Universitas Semarang
Fakultas Teknologi Informasi dan Komunikasi
Teknik Informatika

Flow Programming IoT dengan MicroPython dan Node-RED

Modul Pelatihan SKPI Mahasiswa

Oleh:
Alauddin Maulana Hirzan, S. Kom., M. Kom
NIDN. 0607069401 | NIS. 06557003102238

Daftar Isi

Daftar Gambar

Pendahuluan

0.1 Internet of Things (IoT)

Internet of Things (IoT) mengacu pada jaringan objek sehari-hari yang terhubung ke internet, memungkinkan mereka untuk mengumpulkan, mengirim, dan menerima data. Benda-benda ini, yang sering disebut "perangkat pintar", dapat mencakup apa saja, mulai dari peralatan rumah tangga dan mobil hingga mesin industri dan perangkat medis. Melalui sensor dan perangkat lunak, perangkat IoT dapat memantau lingkungannya, berinteraksi satu sama lain, dan dikendalikan dari jarak jauh oleh pengguna. Koneksi ini membantu mengotomatiskan proses, memberikan wawasan yang berguna, dan dapat membuat hidup lebih nyaman, seperti dengan memungkinkan orang untuk mengontrol lampu dengan smartphone atau memantau kesehatan mereka secara real-time.

0.2 Wokwi

Wokwi adalah simulator elektronik online yang memungkinkan pengguna untuk mensimulasikan berbagai mikrokontroler seperti Arduino, ESP32, dan Raspberry Pi Pico, sehingga mereka dapat menguji dan mengembangkan proyek mereka secara virtual. Platform ini dilengkapi dengan alat untuk simulasi WiFi, debugging, dan perpustakaan besar komponen elektronik.

0.3 MicroPython

MicroPython adalah implementasi ringan dari bahasa pemrograman Python yang dirancang untuk berjalan pada mikrokontroler dan lingkungan dengan sumber daya terbatas. Ini memungkinkan pengembang untuk menulis kode Python untuk mengontrol perangkat keras, sehingga memudahkan pengembangan sistem tertanam dan aplikasi IoT. MicroPython mencakup subset dari perpustakaan standar Python dan dioptimalkan untuk kinerja dan penggunaan memori, sehingga dapat berjalan pada perangkat dengan sumber daya terbatas. MicroPython mendukung berbagai platform mikrokontroler, termasuk ESP8266, ESP32, dan Raspberry Pi Pico, di antara lainnya.

0.4 Flow Programming

Flow Programming adalah pendekatan visual untuk pengkodean, di mana pengguna menghubungkan blok atau "node" untuk membuat urutan operasi yang dikenal sebagai "aliran". Alih-alih pengkodean baris per baris tradisional, pengguna menarik dan melepas

node yang mewakili fungsi yang berbeda (seperti input, output, dan proses) ke kanvas dan menghubungkannya dengan kabel virtual, sehingga menciptakan jalur yang mudah diikuti untuk langkah-langkah pemrosesan data.

1. **Node**:
   Blok bangunan dalam Node-RED. Setiap node memiliki fungsi tertentu, seperti membaca input dari sensor, memproses data, atau mengirim pesan. Ada berbagai jenis node, seperti node input (untuk mengumpulkan data), node output (untuk mengirim data), dan node fungsi (untuk memproses atau mengubah data).

2. **Flow/Alir**:
   Urutan node yang terhubung yang mendefinisikan bagaimana data bergerak melalui program dan apa yang terjadi pada setiap langkah. Alur diatur dengan menghubungkan node, menunjukkan jalur data yang jelas dan memungkinkan proses yang kompleks dipecah menjadi bagian-bagian yang dapat dikelola dan saling berhubungan.

3. **Message/Pesan**:
   Data ditransmisikan melalui node dalam aliran sebagai pesan, sering kali dalam format JSON. Setiap pesan berisi informasi yang berpindah dari satu node ke node berikutnya. Pesan dapat dimanipulasi saat melewati node fungsi, menciptakan peluang untuk menyaring, memformat, atau menganalisis data secara real-time.

4. **Debugging dan Pemantauan**:
   Node-RED menyertakan panel debug, di mana pengguna dapat memantau output pada titik mana pun dalam aliran, sehingga memudahkan untuk memecahkan masalah dan menyempurnakan proses.

0.5 Node-RED

Node-RED adalah alat pengembangan berbasis alur kerja dengan kode minimal untuk pemrograman visual, yang awalnya dikembangkan oleh IBM untuk menghubungkan perangkat keras, antarmuka pemrograman aplikasi (API), dan layanan online sebagai bagian dari Internet of Things (IoT).

0.6 Message Queueing Telemetry Transport

MQTT adalah protokol perpesanan ringan yang dirancang untuk komunikasi antar perangkat di lingkungan jaringan dengan bandwidth rendah, latensi tinggi, atau tidak dapat diandalkan, yang membuatnya ideal untuk aplikasi Internet of Things (IoT). Protokol ini beroperasi dengan model publish-subscribe:

1. **Model Publish-Subscribe**:
   Perangkat (atau klien) tidak berkomunikasi secara langsung. Sebaliknya, mereka terhubung ke broker pusat, yang mengelola distribusi pesan. Perangkat yang mengirim data menerbitkannya ke topik tertentu. Perangkat lain yang membutuhkan data ini berlangganan ke topik yang relevan. Broker merutekan pesan dari penerbit ke pelanggan.

2. **Topik**:
   Topik adalah kategori atau saluran yang mengatur pesan. Setiap pesan diberi la-

bel dengan sebuah topik, sehingga pelanggan dapat menyaring pesan dan hanya menerima informasi yang mereka butuhkan. Topik memiliki hirarki, sehingga memungkinkan penargetan yang spesifik (misalnya, rumah/ruang tamu/suhu).

3. **QoS (Kualitas Layanan)**:
   MQTT menawarkan tiga tingkat pengiriman pesan: Paling banyak sekali, Paling sedikit sekali, dan Tepat sekali, yang memungkinkan pengguna untuk mengontrol keandalan pengiriman pesan berdasarkan persyaratan aplikasi.

Persiapan Pelatihan

Agar pelatihan dapat berjalan dengan lancar, mahasiswa diwajibkan memenuhi persyaratan berikut baik dalam bentuk perangkat keras maupun lunak:

0.7 Perangkat Keras

Pelatihan SKPI ini hanya memerlukan Browser dan aplikasi Node-RED, sehingga tidak dibutuhkan spesifikasi yang tinggi.

- Prosesor 4 Inti
- Memori RAM 4GB
- Harddisk 500GB
- Keyboard
- Mouse

0.8 Perangkat Lunak

Perangkat lunak berikut ini wajib diinstall oleh mahasiswa demi lancarnya praktikum:

- Web Browser
- NodeJS **Windows Installer (.msi)**(Download)

Bab 1

Wokwi #1 - ESP32 dan Komponen Sensor

1.1 Mengenai Wokwi #1 - ESP32 dan Komponen Sensor

Di bagian ini mahasiswa diajarkan bagaimana menggunakan Wokwi untuk mensimulasikan Internet of Things dengan ESP32 dan bahasa pemrograman MicroPython

1.2 Perangkat Lunak Dibutuhkan

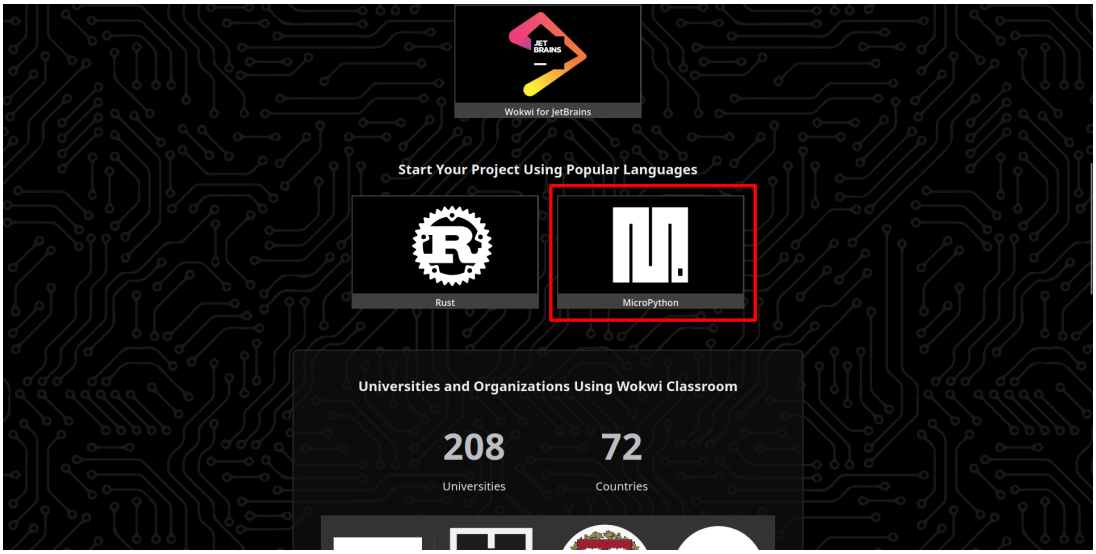
1. Web Browser

1.3 Tutorial

1. Buka Web Browser lalu navigasikan ke URL : https://wokwi.com

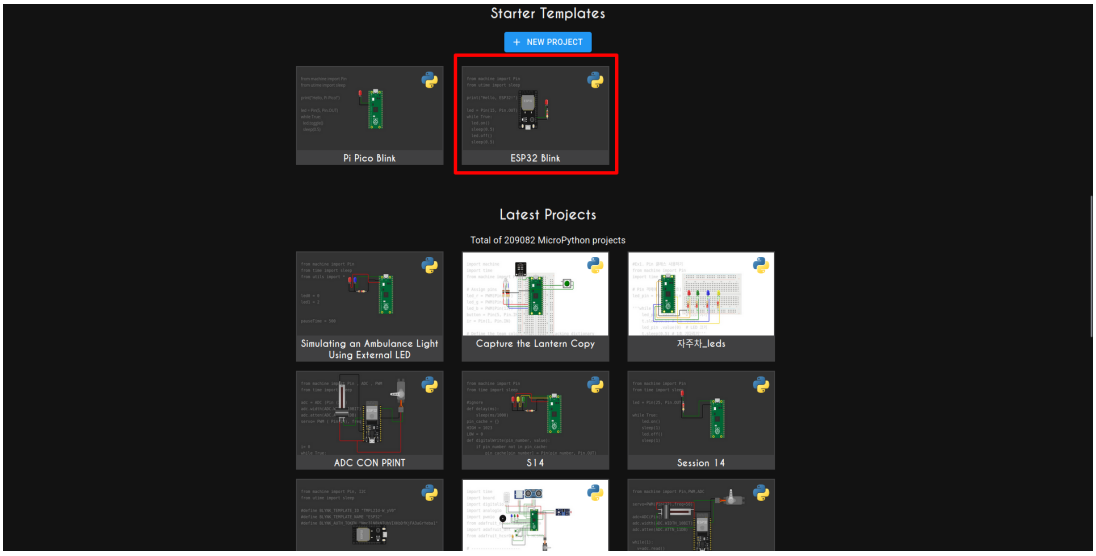
Gambar 1.1: Halaman Web Wokwi

2. (**WAJIB**) Registrasikan akun agar projek tidak hilang
3. Scroll turun dan cari ***Start Your Project Using Popular Languages***. Lalu klik **MicroPython**



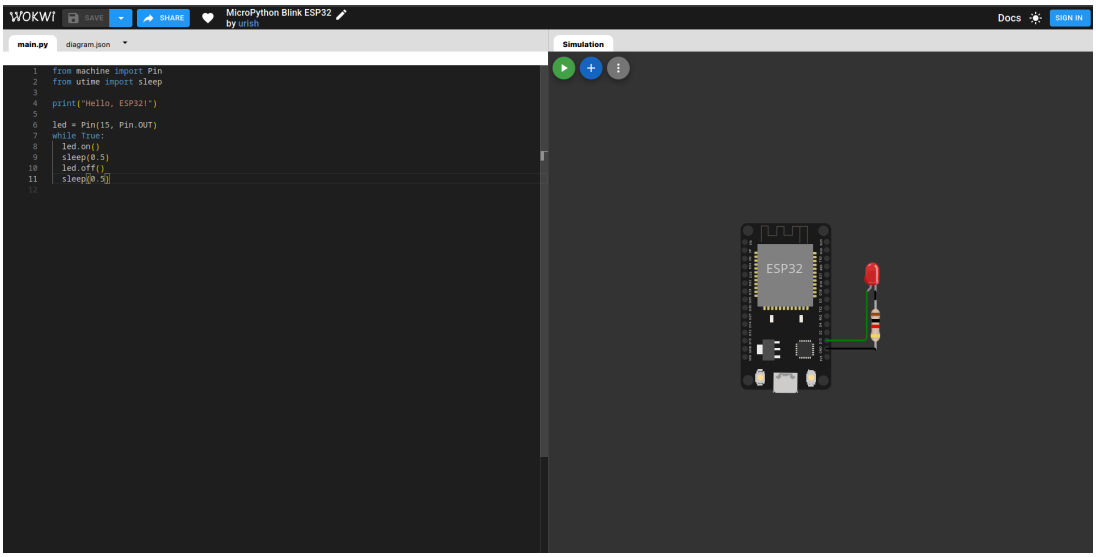
Gambar 1.2: Memulai Projek MicroPython

4. Cari ***Starter Templates***, lalu klik **ESP32 Blink**



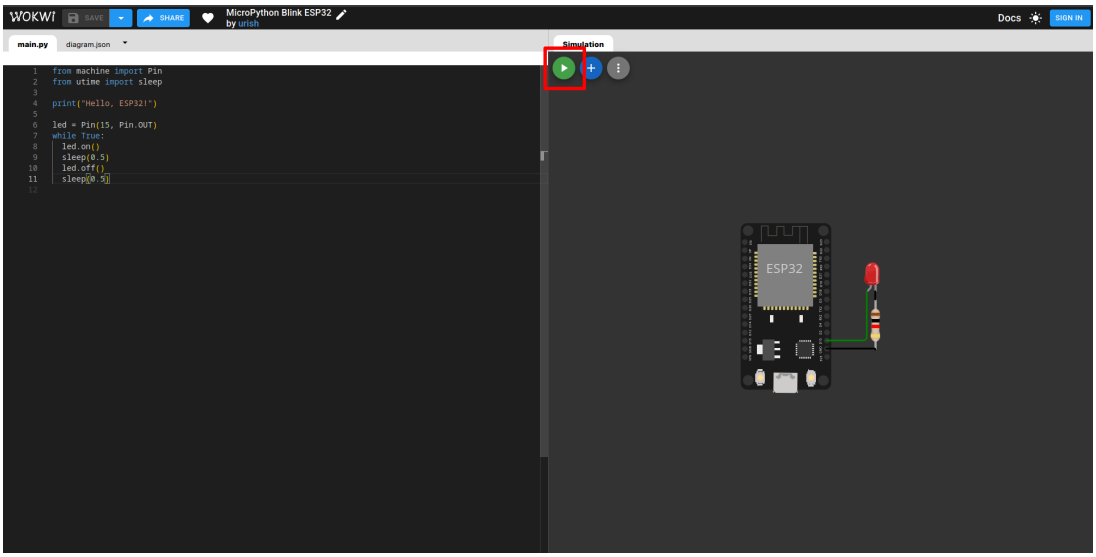
Gambar 1.3: Memilih ESP32 Blink

5. Ketika selesai dibuka, Wokwi akan menampilkan ESP32 beserta LED yang sudah terhubung.



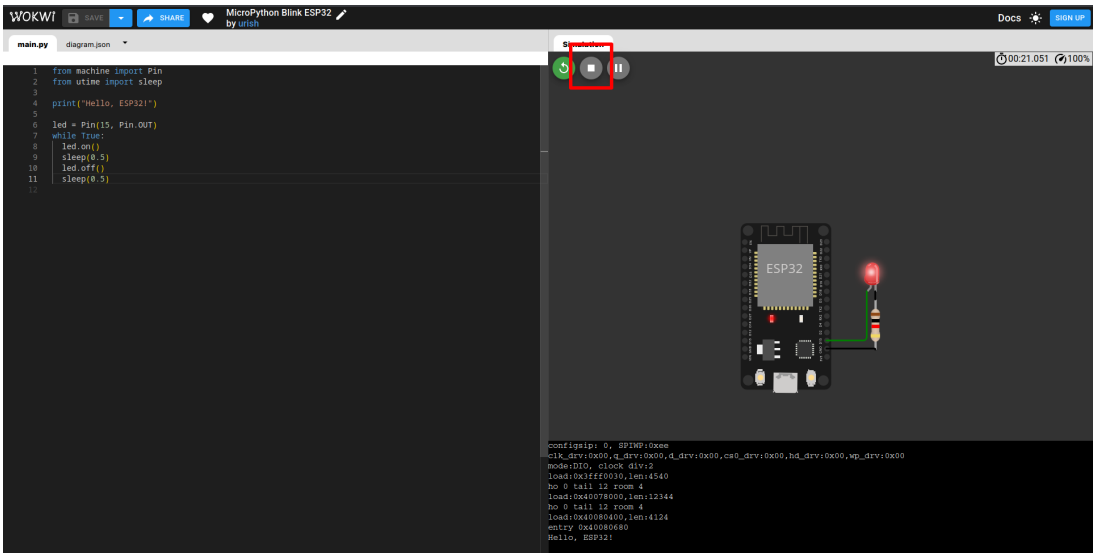
Gambar 1.4: Tampilan ESP32 dan LED

6. Uji ESP32 dengan klik tombol **Run** warna hijau di panel kanan



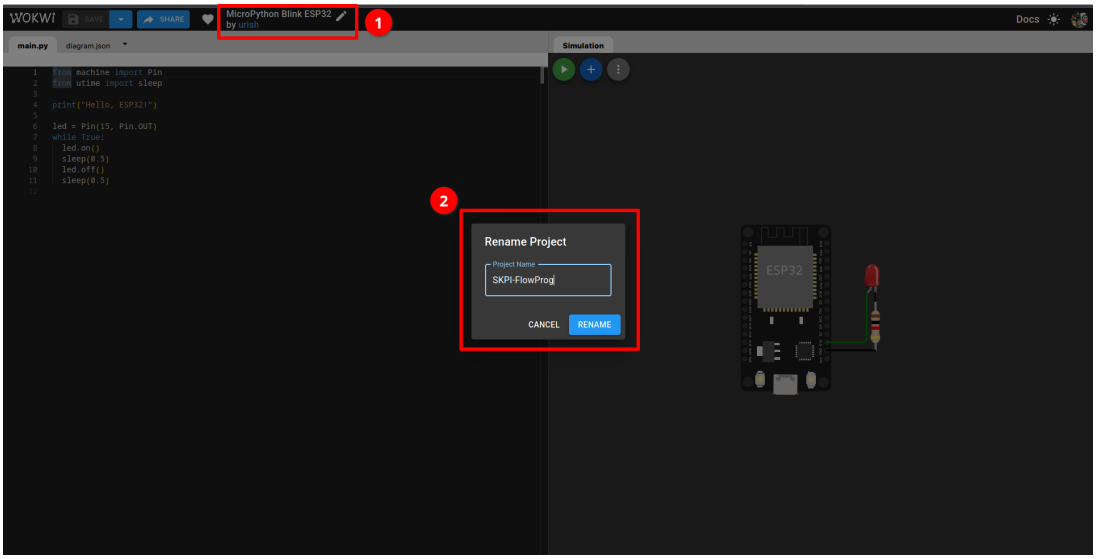
Gambar 1.5: Menguji ESP32 dan LED

7. LED akan berkedip tanda menyala. Jika berhasil, klik **Stop**



Gambar 1.6: ESP32 dan LED Berhasil

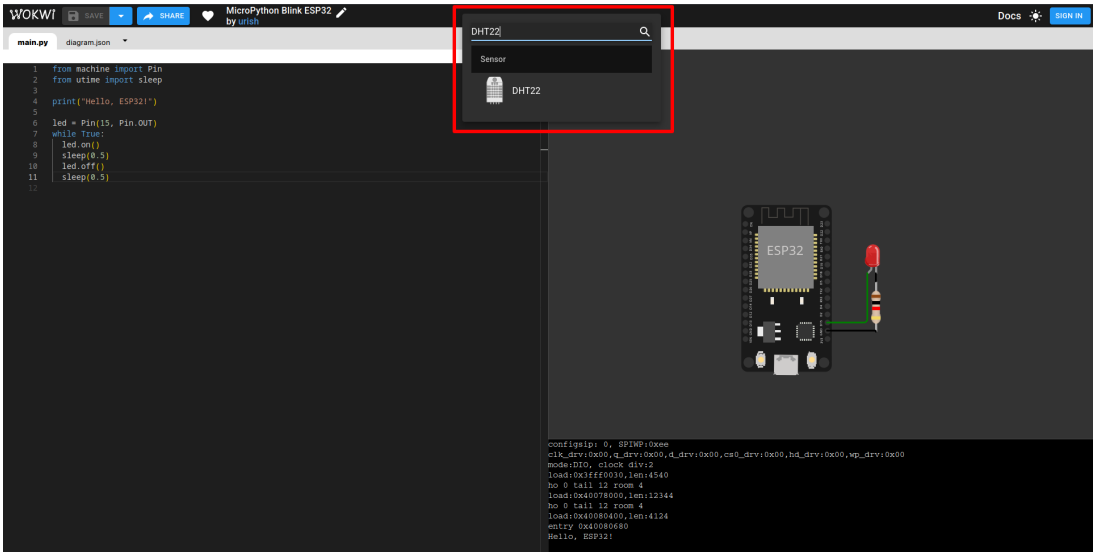
8. Ubah nama projek dengan **SKPI-FlowProg** di bagian atas dengan cara klik **Pensil** di samping tulisan **MicroPython Blink ESP32**



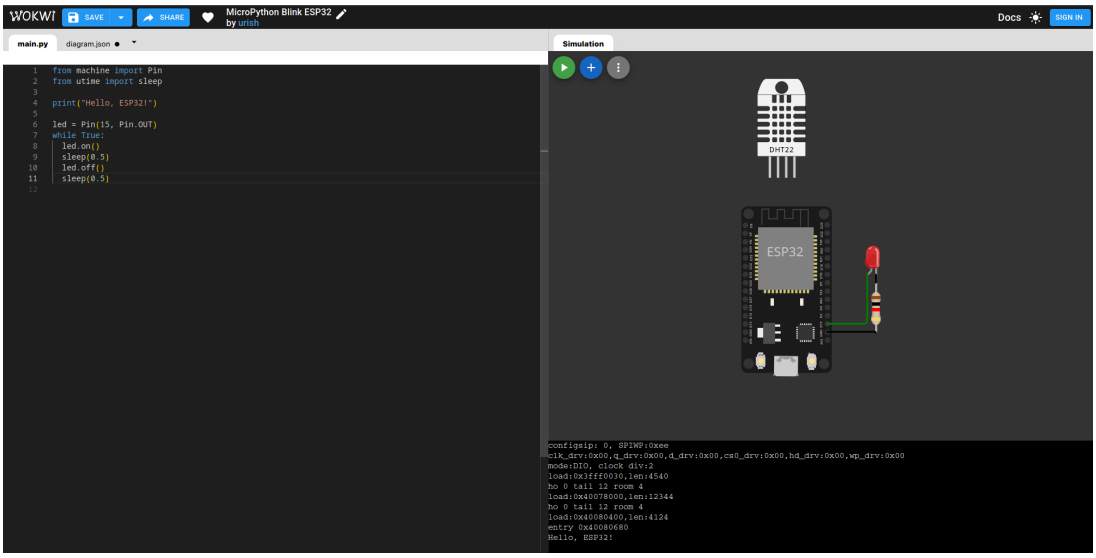
Gambar 1.7: Simpan Projek

9. Berikutnya adalah menambahkan sensor **DHT22**. Klik tombol **Plus** warna biru (akan muncul jika sudah stop), dan pilih **DHT22**

Gambar 1.8: (1) Menambahkan Sensor DHT22

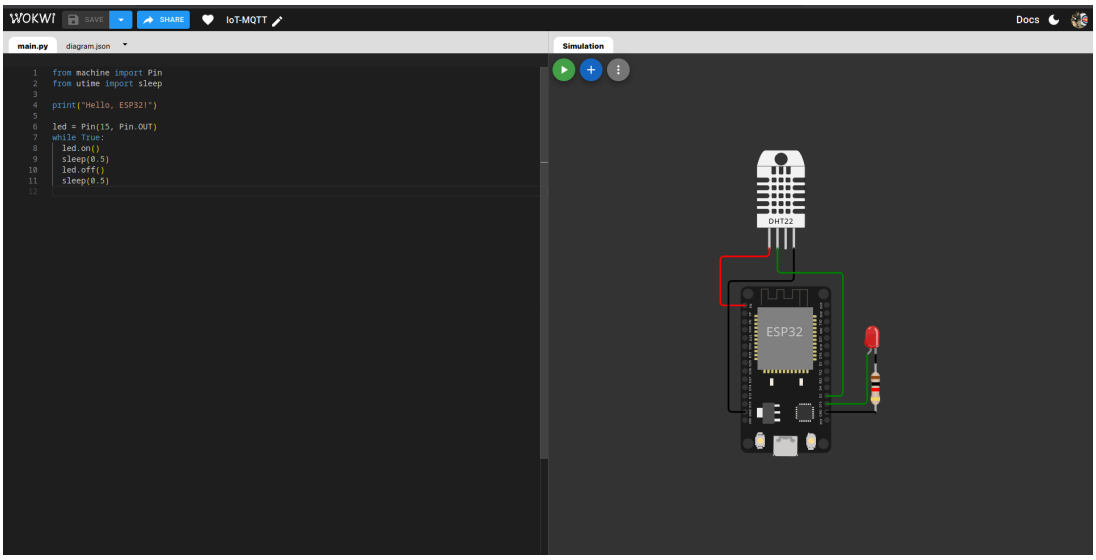


Gambar 1.9: (2) Menambahkan Sensor DHT22



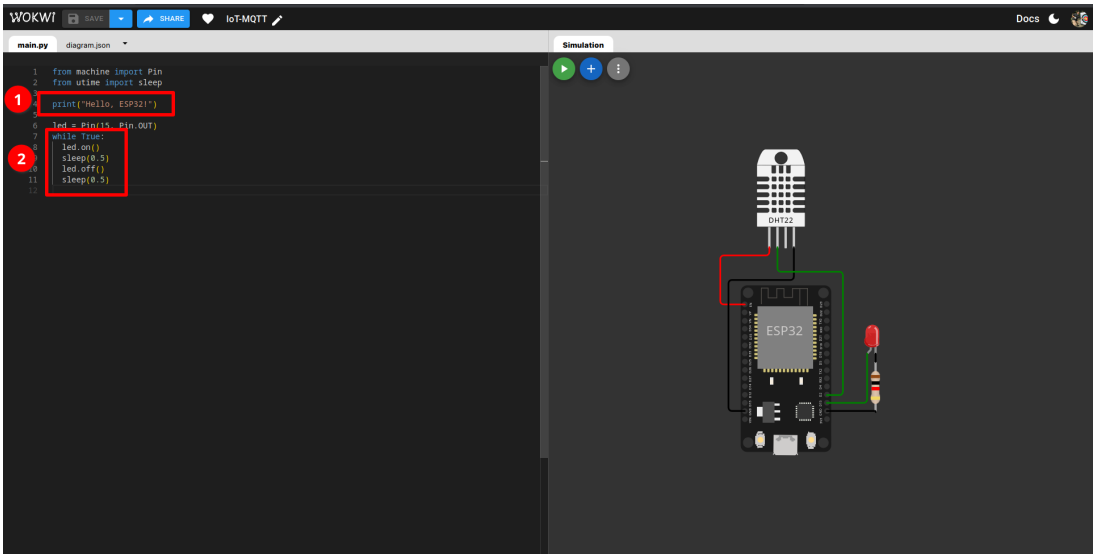
Gambar 1.10: (3) Menambahkan Sensor DHT22

10. Hubungkan DHT22 ke ESP32 dengan konfigurasi berikut:
    - **dht1: VCC → esp: EN**
    - **dht1: SDA → esp: D2**
    - **dht1: GND → esp: GND.2**



Gambar 1.11: Rangkaian DHT22 dan ESP32

11. Untuk membuat sensor bisa terkoneksi dan terbaca oleh ESP32, kode MicroPython perlu dimasukkan. Sebelumnya hapus bagian berikut:

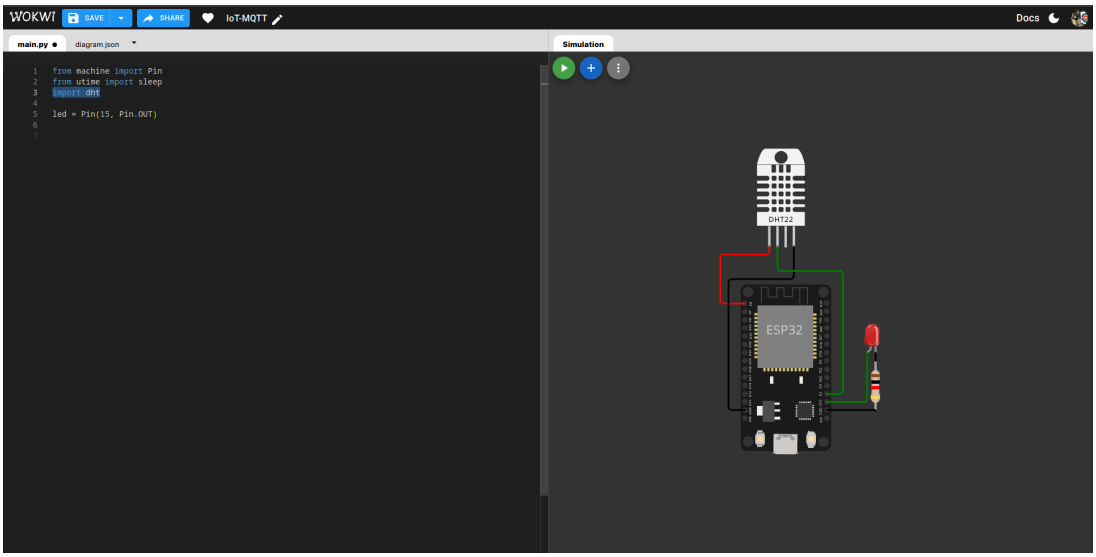


Gambar 1.12: Hapus Kode

12. Masukkan kode berikut setelah **from utime import sleep**. Python menggunakan **4 spasi** dan bukan **Tab**

Potongan Kode

```
import dht
```

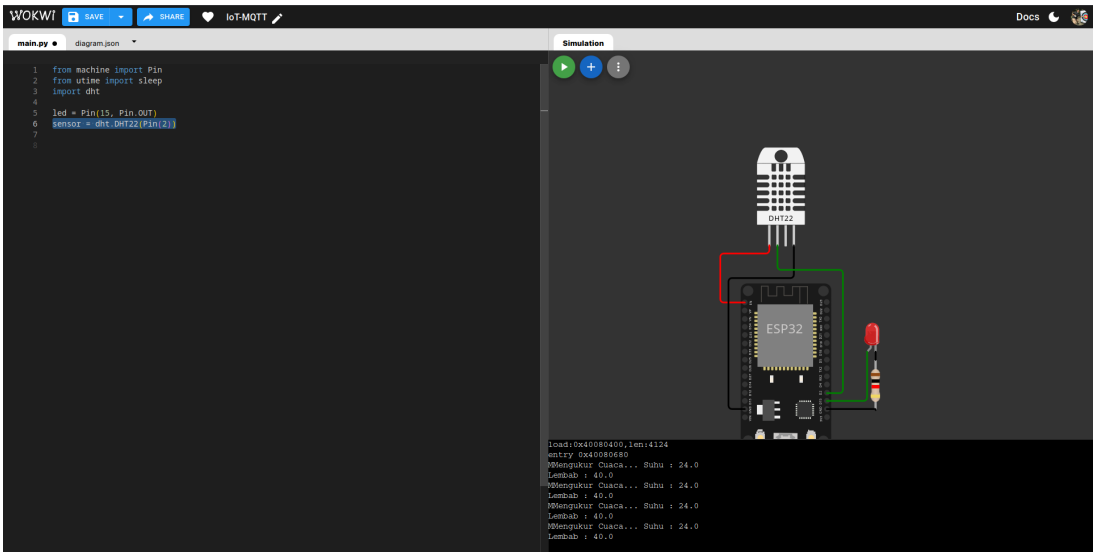


Gambar 1.13: Import Library DHT22

13. Setelah itu inisialisasi sensor DHT22 dengan kode berikut, letakkan setelah **led = Pin**

Potongan Kode

```
sensor = dht.DHT22(Pin(2))
```

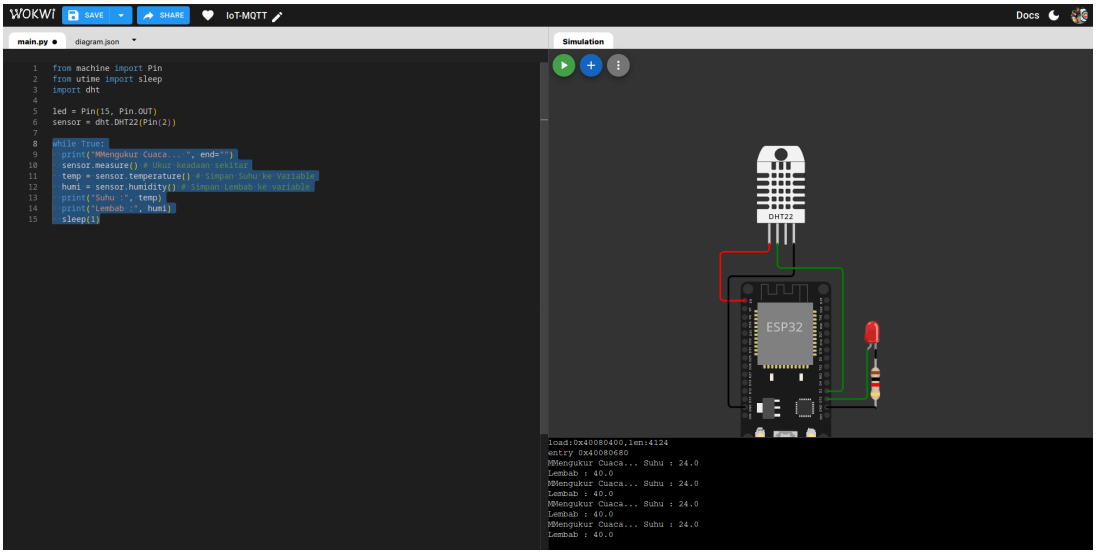


Gambar 1.14: Inisialisasi Sensor DHT22

14. Untuk membaca sensor, masukkan kode berikut setelah **sensor = dht.DHT22(Pin(2))** (Peringatan: Gunakan 4 spasi):

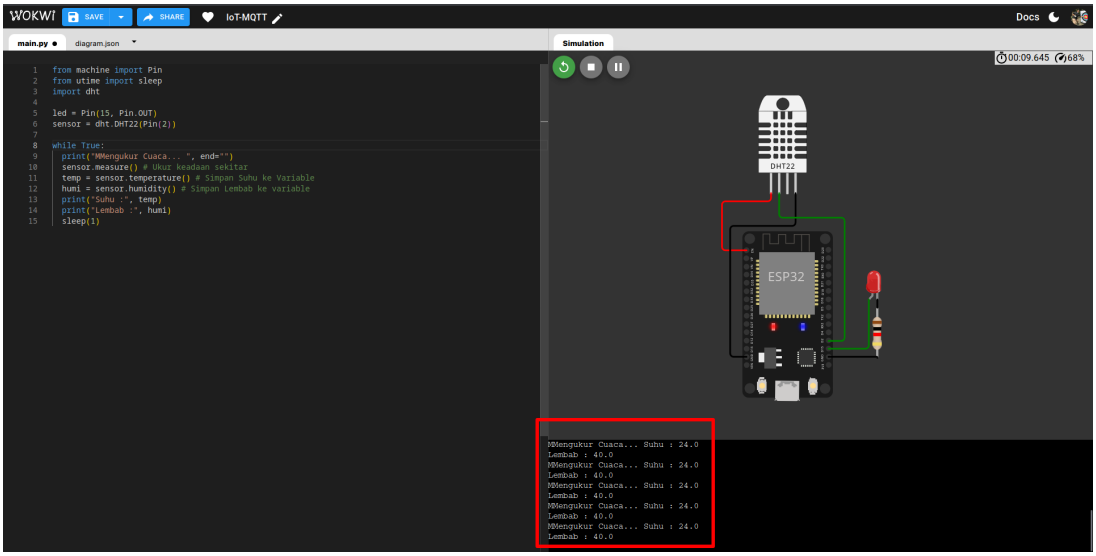
Potongan Kode

```
while True:
    print("Mengukur Cuaca... ")
    sensor.measure() # Ukur keadaan sekitar
    temp = sensor.temperature() # Simpan Suhu ke Variable
    humi = sensor.humidity() # Simpan Lembab ke variable
    print("Suhu :", temp)
    print("Lembab :", humi)
    sleep(1)
```



Gambar 1.15: Kode Pembaca Sensor

15. Untuk menguji apakah sensor sudah terbaca dengan baik, maka klik **Run** dan perhatikan Log di bawah

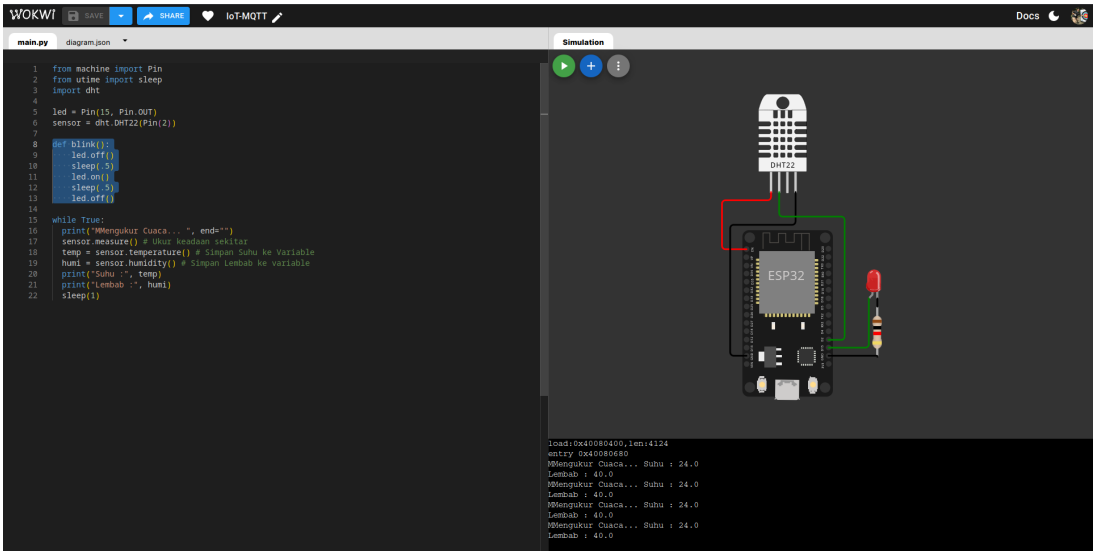


Gambar 1.16: Kode Pembaca Sensor

16. Untuk membuat LED menyala ketika pengukuran dilakukan, masukkan kode ini sebelum **while True:**

Potongan Kode

```
def blink():
    led.off()
    sleep(.5)
    led.on()
    sleep(.5)
    led.off()
```

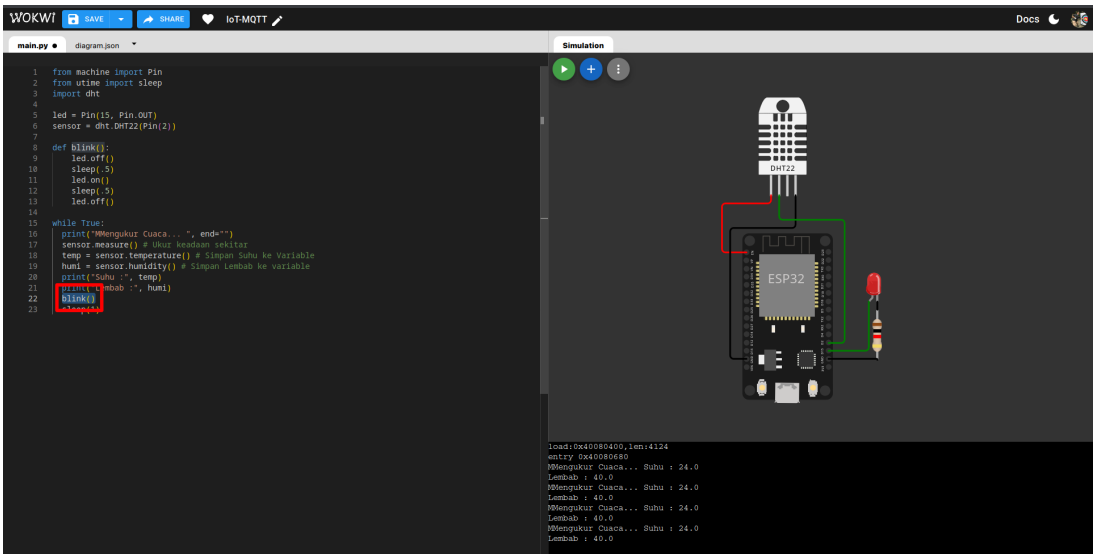


Gambar 1.17: Kode Blink LED

17. Agar LED bisa menyala, masukkan kode berikut sebelum **sleep(1)**

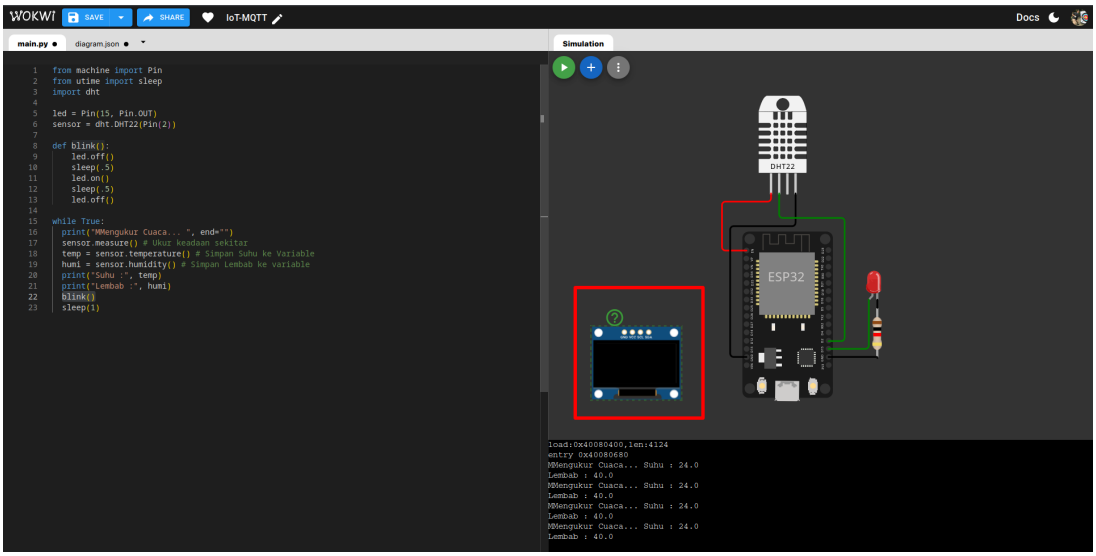
Potongan Kode

```
blink()
```



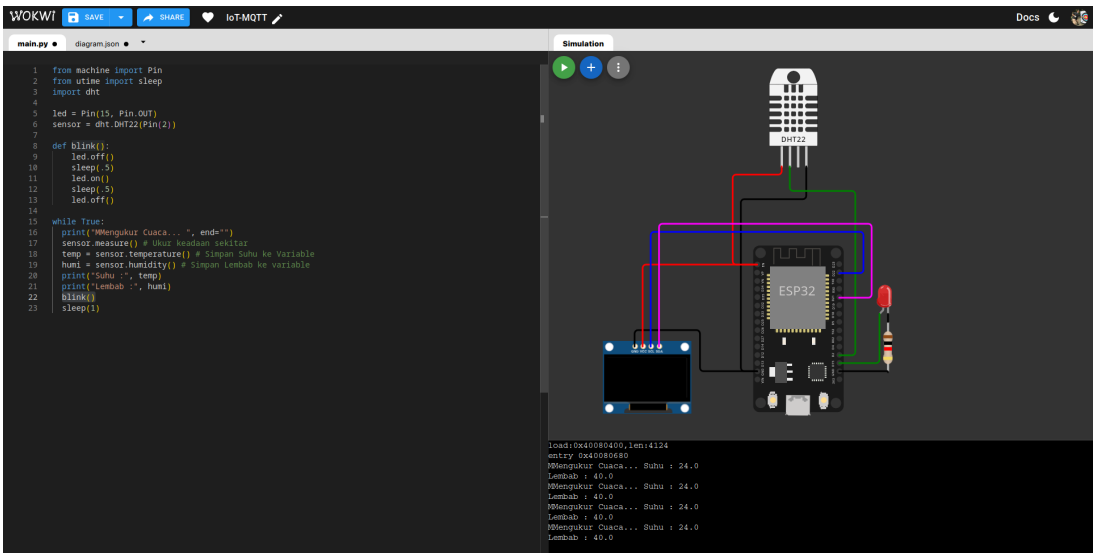
Gambar 1.18: Eksekusi Kode Blink

18. Untuk bisa menampilkan hasil ke bentuk LCD, masukkan komponen SSD1306 OLED



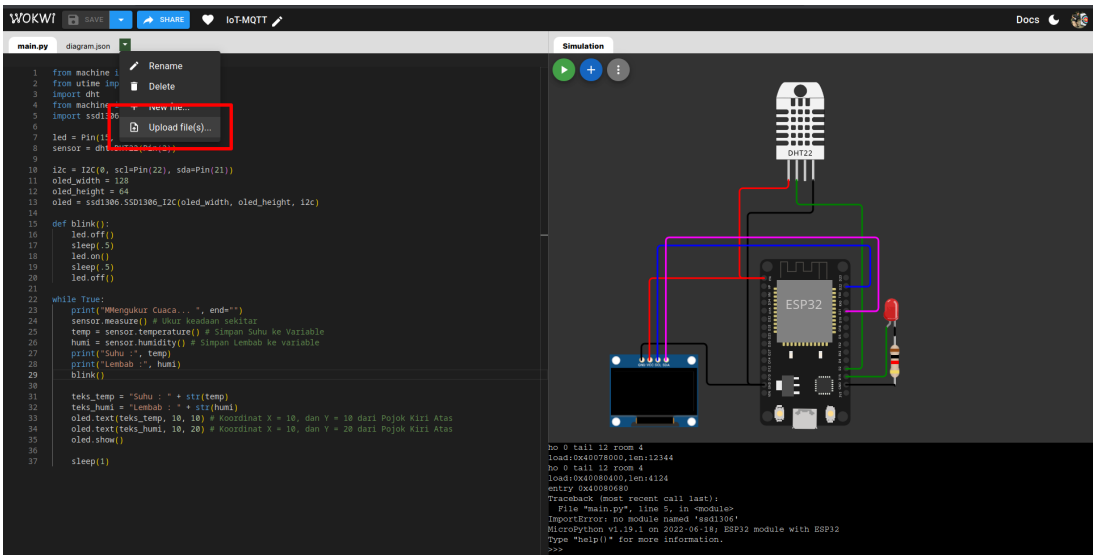
Gambar 1.19: Memasukkan Komponen SSD1306 OLED

19. Hubungkan SSD1306 OLED ke ESP32 dengan konfigurasi berikut:
    - **oled1: GND → esp: GND.2**
    - **oled1: VCC → esp: EN**
    - **oled1: SCL → esp: D22**
    - **oled1: SDA → esp: D21**



Gambar 1.20: Konfigurasi SSD1306 OLED

20. Agar bisa SSD1306 OLED bisa digunakan, unduh file **ssd1306.py** lalu unggah ke **Wokwi**
    - http://bit.ly/44tfsLo

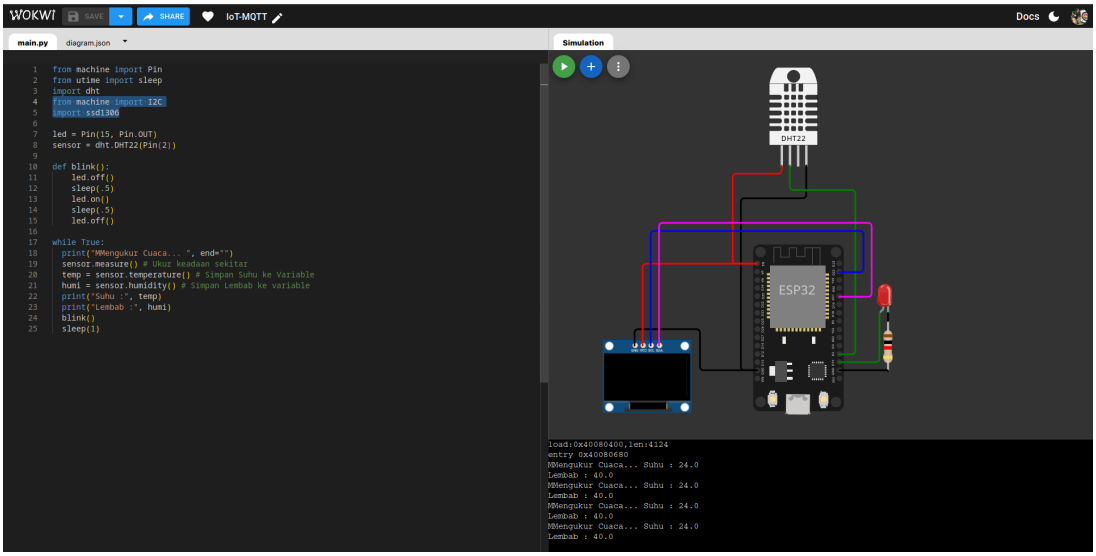


Gambar 1.21: Unggah Library SSD1306 OLED

21. Setelah itu, tambahkan kode berikut agar ESP32 dapat mengakses SSD1306 OLED. Masukkan library berikut setelah **import dht**:

Potongan Kode

```
from machine import I2C
import ssd1306
```

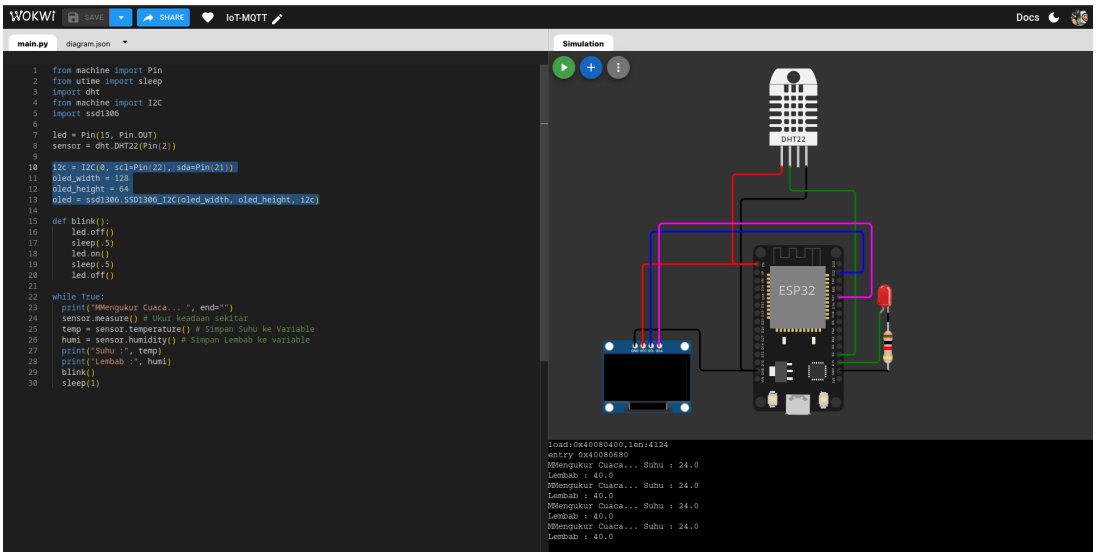


Gambar 1.22: Library SSD1306 OLED

22. Untuk menginisialisasi SSD1306 OLED masukkan kode berikut setelah **sensor = dht.DHT22(Pin(2))**

Potongan Kode

```
i2c = I2C(0, scl=Pin(22), sda=Pin(21))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)
```

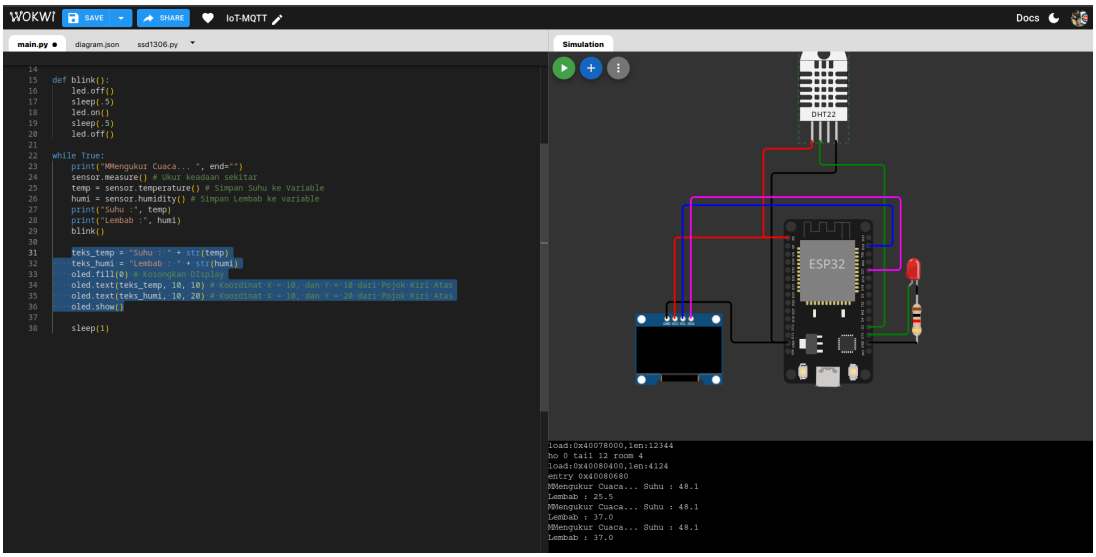


Gambar 1.23: Inisialisasi SSD1306 OLED

23. Untuk menampilkan teks ke display OLED, masukkan kode berikut setelah kode **blink()**

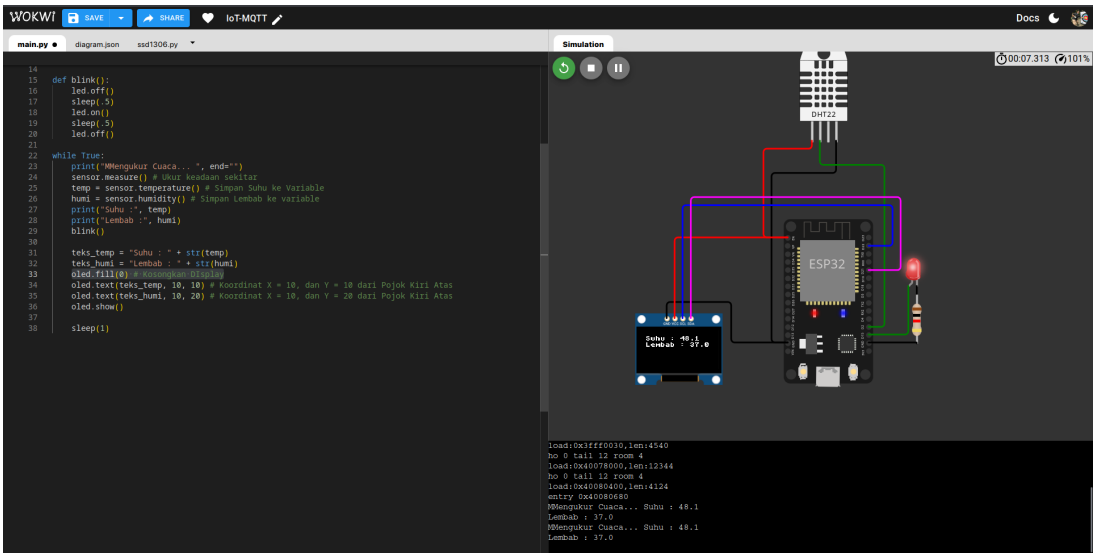
Potongan Kode

```
teks_temp = "Suhu : " + str(temp)
teks_humi = "Lembab : " + str(humi)
oled.fill(0) # Kosongkan Display
oled.text(teks_temp, 10, 10) # Koordinat X = 10, dan Y = 10
oled.text(teks_humi, 10, 20) # Koordinat X = 10, dan Y = 20
oled.show()
```



Gambar 1.24: Kode Display OLED

24. Tes kode dengan menjalankan program dan lihat hasilnya



Gambar 1.25: Hasil Kode

25. Simpan projek agar bisa dibuka kembali

1.4 Tugas SKPI

1. Tulis kode MicroPython yang membaca suhu dan kelembaban dari sensor DHT22 setiap 2 detik dan menyalakan LED apabila suhu $>$ 30°C atau kelembaban $>$ 60%.
2. Download Proyek via Save $>$ **Download project ZIP**
3. Beri nama file NIM.ZIP
4. Kirim ke Google Drive : Tugas SKPI

Bab 2

Wokwi #2 - Internet dan MQTT

2.1 Mengenai Wokwi #2 - Internet dan MQTT

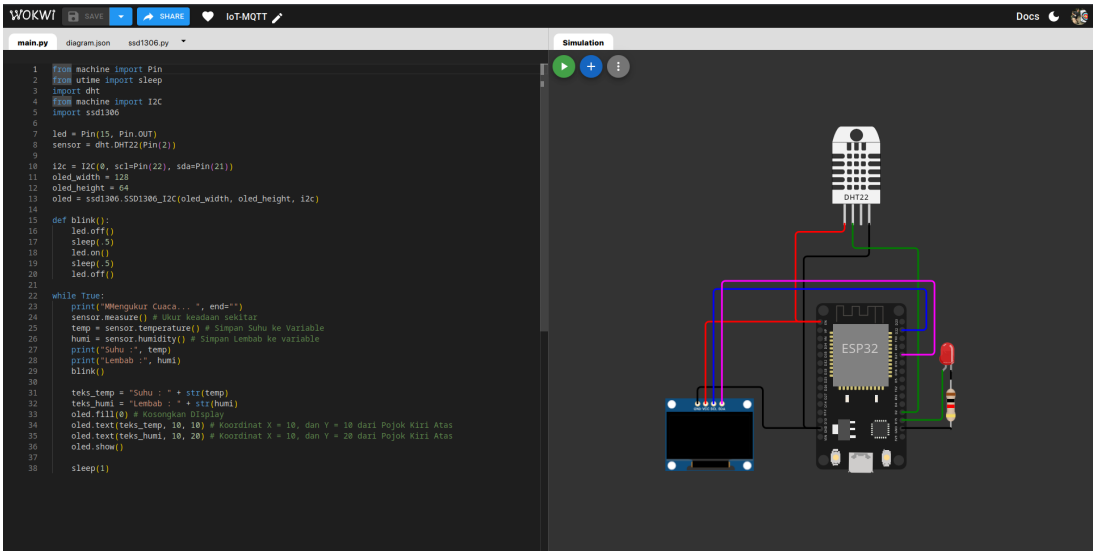
Di bagian ini mahasiswa diajarkan bagaimana menggunakan Wokwi untuk menghubungkan ESP32 ke Internet dan layanan MQTT

2.2 Perangkat Lunak Dibutuhkan

1. Web Browser

2.3 Tutorial

1. Login akun ke **Wokwi** dan buka kembali projek yang sudah dibuat.

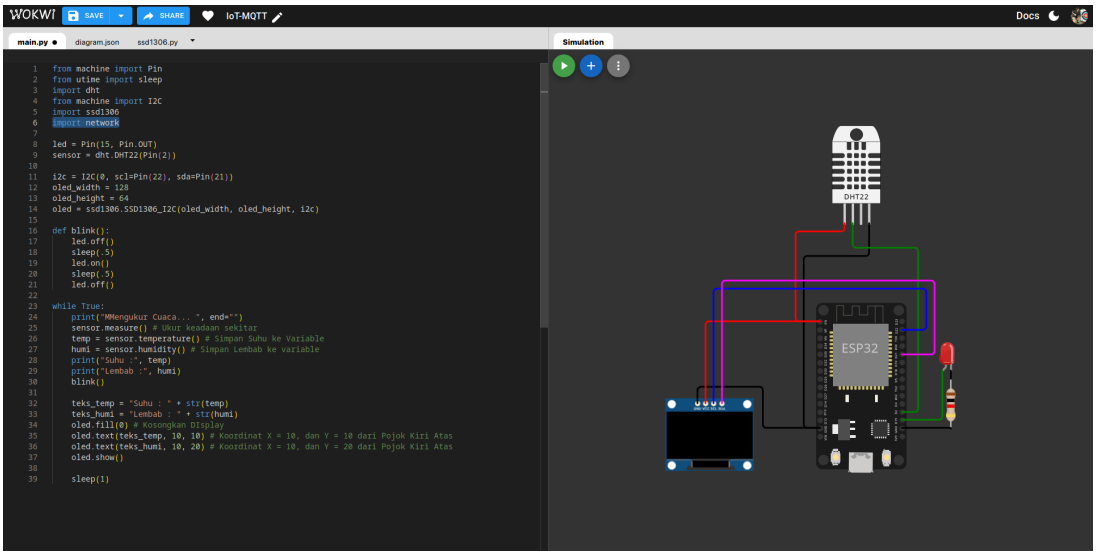


Gambar 2.1: Buka kembali projek

2. Untuk menghubungkan ke Internet, bisa dilakukan dengan mudah. Tambahkan library utama berikut setelah kode **import ssd1306**

Potongan Kode

```
import network
```

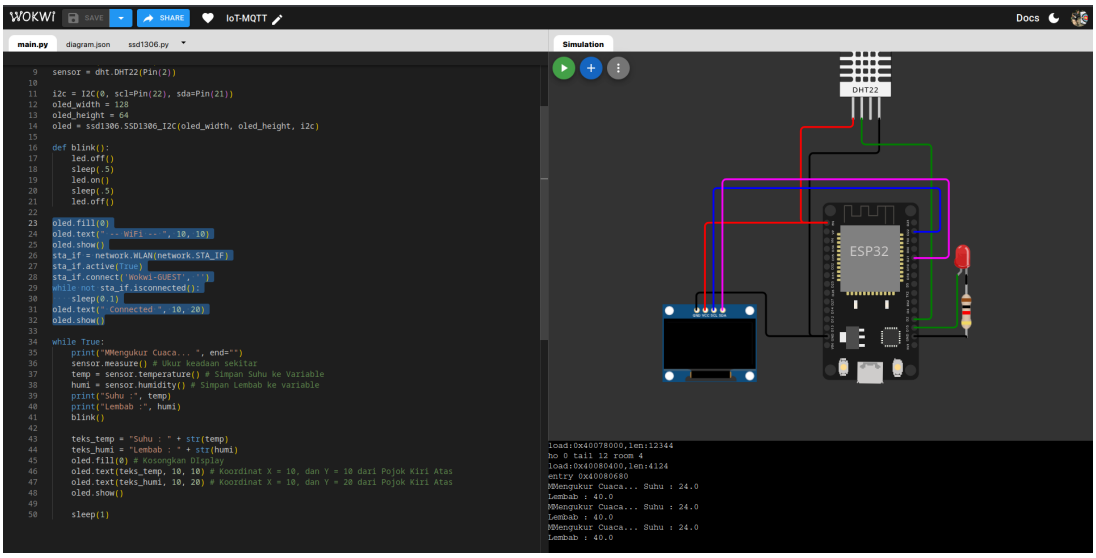


Gambar 2.2: Impor Library Network

3. Untuk menghubungkan ke Internet, masukkan kode berikut sebelum kode **while True:**

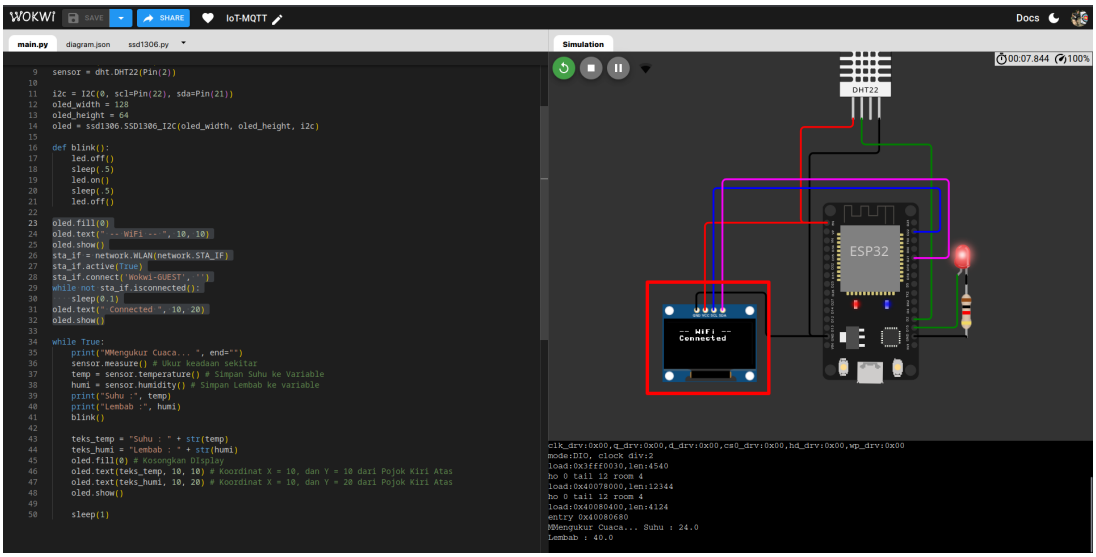
Potongan Kode

```
oled.fill(0)
oled.text(" -- WiFi -- ", 10, 10)
oled.show()
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('Wokwi-GUEST', '')
while not sta_if.isconnected():
    sleep(0.1)
oled.text(" Connected ", 10, 20)
oled.show()
```



Gambar 2.3: Kode Penghubung ke Internet

4. Tes perangkat apakah berhasil terhubung atau tidak

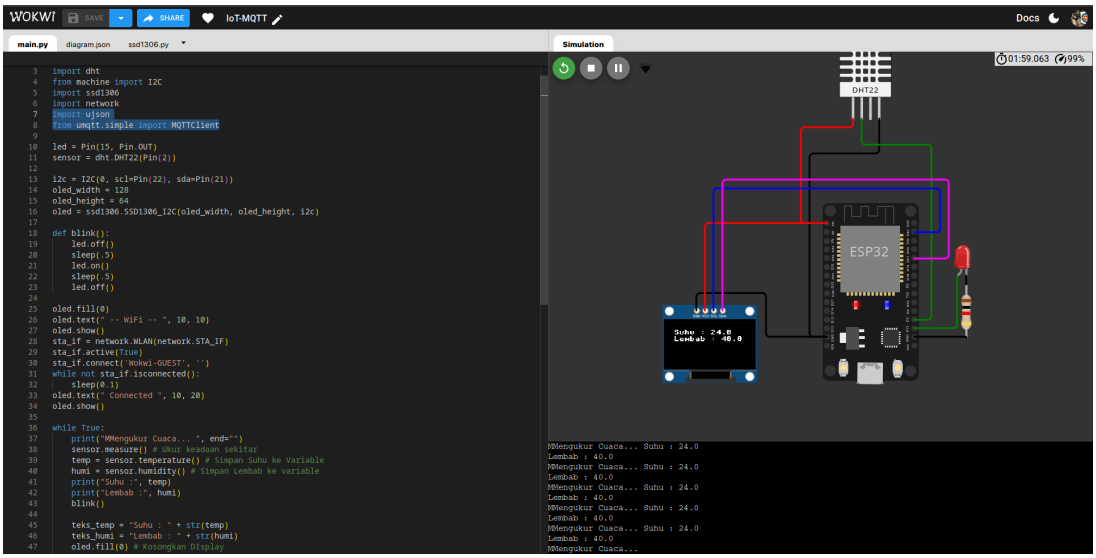


Gambar 2.4: Kode Penghubung ke Internet

5. Jika sudah berhasil, masukkan kode library berikut untuk mengirimkan data-data sensor ke protokol MQTT. Letakkan di bawah **import network**

Potongan Kode

```
import ujson
from umqtt.simple import MQTTClient
```



Gambar 2.5: Impor Library MQTT dan JSON

6. Berikutnya adalah membuat parameter untuk protokol MQTT. Pilih salah satu broker MQTT beserta port nya dari list berikut. Lalu masukkan kode sebelum **led = Pin()**

   (a) broker.mqtt.cool 1883

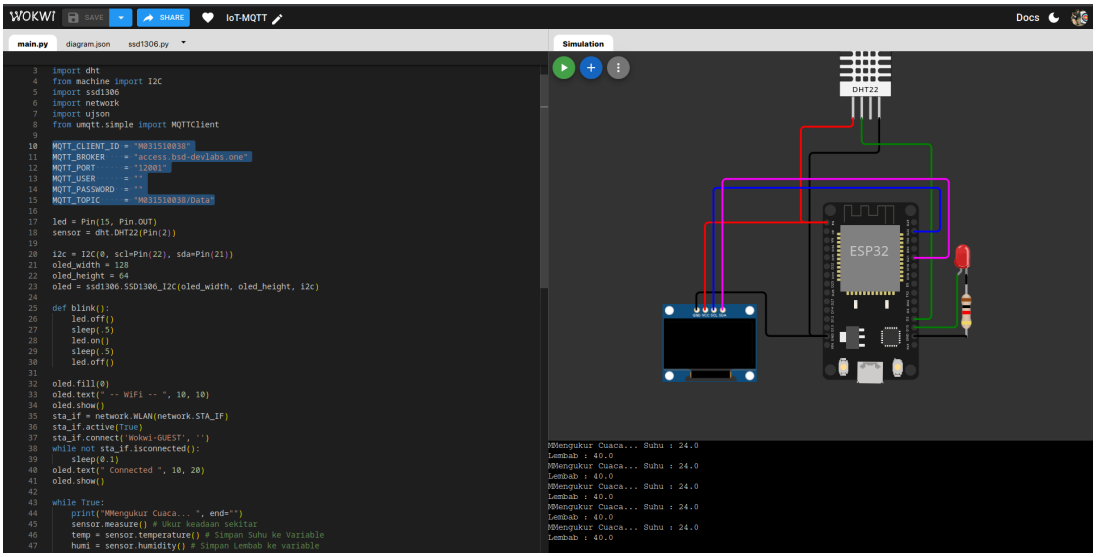
   (b) test.mosquitto.org 1883

   (c) broker.hivemq.com 1883

(d) access.bsd-devlabs.one 12001

Potongan Kode

```
MQTT_CLIENT_ID = "NIM" # Ganti NIM dengan NIM Mahasiswa
MQTT_BROKER    = "NAMA" # Ganti NAMA dengan 1 Server
MQTT_PORT      = "PORT" # Ganti PORT dengan Nomor Server
MQTT_USER      = "" # Kosong
MQTT_PASSWORD  = "" # Kosong
MQTT_TOPIC     = "NIM/Data" # Ganti NIM dengan NIM Mahasiswa
```



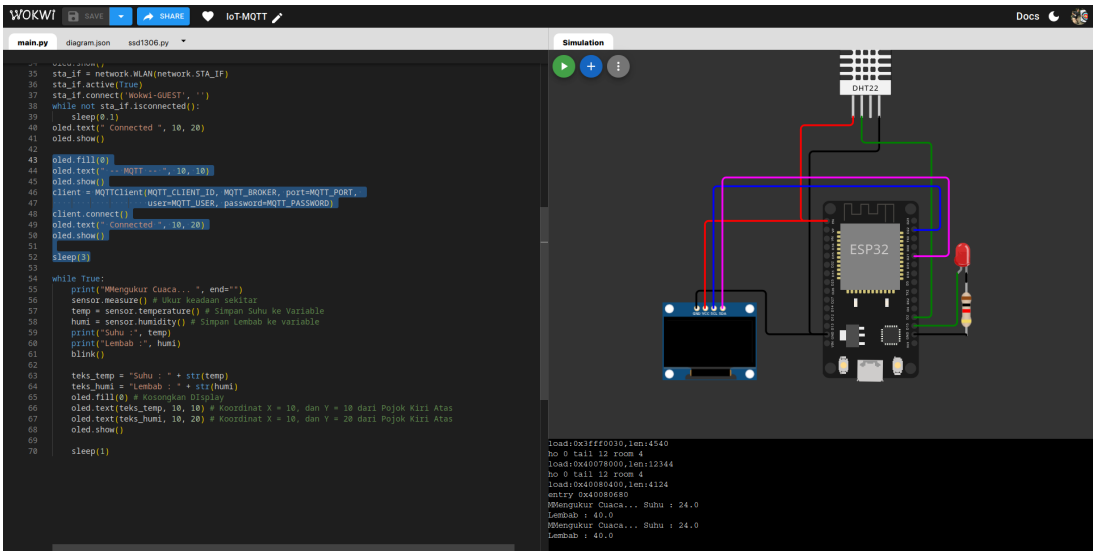
Gambar 2.6: Konfigurasi MQTT

7. Setelah mengkonfigurasikan parameter MQTT, lanjutkan dengan kode penghubung ke MQTT. Tambahkan sebelum **while True:**

Potongan Kode

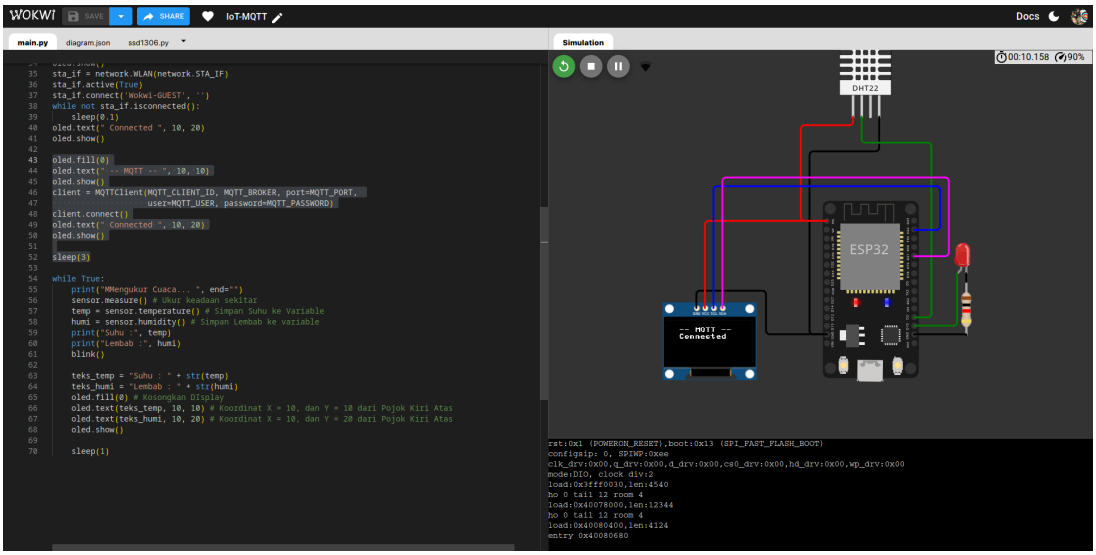
```
oled.fill(0)
oled.text(" -- MQTT -- ", 10, 10)
oled.show()
client = MQTTClient(MQTT_CLIENT_ID, MQTT_BROKER, port=MQTT_PORT,
                    user=MQTT_USER, password=MQTT_PASSWORD)
client.connect()
oled.text(" Connected ", 10, 20)
oled.show()

sleep(3)
```



Gambar 2.7: Kode Menghubungkan MQTT

8. Tes kode dengan menekan **Run**

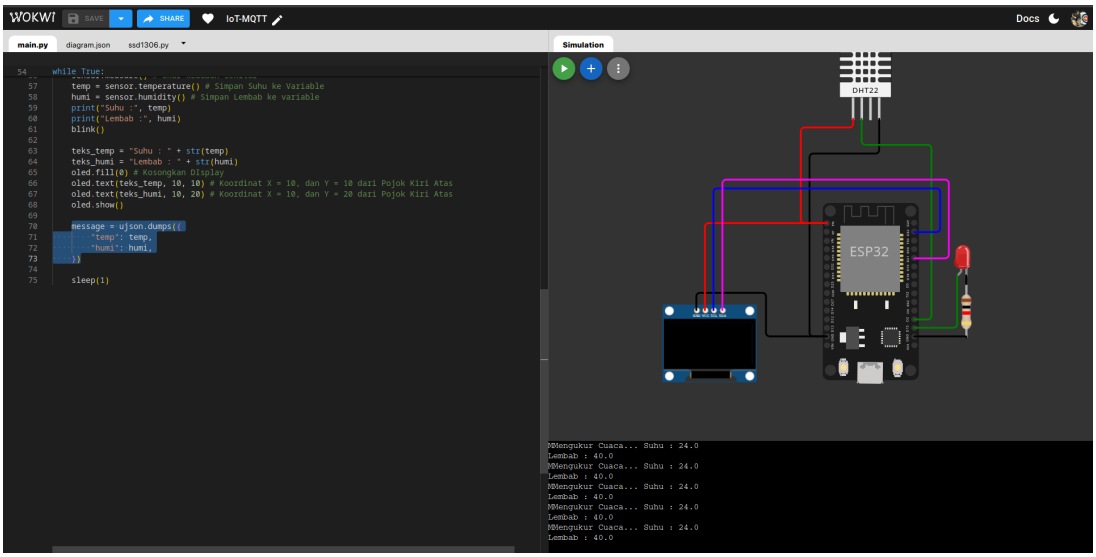


Gambar 2.8: Perangkat Terhubung ke MQTT

9. Berikutnya adalah menambahkan kode untuk membungkus data sensor dalam format JSON sebelum dikirimkan ke Server. Letakkan sebelum **sleep(1)** di bagian akhir kode

Potongan Kode

```
message = ujson.dumps({
    "temp": temp,
    "humi": humi,
})
```



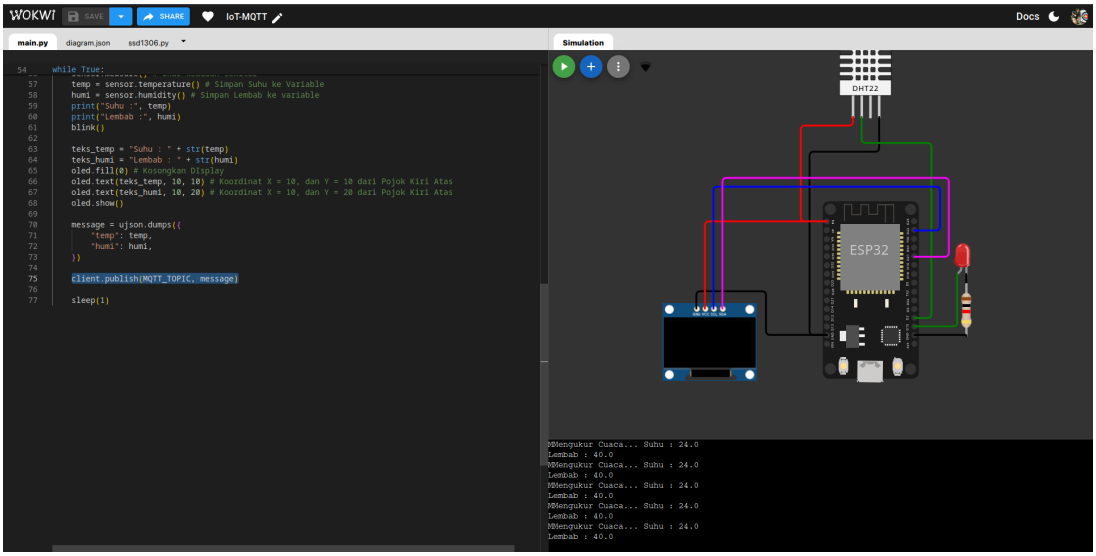
Gambar 2.9: Kode Pembungkus JSON

NIM/Data

10. Setelah data sensor dibungkus dengan JSON, maka langkah terakhir adalah mengirimkannya ke Server. Tambahkan kode setelahnya berikut ini:

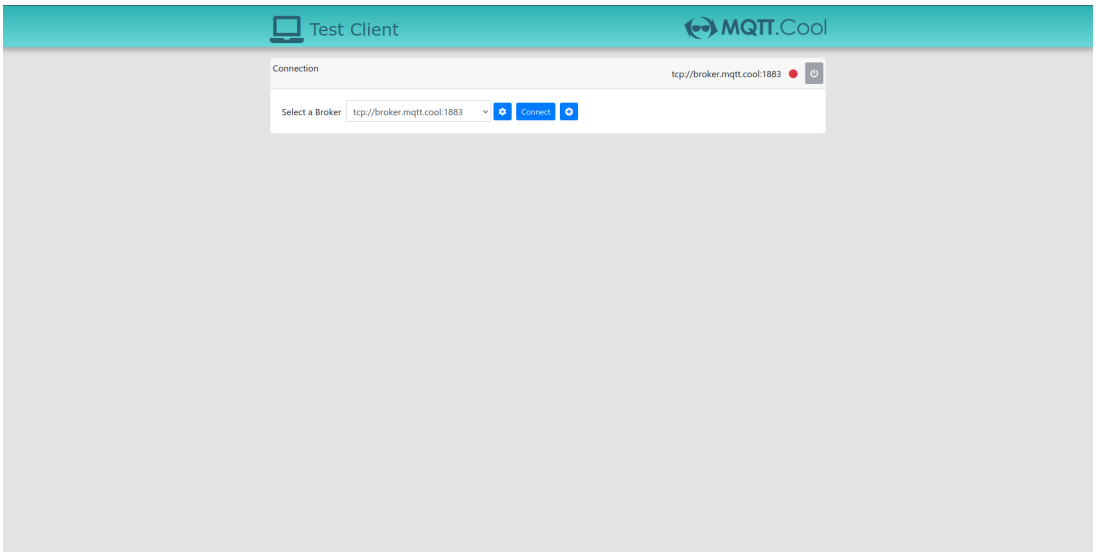
Potongan Kode

```
client.publish(MQTT_TOPIC, message)
```



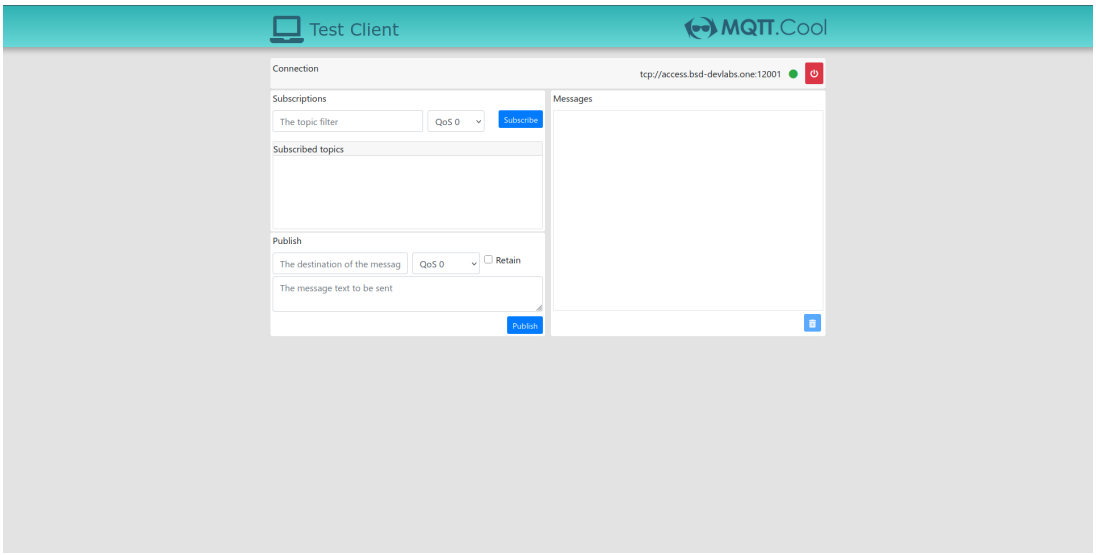
Gambar 2.10: Kode Kirim Data ke MQTT

11. Untuk mengetes apakah berhasil atau tidak, buka website : https://testclient-cloud.mqtt.cool/. Lihat Daftar Server 6 yang digunakan. Jika menggunakan server 1-3, server tersebut sudah tersedia langsung. Tapi jika menggunakan server 4 harus menambahkan sendiri. Lalu klik **Connect**



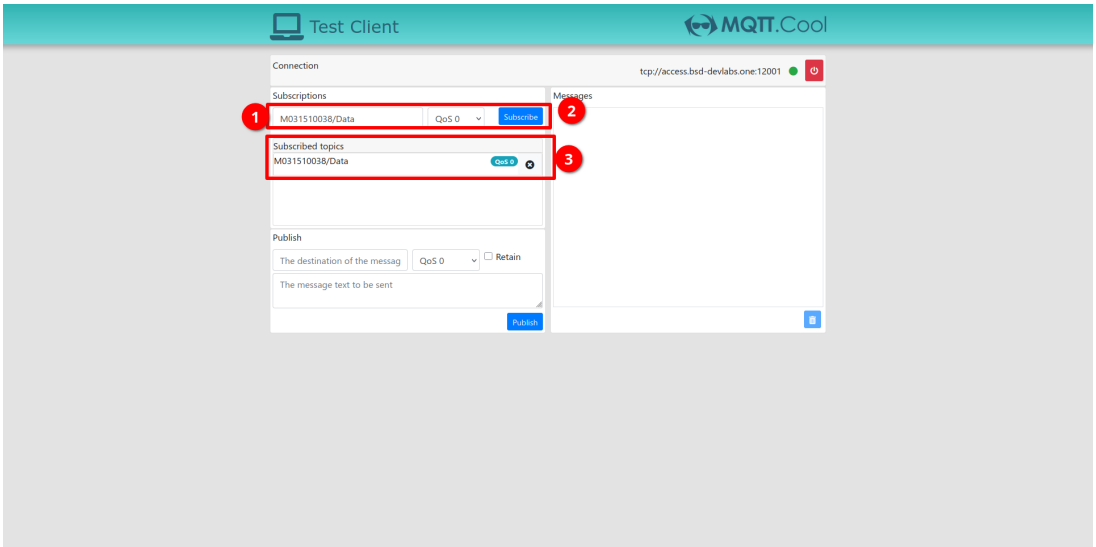
Gambar 2.11: Website MQTT Client

12. Jika terhubung, maka tampilan akan berubah seperti berikut:



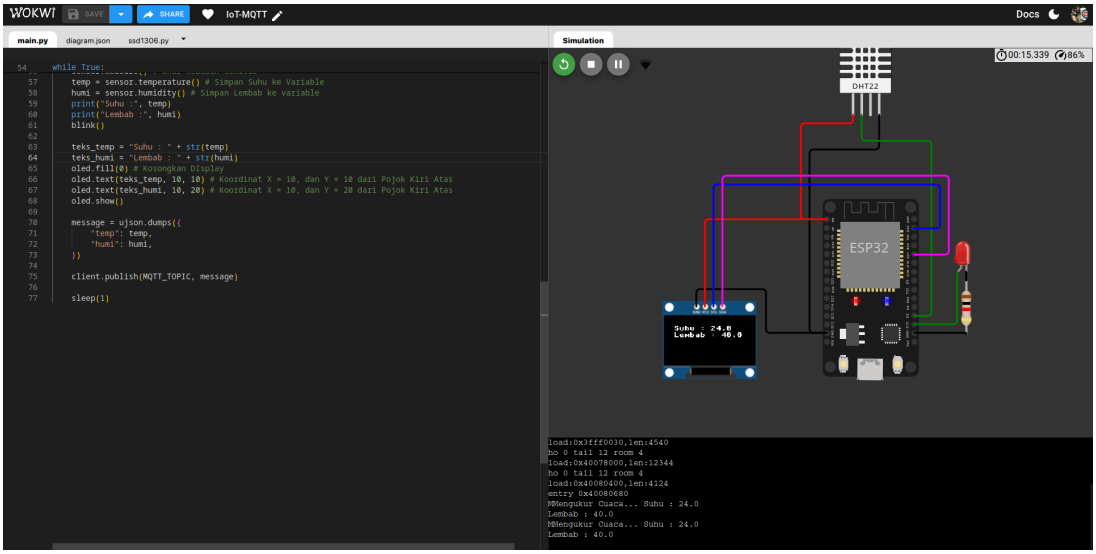
Gambar 2.12: Tampilan Client Setelah Connect

13. Masukkan Topic sesuai **NIM/Data** di bagian **Subcription**, lalu klik **Subscribe**

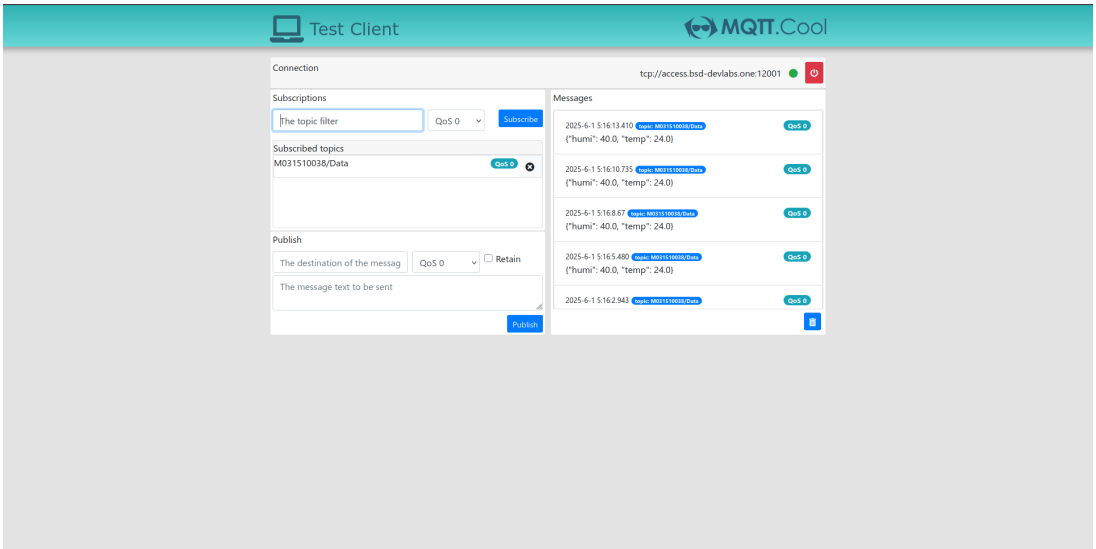


Gambar 2.13: Tampilan Client Setelah Connect

14. Jika sudah **subscribe**, kembali ke Wokwi dan klik **Run/Play** tunggu beberapa saat hingga MQTT Client mendapatkan data

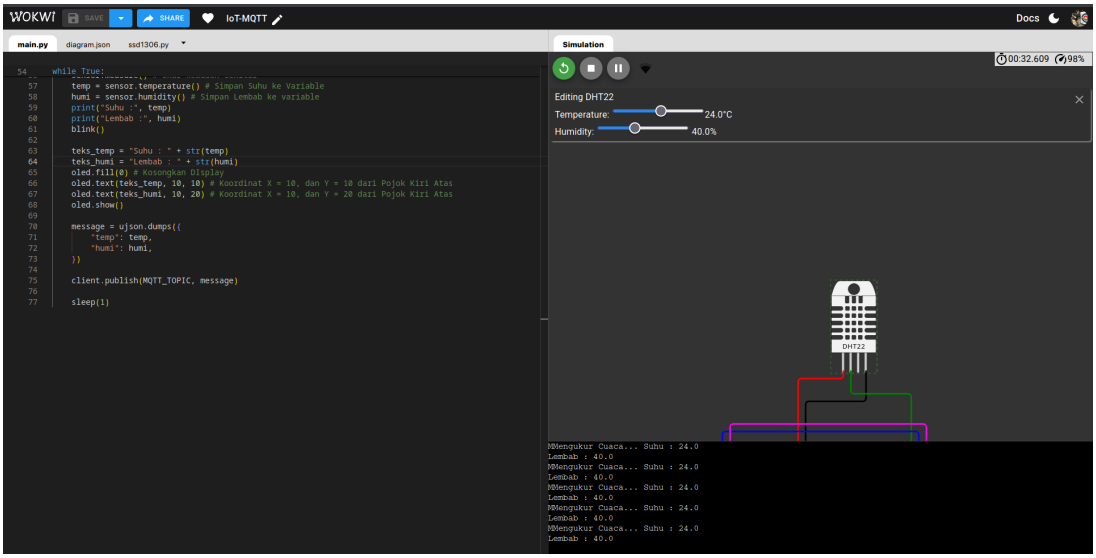


Gambar 2.14: Perangkat IoT Mengirimkan Data



Gambar 2.15: Client Menerima Data

15. Untuk mengubah nilai sensor DHT22, cukup klik DHT22 dan ubah slider yang muncul



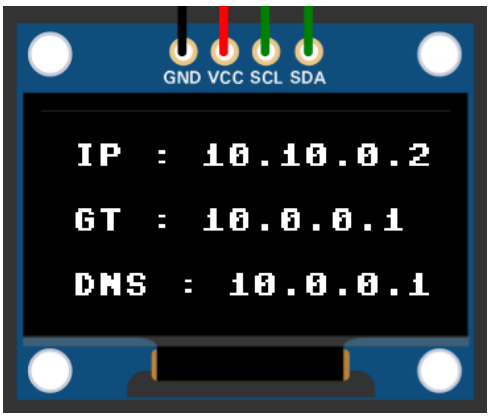
Gambar 2.16: Mengubah Output DHT22

16. Wokwi sudah berhasil mengirimkan data dan melanjutkan ke tahap 3

2.4 Tugas SKPI

1. Tulis kode MicroPython untuk menampilkan IP, Gateway dan DNS perangkat. Gunakan potongan kode berikut untuk mendapatkan info IP:
   - ip = sta_if.ifconfig()[0]
   - gt = sta_if.ifconfig()[2]
   - dns = sta_if.ifconfig()[3]

2. Tampilkan setelah Wi-Fi *Connected*dan sebelum MQTT terhubung
3. Berikan **Delay 3 Detik** sebelum MQTT berlangsung



4. Download Proyek via Save > **Download project ZIP**
5. Beri nama file NIM.ZIP
6. Kirim ke Google Drive : Tugas SKPI

Bab 3

Flow Programming Node-RED #1

3.1 Mengenal Flow Programming

Di bagian ini mahasiswa diajarkan melakukan *Flow Programming* untuk menangkap data dari MQTT, diolah dan disimpan ke dalam bentuk CSV.

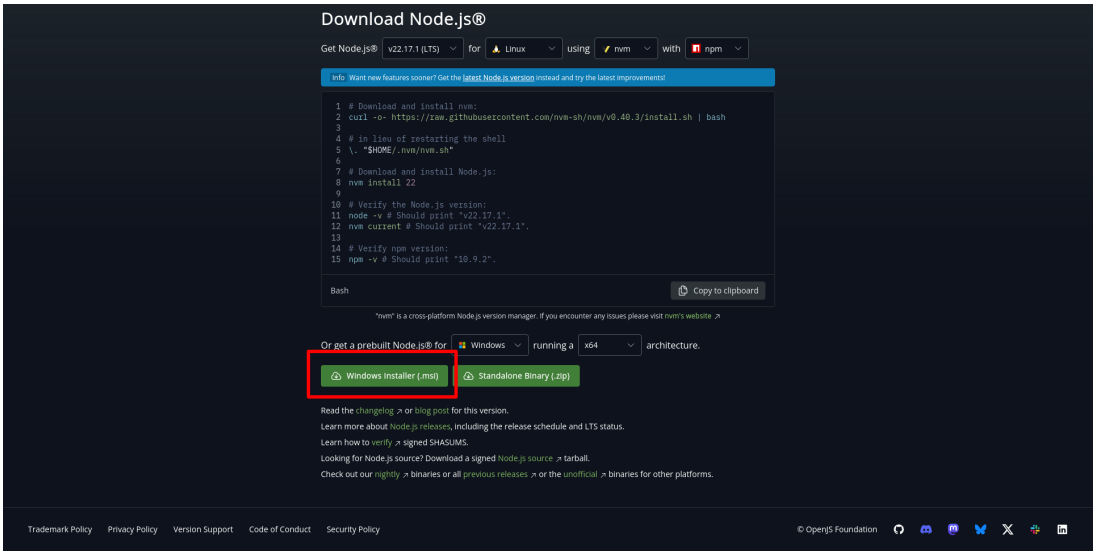
3.2 Perangkat Lunak Dibutuhkan

- NodeJS **Windows Installer (.msi)**(Download)

3.3 Tutorial

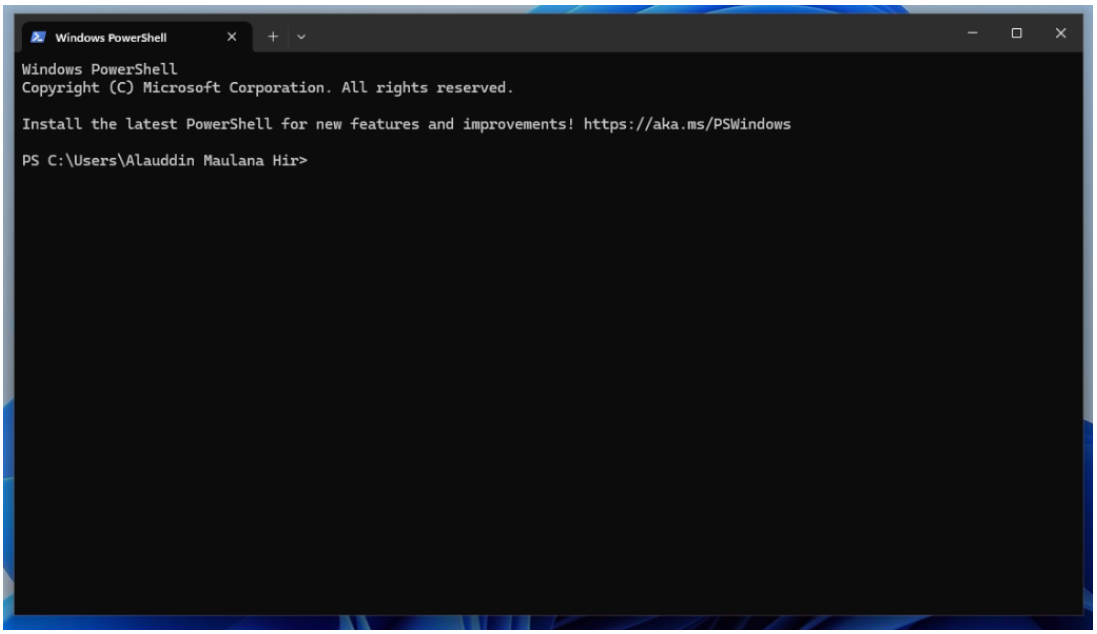
3.3.1 Instalasi NodeJS

1. Unduh program **NodeJS.msi** dilink yang sudah diberikan



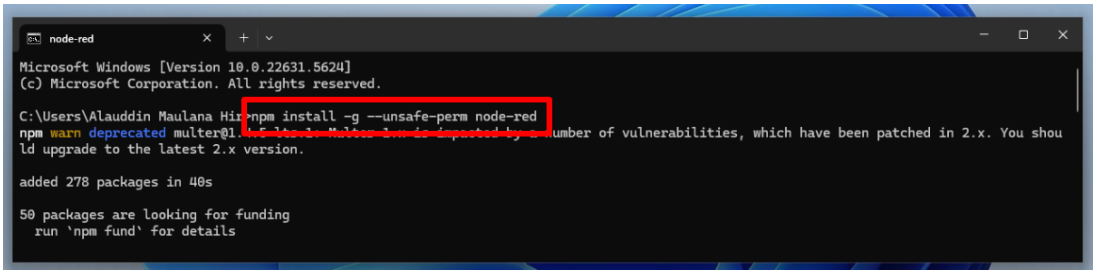
Gambar 3.1: Unduh Program

2. Setelah instalasi selesai, buka **Command Prompt**



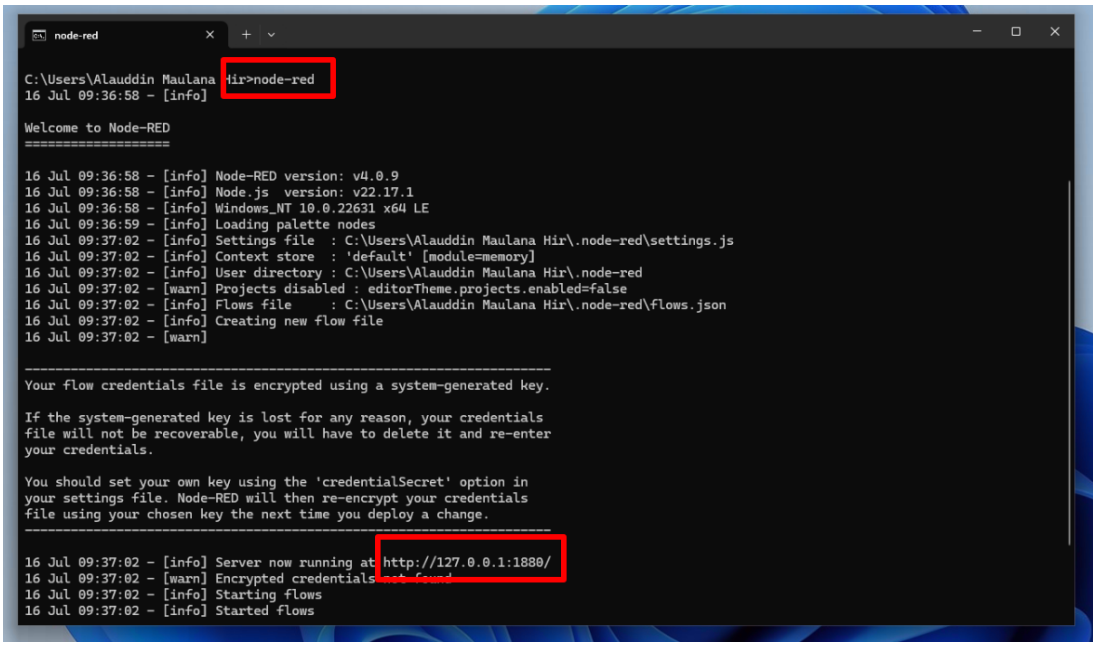
Gambar 3.2: Buka Command Prompt

3. Install **Node-RED** dengan menggunakan perintah **npm install -g --unsafe-perm node-red**



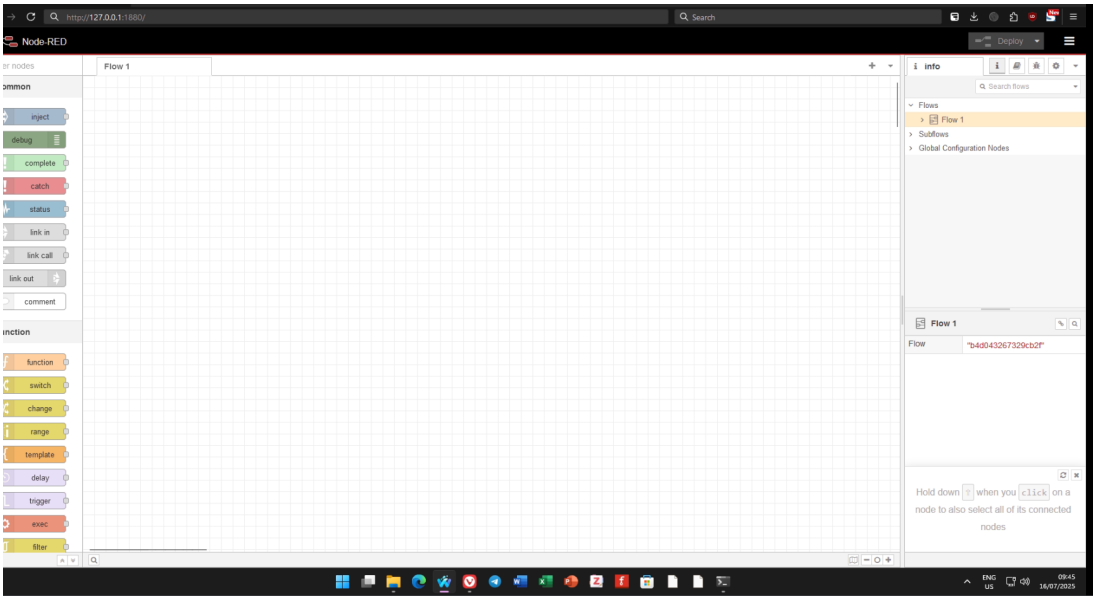
Gambar 3.3: Instalasi Node-Red

4. Jalankan **Node-RED** melalui **Command Prompt** tersebut dengan menggunakan perintah **node-red**



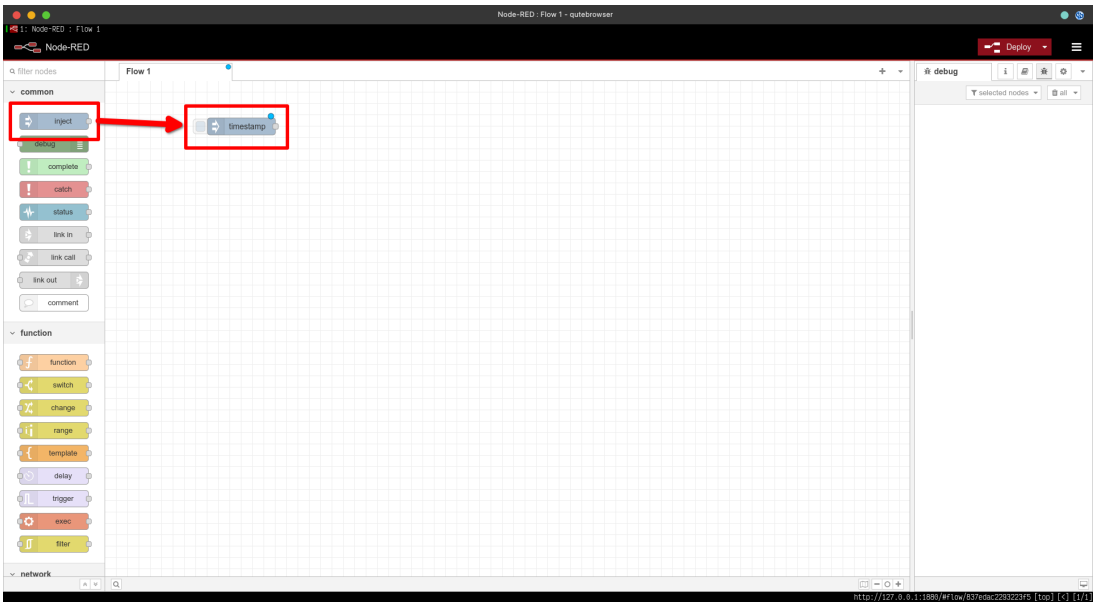
Gambar 3.4: Jalankan Node-Red

5. Dengan menggunakan browser buka `http://127.0.0.1:1880/`. Maka akan muncul tampilan berikut



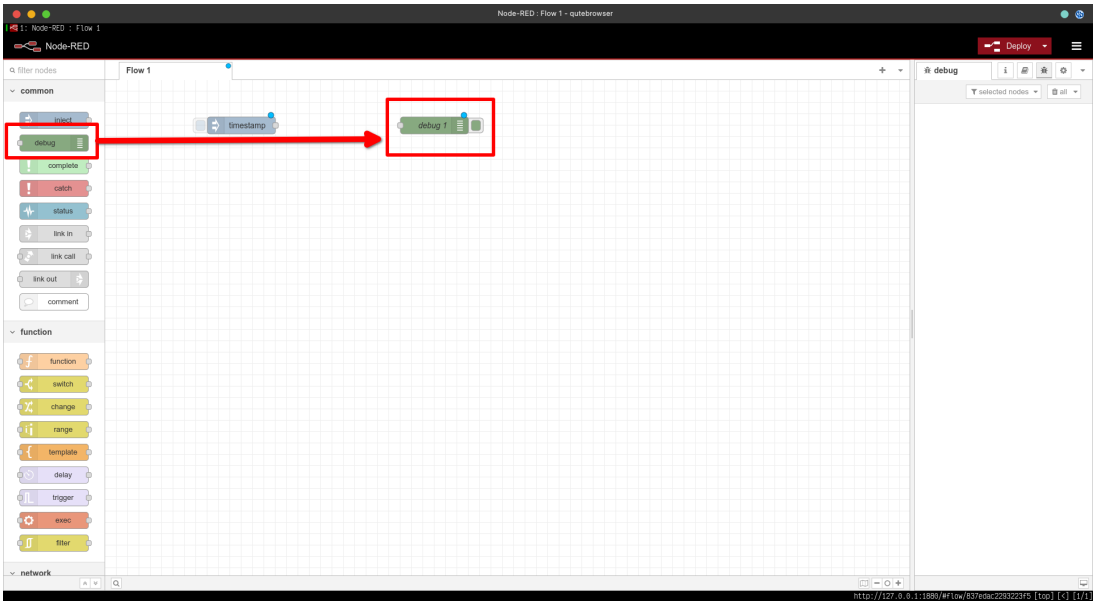
Gambar 3.5: Tampilan Program

6. Untuk membuat **flow/alir** sederhana, masukkan **Common** → **inject** ke dalam kanvas. Node **inject** berfungsi untuk mengirimkan data ke node berikutnya



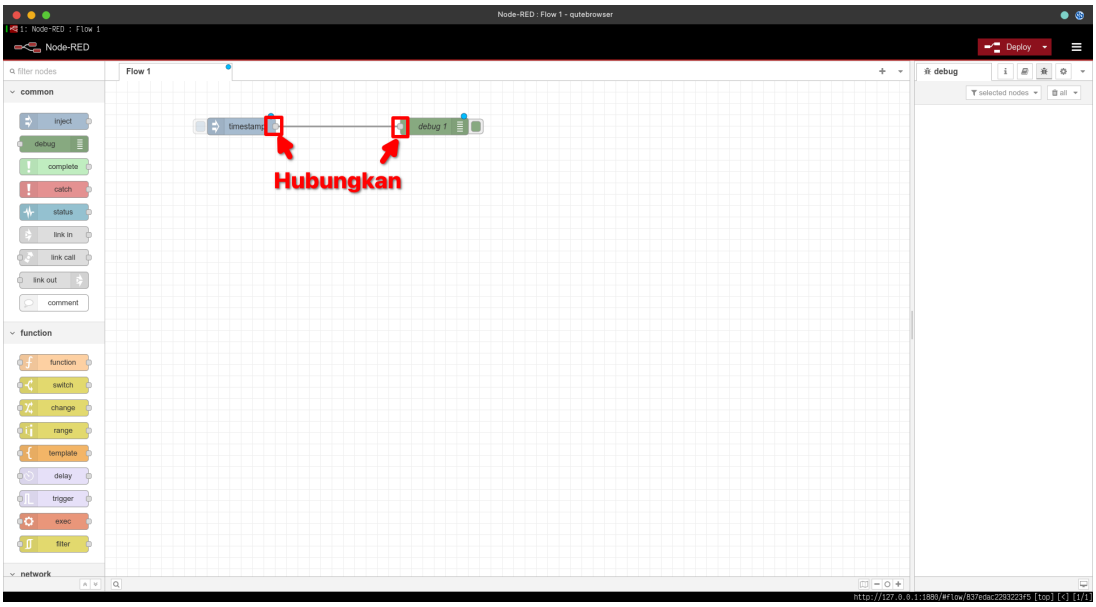
Gambar 3.6: Masukkan Node inject

7. Untuk menerima data yang dikirim dari **inject** akan diterima dengan node **Common** → **debug**. Node **debug** berfungsi untuk mengecek apakah data benar-benar dikirimkan atau tidak.



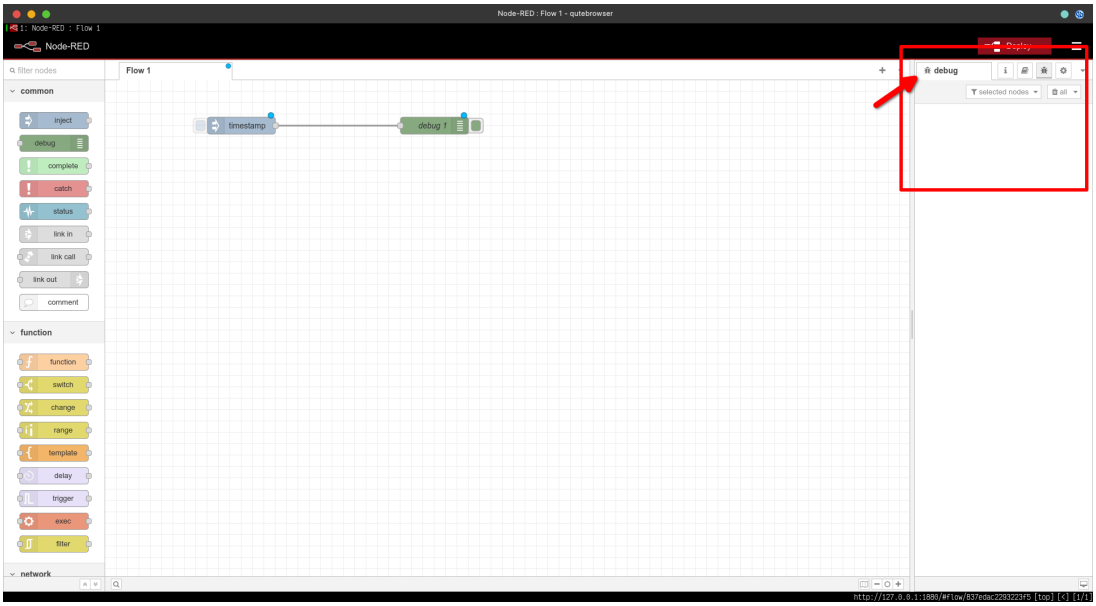
Gambar 3.7: Masukkan Node debug

8. Untuk membuat **alir/flow** hubungkan lingkaran kecil **inject** ke lingkaran kecil **debug**



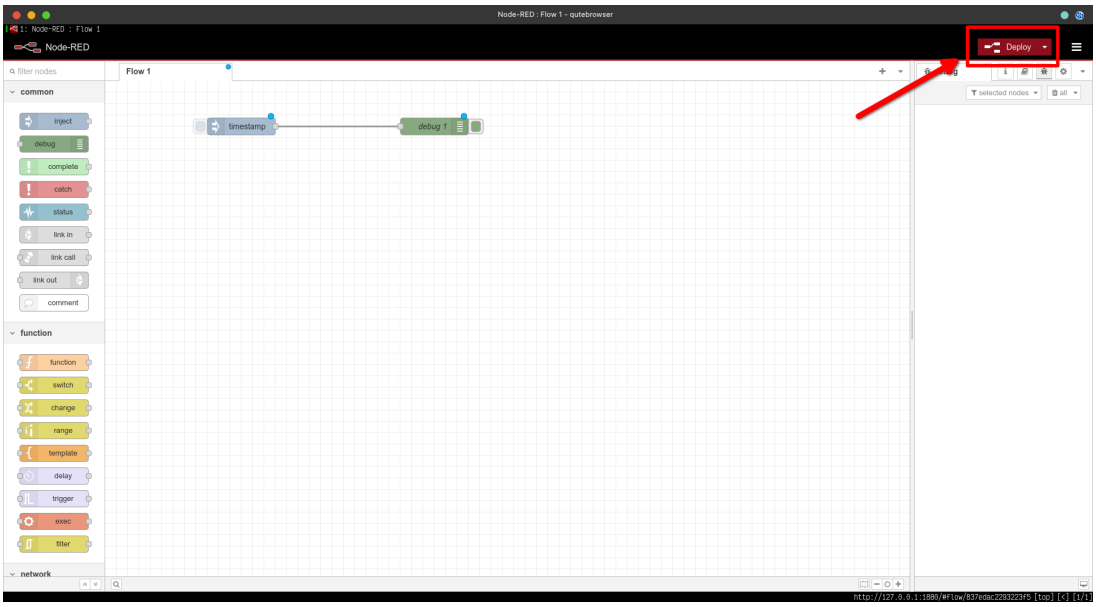
Gambar 3.8: Menghubungkan Node

9. Untuk mengetes apakah **alir/flow** berjalan dengan baik, pertama adalah memastikan **window debug** ditampilkan



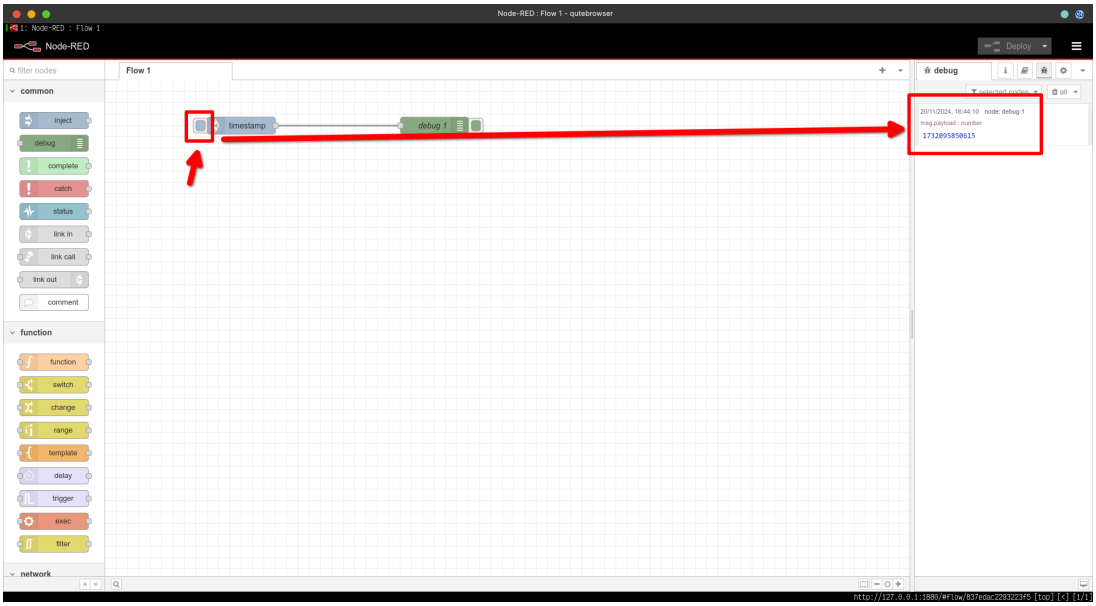
Gambar 3.9: Tampilan Window Debug

10. Jika sudah muncul, klik **Deploy** untuk mengaktifkan **alir/flow**



Gambar 3.10: Mengaktifkan Alir

11. Untuk mengirim data secara **manual**, cukup tekan **tombol** yang ada di sisi kiri **node inject** dan hasil akan ditampilkan di window **debug** sebelah kanan

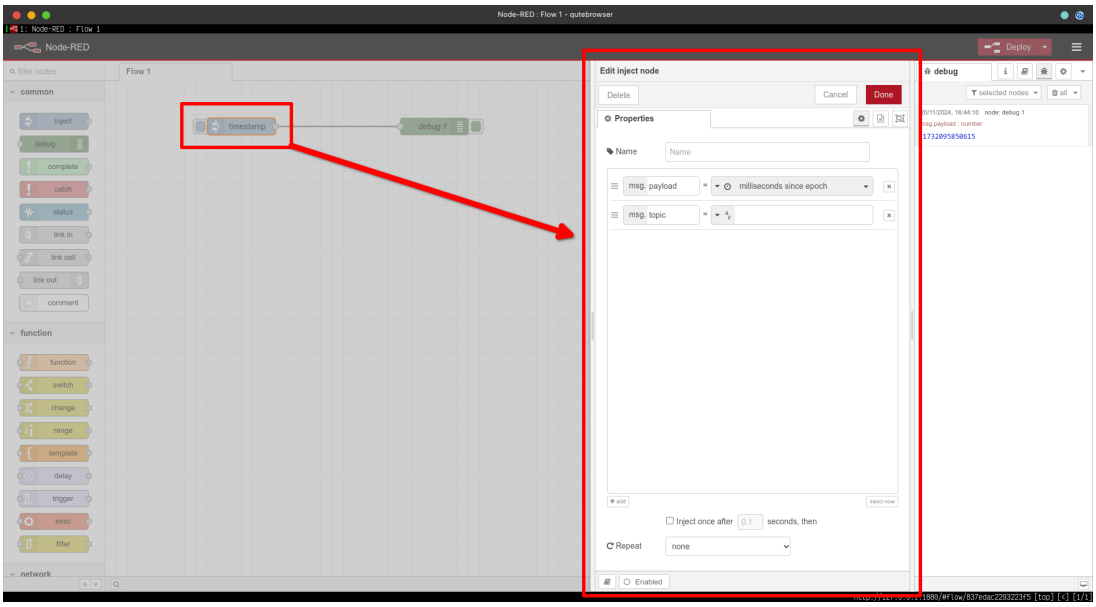


Gambar 3.11: Mengirim Data

12. Node **inject** dapat dimodifikasi dengan bahasa **Javascript** untuk mengirimkan data **random** secara otomatis nantinya.

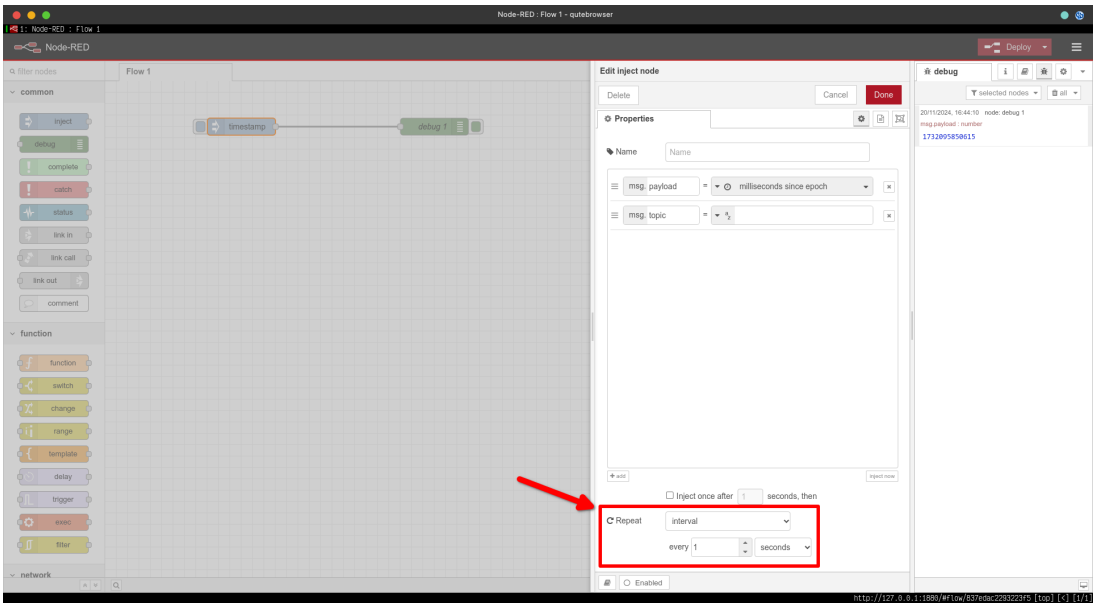
3.3.2 Otomatisasi Injeksi Data

1. Untuk mengotomatiskan injeksi data secara berkala, **Klik Dua Kali** node **inject** untuk menampilkan **properti** node **inject**.



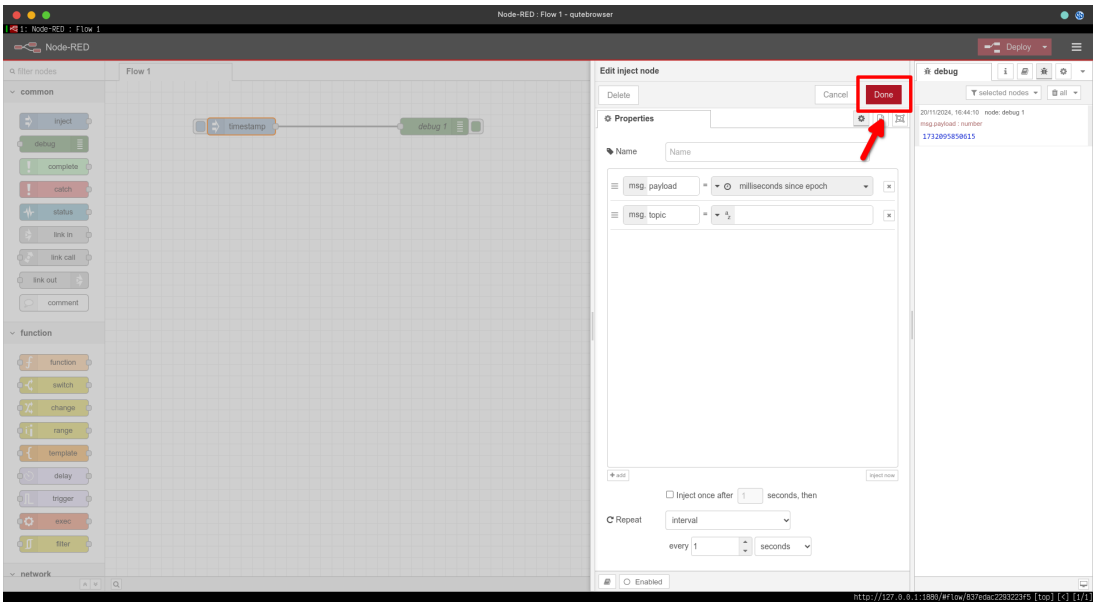
Gambar 3.12: Menampilkan Properti Inject

2. Untuk mengotomatiskan pengiriman data, ubah **Properti Repeat** di bagian bawah dari **none** → **interval**.



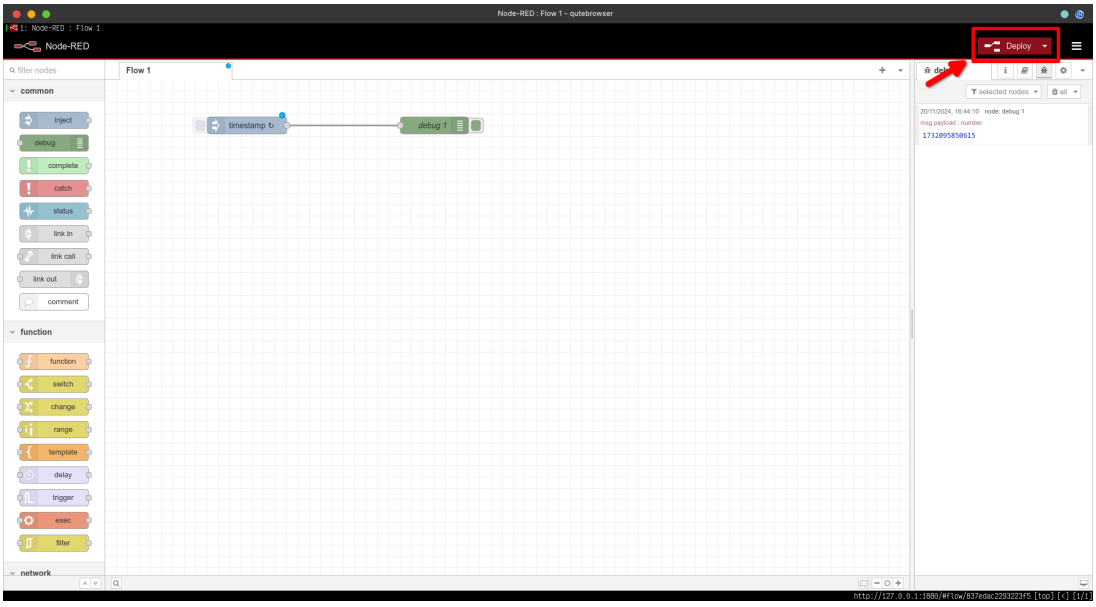
Gambar 3.13: Mengubah Mode Pengiriman

3. Klik **Done** ketika sudah selesai mengubah konfigurasi interval



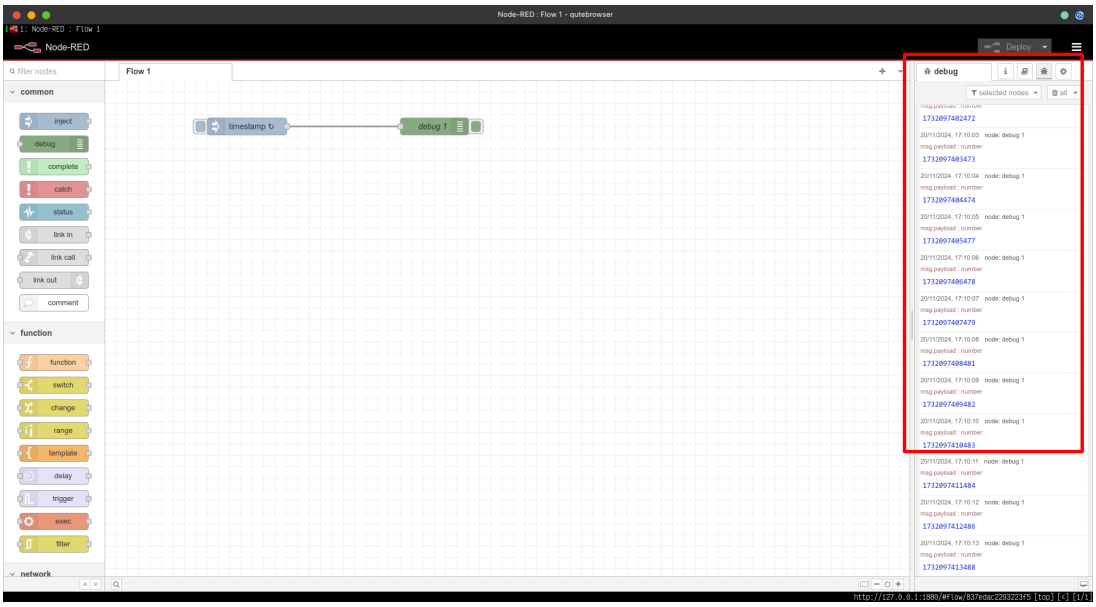
Gambar 3.14: Menyimpan Konfigurasi Inject

4. **Deploy Ulang** agar **alir/flow** diterapkan



Gambar 3.15: Menerapkan Perubahan

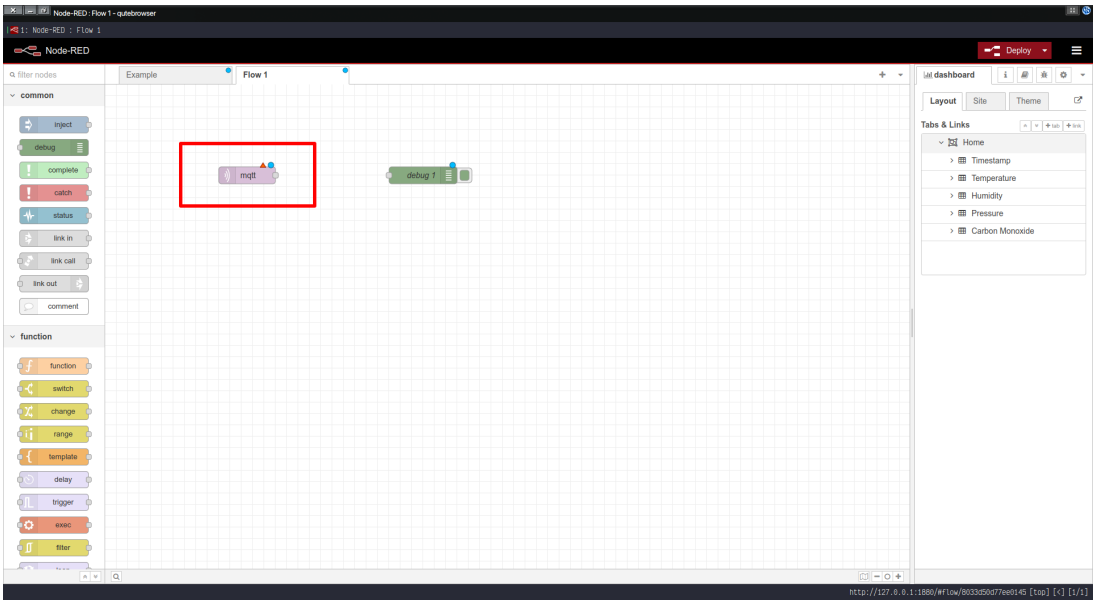
5. Pengiriman data akan dikirimkan secara otomatis. Jika tidak muncul di **debug**, Klik **tombol** yang ada di **inject** dan lihat data yang dikirimkan



Gambar 3.16: Hasil Otomatisasi Pengiriman Data

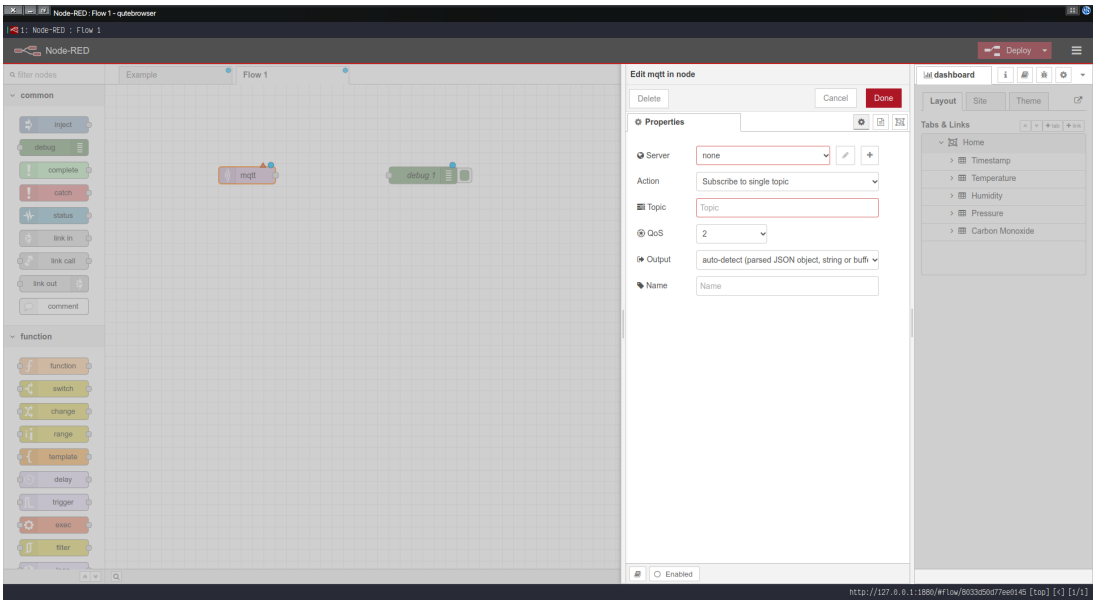
3.3.3 Menerima Data dari MQTT

1. Buka kembali Wokwi dan projek yang sudah dibuat.
2. Hapus node **Inject** tadi, dan masukkan node baru kategori **network** dengan nama **mqtt in**



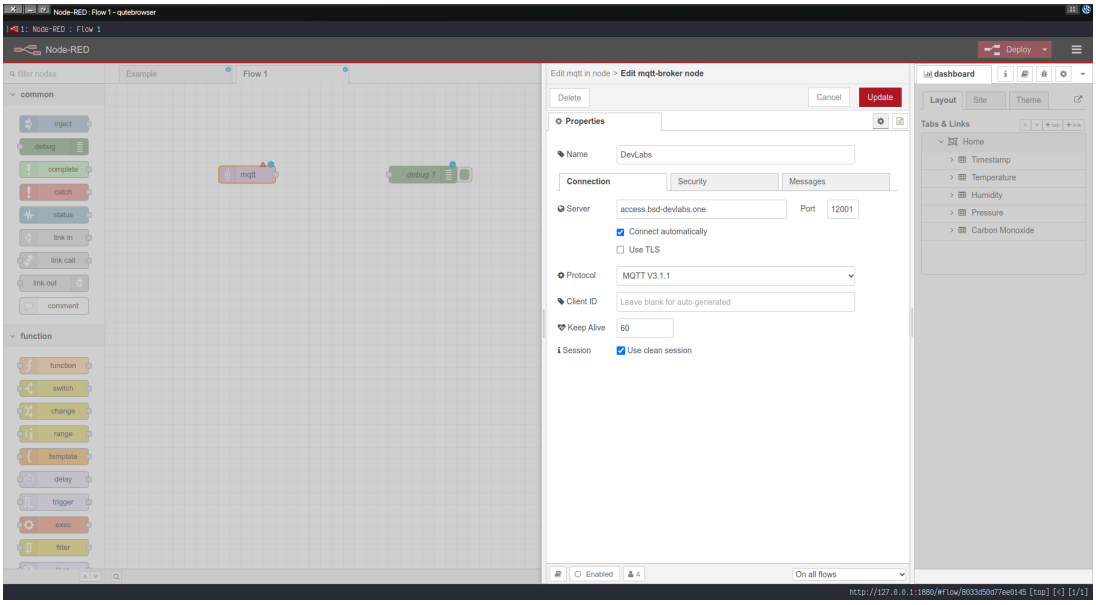
Gambar 3.17: Memasukkan Node MQTT in

3. Namun karena node ini belum dikonfigurasikan dengan benar, maka muncul peringatan di node tersebut. Untuk mengkonfigurasikan node tersebut, **Double Klik** mqtt in, dan akan muncul window konfigurasi



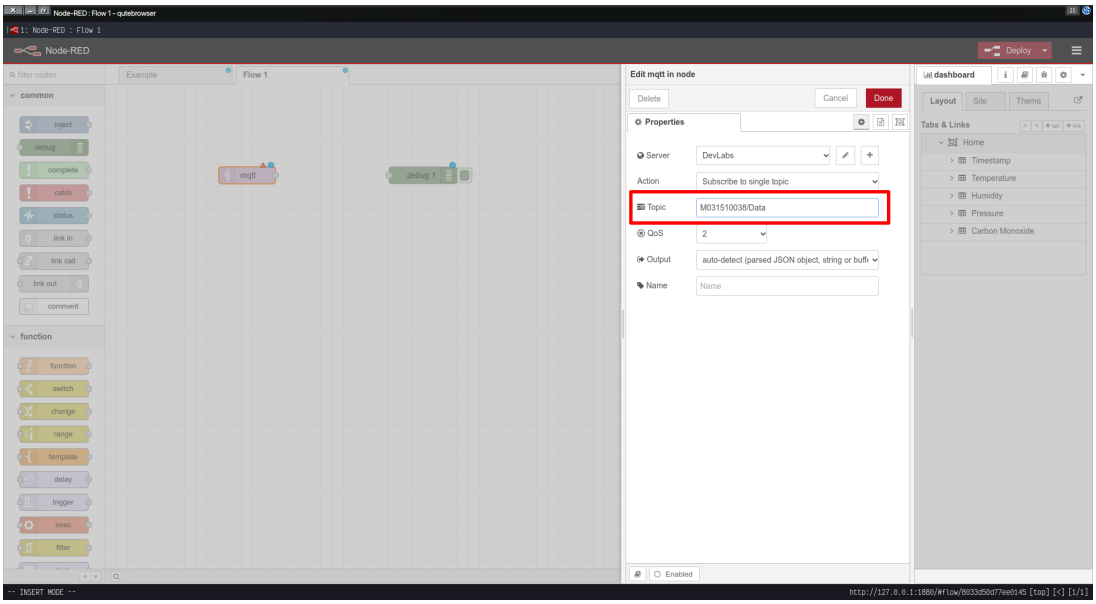
Gambar 3.18: Membuka Konfigurasi Node MQTT in

4. Klik tombol + **Tambah**, Masukkan Informasi di tab **Connection** berikut lalu klik **Add**:
   - Name : Server
   - Server : (Masukkan URL dari server yang dipilih sebelumnya)
   - Port : 1883 (Tergantung server)



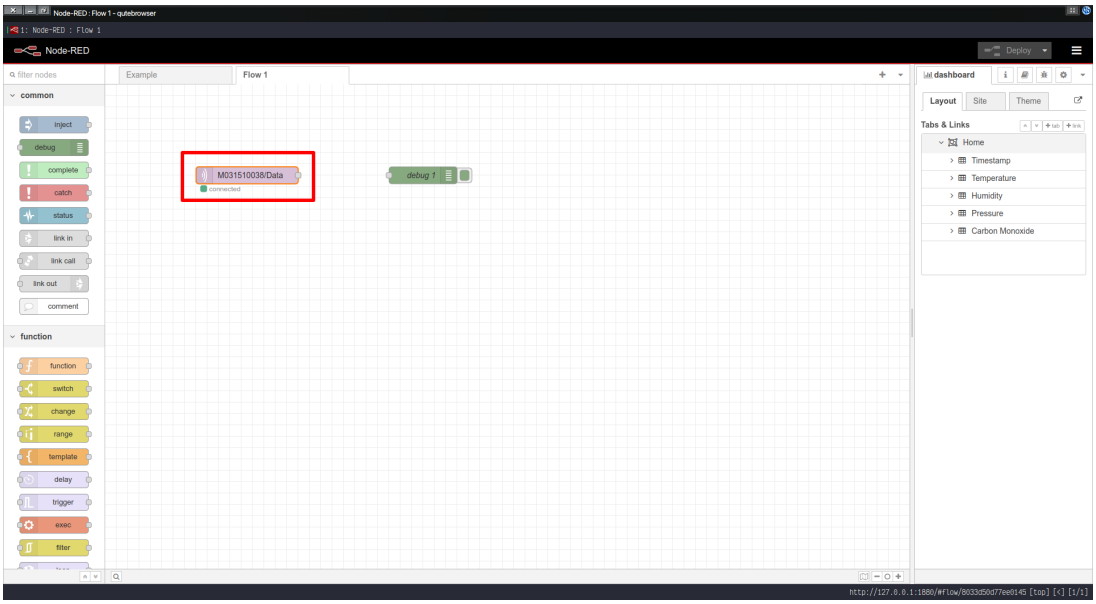
Gambar 3.19: Konfigurasi Connection MQTT

5. Setelah menambahkan server MQTT, masukkan **Topic** sesuai konfigurasi sebelumnya (NIM/Data). Lalu klik **Done**:



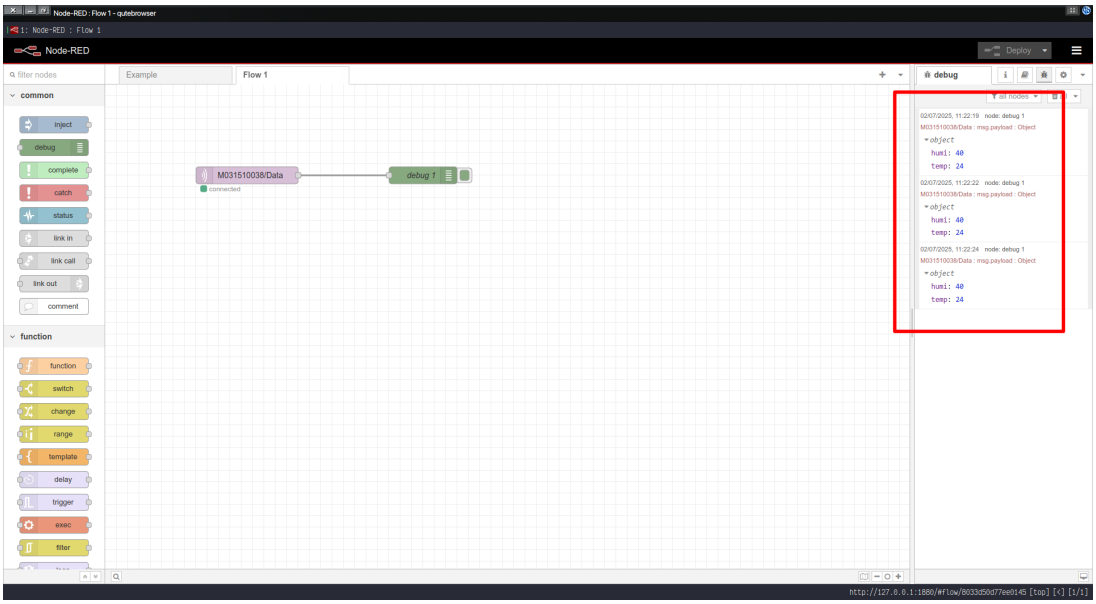
Gambar 3.20: Konfigurasi Topik MQTT

6. Klik **Deploy** untuk mengecek apabila node **mqtt in** berhasil menghubungi server. Jika berhasil akana da tulisan **Connected**



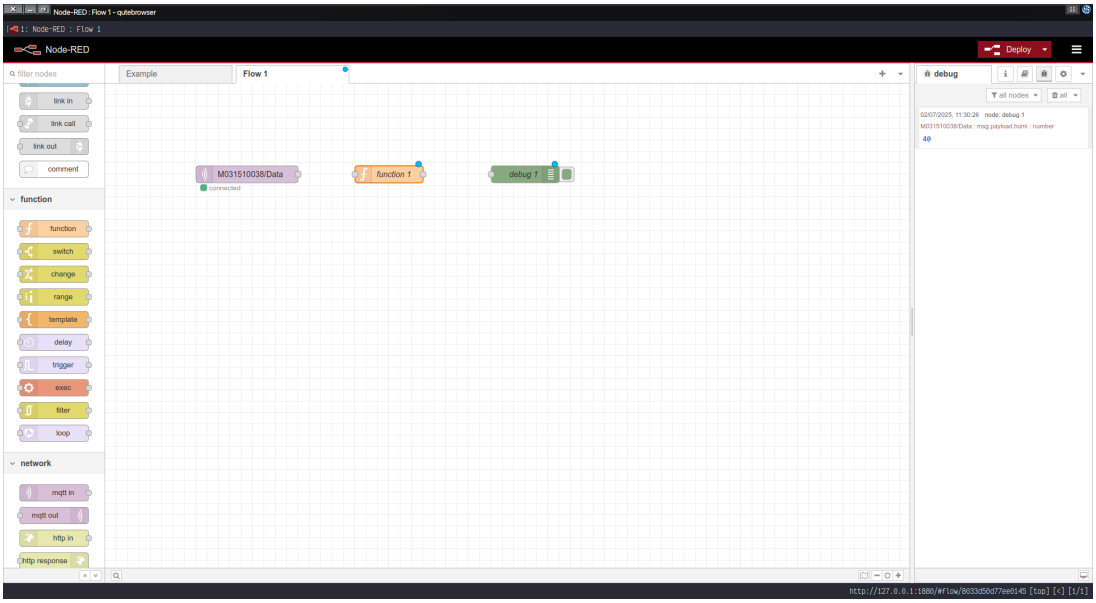
Gambar 3.21: Mengetes Koneksi ke Server

7. Hubungkan node **mqtt in** dan **debug** dan klik **Deploy** untuk menguji apakah data dari **Wokwi** bisa diterima oleh Node-RED. Jangan lupa untuk menjalankan IoT Wokwi sebelum melihat hasilnya



Gambar 3.22: Hasil data Wokwi di Node-RED

8. Hapus sambungan antara **mqtt in** dengan **debug**. Lalu masukkan node **function** dari kategori **function** i tengah-tengah keduanya



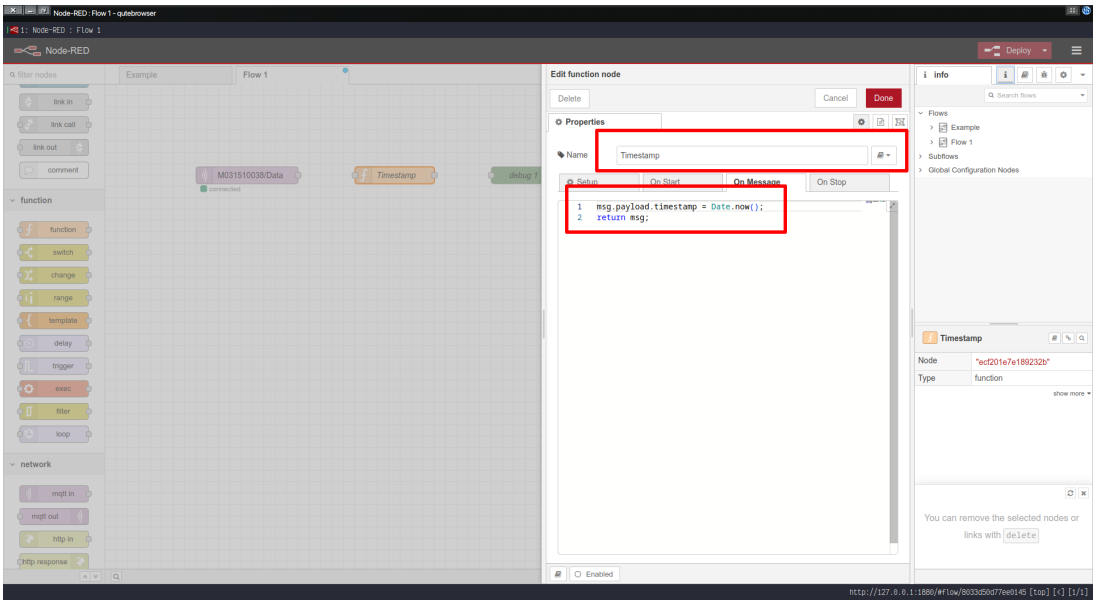
Gambar 3.23: Memasukkan Node function

9. **Double Klik** node **function** dan masukkan konfigurasi berikut:

   - Name : Timestamp

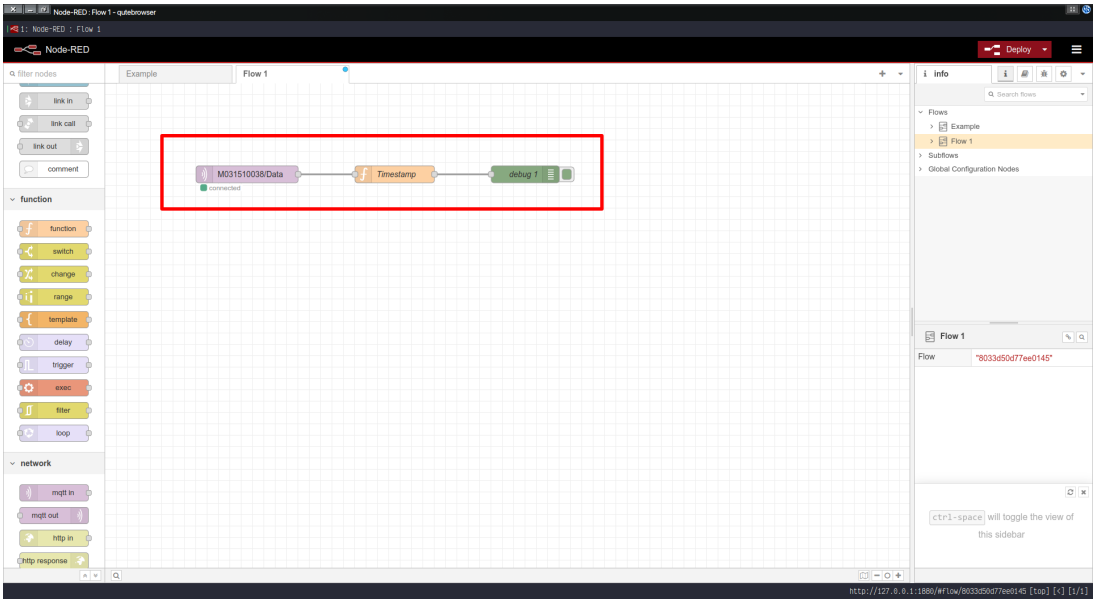
Potongan Kode

```
msg.payload.timestamp = Date.now();
return msg;
```



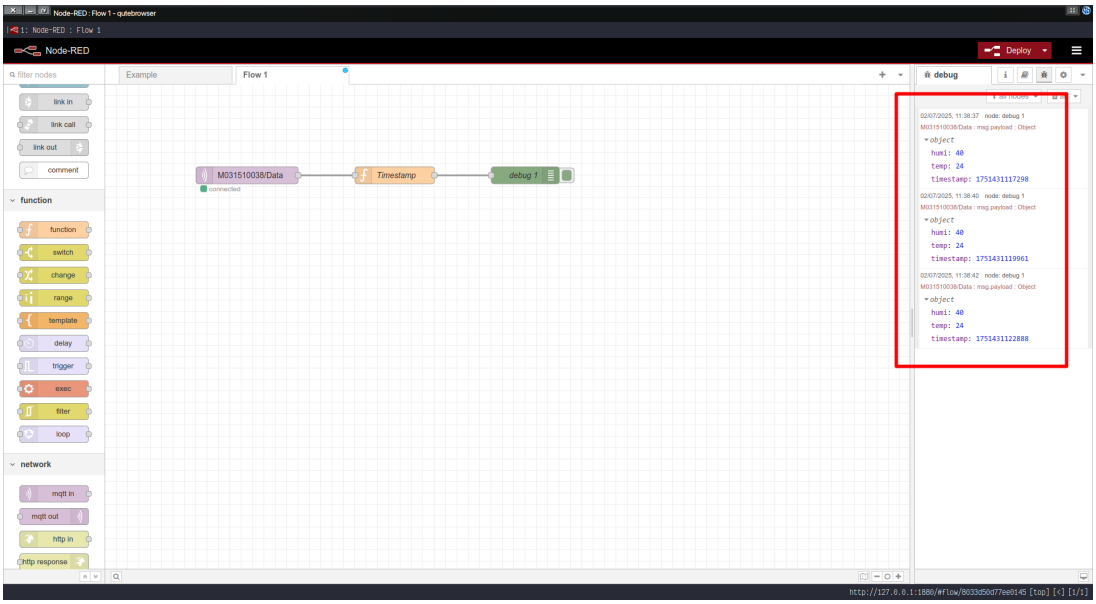
Gambar 3.24: Konfigurasi Node Function

10. Jika sudah memasukkan konfigurasi, simpan dan hubungkan masing-masing node. Lalu klik **Deploy**



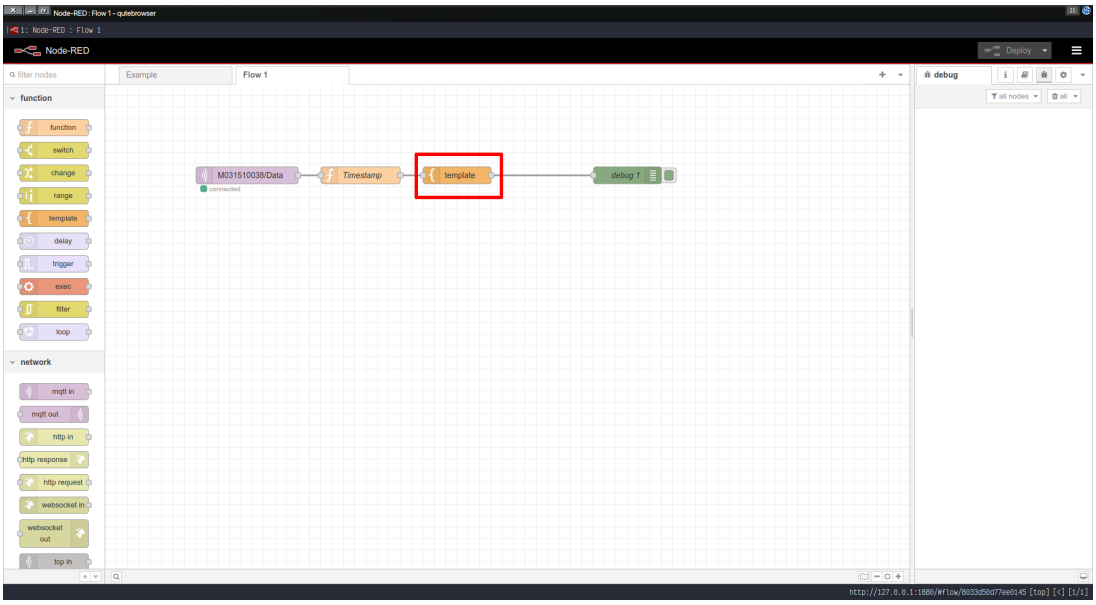
Gambar 3.25: Menghubungkan Semua Node

11. Jalankan **Wokwi** untuk melihat hasil nya. Hasil terbaru akan menampilkan data baru berupa timestamp **Data Diterima**.



Gambar 3.26: Hasil dengan Timestamp

12. Karena format yang dibawa oleh Node-RED adalah Objek Javascript, maka format ini harus dibentuk secara manual dengan menggunakan node **Template** dari kategori **function** tepat di tengah-tengah antara **Timestamp function** dan **debug**

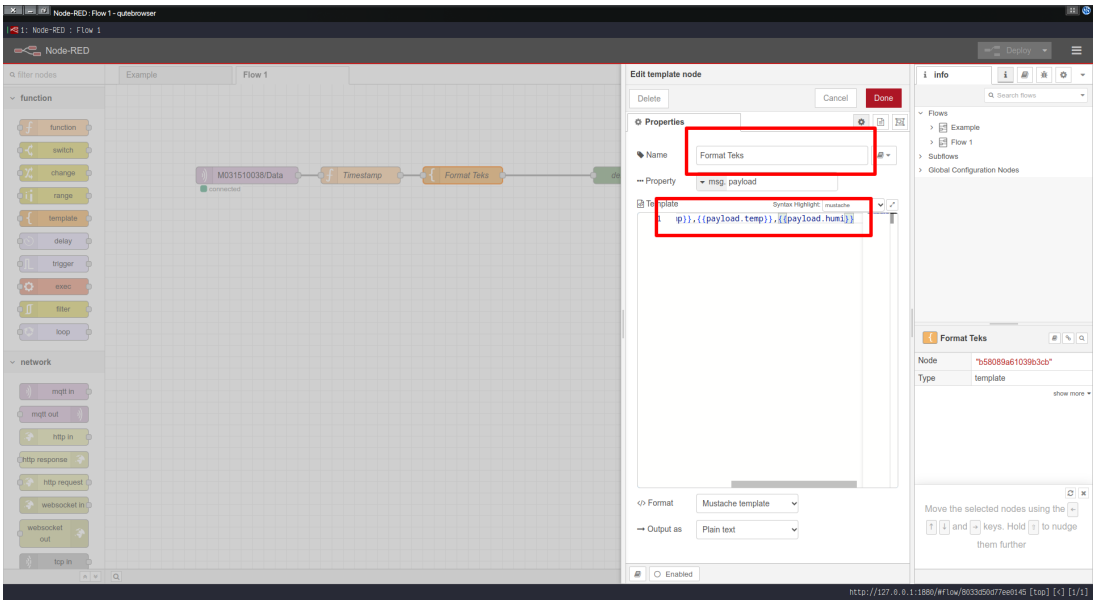


Gambar 3.27: Tambah node Template

13. Untuk bisa memformat ulang data dari MQTT, masukkan konfigurasi node berikut:
    - Name: Format Teks

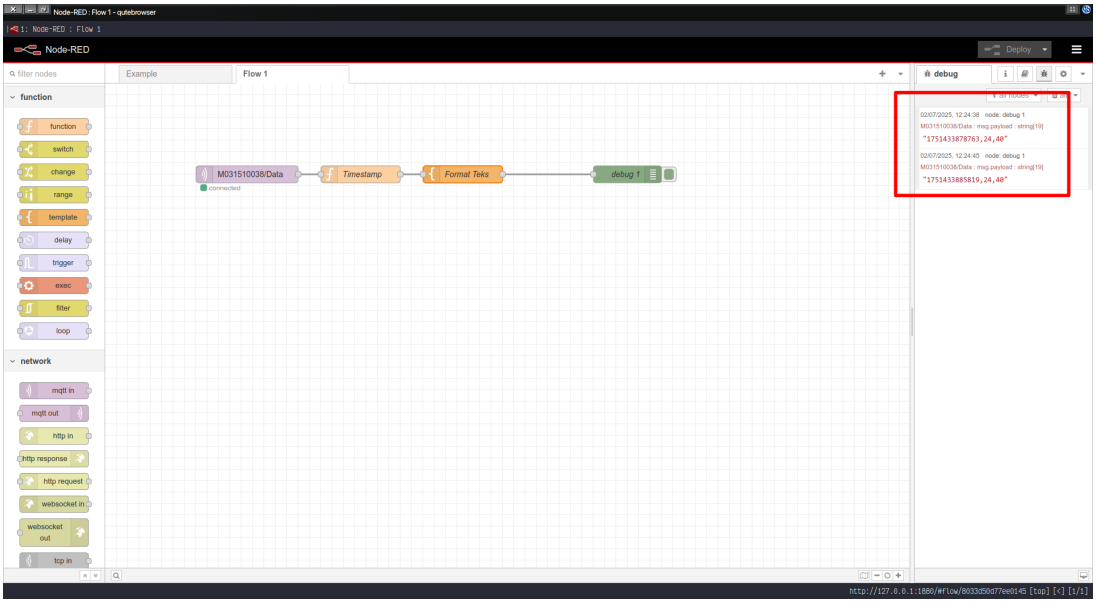
Potongan Kode

```
{{ payload.timestamp}},{{ payload.temp}},{{ payload.humi}}
```



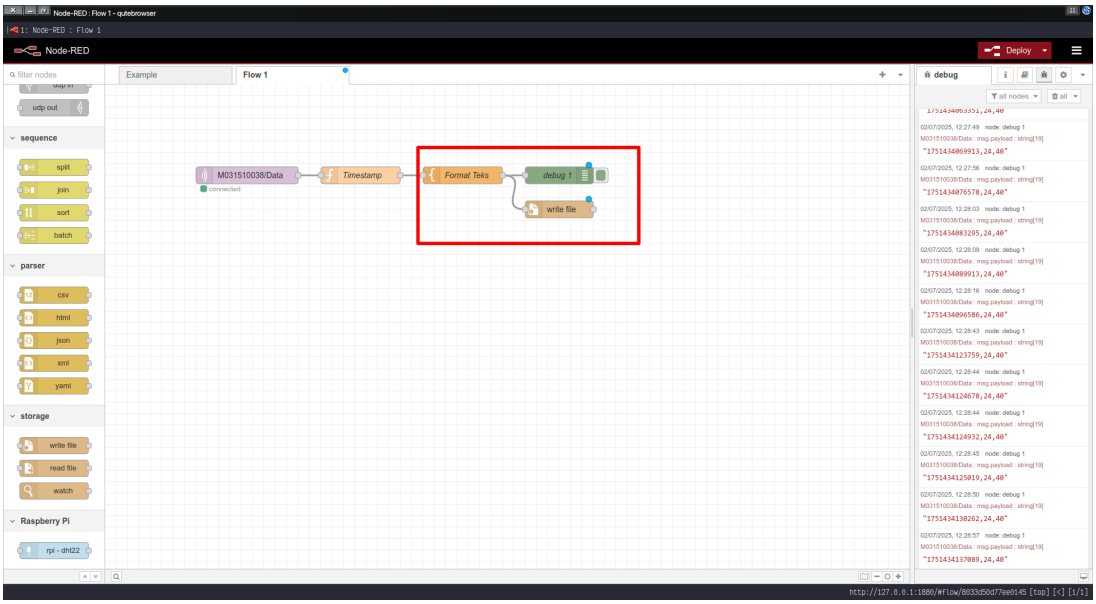
Gambar 3.28: Konfigurasi Node Template

14. Tes dengan menjalankan **Wokwi** dan lihat hasilnya



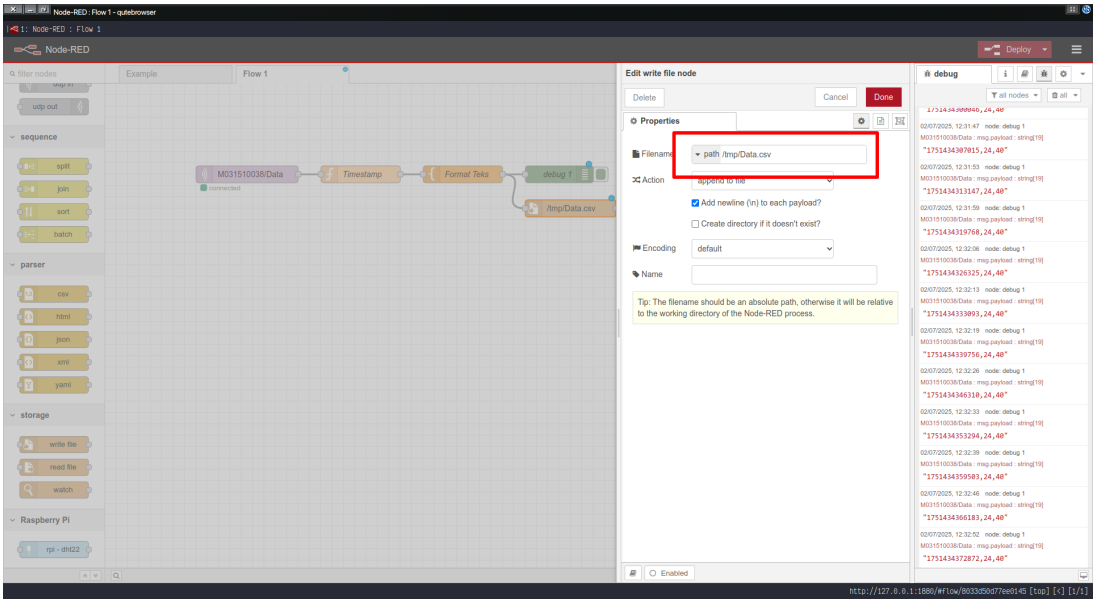
Gambar 3.29: Hasil Format Template

15. Untuk menyimpan dalam file, gunakan node **Write File** dan letakkan di bawah node **debug**, dan sambungkan dengan node **Template**



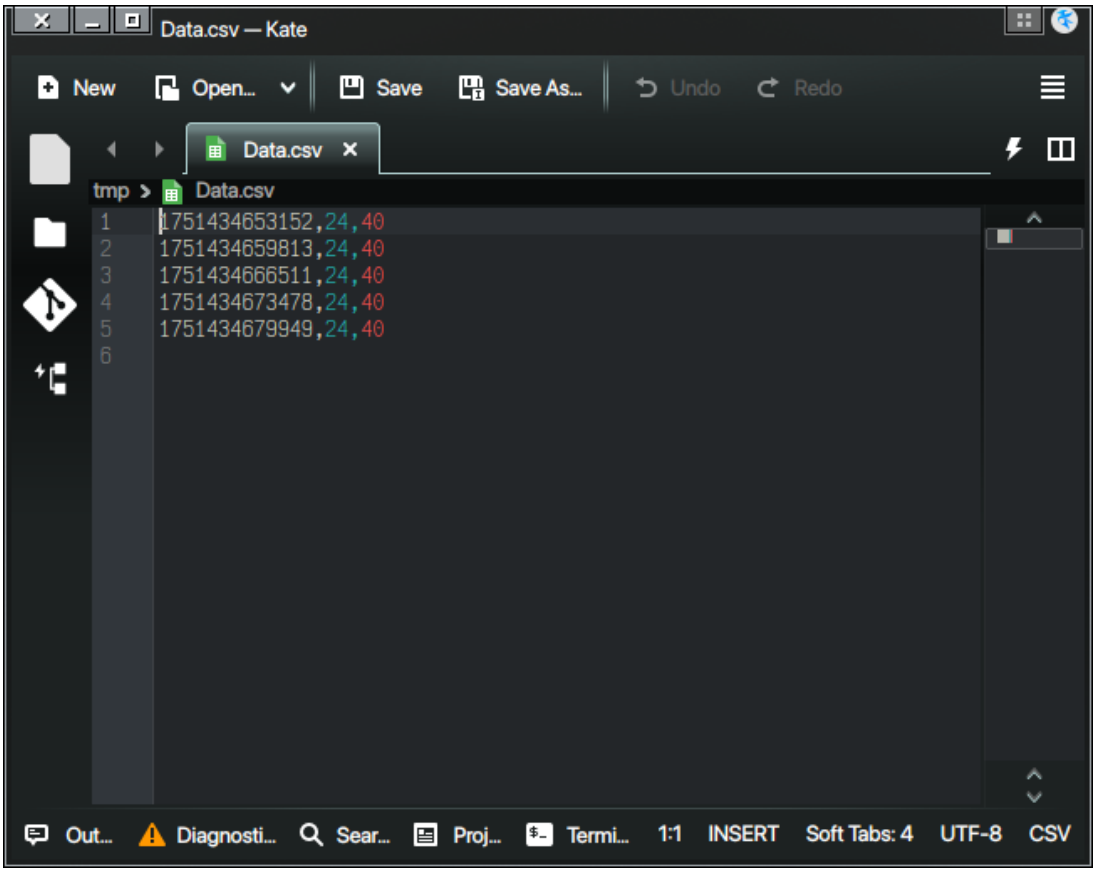
Gambar 3.30: Node Write File

16. Konfigurasikan node **write file** seperti berikut:
    - **path**
        - Windows : C:\Data.csv
        - Linux/Unix : /tmp/Data.csv



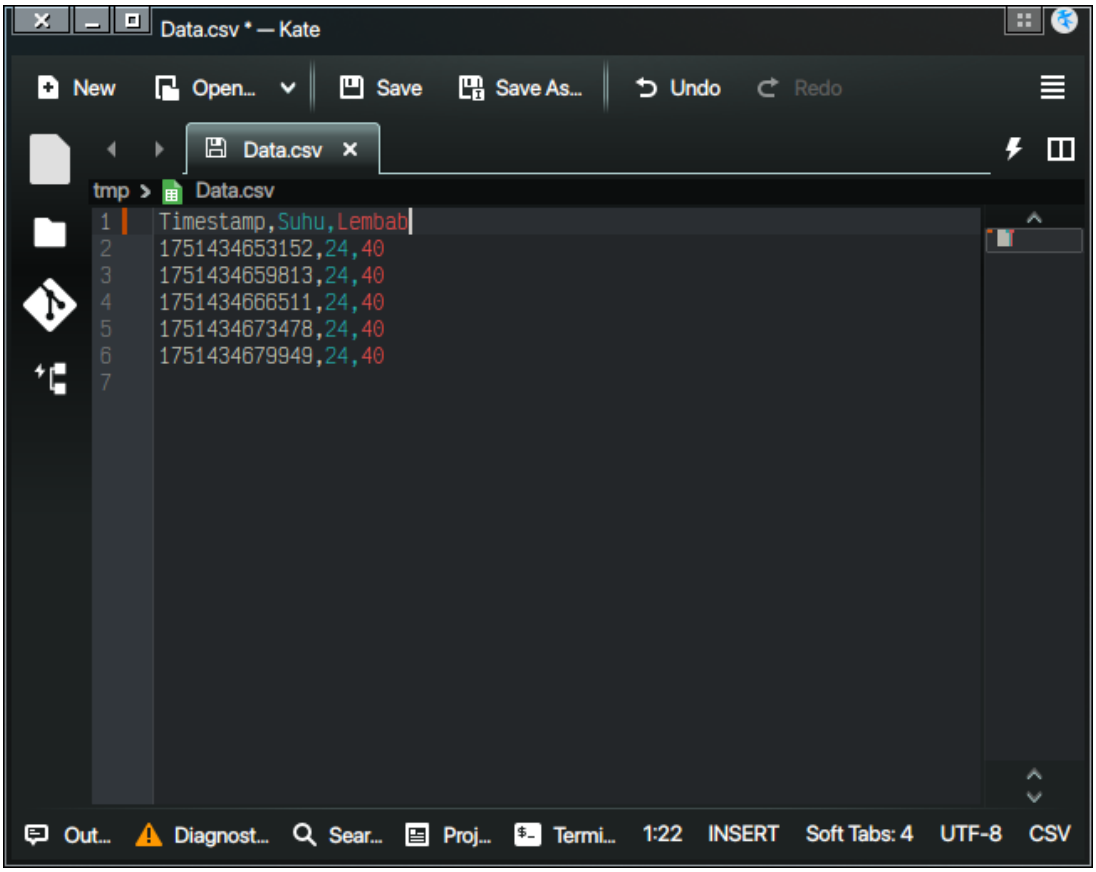
Gambar 3.31: Konfigurasi node write file

17. Klik **Done**, lalu klik **Deploy** dan tes menggunakan **Wokwi**. Tunggu beberapa saat hingga beberapa data telah ditulis dan cek file **Data.csv**

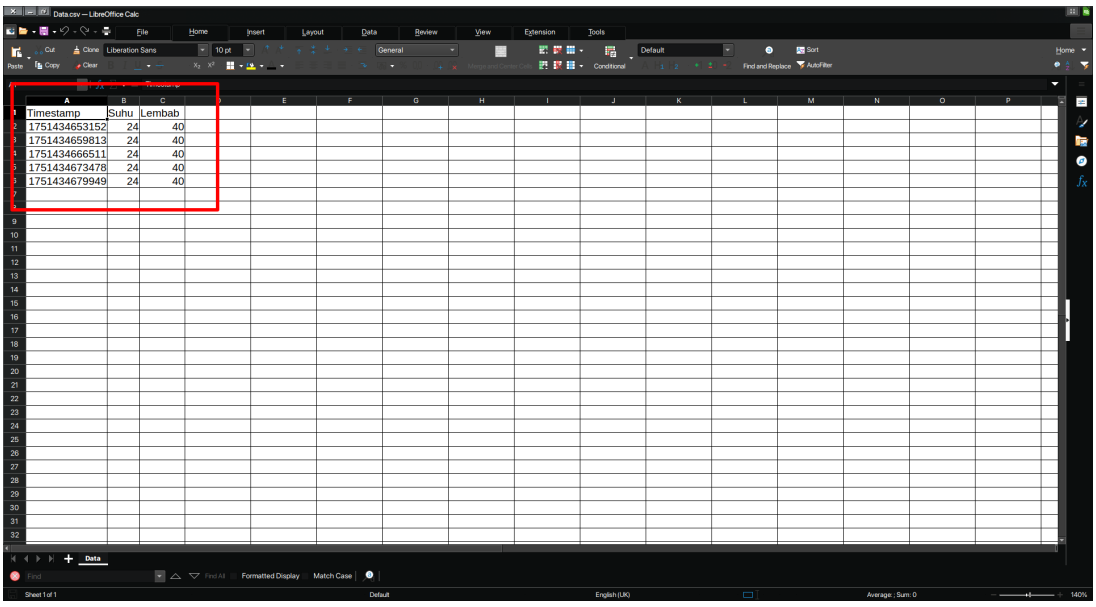


Gambar 3.32: Hasil **Data.csv**

18. Agar file CSV dapat dibuka sepenuhnya oleh **Excel**/**LibreOffice Calc**. Tambahkan header berikut **Timestamp,Suhu,Lembab** di baris paling atas. Simpan dan buka dengan Excel

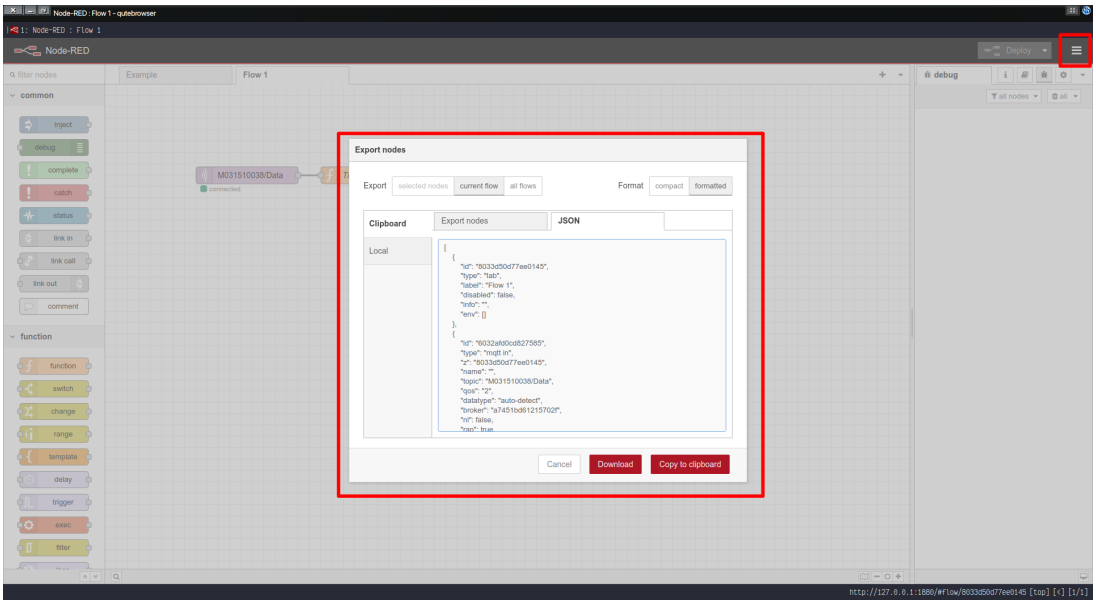


Gambar 3.33: Hasil **Data.csv** Modifikasi



Gambar 3.34: Buka **Data.csv** dengan Excel

19. Untuk menyimpan hasil yang sudah dibuat ini. Klik **Tiga Strip** yang ada di kanan atas. Pilih **Export** lalu pilih **Download**. Lalu simpan file di tempat yang aman



Gambar 3.35: Simpan Hasil Node-RED

3.4 Tugas SKPI

1. Kumpulkan banyak data dari Wokwi dan variasikan suhu dan kelembaban yang ada di Sensor.
2. Cek hasil CSV
3. Beri nama file NIM.ZIP
4. Kirim ke Google Drive : Tugas SKPI

Bab 4

Flow Programming Node-RED #2

4.1 Mengenal Dashboard Flow Programming

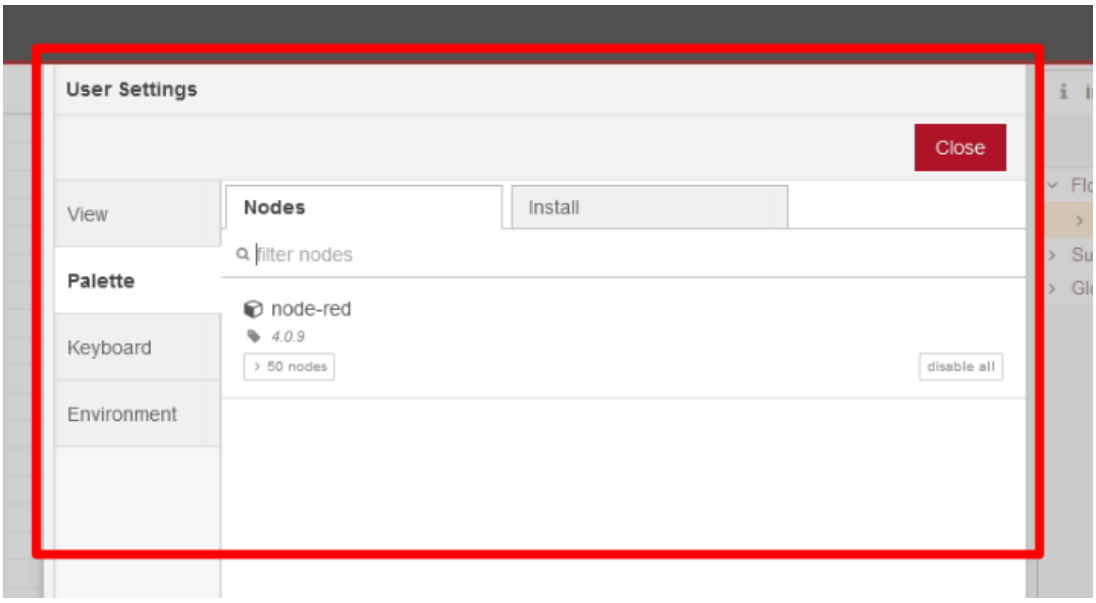
Di bagian ini mahasiswa diajarkan melakukan *Flow Programming* untuk menangkap data dari MQTT, menampilkan data dalam bentuk Dashboard

4.2 Perangkat Lunak Dibutuhkan

- NodeJS **Windows Installer (.msi)**(Download)

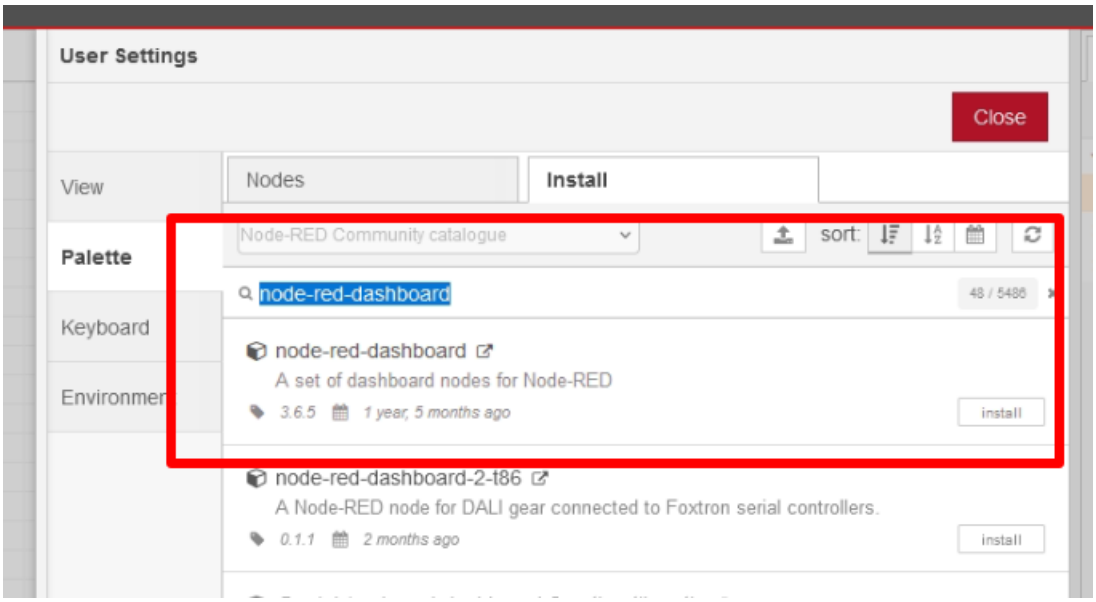
4.3 Tutorial

1. Buka kembali projek **Wokwi** lalu buka aplikasi **Node-RED** dengan menggunakan **Command Prompt**
2. Sebelum memulai pembuatan dashbor, instal terlebih dahulu **Library** tambahan dengan klik **Tiga Strip** lalu pilih **Manage Palette**



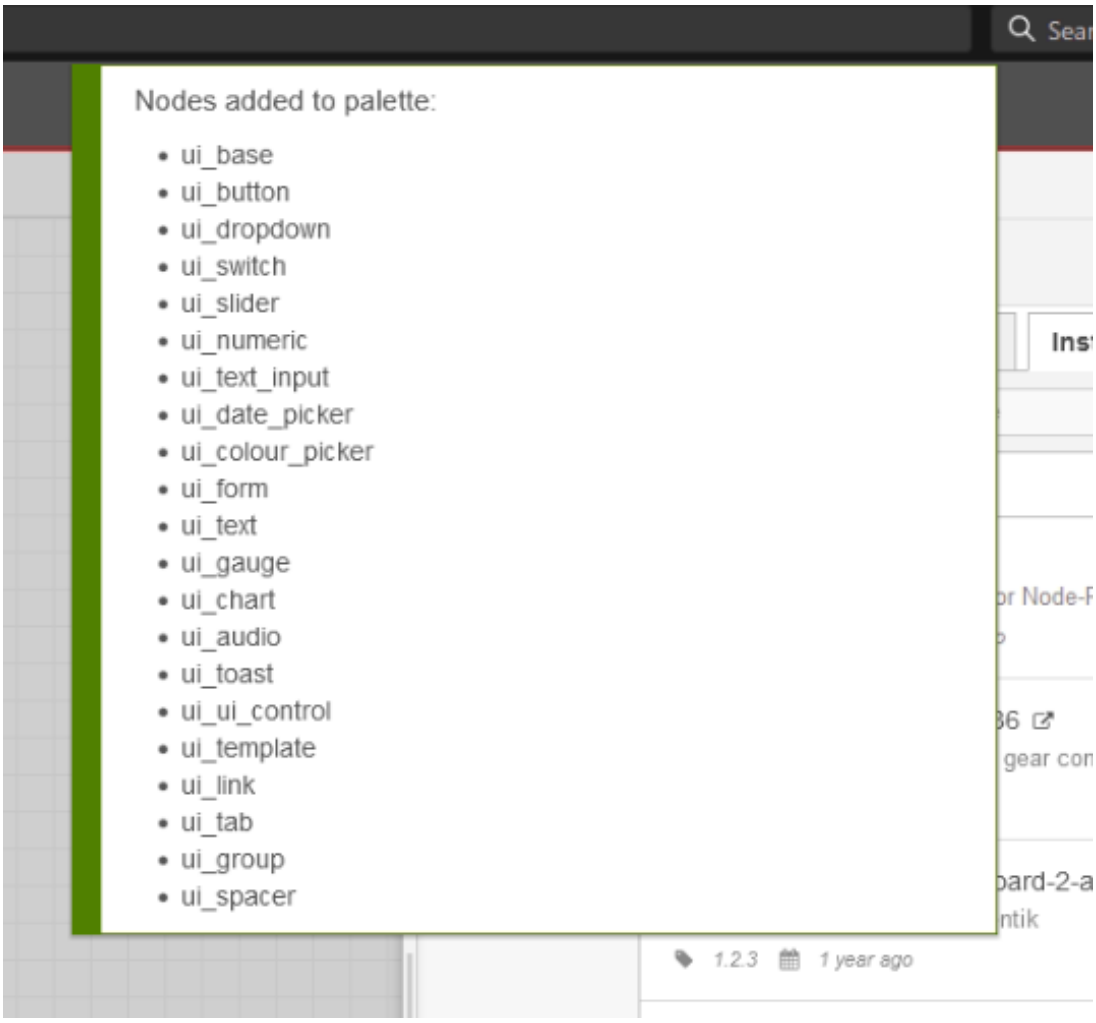
Gambar 4.1: Manajemen Palet

3. Di bagian **Install** ketik **node-red-dashboard**



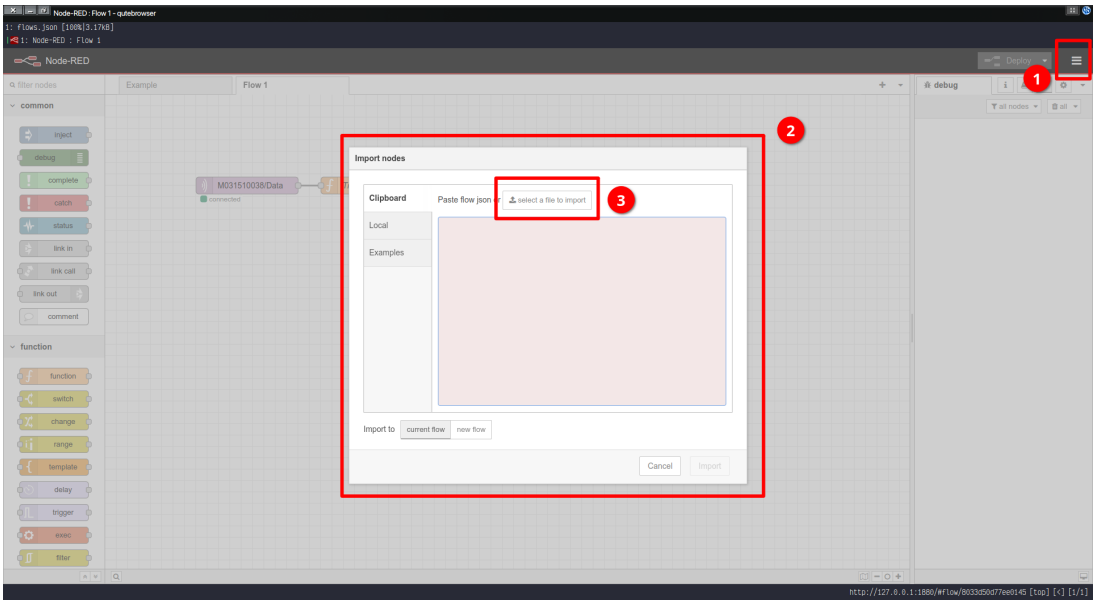
Gambar 4.2: Instalasi Dashboard

4. Klik **Install**, pilih **Install** dan tunggu hingga selesai



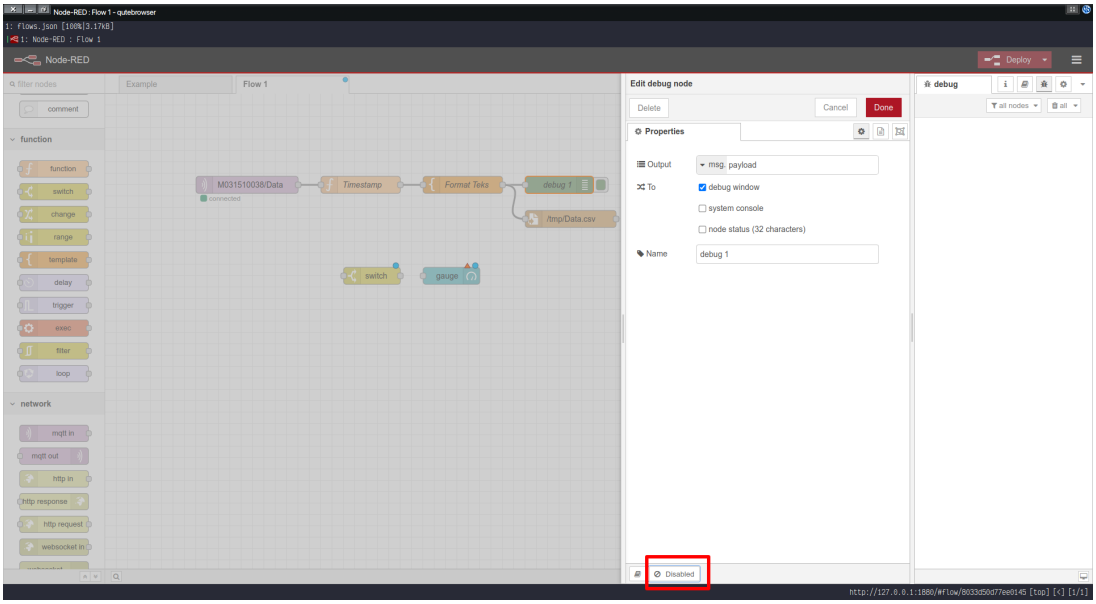
Gambar 4.3: Install node-red-dashboard

5. Untuk membuka projek yang sudah dibuat, klik **Tiga Strip** lalu pilih **Import**. Di window nya, pilih **Select a file to import**



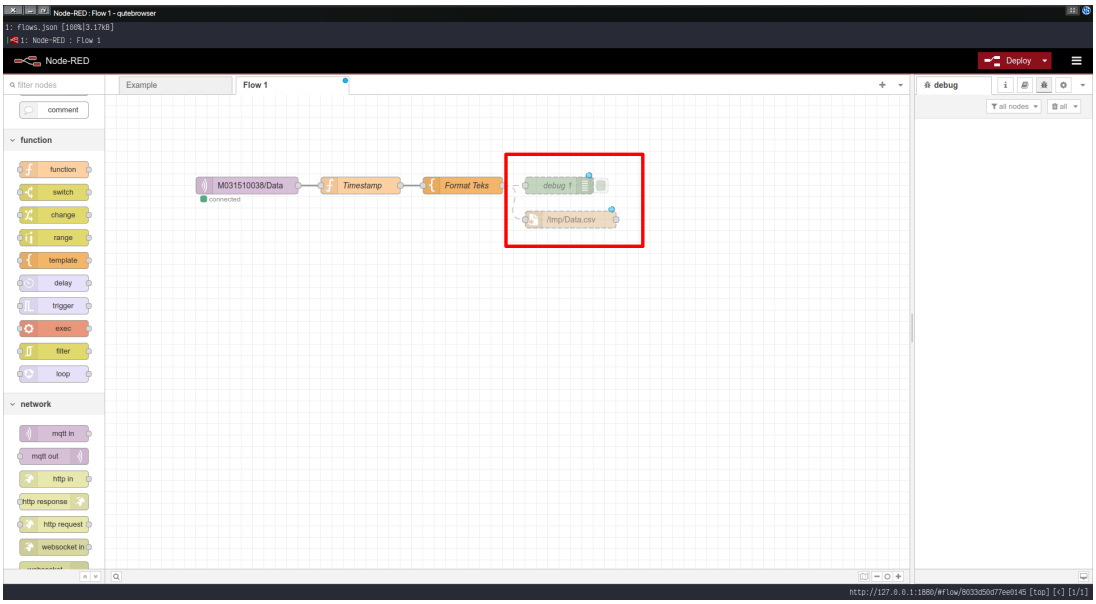
Gambar 4.4: Import Proyek

6. Agar node **write file** dan **debug** tidak mengganggu, maka perlu di **disable** terlebih dahulu. **Double Click** secara bergantian **write file** dan **debug**. Lalu klik **enable** menjadi **disable**



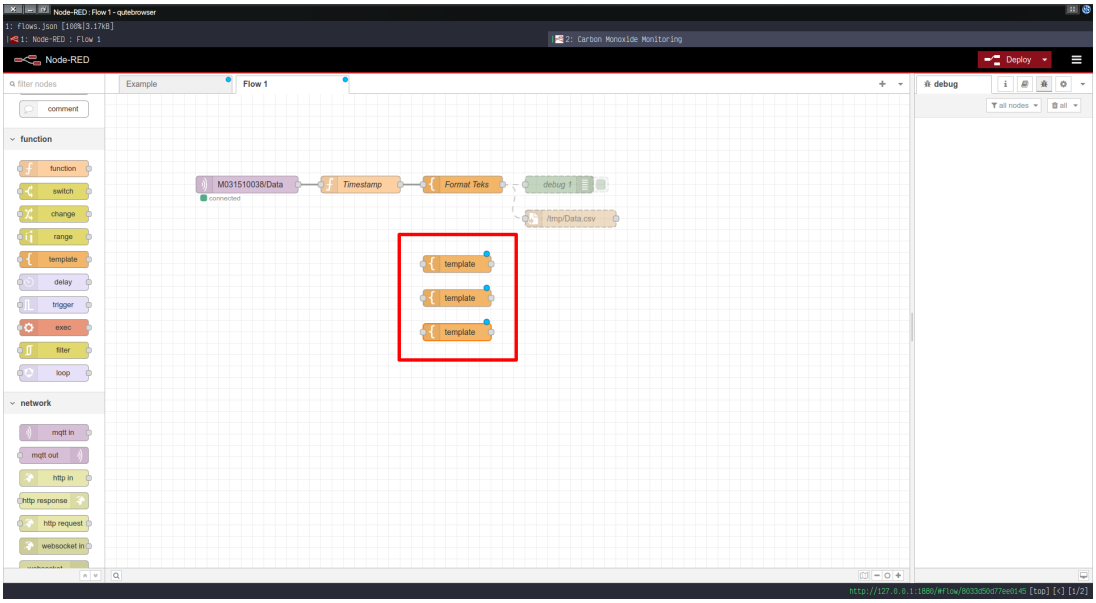
Gambar 4.5: Disable Node

7. Setelah dimatikan, kedua node akan menjadi transparan



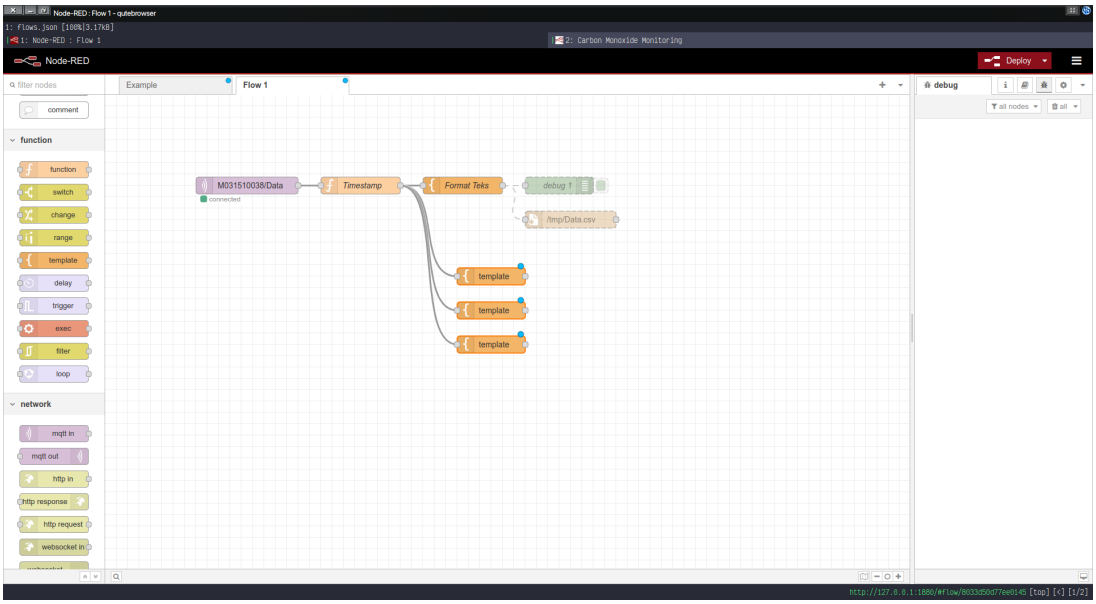
Gambar 4.6: Hasil disable node

8. Masukkan 3 node **Template** seperti berikut



Gambar 4.7: Memasukkan node Template

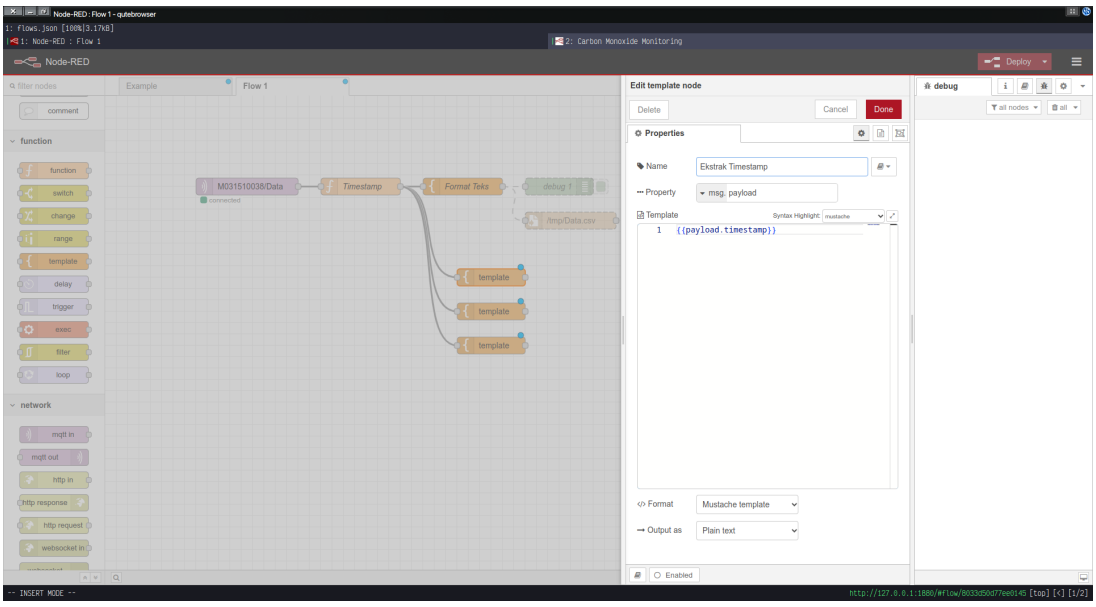
9. Hubungkan node **Timestamp** dengan **Template** tersebut



Gambar 4.8: Menghubungkan ke node Template

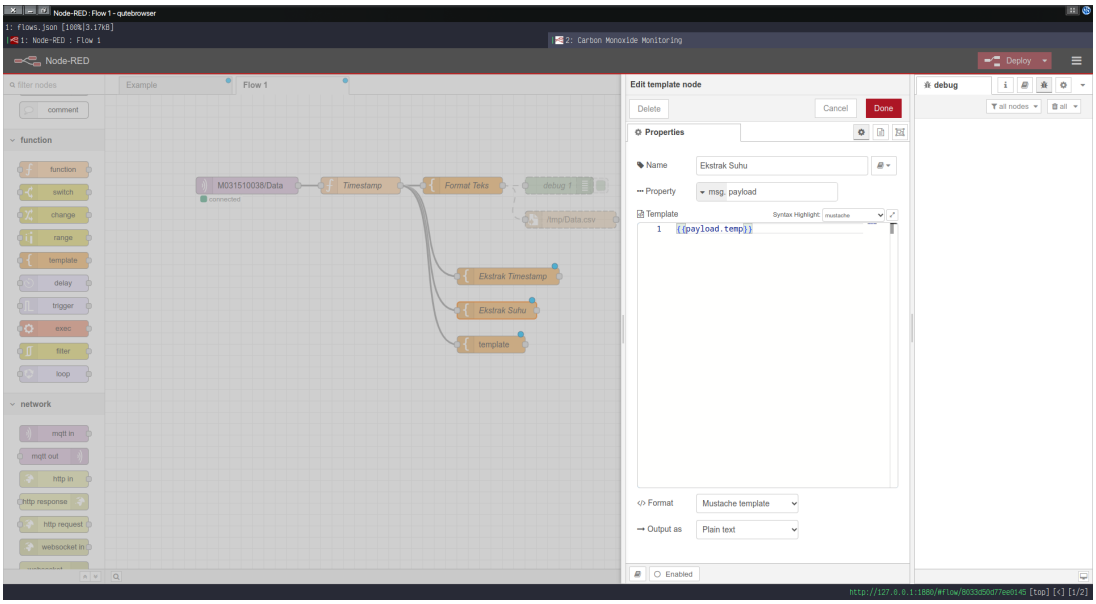
10. Konfigurasikan masing-masing node **Template** untuk mengekstrak masing-masing data seperti **Timestamp**, **Suhu**, dan **Kelembaban**. Berikut konfigurasi tiap node **Template**

    - Template : Timestamp
        - Name : Ekstrak Timestamp
        - Property : {{payload.timestamp}}



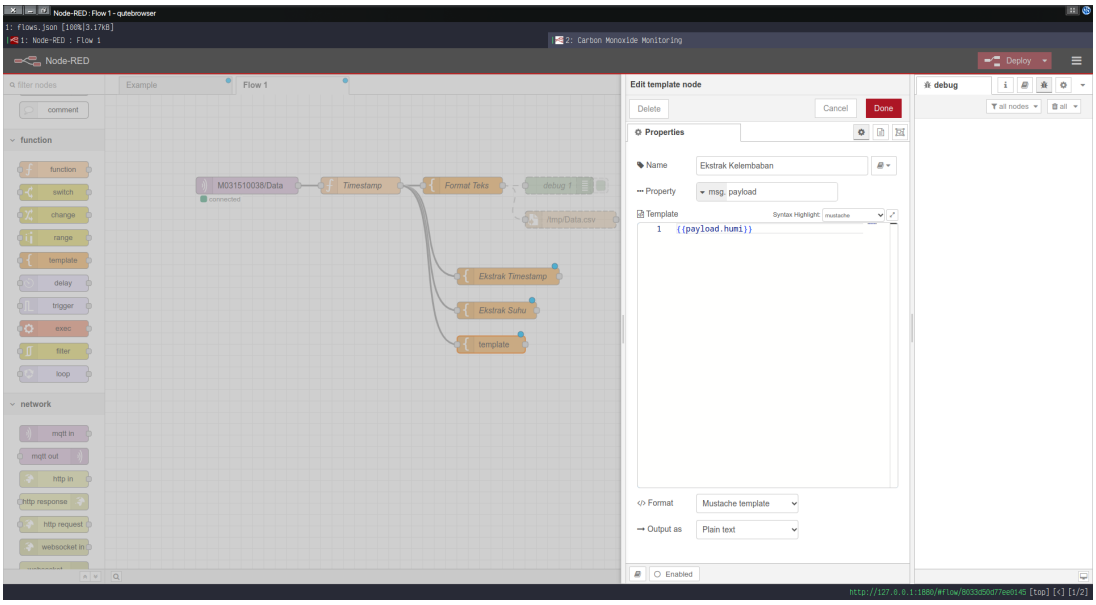
Gambar 4.9: Konfigurasi Template Timestamp

    - Template : Suhu
        - Name : Ekstrak Suhu
        - Property : {{payload.temp}}



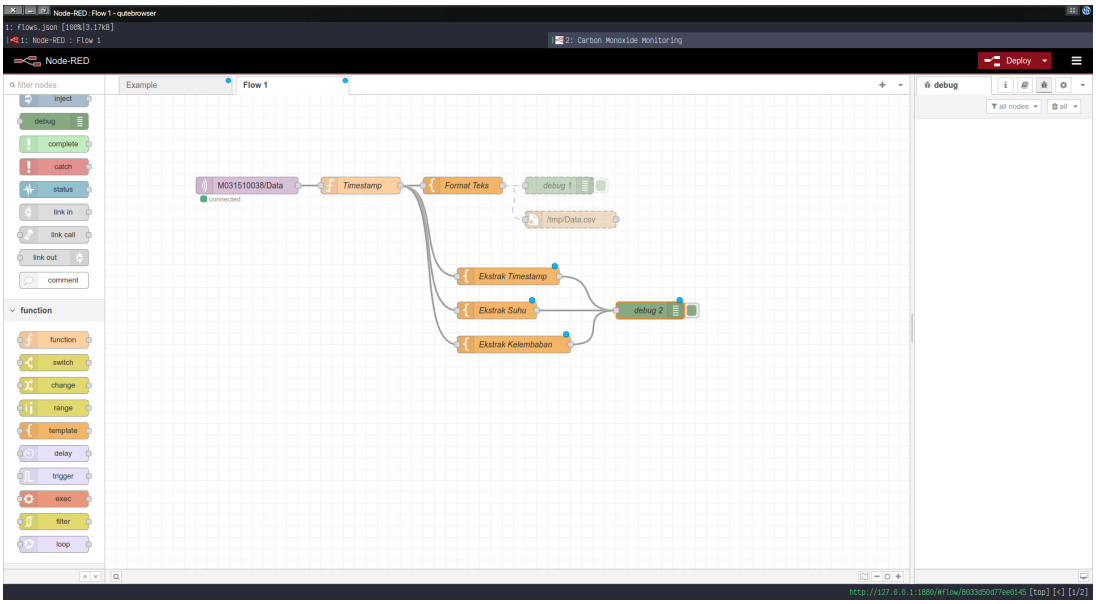
Gambar 4.10: Konfigurasi Template Suhu

- Template : Suhu
  - Name : Ekstrak Kelembaban
  - Property : {{payload.humi}}



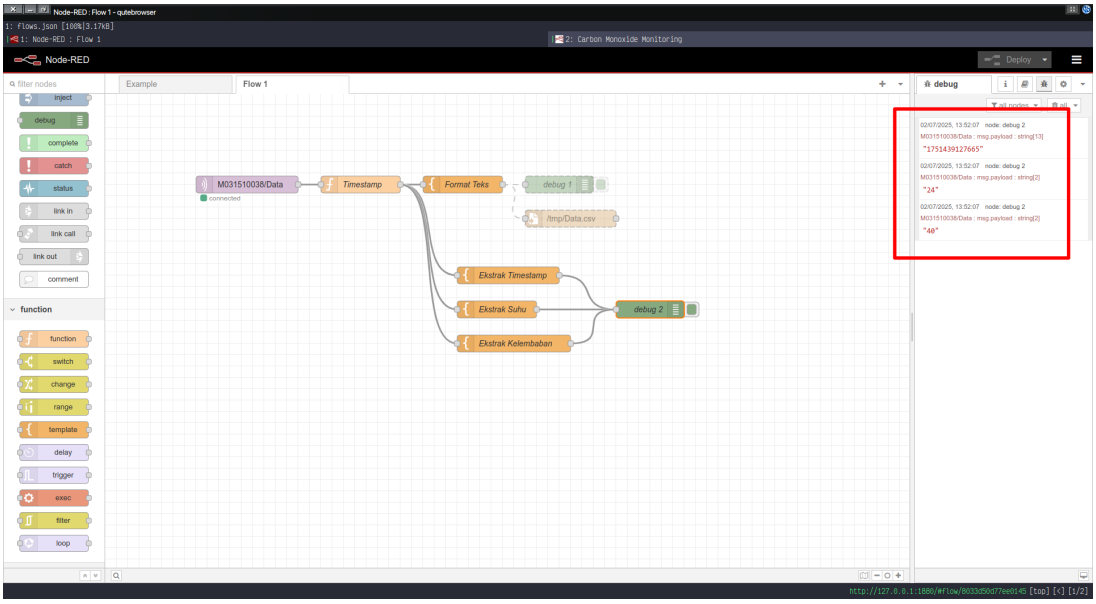
Gambar 4.11: Konfigurasi Template Kelembaban

11. Tambahkan node **debug** baru dan hubungkan ke semua template node baru untuk mengecek *output* dari ketiga node



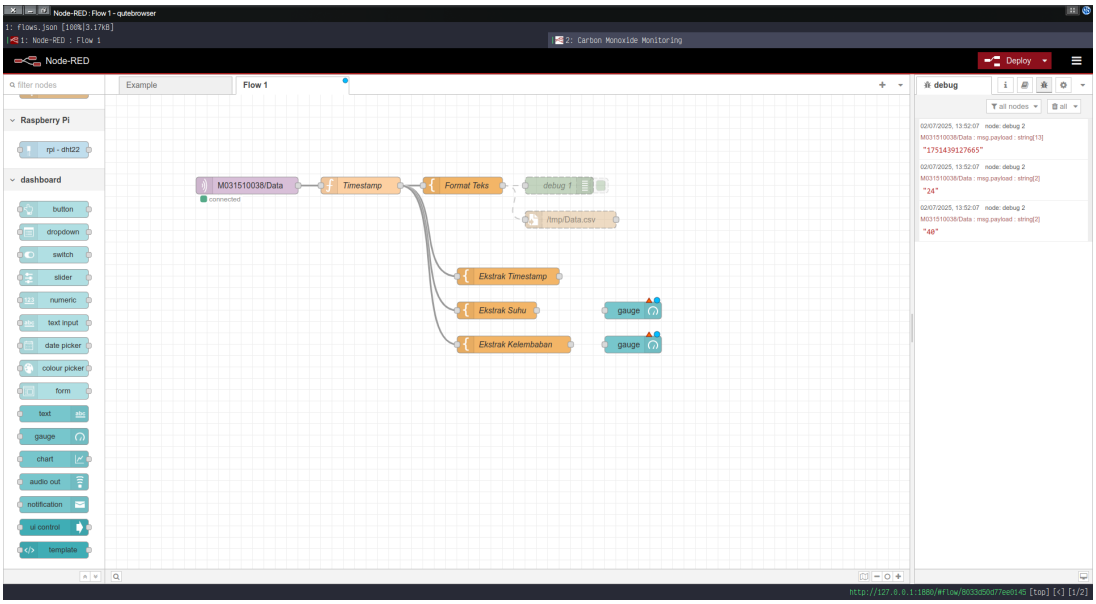
Gambar 4.12: Tambah node debug

12. Klik **Deploy** dan jalankan **Wokwi** untuk mengecek pengiriman data. Jika muncul 3 nilai, maka template node sukses.



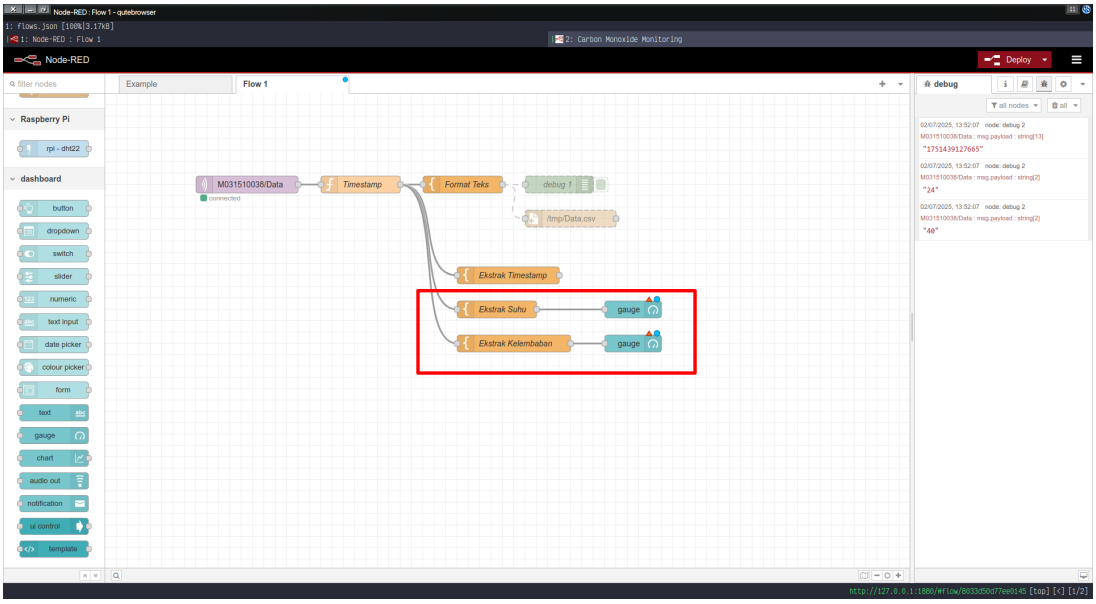
Gambar 4.13: Hasil ketiga node

13. Hapus node **debug** baru tersebut, dan tambahkan 2 node **Gauge** dari kategori **Dashboard**



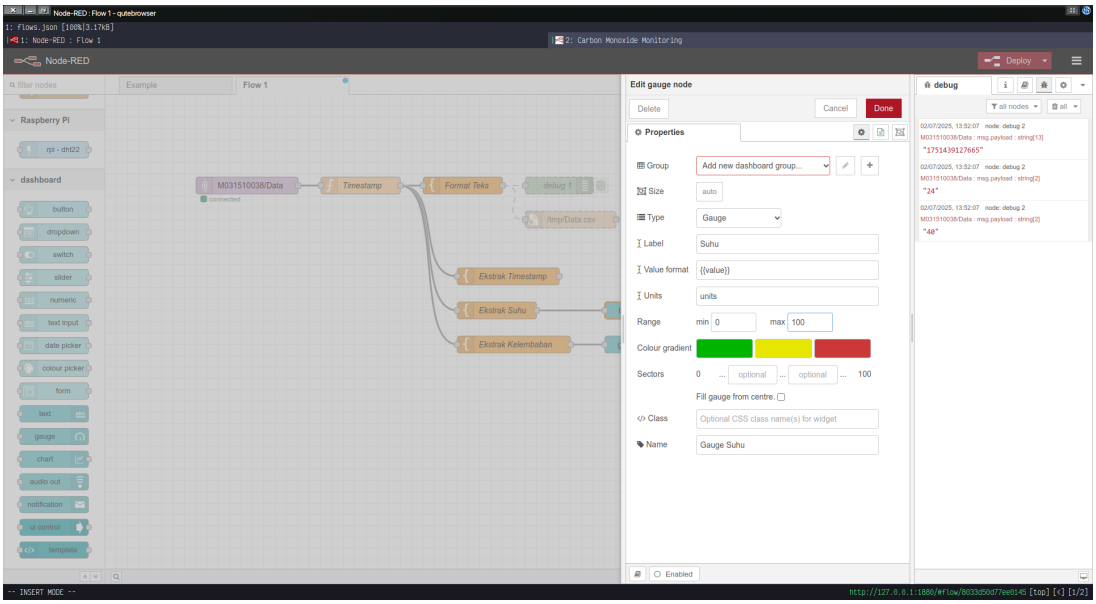
Gambar 4.14: Tambahkan node Gauge

14. Hubungkan node **Template Ekstrak Suhu** dan **Template Ekstrak Kelembaban** dengan masing-masing Gauge nya



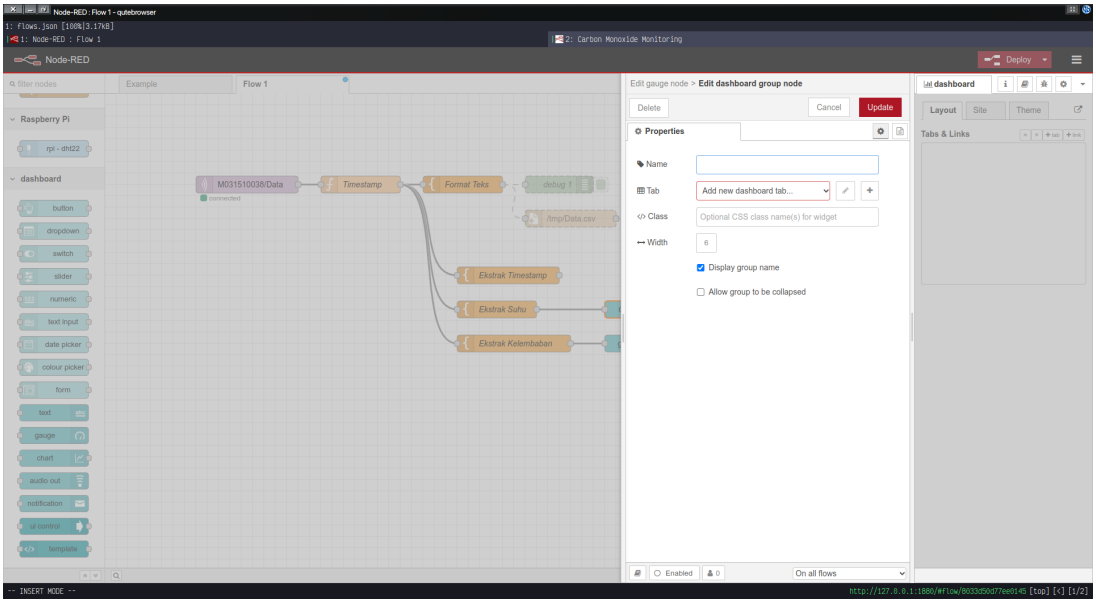
Gambar 4.15: Menghubungkan node Gauge

15. Konfigurasi node **Gauge Suhu** dengan konfigurasi berikut:
    - Label : Suhu
    - Min : 0
    - Max : 100
    - Name : Gauge Suhu



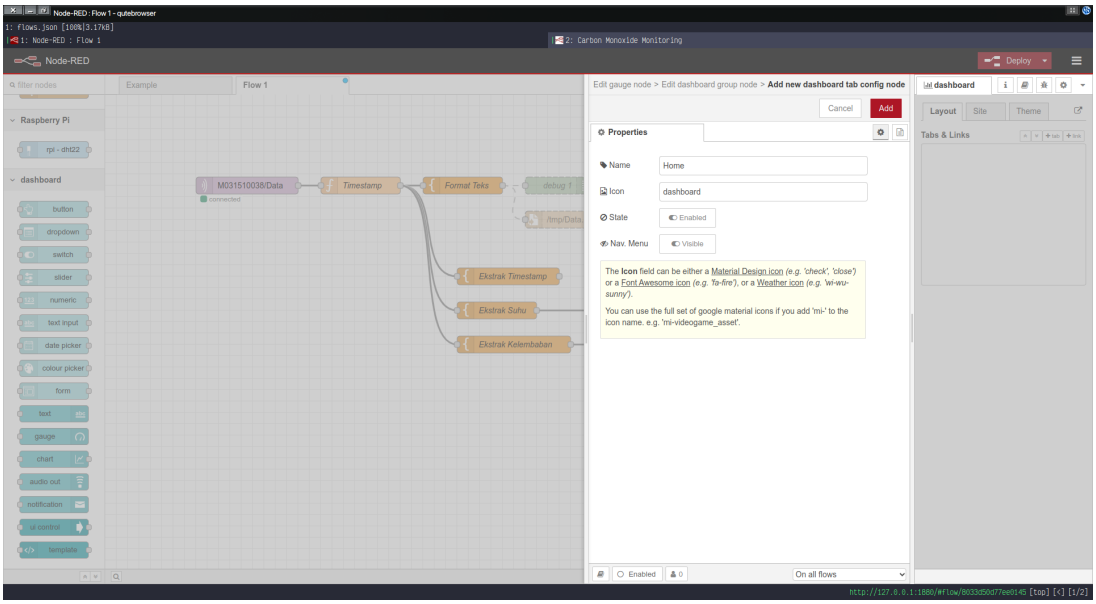
Gambar 4.16: Konfigurasi Gauge Suhu

16. Agar tertata dengan baik, di bagian **Group** klik + **Tambah**.



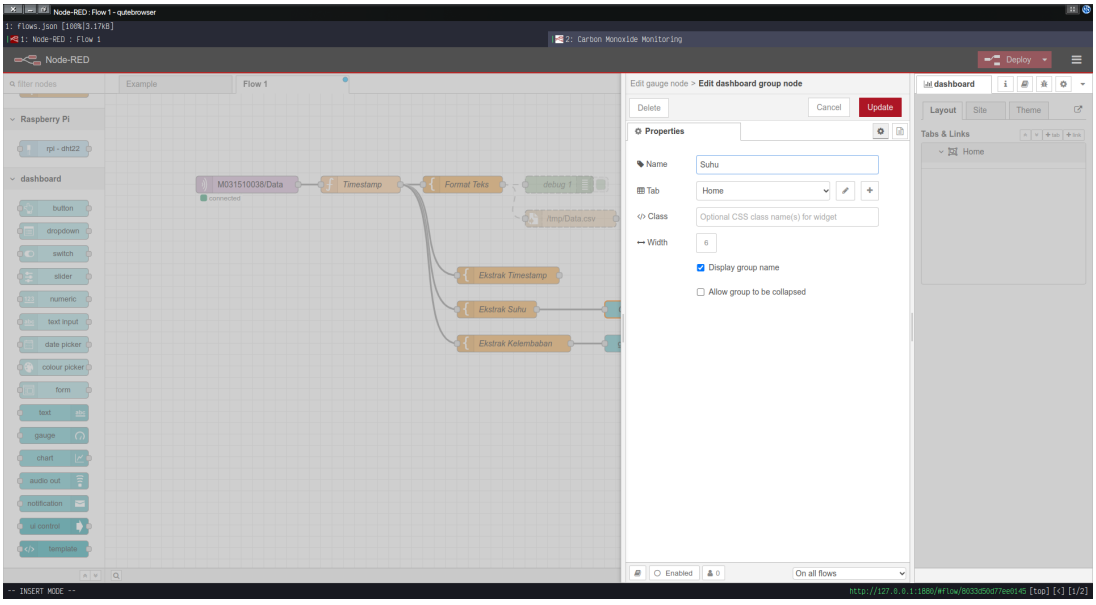
Gambar 4.17: Membuat Group baru

17. Di bagian **Tab** klik + **Tambah** lalu masukkan **Name : Home** lalu Klik **Add**



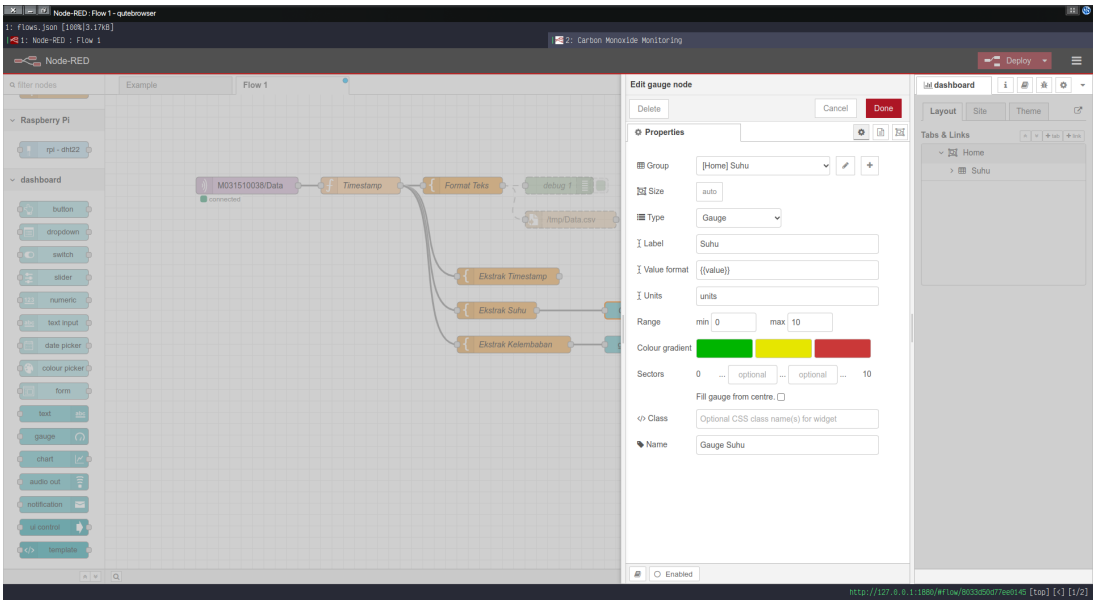
Gambar 4.18: Membuat Tab Home

18. Secara otomatis **Node-RED** akan kembali ke konfigurasi **Group** dan memilih Tab **Home**. Masukkan **Name : Suhu** dan klik **Add/Update**



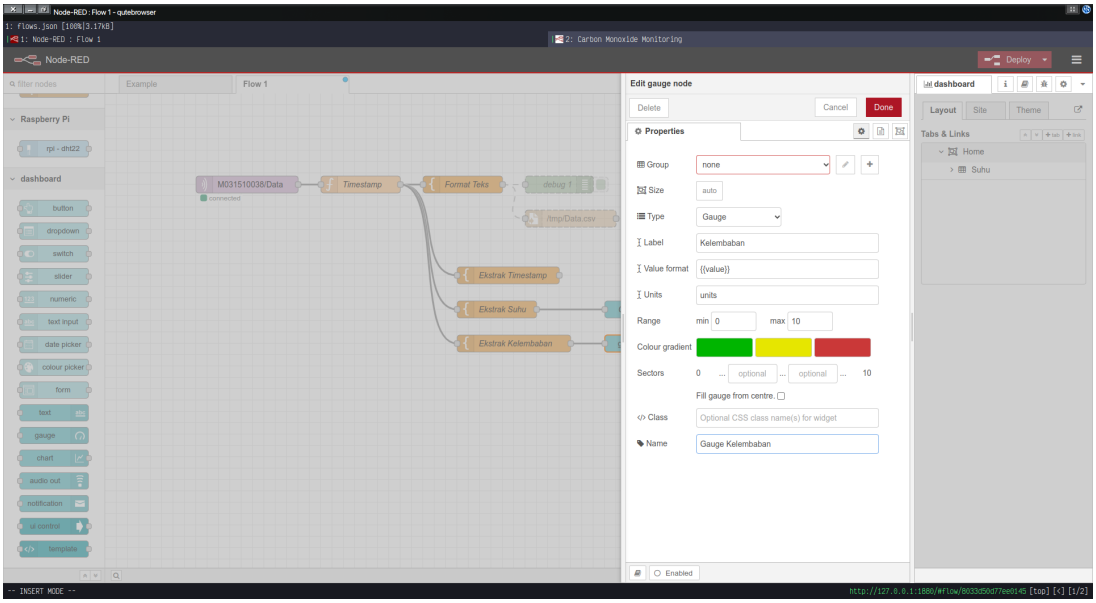
Gambar 4.19: Membuat Group Suhu

19. **Node-RED** akan kembali ke konfigurasi **Gauge** dengan Group terpilih otomatis. Klik **Done**



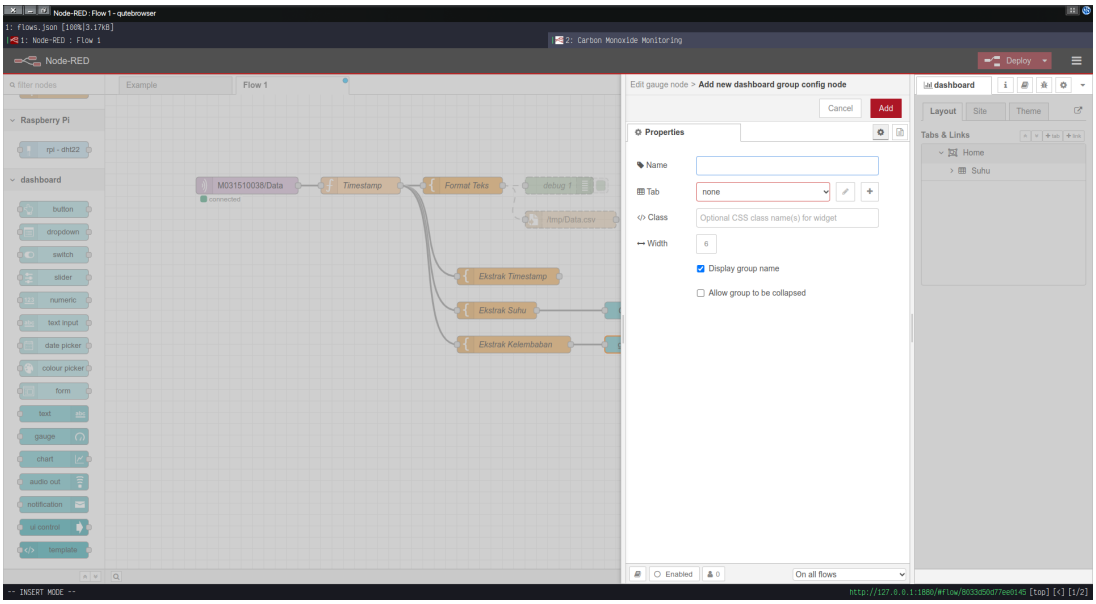
Gambar 4.20: Konfigurasi Gauge Suhu

20. **Double Click** Gauge kedua, dan berikan konfigurasikan seperti berikut:
    - Label : Kelembaban
    - Min : 0
    - Max : 100
    - Name : Gauge Kelembaban



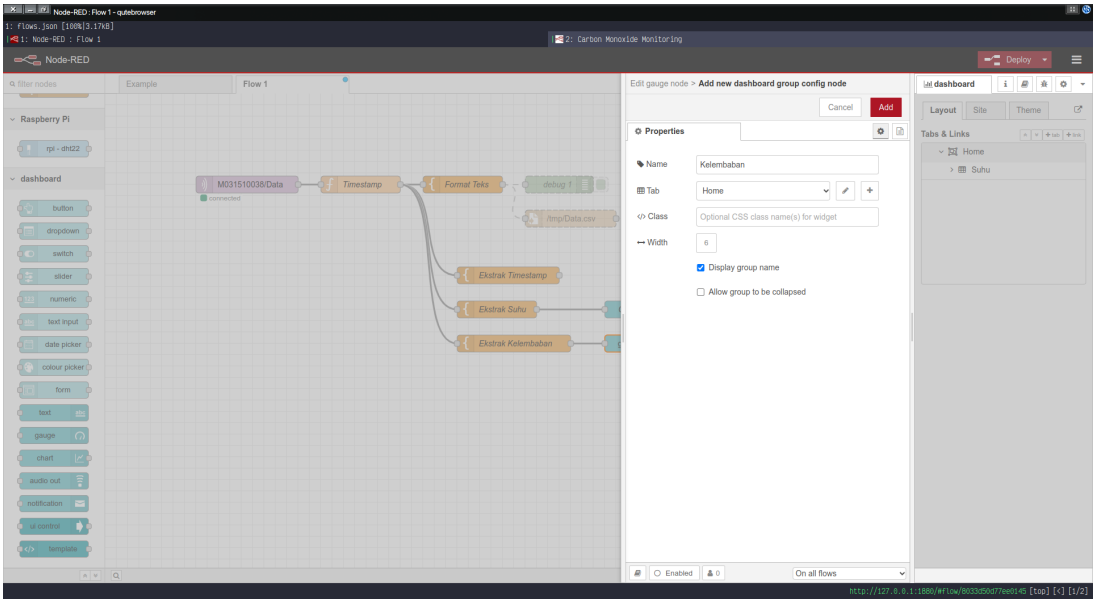
Gambar 4.21: Konfigurasi Gauge Kelembaban

21. Untuk kelembaban memerlukan **Group** yang berbeda. Klik + **Tambah** dan Node-RED menampilkan window baru



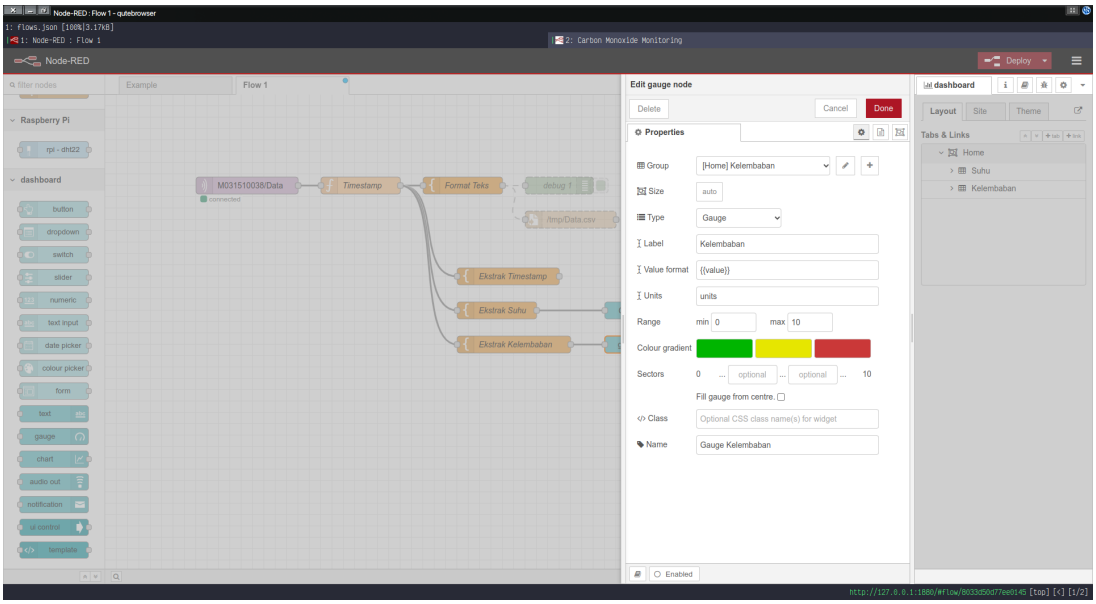
Gambar 4.22: Konfigurasi Group Gauge Kelembaban

22. Ubah Tab ke **Home**, lalu masukkan **Name : Kelembaban**



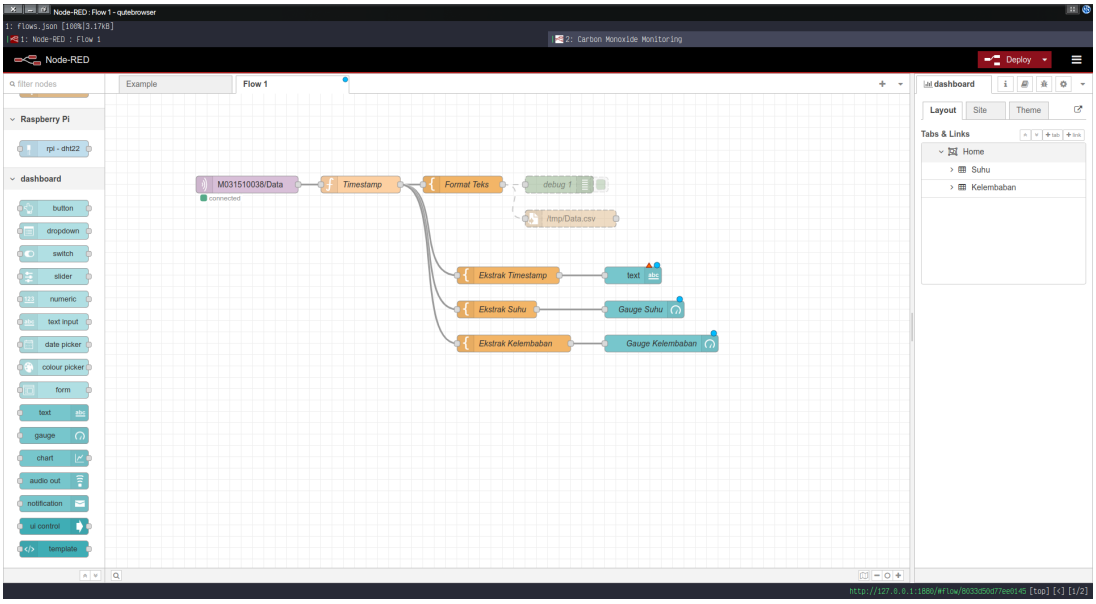
Gambar 4.23: Konfigurasi Group Gauge Kelembaban

23. Konfigurasi Group sudah lengkap untuk Gauge Kelembaban. Selesaikan dengan klik **Done**



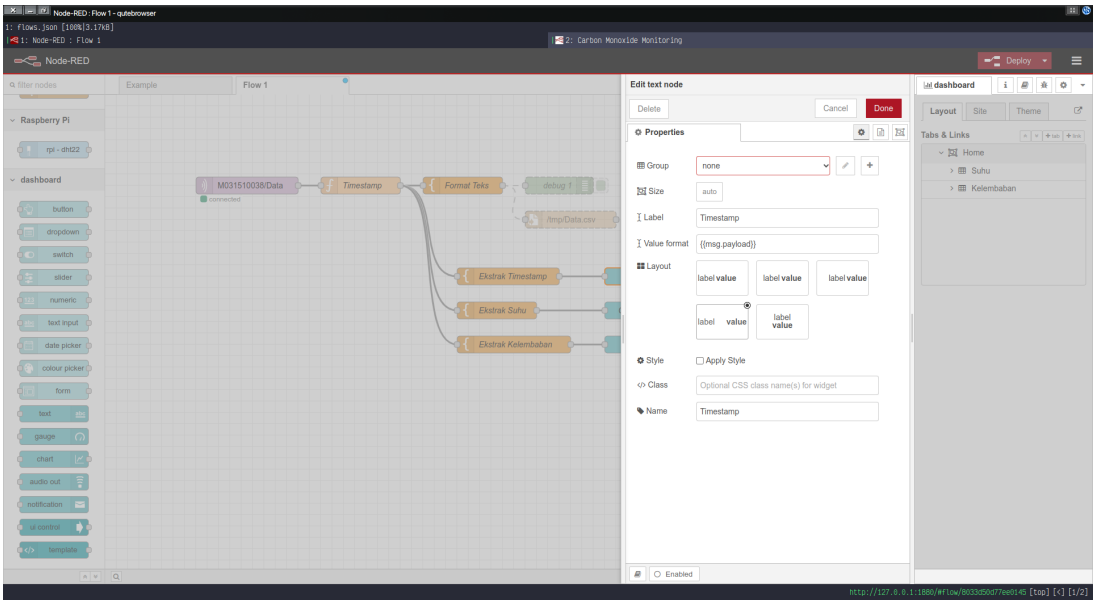
Gambar 4.24: Konfigurasi Gauge Kelembaban

24. Terakhir adalah menambahkan node **Text** dari **Dashboard** dan hubungkan ke Template **Timestamp**



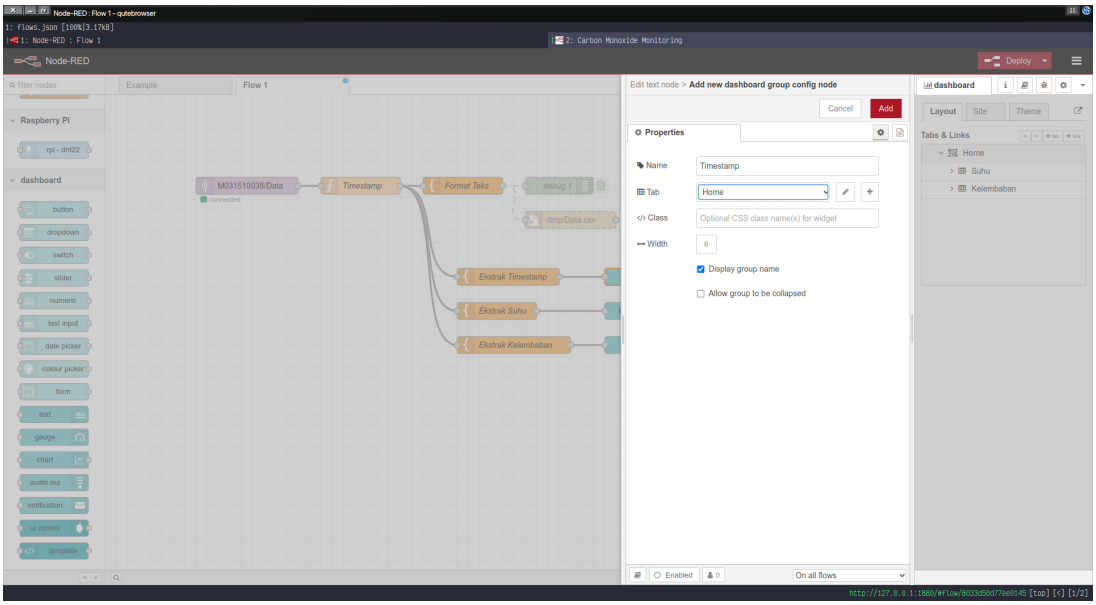
Gambar 4.25: Node Text baru

25. **Double Click** node **Text** dan Konfigurasikan seperti berikut:
    - Label : Timestamp
    - Name : Timestamp



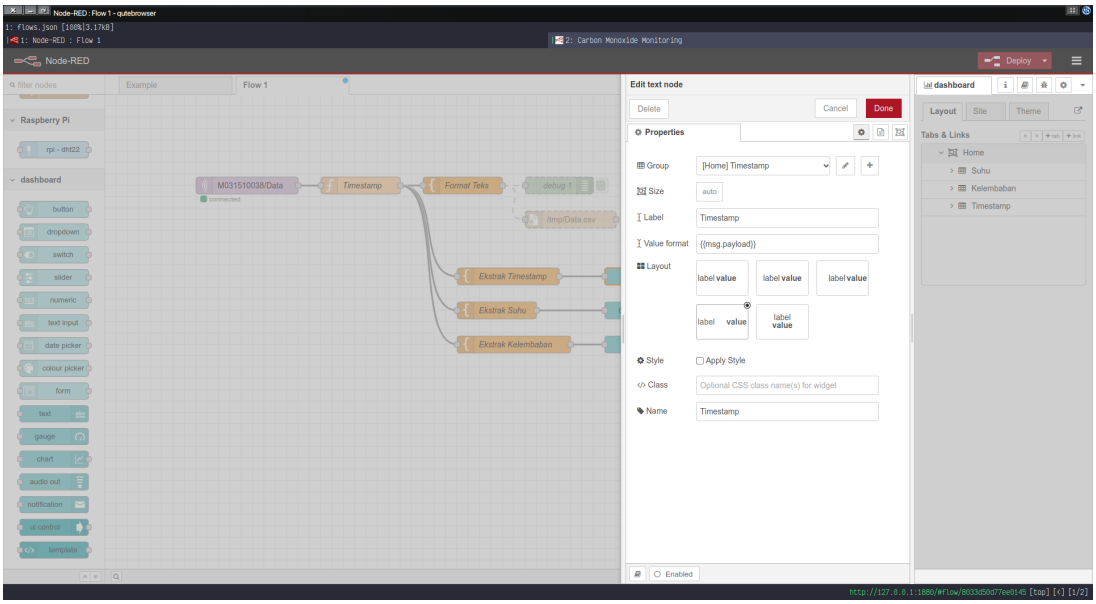
Gambar 4.26: Konfigurasi Node Text baru

26. Untuk konfigurasi **Group** untuk node **Text**. Klik + **Tambah**. Lalu dibagian konfigurasi **Group**, masukkan **Name : Timestamp**. Pilih Tab dengan **Home**. Jika sudah klik **Add**



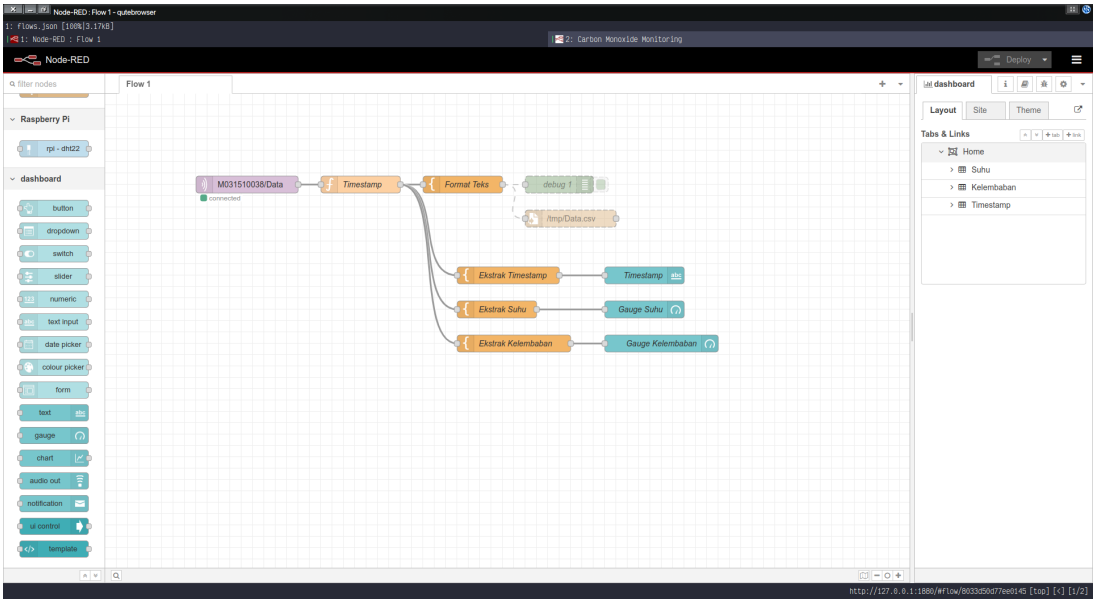
Gambar 4.27: Konfigurasi Group Node Text baru

27. Jika konfigurasi sudah lengkap untuk node **Text**, klik **Done**



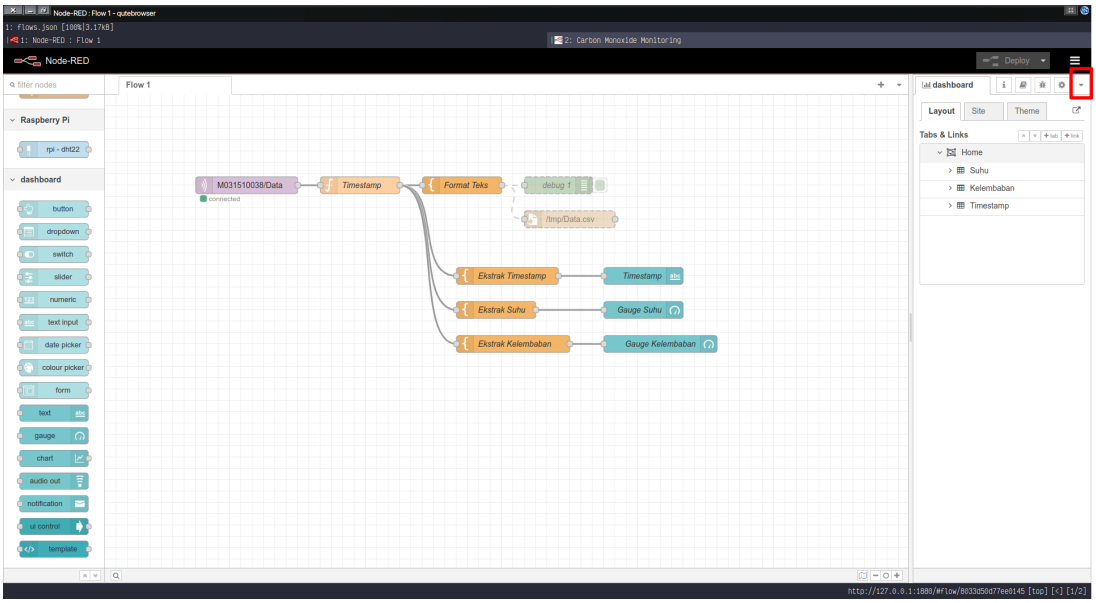
Gambar 4.28: Konfigurasi Node Text

28. Klik **Deploy** untuk finalisasi.



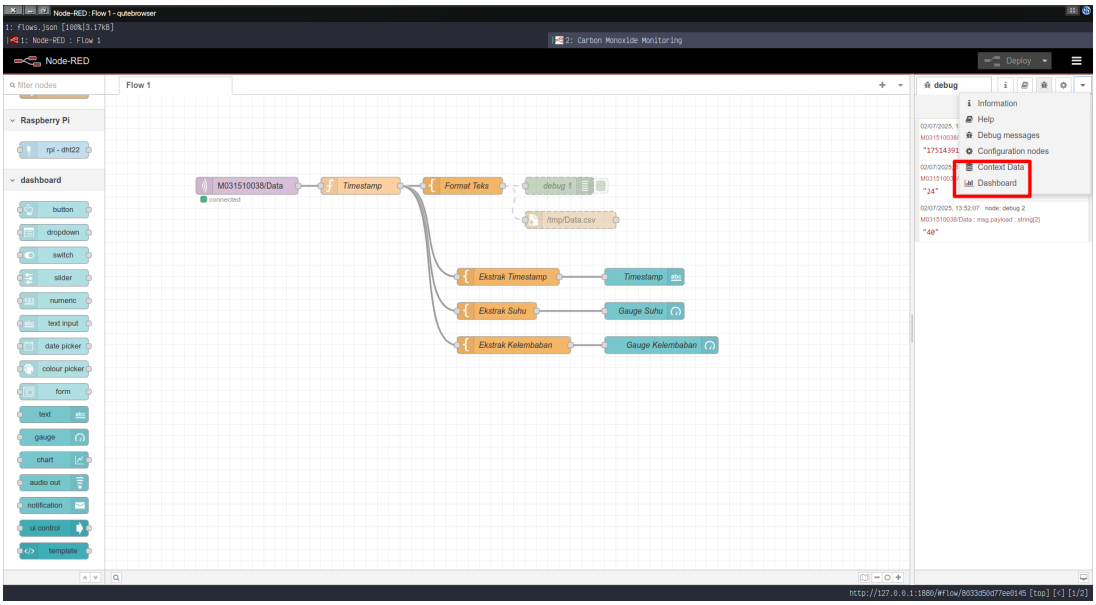
Gambar 4.29: Finalisasi Dashboard

29. Untuk melihat **Dashboard**, klik **Segitiga Kecil** di bawah **Tiga Strip**



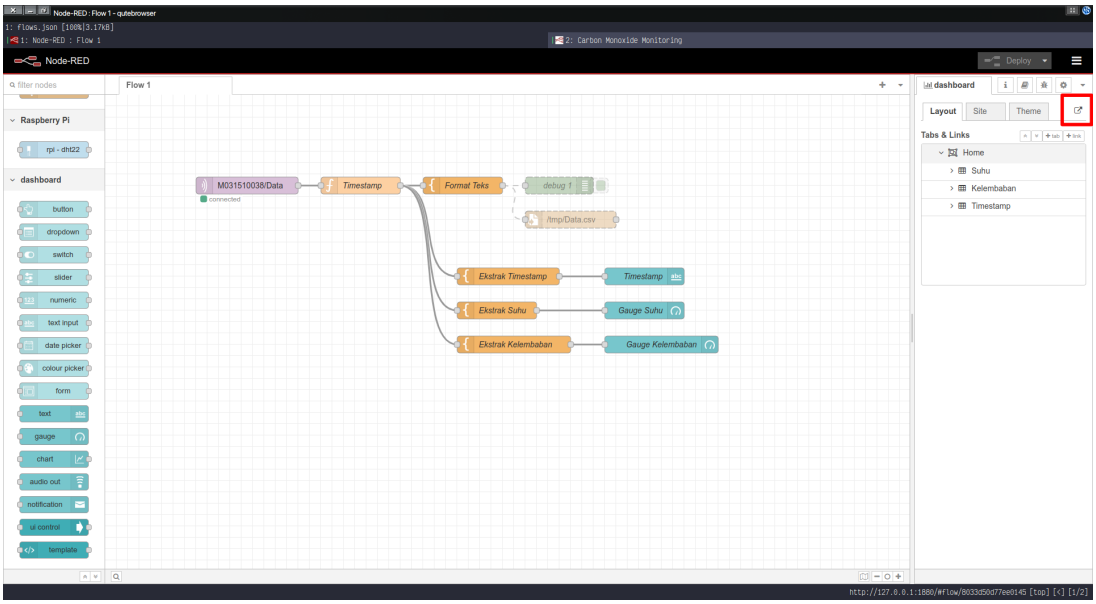
Gambar 4.30: Segitiga kecil

30. Pilih **Dashboard**



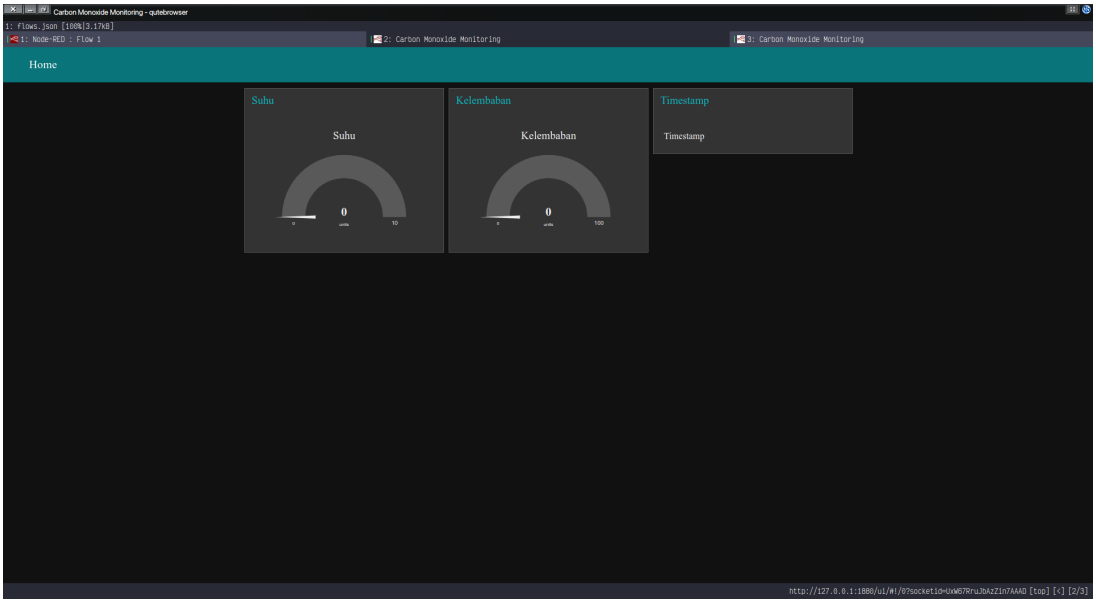
Gambar 4.31: Tukar Menu Dashboard

31. Klik **New Window** di bawah **Segitiga kecil** tadi



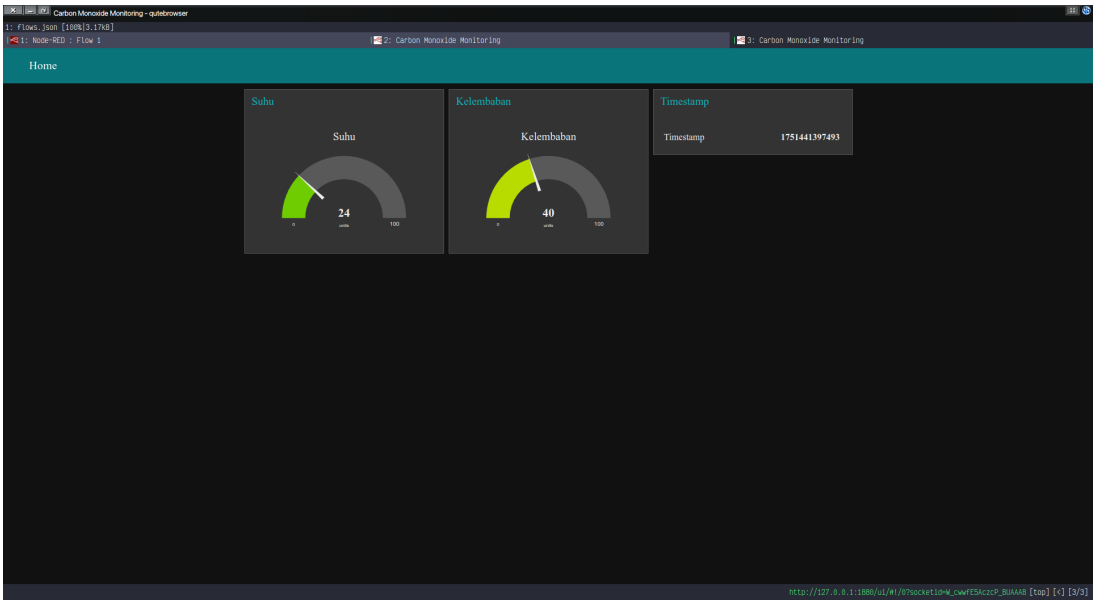
Gambar 4.32: New Window

32. Maka akan muncul tampilan Dashboard



Gambar 4.33: Tampilan Dashboard

33. Untuk mengetes, jalankan **Wokwi**



Gambar 4.34: Dashboard dan Data Wokwi

34. Gunakan **Wokwi** untuk mengatur **DHT22** sesuai keinginan

4.4 Tugas SKPI

1. Tambahkan tampilan **Text** untuk tiap data di Dashboard
2. Tambahkan tampilan diagram garis dashboard untuk tiap data
3. Ekspor sebagai JSON lalu kompres ke ZIP
4. Beri nama file NIM.ZIP
5. Kirim ke Google Drive : Tugas SKPI