**Week 3 Lab 1 Report**

**Name:** Javon Teo Tze Kai

**Student ID:** A0233706J

**Lab Group:** B2

**Task 1:**

I made the program such that:

1. the LED is lit up for 4 seconds and then off for 4 seconds,
2. the LED is lit up for 2 seconds and then off for 2 seconds,
3. the LED is permanently on.

I decided to put the first 2 steps in the setup function as it would only be run once, and the third step would be in the loop function to make sure that it is permanently on.

void setup() {

  // initialize digital pin 5 as an output.

  pinMode(LED, OUTPUT);

  digitalWrite(LED, HIGH);  // turn the LED on (HIGH is the voltage level)

  delay(4000);                      // wait for 4 second

  digitalWrite(LED, LOW);   // turn the LED off by making the voltage LOW

  delay(4000);

  digitalWrite(LED, HIGH);  // turn the LED on (HIGH is the voltage level)

  delay(2000);                      // wait for 2 second

  digitalWrite(LED, LOW);   // turn the LED off by making the voltage LOW

  delay(2000);

}

// the loop function runs over and over again forever

void loop() {

  digitalWrite(LED, HIGH);  // turn the LED on (HIGH is the voltage level)

}

**Task 2:**

My program makes the LED pulse up and down in brightness in 4 stages: duty parameter set to 0, 50, 100, and 150. I firstly set a Boolean variable increasing such that if the current brightness had not reached 150, the increasing variable would be set to true. Once it reached 150, the increasing variable would be set to false.

The current brightness variable increments by 50 and decrements by 50 if the increasing variable is set to true or false respectively, causing the LED to increase and decrease in brightness.

if (increasing) {

    current\_brightness += 50;

  } else {

    current\_brightness -= 50;

  }

  ledcWrite(LEDCHANNEL, current\_brightness);

  delay(1000);

  if (current\_brightness == high\_brightness) {

    increasing = false;

  } else if (current\_brightness == low\_brightness) {

    increasing  = true;

  }

**Task 3:**

I declared a global variable pressedTime to record the latest time the button was pressed. I also set the debounce time as 50 milliseconds. If there was a change in state of the button detected within 50ms of the pressedTime, this would indicate that it probably would have been caused by a bounce, and hence the program ignores it.

 if (digitalRead(SWITCH) == LOW) {

   if ((millis() - pressedTime) > DEBOUNCE\_TIME) {

     state = !state;

     pressedTime = millis();

   }

  }

**Task 4:**

Similar approach to task 3. However, I used a static volatile variable to record the latest time the button was pressed. A change in state of the button within the debounce time of 50ms would be ignored. This conditional logic and static variable are declared in the isr function.

void IRAM\_ATTR isr() {

  static volatile unsigned long lastInterruptTime = 0;

  volatile unsigned long interruptTime = millis();

  if ((interruptTime - lastInterruptTime) > DEBOUNCE\_TIME) {

    pressCount++;

    pressed = true;

  }

  lastInterruptTime = interruptTime;

}

**Task 5:**

When the DHT11 was undisturbed, the temperature and humidity recorded averaged 24.50°C and 52.00% respectively. While when it was blown on, the temperature and humidity rose above 25.00°C and 60.00% respectively. As such, these were the threshold I used to make the LED turn on: when the humidity or temperature rose beyond their threshold, the code would call the ledcWrite function with the duty parameter set to 150. Otherwise, the same function would be called with the duty parameter set to 0. Hence, the LED would be turned on or off depending on the humidity and temperature.

  if ((temperature > 25.00) || (humidity > 60.00)) {

    current\_brightness = 150;

  } else {

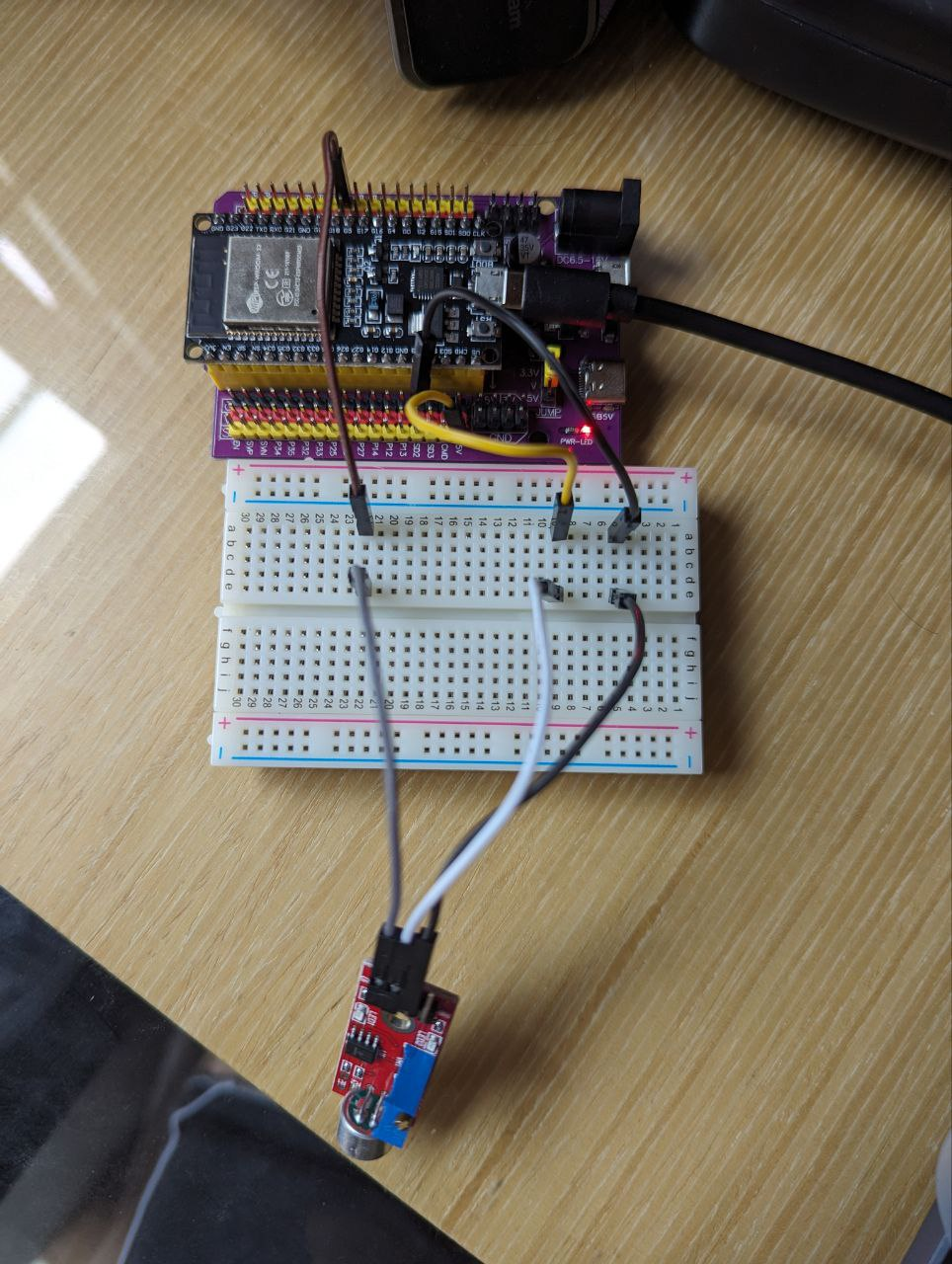
    current\_brightness = 0;

  }

  ledcWrite(LEDCHANNEL, current\_brightness);

As this task was completed in an air-conditioned environment, the threshold should be adjusted if it were to be conducted in a different environment.

**Task 6:**



I decided to try out the KY-038 Sound Sensor Module. I connected the Digital Output on the sensor to GPIO 5 on the ESP32. I then created a simple program that could be used to detect a clap sound received by the sensor. Here is the code excerpt:

#define PIN 5

unsigned long last\_event = 0;

void setup() {

  pinMode(PIN, INPUT);

  Serial.begin(115200);

}

void loop() {

  int output = digitalRead(PIN);

  if (output == LOW) {

    if (millis() - last\_event > 25) {

      Serial.println("Clap sound was detected!");

    }

    last\_event = millis();

  }

}