Room Occupancy Management System (ROMS)
Build and Evaluation Report

ESE Group 2

Arpan Dhamane, Chiamaka Aligwekwe, Daniel Takyi, Dwijen Kapadia

A: Project Report

Introduction

Purpose and Goals

The University of Regina Facilities Management department is in the process of updating its dashboard for a new energy consumption management system. The primary objectives of our project are to find data that can relate to occupancy of a fixed area, obtain the data without interfering with everyday school events, and create a final product that is of use to the University of Regina's new energy consumption dashboard. Our system, the Room Occupancy Management System (ROMS), is a portable and easily relocatable device that collects data related to occupancy from its external environment, formats the data, and uploads it to a specified dashboard.

Requirements:

- System has one sensor module containing a CO2 sensor, Temperature sensor and Noise level sensor
- Measure CO2 levels to the accuracy of ± 50ppm (0 - 5000ppm range)(±1bit ADC)
- Measure noise level to the accuracy of ± 1.5dB (30dBa to 130 dBa)(±1bit ADC)
- Measure temperature to the accuracy of ± 1°C (−55°C to 150°C Range)(±1bit ADC)
- Sensor data saved in a .csv file every 1 minute
- Files (.csv) transferred to dashboard every 5 minutes
- Using FTP (File Transfer Protocol) to transfer files from Raspberry Pi to dashboard via wireless connection
- Using UART to transfer sensor data from STM32 microcontroller to Raspberry Pi via USB
- System calibrates one of its outputs depending on the user selected room.
- Using data to calculate occupancy with CO2, noise and temperature.
- System will be held together in a non-conductive enclosure (size = 11-51/64" Length x 7-55/64" Width x 6-9/32" Height)

Methods and Material

System Design Process

ROMS started off being visualized at the system level. After identifying different ways to solve the client's problem a system block diagram was created. The system level block diagram was kept at the highest level possible, to allow for freedom further down the line in our design process. We kept blocks at the module level so that they could be broken down later in lower design levels. After settling on a system block diagram, the group started to break it down to lower levels and tackle each subsystem of ROMS on its own. We split up the subsystems so that each individual would . Group members with a more detailed background in embedded systems, would work on the Microcontroller and Communication Module subsystems. Other members were more well versed in power systems. So they would be put in charge of the power subsystem and Sensor Module subsystem. Each individual in the group, defined their given subsystem down to the component level and came up with multiple alternatives for their subsystem. We would then convene as a group to decide which alternative to use for our final design. For example, we used a morphological chart to decide what device should be used for the Communication Module. The devices we compared were a STM32 with wifi, adding a wifi module to our wifi-less STM32, an Arduino Uno, and a Raspberry Pi. We compared these devices by clock speed, RAM, GPIO ports, connectivity, and cost. As a group, we came to a decision on the Raspberry Pi. After doing this for each subsystem, we had our system down to the component level. A gantt chart was then created to organize the group for our building stage.

System Build Process

We strategically implemented a build process in regards to getting occupancy as our overall objective. At first, the team researched what readings are required to obtain occupancy without obstructing everyday school life. With the advice of Doug Wagner and Paul Laforge, we decided to go with obtaining Carbon Dioxide (CO_2), temperature and noise level readings. For each reading to be obtained from an indoor environment, we researched on the different sensors that would give us this data. At the same time, we also looked into the components that would receive and display the sensor data onto a dashboard. It was recommended to use two components; a microcontroller and a card sized computer. We ultimately went with the Nucleo-F103RB microcontroller as our sensor data receiver and the Raspberry Pi 3 B+ as the dashboard communicator. During the process of testing the sensors, our original CO_2 sensor (DFRobot SEN0159) had a preheat time of 30 minutes to 48 hours, which made us switch to our second choice for the CO_2 sensor (DFRobot SEN0219) for a faster preheat time but at the cost of less accuracy. The choice to make all sensor inputs analog was due its simplicity and economy of pins on our microcontroller. During this time, we planned on using the LM35 temperature sensor because we had used it in our previous labs. We also chose the DFRobot SEN0232 as our noise level sensor. It quickly came to our attention that the temperature sensor would need a buffer for the conversion in the program to work correctly.

The group took a trip during reading week to integrate the system together and test it at the University of Regina lab rooms. Multiple problems had occurred and needed attention; CO_2 sensor was damaged during shipping, University of Regina did not allow permission for FTP

usage, and the noise level sensor was sending the wrong output as recommended by the NIOSH Sound Level Meter app. We first ordered a new CO_2 sensor, then used the University's Digi-Sense meter to get reference data for calibrating our noise level sensor. Another decision was made to build a PCB (Printed Circuit Board)/protoboard for both the user interface (LCD display, user push button and dial). After returning back home, the individual work was assigned and completed in a 2 week period. The PCB was built and implemented to connect and power the microcontroller, sensor module and user interface. Initially, we planned on building our own power supply but were advised against it due to major safety issues and long wait times for verifications from CSA (Canadian Standards Association). Alternatively, we powered the system with two 20,000 mAh USB-A power banks, one for the Raspberry Pi and another for the rest of the hardware.

The enclosure was commercially bought due time constraints and the 3D print job failure that occurred with the MakerBot at the University of Regina. We created holes for passive ventilation for all the internal hardware because the clear commercial lid of the enclosure was providing an airtight seal. Separate smaller holes were placed to mount the sensors externally, and a mounting panel was placed inside the enclosure to hold the PCB and the Raspberry Pi in place. The user interface was soldered onto a protoboard and mounted near the clear lid using L-shaped brackets. During our final calibrations, we tested all sensors and softwares to work simultaneously. To calibrate the sensors, we used online sources, home thermostat, and previous data from the Digi-Sense meter because we did not have access to more accurate equipment (Covid-19 restrictions). The time constraint for implementing a PCB made us make a tough but important decision to order it from a manufacturer in the USA. This cost us nearly $200 CAD more than what it would have cost if we ordered from a manufacturer in China. The cost and scarcity of the CO_2 sensor was also another reason we did not go for a more accurate and faster sensor. To build one system, our costs were approximately $500 CAD.

System Work Process

To power on the system, the user needs to push the power button on both power banks until their respective LEDs turn on. After being powered on there is a delay in the system that makes the user wait for a time before the system is ready to start. This is done because of the CO_2 sensor's preheat time. After the delay, the user is able to choose the room they are in from a selection of various rooms. They can span through the different options using the potentiometer as a knob, and they can confirm their choice using the push button. Once the room is selected, the system enters the task kernel, and will continue to switch between tasks until the system is powered down.

After a certain time, the sensor begins to collect analog data from the room it is placed in and sends the raw data values to the microcontroller. In the microcontroller, the sensor data values are converted to digital values, and then to each sensor's respective units using the conversion formulas stated in the datasheet of each sensor. The temperature sensor (LM35) is outputting a millivolts value that is converted to degrees Celsius using the scale factor which was given in the datasheet. The carbon dioxide sensor (SEN0219) gives out an analog output of 0.4V to 2V prior to being converted to parts per million(ppm) units and the noise level sensor (SEN0232) was giving out 0.6V to 2.6V output prior to being converted to A-weighted decibel

(dBa) units. These converted digital values are then displayed on the LCD screen. The lcd screen displays the room number the sensor is in (which is selected based on user selection option) , the temperature(in Celsius), CO_2 level(in ppm) and noise level(in dBa) the room the system is placed in. While the sensor data values are sent to the lcd screen to be displayed, they are also sent to the communication module along with the date and time the measurements were taken.

In the communication module, the raspberry pi collects the sensor data values along with the room selected by the user and saves them into a comma-separated value (CSV) file, this process is repeated every 5 minutes. The raspberry pi then creates a device number for the system which is stored into the CSV file, having its own separate column. The updated CSV file containing the date and time the data values were collected, along with the sensor measurements, the room location and the device number is then sent to the simulation website we created to act as the university dashboard via FTP (file transfer protocol). This process is done every 1 minute.

Outputs & Measures of System

Our system measures temperature, carbon dioxide, and noise level. The sensors are outputting degrees Celsius, parts per million (ppm), and A-level weighted decibels (dBa), respectively. One of the outputs of our system is the Liquid Crystal Display (LCD). These sensor values are displayed on the LCD screen in their respective units. Their values are updated on the LCD screen approximately every 1 millisecond. The Raspberry Pi is outputting a comma separated value (.csv) file which displays the data gathered from the sensors, along with the room it's getting data from, the time, the date, and the device number. The device number is a unique number that is given to identify the device from which the data is coming from. This allows for a user viewing the dashboard to see what device the room data is coming in from.

Regarding the accuracy of the sensors, we compared the sensor data to external sources as explained earlier. The CO_2 sensor (accuracy of was displaying values of approximately 450 ppm to 480 ppm (refer to "arpan_backyard_data.pdf") when exposed to the outdoor environment (in our case, a backyard), but expected levels should be 250 ppm to 400 ppm for outdoor air levels. Taking into account that the sensor was placed on a window ledge and was close to a CO_2 source (gas stove) we justified our CO_2 sensor needed slight calibrations to output the correct readings. As for the temperature sensor, the same backyard data (refer to "arpan_backyard_data.pdf") where the outdoor temperature was 14℃. We also compared the temperature sensor to a home thermostat (refer to "**Figure 1**"). The resultant values indicated the temperature sensor did not need a calibration. For the last sensor, the noise sensor was calibrated during our noise level testing at the University of Regina ED435, where two data sets (first data from UofR Digi-Sense meter, second data set from NIOSH sound level meter app) were compared to the noise sensor output (refer to "**Figure 2**"). The sensor was then calibrated to match the values of the two data sets in its given environment. Further calibrations are expected to be for future implementation. The CO_2 sensor is expected to be superseded by other sensors which have a higher accuracy and an improved preheat time, but will be more expensive.

The battery is expected to last around 36 hours to 48 hours. This estimate was done through a system endurance test (refer to "battery_test.pdf"), as the system was constantly running for around 7.5 hours until a loss of internet stopped the dashboard process.

Figure 1.

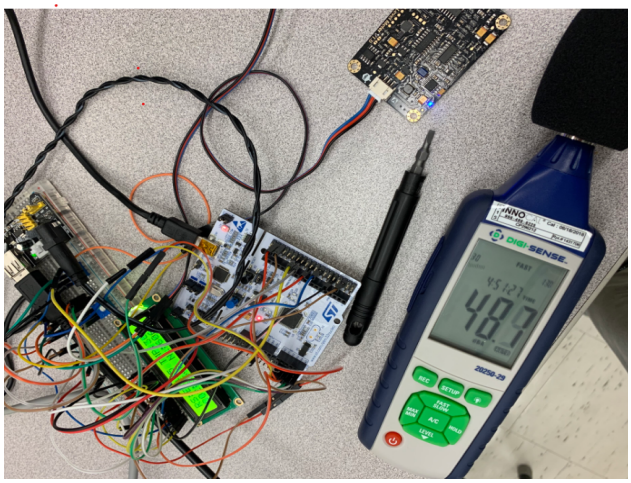


Figure 2.

Successes and Complexities

The sensors in the system are gathering data from the outside environment. We're using this data to find occupancy of the defined room. Due to pandemic restrictions, not enough raw data with specified parameters was collected. Because of that, we took a methodical approach to analyze the results. We looked at several research papers that related occupancy to the data metrics. All the relationships found in the research paper were developed according to the size of the testing room.

A paper published by Tianjin University determined the relationship between temperature and occupancy. The relationship was adapted to change with different seasons. The accuracy of this relationship was not tested with our project, but with the data we collected, it was concluded that with high occupancy the temperature will steadily increase.

Relating carbon dioxide with room occupancy involved many variables (i.e. temperature, air flow, room size). While doing extensive research, a relationship was proven in *Alessandro Franco's* research paper about *Measurement of CO_2 Concentration for Occupancy Estimation in Educational Buildings with Energy Efficiency Purposes*. This paper reviews how this relationship can vary depending on many external factors such as room size, room temperature and the air flow ventilation throughout the room. For the purpose of this project, we assumed that if the device is remaining in a fixed room, then the room size will be predetermined in the system code. As for the other factors, the data from the remaining sensors can backup any discrepancies in sensor readings to approximate the occupancy in the room.

The relationship between noise level and occupancy was difficult to find. Research has been done regarding this topic, which concluded that noise level would be high when occupancy is high in a given room. As mentioned above, because of the lack of raw data, we weren't able to analyze any patterns with different numbers of people in a testing room. But with the system's

ability to gather data from all three sensors, we are able to find an approximate occupancy number considering if one of the sensors is giving false data due to external disturbance.

In conclusion, the system successfully collected data from the outside environment and displayed it on the prototype dashboard. Although we weren't able to test if the data matches the occupancy value in different environments, the second phase implementation of the project will be able to do so.

Conclusion

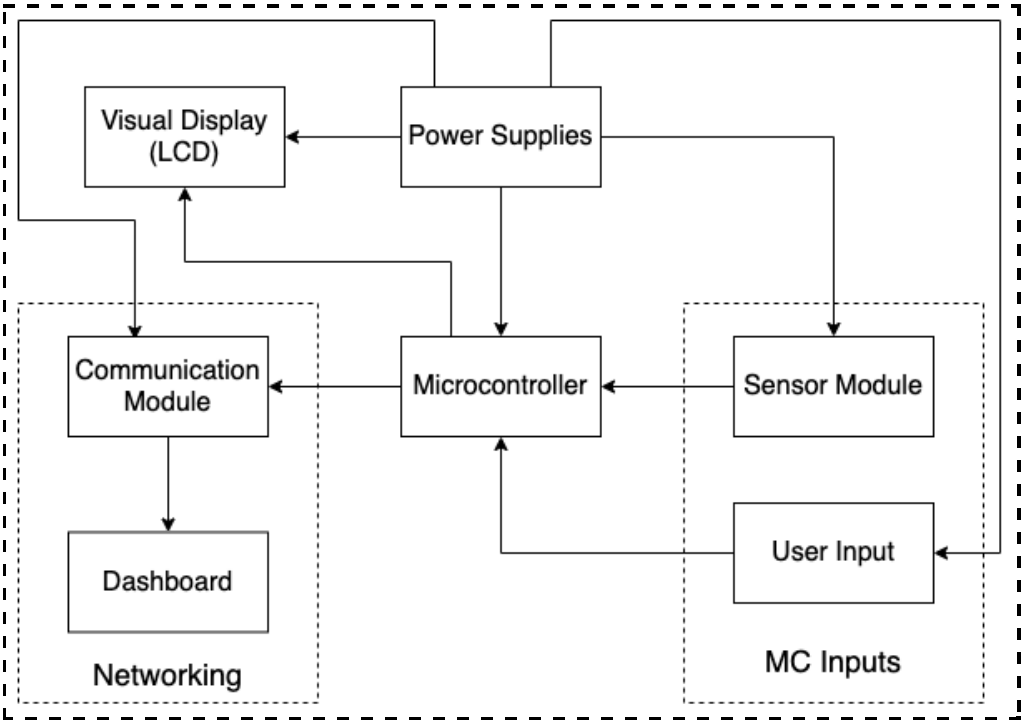
During the course of designing the system, we were able to implement the engineering design process which we realised was of importance and crucial to creating a well-designed project. Throughout the process of completing this project, we built on many of our technical, analytical and management skills. Aforementioned, the Raspberry Pi was required to be coded in Python to perform the functions it did for the project. This skill was learnt throughout the project using school provided and external resources. The project also had a mandatory PCB component that we learnt to design using the Eagle software and advised by Dave Duguid. The project also developed our skills of project management and incremental testing. This proved vital to us when the FTP process of the communication module wasn't working accordingly, so we used the code that we documented earlier in process and resolved the error. We also learnt the importance of proper time management which led to a well-designed project given that tasks were done online due to COVID-19 restrictions. We also acquired the skill of system parts selection and ordering.

However, there were some parts of the project we could have designed differently to avoid setbacks during the design process. A more detailed research on sensor functionality/complexity as well as time period involved in 3D printing of the enclosure to best house our system should have been implemented. In the case of the power module, designing our own power system would have been implemented to further improve its efficiency. Due to the process and time involved in the building of a power system (e.g CSA approval), we were not able to achieve this.

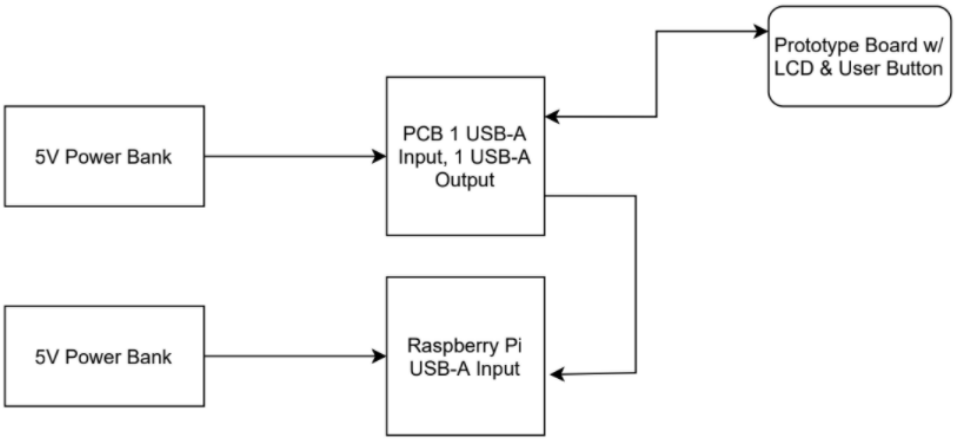
For future versions of our system we would train a dataset so we could accurately calculate for occupancy. For this to happen we would need an area where people can gather safely. The system would also alert Facilities Management as to when sensor readings like carbon dioxide levels approach a dangerously high threshold or when a room gets too hot. It could also alert FM when a room or building is over capacity. With access to the universities energy dashboard we could try to correlate occupancy to energy consumption. Creating more of these devices to monitor multiple hallways and rooms in a building is a future goal for phase 2 of this project as well. It would allow us to monitor the occupancy of an entire building. Once we know the occupancy in a building, FM would be able to trace the amount of power consumption in a building like heat and lighting based on which parts of a building are rarely used and which parts of a building need it. This would be of use to the university in energy conservation. However, all these ideas require people and would need to be worked on after it is safe to gather.

B: Schematic/Physical Construction Diagrams

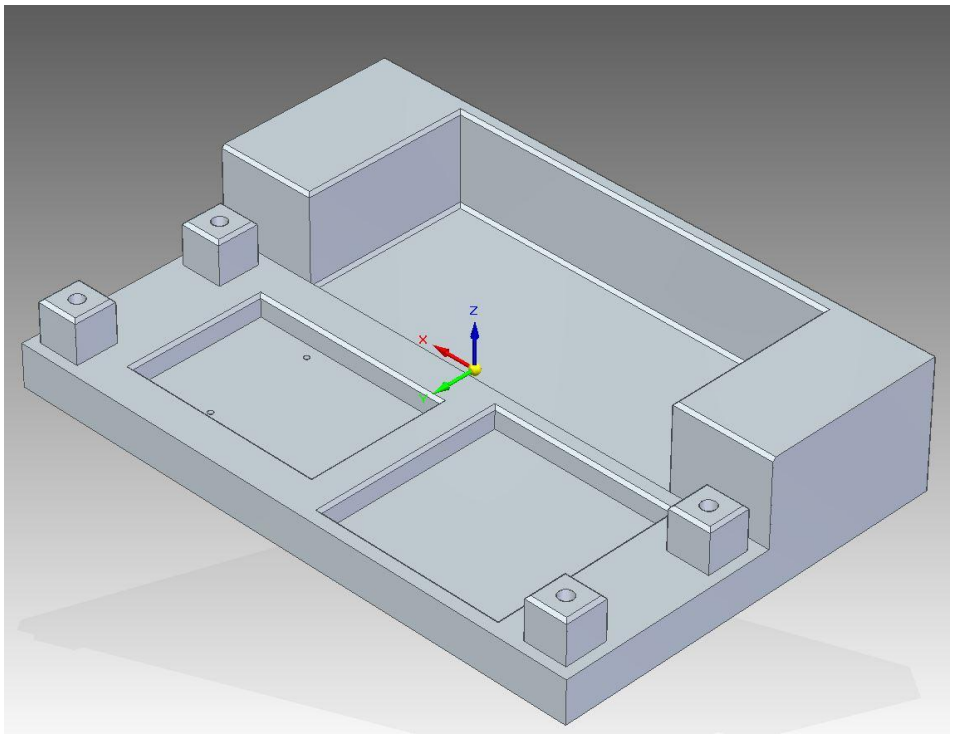
System Block Diagram

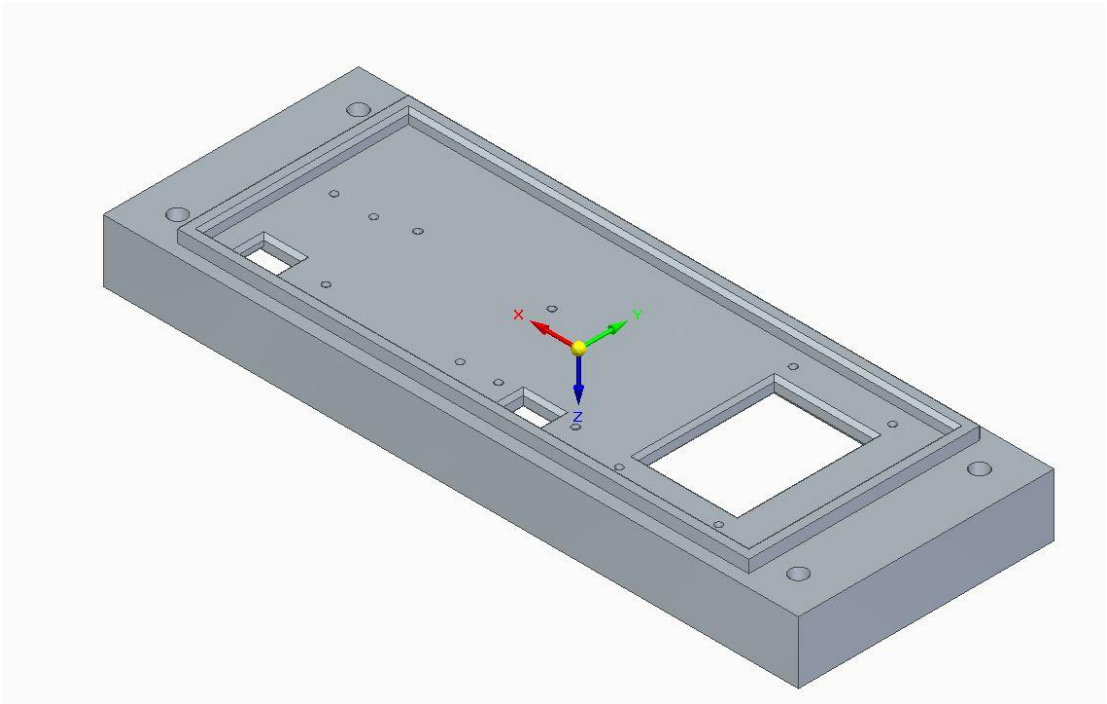


Power Subsystem Block Diagram

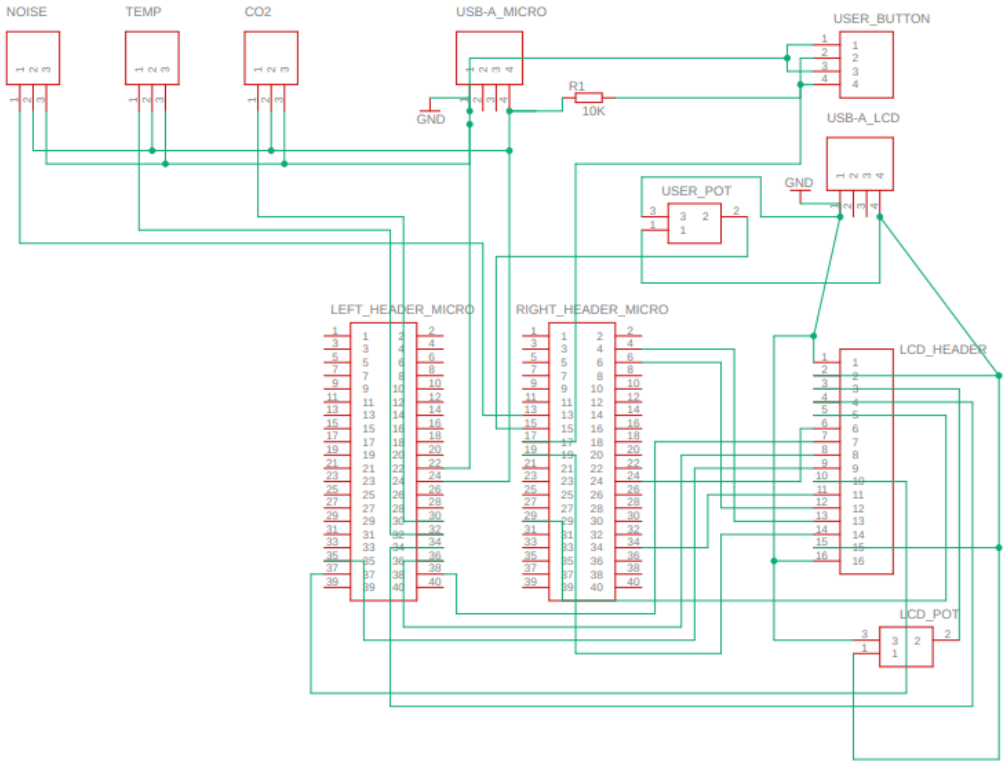


Enclosure 3D Model

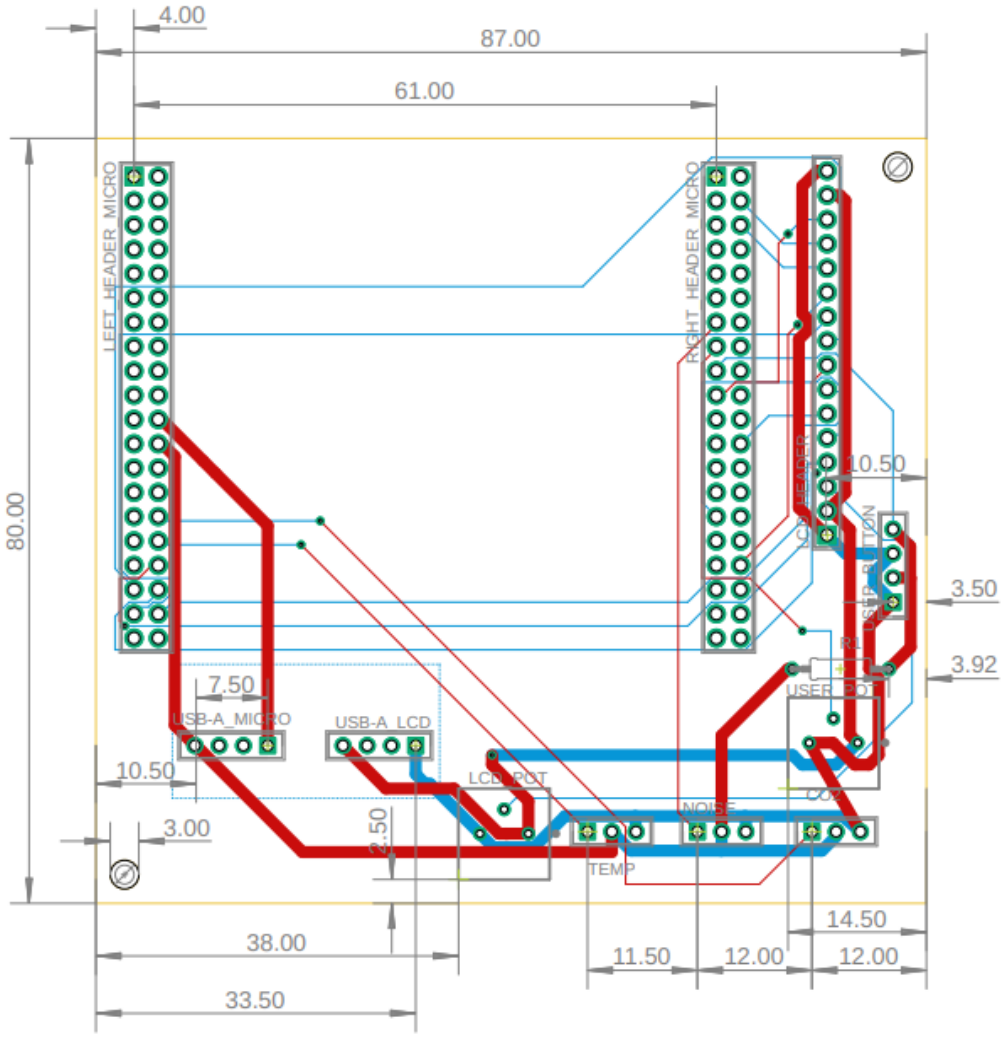




PCB Connection Diagram

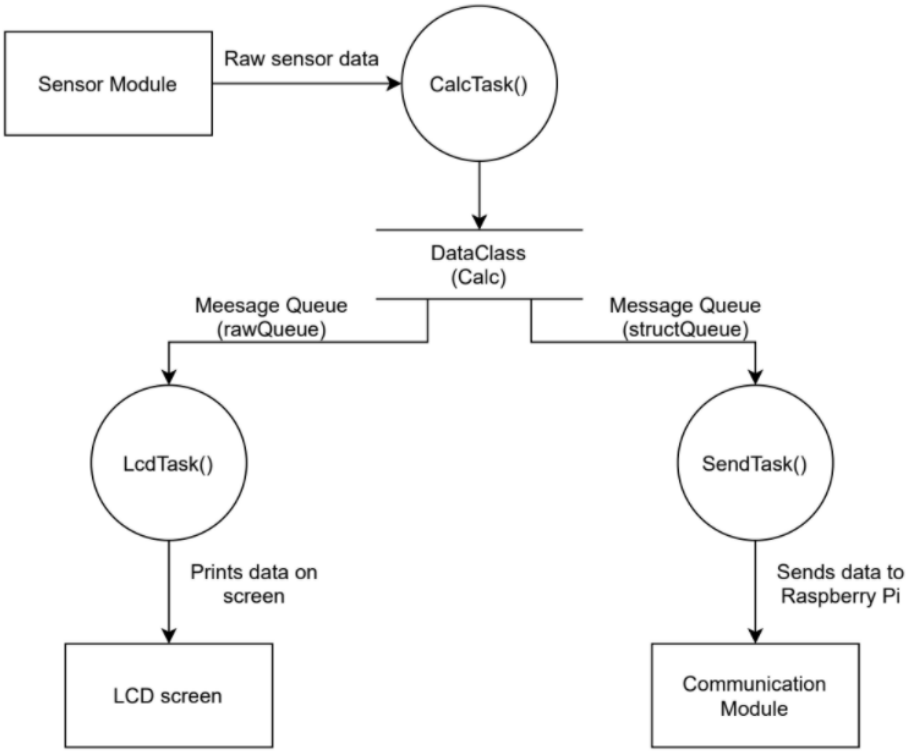


PCB Circuit Diagram

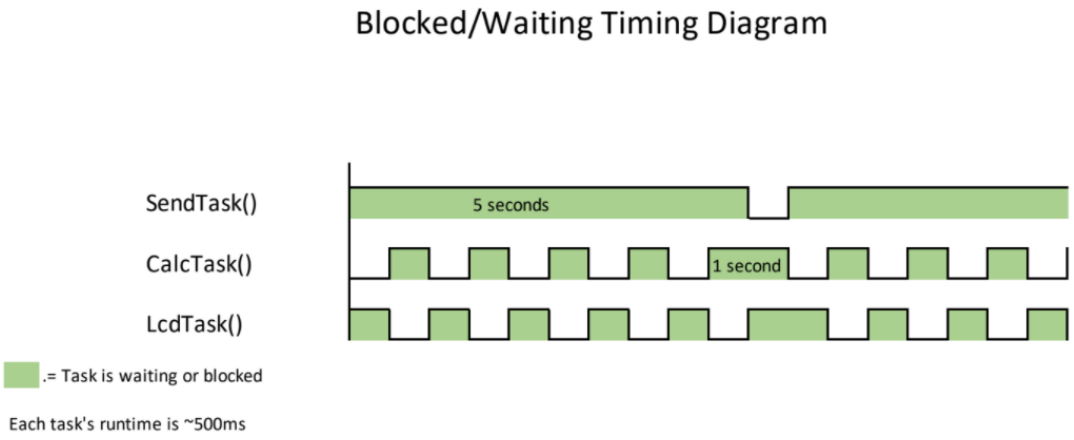


C: Software Diagrams/design structure & Listings

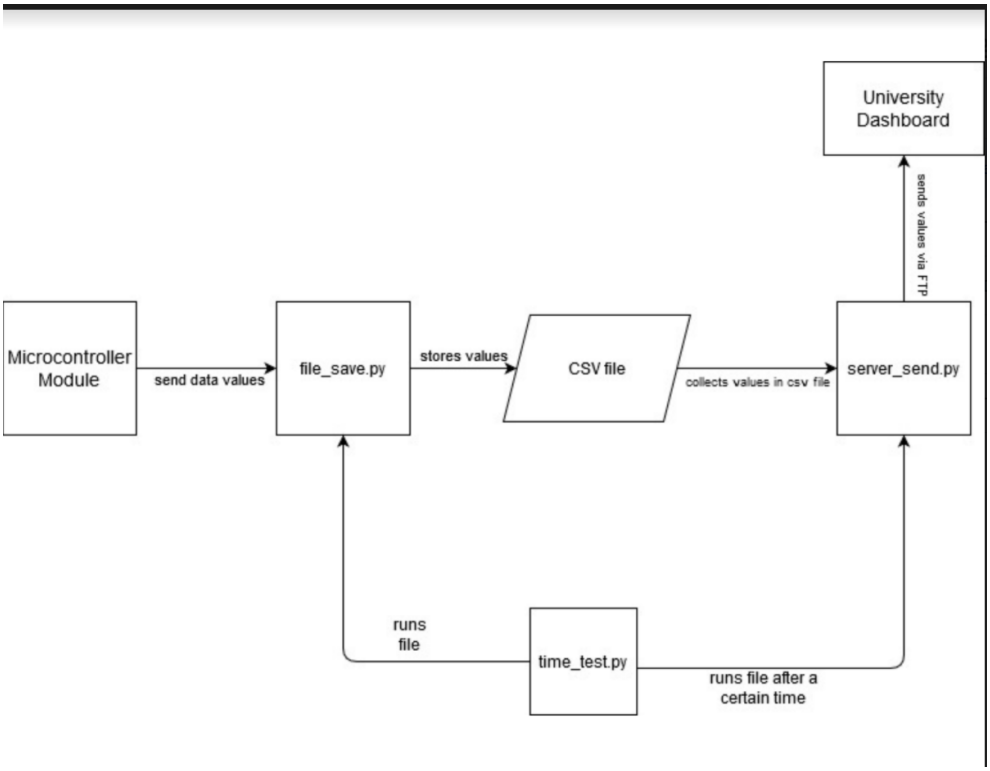
Data Flow Diagram



System Timing Diagram



Communication Subsystem Block Diagram



Project Software Listings

Project Code: https://github.com/dkatakyi/CapstoneRepo.git
Remote path to Microcontroller codes : CapstoneRepo/MicrocontrollerCode/Core/Src/main.c
Remote path to Microcontroller codes : CapstoneRepo/MicrocontrollerCode/OccupancyDetector.ioc
Remote path to User codes : CapstoneRepo/MicrocontrollerCode/Core/Src/usr/.*
Remote path to Communication module codes: CapstoneRepo/RaspberrypiCode/.*

D: References

- Sun, Y., Wang, Z., Zhang, Y., & Sundell, J. (2011, November). In China, Students in Crowded Dormitories with a Low Ventilation Rate Have More Common Colds: Evidence for Airborne Transmission. Retrieved February/March, 2021, from https://www.researchgate.net/
- Assimakopoulos, M. (2017, July). On the comparison of occupancy in relation to energy consumption and indoor environmental quality: A case study. Retrieved March/April, 2021, from www.sciencedirectassets.com
- Welcome to pyserial's documentation¶. (n.d.). Retrieved April 12, 2021, from https://pyserial.readthedocs.io/en/latest/
- Bennett, J. (2019, March 31). Screen sharing a Raspberry Pi from a Mac. Retrieved April 12, 2021, from https://www.jimbobbennett.io/screen-sharing-a-raspberry-pi-from-a-mac/
- Monitor connection. (n.d.). Retrieved April 12, 2021, from https://www.raspberrypi.org/documentation/setup/monitor-connection.md
- Raspberry Pi. (n.d.). Raspberry Pi desktop for PC and Mac. Retrieved April 12, 2021, from https://www.raspberrypi.org/software/raspberry-pi-desktop/
- Raspberry Pi UART communication using Python and C: Raspberry Pi. (n.d.). Retrieved April 12, 2021, from https://www.electronicwings.com/raspberry-pi/raspberry-pi-uart-communication-using-python-and-c
- Pysftp. (n.d.). Retrieved April 12, 2021, from https://pypi.org/project/pysftp/
- Carbon dioxide. (2021, January 29). Retrieved April 12, 2021, from https://www.dhs.wisconsin.gov/chemical/carbondioxide.htm#:~:text=250%20-%20400%20ppm%3A%20background%20
- STM32CubeIDE, libraries: stm32f1xx.h, cmsis_os.h, stdbool.h, stm32f1xx_hal.h

E: Appendices

Sensor Selection Table

Sensor	Input Voltage	Pre-heating Time/delay	Output Voltage	Conversion	Accuracy	Pass Test (Y/N)
SEN0159(CO2)	5V	0.5 hours to 48 hours	N/A	N/A	N/A	N
DFR0067 (Temp)	5V	Longer delay in code	N/A	N/A	N/A	N
SEN0219(CO2)	5V	3 minutes - 6 minutes	0.6V	V*50000/16 = 625 ppm	Yet to be finalized	Y
LM35(Temp)	5V	N/A	0.231V	V/0.01=23.1°C	Yet to be finalized	Y
SEN0232 (Noise)	5V	N/A	0. 6V	V*50 = 30 dBa	Yet to be finalized	Y

Sensor Calibration

See attached:
"arpan_backyard_data.pdf"
"battery_test.pdf"

Distribution of Work

See attached: *"GanttChart.pdf"*

Datasheets

See attached:
"CO2_Datasheet.pdf"
"Noise_Datasheet.pdf"
"PB_Datasheet.pdf"
"Pi Spec Sheet.pdf"
"Temp_Datasheet.pdf"

Project Links

System Demonstration:
https://drive.google.com/file/d/1TUD0zL3t_-CGni0Mn45NEuY90ytRfxuK/view?usp=sharing