SeatDetect

By

Owen Brown (owenmb2)

Yue Li (yuel7)

Huey Nguyen (hueyn2)

Final Report for ECE 445, Senior Design, Spring 2021

TA: Haoqing Zhu

May 5, 2021

Project #18

Abstract

SeatDetect is a human presence detection system created for use in libraries and potentially other areas to detect and map out available seating. The system uses a thermopile sensor for its detection and relays the data over Wi-Fi through an ESP32 to a DynamoDB database where the available seating is mapped out and available for viewing on a website.

Contents

1 Introduction

1.1 Objective

Before COVID-19, there was always a shortage of tables and seats at the Grainger Engineering Library, especially during weeknights. The cubicles were almost always full and if they were not, they were occupied by miscellaneous items. During those times, it would be extremely helpful to know which table in the library is open instead of walking in circles and waiting for a table to open. When there are absolutely no seats available, students spend an excessive amount of time coming to Grainger just to realize there is not an available cubicle to take. Our team seeks a way to alleviate this issue and promote a more accessible library environment.

To solve this problem, the team will design and implement a device for each table that measures the temperature of the surrounding area which can tell the user whether the cubicle is occupied in advance. The occupancy status should transmit through Wi-Fi to the end-user, and the status should be updated in a timely manner after each change to ensure real-time accessibility. Therefore, the user should be able to view these changes, physically locate the desired location, and occupy the free seat. This allows students, the primary users, to save lots of time before entering Grainger, making the library more easily accessible using digital solutions.

In addition to the main problem, the solution should also solve additional issues regarding library policies. For instance, the library staff can enforce more policies such as a maximum amount of inactivity time within each cubicle. This eliminates the issue of students holding spots for others, or themselves, which is especially an issue during crowded times. The team strives to allow all library patrons an equal amount of accessibility in Grainger library and seeking to cultivate a fair environment for all.

1.2 Background

While there are no known existing solutions to this specific problem, solutions to similar problems are already present today. One of these similar problems is finding a parking spot in a parking garage. This problem is solved using infrared technology [1], AI [2], and using various other sensor types. The idea of using a type of sensor to detect an object can be adapted to fit the problem statement, which is to detect humans. Similar to how the parking spot communicates to the driver whether the spot is occupied, the solution must also find a means of real-time communication to inform users of status updates.

From here, it is easy to draw the parallel between finding a parking spot and finding a seat in the library. Since these technologies are utilized in real-world applications, it proves that this problem can be solved in an affordable way, especially since the technology needed to detect a

person in a booth is arguably cheaper than detecting a car in a parking spot. In efforts of increasing accessibility and functionality, the project includes an additional web user interface as the primary method in identifying availability spaces.

Due to the scale of the team's proposed project, only the designing and testing of the prototype for one unit will be carried out. However, if the project is scaled up to cover all of Grainger Engineering Library, it would require a much greater number of sensors and power supply. The throughput of the data being transferred would also be much higher. These are all factors to consider, especially when planning to increase the scalability to truly solve the main problem statement.

1.3 High Level Requirements

The system includes the following high-level requirements that are desired to be met to ensure full functionality. This list includes one major requirement for each major level of the system out of the hardware, firmware, and software respectively.

1. An accuracy of occupancy status over 95% on repeated tests of the same booth is the benchmark. This is measured by running 100 occupancy sensing tests on the same sensor unit and counting how many successful readings there are out of total attempts.
2. The occupancy status should change from unavailable to available within 15 mins after the seat is no longer occupied. This is to account for brief periods of inactivity such as bathroom breaks. However, the status should change from available to unavailable immediately after the seat becomes occupied.
3. The transfer of user data for occupancy status updates on the mobile/web app should be within 30 seconds of a status change. This is set to ensure the system of the device and web/phone app of all users can be serviced and accessible.

1.4 Block Diagram

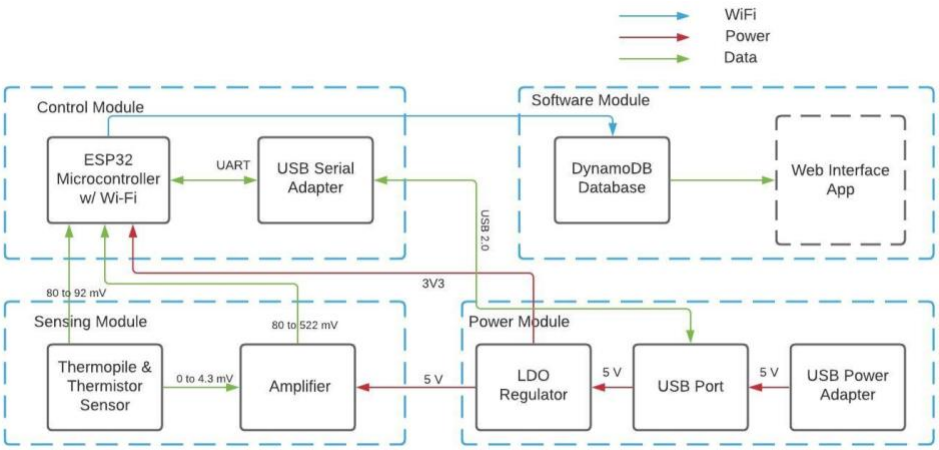

Figure 1. Block Diagram

Figure 1 represents the block diagram for SeatDetect. SeatDetect consists of four main modules: a power supply, a software module, a control unit, and a sensing module. The power module ensures that the system can be powered continuously all day and night with the proper 3.3V. The sensing module is the human occupancy detection system. The sensing module sends the raw occupancy data to the control unit which contains a microcontroller with integrated Wi-Fi to process and send the status to the software module. The software module displays the status data to the end-users, the students.

1.5 Physical Design

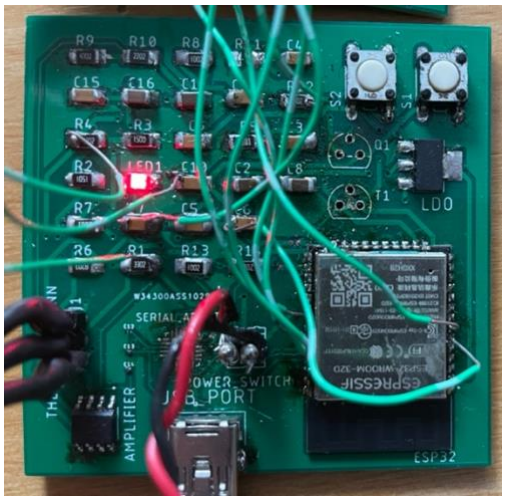

Figure 2. Physical Printed Circuit Board

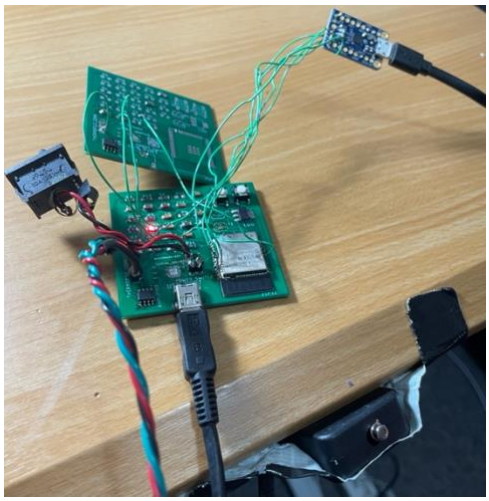

Figure 3. Physical System

Figure 2 represents the physical PCB (printed circuit board) and Figure 3 represents the physical system. Some adjustments were made to the final system for debugging purposes, but the final PCB in the real use case would look much cleaner, without the need of any of the green external wires seen here. The extra PCB is used for further amplification of the sensor signals for the ESP32 to read them with more accuracy. Most of the wires are for the serial adapter module that was used in place of the more integrated serial adapter which was unable to be successfully soldered on the main board. Also, in the real use case the whole system is meant to fit into an electronic housing with holes cut out for the sensor wiring, the USB port, and mounting of the power switch. It is important to mention that the sensor can be seen in the bottom right of Figure 3. Note the orientation of the sensor at a 45° angle at stomach level which allows the avoidance of seat heat while still catching the heat of the human. The sensor is housed in an electronic box for protection and is taped to the table, although in the real use case it would most likely be mounted on a bracket.

2 Design

2.1 Design Procedure

2.1.1 Power Module

The power module is a wired USB power supply for increased scalability and consistency, which is required to keep the communication network up continually. In a network with multiple sensing modules, this power module is used to power the whole system.

2.1.1.1 USB Power Adapter

The power adapter converts the grid electricity into DC power that is used for the whole system. This was not designed but simply bought from a third party.

2.1.1.2 USB Port

The mini–B female USB port receives power from the power adapter and allows for programming the ESP32 from a computer (at separate times). The port connects to the voltage regulator for power supply and a serial adapter for communication to the ESP32. When a computer is hooked up to the port, the power adapter wouldn't be, and vice versa.

2.1.1.3 LDO Regulator

The LDO regulator steps down the voltage from the power adapter from 5V to 3.3V for the ESP32's power requirement.

2.1.2 Sensing Module

The sensing module senses if the user is in the booth using a thermopile sensor and some residual circuitry. It relays passive signals from the thermopile to the control module where it is then processed to sensible and understandable data.

Figure 18 represents an example interface circuit for the sensor [3].

$$V_{out} = V_s\left(1 + \frac{R_6}{R_5}\right) + V_{th} \tag{1}$$

Equation (1) is the equation for V_{out}, which is derived from simple ideal op-amp analysis. As seen in Figure 18, V_s represents the voltage generated from thermal energy by the thermopile while V_{th} represents the voltage of a node that depends on the value of R_{th}, the resistance of the thermistor that depends on ambient temperature. Using the equations for V_s and V_{th}, equation (1) can be rearranged into equation (2) and used to find T_O (K), the object temperature, otherwise known as the temperature of the person sitting at the booth.

$$T_O = \sqrt[B]{\frac{V_{out} - V_{th}}{S\left(1 + \frac{R_6}{R_5}\right)} + T_A^B} \quad (2)$$

The derivation of equation (2) is discussed further in section 2.2.2.

2.1.2.1 Thermopile Sensor

The ZTP-135SR is a thermopile sensor that detects human presence by thermal energy correspondence. This sensor is the cheapest option available on the market at around $5 compared to $15 which fits the use case of the project very well. In the use case, there will be one sensor for each booth, which increases the cost drastically. The most expensive component in the scaling is the sensor. The only way to justify the cost for the solution is to keep the cost low to make the product marketable. Using a $40 thermal camera for every booth is not a very viable choice, as the product would not be very marketable with the high price.

2.1.2.2 Amplifier

The amplifier needs to amplify the signal from the sensor as the signal voltages are much too small to be read by the ESP32.

2.1.3 Control Module

The control module interfaces with every single module. It also relays preprocessed data from the sensing module to the software module and includes dynamic memory storage included within the ESP32 microcontroller.

2.1.3.1 ESP32 Microcontroller w/ Wi-Fi

The ESP32 microcontroller receives the sensor signals, processes the data into whether the seat is occupied or not, and sends the data to the database over Wi-Fi. Using Wi-Fi communication allows for faster data transfer and scalability between multiple sensor data being processed through the microcontroller, which is then relayed to a hosted DynamoDB database.

The ESP32 does have limitations when it comes to the limited number of analog to digital (ADC) pins, especially when operating with the use of Wi-Fi, which cuts the number of available ADC pins in half, from 20 to 10. The current design of this system relies on 2 ADC pins per sensing module (i.e., per booth), which only allows for 5 sensing modules. While it is possible to have a system like this design and only use 1 ADC pin, the ESP32 still is only left with 10 sensing modules before having to duplicate the control and power modules, adding significant cost to the system. With few alternatives available to the ESP32 that have significantly more pins, using techniques like shift register multiplexing would possibly be the most viable option for scaling the system. While this would increase the time that it takes to process an updated status, the

30 second update time high level requirement would be more than enough time to execute all the booths' statuses.

2.1.3.2 USB Serial Adapter

The USB serial adapter is necessary to enable the programming of the ESP32. It takes in the uploading of firmware from the computer by converting the USB data to universal asynchronous receiver-transmitter (UART) data that the ESP32 can then interpret.

2.1.4 Software Module

The software module consists of a web-based application that allows users to check availability and location of seats as well as a database that stores the information pertaining to each seat.

2.1.4.1 DynamoDB Database

The DynamoDB database stores the data and the ability to view the status of a seat (occupied by a person, personal items, inactivity, etc.). DynamoDB is chosen due to its durable and multi-active database capabilities.

SeatID (Integer)	Status (Boolean)	lastUpdatedTime (DateTime)
1	FALSE	2021-02-13T17:09:42.411
2	TRUE	2021-02-13T17:09:42.411

Table 1. DynamoDB Table

Table 1 represents the data transmitted by the sensors must be persisted into the DynamoDB database. This database should have the Seat ID as a primary key and it stores the perspective occupancy status and the time which it was last updated.

2.1.4.2 Web Interface App

The Web Interface App allows the user to view the map of Grainger Engineering Library (for now) and its corresponding tables on each floor. It is available to view on a standard web browser on a computer. The interface is interactive so that users could select which floor to view availability, and the corresponding status of each cubicle at the specified location. The user-friendly design and lay-out is intended to allow the user to find the seat relatively easy, as the web page displays the location and orientation of each cubicle in a clear and intuitive way which simplifies the view of Grainger Engineering Library.

2.2 Design Details

The full detailed schematic, board layout, and code flowcharts can be found in the respective appendices.

2.2.1 Power Module

The design of the power module is very simple and therefore is not necessary to expand upon in any further detail.

2.2.2 Sensing Module

This section will further discuss the derivation of equation (2).

$$V_s = S(T_O^B - T_A^B) \tag{3}$$

$$\mathrm{S} = \text{Sensitivity coefficient (mV)}$$

$$\mathrm{T_O} = \text{Object temperature (K)}$$

$$\mathrm{T_A} = \text{Ambient temperature (K)}$$

$$\mathrm{B} = \text{Coefficient } (\sim 4)$$

Equation (3) is the equation for the thermopile voltage V_s [3]. This equation is not useful for calculating T_O, unless S and T_A are known values. Measuring T_O is the main purpose of this circuit because T_O represents the temperature of the person sitting at the booth, which is needed to detect the presence of a user.

Figure 19 [5] represents the plot of V_s vs. T_O when $T_A = 25\ °C$. While this plot is not useful by itself since it only provides values of V_s for a fixed T_A, it is useful for approximating S using a rearrangement of equation (3).

$$S = \frac{V_s}{T_O^B - T_A^B} \tag{4}$$

To approximate S using equation (4), the variables that most closely resemble the actual application is used in the calculation. The application takes place in the Grainger Engineering Library, which is most likely using $T_A \approx 22\ °C$ corresponding to room temperature and $T_O \approx 37\ °C$ corresponding to body temperature. Figure 19 can be used to find that $V_s \approx 2.65\ mV$ at $T_O = 37\ °C$. Then, equation (4) can be used to estimate $S = 1.96 \times 10^{-12}$.

If $T_A = 22\ °C$ always held true, no further calculations would be necessary as all the variables would be solved for, but it is better to account for the ambient temperature as well, as the room temperature of Grainger is not always going to be the same. Since V_{th} is a function of R_{th} which is a function of T_A, R_{th} needs to be calculated next, as V_{th} is obtained by simply probing the node seen in Figure 18.

Since V_{th} is a function of R_{th} which is a function of T_A, R_{th} needs to be calculated next, as V_{th} is obtained by simply probing the node seen in Figure 18. R_{th} can be solved for by doing analysis on the surrounding resistor divider circuit.

$$R_{eq} = \frac{R_2(R_4 + R_{th})}{R_2 + R_4 + R_{th}} \tag{5}$$

$$V_x = \frac{V_{in}(R_{eq} + R_3)}{R_1 + R_3 + R_{eq}} \tag{6}$$

$$V_y = \frac{V_{in}R_3}{R_1 + R_3 + R_{eq}} \tag{7}$$

$$V_{th} = \frac{R_4(V_x - V_y)}{R_4 + R_{th}} + V_y \tag{8}$$

Equations (5), (6), (7), and (8) are created using Figure 18, where R_{eq} is the equivalent resistance between nodes V_x and V_y. These four equations can then be used to solve for R_{th} in equation (9).

$$R_{th} = \frac{(V_{in} - V_{th})[R_2(R_3 + R_4) + R_3R_4] - V_{th}R_1(R_2 + R_4)}{V_{th}(R_1 + R_2 + R_3) - V_{in}R_3} \tag{9}$$

The reason a resistor divider is used in this circuit is to linearize V_{th} with temperature because R_{th} has an exponential relationship with temperature as seen in equation (10) [4].

$$R_{th} = R_2 = \frac{R_1}{e^{\beta(\frac{1}{T_1} - \frac{1}{T_2})}} \tag{10}$$

Equation (10) represents the method to approximate R_{th} at temperature $T_A = T_2\ (K)$ using a known temperature $T_1\ (K)$ and the resistance R_1 at T_1. The equation also uses the variable beta, which is found in the thermopile's datasheet [5] to be $\beta = 3960\ K$. This equation can be rearranged and used to solve for T_A.

$$T_A = T_2 = \frac{\beta T_1}{\beta - \ln\left(\frac{R_1}{R_{th}}\right)T_1} \tag{11}$$

Using Table 3 [5], any T_1 near T_A can be used in equation (11).

Finally, now that all the variables are accounted for or solvable, equation (1) can be rearranged into equation (2) and used to find T_O (K), the object temperature, otherwise known as the temperature of the person sitting at the booth.

2.2.3 Control Module

From section 2.1.2, the sensing module voltages V_{out} and V_{th} can be received and processed by the sensing module into temperatures T_O (object temperature) and T_A (ambient temperature) using equations (2) and (11), respectively. Both temperatures can be used to determine whether a person is sitting at the booth. This can be done by taking the difference of them. This temperature difference, T_D, will need to have a threshold value set in the firmware to determine whether a person is sitting at the booth.

For accuracy purposes, the worst-case temperatures are considered so that the reason for failure is never due to environmental factors that are unaccounted for. The worst-case temperatures would be a high room temperature and a low body temperature. For calculation purposes, $T_A = 26\ °C$ and $T_O = 35.5\ °C$ are the assumed worse cases, which is a T_D of $9.5\ °C$. The room temperature should almost never reach that high assuming a working air conditioning in the Grainger Engineering Library and the body temperature is just above hypothermia levels. Knowing the *actual* T_D should never be less than $9.5\ °C$, the system can be set up to signal human presence when T_D is $> 8\ °C$ to give even more room for error.

In ideal circumstances, there would be no error from the sensor. However, because it is not required to update the status of a seat immediately according to requirement #3 in section 1.3, there is some time available to account for errors from the sensor, which can be especially useful considering the sensor is on the cheaper side. To do this, a simple moving average filter is used. A simple moving average simply takes the average of the last 10 (in this case) readings. This will cause the status to update more slowly, but the firmware is set to take readings every couple of seconds, so the status will be able to reflect the changes in the correct amount of time. This technique is used to filter out any outliers that are caused either from the sensing module or the control module so that the status doesn't change because of a misreading. This filter also eliminates the errors associated with people walking behind the chair, as they will only be there momentarily and therefore only for a singular or a couple readings.

The other major component in the firmware is the logic that accounts for requirement #2 in section 1.3, which accounts for people leaving for bathroom breaks. This logic is simple in that it simply checks if that status has changed from occupied to unoccupied and then waits for 15 minutes before checking the status again and then proceeding to send that status change, if there is one (i.e., the person did not come back), to the database.

The firmware logic for the control module is laid out it in a flowchart that can be found in Appendix B

2.2.4 Software Module

2.2.4.1 Cloud Computing

Once the control module finishes preparing data, it sends a message to the MQTT Client as shown in the flowchart in Figure 15. Once the MQTT Client receives the message as illustrated in Figure 4, the AWS IOT rule is created to trigger the lambda function shown in which parses this JSON file into different columns in the DynamoDB table shown in Figure 6. The entire process can be illustrated in the flowchart displayed in Figure 16.

Figure 4. MQTT Client receiving a message

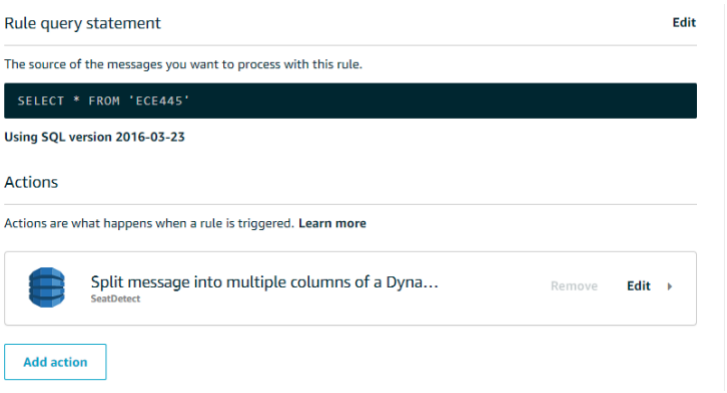

Figure 5. AWS IOT Lambda Function

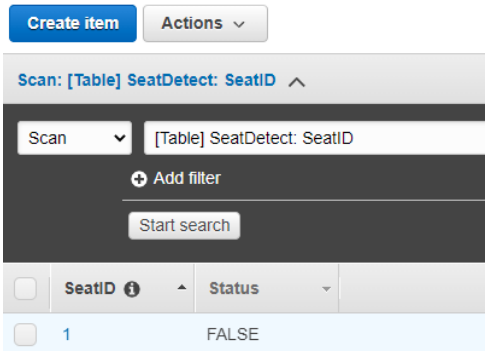

Figure 6. Production DynamoDB table

2.2.4.2 Front End

The front end of this web application is constructed using Python Flask coupled with HTML and CSS.

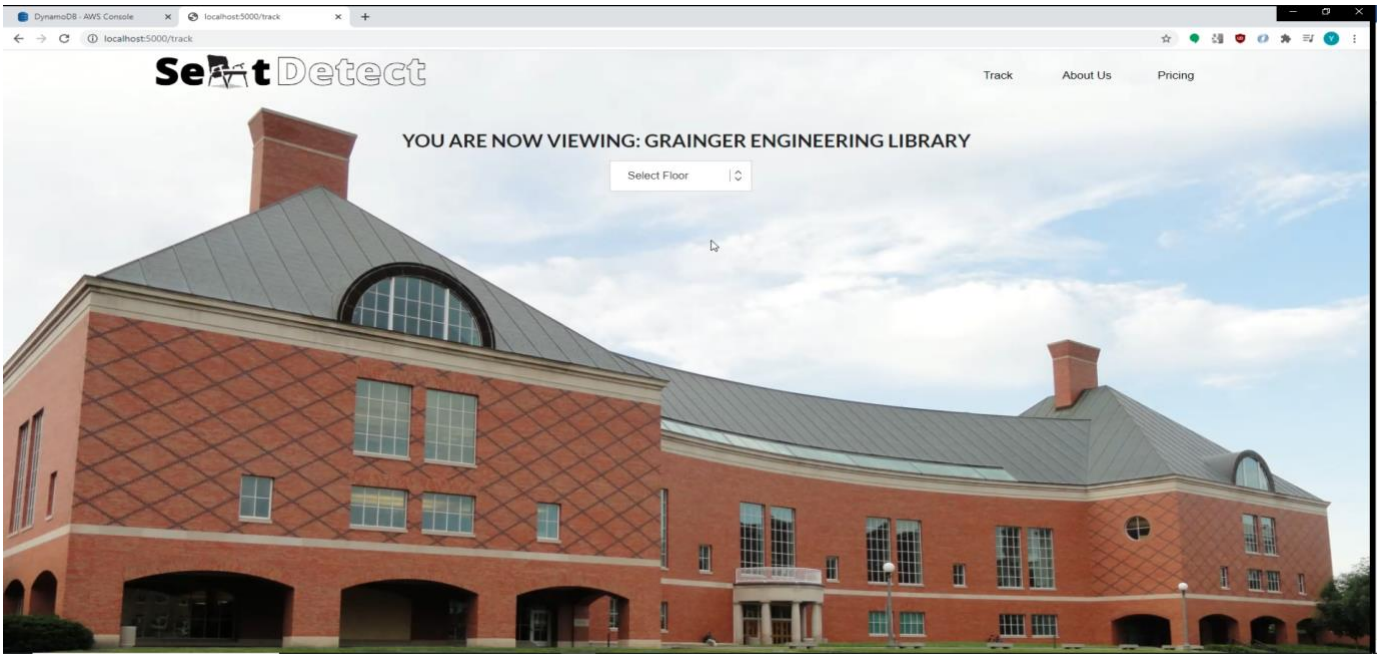

Figure 7. Website Home Page

Figure 7 represents the design of the home page for the web application. From this page, the user can select the floor that they would like to view the available seating of or they can view the “About Us” or “Pricing” sections.

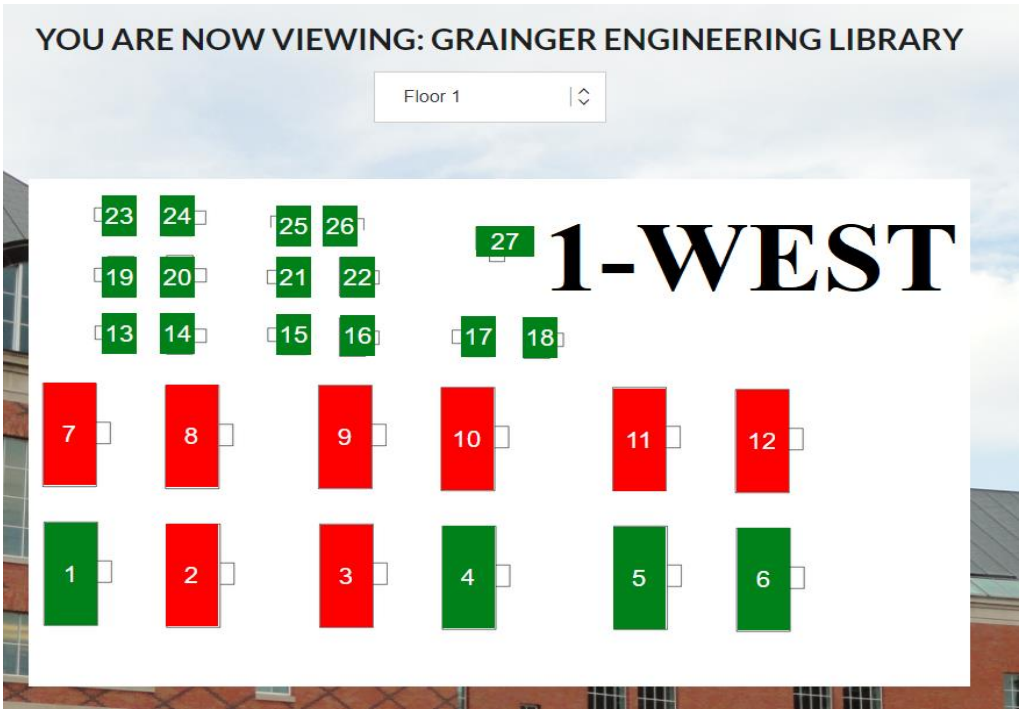

Figure 8. Website Tracking Page

Figure 8 represents the design of the tracking page for the web application. On this page the user can see the available seating on the selected floor of the Grainger Engineering Library by looking for the green spots on the available map.

2.2.4.3 Back end

The back end for this web application is constructed using Python with the package Boto3. The process scans through each seat and see if the status for that seat has been changed for not, if so, it is going to reflect the change to the end-user, if not, it moves on to the next seat. This process runs every 20 seconds. While for demonstration purposes the cost of reading a single record in the database is negligible, this is done to keep the cost of maintenance of the DynamoDB manageable in the event of the project being scaled up as constantly reading hundreds of records from the database can be expensive. The flow chart of the back-end application looks like Figure 17.

3 Requirements & Verification

3.1 High-Level Requirements

The high-level requirements were briefly discussed in section 1.3 and therefore will be referenced by their respective number in the list.

3.1.1 Requirement 1

The first requirement was tested by monitoring the occupancy status of the system while sitting down and leaving the booth repeatedly therefore changing the physical status of the system. The successful status readings are then recorded and then divided by the total number of physical status changes to get the final accuracy. This requirement was achieved before demonstration day, with 100% in the ideal circumstances. This accuracy held true until the human came back 6-12 inches away from the sensor to which the accuracy decreased significantly to almost 50% as the sensor was not able to accurately measure the human presence due to the low voltage it was receiving from the low heat available. The accuracy barely held true when adding an excessive amount of clothing articles, but the 100% accuracy was still achieved.

As mentioned before, this requirement only held true the day before demonstration day, but on demonstration day this requirement failed. The sensing did work at a very close distance on exposed skin, but this was not sufficient to justify verifying the requirement, as it is a scenario outside the actual use case. The failure of this requirement is discussed further in section 3.2 where the failure of a low-level requirement that caused this requirement to fail is discussed.

3.1.2 Requirement 2

The second requirement was tested by changing the 15-minute requirement to 15 seconds for ease of testing and simply occupying the seat and then preforming two different actions afterwards to test all the outcomes. The first outcome was leaving the seat unoccupied after the 15 second period to see if the seat is marked correctly as unoccupied and the second outcome was tested by doing the same thing but instead occupying the seat and checking if it is marked correctly as unoccupied. This requirement was met with no issues, as the status was updated correctly in every expected outcome.

3.1.3 Requirement 3

The third requirement was tested by changing the status, either from unoccupied to occupied or vice versa, seeing the change occur in the serial monitor of the Arduino IDE, and then timing how long it took for the status to change in the database. The time that it took to update was unmeasurable as it was under 1 second. Therefore, this requirement was met with no issues.

3.2 Low-Level Requirements

The complete list of low-level requirements for each module is available for viewing in Appendix F This section will only discuss the low-level requirements that failed, ultimately causing the failure of high-level requirement #1.

It can be seen in Table 6 that requirements #2 and #3 failed. Both requirements failed due to the unexpectedly low voltage of V_s, the voltage of the thermopile. V_s is passively generated from infrared (IR) radiation. Therefore, the sensor relies on the IR signals being as clear to see as possible.

In equation (3), V_s depends on a few variables, but the only one that could be causing the requirements to fail is the sensitivity coefficient S, which was assumed to be constant to make calculations possible, but it changes dynamically with many factors.

$$S = R_{th} N A \varepsilon \sigma F \tag{12}$$

$$\mathrm{N} = \text{Number of thermocouples}$$
$$\mathrm{A} = \text{Seebeck coefficent}$$
$$\varepsilon = \text{Net emissivity}$$
$$\sigma = \text{Stephan's constant}$$
$$\mathrm{F} = \text{Field of view (FOV)}$$

While a few variables in equation (12) are dynamic, the variable most likely causing the failures is the field of view (FOV) of the sensor, F.

Figure 20 [5] represents how V_s changes with changing FOV. To explain the plot, the output of $V_s = 0$ if the object of focus is 70 degrees out of the FOV. If the object is directly perpendicular to the sensor, $F = 100$.

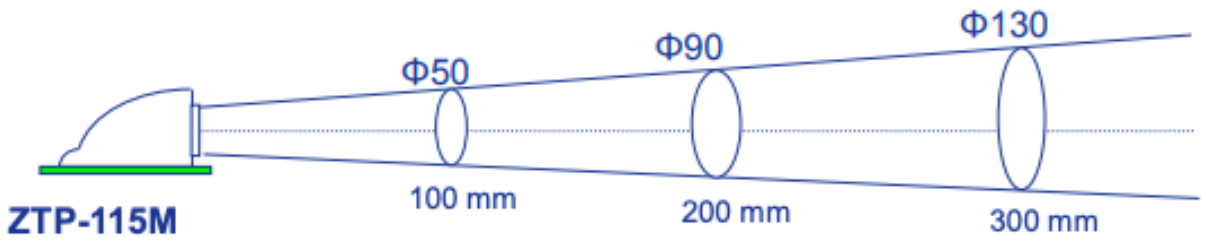

Figure 9. Sensor FOV Diagram

Figure 9 [3] represents the FOV for a related (in the same family) Amphenol sensor. While not the same sensor as the one that used in the system, this figure most likely represents a similar FOV of the ZTP-135SR. For F to be maximized, the FOV must only include the object that the

sensor is measuring the temperature of. In the use case of the sensor, the human will be around 300mm away from the sensor. The FOV is small enough for the human to completely fill the "circle" seen in Figure 9. In the previous calculation of S that used Figure 19 for its estimation, $F = 85$ [5], which likely is a good estimate for the use case, as the human might sometimes be further away than 300mm. Therefore, the failure of the requirements is most likely not due to a miscalculation, but these calculations can now be used to explain the reason for failure, as the FOV of the sensor can be affected by external factors.

Figure 21 shows the decrease of V_{out} over time. The dotted green line on the plot represents the expected value for V_{out}, which was achieved with the sensor in the circuit on the first day. However, over time this value decreased almost exponentially over time.

It is very important to point out the last data point on the plot, which does not seem to follow the exponential decrease trend. This data point was taken after simply cleaning the sensor with some alcohol. While V_{out} did not come all the way back up to its expected value, this change proves that the problem was simply with a contaminated lens. Therefore, the conclusion can easily be drawn that the reason for voltage still not coming all the way up to the sensor is due to the damage that the lens must have taken over countless hours in the laboratory putting the system together and testing it, along with contamination from testing techniques such as pressing up a finger against the lens to see what voltage output that it causes.

Earlier it was concluded that the most likely variable to affect this voltage to change so rapidly is the FOV of the sensor, so this conclusion makes sense given that any contamination of the sensor can negatively affect any transmission of IR light which is the reason FOV is a variable in the first place.

However, it is important to note that looking back at Table 6, requirement #3 is the most crucial for the success of high-level requirement #1. In other words, high-level requirement #1 was successful until demonstration day on 4/26/21. In Figure 21, on 4/26/21, V_{out} became so low of a value that the sensor was not able to pick up any IR radiation. On 4/27/21, V_{out} became high enough to compensate for the error in the firmware to be able to pass requirement #3 and therefore passing high-level requirement #1, while still failing requirement #2 on Table 6.

Regardless, solutions to fixing requirement #2 are discussed in section 5.3, since it is still an important requirement to meet to ensure the integrity of the system, even though this application of the sensor might not require exact temperatures to function completely, it is still desired to reduce the error of the system to ensure more stability and reduce changes to the firmware for calibration purposes.

4 Costs

4.1 Parts

Table 4 represents the parts cost breakdown for the SeatDetect system. The total amounts to $28.16 for the parts only.

4.2 Labor

Team Member	Dollars per Hour	Hours per Week	Weeks	Multiplier	Total
Owen Brown	$40	15	12	2.5	$18000
Yue Li	$40	15	12	2.5	$18000
Huey Nguyen	$40	15	12	2.5	$18000
Total	$120	45	36	2.5	$54000

Table 2. Labor Cost Breakdown

Table 2 represents the labor cost breakdown for each team member and in total. The hourly wage is a representation of the average starting salary for each of the respective majors.

4.3 Total

$28.16 in parts plus the $54,000 in labor amounts a total cost of $54,028.16 for the project.

5 Conclusion

5.1 Accomplishments

The main accomplishments of the project include the smooth firmware and the quick, responsive software with a user-friendly interface.

The firmware accomplishment was facilitated by the incorporation of AWS technologies to ensure a smooth transition from hardware to software. The modularity of firmware made performing unit tests and debugging extremely easy. For example, when the ESP32 attempts to send a message to the subscribed topic, AWS CloudWatch metrics would accurately display what data was sent with accurate timestamps. In addition, the lambda function that parses the JSON message sent by the ESP32 to the DynamoDB table was also unit tested by customizing correct and erroneous messages to send to the database and monitoring the behavior of the process.

The software component of this project is also a big success as it accurately reflects the status changes of a particular seat in a timely manner. The selection of Python Flask as the main framework facilitated the accomplishment in the software portion. The native support of HTML and CSS with this framework made the linkup between frontend and backend straightforward. In addition, since this is a Python framework, other packages such as Boto3 is conveniently used to establish connection with AWS, making the constant retrieval process from the database significantly easier.

5.2 Uncertainties

As discussed in 3.1.1, our high-level requirement of achieving 95% accuracy through repeated tests on a singular seat was not achieved on demonstration day mostly due to the uncertainties surrounding the thermopile sensor.

The theme of this project was established in the beginning that a little accuracy can be sacrificed in exchange for significant reduction in cost. However, as discussed in section 3.2, the deterioration of the sensor resulted in great difficulties testing the system as the output voltage would vary day by day, making unit tests near impossible.

5.3 Alternatives

Continuing from section 3.2, it was concluded that the sensor greatly deteriorated physically over the period of the building and testing of the sensor and the complete system. This section will discuss improvements that can be made to the current sensor housing to make the sensor less susceptible to damage and contamination for the real use case which would be countless hours around unpredictable students.

Figure 10. Original Sensor Housing Open

Figure 11. Original Sensor Housing Closed

Figure 10 and Figure 11 represent the original sensor housing placement. This housing consists of an electronics box purchased online that contains a hole exactly large enough for the sensor to fit through. However, the entire sensor head sits on the outside of the case leaving it more susceptible damage and contamination.

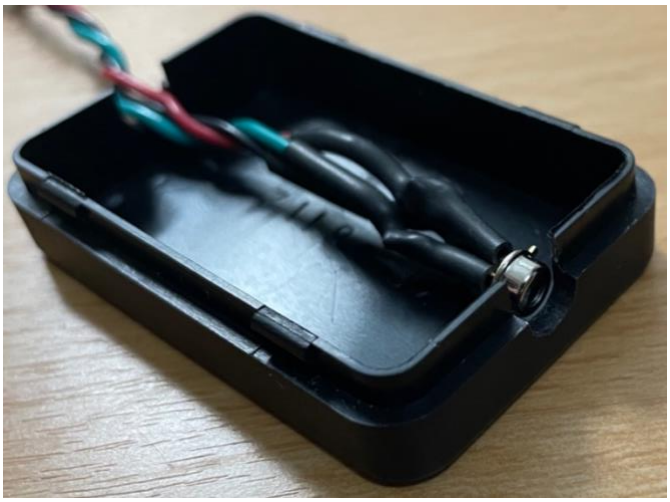

Figure 12. Alternative Sensor Housing

Figure 12 represents a much better housing arrangement for the sensor that consists of attaching the sensor to the inside of the case instead of the outside. This placement allows for full FOV of the sensor while mitigating the contamination and damage that the sensor previously experienced. Of course, the sensor will still experience some contamination, but the sensors will need to be cleaned much less this way allowing for the preservation of the voltage output of the sensor for a longer period.

Of course, this solution has not actually been tested but is only a theory to fix the original problems of the system. Alternative housings should also be tested before moving onto alternative solutions.

If this solution does not work, it can be concluded that for this use case, the cheap sensor may simply just be too cheap. In this case, more expensive thermopile sensors can be considered. Another solution that can be considered is opting for a movement-based solution with the use of passive IR (PIR) sensors. This solution does have its own drawbacks, which are the same drawbacks that come to mind when thinking of the drawbacks of a motion-sensing automatic light system, in that it requires movement from the user every so often to accurately detect presence, whereas the temperature-based system outlined in this report does not rely on unpredictable behavior of humans, but instead is able to account for most behaviors.

5.4 Ethical Considerations

Ethics and Safety is imperative to successfully carry out the project. During this difficult time of COVID-19, it is especially important that the team follows closely IEEE Code of Ethics #1 [7], that the team does not put other people's health in harm's way while conducting this project. That means to closely follow the CDC guidelines as well as to build and test the project with as little face to face interactions as possible. When going to the lab is necessary, precautions such as wearing gloves, using hand sanitizer regularly needs to be taken extremely seriously.

In addition to paying attention to safety, it is also important that the team follows the guidelines of IEEE Code of Ethics #5, which suggests that the team needs "to seek, accept and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic" [7]. The team needs to take advantage and make the most out of the weekly TA meetings and be proactive when possible. Furthermore, the team is planning on following this guideline and conducting surveys of the prototype once it starts working to receive feedback to further improve the product. One ethics concern one would consider relates to the issue of privacy. It can be a source of concern because the team is collecting the seat occupancy data for the entire library and understandably people might feel uncomfortable dealing with their occupancy status being collected despite not collecting any personal information. Right now, the team does not see a solution that can resolve this concern but will continue exploring options that can make the end users feel more comfortable.

5.5 Overall Impact

The original mission of this project was to help students reduce the wait time when libraries get busy. As this system scales up, it can be used not only other libraries, but fitness centers, tennis courts and restaurants to greatly reduce wait time. As of now, there are very few products like this on the market to help people in highly populated areas to find seats conveniently so the impact on economically is immense.

5.6 Future Work

This system has great potential to solve the problem initially outlined in section 1.1. While this specific system provided an adequate solution, there is still a lot of work that can be done to improve the system.

As mentioned in section 5.3, the future work to the hardware will include the testing of alternative sensor housings and potentially completely different sensors to find the best balance between cost and functionality.

Further, as mentioned in section 2.2.4, the process of information retrieval from the database happens every time the user refreshes the tracking page, instead of updating the page constantly. This was done for this project only to reduce costs of maintenance and difficulty. If time permits, the system will benefit from the tracking page updating the seat status as that would be more user-friendly despite being much more difficult technically.

Lastly, the modularity of the software module can be improved. For demonstration purposes, only one seat was programmed to retrieve and update information. As a result, the logic illustrated in Figure 17 was not followed exactly. If more than one seat is programmed, different methods can be used to passively check if the status of an individual seat has been updated instead of retrieving every single status of all the seats.

References

[1] D. Roos, “How Parking Garages Track Open Spaces, and Why They Often Get It Wrong,” *HowStuffWorks*. [Online]. Available: https://electronics.howstuffworks.com/everyday-tech/how-parking-garages-track-open-spaces-why-they-often-get-it-wrong.htm

[2] *ParkingDetection*. [Online]. Available: https://www.parkingdetection.com/

[3] *Mouser*. [Online]. Available: https://www.mouser.com/pdfDocs/AAS_Infrared%20Sensor.pdf

[4] *Ametherm*. [Online]. Available: https://www.ametherm.com/thermistor/ntc-thermistor-beta

[5] “ZTP-135SR Datasheet.” Thermometrics.

[6] H. Haraki, “Demystifying Thermopile IR Temp Sensors,” *Fierce Electronics*. [Online]. Available: https://www.fierceelectronics.com/components/demystifying-thermopile-ir-temp-sensors

[7] ieee.org, "IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 08-Feb-2021].

Appendix A Hardware Development

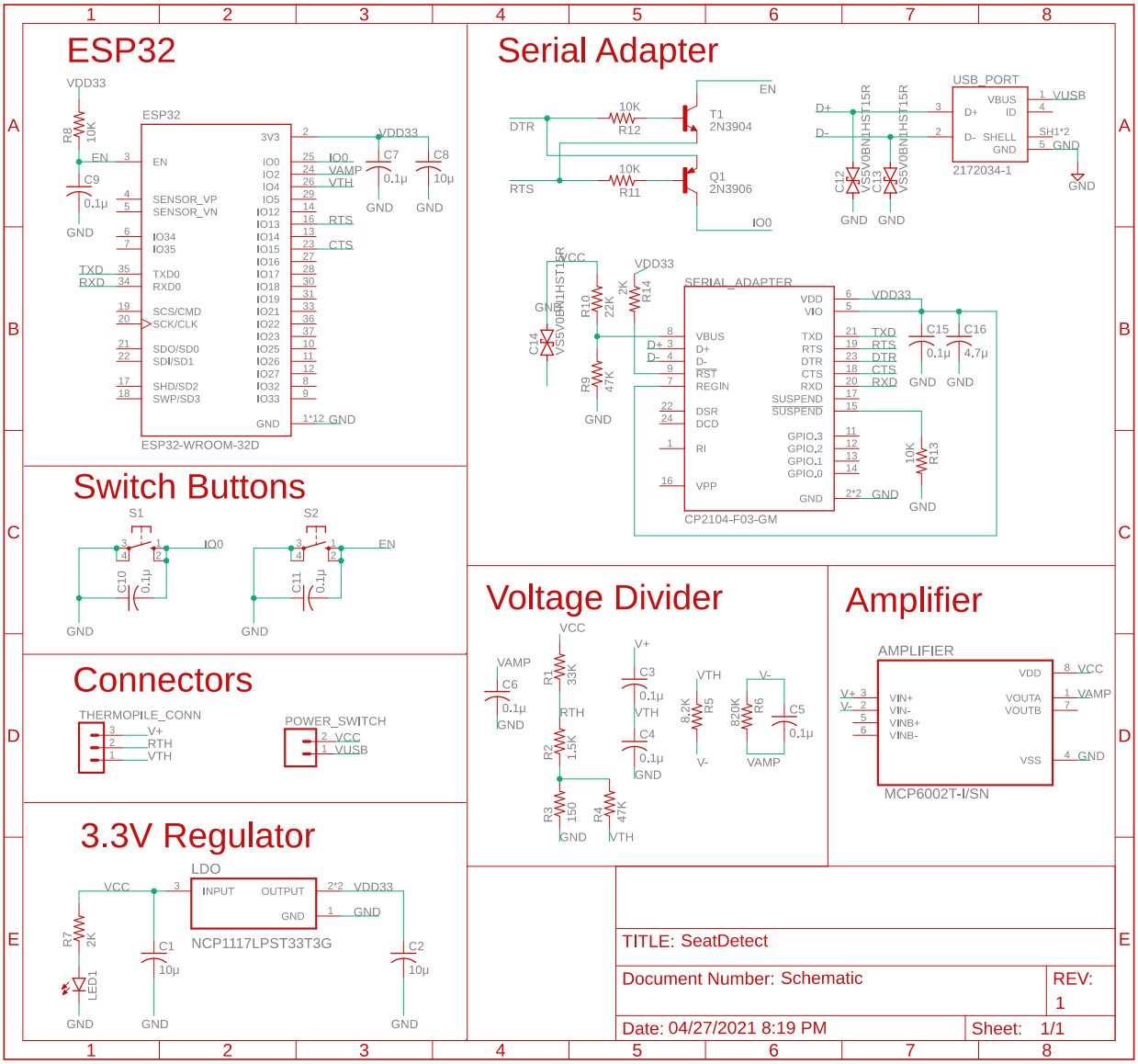

Figure 13. EAGLE Schematic

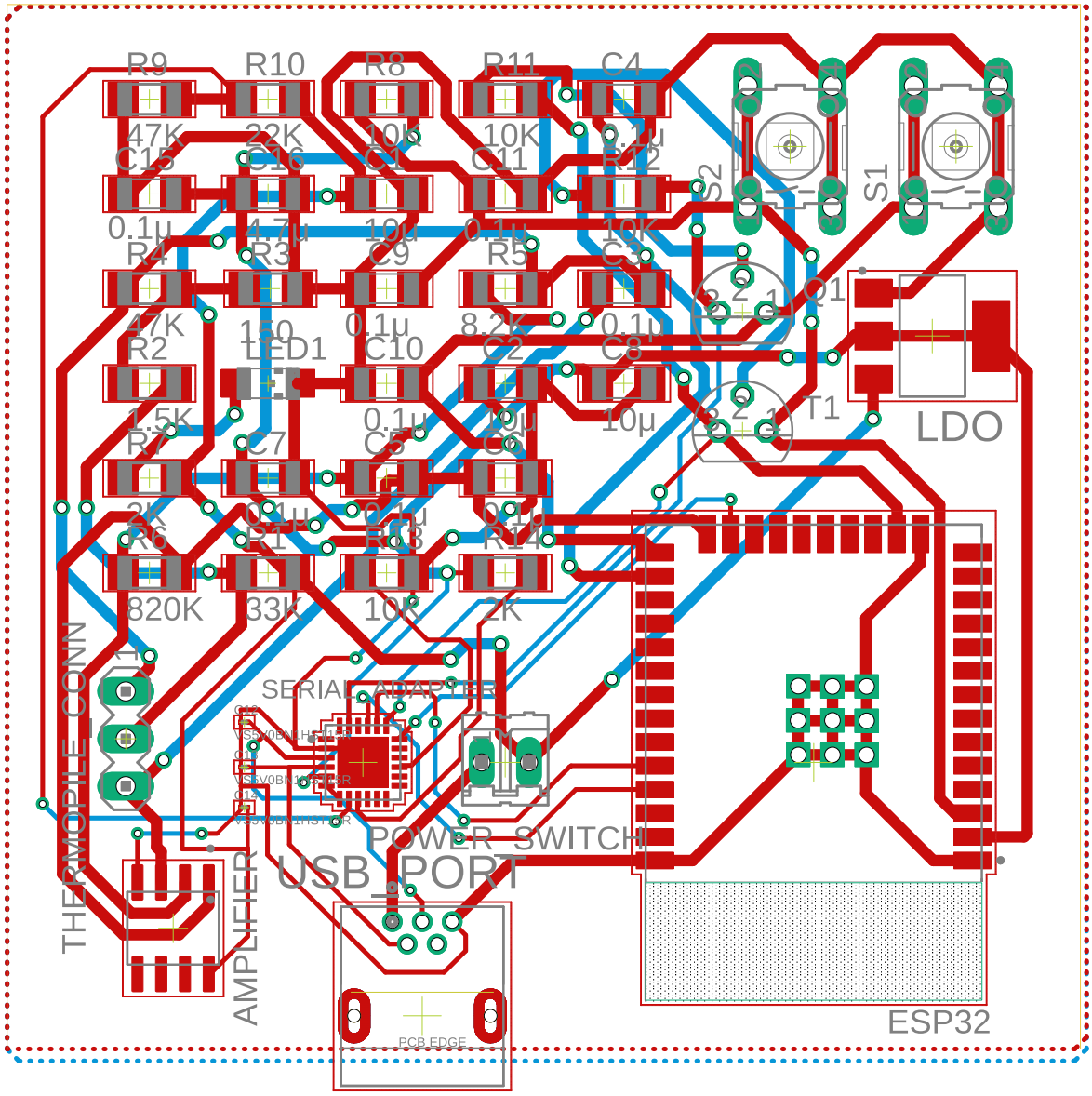

Figure 14. EAGLE PCB Layout

Appendix B Firmware Development

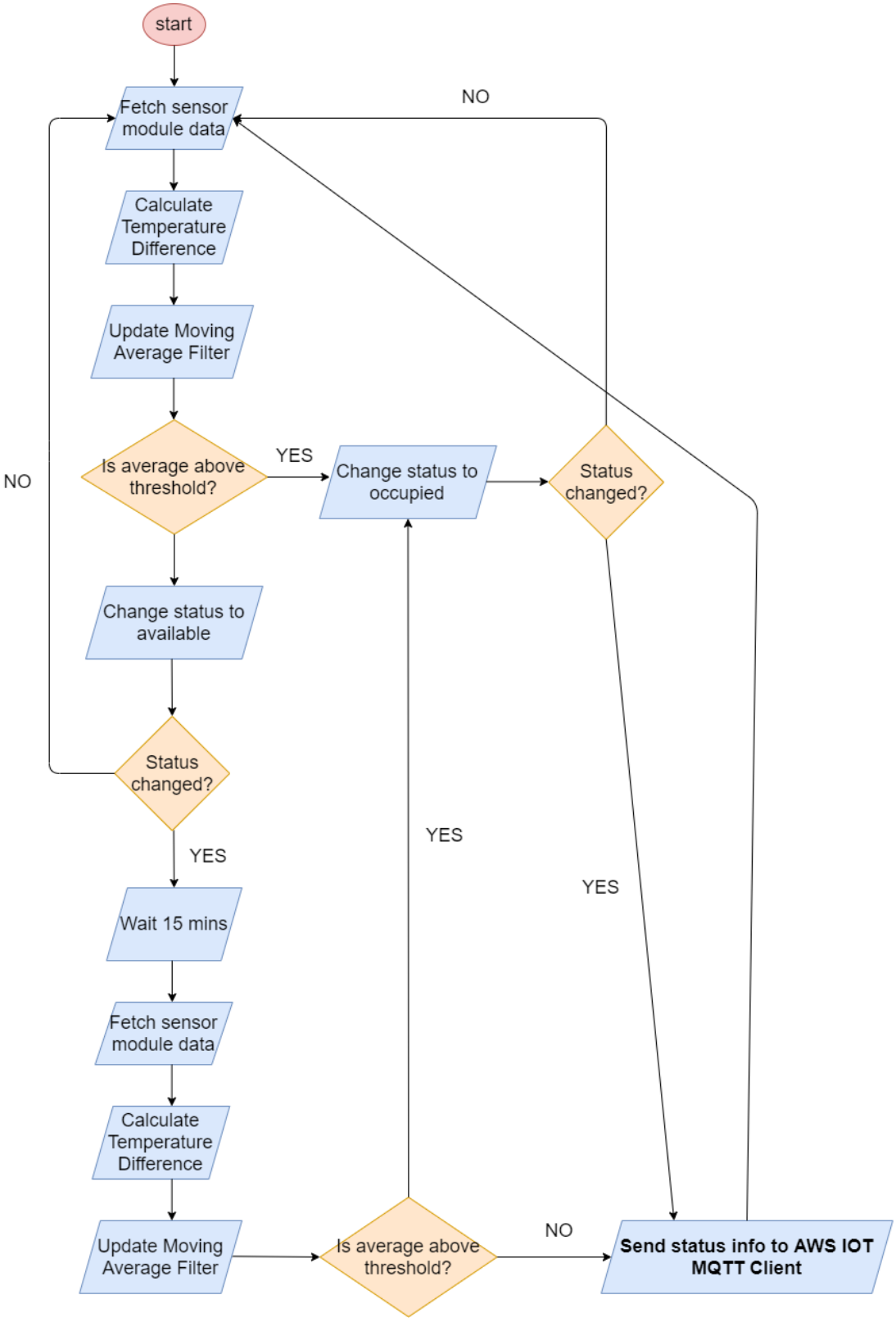

Figure 15. Firmware Flowchart

Appendix C Software Development

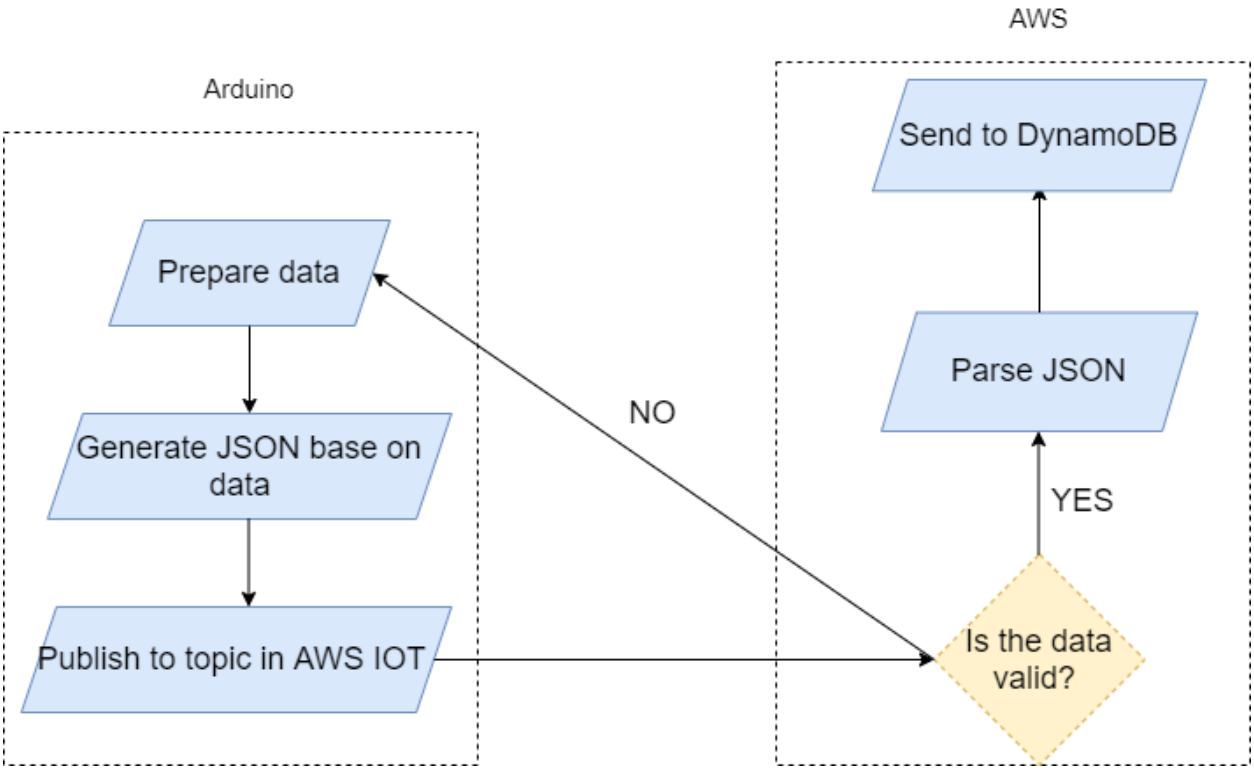

Figure 16. AWS Flowchart

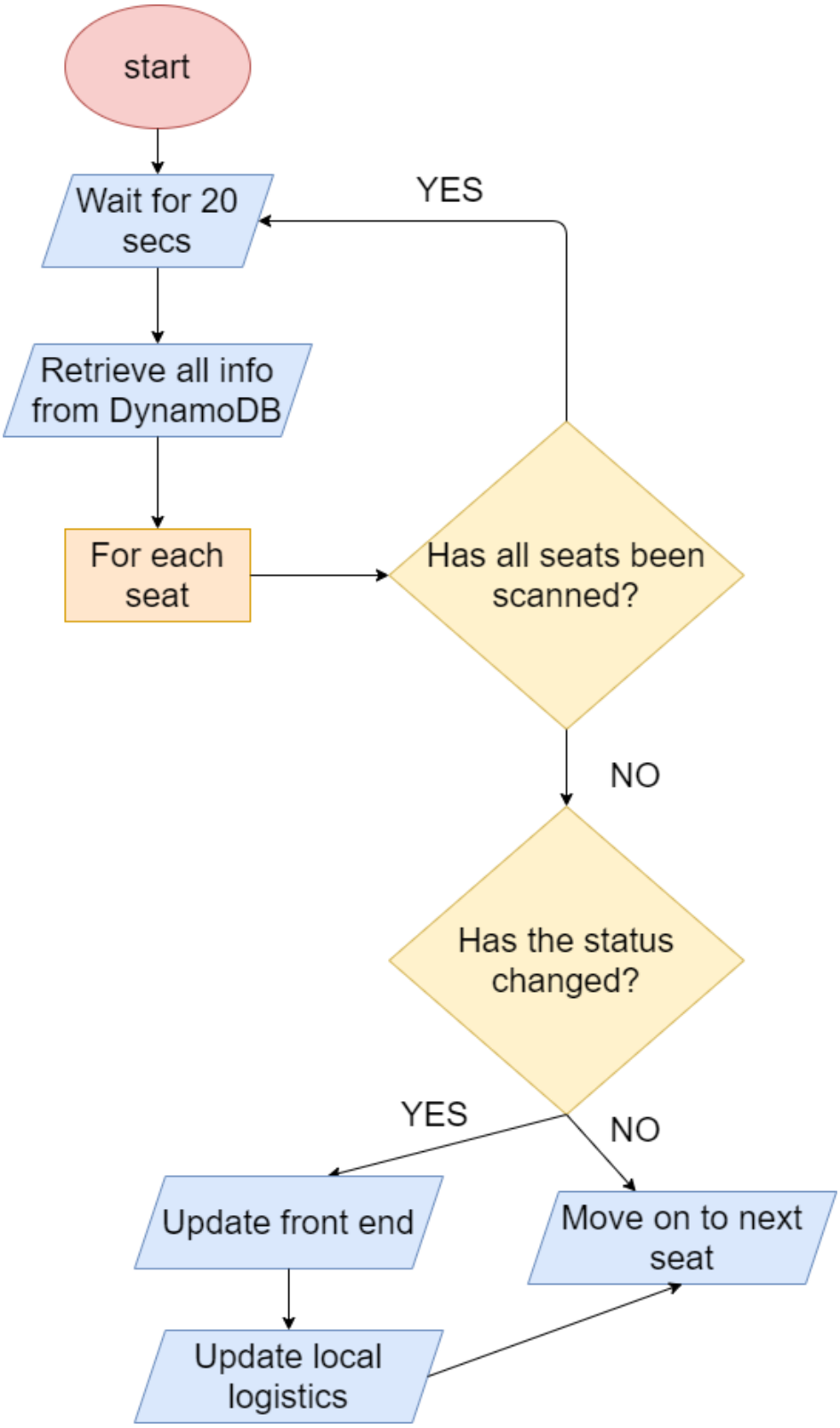

Figure 17. Application Flowchart

Appendix D Large Figures

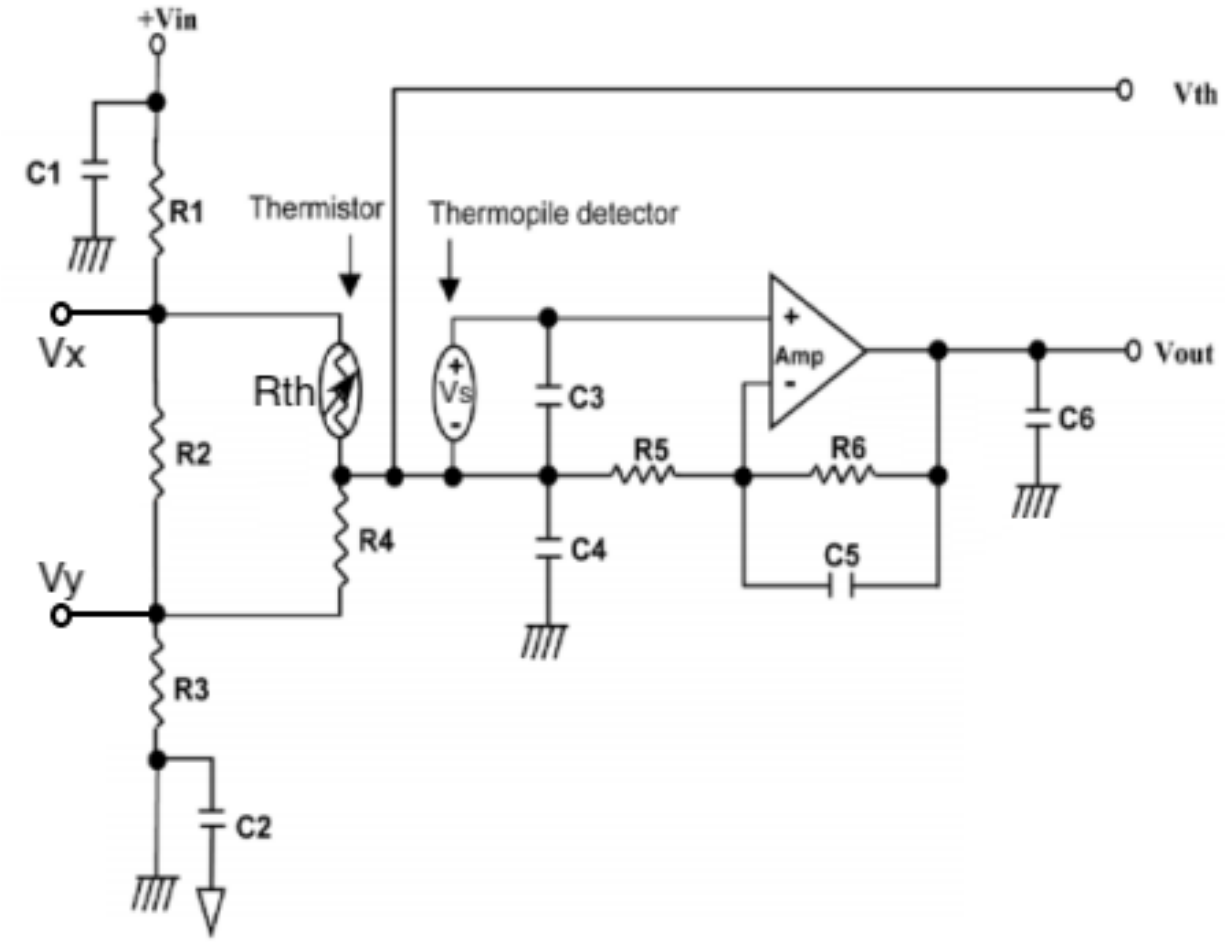

Figure 18. Sensing Module Circuit

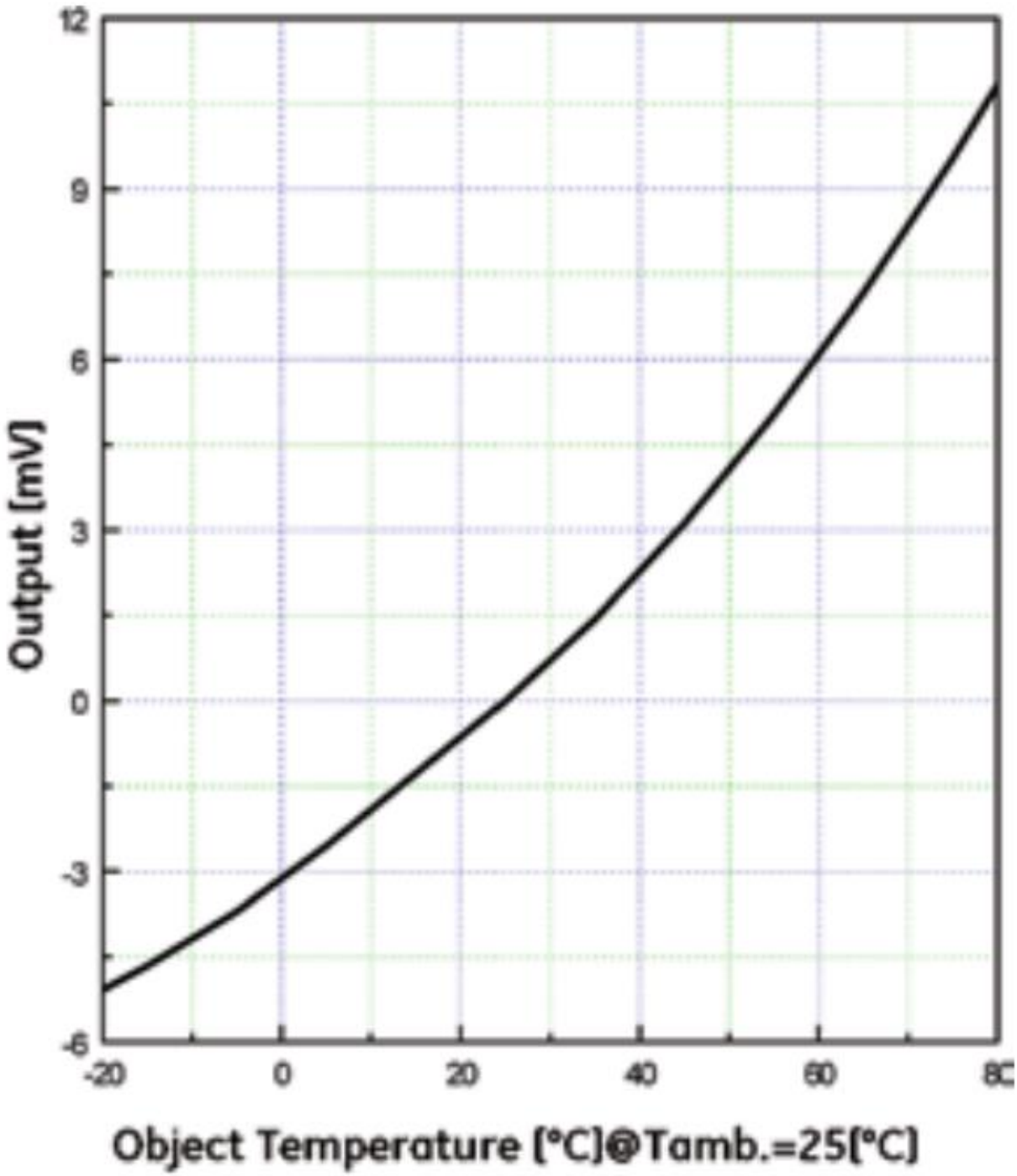

Figure 19. Thermopile Voltage Plot

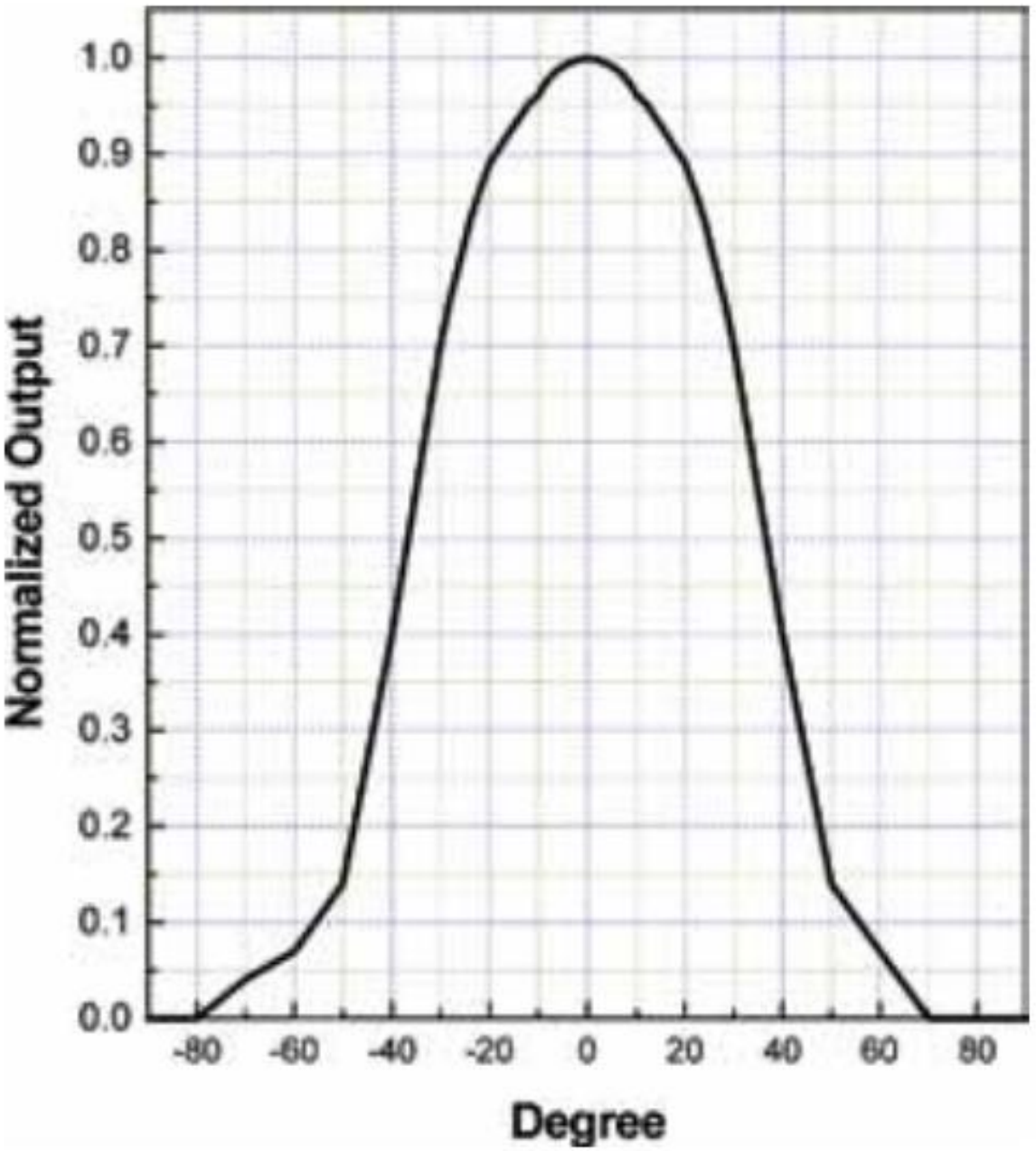

Figure 20. Sensor FOV Plot

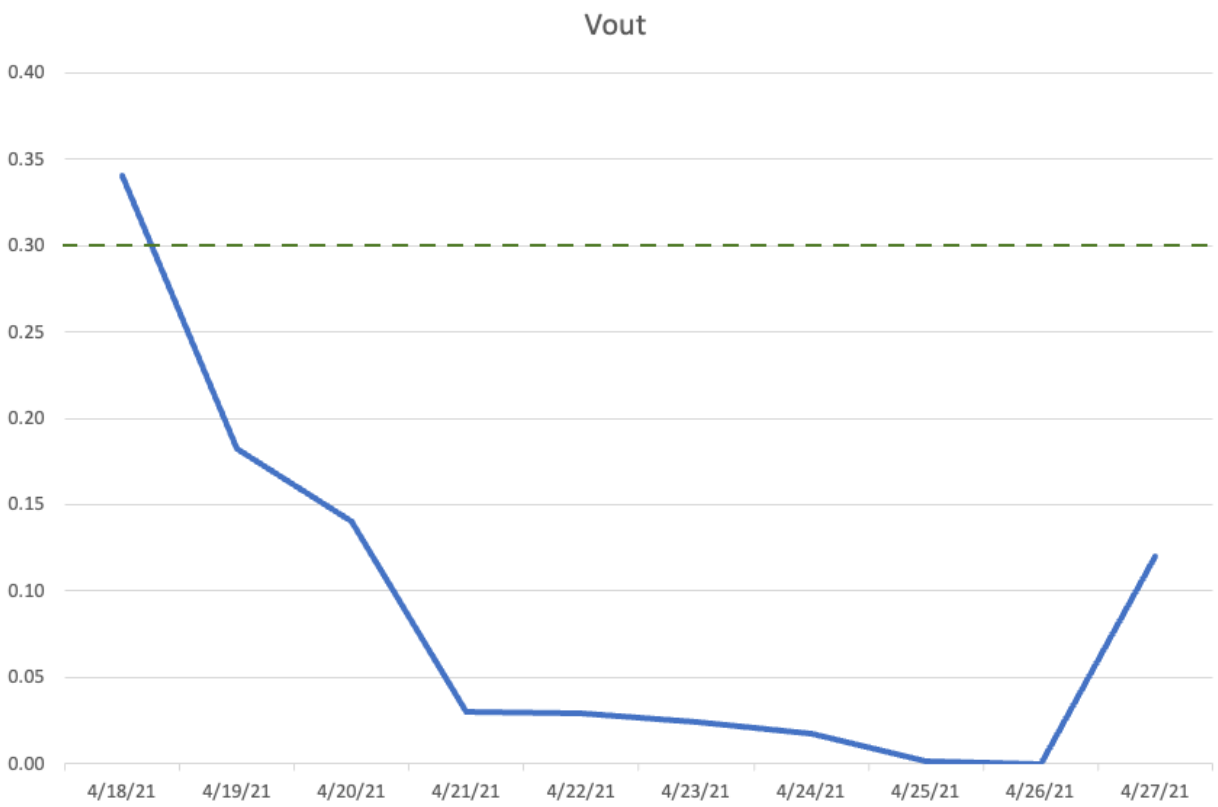

Figure 21. V_{out} Decrease

Appendix E Large Data Tables

T (°C)	R_{th} (kΩ)
-20	948
-15	716
-10	545
-5	419
0	324
5	253
10	198
15	157
20	125
25	100
30	81
35	65
40	53
45	46
50	36
55	30
65	20
70	17
75	14
80	12
85	10
90	9
95	7
100	6

Table 3. R_{th} vs. T

Part Number	Description	Manufacturer	Quantity	Unit Cost	Ext. Cost
-	Box PCB	LeMotech	1	$7.19	$7.19
-	Box Sensor	YXQ	1	$11.25	$11.25
C1206C104K5RAC7800	Capacitor 0.1µ	KEMET	10	$0.076	$0.76
C1206C475K4PACTU	Capacitor 4.7µ	KEMET	1	$0.30	$1.54
C1206C106K9PACTU	Capacitor 10µ	KEMET	4	$0.27	$1.08
ESP32-WROOM-32D (4MB)	ESP32 Module	Espressif Systems	1	$3.80	$3.80
NCP1117LPST33T3G	LDO Regulator	ON Semiconductor	1	$0.49	$0.49
APTR3216SURCK	LED Red	Kingbright	1	$0.32	$0.32
102-1031-BL-00100	Mini-B Cable	CNC Tech	1	$2.78	$2.78
2172034-1	Mini-B Port	TE Connectivity	1	$0.98	$0.98
61300211121	Pin Header 2 Position	Würth Elektronik	1	$0.13	$0.13
61300311121	Pin Header 3 Position	Würth Elektronik	1	$0.12	$0.12
SWI5-5-N-I38	Power Adapter	CUI Inc.	1	$6.30	$6.30
CRGCQ1206F	Resistor	TE Connectivity	14	$0.10	$1.40
CP2104-F03-GM	Serial Adapter	Silicon Labs	1	$1.72	$1.72
RA11131121	Switch Rocker	E-Switch	1	$0.57	$0.57
B3F-1000	Tactile Switch	Omron	2	$0.28	$0.56
ZTP-135SR	Thermopile	Amphenol Advanced Sensors	1	$5.11	$5.11
2N3906-AP	Transistor PNP	Micro Commercial Co	1	$0.18	$0.18
2N3904-AP	Transistor NPN	Micro Commercial Co	1	$0.18	$0.18
VS5V0BN1HST15R	TVS Diode	Rohm Semiconductor	3	$0.33	$0.99
20675-2	Wire Sleeving	TE Connectivity	1	$1.24	$1.24

Table 4. Parts Cost Breakdown

Appendix F Requirement and Verification Tables

Control Module		
Requirements	Verifications	Status
1. The control unit must be able to reliably transfer up to 10kB of data within 1 second to ensure 6.8kB can be sent out to convey occupancy status and cubicle ID.	1. a. Connect the control unit wirelessly to a computer as a wireless serial device. Add the name of the personal hotspot under WIFI_SSID[] and login credentials into WIFI_PASSWORD[] in Arduino code to connect to a Wi-Fi network. b. Program the control unit to transmit the data over the serial link and record the time in the control unit it took to transfer the data.	1. Yes (<1sec)
2. When the USB port is plugged into a computer, the device must show up as working.	2. Plug the device into a computer with the CP210x software drivers installed and verify it is listed as a serial device.	2. Yes
3. The control unit must be able to persist the data with the correct SeatID into the backend DynamoDB table of the software module within 10 seconds of receiving the signal.	3. a. Repeatedly occupy and leave one seat containing the sensor and b. Confirm that the record in the database has been updated.	3. Yes (<1sec)

Table 5. Control Module Requirements and Verifications

Sensing Module		
Requirements	Verifications	Status
1. The amplifier must amplify the voltage from the sensor by a factor of 100 ± 5% (gain of 100).	1. Measure the voltage before (V_{in}) and after (V_{out}) the op-amp while the complete circuit is under operation. V_{out}/V_{in} must be equal to 100.	1. Yes (101.7)
2. V_{out} must be between 80 and 520 mV.	2. Put the system under the highest and lowest room temperatures possible while monitoring V_{th} to see if it is within range. Stand in front of sensor and away from sensor while monitoring V_{out} to see if it is within range.	2. No (34.7mV)
3. V_{out} must achieve at least a 50% decrease when the human leaves the seat	3. Stand in front of sensor and away from sensor while measuring V_{out}. Divide the first measurement by the second and subtract it from 1 to see if it decreased by enough.	3. No

Table 6. Sensing Module Requirements and Verifications

Power Module		
Requirements	Verifications	Status
1. The subsystem must be capable of outputting a regulated 3.3V ± 0.1V at 750mA. 2. Maintain thermal stability below 125°C	1. Provide the subsystem with 5V from the power source and connect the regulated output to a resistive load pulling 750mA, and at the same time measure the steady state output voltage using an oscilloscope. 2. Use an IR thermometer to ensure the IC stays below 125°C	1. Yes (3.314V) 2. Yes (46.3°C)

Table 7. Power Module Requirements and Verifications

Software Module		
Requirements	Verifications	Status
1. The software module must reflect the status changes detected by the control module and display such a change to the end user within 30 seconds. 2. The web interface app must be able to establish a connection with the DynamoDB database. 3. The front end must contain comprehensive internal linking page.	1. Repeatedly occupy a singular seat and leave it and confirm if the status has been reflected to the end user each time. 2. Manually change status values in DynamoDB to see if the status has been reflected to the end user each time. 3. Click through all combinations of hyperlinks and ensure that all hyperlinks direct to the correct page	1. Yes (<1sec) 2. Yes 3. Yes

Table 8. Software Module Requirements and Verifications