# Indoor Air Quality Capstone

***Team 1***


Portland State University

Maseeh College of Engineering & Computer Science

Department of Electrical & Computer Engineering

*Authors:*

Adam Dezay

Manuel Garcia

Brandon Hippe

Mercedes Newton

*Industry Sponsor:*

Dr. David Burnett

*Faculty Advisor:*

Dr. John Acken



Senior Project Developement

Portland Oregon

# Contents

# List of Figures

# List of Tables

# Executive Summary

Air quality is very important for safe working conditions. In every indoor space there should be a monitoring system for environmental risks and air pollutants such as CO2 and fine particulate matter (PM2.5), as well as ventilation rates. Our team's goal was to build a device to monitor elevated or dangerous quantities of these pollutants using as many commercial off-shelf components and open-source software as possible. Our aim was to create 3 to 10 wireless, battery powered initial prototypes that each have at least one year life span.

This project is sponsored by the Wireless Environmental Sensor Technologies (WEST) Lab in the Electrical and Computer Engineering Department at Portland State, run by Dr. David Burnett. The project's faculty advisor is Dr. John Acken.

This Project will be improving on the project from spring of 2022 to report readings of carbon dioxide and particulate as well as airflow instead N2O in an indoor environment instead of an outdoor environment.

The team was successful in the project. We completed the project with 4 working nodes that communicate with each other using SmartMesh IP. The units can achieve a battery life of a year with reasonable capture times. Each unit is also equipped with the SGP30 CO2 sensor and SPS30 PM2.5 sensor to collect data. The system is housed in a cut acrylic box that can be mounted on walls or ceilings.

This report details the background of this project, project requirements, and our approach to tackling this project. We also included appendixes that contain the operation manual, issues we faced and solutions, and the components and software we used throughout the project.

# Background

Our team wanted to make it easy to continuously monitor an indoor environment, and report data back to a host. Our aim was to make monitoring easy by making cheap, reliable devices that do not require frequent recharging. Three features of indoor air quality are expected to be monitored: 2.5 m particle count or PM2.5, carbon dioxide (CO2) concentration, and ventilation air speed.

Regarding CO2, statistically significant decrements occurred in cognitive performance (decision making, problem resolution) starting at 1000 ppm. CO2 concentration is also a good proxy for ventilation; high CO2 levels mean the room is poorly ventilated, which increases the risk of passing airborne diseases such as COVID19.

Regarding PM2.5, the WHO recommends an upper limit of 5 µg/m3 (microgram per cubic meter) average annually and 15 µg/m3 average over a 24 hour period .

An air speed sensor, or anemometer, can help us calculate how much air is flowing into or out of a room and help understand why CO2 and/or PM2.5 is high.

The system is based on components past capstone teams have successfully incorporated such as the TI MSP430 (selected for its particularly low-power sleep modes). In this iteration, a goal was to replace the closed-source proprietary SmartMesh IP wireless system with the OpenWSN open-source wireless networking system.

# 1 Requirements

The sensor system must:

- Sense PM2.5 and $CO_2$ often enough and accurately enough to ascertain indoor air quality relevant to occupants
- Include at least one node capable of sensing airspeed
- Maximize its battery life for sustained operation
- Locally store its measurement data
- Use as many commercial off-shelf components as possible
- Have an enclosure
- Wirelessly share its measurement data with a central monitoring system with communication range of at least 10 meters
- Utilize SmartMesh IP
- Have 3 iterations that cost no more than $1,000 total

The sensor system should:

- Include capability for any node to sense airspeed
- Have a battery life of at least 1 year
- Be open-source to the extent possible when using commercial off-the-shelf components
- Upgrade SmartMesh IP to Utilize a low-power Wireless Sensor Network (WSN) system like OpenWSN
- Utilize Texas Instruments MSP430/432 class microcontroller unit
- Have 10 iterations that cost no more than $3,000 total

- Utilize 18560 lithium ion battery cell/s

The sensor system may:

- Source all power from a single 18650 battery
- Monitor other environmental conditions such as temperature and humidity
- Include self-configuration capability to detect which sensors are connected and adjust sampling rates accordingly
- Be usable outdoors
- Include a visualization dashboard to monitor the data graphically
- Be able to incorporate many more (greater than 10) sensor modules
- Match lifetime of a fire detector (approximately 10 years)

# 2 Stakeholders and Preexisting Design

## 2.1 Stakeholders

Industry Sponsor : Dr. David Burnett (Principal Investigator of WEST Lab at Portland State University)
Faculty Advisor: Dr. John Acken
Engineers:

- Adam Dezay
- Manuel Garcia
- Brandon Hippe
- Mercedes Newton

Customer: Any business or person in need of monitoring changing air quality conditions.

## 2.2 Preexisting Design

This project used some of the research done in last year's capstone project. The final report from last year's project can be found under the documentation folder in the team's Github repository. The file is titled "Team 13 - Final Report - SensorSuit - 2.pdf". Link to the Github can be found in the project resources section.

**Figure 1:** Previous Teams Project

Even though last year's project measured temperature, humidity, and N2O outdoors, both projects share the same controller, the MSP430. They both also use SmartMesh IP as a way to connect multiple nodes.

We faced many challenges with the microcontroller as well as SmartMesh that last year's team did not experience. Check Appendix A to see the solutions to unexpected problems we encountered that the previous team did not.

# 3 Objective and Deliverables

The following represent deliverables our team identified as significant to doccument the project:

- Complete documentation:
  - Project proposal
  - Weekly Progress Reports
  - Final report
  - ECE Capstone Poster Session poster
- Summary project report in the style of an IEEE conference paper
- Bill of materials
- System schematic and functional diagram
- Design files for any custom mechanical components or PCBs
- Source code
- Operation guide
- Demonstration of hardware
- Three to ten sensor system prototypes

# 4 Design

When selecting sensors for this project, the team had to select PM2.5 and $CO_2$ sensors that were energy efficient, relatively accurate for our setting, and fit within a budget of $3,000 for all 10 possible nodes.

When selecting the $CO_2$ sensor, the team considered selecting a true $CO_2$ sensor like the SCD30 which would produce very accurate readings yet was more expensive than an ECO2 (estimated $CO_2$) sensor which would prove to be less accurate due to its nature of estimating $CO_2$ based on ethanol and H2 levels in the air. Our team ultimately chose to go with the SGP30 (estimated $CO_2$) sensor as we are merely sensing elevated levels of carbon dioxide and therefore do not need to get incredibly precise readings. Anything over 1K PPM would simply be very high and might constitute an evacuation of room or need for more airflow.

| Sensor | SGP 30 | SCD 30 |
|---|---|---|
| Type | E $CO_2$ | $C0_2$ |
| Additional sensors | VOC | temperature and humidity |
| Range | 400-60k PPM | 400-10k PPM |
| Accuracy | +- 15% | +-30PPM +3% |
| Price | 17.50 | 58.95 |

**Table 1:** $CO_2$ Sensor Finalists

## 4.1 PM2.5 Sensor

For the PM2.5 sensor, the team decided to go with the SPS30 as it was the least power hungry among our options at the time. It was also the most readily available, and was sourced from Sensirion like the SPG30 which would cut costs if this product were to be mass produced. The PM2.5 Sensor Finalists table below shows the

specific trade offs our team considered when selecting this sensor.

| Sensor | PMS5003 | PAS-IN-01 | SPS30 |
|---|---|---|---|
| Range (concentration) | 0-500 ug/m^3 | 0-1500 ug/m^3 | 0-1000 ug/m^3 |
| Accuracy | ±10% or ±10 ug/m3, whichever is greater | ±10% or ±10 ug/m3, whichever is greater | ±10% or ±10 ug/m3, whichever is greater |
| Power Consumption (on) | ∼500mW | ∼350mW | ∼275mW |
| Price | $39.95 | $19 | $49.53 |
| wait time for accurate data | 30 seconds | 15 seconds | 10 seconds |

**Table 2:** PM2.5 Sensor Finalists

## 4.2 Anemometer

When Selecting an anemometer, we originally preferred the concept of designing and building an ultrasonic anemometer as it was significantly more energy efficient than hot wire anemometer options. Due to issues with building a custom sensor as well as difficulties with procuring parts in a speedy manner, we decided to pivot to a hot wire anemometer. This selection produced a drastic detriment to our anticipated battery life. We originally calculated that we could complete the project with just 1 battery, however we needed to add 3 more batteries to achieve the same battery life of a year with the substitution of a hot wire anemometer. To accommodate for the increased temperature, we also had to give the part extra space and keep it as far away from our other sensors as possible. We suggest future builds expand upon our ultrasonic anemometer research, implementation and documentation in an attempt to prevent the use of hot wire anemometers in this setting.

| Sensor | Custom Ultrasonic Anemometer | PAS-OUT-01Wind Sensor Rev. C |
|---|---|---|
| Range | Unknown and size dependent | 0-60 MPH |
| Power consumption (on) | ∼14.52mW | ∼150mW |
| Accuracy | Unknown and size dependent | Can detect "small puff of air at a distance of 18-24 inches" |
| Price | $7.90 | $21.95 |
| Relative size | ∼25-36 cm apart | .68" x 1.59" x .25" |

**Table 3:** Anemometer Sensor Finalists

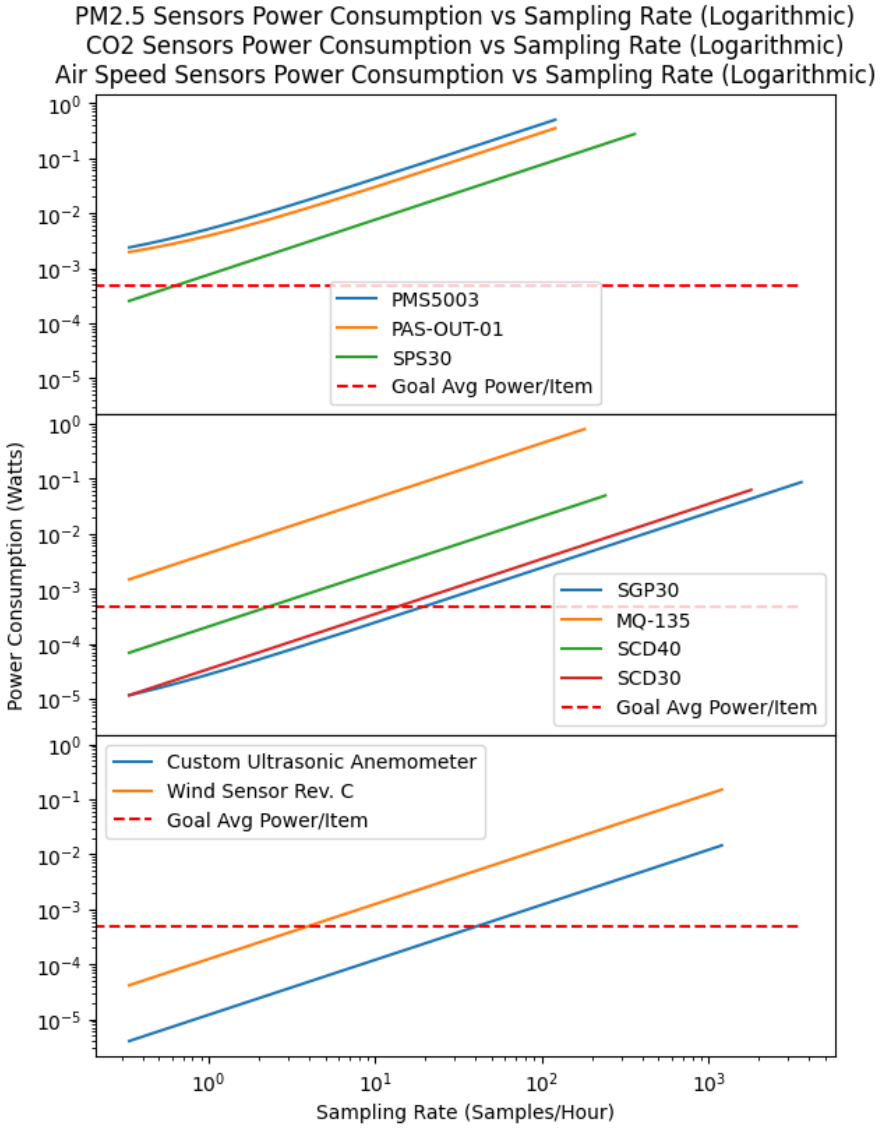


**Figure 2:** Power consumption measurements with sensor options

In Figure 2, the team created a python script to calculate the power consumption based on listed values from the specification sheets of the sensors. The python script can be found under the power testing folder in the documentation folder. The python

script is titled PowerData.py. The link for the team Github page can be found in the project resources section. The red line shows the sampling rate needed to hit 1 year of battery life based on the estimates.

## 4.3 Enclosure

When deciding on our enclosure. We followed the SWaP-C model which focuses on size, weight, power, and cost. For our first iteration, we built a system utilizing Portland State University's 3D printing. The 3D printer at the time only offered SLA. The helpers at the Electronics Prototyping Lab (EPL) recommended we switch to laser printing acrylic as it is much cheaper, faster, and is clear. To cut the enclosure, we used MakerCase(https://www.makercase.com//) as recommended by the EPL. The laser cutter utilized at the time of project was the QD-1390.

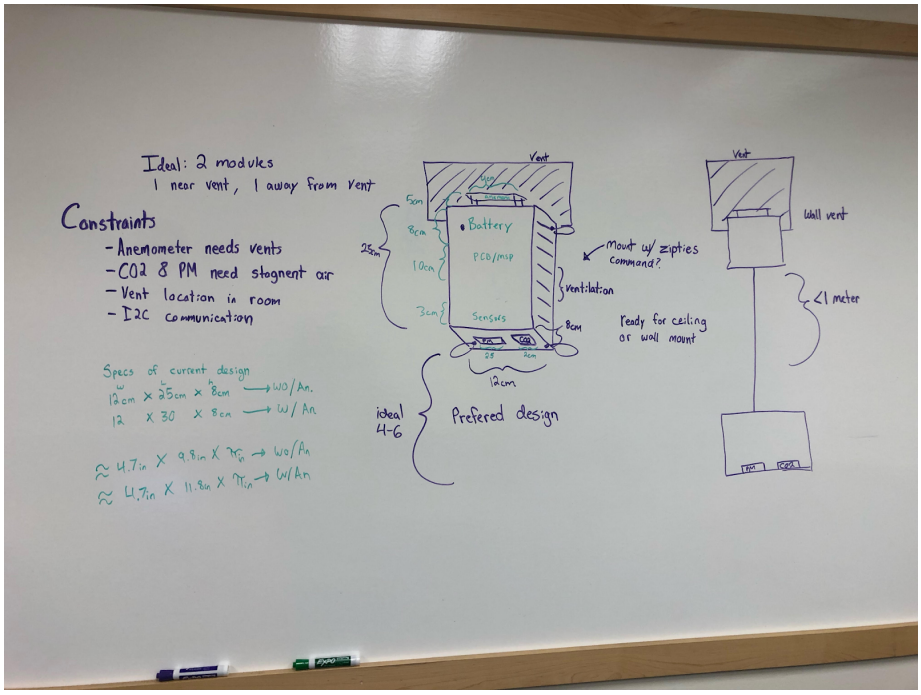


**Figure 3:** Original concepts for enclosure

When designing the enclosure, we originally came up with the two ideas shown above. The rightmost diagram represented 2 enclosures separated by an I2C cable. This would allow us to get accurate readings for $CO_2$ and PM2.5 in the bottom enclosure, and accurate anemometer readings in the above enclosure. Having two nodes seperated by an I2C cable would allow for the anemometer to be placed near a vent while the $CO_2$ and PM2.5 sensors could be placed in the desired range for accurate readings. This design proved to be infeasible because we were limited by the maximum 1 meter length for an I2C cable. We switched to the design on the left focusing on increased ventilation and allowing for proper separation of parts to ensure no damage is caused by the batteries. We also aligned the sensors to face downwards while the anemometer could face sideways towards the vent. Final dimensions came out to be 12 X 25 X 8 cm without an anemometer and 12 X 30 X 8 cm with an anemometer. This design requires several modules to be placed in one room, with an anemometer node being placed in front of an air vent while a second node without an anemometer could be placed in a stagnent area of the room. This accounts for

both air entering the room as well as an accurate measurement of the air quality in the room. The non anemometer node should ideally be placed four to six feet above the ground as according the Environmental Protection Agency (EPA) for the most accurate $CO_2$ and PM2.5 readings

## 4.4 System Design

At the highest level view our system design started very simple. We have our microcontroller that communicates and receives data from all of our sensors, we have that microcontroller communicate with smartmesh to send out our measured data, and lastly we have a battery powering it all. We ended up having to make this system more complicated in order to extend our battery life and make the system generally more robustly. We added pmos transistors in order to cut off power and put our sensors in a deeper sleep then their internal circuits would allow. We added nmos transistors in order to perform logic level conversion for our 5V sensors. We added 3.3V and 5V power conversion in order to better supply off of our 3.7V batteries. Lastly, we added further functionality to our circuit with hardware system low battery warnings. Combined we now have a a fair amount of added resistors, voltage dividers, mosfets, and breakouts that allow for a more user friendly experience.

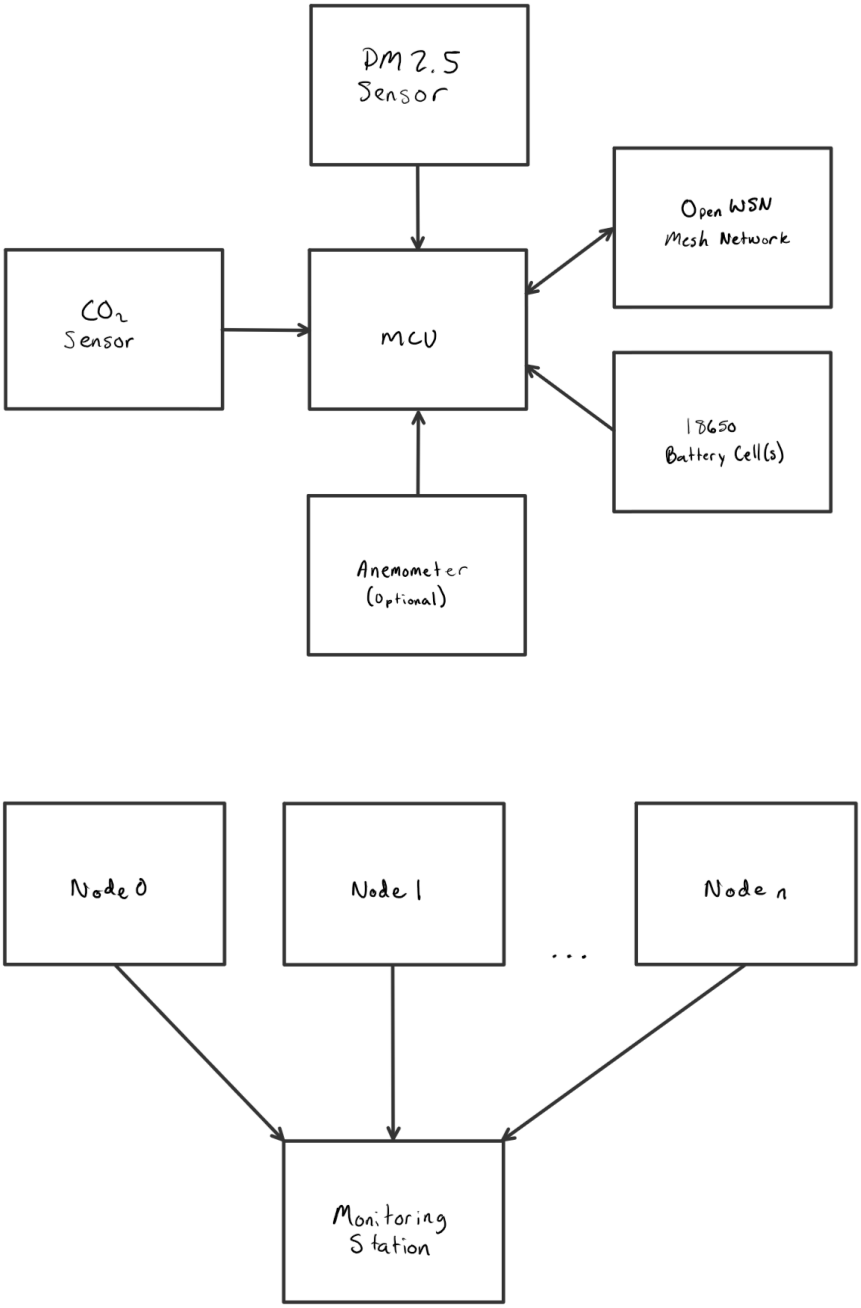


**Figure 4:** Block diagram

## 4.5 PCB Design

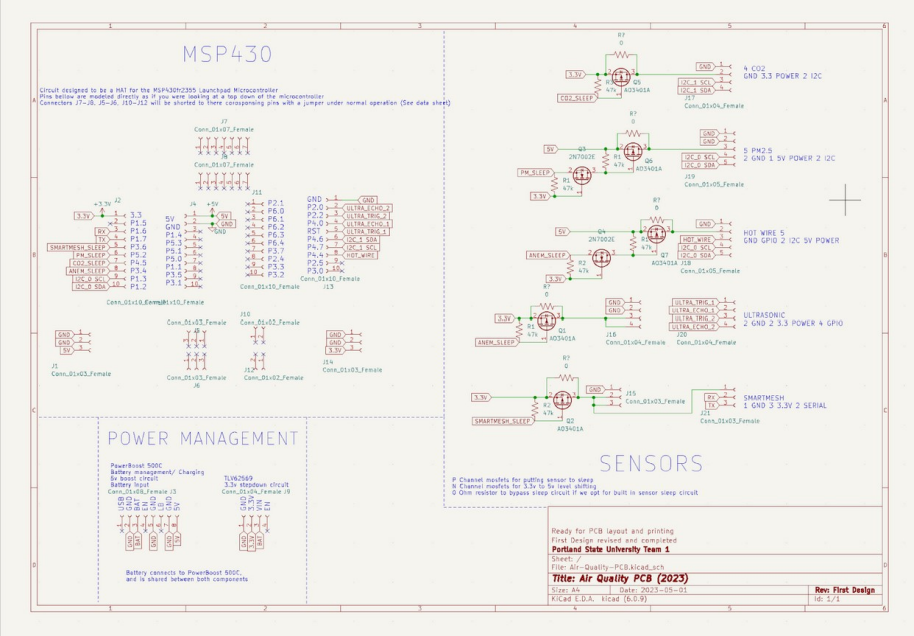


**Figure 5:** PCB Design

For our PCB we decided that we wanted our system to be as compact and easy to use as possible, with no mistakes on how or where things plugged in. In order to make this a reality we made a "Hat" For the msp430, or in simpler terms a PCB that is placed directly on top of the msp430 pins. This was a challenge because we were not able to find standard measurements for the MSP430fr2355 microcontroller, and had to take 15 or so measurements using digital calipers.

Designing a hat as our board design created unique opportunities that allowed us to streamline and simplify connections, and ended up leading to a cleaner looking final design. We were able to position our power management as close to the microcontoller power ports as possible, and place our signal wires extremely close to where the signals originated from, limiting potential losses that we might have had with longer or messier signal routing.

## 4.6 Scheduling

The team Created a followed the following Gantt charts to reflect the schedule for the project beginning in December of 2022 and concluding in June of 2023.

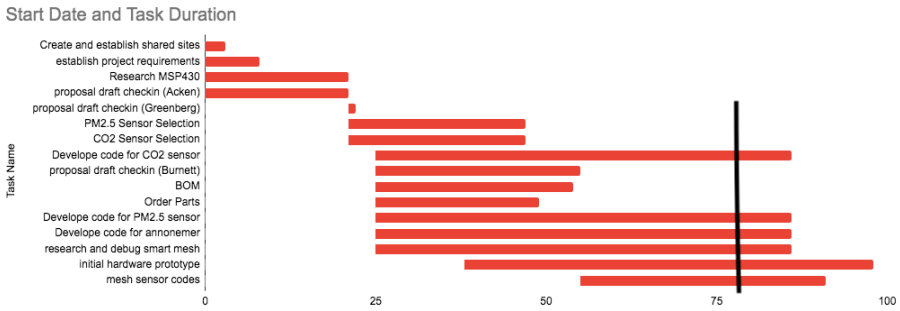


**Figure 6:** Gantt chart for winter term

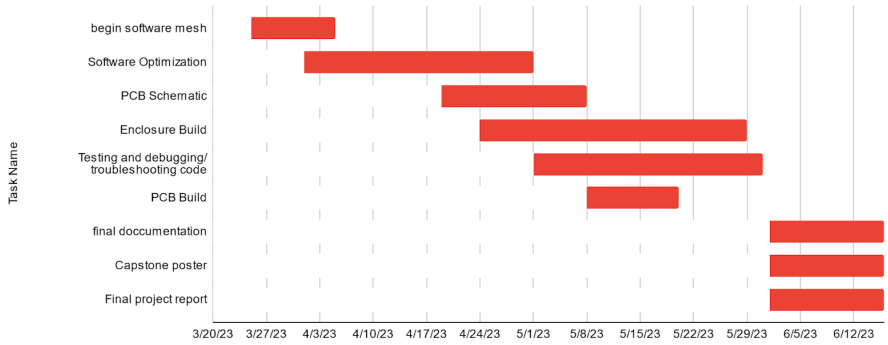


**Figure 7:** Gantt Chart for spring term

# 5 Test Plan

To test our modules we are employing a wide variety of stress testing. First we are starting some "long term" tests, where we are running modules 24/7 now that they are working to see if we encounter any issues with running constantly for so long. Next we are stress testing by randomly disconnecting sensors in the circuit to see how they react given random failures. This simulates a whole host of unknown issues that we might have during multi year operations. A third test we are performing is brown out testing, where we are powering our circuit with a power supply to see how everything reacts to voltages that are lower than expected during normal operation. And finally we are performing re-connection integrity tests, where we are testing our systems mesh capabilities forcing a node to hop between a second node to get to the main PC, as well as what happens when a node is brought out of range and then brought back. As more potential issues are brought up we will further edit our test plan.

# 6 Results

Our project was successful in satisfying all requirements. Our team was able to build 4 working nodes that all communicate using SmartMesh. The $CO_2$ and PM2.5 sensors are working properly within the acrylic enclosure built. The tables below show estimations of how often we can collect and publish data based on how long we want our batteries to last. The captured intervals can be modified using only 1 line in the code for the system. As you can see, the hot wire anemometer is the most power hungry part of this project. We recommend that teams look into using an alternative option like the ultrasonic anemometer.

| Sensor | 3 Months Battery Life | 6 Months Battery Life | 1 Year Battery Life |
|---|---|---|---|
| CO_2 | 8.5 min (510 sec) | 18 min (1080 sec) | 37 (2220 sec) |
| PM2.5 | 16 min (960 sec) | 34 min(2040 sec) | 72 min (4320) |

**Table 4:** Component Measurement Periods for 3 month, 6 month, and 1 year battery life for units without anemometer

| Sensor | 3 Months Battery Life | 6 Months Battery Life | 1 Year Battery Life |
|---|---|---|---|
| CO_2 | 12 min (720 sec) | 25 min (1500 sec) | 51 (3060 sec) |
| PM2.5 | 22 min (1320 sec) | 48 min (2880 sec) | 105 min (6330) sec |
| Ultrasonic Anemometer | 0.167 min (10 sec) | 0.333 min (20 sec) | 0.667 min (40 sec) |

**Table 5:** Component Measurement Periods for 3 month, 6 month, and 1 year battery life for units with an ultrasonic anemometer

| Sensor | 3 Months Battery Life | 6 Months Battery Life | 1 Year Battery Life |
|---|---|---|---|
| CO_2 | 40 min (2400 sec) | 90 min (5400 sec) | 220 min (13200 sec) |
| PM2.5 | 47 min (2820 sec) | 110 min (6600 sec) | 260 min (15600 sec) |
| Hotwire Anemometer | 16 min (960 sec) | 36 min (2160 sec) | 86 min (5160 sec) |

**Table 6:** Component Measurement Periods for 3 month, 6 month, and 1 year battery life for units with a hot wire anemometer

| Configuration | 3 Months Battery Life | 6 Months Battery Life | 1 Year Battery Life |
|---|---|---|---|
| Without Annemometer | ∼820 KB | ∼810 KB | ∼700 KB |
| Ultrasonic Anemometer | ∼16.15 MB | ∼16.14 MB | ∼16.27 MB |
| Hotwire Anemometer | ∼370 KB | ∼334 KB | ∼258 KB |

**Table 7:** Log file sizes after draining batteries for each configuration set to 3 month, 6 month, and 1 year battery life

# Post Mortem

In retrospect, we feel our project was very successful. Our team was able to collaborate in order to execute this project. Our team played to each of our specific talents which made for an excellent work environment.

Our team faced a variety of challenges in regards to using MSP430's, smartmesh, enclosure building and pcb design. Connecting each team member to their respective MSP430 proved to be more challenging than many of us had anticipated. Documentation surrounding MSP430 was scarce and oftentimes unhelpful. Most of our team members needed to delete Energia files and redownload before each use to connect with the MSP430. SmartMesh also had many problems that previous teams have not encountered. Our original ultrasonic anemometer proved to be much harder to work with which is why we switched to the hotwire option despite the huge discrepancy in power use.

The combination of all these problems and our approach to doing each part separately proved unfruitful as we had not built a prototype by the midpoint of the timeline. This is when we switched to dedicating a set time to work on the project weekly, in person, as a team. This proved to accelerate our progress as it allowed us to help each other at roadblocks. We also learned that some of our MSP430s were much more prone to having issues than others. This allowed us to rapidly

prototype a working node on the fastest system and MSP430. This also allowed different members to shift rapidly to complete auxiliary projects like reports and presentations.

Many thanks to our industry sponsor and faculty advisor who requested that we present our progress a several times throughout our project as well as requesting an updated Gantt chart weekly. This allowed us to compile our progress and have a clear path at a glance. Weekly meetings with at least 1 of the advisor or sponsor also proved necessary to work through a multitude of problems and is credited to keeping us working on the requirements.

Documentation was something that we excelled at throughout the project. Notes of weekly meetings were documented on the team's Trello board. Weekly reports and any documentation was copied to the team's Google Drive, Github repository, Trello board, and emailed to the advisor and sponsor. We also built a website to host some of this information.

# Project Resources

Software links List of used software and links:

- Energia (https://energia.nu/download/)
- Python (https://www.python.org/downloads/)
- KiCad(https://www.kicad.org/download/)
- Github Repository (https://github.com/brandonhippe/Indoor-Air-Quality-Monitor)
- FreeCAD (https://www.freecad.org/downloads.php)
- JLCPCB (https://cart.jlcpcb.com/quote)
- MakerCase ($https://www.makercase.com//$)

Versions used at time of build:

- Energia 1.8.10E23
- Python 3.11
- KiCad 7.0.2
- FreeCAD 0.20.2

# Software Manual

## 6.1 Setting up the Host Node

- Download Python. Links to download as well versions used at the time can be found in the project resources section of this report. Install matplotlib, tkinter, and PySerial using the command pip install ¡module name¿ in a python terminal.
- Clone or download the Github Repository as a .zip file. If downloaded as a .zip file, extract all of the contents.
- Open device manager, and expand the Ports drop down. Plug the DC2274A-A manager into the computer. 4 new COM ports will appear. Take note of the largest of these new ports.
- Navigate to the Code/Host Node/app/SensorDataReceiver folder. Run the SensorDataReceiver.py script. In the popup window, under port name, type in the COM port from step 3, including COM. Click connect, and the box should turn green to indicate successful connection.
- From the same folder, run the iaqGraphing.py script. A window with 3 plots should appear on screen. On the bottom, click the Plot Example button. Example data should get plotted in each of the plots.
- Now click on the Plot Data button. The graph will now plot new data received from the sensors approximately every 10 seconds, and display all the data over the last 24 hours.

## 6.2 Setting up Sensor Nodes

- Download Energia. Links to download as well versions used at the time can be found in the project resources section of this report

- Open the Code/Sensor Mote/libraries folder in the downloaded repository. Copy the contents of this folder into the library folder of your Energia installation (for easiest use, move the Energia Folder to the C drive, so the path is something like C:-1.8.10E23).
- Launch Energia and make sure to select the "MSP-EXP430FR2355LP" in the board section under the Tools tab
- Under File, press open, then navigate to the downloaded repository. Navigate to $Code/SensorMote/indoor_air_quality_v2 and open indoor_air_quality_v2.ino$.
- Click on the red checkmark to verify the code, and make sure it compiles without errors.
- Remove the lid of the sensor node
- Plug the other end of the USB cable plugged into the MSP430 into the computer
- Open device manager, and check under the Ports drop down for a device named MSP Application UART 1, and take note of the COM port number associated with it
- In Energia, select the COM port from step 10 in the Port section under the Tools tab.
- Open the Serial Monitor by clicking on the magnifying glass in the top right corner. Have this visible to see debug output after programming the MSP430.
- Click the red right arrow next to the checkmark to upload the code to the MSP430.
- Watch the Serial Monitor. You should see messages related to the SmartMesh system connecting to the host node. Once it connects, you'll see each of the installed sensors take their first measurements and send them to the host node.
- On the SensorDataReceiver.py terminal, verify that the first measurements from the sensors are received. It is now safe to unplug the USB cable from the

computer, place it back in the sensor node, close the sensor node, and mount the node wherever you'd like to collect indoor air quality data.

- Repeat steps 7-14 for any additional sensor nodes.

# Hardware Manual

## 6.3 Part 1: Building Enclosure

- Go to MakerCase(https://www.makercase.com//). Dimensions: 12 X 25 X 8 cm without an anemometer 12 X 30 X 8 cm with an anemometer
- Export the file as a DXF file
- Follow the instructions starting from 2.4.6 on the EPL's website (https://psu-epl.github.io/doc/equip/laser/QD-1390/using-this-machine)

## 6.4 Part 2: Building PCB

- Go to GitHub Repository and save all the .gbr files under the Air-Quality-PCB folder.

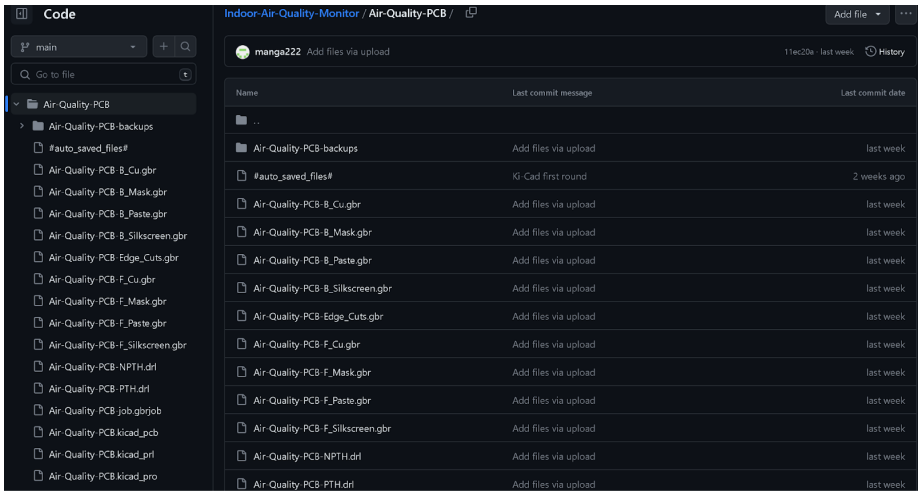


**Figure 8:** GitHub Repository

- Open the website of your PCB manufacturer and go to their ordering page
- We used JLCPCB for the project as they were the cheapest and fastest at the time. Link can be found in the project resources section

- Drag and drop all the .gbr files
- Select your options
- We selected the default options at time but chose to go with faster delivery
- Solder on connection points so the MSP430 can be seated onto the PCB
- Solder pins onto powerboost charger as well as the 3.3 converter
- We ended up removing LED(power and Low indicators) since they consumed too much power and were always on

# A Appendix 1: BOM

| Quantity | Model # and description | Link | Manufacturer | Price per unit | Total Price |
|---|---|---|---|---|---|
| 1 | CG-Anem (Wind Sensor With I2C Interface) | https://www.tindie.com/products/climateguard/wind-sensor-with-i2c-anemometr-arduino/ | Climate Guard | 23 | 23 |
| 1 | SGP30: ($eCO_2$ Sensor) | https://www.digikey.com/en/products/detail/adafruit-industries-llc/3709/8258468 | Adafruit | 17.5 | 17.5 |
| 1 | US-100 Ultrasonic Distance Sensor Module | https://www.digikey.com/en/products/detail/adafruit-industries-llc/4019/9808308?s=N4IgTCBcDaIK4GcC0BGADGkBdAvkA | Adafruit | 6.95 | 6.95 |
| 1 | SPS30 (PM2.5 Sensor) | https://www.digikey.com/en/products/detail/sensirion-ag/SPS30/9598990 | Sensirion | 50.25 | 50.25 |
| 4 | P Channel Mosfet | https://www.digikey.com/en/products/detail/alpha-omega-semiconductor-inc/AO3401A/1855773 | Alpha-Omega | 0.35 | 1.4 |
| 2 | N Channel Mosfet | https://www.digikey.com/en/products/detail/alpha-omega-semiconductor-inc/AO3422/1855787 | Alpha-Omega | .33 | .66 |
| 6 | 47k Resistor | https://www.digikey.com/en/products/detail/yageo/RC1206FR-0747KL/728931 | Yageo | 0.038 | .23 |
| 3 | 0 ohm Resistor | https://www.digikey.com/en/products/detail/yageo/RC1206JR-070RL/729184 | Yageo | 0.05 | .15 |
| 9 | Pin Header | https://www.digikey.com/en/products/detail/sullins-connector-solutions/PPPC101LFBN-RC/810182 | Sullins | 0.57 | 5.13 |
| 2 | Pin Header Male | https://www.digikey.com/en/products/detail/molex/0022284104/313970 | Molex | 0.67 | 1.34 |
| 1 | MSP-EXP430FR2355 | https://www.digikey.com/en/products/detail/texas-instruments/MSP-EXP430FR2355/9491427 | Texas Instuments | 15.33 | 15.33 |
| 4 | USE-1860-3500PCB(3.5Ah, 3.7V, Li-Ion, 18650, JST) | https://www.digikey.com/en/products/detail/us-electronics-inc/USE-18650-3500PCB/15098173 | US Electronics inc | 7.49 | 29.96 |
| 1 | BK-18650-PC8 (4 cell 18650 battery holder) | https://www.digikey.com/en/products/detail/mpd-memory-protection-devices/BK-18650-PC8/2330515 | MPD | 8.06 | 8.06 |
| 1 | DC9003A-C (Smart Mesh) | https://www.mouser.com/ProductDetail/Analog-Devices/DC9003A-C?qs=ytflelh7QUUd5QPivzMO2w%3D%3D | Analog Devices | 642 | 642 |

**Table 8:** BOM

Total Price: $159.97 excluding given parts $175.30 including MSP430

# A Appendix 2: Troubleshooting

## A.1 Energia Issues

Make sure "Energia MSP430 boards" is updated to version 1.0.7 or later by going to Tools, then Board, then Board Manager.

## A.2 Java Runtime Environment error

To solve this issue, You must save Energia to your C: Folder and open it from there each time. The Desktop Shortcut will keep giving you this error.

## A.3 Sensor Issues

Sensors drawing too much power:
We removed the power and low power LEDs on all of the sensors used as they were drawing too much.
CO2 sensor showing 400 PPM or 55K PPM:
Stop calibrating, run it for 30 seconds.

# References

Chapman, Stephen J. *Electric Machinery and Power System Fundamentals.* McGraw-Hill, 2002.

## Authors

**Adam Dezay** BS candidate in the Electrical Engineering department at Portland State with expected graduation date of September 2023. Adam works for Analog Devices at the time of publishing

**Manuel Garcia** BS candidate in the Electrical Engineering department at Portland State University with graduation date in June 2023. Manuel is pursuing his dream of becoming a stay at home father with great success so far.

**Brandon Hippe** BS candidate in the Computer Engineering department at Portland State University with graduation date in June 2023. Brandon is seeking to complete his masters after graduation.

**Mercedes Newton** is a BS candidate in the Electrical & Computer Engineering department with a Mathematics Minor at Portland State University with expected graduation in June 2023. Mercedes has accepted a position as an Electrical Engineer at Jacobs engineering starting in July of 2023.

## Acknowledgements

The authors would like to acknowledge their Industry Sponsor Dr. David Burnett and their Faculty Advisor Dr. John Acken for their commitment to both the project and our teams growth.

## Disclaimer

The contents of this report reflect the views of the authors, who are solely responsible for the facts and the accuracy of the material and information presented herein. This document is disseminated in the interest of information exchange.