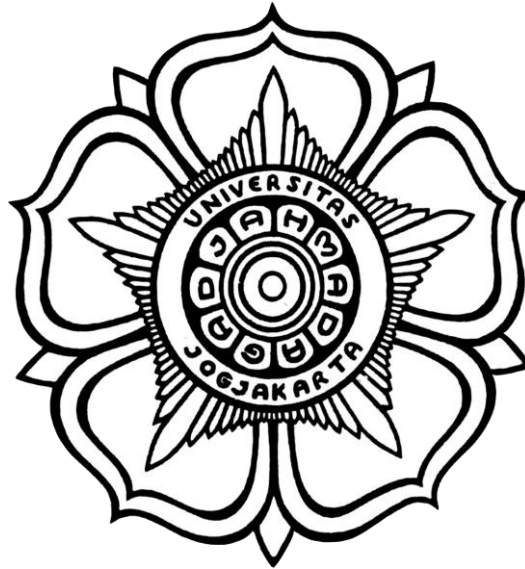


**LAPORAN PRAKTIKUM**  
**ELEKTRONIKA MESIN LISTRIK DAN TEKNIK KENDALI**

*“Analog to Digital Converter and Digital to Analog Converter”*

Dosen Pengampu: Irfan Bahiuddin, ST, M.Phil., Ph.D.



Disusun Oleh:

Kelompok 4

Chaesar Syaefuddin (19/441195/SV/16547)

Yeyen Karunia (19/441215/SV/16567)

Kelas: ARM 2

**DEPARTEMEN TEKNIK MESIN**

**SEKOLAH VOKASI**

**UNIVERSITAS GADJAH MADA**

**YOGYAKARTA**

**2022**

## BAB I. DESKRIPSI KASUS

Dalam kasus ini akan membuat program ESP32 dengan ADC (Analog to Digital Converter) agar dapat membaca nilai Potensiometer, Sensor LM35, dan Sensor DHT11. Project ini menggunakan mikrokontroller ESP32 yang dihubungkan dengan perangkat-perangkat elektronika seperti potensiometer, sensor LM35, dan sesor DHT11. Nilai-nilai yang telah terbaca akan diteruskan pada Phyton dengan mekanisme *Smoothing Reading* agar data yang dihasilkan menjadi lebih terstruktur dan baik. Selain itu, di laporan kali ini akan membahas terkait literatur DAC (Dialog to Analog Converter) yang terhubung dengan mikrokontroller ESP32.

## BAB II. KOMPONEN YANG DIGUNAKAN

### 2.1 Mikrokontroller ESP32

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*. Terlihat pada gambar dibawah merupakan pin out dari ESP32. Pin tersebut dapat dijadikan input atau output untuk menyalakan LCD, lampu, bahkan untuk menggerakkan motor DC. Pada mikrokontroler ini sudah tersedia modul wifi dan bluetooth sehingga sangat mendukung untuk membuat sistem aplikasi Internet of Things. Memiliki 18 ADC (*Analog Digital Converter*), 2 DAC, 16 PWM, 10 Sensor sentuh, 2 jalur antarmuka UART, pin antarmuka I2C, I2S, dan SPI.



Gambar 1. Mikrokontroller Esp32

ESP32 menggunakan prosesor dual core yang berjalan di instruksi Xtensa LX16 [3], ESP32 memiliki spesifikasi seperti yang ditampilkan pada tabel 1.

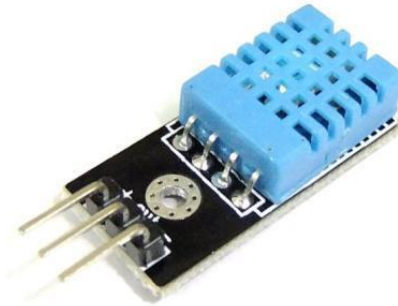
**Tabel 1. Spesifikasi Mikrokontroler Esp32**

Atribut	Detail
CPU	Tensilica Xtensa LX6 32bit Dual-Core di 160/240MHz
SRAM	520KB
FLASH	2MB (max. 64MB)
Tegangan	2.2V sampai 3.6V
Arus Kerja	Rata-rata 80mA
Dapat diprogram	Ya (C, C++, Python, Lua, dll)
Open Source	Ya
<b>Konektivitas</b>	
Wi-Fi	802.11 b/g/n
Bluetooth	4.2BR/EDR + BLE
UART	3
<b>I/O</b>	
GPIO	32
SPI	4
I2C	2
PWM	8
ADC	18 (12-bit)
DAC	2 (8-bit)

## **2.2 Sensor DHT11**

Sensor DHT11 merupakan sensor dengan kalibrasi sinyal digital yang mampu memberikan informasi suhu dan kelembaban. Sensor ini tergolong komponen yang memiliki tingkat stabilitas yang sangat baik. Produk dengan kualitas terbaik, respon pembacaan yang cepat, dan kemampaan anti-interference, dengan harga yang terjangkau. DHT11 memiliki fitur kalibrasi yang sangat akurat. Koefisien kalibrasi ini disimpan dalam OTP program memory, sehingga ketika internal sensor mendeteksi sesuatu suhu atau kelembaban, maka module ini membaca koefisien sensor tersebut. Ukurannya yang kecil, dengan transmisi sinyal

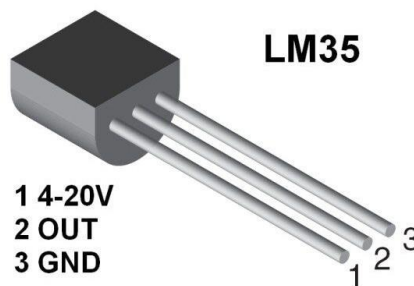
hingga 20 meter, membuat produk ini cocok digunakan untuk banyak aplikasi-aplikasi pengukuran suhu dan kelembaban.



Gambar 2. Sensor DHT11

### 2.3 Sensor LM35

Sensor suhu LM35 adalah komponen elektronika yang memiliki fungsi untuk mengubah besaran suhu menjadi besaran listrik dalam bentuk tegangan. LM35 memiliki keakuratan tinggi dan kemudahan perancangan jika dibandingkan dengan sensor suhu yang lain dan juga mempunyai keluaran impedansi yang rendah dan linieritas yang tinggi sehingga dapat dengan mudah dihubungkan dengan rangkaian kendali khusus serta tidak memerlukan penyetelan lanjutan.

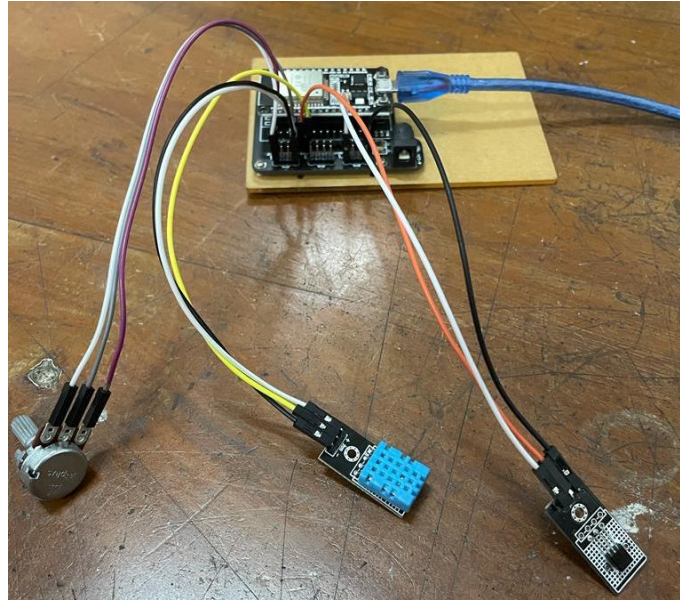


Gambar 3. Sensor LM35

## BAB III. RANGKAIAN PROGRAM

### 3.1 Rangkaian Elektronika

Pada kasus ini, rangkaian mikrokontroler Esp32, sensor DHT11, dan sensor LM35 akan disusun dengan skema rangkaian seperti dibawah ini.



Gambar 4. Rangkaian Elektronika

### 3.2 Kode Arduino

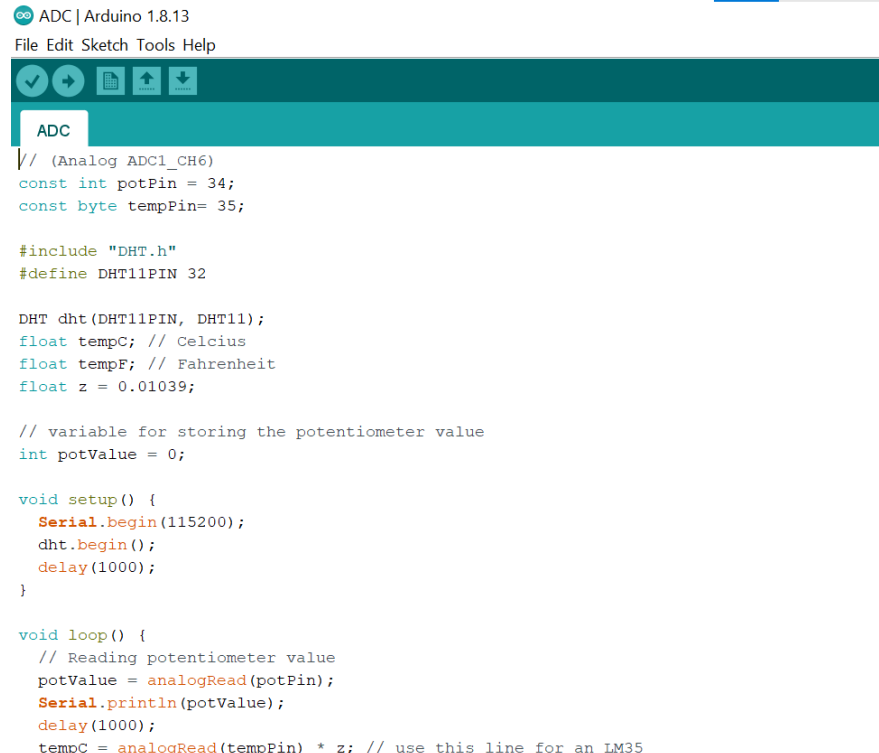
```
// (Analog ADC1_CH6)
const int potPin = 34;
const byte tempPin= 35;

#include "DHT.h"
#define DHT11PIN 32

DHT dht(DHT11PIN, DHT11);
float tempC; // Celcius
float tempF; // Fahrenheit
float z = 0.01039;

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

```
void setup() {  
  Serial.begin(115200);  
  dht.begin();  
  delay(1000);  
}  
  
void loop() {  
  // Reading potentiometer value  
  potValue = analogRead(potPin);  
  Serial.println(potValue);  
  delay(1000);  
  tempC = analogRead(tempPin) * z; // use this line for an LM35  
  //tempC = (analogRead(tempPin) * calibration) - 50.0; // use this line for a  
  TMP36  
  tempF = (tempC * 1.8 )+ 32.0; // C to F  
  
  Serial.print("Temperature is ");  
  Serial.print(tempC, 1); // one decimal place  
  Serial.print(" Celcius ");  
  Serial.print(tempF, 1);  
  Serial.println(" Fahrenheit");  
  delay(100);  
  float humi = dht.readHumidity();  
  float temp = dht.readTemperature();  
  Serial.print("Temperature: ");  
  Serial.print(temp);  
  Serial.print("°C ");  
  Serial.print("Humidity: ");  
  Serial.println(humi);  
  delay(200);  
}
```

The image shows the Arduino IDE interface. At the top, there's a title bar with 'ADC | Arduino 1.8.13'. Below it is a menu bar with 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. A toolbar with icons for opening, saving, and running is visible. The main area shows a sketch named 'ADC' with the following code:

```
// (Analog ADC1_CH6)
const int potPin = 34;
const byte tempPin= 35;

#include "DHT.h"
#define DHT11PIN 32

DHT dht(DHT11PIN, DHT11);
float tempC; // Celcius
float tempF; // Fahrenheit
float z = 0.01039;

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

void setup() {
  Serial.begin(115200);
  dht.begin();
  delay(1000);
}

void loop() {
  // Reading potentiometer value
  potValue = analogRead(potPin);
  Serial.println(potValue);
  delay(1000);
  tempC = analogRead(tempPin) * z; // use this line for an LM35
```

Gambar 5. Arduino Code

### 3.3 Kode Phyton

```
from ast import While
import string
from numpy import append
import serial
import time
import matplotlib.pyplot as plt
from drawnow import *

Hum=[]
Temp=[]
Pot=[]
plt.ion()
cnt=0
def makeFig():
    plt.ylim(40,99)
    plt.title('Read Live Data Sensor & Potensio')
```

```

plt.grid(True)
plt.ylabel('Hum %')
plt.plot(Hum, 'ro-', label='Persen %')
plt.legend(loc='upper left')
plt2=plt.twinx()
#plt.ylim(93500,93525)
plt2.plot(Temp, 'b^-', label='Temperature °C')
plt2.set_ylabel('Temperature C')
plt2.ticklabel_format(useOffset=False)
plt2.legend(loc='upper right')
plt3=plt.twinx()
plt3.plot(Pot, 'g-', label='Potensio ADC')
plt3.set_ylabel('Potensio ADC')
plt3.ticklabel_format(useOffset=False)
plt3.legend(loc='upper center')

# make sure the 'COM#' is set according the Windows Device Manager
ser = serial.Serial('COM3', 115200, timeout=1)
time.sleep(2)

while True:
    b = ser.readline()      # read a byte string
    if len(b) != 0:
        string1 = b.decode()
        string = string1.rstrip() # remove \n and \r

        index=string.split(',')
        H=float(index[0])
        T=float(index[1])
        P=float(index[2])

        Hum.append(H)

```



```

Temp.append(T)
Pot.append(P)

drawnow(makeFig)

plt.pause(0.000001)

cnt=cnt+1

if(cnt>50):
    Hum.pop(0)
    Temp.pop(0)
    Pot.pop(0)

```

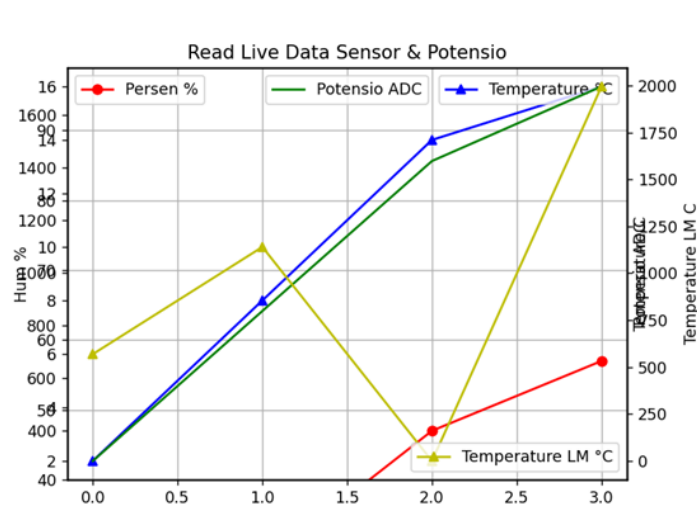
```

1  from ast import While
2  import string
3  from numpy import append
4  import serial
5  import time
6  import matplotlib.pyplot as plt
7  from drawnow import *
8  Hum=[]
9  Temp=[]
10 Pot=[]
11 plt.ion()
12 cnt=0
13 def makeFig():
14     plt.ylim(40,99)
15     plt.title('Read Live Data Sensor & Potensio')
16
17     plt.grid(True)
18     plt.ylabel('Hum %')
19     plt.plot(Hum, 'ro-', label='Persen %')
20     plt.legend(loc='upper left')
21     plt2=plt.twinx()
22     #plt.ylim(93500,93525)
23     plt2.plot(Temp, 'b-', label='Temperature °C')
24     plt2.set_ylabel('Temperature C')
25     plt2.ticklabel_format(useOffset=False)
26     plt2.legend(loc='upper right')
27     plt3=plt.twinx()
28     plt3.plot(Pot, 'g-', label='Potensio ADC')
29     plt3.set_ylabel('Potensio ADC')
30     plt3.ticklabel_format(useOffset=False)
31     plt3.legend(loc='upper center')

```

Gambar 6. Phyton Code

### 3.4 Hasil Simulasi



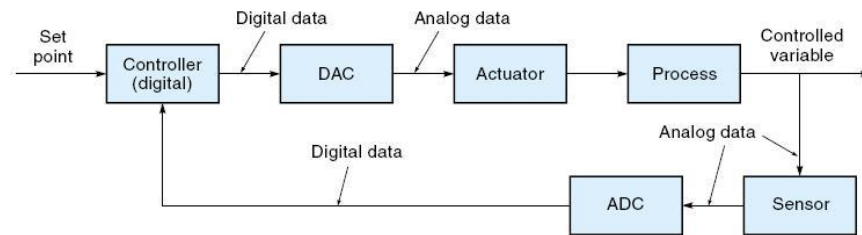
## BAB IV. DIGITAL TO ANALOG CONVERTER

DAC adalah perangkat yang digunakan untuk mengkonversi sinyal masukan dalam bentuk digital menjadi sinyal keluaran dalam bentuk analog (tegangan). Tegangan keluaran yang dihasilkan DAC sebanding dengan nilai digital yang masuk ke dalam DAC. Sebuah DAC menerima informasi digital dan mentransformasikannya ke dalam bentuk suatu tegangan analog. Informasi digital adalah dalam bentuk angka biner dengan jumlah digit. Konverter D/A dapat mengonversi sebuah word digital ke dalam sebuah tegangan analog dengan memberikan skala output analog berharga nol ketika semua bit adalah nol dan sejumlah nilai maksimum ketika semua bit adalah satu. Angka biner sebagai angka pecahan. Aplikasi DAC banyak digunakan sebagai rangkaian pengendali (driver) yang membutuhkan input analog seperti motor AC maupun DC, tingkat kecerahan pada lampu, Pemanas (Heater) dan sebagainya. Umumnya DAC digunakan untuk mengendalikan peralatan computer.

Untuk aplikasi modern hampir semua DAC berupa rangkaian terintegrasi (IC), yang diperlihatkan sebagai kotak hitam memiliki karakteristik input dan output tertentu. Karakteristik yang berkaitan adalah:

1. **Input Digital:** Jumlah bit dalam sebuah word biner paralel disebutkan di dalam lembar spesifikasi.
2. **Catu Daya:** Merupakan bipolar 2 V hingga  $\pm 18$  V seperti yang dibutuhkan oleh amplifier internal.
3. **Suplai Referensi:** Diperlukan untuk menentukan jangkauan tegangan output dan resolusi dari konverter. Suplai ini harus stabil, memiliki ripple yang kecil.
4. **Output:** Sebuah tegangan yang merepresentasikan input digital. Tegangan ini berubah dengan step sama dengan perubahan bit input digital. Output aktual dapat berupa bipolar jika konverter didesain untuk menginterpretasikan input digital negatif.
5. **Offset:** Karena DAC biasanya di implementasikan dengan op-amp, maka mungkin adanya tegangan output offset dengan sebuah input nol. Secara khusus, koneksi akan diberikan untuk mendukung pengesetan ke harga nol dari output DAC dengan input word nol

6. **Mulai konversi:** Sejumlah rangkaian DAC memberikan sebuah logika input yang mempertahankan konversi dari saat terjadinya hingga diterimanya sebuah perintah logika tertentu (1 atau 0). Dalam ini, word input digital diabaikan hingga diterimanya input logika tertentu. Dalam sejumlah hal, sebuah buffer input diberikan untuk memegang (hold) word digital selama dilakukannya konversi hingga selesai



## BAB V. KESIMPULAN

Pada kasus ADC ini akan membaca ADC 0, 1, dan 2 untuk mendapatkan nilai dari sensor LM35, sensor DHT11, dan Potensiometer. Pembacaan ADC dilakukan dengan proses smoothing, yaitu proses yang membuat pembacaan ADC semakin halus. Setelah mendapatkan nilai ADC, nilai ADC diubah ke dalam bentuk float dan disimpan ke variable yang bertipe data float. Dengan begitu, data yang dihasilkan menjadi lebih baik apabila menggunakan pembacaan halus atau *smoothing reading*.