Team 33
Design Document

Air Pollution Mapping Bands

Chirag Nanda
Vedant Agrawal
Vatsin Shah

Table of Contents

1. Introduction

1.1 Problem Overview

As air pollution has increased globally, the need for pollution tracking has increased in tandem. Today, most cities take readings using satellites as well as sensors scattered around the city to collect an aggregate reading of city-wide air quality[1]. While this may give a good estimate of the air pollution over a city-wide area, the air quality of individual localities and streets may differ vastly.

Air pollution can change over the course of a day. A variety of factors including traffic, population density, the operation of office buildings, and factories can influence the air quality. A more dynamic calculation of air quality can help people decide which routes to take and which places to avoid. Some cities like Barcelona and Chicago have tried implementing IOT based air pollution trackers embedded into city-wide infrastructure to aid in this effort. Google has even tried to fit street view cars with sensors to track pollution levels[2]. Nonetheless, these devices are often extremely expensive. For instance, the sensor nodes used in Chicago cost around five thousand dollars per node.[3] Additionally, the sensors are often spread far apart, preventing accurate locality-centric/streetwise data collection of pollution.

1.1 Solution Overview

Our solution to this problem is to create a cheap wearable band and an accompanying mobile app that will continuously monitor the air quality around the user. The broader idea is to have thousands of users wear this band to help contribute to a city-wide map that everyone can access. Nevertheless, within the time constraints of the course we plan to first create a proof of concept of the band and a simple application that gives alerts to the user about their general vicinity. The app can keep a personal record of air pollutant levels of the places they visited on a map.

We aim to keep track of carbon dioxide and carbon monoxide. Additionally, since this band will be portable, it has the potential to be useful as a warning device in indoor spaces. Hence, we also wish to sense propane as it is a common flammable gas. The band can then help find poorly ventilated areas and even warn users of potential gas leaks in places like warehouses and storage rooms. For our project we only plan to build one band. However, we plan to have multiple profiles on our app to test how multiple users can update the same map with the pollution data they collect.

1.2 Visual aid

A mockup of our idea is shown in figure 1. All sizes are approximate for now. We picked 3cm for the thickness of the band because the height of the tallest component we plan to use is 2.5cm. We will try to keep the size of our final device within these approximations if possible.

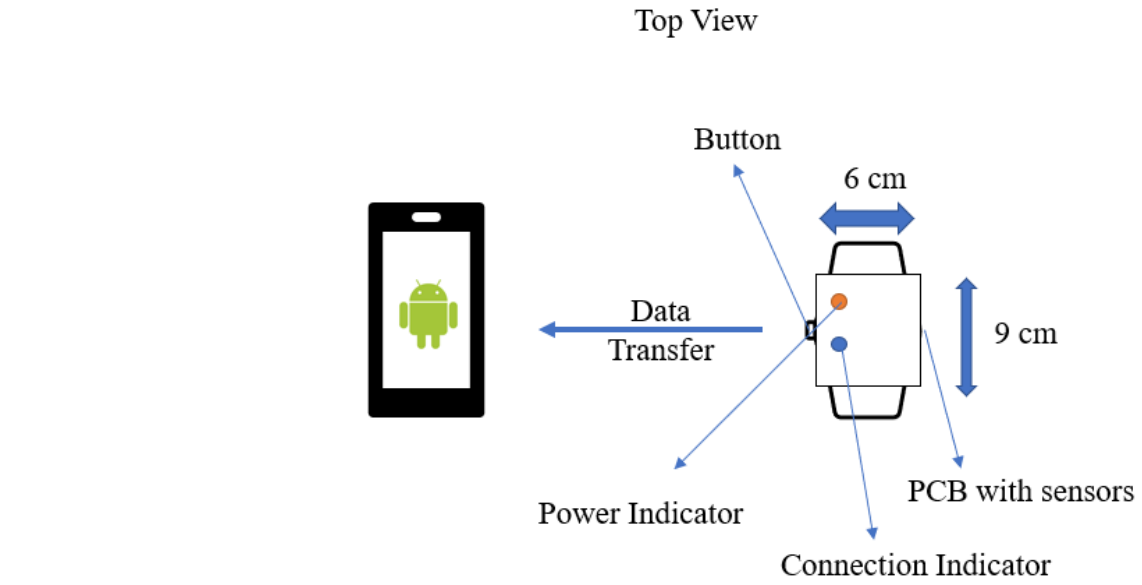


Figure 1: Top view of band

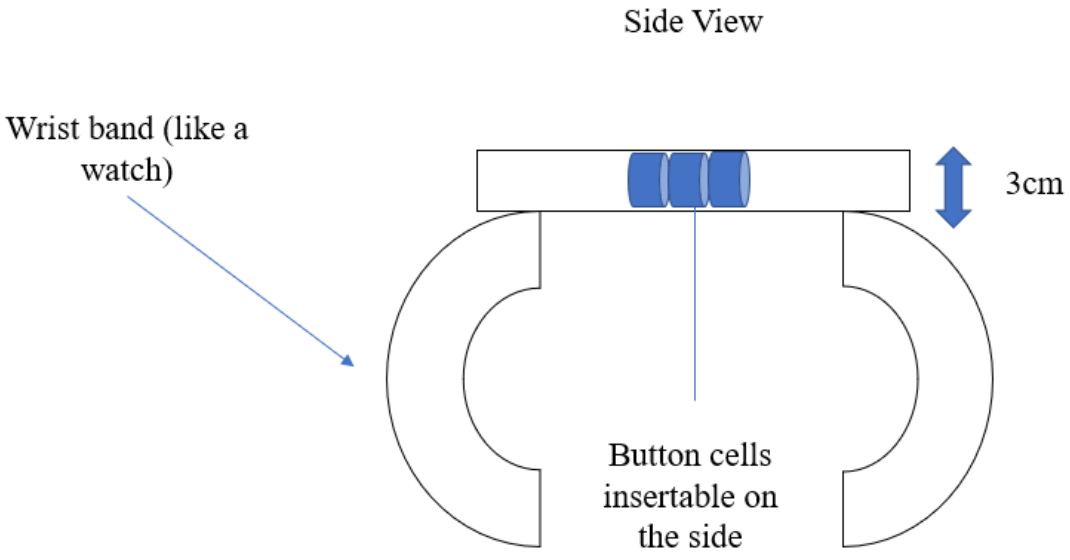


Figure 2: Side view of band

1.3 High-level Requirements

1. The propane sensor should be able to simply detect the presence of propane since any quantity of flammable gas is dangerous. The carbon monoxide sensor should be able to detect up to 200 ppm. The carbon dioxide sensor should be able to detect upto 10000 ppm. We have picked these values based on USDA determined values of dangerous exposure. [4] [5]

2. The app will need to be able to take pollution data from the band and update the map accordingly. Since the data will be a continuous stream, as a design choice we will update the app at a specific period of time as pollutant values will not be changing continuously. The app must also be able to warn the user in the form of a notification or sound if the pollution level is not safe or if propane was detected (since it is flammable).

3. Our band needs to be wearable and must have around 1-3 hours battery life to be able to track pollution data when a person makes their commute. We plan to create the housing for our circuitry using a 3D printer

2. Design

2.1 Block Diagram

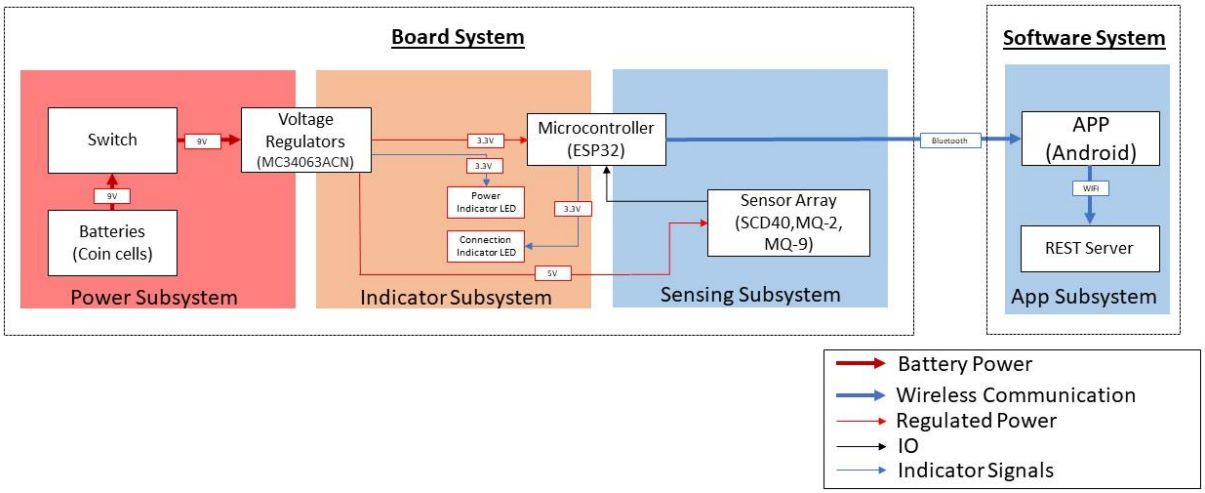


Figure 3: Block diagram of the project

Figure 3 shows the high-level block diagram for our project. Our design can be broken down into two broad categories: the board system and the software system. The board system includes the power subsystem, indicator subsystem, and sensing subsystem. The power subsystem is responsible for converting the 9V supply into 3.3V and 5V to be used by our microcontroller and sensor array respectively. The indicator subsystem relays whether or not the band is powered on as well as whether or not a device is connected to the band. The sensing subsystem is responsible for collecting and transmitting pollutant data to the connected device. The board system meets our first high level requirement by being responsible for powering our sensors and microcontroller to monitor and send pollutant data to the app. To maintain our third high level requirement, we plan to keep the design minimal and ensure that our board is as small as possible so that the band is compact and wearable. The software system meets our second high level requirement and consists of the app subsystem. The app subsystem's main goal is to visualize the pollutant data on a map and maintain a centralized map across different user profiles using a REST server.

2.2 Subsystem Description

2.2.1 App Subsystem

2.2.1.1 Overview

We intend on designing an Android application to produce human-readable values of the output of the sensors and use these values in a decision model to alert the user. The application would also serve the purpose of displaying the heat map, alerting the user if they enter a contaminated region, and an interface to interact with a server to indicate contaminated regions.
This subsystem aims to solve the second high-level requirement of interacting with the sensor data and update the shared map with those values to handle user alerts

2.2.1.2 Requirements and Verification

Requirements	Verification
Connect the phone to the bluetooth module of ESP32 MCU and periodically receive bluetooth packets and unpack them without losing any data in the app.	Compare the values sent by the microcontroller (using print statements to the serial monitor) and the values obtained by unpacking the bluetooth packets.
The app should alert the user if the carbon dioxide and carbon monoxide values go above a certain threshold. Additionally, any detection of propane gas should be notified to the user.	Send dummy test values from ESP32 MCU for each gas above and below their respective threshold values and verify that the app only notifies if these values cross the threshold. Additionally, we can also artificially simulate conditions that would cause the sensors to detect high concentration of gasses to verify the app alerts.
The app should be able to communicate with the central server to send the local gas values data (if they cross the threshold) and current GPS location.	Send dummy test values from ESP32 MCU for each gas above and below their respective threshold values and verify that the app only sends a POST request to the server if the values cross the threshold. Also, verify if the values received by the server and the GPS coordinates matches the values sent by the app. In the case where gas values for the same rounded off GPS coordinate exist on the server,
The server should maintain a ledger of all the	Send multiple dummy test values from the

values sent to it from the app when a contamination is detected. It should also round off the GPS coordinates by a predetermined degree to group the values in 100m radius together on the map to adhere to the safety regulation suggested by REVIHAAP paper[6] .	app to the server using POST requests and verify that these values are accurately recorded by the server using print statements. Also verify that the rounding off code works as intended and only rounded off GPS values are stored in the ledger. In the case where gas values for the same rounded off GPS coordinate exist on the server, verify that it updates it with the average of the existing and current value.
The app should periodically fetch the ledger data from the server to display the contaminations on a human readable map.	Inject the server with some dummy values and verify that the app automatically executes GET requests periodically. Ensure that the ledger on the server matches exactly with the local ledger after the GET request is fulfilled. Since the Google Maps API is going to be used to display the heat map, ensure that the coordinates from the ledger are accurately displayed on the map.
The app should alert the user if they enter a zone that was marked as contaminated by other users.	Inject the server with dummy test values of gas concentration that exceed the threshold values. Verify that the app rounds off the current GPS coordinates (by the same degree as done while posting the values) and checks for it in the obtained ledger. If the coordinates match up, verify that the app sends a notification.

Table 1: Requirements and verification of the app subsystem

2.2.2 Power Subsystem

2.3.1.1 Overview

This subsystem takes in battery voltage (~9V) from three 3V cells and steps it down to 5V for powering the sensors and 3.3V for powering the ESP32. The battery, after a power switch, also connects to the indicator subsystem which indicates that the device has been powered on. The power subsystem needs to step voltage down while wasting minimum possible energy in order to maintain high battery life. The ideal way to do this would be a buck converter for each voltage while keeping the same 9V input for each voltage required as less energy wastage in step down means higher battery life over time. The microcontroller operates on 3.3V and sensor array

operates on 5V, so both of those stepdowns will be required. Hence, the ESP 32 and sensors will have separate voltage regulators. At the end of each regulator there needs to be a denoise capacitor to smooth out the power supply from the regulators.

2.3.1.2 Requirements and Verification

Requirements	Verification
The 9V input must come from three 3V button cells to keep the design compact. This voltage should not drop below 5V as we use a Buck converter to step the voltage down.	The upper limit can be verified using a voltmeter and a fresh set of cells, while the lower limit can be verified using a voltmeter and a discharged set of cells.
When the band is not being used, there needs to be a switch between the battery and voltage regulators to turn off the device and conserve power	Operation of the power switch can be verified using continuity tests in a multimeter.
The sensors expect a 5V input with a tolerance of ±0.1V from the voltage regulator.	Output of the voltage regulator for the 5V input and its stability over time must be verified using an oscilloscope.
ESP32 expects input of 3.3V but can accept between 2.2 to 3.6V from the voltage regulator.	Output of the voltage regulator for the 3.3V input and its stability over time must be verified using an oscilloscope.

Table 2: Requirements and verification of the Power subsystem

2.2.2 Indicator Subsystem

2.2.2.1 Overview

The purpose of this subsystem is to indicate whether the band is powered up, if the system is ready to be connected to, or if some device is already connected to the system. It consists of two LEDs, one for indicating power and one for indicating the status of connection. The power indication LED will be directly connected to the same voltage regulator used by the sensor array. The connection LED will be multicolored and powered by the microcontroller. Depending on the connection status of the band, the connection LED will display a different color.

2.2.2.2 Requirements and Verification

Requirements	Verification
The power indication LED needs to turn on when the device is switched on.	Potential difference across switch's end and the state of the LED can be monitored simultaneously to verify the working of the power indicator LED.
The connection indication LED needs to turn on when the microcontroller finishes setup. By default, the LED needs to shine red upon turning on the band.	Potential difference across switch's end, microcontroller, and the state of the LED can be monitored simultaneously to verify the working of the connection indicator LED once the band is turned on.
The connection indication LED needs to change to a different color depending on the state of connection. If a mobile device is not paired with the band, it should shine red. Once a device pairs with the band it should shine green.	Microcontroller's state (using print statements to the serial monitor) and the state of the LED can be monitored simultaneously to verify the working of the state indicator LED. The color should change as soon as a device is connected to the microcontroller.

Table 3: Requirements and verification of the Indicator subsystem

2.2.3 Sensing Subsystem

2.2.3.1 Overview

The sensing subsystem involves the microcontroller (ESP32) and an array of sensors including:

a) MQ-2 Semiconductor Sensor for Combustible Gas like propane[7]
b) MQ-9 Semiconductor Sensor for Carbon Monoxide[8]
c) Semiron's SCD41 Carbon Dioxide Sensor[9]

All the sensors mentioned above use 5V power each. The purpose of this subsystem is to measure pollution levels and send the data over to the connected phone using bluetooth or WiFi. The Carbon Monoxide and Propane sensors connect to the ESP32[10] via the analog input pins available on the microcontroller. The Carbon Dioxide sensor is a digital sensor that connects via I2C communication protocol. This analog data will be converted to ppm values via the conversion graphs for each sensor (figures 4 and 5). The microcontroller will send the recorded PPM values to the app over the chosen wireless transfer protocol every 5 minutes or whenever a dangerous amount of any gas is detected. The decision to send every 5 minutes is based on the fact that air quality readings do change rapidly unless there is a gas leak. This subsystem specifically meets our first high level requirement. Additionally, these sensors are 2.5cm in

height including connection pins, so they have a low profile, allowing us to fit them in a bracelet (to meet high level requirement 3).

2.2.3.2 Requirements and Verification

Requirements	Verification
Read analog data from MQ-2 and MQ-9 sensors into digital data using analog input pins on ESP 32.	Read voltage output using an oscilloscope and convert the voltage reading to digital reading mathematically. Compare this value to the value read by the microcontroller using print statements.
Convert sensor data into PPM measurements.	Convert voltage readings to PPM readings using the conversion graphs for each sensor. This can be done by linearizing the graph in 100 ppm segments so that we can map voltage values to approximate PPM value using the appropriate load resistors mentioned in the datasheets. See figure 4 for MQ 9 and figure 5 for MQ 2 sensors.
Establish successful I2C connection between ESP 32 and SCD 41 to read data from the sensor.	Send wakeup requests and receive confirmation to make sure we can communicate with the sensor.
Establish successful bluetooth connection between ESP 32 and app to reliably send data over.	Send dummy packets over bluetooth using the appropriate protocol to ensure we can receive data on the app.
Ensure that the sensor data is sent to the app every 5 minutes	Use a timer on ESP 32 to interrupt the normal flow of code to send collected data over bluetooth and verify on app that the packets are received every five minutes.

Table 4: Requirements and verification of the Sensor subsystem

2.2.3.3 Sensor Plots

Figure 4: Voltage vs PPM (MQ-9)[8]

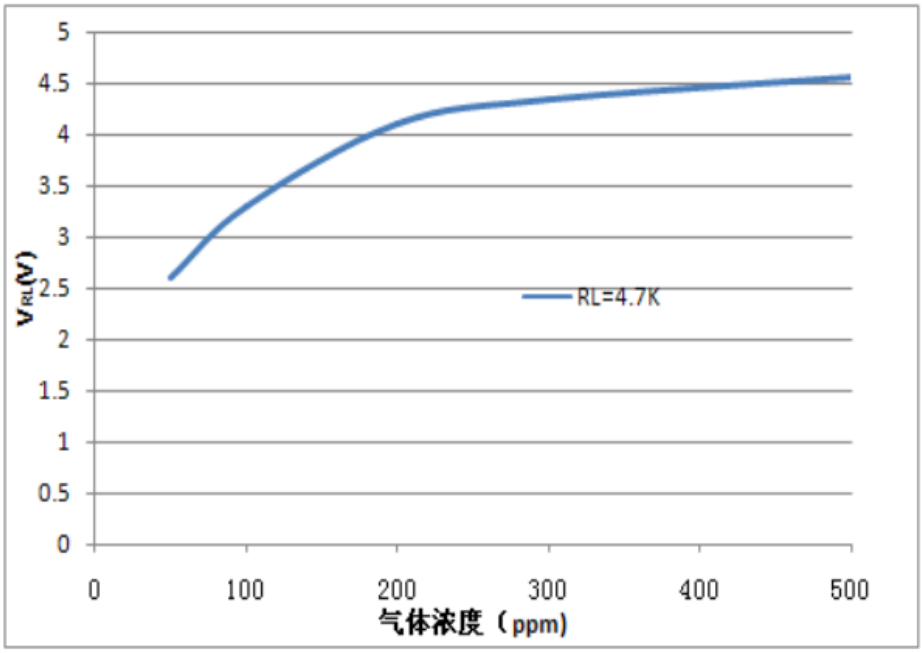
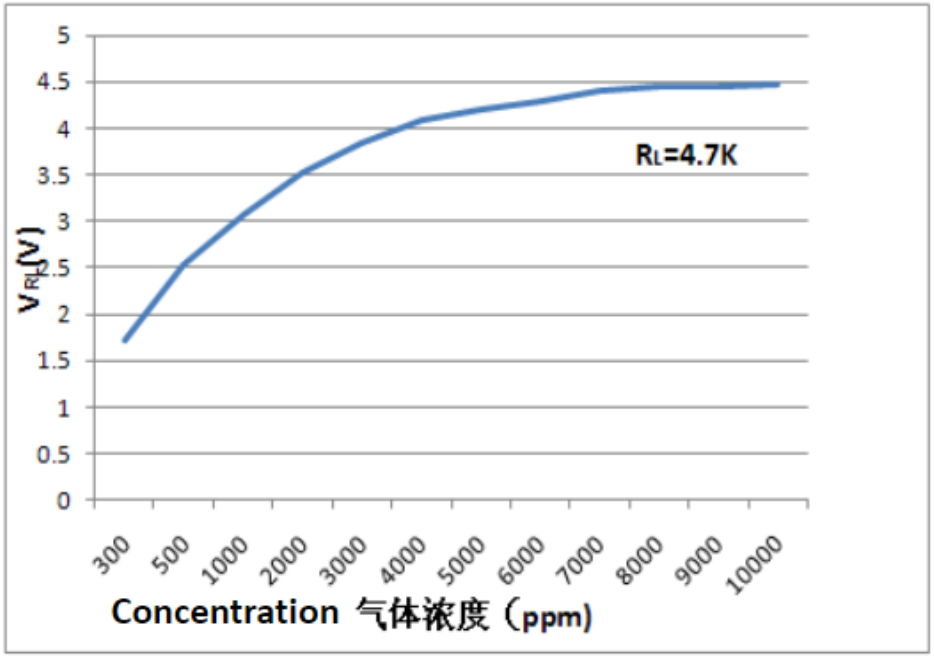


Figure 5: Voltage vs PPM (MQ-2)[7]



2.4 Tolerance Analysis

For each sensor we have calculated the approximate error using graphs from their datasheet that catalog the relative readings based on various temperatures and humidity. Since we will be testing our bands in the spring in Champaign, we have assumed that the temperature will be between 15-25°C and the humidity will be around 55%[11]. To calculate the percentage error, we make use of the following formula:

$$\boldsymbol{\delta} = \left|\frac{(measured\ value - absolute\ value)}{absolute\ value}\right| * 100$$

1) **MQ-2:**

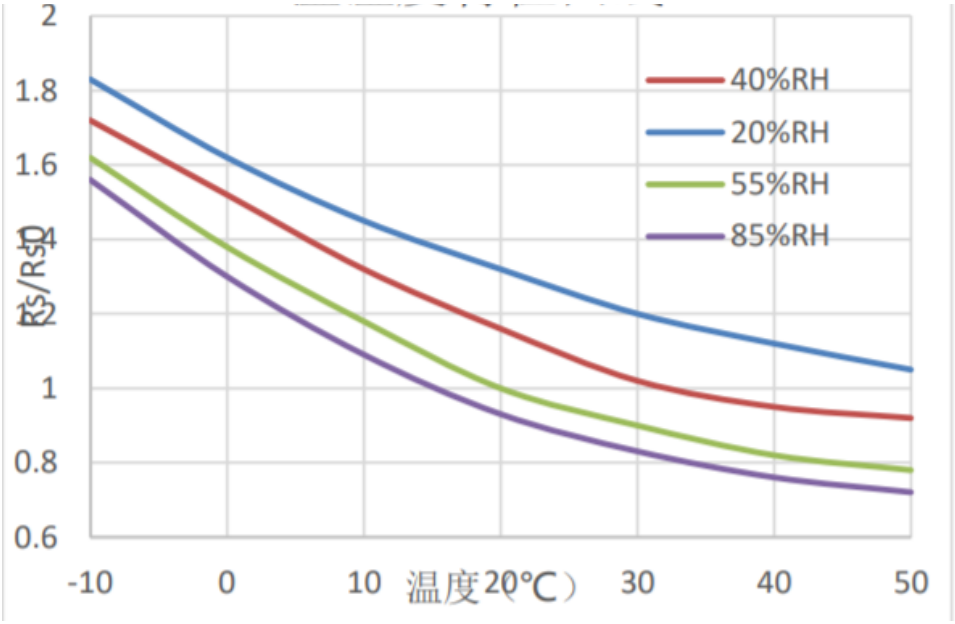


Figure 6: Graph of MQ-2's relative temperature/humidity characteristics[7]

The y-axis in figure 6 represents the ratio of Rs/Rso and the x-axis represents temperature. Rs (the measured value) is the resistance of the sensor in 2000ppm of propane in various temperatures and pressures. Rso (the absolute value) is the resistance of the sensor in 2000ppm propane under 20°C/55% relative humidity. Relative Humidity can be assumed to be 55% (the green curve) and operating temperature is 15-25°C.

Relative error at 15°C:

$$\left|\frac{R_s - R_{so}}{R_{so}}\right| * 100$$

$$= \left(\left|\frac{R_s}{R_{so}} - 1\right|\right) * 100$$

$$= (1.1\text{-}1)*100$$

$$= \mathbf{10\%}$$

Relative error at 20°C:

$$\left|\frac{R_s - R_{so}}{R_{so}}\right| * 100$$

$$= \left(\left|\frac{R_s}{R_{so}} - 1\right|\right) * 100$$

$$= (|1 - 1|)*100$$

$$= \mathbf{0\%}$$

Relative error at 25°C:

$$\left|\frac{R_s - R_{so}}{R_{so}}\right| * 100$$

$$= \left(\left|\frac{R_s}{R_{so}} - 1\right|\right) * 100$$

$$= (|0.95 - 1|)*100$$

$$= \mathbf{5\%}$$

2) MQ-9:

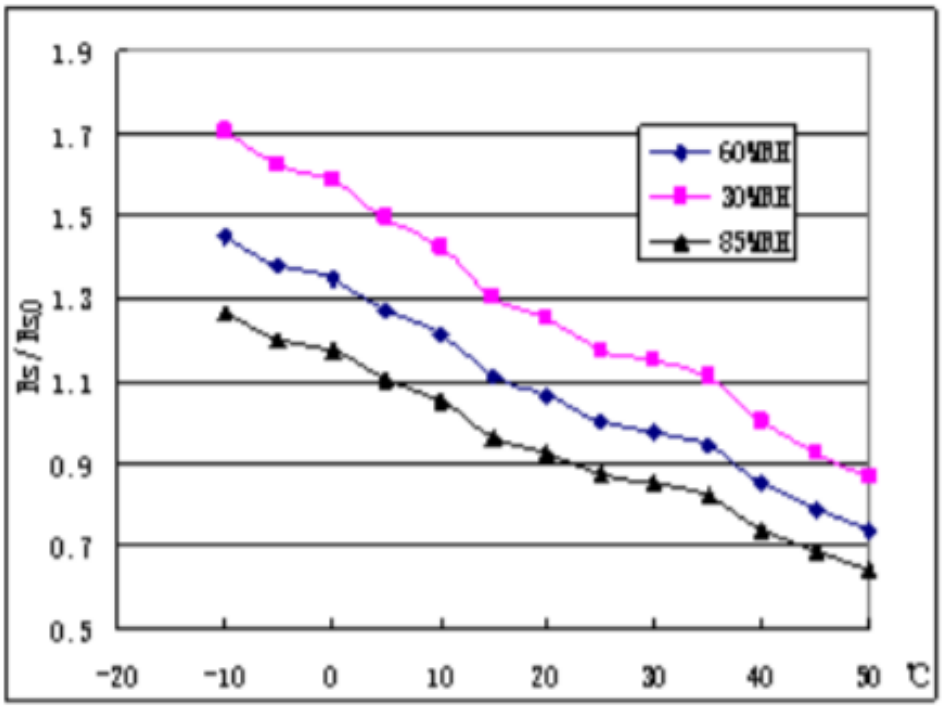


Figure 7: Graph of MQ-9's relative temperature/humidity characteristics[8]

The y-axis in figure 7 represents the ratio of Rs/Rso and the x-axis represents temperature. Rs (the measured value) is the resistance of the sensor in 150ppm of CO in various temperatures and pressures. Rso (the absolute value) is the resistance of the sensor in 150ppm CO under 20°C/55% relative humidity. Relative Humidity can be assumed to be 55% (the green curve) and operating temperature is 15-25°C. Relative Humidity can be assumed to be 60% (closest to 55%) (the blue curve) and operating temperature is 15-25°C.

Relative error at 15°C:

$$\left|\frac{R_s - R_{so}}{R_{so}}\right| * 100$$

$$= \left(\left|\frac{R_s}{R_{so}} - 1\right|\right) * 100$$

$$= (|1.1 - 1|)*100$$

$$= \mathbf{10\%}$$

Relative error at 20°C:

$$\left|\frac{R_s - R_{so}}{R_{so}}\right| * 100$$

$$= \left(\left|\frac{R_s}{R_{so}} - 1\right|\right) * 100$$

$$= (|1.07 - 1|)*100$$

$$= \mathbf{7\%}$$

Relative error at 25°C:

$$\left|\frac{R_s - R_{so}}{R_{so}}\right| * 100$$

$$= \left(\left|\frac{R_s}{R_{so}} - 1\right|\right) * 100$$

$$= (|1 - 1|)*100$$

$$= \mathbf{0\%}$$

3) **SCD40:**
   At high accuracy, ±(40ppm + 5%) should be expected error. The range of high accuracy measurements is 400 - 2000ppm. At higher ppm, the inaccuracies increase, though the datasheet does not quantify the exact amount[9].

We have also highlighted possible faults in our system and our proposed solutions to them:

Pain point: Due to some external or internal factors, the sensors can occasionally transmit faulty values. These faulty values can severely affect the accuracy of our data.

Solution: We will try to mitigate this issue by maintaining a running average of the collected values and pushing this average to the server. By maintaining a running average, we can smooth out the inaccurate spikes in our data.

Pain point: Faulty bracelets will always be transmitting inaccurate data to the server which affect the accuracy of the data for other users.

Solution: We will try to minimize the effect of faulty bracelets by averaging the received value with the current value on the server side. The hope is that with enough non-faulty functioning bracelets in the region, the error can be offset easily as the average would be closer to the accurate values.

Pain point: The power expected by sensors and the microcontroller is very precise. Sensors expect a 5V input with a tolerance of ±0.1V and the ESP32 expects input of 3.3V but can accept 2.2 to 3.6V.

Solution: Using buck converters, which have a feedback loop each, we should be able to precisely control the voltage step down from ~9V from the battery to the desired 5V and 3.3V irrespective of the load.

3. Cost and Schedule

3.1 Cost analysis

The cost of the project can be divided into the labor cost and parts cost.

3.1.1 Labor Cost

We assume that we will spend an average of 10 hours a week for a total of 10 weeks of this semester working on our senior project. Additionally, we are taking $32 as our per hour wage because that is the average salary of ECE interns in Illinois[12]. Using these numbers the total labor cost is:

$$Labor\ Cost\ = \$/hour\ *\ 2.5\ *\ hours\ to\ complete\ *\ people$$
$$Labor\ Cost\ = \$32/hour\ *\ 2.5\ *\ 10 hours/week\ *\ 10\ weeks\ *\ 3\ people$$
$$Labor\ Cost\ = \$24000$$

3.1.2 Parts Cost

Aside from the exact costs for required parts, we are also including a row for miscellaneous costs for parts that we may already have from lab kits of previous courses including resistors, capacitors, and push button switches. Additionally, as a fail safe, we plan to buy an excess of specific parts which we are not confident about using including the CO sensor, propane sensor, and buck converter.

Part Name	Part Number	Quantity	Vendor	Cost	Total
CO2 Sensor	SCD41	1	Sparkfun	$60.00	$60.00
CO Sensor	MQ9	2	Sparkfun	$4.95	$9.90
Propane Sensor	MQ2	2	Sparkfun	$5.95	$11.90
ESP32	ESP32 C3 WROOM	1	Sparkfun	$3.50	$3.50
Buck Converter	LMR36015FBRNXT	6	Mouser	$2.83	$16.98
3V button cell	CR2032	10	Amazon	$0.80	$8.00
LED	1528-2761-ND	2	Digikey	$5.95	$11.90
Miscellaneous Costs					$5.00
Total					**$127.18**

Table 5: Cost of parts

3.1.3 Total Cost

$$Total\ Cost\ = Labor\ Cost\ +\ Parts\ Cost$$
$$Total\ Cost\ = \$24000\ +\ \$127.18\ = \$24127.18$$

3.2 Schedule

Week	Chirag Nanda	Vatsin Shah	Vedant Agrawal
2/7	Work on Project proposal and talk to machine shop		
2/14	Work on design document		
2/21	Complete circuit schematic for power subsystem. Finalize and order parts for the power subsystem	Complete circuit schematic for sensor subsystem. Finalize and order parts for the sensor subsystem.	Complete circuit schematic for indication subsystem. Finalize and order parts for the indication subsystem
2/28	Finalize pcb design for power subsystem	Finalize pcb design for sensor subsystem	Finalize pcb design for indication subsystem
3/7			
3/21			
3/28			
4/4			
4/11			
4/18	Prepare for mock demo Finalize and verify all components		
4/25	Prepare for final presentation Work on the final report		

Table 6: Planned weekly schedule

4. Ethics and Safety

4.1 Ethical Concerns

Since we will be tracking location data of the user, it could be a possible violation of IEEE[12] and ACM[13] privacy standards. To ensure privacy, we will log the location data but keep the user anonymous on our map. Our app will only associate pollutant data to the user's location and no trace of the user's identity will be recorded. Additionally, we only start using the user's gps location after appropriate in-app permissions are given.

4.2 Safety Concerns

The biggest safety concern during the development of our bands lies in testing the bands for detection of harmful levels of gasses. To test carbon dioxide and carbon monoxide levels we plan on positioning the sensor at different distances from a lit flame (using a candle or bunsen burner). To ensure safety while doing these tests, we will only work in a well ventilated laboratory with a fire extinguisher. However, for testing propane detection we plan to purchase a propane tank. This is a safety issue as propane is flammable. Hence we will be testing with the tank only outdoors and ensure proper storage of the tank when we are not testing our project.

Since we are making a wearable band, we also need to make sure that all the circuitry is well insulated. We will create a proper housing for the pcb and batteries to ensure that no wires or circuitry is exposed. To create this enclosure we will make use of the machine shop or 3D print a suitable structure for our circuitry.

References

1. “How Is Air Quality Measured?” NOAA SciJinks – All About Weather, https://scijinks.gov/air-quality/. Accessed 7 Feb. 2022.

2. “Some Google Street View Cars Now Track Pollution Levels | Colorado Public Radio.” Colorado Public Radio, Colorado Public Radio, 30 July 2015, https://www.cpr.org/2015/07/30/some-google-street-view-cars-now-track-pollution-levels/.

3. “How Cities Are Using the Internet of Things to Map Air Quality .” Data-Smart City Solutions, Harvard, https://datasmart.ash.harvard.edu/news/article/how-cities-are-using-the-internet-of-things-to-map-air-quality-1025. Accessed 7 Feb. 2022.

4. “Carbon Dioxide Health Hazard Information Sheet .” FSIS USDA, USDA, https://www.fsis.usda.gov/sites/default/files/media_file/2020-08/Carbon-Dioxide.pdf. Accessed 8 Feb. 2022

5. “What Are the Carbon Monoxide Levels That Will Sound the Alarm?” Kidde, 19 Aug. 2019, https://www.kidde.com/home-safety/en/us/support/help-center/browse-articles/articles/what_are_the_carbon_monoxide_levels_that_will_sound_the_alarm_.html.

6. “Proximity to Roads, NO2, Other Air Pollutants and Their Mixtures - Review of Evidence on Health Aspects of Air Pollution – REVIHAAP Project - NCBI Bookshelf.” National Center for Biotechnology Information, https://www.ncbi.nlm.nih.gov/books/NBK361807/. Accessed 9 Feb. 2022.

7. “MQ-2 Semiconductor Sensor for Combustible Gas .” Sparkfun, https://cdn.sparkfun.com/assets/3/b/0/6/d/MQ-2.pdf. Accessed 9 Feb. 2022.

8. “MQ-9 Semiconductor Sensor for CO/Combustible Gas.” Sparkfun, https://cdn.sparkfun.com/assets/d/f/5/e/2/MQ-9B_Ver1.4__-_Manual.pdf. Accessed 9 Feb. 2022.

9. “Sensors SCD4x Datasheet”. Sparkfun, https://cdn.sparkfun.com/assets/d/4/9/a/d/Sensirion_CO2_Sensors_SCD4x_Datasheet.pdf. Accessed 9 Feb. 2022.

10. “ESP 32 Series Datasheet.” Digikey, https://www.digikey.com/htmldatasheets/production/2845213/0/0/1/esp32-datasheet.html?utm_adgroup=xGeneral&utm_source=google&utm_medium=cpc&utm_campaign=Dynamic%20Search_EN_Product&utm_term=&utm_content=xGeneral&gclid=CjwKCAiA6Y2QBhAtEiwAGHybPRh4PBaR9pZXi6. Accessed 9 Feb. 2022.

11. “Weather in Champaign.” Champion Traveler, https://championtraveler.com/dates/best-time-to-visit-champaign-il-us/. Accessed 10 Feb. 2022.

12. “Average Electrical Engineer Internship Salary 2022: Hourly and Annual Salaries.” Zippia - Find Jobs, Salaries, Companies, Resume Help, Career Paths and More, 18 May 2020, https://www.zippia.com/electrical-engineer-internship-jobs/salary/.

13. “IEEE - IEEE Code of Ethics.” IEEE - The World’s Largest Technical Professional Organization Dedicated to Advancing Technology for the Benefit of Humanity., https://www.ieee.org/about/corporate/governance/p7-8.html. Accessed 9 Feb. 2022.

14. “Code of Ethics.” Association for Computing Machinery, https://www.acm.org/code-of-ethics. Accessed 9 Feb. 2022.