Department of Electronic and Telecommunication Engineering
University of Moratuwa

EN3023 – Electronic Design Realization

Group Project – IoT Controlled SMART Power Extension Cord

Name	Index Number
190164M	E.M.S.S.N. Ekanayake
190356E	N.D. Liyanage
190484T	R.M.P.A.P. Rajapaksha
190497K	R.A.D.V. Ranasinghe

Table of Contents

Problem Description and Proposed Solution

In our usual day-to-day life, we often forget to switch off our devices in our household when we go out, and it causes power wastages, damages to the devices, burnings, and many other problems.

So, we proposed **an IOT based SMART power extension cord** which can be connected to your home Wi-Fi and allows you to

- Monitor and change the state of the sockets (on/off) and
- Set timers for individual sockets.

Our proposed design allows the user to control each socket outlet on the extension cord both physically and by wireless control through Wi-Fi. Thus, the user can control and monitor the devices connected to the cord from any place out of the house. Moreover, it provides other controlling features including setting timers and configuring the switches for separate devices through software means.

The online control of the device will happen through a webpage, instead of using a dedicated mobile app. This allows the users to utilize the device simply by using an internet browser via any smart device, thus no software installation is required. The users can create a private account for their devices on the webpage enabling them to manage the devices in a private and secure manner. To access the device through the webpage, you can either enter the device number in the webpage or scan the QR code printed on the front face of the device.

Idea Validation, Verification, and Justification

The design idea of this project was properly evaluated and validated by collecting ideas via a questionnaire shared through a google form. Most of the responses agreed with the necessity of a solution for the given problem, and they were satisfied with our proposed solution.

A smart power extension cord is more advantageous compared to a household socket outlet or a typical power extension cord. It is useful in saving power by turning off the switches that you usually forget to switch off when you leave the house. And it enables the user to monitor the daily usage of the electronic devices, and more importantly, it can be used to improve the life span of the electronic devices by setting timers to avoid over usage and overcharging.

With the functionalities specified above, our device can operate as a portable, affordable, and easy-to-use alternative to modern household *smart switches*, which require permanent installation and a higher cost.

Functionality

Physical User Interface

Our IOT based SMART power extension cord looks exactly like a usual power extension cord but can be controlled both physically and through Wi-Fi. Here, we have used momentary push buttons for the design (i.e., the state of the push button does not change after pressing) contrast to the usual 2-state push button designs. To indicate power being properly delivered to the outlets, power indicating LEDs have been included for each socket. Functionality is much simpler, when you push a push button the state of the socket will alternate.

Graphical User Interface (GUI)

The online control of the device will happen through a webpage, instead of using dedicated mobile app. So, no specific installation is required. Control it using any of your smart phones, laptops, and computers.

How to connect to a new Wi-Fi

When the ESP32 is switched on and if no Wi-Fi network is connected then the ESP32 acts as an access point. We can switch on Wi-Fi in our phones and connect to ESP32 by typing "password" as the password and then we can log into 192.168.4.1 and configure any available Wi-Fi connection by giving the ssid and password.

How to connect to GUI

We can connect to the user interface using http://20.197.2.117:1880/ui remotely from any internet accessing device. And we can get the dashboard to the home-screen from Google Chrome.

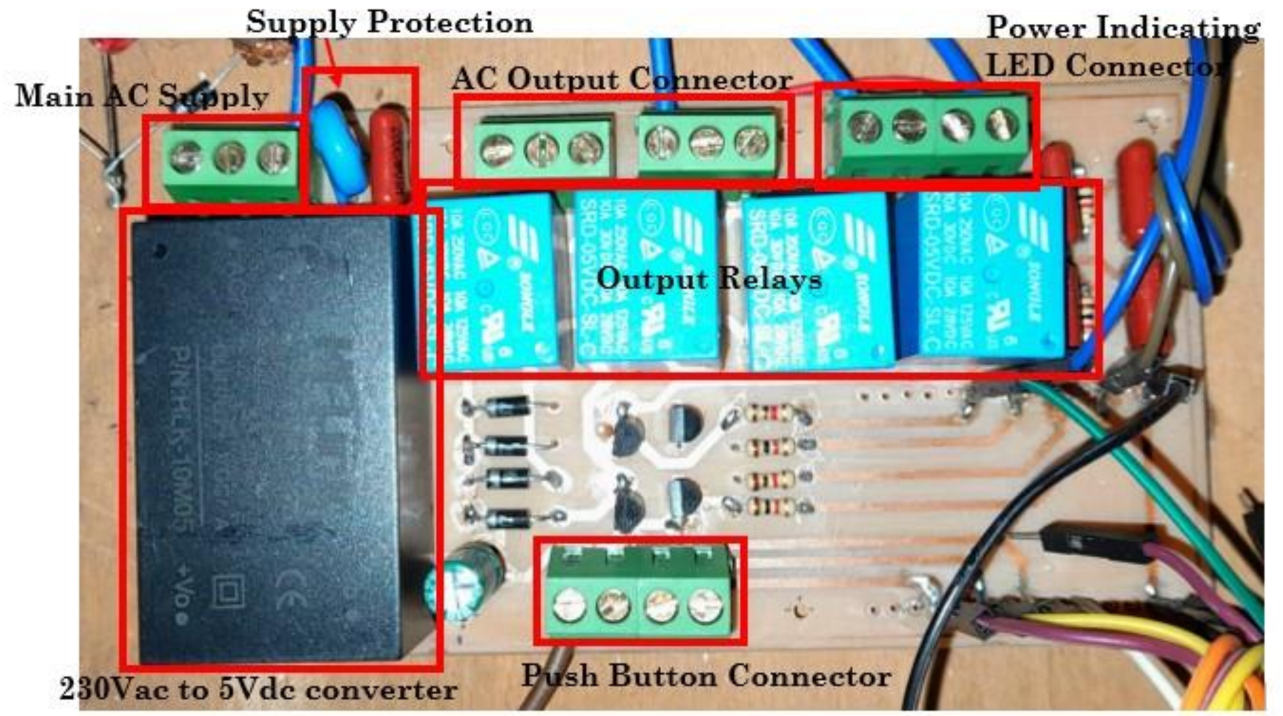
How to use the Device

Remote control can be done from any place since the user interface runs in the cloud. The data publishing and subscribing is done by a MQTT broker. Normal switching ON and OFF can be done by the UI. The state of the switch can be seen by the user in the user interface.

A timer is also there. First, we need to switch on the port and then set a timer value. Then after counting, the switch is turned off automatically and the state can be seen in the user interface. And to make the access to GUI easier, you can scan the QR Code given on the top of the device.

Circuity and Component Selection

The following figures shows our final PCB used for the demonstration.



The main components and output connectors are named in the above diagram. The 5.08mm pitch screw terminal blocks were used for each connector and AC outputs are always obtained from the first and last connections to avoid probable electrical short circuits. The Hi-Link 230V ac to 5V dc module gives a steady DC supply at 4.86V when all the output relays are closed. To provide the protection against electrical surges 10D561K varistor is added with the main supply. A Safety capacitor of 0.1uF, 400V is used in the input and another 220uF, 25V smoothing capacitor is used for the 5V dc output which keeps the ripple less than 30mV.

Power Indicating LED bulbs are directly driven by the AC supply provided to each socket instead of running it using the microcontroller esp32. To limit the current through the LEDs, a series 1kΩ resistor and a 400V, 0.1uF capacitor were used, and they are mounted onto the PCB. To get the maximum luminance, place the LED in a Wheatstone bridge of diodes.

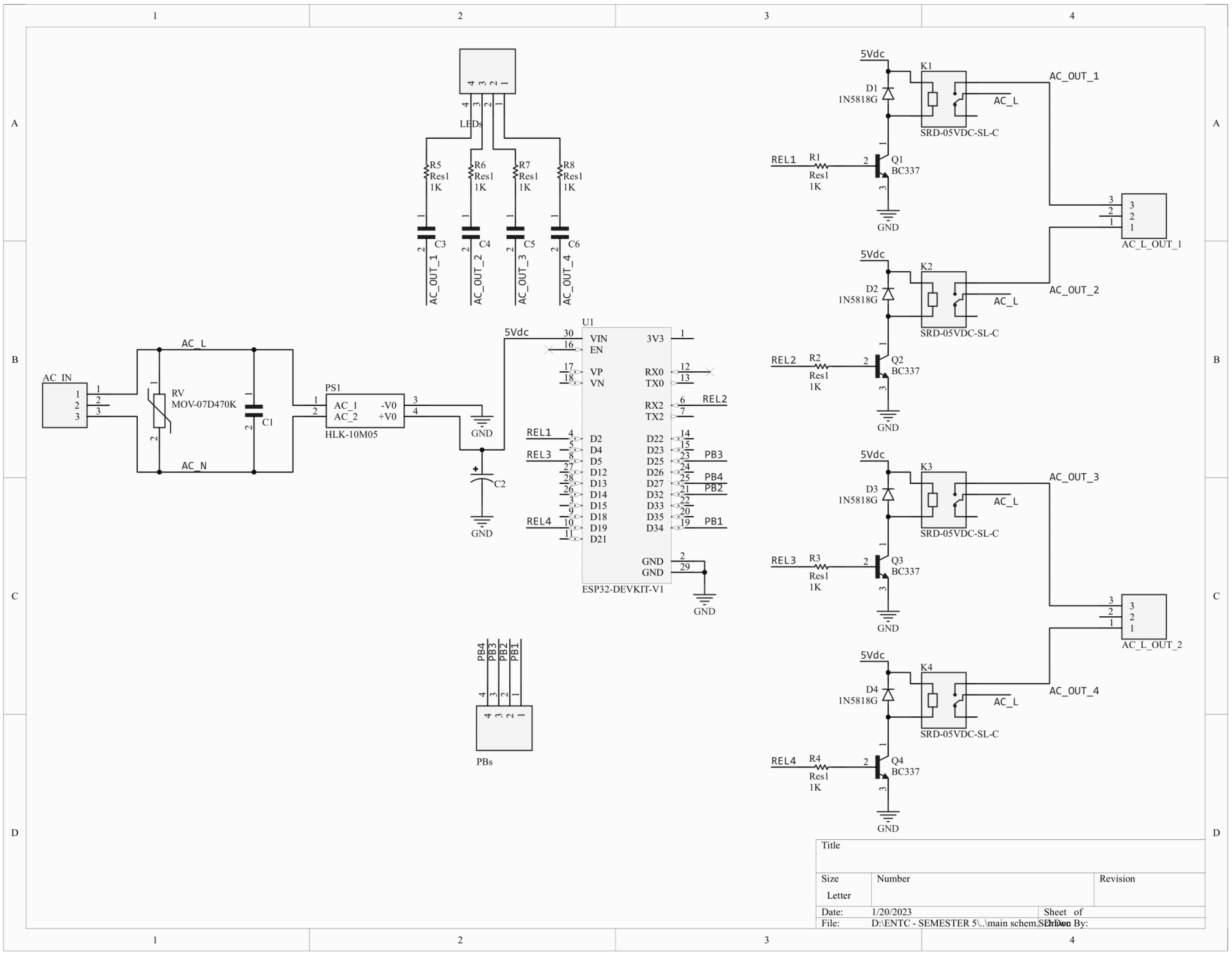
The circuit was designed to support up to maximum of 4 sockets where each outlet can withstand a maximum of 10A current draw. Also, since the extension cord is protected with another 10A fuse, the total current drawn from all the sockets must be limited to 10A as well.

PCB Design

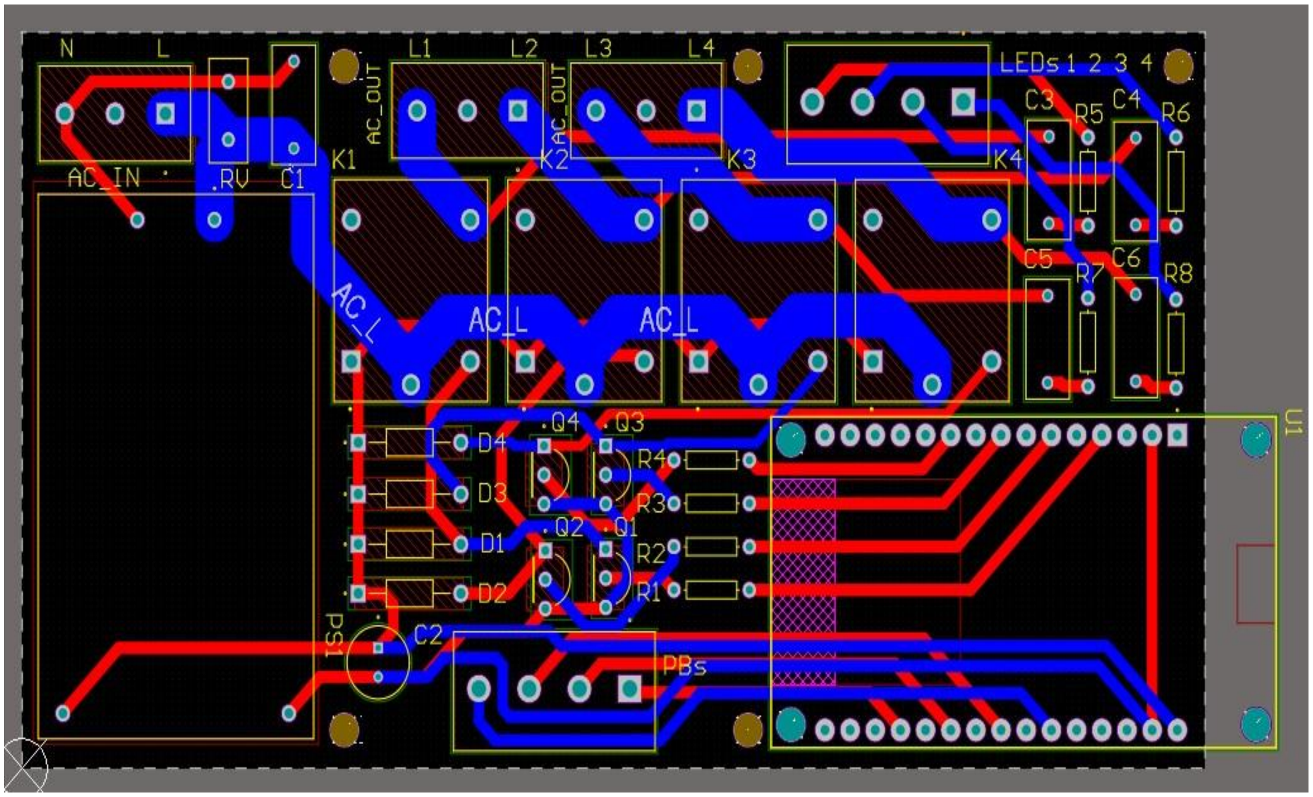
The PCB was designed as a two-layer circuit board and printed using double layer cupper clad boards. A trickier design point of the PCB was integrating the high current paths that carries up to 10A current onto the circuit board. To minimize the heating of such paths, the width of those traces was calculated using formulas from IPC-2221, and a width of 5mm was chosen. To support power dissipation further, the high current paths were laid on the bottom layer of the PCB where no electrical components were mounted so directly open to the air flow.

The design documents of the PCB are given in the following sections.

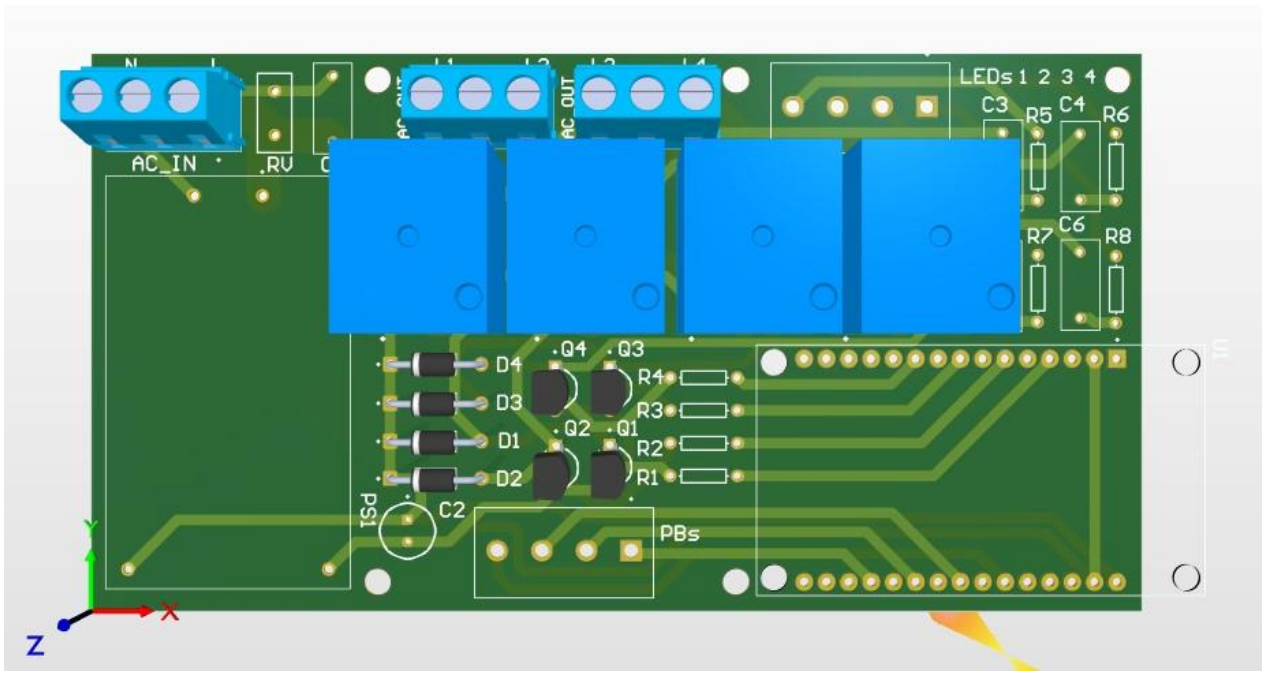
Schematics

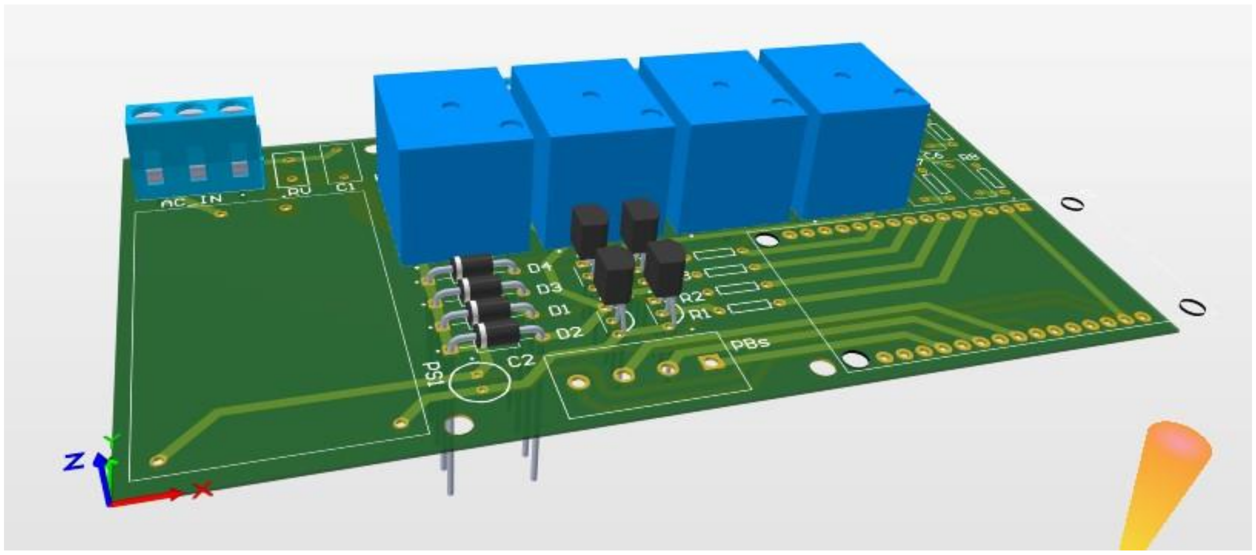


PCB layout (2D view)

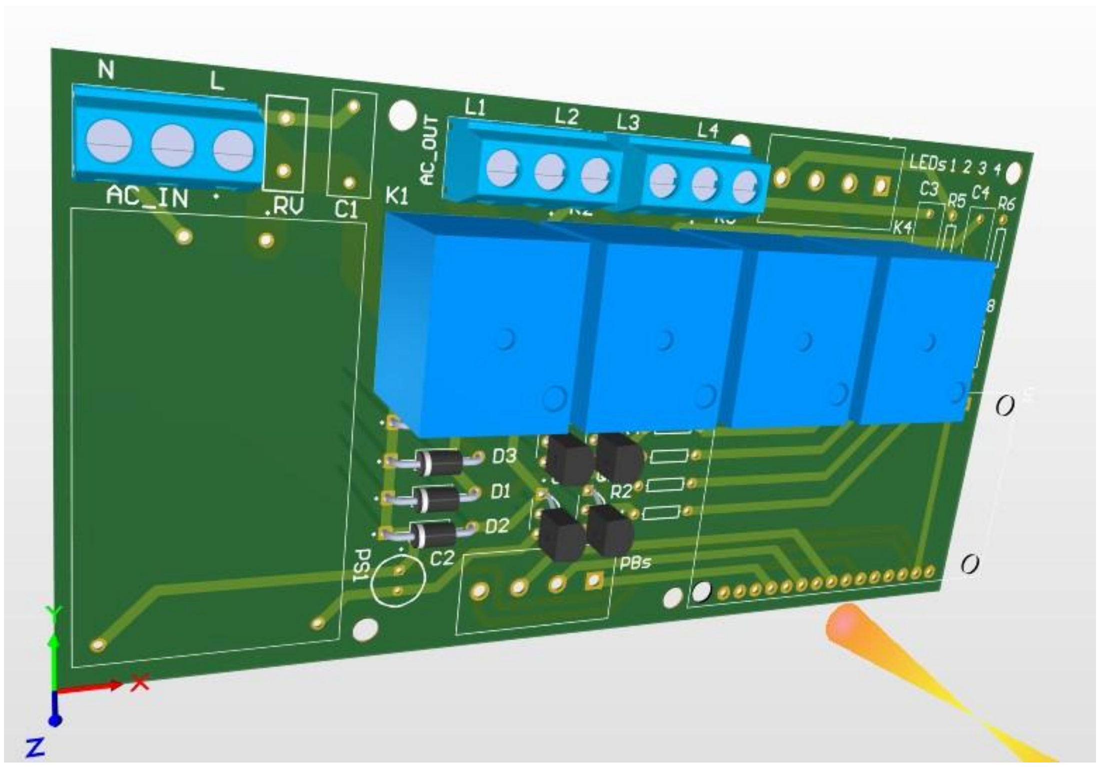


PCB Layout (3D views)





AC_IN
RV
C1
D4
D3
D1
D2
C2
PS1
R1
R2
PBs
C6
R8
Z
Y
X

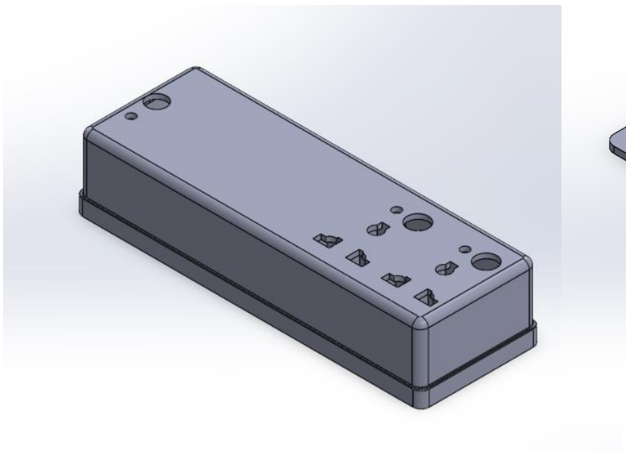


N
L
AC_IN
RV
C1
K1
AC_OUT
L1
L2
L3
L4
LEDs 1 2 3 4
C3
R5
C4
R6
K4
D3
D1
D2
R2
C2
PS1
PBs
Y
X
Z

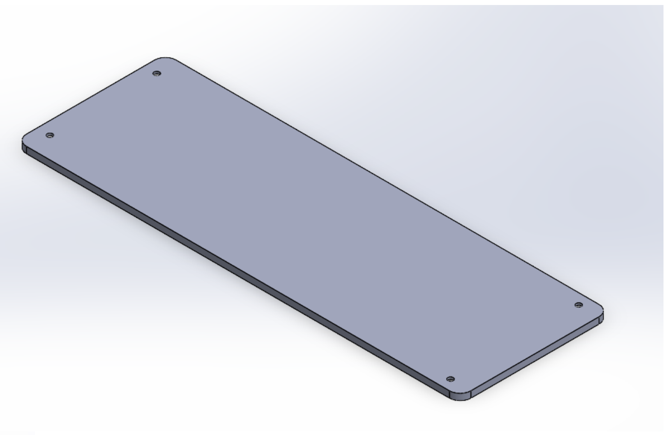
Enclosure Design

Overview

This section presents the design of the enclosure for the Smart Extension Cord. The enclosure was designed to be easy to install and use, which contains mainly two parts in the enclosure. The top part acts as a holder for all the components including PCB. Bottom part, which acts as a lid, is attached to the top part using four screws. The material used was PLA.



Top Part



Bottom Part

Design Process

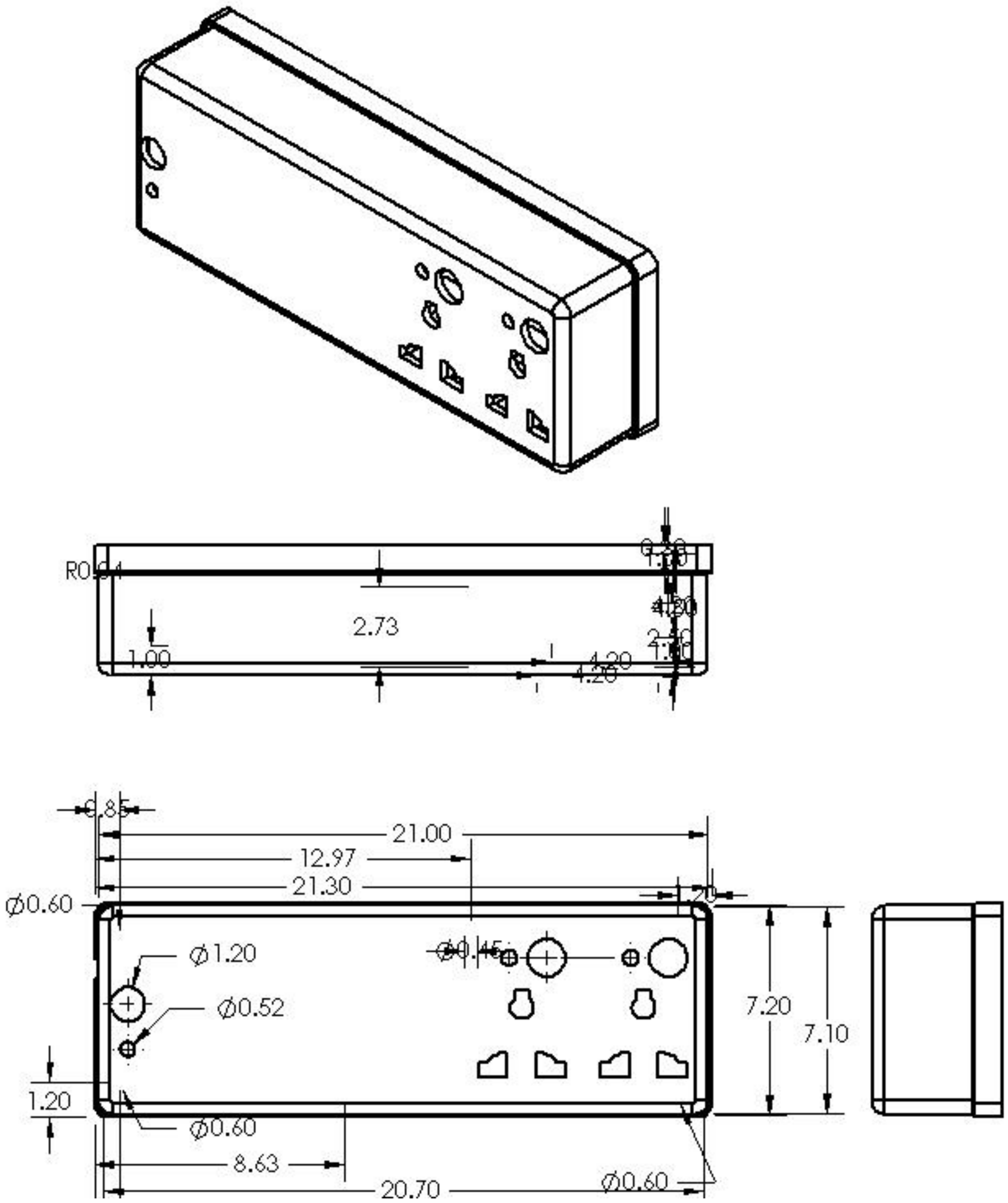
Research

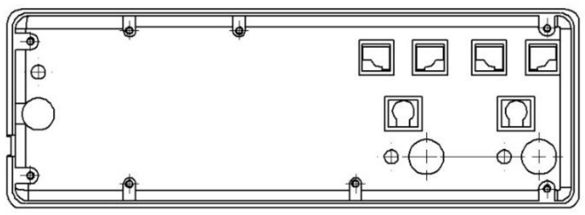
The research for modeling the enclosure was done by referring to an existing typical power extension cord used in Sri Lanka. Its physical and electrical connectivity between parts, spacing, and dimensions were considered when developing the design. Standard 3-pin Universal Power Sockets were included as the sockets.

Standard 3-pin Universal Socket

Drawings -Top Part

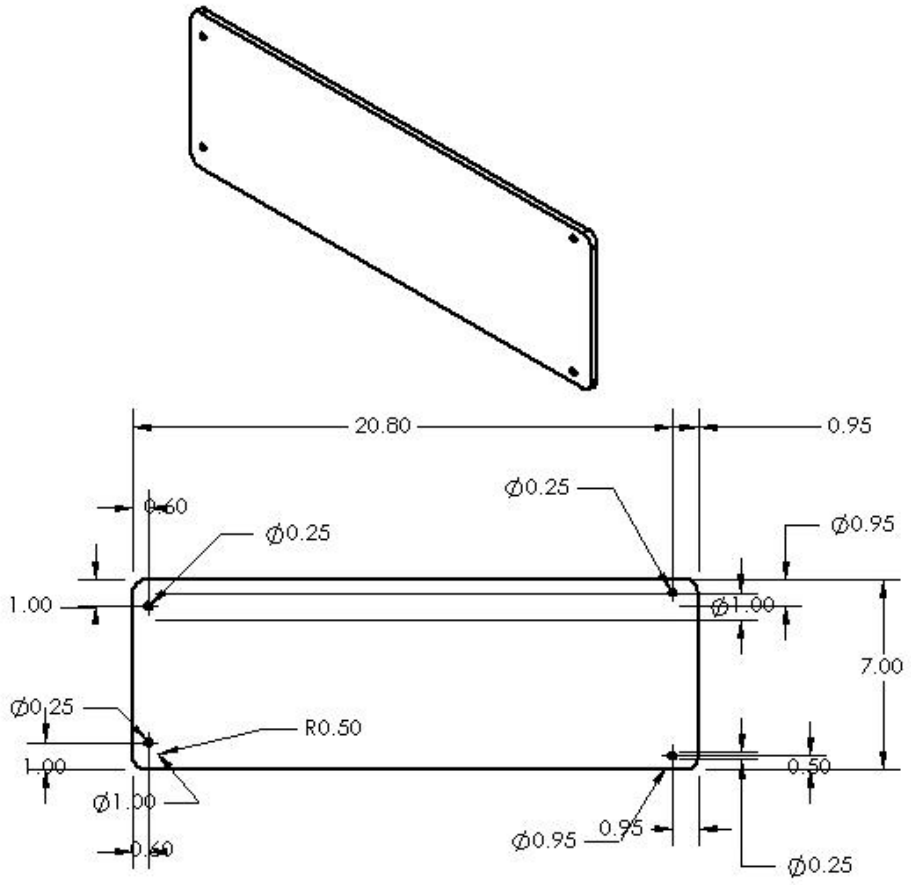
*Dimensions in CM


R0.04
2.73
1.00
0.20
1.10
4.20
4.20
4.20
2.50
1.50
0.85
21.00
12.97
21.30
.70
Ø0.60
Ø1.20
Ø0.45
Ø0.52
1.20
Ø0.60
8.63
20.70
Ø0.60
7.20
7.10



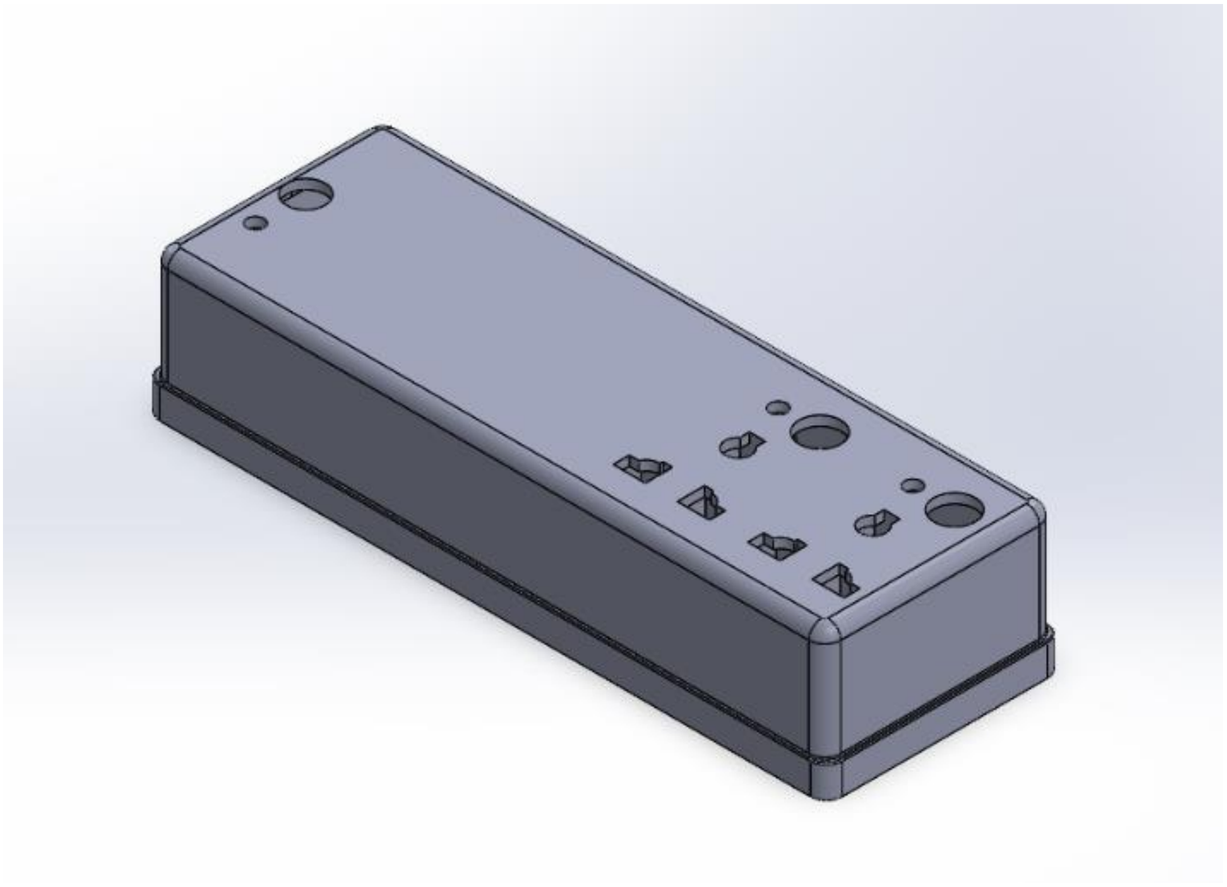
Drawings - Bottom Part

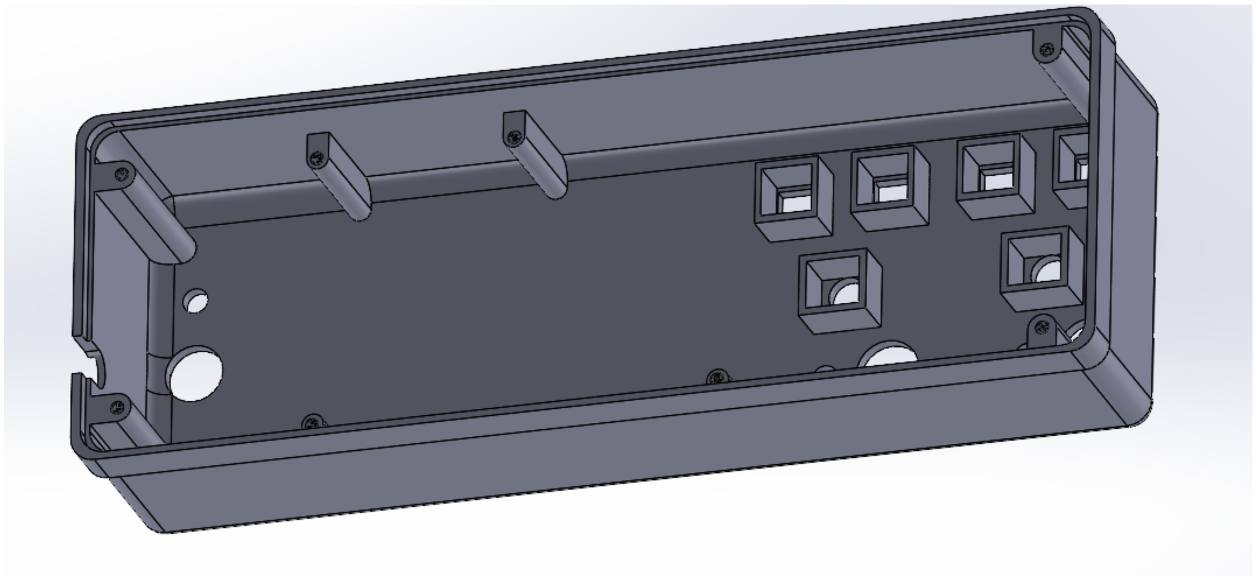
*Dimensions in CM

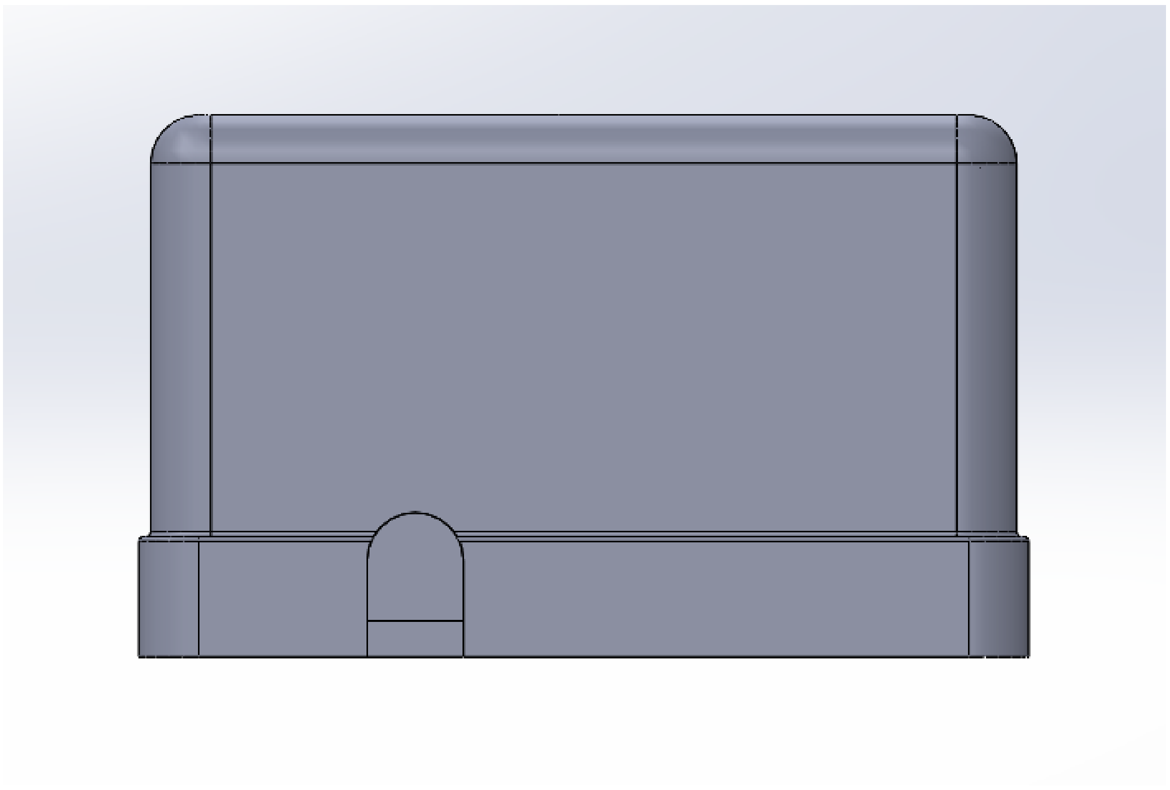


Images of Design

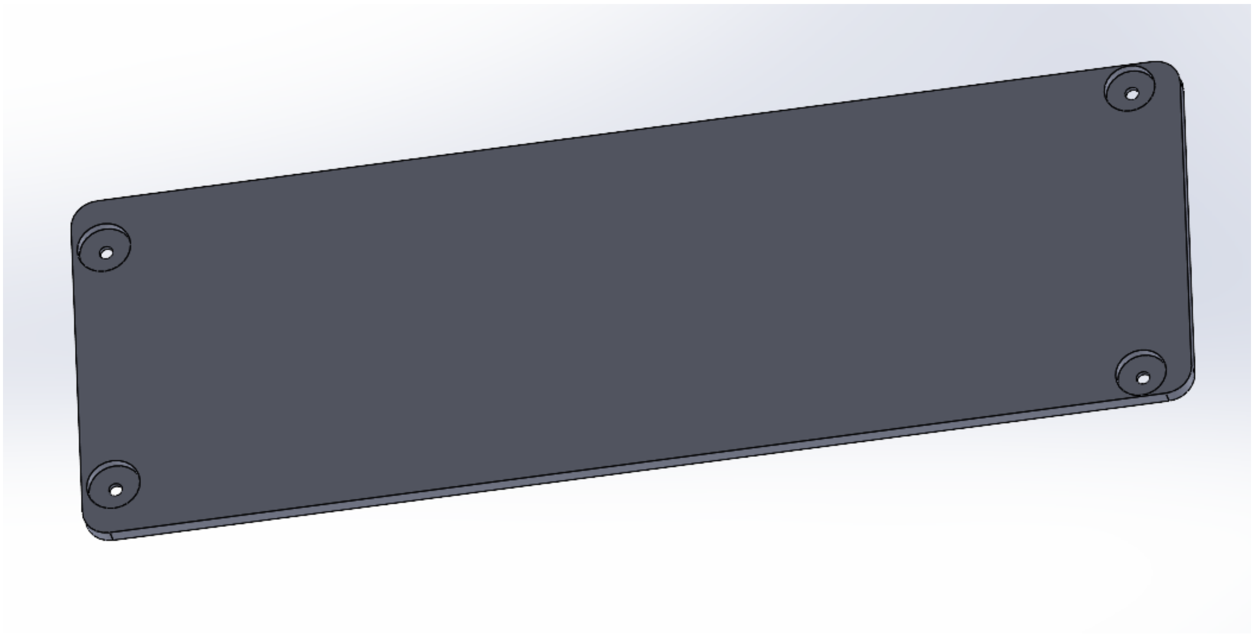
Top Part



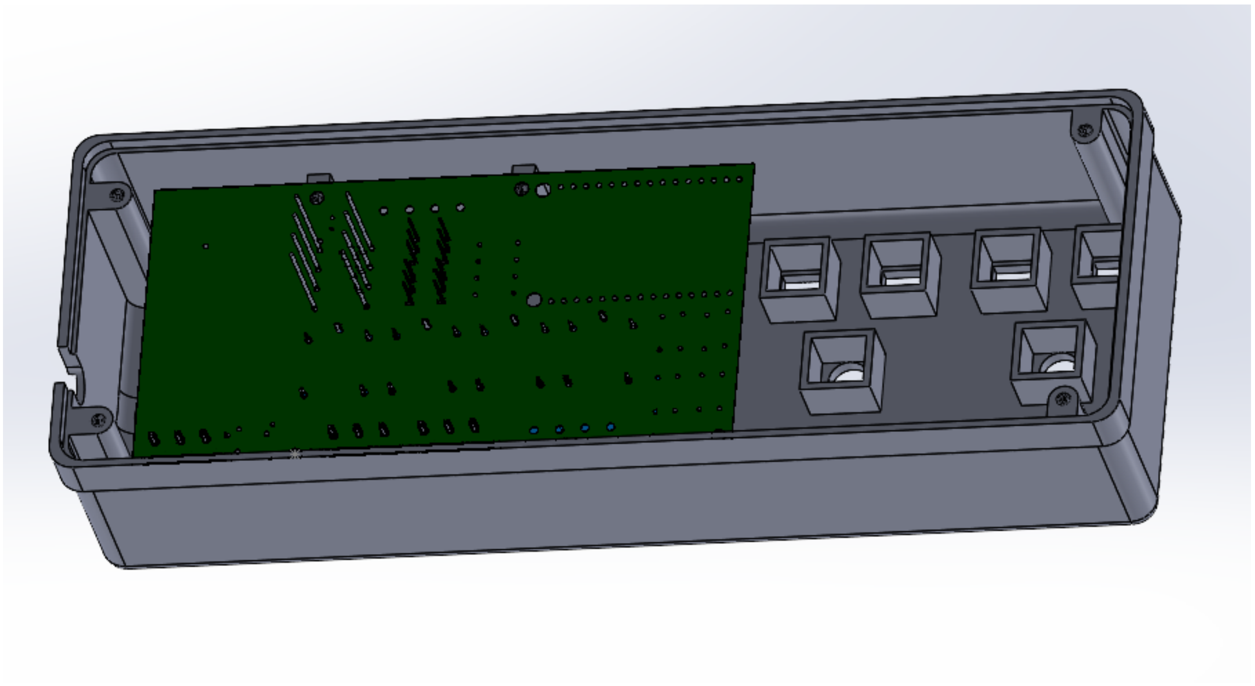


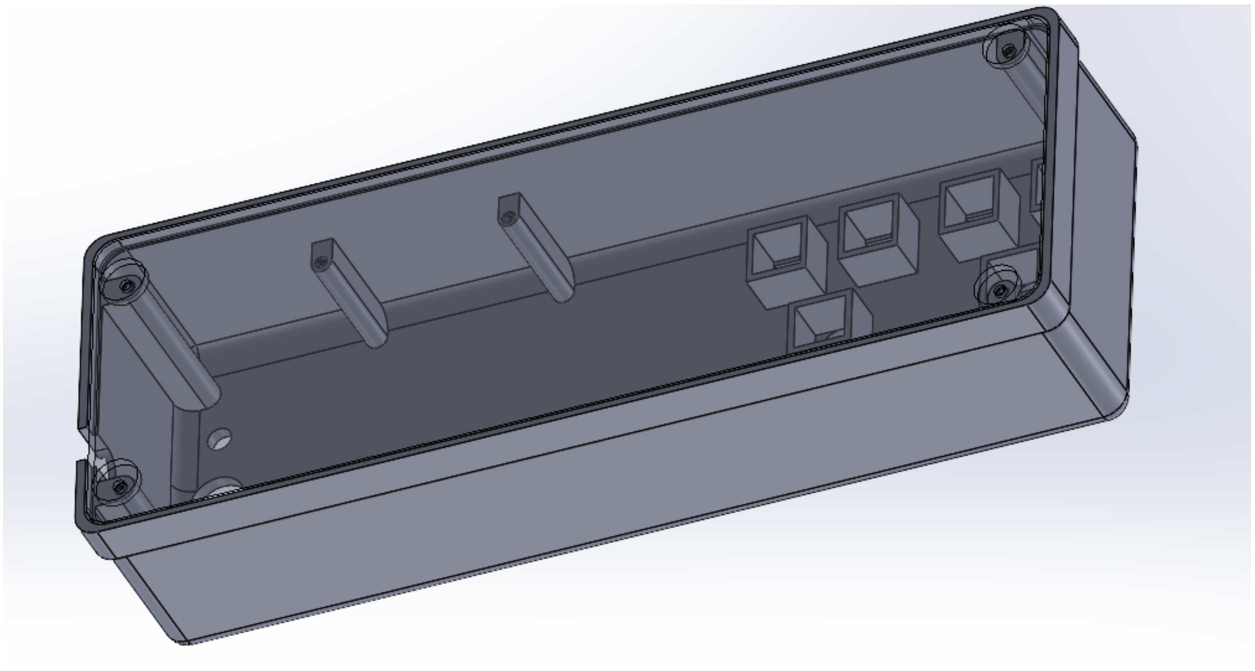


Bottom Part

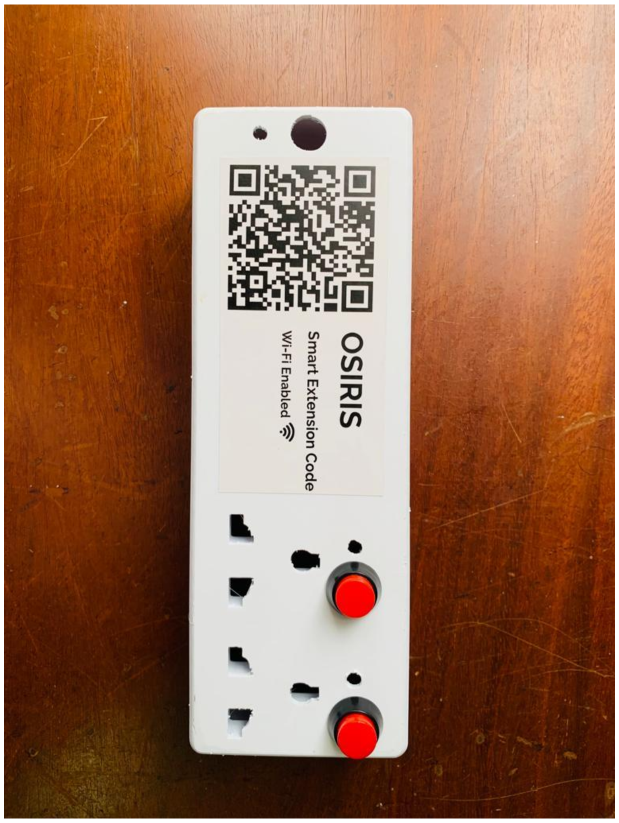


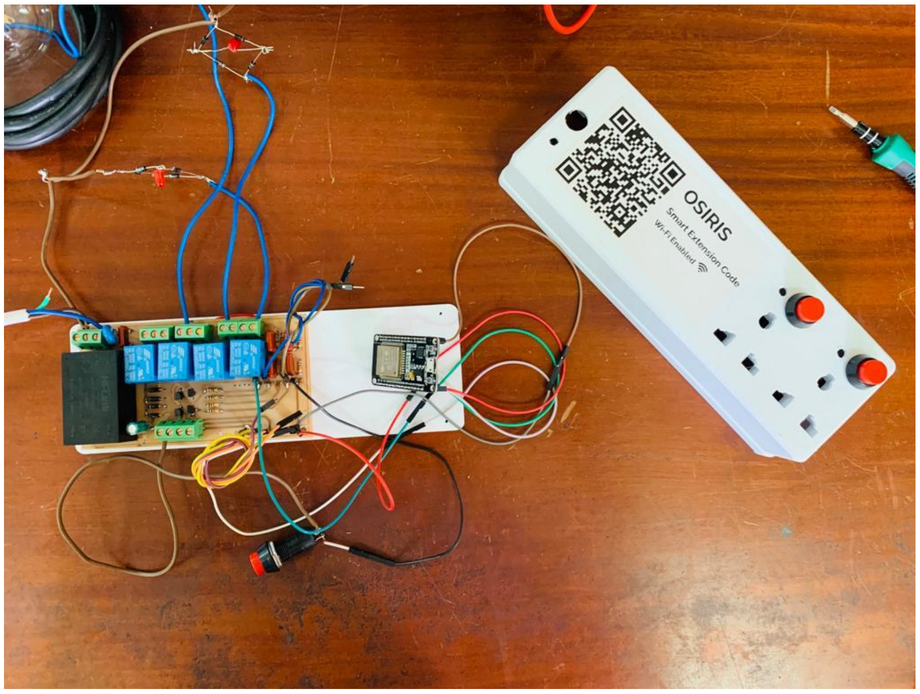
Assembly





Final Enclosure





Material

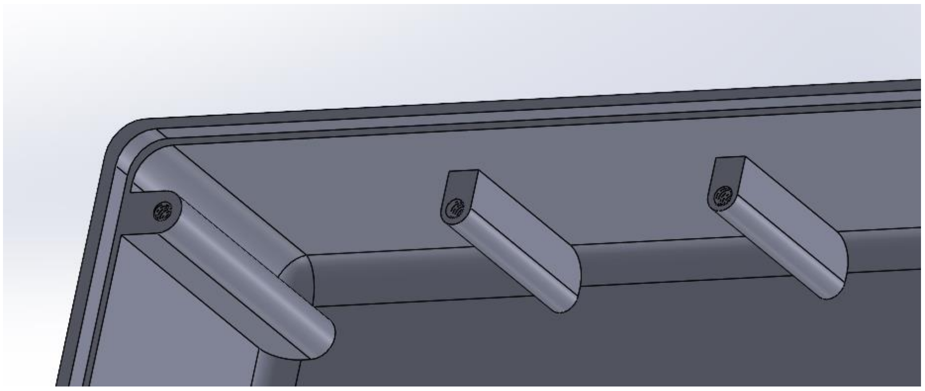
PLA (Polylactic acid) was used as the material for the enclosure. Some advantages of using PLA for enclosures are,

1. Biodegradable: PLA is made from natural materials such as corn starch, sugarcane, or cassava, which makes it biodegradable and more environmentally friendly compared to traditional plastics.
2. Low toxicity: PLA is generally considered to be a safe and non-toxic material, making it suitable for use in enclosures that may come into contact with food or other sensitive materials.
3. Easy to print: PLA is a popular choice among 3D printer users because of its low melting point and easy-to-use properties. It is easy to print with and produces high-quality prints with minimal warping or shrinkage.
4. Strong and stiff: PLA is a strong and stiff material that can produce robust enclosures that can withstand stress and maintain their shape.
5. Variety of colors: PLA is available in a wide range of colors, which allows you to choose the color that best suits your enclosure design.
6. Cost-effective: PLA is relatively inexpensive compared to other 3D printing materials, making it a cost-effective option for enclosure printing.

Design considerations & Problems faced

Important key points, constraints, and problems we faced when designing and printing the enclosure are listed below.

- Maximum dimensions for printing in typical 3d printers are (22×22×5) cm. Therefore, the design has to be built including no more than two sockets to avoid complexity in setting up the enclosure.
- The pillars for screw holes which connect the top and bottom parts were connected to the walls instead of designing them isolated. Which may improve the print quality and strength.



- Clearance of about 0.5 mm were left for connecting the top and bottom parts.

User Manual

Osiris

SMART Extension Cord

User Manual

The Osiris SMART Extension Cables allow you to power up and monitor all of your household devices from anywhere in the world using your smart devices just by keeping the extension cable connected to your home Wi-Fi. Although our extension cable is designed to seem and feel like a typical power extension cable, its IoT enabled capabilities will turn your house into a modern SMART home where all of the control lie in your hand.

Our product series come with a varying number of *two to four socket outlets* for you to choose depending on your requirement.

This document will guide you with all the necessary steps to install an Osiris SMART Extension cable in your house and to use it safely to control the household devices via your smart devices.

Device Installation

Notice – You can use the Osiris power extension cables even without the use of Wi-Fi as a usual power extension cable. Each pressing of a push button will switch the state of the associate socket from on to off and vice versa. To indicate the power being supplied, each outlet accompanies a power indicating LED, and a short ticking sound will occur when the state of the socket gets changed.

Step 1 – Connect the power extension cable to the mains supply

Notice – The Osiris SMART Extension Cables are specifically designed to operate in a wide range of supply conditions. It can be used with the typical 100 - 230V, 50 – 60 Hz main supplies.

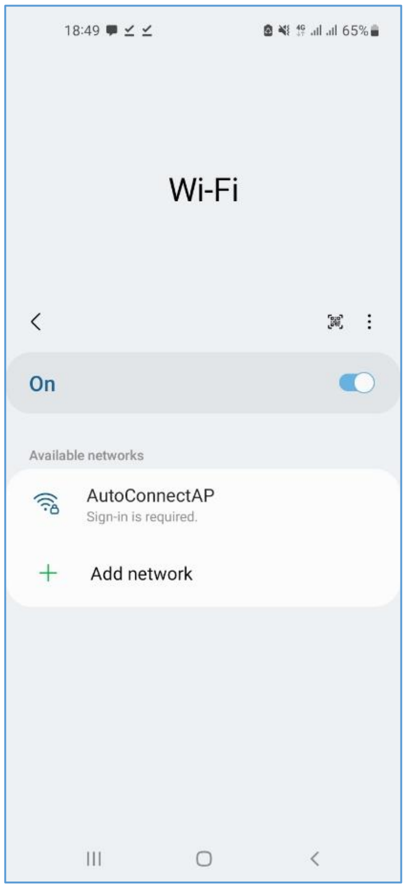


Figure 1

Step 2 – Turn on Wi-Fi in one of your smart devices

Shortly after powering up the extension cable, a Wi-Fi connection with the name **AutoConnectAP** will appear in the Available Wi-Fi Networks list as depicted in figure 1.

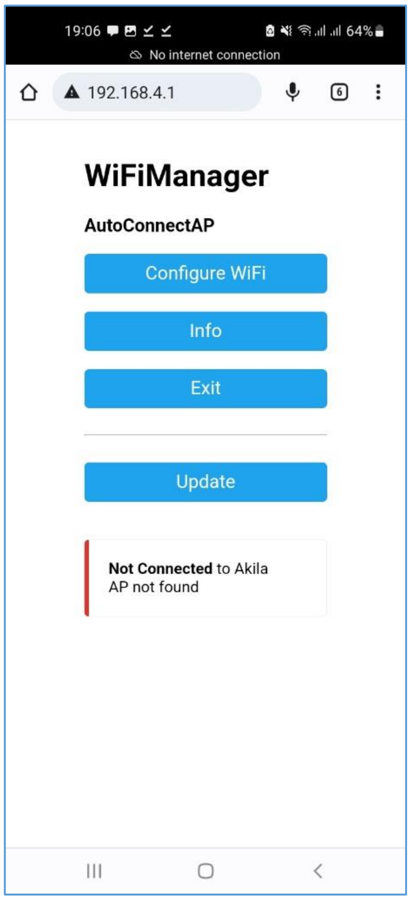
Connect with that Wi-Fi connection; type as "password" in the Wi-Fi password/network security key section.

Step 3 – Go to Wi-Fi Manager

In the same device you used to connect to the extension cable, type in the IP Address **192.168.4.1** in a web-browser. This will direct you to the **Wi-Fi Manger** Window; see figure 2. Click on the **Configure WiFi** option.

Step 4 – Enter the Wi-Fi Credentials

In the prompted page (figure 3), fill in the username and password of your home Wi-Fi (or any other Wi-Fi connection through which you wish to connect with the extension cable). Then, choose **Save** option. If the credentials were uploaded to the device correctly, a page must appear as given in figure 4 saying credentials were **saved**.



Not Connected to Akila
AP not found

Figure 2

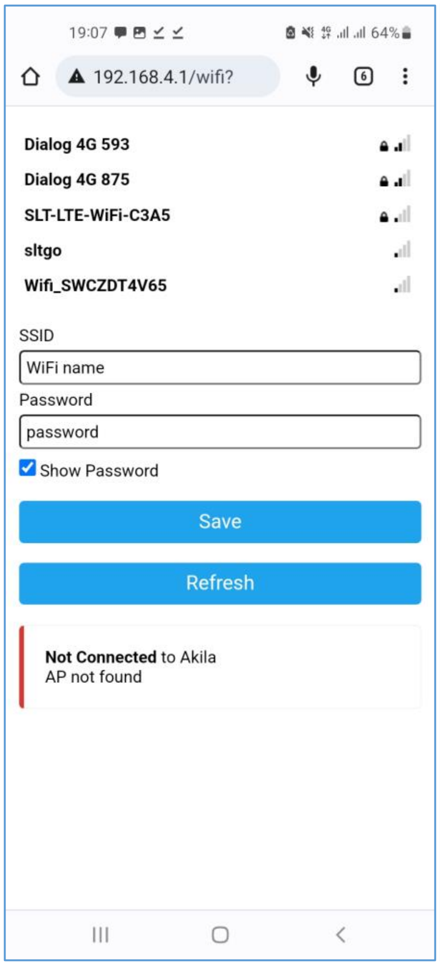
Step 5 – Turn on the Wi-Fi you specified

Once the Wi-Fi details are given, the Osiris power extension cable will automatically connect to the specified Wi-Fi connection whenever it is available. Then, you must see it in the connected devices list of your Wi-Fi or mobile hotspot manager window as **esp32-911250**. See figure 5.

After configuring the Wi-Fi credentials, if the device does not get instantly connected to the specified network, try powering off the extension cable and powering it up again.

If still not connecting, try lowering the Bandwidth of your Wi-Fi network to 2.4GHz.

If you want to connect the device to a new Wi-Fi connection, turn off all the previously configured Wi-Fi networks. Then, the device will start to act as a Wi-Fi access point again, and repeat the steps from step number 1 to 3 to and the new Wi-Fi connection to the cable.


19:07 64%

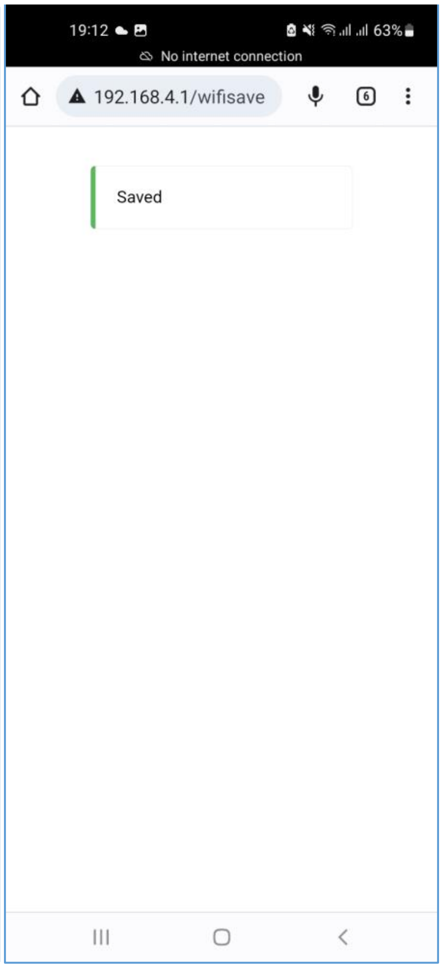
192.168.4.1/wifi?

Dialog 4G 593
Dialog 4G 875
SLT-LTE-WiFi-C3A5
sltgo
Wifi_SWCZDT4V65

SSID: WiFi name
Password: password
[✓] Show Password

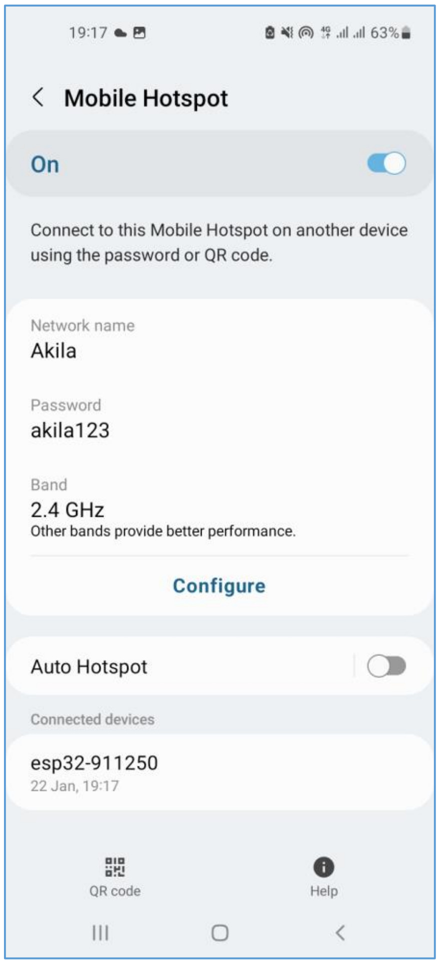
Not Connected to Akila
AP not found

Figure 3


19:12 63%
No internet connection
192.168.4.1/wifisave

Saved

Figure 4


19:17 63%

On

Connect to this Mobile Hotspot on another device using the password or QR code.

Network name
Akila

Password
akila123

Band
2.4 GHz
Other bands provide better performance.

Auto Hotspot

Connected devices
esp32-911250
22 Jan, 19:17

Figure 5

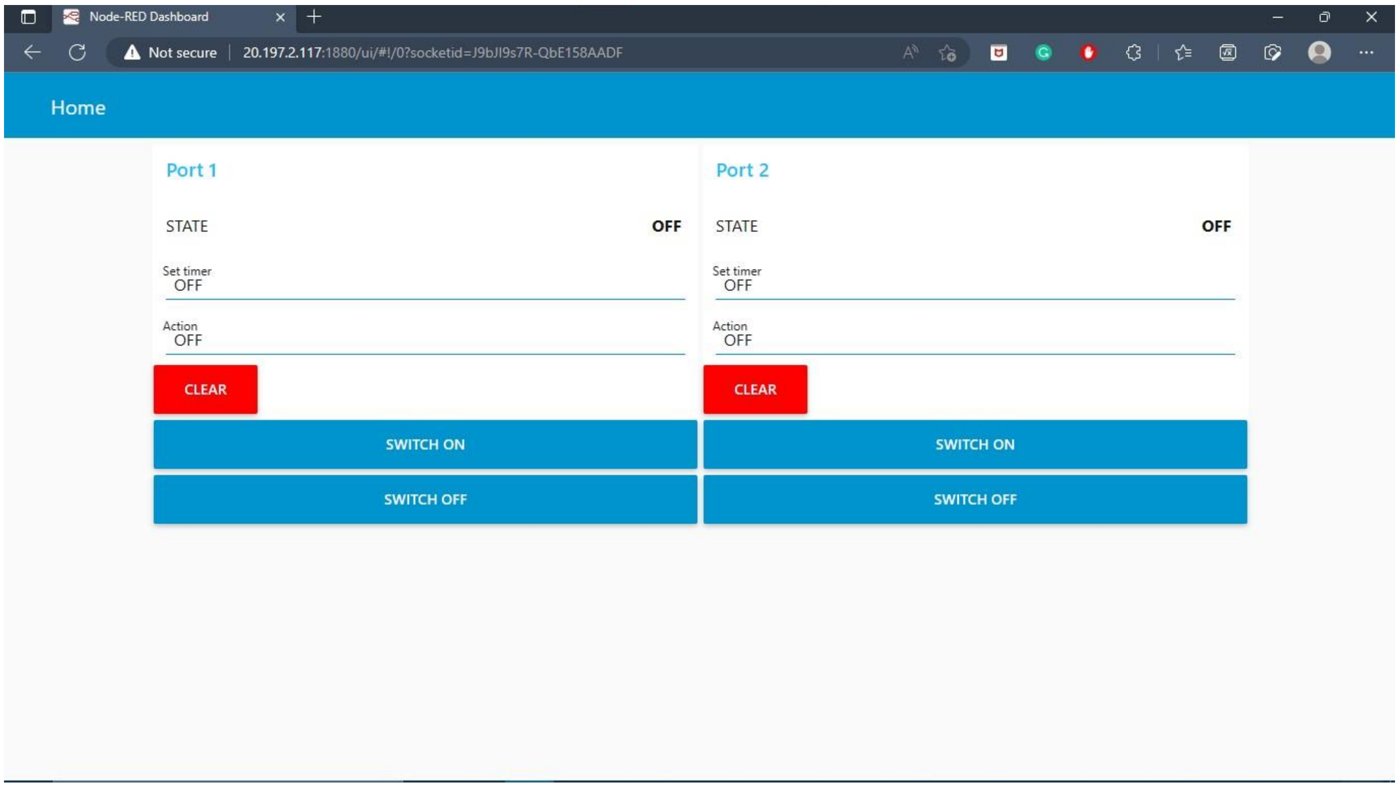
Control via Wi-Fi

Type in the IP address 20.197.2.117:1880 in your web browser from any smart device you have. This will direct you to the following web page given in figure 6. (the appearance of the web page may vary from the device you are accessing the web page.)

The User Interface is mostly self-explanatory; the current states of sockets will be shown aside each port number (port is same as sockets).

To turn on a socket, click on the corresponding **turn on** button. Similarly, clicking on the **turn off** button will switch off the relevant socket.

To set a timer, type in the time value in minutes in the appropriate **Set timer** section. Timer will begin to count from the instance you hit enter or return after specifying the timer value.



Port 1

STATE OFF

Set timer OFF

Action OFF

Port 2

STATE OFF

Set timer OFF

Action OFF

Warnings

Since the cable is powered up with the mains supply, handle it with proper care. Do not spill water over the device or handle the device with wet hands when plugged into the main supply.

Each socket outlet can support a maximum current output of 10A. However, the total current drawn from all the sockets also must not exceed the 10A limit. Therefore, when connecting multiple devices to the extension cable, be mindful about the cumulative current requirement.

Code

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <WiFiManager.h>

WiFiClient espClient;
PubSubClient client(espClient);

long CurrentTime= millis();
long DisconnectedTime = 0.0000;
const int BAUD_RATE = 115200;
bool  PrevConnectionStatus = false;
long WifiReconnectionInterval = 5000000;
bool btn_1_change = true;
bool btn_2_change = true;


const int buttonPin1 = 12;
const int buttonPin2 = 13;
const int relaypin1 = 14;
const int relaypin2 = 25;

long lastMsg = 0;
char msg[50];
int value = 0;

int timer1 = 360;
int Ontime1 = 0;
int buttonPushCounter1 = 0;
int buttonState1 = 0;
int lastButtonState1 = 0;

int timer2 = 360;
int Ontime2=0;
int buttonPushCounter2 = 0;
int buttonState2 = 0;
int lastButtonState2 = 0;

const char
  *ssid = "…………",           //change here to nearby wifi ssid
  *password = "………”,  //wifi password
  *mqtt_server = "y3d5b6c8.us-east-1.emqx.cloud";

void WIFI_init() {
  //This part just used for testing, to connect to wifi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
```

```
}

void reconnect() {
  //Connecting to MQTT broker
  Serial.print("Attempting MQTT connection...");
  if (client.connect("ESP8266Client", "Vinuja", "Vinuja@199534")) {
    Serial.println("connected");
    PrevConnectionStatus = true;
    // Subscribe
    client.subscribe("output1");
    client.subscribe("output2");
    client.subscribe("Timer1");
    client.subscribe("Timer2");
    Serial.println("Subscribed");
  } else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" System Will Try to Reconnect in 5s");
  }
}

void Publish(){
  //When a button pressed manually, it is publishing to the MQTT broker and
also switching the circuit
  if (buttonState1){
    if (btn_1_change){
      client.publish("switch_1", "ON");
      digitalWrite(relaypin1,HIGH);
      btn_1_change = !btn_1_change;
    }
     else{
      client.publish("switch_1", "OFF");
      digitalWrite(relaypin1,LOW);
      btn_1_change = !btn_1_change;

     }
    buttonState1 = 0;
  }
  if (buttonState2){
    if (btn_2_change){
      client.publish("switch_2", "ON");
      digitalWrite(relaypin2,HIGH);
      btn_2_change = !btn_2_change;
    }
     else{
      client.publish("switch_2", "OFF");
      digitalWrite(relaypin2,LOW);
      btn_2_change = !btn_2_change;
     }
    buttonState2 = 0;
  }


}
void Wifi_Access(){
  //Accessing a wifi network, configuring wifi
  WiFi.mode(WIFI_STA); // explicitly set mode, esp defaults to STA+AP
    // it is a good practice to make sure your code sets wifi mode how you
want it.
```

```
    // put your setup code here, to run once:
    Serial.begin(115200);

    //WiFiManager, Local intialization. Once its business is done, there is
no need to keep it around
    WiFiManager wm;

    // reset settings - wipe stored credentials for testing
    // these are stored by the esp library
    // wm.resetSettings();

    // Automatically connect using saved credentials,
    // if connection fails, it starts an access point with the specified name
( "AutoConnectAP"),
    // if empty will auto generate SSID, if password is blank it will be
anonymous AP (wm.autoConnect())
    // then goes into a blocking loop awaiting configuration and will return
success result

    bool res;
    // res = wm.autoConnect(); // auto generated AP name from chipid
    // res = wm.autoConnect("AutoConnectAP"); // anonymous ap
    res = wm.autoConnect("AutoConnectAP","password"); // password protected
ap

    if(!res) {
        Serial.println("Failed to connect");
        //ESP.restart();
    }
    else {
        //if you get here you have connected to the WiFi
        Serial.println("connected...yeey :)");
    }
}

void callback(String topic, byte* message, unsigned int length) {
  // Subscribing data from the MQTT broker is subscribed with respect to the
topic
  String messageTemp;
  for (int i = 0; i < length; i++) {
    //Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();

  // Feel free to add more if statements to control more GPIOs with MQTT


  if(topic=="output1"){

      if(messageTemp == "ON"){
        digitalWrite(relaypin1, HIGH);
        Serial.println("ON_1");

      }
      else if(messageTemp == "OFF"){
        digitalWrite(relaypin1, LOW);
        Serial.println("OFF_1");
      }
  }
  if(topic=="output2"){

      if(messageTemp == "ON"){
        digitalWrite(relaypin2, HIGH);
        Serial.println("ON_2");

      }
      else if(messageTemp == "OFF"){
        digitalWrite(relaypin2, LOW);
        Serial.println("OFF_2");
      }
  }
  if(topic == "Timer1"){
    Serial.println("I am Here");
    timer1=messageTemp.toInt();
    Ontime1 = millis();
    Serial.println(timer1);
  }
  if(topic == "Timer2"){
    Serial.println("I am Here Bro");
    timer2=messageTemp.toInt();
    Ontime2 = millis();
    Serial.println(timer2);
  }
  Serial.println();
}
void UPDATE_CONNECTED_DEVICE() {
  if (!client.connected() && abs((int)CurrentTime - DisconnectedTime) >
WifiReconnectionInterval)
    reconnect();
  if (PrevConnectionStatus && !client.connected()) {
    DisconnectedTime = CurrentTime;
    PrevConnectionStatus = false;
  }
}

void CheckTimer(){
  //Timer controller
  if((millis()-Ontime1)>timer1*60*1000){
    client.publish("switch_1","OFF");
    digitalWrite(relaypin1,LOW);
    Serial.println("1_off");
    timer1=360;
  }
  if((millis()-Ontime2)>timer2*60*1000){
    client.publish("switch_2","OFF");
    digitalWrite(relaypin2,LOW);
    Serial.println("2_off");
    timer2=360;
  }

}

void IRAM_ATTR btn_1(){
  buttonState1 = 1;

}
```

```
void IRAM_ATTR btn_2(){
  buttonState2 = 1;

}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(BAUD_RATE);
  pinMode(buttonPin1, INPUT_PULLUP);

  pinMode(buttonPin2, INPUT_PULLUP);
  pinMode(relaypin1, OUTPUT);
  pinMode(relaypin2, OUTPUT);
  attachInterrupt(buttonPin1, btn_1, RISING);
  attachInterrupt(buttonPin2, btn_2, RISING);

  Serial.println("Serial Monitor Intialized.... !");
  delay(1000);
  //WIFI_init();
  Wifi_Access();
  client.setServer(mqtt_server, 15742);
  client.setCallback(callback);
  reconnect();
  client.publish("switch_1", "OFF");
  //Serial.println("HHHH");
  client.publish("switch_2", "OFF");
  //Serial.println("RCB");

}

void loop() {
  // put your main code here, to run repeatedly:
  client.setCallback(callback);
  client.loop();
  client.setServer(mqtt_server, 15742);
  UPDATE_CONNECTED_DEVICE();
  Publish();
  CheckTimer();




}
```