UNIVERSITY OF CALIFORNIA

Los Angeles

Readout Circuit System for In_2O_3/RGO Nanocomposite Gas Sensors

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

by

Cheng-Yi Lin

2015

ABSTRACT OF THE THESIS

Readout Circuit System for In_2O_3/RGO Nanocomposite Gas Sensors

by

Cheng-Yi Lin

Master of Science in Electrical Engineering

University of California, Los Angeles, 2015

Professor Kang L. Wang, Co-Chair

Professor Xiangfeng Duan, Co-Chair

A readout circuit system for In_2O_3/RGO nanocomposite gas sensors using open-source software has been developed for the first time. The readout system adopts a Raspberry Pi as an electronic control unit and incorporates different electronics components to realize the function of a source measure unit (SMU). During the operation, real-time results of measured gas concentrations can be accessed through the Internet and alarm functions are also included.

All control programs were written in Python language. Using this readout system, current response of gas sensors toward oxygen concentrations (2,000—32,000 ppm) in argon environment at 140 ^{o}C are in a good agreement with the data measured by Agilent SMU (B2902A). Furthermore, temperature effects and transient response of the proposed system are investigated. The success of this readout system demonstrates the potential use of open-source hardware to construct scientific instruments with the advantages of miniaturization, low cost, flexible design, and Internet access.

The thesis of Cheng-Yi Lin is approved

Oscar Stafsudd

Kang L. Wang, Committee Co-Chairs

Xiangfeng Duan, Committee Co-Chairs

University of California, Los Angeles

2015

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronyms	Meaning	Acronyms	Meaning
ADC	Analog to digital converter	ppm	Parts-per-million
API	Application programming interface	RGO	Reduced graphene oxide
.csv	Common separate value	R_s	Sensor resistance
DHCP	Dynamic host configuration protocol	R_{In2O3}	Resistance of In_2O_3/RGO
DIP	Dual in-line package	R_f	Feedback resistor
FOSS	Free and open-source software	SMU	Source measure units
GPIO	General purpose input/output	SMS	Short message service
GO	Graphene oxide	SSH	Secure shell
IC	Integrated circuits	sccm	Standard cubic centimeters per minute
I_d	Drain current	TEM	Transmission electron microscopy
In-Amp	Instrumentation amplifier	VNC	Virtual network computing
LCD	Liquid crystal display	V_g	Gate voltage
LED	Light emitting diode	V_d	Drain voltage
MFCs	Mass flow controllers	V_s	Adjustable bias
MSOP	Mini small outline package	XPS	X-ray photoelectron spectroscopy

Acknowledgements

I would like to express my appreciation to those who helped me to complete this thesis. I am deeply grateful to both of my advisors Dr. Duan and Dr. Wang for their help, support and suggestions. Furthermore, I am much obliged to Prof. Stafsudd for his kindness and review. As for the device, I am indebted to Dr. Mengning Ding, Dr. Fanli Meng, and Lin Mei. Without their supports and discussions, this thesis could have not been achieved. Special thanks are given to Ministry of Education, Taiwan, and PME Prof. Lin for their scholarships in this research. I wish to thank to all my colleagues. Especially I am obliged to Nathan for the thesis revision, as well as Kaveh Shoorideh and Hao Wu for their helpful discussions on computer programming. I also feel fortunate to have the opportunity to learn valuable experience from Dr. Yuxi Xu, Dr. Anxiang Yin, Hung-Chieh Cheng and Dr. Qiyuan He.

I could have not done this work without the support of my family. I would like to give my special thanks to my parents.

Chapter I: Introduction

1.1 Background

As technology advances, open-source software and data sharing is becoming more and more important nowadays. In the past, most research projects utilized purchased hardware equipment with computer software provided by vendors. Measurements may require sophisticated, expensive and even bulky apparatuses. In electrical measurements, source measure units (SMUs) are very common and useful instruments that characterize electrical properties of devices such as current and voltage. However, the price of high-class SMUs, Keithley 4200 SCS for example, can cost easily more than $40,000. Other general SMUs (Agilent B2902A) are less expensive but they still cost around $8,900. This high instrument cost might limit the application of biosensor and biomedical instrumentation, especially in developing countries where people lack medical and infrastructural resources. Therefore, developing cost-effective instruments can meet the increasing demand for scientific equipment in resource-limited settings. With the help of free and open-source software (FOSS), designers/users can now create open-source scientific hardware, which can radically reduce the costs of their experimental work.[1-3]

A key that promotes open-source hardware project is the Arduino electronic prototyping platform.[4, 5] In the past few years, many universal microcontrollers have become available. In addition to Arduino, another prominent example is the Raspberry Pi—a single printed circuit

board (PCB) microcomputer, which was introduced to the market in 2012.[6] The original purpose of the Raspberry Pi was to enhance education in computer science, electronics, and automation.[7, 8] A plethora of practical applications of this universal platform have emerged soon after the sales had started. Typical applications include: data logging[9], controlling displays[10], and automation of household appliances. The Raspberry Pi (general-purpose microcomputer) therefore demonstrates a great potential to construct analytic instrumentation, such as SMUs.

In many countries, the issue of air quality is still a main concern, as a clean air supply is imperative to our health and environment. Gas sensors are electronic devices that can be used to monitor the target gas concentration by their electrical current signals which are typically measured by SMUs. Among different types of gas sensors, metal oxide gas sensors have been widely adopted in portable gas detection systems due to the advantages of compact size, low cost and simple measurement. Although metal oxide nanostructures and reduced graphene oxide (RGO) sheets have been widely studied in lithium-ion storage and capacitor,[11, 12] their applications in gas sensing are still limited. The performance of this type of new materials such as In_2O_3/RGO needs to be investigated. Additionally, when gas sensors are used to monitor flammable or toxic gas, it is highly desirable to have an alarm system to indicate abnormal situations.

1.2 Motivation and Objective

The open-source paradigm is now enabling the development of open-source scientific hardware, thus leading to a cost reduction and design flexibility in experimental work. To detect target gas concentration through nanocomposite gas sensors, this work aims to develop a smart readout circuit system with alert functionality to be an alternative of an expensive SMU. The novel In_2O_3/RGO nanocomposite gas sensors are adopted to detect different concentration of oxygen. The responses are measured by the proposed readout system using a Raspberry Pi and are compared with a commercialized SMU (Agilent B2902A) to evaluate its performance. The goal of this thesis is to design and implement a low-cost readout circuit that can provide a reasonable accuracy, while at the same time equip with alarm system to indicate abnormal gas concentrations. The data measured by the proposed system is logged and plotted on a webpage, so that users can remotely monitor the time-resolved current response of gas sensors through the Internet, which brings a lot of convenience.

1.3 Organization of this Thesis

The organization of this thesis is as follows, there are seven chapters in this thesis. Chapter I introduces the background of open-source scientific hardware. A readout circuit with alarm system is developed in this work using open-source software and Raspberry Pi microcontroller. The motivation of this work is described in Chapter I as well. Chapter II

provides a literature review of nanocomposite gas sensors and different readout circuit system. Chapter III describes the fabrication of In_2O_3/RGO nanocomposite gas sensors and construction of the readout circuit. The measurements of different unit of readout system are mentioned in Chapter IV. The experiment results of detection of oxygen are stated in Chapter V. Finally, the conclusion and future work are presented in Chapter VI and Chapter VII, respectively.

Chapter II: Literature Survey

2.1 Nanocomposite Gas Sensors

Semiconductor based sensors have been developed as an alternative gas sensing technology with many potential advantages such as low cost, easy production, compact size and simple measurement electronics, but usually with limited sensitivity.[13, 14] To this end, considerable efforts have been made in exploring various nanostructures for improved sensor sensitivity, e.g., by controlling the morphology of nanoscale metal oxides[15, 16] and/or by incorporating noble metal catalysts on a nanocrystalline metal oxide matrix.[17-19] Graphene possesses many unique characteristics such as high surface area and low electronic noise,[20-22] and therefore has shown great potential for forming high performance hybrid nanostructures with semiconductor nanomaterials. Although nanocomposites of metal oxide nanostructures and reduced graphene oxide (RGO) sheets have already been widely investigated in lithium-ion storage and super-capacitor, their applications in gas sensing have not been widely studied. Here we report a novel In_2O_3/RGO nanocomposite as a new sensing material for highly sensitive detection of ambient gas, such as oxygen. The In_2O_3/RGO nanocomposite is prepared through a one-step wet chemical approach and exhibits higher sensitivity at higher operating temperature. The systematic studies show that the RGO played an important role in improving the sensing performance, control of the In_2O_3 particle size, and enhancement of the physisorption.

2.2 Detection Mechanism

For the pure In_2O_3 film which was formed by particle stacking as shown in Fig. 2. 1 (a), the electron needed to jump the particle boundary barrier during the transport process. The resistance of the film (R_{In2O3}) could be express as follows:

$$R_{In2O3} = \sum R_p + \sum R_c \quad (1)$$

Where R_p and R_c are the internal resistance of the In_2O_3 particles and the contact resistance at particle boundaries, respectively. As discussed above, R_p of small particles could be modulated dramatically upon reactions with analyte, while R_c is nearly a constant.

For the In_2O_3/RGO nanocomposite, RGO was dispersed in the In_2O_3 particles layer by layer as shown in Fig. 2. 1 (b). The electron transport property of the nanocomposite device was significantly enhanced as a result of the high electron mobility of RGO itself[23] and the largely reduced contact resistance R_c. The modulation of R_c is mainly because the change from particle-particle contact to particle-RGO contact, which was formed through covalent bonding between In_2O_3 colloids and RGO when the nanocomposite was produced. The excellent particle-RGO contact largely decreased the R_c in formula (1). The electrons easily transported through the nanocomposite layers as shown Figure 1b. The film resistance (R_{film}) was largely decreased and almost relied on R_p, so the size effect of the In_2O_3 particles could be brought into play more efficiently.

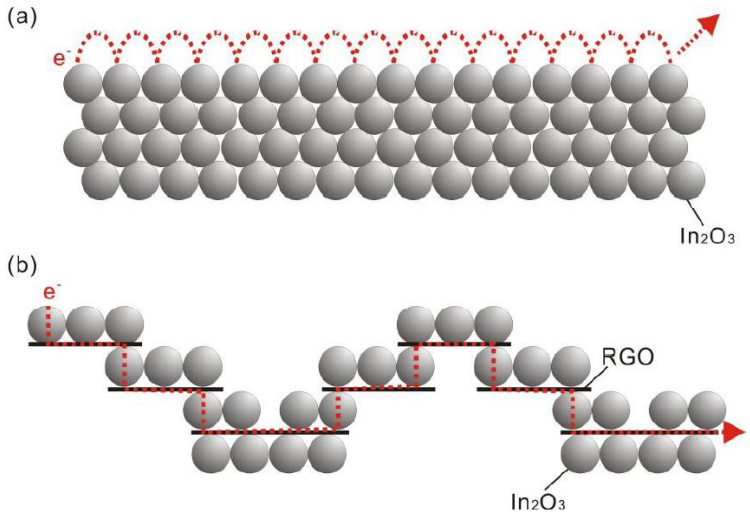

Fig. 2. 1 Schematic illustration of the electron transport in the films consisted of (a) pure In_2O_3 particles and, (b) In_2O_3/RGO nanocomposites.

In air, where is the most common operating ambience for a gas sensor, the ionosorption of oxygen is particularly important. We chose oxygen as our detection target to investigate the functionality and performance of the proposed readout circuit. Adsorbed oxygen on the semiconductor surface tends to become a negatively charged surface state (acceptor-type), leading to a charge transfer from the surface conduction band to the adsorbed oxygen. Assuming a flat band bending prior to the adsorption of oxygen, the charge transfer process will result in the formation of a surface depletion layer, as illustrated in Fig. 2. 2, thus leading to a decrease (an increase) of surface conductivity for an n-type (p-type) semiconductor.[24]

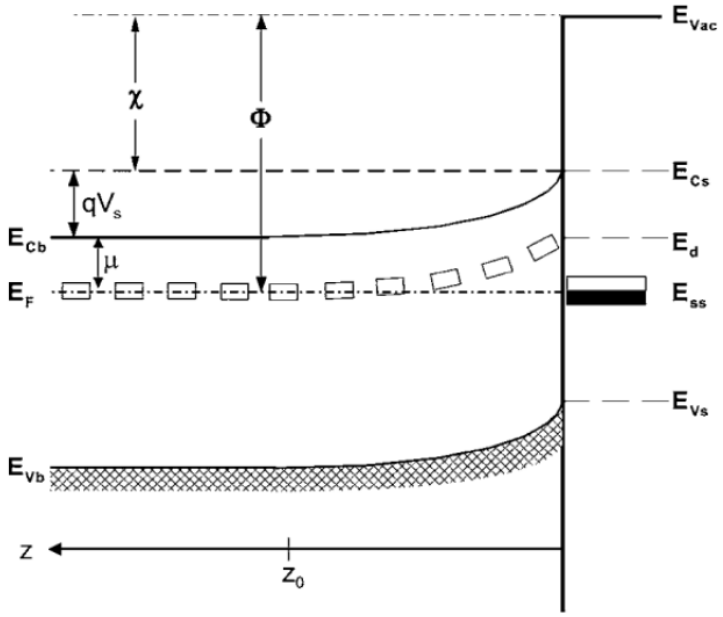

Fig. 2. 2 Surface band diagram after ionosorption of oxygen for an n-type semiconductor. Reproduced from Reference Citation[24] with permission of The Springer.

For a gas sensor, the response (S) to gas can be defined here as

$$S = \frac{\Delta I}{I} = \frac{I_{gas} - I_{air}}{I_{air}} \times 100\ \% \qquad (2)$$

where I_{air} and I_{gas} are the currents of the sensing material in air and the mixture of air and target gas, respectively.

2.3 Current Readout Circuit System

For the application of sensor technology, the readout circuit systems with small form factors are desirable. Medical diagnostics are moving from a stage which relies on physician visits and large centralized laboratories to a new stage where individuals can do their own analysis. For example, researchers who studied giant magnetoresistive (GMR) spin-valve sensors addressed this issue by developing a handheld diagnostic system.[25] The initial

laboratory test station occupying an entire room has been successfully miniaturized into a portable platform. They demonstrated a rapid, multiplex immunoassay platform (nanolab) that is battery-powered and ultraportable (Fig. 2. 3), allowing the assay to be run anywhere.

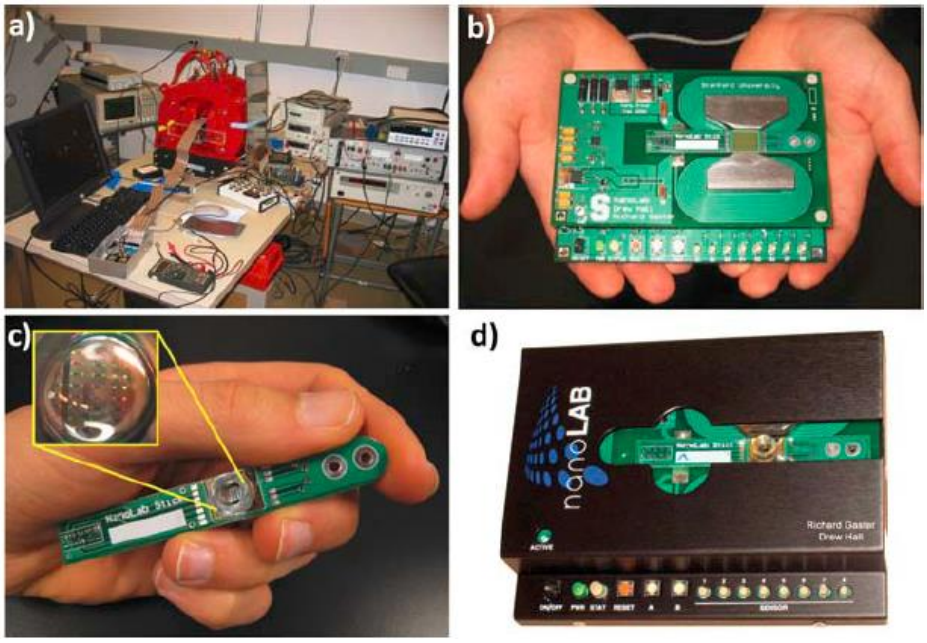

Fig. 2. 3 (a) Initial laboratory test station with sophisticated instrumentation, (b) the proposed handheld point-of-care diagnostic device, (c) Photo of the disposable stick and reaction well where the assay is run. Insert: Inside the reaction well contains an array of GMR sensors, (d) Image of the ‘‘nanoLAB’’ device with case and test stick. Reproduced from Reference Citation[25] with permission of The Royal Society of Chemistry.

Readout circuits can also be used to investigate the intrinsic charging status at dielectric-electrolyte interface (DEI), which plays a critical role for electrofluidic gating in microfluidics and nanofluidics. Through capacitive coupling between the surface charges and the floating extended gate, the signal is analyzied by a low-cost, off-chip extende gate field effect transistor configuration.[26] They developed a readout circuit to characterize the extended gate field effect transistor (Fig. 2. 4 (c)).

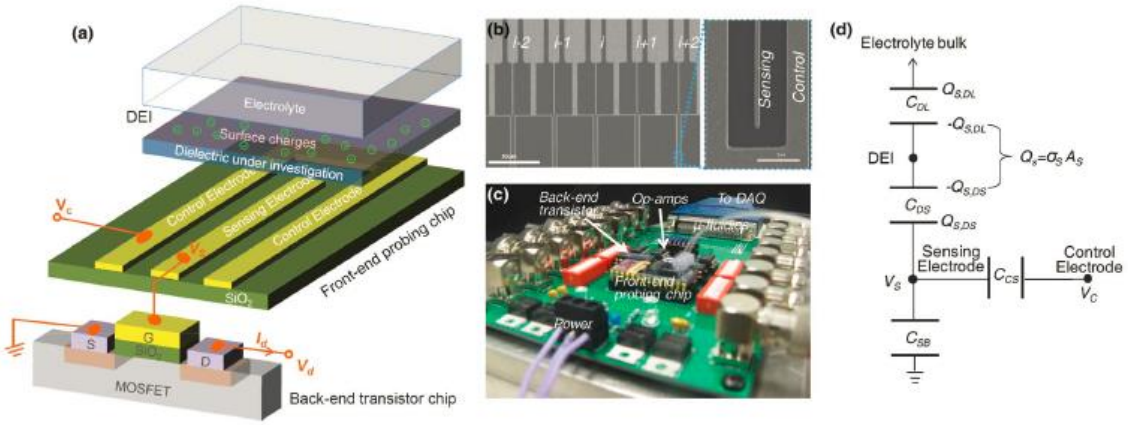

Fig. 2. 4 (a) Schematic of the off-chip extended gate configuration, (b) scanning electron microscope (SEM) of the fabricated front-end probing chip, (c) photo of designed PCB circuit, (d) The equivalent signal circuit model of the system. Reproduced from Reference Citation[26] with permission of The Royal Society of Chemistry.

Researchers used commercially available, metallized paper to construct low-cost, thin, and pliable touch pads.[27] The associated electronics with the individual keys in the touch pads detect changes in capacitance or contact with fingers by using the effective capacitance of the human body and the electrical impedance across the tip of a finger. To measure the changing effective capacitance of the buttons and demonstrate interactive applications using paper-based touch pads, the Arduino platform (UNO and MEGA 2560) was used to develop this project. The paper-based, capacitive touch pads can be applied to disposable and flexible electronics (Fig. 2. 5), boosting the user interface for biomedical instrumentation in the developing world.

(a)

(b)

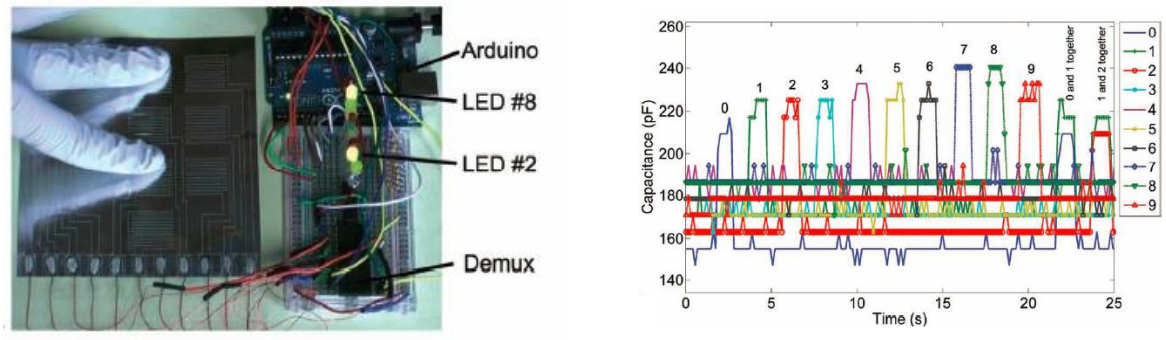

Fig. 2. 5 Touch pad with 10 keys that produced measurable changes in capacitance (a) consturcted keypad and circuit, (b) measurement results. Reproduced from Reference Citation[27] with permission of The John Wiley and Sons.

Gas chromatography (GC) is a frequently used technique in analytical chemistry to qualitatively and quantitatively analyze volatile organic molecules. Procedures typically consist of sample preparation in the laboratory, injection of samples into the capillary column, separation of analytes, and detection by one of several available detection systems, including mass spectrometry (MS). However, for monitoring these dynamic chemical systems, this mode of operation is not convenient. Therefore, researchers have developed a simple device for automated sampling of dynamic chemical processes, and an instantaneous analysis by GC-MS apparatus.[28] The proposed system incorporates two pinch valves, a peristaltic pump, a thermoshaker, and an electronic control unit (Raspberry Pi microcomputer) with several relays, as shown in the Fig. 2. 6. The system not only can control the operation of the pump and actuation of the pinch valves, but it also can initiate GC-MS runs. By characterization with artificial samples, the proposed system has successfully demonstrates the capabilities of

monitoring of transesterification reactions catalyzed by a single microbead, as well as monitoring of the extraction of natural products from plant tissues.

(a) (b)

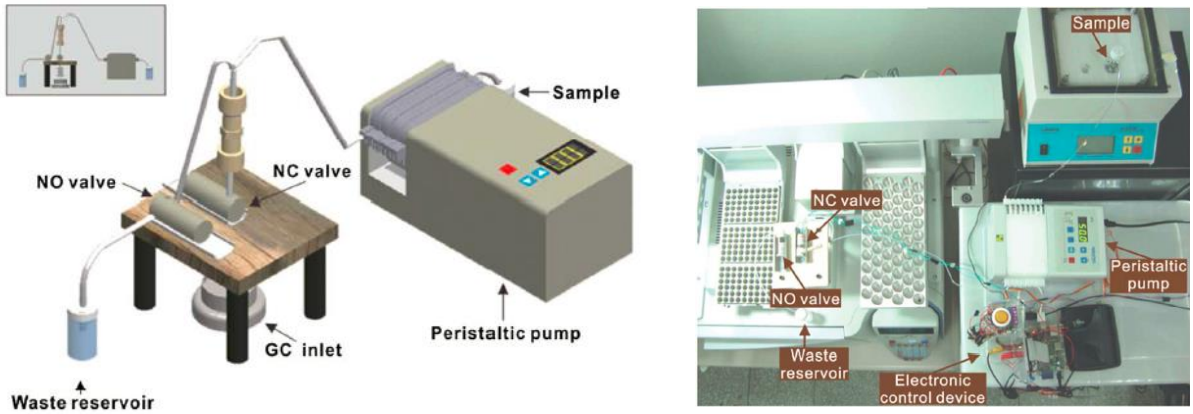

Fig. 2. 6 System for online sampling and sample introduction to GC-MS using pinch valves: (a) schamatic of experimental setup, (b) view of the assembled device. Reproduced from Reference Citation[28] with permission of The Royal Society of Chemistry.

Chapter III: Fabrication of In_2O_3/RGO Gas Sensors and Design of Readout Circuit System

In this chapter, the syntheses and fabrication of In_2O_3/RGO nanocomposite gas sensors are presented first. Different techniques are then performed to characterize the In_2O_3/RGO nanocomposite. Next, the design of readout circuit system is proposed to monitor the electrical current of the gas sensors and provide alert messages. The information of the electronic components used in this circuit and design flow of application programs is included for implementation.

3.1 Preparation of In_2O_3/RGO Gas Sensors

High-quality graphene oxide (GO) was obtained by a modified Hummers' method. The In_2O_3/GO nanocomposite was prepared through a one-step wet chemical method by direct hydrolyzing indium nitrate in the presence of GO. Fig. 3. 1 illustrates the schematic formation pathway for the In_2O_3/RGO nanocomposites. A GO solution was added into an indium nitrate solution with a controlled ratio. The reaction mixture was then heated at 50 ℃ for 4 hours to allow the indium ions to be adsorbed onto GO sheets through coordination interactions with the carboxylic, hydroxyl and epoxy groups.[29] The adsorbed indium ions can then act as the seeds for the subsequent indium ions precipitation through a hydrolysis and dehydration reaction to produce In_2O_3 colloidal particles on GO sheets.[30] The resulted

In_2O_3/GO composites were collected by centrifugation and annealed at a controlled temperature (200 ℃) in a reducing atmosphere that results in In_2O_3/RGO nanocomposites.

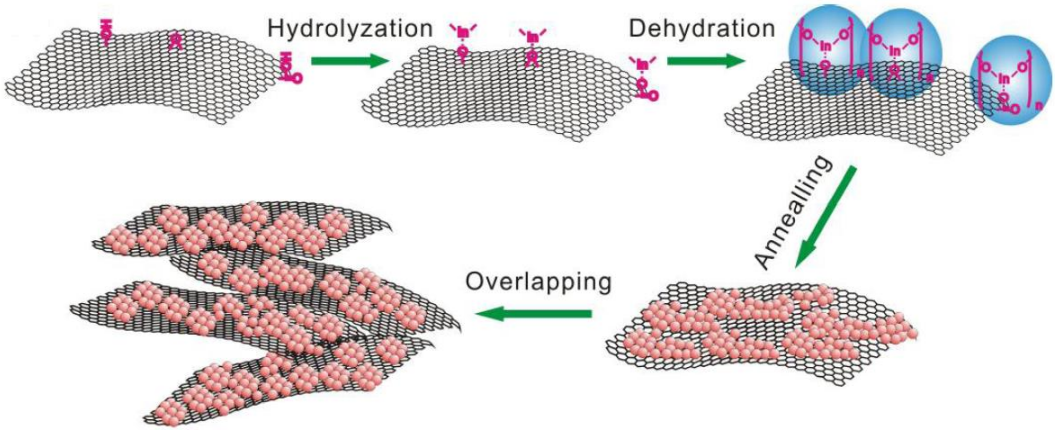

Fig. 3. 1 Schematic demonstration of the formation process of the In_2O_3/RGO nanocomposite.

Transmission electron microscopy (TEM) studies show that the In_2O_3 nanoparticles were uniformly nucleated on the RGO sheet with cuboid morphology of a mean length of 7-8 nm and mean width of 4-5 nm (Fig. 3. 2 (a)). High resolution TEM (HRTEM) studies show clear lattice fringes with a separation distance of 0.179 and 0.292 nm, which can be indexed to In_2O_3 (440) and (222) lattice planes (Fig. 3. 2 (b)).

(a) (b)

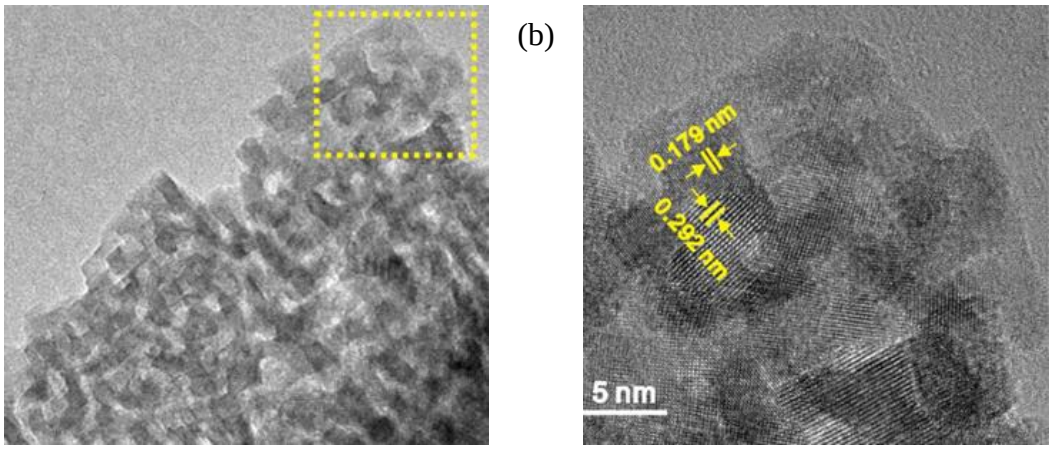

Fig. 3. 2 (a) TEM image of the In_2O_3/RGO nanocomposite, (b) HRTEM image of the selected area in (a).

The x-ray photoelectron spectroscopy (XPS) analysis was conducted in the region of 0-1200 eV to characterize the surface composition and chemical states of the species in the In_2O_3/RGO nanocomposite as shown in Fig. 3. 3. Fig. 3. 3 (a) shows the wide-survey XPS spectrum of the In_2O_3/RGO nanocomposite, which indicates that the product contains In, C and O as the main components with very strong peaks. The presence of In_2O_3 could be further confirmed by the In 3d spectrum (Fig. 3. 3 (b)), in which two symmetrical peaks at 445.2 eV and 452.7 eV were observed corresponding to In $3d_{5/2}$ and In $3d_{3/2}$, respectively. C 1s spectra were used to analyze the degree of reduction of the nanocomposite. Comparing Fig. 3. 3 (c) with Fig. 3. 3 (d), we observed that the peaks at 284.5 and 288.9 eV, which belonged to nonoxygenated ring C and carboxyl groups at the edge of GO, did not decrease obviously, while the peak at 286.7 eV decreased substantially, indicating hydroxyl and epoxy groups on the plane of GO were reduced after H_2 treatment, although the peak did not disappear entirely. Some functional groups still left on RGO sheets including hydroxyl and epoxy groups on the planes and carboxyl groups at the edges, which facilitated the adsorption of CH_4 molecule.

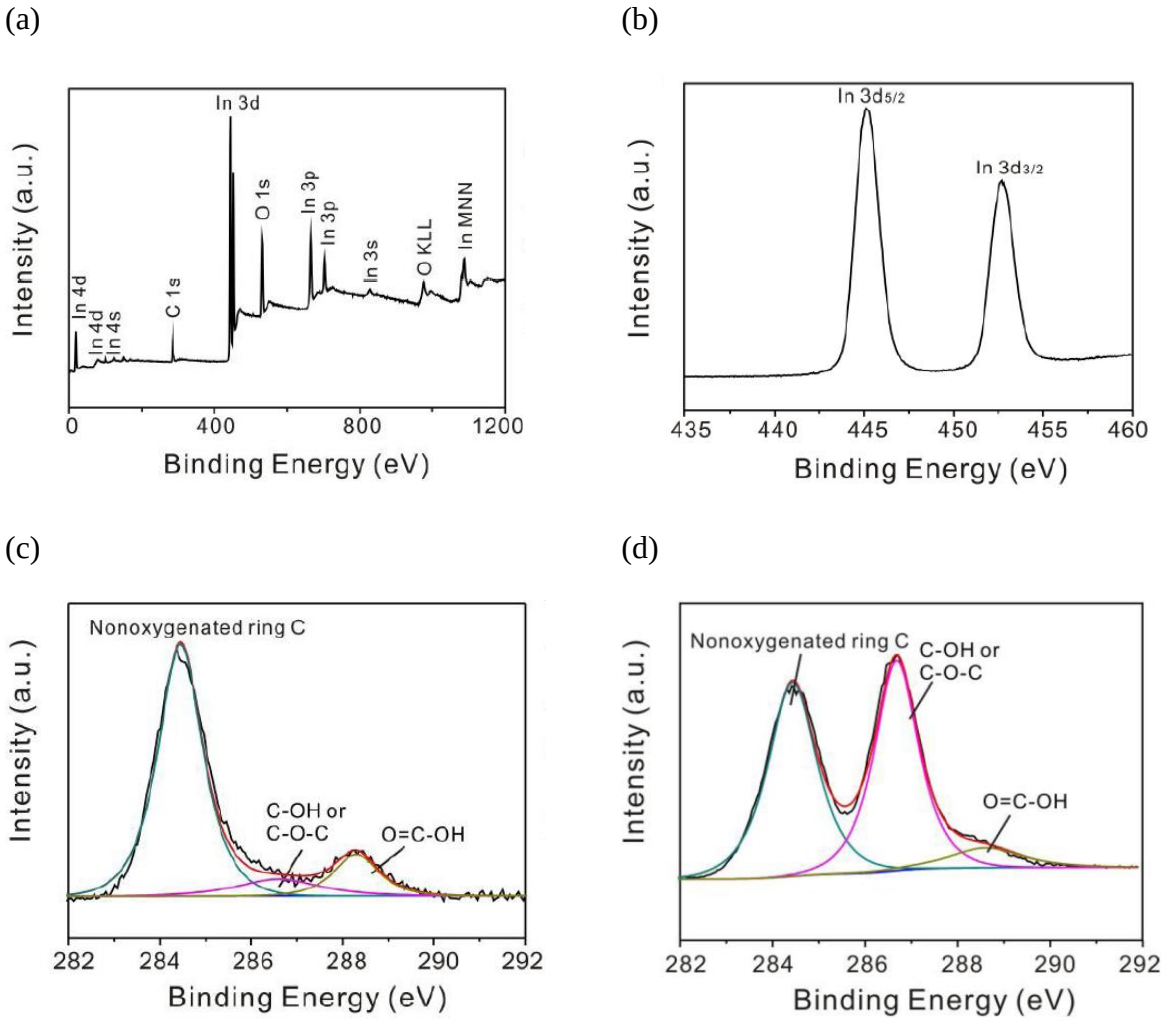

Fig. 3. 3 (a) Wide-survey XPS spectrum of the In_2O_3/RGO nanocomposite, (b) In 3d and (c) C 1s XPS spectra of the In_2O_3/RGO nanocomposite, (d) C 1s XPS spectrum of the GO.

After the preparation of the In_2O_3/RGO nanocomposite film, we coated the materials between two electrodes of ceramic chip carrier, which corresponds to two pins connected with Bayonet Neill–Concelman (BNC) cables for electrical measurements. The completed In_2O_3/RGO gas sensor is shown in Fig. 3. 4.

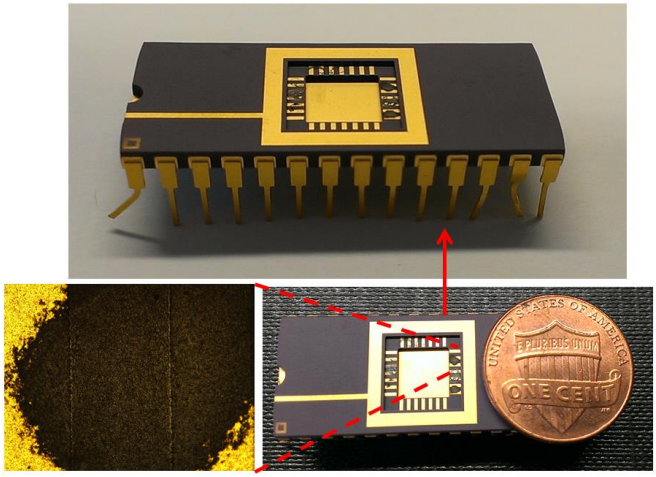

Fig. 3. 4 Gas sensors with In_2O_3/RGO nanocomposite coated on a ceramic carrier.

3.2 Design of Readout Circuit System

3.2.1 Concept of Smart Readout System

The design of this project is dedicated to developing a smart readout system, which not only provides the information of gas sensors but also offers an alert system that notifies users when gas sensors detect an abnormal concentration. Secure shell (SSH) terminal and liquid crystal display (LCD) screen can display the real-time current of gas sensors. Data is also logged on the webpage of the Plotly company. Thus, all users can access and view the electrical current of the gas sensors on a webpage through both computers and smartphones with an Internet connection. Gas sensors can are adopted to detect flammable gases or volatile organic compounds. For this application, it is dangerous when the concentration of flammable gas becomes too high. Therefore, we include the design of an alarm system to highlight this situation. A red LED light will be turned on when the gas sensors detect abnormal concentrations. Furthermore, the Raspberry Pi microcomputer will also send alert

text messages to users so that it instantly reminds users of the monitored gas concentration. All above-mentioned functions are achieved by using the Python programming language. The concept of the developed readout circuits system and its application scenarios associated with gas detection are illustrated in Fig. 3. 5.

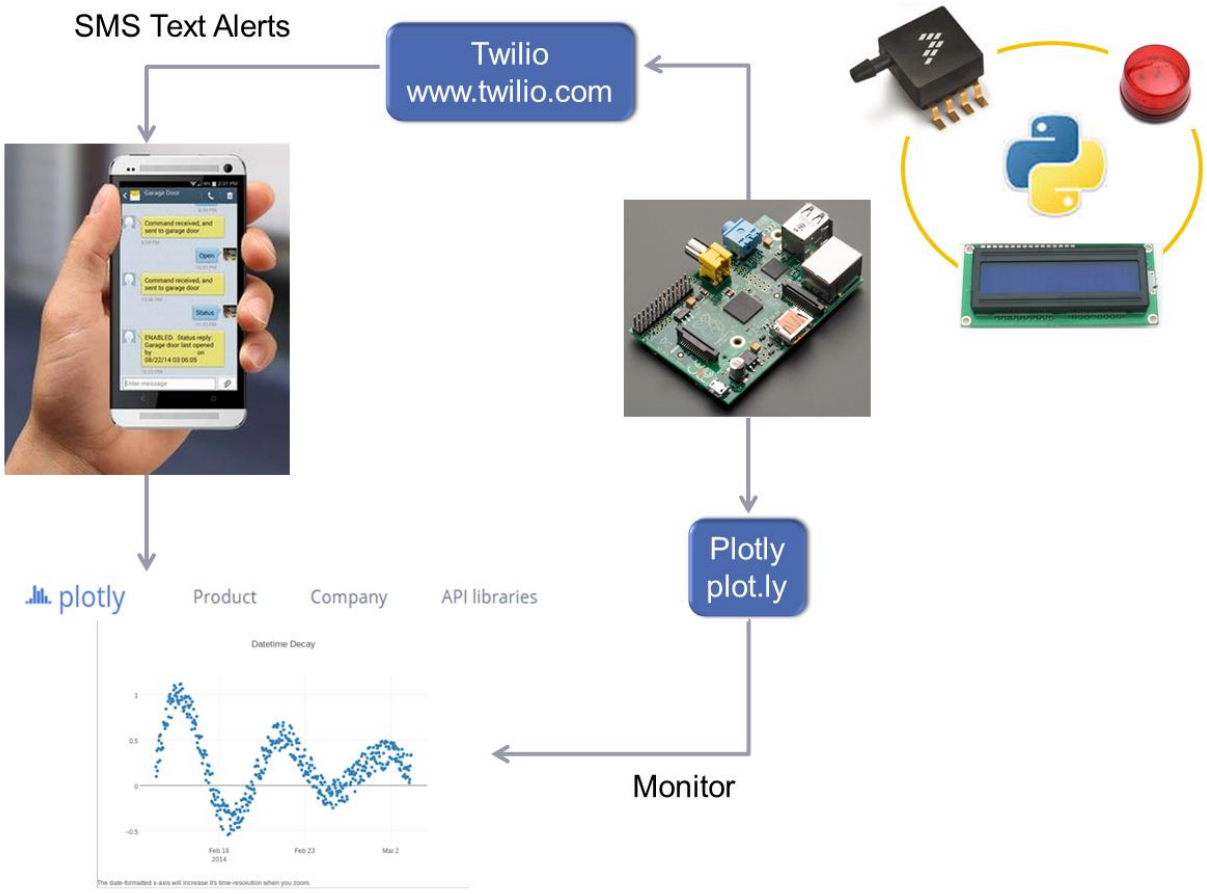

Fig. 3. 5 Concept of the readout system for gas sensor.

3.2.2 Design of Readout Circuit

The proposed readout circuit system was built on a breadboard, which was easier for prototyping. For biosensing applications, the conductance of gas sensors is modulated by gas molecules, thus leading to variation in the electrical current. In other words, the equivalent

resistance of the gas sensors is changed upon exposure to different gas molecules. Thus, gas sensors can be regarded as variable resistors (R_s). Our targets current of interest is at the level of a few μA to hundreds of nA. Raspberry Pi is a Linux-based digital microcomputer, which can be utilized to record and analyze the electrical current. For using digital signal processing in the Raspberry Pi, analog input signals have to be converted to digital signals. Accordingly, an analog to digital converter (ADC) was used to perform this task prior to interfacing with the Raspberry Pi. In order to convert current signal of gas sensors to voltage signal that can be processed by ADC, we adopted the configuration of a current-to-voltage converter which consists of an amplifier and a resistor in a negative feedback loop (R_f) to amplify the signal of interest.[31-33] The readout circuit was expected to provide the flexibility of setting different bias condition for gas sensors. A potentiometer was connected to the 3.3 V supply of the Raspberry Pi. According to the principle of voltage dividers, the potentiometer can thereby supply an adjustable bias (V_s) range from 0 to 3.3 V. All analog voltage signals were collected by a 10-bit ADC in order to convert to digital signals, followed by sending the signals to Raspberry Pi through the General Purpose Input/Output (GPIO) ribbon for data processing. In addition, an LCD screen was connected with the Raspberry Pi and controlled through GPIO pins to display the real-time electrical current of the gas sensors. A red LED was also connected with the GPIO so that it can be turned on/off to indicate alarm situations. A red LED was in series with a 10 kΩ resister to limit maximum current flowing through the LED,

thus preventing LED from burning out. The schematic of the proposed readout circuit system is shown in Fig. 3. 6.

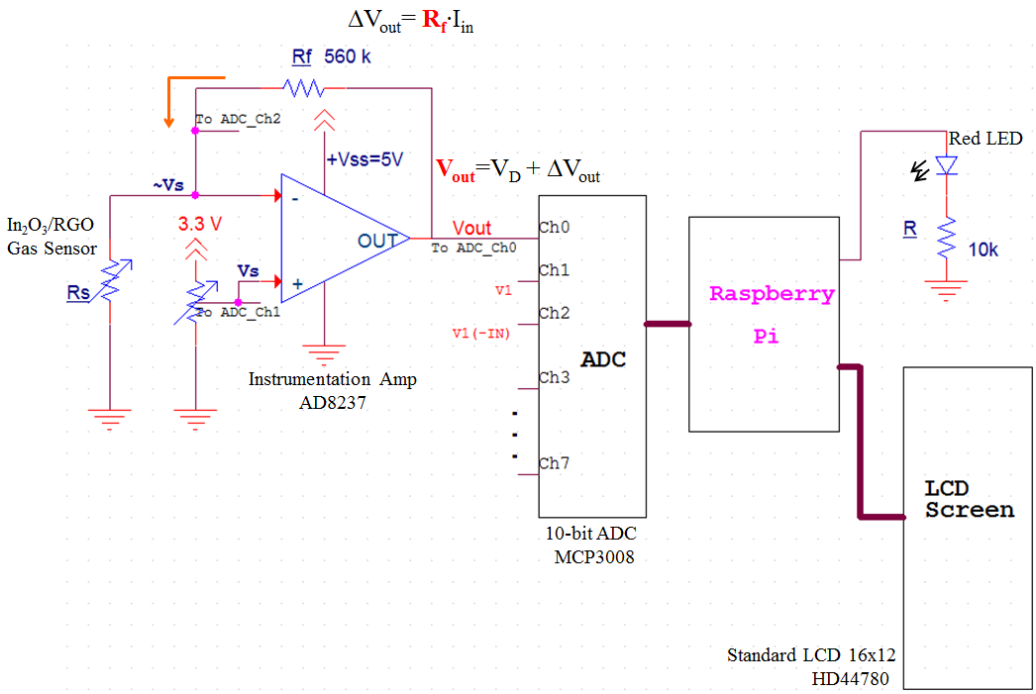

Fig. 3. 6 Readout circuit system featuring a current-to-voltage configuration with a feedback resistor R_f at amplifying stage.

3.2.3 Hardware: Electronics Components

The main electronics components for the data acquisition unit are the data processing system, ADC and instrumentation amplifier (In-Amp). The electronics components used for the prototyping and the function are described as follows.

3.2.3.1 Data Processing System

Raspberry Pi is a digital microcomputer that can be used for data processing and it is commonly utilized to run Linux. Raspberry Pi, Model B, 512M ($39.95) was used for the development of the readout circuit, as shown in Fig. 3. 7. Pi can process the digital signal

from ADC and interface with other electronics components on circuit boards through GPIO pins (see Fig. 3. 8).[34, 35] Raspberry Pi, Model B+, 512M ($39.95) replaced Model B in July 2014 and it was also utilized in this project, when we started including a LCD screen in the readout system. One advantage of Model B+ is that it provides more GPIO pins, which in turn allows users to incorporate more functionality to Pi.

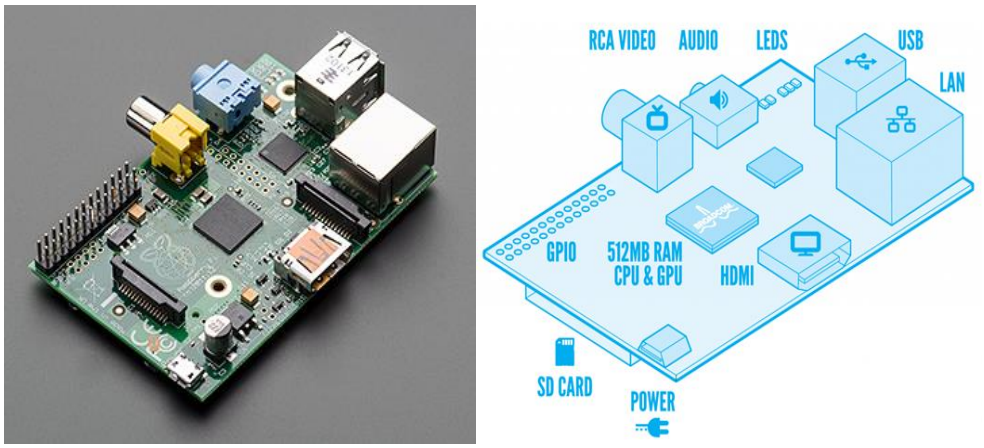

Fig. 3. 7 Raspberry Pi, Model B, 512M. From reference[35]

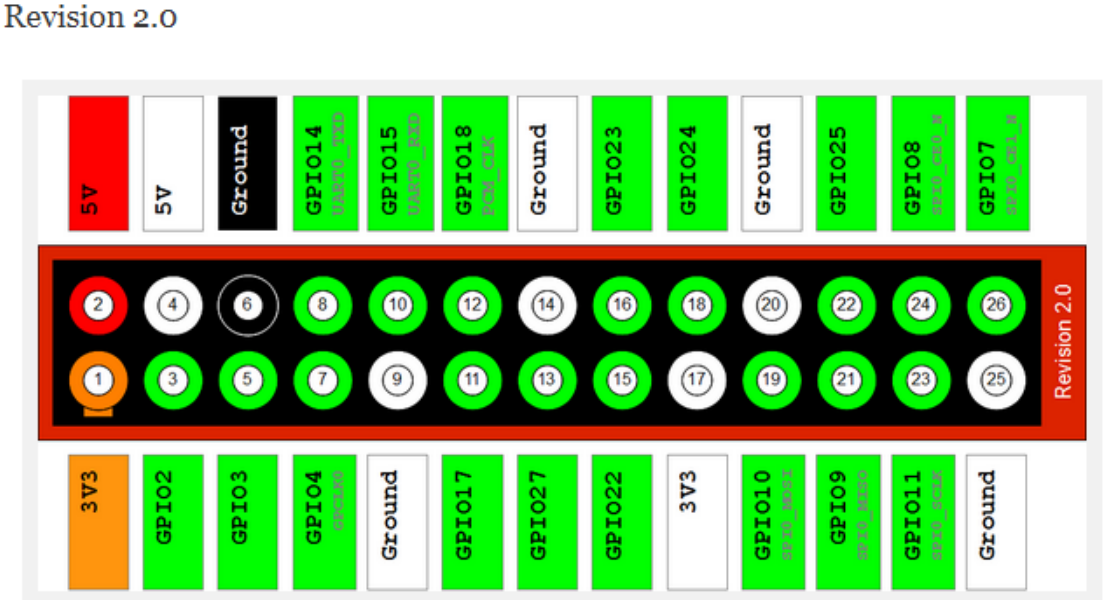

Fig. 3. 8 Raspberry Pi GPIO Layout – Revision 2. From reference[36]

3.2.3.2 Analog to Digital Converter

The ADC can convert analog signals from the gas sensors to a digital signal and then send

that digital signal to the Raspberry Pi for data processing. The 10-bit ADC (Microchip, MCP3008, $2.63) with a serial peripheral interface (SPI) was used for the development of the readout circuit, as shown in Fig. 3. 9. It is worth noting that the maximum input voltage of the ADC cannot exceed V_{REF} which is 5 V for the proposed design.

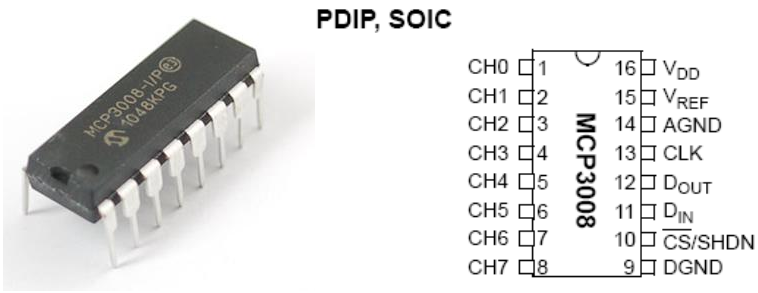

Fig. 3. 9 Product images and pin configuration of 8-channel 10-bit ADC (MCP3008) with SPI Interface. From reference [37]

3.2.3.3 Instrumentation Amplifier

In order to minimize the error caused by amplifying stages and to develop portable readout circuits, an instrumentation amplifier (Analog Device, AD8237, $2.55) with a low input bias current 1 nA and a single power supply was used, as shown in Fig. 3. 10. Gain of amplifiers can be set by two external resistors (see Fig. 3. 11) and the gain is from 1 to 1000 according to the formula $G=1+R_2/R_1$.

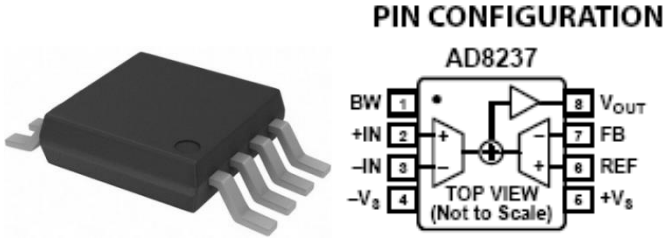

Fig. 3. 10 Product images and pin configuration of instrumentation amplifier (AD8237) with low input bias current. From reference[38]

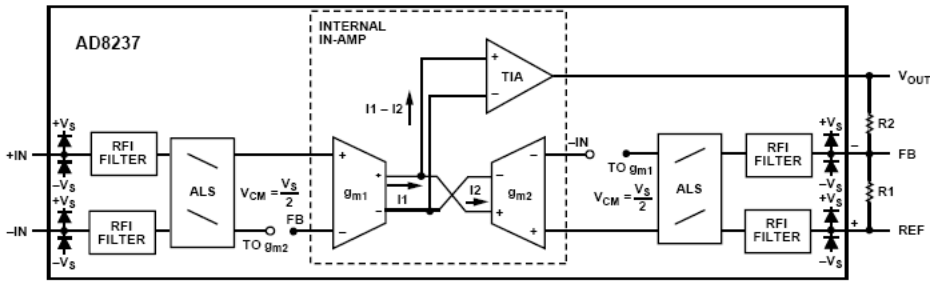

Fig. 3. 11 Simplified schamatic of AD8237 with two external gain resistors (R_1 and R_2). From reference[38]

3.2.3.4 ICs Adapters

Integrated circuits (IC) are put into a protective package to allow easy handling and assembley onto the printed circuit board (PCB). Traditionally, a dual in-line package (DIP) is used, which means the ICs pins are through-hole type. In contrast, the package of the In-Amps (AD8237) is an 8-lead mini small outline package (MSOP), which is just one type of surface-mounted package. The advantage of surface-mounted ICs is that they occupy much less volume than an equivalent DIP. However, MSOP ICs are not easy for prototyping compared with DIP ICs. Thus, we used adapters (Logical Systems Inc., PA-MSD3SM18-08, $5.00) to convert the MSOP to DIP, as shown in Fig. 3. 12.

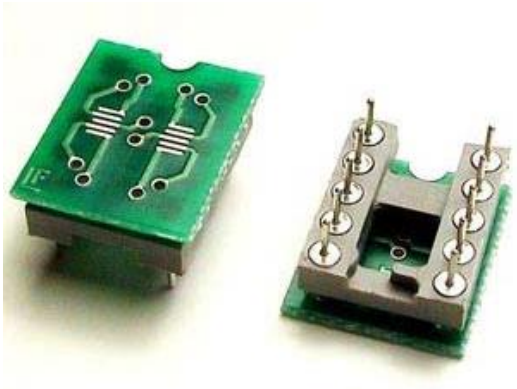

Fig. 3. 12 Product images of adapter 8MSOP-to-8DIP (PA-MSD3SM18-08). From reference[39]

In order to connect MSOP ICs on adaptors, soldering was applied. With the help of flux and soldering gun with sharp tip as well as right amount of solder iron, MSOP ICs can be fixed to adapters very well. Fig. 3. 13 (a) and (b) show the process of applying flux and using soldering gun, respectively. Fig. 3. 13 (c) is the completed device of In-Amp.

(a) (b) (c)

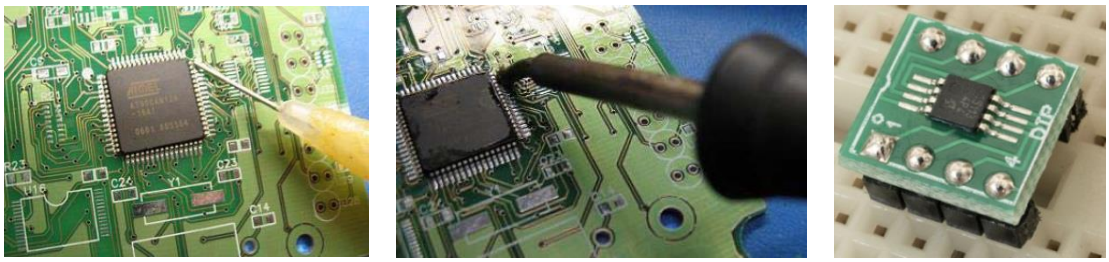

Fig. 3. 13 (a) Applying flux around the pins of ICs, (b) using soldering gun with right amount of solder iron to solder the pins of ICs, (c) complete In-Amp with 8-lead DIP configuration.

3.2.3.5 Dual N-Channel Matched Pair MOSFET Array

Prior to the measurement of In_2O_3/RGO nanocomposite gas sensors, commercial N-Channel MOSFETs (Advanced Linear Devices, ALD1116, $2.02) were adopted as a control experiment to test the function of proposed circuit. A Dual N-channel MOSFET array and its product image is shown in Fig. 3. 14. The drain current of ALD1116 is in milliamp range and its output characteristic is shown in Fig. 3. 15.

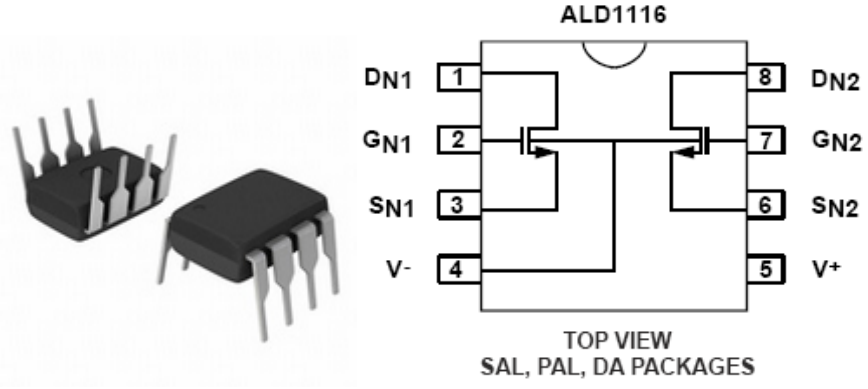

Fig. 3. 14 Product images and pin configuration of ALD1116. From reference[40]

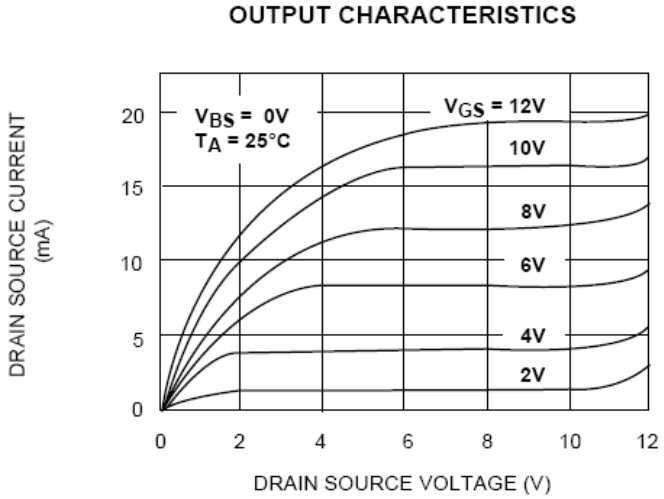

Fig. 3. 15 Output characteristics of NMOSFET (ALD1116). From reference[40]

3.2.4 Software: Python Programming

Python is a widely used general-purpose, high-level programming language. Python programming was adopted herein to program the Raspberry Pi. We can edit codes through SSH or text editors, such as gedit (for Linux computer) or Notpad++ (for Windows computer). The flow chart of the basic concepts of our Python script for the proposed readout circuit is shown in Fig. 3. 16.

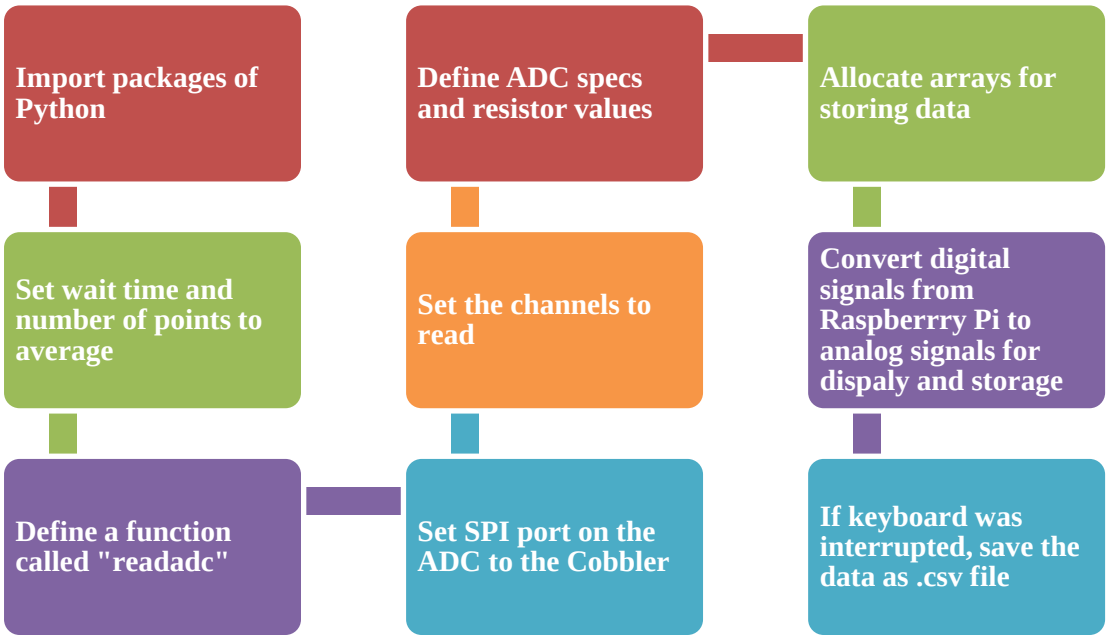

Fig. 3. 16 Flow chart of Python script for the data acquisition unit in the proposed readout circuit.

In order to use other functions, we imported different open-source packages in the beginning of the Python program. Because Raspberry Pi kept sampling the voltage signals from ADC during measurements, we can specify the wait time (sampling time interval) and number of data points we want to average. Signal read by ADC was interpreted by a function called “readadc.” According to the specification from MCP3008, we used readadc to compare different states of pins from input channels (Clock, MOSI, MISO and CS) to generate binary signals (Fig. 3. 17). One note is that the ADC provides two different modes (Single and Differential) to capture the signal. We can define the mode we want in readadc by commandout |= 0x18 or 0x10 for single or differential mode, respectively.

```
# read SPI data from MCP3008 chip, 8 possible adc's (0 thru 7)
def readadc(adcnum, clockpin, mosipin, misopin, cspin):
        if ((adcnum > 7) or (adcnum < 0)):
                return -1
        GPIO.output(cspin, True)

        GPIO.output(clockpin, False)  # start clock low
        GPIO.output(cspin, False)     # bring CS low

        commandout = adcnum
        # Value below selects the ADC capture mode: 0x10 for diff mode, 0x18 for single-ended mode.
        commandout |= 0x18  # start bit + SGL/DIFF bit
        commandout <<= 3    # we only need to send 5 bits here
        for i in range(5):
                if (commandout & 0x80):
                        GPIO.output(mosipin, True)
                else:
                        GPIO.output(mosipin, False)
                commandout <<= 1
                GPIO.output(clockpin, True)
                GPIO.output(clockpin, False)
```

Fig. 3. 17 Definition of a function called “readadc” which can generate binary signals for ADC and set the mode (single/differential mode) of ADC.

Then, we set the SPI ports on ADC to the pins on Cobbler. Also, we defined the ADC specification, such as 10-bits resolution and V_{REF}, as well as the resistor values that was used for the feedback resistor (R_f). An array with six dimensions was allocated in advance for storage of data, such as time, V_{out}, V_s, V(-IN), I_{out} and so on.

A while-loop was utilized to continuously sample the signals from the ADC and display the results (Fig. 3. 18) until the keyboard interrupted the loop. ADC signals (0~1024) are scaled to the analog signal in the range of 0 to V_{REF}. Then, the electrical current of gas sensors (I) is calculated by ohm’s law ($\Delta V/Rf$), where ΔV is equal to V_{out} ─ V(-IN). After data conversion, the data is stored in an array and the average value or standard deviation is displayed on the SSH screen according to the formulas specified in the Python script. If the keyboard was interrupted by Ctrl+C, the program will stop and save the measured data in

common separate value (.csv) file.

```
try:
    while True:
        count += 1
        # read the analog pin
        Vout_value = readadc.readadc(Vout_adc, readadc.PINS.SPICLK, readadc.PINS.SPIMOSI, readadc.PINS.SPIMISO, readadc.PINS.SPICS)
        Vd_value = readadc.readadc(Vd_adc, readadc.PINS.SPICLK, readadc.PINS.SPIMOSI, readadc.PINS.SPIMISO, readadc.PINS.SPICS)
        VIN_value = readadc.readadc(VIN_adc, readadc.PINS.SPICLK, readadc.PINS.SPIMOSI, readadc.PINS.SPIMISO, readadc.PINS.SPICS)
        elapsed_time = time.time() - start_time

        Vout = float(Vout_value) * VRES/1000  # scale the output from mV to V
        Vd = float(Vd_value) * VRES/1000
        VIN = float(VIN_value) * VRES/1000
        I = float((Vout - VIN)/Rf)*1e6  # scale I (uA)

        data[0].append(elapsed_time)
        data[1].append(Vout)
        data[2].append(Vd)
        data[3].append(VIN)
        data[4].append(I)
        Voutdata = data[1][-n:]
        Vout_ave = sum(Voutdata)/n
        Vout_stdev = (sum([(x - Vout_ave) ** 2  for x in Voutdata])/n) ** 0.5
        Vddata = data[2][-n:]
        Vd_ave = sum(Vddata)/n
        Vd_stdev = (sum([(x - Vd_ave) ** 2  for x in Vddata])/n) ** 0.5
        VINdata = data[3][-n:]
        VIN_ave = sum(VINdata)/n
        Idata = data[4][-n:]
        I_ave = sum(Idata)/n
        data[5].append(I_ave)
        I_stdev = (sum([(x - I_ave) ** 2  for x in Idata])/n) ** 0.5


        stream.write({'x': elapsed_time, 'y': I_ave})   # write the data to plotly
        sys.stdout.write('\r'+' '*100+'\r')     #Expression setting on the screen, cannot exceed 100
        sys.stdout.write("t=%0.2fs Vout_ave=%.3fV; Vs_ave=%.3fV; V(-IN)_ave=%.3fV; I_ave=%.3fuA\r" % (elapsed_time,Vout_ave,Vd_ave,
        VIN_ave,I_ave) )
        sys.stdout.flush()
        time.sleep(wait)
```

Fig. 3. 18 Parts of while-loop in Python script.

3.2.5 Interfacing with Raspberry Pi

By default, the Raspberry Pi is set in dynamic IP mode and is denoted by Dynamic Host Configuration Protocol (DHCP), which will be automatically given an IP address by the router when the Pi is connected to a network. Whenever the Pi is removed from the network (i.e. turning it off), the temporary IP can change when next time Pi is connected to the network. Having a static IP can make repeated access to the Raspberry Pi much simpler. We can set a static IP in the network configuration through nano text editor. The command line is "sudo nano /etc/network/interfaces", as shown in Fig. 3. 19.

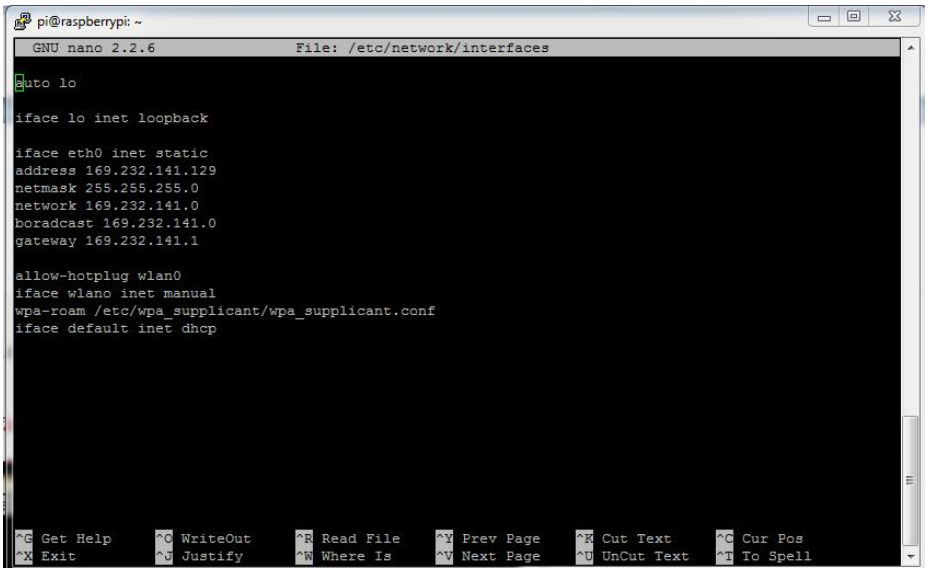

Fig. 3. 19 Editing network configuration to set a static IP.

Since Pi has the same IP address every time, we can connect Pi by assigning the IP address in SSH client program. Putty can be used to open an SSH in Windows operating system. The IP address is “169.232.141.129” for this case and port is “22” (Fig. 3. 20).

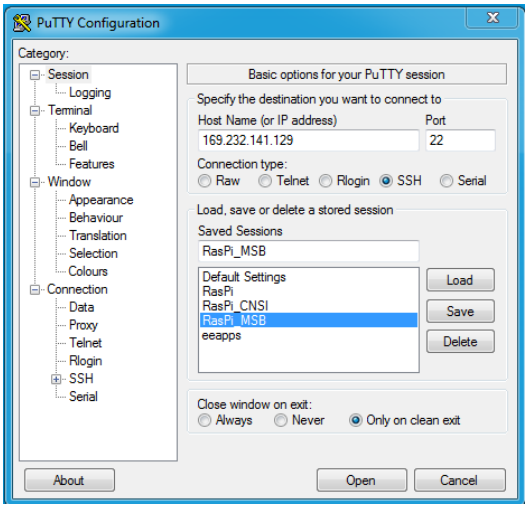

Fig. 3. 20 Setting on the Putty configuration.

After successful connection to the Raspberry Pi, the login account is “pi” and the password is “raspberry”. We also can use virtual network computing (VNC) for remote

access of the Raspberry Pi. The desktop background and interface are shown in Fig. 3. 21.

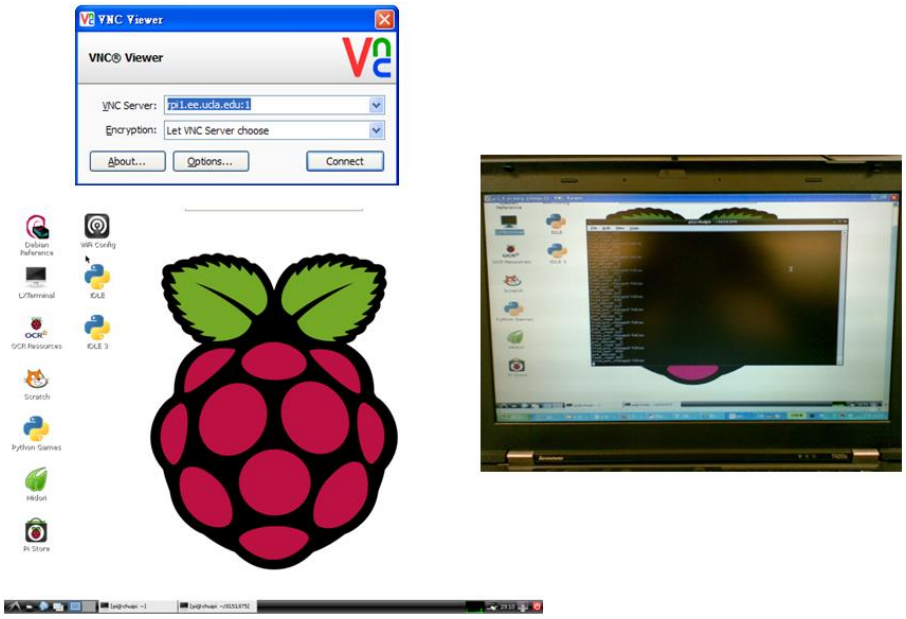

Fig. 3. 21 VNC login interface and the desktop background of Raspberry Pi.

Then, we have to change the directory to the folder containing the files we want to execute by commanding "**cd** <folder name>". By default, files are saved in home/pi. Prior to running the program, we have to make .py file executable by commanding "**chmod +x** <file name>". We can then start executing the program by entering "**sudo ./**<file name>", while using "**Ctrl+C**" to stop the program. The results of execution on the SSH terminal is shown in Fig. 3. 22.

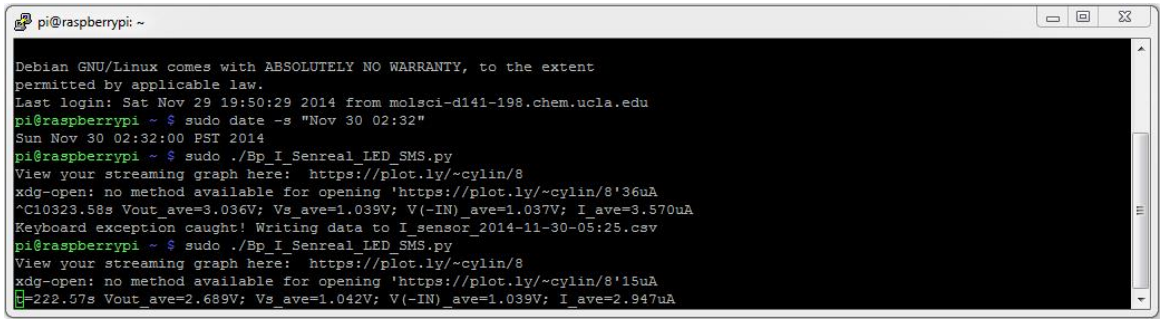

Fig. 3. 22 Results of measurements of current displaying on the SSH screen.

After running the program, FTP was used to access the saved files, as shown in Fig. 3. 23. The files were saved as .csv format and they were stored in the directory (home/pi).

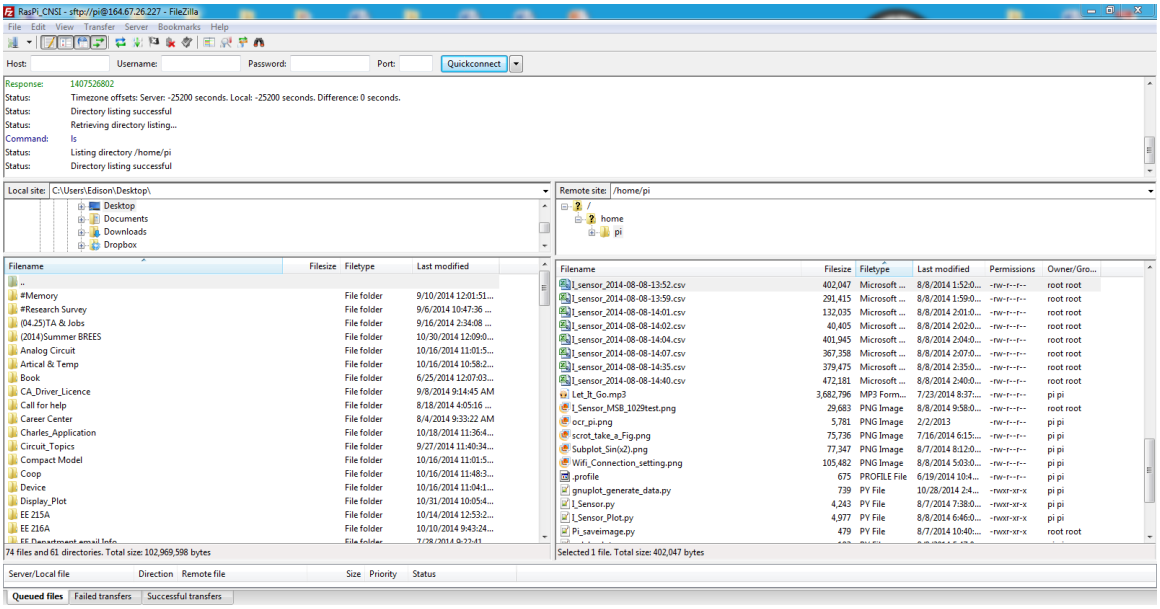

Fig. 3. 23 Using SFTP (FileZilla) to access the files stored in Raspberry Pi.

3.2.6 Construction of LCD Display

Standard liquid crystal display (LCD) 16x2 was adopted to show the real-time current of gas sensor (Fig. 3. 24). We connected an inexpensive HDD44780 compatible LCD to the Raspberry pi using 6 GPIOs. The LED backlight for the LCD should be used instead of an Electroluminescent (EL) backlight because EL backlights draw too much current (200 mA) compared with LED backlights (10-40 mA). We used a 10 kΩ potentiometer to control the contrast of LED backlights. Most LCDs have a strip of 16 pins on the top, so we need to solder the header to the LCD.

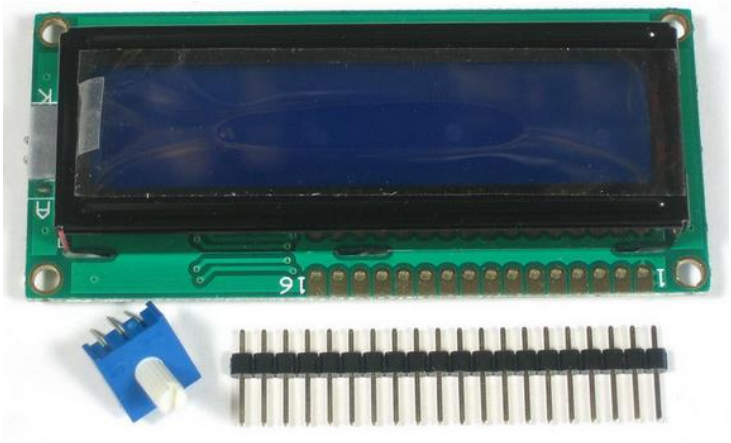

Fig. 3. 24 Photo of 16x2 LCD screen.

The wiring pins are summarized in Table 1 and Fig. 3. 25.[41] The **data pins** send data to the display (toggled high/low). Only the write mode was used, which means it is not reading any data. The **register select** pin has two uses. When pulled low it can send commands to the LCD (like position to move to, or clear the screen). This is referred to as writing to the instruction or command register. When toggled the other way (1) the register select pin goes into a data mode and will be used to send data to the screen. The **Read/Write** pin will be pulled low (write only) as we only want to write to the LCD based on this setup. The **enable pin** will be toggled to write data to the registers.

Table 1 Summary of LCD Pinout

Pin number of LCD	Connected to	Note
1	Ground	
2	+5 V	V_{CC}: 5 V not 3.3 V
3	Middle of the potentiometer	Contrast adjustment (Vo) from potentiometer
4	Cobber #25	Register Select (RS). RS=0: Command, RS=1: Data
5	Ground	Read/Write (R/W), Not used herein
6	Cobber #24	Clock: Falling edge triggered
7	Not used	Bit 0
8	Not used	Bit 1
9	Not used	Bit 2
10	Not used	Bit 3
11	Cobber #23	Bit 4
12	Cobber #17	Bit 5
13	Cobber #21	Bit 6
14	Cobber #22	Bit 7
15	+5 V	Backlight LED Anode (+)
16	Ground	Backlight LED Cathode (-)

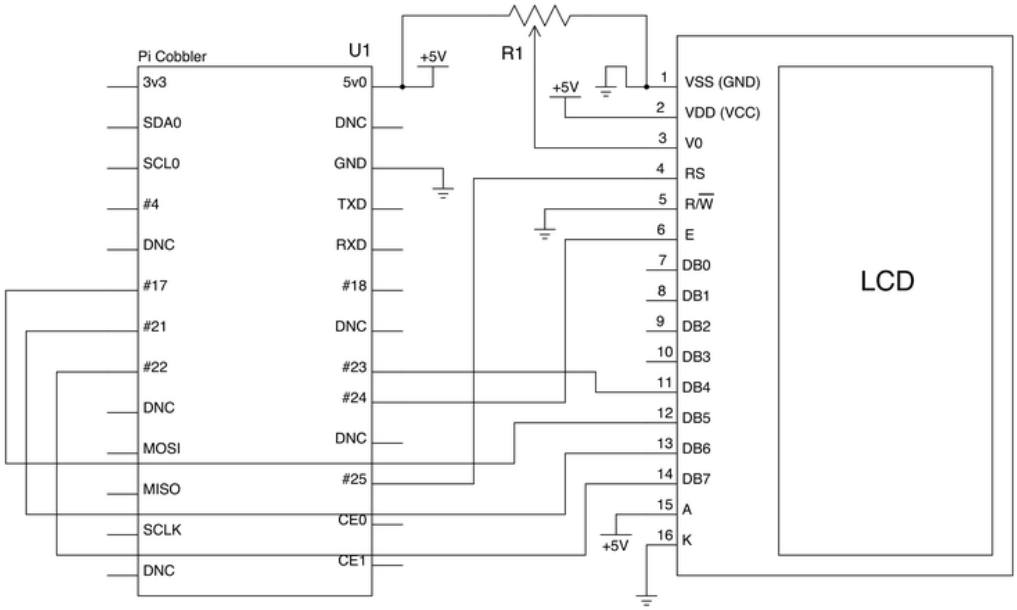

Fig. 3. 25 Schematic of the wiring diagram. From reference[41]

3.2.7 Design of Alarm System

To have an alarm message when gas concentration goes too high, we decided to incorporate a red LED light and short message service (SMS) text messaging system. If the electrical current level reads by the readout circuit is above a certain value, this indicates that the corresponding gas concentration is too high. This situation will trigger the alarm system to light up the red LED and send an alert text message. Users can monitor the real-time current through either the SSH terminal or LCD screen. Furthermore, all the electrical current data is logged on the webpage, so users can observe the real-time plots of the gas sensors through the Internet by using either computers or smart phones. After the problem of abnormal gas concentration is resolved, user will receive a SMS text message to notify the release of the alarm.

Chapter IV: Measurement

In this chapter, the comparison of measured current of commercialized NMOSFET between the proposed readout circuit and SMU is described in the first part. The second part illustrates the testing results of different units in the proposed system; the last part demonstrates the experimental setup for gas sensing including the designed test chamber.

4.1 Current Measurement with NMOSFET

To verify the function of the proposed readout circuit, a commercialized N-channel MOSFET (ALD1116) was used as a target device for current measurement (Fig. 4. 1). The gain of In-Amp was set as G= 1001 and the feedback resistor (R_f) was 552 Ω. A potentiometer was adopted to provide adjustable V_g. Drain bias was set by the voltage of inverting node (V_{-IN}) of the amplifier. Due to the property of virtual ground, when the gain of the amplifier is large, voltage of the inverting node will be approximately equal to one of the non-inverting node (i.e. $V_{-IN} = V_{IN}$). Therefore, the drain bias can be equivalently set by V_{IN}, which was 3.3 V for test purposes. It should be noticed that the input voltages for the ADC have to be smaller than the reference voltage of the ADC ($V_{input} < V_{REF}$) to ensure normal operation. In our setting, V_{REF} is 5 V and thus the range of V_g, V_d and V_{out} are safe for the ADC.

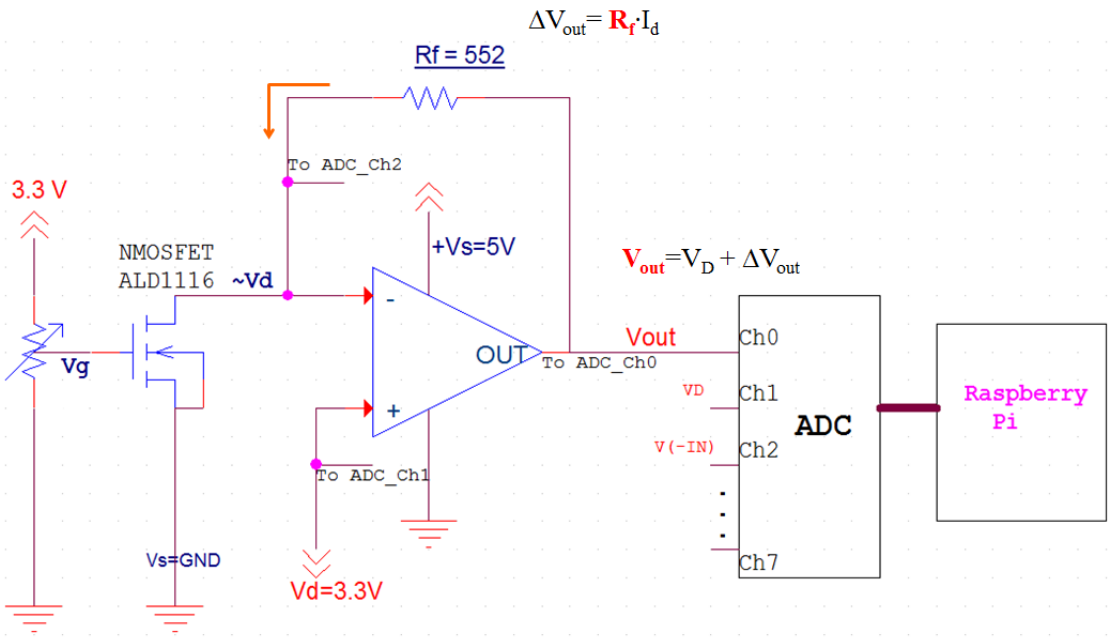

Fig. 4. 1 Equivalent circuit of the proposed readout circuit system.

The circuit measurements were performed by a SMU (B2902A) and compared with the proposed system using the Raspberry Pi. The implementation of the proposed readout circuit and experimental setup are shown in Fig. 4. 2.

(a) (b)

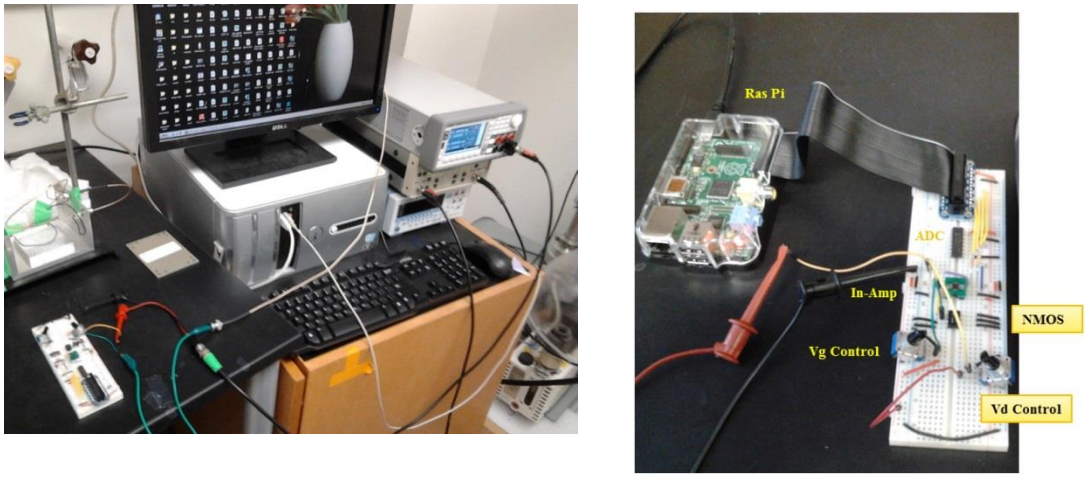

Fig. 4. 2 Current measurement performed (a) by the SMU (Agilent B2902A), (b) by proposed the readout system using the Raspberry Pi.

Given V_d=3.3V and V_s=GND, V_g was varied from 0 V to 3.3 V. Generated drain currents mostly passed through the inverting terminal (-IN) of In-Amp and flowed through R_f to V_{out}. The voltage difference across the R_f was analyzed by ADC. The voltage difference ($\Delta V = V_{out} - V(\text{-IN})$) divided by R_f was the calculated drain current ($I_d = \Delta V_{out}/R_f$) read by Raspberry Pi. The measuring results from Raspberry Pi were compared with the currents read by the SMU, as shown in Fig. 4. 3. The drain currents are within a few milli-ampere ranges, which are consistent with the data from the specification. Furthermore, drain currents read by the Raspberry Pi and SMU are in a good agreement, which demonstrate the capability of the current measurement of the proposed readout circuit system.

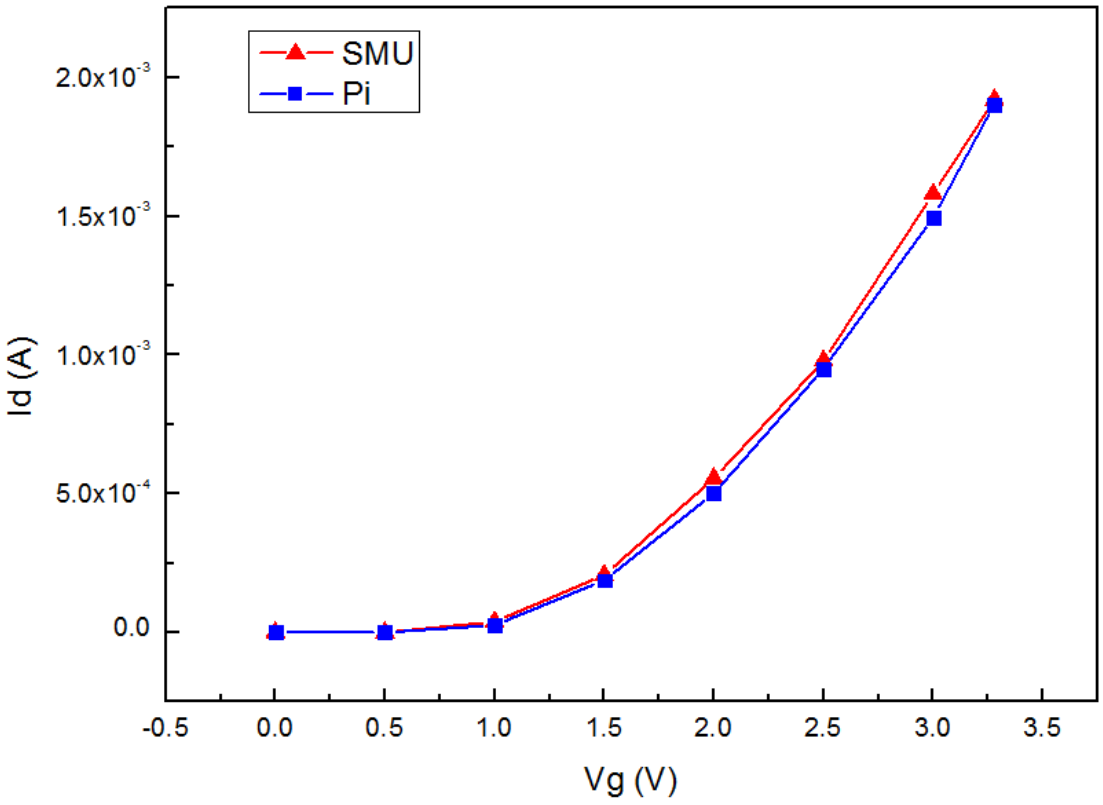

Fig. 4. 3 Comparison of current measurement between SMU and proposed readout circuit.

4.2 Data Display on the Webpage

Since the recorded voltage and current data were saved in .csv files, we can further process the data through Excel. In order to make the data more accessible, we planned to store the data on a webpage so that users can view the data through the Internet directly. There were two methods to achieve this idea. The first one was to plot the data recorded in the .csv file and show the graph on a static IP address. By installing the Matplotlib (see Fig. 4. 4), the microcomputer can process the data, generate a plot as a .png image, and show the plot to the designated IP address automatically (Fig. 4. 5).

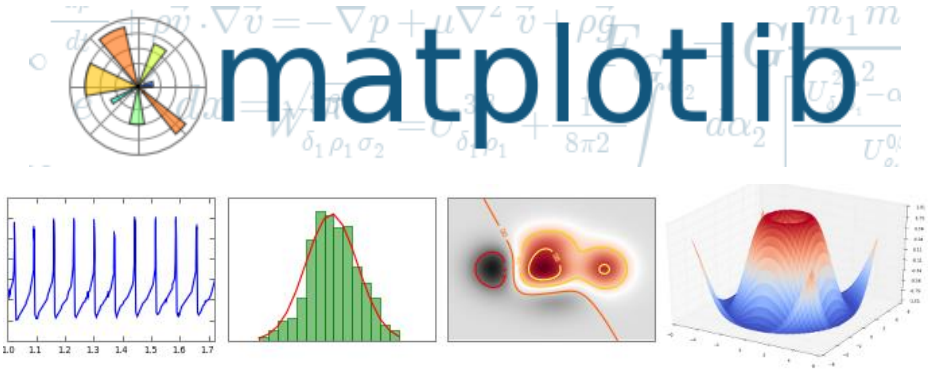

Fig. 4. 4 Matplotlib is an open source package, which provides Python 2D plotting library.

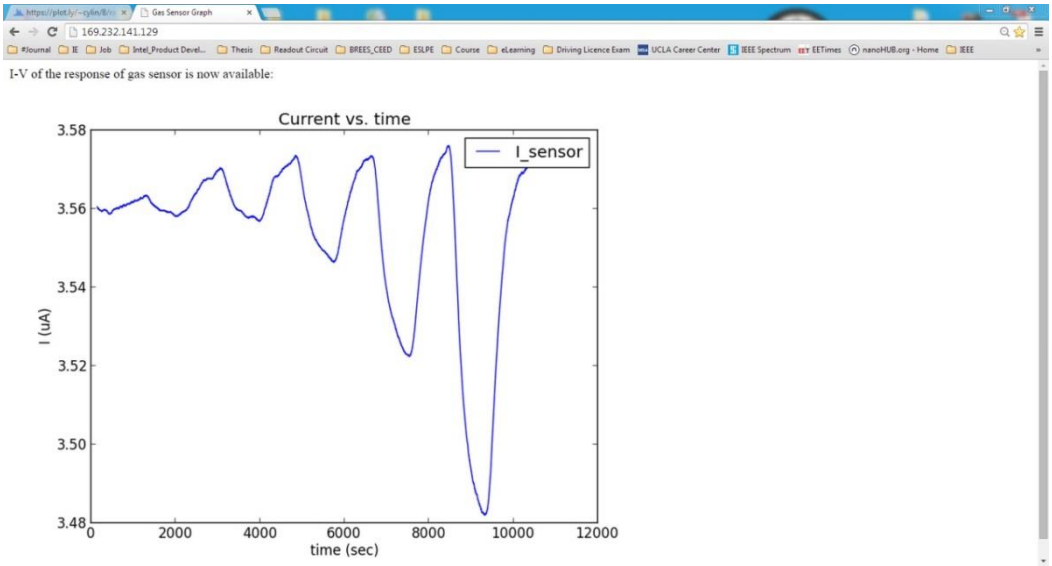

Fig. 4. 5 Measured current displaying on a static IP after the running program ends.

The disadvantage of the first method was that users can only view the plot after the execution of the program was completed. For monitoring the electrical current of a gas sensor, we wanted to have the real-time current data displaying on the webpage. This can be done by using the microcomputer itself. However, the efficiency of running the program is too low, delaying the time of data acquisition. Another way of achieving real-time web display is to share the load of running the program by logging the data online to a separate website. Plotly is a website that provides online analytics and data visualization, as shown in Fig. 4. 6.

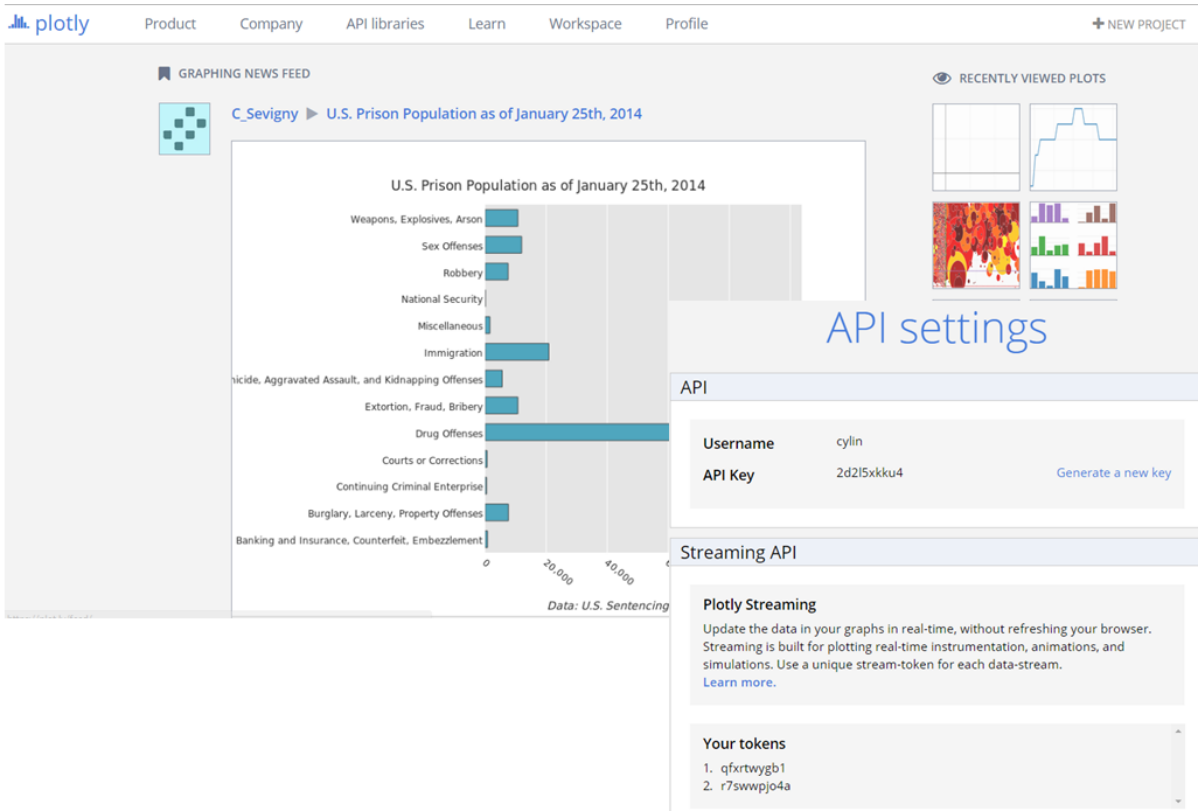

Fig. 4. 6 Plotly provides online graphing, analytics, and statistic tools.

According to the application programming interface (API) settings provided to registered account users, we can specify this information in the program code, allowing

measured data to log onto the Plotly website directly. The plot was shown in the URL according to the information given in the SSH terminal, as indicated Fig. 4. 7. The real-time measuring results displaying on the Plotly website are shown in and Fig. 4. 8.

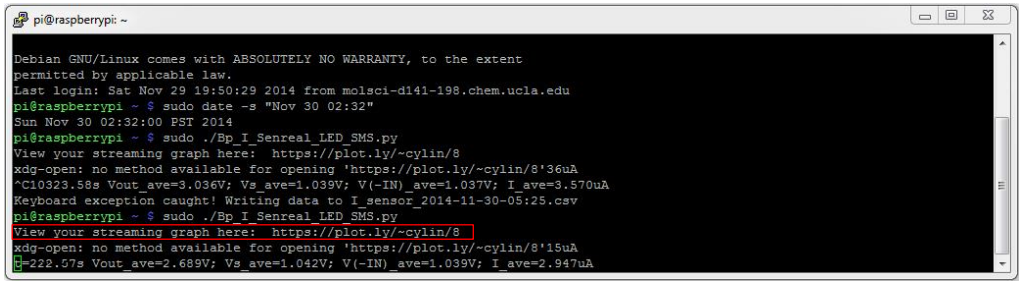

Fig. 4. 7 URL information of the plot (red rectangle) given in the SSH terminal.

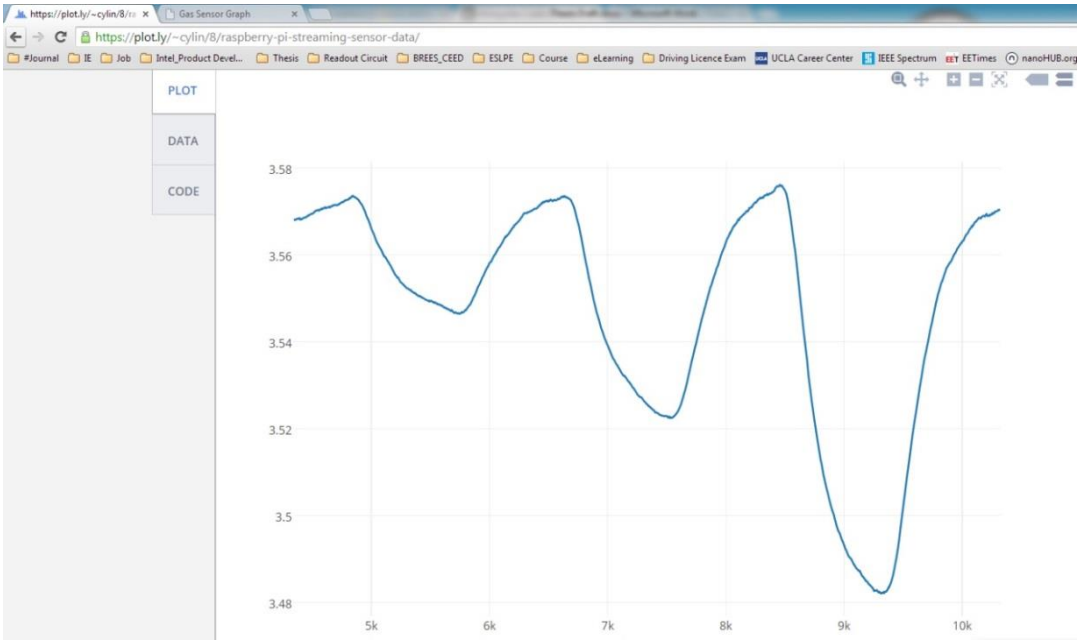

Fig. 4. 8 Real-time current displaying on the website of Plotly.

4.3 LCD Screen Display

In addition to the data displayed on the webpage, the standard liquid crystal display (LCD) 16x2 was adopted to show the real-time electrical current of gas sensor. We connected an inexpensive HDD44780 compatible LCD to the raspberry pi using 6 GPIOs. We used a 10

kΩ potentiometer to control the contrast of LED backlights. The potentiometer was adjusted until words were clearly shown on screen. For test purpose, we assumed the system was used in two channel configurations to record current values from two different sensors (In_2O_3/RGO and ZnO/RGO), as shown in Fig. 4. 9.

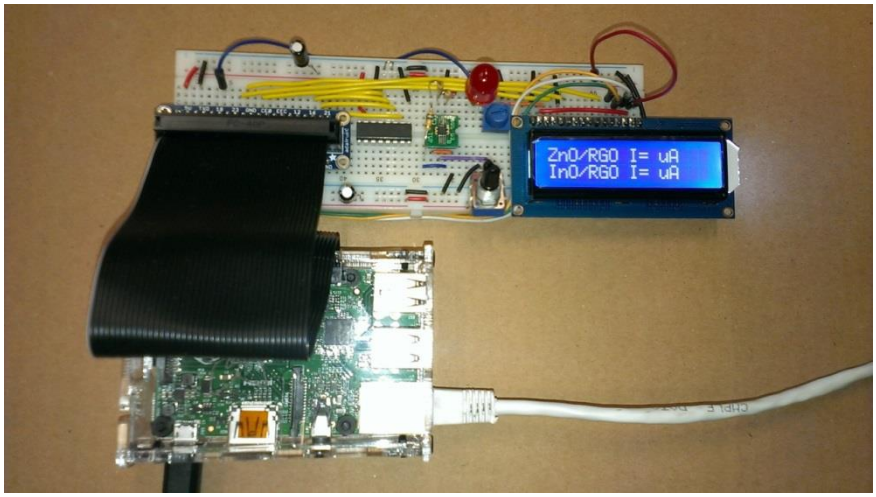

Fig. 4. 9 Setup and test message of the LCD screen.

In the development of LCD display, we found that the Raspberry Pi, Model B cannot provide enough GPIO pins for the LCD, ADC and LED simultaneously. The LCD is connected with six GPIOs, ADC uses four GPIOs and LED needs one GPIO. If the ADC and LCD share the same GPIOs, this will cause a confliction between them, leading to incorrect values. Thus, we changed to use Model B+, which provides up to 40 pins. One of the advantages of upgraded Model B+ is that it has more GPIOs enable users to add more functionality to the Raspberry Pi. The real-time display of the electrical current of the In_2O_3/RGO gas sensor is shown in Fig. 4. 10.

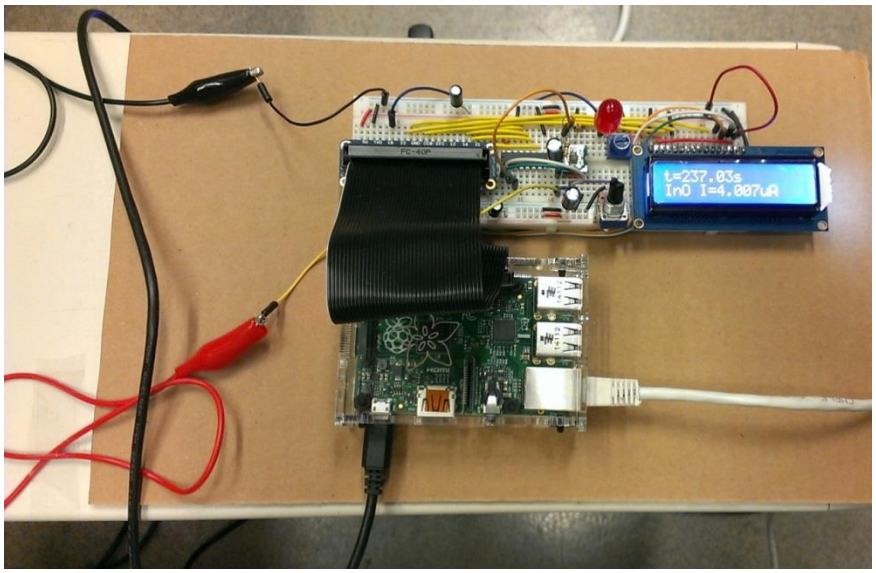

Fig. 4. 10 Real-time display of current on LCD screen.

4.4 LED Alarm Signal

The current variation (ratio) of gas sensors increases as the gas concentration increases. A red LED is used to present an alarm situation when gas concentration goes too high. The GPIO pin can output nominal 3.3 V (high state) or 0 V (low state). By installing wiringPi library, we can command the GPIO, as shown in Fig. 4. 11.

```
# Download and install wiringPi for controlling LED

sudo apt-get install git-core

# make sure that Pi has the latest version of Raspbian
sudo apt-get update
sudo apt-get upgrade

# To obtain WiringPi using GIT
git clone git://git.drogon.net/wiringPi

# Fetch an updated version if you have already used the clone operation
cd wiringPi
git pull origin

# Build/install wiringPi
cd wiringPi
./build
```

Fig. 4. 11 Installation of wiringPi for commanding GPIO pins on Raspberry Pi.

Supposed that we set 2 μA as the threshold current, if the current was greater than 2 μA, GPIO pin 18 outputs 3.3 V, turning on the red LED. The program code is shown in Fig. 4. 12. A resistor is in series with LED to limit the maximum current flow, preventing from burning the LED. The equivalent circuit model and real setup of LED when it turns on and off are shown in Fig. 4. 13.

```
# LED alarm signal
def ledalarm(current):
    GPIO.setup(18,GPIO.OUT)
    if(current > 2):
        GPIO.output(18, True)
    else:
        GPIO.output(18, False)
```

Fig. 4. 12 Program code of commanding LED.

(a) (b) (c)

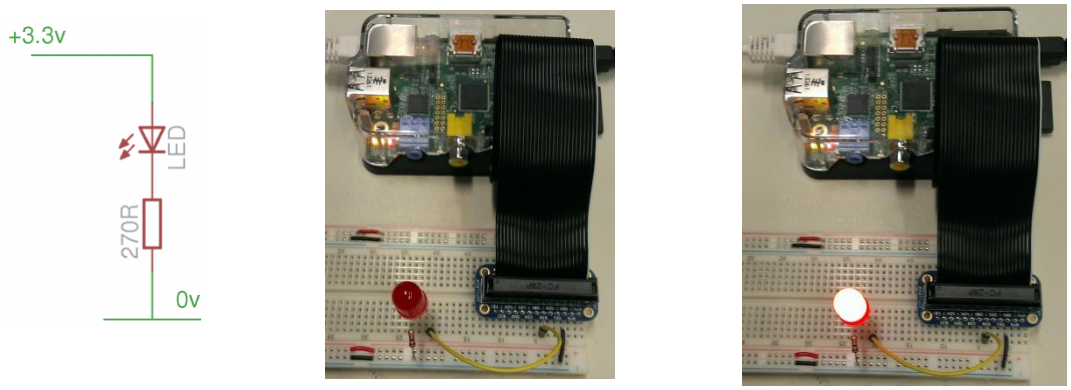

Fig. 4. 13 (a) Equivalent circuit of LED setup, (b) a red LED connected with GPIOs when it is in off state, (c) when it is in on state.

4.5 SMS Text Alert

Short message service (SMS) text can provide instant messages to mobile phones to indicate alarm situation or release of alarm situation. Twilio is a cloud communication

company that provides the service of sending and receiving text messages as well as making and receiving phone calls (see Fig. 4. 14). We developed a program to communicate with Twilio and used its function to send alert text messages through its web service APIs.

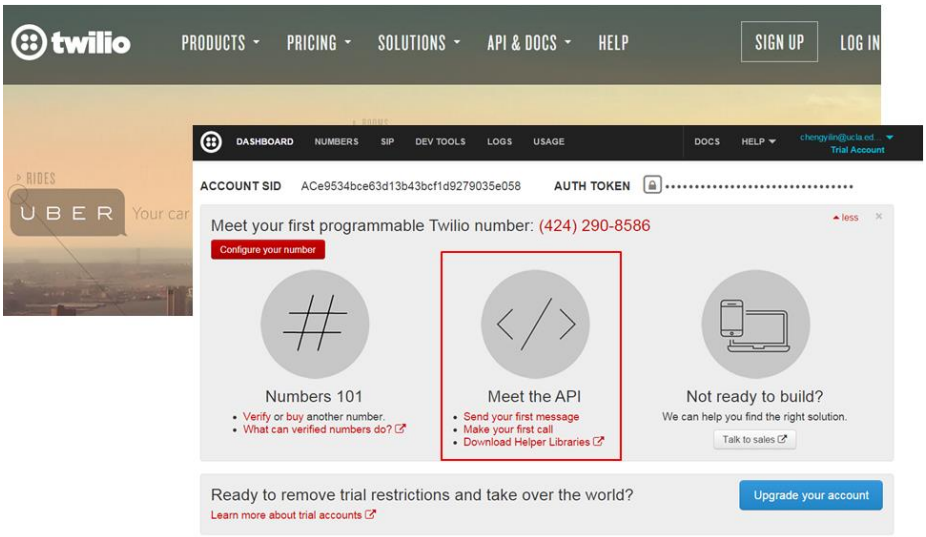

Fig. 4. 14 The website information of Twilio.

For test purpose, we adjusted a potentiometer to change the voltage across a 10 kΩ resistor, and thereby current varied and read by the system. A program was developed to monitor the electrical current of the sensor. If current went higher than 2 μA, the readout circuit would send an alert message to Twilio and then it delivered the alert message (Alarm: Gas concentration is too high) to our mobile phones, as shown in Fig. 4. 15. At the same time, a red LED was turned on to indicate the alarm situation. On the other hand, when the alarm situation was resolved, the user will receive a text message again showing “Alarm release” (Fig. 4. 15).

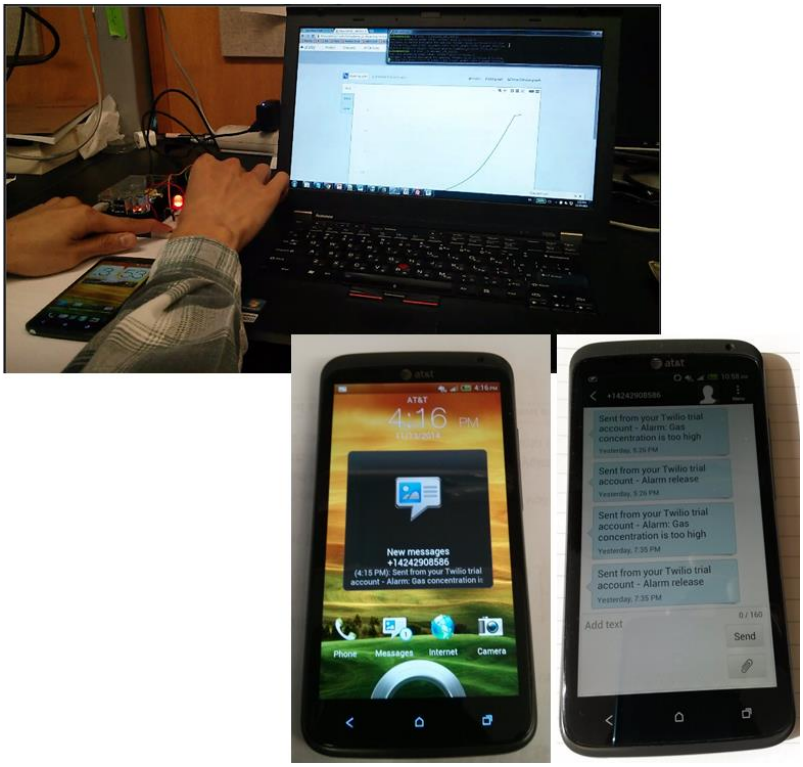

Fig. 4. 15 Alert text messages received on the mobile phone and the corresponding message content.

4.6 Two Channel System

It is desirable to have two channels to record two sensors signal at the same time because it is more efficient. For Angilent B2900A, it can provide up to two channels for measurements. In practice, the setup of our gas chamber can just support two channel measurements. The ceramic carrier was coated with two different nanocomposite films. The nanocomposite material was connected with tow electrodes on the ceramic carrier, where two electrodes were defined as voltage supply pin and ground pin, for each sensor. Ground pins of each sensor were tied together. Therefore, it took only three pins connected with the Agileent SMU through the BNC connector to achieve two-channel measurement. Given this demand, a two-channel system was developed herein.

In order to to provide two different bias conditions, a potentiometer and different resistors are in series as a voltage divider. Two jumper wires were connected to two different points of the voltage divider to provide two different biases. The potentiometer provides the flexibility of adjusting bias according to the values that users want. Two instrumentation amplifiers were utilized to amplify the current from two different sensors. All analog data were collected by the ADC. The ADC provides 8 channels which in principle can monitor currents of four sensors. The equivalent circuit model and the implementation of the circuits are shown in Fig. 4. 16.

(a) (b)

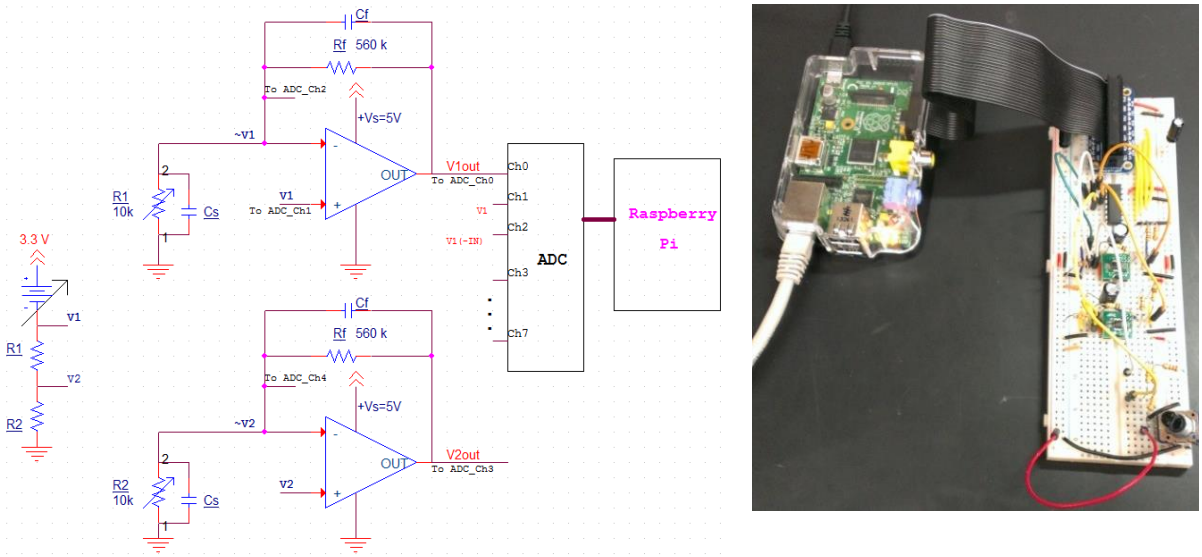

Fig. 4. 16 (a) Equivalent circuit model, (b) implementation of the circuit.

Assuming that ZnO/GRO is biased at 0.05 V and In_2O_3/GRO is biased at 1 V, the corresponding maximum currents are ~5 μA and ~3 μA, respectively (Fig. 4. 17).

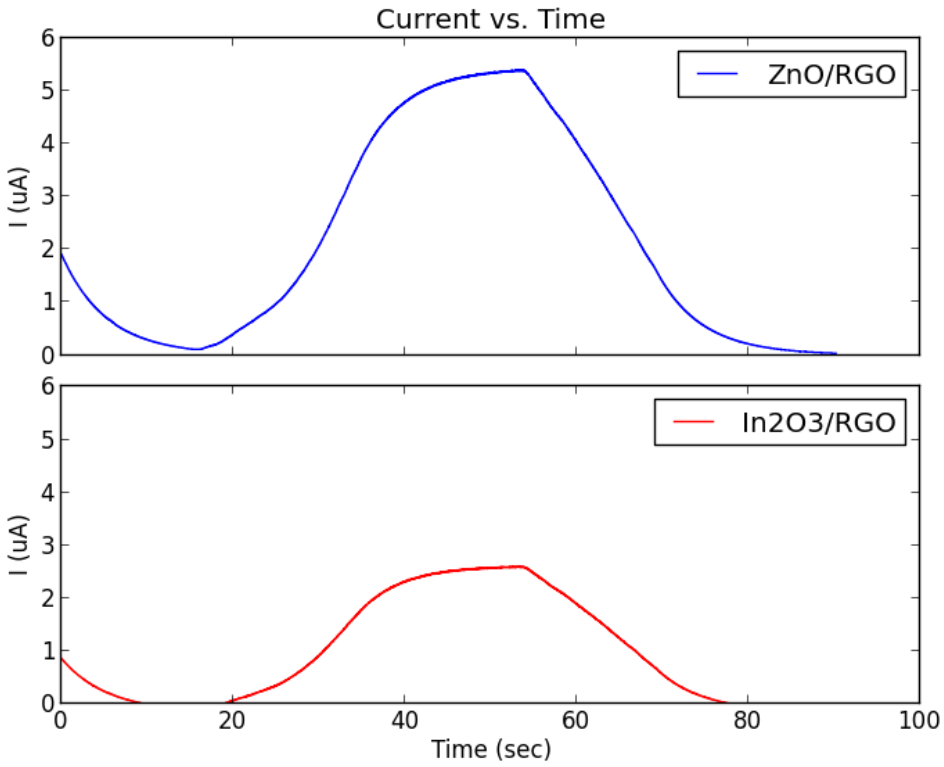

Fig. 4. 17 Test results of measurements using two channel systems.

4.7 Gas-Sensing Measurement Setup

Gas-sensing measurements were performed by using a computer-controlled custom-made gas-detection system as shown in Fig. 4. 18. The Raspberry Pi microcomputer was developed as the SMU and connected with gas sensors in the test chamber to record their currents, and thus monitoring gas concentration. The gas sensor was placed on a heating plate (Watlow CER-1-01-00003) which was connected to a DC power supply. A temperature sensor (type K thermocouple) was assembled in the heating plate, by which the temperature of the heating plate could be monitored. Air was used as carrier gas to flow through the test chamber. Mass flow controllers (MFCs, Omega FMA 5400/5500 series) were used to control

the flow rate of gases and to adjust the concentration of target gases (Oxygen in air), by which the target gases were proportionally mixed with the carrier gas. MFCs were connected to a computer, by which the flow rate of gases could be controlled and displayed in real time through LabVIEW program. The Raspberry Pi microcomputer was operated under the condition that bias voltage was 1 V and the sampling rate of acquiring data is 0.5 s. The photos of experimental setup and proposed readout circuit system using Raspberry Pi Model B+ are shown in Fig. 4. 19 and Fig. 4. 20, respectively.

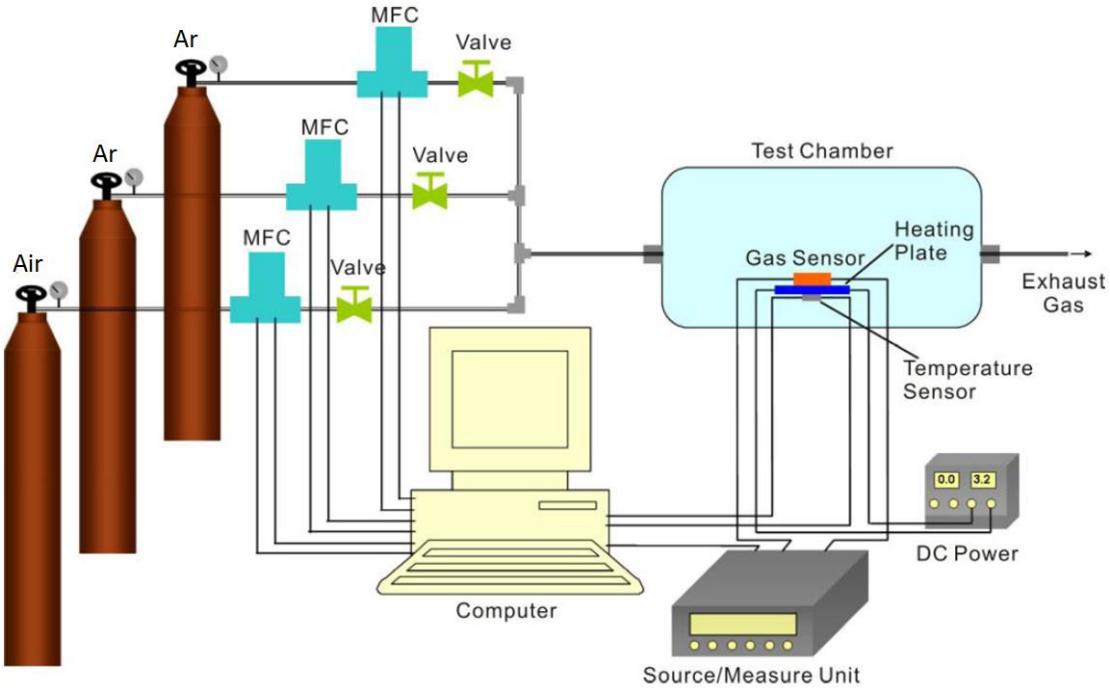

Fig. 4. 18 Schematic of experimental setup for gas sensing measurement.

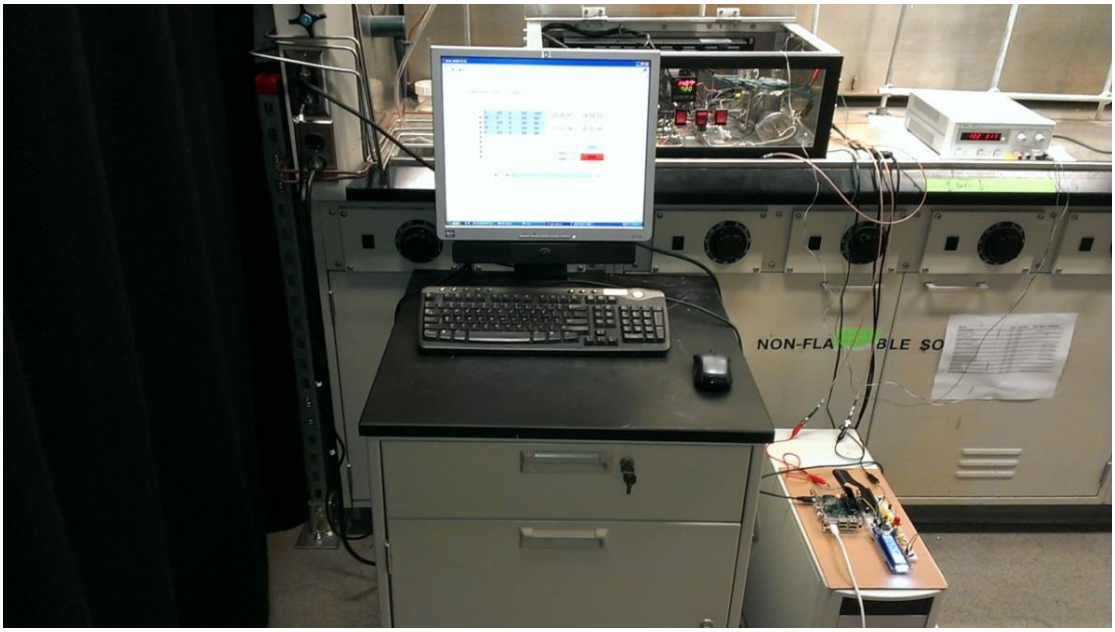

Fig. 4. 19 Photo of experimental setup.

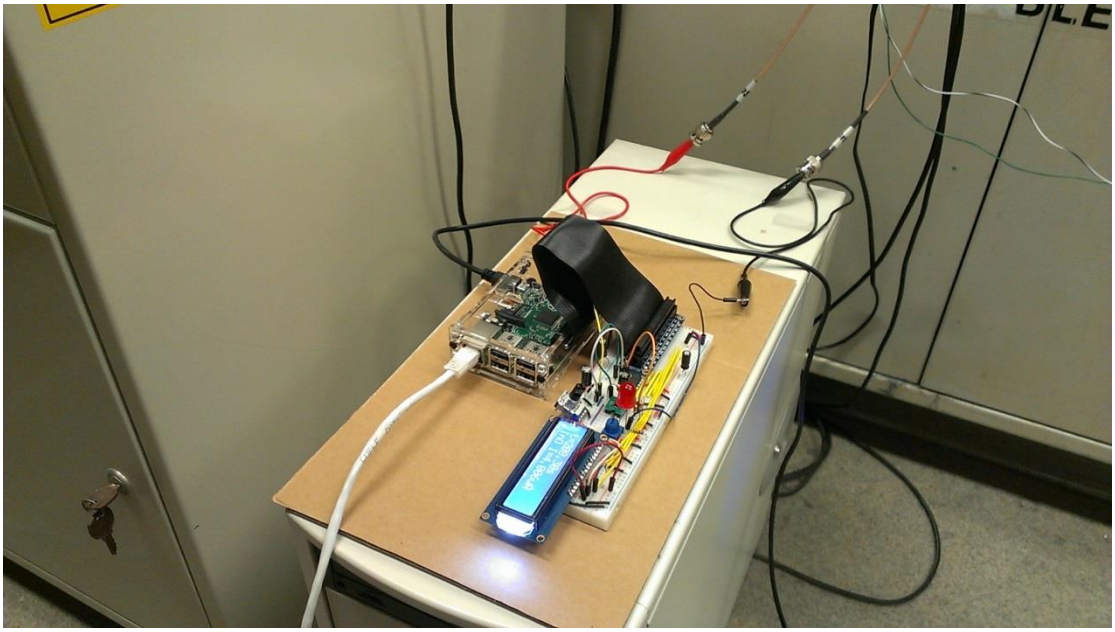

Fig. 4. 20 Photo of proposed readout circuit system.

Chapter V: Results and Discussion

In this chapter, the noise effects in electrical measurements are examined in the first part. Based on the optimized configuration, the performance of the proposed readout system is evaluated by responses of gas sensors to various oxygen concentration and different temperature, followed by a test of the transient response to characterize the dynamic behavior of the proposed circuit.

5.1 Noise Effect of Measurements in Gas Chamber

Most ICs suffer performance degradation due to noise, such as ripple or noise on the power supply pins. In order to minimize noise effects, “decoupling capacitors” are widely adopted to shunt noise through the capacitor. An alternative name is “bypass capacitor” as it is used to bypass the noise from power supply or other components. A good rule concerning bypass capacitors is to always include them in the circuit. If they are not included, the power supply noise may very well eliminate any chance for 10-bit ADC precision.

By-pass capacitors belong in two locations on the board: one at the power supply (10 μF to 100 μF) and one for the active device (digital and analog). The value of the device’s by-pass capacitor is dependent on the device under test. If the bandwidth of the device is less than or equal to ~1 MHz, a 1 μF will reduce injected noise dramatically. If the bandwidth of the device is above ~10 MHz, a 0.1 μF capacitor is probably appropriate. In between these

two frequencies, both or either one could be used. Therefore, a capacitor (C_f) across feedback resistor was included. Another capacitor (C_s) was in parallel with the gas sensor denoted as R_s. The equivalent circuit is shown in Fig. 5. 1.

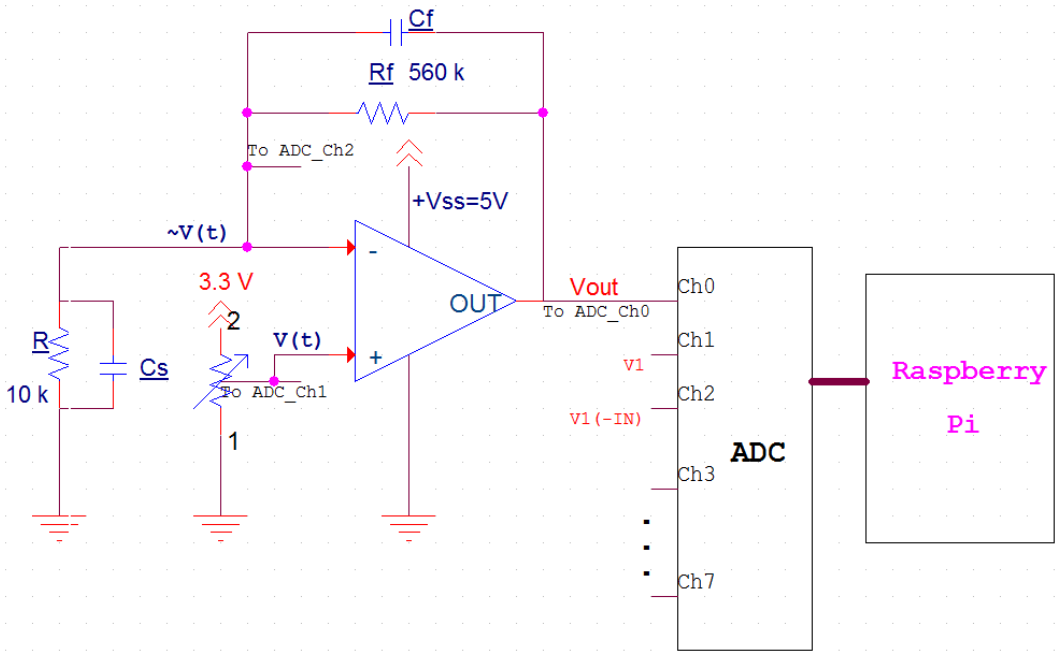

Fig. 5. 1 Equivalent circuit with include C_s and C_f.

To investigate noise effect, we placed a fixed resistor (R=10 kΩ) in the test chamber for preliminary study. By adjusting different supply voltages V(t) across the 10 kΩ resistor, we created current variation as a function of time. We used several common capacitor values (0.01, 0.1, 10 and 100 μF) to investigate noise reduction due to the bypass capacitors (C_f and C_s). These capacitors (C_f and C_s) with resistors (R_f and R) formed as low-pass filters. The time-resolved current response of different values of C_f and C_s are summarized in Fig. 5. 2.

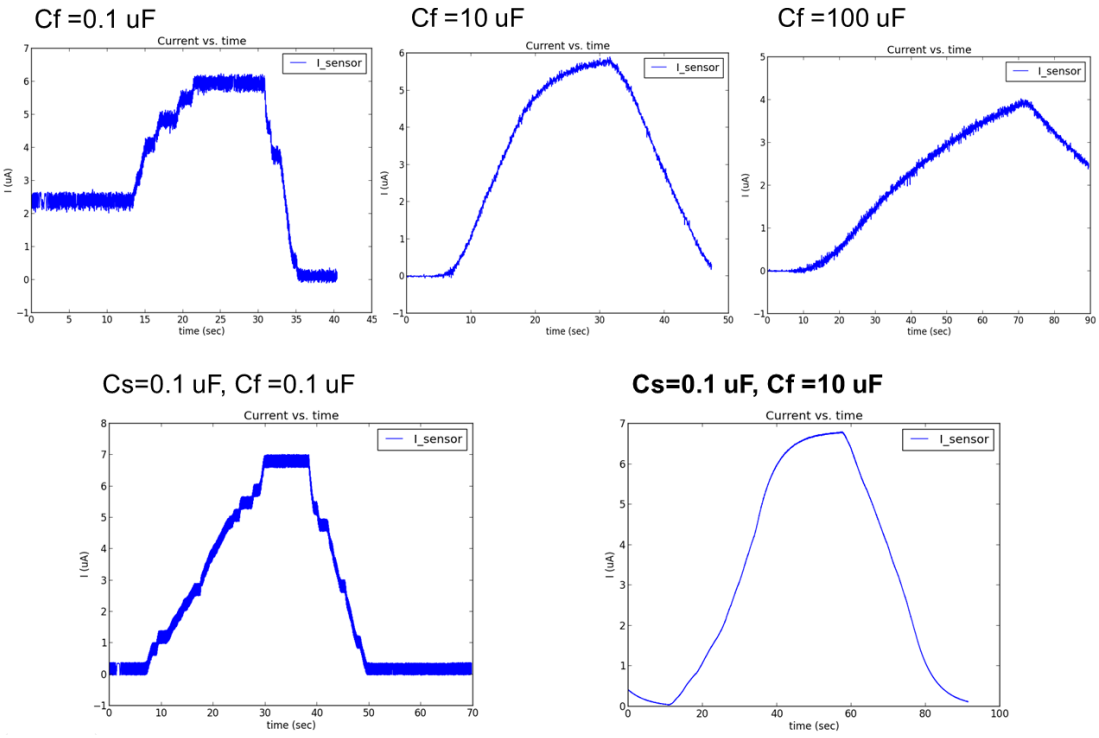

Fig. 5. 2 Noise effects in time-resolved current response due to different C_s and C_f.

In Fig. 5. 2, we observed that 0.1-μF capacitors cannot suppress noise effectively while 10-μF capacitor reduces noise substantially. The combination of C_s (0.1 μF) plus C_f (10 μF) had the best results, with clean current response curves. The test result of this configuration was plot on the designated IP (Fig. 5. 3).

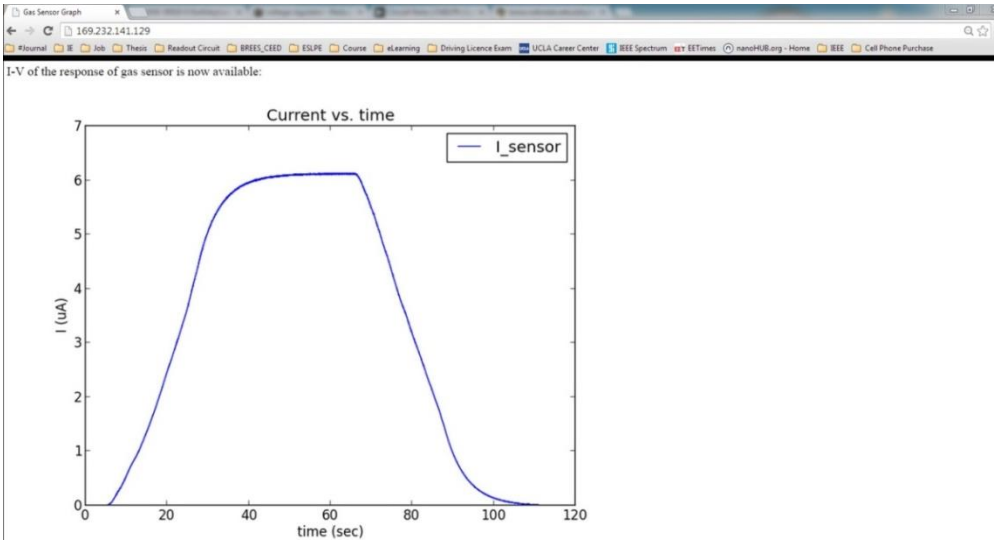

Fig. 5. 3 Test measurement data display on the webpage.

5.2 Response to Different Concentrations of Oxygen

The characterization of the In_2O_3/RGO gas sensors was conducted to evaluate the performance of developed readout circuit system. The current variation and the variation ratio ($\Delta I/I_o$) of the In_2O_3/RGO gas sensors toward different oxygen concentrations were measured by both an Agilent SMU and the Raspberry Pi microcomputer. Given the capability of the gas chamber system and the gas readily available, we designed different oxygen concentrations as summarized in Table 2. We assumed that air gas (V_air) containing 20% oxygen and argon (V_Ar) is a dilute gas carrying no oxygen.

Table 2 MFC settings for various oxygen concentrations

V_air (sccm)	V_Ar1 (sccm)	V_Ar2 (sccm)	V_Ar3 (sccm)	O_2 (ppm)
0	100	200	500	0
8	92	200	500	2000
16	84	200	500	4000
32	68	200	500	8000
64	36	200	500	16000
100	0	200	500	25000

Typically, gas sensors exhibit higher response at higher temperature due to higher reaction rates. However, temperature higher than 200 ℃ could cause decomposition or degradation of In_2O_3/RGO nanocomposites. According to these considerations and previous studies, we chose 140 ℃ as the operating temperature. Prior to the measurements, the sensor was heated to 140 ℃ under the argon environment for 2 hours (7200 s), thus having stable

baseline and more obvious response. After 2-hours of stabilization, argon gas kept flowing for 20 minutes (1200 s) to give a current value as a baseline, followed by a series of different oxygen concentrations with 15 minutes (900 s) response/recovery time. The change of current depends on oxygen concentration. Adsorbed oxygen on nanocomposite surface tends to become a negatively charged surface state, leading to a charge transfer from the surface conduction band to adsorbed oxygen. This charge transfer causes a decrease of surface conductivity of nanocomposite gas sensors. The current variation (ΔI) of an In_2O_3/RGO sensor upon exposure to various oxygen concentrations from 2,000 ppm to 32,000 ppm is shown in Fig. 5. 4. The results were measured separately by an Agilent SMU and Raspberry Pi and the applied voltage was 1 V which is supplied by each system. Since the current was measured separately by two different systems, the baselines were off by approximate 60 nA. We shifted the baselines of measured currents together for fair comparison (Fig. 5. 5). The Agilent SMU provided a clean current response, while the Raspberry Pi measured a current variation with serious noise. A rough estimation of variation ratios would be similar. Some spikes observed in the SMU curve were due to the valves closure/opening controlled by the MFCs.

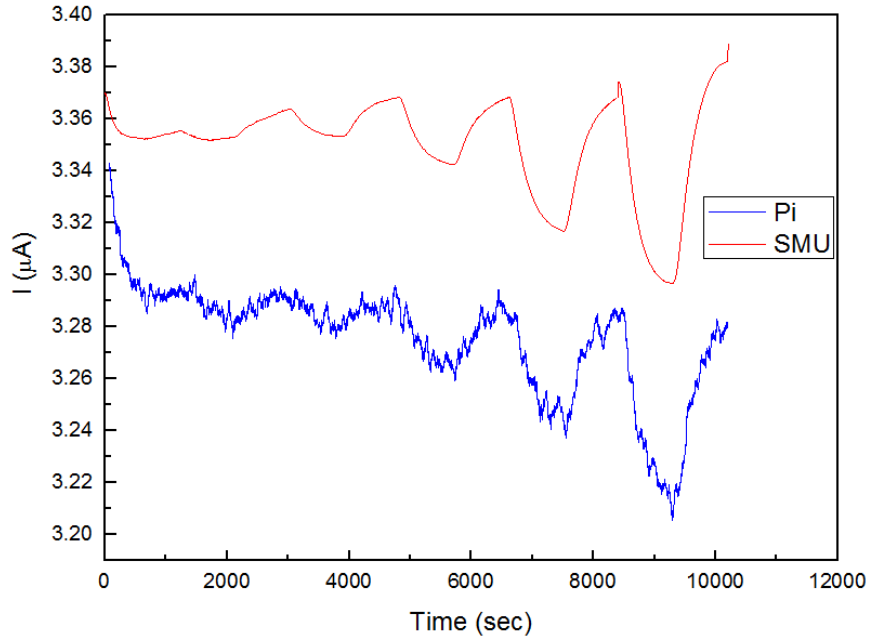

Fig. 5. 4 Preliminary results of response curves of In_2O_3/RGO measured by proposed system and SMU.

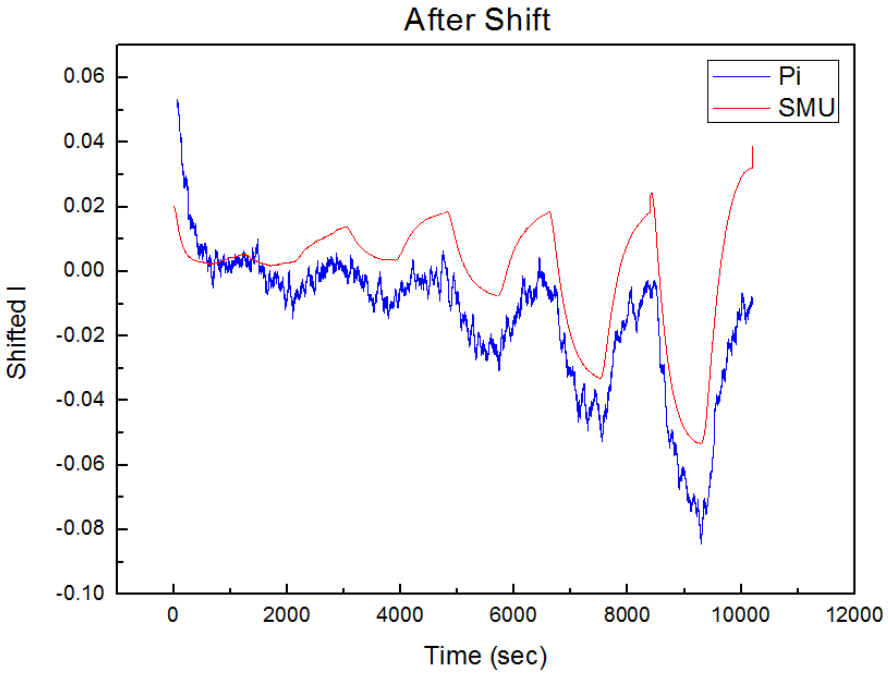

Fig. 5. 5 Response curves of In_2O_3/RGO after baselines are shifted together.

To define the current response in the noisy curve, we used Adjacent-Averaging (4179 points) smooth technique in Origin (ver. 8.5; OriginLab). The current variation ratio ($\Delta I/I_o$) is calculated with respect to the initial current (I_o) at t = 900 s, right before the oxygen exposure.

The calculated variation ratios upon exposure to different oxygen concentrations were summarized in Fig. 5. 6.

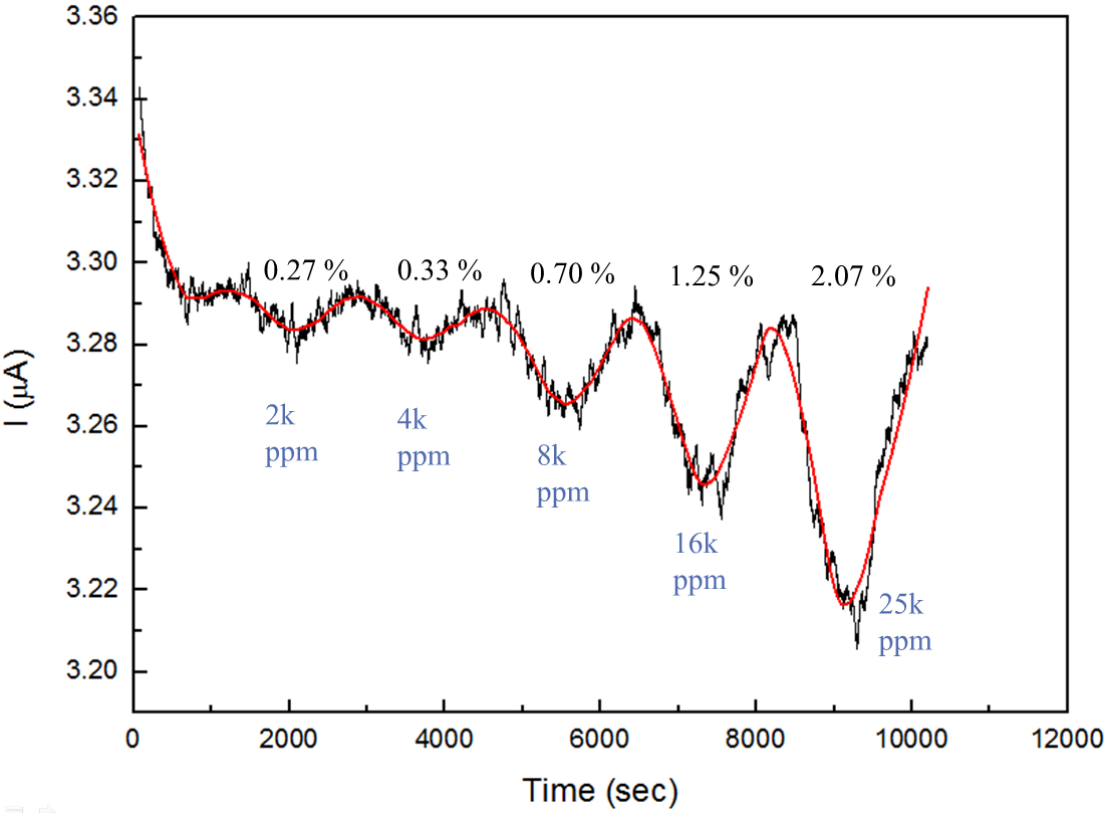

Fig. 5. 6 Use adjacent-averaging of 4179 points to depict the response curves.

The noisy response curve might be caused by improper filter design or the sampling rate of ADC and LCD. To improve the performance of proposed readout circuit system, various testing was performed. The optimal settings were when low-pass filters were constructed by C_s=C_f=10 μF and a sampling rate for the ADC of 0.5 s. Given this setting, new measurements of current response toward different oxygen concentrations (Table 3) was conducted. Two separate measurements recorded by the Agilent SMU and Raspberry Pi were performed continuously with the same conditions so that the results could be as close as possible.

Table 3 New MFC settings for different oxygen concentrations

V_air (sccm)	V_Ar1 (sccm)	V_Ar3 (sccm)	O_2 (ppm)
0	100	500	0
6	94	500	2000
12	88	500	4000
24	76	500	8000
48	52	500	16000
96	4	500	32000

The time-resolved current response of the In_2O_3/RGO gas sensors to different oxygen concentrations is presented in Fig. 5. 7. Compared with previous testing, the green response curve of the Raspberry Pi carries far less noise. For fair comparison, we shifted the baselines of two response curves together (Fig. 5. 8). Two curves, measured by the Agilent SMU and Raspberry Pi respectively, almost overlap, demonstrating that the capabilities of SMU and proposed readout circuit are comparable.

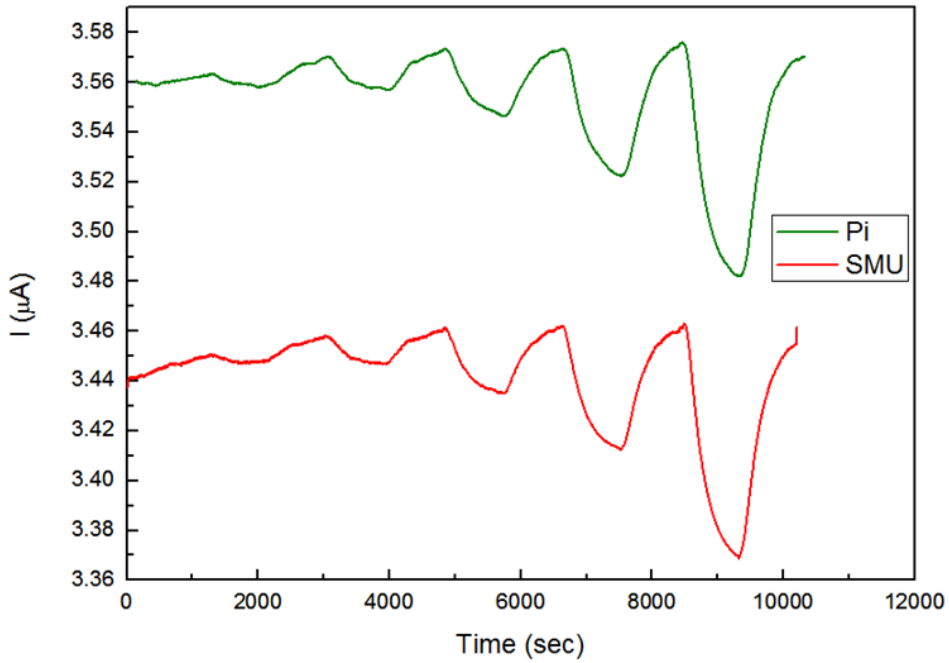

Fig. 5. 7 Response curves of the In_2O_3/RGO gas sensor to different concentrations of oxygen.

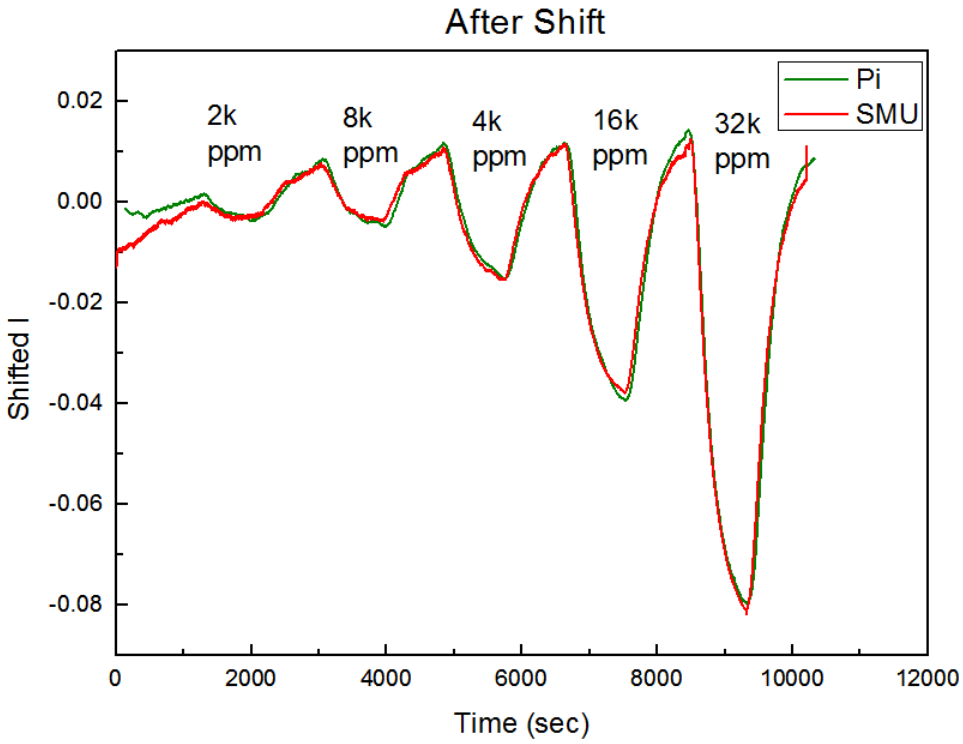

Fig. 5. 8 Response curves of (a) after the baselines are shifted together.

Significantly, the In_2O_3/RGO nanocomposite device showed prominent response upon introducing different concentrations of oxygen operating at 140 ℃. From 2,000 to 32,000 ppm, the response of the In_2O_3/RGO nanocomposite increased linearly as shown in the calibration curve (Fig. 5. 9).

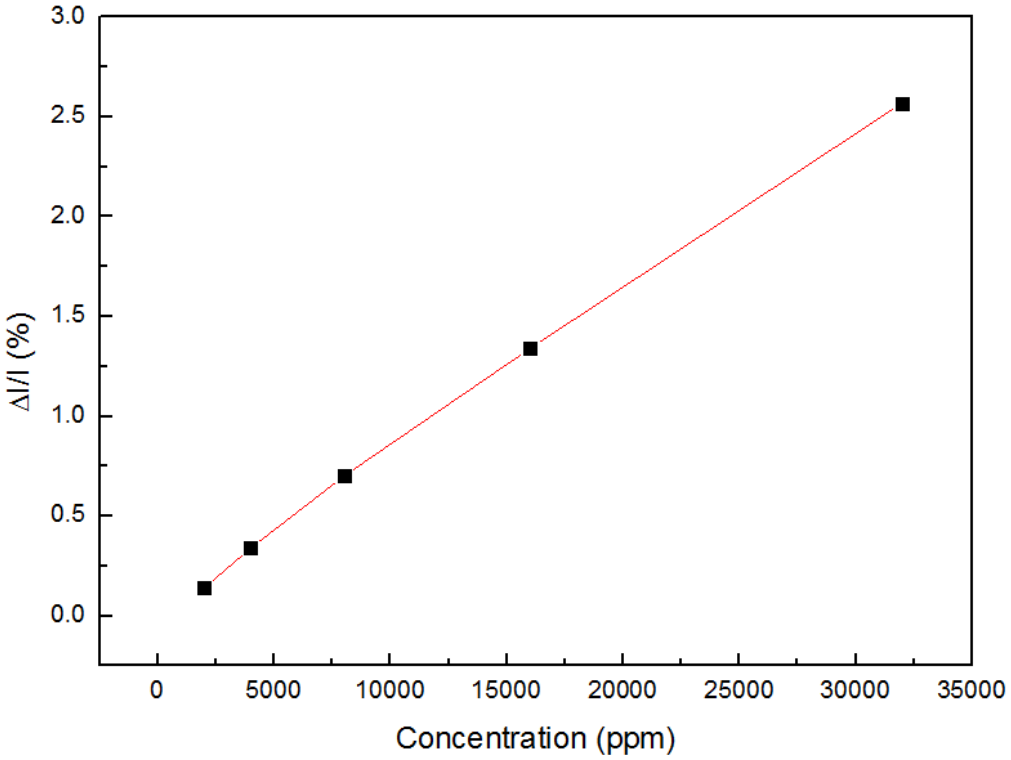

Fig. 5. 9 Plot of the response versus gas concentration obtained from Fig. 5. 8.

5.3 Response to Different Temperature

Detection performance of metal oxide based gas sensors is usually highly dependent on the operating temperatures. We have therefore further evaluated the response of the In_2O_3/RGO nanocomposite sensors to 32,000 ppm oxygen at different operating temperatures from 80 ℃ to 180 ℃ (Fig. 5. 10). Considering the chemical stability of RGO, we have kept the operating temperature below 200 ℃. The changes in the In_2O_3 film resistance has been attributed to the surface reactions of the target gases with the adsorbed oxygen species.[42] Depending on the working temperature, there are different oxygen species including molecular (O_2^-) and atomic (O^-, O^{2-}) ions on the surface. Generally, the molecular form dominates below 150 ℃, while the atomic species become increasingly important at higher

temperature. Since atomic oxygen species are more reactive, the response of In_2O_3/RGO nanocomposite device increases as temperature increases. Operation at high temperatures can often cause practical problems such as coalescence and structural changes of the sensor material which lead to sensor instability and response variation, as well as potential safety challenges.[43] Therefore, many studies work on developing sensing materials with low operating temperatures, which can avoid structural changes and reduce the power consumption/manufacturing cost.

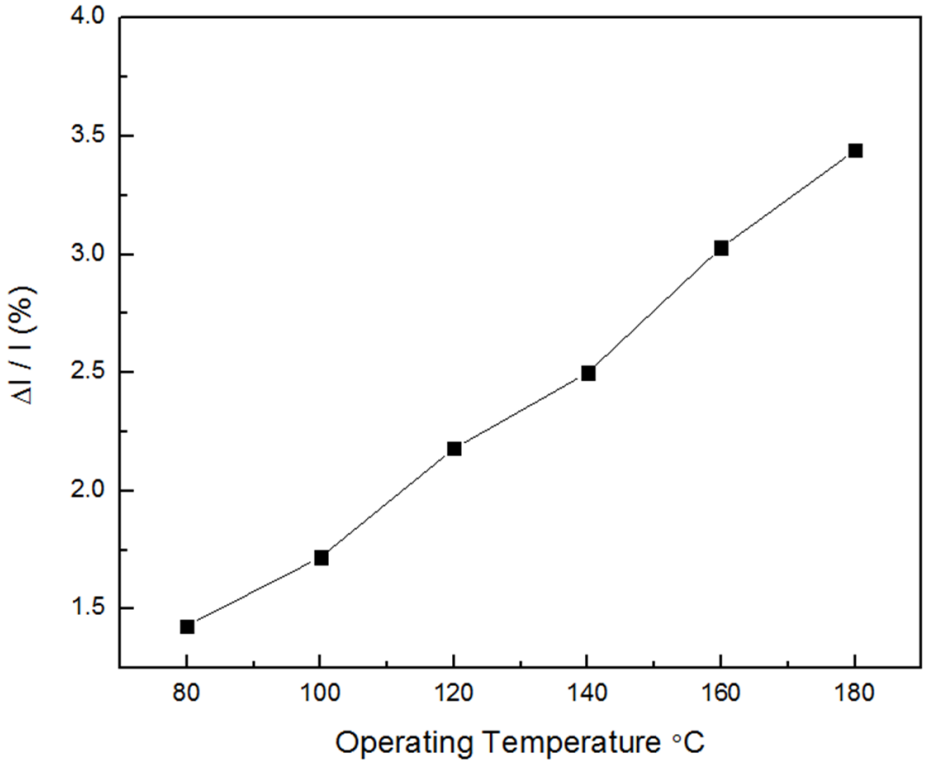

Fig. 5. 10 Current variation ratio of In_2O_3/RGO gas sensors versus operating temperature.

5.4 Head-to-Head Measurement with SMU

In order to test the capability of resolving the transient response of the Raspberry Pi, a head-to-head measurement was conducted by Agilent B2902A and Raspberry Pi, as shown in Fig. 5. 11. For fair comparison, all the biases were supplied by the Raspberry Pi. The conditions were Vs=1 V and temperature is 140 ^{o}C. For this measurement, Raspberry Pi has to be grounded with Keithley to obtain a stable readout value. Otherwise, the readout values displayed on SSH terminals would fluctuate substantially and read wrong values.

(a) (b)

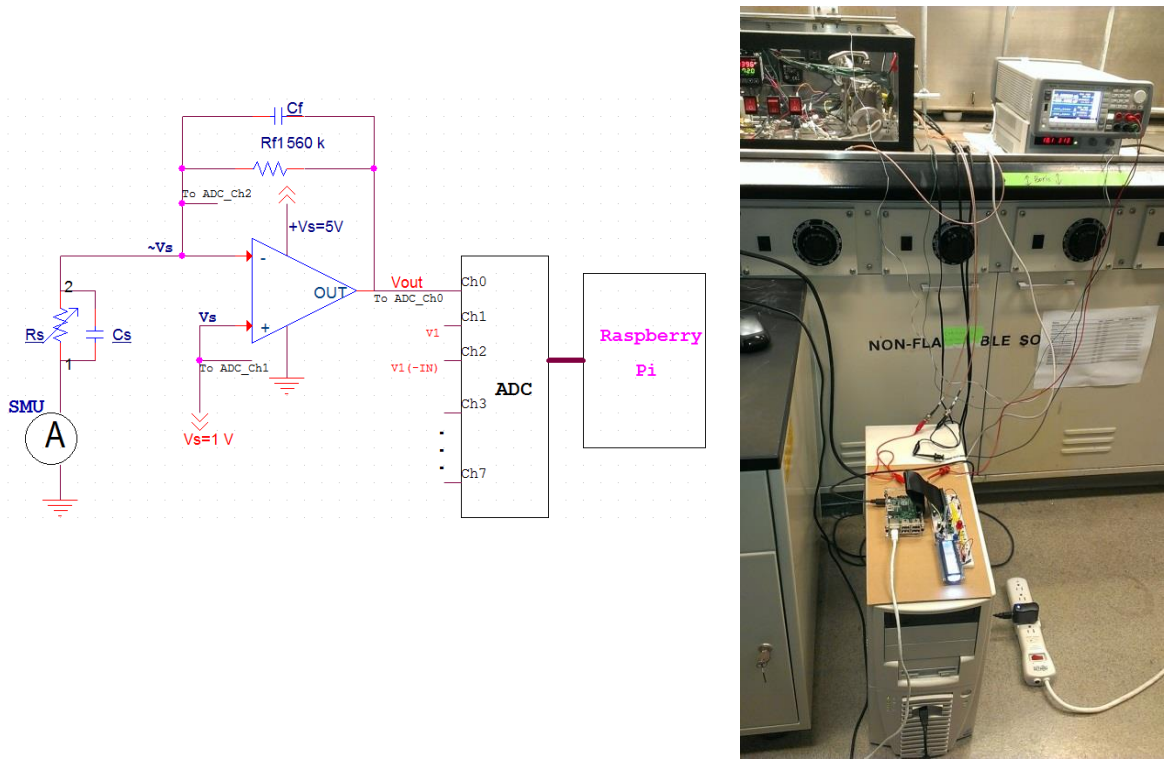

Fig. 5. 11 Direct head-to-head current measurement performed by Agilent and Raspberry Pi in series. (a) equivalent circuit model, (b) experimental setup.

The time-resolved current responses were shown in Fig. 5. 12. The red line was measured by Agilent SMU which was regarded as the standard for the measurement and the blue line was read by the Raspberry Pi. At ~8.7 sec, a supplied voltage (V_s) 1 V was applied and a sudden rise in current was observed. We observed that the dynamic response of the Raspberry Pi was not so sensitive to capture the whole transient response of the gas sensors, especially in the regime of the initial current change. The current deviation of average steady-state current was around around 0.78 % (Table 4). The time that the Pi spent to reach the average steady-state current was around 33.78 s, whereas SMU can demonstrate a sharp response when a voltage is applied. The delayed response of the Pi might be attributed to charging of decoupling capacitor, which can be characterized by a time constant (RC).

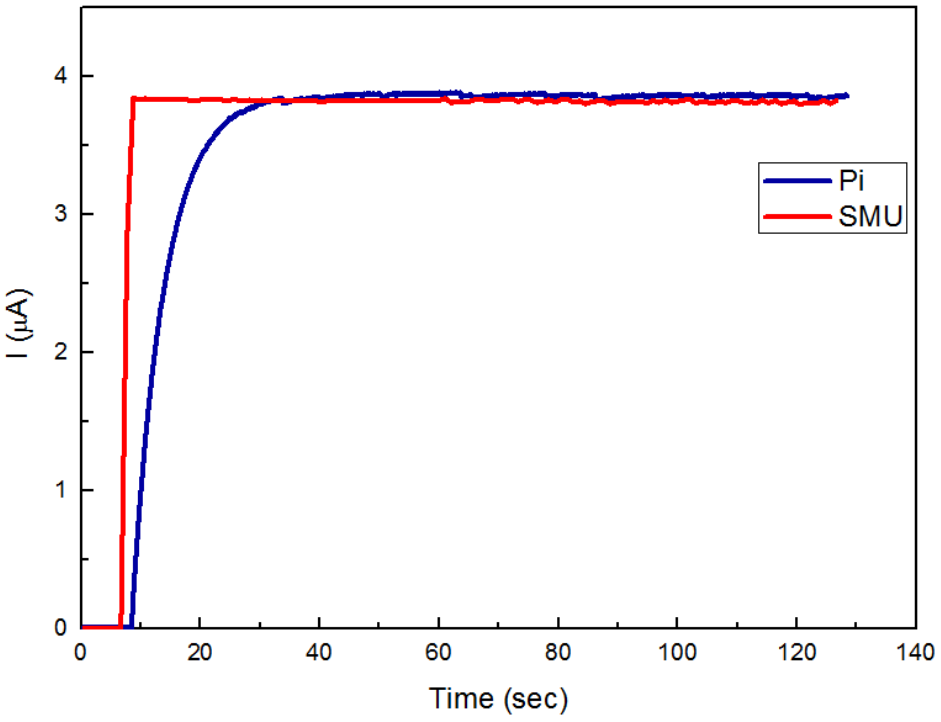

Fig. 5. 12 Time-resolved drain current response.

Table 4 Summary of current and time deviation

	Steady State Current (μA)	**Time (sec)**
Measured by SMU	3.83	~ at 9.73
Measured by Pi	3.86	~ at 33.78
Deviation	0.78 %	24.05

After the confirmation with Agilent SMU, the Raspberry Pi has been shown to possess the capability of measuring current at the same level and trend as Agilent, although there was some deviation. In other words, the proposed system can provide enough accuracy for target application—gas sensing.

Chapter VI: Conclusion

In this thesis, a new type of readout system using a Raspberry Pi microcomputer as the electronic control and data processing units were proposed and applied to the gas sensing of novel In_2O_3/RGO nanocomposites for the first time. The reasonable accuracy, low cost and flexible design was enabled by open-source software. The current responses of nanocomposite gas sensors toward oxygen were measured and compared between the proposed system and a commercialized Agilent SMU. The data measured by the proposed system is logged and plotted on a webpage in real-time, which are in a good agreement with the results measured by Agilent SMU. Users can easily check the current values from wherever there is Internet access. When gas sensors detect a threshold current value that corresponds to a gas concentration that is too high, the Raspberry Pi can trigger alarm functions to turn on a red LED and send out an alert text message.

The sensitivities of the In_2O_3/RGO gas sensors were examined by operation in various oxygen concentrations at 140 ^oC. Current variation ratio increases linearly as the O_2 concentration increases. For minimum concentration 2,000 ppm O_2 that can offer by current experimental setup, the proposed readout circuit still can resolve the appreciable variation ratio of 0.14 %. In addition, current variation ratios at different temperature were investigated. In_2O_3/RGO sensors demonstrate higher response at higher temperature due to the higher reaction rate of oxygen adsorption/desorption on the sensor surface. Finally, the transient

response of gas sensor at 140 ^{o}C toward 1 V applied bias was investigated. The proposed readout circuit exhibits a small amount of time delay to reach the same current value read by the Agilent SMU. Overall, the proposed system not only achieves satisfied performance in gas sensing applications, but also provides a smart alert system. We have successfully demonstrated the usefulness of a general purpose microcomputer (Raspberry Pi) in construction of low cost and open-source architecture analytic instruments.

Chapter VII: Future Work

The proposed readout circuitry was built on a breadboard for easy prototyping. To obtain better circuit performance and compact size, the readout circuit system could be constructed on a printed circuit board (PCB) instead. Potential leakage current, parasitic capacitance, shielding, and noise problems could be further reduced. Using design software to construct circuit layout for PCB can bring a lot more convenience for designers after prototyping. In addition, given the investigation of transient response, the speed of current measurement has not been optimized. The delay could be attributed to the RC time constant introduced by either decoupling or parasitic capacitance in the circuitry or data acquisition time limited by the efficiency of program execution. When it comes to characterizing the fast response of gas sensors in terms of response/recovery time, the speed of current proposed readout circuit could be further improved. More functionality of using Raspberry Pis to construct a smart readout system could be further explored, which cannot be offered by present SMU. For example, we can try to integrate a readout system with the MFCs control system. When receiving alert text messages, we can adjust the flow rates of MFCs instantly through text or SSH command lines, thus controlling gas concentration timely and remotely. The great design flexibility could be achieved by the abundant resource of free and open-source software.

Appendix 1

Main Control Program

```
#!/usr/bin/env python
import plotly.plotly as py
import json
import time
import readadc
import datetime
import os
import RPi.GPIO as GPIO
import sys
import array
import Adafruit_CharLCD
from subprocess import *

# Real-time communication with plotly website
with open('./config.json') as config_file:
    plotly_user_config = json.load(config_file)

py.sign_in(plotly_user_config["plotly_username"], plotly_user_config["plotly_api_key"])

url = py.plot([
    {
        'x': [], 'y': [], 'type': 'scatter',
        'stream': {
            'token': plotly_user_config['plotly_streaming_tokens'][0],
            'maxpoints': 10000
        }
    }], filename='Raspberry Pi Streaming Values')

print "View your streaming graph here: ", url

# LED alarm signal
def ledalarm(current, time):
    GPIO.setup(18,GPIO.OUT)
    if(3.7 < current < 3.85 and time > 300):
```

```
            GPIO.output(18, True)
        else:
            GPIO.output(18, False)

# SMS text alarm message
def SMStext(incurrent, indone, time):
    from twilio.rest import TwilioRestClient
    # Find these values at https://twilio.com/user/account
    account_sid = "ACe9534bce63d13b43bcf1d9279035e058"
    auth_token = "9d4713cac479be1fe902f4cdb69138d9"
    client = TwilioRestClient(account_sid, auth_token)
    if(3.7 < incurrent < 3.85 and indone == 0 and time > 300):
            message = client.messages.create(to="+13105929865", from_="+14242908586",
                                             body="Alarm: Gas concentration is too high")
            indone = 1
    elif(incurrent > 3.85 and indone ==1 and time > 300):
            message = client.messages.create(to="+13105929865", from_="+14242908586",
                                             body="Alarm release")
            indone = 0
    return indone


GPIO.setmode(GPIO.BCM)   #use BCM(Broadcom) pin layout
wait = 0.25
n = int(200)      # number of points to keep in moving average average

# temperature sensor middle pin connected channel 0 of mcp3008
Vout_adc = 0              # set the channel_0 to read
Vd_adc = 1                # set the channel_1 Vd to read
VIN_adc= 2                 # set the channel_2 V(-IN) to read
readadc.initialize()

stream = py.Stream(plotly_user_config['plotly_streaming_tokens'][0])
stream.open()


ADCBITS = 10              # number of bits in ADC
ADCRES = float(pow(2,ADCBITS)-1)      # calculated ADC resolution
```

```
ADCREF = float(5110)          # ADC reference value (5110 mV (5V) in this case, change as needed)
VRES = ADCREF/ADCRES          # Voltage resolution
Rf = float(560000)

start_time = time.time()
data = [[],[],[],[],[],[]] # initialize 6D matrix, 0th for time, 1st for Vout, 2nd for Vd bias, 3th for V(-IN), 4th for I, 5th for I_ave

# The main sensor reading and plotting loop
count=0
done = 0
try:
    while True:
        count += 1
        # read the analog pin
        Vout_value = readadc.readadc(Vout_adc, readadc.PINS.SPICLK, readadc.PINS.SPIMOSI, readadc.PINS.SPIMISO, readadc.PINS.SPICS)
        Vd_value = readadc.readadc(Vd_adc, readadc.PINS.SPICLK, readadc.PINS.SPIMOSI, readadc.PINS.SPIMISO, readadc.PINS.SPICS)
        VIN_value = readadc.readadc(VIN_adc, readadc.PINS.SPICLK, readadc.PINS.SPIMOSI, readadc.PINS.SPIMISO, readadc.PINS.SPICS)
        elapsed_time = time.time() - start_time

        Vout = float(Vout_value) * VRES/1000   # scale the output from mV to V
        Vd = float(Vd_value) * VRES/1000
        VIN = float(VIN_value) * VRES/1000
        I = float((Vout - VIN)/Rf)*1e6   # scale I (uA)

        data[0].append(elapsed_time)
        data[1].append(Vout)
        data[2].append(Vd)
        data[3].append(VIN)
        data[4].append(I)
        Voutdata = data[1][-n:]
        Vout_ave = sum(Voutdata)/n
        Vout_stdev = (sum([(x - Vout_ave) ** 2   for x in Voutdata])/n) ** 0.5
        Vddata = data[2][-n:]
```

```
            Vd_ave = sum(Vddata)/n
            Vd_stdev = (sum([(x - Vd_ave) ** 2   for x in Vddata])/n) ** 0.5
            VINdata = data[3][-n:]
            VIN_ave = sum(VINdata)/n
            Idata = data[4][-n:]
            I_ave = sum(Idata)/n
            data[5].append(I_ave)
            I_stdev = (sum([(x - I_ave) ** 2   for x in Idata])/n) ** 0.5

            stream.write({'x': elapsed_time, 'y': I_ave})     # write the data to plotly
            sys.stdout.write('\r'+' '*100+'\r')       #Expression setting on the screen
            sys.stdout.write("t=%0.2fs Vout_ave=%.3fV; Vs_ave=%.3fV; V(-IN)_ave=%.3fV;
I_ave=%.3fuA\r" % (elapsed_time,Vout_ave,Vd_ave,VIN_ave,I_ave) )
            sys.stdout.flush()
            time.sleep(wait)

            ledalarm(I_ave, elapsed_time)
            done = SMStext(I_ave, done, elapsed_time)

            lcd = Adafruit_CharLCD.Adafruit_CharLCD()
            lcd.begin(16, 1)
            lcd.message("t=%0.2fs\n" % (elapsed_time))
            lcd.message('InO I=%.3fuA' % (I_ave))
            time.sleep(wait)      # Keep the message on the LCD screen

except KeyboardInterrupt:
    sys.stdout.write('\n')
    # .csv write module
    import csv
    import datetime
    date_string= datetime.datetime.now().strftime("%Y-%m-%d-%H:%M")
    filename = 'I_sensor_' + date_string + '.csv'
    print('Keyboard exception caught! Writing data to %s' % (filename))
    fl = open(filename, 'w')
    writer = csv.writer(fl)
    writer.writerow(['time (s)', 'Vout (V)', 'Vd (V)', 'V(-IN) (V)', 'I (uA)', 'I_ave (uA)']) #if
needed
    ar=zip(data[0],data[1],data[2],data[3],data[4],data[5])
```

```
    for values in ar:
        writer.writerow(values)
    fl.close()
    GPIO.cleanup() # Reset all pins of GPIO ports to avoid runtime warring

# Plot and save as png image on www
import matplotlib
matplotlib.use('Agg')

import numpy as np
import matplotlib.pyplot as plt

datapath = '/home/pi/'+filename
data = np.genfromtxt(datapath, delimiter=',', skip_header=60*4, skip_footer=0, names=['x',
'y'], usecols=(0,-1))
np.seterr(all='ignore') # Set how floating-point errors are handled in numpy

fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.set_title("Current vs. time")
ax1.set_xlabel('time (sec)')
ax1.set_ylabel('I (uA)')

ax1.plot(data['x'], data['y'], color='b', label='I_sensor')
leg = ax1.legend()

plotfilename = '/var/www/cgi-bin/I_Sen_LED_SMS.png'

plt.savefig(plotfilename)
```

Appendix 2

Program code of "readadc.py"

```
import RPi.GPIO as GPIO
# change these as desired - they're the pins connected from the
# SPI port on the ADC to the GPIO Pins on the Raspi

# MCP3008 to Raspi (PiCobbler) Pin connections
class PINS:
    SPICLK = 12
    SPIMISO = 16
    SPIMOSI = 20
    SPICS = 21

 # set up the SPI interface pins
def initialize():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(PINS.SPIMOSI, GPIO.OUT)
    GPIO.setup(PINS.SPIMISO, GPIO.IN)
    GPIO.setup(PINS.SPICLK, GPIO.OUT)
    GPIO.setup(PINS.SPICS, GPIO.OUT)

# Function to read data from Analog Pin 0 from MCP3008 (don't need to edit)
# This function will be called in our loop to get the current sensor value
def readadc(adcnum, clockpin, mosipin, misopin, cspin):
    if ((adcnum > 7) or (adcnum < 0)):
            return -1
    GPIO.output(cspin, True)

    GPIO.output(clockpin, False)  # start clock low
    GPIO.output(cspin, False)     # bring CS low

    commandout = adcnum
    commandout |= 0x18   # start bit + single-ended bit
    commandout <<= 3     # we only need to send 5 bits here
    for i in range(5):
```

```
        if (commandout & 0x80):
                GPIO.output(mosipin, True)
        else:
                GPIO.output(mosipin, False)
        commandout <<= 1
        GPIO.output(clockpin, True)
        GPIO.output(clockpin, False)

adcout = 0
# read in one empty bit, one null bit and 10 ADC bits
for i in range(12):
        GPIO.output(clockpin, True)
        GPIO.output(clockpin, False)
        adcout <<= 1
        if (GPIO.input(misopin)):
                adcout |= 0x1

GPIO.output(cspin, True)

adcout /= 2           # first bit is 'null' so drop it
return adcout
```

Appendix 3

Program code of "Adafruit_CharLCD.py"

```
#!/usr/bin/python
from time import sleep

class Adafruit_CharLCD(object):
    # commands
    LCD_CLEARDISPLAY          = 0x01
    LCD_RETURNHOME            = 0x02
    LCD_ENTRYMODESET          = 0x04
    LCD_DISPLAYCONTROL        = 0x08
    LCD_CURSORSHIFT           = 0x10
    LCD_FUNCTIONSET           = 0x20
    LCD_SETCGRAMADDR          = 0x40
    LCD_SETDDRAMADDR          = 0x80

    # flags for display entry mode
    LCD_ENTRYRIGHT            = 0x00
    LCD_ENTRYLEFT             = 0x02
    LCD_ENTRYSHIFTINCREMENT = 0x01
    LCD_ENTRYSHIFTDECREMENT = 0x00

    # flags for display on/off control
    LCD_DISPLAYON             = 0x04
    LCD_DISPLAYOFF            = 0x00
    LCD_CURSORON              = 0x02
    LCD_CURSOROFF             = 0x00
    LCD_BLINKON               = 0x01
    LCD_BLINKOFF              = 0x00

    # flags for display/cursor shift
    LCD_DISPLAYMOVE           = 0x08
    LCD_CURSORMOVE            = 0x00

    # flags for display/cursor shift
    LCD_DISPLAYMOVE           = 0x08
```

```
    LCD_CURSORMOVE      = 0x00
    LCD_MOVERIGHT       = 0x04
    LCD_MOVELEFT        = 0x00

    # flags for function set
    LCD_8BITMODE        = 0x10
    LCD_4BITMODE        = 0x00
    LCD_2LINE           = 0x08
    LCD_1LINE           = 0x00
    LCD_5x10DOTS        = 0x04
    LCD_5x8DOTS         = 0x00

    def __init__(self, pin_rs=25, pin_e=24, pins_db=[23, 17, 27, 22], GPIO=None):
        # Emulate the old behavior of using RPi.GPIO if we haven't been given
        # an explicit GPIO interface to use
        if not GPIO:
            import RPi.GPIO as GPIO
            GPIO.setwarnings(False)
        self.GPIO = GPIO
        self.pin_rs = pin_rs
        self.pin_e = pin_e
        self.pins_db = pins_db

        self.GPIO.setmode(GPIO.BCM)
        self.GPIO.setup(self.pin_e, GPIO.OUT)
        self.GPIO.setup(self.pin_rs, GPIO.OUT)

        for pin in self.pins_db:
            self.GPIO.setup(pin, GPIO.OUT)

        self.write4bits(0x33)   # initialization
        self.write4bits(0x32)   # initialization
        self.write4bits(0x28)   # 2 line 5x7 matrix
        self.write4bits(0x0C)   # turn cursor off 0x0E to enable cursor
        self.write4bits(0x06)   # shift cursor right

        self.displaycontrol = self.LCD_DISPLAYON | self.LCD_CURSOROFF |
self.LCD_BLINKOFF
```

```
        self.displayfunction = self.LCD_4BITMODE | self.LCD_1LINE |
self.LCD_5x8DOTS
        self.displayfunction |= self.LCD_2LINE

        # Initialize to default text direction (for romance languages)
        self.displaymode = self.LCD_ENTRYLEFT |
self.LCD_ENTRYSHIFTDECREMENT
        self.write4bits(self.LCD_ENTRYMODESET | self.displaymode)   # set entry mode

        self.clear()

    def begin(self, cols, lines):
        if (lines > 1):
            self.numlines = lines
            self.displayfunction |= self.LCD_2LINE

    def home(self):
        self.write4bits(self.LCD_RETURNHOME)   # set cursor position to zero
        self.delayMicroseconds(3000)   # this command takes a long time!

    def clear(self):
        self.write4bits(self.LCD_CLEARDISPLAY)   # command to clear display
        self.delayMicroseconds(3000)   # 3000 microsecond sleep, clearing the display takes a
long time

    def setCursor(self, col, row):
        self.row_offsets = [0x00, 0x40, 0x14, 0x54]
        if row > self.numlines:
            row = self.numlines - 1   # we count rows starting w/0
        self.write4bits(self.LCD_SETDDRAMADDR | (col + self.row_offsets[row]))

    def noDisplay(self):
        """ Turn the display off (quickly) """
        self.displaycontrol &= ~self.LCD_DISPLAYON
        self.write4bits(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def display(self):
```

```
        """ Turn the display on (quickly) """
        self.displaycontrol |= self.LCD_DISPLAYON
        self.write4bits(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def noCursor(self):
        """ Turns the underline cursor off """
        self.displaycontrol &= ~self.LCD_CURSORON
        self.write4bits(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def cursor(self):
        """ Turns the underline cursor on """
        self.displaycontrol |= self.LCD_CURSORON
        self.write4bits(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def noBlink(self):
        """ Turn the blinking cursor off """
        self.displaycontrol &= ~self.LCD_BLINKON
        self.write4bits(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def blink(self):
        """ Turn the blinking cursor on """
        self.displaycontrol |= self.LCD_BLINKON
        self.write4bits(self.LCD_DISPLAYCONTROL | self.displaycontrol)

    def DisplayLeft(self):
        """ These commands scroll the display without changing the RAM """
        self.write4bits(self.LCD_CURSORSHIFT | self.LCD_DISPLAYMOVE |
self.LCD_MOVELEFT)

    def scrollDisplayRight(self):
        """ These commands scroll the display without changing the RAM """
        self.write4bits(self.LCD_CURSORSHIFT | self.LCD_DISPLAYMOVE |
self.LCD_MOVERIGHT)

    def leftToRight(self):
        """ This is for text that flows Left to Right """
        self.displaymode |= self.LCD_ENTRYLEFT
        self.write4bits(self.LCD_ENTRYMODESET | self.displaymode)
```

```
    def rightToLeft(self):
        """ This is for text that flows Right to Left """
        self.displaymode &= ~self.LCD_ENTRYLEFT
        self.write4bits(self.LCD_ENTRYMODESET | self.displaymode)

    def autoscroll(self):
        """ This will 'right justify' text from the cursor """
        self.displaymode |= self.LCD_ENTRYSHIFTINCREMENT
        self.write4bits(self.LCD_ENTRYMODESET | self.displaymode)

    def noAutoscroll(self):
        """ This will 'left justify' text from the cursor """
        self.displaymode &= ~self.LCD_ENTRYSHIFTINCREMENT
        self.write4bits(self.LCD_ENTRYMODESET | self.displaymode)

    def write4bits(self, bits, char_mode=False):
        """ Send command to LCD """
        self.delayMicroseconds(1000)   # 1000 microsecond sleep
        bits = bin(bits)[2:].zfill(8)
        self.GPIO.output(self.pin_rs, char_mode)
        for pin in self.pins_db:
            self.GPIO.output(pin, False)
        for i in range(4):
            if bits[i] == "1":
                self.GPIO.output(self.pins_db[::-1][i], True)
        self.pulseEnable()
        for pin in self.pins_db:
            self.GPIO.output(pin, False)
        for i in range(4, 8):
            if bits[i] == "1":
                self.GPIO.output(self.pins_db[::-1][i-4], True)
        self.pulseEnable()

    def delayMicroseconds(self, microseconds):
        seconds = microseconds / float(1000000)   #divide microseconds by 1 million for
seconds
        sleep(seconds)
```

```
    def pulseEnable(self):
        self.GPIO.output(self.pin_e, False)
        self.delayMicroseconds(1)   # 1 microsecond pause - enable pulse must be > 450ns
        self.GPIO.output(self.pin_e, True)
        self.delayMicroseconds(1)   # 1 microsecond pause - enable pulse must be > 450ns
        self.GPIO.output(self.pin_e, False)
        self.delayMicroseconds(1)   # commands need > 37us to settle

    def message(self, text):
        """ Send string to LCD. Newline wraps to second line"""
        for char in text:
            if char == '\n':
                self.write4bits(0xC0)   # next line
            else:
                self.write4bits(ord(char), True)


if __name__ == '__main__':
    lcd = Adafruit_CharLCD()
    lcd.clear()
    lcd.message("ZnO/RGO I= uA\nInO/RGO I= uA")
```

References

1. Pearce, J. M. *Science* **2012,** 337, (6100), 1303-1304.
2. Pearce, J. M. *Nature* **2014,** 505, (7485), 618-618.
3. Urban, P. L. *Journal of Chemical Education* **2014,** 91, (5), 751-752.
4. Arduino website. http://www.arduino.cc/ (3 December, 2014),
5. Chiu, S.-H.; Urban, P. L. *Biosensors and Bioelectronics* **2015,** 64, (0), 260-268.
6. Raspberry Pi website. http://www.raspberrypi.org/ (3 March, 2014),
7. Upton, E. *Computer* **2013,** 46, (10), 14 - 16.
8. Geyer, M. J. *Journal of Chemical Education* **2014,** 91, (11), 2005-2006.
9. Leccese, F.; Cagnetti, M.; Calogero, A.; Trinca, D.; Pasquale, S.; Giarnetti, S.; Cozzella, L. *Sensors* **2014,** 14, (5), 9290-9312.
10. Poljak, J.; Botella, G.; García, C.; Poljaček, S.; Matías, M.; Tirado, F. *Sensors* **2013,** 13, (11), 14277-14300.
11. Kim, H.; Kim, S.-W.; Park, Y.-U.; Gwon, H.; Seo, D.-H.; Kim, Y.; Kang, K. *Nano Res.* **2010,** 3, (11), 813-821.
12. Leng, K.; Zhang, F.; Zhang, L.; Zhang, T.; Wu, Y.; Lu, Y.; Huang, Y.; Chen, Y. *Nano Res.* **2013,** 6, (8), 581-592.
13. Zampolli, S.; Elmi, I.; Ahmed, F.; Passini, M.; Cardinali, G. C.; Nicoletti, S.; Dori, L. *Sensors and Actuators B: Chemical* **2004,** 101, (1–2), 39-46.

14. Meng, F.-L.; Jia, Y.; Liu, J.-Y.; Li, M.-Q.; Sun, Y.-F.; Liu, J.-H.; Huang, X.-J. *Analytical Methods* **2010,** 2, (11), 1710-1714.

15. Biaggi-Labiosa, A.; Solá, F.; Lebrón-Colón, M.; Evans, L.; Xu, J.; Hunter, G.; Berger, G.; González, J. *Nanotechnology* **2012,** 23, (45), 455501.

16. Yu, J.; Wen, H.; Shafiei, M.; Field, M. R.; Liu, Z. F.; Wlodarski, W.; Motta, N.; Li, Y. X.; Kalantar-zadeh, K.; Lai, P. T. *Sensors and Actuators B: Chemical* **2013,** 184, (0), 118-129.

17. Oliaee, S. N.; Khodadadi, A.; Mortazavi, Y.; Alipour, S. *Sensors and Actuators B: Chemical* **2010,** 147, (2), 400-405.

18. Wagner, T.; Bauer, M.; Sauerwald, T.; Kohl, C. D.; Tiemann, M. *Thin Solid Films* **2011,** 520, (3), 909-912.

19. Basu, S.; Basu, P. K. *Journal of Sensors* **2009,** 2009.

20. Schedin, F.; Geim, A. K.; Morozov, S. V.; Hill, E. W.; Blake, P.; Katsnelson, M. I.; Novoselov, K. S. *Nat Mater* **2007,** 6, (9), 652-655.

21. Weiss, N. O.; Zhou, H.; Liao, L.; Liu, Y.; Jiang, S.; Huang, Y.; Duan, X. *Advanced Materials* **2012,** 24, (43), 5776-5776.

22. Weiss, N. O.; Duan, X. *NPG Asia Mater* **2013,** 5, e74.

23. Mao, S.; Pu, H.; Chen, J. *RSC Advances* **2012,** 2, (7), 2643-2662.

24. Barsan, N.; Weimar, U. *Journal of Electroceramics* **2001,** 7, (3), 143-167.

25. Gaster, R. S.; Hall, D. A.; Wang, S. X. *Lab on a Chip* **2011,** 11, (5), 950-956.

26. Guan, W.; Rajan, N. K.; Duan, X.; Reed, M. A. *Lab on a Chip* **2013,** 13, (7), 1431-1436.

27. Mazzeo, A. D.; Kalb, W. B.; Chan, L.; Killian, M. G.; Bloch, J.-F.; Mazzeo, B. A.; Whitesides, G. M. *Advanced Materials* **2012,** 24, (21), 2850-2856.

28. Ting, H.; Hu, J.-B.; Hsieh, K.-T.; Urban, P. L. *Analytical Methods* **2014,** 6, (13), 4652-4660.

29. Chen, Y.-L.; Hu, Z.-A.; Chang, Y.-Q.; Wang, H.-W.; Zhang, Z.-Y.; Yang, Y.-Y.; Wu, H.-Y. *The Journal of Physical Chemistry C* **2011,** 115, (5), 2563-2571.

30. Li, K.; Luo, Y.; Yu, Z.; Deng, M.; Li, D.; Meng, Q. *Electrochemistry Communications* **2009,** 11, (7), 1346-1349.

31. Alexander, C.; Sadiku, M., *Fundamentals of Electric Circuits*. 5 ed.; McGraw-Hill Science: 2012.

32. Razavi, B., *Design of Analog CMOS Integrated Circuits*. 1 ed.; McGraw-Hill Science: 2000.

33. Carusone, T. C.; Johns, D. A.; Martin, K. W., *Analog Integrated Circuit Design*. Wiley: 2011.

34. Raspberry Pi website. http://www.raspberrypi.org/quick-start-guide (1 December, 2014).

35. Adafruit Industries website. http://www.adafruit.com/products/998 (1 December, 2014).

36. Raspberry Pi Spy, Simple Guide to the RPi GPIO Header and Pins.

http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/ (1 December, 2014).

37. Digi-Key website, MCP3008 datasheets. http://ww1.microchip.com/downloads/en/DeviceDoc/21295d.pdf (1 December, 2014).
38. Analog Device website, AD8237 datasheets. http://www.analog.com/static/imported-files/data_sheets/AD8237.pdf (1 December, 2014).
39. Digi-Key website. http://www.digikey.com/product-detail/en/PA-MSD3SM18-08/309-1120-ND/2350160 (1 December, 2014).
40. Advanced Linear Device website. http://www.aldinc.com/pdf/ALD1116.pdf (1 December, 2014).
41. Adafruit Industries website. https://learn.adafruit.com/drive-a-16x2-lcd-directly-with-a-raspberry-pi/wiring (1 December, 2014).
42. Sun, Y.-F.; Liu, S.-B.; Meng, F.-L.; Liu, J.-Y.; Jin, Z.; Kong, L.-T.; Liu, J.-H. *Sensors* **2012,** 12, (3), 2610-2631.
43. Comini, E. *Analytica Chimica Acta* **2006,** 568, (1–2), 28-40.