

7TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES

Ayala Boulevard, Ermita, Manila

CIT-ELECTRONICS DEPARTMENT

**CPET11L-M – Microprocessor and Microcontroller Systems, Lab**

**1st Semester, SY 2-24-2025**

<b>Name:</b> Joseph C. Arenas	<b>Instructor:</b> Prof. Tristan Mopas
<b>Course &amp; Section:</b> BET-CPET- 3A	<b>Date Submitted:</b> September 27, 2025

**Activity 4**

**Topic 1: Voice Activated LED with ESP 32, Google Assistant, and Alexa**

**I. OBJECTIVES**

- To apply practical knowledge in using the ESP32
- To explain the functions and components of the related topics
- To implement the ESP32, 2 Channel Relay Module, Push button, resistors, and LED components in a working circuit.
- To develop and enhance problem-solving skills related to the topics

**II. EQUIPMENT AND MATERIALS**

**HARDWARE**

- ESP32
- Breadboard
- Jumper Wires (Male-to-Male & Female-to-Male)
- Laptop
- Push button
- 220 $\Omega$  Resistor x2
- 2-Channel Relay Module
- Red and Green LEDs

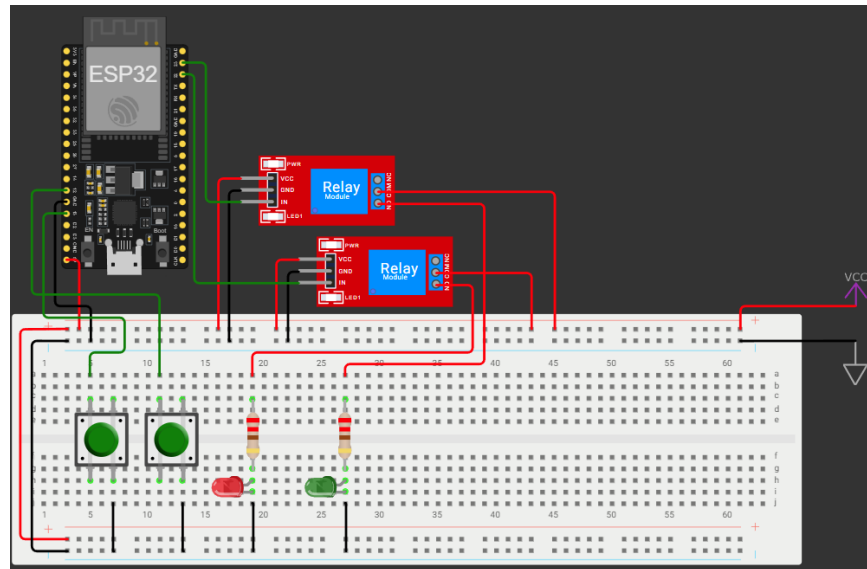
## SOFTWARE

- Arduino IDE with libraries for specified components.
- MS Word
- Wokwi

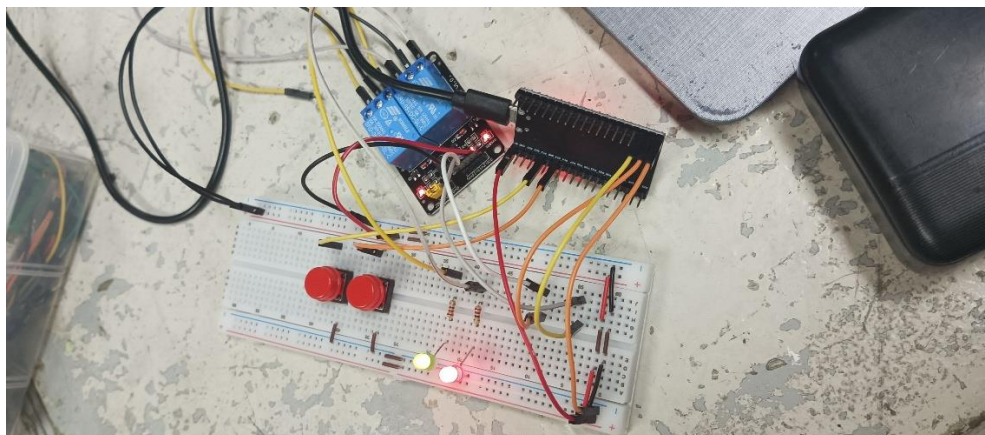
## III. DIAGRAM

### ===== TOPIC 1: Voice Activated LED with ESP 32, Google Assistant, and Alexa =====

#### A. Wokwi Simulation



#### B. Breadboard



## C. Source Code

```
#include <Arduino.h>
#include <WiFi.h>
#include "SinricPro.h"
#include "SinricProSwitch.h"

// ===== WiFi & Sinric Pro Credentials =====

#define WIFI_SSID   "matrix"
#define WIFI_PASS   "135792468"
#define APP_KEY     "6e41d7c9-a153-49f5-9b1e-5e26b27c31eb"
#define APP_SECRET  "4a01b0b8-62aa-4706-8458-e7614c87f710-8d9d90df-ca7b-410b-96e5-dacefc1a21cd"

// Sinric Device IDs (create 2 Switch devices in SinricPro)
#define DEVICE_ID_1 "68d438dd51811ad2b7504752"
#define DEVICE_ID_2 "68d4394451811ad2b75047d7"

// ===== Pin Definitions =====

#define RELAY1_PIN 23
#define RELAY2_PIN 22
#define BUTTON1_PIN 13
#define BUTTON2_PIN 12
#define LED1_PIN  21
#define LED2_PIN  19

#define DEBOUNCE_TIME 250
#define BAUD_RATE 9600
```

```
// Track button states for debounce
bool lastBtn1 = HIGH, lastBtn2 = HIGH;
unsigned long lastTime1 = 0, lastTime2 = 0;

// ===== Helper Functions =====

void setRelay(int relayPin, int ledPin, bool state) {
    digitalWrite(relayPin, state ? LOW : HIGH); // Relay active LOW
    digitalWrite(ledPin, state ? HIGH : LOW);
    // LED follows ON/OFF
}

// ===== Sinric Callbacks =====

bool onPowerState1(const String &deviceId, bool &state) {
    Serial.printf("[Sinric] Device1 -> %s\n", state ? "ON" : "OFF");
    setRelay(RELAY1_PIN, LED1_PIN, state);
    return true;
}

bool onPowerState2(const String &deviceId, bool &state) {
    Serial.printf("[Sinric] Device2 -> %s\n", state ? "ON" : "OFF");
    setRelay(RELAY2_PIN, LED2_PIN, state);
    return true;
}
```

```
// ===== Setup =====

void setup() {
  Serial.begin(BAUD_RATE);
  pinMode(RELAY1_PIN, OUTPUT);
  pinMode(RELAY2_PIN, OUTPUT);
  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);
  pinMode(BUTTON1_PIN,
    INPUT_PULLUP);
  pinMode(BUTTON2_PIN,
    INPUT_PULLUP);

  // default OFF
  setRelay(RELAY1_PIN, LED1_PIN, false);
  setRelay(RELAY2_PIN, LED2_PIN, false);
  // WiFi
  Serial.println("Connecting to WiFi...");
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.print("\nWiFi connected: ");
  Serial.println(WiFi.localIP());

  // Sinric Pro
  SinricProSwitch &switch1 =
  SinricPro[DEVICE_ID_1];

  SinricProSwitch &switch2 =
  SinricPro[DEVICE_ID_2];

  switch1.onPowerState(onPowerState1);
  switch2.onPowerState(onPowerState2);

  SinricPro.begin(APP_KEY,
  APP_SECRET);

  SinricPro.restoreDeviceStates(true);
}
```

```
// ===== Loop =====

void loop() {
  SinricPro.handle();

  unsigned long now = millis();

  // --- Button 1 ---
  bool btn1 = digitalRead(BUTTON1_PIN);

  if (btn1 != lastBtn1 && now - lastTime1 >
  DEBOUNCE_TIME) {
    lastTime1 = now;

    if (btn1 == LOW) { // Pressed

      bool currentState =
      (digitalRead(RELAY1_PIN) == LOW);

      setRelay(RELAY1_PIN, LED1_PIN,
      !currentState);

      SinricProSwitch &sw1 =
      SinricPro[DEVICE_ID_1];

      sw1.sendPowerStateEvent(!currentState);
    }

    lastBtn1 = btn1;
  }

  // --- Button 2 ---
  bool btn2 = digitalRead(BUTTON2_PIN);

  if (btn2 != lastBtn2 && now - lastTime2 >
  DEBOUNCE_TIME) {
    lastTime2 = now;

    if (btn2 == LOW) {

      bool currentState =
      (digitalRead(RELAY2_PIN) == LOW);

      setRelay(RELAY2_PIN, LED2_PIN,
      !currentState);

      SinricProSwitch &sw2 =
      SinricPro[DEVICE_ID_2];

      sw2.sendPowerStateEvent(!currentState);
    }

    lastBtn2 = btn2;
  }
}
```

## IV. PROCEDURE

### A. Preparation

- Gather the required components as stated in Chapter 2.
- Prepare the ESP32 on your preferred device.
- Review the problem for each topic and formulate or search for a circuit diagram given the problem of each topic.
- Simulate your circuit diagram to a circuit simulator such as Wokwi.

### B. Actual

- Download the necessary libraries needed to properly utilize the ESP32 as well as connect it to a compatible IOT app that would render it useful to work with Google Assistant and Alexa. Open the Arduino IDE and go to Preferences > and copy the links by <https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html> and paste it on Additional Boards Manager URLs. On the proceeding window, press okay. Go to the Boards Manager afterwards then install the esp32 library by Espressif Systems afterwards. Download and install the remaining libraries Sinric Pro by Bories Jaeger, WebSockets by Markus Sattler, and ArduinoJson by Benoit Blanchon afterwards. Open Sinric Pro and register account. Once an account has been registered, create two devices (for red and green led). Once all of these are set up, download Google Home and Amazon Alexa on a phone, register an account and link your Sinric Pro account to both.
- For the actual circuit itself, connect the ESP32 micro USB to your preferred device that has Arduino IDE. Connect VCC and GND pins of the ESP 32 to the “+” and “-“ of your breadboard. Afterwards connect pins 23 and 22 to IN1 and IN2 respectively of the 2-Channel Relay Module. Connect pins 13 and 12 of the ESP32 to the push buttons with their adjacent pins going to ground. Connect the VCC and GND pins of the relay module to 5v and ground. Furthermore, connect the Commons of the relay module to 5v and connect their NOs to their respective LEDs with 220Ω Resistor.

### **C. Checking**

- Ensure that the push buttons turn on and off their respective designated LEDs.  
Make sure that Google Home and Amazon Alexa can turn the LEDs on and off either through their UI controls or through voice commands.

### **D. Uploading**

- Upload the final and corrected code to the ESP32.
- Ensure that topic 1 works properly.

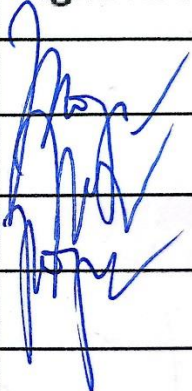
## **V. CONCLUSION**

The experiment showcased the integration of ESP32 with relay modules, push buttons, and Sinric Pro to enable both manual and voice-controlled LED operations through Google Assistant and Alexa. It provided hands-on experience in setting up IoT devices, coding in Arduino IDE, and troubleshooting both hardware and software connections. Overall, the activity strengthened practical skills in microcontroller systems, circuit implementation, and smart home automation concepts.

Name: Arenas, Joseph C.

Section: BET-CPET 3A

Activity No: 4 - LCD Timer

Topic	Date	Time	Signature
12:00 w/ Buzzer on LCD Display <del>Keep quarter timer</del>	09/10/2025	11:47	
<del>311</del> Date & Time with LCD & RTC <del>LCD Timer with RTC</del>	09/10/2025	12:13	
Time Alarm with LCD & RTC	09/24/2025	10:20AM	