National Textile University, Faisalabad

Name:	**Fatima Shahzad**
Class:	**BSCS(A)**
Registration No:	**23-NTU-CS-1026**
Course Name:	**EMBEDDED IOT SYSTEM**
Submitted To:	**SIR NASIR MAHMOOD**

TASK A

FIXED BUTTON ISSUE

WOKWI LINK

https://wokwi.com/projects/445804295906521089

HAND WRITTEN CODE

Assignment No 1

TASK : A

Fixed Button ISSUE

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

// configuration constants ....
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_ADDR 0x3C

// Pin configuration
  Const int LED1_PIN = 18;
  Const int LED2_PIN = 17;
  const int LED3_PIN = 16;
  const int BTN_MODE = 34;
  Const int BTN_RESET = 35;
  const int BUZZER_PIN = 25;

# Debounce constants
constant unsigned long DEBOUNCE_DELAY = 25;
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
                         &Wire, -1);
enum Mode {BOTH_OFF, ALT_BLINK, BOTH_ON, PWM_FADE};
Mode currentMode = BOTH_OFF;
```

```
// Timer & state variables
unsigned long lastToggle = 0;
const unsigned long blinkInterval = 400;

// Button Debounce variables
unsigned long lastModePress = 0;
unsigned long lastResetPress = 0;
int lastModeState = HIGH;
int lastResetState = HIGH;
bool modePressed = false;
bool resetPressed = false;

int fadeVal = 0;
int fadeDir = 1;

// function Declarations ----
void setLedsToOff();
void setLedsToHigh();
void updateOLED(const char *msg);
void handleModeChange();

void setup(){
  Serial.begin(115200);
  Pinmode(LED1_PIN, OUTPUT);
  Pinmode(LED2_PIN, output);
  Pinmode(LED3_PIN, output);
  Pinmode(BTN_Mode, Input_PULLUP);
```

```
Pinmode(BTN-RESET, Input PULLUP);
Pinmode(BUZZER-PIN, OUTPUT);
 void setup(){
 wire.Begin(21,22);
 If(!display.Begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)){
  Serial.printIN("OLED failed!");
  for(;;);
}

updateOLED("BOTH OFF");
Set LEDS To off();
Serial.println("setup complete");
}

Void loop(){
  unsigned long now = millis();

// Mode Button Handler
 int readingmode = Digital Read(BTN_MODE);
if(readingmode != last modestate){
  last mode press = now;
}
  If((now - last Mode Press) > DEBOUNCE_DELAY){
 If(readingMode == LOW && b modePress){
   modepressed = true;
  digitalwrite(LED3_PIN, HIGH);
```

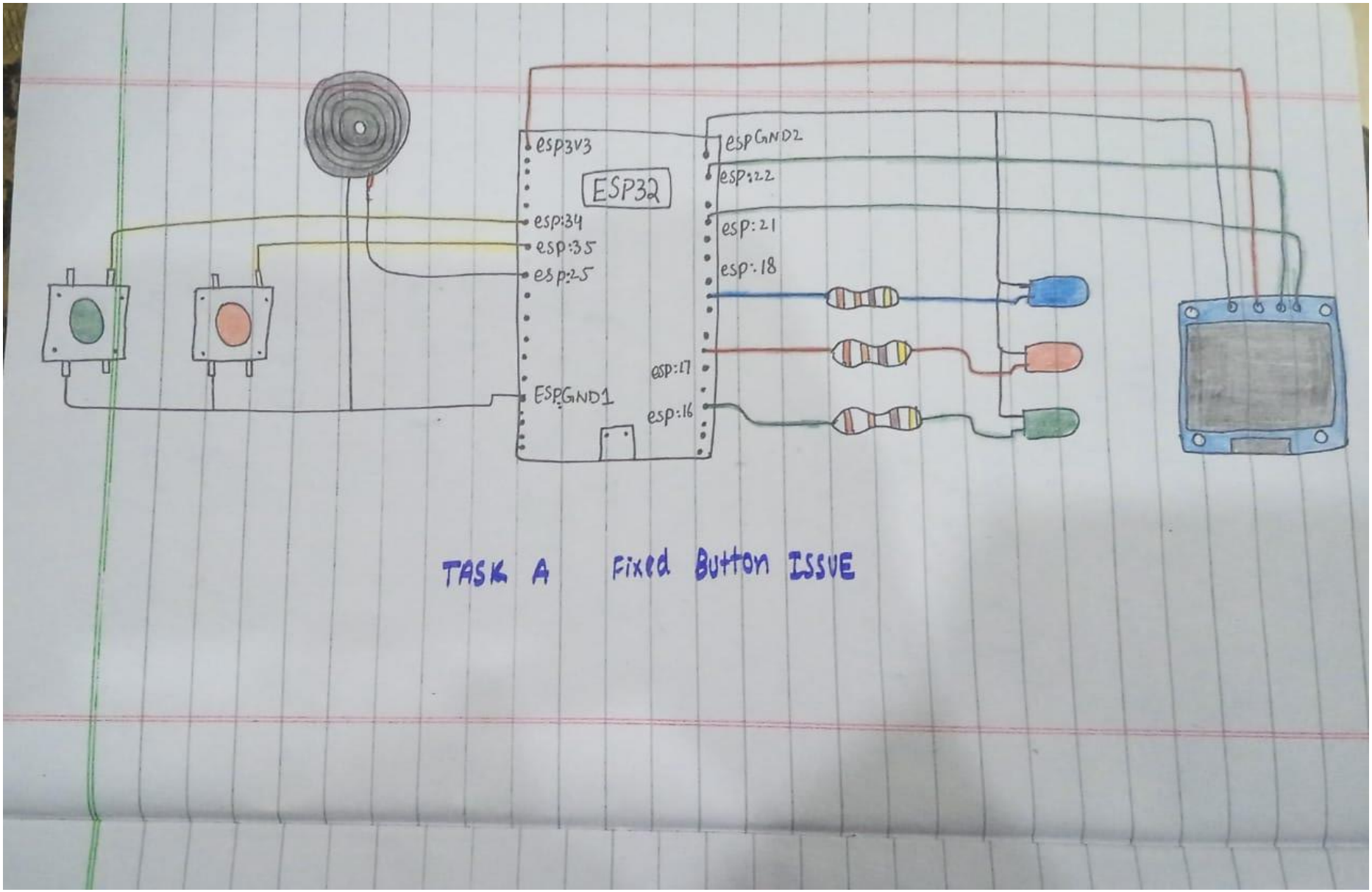
```
HandleModeChange();
 Serial.println("mode button pressed");
}

else if(readingmode == HIGH){
     modepressed = false;
     digitalWrite(LED3_PIN,LOW);
}
}

last ModeState = readingmode;

// RESET BUTTON HANDLER
int ReadingReset = (low && !Reset Pressed){
   resetpress = true;
   digitalwrite(LED3_PIN High);
   CurrentMode = BOTH OFF;
   updateOLED("BOTH OFF");
   setLEDtooff();
   serial.println("Reset button Pressed");
}
  else if(reading reset == HIGH){
       resetpressed = false;
     digitalWrite(LED3_PIN,LOW);
}
}
```

```
lastResetstate = readingreset;

// LED behavior based on currentMode

Switch (currentmode){
   Case BOTH_OFF;
    break;

  Case ALT_BLINK;
  If (now - last Toggle >= blinkInterval){
  last Toggle = now;
  Static bool toggle = false;
  toggle = ! toggle;

  analogwrite (LEDLPIN, toggle ? 255 : 0);
  analogwrite (LED2-PIN, toggle ? 0 : 255);
  analogwrite (LED3_PIN LOW);
}
  break;
  Case BOTH_ON;
   break;
  case PWM FADE:
  fadeval += fadeDir * 1;
  If (fadeval >= 180){ fadeval = 180 ; fadeDir = -1;}
  If (fadeval >= 0) {fade val = 0; fadedir = 1;}
```

```
analogWrite(LED1PIN, fadeval);
analogWrite(LED2PIN, fadeval);
digitalWrite(LED3_PIN, LOW);

    delay(20);
    break;
}
}

// HELPER function
void setLEDsToOff(){
  analogWrite(LED1_PIN, 0);
  analogWrite(LED2_PIN, 0);
  analogWrite(LED3_PIN, LOW);
}
void setLedsToHigh(){
  analogWrite(LED1_PIN, 255);
  analogWrite(LED2_PIN, 205);
  digitalWrite(LED3_PEN, HIGH);
}
void handleModeChange(){
  currentMode = (Mode)((currentMode + 1) % 4);
  switch (currentMode){
    case BOTH_OFF:
      updateOLED("BOTH OFF");
    setLEDsToOff();
    break;
```

```
case ALT-BLINK:
  updateOLED("ALT BLINK");
  SetLEDTo off();
  last Toggle = Millis();
  break;
 case BOTH ON:
  updateOLED("BOTH_ON");
 SetOLEDS To High();
 break;
  case PWM-FADE_FADE:
    updateOLED("PWM FADE");
    fadeval = 0;
    fadedir = 1;
    break;
  }
}
  void updateOLED(const char *msg){
   display. clearDisplay();
   display. clear setText size(2);
  display. set Text color(SSD1306_WHITE);
  display. set cursor(0,20);
   display. printIn(msg);
    display. display();
}
```

TASK A **DIAGRAM**



TASK A Fixed Button ISSUE

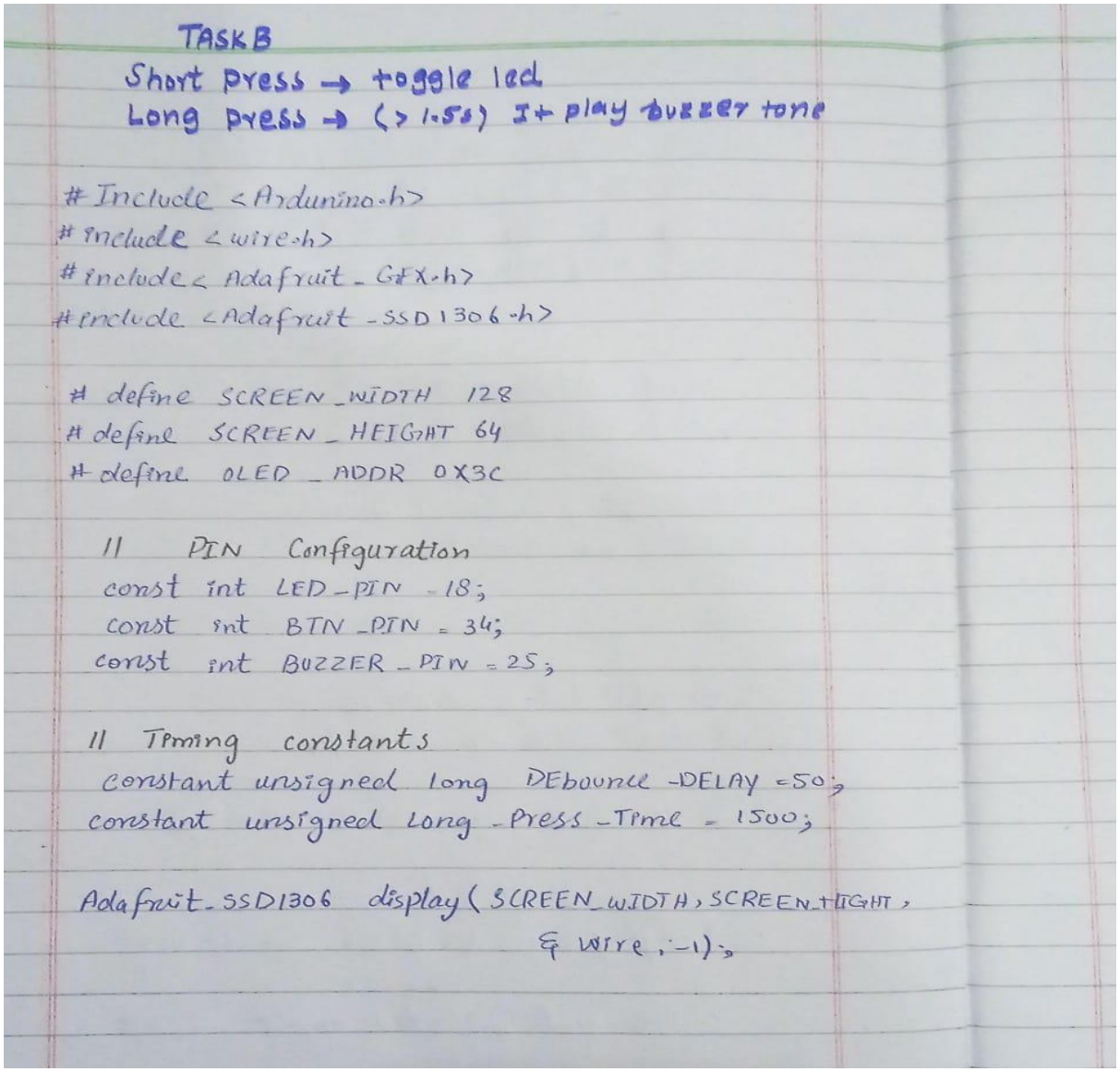
TASK B

SHORT PRESS -> TOGGLE LED

LONG PRESS -> (1.5S) IT PLAY BUZZER TONE

WOKWI LINK

https://wokwi.com/projects/445805777808820225



TASK B
Short press → toggle led
Long press → (> 1.5s) It play buzzer tone

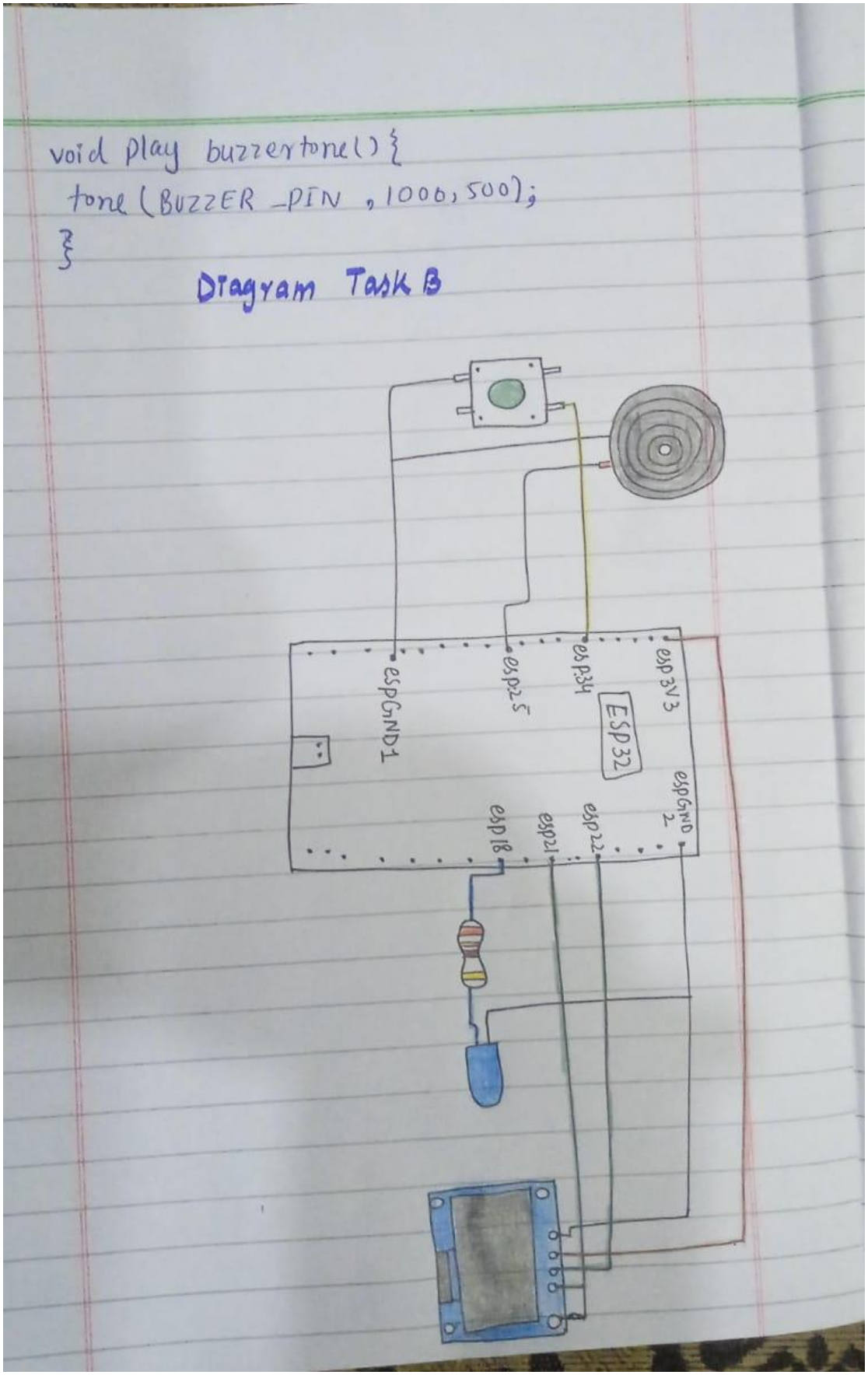
```
#Include <Arduino.h>
#include <wire.h>
#include< Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

# define SCREEN_WIDTH 128
# define SCREEN_HEIGHT 64
# define OLED_ADDR 0X3C

// PIN Configuration
const int LED_PIN = 18;
const int BTN_PIN = 34;
const int BUZZER_PIN = 25;

// Timing constants
constant unsigned long DEbounce-DELAY =50;
constant unsigned long Press_Time = 1500;

Adafruit_SSD1306 display( SCREEN_WIDTH, SCREEN_HIGHT,
                          & Wire, -1);
```

```
// Button state variable

bool buttonPressed = false;
bool ledState = false;
unsigned long pressStartTime = 0;
bool longPressTriggered = false;

// Function declarations
void updateOLED(const char *msg);
void playBuzzerTone();

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

// SET LED & Buzzer Pins

PinMode(LED_PIN, OUTPUT);
PinMode(BUZZER_Pin, OUTPUT);

PinMode(BTN_PIN, INPUT_PULLUP);

Wire.begin(21,22)
If (display.begin(SSD1306_SwitchCAPVCC, OLED, ADDR)
){

Serial.println("setup complete - TaskB");
}
```

```
void loop() {
  unsigned long now = millis();
  int reading = digitalRead(BTN_PIN);

  If( reading == LOW && !buttonPressed ){
   buttonPressed = true;
   pressstartTime = now;
   longPressTriggered = false;
    Serial.println ( "Button Pressed");
  }

  If( buttonpressed & realing == low && !.
                    longpressTriggered){

  If ((now - pressstationtime )>=-now pressTime)
  {
     playBuzzer tone ();
     updateOLED ("BuzzerON");
     Serial. printIn("long press detected");
  }
}

  If(reading == HIGH && buttonpressed){
  unsigned long pressbutton = now -pressstartTime)
  {
     !edstate =! LEDstate;
  digitalwrite (LED_PIN, let state);
```

```
If (Led state){
  updateOLED("LED ON");
  Serial.printIn ("short press LED ON"
} else{
    updateOLED ("LED OFF");
  serial.printIN ("short press.Ledoff")
}

  If (ledstate){
      updateOLED ("LED ON");
} else {
      updateOLED("LED OFF");
}
}
  buttonpressed = false;
  delay (DEBOUNCE_DELAY);
}
}

  void updatedOLED (const char *msg){
    display.clearDisplay();
    display.SetTextsize(2);
    display.setTextcolor (SSD1306_WHITE);
    display.setCursor(0,15);
    display.PrintIn(msg);
    display.display();
}
```

TASK B DIAGRAM



GITHUB link

https://github.com/fatimaghick/1026-_-EMBEDDED-IOT-SYSTEM-.git