MOOD LAMP

Interim Report

Edited By
MONIS RAZA

Centre for Electronic Design and Technology
Netaji Subhas University of Technology
New Delhi, India

Contents

1 Acknowledgement **2**

2 Introduction **3**

3 Six Box Model **4**

4 Project Description **7**
4.1 Block Diagram 7
4.2 Hardware Description 8
4.3 Software Description 13

5 Circuit Design **14**
5.1 LED Panel 14
5.2 Control Board 15

6 Fabricated PCB **17**

7 Code **19**

8 End Result **21**

1 Acknowledgement

I extend my deepest gratitude to Prof. Dhananjay V. Gadre, Associate Professor, ECE Division at Netaji Subhas University of Technology. His constant support and guidance are what led to the completion of this project. CEDT gave me the optimal environment to learn, research, and implement my newly acquired knowledge.

My thanks and appreciation also goes to my friends and seniors at the centre who have helped me out willingly throughout the course of this project.

I would also like to express gratitude to my family who has always supported me in my endeavours.

2 Introduction

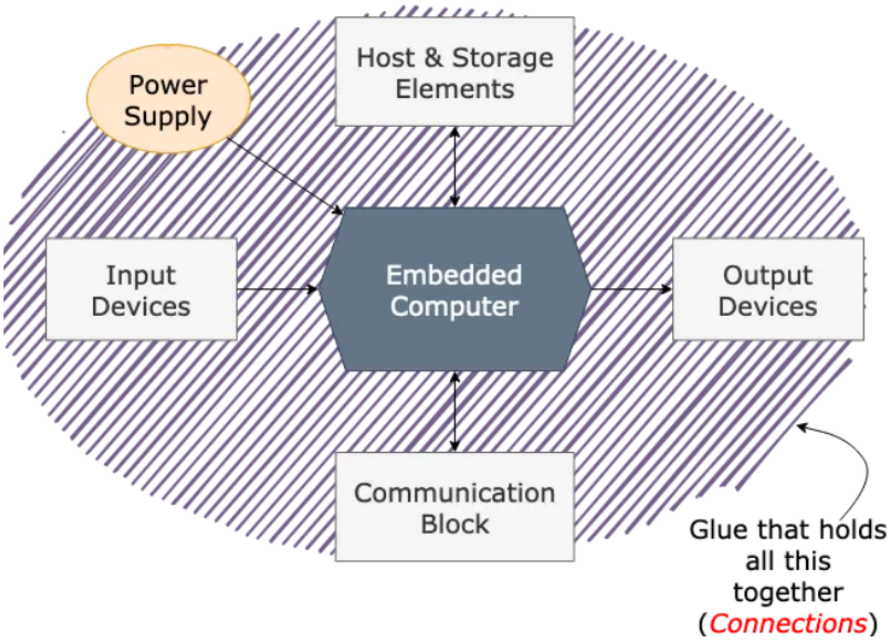
'Mood Lamp' as the name suggests, enables its user to lighten their room with the colour of their liking. The lamp has two modes of operation-

1. Manually changing the red, blue, and green intensities by turning the knobs provided on the enclosure.
2. Contact-less control using an android app using Bluetooth as the technology for wireless communication.

ESP32 WROOM is used as the microcontroller with 3W RGB LEDs being the illuminant driven by a constant current driving circuit.

3 Six Box Model

Every embedded system can be categorised into the following six blocks of operation-[1]



In order to break down our device into these basic building blocks, let's list down the requirements of our device:

- An input mechanism to vary the R, G, B intensities manually.
- A wireless methodology to vary the R, G, B intensities from a distance.
- An LED panel serves as the output of our system.
- A microcontroller to generate PWM signals controlling the LED circuit.
- A constant current circuit to drive the LEDs.

- A supply to power the whole circuit.
- An enclosure to contain the entire circuitry and be the lamp's body.

1. INPUTS: The manual input by turning the knobs is realised by using three 100k panel mount potentiometers whose values are read by the ADC of the microcontroller. In the wireless mode, the inputs are received by the Bluetooth module integrated into the microcontroller and sent by an android app. There is also another input given by a rocker switch deciding the mode of operation of the lamp.

Figure 1: 100k potentiometer

2. OUTPUT: The output of the system is represented by the three 3W RGB SMD LEDs located on the LED panel lighting up the lamp. There is also a power-on LED indicating whether the system is connected to a power source or not.

Figure 2: 3W RGB LED

3. MICROCONTROLLER: ESP32-WROOM is the microcontroller of choice as it has multiple PWM channels and is integrated with Bluetooth and WiFi capabilities.

Figure 3: ESP32-WROOM

4. POWER SUPPLY: A 12V-2A DC adapter is used to give raw 12V which supplies the necessary bias to light up the LED panel. Since the microcontroller operates as 3.3V, LM2576 adjustable buck converter is used to provide the necessary voltage levels.
5. COMMUNICATIONS: An FTDI programmer is used to download the code from the IDE. A six-pin connector is added to the control board, this makes programming the microcontroller repeatedly a much easier task.

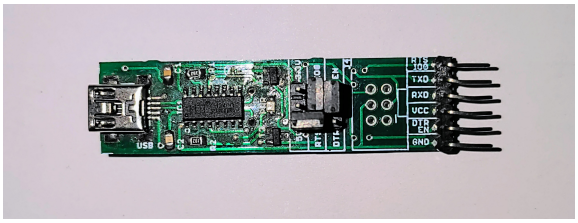


Figure 4: FTDI programmer

4 Project Description

4.1 Block Diagram

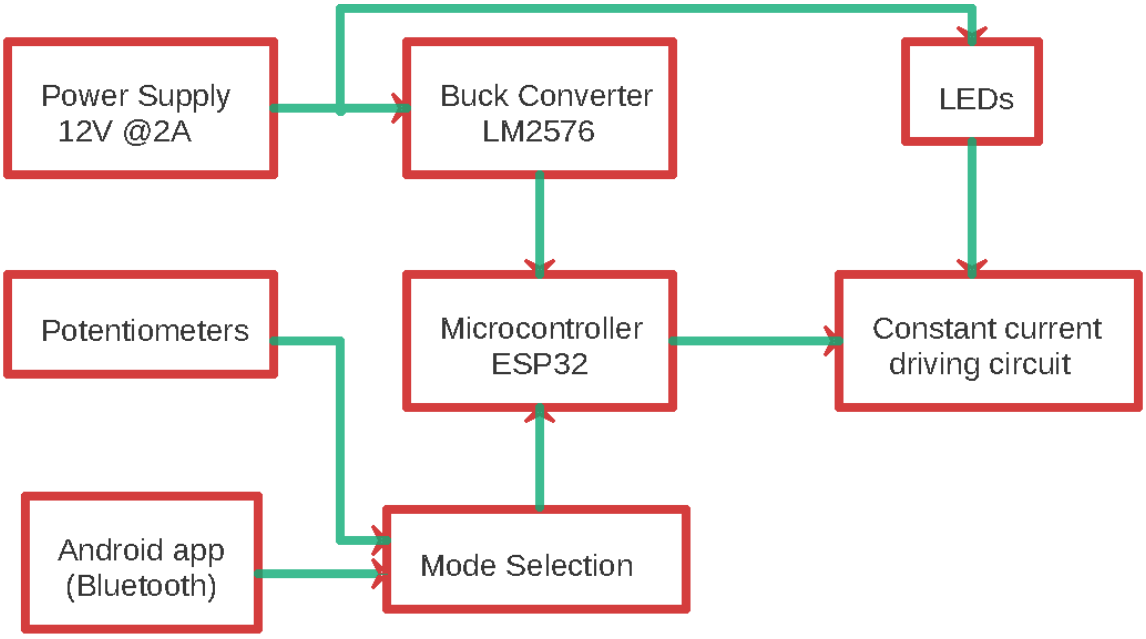


Figure 5: Block Diagram for the system

4.2 Hardware Description

The hardware of this project consists of two boards:

1. LED Panel: The board for LEDs is separate from the rest of the circuitry as they consist of high wattage elements which should be isolated to avoid unwanted transfer of heat to other components such as the microcontroller.

2. Control circuit: It consists of the microcontroller ESP32, the power supply block consisting of a DC barrel jack and the buck converter circuit. It also hosts the constant current driving circuit consisting of NPN transistors (MJE3055T), diodes (1N4007), and the accompanying resistors for biasing and setting the current to comply with the LED specifications and power ratings of the transistors. There are heat sinks for the transistors and the buck converter to take away excess heat to improve device performance and extend its life.

4.2.1 Working of the constant current driving circuit.[2]

In order to understand how the lamp works it is imperative to get familiar with the working of a simple constant current driving circuit serving as the foundation of this device.

The following points describe the nature of this circuit:

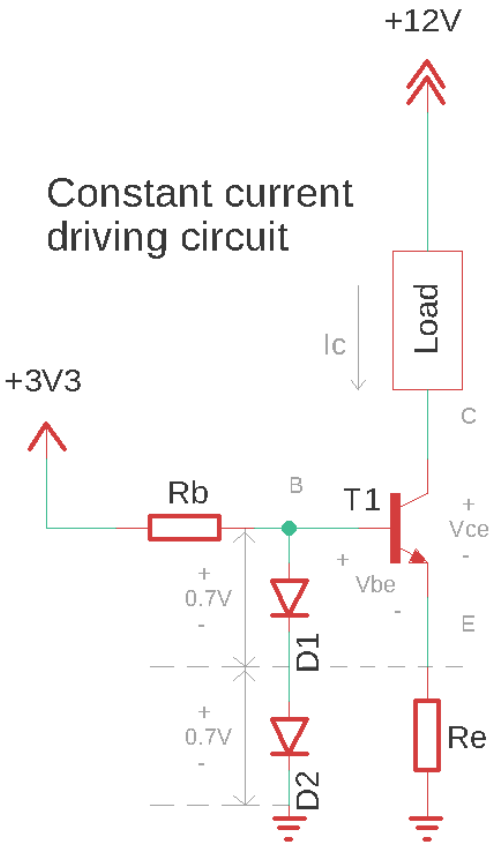
- T_1 is ON if-

$$V_{be} \geq V_{be}(on)$$

T_1 is OFF if-

$$V_{be} < V_{be}(on)$$

- The forward voltage drop across the two diodes connected to the base of T1 serves as a voltage reference for the constant current circuit.
- We know, $I_c = I_e + I_b$
  I_b being very small
  $\Rightarrow I_c \approx I_e$



$$I_c \approx I_e = \frac{V_{D1} + V_{D2} - V_{be}}{R_e}$$

$\Rightarrow R_e$ acts as the current setting resistor.

Wondering how the current is kept constant? Let's start with the collector current equation giving us insight into what controls the current through the load.[3]

$$I_c = I_s * e^{\frac{Vbe}{nVt}}$$

It is apparent that I_c is exponentially dependent on V_{be}.
Observe that if the current through the load increases, the voltage drop across Re also increases thereby decreasing V_{be} which eventually decreases I_c. This corrective action occurs because the voltage appearing at the base is kept constant by the voltage reference set by the diodes.

Thus the current I_c is kept relatively constant even if the voltage across the load changes.

Now the question arises as to how the intensity of LEDs is varied using the microcontroller. The answer to this question is given by the following heading-

4.2.2 LED Brightness and PWM[4]

PWM stands for 'Pulse Width Modulation and is used to generate an analog signal from a digital source, a microcontroller in our case.

Analog signals are known to have continuous values but a microcontroller only has two power states, ON(HIGH) and OFF(LOW).

In PWM the width of the pulse is modulated to change the average power delivered by the digital signal. This average power is controlled by controlling the ON and OFF time of the digital signal generated by the microcontroller.
PWM output is dependent on the following parameters:

1. **Duty Cycle:** The percentage of time in which the PWM signal remains HIGH (on time) is called the duty cycle. If the signal is always ON it is in 100% duty cycle and if it is always off it is 0% duty cycle. The formula to calculate the duty cycle is shown below.

$$Duty\,Cycle = \frac{Turn\,ON\,time}{Turn\,ON\,time + Turn\,OFF\,time}$$

2. **PWM Frequency:** The frequency of a PWM signal determines how fast a PWM completes one period. One Period is the complete ON and OFF time of a PWM signal as shown in the above figure. The formula to calculate the Frequency is given below.

$$Frequency = \frac{1}{Time\,Period}$$

3. **PWM Resolution:** Resolution of a PWM is the number of different steps you can have from zero power to full power. Implying, a 10-bit resolution means that you can have 1024 steps from zero to full power. Consider the example in which PWM is used to control

the intensity of an LED. Using a PWM of 10-bit resolution we can have 1024 different levels of brightness for the LED.

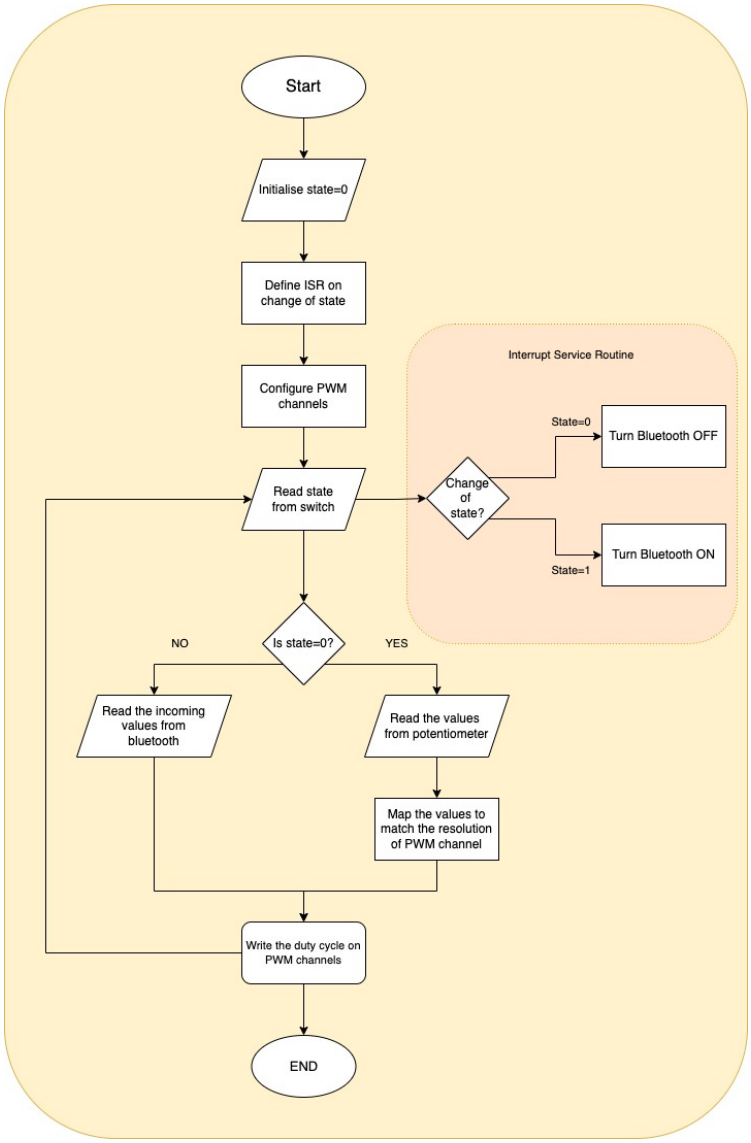
$$Output\,Voltage\,of\,PWM\,signal = Duty\,Cycle * V_{OH}$$

Now one might wonder that all that PWM is doing is turning the LED ON and OFF for different durations, how is that leading to different levels of brightness?[5]

'Persistence of vision' is the answer to this dilemma. Our eyes cannot perceive flickering beyond a certain frequency thus all we intake is the change in average power dissipated by the LED with the changing duty cycle of the PWM signal.

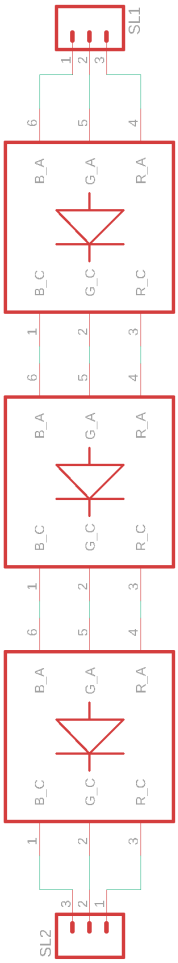
4.3 Software Description

The program for this project is written and compiled on Arduino IDE. The following flowchart explains the logic of code:

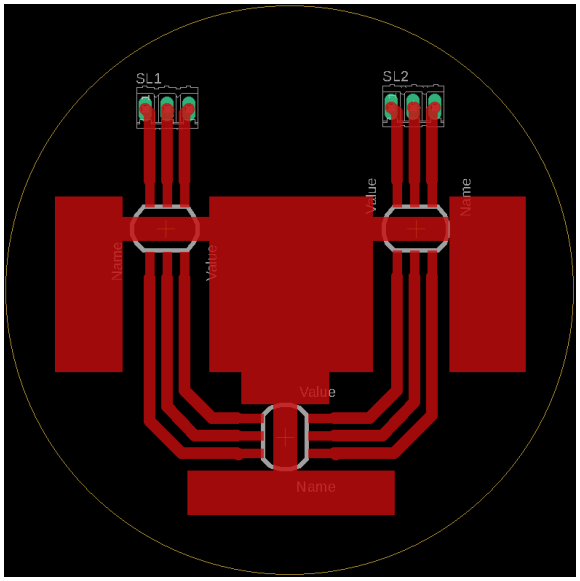


5 Circuit Design

5.1 LED Panel



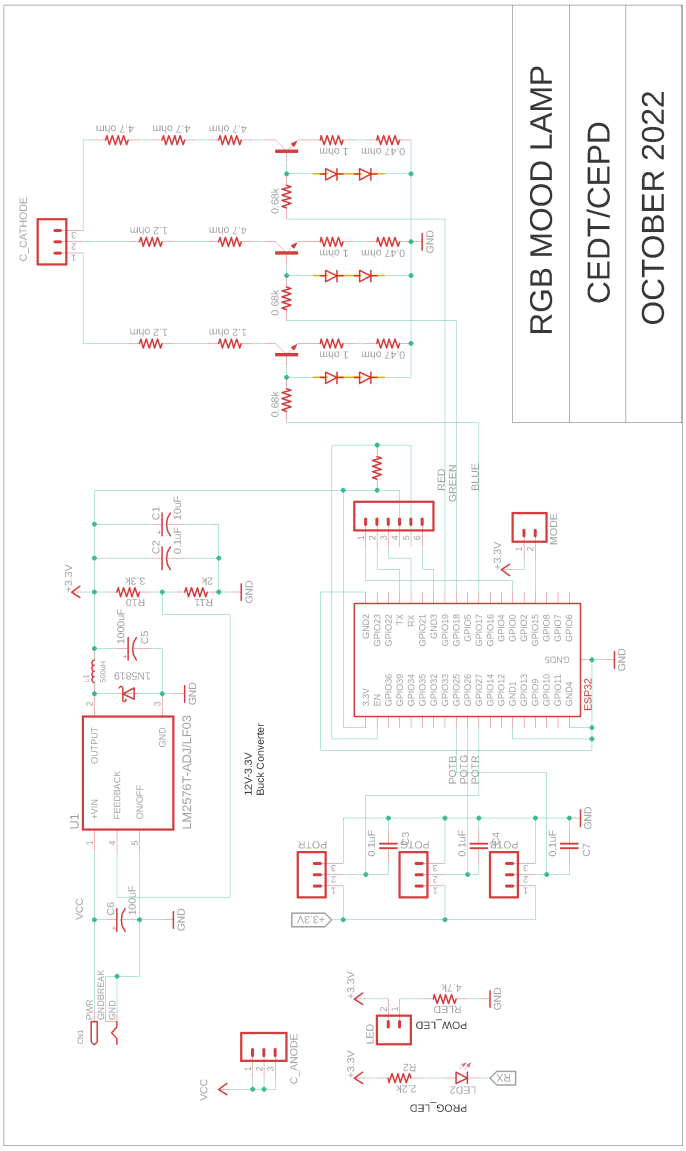
(a) Schematic.



(b) Board Layout.

Figure 6: Schematic and Board Layout of the LED panel.

5.2 Control Board



Schematic for Control Board

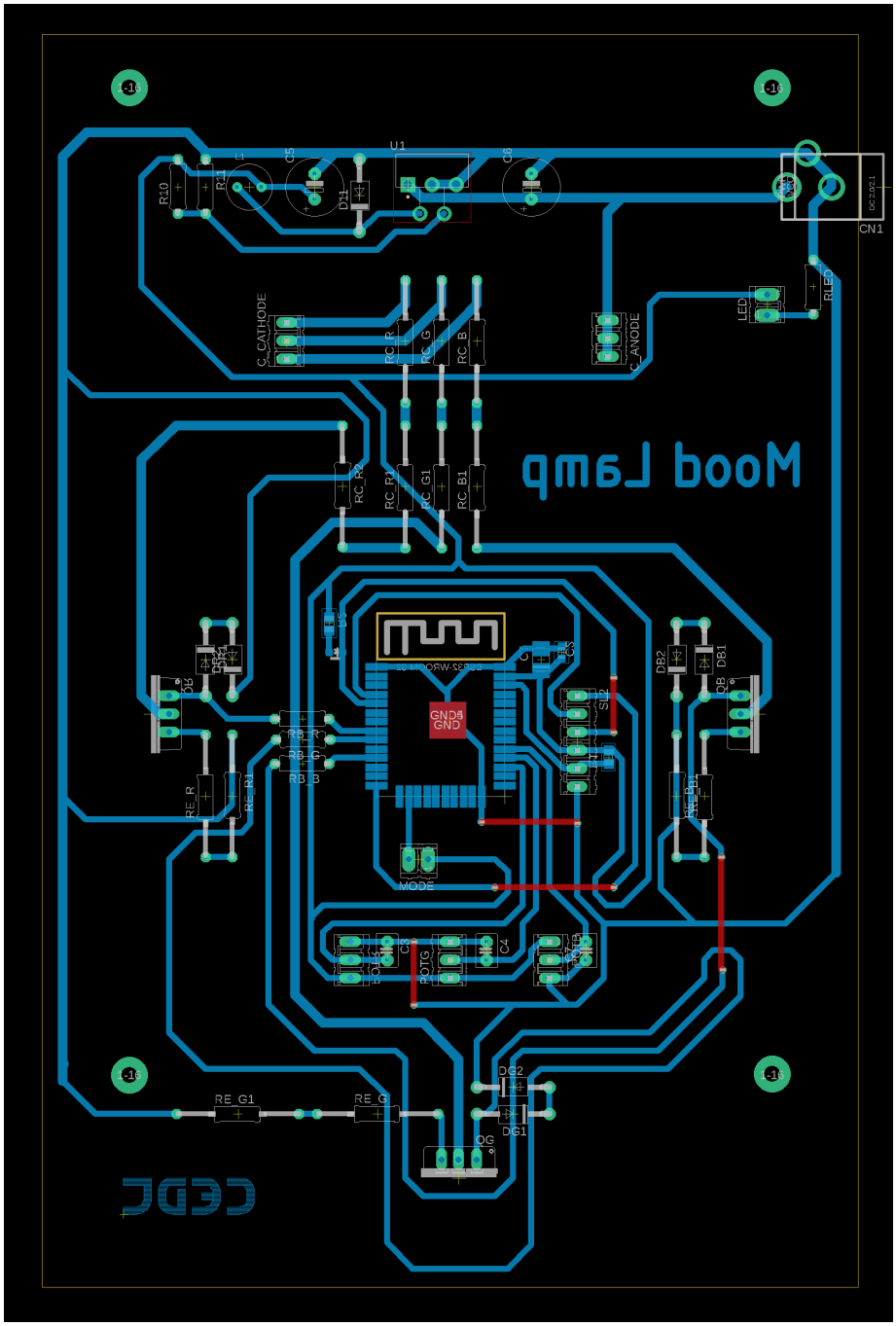


Figure 7: Board Layout

6 Fabricated PCB

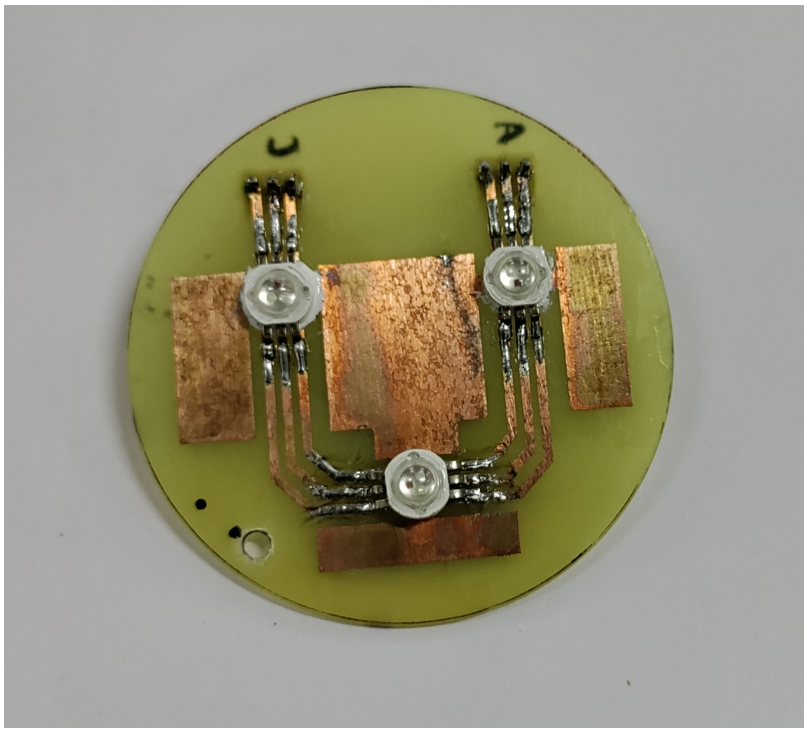


Figure 8: LED Panel

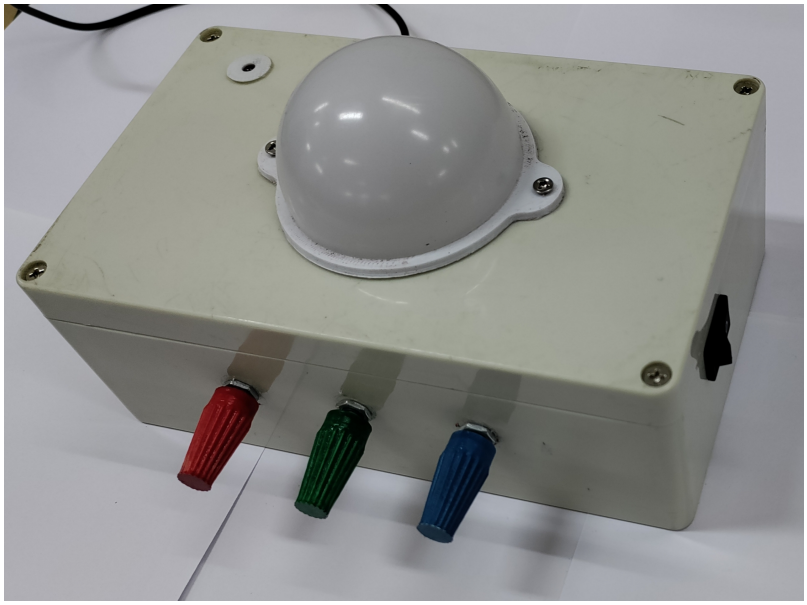
Figure 9: Control Board

7 Code

```
#include "BluetoothSerial.h"

/* Check if Bluetooth configurations are enabled in the SDK */
/* If not, then you have to recompile the SDK */
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

const int pot_R=27;
const int pot_G=26;
const int pot_B=25;

const int red=19;
const int green=18;
const int blue=17;

const int redChannel=1;
const int greenChannel=2;
const int blueChannel=3;

const int man=15;

uint8_t r=0;
uint8_t g=0;
uint8_t b=0;

uint8_t avg_r = 0;
uint8_t avg_g = 0;
uint8_t avg_b = 0;


//state=0 -> potentiometer mode
//state=1 -> bluetooth mode
int state = 0;

//Interrupt Service Routine
void IRAM_ATTR ISR(){

  state = digitalRead(man);

  if(state == 0){

    SerialBT.disconnect();
    SerialBT.end();

   }

   else{
    SerialBT.begin("MOOD LAMP");
    Serial.println("Bluetooth Started! Ready to pair...");
   }
}

```

```
void setup() {
  Serial.begin(115200);
  /* If no name is given, default 'ESP32' is applied */
  /* If you want to give your own name to ESP32 Bluetooth device, then */
  /* specify the name as an argument SerialBT.begin("myESP32Bluetooth"); */
   SerialBT.begin("MOOD LAMP");
   Serial.println("Bluetooth Started! Ready to pair...");

   pinMode(man,INPUT_PULLDOWN);
   attachInterrupt(man, ISR, CHANGE);

 //Configuring the PWM channels
   ledcAttachPin(red,redChannel);
   ledcAttachPin(green,greenChannel);
   ledcAttachPin(blue,blueChannel);

   ledcSetup(redChannel,5000, 8);
   ledcSetup(greenChannel,5000, 8);
   ledcSetup(blueChannel,5000, 8);
}

void loop() {
  // put your main code here, to run repeatedly:
  state = digitalRead(man);

  if(state == 0){  //Potentiometer mode
    for(int i=0;i<256;i++)
    {
    //Mapping 12-bit values to 8-bit
      r = analogRead(pot_R)>>4;
      g = analogRead(pot_G)>>4;
      b = analogRead(pot_B)>>4;

      avg_r += r;
      avg_g += g;
      avg_b += b;
    }
  //taking average to mitigate fluctuations
    avg_r /= 256;
    avg_g /= 256;
    avg_b /= 256;
  }
  else{           //Bluetooth Mode
    if(SerialBT.available()){
      r = SerialBT.read();
      g = SerialBT.read();
      b = SerialBT.read();
    }
  }

//Writing the duty cycles to respective
//channels
  ledcWrite(redChannel, r);
  ledcWrite(greenChannel, g);
  ledcWrite(blueChannel, b);

}
```

8 End Result



(a)

(b)

Figure 10: Enclosed lamp from different angles

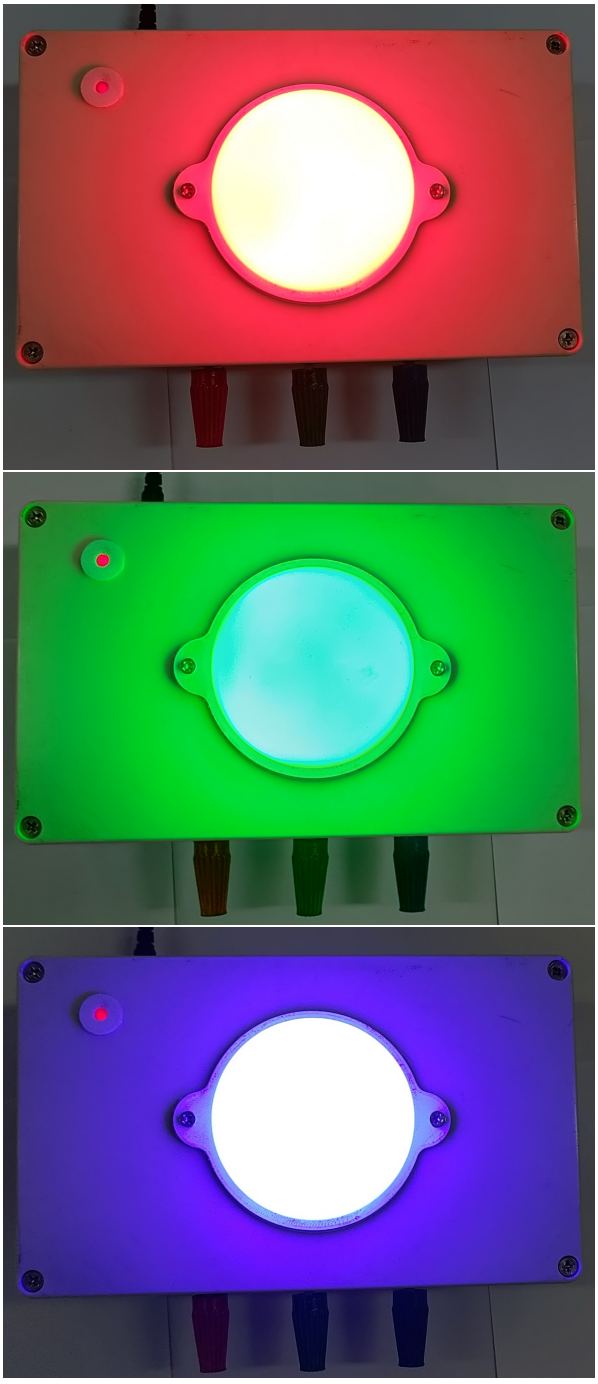


Figure 11: Mood Lamp showing different shades.

References

[1] S. Saha, "Medium.com." https://medium.com/spidernitt/enter-the-world-of-embedded-systems-9e02aafe06b7, Jul 2020. Accessed on 2012-11-29.

[2] A. Garaipoom, "eleccircuit." https://www.eleccircuit.com/constant-current-circuits-transistors/, Oct 2022. Accessed on 2012-11-22.

[3] J. Smith, "forum.arduino.cc." https://forum.arduino.cc/t/single-transistor-constant-current-power-led-driver/129531/2, Nov 2012. Accessed on 2012-11-23.

[4] S. Santos, "Randomnerdtutorials." https://randomnerdtutorials.com/esp32-pwm-arduino-ide/, Nov 2020. Accessed on 2012-11-25.

[5] A. Choudhary, "Circuit digest." https://circuitdigest.com/microcontroller-projects/esp32-pwm-tutorial-controlling-brightness-of-led, Jun 2020. Accessed on 2012-11-27.