IPHONE BREATHALYZER

POINT OH WAIT!

By

Gurpal Bhoot

Senior Project

ELECTRICAL ENGINEETING DEPARTMENT

California Polytechnic State University

San Luis Obispo

June 2012

Table of Contents

Table of Figures

List of Tables

Acknowledgements

I would first like to thank my parents for their continued support and dedication of my education. I would also like to thank Professor Chris Lupo for the invaluable advice and assistance that helped complete this senior year final project. His help and guidance was instrumental in finishing the project in a timely fashion.

Abstract

This final year project combines hardware and software components to create an alcohol breathalyzer that is compatible with the iPhone. The final design of the circuit and the iPhone application is original and combines multiple components to create the final product. Most of those components are seen in a simplified version of a breathalyzer, while the added components take care of the interface to the iPhone.

I. Introduction

The appeal for iPhone applications has never been greater and the market for compatible products continues to grow. Not to say that there isn't enough applications already available for the iPhone and iPad users, but there is always room for more. An idea that has yet to fully impact the Apple App Store is that of an actual breathalyzer. This project uses a unique design to incorporate a combination of hardware and software to display the blood alcohol content of an individual. This idea is unlike anything else that is available in the market today, and has room for expansion to create a new breed of products. With the simple replacement of different sensors and a little modification of the biasing circuit, new products can easily be built that are compatible with the iPhone and iPad.

II. Background

The idea of a breathalyzer is hardly anything new, with hundreds of different models currently available. Yet every one of these breathalyzers requires additional means of power, mostly in the form of batteries. Another aspect of the breathalyzer that separates various models is accuracy. This is certainly one of the more important features of the breathalyzer, which comes down to the type of sensor that is used. There is a variety of gas sensors that detect alcohol available in the market, with the difference coming down to a number of factors. There is the preheat time, which is the required number of hours the sensor needs before being able to deliver accurate results. During this heat up time, it is expected that the sensor requires power. In order to understand the elements of a breathalyzer, the first step is to determine how to collect the blood alcohol content from a sample amount of alcohol in the breath.

II.1 Blood Alcohol

Blood alcohol content (BAC) is measured as a percentage of alcohol that is contained in the blood. The most accurate test for this measurement is blood tests; the only problem for these tests is the method in which they are taken. While these tests are the ideal way in determining a person's BAC level, they are not portable and therefore cannot be used during DUI (Driving Under the Influence) situations, especially by a law enforcement officer. This led to a new method of measuring BAC level, through the breath of a person. It was discovered in the early 1900s that alcohol in the breath has a direct correlation with alcohol in the blood. This type of measurement relies on the amount of alcohol present in the air, which is usually recorded in volume.

II.2 Breathalyzer

A brief overview of the components required for the breathalyzer is discussed in this section. These components include a gas sensor, microcontroller and a display panel. The

interface between each of these devices is discussed in section. The components are shown in Table I below along with Figures 1, 2 and 3.

Table I: BREATHALYZER COMPONENTS

COMPONENT	OPERATION	IMAGE
Alcohol Sensor	Detects the amount of alcohol present in the air.	 **Figure 1: Alcohol Sensor**
Microcontroller	Used to control the sensor and display screen, by deciding when to turn the sensor on and what to display on the screen. Also, used for calibrating the sensor.	 **Figure 2: Microcontroller**
Display Screen	Displays the final BAC value.	 **Figure 3: Serial LCD Screen**

Figure 1 shows a typical alcohol sensor. The alcohol sensor operates by heating an element consisting of Alumina and SnO2 (Tin Oxide). Alcohol molecules react with the alumina and SnO2 and create acetic acid. The acetic acid then enables current flow. The higher the content of alcohol, the more current flows. This principle can be used to create voltage divider where voltage changes with alcohol content in the air. Figure 2 shows a microcontroller. The microcontroller controls and applies the heater voltage and captures

the analog voltage of the voltage divider. The analog voltage is converted in to a digital value using the analog to digital converter in the microcontroller. This is used to calculate the blood alcohol content as a percentage. The result is then shown on the serial display (Figure 3). The microcontroller is also used to calibrate the sensor.

III. Requirements and Procedure

III.1 Requirement

The requirement for this project to be considered successful is the accurate display of the BAC level using an iPhone application. In order to have an accurate BAC level, the sensor needs to be calibrated, converting the output sensor voltage into a readable BAC percentage. This requires a circuit that allows for the sensor to be turned on for a period of time before it is ready to take measurements.

III.2 Procedure

The following steps are required in order to classify this project a success.

1. Design a circuit that allows an alcohol sensor to function and make recordings.
2. Calibrate the sensor to show the user's BAC level.
3. Construct an application for the iPhone.
4. A communication protocol between the circuit and the iPhone application.
5. Read and display the output BAC level on the iPhone application

The first two steps are tied together, as the circuit being constructed must take into account the need for calibration. Should there need to be an alteration in the design of the circuit to accommodate the calibration, it will be taken care of at this stage. Once the circuit is constructed, the process of creating an application for the iPhone can begin. The iPhone application should be user friendly and able to read and contain a digital interpretation of the BAC level as well as a graphical user interface. The application must be easy to use and the whole process of determining one's BAC level should be a quick process; the time from when the app starts to when the BAC level is shown shouldn't take longer than 10 seconds per measurement.

IV. Design

The design of the circuitry is based on the different components of a breathalyzer that are discussed in section II Background. Along with these components, there are several more needed to ensure that the requirements were met. To create the communication between the microcontroller and the iPhone, a component called “HiJack” was used. The HiJack takes in an analog voltage and modulates it so the iPhone can read the incoming transfer through the audio headset. This is achieved using frequency shift keying and allows for a smooth route from the alcohol sensor’s data pin to the iPhone itself. Of course, the rest of the design can be found on the software side of the project, consisting in the iPhone application. Before any type of communication can exist, the first step is to choose the right alcohol sensor for this type of project.

IV.1 Alcohol Sensor

There are quite a few alcohol sensors that are available on the market, but two were chosen due to operation and availability. An additional requirement is to use a alcohol sensor that is fairly cheap and provides a quick response to alcohol that is detected in the air. Another necessity is to use a sensor that has low power consumption. This is based on the amount of current required to operate a sensor and obtain an accurate BAC reading. After doing some research, it turned out most of the sensors available operated at 5V with the heater requiring around 120 mA. With the amount of power this sensor would draw under those conditions, it seemed like a reasonable amount of power, as long as the sensor wasn’t turned on for an extended amount of time. The two sensors that were chosen to investigate further are the MQ303A and the MQ-3 sensors.

IV.1.1 MQ-3

The first sensor purchased was the MQ-3. The reason for this sensor was the affordable price at only $5.50. The positives for this sensor are:

- Good sensitivity to gas
- Low cost
- Simple circuit

The biasing circuit for this device was indeed very simple. With the microcontroller, 5VDC power and a load resistor the only components needed to operate the sensor. The circuit for this sensor is shown in Figure 4 below.

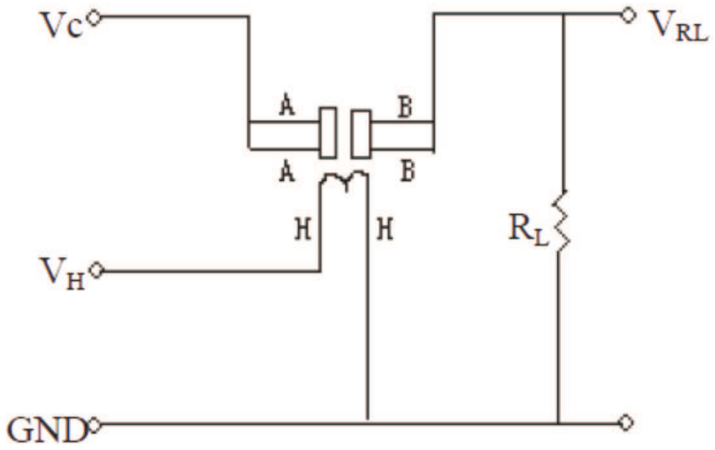


Figure 4: MQ-3 Test Circuit[1]

Figure 4, shows that the sensor is provided a voltage from the power source (Vc) and the output voltage (V_{RL}) measured across a load resistance (R_L). The output voltage can be directly connected to an analog input pin on the Arduino Uno microcontroller and calibrated to calculate an appropriate BAC level.

One of the problems encountered with this particular sensor is the time required for the heater. The MQ-3 datasheet shows that 48 hours of pre-heat time is required, this means the heater voltage V_H requires a 5VDC supply in order for the sensor to give off accurate readings. This was deemed not acceptable, due to the amount of power this would drain. The conditions stated in the MQ-3 datasheet stated that the voltage V_C needs to equal V_H

[1] SOURCE: http://www.sparkfun.com/datasheets/Sensors/MQ-3.pdf

at 5VDC, with the heater requiring a current of around 120mA. This would result in at least 600mW of power consumption for 48 hours, which is a lot of power for the iPhone to provide. This would also mean that the alcohol sensor needs to be powered for 48 hours before it can be used. Based on web searches, the maximum power that can be supplied by the iPhone is 500mw, so this sensor cannot be used unless an external power supply is also used.

IV.1.2 MQ303A

The second sensor is very similar to the first sensor and operates similarly. The same amount of voltage is needed and the sensor's heater requires the same amount of current. The biggest advantage the MQ303A sensor had over the previous sensor was the amount of pre-heat time required. Instead of the 48 hours, only 24 hours heat time is required. Also, an additional option exists to significantly reduce the heat time. The test circuit for this sensor is shown below in Figure 5.

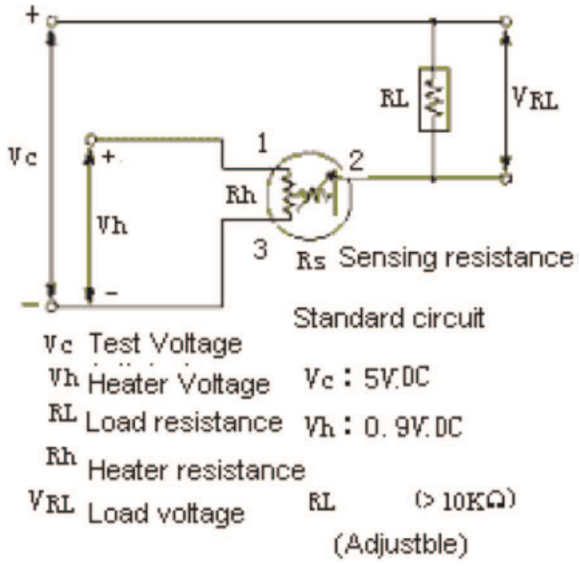


Figure 5: MQ303A Test Circuit[2]

As seen in Figure 5, the voltage (V_C) required to operate the device is 5V, however only

[2] SOURCE: http://www.seeedstudio.com/depot/images/product/MQ303A.pdf

0.9V is required for the heater (Vh). Once the heater is turned on, the resulting voltage across the load resistance (RL) is is proportional to and determines how much alcohol is present in the air. A different method for preheating the sensor is stated in the sensor's datasheet; this is to apply a 2.2V to the heater (Vh) for about 5 to 10 seconds. This would significantly decrease the preheat time and enable quick measurements. Furthermore, this also significantly reduces the power consumption. The power required for preheat is 264mW and the power required during BAC measurement is only 108mW. This meets the power requirements of the iPhone. This sensor allows the time from starting the app to showing BAC level to be fast (6 to 10 seconds)

IV.1.3 Sensor Summary

With both sensors operating at very similar conditions, it was easy to choose between the two. The first sensor's, MQ-3, main issue was the pre-heat time; giving the second sensor, MQ303A, a clear advantage. Also, the fact that this sensor had a separate pin which is used for supplying voltage directly to the heater gave the circuit more flexibility, making the circuit easier to work with. Once the right sensor was chosen, further investigation went in to the circuit and ways of improving. The MQ303A sensor was also available on a printed circuit board that came with a load resistance of 10kΩ. Therefore, the connectivity between the alcohol sensor and the microcontroller is straight forward. The schematic for this sensor board can be seen in Figure 6 below.

This schematic shows the various components and the alcohol sensor (U1). The jumper (J1) on the upper left hand side is used to select the power source (either USB or battery). When the jumper (JP1) is used to select USB power, the voltage VCC provides the power from USB to COM after the two diodes perform a voltage drop. The regulator U2 then provides the voltage for the heater.

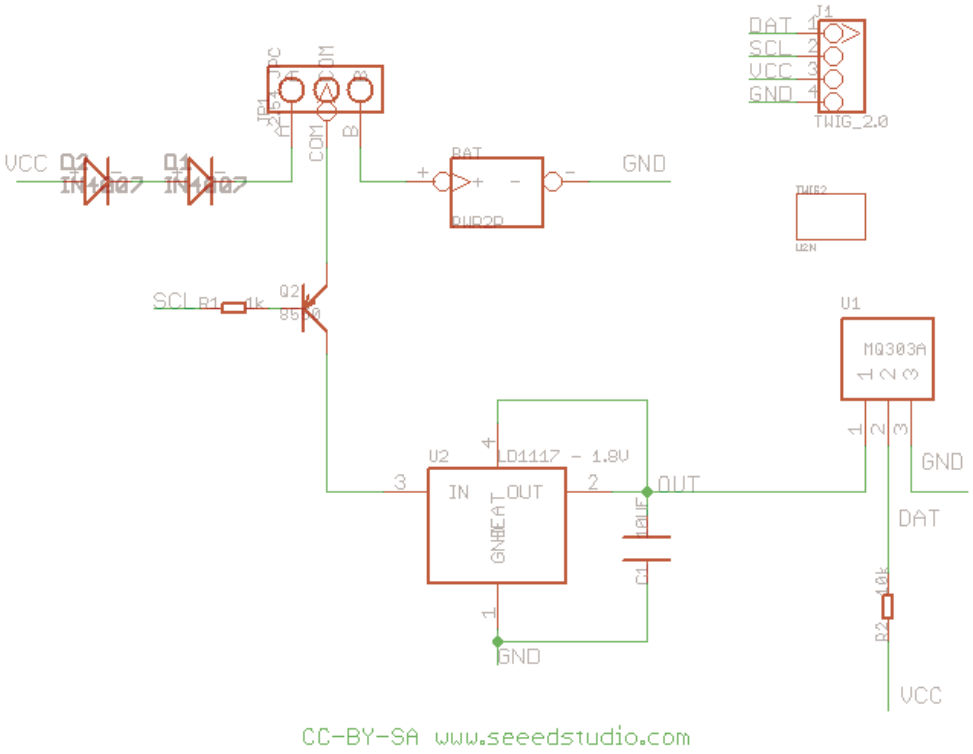


Figure 6: Grove Alcohol Sensor Schematic[3]

As will be discussed in the section containing the HiJack information, a change in the amount of voltage being supplied to the circuit is required. For now, it is assumed that value is 3.3V instead of the original 5V. With a 3.3V supply, the voltage after the two diodes would be less than the 2.2V required for fast preheat time. So the path chosen is to use the battery pin, which would bypass the voltage drops and allow a full 3.3V to enter the emitter pin for the PNP transistor, Q2. Since the PNP transistor acts like a switch, it can be seen the SEL pin is active low, where 0V at the SEL pin would allow the transistor to operate in the forward-active region. Therefore, approximately 3.3V would be seen at the collector junction of this BJT when the SEL pin is set to low. The SEL pin is then connected to the microcontroller. This allows the Arduino board to control the sensor and turn the sensor on and off as required.

The LD1117 is a voltage regulator which connects to the heater pin on the MQ303A

[3] SOURCE: http://www.seeedstudio.com/wiki/Grove_-_Alcohol_Sensor

sensor. Once the heater pin has received the 2.2V needed to preheat the sensor for the allotted time (5 seconds), the alcohol sensor is ready to take accurate measurements. For the next step, there were a number of methods that were taken into consideration. The DAT pin, which is the analog voltage across the load resistance (10kΩ), can either be connected to the Arduino microcontroller or straight to the input of the HiJack. This part of the design is discussed in the following section, based on the microcontroller that will be used.

IV.2 Microcontroller Software

The primary microcontroller used for this project is the Arduino Uno. The reason this microcontroller was chosen was due to the 5V operating power and the number of the I/O pins, making for an easier process to calibrate the alcohol sensor. The Arduino Uno has the ability to operate at 3.3V, which is used to lower the total power consumption. There are analog output pins that are suited for both sets of voltage levels; therefore, the transition to 3.3V was not a major problem. The microcontroller is used to heat the alcohol sensor, which would send a digital voltage, either 3.3V or 0V, to the SEL pin on the sensor board.

The method for receiving the output voltage from the sensor can be implemented in a couple of different ways. The first method is to connect the DAT pin to an I/O pin on the microcontroller. This method would require that voltage to be calibrated into an appropriate BAC level using the testing phase of this project. In order for this to be successful, the voltage would then need to be transferred from the microcontroller to the HiJack device. There are many functions the Arduino board is capable of managing, however the digital write function fails to fit the needs for this part of the project. If the sensor was to be calibrated during this phase of the design, the calibrated BAC level would need to be transferred to the iPhone through the HiJack device. The digital write function only allows PWM signals to be written out, which is not the desired output signal. Therefore, a different method needed to be used which allows serial communication with the HiJack.

The second method that could be portrayed would be the direct connection between the

sensor's DAT pin and the HiJack device. This would overcome the microcontroller's inability to adjust digital signals. The microcontroller would still control whether the sensor's heater is turned on or not, but the destination of the output voltage would go directly to the HiJack. This part of the design will be explained in more depth during the next section.

IV.3 iPhone Connectivity

One of the most important steps in designing the overall project is the communication between the microcontroller and the iPhone. The first idea was to go through the dock connection using the 30 pin proprietary connector. However, to obtain the technical details on the connector, including the pin out and the power requirements requires joining the Apple MFi (Made for iPhone) Program. During research, it was also discovered that an authentication chip may also be required. The authentication chip is also only available to members of the MFi. The Apple MFi Program is only available to registered companies, so this was no longer considered an option. An alternative method had to be used. This required further research and investigation.

The solution came in the form of a component called the HiJack. This device uses frequency-shift keying to communicate with the iPhone. This communication uses the headset jack of the iPhone, where the inputs to the HiJack would come from the microcontroller.

Another adjustment came with the voltage source. The first plan was to use the power from the iPhone to power the sensor and the circuit. But as a headphone jack does not supply power, the power supply needed an alternative source. For the purposes of this project, an external power supply is needed to power the iPhone Breathalyzer. There was also the problem that was briefly mentioned in the sensor summary; the HiJack could only handle 3.3V. More specifically, the TI MSP430 (microcontroller from Texas Instruments) which is the HiJack is based on, can only tolerate a maximum of 3.3V. Therefore, the 5V, which powered the sensor, needed to be altered, and since the sensor heater only required a maximum of 2.2V this problem could be handled quite easily. The power source was

decreased from 5V to 3.3V as this would provide enough power and voltage to heat the alcohol sensor. The only problem would be the accuracy of the sensor; with the decreased VCC value there would be smaller ranges to work with for the corresponding BAC percentages. Again, for the purpose of this project, we would only need to justify if the legal limit had been passed, so determining whether the user had been over the limit could easily be justified by this decreased amount of accuracy. The following section will discuss the HiJack device further.

IV.3.1 HiJack

The HiJack component was discovered as a means of transmitting an analog voltage through the audio input of the iPhone. This component was chosen for this project due to that very description, which allowed a quick route to displaying the BAC level on the iPhone. The HiJack first converts the incoming analog voltage to a digital voltage using an ADC, Analog-to-Digital Converter. Once the voltage has been converted, frequency shift keying is used to modulate the digital signal into an audio tone which the iPhone audio set can read. The microcontroller that handles the conversion and modulation is shown in Figure 7.

The TI MSP430 in Figure 7 takes the analog voltage from the sensor and converts this to a digital voltage. The input analog voltage connects to the microcontroller through pin 6, which directly gets put into the ADC. After the conversion, the voltage is modulated through frequency-shift keying. This ultimately turns the digital voltage into an audio tone which can be read by the iPhone application software. From the conversion, the modulated signal is outputted out of the TI microcontroller through pin 36, which is then sent into the iPhone through the microphone input on the headphone jack.

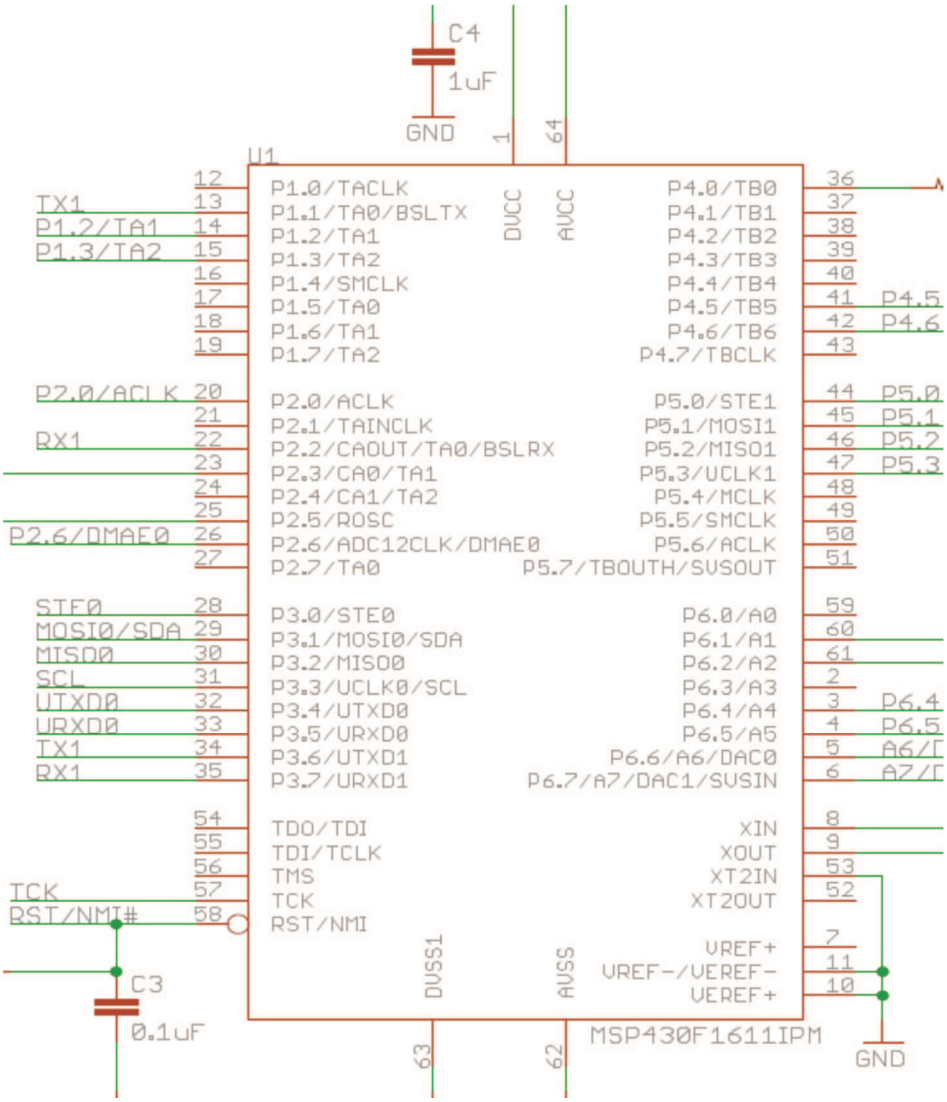


Figure 7: TI MSP430 from HiJack Device[4]

Besides the TI microcontroller, there are three additional pins that the HiJack requires: VCC, GND and DATA. The pins are all self-explanatory; this is another reason why this type of device was perfect for this project. The DATA pin is connected directly to the alcohol sensor. The rest of the process takes place in the software application. This part will be explained further in the following section.

[4] SOURCE: http://www.seeedstudio.com/wiki/images/8/8b/Twig_Alcohol_Sensor_v9.0b.schematic.png

IV.4 iPhone Software

The software tool used for constructing the iPhone application was XCode 4.3.2. This tool came with multiple templates from which to start the application; the template chosen was the View Based App. This was chosen in order to meet the requirement of this project, to show the BAC level on the application. This template allowed for design and multiple visual options to show the user's BAC limit in comparison with the BAC limit in the user's present location. The programming language used within XCode and for the iPhone is Objective-C.

Before the HiJack could communicate with the iPhone through the headset, an app needed to be constructed which would receive the incoming tone and be able to put a digital value on it. There were several steps which needed to be taken in order to make this possible. The first step was to ensure all the needed files and frameworks were added to the project file. The framework needed, AudioToolbox, allowed the HiJack's output going into the headset to be read from within the application. A library was used to read the incoming audio tone and convert into a digital value that could be shown on the application itself. A function was written to receive the incoming signal. A snippet of code is shown in the Figure 8 below.

```
-(int) receive:(UInt8)data {
        double sensorValue=data/77.1;
        self.viewController.sensorValue = sensorValue;
        return 0;
}
```

Figure 8: HiJack Receive Function

The library that was created for this device needs only the above function to assign the signal to a variable. The above function assigns the variable *data* as a number from 0-255. This number is compared to the voltage that is given to the VCC pin from the HiJack device. For example, if the voltage of the DATA pin matched the VCC voltage, then *data* = *255*. The first step was converting this variable to a recognizable number, which meant dividing the variable *data* by ***77.1***. This digit was chosen after a number of tests, giving this

variable a meaningful number. Therefore, the variable *sensorValue* would be exactly that, the voltage coming out of the output of the sensor.

The next phase for the development of the application was to accurately read the voltage coming from the HiJack. The voltage that is read on the iPhone would need to match that of the incoming voltage. This was done by creating a text field on the application and displaying the contents of the variable, *sensorValue*.

Figure 9 shows the test circuit used to calibrate the iPhone voltage to the measured voltage. This consists of a power supply which connects to the VCC pin on the HiJack. A voltage divider consisting of two 100kΩ resistors is used to create a reference voltage of 1.65V. This voltage is connected to the DATA pin on the HiJack device and is connected to the iPhone through the headset.

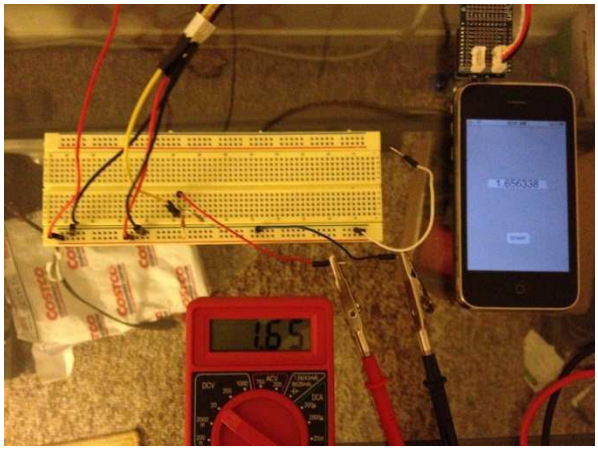


Figure 9: Test Circuit for calibrating iPhone Voltage

Figure 10 shows the measured voltage of 1.65V across the potential divider using a digital meter.

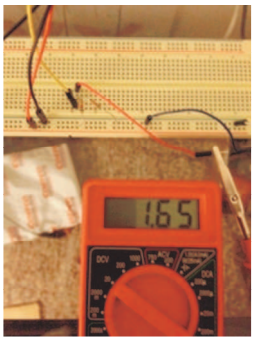


Figure 10: Potential Divider Measured Voltage

Figure 11 shows the voltage calculated and displayed on the iPhone (1.65V). The HiJack is the device plugged in to the headphone socket of the iPhone.

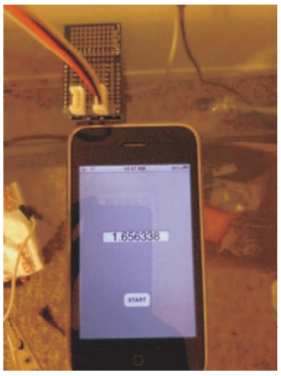


Figure 11: iPhone Calculated Voltage and HiJack

Once the calibration was accomplished, the rest was down to the design of the application. Since the analog voltage was being read on the iPhone with accuracy within 100mV, the calibration and testing procedure was underway.

IV.4.1 Application Design

The one requirement for this project came in the form of displaying the actual BAC level on the iPhone through an application. This was the first responsibility on putting together the design for the application. Another part of the application, which would be useful, is the BAC limit of the user's present location. On top of a digital representation of the user's BAC

level and the current BAC limit, a graphical representation is required. All of these features can be seen in the first version of the iPhone application. Figure 12 shows the application when first started on the iPhone. The features discussed previously are available in the current version of the application. Shown on the left hand side is the user's BAC level in comparison with the local limit. On the right hand side, a digital representation of this comparison is displayed. There are 15 bars, filled with colors that range from green to red, with the middle bars consisting of various shades of yellow. The closer the user is to the local limit, the more bars are filled from the bottom up.

Figure 12: iPhone Application – Startup

Table II shows various screen shots of the iPhone software application in use. Figure 13 shows the screen shot of the iPhone application as the blood alcohol is being measured. During this time, the user is blowing on to the sensor. Figure 14 shows the results when approaching the legal limit in California. Figure 15 shows the results when the legal limit in California is exceeded.

The simple features discussed above have been inserted into this version of the app. Shown on the left hand side is the user's BAC level in comparison with the local limit. On the right hand side, a digital representation of this comparison is displayed. There are 15

bars, filled with colors that range from green to red, with the middle bar consisting of the color yellow. The closer the user is to the local limit, the more bars are filled from the bottom up. A few figures are displayed in the following table, showing the process the application goes through to show the user their BAC level.

The first picture shows the application running after the start button had been pressed. There are a few features which are updated while the application is in progress. There is also somewhat of a theme that follows, with some sports references alerting the user how close to the BAC limit they are.

Table II: IPHONE APPLICTION SOFTWARE

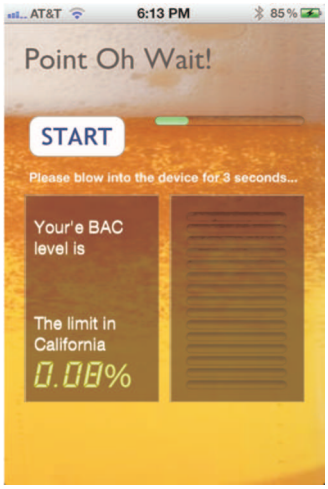
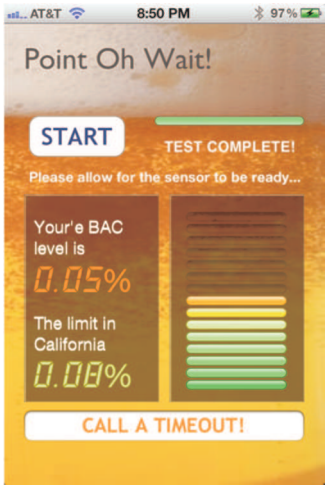
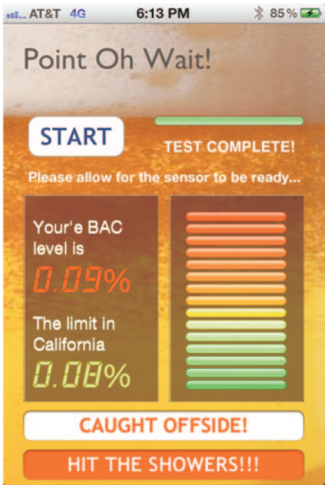
		

Figure 13: iPhone App 1	**Figure 14: iPhone App 2**	**Figure 15: iPhone App 3**
This picture shows that the test is in progress. The progress bar next to the START button will move along and the instructions below that will inform the user how long to blow into the device.	In this image, the test has completed and the results are shown. The user's BAC level is displayed in digital form and also in graphical form. As the user gets closer to the limit, the bar graph moves closer to the top.	Figure 15 shows a different result from the previous image. The test has been completed and the user's BAC is greater than the local limit. There are also alert messages displayed on the bottom of the application.

V. Testing and Calibrating

The primary testing requirement for this project is the calibration and accuracy based on alcohol consumed. The manufacturer of the sensors recommends determining the sensor resistance when it is exposed to 0.4mg of alcohol per liter in air (04mg/l). However, this method requires obtaining the necessary equipment that can accurately provide an air environment with 0.4mg/l. This proved to be very difficult to obtain. To overcome this, a calculation method was used. The Pennsylvania Liquor Control Board has published a report showing approximate blood alcohol contents based on gender, weight and amount of alcohol consumed. These results are shown in Table III below.

Table III: APPROXIMATE BLOOD ALCOHOL PERCENTAGE

	APPROXIMATE BLOOD ALCOHOL PERCENTAGE					
DRINKS	MALE (WEIGHT IN POUNDS)			FEMALE (WEIGHT IN POUNDS)		
	140	160	180	120	140	160
0	0.00	0.00	0.00	0.00	0.00	0.00
1	0.03	0.02	0.02	0.04	0.03	0.03
2	0.05	0.05	0.04	0.08	0.07	0.06
3	0.08	0.07	0.06	0.11	0.10	0.09
4	0.11	0.09	0.08	-	-	-

To calibrate this project, three males and three females were selected that closely matched the body weight of the people used in the Pennsylvania Liquor Control Board experiment. As mentioned previously, the voltage across the load resistance of the sensor changes with the amount of alcohol in the air. Therefore, the blood alcohol results can easily be converted to a corresponding voltage across the sensor. The sensor voltage was measured for each of the males and females. The definition of one drink is the same used by the Pennsylvania Liquor Control Board, each 1½ oz. of 80 proof liquor, 12 oz. of beer or 5 oz. of table wine. The results are shown in

Table IV.

Table IV: ALCOHOL SENSOR VOLTAGE

	ALCOHOL SENSOR VOLTAGE					
DRINKS	MALE (WEIGHT IN POUNDS)			FEMALE (WEIGHT IN POUNDS)		
	140	**160**	**180**	**120**	**140**	**160**
0	3.27	3.27	3.27	3.27	3.27	3.27
1	1.89	1.95	1.98	1.81	1.85	1.87
2	1.73	1.75	1.80	1.49	1.62	1.68
3	1.51	1.61	1.71	1.22	1.35	1.44
4	1.31	1.45	1.50	-	-	-

Although the results in Table IV correlate to those with the Pennsylvania Liquor Control Board, there are some minor differences. For example, a male weighing 140 pounds with 4 drinks should have the same blood alcohol content as a female weighing 120 pounds after three drinks. Factors that were not considered in this experiment were time, and the effect of temperature on the sensor. Either of these could explain the difference between the results. Further investigation revealed that alcohol sensors are susceptible to temperature and humidity. To overcome this susceptibility, one could use a temperature and humidity sensor and compensate for the effects. An alternative solution, which is yet to be confirmed, is to use two sensors and use the difference between the sensors to determine the blood alcohol level. Using this method, the user blows on one, and the other is used to calculate the base level. The difference between to the two sensors is then used to calculate the blood alcohol content.

.

VI. Demonstration

A demonstration of the breathalyzer was scheduled with my project advisor show that the breathalyzer was indeed working and was the sensor would react when alcohol was present in the air. The process for this demonstration was to give the sensor enough alcohol to show the amount of alcohol in the air reached at least 0.01% BAC. This was accomplished by using Scope mouthwash, which contains alcohol in the form of ethanol, and to breathe onto the sensor and demonstrate the working state of the project. At first, the amount of alcohol in the mouthwash wasn't enough to show a change in percent BAC. However, after a couple of attempts the demonstration was confirmed successful when the display on the iPhone application showed an increase in this value, at 0.02%.

VII. Conclusion

The methods and design discussed in this report demonstrate a fully functional breathalyzer for the iPhone. The goal of the project was met, which included

1. Designing a circuit that allows an alcohol sensor to function and make recordings
2. Calibrating the sensor to show the user's BAC level
3. Constructing an application for the iPhone
4. Developing a communication protocol between the circuit and the iPhone application
5. Display and read the BAC level on the iPhone application.

Indeed, the current project is only a prototype, but a working prototype nonetheless.

One aspect of this project which took the most time was finding the right sensor to use given the power requirements of the iPhone. The first sensor considered required more power than the iPhone is able to provide. An alternative was found, but this in turn created an issue that the selected components required different supply voltages. After further investigation, this was resolved by using 3.3V as the core supply insuring all components can operate at 3.3V.

The original specification of the iPhone Breathalyzer required the project to be powered directly from the 30 pin proprietary connector on the iPhone. However, the pin-out, the authentication chip, and the proprietary connector are only available to members of the Apple MFi Program. The Apple MFi Program is only open to registered companies. Two alternatives were investigated to overcome this problem. The first required "jail breaking" the iPhone. Since this voids the iPhone warranty, this was discounted. The second option uses the headphone jack with frequency shift keying to communicate with the iPhone. This method was chosen although the headphone jack does not provide sufficient power for the portable unit. However, the project has been designed to comply with the power

specifications available on the 30 pin connector. The communication protocol used for the headphone jack can also be used on the 30 pin connector. With this in mind, there is no need for altering the design used in this project; this design will be more than sufficient.

VIII. Advancements

This project shows in detail the aspects of the prototype of the iPhone Breathalyzer. All of the components discussed in this report can be implemented on to a single printed circuit board. The schematics for this are shown in Appendix E with the final bill of materials shown in Appendix D.

Further product improvements can also be made. This includes:

- Updating the design for iPhone 30 pin connector. This requires joining Apples MFi Program
- Using two sensors to overcome the effect of temperature and humidity on the sensor
- Replacing the iPhone interface with an interface for Android based phones

These improvements could ensure the successful redesign of the project and allow for a highly marketable product.

IX. Bibliography

1. Seeed Studio Wiki. “Grove - Alcohol Sensor”. October 26, 2011. [Online]. Available: http://www.seeedstudio.com/wiki/Grove_-_Alcohol_Sensor. [Accessed December 10, 2011].
2. SparkFun Electronics. “Alcohol Gas Sensor MQ-3”. [Online]. Available: http://www.sparkfun.com/products/8880. [Accessed December 15, 2011].
3. Seeed Studio. “Grove – Alcohol Sensor”. [Online]. Available: http://www.seeedstudio.com/depot/grove-alcohol-sensor-p-764.html. [Accessed December 15, 2011].
4. Seeed Studio. “Arduino UNO Rev3”. [Online]. Available: http://www.seeedstudio.com/depot/arduino-uno-rev3-p-694.html?cPath=132_133 [Accessed December 15, 2011].
5. Arduino. “Arduino Uno”. [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardUno. [Accessed January 10, 2012].
6. University of Michigan. “Project HiJack”. CSE Division, University of Michigan. [Online]. Available: http://web.eecs.umich.edu/~prabal/projects/hijack/. [Accessed February 17, 2012].
7. Seeed Studio. “Hijack Development Pack”. [Online]. Available: http://www.seeedstudio.com/depot/hijack-development-pack-p-865.html?cPath=174. [Accessed February 18, 2012].
8. Seeed Studio Wiki. “Hijack”. [Online]. Available: http://www.seeedstudio.com/wiki/index.php?title=Hijack#Introduction. [Accessed February 25, 2012].
9. Seeed Studio Wiki. “”Hijack mainboard”. [Online]. Available: http://garden.seeedstudio.com/images/d/da/Hijack_mainboard.PDF. [Accessed February 25, 2012].
10. Seeed Studio Wiki. “Programmer”. [Online]. Available: http://garden.seeedstudio.com/images/9/97/HiJack_Programmer.PDF. [Accessed February 25, 2012].

11. Pennsylvania Liquor Control Board. “Alcohol Impairment Chart”. [Online]. Available: http://www.lcb.state.pa.us/portal/server.pt/community/alcohol___the_law/17511/alcohol_impairment_chart/611972. [Accessed March 20, 2012].
12. Highway Safety Research Center. “Blood Alcohol Concentration (BAC)”. The University of North Carolina. [Online]. Available: http://www.hsrc.unc.edu/safety_info/alcohol/blood_alcohol_concentration.cfm. [Accessed March 22, 2012].
13. Counseling Center. “Blood Alcohol Level”. California State University Bakersfield. [Online]. Available: http://www.csub.edu/counselingcenter/mentalhealth/bloodalcohol.shtml. [Accessed March 22, 2012].

X. Appendix A: Senior Project Analysis

Summary of Functional Requirements

The iPhone Breathalyzer measures the amount of alcohol in the user's breath. The iPhone Breathalyzer consists of two primary components.

1. A portable unit that plugs directly in to the iPhone. The portable unit contains a blood alcohol sensor, sensor biasing circuitry, a microcontroller with an embedded analog to digital converter, and an interface to the iPhone.
2. An iPhone software application that captures, calculates, and displays the blood alcohol level as a percentage of alcohol in the blood.

The iPhone Breathalyzer uses the iPhone's GPS to determine the location and the local legal blood alcohol limit for driving an automobile. This value is compared with the measured blood alcohol limit and a warning is displayed if the blood alcohol limit is exceeded or close to the legal limit.

Primary Constraints

The design of the iPhone Breathalyzer encompassed three main challenges.

1. The main challenge for this project was to find the right alcohol sensor that met the power requirements of the iPhone. The initial sensor used, although very cost effective, required 48 hours of heat time and 600mW of power for this period of time. A sensor was needed with the ability to control the heater voltage and have a low heating time. Furthermore, sensors are susceptible to both humidity and temperature. Both humidity and temperature can change the resistance of the sensor. This susceptibility can be overcome by using two sensors and using the difference between the sensors to calculate the blood alcohol level. With this methodology, the user only blows on one sensor, while the other is used to determine the sensor resistance without alcohol.

2. The original specification of the iPhone Breathalyzer required the portable unit to be powered directly from the 30 pin proprietary connector on the iPhone. However, the pin-out, the authentication chip, and the proprietary connector are only available to members of the Apple MFi Program. The Apple MFi Program is only open to registered companies. Two alternatives were investigated to overcome this problem. The first required "jail breaking" the iPhone. Since this voids the iPhone warranty, this was discounted. The second option uses the headphone jack with frequency shift keying to communicate with the iPhone. This method was chosen although the headphone jack does not provide sufficient power for the portable unit. However, the portable unit has been designed to comply with the power specifications available on the 30 pin connector. The communication protocol used for the headphone jack can also be used on the 30 pin connector.
3. Accurate measurements of the alcohol sensor can only be obtained if the alcohol sensor is calibrated correctly. The resistance of an alcohol sensor changes as it is exposed to alcohol. To calibrate the sensor, the manufacturer recommends determining the sensor resistance (Rs) when it is exposed to 0.4mg/L alcohol in the air. All alcohol calculations are then based on the value of sensor resistance at 0.4mg/L. However, this method requires obtaining equipment that can accurately provide an air environment with 0.4mg of alcohol per liter. This proved to be very difficult to obtain. To overcome this, a calculation method was used to determine the amount of alcohol in the breath. The calculation was based on body weight, amount of alcohol consumed, and elapsed time.

Economic

Economic Impacts

Commercialization of the iPhone Breathalyzer will have economic impact on three items. These are

1. Investment
    a. Developing and releasing the iPhone breathalyzer as a consumer product requires financial investment. It is anticipated that the financial investment will be obtained from either angel investors or venture

capital firms. Both angel investment and the venture capital will be sought from the United States. If successful, the profits will be released in the United States.

2. Manufacture
    a. It is anticipated that the iPhone Breathalyzer will be manufactured in Asia (China or Taiwan). The primary reason for this is cost. There are two aspects to the cost – raw materials and the labor/manufacturing costs. All raw materials for the iPhone Breathalyzer originate from Asia; this includes the sensor, the l semiconductor devices and the housing. It is anticipated that the final device will use green materials, for example lead free solder. This is especially important as users will be breathing in to the iPhone Breathalyzer.
3. Consumption
    a. It is anticipated to market and sell the iPhone Breathalyzer across the world. However, based on the use model, the United States and Europe represent the largest markets. This is due to the amount disposable income and the consumer habits. For example, the iPhone Breathalyzer would not be popular in third world countries or countries where alcohol is prohibited. However, it would be popular in the United States.

Cost & Benefits

The iPhone Breathalyzer has two main costs throughout the life cycle (life cycle is assumed to be from product development to volume manufacture and obsolescence). These costs are

1. Development Cost. This is the cost to develop the prototype. The prototype is used to demonstrate and provide proof of the iPhone Breathalyzer capabilities. The prototype has been demonstrated at the completion of the Senior Project. The original budget for the development was $200. This budget was part of my college

education provided by my parents. The final development costs are summarized in Table V below.

Table V: COMPONENT DEVELOPMENT COSTS

Component Development Costs					
Part Number	**Manufacturer**	**Description**	**Quantity**	**Cost**	**Total Cost**
IOS	Apple	iOS Developer Program	1	$99.00	$99.00
TES52959P	Seeed Studio	Hijack Development Pack	1	$79.00	$79.00
MQ303A	Hanwei Electronics	Alcohol Sensor	2	$5.50	$11.00
MQ-3	Hanwei Electronics	Alcohol Sensor	2	$4.95	$9.90
ARD132D2P	Seeed Studio	Arduino microcontroller development board	1	$29.90	$29.90
276WBU301	Radio Shack	Bread board	1	$8.99	$8.99
	Radio Shack	Various components (resistors, capacitors, wires etc)		$15.00	$15.00
	Listerine	Mouthwash	1	$4.69	$4.69
				Total Cost	$257.48

Additional equipment used for the development was available at the Cal Poly Electrical Engineering lab. This included a digital multi-meter, power supply, and an oscilloscope. Personal equipment used for development included an Apple MacBook Pro and a digital multi-meter.

2. Manufacturing Cost. This is the cost to build volume production. These costs are based on component costs from Digikey and Mouser. The PCB, plastics, and Apple items are estimated based on discussions with various companies (eASIC, Xilinx and Seeed Studio). The final Bill of Materials is shown in Table VI.

Table VI: FINAL BILL OF MATERIALS

Final Bill of Materials						
Item	**Part Number**	**Manufacturer**	**Description**	**Quanity**	**Cost**	**Total Cost**
C1	1uF	TDK	Capacitor	1	$0.01	$0.01
C2	0.1uF	TDK	Capacitor	1	$0.01	$0.01
C3	4.7nF	TDK	Capacitor	1	$0.01	$0.01
C5	0.1uF	TDK	Capacitor	1	$0.01	$0.01
C6	0.1uF	TDK	Capacitor	1	$0.01	$0.01
J1	TBD	Apple	iPhone connector	1	$0.50	$0.50
LD1	VLMG3100-GS08	Vishay	Green LED	1	$0.10	$0.10
R1	10M	Panasonic	Resistor	1	$0.01	$0.01
R2	20M	Panasonic	Resistor	1	$0.01	$0.01
R3	5.1M	Panasonic	Resistor	1	$0.03	$0.03
R4	100K	Panasonic	Resistor	1	$0.01	$0.01
R5	10K	Panasonic	Resistor	1	$0.01	$0.01
R6	10K	Panasonic	Resistor	1	$0.01	$0.01
R7	1K	Panasonic	Resistor	1	$0.01	$0.01
R8	100K	Panasonic	Resistor	1	$0.01	$0.01
R9	100K	Panasonic	Resistor	1	$0.01	$0.01
R10	100K	Panasonic	Resistor	1	$0.01	$0.01
R11	100K	Panasonic	Resistor	1	$0.01	$0.01
R12	499	Panasonic	Resistor	1	$0.01	$0.01
R13	2K	Panasonic	Resistor	1	$0.01	$0.01
S1	MQ303A	Hanwei Electronics	Alcohol Sensor	2	$1.15	$2.30
U1	MSP430F2001IPWR	Texas Instruments	Microcontroller	1	$0.48	$0.48
U2	TBD	Apple	Apple security chip	1	$2.00	$2.00
Y1	MA-505 24.5760M-C0	Epson Toyocom	24.576 MHz crystal	1	$0.38	$0.38
	PCB			1	$1.25	$1.25
	Plastics			1	$1.00	$1.00
				Total Cost		**$8.20**

Timing

The iPhone Breathalyzer is a consumer product. It is anticipated that the first commercial version of the iPhone Breathalyzer will take approximately 6 to 8 months to develop and release as a consumer product. The typical life of consumer products is 12 to 18 months. The product will follow a typical consumer cycle. However, product enhancements could be introduced during the life of the product; this includes items such as games based on the iPhone Breathalyzer. The iPhone Breathalyzer does not require maintenance or operation costs. If the iPhone Breathalyzer fails outside of the warranty period, consumers will simply dispose of the product and purchase a new one. The original estimated development time is

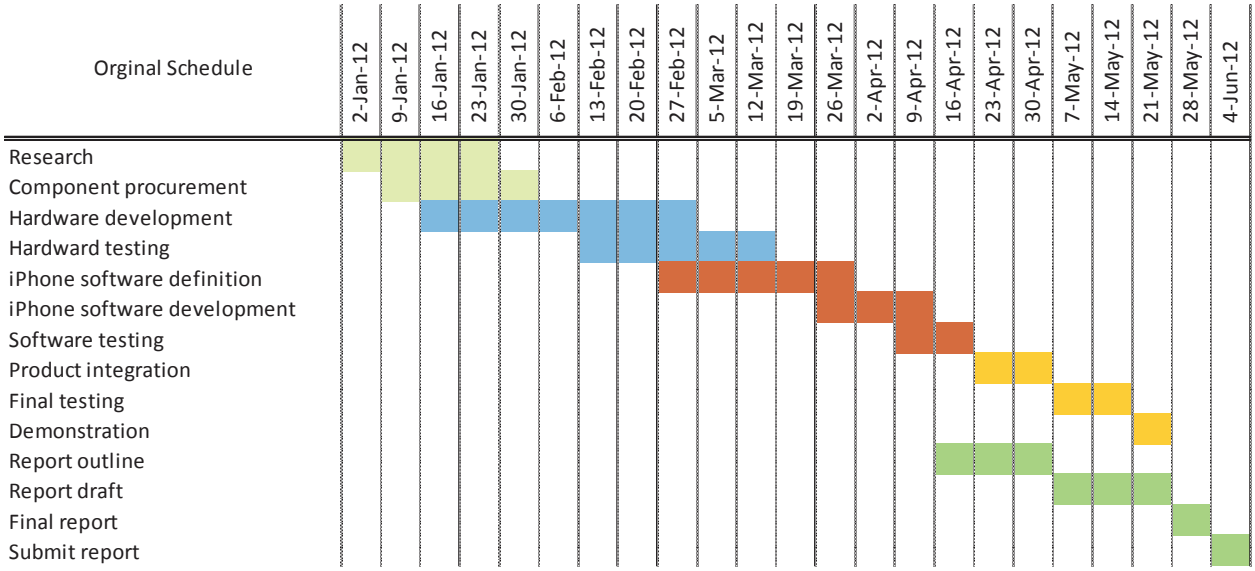


Figure 16: Original Development Schedule

The actual development time is shown in Figure 17.

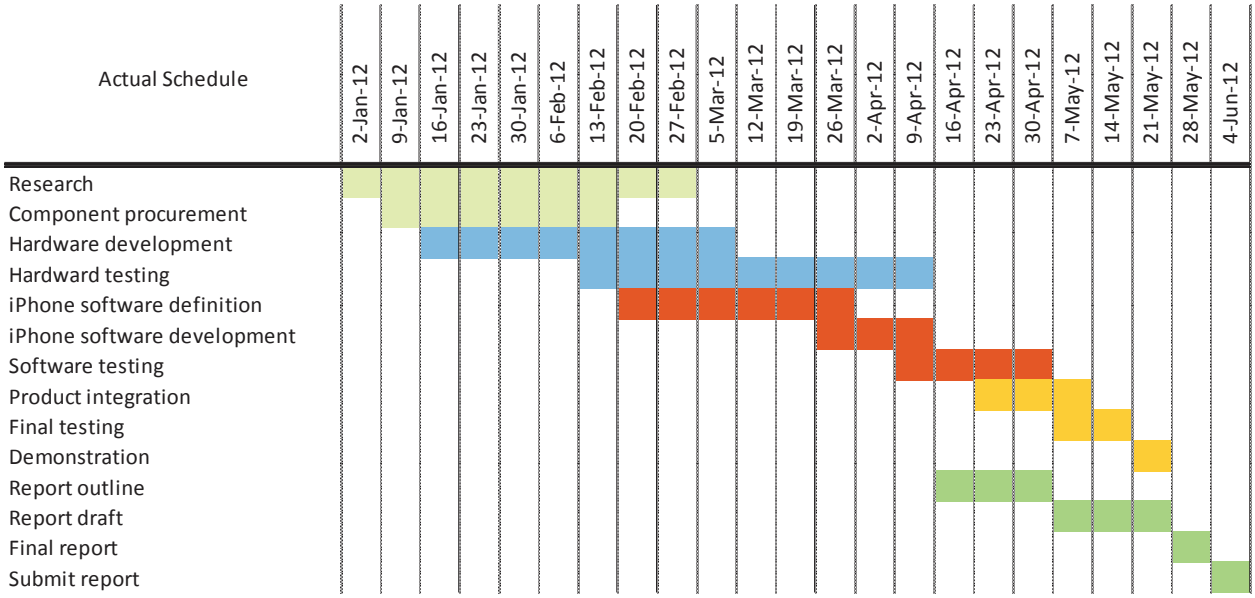


Figure 17: Actual Development Schedule

The prototype of the iPhone breathalyzer is actually the beginning and not the end. This product can be the basis of a new company dedicated to providing iPhone based sensor products. The core technology can spawn many new products such as temperature,

moisture, heart rate. Additionally, other smart phones such as Android can be supported using the same core technology.

Commercial Manufacture

It is anticipated to manufacture the iPhone Breathalyzer on a commercial basis. The summary of the annual revenue, costs, and profits is shown in Table .

Table VII: ANNUAL REVENUE, PROFIT AND COST

Units sold per year	500,000
Manufacturing cost	$10.00
Purchase price	$14.95
Profit per year	$2,475,000

Environmental

The iPhone Breathalyzer does not use any materials or resources that are considered harmful to the environment. For example, there are no chemical or hazardous materials, no impact to air emissions, or any use of substances during manufacture that would violate environmental laws or regulation. As an example, the manufacture of the iPhone Breathalyzer will also use lead free solder.

Manufacturability

The most challenging aspect of manufacturing the iPhone Breathalyzer as a volume consumer product is the calibration of the alcohol sensor. To overcome this, a special alcohol chamber that has a predetermined level of alcohol in the air will need to be designed and used. However, the calibration needs to be completed quickly; otherwise the manufacturing costs will quickly increase.

Sustainability

Although the iPhone Breathalyzer is designed to be maintenance free. The long term effect of the sensors calibration is currently unknown. The sensor manufacturer has not been able to confirm any calibration effects based on usage.

There are many opportunities for improving the iPhone Breathalyzer. This includes modifying the hardware interface to the iPhone so that it can be used with Android based phones. Further modifications could be made to the software which enables additional usage. For example, adding the ability to record the user and the respective results in to a picture or video. This could then be emailed to a parole officer so that they can remotely monitor alcoholics. Additionally, software developers could create games based on the hardware. The upgrading of the software can be easily installed by using the Apple App Store.

Ethical

Although the iPhone Breathalyzer is designed to assist the consumer in determining the blood alcohol level, it may be perceived that it is encouraging the drinking of alcohol. Or perhaps, encouraging drinking and driving, especially if the iPhone Breathalyzer suggests that the user is close to the legal drinking level. There are aspects where the iPhone Breathalyzer could be misused; this includes contesting the results of a Police breathalyzer test with that of the iPhone Breathalyzer. Especially if there is a fatality involved. A further misuse of the iPhone Breathalyzer is to determine how fast one can get drunk or reach a level of blood alcohol content.

For clarification, the author strongly recommends that no matter how little alcohol is consumed; one should never drink and drive.

Health and Safety

Health and safety is one of the primary benefits of the iPhone Breathalyzer. If used for the original purpose, the iPhone Breathalyzer could insure that less alcohol is consumed. Furthermore, it may be a tool to reduce the number of people who drink and drive, thereby reducing fatalities related to drink driving. The iPhone Breathalyzer can also be used to reduce or control the amount of alcohol consumed, especially with sports athletes who may

have a limit based on their exercise and conditioning schedule.

Social and Political

Social and Political issues are the main areas of contention of usage and perception. On the positive side, I foresee many social gatherings where the IPhone Breathalyzer is the main topic of discussion and usage. Many people will want to simply try and see the results. Consumers may also try to see the effect and results of drinking different types of alcoholic drinks, or possibly creating games that would contend with 'beer pong". There is also a strong possibility that consumers may perceive that the iPhone Breathalyzer promotes drinking and driving. Furthermore, if a consumer uses the iPhone Breathalyzer and determines that they are within the legal limit to drive, but are then stopped and tested with a 'Police Breathalyzer. The Police Breathalyzer may show that they are above the legal limit and therefore are prosecuted. The lawyers would use the results of the iPhone Breathalyzer to defend their clients and try to prove that the provider of the iPhone Breathalyzer is at fault or the Police Breathalyzer is not accurate. This may eventually force the iPhone Breathalyzer to be removed from the market or be made illegal, this would be the many issues seen with Radar Detectors and Radar Jammers.

There are four primary stakeholders for this project. These are as follows:

1. Bars, restaurants and breweries. This group is considered direct stakeholders who have a goal of increasing revenue and profits. The iPhone Breathalyzer could either help increase revenue and profits or decrease revenue and profits. Increase in revenue and profits are achieved by consumers drinking more as they are able to measure the alcohol limit. However, the opposite could also occur as consumers drink less when the alcohol limit is reached. Although, it could be argued that when a consumer reaches the legal limit, they would switch to a non-alcohol drink or start to consume food.
2. Law Enforcement. This group would benefit as the number of consumers who drive while under the influence should decrease dramatically. This would enable law enforcement to focus their resources on alternative and perhaps more important

crimes. This is not to say that drinking under the influence of alcohol is not a serious crime (in my opinion it is a serious crime).

3. Taxis and public transport. This group is an indirect stakeholder and benefits by increased revenue and business. It is expected that consumers who are over the legal driving limit would use taxis or public transport to get to their destinations.
4. Component Suppliers. This group would benefit as the demand of the iPhone Breathalyzer increases; this group is able to generate more revenue by selling more components.

Development

A key part of this project required development of the software application running on the iPhone. Prior to this project, I had no experience on the tools or the language required for developing iPhone applications.

This project also required the iPhone Breathalyzer to be powered directly from the iPhone. Analysis based on static and dynamic power consumption became a key parameter. As a result, a large amount of time was spent analyzing datasheets based on power consumption.

XI. Appendix B: Sensor Datasheets

XI.1 MQ-3 Sensor

MQ-3 Semiconductor Sensor for Alcohol

Sensitive material of MQ-3 gas sensor is SnO_2, which with lower conductivity in clean air. When the target alcohol gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-3 gas sensor has high sensitity to Alcohol, and has good resistance to disturb of gasoline, smoke and vapor. The sensor could be used to detect alcohol with different concentration, it is with low cost and suitable for different application.

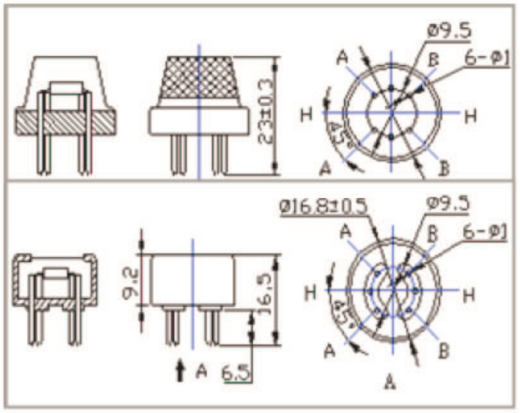
Character

* Good sensitivity to alcohol gas
* Long life and low cost
* Simple drive circuit

Application

* Vehicel alcohol detector
* Portable alcohol detector

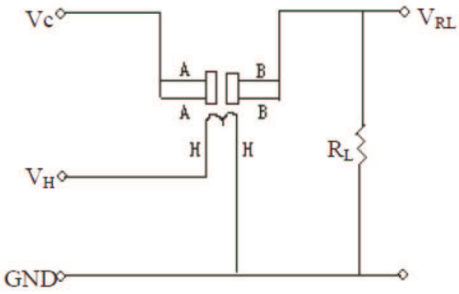
Configuration



Technical Data

Basic test loop

Model No.			MQ-3
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			Alcohol gas
Concentration			0.04-4mg/l alcohol
Circuit	Loop Voltage	V_c	≤24V DC
	Heater Voltage	V_H	5.0V±0.2V AC or DC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤900mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 0.4mg/l alcohol)
			Rs(in air)/Rs(0.4mg/l



The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage（VH） and test voltage（VC）. VH used to supply certified working temperature to the sensor, while VC used to detect voltage (VRL) on load resistance （RL） whom is in series with sensor. The

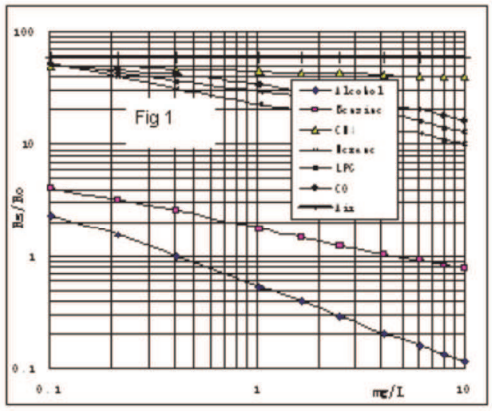
	Sensitivity	S	Alcohol)≥5
	Slope	α	≤0.6(R_{300ppm}/R_{100ppm} Alcohol)
Condition	Tem. Humidity		20℃±2℃；65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V； V_H: 5.0V±0.1V
	Preheat time		Over 48 hours

sensor has light polarity, Vc need DC power. VC and VH could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable RL value is needed:

Power of Sensitivity body(Ps):

$Ps=Vc^2\times Rs/(Rs+RL)^2$

Sensitivity Characteristics



Influence of Temperature/Humidity

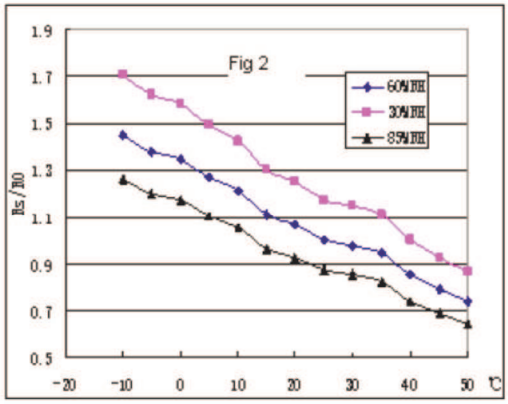
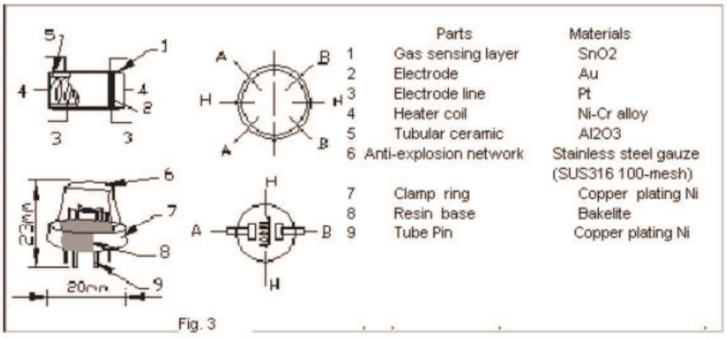


Fig.1 shows the typical sensitivity characteristics of the MQ-3, ordinate means resistance ratio of the sensor (Rs/Ro), abscissa is concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 0. 4mg/l alcohol. All test are under standard test conditions.

P.S.: Sensitivity to smoke is ignite 10pcs cigarettes in $8m^3$ room, and the output equals to 0.1mg/l alcohol

Fig.2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor (Rs/Ro), Rs means resistance of sensor in 0.4mg/l alcohol under different tem. and humidity. Ro means resistance of the sensor in environment of 0.4mg/l alcohol, 20℃/65%RH

Structure and configuration



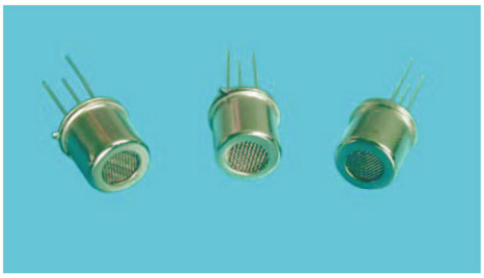
Structure and configuration of MQ-3 gas sensor is shown as Fig. 3, sensor composed by micro AL2O3 ceramic tube, Tin Dioxide (SnO2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-4 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.

XI.2 MQ303A Sensor

MQ303A Alcohol Sensor

Character

* High sensitivity
* Fast response and resume
* Long life and low cost
* Mini Size



Application

MQ303A is semiconductor sensor is for Alcohol detection, It has good sensitivity and fast response to alcohol, suitable for portable alcohol detector.

Configuration

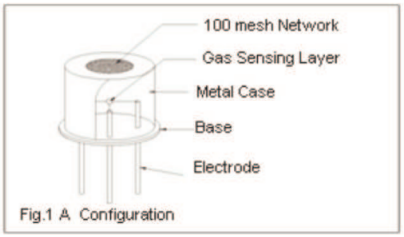


Fig.1 A Configuration

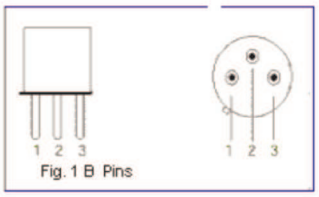
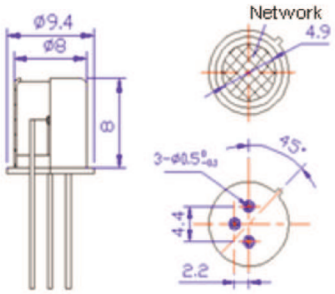


Fig. 1 B Pins

Fig.1 C Circuit

Technical Data

Model No.			MQ303A
Sensor Type			Semiconductor
Standard Encapsulation			Metal
Detection Gas			Alcohol
Concentration			20-1000ppm Alcohol
Standard Circuit Conditions	Heater Voltage	V_H	0.9V ± 0.1V AC or DC
	Loop Voltage	V_c	≤6V DC
	Load Resistance	R_L	Adjustable
	Heater Resistance	R_H	4.5Ω ± 0.5 Ω (Room Tem.)
	Heater Current	I_H	120±20mA
	Heater Power	P_H	≤ 140 mW
Character	Sensor Consumption	P_S	≤10 mW
	Sensing Resistance	R_s	4KΩ-400KΩ (in air)
	Sensitivity	S	Rs(in air)/Rs(125ppm Alcohol)≥3
	Slope	α	0.50 ± 0.15 (R_{300ppm}/R_{100ppm} Alcohol)
Condition	Tem. Humidity		20℃±2℃; 65%±5%RH
	Standard test circuit		Vc:3.0 V±0.1 V DC ; V_H: 0.9 V±0.1 V DC
	Preheat time		Over 48 hours



-

Sensitivity Characteristics

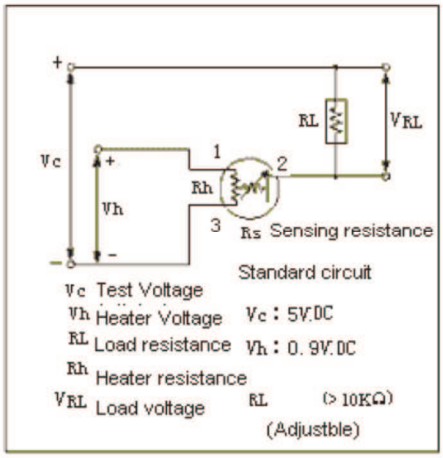


Fig 2 Standard Test Circuit

Fig.2 is the test circuit. You could get resistance change from voltage change on fixed or adjustable load resistance. Normally, it will take several minutes preheating for sensor enter into stable working after electrified; oryou could give 2.2±0.2V high voltage for 5-10secs before test, which make sensor easily stable.

Influence of Temperature/Humidity

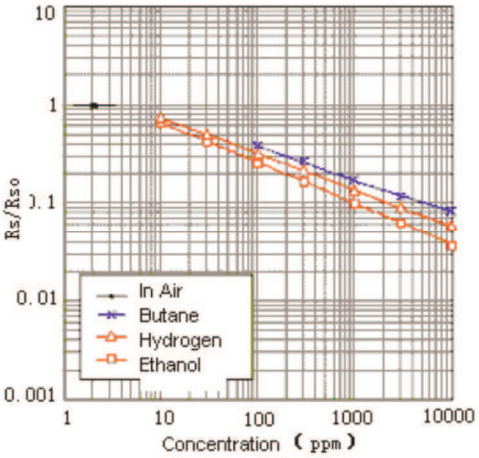
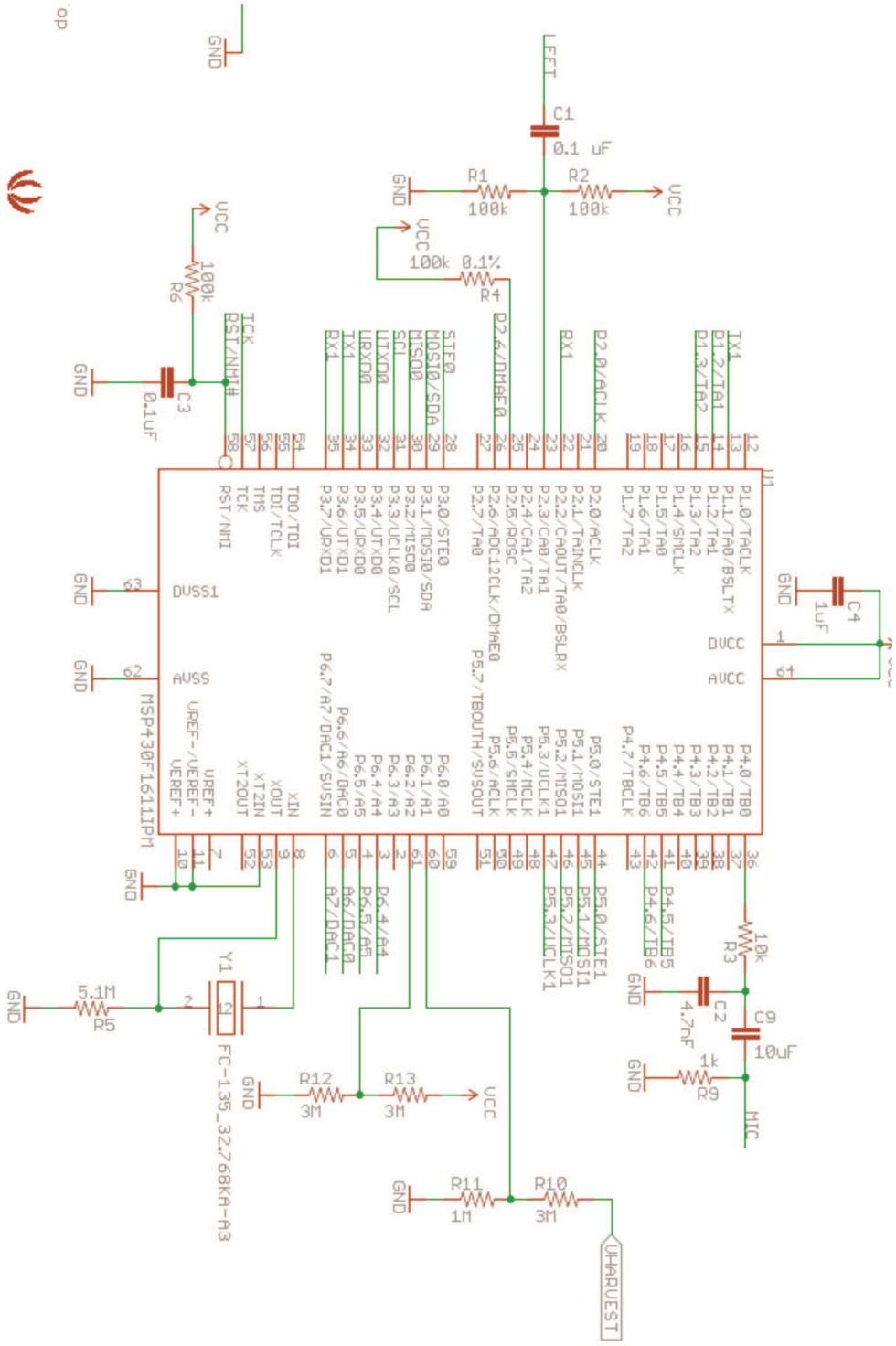


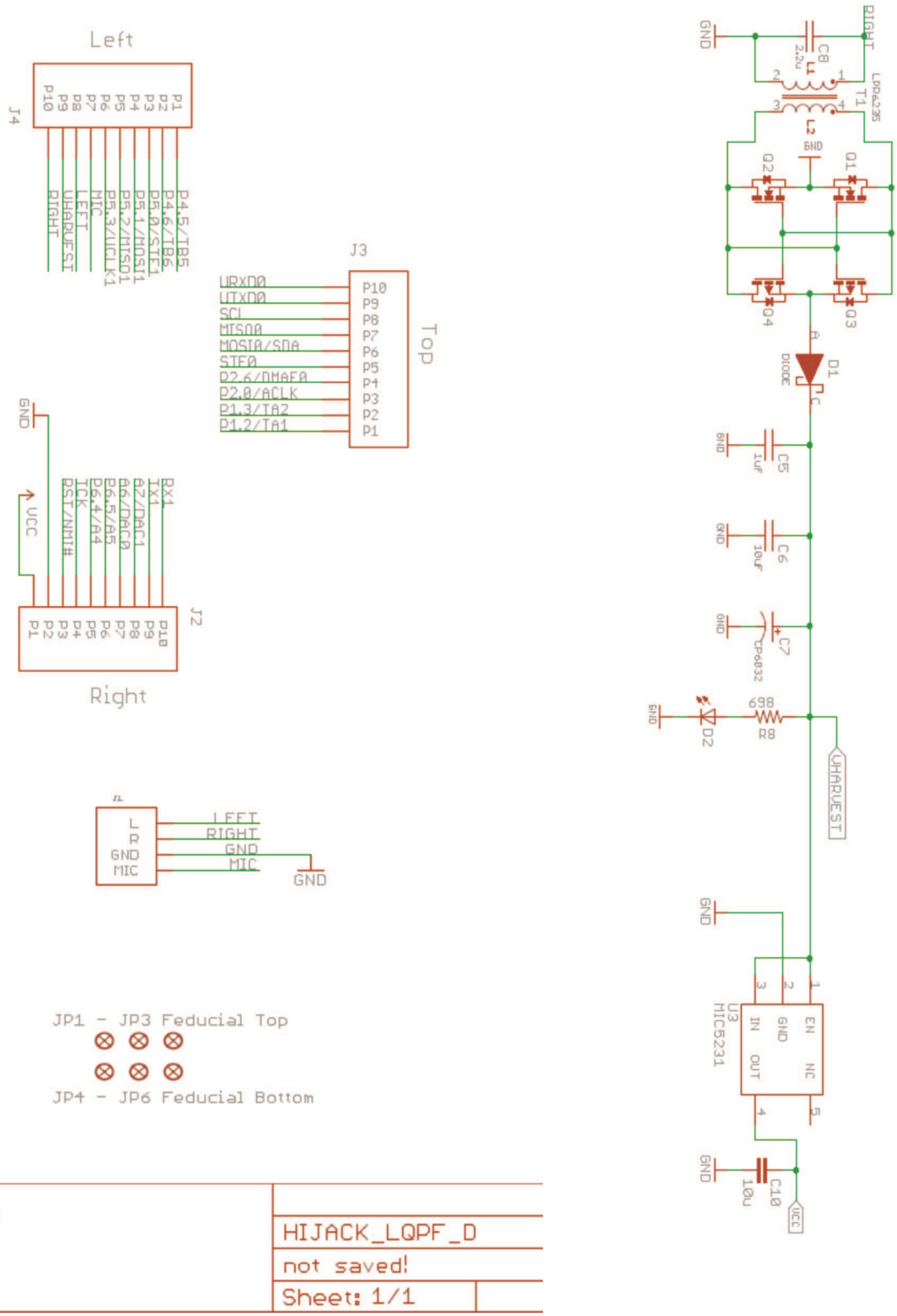
Fig 3 Sensitivity

Fig 3 is sensitivy of MQ303A, it reflects relations between resistance and gas concentration, resistance of the sensor reduce when gas concentration increases

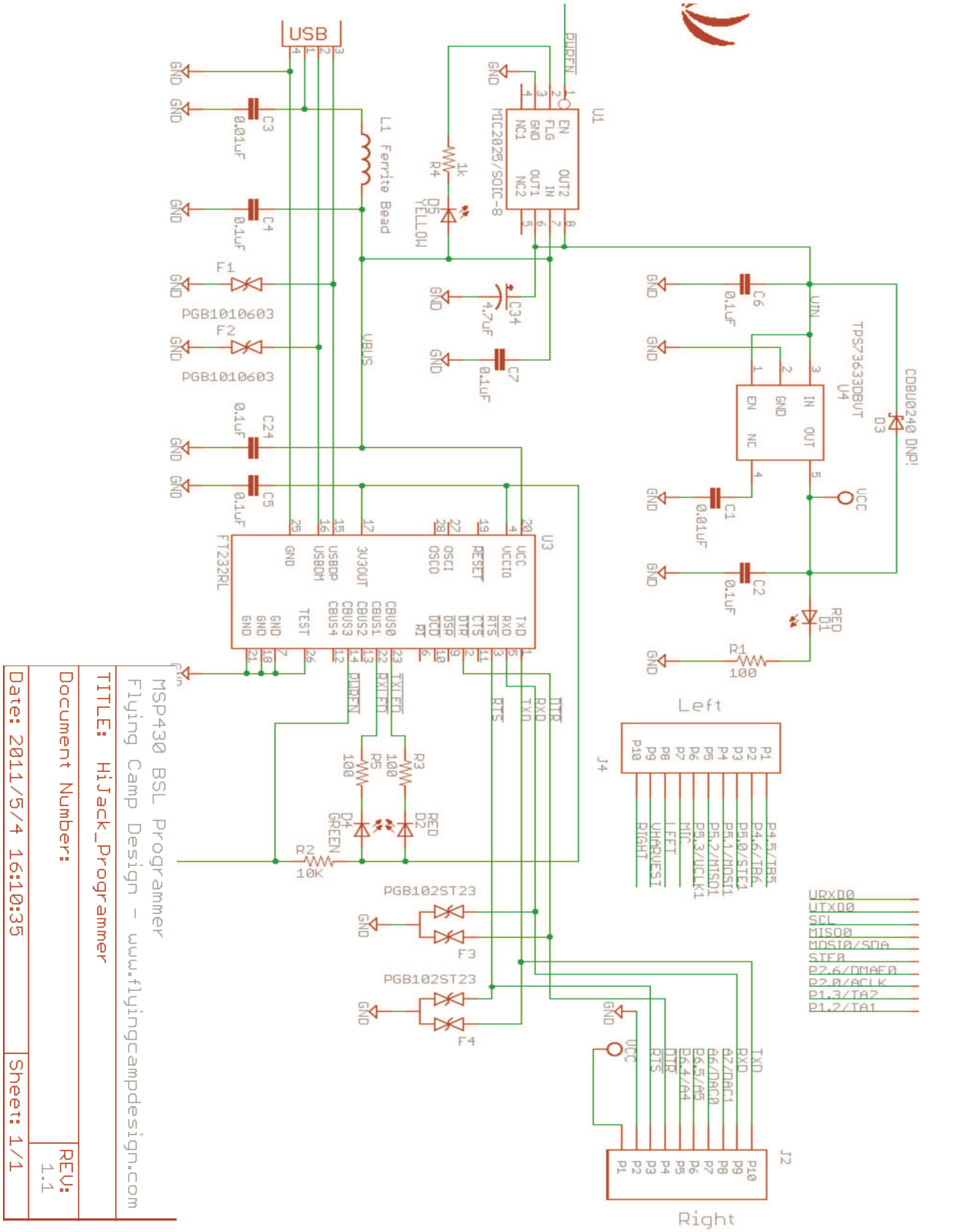
XII. Appendix C: HiJack Schematics

XII.1 HiJack Mainboard

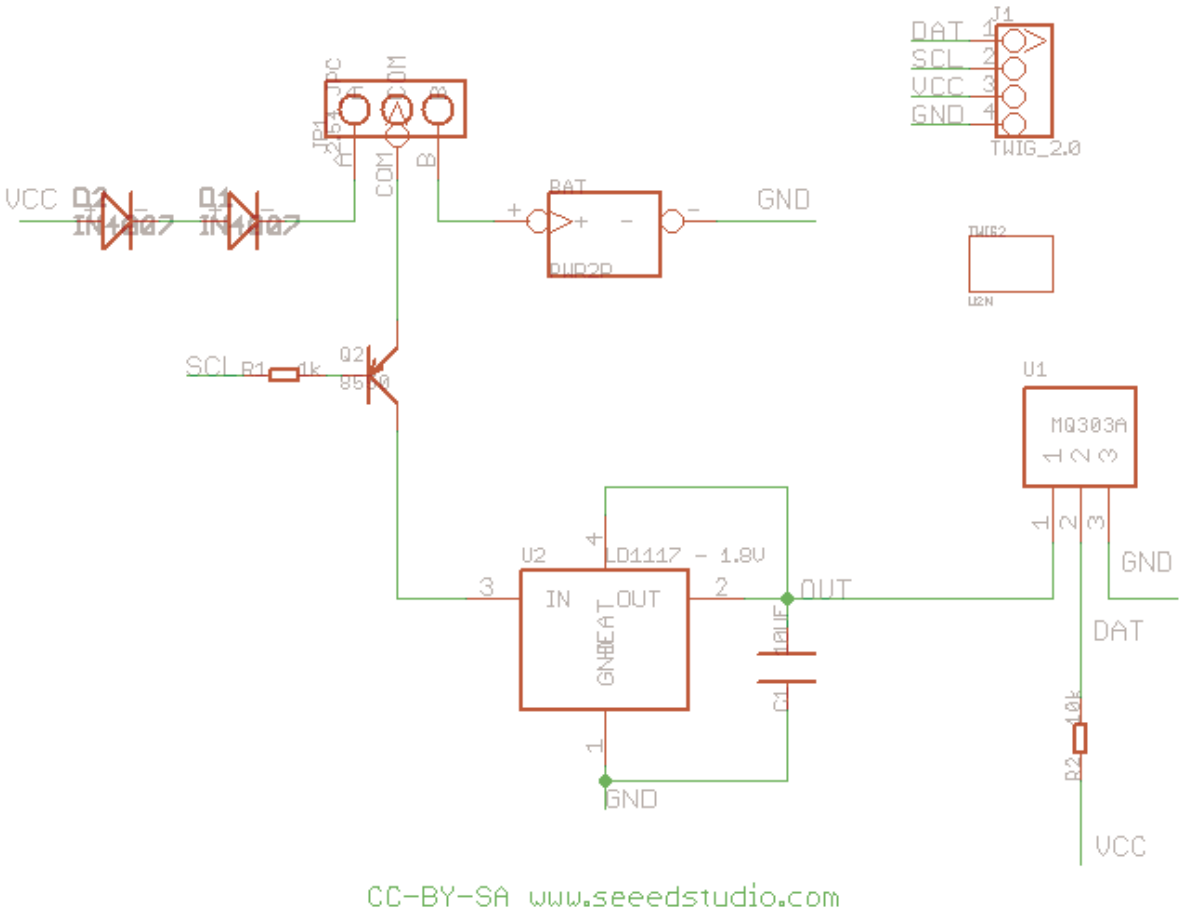



Left
J4
P10
P9
P8
P7
P6
P5
P4
P3
P2
P1
P4.5/TB5
P4.6/TB6
P5.0/STE1
P5.1/MOSI1
P5.2/MISO1
P5.3/UCLK1
MIC
LEFT
VHARVEST
RIGHT
J3
UTXD0
URXD0
SCL
MISO0
MOSI0/SDA
SIE0
P2.6/DMAE0
P2.0/ACLK
P1.3/TA2
P1.2/TA1
Top
GND
VCC
RX1
TX1
A7/DAC1
A6/DAC0
P6.5/A5
P6.4/A4
TCK
RST/NMI#
J2
Right
J1
L R
GND
MIC
LEFT
RIGHT
GND
MIC
GND
JP1 - JP3 Feducial Top
JP4 - JP6 Feducial Bottom
RIGHT
C8
2.2u
T1
LPR6235
L2
GND
Q2
Q1
Q4
Q3
D1
DIODE
C5
1uF
C6
10uF
C7
TP6032
D2
698
R8
VHARVEST
U3
MIC5231
IN
GND
EN
NC
OUT
C10
1.0u
VCC
HIJACK_LQPF_D
not saved!
Sheet: 1/1

XII.2 HiJack Programmer



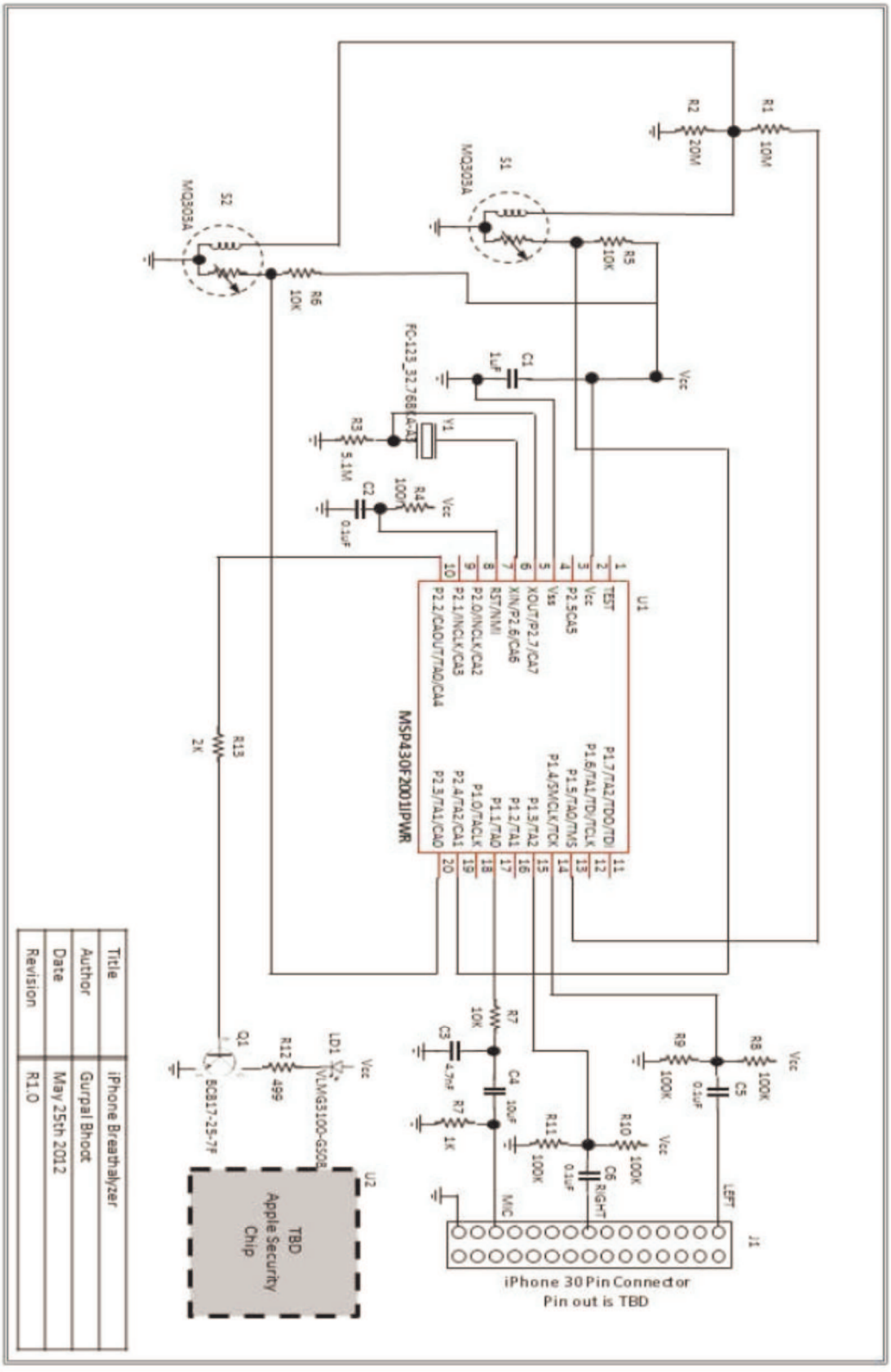
XII.3 Grove Alcohol Sensor Schematic



XIII. Appendix D: Final Bill of Material

Final Bill of Materials						
Item	Part Number	Manufacturer	Description	Quanity	Cost	Total Cost
C1	1uF	TDK	Capacitor	1	$0.01	$0.01
C2	0.1uF	TDK	Capacitor	1	$0.01	$0.01
C3	4.7nF	TDK	Capacitor	1	$0.01	$0.01
C5	0.1uF	TDK	Capacitor	1	$0.01	$0.01
C6	0.1uF	TDK	Capacitor	1	$0.01	$0.01
J1	TBD	Apple	iPhone connector	1	$0.50	$0.50
LD1	VLMG3100-GS08	Vishay	Green LED	1	$0.10	$0.10
R1	10M	Panasonic	Resistor	1	$0.01	$0.01
R2	20M	Panasonic	Resistor	1	$0.01	$0.01
R3	5.1M	Panasonic	Resistor	1	$0.03	$0.03
R4	100K	Panasonic	Resistor	1	$0.01	$0.01
R5	10K	Panasonic	Resistor	1	$0.01	$0.01
R6	10K	Panasonic	Resistor	1	$0.01	$0.01
R7	1K	Panasonic	Resistor	1	$0.01	$0.01
R8	100K	Panasonic	Resistor	1	$0.01	$0.01
R9	100K	Panasonic	Resistor	1	$0.01	$0.01
R10	100K	Panasonic	Resistor	1	$0.01	$0.01
R11	100K	Panasonic	Resistor	1	$0.01	$0.01
R12	499	Panasonic	Resistor	1	$0.01	$0.01
R13	2K	Panasonic	Resistor	1	$0.01	$0.01
S1	MQ303A	Hanwei Electronics	Alcohol Sensor	2	$1.15	$2.30
U1	MSP430F2001IPWR	Texas Instruments	Microcontroller	1	$0.48	$0.48
U2	TBD	Apple	Apple security chip	1	$2.00	$2.00
Y1	MA-505 24.5760M-C0	Epson Toyocom	24.576 MHz crystal	1	$0.38	$0.38
	PCB			1	$1.25	$1.25
	Plastics			1	$1.00	$1.00
				Total Cost		**$8.20**

XIV. Appendix E : Final Schematics



XV. Appendix F: Software Code

In this section, all of the code that was written up to ensure each component was working is shown. First, the code used to program the Arduino microcontroller will be displayed; followed by all of the code used to construct the iPhone application. The Arduino code was written on the specific program used for Arduino boards, Arduino.exe, and the program used for the iPhone development was XCode 4.3.2.

XV.1 Arduino Uno Code

```
#define heatPin A0

void setup()
{
 pinMode(heatPin, OUTPUT);
 digitalWrite(heatPin, HIGH);
 Serial.begin(9600);
}

void loop()
{

 digitalWrite(heatPin, LOW);
}
```

XV.2 iPhone Code

XV.2.1 AlcoholAppDelegate.h

```
//
//  AlcoholAppDelegate.h
//  Alcohol
//
//  Created by Gurpal Bhoot on 3/27/12.
//  Copyright 2012. All rights reserved.
//

#import <UIKit/UIKit.h>
```

```
#import "HiJackMgr.h"

@class AlcoholViewController;

@interface    AlcoholAppDelegate    :    NSObject    <UIApplicationDelegate,
HiJackDelegate> {
    UIWindow *window;
    AlcoholViewController *viewController;
    HiJackMgr* hiJack;
}

@property (nonatomic, retain) IBOutlet UIWindow *window;
@property    (nonatomic,    retain)    IBOutlet    AlcoholViewController
*viewController;

@end
```

XV.2.2 AlcoholAppDelegate.mm

```
//
//  AlcoholAppDelegate.m
//  Alcohol
//
//  Created by Gurpal Bhoot on 3/27/12.
//  Copyright. All rights reserved.
//

#import "AlcoholAppDelegate.h"
#import "AlcoholViewController.h"

@implementation AlcoholAppDelegate

@synthesize window;
@synthesize viewController;


#pragma mark -
#pragma mark Application lifecycle

-(int) receive:(UInt8)data {
    double sensorValue=data/77.1;
    self.viewController.sensorValue = sensorValue;
    return 0;
}

-              (BOOL)application:(UIApplication              *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    hiJack = [[HiJackMgr alloc] init];
    [hiJack setDelegate:self];

    // Override point for customization after application launch.

    // Set the view controller as the window's root view controller and
display.
    self.window.rootViewController = self.viewController;
    [self.window makeKeyAndVisible];
```

```
    return YES;
}


- (void)applicationWillResignActive:(UIApplication *)application {
    /*
     Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
     Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates. Games should use this method to pause the game.
     */
}


- (void)applicationDidEnterBackground:(UIApplication *)application {
    /*
     Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
     If your application supports background execution, called instead of applicationWillTerminate: when the user quits.
     */
}


- (void)applicationWillEnterForeground:(UIApplication *)application {
    /*
     Called as part of  transition from the background to the inactive state: here you can undo many of the changes made on entering the background.
     */
}


- (void)applicationDidBecomeActive:(UIApplication *)application {
    /*
     Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.
     */
}


- (void)applicationWillTerminate:(UIApplication *)application {
    /*
     Called when the application is about to terminate.
     See also applicationDidEnterBackground:.
     */
}


#pragma mark -
#pragma mark Memory management
```

```
- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application {
    /*
     Free up as much memory as possible by purging cached data objects
that can be recreated (or reloaded from disk) later.
     */
}


- (void)dealloc {
    [hiJack release];
    [viewController release];
    [window release];
    [super dealloc];
}

@end
```

XV.2.3 AlcoholViewController.h

```
//
//  AlcoholViewController.h
//  Alcohol
//
//  Created by Gurpal Bhoot on 3/27/12.
//  Copyright. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface AlcoholViewController : UIViewController {

    IBOutlet UIButton *start;
    IBOutlet UIProgressView *sensorProgress;
    NSTimer *chargeSensor;
    NSTimer *blowTime;

    IBOutlet UILabel *testComplete;
    IBOutlet UILabel *instruction;
    IBOutlet UILabel *results;
    IBOutlet UITextField *resultsMessage;
    IBOutlet UITextField *resultsResponse;

    IBOutlet UITextField *leftBack;
    IBOutlet UITextField *rightBack;

    IBOutlet UIProgressView *bar1;
    IBOutlet UIProgressView *bar2;
    IBOutlet UIProgressView *bar3;
    IBOutlet UIProgressView *bar4;
    IBOutlet UIProgressView *bar5;
    IBOutlet UIProgressView *bar6;
    IBOutlet UIProgressView *bar7;
    IBOutlet UIProgressView *bar8;
    IBOutlet UIProgressView *bar9;
    IBOutlet UIProgressView *bar10;
    IBOutlet UIProgressView *bar11;
    IBOutlet UIProgressView *bar12;
```

```
    IBOutlet UIProgressView *bar13;
    IBOutlet UIProgressView *bar14;
    IBOutlet UIProgressView *bar15;


}

@property (nonatomic) double sensorValue;

- (IBAction)startSensor:(id)sender;
- (IBAction)doneEditing:(id)sender;
- (void)initiateSensor;
- (void)recordData;
- (void)displayBAC;


@end
```

XV.2.4 AlcoholViewController.h

```
//
//  AlcoholViewController.m
//  Alcohol
//
//  Created by Gurpal Bhoot on 3/27/12.
//  Copyright. All rights reserved.
//

#import "AlcoholViewController.h"

@implementation AlcoholViewController

@synthesize sensorValue;

- (IBAction)startSensor:(id)sender {

    // Initiate all of the Sensor Bars at 0.00%
    sensorProgress.progress = 0.0;
    bar1.progress = 0.0;
    bar2.progress = 0.0;
    bar3.progress = 0.0;
    bar4.progress = 0.0;
    bar5.progress = 0.0;
    bar6.progress = 0.0;
    bar7.progress = 0.0;
    bar8.progress = 0.0;
    bar9.progress = 0.0;
    bar10.progress = 0.0;
    bar11.progress = 0.0;
    bar12.progress = 0.0;
    bar13.progress = 0.0;
    bar14.progress = 0.0;
    bar15.progress = 0.0;

    // Initiate
    [start setHighlighted:TRUE];

    // Initiate all of the temporary text boxes to nil and colors to
```

```
clear
    results.text = nil;
    resultsMessage.text = nil;
    resultsMessage.backgroundColor = [UIColor clearColor];
    resultsMessage.alpha = 0.5;
    [resultsMessage setBorderStyle:UITextBorderStyleNone];
    testComplete.text = nil;
    resultsResponse.text = nil;
    [resultsResponse setBorderStyle:(UITextBorderStyleNone)];
    [resultsResponse setBackgroundColor:([UIColor clearColor])];

    //Initialize timer which will tell how long the sensor needs to be
heated up for
    chargeSensor    =    [NSTimer    scheduledTimerWithTimeInterval:0.00
target:self selector:@selector(initiateSensor) userInfo:nil repeats:YES];
}

- (void)initiateSensor {

    // Disenable the timer
    start.userInteractionEnabled = FALSE;

    // Give the first timer steps or progress
    sensorProgress.progress = sensorProgress.progress + .00008;

    // If the timer is finished start the next timer
    if (sensorProgress.progress == 1.0) {
        [chargeSensor invalidate];
        sensorProgress.progress = 0.0;
        instruction.text = [NSString stringWithFormat:@"Please blow into
the device for 3 seconds..."];

        blowTime    =    [NSTimer    scheduledTimerWithTimeInterval:0.00
target:self selector:@selector(recordData) userInfo:nil repeats:YES];
    }
}

- (void)recordData {

    // Disenable the second timer
    start.userInteractionEnabled = FALSE;

    // Give the timer progress steps
    sensorProgress.progress = sensorProgress.progress + .00007;

    // IF the timer has finished, relay the results from the sensor
    if (sensorProgress.progress == 1.0) {
        [blowTime invalidate];

        start.userInteractionEnabled = TRUE;
        [start setHighlighted:FALSE];


        instruction.text = [NSString stringWithFormat:@"Please allow for
the sensor to be ready..."];
        testComplete.text    =    [NSString    stringWithFormat:@"TEST
COMPLETE!"];
```

```
        [self displayBAC];
    }
}

- (void)displayBAC {

    if (sensorValue > 2.11) {

        results.textColor = [UIColor greenColor];
        results.text = [NSString stringWithFormat:@"0.00%%"];

        [resultsMessage setBorderStyle:UITextBorderStyleBezel];
        resultsMessage.backgroundColor = [UIColor blackColor];
        resultsMessage.textColor = [UIColor whiteColor];
        resultsMessage.text  =  [NSString  stringWithFormat:@"YOUR'E  WIDE
OPEN!"];

        bar1.progress = 1.0;
    }

    else if (sensorValue < 2.12 && sensorValue > 1.99) {

        results.textColor = [UIColor greenColor];
        results.text = [NSString stringWithFormat:@"0.01%%"];

        bar1.progress = 1.0;
        bar2.progress = 1.0;
        bar3.progress = 1.0;

    }

    else if (sensorValue < 2.00 && sensorValue > 1.92) {

        results.textColor = [UIColor yellowColor];
        results.text = [NSString stringWithFormat:@"0.02%%"];

        bar1.progress = 1.0;
        bar2.progress = 1.0;
        bar3.progress = 1.0;
        bar4.progress = 1.0;

    }

    else if (sensorValue < 1.93 && sensorValue > 1.84) {

        results.textColor = [UIColor yellowColor];
        results.text = [NSString stringWithFormat:@"0.03%%"];

        bar1.progress = 1.0;
        bar2.progress = 1.0;
        bar3.progress = 1.0;
        bar4.progress = 1.0;
        bar5.progress = 1.0;
        bar6.progress = 1.0;

    }
```

```
    else if (sensorValue < 1.85 && sensorValue > 1.77) {

        results.textColor = [UIColor yellowColor];
        results.text = [NSString stringWithFormat:@"0.04%%"];

        bar1.progress = 1.0;
        bar2.progress = 1.0;
        bar3.progress = 1.0;
        bar4.progress = 1.0;
        bar5.progress = 1.0;
        bar6.progress = 1.0;
        bar7.progress = 1.0;

    }

    else if (sensorValue < 1.78 && sensorValue > 1.72) {

        results.textColor = [UIColor orangeColor];
        results.text = [NSString stringWithFormat:@"0.05%%"];

        [resultsMessage setBorderStyle:UITextBorderStyleRoundedRect];
        resultsMessage.alpha = 1.0;
        resultsMessage.backgroundColor = [UIColor whiteColor];
        resultsMessage.textColor = [UIColor orangeColor];
        resultsMessage.text   =   [NSString   stringWithFormat:@"CALL   A
TIMEOUT!"];

        bar1.progress = 1.0;
        bar2.progress = 1.0;
        bar3.progress = 1.0;
        bar4.progress = 1.0;
        bar5.progress = 1.0;
        bar6.progress = 1.0;
        bar7.progress = 1.0;
        bar8.progress = 1.0;

    }

    else if (sensorValue < 1.73 && sensorValue > 1.62) {

        results.textColor = [UIColor orangeColor];
        results.text = [NSString stringWithFormat:@"0.06%%"];

        bar1.progress = 1.0;
        bar2.progress = 1.0;
        bar3.progress = 1.0;
        bar4.progress = 1.0;
        bar5.progress = 1.0;
        bar6.progress = 1.0;
        bar7.progress = 1.0;
        bar8.progress = 1.0;
        bar9.progress = 1.0;
        bar10.progress = 1.0;

    }

    else if (sensorValue < 1.63 && sensorValue > 1.54) {
```

```
        results.textColor = [UIColor orangeColor];
        results.text = [NSString stringWithFormat:@"0.07%%"];

        bar1.progress = 1.0;
        bar2.progress = 1.0;
        bar3.progress = 1.0;
        bar4.progress = 1.0;
        bar5.progress = 1.0;
        bar6.progress = 1.0;
        bar7.progress = 1.0;
        bar8.progress = 1.0;
        bar9.progress = 1.0;
        bar10.progress = 1.0;
        bar11.progress = 1.0;

    }

    else if (sensorValue < 1.53 && sensorValue > 1.47) {

        results.textColor = [UIColor redColor];
        results.text = [NSString stringWithFormat:@"0.08%%"];

        bar1.progress = 1.0;
        bar2.progress = 1.0;
        bar3.progress = 1.0;
        bar4.progress = 1.0;
        bar5.progress = 1.0;
        bar6.progress = 1.0;
        bar7.progress = 1.0;
        bar8.progress = 1.0;
        bar9.progress = 1.0;
        bar10.progress = 1.0;
        bar11.progress = 1.0;
        bar12.progress = 1.0;
        bar13.progress = 1.0;

    }

    else {

        results.textColor = [UIColor redColor];
        results.text = [NSString stringWithFormat:@"0.09%%"];

        resultsMessage.alpha = 1.0;
        [resultsMessage setBorderStyle:UITextBorderStyleRoundedRect];
        resultsMessage.textColor = [UIColor redColor];
        resultsMessage.backgroundColor = [UIColor whiteColor];
        resultsMessage.text    =    [NSString    stringWithFormat:@"CAUGHT
OFFSIDE!"];

        [resultsResponse setBorderStyle:UITextBorderStyleRoundedRect];
        [resultsResponse setBackgroundColor:([UIColor redColor])];
        resultsResponse.text   =   [NSString   stringWithFormat:@"HIT   THE
SHOWERS!!!"];


        bar1.progress = 1.0;
        bar2.progress = 1.0;
```

```
        bar3.progress = 1.0;
        bar4.progress = 1.0;
        bar5.progress = 1.0;
        bar6.progress = 1.0;
        bar7.progress = 1.0;
        bar8.progress = 1.0;
        bar9.progress = 1.0;
        bar10.progress = 1.0;
        bar11.progress = 1.0;
        bar12.progress = 1.0;
        bar13.progress = 1.0;
        bar14.progress = 1.0;
        bar15.progress = 1.0;

    }
}


- (IBAction)doneEditing:(id)sender {
    [sender resignFirstResponder];
}


/*
// The designated initializer. Override to perform setup that is required
before the view is loaded.
-    (id)initWithNibName:(NSString    *)nibNameOrNil    bundle:(NSBundle
*)nibBundleOrNil {
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
*/

/*
//  Implement  loadView  to  create  a  view  hierarchy  programmatically,
without using a nib.
- (void)loadView {
}
*/


// Implement viewDidLoad to do additional setup after loading the view,
typically from a nib.
- (void)viewDidLoad {
    [super viewDidLoad];
}


/*
//  Override  to  allow  orientations  other  than  the  default  portrait
orientation.
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)inte
rfaceOrientation {
    // Return YES for supported orientations
```

```
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
*/

- (void)didReceiveMemoryWarning {
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

- (void)viewDidUnload {
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (void)dealloc {
    [super dealloc];
}

@end
```