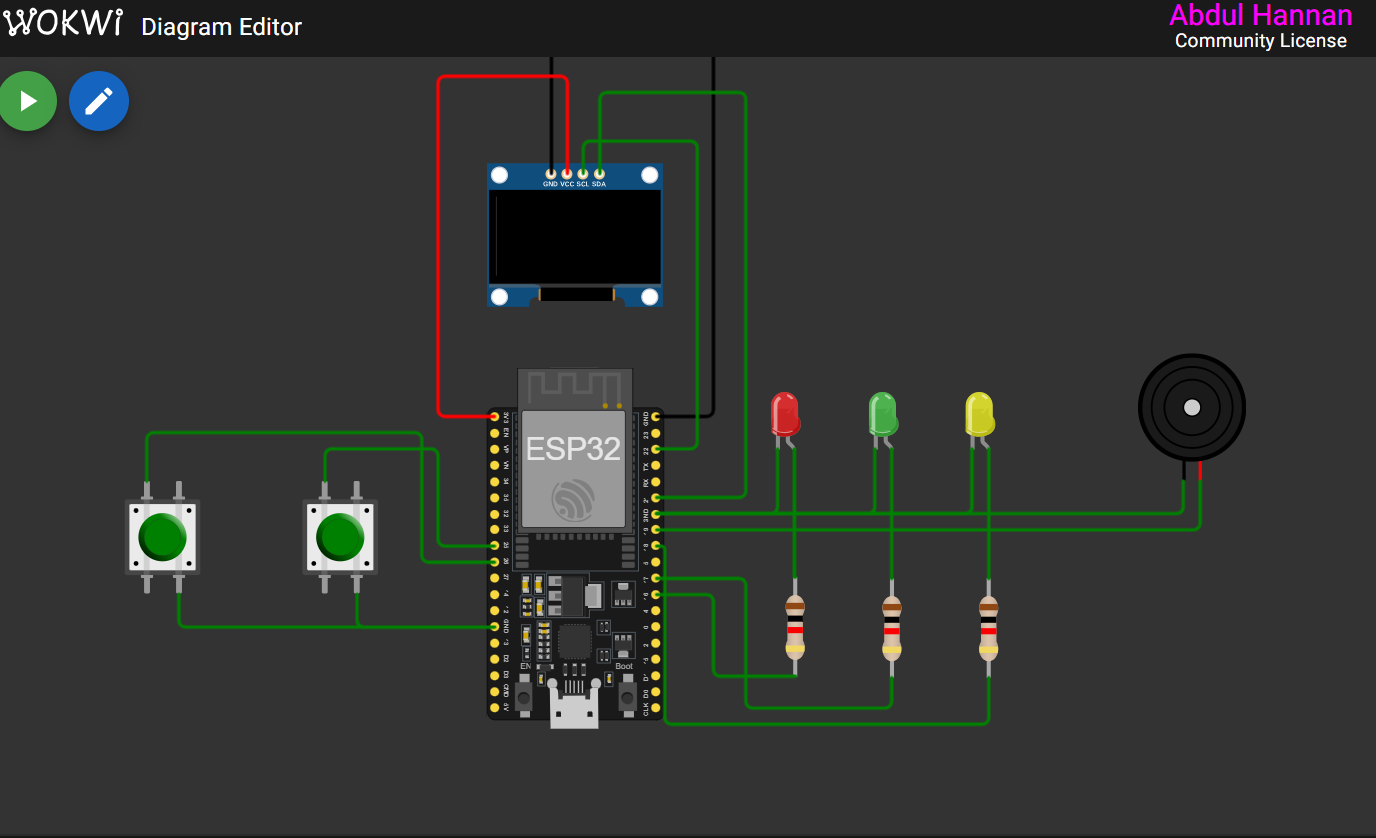
|  |  |
| --- | --- |
| Name | Abdul Hannan Ibrahim |
| Registration # | 23-NTU-CS-1002 |
| Assignment# | 1 |
| Course Name | Embedded IOT Systems |
| Section | BSCS-5th-A |
| Department | Computer Science |
| Submit To | Mr. Nasir Mehmood |
| Date | 23-09-2025 |

A picture containing text, tableware, plate, dishware

Description automatically generated

**Circuit:**

****

**Task 1:**

**Code:**

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_RESET    -1

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

// LEDs

#define LED\_YELLOW 18

#define LED\_GREEN 17

#define LED\_RED 16

// Buttons

#define BTN\_CYCLE 25

#define BTN\_RESET 26

// PWM fade

int fadeValue = 0;

int fadeDirection = 1;

// Button debounce

const unsigned long DEBOUNCE\_MS = 30;

int lastReadingCycle = HIGH, stableCycle = HIGH, lastStableCycle = HIGH;

unsigned long lastDebounceCycle = 0;

int lastReadingReset = HIGH, stableReset = HIGH, lastStableReset = HIGH;

unsigned long lastDebounceReset = 0;

// LED mode: 0=OFF, 1=ALT BLINK, 2=ALL ON, 3=PWM FADE

int ledMode = 0;

bool blinkState = false;

unsigned long lastBlinkTime = 0;

const unsigned long BLINK\_INTERVAL = 500;

const unsigned long FADE\_INTERVAL = 10;

unsigned long lastFadeTime = 0;

void setup() {

  pinMode(BTN\_CYCLE, INPUT\_PULLUP);

  pinMode(BTN\_RESET, INPUT\_PULLUP);

  pinMode(LED\_YELLOW, OUTPUT);

  pinMode(LED\_GREEN, OUTPUT);

  pinMode(LED\_RED, OUTPUT);

  // OLED init

  if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) { for(;;); }

  display.clearDisplay();

  display.setTextSize(2);

  display.setTextColor(SSD1306\_WHITE);

  // Initial LED off

  analogWrite(LED\_YELLOW, 0);

  analogWrite(LED\_GREEN, 0);

  analogWrite(LED\_RED, 0);

}

void loop() {

  unsigned long currentMillis = millis();

  // -------- Handle Cycle Button --------

  int readingCycle = digitalRead(BTN\_CYCLE);

  if (readingCycle != lastReadingCycle) lastDebounceCycle = currentMillis;

  if (currentMillis - lastDebounceCycle >= DEBOUNCE\_MS) {

    if (stableCycle != readingCycle && lastStableCycle == HIGH && readingCycle == LOW) {

      ledMode++;

      if (ledMode > 3) ledMode = 0;

      fadeValue = 0; fadeDirection = 1; blinkState = false;

      lastBlinkTime = currentMillis; lastFadeTime = currentMillis;

    }

    stableCycle = readingCycle;

  }

  lastReadingCycle = readingCycle;

  lastStableCycle = stableCycle;

  // -------- Handle Reset Button --------

  int readingReset = digitalRead(BTN\_RESET);

  if (readingReset != lastReadingReset) lastDebounceReset = currentMillis;

  if (currentMillis - lastDebounceReset >= DEBOUNCE\_MS) {

    if (stableReset != readingReset && lastStableReset == HIGH && readingReset == LOW) {

      ledMode = 0;

      analogWrite(LED\_YELLOW, 0);

      analogWrite(LED\_GREEN, 0);

      analogWrite(LED\_RED, 0);

      fadeValue = 0; fadeDirection = 1; blinkState = false;

      lastBlinkTime = currentMillis; lastFadeTime = currentMillis;

    }

    stableReset = readingReset;

  }

  lastReadingReset = readingReset;

  lastStableReset = stableReset;

  // -------- LED modes --------

  switch(ledMode){

    case 0: // All OFF

      analogWrite(LED\_YELLOW, 0);

      analogWrite(LED\_GREEN, 0);

      analogWrite(LED\_RED, 0);

      break;

    case 1: // Alternate blink

      if (currentMillis - lastBlinkTime >= BLINK\_INTERVAL) {

        blinkState = !blinkState;

        lastBlinkTime = currentMillis;

      }

      analogWrite(LED\_YELLOW, blinkState ? 255 : 0);

      analogWrite(LED\_GREEN, blinkState ? 0 : 255);

      analogWrite(LED\_RED, blinkState ? 255 : 0);

      break;

    case 2: // All ON

      analogWrite(LED\_YELLOW, 255);

      analogWrite(LED\_GREEN, 255);

      analogWrite(LED\_RED, 255);

      break;

    case 3: // PWM fade

      if (currentMillis - lastFadeTime >= FADE\_INTERVAL) {

        fadeValue += fadeDirection \* 5;

        if (fadeValue >= 255) { fadeValue = 255; fadeDirection = -1; }

        if (fadeValue <= 0)   { fadeValue = 0; fadeDirection = 1; }

        analogWrite(LED\_YELLOW, fadeValue);

        analogWrite(LED\_GREEN, fadeValue);

        analogWrite(LED\_RED, fadeValue);

        lastFadeTime = currentMillis;

      }

      break;

  }

  // -------- OLED display --------

  display.clearDisplay();

  display.setCursor(0,20);

  switch(ledMode){

    case 0: display.println("ALL OFF"); break;

    case 1: display.println("BLINK"); break;

    case 2: display.println("ALL ON"); break;

    case 3: display.println("PWM FADE"); break;

  }

  display.display();

}

**Task 2:**

Code:

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_RESET    -1

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

#define BUTTON\_PIN 25

#define LED\_PIN    18

#define BUZZER\_PIN 15

const unsigned long LONG\_PRESS\_MS = 1500;

const unsigned long DEBOUNCE\_MS = 50;

int lastButtonState = HIGH;

int stableState = HIGH;

unsigned long lastDebounce = 0;

unsigned long pressStart = 0;

void setup() {

  pinMode(BUTTON\_PIN, INPUT\_PULLUP);

  pinMode(LED\_PIN, OUTPUT);

  pinMode(BUZZER\_PIN, OUTPUT);

  if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) for(;;);

  display.clearDisplay();

  display.setTextSize(2);

  display.setTextColor(SSD1306\_WHITE);

  digitalWrite(LED\_PIN, LOW);

  digitalWrite(BUZZER\_PIN, LOW);

}

void loop() {

  unsigned long currentMillis = millis();

  int reading = digitalRead(BUTTON\_PIN);

  // Debounce

  if (reading != lastButtonState) lastDebounce = currentMillis;

  if (currentMillis - lastDebounce >= DEBOUNCE\_MS) {

    if (stableState != reading) {

      stableState = reading;

      if (reading == LOW) pressStart = currentMillis;  // button pressed

      else {  // button released

        unsigned long pressDuration = currentMillis - pressStart;

        display.clearDisplay();

        display.setCursor(0,20);

        if (pressDuration < LONG\_PRESS\_MS) {

          // Short press → toggle LED

          digitalWrite(LED\_PIN, !digitalRead(LED\_PIN));

          display.println("Short Press");

        } else {

          // Long press → buzzer

          tone(BUZZER\_PIN, 1000, 500);

          display.println("Long Press");

        }

        display.display();

      }

    }

  }

  lastButtonState = reading;

}