**National Textile University, Faisalabad**

A logo with blue and red squares

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **Name:** | **Fiza Mahboob** |
| **Class:** | **BSCS(A)** |
| **Semester:** | **5th** |
| **Registration No:** | **23-NTU-CS-1027** |
| **Course Name:** | **EMBEDDED IOT SYSTEM** |
| **Assignment:** | **1** |
| **Submitted To:** | **Sir Nasir Mahmood** |

**ESP32 Press-Type Detection (Short & Long Press)**

**1. Project Overview**

This project demonstrates **press-type detection using a single push button** with an **ESP32 microcontroller** and an **OLED display (SSD1306)**.  
The system differentiates between **short press** and **long press** actions:

* A **short press** toggles an LED ON or OFF.
* A **long press (more than 1.5 seconds)** triggers a **buzzer tone**.  
  The OLED display provides **real-time feedback** on the current system state (“LED ON”, “LED OFF”, “BUZZER ON”).

**2. Project Objectives**

* To implement **press-duration-based input detection** using a single button.
* To demonstrate **event-driven programming** using ESP32 and non-blocking timing logic.
* To interface an **OLED display via I2C** for visual feedback.
* To practice **hardware–software integration** using LED and buzzer peripherals.
* To improve understanding of **input handling with debouncing** in embedded systems.

**3. Components Used**

| **Component** | **Quantity** | **Description** |
| --- | --- | --- |
| ESP32 Dev Board | 1 | Main microcontroller unit |
| Push Button | 1 | Used for short & long press detection |
| LED | 1 | Indicates short-press toggle |
| Buzzer | 1 | Plays tone on long press |
| OLED Display (SSD1306, I2C) | 1 | 128×64 OLED for visual feedback |
| Resistor (10kΩ) | 1 | Used for button protection (optional) |
| Breadboard & Jumper Wires | — | For circuit connections |

**4. Circuit Connections**

| **Component** | **ESP32 Pin** | **Description** |
| --- | --- | --- |
| LED | GPIO 18 | Turns ON/OFF on short press |
| Button | GPIO 34 | Input pin (reads LOW when pressed) |
| Buzzer | GPIO 25 | Produces tone on long press |
| OLED SDA | GPIO 21 | I2C Data Line |
| OLED SCL | GPIO 22 | I2C Clock Line |
| VCC & GND | 3.3V & GND | Power and Ground connections |

**Note:**

* The button is configured with INPUT\_PULLUP, meaning it reads **HIGH** when idle and **LOW** when pressed.
* The OLED uses standard I2C pins (SDA=21, SCL=22).

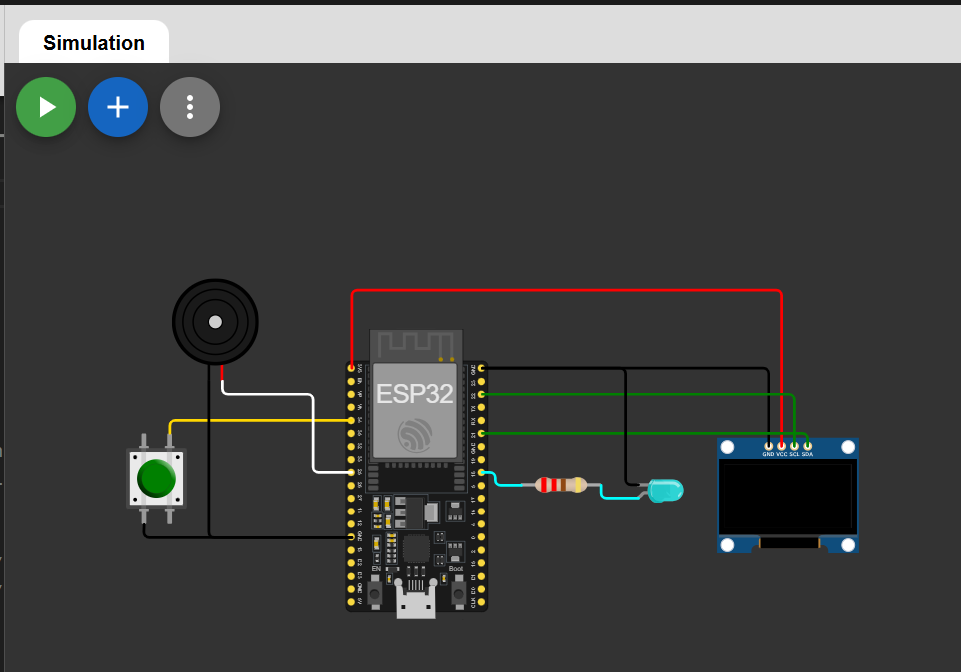
**5. Circuit Diagram (Wokwi Simulation)**

Design the circuit in **Wokwi.com** using:

* OLED (SSD1306) connected via I2C pins 21 (SDA) and 22 (SCL).
* Push button on GPIO 34 connected to GND.
* LED on GPIO 18 and Buzzer on GPIO 25.

**Wiring color suggestion:**

* **Red:** Power (VCC)
* **Black:** Ground (GND)
* **Yellow:** Signal lines (LED/Buzzer/Button)
* **Green:** I2C data lines (SDA/SCL)



**6. Working Principle**

| Press Type | Duration | Action Performed | OLED Message | Device |
| --- | --- | --- | --- | --- |
| Short Press | < 1.5 seconds | Toggles LED ON/OFF | “LED ON” / “LED OFF” | LED |
| Long Press | ≥ 1.5 seconds | Activates buzzer tone for 0.5s | “BUZZER ON” | Buzzer |

**Operation Flow:**

1. When the button is pressed, ESP32 records the press start time.
2. If the press lasts less than 1.5 seconds → LED toggles.
3. If the press exceeds 1.5 seconds → buzzer tone is played.
4. OLED updates after each event to show the current system state.

**7. Features Implemented**

* Detection of **short and long button presses** using millis().
* **Debounce logic** to prevent false triggering.
* **OLED display feedback** for real-time state updates.
* **Tone generation** on long press using tone() function.
* **Clean, modular structure** with descriptive variable names and comments.

**8. Source Code**

#include <Arduino.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

// --- OLED Display Configuration ---

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_ADDR 0x3C

// --- Pin Assignments ---

const int LED\_PIN = 18;      // LED output pin

const int BTN\_PIN = 34;      // Button input pin (connected to GND)

const int BUZZER\_PIN = 25;   // Buzzer output pin

// --- Timing Parameters ---

const unsigned long DEBOUNCE\_DELAY = 50;       // Debounce time for stable button input

const unsigned long LONG\_PRESS\_TIME = 1500;    // Long press threshold (in milliseconds)

// --- OLED Display Object ---

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

// --- State Variables ---

bool buttonPressed = false;         // Tracks current button state

bool ledState = false;              // Stores LED ON/OFF state

unsigned long pressStartTime = 0;   // Time when button was first pressed

bool longPressTriggered = false;    // Prevents repeated long-press action

// --- Function Prototypes ---

void updateOLED(const char \*msg);

void playBuzzerTone();

void setup() {

**Serial**.begin(115200);

  // Configure I/O pins

  pinMode(LED\_PIN, OUTPUT);

  pinMode(BUZZER\_PIN, OUTPUT);

  pinMode(BTN\_PIN, INPUT\_PULLUP);   // Using internal pull-up (LOW when pressed)

  // Initialize OLED with I2C

**Wire**.begin(21, 22);

  if (!display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR)) {

**Serial**.println("OLED initialization failed!");

    while (true);  // Stop execution if OLED fails

  }

  digitalWrite(LED\_PIN, LOW);   // Start with LED turned OFF

  updateOLED("LED OFF");

**Serial**.println("System initialized - Task B");

}

void loop() {

  unsigned long now = millis();

  int reading = digitalRead(BTN\_PIN);

  // Detect when button is pressed (FALLING edge)

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

    buttonPressed = true;

    pressStartTime = now;

    longPressTriggered = false;

**Serial**.println("Button pressed");

  }

  // Handle long press detection

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

    if ((now - pressStartTime) >= LONG\_PRESS\_TIME) {

      longPressTriggered = true;

      playBuzzerTone();

      updateOLED("BUZZER ON");

**Serial**.println("Long press detected");

    }

  }

  // Detect when button is released (RISING edge)

  if (reading == HIGH && buttonPressed) {

    unsigned long pressDuration = now - pressStartTime;

    // Execute short press action if not a long press

    if (pressDuration < LONG\_PRESS\_TIME && !longPressTriggered) {

      ledState = !ledState;

      digitalWrite(LED\_PIN, ledState);

      if (ledState) {

        updateOLED("LED ON");

**Serial**.println("Short press → LED ON");

      } else {

        updateOLED("LED OFF");

**Serial**.println("Short press → LED OFF");

      }

    }

    // Restore OLED after long press ends

    else if (longPressTriggered) {

      if (ledState) updateOLED("LED ON");

      else updateOLED("LED OFF");

    }

    buttonPressed = false;

    delay(DEBOUNCE\_DELAY);

  }

}

// --- OLED Text Update Function ---

// Clears the screen and displays the provided message

void updateOLED(const char \*msg) {

  display.clearDisplay();

  display.setTextSize(2);

  display.setTextColor(SSD1306\_WHITE);

  display.setCursor(0, 15);

  display.println(msg);

  display.display();

}

// --- Buzzer Activation Function ---

// Produces a 1000Hz tone for 0.5 seconds

void playBuzzerTone() {

  tone(BUZZER\_PIN, 1000, 500);

}

**9. Code Explanation**

**a. Setup Section**

* Initializes **Serial Monitor**, **OLED Display**, and all I/O pins.
* Displays the initial message **“LED OFF”** on OLED.

**b. Loop Section**

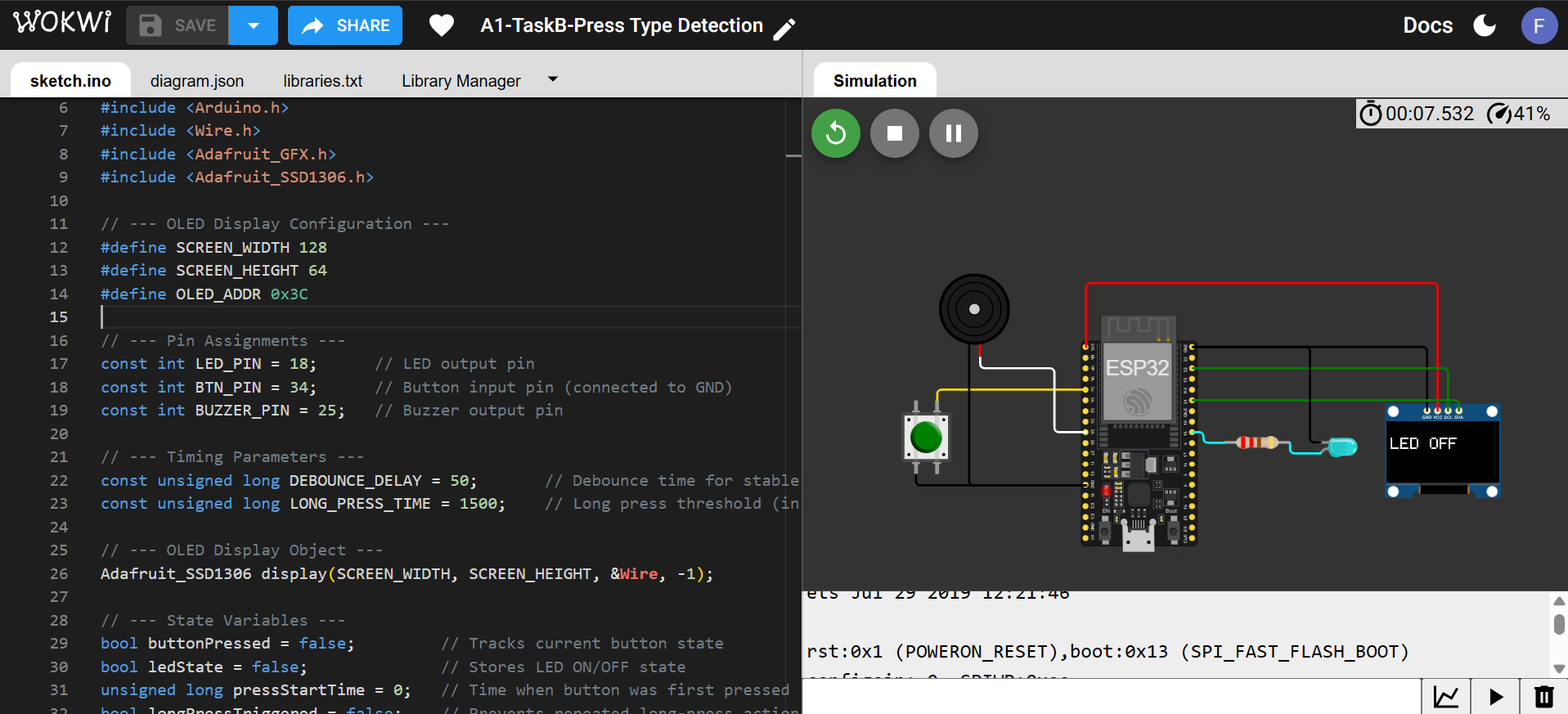
* Continuously reads the button input.
* Uses **millis()** to measure press duration.
* Executes either short press (LED toggle) or long press (buzzer tone).
* Updates the OLED with the corresponding message.

**c. Helper Functions**

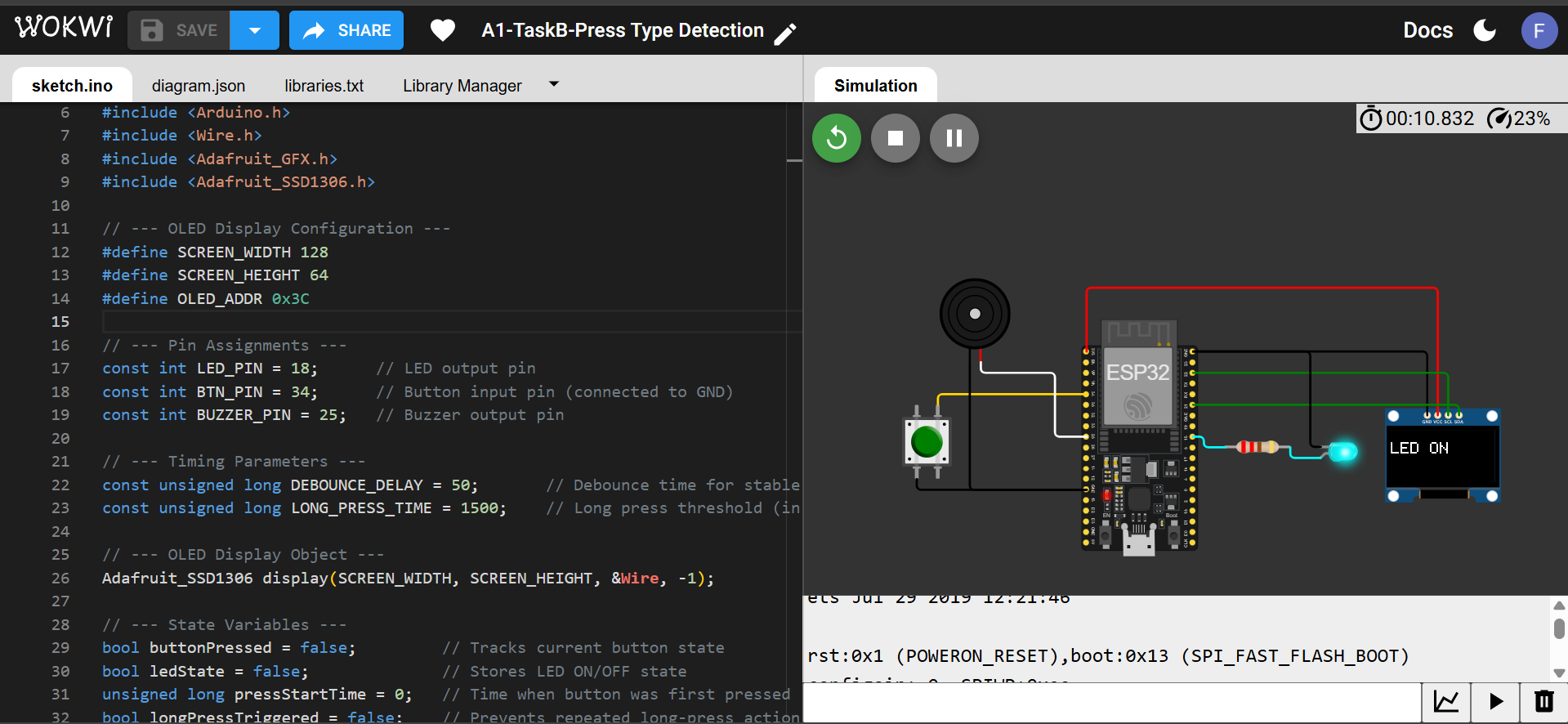
1. **updateOLED(msg)** – Clears the screen and displays a new message.
2. **playBuzzerTone()** – Generates a 1000 Hz tone for 500 milliseconds.

**10. Output Demonstration**

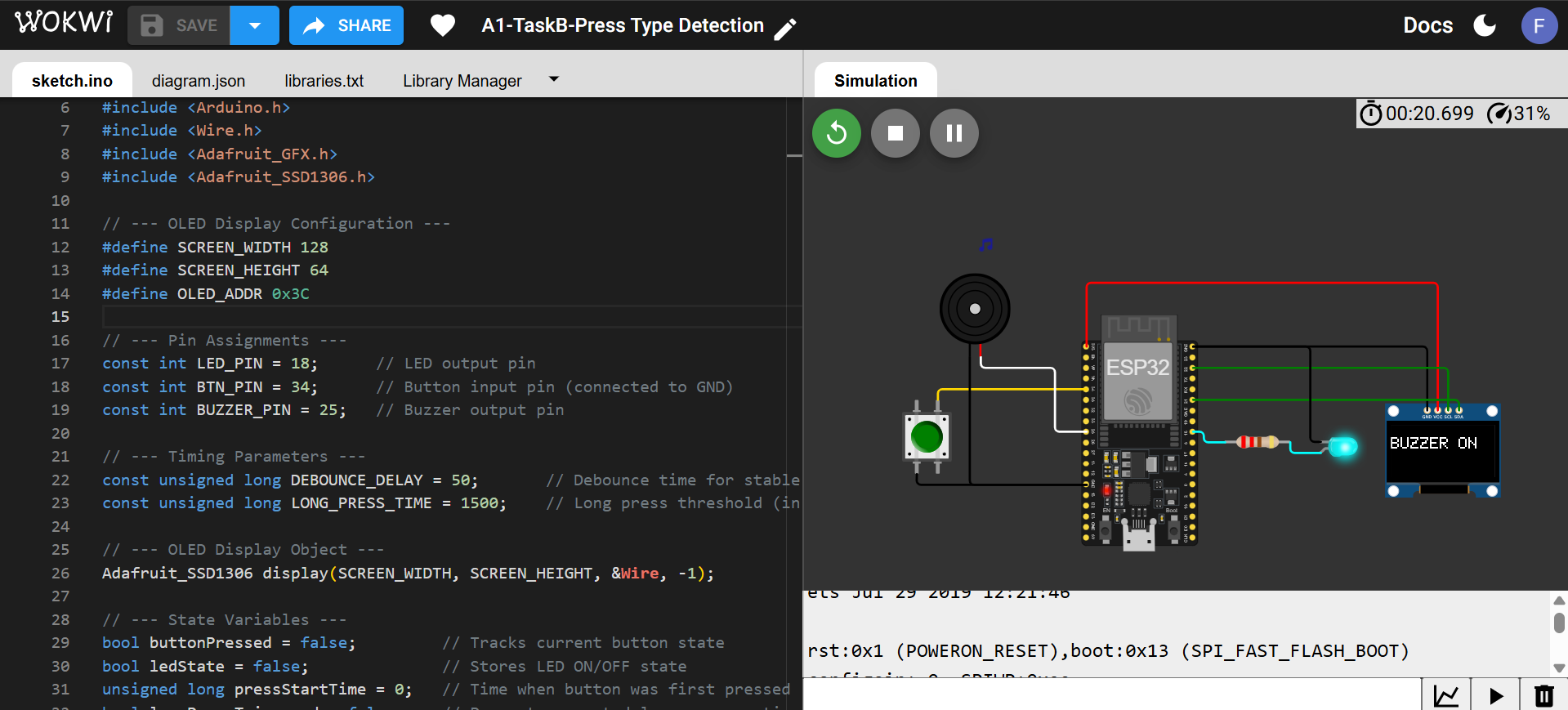
**Short Pressed-LED OFF(Toggle)**

****

**Again short Pressed-LED ON:**

****

**Long Pressed(>1.5) -Buzzer Tone:**

****

**12. Conclusion**

This project successfully implements **press-type detection logic** using ESP32 and demonstrates the integration of multiple peripherals — LED, Buzzer, and OLED Display.  
It highlights **event-driven programming**, **debounce control**, and **timing-based input recognition** without using blocking delays.  
Through this task, practical understanding of **real-time embedded systems** and **I2C communication** has been achieved.