University of Moratuwa
Faculty of Engineering
Department of Electronic & Telecommunication Engineering

EN2160 - Electronic Design Realization
Individual Project
Project Report - Smart LED studio light system

Index No.	Name
200222K	Himeka S.H.D

Abstract

In this report there is a detailed description of building a smart LED studio light which is useful for photography and videography. This can give out several colors, patterns and brightness leveled light. This can be controllled wirelessly via blutooth which is a major deviation from the existing studio lights in the market.

Introduction

For this project, we were given the freedom to select any electronic product of our choice which is less than 50$ in the market. We needed to go through all the design processes, identify the existing and possible requirements, add new features, and make that product a much less expensive user-friendly marketable product. I selected a Smart LED studio lighting system for my project after observing a product that was around 42$ available in the market. As the main microcontroller an ESP32 chip was used. Several additional features were introduced, making the device much more attractive. This report includes the full design process from scratch. It mainly considered the user-centered design approach.

Contents

1 Problem Statement

Lighting arrangements are crucial especially when it comes to videography and photography. The unavailability of proper lighting makes the purpose less efficient. Apart from that professionals prefer better lighting conditions to work in. There may be instances where home or office lights do not give out that bright environment that you were expecting. Furthermore, lighting systems can become a great mode of entertainment to relax stressed user minds. The market offers several lighting devices, but most of them are expensive and do not provide all the necessary features together. So, having a specialized separate light system is a viable solution on that regards.

2 Design Approach

2.1 Conceptual Design

2.1.1 Enclosure Designs

These are the enclosure designs that were sketched at the conceptual stage.

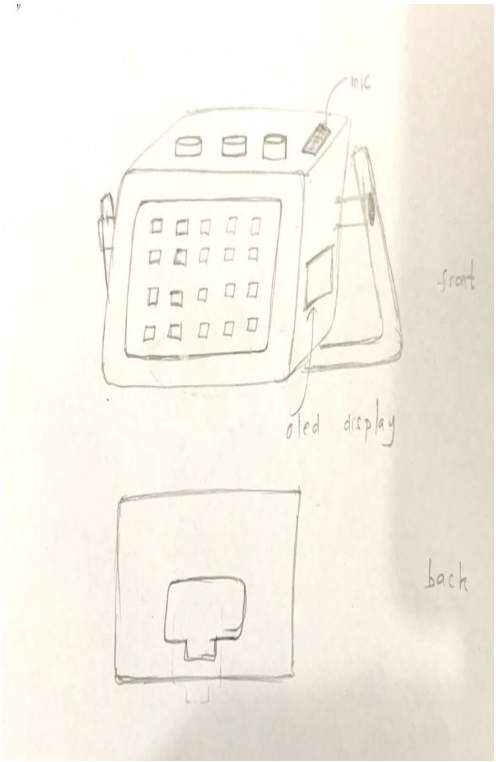


Figure 1: Design 1

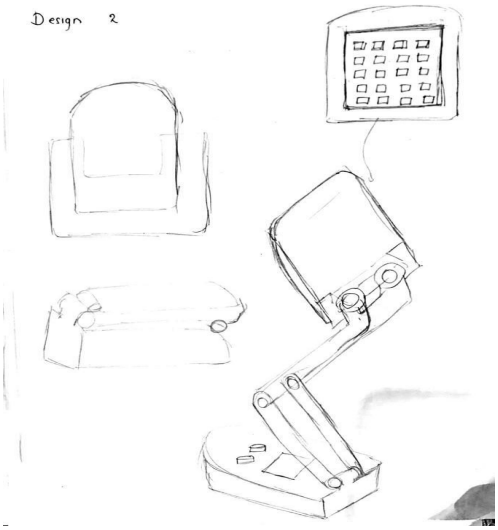


Figure 2: Design 2

Figure 3: Design 3

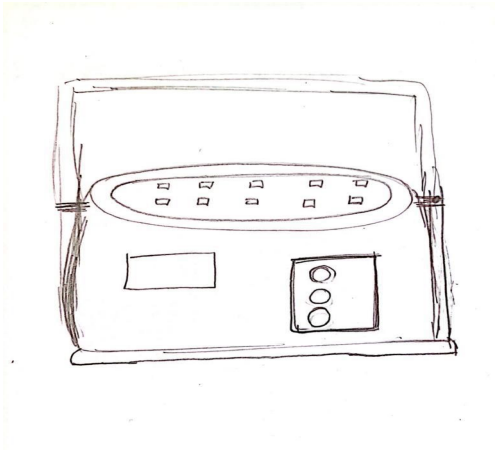


Figure 4: Design 4

briefing a little into the 4 enclosure designs;

1. This design was inspired by the stage lights which are already in the market. This design can be only kept on

tables.

2. This kind of looks like a table lamp. Can be compacted into pocket size. Portable and adjustable as per needs.

3. Can be fixed into holders on walls, tables or any surfaces. Can be kept on tables as well. An arm to carry it anywhere you want

4. Portable, but light directed only upwards in this design. Can be only kept on surfaces and not fixed anywhere.

2.1.2 Circuit Design

At the conceptual stage, different ways of led panel implementations were considered.

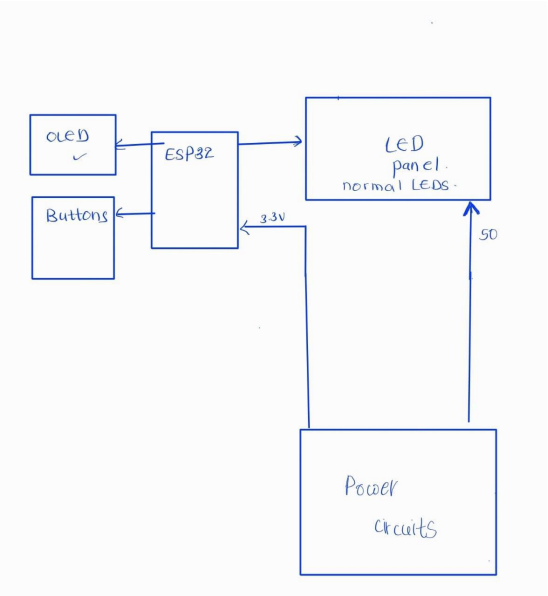


Figure 5: Design 1

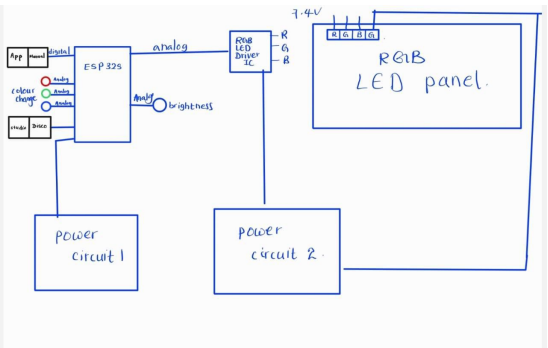


Figure 6: Design 2

Into the 4 circuit designs;

1. A basic design by using normal analog LEDs connected in series, controlled by the micro-controller ESP32. There'll be buttons for inputs and OLED for output display. Considred designing a power circuit to suit the needs.

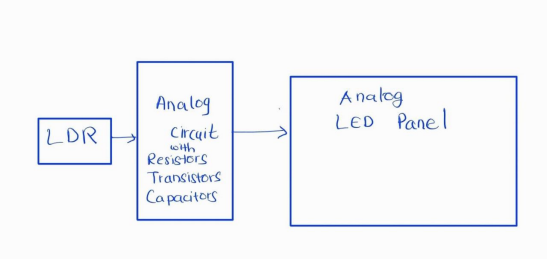


Figure 7: Design 3

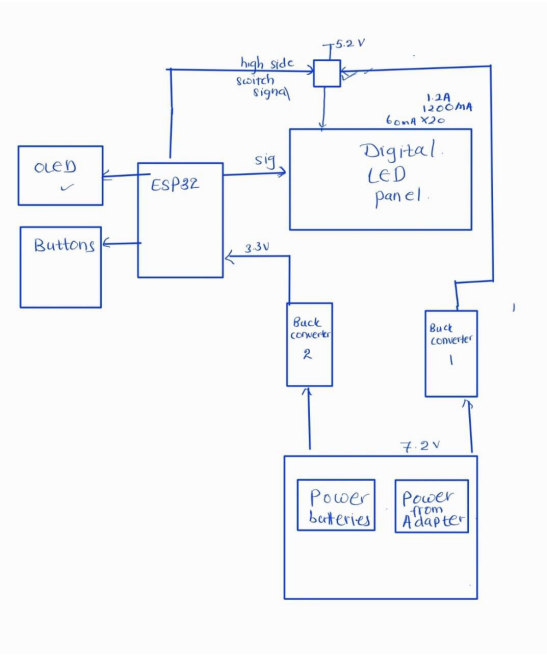


Figure 8: Design 4

2. This design compromises of 30 RGB analog LEDs controlled by the microcontroller esp32 via a LED driver IC. There'll be potentiometers for analog inputs and OLED for output display. Two separate power circuits will be designed to power up ESP32 and the LED panel.

3. LDR will controll the analog LED panal through a analog circuitry.

4. The design contains 20 RGB digital LEDs, controlled by ESP32. A single power circuit to power up both micro controller and the LED panel. Buttons for inputs and OLED for output display.

2.1.3 Evaluation of the designs

In order to select the best enclosure and circuit design out of the given, I've considered a few mostly related evaluation criteria and selected the best.

		Design 1	Design 2	Design 3	Design 4
1	Form and Aesthetics	3	4	4	2
2	Ergonomics and User Interaction	2	2	5	3
3	Material Selection and Production	1	1	4	2
4	Portability and Mounting Options	3	2	5	4
5	Cable Management	1	1	4	4
6	Heat Dissipation and Ventilation	3	1	4	3
7	Cost Effectiveness	2	1	4	3
8	Modularity and Expandability	2	2	4	4
9	Accessibility for Maintenance	2	2	5	3
10	Compliance with Safety Standards	4	2	4	4
	Total	23	18	43	32

Figure 9: Evaluation enclosure designs.

		Design 1	Design 2	Design 3	Design 4
1	Functionality and Performance	2	3	2	3
2	Efficiency and Power Consumption	1	2	2	3
3	Integration with Control System	1	3	1	4
4	Safety Considerations	1	2	1	3
5	Noise and Interference	2	1	1	2
6	Scalability and Flexibility	3	2	2	3
7	Component Selection and Reliability	2	2	4	3
8	PCB Layout and Assembly	2	2	4	3
9	Testability and debugging	3	1	3	4
10	Cost Effectiveness	4	2	4	2
	Total	21	20	24	30

Figure 10: Evaluation circuit designs

According to the above tables I've selected **Enclosure design 3** and **Circuit design 4** as the best pair to proceed with. It seems to be the most cost-effective, functional, User-friendly, and efficient duo as per the little survey I did in this regard.

2.2 User Centered Design

Although I have carried out a problem verification in the problem identifying phase, before starting off with the process, A user feedback analysis was carried out. I Presented the selected enclosure design and features (with the chosen circuit block diagram to some randomly selected users) and received their feedback.

Moreover, I could carry out a small survey to know the user view of the selected design. Here are some results collected;

1. **Undergraduate at CSE Moratuwa;**
   "Wow, I'm really impressed with the existing features of your smart studio LED light system! However, as an electronics enthusiast, I think it would be amazing if you could incorporate customizable lighting effects and integrate the lights with voice assistants like Alexa or Google Assistant. Also, adding a clipper to the enclosure design would be fantastic, as it would allow users to easily attach the light system to surfaces like tables or laptop screens."

2. **Random shop owner**
   "I must say, your smart studio LED light system has a sleek design and a user-friendly interface. It would be great if you could improve its compatibility with other smart home devices, such as smart hubs or lighting automation systems. That way, customers can seamlessly integrate your LED lights into their existing smart home setups, making their lives even more convenient. Additionally, the enhanced aesthetic look and the ability to fix it to a table lamp holder."

3. **An interior designer**
   "As an interior designer, I must say this has an incredible aesthetic appeal. To make it even more versatile, have you thought about offering interchangeable panels or covers in different colors and finishes? This would allow users to personalize the appearance of the lights and seamlessly integrate them into their home decor. It would definitely be a selling point for design-conscious customers. Additionally, the inclusion of wireless or Bluetooth connectivity for mobile app control would be a great addition."

4. **A random house-wife**
   "Your smart studio LED light system seems like a perfect solution for me. I am very interested in the energy-saving aspect of the product. Being able to reduce my electricity bill by using the smart features of the lamp holder is a huge advantage."

Given these reviews, I changed my conceptual design to fit into these feedback.

Here's the user-centered design I've arrived upto.

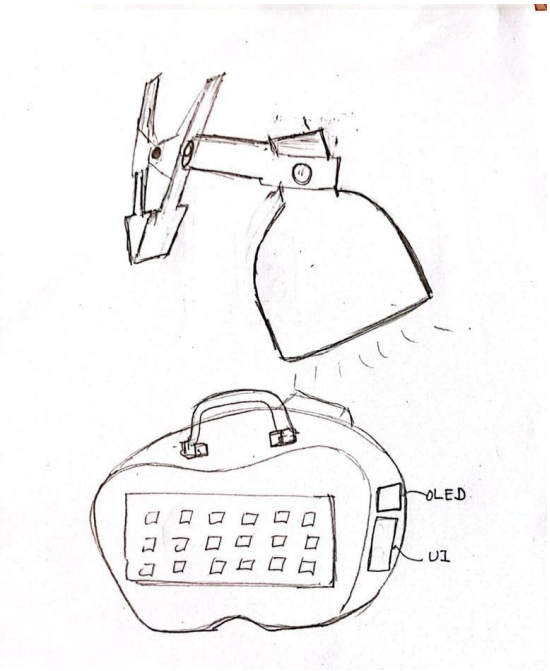


Figure 11: User Centered Design.

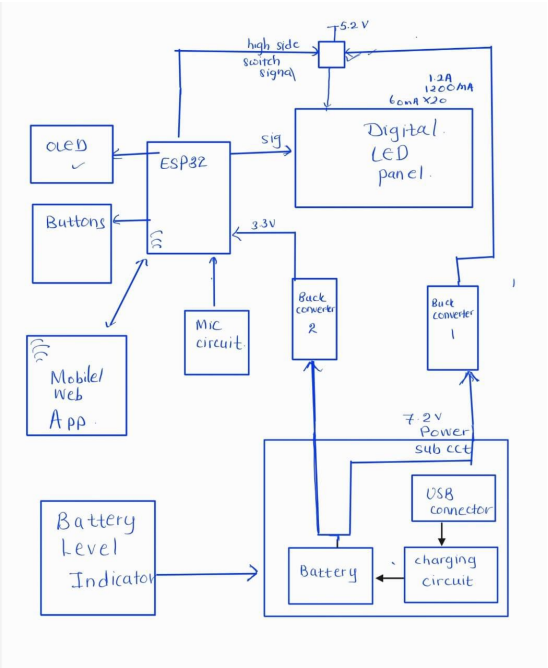


Figure 12: User Centered Design.

2.2.1 Comparison between presented existing conceptual design and the user-centered design

Enclosure design changes

- Added a clipper to get it plugged on to particular surfaces; like tables, laptop screens etc.
- Enhanced its aesthetic look
- Still can fix it to a table lamp holder.

Circuit Design Changes

- Added a mic circuit for a music visualization
- Standby and working mode included additionally; for the light to be active only when a clap is detected
- Wireless connectivity/ Bluetooth connectivity to control the system with a mobile app or web-based app.

Adding the given features, I moved on to the prototype designing stage

3 Prototype Design

The next step of the design process was to select the appropriate components and make the schematics.

3.1 Basic Component selection

The component selection was done in line with the required product specifications

- **Main Microcontroller- ESP32 Wroom 32D**
  The User centered design required a normal microcontroller with inbuilt wireless activity. (Wifi or Bluetooth). Also, given my i/o pin requirements, I decided to go ahead with the ESP32. I decided to use the chip on the schematic and use a development board for programming purposes.

Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power supply voltage	3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T	Operating temperature	–40	-	85	°C

Figure 13: Recommended Operating Conditions

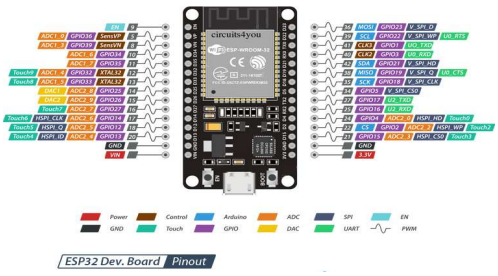


Figure 14: ESP32

- **Display- OLED**
  Compact with a 3 cm diagonal screen, the monochrome 0.96-inch AZ-Delivery OLED display is simple to view due to the panel's high contrast. The in-built SSD1306 chip individually controls each of the 128 x 64 OLED pixels that make up the display.

Figure 15: 0.96" Oled as the display

- **Digital LED panel**
  Although I've tried to go with analog rgb LEDs, I decided to use digital SK6812 LEDs because of its in built features. SK6812 is an LED with built-in control circuitry, offering consistent colors, low power consumption, and easy installation. It uses PWM technology and allows cascading multiple LEDs together.20 such LEDs coonected in series here.

Figure 16: SK6812

recommended input voltage 5V

3.2 Power Utilization

First decided to make it battery powered since a feature was given to make it portable. Then the maximum input voltage and current ratings to the components in the circuitry were identified.
For ESP32 and other components

- Voltage rating - 3.3V
- Current rating - 1.1A

For the LED Panel

- Voltage rating - 5V
- Current rating - 2A

To meet the requirements 2 separate buck converter circuits were used.

1. **Buck Converter to ESP32**
   This is a switching circuit with the IC LM38020DDA to output 3.3V to the ESP circuitry with a maximum current rating of 1100mA.

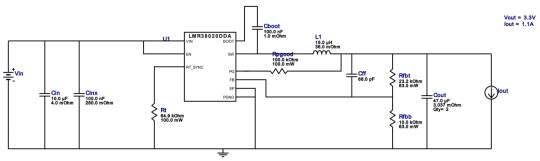


Figure 17: Power to ESP32

2. **Buck Converter to LED Panel**
   This is a switching circuit with the IC LM26003 to output 5V to the LED Panel with a maximum current rating of 2A.

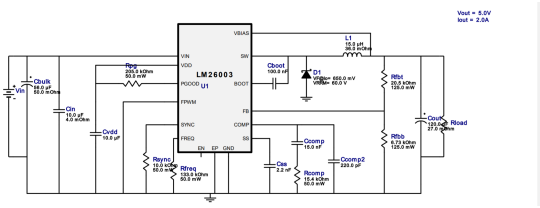


Figure 18: Power to LED Panel

for both these circuits input voltage can be in the range 6V-10V.

As the main power source I decided using 2 Li-Po batteries in series(3.7V rated 2500mA maximum current output rated). which gives out 7.4V output. To make it

more user friendly, powering through a 9V ac adapter was also included. Next to switch between two input power supplies a switching circuit was added.

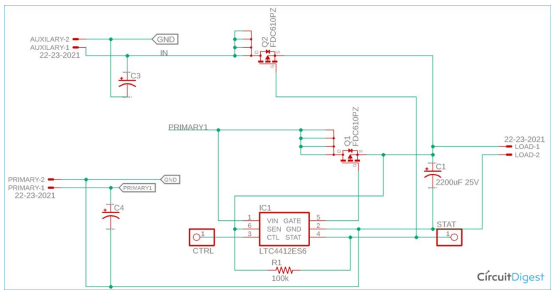


Figure 19: Power path controller circuit

The requirement of the application is to, it should always need to remain ON during power failures by using and additional power source that is available. For example, when it is powered using an adapter needs to switch to a battery or an auxiliary power supply without interrupting the operation of the circuit in the event of a power interruption.
In these above-mentioned cases, a Power Path Controller Circuit will be helpful. Basically, a power path control circuit will switch the main power of the circuit board depending on the power source available by controlling the path from where the power comes into the circuit.

3.3 LED Panel

3.3.1 Mosfet Highside switch

In order to make this more efficient, to make the LEDs work only when needed, this high side switch was included.
This circuitry uses the AP22615 IC. A input voltage in range of 3V-5.5V is accepted. The IC only gets activated with the high enable signal from ESP32.

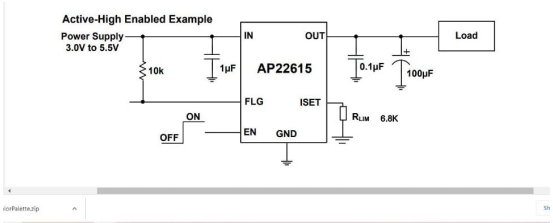


Figure 20: highside switch circuit

3.4 Schematic Designs

Attached to the appendices.
All schematics were drawn using **Altium Designer 23.3.1**
An attempt to isolate each circuitry was taken. Used zero Ohm resistors(Ferride beads) in that case.

3.5 Simulations

I used **Wokwi** online simulator to simulate the circuit before designing the PCB.A part of the circuit was simulated thereby.The simulation is shown below.

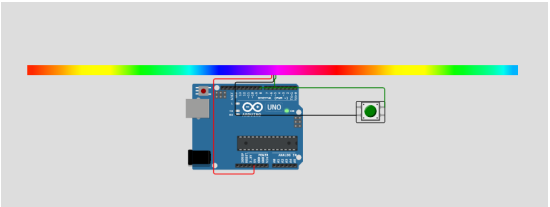


Figure 21: simulation using arduino

3.6 Initial Testing

After the circuit simulations, I implemented the circuit on the breadboard and tested it. I used a locally sourced led strip with 36 leds, using an esp32 and 3 buttons I could change colours, patterns and brightness levels.

3.7 PCB Designing and Soldering

Next, I proceeded with the PCB design. I designed two 4 layer PCBs for the main circuit and the LED Panel using Altium designer 23.3.1.
The signal quality was a main concern when pcb designing.The 2nd and 4th planes were connected to the ground. Added a separate power plane.(layer3) Traces were routed using copper pours. Some of the components were routed at the bottom layer(Layer 4) as well
PCB was printed from JLCPCB China, with a thickness of 1.6mm

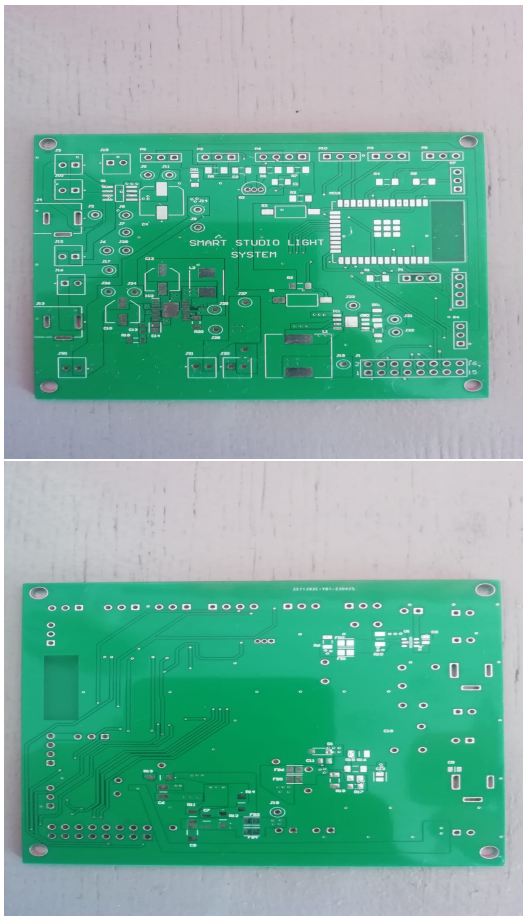


Figure 22: PCB before soldering

Then soldered the PCB with internationally sourced components from mouser.lk.

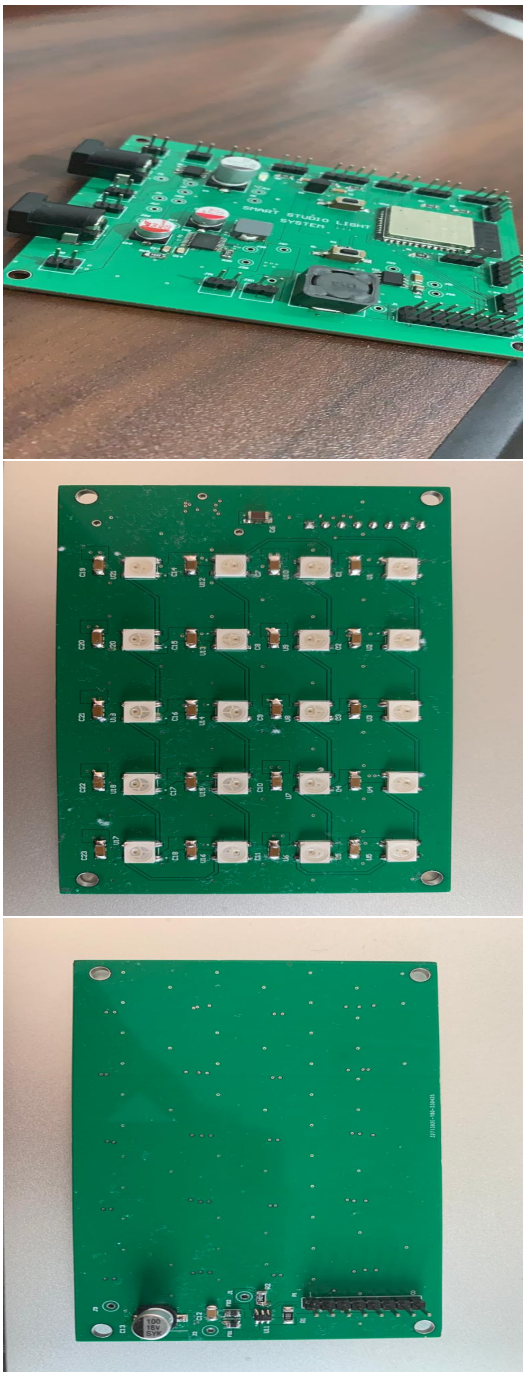


Figure 23: PCB after soldering

3.7.1 Programming ESP32 chip using a development board

For the programming purposes another ESP32 development board was used, connected it as follow and programmed through TX,RX pins on both PCB and ESP32 board.

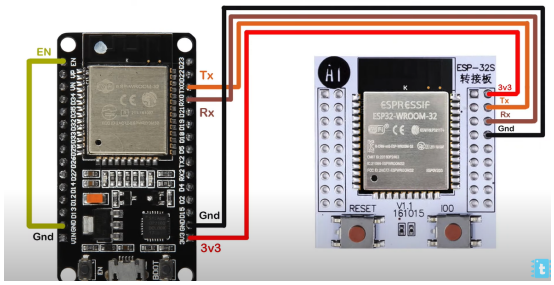


Figure 24: Programming connections.

After connecting it as above, when uploading the hold button is kept n pressed while reset button is pressed once and released.

3.7.2 PCB Testing

Tested the connections, and the working of each core components like oled, highside switch, LED panel, buttons etc. Upon confirming that there were no disconnections, and the components are working well, we tested each isolated circuits in the PCB in the lab and checked if we got the relevant outputs.

- Tested the buck converter to ESP and confirmed the 3.3V output
- Tested the buck converter to LED Paned and confirmed the 5V output
- Tested soldering 1 LED
- Tested the LED panel
- Tested the components
- Soldered the ferride beads and again tested the PCBs

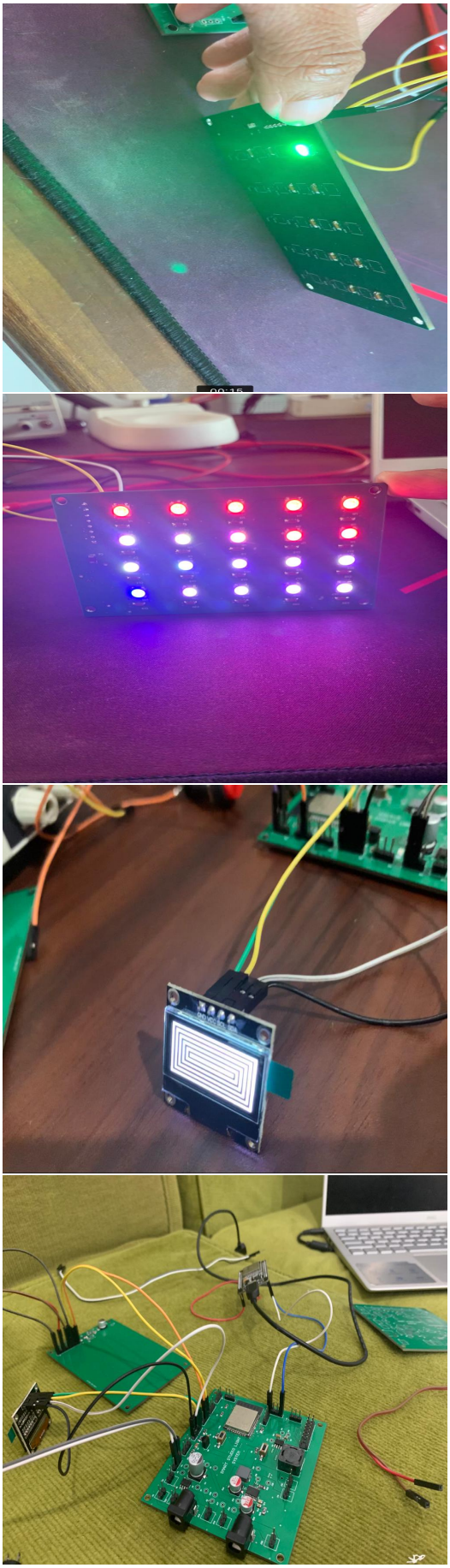


Figure 25: PCB Testings

3.8 Enclosure Design

Enclosure designing was done using **Solidworks 2021**, moldability was a key concern. Therefore designed it keeping up with the draft angles, into 2 parts as core and cavity. Used surface modelling for the designing. First, front, top, bottom sketches were imported into the solidworks, then the sketches were drawn.
Next the surface modelling was done, and then did a surface knit to make it a solid. Then the body was shelled and splitted into 2 parts.
Molding activity was implemented next. Designs are attached to the apendices.

3.9 Results

Achievable features at the prototype level:

- Changing colors, brightness and few patterns upon the button press
- Portability

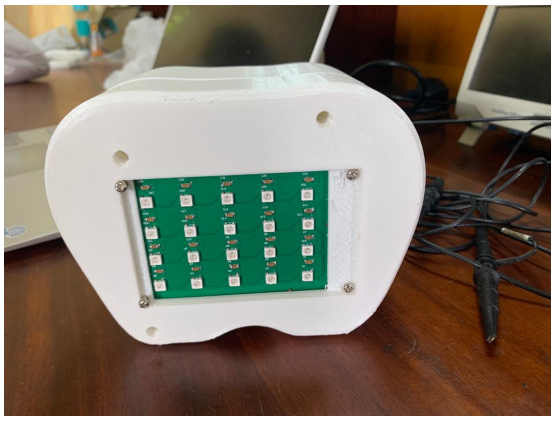


Figure 26: Prototype

4 Final Design

4.1 Changes from the prototype

As per the user inputs for the user centered design and feedback for the prototype, the final device was further improved.

- Included the Bluetooth activity
- Changed the enclosure
- Added the mic circuit

4.2 Enclosure Design

In contrast to the prototype, I've made this enclosure more compact and portable. It was 3D printed. The solidwork designs are as below.

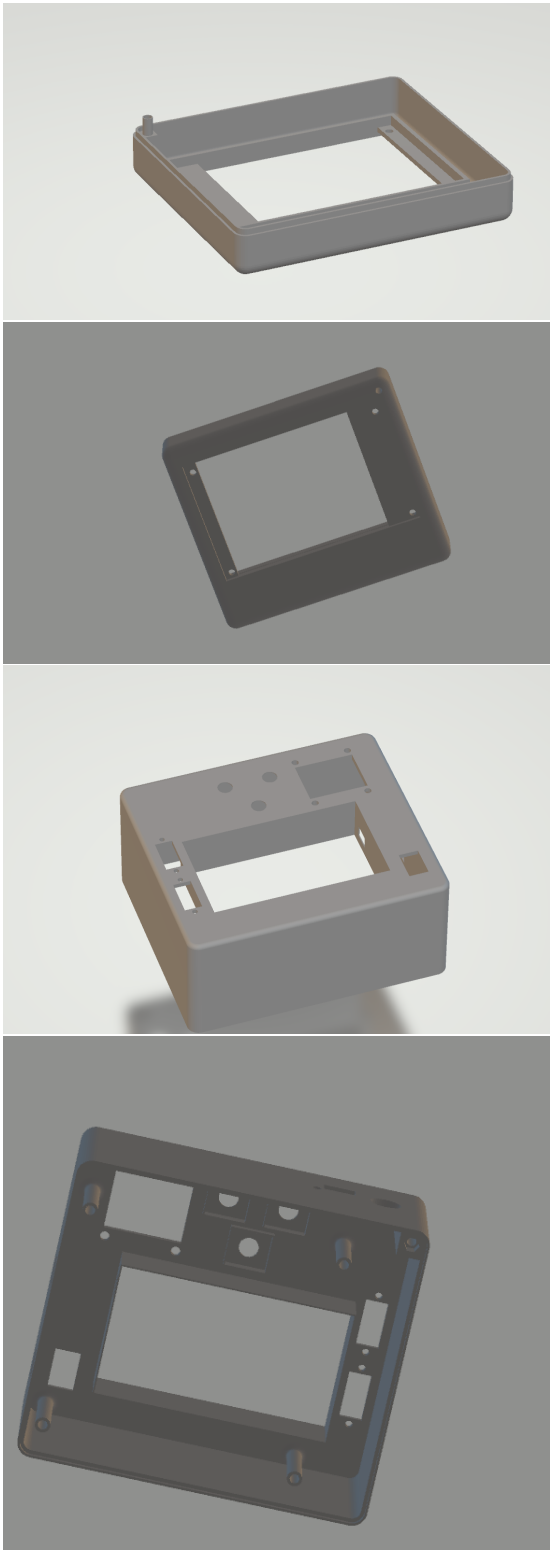


Figure 27: Final Solidwork design

4.3 Results

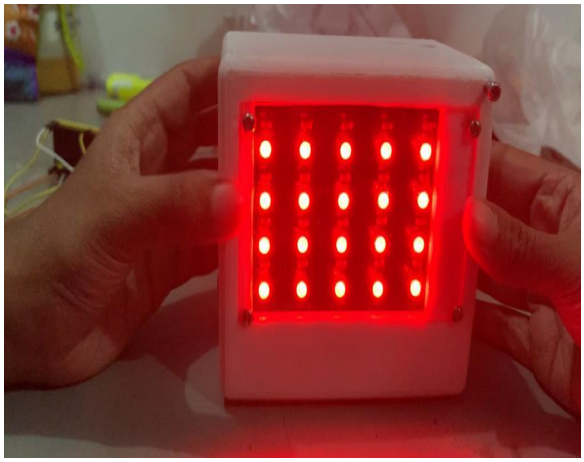
Device functions in 2 Modes;

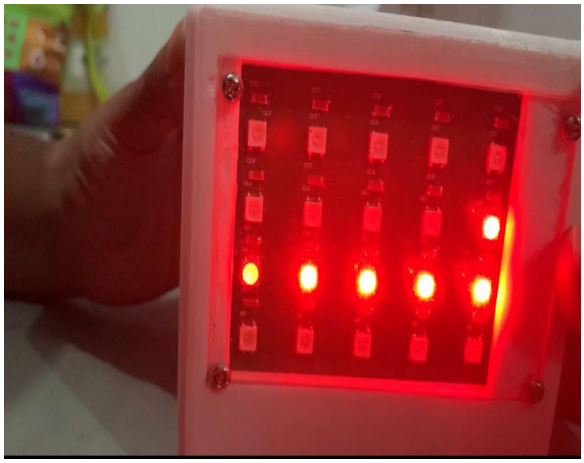
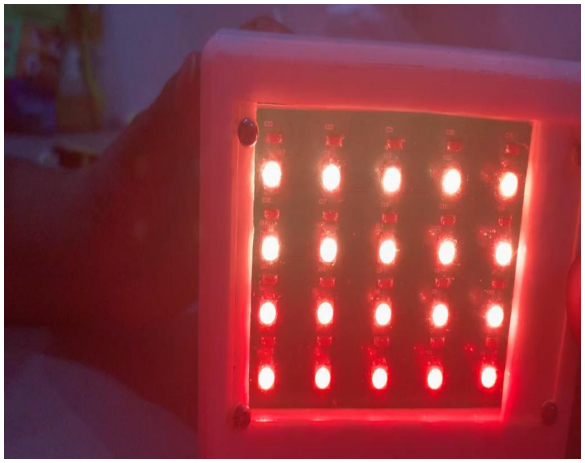
- App Control Mode
- Manual Control Mode

The device can change color, brightness, and patterns upon button press in Manual Control Mode.

- Selected Colors : Red,Blue,Green,Yellow,White
- Selected brightness Levels : 10 levels from 10%-100%
- Selected Patterns : Solid, Fire2012, Change Pallete, Fade all, Color Snake

Optimized the code to have minimum delays with the button press.
Used Blutooth serial Command to control the colors, Brightness and patterns in the app Mode.



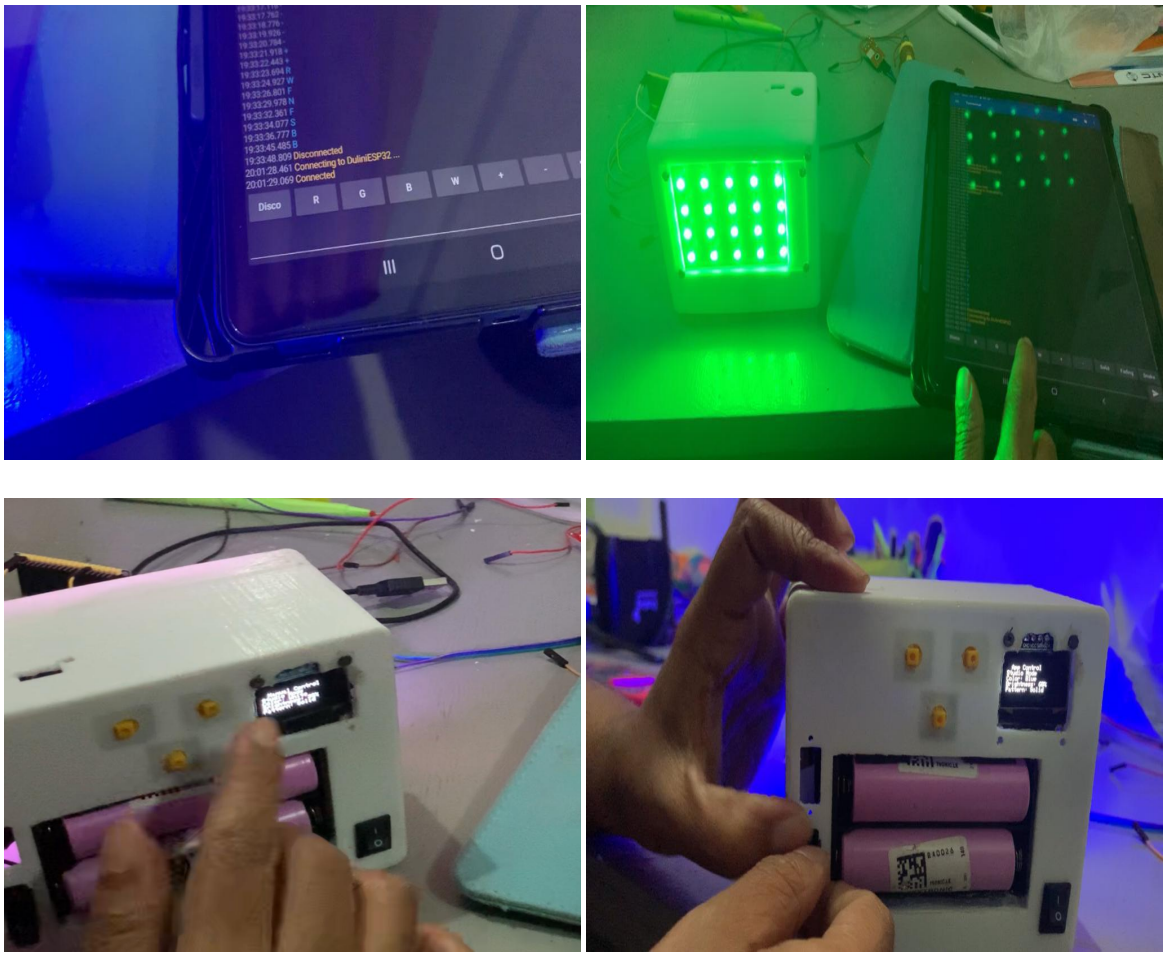


Figure 29: Final Device

5 Bill of Materials

The generated BOM using altium software is attached to the appendices.

6 Suppliers

Most of the components were internationally sourced from mouser electronics. The designed PCB was manufactured and imported from China JLCPCB. The design enclosure was 3D printed using normal white plastic filament.

7 Assembly Instructions

Since the device compromises of 2 PCBs when assembling the device should be concerned about the connecting procedures.

- Power off the switch
- Place the batteries in the battery holder
- Confirm if the appropriate voltages are at headers J20 and J31 using a multi meter, before moving forward.
- Connect power to ESP and connect power to the LED panel by connecting appropriate pins
- Connect the signal pins to LED Panel (header P10 in main PCB and P1 header in LED panel PCB)
- Press the power on switch and you can witness the functioning of the device.

8 Future Improvements

As further improvements, I plan to integrate the music visualization part into the device, also include a working and standby mode and to make the enclosure more compact, portable and plug-gable onto surfaces. Also would develop a user friendly Mobile app to replace te existing Bluetooth Serial Command

9 Acknowledgement

Throughout this project, I received the guidance from so many personalities. I would like to extend my sincere gratitude to our dear Lecturers for giving us valuable insights into the arena of product designing. Moreover I'm really grateful to my mentors from senior batches, Mr.Pahan and Ms. Chathuni, whose unwavering commitment and keen enthusiasm in our undertaking contributed to the success of this project. Additionally, I would like to thank all those who contributed in even the slightest way to achieve this goal.

10 Bibliography

1. Generating Sine Waves from Triangle Waves
2. Power path Controller
3. ESP32 with bluetooth
4. SK6812 Datasheet
5. ESP32 Wroom 32D datasheet
6. Highside switch datasheet
7. RGB LED strips
8. ESP32 Pinout Reference: Which GPIO pins should you use?

11 Smart LED Studio Light-User Manual

Congratulations on your purchase of the Smart LED Studio Light! This versatile lighting system is perfect for photography, videography, and even entertainment purposes. It comes with various features, including multiple colors, patterns, and adjustable brightness levels. The device can be controlled wirelessly via Bluetooth, allowing you to customize the lighting according to your needs.

- Package Contents
  - Smart LED Studio Light
  - 2 Li-Po Batteries (pre-installed)
  - 9V AC Adapter
  - User Manual
- Getting Started
  - Charging the Batteries:
    Before using the Smart LED Studio Light, ensure that the batteries are fully charged. You can charge them using the Li Po battery charging module.If using 9V adapter, connect the adapter to the charging port on the device and plug it into a power outlet.
  - Powering On/Off:
    The device has a power switch. Slide it to the ON position to power on the light. To turn it off, slide the switch to the OFF position.
- Manual Control Mode
  - Changing Colors:
    Press the buttons on the device to change the colors of the LED light. Each button corresponds to a specific color (e.g., Red, Blue, Green, Yellow, White).
  - Adjusting Brightness:
    Use the brightness control buttons to adjust the brightness level of the LED light. There are 10 levels, ranging from 10% to 100% brightness.
  - Selecting Patterns:
    The device offers various lighting patterns. Use the pattern buttons to cycle through and select the desired pattern (e.g., Solid, Fire2012, Change Pallete, Fade all, Color Snake).
- App Control Mode (Bluetooth)
  - Download the App:
    Search for the "Blutooth Serial Command" app on your smartphone's app store (compatible with both Android).
  - Bluetooth Pairing:
    Make sure Bluetooth is enabled on your smartphone. Open the app and follow the on-screen instructions to pair it with the Smart LED Studio Light. Once connected, you can control the colors, brightness, and patterns using the app interface.
- Safety Precautions
  - Do not expose the Smart LED Studio Light to extreme temperatures, direct sunlight, or moisture. Operate the device within the recommended temperature range (0°C to 40°C) and avoid using it in high humidity environments.

- Keep the device out of reach of children and pets. The Smart LED Studio Light contains small parts that may pose a choking hazard if swallowed. In addition, the LED panel may generate heat during operation, so avoid touching it while in use.
- Avoid looking directly into the LED light source, especially at maximum brightness levels. Prolonged exposure to intense LED light may cause discomfort or eye strain. If you experience any discomfort, reduce the brightness or take breaks during extended use.
- When using the Smart LED Studio Light in a photography or videography setup, ensure that the device is stable and securely mounted to prevent accidental falls or damage.
- Do not use the Smart LED Studio Light near flammable materials or open flames. The LED panel generates heat during operation, and placing it near flammable objects may create a fire hazard.
- Ensure that the provided 9V AC Adapter is used with the device. Using unauthorized or incompatible power adapters may cause damage to the Smart LED Studio Light or pose a safety risk.
- If you notice any damage to the device, including cracks, exposed wires, or loose components, stop using it immediately. Disconnect the power source and contact customer support for assistance.
- The Smart LED Studio Light is intended for indoor use only. Avoid using it in wet or damp conditions to prevent electrical hazards.
- Before cleaning the device, disconnect the power source and wait for it to cool down. Use a soft, dry cloth to clean the exterior of the device. Avoid using liquid cleaners or submerging the device in water.
- In case of a power outage or other electrical issues, disconnect the Smart LED Studio Light from the power source immediately to avoid damage from power surges.

- Warranty and Customer Support

  The Smart LED Studio Light comes with a limited warranty period. Please refer to the warranty documentation provided with the product for details.

  If you encounter any issues or have questions about the device, contact customer support for prompt assistance. Customer support contact information can be found in the product packaging.

- Disclaimer

  a. The Smart LED Studio Light is an electronic device, and as with all electronic devices, it may be susceptible to malfunctions or errors. While every effort has been made to ensure the accuracy and reliability of this product, the manufacturer and seller are not liable for any direct, indirect, incidental, or consequential damages that may arise from the use or misuse of this device.

  b. The user assumes all responsibility for the safe and proper use of the Smart LED Studio Light. It is essential to read and follow the safety precautions and instructions provided in this user manual to avoid accidents or damage to the device.

  c. The Smart LED Studio Light is not a medical device and should not be used for any medical or therapeutic purposes. If you have any medical conditions or concerns, consult a healthcare professional before using the device.

- Troubleshooting Guide

  - The Smart LED Studio Light does not power on:
    * Check the battery connection and ensure that the batteries are fully charged.

* Verify that the power switch is in the ON position.
* If using the 9V AC Adapter, ensure it is properly connected to the device and the power outlet.
* If the device still does not power on, try using a different set of fully charged batteries or a different power adapter.

– The LED light does not change colors or patterns:
* Ensure that the device is in the correct mode (Manual Control Mode or App Control Mode).
* If using the app, verify that the app is properly connected to the Smart LED Studio Light via Bluetooth.
* Check the connection between the LED panel and the main PCB. Make sure all connections are secure.
* If using the Manual Control Mode, check the buttons for any debris or dirt that may hinder their functionality.

– The Smart LED Studio Light is not responding to app commands:
* Make sure that Bluetooth is enabled on your smartphone and that it is within the operating range of the device (typically up to 30 feet).
* Close the app and reopen it to re-establish the connection.

– The LED light flickers or dims randomly:
* Ensure that the power source (batteries or AC Adapter) provides a stable voltage output.
* Check for any loose connections or faulty wiring in the circuitry.
* If using the app, adjust the brightness and color settings to see if the flickering stops.

– The Smart LED Studio Light gets hot during extended use:
* The LED panel generates heat during operation, which is normal. However, if the device becomes excessively hot, turn it off and allow it to cool down before further use.
* Reduce the brightness level or take breaks during extended use to prevent overheating.

– The Smart LED Studio Light does not respond to button presses:
* Check for any debris or dirt around the buttons that may hinder their functionality.
* Clean the buttons if necessary.
* Verify that the button connections are secure and properly soldered.

– The Smart LED Studio Light is not stable or securely mounted:
* If using the device on a stand or mount, ensure that it is properly attached and tightened.
* Avoid placing the device on uneven or unstable surfaces to prevent accidental falls.
* If the problem persists, contact customer support for further assistance.

Enjoy the full potential of your Smart LED Studio Light and explore its versatile features to enhance your photography and videography projects!

12 Appendices

12.1 Schematic Designs

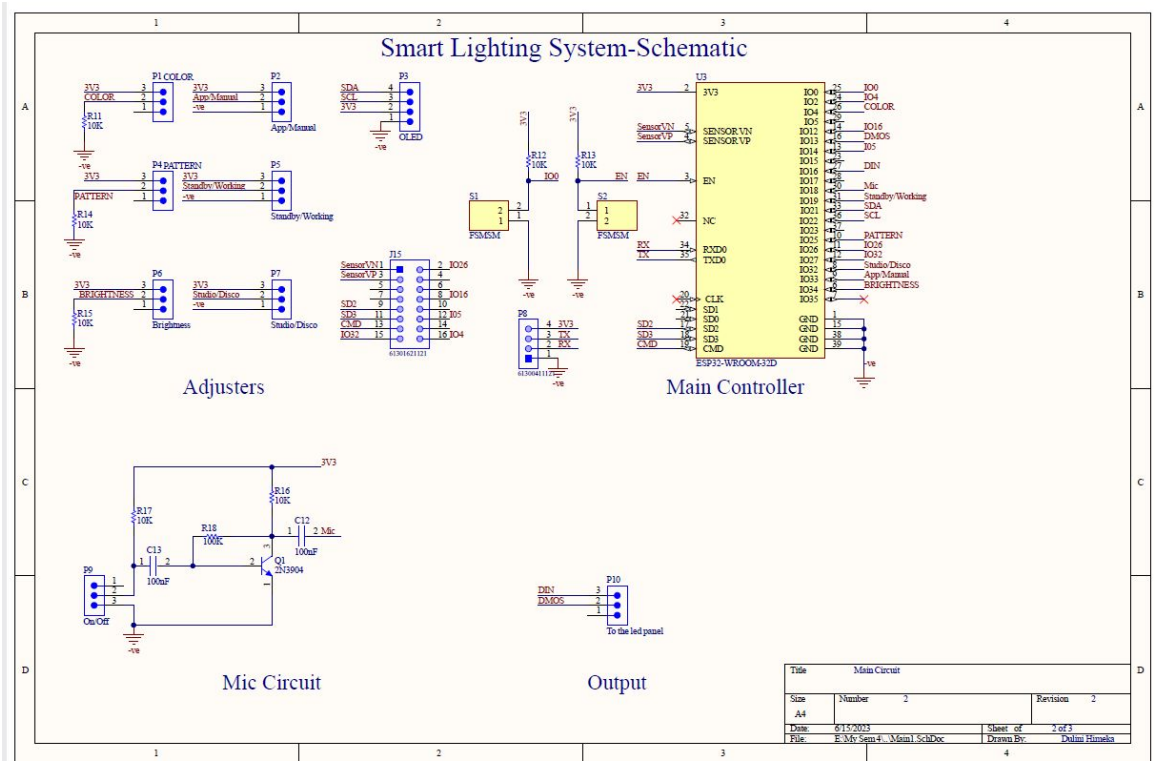


Figure 30: Main schematic

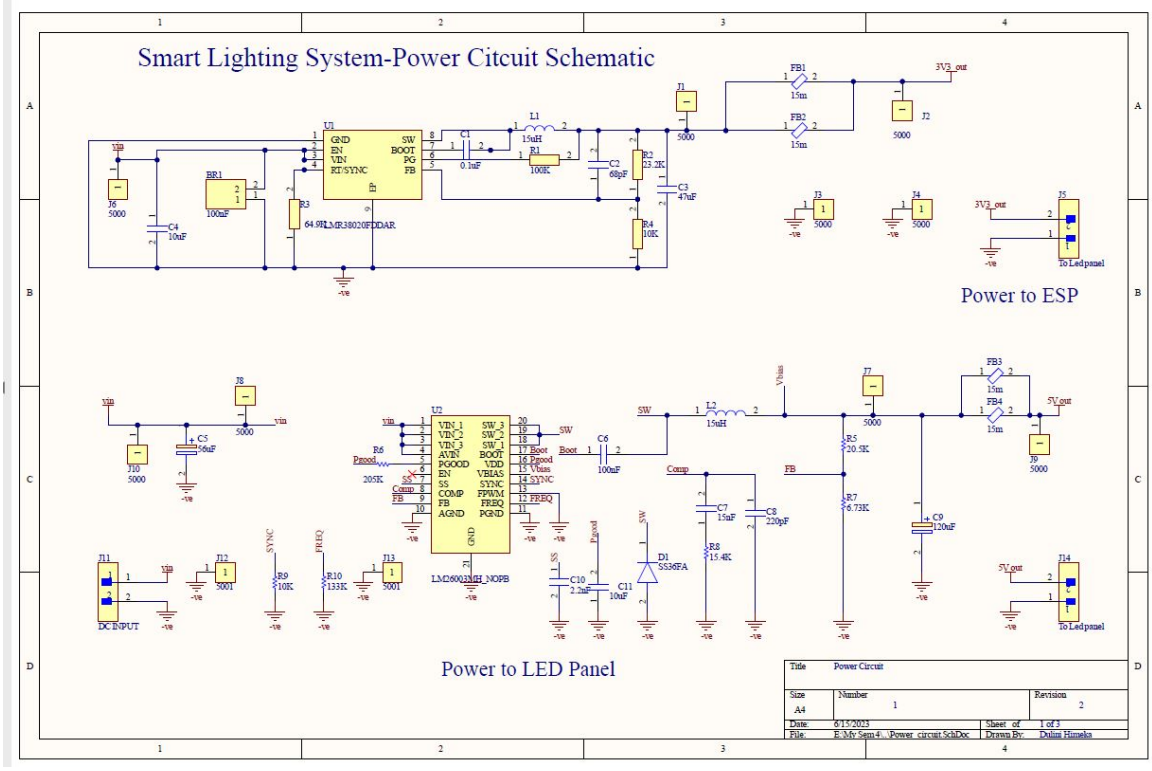


Figure 31: Power circuits

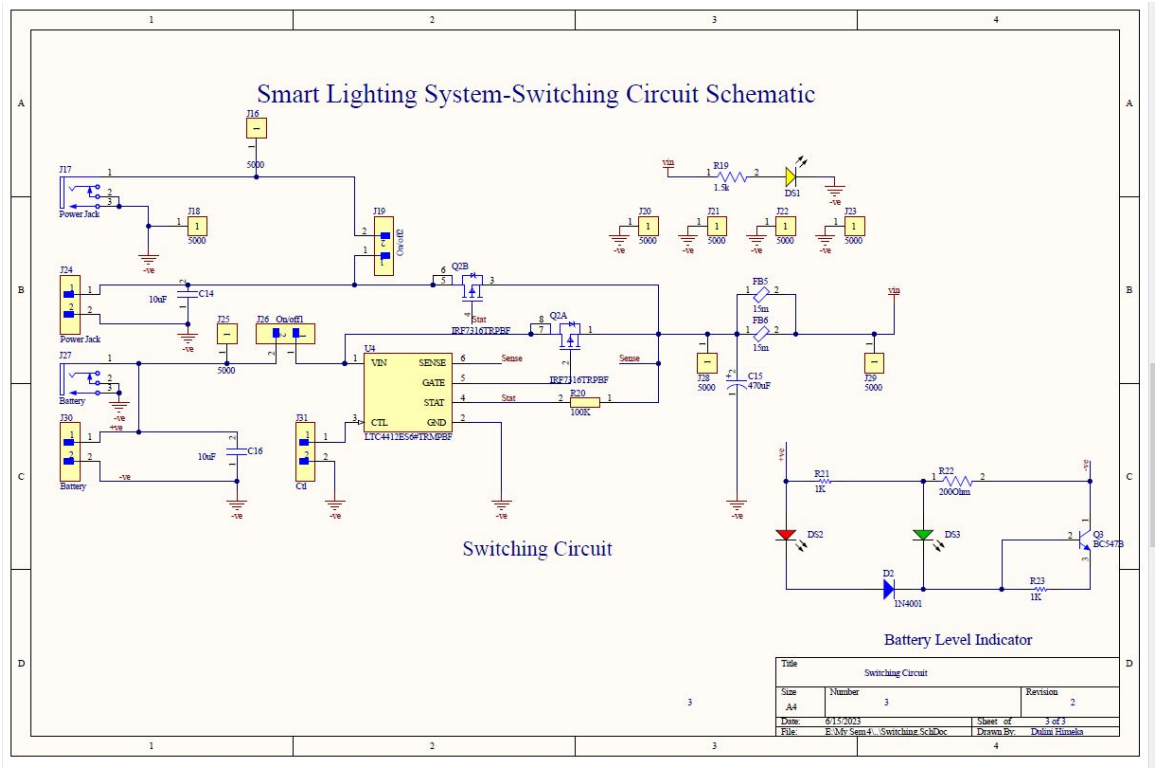


Figure 32: Switching circuit

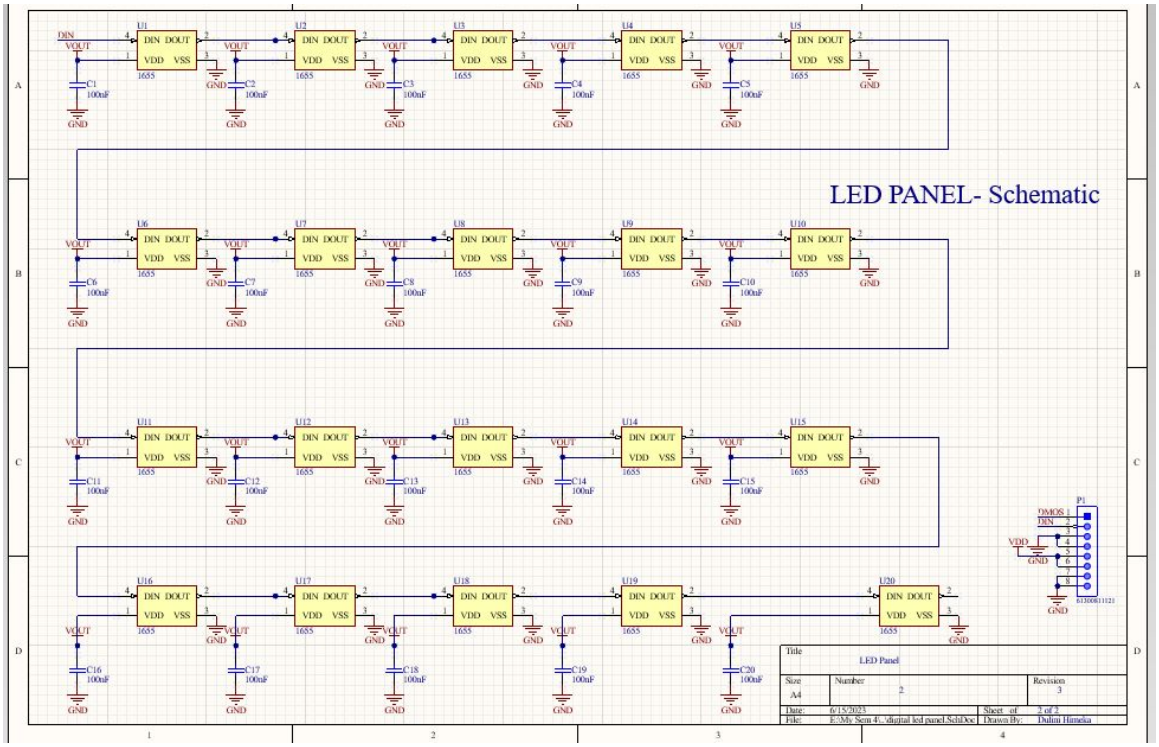


Figure 33: LED Panel

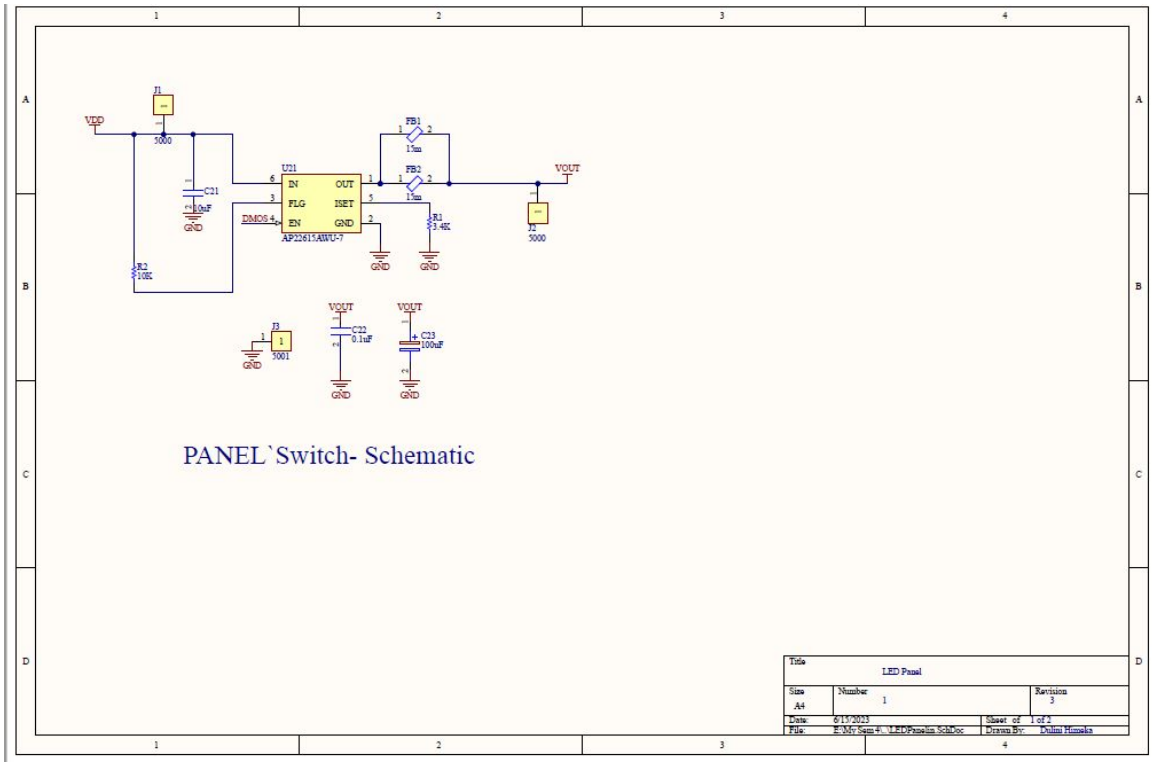


Figure 34: Mosfet highside switch

12.2 PCB Designs

Main Circuit

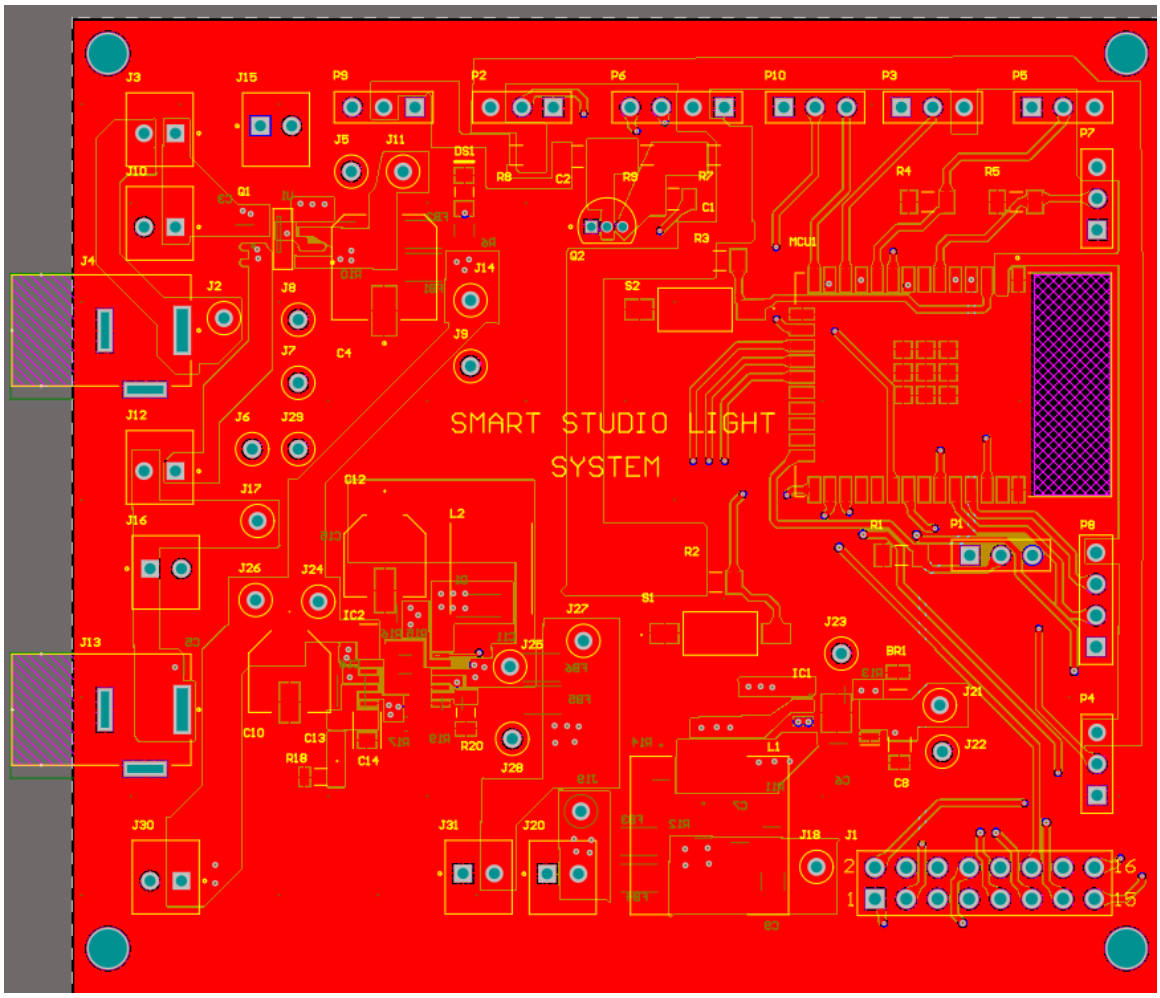


Figure 35: Top Layer

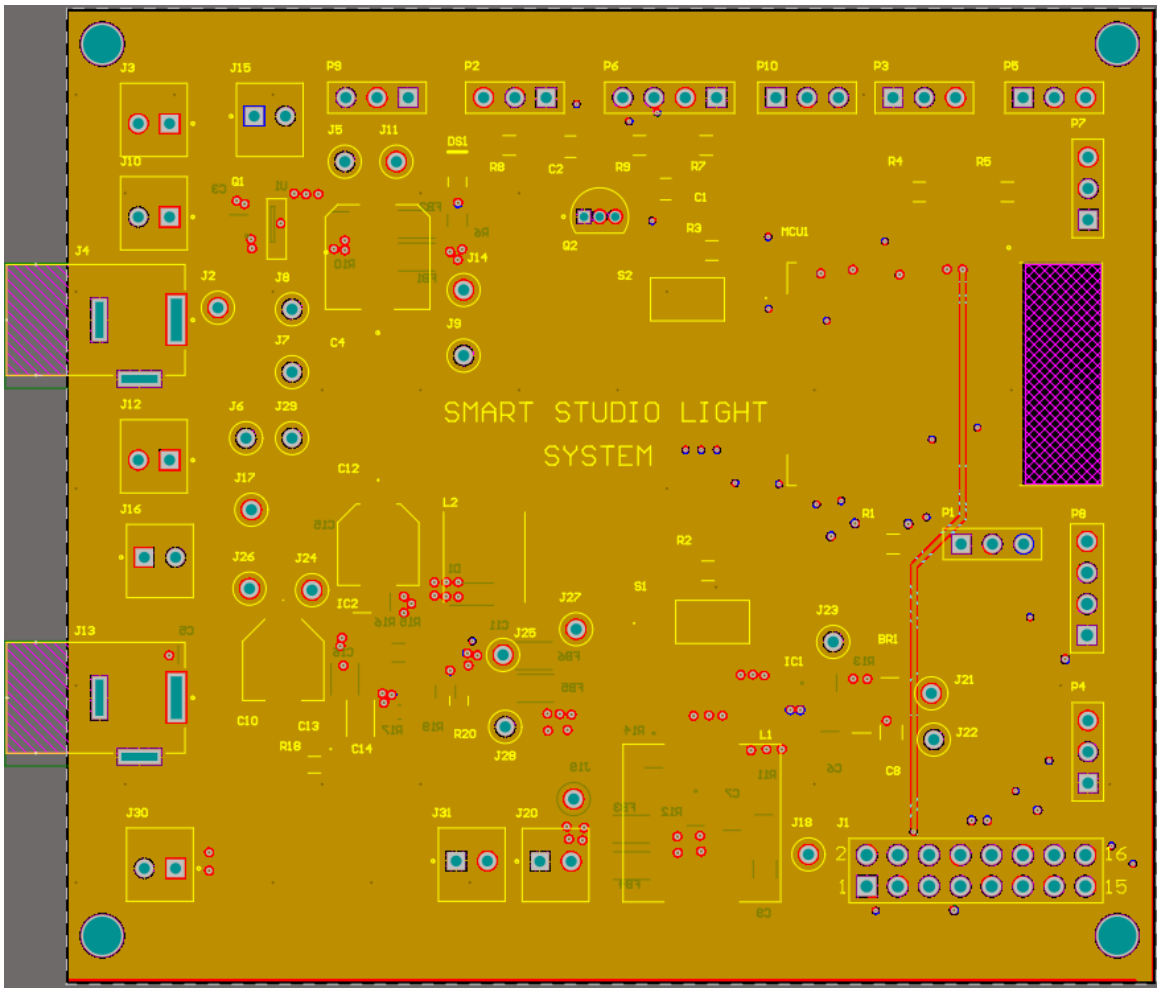


Figure 36: Layer 2

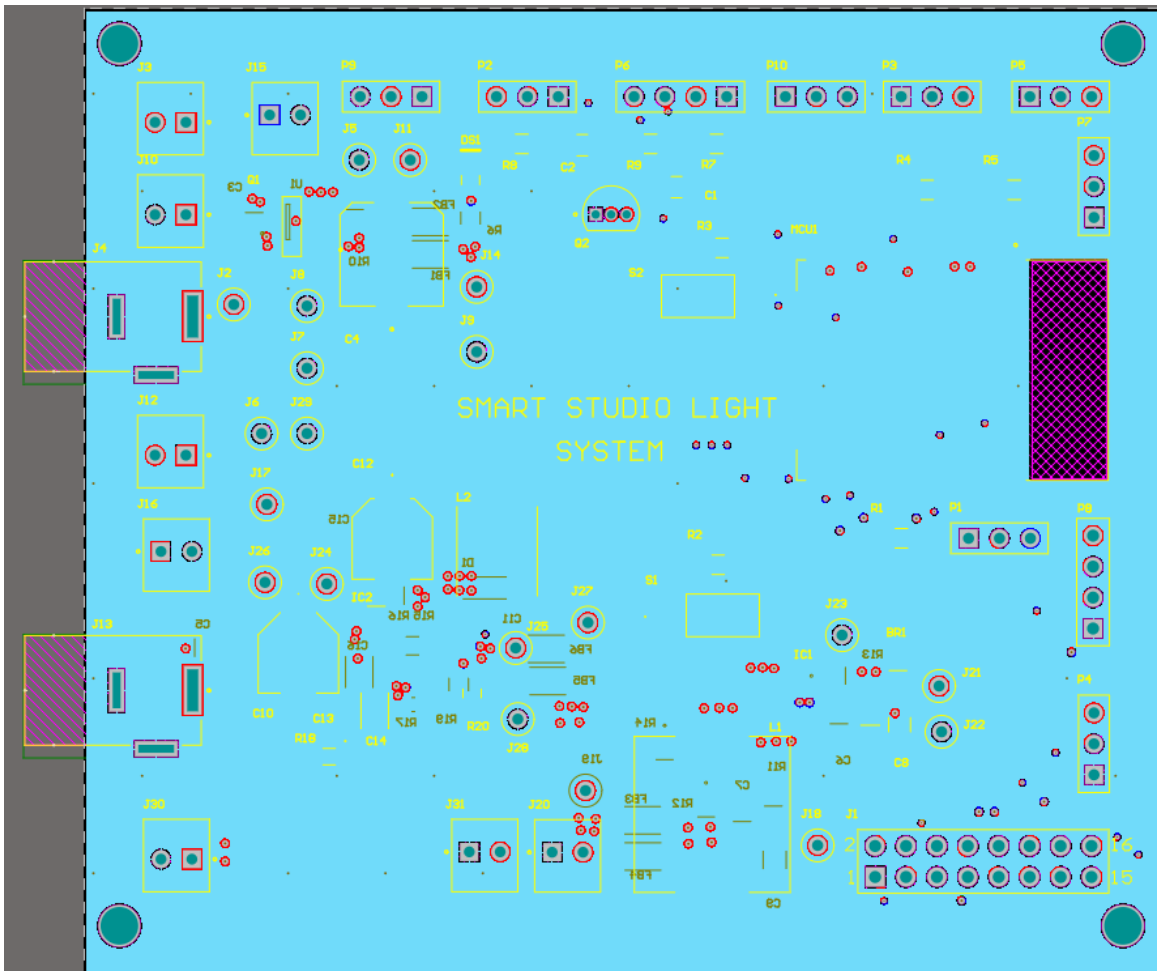


Figure 37: Layer 3

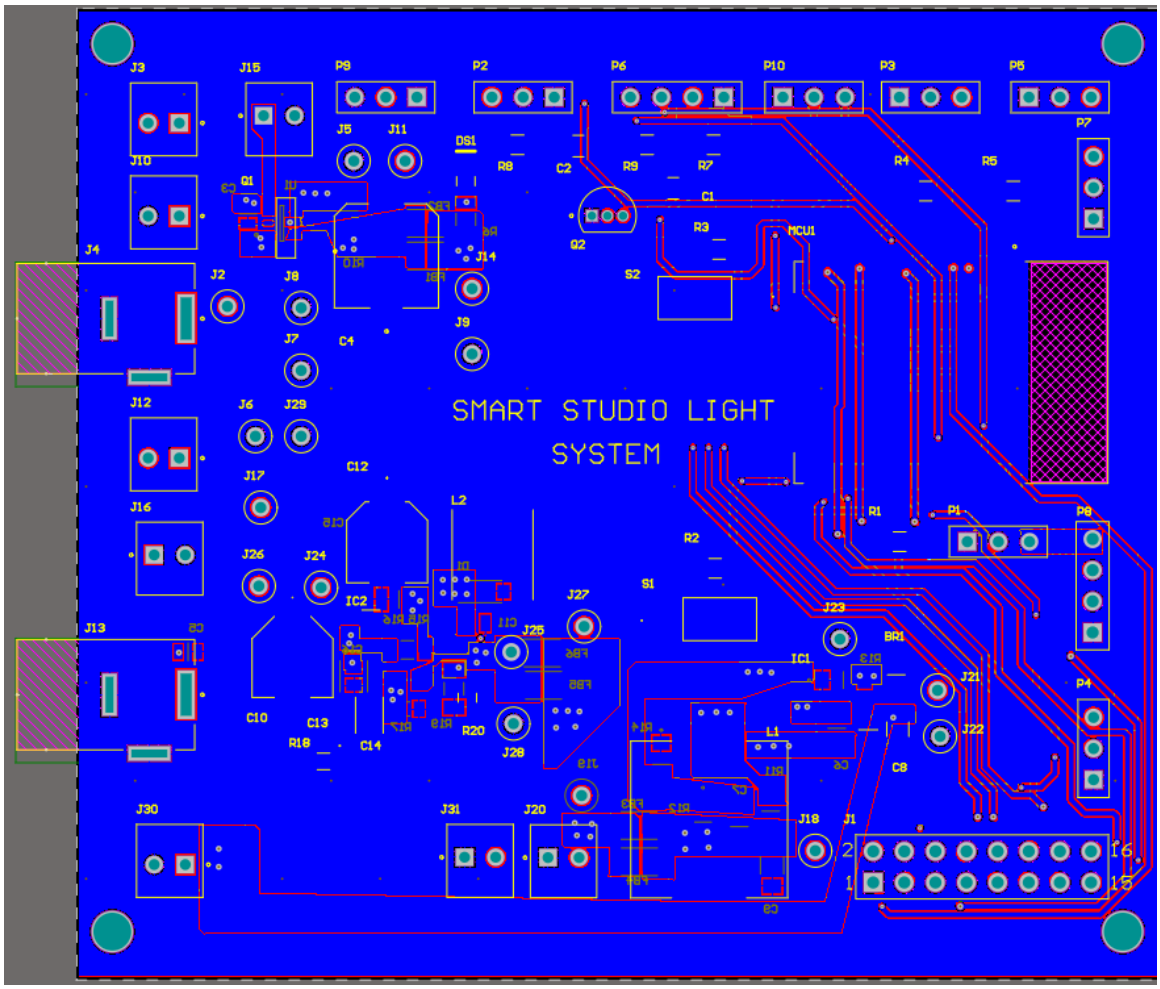


Figure 38: Bottom Layer

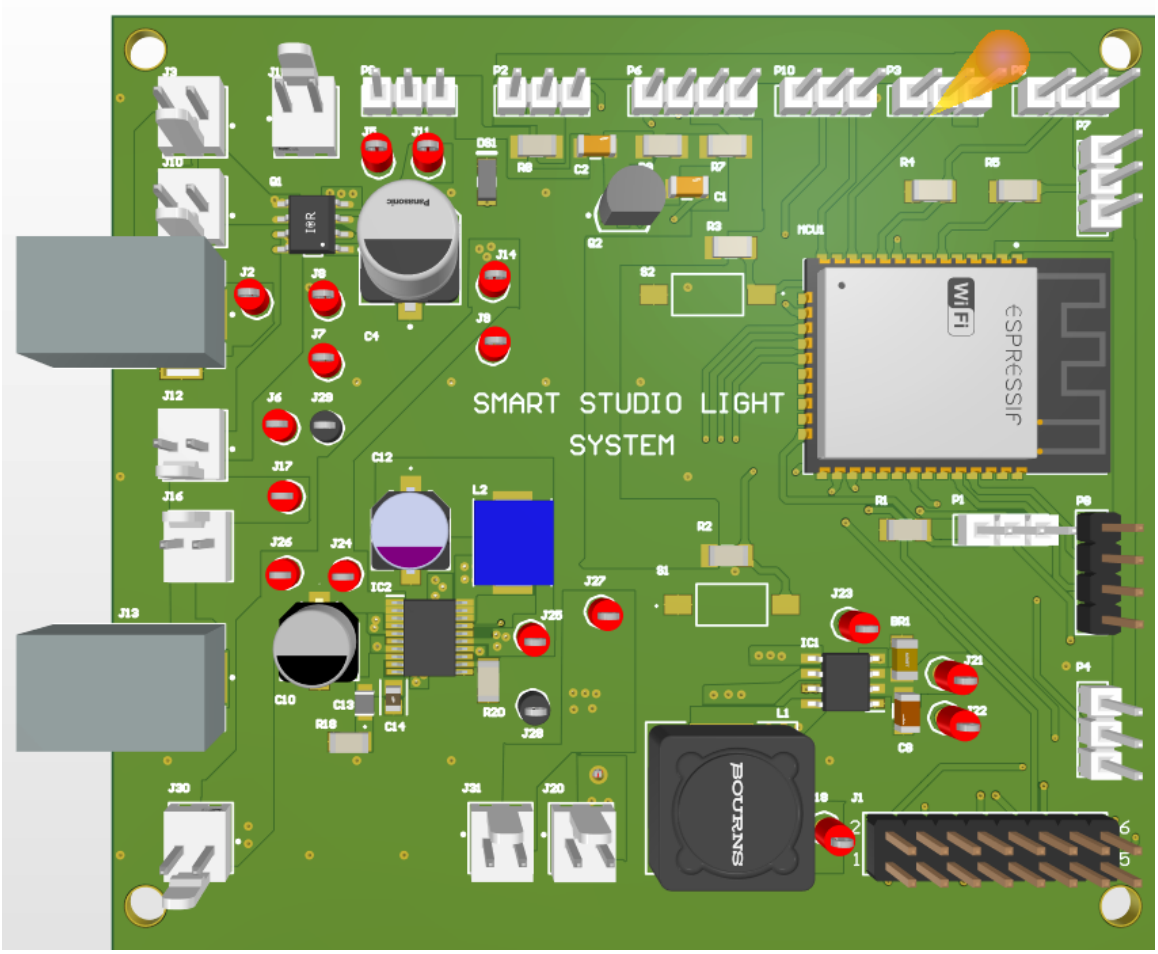


Figure 39: View top

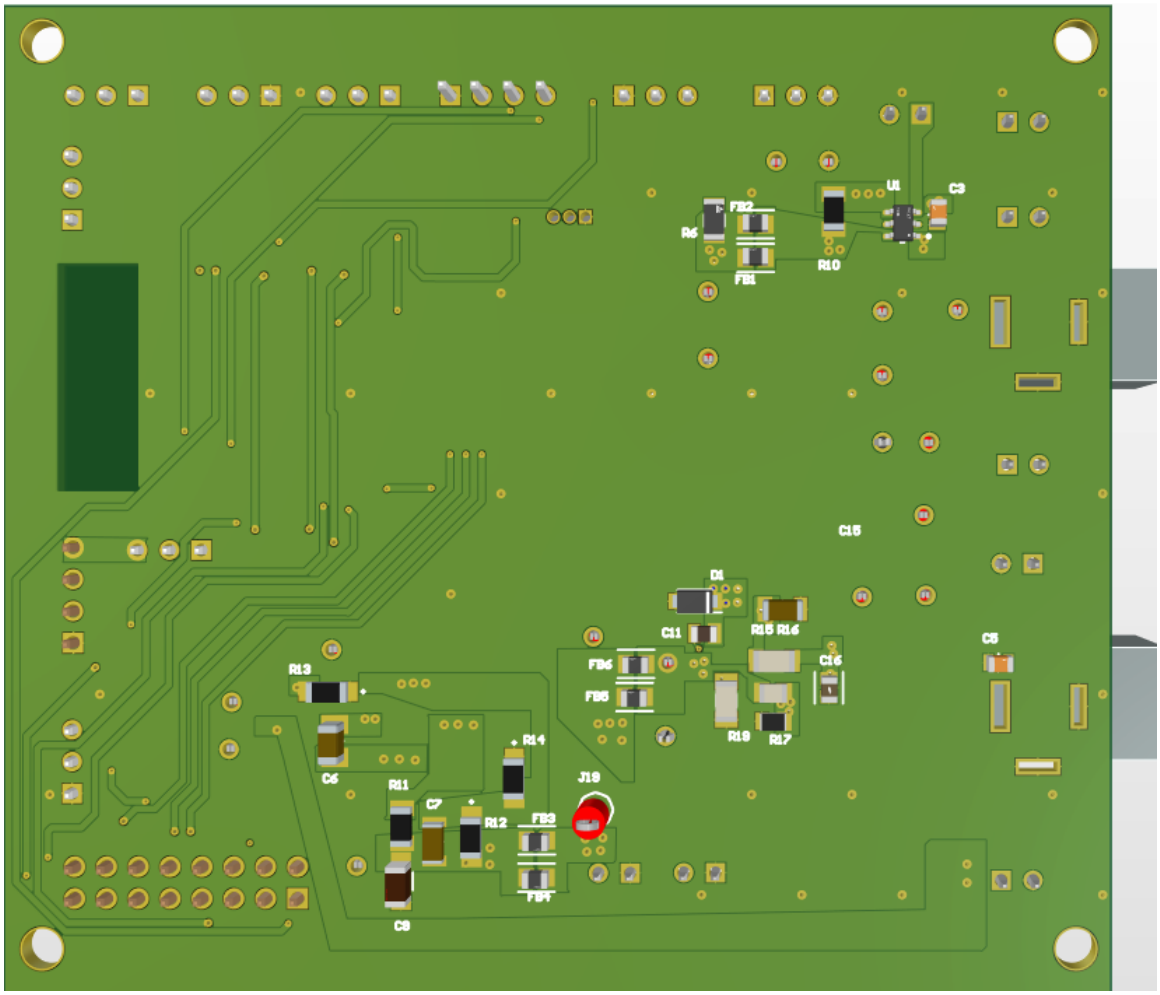


Figure 40: View Bottom

LED Panel

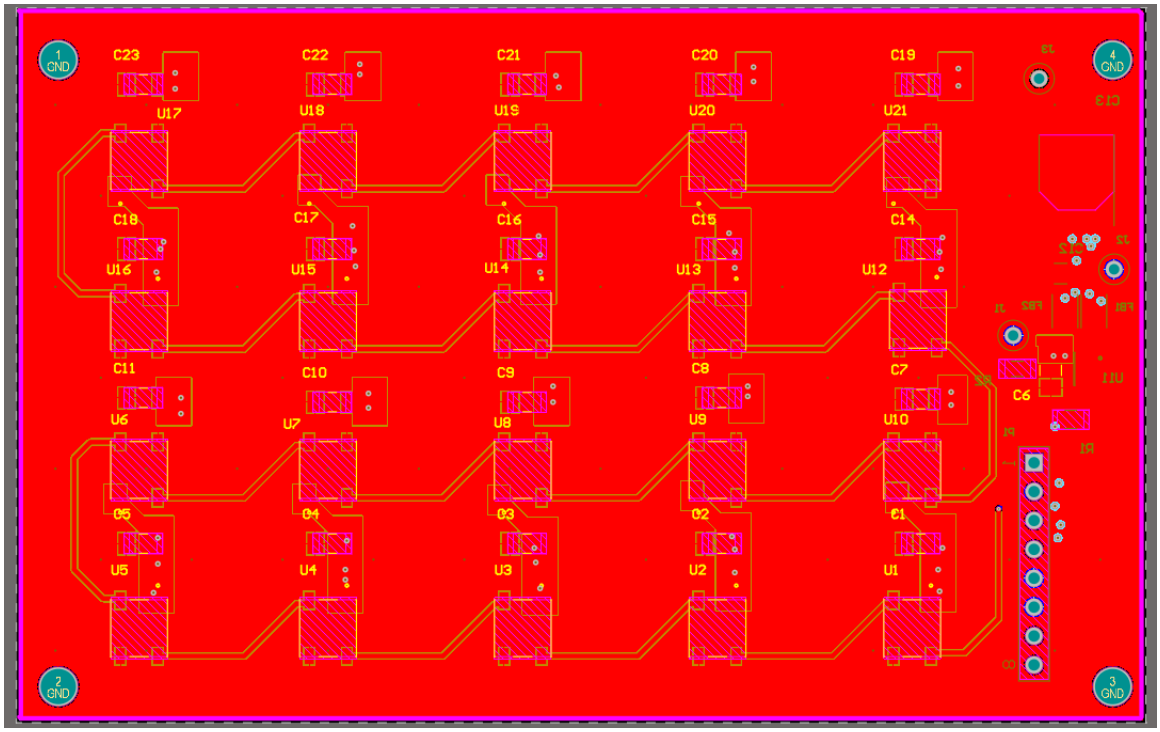


Figure 41: Top Layer

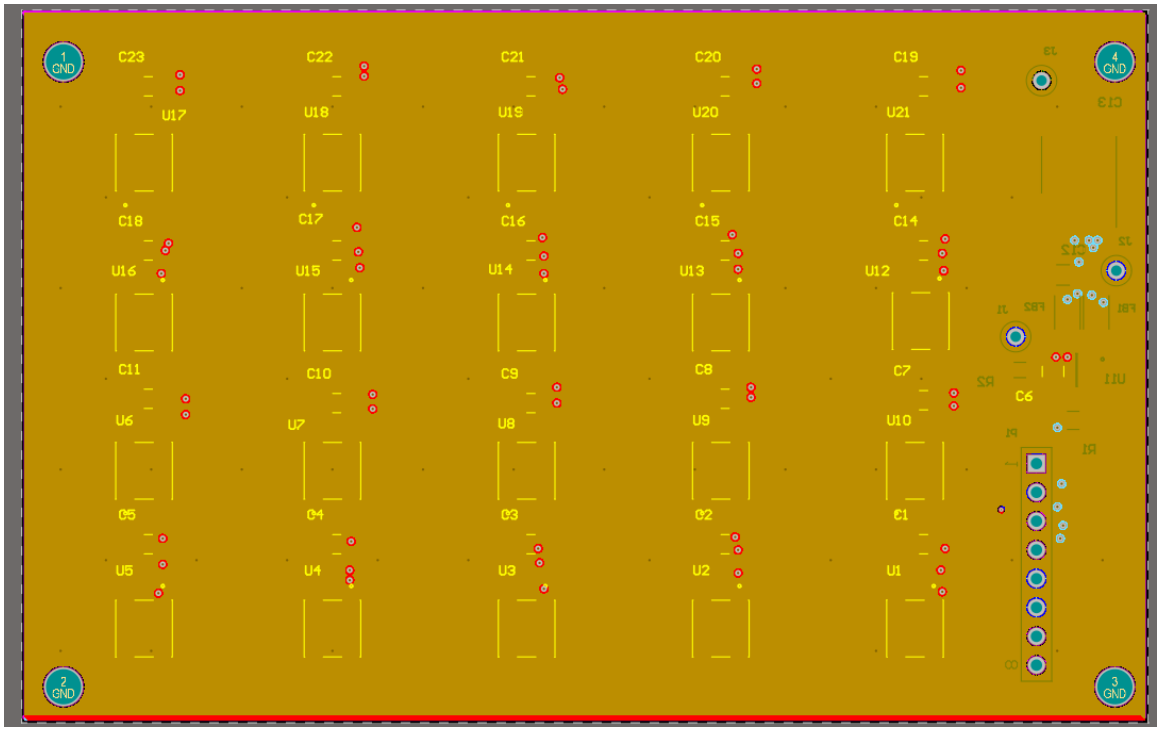


Figure 42: Layer 2

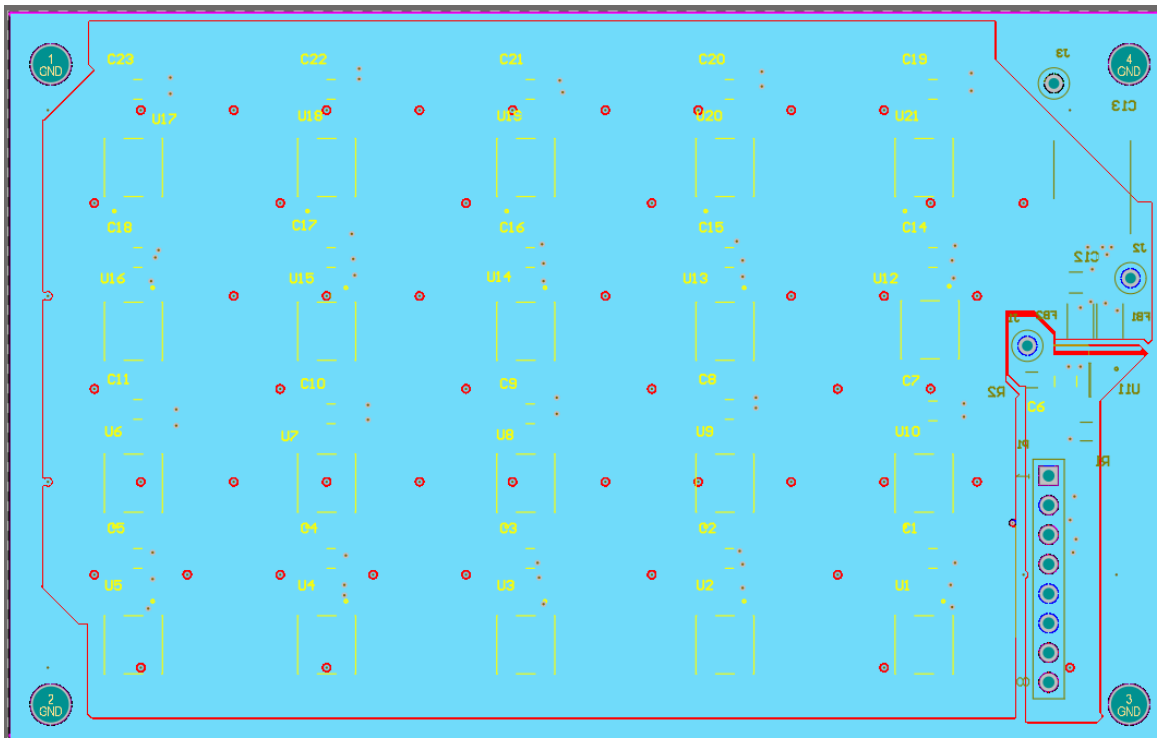


Figure 43: Layer 3

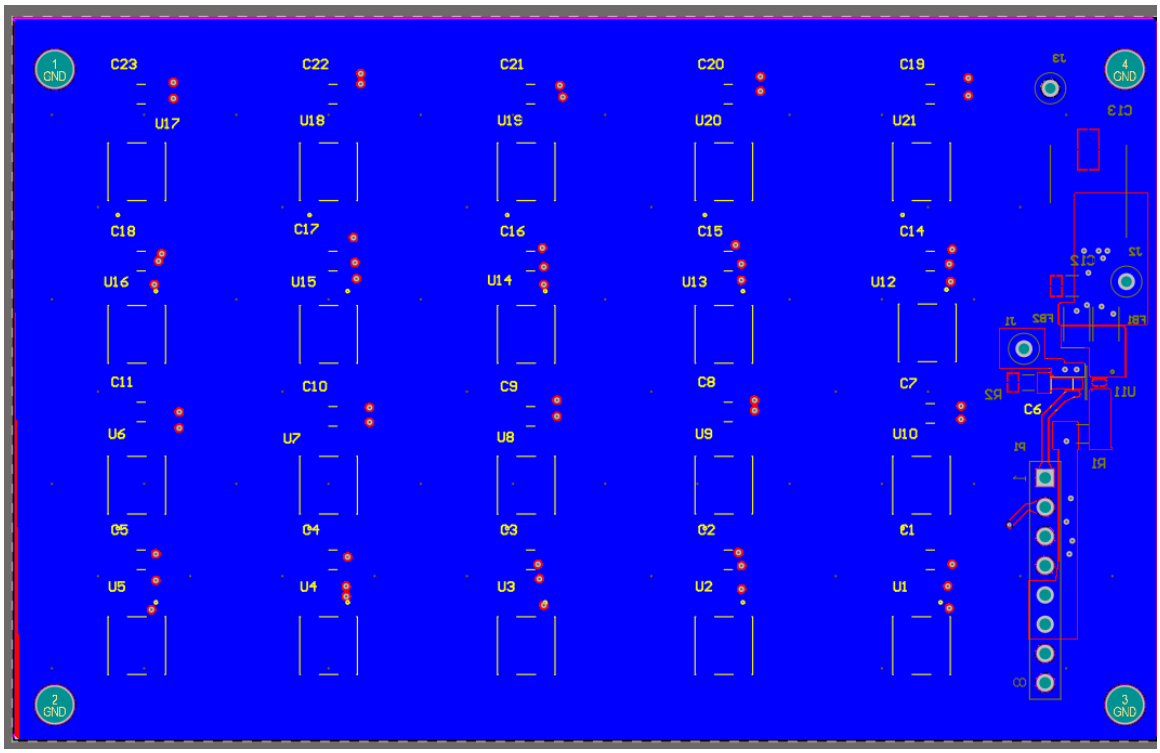


Figure 44: Bottom Layer

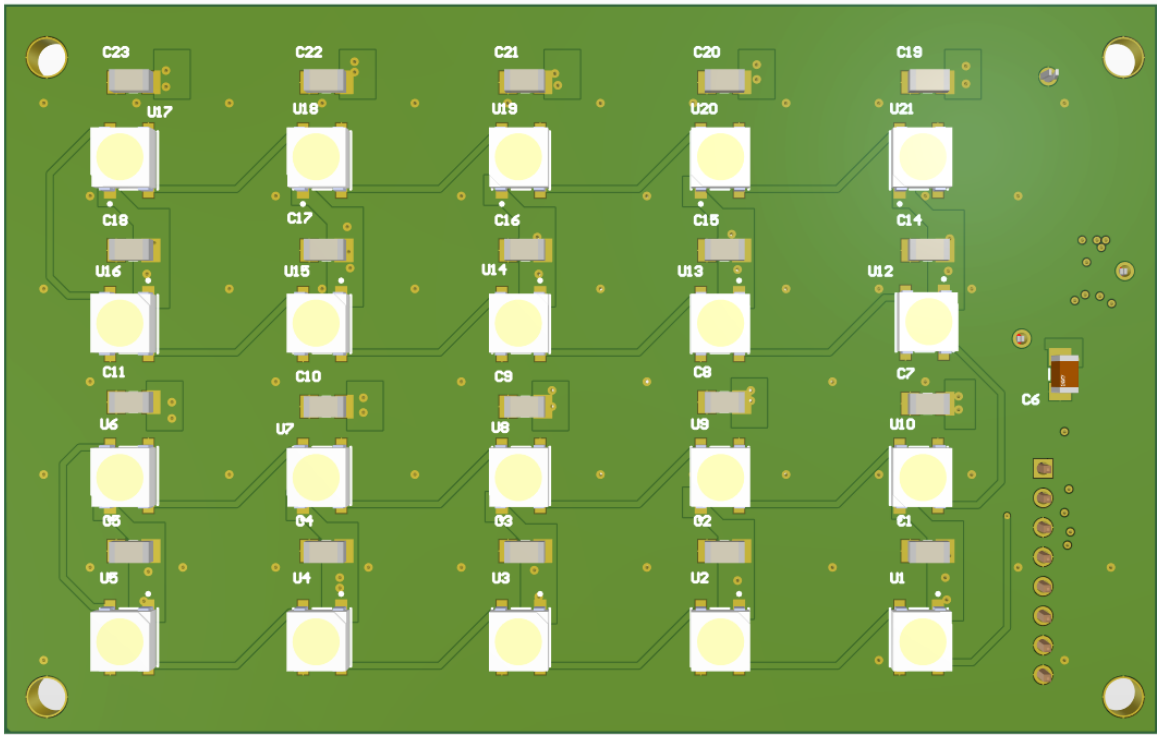


Figure 45: View Top

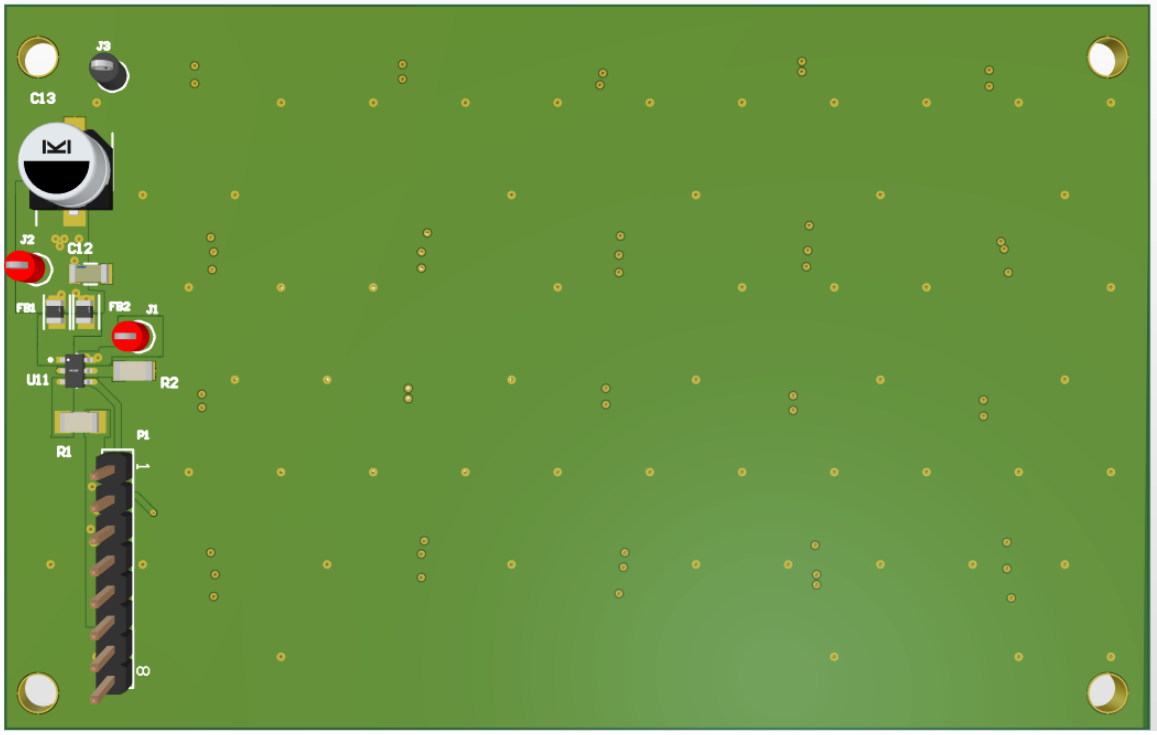


Figure 46: View Bottom

12.3 Gerber and Drill Files

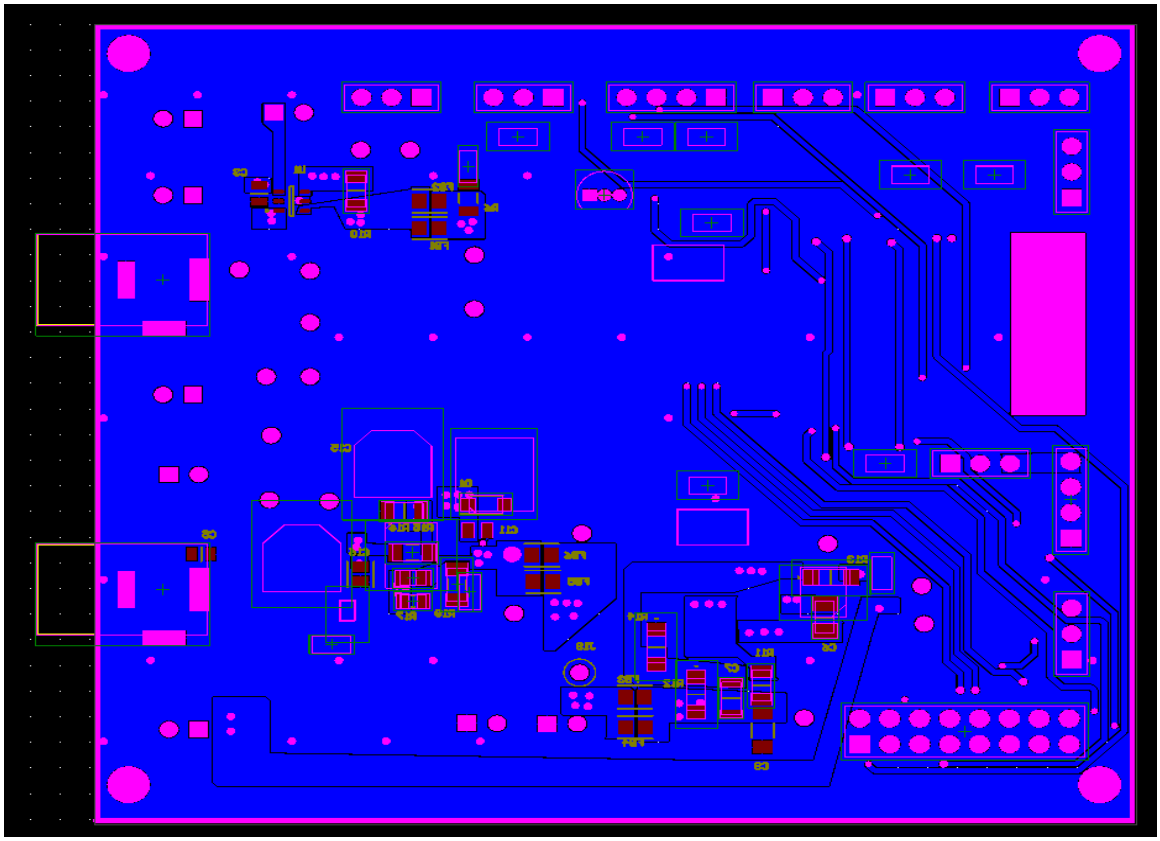


Figure 47: Gerber file- Main PCB

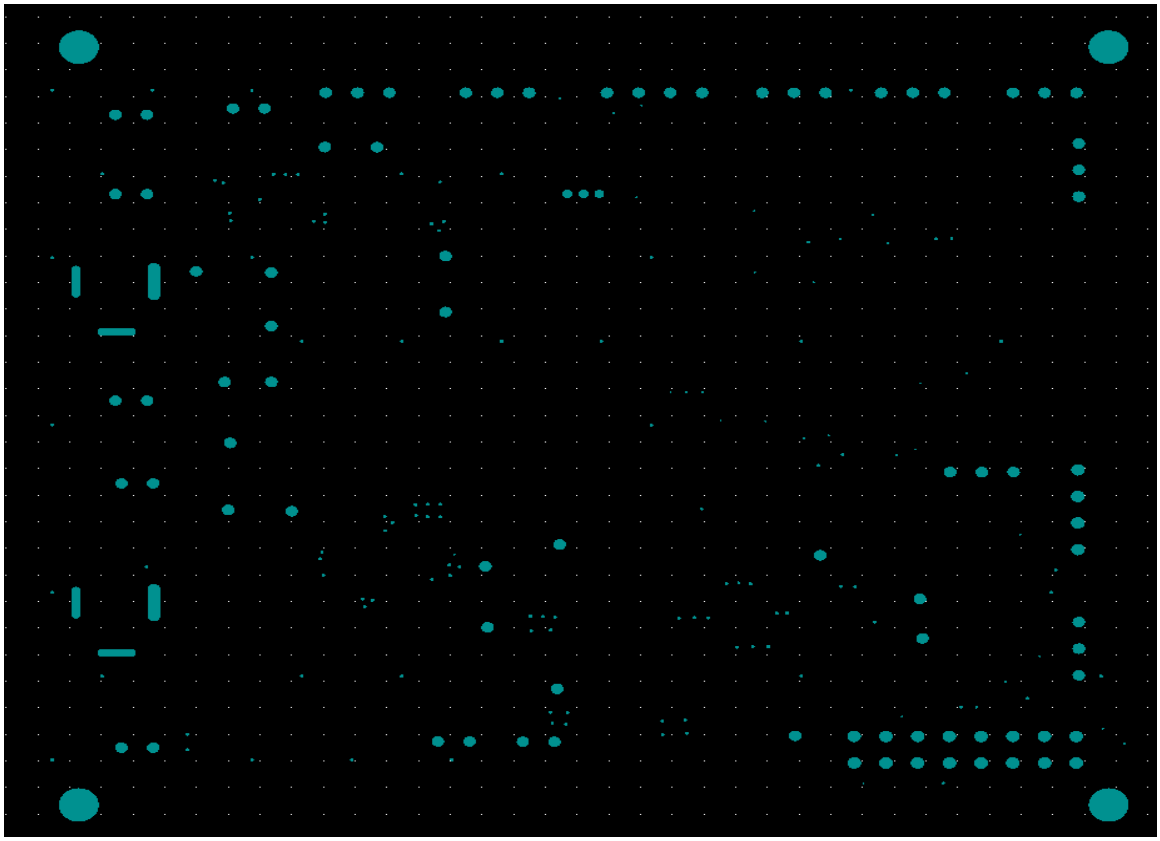


Figure 48: Drill file - Main Circuit

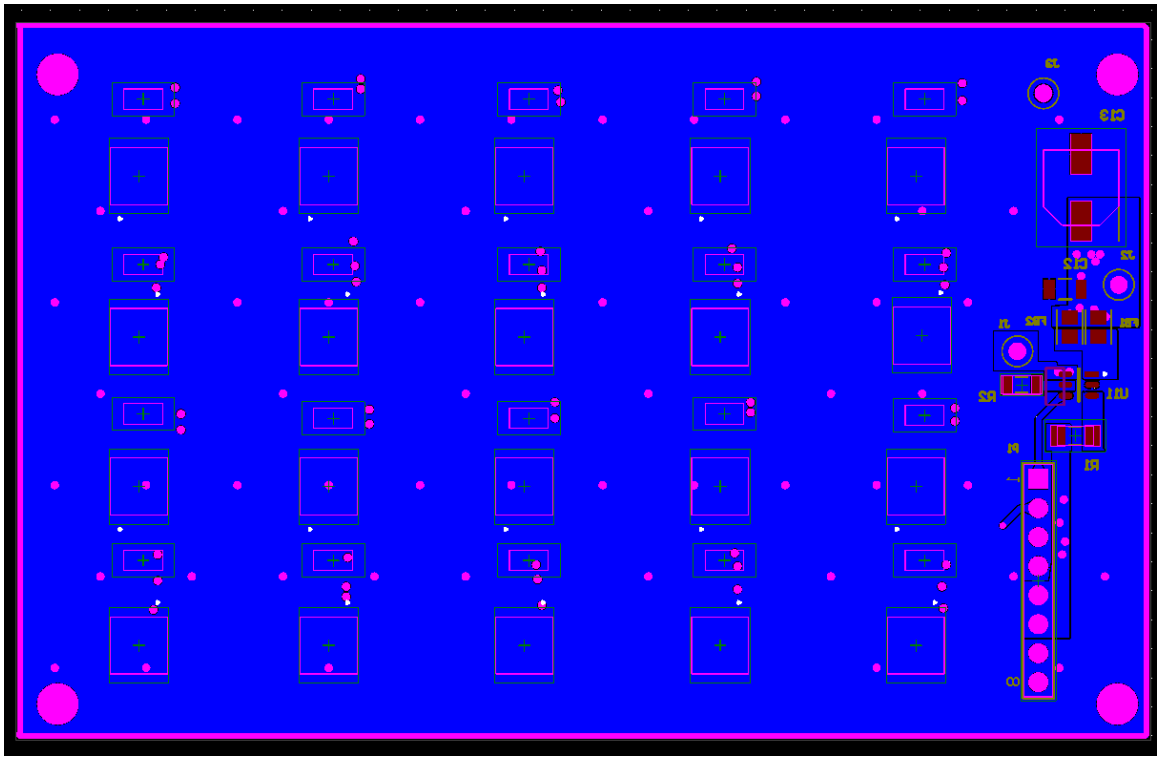


Figure 49: Gerber file- LED Panel

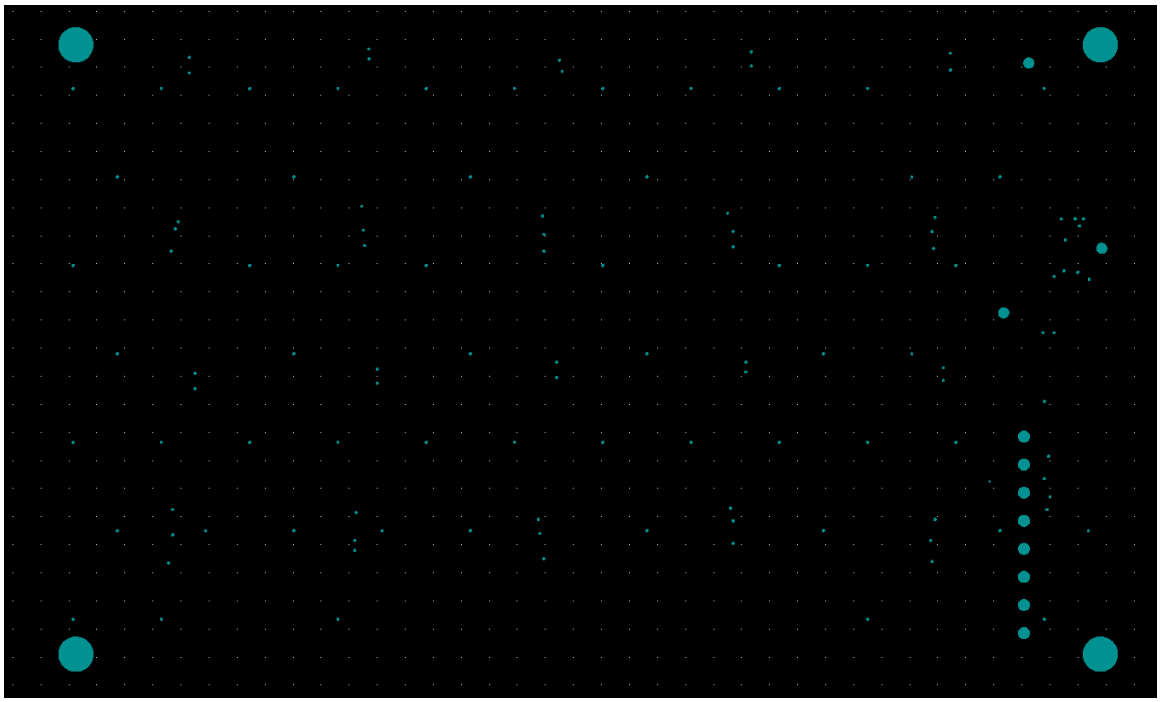


Figure 50: Drill File - LED Panel

12.4 Enclosure Designs

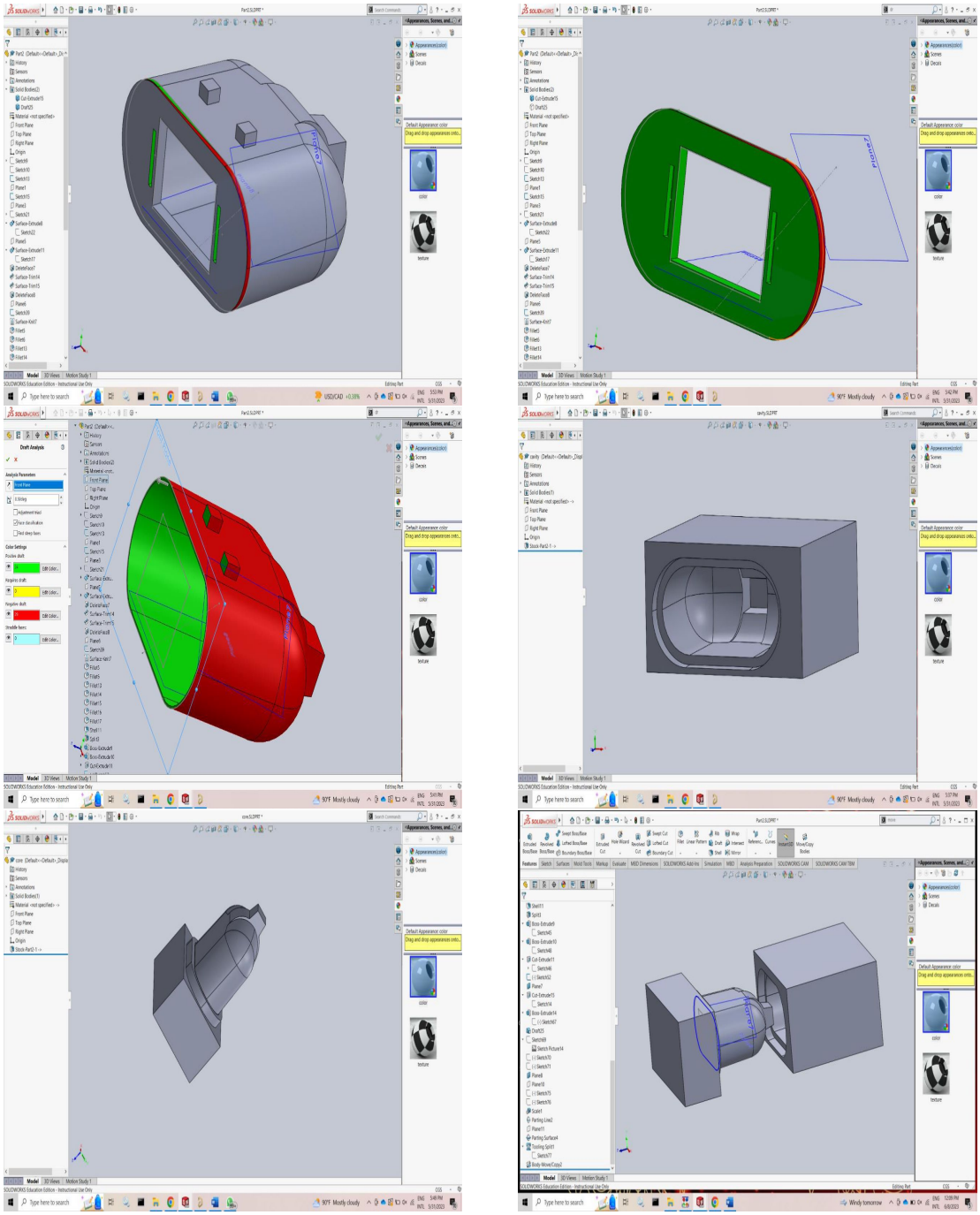


Figure 51: Enclosure designs Prototype

12.5 Bill Of Materials

Line #	Name	Description	Designator	Revision ID	Revision State	Revision Status	Quantity	Manufacturer 1	Manufacturer Part Number 1	Manufacturer Lifecycle 1	Supplier 1	Supplier Part Number 1	Supplier Unit Price 1	Supplier Subtotal 1
	C1206C104KSRACTU	CAP, Ceramic, 0.1 uF, +/- 10%, 50 V, -55 to 125 degC, 1206 (3216 Metric), RoHS, Tape and Reel	C1, C2, C3, C4, C5, C7, C8, C9, C10, C11, C14, C15, C16, C17, C18, C19, C20, C21, C22, C23	CMP-1037-04202-2	Released	Up to date	20	KEMET	C1206C104KSRACTM	Volume Production	Newark	96Y7250	0.02	0.4
	1206ZD106KAT2A	General Purpose Ceramic Capacitor, 1206, 10uF, 10%, X5R, 15%, 10V	C6	CMP-04427-021870-1	New From Design	Up to date	1	Kyocera AVX	1206ZD106KAT2A	Volume Production	Arrow Electronics	1206ZD106KAT2A	0.5108	0.5108
	C1206C104K1RACTU	CAP CER 0.1UF 100V X7R 1206	C12	CMP-03020-000153-1	New From Design	Up to date	1	KEMET	C1206C104K1RACTU	Volume Production	Newark	70K9165	0.051	0.051
	100uF	Capacitor Polarised	C13		Not managed		1							
	BLM21PG300SH1D	Chip Ferrite Bead, 0805, 30 Ω @ 100MHz, 0.014Ω , 4A	FB1, FB2	CMP-06046-001820-1	New From Design	Up to date	2	Murata	BLM21PG300SH1D	Volume Production	Arrow Electronics	BLM21PG300SH1D	0.1406	0.2812
	5000	PC TEST POINT MINIATURE RED	J1, J2	CMP-19636-000027-1	New From Design	Up to date	2	Keystone Electronics	5000	Volume Production	Farnell	2250281	0.25235	0.50469
	5001	PC TEST POINT MINIATURE BLACK	J3	CMP-2000-05181-2	Released	Up to date	1	Keystone Electronics	5001	Volume Production	Farnell	2301278	0.25235	0.25235
	61300811121	THT Vertical Pin Header WR-PHD, Pitch 2.54 mm, Single Row, 8 pins	P1	CMP-1502-01076-1	Released	Out of date	1	Wurth Electronics	61300811121	Volume Production	Digi-Key	732-5321-ND	0.4	0.4
	RC1206FR-0710KL	Chip Resistor, 10 KOhm, +/- 1%, 0.25 W, -55 to 155 degC, 1206 (3216 Metric), RoHS, Tape and Reel	R1	CMP-1014-00623-2	Released	Up to date	1	Bourns	CR1206-FX-1002ELF	Volume Production	Newark	84W7259	0.006	0.006
	ERJ-8ENF3401V		R2	CMP-2003-00314-1	Released	Up to date	1	Panasonic	ERJ-8ENF3401V	Not Recommended for New Design	Arrow Electronics	ERJ-8ENF3401V	0.0307	0.0307
	1655	Intelligent Control LED Integrated Light Source, 3.5 to 5.3 V, -25 to 80 degC, 4-Pin SMD, RoHS, Tape and Reel	U1, U2, U3, U4, U5, U6, U7, U8, U9, U10, U12, U13, U14, U15, U16, U17, U18, U19, U20, U21	CMP-2000-05026-1	Released	Up to date	20	Adafruit Industries	1655	Unknown	Mouser	485-1655	4.5	90
	AP22615AWU-7	IC PWR SWTCH N/P-CHAN 1:1 TSOT26	U11	CMP-12757-000023-1	New From Design	Up to date	1	Diodes	AP22615AWU-7	Volume Production	Arrow Electronics	AP22615AWU-7	0.8008	0.8008

Figure 52: BOM-LED Panel

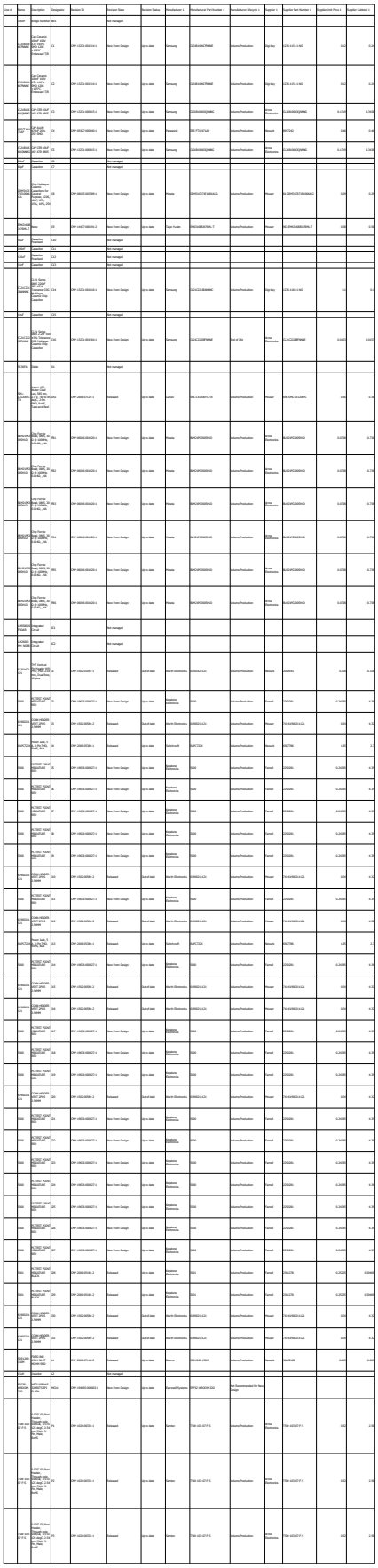


Figure 53: BOM-Main CCT

12.6 Suppliers

10		**Mouser #:** 647-RSA1A121MCN1GS **Mfr. #:** RSA1A121MCN1GS **Mfr.:** Nichicon **Customer #:** Customer # Add	Aluminum Organic Polymer Capacitors 10V 120uF 20% Tol. RoHS Compliant	2 **Packaging:** **Cut Tape	2 Ships Now	$0.88	**$1.76**
11		**Mouser #:** 581-08053C104K **Mfr. #:** 08053C104KAT2A **Mfr.:** KYOCERA AVX **Customer #:** Customer # Add	Multilayer Ceramic Capacitors MLCC - SMD/SMT 25V 0.1uF X7R 0805 10% RoHS Compliant By Exemption	10 **Packaging:** **Cut Tape	10 Ships Now	$0.034	**$0.34**
12		**Mouser #:** 647-UCM1E471MNL1GS **Mfr. #:** UCM1E471MNL1GS **Mfr.:** Nichicon **Customer #:** Customer # Add	Aluminum Electrolytic Capacitors - SMD 25V 470uF TOL 20% AEC-Q200 RoHS Compliant	2 **Packaging:** **Cut Tape	2 Ships Now	$0.75	**$1.50**
13					10 Ships Now	$0.074	**$0.74**

Sort	Product Detail		Description	Quantity	Availability	Unit Price (USD)	Ext. Price (USD)
		**Mouser #:** 187-CL21A106KOQNNNF **Mfr. #:** CL21A106KOQNNNF **Mfr.:** Samsung Electro-Mechanics **Customer #:** Customer # Add	Multilayer Ceramic Capacitors MLCC - SMD/SMT 10uF+/-10% 16V X5R 2 0805 RoHS Compliant	10 **Packaging:** **Cut Tape			
14		**Mouser #:** 603-RC0805FR-0710KL **Mfr. #:** RC0805FR-0710KL **Mfr.:** YAGEO **Customer #:** Customer # Add	Thick Film Resistors - SMD 10 kOhms 125 mW 0805 1% RoHS Compliant By Exemption	100 **Packaging:** **Cut Tape	100 Ships Now	$0.014	**$1.40**
15		**Mouser #:** 710-865080443008 **Mfr. #:** 865080443008 **Mfr.:** Wurth Elektronik **Customer #:** Customer # Add	Aluminum Electrolytic Capacitors - SMD WCAP-ASLI 56uF 25V 20% SMD/SMT RoHS Compliant	2	2 Ships Now	$0.27	**$0.54**
16		**Mouser #:** 603-AC0805FR-7W3K3L **Mfr. #:** AC0805FR-7W3K3L **Mfr.:** YAGEO **Customer #:** Customer # Add	Thick Film Resistors - SMD 3.3kOhms 1/4W 0805 1 % AEC-Q200 Double Power Version RoHS Compliant By Exemption	100 **Packaging:** **Cut Tape	100 Ships Now	$0.034	**$3.40**
17		**Mouser #:** 603-RC1206FR-07133KL **Mfr. #:** RC1206FR-07133KL **Mfr.:** YAGEO **Customer #:** Customer # Add	Thick Film Resistors - SMD 133 kOhms 250 mW 1206 1% RoHS Compliant By Exemption	4 **Packaging:** **Cut Tape	4 Ships Now	$0.10	**$0.40**
18		**Mouser #:** 603-RT0805BRD076K73L **Mfr. #:** RT0805BRD076K73L **Mfr.:** YAGEO **Customer #:** Customer # Add	Thin Film Resistors - SMD 1/8W 6.73K Ohms 0.1% RoHS Compliant By Exemption	2 **Packaging:** **Cut Tape	2 Ships Now	$0.36	**$0.72**
19		**Mouser #:** 926-LM26003MHX/NOPB **Mfr. #:** LM26003MHX/NOPB **Mfr.:** Texas Instruments **Customer #:** Customer # Add	Switching Voltage Regulators 3A Switching Regulator with High Efficiency Sleep Mode 20-HTSSOP -40 to 125 RoHS Compliant	2 **Packaging:** **Cut Tape	2 Ships Now	$3.94	**$7.88**

Sort	Product Detail		Description	Quantity	Availability	Unit Price (USD)	Ext. Price (USD)
20	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	506-FSMSM FSMSM TE Connectivity Customer # Add	Tactile Switches 3.5X6 SMT TACT TACT SWITCH RoHS Compliant	2	2 Ships Now	$0.33	**$0.66**
21	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	512-SS36FA SS36FA onsemi Customer # Add	Schottky Diodes & Rectifiers 60V 3A Schottky B arrier Rectifier RoHS Compliant By Exemption	3 **Packaging:** **Cut Tape	3 Ships Now	$0.53	**$1.59**
22	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	963-EMK316BBJ476ML-T EMK316BBJ476ML-T TAIYO YUDEN Customer # Add	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 16VDC 47uF 20% X5R RoHS Compliant	3 **Packaging:** **Cut Tape	3 Ships Now	$0.58	**$1.74**
23	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	806-KLDX-0202-AC KLDX-0202-AC Kycon Customer # Add	DC Power Connectors 2mm PCB JACK CRIMPED CRIMPED LEADS RoHS Compliant	2	2 Ships Now	$0.70	**$1.40**
24	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	621-AP22615AWU-7 AP22615AWU-7 Diodes Incorporated Customer # Add	Power Switch ICs - Power Distribution 3.0A ADJ SINGLE CH PWR DIST SWITCH RoHS Compliant	2 **Packaging:** **Cut Tape	2 Ships Now	$0.94	**$1.88**
25	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	667-ERJ-6ENF2052V ERJ-6ENF2052V Panasonic Customer # Add	Thick Film Resistors - SMD 0805 20.5Kohms 1% AEC-Q200 RoHS Compliant By Exemption	10 **Packaging:** **Cut Tape	10 Ships Now	$0.084	**$0.84**
26	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	512-2N3904TFR 2N3904TFR onsemi Customer # Add	Bipolar Transistors - BJT NPN Transistor General Purpose RoHS Compliant	2	2 Ships Now	$0.38	**$0.76**
27					2 Ships Now	$1.25	**$2.50**

Sort	Product Detail		Description	Quantity	Availability	Unit Price (USD)	Ext. Price (USD)
	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	942-IRF7316TRPBF IRF7316TRPBF Infineon Customer # Add	MOSFET MOSFT DUAL PCh -30V 4.9A RoHS Compliant	2 **Packaging:** **Cut Tape	r		
28	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	356-ESP32WROOM-32D ESP32-WROOM-32D-N4 Espressif Customer # Add	Multiprotocol Modules SMD Module, ESP32-D0WD, 32Mbits SPI flash, UART mode, PCB antenna	2 **Packaging:** **Cut Tape	2 Ships Now	$4.08	**$8.16**
29	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	652-SRP7028A-150M SRP7028A-150M Bourns Customer # Add	Power Inductors - SMD 15uH 20% SMD 7028 AEC-Q200 RoHS Compliant	2 **Packaging:** **Cut Tape	2 Ships Now	$1.38	**$2.76**
30	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	667-ERJ-8ENF1542V ERJ-8ENF1542V Panasonic Customer # Add	Thick Film Resistors - SMD 1206 15.4Kohms 1% AEC-Q200 RoHS Compliant By Exemption	4 **Packaging:** **Cut Tape	4 Ships Now	$0.18	**$0.72**
31	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	71-CRCW1206-205K-E3 CRCW1206205KFKEA Vishay Customer # Add	Thick Film Resistors - SMD 1/4watt 205Kohms 1% RoHS Compliant By Exemption	10 **Packaging:** **Cut Tape	10 Ships Now	$0.072	**$0.72**
32	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	810-CGA1A2C0G1E680J CGA1A2C0G1E680J030BA TDK Customer # Add	Multilayer Ceramic Capacitors MLCC - SMD/SMT CGA 0201 25V 68pF C0G 5% AEC-Q200 RoHS Compliant	1 **Packaging:** **Cut Tape	1 Ships Now	$0.10	**$0.10**
33	 **Mouser #:** **Mfr. #:** **Mfr.:** **Customer #:**	652-SRR1260-150M SRR1260-150M Bourns Customer # Add	Power Inductors - SMD 15uH 20% SMD 1260 RoHS Compliant	2 **Packaging:** **Cut Tape	2 Ships Now	$1.26	**$2.52**
	View Additional Product Info						

12.7 Device's Code

```
#include <FastLED.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include "BluetoothSerial.h"


#define UPDATES_PER_SECOND 100

#define DINPin 12  // // Pin connected to the data input of the LED panel mara case eka
#define DMOSPin 13

#define OLED_ADDR 0x3C  // OLED display I2C address
#define OLED_WIDTH 128  // OLED display width, in pixels
#define OLED_HEIGHT 64  // OLED display height, in pixels

#define LED_TYPE SK6812
#define NUM_LEDS 20
#define COLOR_ORDER GRB


// // Check if Bluetooth configs are enabled
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

// // Bluetooth Serial object
BluetoothSerial SerialBT;


//const int standbyWorkingPin = 19;  // Pin connected to standby_working switch sensor VN
//const int oledPin = 3; // Pin connected to OLED display
const int appManualPin = 16;
//const int studiDiscoPin = 39;

const int colorButtonPin = 32;  // Pin connected to the color button
const int brightnessButtonPin = 34;
const int patternButtonPin = 33;

//const int appManualPin = 33;
//const int studiDiscoPin = 16;
const int micValue = 27;


Adafruit_SSD1306 display(OLED_WIDTH, OLED_HEIGHT, &Wire, -1);
volatile int colorOption;
volatile int appColorOption;
volatile int brightnessOption;
volatile int patternOption;
volatile int counter;
volatile int discoValue;

volatile int appManualControl;

bool gReverseDirection = false;

CRGBPalette16 currentPalette;
TBlendType currentBlending;

extern CRGBPalette16 myRedWhiteBluePalette;
extern const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM;

CRGB leds[NUM_LEDS];

```

```
64
65 String message = "";
66 char incomingChar;
67
68 void showStrip() {
69   FastLED.show();
70 }
71
72
73 String getColorName(uint8_t index) {
74   switch (index) {
75     case 0:
76       return "Red";
77     case 1:
78       return "Blue";
79     case 2:
80       return "Green";
81     case 3:
82       return "Yellow";
83     case 4:
84       return "White";
85   }
86 }
87 String getBrightnessLevel(uint8_t index) {
88   switch (index) {
89     case 0:
90       return "10%";
91     case 1:
92       return "20%";
93     case 2:
94       return "30%";
95     case 3:
96       return "40%";
97     case 4:
98       return "50%";
99     case 5:
100      return "60%";
101    case 6:
102      return "70%";
103    case 7:
104      return "80%";
105    case 8:
106      return "90%";
107    case 9:
108      return "100%";
109  }
110 }
111 String getPatternName(uint8_t index) {
112   switch (index) {
113     case 0:
114       return "Solid";
115     case 1:
116       return "Fire2012";
117     case 2:
118       return "Color Palatte";
119     case 3:
120       return "Fade all";
121     case 4:
122       return "Color snake";
123   }
124 }
125
126
127 String appControlMode(uint8_t index) {
128   switch (index) {
129     case 0:
130       return "App Control";
131     case 1:
```

```
      return "Manual Control";
  }
}

void activateLEDPanel() {
  digitalWrite(DMOSPin, HIGH);  // Set DMOS pin to high signal
  //delay(100);                   // Wait for the panel to activate (adjust
    delay time if needed)
}


void updateOLED(const char* message) {
  delay(20);
  display.clearDisplay();       // Clear the OLED display buffer
  display.setTextSize(1);       // Set the text size
  display.setTextColor(WHITE);  // Set the text color to white
  display.setCursor(10, 10);    // Set the text position
  display.println(message);     // Print the provided message on the display
     buffer
  display.display();            // Display the updated buffer on the OLED
    display
}
//////////////////////////////////////App connect
    ////////////////////////////////////////////////////
void setColor(uint8_t red, uint8_t green, uint8_t blue) {
  int brightness2 = 250;
  for (int i = 0; i < NUM_LEDS; i++) {
    leds[i].setRGB(red, green, blue);
    leds[i].nscale8_video(brightness2);
  }
  FastLED.show();
}

void increaseBrightness(int q) {
  // Code to increase brightness of the LED panel
  for (int i = 0; i < NUM_LEDS; i++) {

    int brightness2 = (q + 1) * 51;
    leds[i].nscale8_video(brightness2);
    q++;
  }
  FastLED.show();
}
void decreaseBrightness(int q) {
  // Code to increase brightness of the LED panel
  for (int i = 0; i < NUM_LEDS; i++) {

    int brightness2 = (q - 1) * 51;
    leds[i].nscale8_video(brightness2);
    q--;
  }
  FastLED.show();
}

void handleAppControl(String incomingString) {
  if (incomingString.length() > 0) {
    char incomingChar = incomingString.charAt(0);
    switch (incomingChar) {
      case 'R':  // Red color
        // setColor(255, 0, 0);
        if (patternOption == 0) {
          colorOption = 0;
          setColor(255, 0, 0);
        } else {
          updateOLED("Update to Pattern 1 ");
        }
        break;

```

```
196       case 'G':  // Green color
197         // setColor(0, 255, 0);
198         if (patternOption == 0) {
199           colorOption = 2;
200           setColor(0, 255, 0);
201         } else {
202           updateOLED("Update to Pattern 1 ");
203         }
204         break;
205 
206       case 'B':  // Blue color
207         // setColor(0, 0, 255);
208         if (patternOption == 0) {
209           colorOption = 1;
210           setColor(0, 0, 255);
211         } else {
212           updateOLED("Update to Pattern 1 ");
213         }
214         break;
215 
216       case 'W':  // White color
217         // setColor(255, 255, 255);
218         if (patternOption == 0) {
219           colorOption = 4;
220         } else {
221           updateOLED("Update to Pattern 1 ");
222         }
223         break;
224 
225       case '+':  // Increase brightness
226         //increaseBrightness(5);
227         if (brightnessOption >= 9) {
228           brightnessOption = 9;
229         } else {
230           brightnessOption++;
231         }
232         break;
233 
234       case '-':  // Decrease brightness
235         //decreaseBrightness(5);
236         if (brightnessOption <= 0) {
237           brightnessOption = 0;
238         } else {
239           brightnessOption--;
240         }
241         break;
242 
243       case 'S':  // Decrease brightness
244         //decreaseBrightness(5);
245         patternOption = 0;
246         //Serial.println("-");
247         break;
248 
249       case 'F':  // Decrease brightness
250         //decreaseBrightness(5);
251         patternOption = 3;
252         //Serial.println("-");
253         break;
254 
255       case 'N':  // Decrease brightness
256         //decreaseBrightness(5);
257         patternOption = 4;
258         //Serial.println("-");
259         break;
260 
261       case 'D':  // Decrease brightness
262         //decreaseBrightness(5);
263         patternOption = 0;
```

```
        colorOption = 5;
        discoValue = analogRead(micValue);
        if (discoValue < 50) {
          setColor(238, 130, 238);
        } else if (discoValue < 100) {
          setColor(75, 0, 130);
        } else if (discoValue < 150) {
          setColor(0, 0, 255);
        } else if (discoValue < 200) {
          setColor(0, 128, 0);
        } else if (discoValue < 250) {
          setColor(255, 255, 0);
        } else if (discoValue < 300) {
          setColor(255, 165, 0);
        } else if (discoValue > 300) {
          setColor(255, 0, 0);
        }
        //Serial.println("-");
        break;

      default:
        break;
    }
  }
}




// void selectPattern() {
//   // Code to select pattern on the LED panel (if applicable)
//   // Example: Blink all LEDs
//   for (int i = 0; i < numLeds; i++) {
//     ledStrip.setPixelColor(i, ledStrip.Color(255, 255, 255));
//   }
//   ledStrip.show();
//   delay(500);
//   ledStrip.clear();
//   ledStrip.show();
// }
////////////////////////////////////////appcontrol finished
    ////////////////////////////////////////////




////////////////////////////////////////LED LIBRARIES
    ////////////////////////////////////////////////
#define COOLING 55
// SPARKING: What chance (out of 255) is there that a new spark will be lit?
// Higher chance = more roaring fire.  Lower chance = more flickery fire.
// Default 120, suggested range 50-200.
#define SPARKING 120

void Fire2012() {
  // Array of temperature readings at each simulation cell
  static uint8_t heat[NUM_LEDS];

  // Step 1.  Cool down every cell a little
  for (int i = 0; i < NUM_LEDS; i++) {
    heat[i] = qsub8(heat[i], random8(0, ((COOLING * 10) / NUM_LEDS) + 2));
  }

  // Step 2.  Heat from each cell drifts 'up' and diffuses a little
  for (int k = NUM_LEDS - 1; k >= 2; k--) {
    heat[k] = (heat[k - 1] + heat[k - 2] + heat[k - 2]) / 3;
  }

```

```
  // Step 3.  Randomly ignite new 'sparks' of heat near the bottom
  if (random8() < SPARKING) {
    int y = random8(7);
    heat[y] = qadd8(heat[y], random8(160, 255));
  }

  // Step 4.  Map from heat cells to LED colors
  for (int j = 0; j < NUM_LEDS; j++) {
    CRGB color = HeatColor(heat[j]);
    int pixelnumber;
    if (gReverseDirection) {
      pixelnumber = (NUM_LEDS - 1) - j;
    } else {
      pixelnumber = j;
    }
    leds[pixelnumber] = color;
  }
}

void FillLEDsFromPaletteColors(uint8_t colorIndex) {
  uint8_t brightness = 255;

  for (int i = 0; i < NUM_LEDS; ++i) {
    leds[i] = ColorFromPalette(currentPalette, colorIndex, brightness,
    currentBlending);
    colorIndex += 3;
  }
}


void ChangePalettePeriodically() {
  uint8_t secondHand = (millis() / 1000) % 60;
  static uint8_t lastSecond = 99;

  if (lastSecond != secondHand) {
    lastSecond = secondHand;
    if (secondHand == 0) {
      currentPalette = RainbowColors_p;
      currentBlending = LINEARBLEND;
    }
    if (secondHand == 10) {
      currentPalette = RainbowStripeColors_p;
      currentBlending = NOBLEND;
    }
    if (secondHand == 15) {
      currentPalette = RainbowStripeColors_p;
      currentBlending = LINEARBLEND;
    }
    if (secondHand == 20) {
      SetupPurpleAndGreenPalette();
      currentBlending = LINEARBLEND;
    }
    if (secondHand == 25) {
      SetupTotallyRandomPalette();
      currentBlending = LINEARBLEND;
    }
    if (secondHand == 30) {
      SetupBlackAndWhiteStripedPalette();
      currentBlending = NOBLEND;
    }
    if (secondHand == 35) {
      SetupBlackAndWhiteStripedPalette();
      currentBlending = LINEARBLEND;
    }
    if (secondHand == 40) {
      currentPalette = CloudColors_p;
      currentBlending = LINEARBLEND;
    }
```

```
    if (secondHand == 45) {
      currentPalette = PartyColors_p;
      currentBlending = LINEARBLEND;
    }
    if (secondHand == 50) {
      currentPalette = myRedWhiteBluePalette_p;
      currentBlending = NOBLEND;
    }
    if (secondHand == 55) {
      currentPalette = myRedWhiteBluePalette_p;
      currentBlending = LINEARBLEND;
    }
  }
}

// This function fills the palette with totally random colors.
void SetupTotallyRandomPalette() {
  for (int i = 0; i < 16; ++i) {
    currentPalette[i] = CHSV(random8(), 255, random8());
  }
}

void SetupBlackAndWhiteStripedPalette() {
  // 'black out' all 16 palette entries...
  fill_solid(currentPalette, 16, CRGB::Black);
  // and set every fourth one to white.
  currentPalette[0] = CRGB::White;
  currentPalette[4] = CRGB::White;
  currentPalette[8] = CRGB::White;
  currentPalette[12] = CRGB::White;
}

// This function sets up a palette of purple and green stripes.
void SetupPurpleAndGreenPalette() {
  CRGB purple = CHSV(HUE_PURPLE, 255, 255);
  CRGB green = CHSV(HUE_GREEN, 255, 255);
  CRGB black = CRGB::Black;

  currentPalette = CRGBPalette16(
    green, green, black, black,
    purple, purple, black, black,
    green, green, black, black,
    purple, purple, black, black);
}


const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM = {
  CRGB::Red,
  CRGB::Gray,  // 'white' is too bright compared to red and blue
  CRGB::Blue,
  CRGB::Black,

  CRGB::Red,
  CRGB::Gray,
  CRGB::Blue,
  CRGB::Black,

  CRGB::Red,
  CRGB::Red,
  CRGB::Gray,
  CRGB::Gray,
  CRGB::Blue,
  CRGB::Blue,
  CRGB::Black,
  CRGB::Black
};

void fadeall() {
```

```
  for (int i = 0; i < NUM_LEDS; i++) { leds[i].nscale8(250); }
}

// ////////////////////////////////////////////////////////////////////////////////////////




void setup() {
  delay(3000);  // power-up safety delay
  // FastLED.addLeds<LED_TYPE, DINPin, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(TypicalLEDStrip);
  //FastLED.setBrightness( digitalRead(brightnessButtonPin) );
  //FastLED.addLeds<NEOPIXEL, DINPin>(strip, NUM_LEDS);
  FastLED.addLeds<LED_TYPE, DINPin, GRB>(leds, NUM_LEDS).setCorrection(TypicalLEDStrip);
  //FastLED.addLeds<NEOPIXEL, DINPin>(leds, NUM_LEDS);

  Serial.begin(115200);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
      ;
  }

  //pinMode(standbyWorkingPin, INPUT);
  pinMode(appManualPin, INPUT);
  //pinMode(studiDiscoPin, INPUT);

  pinMode(brightnessButtonPin, INPUT);
  pinMode(colorButtonPin, INPUT_PULLUP);
  pinMode(patternButtonPin, INPUT_PULLUP);
  pinMode(micValue, INPUT);
  discoValue = 0;
  counter = 0;
  colorOption = 0;
  brightnessOption = 0;
  patternOption = 0;



  pinMode(DMOSPin, OUTPUT);

  currentPalette = RainbowColors_p;
  currentBlending = LINEARBLEND;

  // Serial.begin(115200);
  // // Bluetooth device name
  SerialBT.begin("DuliniESP32");
  Serial.println("The device started, now you can pair it with bluetooth!");
  String displayMessage = "Smart Lamp";
}

void loop() {

  bool mode = digitalRead(appManualPin) == LOW;
  if (mode) {
    appManualControl = 0;
    if (SerialBT.available()) {
      char incomingChar = SerialBT.read();

      message = String(incomingChar);

      handleAppControl(message);
      delay(20);
```

```
    }
  }

  else {
    appManualControl = 1;
    if (counter >= 10) {
      int brightness = digitalRead(brightnessButtonPin);
      int Color = digitalRead(colorButtonPin);
      int pattern = digitalRead(patternButtonPin);

      if (pattern == LOW) {
        if (patternOption == 4) {
          patternOption = 0;
        } else {
          patternOption++;
        }
      }

      if (brightness == HIGH) {
        if (brightnessOption == 9) {
          brightnessOption = 0;
        } else {
          brightnessOption++;
        }
      }

      if (Color == LOW) {
        if (patternOption == 0) {
          if (colorOption == 4) {
            colorOption = 0;
          } else {
            colorOption++;
          }
        } else {
          updateOLED("Update to Pattern 1 ");
        }
      }
      counter = 0;
    } else {
      counter++;
    }
  }
  activateLEDPanel();
  int brightness1 = (brightnessOption + 1) * 25;

  switch (patternOption) {
    case 0:  // Solid color

      switch (colorOption) {

        case 0:  // Red
          for (int i = 0; i < NUM_LEDS; i++) {
            leds[i].setRGB(255, 0, 0);
            leds[i].nscale8_video(brightness1);
          }
          FastLED.show();

          //delay(10);
          break;
        case 1:  // Blue
          for (int i = 0; i < NUM_LEDS; i++) {
            leds[i].setRGB(0, 0, 255);
            leds[i].nscale8_video(brightness1);
          }
          FastLED.show();

          //delay(10);
          break;
```

```
        case 2:  // Green
          for (int i = 0; i < NUM_LEDS; i++) {
            leds[i].setRGB(0, 255, 0);
            leds[i].nscale8_video(brightness1);
          }
          FastLED.show();

          //delay(10);
          break;
        case 3:  // Yellow
          for (int i = 0; i < NUM_LEDS; i++) {
            leds[i].setRGB(255, 255, 0);
            leds[i].nscale8_video(brightness1);
          }
          FastLED.show();

          //delay(10);
          break;
        case 4:  // White
          for (int i = 0; i < NUM_LEDS; i++) {
            leds[i].setRGB(255, 255, 255);
            leds[i].nscale8_video(brightness1);
          }
          FastLED.show();

          //delay(10);
          break;

        case 5:  // White
          //delay(10);
          break;
      }
      break;
    case 1:
      Fire2012();
      showStrip();
      break;
    case 2:
      ChangePalettePeriodically();

      static uint8_t startIndex = 0;
      startIndex = startIndex + 1; /* motion speed */

      FillLEDsFromPaletteColors(startIndex);
      showStrip();
      break;
    case 3:
      static uint8_t hue = 0;
      // First slide the led in one direction
      for (int i = 0; i < NUM_LEDS; i++) {
        // Set the i'th led to red
        leds[i] = CHSV(hue++, 255, 255);
        leds[i].nscale8_video(brightness1);
        // Show the leds
        FastLED.show();
        // now that we've shown the leds, reset the i'th led to black
        // leds[i] = CRGB::Black;
        fadeall();
        // Wait a little bit before we loop around and do it again
        delay(5);
      }
      for (int i = (NUM_LEDS)-1; i >= 0; i--) {
        // Set the i'th led to red
        leds[i] = CHSV(hue++, 255, 255);
        leds[i].nscale8_video(brightness1);
        // Show the leds
        FastLED.show();
        // now that we've shown the leds, reset the i'th led to black
```

```
        // leds[i] = CRGB::Black;
        fadeall();
        // Wait a little bit before we loop around and do it again
        delay(5);
      }
      break;
    case 4:
      for (int i = 0; i < NUM_LEDS; i++) {
        // set our current dot to red
        leds[i] = CRGB::Red;
        FastLED.show();
        // clear our current dot before we move on
        leds[i] = CRGB::Black;
        leds[i].nscale8_video(brightness1);
        delay(5);
      }

      for (int i = NUM_LEDS - 1; i >= 0; i--) {
        // set our current dot to red
        leds[i] = CRGB::Red;

        FastLED.show();
        // clear our current dot before we move on
        leds[i] = CRGB::Black;
        leds[i].nscale8_video(brightness1);
        delay(5);
      }
      break;
  }

  String displayMessage = appControlMode(appManualControl) + "\nStudio Mode"
                                                            "\nColor: "
                          + getColorName(colorOption) + "\nBrightness: " +
    getBrightnessLevel(brightnessOption) + "\nPattern: " + getPatternName(
    patternOption);
  updateOLED(displayMessage.c_str());
  delay(25);
}
```

Listing 1: Device Code