EMBEDDED SYSTEMS ANALYZING THE ENVIRONMENT

Bjorn Melin
Computer Science
Augsburg University
Minneapolis, MN 55454
melinb@augsburg.edu

Joseph Hentges
Computer Science
Augsburg University
Minneapolis, MN 55454
hentgesj@augsburg.edu

April 1, 2020

ABSTRACT

This project demonstrates the use of embedded systems integrated into environmental studies and building controls. In a southwest-facing glass-enclosed stairwell in the Hagfors Center at Augsburg University, embedded systems are used to observe whether or not outdoor environmental parameters such as light intensity and temperature create a greenhouse effect in this more than 50-foot column of enclosed air. This project implements a network of Raspberry Pi Zeros each monitoring and recording temperature and ambient light intensity sensors. This network relays the data to a separate computer where it is fed into custom software, analyzed to see whether or not there is a greenhouse effect occurring, and will be used to predict future temperatures North-West staircase of Hagfors Center. This study provides a consequential implementation of embedded systems which shows how they can be of great use when studying the environment.

1 Introduction

Embedded systems have become an imperative component of the present day field of computing. From pencil sharpeners to electric vehicles, embedded systems are at the center of modern infrastructure. The internet of things (IOT) and embedded systems are arguably the most important developments of this century. S. Mallon has noted, "IoT provides a platform that creates opportunities for people to connect these devices and control them with big data technology, which in return will promote efficiency in performance, economic benefits and minimize the need for human involvement. It's the most important development of the 21st century" [6]. Use cases for IoT and embedded systems are seemingly endless. Allowing for the automation of basic tasks, communicating and sharing information in ways never imaginable in previous decades, and real-time complex data analysis.

With different disciplines finding use of IOT, it has become a highly researched area, and it is not a surprise that interest continues to grow in the increased technological advancements of automobiles, and embedded aerial navigation in commercial airlines. However, largely ignored by the general public, IOT is having a significant impact on environmental monitoring. Research out of the KS Institute of Technology, in Bangalore, India, outlines where and how embedded systems can be used to monitor and manage the different aspects of agriculture [8]. The paper, Intelligent Monitoring Device for Agricultural Greenhouse Using IOT, lays out a system for relaying vital crop information through the cloud to a monitoring system. It explores the benefits of implementing embedded environmental monitoring in commercial farming, as well as where it would have the greatest impact. However, what this paper is missing, is the actual implementation of such a system.

In recent years an expansion of embedded systems in commercial goods has shifted the focus of IoT research. Ultimately, this has pulled away interest in how these types of systems can be used to analyze the world around us. Incorporating embedded systems with IoT offers a wide range of use cases, in which many could be used to analyze and improve the environment. Our initial informational research into IoT and embedded systems revealed a gap in the market. This further expanded by a few basic projects immediately showed us how and where we should focus our research project. Prior work in the field uncovered where embedded technology has low utilization, and through exploring environmental use cases, we thought of ways in which we could analyze our local environment to provide actionable insight using IoT. We concluded that this study would attempt to answer a primary question. That question being, how can embedded systems be integrated into environmental studies and building controls to monitor the greenhouse effect in an indoor environment?

To answer this question, this study dealt with constructing a network of embedded systems to monitor temperature and ambient light levels in a southwest-facing glass-enclosed stairwell in the Hagfors Center at Augsburg University. The network of embedded systems implemented in this study were constructed to observe whether or not outdoor parameters such as light intensity and temperature create a greenhouse effect in this more than 50-foot column of enclosed air. The purpose of this study is not only to see how embedded systems can be integrated into environmental studies and building controls to monitor the greenhouse effect, but to also provide insight to Augsburg University as to how they can save money on HVAC by redistributing excess heat accumulated in the stairwell throughout the building, thereby cutting down costs on heating and cooling by the University.

In order to determine whether the greenhouse effect causes frequent temperature fluctuations in the stairwell, a network of Raspberry Pi Zeros are implemented. With each Pi in the network monitoring and recording temperature and ambient light intensity sensors. The network relays the data to a separate computer where it is fed into custom software, analyzed to see whether or not there is a greenhouse effect occurring, and will be used to predict future temperatures North-West staircase of Hagfors Center. Here, the study provides a consequential implementation of embedded systems which shows how they can be of great use when studying the environment.

To implement each Raspberry Pi Zero in our network of embedded systems, an operating system was installed, sensors were attached and configured, software was written, and functionality testing was done on each device. This project includes custom written software which pulls the value off of the corresponding pins for each sensor and writes the data to a plain text file format. Using crontab scheduling, the custom software is run once every ten minutes. Once the data has been written to a plain text file on the Pi, it is transferred over Wi-Fi to a server computer which produces plots of the data twice a day. The resulting plots are then stored for further analysis.

Upon completion of this project, a network of Raspberry Pi Zero's was successfully constructed, programmed, tuned, and implemented in the testing environment. Additionally, all systems in the network properly collected and handled the data from each sensor accordingly. This study provides an implementation of embedded systems which shows how they can be of great use when studying the environment. In addition, this study will provide Augsburg University with clear and actionable results as to how HVAC can better be handled in Hagfors Center in a manner which minimizes cost and maximizes efficiency.

2 Methods

The methods of implementation for this study included six core stages: determining project workflow method, identifying necessary resources and gathering materials, building a network of embedded systems, custom software development, testing functionality and refactoring, and analysis of data collected by the system.

2.1 Project Workflow

To begin this study, a project workflow method was determined. For this study, flexibility and communication between team members were of great importance. When determining the method of workflow, we considered several factors which could affect the study. Some main concerns for the workflow of this study included awaiting the arrival of parts and materials, allotting enough time for background research and testing demonstrations of unfamiliar components, ensuring researchers workloads were evenly distributed, minimizing the dependencies of tasks, and keeping researchers on the same page as one another. Based on the primary concerns regarding the workflow of this study, it was decided that agile methodology would be effectuated. In addition to addressing all primary concerns of this study, we thought it important to practice agile methodology given its increased use amongst companies worldwide.

2.2 Resources and Materials

Once the workflow method of this study had been fixed, the resources needed to execute the project were identified and materials were gathered. Intensive research was carried out to determine how the network of embedded systems would be built, decide what technological components would be used, establish which programming languages would best suit the needs of the project, and the various parts and hardware that would be needed to successfully implement this study. For this study, it was decided that the network of embedded systems would consist of five Raspberry Pi Zeros. The Raspberry Pi Zeros would be equipped with a temperature sensor, an ambient light sensor, set up with custom software, powered by battery, and enclosed in a protective casing. Each Pi would be configured to fit its purpose in this study by installing the Linux based Raspbian operating system via a secure digital (SD) card, writing custom software to the SD card, and setting up each Pi so that the custom software begins execution on boot of the system. In this situation, the custom software developed for each Pi was written using the Python interpreted programming language with aid by the Linux command line.

Once we had solidified how to build the network of embedded systems, what technology to use, and the programming languages which would be employed for this study, we then had to identify and obtain each material or piece of required hardware. The materials used to build the final network of systems composed of five of each Raspberry Pi Zeros, photoresistors to measure ambient light levels, thermistors to measure temperature, PCF8591 Analog-to-Digital Digital-to-Analog converter modules to represent analog signals of sensors digitally, 40-pin headers, Perma Proto Bonnets to solder each systems components onto, DC battery pack holders, DC to Micro USB B adaptors, 8 GB SD cards, AA batteries to power each system, and a pack of 10k ohm resistors. Although those materials compose each of the completed systems in the network, additional materials were required for building and testing the first system. To build the first Pi in the network, in addition to the parts listed above we made use of general jumper wires, a breadboard, a GPIO expansion board and wire, a mini HDMI to HDMI converter cable, a USB-A to Micro USB-B converter cable, a desktop monitor, a keyboard, and a mouse. All materials used in this study were purchased online from *Amazon* and *Adafruit*.

2.3 Building the System

Next, we moved onto the third stage of the study in which the network of embedded systems was constructed. Once all materials had been gathered, we set out to build the first system in the network of Raspberry Pi Zeros. First, the Raspbian operating system was downloaded from the Raspberry Pi website [1]. For the initial system in the network, *Raspbian Buster with desktop and recommended software* was downloaded so that software could be written and tested on the Raspberry Pi itself. The remaining four systems in the network were equipped with *Raspbian Buster Lite* and accessed through USB connection on a separate machine. Once *Raspbian Buster with desktop and recommended software* had been downloaded to a local Macintosh machine, an SD card was inserted into the machine's SD port. The Raspbian image was burned into the SD card [5], ssh was enabled [4], and network information was added to the card. Then, once configured, the SD card was inserted into the Raspberry Pi Zero, the Pi was plugged into a desktop monitor, and started up.

After the Pi had been configured with *Raspbian Buster with desktop and recommended software*, the systems sensors were implemented. Initially, a 40-pin header was soldered onto the Pi so that various electrical components could be attached. The GPIO expansion board and wire were plugged into the header pins on the board and the expansion board was attached to a breadboard. Now that the Pi had been attached to a breadboard and code could be written and tested on a desktop monitor, we equip the Pi with temperature and ambient light measuring sensors. To measure temperature and ambient light levels, a generic thermistor and photoresistor were used, respectively.

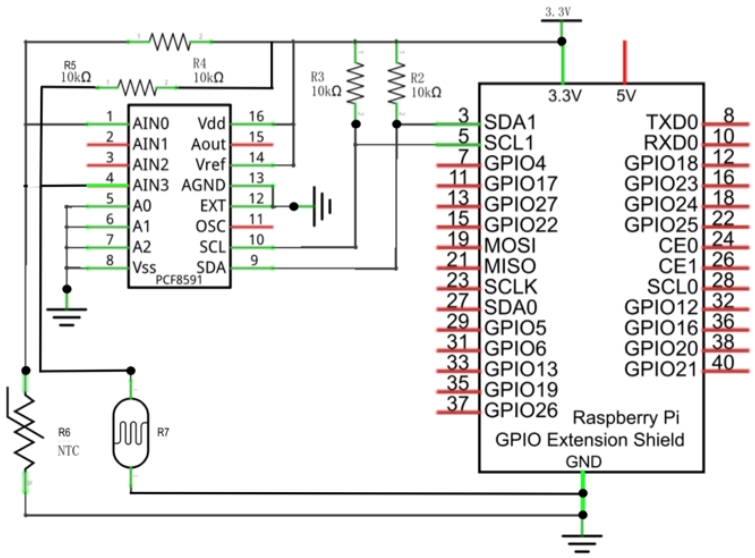

Figure 1: Circuit Schematic Diagram

In order to achieve a circuit which would allow for a thermistor and photoresistor to be attached onto each Pi so that their corresponding data could be accurately gathered from their surrounding environment, a schematic diagram was created shown above by Figure 1. Immediately following the completion of the circuit's schematic diagram, the circuit was implemented on the breadboard attached to the initial Pi in the network. On the breadboard, a PCF8591 Analog-to-Digital Digital-to-Analog converter module was attached in the center of the board. The SCL and SDA pins of the PCF8591 were attached to the SCL1 and SDA1 pins of the Raspberry Pi via the GPIO extension board. Resistors were also attached to the SCL and SDA pins of the PCF8591 and connected to the 3.3V power pin of the Pi to supply power to the sensors. Next, the EXT and AGND pins of the PCF8591 were attached to the GND pin of the Pi, and the VREF and vDD pins of PCF8591 to the 3.3V power pin of the Pi. Then, on the other side of the PCF8591, the A0, A1, A2, and Vss pins were all grounded to the GND pin of the Pi. Finally, for each of the two sensors, one end of the sensor was connected to the 3.3V power pin of the Pi and the other end was grounded. The ends of the photoresistor and thermistor wired to the power supply were then connected to the AIN0 and AIN2 analog-in pins of the PCF8591 using jumper wires, respectively. This initial breadboard version of the system was used for developing and testing the custom software written for this study.

Lastly, once the custom software had been written, tested, and finalized, the final version of the first Pi in the network was constructed. To build the final version of the initial Pi, the circuit and all of its components were transposed from the breadboard to a Perma Proto Bonnet. First, a 40-pin female header was soldered onto the bonnet. Following Figure 1, the circuit was then soldered together and implemented on the bonnet. Once the circuit had been built, the Perma Proto Bonnet was attached to the 40-pin male header on the Pi. At that point, the circuit of the initial system had been translated from the breadboard onto the bonnet, compartmentalizing the system into a more portable unit.

The completed Pi was then attached to a battery pack which provided an external power supply to the system and enclosed in a hardened plastic casing. After that, the completed system was tested once more to verify its functionality and upon successful testing, the remaining four Pi's in the network were built to match the initial system.

2.4 Custom Software Development

There were two primary languages we looked into for developing the software for this project, Python and C. The languages both offer a variety of open libraries for dealing with the data generated by the sensors. Therefore, the languages internal strengths and weaknesses became the deciding factors on which language we chose to use. One of the greatest strengths of the C language is its speed; so in many instances this would be the key factor language choice. However, the ease of use of Python, and its readability gives it a far more expandable and easier to write code base. Also, with Python exploding in popularity, the number of open source libraries makes the language far more useful in most cases. The two languages have their key advantages and disadvantages, so we thought it was important to explore both in the development of this software.

During the initial build of the first Raspberry Pi in the network, custom software was written following example code provided by *Freenove* in *Tutorials* [2]. *Freenove* offers both C and Python examples of the code, so we initially wrote two versions using both languages. This initial code was a far more stripped back version of the final version, but it gave an idea of how the code base would be designed. The C version of the code was far lengthier than the Python code, and required initial setup for processing the data. On the other side, the initial setup for Python was done within the *RPI.GPIO* and *smbus* libraries we used. Early on we noticed that the setup piece of the code was one of the hardest parts to follow in the C version, and the easiest to follow in that of Python. Viewed as either a negative or positive, Python abstracted much of the hard to follow setup of the data collection process. The complexities presented by the C languages setup in terms of abstraction were a key determinant in the decision to use Python as the language for our software.

The data collection and analysis portion of this project stemmed from our software design, and how we are storing and managing the data generated by devices. A few key pieces of our data included: the timestamp, sensor voltage reading, and calculation of actual reading. Therefore, as a result of having two sensors, the total data collected includes:

- Timestamp
- Light sensor voltage
- Light sensor calculated value
- Temperature sensor voltage
- Temperature sensor calculated

The two voltage readings, light and temperature, come in as analog signals, and through the use of the *smbus* library we were able to convert them to a digital value. These voltages then needed to be converted to the actual values they represent, ambient light value (0 to 255), and degree in Kelvin.

To calculate the ambient light value, we had to divide the analog value we read in by the voltage of the pin multiplied by 255. We chose 255 because it is the maximum value of the RGB color pallet. This calculated value then rises and lowers with the amount of light read by the sensor. We calculated the reading to rise with less light and lower with more light, so it can be visualized as the color white darkening as the value rises.

In a similar fashion we calculated the voltage of the temperature sensor using the same formula as the photoresistor. From this value, we then needed to calculate the resistance value of the thermistor. Following the code provided by *Freenove* we were then able to then use the resistance value to calculate the temperature in Kelvin, using the formula below. From the temperature in Kelvin, the Celsius and Fahrenheit temperatures were a straightforward calculation.

$$tempK = \frac{1}{(\frac{1}{273.15+25}) + math.log(\frac{Rt}{10}) * 3950} \tag{1}$$

To access the data collected, we decided printing out to two local text files would suffice for the initial proof of design for the software. So as a part of the data collection process, after completing the data calculations, we printed the timestamp, voltage for both sensors, as well as the calculated light value for the ambient light sensor, and the calculated Fahrenheit temperature for the temperature sensor. In order to maintain previous data and, we appended each new set of values to the files.

Finally, in order to preserve battery power, and limit the amount of resources being used by the script, we reformatted the script to run as a cron job. To reformat, we removed the infinite loop responsible for continuously reading, calculating and printing the data, and replaced it with a basic function that would run each time the script was called. This format allowed the script to be added to cron as a task we could run at a specified frequency. Due to light being an important component of our analysis, we decided to prioritize the light reading in the decision of the frequency of the job. Therefore, to maintain constant read of the light, we ran the cron job every minute.

2.5 Testing

With our custom software initially working off of our observations and outside resources, the testing portion of the project is arguably the most important. We had to conduct multiple sessions for testing the accuracy of the sensors, as well as the calculations on sensor data made in the code. Sensor accuracy is the key component of this project, so, determining which sensors are the most accurate, and how accurate they are was a large portion of the testing process.

Our first design of the system used a DHT-11 Temperature and Humidity sensor, which is a two-in-one sensor system that allowed us to simplify the overall Raspberry Pi build design. The key benefits to using this sensor lied in its low cost, and limited resource use of the build, however, its drawbacks include a small range of humidity and temperature readings, two second read delay, and accuracy concerns. Initial testing using the sensor solidified these concerns as it showed to rarely give a non-null reading, and the sensor delay also hindered the design of the software.

As a result of the lack-luster results of the DHT-11 sensor, we dropped keeping track of humidity, and shifted to using the thermistor temperature, and photoresistor sensors. The hope with these was that, since they read only one value it would give them improved accuracy, and improve the data collection of the overall system. After a few overnight runs of the system, the data we collected established that the new design was the superior of the two. We no longer received null data from the sensors, and through outside device testing along with code calculation tweaks, we determined that they maintained a strong accuracy across the board.

After determining the Raspberry Pi systems were accurate, the layout of the network became the next most important piece of the data collection process. We were not convinced the individual systems could make a general reading of the room as a whole, so the placement of each device, and a combination of each device's data readings would be the best indicator of how the room reacted to the outside environment. The key places we wanted to get readings of were beside the windows, and along the far inside wall of each floor. However, since we were limited in the number of devices we built, determining the best placements required testing. Due to limited overnight access to the stairwell, we concluded that the Linux Lab in Hagfors Center at Augsburg University would be the next best location to test our systems. The Linux Lab is located on the third floor of the building and has windows along the entire outer wall of the room. Thus, providing an almost identical testing environment as the south-west stairwell in Hagfors Center.

Initially, we had a single device to test on, so over the course of the week, we moved the device to different locations in the room. This was not for data collection, but rather to see which locations made for the best data, which was determined by how it responded to the time of the day, and expected values. From the location testing process, we concluded that alongside the windows made for the best data. Once the best device locations were determined, we started the data collection process.

2.6 Data Analysis

Finally, we reached the final stage of this study which prompted for the orchestration of thorough analysis of the data generated by the network. The purpose of the data collected by each Pi in the network of embedded systems was to be used to predict future temperatures and determine whether or not a greenhouse effect is transpiring in the North-West staircase of Hagfors Center. To draw these conclusions from the data collected by the Pi's, the text files containing the temperature and ambient light level readings for each Pi were transferred to a separate server which was used to perform daily analysis on the data.

On a separate server, software was written in the R programming language which allowed us to visualize temperature and ambient light levels for each Pi in the network over time. To implement this analytical software, first the text files for each of the Pi's in the network were read in by the software converted into data frames. For each of the five Pi's in the network, two data frames were created to store the temperature data and ambient light level data, respectively. In the current iteration of this study, separate data frames were created for each day that data was collected as well. The decision to separate the data gathered by each Pi's sensors into separate data frames by day was made with the purpose of generating plots of the data which would clearly show trends in the data and would be easily interpretable. In later iterations of this project where there is a larger amount of data to use for analysis, this analytical software will be revised so that data for each sensor, for all days, is stored and accessed through a single data frame for each of the five sensors.

Once the raw text files had been read into R and converted into data frames, the data frames were tidied. First, to begin tidying the data frames, all entries in the data collected before the Pi's were properly positioned in their final testing locations were removed from each data frame. To ensure that all data collected before the Pi's were in position was removed, the time in which each Pi was placed in its proper testing location was recorded during testing. Using the times recorded, all data for each system which was gathered before the system had been set in its proper testing environment was removed. Second, the entries in the tables were summarized into 15 minute intervals by computing the average of the temperature and ambient light levels in each 15 minute interval. Third, all names of the variables in the data were changed for ease of access in R. For ambient light data, timestamps of entries were split into two distinct variables called *Date* and *Time*, voltage readings were stored as *Voltage*, and analog to digital converter (ADC) levels were stored as *ADC_Light_Value*. Similarly, for temperature data, timestamps of entries were split into *Date* and *Time*, voltage was stored as *Voltage*, and temperature levels were stored as *Temp_Fahr* since temperatures were collected in degrees Fahrenheit.

Succeeding the tidying of all data frames, a function was implemented which allowed for custom x-axis labels to be applied to each plot generated. As basic as this sounds, this was actually an important step in constructing the analytical plots for this study. This being as setting custom breaks in the times printed on the x-axis of the plots greatly increased readability and clarity in the plots. In short, this function added a new field to each data frame named xlab which contained the value of the Time variable of every fifth entry in the data frame, leaving all other entries in the column empty.

Finally, plots displaying the data gathered by each sensor per day were generated. Plots were constructed using *ggplot2* from the *tidyverse* package to visualize the data. Each plot used a unique temperature or ambient light level data frame as described in the previous paragraphs with the Time variable measured on the x-axis and the *Temp_Fahr* or *ADC_Light_Value* measured on the y-axis. All plots were faceted using the *Date* variable, x-axis labels were assigned using the *xlab* variable from each corresponding data frame, and various themes were applied to each plot. Ensuing plot generation for all data collected in this iteration of the study, results were drawn through rigorous analysis.

3 Results

From initial inspection of our data we noticed large fluctuations on the temperature reading. At the time, these temperature changes did seem to line up with the night/day cycle, however very few conclusions could be made. The graphs below, show these fluctuations.

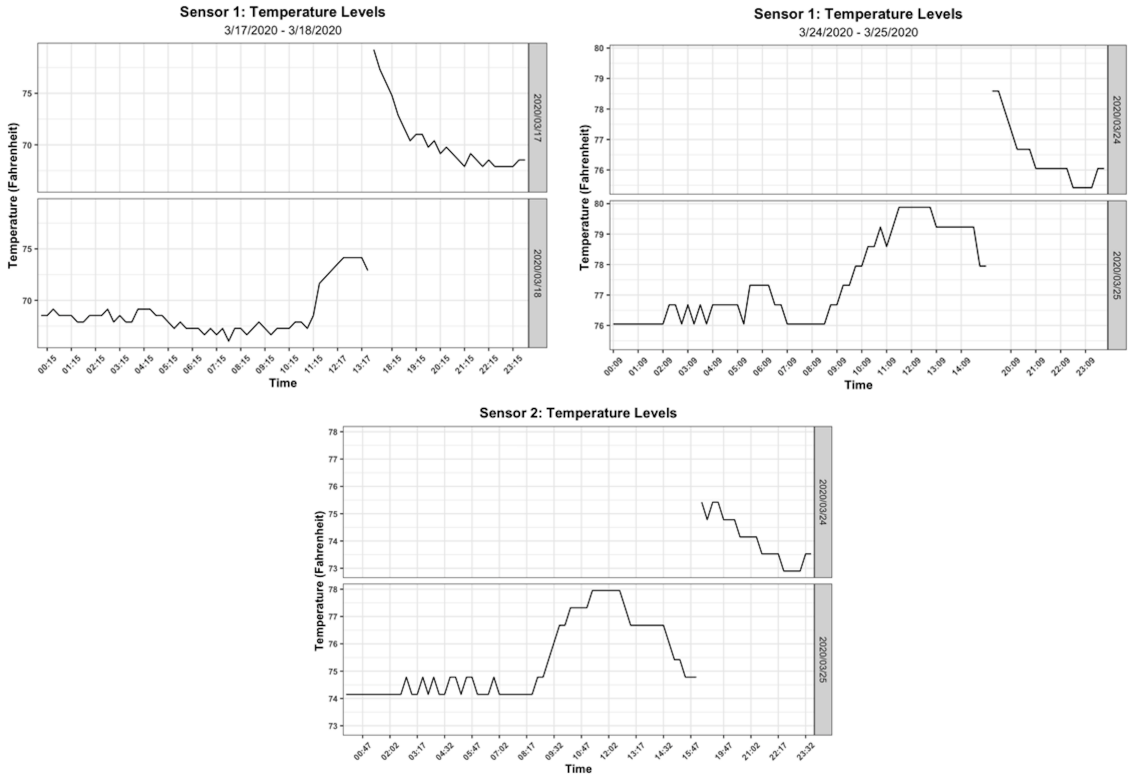

Figure 2: Temperature Levels

The graphs display two days each, and the lines connect along the top-right and bottom-left lines. Following the timestamps labeled along the x-axis, the temperature change is clear; peaking at around noon each day, then petering off to around 70 degrees Fahrenheit.

To show the Hagfors Center at Augsburg University suffered from the greenhouse effect, we needed to connect the temperature changes to light. Therefore, we focused data collection on light levels throughout the day. Displayed on the following page are three ambient light level graphs.

The ambient light readings also show a very similar progression to the temperature reading. Far more drastic in its change, the light readings drop off completely at night, but most importantly, the times in which the light begins to increase, the temperature begins to also rise.

Displayed side by side, these two figures show a clear connection between the two values. Both the temperature and light readings match in their progression between the two value spectrums. Based on the results we've gathered thus far, we cannot be confident in providing solid conclusions. However, from the set of data gathered by our systems so far, we do see strong evidence of a greenhouse effect occurring in the south-west facing Linux Lab in Hagfors Center at Augsburg University.

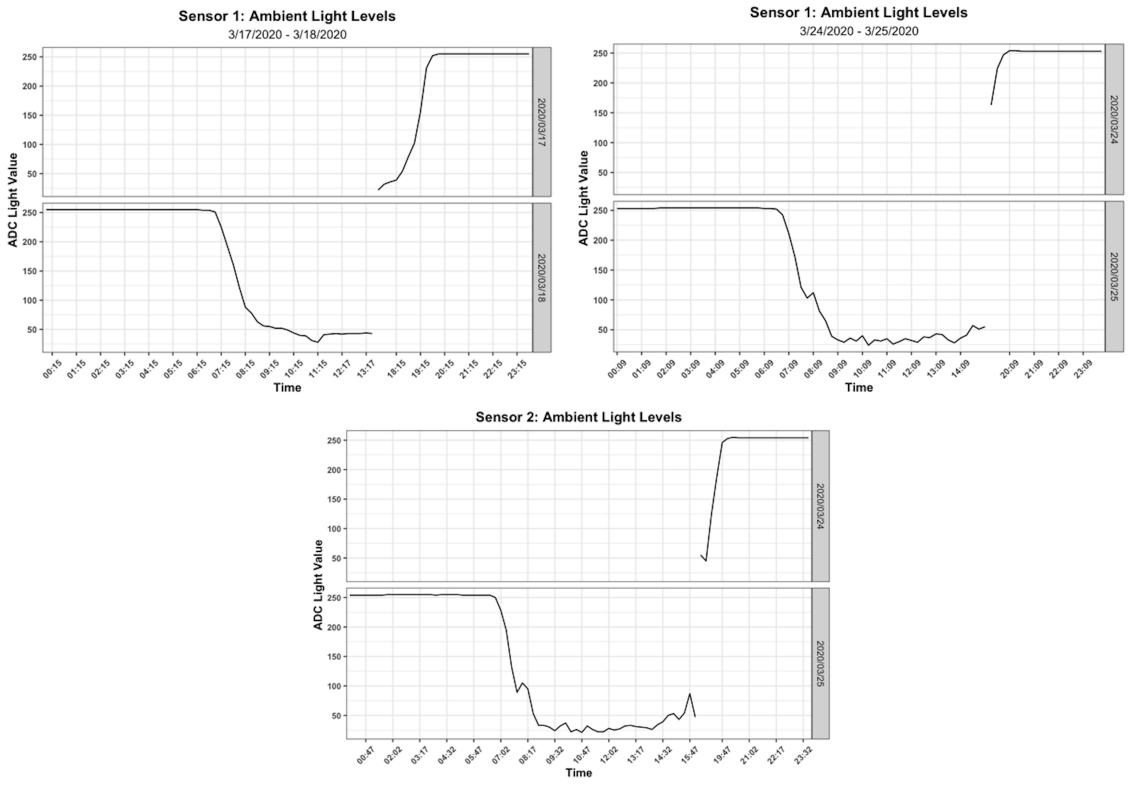

Figure 3: Ambient Light Levels

4 Discussion

Due to the Covid-19 outbreak in Minnesota, complications arose which prohibited our ability to execute this study to its full potential given the timeframe. Amongst other complications caused by Covid-19, our primary limitations from this outbreak included the shutdown of the Augsburg University campus and facilities, trade bans and border closures, and a government mandated stay at home order. These limitations led to us not having necessary hardware components for our systems delivered, strenuous efforts needing to be taken with respect to communication and collaborating in person on the project, the inability to gather desired data from the southwest-facing glass-enclosed stairwell in the Hagfors Center at Augsburg University and hence, difficulties obtaining the desired results of this study.

Our research in embedded systems has answered many questions, there is still quite a bit that should be explored. Our process for building the Raspberry Pi systems as well as developing the initial software for data collection relied primarily on trial and error. Through this process we were able to develop a system that can accurately gather data, while also preserving battery power on the remote system. Additionally, based on our testing of both battery and wall powered systems, the battery powered system did not have a noticeable difference in the accuracy of the data collected, which shows these types of computer systems can run off batteries without much worry.

Based on our analysis of the data we gathered, our original hypothesis is most likely correct. Results show that in almost identical testing conditions as in the stairwell in Hagfors Center, the greenhouse effect is occurring. Therefore we can conclude, without certainty, that the outside weather and environment causes a greenhouse effect inside the south-west facing stairwell of Hagfors Center at Augsburg University. Most importantly, our analysis of the data collected by the Raspberry Pi systems show that they can accurately gather temperature and ambient light environmental data.

5 Future Work

In the near future, we plan to finish this iteration of the study by building the remaining systems in the network of Pi's and collecting data from the southwest-facing stairwell in Hagfors Center at Augsburg University rather than the Linux Lab in Hagfors Center. Once these tasks have been completed, we plan to expand on the results of this study by collecting a larger set of data using the network of embedded systems and performing more in depth analysis on the data collected. Specifically, we would like to collect data on a broader variation of days. By collecting data throughout the various seasons, on cloudy and sunny days, or on rainy and snowy days would provide more useful and actionable results than the current iteration of this study.

Alongside a better variety of data, streamlining the data collection process is one of the biggest pieces we would like to improve. As is, data collection requires more than infrequent human interaction, be as battery replacement, data transfers and device monitoring. To improve these aspects of the process we plan on further developing the custom software to make use of cloud services, such as AWS and Google Cloud. Through the use of these services, we are planning on creating a data pipeline to move the data directly from each device to a remote data warehouse. Warehousing the data would allow us to easily pull data from specific devices as well as clean up the transfer process, which currently requires retrieving sim-cards from the devices. For monitoring the devices, we also plan on developing software to relay device statistics, such as CPU usage, and remaining battery power. Overall, the future work on this project is primarily focused on improving our data results as well as further making use of the data for analysis purposes.

References

[1] R. Pi, "Download Raspbian for Raspberry Pi," https://www.raspberrypi.org/downloads/raspbian/, Feb. 2020.

[2] Freenove, "Freenove RFID Starter Kit for Raspberry Pi," Sep. 2016. [Online]. Available: https://github.com/Freenove/Freenove_RFID_Starter_Kit_for_Raspberry_Pi

[3] Adafruit, "Adafruit Industries, Unique & fun DIY electronics and kits," Mar. 2020, publisher: Adafruit. [Online]. Available: https://www.adafruit.com/categories

[4] M. Allen, "Setup Pi Zero W Headless Wifi," Dec. 2019, publisher: Desert Bot. [Online]. Available: https://desertbot.io/blog/setup-pi-zero-w-headless-wifi

[5] R. Pi, "Setting up your Raspberry Pi," 2020, publisher: Raspberry Pi. [Online]. Available: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2

[6] S. Mallon, "IoT Is The Most Important Development of The 21st Century," Oct. 2018, publisher: SmartData Collective. [Online]. Available: https://www.smartdatacollective.com/iot-most-important-development-of-21st-century/

[7] J. Geerling, "Mount a Raspberry Pi SD card on a Mac (read-only) with osxfuse and ext4fuse," Jun. 2017, publisher: Jeff Geerling. [Online]. Available: https://www.jeffgeerling.com/blog/2017/mount-raspberry-pi-sd-card-on-mac-read-only-osxfuse-and-ext4fuse

[8] G. Pavithra, "Intelligent Monitoring Device for Agricultural Greenhouse Using IOT," in *Journal of Agricultural Science and Food Research*, vol. 9. J Agri Sci Food Res, an open access journal, Apr. 2018, pp. 1–4. [Online]. Available: https://www.longdom.org/open-access/intelligent-monitoring-device-for-agricultural-greenhouse-using-iot.pdf