

AIR SHIELD

FINAL REPORT

EENG 4990: SENIOR DESIGN 2

3 CREDIT HOURS

FALL 2015

Team Members

Shabuktagin Photon Khan

Juan Pineda-Aguirre

Eric Nguyen

Tika Malla

Advisor: Professor Kamesh Namuduri

University of North Texas

Department of Electrical Engineering

Contents

Introduction.....	1
Report.....	1
Air Shield.....	1
Motor and the Fan.....	2
Flow Chart	3
Sensors	4
Gas Sensor (MQ-9).....	4
Dust Sensor (PPD42NS).....	5
Temperature/Humidity Sensor (DHT11).....	5
Modification.....	6
Why?.....	6
Rebuild the Circuit.....	6
Air Quality Index	7
Gas Sensor Datasheets	11
MQ-9 Calculation	13
Dust Sensor Datasheets.....	14
PPD42NS Calculation.....	15
Motor Datasheets	16
Motor Calculation	18
Transmitter/ Receiver.....	19
Distance Measurement.....	19
Antenna Calculation.....	22
Setup	23
Debugging.....	28
Conclusion	30
Problems and Improvement	30
Limitations	30
PCB.....	33
PSPICE	33
PID Controller.....	33

Cost	34
MERV Filter	34
Work Distribution	35
Project Schedule.....	35
Standards and Ethical Issues	36
Results.....	37
Arduino (USB Serial Monitor)	37
Data Acquisition	39
LCD Screen Display	41
References.....	44
Arduino Codes	45
First Micro-controller.....	45
Second Micro-controller	50
MATLAB codes.....	52
Individual Plot.....	52
Polluted Air in Room Estimation.....	52
Index	54

Shabuktagin Photon Khan (DO NOT COPY)

Introduction

Air Shield project is designed to purify the air in order to help people who suffer from respiratory problems such as asthma and allergies. In addition, the Air shield is designed to provide protection for a small indoor area in case of fire or gas leak. The air shield will be used indoors, primarily for personal use to ensure clean air for sensitive people. It would be mostly beneficial to children who have the highest rates of respiratory problems as well as been very sensitive to pollutants such as pollens, dust and second hand smoking. The idea is to absorb gases, smoke and dust particles using an air suction fan, the purifying the air using MERV filters. This project was first started to be a part of a Quad-Copter enhancement. As time progressed, we decided this project to be a stand-alone project. Therefore, the Air shield will be used as a public health safety system. This is a two semester long project, on our final semester we decided to reduce the number of micro-controllers, make it user friendly, make the code open source, decided to work on MERV filters, follow the standards, write proper algorithm for Arduino, rebuild the circuit and reduce the working power

Report

Air Shield

It consist of a motor which pressurize the air through a filtration system. It is an automated air purifier which can clean or shield a small area. It can be used indoors and for medical purposes. Initially our first design was taken from “Air Umbrella” concept, however, due to some specification and requirements we decided to do it from scratch with a complete new look.

Figure 1, shows the old design of the Air Shield and the 3D model was done using Tinker-CAD.

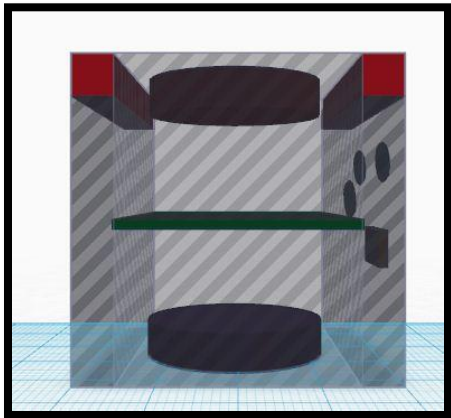


Figure 1: 3D Model of Air Shield (Old)

Figure 2, shows the changes that have been made for the new Air Shield Design. The top surface of the Air Shield would be empty and the four sides of the box would be covered with MERV filters. The motor in the middle would spread the air on four sides.

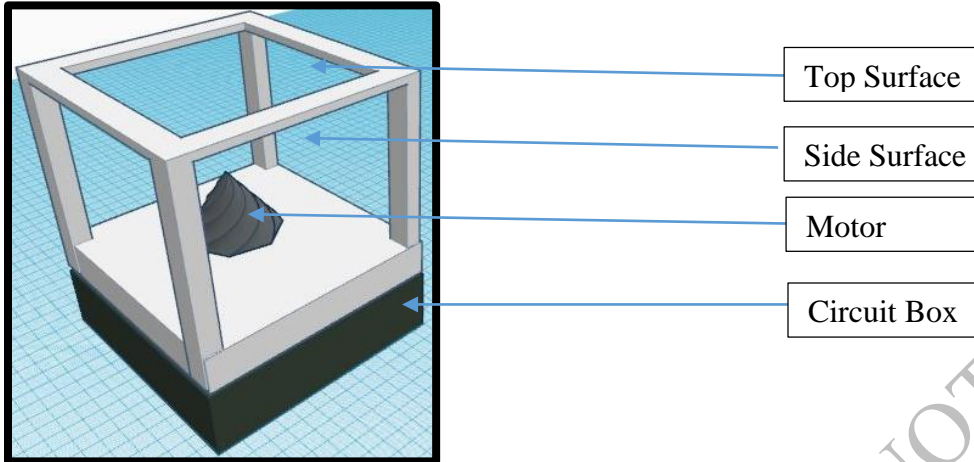


Figure 2: 3D Model of Air Shield (New)

Motor and the Fan

For the purpose of our Air Shield we used Centrifugal Fan. While doing the research for the fans, we found a Centrifugal Fan design that seemed to be the best compatible fan for our project as it fits our criteria for the efficiency range that we need to for the air shield. After finalizing the design for that centrifugal fan that we will use, we took the permission from the person who made the design of this particular centrifugal fan that we are using.

In centrifugal flow the direction of the airflow changes twice. Once, when it is entering and the second time when it is leaving. The airflow leaving is then later categorized according to the blade configuration which is forward curved, backward curved and radial. Each type has its own application range and limits.

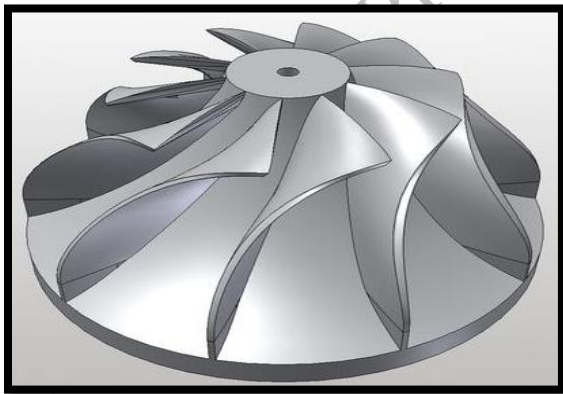


Figure 3: Auto-Cad version of the Centrifugal Fan

Flow Chart

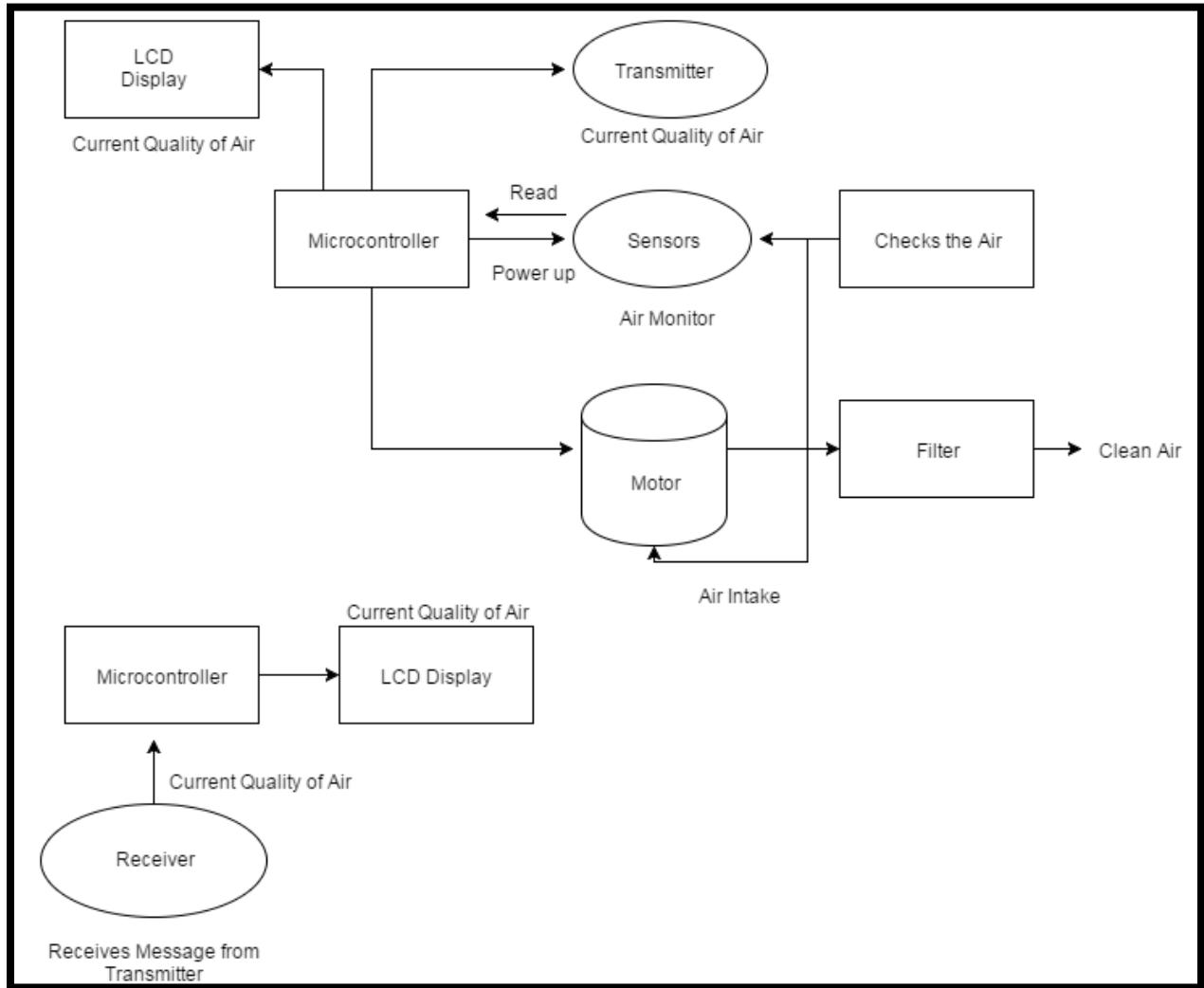


Figure 4: Air Shield Flow Chart

Shabuktajin

Sensors

In order to make the Air Shield work automatically we decided to use multiple sensors to control the functionality of the Air Shield based on different factors in the environment. For this we are using three different sensors which are Gas sensor, Dust Sensor and Temperature/Humidity Sensor. Our first response sensors are the gas sensor and the dust sensor, taking into consideration that our project aims to protect people from PPM concentrations of gas and dust particles. The Air Shield have DHT11 sensor in order to be able to detect temperature and humidity in the environment for a better usage.

Gas Sensor (MQ-9)

It is a gas sensor that detects Carbon monoxide, Coal Gas and Combustible Gas. It uses a process known as oxygen vacancies model, “In the oxygen-vacancy model, reducing gases, such as CO, react with oxygen in the surface of the metal oxide to produce gaseous CO₂. The oxygen vacancy left behind then ionizes and produces an electron in the conduction band of the metal oxide that increases the electrical conductivity”(2). The metal oxide been used in the gas sensor is Tin Oxide (SnO₂), which generates the electrical conductivity as it reacts with Carbon Monoxide. This electrical conduction generates an output voltage that is proportional to the Carbon Monoxide concentration, the higher the concentration of gas, the higher the output voltage. After the chemical reaction occurs, a reverse reaction then occurs in which particle of oxygen in the air, fills up the oxygen vacancies of the gas sensor, decreasing the conduction of electricity.



Figure 5: MQ-9 Gas Sensor

Dust Sensor (PPD42NS)

The PPD42NS is designed to detect particles of dust of the size bigger than one micrometer in which particles of house dust, pollen and cigarette smoke could be detected. The sensor consists on a black box that holds a phototransistor and an emitter diode arranged diagonally from each other. The emitter diode emits constant light and the phototransistor takes the light to conduct electricity. In case of dust particles inside the cube box, the light coming from the emitter diode gets blocked and the phototransistor decreases its conductivity. Such changes of conduction is taken as a signal, where the higher the concentration of dust the lower the conduction of electricity. In addition, the sensor has a heater (resistor) to generate heat and create an updraft current of air, drawing outside air into the cube. The reason behind it is to constantly bring air to test for dust conditions.

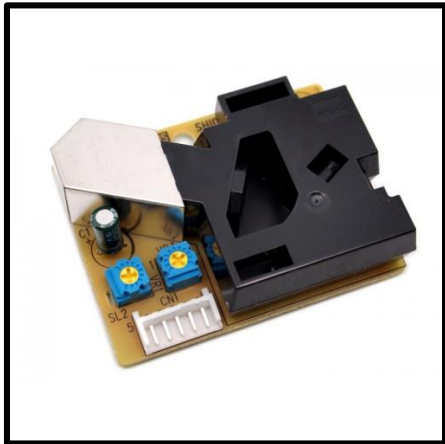


Figure 6: PPD42NS Dust Sensor

Temperature/Humidity Sensor (DHT11)

DHT11 digital temperature and humidity sensor contains a calibrated digital signal output of the temperature and humidity. The sensor includes a resistive sense of wet components and an NTC temperature measurement devices which is connected with a high-performance 8-bit microcontroller.



Figure 7: DHT11 Temperature/Humidity Sensor

Modification

Why?

Our first aim in this semester was to reduce the number of micro-controllers. In the beginning, we were using four microcontroller two of them were Texas TI Launch Pad and other two were Arduino. Two Launch Pads and one Arduino were used with the filtrations system and the other Arduino was used for receiving data from the filtration system. Previously, we were using I^2C to transfer data from one micro-controller to another and that process was quiet slow. The main reason behind the modification were the power consumption and each of the microcontroller at least requires 9v to function properly. For better efficiency of the Air Shield we as a team decided to go with two micro-controller. The micro-controller we are using right now is Arduino only. One micro-controller is used directly with the filtration system and other is used for the receiver. This would reduce the overall cost of the system.

Rebuild the Circuit

Figure 8, it shows the circuit that was built before. It was messy and it was difficult to debug and find out where in the circuit is having problem. Therefore, the circuit was rebuild from the scratch. **Figure 9**, shows the new circuit that was made. Also, each breadboard was used as a section of the Air Shield Circuit. This means, the first breadboard has all the sensors with a voltage regulator. The second breadboard consist of the microcontroller. The third one consist of help pushbutton, 595 shift register and a transmitter. The fourth one consists of LCD display and a transistor which regulates the speed of the motor

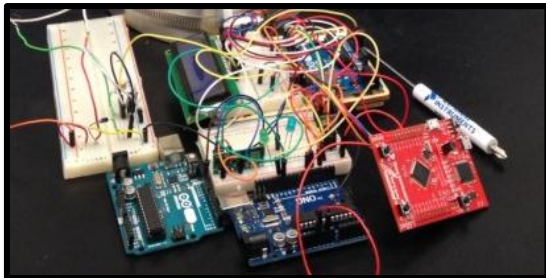


Figure 8: Old Circuit of the Air Shield

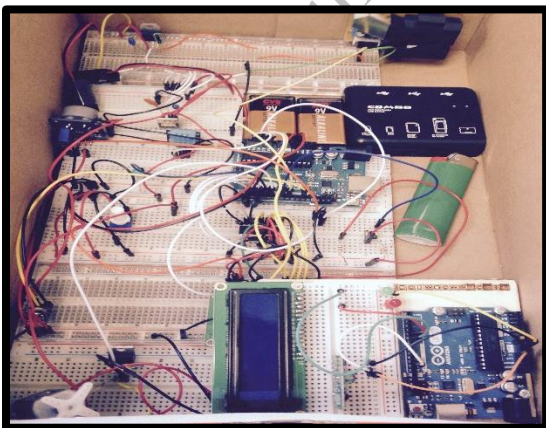


Figure 9: New Circuit of the Air Shield

Air Quality Index

Air Quality Index (AQI) calculates the five major air pollutants regulated by the Clean Air Act: ground-level ozone (O₃), particle pollution or particulate matter (PM_{2.5} and PM₁₀), carbon monoxide (CO), sulfur dioxide (SO₂), and nitrogen dioxide (NO₂). AQI gives us information by telling how clean or polluted our indoor air is, along with health concerns. AQI transforms air quality data into number to help the people understand when to take action to protect their health.

Figure 10 demonstrates the AQI number with respect to health conditions.

Air Quality Index (AQI)	
Good	0
Moderate	51
Unhealthy for Sensitive people	101
Unhealthy	151
Very Unhealthy	201
Hazardous	301

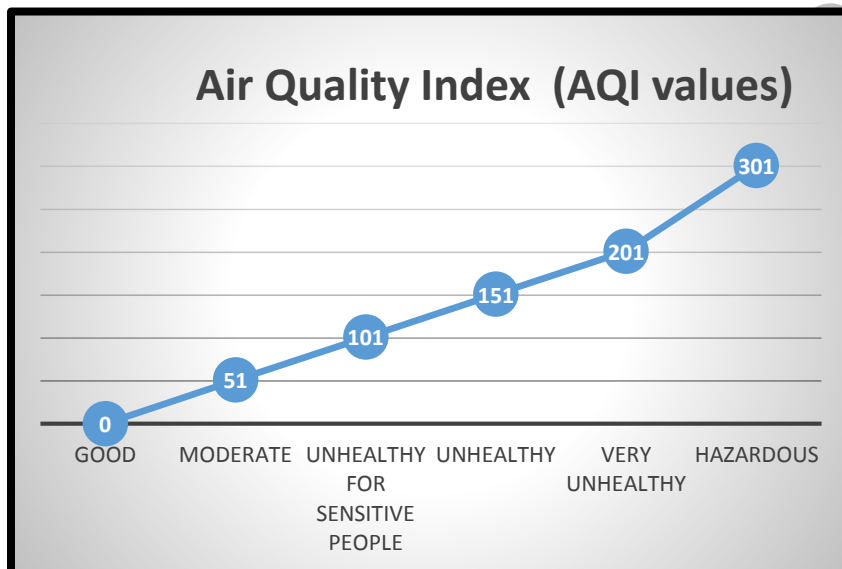


Figure 10: Air Quality Index

We took data from the air quality government website to compare concentration with respect to AQI. The following graphs will show the relations.

For Carbon Monoxide

CO (PPM)	AQI
0	0
5	56
10	109
15	193
20	231
25	264
30	297
35	346
40	396
45	445
50	496

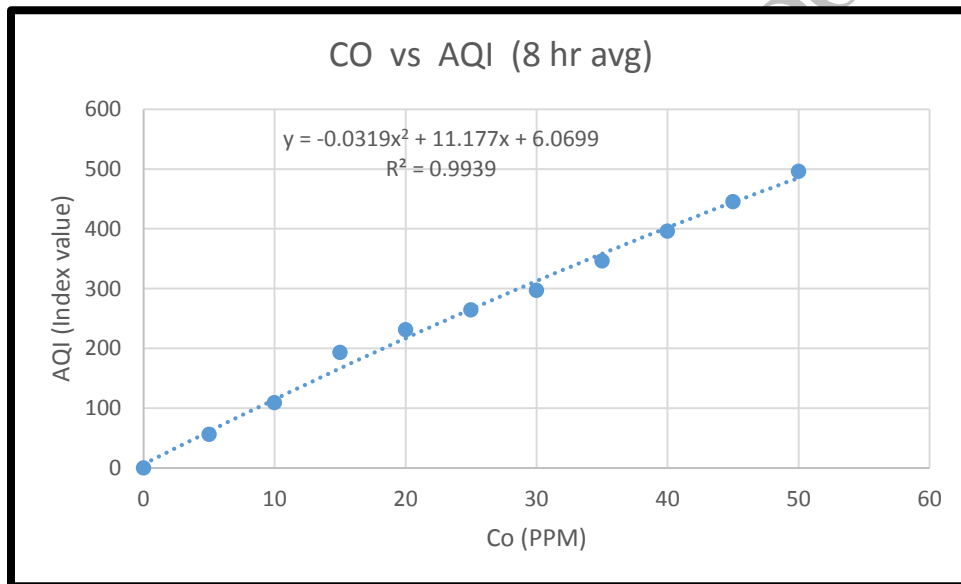


Figure 11: CO (PPM) vs AQI

For Dust (PPM 2.5)

PPM_2.5 (ug/m3)	AQI
0	0
50	137
100	174
150	200
200	250
250	300
300	350
350	400
400	434
450	467
500	500

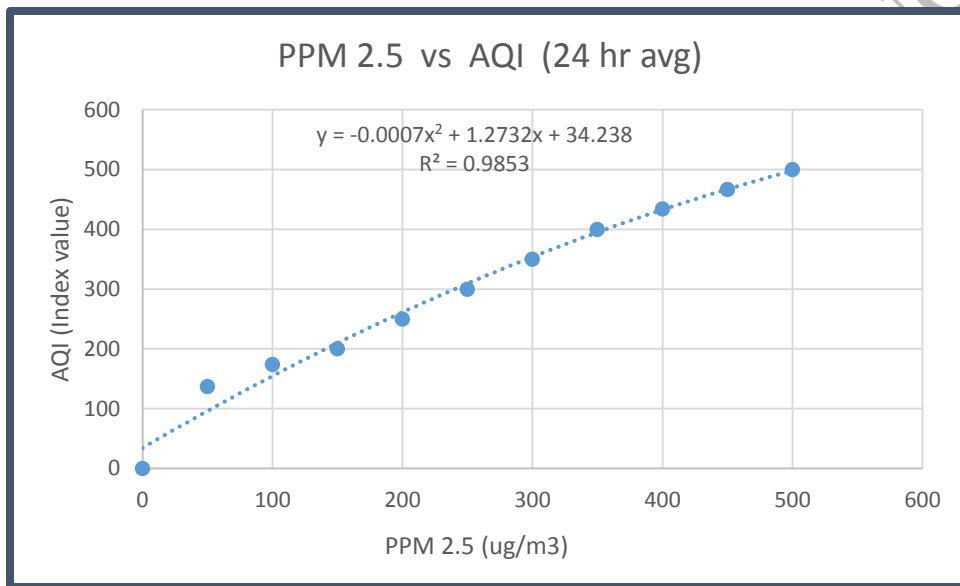
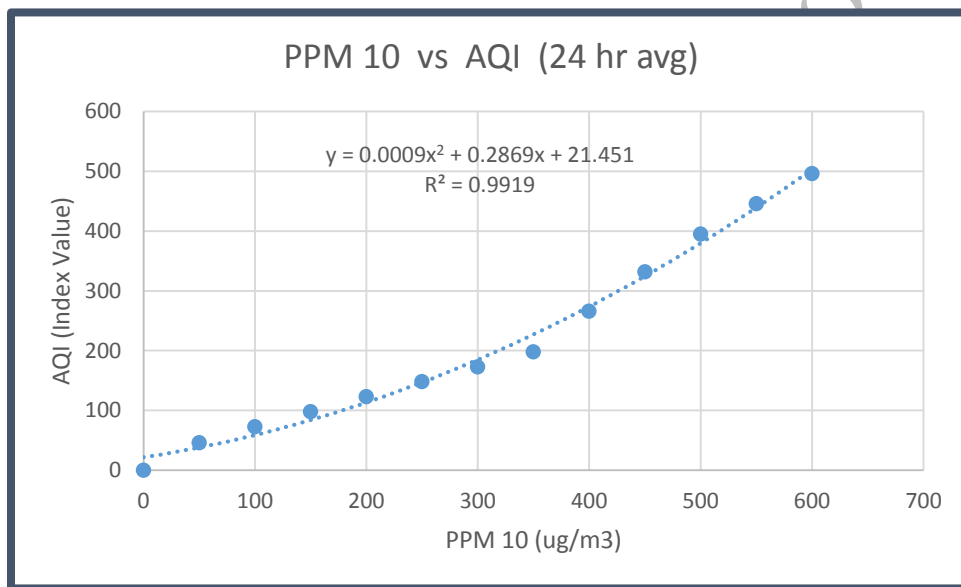


Figure 12: Dust (PPM 2.5) vs AQI

For Dust (PPM 10)

PM_10 (ug/m3)	AQI
0	0
50	46
100	73
150	98
200	123
250	148
300	173
350	198
400	266
450	332
500	395
550	446
600	496

**Figure 13: Dust (PPM 10) vs AQI**

Gas Sensor Datasheets

The sensitivity graph in the MQ-9 sensor's datasheet is shown below which was given with the product.

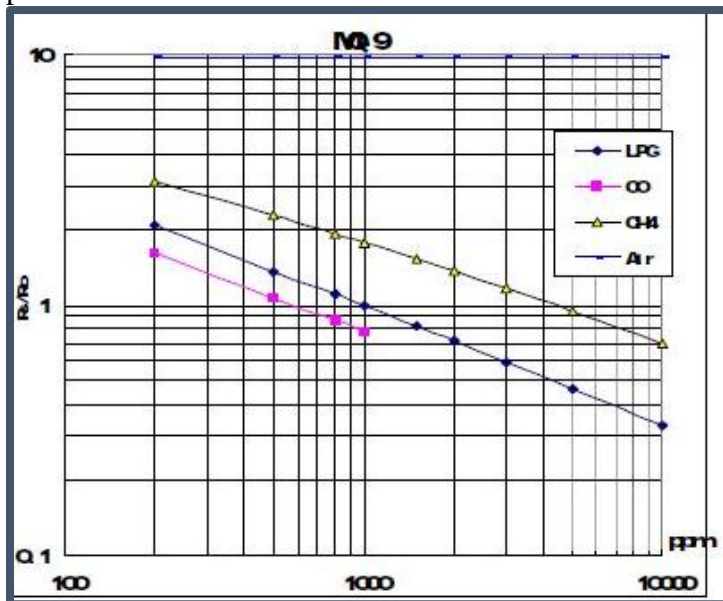


Figure 14: R_s/R_L vs PPM

Operation Principle

The surface resistance is R_s . It is obtained through effected voltage signal output of the load resistance R_L with series-wound. The relationship between them is given below:

$$V_{RL} = \frac{\text{arduinovalue}}{1024} * 5$$

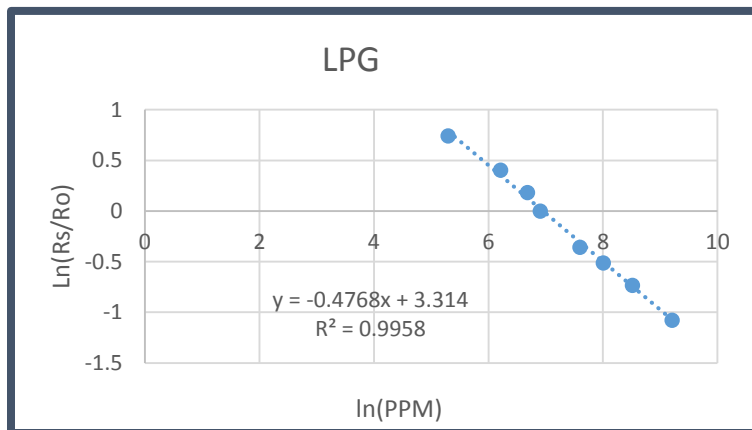
$$\frac{R_s}{R_L} = \frac{V_c - V_{RL}}{V_{RL}}$$

$$R_o = R_L$$

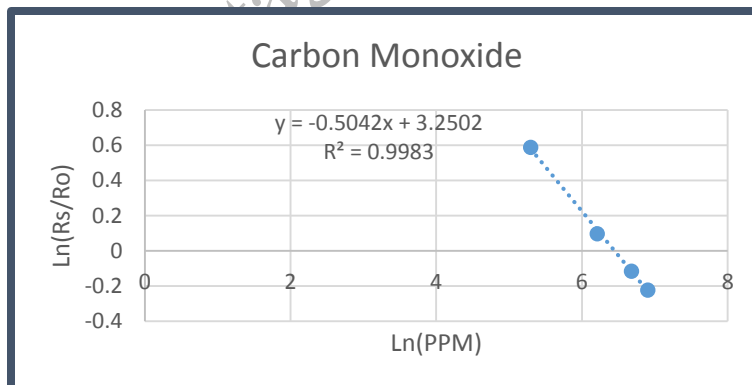
From datasheet, the PPM was transferred to log x-axis and $\frac{R_s}{R_L}$ was changed to log y-axis using Microsoft Excel. Since, the datasheet was in log scale. It helped us to give an equation with respect to PPM. Therefore, from microcontroller we will get the value for V_{RL} and that value is used to find out the value for $\frac{R_s}{R_L}$. At the end, we will use the $\frac{R_s}{R_L}$ value to find out PPM. Number of equations were used to calculate the value the PPM. The Microsoft Excel work is shown in next page. R_L is considered to be 20,000 ohms. We also assumed that the temperature and the humidity is constant.

For Liquefied Petroleum Gas

PPM	Rs/Ro	ln(PPM)	ln(Rs/Ro)
200	2.1	5.298317	0.741937
500	1.5	6.214608	0.405465
800	1.2	6.684612	0.182322
1000	1	6.907755	0
2000	0.7	7.600902	-0.35667
3000	0.6	8.006368	-0.51083
5000	0.48	8.517193	-0.73397
10000	0.34	9.21034	-1.07881

**Figure 15:** Relationship graph for LPG**For Carbon Monoxide**

PPM	Rs/Ro	ln(PPM)	ln(RS/R0)
200	1.8	5.298317	0.587787
500	1.1	6.214608	0.09531
800	0.89	6.684612	-0.11653
1000	0.8	6.907755	-0.22314

**Figure 16:** Relationship graph for CO

For Methane

PPM	Rs/Ro	ln(PPM)	ln(Rs/Ro)
200	3.1	5.298317	1.131402
500	2.4	6.214608	0.875469
800	1.9	6.684612	0.641854
1000	1.8	6.907755	0.587787
2000	1.5	7.600902	0.405465
3000	1.3	8.006368	0.262364
5000	0.95	8.517193	-0.05129
10000	0.7	9.21034	-0.35667

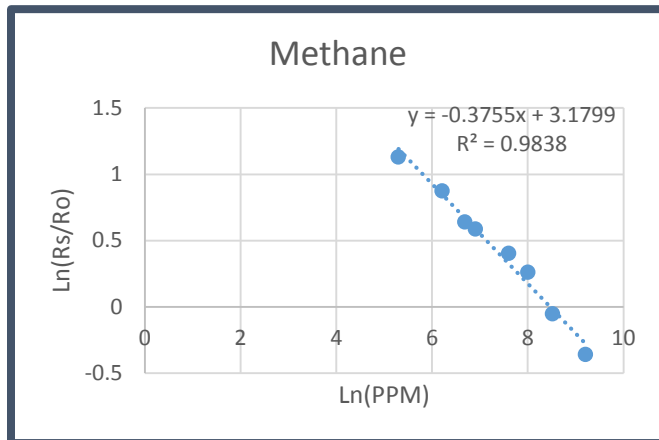


Figure 17: Relationship graph for Methane

MQ-9 Calculation**For Liquefied Petroleum Gas**

$y = -0.4768x + 3.314$, derived from Figure 14

$$\ln\left(\frac{R_s}{R_L}\right) = (-0.4768 * \ln(PPM)) + 3.314$$

$$\ln(PPM) = (3.314 - \frac{R_s}{R_L}) / 0.4768 = \text{value}; PPM = e^{\text{value}}$$

For Carbon Monoxide

$y = -0.5042x + 3.2502$, derived from Figure 15

$$\ln\left(\frac{R_s}{R_L}\right) = (-0.5042 * \ln(PPM)) + 3.2502$$

$$\ln(PPM) = (3.2502 - \frac{R_s}{R_L}) / 0.5042 = \text{value}; PPM = e^{\text{value}}$$

For Methane

$y = -0.3755x + 3.1799$, derived from Figure 16

$$\ln\left(\frac{R_s}{R_L}\right) = (-0.3755 * \ln(PPM)) + 3.1799$$

$$\ln(PPM) = (3.1799 - \frac{R_s}{R_L}) / 0.3755 = \text{value}; PPM = e^{\text{value}}$$

Dust Sensor Datasheets

The sensitivity graph in the PPD42NS sensor's datasheet is shown below which was given with the product.

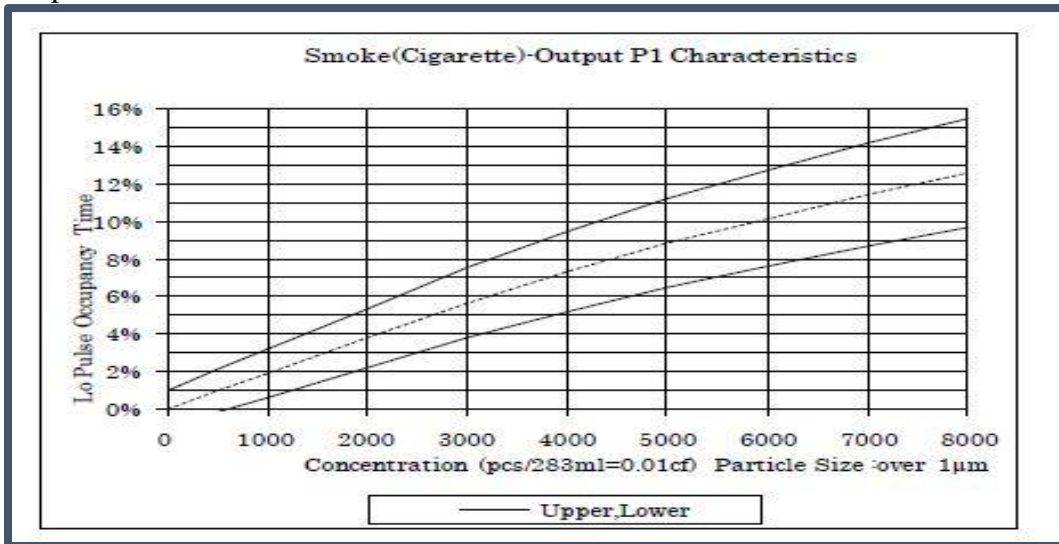


Figure 18: Concentration vs Low Pulse Occupancy Time

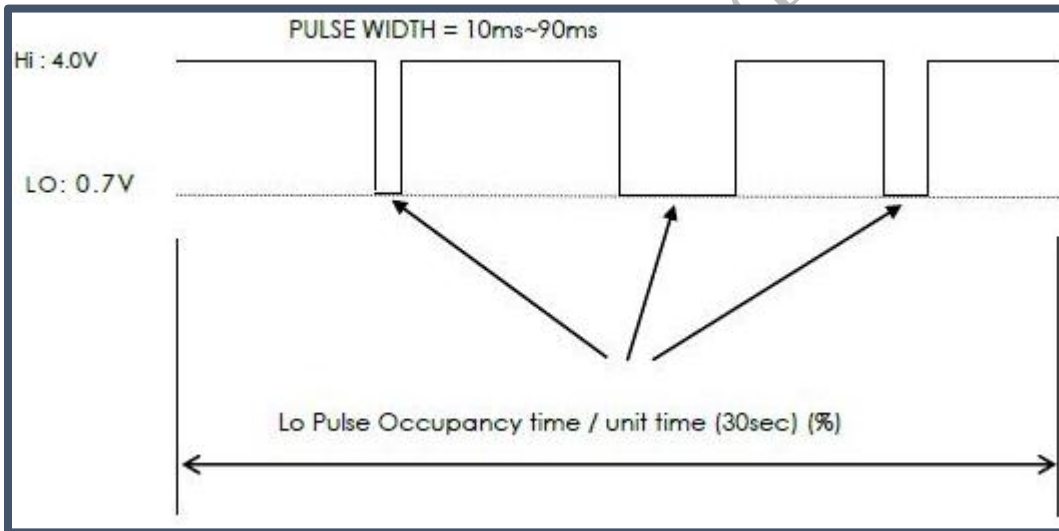


Figure 19: Measuring LPOT

We used the Arduino to calculate how long the pulse was low during the presence of dust particle. Therefore,

$$LPOT = \frac{lowpulseoccupancy}{sampletime_{ms}}$$

We took the data points from Figure 17 and changed the axis for Concentration and Low Pulse Occupancy Time (LPOT). Therefore, using the Microsoft Excel we changed the LPOT axis to x-axis and changed the concentration axis to y-axis. This helped us to get an equation with respect to concentration. Since, LPOT was found out using microcontroller. The Microsoft Excel work is shown next page.

LPOT	Concentration
0	0
2	1000
3.8	2000
5.6	3000
7.2	4000
9	5000
10.2	6000
11.5	7000
12.7	8000

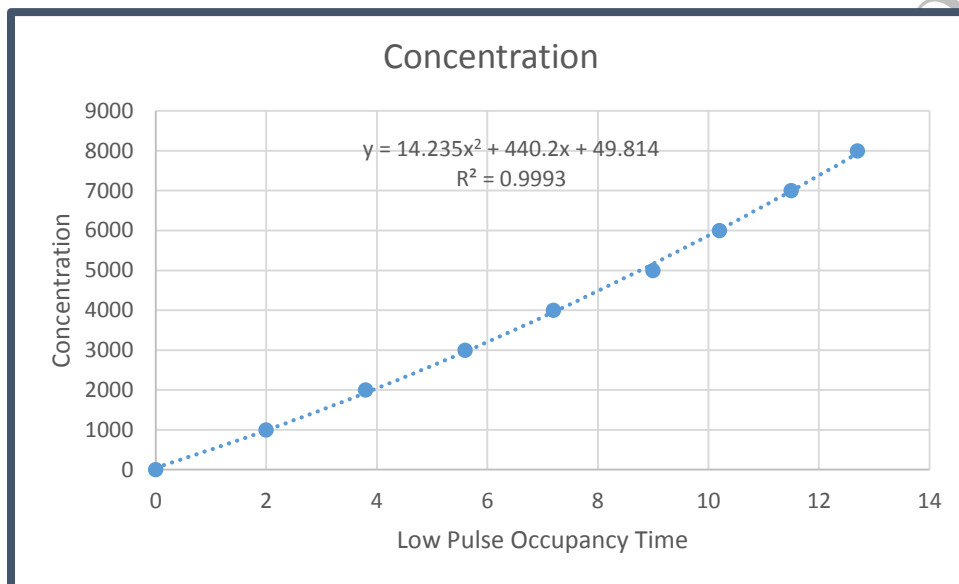


Figure 20: LPOT vs Concentration

PPD42NS Calculation

$y = 14.235x^2 + 440.2x + 49.814$, this equation is obtained from Figure 18.

$$\text{Concentration} = 14.235(LPOT^2) + 440.2(LPOT) + 49.814$$

However, we are not going to use this equation. Instead we would be using an equation which is insisted by experts and gives a similar graph that we acquired.

$$LPOT = \frac{\text{lowpulseoccupancy}}{\text{sampletime}_{ms}}$$

$$\text{Concentration} = 1.1 * LPOT^3 - 3.8 * LPOT^2 + 520 * LPOT + 0.62$$

Motor Datasheets

We would be using 9V dc motor but for the testing purpose we used the 5v motor dc motor. All the data below are for 5v motor dc motor.

5v dc motor gives around 9000 rpm.

We used analog discovery to see the pulse width modulation from Arduino to the motor. The graphs are given below. We also compared the value that was given from Arduino. The Arduino usually write value from (0 to 255) which is the same representation as the (0V to 5V)

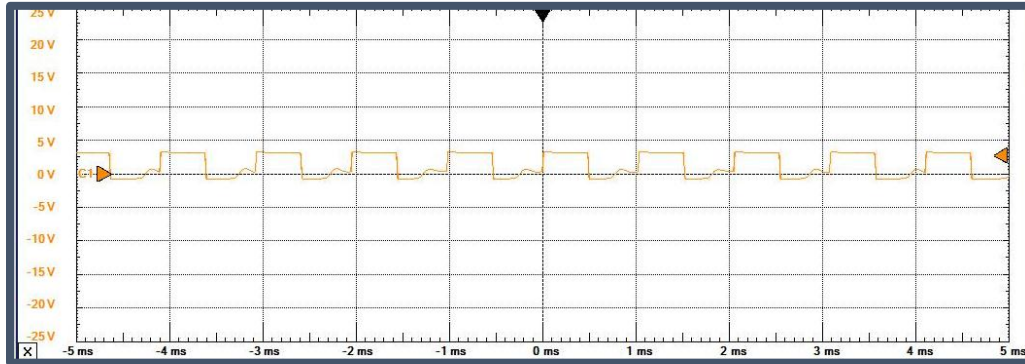


Figure 21: Pulse width Modulation for 127

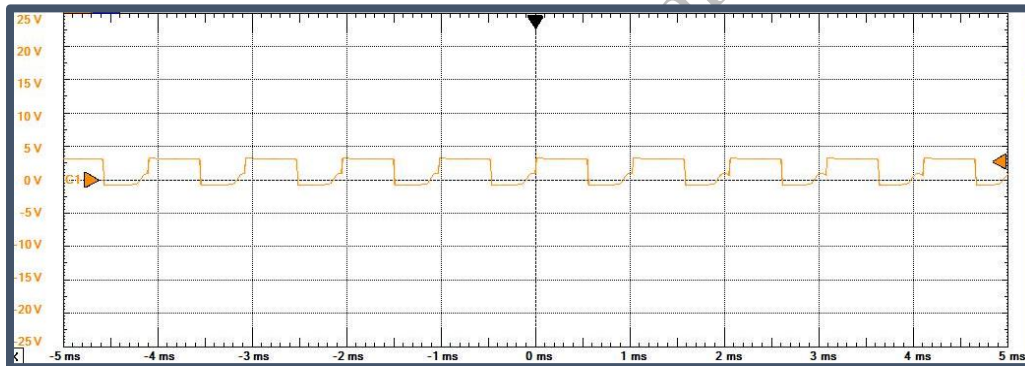


Figure 22: Pulse width Modulation for 137

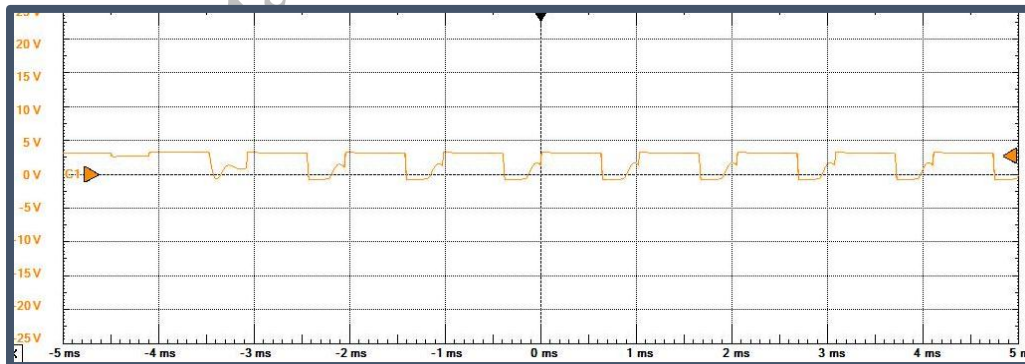


Figure 23: Pulse width Modulation for 157

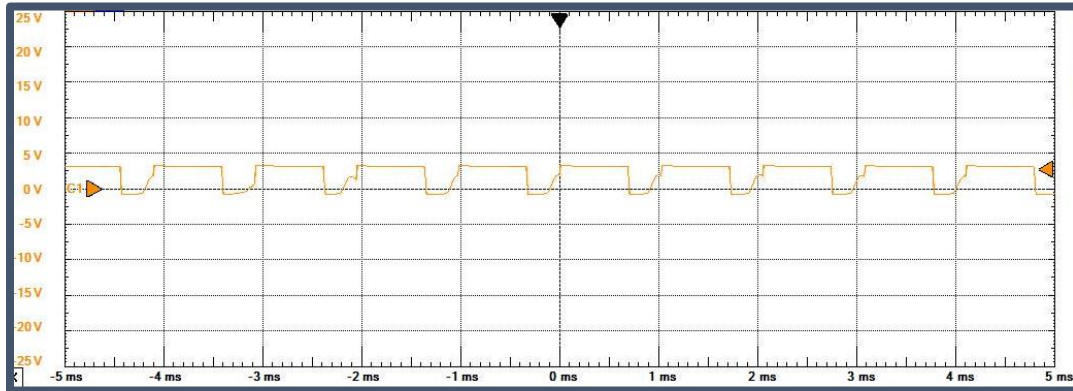


Figure 24: Pulse width Modulation for 171

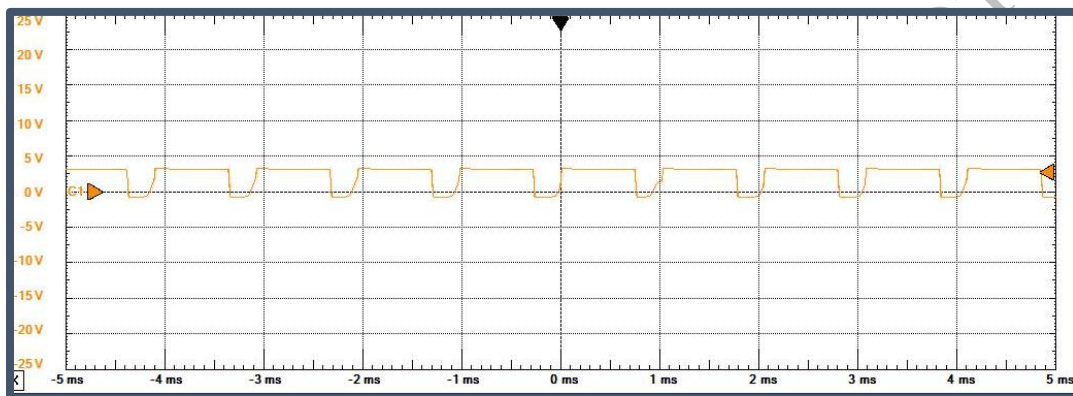


Figure 25: Pulse width Modulation for 186

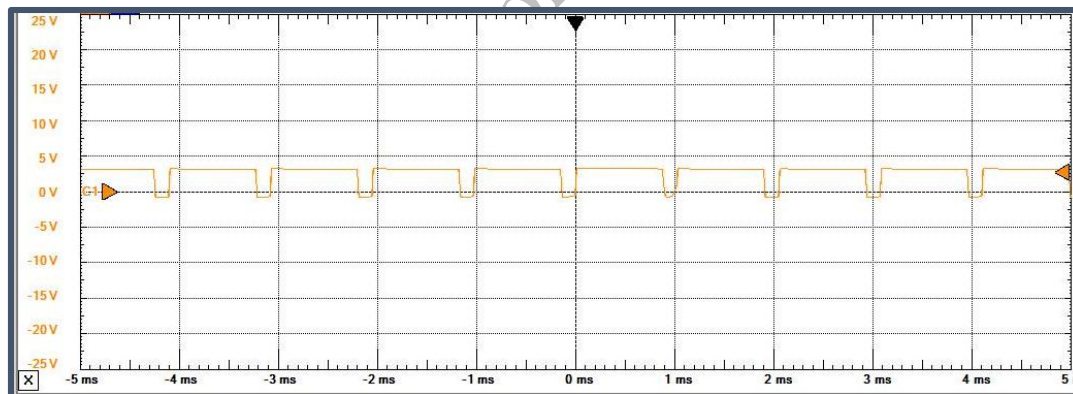


Figure 26: Pulse width Modulation for 215

From Figure (21 to 26), there is a trend. As the value of Arduino increase the width of low pulse decreases in a given duty cycle. Therefore, we can calculate the PWM from this figures and compare it with value obtained from Arduino. Then PWM can be used to find the voltage given to motor. That voltage will be used to find out the RPM of the motor.

Motor Calculation

From datasheet,

$$PWM = \frac{\text{High pulse}}{\text{High Pulse} + \text{Low Pulse}} \quad (\text{at a given duty cycle which is } 1\text{ms})$$

$$\text{Voltage} = PWM * 3.26 \quad (\text{The value } 3.26 \text{ is being provided by the microcontroller})$$

$$RPM = k * (\text{Voltage})$$

If RPM = 9000 at max volt 5

$$k = 1800;$$

Arduino	PWM	Volt (V)	RPM (rad/s)
		1	1800
120	0.5	1.63	2934
137	0.55	1.793	3227.4
157	0.65	2.119	3814.2
171	0.7	2.282	4107.6
186	0.775	2.5265	4547.7
217	0.9	2.934	5281.2
255	1	3.26	5868

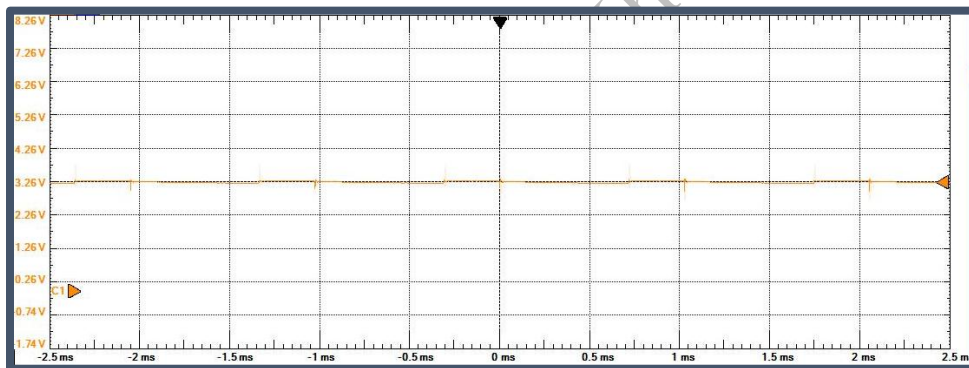


Figure 27: Voltage given to Motor

Transmitter/ Receiver

From the datasheet, the relation between baud rate and the distance was found out.

Baud Rate (Bps)	Range
9000	0
7000	90
5000	100
2000	150
1000	200

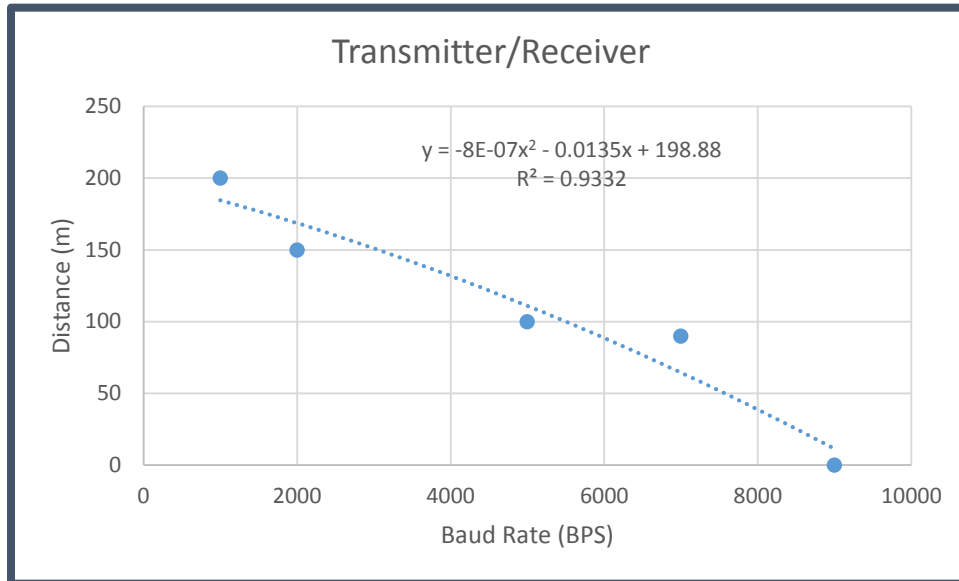


Figure 28: Baud Rate vs Distance

Distance Measurement

For the transmitter and receiver, the signal strength was difficult to perform since it is a one way communication. The transmitter uses amplitude shift keying to transmit the messages. Therefore, to find out the maximum viable distance from transmitter to receiver, we made an algorithm that will make the led (which is attached to the receiver) to blink whenever it receives the full message. We counted the maximum number of blinks for the led per minute for different distances. The data are given next page. It is seen that the receiver can receive maximum of 18 messages per minute (therefore, it receives messages after every 3.33 seconds)

For 3.26 V

Blinks/min	Distance (cm)
15	20
15	50
15	70
10	90
3	120

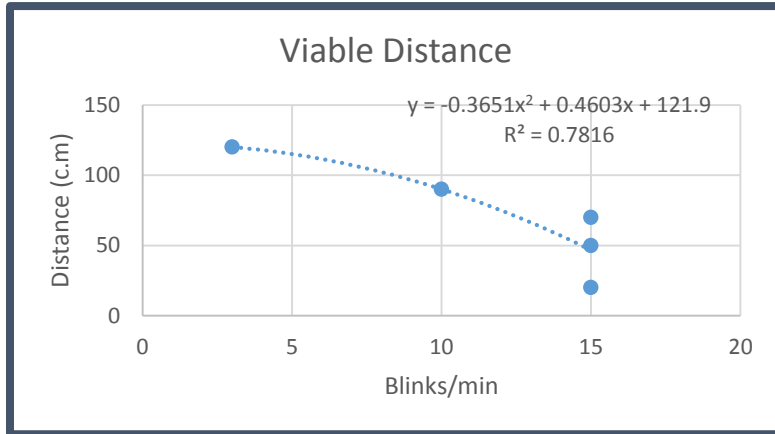


Figure 29: Blinks/min vs Distance for 3.26V

For 5V

Blinks/min	Distance
18	20
16	50
16	70
13	90
11	120

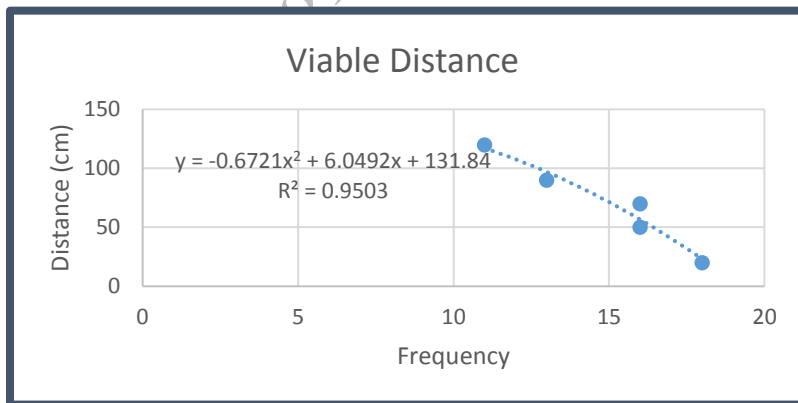


Figure 30: Blinks/min vs Distance for 5V

For 9V

Blinks/min	Distance
18	20
18	50
18	70
18	90
18	120
0	190

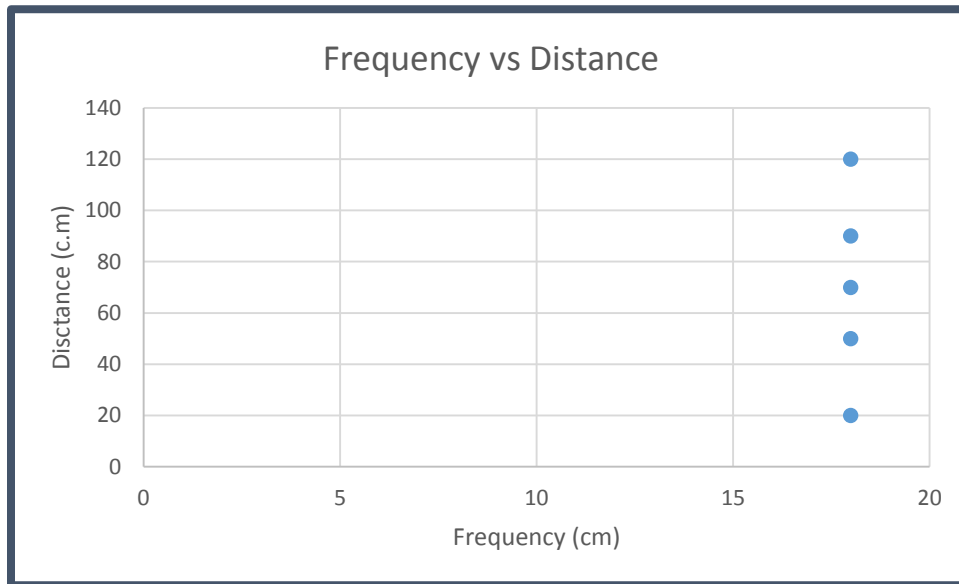


Figure 31: Blinks/min vs Distance for 9V

From **Figure 29**, we got an equation, $y = -0.3651x^2 + 0.4603x + 121.9$

So we can assume that when $x = \text{blinks/min} = 0$, $y = \text{distance} = 121.9 \text{ cm}$

From **Figure 30**, we got an equation, $y = -0.6721x^2 + 6.0492x + 131.84$

So we can assume that when $x = \text{blinks/min} = 0$, $y = \text{distance} = 131.84 \text{ cm}$

From **Figure 31**, we can assume that when $x = \text{blinks/min} = 0$, $y = \text{distance} = 190 \text{ cm}$

Therefore it leads to final graph which is the relationship between voltages and distance (transmitter and receiver).

Voltage	Distance (cm)
3.26	121.9
5	131.8
9	190

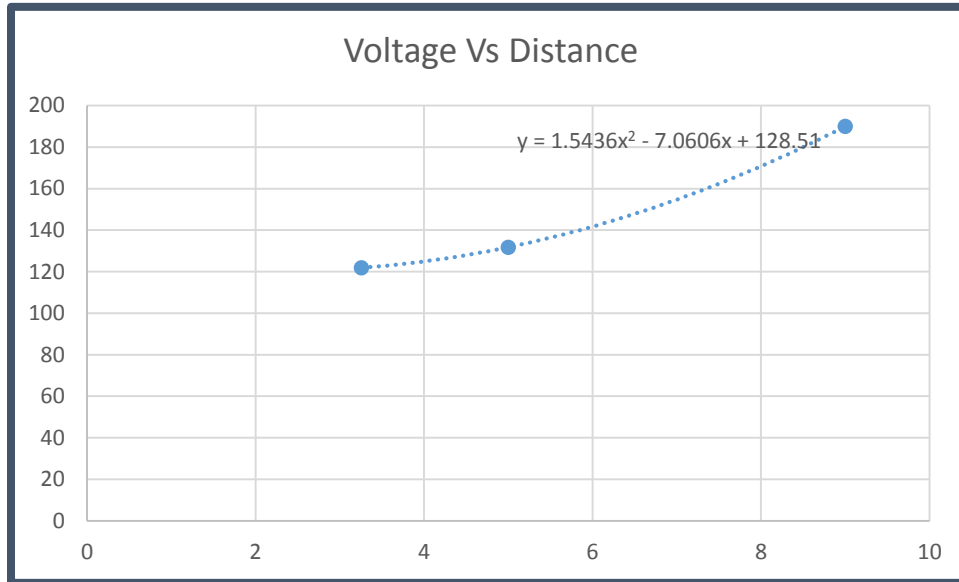


Figure 32: Voltages vs Distance

From Figure 30, we get $y = 1.5436x^2 - 7.0606x + 128.51$

From datasheet we know that transmitter's working range is from (3V to 12V). There for when $x = \text{voltage} = 12$, $y = \text{distance} = 226\text{cm}$

The working distance from transmitter and receiver can be increased using antenna.

Antenna Calculation

The antenna needs to be monopole, Omni-directional.

We know,

$F = 433\text{MHz}$, it is the transmitter's frequency

$c = f\lambda$; $c = \text{speed of the light}$

$$\lambda = \frac{3.10^8}{433\text{MHz}}; \lambda = 0.693\text{m}$$

Since, 0.693m is very long, quarter wavelength is used for the antenna

$$\frac{\lambda}{4} = 17.32\text{ cm}$$

Setup

Transmitter and Receiver

The setup section of the Transmitter and Receiver is given below

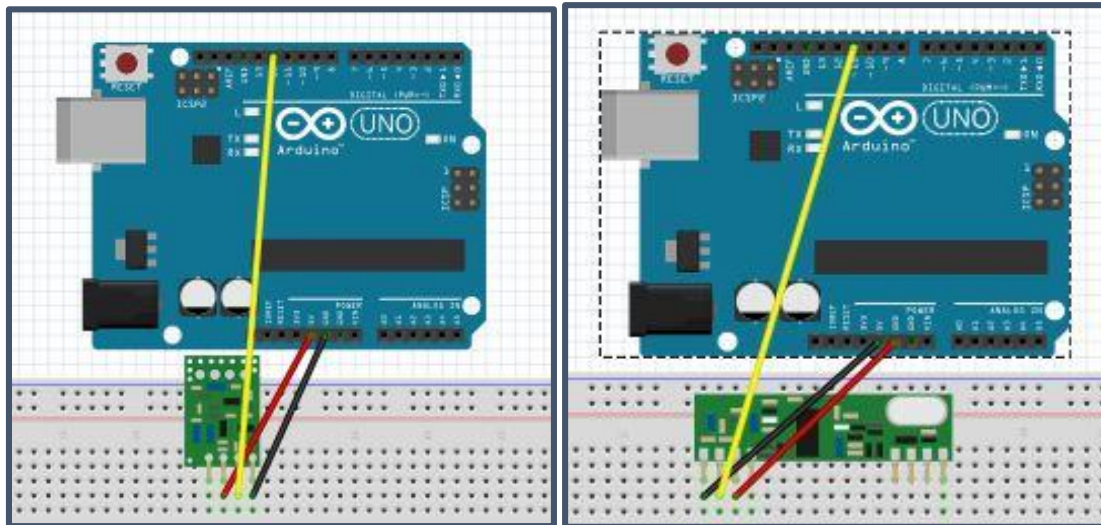


Figure 33: Transmitter (left) and Receiver (right)

LCD screen and Display

The old setup for the LCD screen display is shown below

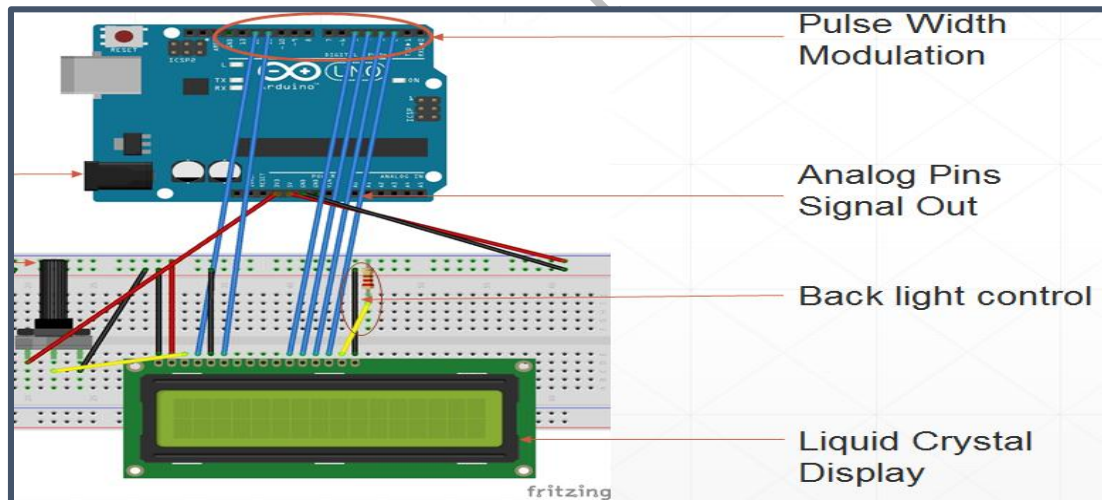


Figure 34: Old Setup of LCD screen

To reduce the number of pins in micro-controller we used a shift register. It reduces the pins required for the LCD in microcontroller from 6 to 3. When the sensors detect pollutants e.g. Carbon Monoxide and Combustible gas, Arduino sends “HIGH” output to Warning LED (Blue). Liquid Crystal Display shows the different kinds of warning for different pollutants. When the area is clean, it will show the current temperature of the room.

Temperature/ Humidity Sensor

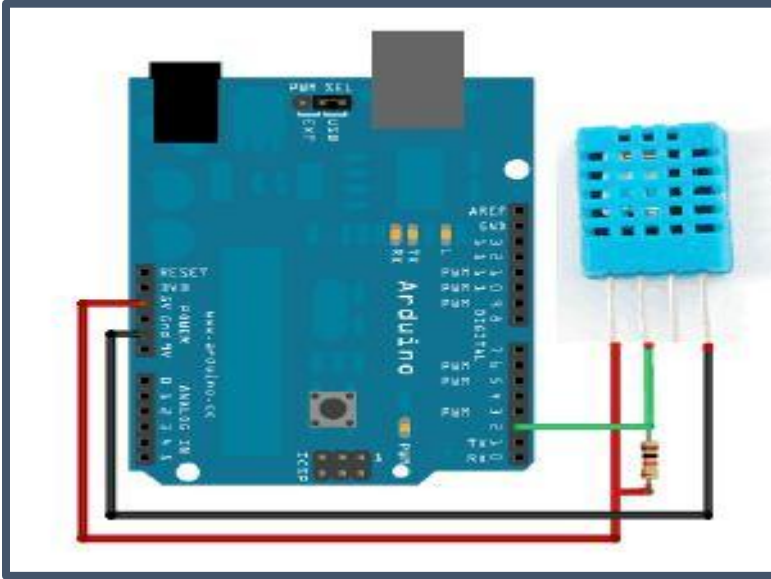


Figure 35: Setup for Temperature/Humidity Sensor

Gas Sensor/Dust Sensor

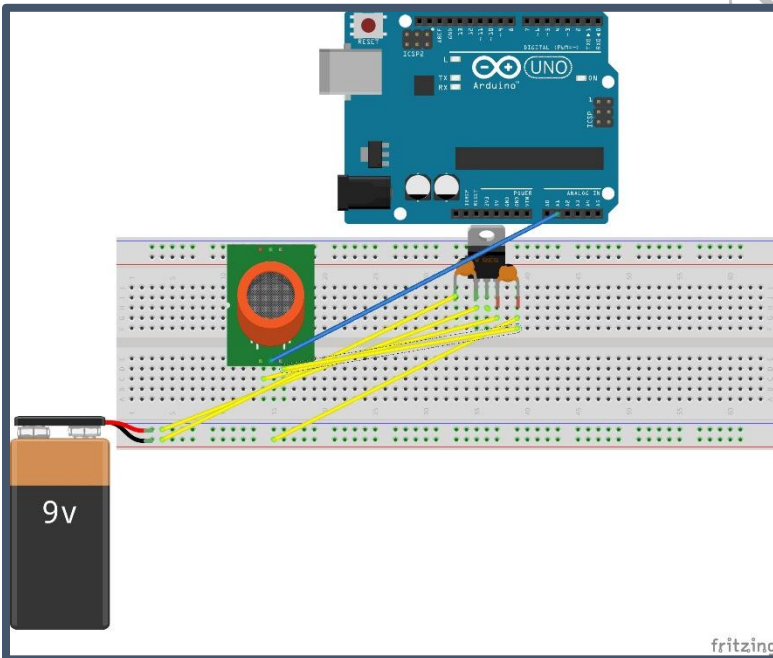


Figure 36: Setup for Gas and Dust Sensor

The dust sensor is setup in the similar way as the gas sensor and it uses the same voltage regulator.

Help button

It is used for emergency cases whenever the pushbutton is pressed it will send “Help :O” from the transmitter.

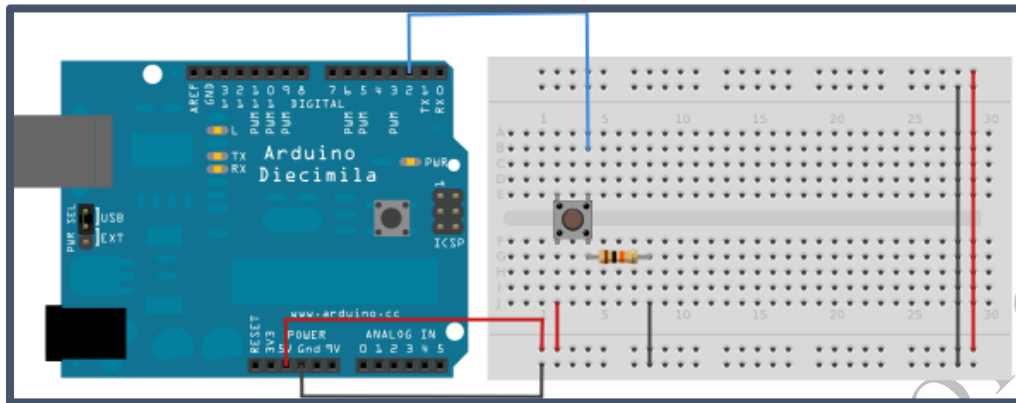


Figure 37: Pushbutton

Voltage Regulator

From the Arduino micro-controller it will power up the LCD, temperature and humidity sensor, transmitter and the motor. Therefore, for gas sensor and dust sensor external 9v battery is used to power them. Both gas and dust sensor requires 5v. To convert 9v to 5v we used a voltage to regulator to do the job.

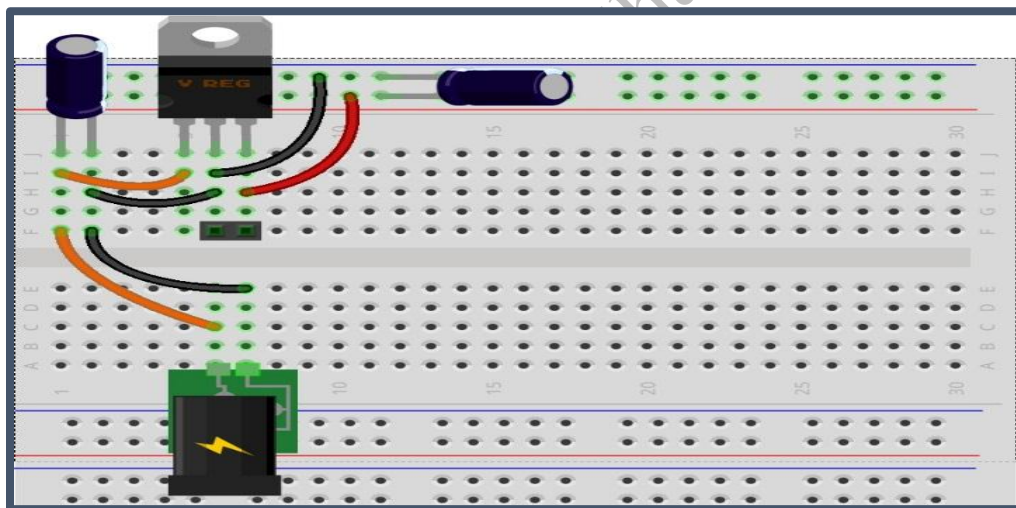


Figure 38: Setup for Voltage Regulator

LED

One LED is used with the microcontroller with transmitter. This blue LED will glow whenever there is harmful gas or dust particles are present.

Three different LEDs were used with receiver microcontroller. The green LED was used to indicate that the indoor air quality is safe. Red LED glows when there the air is polluted with harmful gas and dust particles. Blue LED determines whether the receiver is receiving the any new signals or not. More the blue LED blinks, more it receives the messages.



Figure 39: LED

Filtration System

The four sides will be covered with MERV filters. The picture below shows the approximate design of the system but it would be much smaller than this.



Figure 40: Setup for MERV filters

Overall (Approximate Setup)

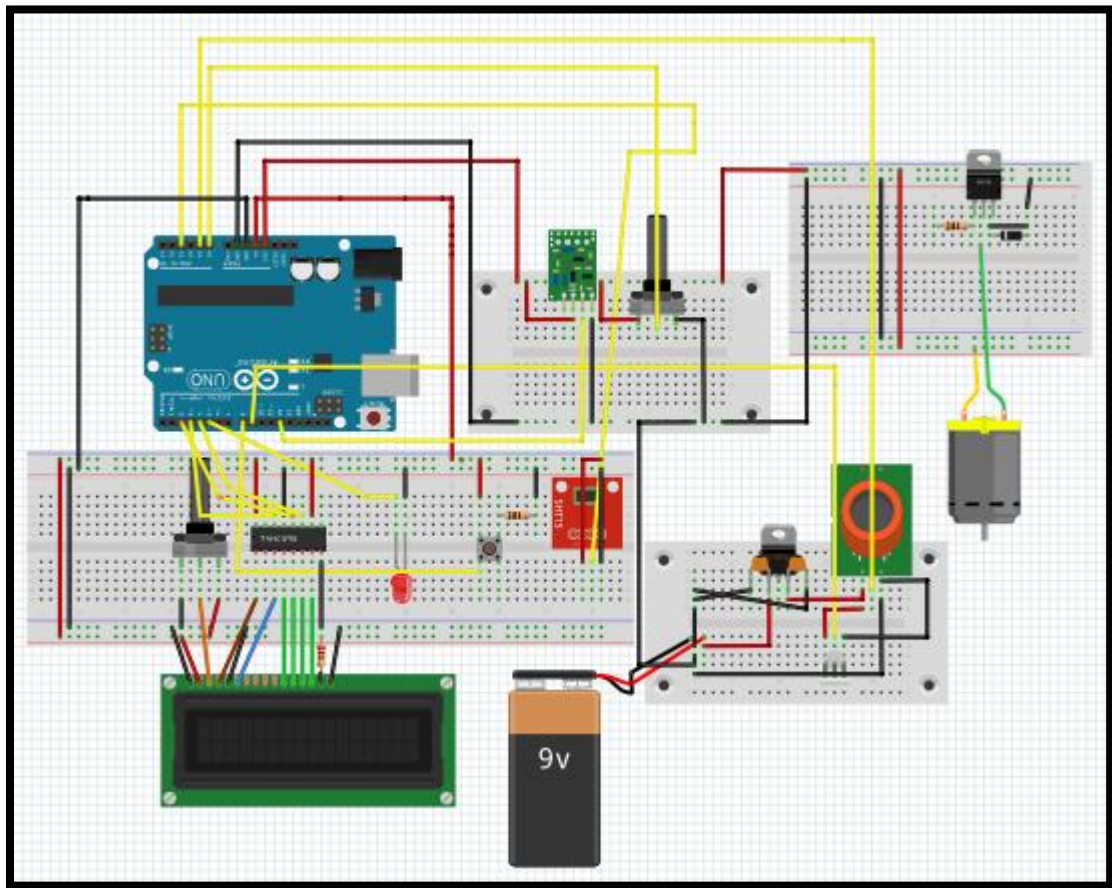


Figure 41: Overall Approximate Setup

Shabuktagin Photo

Debugging

It was the most difficult part of Arduino. All the debugging were done manually using Analog Discovery to find out the correct voltages and the precision errors. This is a non-intrusive method. Some of the graphs are given in this section

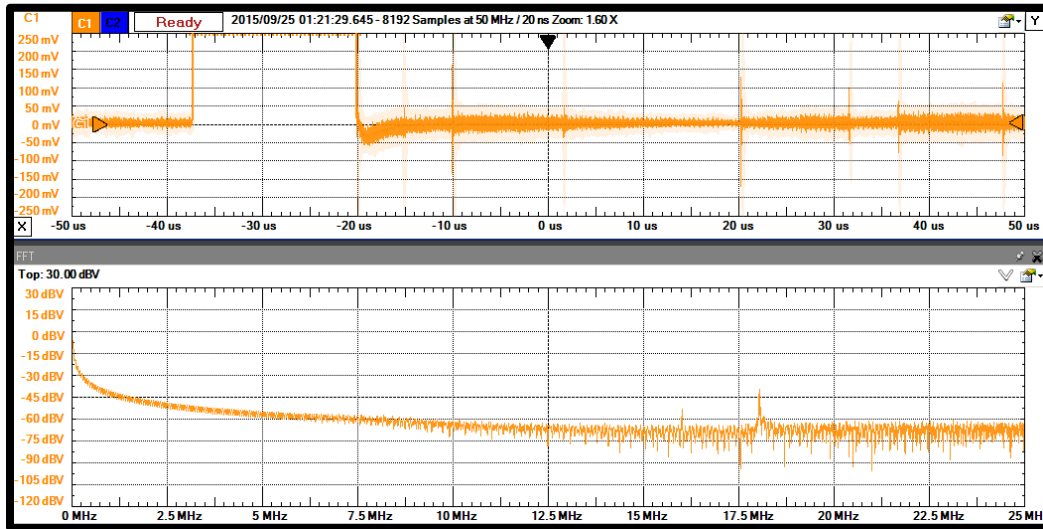


Figure 42: LCD Pin 2 Voltage

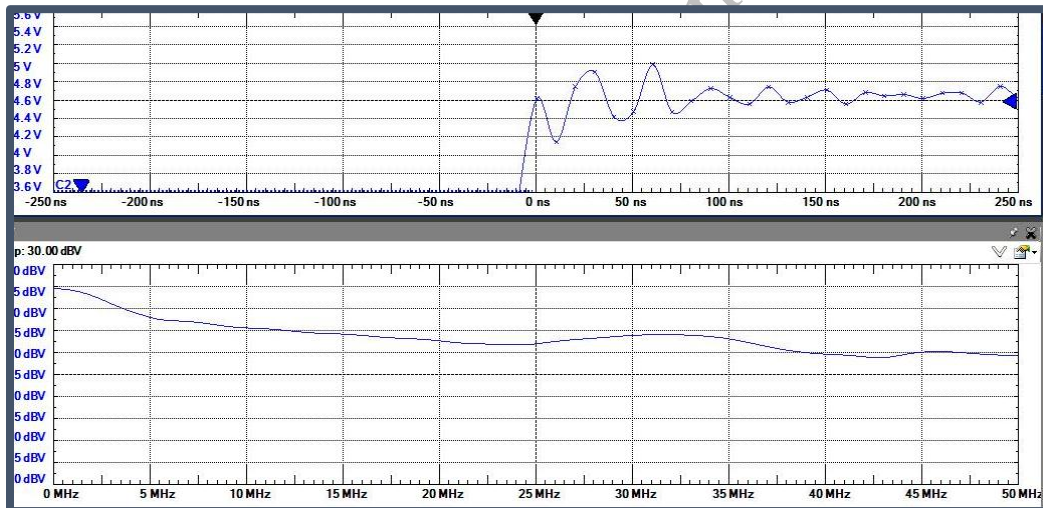


Figure 43: LCD Pin 3 Voltage

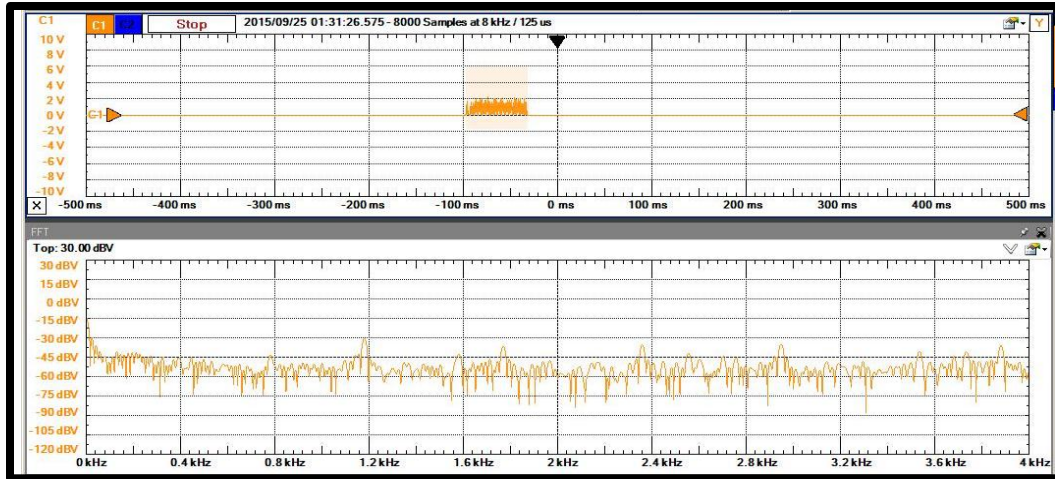


Figure 44: LCD Pin 5 Voltage

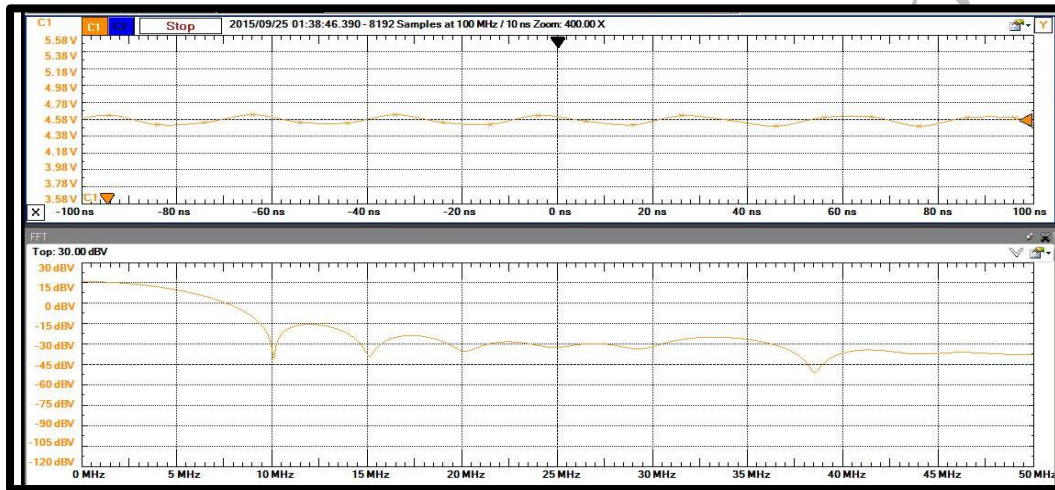


Figure 45: Blue LED Pin 5 Voltage

Shabuktagin Photo

(COPY)

Conclusion

Problems and Improvement

Limitations

Voltage Calculation Errors

For 3.3V

$$\text{Error (\%)} = \frac{\text{Expected Value} - \text{Original Value}}{\text{Expected Value}} * 100$$

$$= \frac{3.3 - 3.26}{3.3} * 100 = 1.2(\%)$$

For 5V

$$\text{Error (\%)} = \frac{\text{Expected Value} - \text{Original Value}}{\text{Expected Value}} * 100$$

$$= \frac{5 - 4.58}{5} * 100 = 8.4(\%)$$

Delay

On our previous circuit setup, there used to be a delay between the gas sensor and data acquisition. We fixed it using the correct pins for correct sensors.

When the gas sensor or the dust sensor recognizes the pollution, sometimes it take few seconds more for the LCD to show the “Warning” messages. There is delay between transmitter and receiver due to different factors. One of the factor is if there is too many obstacles in the channel like walls then the receiver might receive the full message late.

Environmental Protection Agency (EPA)

According to their standards, there are 5 major pollutants, out of 5 only 4 of them can be measured using our product. 5 major pollutants are ground level ozone, particle pollution, carbon monoxide, sulfur dioxide, nitrogen dioxide.

- Dust Sensor detects particle pollution
- Gas Sensor detects volatile organic compound (VOC), Carbon Monoxide.
- Sulphur Dioxide sensor is excluded

Ozone: Good ozone is what shields us from ultraviolet light which exists in upper atmosphere. Bad ozone is the one which is created on the ground. Ground level ozone is created from NOx, Volatile organic compound, heat and sunlight. Therefore, by measuring VOC we can find out ground ozone level.

Particle Pollution: Both fine particle and the course particle consists of solid and liquid droplet.

Carbon Monoxide: It is a colorless gas which enters through bloodstream which reduces the capacity of the blood to carry oxygen for organs.

Temperature and Humidity: Temperature is assumed to be 20C and the humidity is considered to be 33%.

Temp	Rs/Ro (85%)	Temp	Rs/Ro (33%)
-10	1.2	-10	1.45
5	0.98	5	1.12
10	0.88	10	1
20	0.75	20	0.88

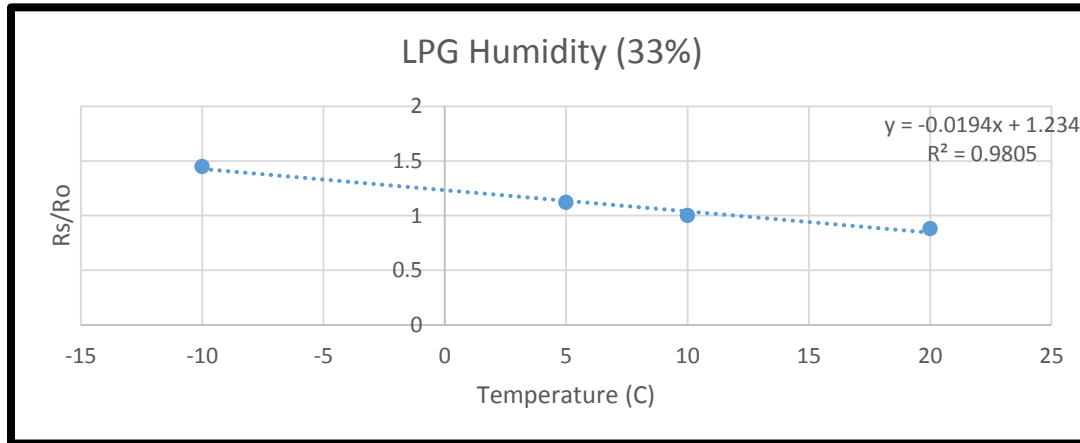


Figure 46: Liquefied Petroleum Gas varying with Temperature (33% Humidity Constant)

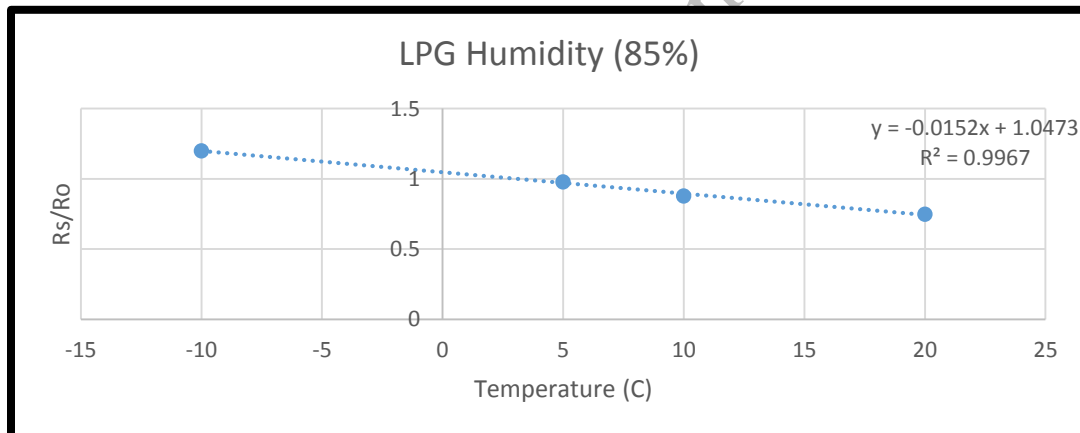


Figure 47: Liquefied Petroleum Gas varying with Temperature (85% Humidity Constant)

Transmitter and Receiver

It either works in 315MHz or 433MHz and it follows the proper FCC standards. It is only one way communication. It uses amplitude shift keying. It works only between 40m to 100m range. The range is higher when we send "HIGH" or "LOW" signal. When we send a message the range is lower.

Gas Sensor

It can measure from 20ppm indoor which is a concern for us because 20ppm is AQI=201 which is very risky for the sensitive people.

Dust Sensor

The sensor measure concentration in number of particles per cubic feet ($\frac{pcs}{CF}$). However AQI standard measures concentration in ug/m^3 . Therefore, we assumed 1 pcs=1ug.

Trigger Value for Gas Sensor

We will use 4.5 (ppm) which is 101 (AQI)

$\ln(PPM) = (3.2502 - \frac{R_s}{R_L})/0.5042$, derived from Figure 15

Find the value for $\frac{R_s}{R_L}$

$$\frac{R_s}{R_L} = \frac{V_c - V_{RL}}{V_{RL}}$$

Find the value for V_{RL}

$$V_{RL} = \frac{\text{arduinovalue}}{1024} * 5$$

Find the value for Arduino value

Arduino value = 153

While testing, we figured out the Arduino value gives around 30 to 60 when the air is clean. Therefore, we adjusted the trigger value to 90.

Trigger Value for Dust Sensor

We will use 2.8 (ppm) with (101) AQI

$2.8 \frac{ug}{m^3}$; m^3 is converted to cubic feet

$$0.01ft^3 = 0.00028m^3$$

which gives $10,000 \left(\frac{ug}{0.01ft^3} \right)$

Assuming 1pc = 1ug

Therefore, the trigger value is $10000 \left(\frac{pcs}{0.01ft^3} \right)$

PCB

We are using lots of breadboard for this project. We are planning to work on PCB so that circuit space gets really small and compatible. The following image illustrates the PCB circuit design that will be implement in the near future.

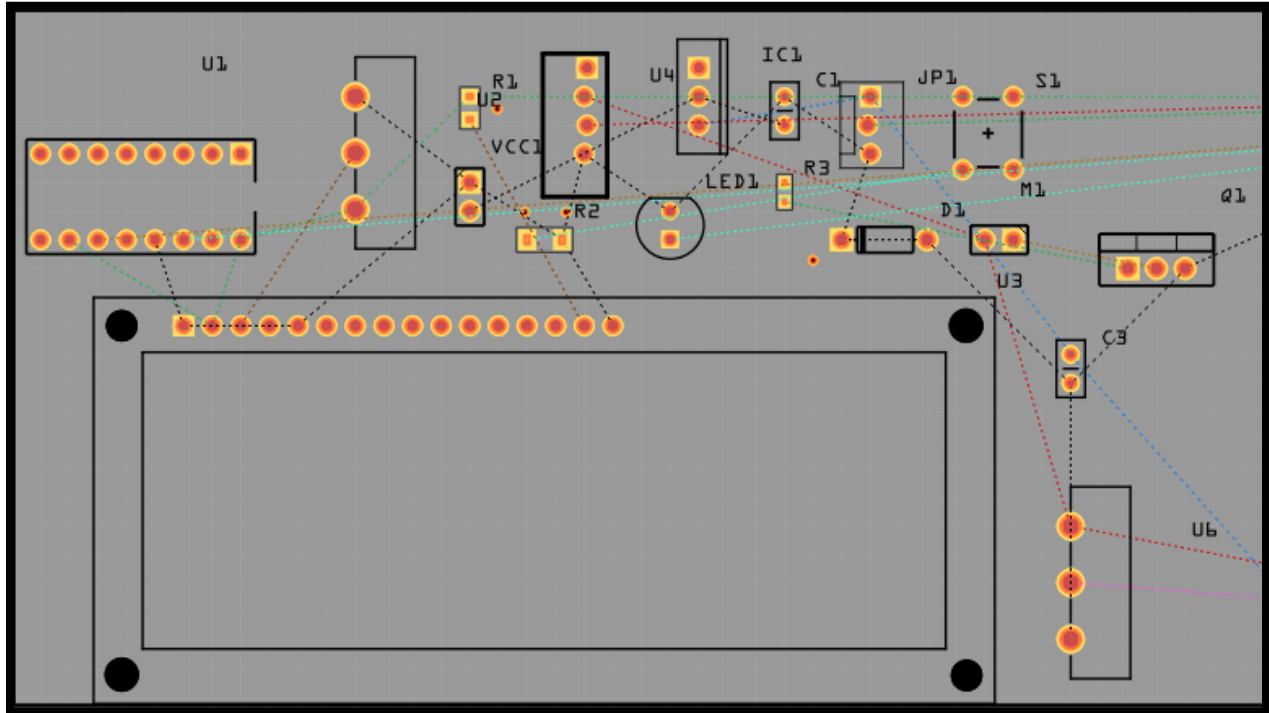


Figure 48: PCB

PSPICE

We used DC booster because only 3.26V is given to transmitter and motor. Instead of 3.26V, we want to supply 9V which can be done using DC booster. To analyze which circuit is best we would be using PSPICE to select the correct schematics for it.

PID Controller

Our motor calculation is based on the datasheet from the motor. We do not know the actual RPM of the motor. We attached a magnet at the back of the motor and have a Hall Effect sensor attached to it. It will help us to measure the frequency count of the motor which is the RPM. Using the RPM we will use the PID library which will determine the error between the set-point value of the RPM and the actual RPM from the motor. We can use this error to correct the speed of the motor using the algorithm.

Cost

We are still working to reduce the value of the product.

No:	Item	Quantity	Cost
1	Intake Fan	1	\$5
2	3D Printed Centrifugal Fan	1	\$9
3	PPD42NS Dust Sensor	1	\$8
4	MQ-9 Gas Sensor	1	\$5
5	Uno Microcontroller (Works like Arduino)	2	\$8
6	Transmitter and Receiver Kit	1	\$5
7	Liquid Crystal Display	2	\$10
8	Battery 9V	3	\$11
9	MERV Filter	4	\$16
10			
11			
	Approximate Total Cost		\$77

MERV Filter

For the filters, we have different ranges. The figure below shows all the necessary information.

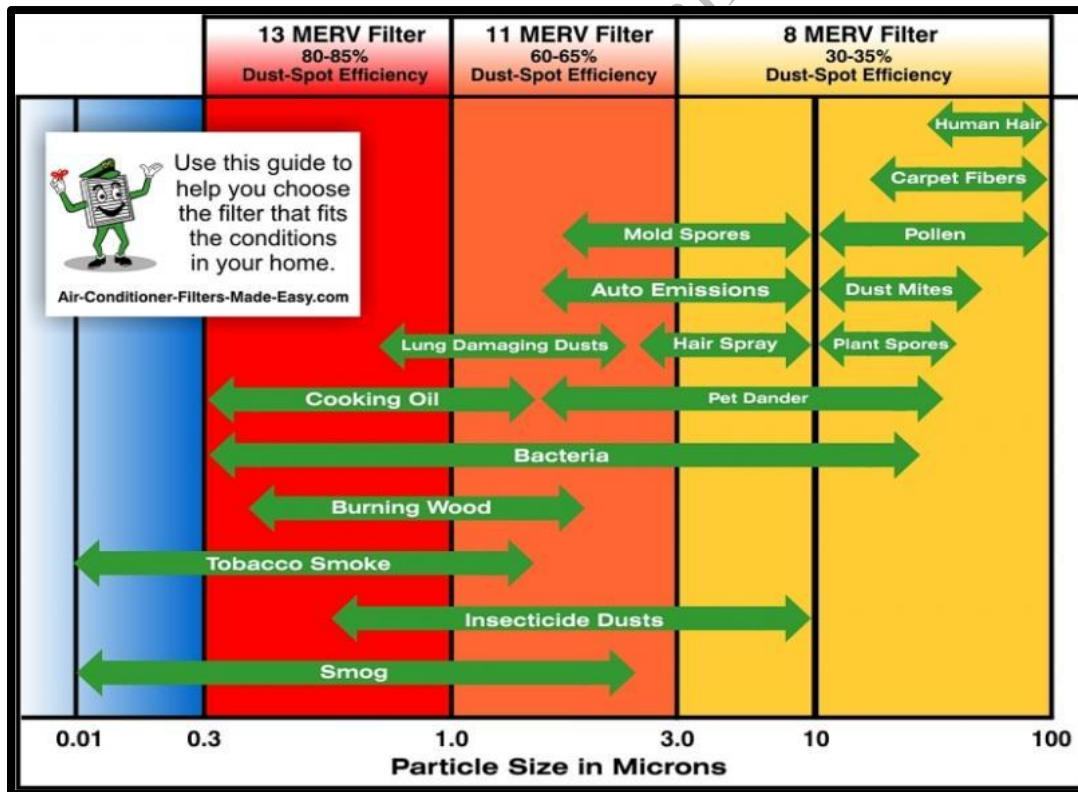


Figure 49: MERV Filters Chart

Work Distribution

Eric Nguyen worked with the design of the Air Shield and the motors. Shabuktagin Photon Khan worked with transmitter and receiver, the Arduino codes and merging the testing units and the Liquid Crystal Display using Arduino microcontroller. Juan Pineda-Aguirre worked to control the motor using multiple sensors such as a gas sensor, temperature/humidity and a dust sensor controlled by the Arduino. Tika Malla worked through different filters, taking notes of the tests and at the end worked as a quality control supervisor.

Project Schedule

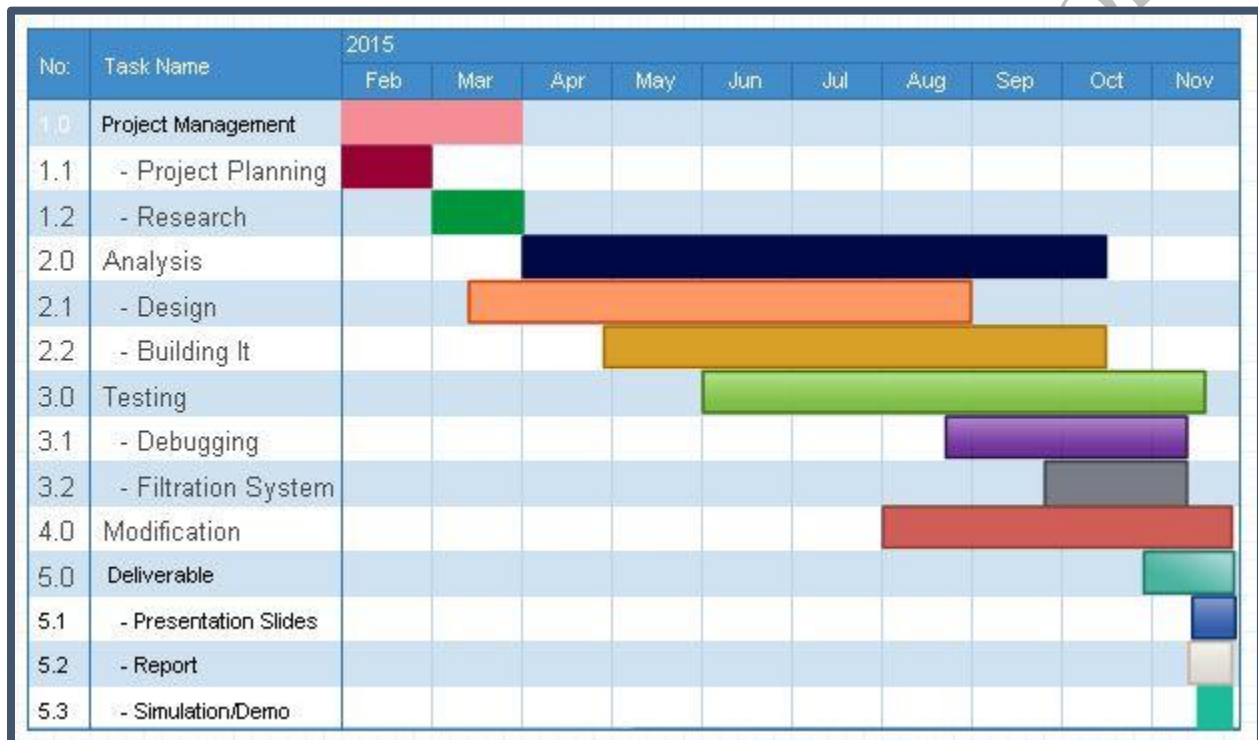


Figure 50: Project Schedule

Standards and Ethical Issues

- IEEE 802-Wired and Wireless Communication Standards: The IEEE 802 Standard comprises a family of networking standards that cover the physical layer specifications of technologies from Ethernet to wireless.
- IEEE C2-1997: National Electric Safety Code: This standard covers basic provisions for safeguarding of persons from hazards arising from the installation, operation, or maintenance of 1) conductors and equipment in electric supply stations, and 2) overhead and underground electric supply and communication lines. It also includes work rules for the construction, maintenance, and operation of electric supply and communication lines and equipment
- National Ambient Air Quality Standards (NAAQS) : EPA has set National Ambient Air Quality Standards for six principal pollutants, which are called "criteria" pollutants. Those components are Carbon Monoxide, Lead, Nitrogen dioxide, Ozone, Sulfur Dioxide and particle pollution.

Controlling the quality of air has become a major concern as it is necessary for human life. Research on indoor Air quality has shown that air indoor is two to five times more likely to be polluted than the outdoor air resulting in the rise in research on environmental interventions, such as balancing benefits and risks. Different types of air pollutants reflect distinct ethical challenges. One of the most significant problem in most of the countries is the air pollution from industrial sources. So air quality reflects the sum of many different sources, which makes it difficult to identify the main reason that should be blamed. And it won't be wrong to say that everyone undertakes activities that release pollutants. Which makes us all collectively responsible.

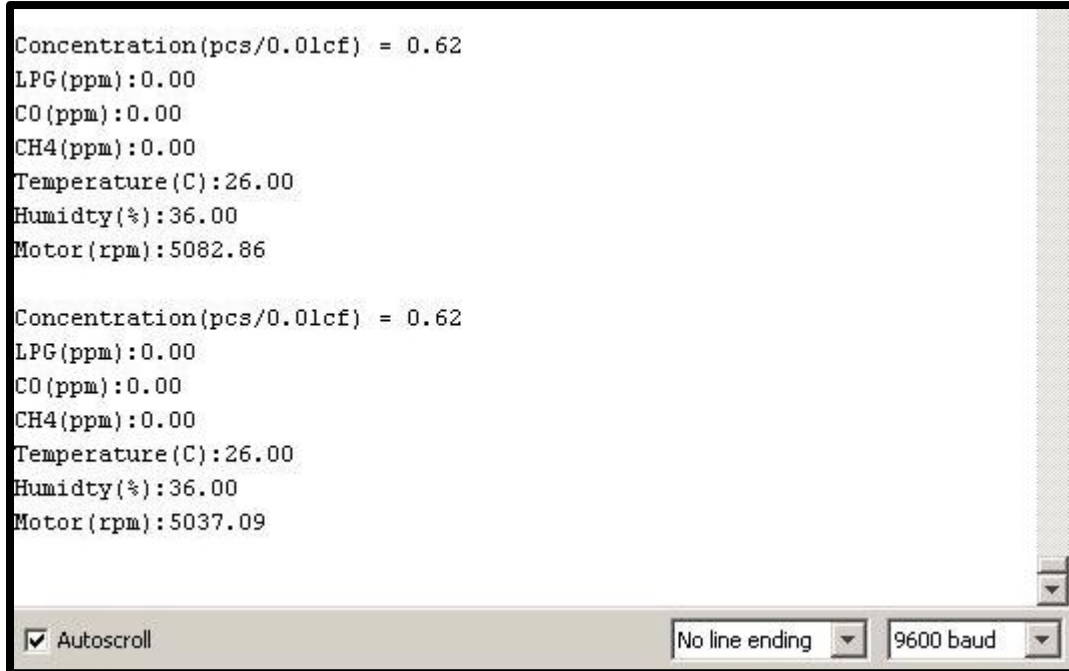
Dramatic reductions in air pollution has been done with clean air act, preventing hundreds of thousands of cases of serious health effects each year in United States. It has solved multiple air pollution problems but we still have a long way to go. Despite the dramatic progress to date, air pollution still continues to threaten Americans' health and welfare.

Results

Arduino (USB Serial Monitor)

First Microcontroller

When the room is safe:



The screenshot shows the Arduino USB Serial Monitor window with the following data displayed:

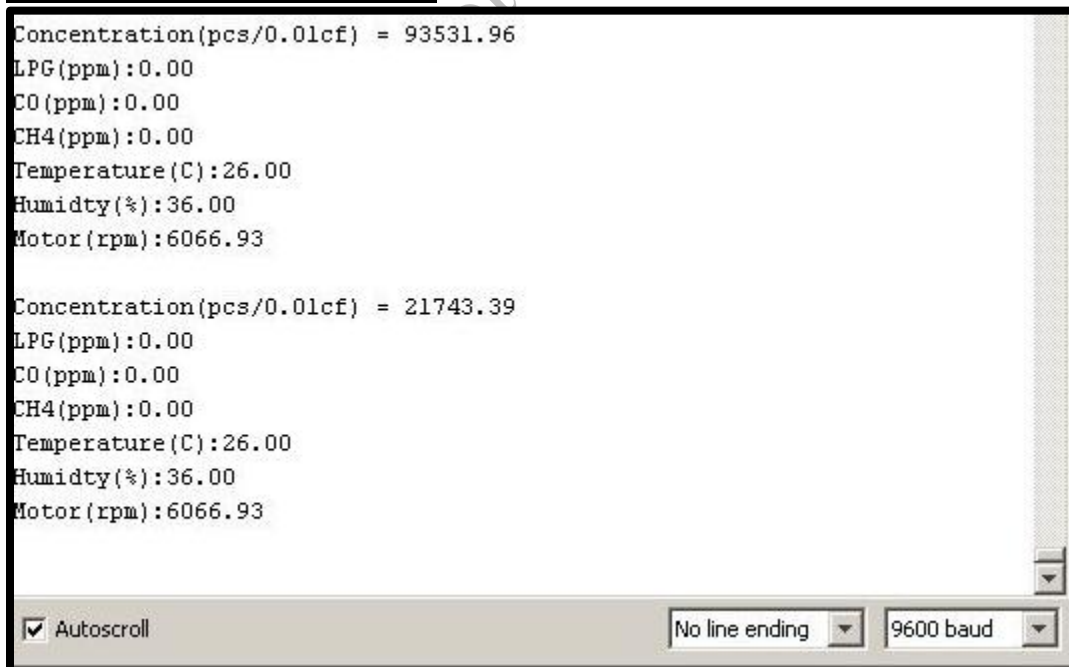
```
Concentration(pcs/0.01cf) = 0.62
LPG(ppm):0.00
CO(ppm):0.00
CH4(ppm):0.00
Temperature(C):26.00
Humidity(%):36.00
Motor(rpm):5082.86

Concentration(pcs/0.01cf) = 0.62
LPG(ppm):0.00
CO(ppm):0.00
CH4(ppm):0.00
Temperature(C):26.00
Humidity(%):36.00
Motor(rpm):5037.09
```

The window includes a checked 'Autoscroll' checkbox, a 'No line ending' dropdown menu, and a '9600 baud' dropdown menu.

Figure 51: Room is Safe (Arduino 1)

When there is high amount of dust:



The screenshot shows the Arduino USB Serial Monitor window with the following data displayed:

```
Concentration(pcs/0.01cf) = 93531.96
LPG(ppm):0.00
CO(ppm):0.00
CH4(ppm):0.00
Temperature(C):26.00
Humidity(%):36.00
Motor(rpm):6066.93

Concentration(pcs/0.01cf) = 21743.39
LPG(ppm):0.00
CO(ppm):0.00
CH4(ppm):0.00
Temperature(C):26.00
Humidity(%):36.00
Motor(rpm):6066.93
```

The window includes a checked 'Autoscroll' checkbox, a 'No line ending' dropdown menu, and a '9600 baud' dropdown menu.

Figure 52: Room is not Safe (Arduino 1, Dust)

When there is high amount of gas:

```

LPG(ppm):71.90
CO(ppm):50.22
CH4(ppm):159.41
Temperature(C):26.00
Humidity(%):36.00
Motor(rpm):6066.93

Concentration(pcs/0.01cf) = 0.62
LPG(ppm):2.00
CO(ppm):1.69
CH4(ppm):1.68
Temperature(C):22.00
Humidity(%):33.00
Motor(rpm):6066.93

```

Figure 53: Room is not Safe (Arduino 1, Gas)**Second Microcontroller****When air is clean:**

```

COM8 (Arduino Uno)
Device is ready
Current Status: Safe :)
Current Status: Safe :)
Current Status: Safe :)
Current Status: Safe :)

```

Figure 54: Room is Safe (Arduino 2)**When air is polluted:**

```

Current Status: Warning
Current Status: Warning
Current Status: Warning

```

Figure 55: Room is not Safe (Arduino 2)

Data Acquisition

USB Serial and MATLAB

We can collect data using MATLAB and collect the data with respect to number of samples using USB serial. Only one kind of data can be passed through the USB serial (ex: if we are plotting for gas sensor then plotting for dust sensor will not be possible). The MATLAB code is given at the end of the report.

For Individual Plot (Gas Data):

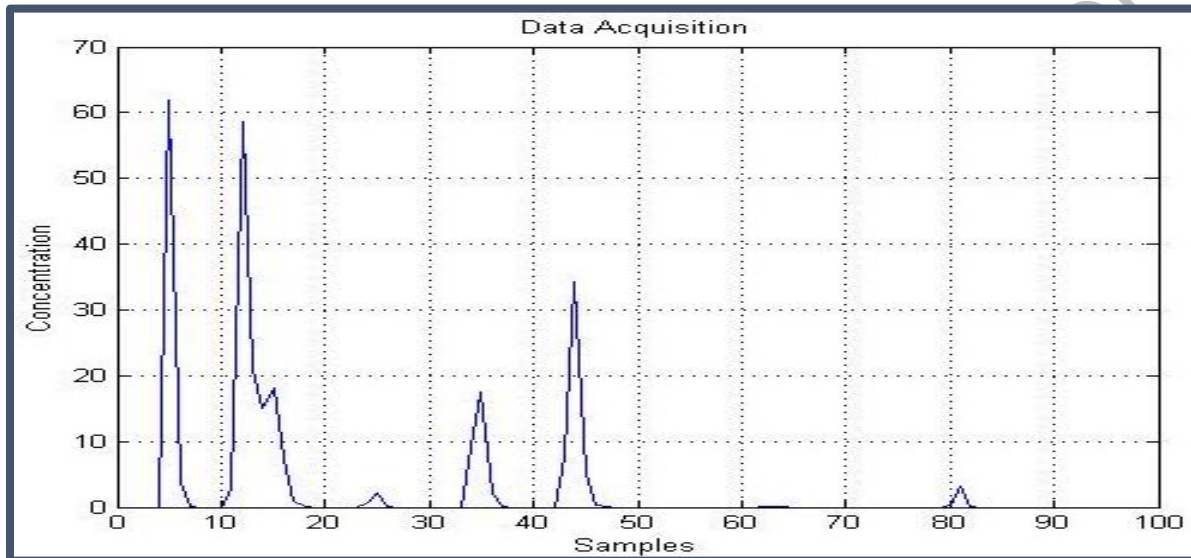


Figure 56: Data Acquisition through MATLAB

Polluted Air in Room Estimation:

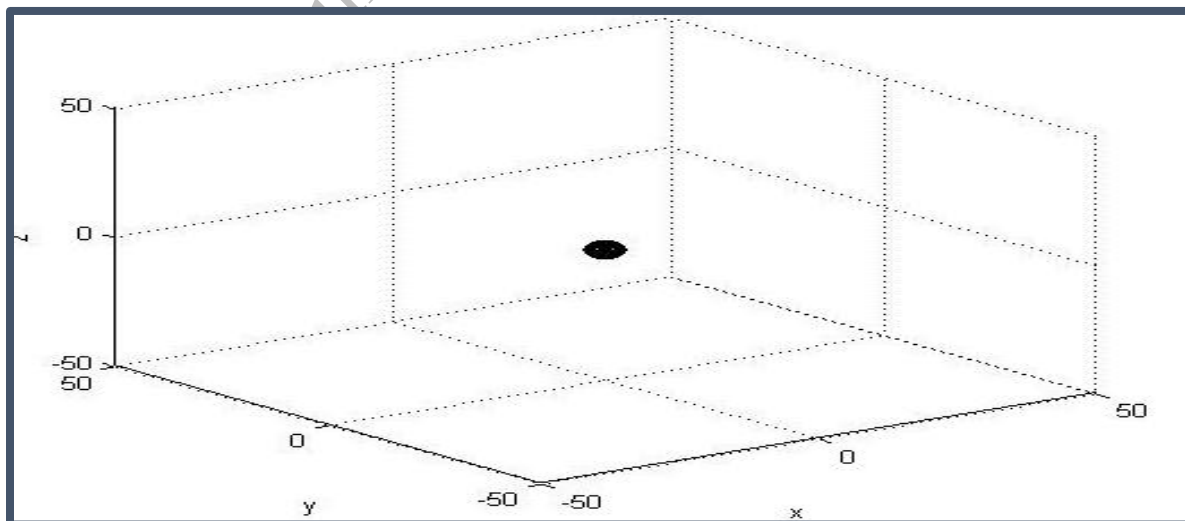


Figure 57: Low Amount of Pollution

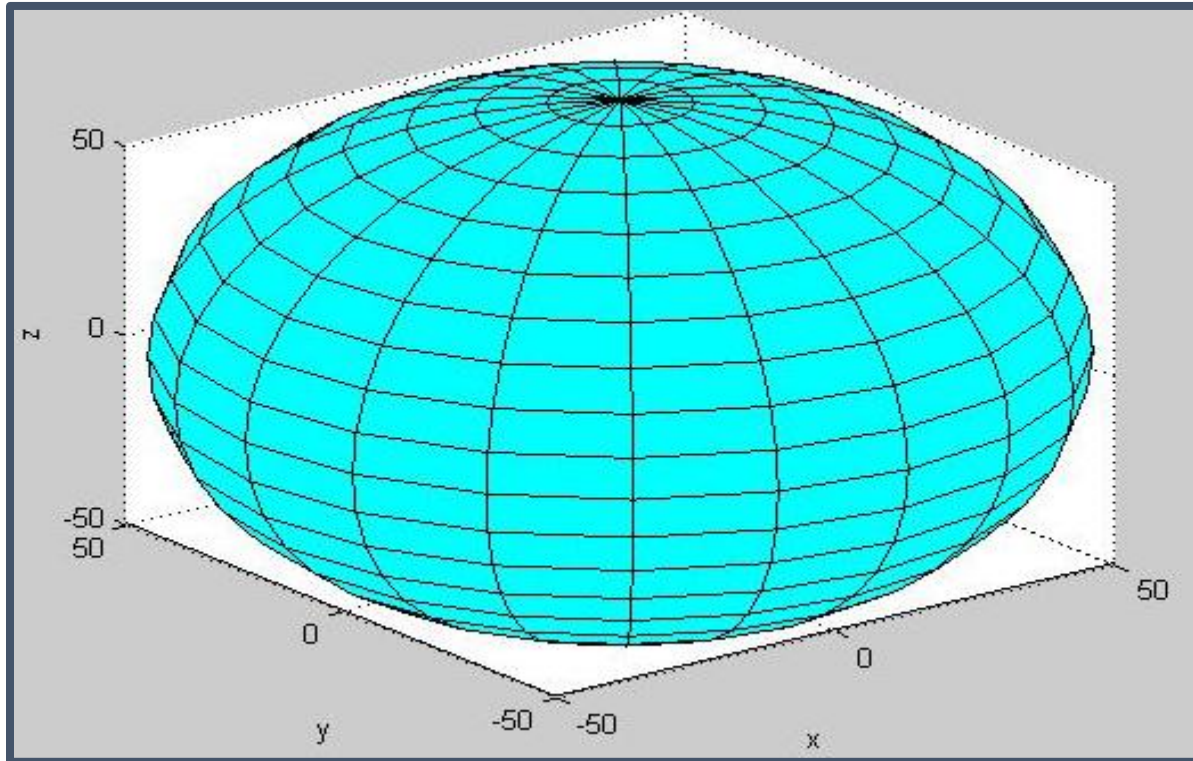


Figure 58: High Amount of Pollution

Shabuktagin Photon KIR

LCD Screen Display

Microcontroller 1

When the area is clean:

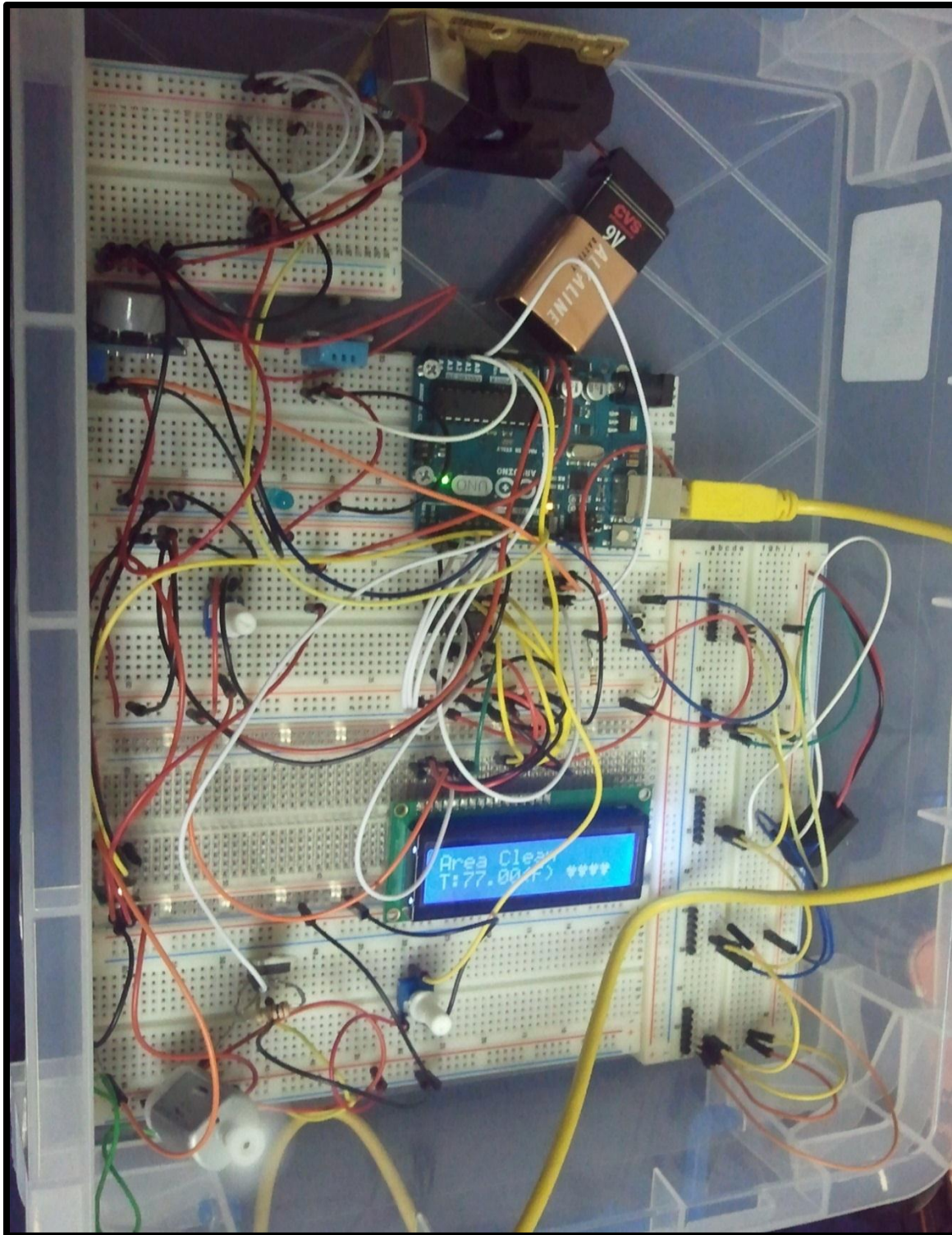


Figure 59: Microcontroller 1, Area Clean

When dust is detected:



Figure 60: Microcontroller 1, Dust Detection

When gas is detected:

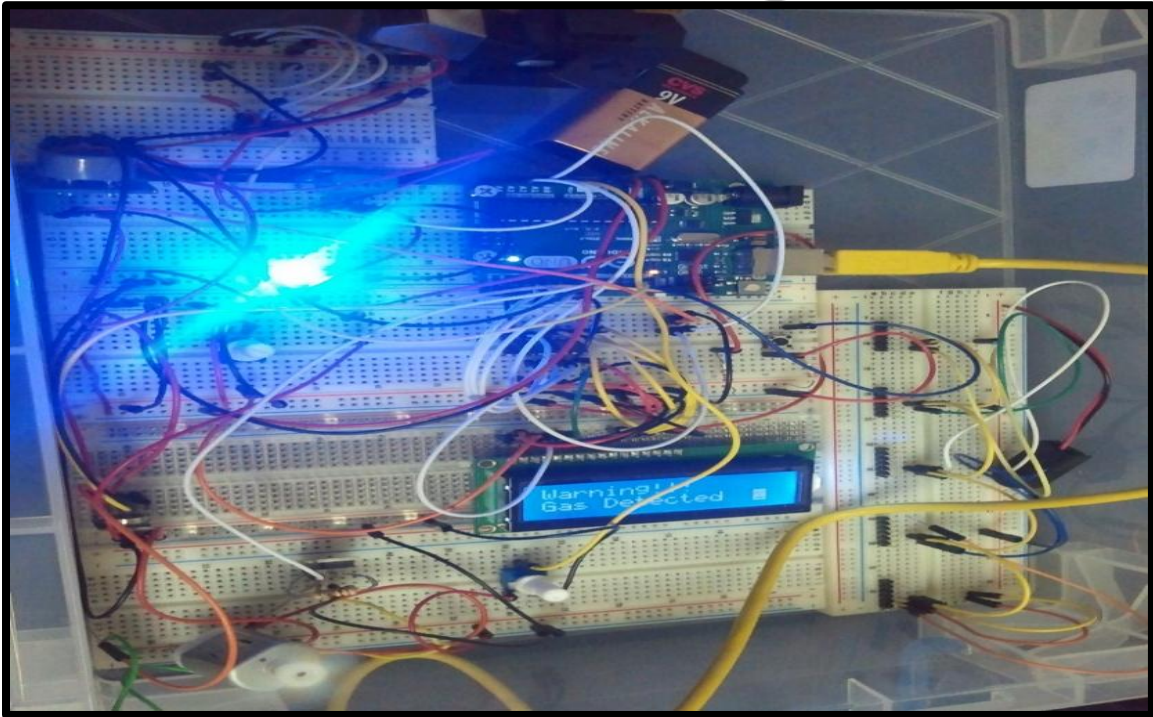


Figure 61: Microcontroller 1, Gas Detection

Microcontroller 2

When the area is clean:

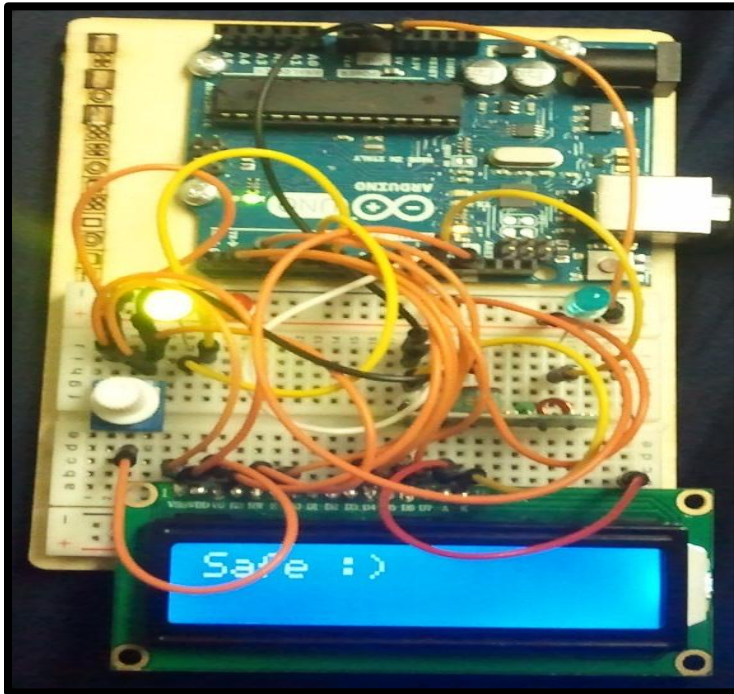


Figure 62: Microcontroller 2, Safe©

When the area is polluted:

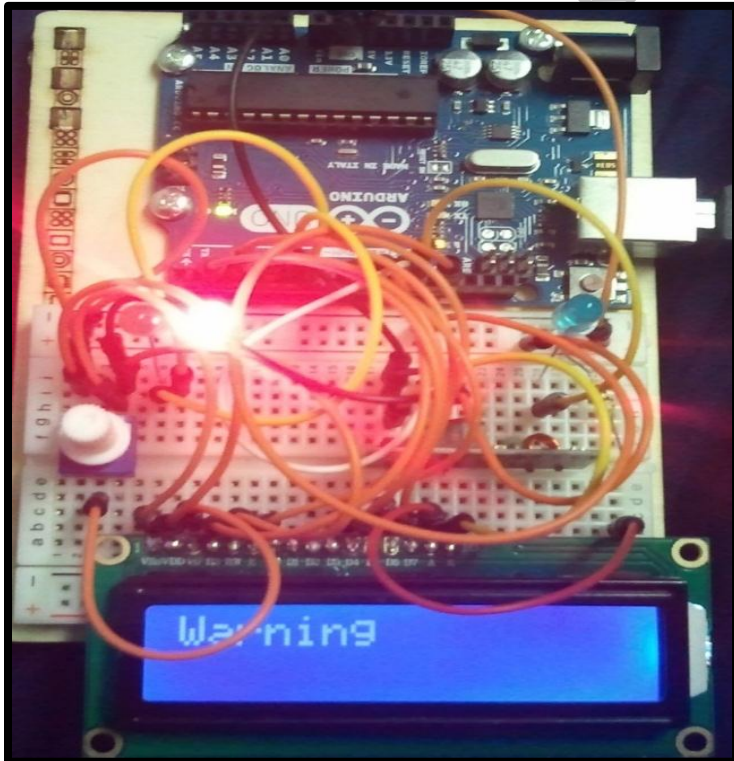


Figure 63: Microcontroller 2, Warning!

References

- [1] Greiner, Thomas H. "Carbon Monoxide Poisoning: Checking for Complete Combustion (AEN-175)." Agricultural and Biosystems Engineering. Iowa State University College of Engineering College of Agriculture and Life Sciences, 8 Sept. 1997. Web. 30 Apr. 2015. <<http://www.abe.iastate.edu/extension-and-outreach/carbon-monoxide-poisoning-checking-for-complete-combustion-aen-175/>>. [1]
- [2] X. Du, Y. Du, and S.M George. "CO Gas Sensing by Ultrathin Tin Oxide Films Grown by Atomic Layer Deposition Using Transmission FTIR Spectroscopy." Departments of Chemical and Biological Engineering and of Chemistry and Biochemistry, University of Colorado, Boulder, Colorado 80309, Revised: June 27, 2008 [2]
- [3] Flame Light Datasheet for 5mm Silicon PIN Photodiode, T-1 ¾ PD 333-3B/H0/L2, (Everlight, May.27.2013) < <http://www.everlight.com/file/ProductFile/PD333-3B-H0-L2.pdf>>
- [4] Gas Sensor Datasheet for MQ-9 Gas Sensor, (HANWEI ELETRONICS CO.,LTD) < <http://www.sgbotic.com/products/datasheets/sensors/MQ-9.pdf>>
- [5] <http://www.saylor.org/site/wp-content/uploads/2011/09/Chapter-3.5-Fans-Blowers.pdf>
- [6] Wikipedia. Wikimedia Foundation. Web. 30 Apr. 2015. <http://en.wikipedia.org/wiki/Centrifugal_fan#Fan_ribs>.
- [7] Web. 30 Apr. 2015. <<http://www.tcf.com/docs/fan-engineering-letters/fan-performance-characteristics-of-centrifugal-fans---fe-2400.pdf?Status=Master>>.
- [8] "Centrifugal Impeller / Turbine / Compressor by Engi-IRL." - Thingiverse. Web. 30 Apr. 2015. <<http://www.thingiverse.com/thing:590416>>.
- [9] "Grove - Dust Sensor." Wwww.epictinker.com. Web. 30 Apr. 2015. <<http://www.epictinker.com/Grove-Dust-Sensor-p/sen12291p.htm>>.

Arduino Codes

First Micro-controller

It is responsible for the control Air filtration system, Air monitoring and the Transmitting Data.

```
#include <LiquidCrystal595.h>
#include <VirtualWire.h>
#include <math.h>
#include "DHT.h"
```

```
#define DHTPIN A3
#define DHTTYPE DHT11
```

```
byte loveChar[8] = {
    B00000,
    B01010,
    B11111,
    B11111,
    B01110,
    B01110,
    B00100,
    B00000
};
```

```
byte skeletonChar[8] = {
    B11111,
    B11111,
    B10101,
    B11111,
    B11011,
    B11111,
    B10001,
    B11111
};
```

```
const int transmit_pin=12;
LiquidCrystal595 lcd(2,3,4);
```

```
int warning=5;
int motorpin=6;
int motorspeed=0;
int potentiometer=0;
float information=0; //Gas Sensor
float originaltemperature=0; //Temperature Sensor
float humidity=0; //Humidity
float PPM1=0; //LPG
float PPM2=0; //CO
float PPM3=0; //Methane
```

```
float ratio2=0;
float Rs=0;
float RI=20000;
float VRL=0;

float fahrenheit=0;
const int buttonPin = 8;
DHT dht(DHTPIN, DHTTYPE);

int dustpin = 9;
unsigned long duration;
unsigned long starttime;
unsigned long samptime_ms=250;
unsigned long lowpulseoccupancy = 0;
float ratio = 0;
float concentration = 0;    //Dust Sensor
float pulseWM=0;
float volt=0;
float rpm=0;

void setup()
{
  Serial.begin(9600);
  lcd.begin(16,2);
  dht.begin();
  pinMode(warning,OUTPUT);
  vw_set_tx_pin(transmit_pin);
  vw_setup(2000);
  starttime = millis();
}

void loop()
{
  helpbutton();
  potentiometer=analogRead(A0);
  information=analogRead(A1);
  ppmcalculation();
  originaltemperature = dht.readTemperature();
  humidity = dht.readHumidity();

  //Data Sheet and the Temperature Conversion
  fahrenheit=(originaltemperature*1.8)+32;

  //Measuring Concentration of Dust Sensor
  duration = pulseIn(dustpin, LOW);
```

```
lowpulseoccupancy = lowpulseoccupancy+duration;

if ((millis()-starttime) > samptime_ms)
{
  ratio = lowpulseoccupancy/(samptime_ms*10.0);
  concentration = 1.1*pow(ratio,3)-3.8*pow(ratio,2)+520*ratio+0.62;
  Serial.print("Concentration(pcs/0.01cf) = ");
  Serial.println(concentration);
  lowpulseoccupancy = 0;
  starttime = millis();
}

//Dangerous Chemicals/Gases or not
if (information>=90 || concentration>=10000)
{
  serialinformation();
  digitalWrite(warning,HIGH);
  send("Warning");
  lcd.clear();
  if(information>=90)
  {
    lcd.setCursor(0,0);
    lcd.print("Warning!!!");
    lcd.setCursor(0,1);
    lcd.print("Gas Detected");
    skeleton();
    delay(500);
  }

  if(concentration>=10000)
  {
    lcd.setCursor(0,0);
    lcd.print("Warning!!!");
    lcd.setCursor(0,1);
    lcd.print("Dust Detected");
    skeleton();
    delay(500);
  }
}

else
{
  clean();
}
}
```

```
void clean (void)
{
  serialinformation();
  send("Safe :)");
  digitalWrite(warning,LOW);

  lcd.setCursor(0,0);
  lcd.print("Area Clean");
  lcd.setCursor(0,1);
  lcd.print("T:");
  lcd.print(fahrenheit);
  lcd.print("(F) ");

  lcd.createChar(0, loveChar);
  lcd.setCursor(11,1);
  lcd.write(byte(0));
  lcd.setCursor(12,1);
  lcd.write(byte(0));
  lcd.setCursor(13,1);
  lcd.write(byte(0));
  delay(500);
}

void send (char *message)
{
  vw_send((uint8_t *)message, strlen(message));
  vw_wait_tx();
}

void skeleton(void){
  lcd.createChar(0, skeletonChar);
  lcd.setCursor(14,1);
  lcd.write(byte(0));
}

void helpbutton(void){
  int reading = digitalRead(buttonPin);
  if (reading == HIGH) {
    send("Help :0");
  }
}

void serialinformation(void){
  Serial.print("LPG(ppm):");
  Serial.println(PPM1);
  Serial.print("CO(ppm):");
```

```
Serial.println(PPM2);
Serial.print("CH4(ppm):");
Serial.println(PPM3);
Serial.print("Temperature(C):");
Serial.println(originaltemperature);
Serial.print("Humidity(%):");
Serial.println(humidity);

if (information>=90|| concentration>=10000)
{
  motorspeed=255;
  analogWrite(motorpin,motorspeed);
}
else
{
  motorspeed= map(potentiometer,0,1023,120,255);
  analogWrite(motorpin,motorspeed);
}
motorspeedcalculation();
Serial.print("Motor(rpm):");
Serial.println(rpm);
Serial.print("\n");
}

void ppmcalculation()
{
  //Data Sheet conversion
  VRL=(information/1024)*5;
  Rs=R1*((5-VRL)/VRL);
  ratio2=Rs/R1;
  PPM1=exp((3.314-ratio2)/0.4768);
  PPM2=exp((3.2502-ratio2)/0.5042);
  PPM3=exp((3.1799-ratio2)/0.3755);
}

void motorspeedcalculation()
{
  //Data Sheet conversion
  pulseWM=(0.0039*motorspeed)+0.0394;
  volt=pulseWM*3.26;
  rpm=volt*1800;
}
```

Second Micro-controller

It is responsible to receive the data from the transmitter.

```
#include <LiquidCrystal.h>
#include <VirtualWire.h>

const int receive_pin = 11;
LiquidCrystal lcd(10,9,5,4,3,2);
byte message[VW_MAX_MESSAGE_LEN];
byte messageLength = VW_MAX_MESSAGE_LEN;
int pos = 0;
char string1[ ] = "Safe :)";
char string2[ ] = "Help :0";
int warningpin=7;
int safepin=6;
const int led_pin = 12;

void setup()
{
  Serial.begin(9600);
  Serial.println("Device is ready");
  lcd.begin(16,2);
  pinMode(warningpin,OUTPUT);
  pinMode(safepin,OUTPUT);
  vw_set_rx_pin(receive_pin);
  vw_setup(2000);
  vw_rx_start();
}

void loop()
{
  if (vw_get_message(message, &messageLength))
  {
    digitalWrite(led_pin, HIGH);
    Serial.print("Current Status: ");
    if(pos < 2)
    {
      lcd.setCursor(0, pos);
    }
    else
    {
      pos=0;
      lcd.clear();
    }
  }
}
```

```
for (int i = 0; i < messageLength; i++)
{
  Serial.write(message[i]);
  lcd.print((char)message[i]);
  if (string1[i]==((char)message[i]))
  {
    digitalWrite(safepin,HIGH);
    digitalWrite(warningpin,LOW);
  }
  else if (string2[i]==((char)message[i]))
  {
    digitalWrite(warningpin,LOW);
    delay(200);
    digitalWrite(warningpin,HIGH);
    delay(200);
  }
  else
  {
    digitalWrite(safepin,LOW);
    digitalWrite(warningpin,HIGH);
  }
  pos++
}
Serial.println();
digitalWrite(led_pin, LOW);
}
}
```

Shabuktagin Photon Khan (DO NOT COPY)

MATLAB codes

For individual reason one at a time, which is, we can only plot for gas sensor and cannot simultaneously plot for dust sensor.

Individual Plot

```
delete(instrfindall);
clear all;
clc;
```

```
arduino=serial('COM9','BaudRate',9600);
fopen(arduino);
```

```
%Datas after every 4s
y=zeros();
for i=1:1:10
    y(i)=fscanf(arduino,'%f');
    drawnow;
    plot(y);
    xlabel('Samples');
    ylabel('Concentration');
    title('Data Acquisition');
    grid on;
    hold on;
end
```

```
fclose(arduino);
delete(arduino);
close all;
```

Polluted Air in Room Estimation

```
delete(instrfindall);
clc;
close all;
clear all;

%% Take Datas from arduino
arduino=serial('COM9','BaudRate',9600);
fopen(arduino);
%% set axis and figure
myaxes=axes('xlim',[-50 50],'ylim',[-50 50],'zlim',[-50 50]);
view(3);
```



```
grid on;
hold on;
xlabel('x');
ylabel('y');
zlabel('z');

%% Generating Sphere
[x y z]=sphere();
%% Shapes
h(1) = surface(x,y,z,'FaceColor','cyan');

%% Create a group Object
combinedobject=hgtransform('parent',myaxes);
set(h,'parent',combinedobject)

%% Animation of the group Object
formatSpec = '%d %f';
sizeA = [1 1];
for i = 1:1:10
    q=fscanf(arduino,formatSpec,sizeA);
    % Scaling matrix
    Sxy = makehgtform('scale',1+ abs(q));
    % set the transform Matrix property
    set(combinedobject,'Matrix',Sxy)
    drawnow;
end
fclose(arduino);
delete(arduino);
close all;
```

Shabuktagin Photon Khan (DO NOT COPY)

Index

- air purifier, 1
 Air Quality Index, 7
 Air Shield, 1, 2, 3, 4, 6, 7, 35
 airflow, 2
 antenna, 24, 25
 AQI, 7, 8, 9, 10, 11, 33
 Arduino, 1, 6, 16, 17, 19, 20, 26, 28, 29, 33, 34, 35, 45
 blink, 21
 bloodstream, 32
 carbon monoxide, 7, 31
 Carbon monoxide, 4
 Carbon Monoxide, 4, 8, 13, 15, 26, 31, 32, 44
 centrifugal, 2, 44
 circuit, 1, 6, 31, 34
 concentrations, 4
 data, 6, 7, 8, 16, 17, 21, 31, 38, 50
 datasheet, 11, 12, 15, 20, 21, 24
 DC booster, 34
 debug, 6
 DHT11, 4, 5, 45
 dust particles, 1, 4, 5, 28
 errors, 29
 filters, 1, 2, 28, 29, 34, 35
 gas sensor, 4, 27, 28, 31, 35, 39, 52
 ground level ozone, 31
 indoor air, 7, 28
 Launch Pads, 6
 LCD display, 6
 LED, 26, 28, 30
 Liquefied Petroleum Gas, 13, 14, 32
 Low Pulse Occupancy Time, 15, 16
 LPOT, 16, 17
 MATLAB, 38, 52
 medical purposes, 1
 Methane, 14, 15, 45
 microcontroller, 5, 6, 12, 16, 26, 28, 35
 Microsoft Excel, 12, 16
 motor, 1, 2, 6, 17, 19, 28, 34, 35
 non-intrusive, 29
 Operation Principle, 12
 PCB, 34
 PM10, 7
 PM2.5, 7
 pollution, 7, 31
 PPD42NS, 5, 15, 17, 34
 PPM, 4, 8, 9, 10, 11, 12, 13, 14, 15
 precision, 29
 PSPICE, 34
 pulse width modulation, 17
 purify, 1
 receiver, 6, 21, 24, 28, 31, 35
 sensitive, 1, 33
 sensitivity, 11, 15
 setup, 25, 27, 31, 46, 50
 signal, 5, 12, 21, 33
 speed, 6
 standards, 1, 31, 33
 Sulphur Dioxide, 31
 Temperature, 4, 5, 26, 32, 45, 46, 49
 Tinker-CAD, 1
 transmitter, 6, 21, 24, 25, 27, 28, 31, 34, 35, 50
 voltage to regulator, 28

Attendance Sheet

Advisor: Dr. Namuduri

Week of	Group Member(s) Attending				Advisor Signature	Notes
	SHABUKTAGAN MAYTON KIMAN	ERIC NGUYEN	Juan Pineda	Tika Kumari Mullie		
X 30-Aug 9/10/2015						
6-Sep 9/17/2015	✓	✓	✓	✓	Kamru	1. when will we apply? Any -va impact?
13-Sep	✓	✓	✓	✓	Kamru	2. progress is good. Read about limitations
20-Sep	✓	✓	✓	✓	Kamru	3. Discuss the plots.
27-Sep	✓		✓		Kamru	4. start working on reports.
4-Oct	✓	✓	✓	✓	Kamru	Same as above
11-Oct	✓	✓	✓	✓	Kamru	Start on presentation
18-Oct						
25-Oct	✓	✓	✓	✓	Kamru	"
1-Nov	✓	✓	✓	✓	Kamru	Made good progress
8-Nov						
15-Nov	✓	✓	✓	✓	Kamru	
22-Nov						
29-Nov	✓	✓	✓	✓	Kamru	
6-Dec	✗					
13-Dec						

Shabuktagan