National Textile University, Faisalabad

Department of Computer Science

Name:	Muhammad Hamza Faisal
Class:	BSCS-B
Registration No:	23-NTU-CS-1066
assignment	Assignment 1
Course Code:	
Course Name:	Embedded iot systems
Submitted To:	Sir Nasir Mehmood
Submission Date:	26-10-2025

Task 1 – ESP32-Based Multi-Device System (LEDs, Buttons, OLED, and Buzzer)

Overview

This task involves programming an ESP32 Devkit-C V4 to manage multiple hardware components — including LEDs, push buttons, a buzzer, and an OLED screen — through the Adafruit SSD1306 library. The goal was to create an interactive embedded system that provides immediate user feedback.

Working Process

- Each button is configured to control a unique action, such as toggling LED states or turning the buzzer on and off.
- The OLED display serves as a real-time feedback tool, showing clear messages like **“LED Turned On”**, **“LED Turned Off”**, or **“Buzzer Enabled”**.
- Hardware pin mapping:
    - LEDs → GPIO pins 2, 4, and 5
    - Buzzer → GPIO 15
    - Buttons → GPIO 26 and 27 with internal pull-up resistors
    - OLED display → I^2C communication on GPIO 21 (SDA) and GPIO 22 (SCL), address **0x3C**
- Each LED uses a 420Ω resistor to limit current flow.
- The complete setup was built and verified through **Wokwi simulation** to ensure expected performance.

System Behaviour

When any button is pressed, the ESP32 detects the signal instantly, updates the corresponding output device, and displays the status message on the OLED. The overall system reaction time is under one second, providing smooth, real-time control.

Learning Outcome

This experiment helped in understanding how to integrate multiple input and output peripherals with the ESP32 and how to present real-time responses using a graphical OLED interface.

Task 2 – Detecting Button Press Duration (Short vs Long Press)

Overview

The second task demonstrates how to differentiate between short and long button presses using ESP32. The project integrates an LED, a buzzer, and an OLED screen for visual and auditory feedback based on press duration.

Working Process

- A single push button is connected to **GPIO 25** as an input device.
- The program measures the time interval for which the button remains pressed using the **millis()** function.
- If the button is pressed for **less than 1.5 seconds**, it is counted as a *short press*, causing the LED (GPIO 5) to toggle.
- If the button remains pressed for **more than 1.5 seconds**, it is considered a *long press*, activating the buzzer (GPIO 18) for 0.5 seconds.
- The OLED screen connected via I^{2}C (SDA 21, SCL 22) displays corresponding messages such as:
    - "Short Press → LED Toggled"
    - "Long Press → Buzzer Activated"
- Once the event is completed, the program resets automatically to detect the next button input.

System Timing

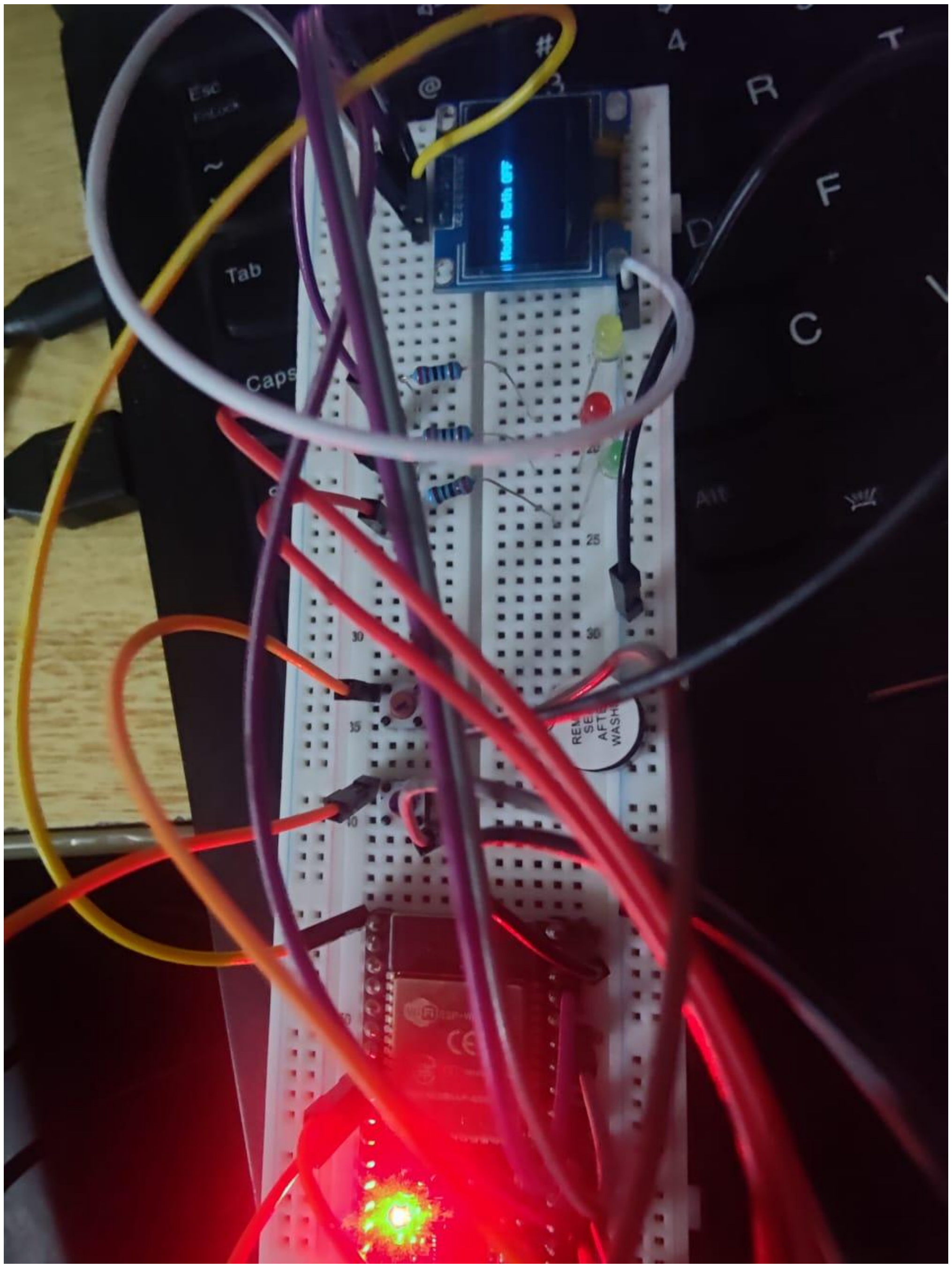
- **Short Press:** Duration < 1.5 seconds
- **Long Press:** Duration > 1.5 seconds
- OLED updates appear immediately, ensuring responsive feedback for every press.

Learning Outcome

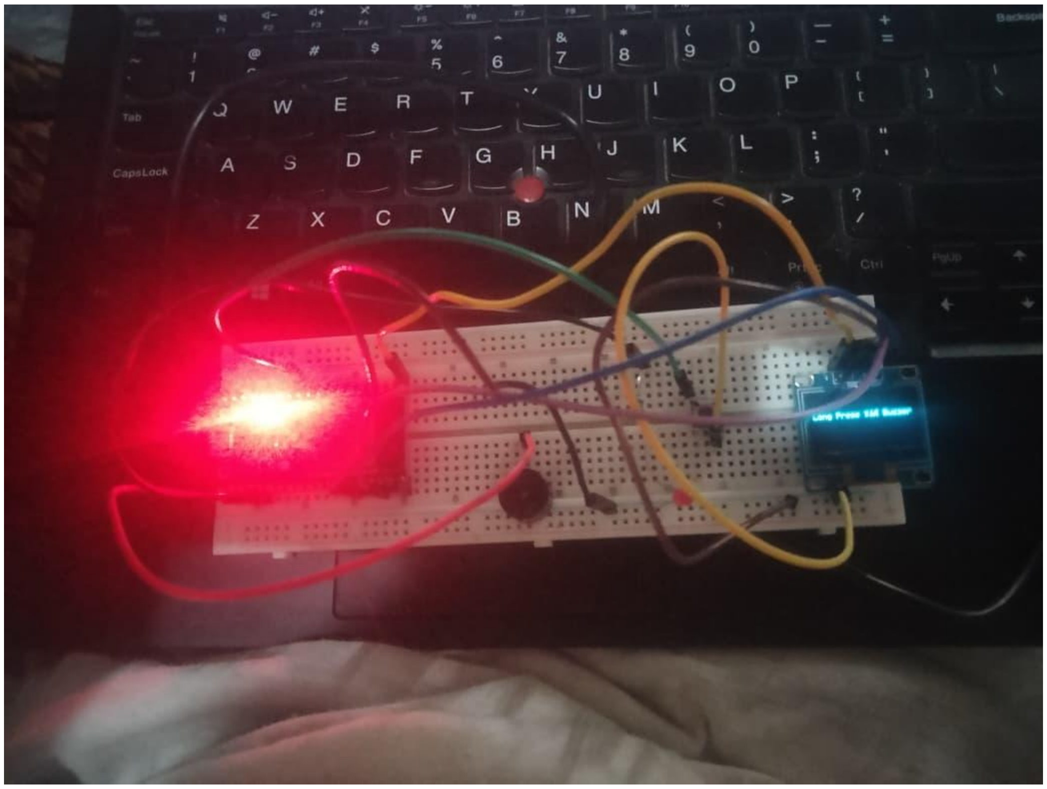
Through this task, the concept of detecting press durations using timing functions was understood. It also provided experience in designing responsive systems that combine multiple feedback mechanisms like light, sound, and visual messages.

Pictures:

Task1:

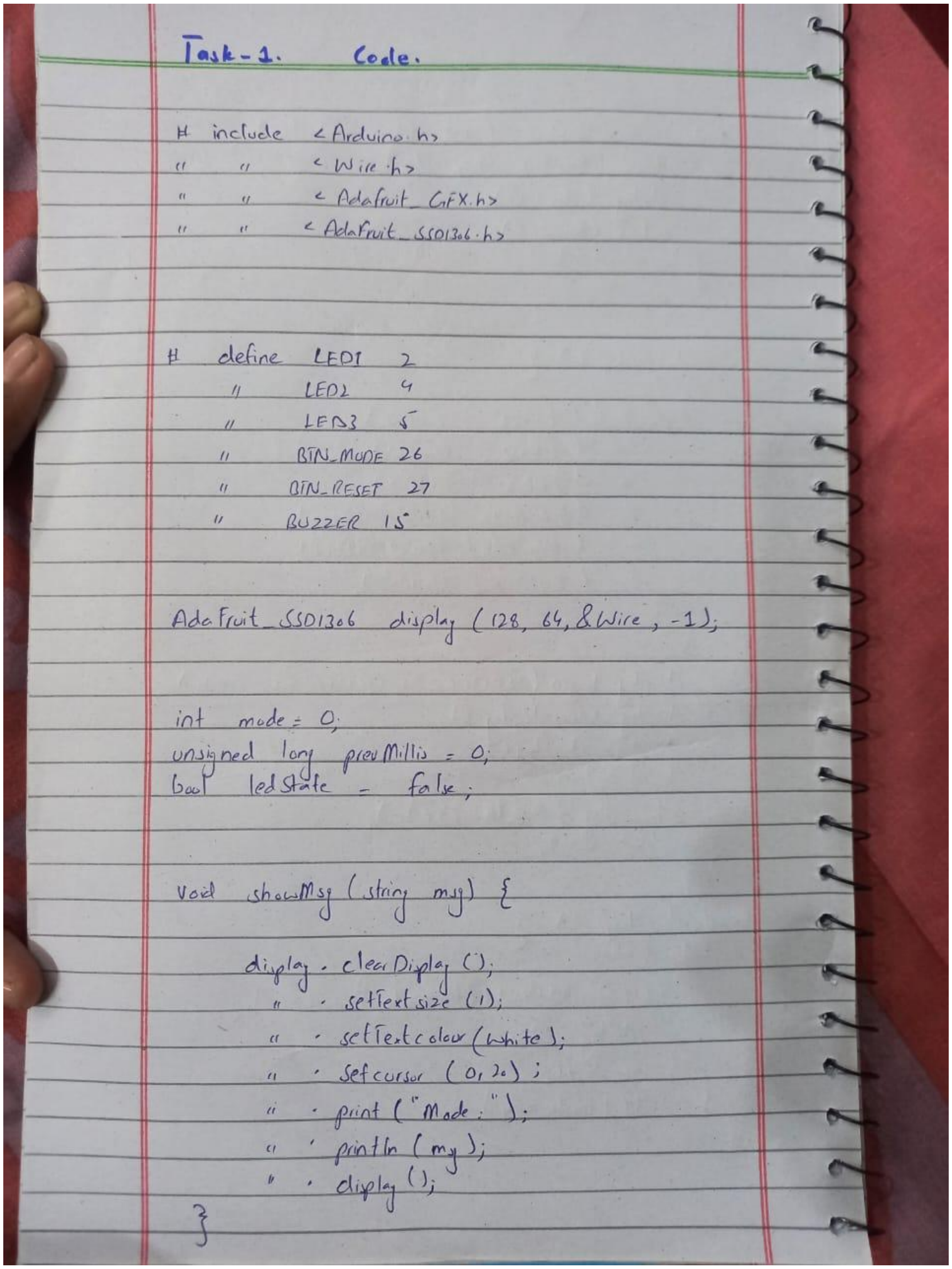


Task2:



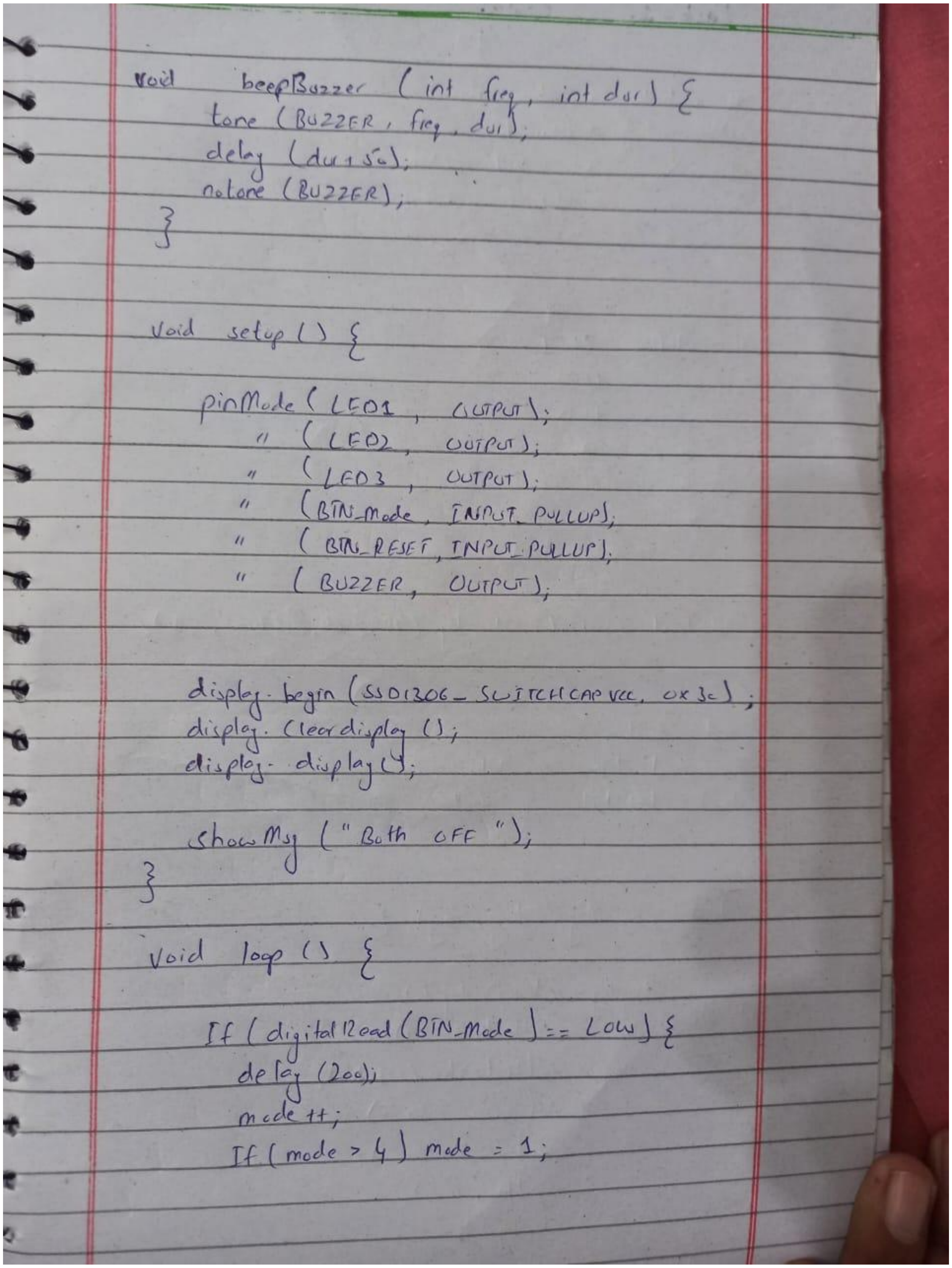
Handwritten code + kit diagram:

Task1 :

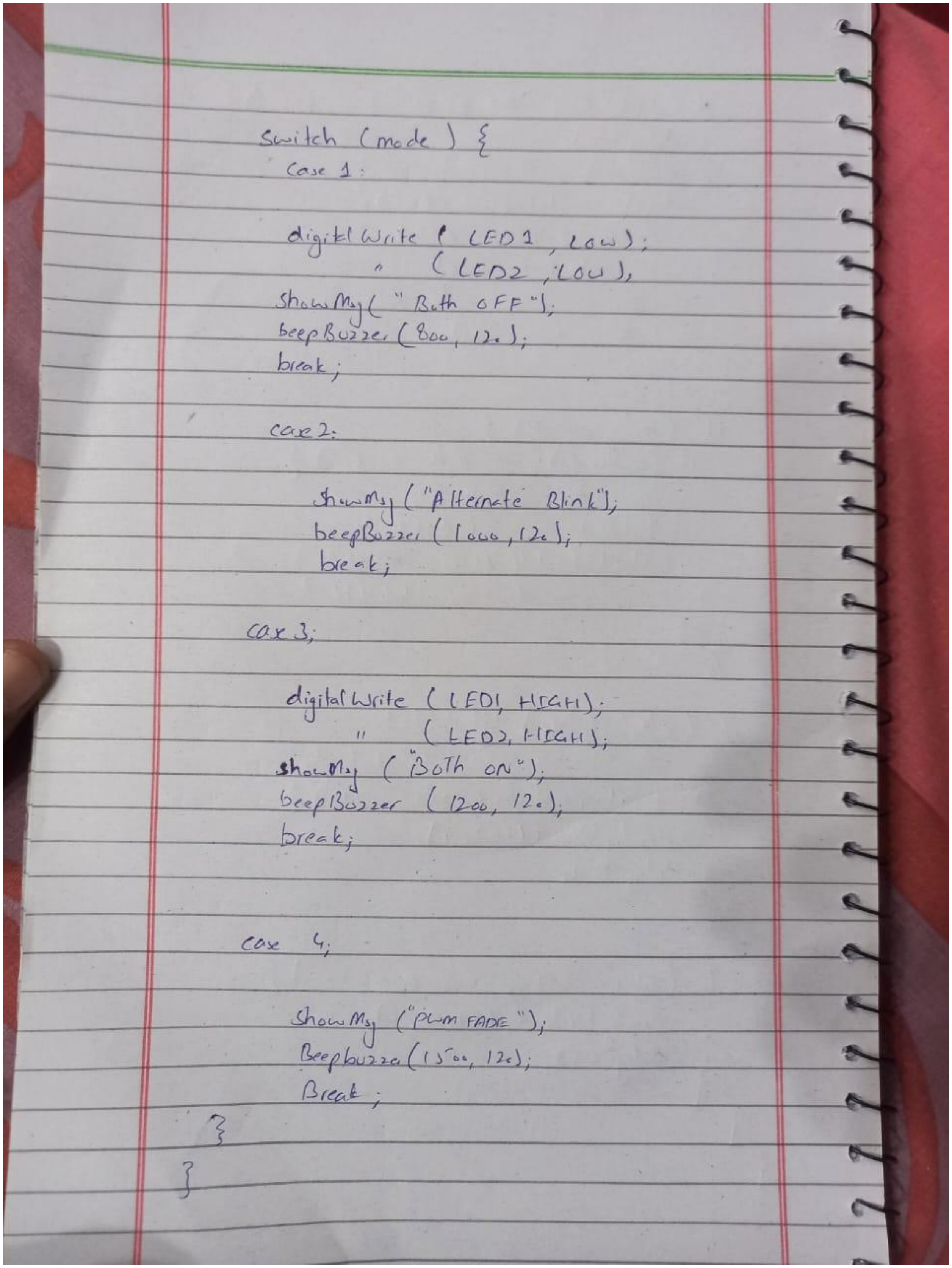


Task-1. Code.

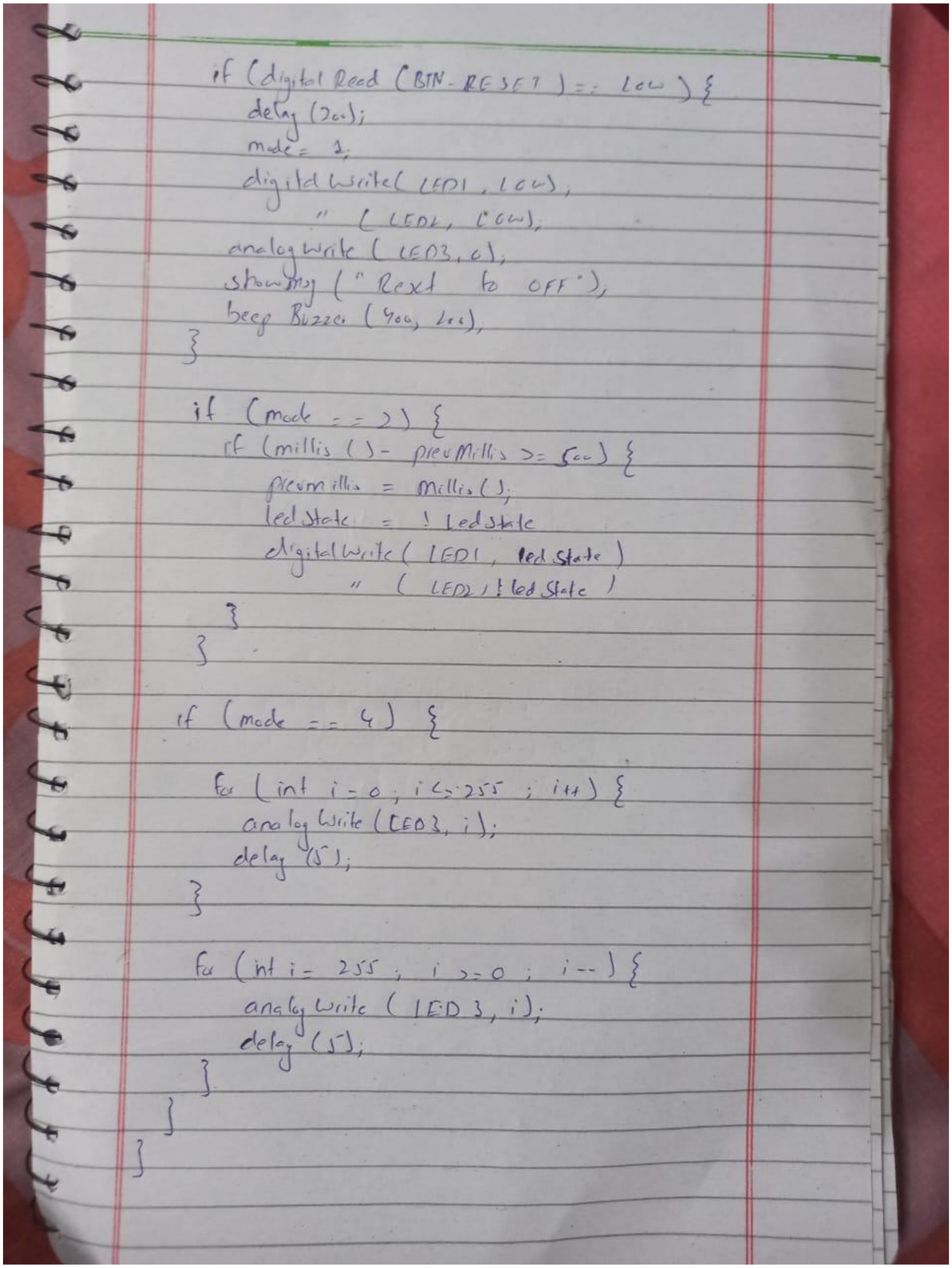
```
# include <Arduino.h>
"     "   <Wire.h>
"     "   <Adafruit_GFX.h>
"     "   <AdaFruit_SSD1306.h>


# define LED1 2
  "    LED2 4
  "    LED3 5
  "    BTN_MODE 26
  "    BTN_RESET 27
  "    BUZZER 15

AdaFruit_SSD1306 display (128, 64, &Wire, -1);

int mode = 0;
unsigned long prevMillis = 0;
bool ledState = false;

Void showMsg (string msg) {

     diplay . clearDiplay ();
       "    . setTextsize (1);
       "    . setTextcolour (white);
       "    . Setcursor (0, 20);
       "    . print ("Mode: ");
       "    . println (msg);
       "    . diplay ();
}
```



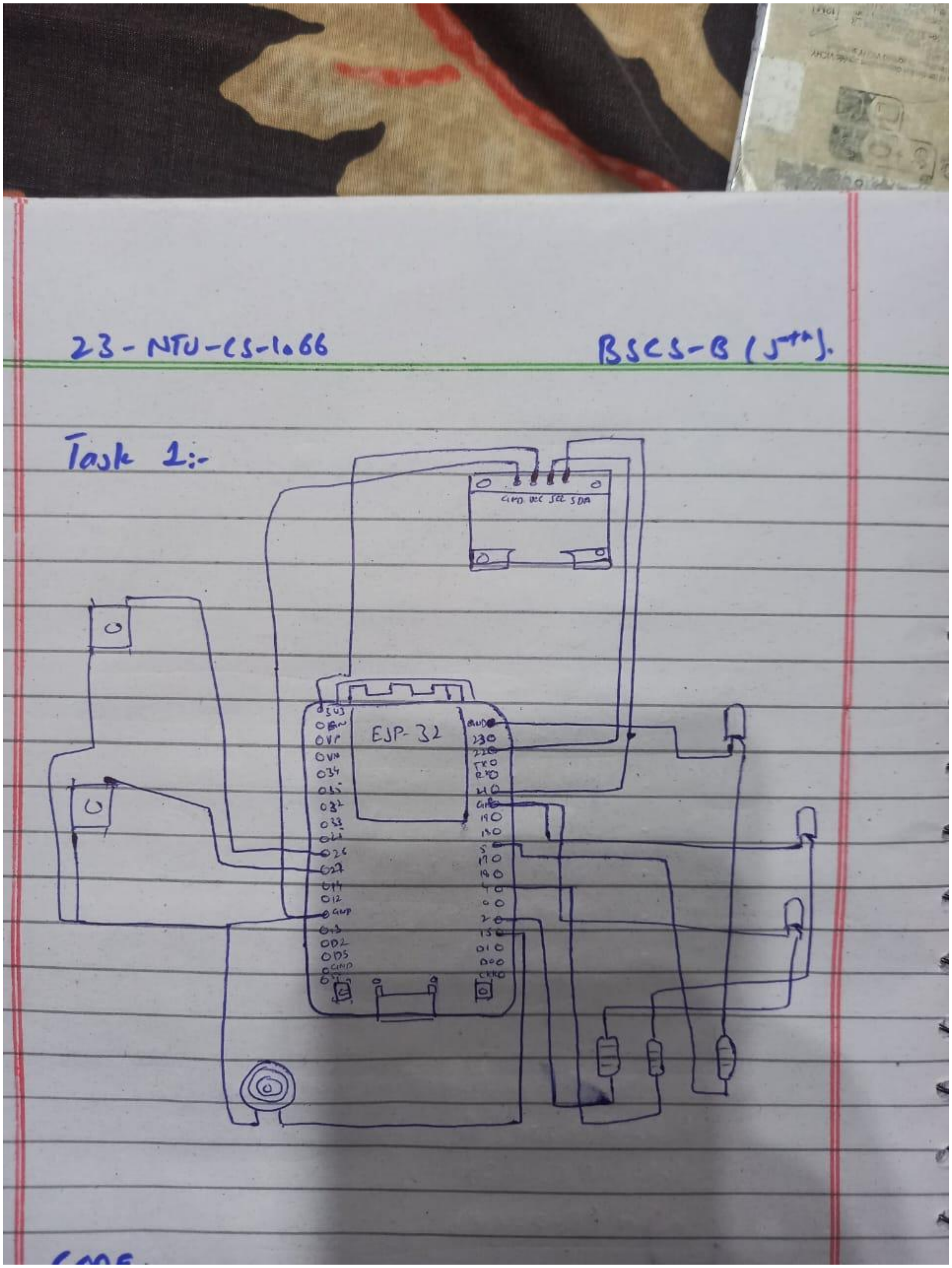
```
void   beepBuzzer (int freq, int dur) {
    tone (BUZZER, freq, dur);
    delay (dur+50);
    notone (BUZZER);
}


void setup () {

    pinMode (LED1,  OUTPUT);
       "    (LED2,  OUTPUT);
       "    (LED3,  OUTPUT);
       "    (BTN_mode, INPUT_PULLUP);
       "    (BTN_RESET, INPUT_PULLUP);
       "    (BUZZER,  OUTPUT);


    display.begin (SSD1306_SWITCHCAPVCC, 0x3c);
    display.cleardisplay ();
    display.display();

    showMsg ("Both OFF");
}

void loop () {

    If (digitalRead (BTN_mode) == LOW) {
      delay (200);
      mode ++;
      If (mode > 4) mode = 1;
```



```
switch (mode) {
  case 1:

   digitalWrite ( LED1 , LOW);
        "     ( LED2 , LOW),
  showMsg( "Both OFF");
  beepBuzzer (800, 120);
  break;

  case 2:

     showMsg ("Alternate Blink");
     beepBuzzer ( 1000, 120);
     break;

 case 3;

   digitalWrite ( LED1, HIGH);
        "      ( LED2, HIGH);
  showMsg ( "Both ON");
  beepBuzzer ( 1200, 120);
  break;


 case 4;

    showMsg ("PWM FADE ");
    Beepbuzzer(1500, 120);
    Break;
  }
}
```



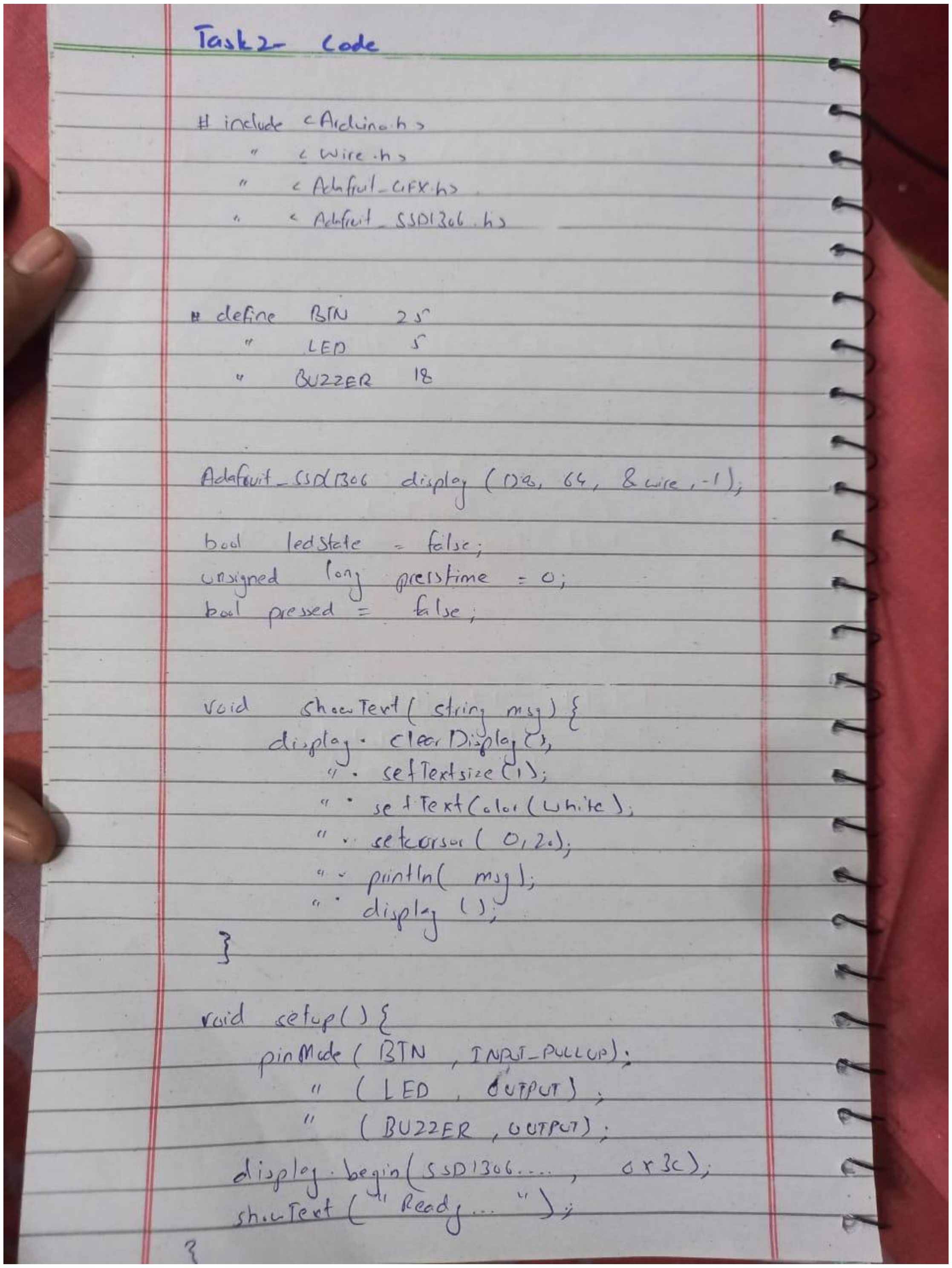
```
if (digital Read (BTN-RESET ) == LOW ) {
    delay (200);
    mode= 1;
    digital Write( LED1 , LOW),
         " ( LED2, LOW),
   analog Write ( LED3, 0);
   showMsg ("Rext  to  OFF"),
   beep Buzzer (400, 200),
}

if (mode == 2) {
   if (millis () - prevMillis >= 500) {
       prevmillis = millis ();
       ledState = ! ledState
       digitalWrite ( LED1 , led State )
             " ( LED2 , ! led State )
     }
 }

if (mode == 4) {

   for ( int i = 0 ; i <= 255 ; i++) {
     analog Write (LED3, i);
     delay (5);
 }

  for (int i = 255 ; i >= 0 ; i--) {
     analog Write ( LED 3, i);
     delay (5);
   }
 }
}
```



23-NTU-CS-1066 BSCS-B (5th).

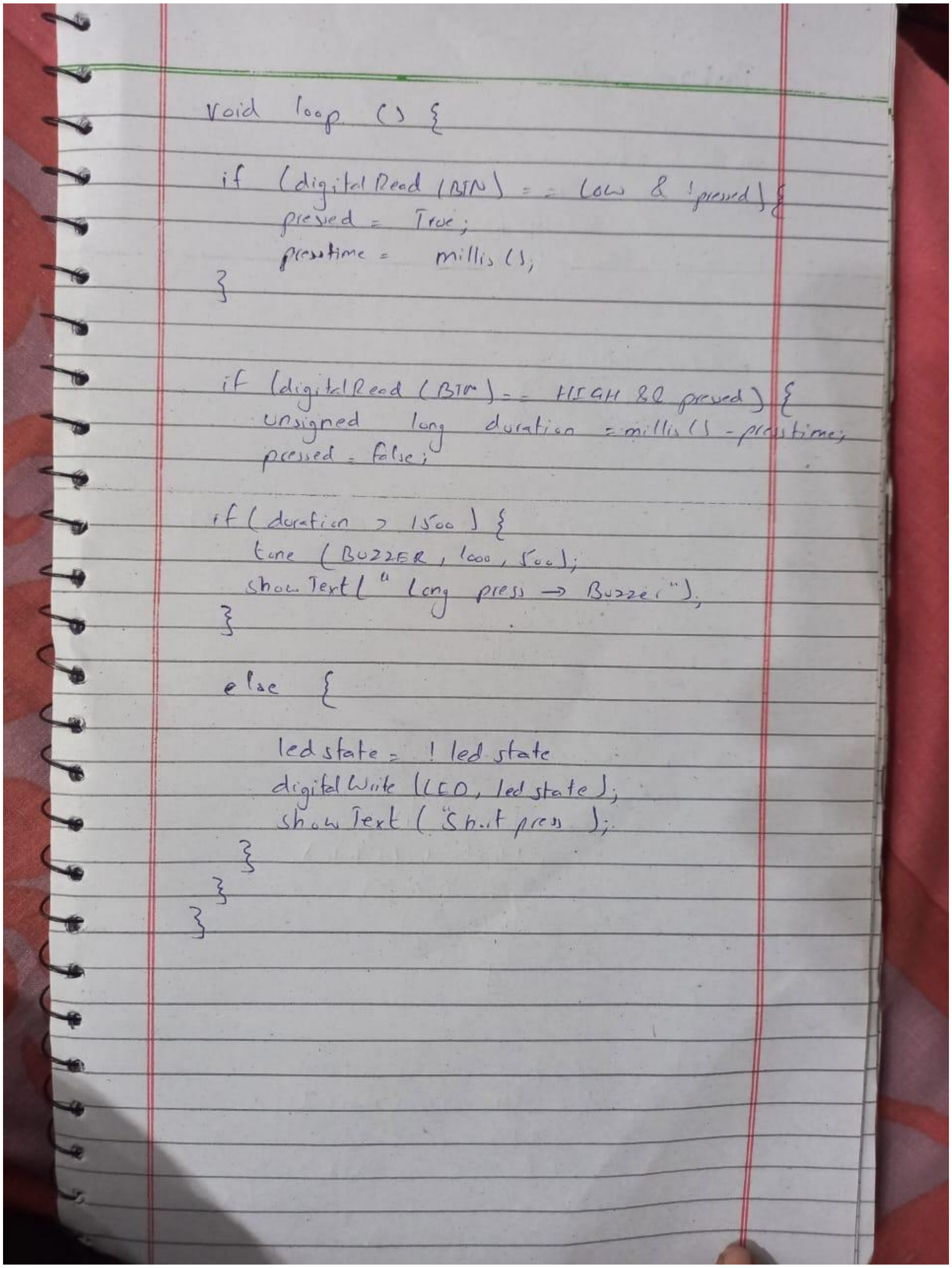
Task 1:-

Task 2:

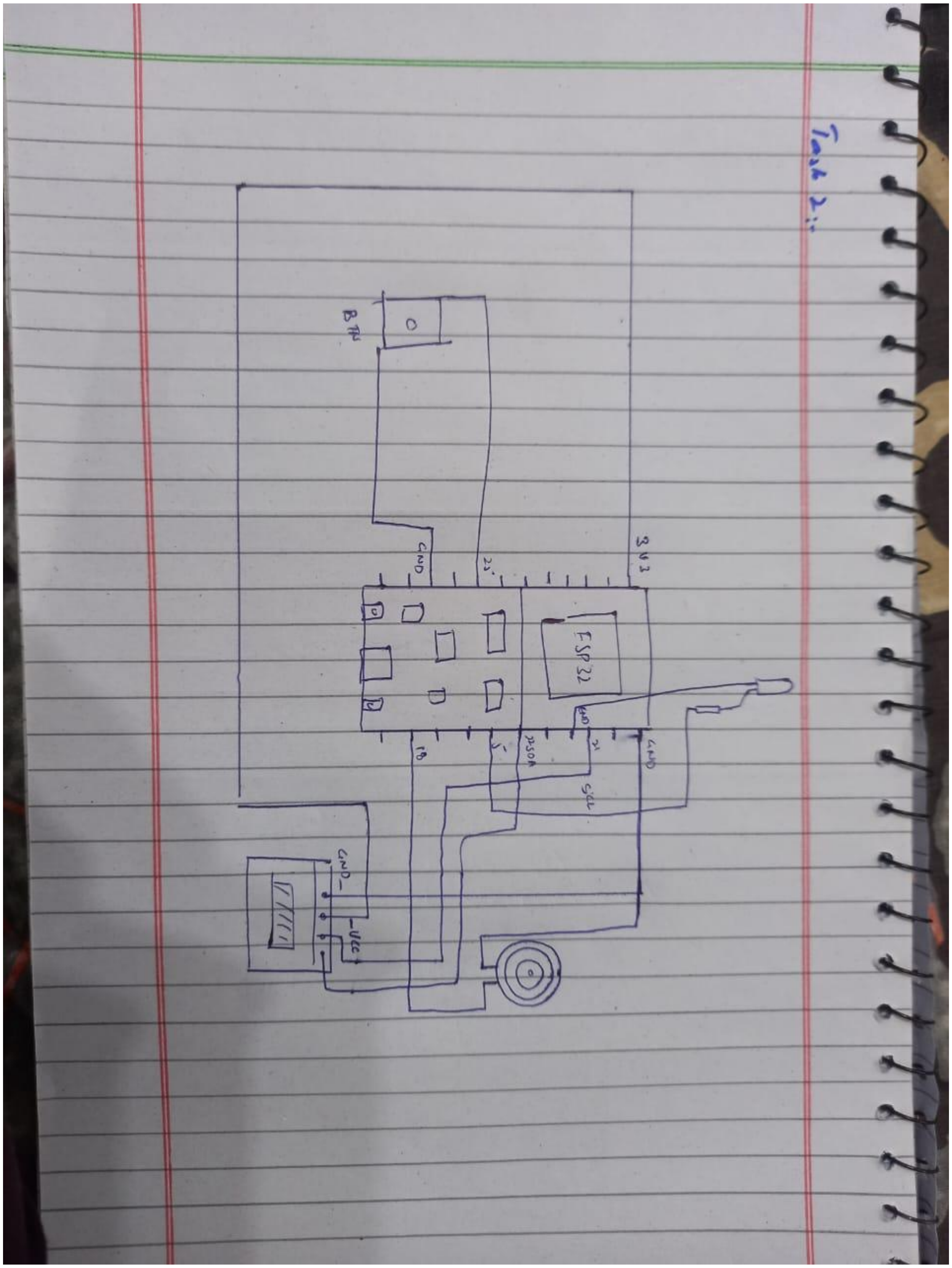


Task2- Code

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

# define  BTN     25
    "     LED     5
    "     BUZZER  18

Adafruit_SSD1306  display (128, 64, &wire, -1);

bool  ledState  = false;
unsigned  long  pressTime  = 0;
bool  pressed  =  false;

void   showText( string msg) {
     display. clearDisplay();
        " . setTextsize(1);
        " . setTextColor(White);
        " . setCursor( 0,20);
        " . println( msg);
        " . display ();
 }

void  setup() {
    pinMode( BTN  , INPUT_PULLUP);
       "   ( LED  , OUTPUT) ;
       "   ( BUZZER , OUTPUT);
    display.begin(SSD1306....,    0x3c);
    showText( "Ready... ");
}
```



```
void loop () {

 if (digitalRead (BTN) == Low & !pressed) {
      presed = True;
      presstime =   millis ();
 }


 if (digitalRead (BTN) == HIGH && pressed) {
    unsigned long duration = millis () - presstime;
    pressed = false;

 if (duration > 1500) {
    tone (BUZZER, 1000, 500);
    showText (" Long press → Buzzer");
 }

 else {

      ledstate = ! ledstate
      digitalWrite (LED, ledstate);
      showText ("Short press );
   }
  }
}
```


Task 2:
BTN
GND
25
3V3
ESP32
18
5
21
SCL
GND
GND
VCC

Wokwi links:

- Task1: https://wokwi.com/projects/445706554966735873
- Task2: https://wokwi.com/projects/445798744803145729

GitHub repository link:

https://github.com/mhamzafaisal/IOT-23-NTU-CS-B-1066.git