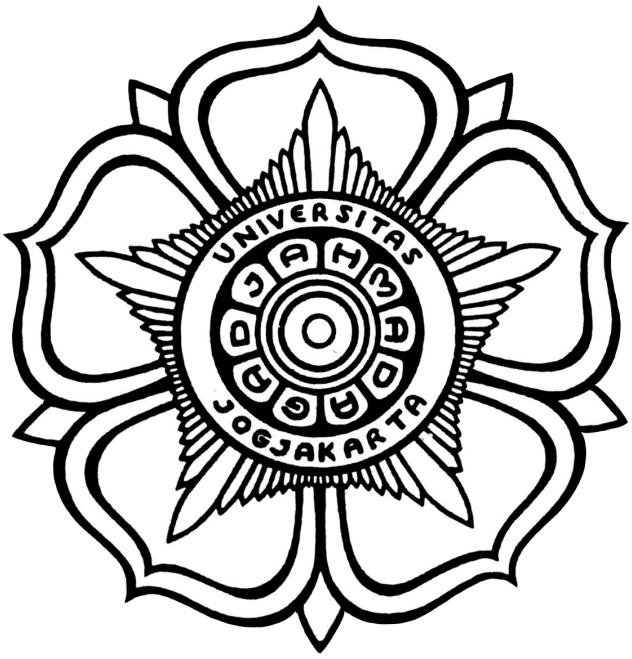
LAPORAN PRAKTIKUM ELEKTRONIKA MESIN LISTRIK DAN TEKNIK KENDALI

"*Internet of Things (IoT) Cloud Provider/Service & ADC-DAC*"

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



Disusun Oleh:
Kelompok 1
Catur Wardana (19/441194/SV/16546)
Santi Rahayu (19/441209/SV/16561)

Kelas: ARM 2

TEKNOLOGI REKAYASA MESIN

DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2022

Daftar Isi

Laporan 1. *Protocol IoT HTTP*

BAB 1. Pengenalan Kasus

Kasus pada pada tugas 1 ini adalah Esp32 Publishing Potentiometer Reading to SQL Database using Apps Python Flask. Pengiriman menggunakan Fungsi dari Protokol IoT HTTP POST dari esp32 ke python flask dan data yang diterima harus dimasukkan kedalam database menggunakan CRUD.

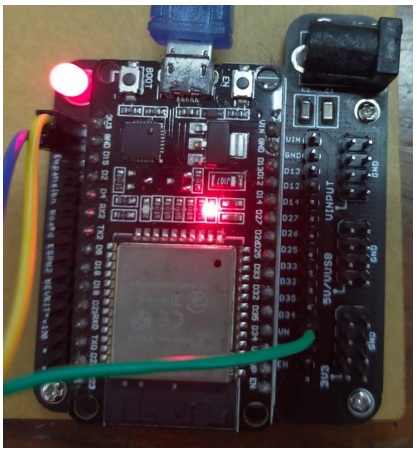
BAB 2. Komponen yang Digunakan

2.1 Potensiometer

Gambar 2.1 Potensiometer

Potensiometer (POT) adalah salah satu jenis resistor yang nilai resistansinya dapat diatur sesuai dengan kebutuhan dari rangkaian elektronika ataupun kebutuhan pemakainya. Potensiometer merupakan Keluarga Resistor yang tergolong dalam Kategori Variable Resistor. Secara struktur, potensiometer terdiri dari 3 kaki Terminal dengan sebuah shaft atau tuas yang berfungsi sebagai pengaturnya (Fransisko, P., 2019).

2.2 ESP32



Gambar 2.2 ESP32

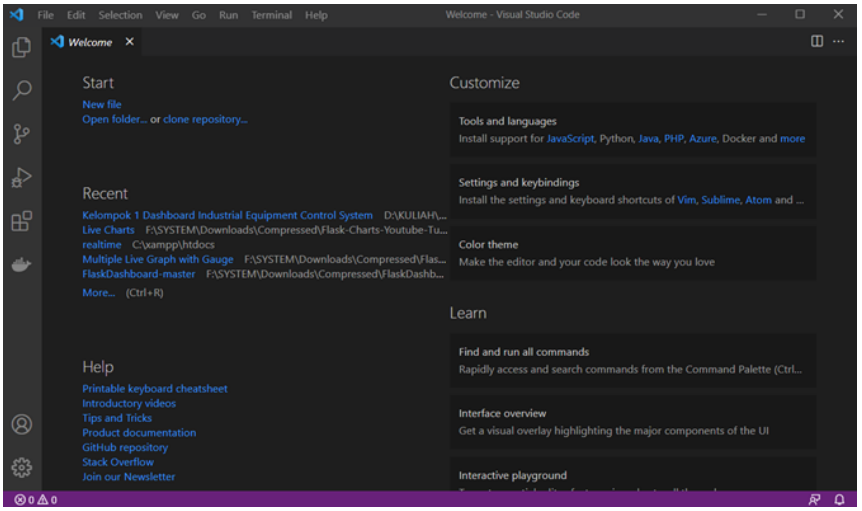
ESP32 adalah nama dari mikrokontroler yang dirancang oleh perusahaan yang berbasis di Shanghai, China yakni Espressif Systems. ESP32 menawarkan solusi jaringan WiFi yang mandiri sebagai jembatan dari mikrokontroler yang ada ke jaringan WiFi. ESP32 menggunakan prosesor dual core yang berjalan di instruksi Xtensa LX16 [3], ESP32 memiliki spesifikasi seperti yang ditampilkan pada tabel 1.

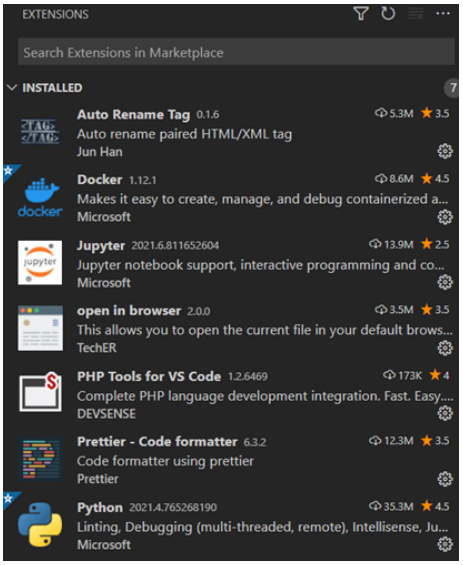
No	Atribut	Detail
1.	Tegangan	3.3 Volt
2.	Prosesor	Tensilica L108 32 bit
3.	Kecepatan Prosesor	Dual 160 Mhz
4.	RAM	520K
5.	GPIO	34
6.	ADC	7
7.	Dukungan 802.11	11b/g/n/e/i
8.	Bluetooth	BLE (Bluetooth Low Energy)
9.	SPI	3
10.	12C	2
11.	UART	3

Jika dilihat dari spesifikasi pada tabel maka mikrokontroler ESP32 dapat dijadikan pilihan untuk digunakan pada alat peraga interface mikrokontroler karena mikrokontroler ini memiliki interface yang lengkap, juga memiliki WiFi yang sudah tertanam pada mikrokontroler sehingga tepat untuk digunakan pada alat peraga atau trainer Internet of Things. Pada gambar 1 merupakan pin out dari GPIO pada ESP32 (Kusumah, H. dan Pradana, R.A., 2019).

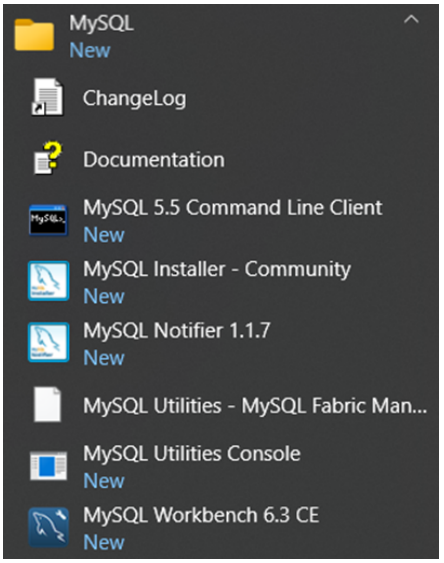
2.3 Persiapan Aplikasi dan Extensi yang akan digunakan

1. Langkah pertama dalam pembuatan project ini adalah membuka aplikasi – aplikasi penunjang pembuatan project seperti Visual Studio Code.

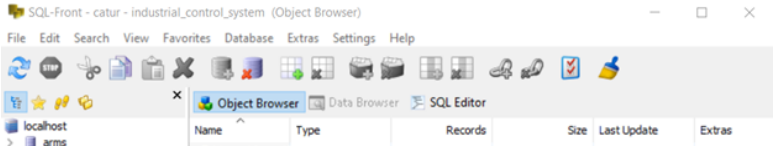




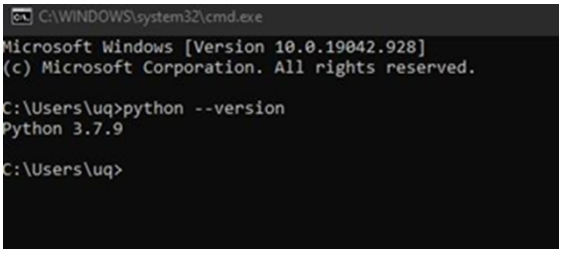
2. MySQL Installer



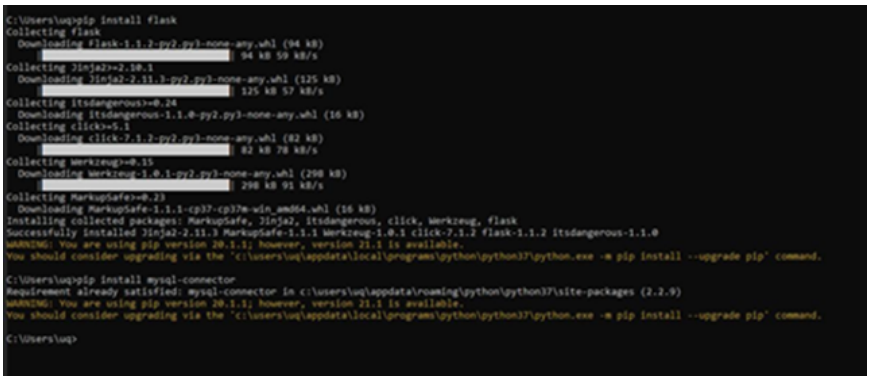
3. Mysql Front



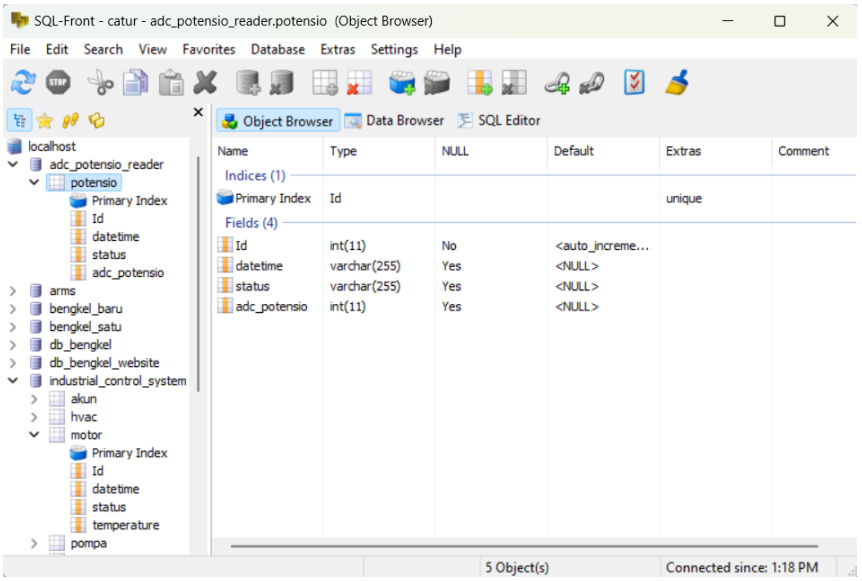
4. Python



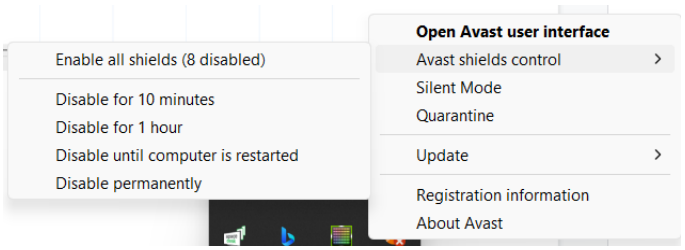
5. Extensi Python (Flask, flask_mysqldb, datetime, time)



- Selanjutnya adalah membuat database di localhost agar nantinya sistem dapat terkoneksi dengan database secara otomatis.

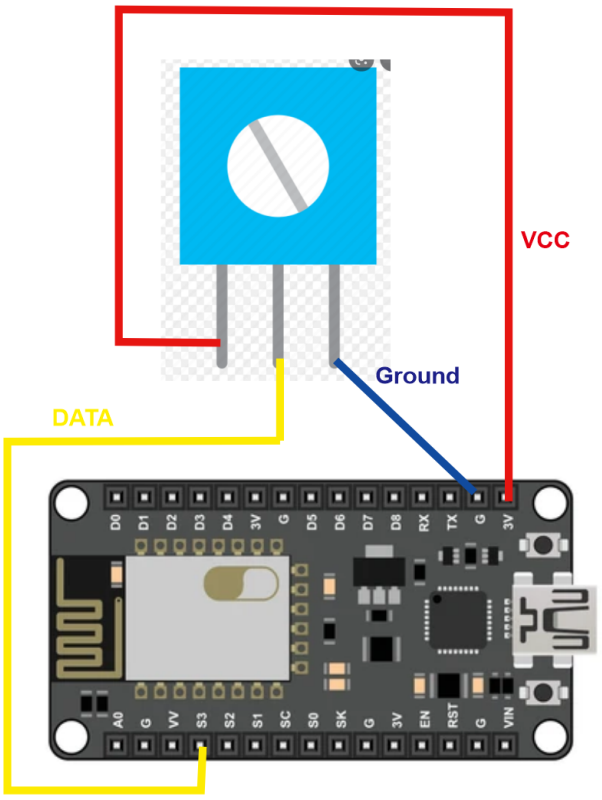


- Firewall atau antivirus harus dimatikan



BAB 3. Pengerjaan

3.1 Rangkaian ESP dan Potensio

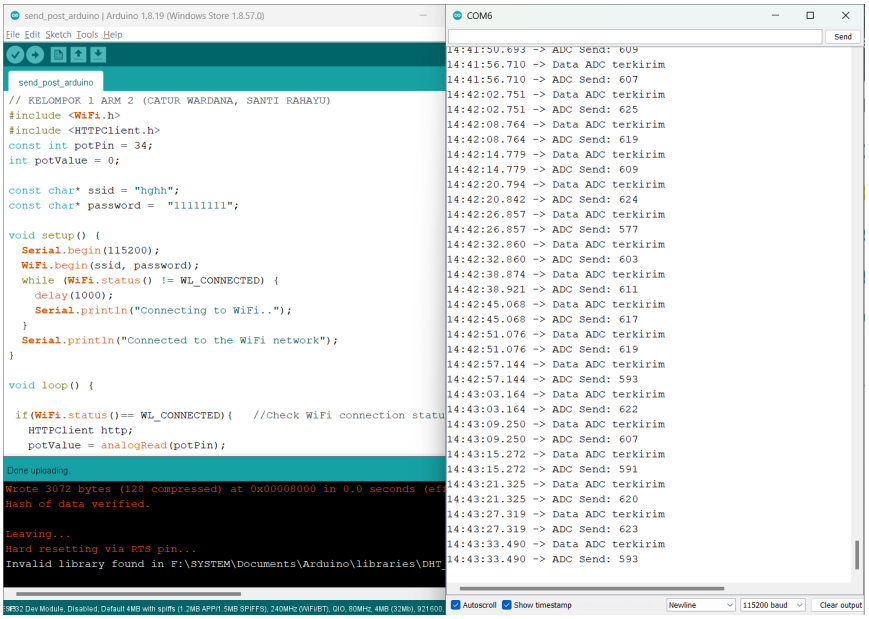


- Data dari Potensio dibaca pada pin 34
- Kedua kaki di ujung kanan dan kiri potensio adalah pin VCC dan GND

3.2 ESP32 CODE

```
// KELOMPOK 1 ARM 2 (CATUR WARDANA, SANTI RAHAYU)
#include <WiFi.h>
#include <HTTPClient.h>
const int potPin = 34;
int potValue = 0;
```

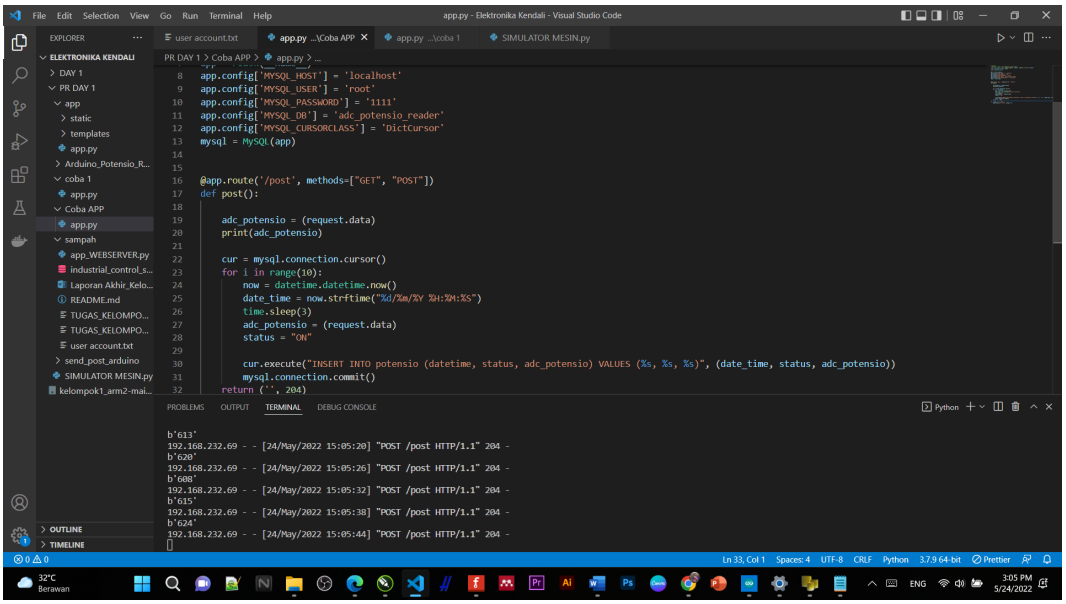
```
const char* ssid = "hghh";
const char* password =  "11111111";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println("Connected to the WiFi network");
}

void loop() {

 if(WiFi.status()== WL_CONNECTED){   //Check WiFi connection status
   HTTPClient http;
   potValue = analogRead(potPin);
   int adc;
   adc = potValue;

   http.begin("http://192.168.232.50:5000/post");
   http.addHeader("Content-Type", "text/plain");

   int httpResponseCode = http.POST(String(adc));

   if(httpResponseCode >=0 || httpResponseCode <0);
  {
    Serial.print("Data ADC terkirim");
  }
   Serial.print('\n');
   Serial.print("ADC Send: ");
   Serial.println(adc);
   http.end();
 }
 else{
    Serial.println("Error in WiFi connection");
 }
  delay(1000);
}
```



3.2 PYTHON FLASK CODE

```
# app.py
# KELOMPOK 1 ARM 2 (CATUR WARDANA, SANTI RAHAYU)
from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysqldb import MySQL
import datetime, time

app = Flask(__name__)
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = '1111'
app.config['MYSQL_DB'] = 'adc_potensio_reader'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'
mysql = MySQL(app)

@app.route('/post', methods=["GET", "POST"])
def post():

    adc_potensio = (request.data)
    print(adc_potensio)

    cur = mysql.connection.cursor()
    for i in range(10):
        now = datetime.datetime.now()
        date_time = now.strftime("%d/%m/%Y %H:%M:%S")
```

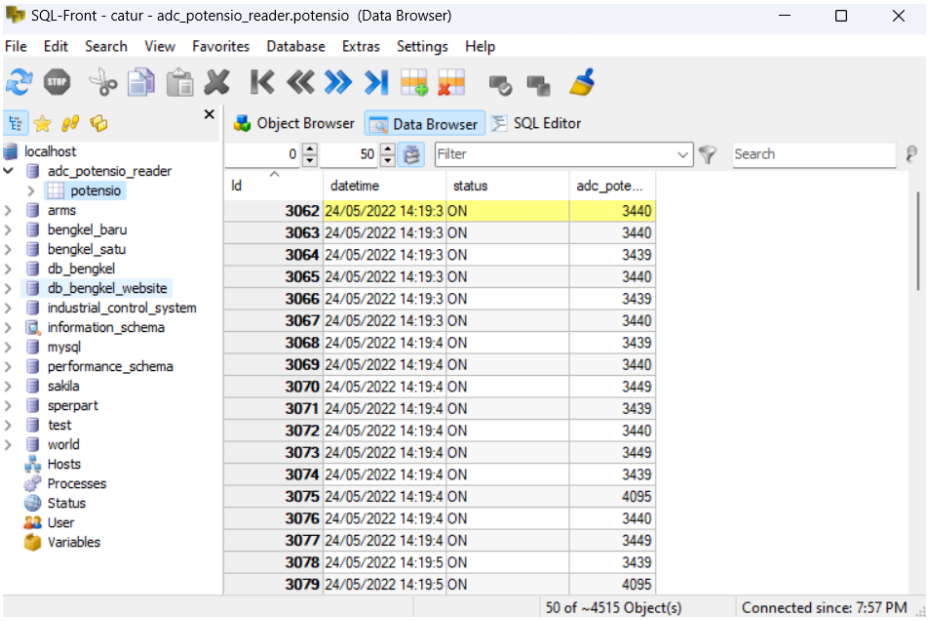
```
        time.sleep(3)
        adc_potensio = (request.data)
        status = "ON"

            cur.execute("INSERT INTO potensio (datetime, status, adc_potensio)
VALUES (%s, %s, %s)", (date_time, status, adc_potensio))
        mysql.connection.commit()
    return ('', 204)

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```



BAB 4. Kesimpulan

Kesimpulan proses kerja program:

1. Fungsi Code yang ada di ESP adalah mengambil data ADC dari sebuah tegangan yang mengalir ke pin 34.
2. Pembacaan ADC dikirimkan menggunakan Protokol HTTP Post ke Python Flask dengan metode local network dengan mengkoneksikan komputer dan esp ke satu router.
3. Python Flask menerima data Protokol HTTP Post dari ESP32 menggunakan fungsi Python Flash GET.
4. Python Flask melakukan fungsi CRUD data yang diperoleh dari esp ke MYSQL database menggunakan Fungsi Python POST.
5. Nilai ADC dari ESP berhasil dikirimkan ke Python Flask dan berhasil dimasukkan kedalam Database Local.



Daftar Pustaka

Fransisko, P., 2019. Media Promosi Elektronik untuk Produk-Produk di Supermarket Menggunakan Arduino Nano. Jurnal Sistem Cerdas dan Rekayasa (JSCR), 1(1).

Kusumah, H. and Pradana, R.A., 2019. Penerapan Trainer Interfacing Mikrokontroler Dan Internet of Things Berbasis Esp32 Pada Mata Kuliah Interfacing. Journal Cerita, 5(2), pp.120-134.

Laporan 2. *Protocol IoT MQTT*

I. Deskripsi Kasus

Dalam proyek ini kami akan membuat server web mandiri dengan micro framework Python Flask yang dapat mengoperasikan dua LED dari ESP32 menggunakan protokol MQTT.

II. Komponen dan Ekstensi aplikasi yang digunakan

1. Komponen yang digunakan

- ESP32 board dengan chip ESP-WROOM-32
- Resistor
- Kabel jumper
- LED 2 buah

2. Persiapan Ekstensi aplikasi yang digunakan

A. Install dan jalankan Mosquitto Broker

MQTT adalah singkatan dari Message Queuing Telemetry Transport. Mosquitto MQTT adalah protokol pesan sederhana, yang dirancang untuk perangkat terbatas

dengan bandwidth rendah. Jadi, ini adalah solusi sempurna untuk bertukar data antara beberapa perangkat IoT.

Komunikasi MQTT berfungsi sebagai sistem publish dan subscribe. Perangkat memublikasikan pesan tentang topik tertentu. Semua perangkat yang berlangganan topik tersebut menerima pesan tersebut.

Broker MQTT bertanggung jawab untuk menerima semua pesan, memfilter pesan, memutuskan siapa yang tertarik padanya, dan kemudian menerbitkan pesan ke semua klien yang berlangganan.

B. Python Web Server with Flask

Untuk menginstal Flask, kami harus menginstal pip.

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
pi@raspberrypi ~ $ sudo apt-get install python-pip python-flask
```

Kemudian, kami menggunakan pip untuk menginstal Flask dan dependensinya

```
pi@raspberrypi ~ $ sudo pip install flask
```

C. Install Python Paho-MQTT

Paket Paho-MQTT menyediakan kelas klien yang memungkinkan aplikasi untuk terhubung ke broker MQTT untuk mempublikasikan pesan, dan untuk berlangganan topik dan menerima pesan yang dipublikasikan. Dalam contoh ini, server web Python akan mempublikasikan pesan ke ESP32 untuk mengaktifkan dan menonaktifkan GPIO.

Untuk menginstal paho-mqtt jalankan perintah berikut

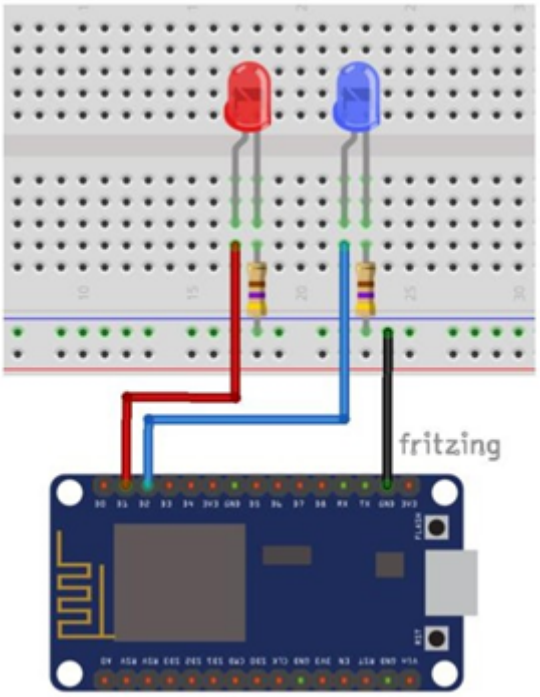
```
pip install paho-mqtt
```

D. Instal Aplikasi Penunjang

Kami menggunakan Visual Studio Code untuk menunjang pembuatan coding python dan html yang akan dibuat.

III. Tahap Pengerjaan

1. Pembuatan Skema Rangkaian



2. Pembuatan ESP32 code di Arduino IDE


```
// Loading the ESP32 WiFi library and the PubSubClient library
// catur && santi
#include <WiFi.h>
#include <PubSubClient.h>
#define WIFI_TIMEOUT_MS 20000

// Change the credentials below, so your ESP8266 connects to your router const char* ssid = "UGM-Hotspot";
const char* password = "";

// Change the variable to your Raspberry Pi IP address, so it connects to your MQTT broker
const char* mqtt_server = "10.33.162.50";
```

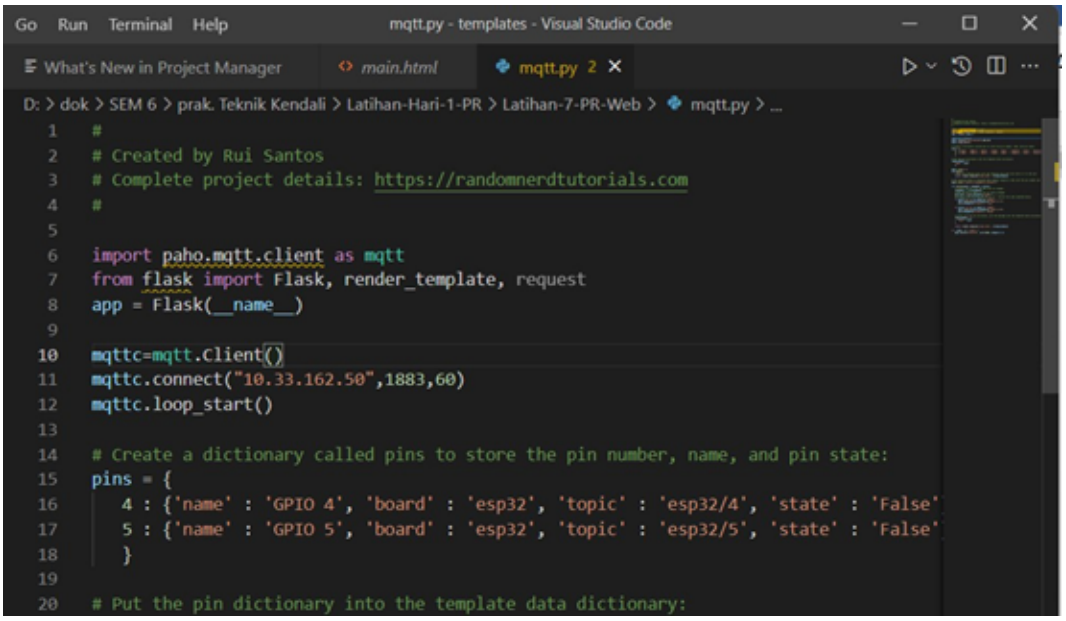
```
// Initializes the espClient WiFiClient espClient; PubSubClient client(espClient);

// Connect an LED to each GPIO of your ESP8266 const int ledGPIO5 = 27;
const int ledGPIO4 = 26;

// Don't change the function below. This functions connects your ESP8266 to your router
void connectToWiFi(){ Serial.print(""); Serial.println("Connecting to WiFi"); WiFi.mode(WIFI_STA); WiFi.begin(ssid, password);

unsigned long startAttemptTime = millis();

while (WiFi.status() != WL_CONNECTED && millis () - startAttemptTime
< WIFI_TIMEOUT_MS){
Serial.print("."); delay(500);
}

if(WiFi.status() != WL_CONNECTED){ Serial.println("Failed!");
}
else { Serial.print("Connected"); Serial.println(WiFi.localIP());
}
}

// This functions is executed when some device publishes a message to a topic that your ESP8266 is subscribed to
// Change the function below to add logic to your program, so when a device publishes a message to a topic that
// your ESP8266 is subscribed you can actually do something void callback(String topic, byte* message, unsigned int length) { Serial.print("Message arrived on topic: ");
Serial.print(topic); Serial.print(". Message: "); String messageTemp;

for (int i = 0; i < length; i++) { Serial.print((char)message[i]); messageTemp += (char)message[i];
}

Serial.println();

// Feel free to add more if statements to control more GPIOs with MQTT
```

```
// If a message is received on the topic home/office/esp1/gpio2, you check if the message is either 1 or 0. Turns the ESP GPIO according to the message if(topic=="esp32/4"){
Serial.print("Changing GPIO 4 to "); if(messageTemp == "1"){ digitalWrite(ledGPIO4, HIGH); Serial.print("On");
}
else if(messageTemp == "0"){ digitalWrite(ledGPIO4, LOW); Serial.print("Off");
}
}
if(topic=="esp32/5"){ Serial.print("Changing GPIO 5 to "); if(messageTemp == "1"){ digitalWrite(ledGPIO5, HIGH); Serial.print("On");
}
else if(messageTemp == "0"){ digitalWrite(ledGPIO5, LOW); Serial.print("Off");
}
}
Serial.println();
}

// This functions reconnects your ESP8266 to your MQTT broker
// Change the function below if you want to subscribe to more topics with your ESP8266
void reconnect() {
// Loop until we're reconnected while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
// Attempt to connect
/*
YOU NEED TO CHANGE THIS NEXT LINE, IF YOU'RE HAVING PROBLEMS WITH MQTT MULTIPLE CONNECTIONS
To change the ESP device ID, you will have to give a unique name to the ESP8266.

Here's how it looks like now:
if (client.connect("ESP8266Client")) {
If you want more devices connected to the MQTT broker, you can do it like this:
if (client.connect("ESPOffice")) { Then, for the other ESP:
if (client.connect("ESPGarage")) {
That should solve your MQTT multiple connections problem

THE SECTION IN loop() function should match your device name
*/
if (client.connect("ESP32Client")) { Serial.println("connected");
// Subscribe or resubscribe to a topic
```

```
// You can subscribe to more topics (to control more LEDs in this example)
client.subscribe("esp32/4");
client.subscribe("esp32/5");
} else { Serial.print("failed, rc="); Serial.print(client.state());
Serial.println("try again in 5 seconds");
// Wait 5 seconds before retrying delay(5000);
}
}
}

// The setup function sets your ESP GPIOs to Outputs, starts the serial communication at a baud rate of 115200
// Sets your mqtt broker and sets the callback function
// The callback function is what receives messages and actually controls the LEDs
void setup() { pinMode(ledGPIO4, OUTPUT); pinMode(ledGPIO5, OUTPUT);

Serial.begin(115200); connectToWiFi(); client.setServer(mqtt_server, 1883); client.setCallback(callback);
}

// For this project, you don't need to change anything in the loop function.
// Basically it ensures that you ESP is connected to your broker

void loop() {
if (!client.connected()) { reconnect();
}
if(!client.loop())
/*
YOU NEED TO CHANGE THIS NEXT LINE, IF YOU'RE HAVING PROBLEMS WITH MQTT MULTIPLE CONNECTIONS
To change the ESP device ID, you will have to give a unique name to the ESP8266.
Here's how it looks like now: client.connect("ESP8266Client");
If you want more devices connected to the MQTT broker, you can do it like this:
client.connect("ESPOffice"); Then, for the other ESP: client.connect("ESPGarage");
That should solve your MQTT multiple connections problem

THE SECTION IN recionnect() function should match your device name
*/ client.connect("ESP32Client");
}
```

3. Pembuatan Python Script MQTT di Visual Studio Code

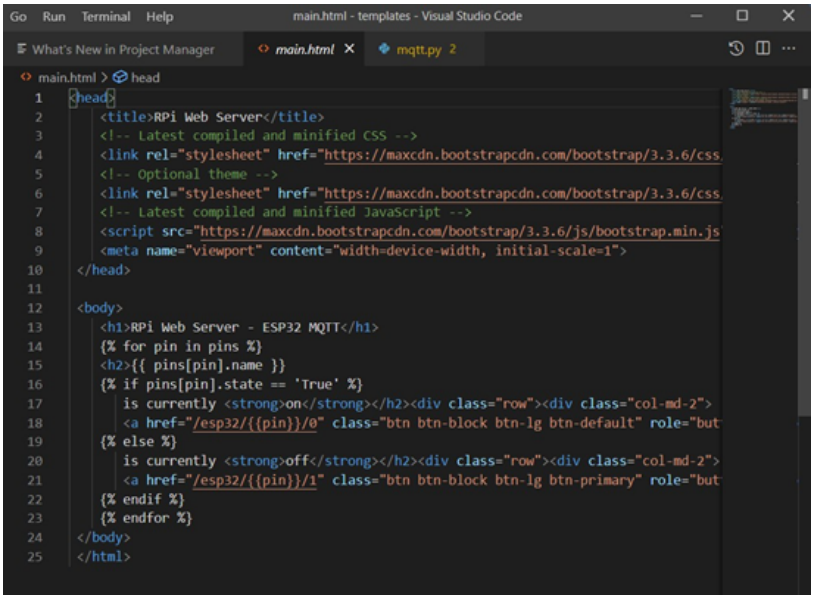
Ini adalah skrip inti dari aplikasi kami. Ini mengatur server web dan ketika tombol-tombol ini ditekan, ia menerbitkan pesan MQTT ke ESP32.




```
# Created by Rui Santos
# Complete project details: https://randomnerdtutorials.com #
import paho.mqtt.client as mqtt
from flask import Flask, render_template, request
app = Flask( name )

mqttc=mqtt.Client()
mqttc.connect("10.33.162.50",1883,60)
mqttc.loop_start()

# Create a dictionary called pins to store the pin number, name, and pin state:
pins = {
   4  : {'name' : 'GPIO 4', 'board' : 'esp32', 'topic' : 'esp32/4', 'state' : 'False'},
   5  : {'name' : 'GPIO 5', 'board' : 'esp32', 'topic' : 'esp32/5', 'state' : 'False'}
   }

# Put the pin dictionary into the template data dictionary:
templateData = {
'pins' : pins
}
```

```
@app.route("/")
def main():
# Pass the template data into the template main.html and return it to the user
    return render_template('main.html', **templateData)

# The function below is executed when someone requests a URL with the pin number and action in it:
@app.route("/<board>/<changePin>/<action>")
def action(board, changePin, action):
    # Convert the pin from the URL into an integer:
    changePin = int(changePin)
    # Get the device name for the pin being changed:
    devicePin = pins[changePin]['name']
    # If the action part of the URL is "on," execute the code indented below:
    if action == "1" and board == 'esp32':
        mqttc.publish(pins[changePin]['topic'],"1")
        pins[changePin]['state'] = 'True'

    if action == "0" and board == 'esp32':
        mqttc.publish(pins[changePin]['topic'],"0")
        pins[changePin]['state'] = 'False'

        # Along with the pin dictionary, put the message into the template data dictionary:
    templateData = {
    'pins' : pins
    }
    return render_template('main.html', **templateData)

if name == " main ":
    app.run(host='0.0.0.0', port=8080, debug=False)
```

4. Pembuatan File HTML



<head>

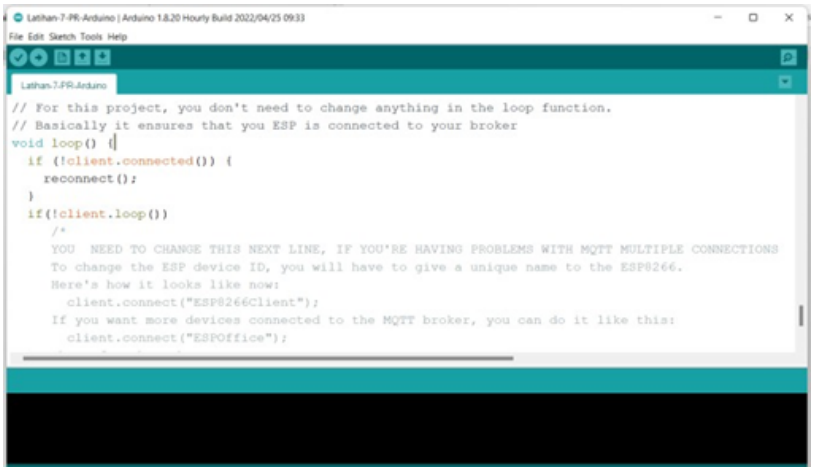
<title>RPi Web Server</title>
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjW PGmkzs7" crossorigin="anonymous">
<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap- theme.min.css" integrity="sha384-fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV
+0En5r" crossorigin="anonymous">

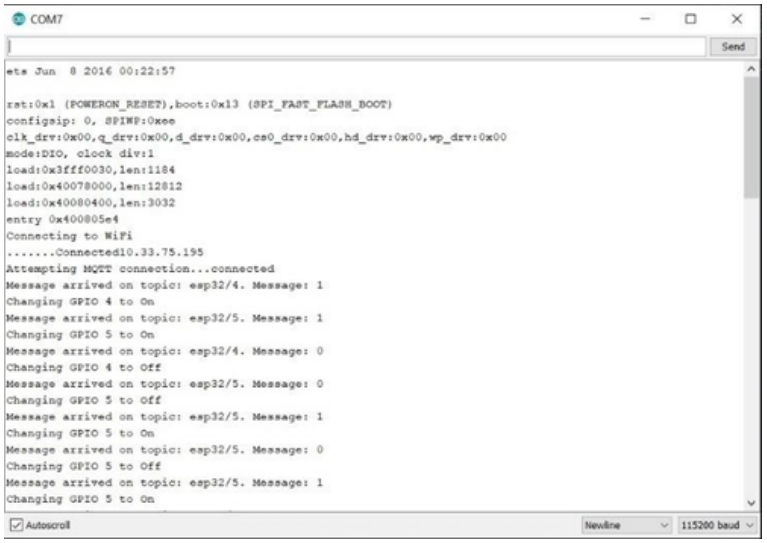
<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHV P9aJ7xS" crossorigin="anonymous"></script>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>

```
<body>
  <h1>RPi Web Server - ESP32 MQTT</h1>
  {% for pin in pins %}
  <h2>{{ pins[pin].name }}
  {% if pins[pin].state == 'True' %}
    is currently <strong>on</strong></h2><div
class="row"><div class="col- md-2">
    <a href="/esp32/{{pin}}/0" class="btn btn-block
btn-lg btn-default" role="button">Turn
off</a></div></div>

  {% else %}

    is currently <strong>off</strong></h2><div
class="row"><div class="col-md-2">

<a href="/esp32/{{pin}}/1" class="btn btn-block btn-lg btn-primary"
role="button">Turn on</a></div></div>

  {% endif %}

  {% endfor %}
</body>
</html>
```

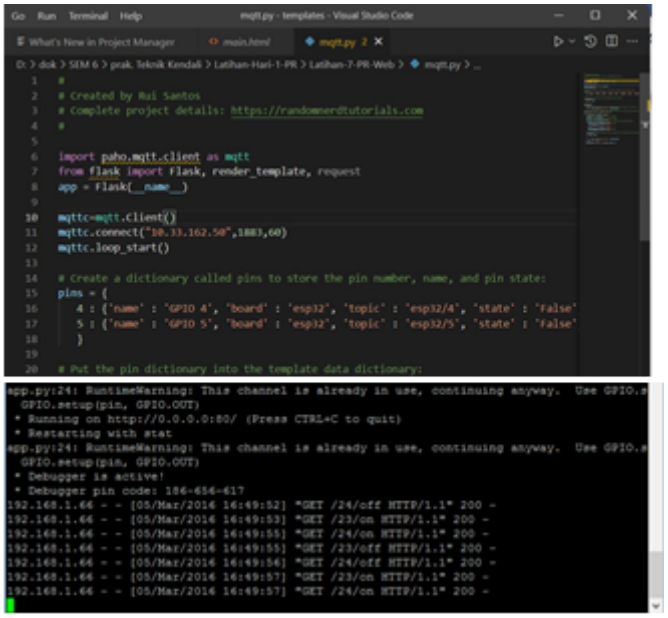
IV. Upload Program dan Launch the Web Server

1. Upload ESP32 Program Code di Arduino IDE

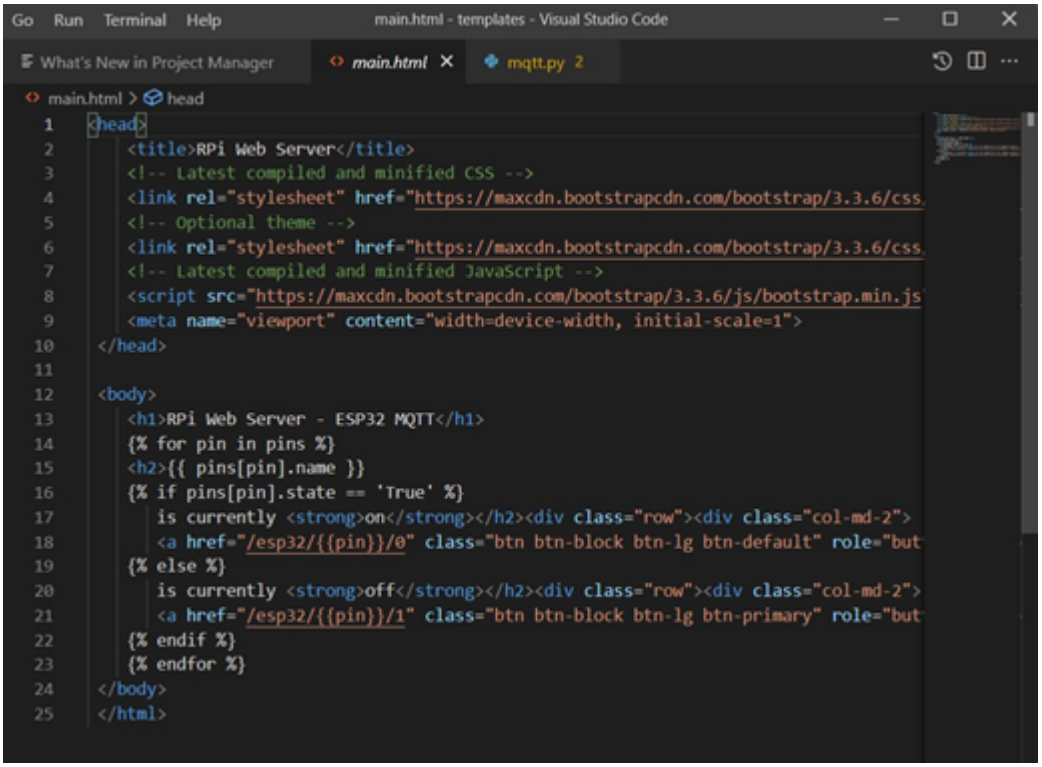




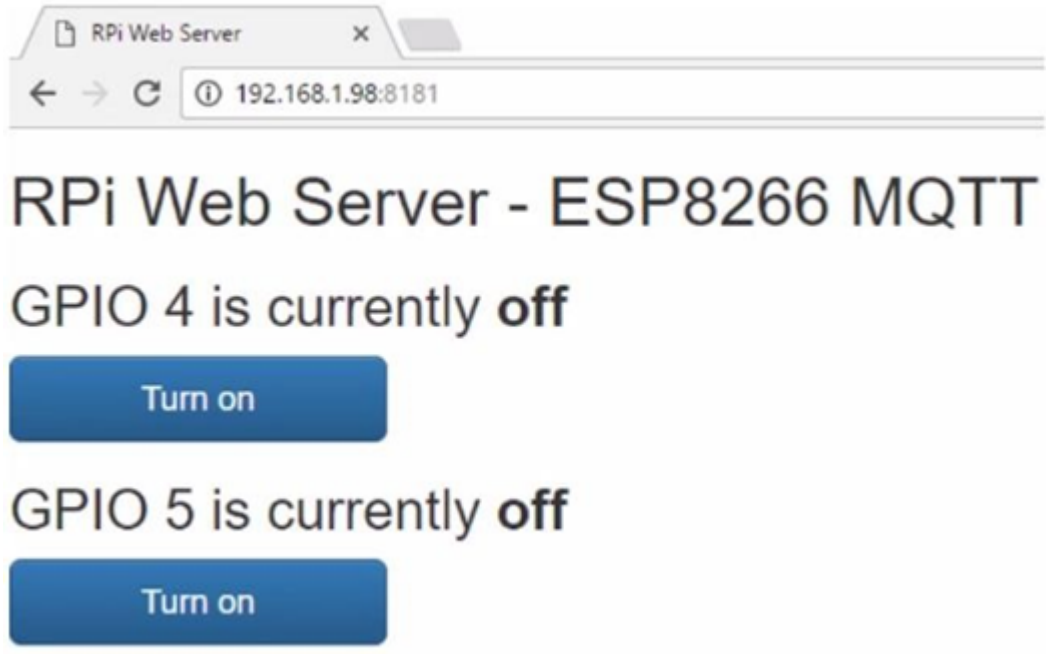
2. Run mqtt program with python app.py



3. Run HTML code di Visual Studio Code



4. Demonstrasi Web Server



Untuk membuka alamat web server yang dibuat, kami menggunakan ip addres dan penyesuaian port yang dipakai, kedua button “Turn on” bisa dioperasikan untuk connect ke rangkaian ESP32 yang memberikan perintah on/off pada kedua LED yang tersambung. Status “Turn on”dan “off” pada web akan berubah menjadi “Turn Off”dan “on”jika kami tekan tombol Turn on.

Laporan 3. ***Cloud IoT Think Speak***

BAB 1. Pengenalan Kasus

Kasus ini bertujuan untuk mengetahui bagaimana cara mengirimkan data sensor Ln35 dan nilai ADC Potensiometer menggunakan plot IoT ke website Thingspeak.

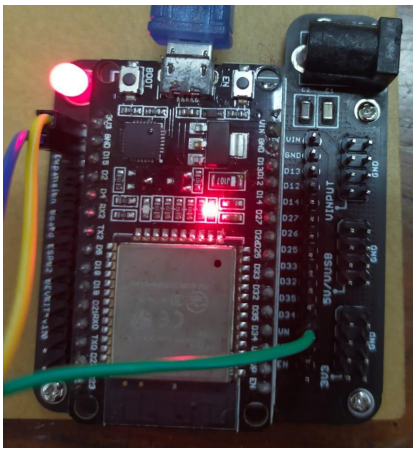
Internet of Things (IoT) adalah sebuah konsep yang bertujuan untuk memperluas manfaat konektivitas internet yang terhubung secara terus menerus -kemampuan untuk berbagi data, remote control, dan lain-lain, serta pada objek di dunia nyata. Misalnya, makanan, elektronik, barang koleksi, peralatan apapun, termasuk makhluk hidup yang semuanya terhubung ke jaringan lokal dan global melalui sensor yang tertanam dan aktif.

BAB 2. Landasan Teori

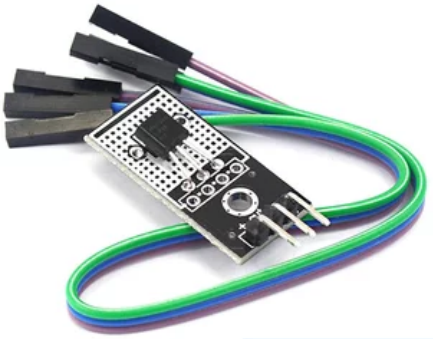
Berikut adalah contoh program Arduino IoT menggunakan ESP32 Wireless Module:

Step 1: Menyiapkan material yang dibutuhkan

1. ESP-32

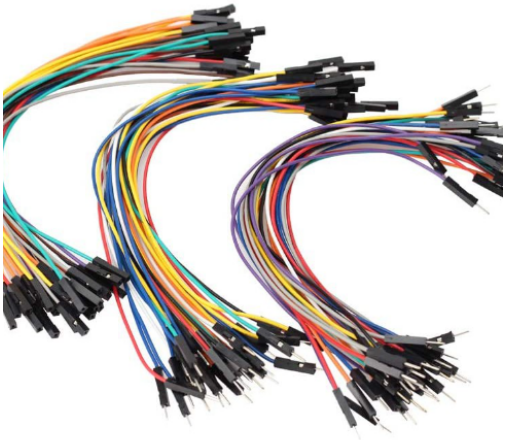


2. Sensor LM35

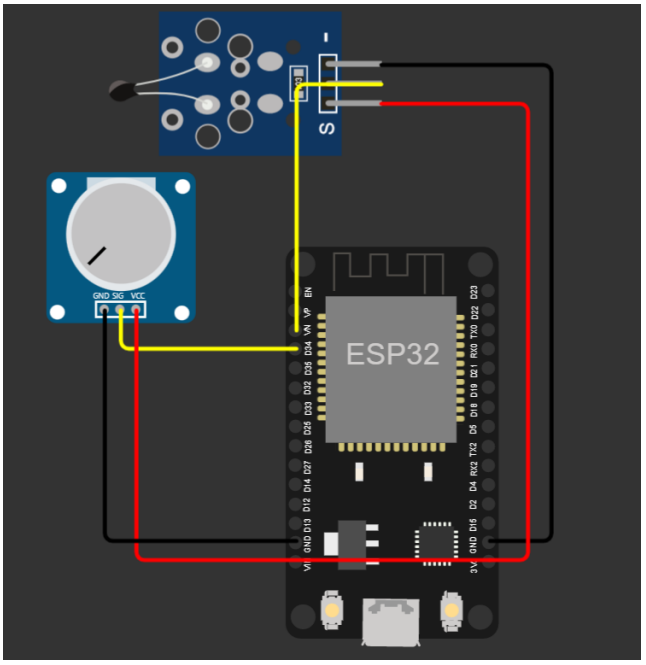


3. Potensio

4. Male to Female Jumper Wires



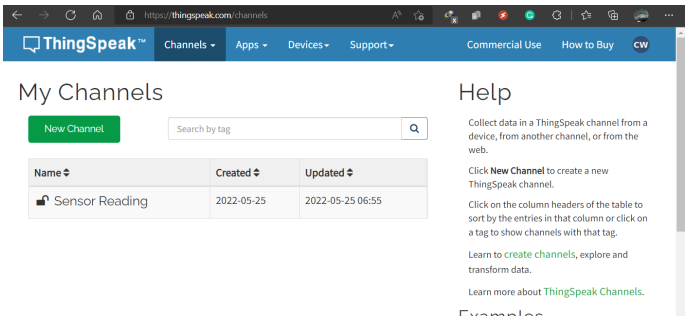
Step 2: Setup



Menghubungkan kaki data potensio ke pin 34 dan kedua kaki tepi masing masing ke 3v3 dan gnd, Kaki LM35 tengah ke pin 36 dan kedua kaki tepi ke 3v3 dan gnd.

Step 3: Membuat Akun Thingspeak

Setelah menghubungkan semua material yang dibutuhkan, kunjungi https://thingspeak.com/ dan buat akun disana. Kemudian buat new channel.

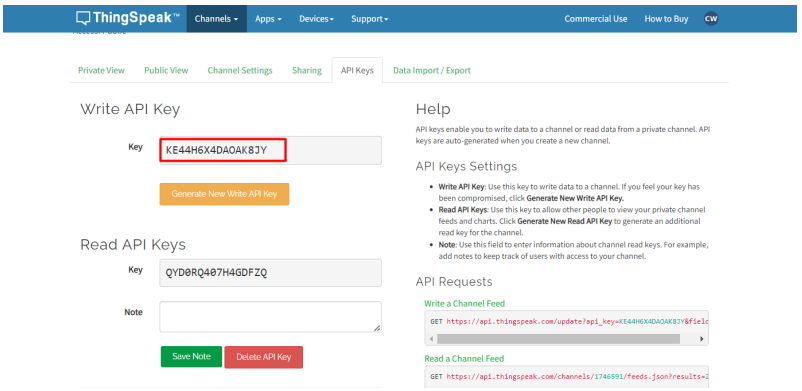


Membuat akun thingspeak dan membuat 1 channel untuk menampilkan data yang akan kita kirim.

Setelah menghubungkan semua material yang dibutuhkan, kunjungi https://thingspeak.com/ dan buat akun disana. Kemudian buat new channel.

Step 4: Kode

Setelah itu, upload kode dibawah ini dan pastikan seluruh data telah terisi meliputi API key dan Field name, serta pengaturan Access Point yang akan terhubung ke Modul Nirkabel ESP-32 (AP dan Password).



(API Key Channel ThinkSpeak)

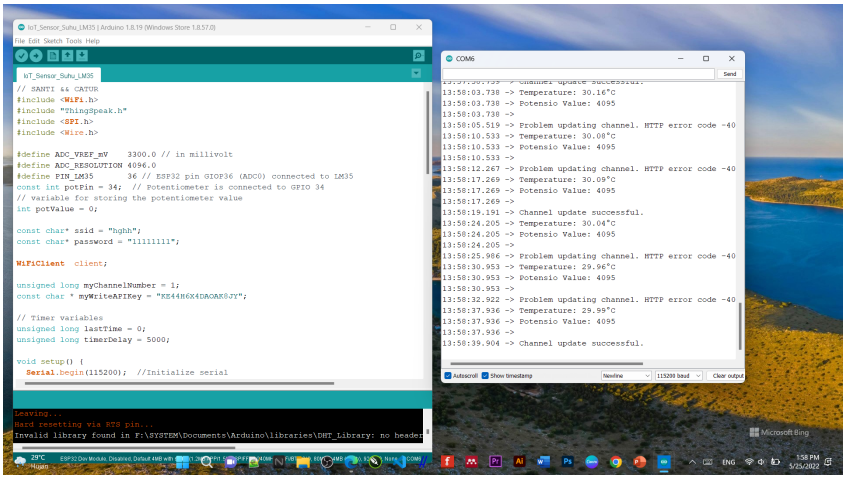
```
// SANTI && CATUR
#include <WiFi.h>
#include "ThingSpeak.h"
#include <SPI.h>
#include <Wire.h>
```

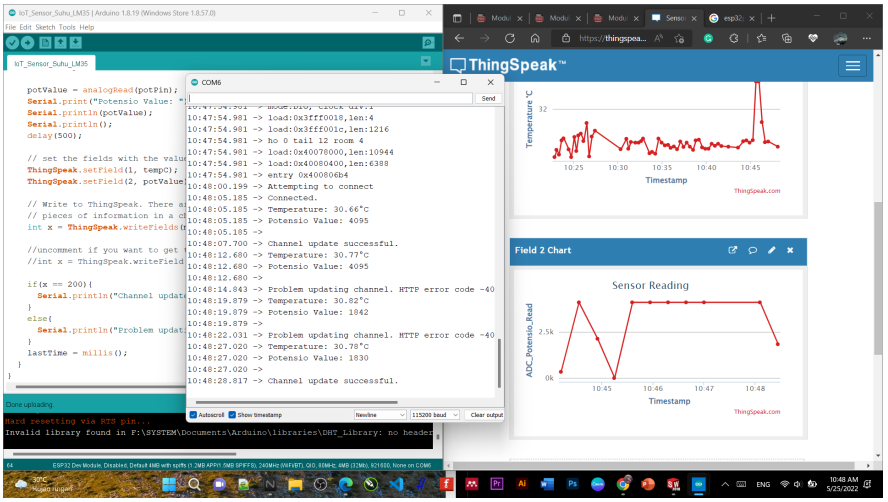
```
#define ADC_VREF_mV    3300.0 // in millivolt
#define ADC_RESOLUTION 4096.0
#define PIN_LM35       36 // ESP32 pin GIOP36 (ADC0) connected to LM35
const int potPin = 34;  // Potentiometer is connected to GPIO 34
// variable for storing the potentiometer value
int potValue = 0;

const char* ssid = "hghh";    //Password Wifi
const char* password = "11111111";  //Password Wifi

WiFiClient  client;

unsigned long myChannelNumber = 1;
const  char  *  myWriteAPIKey  =  "KE44H6X4DAOAK8JY";  //AMBIL  DARI
Channel WEB THINKSPEAK

// Timer variables
unsigned long lastTime = 0;
unsigned long timerDelay = 5000;

void setup() {
  Serial.begin(115200);  //Initialize serial

  WiFi.mode(WIFI_STA);

  ThingSpeak.begin(client);  // Initialize ThingSpeak
}

void loop() {
  if ((millis() - lastTime) > timerDelay) {

    // Connect or reconnect to WiFi
    if(WiFi.status() != WL_CONNECTED){
      Serial.print("Attempting to connect");
      while(WiFi.status() != WL_CONNECTED){
        WiFi.begin(ssid, password);
        delay(5000);
      }
      Serial.println("\nConnected.");
    }

    // Get a new temperature reading
```

```
  // 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 / 100;
  // convert the °C to °F
  float tempF = tempC * 9 / 5 + 32;

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

  potValue = analogRead(potPin);
  Serial.print("Potensio Value: ");
  Serial.println(potValue);
  Serial.println();
  delay(400);

  // set the fields with the values
  ThingSpeak.setField(1, tempC);
  ThingSpeak.setField(2, potValue);

   // Write to ThingSpeak. There are up to 8 fields in a channel, allowing you to store up to 8 different
  // pieces of information in a channel.  Here, we write to field 1.
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

  //uncomment if you want to get temperature in Fahrenheit
        //int  x  =  ThingSpeak.writeField(myChannelNumber,  1,  temperatureF, myWriteAPIKey);

  if(x == 200){
    Serial.println("Channel update successful.");
  }
  else{
    Serial.println("Problem updating channel. HTTP error code " + String(x));
```

```
    }
    lastTime = millis();
  }
}
```

Step 5: Monitor it





Setelah terupload, pastikan agar wifi terhubung dengan koneksi internet. Kemudian buka serial monitoring untuk melihat data AT command. Data yang dikirimkan merupakan data acak. Data dapat diubah dengan data sensor atau lain-lain.

BAB 3. Kesimpulan

Data sensor suhu LM35 dan Potensio meter dapat terkirim ke web Thingspeak tepatnya pada channel yang sudah kita buat.

ThingSpeak for IoT Projects. https://thingspeak.com/. Diakses tanggal 25 Mei 2022.

Laporan 4. *Cloud IoT Firebase*

BAB 1. Pengenalan Kasus

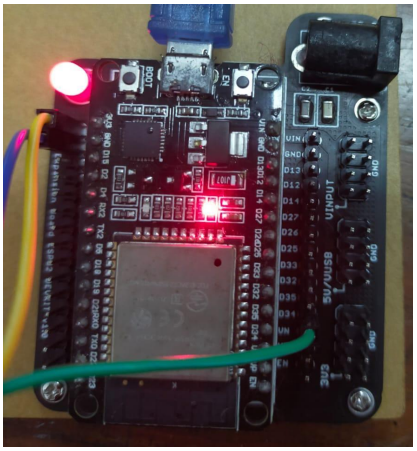
Kasus ini bertujuan untuk mengetahui bagaimana cara mengirimkan text ke suatu database yang bernama firebase.

Firebase adalah BaaS (Backend as a Service) yang saat ini dimiliki oleh Google. Firebase adalah solusi yang ditawarkan oleh google untuk mempermudah Mobile Apps Developer. Banyaknya fitur yang ditawarkan oleh firebase memungkinkan apps developer mengembangkan aplikasi dengan mudah. Pada proyek akhir ini fitur pada firebase yang digunakan adalah firebase Real Time Database. Firebase Real Time Database adalah fitur database yang dapat diakses secara real time oleh pengguna aplikasi (yadza, Q., dkk., 2018)

BAB 2. Penyelesaian Kasus

Berikut adalah contoh program Arduino IoT menggunakan ESP-32 Wireless Module:

Step 1: Menyiapkan material yang dibutuhkan

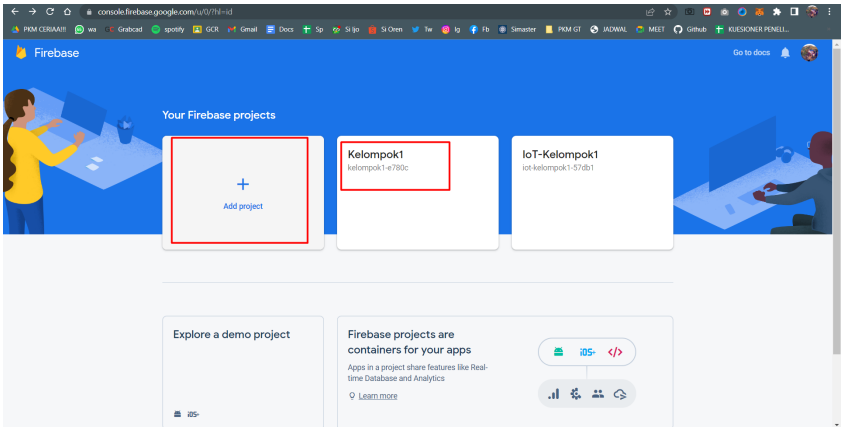


1. ESP-32 Wireless Module

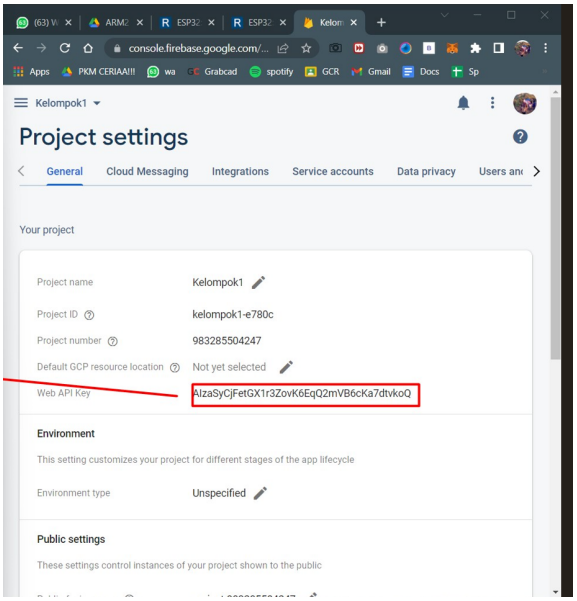
Step 2: SetUp

Hubungkan ESP-32 dengan komputer

Step 3: Buat akun Firebase

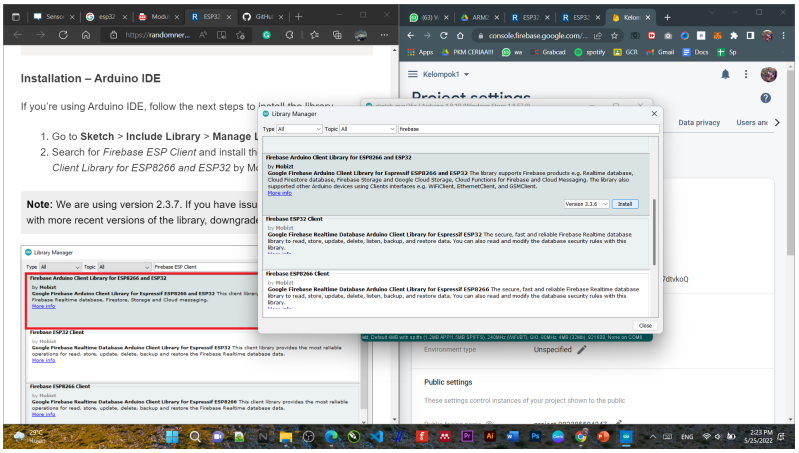


Setelah menghubungkan semua material yang dibutuhkan, kunjungi https://console.firebase.google.com/ dan buat akun menggukan akun google. Kemudian klik add project. dalam percobaan kali ini kami sudah membuat project bernama Kelompok1.



Setelah itu buka Project setting dan copy/simpan Web API key dari project yang kita buat.

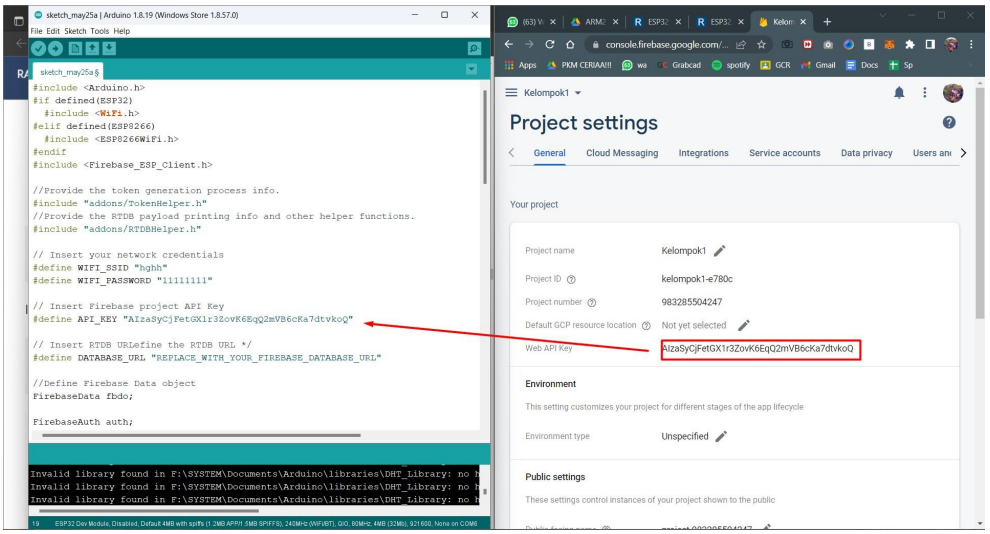
Step 4: Install Library



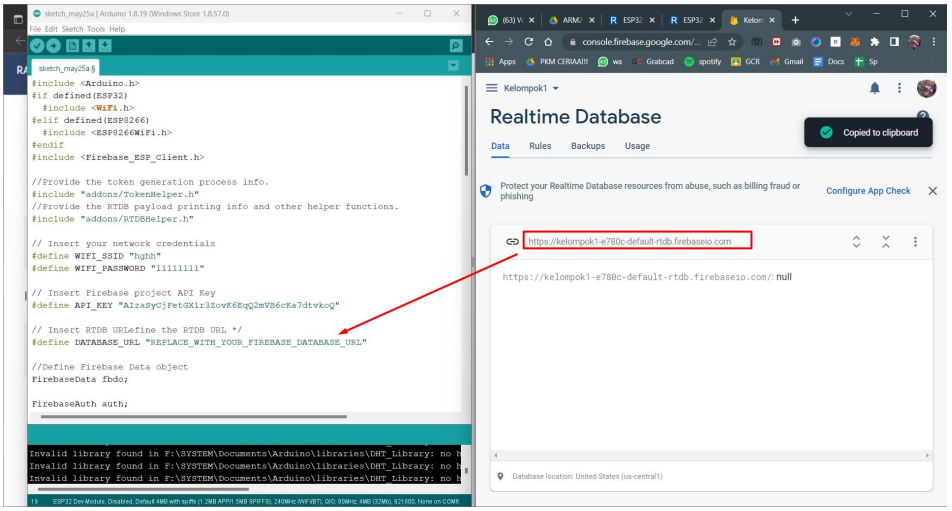
Install library pada Arduino IDE

Step 5: Kode

Setelah itu, upload kode dibawah ini dan pastikan seluruh data telah terisi meliputi API key dan web project dari project yang ada sudah dibuat pada Firebase, serta pengaturan Access Point yang akan terhubung ke Modul Nirkabel ESP-32(AP dan Password) dari WIFI yang kita pakai.



(Web API Key)



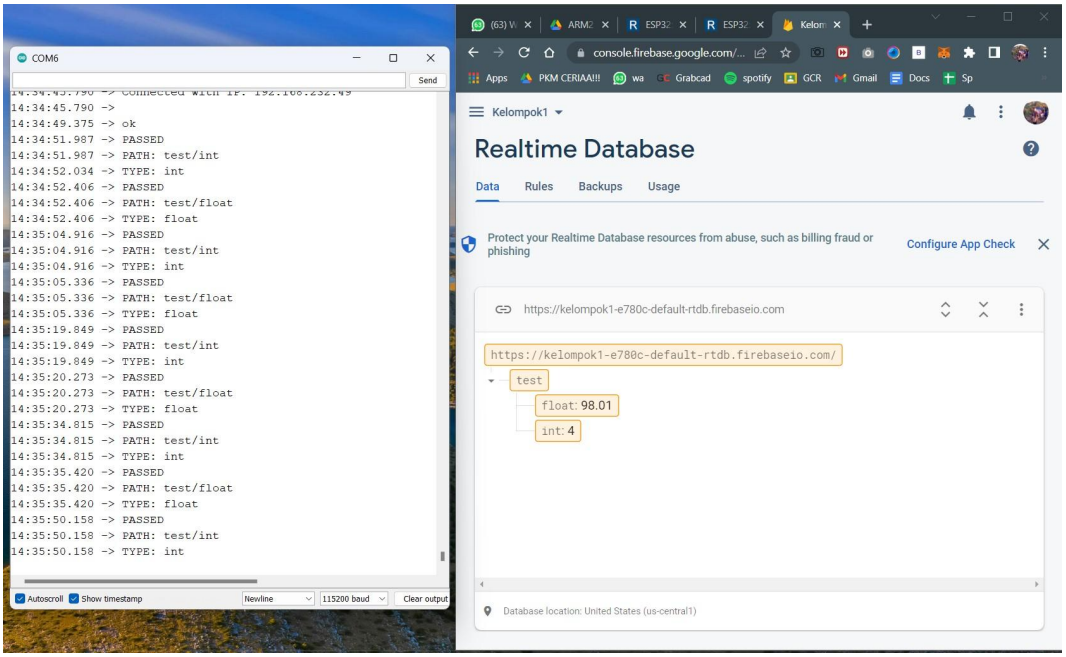
(Web ULR Project)

```
#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "hghh"
#define WIFI_PASSWORD "11111111"

// Insert Firebase project API Key
#define API_KEY "AIzaSyCjFetGX1r3ZovK6EqQ2mVB6cKa7dtvkoQ"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL
"https://kelompok1-e780c-default-rtdb.firebaseio.com/"

//Define Firebase Data object
FirebaseData fbdo;
```

```
FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
int count = 0;
bool signupOK = false;

void setup(){
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;

  /* Sign up */
  if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("ok");
    signupOK = true;
  }
  else{
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }

  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
}
```

```
void loop(){
  if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 15000 ||
sendDataPrevMillis == 0)){
    sendDataPrevMillis = millis();
    // Write an Int number on the database path test/int
    if (Firebase.RTDB.setInt(&fbdo, "test/int", count)){
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
    }
    else {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
    }
    count++;

    // Write an Float number on the database path test/float
    if (Firebase.RTDB.setFloat(&fbdo, "test/float", 0.01 + random(0,100))){
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
    }
    else {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
    }
  }
}
```

Step 6: Monitor it



Setelah terupload, pastikan agar wifi terhubung dengan koneksi internet. Kemudian buka serial monitoring untuk melihat data AT command. Data yang dikirimkan merupakan test berupa data Float dan Int. Data dapat diubah dengan data sensor atau lain-lain.

BAB 3. Kesimpulan

Data dengan Tipe Int dan Float dapat terkirim ke Firebase tepatnya pada Project yang sudah kita buat. Data yang dikirim dapat berupa data sensor atau data yang lainnya.

Daftar Pustaka

Syadza, Q., Permana, A.G. and Ramadan, D.N., 2018. Pengontrolan Dan Monitoring Prototype Green House Menggunakan Mikrokontroler Dan Firebase. EProceedings of Applied Science, 4(1).

Laporan 5. ***Cloud IoT Arduino IoT***

BAB 1. Pengenalan Kasus

Kasus ini bertujuan untuk mengetahui bagaimana cara mengirimkan data ke Platform IoT Arduino Cloud IoT dan bagaimana melakukan perintah kontrol kepada aktuator yang berada jauh dengan tempat kita namun masih ada internet pada tempat tersebut.

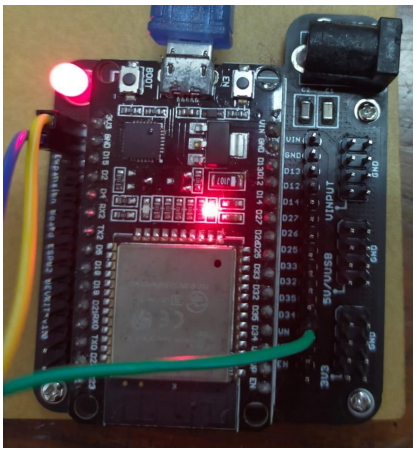
Kasus pada kali ini adalah pengendalian nyala lampu dan pembacaan hasil suhu dan kelembaban dari sensor DHT11 menggunakan Platform IoT - Arduino cloud IoT.

BAB 2. Penyelesaian Kasus

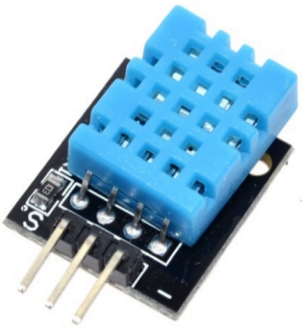
Berikut adalah Step penyelesaian kasus :

Step 1: Menyiapkan material yang dibutuhkan

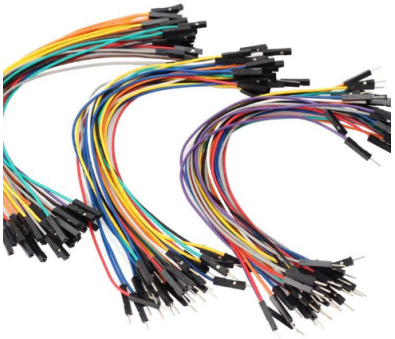
5. ESP-32



6. DHT 11

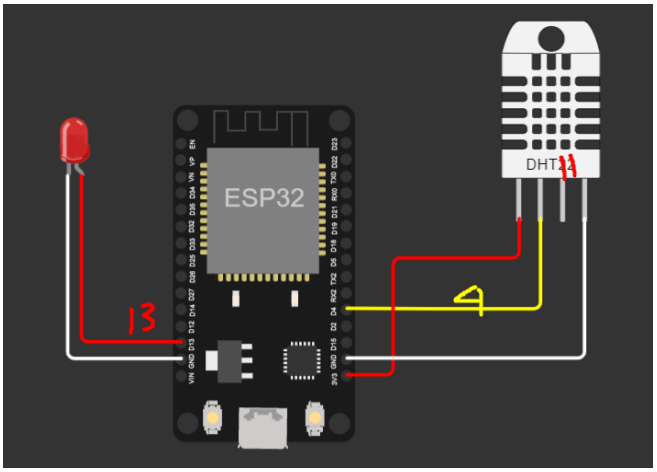


7. Jumper Wires



8. Lampu LED

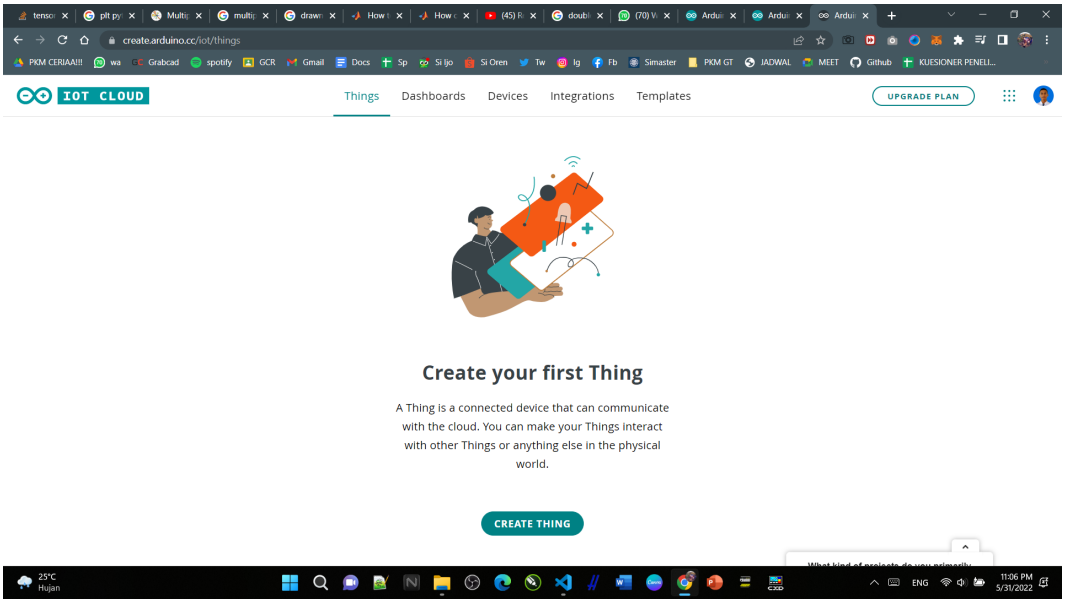
Step 2: SetUp



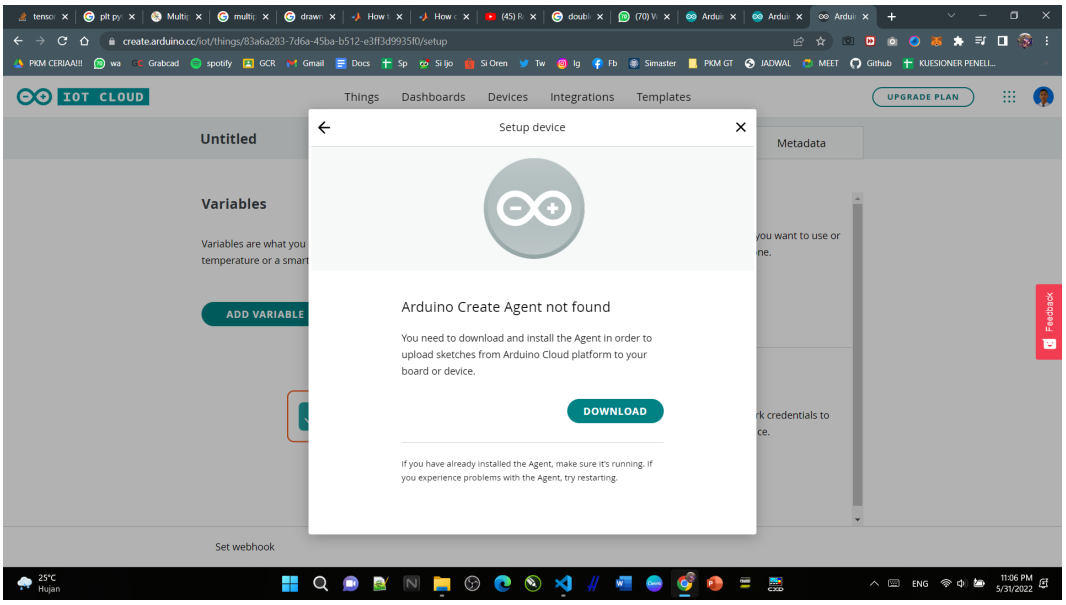
Data DHT ke pin 4 ESP32, VCC=3v3, GND=GND
VCC LED ke pin 13 ESP32 , GND=GND

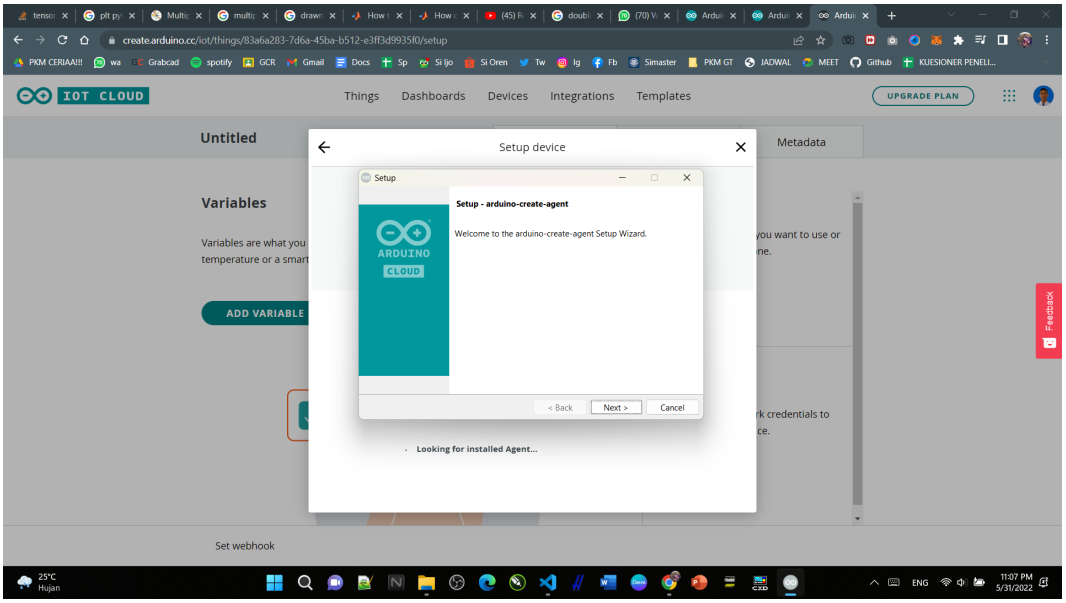
Step 3: Koneksikan ESP32 ke Laptop menggunakan Micro USB

Step 4: Membuat Akun pada Platform Arduino IoT, Buat Project dengan klik Create Things

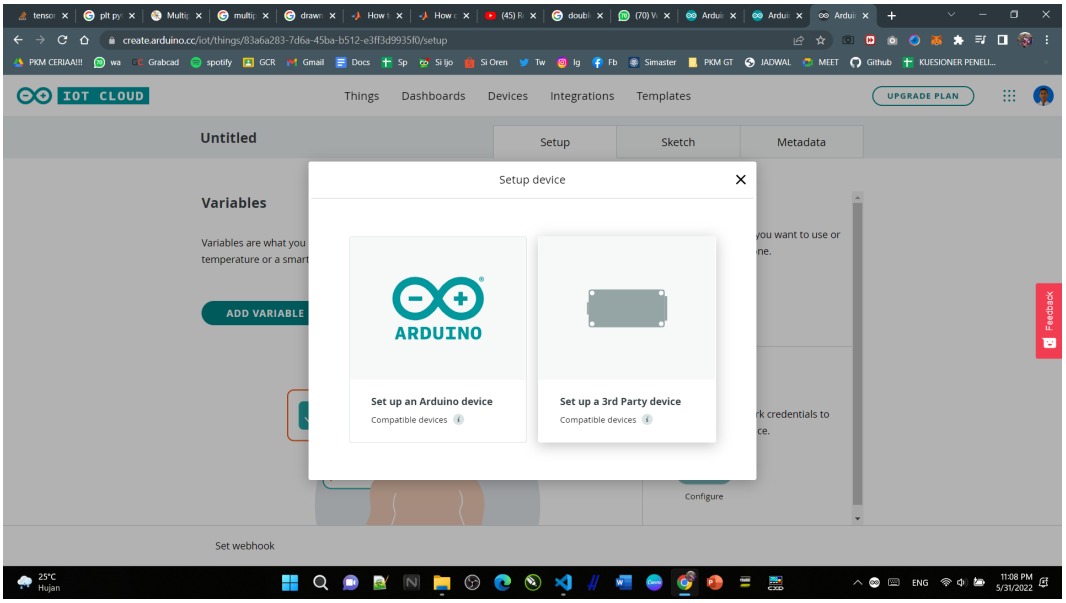


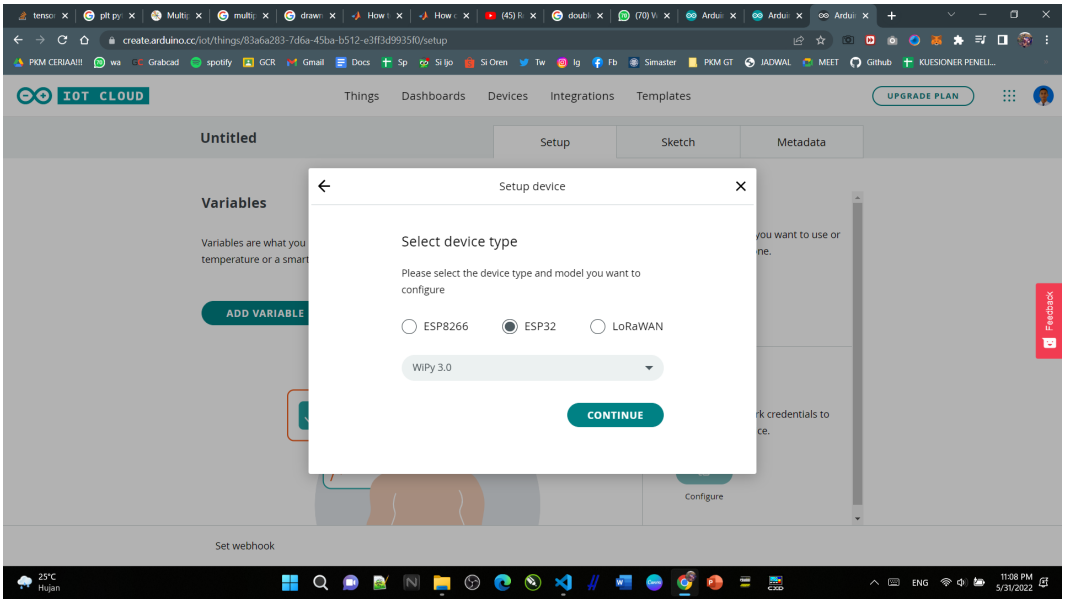
Step 5: Install arduino step agent:



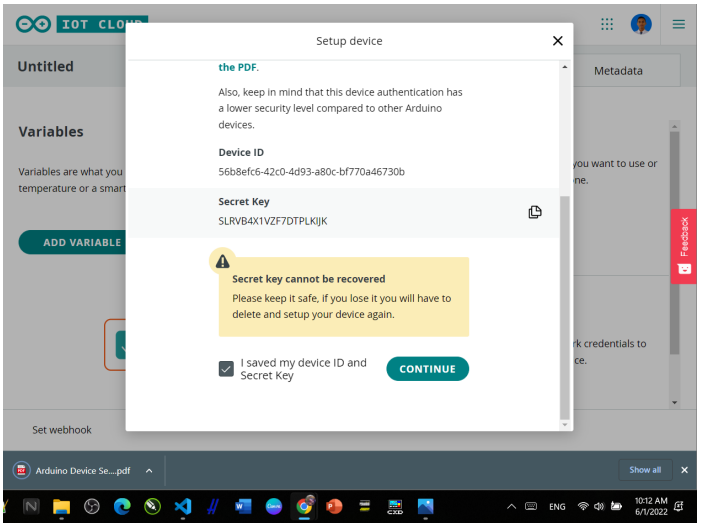


Step 6: Mendefinisikan perangkat yang digunakan, pada praktikum ini digunakan ESP32

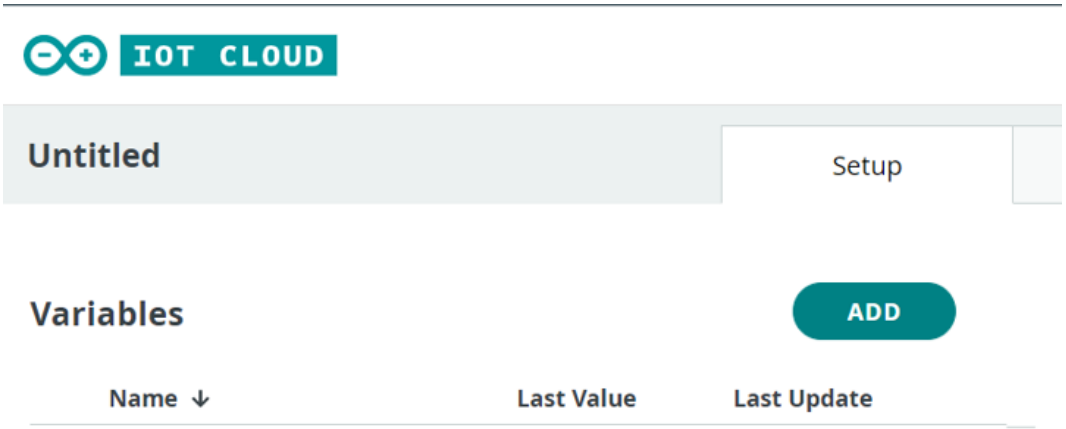


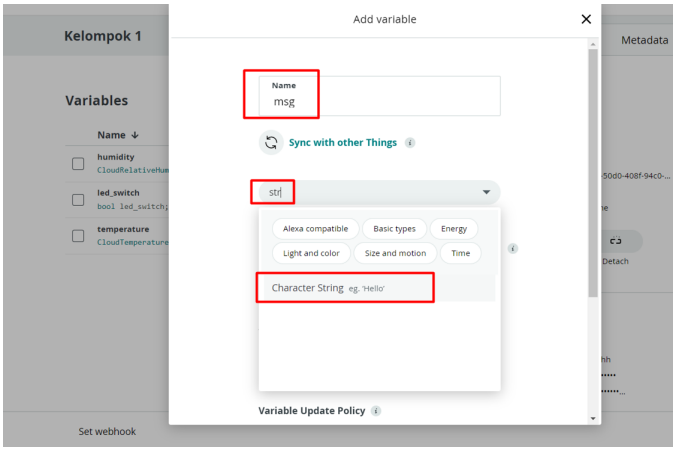


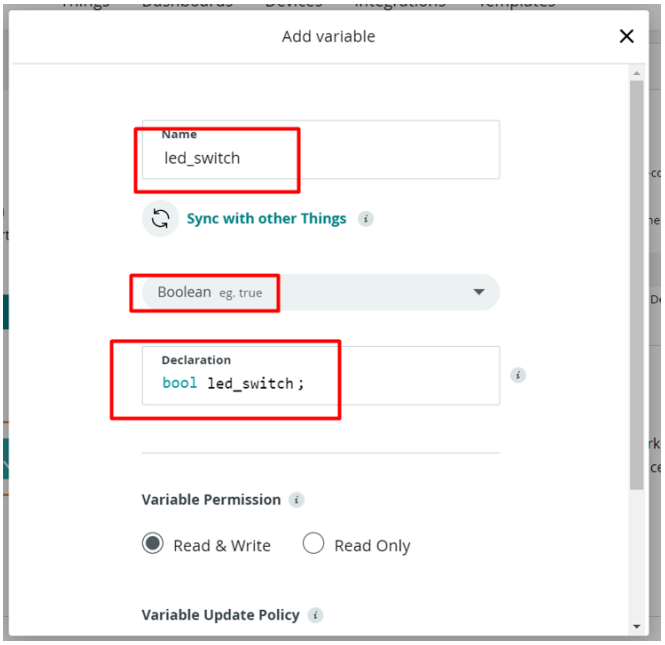
Kemudian akan didapatkan Secret Key seperti gambar dibawah



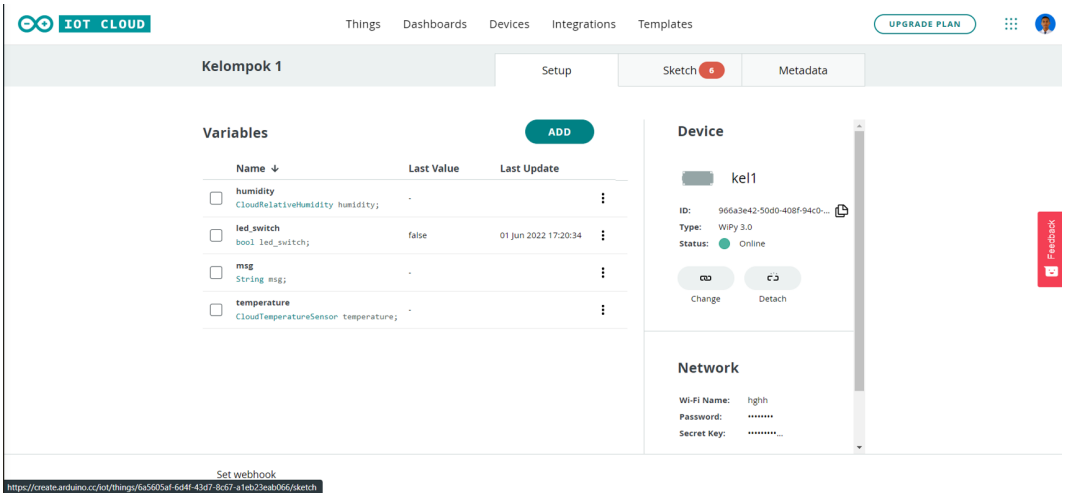
Step 7: Membuat Variabel yang dibutuhkan dengan melakukan klik ADD



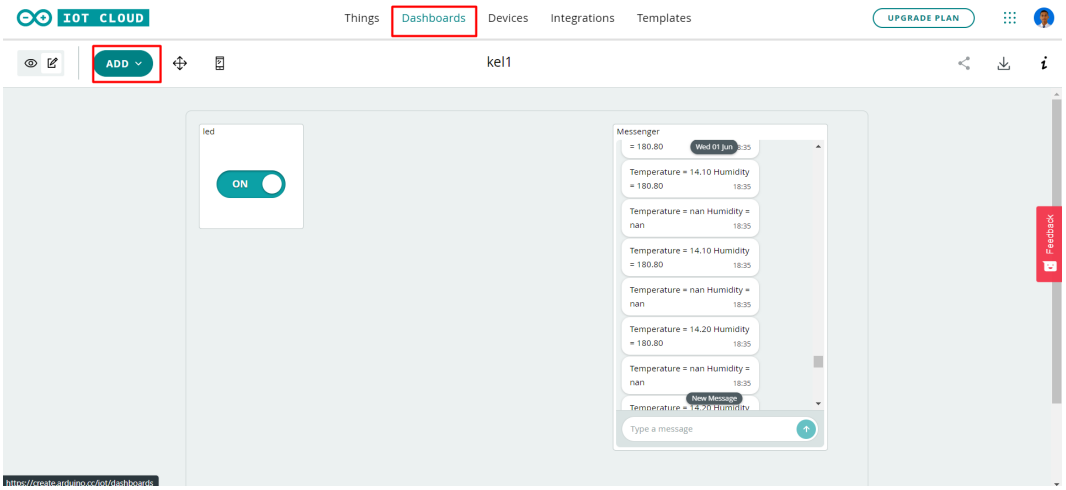




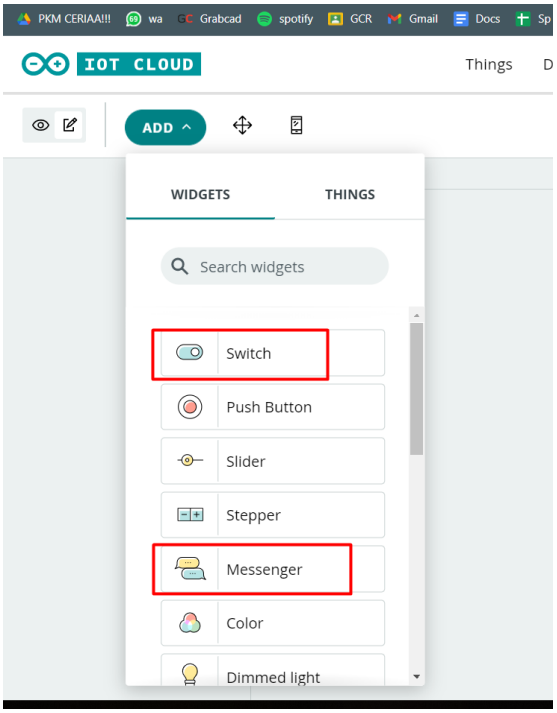
Membuat juga variabel temperature dan humidity



Step 8: Membuat dashboard IoT

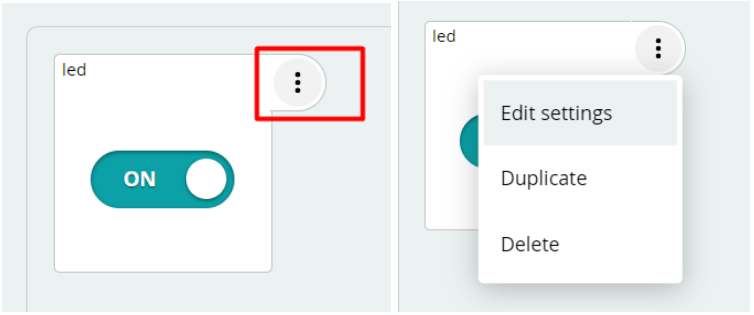


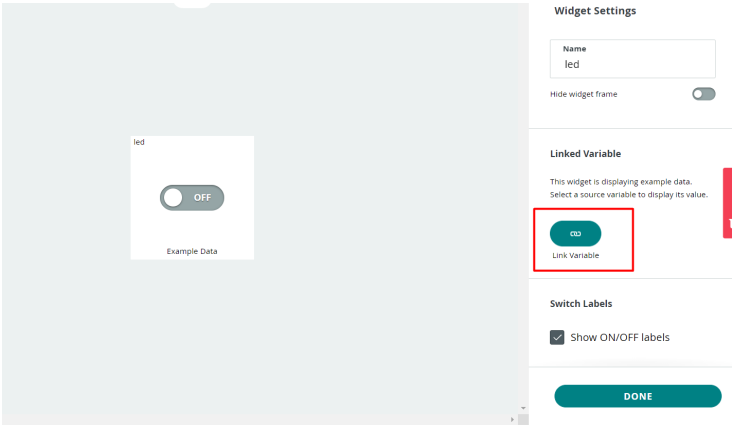
Klik pada ADD dan masukkan switch serta messenger
untuk penampil data boleh berupa gauge, value, atau percentage, pada kasus ini kami mencoba menggunakan Messenger.



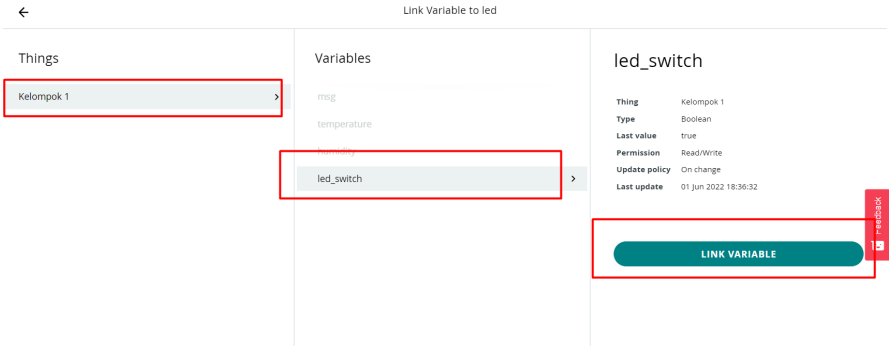
Step 9 : mendefinisikan fungsi switch dan messenger sesuai kasus

Klik titik 3 pada pojok LED > masuk ke edit setting > Link Variable

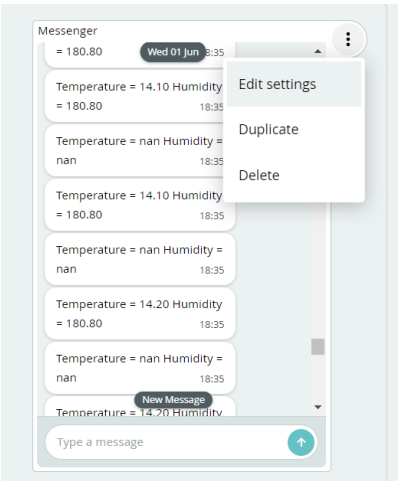


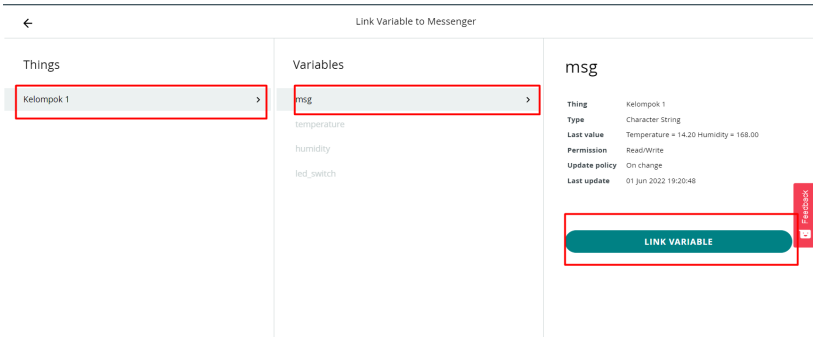


Kemudian arahkan ke led switch

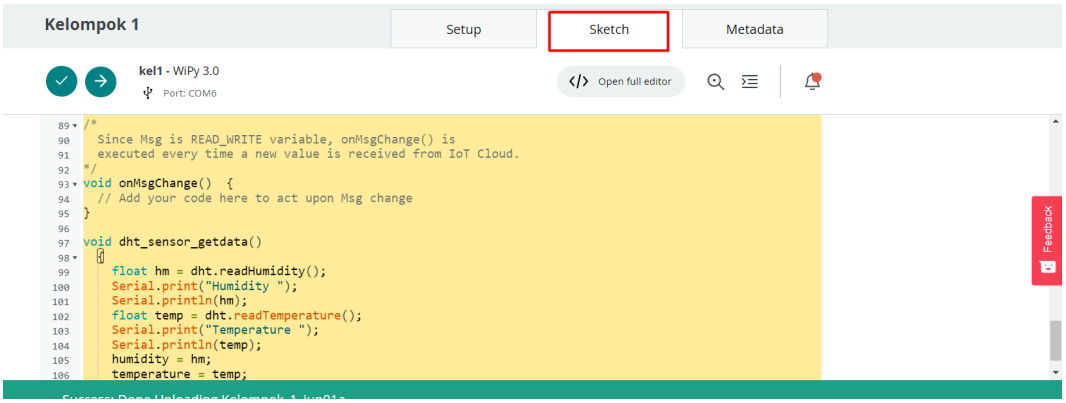


Melakukan Langkah yang sama dengan messenger





Step 10 : Masukkan Kode yang sesuai dengan variabel yang sudah didefinisikan

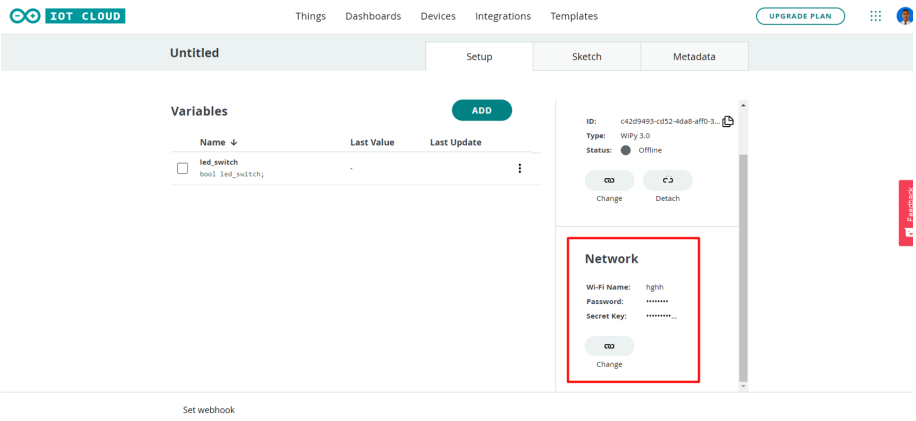



```
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/6a5605af-6d4f-43d7-8c67-a1eb23eab066

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to the Thing

  String msg;
  CloudTemperatureSensor temperature;
  CloudRelativeHumidity humidity;
  bool led_switch;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"
#include "DHT.h"
#define DHTpin 4
```

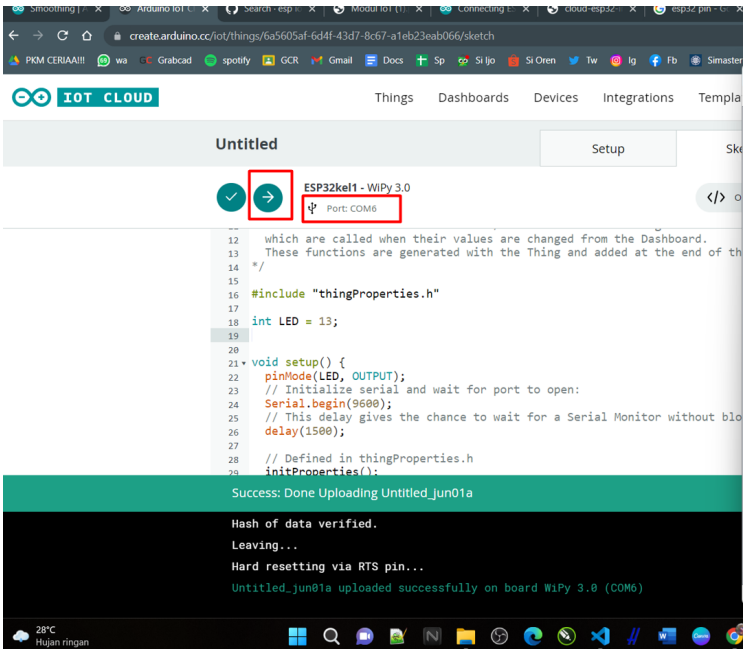
```
#define DHTTYPE DHT11
DHT dht(DHTpin,DHTTYPE);
int LED = 13;

void setup() {

  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if
none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  // Your code here
  dht_sensor_getdata();

  CloudRelativeHumidity humidity;
  String msg;
  CloudTemperatureSensor temperature;
}

/*
```

```
  Since LedSwitch is READ_WRITE variable, onLedSwitchChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onLedSwitchChange()  {
  // Add your code here to act upon LedSwitch change
    if(led_switch){
    digitalWrite(LED, LOW);
  }
  else{
    digitalWrite(LED, HIGH);
  }
}

/*
  Since Humidity is READ_WRITE variable, onHumidityChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onHumidityChange()  {
  // Add your code here to act upon Humidity change
}

/*
  Since Msg is READ_WRITE variable, onMsgChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onMsgChange()  {
  // Add your code here to act upon Msg change
}

void dht_sensor_getdata()
  {
    float hm = dht.readHumidity();
    Serial.print("Humidity ");
    Serial.println(hm);
    float temp = dht.readTemperature();
    Serial.print("Temperature ");
    Serial.println(temp);
    humidity = hm;
    temperature = temp;
    msg="Temperature = " + String (temperature)+"  Humidity = " +
String(humidity);
  }
```

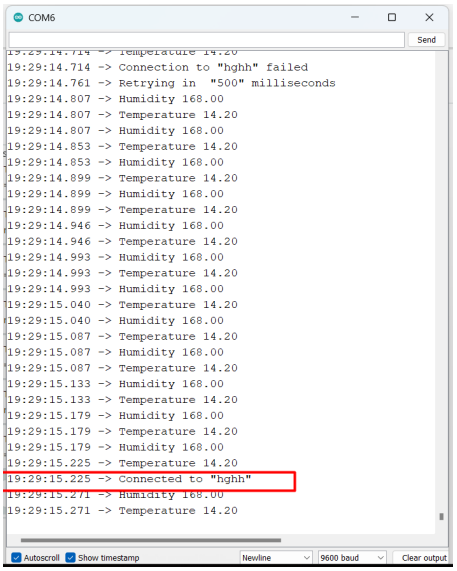
Step 11: Masukkan Secret Key dan SSID wifi + password pada opsi network



Step 12 : Compiling kode, memastikan bahwa esp terdeteksi, dan jika sudah terkoneksi maka bisa klik Upload sampai terdapat notifikasi Succes Done Uploading

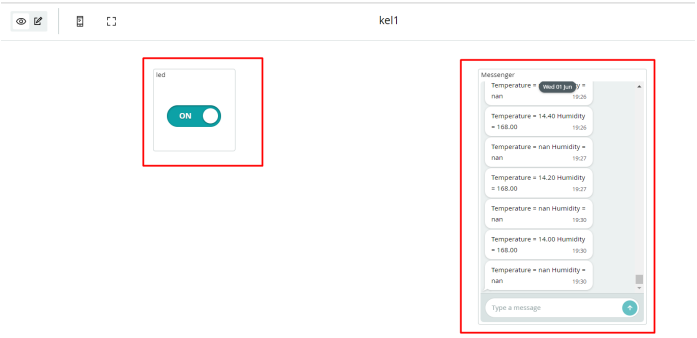


Step 13 : Uji coba, dengan melihat serial monitor dan pastikan wifi terhubung

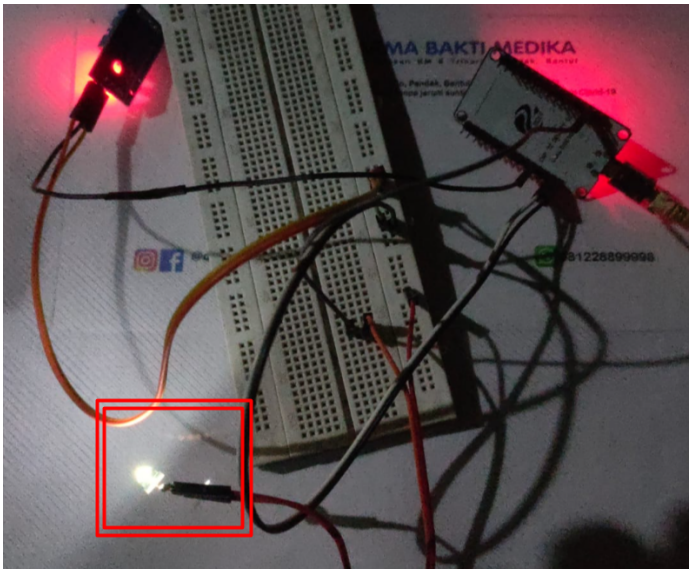


(pembacaan sensor salah dikarenakan sensor DHT11 yang kami punya sudah error)

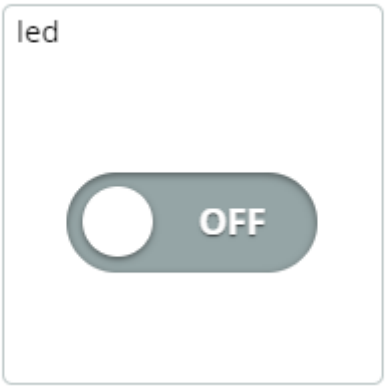
Step 13 : Uji coba, pada monitoring web dan kendali LED

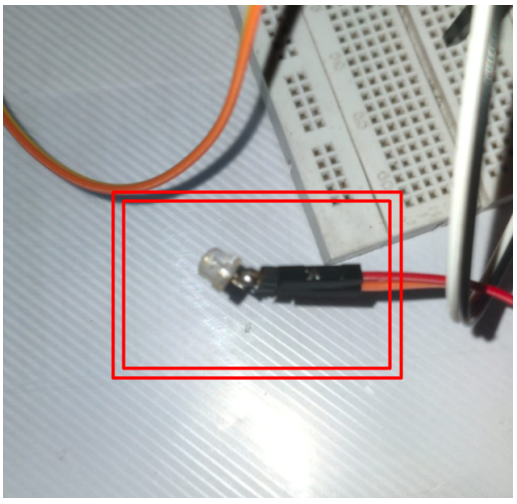


Switch LED kondisi High dan lampu LED menyala

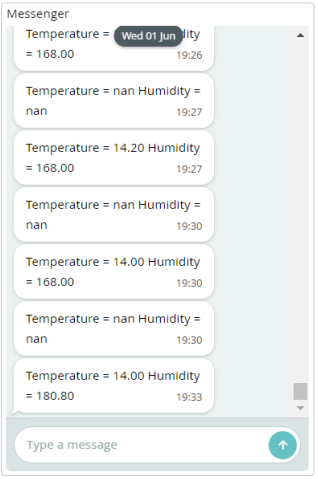


Switch LED LOW LED padam





Data Temperature dan Humidity juga ditampilkan (PEMBACAAN ERROR dikarenakan sensor yang kami miliki rusak)



BAB 3. Kesimpulan

Platform Arduino Cloud IoT dapat digunakan sebagai dashboard untuk membuat sebuah sistem IoT sederhana sebagai pemantauan sensor atau data dari sebuah mikrokontroler dan dapat mengirimkan perintah ke mikrokontroler secara cepat dan praktis. Dalam kasus ini kami berhasil membuat sebuah IoT sederhana pembacaan data DHT11 dan Pengendalian On OFF lampu LED.

Laporan 6. *Device Smart Plug*

BAB 1. Pengenalan Kasus

Membuat sebuah program untuk melakukan pengendalian komponen elektronik IoT Smart Plug dari jarak jauh menggunakan sistem IoT.

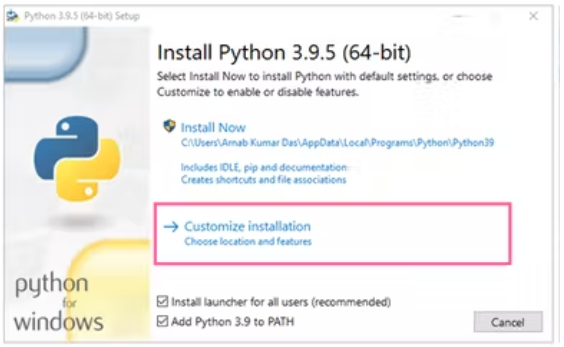
BAB 2. Penyelesaian Kasus

Step 1 : Menginstall Visual Studio Code

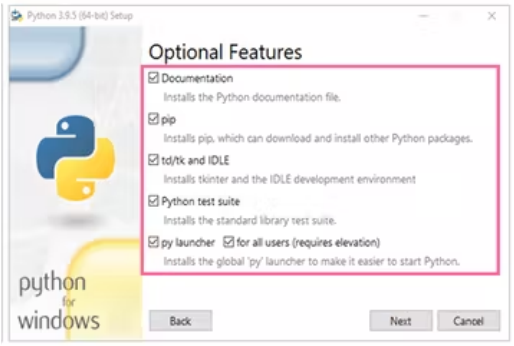
Step 2 : Menginstall Python

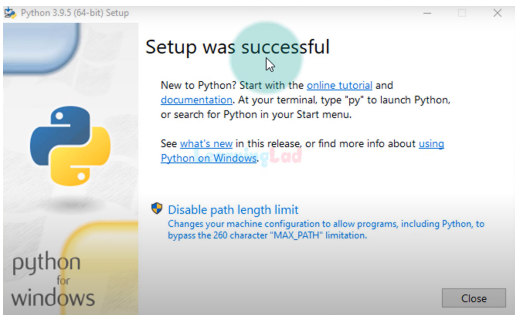
Python 3.9

Pengunduhan python dapat dilakukan melalui :
https://www.python.org/downloads/windows/



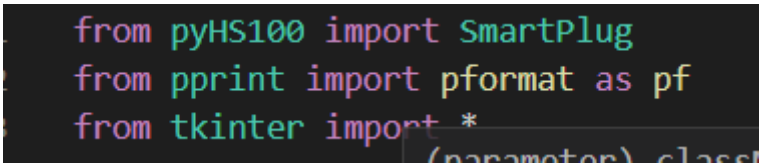
Install Python
Harus dipastikan bahwa PATH harus di ceklis





Ceklis setiap Path

Step 2 : Melakukan install library python yang dibutuhkan



Step 3 : Run Kode dibawah

```
from pyHS100 import SmartPlug
from pprint import pformat as pf
from tkinter import *
```

```
window_main = Tk(className='Smart Plug Control Switch',)
window_main.geometry("400x200")

plug = SmartPlug("192.168.0.1") # data IP dari rotter utama device yang dimana
bisa membaca dari wireless properties bisa diganti sesuai yang terbaca, untuk
manualnya bisa menggunakan code discover device secara terpisah dari thonny
print("Hardware: %s" % pf(plug.hw_info))
print("Full sysinfo: %s" % pf(plug.get_sysinfo())) # this prints lots of information
about the device
print("Current state: %s" % plug.state)
plug.turn_off()
plug.turn_on()


plug.state = "ON"
print("Current state: %s" % plug.state)

print("Current LED state: %s" % plug.led)
plug.led = True # turn off led
print("New LED state: %s" % plug.led)

def plugFunctionON() :
    plug.state = "ON"
def plugFunctionOFF() :
    plug.state = "OFF"
def close():
    window_main.destroy()

myButton1 = Button(window_main, text="Plug ON",
command=plugFunctionON)
myButton1.config(width=20, height=2)
myButton2 = Button(window_main, text="Plug
OFF",command=plugFunctionOFF)
myButton2.config(width=20, height=2)
myButton3 = Button(window_main, text="Exit",command=close)
myButton3.config(width=20, height=2)
myButton1.pack()
myButton2.pack()
myButton3.pack()

window_main.mainloop()
```

Laporan 7. ADC (*Analog to Digital Converter*)

BAB 1. Pengenalan Kasus

Kasus ini bertujuan untuk mengetahui bagaimana cara kerja ADC dari pembacaan sebuah sensor yang kemudian sensor tersebut harus di filter datanya sehingga data yang ditampilkan tidak mengalami perbedaan yang signifikan antara pembacaan 1 dan seterusnya, data harus dikirim ke serial communication dengan perangkat pc dan ditampilkan dalam bentuk plot python GUI.

ADC (Analog to Digital Converter) adalah sebuah rangkaian elektronika yang dapat mengubah besaran analog menjadi besaran digital. Pada setiap sensor yang berbasis mikrokontroler (sebagai pusat pengolah data) diperlukan adanya rangkaian ADC (Analog to Digital Converter) untuk mengubah sinyal yang diterima oleh sensor untuk menjadi besaran digital supaya sinyal tersebut bisa diterjemahkan atau dibaca mikrokontroler. Sensor- sensor disini dapat berupa sensor suhu, sensor level, sensor tekanan, dan lain lain. (Sagita, S.M., dkk 2015)

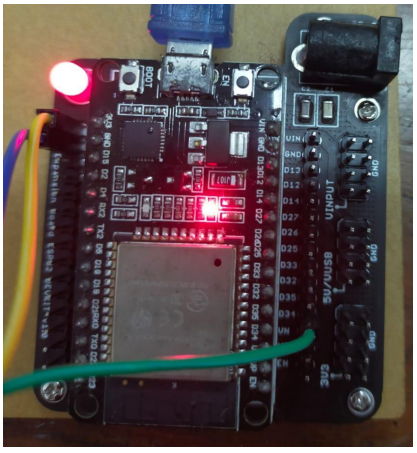
Namun Nilai ADC yang dibaca oleh ESP dari beberapa sensor mengalami perubahan yang lumayan fluktuatif dikarenakan daya yang bekerja pada esp kadang tidak stabil, maka dari itu dibutuhkan teknik Smoothing. Smoothing ADC reading sensor merupakan teknik digunakan untuk memperhalus pembacaan sinyal analog dari berbagai sensor. Pada kasus ini kami menggunakan teknik Smoothing AVERAGE SUM yaitu dengan mengumpulkan pembacaan sensor sampai jumlah X dan dibagi dengan jumlah X yang ditentukan kemudian menampilkannya setelah terkumpul. Dengan langkah smoothing tersebut maka pembacaan sensor tidak akan terlalu fluktuatif perubahannya.

BAB 2. Penyelesaian Kasus

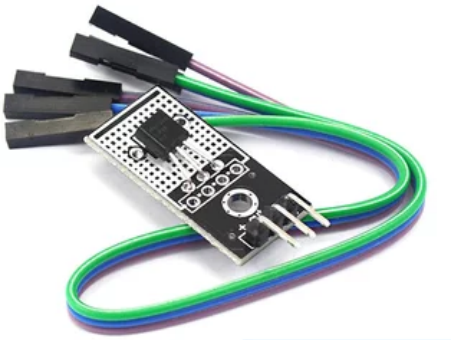
Berikut adalah Step penyelesaian kasus :

Step 1: Menyiapkan material yang dibutuhkan

9. ESP-32

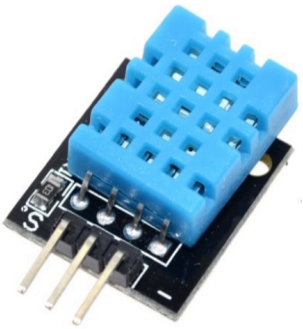


10. Sensor LM35

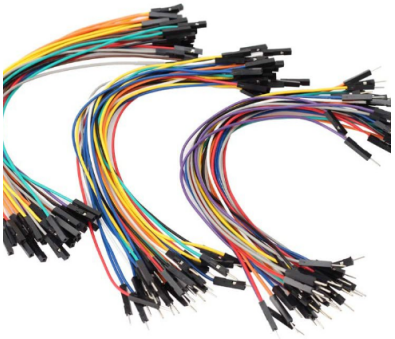


11. Potensio

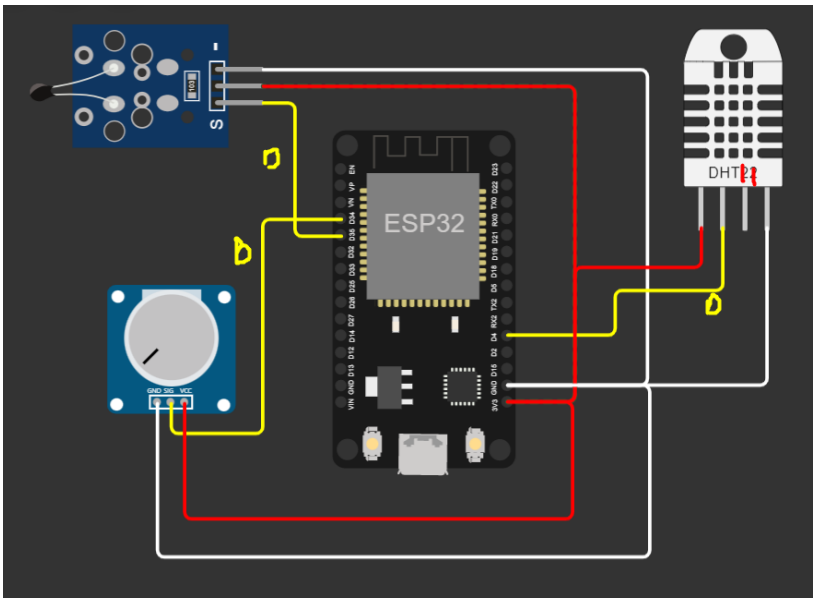
12. DHT 11



13. Jumper Wires

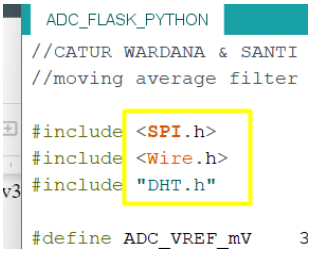


Step 2: SetUp



Data LM35 ke pin 35 ESP32 , VCC=3v3, GND=GND
Data DHT ke pin 4 ESP32, VCC=3v3, GND=GND
Data Potensio ke pin 34 ESP32 VCC=3v3, GND=GND

Step 3: Install Library Arduino



Install library pada Arduino IDE yang dibutuhkan

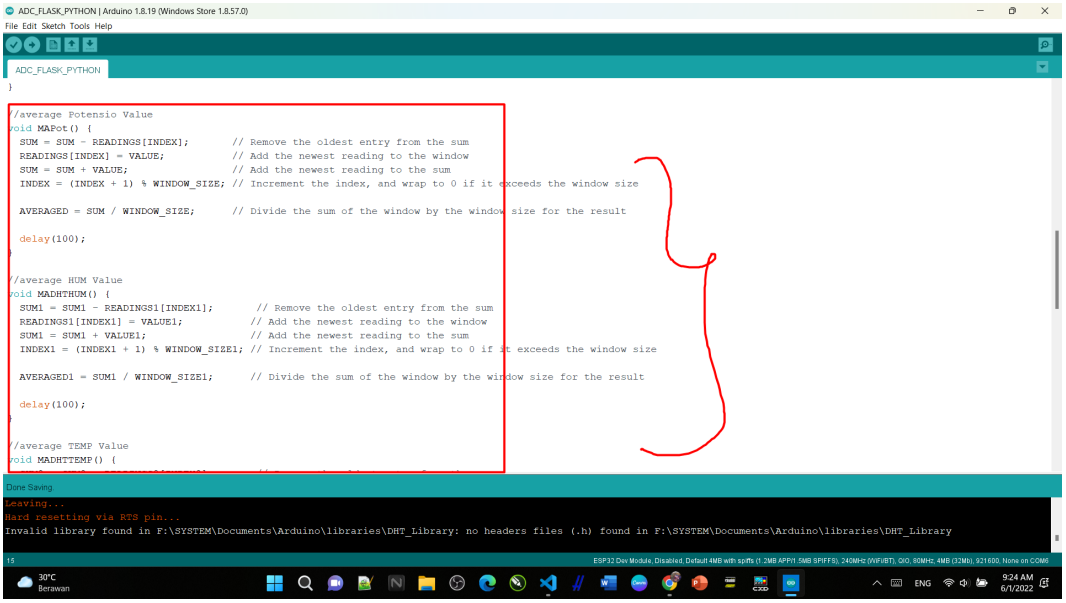
Step 4: Install Library python

```
GUI DHT & POT & LM35_CATUR - SANTI.py > makeFig
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 *
```

Install library pada Python menggunakan CMD

pip install pyserial
pip install drawnow
pip install numpy
pip install matplotlib

Step 5: Kode Arduino



Beberapa Fungsi AVERAGE SUM Smoothing

```
//CATUR WARDANA & SANTI RAHAYU
//moving average filter

#include <SPI.h>
#include <Wire.h>
#include "DHT.h"

#define ADC_VREF_mV    3300.0 // in millivolt
#define ADC_RESOLUTION 4096.0
#define PIN_LM35       35 // ESP32 pin GIOP36 (ADC0) connected to LM35
#define DHTPIN 4
#define DHTTYPE DHT11   // DHT 11
```

```
DHT dht(DHTPIN, DHTTYPE);

const int potPin = 34;
int potValue = 0;

int INDEX = 0;
int INDEX1 = 0;
int INDEX2 = 0;
int INDEX3 = 0;

int VALUE = 0;
int VALUE1 = 0;
int VALUE2 = 0;
int VALUE3 = 0;

int SUM = 0;
int SUM1 = 0;
int SUM2 = 0;
int SUM3 = 0;

const int WINDOW_SIZE = 10;
const int WINDOW_SIZE1 = 10;
const int WINDOW_SIZE2 = 10;
const int WINDOW_SIZE3 = 10;

int READINGS[WINDOW_SIZE];
int READINGS1[WINDOW_SIZE1];
int READINGS2[WINDOW_SIZE2];
int READINGS3[WINDOW_SIZE2];

int AVERAGED = 0;
int AVERAGED1 = 0;
int AVERAGED2 = 0;
int AVERAGED3 = 0;


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

//average Potensio Value
```

```
void MAPot() {
  SUM = SUM - READINGS[INDEX];       // Remove the oldest entry from the
sum
  READINGS[INDEX] = VALUE;           // Add the newest reading to the window
  SUM = SUM + VALUE;                 // Add the newest reading to the sum
  INDEX = (INDEX + 1) % WINDOW_SIZE; // Increment the index, and wrap to
0 if it exceeds the window size

  AVERAGED = SUM / WINDOW_SIZE;      // Divide the sum of the window by
the window size for the result

  delay(100);
}

//average HUM Value
void MADHTHUM() {
  SUM1 = SUM1 - READINGS1[INDEX1];       // Remove the oldest entry from
the sum
  READINGS1[INDEX1] = VALUE1;            // Add the newest reading to the
window
  SUM1 = SUM1 + VALUE1;                  // Add the newest reading to the sum
  INDEX1 = (INDEX1 + 1) % WINDOW_SIZE1; // Increment the index, and
wrap to 0 if it exceeds the window size

  AVERAGED1 = SUM1 / WINDOW_SIZE1;      // Divide the sum of the
window by the window size for the result

  delay(100);
}

//average TEMP Value
void MADHTTEMP() {
  SUM2 = SUM2 - READINGS2[INDEX2];       // Remove the oldest entry from
the sum
  READINGS2[INDEX2] = VALUE2;            // Add the newest reading to the
window
  SUM2 = SUM2 + VALUE2;                  // Add the newest reading to the sum
  INDEX2 = (INDEX2 + 1) % WINDOW_SIZE2; // Increment the index, and
wrap to 0 if it exceeds the window size

  AVERAGED2 = SUM2 / WINDOW_SIZE2;      // Divide the sum of the
window by the window size for the result
```

```
  delay(100);
}

//average TEMP35 Value
void MADHTTEMP35() {
  SUM3 = SUM3 - READINGS3[INDEX3];       // Remove the oldest entry from the sum
  READINGS3[INDEX3] = VALUE3;           // Add the newest reading to the window
  SUM3 = SUM3 + VALUE3;                 // Add the newest reading to the sum
  INDEX3 = (INDEX3 + 1) % WINDOW_SIZE3; // Increment the index, and wrap to 0 if it exceeds the window size

  AVERAGED3 = SUM3 / WINDOW_SIZE3;      // Divide the sum of the window by the window size for the result

  delay(100);
}

void loop() {

  // Read PotValue
  VALUE = analogRead(potPin);
  MAPot();

  // Read HUM
  VALUE1 = dht.readHumidity();
  MADHTHUM();

  // Read temperature as Celsius (the default)
  VALUE2 = dht.readTemperature();
  MADHTTEMP();

  // Read Temp LM35
  int adcVal = analogRead(PIN_LM35); // convert the ADC value to voltage in millivolt
  int milliVolt = adcVal * (ADC_VREF_mV / ADC_RESOLUTION); // convert the voltage to the temperature in °C
  VALUE3 = milliVolt / 100;
  MADHTTEMP35();
```

```
  //send array to serial
  Serial.print(AVERAGED);
  Serial.print(",");
  Serial.print(AVERAGED1);
  Serial.print(",");
  Serial.print(AVERAGED2);
  Serial.print(",");
  Serial.print(AVERAGED3);
  Serial.print(",");

  delay(1000);

}
```

Step 6: Code Python Serial Read & PLOT GUI

```
#code Python GUI PLOT
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=[]
Temp35=[]
plt.ion()
cnt=0

f = plt.figure()
f.set_figwidth(10)
f.set_figheight(7)

def makeFig():

   plt.subplot(2, 2, 1)
   plt.ylim(40,99)
   plt.title('Read Live Data Sensor DHT HUMUDITY')
```

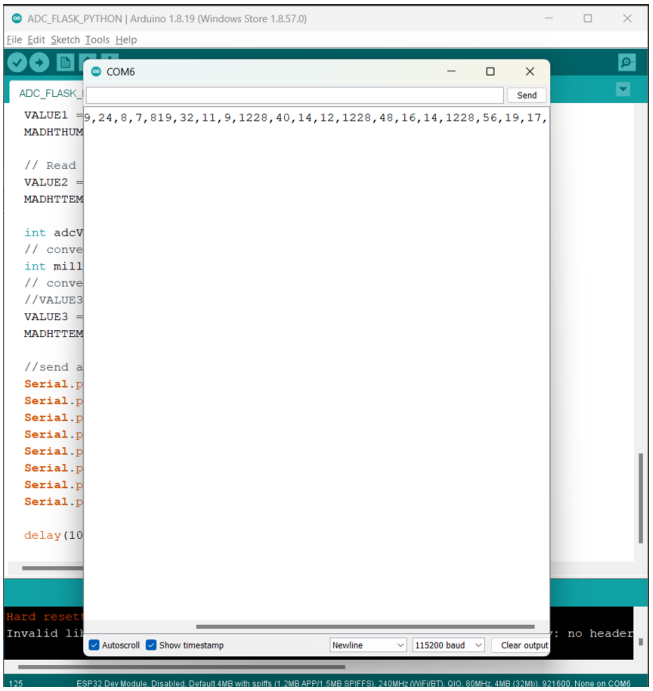
```
    plt.grid(True)
    plt.ylabel('Hum %')
    plt.plot(Hum, 'ro-', label='Persen %')
    plt.legend(loc='upper left')

    plt.subplot(2, 2, 2)
    #plt=plt.twinx()
    plt.title('Read Live Data Sensor DHT TEMPERATURE')
    plt.plot(Temp, 'b^-', label='Temp °C')
    plt.ylabel('Temp C')
    plt.ticklabel_format(useOffset=False)
    plt.legend(loc='upper left')

    plt.subplot(2, 2, 3)
    #plt3=plt.twinx()
    plt.title('Read Live Data ADC POTENSIO')
    plt.plot(Pot, 'g-', label='Potensio ADC')
    plt.ylabel('Potensio ADC')
    plt.ticklabel_format(useOffset=False)
    plt.legend(loc='upper left')

    plt.subplot(2, 2, 4)
    #plt4=plt.twinx()
    plt.title('Read Live Data Sensor LM35 TEMPERATURE')
    plt.plot(Temp35, 'y^-', label='Temp LM °C')
    plt.ylabel('Temp LM C')
    plt.ticklabel_format(useOffset=False)
    plt.legend(loc='upper left')

# make sure the 'COM#' cocok dengan serial yang didgunakan pada esp
ser = serial.Serial('COM6', 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(',')
      P=float(index[0])
      H=float(index[1])
      T=float(index[2])
      T35=float(index[3])

      Pot.append(P)
      Hum.append(H)
      Temp.append(T)
      Temp35.append(T35)

      drawnow(makeFig)
      plt.pause(0.000001)
      cnt=cnt+1
      if(cnt>50):
          Pot.pop(0)
          Hum.pop(0)
          Temp.pop(0)
          Temp35.pop(0)
```

Step 6: Monitor it

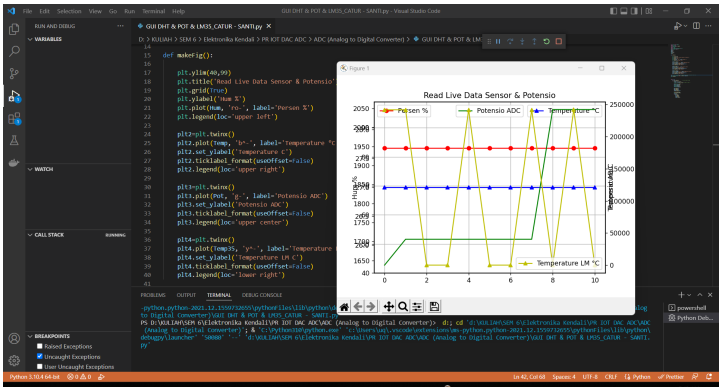
Arduino Array send - data dikirimkan setiap 1100 milisecond dengan pemisah Koma (,)

```
//send array to serial
Serial.print(AVERAGED);
Serial.print(",");
Serial.print(AVERAGED1);
Serial.print(",");
Serial.print(AVERAGED2);
Serial.print(",");
Serial.print(AVERAGED3);
Serial.print(",");

delay(1000);
```

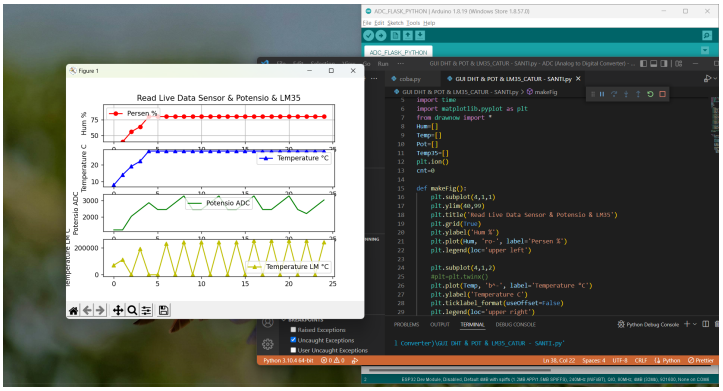


Pyhton GUI PLOT

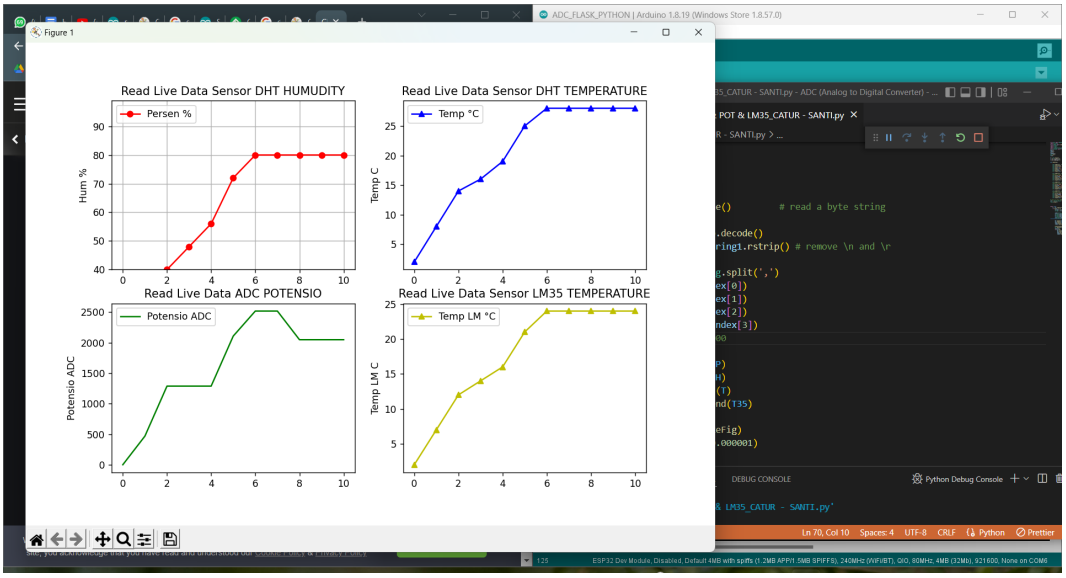
kami mencoba beberapa percobaan penampil



Tampilan 1



Tampilan 2



Tampilan 3 adalah tampilan terbaik

BAB 3. Kesimpulan

Monitoring data sensor Temperatur DHT11, Humidity HDT11, Temperatur LM35, dan ADC Value Potensio sukses dilakukan dengan melakukan smoothing reading menggunakan metode AVERAGE SUM.

Selain itu data berhasil dikirimkan pada serial communication dan ditampilkan secara realtime menggunakan library **drawnow** dan di Plot menggunakan library **Matplotlib**

Daftar Pustaka

Sagita, S.M., Khotijah, S. dan AMALIA, R., 2015. Pengkonversian Data Analog Menjadi Data Digital Dan Data Digital Menjadi Data Analog Menggunakan Interface PPI 8255 Dengan Bahasa Pemrograman Borland Delphi 5.0. Faktor Exacta, 6(2), pp.168-179.

Laporan 8. DAC (*Digital To Analog Converter*)

I. Pengertian DAC

Komputer untuk dapat berhubungan dengan perangkat luar membutuhkan penghubung atau perantara, sehingga dapat dimengerti oleh komputer. Perangkat luar tersebut dapat berupa pengendalian, penerimaan atau pengiriman data.

Sebuah digital to analog converter adalah perangkat elektronika yang digunakan untuk mengkonversi kode digital menjadi sinyal analog. Konversi digital ke analog merupakan cara utama bagi peralatan digital seperti sistem berbasis komputer yang mampu menerjemahkan data digital menjadi sinyal dunia nyata yang lebih dimengerti atau bisa digunakan oleh manusia. DAC dirancang

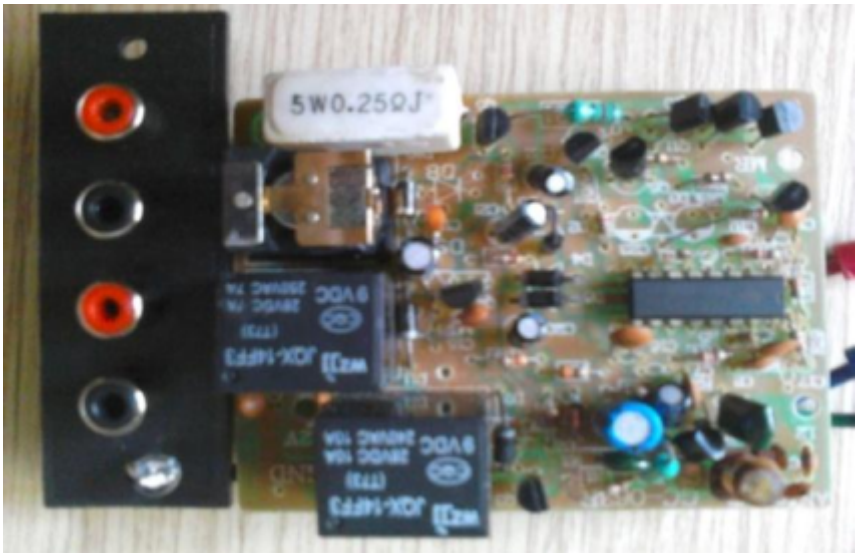
untuk menerima data input digital dalam bentuk serial (satu bit pada satu waktu) sehingga ini hanya memiliki pin input digital tunggal (Wibowo, S.A., Pramono, dkk., 2012).

Menurut Sagita, S.M., Khotijah, S. dan Amalia, R., (2015) Digital to Analog Converter (DAC) dapat dibangun menggunakan penguat penjumlah inverting dari sebuah operasional amplifier (Op-Amp) yang diberikan sinyal input berupa data logika digital (0 dan 1).

II. Prinsip Kerja DAC

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 komputer.

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



Gambar 2.1 Rangkain DAC

Prinsip kerja utama dari alat ini adalah mengubah isyarat data digital (kode-kode biner) menjadi isyarat analog (tegangan analog) sesuai dengan isyarat harga digital tersebut. Untuk menyatukan data analog tersebut diproses pada mixer, sehingga data analog tersebut menjadi data analog yang akurat. Karena transmitter hanya bisa mengirim data digital maka, data analog diubah menjadi digital pada analog digital converter (ADC). Dengan bantuan rangkaian pulsa yang menggunakan IC 555, data digital diperkuat sinyalnya sehingga saat akan dikirim lewat transmitter akan mudah terkirim (Tohari, T., 2015).

Daftar Pustaka

Sagita, S.M., Khotijah, S. dan AMALIA, R., 2015. Pengkonversian Data Analog Menjadi Data Digital Dan Data Digital Menjadi Data Analog Menggunakan Interface PPI 8255 Dengan Bahasa Pemrograman Borland Delphi 5.0. Faktor Exacta, 6(2), pp.168-179.

Tohari, T., 2015. Fungsi Digital Analog Converter Pada Sistem Peringatan Dini Pengendalian Banjir Dengan Elektronik Data Proses. Power Elektronik: Jurnal Orang Elektro, 3(1).

Wibowo, S.A., Pramono, S.H., Julius, M. dan Wijono, W., 2012. Desain 8 Bit R3R Ladder Digital To Analog Converter. Jurnal EECCIS, 6(2), pp.131-138.

Daftar Pustaka

Fransisko, P., 2019. Media Promosi Elektronik untuk Produk-Produk di Supermarket Menggunakan Arduino Nano. Jurnal Sistem Cerdas dan Rekayasa (JSCR), 1(1).

Kusumah, H. dan Pradana, R.A., 2019. Penerapan Trainer Interfacing Mikrokontroler Dan Internet of Things Berbasis Esp32 Pada Mata Kuliah Interfacing. Journal Cerita, 5(2), pp.120-134.

ThingSpeak for IoT Projects. https://thingspeak.com/. DIakses tanggal 25 Mei 2022.

Syadza, Q., Permana, A.G. dan Ramadan, D.N., 2018. Pengontrolan Dan Monitoring Prototype Green House Menggunakan Mikrokontroler Dan Firebase. Proceedings of Applied Science, 4(1).

Wibowo, S.A., Pramono, S.H., Julius, M. dan Wijono, W., 2012. Desain 8 Bit R3R Ladder Digital To Analog Converter. Jurnal EECCIS, 6(2), pp.131-138.

Sagita, S.M., Khotijah, S. dan AMALIA, R., 2015. Pengkonversian Data Analog Menjadi Data Digital Dan Data Digital Menjadi Data Analog Menggunakan Interface PPI 8255 Dengan Bahasa Pemrograman Borland Delphi 5.0. Faktor Exacta, 6(2), pp.168-179.

Tohari, T., 2015. Fungsi Digital Analog Converter Pada Sistem Peringatan Dini Pengendalian Banjir Dengan Elektronik Data Proses. Power Elektronik: Jurnal Orang Elektro, 3(1).