Holiday Arboreal Lights Project

sddec18-10

About Us

Rajiv Bhoopala
Computer Engineer
Android Dev/Web Server

Michael Scholl
Computer Engineer
Image Processing

Robert Tyynismaa
Computer Engineer
Android Dev

Aaron Hudson
Computer Engineer
RPi Systems Dev

Justin Falat
Computer Engineer
Web App/Server Dev

Client and Advisor: Dr. Thomas Daniels

Problem Statement

- Many people decorate their homes with a set of lights
- However current holiday lights are limited by customizability
- Currently, to decorate an arboreal display we must first visualize the display and lay the lights accordingly

Our Solution

- The idea for our solution is for users to set up RGB LED lights on a tree and then upload patterns to the string of lights
- An android application will send a picture of the RGB LED lights to web server
  - The web server will create a model of the LEDs
- Raspberry Pi will then power the lights
- User will be able to select many different colors and patterns

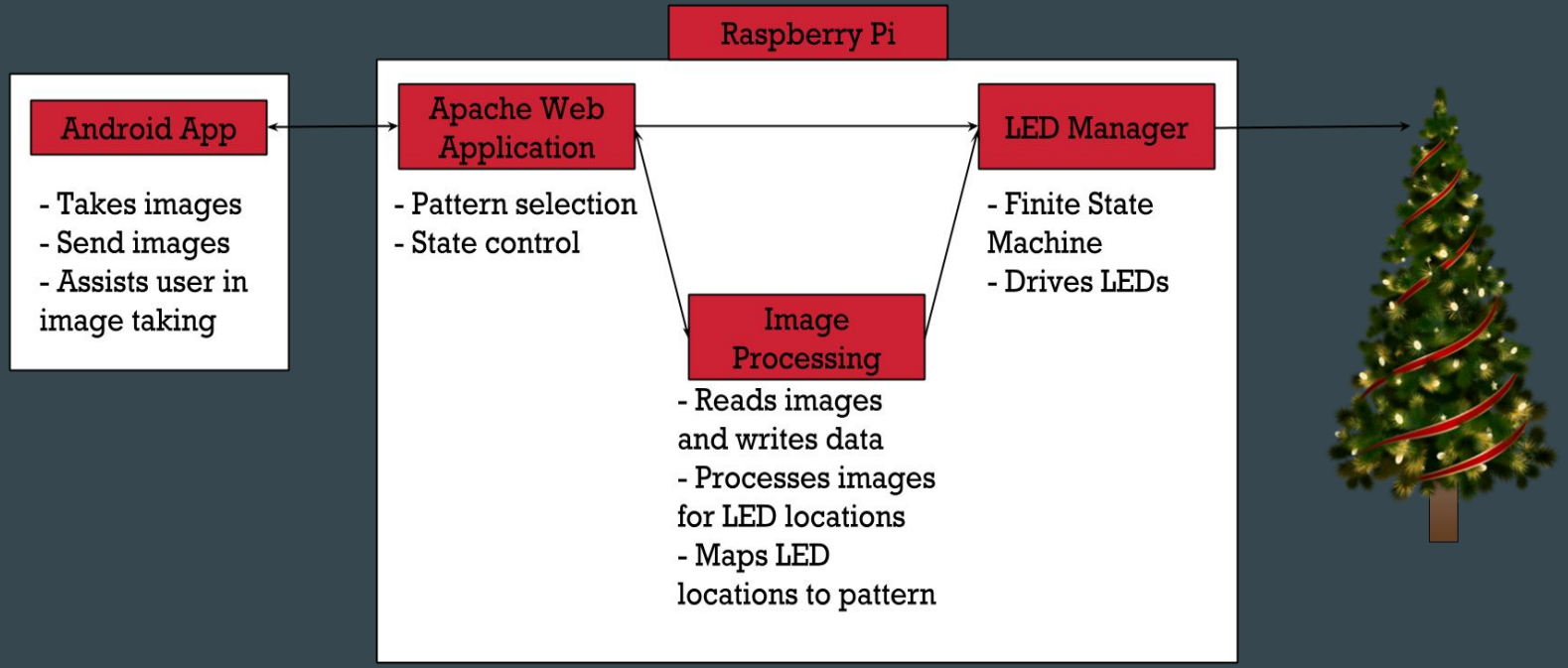
Functional Requirements

- Raspberry Pi PWM controller sends RGB values to LED
- Android App takes pictures for calibration process and sends them to the Web App
- Web App selects what patterns to display on the tree
- Web App controls state of LED Manager by creating and deleting .lck files
- LED Manager drives lights

Non-Functional Requirements and Constraints

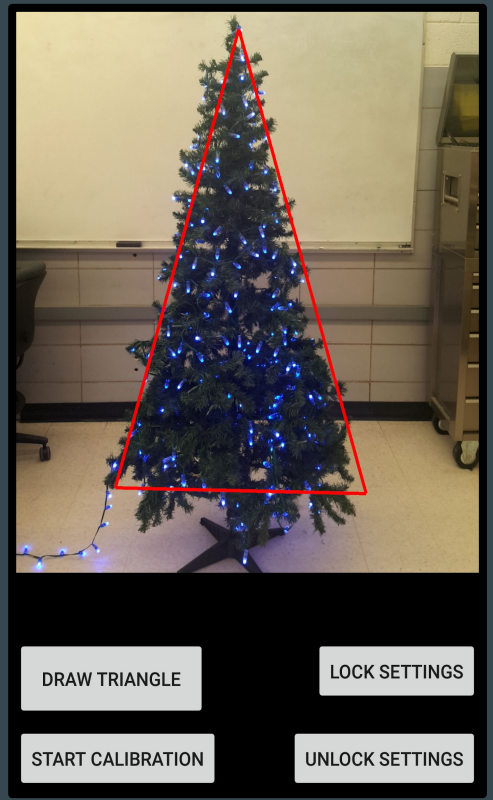
- Android App must be responsive and easy to use
- Calibration process must be energy efficient to limit battery use
- Web App must control state of LED Manager while maintaining mutual exclusion
- System must be able to run for long periods of time without fail
- Raspberry Pi 3B processor and storage limitations
- Android only mobile application
- Android App and Web App must communicate via WiFi

Block Diagram



Android Application

- Communicate with Web Server
  - Start calibration process
  - Communicates LED index for calibration
  - Send images
- In-App functions
  - Take and save pictures
  - Lock and Unlock Settings
  - Draw app-side triangle

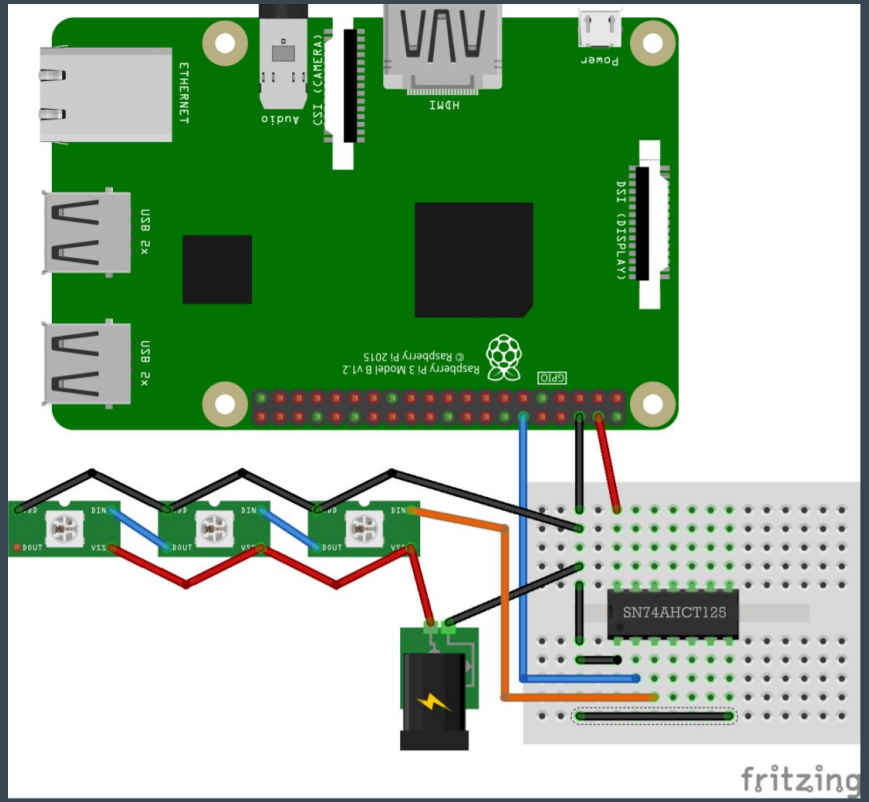


Android Development

- Started off with the Camera2Basic sample provided by Google
  - Basics of Camera2 API
- Added functions to lock and unlock camera settings
- Tried different methods for image capture
  - CountdownTimer
  - Video Intent/Camera2Video
  - HTTP handshake with Web Server
- Unit and Integration Testing

Raspberry Pi Overview

- Three subsystems
  - Web App
  - Image Processing
  - LED Manager
- LED control and power
  - Data: 3.3V out to 5V in
    - Level shifter
    - GPIO 18 PWM
  - Power: 12V30A supply to LEDs



Web Application and Server

- Main functionality of Web Application
  - Pattern Selection
  - Display project information
    - Team/Project info, brief component break down
- Hosted locally using Apache2 web server on the Raspberry Pi
  - Web Application built with HTML, CSS, JS for Web App content and functionality
- Server hosts PHP scripts for:
  - Pattern selection and .lck file manipulation
  - Acceptance of files from Android Application
  - Calibration process

Web Application - Pattern Selection

- Allows the user to choose from a set of preset patterns to display on the tree
- Once the pattern is selected, the Web Application creates and deletes .lck files, controlling the state of the LED Manager
  - Updates the pattern to display while maintaining mutual exclusion of the pattern file

LED Manager

- Finite State Machine
  - Idle, Running, Calibration
  - Polls .lck files
- Drives WS2811 LEDs
  - RPi WS281x Library
  - PWM via GPIO
- Reads pattern input from .txt file
  - Assigns LED index to RGB value
  - Refreshes LED state

LED Manager Development

- Replace FastLED C library
  - Arduino Uno for initial testing
  - Raspberry Pi Model 3B for final implementation
- Using RPi WS281x Library
  - C, Go, Python library for driving WS281x LEDs (shift registers)
  - Creates LED object, updates indices, refreshes
- Need to control LEDs and tree state
  - Finite State Machine
  - Sockets vs lock files vs watchgod for state control
  - Settled on lock files using os.open() functions

LED Manager Testing

- Unit testing paired with Web App
  - Lock file state changes
    - Proper state sequence when updating patterns
  - Individual state duties
    - Polling for changes, driving lights
- Acceptance testing
  - LED colors
    - What the colors should look like
  - Refresh rate of tree
    - Too slow or too fast

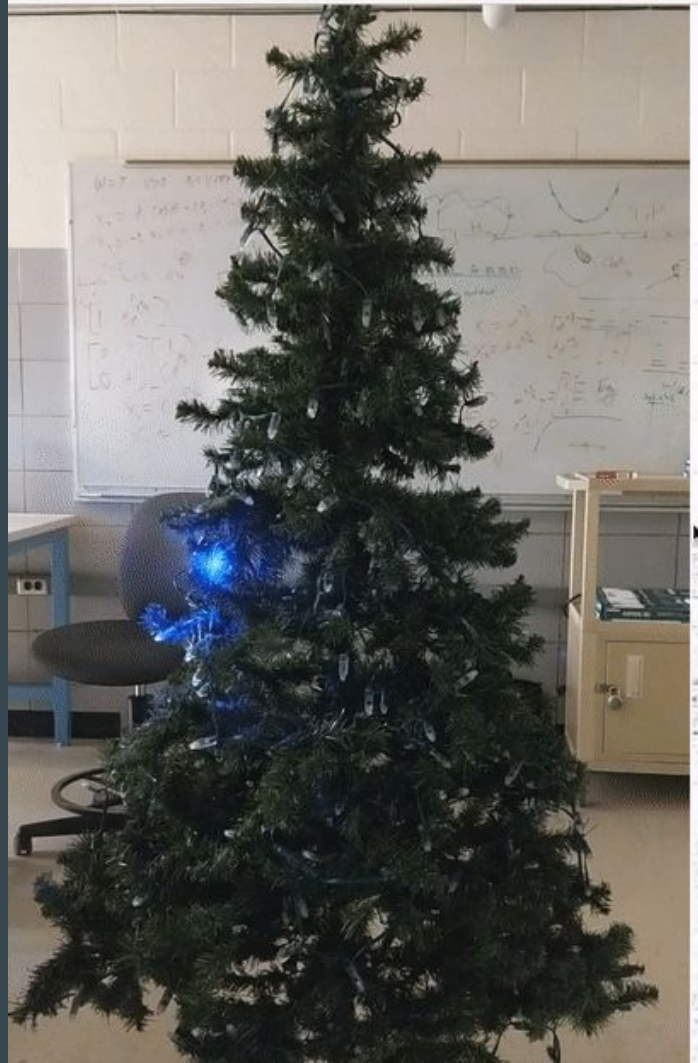


Image Processing

- Image subtraction
  - Scikit-image
  - Comparison value
- Blob detection
  - OpenCV
  - Parameter calibration
  - LED location
- Common errors
  - Light pollution
  - Multiple LED's

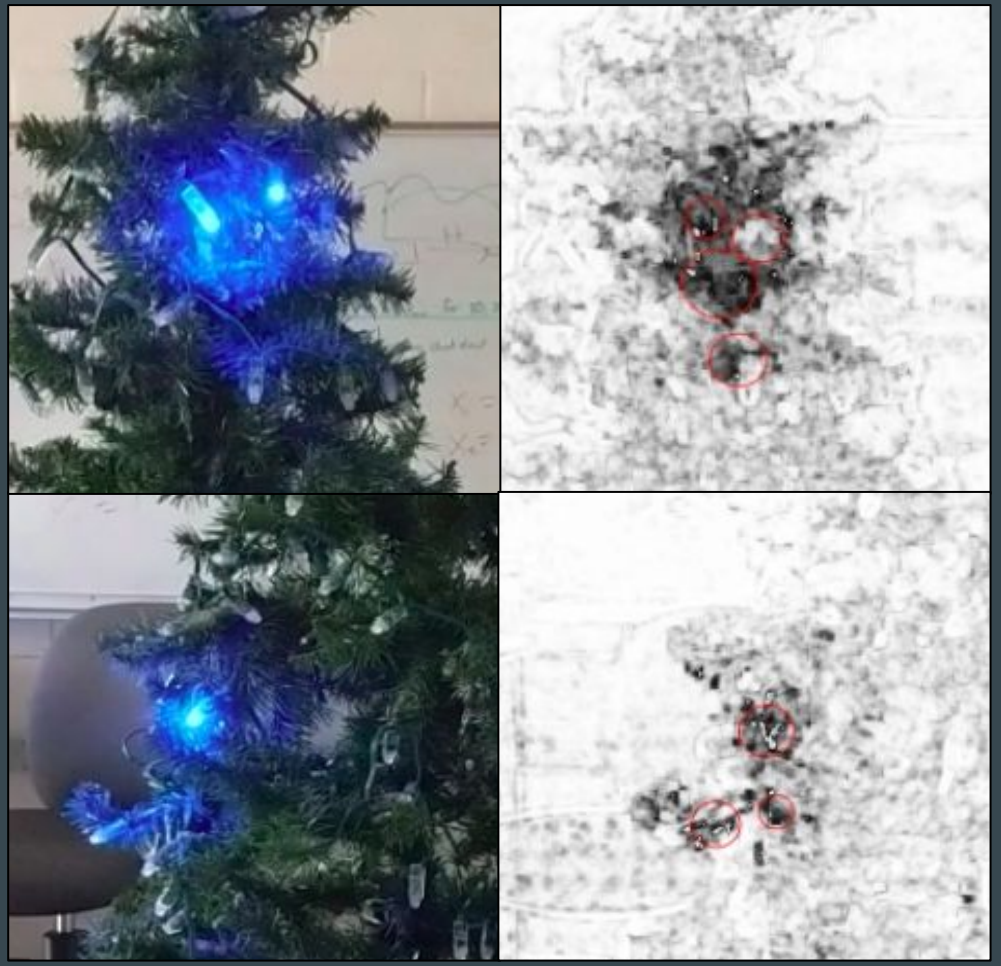
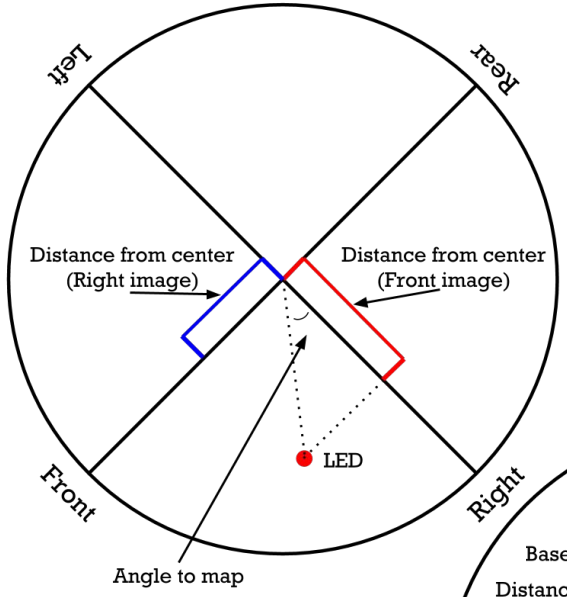
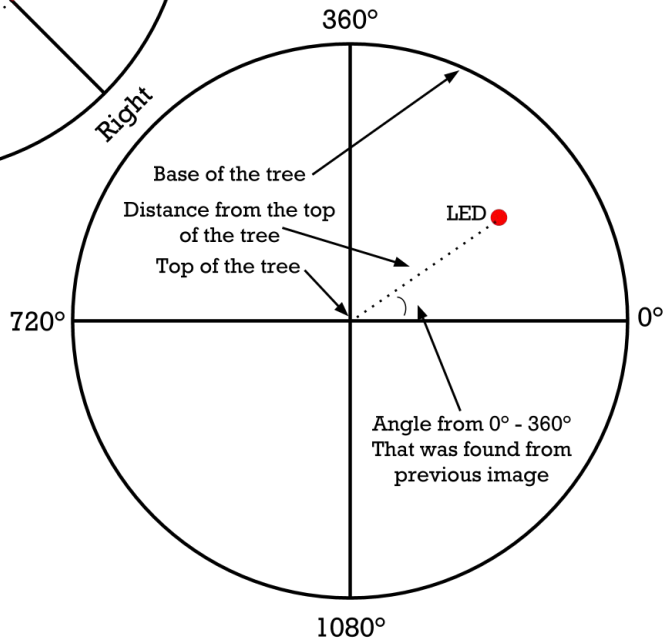


Image Processing

- Top Image
  - Using front/right photo
  - Distance from center
  - Angle taken from R/L axis
- Bottom Image
  - Angle conversion
  - Data storage
- Custom Patterns
  - Image extraction





Future Work

- Improving image analysis accuracy
  - Unlit image for each lit LED image
- Better estimate the LED locations we are unable to find
- Implement states in the Android App
  - Display current progress of calibration
- Other quality of life improvements to Android App
  - Delete images immediately upon send to free up space
- In-depth analysis of power usage
- Support for more advanced patterns in Web App and LED Manager
  - Custom patterns from uploaded images, manually created patterns

Q&A

References

RPi WS281x - https://github.com/rpi-ws281x/rpi-ws281x-python

watchgod - https://pypi.org/project/watchgod/