Modular LED Wall Panels: Final Report

Team 29: Adam Chung, James Prince, Kyle Salzberg
ECE 445
Professor: Victoria Shao
TA: Hanyin Shao

May 4, 2022

Abstract

The smart home industry has grown tremendously in the past decade and continues to promise sustained growth as more connected devices are introduced to the market. LED decorations have gained immense traction with a variety of offerings from LED strips to modular panels. These products often have limited customizability and limit customers to a linear strip or a simple panel that cannot display text or visual effects. Our LED wall panels aim to fill the gap by providing additional functionality to the users with the ability to display text, images, and various effects. We successfully developed 4 working prototypes that can display text and dynamic effects. While our project was mostly successful, there are additional features to be implemented in future iterations such as dynamically hot-swappable panels that automatically detect their configuration as well as support for images.

Contents

1 Introduction

1.1 Problem

In recent years, LED decorations and the IoT marketplace have grown immensely in popularity. However, many of the commercially available products are either overpriced or provide very little customizability. Currently, most LEDs are only available as strips, with few able to display text or images. The linearity of LED strips hinders the users' ability to create 2-D displays tailored to their specific room dimensions. On the other hand, the lack of text and images on current modular LED panels limits the possibilities for dynamic displays that can provide useful information.

1.2 Solution

To overcome these limitations and solve this problem, we designed and implemented four LED panels that are capable of displaying customizable texts or images. The modular design allows for the user to connect together as many panels as desired, in relatively any shape. For example, with four tiles, the user can create either a 2x2 square, a 4x1 line, or an "L" shape.

The design is focused around a "core panel" containing the main control unit, which can then be connected to "expansion panels" in the manner described above. The control panel determines the configuration of the 4 panels, deciding the manner in which to place the text/images (e.g. text will scroll if too small to fit on one panel). Each panel contains an array of 8x8 serially addressable LEDs which can be illuminated in any color. Utilizing a smartphone, the user can choose what image or dynamic display will be shown on the panels. In our case, the user can choose between a "ball" mode in which a ball bounces around the panels according to their configuration, or a "text" mode where the user can display scrolling text entered from their Bluetooth device.

1.3 Visual Aid

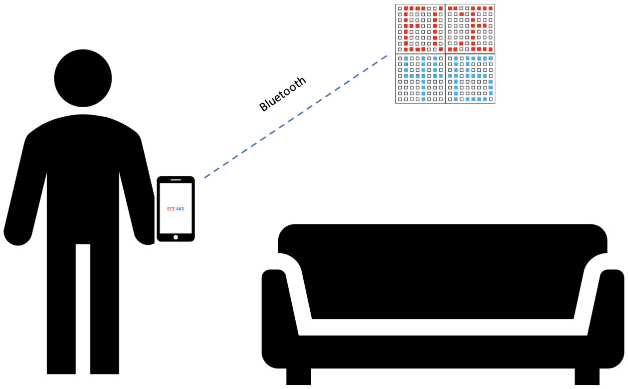


Figure 1: Configuration example 1 of four panels

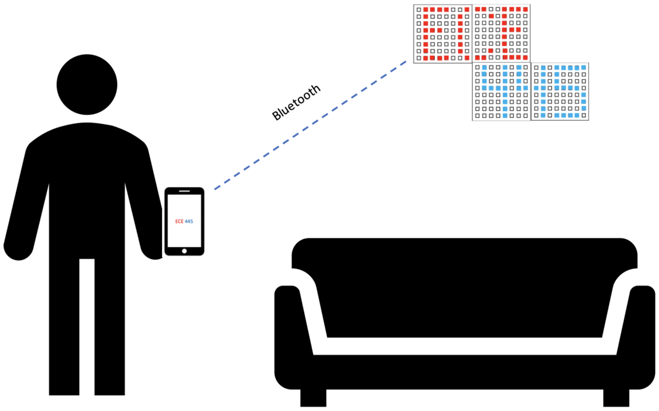


Figure 2: Configuration example 2 of four panels

Figure 1 and figure 2 above is how we envision a real-life version of our product being used. We imagine the users of our product hanging up our LED panels on their wall in any desired configuration. Figure 1 and 2 depict two sample configurations the panels can be configured in.

1.4 High-level Requirements

- The four panels must be able to display text, images, and dynamic effects that can adapt to current configuration. If text is too small to fit within boundaries, it will scroll across panels.
- ”Core panel” must automatically recognize any change to the configuration of expansion tiles, updating the display output to each tile to fit within the new boundaries in under 1 second.
- Panels must be able to be controlled through Bluetooth by a smartphone or other external device (must connect to Bluetooth device within 5-7 seconds).

The three high-level requirements above were our initial high-level requirements for our project. We were able to successfully meet the first and third requirements which will be discussed later in this report. However, we were only able to implement a part of our second requirement. While we were able to configure the boards in any configuration, we had to manually enter in the configuration of the boards instead of the core panel automatically recognizing the configuration. This will also be discussed in more detail later in this report.

2 Design

2.1 Block Diagram

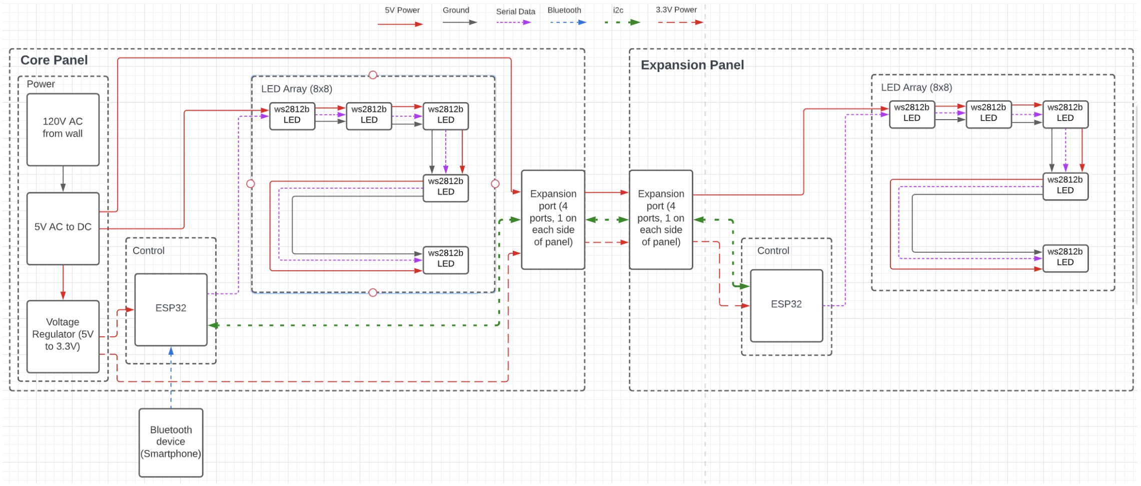


Figure 3: Block Diagram

Figure 3 above depicts the block diagram for our project. It can be seen that the project consists of both a core and expansion panel. The core panel contains the power subsystem which will provide power to all other boards. Additionally, the core panel is where the user will connect their smart device over Bluetooth. Each panel has four expansion ports that transmit the I2C lines (SCL and SDA) as well as power and ground to neighboring panels. The expansion panel is almost identical to the core panel, but without the power subsystem. Lastly, each panel has an 8x8 array of individually addressable LEDs.

2.2 Power

2.2.1 Description

The power subsystem for our project was relatively simple. The subsystem consisted of a power adapter that was plugged into a standard North American outlet which outputs 120 V AC rated at 15 A. Utilizing the power adapter we stepped down the 120 V to a 5 V DC supply. Additionally, we used a LM1117-3.3 voltage regulator to step down the 5V supply to 3.3V. This was necessary because the microcontroller that we used was the ESP32 [1] which runs off of 3.3 V.

2.2.2 Schematic

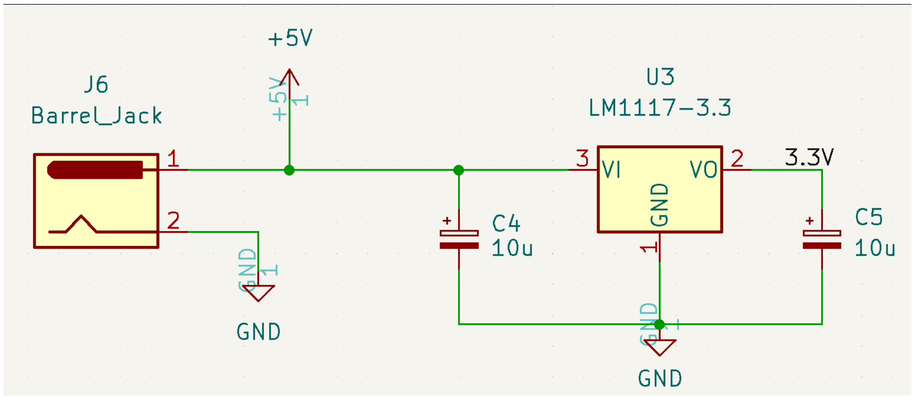


Figure 4: Power Schematic

Figure 4 above depicts the schematic for the power subsystem. As described previously, we can see that it consists of a barrel jack where the power adapter can be plugged into, as well as as the LM1117-3.3 voltage regulator [2]. It should be noted that we chose a power adapter that was rated for 15A due to current requirements for the LEDs.

2.3 Control

2.3.1 Description

For our control subsystem we chose to utilize the ESP32-WROOM-32D. Since one of the main requirements of our project was to have Bluetooth capabilities, the built-in Bluetooth connectivity of the ESP32 was a major deciding factor in why we chose this microcontroller. The Bluetooth capabilities of the ESP32 made it an ideal choice for our project. Additionally, the ESP32 was in stock, and had sufficient memory for our design making it a desirable microcontroller.

Only the ESP32 on the core panel is required to be connected to over Bluetooth. Once connection was established, this ESP32 acted as the "master" in I2C communications, while the ESP32 chips on the expansion panels acted as "slaves". Using the I2C protocol, the core panel sends out sends a series of data packets to each microcontroller. Each expansion panel ESP32 then parses the data and lights up its LEDs accordingly. Since only the core panel utilizes the Bluetooth capabilities of the ESP32, we would likely choose to use a cheaper ESP model that does not contain Bluetooth on our expansion panels in future designs.

It should be noted that we initially had issues flashing our code onto the ESP32. We realized that we did not initially connect the enable and GPIO0 pins which are necessary for programming. On our second PCB we corrected this issue and we were able to successfully load our code.

2.3.2 Schematic

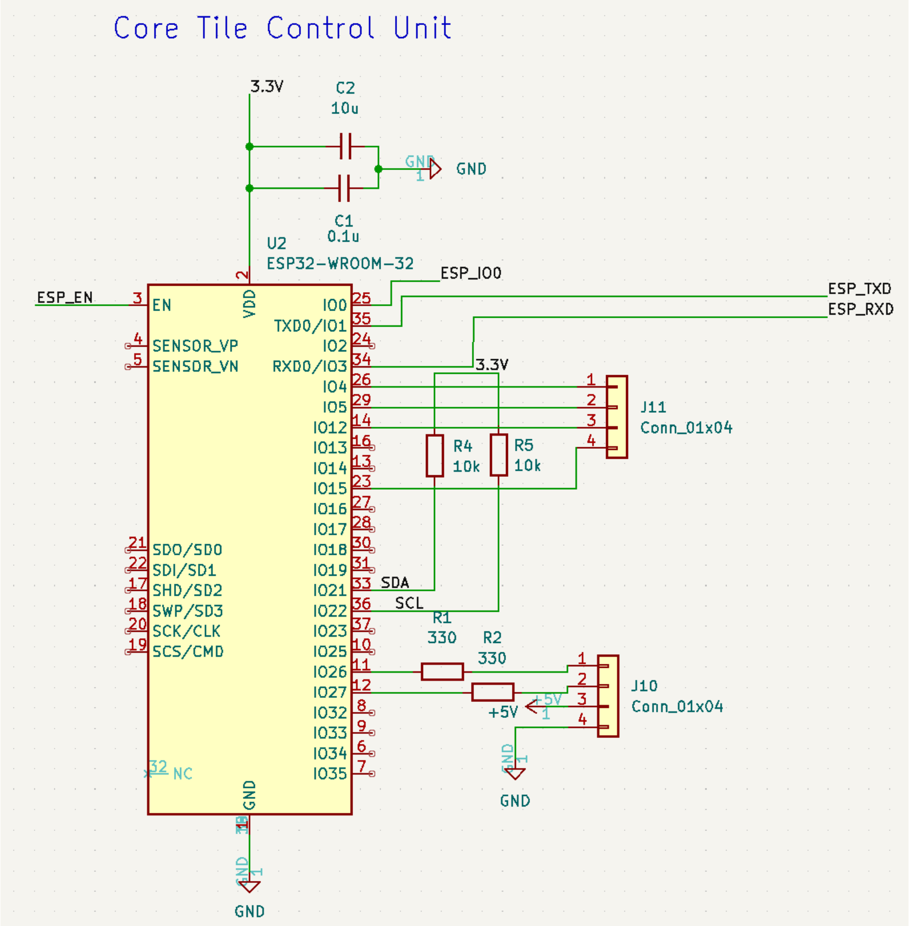


Figure 5: Control Schematic

2.4 Expansion Ports

2.4.1 Description

Each panel contains four expansion ports that receive both power and ground, as well as the I2C lines (SCL and SDA) from its neighboring panels. For our project, we utilized JST PH-2.0 connectors. These connectors required the use of crimped wires which led to some difficulties establishing good connections.

The machine-crimped wires that we acquired were therefore essential in maintaining consistent connections between the panels. Connectors on each side of every panel were necessary because the user can configure the panels in any configuration that cannot be predetermined. If we were to redesign our project, we would likely use sturdier connectors that didn't require the wires to be pre-crimped, and can withstand multiple re-installations.

2.4.2 Schematic

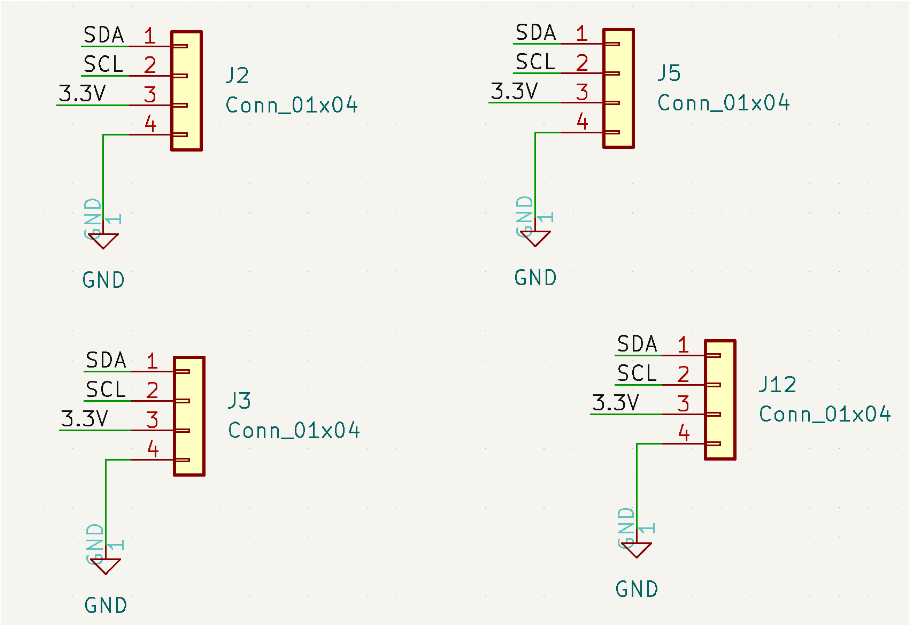
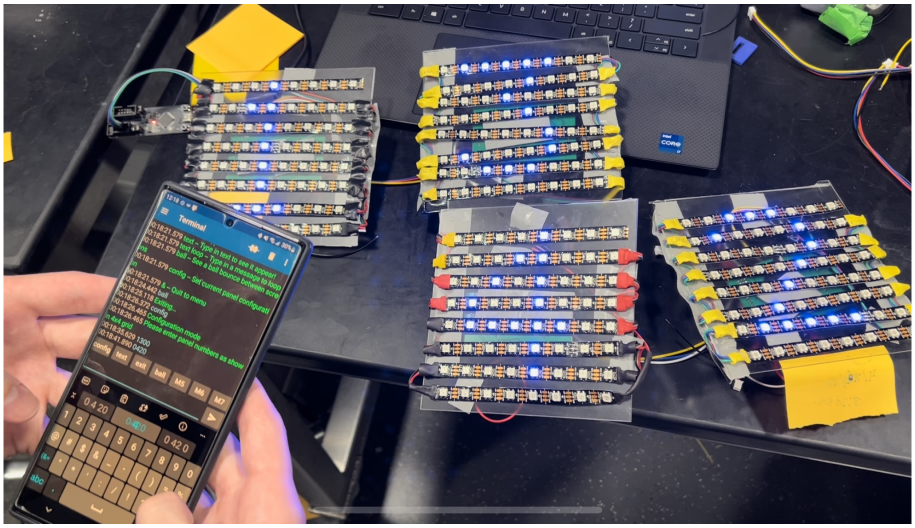


Figure 6: Expansion Ports Schematic

2.5 Using the Panels

There are four steps to use the LED panels:

1. Connect phone to core panel ESP32 over Bluetooth
2. Enter configuration mode and enter in configuration of panels on phone
3. Type in ”text” or ”ball” mode to enter desired mode of operation
4. Text/ball is displayed on panels



```
0:18:21.579 text – Type in text to see it appear!
0:18:21.579 text loop – Type in a message to loop
0:18:21.579 ball – See a ball bounce between scre
ens
00:18:21.579 config – Set current panel configurati
on
00:18:21.579 & – Quit to menu
00:18:24.442 ball
00:18:25.118 Exiting...
00:18:26.372 config
00:18:26.465 Configuration mode
00:18:26.465 Please enter panel numbers as show
n 4x4 grid
00:18:35.629 1300
00:18:41.890 0420
```

Figure 7: LED panels in configuration mode

Figure 7 above depicts the LED panels in configuration mode. Each panel is lit up with a specific number that the user then enter into their phone. In this example, the user would enter ”1300”, followed by ”0420”, ”0000”, and ”0000” to configure the panels. While this way of configuring the panels does not meet our initial goal of the boards automatically recognizing their configuration, we don’t see this method as too much of a burden. Most people will be hanging up the boards and configuring them only once, so taking an extra 30 seconds to do so is not much of a hassle.

3 Verification

3.1 Power

The power subsystem for our project consisted of converting a 120V power source (outlet) into 5V and 3.3V power. The components needed for this were relatively straightforward, consisting of off the shelf parts. Because of this, our verification of the power subsystem consisted mostly of checking that our components act as their data sheets say.

For our power source, which converted the 120V wall source to 5V, we were able to verify functionality by using a multi-meter. We checked that voltage from the power supply was indeed 5V, and that when connected to our PCB, each 5V component was receiving that power.

For the conversion of our 5V power to 3.3, we used a surface mounted voltage regulator on our PCB. To verify this component, we similarly used a multimeter to verify that the input signal of 5V was being properly converted to 3.3V.

We were able to verify that each component in our power subsystem worked as specified by the manufacturer.

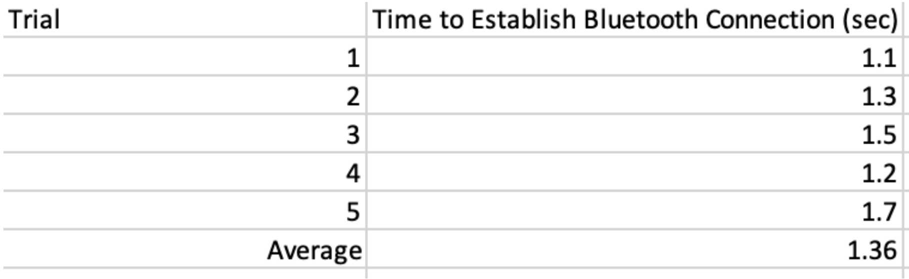
3.2 Control

One of the most important subsystems in our project is the control subsystem. This subsystem consists of our core panel's ESP32 microcontroller and its accompanying code. To verify the parts of this system, we mostly relied on visual verifications while using the product. This is because this subsystem is mostly just the programs that our panels run, and we can verify their functionally by simply observing them.

Firstly, we needed to verify the Bluetooth connection between a user's phone and the core panel. We connect the ESP32 using a serial Bluetooth terminal on an android phone. The process of connecting consists of tapping the ”connect” icon in the app, and after around 1 second, a confirmation of the connection will be displayed in the terminal. From our testing, connecting and pairing Bluetooth happened with few errors and in reasonable time.

Figure 8 below depicts the average time to connect to Bluetooth of five trials. To run this test we simply timed how long it took to connect to the ESP32 Bluetooth from our smartphone. We observed the average time to connect to Bluetooth was 1.36 seconds which is well under our five to seven second requirement.

We also needed to verify the programs we ran on the panels. At the time of the demo, these were ”ball” and ”text” modes. To verify these programs we simply ran each several times, and with several different configurations. When running



Trial	Time to Establish Bluetooth Connection (sec)
1	1.1
2	1.3
3	1.5
4	1.2
5	1.7
Average	1.36

Figure 8: Bluetooth connection time verification

them, we were visually checking for any glitches, lag, or improper boundaries. However, we did find a few minor issues when testing. We noticed that with more panels added, there is a slight decrease in frame rate. This was because of the nature of our I2C implementation; more panels simply meant more data needed to be sent by our control panel, thus taking more time per frame. Luckily, though, the difference in speed was negligible. One other thing we noticed with the data transmission was very occasionally some I2C packet data would be lost. Our code is structured so that this will not be a major point of failure, but rather a very minor visual glitch. What happens when a packet is lost is that for that one frame, a single pixel's data will be lost or have the wrong color.

Overall we were able to fully verify that the control system acts as desired and does not negatively interfere with the user's experience.

3.3 Expansion

Our expansion subsystem comprised of the ESP32 microcontrollers on our expansion panels, as well as the ports and wires connecting them together. Again, our verification for this subsystem was mostly visual, checking to see if the components worked as they should.

To check the expansion panel's microcontroller, we simply connected it to our core panel and ran a program. We verified that each panel was displaying what it should, and that it was in sync with all the other panels.

To verify our ports and connections, we used a multimeter to probe the power sources between the panels. By doing this we checked that both the ports and cables were working properly.

4 Cost and Schedule

4.1 Cost

4.1.1 Labor Costs

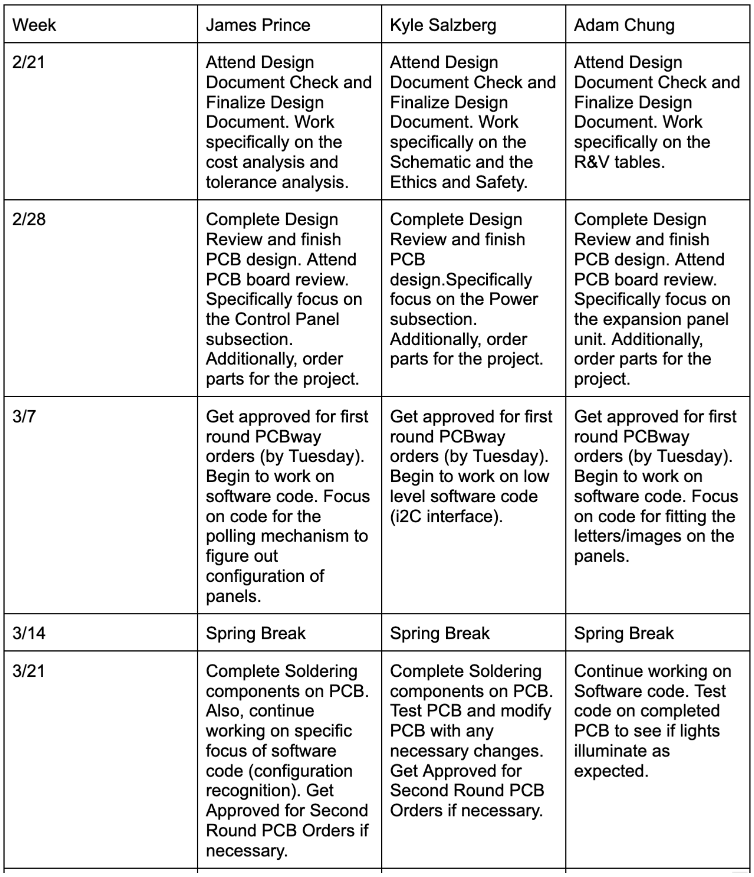
Our hourly cost was calculated using the average hourly cost for an electrical engineer in 2020 in the United States ($49.71/hr) according to the Bureau of Labor Statistics [3]

Name	James Prince	Kyle Salzberg	Adam Chung
Rate	$49.71/hr	$49.71/hr	$49.71/hr
Overhead Cost	2.5x Overhead	2.5x Overhead	2.5x Overhead
Hours Worked	100 hours	100 hours	100 hours
Total Labor Cost	$12,427.5	$12,427.5	$12,427.5

4.1.2 Parts

Part	Part Number	Unit Price	Quantity	Total
ESP32	ESP32-WROOM-32D	\$4.65	4	\$18.6
WS2812B Individually Addressable LEDs [4] (300 LEDs on strip)	WS2812B	\$31.99	1	\$31.99
Voltage Regulator	LM1117-3.3	\$1.77	1	\$1.77
Barrel Jack	PJ-063AH [5]	\$1.59	1	\$1.59
ESP Programmer	1965-ESP-PROG-ND	\$12.00	1	\$12.00
ESP Programmer	1965-ESP-PROG-ND	\$12.00	1	\$12.00
Programmer Header	3M156386-06-ND	\$1.59	4	\$6.39
Expansion Connectors	JST-PH 2.0	\$12.99	1 kit	\$12.99
Resistors	1206 SMD Kit	\$7.99	1 kit	\$7.99
Capacitors	0805 SMD Kit	\$7.99	1 kit	\$7.99
Polycarbonate Boards	N/A	\$5.48	4	\$21.92
Power Adapter	ALITOVE 5V 15A	\$29.99	1	\$21.99
Machine Crimped Wires	Keszoox PH 2.0mm JST	\$10.99	1	\$10.99
Total				\$156.21

4.2 Schedule



Week	James Prince	Kyle Salzberg	Adam Chung
2/21	Attend Design Document Check and Finalize Design Document. Work specifically on the cost analysis and tolerance analysis.	Attend Design Document Check and Finalize Design Document. Work specifically on the Schematic and the Ethics and Safety.	Attend Design Document Check and Finalize Design Document. Work specifically on the R&V tables.
2/28	Complete Design Review and finish PCB design. Attend PCB board review. Specifically focus on the Control Panel subsection. Additionally, order parts for the project.	Complete Design Review and finish PCB design.Specifically focus on the Power subsection. Additionally, order parts for the project.	Complete Design Review and finish PCB design. Attend PCB board review. Specifically focus on the expansion panel unit. Additionally, order parts for the project.
3/7	Get approved for first round PCBway orders (by Tuesday). Begin to work on software code. Focus on code for the polling mechanism to figure out configuration of panels.	Get approved for first round PCBway orders (by Tuesday). Begin to work on low level software code (i2C interface).	Get approved for first round PCBway orders (by Tuesday). Begin to work on software code. Focus on code for fitting the letters/images on the panels.
3/14	Spring Break	Spring Break	Spring Break
3/21	Complete Soldering components on PCB. Also, continue working on specific focus of software code (configuration recognition). Get Approved for Second Round PCB Orders if necessary.	Complete Soldering components on PCB. Test PCB and modify PCB with any necessary changes. Get Approved for Second Round PCB Orders if necessary.	Continue working on Software code. Test code on completed PCB to see if lights illuminate as expected.

3/28	Complete Individual Progress Reports. Continue working on software.	Complete Individual Progress Reports. Perform verification tests on PCB	Complete Individual Progress Reports. Continue working on software.
4/4	Finalize software and begin working on demonstration. Make sure the project is completely verified and reliable.	Begin working on demonstration. Start preparing a printed out version of the block diagram, high level requirements, and RV table.	Finalize software and begin working on demonstration. Make sure the project looks polished and is functional.
4/11	Finalize and practice demonstration. Begin working on the final presentation. Specifically work on how to describe the software of the project during the presentation.	Finalize and practice demonstration. Begin working on the final presentation. Specifically work on how to describe the hardware of the project during the presentation.	Finalize and practice demonstration. Begin working on the final presentation. Specifically work on the design format of the presentation.
4/18	Complete Mock Demo.	Complete Mock Demo.	Complete Mock Demo.
4/25	Finalize project and demo completed design. Work on the final paper. Focus on describing the software components of the final paper.	Finalize project and demo completed design. Work on the final paper. Focus on describing the hardware components of the final paper	Finalize project and demo completed design. Work on the final paper. Focus on describing the software components of the final paper.
5/2	Finalize and turn in the final paper.	Finalize and turn in the final paper	Finalize and turn in the final paper.

5 Ethics

While there are not an excessive amount of ethical issues that arise with our project, we still must take into consideration a few key ethical standards. Code I.1 of the 7.8 IEEE Code of Ethics states that we must "protect the privacy of others" [6]. With users of our product potentially entering in sensitive information to be displayed on the LED panels (e.g., text messages) we must take measures to ensure the privacy of this information. Since we don't plan on storing any user data, we must inform the user of this, but also ensure the user knows of the potential risk of sharing any private information. After actually implementing our project this wasn't too much of a concern since the user is only entering information to the Bluetooth terminal of their own choosing. Additionally, during the development of our project we made sure to be honest and trustworthy with our teammates and any others who provide us with assistance. As Section 1.3 of the ACM Code of Ethics states, we must be "honest about [our] qualifications, and about any limitations in [our] competence to complete a task" [7]. This is a crucial standard to follow during development, for designing anything that is outside of our qualifications could lead to potential failure and harm. After going through the development of our project we made sure to abide by these standards.

6 Conclusion

Overall, this semester we were able to accomplish the main goals of our project. We were successfully able to design and implement four LED panels that could be configured in any configuration and be controlled through Bluetooth. We successfully created modular panels that function cohesively as one larger LED array, as shown in previous sections. The two main modes that we implemented were "ball" and "text" modes. The "ball" mode successfully determined the boundaries of the panels and bounced off the edges of the current configuration. In "text" mode, the panels successfully displayed a scrolling text when too large to fit within the current orientation.

While we were ultimately successful in accomplishing most of our goals, we were not able to have the panels automatically recognize changes in their configuration. In future iterations of our project, we would hope to implement this functionality by redesigning the communication protocol between panels.

While the manufacturer—provided specification sheet claimed that 256 LEDs requires approximately 11.5 W, after extensive testing, we determined that our systems had a far lower power requirement and could run with a 5 W adapter. The only power issues that we would potentially run into were lights dimming at the end of the matrix, which is only a problem when running the lights at full brightness, all illuminated white — a very rare use case.

References

[1] Espressif Systems. ESP32-WROOM-32. 2022.

[2] Texas Instruments. ”LM1117 800-mA, Low-Dropout Linear Regulator”.SNOS412O. 2020.

[3] Bureau of Labor Statistics, Electrical and Electronics Engineers, [Online].Available: https://www.bls.gov/ooh/architecture-and-engineering/electrical-andelectronics-engineers.htm: :text. September 2021.

[4] Worldsemi. ”WS2812B Intelligent control LED integrated light source”.

[5]CUI Devices. ”DC Power Jack”. PJ-063AH.

[6]IEEE Policies, Section 7.8 Code of Ethics,
Standard I.1, [Online]. Available:https://www.ieee.org/about/corporate/governance/p7-8.html. June 2020.

[7] ACM Code of Ethics and Professional Conduct, Section 1.3. [Online]. Available: https://www.acm.org/code-of-ethics. June 2018.

Appendix A: Core Panel PCB Design

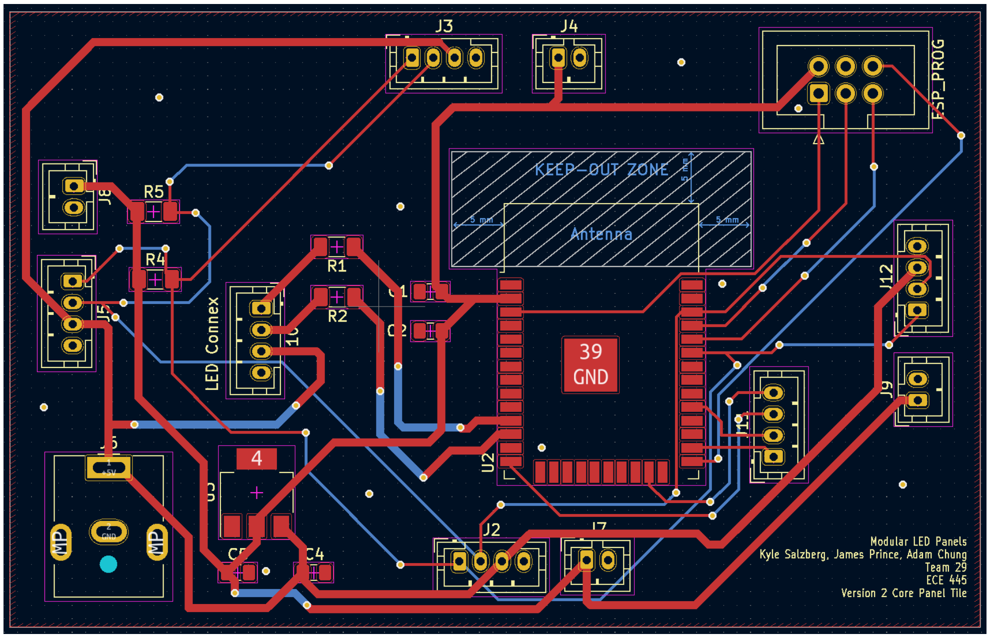


Figure 9: Final core panel PCB design

Appendix B: Expansion Panel PCB Design

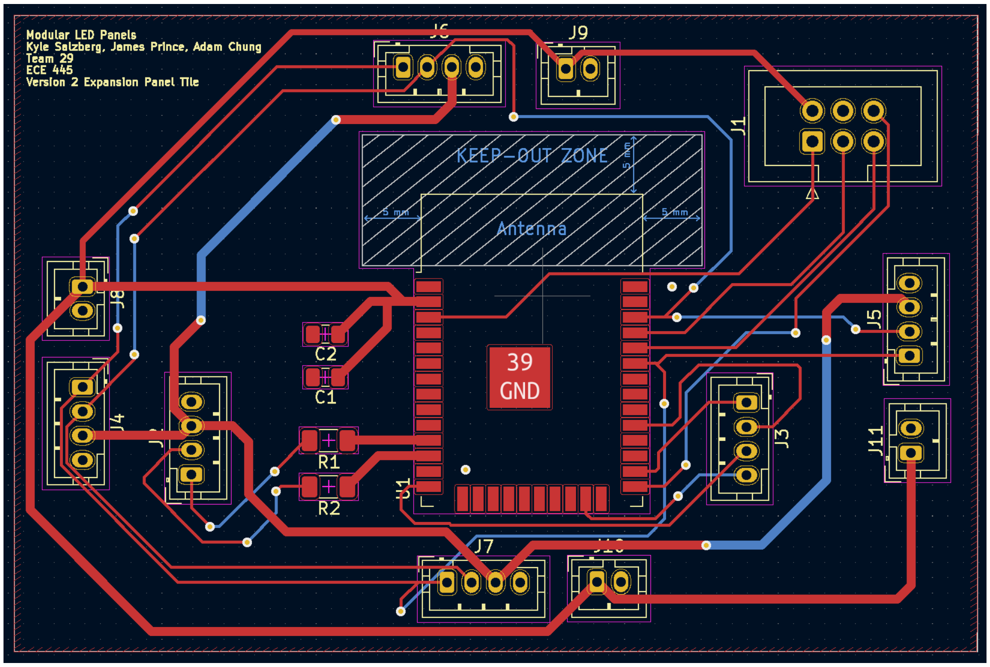


Figure 10: Final expansion panel PCB design

Appendix C: Power Requirements and Verifications

Requirements	Verification	Verified Status
1. Must receive data from either core tile or another Expansion Tile and parse contents	1A. Check on software the bit stream received is correct	Yes
2. Must be able to pass on data to other Expansion Tiles when packet is not addressed to it	2A. Verify that subsequent Expansion Tiles are displaying the expected LED output as listed in 3A.	Yes
3. Must be able to process then display packet data address to it onto the led array.	3A. Verify on singular expansion tile that LED matrix is displaying correct result. Compare to expected output on phone.	Yes

Appendix D: Core Tile Compute Unit Requirements and Verifications

Requirements	Verification	Verified Status
1. Must be able to establish Bluetooth connection with smartphone within 5-7 seconds	1A. Connect the phone to the compute unit. 1B. Send Bluetooth serial data from the phone to the ESP32. 1C. Verify ESP32 receives input and then sends data to expansion tile microcontrollers through I2C.	Yes
2. Must be able to process the algorithm to determine overall tile layout within 1 second.	2A. Verify display output with any arbitrary tile layout as specified in verification step 1A, 1B, and 1C. 2B. Update the tile layout after powering down the system. 2C. Repeat step 2A and verify the display output has adapted for the updated panel layout.	No
3. Must be able to process and send texts and images for each connected expansion tile within 30ms	3A. Connect one or more Expansion Units to the Core Unit and power the device. 3B. Connect the core panel to a Bluetooth device and send text and images as listed in step 1A. 3C. Verify LED output across panels and check each panel has texts and images correctly displayed across multiple panels.	No
4. Must be able to synchronize all the Expansion Units and refresh the LED output at 30 Hz.	4A. Use a slow-motion video to verify frames being updated at 30 Hz.	Yes

Appendix E: Expansion Tile Compute Unit Requirements and Verifications

Requirements	Verification	Verified Status
1. Must receive data from either core tile or another Expansion Tile and parse contents	1A. Check on software the bit stream received is correct	Yes
2. Must be able to pass on data to other Expansion Tiles when packet is not addressed to it	2A. Verify that subsequent Expansion Tiles are displaying the expected LED output as listed in 3A.	Yes
3. Must be able to process then display packet data address to it onto the led array.	3A. Verify on singular expansion tile that LED matrix is displaying correct result. Compare to expected output on phone.	Yes

Appendix F: LED Array Requirements and Verifications

Requirements	Verification	Verified Status
1. Must be provided with 5V±0.3 V power	1A. Use multimeter to confirm steady 5V±0.3 V output.	Yes
2. Must display correct image or text	2A. Visually verify that each LED matrix is displaying the correct text/image.	Yes

Appendix G: Expansion Port Requirements and Verifications

Requirements	Verification	Verified Status
1. Must transfer 5±0.3 V power and clock and data on SDL/SDA lines to next panel.	1A. Use oscilloscope to confirm steady 5±0.3 V output and correct SCL/SDA values	Yes
2. Must physically stay connected without human assistance.	1A. Visually determine that panels are connected, and staying connected.	Yes