Task - A

```
// Library
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// OLED
#define Screen_Width 128
#define Screen_Height 64
Adafruit_SSD1306 display(Screen_Width, Screen_Height, &Wire, -1);

// LED
#define Led1 19
#define Led2 18
#define Led3 5
// BTN
#define btn_Mode 25
#define btn_Reset 26

// Bools
Volatile int mode = 0;
Volatile bool modeChanged = false;
Volatile bool resetPressed = false;

// Timer
hw_timer_t * debounceTimer = NULL;
volatile bool debounce_Active = false;
```

// PWM

```
#define PWM_CH 0
#define FREQ 5000
#define RES 8

int brightness = 0;
int fadeAmount = 15;
unsigned long previousMillis = 0;
unsigned long lastFade = 0;
bool ledState = LOW;
```

// Methods

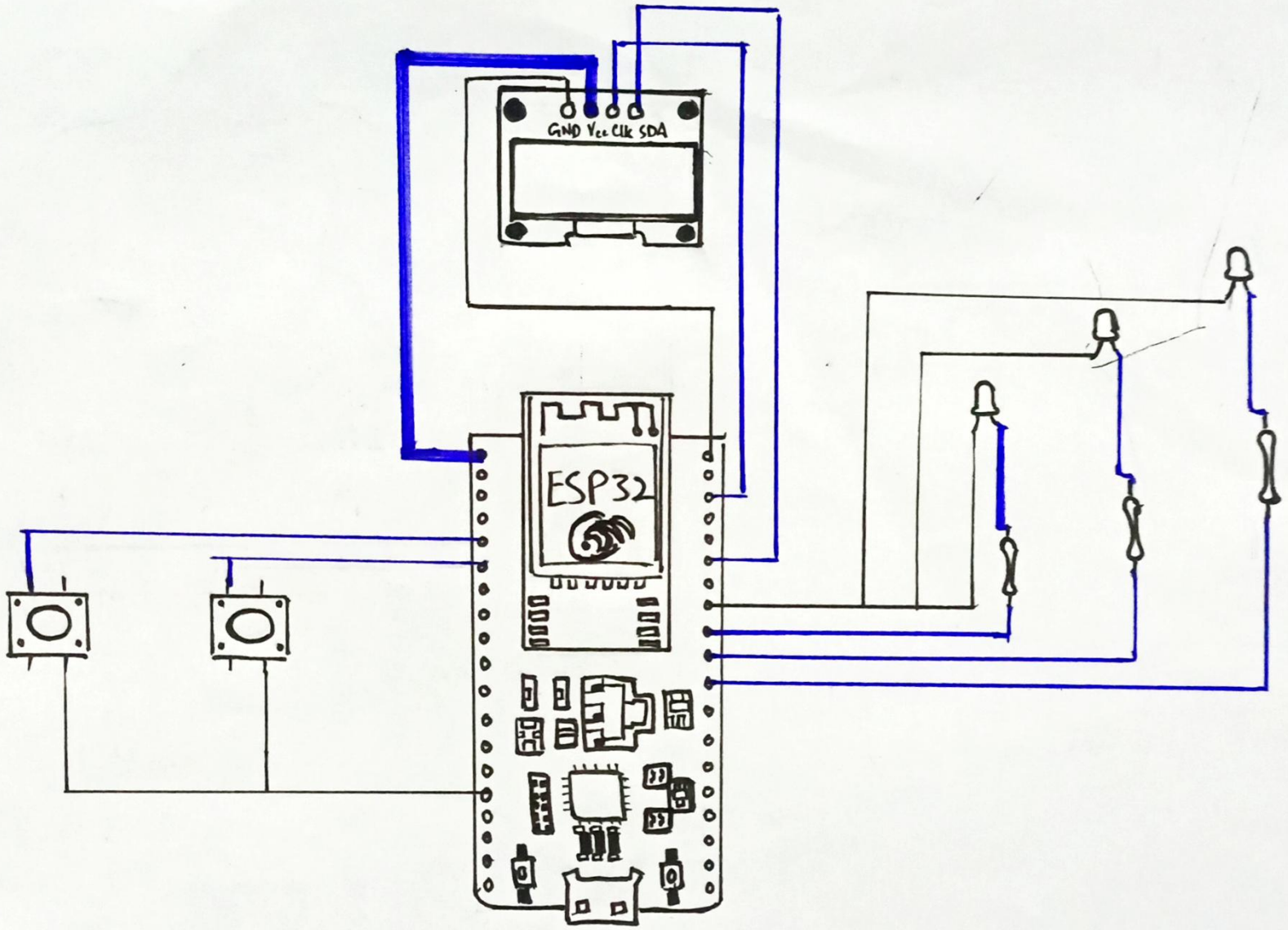
```
void IRAM_ATTR resetDebounce()
{
    debounceActive = false;
    timerAlarmDisable( debounceTimer );
    timerWrite (debounceTimer, 0);
}

void IRAM_ATTR BTN_Pressed_Mode()
{
    if(debounceActive)
  {  return; }
    debounceActive = True;
    timerWrite (debounceTimer, 0);
    timerAlarmWrite ( debounceTimer, 2000000, false);
    timerAlarmEnable ( debounceTimer);
    Mode Changed = True;
}
```

```
void IRAM_ATTR BTN_Pressed_Reset()
{
  if (debounceActive)
  { return; }
   debounceActive = True;
   timerWrite ( debounceTimer, 0);
   timerAlarmWrite( debounceTimer, 2000000, false);
   timer Enable (debounceTimer);
   resetPressed = true;
}

void showMode (const char* Text)
{
   display.clearDisplay();
   display.setTextSize(2);
   display.setTextColor(SSD1306_White);
   display.setCursor((screen_Width-strlen(text)x12)/2, (screen_Heigh-16)/2)
   display.print(Text)
   display.display();

Void setup()
{
   if (!display.begin(SSD1306_SWITCHAPVCC, 0x3C))
   { for(;;); }

  dispinMode (Led1, Output);
   pinMode( Led2, Output);
   pinMode (Led3, Output);
   pinMode (BTN_Mode, Input_PULLUP);
   pinMode (BTN_Reset, Input_PULLUP);
```

```
digitalWrite (Led1, LOW);
digitalWrite (Led2, LOW);
digitalWrite (Led3, LOW);

attachInterrupt(digitalPinToInterrupt(BTN_Mode), BTN_Pressed_Mode, Falling);
attachInterrupt(digitalPinToInterrupt(BTN_Reset), BTN_Pressed_Resit, Falling);

debounceTimer = TimerBegin(0, 80, true);
timerAttachInterrupt(debounceTimer, &resetDebounce, true);
timerAlarmWrite(debounceTimer, 2000000, false);
timerAlarmDisable(debounceTimer);

LedcSetup(PWM_CH, FREQ, RES);
LedAttachPin(Led3, PWM_CH);

}

Void Loop()
{
  if (resetPressed)
  {
    mode = 0;
    digitalWrite(Led1, LOW);
    digitalWrite(Led2, LOW);
    resetPressed = false;
    showMode("Default");
  }
```

```
if (modeChanged)
{
  mode ++;
  if (mode > 4)  mode = 0;
  modeChanged = false;
}


switch (mode) {
   case 0:
      showMode("Default");
      digitalWrite(Led1, LOW);
      digitalWrite(Led2, LOW);
      break;

   case 1:
      showMode("Both OFF");
      digitalWrite(Led1, LOW);
      digitalWrite(Led2, LOW);
      break;

   case 2:
      showMode("Blinking");
      if (millis() - previousMillis >= 500)
      { previousMillis = Millis();
        ledState = !ledState;
        digitalWrite(led1, ledState);
        digitalWrite(Led2, !ledState);
      }
      break;
```

```
Case 3:
    showMode("Both ON");
    digitalWrite(Led1, HIGH);
    digitalWrite(Led2, HIGH);
    break;

Case 4:
    showMode("PWM");
    digitalWrite(Led1, LOW);
    digitalWrite(Led2, LOW);
    if (millis() - lastFade >= 2) {
      lastFade = millis();
      LedcWrite(PWM_CH, brightness);
      brightness += fadeAmount;
      if (brightness <= 0 || brightness >= 255)
        { fadeAmount = -fadeAmount; }
    break;

  }

}
```


GND Vcc Clk SDA
ESP32