DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

UNIVERSITY OF MORATUWA

EN2160 - Electronic Design Realization

Final Project Report

200728R

H.D.K.G. Wijesiri

Table of Contents

1. **Abstract** 3
2. **Product description** 4
3. **Features and Specifications** 5
4. **Block Diagram** 6
5. **Schematics** 7
6. **PCB** 8
7. **Enclosure Design** 11
8. **BOM** 14
9. **Assembly Instructions** 16
10. **Testing** 19

1. Abstract

The Pomodoro Timer is a time management tool designed to enhance productivity and focus during work sessions. Inspired by the Pomodoro Technique, this application subdivides work into intervals, typically 25 minutes long, separated by short breaks. The primary goal is to maintain concentration and avoid burnout by breaking tasks into manageable segments. This report presents the development of a Pomodoro Timer application, its features, and functionality. The timer allows users to set work and break durations according to their preferences, providing flexibility in managing different tasks. The application employs an intuitive user interface, enabling easy navigation and interaction. This report presents the development of a Pomodoro Timer application, its features, and functionality. The timer allows users to set work and break durations according to their preferences, providing flexibility in managing different tasks. The application employs an intuitive user interface, enabling easy navigation and interaction.

2. Product description

Pomodoro is a technique that is used to efficiency studying method. The Pomodoro Timer is a productivity tool based on the Pomodoro Technique, a time management method that breaks work into focused intervals with short breaks in between. It mainly focused to student in schools and undergraduates. It mainly studying time break in to two parts that are learning time and interval. Features:

1.It can count to 25 minutes and give 5 minute break.
2.It has a display to indicate start, end and time.
3.It has a alarm to inform end of the 25 minutes.
4.It has an LED system to indicate how much time do you have to study.
5.It gives the total time count of the day to user

- Introducing our user-friendly Pomodoro Timer, a smart and intuitive device designed to help you maximize productivity and effectively manage your time. If you often find yourself getting distracted or struggling to stay focused, our innovative timer is here to provide a simple yet powerful solution.

- The Pomodoro Timer follows the popular Pomodoro Technique, a time management method that involves breaking your work into focused intervals, typically 25 minutes long, followed by short 5-minute breaks. This technique has been proven to enhance concentration, increase efficiency, and reduce burnout, making it an ideal choice for students, professionals, and anyone seeking to optimize their work routines.

- Using our Pomodoro Timer is a breeze. The clear and easy-to-read OLED display shows the remaining time for each study and break session, helping you stay on track and accomplish your tasks effectively. When it's time to focus, vibrant LED lights will indicate the study period, and during breaks, a different color will remind you to take a well-deserved breather. As a gentle and unobtrusive sound chimes, you'll smoothly transition between work and rest, eliminating the need to constantly check the clock and ensuring a seamless workflow.

- Our commitment to the environment led us to include a rechargeable battery, reducing the waste associated with disposable batteries. The Type-C charging port allows for quick and convenient charging, ensuring that your Pomodoro Timer is always ready for use whenever you need it. Its compact and lightweight design makes it portable and easy to carry, enabling you to maintain your productivity and time management routines wherever you go.

3. Features and Specifications

1. Charging:

The Pomodoro Timer device is equipped with a single rechargeable battery as its power source, which simplifies maintenance and reduces environmental impact through battery reusability. To charge the device, users can utilize the Type-C charging port on the PCB (Printed Circuit Board), which supports rapid charging and is compatible with the widely adopted Type-C standard, enhancing the product's market compatibility and making it easy for users to charge using common charging cables.

2. Display:

The device features an OLED display that provides essential information for the Pomodoro technique, a time management method. It shows both the "study time" and "interval time" phases. During the "study time" phase, users can focus on their work, while during the "interval time" phase, they can take a short break before resuming their tasks. Additionally, the OLED display also presents real-time temperature and humidity readings, which can help users create a comfortable working environment.

3. LED Pattern:

To visually indicate the different phases of the Pomodoro technique, there are LEDs on the PCB. These LEDs serve two purposes:

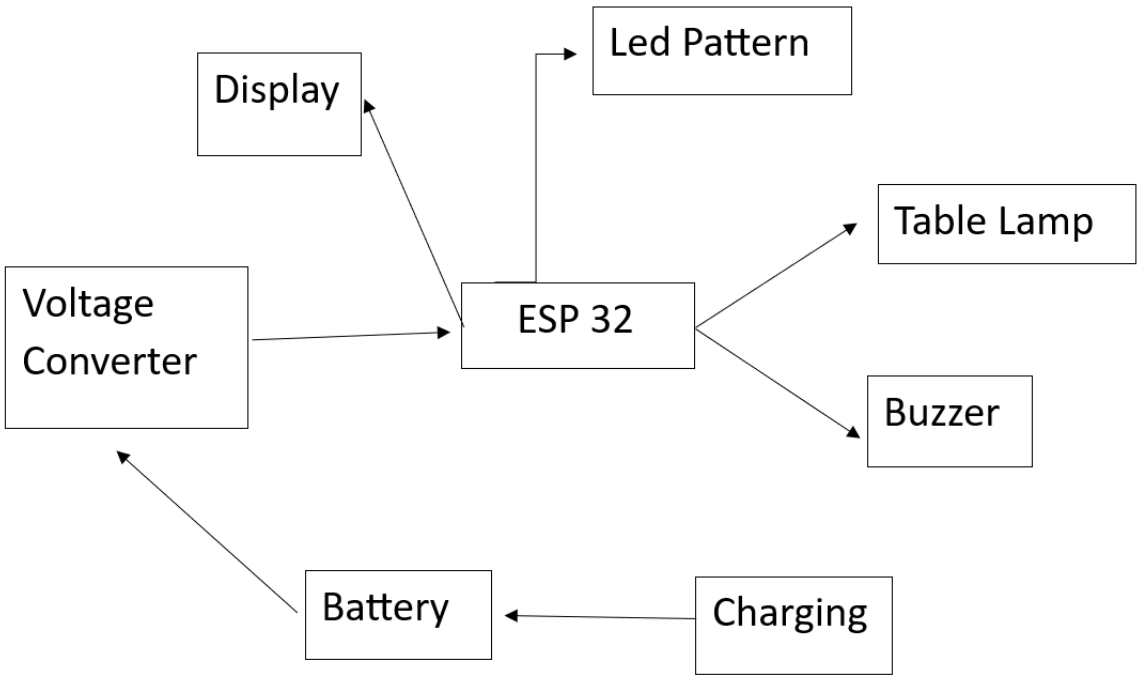
- Study and Interval Time: The LEDs will display a specific pattern or color to indicate when it's time to focus on studying (the "study time" phase) and when it's time to take a break (the "interval time" phase).
- Charging Indication: The LEDs will also provide a visual cue to show that the device is currently charging. This feature helps users know the charging status at a glance.

4. Buzzer:

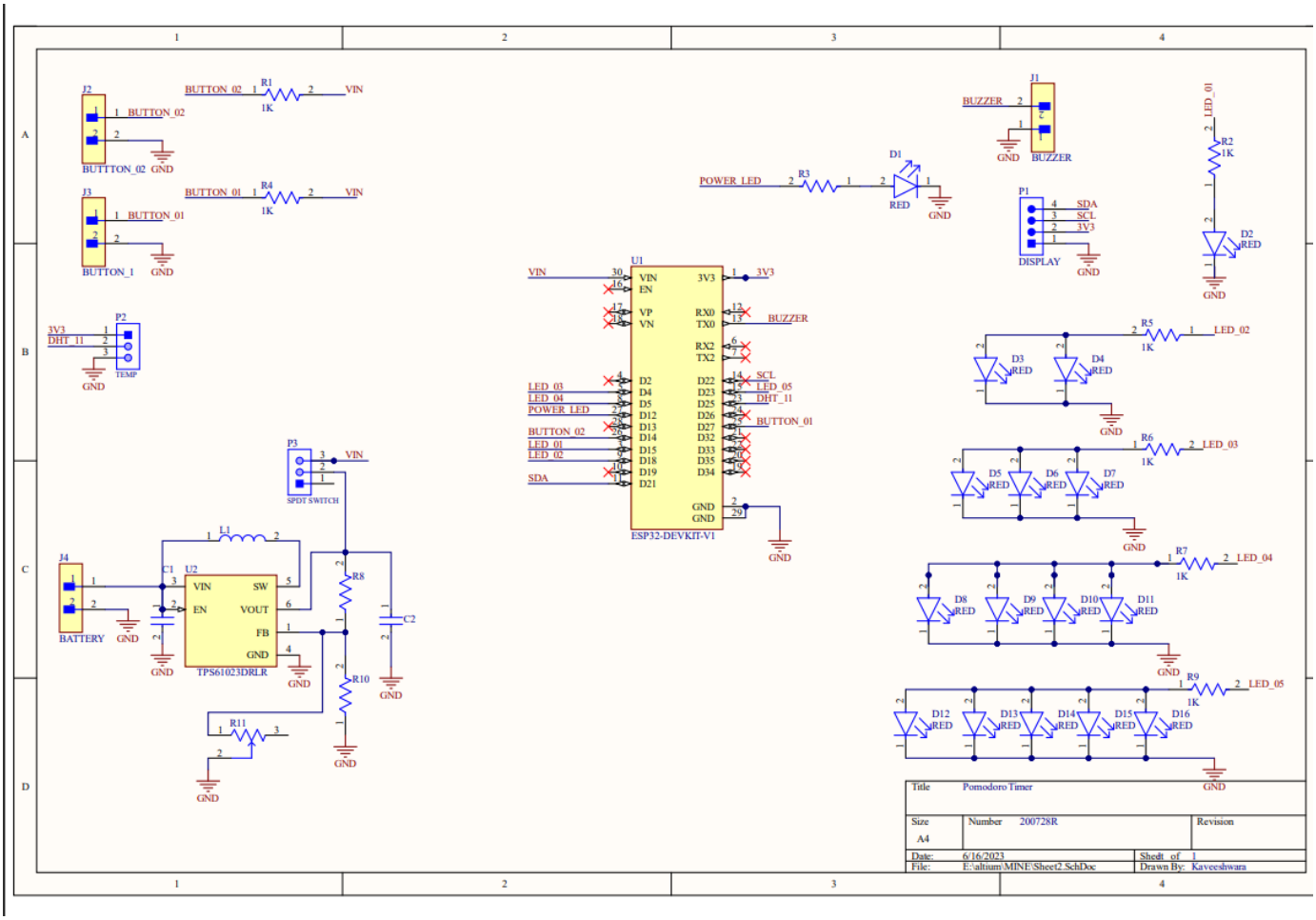
The Pomodoro Timer includes a buzzer that provides audio signals to indicate the end of a study or interval session. When the "study time" is over, the buzzer will alert the user to take a break, and when the "interval time" is over, it will signal the user to resume their work. This audio notification ensures that users stay on track with their work and breaks without needing to check the display constantly.

4. Block Diagram

This is the block diagram of the pomodoro timer.

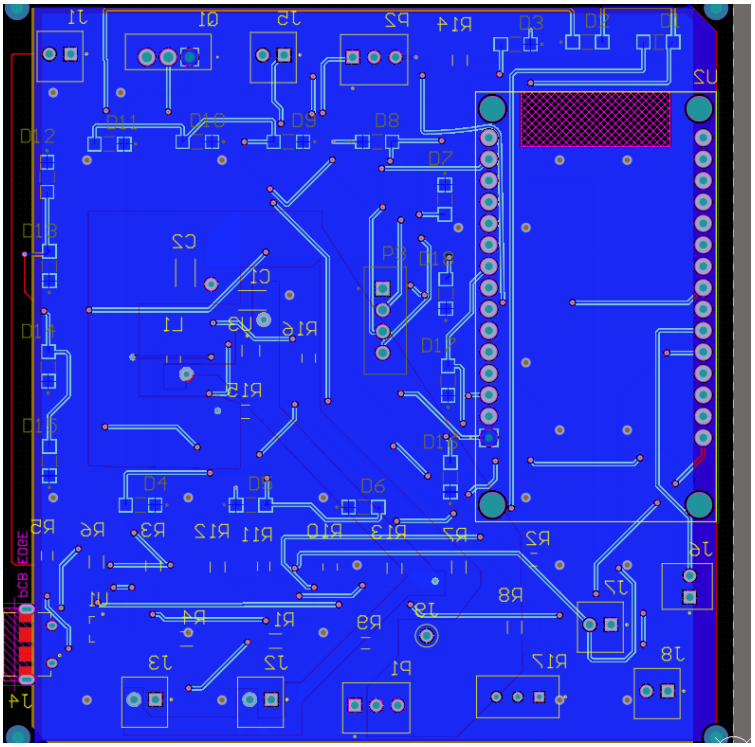


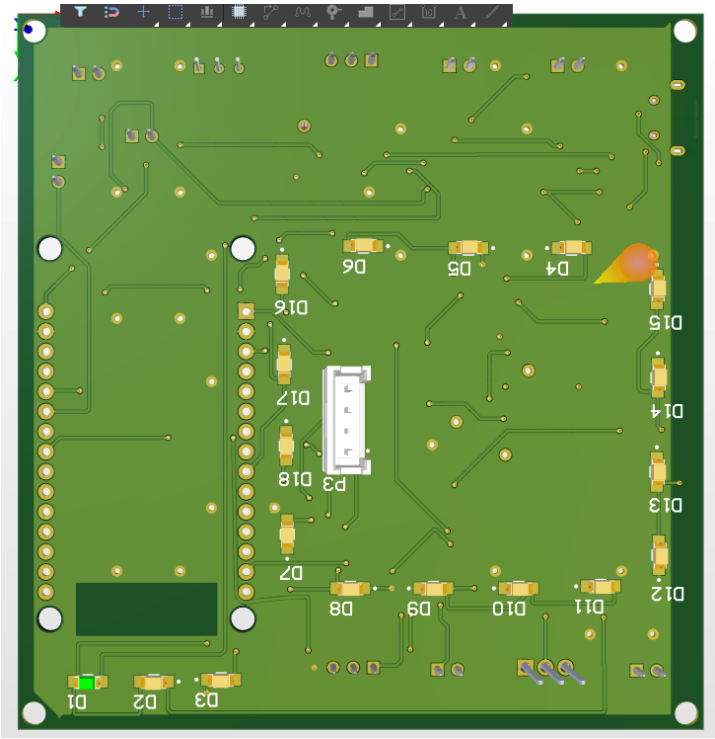
5. Schematics



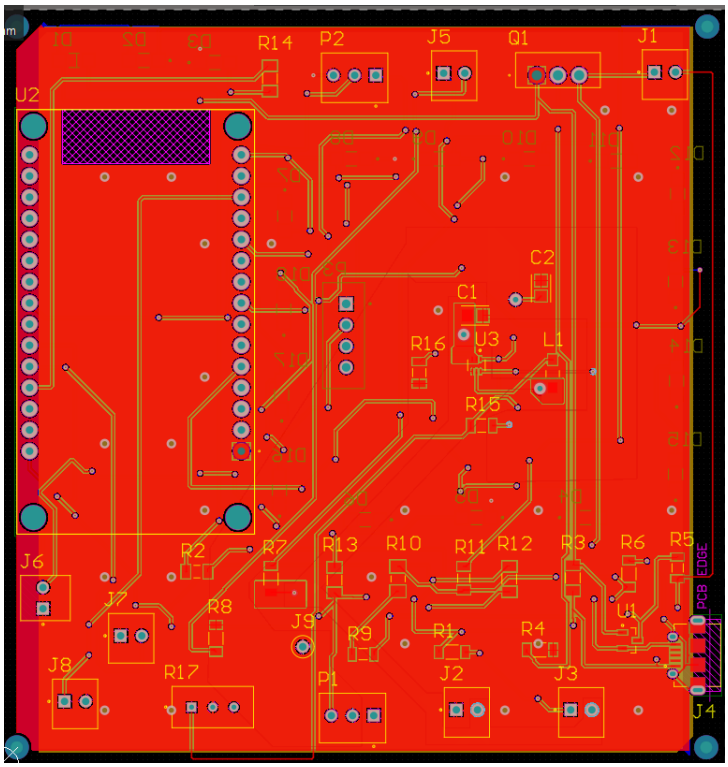
6. PCB

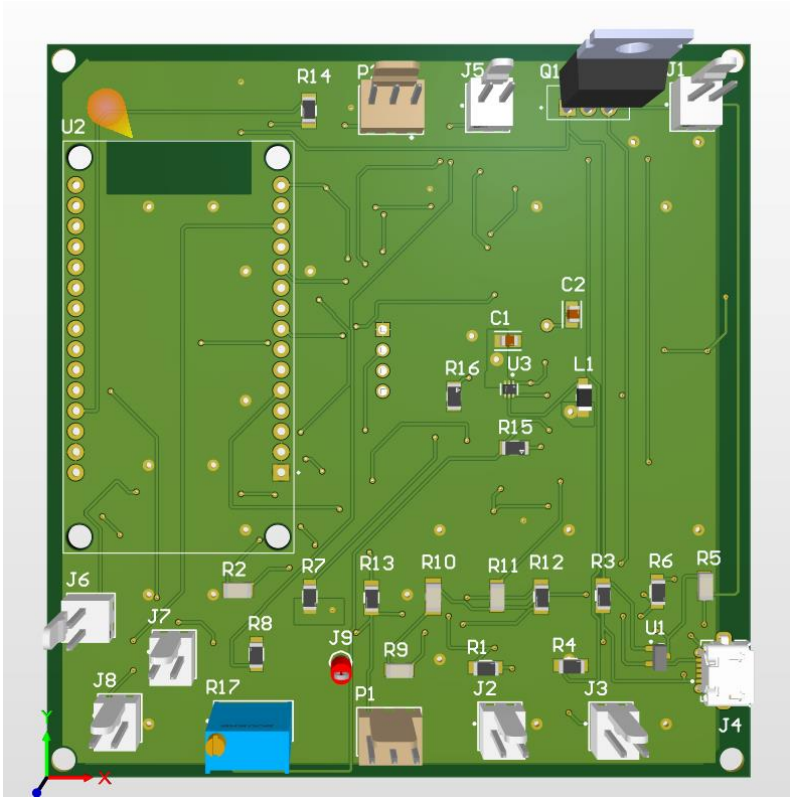
Bottom View



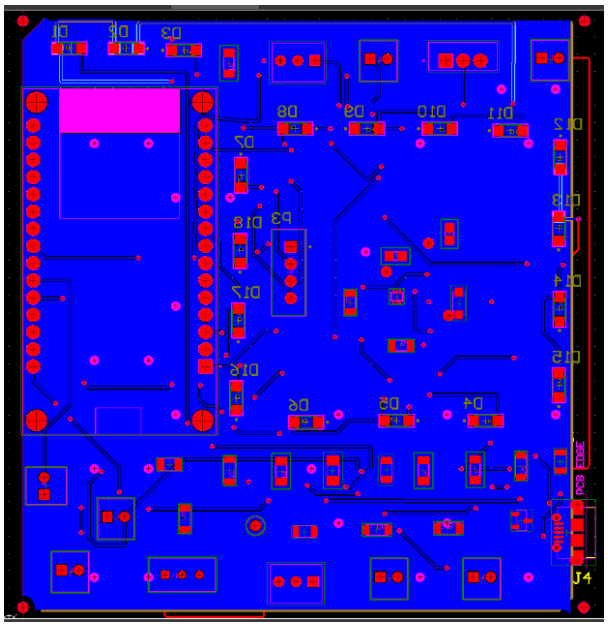


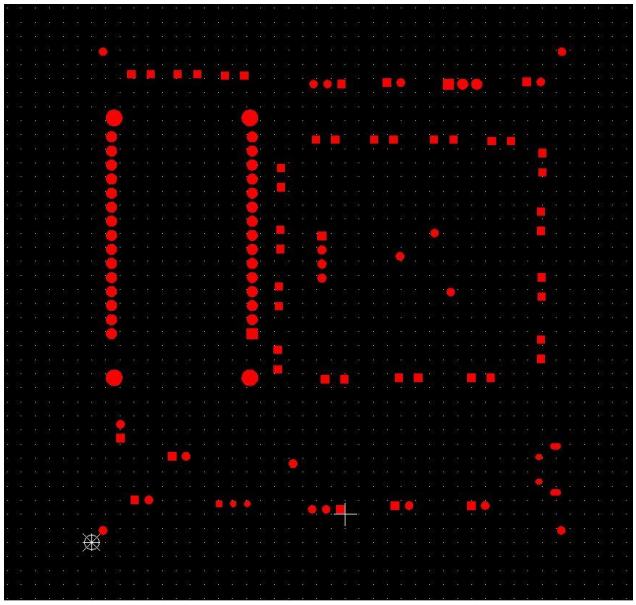
Top view

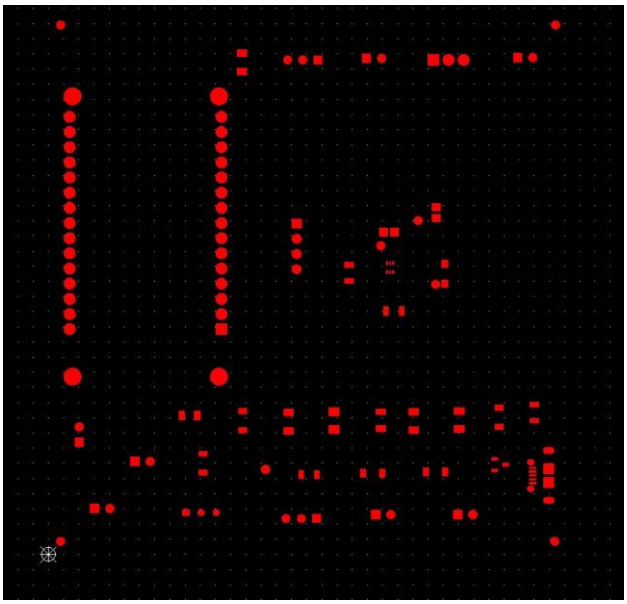

R14
P2
J5
Q1
J1
U2
C2
C1
R16
U3
L1
R15
R2
R7
R13
R10
R11
R12
R3
R6
R5
J6
J7
R8
J9
R9
R1
R4
U1
J8
R17
P1
J2
J3
J4
PCB EDGE

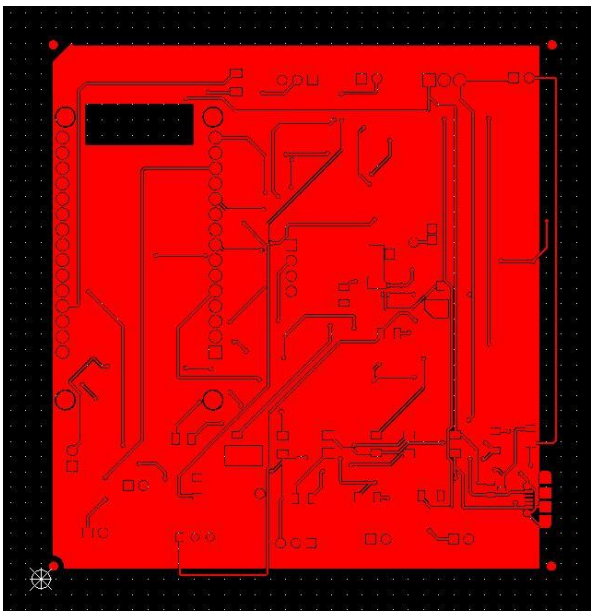

R14
J5
Q1
J1
U2
C2
C1
R16
U3
L1
R15
R2
R7
R13
R10
R11
R12
R3
R6
R5
J6
J7
R8
J9
R9
R1
R4
U1
J8
R17
P1
J2
J3
J4

Gerber Files



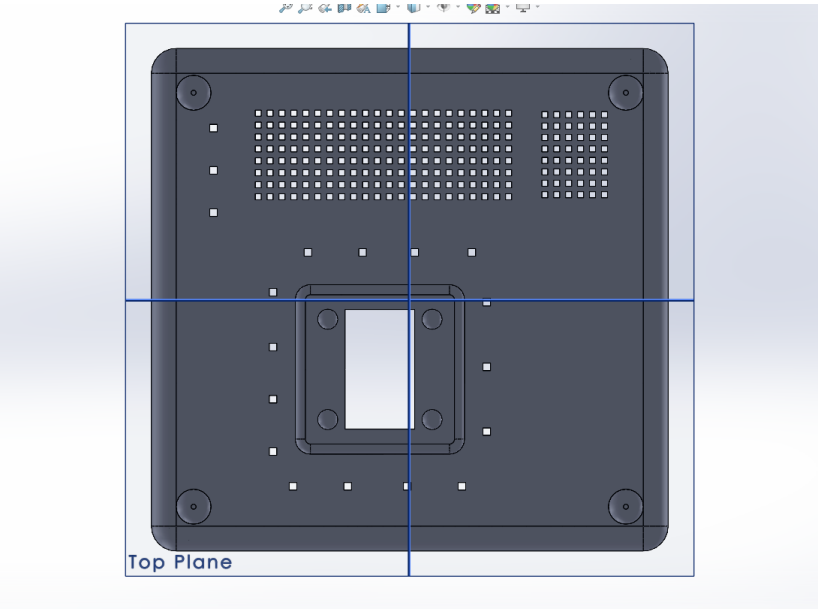


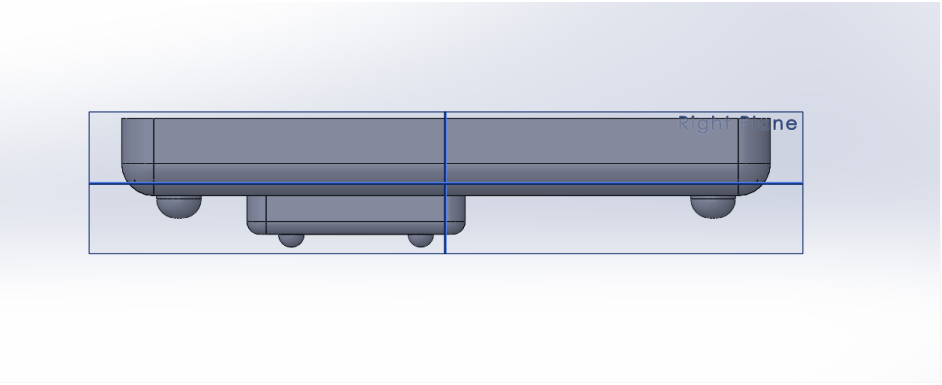


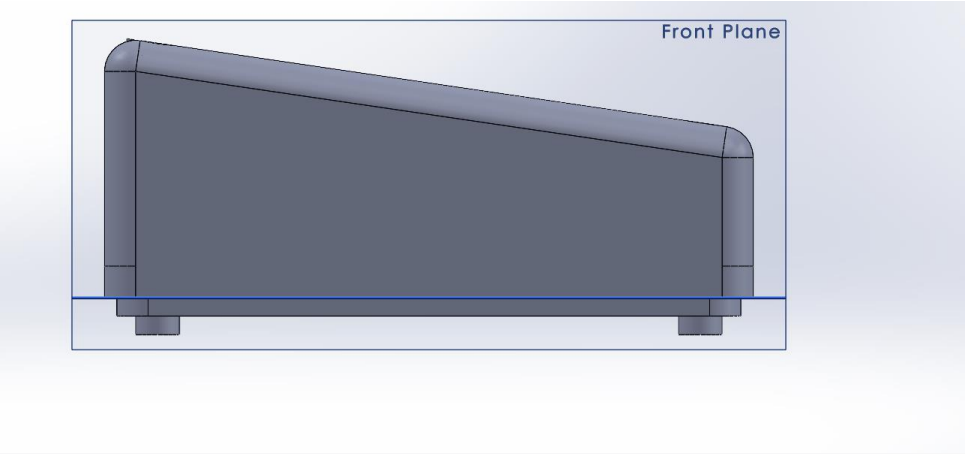


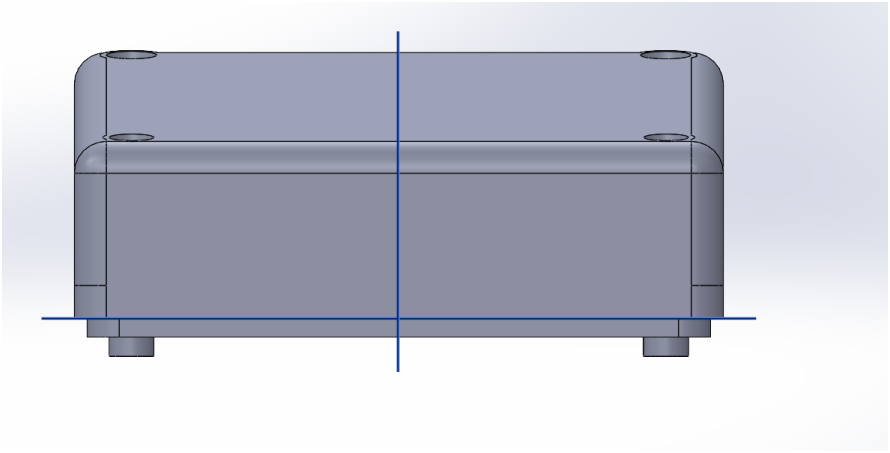
7. Enclosure Design

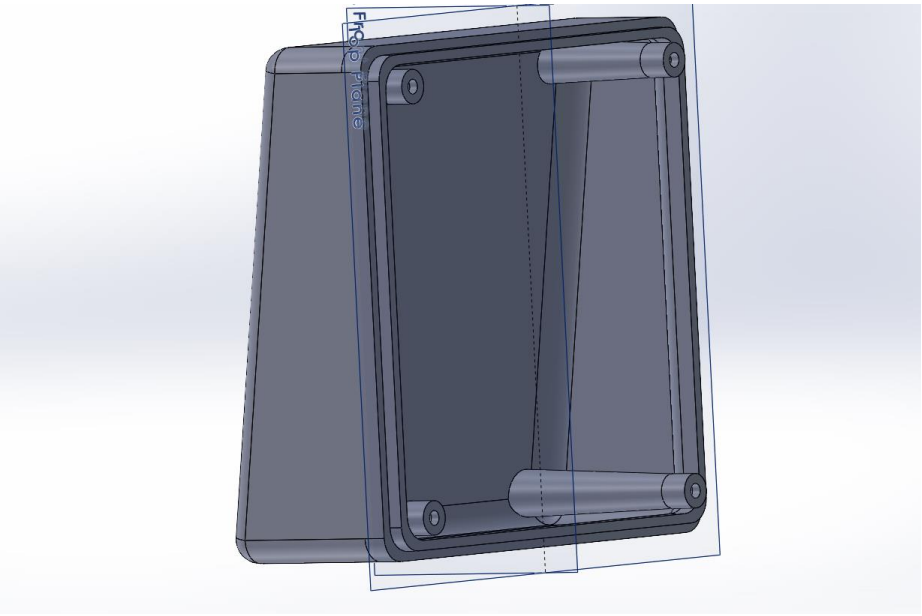
The box and cover components of the Pomodoro Timer enclosure are thoughtfully designed using SolidWorks, providing a secure housing for internal components. The box houses the PCB, rechargeable battery, and Type-C charging port, while the cover securely mounts the OLED display. Both components are connected using discreet nails for a polished appearance.






Front Plane




Front Plane

8. BOM

Part Number	Description	Designator	Supplier	Supplier Unit Price	Quantity
CMP-06035-013048-1	Chip Multilayer Ceramic Capacitors for General Purpose, 0805, 10uF, X5R, 15%, 20%, 25V	C1, C2	Mouser	0.57	2
CMP-2000-05061-2	LED GREEN DIFFUSED 1206 SMD	D1	Mouser	0.2	1
CMP-21615-000001-1	LED RED DIFFUSED CHIP SMD	D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16, D17, D18	Mouser	0.2	17
CMP-1502-00584-2	CONN HEADER VERT 2POS 2.54MM	J1	Mouser		1
CMP-1502-00584-2	CONN HEADER VERT 2POS 2.54MM	J2	Mouser	0.1	1
CMP-1502-00584-2	CONN HEADER VERT 2POS 2.54MM	J3	Mouser	0.1	1
CMP-2000-05610-1	Right Angle Female Type AB Micro USB Connector, 30 V, 1.8 A, -30 to 85 degC, RoHS, Tape and Reel	J4	Mouser	0.1	1
CMP-1502-00584-2	CONN HEADER VERT 2POS 2.54MM	J5	Mouser	0.1	1
CMP-1502-00584-2	CONN HEADER VERT 2POS 2.54MM	J6	Mouser	0.1	1
CMP-1502-00584-2	CONN HEADER VERT 2POS 2.54MM	J7	Mouser	0.1	1
CMP-1502-00584-2	CONN HEADER VERT 2POS 2.54MM	J8	Mouser	0.1	1
CMP-19636-000027-1	PC TEST POINT MINIATURE RED	J9	Mouser	0.1	1
CMP-0220-00021-1	SMD EMI Suppression Ferrite Bead WE-CBF, Z = 50 Ohm	L1	Mouser	0.6	1
CMP-1502-00585-1	Male Locking Header WR-WTB, THT, Vertical, pitch 2.54 mm, 1 x 3 position	P1	Mouser	0.1	1
CMP-1502-00585-1	Male Locking Header WR-WTB, THT, Vertical, pitch 2.54 mm, 1 x 3 position	P2	Mouser	0.1	1

CMP-2000-05548-1	Male Header, Pitch 2.54 mm, 1 x 4 Position, Height 12.8 mm, Tail Length 3.5 mm, -55 to 105 degC, RoHS, Bulk	P3	Mouser	0.1	1
IRFZ44NPBF	N-Channel 55 V 49A (Tc) 94W (Tc) Through Hole TO-220AB	Q1	Mouser	0.4	1
CMP-2100-03681-2	RES SMD 0 OHM JUMPER 1/4W 1206	R1, R4, R6, R7, R8	Mouser	0.2	5
CMP-2003-04481-1		R2	Mouser	0.2	1
CMP-07231-002641-1	Thick Film Chip Resistors 1206 1kΩ 0.25W 1% 100ppm/°C	R3, R12, R13, R14	Mouser	0.2	4
CMP-2003-00848-1		R5, R9	Mouser	0.2	2
CMP-2003-04438-1		R10	Mouser	0.2	1
CMP-2003-00118-1		R11	Mouser	0.2	1
CMP-2003-04807-2	RES Thick Film, 10kΩ, 1%, 0.25W, 100ppm/°C, 1206	R15, R16	Mouser	0.2	2
CMP-2000-05583-2	TRIMMER 500K OHM 0.5W PC PIN TOP	R17	Mouser	0.2	1
CMP-0242-00062-1	Programmable Voltage Reference, 2.5 to 36V, 3-Pin SOT23-3, Tape and Reel	U1	Mouser	0.91	1
ESP32-DEVKIT-V1	Dual core, Wi-Fi: 2.4 GHz up to 150 Mbits/s,BLE (Bluetooth Low Energy) and legacy Bluetooth, 32 bits, Up to 240 MHz	U2	Mouser	0.4	1
CMP-04918-000638-1	FINISH GOODS FOR TPS61023	U3	Mouser	0.8	1

9. Assembly Instructions

- Step 1: Gathering the Components

Before you start assembling your Pomodoro Timer, make sure you have all the necessary components ready. These include the PCB (Printed Circuit Board), OLED display, rechargeable battery, Type-C charging port, LEDs, buzzer, and any other electronic components required for the device.(I mentioned components in the BoM document)

- Step 2: Preparation

Find a clean and static-free workspace to assemble your Pomodoro Timer. Place the PCB on the work surface and organize all the components in a manner that allows easy access during the assembly process.

- Step 3: Soldering

Begin by soldering the Type-C charging port onto its designated location on the PCB. Take care to align the port correctly and ensure a stable connection without any loose joints.

- Step 4: Attaching the OLED Display

Gently connect the OLED display to its designated connector on the PCB. Make sure the connection is secure, and the display is oriented correctly according to the markings on the PCB. (for the attaching Oled display use 2mm screw nails)

- Step 5: Adding LEDs

Now, it's time to solder the LEDs onto the PCB. Follow the provided markings on the PCB for study time, interval time, and charging indication LEDs. Double-check the polarity of each LED to avoid any potential issues with the lighting later on.

- Step 6: Installing the Buzzer

Solder the buzzer onto its designated location on the PCB. Pay attention to the correct polarity to ensure it functions as intended during operation.

- Step 7: Battery Placement

Carefully position the rechargeable battery on the PCB, considering available space and proper alignment with the battery connector on the board.

- Step 8: Completing the Circuit

Take a moment to review all the soldered connections and components on the PCB. Ensure there are no loose connections or solder bridges that might interfere with the proper functioning of the timer.

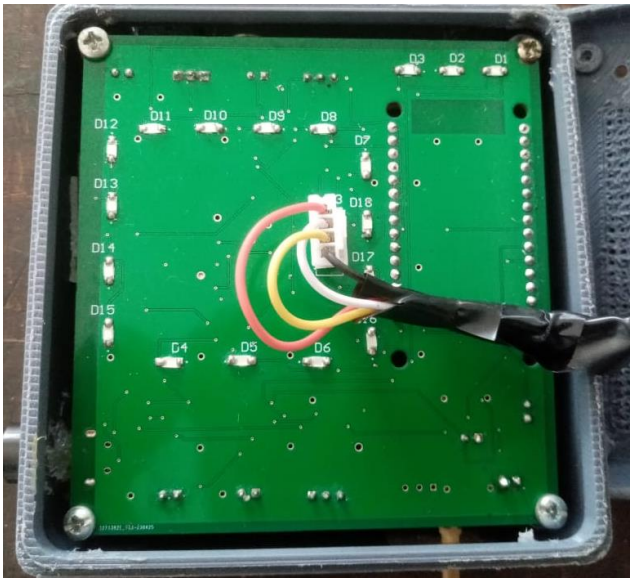
- Step 9: Testing the Assembly

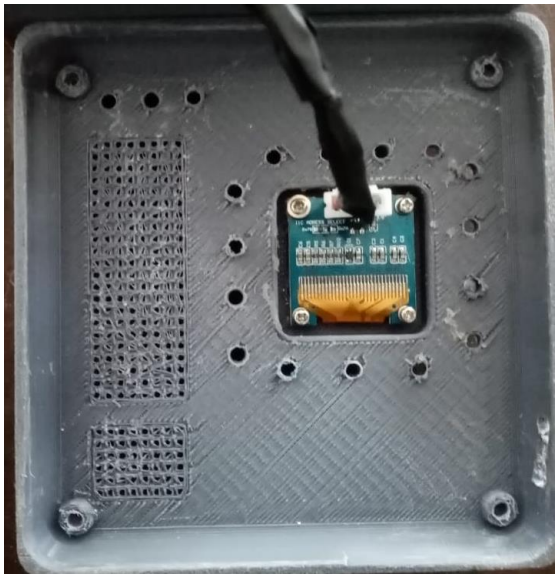
Before enclosing the PCB in the casing, power on the device and conduct a thorough test. Check if the OLED display shows the correct information, if the LEDs light up according to their designated functions, and if the buzzer produces the expected sounds.

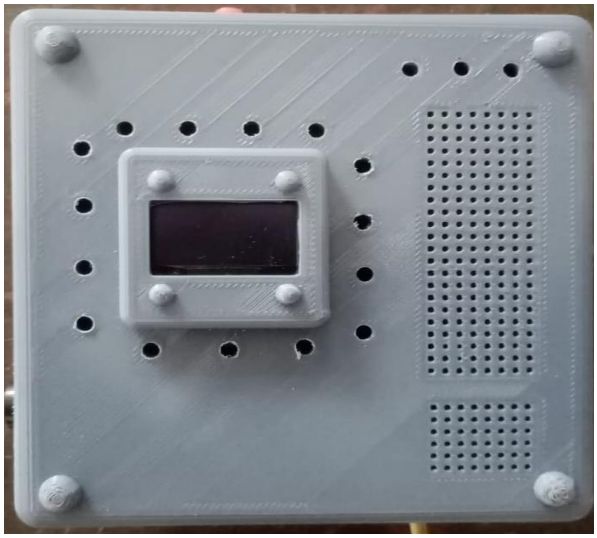
- Step 10: Final Enclosure

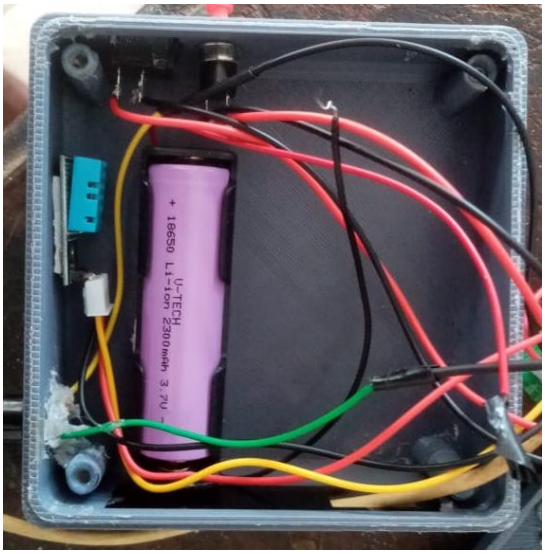
If the test is successful, carefully place the assembled PCB into the designated casing. Secure the casing tightly to protect the internal components from external damage and ensure a neat and professional appearance.

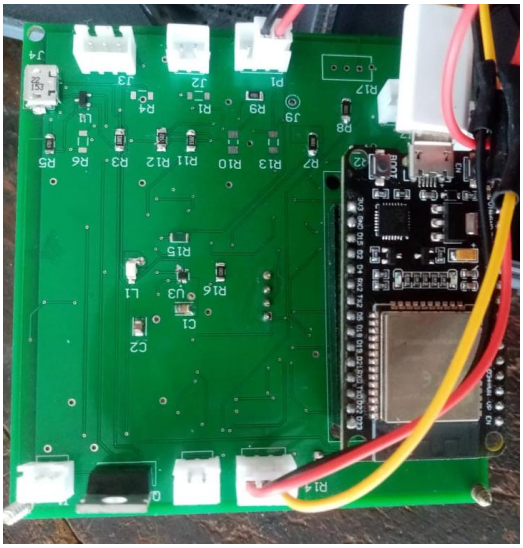
For other all nails use 3mm nails.











10. Testing

To make sure your Pomodoro Timer works properly, follow these easy steps to test its functions. First, turn it on and check if the display shows the right time or asks you to set study and break times. Test the buttons to see if they respond correctly. Look at the LED lights to ensure they light up correctly for study, break, and charging. Check if the buzzer makes a sound during intervals. Set different times and see if the timer counts down properly. Connect it to a power source and check if the charging indicator works. Confirm that the timer starts intervals and reminds you to take breaks

Power On: Ensure the timer's rechargeable battery is charged or connected to a power source. Turn on the device.

Display Test: Verify that the OLED display is working correctly. It should show the current time or a prompt to set study and interval times.

Button Functions: Test the functionality of any buttons or switches on the timer. Ensure they respond appropriately to inputs.

LED Indicators: Check if the LED indicators for study time, interval time, and charging are lighting up correctly according to the timer's state.

Buzzer Test: Set a short interval and ensure the buzzer produces an audible sound. Repeat for the study time interval.

Time Accuracy: Set different time intervals and observe if the timer accurately counts down and transitions between study and interval phases.

Charging Test: Connect the timer to a power source using the Type-C charging port. Confirm that the charging LED indicator activates and the battery charges properly.

Interval and Break Management: Observe if the timer properly initiates interval times and notifies you when it's time to take a break.

Study Sessions: Confirm that the study time sessions begin on time and alert you when the session is over.

Pause and Reset Functions: Test if the timer can be paused during intervals and reset to start a new session.

Repeat Testing: Perform multiple test runs to ensure consistent functionality.

For the testing use following codes:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h> // Include the DHT library

#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); // Initialize the
OLED display library

const int TIMER_PIN = 2; // the pin number for the timer output
const int BUZZER_PIN = 17; // Pin connected to the buzzer
const int TIMER_INTERVAL_MS = 1000; // timer interval in milliseconds
const int TIME_SLOT_MINUTES = 3; // time slot in minutes
const int TIME_SLOT_INTERVAL = 2; // interval time slot in minutes
int stage = 0; // to swap between two stages
volatile unsigned long previousMillis = 0; // previous time in milliseconds
volatile int secondsCounter = 0; // counter for elapsed seconds
volatile bool timerExpired = false; // flag for timer expiry

// DHT22 Sensor
#define DHT_PIN 12 // Pin connected to DHT22 sensor
#define DHT_TYPE DHT22 // DHT sensor type
DHT dht(DHT_PIN, DHT_TYPE); // Initialize the DHT sensor library

void setup() {
  Serial.begin(115200);
  pinMode(TIMER_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(15, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(23, OUTPUT);

  dht.begin(); // Initialize the DHT sensor

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // Initialize the OLED display
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(0, 0);
  display.print("Time: 0:00");
  display.display();
```

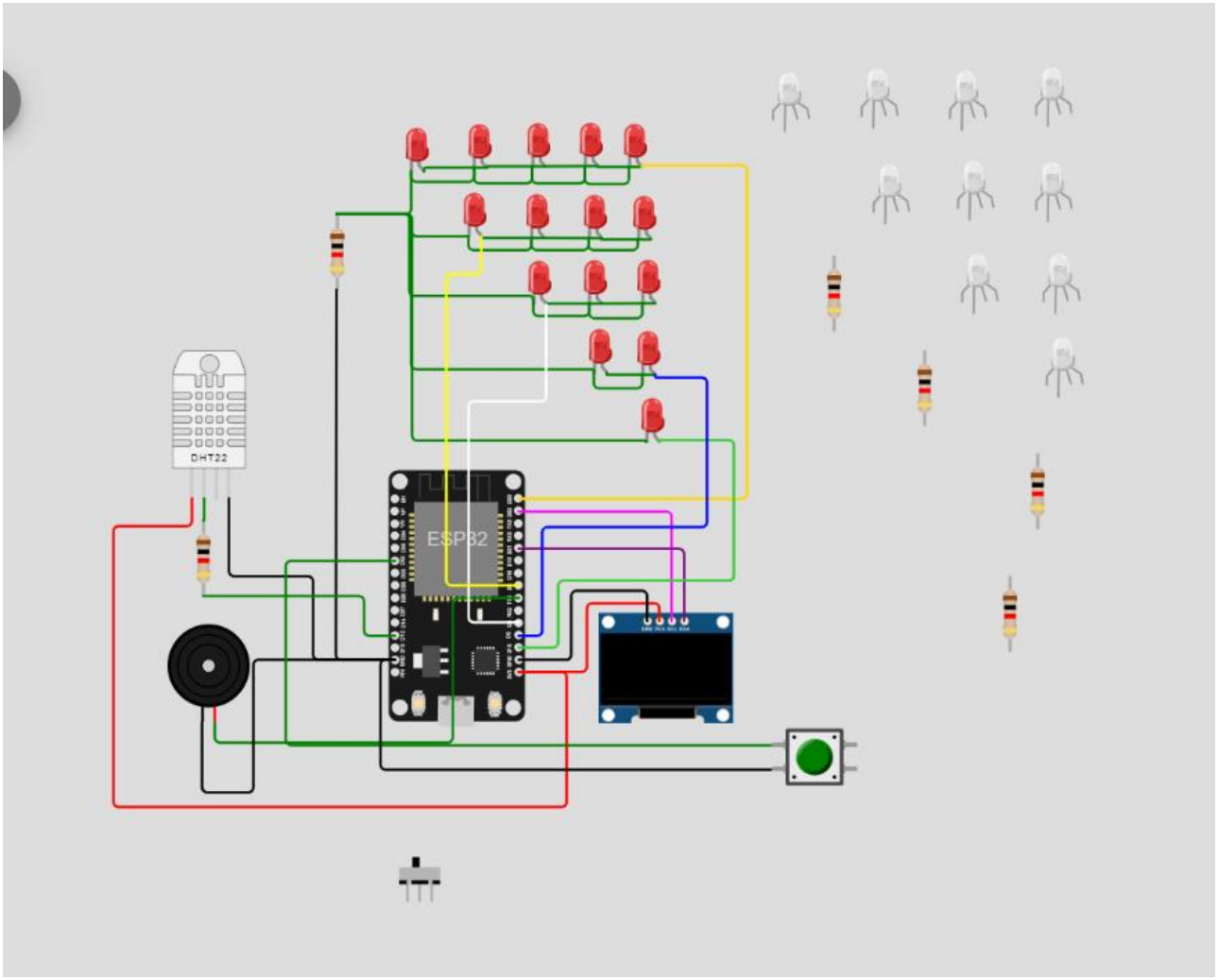
```
}

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= TIMER_INTERVAL_MS) {
    previousMillis = currentMillis;
    if (stage == 0) {
      secondsCounter++;
      if (secondsCounter >= TIME_SLOT_MINUTES * 60) {
        // timer has expired
        digitalWrite(TIMER_PIN, LOW);
        timerExpired = true;
      }

      updateDisplay();
      if (timerExpired) {
        // timer has expired, do something here
        stage = 1;
        secondsCounter = 0;
        timerExpired = false;
        activateBuzzer();
      }
    }
    if (stage == 1) {
      secondsCounter++;
      if (secondsCounter >= TIME_SLOT_INTERVAL * 60) {
        // timer has expired
        digitalWrite(TIMER_PIN, LOW);
        timerExpired = true;
      }

      updateDisplay();
      if (timerExpired) {
        // timer has expired, do something here
        stage = 2;
        secondsCounter = 0;
        timerExpired = false;
        activateBuzzer();
      }
    }
  }
}

void updateDisplay() {
  int minutes = secondsCounter / 60;
  int seconds = secondsCounter % 60;

  // Read temperature from the DHT22 sensor
```

```
  float temperature = dht.readTemperature();

  display.setCursor(0, 0);
  display.print("Time: ");
  display.print(minutes);

  // Display temperature below the time
  //display.setCursor(0, 40);
  //display.print("Temp: ");
  //display.print(temperature);
  //display.print("C");

  if (minutes > 5.0) {
    digitalWrite(15, LOW);
    digitalWrite(2, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(23, LOW);
  }
  else if (minutes >= 5.0) {
    digitalWrite(15, HIGH);
    digitalWrite(2, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(23, HIGH);
  }
  else if (minutes >= 4.0) {
    digitalWrite(15, HIGH);
    digitalWrite(2, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(23, LOW);
  }
  else if (minutes >= 3.0) {
    digitalWrite(15, HIGH);
    digitalWrite(2, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(23, LOW);
  }
  else if (minutes >= 2.0) {
    digitalWrite(15, HIGH);
    digitalWrite(2, HIGH);
    digitalWrite(4, LOW);
    digitalWrite(18, LOW);
    digitalWrite(19, LOW);
  }
  else if (minutes >= 1.0) {
```

```
    digitalWrite(15, HIGH);
    digitalWrite(2, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(23, LOW);
  }

  display.print(":");
  if (seconds < 10) {
    display.print("0");
  }
  display.print(seconds);
  //display.display();


  // Display temperature below the time
  display.setCursor(0, 20);
  display.print("Temp: ");
  display.print(temperature);
  display.print("C");
  display.display();
  display.clearDisplay();
}

void activateBuzzer() {
  digitalWrite(BUZZER_PIN, HIGH); // Turn on the buzzer
  delay(3000); // Wait for 3 seconds
  digitalWrite(BUZZER_PIN, LOW); // Turn off the buzzer
  Serial.println("dbnhcbhbwvef");
}
```

For the simulate use following link :

https://wokwi.com/projects/361635326619138049



References :

https://www.alldatasheet.com/view.jsp?Searchword=Esp32%20datasheet&gad=1&gclid=Cj0KCQjw5f2lBhCkARIsAHeTvlgQoCJ6Br3UEwdj4njg3G7d9eOX8cuN0faZ-SXxV6XMec5bR8UQQM8aAtE0EALw_wcB

https://www.mouser.com/ProductDetail/710-742792114