National Textile University, Faisalabad

Department of Computer Science

Name: Ayesha
Reg_No: 23-NTU-CS-1020
Subject: Embedded IoT
Assignment No: 01
Submitted to: Sir Nasir

Introduction:

The object of this task is to design a smart LED mode control system utilizing the ESP32. The system permits the user to modify different LED modes utilizing push buttons and presents the current mode on an OLED screen. Furthermore, a buzzer is used for audible reactions in user interactions. Four LED modes are performed, which involve Both Off, Alternate Blink, Both On, and PWM Fade, which assist to acknowledge practical control of digital and PWM outputs.

Objectives

The main objectives of this task are:

1. To map a button-controlled LED mode system utilizing the ESP32.
2. To perform multiple LED operating modes such as OFF, Alternate Blink, ON, and PWM Fade.
3. To show the latest operating mode on an OLED screen utilizing the I²C communication.
4. To implement button debouncing techniques.
5. To allow buzzer reaction for user interactions.

Required Components:

- ESP32 Board
- 3 LED
- 2 PushButton
- 1 Buzzer
- 2 Resistor
- OLED

Pin Configuration:

Component	PIN
Button 1	GPIO 32
Button 2	GPIO 33
LED 1	GPIO 5
LED 2	GPIO 4
Buzzer	GPIO 26
OLED SDA	GPIO 21
OLED SCL	GPIO 22

Task A —

Coding: Use one button to cycle through LED modes (display the current state on the OLED):

1. Both OFF
2. Alternate blink
3. Both ON
4. PWM fade

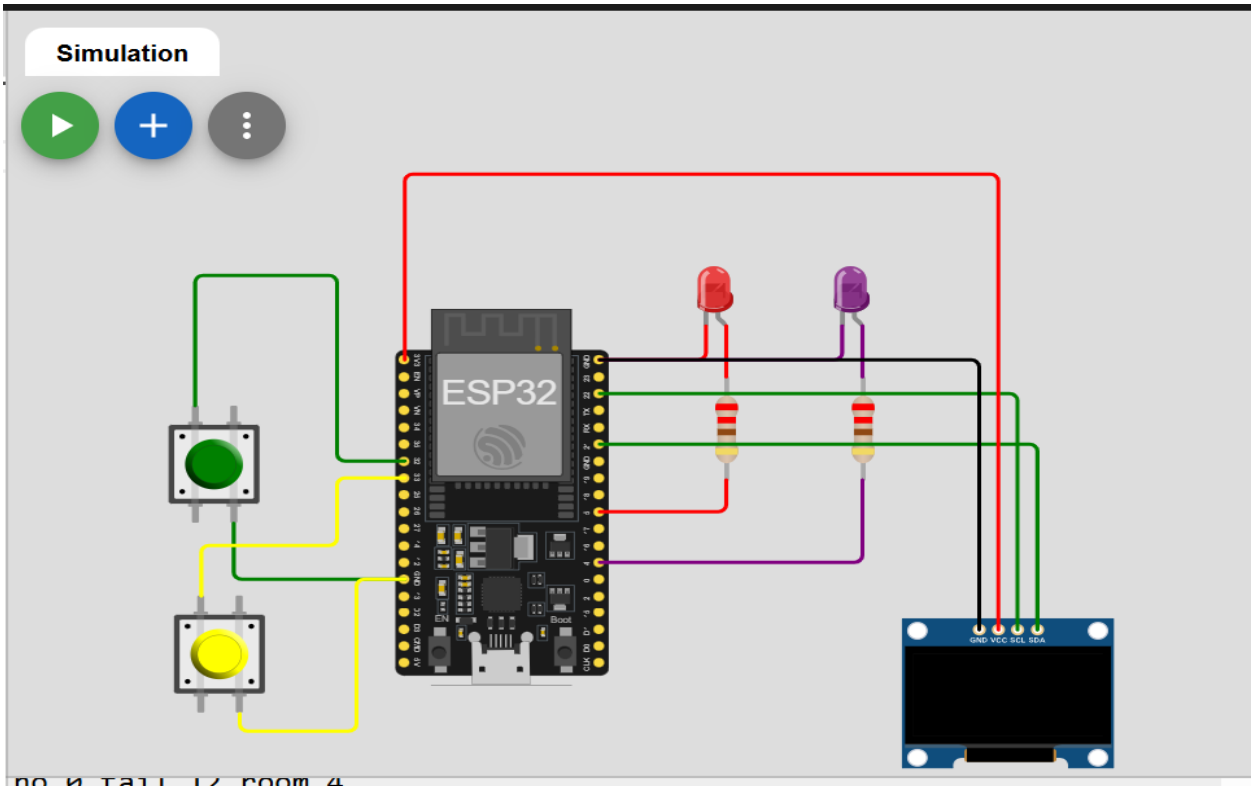
Use the second button to reset to OFF.

Explanation of code:

In this task I use the given libraries:

```
5   #include <Arduino.h>
6   #include <Wire.h>
7   #include <Adafruit_GFX.h>
8   #include <Adafruit_SSD1306.h>
```

<Arduino.h> : I use this library for Arduino functions and data types.
<Wire.h> : It enable I2C communication and communicate with OLED and sensors
<Adafruit_GFX.h> I need this library to display text and shape on the OLED
<Adafruit_SSD1306.h> this library handles communication with screen hardware.



In the given code pin configuration shows which shows the connection of wire with different components

```
8   // --- Pin configuration ---
9   #define BUTTON1_PIN 32    // Mode cycle button
10  #define BUTTON2_PIN 33    // Reset button
11  #define LED1_PIN 5       // First LED
12  #define LED2_PIN 4       // Second LED
13
14  #define SDA_PIN 21        // I2C SDA
15  #define SCL_PIN 22        // I2C SCL
```

These pin definitions help the Arduino to understand where each hardware part is physically attached so it can control them precisely.

```
17  // --- OLED setup ---
18  #define SCREEN_WIDTH 128
19  #define SCREEN_HEIGHT 64
20  Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
21
```

The first 2 lines show the oled display size. &wire use for I2C communication and -1 shows that there is no reset pin.

```
22  // --- Program variables ---
23  int currentMode = 0;      // 0=OFF, 1=Alternate, 2=ON, 3=PWM
24  const int totalModes = 4;
25  unsigned long previousMillis = 0;
26  const long interval = 500;  // Blink interval
27  bool led1State = false;
28  bool led2State = false;
29  int pwmValue = 0;
30  bool pwmDirection = true;  // true = increasing, false = decreasing
31
```

These lines of code show:

- Current mode
- Total mode
- Time Interval
- LED state

```
31
32  // Button debouncing variables
33  int button1State = 0;
34  int lastButton1State = 0;
35  int button2State = 0;
36  int lastButton2State = 0;
37  unsigned long lastDebounceTime1 = 0;
38  unsigned long lastDebounceTime2 = 0;
39  const unsigned long debounceDelay = 50;
40
```

It is used to prevent multiple triggers with single press.

```
41  // --- Mode names for display ---
42  const char* modeNames[] = {
43    "BOTH OFF",
44    "ALTERNATE BLINK",
45    "BOTH ON",
46    "PWM FADE"
47  };
```

It shows the name of state.

```
48
49  void setup() {
50    Serial.begin(115200);
51
52    pinMode(BUTTON1_PIN, INPUT_PULLUP);
53    pinMode(BUTTON2_PIN, INPUT_PULLUP);
54    pinMode(LED1_PIN, OUTPUT);
55    pinMode(LED2_PIN, OUTPUT);
56
57    digitalWrite(LED1_PIN, LOW);
58    digitalWrite(LED2_PIN, LOW);
59
60    Wire.begin(SDA_PIN, SCL_PIN);
61
62    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
63      Serial.println("SSD1306 allocation failed");
64      for (;;);
65    }
66
67    display.clearDisplay();
68    display.setTextColor(SSD1306_WHITE);
69    display.setTextSize(1);
70
71    updateDisplay();
72  }
```

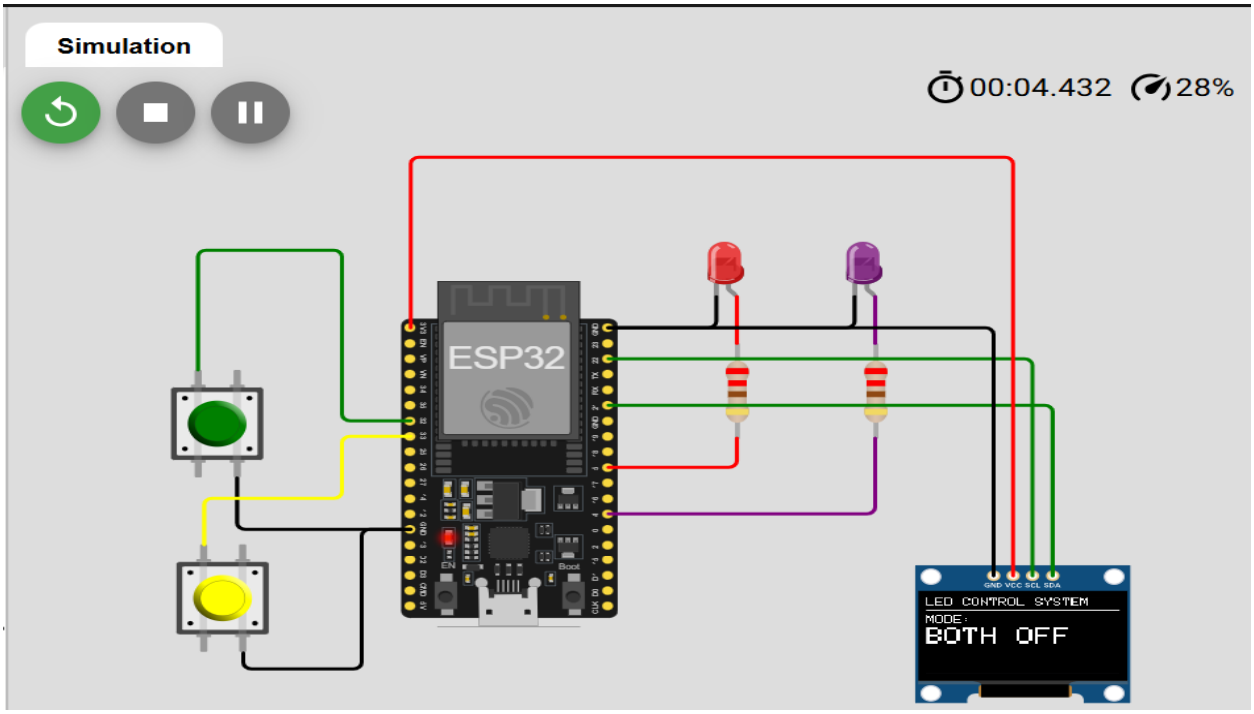
It prepares the setup before the main loop starts.

```
74  void loop() {
75    int reading1 = digitalRead(BUTTON1_PIN);
76    int reading2 = digitalRead(BUTTON2_PIN);
77
78    if (reading1 != lastButton1State) {
79      lastDebounceTime1 = millis();
80    }
81    if ((millis() - lastDebounceTime1) > debounceDelay) {
82      if (reading1 != button1State) {
83        button1State = reading1;
84        if (button1State == LOW) {
85          cycleMode();
86        }
87      }
88    }
89    if (reading2 != lastButton2State) {
90      lastDebounceTime2 = millis();
91    }
92    if ((millis() - lastDebounceTime2) > debounceDelay) {
93      if (reading2 != button2State) {
94        button2State = reading2;
95        if (button2State == LOW) {
96          resetToOff();
97        }
98      }
99    }
```

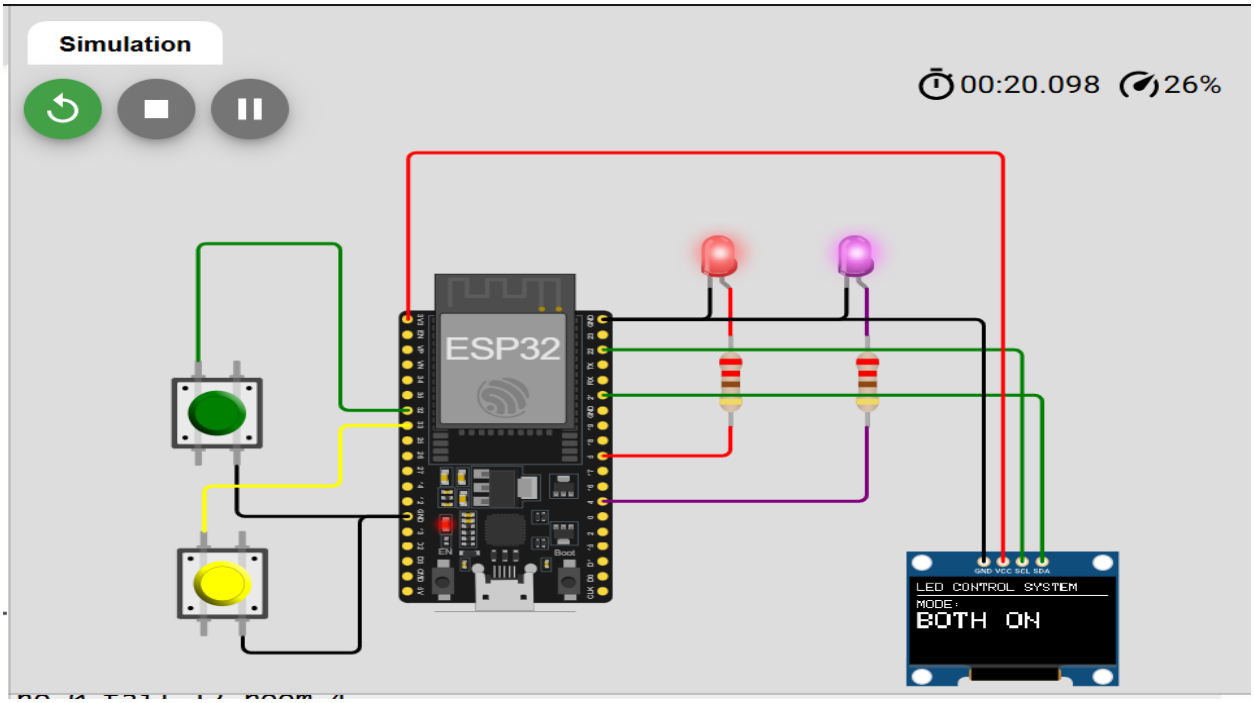
This part of code checks the button states if button 1 is pressed it changes the LED mode but if button 2 is pressed it reset the states.

```
103   switch(currentMode) {
104     case 0:
105       digitalWrite(LED1_PIN, LOW);
106       digitalWrite(LED2_PIN, LOW);
107       break;
108
109     case 1:
110       handleAlternateBlink();
111       break;
112
113     case 2:
114       digitalWrite(LED1_PIN, HIGH);
115       digitalWrite(LED2_PIN, HIGH);
116       break;
117
118     case 3:
119       handlePWMFade();
120       break;
121   }
122 }
123 void cycleMode() {
124   currentMode = (currentMode + 1) % totalModes;
125   updateDisplay();
126 }
```

It shows different cases which handle the different mode
In cycle mode when button is pressed it add 1 to move to the next mode.
In it case 0 show the scenario when both LED are off



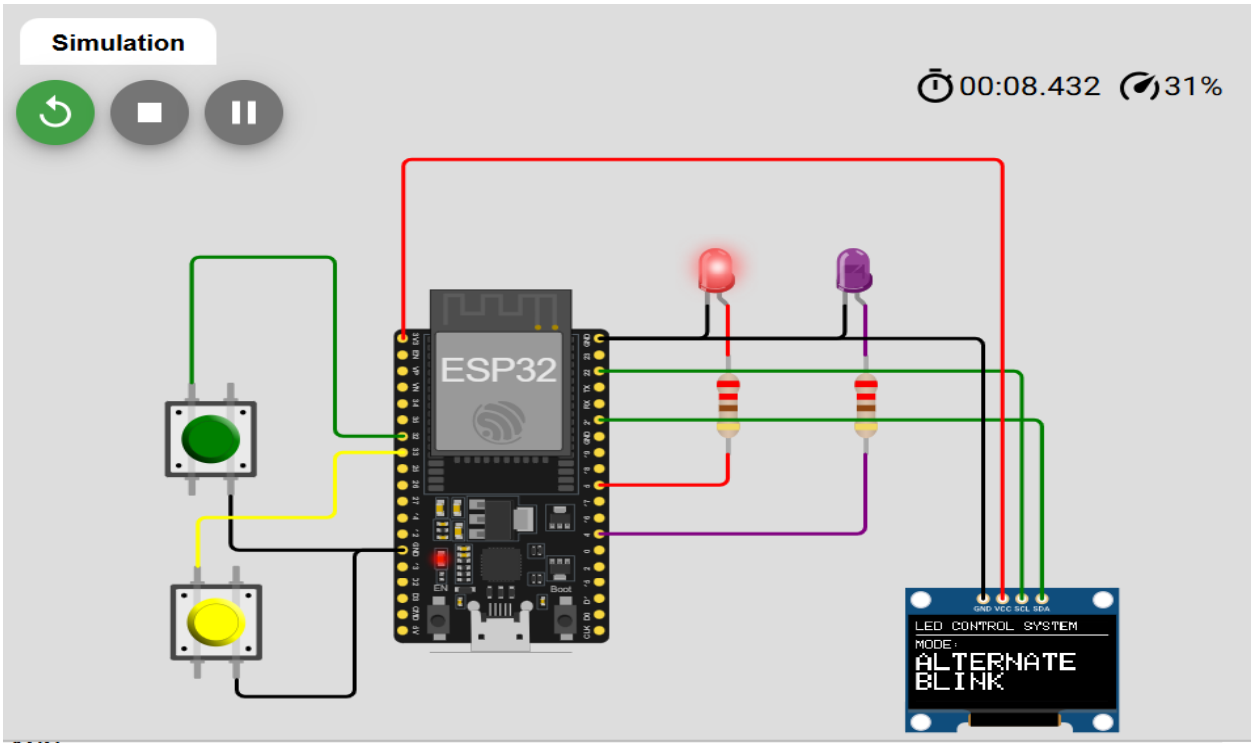
While case 2 control the situation in which both LED are on.



Case 1 and 2 are described in below explanation.

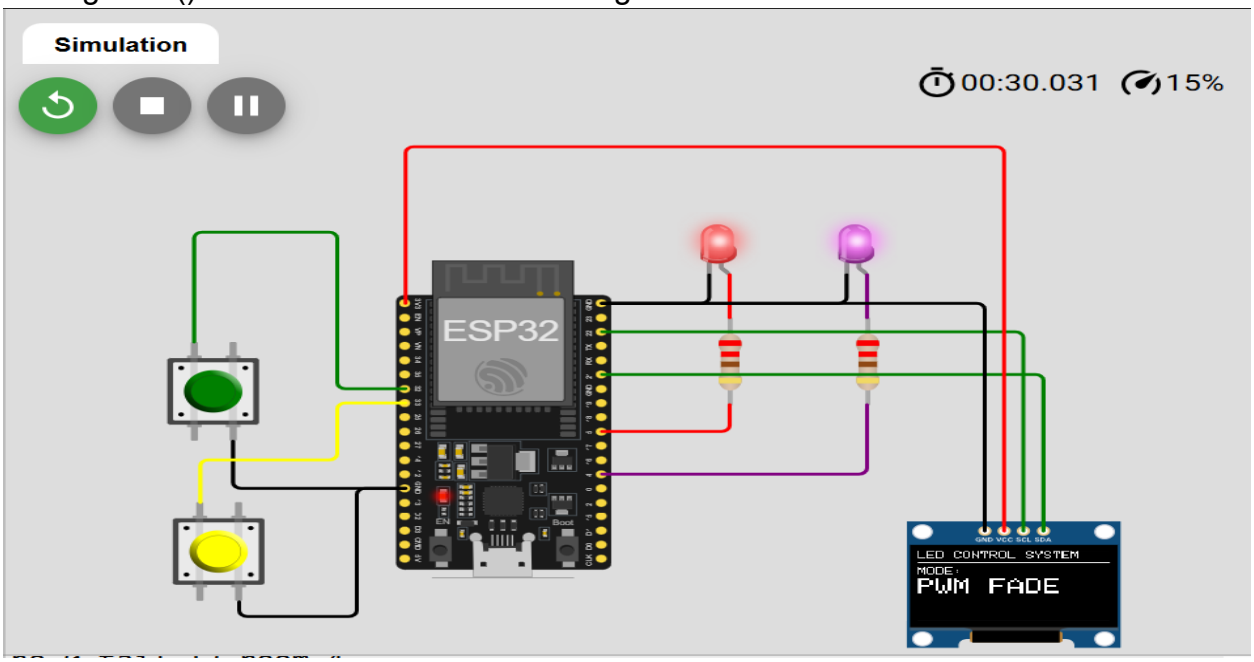
```
127
128   void resetToOff() {
129     currentMode = 0;
130
131     // Force LEDs OFF instantly
132     digitalWrite(LED1_PIN, LOW);
133     digitalWrite(LED2_PIN, LOW);
134
135     updateDisplay();
136   }
137   void handleAlternateBlink() {
138     unsigned long currentMillis = millis();
139
140     if (currentMillis - previousMillis >= interval) {
141       previousMillis = currentMillis;
142
143       led1State = !led1State;
144       led2State = !led1State;
145
146       digitalWrite(LED1_PIN, led1State);
147       digitalWrite(LED2_PIN, led2State);
148     }
149   }
```

It shows when button 2 is pressed both LED turn off and display the alternate blink function which shows how to handle both led blinks that when 1 led is on other is off.



```
150   void handlePWMFade() {
151     unsigned long currentMillis = millis();
152
153     if (currentMillis - previousMillis >= 20) {
154       previousMillis = currentMillis;
155
156       if (pwmDirection) {
157         pwmValue += 5;
158         if (pwmValue >= 255) {
159           pwmValue = 255;
160           pwmDirection = false;
161         }
162       } else {
163         pwmValue -= 5;
164         if (pwmValue <= 0) {
165           pwmValue = 0;
166           pwmDirection = true;
167         }
168       }
169
170       analogWrite(LED1_PIN, pwmValue);
171       analogWrite(LED2_PIN, pwmValue);
172     }
173   }
```

It controls the PMW cycle and displays the function how we control it, and we use analogWrite() function in it to control the brightness level.



```
174  void updateDisplay() {
175    display.clearDisplay();
176    display.setTextSize(1);
177
178    display.setCursor(0, 0);
179    display.println("LED CONTROL SYSTEM");
180    display.drawLine(0, 10, 127, 10, SSD1306_WHITE);
181
182    display.setCursor(0, 15);
183    display.print("MODE: ");
184
185    display.setTextSize(2);
186    display.setCursor(0, 25);
187    display.println(modeNames[currentMode]);
188
189    display.display();
190  }
```

These lines adjust the text on screen where we display which line on OLED

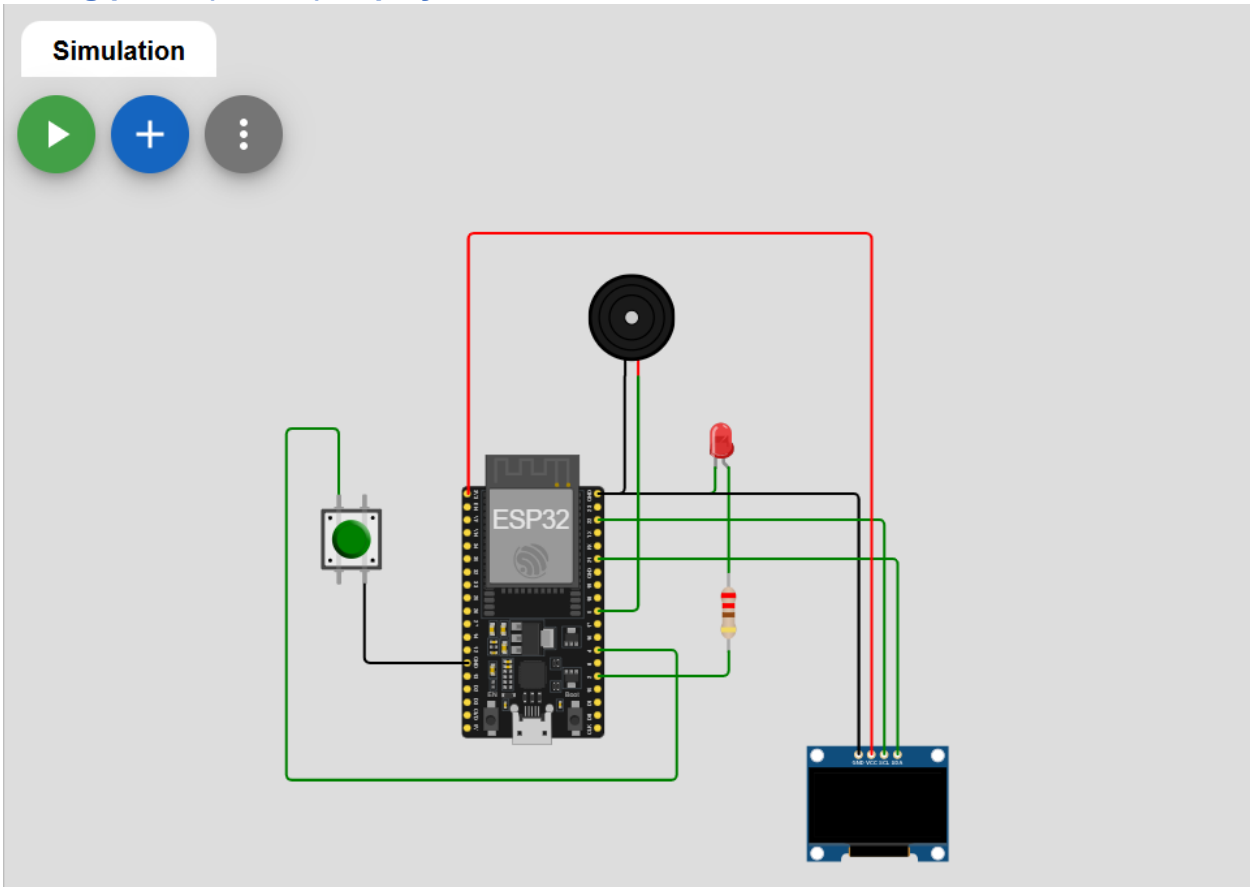
Wokwi Link:

https://wokwi.com/projects/445625588391625729

Task B — Coding:

Use a single button with press-type detection (display the event on the OLED):

- **Short press → toggle LED**
- **Long press (> 1.5 s) → play a buzzer tone**



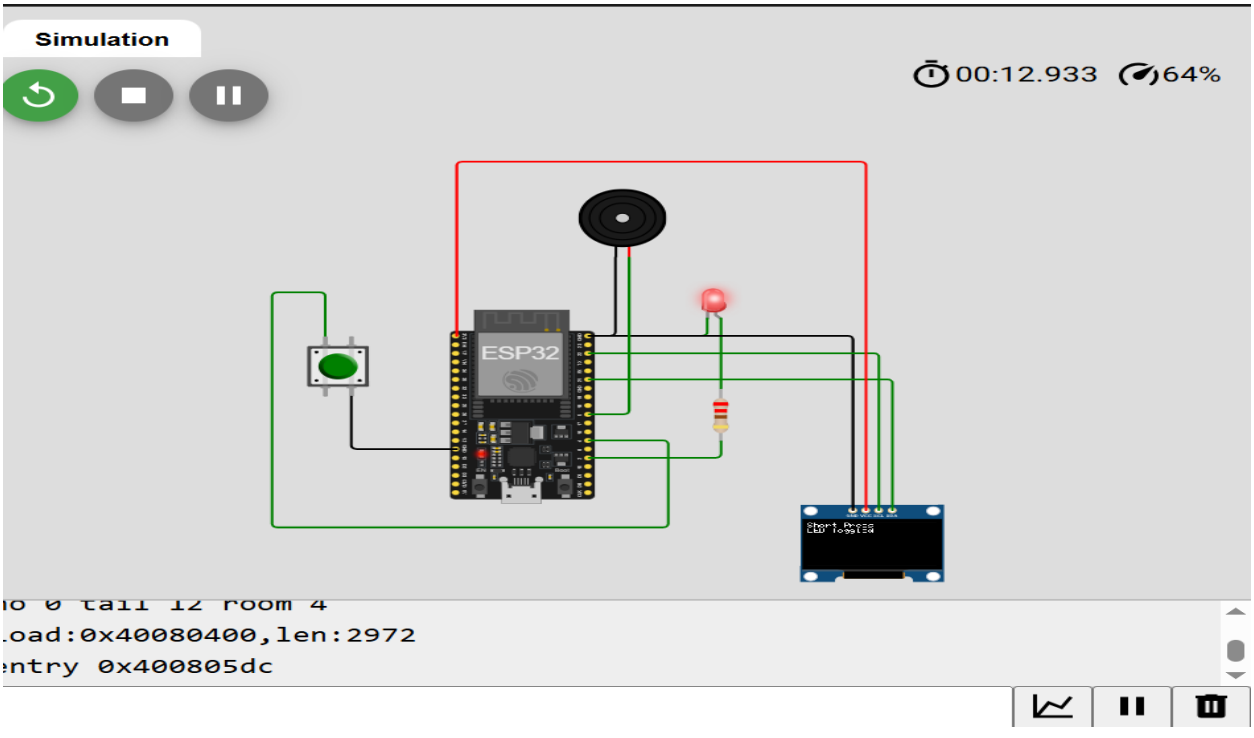
This is the setup for task B

```
4
5   #include <Wire.h>
6   #include <Adafruit_GFX.h>
7   #include <Adafruit_SSD1306.h>
8
9   #define BUTTON 4
10  #define LED 2
11  #define BUZZER 5
12
```

I use three libraries in this task which I explained in task A. The other lines show the numbers of buttons, LEDs, and buzzer pins connected to the ESP32.

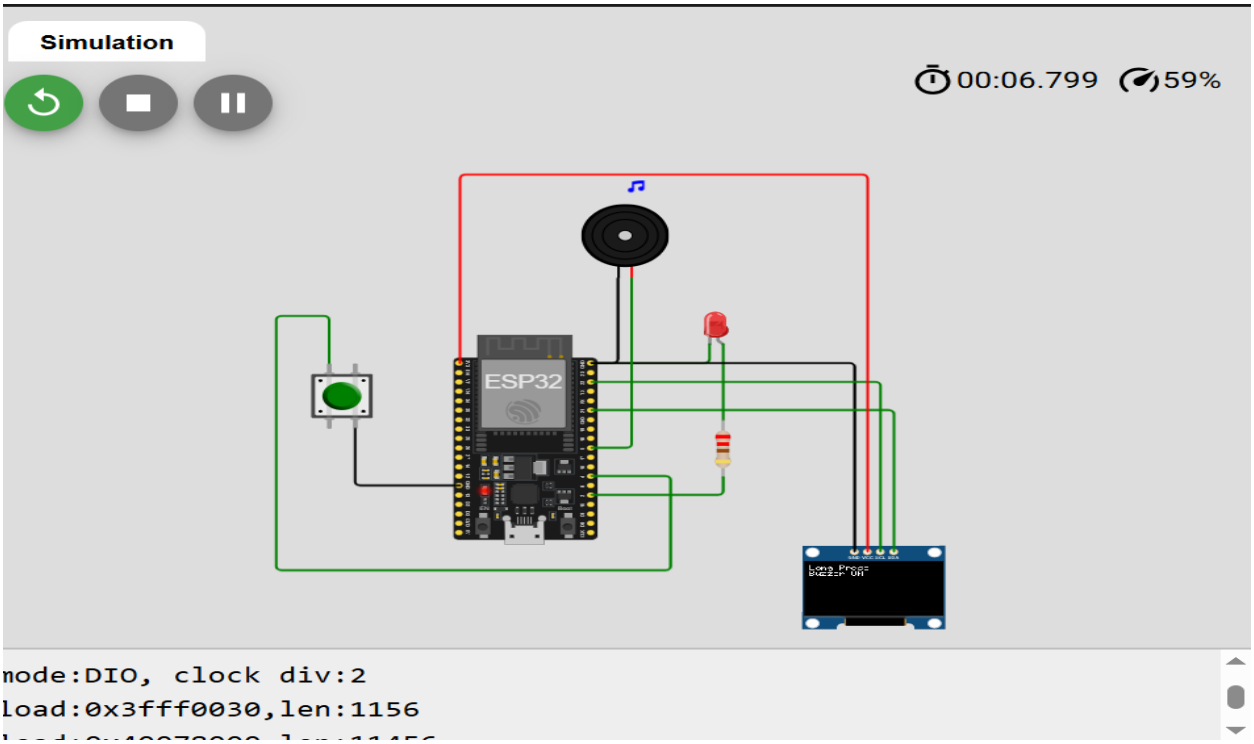
```
15  unsigned long pressTime = 0;
16  bool buttonState = false;
17  bool lastButtonState = false;
18  bool ledState = false;
19
20  void setup() {
21    pinMode(BUTTON, INPUT_PULLUP);
22    pinMode(LED, OUTPUT);
23    pinMode(BUZZER, OUTPUT);
24
25    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
26    display.clearDisplay();
27    display.setTextSize(1);
28    display.setTextColor(WHITE);
29    display.setCursor(0, 0);
30    display.println("Ready...");
31    display.display();
32  }
```

These lines record
The press time because short press turns on the LED and long press turn on the buzzer it also checks the button and led state.
In setup() function it checks the button as input and LED and buzzer as output state
In the last lines of code, it controls how the text is displayed on OLED and the color of text is set as white.



```
34  void loop() {
35    buttonState = digitalRead(BUTTON) == LOW;
36    if (buttonState && !lastButtonState) {
37      pressTime = millis(); // button just pressed
38    }
39    // when button released
40    if (!buttonState && lastButtonState) {
41      unsigned long pressDuration = millis() - pressTime;
42
43      display.clearDisplay();
44      display.setCursor(0, 0);
45
46      if (pressDuration < 1500) {
47        // short press → toggle LED
48        ledState = !ledState;
49        digitalWrite(LED, ledState);
50        display.println("Short Press");
51        display.println("LED Toggled");
52      }
53      else {
54        // long press → buzzer tone
55        display.println("Long Press");
56        display.println("Buzzer ON");
57        tone(BUZZER, 1000, 500); // 1kHz tone for 0.5s
58      }
```

The first line checks whether the button is pressed.
If button is pressed it can store the time. Pressduration variable count how long the button was pressed.
If pressduration is <1.5ms then turn on the LED otherwise turn on the buzzer if duration of press is long



Wokwi Link:

https://wokwi.com/projects/445499738958735361

Conclusion:

These tasks successfully performed a button-controlled LED mode utilizing the ESP32 microcontroller. The system demonstrated four operating modes involving OFF, Alternate Blink, ON, and PWM Fade, enable us to understand digital and PWM control.