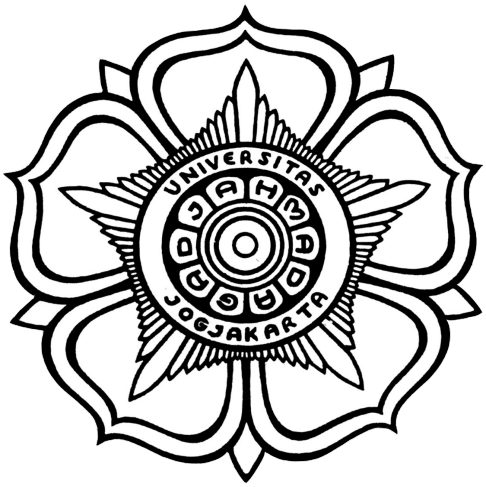
LAPORAN TUGAS KELOMPOK

PRAKTIK TEKNIK KENDALI DAN MESIN LISTRIK

“Modul ADC & DAC Mikrokontroler ESP32**”**



Disusun Oleh:

Kelompok 12/ARM 2

Mario Rusandira Wijaya (19/447118/SV/16837)

Muhammad Taufik Hermawan (19/447314/SV/17008)

DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2022

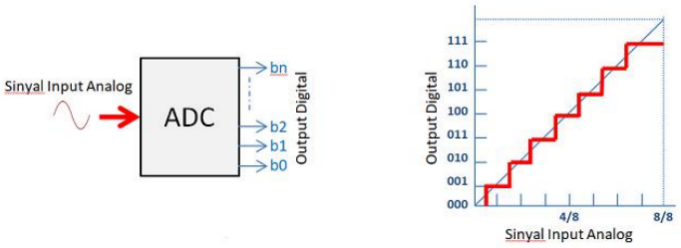
BAB I
PENDAHULUAN

A. Latar Belakang

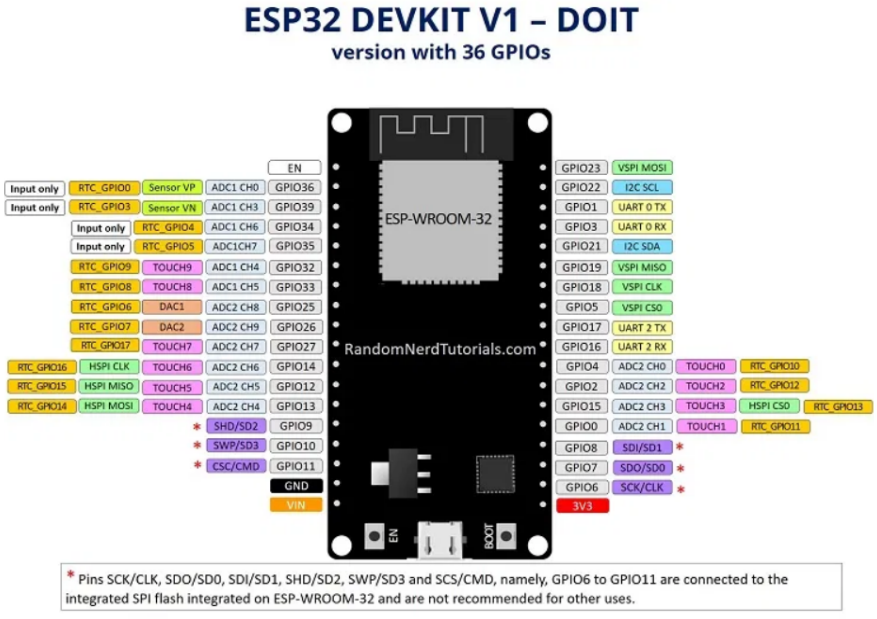
1. Analog to Digital Converter (ADC)

Analog to Digital Converter merupakan rangkaian yang mengubah nilai tegangan kontinu menjadi nilai biner yang dapat dimengerti oleh perangkat digital sehingga dapat digunakan untuk komputasi digital.

Sinyal Analog pada kenyataannya berasal dari berbagai sumber dan sensor yang mengukur suara cahaya, gerakan serta suhu dan nilai akan terus berubah (kontinu) sehingga memberikan nilai yang berbeda dalam jumlah yang tak terbatas. ADC berfungsi mengubah dua domain yang berbeda dari sinyal analog yang (kontinu) menjadi sinyal digital yang diskrit.



Urutan proses kerja Analog to Digital Converter yaitu mengambil sampel sinyal analog, mengukur dan mengubahnya menjadi nilai digital yang berbentuk nilai biner. ADC dapat dikirimkan ke komputer melalui kabel atau wireless dalam bentuk wifi, sinyal 1V menggunakan ADC 3 bit apabila dikonversi menjadi biner yaitu 111 menghasilkan 8 tingkatan pembagian untuk output 1V. Untuk kecepatan sample speed atau sample rate ADC optimal memiliki rasio pengambilan hingga 300Ms/s



2. Digital to Analog Converter (DAC)

Digital to Analog Converter merupakan rangkaian yang mengubah sinyal digital yang berbentuk biner (0 dan 1) menjadi sinyal Analog yang kontinu (arus atau tegangan) sehingga dapat dimengerti oleh perangkat analog, atau kebalikan dari ADC Analog to Digital Converter. DAC mengubah Bit menjadi sinyal analog dalam bentuk tegangan maupun arus listrik.

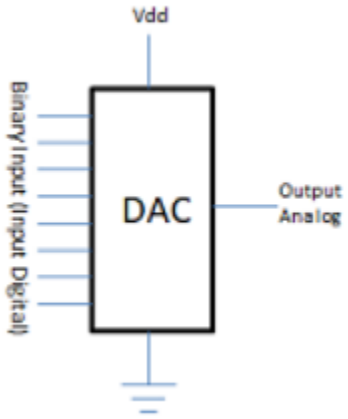


Diagram DAC

Sinyal diubah biasanya digunakan untuk menggerak diafragma speaker, motor, pengendali suhu.

Data biner digital adalah dalam bentuk bit. Bobotnya adalah 2n dimana n adalah posisi Bit dari sisi kanan yang dimulai dari 0. Pada ESP32 DAC terdapat 2 saluan terhubung ke GPIO 25 dan GPIO 26 .

3. Potensiometer

Potensiometer merupakan jenis resistor yang berfungsi untuk mengatur resistensi, tegangan, dan juga arus listrik yang mengalir dalam suatu rangkaian listrik.

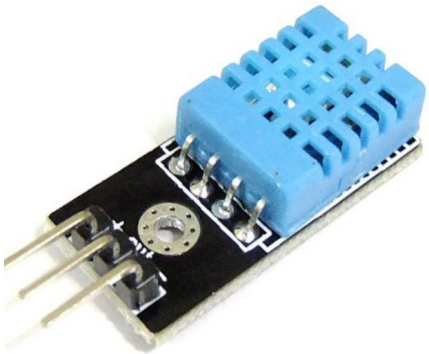
Resistor bekerja sebagai komponen pasif dalam rangkaian elektronika. Konstruksi potensiometer dikategorikan menjadi bagian geser (wiper) dan bagian non geser. Kedua ujung terminal paling pinggir dihubungkan dengan kedua ujung elemen resistif sehingga resistansinya seragam, ini adalah bagian non-geser dari potensiometer.

4. Sensor LM35

Sensor LM35 merupakan sensor temperatur yang berfungsi mengkonversi besaran temperatur menjadi besaran listrik dalam bentuk tegangan. IC pada sensor LM35 berfungsi mendeteksi temperatur lingkungan Sensor suhu LM35 memiliki keunggulan dibandingkan dengan sensor suhu yang dikalibrasi ke satuan Kelvin yaitu tidak perlu adanya pengurangan tegangan konstan yang besar dari output untuk mendapatkan skala keluaran Celcius.

Spesifikasi Sensor LM35 tegangan kerja berkisar 4 Volt DC - 30 Volt DC, output linear dengan kenaikan tegangan 10mV untuk tiap kenaikan temperatur sebesar 1**°C,** Akurasi kurang lebih 0.5**°C** pada temperatur ruangan 25**°C,**mempunyai tiga buah kaki yaitu +Vs, GDN, dan Vout.

5. DHT-11



DHT11 merupakan sensor yang digunakan untuk mengukur temperatur dan kelembaban yang berfungsi untuk mensensing objek yang memiliki output tegangan analog yang dapat diolah lebih lanjut menggunakan mikrokontroler, memiliki spesifikasi sebagai berikut:

- ➔ Tegangan masukan : 5 Vdc
- ➔ Rentang temperatur :0-50 ° C kesalahan ± 2 ° C
- ➔ Kelembaban :20-90% RH ± 5% RH error

DHT11 terdiri dari 4 pin tetapi yang dipakai hanya 3 pin saja, yaitu VCC(+) untuk tegangan input 5V, GND - yaitu Ground, Data berfungsi Data output serial

6. Kabel Jumper

Kabel Jumper merupakan kabel elektrik sebagai konduktor listrik yang memiliki pin konektor di setiap ujungnya dan memungkinkanmu untuk menghubungkan dua komponen.

Jenis -jenis kabel jumper:

- ➔ Kabel Jumper Male to Male
- ➔ Kabel Jumper Male to Female
- ➔ Kabel Jumper Female to Female

Kabel jumper memiliki beberapa kelebihan yaitu memiliki konektor di ujungnya yang sangat memudahkan kita dalam memasang maupun melepas kabel ke komponen, memiliki warna bervariasi yang memudahkan kita dalam membuat rangkaian

BAB 2

Metodologi

A. Tugas

Coba buat program Esp32 ADC untuk membaca potensiometer, sensor LM35 dan DHT-11/DHT-22 (gunakan adc smoothing reading). Lanjutkan dengan mengirimkan data bacaan Esp32 ini ke laptop/pc dengan membuat aplikasi python (tampilkan dalam bentuk teks dan chart/plot) Selain Analog to Digital Converter (ADC), Esp32 juga memiliki pin yang berfungsi sebagai Digital to Analog Converter (DAC), dimana di DAC adalah kebalikan dari ADC.Buatlah laporan untuk semua item langkah-langkah yang disebutkan di atas (Yang dikumpulkan adalah laporan, file code esp32 dan juga python jika ada unsur aplikasi pythonnya)

B. Permulaan

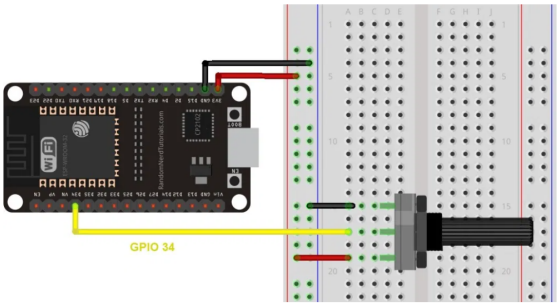
Pada project ini ada beberapa komponen yang dibutuhkan diantaranya Esp32 ADC, potensiometer, sensor LM35 dan DHT-11/DHT-22 (menggunakan ADC smoothing), dan kabel jumper.

C. Code

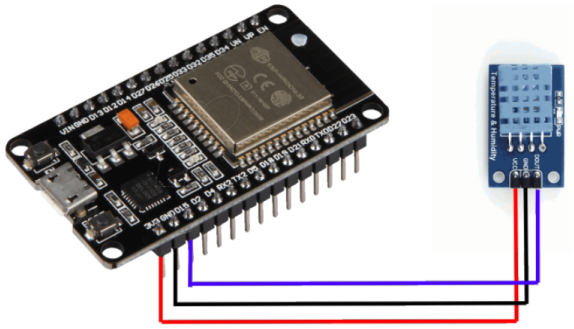
1. Persiapan Skematik Rangkaian

Pertama, buat skema rangkaian yang akan dibuat agar memudahkan dalam merangkai Esp32 dan lain-lain.

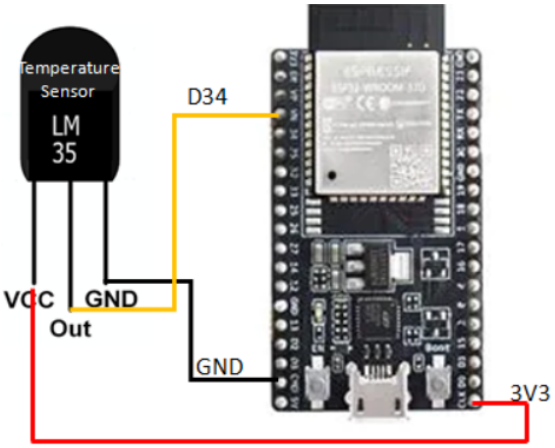
- Esp32 to Potentiometer



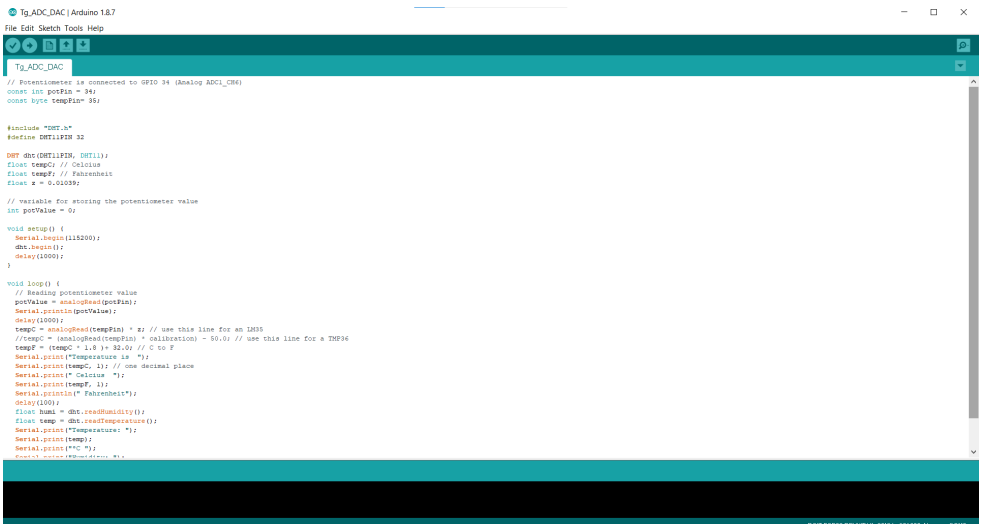
- Esp32 to DHT11



- Esp32 to LM35



2. Code Arduino



```
float floatMap(float x, float in_min, float in_max, float out_min, float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
#define ADC_VREF_mV    3300.0 // in millivolt
#define ADC_RESOLUTION 4096.0
#define PIN_LM35       25 // ESP32 pin GIOP36 (ADC0) connected to LM35
#include "DHT.h"
#define DHTPIN 27
#define DHTTYPE DHT11
```

```
DHT dht(DHTPIN, DHTTYPE);
float tempC; // Celcius
float tempF; // Fahrenheit
float z = 0.01039;
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  Serial.println(F("DHT22test"));
  dht.begin();
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin GIOP36:
  int analogValue = analogRead(36);
  // Rescale to potentiometer's voltage (from 0V to 3.3V):
  float voltage = floatMap(analogValue, 0, 4095, 0, 3.3);

  // print out the value you read:
  Serial.print("Analog: ");
  Serial.print(analogValue);
  Serial.print(", Voltage: ");
  Serial.println(voltage);
  delay(1000);

   // read the ADC value from the temperature sensor
  int adcVal = analogRead(PIN_LM35);
```

```
// convert the ADC value to voltage in millivolt
float milliVolt = adcVal * (ADC_VREF_mV / ADC_RESOLUTION);
// convert the voltage to the temperature in °C
float tempC = milliVolt / 10;
// convert the °C to °F
float tempF = tempC * 9 / 5 + 32;

Serial.print("DatafromLM35\n");

// print the temperature in the Serial Monitor:
Serial.print("Temperature: ");
Serial.print(tempC);   // print the temperature in °C
Serial.print("°C");
Serial.print("  ~  "); // separator between °C and °F
Serial.print(tempF);   // print the temperature in °F
Serial.println("°F");

delay(500);

// Wait a few seconds between measurements.
delay(2000);

Serial.print("DatafromDHT22\n");
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
float h = dht.readHumidity();
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
```

```
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

   // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
     Serial.println(F("Failed to read from DHT sensor!"));
     return;
   }

// Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%Temperature:"));
  Serial.print(t);
  Serial.print(F("°C"));
  Serial.print(f);
  Serial.print(F("°FHeat index:"));
  Serial.print(hic);
  Serial.print(F("°C"));
  Serial.print(hif);
  Serial.println(F("°F"));
}
DAN
float floatMap(float x, float in_min, float in_max, float out_min, float out_max) {
```

```
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
#define ADC_VREF_mV    3300.0 // in millivolt
#define ADC_RESOLUTION 4096.0
#define PIN_LM35       25 // ESP32 pin GIOP36 (ADC0) connected to LM35
#include "DHT.h"
#define DHTPIN 27
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
float tempC; // Celcius
float tempF; // Fahrenheit
float z = 0.01039;
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  //Serial.println(F("DHT22 test!"));
  dht.begin();
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin GIOP36:
  int analogValue = analogRead(36);
  // Rescale to potentiometer's voltage (from 0V to 3.3V):
  float voltage = floatMap(analogValue, 0, 4095, 0, 3.3);

  // print out the value you read:
```

```
//Serial.print("Analog: ");
//Serial.println(analogValue);
//Serial.print(", Voltage: ");
//Serial.print(voltage);
//delay(1000);

 // read the ADC value from the temperature sensor
int adcVal = analogRead(PIN_LM35);
// convert the ADC value to voltage in millivolt
float milliVolt = adcVal * (ADC_VREF_mV / ADC_RESOLUTION);
// convert the voltage to the temperature in °C
float tempC = milliVolt / 10;
// convert the °C to °F
float tempF = tempC * 9 / 5 + 32;

//Serial.print("Data from LM35 \n");

// print the temperature in the Serial Monitor:
//Serial.print("Temperature: ");
Serial.print(" , ");
//Serial.println(tempC);   // print the temperature in °C
Serial.print("°C");
//Serial.print("  ~  "); // separator between °C and °F
//Serial.print(tempF);   // print the temperature in °F
//Serial.println("°F");

delay(500);
```

```
// Wait a few seconds between measurements.
//delay(2000);

//Serial.print("Data from DHT22\n");
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
//float h = dht.readHumidity();
// Read temperature as Celsius (the default)
//float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
//float f = dht.readTemperature(true);

 // Check if any reads failed and exit early (to try again).
//if (isnan(h) || isnan(t) || isnan(f)) {
  //  Serial.println(F("Failed to read from DHT sensor!"));
   // return;
 //d }

// Compute heat index in Fahrenheit (the default)
//float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit = false)
//float hic = dht.computeHeatIndex(t, h, false);

//Serial.print(F("Humidity: "));
//Serial.print(h);
//Serial.print(F("%Temperature:"));
//Serial.print(t);
//Serial.print(F("°C"));
```

```
  //Serial.print(f);
  //Serial.print(F("°FHeat index:"));
  //Serial.print(hic);
  //Serial.print(F("°C"));
  //Serial.print(hif);
  //Serial.println(F("°F"));
}
```

Memakai 2 arduino dikarenakan untuk program python dikarenakan fungsi printin tidak dapat berupa string

3. Code Python

```
import serial
import time
import numpy
import matplotlib.pyplot as plt
from drawnow import *


tempF = []
pressure = []


arduinoData = serial.Serial('COM8',9600)
plt.ion()       #Tell matplotlib you want interactive mode to plot live data
cnt = 0


def makeFig():              # Create a function that makes our desired plot
```

```
        #plt.ylim(490,530)          # Menambahkan limit sumbu y
        plt.title('My Live Streaming Sensor Data')
        plt.grid(True)           # Tambahkan grid
        plt.ylabel('Temp F')     # Tambahkan label
        plt.plot(tempF, 'ro-', label='potensiometer')
        plt.legend(loc='upper left')
        plt2 = plt.twinx()
        plt2.plot(pressure, 'b^-')



    while (1):
        arduinoString = arduinoData.readline()
        print(arduinoString)
        arduinoString = arduinoString.decode('UTF-8')
        dataArray = arduinoString.split(',')
        temp = float (dataArray[0])
        P = float (dataArray[1])
        print (temp)#,",",P   # cek data sudah bisa di-split atau
belum
        tempF.append(temp)
        pressure.append(P)
        print (P)         # cek append sudah bisa atau belum
        drawnow(makeFig)     # Call drawnow to update our live
graph
        plt.pause(.000001)
        cnt = cnt + 1
        if(cnt > 50):        # setting sumbu x = 50, agar data
tidak menumpuk
```

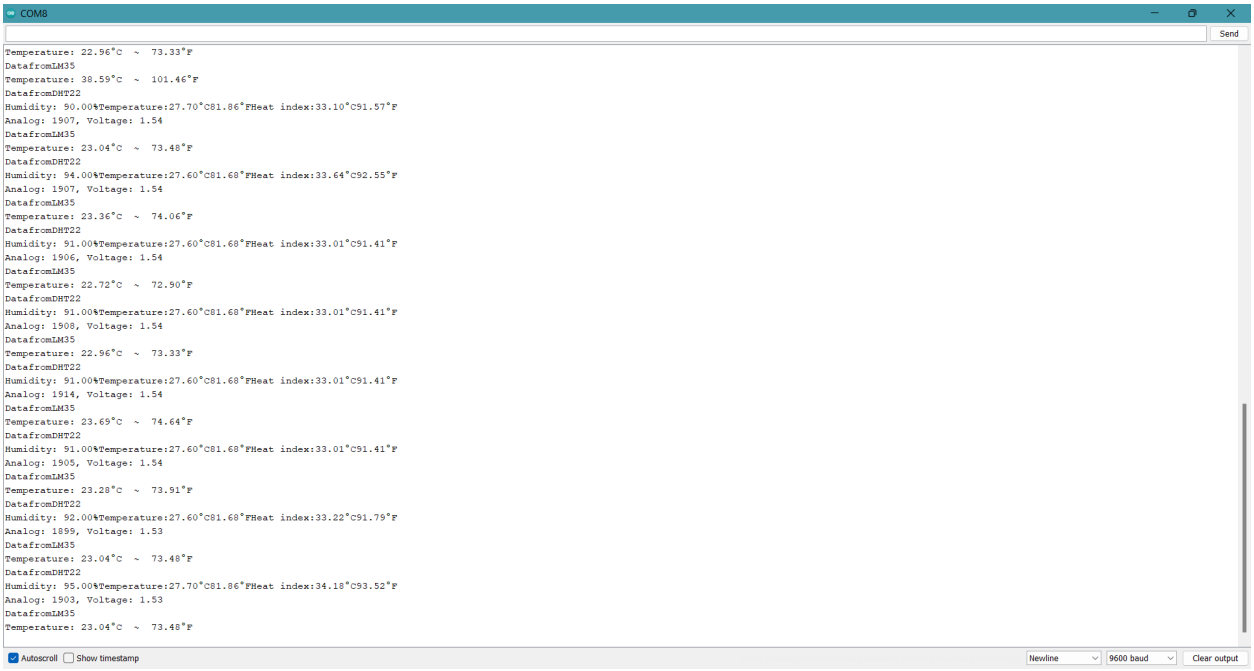
```
            tempF.pop(0)

            pressure.pop(0
```

BAB 3

HASIL PRAKTIKUM

1. Hasil Arduino



2. Hasil Python