Holiday Arboreal Light Project

Design Document

SDDEC18-10

Adviser: Tom Daniels
Client: Thomas Daniels
Aaron Hudson — Web Server/IPC Communication
Robert Tyynismaa — Android Developer
Rajiv Bhoopala — Web Application/Server
Michael Scholl — Android Developer
Mir Ahbab — Electrical Engineer/Microcontroller
Justin Falat — Android Developer

Email: sddec18-10@iastate.edu
Website: http://sddec18-10.sd.ece.iastate.edu/

Revised: 4-24-18-V1

Table of Contents

List of figures/tables/symbols/definitions

PWM - Pulse width modulation

RGB - Red Green Blue

LED - Light emitting diode

IPC - Interprocess Communication

YUV - Y stands for the luma component (the brightness) and U and V are the chrominance (color) components

1.1 Introduction

1.2 Acknowledgement

We would like to thank Dr. Tom Daniels for the assistance in researching and developing the plan for the project thus far.

1.3 Problem and Project Statement

Christmas is a time for celebration and with celebration comes the arrival of holiday displays. Many people decorate their homes and other objects like trees with sets of lights. However many products on the market are limited to individual design and creativity by not being customizable. For instance, in order to decorate an arboreal display in a pattern you would have first visualize that display and then lay the lights accordingly. The complexity changes when the person wants the patterns displayed to change. Thus, our team has decided to tackle this problem of being able to create complicated displays in a simple manner.

In order to provide more customization, our team wants to utilize technology in the form of smart phones and web apps. Our idea is for users to set up their lights on a tree and then upload patterns to that string of lights. Smartphone cameras will be used to record the position of LED's within the display. The data will be sent to our web app/android app where a 3d model of the display with LED's will be created. Users are then able to decorate their display through the web app/android app. Thus, the ability for customization is far greater than your average string of lights.

1.4 Operational Environment

Our light display will be set up in a person's home, most likely arranged around a christmas tree. As such our devices must not generate enough heat so as to start a fire. The system we design must also not draw too much power. Lastly, since this will be set up in a person's home and put on display, everything in the final design must aesthetically pleasing.

1.5 Intended Users and Uses

The intended users for our holiday lights are people who are interested in programmable LED lights, but are also interested in arts and gadgets. Our end goal is to create a product that can be used by anyone, even those who are not familiar with technology.

Although our project is mainly focused on creating programmable LED for standard pagan holiday arboreal displays, our final product can be used with anything in mind. Some other uses include displaying the programmable LED lights on stairs, desks, bed frames, hangers, etc.

1.6 Assumptions and Limitations

Assumptions:

- User will only have one controller per house
- The user has a working WiFi connection
- User owns a smartphone with video functionality

Limitations:

- Keep cost at a minimum for components
- Have to use specific string of lights that are either individually addressable or use PWM

1.7 Expected End Product and Deliverables

Mobile Application - An Android Application for the user to perform set up. The user takes video from different points around the display, and the app will process the video to determine the location of each LED on the string. This will create a 3d representation of the display for the user, and they will be able to select a pattern to displayed. This information will be sent to a web server on the Raspberry Pi.

LED lights w/ Raspberry Pi Controller -The Raspberry Pi will have two modes: set-up and display. For set-up, the Raspberry Pi will send PWM signals to the string of lights so as to only have a certain amount lit up for a set of frames in the video that the Android app is capturing. After the location of each LED has been found and the user has selected a pattern, the Raspberry Pi will send the PWM signal for that pattern.

Web Server/Application - The Raspberry Pi will also host a web server and application for interaction with the user and communication between the different components of the project. The server will take inputs from the mobile application such as the light positions, as well as the byte stream of the color pattern from the mobile and web applications. The server will also store the current byte stream and state for easy access from the web app, mobile app, or PWM controller.

2 Specifications and Analysis

2.1 Proposed Design

Our design will be split into three component areas: hardware, mobile application, and the web app/server.

Hardware:

For the hardware, we will use a raspberry pi as a controller. This will be used to receive data in the form of images sent from a camera and then extract data to be sent to our webapp. For communication, we will use wifi to connect our different devices and app. We will use PWM addressable LED light strings for the type of lights and our controller will dictate which lights to turn on and when. A power source will be provided to power the controller and lights. We will include fuses in our design to ensure safety. The circuitry

and controller will be kept in a box with the back side left covered with a grate for ventilation. The box will be covered in wrapping paper to look like a present.

The lights are strung up on a tree and powered through a 30V/12A supply, and connected to an Arduino at the moment for testing purposes. It is possible to select the pattern and brightness and send this to the lights.

Mobile Application:

We will use a smartphone camera to obtain data about the positions of each LED on display. The camera will have a user interface in the form of a triangle in the center of the screen. The triangle will line up with the christmas tree. 4 images will need be captured at different angles. The data will be sent to the controller and the position on the LEDs at each angle will be compared to create the final model to be sent to the web app. This has been discussed and a basic app has been pushed to the gitlab, but more work needs to be done. An algorithm has been developed for detecting LED positions but still needs to be implemented.

The algorithm works by using the android camera library to take pictures of the tree, one with all of the LEDs on, and one with all off. The android application's picture method returns the image in YUV array format, and we will use the Y information, which is the luminance of the image. We subtract the Y values from the unlit image from the Y values in the lit image and clamp it, such that no values go negative. Then we search through the array and find areas of significant brightness; these translate to the x and y coordinates of our LEDs. We then send a signal to light up one specific LED at a time and take a picture. We do the same process of subtracting the Y values from the unlit image and then iterate through and find the area of brightness in the image of the one LED; if that matches up with one of the x,y positions from the first portion, we have found an LED location.

Lit LEDs(left), unlit LEDs(middle), and lit image w/ identification(right)

Web App/Server:

The web server, which is an Apache HTTP server, running on the Raspberry Pi will receive information from the mobile application about the location of the lights, and the web application on the Pi will provide a user interface for the user to select a pattern/make their own.

The server will receive the communication from the app and store it in a text file that will be read by the PWM controller depending on the state of the lights. If the lights are currently illuminated, there will be a semaphore/mutex lock preserving the state of the lights, so the byte stream will be stored in a file that is not currently being read or written to.

The web application will allow the user to create custom patterns and save them directly into a file that can also be read by the PWM controller in the same way as mentioned in the previous paragraph. The web application will also allow the user to load pre-built patterns onto the lights.

2.2 Design Analysis

Hardware:

The Raspberry Pi fits our needs by providing a good means of IPC between the mobile application, the web application, and the web server. An Arduino or an equivalent device would not have been enough, as it was deemed easiest to communicate over internet, and that was what our client has specified as their preferred mode of communication.

A major drawback is that the Raspberry Pi only outputs at 3.3V, whereas the lights need to be driven by 5V. However, as this is a common issue that others have run into using the Raspberry Pi, it was easy to find a 3.3V to 5V level shifter; it is an additional cost(relatively small) and another possible point of failure for our product.

Mobile Application:

The algorithm for the detection of the lights is strong in that it will work in most lighting conditions and tree types, and it is relatively simple and straightforward and doesn't require any recursion or anything similar. When looking at the runtime of the algorithm, it could easily take a couple of minutes to go through the entire calibration process. One major consumer of time will be the communication between the app and the web server on the Pi, as the app will have to request a specific configuration of the lights before each picture and wait for a signal back that the lights are displaying the pattern.

The arrays of YUV data returned by the Android take picture method will be of length width * height of the resolution of the image, so we will have to find the ideal resolution that brings the size of this array down while also maintaining the quality of image to distinguish the LED locations. Each step of the algorithm requires iterating through each array, so our minimum, best case runtime will be number of LEDs * (width * height) of the resolution of the image + (width * height) (for the initial lit/unlit).

Web App/Server:

A primary strength of hosting the configuration and image creation on the web application is the ease of use for the user. The user will be able to connect to the app via web browser, which will allow the user to easily modify or change existing light patterns and colors as well as choose which one to load onto the tree.

The web server will be hosted on the Raspberry Pi, and will be running Apache. Both the Android application and Apache server can communicate using PHP, GET and POST requests. The web server will receive the byte stream of the LED positions from the android application, and then the user will be able to choose their patterns and colors.

3 Testing and Implementation

Our design can be broken into three functional areas: the circuit, the camera, and the web app. We will need to test all three areas as well as the communication channels between each area.

Hardware/Circuit Tests:

These tests will revolve around making sure the hardware is connected properly and safely. Tests will be conducted so as to make sure that fuses are blown correctly when a threshold current is crossed. Current needs to be measured across components so to make sure they fall within component ratings. This is to ensure that components do not get burned out. Lastly tests in this category will revolve around our controller and whether information can be processed and calculations run correctly.

Mobile Application Tests:

These tests will be conducted so as to make sure that the camera interface we utilize is able to function properly and correctly obtain data. Initial tests will involve obtaining the the types of lights that the camera can detect as well as the environment light intensity that yields the best results. From there, testing will be conducted to see if the camera user interface works correctly.

Web App Tests:

The web app tests will include testing if the web app receives the data from the mobile application, and is also able to send the data to the hardware.

Cross Communications Testing:

These tests are to ensure that the system is able to communicate and transfer data correctly.

Camera to Hardware: Test to determine if data from the camera is received by the controller

Hardware to WebApp: Test to see if the controller can send data to the webapp

WebApp to Hardware: Test to see if the hardware can process data from the webapp to change the light designs.

3.1 Interface Specifications

In our design, there are multiple areas in which our different devices will have to communicate with one another. The mobile application will have to be able to send data to the web application, which then will have to be sent to the hardware, which will power the LED lights. The web application will also have the ability to create a custom image to be displayed on the tree or load a pre-built image for display. The web app will then send the byte-stream to the server for use.

3.2 Hardware and software

The hardware/software we will be testing include the hardware/circuit tests, the mobile application tests, and web application tests.

The mobile application tests will include taking a picture from the camera and confirming if the application can pick up the LED lights correctly. Next, confirming that the user can select LEDs and choose colors/patterns for them.

The web application tests will include receiving the data from the mobile application and then sending it to the hardware. The first check will be to see if the data is accurate. The second test will be to confirm if the data is sent to the hardware.

The hardware tests include receiving the data from the web application and then converting it into what the user sees on the tree. Best case scenario is exactly what the user picked out on the mobile application (colors/patterns).

3.3 Functional Testing

The first major step in testing our project is to make sure that the lights light up appropriately with pre-determined patterns. We have used an Arduino Esplora and the FastLED library to ensure that each light is of the appropriate brightness and that the power supply we are using supplies enough power so that there is no drop off of current to lights at the end of the strand.

In addition to testing the that the LEDs light properly, it is important that we ensure the safety of the consumer. We will do this by ensuring that the lights cannot flash at certain frequencies that have been found to be problematic in the potential cause of seizures. We will utilize the IEEE standard for LED lighting flicker and potential health concerns, to ensure that we avoid such frequencies and protect our consumers from such harm.

After acquiring the Raspberry Pi 3 we needed to create a voltage level shifter circuit in order to step the 3.3V output from the Pi to a 5V input signal for the LEDs. In order to test this circuit, we are using a white box test method. This is because we are aware of all the

components being used in this circuit and want to ensure that the level shifter circuit is working and behaving correctly before connecting it to our hardware (Pi and LEDs).

For testing the detection of the lights via the Android app, we have a program that allows us to subtract an image of the unlit tree from the lit tree, and then searches through that grayscale image for pixels with a brightness above a certain threshold, usually 300. Once it finds them, it highlights those pixels with a red point.

The mobile application will be tested using jUnit tests. jUnit is common for java applications and allow us to easily find bugs and fix them. jUnit is already built into Android Studio, so the group will have no problem using it. Testing the application on its own will consist of the user navigating through the different parts of the application on several android devices.

In regards to further testing of our Android app we are implementing regression testing to ensure that the app remains functional as we add different features. This is an appropriate testing method for us because frequent testing at each update will help to limit potential issues that may arise as well as ensure that previously working components of the app will remain functional.

3.4 Non-Functional Testing

Some of the big non-functional testing that will have to occur within our product will be usability. The goal for the application is to be very user friendly so that your average consumer understand how to operate the product. This also allows better performance and data acquired making the whole process easier.

Testing of the selected enclosure will consist of testing to ensure it is waterproof and in-turn protect the hardware from the winter elements. In addition to testing the seal on the enclosure, it is important to test that the hardware will not overheat while in the enclosure.

3.5 Modeling and Simulation

At the current stage of our project we do not have an modeling and simulation aspect of our project. In 492 we may have the ability to model or simulate our android application when it has the basics of our algorithm complete.

3.6 Process

	Present Project Version	Future Project Versions
Android Application	Uses Camera2 API to passthrough feed from camera to display inside application. This API allows	Successfully utilizes camera to calibrate LEDs through image subtraction of unlit tree from lit tree. Connects

	us to obtain camera settings and lock them into place to ensure minimal light disturbance.	to web server/Pi to send bitstream to generate PWM wave to drive lights, receives confirmation that PWM wave was successfully generated.
Raspberry Pi	Web server successfully setup on Pi, static IP obtained and able to accept post requests to php scripts.	Successfully receives PWM bitstream for light pattern, generates PWM wave to drive lights and provides confirmation to application.
Level Shifter Circuit	We have acquired the level shifter chip and designed a circuit to step 3.3V to 5V.	Successfully steps 3.3V output from Pi to 5V input for LEDs.
LEDs	Successfully displays pre-set patterns from Arduino FastLED library.	Successfully displays proper patterns on tree.

3.7 Implementation Issues and Challenges

Challenges that the project will run into later in testing will be recording the image consistently. As we will not know the users tree and lighting settings the goal of our android application will be to have a universal usability. So some of the challenges will be getting the same image that we can analyze so that we can get accurate data. This challenge will be worked on later in the project when we change the settings of our testing. Moving the tree, changing camera angles, or changing lighting could be different ways to test these challenges.

3.8 Results

We have not done any testing yet (besides ensuring the LED strings work), so there are no results available at this time.

4 Closing Material

4.1 Conclusion

3 LED strands have been connected and strung up on a tree. They have been connected to a power supply and an Arduino to test to make sure the LEDs have the desired output. An algorithm has been designed to detect the locations of the LEDs given two images (lit and

unlit) and to find each individual one after that. Equations have been written to convert 2D coordinates into cylindrical.

The goal is to have a mobile application that can detect the location of the lights in a 3D space. The user will then be able to select a pattern/create their own and have that displayed on the lights through the mobile application communicating with the controller.

At this point, our plan is to begin working on the LED detection in the mobile application and driving the lights through a Raspberry Pi, as well as inter-device communication between the two. This will provide the base for the more complex parts that will still need to be implemented (e.g. matching pattern to location of LEDs, user creating their own pattern).

4.2 References

[1]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.

[2]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

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

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

[5] Jeremy Garff, “Raspberry Pi has a PWM module,” 2016, GitHub repository, https://github.com/jgarff/rpi_ws281x.wiki.git

[6] Daniel Garcia & Mark Kriegsman, “FastLED library,” October 2014, GitHub repository, https://github.com/FastLED/FastLED/wiki/Overview

4.3 Appendices