Learning Holiday Light Project

Final Report

SDMAY21-48
Client and Advisor: Dr. Daniels
Team members and roles:
Christopher Woods (Chief Software Engineer)
Jacob Martin (Chief Computer Engineer)
Ashkirat Singh (Meeting Facilitator)
Mitchell Wadle (Meeting Scribe)
Ty Gardner (Chief Engineer (computer vision))
Joyeux Noel (Report Manager)

Team email: sdmay21-48@iastate.edu
Team website: https://sdmay21-48.sd.ece.iastate.edu/

Revised: 4/25/2021

Table of Contents

1 Introduction 3

1.1 Problem and Project Statement 3

1.2 Requirements 3

1.3 Project Design 4

1.4 Expected End Product and Deliverables 5

1.5 Evolution from E E/CPR E/S E 491 7

2 Implementation 7

2.1 Implementation Details 7

2.2 Relevant Standards 8

2.3 Engineering Constraints 8

3 Security Concerns and Countermeasures 8

3.1 Physical Security 8

3.2 Cybersecurity 9

4 Testing 9

4.1 Testing Process 9

4.2 Unit Testing 10

4.3 Interface Testing 10

4.4 Acceptance Testing 10

4.5 Results 11

5 Context of Work 12

5.1 Related Products 12

5.2 Related Literature 12

6 Appendix A: Operating Manual 14

7 Appendix B: Alternate Design Versions 16

8 Appendix C: Other Considerations 16

1 Introduction

1.1 Problem and Project Statement

One popular way to celebrate the holiday season is with light displays. Many people like to push the creative boundaries of what is possible, using different lighting schemes. However, we are looking for a way to independently control each light, allowing even further customizability. Any approach to this problem also needs to be accessible to a non-technical audience because of the widespread popularity of light displays. This problem was proposed by Dr. Daniels on behalf of his wife.

Our solution is expanding upon the previous work done by senior design teams on this project. The solution thus far utilizes a string of controllable color LEDs. These LEDs can be laid out in various ways: cone(tree), cylinder, and sheet. Once the LEDs are laid in the desired position, a calibration procedure is run. This calibration is done by taking a series of images of the lights using a Raspberry Pi camera mounted on a tripod. These images are then used to produce a 3-D mapping of each of the lights. With the calibration completed, the lights can be individually programmed to display different colors, or a sequence of colors, using a web tool. The coordination of each of these pieces is done with another Raspberry Pi that is always connected to the lights.

1.2 Requirements

Functional Requirements:

- LED’s, Lazy Suzan, and Control Box are simple to install.
- The LED’s are able to be controlled remotely yet also operate independently.
- The web server provides an interface for the control and calibration of the LEDs.
- The camera system is able to map LEDs through a short calibration process, allowing patterns to be displayed on the lights in an easily recognizable fashion.
- The calibration process is easy to follow without any prior knowledge.

Economic Requirements:

- The entire product is to be a replacement to a standard set of Christmas tree lights, and the price should be reasonable for the typical person to purchase (< $100)

Environmental Requirements:

- Should operate on both live and fake tees, indoors; without causing any harm.

UI Requirements:

- Web server interface is simple and allows the user to change between patterns in as few steps as possible.
- Web server interface does not include unnecessary features
- Calibration instructions and interface are easy to understand even for a novice
- Web server organization is minimal as possible while meeting the functional requirements

Non-Functional Requirements:

- Material cost should be under $50 in total
- Calibration should take no more than 1 hour
- The time to update a pattern/light should be no more than 3 seconds.

1.3 Project Design

This semester we began by finishing up on testing the code from the previous teams, and also began creating code which would fully implement the calibration procedure. The design uses two Raspberry Pi computers, one as a camera to calibrate the lights and the other to control the lighting of the tree. We came to the conclusion that the current design is functional, but could be improved in a number of ways. The four most important parts of the project we chose to address were optimizing the calibration, automation of the calibration procedure, reducing the communication steps between the two Raspberry Pis, and devising a more clever way to generate color patterns in the web-app.

The first aspect we chose to tackle was the optimization of the calibration procedure. To begin with, we spent a sizable amount of time trying to successfully conduct the procedure. The code documentation provided by the previous teams was not sufficient, so we ran into multiple problems running the programs. With the calibration running we ran a series of tests to determine the effectiveness of the current implementation. These tests varied the color of the lights as well as the sequence and duration of illuminating each light. After testing we decided to rework the calibration procedure, this time focusing on creating a reliable SSH connection between the Raspberry Pis and reducing the overall runtime.

The second part of our project we chose to address was the automation of the calibration procedure. By reducing the need for human interaction with the system, we diminish the potential for human error and make it easier for a non-technical person to operate. The calibration process involves rotating the tree in order to take photographs of the different faces of the object. The system originally sat on a wooden Lazy Susan to allow for rotation and relied on a person to adjust the angle when given a prompt. We chose to build a new Lazy Susan using a gearing system with a stepper motor (ROB-10551 12V). Using this we are able to control the tree’s rotation from the GPIO pins of the tree RaspberryPi as part of the calibration sequence.

1.4 Expected End Product and Deliverables

The end product deliverables are outlined in detail below, including our technical specifications as well as our estimated delivery dates:

Control Box + Lights

Delivery date: Oct 11th

The control box contains the Raspberry Pi, also known as the “Tree Pi,” along with a 12V Power Supply that plugs into a standard wall socket. This Power Supply routes through a 12V to 5V voltage regulator in order to power the Raspberry Pi, and it also powers the lights, which plug directly into the control box. The lights, which are based on WS2811 LED chipsets, are controlled through PWM control from the Tree Pi within the box. The Pi also has an attached screen that displays various information about the current state of the tree, including the IP that should be connected to the computer in order to open the user interface.

The box itself is a holiday-themed tin box with a transparent lid with a snowflake design on top. One of the sides of the box has a cutout with the various plugin supports needed to support our control box (i.e., lights, power supply, and USB ports). The box also has internal bracing with extra rods of metal screwed to ensure no damage is caused to the electrical components.

User Interface Application

Delivery date: April 23rd

The user interface application is a web application that is hosted on the Tree Pi over an Apache PHP server. The IP address is displayed on the LCD screen in the control box, and the user is able to access it from any device on the same network, which allows the customer to edit the lighting configuration on any device with a web browser and internet connection, such as a smartphone or personal computer. This web application allows the user to modify the current lighting configuration on the tree, such as activating preset patterns, creating new patterns, scanning in existing images to be mapped onto the lights, or modifying the colors of individual lights on the tree for testing purposes. The application also allows the user to initiate a new lighting calibration if the lights have been moved from the stored configuration.

Calibration Device

Delivery date: April 23rd

The calibration device, or the Tree Pi, inside the control box (made up of a Raspberry Pi, and a power supply), will work with another Raspberry Pi and an RPi camera, or the Camera Pi, which will take images that are later shared with the Tree Pi via a samba file share as a jpg file type. On the Camera Pi will be an LCD screen that will display the IP address of the RPi, and the communication with the LCD screen will take place using the I2C protocol. The motivation behind this setup is that it would make the configuration of the device fully autonomous and the user won't require an additional monitor to connect the two systems together manually.

The setup is also intended to work such that the patterns that are displayed on the tree rotates instead of being a static pattern, which is where the last senior design group left off. We will use polar coordinates to map a few patterns that are predefined and render those onto the tree for display. The procedure of optimizing the pattern will be performed inside the control box and this is explained in the following steps:

1. The Camera Pi takes a few photos and sends them to the Tree Pi
2. The Tree Pi uses a color filter to locate the LEDs using the photos that are received and starts identifying the lights at the base of the tree

   Note: We will work with colors red and purple as these will make the task of the filter easier since the processing of information can be done with two lights with one picture

3. The Lazy Susan is given the command to rotate the tree (this process is further described below) until the next lowest LED is visible in the picture and is identified on the left and right edges of the view a few seconds later using step 1 above.
4. The lights that have been identified are mapped according to how they fit numerically in the strand
5. The process repeats until all the lights from the bottom to top are mapped

Lazy Susan

Delivery date: April 23rd

The Lazy Susan will be at the base of the tree and will be powered by a 12V power supply. This component consists of a circular wooden block and 12V servo motor that can be controlled using a PWM (pulse width modulation) signal that is transmitted from the Tree Pi, which sits on top of the Lazy Susan and underneath the Christmas tree. This Lazy Susan allows us to control the rotation of the tree, which allows us to scan the entire tree in sections during the calibration. It also lets us rotate the tree for possible patterns that go around the entire tree.

1.5 Evolution from E E/CPR E/S E 491

Since the first iteration of this project, we changed a couple of things. The first change that we made was in the calibration procedure. Instead of taking an image of the tree with no LED turned on and subtracting the image of the LED that gets calibrated, we are now taking a photo of the tree with a blue LED turned on and using the function MinEnclosingCircles from OpenCV2 to detect the LEDs - this is so that if the camera gets shifted even a little bit we are sure that the LED that is being calibrated gets detected by our program, so the amount of time that the calibration procedure takes, which we initially guessed would take around 15 minutes was drastically delayed for a more foolproof way of detecting the LEDs using the updated image processing algorithm. The next big change that we made to our project since CprE 491 was in using a newly constructed Lazy Susan, albeit with a similar design and look, but which rotates better and has no obstruction at any of the points, therefore the rotation of the Lazy Susan is a lot smoother when the stepper motor is placed at the desired position in the center with a small gear to rotate the tree.

2 Implementation

2.1 Implementation Details

Our plan for this semester was following the proposed design from CprE 491. We divided our team into four subsections, each with a focus on the part of the design. The first team focused on optimizing the programs used to calibrate the lights. This involved receiving a set of parameters from the web application, taking images of each morning at different angles, and using computer vision techniques, provided by the Python library OpenCV to estimate the light's Cartesian coordinates. The second team implemented a method to map the positional coordinates produced by the first team to a conical system. This is to be used by the web application to display the positions of the lights on a 3-D tree. The third team was responsible for devising a way to map the light information after calibration onto predetermined patterns and animations of lights. This allowed for patterns to be used across different arrangements of lights. the last team worked on the motorization of the Lazy Susan. The Lazy Susan is used during calibration to rotate the tree so that images can be taken at multiple angles for more diverse data. We used a stepper motor controlled by the tree Raspberry Pi.

For this project, we made sure to meet in person at the TLA while also following the COVID-19 safety procedures, which ensured timely completion of the different tasks, such as the calibration of the LEDs and the motorization of the Lazy Susan. Our team ran through several tests that helped us clarify the best approach for LED detection, such as using MinEnclosingCircles using OpenCV library in Python instead of doing image subtraction of the image when no LED is turned on with that when we calibrate the LEDs, and we also narrowed down our selection of motors which we could work with for the Lazy Susan during our tests.

2.2 Relevant Standards

Circuit Standards: I^{2}C, 1789-2015 IEEE LED Safety Standard

Wireless Communication Standards: IEEE 802.11, SSH Protocol, IPv4

2.3 Engineering Constraints

There were a number of constraints that we needed to consider over the development of the product. First, we were limited by the hardware of the Raspberry Pi and what it is capable of doing. This resulted in us using Python for most of the project, which brought about difficulties related to the libraries we could use. Another constraint we needed to consider in our design was the SSH communication between the two Raspberry Pi's during the calibration procedure. We established the connection over a wireless access point generated by the control box Raspberry Pilastrder to execute programs remotely on the camera Raspberry Pi. However, sending and receiving the information does not occur instantaneously, affecting the overall calibration procedure time. We also needed to consider the problem of rendering LED patterns in a specified way by our client. The product needed to render each pattern, according to the LED mappings, on the fly when a pattern is displayed. Initially our solution would store a static pattern that could then be loaded. This change presented some challenges but ultimately resulted in a more usable product.

3 Security Concerns and Countermeasures

3.1 Physical Security

Security concern for our project is the control box, or the Tree Pi, overheating and potentially catching on fire which not only sabotages the whole project but also puts the user at serious Risk. To mitigate this physical danger, our team is planning to have the Tree Pi turned on with the lid of the control box always open (and operating indoors such that it avoids wet conditions), which would allow for more excellent airflow to the Raspberry Pi and the power supply - and making sure that the box always stays in a cool, and dry condition.

3.2 Cybersecurity

Our team estimates that as long as the user has WiFi access and connected to the same network as the Tree Pi, there should be little to no cybersecurity concern through this project because there is very little that the user can do when they are already using the web application to calibrate the LEDs or change the patterns on the tree. The Tree Pi, which hosts the webserver, allows very few operations, like allowing the Camera Pi to join and send files via SSH, and allows the user to connect to the network if they know the right IP address and change the patterns, or calibrate the LEDs.

4 Testing

4.1 Testing Process

Some of the functional testing challenges that our team has faced thus far include testing the calibration procedure using a web app, which gives errors when we press the button to calibrate. The primary reason for this is the logic behind the software implemented by the previous senior design team, which has many unresolved bugs and, by and large, an inefficient design that includes several versions of the same code with little to no comments.

Further functional testing challenges include the Lazy Susan which at this point is yet to be implemented with the rest of the project because while we have the apparatus ready (i.e. the wooden pieces that serve as the Lazy Susan), the functionality to rotate the tree autonomously has not been implemented yet; however, our team has come up with an idea to use a motor which would provide the tree with just enough torque to rotate the tree a full 360 degrees to complement the calibration procedure.

The type of motor that we use for the Lazy Susan is a non-functional testing challenge that we faced. We first began with a servo motor, but this proved to be ineffective in rotating the tree smoothly and completely because the servo motor that we chose only rotates 180 degrees and does not have enough torque to fully rotate a 35 lb. tree. Therefore, our current design includes a stepper motor (Nema 17), which is more powerful than a servo motor. However, using a servo motor means that our team has to limit the maximum current that is used in the system using a formula that depends on the V_{ref} in order to avoid the issue of overheating or breaking the motor driver (A4988 Pololu). The formula for I_{MAX} is as follows (as determined by the manufacturer datasheet):

$$I_{MAX} \textit{ (in Amps)} = V_{ref} / (8 * R_{cs}) \quad \text{where } R_{cs}=0.068\ \Omega$$

Therefore, if we wish to have a maximum current, I_{MAX}, of 1 Amp, we would have to choose the V_{ref} equal to 0.544 V. Another operational challenge is to make sure that the driver continuous current is greater than the current motor rating (0.16 Amps in our case) that is specified by the manufacturer (Lin Engineering Ltd.) to ensure a smooth operation for the tree rotation.

Some other non-functional testing challenges that the team has also faced include which LEDs we can avoid turning on during the calibration process to save time or if the calibration should begin from the base or the top of the tree. In addition, which software we build our programs in are also part of the non-functional testing, which can be subject to change if we wish to improve the performance of our project.

4.2 Unit Testing

Software Testing

As far as software testing, we must verify that our written software code links with the hardware parts. We came up with three main software components to work with. These components include calibration communication, calibration coordination mapping, and pattern storage.

Hardware Testing

The control box will be checked to make sure that the power supply is providing the correct outputs using the oscilloscope. There will be tests to make sure the SN74AHC125 is level shifting the 3.3 volts to 5 volts for the data pins to the LEDs. Additionally, The tree will sit on a motorized Lazy Susan that will essentially be controlled by a stepper motor (Nema 17) to rotate the tree so pictures can easily be captured.

4.3 Interface Testing

To begin with, the hardware and software interface through the Raspberry Pi pins. The power system associates with the Raspberry Pi ground and 5v pins to supply power. The ground should be reliable, so the power circulation framework detects similar voltage over the ground and 5v connections as the Pi to be fueled. The same goes for the data signal on the Pi and the 3v:5v level shifter circuit.

The camera setup interfaces the camera with the Raspberry Pi through the camera connector. This is inherent to Raspberry Pi, making it sure of its function since we didn't create it ourselves.

4.4 Acceptance Testing

We can make sure that our design requirements are met by going through a complete setup process start-to-finish just as the client would, either by ourselves or, if possible, by having the client / another person do it. This would allow us to not only make sure that we can establish connections between the Tree Pi and the Camera Pi, calibrate the tree, and save and load patterns, but also that it is all easy to set up and use. If the client can watch this process, it will allow us to know if our setup process is too complicated, and to see if the saved patterns are aesthetically pleasing and if the web app has all the client's requests and meets their specifications both functionally and design-wise. The more functional requirements, such as quick calibration and fast communication between the web app and the lights, can be tested using repeated trials.

4.5 Results

The first thing that we had to test with our project was the existing work that had been done by the last team that had this project. To do this, we had to test in stages.

We first verified that the lighting system worked, along with the power system for the control box. After plugging in the power box, powering up the raspberry pi, connecting the lights, and plugging a monitor into the pi, we were able to verify that the Tree Pi was functional and was able to send power to the lighting system.

We next needed to check that the Camera Pi was functioning correctly. After starting it up, we found that it had no internet connection and thus was unable to send pictures to the Tree Pi nor receive any instructions to take these said pictures. After troubleshooting, we found out that the script to set up the access point on the Tree Pi was no longer working correctly due to an issue with the internet settings on the Tree Pi stemming from it losing its "certificate" with ISU's NetReg system for the internet.

After dealing with this issue and connecting our Tree Pi access point running, we could connect to it and test the web app that accompanied the lights apparatus. Usually, we would have to verify that the lights calibration software worked correctly first. Still, the Tree Pi already had a stored configuration for the lights that were already strung on the tree so that we could save that for later.

For the web app, we wanted to make sure that you could save patterns, apply patterns, and dynamically change the lights on the tree individually. There were already existing patterns held on the tree, so we could load them up and confirm that they looked the same on the website vs. on the real tree. We were then able to change lights on the pattern, mainly adding a red light at the top and a blue line at the bottom, and we were able to save this pattern, load it, and see the results on the tree. Some of the lights were not in their intended locations, but we chalked this up to the lights shifting while the tree was in transit.

To make sure that it was just an outdated calibration, we needed to run a new one to make sure that there were no errors in the web app's 3D tree display functionality. However, we ran into many issues when we were calibrating.

At first, our camera didn't seem even to be accepting requests for test pictures, but this turned out to be a combination of a malformed stored IP and a caching error. However, once we were able to properly get and receive images between the Tree Pi and the Camera Pi, we ran into a slew of errors, which we could narrow down into five critical issues with the existing calibration code.

The webpage for the calibration called a python file named "joe_treesolv.py," but when running this both from the webpage and locally, we ran into quite a few issues.

For starters, there was no lock file, so the infinite loop that sends signals to the lights to display the pattern never stops running. This causes two issues: the calibration program is not able to update the lighting configuration file if the main loop is also editing this file since we can't read and write the file at the same time from two different processes. Also, since the coil is constantly displaying the pattern, the calibration program cannot show lights one-by-one for the camera to take pictures.

To remedy this problem, we tried killing the primary loop process manually. However, this led to other issues in the code. For starters, the code required a paramiko dependency for the SSH connection and a pyswarm dependency for optimization. However, the written paramiko code only works on 3.4 and below, and the written pyswarm code only works on 3.7 and above. This contradiction makes the code unable to compile no matter what version of python you try. There was also a missing OS import, and a required argument for the program was not called by the webpage. Additionally, an oversight in the main loop caused the infinite loop to be run whenever the MasterMain.py file was imported because a check for "__name__ == '__main__'" was never done.

These numerous errors forced us to decide to scrap the old calibration code and rewrite it entirely ourselves, ending the testing process on the calibration section of our project. However, the web app worked fine, as did the control box, Camera Pi, and Tree Pi.

5 Context of Work

5.1 Related Products

Another product that is similar to our project that we found in our research is called Twinkly Smart Lights (link: https://www.twinkly.com/), which allows the user to calibrate the LEDs with a mobile application and also allows them to change the patterns of the LEDs on the Christmas tree. However, this product is very expensive, in the range of $120 to $170 for a single string of lights. This product does not use any form of image processing or dual Raspberry Pi setup, which would cost under $50, as used in our project to calibrate and control the lights.

5.2 Related Literature

There is some scholarly research that has been done on computer vision techniques related to those used in our calibration procedure. One such paper presents an algorithm for detecting contours in an image or the curves that can describe the shape of an object. (link: https://ieeexplore.ieee.org/abstract/document/5557884) We didn't implement their algorithm;

however, it helped us understand the capabilities of contour detection. Instead, we used functions from the library OpenCV to apply multiple filters before finding the contours with another procedure.

Appendix A: Operating Manual

Parts List

To get started, first, make sure that you have:

- Control Box
- Lights Array
- Christmas Tree
- Motorized Lazy Susan
- Camera Pi
- MicroUSB power cable
- Tripod
- Clear, 7' area in front of the tree
- A personal device that can wirelessly connect to the internet

Tree Setup

1. Place the Lazy Susan in the desired location for the Christmas Tree, close to a power outlet.
2. Place the Control Box onto the Lazy Susan, threading the power cable through the hole in the center.
3. Plug the Lazy Susan into the Control Box.
4. Place the Christmas Tree onto the center of the Lazy Susan.
5. Evenly distribute the Light Array around the Christmas Tree, making sure to leave no significant gaps with no lights and ensuring that the final light is at the very tip of the tree.
6. Plug the Lights Array into the Control Box.
7. Plug the Control Box into a nearby power outlet.
8. Turn the power switch on the Control Box to "ON."
    - The Tree Pi inside the Control Box may take a couple of minutes to finish.
9. Check on a personal device that the Wireless Access Point "sdmay21-48" is running and can be connected to via a Wi-Fi connection.
    - If it is not shown, wait for the startup process to complete in the Control Box.
10. Connect to the access point using the password "christmastree."
    - While connected to the access point, the device will have no access to the internet.
11. Navigate to http://192.168.155.1 in the browser, the homepage of the web application.
12. Select any default pattern and hit "load" to test if the lights will work on the tree.
    - This is just to test if the lights work; the pattern will not be correct because the tree has not been calibrated yet.

Calibration Setup

1. Place the tripod around 7' from the tree.

2. Place the Camera Pi onto the tripod, facing the tree.
3. Ensure that the area between the camera and the tree is clear of obstructions and that there are no extraordinarily reflective or bright blue objects directly behind the tree.
4. Plug the Camera Pi into an outlet using the MicroUSB power cable.
5. Turn the Camera Pi on.
    - Once again, this may take a couple of minutes to finish.
6. Navigate to the Calibration page on the website from your personal device.
7. Take a test picture to ensure that the camera works and that the web application can connect to it.
    - Ensure the entire tree can be seen by the camera - if it cannot, move the tripod accordingly.
8. Count the number of lights between the bottom of the tree and the control box.
    - These are hanging, and they should not be included in the calibration or in displaying patterns.
9. Enter this number into the input box, and click the “Begin Calibration” button.
10. Wait for this process to finish, making sure nothing disturbs the tripod, and no obstructions appear between the camera and the tree.
    - This process can take up to an hour.
    - Throughout the calibration process, the tree will occasionally rotate upon the Lazy Susan -- make sure that rotations will not get the tree tangled or damage anything nearby.
11. Once calibration is finished, remaining decorations such as tinsel and ornaments may be applied to the tree.
    - For ideal pattern viewing, make sure that these decorations do not obstruct any of the lights.
12. The Camera Pi and MicroUSB power cable can also be put away, as they are only necessary for calibration.

Displaying Patterns

1. Once calibration is complete, patterns can be displayed on the tree.
2. Patterns are stored as circular images which are mapped onto the tree.
    a. The center of the circle represents the tip of the tree
    b. Colors farther from the center are mapped to colors farther down the tree, where the circle’s edge is the bottom of the tree.
3. Patterns can be uploaded in a .jpg or .png file format using the “Upload Pattern” button.
4. The web application also comes with default patterns, such as gradients and other images.
5. These can be displayed simply by selecting the pattern on the sidebar and clicking “Load Pattern.”
6. These can also be rotated around the tree and even animated to rotate automatically, using the corresponding parameters on the web application.

Appendix B: Alternate Design Versions

Lazy Susan

We designed a few different versions for an updated Lazy Susan. We settled on the design that could be modified and include space for design changes if one design failed. Our design consists of a gear on the bottom piece of the Lazy Susan, which the gear attached to the motor turns on. An alternate design would be to apply a large motor further from the center of rotation on the bottom piece of the susan. This would reduce the overall torque needed to turn the tree and allow for a wider range of stepper motors.

Another alternate design element would be to add sensors to the susan to read values and understand when the tree has turned 90 degrees. We looked at a number of different optical sensors and switches to be placed on the Lazy Susan. An optical sensor would be the best option for understanding the turn of the Lazy Susan. A switch would also work. The main purpose of this is to understand where to start decelerating the stepper motor so it stops at the correct location.

Appendix C: Other Considerations

Internet

Due to the university having a very ‘secure protected’ network, we were unable to connect the Raspberry Pis directly to the internet. We instead had to set up wireless hotspots and connect to them whenever we wanted to download something. This also emulated the end user’s home system which allowed for more accurate testing, but with the disadvantage of inconsistent internet connectivity.

Library support on Raspberry Pi

Some of the libraries we used did not install seamlessly on Raspberry Pi. OpenCV was the real trouble maker. We had to download and compile the source code by hand. This process took roughly 6 hours from start to finish.

Later in development, once we had the TreePi set up as an access point, we realized we needed to install some more libraries on it. But since it was not able to connect to the internet, we instead downloaded the packages to our computers and FTP transferred them onto the Raspberry pi and installed them that way.

COVID-19

COVID-19 was a big obstacle while working on this project. The biggest problem was having an easily accessible work environment. We could not reserve a room accessible only to our team; we had to set up space in the TLA. COVID-19 safety protocols meant that in order to work in the labs, students must reserve time to work and could only work during that time. The lab was also being used by other classes for lab experiments. This meant that we could not meet and work during other classes' scheduled lab times. The lab was also only open at certain times during the week and even less on the weekends. This also limited the time we were able to spend working in the lab. Overall, COVID-19 served as a significant obstacle for working on our senior design project.