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 is 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	<300W
Visibility by number of people	~100s

II. Design

A. Overview

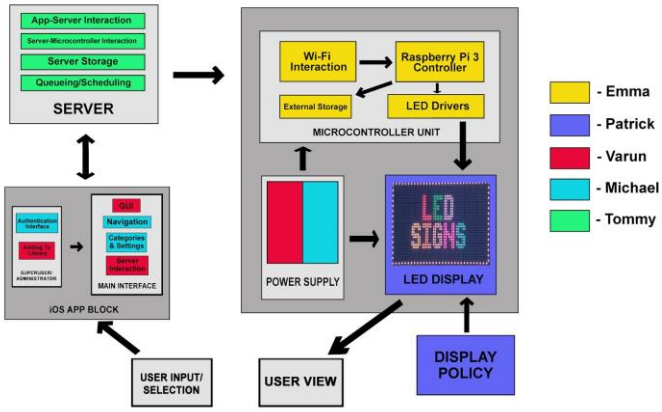


Figure 2: Block Diagram

Our overall design revolves around creating our own LED board with resolution designed to optimize viewing from further distances than a television can allow. The LEDs we will be basing our 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.

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).

Alternative technologies we considered were LED strips and flood lights, but ultimately we chose the WS2812 RGB LEDs, shown in Figure 3. The LED strips were ruled out because they 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.

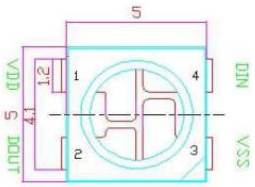
Figure 3: WS2812 RGB LEDs

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 will take input from the server, which initially receives information via user input from the iOS app, and will send that to the display.

B. Block 1:LED Display

This will display an image controlled by a microcontroller. The display will be 1 meter by 1 meter and will be made up of nine 1 foot by 1 foot PCB's. Neopixels are what will be mounted on the PCB's. We needed LEDs that can display a variety of different colors and are individually addressable. This is why we settled on the WS2812 LED which satisfies both of those requirements and also has libraries available for the Raspberry Pi that help the Pi communicate with the LED's. We were planning on having a pixel pitch of 30mm which would mean we would have 900 total LED's. The LED's will be connected serially to each other. Each color will draw 20mA (red, blue, and green) so when an LED is white all the colors are active. So, if each LED is white then they will each be drawing 60mA and with 5V of supply voltage each LED consumes 0.300 Watts [5]. With the previous calculations we can conclude that a display with 900 LED's would consume 270 Watts maximum. In order to handle the high current we will need to use gauge wire and have that wire split off to power the 9 PCB's. The PCB's need to be designed that that each LED is powered and grounded properly and that the data ports are connected to the previous ad next LED.

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 4: WS2812 Pin Configuration

C. Block 2: Microcontroller and Peripherals

The Raspberry Pi 2 [6] computer will control the addressable RGB neopixels in the display. The Pi will communicate 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. We need to understand the serial communication protocol to use, and an understanding of programming is needed to complete this block.

The program controlling server communication and image processing and display will be written in Python. So far, a Raspberry Pi has not been deployed to this task. Python as well as the basic image processing tasks mentioned will need to be learned. The task of controlling neopixels and testing unique light patterns with an open source Arduino library has been achieved using and Arduino Uno.

D. Block 3: iOS App

For our user interface for our senior design project we're creating 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 MDR deliverables, we had a basic graphical user interface created with basic navigation between two pages as shown in Figure 5. On the second page we included choices for images to display with a button associated with it to increment a counter. Our next step is to set it up so that when the button is pressed it will communicate with the server of the user's choice. Most of the programming skills learned from previous object oriented programming classes are relevant to this task but we need to still learn the specific implementations for swift. Specific areas of focus to create an app include server-app interaction, authentication process for advanced users, advanced users updating display library, and Wi-Fi fence to prevent users outside the designated area from interacting with the display. For a fully functional app we need to test to see if

it communicates between the server and if it sends the correct information to update the display accordingly. Our last test will include checking to see if our Wi-Fi fence correctly blocks users from outside the designated distance.

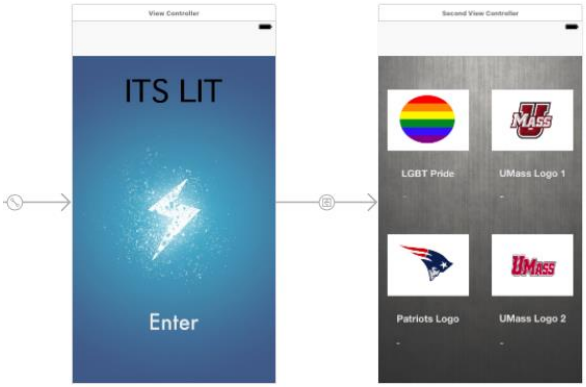


Figure 5: iOS App GUI

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 will store 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].

Currently, we need to research more on how to integrate the server and the iOS app so that they can share data. This will also aid in the incorporation of the server and Raspberry Pi interaction in which the image will be broken down and sent to the display. We also need to research if the implementation of a queuing algorithm on the server is possible.

F. Block 5: Display Policy

This will be our policy on how we decide which images will be eligible to be displayed. Firstly we will need to ensure that the image can be displayed clearly on the display. We also will not be showing any images that contain profanities. Finally we have been attempting to find images that are compelling and will be used daily by the students interacting with the display. We are also working on getting a location for the display. The display will be deployed indoors facing outside. As a group we have 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 have reached out to a faculty member (Caroline Aragon) that our adviser suggested to us to help get a location setup. She agreed that the student union would be a suitable location and we will be having a sit down meeting with her when classes resume in the spring.

III. Project Management

As a group, we were able to complete all of our MDR deliverables that we proposed at PDR to our advisors, Professor Krishna and Professor Fraiser. The table below outlines what we had promised at PDR. For CDR, we plan to have some of the sub-systems integrated together, while the design of the display and its incorporation with the microcontroller will also progress forward. We plan to have working communication between the iOS app and the server so that data and commands can be transferred between the two sub-systems.

TABLE I
MDR Deliverables

Deliverable	Status
Portion of display design tested	Complete
Microcontroller driving display	Complete
App GUI, layout, and user input tested	Complete
Server Created, tested with Raspberry Pi	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 MDR and then assigned lead roles for each sub-system. These lead-roles are not set in stone as we understand that a project of this magnitude requires complete teamwork and effort. 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.

Figure 6: Gantt Chart

IV. Conclusion

Currently we are on track, after meeting the deliverables for our MDR presentation to Professor Krishna and Professor Fraiser. Our goal for CDR is to integrate several of the sub-systems as well as demonstrate progress on the design of the display and its incorporation with our microcontroller, as stated in the previous section. The lead roles for each sub-system will be the same, and as before we will assist each other when necessary.

For CDR, we are planning to have the server and iOS app integrated so that input data from the app can be sent to the app and recognized. We also would like the design of our LED boards completed so that we can get them created in a timely manner. Lastly is to create additional functionalities to the iOS app's GUI. Even more so than MDR, we will be working together as a team on the progression of these goals during this integration phase of the project.

Acknowledgment

First, we would like to thank our faculty advisor Professor Mclaughlin for his professional and unbiased guidance throughout the semester. We would also like to thank Professor Krishna and Professor Fraiser for their honest opinions and feedback. Lastly, we would like to thank Professor Hollot and Francis Caron for the work they do to make this course possible.

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.