SeatDetect

By

Owen Brown (owenmb2)

Yue Li (yuel7)

Huey Nguyen (hueyn2)

Design Document for ECE 445, Senior Design, Spring 2021

TA: Haoqing Zhu

March 4, 2021

Project #18

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 up. Also, 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.

To solve this problem, we will design and implement a motion-sensing device for each table, which can tell the user whether the cubicle is occupied, in advance. The occupancy status will be transmitted 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. The user should be able to see the change and walk to the desired location and occupy the free seat.

In addition to this main problem, the solution will also solve additional issues that revolve around library policies. For instance, the solution makes it easier for the library staff to enforce policies such as a maximum amount of inactivity time within each cubicle, which eliminates the issue of holding spots for others, or themselves.

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 our problem statement, which is to detect humans. Similar to how the parking spot will communicate to the driver whether the spot is occupied, our 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. To efforts of increasing accessibility and functionality, our project includes an additional web user interface as the primary method in identifying availability spaces.

1.3 Physical Design

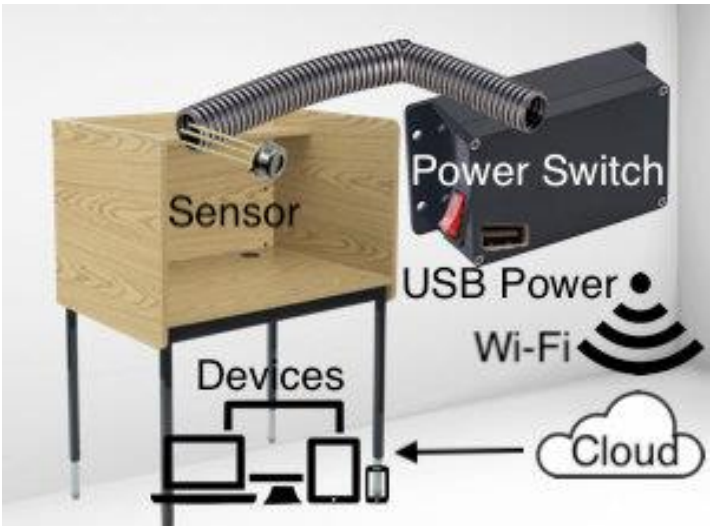

Figure 1. SeatDetect Physical Design Diagram

Figure 1 represents the physical design diagram for SeatDetect. Its main components include the electronics box, booth and sensor where the end users will be detected, and the database system.

1.4 High-Level Requirements List

- An accuracy of occupancy status over 95% on repeated tests of the same booth is the benchmark. This will be measured by running multiple occupancy sensing tests on the same sensor unit and counting how many successful readings there are out of total attempts.
- 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.
- 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.

2 Design

2.1 Block Diagram

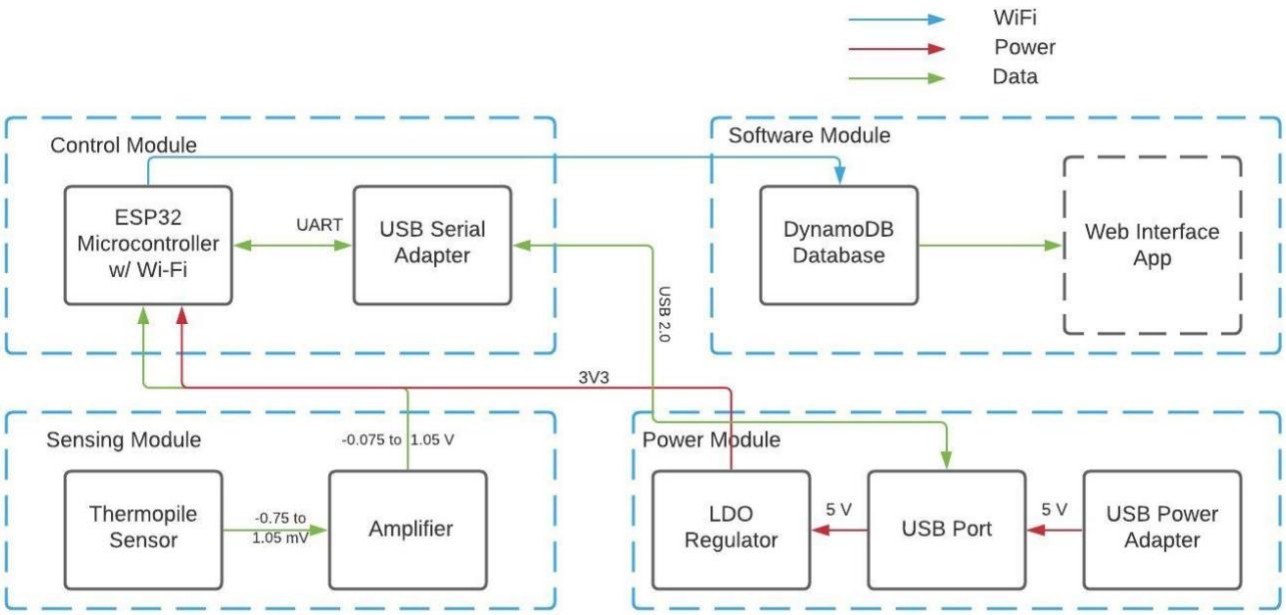

Figure 2. Block Diagram

Figure 2 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 system ensures that the system can be powered continuously all day and night with the proper 3.3V. The control unit contains a microcontroller with integrated Wi-Fi, which will handle the data received from the sensing module. Lastly, the software module displays the status data to the end users, the students.

2.2 Physical Design

The physical design considers the PCB dimensions as well as the dimensions of the chosen sensor. Power comes from the USB port that is attached to the board and will stick out of the physical design. This allows for the USB mini-B port to allow powering the device and interacting with the microcontroller. Beyond the physical device, the device interacts with the database which relays data to a web user interface over Wi-Fi.

2.3 Functional Overview

2.3.1 Control Module

The control module interfaces with every single module. It is the module that needs power. It also relays preprocessed data from the sensing module to the software module and includes dynamic memory storage in the form of an SD card.

2.3.1.1 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.

2.3.1.2 USB Serial Adapter

The USB serial adapter is necessary to enable the programming of the ESP32. It takes in the upload of code from a computer by converting the USB data to serial data that the ESP32 can then interpret.

Control Module	
Requirements	**Verifications**
The control unit must be able to reliably transfer up to 0.02kB of data within 1 second to ensure 0.0068kB can be sent out within mS to convey occupancy status and cubicle ID.	Connect the control unit wirelessly to a computer as a wireless serial device. Generate 4kB of random bytes using Python (or similar), then 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.
When the USB port is plugged into a computer, the device must show up as a working.	Plug the device into at least 2 windows computers, and a UNIX based computer and verify it is listed as a serial device without additional software.

Table 1. Control Module Subsystem Requirements and Verifications

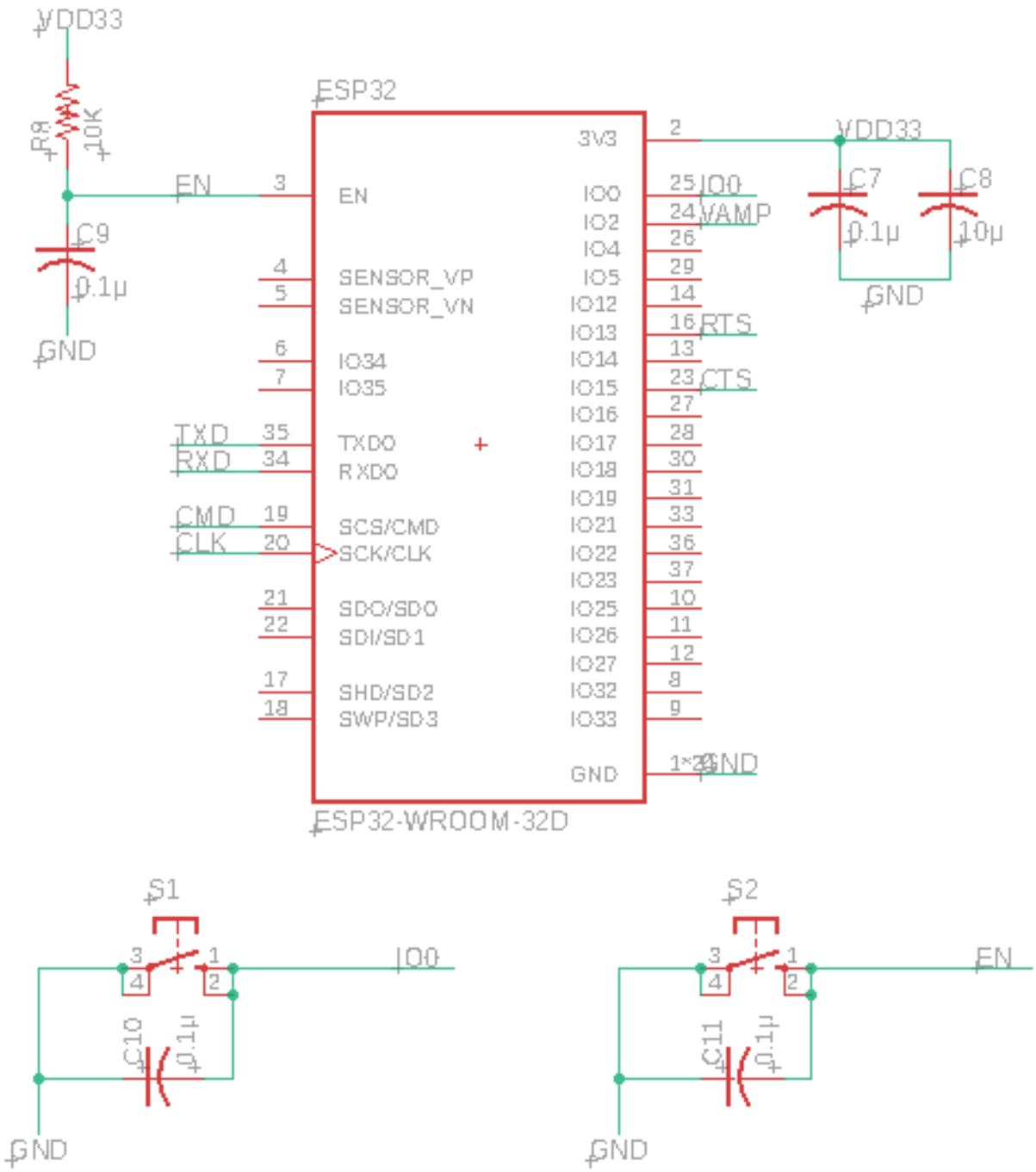

Figure 3. ESP32 Circuit Schematic

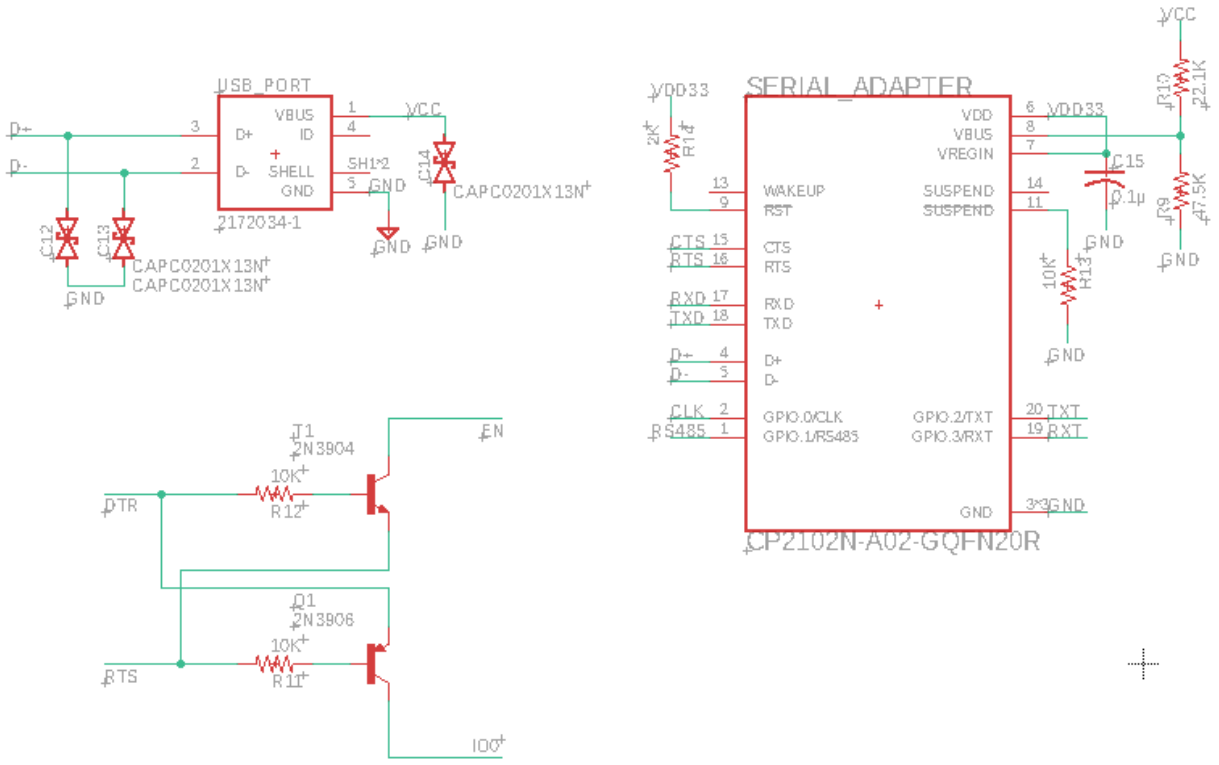

Figure 4. Serial Adapter Circuit Schematic

2.3.2 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.

2.3.2.1 Thermopile Sensor

ZTP-135SR is a thermopile sensor that detects human presence by thermal energy correspondence.

2.3.2.2 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.

Sensing Module	
Requirements	**Verifications**
The amplifier must amplify the voltage from the sensor by 100x (gain of 100).	Measure the voltage before (V_{in}) and after (V_{out}) the op-amp while the complete circuit is under operation. $\frac{V_{out}}{V_{in}}$ must be equal to 100.

Table 2. Sensing Module Subsystem Requirements and Verifications

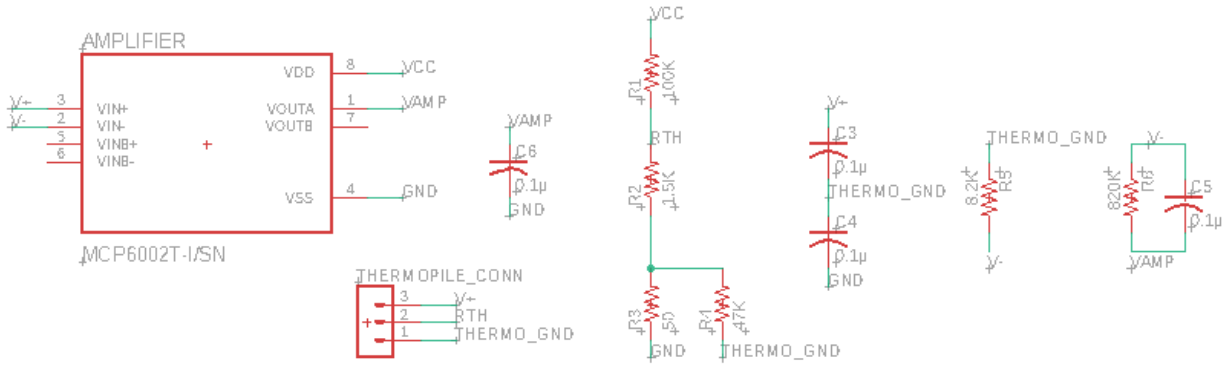

Figure 5. Sensor Circuit Schematic

2.3.3 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 will be used to power the whole system. Because the addition of sensing modules does not add much additional power requirements from the control module, the addition of sensing modules will not require significantly more power from the power module. In the scenario where enough sensing modules are adding to warrant additional control modules due to the I/O limitations of the ESP32, there would be also be a need for more power modules, and therefore the whole system would be duplicated except the software module. The following requirement does not cover the ladder case.

2.3.3.1 USB Power Adapter

The power adapter converts the grid electricity into DC power that is used for the whole system.

2.3.3.2 USB Port

The mini–B female USB port receives power from the power adapter and also 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 will not be, and vice versa.

2.3.3.3 LDO Regulator

The LDO regulator steps down the voltage from the power adapter for the ESP32.

Power Module	
Requirements	**Verifications**
The subsystem must be capable of outputting a regulated 3.3V ± 0.1V at 750mA.	Provide the subsystem with 4V from a bench power supply 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.
Maintain thermal stability below 125°C	Use an IR thermometer to ensure the IC stays below 125°C

Table 3. Power Module Subsystem Requirements and Verifications

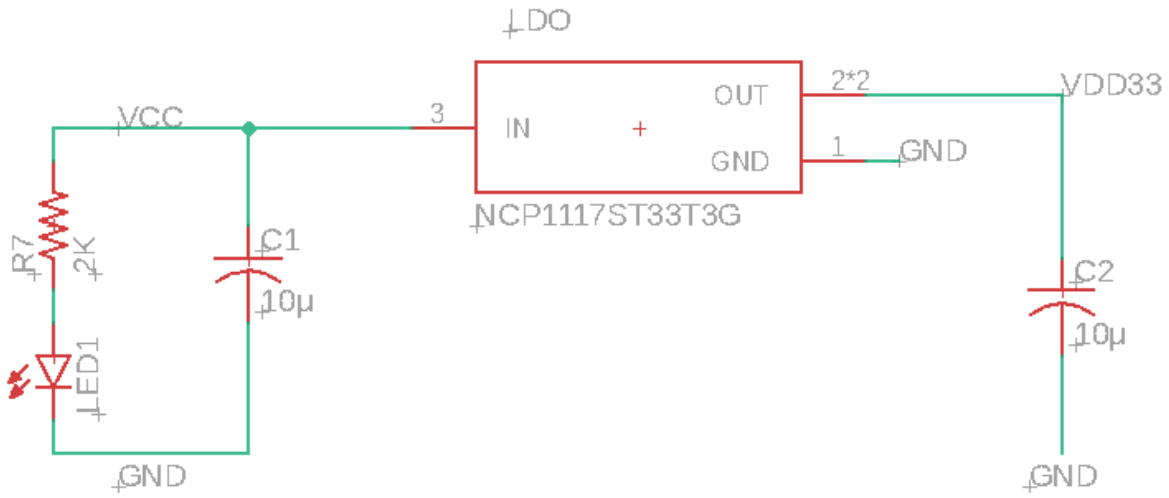

Figure 6. Power Circuit Schematic

2.3.4 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.

2.3.4.1 DynamoDB Database

Stores the data and the ability to view the status of a seat (occupied by a person, personal items, inactivity, etc.).

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

Table 4. DynamoDB Table

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 will store the perspective occupancy status and the time which it was last updated. As a table it will look like **Error! Reference source not found.**.

2.3.4.2 Web Interface App

The Web Interface App allows the user to view the map of Grainger Library (for now) and its corresponding tables on each floor.

2.3.4.2.1 Front End

The front end of this web application will be constructed using Python Flask coupled with HTML and CSS. The front end of this application will have 2 versions, one for the web browser, another for the mobile phone. The web browser version will look like Figure 7 and the mobile version will look like Figure 8.

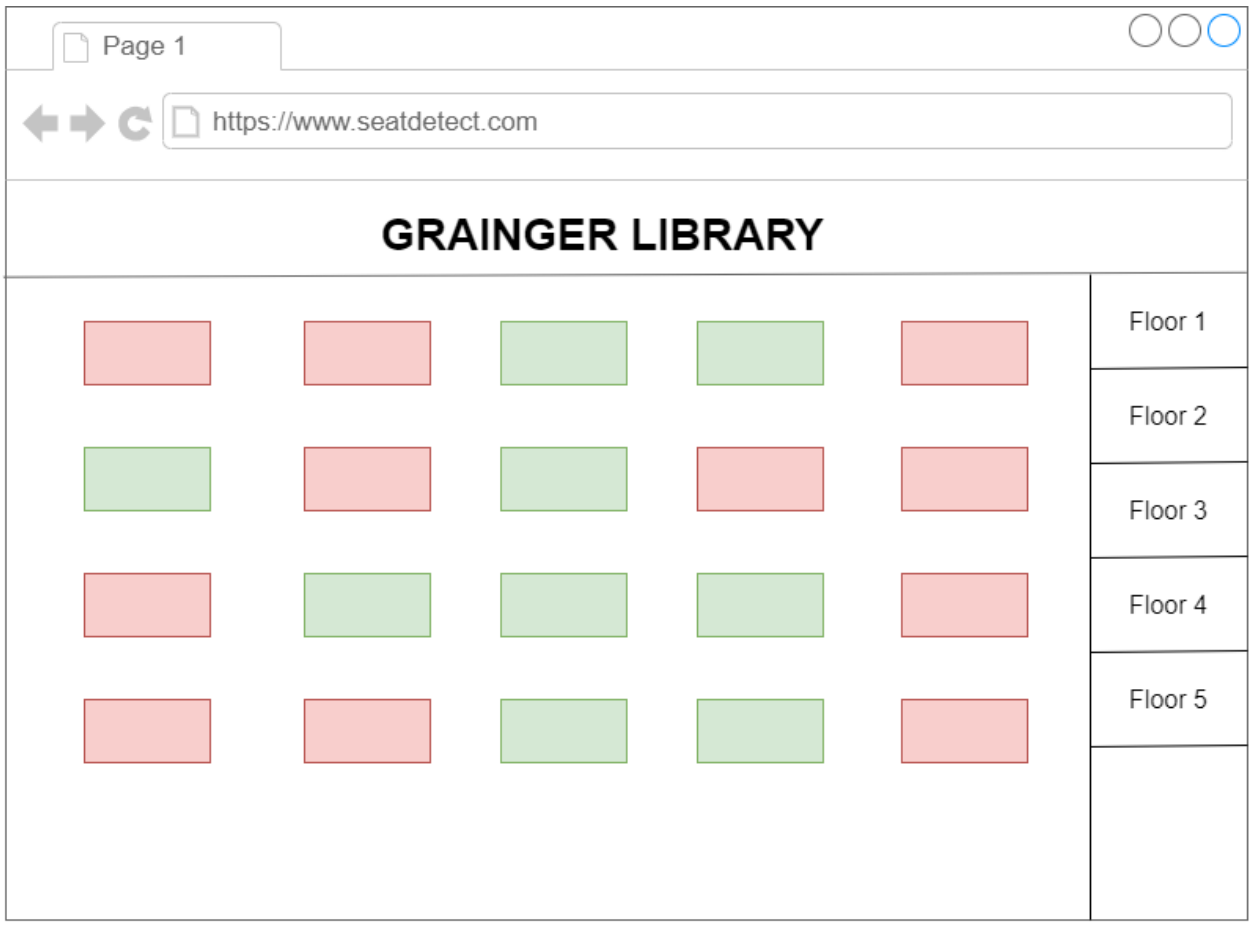

Figure 7. Browser version of the app

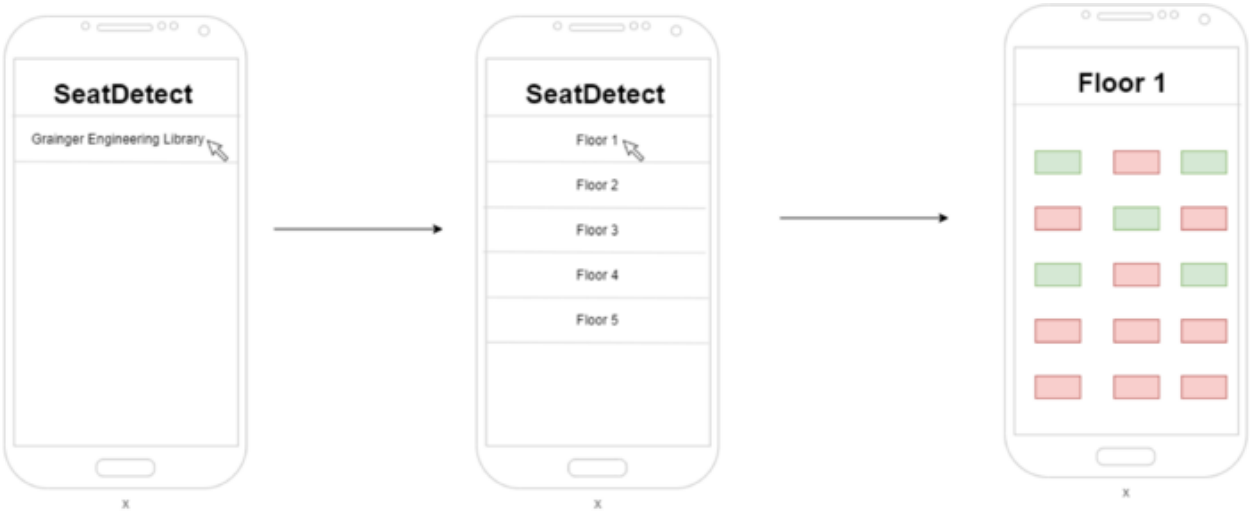

Figure 8. Mobile version of the app

2.3.4.2.2 Back end

The back end for this web application will be constructed using Python with the package Boto3. The logic consists of an infinite while loop that runs every 20 seconds. The process will scan 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 will move on to the next seat. The flow chart of the back-end application will look like Figure 9.

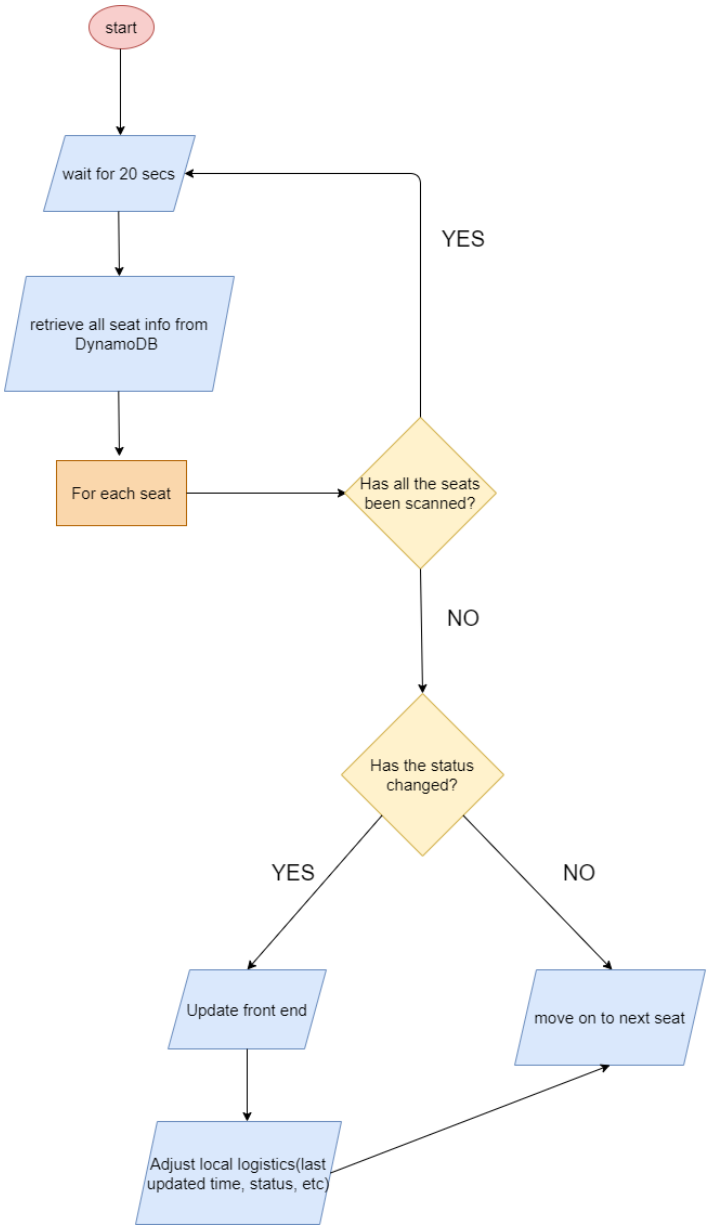

Figure 9. Application Flowchart

Software Module	
Requirements	**Verifications**
The software module must reflect the status changes detected by the control module and display such a change to the end user.	Occupy the seat and unoccupy it multiple times to see if the status has been reflected to the end user each time.
The web interface app must be able to establish a connection with the DynamoDB database.	Manually change status values in DynamoDB to see if the status has been reflected to the end user each time.

Table 5. Software Module Subsystem Requirements and Verifications

2.4 Tolerance Analysis

One difficulty of this project deals with the accuracy of the thermopile sensor. To achieve a 95% accuracy of detection, we need to make sure that the thermopile sensor is as accurate as possible. We need to find out exactly how accurate it is and if the sensor can indeed differentiate between body heat and room temperature in the case when room temperature is high. If the sensor is unable to do that, we need to find an additional sensor to pair with the thermopile sensor to ensure the accuracy.

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 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 6. Labor Cost Breakdown

Table 6 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.

3.1.2 Parts

Part Number	Description	Manufacturer	Quantity	Unit Cost	Ext. Cost
ESP32-WROOM-32D (4MB)	ESP32	Espressif Systems	1	$3.80	$3.80
NCP1117ST33T3G	LDO Regulator	ON Semiconductor	1	$0.48	$0.48
	Red LED		1		
	Capacitor		12		
MCP6002T-I/SN	Operational Amplifier	Microchip Technology	1	$0.33	$0.33
-	PCB	PCBWay	1	$5.00	$5.00
MA03-1	Pin Header				
	Resistor		14		
B3F-1000	Tactile Switch	Omron	2	$0.28	$0.56
ZTP-135SR	Thermopile	Amphenol Advanced Sensors	1	$5.11	$5.11
VS5V0BN1HST15R	TVS Diode	Rohm Semiconductor	3	$0.33	$0.99
102-1031-BL-00100	USB A to Mini-B Cable	CNC Tech	1	$2.78	$2.78
2172034-1	USB Mini-B Port	TE Connectivity	1	$0.98	$0.98
SWI5-5-N-I38	USB Power Adapter	CUI Inc.	1	$6.30	$6.30
CP2102N-A02-GQFN20R	USB Serial Adapter	Silicon Labs	1	$1.33	$1.33

Table 7. Parts Cost Breakdown

Table 7 represents the parts cost breakdown for the SeatDetect system.

3.1.3 Total

Total Cost:

$54,000 (Labor) + $ (Parts) = $

3.2 Schedule

Week	Owen	Huey	Yue
3/8	Finalize PCB Schematic Finalize I/O, control unit, and power supply parts Finish ordering parts	Finalize PCB Schematic Finalize I/O, control unit, and power supply parts Finish ordering parts	Finalize PCB Schematic Start UI design for web interface and base for app Finish ordering parts
3/15	Finish PCB layout Start assembling PCB	Finish PCB layout Start assembling PCB	Work on building / designing web app Design software state diagrams
3/29	Work on control unit design Implement power supply with hardware	Work on control unit design Set up database and connect to web app	Finish web app Finish display and I/O testing Set up database and connect to web app
4/5	Finish power supply and control unit implementation Start testing and verification	Finish control unit implementation Assist finalizing and debugging Wi-Fi software	Combine, test, and verify ESP32 Wi-Fi and web app Start testing and verification
4/12	Full-system testing	Full-system testing	Full-system testing
4/19	Work out any bugs/issues and prepare for mock demo	Work out any bugs/issues and prepare for mock demo	Work out any bugs/issues and prepare for mock demo
4/26	Demo, system testing, and start final paper	Demo, system testing, and start final paper	Demo, system testing, and start final paper
5/3	Finish final paper	Finish final paper	Finish final paper

Table 8. Schedule

Table 8 represents the planned schedule for each team member for the remaining 8 weeks of the semester.

4 Ethics and Safety

Ethics and Safety will be imperative to successfully carry out our project. During this difficult time of COVID-19, it is especially important that we follow closely IEEE Code of Ethics #1 [3], that we do 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 our 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 we follow the guidelines of IEEE Code of Ethics #5, which suggests that we need "to seek, accept and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic" [3]. We as a team need to take advantage and make the most out of the weekly TA meetings and be proactive when possible. Furthermore, we plan on following this guideline and conducting surveys of our prototype once it starts working to receive feedback to further improve our product. One ethics concern one would consider relates to the issue of privacy. It can be a source of concern because we are 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.

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