Holiday Lights Project

Team Number 10

Client
Dr. Daniels

Adviser
Dr. Daniels

Team Members/Roles
Steven Williams/Hardware Lead
Jake Grace/Software Lead
Joseph Nunez/Meeting Scribe
Valery Smith/Signal Processing Specialist
Thien Nguyen/Webmaster
Chad Griggs/Report Manager

Team Email
sddec19-10@iastate.edu

Team Website
Sddec19-10.sd.ece.iastate.edu

Table of Contents

Frontal Material

List of Figures

List of Definitions

1) Raspberry Pi - Also referred to as “Pi”, is a “low cost, credit-card sized computer” that “enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python” [1]
2) LED - Also known as a “Light Emitting Diode”, these “produce light approximately 90% more efficiently than incandescent light bulbs” [2], are substantially smaller and are more readily individually programmable.
3) SSH - The Secure Shell Protocol “is a method for secure remote login from one computer to another” and allows for the use of the File Transfer Protocol. [3]
4) RGB - Red, Green, Blue. The three colors human eyes are tuned to receive and the three colors used in a pixel.
5) PWM - Pulse Width Modulation. The variance in time high over time total (duty cycle) in a time varying digital signal.
6) LAMP - Linux Apache MySQL PHP
7) JSON - JavaScript Object Notation, is a lightweight data-interchange format which is platform independent [15].

Introduction

Acknowledgement

We would like to acknowledge Dr. Tom Daniels for his time and expertise with this project, as well as the previous senior design group for laying a basic foundation onto which we improved. Also, we would like to thank Matt Post and Lee Harker from ETG for help with ordering parts.

Problem and Project Statement

During the holiday season, it is customary to display colorful lights on trees or other surfaces such as roofs or bushes for the enjoyment of family, friends and neighbors. We have created a system that shifts from the traditional static displays to interactive and dynamic displays.

A previous senior design group had previously made an attempt at this using a single Raspberry Pi and an android phone app. We have improved their design by using a web based interface in lieu of the android app, and a second raspberry pi to act as a standalone camera and image processor.

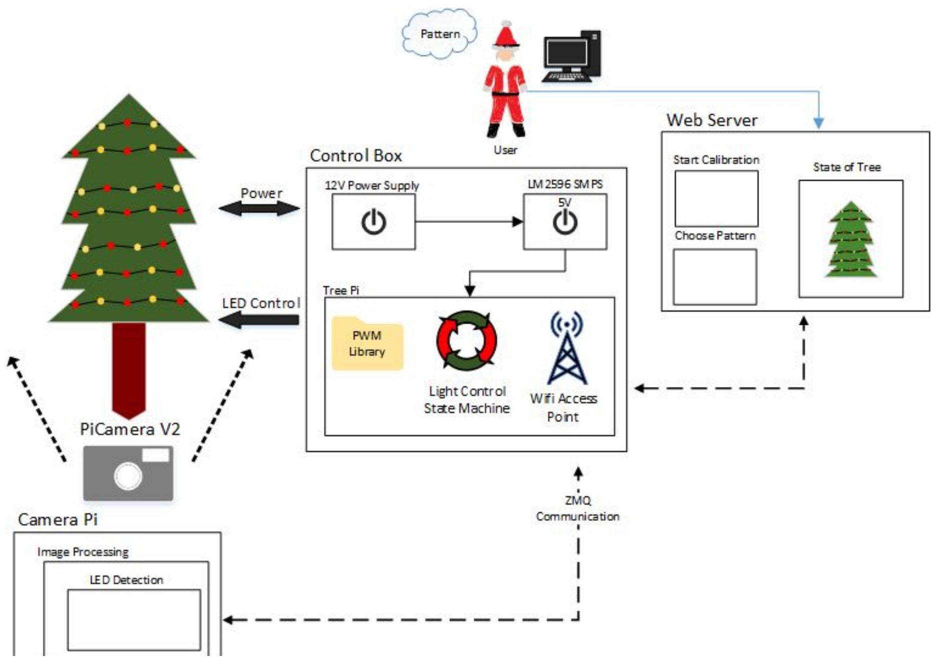
Project Design

The physical system consists of two major sections referred to as the “TreePi” and “CameraPi”. They communicate with each other using SSH protocols.

The TreePi consists of; a lazy susan upon which the tree is placed and can easily be rotated for calibration purposes, a decorative reinforced metal box to house the electronic components, and the electrical components within the box. The components in the box are; an IEC320 C14 120VAC female power adaptor, 120VAC to 12VDC 30A power supply, 12VDC to 5VDC buck converter, the Raspberry Pi itself coupled with a real time clock, and a WS2811 based LED strand which can contain an arbitrary number of LEDs (250 lights were used for our test setup). The TreePi uses a simplified LAMP stack for hosting the web server, minus the MySQL. It also uses Python as the brains of the light control system.

The CameraPi system consists of; a Raspberry Pi coupled with a Raspberry Pi camera, a case to house the camera and Raspberry Pi, a Micro USB power adaptor and cable to supply power to the system, and a tripod where the components above are mounted. The CameraPi uses OpenCV Python bindings for analyzing the photos taken with the pi camera and communicating the data back to the TreePi.

Fig. 1



Implementation Details

Hardware Implementation

The system operates at standard DC voltage levels including 12V (input power), 5V (light logic and Raspberry Pi power), and 3.3V (Raspberry Pi logic). The current draw of the overall system varies greatly depending on the brightness and number of lights lit at a given time, however, it generally ranges from 0.75A (with the lights off and the TreePi in a standby mode), to 6A (with the TreePi hosting the web server and access point, performing calibration, and the lights at full color and brightness).

As can be expected, drawing this much power leaves much of it dissipated as heat within the metal case. The lid and siding are perforated to allow airflow and maintain a consistent, cool temperature around the electronics. The case itself was originally constructed using a recycled cookie tin which proved to be ineffective at maintaining its general form, so it was reinforced by aluminum plates. Additionally, as the case is conductive, appropriate grounding and isolation measures were taken to avoid risk of electrocution.

Software Implementation

The TreePi has a constant Python state machine that starts on boot and runs indefinitely. This manages the LED strip on the tree by checking the file, LED_cordinates. JSON for changes made by the webserver and updates the LED strip accordingly. The Apache server also starts on boot-up and is how the user interacts with the Python state machine. The TreePi is set up as a wireless access point, this is so it is a closed system (it can not be accessed from outside entities) and so it can have a static IP and be available for the CameraPi to connect to it (The CameraPi also having a static IP on the network).

The CameraPi connects to the TreePi access point by default. This allows them to communicate via SSH for when the TreePi issues commands to the CameraPi without any special set-up by the user. From there, the user interacts with the system via the home page hosted on the TreePi where they are able to save and update the tree, as well as load preset patterns. The user also has the option to run the calibration sequence by navigating to the calibration page and following the instructions. The calibration sequence works as follows:

1) The user positions the CameraPi on the tripod at least 70 inches away from the tree.
2) The user takes a sample photo to ensure that the tree is in full view for optimal calibration.
3) The calibration sequence is run in which all LEDs are lit up in order and a picture is taken. The (x,y) coordinates of each LED found in the photo is saved to a file.
4) Once the LEDs have all been lit up, the user will be prompted to rotate the tree a specified number of degrees. This will capture all remaining lights
5) The output of the calibration sequence will be sent to the TreeSolv.py script that will calculate the best position for each LED based on the numerous (x,y) coordinates for each.
6) Finally, a JSON file is retrieved from TreeSolv.py that is used in the home page to update the model of the tree for the user to create patterns with.

Each pattern the user creates is saved to a JSON file that details the (x, y, z) positions of the LED as well as the RGBA values specified by the user for their pattern. When the patterns are loaded, we iterate through every LED in the strand and set that LED index equal to the equivalent LED listed in the pattern file.

Testing Process and Results

Hardware Testing

Using an oscilloscope, the voltage levels within the Tree Pi enclosure were verified. From the 120V source, it was verified that power supply outputs 12V. It was verified that the SMPS LM2596 correctly is bucking down the 12V from the power supply to 5V for the Raspberry Pi. Also using an oscilloscope, the data signal to the LEDs was verified for integrity to make sure it was within acceptable parameters. Overall, the system is able to operate for extended periods of time without overheating. The hardware containment unit is robust and does not bend or receive damage while system is in use, cords are plugged/unplugged, or in user-handling.

Software Testing

The software was testing was subdivided into different phases and methods. The calibration process was constructed in pieces that were tested individually and then slowly integrated together. We started by developing rudimentary implementations of each step in the calibration and working up to more complex methods and procedures. The bulk of the testing for calibration was spent on getting the LED detection to work as well as we could. In the end it worked better than we could have expected by actually detecting LEDs shining through the tree and not just on the surface.

The web server interface and integration testing came very late in our development. There wasn't much available to test for awhile until we were clear on how the data representing the LEDs should look and how we were controlling the lights.

References

[1] "What is a Raspberry Pi?," *Raspberry Pi*. [Online]. Available: https://www.raspberryPi.org/help/what- is-a-raspberry-Pi/. [Accessed: 20-Apr-2019].

[2] "Learn About LED Lighting," Learn About LED lights | ENERGY STAR. [Online]. Available: https://www.energystar.gov/products/lighting_fans/light_bulbs/learn_about_led_bulbs. [Accessed: 20-Apr-2019].

[3] "Home," – *Secure Remote Login and File Transfer | SSH.COM*. [Online]. Available: https://www.ssh.com/ssh/protocol/. [Accessed: 20-Apr-2019].

[4] A. Wilkins, J. Veitch and B. Lehman, "LED lighting flicker and potential health concerns: IEEE standard PAR1789 update," 2010 IEEE Energy Conversion Congress and Exposition, Atlanta, GA, 2010, pp. 171-178.

[5] IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," in *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* , vol., no., pp.1-3534, 14 Dec. 2016

[6] IEEE Standard for Camera Phone Image Quality," in IEEE Std 1858-2016 (Incorporating IEEE Std 1858-2016/Cor 1-2017) , vol., no., pp.1-146, May 5 2017

[7] LEDWORKS S.R.L. (2016). Twinkly Smart Decoration. Via Arcivescovo Calabiana, Italy. Company, https://www.twinkly.com/

[8] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000. https://opencv.org/

[9] Nxp Semiconductors. "I2C-bus specification user manual." I2C-bus specification. 4 Apr. 2014. NXP. 26 Mar. 2019 <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>.

[10] Raspberry Pi Foundation. "GPIO." GPIO - Raspberry Pi Documentation. 2015. 26 Mar. 2019 <https://www.raspberrypi.org/documentation/usage/gpio/>.

[11] JSON:API. (n.d.). Retrieved from https://jsonapi.org/

[12] “802.11-2012 - IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” IEEE. [Online]. Available: https://standards.ieee.org/standard/802_11-2012.html. [Accessed: 24-Apr-2019].

[13] R. Altkorn, S. Milkovich, and G. Rider, Light emitting diode safety and safety standards - IEEE Conference Publication. [Online]. Available: https://ieeexplore.ieee.org/document/1529515. [Accessed: 24-Apr-2019].

[14] 1st Initial. , "IEC 60320-1 ," IEC , Vol. , no. , pp. , 2015.[Standard]. https://webstore.iec.ch/publication/63846. [Accessed 4/24/2019].

[15] D. Crockford, Introducing JSON, Edition of book, : , 1998, p. .

Appendix A : Operation Manual

Getting Started

Welcome to the SDDEC19-10 senior design project. First, be sure you have all the listed parts.

- 1 Holiday Tree
- 1 Decorative controller box
- 1 lazy susan
- 1 Raspberry Pi enclosure with camera attached
- 1 power cord for CameraPi
- 1 strand of LED lights (this can consist of daisy chained shorter strands)

Setup

Follow these steps to get your holiday lights system set up.

TreePi setup

1. Place the lazy susan in the location you want to set up the system. Place your tree centered and secured on the lazy susan. Place the control box on the lazy susan under the tree. See figure 2 for example

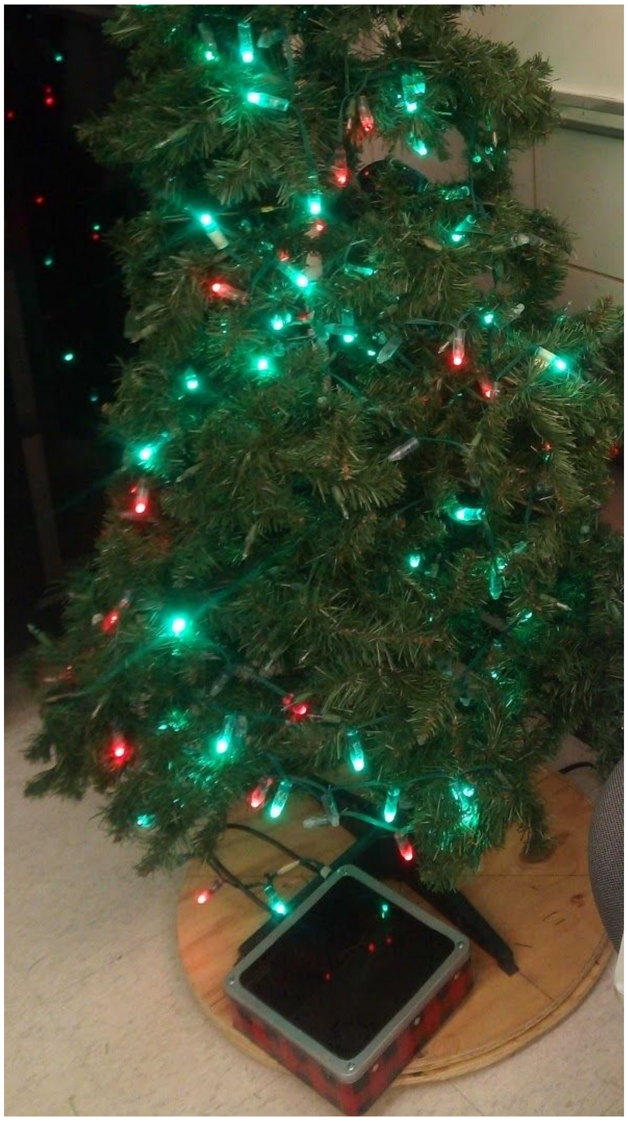
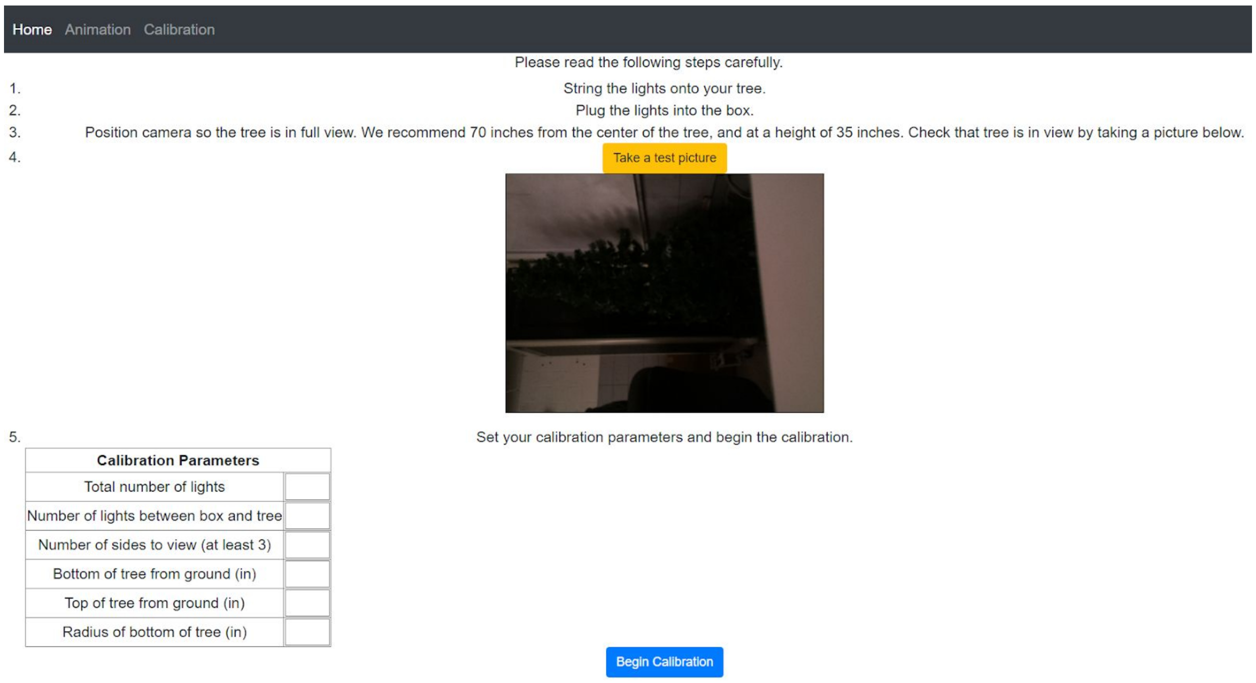


Fig. 2

   a. Make sure that you are able to plug in the power cord. You may need to run an extension cord depending on the location of your outlets
   b. DO NOT set up the system outside, this is an INDOOR ONLY use system.

2. If the tree does not have the LED strip on it, string the LED strip on the tree.

a. For best results, try to not let too many of the LEDs fall inside the tree. The closer they are to the surface of the tree, the easier it will be for the detection system to get accurate readings of the lights.



Please read the following steps carefully.

1. String the lights onto your tree.
2. Plug the lights into the box.
3. Position camera so the tree is in full view. We recommend 70 inches from the center of the tree, and at a height of 35 inches. Check that tree is in view by taking a picture below.
4.

5. Set your calibration parameters and begin the calibration.

Calibration Parameters	
Total number of lights	
Number of lights between box and tree	
Number of sides to view (at least 3)	
Bottom of tree from ground (in)	
Top of tree from ground (in)	
Radius of bottom of tree (in)	

Fig. 3

3. Plug in the strip to the control box and then plug the power cord from the lazy susan into and outlet.
4. Switch on the control box
   a. The TreePi should automatically start up the webserver and the light controller program will be running. Completed start-up is indicated by the availability of the WiFi access point.
   b. If nothing seems to happen at first, give it a little time to do it's start-up procedure. It usually takes about 1 minute to be completely ready.

5. Connect to the access point.
    a. Once the TreePi is running, you should see a WiFi access point pop up on your WiFi enabled devices. The access point will be called SDDEC19-10
    b. The password to the access point is christmastree
    c. Be aware, the access point does not actually have internet connection. This means that will not be able to do anything over the internet while connected to the access point.
6. Navigate to 192.168.155.1 in your browser.
    a. This should bring up the homepage of the TreePi web server.
7. To test and see if the whole system is working, choose a pattern loaded on the website or create a new one and load it onto the tree. If you can change the lights, the system is running properly.
    a. If you are unsure how to change patterns, see the section titled 'Creating Patterns'

CameraPi setup

1. Take the CameraPi enclosure and set it up on the tripod.
2. Place the tripod about feet away from the tree and try to point it towards the center of the tree, any misalignment will be corrected in the calibration stage, but it is best not to have to plug it in again.
3. Plug the micro-USB cable into the CameraPi and then into an outlet(extension cord may be necessary, not included).

4. Navigate to the calibration page on the website and attempt to capture a picture. If a picture is taken, the CameraPi is working correctly. If not try unplugging the micro-USB power cord of the CameraPi and plugging it back in.

Calibration

Now that your system is online and running you will want to calibrate it.

1. If you haven't already, follow the previous steps for setting up each of the Raspberry Pis
2. Once you have the TreePi online and are connected to it's network, navigate to the home page.
3. From there, navigate to the Calibration page to begin the calibration process.
4. There are some instructions on the calibration page to help guide you through the process.
    a. Position the camera such that the tree is in full view.
        i. It is recommended to position the camera roughly 70 inches from the center of the tree on the tripod.
    b. Once the camera has been placed, record the measurements asked for on the Calibration page and press the 'Test image' button to determine if the camera is properly connected and aligned with the tree.
    c. Then select the 'Begin calibration' button
    d. The camera and tree will communicate back and forth to calibrate the tree. The process will pause after each set of calibration data to allow the user to rotate the tree.
        i. Make sure to rotate the tree based on the number of rotations input

ii. 360 / sides = rotation amount

Creating Patterns

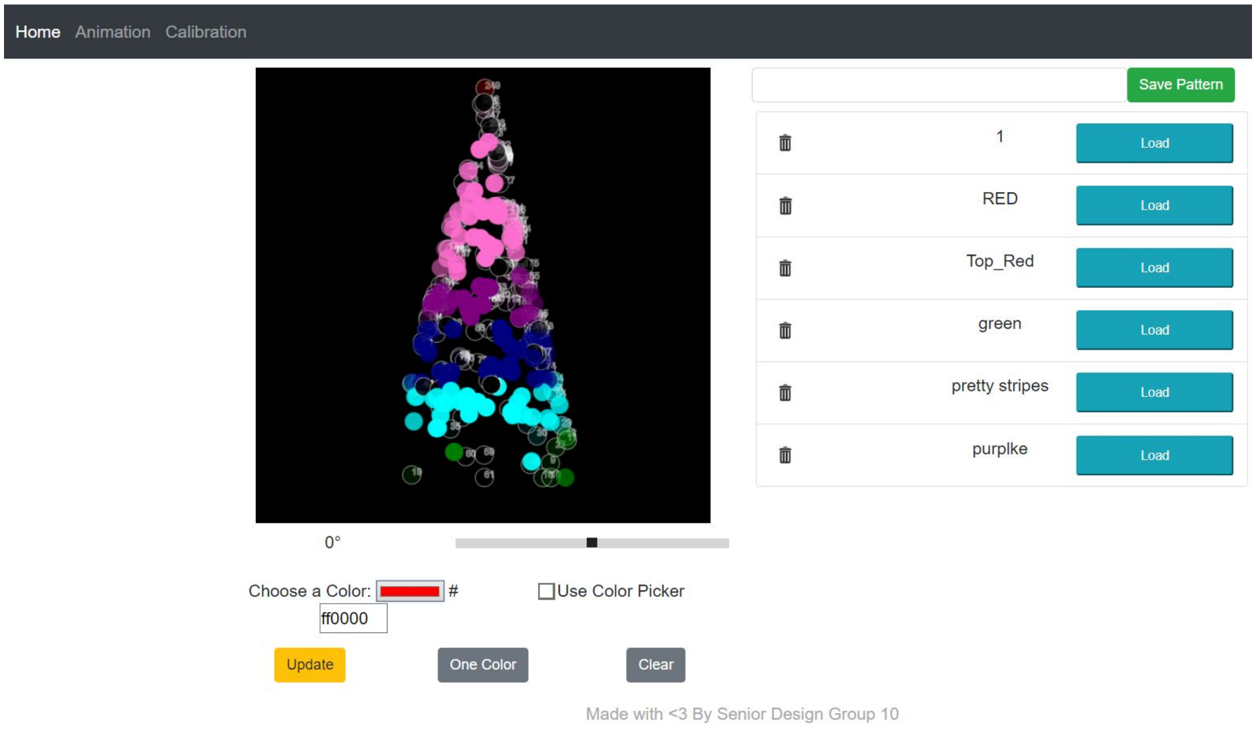


Fig. 4

1. Using the website interface, creating patterns can be fun and easy!
2. From the home page, use the provided color chooser to select the color you want.
3. Click the circles on the digital tree to change the color of the lights.
4. Use the slider to rotate the digital tree around.
5. To see how your pattern looks, use the update button to display the pattern on the tree.
6. If you are happy with your pattern, give it a name and select the save pattern button.

Troubleshooting

1. If something doesn’t seem to be working right, first try turning off the system completely and restart it.
2. If the CameraPi is not connecting to the TreePi, make sure you turn on the TreePi first and let it’s access point come online before turning on the CameraPi
3. For best calibration results
    a. Try to calibrate the system in a dark room.
    b. Try to avoid having bright blue lights anywhere near the system while it calibrates.
        i. Blue is the color used for calibration and other sources of that color can trick the system.
    c. If you follow the above steps and the calibration is still not working properly, try using a temporary backdrop behind the system such as a large dark bed sheet.

Appendix B : Alternate Design Versions

Android App

The previous team chose to use an Android app to take pictures of the tree during calibration. We decided not to do that because it would be OS specific (Android only) and would be very difficult to keep consistent over multiple devices. It also meant that all images would need to be sent to the host Raspberry pi or the image processing would need to happen on the phone. Sending that many images would be a huge amount of data and we have no way of knowing what kind of hardware each phone users have would be capable of, so the image processing could take forever. Lastly, we have no control over each phones camera so we can not expect any sort of 'standard' image to use for processing and that would make the calibration calculations extremely difficult.

Connecting to User's WiFi

An alternate way for the user to connect to the system would have been through connecting the system to their home WiFi and reaching the webserver on their own network. This allows both the system and the user to maintain access to the internet, and easier for the user to access reference images stored on the internet for assistance with their designs. However, for the user to set-up the system on their network would require the user to connect to both the TreePi and CameraPi with an HDMI monitor and a USB keyboard, and input the CameraPi IP address into the calibration page. Connecting the Raspberry Pis with an HDMI monitor and USB keyboard raises the required ability level and responsibilities of our users significantly and transports a significant amount of overhead to the user. By creating our system with its own WAP (Wireless Access Point) we increase the system's user friendliness.

//notes

- Parallax detection
- Actual 3D coord mapping (instead of just to surface)
- Connection to user WiFi (instead of AP mode)
- Patterns from images(vs hand drawing)

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.

Python

Python is touted as an easy to use, friendly programming language. Few in our team had used Python before, but during our research phase it appeared to be the best solution to our problem. Due to being easy to integrate with Raspberry Pis, and libraries available for much of the functionality we needed including the WS2811 lights. We encountered numerous issues with Python due to the number of versions available and what versions each library we used required.

The first iteration of our Raspberry Pis were using Raspbian 8 (Jessie). The GUIs relied on Python 2.7 and they also had Python 3.4 installed. Once we realized this, we reformatted the SD cards with a newer version of Raspbian and installed newer versions of Python.

Something that we realized toward the end of the project is that we could have used Python virtual environments to ensure we are calling the right python version.

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 setup 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.