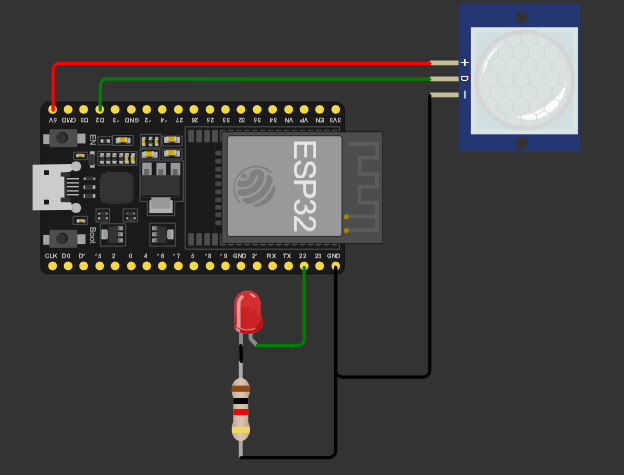
**PRACTICE 4: MODERN IOT**

**1. PIR motion sensor**

****

**code:**

const int led =5;

const int pir =19;

void setup(){

pinMode(led,OUTPUT);

pinMode(pir,INPUT);

Serial.begin(9600);

}

void loop(){

const int IP=digitalRead(pir);

Serial.println(IP);

delay(100);

if(IP==1){

digitalWrite(led,HIGH);

Serial.println("Phát hiện chuyển động….");

delay(5000);

digitalWrite(led, LOW); // Tắt LED

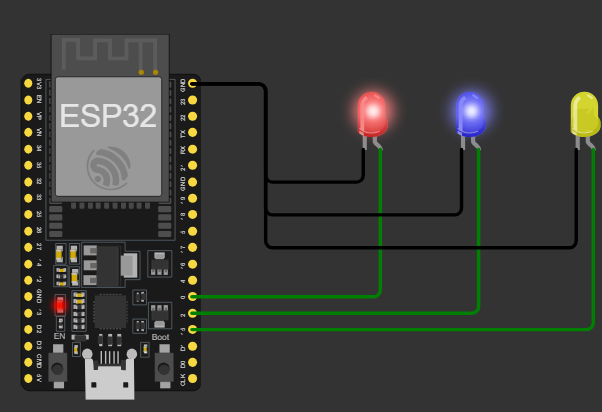
Serial.println("Chờ 1.2 giây...");

delay(1200);

}

}

2. Three LED



code:

#define LED1 0 // Define GPIO for LED1

#define LED2 2 // Define GPIO for LED2

#define LED3 15 // Define GPIO for LED3

void setup() {

pinMode(LED1, OUTPUT);

pinMode(LED2, OUTPUT);

pinMode(LED3, OUTPUT);

Serial.begin(115200);

Serial.println("ESP32 đang hoạt động!");

}

void loop() {

// Turn LEDs on sequentially

digitalWrite(LED1, HIGH);

delay(500);

digitalWrite(LED2, HIGH);

delay(500);

digitalWrite(LED3, HIGH);

delay(500);

// Turn LEDs off sequentially

digitalWrite(LED1, LOW);

delay(500);

digitalWrite(LED2, LOW);

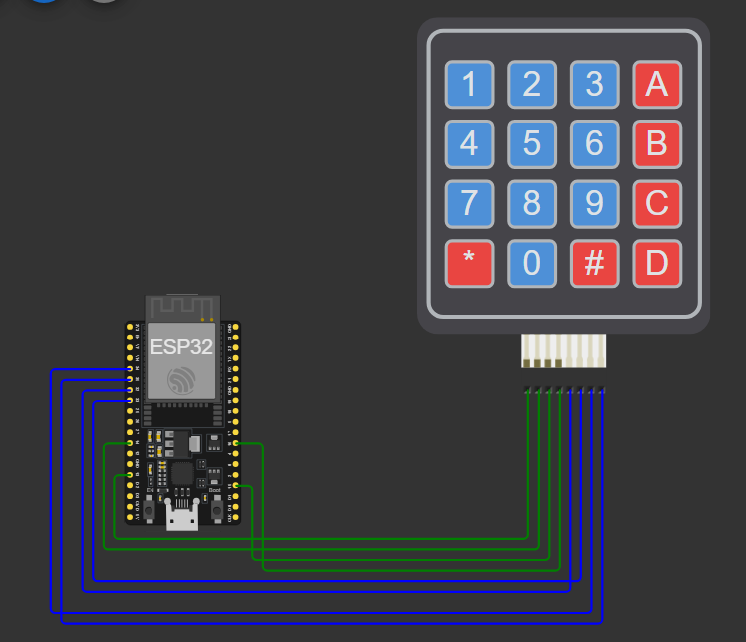
delay(500);

digitalWrite(LED3, LOW);

delay(500);

}

3. Keypad



Code:

import machine

import time

# Define rows and columns

ROWS = [13, 14, 15, 16] # GPIO pins for rows - OUTPUTS

COLS = [32, 33, 34, 35] # GPIO pins for columns - INPUTS

# Keypad layout

KEYPAD = [

['1', '2', '3', 'A'],

['4', '5', '6', 'B'],

['7', '8', '9', 'C'],

['\*', '0', '#', 'D']]

# Configure rows as outputs

row\_pins = [machine.Pin(pin, machine.Pin.OUT) for pin in ROWS]

# Configure columns as inputs with pull-up resistors

col\_pins = [machine.Pin(pin, machine.Pin.IN, machine.Pin.PULL\_UP) for pin in COLS]

def scan\_keypad():

for row\_idx, row\_pin in enumerate(row\_pins):

# Activate the current row

row\_pin.value(1)

for col\_idx, col\_pin in enumerate(col\_pins):

if col\_pin.value() == 0: # Check if the column is pulled LOW

row\_pin.value(0) # Deactivate the row before returning

return KEYPAD[row\_idx][col\_idx] # Return the key

row\_pin.value(0) # Deactivate the current row

return None # No key pressed

# Main loop

while True:

key = scan\_keypad()

if key:

print(f"Key Pressed: {key}")

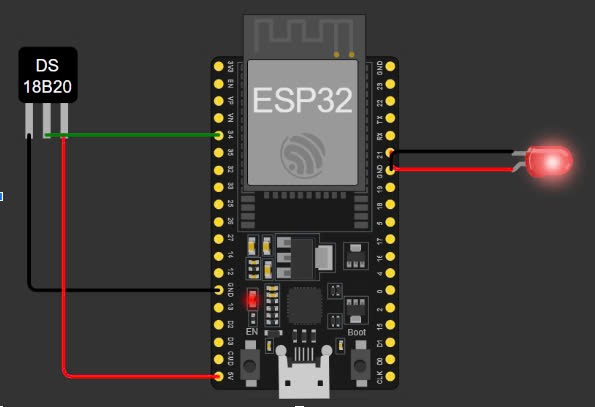
time.sleep(0.1) # Debounce delay

4.

Refactor the temperature sensing application to include a flashing/blinking LED.

As the temperature increases, the LED should flash faster. On the other hand,

when the tempera-ture decreases the LED blinks slower.



code:

#define TEMP\_SENSOR\_PIN 34 // LM35 connected to GPIO 34 (ADC1)

#define LED\_PIN 21 // LED connected to GPIO 2

void setup() {

Serial.begin(115200);

pinMode(LED\_PIN, OUTPUT);

}

void loop() {

int sensorValue = analogRead(TEMP\_SENSOR\_PIN);

float voltage = sensorValue \* (3.3 / 4095.0); // Convert ADC value to voltage (ESP32 is 12-bit ADC)

float temperature = voltage \* 100.0; // Convert voltage to Celsius for LM35

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println(" °C");

// Calculate LED blink delay based on temperature

int blinkDelay = map(temperature, 20, 40, 1000, 100); // Higher temp = faster blink

// Blink LED

digitalWrite(LED\_PIN, HIGH);

delay(blinkDelay);

digitalWrite(LED\_PIN, LOW);

delay(blinkDelay);

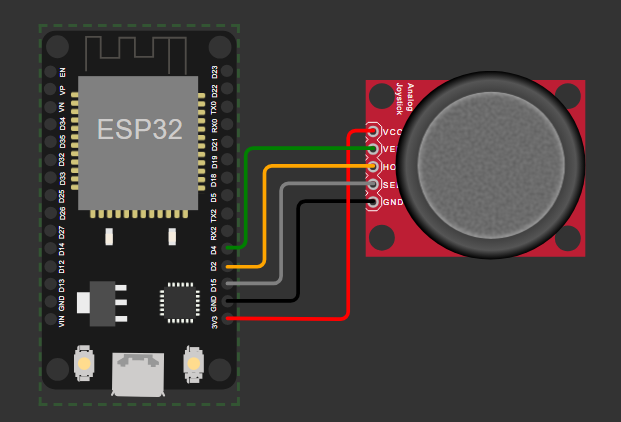
}

5.

Wokwi has an analog joystick component that provides a range of analog values

depending on the direction. Create an application that prints out the position of

the joystick



code:

const int pinX = 2; // P0 (GPIO00 - ADC11)

const int pinY = 4; // P2 (GPIO02 - ADC12)

const int pinSW = 15; // P15 (GPIO15)

int valueX = 0; // Analog

int valueY = 0; // Analog

int valueZ = 0; // Digital

void setup() {

Serial.begin(9600);

}

void loop() {

valueX = analogRead(pinX);

Serial.print("X:");

Serial.print(valueX, DEC);

valueY = analogRead(pinY);

Serial.print(" | Y:");

Serial.print(valueY, DEC);

valueZ = digitalRead(pinSW);

Serial.print(" | Z: ");

Serial.println(valueZ, DEC);

delay(100);

}

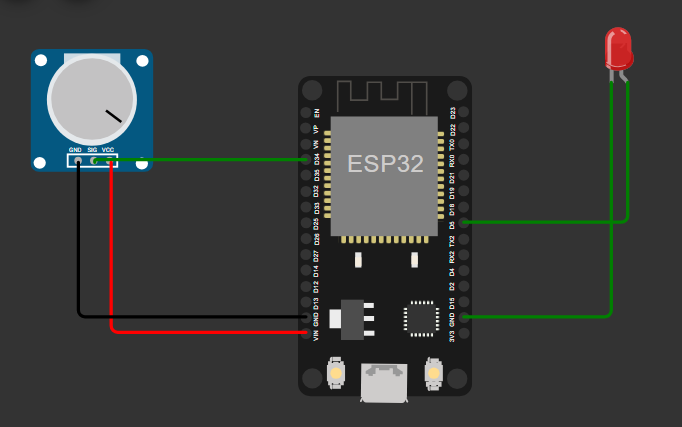
6.

Using the LED bar and the rotating potentiometer, create an application that

replicates a volume control dial. Meaning as the potentiometer dial is rotated

clockwise, an increasing amount of LEDs light up indicating higher volume.

Conversely, as the dial is rotated counter-clockwise, LEDs are turned off.



code:

#define POTENTIOMETER\_PIN 35 // ESP32 pin GPIO35 (ADC0) connected to Potentiometer pin

#define LED\_PIN 26 // ESP32 pin GPIO26 connected to LED's pin

// the setup routine runs once when you press reset:

void setup() {

// initialize serial communication at 9600 bits per second:

Serial.begin(9600);

// declare LED pin to be an output:

pinMode(LED\_PIN, OUTPUT);

}

// the loop routine runs over and over again forever:

void loop() {

// reads the input on analog pin A0 (value between 0 and 4095)

int analogValue = analogRead(POTENTIOMETER\_PIN);

// scales it to brightness (value between 0 and 255)

int brightness = map(analogValue, 0, 4095, 0, 255);

// sets the brightness LED that connects to pin 3

analogWrite(LED\_PIN, brightness);

// print out the value

Serial.print("Analog value = ");

Serial.print(analogValue);

Serial.print(" => brightness = ");

Serial.println(brightness);

delay(100);

}