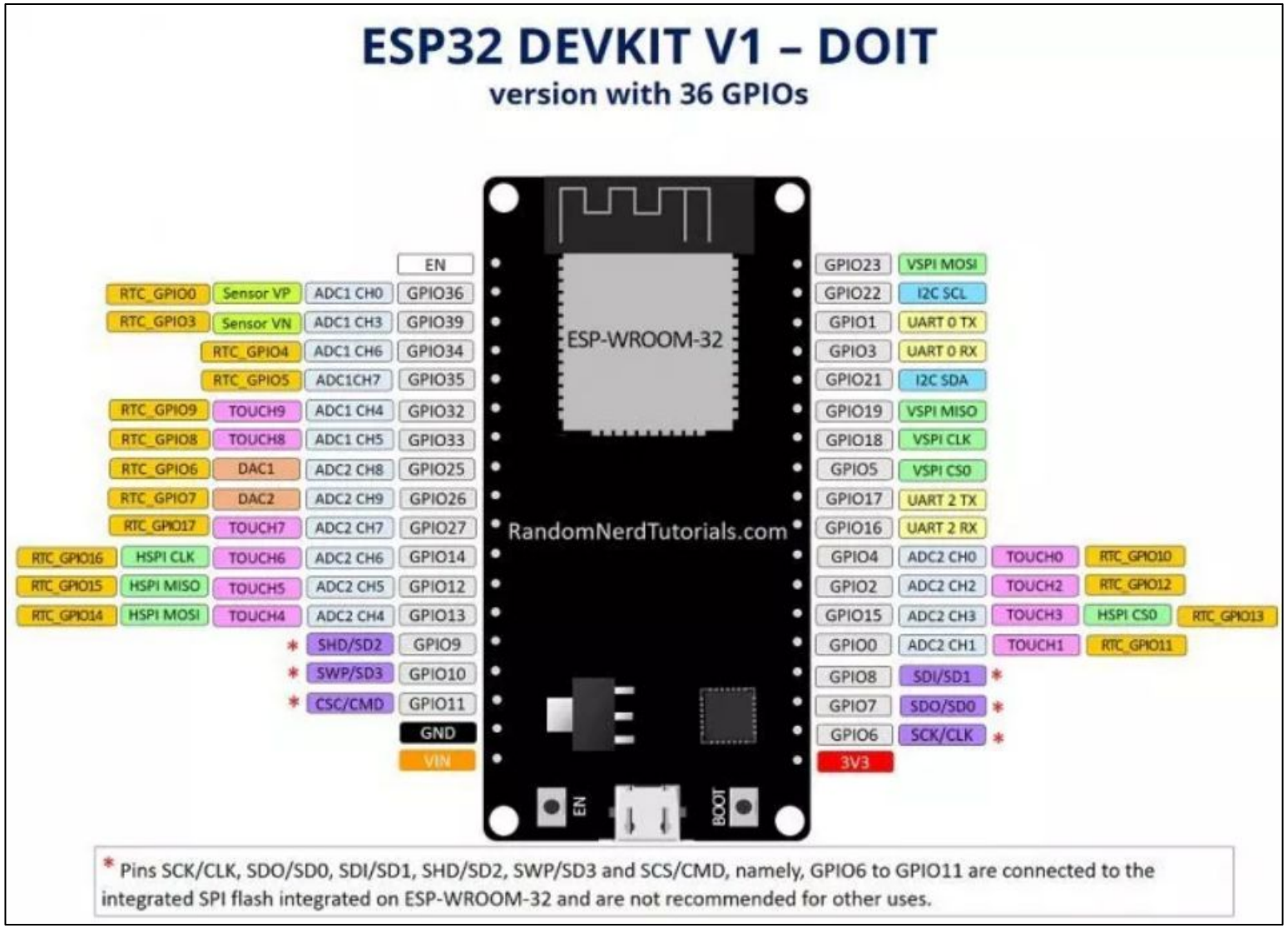
Basic Logic Circuits Using ESP32 Microcontroller

DEVKIT C

PINOUT Diagram



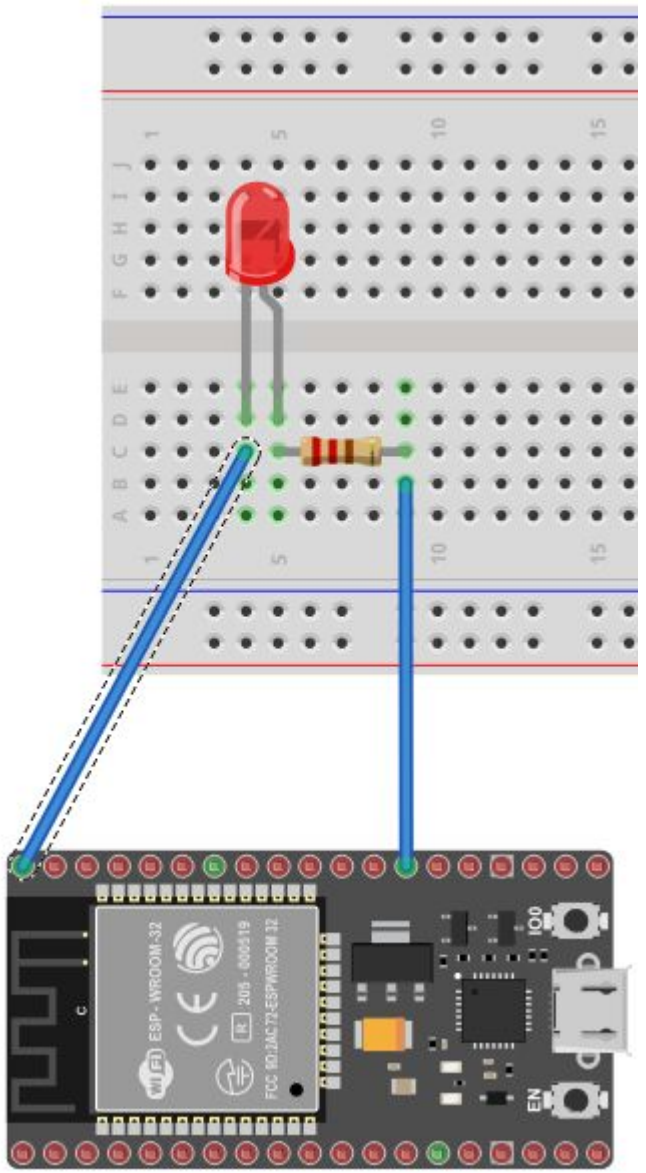
Note: The latest revision board might have more GPIOs. Those might be just for GND and 3.3V so, don’t panic.
Pin 36-39 can only be used as input pins.

PART 1 - The LED is LIT

LED Connections:

- Negative terminal of LED (shorter end) to GND (ground of ESP)
- Positive terminal of LED to one end of 1k ohm resistor.
- Pin 9 of ESP to other end of resistor.

Circuit Diagram



Coding time!!

```
/*
    Blinks on-board LED
*/
void setup()
{
  pinMode(4, OUTPUT);
}
void loop()
{
  digitalWrite(4, HIGH);
  delay(1000);      //try changing this value
  digitalWrite(4, LOW);
  delay(1000);      //try changing this value
}
```

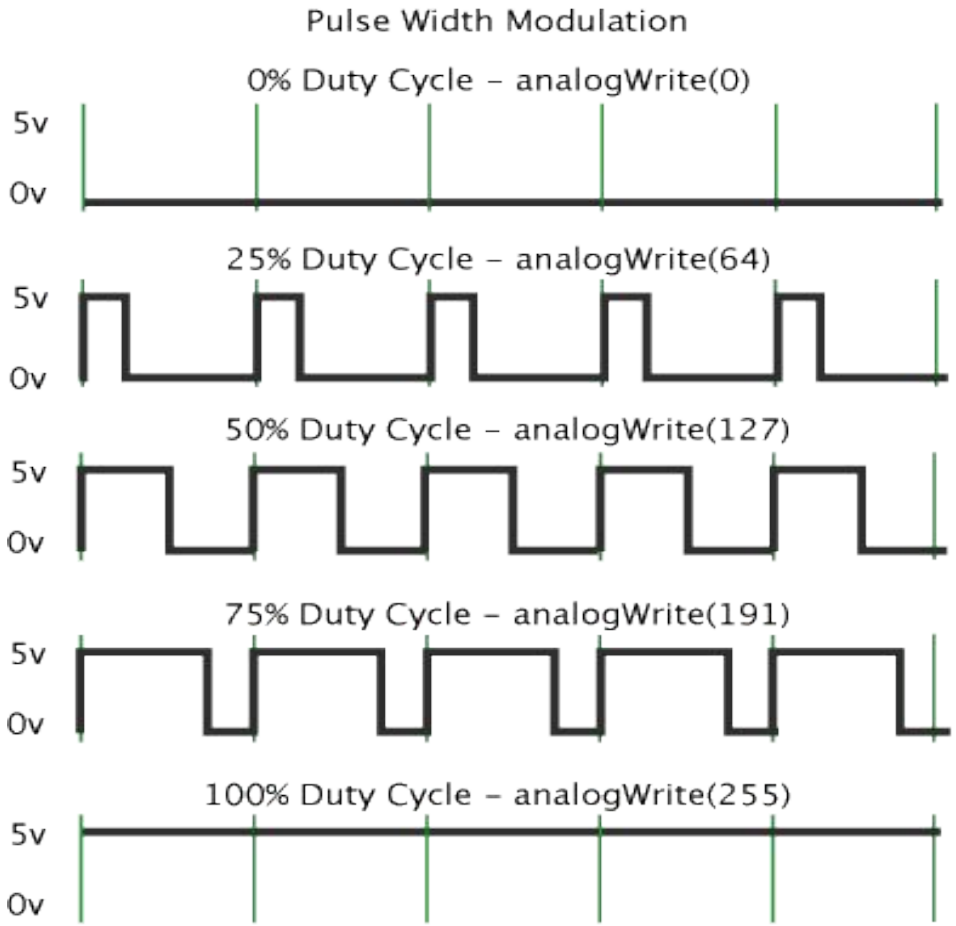
LED BLINKING

- Upload the code. The led will start blinking .
- **Delay function**:When we write delay(t), it means the code will not go to the next line of code until t milliseconds after the delay function is called.
- pinMode() states whether a pin will generate signals, or read external ones.
- **digitalWrite:**output,is either HIGH or LOW.

- Try changing the delay values and uploading the code.Observe how the LED blinks.

PART 2 - Fade In and Out

```
void setup()
{
  pinMode(4, OUTPUT);
}
void loop()
{
  for (int i=0; i<=255; i++)
  {
   analogWrite(4,i);   // you can use this function after installing the analogWrite library in IDE
   delay(50);
  }
  for (int i=255; i>=0; i--)
  {
   analogWrite(4,i);
   delay(50);
  }
}
```



Clarification on analog output

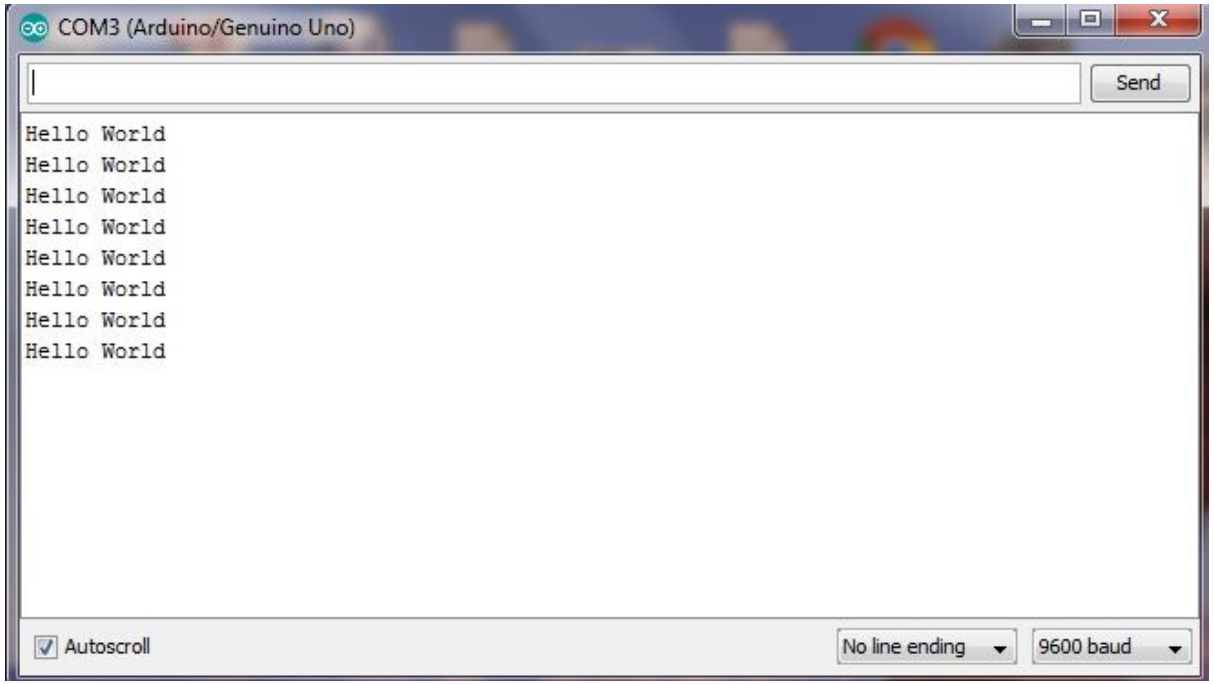
- analogWrite function for ESP-32 is currently not implemented in the Arduino IDE. You can install ESP32 analogWrite library by ERROPix through library manager.
- Instead there is a different set of functions available which do this function: ledc functions. To use them do the following things...
    - pinMode(GPIO_PIN, OUTPUT); // GPIO_PIN is the pin no on the Dev Board
    - ledcSetup(Channel, freq, resolution); // esp-32 has 16 channel ADC use any one of them
    - ledcAttachPin(GPIO_PIN, Channel);
- Add above three lines in the setup() function. Setup for PWM is now complete.
- Use the following function in loop() function: ledcWrite(Channel, dutyCycle); (Note: If you're using a resolution of 8-bits then the dutyCycle has to be less than 255.)

PART 3 – The Serial Monitor & Serial Communication

- The **Serial** library gives us a way to establish communication between the microcontroller and the computer. (Using the UART communication protocol.)
- Can be used to send back the values recorded by various sensors back to the PC.

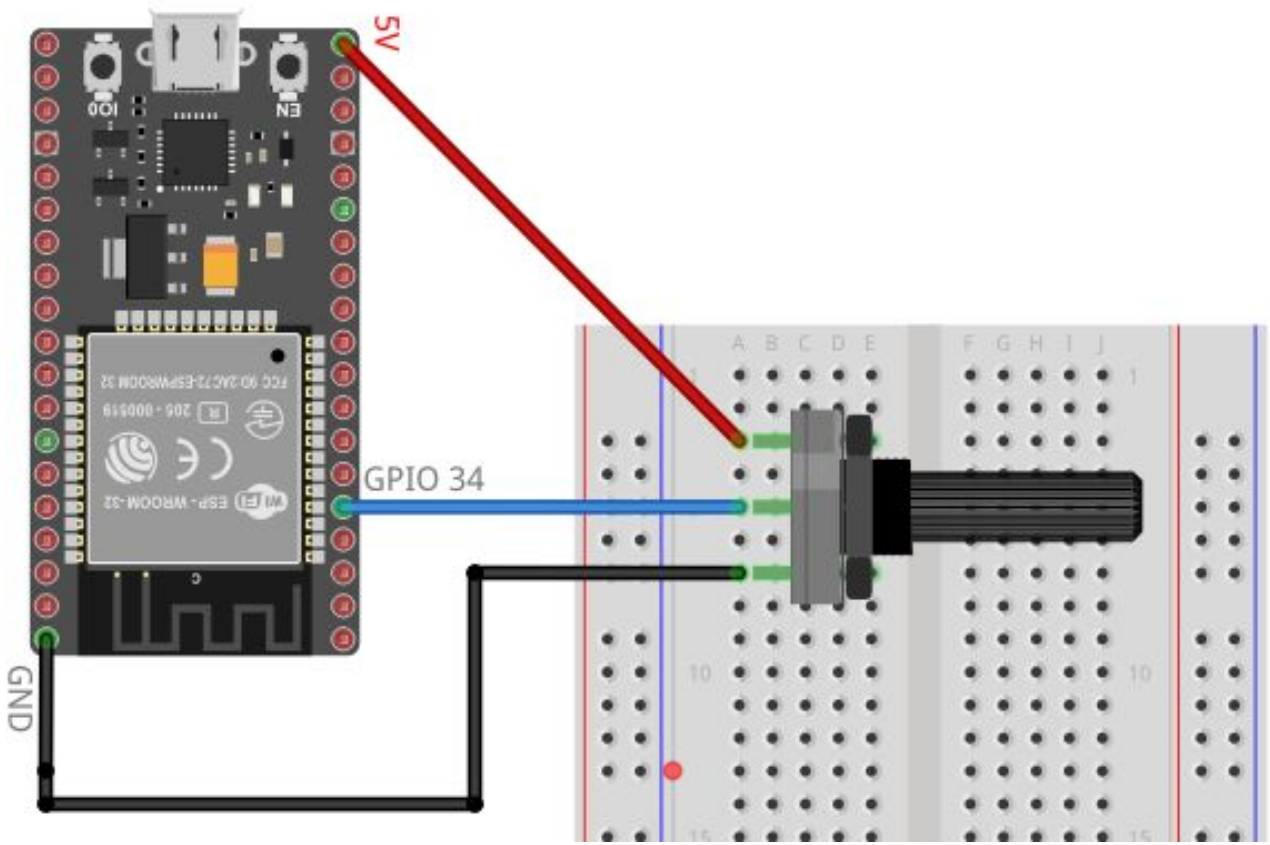
Say Hello to Arduino

```
void setup() {
    Serial.begin(9600);      // open the serial port at 9600 bps:
}

void loop(){
    Serial.println("Hello World");
}
```



PART 4 – READING AN ANALOG SIGNAL

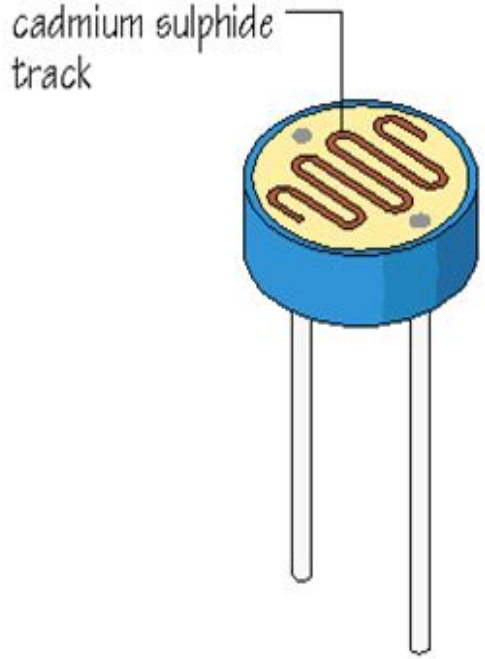
- Connect the Potentiometer according to the circuit, upload the sketch, open the serial monitor, and make sure that the correct COM port and baud rate are selected.

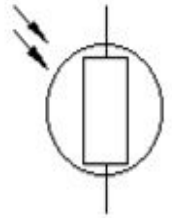


PART 5 – The Light Controlled Lamp

LDR

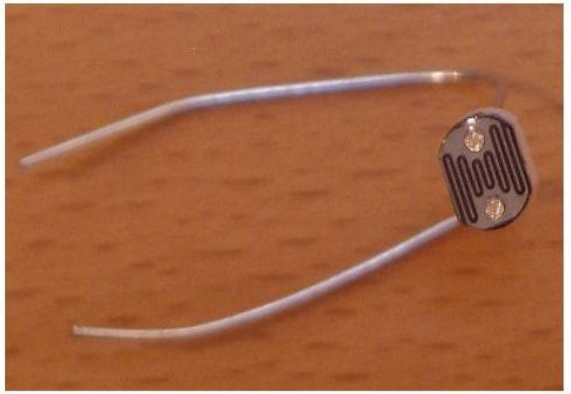
- A photoresistor (or light-dependent resistor, **LDR**, or photocell) is a light-controlled variable resistor.
- The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity.
- **Brighter light = lower resistance**





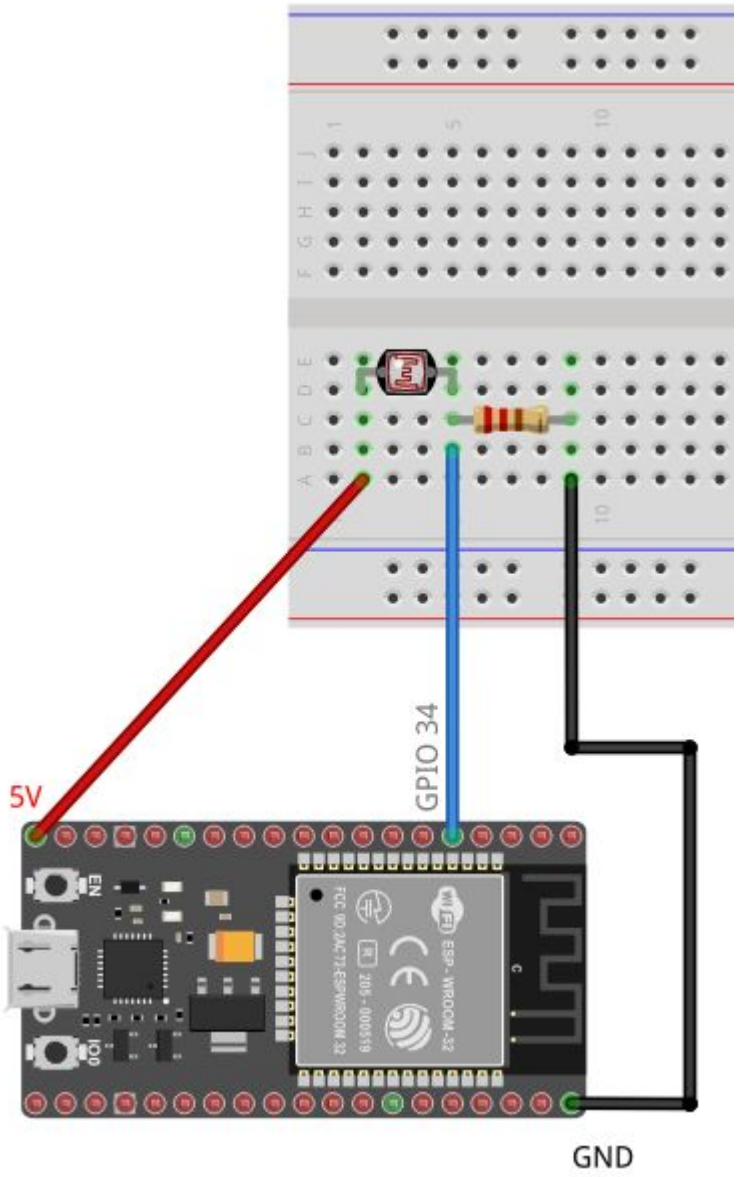
circuit symbol

schematic symbol



Creating a Voltage Divide

- Now, using the LDR, we will implement a voltage divider, which is similar to the potentiometer, except this one will be governed by a light source.
- After, connecting the circuit upload the sketch that will measure how the voltage across the LDR is varying.



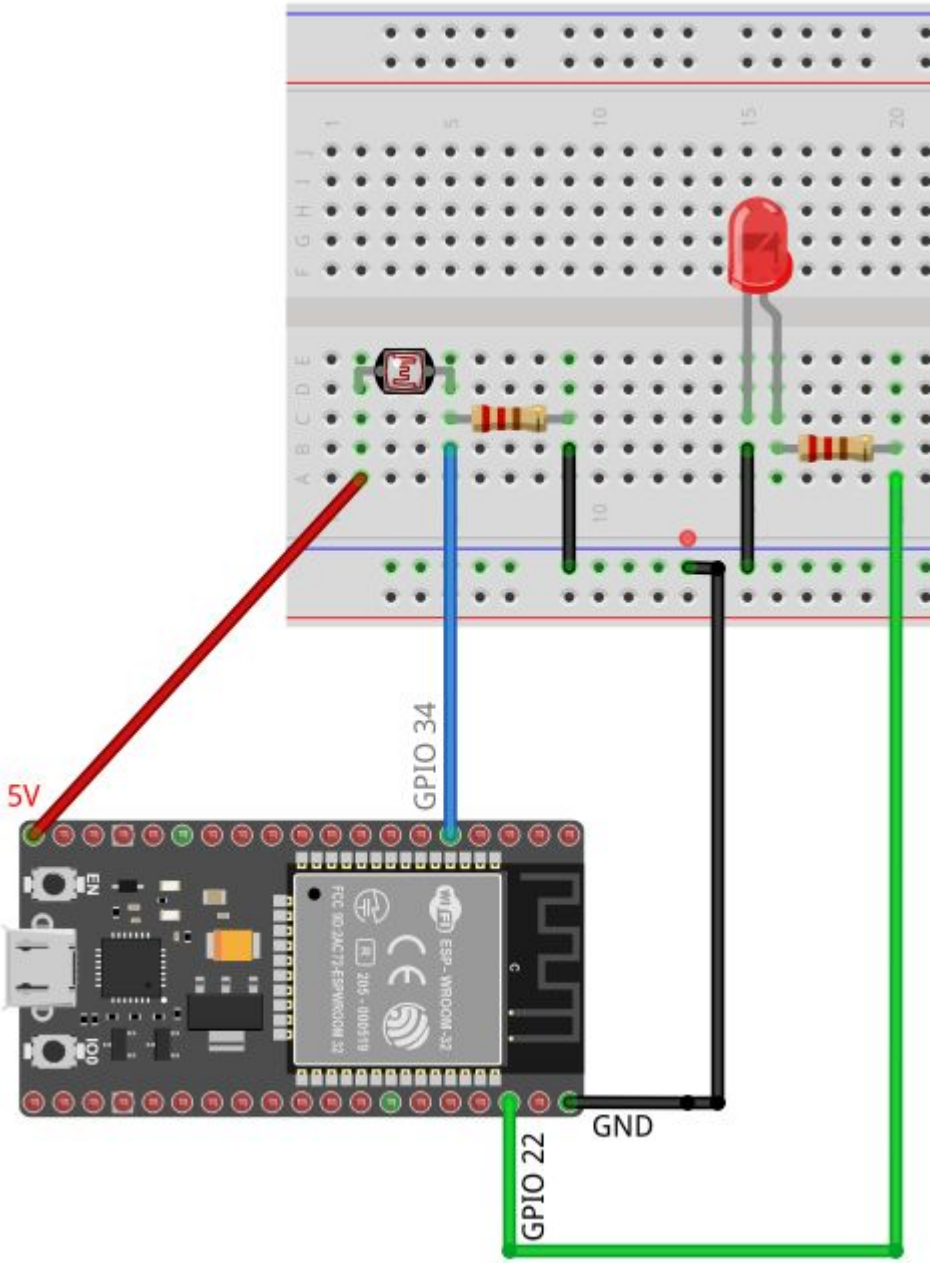
Code-1
(Testing LDR Readings)

```
void setup()
{
  Serial.begin(9600);
  pinMode(34, INPUT);
}
void loop()
{
  int ldr = analogRead(34); //values from 0-4095
  Serial.println(ldr);
}
```

- Open the Serial Monitor and see how the values on the LDR change with varying intensity of light.

Adding the LED.

- Now we add an LED, connecting the +ve terminal to Pin 22 of the ESP via a resistor.
- Upload the sketch(try writing code yourself) and control the LED by changing the lighting conditions of the LDR.

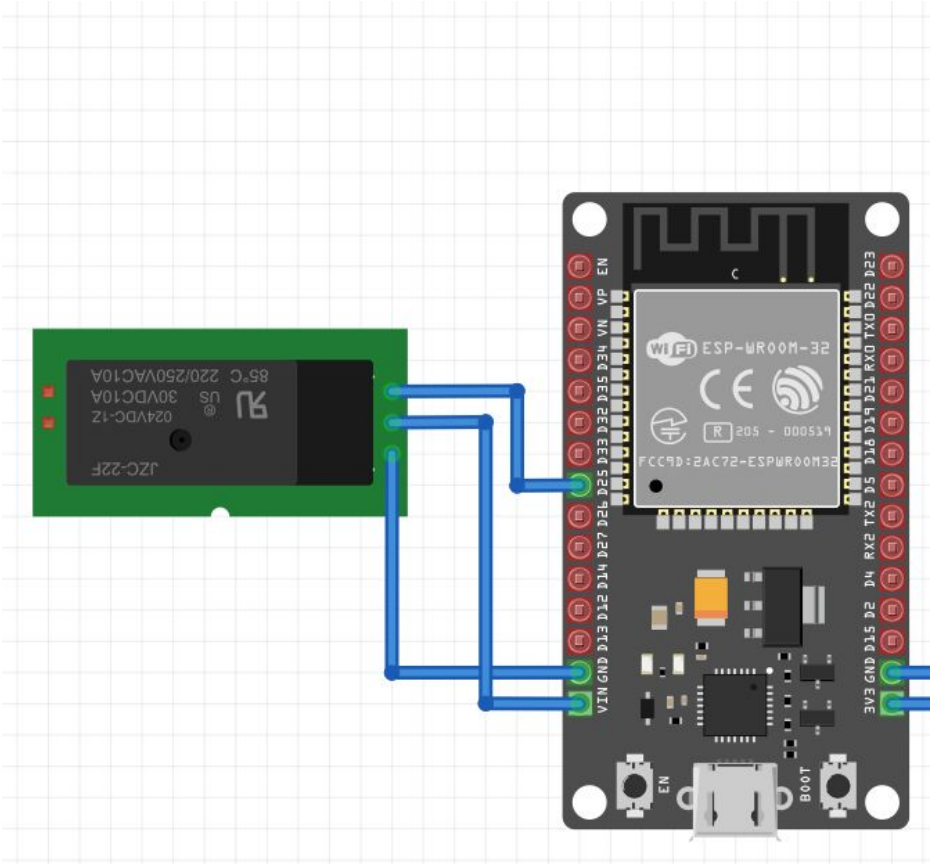


Code - 2

```
void setup()
{
     Serial.begin(9600);
     pinMode(34, INPUT);
     pinMode(22, OUTPUT);
}
void loop()
{
     int ldr = analogRead(A0); //values from 0-4095

     //Serial.println(ldr);

     int thresh = 300;
     if (ldr <= thresh) {

         digitalWrite(22, HIGH);               //turn LED on
         Serial.println("LDR is DARK, LED is ON");
     }
     else {

         digitalWrite(22, LOW);          //turn LED off
         Serial.println("---------------");
       }
}
```

Interfacing a Relay

- We will be using a Relay switch which operates on 5V.
- When a high input is given to the input pin the Relay turns on i.e. the switch closes.
- Vin of Relay is connected to one of the GPIO of ESP-32
- Vcc is connected to Vin which gives out 5V
- We have used serial input to decide the state of the relay



Code - 1

```
/*
 * In this Example I have used the serial monitor to control the behavior of the LED.
 * I didn't use the analog inputs in the other side to control the relay.
 * The Relay used in this example is a high level triggered relay.
 * AUTHOR: Harshal Deshpande
*/

#define RELAY_PIN 25

unsigned char IncomingByte;

void setup() {
  pinMode(RELAY_PIN,OUTPUT);
  digitalWrite(RELAY_PIN,LOW);
  Serial.begin(115200);
}
```

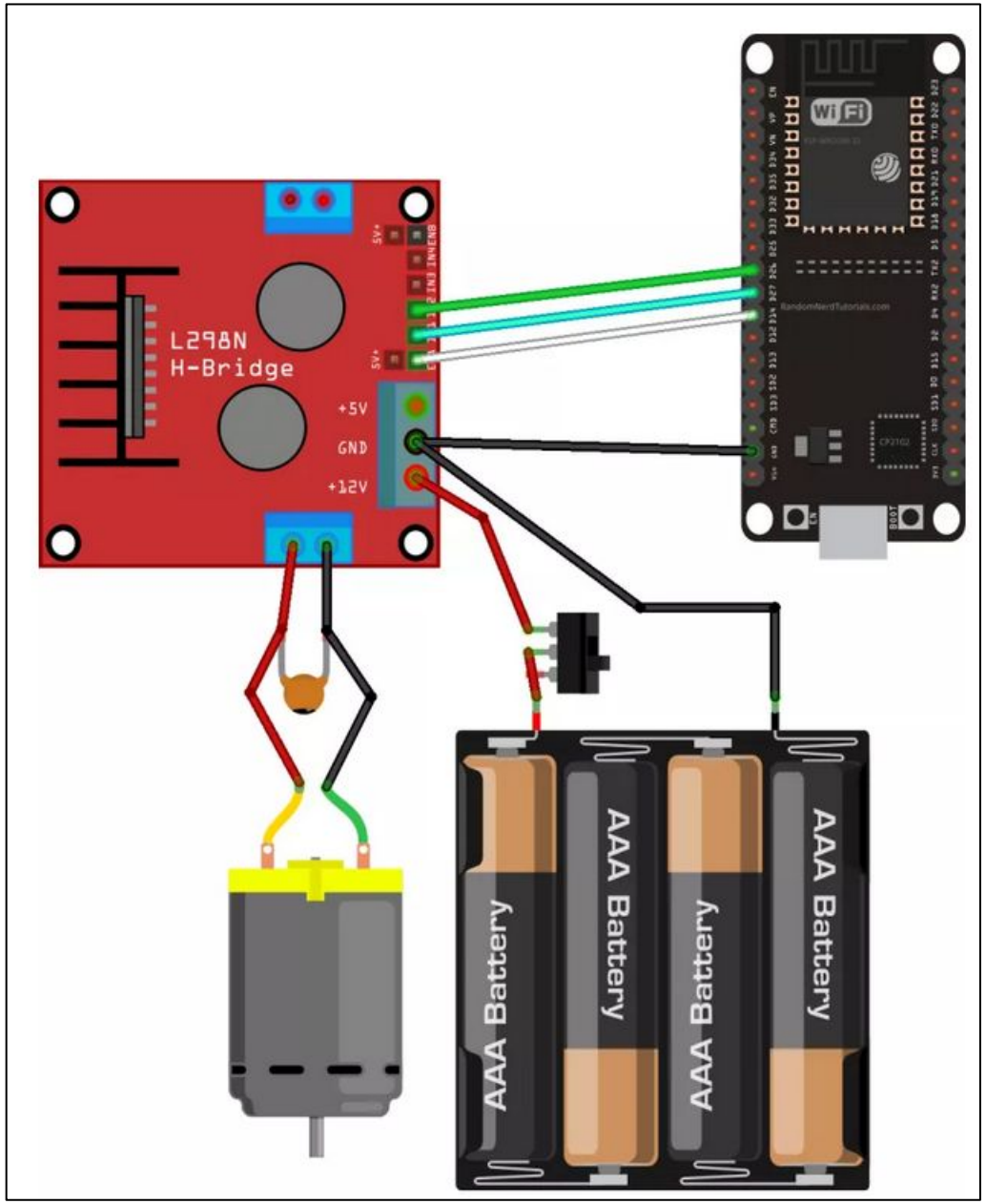
Code - 2

```
void loop() {
  if(Serial.available() > 0){
    // Similar to Internal Interrupt
    IncomingByte = Serial.read();

    if (IncomingByte == 'Y' || IncomingByte == 'y'){
      digitalWrite(RELAY_PIN,HIGH);
      Serial.println("Turning ON the RELAY");
    }
    else if (IncomingByte == 'n' || IncomingByte == 'N'){
      digitalWrite(RELAY_PIN,LOW);
      Serial.println("Turning OFF the RELAY");
    }
  }
}
```

If y is typed in the terminal then
the relay is turned on
If n is typed in the terminal then
the relay remains in the off state

Interfacing a DC motor:

- A DC Motor needs a Motor driver to control its output. In our case we will be using a L298N based motor driver. To get the details of the motor driver go through it’s datasheet.
- The DC motor is connected to OUT1 and OUT2 on the motor driver. The motor driver is powered used 4 AAA batteries.(You can also use a 3-cell lithium polymer battery)
- As shown in the schematic there are 3 wire coming from ESP to motor driver. White-Enable wire, Cyan-IN1, Green-IN2. IN1 and IN2 decide the direction of rotation of the motor and the Enable decides the speed of the rotation.
- A PWM signal is given on the Enable pin on the motor driver.



Code-1

```
#define enA 25
#define in1 26
#define in2 27

// Setting PWM Properties
const int freq = 5000;
const int Channel = 0;          // can choose any channel between 1-16
const int resolution = 8;    // no. of bits
int dutyCycle = 200;
```

Code-2

```
void setup() {
  pinMode(enA, OUTPUT);
  ledcSetup(Channel, freq, resolution);
  ledcAttachPin(enA, Channel);

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

  // Set initial rotation direction
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);

  ledcWrite(Channel, 100);

  Serial.begin(115200);
  Serial.println("Testing DC Motor.");
}
```

Code-3

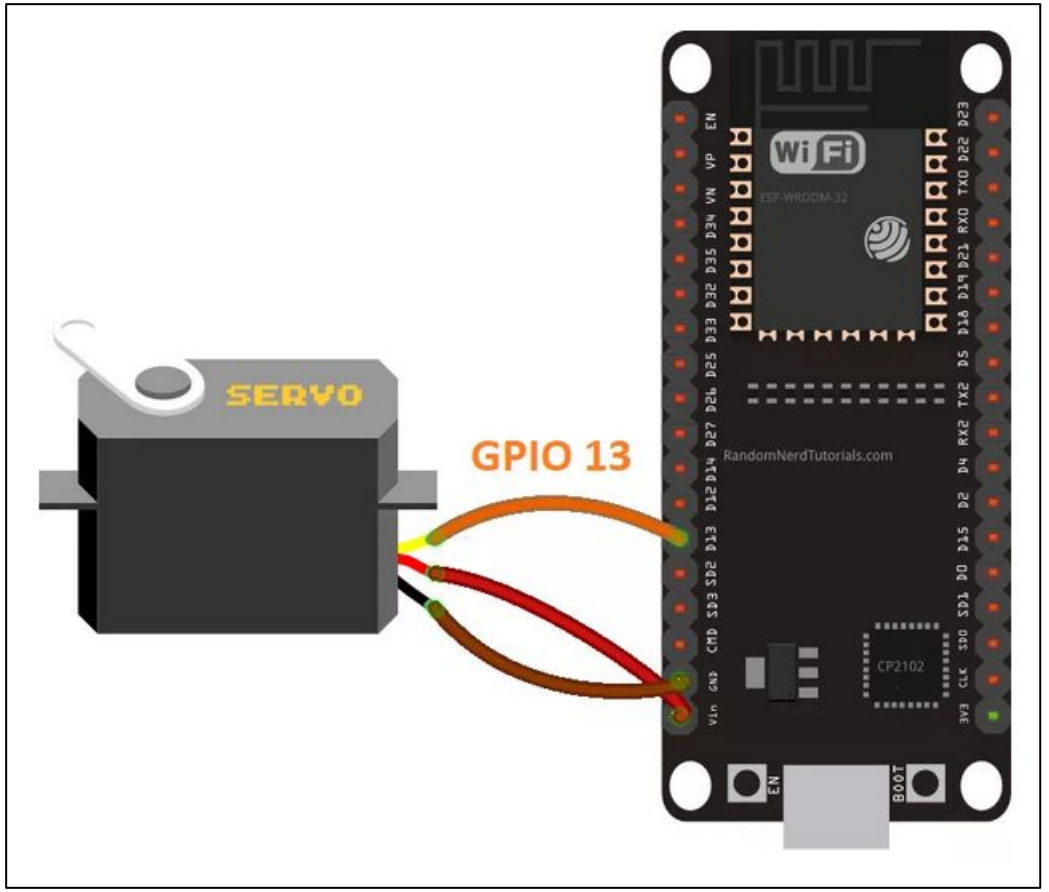
```
void loop() {

  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);

  while (dutyCycle <= 250){
    ledcWrite(Channel, dutyCycle);         // PWM on the enable pin
    Serial.print("Forward with duty cycle: ");
    Serial.println(dutyCycle);
    dutyCycle = dutyCycle + 5;
    delay(200);
  }
  dutyCycle = 100;

}
```

Code - Explanation

- The above code example is used to vary the speed of the motor continuously. The dutyCycle is constantly being varied in the loop() function.
- IN1 and IN2 fixes the direction of rotation. If both are set to low then the motor will not rotate.

Interfacing Servo:

- To interface a servo with ESP32 you will first need to install library from the following link: https://github.com/RoboticsBrno/ESP32-Arduino-Servo-Library/
- Follow the instructions given in the getting started guide to install the library.
- The following code will rotate the servo from 0to180 and 180to0 again and again.



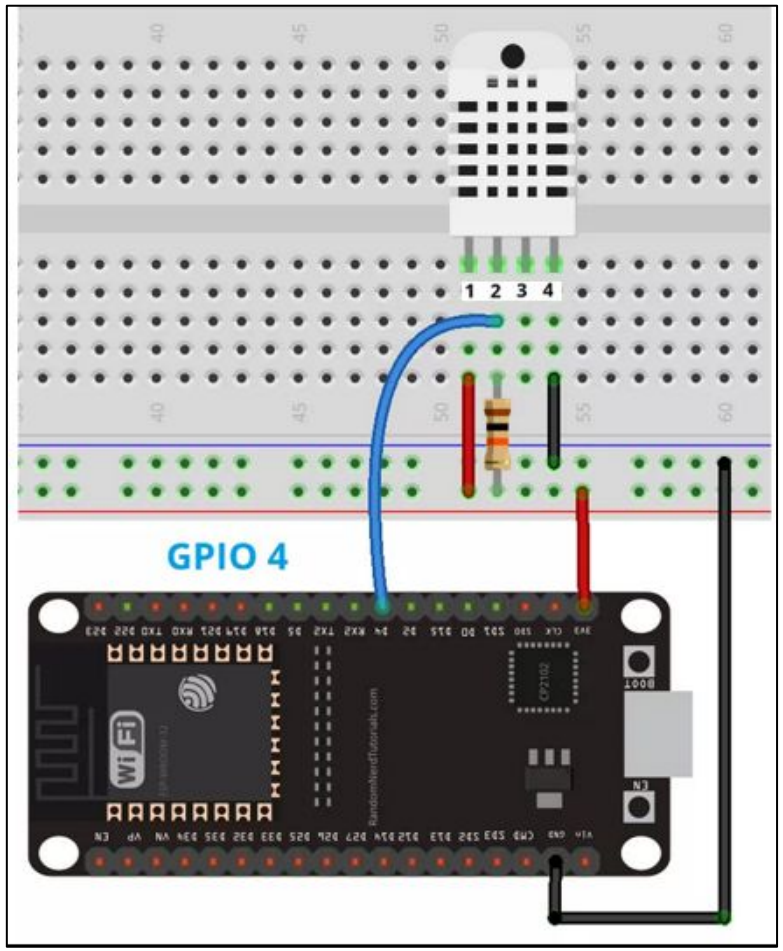
Code

```
#include <Servo.h>

#define SERVO_PIN 13

Servo MyServo;

int pos = 0;

void setup() {
  MyServo.attach(SERVO_PIN);
}




void loop() {
  for(pos = 0;pos<180;pos++){
    MyServo.write(pos);
```

Interfacing DHT-22 or DHT-11:

- The DHT22 sensor has a better resolution and a wider temperature and humidity measurement range. However, it is a bit more expensive, and you can only request readings with 2 seconds interval.
- The DHT11 has a smaller range and it's less accurate. However, you can request sensor readings every second. It's also a bit cheaper.
- To use these sensors you'll need to install the following libraries: DHT sensor library and Adafruit Unified Sensor Library which you can do through the lib

DHT pin	Connect to
1	3.3V
2	Any digital GPIO; also connect a 10k Ohm pull-up resistor
3	Don't connect
4	GND



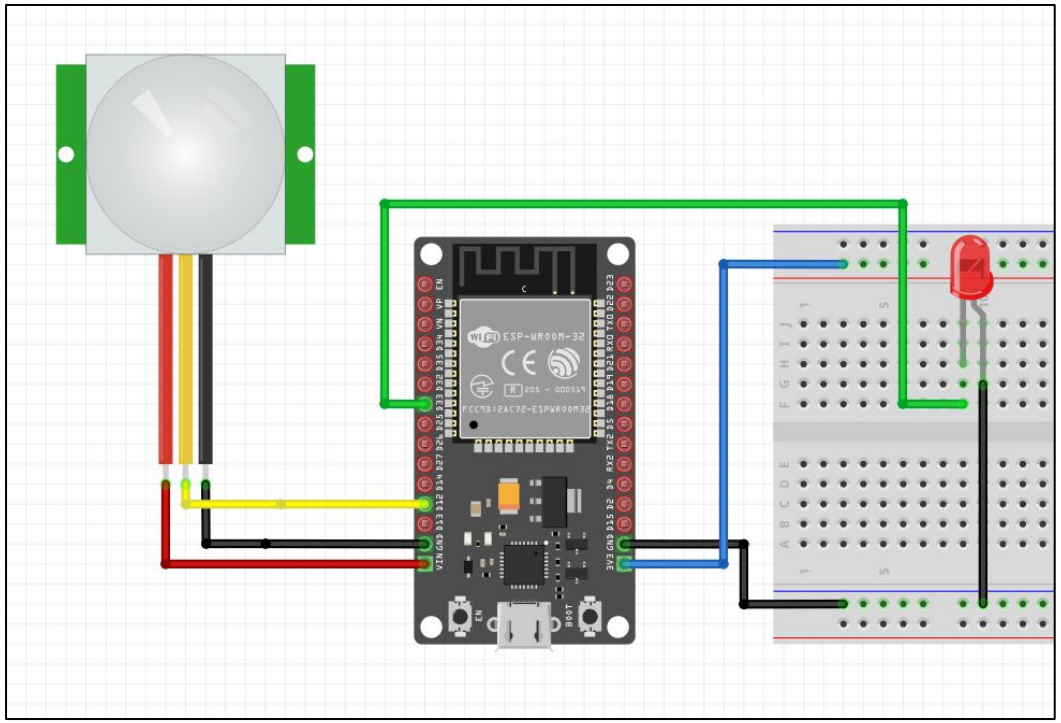
Code

```
#include <DHT.h>

#define DHT_PIN 26
#define DHTTYPE 22

DHT dht(DHT_PIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();
}
```

```
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
```

Interfacing PIR Sensor:

- PIR sensor gives high output when motion is detected by the sensor.
- The built-in potentiometers are for adjusting the delay of output(left) and the sensitivity(right).



Code-1

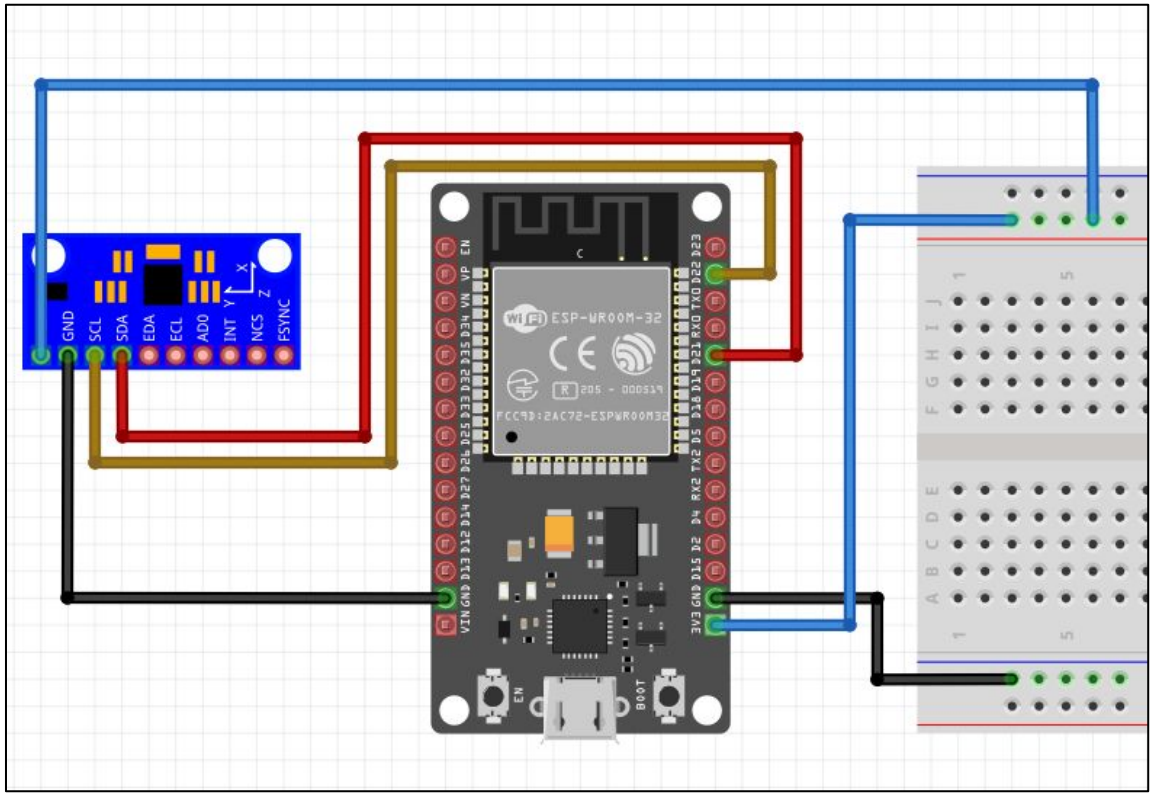
```
#define LED_PIN 33
#define SENSOR_PIN 12

int State = LOW;
int SensorVal = 0;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(SENSOR_PIN, INPUT);
  Serial.begin(115200);
}
```

Code

```
void loop(){
  SensorVal = digitalRead(SENSOR_PIN);
  if (SensorVal == HIGH) {
    digitalWrite(LED_PIN, HIGH);
    delay(100);

    if (State == LOW) {
      Serial.println("Motion detected!");
      State = HIGH;
    }
  }
  else {
      digitalWrite(LED_PIN, LOW);
      delay(200);
      if (State == HIGH){
        Serial.println("Motion stopped!");
        State = LOW;
    }
  }
}
```

Interfacing MPU-9250

- MPU9250 operates on 3.3V. So, connect Vcc to 3.3V pin on ESP-32.
- SCL on MPU is connected to SCL(pin 22) and SDA is connected to pin 21 on ESP
- To use this sensor we will be using the following library: MPU9250_asukiaaa. You can install it using the library manager



Code-1

```
#include <MPU9250_asukiaaa.h>

MPU9250_asukiaaa mySensor;

void setup() {
Serial.begin(115200);
Serial.println("started");
Wire.begin();

mySensor.setWire(&Wire);
mySensor.beginAccel();
mySensor.beginMag();
}
```

Code-2

```
void loop() {
mySensor.accelUpdate();
Serial.println("print accel values");
Serial.println("accelX: " + String(mySensor.accelX()));
Serial.println("accelY: " + String(mySensor.accelY()));
Serial.println("accelZ: " + String(mySensor.accelZ()));
Serial.println("accelSqrt: " + String(mySensor.accelSqrt()));

mySensor.magUpdate();
Serial.println("print mag values");
Serial.println("magX: " + String(mySensor.magX()));
Serial.println("maxY: " + String(mySensor.magY()));
Serial.println("magZ: " + String(mySensor.magZ()));
Serial.println("horizontal direction: " + String(mySensor.magHorizDirection()));

Serial.println("at " + String(millis()) + "ms");
delay(2000);
}
```