

National Textile University, Faisalabad



Department of Computer Science

Name:	MUHAMMAD AHMAD SATTAR
Class:	BSCS 5 th B
Registration No:	23-NTU-CS-1059
Assignment	Assignment: 1
Course Name:	Embedded IoT Systems (CSE-3080)
Submitted To:	Sir Nasir Mahmood
Submission Date:	25 - OCT - 2025

ASSIGNMENT 1

QUESTION 3

TASK -1

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

CODE:

```
/*
Name: Ahmad Sattar
REG #: 23-NTU-CS-1059
TASK-1: LED MODE CONTROLLER WITH OLED
*/

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

// --- Screen & Display Settings ---
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// --- Hardware Pin Assignments ---
#define LED_YELLOW_PIN 16
#define LED_BLUE_PIN 18
#define LED_CYAN_PIN 17
#define MODE_BTN_PIN 33
#define RESET_BTN_PIN 27

// --- LEDC PWM Configuration ---
#define PWM_CHAN_YELLOW 0
#define PWM_CHAN_BLUE 1
#define PWM_CHAN_CYAN 2
#define PWM_FREQ 4000
```

```

#define PWM_RESOLUTION 8

// --- Global Variables ---
hw_timer_t *blinkTimer = nullptr;
int currentMode = 0;
int blinkStep = 0;
bool prevModeBtnState = HIGH;
bool prevResetBtnState = HIGH;
unsigned long lastPressTime = 0;
const int debounceDelay = 600;
volatile unsigned long timerTick = 0;

// --- Interrupt Service Routine (ISR) for Timer ---
void IRAM_ATTR onTimerInterrupt() {
    timerTick++;
}

// --- Function to Update the OLED Display ---
void updateDisplay() {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(15, 0);
    display.println("LED PANEL");
    display.drawLine(0, 20, 127, 20, SSD1306_WHITE);
    display.setTextSize(1);
    display.setCursor(10, 35);

    if (currentMode == 0) display.print("Mode: OFF");
    else if (currentMode == 1) display.print("Mode: Blink");
    else if (currentMode == 2) display.print("Mode: ON");
    else if (currentMode == 3) display.print("Mode: PWM Fade");

    display.display();
}

// --- Setup Function ---
void setup() {
    Serial.begin(115200);

    pinMode(LED_YELLOW_PIN, OUTPUT);
    pinMode(LED_BLUE_PIN, OUTPUT);
    pinMode(LED_CYAN_PIN, OUTPUT);
    pinMode(MODE_BTN_PIN, INPUT_PULLUP);
    pinMode(RESET_BTN_PIN, INPUT_PULLUP);
}

```

```

if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    while (true);
}

ledcSetup(PWM_CHAN_YELLOW, PWM_FREQ, PWM_RESOLUTION);
ledcSetup(PWM_CHAN_BLUE, PWM_FREQ, PWM_RESOLUTION);
ledcSetup(PWM_CHAN_CYAN, PWM_FREQ, PWM_RESOLUTION);

ledcAttachPin(LED_YELLOW_PIN, PWM_CHAN_YELLOW);
ledcAttachPin(LED_BLUE_PIN, PWM_CHAN_BLUE);
ledcAttachPin(LED_CYAN_PIN, PWM_CHAN_CYAN);

blinkTimer = timerBegin(0, 80, true);
timerAttachInterrupt(blinkTimer, &onTimerInterrupt, true);
timerAlarmWrite(blinkTimer, 1000000, true);
timerAlarmEnable(blinkTimer);

ledcWrite(PWM_CHAN_YELLOW, 0);
ledcWrite(PWM_CHAN_BLUE, 0);
ledcWrite(PWM_CHAN_CYAN, 0);
updateDisplay();
}

// --- Main Loop ---
void loop() {
    bool modeBtnState = digitalRead(MODE_BTN_PIN);
    bool resetBtnState = digitalRead(RESET_BTN_PIN);

    if (millis() - lastPressTime > debounceDelay) {
        if (modeBtnState == LOW && prevModeBtnState == HIGH) {
            currentMode = (currentMode + 1) % 4;
            blinkStep = 0;
            updateDisplay();
            lastPressTime = millis();
        }
        if (resetBtnState == LOW && prevResetBtnState == HIGH) {
            currentMode = 0;
            blinkStep = 0;
            updateDisplay();
            lastPressTime = millis();
        }
    }
    prevModeBtnState = modeBtnState;
    prevResetBtnState = resetBtnState;
}

```

```

if (currentMode == 0) {
    ledcWrite(PWM_CHAN_YELLOW, 0);
    ledcWrite(PWM_CHAN_BLUE, 0);
    ledcWrite(PWM_CHAN_CYAN, 0);
}
else if (currentMode == 1) {
    static unsigned long lastTickHandled = 0;
    if (timerTick != lastTickHandled) {
        lastTickHandled = timerTick;
        blinkStep = (blinkStep + 1) % 3;
        if (blinkStep == 0) {
            ledcWrite(PWM_CHAN_YELLOW, 255);
            ledcWrite(PWM_CHAN_BLUE, 0);
            ledcWrite(PWM_CHAN_CYAN, 0);
        } else if (blinkStep == 1) {
            ledcWrite(PWM_CHAN_YELLOW, 0);
            ledcWrite(PWM_CHAN_BLUE, 255);
            ledcWrite(PWM_CHAN_CYAN, 0);
        } else {
            ledcWrite(PWM_CHAN_YELLOW, 0);
            ledcWrite(PWM_CHAN_BLUE, 0);
            ledcWrite(PWM_CHAN_CYAN, 255);
        }
    }
}
else if (currentMode == 2) {
    ledcWrite(PWM_CHAN_YELLOW, 255);
    ledcWrite(PWM_CHAN_BLUE, 255);
    ledcWrite(PWM_CHAN_CYAN, 255);
}
else if (currentMode == 3) {
    for (int dutyCycle = 0; dutyCycle <= 255 && currentMode == 3; dutyCycle++) {
        ledcWrite(PWM_CHAN_YELLOW, dutyCycle);
        ledcWrite(PWM_CHAN_BLUE, dutyCycle);
        ledcWrite(PWM_CHAN_CYAN, dutyCycle);
        delay(5);
        if (digitalRead(MODE_BTN_PIN) == LOW || digitalRead(RESET_BTN_PIN) == LOW)
            return;
    }
    for (int dutyCycle = 255; dutyCycle >= 0 && currentMode == 3; dutyCycle--) {
        ledcWrite(PWM_CHAN_YELLOW, dutyCycle);
        ledcWrite(PWM_CHAN_BLUE, dutyCycle);
        ledcWrite(PWM_CHAN_CYAN, dutyCycle);
        delay(5);
    }
}

```

```

    if (digitalRead(MODE_BTN_PIN) == LOW || digitalRead(RESET_BTN_PIN) == LOW)
return;
    }
}

}

```

BUILD:

```

src > main.cpp > ...
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <Adafruit_GFX.h>
4 #include <Adafruit_SSD1306.h>
5
6 #define W 128
7 #define H 64
8 Adafruit_SSD1306 scr(W, H, &Wire);
9
10 #define LED_Y 4
11 #define LED_G 0
12 #define LED_R 2
13 #define BTN_MODE 26
14 #define BTN_RST 27
15
16 #define CH_Y 0
17 #define CH_G 1
18 #define CH_R 2
19 #define FREQ 4000
20 #define RES 8
21
22 hw_timer_t *blinkT = null;
23
24 int modeSel = 0;
25 int blinkStep = 0;
26 bool oldMode = HIGH;
27 bool oldRst = HIGH;
28 unsigned long tPrev = 0;
29 const int tDelay = 600;
30 volatile unsigned long ti;
31
32 void IRAM_ATTR timerTick(
33 | tick++;
34 | }
35
36 void showScreen() {
37 | scr.clearDisplay();
38

```

```

Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-cpu.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-dac.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-gpio.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-i2c-slave.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-i2c.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-ledc.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-matrix.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-misc.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-psram.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-rgb-led.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-rmt.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-signaldelta.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-spi.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-time.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-timer.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-tinyusb.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-touch.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\esp32-hal-uart.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\Firmware_msc_fat.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\libb64\cdecode.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\libb64\cencode.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\main.cpp.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\stdlib\noniso.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\wiring_pulse.c.o
Compiling .pio\build\nodemcu-32s\Framework\Arduino\wiring_shift.c.o
Archiving .pio\build\nodemcu-32s\lib\Framework\Arduino.a
Indexing .pio\build\nodemcu-32s\lib\Framework\Arduino.a
Linking .pio\build\nodemcu-32s\Firmware.elf
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: [=====] 6.8% (used 22120 bytes from 327680 bytes)
Flash: [=====] 23.7% (used 31153 bytes from 1310720 bytes)
Building .pio\build\nodemcu-32s\Firmware.bin
esptool.py v4.9.0
Creating esp32 image...
Merged 2 ELF sections
Successfully created esp32 image.
===== [SUCCESS] Took 58.02 seconds =====
Terminal will be reused by tasks, press any key to close it.

```

UPLOAD:

```

src > main.cpp > ...
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <Adafruit_GFX.h>
4 #include <Adafruit_SSD1306.h>
5
6 #define W 128
7 #define H 64
8 Adafruit_SSD1306 scr(W, H, &Wire);
9
10 #define LED_Y 4
11 #define LED_G 0
12 #define LED_R 2
13 #define BTN_MODE 26
14 #define BTN_RST 27
15
16 #define CH_Y 0
17 #define CH_G 1
18 #define CH_R 2
19 #define FREQ 4000
20 #define RES 8
21
22 hw_timer_t *blinkT = null;
23
24 int modeSel = 0;
25 int blinkStep = 0;
26 bool oldMode = HIGH;
27 bool oldRst = HIGH;
28 unsigned long tPrev = 0;
29 const int tDelay = 600;
30 volatile unsigned long ti;
31
32 void IRAM_ATTR timerTick(
33 | tick++;
34 | }
35
36 void showScreen() {
37 | scr.clearDisplay();
38

```

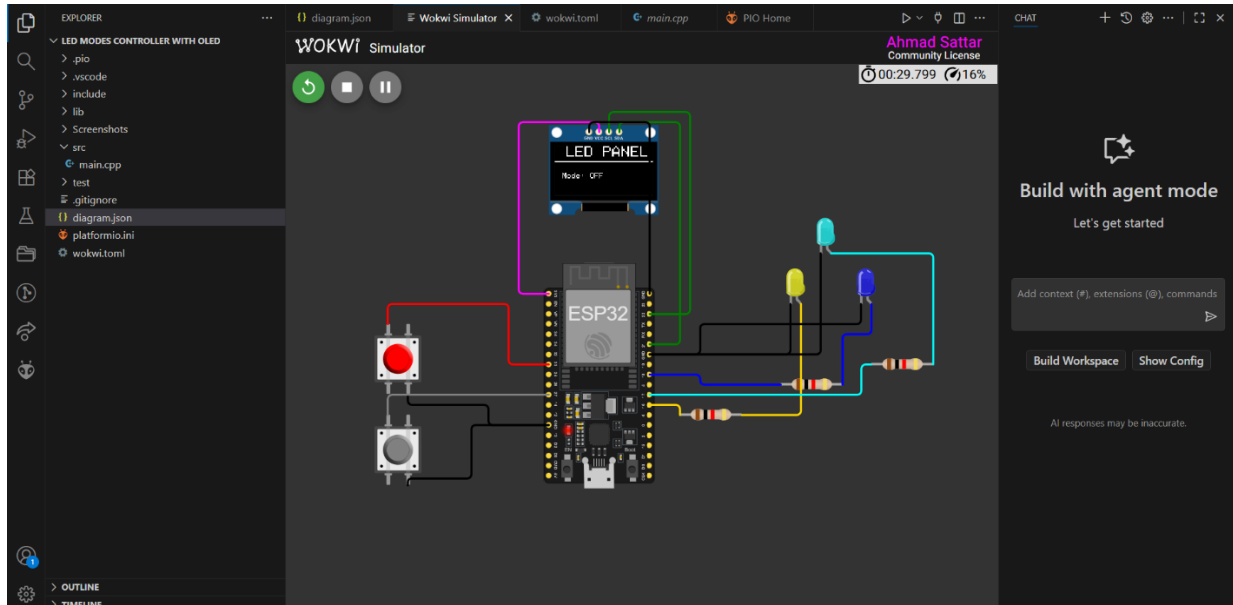
```

Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00005fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Flash will be erased from 0x0000e000 to 0x0000ffff...
Flash will be erased from 0x00010000 to 0x0005cfff...
SHA digest in image updated
Compressed 17536 bytes to 12202...
Writing at 0x00001000... (100 %)
Wrote 17536 bytes (12202 compressed) at 0x00001000 in 0.5 seconds (effective 277.9 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 146...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (146 compressed) at 0x00008000 in 0.0 seconds (effective 631.9 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 864.9 kbit/s)...
Hash of data verified.
Compressed 311520 bytes to 174481...
Writing at 0x00010000... (9 %)
Writing at 0x0001b81b... (18 %)
Writing at 0x000273ac... (27 %)
Writing at 0x0002c970... (36 %)
Writing at 0x00032303... (45 %)
Writing at 0x000370e1... (54 %)
Writing at 0x0003d199... (63 %)
Writing at 0x00042769... (72 %)
Writing at 0x00047b82... (81 %)
Writing at 0x0004f4da... (90 %)
Writing at 0x000585b3... (100 %)
Wrote 311520 bytes (174481 compressed) at 0x00010000 in 4.4 seconds (effective 560.8 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
===== [SUCCESS] Took 22.33 seconds =====
Terminal will be reused by tasks, press any key to close it.

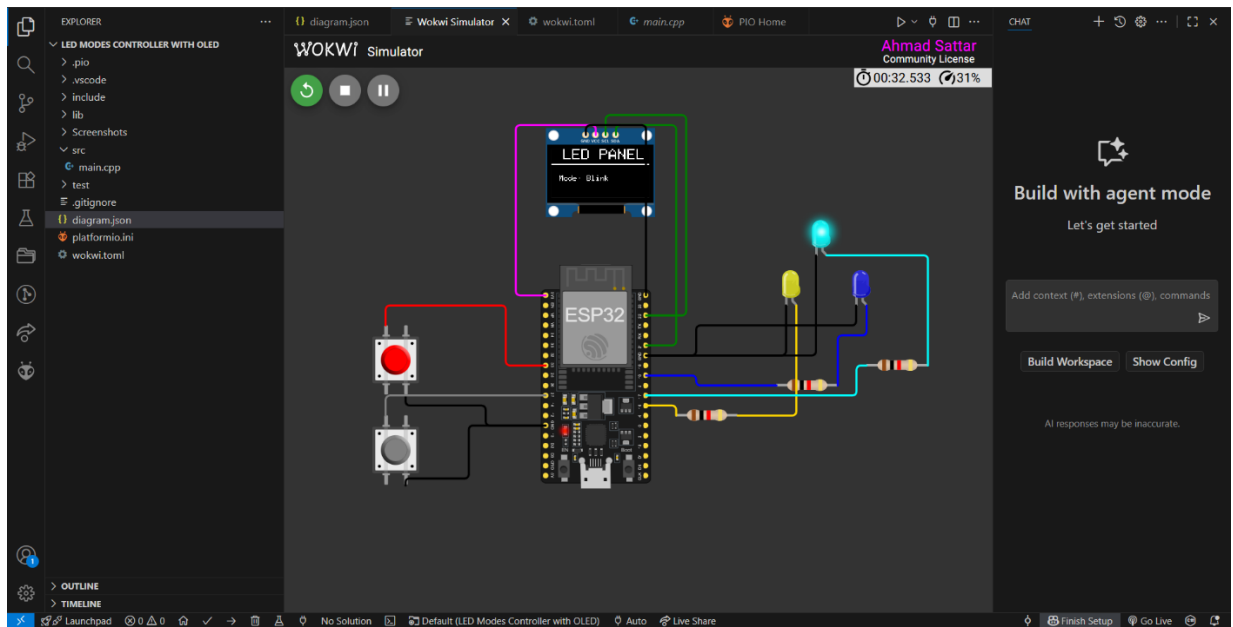
```

WOKWI DEMO:

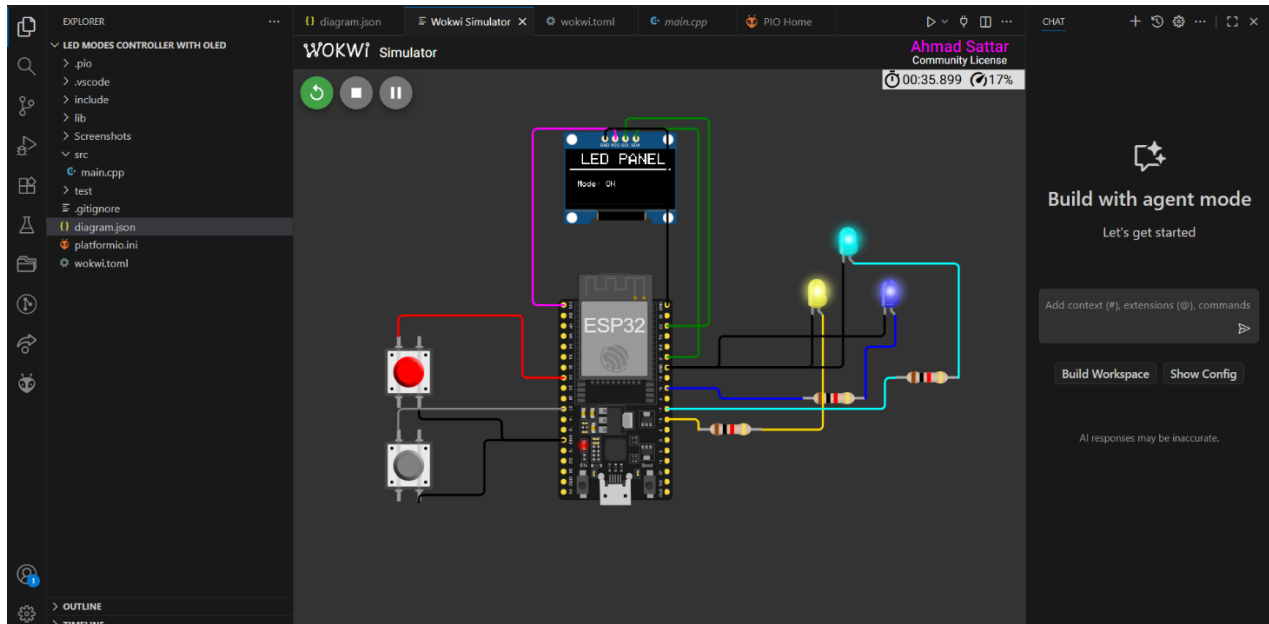
1. Both OFF



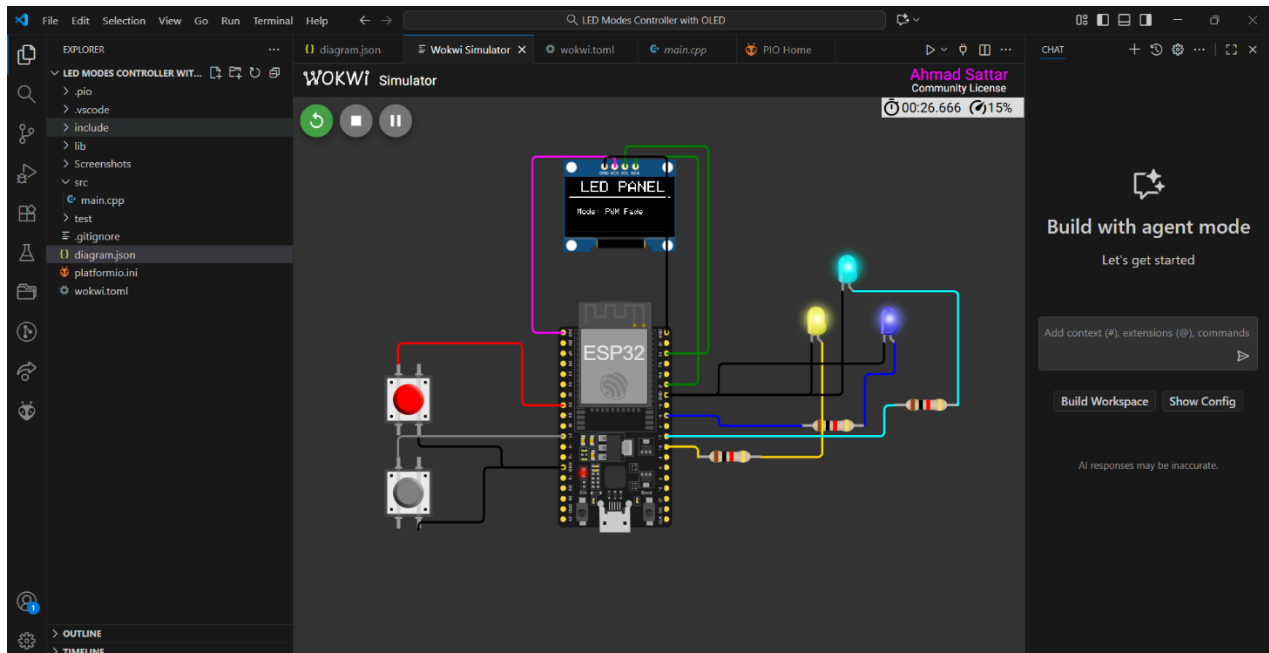
2. Alternate blink



3. Both ON

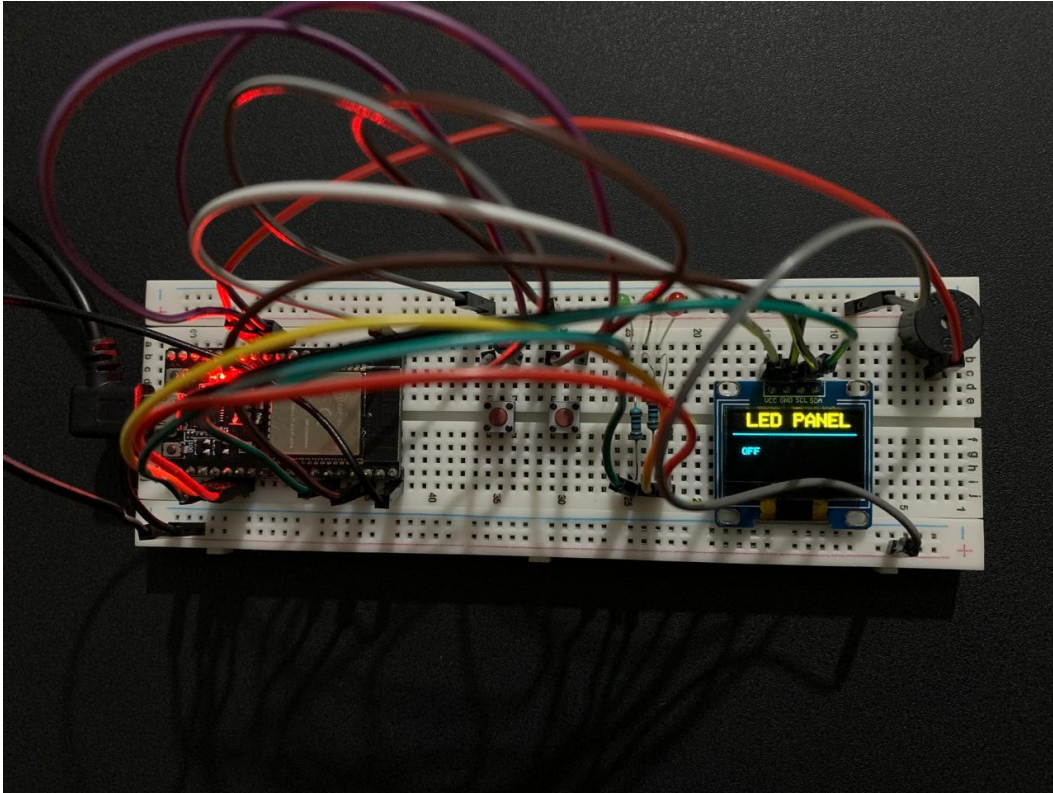


4. PWM fade

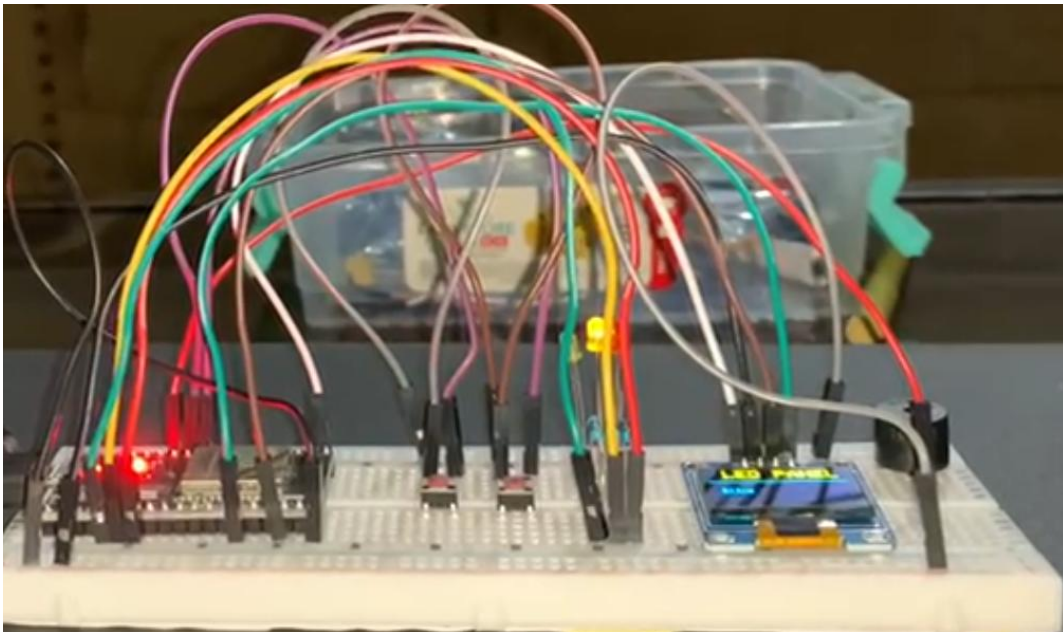


HARDWARE DEMO:

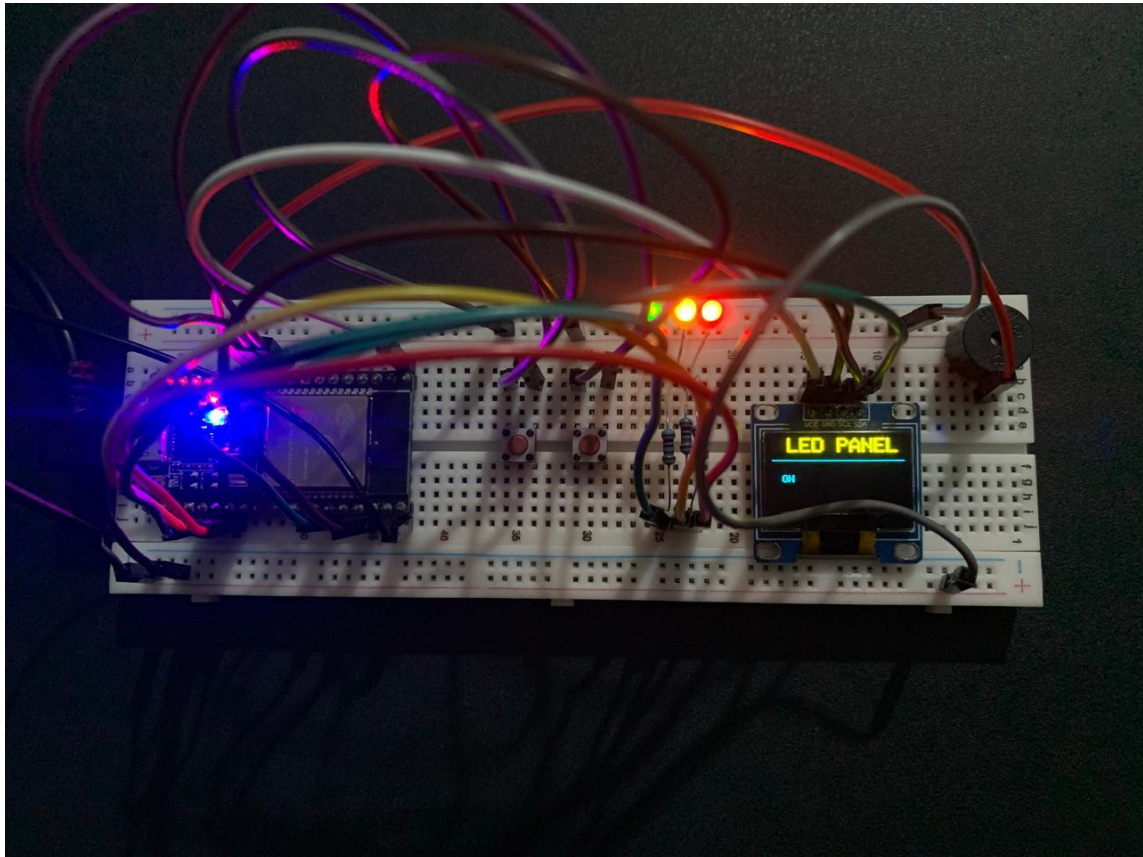
1. Both OFF



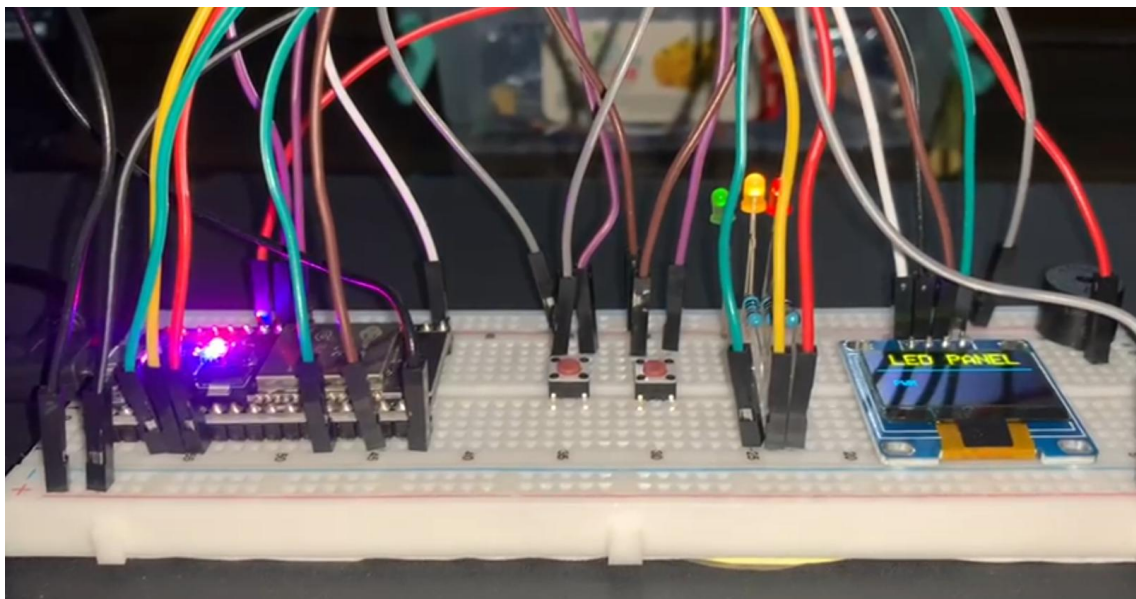
2. Alternate blink



3. Both ON



4. PWM fade



TASK – 2

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

CODE:

```
/*
Name: Ahmad Sattar
REG #: 23-NTU-CS-1059
TASK-2: LED & buzzer with Button
*/

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

#define OLED_W 128
#define OLED_H 64
Adafruit_SSD1306 display(OLED_W, OLED_H, &Wire, -1);

//PINS
#define LED_PIN 17
#define BUTTON_PIN 27
#define BUZZER_PIN 16

bool isLedOn = false;
bool buttonPressed = false;
bool longPressTriggered = false;
unsigned long pressStartTime = 0;
const unsigned long longPressTime = 2000;

void showMessage(const char* text) {
    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE);
    display.setTextSize(1);
    display.setCursor(0, 25);
    display.println(text);
    display.display();
}
```

```

}

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

    pinMode(LED_PIN, OUTPUT);
    pinMode(BUZZER_PIN, OUTPUT);
    pinMode(BUTTON_PIN, INPUT_PULLUP);

    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        while (true);
    }

    showMessage("initialize");
}

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

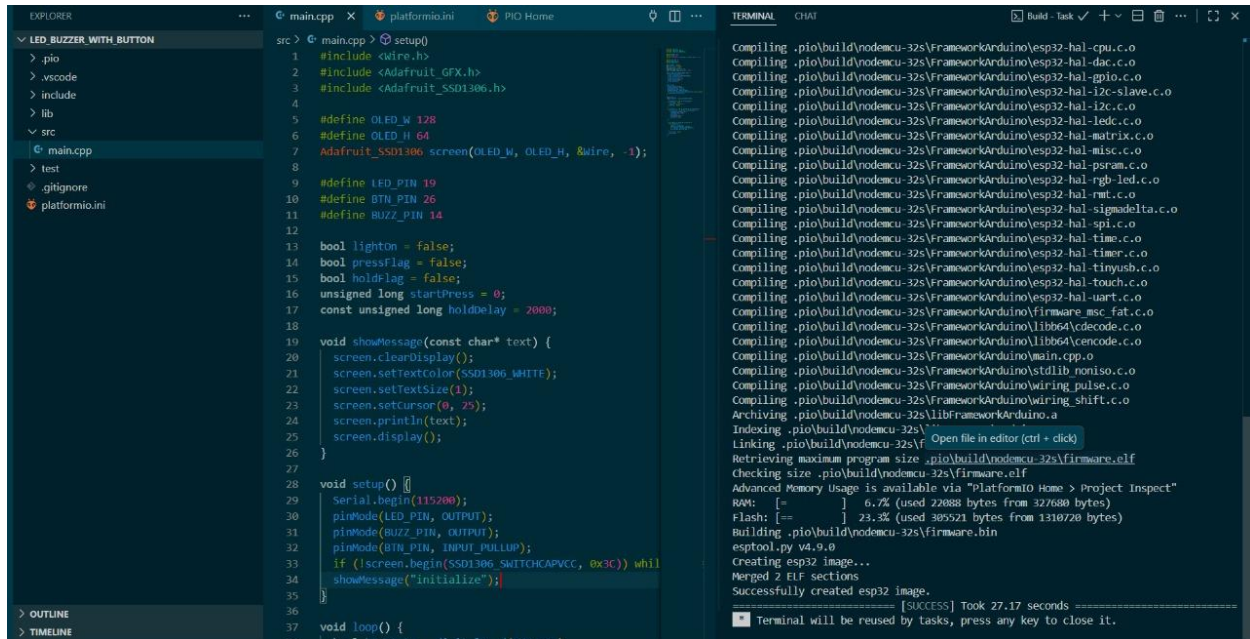
    // Button pressed initially
    if (buttonState == LOW && !buttonPressed) {
        buttonPressed = true;
        pressStartTime = millis();
        longPressTriggered = false;
    }

    // Check for long press
    if (buttonState == LOW && buttonPressed && !longPressTriggered) {
        if (millis() - pressStartTime >= longPressTime) {
            showMessage("  --Buzzer ON--");
            tone(BUZZER_PIN, 1500);
            delay(500);
            noTone(BUZZER_PIN);
            longPressTriggered = true;
        }
    }

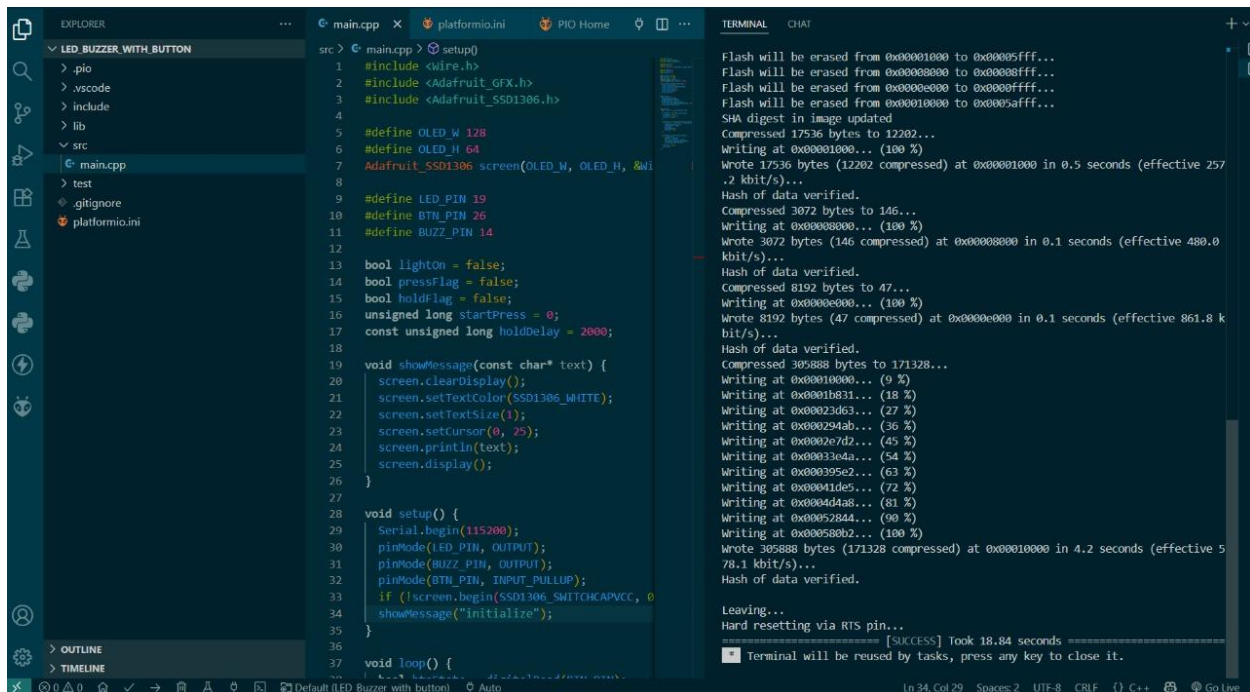
    // Button released
    if (buttonState == HIGH && buttonPressed) {
        if (!longPressTriggered) {
            isLedOn = !isLedOn;
            digitalWrite(LED_PIN, isLedOn);
            if (isLedOn) showMessage("  --LED ON--");
            else showMessage("  --LED OFF--");
        }
    }
}

```


BUILD:

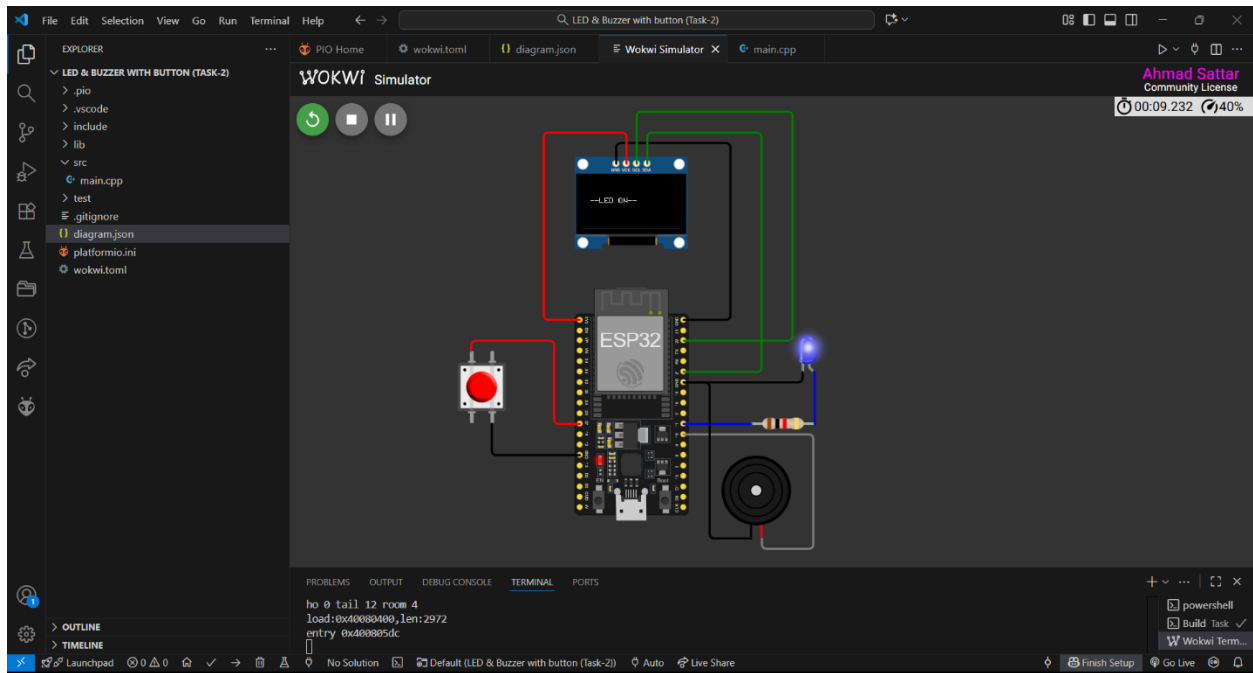


UPLOAD:

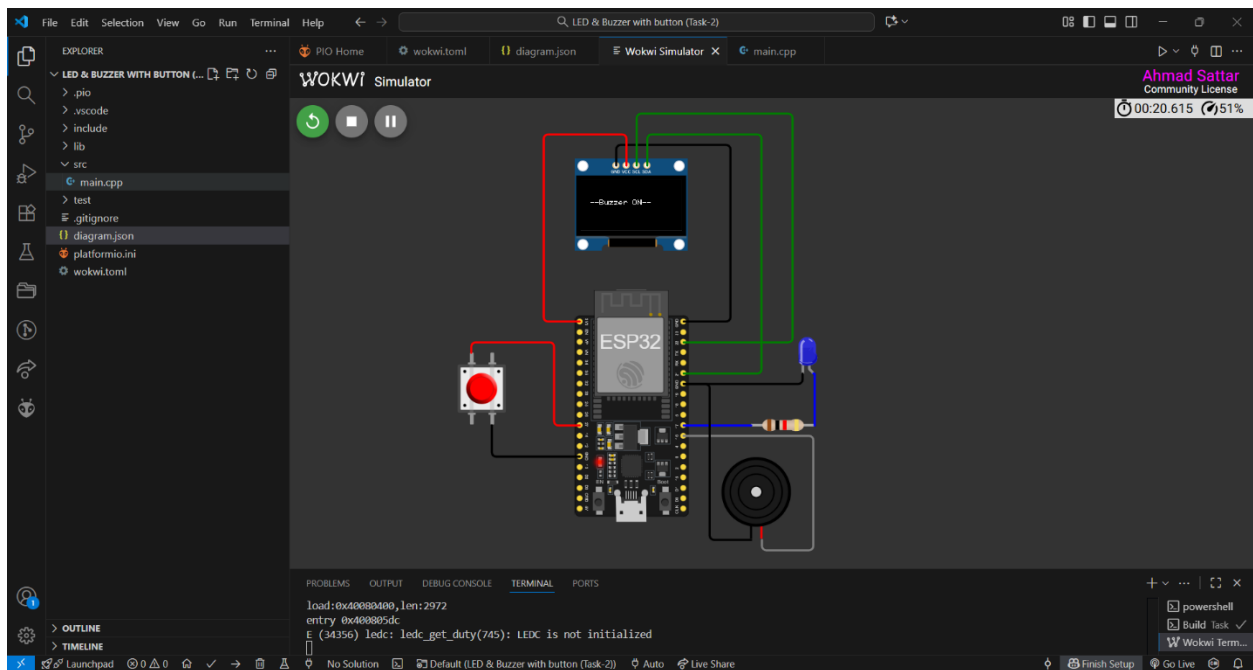


WOKWI DEMO:

➤ Short press → toggle LED

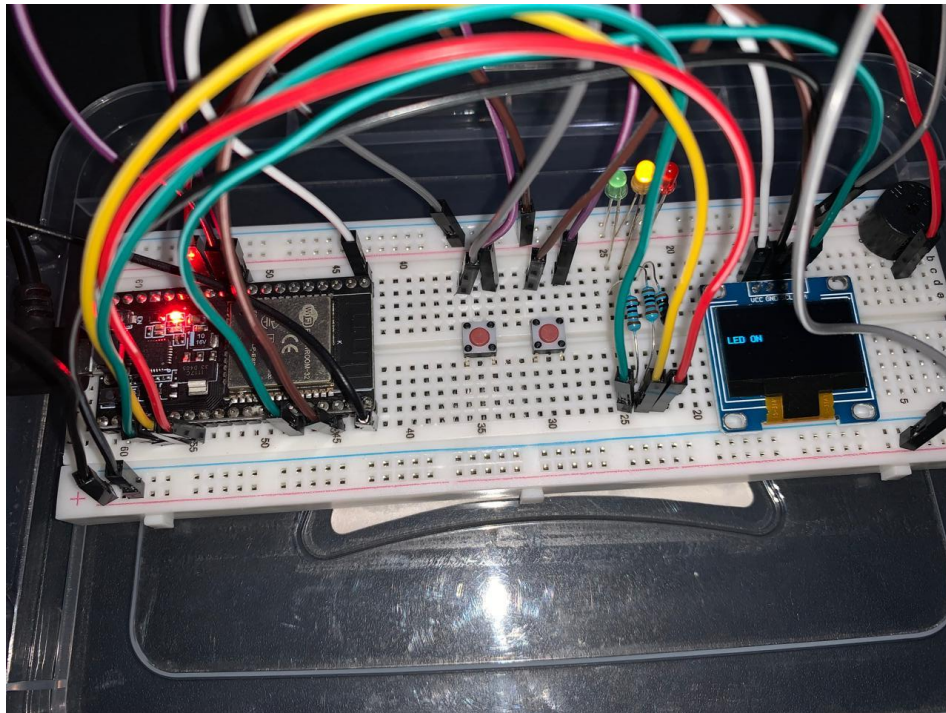


➤ Long press (> 1.5 s) Buzzer sounds

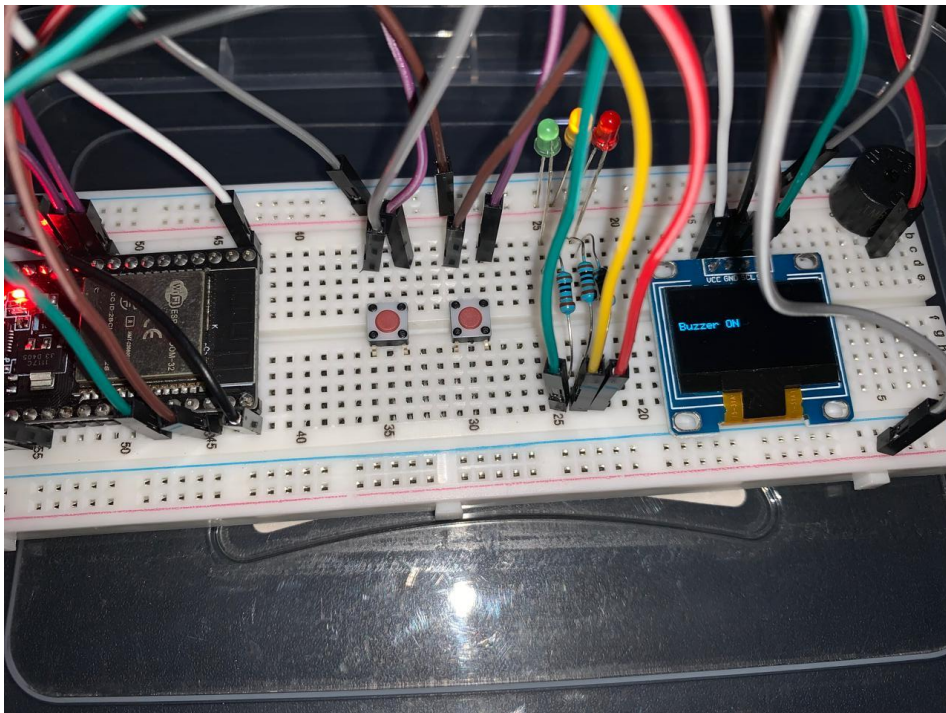


HARDWARE DEMO:

- Short press → toggle LED



- Long press (> 1.5 s) Buzzer sounds



HAND DRAWN CIRCUIT

