Holiday Light Display

Design Document

Team 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: 10/25/2020 (Version 2)

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

Circuit Standard: I2C Protocol

Wireless Communication Standard: IEEE 802.11, SSH Protocol

Summary of Requirements

List all requirements as bullet points in brief.

- Light display, and Raspberry Pis should function at typical ambient temperatures during Christmas
- Lighting pattern should be configured within a short time (typically a few seconds)
- The camera should function in the dark if the Christmas tree is set outside or next to a window
- The lighting display should be easily controllable by the user

Applicable Courses from Iowa State University Curriculum

All ISU courses whose contents were applicable to our project are listed below:

- EE 201 / 230
- COM S 227 / 228
- CPR E 281 / 288
- S E 329
- COM S 309 / 311
- MATH 166 / 265

New Skills/Knowledge acquired that was not taught in courses

Some new skills/knowledge that our team acquired which was not part of our Iowa State curriculum in order to complete this project are as follows:

- Speech communication
- Using Python libraries
- Sifting through data in other developers' programs
- Working with multiple microcontrollers that interact synchronously with one another

Table of Contents

1 Introduction 5

1.1 Acknowledgement 5

1.2 Problem and Project Statement 5

1.3 Operational Environment 6

1.4 Requirements 6

1.5 Intended Users and Uses 7

1.6 Assumptions and Limitations 7

1.7 Expected End Product and Deliverables 8

2 Project Plan 10

2.1 Task Decomposition 10

2.2 Risks And Risk Management/Mitigation 11

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 12

2.4 Project Timeline/Schedule 13

2.5 Project Tracking Procedures 14

2.6 Personnel Effort Requirements 14

2.7 Other Resource Requirements 16

2.8 Financial Requirements 16

3 Design 18

3.1 Previous Work And Literature 18

3.2 Design Thinking 19

3.3 Proposed Design 19

3.4 Technology Considerations 20

3.5 Design Analysis 20

3.6 Development Process 21

3.7 Design Plan 21

4 Testing 23

4.1 Unit Testing 24

4.2 Interface Testing 25

4.3 Acceptance Testing 25

4.4 Results 25

List of figures/tables/symbols/definitions

- Figure 1. Preliminary Gantt Chart Showing the Task Decomposition (starting Sept 27 and ending Nov 15) (section 2.1), pg. 10
- Figure 2. Annotated Gantt Chart Depicting each Project Deliverable (section 2.4), pg. 12

- Figure 3. Work Breakdown Structure Diagram Showing Our Plan for the Two Semesters (Sept 2020 - Nov 2020) (section 2.4), pg. 13
- Table 1. Personnel Effort Requirements (section 2.6), pg. 14 -15
- Figure 4. Module Diagram of Use-case 1 for the Website (section 3.7), pg. 22
- Figure 5. Module Diagram of Use-case 2 for the Web Application (section 3.7), pg. 22
- Figure 6. Calibration Code Errors (section 4.4), pg. 26

1 Introduction

1.1 Acknowledgement

We would like to thank Dr. Daniels for his assistance in the progress of our project. His guidance and wisdom were invaluable in helping us tackle some of the difficult problems we encountered. We would also like to acknowledge the previous senior design teams, for their contributions to the project thus far.

1.2 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 a variety of 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.

1.3 Operational Environment

The light system should be usable indoors as well as outdoors. This is because holiday lights are often used as part of decoration schemes year-round, in a variety of conditions such as rain or snow. Therefore, any electronics in the final products need to be properly protected. Furthermore, the calibration procedure, which relies on taking photographs of the lights, should also function regardless of background. This is important because it is common for Christmas trees to be placed in front of windows.

1.4 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 control and calibration of the LED's.
- Camera system is able to map LED's 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:

- Entire product is to be a replacement to a standard set of christmas tree lights, the price should be reasonable for the typical person to purchase.

Environmental Requirements:

- Should operate on both live and fake trees, 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

1.5 Intended Users and Uses

There are two intended types of users of this product, the calibrator and the end user (in many cases these will be the same person). The calibrator is to set up the lights on a tree or other holiday themed shrubbery, along with the Lazy Suzan and the Control Box. Then the calibrator must use the camera to teach the controller where the LED's are placed on the tree. To do this they must be able to use a smartphone or computer to connect to the Control Box's WiFi network, and then follow the included instructions. The end user will use the Web server interface to program the Control Box, choosing patterns and colors to be displayed on their holiday decor of choice. This should be very easy for even a novice to complete. To do this end user must also be able to connect to the Control Box's WiFi network, but no other setup is required.

1.6 Assumptions and Limitations

Assumptions:

- The user has a working WiFi connection that is compatible with the Raspberry Pi and the Camera Pi.
- The user has a Christmas Tree or some equivalent object to house the lighting system.
- The user has enough space to put the camera at least six feet away from the lighting system.
- The user has a device that is able to access the website that the control box hosts.
- The user has working electricity.

Limitations:

- The speed of the light calibration is limited by how fast the Camera Pi is able to transmit the images to the control box.
- The accuracy of the light calibration is dependent on a low number of distractions in the image with the tree.
- The speed of patterns is limited by how fast the lights are able to change based on commands given by the control box.

1.7 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 from the computer in order to open the user interface.

The box itself is a holiday-themed tin box with a transparent lid that has 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 in to ensure no damage is caused to the electrical components.

User Interface Application

Delivery date: April 23th

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 webapp 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 23th

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 a 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 23th

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.

2 Project Plan

2.1 Task Decomposition

The task of optimizing the lighting pattern on the Christmas tree can be decomposed into the following set of subtasks below (please view figure 1 which follows the list below showing the tentative schedule for each subtask):

1. Prepare the Raspberry Pi hub enclosure, or the Tree Pi, so that it acts as a 'smart' device that gives instructions to the Camera Pi and requests certain data to be delivered back, namely the photos of the tree with the lights turned on taken from a fixed distance.
    - For this subtask we are planning to use the IEEE 802.11 Wifi standard, as well as the SSH protocol for linking the two Raspberry Pis to the network for a secure communication channel
2. Prepare the Tree Pi for optimization of the light pattern using image processing techniques is equally important in achieving the client's requirements
    - For this subtask we are using digital image filters, like the standard hsv filter, as well as the Python programming language to efficiently process the images that are transmitted to the Tree Pi from the Camera Pi
3. Configure the Camera Pi so that it acts as a 'dumb' device that follows the instructions of the Raspberry Pi hub enclosure
    - For this subtask we will limit the functionality of the Camera Pi to only two tasks: take photos of the Christmas tree, using the Pi camera and send those over to the Tree Pi
4. Set up the website so that the Raspberry Pi hub enclosure can connect with the user in setting up the pattern
    - For this subtask we will work with, and elaborate on the work that was already done by the previous senior design groups
5. Set up the lazy susan such that the motors synchronize with the optimization in subtask #2 above, and follow the commands of the Tree Pi
    - For this subtask we will use pwm (pulse width modulation) to control the motors that rotate the Christmas tree
6. Render the patterns in 2D and map them in 3D
    - For this subtask we will work with mathematical tools and scripts using Python, and/ or C++

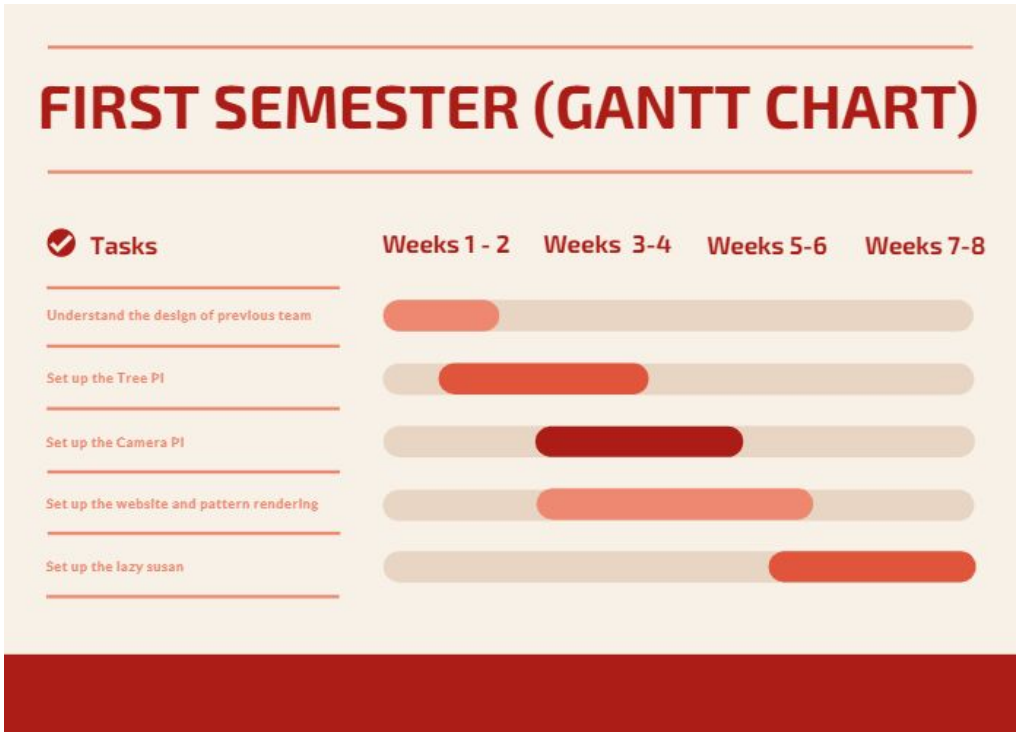

Figure 1. Preliminary Gantt Chart Showing the Task Decomposition (starting Sept 27 and ending Nov 15)

2.2 Risks And Risk Management/Mitigation

The following is our assessment of the risks involved in each corresponding subtask listed in the Task Decomposition section above:

1. Setting up the Tree Pi; our estimated probability for risk for this subtask is 0.6
    a. The high probability of risk is mainly because this is a main component which needs to work with several other components in this project, so there is a high likelihood that some components do not work, or there are bugs in code which can cause system from booting up properly or functioning as desired
    b. Risk mitigation plan: Use plenty of backups throughout the course of the project, or use a newer Raspberry Pi board, if necessary, if there is issue with power consumption and heat
2. Setting up optimization capacity on the Tree Pi; our estimated probability for risk for this subtask is 0.4
    a. The probability of risk is low because several previous senior design groups have already worked on this and the project seems to work during our tests

3. Configuring the Camera Pi; our estimated probability for risk for this subtask is 0.5
    a. The probability of risk is relatively high because the previous senior design team configured the Camera Pi also as a 'smart' device; in other words, part of the image processing takes place on the Camera Pi, which we do not want ideally
    b. Risk mitigation plan: During the first semester of this project, we narrow the image processing capacity down to only the Tree Pi and remove all unnecessary functionality currently resting on the shoulders of the Camera Pi
4. Setting up the website; our estimated probability for risk for this subtask is 0.2
    a. The probability of risk is low because this step is already taken care of by the previous senior design teams who worked on this project, and we build on whatever is already accomplished from before
5. Setting up the lazy susan; our estimated probability for risk for this subtask is 0.4
    a. The probability of risk is not too high because we don't expect a failure in this subtask to jeopardize the entire project. This is because the rotational speed is under control most of the time and it would be someone closely monitored by the team during testing
6. Rendering the patterns in 2D and mapping them onto 3D; our estimated probability for risk for this subtask is 0.6
    a. The probability of risk is high because the code and the math that we have to use to compute the values for the optimization could pose a great challenge, and that putting the lights on a different surface may not work as planned when we optimize the display
    b. Risk mitigation plan: Make sure that some pattern is displayed on the tree that is taken from backups of the code and math used earlier, and verify that optimization works correctly at least in theory for a surface, namely that of a cone, that is well-defined and integrates with the rest of the components

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Some key milestones and metrics in our proposed project are as follows:

- Commenting all the code (Oct 5th)
- Testing the design of previous teams (Oct 9th)
- Camera Optimization (Oct 23rd)
- Update Lazy Susan (Oct 26th)
- Light Mapping (Nov 6th)
- Light Pattern Rendering (Nov 13th)

Some key criteria which will be used for evaluating the project are as follows:

- The code is easy to follow such that if another team were to finish this project in SE/CPR E/EE 492, they will be able to understand the logic and all the interactions between each component
- The Camera Pi successfully takes photos of the tree from a fixed distance of 6 ft. and sends it to the Tree Pi
- The Tree Pi runs all the functions or scripts to optimize the light pattern and also hosts the website for the user interface, and communicates with the Camera Pi
- The light patterns are well-defined and the code used as well as the math logic are functional in the rendering of the patterns onto the Christmas tree
- The lazy susan rotates the Christmas tree according to the desired rotational speed while the pattern is displayed on the tree; the rotational speed must not be too fast such that the tree may fall

2.4 Project Timeline/Schedule

For our project, the following figure 2 shows an annotated Gantt chart with all our deliverables for the first semester as well as the estimated time for completion:

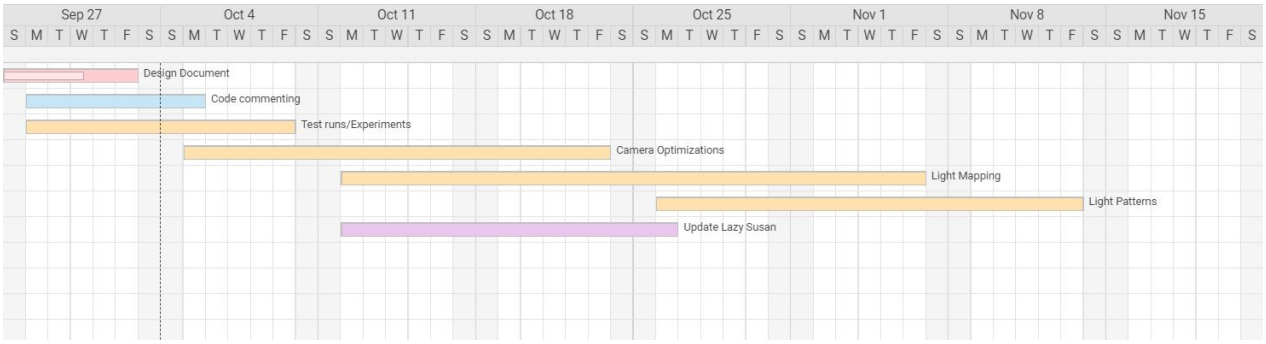

Figure 2. Annotated Gantt Chart Depicting each Project Deliverable (Sept 2020 - Nov 2020)

Our team is responsible for completing this project by May 2021. For this project, we will seek to accomplish as much as possible by Nov 15, 2020, and pick up where we left off in Spring 2021. Our plan is to do the testing involved with learning the code and design of the previous teams in the first semester, and edit and make changes to the code and define the patterns and figure out how we would render the pattern onto the tree. And, the plan is to continue testing and refining our design in the second iteration. The figure 3 below

displays our summarized plan for the full development through till completion of this project:

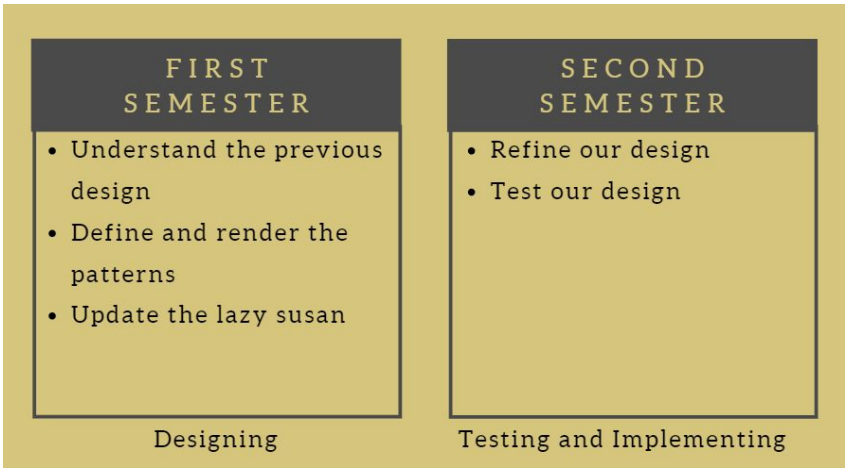

Figure 3. Work Breakdown Structure Diagram Showing Our Plan for the Two Semesters

2.5 Project Tracking Procedures

During the course of this project, a variety of project management tools will be used to help keep track of tasks and important due dates.

As a way of keeping track of assignment due dates and other assignments we will be using Trello to set and check on the progress of assignments due. Trello is easy to use and allows us to easily see what assignments are due and who is working on what. When it comes to the coding/implementation side of the project we will be using git and github to push code and work together effectively on the project. Using git will allow us to show what each project member is working on and any issues that come up. Using git will also allow us to push our code and not work over or edit someone else's code. We can work and merge code so that everything gets done in a timely and efficient manner.

2.6 Personnel Effort Requirements

Team Members	Personnel Effort	Hours
Christopher Woods	• Determine all technology requirements of the project and prepare	Varies

	appropriate growth and development plans. • Develop, execute and maintain achievement of all departmental goals and objectives. • Monitor competition and process, assess market requirements and prepare strategies accordingly.	
Jacob Martin	• Administer all computer operations and provide support to all production systems • Identify and resolve software problems as they arise • Stay up to date on technologies in order to ensure a successful procedure	Varies
Ashkirat Singh	• Establishes meeting dates and agenda to achieve immediate and long-term team goals. • Facilitates member interactions to prepare attendees for informed contributions and consensus decision making • Guides team to decisions, assignments, problem resolutions, resource allocations that position members for productive work	Varies
Mitchell Wadle	• Distributes meeting notice (date, time, place, agenda) days before meeting. • Distributes information to be discussed. • Documents in key elements of meeting discussions, actions taken, assignments made, next meeting date and purpose • Posts meeting minutes to	Varies

	team repository within 24 hours of meeting	
Ty Gardner	• Develops and monitors schedule for design of power system • Represents power system design in overall project • Leads effort on power system design, delegating and monitoring work as needed to deliver expected work on schedule	Varies
Joyeux Noel	• Establishes timeline for report preparation, with intermediate deadlines • Leads assignment of materials preparation to individual members • Oversees preparation and receipt of report components • Leads report integration, proof-reading, and packaging for submittal	Varies

Table 1. Personnel Effort Requirements

2.7 Other Resource Requirements

With the idea of improving this project by taking the system created by the previous senior design group, there are a few things we might need to make this improvement. We are looking to motorize the lazy susan, in making that happen, a motor is required to easily rotate the object (tree). Other than that, anything that gets damaged throughout the process of our project will be replaced.

2.8 Financial Requirements

Taking off with what the previous team left behind, there is a possibility that some components may require some finances. If necessary we may get an additional RPI in supporting what's already there. Christmas lights (LED's) can be fragile and in any case that happens, we will need to purchase new ones. If to occur, all that will require finances. Additionally, for any other parts that break throughout the course of our project building, to replace it will require finances.

3 Design

3.1 Previous Work And Literature

We took over this project with a great deal of work completed. As described by our client and the previous teams design document, things already accomplished include:

- Math for understanding where the lights are on the tree
- The tree and camera pi's and the tree components
- Written code for calibrating and running the patterns
- The web page of changing the design patterns and calibrating

Advantages:

- Most of the math, physical components and code has been written and designed already. This saves us time by not having to design and code everything but instead work on optimizing everything already working.
- We were given a project that runs and produces some final result which we can work on and optimize.
- Left a great base to continue working and optimize code and processes left over.

Shortcomings:

- A great deal of the code left for us was uncommented and, at times, hard to read. After looking at the code, there seems to be a great deal of code that is not used in some files. We concluded that this was an attempt at making changes and updating some elements of the code that ended up not working. Instead of deleting the non-working code, they just left everything in the files without any explanation.
- Running some of the code left behind has shined a light on some issues in calibrating the lights and updating the patterns correctly.
- Not sure if working through the issues present would take less time than redoing entire sections of the project

This kind of product exists but not designed the same way. This product has been developed in a number of ways including using an arduino to program when the lights turn on and (using an app) set lights and patterns on the tree given you set the lights on the tree right.

3.2 Design Thinking

The first choice we made was to go over everything left over from the last team and see what works and what doesn't. During these tests, we came up with a few things to make a priority and a few things as extra that we can work on if we get to it.

Priority items to get to:

- Get what we have working as described in the other teams design document.
- Update/optimize code and patterns
- Update lazy susan

Extras:

- Redo light capturing process (to see if we can more accurately capture lights)

3.3 Proposed Design

We began the semester by testing the design implemented by the previous senior design teams. 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. We are still in the process of determining the most effective calibration procedure, but this will result in a more reliable and user friendly product, both of which are requirements for the project.

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. Currently the system sits on a wooden lazy susan to allow for rotation, but relies on a person to adjust the angle when given a prompt. We chose to attach a stepper motor to the lazy susan, controlling it from the GPIO pins of the tree RaspberryPi. This will allow for the rotation to be conducted by the calibration program sequence. We are currently in the process of testing the stepper motor to see whether it will be able to provide sufficient torque to rotate the lazy susan, as well as whether it will be consistent enough to rely upon.

So far we have not tried to change anything with respect to reducing the communication steps between the Raspberry Pis or improving the pattern storage system on the web-app. However, we plan to refactor and combine some of the programs that are currently part of the calibration process. Doing this should highlight some of the redundancy that is present and help us achieve a quicker calibration.

3.4 Technology Considerations

There are a variety of trade offs made in terms of the technology used in this project. One strength is the ease of use of the CameraPi, this is very forgiving because it is a very popular camera module with many compatible tools and libraries. This helps make the code on the CameraPi fairly simple at least in theory. This is especially nice considering the clunkiness involved in just SCPing the image data between the Pis. Another approach could have been to do more processing on both Pis and just SCP the data with the LED locations between them, then both would be fairly complex in terms of their individual code.

A weakness of our design is that the user and the CameraPi both need to be on the TreePi's WiFi network. This means that the user cannot be connected to the internet while setting up the system. It also means that no updates or development may be remotely deployed to the system. Ideally this system could connect to the users home WiFi network in addition to the current system, this way the user could manage the lights without disconnecting from their home network and updates could be pushed over the internet.

3.5 Design Analysis

We are still in the process of redesigning and testing the calibration procedure, but when we are finished it should work and be more efficient than the previous iteration. Initially

we thought to use a servo motor to control the rotation of the lazy susan, but discovered that the servo motor we had access to was not strong enough and only rotated 180°. To overcome this we decided to use a stepper motor instead, which should provide us with the torque and range of motion that we need.

3.6 Development Process

Our project is taken over from a previous Senior Design group, this makes the Waterfall development strategy the most practical. It isn't very reasonable to revise the core plan created by the previous group. It makes more sense for us to pursue a continuous process of testing, designing, and implementing. Due to the fact that we took over from a previous project group, we have the privilege of having a sizable code base to test and revise as issues arise.

Despite the limitations in terms of scope of the project having taken over from another group, a component where we are able to have more developmental discretion is with the Lazy Suzan system. We chose to redesign this component to make the rotation automated and streamline the calibration process. We have been pursuing a agile development model throughout this process, breaking the project into smaller portions that can be tested on their own in order to build up to the final product.

3.7 Design Plan

Within the context of our project requirements, a few use-cases would look like the following:

Use-case #1:

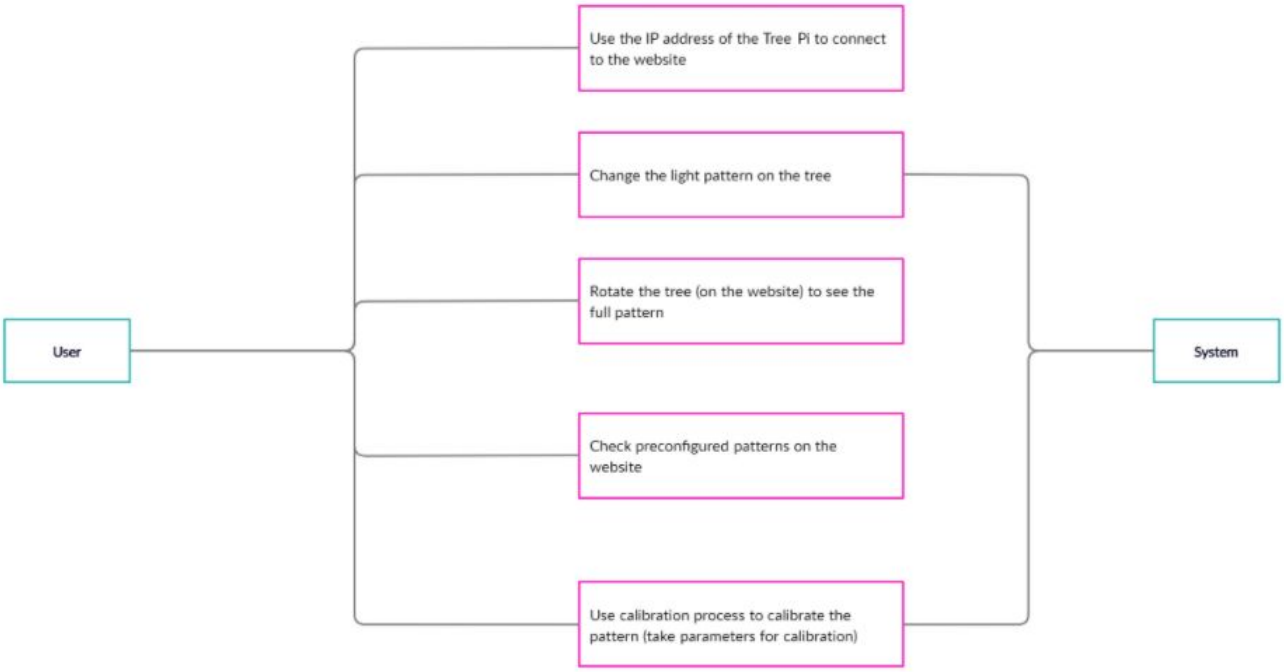

Figure 4. Module Diagram of Use-case 1 for the Website

Use-case #2 (similar to the use-case #1 above):

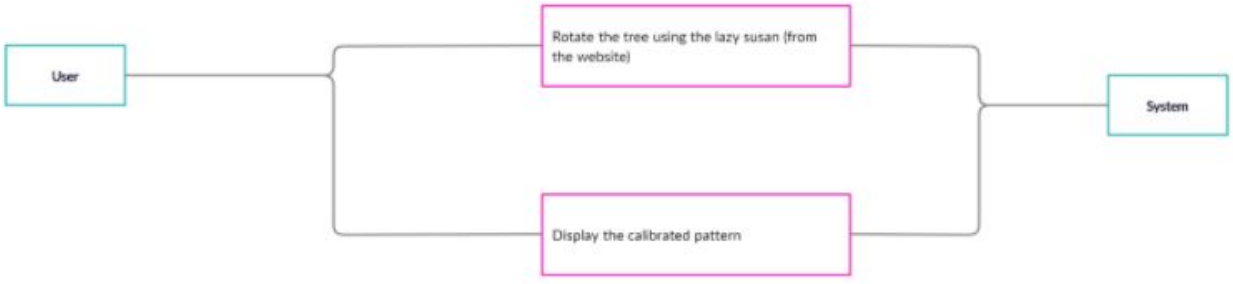

Figure 5. Module Diagram of Use-case 2 for the Web Application

Some of the module constraints tied to the project requirements are as follows:

- Our project is limited to the software development tools already utilized by the previous team in our calibration; therefore, we will have to either improve the code that is already being used, or come up with our own which meets the desired specifications
- We have to use the webapp for the calibration; so, we would need access to a Wi-Fi at all times
- We have to choose V_{ref} that is smaller (typically 0.544 V for our operation) than what the system is capable of handling in order to ensure that the maximum current flowing through stepper motor (used in the lazy susan) does not result in overheating or any significant damage to the Raspberry Pi and the motor driver, and also that the lazy susan has a steady, and smooth rotation
- We also have to use a decoupling capacitor (at least 47 µF, but typically 100 µF) to protect the driver board of the stepper motor (A4988 Pololu) from any voltage spikes

4 Testing

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

Further functional testing challenges includes 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):

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

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 functional challenge is to make sure that the driver continuous current is greater than the motor current rating (0.16 Amps in our case) that is specified by the manufacturer (Lin Engineering Ltd.) in order 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.

4.1 Unit Testing

Software Testing

As far as software testing, it is crucial we 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.

4.2 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 so the Pi can be fueled. Same goes with the data signal on the Pi and the 3v:5v level shifter circuit.

The camera set up 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.

4.3 Acceptance Testing

We can make sure that our design requirements are met by going through a full 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 are able to 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 is able to 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 webapp 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 webapp and the lights can be tested by us with repeated trials.

4.4 Results

The first thing that we had to test with our project was the existing work that had been done by the previous team that had this project. In order 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 properly. 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

properly 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 were able to connect to it and test the webapp that accompanied the lights apparatus. Normally we would have to verify that the lights calibration software worked correctly first, but the Tree Pi already had a stored configuration for the lights that were already strung on the tree, so we were able to save that for later.

For the webapp, 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 saved on the tree, so we were able to 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, particularly 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 webapp’s 3D tree display functionality. However, we ran into many issues when we were calibrating.

At first our camera didn’t seem to even 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 were able to narrow down into five key issues with the existing calibration code, shown below in Figure 6:

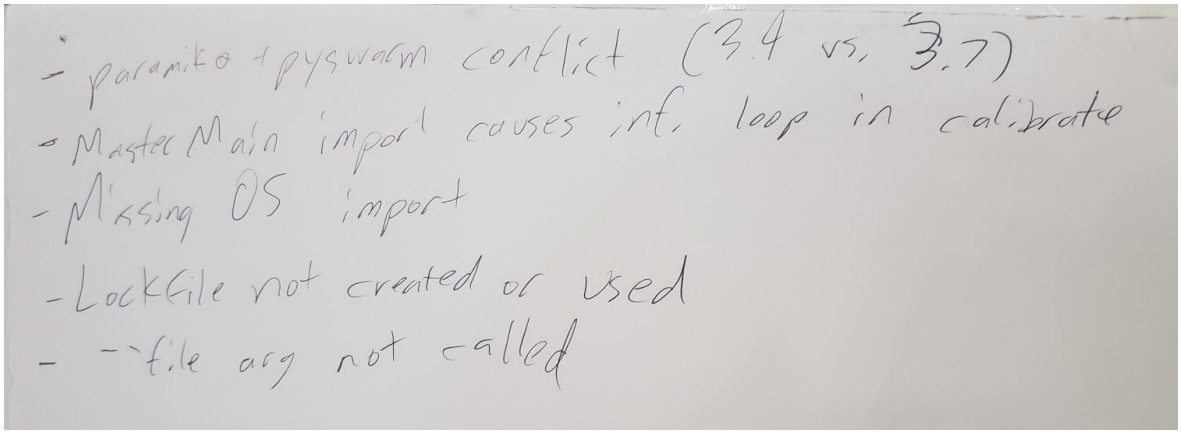

Figure 6. Calibration Code Errors

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 lockfile, 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 loop is constantly displaying the pattern, the calibration program isn't able to display lights one-by-one for the camera to take pictures.

To remedy this problem, we tried killing the main 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 make the decision to scrap the old calibration code, and completely rewrite it ourselves, ending the testing process on the calibration section of our project. However, the webapp worked fine, as did the control box, Camera Pi, and Tree Pi.