

National Textile University, Faisalabad



Department of Computer Science

Name:	MOHAMMAD ABU OBAIDA
Class:	BSCS 5 th B
Registration No:	23-NTU-CS-1052
Course Name:	Embedded IoT Systems
Submitted To:	Sir Nasir Mahmood
Submission Date:	25 - OCT - 2025

Question#3

Task-1

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

CODE:

```
//Name : Abu Obaida
//REG NO : 23-NTU-CS-1052
//Class:BSCS-5th-B

#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>    // Graphics library for OLED
#include <Adafruit_SSD1306.h> // OLED display driver library

// --- OLED display setup ---
#define W 128    // Screen width
#define H 64     // Screen height
Adafruit_SSD1306 scr(W, H, &Wire, -1); // Create display object

// --- Pin connections ---
#define LED_Y 4    // Yellow LED pin
#define LED_G 0    // Green LED pin
#define LED_R 2    // Red LED pin
#define BTN_MODE 26 // Button 1 → used to change LED mode
#define BTN_RST 27  // Button 2 → used to reset to OFF mode

// --- PWM channel setup (ESP32 uses PWM channels for analog control) ---
#define CH_Y 0
#define CH_G 1
#define CH_R 2
#define FREQ 4000 // PWM frequency (Hz)
#define RES 8     // 8-bit resolution (0-255 duty range)

// --- Variables ---
hw_timer_t *blinkT = nullptr; // Hardware timer for blinking
```

```

int modeSel = 0;           // Current LED mode (0=OFF,1=Blink,2=ON,3=PWM
fade)
int blinkStep = 0;         // Step for blinking colors
bool oldMode = HIGH;       // Stores previous button state
bool oldRst = HIGH;
unsigned long tPrev = 0;    // Stores last button press time
const int tDelay = 600;    // Debounce delay (ms)
volatile unsigned long tick = 0; // Counts timer ticks (used for blinking)

// --- Timer interrupt function ---
// Runs automatically every 1 second (in setup we set 1,000,000 µs = 1 sec)
void IRAM_ATTR timerTick() {
    tick++; // Increase tick count each second
}

// --- Function to show current mode on OLED ---
void showScreen() {
    scr.clearDisplay();
    scr.setTextSize(2);
    scr.setTextColor(SSD1306_WHITE);
    scr.setCursor(15, 0);
    scr.println("LED PANEL");
    scr.drawLine(0, 20, 127, 20, SSD1306_WHITE); // Divider line
    scr.setTextSize(1);
    scr.setCursor(10, 35);
    // Show text according to current mode
    if (modeSel == 0) scr.print("OFF");
    else if (modeSel == 1) scr.print("Blink");
    else if (modeSel == 2) scr.print("ON");
    else if (modeSel == 3) scr.print("PWM");
    scr.display(); // Update display
}

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

    // Set LED pins as output
    pinMode(LED_Y, OUTPUT);
    pinMode(LED_G, OUTPUT);
    pinMode(LED_R, OUTPUT);

    // Set button pins as input with pull-up resistors
    pinMode(BTN_MODE, INPUT_PULLUP);
    pinMode(BTN_RST, INPUT_PULLUP);
}

```

```

// Initialize OLED display
if (!scr.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    while (true); // Stop if OLED fails
}

// Set up PWM for each LED
ledcSetup(CH_Y, FREQ, RES);
ledcSetup(CH_G, FREQ, RES);
ledcSetup(CH_R, FREQ, RES);

// Attach LEDs to PWM channels
ledcAttachPin(LED_Y, CH_Y);
ledcAttachPin(LED_G, CH_G);
ledcAttachPin(LED_R, CH_R);

// Set up hardware timer to trigger every 1 second
blinkT = timerBegin(0, 80, true); // Timer 0, prescaler 80 1 tick = 1
µs
timerAttachInterrupt(blinkT, &timerTick, true);
timerAlarmWrite(blinkT, 1000000, true); // 1,000,000 µs = 1 second
timerAlarmEnable(blinkT);

// Start with all LEDs OFF
ledcWrite(CH_Y, 0);
ledcWrite(CH_G, 0);
ledcWrite(CH_R, 0);

showScreen(); // Display initial mode (OFF)
}

void loop() {
    bool nowMode = digitalRead(BTN_MODE); // Read Mode button
    bool nowRst = digitalRead(BTN_RST);   // Read Reset button

    // --- Handle button presses (with debounce delay) ---
    if (millis() - tPrev > tDelay) {
        if (nowMode == LOW && oldMode == HIGH) { // If Mode button pressed
            modeSel = (modeSel + 1) % 4;          // Go to next mode (0-3)
            blinkStep = 0;
            showScreen();                          // Update OLED
            tPrev = millis();
        }
        if (nowRst == LOW && oldRst == HIGH) { // If Reset button pressed
            modeSel = 0;                          // Back to OFF
            blinkStep = 0;
        }
    }
}

```

```

        showScreen();
        tPrev = millis();
    }
}

oldMode = nowMode; // Save button states
oldRst = nowRst;

// --- LED behavior based on mode ---
if (modeSel == 0) {
    // Mode 0: All LEDs OFF
    ledcWrite(CH_Y, 0);
    ledcWrite(CH_G, 0);
    ledcWrite(CH_R, 0);
}
else if (modeSel == 1) {
    // Mode 1: Alternate blinking (changes every second)
    static unsigned long lastTick = 0;
    if (tick != lastTick) { // Every time the timer ticks
        lastTick = tick;
        blinkStep = (blinkStep + 1) % 3;
        if (blinkStep == 0) {
            ledcWrite(CH_Y, 255);
            ledcWrite(CH_G, 0);
            ledcWrite(CH_R, 0);
        } else if (blinkStep == 1) {
            ledcWrite(CH_Y, 0);
            ledcWrite(CH_G, 255);
            ledcWrite(CH_R, 0);
        } else {
            ledcWrite(CH_Y, 0);
            ledcWrite(CH_G, 0);
            ledcWrite(CH_R, 255);
        }
    }
}
else if (modeSel == 2) {
    // Mode 2: All LEDs ON
    ledcWrite(CH_Y, 255);
    ledcWrite(CH_G, 255);
    ledcWrite(CH_R, 255);
}
else if (modeSel == 3) {
    // Mode 3: Smooth fading using PWM
    for (int i = 0; i <= 255 && modeSel == 3; i++) {

```

```

    ledcWrite(CH_Y, i);
    ledcWrite(CH_G, i);
    ledcWrite(CH_R, i);
    delay(5);
    if (digitalRead(BTN_MODE) == LOW || digitalRead(BTN_RST) == LOW) return; //
Stop fade if button pressed
}
for (int i = 255; i >= 0 && modeSel == 3; i--) {
    ledcWrite(CH_Y, i);
    ledcWrite(CH_G, i);
    ledcWrite(CH_R, i);
    delay(5);
    if (digitalRead(BTN_MODE) == LOW || digitalRead(BTN_RST) == LOW) return;
}
}
}
}

```

BUILD:

```

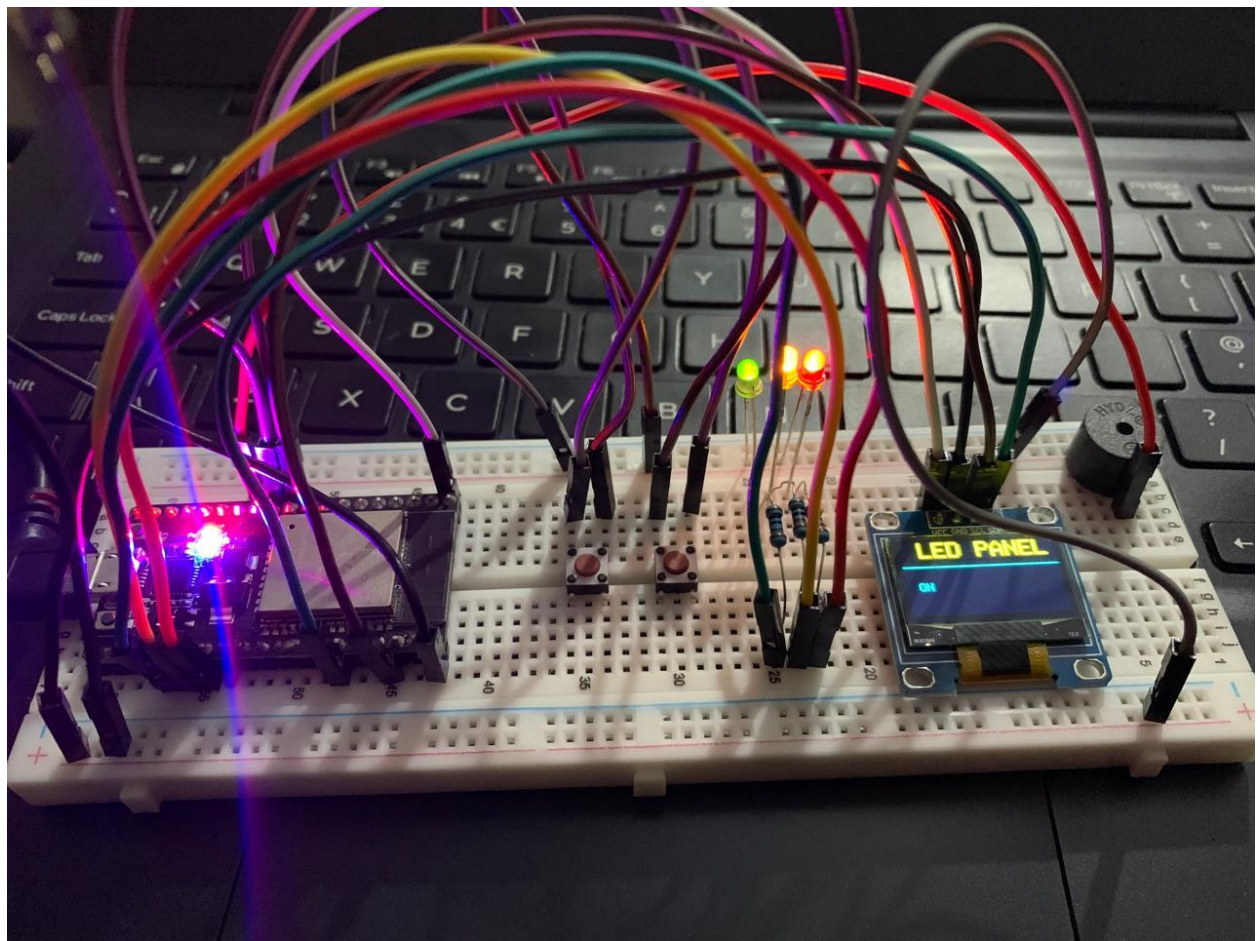
EXPLORER
PIO Home
main.cpp
src
  main.cpp
  test
  platformio.ini
  ...
  > .pio
  > .vscode
  > include
  > lib
  > src
  > test
  > .gitignore
  > platformio.ini

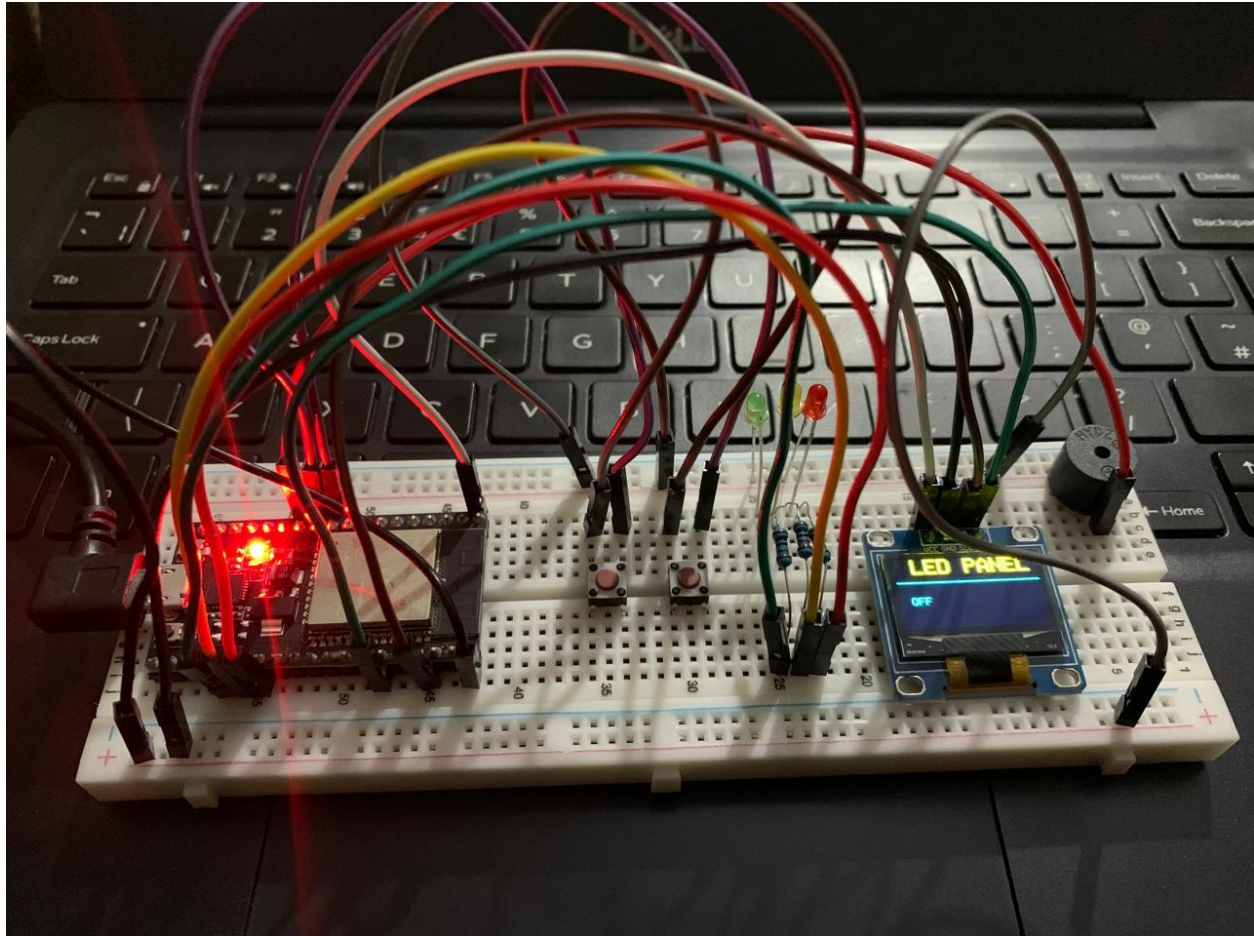
src > main.cpp
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <Adafruit_GFX.h>
4 #include <Adafruit_SSD1306.h>
5
6 #define W 128
7 #define H 64
8 Adafruit_SSD1306 scr(W, H, &Wire);
9
10 #define LED_Y 4
11 #define LED_G 0
12 #define LED_R 2
13 #define BTN_MODE 26
14 #define BTN_RST 27
15
16 #define 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 t;
31
32 void IRAM_ATTR timerTick() {
33   t++;
34 }
35
36 void showScreen() {
37   scr.clearDisplay();
38 }

TERMINAL
CHAT
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-cpu.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-dac.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-gpio.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-i2c-slave.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-i2c.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-ledc.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-matrix.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-misc.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-psram.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-rgb-led.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-rmt.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-sigmadelta.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-spi.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-time.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-timer.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-tinyusb.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-touch.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/esp32-hal-uart.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/Firmware_msc_fat.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/libb64/cdecode.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/libb64/cencode.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/main.cpp.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/utlib_noniso.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/wiring_pulse.c.o
Compiling .pio/build/nodemcu-32s/FrameworkArduino/wiring_shift.c.o
Archiving .pio/build/nodemcu-32s/lib/FrameworkArduino.a
Indexing .pio/build/nodemcu-32s/lib/FrameworkArduino.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 311153 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:





Task-2

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

CODE:

```
//Name : Abu Obaida
//REG NO : 23-NTU-CS-1052
//Class:BSCS-5th-B

#include <Wire.h>                // Library for I2C communication
#include <Adafruit_GFX.h>        // Graphics library for OLED display
#include <Adafruit_SSD1306.h>    // Library for controlling SSD1306 OLED

#define OLED_W 128              // OLED screen width in pixels
#define OLED_H 64              // OLED screen height in pixels
Adafruit_SSD1306 screen(OLED_W, OLED_H, &Wire, -1); // Create OLED object
```

```

#define LED_PIN 19           // LED connected to pin 19
#define BTN_PIN 26           // Button connected to pin 26
#define BUZZ_PIN 14          // Buzzer connected to pin 14

bool lightOn = false;        // Keeps track if LED is ON or OFF
bool pressFlag = false;      // True when button is pressed
bool holdFlag = false;       // True when button is held for long
unsigned long startPress = 0; // Stores the time when button was first
pressed
const unsigned long holdDelay = 2000; // 2 seconds hold time to trigger buzzer

// Function to show a message on the OLED screen
void showMessage(const char* text) {
    screen.clearDisplay();      // Clear the screen
    screen.setTextColor(SSD1306_WHITE); // Set text color to white
    screen.setTextSize(1);      // Set text size to small
    screen.setCursor(0, 25);     // Set text position (x=0, y=25)
    screen.println(text);        // Print the given text
    screen.display();           // Display it on the screen
}

void setup() {
    Serial.begin(115200);        // Start serial communication
    pinMode(LED_PIN, OUTPUT);    // Set LED pin as output
    pinMode(BUZZ_PIN, OUTPUT);   // Set buzzer pin as output
    pinMode(BTN_PIN, INPUT_PULLUP); // Set button pin as input with pull-up
    resistor

    // Initialize the OLED display
    if (!screen.begin(SSD1306_SWITCHCAPVCC, 0x3C))
        while (true); // Stop if the display is not found

    showMessage("initialize");    // Show startup message
}

void loop() {
    bool btnState = digitalRead(BTN_PIN); // Read button state

    // When button is first pressed
    if (btnState == LOW && !pressFlag) {
        pressFlag = true;          // Mark that button is pressed
        startPress = millis();      // Record the time of press
        holdFlag = false;          // Reset hold flag
    }
}

```

```

// Check if button is being held down
if (btnState == LOW && pressFlag && !holdFlag) {
    if (millis() - startPress >= holdDelay) { // Held for more than 2 seconds
        showMessage("Buzzer ON"); // Show message on screen
        tone(BUZZ_PIN, 1500); // Turn buzzer ON (1500Hz tone)
        delay(500); // Wait for half a second
        noTone(BUZZ_PIN); // Turn buzzer OFF
        holdFlag = true; // Mark that long press was handled
    }
}

// When button is released
if (btnState == HIGH && pressFlag) {
    if (!holdFlag) { // If it was a short press
        lightOn = !lightOn; // Toggle LED state
        digitalWrite(LED_PIN, lightOn); // Turn LED ON or OFF

        // Show LED status on screen
        if (lightOn) showMessage("LED ON");
        else showMessage("LED OFF");
    }
    pressFlag = false; // Reset button press flag
    delay(250); // Small delay to avoid bouncing
}
}

```

BUILD:

```
src > main.cpp > setup()
1 #include <Wire.h>
2 #include <Adafruit_GFX.h>
3 #include <Adafruit_SSD1306.h>
4
5 #define OLED_W 128
6 #define OLED_H 64
7 Adafruit_SSD1306 screen(OLED_W, OLED_H, &Wire, -1);
8
9 #define LED_PIN 19
10 #define BTN_PIN 26
11 #define BUZZ_PIN 14
12
13 bool lightOn = false;
14 bool pressFlag = false;
15 bool holdFlag = false;
16 unsigned long startPress = 0;
17 const unsigned long holdDelay = 2000;
18
19 void showMessage(const char* text) {
20   screen.clearDisplay();
21   screen.setTextColor(SSD1306_WHITE);
22   screen.setTextSize(1);
23   screen.setCursor(0, 25);
24   screen.println(text);
25   screen.display();
26 }
27
28 void setup() {
29   Serial.begin(115200);
30   pinMode(LED_PIN, OUTPUT);
31   pinMode(BUZZ_PIN, OUTPUT);
32   pinMode(BTN_PIN, INPUT_PULLUP);
33   if (!screen.begin(SSD1306_SWITCHCAPVCC, 0x3C)) while
34     showMessage("Initialize");
35 }
36
37 void loop() {
38   // ...
39 }
```

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-sigmadelta.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-libb64decode.c.o
Compiling .pio/build/nodemcu-32s/framework-arduino-libb64encode.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/libframework-arduino.a
Indexing .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: [6.7% (used 22088 bytes from 327680 bytes)]
Flash: [23.3% (used 305321 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 27.17 seconds
Terminal will be reused by tasks, press any key to close it.

UPLOAD:

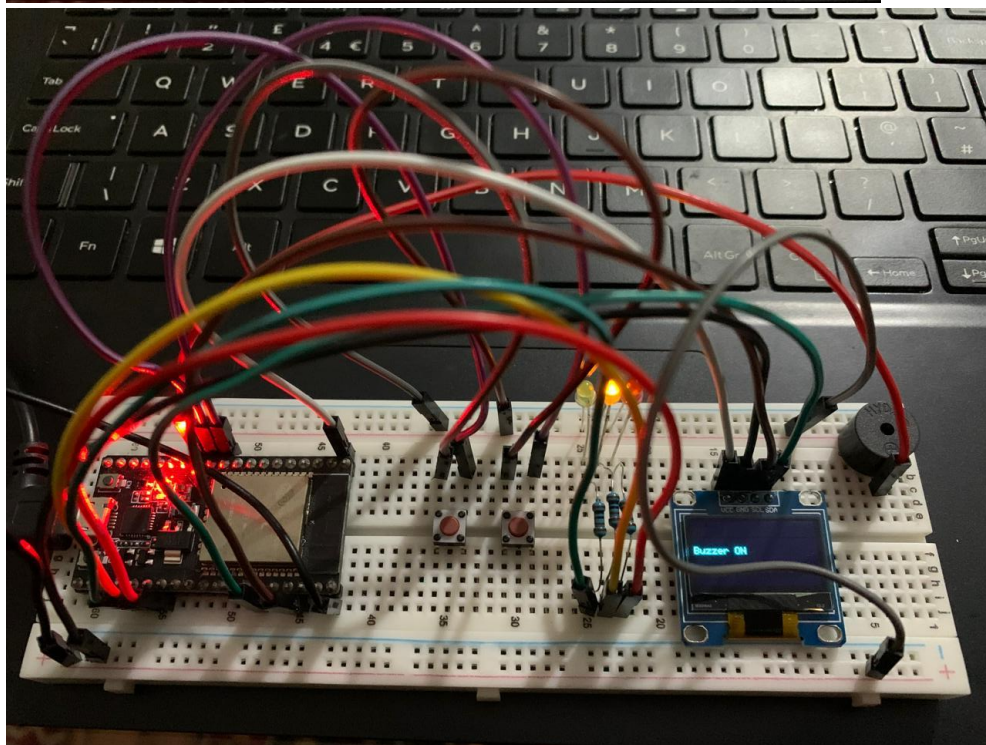
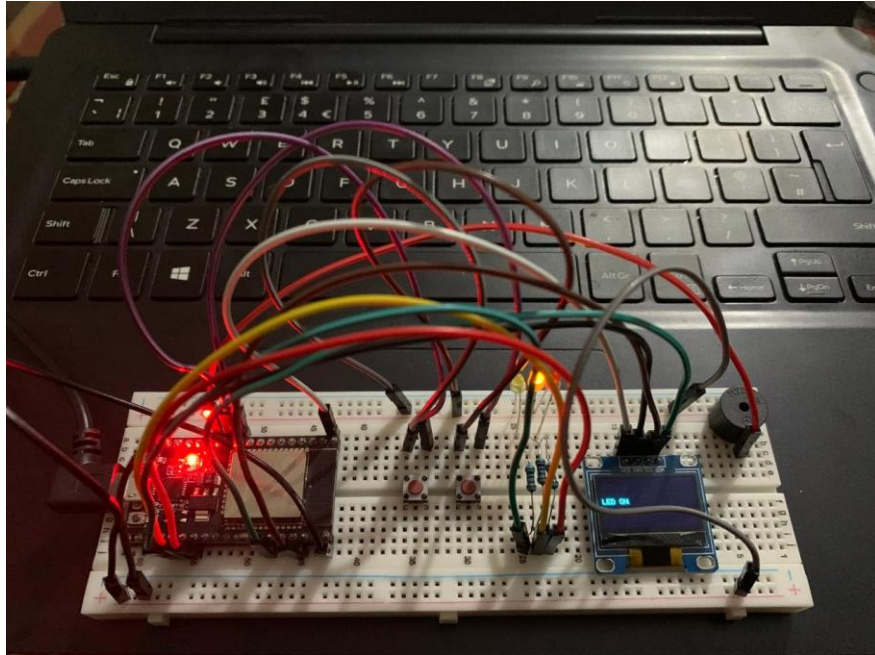
```
src > main.cpp > setup()
1 #include <Wire.h>
2 #include <Adafruit_GFX.h>
3 #include <Adafruit_SSD1306.h>
4
5 #define OLED_W 128
6 #define OLED_H 64
7 Adafruit_SSD1306 screen(OLED_W, OLED_H, &Wire, -1);
8
9 #define LED_PIN 19
10 #define BTN_PIN 26
11 #define BUZZ_PIN 14
12
13 bool lightOn = false;
14 bool pressFlag = false;
15 bool holdFlag = false;
16 unsigned long startPress = 0;
17 const unsigned long holdDelay = 2000;
18
19 void showMessage(const char* text) {
20   screen.clearDisplay();
21   screen.setTextColor(SSD1306_WHITE);
22   screen.setTextSize(1);
23   screen.setCursor(0, 25);
24   screen.println(text);
25   screen.display();
26 }
27
28 void setup() {
29   Serial.begin(115200);
30   pinMode(LED_PIN, OUTPUT);
31   pinMode(BUZZ_PIN, OUTPUT);
32   pinMode(BTN_PIN, INPUT_PULLUP);
33   if (!screen.begin(SSD1306_SWITCHCAPVCC, 0
34     showMessage("Initialize");
35 }
36
37 void loop() {
38   // ...
39 }
```

Flash will be erased from 0x00001000 to 0x00005fff...
Flash will be erased from 0x00008000 to 0x0000ffff...
Flash will be erased from 0x0000e000 to 0x0000ffff...
Flash will be erased from 0x00010000 to 0x0005ffff...
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 257.2 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 146...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (146 compressed) at 0x00008000 in 0.1 seconds (effective 480.0 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 861.8 kbit/s)...
Hash of data verified.
Compressed 305888 bytes to 171328...
Writing at 0x00010000... (9 %)
Writing at 0x00010831... (18 %)
Writing at 0x00023d63... (27 %)
Writing at 0x000294ab... (36 %)
Writing at 0x0002e7d2... (45 %)
Writing at 0x00033e4a... (54 %)
Writing at 0x000395e2... (63 %)
Writing at 0x00041des... (72 %)
Writing at 0x0004d4a8... (81 %)
Writing at 0x00052844... (90 %)
Writing at 0x000580b2... (100 %)
Wrote 305888 bytes (171328 compressed) at 0x00010000 in 4.2 seconds (effective 578.1 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
[SUCCESS] Took 18.84 seconds
Terminal will be reused by tasks, press any key to close it.

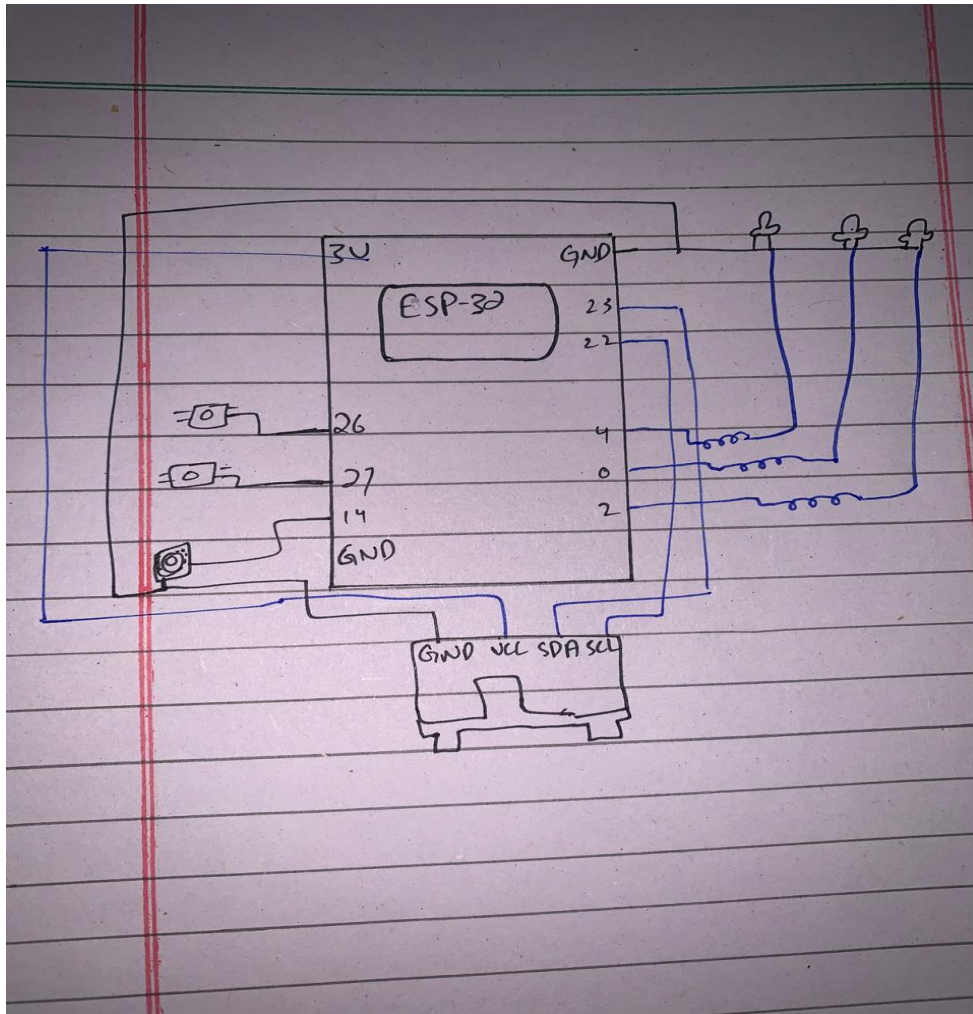
WOKWI Diagram:



HARDWARE OUTPUT:



HAND DRAWN CIRCUIT



PIN DIAGRAM:

<u>Pin No</u>	<u>Name</u>	<u>Function</u>	<u>Use Case</u>
GND	Ground	Common Ground	For all LEDs, Buzzer, Buttons, OLED
15	GPIO 26	Pin for mode Button Output for Button (Modebtn)	
16	GPIO 27	Pin for reset Button Output for Button (Resetbtn)	
17	GPIO 14	Pin for Buzzer	Output for Buzzer
3v3	Power	3.3V Output Power	OLED VCC
36	GPIO 22	I2C SCL	OLED SCL

<u>Pin No</u>	<u>Name</u>	<u>Function</u>	<u>Use Case</u>
39	GPIO 23	I2C SDA	OLED SDA
24	GPIO 4	Pin for Yellow LED	Output for Yellow LED
23	GPIO 0	Pin for Green LED	Output for Green LED
22	GPIO 2	Pin for Red LED	Output for Red LED