A picture containing text, tableware, plate, dishware

Description automatically generated

**Embedded Iot**

**Name:** Abdul Mannan Ibrahim **ROLL#:** 23-NTU-CS-1003

**Assignment:**1 **Course Name:** Embedded IoT

**Semester:** 5th **Section:** BSCS-A

**Department:** DCS **Date:** 26 OCT,2025

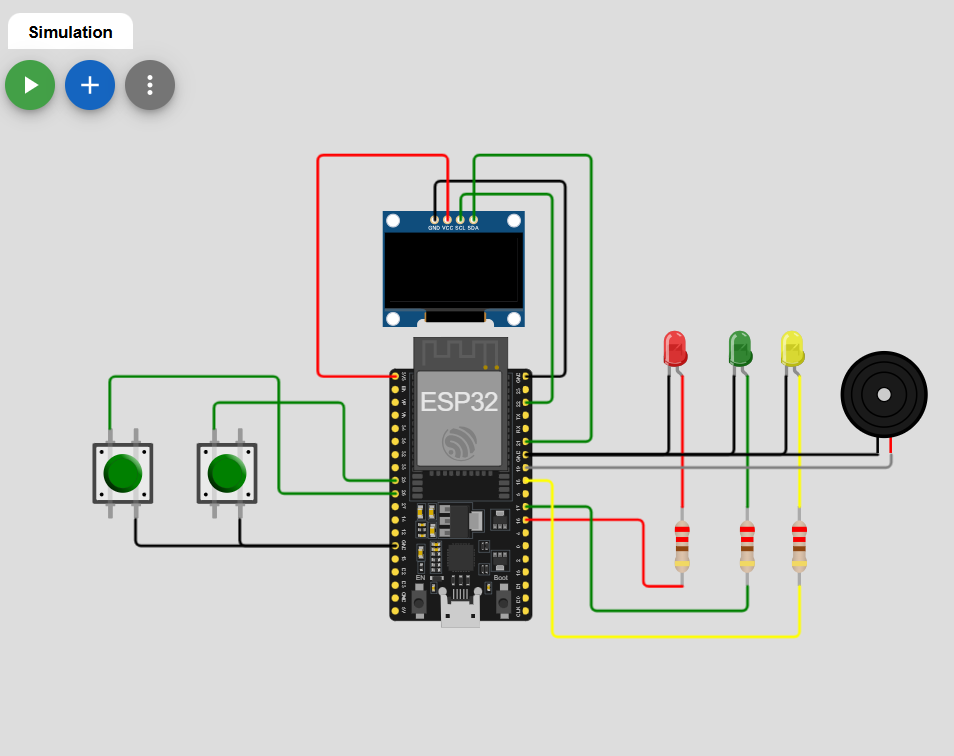
****

**Question 3----Implementation**

**Circuit-Diagram**

**Draw a Wokwi circuit and draw a neat hand-sketch including**

* **2 push buttons**
* **3 LEDs**
* **1 Buzzer**
* **1 OLED**



**Task A—Coding**

**Use On Buttonto cycle through LED Models(Display the current state on OLED):**

* **Both OFF**
* **Alternate Blink**
* **Power ON**
* **PWM Fade**

**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);

// LED pins

#define LED\_YELLOW 18

#define LED\_GREEN 17

#define LED\_RED 16

// LED PWM channels (for ESP32)

#define CH\_YELLOW 0

#define CH\_GREEN 1

#define CH\_RED 2

#define PWM\_FREQ 5000

#define PWM\_RES 8 // 8-bit resolution (0-255)

// Button pins

#define BTN\_CYCLE 25

#define BTN\_RESET 26

// Button debounce

int stableButtonState = HIGH;

int lastReading = HIGH;

int lastStableState = HIGH;

unsigned long lastDebounceTime = 0;

const unsigned long DEBOUNCE\_MS = 30;

int stableResetState = HIGH;

int lastResetReading = HIGH;

int lastStableResetState = HIGH;

unsigned long lastResetDebounceTime = 0;

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

int ledMode = 0;

// Blink timing

unsigned long lastBlinkTime = 0;

const unsigned long BLINK\_INTERVAL = 500;

bool blinkState = false;

// PWM fade

int fadeValue = 0;

int fadeDirection = 1;

unsigned long lastFadeTime = 0;

const unsigned long FADE\_INTERVAL = 10;

void setup() {

pinMode(BTN\_CYCLE, INPUT\_PULLUP);

pinMode(BTN\_RESET, INPUT\_PULLUP);

// Initialize OLED

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

for (;;);

}

display.clearDisplay();

display.setTextSize(2);

display.setTextColor(SSD1306\_WHITE);

// Initialize button states

lastReading = digitalRead(BTN\_CYCLE);

stableButtonState = lastReading;

lastStableState = stableButtonState;

lastResetReading = digitalRead(BTN\_RESET);

stableResetState = lastResetReading;

lastStableResetState = stableResetState;

// Attach PWM channels

ledcSetup(CH\_YELLOW, PWM\_FREQ, PWM\_RES);

ledcAttachPin(LED\_YELLOW, CH\_YELLOW);

ledcSetup(CH\_GREEN, PWM\_FREQ, PWM\_RES);

ledcAttachPin(LED\_GREEN, CH\_GREEN);

ledcSetup(CH\_RED, PWM\_FREQ, PWM\_RES);

ledcAttachPin(LED\_RED, CH\_RED);

// Initial LEDs off

ledcWrite(CH\_YELLOW, 0);

ledcWrite(CH\_GREEN, 0);

ledcWrite(CH\_RED, 0);

}

void loop() {

unsigned long currentMillis = millis();

// -------- Handle BTN\_CYCLE --------

int reading = digitalRead(BTN\_CYCLE);

if (reading != lastReading) {

lastDebounceTime = currentMillis;

lastReading = reading;

}

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

if (stableButtonState != reading) {

stableButtonState = reading;

if (lastStableState == HIGH && stableButtonState == LOW) {

ledMode++;

if (ledMode > 3) ledMode = 0;

fadeValue = 0;

fadeDirection = 1;

blinkState = false;

lastBlinkTime = currentMillis;

lastFadeTime = currentMillis;

}

lastStableState = stableButtonState;

}

}

// -------- Handle BTN\_RESET --------

int resetReading = digitalRead(BTN\_RESET);

if (resetReading != lastResetReading) {

lastResetDebounceTime = currentMillis;

lastResetReading = resetReading;

}

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

if (stableResetState != resetReading) {

stableResetState = resetReading;

if (lastStableResetState == HIGH && stableResetState == LOW) {

ledMode = 0;

// Turn off all LEDs

ledcWrite(CH\_YELLOW, 0);

ledcWrite(CH\_GREEN, 0);

ledcWrite(CH\_RED, 0);

fadeValue = 0;

fadeDirection = 1;

blinkState = false;

lastBlinkTime = currentMillis;

lastFadeTime = currentMillis;

}

lastStableResetState = stableResetState;

}

}

// -------- Handle LED modes --------

switch (ledMode) {

case 0: // ALL OFF

ledcWrite(CH\_YELLOW, 0);

ledcWrite(CH\_GREEN, 0);

ledcWrite(CH\_RED, 0);

break;

case 1: // ALL ON

ledcWrite(CH\_YELLOW, 255);

ledcWrite(CH\_GREEN, 255);

ledcWrite(CH\_RED, 255);

break;

case 2: // BLINK

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

blinkState = !blinkState;

lastBlinkTime = currentMillis;

}

ledcWrite(CH\_YELLOW, blinkState ? 255 : 0);

ledcWrite(CH\_GREEN, blinkState ? 0 : 255);

ledcWrite(CH\_RED, blinkState ? 255 : 0);

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; }

ledcWrite(CH\_YELLOW, fadeValue);

ledcWrite(CH\_GREEN, fadeValue);

ledcWrite(CH\_RED, fadeValue);

lastFadeTime = currentMillis;

}

break;

}

// -------- Update OLED --------

display.clearDisplay();

display.setCursor(0, 20);

switch (ledMode) {

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

case 1: display.println("ALL ON"); break;

case 2: display.println("BLINKING"); break;

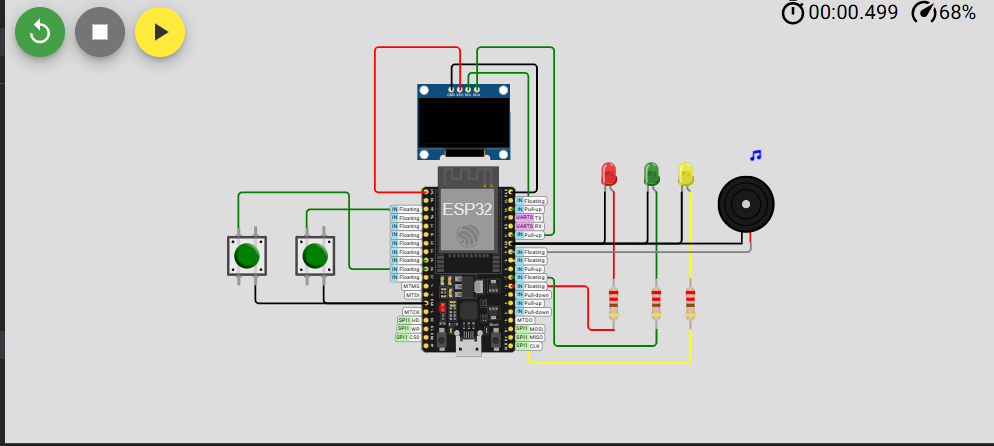
case 3: display.println("PWM FADE"); break;

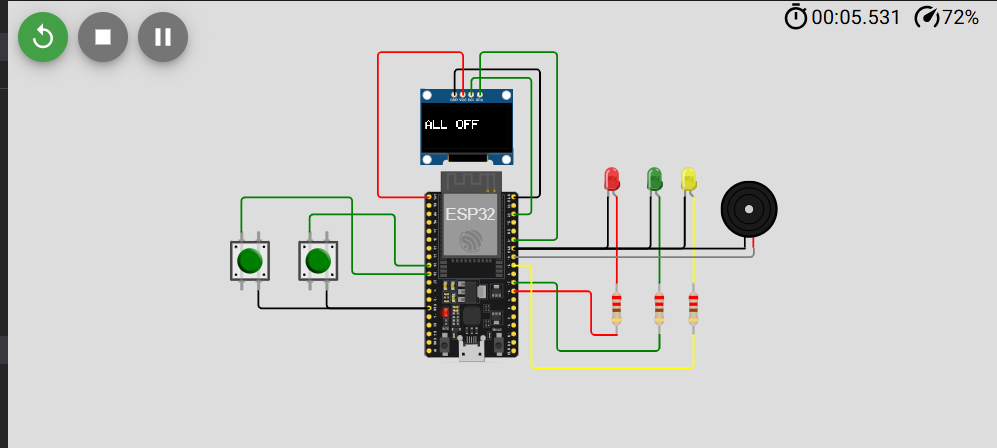
}

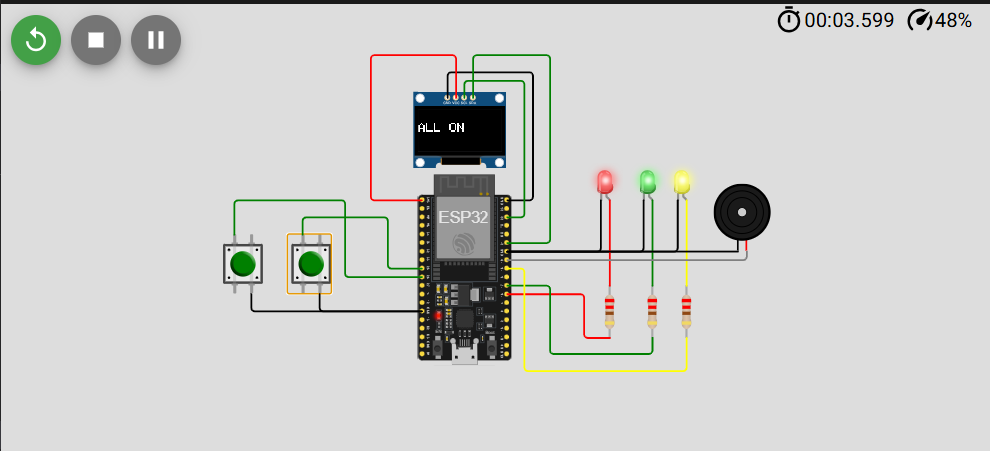
display.display();

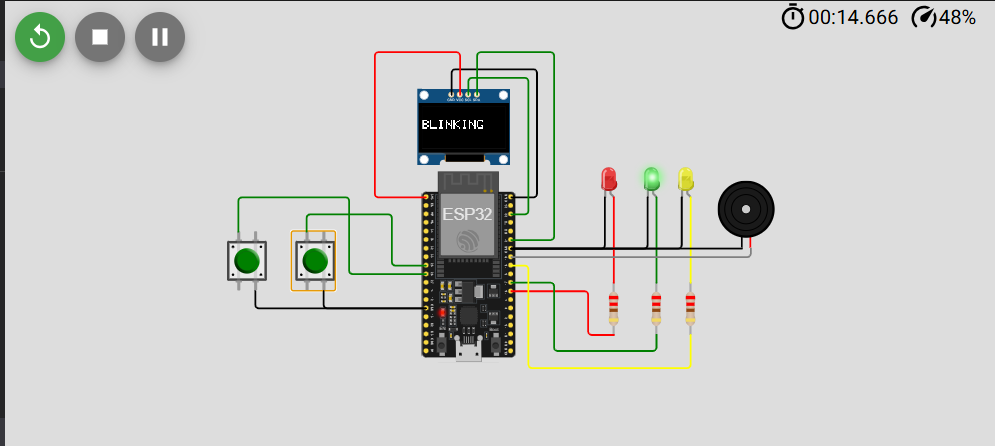
}

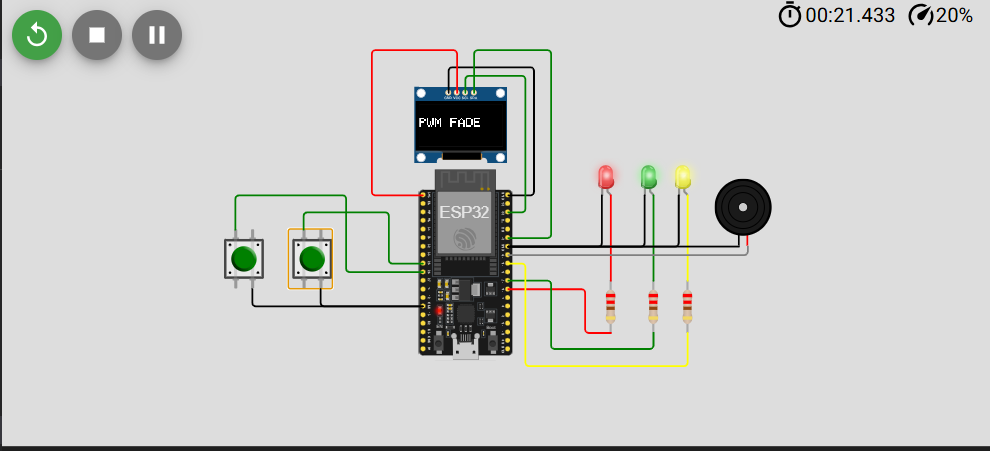
**Output:**

****

****

****

****

****

**Task B**

**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);

// --- Pin Definitions ---

#define BTN\_PIN 25

#define LED\_PIN 18

#define BUZZER 19

// --- LED PWM Setup (ESP32) ---

#define CH\_LED 0

#define PWM\_FREQ 5000

#define PWM\_RES 8 // 8-bit (0–255)

// --- Debounce and Press Detection ---

bool lastButtonState = HIGH;

bool buttonPressed = false;

unsigned long pressStartTime = 0;

unsigned long lastDebounceTime = 0;

const unsigned long DEBOUNCE\_MS = 30;

const unsigned long LONG\_PRESS\_MS = 1500;

// --- LED & Buzzer states ---

bool ledState = false;

bool buzzerOn = false;

unsigned long buzzerStartTime = 0;

const unsigned long BUZZER\_DURATION = 200;

void setup() {

pinMode(BTN\_PIN, INPUT\_PULLUP);

pinMode(BUZZER, OUTPUT);

digitalWrite(BUZZER, LOW);

// Setup LED PWM

ledcSetup(CH\_LED, PWM\_FREQ, PWM\_RES);

ledcAttachPin(LED\_PIN, CH\_LED);

ledcWrite(CH\_LED, 0);

// Setup OLED

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

for (;;);

}

display.clearDisplay();

display.setTextSize(2);

display.setTextColor(SSD1306\_WHITE);

display.setCursor(0, 20);

display.println("Ready");

display.display();

}

void loop() {

unsigned long currentMillis = millis();

bool reading = digitalRead(BTN\_PIN);

// Debounce button

if (reading != lastButtonState) {

lastDebounceTime = currentMillis;

lastButtonState = reading;

}

if ((currentMillis - lastDebounceTime) > DEBOUNCE\_MS) {

// Button pressed

if (!buttonPressed && reading == LOW) {

buttonPressed = true;

pressStartTime = currentMillis;

}

// Button released

if (buttonPressed && reading == HIGH) {

unsigned long pressDuration = currentMillis - pressStartTime;

buttonPressed = false;

if (pressDuration >= LONG\_PRESS\_MS) {

// --- Long Press ---

digitalWrite(BUZZER, HIGH);

buzzerOn = true;

buzzerStartTime = currentMillis;

display.clearDisplay();

display.setCursor(0, 20);

display.println("Long Press");

display.display();

} else {

// --- Short Press ---

ledState = !ledState;

ledcWrite(CH\_LED, ledState ? 255 : 0);

display.clearDisplay();

display.setCursor(0, 20);

display.println(ledState ? "LED ON" : "LED OFF");

display.display();

}

}

}

// Turn off buzzer after duration

if (buzzerOn && (currentMillis - buzzerStartTime >= BUZZER\_DURATION)) {

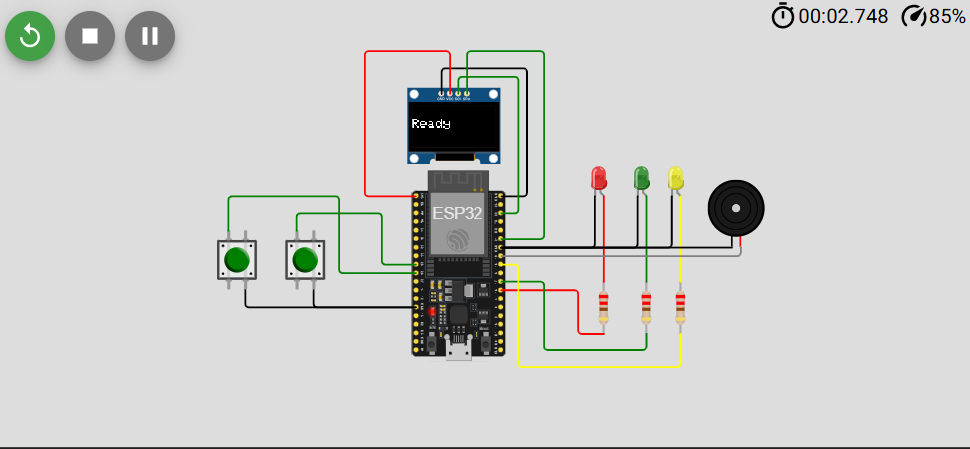
digitalWrite(BUZZER, LOW);

buzzerOn = false;

}

}

**Output:**

****

