



Department of Computer Science

Name:	Muhammad Hanzala Zahid
Class:	BSCS 5th-B
Registration No:	23-NTU-CS-1067
Assignment#	01
Course Name:	Embedded IoT Systems (CSE-3080)
Submitted To:	Sir Nasir
Submission Date:	25-10-2025

TASK-B

Code Screenshots:

```
// Name: Muhammad Hanzala Zahid
// Roll no: 23-NTU-CS-1067

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

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

#define yellowLED 19 // Setting up GPIO pin for Yellow Led
#define blueButton 25 // Setting up GPIO pin for blue Button
#define buzzer 27 // Setting up GPIO pin for Buzzer

bool ledState = false; // Current state of the LED
unsigned long pressTime = 0; // Time when the button was pressed
bool buttonPressed = false; // Flag to track button press state
bool longPress = false; // Flag to track if long press was detected

const unsigned long longPressTime = 2000; // 2 seconds for long press

// Simple but visually better display message
void displayMessage(const char* line1, const char* line2 = "") {
    display.clearDisplay();
    // First line will be bold and large
    display.setTextSize(2); // Large text size
    display.setTextColor(SSD1306_WHITE); // White text
    display.setCursor(15, 10); // Centered cursor
    display.println(line1); // Print first line
    // Second line will be smaller and placed below First line
    if (line2[0] != '\0') { // If second line is not empty
        display.setTextSize(1); // Smaller text size
        display.setCursor(20, 40); // Centered cursor for second line
        display.println(line2); // Print second line
    }
    display.display();
}

void setup() {
    Serial.begin(115200);
    pinMode(yellowLED, OUTPUT); // setting up blue LED pin for getting output
    pinMode(buzzer, OUTPUT); // setting up buzzer pin for getting output
```

```

    pinMode(blueButton, INPUT_PULLUP); // Initialize blue button pin for taking
input
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println("OLED not found!"); // OLED not found
        while (true); // Halt execution if OLED is not found
    }
    displayMessage("Ready", "Press the Button");
}

void loop() {
    bool buttonState = digitalRead(blueButton); // Read button state

    // Detect whether the button is pressed or released
    if (buttonState == LOW && !buttonPressed) {
        buttonPressed = true; // Button is pressed
        pressTime = millis(); // Record the time when button was pressed
        longPress = false; // Reset long press flag
    }

    // Check for long press
    if (buttonState == LOW && buttonPressed) {
        if ((millis() - pressTime > longPressTime)) { // If pressed for more than
2 seconds
            displayMessage("Long Press", "Buzzer ON");
            tone(buzzer, 1000); // It will play tone of 1 kHz for 1 second
            delay(1000);
            noTone(buzzer); // Stop the tone
            displayMessage("Completed", "Buzzer OFF");
            delay(500); // Brief delay to show message
            longPress = true; // Set long press flag
        }
    }

    // Check for short press
    if (buttonState == HIGH && buttonPressed) {
        if (!longPress) { // If it was not a long press
            ledState = !ledState; // Toggle LED state
            digitalWrite(yellowLED, ledState); // Update LED state
            if (ledState == HIGH) { // If LED is on
                displayMessage("LED ON", "Short Press");
            } else {
                displayMessage("LED OFF", "Short Press");
            }
        }
    }

    buttonPressed = false; // Reset button pressed flag
    delay(300); // Debounce delay
}

```

```
}
```

Explanation about code:

This program is designed for an ESP32 (or Arduino) that uses a yellow LED, a buzzer, and an OLED display. When powered on, the display shows “Ready” message, indicating that the system is waiting for user input. The blue button controls the actions, if it’s pressed briefly, the LED toggles between ON and OFF, and the display updates to show the LED’s status. However, if the button is held down for more than two seconds, it’s recognized as a long press, causing the buzzer to sound for one second while displaying a “Buzzer ON” message.

Afterward, the screen confirms that the buzzer has turned off. Overall, the code provides both visual and sound feedback based on the type of button press

Build output:

```
4 #include <Wire.h>
5 #include <Adafruit_GFX.h>
6 #include <Adafruit_SSD1306.h>
7
8 #define SCREEN_WIDTH 128
9 #define SCREEN_HEIGHT 64
10 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
11
12
13 #define yellowLED 19 // Setting up GPIO pin for Yellow led
14 #define blueButton 25 // Setting up GPIO pin for blue Button
15 #define buzzer 27 // Setting up GPIO pin for Buzzer
16
17 bool ledState = false; // Current state of the LED
18 unsigned long pressTime = 0; // Time when the button was pressed
19 bool buttonPressed = false; // Flag to track button press state
20 bool longPress = false; // Flag to track if long press was detected
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Build - Task ✓ + - ▢ ... | ⌂ ×

Dependency Graph
— Adafruit GFX Library @ 1.12.3
— Adafruit SSD1306 @ 2.5.15
— Wire @ 2.0.0

Building in release mode
Retrieving maximum program size .pio/build/esp32dev/firmware.elf
Checking size .pio/build/esp32dev/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 305873 bytes from 1310720 bytes)

[SUCCESS] Took 2.55 seconds

Terminal will be reused by tasks, press any key to close it.

Upload output:

```
4 #include <Wire.h>
5 #include <Adafruit_GFX.h>
6 #include <Adafruit_SSD1306.h>
7
8 #define SCREEN_WIDTH 128
9 #define SCREEN_HEIGHT 64
10 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
11
12
13 #define yellowLED 19 // Setting up GPIO pin for Yellow led
14 #define blueButton 25 // Setting up GPIO pin for blue Button
15 #define buzzer 27 // Setting up GPIO pin for Buzzer
16
17 bool ledState = false; // Current state of the LED
18 unsigned long pressTime = 0; // Time when the button was pressed
19 bool buttonPressed = false; // Flag to track button press state
20 bool longPress = false; // Flag to track if long press was detected
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Upload - Task ✓ + - ▢ ... | ⌂ ×

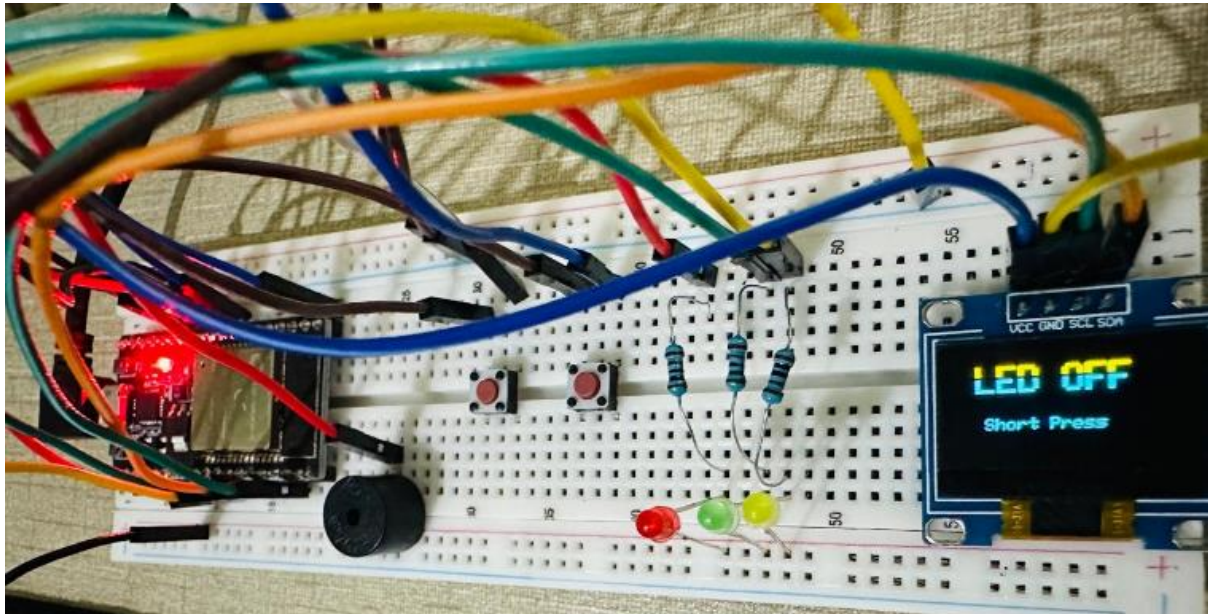
Writing at 0x0003cdd8... (63 %)
Writing at 0x0004233f... (72 %)
Writing at 0x000479a0... (81 %)
Writing at 0x00050089... (90 %)
Writing at 0x000522e1... (100 %)
Wrote 306240 bytes (17206 compressed) at 0x00010000 in 4.2 seconds (effective 581.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

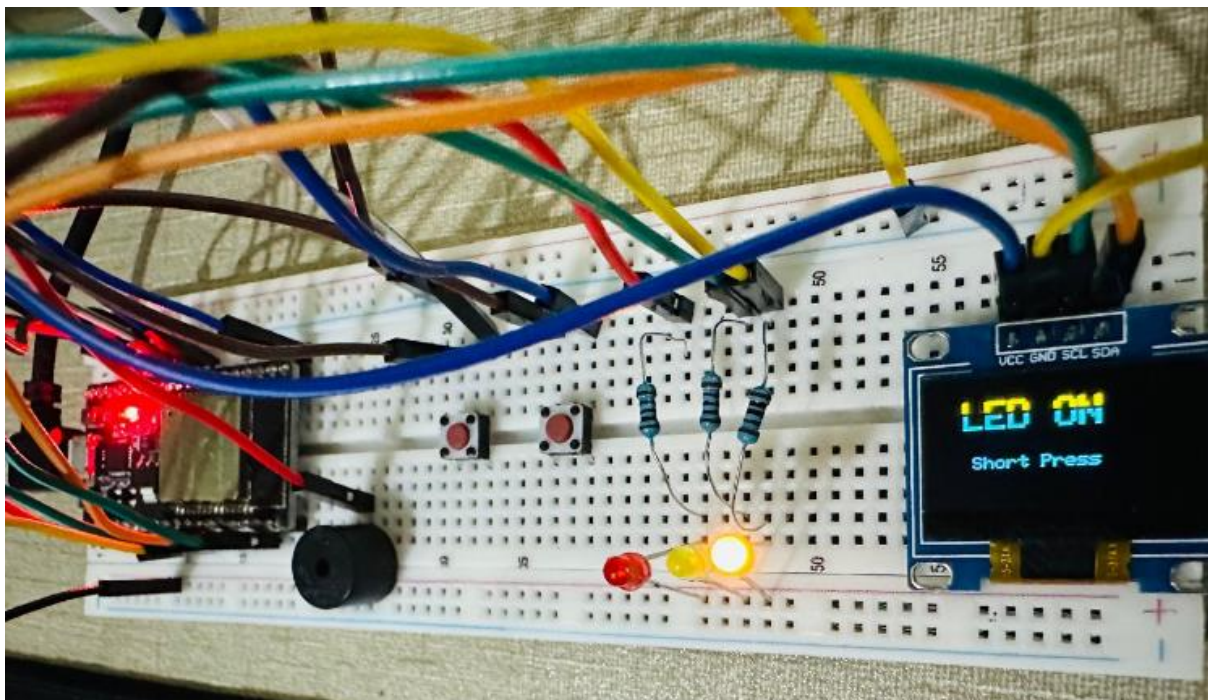
[SUCCESS] Took 10.97 seconds

Terminal will be reused by tasks, press any key to close it.

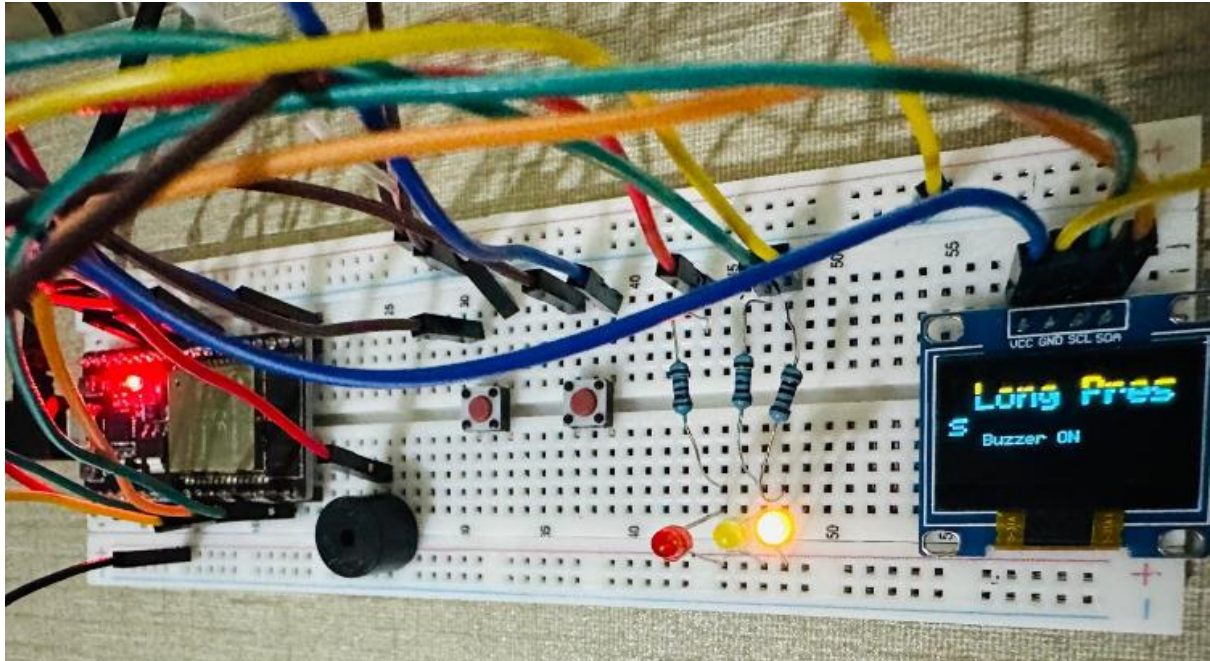
Hardware output:
Short press LED OFF:



Short press LED ON:



Long press:

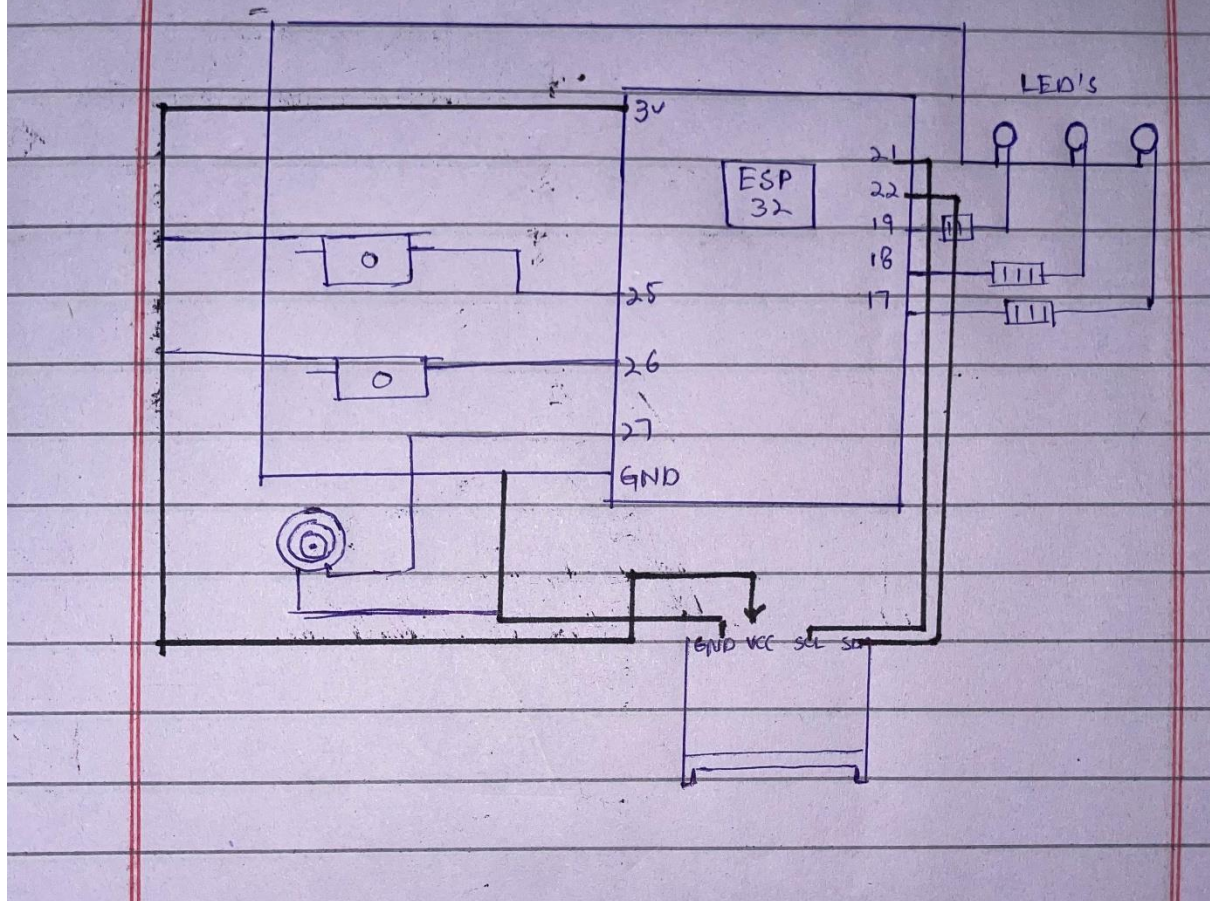


Wokwi project link(task-B):

<https://wokwi.com/projects/445771717191961601>

Circuit Diagram:

Circuit Diagram:-



Pin mapping diagram:

Pin No	Name	Function	Use Case
GND .2	Ground	Common Ground	For all LEDs, Buzzer, Buttons, OLED
25	GPIO 25	Pin for Blue Button	Output for Blue Button (Modebtn)
26	GPIO 26	Pin for White Button	Output for White Button (Resetbtn)
27	GPIO 27	Pin for Buzzer	Output for Buzzer
3v3	Power	3.3V output power	OLED VCC
22	GPIO 22	I2C SCL	OLED SCL
21	GPIO 21	I2C SDA	OLED SDA
19	GPIO 19	Pin for Yellow LED	Output for Yellow LED
18	GPIO 18	Pin for Green LED	Output for Green LED
17	GPIO 17	Pin for Red LED	Output for Red LED