**National Textile University, Faisalabad**

A logo with blue and red squares

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **Name:** | **Fiza Mahboob** |
| **Class:** | **BSCS(A)** |
| **Semester:** | **5th** |
| **Registration No:** | **23-NTU-CS-1027** |
| **Course Name:** | **Embedded IoT Systems** |
| **Assignment:** | **1** |
| **Submitted To:** | **Sir Nasir** |

**ESP32 Multi-Mode LED Controller with OLED Display**

**1. Project Overview**

This project implements a **Multi-Mode LED Controller** using an **ESP32 microcontroller** and an **OLED display** (SSD1306).  
The system allows the user to **cycle between different LED modes** using one push button and **reset all LEDs** using another.

An **OLED screen** displays the current LED mode in real time.  
The program includes **debouncing logic**, **PWM fading**, and **non-blocking timing** for smooth performance.

**2. Project Objectives**

* To control multiple LEDs using push buttons with software debouncing.
* To use **OLED display** for real-time feedback of LED states.
* To apply **PWM (Pulse Width Modulation)** for LED brightness control.
* To practice **non-blocking code structure** using millis() instead of delay().
* To design a complete embedded system circuit using **ESP32 and external peripherals**.

**3. Components Used**

| **Component** | **Quantity** | **Description** |
| --- | --- | --- |
| ESP32 Dev Board | 1 | Main microcontroller |
| Push Buttons | 2 | One for mode change, one for reset |
| LEDs | 3 | Two main LEDs and one indicator LED |
| Resistors (10kΩ) | 2 | Used as pull-down or series protection |
| OLED Display (SSD1306, I2C) | 1 | 128x64 pixel OLED for display output |
| Buzzer | 1 | Audio indication |
| Breadboard & Jumper Wires | — | For circuit connections |

**4. Circuit Connections**

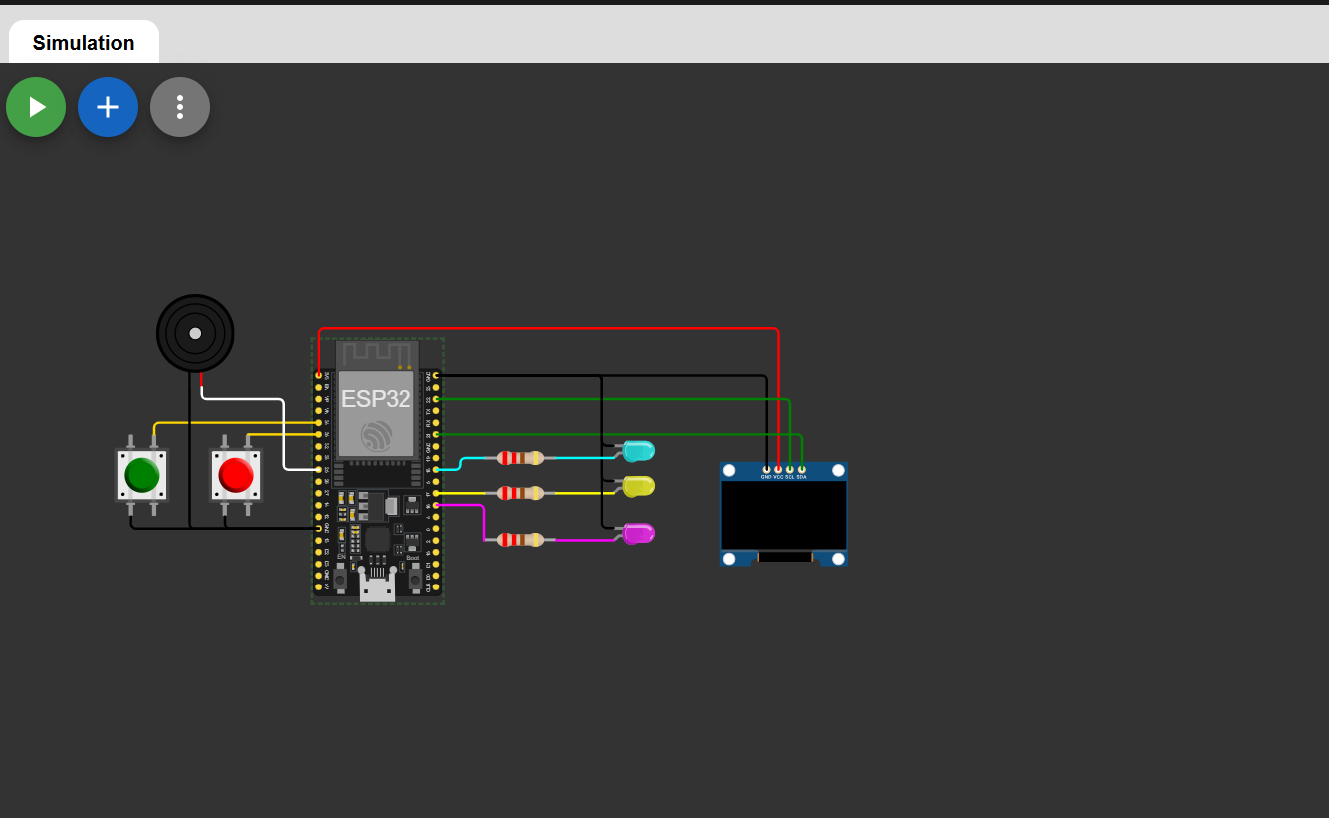
| **Component** | **ESP32** | **Pin Description** |
| --- | --- | --- |
| LED1 | GPIO 18 | Main LED 1 |
| LED2 | GPIO 17 | Main LED 2 |
| LED3 | GPIO 16 | Indicator LED for button press |
| Button 1 (Mode Btn) | GPIO 34 | Used to switch between LED modes |
| Button 2 (Reset Btn ) | GPIO 35 | Used to reset system to “BOTH OFF” |
| OLED SDA | GPIO 21 | Data line (I2C) |
| OLED SCL | GPIO 22 | Clock line (I2C) |
| Buzzer | GPIO 25 | Optional sound output |

**Note:** Buttons use **INPUT\_PULLUP**, so they remain HIGH when idle and LOW when pressed.

**5. Circuit Diagram (Wokwi Simulation)**

You can design this circuit on **Wokwi.com** and export the JSON file.  
**Wiring color suggestions:**

* **Red:** Power (VCC)
* **Black:** Ground (GND)
* **Green:** Data Lines (SDA/SCL)
* **Yellow:** Signal lines for buttons



**6. Working Principle**

The project operates in four LED modes, controlled by the **Mode Button**:

| **Mode** | **Description** | **OLED Message** |
| --- | --- | --- |
| 1. BOTH OFF | All LEDs remain off | BOTH OFF |
| 2. ALT BLINK | LEDs blink alternately | ALT BLINK |
| 3. BOTH ON | Both LEDs stay continuously on | BOTH ON |
| 4. PWM FADE | LEDs fade in/out using PWM | PWM FADE |

**Mode Button (GPIO 34)**

Each press cycles through the modes in sequence (1 → 2 → 3 → 4 → 1).

**Reset Button (GPIO 35)**

When pressed, it resets the mode back to “BOTH OFF”.

**Indicator LED (GPIO 16)**

Briefly lights up when a valid button press is detected (visual confirmation).

**OLED Display (SDA=21, SCL=22)**

Displays the **name of the current mode** in large, readable text.

**7. Features Implemented**

* Button debouncing using time-based logic (millis())
* Non-blocking LED control
* OLED text updates using **Adafruit\_SSD1306** library
* PWM fading effect for LEDs
* Mode cycling and reset logic
* Code written with **meaningful variable names and clear comments**

**8. Source Code**

#include <Arduino.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

// -------- OLED Configuration ---------

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_ADDR 0x3C

// -------- Pin Definitions -----------

const int LED1\_PIN = 18;

const int LED2\_PIN = 17;

const int LED3\_PIN = 16;

const int BTN\_MODE  = 34;

const int BTN\_RESET = 35;

const int BUZZER\_PIN = 25;

// -------- Debounce Configuration --------

const unsigned long DEBOUNCE\_DELAY = 25; // Debounce time in ms

// OLED display object

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

// -------- Operating Modes -----------

enum Mode { BOTH\_OFF, ALT\_BLINK, BOTH\_ON, PWM\_FADE };

Mode currentMode = BOTH\_OFF;

// ---------- Timing Variables -----------

unsigned long lastToggle = 0;       // Tracks LED blink timing

const unsigned long blinkInterval = 400; // Time interval for alternating blink

// --------- Button State Tracking -----------

unsigned long lastModePress = 0;

unsigned long lastResetPress = 0;

int lastModeState = HIGH;

int lastResetState = HIGH;

bool modePressed = false;

bool resetPressed = false;

// --------- Fade Control Variables ------------

int fadeVal = 0;

int fadeDir = 1;

// --------- Function Prototypes -------------

void setLedsToOff();

void setLedsToHigh();

void updateOLED(const char \*msg);

void handleModeChange();

void setup() {

  Serial.begin(115200);

    // Configure pin modes

  pinMode(LED1\_PIN, OUTPUT);

  pinMode(LED2\_PIN, OUTPUT);

  pinMode(LED3\_PIN, OUTPUT);

  pinMode(BTN\_MODE, INPUT\_PULLUP);

  pinMode(BTN\_RESET, INPUT\_PULLUP);

  pinMode(BUZZER\_PIN, OUTPUT);

   // Initialize I2C and OLED

Wire.begin(21, 22);

  if (!display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR)) {

    Serial.println("OLED failed!");

    for (;;); // Stop execution if OLED setup fails

  }

 // Display initial mode

  updateOLED("BOTH OFF");

  setLedsToOff();

  Serial.println("Setup complete");

}

void loop() {

  unsigned long now = millis();

  // -------- MODE Button Handler --------

  int readingMode = digitalRead(BTN\_MODE);

     // Detect change in state

  if (readingMode != lastModeState) {

    lastModePress = now;

  }

    // Debounce check

  if ((now - lastModePress) > DEBOUNCE\_DELAY) {

    if (readingMode == LOW && !modePressed) {

      modePressed = true;

      digitalWrite(LED3\_PIN, HIGH);

      handleModeChange();

      Serial.println("Mode button pressed");

    }

    else if (readingMode == HIGH) {

      modePressed = false;

      digitalWrite(LED3\_PIN, LOW);

    }

  }

  lastModeState = readingMode;

// -------- RESET Button Handler -----------

  int readingReset = digitalRead(BTN\_RESET);

   if (readingReset != lastResetState) {

    lastResetPress = now;

  }

  if ((now - lastResetPress) > DEBOUNCE\_DELAY) {

    if (readingReset == LOW && !resetPressed) {

      resetPressed = true;

      digitalWrite(LED3\_PIN, HIGH);

      currentMode = BOTH\_OFF;

      updateOLED("BOTH OFF");

      setLedsToOff();

      Serial.println("Reset button pressed");

    }

    else if (readingReset == HIGH) {

      resetPressed = false;

      digitalWrite(LED3\_PIN, LOW);

    }

  }

  lastResetState = readingReset;

// ------- LED behavior based on Mode -------

  switch (currentMode) {

    case BOTH\_OFF:

      break;

    case ALT\_BLINK:

    // Alternate LED blinking

      if (now - lastToggle >= blinkInterval) {

        lastToggle = now;

        static bool toggle = false;

        toggle = !toggle;

                analogWrite(LED1\_PIN, toggle ? 255 : 0);

        analogWrite(LED2\_PIN, toggle ? 0 : 255);

        digitalWrite(LED3\_PIN, LOW);

      }

      break;

case BOTH\_ON:

    // Both LEDs stay continuously on

      break;

case PWM\_FADE:

    // Smooth fading of both LEDs using PWM

      fadeVal += fadeDir \* 1;

            if (fadeVal >= 180) { fadeVal = 180; fadeDir = -1; }

      if (fadeVal <= 0)   { fadeVal = 0;   fadeDir = 1;  }

      analogWrite(LED1\_PIN, fadeVal);

      analogWrite(LED2\_PIN, fadeVal);

      digitalWrite(LED3\_PIN, LOW);

      delay(20);

      break;

  }

}

// -------- FUNCTION DEFINITIONS ----------

// Turns off all LEDs

void setLedsToOff() {

  analogWrite(LED1\_PIN, 0);

  analogWrite(LED2\_PIN, 0);

  digitalWrite(LED3\_PIN, LOW);

}

// Turns on all LEDs at full brightness

void setLedsToHigh() {

  analogWrite(LED1\_PIN, 255);

  analogWrite(LED2\_PIN, 255);

  digitalWrite(LED3\_PIN, HIGH);

}

// Handles switching between LED operation modes

void handleModeChange() {

    currentMode = (Mode)((currentMode + 1) % 4);

    switch (currentMode) {

      case BOTH\_OFF:

        updateOLED("BOTH OFF");

        setLedsToOff();

        break;

      case ALT\_BLINK:

        updateOLED("ALT BLINK");

        setLedsToOff();

        lastToggle = millis();

        break;

      case BOTH\_ON:

        updateOLED("BOTH ON");

        setLedsToHigh();

        break;

      case PWM\_FADE:

        updateOLED("PWM FADE");

        fadeVal = 0;

        fadeDir = 1;

        break;

    }

}

// Updates the OLED display with current mode information

void updateOLED(const char \*msg) {

  display.clearDisplay();

  display.setTextSize(2);

  display.setTextColor(SSD1306\_WHITE);

  display.setCursor(0, 20);

  display.println(msg);

  display.display();

}

**9. Code Explanation**

**a. Setup Section**

* Initializes OLED display, sets up pins for LEDs, buttons, and buzzer.
* Starts with the **“BOTH OFF”** mode displayed on OLED.

**b. Loop Section**

* Continuously checks both buttons with debounce logic.
* Handles **mode changes** and **reset actions**.
* Executes LED behavior depending on the current mode.

**c. Helper Functions**

1. **setLedsToOff()**  
   Turns all LEDs OFF.
2. **setLedsToHigh()**  
   Turns all LEDs ON.
3. **handleModeChange()**  
   Increments mode, updates OLED text, and sets LEDs accordingly.
4. **updateOLED(msg)**  
   Clears screen and prints the current mode name.

**10. Output Demonstration**

**Mode 1-Both OFF:**

**A screenshot of a computer

AI-generated content may be incorrect.**

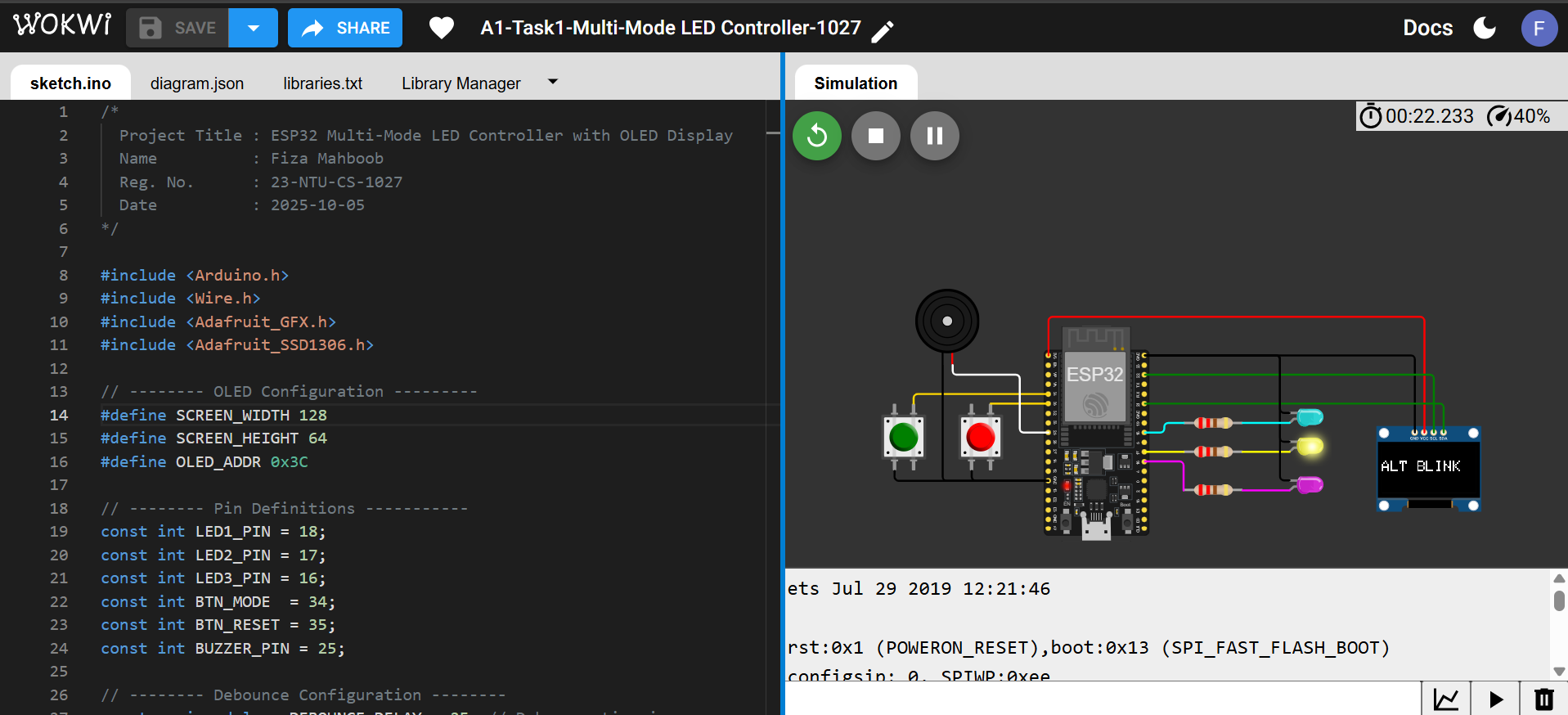
**Mode 2-** **Alternate blink:**

**LED1-ON**

**A screenshot of a computer

AI-generated content may be incorrect.**

**LED2-ON**

****

**Mode3-Both ON:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Mode4-PWM Fade**

**A screenshot of a computer

AI-generated content may be incorrect.**

**RESET-To OFF:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**11. Conclusion**

This project successfully demonstrates how to integrate **multiple hardware peripherals** (LEDs, buttons, buzzer, and OLED) with an ESP32 board.  
Through this implementation, **PWM control**, **debouncing techniques**, and **I2C communication** are effectively utilized.  
It provides a practical understanding of **real-time embedded systems** and **non-blocking event-driven programming**.