National Textile University, Faisalabad

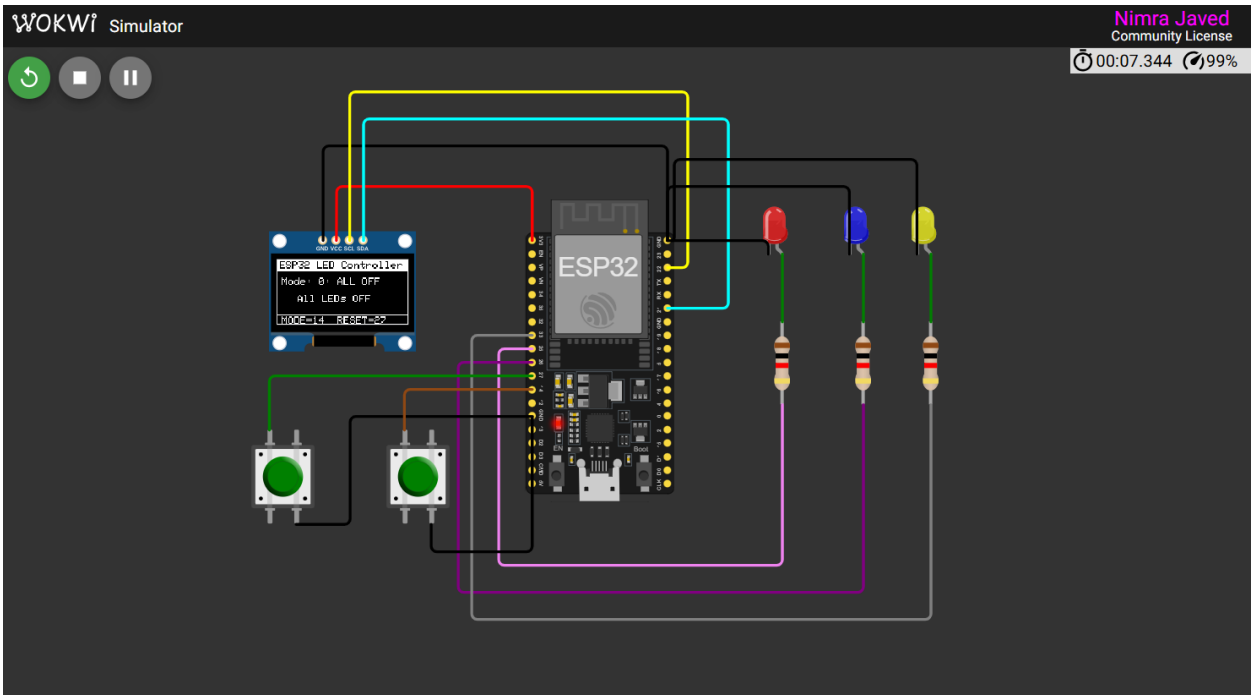
Department of Computer Science

Name	Nimra Javed
Section	BSCS-B
Semester	5th
Registration no.	23-NTU-CS-1082
Course title	Embedded IOT System
Submitted to	Sir Nasir Mahmood
Submission date	23-Oct-2025

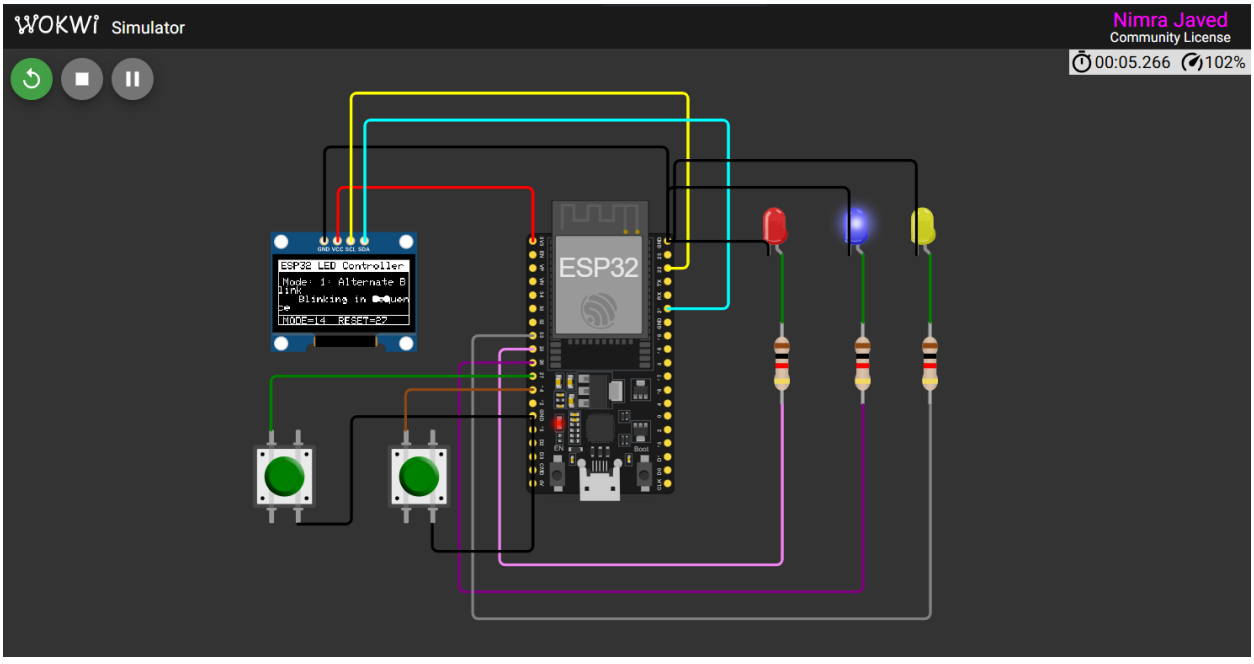
TASK A

Wokwi diagram:

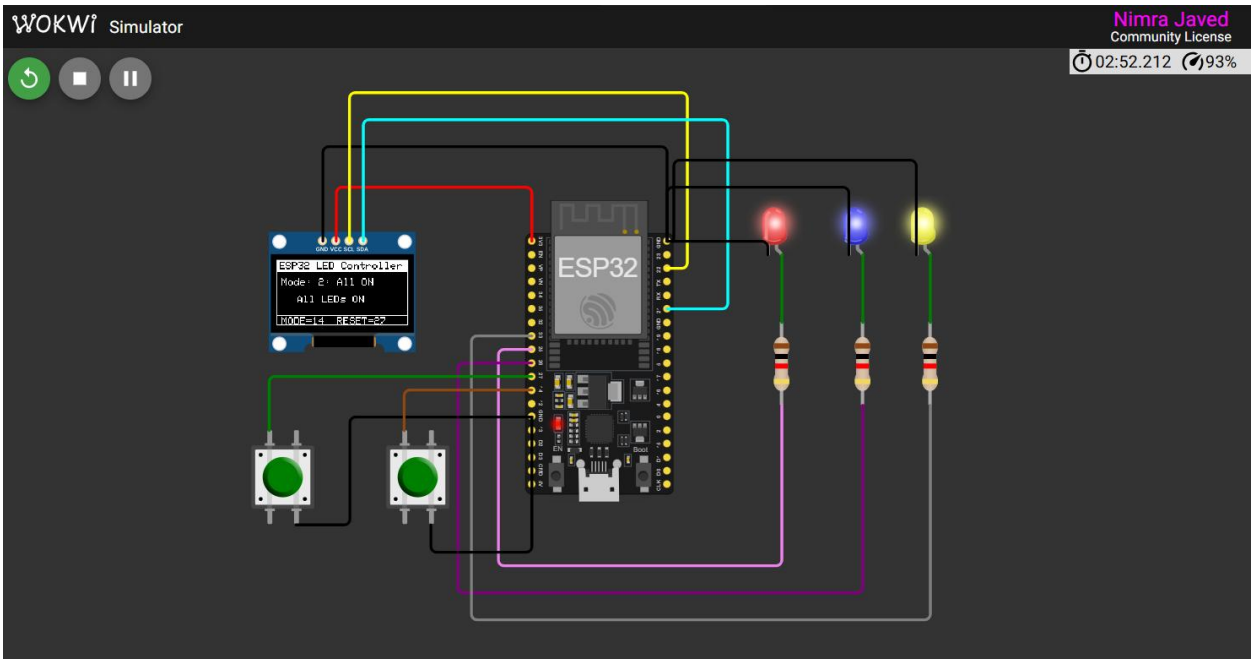
Mode 0:



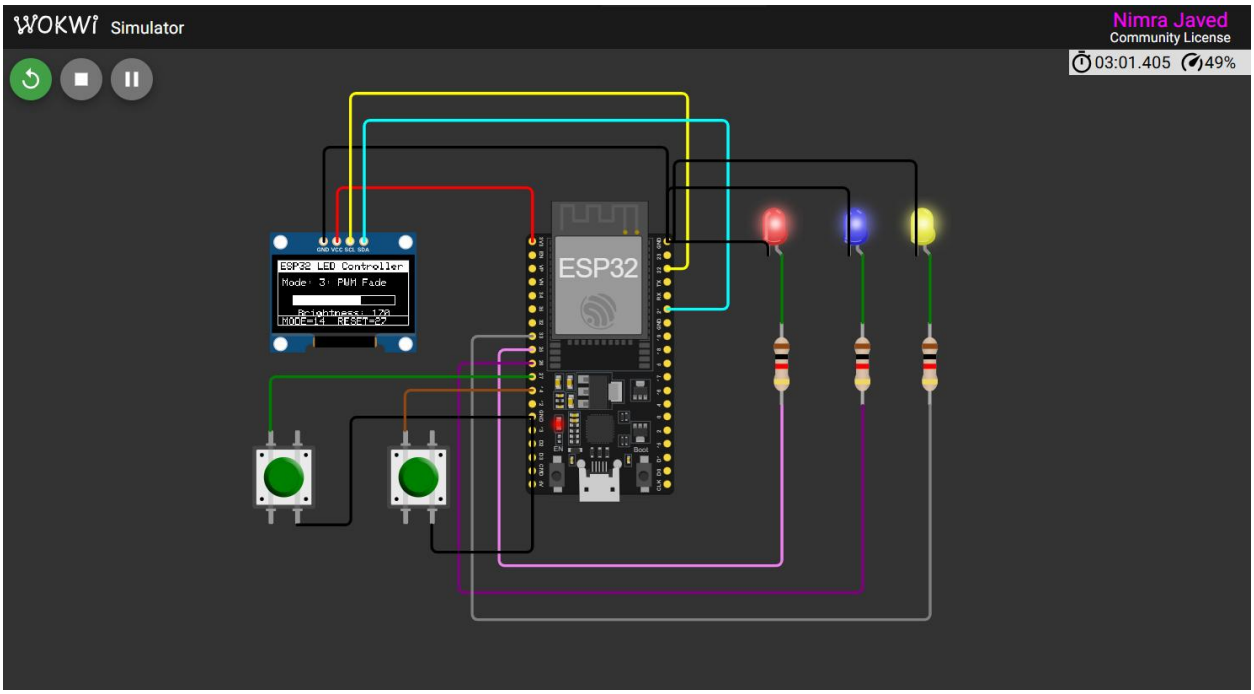
Mode 1:



Mode 2:


WOKWI Simulator
Nimra Javed
Community License
02:52.212
93%
ESP32 LED Controller
Mode: 2: All ON
All LEDs ON
MODE=14 RESET=27
ESP32

Mode 3:


WOKWI Simulator
Nimra Javed
Community License
03:01.405
49%
ESP32 LED Controller
Mode: 3: PWM Fade
MODE=14 RESET=27
ESP32

Pin diagram: (ESP32 with OLED and LEDs)

ESP32 Pin	Label	Connected To	Description
25	LED1	LED1 (via 220Ω resistor)	First LED output
26	LED2	LED2 (via 220Ω resistor)	Second LED output
33	LED3	LED3 (via 220Ω resistor)	Third LED output
14	BTN_MODE	Push Button 1 → GND	Changes LED mode
27	BTN_RESET	Push Button 2 → GND	Resets mode
21	SDA	OLED SDA	I^2C data line
22	SCL	OLED SCL	I^2C clock line
3.3V	VCC	OLED VCC	Power for OLED
GND	GND	OLED, LEDs, Buttons	Common ground

Code:

platformio.ini U | README.md U | main.cpp U | diagram.json U | Wokwi Simulator | PIO Home | diagram.json U

src > main.cpp > loop()

```cpp
// Name: Nimra Javed
// Reg no: 23-NTU-CS-1082
// Section: BS-CS_5th B
// Title: Task A - LED Modes with OLED and Push Buttons

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_ADDR 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// --- Pin Configuration ---
#define LED1 25          // First LED pin
#define LED2 26          // Second LED pin
#define LED3 33          // Third LED pin
#define BTN_MODE 14      // Button to change LED mode
#define BTN_RESET 27     // Button to reset to OFF mode

// --- PWM Configuration ---
#define PWM_CH1 0        // PWM Channel for LED1
#define PWM_CH2 1        // PWM Channel for LED2
#define PWM_CH3 2        // PWM Channel for LED3
#define PWM_FREQ 5000    // PWM frequency in Hz
#define PWM_RES 8        // PWM resolution (8-bit = 0–255)

int mode = 0;                // Current LED mode
unsigned long lastPress = 0;  // For button debounce
int brightness = 0;          // Current LED brightness (for fade)
int fadeAmount = 10;         // Brightness change per loop in fade mode

void setup() {
  Serial.begin(115200);

  // --- Configure buttons as input with internal pull-up ---
  pinMode(BTN_MODE, INPUT_PULLUP);
  pinMode(BTN_RESET, INPUT_PULLUP);
```

platformio.ini U | README.md U | main.cpp U | diagram.json U | Wokwi Simulator | PIO Home | diagram.json U

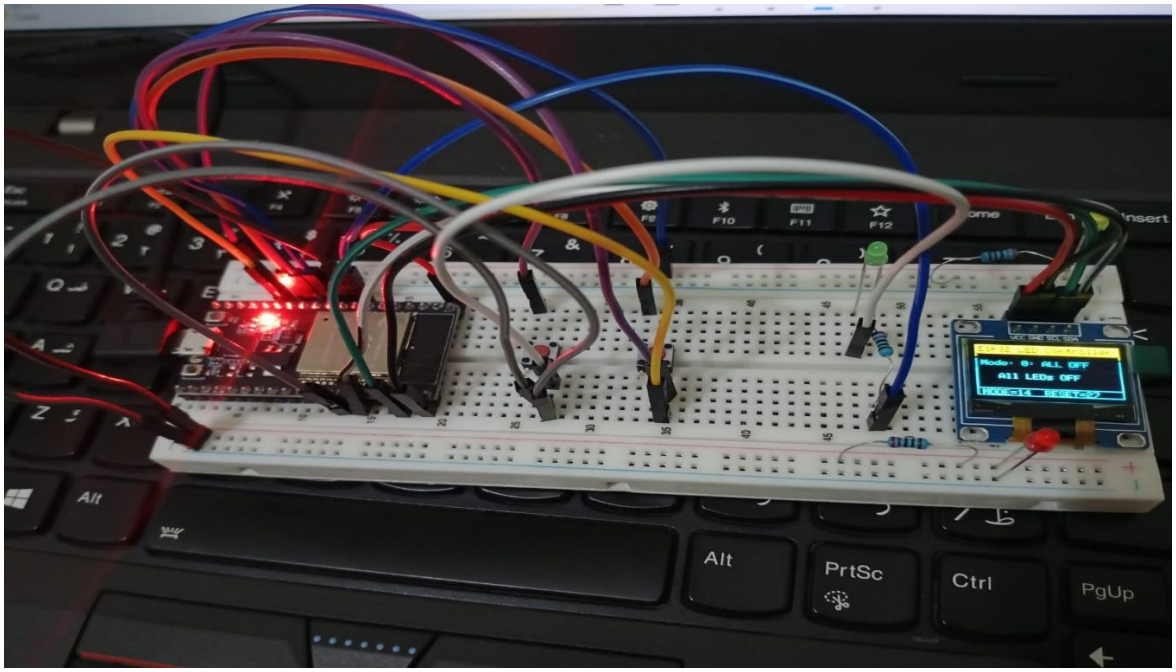
src > main.cpp > loop()

```cpp
void setup() {
  // --- Setup PWM for LEDs ---
  ledcSetup(PWM_CH1, PWM_FREQ, PWM_RES);
  ledcSetup(PWM_CH2, PWM_FREQ, PWM_RES);
  ledcSetup(PWM_CH3, PWM_FREQ, PWM_RES);
  ledcAttachPin(LED1, PWM_CH1);
  ledcAttachPin(LED2, PWM_CH2);
  ledcAttachPin(LED3, PWM_CH3);

  // --- Initialize OLED Display ---
  Wire.begin(21, 22);  // Set custom I2C pins
  if (!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)) {
    Serial.println("OLED not found!");
    while (true);  // Halt program if OLED fails
  }

  // --- Startup message ---
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(15, 25);
  display.println("System Ready...");
  display.display();
  delay(1000);
  display.clearDisplay();
}

/*
  drawUIFrame():
  Draws the OLED layout including header, border,
  and footer with pin information. Displays current mode text.
*/
void drawUIFrame(String modeText) {
  display.clearDisplay();

  // --- Header Bar ---
  display.fillRect(0, 0, 128, 12, SSD1306_WHITE);     // top header bar
  display.setTextColor(SSD1306_BLACK);
  display.setCursor(3, 2);
  display.print("ESP32 LED Controller");
```

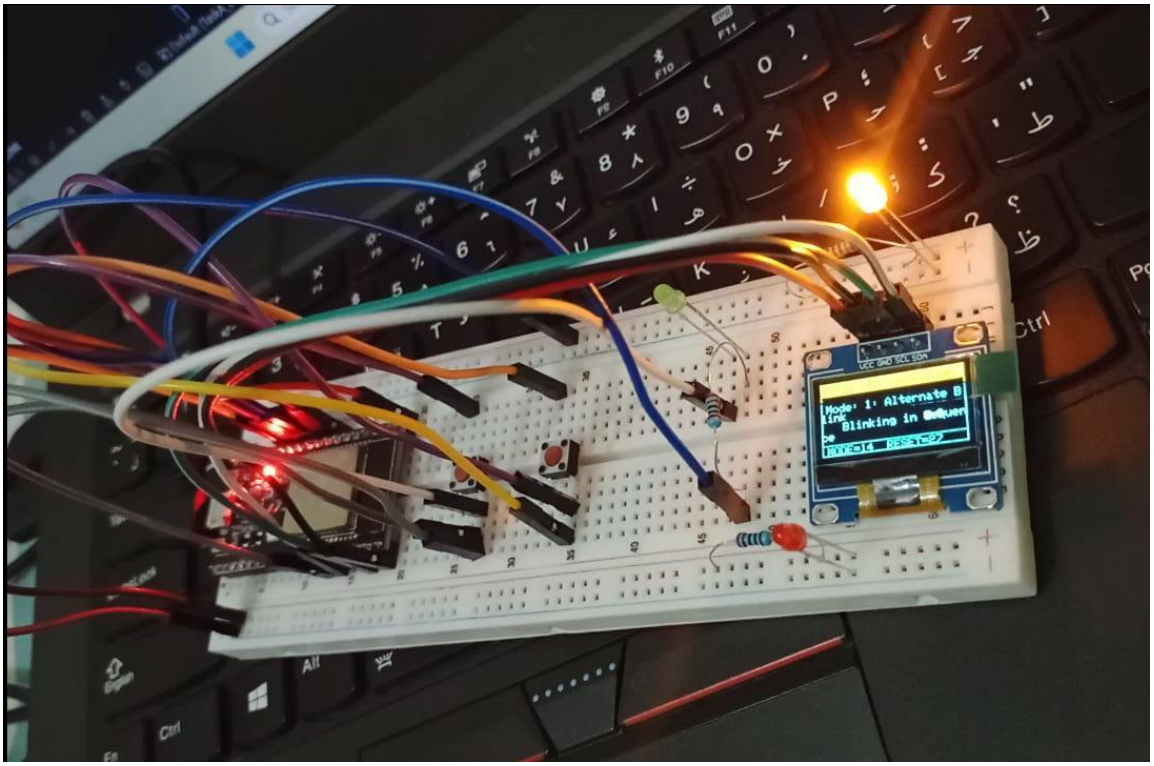
```
void drawUIFrame(String modeText) {
  // --- Outer Border ---
  display.drawRect(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT, SSD1306_WHITE);

  // --- Mode Title ---
  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(1);
  display.setCursor(5, 18);
  display.print("Mode: ");
  display.println(modeText);

  // --- Footer Bar (Instructions) ---
  display.drawLine(0, 54, 128, 54, SSD1306_WHITE);
  display.setCursor(5, 56);
  display.print("MODE=14  RESET=27");
}

void loop() {
  // --- Button Handling (with debounce delay) ---
  if (!digitalRead(BTN_MODE) && millis() - lastPress > 250) {
    mode = (mode + 1) % 4;  // Cycle through 4 modes
    lastPress = millis();
  }
  if (!digitalRead(BTN_RESET) && millis() - lastPress > 250) {
    mode = 0;                // Reset to OFF mode
    lastPress = millis();
  }

  // --- Mode Logic + OLED Display ---
  switch (mode) {

    // MODE 0: All LEDs OFF
    case 0: {
      drawUIFrame("0: ALL OFF");
      ledcWrite(PWM_CH1, 0);
      ledcWrite(PWM_CH2, 0);
      ledcWrite(PWM_CH3, 0);
      display.setCursor(20, 35);
      display.println("All LEDs OFF");
      break;
    }
```

Hardware Output:

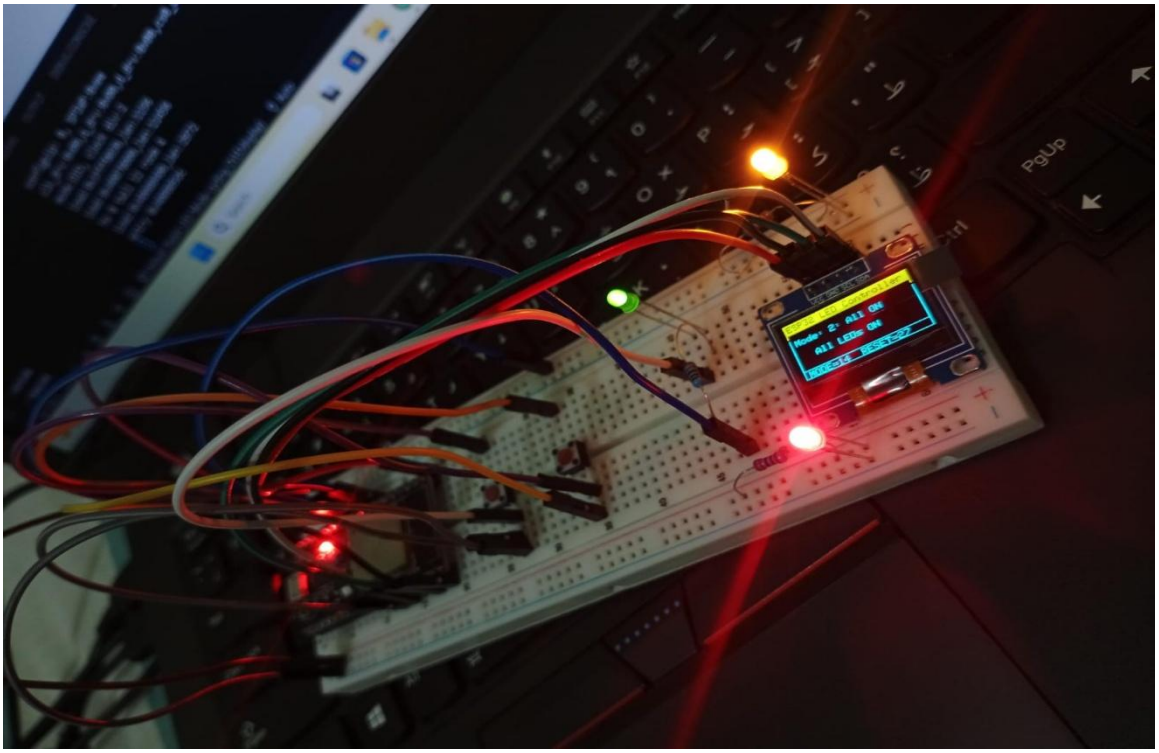
Mode 0:



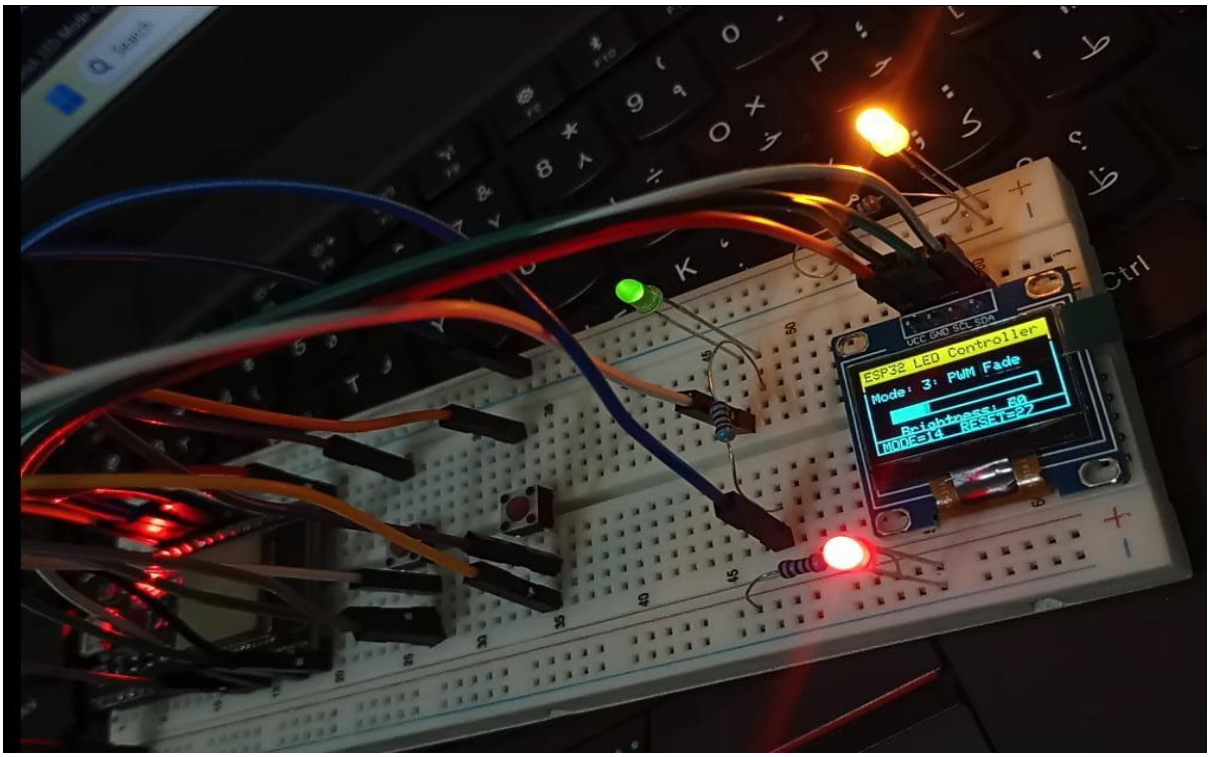
Mode 1:



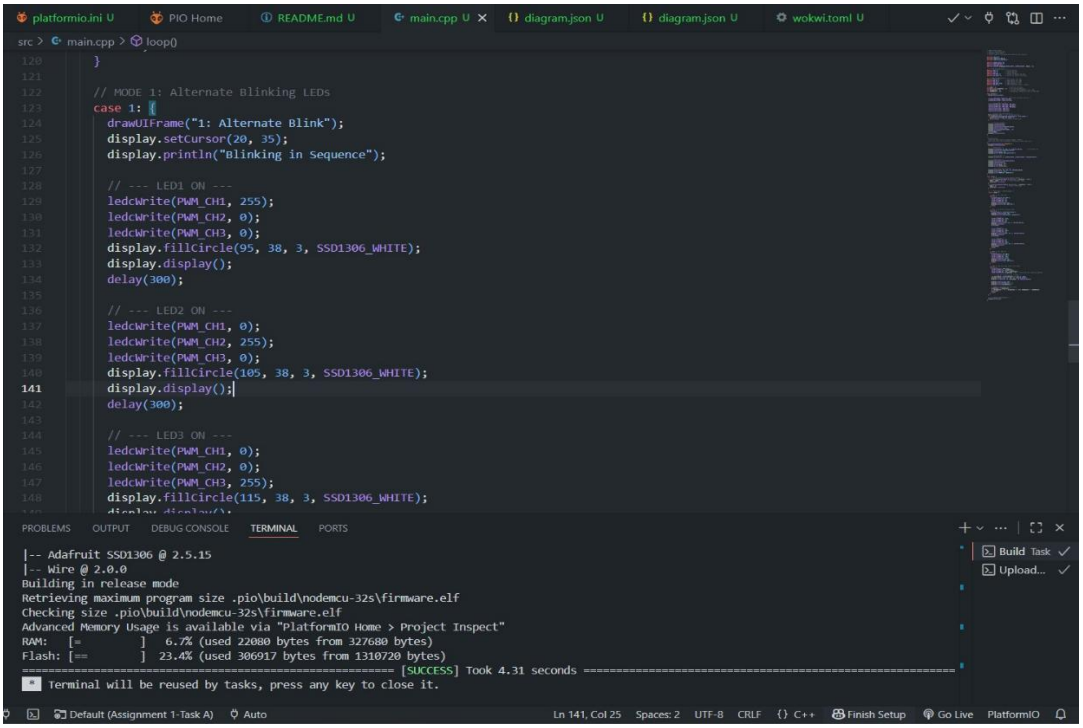
Mode 2:



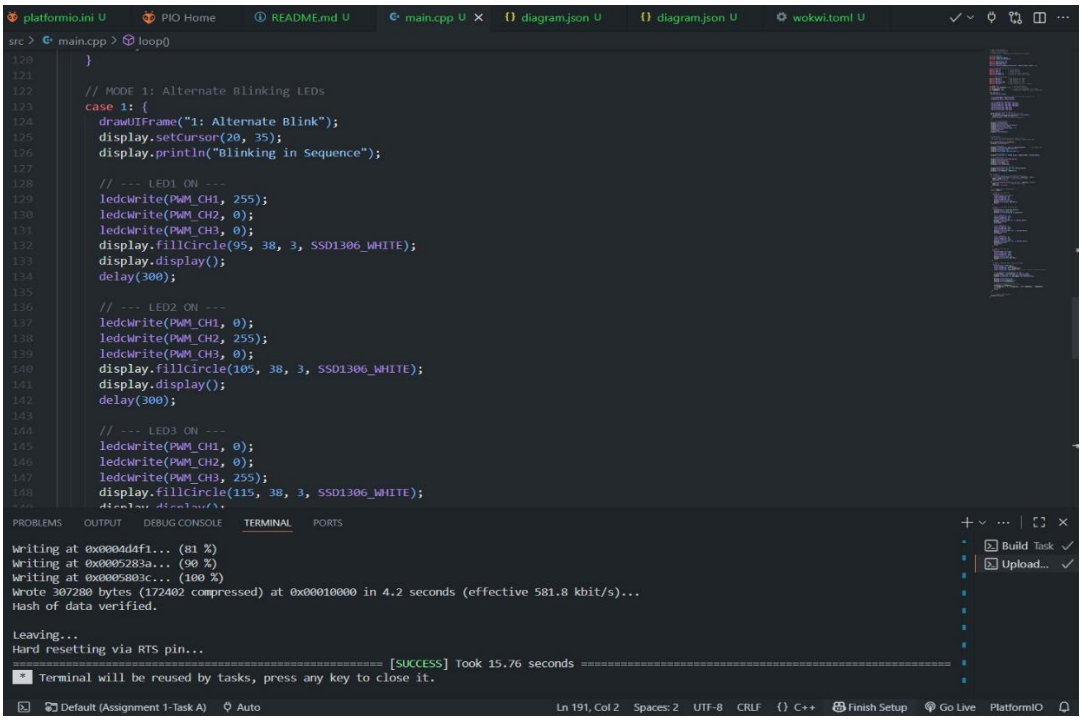
Mode 3:



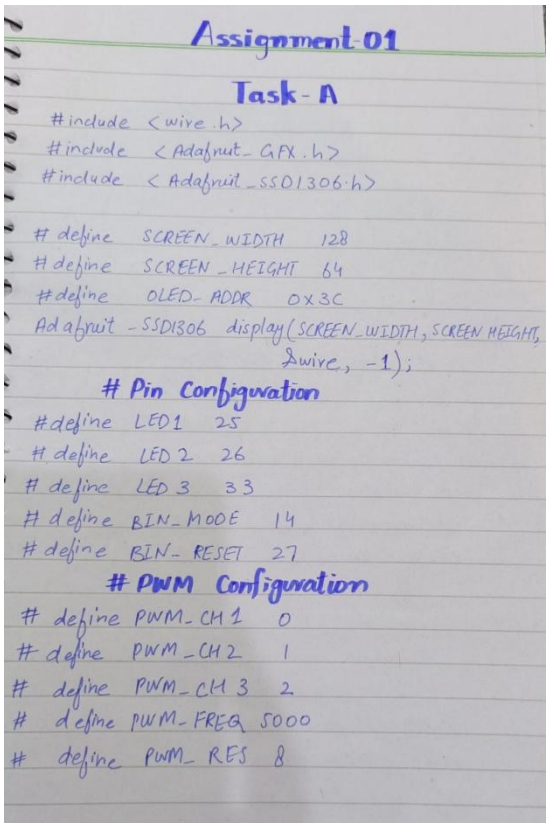
Build Success:

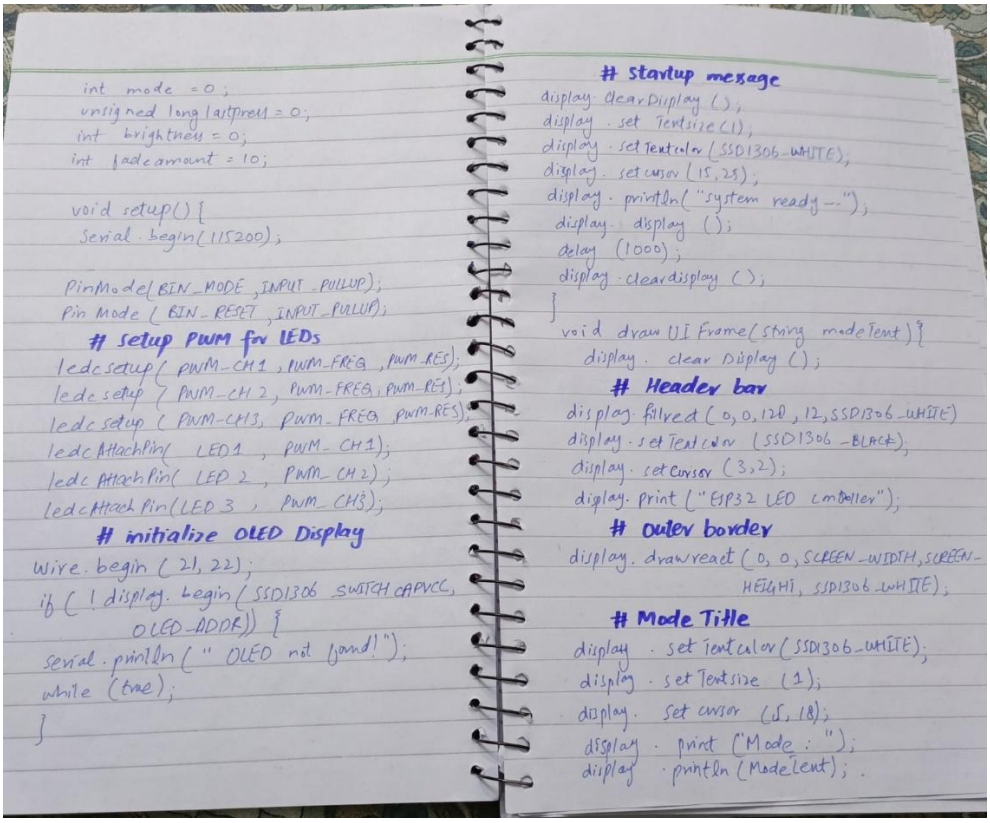


Upload success:



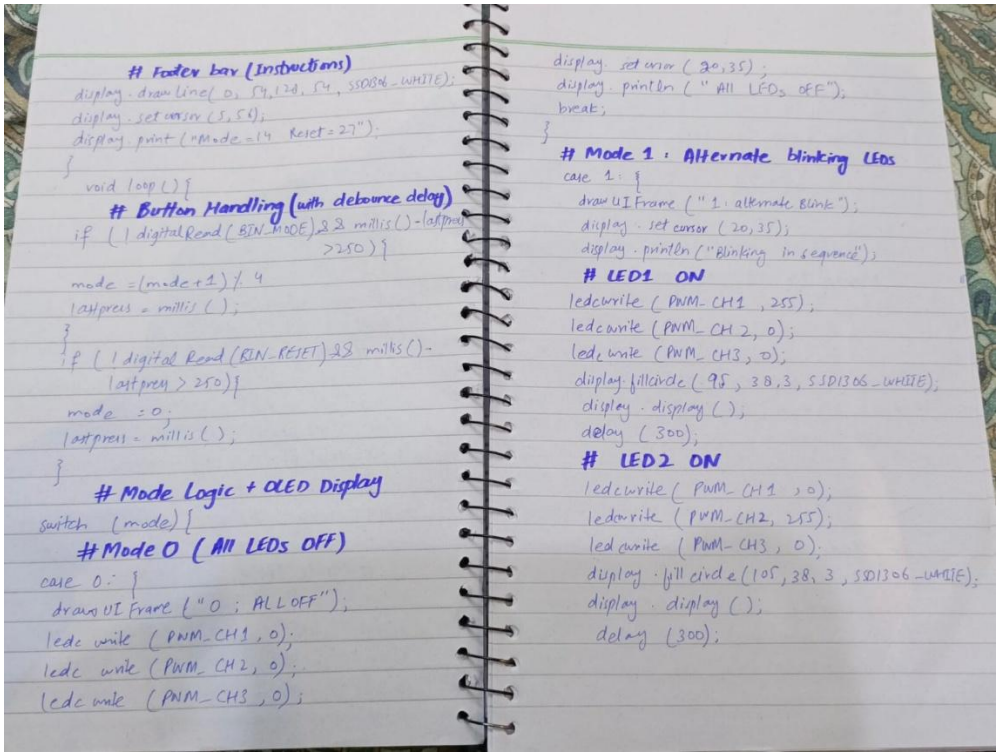
Handwritten Code:





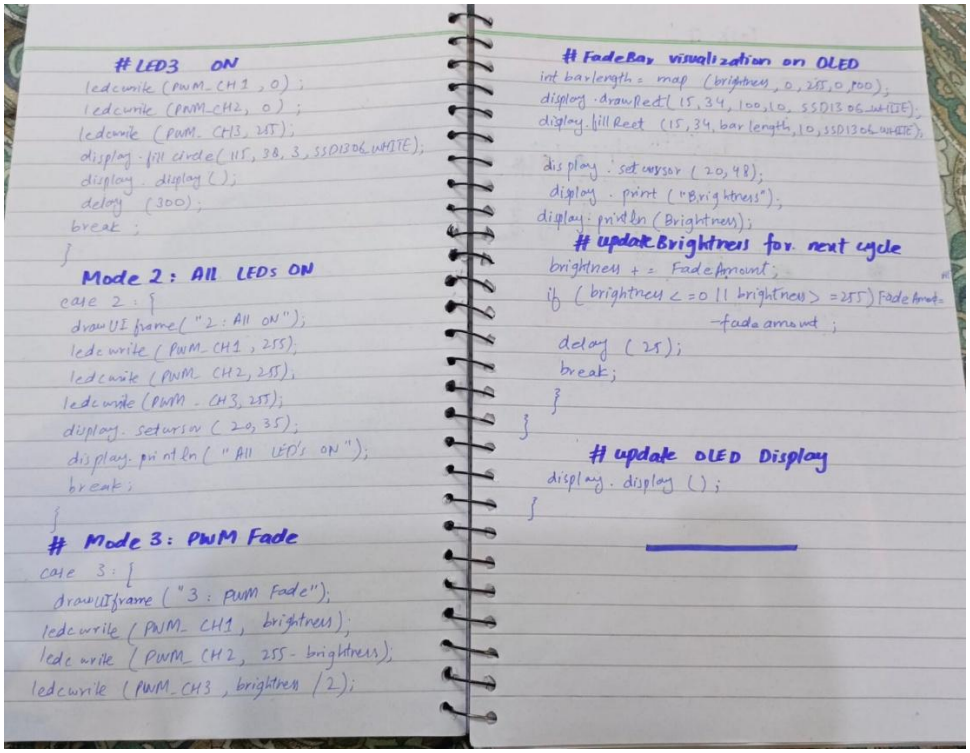
```
int mode = 0;
unsigned long lastPress = 0;
int brightness = 0;
int fadeAmount = 10;

void setup() {
Serial.begin(115200);

PinMode(BIN_MODE, INPUT_PULLUP);
Pin Mode (BIN_RESET, INPUT_PULLUP);
  # setup PWM for LEDs
ledcSetup(PWM_CH1, PWM_FREQ, PWM_RES);
ledcSetup(PWM_CH2, PWM_FREQ, PWM_RES);
ledcSetup(PWM_CH3, PWM_FREQ, PWM_RES);
ledcAttachPin(LED1, PWM_CH1);
ledcAttachPin(LED2, PWM_CH2);
ledcAttachPin(LED3, PWM_CH3);
    # initialize OLED Display
Wire.begin(21, 22);
if (!display.begin(SSD1306_SWITCHCAPVCC,
        OLED_ADDR)) {
serial.println("OLED not found!");
while (true);
}
```

```
  # Startup message
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(15,25);
display.println("system ready...");
display.display();
delay(1000);
display.clearDisplay();
}
void drawUIFrame(String modeText){
display.clearDisplay();
    # Header bar
display.fillRect(0,0,128,12,SSD1306_WHITE)
display.setTextColor(SSD1306_BLACK);
display.setCursor(3,2);
display.print("ESP32 LED Controller");
    # Outer border
display.drawRect(0,0,SCREEN_WIDTH,SCREEN-
                HEIGHT, SSD1306_WHITE);
    # Mode Title
display.setTextColor(SSD1306_WHITE);
display.setTextSize(1);
display.setCursor(5,18);
display.print("Mode: ");
display.println(modeText);
```

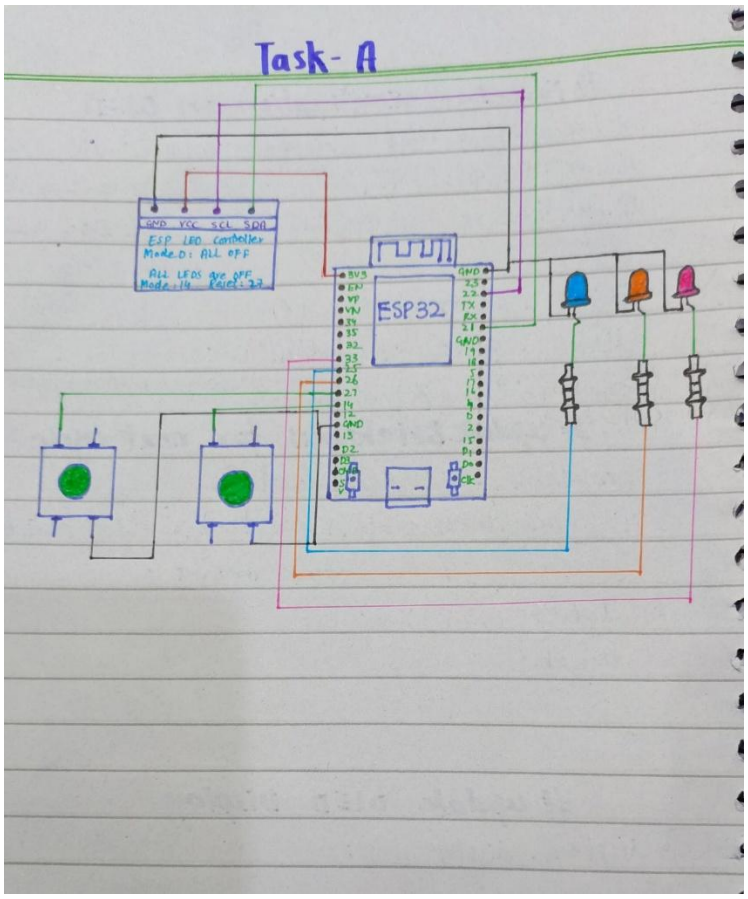


```
    # Footer bar (Instructions)
display.drawLine(0,54,128,54,SSD1306_WHITE);
display.setCursor(5,56);
display.print("Mode=14 Reset=27");
}
void loop(){
    # Button Handling (with debounce delay)
if (!digitalRead(BIN_MODE)&& millis()-lastPress
                >250){

mode = (mode+1) % 4
lastPress = millis();
}
if (!digitalRead(BIN_RESET)&& millis()-
    lastPress > 250){
mode = 0;
lastPress = millis();
}
    # Mode Logic + OLED Display
switch (mode){
    # Mode 0 (All LEDs OFF)
case 0: {
drawUIFrame("0: ALL OFF");
ledcWrite(PWM_CH1, 0);
ledcWrite(PWM_CH2, 0);
ledcWrite(PWM_CH3, 0);
```

```
display.setCursor(20,35);
display.println("All LEDs OFF");
break;
}
    # Mode 1: Alternate blinking LEDs
case 1: {
drawUIFrame("1: alternate Blink");
display.setCursor(20,35);
display.println("Blinking in sequence");
    # LED1 ON
ledcWrite(PWM_CH1, 255);
ledcWrite(PWM_CH2, 0);
ledcWrite(PWM_CH3, 0);
display.fillCircle(95, 38, 3, SSD1306_WHITE);
display.display();
delay(300);
    # LED2 ON
ledcWrite(PWM_CH1, 0);
ledcWrite(PWM_CH2, 255);
ledcWrite(PWM_CH3, 0);
display.fillCircle(105, 38, 3, SSD1306_WHITE);
display.display();
delay(300);
```



```
# LED3 ON
ledcwrite (PWM_CH1, 0);
ledcwrite (PWM_CH2, 0);
ledcwrite (PWM_CH3, 255);
display.fillCircle(115, 38, 3, SSD1306_WHITE);
display.display();
delay (300);
break;
}

Mode 2: All LEDs ON
case 2: {
drawUIframe("2: All ON");
ledcwrite (PWM_CH1, 255);
ledcwrite (PWM_CH2, 255);
ledcwrite (PWM_CH3, 255);
display.setCursor (20, 35);
display.println ("All LED's ON");
break;
}

# Mode 3: PWM Fade
case 3: {
drawUIframe ("3: PWM Fade");
ledcwrite (PWM_CH1, brightness);
ledcwrite (PWM_CH2, 255 - brightness);
ledcwrite (PWM_CH3, brightness / 2);

# FadeBar visualization on OLED
int barLength = map (brightness, 0, 255, 0, 100);
display.drawRect(15, 34, 100, 10, SSD1306_WHITE);
display.fillRect (15, 34, barLength, 10, SSD1306_WHITE);

display.setCursor (20, 48);
display.print ("Brightness");
display.println (Brightness);
# update Brightness for next cycle
brightness += FadeAmount;
if (brightness <= 0 || brightness >= 255) FadeAmount
    = -fadeAmount;
delay (25);
break;
}
}
# update OLED Display
display.display ();
}
```



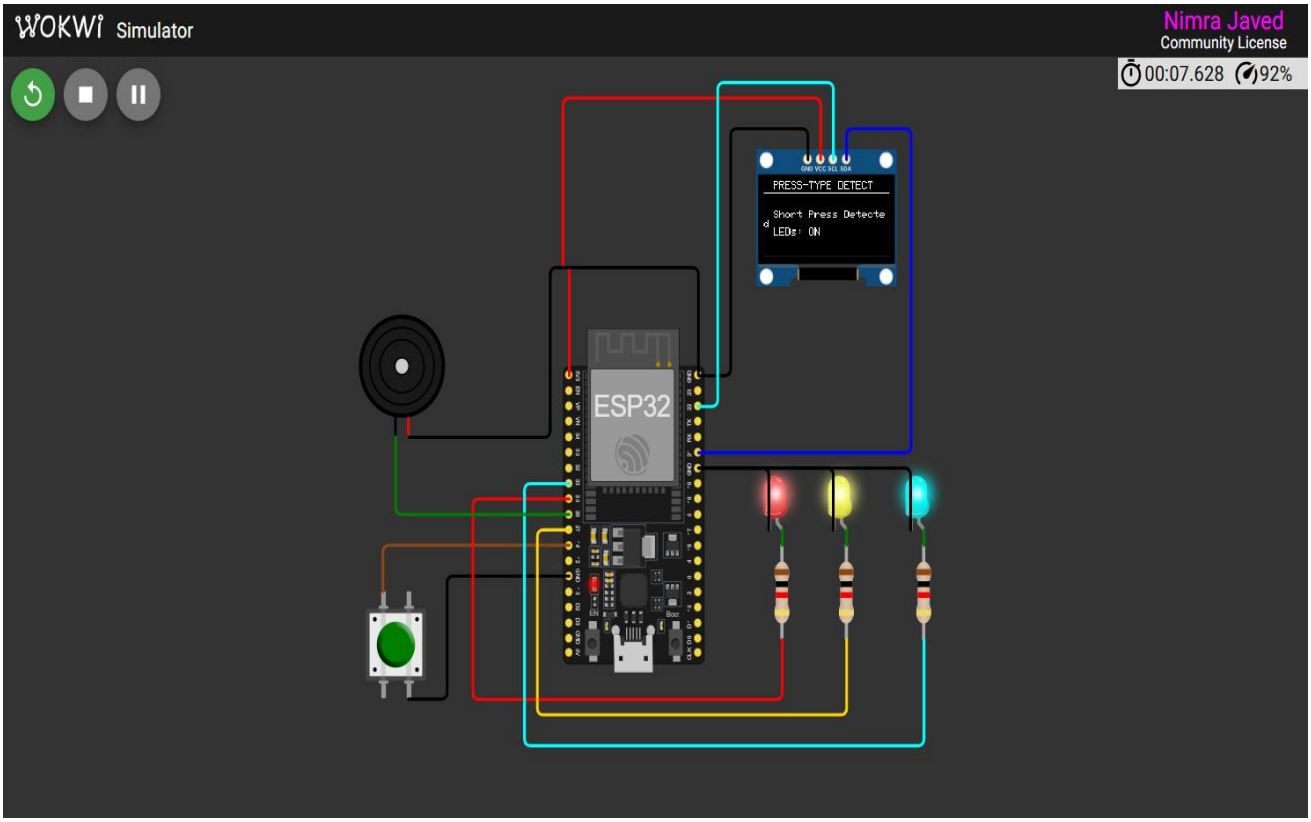
Wokwi Link (Task-A):

https://wokwi.com/projects/445507758525483009

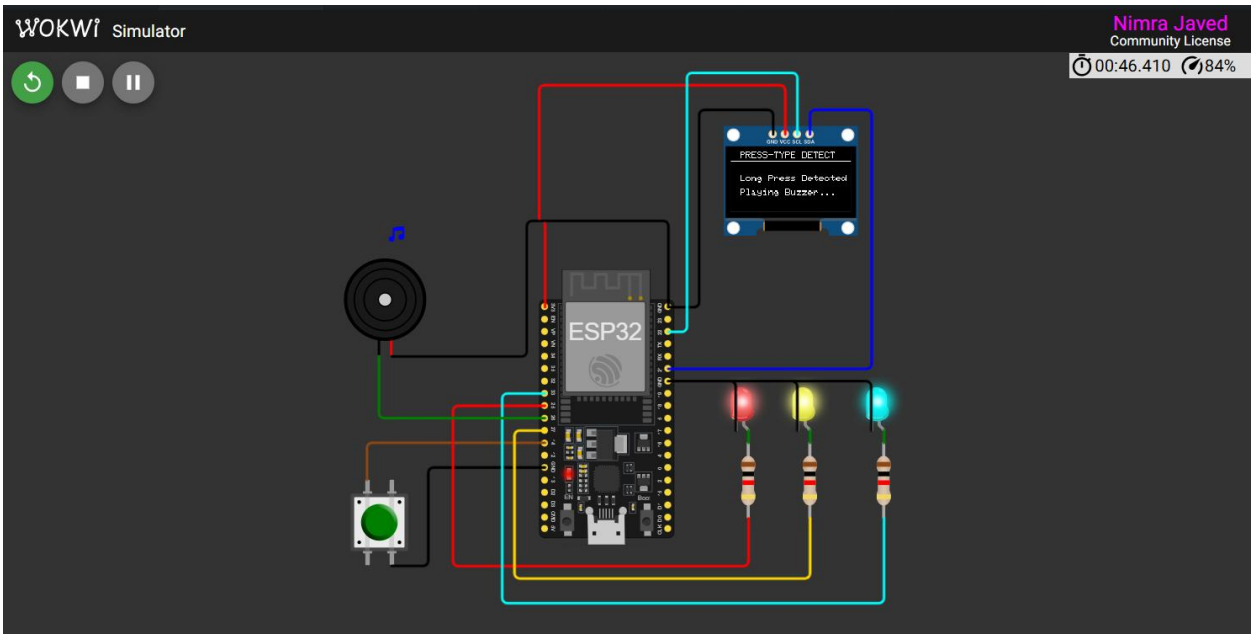
TASK B

Wokwi diagram:

Short press Detected: (LEDS ON)



Long press Detected: (Buzzer)



Pin diagram:

ESP32 Pin	Label	Connected To	Description
25	LED1	LED1 (via 220Ω resistor)	First LED output
27	LED2	LED2 (via 220Ω resistor)	Second LED output
33	LED3	LED3 (via 220Ω resistor)	Third LED output
26	BUZZER_PIN	Buzzer (active buzzer)	Activates buzzer on long press
14	BUTTON_PIN	Push Button → GND	Detects short or long button press
21	SDA	OLED SDA	I²C data line for OLED communication
22	SCL	OLED SCL	I²C clock line for OLED communication
3.3V	VCC	OLED VCC	Power supply for OLED (3.3V)
GND	GND	OLED, LEDs, Buzzer, Button	Common ground connection

Code:

diagram.json U | Wokwi Simulator | diagram.json U | PIO Home | main.cpp U | platformio.ini U | wokwi.toml U

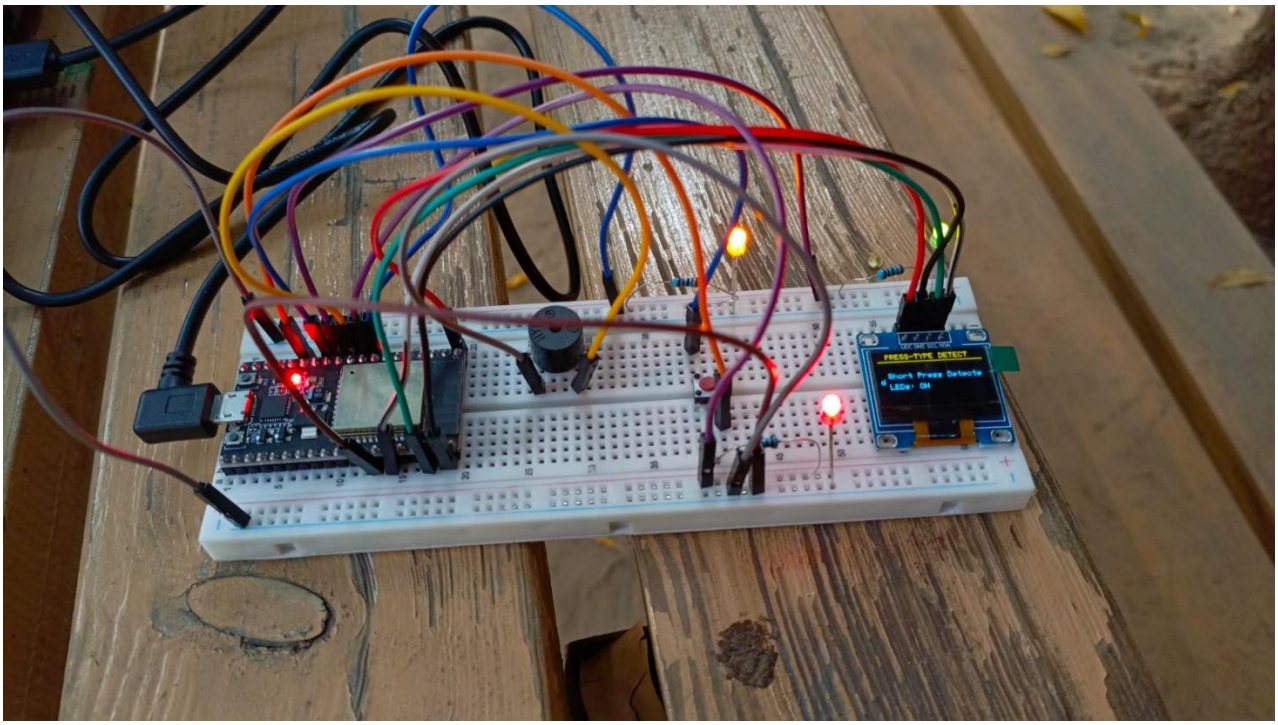
src > main.cpp > ...

```cpp
#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
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// --- Pin Configuration ---
#define LED1 25
#define LED2 27
#define LED3 33
#define BUZZER_PIN 26
#define BUTTON_PIN 14

// --- Variables ---
bool ledState = false;              // Stores ON/OFF state for all LEDs
unsigned long pressStart = 0;       // Time when button was pressed
bool isPressed = false;             // Flag to track button press
const unsigned long longPressTime = 1500;  // 1.5s threshold for long press

// --- Function to display header and message on OLED ---
void showOnOLED(const String &line1, const String &line2 = "") {
  display.clearDisplay();

  // Header area
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(10, 0);
  display.println("PRESS-TYPE DETECT");
  display.drawLine(0, 10, 127, 10, SSD1306_WHITE); // underline

  // Main message
  display.setCursor(10, 25);
  display.println(line1);
  if (line2 != "") {
    display.setCursor(10, 40);
    display.println(line2);
  }
  display.display();
}

void setup() {
  Serial.begin(115200);

  // --- Pin Modes ---
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);  // Button active LOW

  // --- Initialize OLED ---
  Wire.begin(21, 22);
  if (!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)) {
    Serial.println("OLED not found!");
    while (true);
  }

  // --- Startup Screen ---
  showOnOLED("System Ready...", "Press Button");
  delay(1500);

  // Turn off LEDs and Buzzer initially
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(BUZZER_PIN, LOW);
}

void loop() {
  int buttonState = digitalRead(BUTTON_PIN);

  // --- Detect button press start ---
  if (buttonState == LOW && !isPressed) {
    isPressed = true;
    pressStart = millis();
  }
```

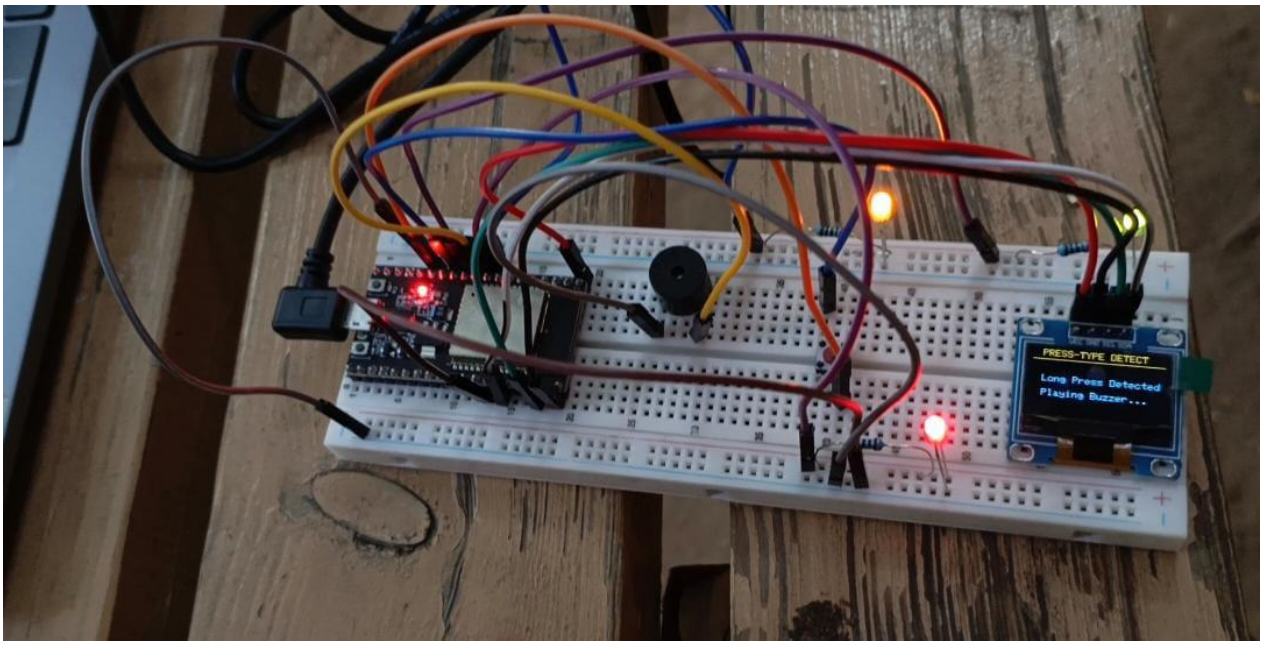
```
  // --- Detect button release ---
  if (buttonState == HIGH && isPressed) {
    unsigned long pressDuration = millis() - pressStart;
    isPressed = false;

    if (pressDuration < longPressTime) {
      // --- SHORT PRESS ACTION: Toggle LEDs ---
      ledState = !ledState;
      digitalWrite(LED1, ledState);
      digitalWrite(LED2, ledState);
      digitalWrite(LED3, ledState);

      showOnOLED("Short Press Detected", ledState ? "LEDs: ON" : "LEDs: OFF");
      Serial.println(ledState ? "LEDs ON" : "LEDs OFF");

    } else {
      // --- LONG PRESS ACTION: Buzzer Tone ---
      showOnOLED("Long Press Detected", "Playing Buzzer...");
      tone(BUZZER_PIN, 1000, 500);  // 1kHz for 0.5s
      Serial.println("Buzzer Tone Played");
    }

    delay(300); // Small delay to avoid flicker
  }
}
```

Hardware Output:

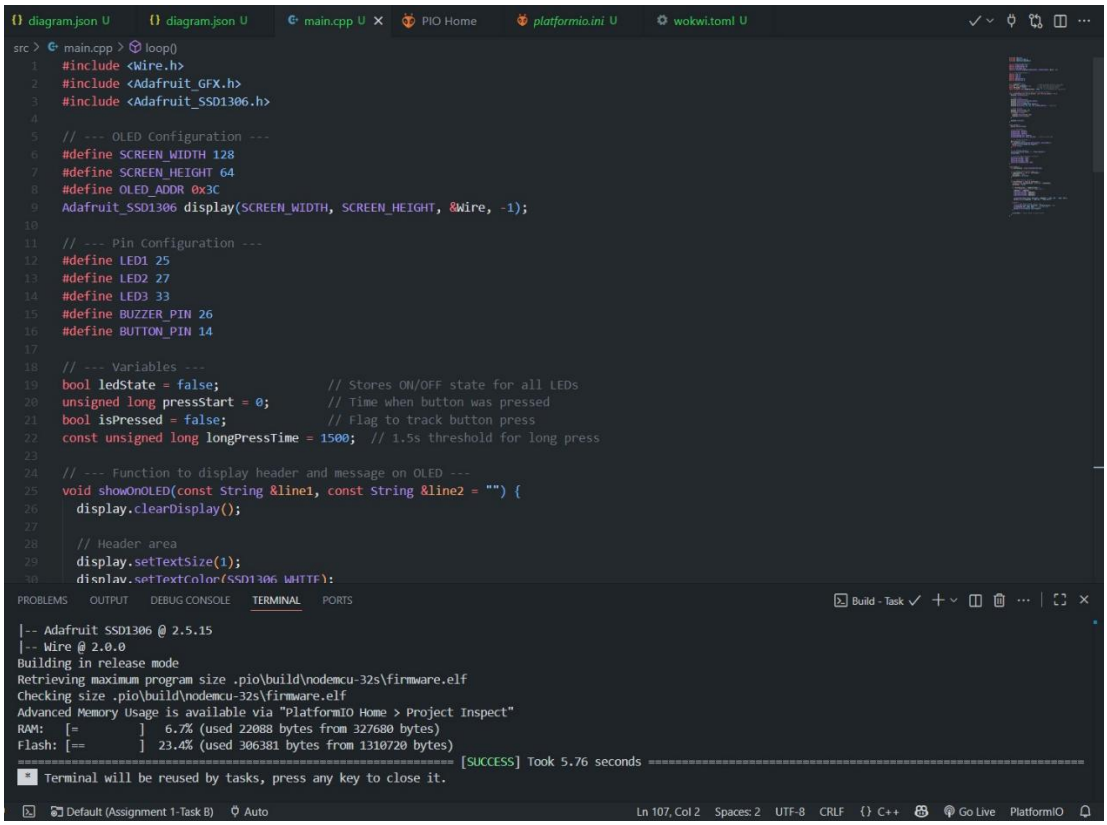
Short press detected:



Long press detected: (buzzer)



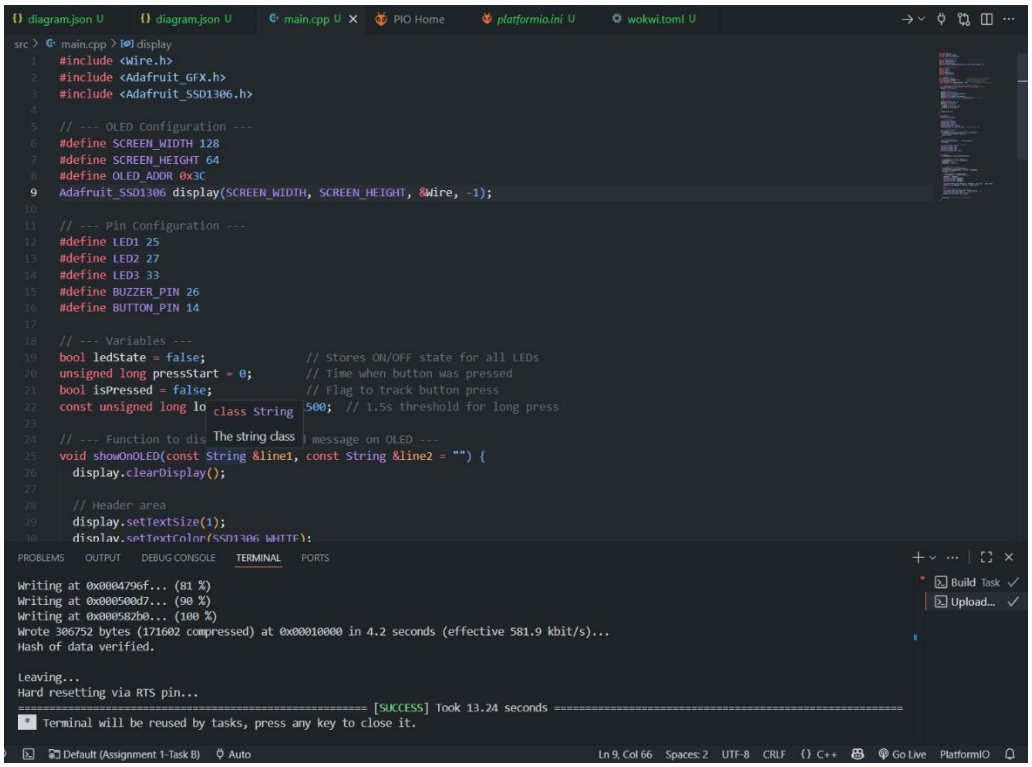
Build Success:



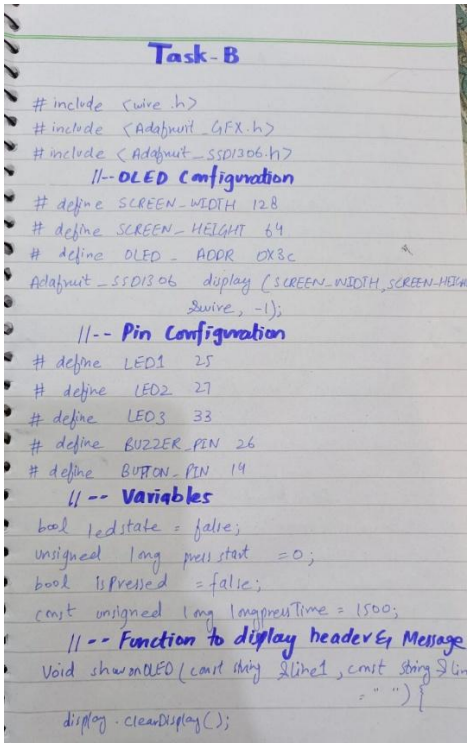
```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// --- OLED Configuration ---
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_ADDR 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// --- Pin Configuration ---
#define LED1 25
#define LED2 27
#define LED3 33
#define BUZZER_PIN 26
#define BUTTON_PIN 14

// --- Variables ---
bool ledState = false;              // Stores ON/OFF state for all LEDs
unsigned long pressStart = 0;       // Time when button was pressed
bool isPressed = false;             // Flag to track button press
const unsigned long longPressTime = 1500;  // 1.5s threshold for long press

// --- Function to display header and message on OLED ---
void showOnOLED(const String &line1, const String &line2 = "") {
  display.clearDisplay();

  // Header area
  display.setTextSize(1);
```

```
|-- Adafruit SSD1306 @ 2.5.15
|-- Wire @ 2.0.0
Building in release mode
Retrieving maximum program size .pio\build\nodemcu-32s\firmware.elf
Checking size .pio\build\nodemcu-32s\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [=         ]   6.7% (used 22088 bytes from 327680 bytes)
Flash: [==        ]  23.4% (used 306381 bytes from 1310720 bytes)
========== [SUCCESS] Took 5.76 seconds ==========
Terminal will be reused by tasks, press any key to close it.
```

Upload Success:



Handwritten Code:



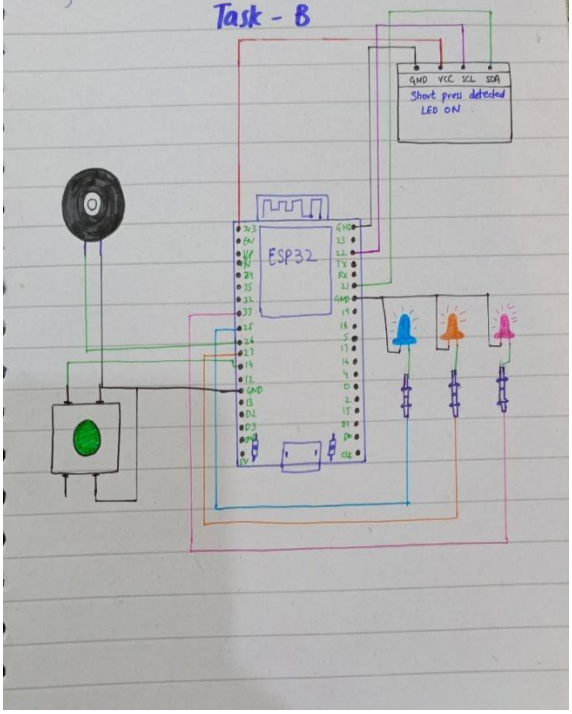
```
// Header Area
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(10,0);
display.println("PRESS TYPE DETECT");
display.drawLine(0, 10, 127, 10, SSD1306_WHITE);
// Main Message
display.setCursor(10,25);
display.println(Line1);
if (Line2 != "") {
  display.setCursor(10, 40);
  display.println(line2);
}
display.display();
}
void setup(){
  Serial.begin(115200);
//-- Pin Modes
PinMode(LED1, OUTPUT);
PinMode(LED2, OUTPUT);
PinMode(LED3, OUTPUT);
PinMode(Buzzer_PIN, OUTPUT);
PinMode(BUTTON_PIN, INPUT_PULLUP);
```

```
//-- Initialize OLED
wire.begin(21,22);
if (!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)
{
Serial.println("OLED not found!");
while (true);
}

showONLED("System Ready...", "Press button");
delay(1500);
// Turn off LED and Buzzer initially
digitalWrite(LED1,LOW);
digitalWrite(LED2,Low);
digitalWrite(LED3, Low);
digitalWrite(BUZZER_PIN, LOW);
}
void loop(){
  int buttonState = digitalRead(BUTTON_PIN);
  //-- Detect Button press start
if (buttonState == Low && !isPressed){
  isPressed = true;
  PressStart = millis();
}
```

```
//-- Detect ButtonRelease

if (buttonstate == HIGH && isPressed){
  unsigned long pressDuration = millis()-pressstart;
  isPressed = false;

if (PressDuration < LongPressTime){
  //-- Short Press action : Toogle LEDs
  ledstate = !ledstate;
  digitalwrite(LED1, ledstate);
  digitalwrite(LED2, ledstate);
  digitalwrite(LED3, ledstate);

  showonOLED("Short PressDetected", ledstate?
        "LEDs: ON": "LEDs : OFF");
  Serial.println(Led ? "LEDs ON":"LED OFF");
}
  else {
  //-- Long Press action: Buzzer Tone --
  ShowONOLED("Long Press detected", "Playing
      buzzer...");
  tone(BUZZER_PIN, 1000, 500);
  Serial.println("Buzzer Tone played");
  }
```

```
delay(300);
}
}
```

Task - B



Wokwi Link (TaskB):

https://wokwi.com/projects/445508998287505409