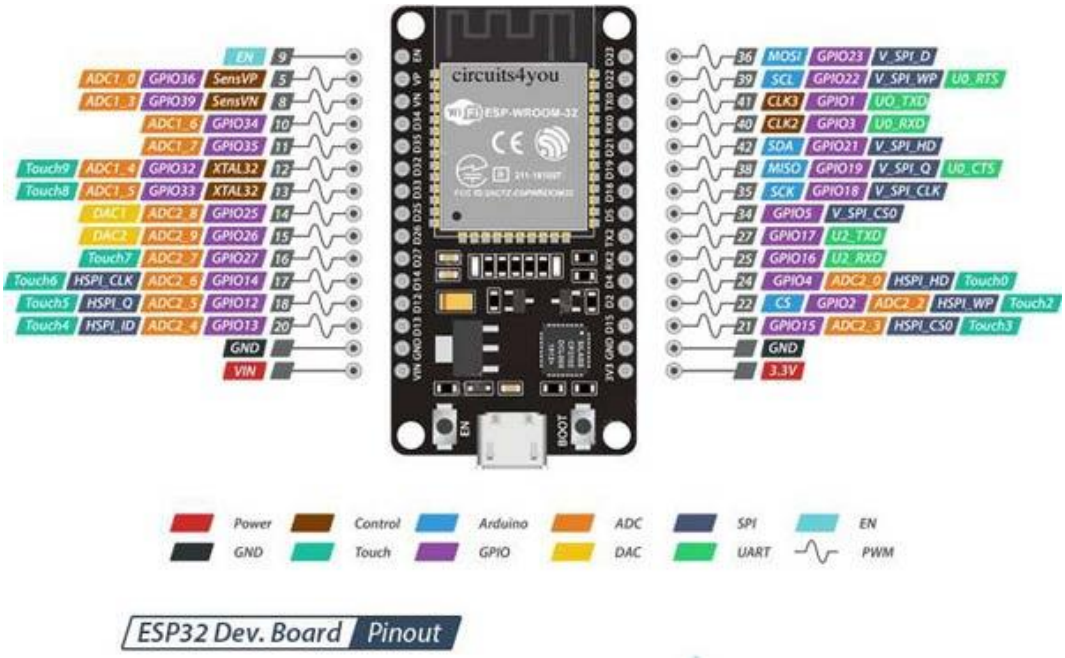
1. **NO. JOBSHEET** : 1
2. **JUDUL** : DASAR PEMROGRAMAN ESP32 UNTUK PEMROSESAN DATA INPUT/OUTPUT ANALOG DAN DIGITAL
3. **TUJUAN**
   1) Mahasiswa dapat memahami dan mengoperasikan GPIO pada ESP32.
   2) Mahasiswa dapat memahami dan melakukan pengolahan data untuk input/output analog dan digital.
   3) Mahasiswa dapat melakukan optimalisasi pembacaan sensor analog menggunakan metode regresi linear.
4. **ALAT DAN BAHAN**
   1) ESP32
   2) Breadboard
   3) Kabel jumper
   4) Potensiometer 10k Ohm (1)
   5) Sensor Capacitive Soil Moisture
   6) LED (5) dan Push Button (3)
   7) Multimeter
   8) Resistor 330,1K, 10K Ohm (@ 3)
5. **TEORI SINGKAT**

ESP-32 adalah mikrokontroler yang dikenalkan oleh Espressif System merupakan penerus dari mikrokontroler ESP8266. Pada mikrokontroler ini sudah tersedia modul WiFi dalam chip sehingga sangat mendukung untuk membuat sistem aplikasi Internet of Things. Perbedaan antara ESP32 dengan ESP8266 adalah pada bagian prosesornya. ESP32 sudah Dual-Core 32 bit, jelas lebih cepat ESP32 secara kinerja. Selain itu modul ini juga mempunyai bluetooth, satu fitur yang tidak ada di ESP8266.

Gambar 5.1 merupakan susuan pin pada modul ESP32. Pada pin out tersebut terdiri dari :

- 18 ADC (Analog Digital Converter)
- 2 DAC (Digital Analog Converter)
- 16 PWM (Pulse Width Modulation)
- 10 Sensor sentuh
- 2 jalur antarmuka UART
- pin antarmuka I2C, I2S, dan SPI




Gambar 5.1 Pinout ESP32

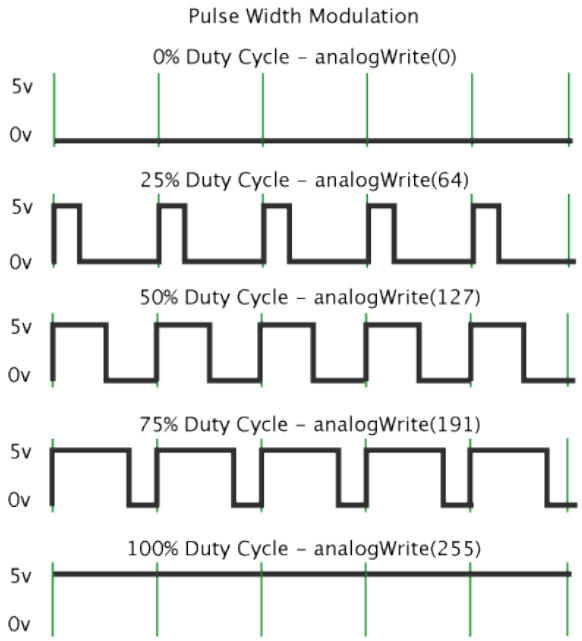
## 5.1 Pulse Wide Modulation (PWM)

Pin analog pada ESP32 (dan mikrokontroller lain pada umumnya) dapat digunakan sebagai *input* dan *output* digital. Hanya saja pin analog memiliki fitur untuk dapat mengubah sinyal analog yang masuk menjadi nilai digital yang mudah diukur. Pin digital hanya dapat mengenali sinyal 0 volt sebagai nilai *LOW* dan 3,3 volt sebagai nilai *HIGH*. Sedangkan pin analog dapat mengenali sinyal pada rentang nilai voltase tersebut.

Pin analog terhubung dengan *converter* pada mikrokontroller yang dikenal dengan istilah *analog-to-digital converter* (disingkat ADC atau A/D). *Converter* ini mengubah nilai analog berbentuk sinyal voltase ke dalam bentuk digital/angka supaya nilai analog ini dapat digunakan dengan lebih mudah dan aplikatif. ESP32 didukung perangkat keras yang mampu membaca input channel ADC hingga resolusi 12 bit. Hal tersebut berarti bahwa ESP32 mampu mendapatkan pembacaan analog mulai dari 0 hingga 4095, di mana 0 sesuai dengan 0V dan 4095 hingga 3.3V. Resolusi channel ADC tersebut dapat dikonversi juga menjadi lebih kecil menggunakan kode dan rentang ADC pada program.

Analog *output* pada *microkontroller* dihasilkan oleh teknik yang dikenal dengan istilah PWM atau *Pulse Width Modulation*. PWM memanipulasi keluaran

digital sedemikian rupa sehingga menghasilkan sinyal analog. Metode PWM menggunakan pendekatan perubahan lebar pulsa untuk menghasilkan nilai tegangan analog yang diinginkan. Pin yang difungsikan sebagai PWM analog *output* akan mengeluarkan sinyal pulsa digital dengan frekuensi 5000 Hz yang mana nilai tegangan analog diperoleh dengan mengubah *duty cycle* atau perbandingan lamanya pulsa HIGH terhadap periode (T) dari sinyal digital tersebut. Mikrokontroler melakukan pengaturan *output* digital ke *HIGH* dan *LOW* bergantian dengan porsi waktu tertentu untuk setiap nilai keluarannya. Durasi waktu untuk nilai *HIGH* disebut *pulse width* atau panjang pulsa. Hal tersebut dapat dilihat pada Gambar 5.2.




Gambar 5.2. Duty cycle pada PWM
Sumber : www.arduino.cc

Kondisi *HIGH* adalah kondisi ketika sinyal berada di atas grafik (3,3V) dan *LOW* adalah ketika sinyal berada di bawah (0V). *Duty cycle* adalah persentasi panjang pulsa *HIGH* dalam satu periode sinyal. Ketika *duty cycle* 0% atau sinyal *LOW* penuh, maka nilai analog yang dikeluarkan adalah 0V atau setara dengan GND. Jika pulsa *HIGH* muncul selama setengah dari periode sinyal, maka *duty cycle* yang dihasilkan adalah 50% yang berarti sinyal analog yang dihasilkan sebesar setengah dari tegangan analog maksimal yaitu 1/2 dari 3,3 V atau sama

dengan 1,65 V. Ketika *duty cycle* 100% atau sinyal penuh maka sinyal yang dikeluarkan adalah 3.3V.

### 5.2 Regresi Linear

Regresi analisis adalah teknik statistika untuk menginvestigasi dan memodelkan hubungan antara variabel dari data statistik sebelumnya. Pengaplikasian regresi cukup banyak dan terjadi hampir di banyak bidang seperti bidang keteknikan, ilmu fisika dan kimia, ekonomi, manajemen, ilmu biologi dan ilmu sosial. Bahkan analisis regresi mungkin lebih banyak digunakan dalam teknik statistik.

Regresi dapat dibedakan menjadi tiga jenis, yaitu regresi linier, regresi multi linier dan regresi tak linier. Di dalam model regresi linier terdapat dua jenis variabel yaitu variabel bebas atau input tegangan (X) dan variabel tak bebas atau output sensor (Y). Dalam bentuk yang paling sederhana, regresi linier direpresentasikan pada persamaan (5.1).

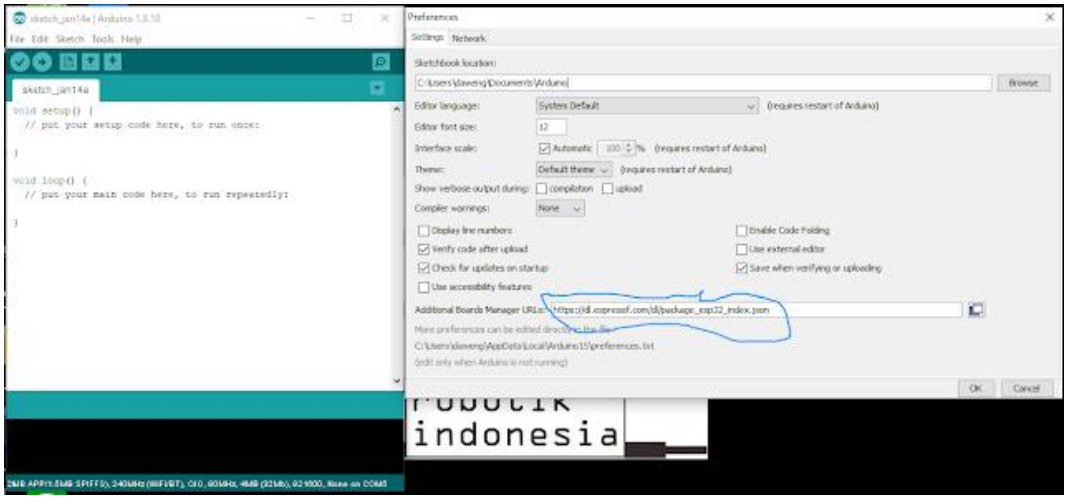
$$Y = A + Bx \tag{5.1}$$

Di mana $A$ disebut sebagai sumbu awal dan $B$ adalah koefisien arah atau koefisien beta.

## 6. LANGKAH PERCOBAAN

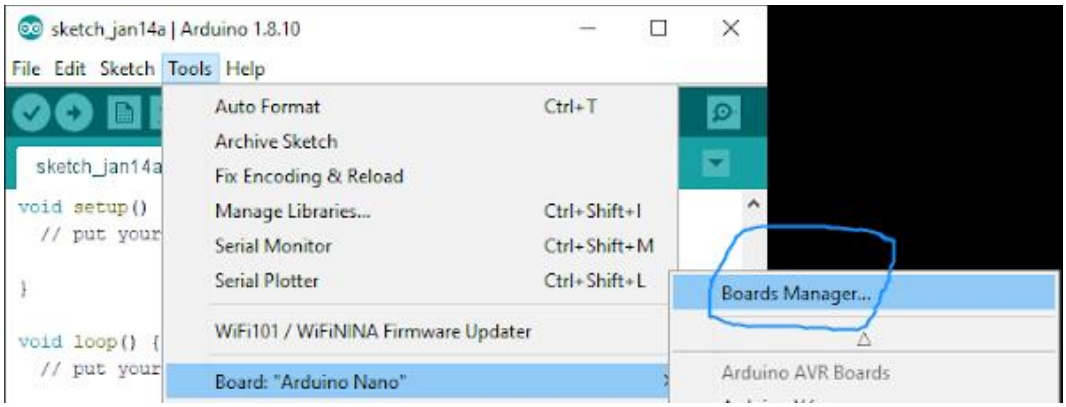
### A. Instalasi Board ESP32 pada Arduino IDE

1. Buka Arduino IDE
2. Kemudian klik Menu File > Preferences

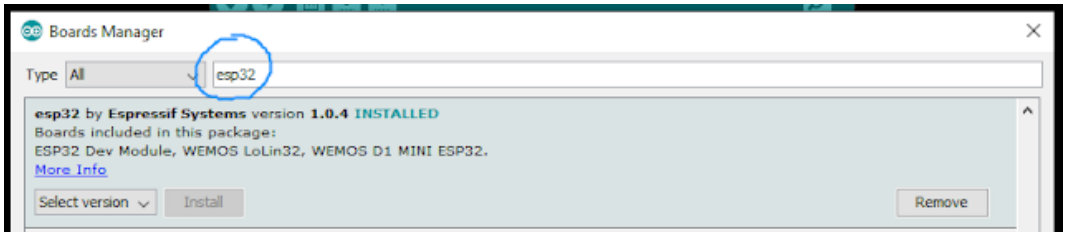


3. Pada kolom Additional ... yang ada dibawah, tambahkan link berikut

   https://dl.espressif.com/dl/package_esp32_index.json
4. Klik menu Tools > Board: > Pilih Boards Manager ...



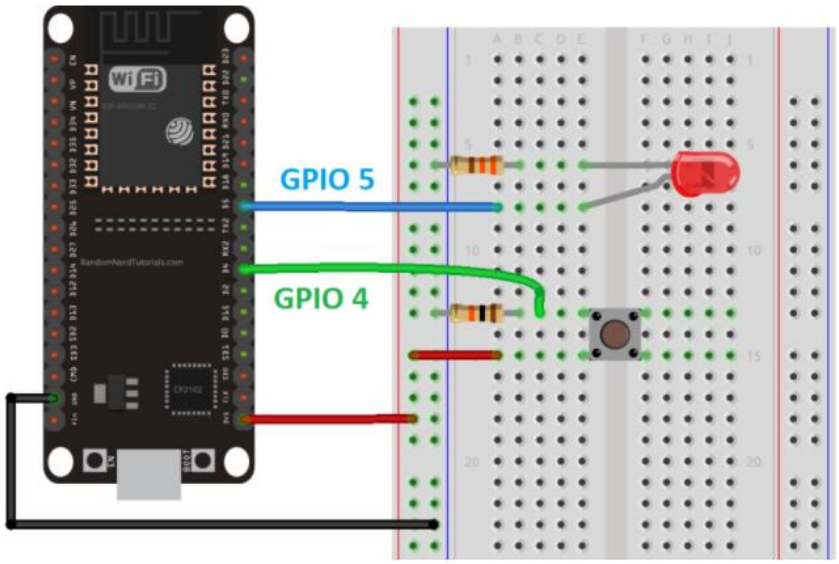

5. Pada kolom pencarian tulis ESP32 kemudian install dan tunggu sampai selesai.




## B. Mengakses GPIO dan PWM ESP32

### a) GPIO

1. Buatlah rangkaian seperti pada Gambar di bawah ini.



2. Buka program example blink, kemudian modifikasi dan buat agar LED dapat melakukan blink dengan interval 100ms, 1 detik, 2 detik dan 3 detik sekali. Setelah itu, buatlah program agar LED dapat blink 1 detik sekali menggunakan timer milis(). Dokumentasikan hasilnya.
3. Buatlah program seperti pada script di bawah ini untuk mengendalikan led menggunakan push button. Kemudian upload program tersebut pada ESP32 dan dokumentasikan hasilnya.

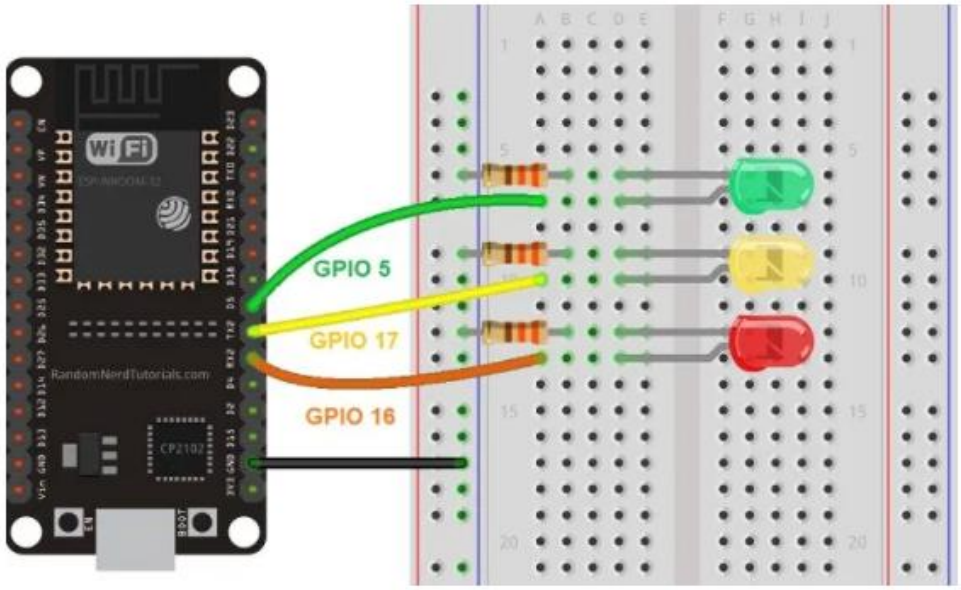
```
// set pin numbers
const int buttonPin = 4;  // the number of the pushbutton pin
const int ledPin =  5;    // the number of the LED pin

// variable for storing the pushbutton status
int buttonState = 0;

void setup() {
  Serial.begin(115200);
  // initialize the pushbutton pin as an input
  pinMode(buttonPin, INPUT);
  // initialize the LED pin as an output
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the state of the pushbutton value
  buttonState = digitalRead(buttonPin);
  Serial.println(buttonState);
  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH
  if (buttonState == HIGH) {
    // turn LED on
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off
    digitalWrite(ledPin, LOW);
  }
}
```

4. Tambahkan 1 LED dan 1 push button pada rangkaian, kemudian kembangkan program agar ketika push button ke-2 ditekan, LED akan melakukan blink setiap 500 ms sekali. Kemudian dokumentasikan hasilnya.

5. Tambahkan 3 LED dan 1 push button pada rangkaian, kemudian kembangkan program agar ketuka push button ke-3 ditekan, LED akan menyala menjadi running led (menyala bergantian dari kiri ke kanan). Setelah itu dokumentasikan hasilnya.

**2) PWM**

1. Buatlah rangkaian seperti pada gambar di bawah ini.




2. Buatlah script program seperti berikut.

```
// the number of the LED pin
const int ledPin = 16;  // 16 corresponds to GPIO16

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0; //PWM Channel
const int resolution = 8; //resolution bit

void setup(){
  // configure LED PWM functionalitites
  ledcSetup(ledChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(ledPin, ledChannel);
}

void loop(){
  // increase the LED brightness
```

```
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }

  // decrease the LED brightness
  for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }
}
```

3. Upload program tersebut, kemudian amati dan analisis apa yang terjadi serta dokumentasikan hasilnya.
4. Buatlah program lanjutan seperti pada script berikut ini.

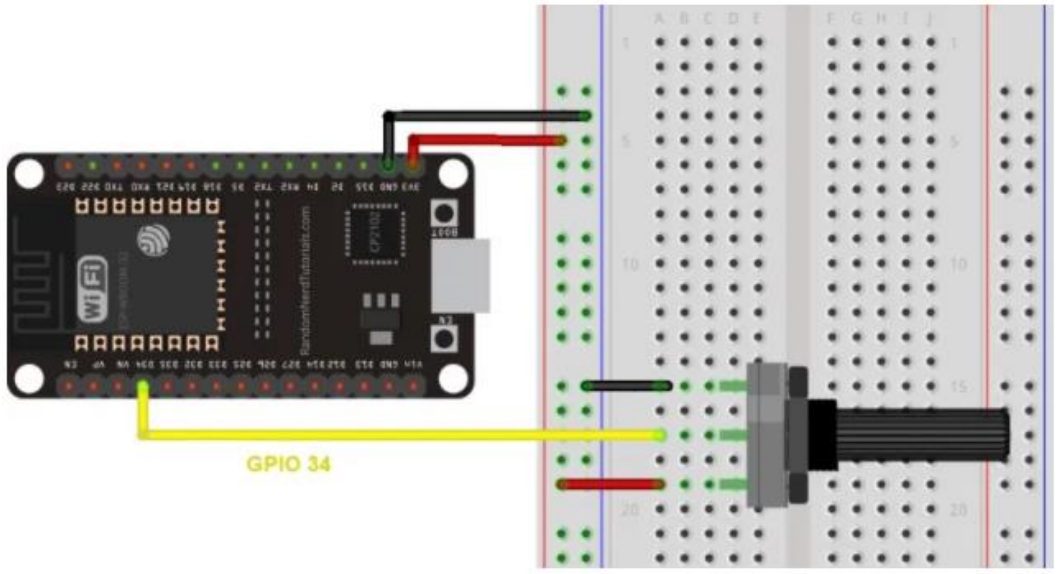
```
// the number of the LED pin
const int ledPin = 16;  // 16 corresponds to GPIO16
const int ledPin2 = 17; // 17 corresponds to GPIO17
const int ledPin3 = 5;  // 5 corresponds to GPIO5

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

void setup(){
  // configure LED PWM functionalitites
  ledcSetup(ledChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(ledPin, ledChannel);
  ledcAttachPin(ledPin2, ledChannel);
  ledcAttachPin(ledPin3, ledChannel);
}

void loop(){
  // increase the LED brightness
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
    // changing the LED brightness with PWM
```

```
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }

  // decrease the LED brightness
  for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }
}
```

5. Upload program tersebut, kemudian amati dan analisis apa yang terjadi serta dokumentasikan hasilnya.

## C. ADC dan DAC

1. Buatlah rangkaian seperti pada gambar di bawah ini.




2. Buatlah program seperti pada script berikut ini.

```
// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;

// variable for storing the potentiometer value
int potValue = 0;

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

```
  delay(1000);
}

void loop() {
  // Reading potentiometer value
  potValue = analogRead(potPin);
  Serial.println(potValue);
  delay(500);
}
```

3. Putar potensiometer secara perlahan agar mendapatkan nilai 0 hingga 4095 pada tampilan serial monitor. Analisis apa yang terjadi dan dokumentasikan hasilnya.
4. Buatlah program seperti pada script berikut ini.Tambahkan LED pada GPIO 5.

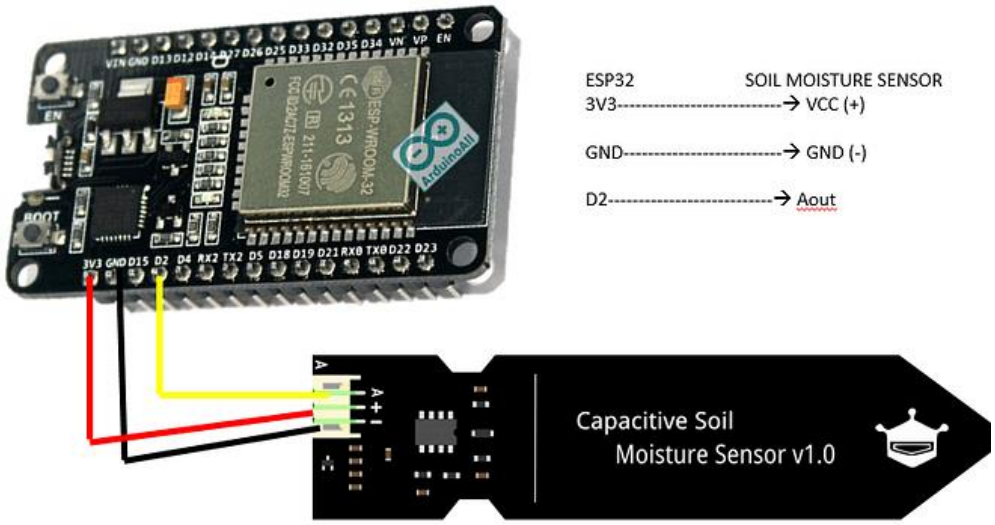
```
// These constants won't change. They're used to give names to the pins used:
const int analogInPin = 34;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 5; // Analog output pin that the LED is attached to

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  Serial.begin(115200); // initialize serial communications at 115200 bps:

// configure LED PWM functionalitites
  ledcSetup(ledChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(analogOutPin, ledChannel);
}

void loop() {
  sensorValue = analogRead(analogInPin); // read the analog in value:
  outputValue = map(sensorValue, 0, 4095, 0, 255); // map it to the range of the analog out:
  analogWrite(analogOutPin, outputValue); // change the analog out value:
```

```
  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(2);
}
```

5. Upload program, kemudian putar potensiometer dari nilai terendah hingga nilai tertinggi. Amati yang terjadi, analisis dan dokumentasikan hasilnya.

## D. Simulasi Pemrosesan Data Menggunakan Regresi Linier

1. Buatlah rangkaian seperti pada Gambar di bawah ini.




2. Buatlah dan upload script program berikut ini.

```
const int numReadings = 10;

int readings[numReadings];      // the readings from the analog input
int readIndex = 0;              // the index of the current reading
int total = 0;                  // the running total
int average = 0;                // the average

int inputPin = 2;

void setup() {
```

```
  // initialize serial communication with computer:
  Serial.begin(9600);
  // initialize all the readings to 0:
  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
    readings[thisReading] = 0;
  }
}

void loop() {
  // subtract the last reading:
  total = total - readings[readIndex];
  // read from the sensor:
  readings[readIndex] = analogRead(inputPin);
  // add the reading to the total:
  total = total + readings[readIndex];
  // advance to the next position in the array:
  readIndex = readIndex + 1;

  // if we're at the end of the array...
  if (readIndex >= numReadings) {
    // ...wrap around to the beginning:
    readIndex = 0;
  }

  // calculate the average:
  average = total / numReadings;
  // send it to the computer as ASCII digits
  Serial.println(average);
  delay(1);        // delay in between reads for stability
}
```

3. Masukkan sensor pada sampel tanah 1-3.
4. Catat hasil pembacaan sensor dan 3 Way Meter pada masing-masing sampel tanah.
5. Ukur tegangan output sensor pada pin Aout (A) menggunakan multimeter. Kemudian catat hasilnya.
6. Upload program berikut pada ESP32.

```
const int numReadings = 10;

int readings[numReadings];      // the readings from the analog input
int readIndex = 0;              // the index of the current reading
int total = 0;                  // the running total
```

```
int average = 0;                // the average
float vs = 0; //sensor voltage

int inputPin = 2;

void setup() {
  // initialize serial communication with computer:
  Serial.begin(9600);
  // initialize all the readings to 0:
  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
    readings[thisReading] = 0;
  }
}

void loop() {
  // subtract the last reading:
  total = total - readings[readIndex];
  // read from the sensor:
  readings[readIndex] = analogRead(inputPin);
  // add the reading to the total:
  total = total + readings[readIndex];
  // advance to the next position in the array:
  readIndex = readIndex + 1;

  // if we're at the end of the array...
  if (readIndex >= numReadings) {
    // ...wrap around to the beginning:
    readIndex = 0;
  }

  // calculate the average:
  average = total / numReadings;
  // send it to the computer as ASCII digits
  Serial.println(average);
  delay(1);        // delay in between reads for stability

vs = (3.3/4095)* average;
Serial.println(vs);
}
```

7. Catat keluaran dari vs.
8. Masukkan semua data pada tabel Excel menggunakan format berikut ini.

| 3 Way Meter | ADC | V out | Vs |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

9. Buatlah scatter chart dengan sumbu X = Vs dan sumbu Y = 3 Way Meter.
10. Klik kanan line pada scatter chart, kemudian pilih add trendline. Kemudian pada format trendline, atur trendline option = linear, checklist pada Display Equaiton on Chart dan Display R-squared value on Chart.
11. Masukkan persamaan yang dihasilkan grafik ke dalam program Arduino. Ganti variabel a dan b dengan nilai pada persamaan yang dihasilkan.
12. Dokumentasikan hasilnya, serta buatlah analisis dan kesimpulannya.

```
int average = 0;                // the average
float vs = 0; //sensor voltage
float moisture=0;

int inputPin = 2;

void setup() {
  // initialize serial communication with computer:
  Serial.begin(9600);
  // initialize all the readings to 0:
  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
    readings[thisReading] = 0;
  }
}

void loop() {
  // subtract the last reading:
  total = total - readings[readIndex];
  // read from the sensor:
  readings[readIndex] = analogRead(inputPin);
  // add the reading to the total:
  total = total + readings[readIndex];
  // advance to the next position in the array:
  readIndex = readIndex + 1;

  // if we're at the end of the array...
```

```
  if (readIndex >= numReadings) {
    // ...wrap around to the beginning:
    readIndex = 0;
  }

  // calculate the average:
  average = total / numReadings;
  // send it to the computer as ASCII digits
  Serial.println(average);
  delay(1);        // delay in between reads for stability

vs = (3.3/4095)* average;
moisture = ax + b;
Serial.println(vs);
Serial.println(moisture);
}
```

## 7. PERTANYAAN DAN TUGAS