/pjsip-apps/bin

        ./pjsua-i686-pc-linux-gnu --auto-answer=200 --no-tcp --id-
sip:3000@$IP_ADDR

fi
```

For more information on how to use this software and where to download it please refer to the following link http://www.opensourcesociety.org/2008/01/20/pjsip-command-line-voip-client-for-linux/.

4.2.2 Avoidance & GPS

The last improvement on the software was the result of the ECE 191 project "Autonomous Navigation System". During this project the team coded an algorithm for object avoidance based on the sensors distance reading. During every fixed period of time, all sensors would be queried. If the distance is above or below a set threshold, each sensor would set a variable to either 0 or 1. All the sensor responses would be concatenated together on a unique variable. Using look up tables, we would find the match for the variable and retrieve the pre-coded action to take (i.e. a reading like 0-0-0-0-1-1-0 would cause the truck to keep going left and forward).

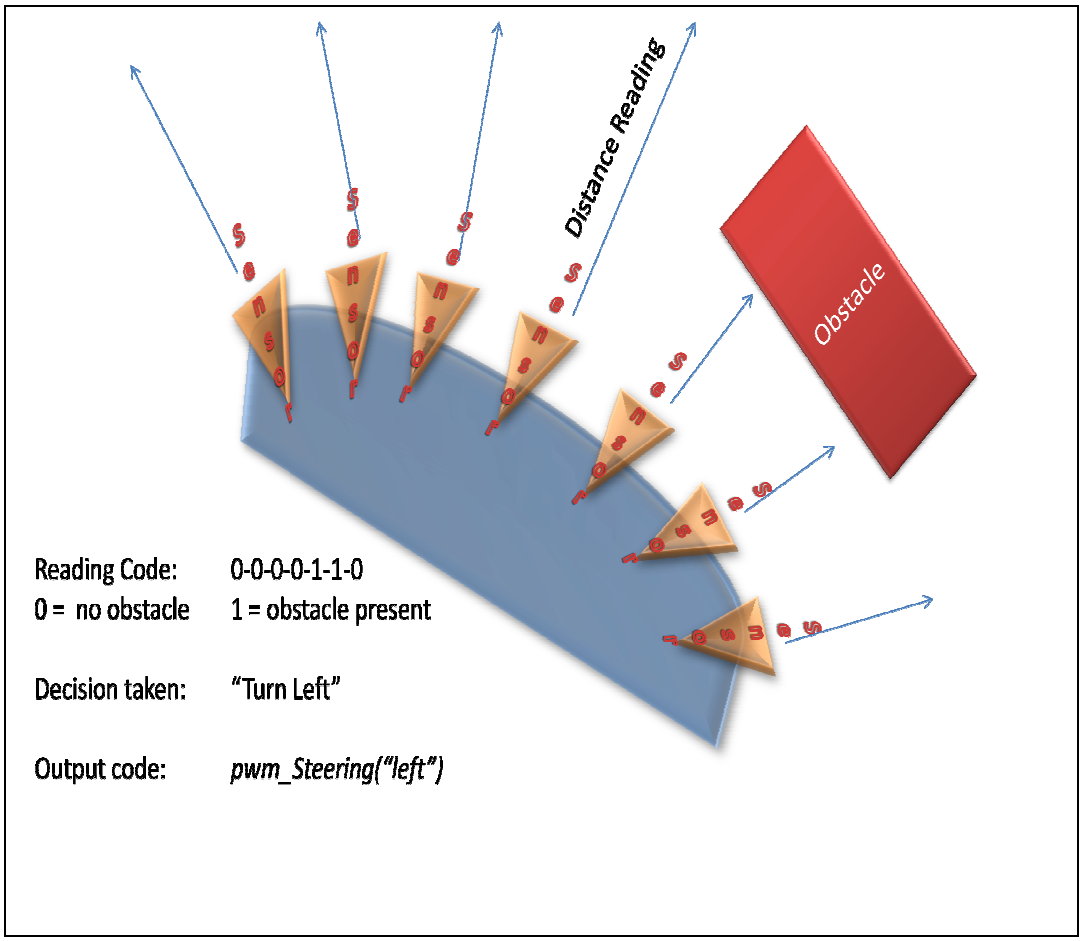

Figure 4.7: Sensor Reading Block Diagram

Furthermore, a GPS was integrated on the latest PCB design which was also used by the ECE191 students. This time, the algorithm would acquire the current GPS location and compare it to a list of pre determinate GPS way points. From the difference between a determinate way point and the current position we could calculate the heading we need to take and how far we are from the final destination.

4.3 Platform Evaluation

Gizmo Tank is the last prototype made so far. It has several important improvements from its previous version such as obstacle avoidance software, GPS, LCD display, SIP server with VoIP capabilities and a better mechanical control. This platform is a robust solution for network deployment and sensor data acquisition. Nevertheless, this platform can be further optimized for particular applications that may differ from the emergency network response. As mentioned earlier, Gizmo is a great research platform which can be modified to adapt to an extensive set of functions.

A main future upgrade for the following generations of Gizmo would be on the re-programmability and diagnostic analysis side. So far, every time a setting change is needed to be done on gizmo, we need to *SSH* into the truck and start, stop, or modify scripts. In future version, a user interface solution with a touch screen will be used to perform these tasks.

Chapter 5: Future Work and Conclusion

Beside keep building more and more version of improved Gizmo robots, I foresee the future work for this project to take a slight different direction. In fact, it is important that the Gizmo project finds new collaborations and new fields of study to be deployed on. Throughout the entire project, different UCSD departments have been showing their interest on the Gizmo platform as well as local industry partners. Every single one of them has a different application in mind. My future goal is to establish several collaborations with departments and industries sharing few Gizmo platforms and see what can they do with it. Only exposing Gizmo to multidisciplinary environments, with collaboration between minds which see things from different points of view, we will be able to take this project to the next level.

In conclusion, Gizmo project represents a great research and learning platform which includes several technology aspects such as: communication, analog and digital circuit design, and audio/video DSP. Gizmo helped promoting learning among students and also became a valuable solution for network deployment.

Chapter 6: Education and Outreach

One of the major efforts of the Gizmo project was in outreach and education. I wanted Gizmo to be accessible to everyone who was interested on it. Gizmo presented a variety of sensors and technology which made it perfect as an educational platform. I would like to particularly thank Professor Charles Tu, Professor Pankaj Das, and Professor Clark Guest for giving me the great possibility of mentoring several ECE 191 projects in the last couple of years. Also, I would like to thank Ms. Collier Kaler for allowing me to mentor Preuss School UCSD senior Internship students.

Here below are the descriptions of the projects mentored so far. All final reports and contact information on the students involved can be found in the ECE191 website or in the Circuits lab webpage:

http://circuitslab.calit2.net

http://ece-classweb.ucsd.edu:16080/spring08/ece191/

Note: Change the above web address to the quarter of interest.

FALL QUARTER 2006:

ECE191: **Autonomous Wi-Fi Sniffer RC truck GIZMO**

Project Goals:

The goal for this project is develop the necessary software to create signal strength surface plots. The software will be installed on the Gizmo truck. Gizmo will record the signal strength of every SSID present in the air each second as it drives around

the area. This data will be associated to Gizmo's respective GPS location, (Gizmo is equipped with a GPS module). The data must be sent over the CalMesh wireless network and stored remotely. In addition, an intuitive and user friendly interface (i.e., a webpage) must be created in order to manage (and display) all of the relevant data (create surface plots, make graphs, retrieve statistics, and derive useful conclusions).

Project Deliverables:

- Create software capable of accurately recording and storing signal strength and GPS data.
- Create a user interface (a webpage) to manage all data. The page will include features to enable creation of surface plots and graphs, and conduct statistical analysis.
- Create the capability of overlapping terrain maps with signal strength surface plots.

ECE191: **G.R.O.B.I. (Gizmo remote operated Bluetooth interface)**

Develop custom circuit including an accelerometer and Bluetooth wireless module that will be installed in the Gizmo truck. Create phone software able to control the Gizmo truck through the Bluetooth enabled cell phone joystick / keypad. Create phone software able to control the Gizmo truck through voice commands. Integrate with the hardware. All equipment necessary for the project has been procured already. Desired skills include knowledge of programming (C and Python will be used),

experience with embedded sensors and/or systems, and interest in local area wireless technologies such as Bluetooth. Experience in circuit / PCB layout is beneficial, although these skills will be taught (and firsthand experience using a PCB etching / milling machine will be provided).

WINTER QUARTER 2007:

ECE191: **Where is my Roomba? Indoor Position Locator System for Roomba**

Project Goals:

The goal for this project is to create a reliable navigation system which will allow locating the Roomba inside a building given a point of origin. The team members will need to design a system from which they can obtain acceleration and direction at any point in time; these data will be used to calculate the X and Y position of the Roomba. Different approaches can used to obtain acceleration (i.e. use infrared sensors to calculate the rpm of the wheels, accelerometer sensors), as well as direction (i.e. compass, Honeywell modules). The team will need to decide which one is the best approach.

Project Deliverables:

- Identify different ways to determine acceleration and direction of the Roomba and determine optimal approach.
- Design a system that obtains acceleration and direction from the Roomba at any point in time.

- Calculate position of the Roomba based on acceleration and direction data.
- Test Navigation system design over CalMesh network.

SPRING QUARTER 2007

ECE191: **Network Sniffer and Bluetooth Navigation System**

Project Goals:

During winter ’07 and fall ‘06 three ECE 191 teams approached the Gizmo and Roomba project obtaining great results. The goal for this new project is to continue from where the other teams left off and to integrate the different technologies together. In particular, a more accurate navigation system needs to be developed for the Roomba (e.g. integrating a accelerometer) and a two way radio (e.g. WRAP board) needs to be addressed in order to replace the current one way radio solution and overcome its limitations (Soekris board). A user interface MUST be created in order to access all the data.

Project Deliverables:

- Replace the current of the current Soekris radio with the new WRAP board
- Debug and improve the current navigation algorithms with the introduction of an accelerometer
- Developed a user interface which will allows data analysis and computations
- Set up a Data Base on the UCSD network.

Extra:

If the team feels confident enough, an old Roomba (currently not working) could be fixed and modified by acquiring the necessary parts. Ultimately, we would like to erase the IRobot embedded software and developed our own which would allow us to make the Roomba go through predetermined paths instead of the current "bump-and-turn" algorithm. To conclude, a Bluetooth interface (developed by the GROBI team in ECE191 FALL '06) will be installed in the Roomba which will allow real time communication between the user and the Roomba.

WINTER QUARTER 2008:

ECE191: **Telekinesis Glove (wireless remote control glove)**

Project Goals:

The goal for this new project is create a new and innovative way to control the Gizmo truck and the Wi-fli Mesh Condor. Using five triple axis accelerometers attached to each finger and a small Bluetooth module, the team will prototype a glove, program a microcontroller and designed a user interface in order to collect and retrieve data.

Project Deliverables:

- Build the necessary harness on the glove to mount the sensors and module.
- Write the necessary software to retrieve the accelerometer data.
- Create a database of gestures which will represent the different commands
- Develop a GUI to collect and retrieve data.

Extra:

Due to the short range in which Bluetooth is effective, a different wireless bridge is in need for the Wi-fli mesh Condor. One approach could be a 2.4GHz bridge. Also, the team could modify a standard RC transmitter and connected to the glove. The team will explore and research different possibilities and decide which one will be the most efficient.

ECE 191: **Gizmo Talk, two-way audio communication**

Project Goals:

The goal of this project is to build a two-way voice communication system over the phone or laptop using voice over IP and/or over Bluetooth to a speaker system built on the truck. Also, a new set of distance sensor will need to be installed on the truck and incorporated with the existing GUI and circuit board.

Project Deliverables:

1) Design a method of audio packet TX via Bluetooth or Wi-Fi that exists on Gizmo.
2) Investigate a preferred sensor technology for distance sensing (Ultrasonic versus Laser) and replace existing sensors
3) Redesign Gizmo's actuation main-board to incorporate added features.
4) Enhance existing GUI for wireless communication selection and general sensor data.
5) Code:

-VB (PC) or Python S60 (Cell phone) for GUI

-Embedded C for sensor to microprocessor interface

-VB for audio board

SPRING QUARTER 2008:

ECE191**: Autonomous Navigation System**

Project Goals:

The goal of this project is to design and install a distance sensor platform on the Gizmo truck which will allow the vehicle to navigate through a maze, parallel park, approach intersection with incoming traffic, and navigate through an unknown area without bumping into objects. The team will have to chose the right sensors, connect them to a microprocessor and write some code which will decide what to do based on the incoming data. Also, some sort of user interface will be needed to collect and display the sensor data and the current action the Gizmo truck is performing.

Project Deliverables:

- Designs distance sensor platform.
- Build and test system in the different tasks.
- Develop a user interface to display real-time data.

Reference Material

[1] http://circuitslab.calit2.net

[2] http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=296-3828-5-ND

[3] http://www.traxxas.com/products/electric/emaxx3905/trx_emaxx3905.htm

[4] http://en.wikipedia.org/wiki/Bluetooth

[5] http://www.atmel.com/dyn/Products/product_card.asp?part_id=3887

[6] http://www.cadsoft.de/download.htm

[7] http://www.pcbexpress.com

[8] http://www.pcbexpress.com/downloads/EAGLE%20Convert-Sunstone%20Protos.pdf

[9] http://www.batchpcb.com

[10] http://www.sparkfun.com

[11] http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=2735

[12] http://winavr.sourceforge.net/download.html

[13] http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

[14] http://www.mobilenin.com/pys60/menu.htm

[15] http://msdn.microsoft.com/en-us/library/ms709358(VS.85).aspx

[16] http://www.iogear.com/product/GBU321/?PHPSESSID=1059883988fe0edeec54d901ea26be1e

[17] http://www.calit2.net/newsroom/article.php?id=1080

[18] http://linux.about.com/od/commands/l/blcmdl8_chat.htm

[19] http://mesh.calit2.net/whzhao/thesis.pdf

[20] http://www.videolan.org/

[21] http://www.calit2.net/newsroom/article.php?id=1205

[22] http://digitalcorrespondent.com/?p=434

[23] http://gizmoserver.calit2.net/traumgeber/gizmo_vids/

[24] http://www.mobilenin.com/pys60/N95_accelerometer.php

[25] http://www.asterisk.org/

[26] http://www.opensourcesociety.org/2008/01/20/pjsip-command-line-voip-client-for-linux/

[27] http://circuitslab.calit2.net

[28] http://ece-classweb.ucsd.edu:16080/spring08/ece191/

[29] http://responsphere.calit2.net/gizmo/

[30] http://www.bluetooth.com/bluetooth/

[31] http://www.leninsgodson.com/courses/pys60/menu.htm

[32] http://www.python.org/doc/

[33] http://www.embedded.com/story/OEG20010821S0096

[34] http://grobi.wetpaint.com/page/Research

[35] http://www.digi.com/pdf/ds_xstreammodule.pdf

[36] http://www.harwin.com/include/downloads/pdfs/PAGE_17.PDF

[37] http://www.sparkfun.com/commerce/product_info.php?products_id=709

[38] http://www.digi.com/products/wireless/point-multipoint/xbee-pro-series1-module.jsp

[39] http://www.digi.com/technology/rf-articles/wireless-zigbee.jsp

[40] http://www.ocean-server.com/compass.html

[41] http://www.sparkfun.com/datasheets/Components/LD1117V33.pdf

[42] http://www.linksys.com/servlet/Satellite?c=L_Product_C2&childpagename=US%2FLayout&cid=1175229403526&pagename=Linksys%2FCommon%2FVisitorWrapper&lid=0352686883B01

[43] http://www.netgate.com/index.php?cPath=26_34

[44] http://www.pcengines.ch/wrap2e3.htm

[45] http://www.mini-box.com/s.nl/sc.8/category.19/.f

[46] http://ucsdnews.ucsd.edu/thisweek/2007/04/23_squirrel.asp

[47] http://www.teamnovak.com/products/esc/hv_pro_esc/index.html

[48] Neil H.E. Weste, David Harris. *CMOS VLSI Design, 3rd Edition.* Addison Wesley. ISBN: 0-321-14901-7. December 2005

[49] Jürgen Scheible, Ville Tuulos. *Mobile Python, Rapid Prototyping of Applications on the Mobile Platform.* Wiley. ISBN: 978-0-470-51505-1. 2007

Appendices

Appendix A: Bill of Materials: Gizmo v.3

Part	Value	Device	Package	Description
+3.3V_TEST	MA01-2	MA01-2	MA01-2	PIN HEADER
3V_LED		LED5MM	LED5MM	LED
5V_LED		LED5MM	LED5MM	LED
C1	20pF	C-US050-024X044	C050-024X044	CAPACITOR, American symbol
C2	20pF	C-US050-024X044	C050-024X044	CAPACITOR, American symbol
C3	10 nF	C-US050-024X044	C050-024X044	CAPACITOR, American symbol
C4	1uF	C-US050-024X044	C050-024X044	CAPACITOR, American symbol
C5	1uF	C-US050-024X044	C050-024X044	CAPACITOR, American symbol
ESC/SERVO	MA03-2	MA03-2	PIN HEADER	
PWM0		MA03-2	MA03-2	PIN HEADER
PWM2		MA03-2	MA03-2	PIN HEADER
Q1		XTAL/S	QS	CRYSTAL
R1		R-US_0204/7	0204/7	RESISTOR, American symbol
R2		R-US_0204/7	0204/7	RESISTOR, American symbol
R3		R-US_0204/7	0204/7	RESISTOR, American symbol
R4	4.7K	R-US_0204/7	0204/7	RESISTOR, American symbol
R5		R-US_0204/7	0204/7	RESISTOR, American symbol
SV1	3.3V_SRC	MA03-2	MA03-2	PIN HEADER
U$1	EB505	EB505	EB505	
U$2	MEGA164-P	MEGA164-P	DIL40	

U$3	ACCEL	ACCEL	FREESCALE_ACCEL	
U$5	MCP1700	MCP1700	MCP1700	
UART1	MA01-2	MA01-2	MA01-2	PIN HEADER

Appendix B: Bill of materials: Gizmo v.4

Part	Value	Device	Package	Description
5VOUT	DCJACK	DCJACK	161-0725	
5V_LED		LED5MM	LED5MM	LED
5V_LED1		LED5MM	LED5MM	LED
12VOUT	DCJACK	DCJACK	161-0725	
12V_LED		LED5MM	LED5MM	LED
BATT1	DCJACK	DCJACK	161-0725	
BATT2	DCJACK	DCJACK	161-0725	
BT	1X2	1X2	1X2	
C1	10 nF	C-US050-024X044	C050-024X044	CAPACITOR, American symbol
C2		C-US050-025X075	C050-025X075	CAPACITOR, American symbol
C4		C-US050-025X075	C050-025X075	CAPACITOR, American symbol
C5		C-US050-025X075	C050-025X075	CAPACITOR, American symbol
C6		C-US050-025X075	C050-025X075	CAPACITOR, American symbol
C10	.1u	C-EU050-025X075	C050-025X075	CAPACITOR, European symbol
C11	.1u	C-EU050-025X075	C050-025X075	CAPACITOR, European symbol
C12	.1u	C-EU050-025X075	C050-025X075	CAPACITOR, European symbol
C13	.1u	C-EU050-025X075	C050-025X075	CAPACITOR, European symbol
CK1		C-EU050-025X075	C050-025X075	CAPACITOR, European symbol

CK2		C-EU050-025X075	C050-025X075	CAPACITOR, European symbol
GRND	1X2	1X2	1X2	
IC1	ATMEGA2560V-8A	ATMEGA2560V-8A	TQFP100	MICROCONTROLLER
IC2		DIL8ROUND	DIL08-ROUND	Dual In Line
IC5	MAX211CAI	MAX211CAI	SSOP28	RS232 TRANSEIVER
JP1	AVR-ISP-6	AVR-ISP-6	AVR-ISP-6	AVR ISP-6 Serial Programming Header
PORTC		MA04-2	MA04-2	PIN HEADER
PORTC1		MA04-2	MA04-2	PIN HEADER
PORTK		MA04-2	MA04-2	PIN HEADER
Q1		CRYTALHC49S	HC49/S	CRYSTAL
R1	4.7K	R-US_0204/7	0204/7	RESISTOR, American symbol
R2		R-US_0204/7	0204/7	RESISTOR, American symbol
R3		R-US_0204/7	0204/7	RESISTOR, American symbol
R4		TRIM_US-B25U	B25U	POTENTIOMETER
R5		R-US_0204/7	0204/7	RESISTOR, American symbol
R6		R-US_0204/7	0204/7	RESISTOR, American symbol
R7		R-US_0204/7	0204/7	RESISTOR, American symbol
RDOX	131	R-US_0204/7	0204/7	RESISTOR, American symbol
RDRED	75	R-US_0204/7	0204/7	RESISTOR, American symbol
S0	1X2	1X2	1X2	
S1	1X2	1X2	1X2	
S2	1X2	1X2	1X2	
S3	1X2	1X2	1X2	
S4		10-XX	B3F-10XX	OMRON SWITCH
SER0		F09HP	F09HP	SUB-D
SER1		F09HP	F09HP	SUB-D
SER		F09HP	F09HP	SUB-D

2				
SER 3		F09HP	F09HP	SUB-D
SV1		MA03-2	MA03-2	PIN HEADER
SV3		MA18-2	MA18-2	PIN HEADER
SV4		MA18-2	MA18-2	PIN HEADER
SW1		SWITCH	SWITCH	
U$1	CO-NO2	CO-NO2	TO99	
U$2	EB505	EB505	EB505	
U$3	ASD20	ASD20	ASD20_PAC	
U0	1X2	1X2	1X2	
U1	1X2	1X2	1X2	
U2	1X2	1X2	1X2	
U3	1X2	1X2	1X2	