CME 495 - Capstone Design Project

Electrical & Computer Engineering

University of Saskatchewan

Final Project Report

Festive Holiday Lights

Group 12

Date:	2023-04-05
Document:	Festive Holiday Lights
Authors:	Syed Aman Rashid (sar215), Bolarinwa Dare (btd222), Tasnim-Ul-Islam Bhuiyan (tab339), Vu Nguyen (vun786), Brix Marcellana (bem346)

Table of Contents

1. Executive Summary **7**
2. Problem Description **8**
2.1. Problem Definition 8
2.2. Scope 8
2.3. Major Objective 8
2.4. Constraints 8
2.5. Considerations 8
3. Requirement Specification **9**
3.1. Requirements 9
3.2. Justification of Requirements 10
R.1 10
R.2 10
R.3 10
R.4 10
R.5 10
R.6 10
R.7 11
R.8 11
R.9 11
R.10 11
R.11 11
R.12 11
4. System Alternatives 12
4.1. Methods used for generating alternatives 12
4.2. Descriptions of the sub-system-level design selected 12
4.3. Analysis of the sub-systems’ alternatives for the system-level design 19
4.3.1 Ability to Meet Technical Requirements 19
4.3.2 Environmental Aspects 19
4.3.3 Economic Aspects 19
4.3.4 Safety Aspects 19
4.3.5 Risk to Project Timeline 19
4.4. Discussion of which candidate design was chosen 20
4.4.1. Grid Dimensions 20
4.4.2. Connectivity of LED to the MCU 20
4.4.3. Driver 20
4.4.4. LED 20
4.4.5. MCU & Language 21
4.4.6. App (Frontend, Backend Language/IDE) 21
4.4.7. Weather Resistance Equipment 21
5. System Design 22
5.1. System Block Diagram 22
5.2. Detailed Design 23

5.2.1 Microcontroller Block 23
5.2.2 LED Block 25
5.2.3 Communication and Messaging 26
5.2.3.1 Driver Data Transmission Method 26
5.2.3.2 ESP32 Modes and Functionality 27
5.2.3.3 Phone App to Microcontroller Messaging 28
5.2.4 Phone App 30
5.2.4.1 Login Screen 31
5.2.4.2 Register Screen 31
5.2.4.3 Forgot Password Screen 32
5.2.4.4 Main Menu 33
5.2.4.5 Customization Menu 34
5.2.4.6 Advanced Customization Menu 35
5.2.4.7 Backend Design 35
5.3.1.Grid Dimensions 37
5.3.2 Power Supply 37
6. System Test Plan 39
6.1 Tests and Requirements Tested 38
6.2 Description of Test Plans and Result 39
TEST.01 39
TEST.02 40
TEST.03 41
TEST.04 41
TEST.05 42
TEST.06 43
TEST.07 44
TEST.08 46
TEST.09 49
TEST.10 51
TEST.11 53
TEST.12 53
TEST.13 54
TEST.14 55
7. Economic Analysis 58
7.1. Parts List and Budget 56
7.2. Overall Project Development and prototyping costs 57
8. Planning 60
8.1. Gantt Chart / Project Schedule 58
8.2. Predicted Timeline 59
8.3. Actual Timeline 61
8.4. Summary of Total Project Hours 63
9. Summary and Future Work 66
10. References 68
11. Appendices 71

List of Figures

- **Figure 1:** WS2811 driver pin layout
- **Figure 2:** MIC2860-D driver pin layout
- **Figure 3:** Pin Layout ESP DevkitC
- **Figure 4:** Pin Layout Arduino UNO R3
- **Figure 5:** A top-level diagram of the holiday festival lights consisting of its 3 major subsystems and the interfaces between the subsystems.
- **Figure 6:** A Visual Display of what the grid and how the output connections from the microcontroller lead to each of the LEDs.
- **Figure 7:** The Microcontroller block and Its Components
- **Figure 8:** Visual Display of the Microcontroller Block and its connection**s**
- **Figure 9:** The LED block and Its components
- **Figure 10:** The PCB Design of the LED block used for this project
- **Figure 11:** Visualize Driver Data Transmission
- **Figure 12:** Microcontroller Instruction Processing Modes
- **Figure 13:** Static Mode Messaging Format
- **Figure 14:** Increment Mode Messaging Format
- **Figure 15:** Instruction Array Sample Result when a “HELLO” message is sent
- **Figure 16:** Phone App Menus illustrating the 4 main screens (the login screen, main menu, customize menu, and advanced customize menu)
- **Figure 17:** Object-Oriented Design of the Holiday Festival Light App
- **Figure 18:** Phone App Menus illustrating the register screen, register pop up, Home Screen
- **Figure 19:** MongoDB Cloud Database tables
- **Figure 20:** Proof of our festive holiday lights app running on an android phone, accompanied. with the proof of its build being complete on the terminal.

Figure 21: Proof of our festive holiday lights app running on an IOS phone, accompanied with the proof of its build being complete on the terminal.

Figure 22: Proof of our festive holiday lights app running on the Web, accompanied with the proof of its build being complete on the terminal.

Figure 23: The Gantt chart of Group 12’s Project Schedule that helped in completing the project

Figure 24: The Gantt chart of Group 12’s Project Schedule that expands on how the Testing was planned out for the second half of the project.

List of Tables

Table 1: List of requirements

Table 2: Sub-system Alternatives

- **Table 3:** Test plan for the requirements
- **Table 4:** Testing to ensure that 80% of the LED’s maximum brightness is utilized in the final product.
- **Table 5:** Verify the IP64 rating of our design as being dust-tight and waterproof.
- **Table 6:** Verify that the festive holiday lights can run all ASCII symbols.
- **Table 7:** Verify for 120 Vac power supply compatibility.
- **Table 8:** Testing to check whether the microcontroller works well at an outside temperature range of -40 to +40 degree Celsius.
- **Table 9:** Testing the festive lights software and reconfiguring light design 5 times without adding new designs.
- **Table 10:** Test Reconfiguring the design from the main menu and count the number of clicks to change the light pattern.
- **Table 11:** Test to check if the app is installed correctly on mobile devices.
- **Table 12:** A test to check if the basic functionality works correctly
- **Table 13:** Verify the total cost of the entire material purchased or the project.
- **Table 14:** Test festive lights for proper exposure using a light meter
- **Table 15:** Test the range of the external antenna connected to the MCU block
- **Table 16:** Verify the durability of the grid as it hangs for a specific period.
- **Table 17:** Part Lists and cost
- **Table 18:** Overall project development and Cost prototyping
- **Table19:** Overall Software Cost
- **Table 20:** Table of predicted Tasks
- **Table 21:** Table of Actual Tasks
- **Table 22:** Planned vs Actual hours for Fall Term
- **Table 23:** Planned vs Actual hours for Winter Term

1. Executive Summary

The Festive Holiday Lights project aimed to develop a system that enables users to control strings of lights, adjust their brightness and color, and create customized patterns and messages on a 2D grid. Our system is designed for outdoor use, runs off 120 Vac, and features an intuitive user interface accessible via a smartphone and/or computer.

Our group conducted a comprehensive analysis of the project requirements and identified a set of key features that were essential to the system's success. We then evaluated several alternative system designs, taking into consideration factors such as cost, reliability, ease of use, and scalability. After careful consideration, a system architecture was selected that the group believed would best meet the client's needs.

The Festive Holiday Lights system consists of a microcontroller-based hardware design that interfaces with a set of LED light strings. Each string comprises multiple addressable RGB LED modules that enable independent control of the color and brightness of each individual LED. We designed the system to be scalable, allowing users to easily expand the size of their light display by adding additional light strings. We also created a working prototype of a 7 by 5 grid, demonstrating the feasibility of the system.

The accompanying software provides an intuitive user interface that allows users to control the light display using a computer or smartphone. We created apps for both iOS and Android platforms, as well as a website, providing a seamless experience for users across devices. Each individual light is customizable via the app, which works over a public communication protocol that was chosen specifically for the Festive Holiday Lights system.

Extensive testing of the system was conducted to ensure that it met all the design requirements[28]. The tests included power consumption analysis, hardware and software testing, and usability testing. The results of the testing plan indicated that the system is highly reliable and produces high-quality, dynamic light displays.

Overall, the Festive Holiday Lights system represents a successful collaboration between our group and our client. We believe that our system is highly innovative and provides a unique solution to the holiday light display market. We have designed the system to be highly scalable and easy to use, and we believe that it has the potential to become a highly popular product among consumers. We look forward to continuing to refine and improve the system, and to introducing it to a wider audience in the near future.

2. Problem Description

2.1. Problem Definition

Most of the display lights used in the holidays are a hassle to modify and set up. Some only display one type of color, brightness is not adjustable, and the user must manually change the structure of the lights just to display a message.

2.2. Scope

We intend to design a user-friendly system that controls strings of lights. The user must be able to customize brightness, colors, patterns, and messages displayed on the festive lights. The festive lights are to be controlled by the user via a phone app or website.

2.3. Major Objective

We need the lights to be customizable through software without the necessity of going outside and manually modifying the light patterns. The software must have full control of each light which allows each user to display specific patterns, and messages, and control the brightness.

2.4. Constraints

Lights must be resistant to different weather conditions since users will use the lights for outdoor displays. Lights must meet North American standards for their core supply voltage.

2.5. Considerations

At first, we must take safety and environmental considerations into account. According to North American standards, appliances in the household can use 120 Voltage alternating current which is at 60 Hz. Lead-containing products also cannot be used in the project as there are adverse health effects of overexposure to lead which may cause abdominal pain and cardiovascular effects. Our product is built to use during one of the most joyous times of the year, therefore we need to make sure that the brightness and loudness of the PWM controller for our holiday lights will not interfere with the area surrounding it

3. Requirement Specification

3.1. Requirements

Requirement Identification	Requirements
R.1	Each LED colour must be able to obtain 0.6 lm to increase visibility for outdoor use.
R.2	The system must be able to control 80% of the light's maximum brightness.
R.3	The light must have a waterproof rating of at least IP64
R.4	The grid design of the system must be at least 7 LEDs by 5 LEDs.
R.5	Power Supply must be able to run out of 120 Vac.
R.6	The Microcontroller chip and LED must at least work between the temperature range of -40℃ to +40 ℃.
R.7	The Phone app/website must have at least five pre-saved patterns stored in the database (backend) for each user to access it from their user profile.
R.8	The User must be able to change the pattern of the lights within a span of 3-6 clicks.
R.9	The App must be usable on both IOS and Android
R.10	The total system cost must not exceed the maximum prototype funding of $250.
R.11	Antenna range must reach fifteen metres without line of sight
R.12	Wire must not sag over 1 mm in a span of a week

Table 1: List of Requirements

3.2. Justification of Requirements

R.1

Each light needs to be bright enough to be visible across the road from the neighbor’s perspective. On extreme weather events (fog, heavy snow, or rain) light does not need to be completely visible, but on clear days, lights should be fully visible, and patterns must be clear.

R.2

To have an efficient choice of lights for the system, we want to be able to run the LED light at least 80% of its maximum brightness. This maximizes the entire value of the LED since there would not be any point if the LED chosen is not used close to its maximum potential.

R.3

Considering the extreme weather conditions in Saskatchewan where it is rainy during summer and snowy in winter, our light must have a certain level of water resistance to be functional outdoors. It is IP64, which means it is dust tight (no ingress of dust) and it is protected against water jets from any angle [1].

R.4

As we want our light to be able to display letters and due to our design measurement, we produce a ratio of 7:5 to display a single letter or digit with one column of lights between each letter/digit. This is the ideal size to display every ASCII character.

R.5

The power supply should be able to run based on North American standards which is 120 Vac. This meets the requirements of the client and the targeted audience of the product.

R.6

As our project is based on making an outdoor product, we considered the typical Saskatchewan weather and produced a range of -40℃ to +40 ℃. Therefore, our choice of microcontroller and light must abide by the mentioned range

R.7

The user must be able to create and save their profile in the app. Their profile will have the option to create and store at least five patterns which can be used later by that respective user with just a few clicks.

R.8

To quantify the user-friendliness of our app, we denoted the number of clicks to measure the app's complexity. We concluded on a range of three to six clicks to change a particular pattern to another one.

R.9

We aim to be able to have our project be used by both IOS and android users, therefore we plan to develop the app using a single codebase programming language which is React Native (React and JavaScript).

R.10

The maximum funding, we have for our prototype is only $250. Keeping the product below this range allows more accessibility to users and makes the product more affordable.

R.11

The antenna must cover a wide range of the area. The average Canadian home is around 1790 square feet [2]. Assuming each house is a square, we will need a range of 12.9 meters to account for the placement of the design. So ideally the antenna should be capable of reaching at least 15 meters or more for coverage.

R.12

The product must be durable and have longevity. This is necessary because it ensures that the product does not collapse and deem unusable within the first month of its use. A durable product encourages better rapport between the product and the users increasing its popularity and bringing in more buyers.

4. System Alternatives

4.1. Methods used for generating alternatives

When generating alternatives, the group considered industry solutions by looking at methods used to approach similar designs. This included the deconstruction of certain designs online and using certain materials as references for the product. The group also considered multiple approaches to the problem and considered whether they are viable or not. This included all the devices used for our product such as drivers, microcontrollers, LEDs, grid designs, and connectivity.

The group also took everyone's knowledge and experiences into consideration when generating alternatives. With diverse experiences in software, multiple programming languages and approaches to the software structure were considered. Open-source programs which were used to solve similar issues done by other individuals were also used as references when approaching the software design of the product. This included software optimization, user interface design, and additional features.

For the researched design, idea, or material to be considered an alternative, it must go through and meet all our generated system requirements. This includes major categories such as cost, performance, durability, simplicity, and testability.

4.2. Descriptions of the sub-system-level design selected

After discussing with the client, the group was given a specific design for the entire system. The required system must be constructed with strings of LED(s), not **strips** in such a way that each LED is fully customizable. Hence to meet the client's requirements, the group decided to split the system design into **sub-system designs.** Each sub-system design(s) has its alternatives (SSA) which are briefly described below:

ID	Sub-System Alternative
SSA.1	Dimension of the LED grid
SSA.2	Connectivity of LED to the MCU
SSA.3	Driver
SSA.4	LED
SSA.5	MCU and Language
SSA.6	App (Frontend, Backend Language & IDE)
SSA.7	Weather Resistance

Table 2: Sub-System Alternatives

SSA.1: Dimensions of the LED grid

Considering R.4, the LED grid needs to be 7 by 5 LED(s) for a single letter to be visible. The group has produced two possible grid designs 7 by 17 and 7 by 29.

- SSA.1.1: 7 by 29
- The grid design can display a 5-letter word, which is perfect for displaying common holiday words like "merry", "happy" and "hello". However, the equipment cost to implement this grid design would exceed the project's budget.
- SSA.1.2: 7 by 17
- Unlike 7 by 29 grid design, this design is only capable of displaying 3-letter words. Therefore, to be fully displayed, the message must be captured by frame and cycle through frames. On the other hand, the total equipment cost for the design is within the budget.

SSA.2: Connectivity of LED to the MCU

The connectivity of the LED lights must account for there major aspects: its feasibility to be constructed before the due date, the technical complexity of the method, and its impact to the budget. Common ideas apply for this project include: a series of lights with branches every 10 cm or so, or a single wire that branches off into parallel lines of LEDs. The series with branches is not ideal for the budget due to the difference of voltages across all points of the grid. This idea would require a variety of resistors and LEDs. The latter option is limited to having similar displays between each parallel line. This idea negates the requirement. Allowing slight modifications to these same narrow the options down to these two options:

- SSA.2.1: Using Token Ring Method to connect LED to the microcontroller
- A communication protocol where all the LEDs are connected in a ring topology. One or more tokens are passed between each led. Whichever LED holds the token is the LED that becomes programmable. This directly targets the main idea of the requirement which is to have to allow programmability for every single LED. One issue with this alternative is that it would require the microcontroller to run at a higher clock rate to make the transition of the display feel seamless.
- SSA.2.2: Using GPIO per Series Method to connect LED to the microcontroller

- This connection method does need a faster clock like the token ring. LEDs are quickly reprogrammable since they are connected directly to the microcontroller. Without using a fast clock, this method of connectivity is less power-hungry than the token ring. Nevertheless, two issues arise using this method. Microcontrollers might not be able to handle constant PWM control for multiple ports. The microcontroller might not even have enough ports. Microcontroller connection to the led grid might become too bulky with multiple wires connecting to each series. This could lead to a less durable product due to the sag caused by the weight of the wires.

SSA.3: Driver

There are multiple drivers in the market that provide the accessibility to program multiple LEDs differently. Two common drivers were commonly used, both are different in terms of their range of capabilities but provide the exact needs this project requires. Following R.2, to control the brightness of the LED, the group has decided to use an LED driver and produce two options: ws2811 and MIC2860-D.

- SSA.3.1: WS2811

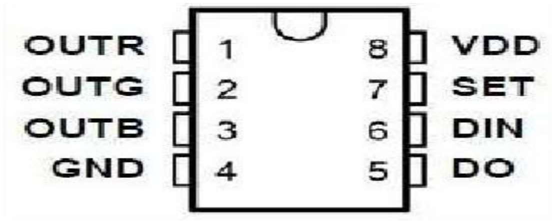


Figure 1: WS2811 driver pin layout [3]

- WS2811 is a three-output channels LED driver, so it allows full control over RGB light brightness using PWM. The only downside of WS2811 is that its maximum current draw is only 18.5mA, which in some cases will prevent LED lights to obtain their full brightness. [3]

 SSA.3.2: MIC2860-D

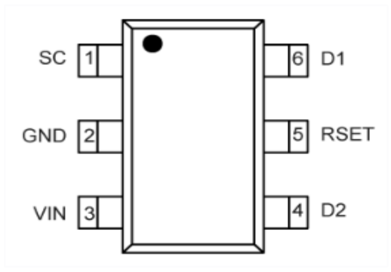


Figure 2: MIC2860-D driver pin layout [21]

- MIC2860-D is a 2 WLED channel which can control the brightness of two LED lights at the same time. The disadvantage is that MIC2860-D can only work with white LED light. Unlike WS2811, the driver’s maximum current draw is not an issue since it can draw up to 33.2 mA [21], which exceeds the current a standard LED needs to obtain full brightness (20mA) [4].

SSA.4: LED

A variety of LEDs in the market can go well with this project. Three major categories were considered when narrowing down the list of led options: brightness, cost, and compatibility. Both the LEDs below are fully capable in their ability to be used outdoors. Budget wise these are one of the cheapest ones in the market and are proven to be fully compatible with the chosen drivers.

 SSA.4.1: 100F5T-RGB-CC

- Since 100F5T-RGB-CC has an RGB LED it can display all three colors. The light maximum forward current is 20mA, which makes the light able to obtain more than 80% of its brightness while using a low current draw driver such as SSA.3.1. Another advantage of this LED is that it comes with a cheap price, which is only $10/100 units [5].

 SSA.4.2: HV-5RGB60

- C513A-WSN is a white LED so it can only display a single color. The light maximum forward current is 30mA [20], which will require a high current draw such as SSA.3.2 to obtain full brightness. Compared with SSA.4.1 cost, this LED is more expensive, C513A-

WSN costs $29/100 units [6].

SSA.5 MCU chip & Language

The project requires the festive holiday lights to be controlled from a certain range by the user from inside their house. As a result, the choice of microcontroller had to be narrowed down to WIFI capable chips only. According to our project budget and antenna range we came up with the following two alternatives:

- SSA.5.1: ESP32-DevkitC v4 chip
- The ESP32-Series chips fit well with the project guidelines prioritizing the Wi-Fi requirement. The ESP32 board shown above uses ESP32-WROOM-32U. This model allows the board to install an external antenna via an IPEX connector and as a result, gives a higher possibility for a wider Wi-Fi range.[7]

  ESP32 can work well in temperatures ranging from -40°C to +125°C. Because of its complex calibration circuitry, the ESP32 can dynamically correct errors in external circuits and adapt to changes in the environment. The ESP32 chips have low power consumption with an optimum operating range of 3.0 -3.6 V. [8]

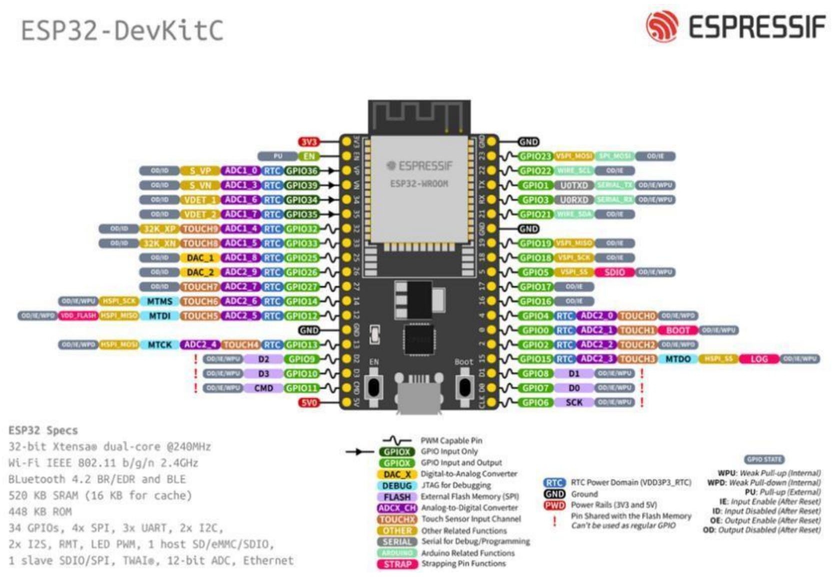

ESP32-DevKitC
ESPRESSIF
ESP32 Specs
32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz
Bluetooth 4.2 BR/EDR and BLE
520 KB SRAM (16 KB for cache)
448 KB ROM
34 GPIOs, 4x SPI, 3x UART, 2x I2C,
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

Figure 3: Pin Layout ESP DevkitC

Figure 3 shows that ESP32-DevkitC has a total of 38 pins. Of those 38 pins, there are 28 GPIO ports which are all PWM-capable pins. The ESP32 devkit uses a Tensilica Dual-Core Xtensa LX6 32-bit microcontroller which follows a Symmetric Multiprocessing Architecture (SMP).[9]

- SSA.5.2: Arduino UNO Wi-Fi Rev3 Chip

- The Arduino UNO chip uses an ATmega 16U2 8-bit microcontroller and has a less complex structure compared to ESP32. The operating temperature range is -40°C to +85°C and the power consumption is 2.7V – 5.5 V. [10]

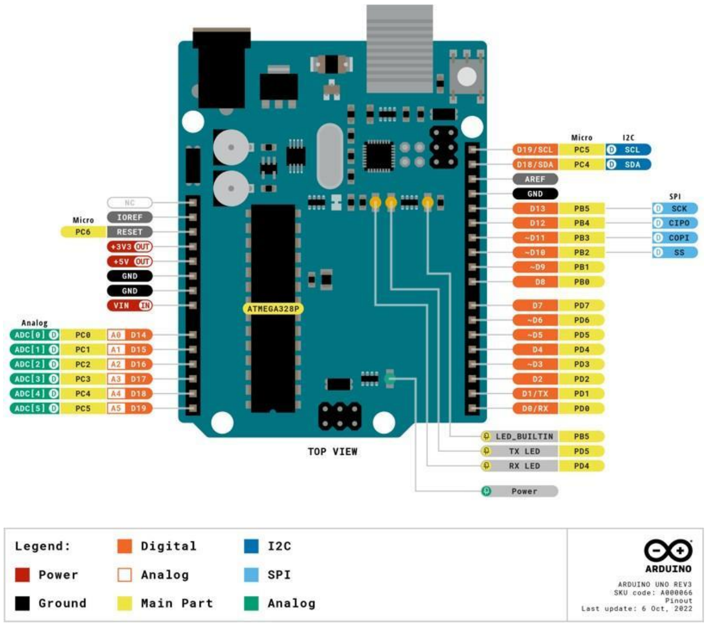


Figure 4: Pin Layout Arduino UNO R3

Figure 4 shows the Arduino chip has a total of 32 pins. Of those 32 pins, there are only 14 GPIO pins and only six of these are PWM capable. This might not be enough to meet the project requirements.[11]

SSA.6 App (Frontend, Backend Language/IDE)

The project requires either an app or a website (R.7). The team decided to proceed with the task of building an app as it is easier for the user to control the holiday lights via a few clicks on their phone rather than going to a website. Hence, we came up with the following alternatives for the app:

- SSA.6.1: DART
  Framework/IDE used: Flutter
- Dart can be used to build both **Android** and **iOS** applications using a single codebase. The language syntax is similar to C#, Java, and JavaScript. This helps for native as well as cross-platform application development.[12]
- SSA.6.2: ReactJS
  Framework/IDE used: React Native
- React Native is an open-source mobile application framework which is used to develop apps for android, apple, and windows platforms. React Native implements native apps for **Android** and **iOS** using React and JavaScript to create a single codebase that can be shared across platforms. The languages used are JavaScript and React which are more coder-friendly and have better resources compared to DART. [13]
- SSA.6.3: SWIFT
  Framework/IDE used: XCode
- SWIFT is an ideal programming language to build iOS, iPadOS, macOS, tvOS and watchOS applications. Apple uses SWIFT to take a modern approach for performance, safety, and software design patterns. Swift is a successor to both the C and Objective-C languages. It includes low-level primitives such as types, flow control, and operators. It also provides object-oriented features such as classes, protocols, and generics, giving the performance and power we demand.[14] XCode brings user interface design, coding, testing, debugging, and submitting to the App Store into a unified workflow.[15]

SSA.7: Weather Resistance Equipment

When it comes to weather resistance options for electrical equipment, there are many products can be found on the market. However, this project requires the festive light have at least IP64 water resistance rated to be used outdoor. Considering this requirement, the group narrowed the sub-system alternative option to the two options below:

- SSA.7.1: 442C Silicone Coating
- 422C is the circuit waterproof coating which meets requirement R.3 since it offers strong

protection against moisture, corrosion, dirt, and dust. It is quick to set up and can be applied to any type of wire [16]. The downside of 442C is that it is highly flammable and can cause serious eye damage [17].

- SSA.7.2: Wirefy Heat Shrink Tubing
- Like 442C, the Wirefy Heat Shrink Tubing also provides an environmental seal against moisture and dirt. Unlike 442C, the Wirefy Heat Shrink Tubing is a safe use product. It is a UL list product, which means the product is fully tested to make sure it meets Canada safety standards. [22]

4.3. Analysis of the sub-systems' alternatives for the system-level design

4.3.1 Ability to Meet Technical Requirements

After doing an analysis of all the sub-system alternatives to meet the project's technical requirements, it is found that SSA.4.1 might not meet R.6 because its operating temperature range is not specified in the product description. SSA.6.3 also does not meet the technical requirements and contradicts R.9 as the project requires an app usable on both android and iOS platforms. SWIFT is used to code applications only for iOS users. Other than the ones mentioned above, the rest of the alternatives met the project's technical requirements.

4.3.2 Environmental Aspects

This analysis will consider all alternative potential threats or issues to the environment. After analysis, all the sub-system alternatives met the environmental aspects of the project.

4.3.3 Economic Aspects

This analysis determines which alternatives posed the risk of exceeding the project's budget. SSA.1.1 will not meet the economic aspect since its equipment cost would go over the budget. SSA.5.2 also did not meet the economic aspect as it costs 49.99 C$ per unit which will contradict R.10.

4.3.4 Safety Aspects

SSA.7.1 is unlikely to meet the project's safety standards since 442C is a hazardous chemical which is highly flammable and can potentially cause severe damage to the user's eyes.

4.3.5 Risk to Project Timeline

Analysis of SSA.1.1, SSA.1.2, SSA.3.1, and SSA.3.2 to identify which alternatives pose the least risk of extending the project timeline. When comparing SSA.1 alternatives, SSA.1.2 would take a longer time to implement due to its larger grid size. For SSA.3 alternatives, SSA.3.2 proved to take longer to connect to the LED light since each LED requires a WS2811 driver, while a MIC2860-D can control 2 LEDs at the same time.

4.4. Discussion of which candidate design was chosen

When choosing the most optimal design or material the group assessed the trade-off between cost, performance, durability, simplicity, and testability.

4.4.1. Grid Dimensions

Out of the 2 grid dimension alternatives, the SSA1.1 ratio was chosen due to its lower equipment requirement, which allows the design to stay within the budget and avoid scheduling risk.

4.4.2. Connectivity of LED to the MCU

SSA2.1 would require a microcontroller that has a faster clock rate. This means the overall system would consume more power over time. The technicality of SSA2.1 would require more research and could cause a schedule risk if it's not tackled early. SSA2.2 is much simpler to implement but it has a higher potential of causing more issues in the long run. The requirement for the microcontroller to have at least 17 GPIO pins can easily be accommodated by searching for more versatile microcontrollers. The durability issue regarding the weight of the horizontal wire that runs across the grid can be compensated by modifying the mounting scheme of the grid.

4.4.3. Driver

Even though SSA3.1 might take longer while we build the final product, it was chosen as the project LED driver due to its feature which allows users to fully customize an RGB light.

4.4.4. LED

As for the LED alternatives, SSA4.1 was chosen because the light can display multiple colors, which is within the project's scope. Moreover, SSA4.1 was cost-friendly compared to SSA4.2, which met R.10.

4.4.5. MCU & Language

Since the project's grid dimensions were chosen to be 7 by 17, it requires at least 17 LED(s) to work simultaneously to display three letters at once. SSA5.2 only has 14 GPIO pins among which only six are PWM-capable pins whereas, SSA5.1 has a total of 28 GPIO pins and all the pins are PWM-capable. Therefore, SSA5.1 (ESP32-DevkitC) was chosen over SSA5.2 as it will be able to support the chosen grid dimensions and display the letters successfully.

4.4.6. App (Frontend, Backend Language/IDE)

SSA6.2 was chosen as the frontend language because Javascript and React is familiar among all the group members and they have more resources online due to a broader community reach. SSA6.2 also meets R.9 because it provides a single codebase for both iOS and android platforms.

4.4.7. Weather Resistance Equipment

SSA7.2 was chosen as the project weather resistance equipment since it meets R.3, and it qualifies Canada safety standards. The SSA7.2 reduces the risk of having to encounter safety risk in assembly. Although it requires a longer assembly time, this can be compensated by narrowing the application of the tube to specified areas of the circuit.

5. System Design

5.1. System Block Diagram

The festival light system consists of 3 major subsystems: The microcontroller block acts as a bridge between the phone app and the led blocks. The led block contains a driver and an led that allows for individual control, and the phone app allows for the user to reprogram the grid wirelessly. For the interfaces, the microcontroller block is connected to the led block through wires for PWM. The led blocks are also connected to the power supply and the phone app is connected wirelessly to the microcontroller block through Wi-Fi.

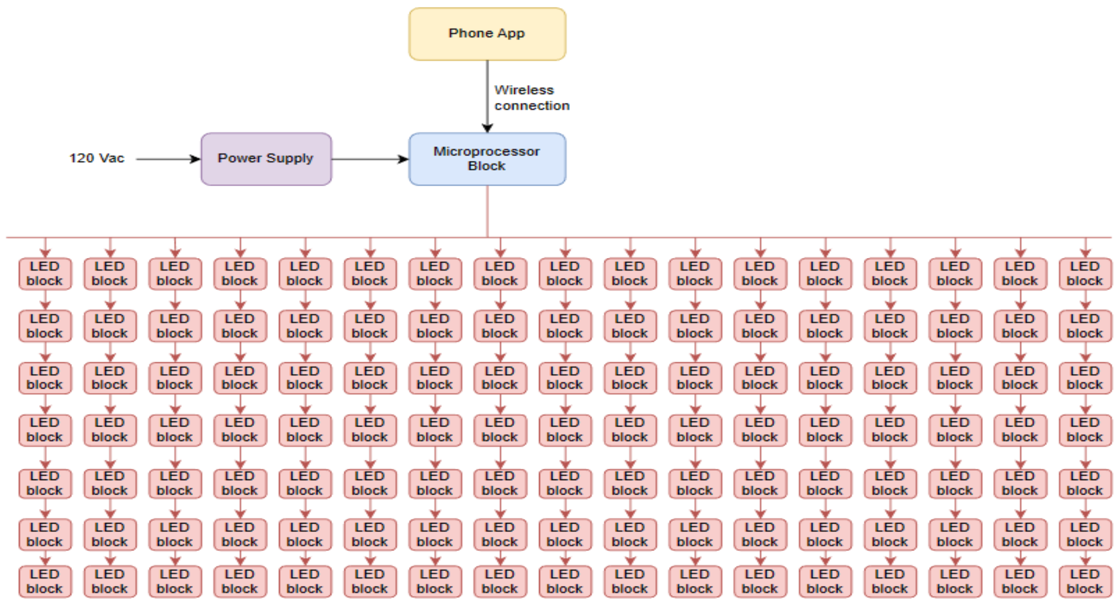


Figure 5: A top-level diagram of the holiday festival lights consisting of its three major subsystems and the interfaces between the subsystems (Note: PSU branches to all LED blocks through the microprocessor block).

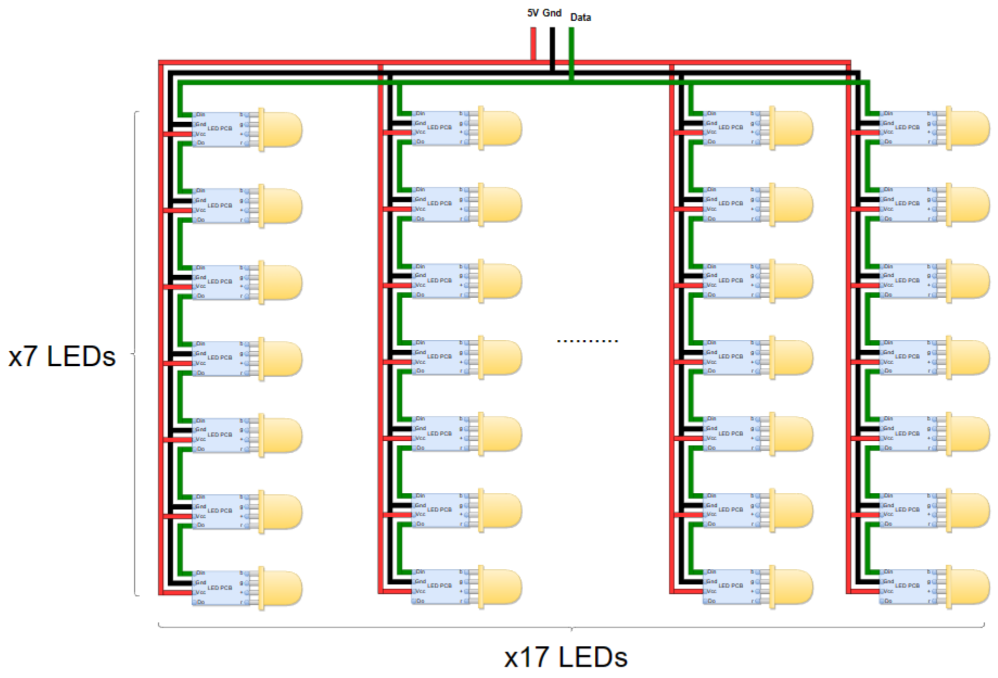


Figure 6: A Visual Display of what the grid and how the output connections from the microcontroller lead to each of the LEDs.

5.2. Detailed Design

5.2.1 Microcontroller Block

The microcontroller block is the control network of the entire system. It contains the ESP32 Series chips that was chosen in the system alternatives. The ESP32 DevkitC contains 28 general purpose input output (GPIO) pins that are viable for PWM, and in the microcontroller block 17 of these GPIO ports will be used to control 17 series of led blocks in the system. The 32-bit microcontroller will require 3.6 Volts and a maximum of about 1 Amp assuming all 17 pins are used.

The microcontroller chip used in the design does have a built-in antenna. The ESP32 also supports the use of an external antenna which allows for an increased range of connection to the Wi-Fi. The ESP32 DevkitC will be controlled to automatically connect to the Wi-Fi whenever device is powered.

As the central network of the entire system, the microcontroller block is connected to multiple interfaces. The system has access to the constant dc voltage from the power supply. It is also connected to the Wi-Fi for MQTT. The microcontroller block has 19 outputs, 2 being the Vdd and the Ground wire that branch off from the power supply, and 17 data wire for PWM control as seen in figure 3. To power the microprocessor, we can simply connect the 5 V pin to 5 V wire from the power supply and the Gnd pin to the Gnd wire from the power supply. According to the ESP32 datasheet, a 22 uF capacitor should be enough for the microcontroller. Since an antenna is also used in this system, a 100 uF capacitor was used. The ESP32 contains an internal voltage regulator allowing the power pins to connect to the microcontroller directly.

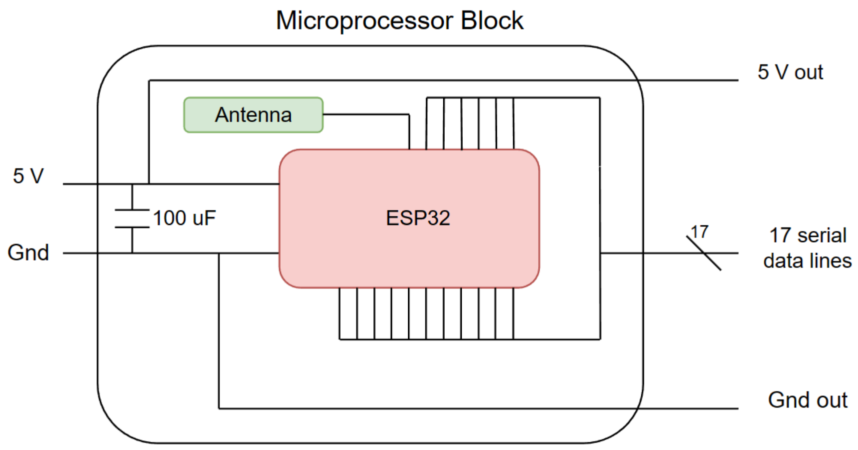


Figure 7: The Microcontroller block and Its Components

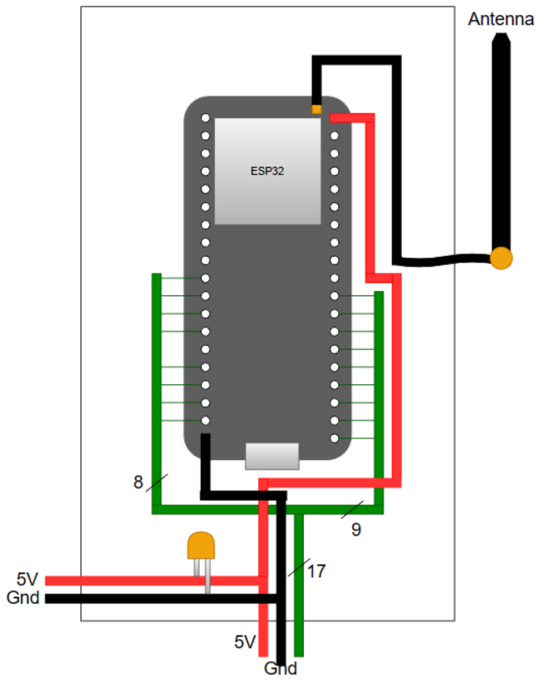


Figure 8: Visual Display of the Microcontroller Block and its connections

5.2.2 LED Block

The led block has two core components: the led and the driver. The 3 diodes seen in figure 7 (figure below) represent the RGB led. Based on the driver pins shown in figure # (driver pin layout), the driver has 3 output controls one for each of the red, green, and blue colors of the led. The RGB LED is also connected to the 5 Volt input from the power supply. The 5 Volt input is connected to a bypass capacitor [18] to remove noise from high frequencies. It is also connected to a resistor for impedance to prevent reflection and hot-swap protection.

The led block contains 2 data pins, an input, and an output. The WS2811 driver allows its data ports to be daisy-chained to each other. This means that a single GPIO port from the microcontroller can output a single serial data signal and control multiple led blocks. The data output of each driver is chained to the next driver allowing full control to each led in a single series.

For the LED block interfaces, the 5 Volt and Ground output that branched off from the microcontroller block are both connected to the led block. The led block will also have a data input and output wire to a daisy chain between led blocks and connect to the microcontroller.

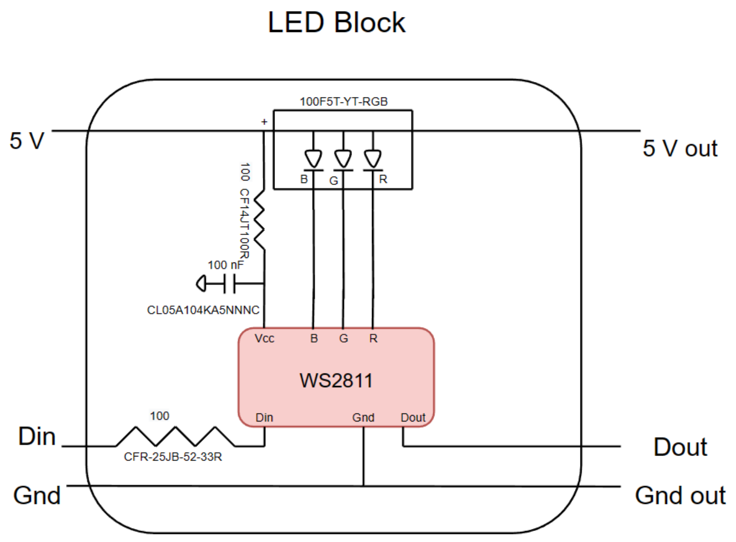


Figure 9: The LED block and Its components

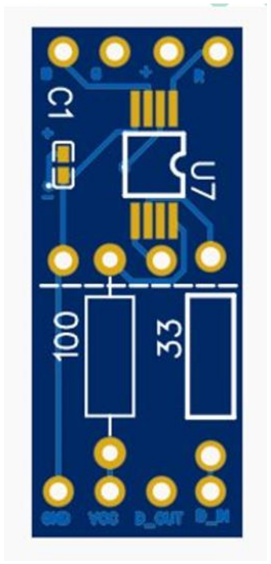


Figure 10: The PCB Design of the LED block used for this project

5.2.3 Communication and Messaging

5.2.3.1 Driver Data Transmission Method

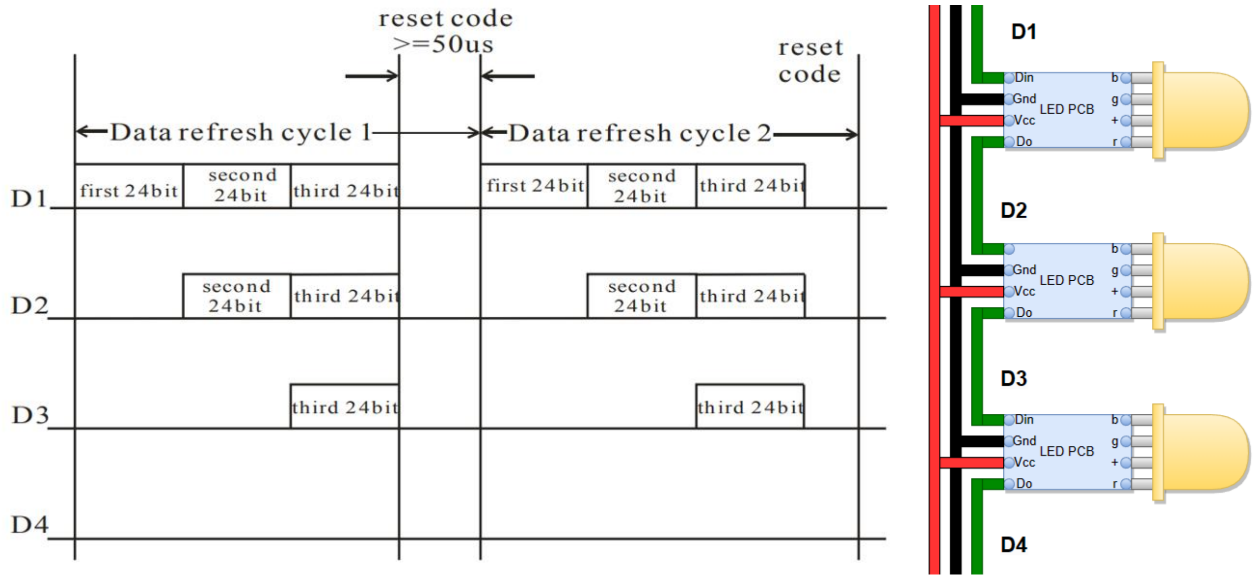


Figure 11: Visualize Driver Data Transmission

The driver has two serial data line pins, one for input and one for output. These data pins take in a sequence of instructions where it utilizes the first 24-bits significant to reprogram its internal PWM control for the red, green, and blue output pins. On the example seen in Figure 11, D1 enters the first LED block where the first 24-bits were used. Using the driver's internal re-amplification mechanism, the first 24-bit of the sequence is removed and the remaining data are retransmitted into the data output line where the next LED block takes the remaining instructions. Between each serial data transmitted, there is a 50us gap to differentiate the instructions from one another. The Arduino development environment offers multiple libraries used to control the driver. The FastLED library was used in due to its simplicity in terms of storing each 24-bit LED instruction into a variable. These instruction variables also have the capability to be stored within an array of instructions allowing for shorter, less complex code.

5.2.3.2 ESP32 Modes and Functionality

Using the FastLED library an internal instruction array was created for the festival holiday lights ESP32 program. This internal instruction array contains an array of seven 24-bit LED instructions where the number of these arrays are 16 times the width of the grid. With this mechanism, the large instruction array can be re-programmed and the system can process this instruction array into the grid however it likes.

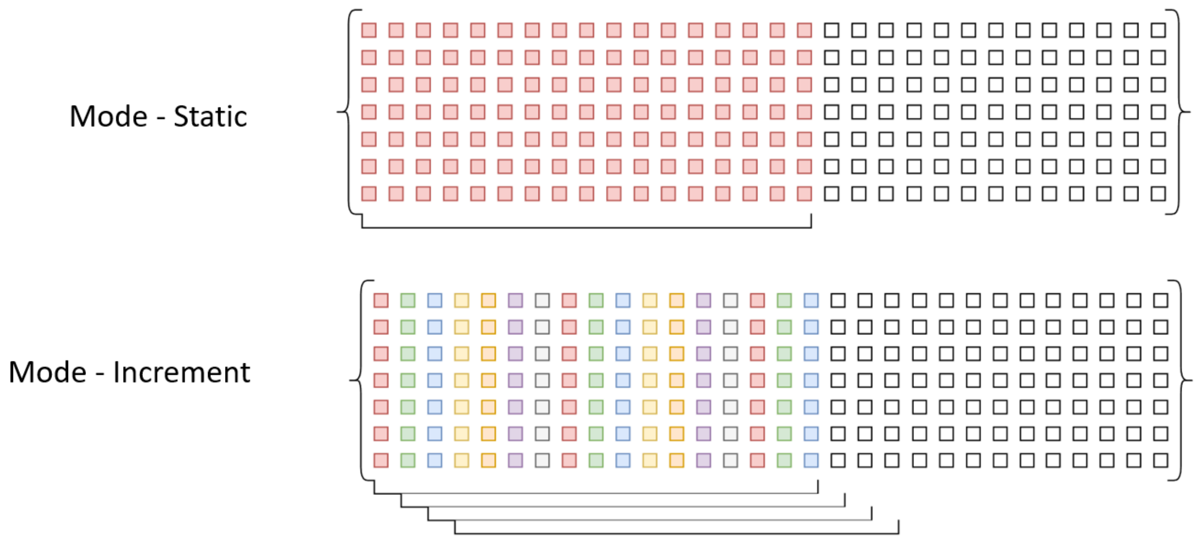


Figure 12: Microcontroller Instruction Processing Modes

As seen from the image in figure 12, two modes were devised for this project to meet the stated system requirements. The squared bracket seen at the bottom of each array indicates the size of the grid and what the grid will display.

The static mode will simply read the first set of instructions in the instruction array based on the grid size. The static mode is used simply for the individual LED control. With this mode the user can simply reprogram the instruction array given the complete grid size and each instruction will directly correspond to each LED.

The increment mode starts at the beginning of the instruction array and increments a single column to the right of each cycle until it fully displays all the instructions contained within the array. This mode is mainly used for message displays. An ASCII library was created to convert the message sent by the user and modify the internal instruction array. The Increment mode will also be initialized once the message is sent, and the system will cycle through the entire message.

5.2.3.3 Phone App to Microcontroller Messaging

Considering the use of WiFi, MQTT (Message Queuing Telemetry Transport) was used as the network protocol for the communication between the phone app and the microcontroller. MQTT is efficient with machine-to-machine communication and its optimal for networks that are unstable due to its persistent sessions.

With the two modes constructed for the microcontroller, two messaging formats were also created due to the difference in the programming mechanism of the two modes. The static mode needs a format the specifies the brightness of each of the red, green, and blue LEDs while the increment mode requires a format where the strings of messages are sent to be converted using the ASCII library. Both formats were controlled to indicate the mode of the microcontroller based on the first digit of the message. The first character of “0” will indicate a static mode and “1” will indicate an increment mode.

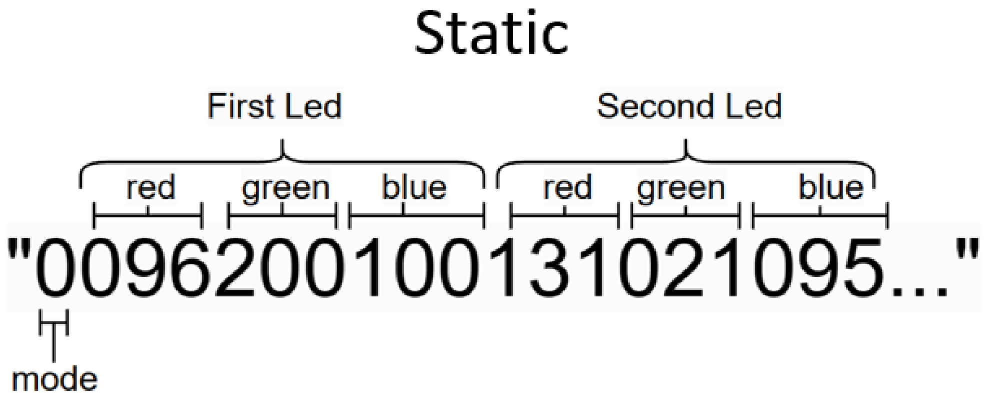


Figure 13: Static Mode Messaging Format

The format for the static messaging mode remains fixed. Each LED color instruction will contain 9 digits where the first 9 digits will always modify the top-left LED of the grid moving to the right once, then the bottom after each 9 digits. This format is used to prevent the need of having to ID each LED on the grid. Since LED's have to also be programed to be turned off, assigning IDs would potentially lead to a longer message. Within the 9-digit instruction for each LED, each color of the LED is assigned a 3 digit value ranging from 0-255 which indicates the designated brightness of each diode. With the programed converter on the microcontroller each of the 3-digit values can easily be converted to the 24-bit instruction for the LED.

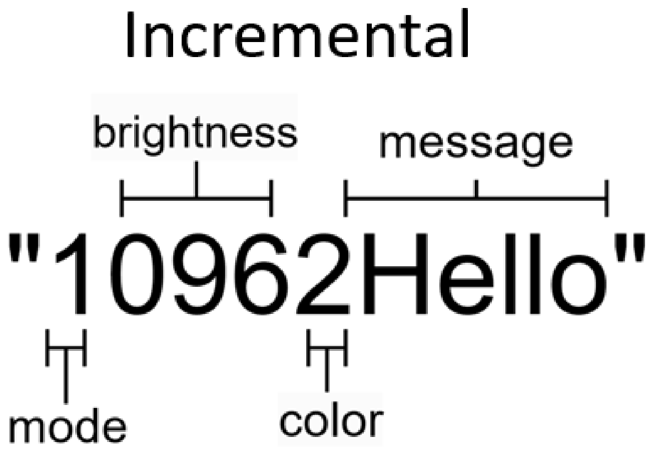


Figure 14: Increment Mode Messaging Format

The format for the incremental messaging mode remains always begins with a 1. The next three digits will control the brightness of the LED which also ranges from 0-255. For the messaging control, the user is purposely limited to choosing from a set of colors to improve user friendliness. Due to this method, the color value is limited to only be a single digit. The remaining part of the message is processed to as the input message sent by the user. Each character will then be processed in the ASCII library to be added into the instruction array as seen below.

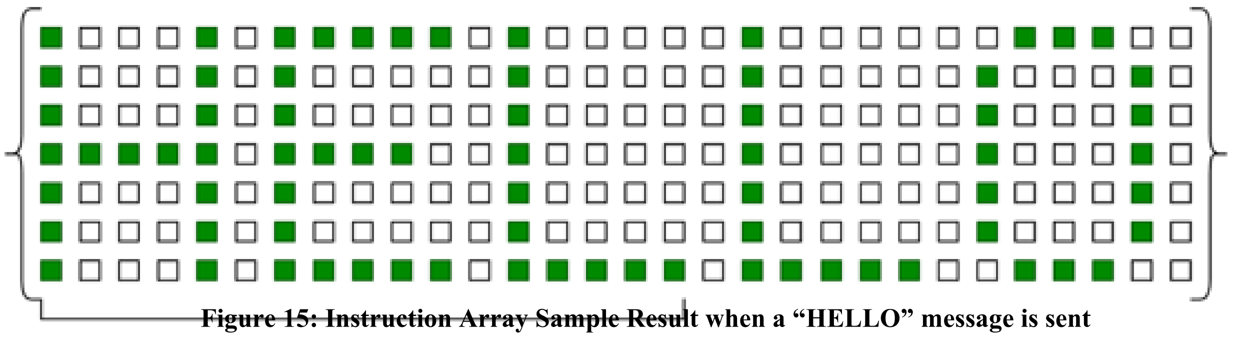


Figure 15: Instruction Array Sample Result when a "HELLO" message is sent

5.2.4 Phone App

The holiday festival lights software is composed of 4 significant user interfaces: the login screen for the user to login into their account, the main menu which allows the user to import specific designs to the led grid, the customization menu which allows the user to quickly write a message as a design, the advanced customization menu which gives access to every single led, and color.

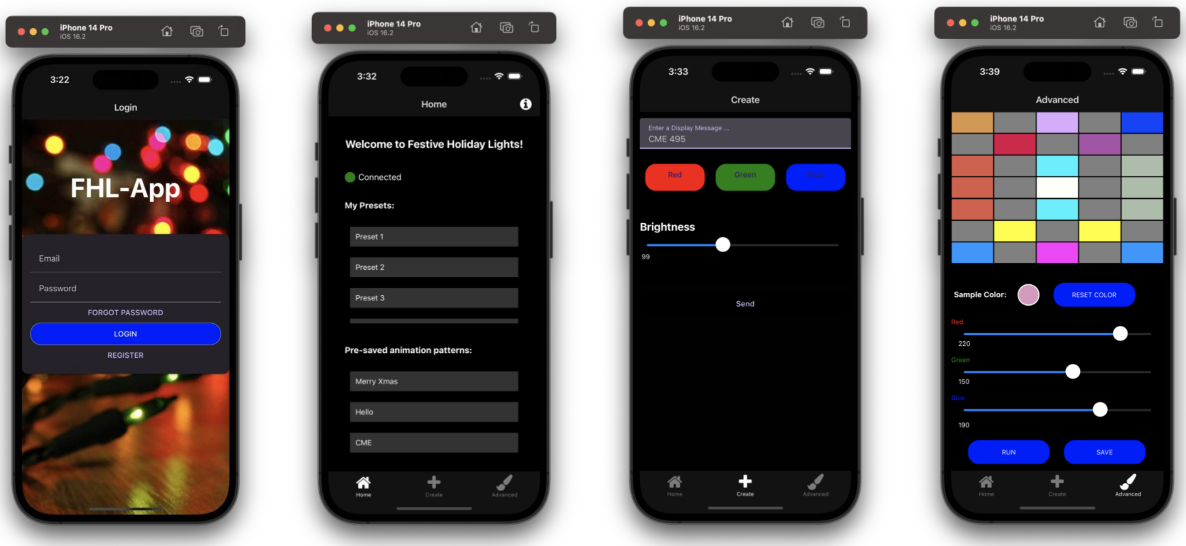
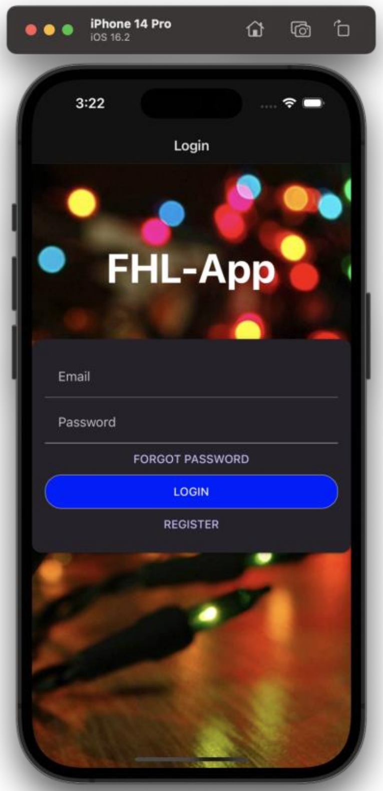


Figure 16: Phone App Menus illustrating the 4 main screens (the login screen, main menu, customize menu, and advanced customize menu)

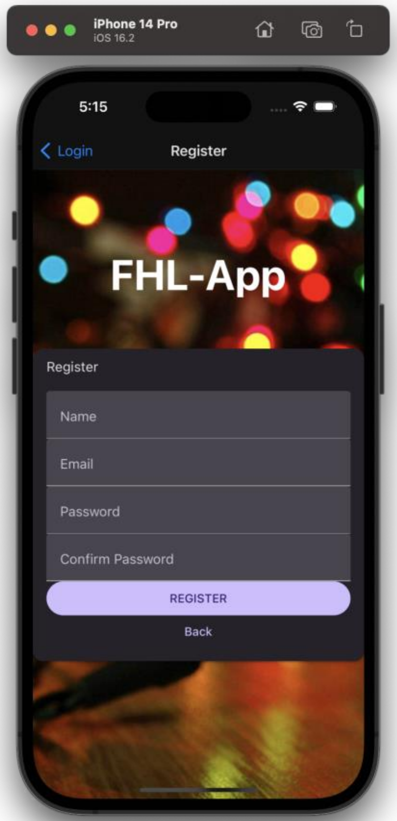
5.2.4.1 Login Screen

A login screen is a form that users fill out with their login credentials to access their accounts. The system checks the credentials and grants access if they are correct.



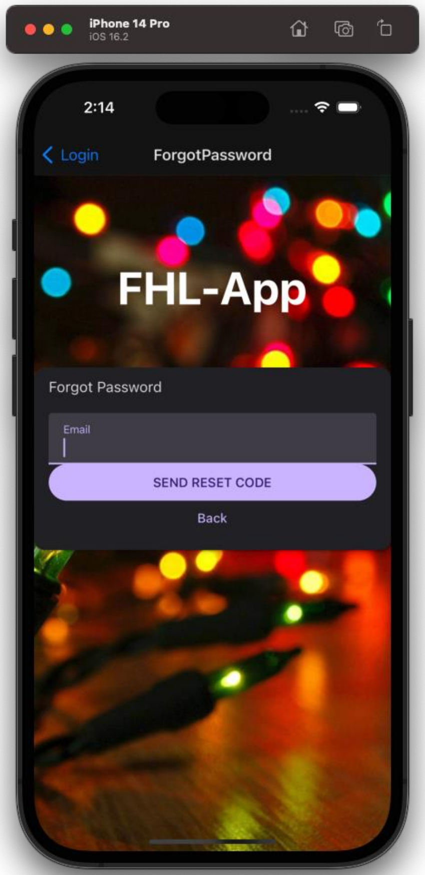
5.2.4.2 Register Screen

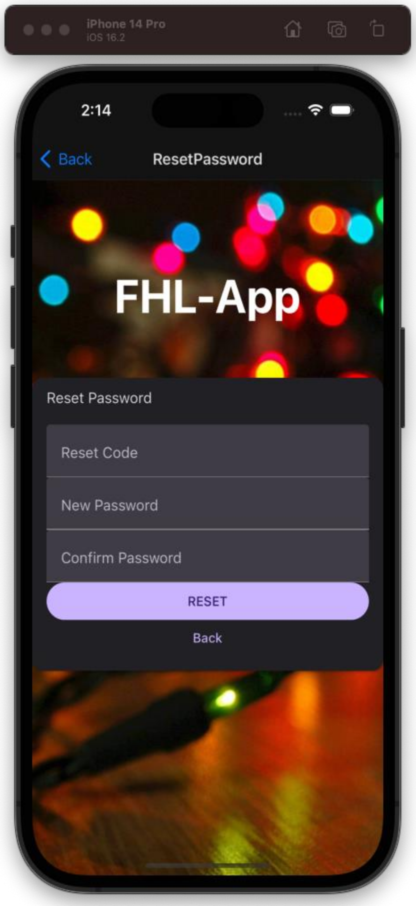
A register screen is a screen where users create their account for the app by entering their name, email, and password. Once the register button is pressed, the newly created account will be saved on the database, which will grant the authorization to the user for logging in.



5.2.4.3 Forgot Password Screen

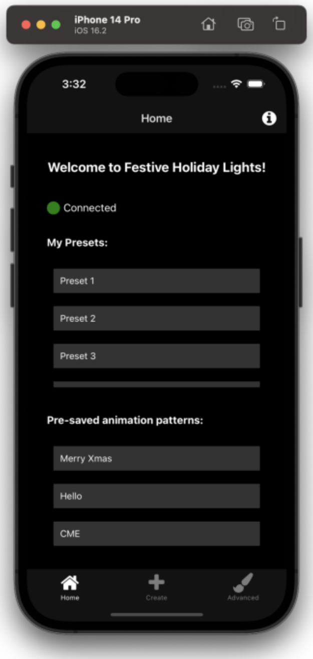
The "forgot password" screens allow users to reset their password in case they forgot their current ones. Once the users press the forgot password button on the login screen, they will first get directed to the screen on the left below, where they will be asked to provide their email to receive the reset code. After providing the email and clicking the "Send Reset Code" button, the database will check if the email exists in it, the database will send the reset code to the email. The user will then be directed to the second screen below, where they will type in the received reset code and reset password, and click a reset button to successfully reset their password. In case the user does not receive any reset code, meaning their email does not exist in the database or not valid. Therefore, they should either check whether the email they used for their account is valid, or they should create a new account. Although our group has included all the screens and the transition between them, the functionality has not yet been implemented due the lack of time in the project.





5.2.4.4 Main Menu

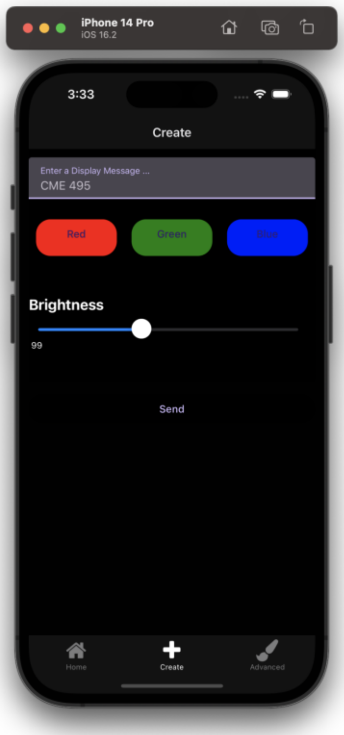
The main menu of the software is designed to account for user friendliness and streamlined control. On this screen, users can view and run the presets they have created. Through the main menu the user can easily modify grid design by choosing a pre-set and import the design by clicking run. The main menu also contains designs stored in the database shown in Figure 8. There are five pre-animated patterns available for all users. Users can press the RUN button to play these patterns. With this design, the user would only need around 2- 3 clicks to change the design in the main menu. This screen provides users with a quick and easy way to access and modify their presets, as well as enjoy pre-made animations without the need for extensive customization.



Welcome to Festive Holiday Lights!

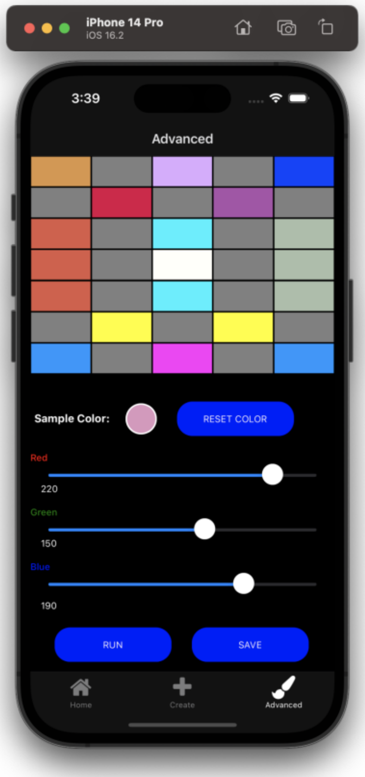
5.2.4.5 Customization Menu

In the customize menu, the user can write the message they want to be displayed in the text box at the top of the screen. They can also specify the color of the message to be displayed. If the message is 3 characters long then the message will be stationary in the grid, otherwise, the message will automatically be cycled across the grid.



5.2.4.6 Advanced Customization Menu

The advanced customization menu allows the users who want to deeply customize the grid to their own liking. Users can choose a led and modify its specific colours. They can also individually save a frame which allows the design to be animated. The advanced screen allows users to individually select LEDs on the grid and set their color using sliders for Red, Green, and Blue values. A sample color circle displays the colour set up by an user based on the 3 sliders values. A reset color button clears the selections. Users can save their pattern design to the database and display it on the home screen, or simply run the pattern using the run button. This screen is ideal for users who want complete control over the colors and layout of their LED grid.



5.2.4.7 Backend Design

To account for maintainability and reusability, the phone app will design using object-oriented programming. Launching updates and adding new features will be much more organized as the app comes into launch. With a model view control (MVC) architecture program files will be much more organized as front end, back end, and interfaces will be grouped in their own categories.

As for the database, mongo will be used for the design since it works well with React JS. Mongo allows for storage in the cloud. This would work well with the design uploading mechanism for the community page age the saved pre-sets of the app.

One critical aspect to note about the app is that some designs are animated, and some are non-animated. These two designs are separated into categories as seen in figure#9 where an animated design is composed of one or more non-animated designs. The user will be able to create their account and store information such as pre-sets, emails, and passwords. The user has access to 3 design storage locations: pre-sets, pre-saved, and community. Here the user can launch the design which is then imported to the display class and converted into the code that the microcontroller can interpret.

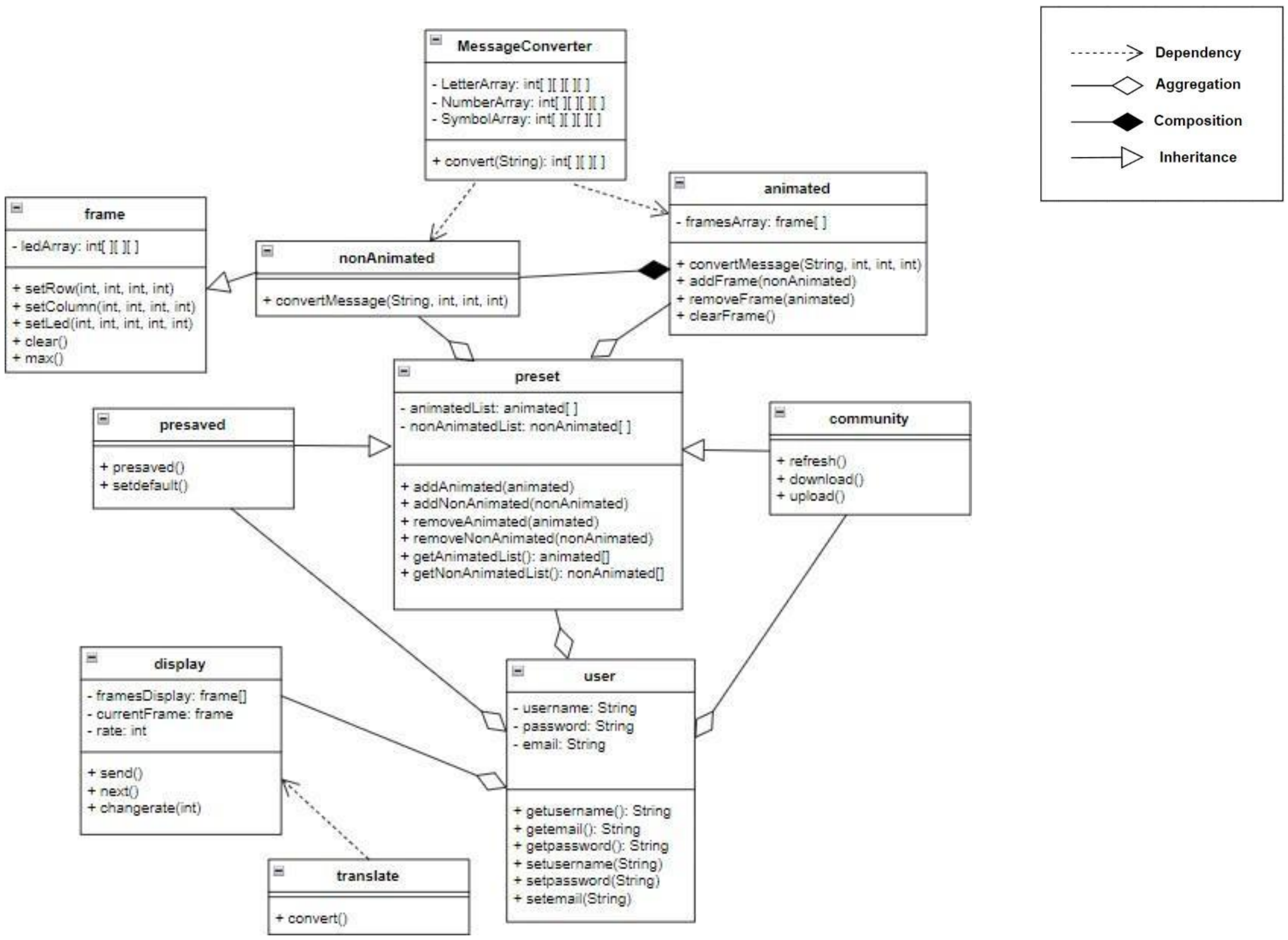


Figure17 : Object-Oriented Design of the Holiday Festival Light App

5.3 Interfaces

The system uses two major interfaces: Wi-Fi and wires. As for the Wi-Fi, the antenna of the microcontroller would need the modem/router to be in range for the connection to be accessible. Based on the dimensions that were measured for the design, the wire required for the product would need to be 16-AWG to account for the voltage drop across the range of the wire from the wall plug to the end of the grid. To increase the longevity of the cables used in the design, there will be multiple hooks in the grid, and it is encouraged that all hooks are to be used.

5.3.1. Grid Dimensions

For holiday festival lights, the ideal targeted audience are distant from the lights. People that are ideally across the road or 1-2 blocks away. The purpose of the light is to attract, pique their interest, and encourage curiosity. With this idea in mind, the chosen length 10 cm between each led block was chosen to give a better picture for those that are far from the light and to prevent interference between each light which would make the display incomprehensible to the viewer. The length of the wire from the grid to the wall plug is set to be around two meters. This allows the users to mount the holiday lights at a higher elevation if they desire.

5.3.2 Power Supply

As for the power requirements, the RGB LEDs and the microcontroller block require under 5 Volts to fully function. With a seven by 17 grid design and each red, green, and blue using 18 mA at full brightness, the system would need 6.6 Amps for the LEDs. The microcontroller would require around 1 Amp to account for dynamic displays that require a lot of switching. Although the Ideal power supply would be 5 V and 9 Amps, the chosen power supply is the LPV-100-5 which has an Ampacity of 12 Amps and meets the criteria and the budget requirements [19].

6. System Test Plan

6.1 Tests and Requirements Tested

Table #3 correlates each test plan with its corresponding requirement that is being evaluated. Section 5.2 covers the procedures of each test case and its evaluation to determine whether the test passes or fails.

Test Identification	Requirement Tested	Description
TEST.01	R.2	Measure the Voltage drop based on maximum current draw controlled
TEST.02	R.3	-Dust Tight: Sprinkle flour on it -Waterproof: Spray water on it
TEST.03	R.4	-Test all possible ASCII symbols -Verify the number of LED in Rows (7) + Columns (5)
TEST.04	R.5	Plug into the wall and check for constant 5V DC
TEST.05	R.6	Test microcontroller functionality in peak Saskatchewan temperature by exposing the device outside for at least 8 hours
TEST.06	R.7	Run software and reconfigure light design five times without adding new designs. (Postman Preferred)
TEST.07	R.8	Reconfigure the design from the main menu and count the number of clicks to change the light pattern
TEST.08	R.9	Install the app on both android and IOS.
TEST.09	R.9	Test the Home and Create Screen of the mobile app
TEST.10	R.9	Test the Advanced Screen of the mobile app
TEST.11	R.10	Perform several cost analyses between multiple materials
TEST.12	R.11	Use a light meter to determine proper exposure
TEST.13	R.12	Test for antenna range in the microcontroller
TEST.14	R.13	Test for durability – Hang design for at least 8 hours

Table 3: Test plan for the requirements

6.2 Description of Test Plans and Result

The tests described below show that test plans. Indicating the required equipment, the procedure to conduct the test, and the expected result. These test plans are used to create a replicable process of measuring certain aspects of the final product that needs to meet the general requirement criteria.

TEST.01

Equipment	Procedure	Expected Result
- led **- driver** **- microcontroller** **- laptop** **- power supply** **- multimeter** **- led datasheet**	1. Connect the LED to the driver. 2. Connect the microcontroller to the driver. 3. Connect the led and the microcontroller to the power supply. 4. Program the microcontroller in the laptop to run the led at full brightness. 5. Measure the voltage drops of each red, green, and blue led. 6. Refer to the RGB LED datasheet and calculate the ratio between the measured forward current and the maximum forward current.	The ratio between the measured forward current from the graph and the maximum forward current of the LED is 0.8 or higher.

Table 4: Testing to ensure that 80% of the LED's maximum brightness is utilized in the final product.

Actual Result: For this test, we ran the LEDs at maximum brightness (white) and measured the current draw caused by the driver. Each LED was measured to be around 55 mA was what we expected based on the driver's datasheet. With the maximum current draw of 20 mA on each diode according to the LED’s datasheet, the maximum for each LED should be around 60 mA. Calculating the ratio between the values, we were able to utilize (55 mA/60 mA) 91.7% of the LEDs maximum brightness meeting the efficiency requirement for this design.

TEST.02

Equipment	Procedure	Expected Result
- Oscillating Spray **- Flour** **- Festival holiday lights prototype**	1. The prototype is to be subjected to an oscillating spray for a minimum of 10 minutes. 2. Sprinkle some flour on the prototype and let it be submerged for at least 12 hours.	An item passes when there is total dust ingress protection with no harmful effects. The item is protected against ingress from low-pressure oscillating jets directed from any angle if there is limited ingress with no harmful effects.

Table 5: Verify the IP64 rating of our design as being dust-tight and waterproof.

Actual Result: For this test, the final product was not able to reach the point where the heat shrink tubing was applied. Although it is crucial to note that the width of the printed circuit board should have a maximum size of 0.5 cm to allow the heat shrink tubing to cover the body of each LED. Nevertheless, everything should be containable within a heat shrink tubing.

TEST.03

Equipment	Procedure	Expected Result
- 7 by 5 grid of led blocks **- microcontroller** **- laptop** **- power supply**	1. Connect the microcontroller to the grid and the power supply. 2. Connect the seven by 5 grid to the power supply. 3. Program in the laptop to cycle through every single ascii character 4. Visually verify if every single ascii character is readable/comprehensible by looking at the seven by 5 led.	The festive lights are expected to display each ASCII character and have every single character be readable in a seven by 5 grid.

Table 6: Verify that the festive holiday lights can run all ASCII symbols.

Actual Result: In the end, our led ratio was able to display most of the ASCII letters, except the "@", single and double quotation mark. The display of the "@" was mainly due to the max dimensions of the letter grid and a 5 by 7 grid of LEDs is simply not enough. Both quotations were mainly because the case statement in the C++ code cannot encapsulate the quotes within quotes.

TEST.04

Equipment	Procedure	Expected Result
- Power Supply **- Multimeter** **- Wall Plug**	1. Solder the wall plugs to AC connection of the power supply. 2. Using the multimeter, measure the DC voltage difference between the power and ground wires.	The expected measured dc voltage should be a constant 5 Volts.

Table 7: Verify for 120 Vac power supply compatibility.

Actual Result: By attaching the electrical plug to the end of the power supply, we were able to measure the DC voltage on the other end with a multimeter with the 120V electrical outlets. The measured value was 5 V which met the expected result.

TEST.05

Equipment	Procedure	Expected Result
- Vertical Outdoor **- Thermometer** **- Microcontroller** **- Power Supply** **- Led Block**	1.Run a design in the microcontroller. 2.Use the thermometer to measure the surrounding temperature. 3.Run the device for 2 hours within a temperature range of -35 to – 40 degrees. Run this experiment for a total of two samples. 4.Run the device for 2 hours within a temperature range of 35 to 40 degrees. Run this experiment for a total of three samples.	The MCU chip should fully operate within the 2-hour period of each of the three samples for the hotter and colder environmental temperature.

Table 8: Testing to check whether the microcontroller works well at an outside temperature range of -40 to +40 degree Celsius.

Actual Result: Although the grid prototype was not able to be tested outside due to its fragility, we were able to test the functionality of the microcontroller on its own under -40 degree weather conditions for 8 hours. This does not confirm the complete durability of the grid but it is important to note that the microcontroller does meet the temperature requirements.

TEST.06

Equipment	Procedure	Expected Result
- Phone **- Phone app**	1.start up the app. 2.Create and save profiles in the app. 3.Once a profile is made, go to the main menu, and verify whether five or more design are already stored in the pre saved designs lists	Five or more designs should already be visible in the main menu screen without having to create a new design.

Table 9: Testing the festive lights software and reconfiguring light design 5 times without adding new designs

Actual Result:

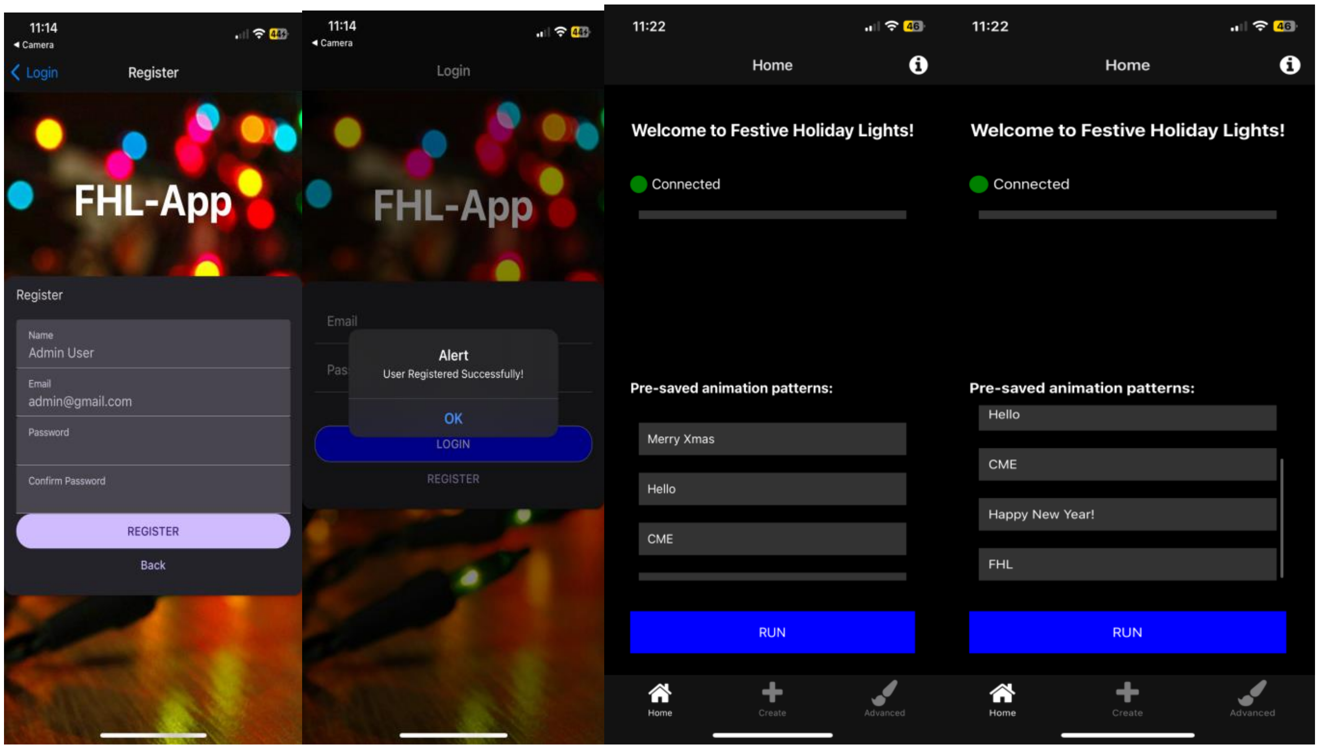
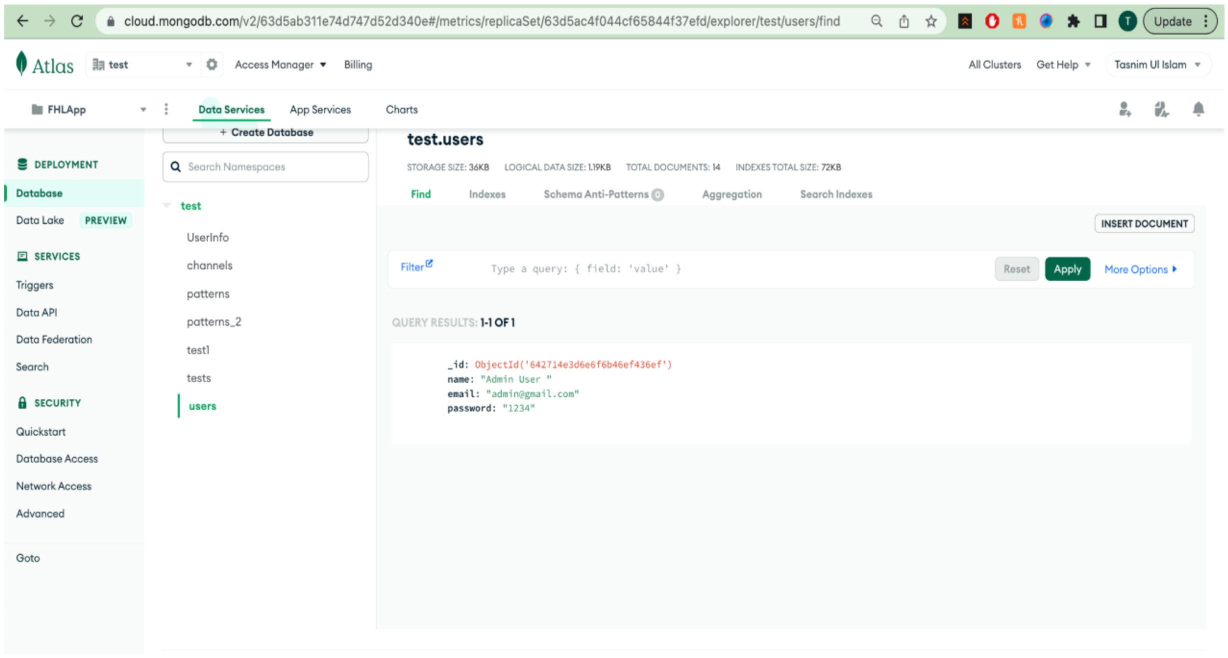


Figure 18: Phone App Menus illustrating the register screen, register pop up, Home Screen



```
_id: ObjectId('642714e3d6e6f6b46ef436ef')
name: "Admin User "
email: "admin@gmail.com"
password: "1234"
```

Figure 19: MongoDB Cloud Database tables

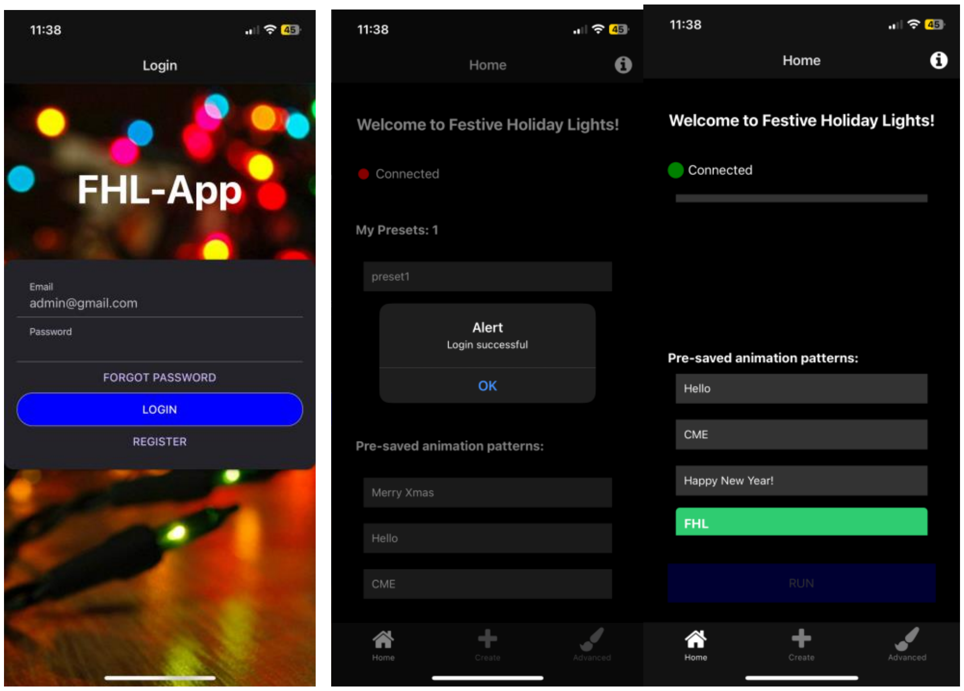
The images shown in Figure 18 and 19 depict a user successfully logging into the festive holiday lights phone app,next we see the homepage of the phone app which has 6 pre-saved animation patterns that the user could use to get familiar with the app. Lastly, we have the cloud database that stores all the login information of the user.

TEST.07

Equipment	Procedure	Expected Result
- Phone **- Phone app**	1.Start up the app. 2.login or create a user profile. 3.from login, go to the main menu. 4.In the main menu, count how many click the program takes to reconfigure the design of grid.	The minimum number of clicks from the main menu, should be within 3-6 clicks.

Table 10: Test Reconfiguring the design from the main menu and count the number of clicks to change the light pattern.

Actual Result:

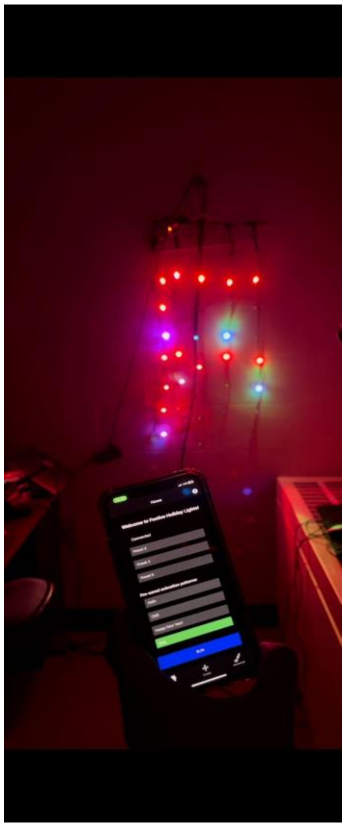


11:38
Login
FHL-App
Email
admin@gmail.com
Password
FORGOT PASSWORD
LOGIN
REGISTER

11:38
Home
Welcome to Festive Holiday Lights!
Connected
My Presets: 1
preset1
Alert
Login successful
OK
Pre-saved animation patterns:
Merry Xmas
Hello
CME
Home Create Advanced

11:38
Home
Welcome to Festive Holiday Lights!
Connected
Pre-saved animation patterns:
Hello
CME
Happy New Year!
FHL
RUN
Home Create Advanced

Actual Grid results:

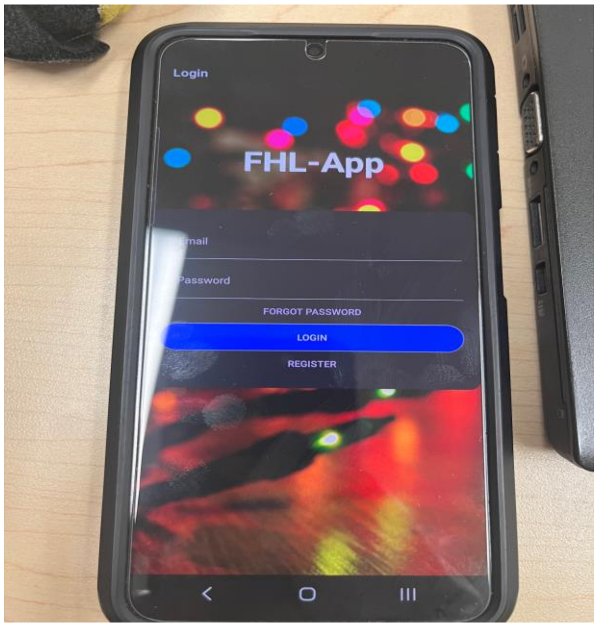


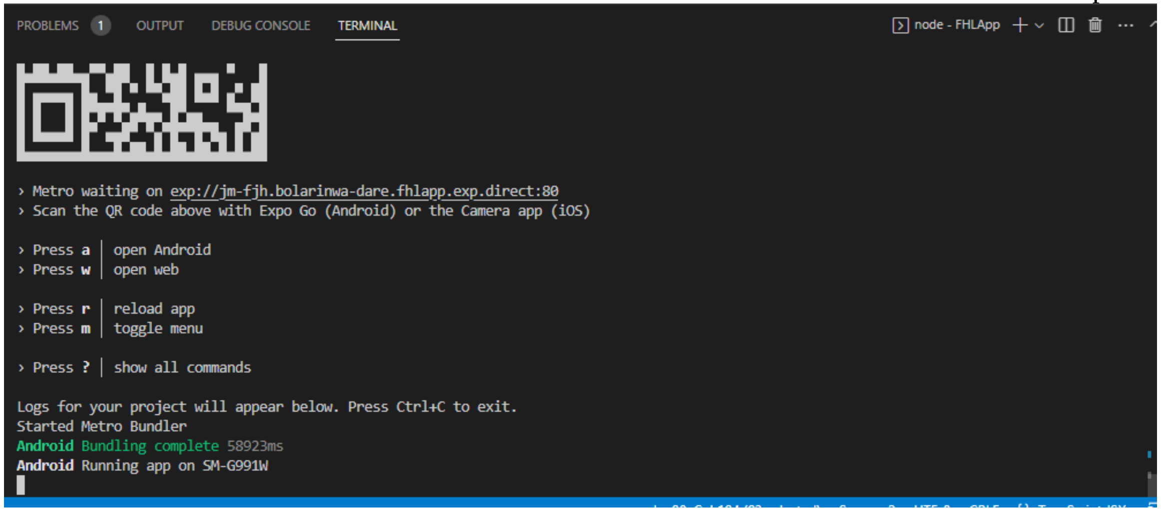
TEST.08

Equipment	Procedure	Expected Result
- Android Phone **- IOS Phone**	1.Android ... 2.Register the IOS device on XCode/App store before deployment. 3.Then use TestFlight and create a new Apple distribution certificate. 4.Archive and upload the build and distribute the app to the group via email or a public link.	The beta testers can successfully download and install the app on their respective devices.

Table 11: Test to check if the app is installed correctly on mobile devices.

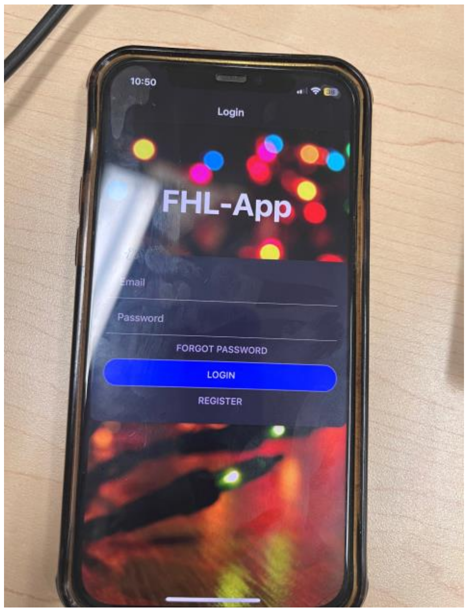
Actual Result:

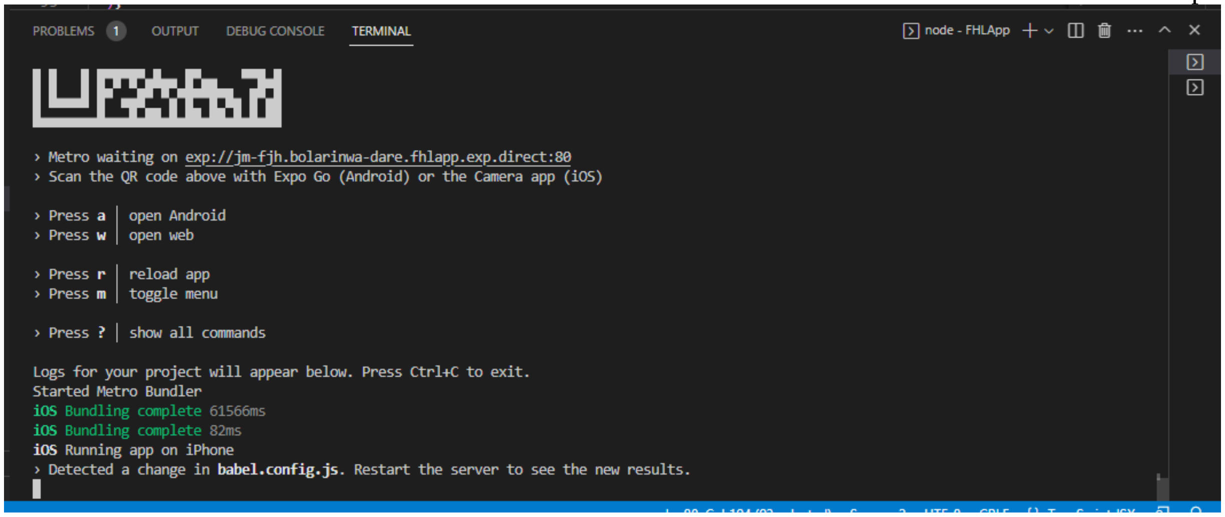




```
› Metro waiting on exp://jm-fjh.bolarinwa-dare.fhlapp.exp.direct:80
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Press a │ open Android
› Press w │ open web

› Press r │ reload app
› Press m │ toggle menu

› Press ? │ show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
Started Metro Bundler
Android Bundling complete 58923ms
Android Running app on SM-G991W
```

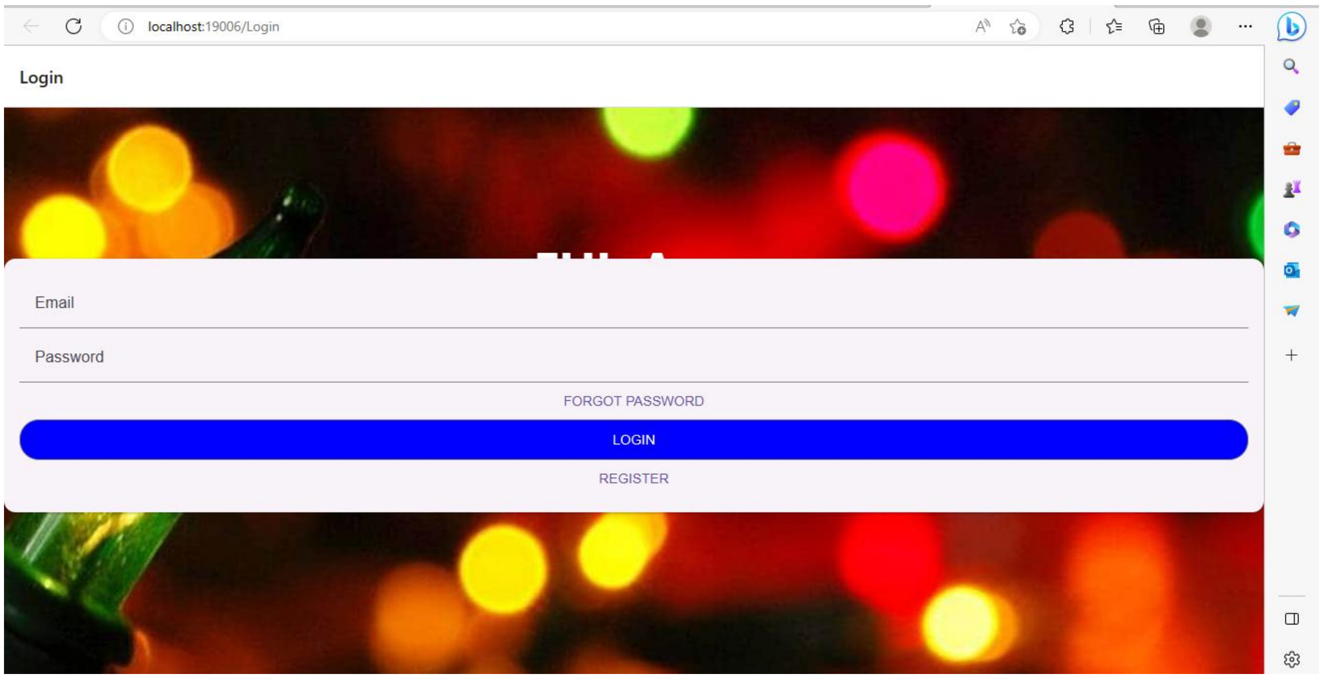
Figure 20: Proof of our festive holiday lights app running on an android phone,accompanied with the proof of its build being complete on the terminal.

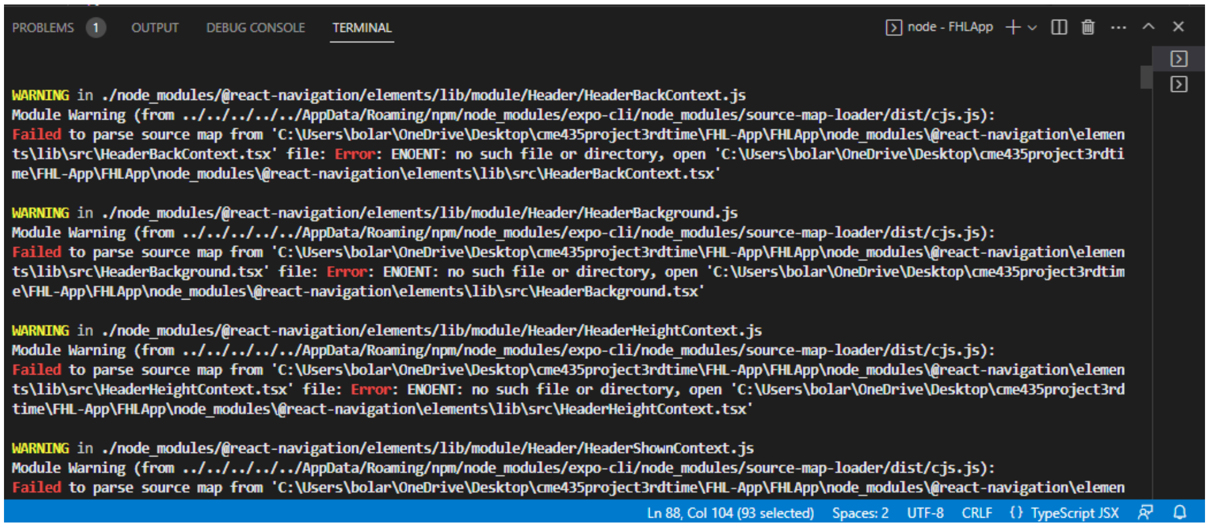




```
› Metro waiting on exp://jm-fjh.bolarinwa-dare.fhlapp.exp.direct:80
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Press a │ open Android
› Press w │ open web

› Press r │ reload app
› Press m │ toggle menu

› Press ? │ show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
Started Metro Bundler
iOS Bundling complete 61566ms
iOS Bundling complete 82ms
iOS Running app on iPhone
› Detected a change in babel.config.js. Restart the server to see the new results.
```

Figure 21: Proof of our festive holiday lights app running on an IOS phone,accompanied with the proof of its build being complete on the terminal.





```
WARNING in ./node_modules/@react-navigation/elements/lib/module/Header/HeaderBackContext.js
Module Warning (from ../../../../../AppData/Roaming/npm/node_modules/expo-cli/node_modules/source-map-loader/dist/cjs.js):
Failed to parse source map from 'C:\Users\bolar\OneDrive\Desktop\cme435project3rdtime\FHL-App\FHLApp\node_modules\@react-navigation\elemen
ts\lib\src\HeaderBackContext.tsx' file: Error: ENOENT: no such file or directory, open 'C:\Users\bolar\OneDrive\Desktop\cme435project3rdti
me\FHL-App\FHLApp\node_modules\@react-navigation\elements\lib\src\HeaderBackContext.tsx'

WARNING in ./node_modules/@react-navigation/elements/lib/module/Header/HeaderBackground.js
Module Warning (from ../../../../../AppData/Roaming/npm/node_modules/expo-cli/node_modules/source-map-loader/dist/cjs.js):
Failed to parse source map from 'C:\Users\bolar\OneDrive\Desktop\cme435project3rdtime\FHL-App\FHLApp\node_modules\@react-navigation\elemen
ts\lib\src\HeaderBackground.tsx' file: Error: ENOENT: no such file or directory, open 'C:\Users\bolar\OneDrive\Desktop\cme435project3rdtim
e\FHL-App\FHLApp\node_modules\@react-navigation\elements\lib\src\HeaderBackground.tsx'

WARNING in ./node_modules/@react-navigation/elements/lib/module/Header/HeaderHeightContext.js
Module Warning (from ../../../../../AppData/Roaming/npm/node_modules/expo-cli/node_modules/source-map-loader/dist/cjs.js):
Failed to parse source map from 'C:\Users\bolar\OneDrive\Desktop\cme435project3rdtime\FHL-App\FHLApp\node_modules\@react-navigation\elemen
ts\lib\src\HeaderHeightContext.tsx' file: Error: ENOENT: no such file or directory, open 'C:\Users\bolar\OneDrive\Desktop\cme435project3rd
time\FHL-App\FHLApp\node_modules\@react-navigation\elements\lib\src\HeaderHeightContext.tsx'

WARNING in ./node_modules/@react-navigation/elements/lib/module/Header/HeaderShownContext.js
Module Warning (from ../../../../../AppData/Roaming/npm/node_modules/expo-cli/node_modules/source-map-loader/dist/cjs.js):
Failed to parse source map from 'C:\Users\bolar\OneDrive\Desktop\cme435project3rdtime\FHL-App\FHLApp\node_modules\@react-navigation\elemen
```

Ln 88, Col 104 (93 selected) Spaces: 2 UTF-8 CRLF {} TypeScript JSX

Figure 22:.Proof of our festive holiday lights app running on the Web, accompanied with the proof of its build being complete on the terminal.

From the figures above, it is evident that the test to see if the app runs on IOS and Android devices was successful. These tests were run on Iphone 12 and galaxy X21 respectively. Also, we were able to run our application on the web as well, and the proof of this is provided in figure 13.

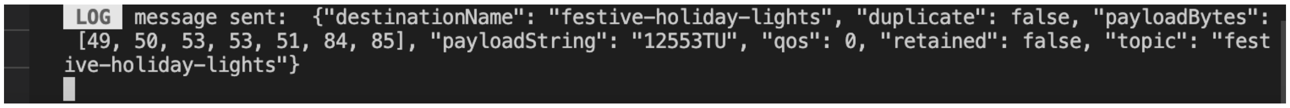
TEST.09

Equipment	Procedure	Expected Result
- Phone **-Festive Holiday Light Prototype** **- Phone App**	1. Open the app 2. On the preset screen, click the create button to move to a customized screen. 3. On the customized screen, enter the display message, select color and brightness. 4.Once everything is set up, click the Send button to show the message on the grid	The prototype should display the correct message created and sent by the app.

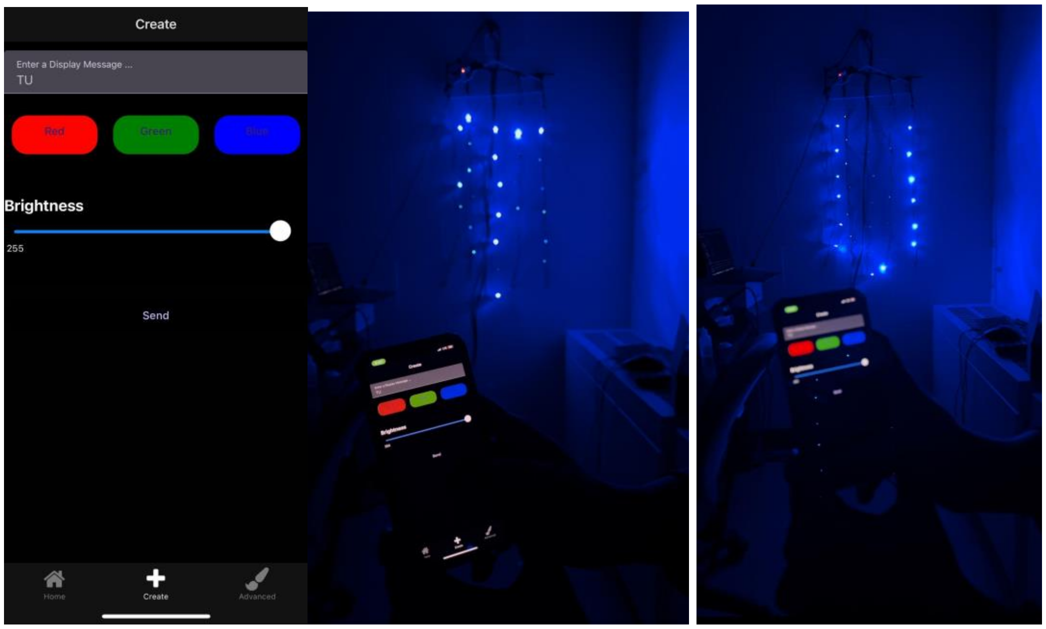
Table 12: A test to check if the basic functionality works correctly

Actual Result:

After successful login, the user navigates to the Create screen and the message "TU" is being put as an input and Blue has been chosen as the color with the maximum brightness. Upon pressing send, the message is sent as shown below. As a result, the actual grid responds by showing the letters "T" and "U" respectively making this test successful with the expected and actual results.



```
LOG  message sent:  {"destinationName": "festive-holiday-lights", "duplicate": false, "payloadBytes":
 [49, 50, 53, 53, 51, 84, 85], "payloadString": "12553TU", "qos": 0, "retained": false, "topic": "fest
ive-holiday-lights"}
```

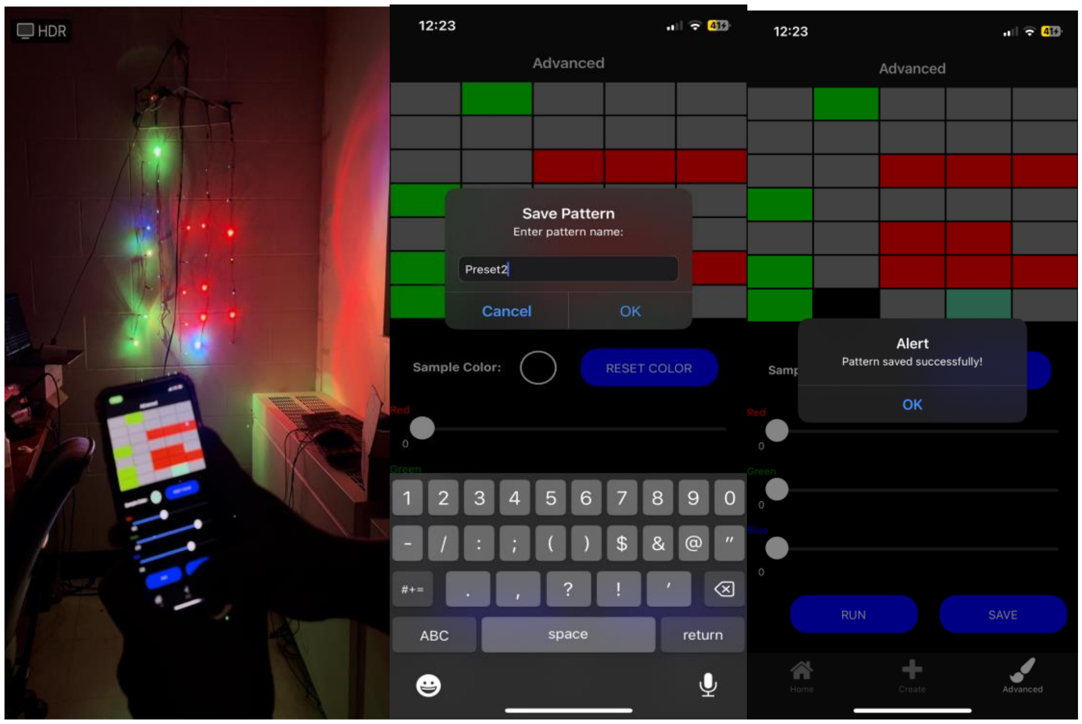


TEST.10

Equipment	Procedure	Expected Result
- Phone **- Festive Holiday Light Prototype** **- Phone App**	1. Open the app 2. Upon successful login, click the Advanced Icon button to move to the advanced screen. 3. On the advanced screen, select any individual led, and select color and brightness. 4. Once everything is set up, click the Run button to show the pattern on the grid. 5. Click the Save button to save the pattern and show it on the home screen.	The prototype should display the correct pattern created and sent by the app.

Table 12: A test to check if the basic functionality works correctly

Actual Result: After successful login, the users navigated to the Advanced, where they selected color using the slider and design their desired pattern on the grid. After that, they saved the pattern and run it by either clicking the "RUN" button or running on Main Menu screen. As a result, the saved pattern displayed correctly on the actual LED grid, as showed in the pictures below.



Save Pattern

Enter pattern name:

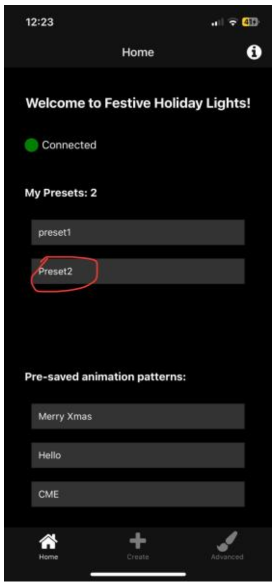
Preset2

Cancel OK

Alert

Pattern saved successfully!

OK



Home

Welcome to Festive Holiday Lights!

Connected

My Presets: 2

preset1

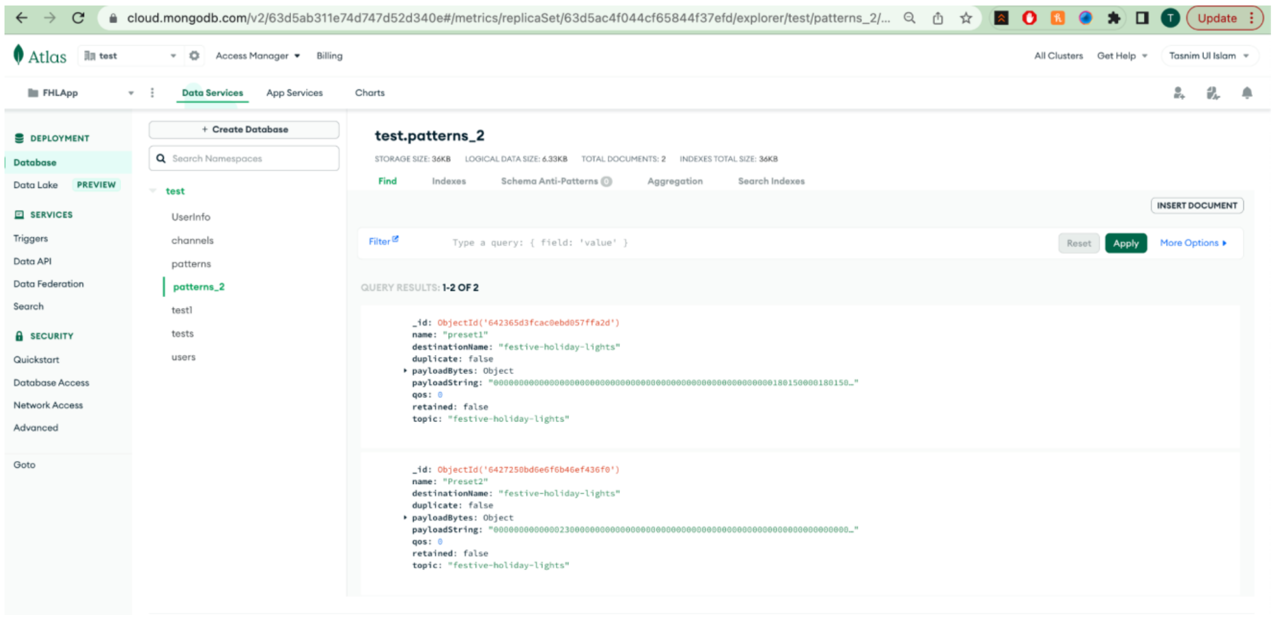
Preset2

Pre-saved animation patterns:

Merry Xmas

Hello

CME



test.patterns_2

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 6.33KB TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-2 OF 2

```
  _id: ObjectId('642365d3fcac0ebd057ffa2d')
  name: "preset1"
  destinationName: "festive-holiday-lights"
  duplicate: false
▸ payloadBytes: Object
  payloadString: "0000000000000000000000000000000000000000000000000180150000180150…"
  qos: 0
  retained: false
  topic: "festive-holiday-lights"
```

```
  _id: ObjectId('6427250bd6e6f6b46ef436f0')
  name: "Preset2"
  destinationName: "festive-holiday-lights"
  duplicate: false
▸ payloadBytes: Object
  payloadString: "0000000000000230000000000000000000000000000000000000000000000000…"
  qos: 0
  retained: false
  topic: "festive-holiday-lights"
```

TEST.11

Equipment	Procedure	Expected Result
- Bill of materials	1. Calculate and record the cost of all the products used for our project.	The expected result is that we do not exceed our budget of $250.

Table 13: Verify the total cost of the entire material purchased for the project.

Actual Result: According to table 17 below, the final cost for our product came to be $244.04, which did not exceed the budget of $250.

TEST.12

Equipment	Procedure	Expected Result
-Festive Holiday Light Prototype **- Laptop** **- light meter**	1. Connect the prototype with the laptop 2.Program the microcontroller with the laptop to turn on the LED light. 3. Use a light meter to measure the brightness of each light color.	The light meter should display at least 0.6 with every color.

Table 14: Test festive lights for proper exposure using a light meter

Actual Result: This test was not completed due to the lack of a spectrometer. A couple of issues to note for the LEDs, however, were the angle of the light dispersion being 30 degrees. This is one of the main issues with using a cheap LED. This has a strong effect on the viewpoint of the grid from the side.

TEST.13

Equipment	Procedure	Expected Result
-MCU chip **-External Antenna** **-Measuring Tape** **-Phone**	1. Connect the external antenna to the MCU chip via an IPEX cable. 2. After the connection is established, use the phone the connect to the wireless service of the MCU. 3. Measure the range with the measuring tape while moving away from the MCU and recording the value.	The range must at least reach a range of 15 m without any walls or line of sight.

Table 15: Test the range of the external antenna connected to the MCU block.

Actual Result: For this test, we tested the system in one of our group's member backyards, which has a range of around 17 to 18 meters, and the system works as expected. In an enclosed room with the school building the range shrunk to 8 to 10 meters.

TEST.14

Equipment	Procedure	Expected Result
- Grid of 7 by 17 led blocks **- A flat and stable wall** **- Wall mounts** **- Marker** **- Measuring tape**	1. Attach the wall mounts to the wall. 2. Hang the led grid for all the available hooks of the grid. 3. Mark a line on the wall to measure the initial position of the wire. 4. Wait 8 hours 5. Mark the position of the wires after 8 hours and measure the difference between the two marks	The measured distance between the initial mark and the mark after 8 hours must be under 0.125 mm.

Table 16: Verify the durability of the grid as it hangs for a specific period.

Actual Result: Although we were not able to complete the full 7 by 17 grid prototypes, we were able to test the 7 by 5 grid as it was hanging in the same position for over a week. This does not fully prove the durability of the actual product, but the design has the potential, especially when proper soldering techniques are applied.

7. Economic Analysis

7.1. Parts List and Budget

The maximum funding for our prototype is only **$250** as provided by EE/CME course, the only part which has not been researched on is the circuit board, but with how efficient the cost has been, it will be within the budget provided. Based on the mentioned consideration the following parts were selected:

Part Description	Manufacturer / Part Number	Estimated Cost ($ cad)	Amount Required
Microcontroller	ESP32-Devkit C v4	20	1
Driver	WS2811	35	120
Power Supply	LPV-100-5	46	1
Wire	LW-162-100	35	100 ft
Heat Shrink Tube	Wirefy	18	100
100 Ω Resistor	CF14JT100R	10.83	500
33 Ω Resistor	CFR-25JB-52-33R	4.88	120
100 nF Capacitor	CL05A104KA5NNNC	0.79	120
LED	100F5T-YT-RGB	20	200
Antenna	Bingfu	10	1
Printed Circuit Boards	TBD	43.56	120
Total Amount:	$244.06		

Table 17: Part Lists and cost

7.2. Overall Project Development and prototyping costs

Project Development Costs			
Item	**Hours**	**Wage($/hr)**	**Subtotal ($)**
Problem Description	20	40	800
Requirement Specification	33	40	1320
Sub-system Alternatives	153	40	6120
System Design	38	40	1520
System Test Plan	22	40	880
Project Planning	15	40	600
Total Cost ($)	11,240		

Prototyping Costs			
Item	**Hours**	**Wage($/hr)**	**Subtotal ($)**
Cost of Parts	N/A	N/A	244.04
Assembly Labour	180	40	7200
System Testing	62	40	2480
Total Cost ($)	9,880.5		

Table 18: Overall project development and Cost prototyping

Software Maintain Cost	
Item	**Cost ($)**
Hosting app on Appstore	$99/year
Hosting app on Google Play	$25

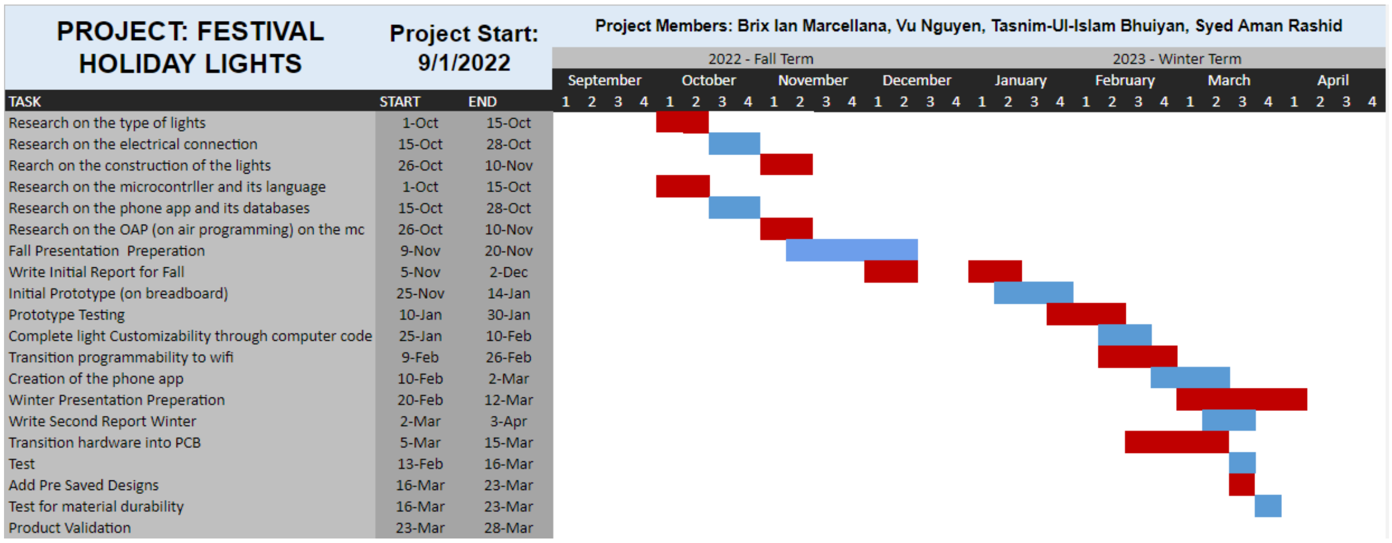
Table19: Overall Software Cost

Cost	**$21,145.5 + $99/year**

8. Planning

8.1. Gantt Chart / Project Schedule

Figure 10 shows the schedule for the updated version of group 12's project planning that helped in completing the capstone project, covering the construction of the prototype, the testing/verification, and the report/presentation management. The only task left to be done is the final report, which is indicated with the yellow highlight, every other one that has a green highlight is complete.



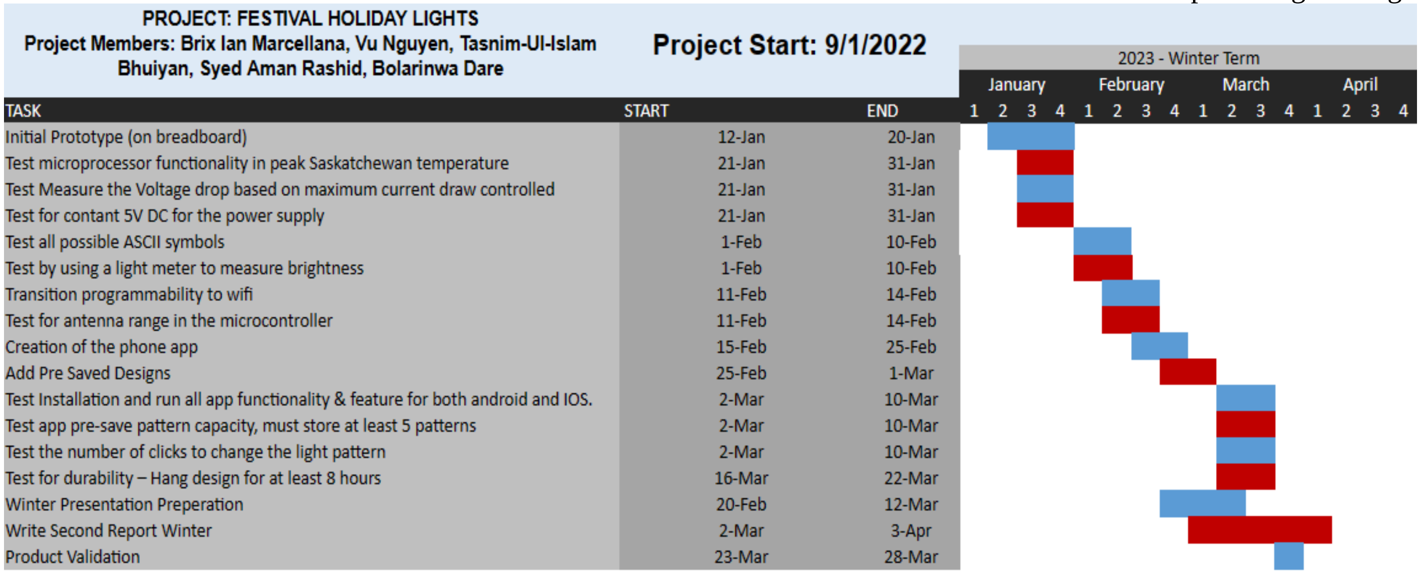
PROJECT: FESTIVAL HOLIDAY LIGHTS

Project Start: 9/1/2022

Project Members: Brix Ian Marcellana, Vu Nguyen, Tasnim-Ul-Islam Bhuiyan, Syed Aman Rashid

TASK	START	END
Research on the type of lights	1-Oct	15-Oct
Research on the electrical connection	15-Oct	28-Oct
Rearch on the construction of the lights	26-Oct	10-Nov
Research on the microcontrller and its language	1-Oct	15-Oct
Research on the phone app and its databases	15-Oct	28-Oct
Research on the OAP (on air programming) on the mc	26-Oct	10-Nov
Fall Presentation Preperation	9-Nov	20-Nov
Write Initial Report for Fall	5-Nov	2-Dec
Initial Prototype (on breadboard)	25-Nov	14-Jan
Prototype Testing	10-Jan	30-Jan
Complete light Customizability through computer code	25-Jan	10-Feb
Transition programmability to wifi	9-Feb	26-Feb
Creation of the phone app	10-Feb	2-Mar
Winter Presentation Preperation	20-Feb	12-Mar
Write Second Report Winter	2-Mar	3-Apr
Transition hardware into PCB	5-Mar	15-Mar
Test	13-Feb	16-Mar
Add Pre Saved Designs	16-Mar	23-Mar
Test for material durability	16-Mar	23-Mar
Product Validation	23-Mar	28-Mar

Figure 23: The Gantt chart of Group 12's Project Schedule that helped in completing the project.



PROJECT: FESTIVAL HOLIDAY LIGHTS
Project Members: Brix Ian Marcellana, Vu Nguyen, Tasnim-Ul-Islam Bhuiyan, Syed Aman Rashid, Bolarinwa Dare

Project Start: 9/1/2022

TASK	START	END
Initial Prototype (on breadboard)	12-Jan	20-Jan
Test microprocessor functionality in peak Saskatchewan temperature	21-Jan	31-Jan
Test Measure the Voltage drop based on maximum current draw controlled	21-Jan	31-Jan
Test for contant 5V DC for the power supply	21-Jan	31-Jan
Test all possible ASCII symbols	1-Feb	10-Feb
Test by using a light meter to measure brightness	1-Feb	10-Feb
Transition programmability to wifi	11-Feb	14-Feb
Test for antenna range in the microcontroller	11-Feb	14-Feb
Creation of the phone app	15-Feb	25-Feb
Add Pre Saved Designs	25-Feb	1-Mar
Test Installation and run all app functionality & feature for both android and IOS.	2-Mar	10-Mar
Test app pre-save pattern capacity, must store at least 5 patterns	2-Mar	10-Mar
Test the number of clicks to change the light pattern	2-Mar	10-Mar
Test for durability – Hang design for at least 8 hours	16-Mar	22-Mar
Winter Presentation Preperation	20-Feb	12-Mar
Write Second Report Winter	2-Mar	3-Apr
Product Validation	23-Mar	28-Mar

Figure 24: The Gantt chart of Group 12's Project Schedule that expands on how the Testing was planned out for the second half of the project.

8.2. Predicted Timeline

Task Number	Task	Deadline	Time required estimate (Hours)	Lead Person	Pre-requisites	Category
1	Problem definition	Oct 4	4	everyone	None	Project planning
2	Initial project plan	Oct 4	4	everyone	None	Project planning
3	Research on the type of lights	Oct 15	16	Vu	None	Research/Review
4	Research on Electrical Connection	Oct 28	16	Brix	None	Research/Review
5	Research on the construction of the lights	Nov 10	16	Vu	None	Research/Review
6	Research on the microcontroller and its language	Oct 15	24	Aman	None	Research/Review
7	Research on the phone app and its database	Oct 28	24	Bola	None	Research/Review
8	Research on the OAP(one air programming) on the microcontroller	Nov 10	24	Aman	None	Research/Review
9	Fall presentation	Nov 20	20	Tasnim	hardware/soft	Project Planning

	Preparation				ware research, initial report	
10	Write Initial report for Fall	Dec 2	35	Bola	hardware/software research	Project Planning
11	Initial Prototype(on breadboard)	Jan 20	20	Brix	hardware/software research	Prototype Design
12	Prototype testing	Jan 30	30	Vu	Initial Prototype	Prototype testing
13	Test microprocessor functionality in peak Saskatchewan temperature	Jan 31	30	Vu	Initial Prototype	Prototype testing
14	Test Measure the Voltage drop based on maximum current draw controlled	Jan 31	45	Aman	Initial Prototype	Prototype testing
15	Test for constant 5V DC for the power supply	Jan 31	25	Brix	Initial Prototype	Prototype testing
16	Test all possible ASCII symbols	Feb 10	60	Tasnim	Complete light Customizability	Prototype testing
17	Test by using a light meter to measure brightness	Feb 10	35	Vu	Initial Prototype	Prototype testing
18	Transition programmability to wifi	Feb 14	50	Bola	Initial Prototype	Prototype testing
19	Creation of the phone app	Feb 14	60	Tasnim	Software Research	Project Design
20	Add Pre-Saved Designs	Mar 1	64	Aman		
21	Test Installation and run all app functionality & features for both android and IOS.	Mar 10	60	Aman	Software Research	Project Design
22	Test app pre-save pattern capacity, must store at least 5 patterns	Mar 10	40	Tasnim	Software Research	Project Design
23	Test the number of clicks to change the	Mar 10	20	Tasnim &Aman	Software Research	Project Design

	light pattern					
24	Test for durability – Hang design for at least 8 hours	Mar 22	8	Brix	Hardware Research	Project Testing
25	Winter Presentation Preparation	Mar 17	15	Tasnim	Software Research, Hardware Research, complete Project	Final
26	Write Second Report Winter	Apr 3	40	Bola	Project completion	Final
27	Product Validation	Mar 28	15	Everyone	Project completion	Final

Table 20: Table of predicted Tasks

8.3. Actual Timeline

Task Number	Task	Deadline	Time required estimate (Hours)	Lead Person	Pre-requisites	Category
1	Problem definition	Oct 4	4	everyone	None	Project planning
2	Initial project plan	Oct 4	4	everyone	None	Project planning
3	Research on the type of lights	Oct 15	16	Vu	None	Research/Review
4	Research on Electrical Connection	Oct 28	16	Brix	None	Research/Review
5	Research on the construction of the lights	Nov 10	16	Vu	None	Research/Review
6	Research on the microcontroller and its language	Oct 15	24	Aman	None	Research/Review
7	Research on the phone app and its database	Oct 28	24	Bola	None	Research/Review
8	Research on the OAP(one air programming) on the microcontroller	Nov 10	24	Aman	None	Research/Review
9	Fall presentation Preparation	Nov 20	20	Tasnim	hardware/software research, initial report	Project Planning

10	Write Initial report for Fall	Dec 2	35	Bola	hardware/software research	Project Planning
11	Initial Prototype (on breadboard)	Feb 27	10	Brix	hardware/software research	Prototype Design
12	Prototype testing	Mar 1	25	Vu	Initial Prototype	Prototype testing
13	Test microprocessor functionality in peak Saskatchewan temperature	Feb 21	8	Vu	Initial Prototype	Prototype testing
14	Measure the Voltage drop based on maximum current draw controlled	Mar 16	1	Aman	Initial Prototype	Prototype testing
15	Test for constant 5V DC for the power supply	Mar 4	2	Brix	Initial Prototype	Prototype testing
16	Test all possible ASCII symbols	Mar 11	25	Tasnim	Complete light Customizability	Prototype testing
17	Test by using a light meter to measure brightness	NIL	0	Vu	Initial Prototype	Prototype testing
18	Transition programmability to wifi	Mar 11	50	Bola	Initial Prototype	Prototype testing
19	Creation of the phone app	Mar 7	120	Tasnim	Software Research	Project Design
20	Add Pre-Saved Designs	Mar 25	3	Aman	Software Research	Project Design
21	Test Installation and run all app functionality & features for both android and IOS.	Mar 26	20	Aman	Software Research	Project Design
22	Test app pre-save pattern capacity, must store at least 5 patterns	Mar 27	5	Tasnim	Software Research	Project Design
23	Test the number of clicks to change the light pattern	Mar 18	1	Tasnim & Aman	Software Research	Project Design
24	Test for durability –	NIL	8	Brix	Hardware	Project Testing

	Hang design for at least 8 hours				Research	
25	Winter Presentation Preparation	Mar 16	8	Tasnim	Software Research, Hardware Research, complete Project	Final
26	Write Final report	Mar 31	12	Bola	Project completion	Final
27	Product Validation	Mar 27	1	Everyone	Project completion	Final

Table 21:Table of Actual Tasks

8.4. Summary of Total Project Hours

Group Member	**Planned hours for Fall Term**	**Actual Hours for Fall Term**
Brix Ian Marcellana	43	60.5
Vu Nguyen	39	50.5
Tasnim-Ul-Islam Bhuiyan	52	39.5
Syed Aman Rashid	52	39.5
Bolarinwa Dare	47	33.5

Table 22: Planned vs Actual hours for Fall Term

Group Member	**Planned hours for Winter Term**	**Actual Hours for Winter Term**
Brix Ian Marcellana	142	~195
Vu Nguyen	144	~171
Tasnim-Ul-Islam Bhuiyan	164	~148
Syed Aman Rashid	166	~150
Bolarinwa Dare	144	~150

Table 23: Planned vs Actual hours for Winter Term

9. Summary and Future Work

For this project, we were able to complete a significant sample of the design and test the prototype for the expected functionality as described by our client. For the design, a complete 7 by 17 grid was expected to be within budget. A 7 by 5 prototype was completed for the sample prototype which was enough to test the base functionalities asked by the client. For this prototype, most of the process was covered including the PCB design and the soldering of the entire grid. A phone app was also designed and created for the project. The prototyped phone app was able to include account functionalities for users, pre-saved designs for user-friendliness, the ability to write and display a message in a second and control every single LED contained in the grid. We were also able to develop communication format protocols that allow the phone app to communicate with the microcontroller and send the message through. Most of the requirements were able to be tested for the sample prototype. Certain requirements were not able to be tested. This includes the brightness of each LED, the weatherproof capability of the entire design, and the durability of the design. The requirements that were met included the accessibility of the app, the user-friendliness of the app, the efficiency of the components chosen, the ability to display a message from the phone app, and the ability to control every single LED in the grid to the desired color.

Going through the process of completing the sample prototype, we encounter major hurdles that prevent or hinder some requirements described for the product. The LEDs were a major component within the design, due to their cheap nature, issues with the angle of dispersion were visibly obvious. Our lack of soldering experience also hindered our objectives in terms of testing specific parts of the design. The fragility of the grid has been a major issue and progressing more into the prototype would cause more maintenance problems. Latency issues were also encountered in the phone app regarding the LED sliders. This is mainly due to the number of components within the UI that require real-time updates. The app also does give the user the ability to reprogram what internet the microcontroller connects to.

This design would greatly benefit from future improvements that address the weakness and include functionalities that add more to the quality of life for the user. The LED used in the design should be replaced with an LED that has a larger dispersion angle or use a facet to help disperse the light of the current led. Regarding the issue with the latency of the sliders, a different library is to be used for the sliders and the components that use a slower update rate. To add more help for the user, inserting a text box along with the slider in the advanced screen would greatly help users accurately adjust the brightness. Adding Bluetooth functionality that allows the user to reprogram the WiFi that the microcontroller connects to. With regards to the message format for the static mode that is transferred between the phone app and the microcontroller, the message can be shortened down to 6 digits per LED instead of 9 digits (3 digits per rgb color). By using hexadecimal, the value of 0 to 255 can be represented as 00 to FF instead which cuts the string length down by 33%. Additionally for the users that can login, ‘forget password’ functionality has not been implemented yet due to the lack of time.

Future Ideas for the App:

- Cycle Speed Control: this allows the user to manually control the speed of the message transitions in the grid.
- Frame Mechanism: allows the user to add individual frames of the grid in the advanced screen, giving the capability to display little animations.
- Community Screen: this allows users to upload their designs on the database where different users can view unique designs created by creative people and use those designs for their own.
- Connection Indicator: a quality-of-life addition that signals the user whether the microcontroller was able to connect to the Wi-Fi

10. References

[1] Annie, “How to choose the right IP rating of LED lights for interior and exterior,” *AIS*, 03- Apr-2019. [Online]. Available: https://www.aisledlight.com/choose-right-ip-rating-led- lights/. [Accessed: 01-Dec-2022].

[2] Rryan, “Average House size in Canada,” *Paradise Developments*, 29-Jun-2021. [Online]. Available: https://paradisedevelopments.com/blog/communities/average-house-size-in-canada/. [Accessed: 01-Dec-2022].

[3] “WS2811 Datasheet.” [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2811.pdf. [Accessed: 01-Dec-2022].

[4] “LEDs,” *Electronics Club*. [Online]. Available: https://electronicsclub.info/leds.htm. [Accessed: 01-Dec-2022].

[5] “Chanzon 100 pcs 5mm RGB multicolor LED diode lights common cathode(clear round transparent 3 color) 4 pin bright lighting bulb lamps electronics components indicator light emitting diodes,” *Amazon.ca: Industrial & Scientific*. [Online]. Available: https://www.amazon.ca/Tricolor-Multicolor-Lighting-Electronics-Components/dp/B01C19ENDM?th=1. [Accessed: 03-Dec-2022].

[6] “C513a-WSN-CW0Z0232 Cree led: Mouser,” *Mouser Electronics*. [Online]. Available: https://www.mouser.ca/ProductDetail/Cree-LED/C513A-WSN-CW0Z0232?qs=UHyCXFkX5EwmRsvRgDalvA%3D%3D. [Accessed: 01-Dec-2022].

[7] “ESP32 series - espressif.” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf . [Accessed: 01-Dec-2022].

[8] “ESP32WROOM32D & ESP32WROOM32U Datasheet.” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf. [Accessed: 01-Dec-2022].

[9] “ESP32-DEVKITC V4 getting started guide,” *ESP*. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html#get-started-esp32-devkitc-board-front. [Accessed: 01-Dec-2022].

[10] “Arduino® Uno R3.” [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf. [Accessed: 01-Dec-2022].

[11] T. A. Team, “Uno R3: Arduino documentation,” *Arduino Documentation | Arduino Documentation*. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3. [Accessed: 01-Dec-2022].

[12] A. Trachim, “Dart / Flutter vs. Swift / Native iOS – which one is better in 2021?,” *itCraft*. [Accessed: 01-Dec-2022].

[13] Adhithi Ravichandran|Reggie Dawson|Richard Monson-Haefel|Roland Guijt|Daniel Stern|Anthony Alampi, “Build mobile applications with react native,” *Pluralsight*. [Online]. Available: https://www.pluralsight.com/paths/build-mobile-applications-with- react-native. [Accessed: 01-Dec-2022].

[14] A. Inc., “Swift,” *Apple Developer*. [Online]. Available: https://developer.apple.com/swift/. [Accessed: 01-Dec-2022].

[15] A. Inc., “Xcode 14 overview,” *Apple Developer*. [Online]. Available: https://developer.apple.com/xcode/. [Accessed: 01-Dec-2022].

[16] “422c liquid - MG chemicals.” [Online]. Available: https://www.mgchemicals.com/downloads/tds/tds-422c-l.pdf. [Accessed: 01-Dec-2022].

[17] “422C Silicone Conformal Coating MG Chemicals UK Limited.” [Online]. Available: https://www.mgchemicals.com/downloads/msds/01%20English%20UK%20SDS/sds- 422c-l%20en%20uk.pdf. [Accessed: 01-Dec-2022].

[18] E. Inc, “Eliminate Power Supply Noise with a bypass capacitor,” *Easybom*. [Online]. Available: http://www.easybom.com/blog/a/eliminate-power-supply-noise-with-a-bypass- capacitor. [Accessed: 01-Dec-2022].

[19] “LPV-100-5: Digi-key electronics,” *Digi*. [Online]. Available: https://www.digikey.ca/en/products/detail/mean-well-usa-inc./LPV-100-5/7704981?utm_adgroup=General&utm_source=google&utm_medium=cpc&utm_campai gn=PMax%3A+Smart+Shopping_Product_Zombie+SKUS&utm_term=&productid=77049 81&gclid=Cj0KCQiAkMGcBhCSARIsAIW6d0C3c0SoJiza1wltu4C4mNhLVG-A1KfTFvMFLrZD_JTtkXSy1gaev8waArRiEALw_wcB. [Accessed: 01-Dec-2022].

[20] “Electronic Components Distributor - Mouser Electronics Canada.” [Online]. Available: https://www.mouser.ca/datasheet/2/723/HB_C513A_WSN_WSS_MSN_MSS-

3011970.pdf. [Accessed: 08-Dec-2022].

[21] “MIC2860-D Datasheet by microchip technology,” Digikey. [Online]. Available: https://www.digikey.ca/htmldatasheets/production/844965/0/0/1/mic2860-d.html. [Accessed: 07-Dec-2022].

[22] “Wirefy heat shrink tubing kit.” [Online]. Available: https://www.amazon.com/Wirefy-180- Heat-Shrink-Tubing/dp/B084GDLSCK. [Accessed: 07-Dec-2022].

[23] T. A. Team, “Learn: Arduino documentation,” *Arduino Documentation | Arduino Documentation*. [Online]. Available: https://docs.arduino.cc/learn/. [Accessed: 02-Apr-2023].

[24]“CP210x USB to Uart Bridge VCP drivers,” *Silicon Labs*, 19-Jan-2023. [Online]. Available: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers. [Accessed: 02-Apr-2023].

[25]P. J. 13 and D. J. 13, “ESP32: Connecting to a WIFI network,” *techtutorialsx*, 18-Jan-2022. [Online]. Available: https://techtutorialsx.com/2017/04/24/esp32-connecting-to-a-wifi-network/. [Accessed: 02-Apr-2023].

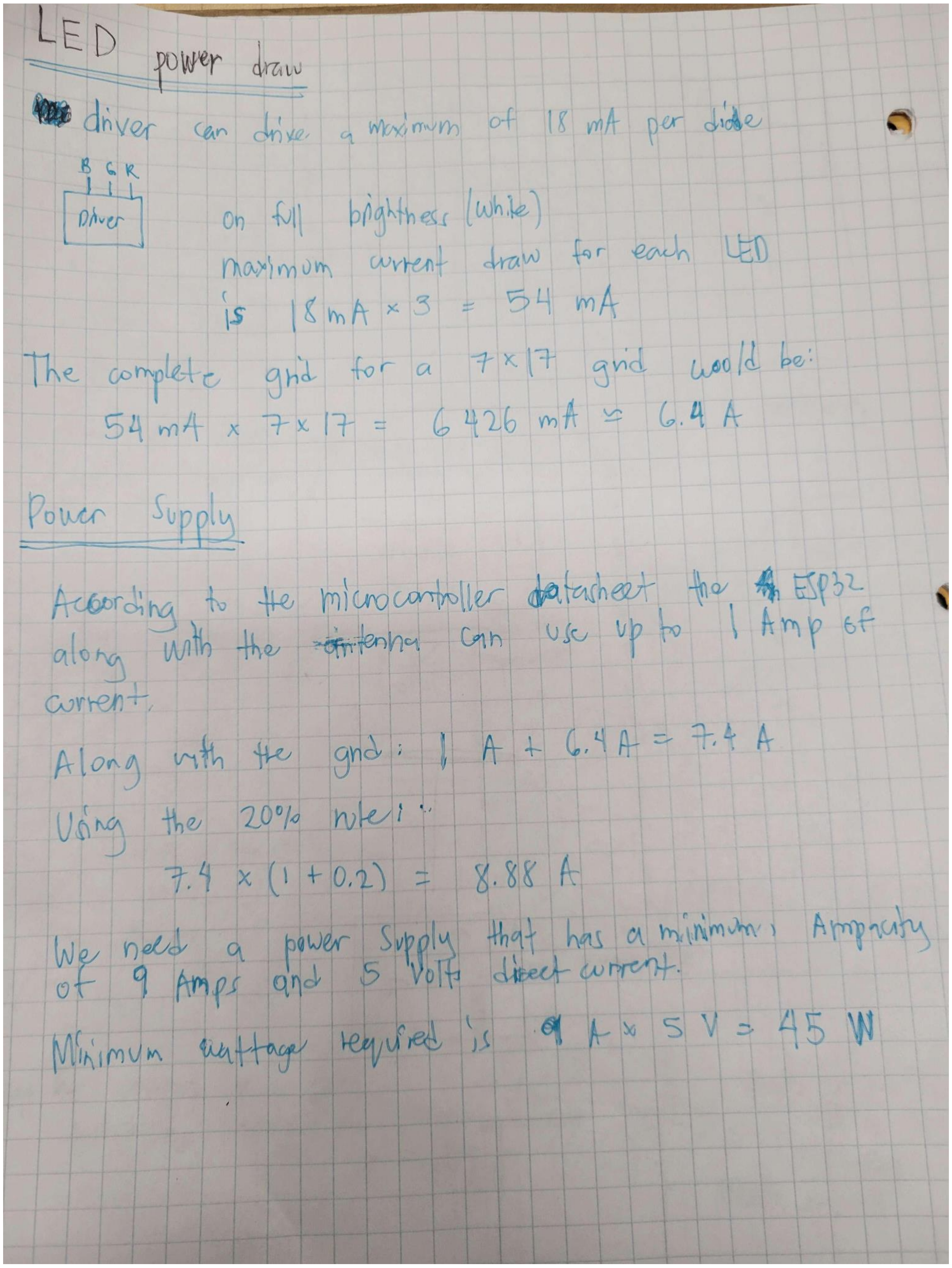
[26]V. Arakelyan, C. Lakehal, and Nick, “Esp32 MQTT client: Publish and subscribe. HiveMQ and BME280 example,” *SwA*, 14-Jun-2020. [Online]. Available: https://www.survivingwithandroid.com/esp32-mqtt-client-publish-and-subscribe/. [Accessed: 02-Apr-2023].

[27] Expo. "Expo CLI Overview." Expo Documentation, Expo, 2021, https://docs.expo.dev/workflow/expo-cli/.

[28] React Native. "Getting Started." React Native Documentation, Facebook, 2021, https://reactnative.dev/docs/getting-started.

[29]R. Gowen, “Festive Holiday Lights.” 01-Sep-2023.

11. Appendices



LED power draw

driver can drive a maximum of 18 mA per diode

B G R

Driver

on full brightness (white)
maximum current draw for each LED
is 18 mA × 3 = 54 mA

The complete grid for a 7 × 17 grid would be:

54 mA × 7 × 17 = 6426 mA ≈ 6.4 A

Power Supply

According to the microcontroller datasheet the ESP32 along with the ~~an~~tenna can use up to 1 Amp of current.

Along with the grid: 1 A + 6.4 A = 7.4 A

Using the 20% rule:

7.4 × (1 + 0.2) = 8.88 A

We need a power supply that has a minimum Ampacity of 9 Amps and 5 Volts direct current.

Minimum wattage required is 9 A × 5 V = 45 W