ITS LIT: An Interactive Light Display

Patrick Browne, EE, Emma Bryce, EE, Varun Menon, EE, Mike Polin, CSE, and Tommy Zhen, CSE

***Abstract*—Current LED displays and televisions are designed for viewing at shorter distances. To achieve discernible imagery with this technology, larger resolution and overall size is necessary. This project aims to create an interactive light display that is designed for viewing from further distances without the need for increasing the size of the overall display. The goal was also to incorporate campus interaction through the use of our iOS app in which people walking by can download and use to interact with the display.**

I. Introduction

Traditionally, television screens and displays were created to be viewed from distances suitable for living rooms and households. This poses a problem if someone wanted to put up a display whose purpose was to be seen from distances that far surpass the usual living space. With televisions, the only solution would be to increase the size of the overall screen and resolution, but at a certain point the size becomes cumbersome for some applications [1]. One of the main goals of our project is to design a light display that can reach viewership from a distance not achievable by a television of the same size with similar effect as represented by Figure 1.

Figure 1: Representation of Effect Desired

Most people, when tasked with creating displays for large audiences, obtain larger screens instead of looking at other aspects of the television. There are small LED boards that have modified resolution and size that allow for an extended viewing range, but those are usually for small scale applications [2]. The problem as a whole has not changed over time, as the simple solution has always been to increase the size of the television to achieve the ultimate goal. We as a team also want to promote campus interaction through the use of our light display. Small scale LED boards like that would not fulfill our need in that regards either as we are trying to modify the resolution through our own design.

Through our initial analysis of the problem and the goal of our design, we have come up with several guidelines and specifications that our light display will need to fulfill as outlined by Table 1. Our light display aims to solve the problems discussed previously, with similar power consumption needs to a standard television.

TABLE I
General Specifications

Specification	Value
Viewable Distance	~60m
Power Consumption	<400W
Visibility by number of people	~100s

II. Design

A. Overview

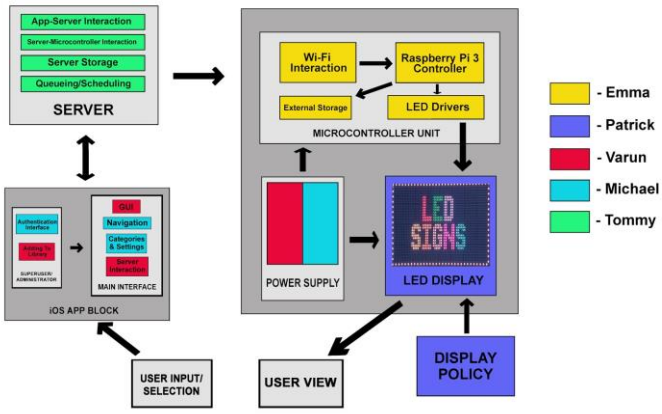


Figure 2: Block Diagram

Our overall design revolved around creating our own LED board with resolution designed to optimize viewing from further distances than a television can allow. The LEDs we based our entire design off of are the WS2812 RGB LEDs [3]. These were chosen for their cost-efficiency as well as their low power consumption for what we need them to accomplish. Alternative technologies we considered were LED strips and

P. Browne from New Jersey (e-mail: pbrowne@umass.edu).
E. Bryce from Massachusetts (e-mail: ebryce@umass.edu).
V. Menon from California (e-mail: vmenon@umass.edu).
M. Polin, from Massachusetts (e-mail: mpolin@umass.edu).
T. Zhen from Massachusetts (e-mail: tzhen@umass.edu).

flood lights, but ultimately we chose the WS2812 RGB LEDs, shown in Figure 3. The LED strips were ruled out because they were of fixed length and did not allow us to change the distance between each LED, also known as pixel pitch, thus limiting our resolution to a preset value [4]. The flood lights were ruled out also due to resolution difficulties because of their size and wide field of coverage.

Figure 3: WS2812 RGB LEDs

The other sub-systems integrate to determine what is sent to the display and how it is done as well a block that governs what is to be shown eventually on our display. The microcontroller we have chosen to help drive our display is the Raspberry Pi. The Raspberry Pi in our final project iteration acts both as the server and the display driver, which receives requests via user input from the iOS app and will update the display accordingly.

B. Block 1:LED Display

This will display an image controlled by a microcontroller. For our display we ended up going with a pixel pitch of 1 inch or 25cm down from our original estimate of 30cm. This decision was made so that all of the PCB's would be uniform when put together. Each LED on the outside was half an inch from the border of the PCB, so that when two PCB's were put side by side the pitch would remain an inch. By decreasing the pitch we were able to increase the number of LED's for the display. Each PCB was able to house 144 LED's with the total display housing 1296 LED's. With the additional LED's the total power consumed by the whole system if each LED was displaying white jumped up to 388.8 Watts. The PCB's were designed using Altium, the schematic of which is shown in Figure 4, and were fabricated by EasyEDA. When the PCB's arrived, they were brought to Worthington Assembly Inc. who helped to populate the boards for us. The PCB's were mounted on a wooden display.

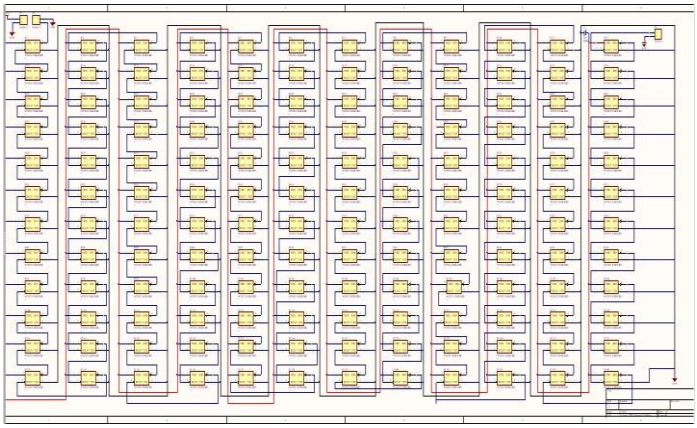


Figure 4: Schematic of PCB Design

The display we ended up using for FPR and demo day was a display of LED strips that contained 900 LED's. We discovered that there were bad LED's that were corrupting the stream of data, because each LED was connected serially which was causing the problem. Below in Figure 5 is an image of the display with the PCBs and one can clearly tell when the signal becomes distorted. We were not able to find which LED's were causing the problem due to timing constraints and debugging issues and as a result is why we ended up using our backup display as shown in Figure 6.

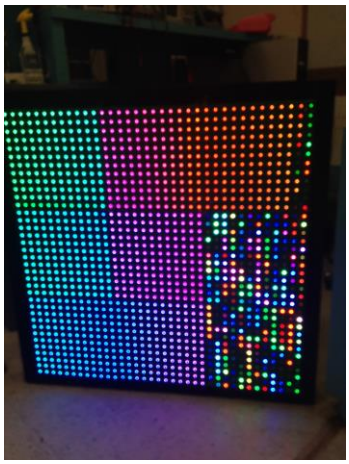


Figure 5: PCB Display with Distorted Signal

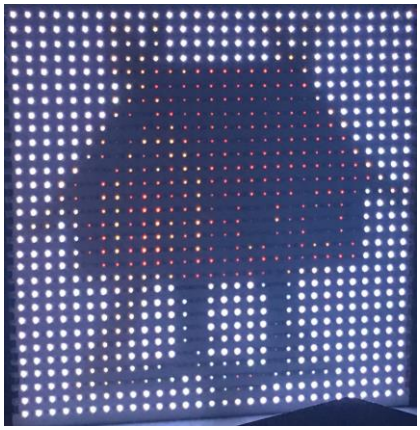


Figure 6: 900 Pixel Backup Display

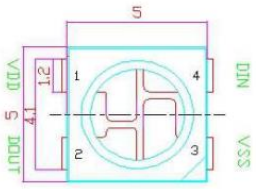
The display composed of PCBs is powered by two Aiposen 5V/60A/300W LED Power Supplies and wired up via 12-gauge THHN copper. Meanwhile, our backup display uses one of these power supplies. Each power supply converts 120/220V AC to 5V DC. Each power supply also is capable of handling a maximum current of 60A. The Raspberry Pi microcontroller is powered by a USB port on power strip that is used to plug all of the power supplies together.

Figure 7: Power Supply

C. Block 2: Microcontroller and Peripherals

The Raspberry Pi 2 [6] computer will control the addressable RGB neopixels in the display. The Pi communicates with the neopixels using their specific timing protocol to set colors on all of the lights in a neopixel display to generate a static image. The image that the Raspberry Pi helps render comes from the server. It downsamples, crops, and displays images of varying sizes and file types. The program in which the image is downsampled into the 900 pixels necessary and driven on to the display, is written in python and accesses the images from a folder on the Raspberry Pi's main directory.

PIN configuration



PIN function

NO.	Symbol	Function description
1	VDD	Power supply LED
2	DOUT	Control data signal output
3	VSS	Ground
4	DIN	Control data signal input

Figure 8: WS2812 Pin Configuration

The program controlling server communication and image processing and display is written in Python. For our final project, we had combined the server/queue and display code onto one Raspberry Pi to help streamline the flow of data between the iOS app and the display. This way we could use the Raspberry Pi to act as the server as well.

D. Block 3: iOS App

For our user interface for our senior design project we created an app using the iOS operating system. The purpose of the app is to connect the user to our programmable display by communicating with the server to send information packets to update the display. To create an app for the iOS operating system one is required to use an IDE called xcode which uses an objected oriented coding language called Swift [7]. For our project's final iteration, the iOS app is as shown in Figure 9. We have an initial login screen for general users and on the second page we included categories for users to choose from. The general public access library consists of several image categories, from which users can select a specific category of their choice. Once in a selected category, users can select from a host of preset images to be displayed. The app interacts with the server with HTTP POST requests, where each image on the app corresponds to an image on the server. When a user picks an image on the app, the name of the image is sent to the server which then puts it in the queue. Our iOS app also includes a super-user portion, where if the super-user submits a link to a picture, it can be sent to the display in real-time. For our final app, we were able to establish the communication with the server and send the correct data to be put on the queue.

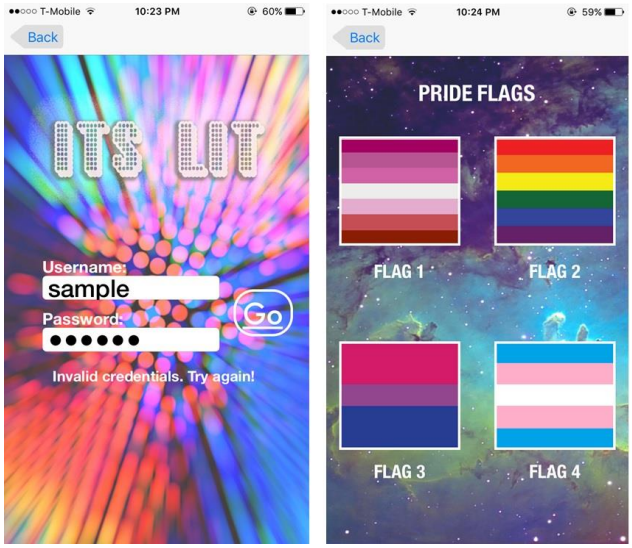


Figure 9: iOS App GUI

E. Block 4: Server

The purpose of the server portion of the project is to store images and help facilitate commands from the app to the Raspberry Pi as well as solve queuing requests from multiple user inputs. The server stores image presets instead of the iOS app in order to save space on the app itself allowing it to operate more smoothly and streamlined.

The server being used is an Apache web server [8], which communicates over the TCP/IP protocol. The server is HTTP based, which will allow for the iOS app to send HTTP requests over the network to the server. In order to communicate with

the server, the IP address of the hosting computer is needed [8].

The server receives requests from the iOS app through a POST request which is accepted via a PHP script on the Raspberry Pi and stores the input string on a MySQL database. The queuing algorithm, which is written in python, continuously checks this database for new requests in order to update the display with. The database and queue handle requests in a FIFO based manner and entries are deleted from the database after the corresponding image is sent to the LED display.

F. Block 5: Display Policy

This was going to be our policy on how we decide which images will be eligible to be displayed. Firstly we needed to ensure that the image can be displayed clearly on the display. We also will not be showing any images that contain profanities. Lastly, the criteria were that we find images that are compelling and will be used daily by the students interacting with the display. The original plan was to have it deployed inside of a building with high visibility to a large gathering of students or in an area with high traffic. As a group we discussed the student union because there is a lot of foot traffic and is still used at night which is when our display will be seen clearly.

We ended up meeting with Professor Caroline Aragon in the spring. She gave us the contact for the head of the university's Public Art Committee. We wanted to make sure everything worked before pursuing an audience with the committee. However, seeing that our PCB display did not properly function, we were not able to fully commit to a pursuing this course of action to deploy the final project.

III. Project Management

As a group, we were able to complete all of our FPR deliverables that we proposed at CDR to our advisors, Professor Krishna and Professor Fraiser. The table below outlines what we had promised at CDR. For FPR, we planned to show that we had a working display as a backup while we worked on the main PCB display. For the demo portion of FPR, we were able to show to our advisors the full functionality of our subsystems from user input to the display updating in real-time.

TABLE I
FPR Deliverables

Deliverable	Status
Display and integration with power supply and Pi	Complete
Display able to receive requests from queue and server	Complete
Advanced app GUI, layout, and user input tested	Complete
Display frame created	Complete

We all work well together as a team and each of us brings different skills to the project. Emma and Patrick are both EEs with similar backgrounds in regards to circuit design. They are working together on the design of the display and driving it with the Raspberry Pi. Varun is also an EE with experience in hardware, but he also has experience with software and is working alongside Mike, who is a CSE with coding experience, on the iOS app. Tommy is also a CSE with more experience in programming, and he is working on the server and integrating that with the other sub-systems.

As a team, we created the list of proposed deliverables for FPR and then continued on our work from CDR with everyone working on their subsystem. As we did with MDR, we had people designated as the lead for each subsystem, while also working together as a team to integrate them. We communicate to each other when we need help and we work together to help that team member solve that problem. Adaptation and communication are key elements to working in a team and we understand it is necessary to the success of our project. As a team we communicate via group chat outside of our weekly team and advisor meetings with Professor Mclaughlin. At our advisor meetings, we discuss our plan for the next week as well as address any questions anyone may have.

IV. Conclusion

After completing the goals of our FPR presentation to Professor Krishna and Professor Fraiser, our final project iteration involved use of our backup display. Our goal was to use the display comprised of the designed PCBs, but due to deadline constraints and debugging issues we could not do so. We were, however, able to demonstrate the full functionality of each subsystem. When a user chooses an image through our iOS app, the corresponding POST request is sent to the server which stores interprets it and stores it in the database until the display is ready to update, at which point the system waits for new input from the user via the app.

APPENDIX

A. Cost

Below is the analysis of the cost of our project with the PCBs. The table outlined below does not reflect the cost required to have the boards populated by Worthington Assembly Inc. and is purely the costs for all of the materials.

Part	Development	Production (1000)
Raspberry Pi	$40	$40
Power Supply	$105	$105
WS2812 LEDs	$145	$100
PCBs	$105	$69.84
Total	$395	$314.84

Acknowledgment

Firstly, we would like to thank our faculty advisor Professor David McLaughlin for his professional and unbiased guidance throughout the semester. We would also like to thank Professor C. Mani Krishna and Professor Stephen Frasier for

their honest opinions and feedback. We would like to thank Professor Christopher Hollot and Francis Caron for the work they do to make this course possible. Lastly, special thanks to Terry Bernard and Jeremy Paradie for their help throughout the year.

REFERENCES

[1] J. Govan, "TV sizes and viewing distance," in Crutchfield, 2016. [Online]. Available: http://www.crutchfield.com/S-eMvermGxthz/learn/learningcenter/home/TV_placement.html.

[2] "Peggy 2," in Evil Mad Scientist. [Online]. Available: http://shop.evilmadscientist.com/productsmenu/tinykitlist/75-peggy2.

[3] "www.i-enet.com - WS2812.pdf,". [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf.

[4] J. Davis, "What is Pixel pitch and why should I care?," www.nanolumens.com, 2011. [Online]. Available: http://www.nanolumens.com/what-is-pixel-pitch-and-why-should-i-care/.

[5] [Online].Available: http://www.seeedstudio.com/document/pdf/WS2812B%20Datasheet.pdf. (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[6] "Raspberrypi2modelb.pdf,". [Online]. Available: https://cdn-shop.adafruit.com/pdfs/raspberrypi2modelb.pdf.

[7] "Start developing iOS Apps (swift): Jump right in,". [Online]. Available: https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/.

[8] "HTTPD - Apache2 web server,". [Online]. Available: https://help.ubuntu.com/lts/serverguide/httpd.html.