**National Textile University, Faisalabad**



**Department of Computer Science**

|  |  |
| --- | --- |
| **Name:** | HAFIZA MINAHILSHABBIR |
| **Class:** | BSCS 5th\_A |
| **Registration No:** | 23-NTU-CS-1031 |
| **Course Name:** | Internet Of Things |
| **Submitted To:** | Sir Nasir |
| **Submission Date:** | 26/10/2025 |

**ASSIGNMENT QUESTION 03  
TASK A**

**Code:**

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define LED1 16

#define LED2 17

#define LED3 18

#define BTN\_MODE 32

#define BTN\_RESET 33

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

int mode = 0;

int brightness = 0;

int fadeAmount = 5;

// Timing variables for non-blocking delays

unsigned long previousBlinkMillis = 0;

unsigned long previousFadeMillis = 0;

unsigned long previousButtonMillis = 0;

unsigned long previousResetMillis = 0;

const long blinkInterval = 200;

const long fadeInterval = 15;

const long buttonInterval = 300;

const long resetDebounce = 200; // debounce for reset button

// Software PWM settings (works in Wokwi)

const unsigned long pwmPeriod = 10; // ms per PWM cycle (~100 Hz)

// lower value -> smoother PWM but more CPU usage

// Blink state variables

int blinkState = 0;

void setup() {

pinMode(LED1, OUTPUT);

pinMode(LED2, OUTPUT);

pinMode(LED3, OUTPUT);

pinMode(BTN\_MODE, INPUT\_PULLUP);

pinMode(BTN\_RESET, INPUT\_PULLUP);

if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) while (1);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(SSD1306\_WHITE);

showMode("All OFF");

// ensure LEDs are off

digitalWrite(LED1, LOW);

digitalWrite(LED2, LOW);

digitalWrite(LED3, LOW);

}

void showMode(const char \*text) {

display.clearDisplay();

display.setCursor(0, 25);

display.print("Mode: ");

display.println(text);

display.display();

}

void resetToMode0() {

mode = 0;

brightness = 0;

fadeAmount = 5;

blinkState = 0;

digitalWrite(LED1, LOW);

digitalWrite(LED2, LOW);

digitalWrite(LED3, LOW);

showMode("Reset to OFF");

}

void loop() {

unsigned long currentMillis = millis();

// --- RESET BUTTON (HIGH PRIORITY) with debounce ---

if (digitalRead(BTN\_RESET) == LOW && (currentMillis - previousResetMillis >= resetDebounce)) {

previousResetMillis = currentMillis;

resetToMode0();

delay(50); // small pause to avoid immediate retrigger

return; // Exit so reset takes effect immediately

}

// --- MODE BUTTON with debounce ---

if (digitalRead(BTN\_MODE) == LOW && (currentMillis - previousButtonMillis >= buttonInterval)) {

previousButtonMillis = currentMillis;

mode++;

if (mode > 3) mode = 0;

// Reset LEDs when changing modes

digitalWrite(LED1, LOW);

digitalWrite(LED2, LOW);

digitalWrite(LED3, LOW);

brightness = 0;

fadeAmount = 5;

blinkState = 0;

previousBlinkMillis = currentMillis;

previousFadeMillis = currentMillis;

switch (mode) {

case 0:

showMode("All OFF");

break;

case 1:

showMode("Alternate Blink");

break;

case 2:

digitalWrite(LED1, HIGH);

digitalWrite(LED2, HIGH);

digitalWrite(LED3, HIGH);

showMode("All ON");

break;

case 3:

showMode("PWM Fade");

break;

}

delay(50); // tiny extra debounce gap

}

// --- MODE BEHAVIORS (NON-BLOCKING) ---

switch (mode) {

case 1: // Alternate Blink

if (currentMillis - previousBlinkMillis >= blinkInterval) {

previousBlinkMillis = currentMillis;

// Turn all LEDs off first

digitalWrite(LED1, LOW);

digitalWrite(LED2, LOW);

digitalWrite(LED3, LOW);

// Turn on the current LED in sequence

switch (blinkState) {

case 0:

digitalWrite(LED1, HIGH);

break;

case 1:

digitalWrite(LED2, HIGH);

break;

case 2:

digitalWrite(LED3, HIGH);

break;

}

blinkState = (blinkState + 1) % 3;

}

break;

case 3: // PWM Fade (software PWM for Wokwi / no analogWrite)

// update brightness at fadeInterval rate (non-blocking)

if (currentMillis - previousFadeMillis >= fadeInterval) {

previousFadeMillis = currentMillis;

brightness += fadeAmount;

// clamp/flip on edges for stable behaviour

if (brightness <= 0) {

brightness = 0;

fadeAmount = abs(fadeAmount);

} else if (brightness >= 255) {

brightness = 255;

fadeAmount = -abs(fadeAmount);

}

}

// software PWM cycle: compute duty within pwmPeriod

{

unsigned long phase = currentMillis % pwmPeriod; // 0 .. pwmPeriod-1 ms

unsigned int onTime = (unsigned long)brightness \* pwmPeriod / 255u; // ms LED should be ON this cycle

bool on = (phase < onTime);

digitalWrite(LED1, on ? HIGH : LOW);

digitalWrite(LED2, on ? HIGH : LOW);

digitalWrite(LED3, on ? HIGH : LOW);

}

break;

// cases 0 and 2 don't need periodic work

default:

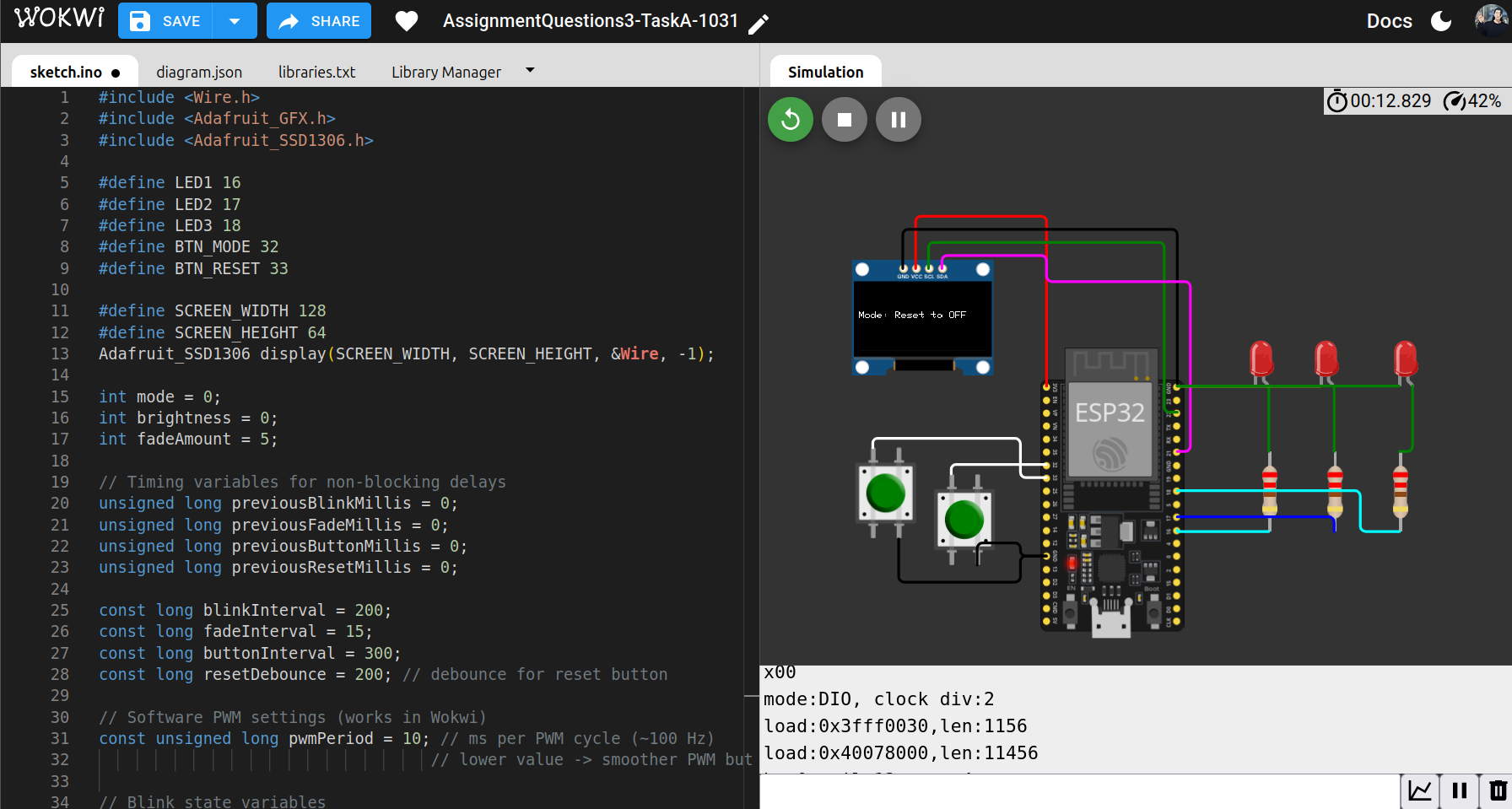
break;

}

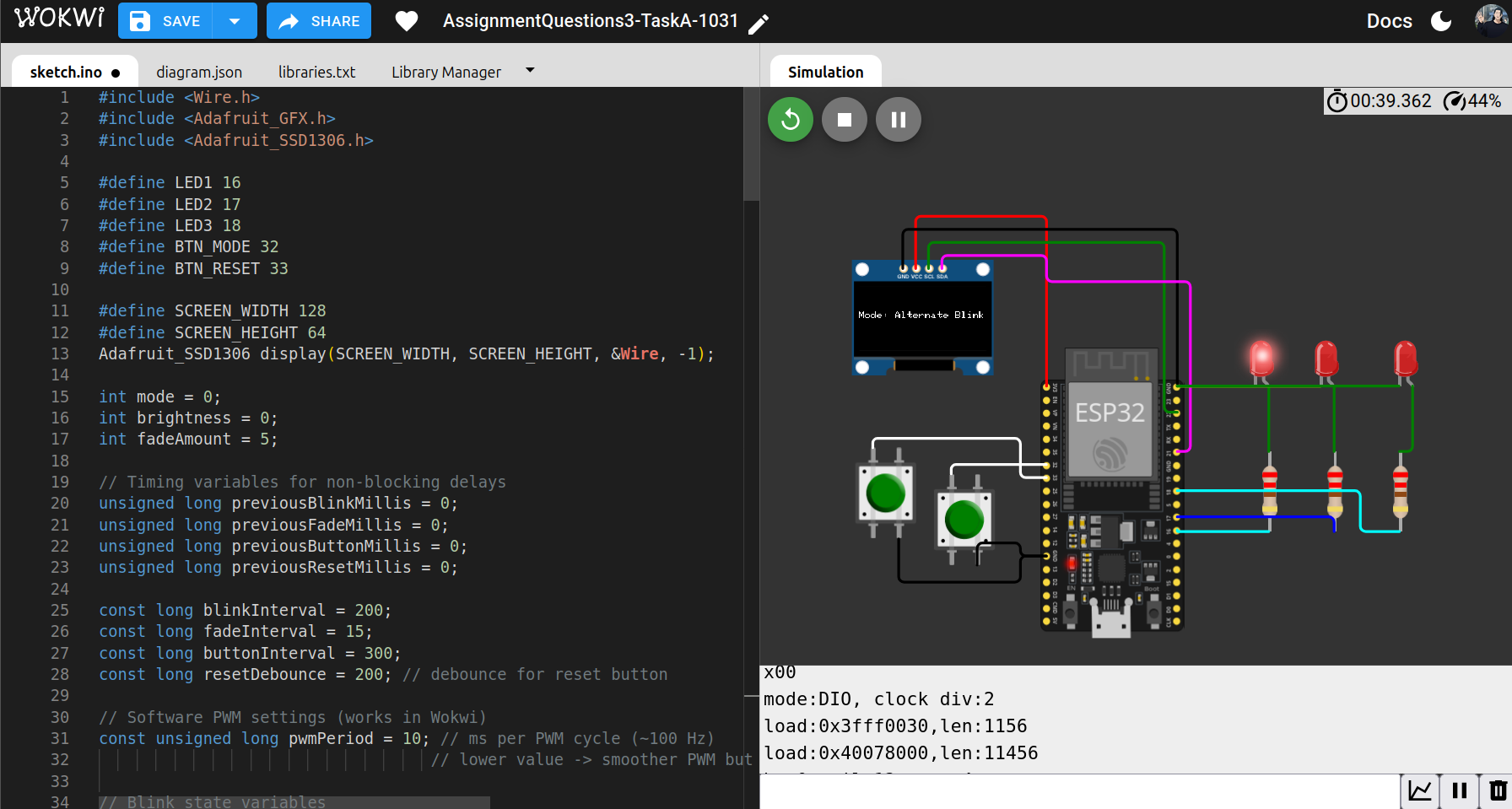
}

**OUTPUT SCREENSHOTS:**

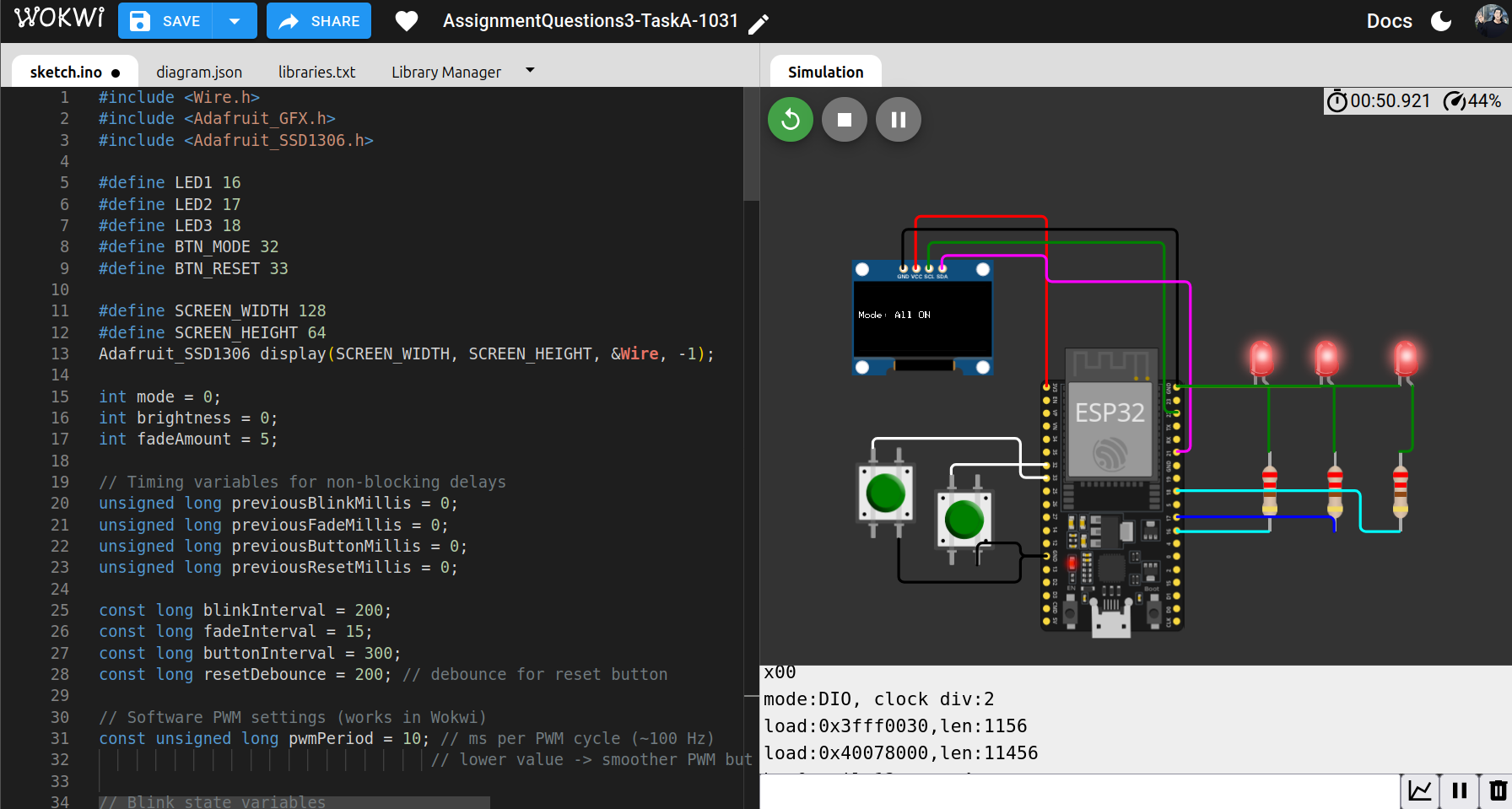
**MODE: All Off**



**MODE: Alternate Blink**

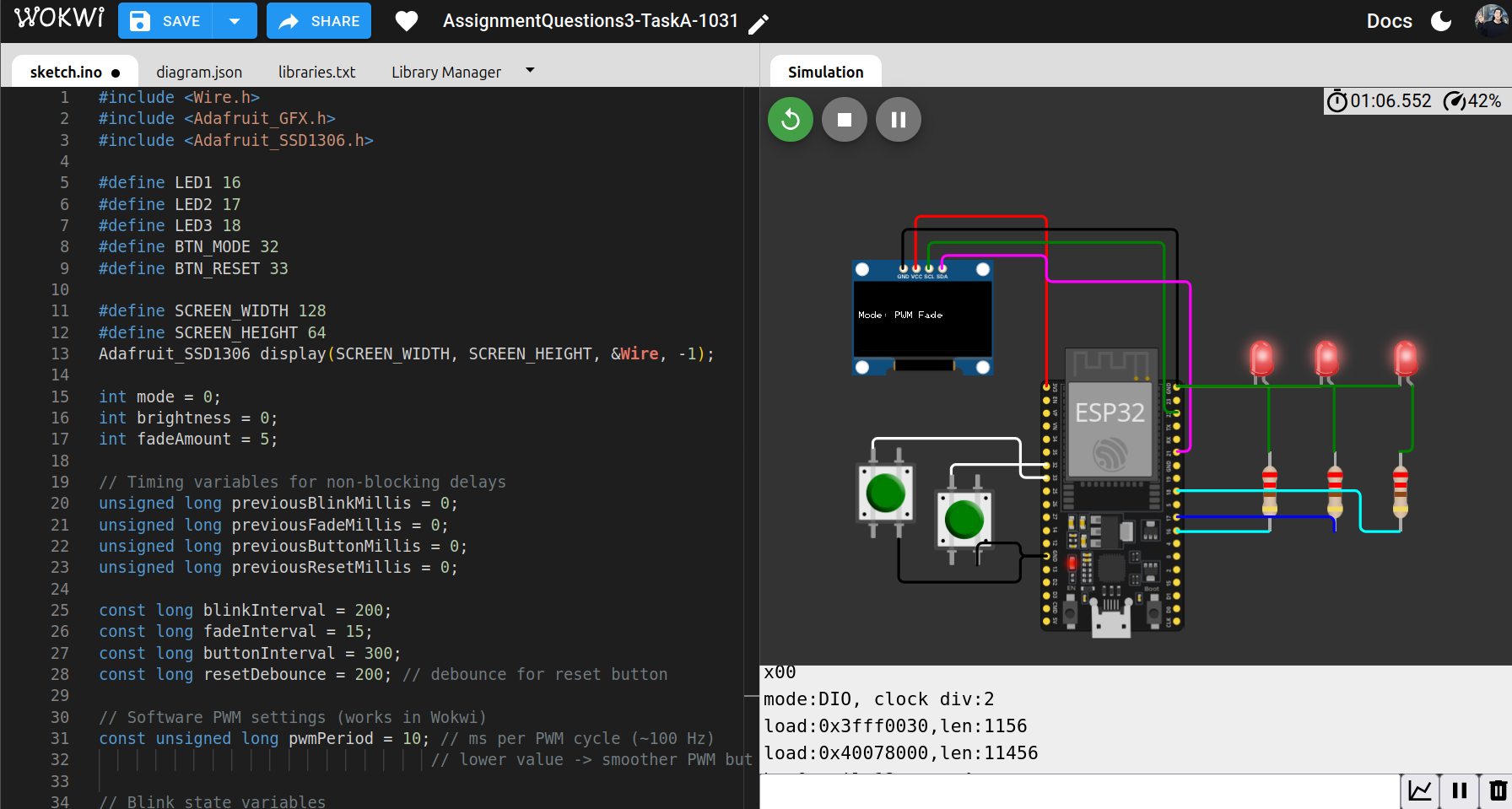


**MODE: All On**

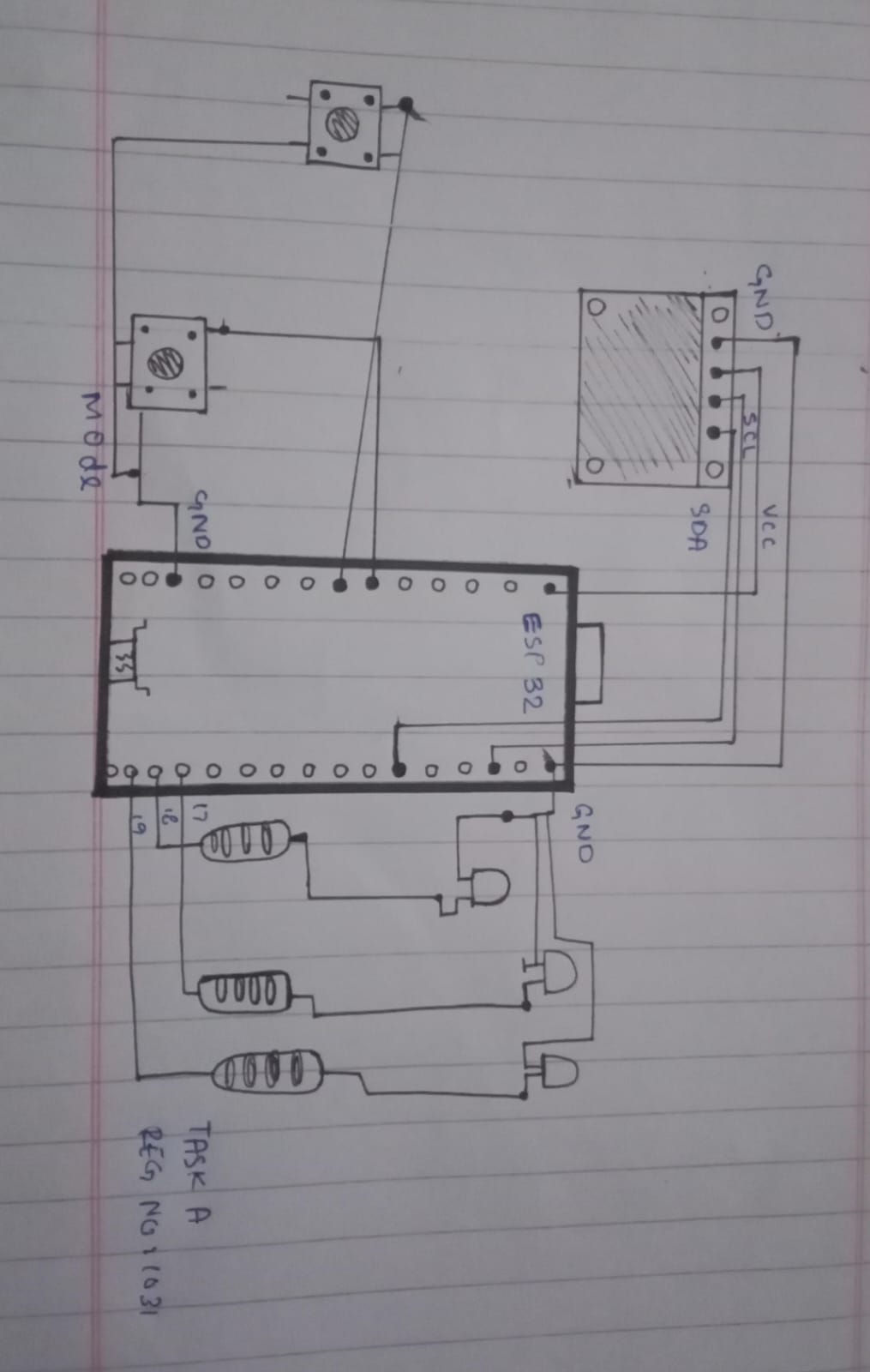


**MODE: Fade**

**WOKVI LINK :** <https://wokwi.com/projects/445804929814828033>



**SKETCH:**



TASK B

**CODE:**

//Hafiza Minahil Shabbir 1031

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

// Pin configuration

const int buttonPin = 32; // single button

const int ledPin = 16; // LED pin

const int buzzerPin = 17; // buzzer pin

// OLED configuration

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

// Variables for button timing

unsigned long buttonPressTime = 0;

bool isButtonPressed = false;

bool ledState = false;

bool longPressActive = false;

const unsigned long longPressDuration = 1500; // 1.5 seconds for long press

const unsigned long debounceDelay = 50; // 50ms debounce time

void showOLED(const char \*message) {

display.clearDisplay();

display.setTextSize(2);

display.setTextColor(SSD1306\_WHITE);

display.setCursor(0, 0);

display.println(message);

display.display();

}

void setup() {

pinMode(buttonPin, INPUT\_PULLUP);

pinMode(ledPin, OUTPUT);

pinMode(buzzerPin, OUTPUT);

Wire.begin(21, 22); // SDA = 21, SCL = 22 for ESP32

Serial.begin(115200);

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

Serial.println("❌ OLED init failed");

while (true);

}

showOLED("Ready");

// Initialize LED to OFF state

digitalWrite(ledPin, LOW);

}

void loop() {

int buttonState = digitalRead(buttonPin);

// Button pressed (LOW because of INPUT\_PULLUP)

if (buttonState == LOW && !isButtonPressed) {

// Wait for debounce period to confirm the press

delay(debounceDelay);

// Check button again after debounce

if (digitalRead(buttonPin) == LOW) {

isButtonPressed = true;

buttonPressTime = millis();

longPressActive = false;

Serial.println("Button pressed - waiting for release");

}

}

// Button is being held

if (buttonState == LOW && isButtonPressed) {

if (!longPressActive && (millis() - buttonPressTime > longPressDuration)) {

// Long press detected – activate buzzer

longPressActive = true;

tone(buzzerPin, 1000);

showOLED("BUZZER");

Serial.println("Long press activated - BUZZER ON");

}

}

// Button released (HIGH because of INPUT\_PULLUP)

if (buttonState == HIGH && isButtonPressed) {

// Wait for debounce period to confirm the release

delay(debounceDelay);

// Check button again after debounce

if (digitalRead(buttonPin) == HIGH) {

noTone(buzzerPin); // stop buzzer immediately when released

if (!longPressActive) {

// Short press – toggle LED

ledState = !ledState;

digitalWrite(ledPin, ledState ? HIGH : LOW);

showOLED(ledState ? "LED ON" : "LED OFF");

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

} else {

// If long press happened

showOLED("Stopped");

Serial.println("Long press stopped");

}

// Reset state

isButtonPressed = false;

longPressActive = false;

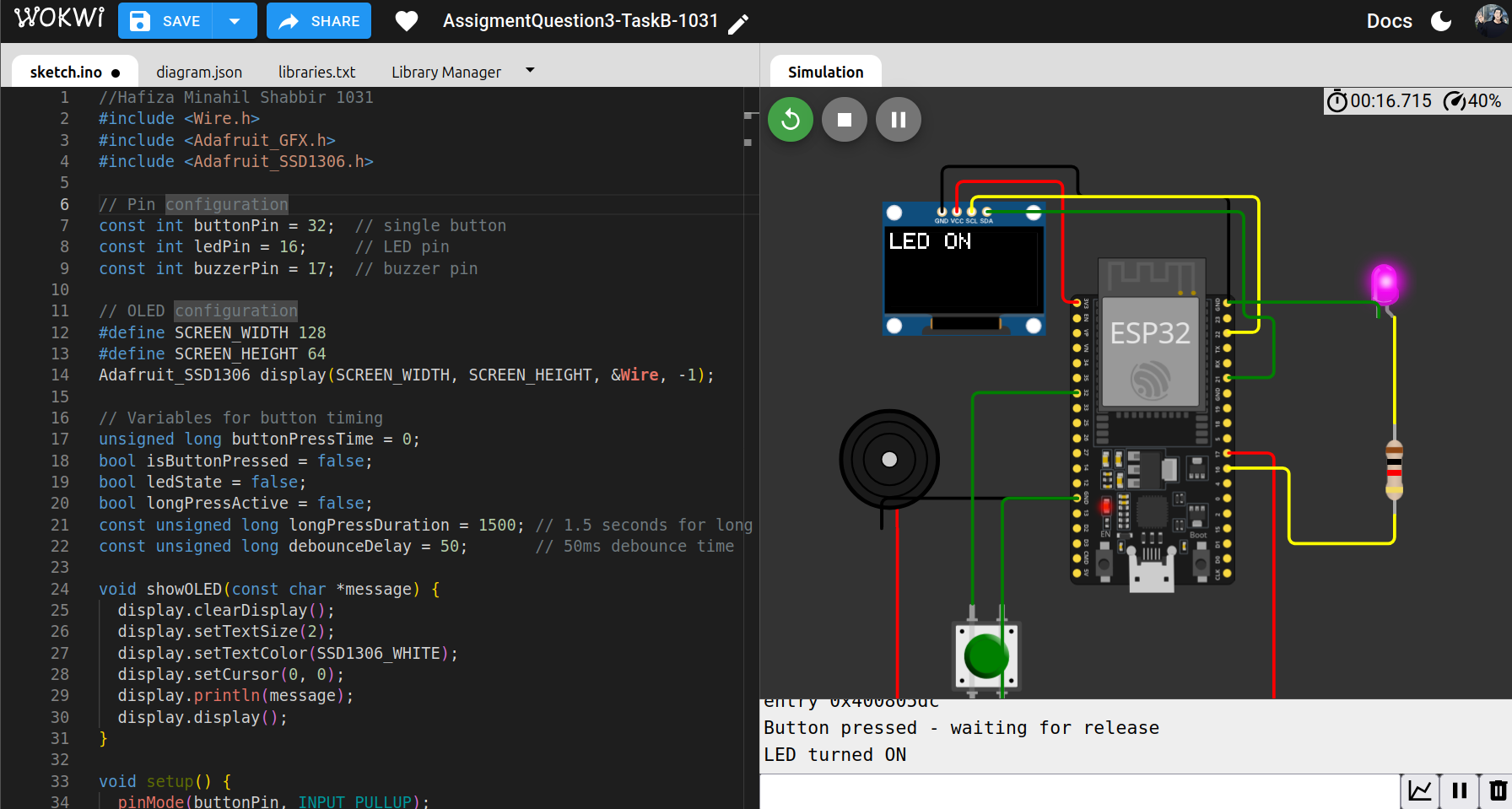
}

}

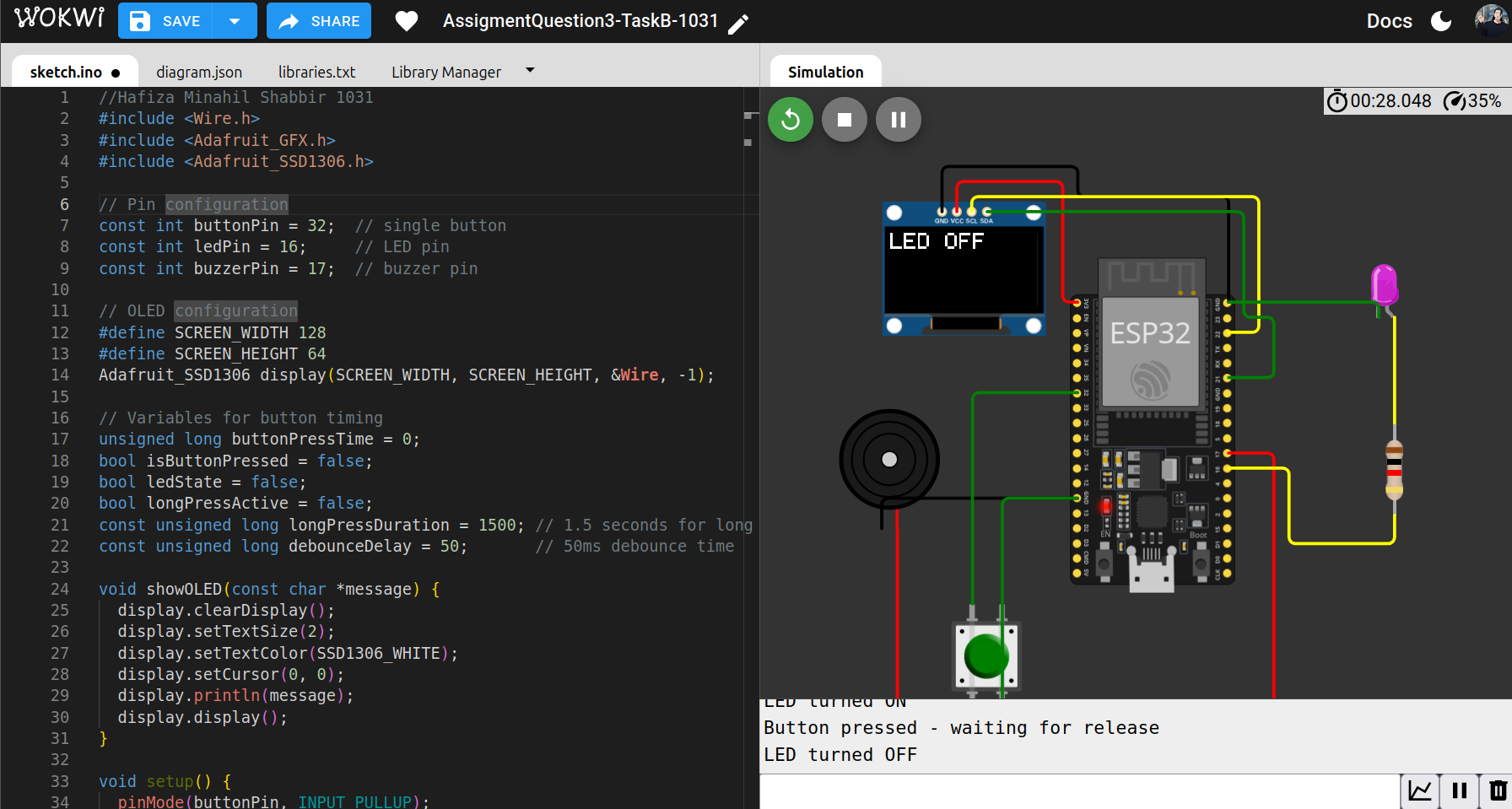
}

**OUTPUT SCREENSHOTS:**

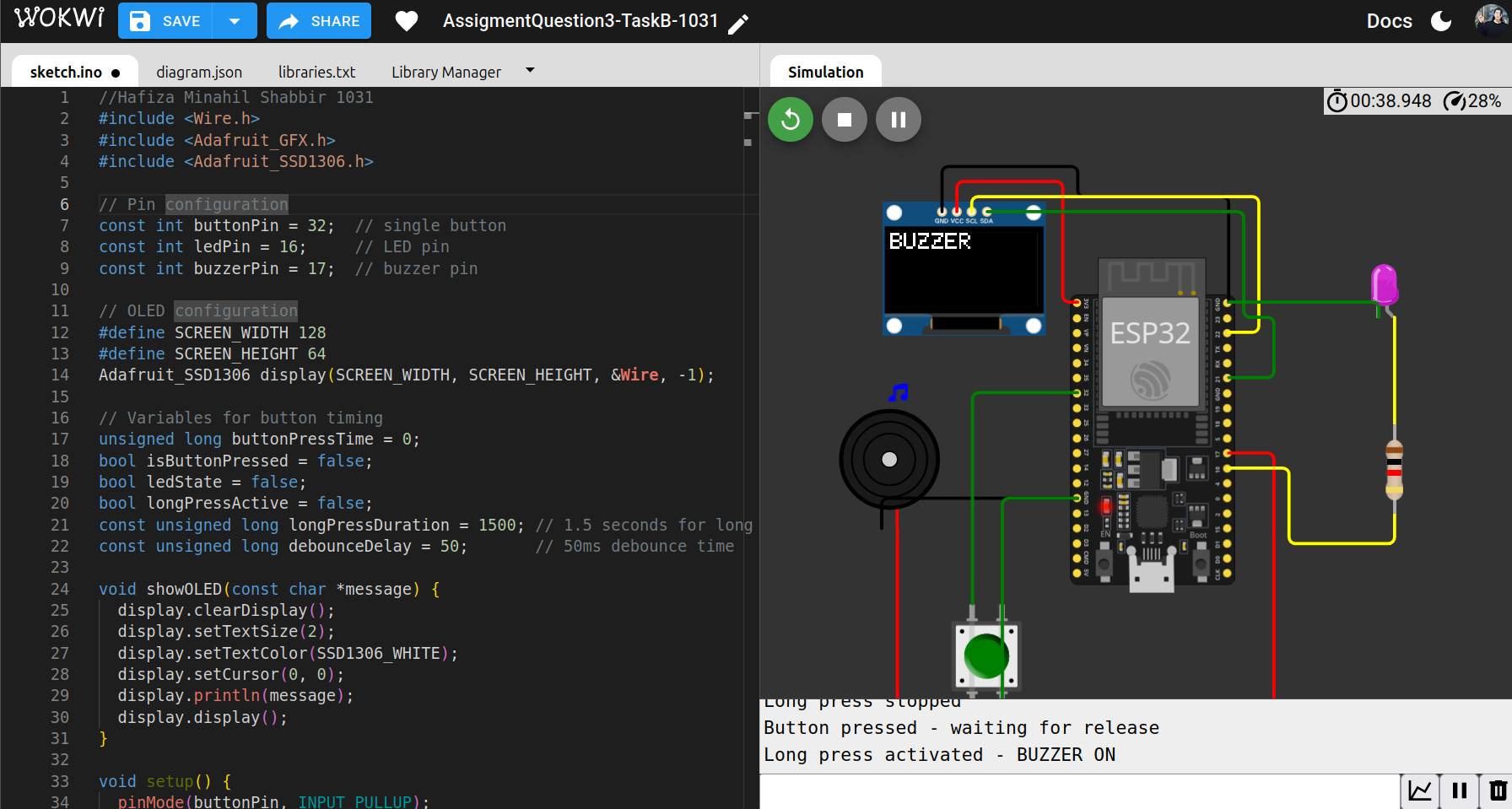
**MODE: LED ON**



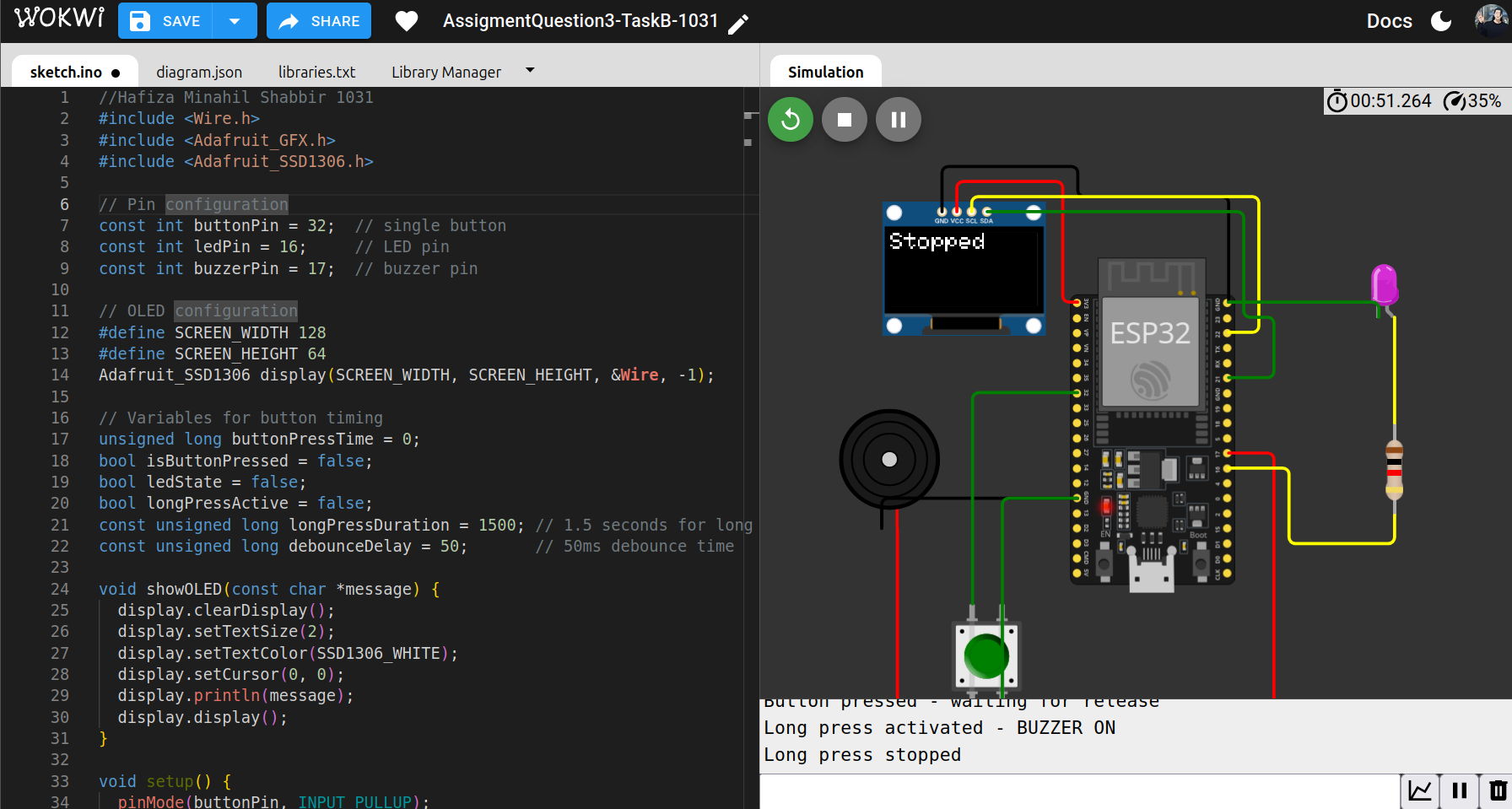
**MODE: LED OFF**



MODE: Buzzer On



**MODE: Buzzer Stopped**



**WOKVI LINK :** <https://wokwi.com/projects/445810132446839809>  
  
**SKETCH:**

