

LAPORAN PRAKTIKUM



Kelompok : 5

Nama Anggota :

1. Alif Rizki Ananta Harahap
2. Andita Farah Salsabila
3. Hakim Asrori
4. Mali Nur Al Isthifa
5. Muhammad Ridzal Maulana
6. Rantika

Kelas : D3TI.2C

Mata Kuliah : Internet Of Things

Praktikum ke / Judul : 6/ Visualisasi Platform Node-Red

Tanggal Praktikum : 6 September 2021

Dosen Pengampu : Ahmad Rifai, S.Tr.Kom., M.Tr.Kom

MODUL 6 – VISUALISASI DENGAN PLATFORM NODE-RED

1.1. Tujuan

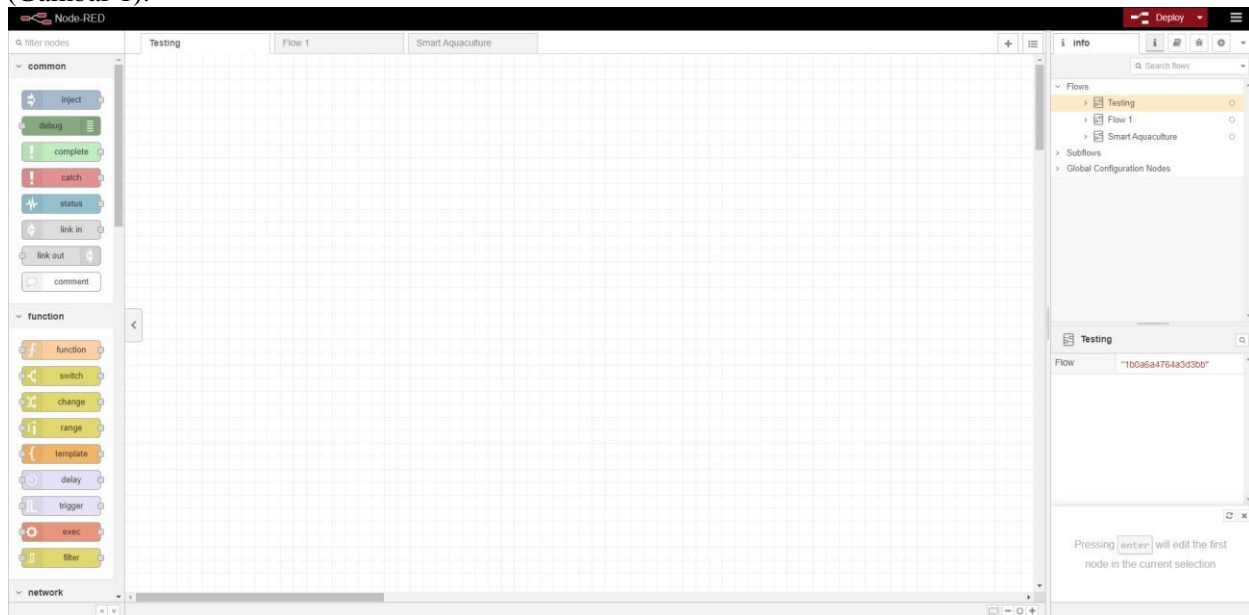
- Mahasiswa mampu untuk menampilkan data yang dikirimkan oleh Node secara realtime.
- Mahasiswa mampu untuk menampilkan data dalam bentuk grafik.
- Mahasiswa mampu untuk menyimpan data yang dikirimkan oleh Node.
- Mahasiswa mampu untuk menampilkan data yang tersimpan dalam bentuk grafik.

1.2. Praktikum

1.2.1. Platform IoT

Sekarang ini dukungan untuk teknologi IoT sudah mulai banyak. Salah satunya adalah beberapa platform dan tools yang menyediakan konektifitas antar node IoT. Menampilkan data dan mengontrol data sudah dapat dilakukan oleh platform-platform tersebut. Platform tersebut diantaranya adalah Node-Red, Blink, ThingSpeak, ThingsBoard, KaaIoT, Openremote dan lain sebagainya. Pada praktikum ini kita akan menggunakan salah satu dari platform atau tools tersebut yaitu Node-Red.

Node-Red merupakan platform yang menyediakan macam-macam pengolahan data hingga visualisasi data dengan menggunakan diagram blok. Berikut ini adalah tampilan antarmuka NodeRed (Gambar 1).



Gambar 1. Antarmuka Node-Red

Untuk dapat menggunakan Node-Red kalian bisa melakukan instalasi Node-Red tersebut melalui website official-nya disini (<https://nodered.org/docs/getting-started/windows>).

Quick Start

1. Install Node.js

Download the latest 14.x LTS version of Node.js from the official [Node.js home page](#). It will offer you the best version for your system.

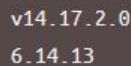
Run the downloaded MSI file. Installing Node.js requires local administrator rights; if you are not a local administrator, you will be prompted for an administrator password on install. Accept the defaults when installing. After installation completes, close any open command prompts and re-open to ensure new environment variables are picked up.

Once installed, open a command prompt and run the following command to ensure Node.js and npm are installed correctly.

Using Powershell: `node --version; npm --version`

Using cmd: `node --version && npm --version`


You should receive back output that looks similar to:



```
v14.17.2.0  
6.14.13
```

2. Install Node-RED

Installing Node-RED as a global module adds the command `node-red` to your system path. Execute the following at the command prompt:



```
npm install -g --unsafe-perm node-red
```

3. Run Node-RED

Once installed, you are ready to run Node-RED.

Gambar 2. Langkah-langkah menginstall Node-Red

1.2.2. Pengenalan Node-Red

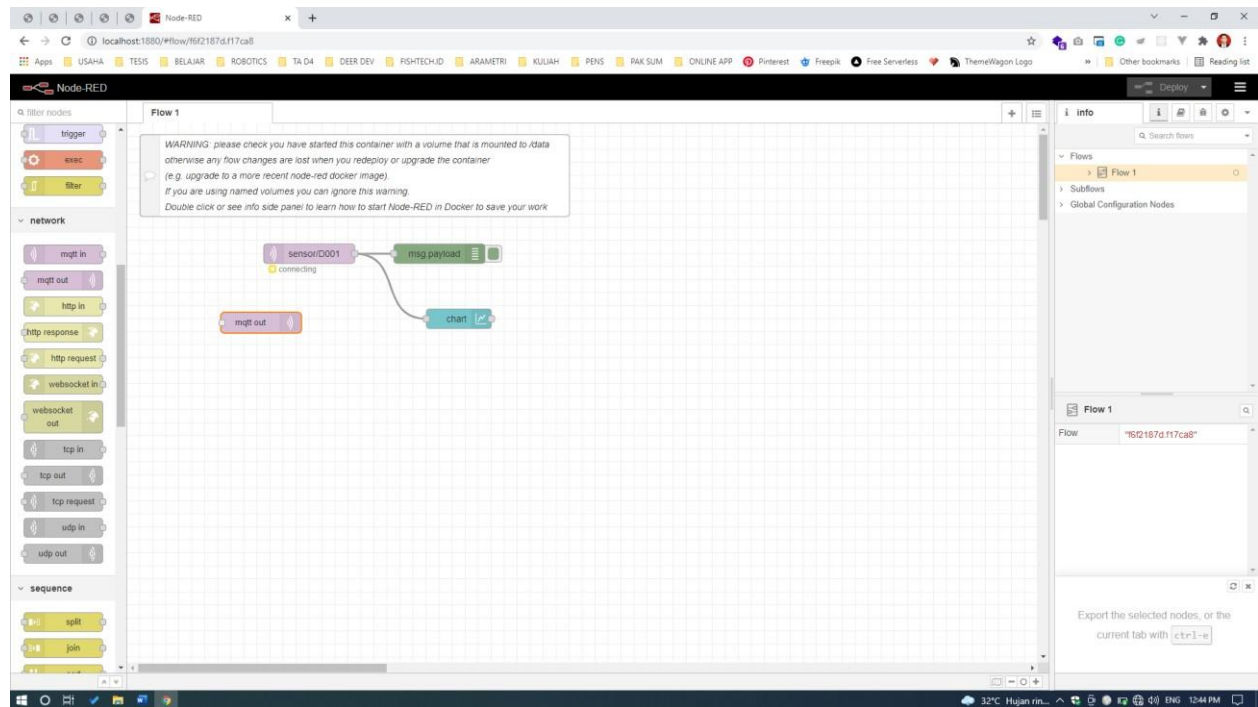
Pada dasarnya platform merupakan suatu aplikasi yang dirancang untuk dapat digunakan secara dinamis agar dapat digunakan untuk membangun suatu sistem dalam bentuk yang lain dan kegunaan yang lain sesuai user yang menggunakannya. Node-Red disini memiliki fitur konektifitas yang dapat terhubung dengan perangkat IoT melalui beberapa protokol komunikasi. Salah satunya yang akan dipratakan adalah protokol komunikasi MQTT. Letak blok diagram komunikasi MQTT berada pada group network seperti berikut.



Gambar 3. Komponen group network

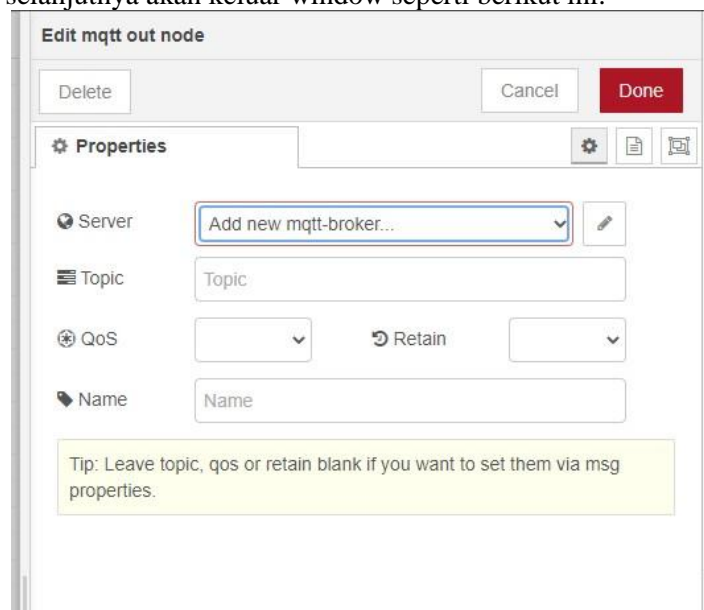
Blok *mqtt in* digunakan untuk melakukan subscribe sedangkan *mqtt out* digunakan untuk melakukan publish data.

Cara menggunakan blok-blok komponen pada Node-Red tersebut cukup dengan melakukan dragn-drop komponen tersebut.



Gambar 4. Drag and Drop Komponen

Membuat koneksi ke MQTT broker dapat dilakukan dengan melakukan double klik pada komponen *mqtt in* atau *mqtt out*, selanjutnya akan keluar window seperti berikut ini.



Gambar 5. Membuat koneksi ke Broker

Klik pada icon pencil maka akan memunculkan konfigurasi koneksi ke Broker seperti berikut ini.

Selanjutnya klik Add untuk melakukan simpan settingan tersebut.

Edit mqtt out node > Add new mqtt-broker config node

Cancel Add

Properties

Name: MATKUL IOT-BROKER HIVEMQ

Connection | Security | Messages

Server: broker.hivemq.com Port: 1883

☐ Use TLS

Protocol: MQTT V3.1.1

Client ID: Leave blank for auto generated

Keep Alive: 60

Session: ☒ Use clean session

Gambar 6. Konfigurasi server Broker

Subscribe merupakan cara untuk mendapatkan data yang dikirimkan oleh masing-masing node yang bertindak sebagai publisher. Dalam Node-Red untuk melakukan subscriber dapat dilakukan dengan menggunakan komponen mqtt in. berikut ini adalah konfigurasi untuk melakukan subscribe.

Edit mqtt in node

Delete Cancel Done

Properties

Server: MATKUL IOT-BROKER HIVEMQ

Topic: tes

QoS: 2

Output: auto-detect (string or buffer)

Name: SUBSCRIBE

Gambar 7. Konfigurasi sebagai subscriber

Sedangkan apabila ingin mengirimkan data yaitu dilakukan dengan melakukan publish data dengan menggunakan komponen mqtt out seperti berikut ini.

Edit mqtt out node

Delete Cancel Done

Properties

Server: MATKUL IOT-BROKER HIVEMQ

Topic: tes

QoS: 0 Retain: false

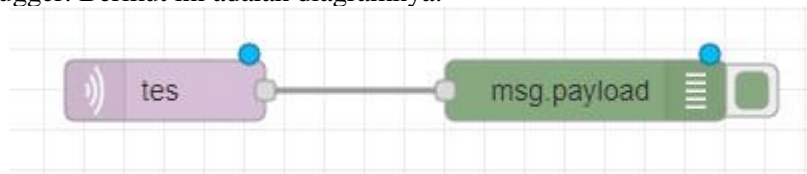
Name: TES

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

Gambar 8. Konfigurasi sebagai publisher

1.2.3. Menampilkan Data yang dikirim Publisher di Node-Red

Untuk menampilkan data yang di kirimkan oleh publisher dapat dilakukan dengan menampilkannya pada window debugger. Berikut ini adalah diagramnya.



Gambar 9. Menampilkan data yang dipublish pada debugger

Komponen  dihubungkan ke . atur konfigurasi pada komponen mqtt in pada broker yang sudah disiapkan dari proses sebelumnya seperti berikut ini.

Edit mqtt in node

Delete Cancel Done

Properties

Server: MATKUL IOT

Topic: tes

QoS: 2

Output: auto-detect (string or buffer)

Name: Name

Gambar 10. Konfigurasi mqtt in sebagai subscriber

Setelah semua konfigurasi selesai jalankan programnya dengan mengklik tombol **Deploy**

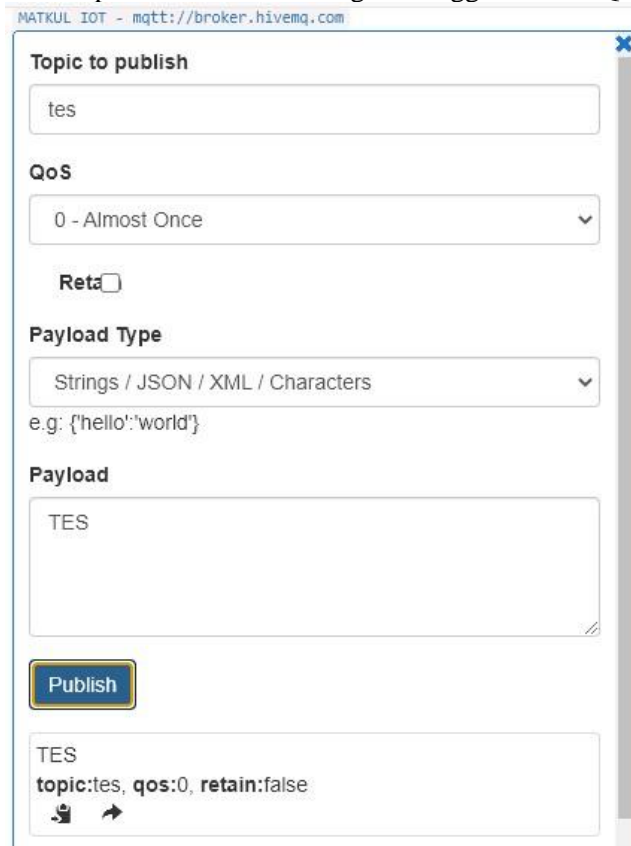


Kemudian buka window debugger pada bagian kanan Node-Red untuk melihat data dari publisher tersebut.



Gambar 11. Window debugger

untuk melakukan publish data dapat disimulasikan dengan menggunakan MQTT Box seperti berikut ini.

A screenshot of the MQTT Box simulation interface. The window has a title bar that says "MATKUL IOT - mqtt://broker.hivemq.com". Inside, there are several fields: "Topic to publish" with the value "tes", "QoS" with a dropdown menu showing "0 - Almost Once", "Retain" with a checkbox, "Payload Type" with a dropdown menu showing "Strings / JSON / XML / Characters", and "Payload" with the value "TES". Below these fields is a "Publish" button. At the bottom, there is a status bar that says "TES" and "topic:tes, qos:0, retain:false".

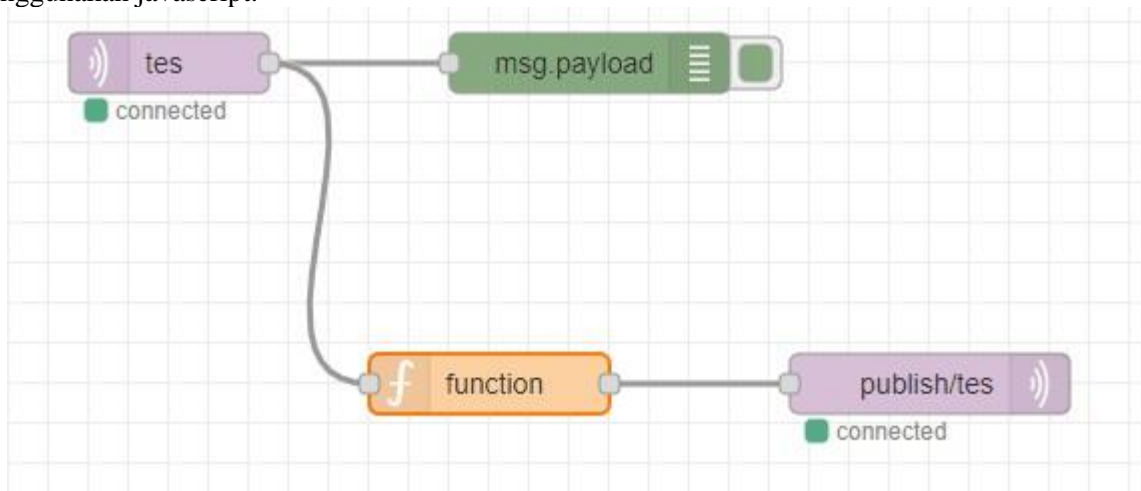
Gambar 12. Simulasi publish data



Gambar 13. Melihat hasil data yang dikirimkan oleh publisher

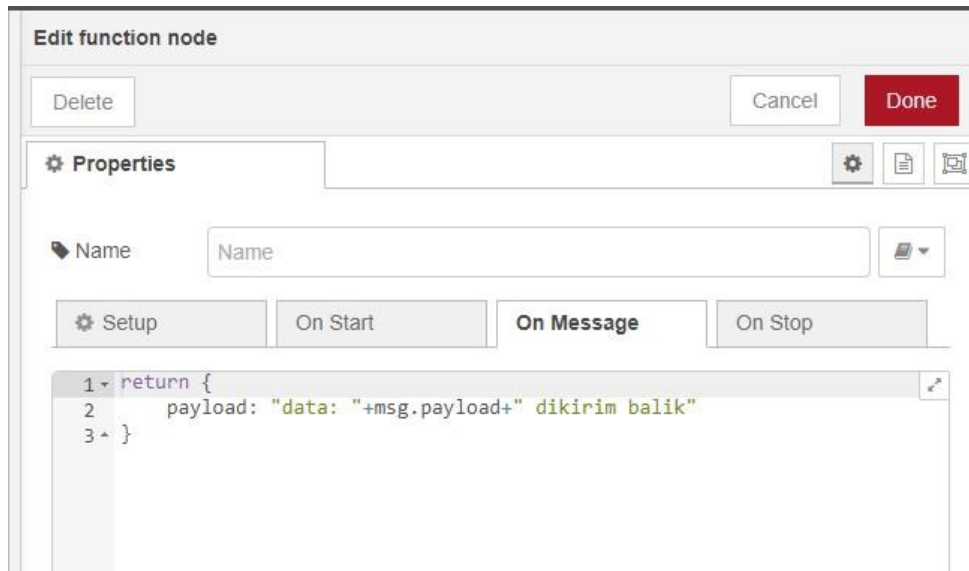
1.2.4. Publish data di Node-Red

Pada Node-Red dapat juga melakukan publish data dengan menggunakan komponen mqtt out. Berikut ini adalah cara untuk melakukan publish data dengan menggabungkan script menggunakan javascript.

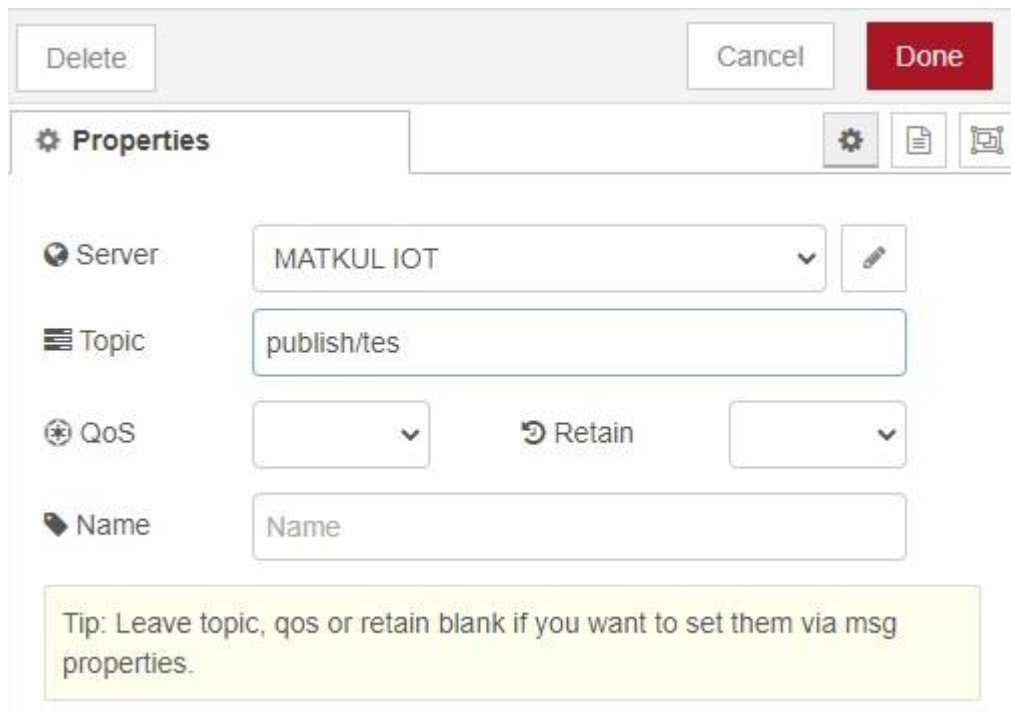


Gambar 14. Diagram publish data dari data yang disubscribe

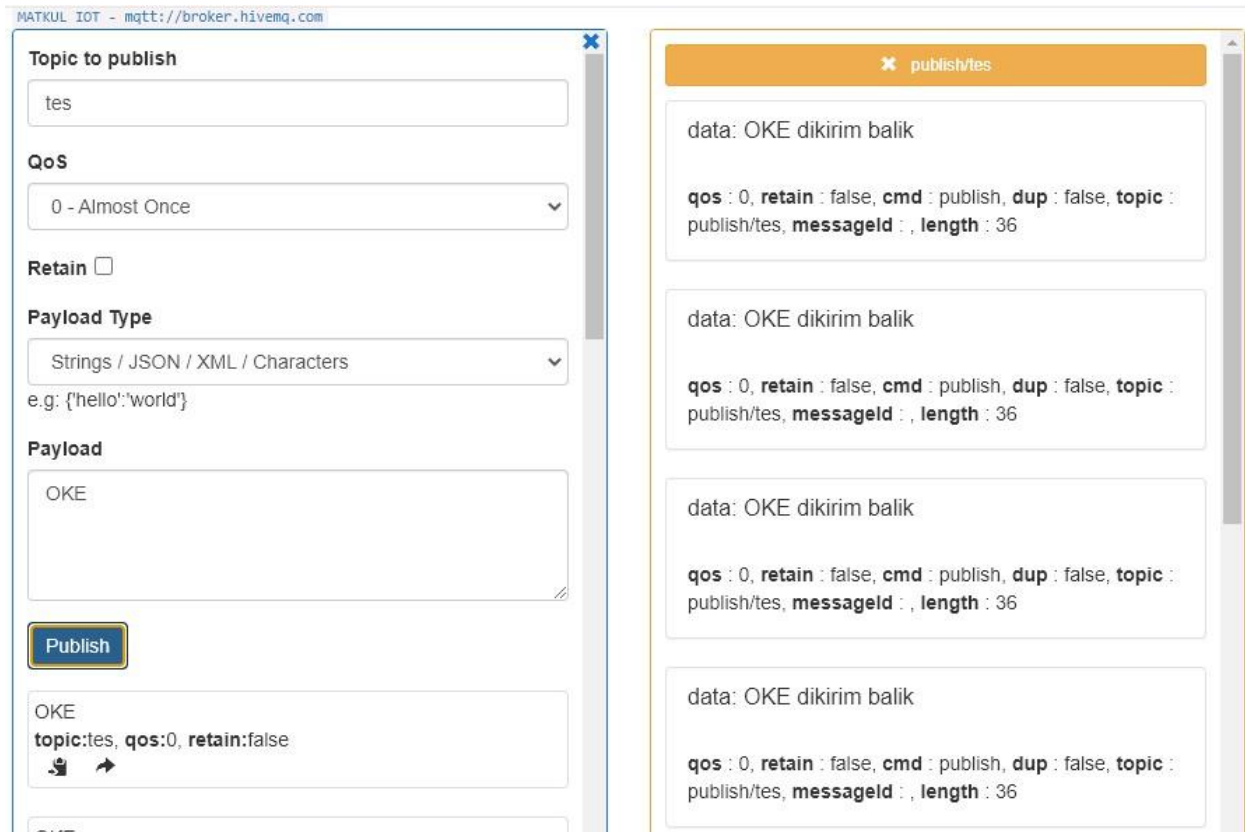
Komponen yang digunakan adalah komponen function dan mqtt out. Komponen function digunakan untuk mensisipkan script dalam bahasa javascript berfungsi untuk mengolah data/mentransformasikan sedangkan komponen mqtt out berfungsi sebagai pengirim (publisher) data dari komponen function. Double klik pada komponen function dan isikan program seperti berikut.



Gambar 15. Isi program pada komponen function



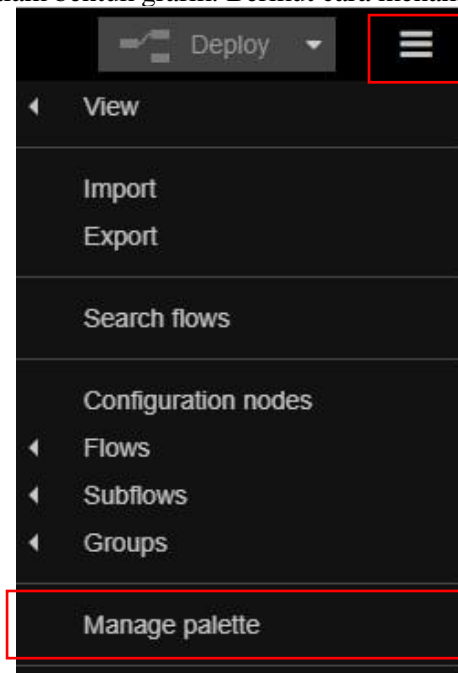
Gambar 16. Konfigurasi komponen mqtt out

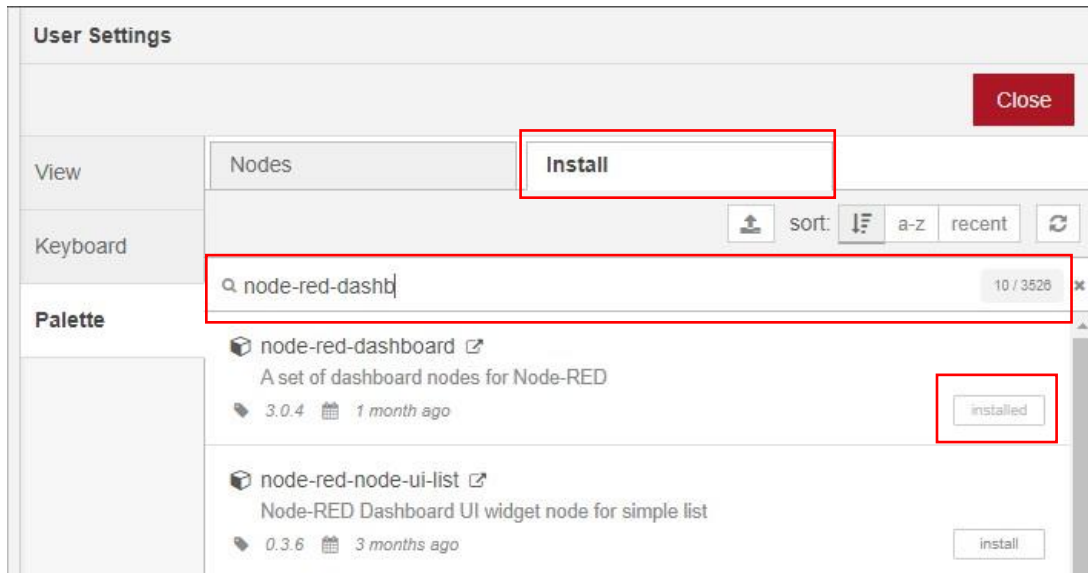


Gambar 17. Hasil simulasi pengiriman data dan subscribe data

1.2.5. Visualisasi data

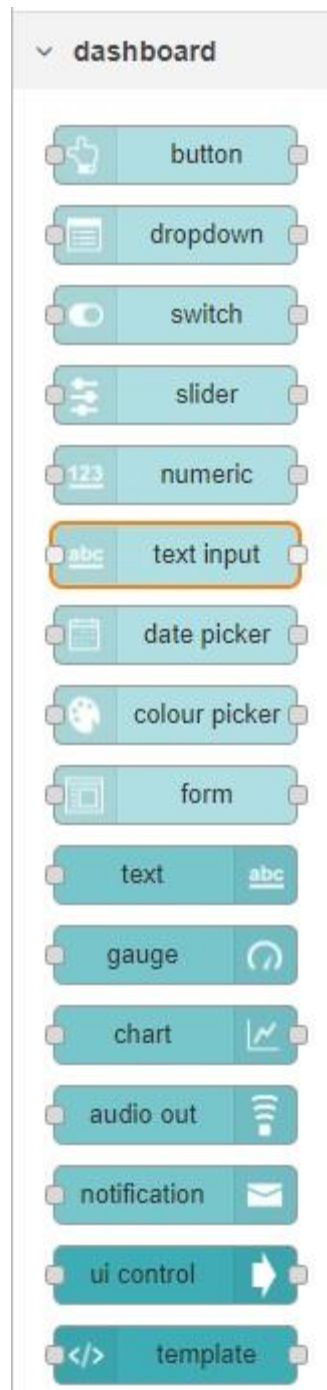
Dalam Node-Red banyak sekali plugin atau palette tambahan yang dapat digunakan untuk pengembangan lebih advance lagi. Salah satunya disini kita akan menggunakan palette tambahan untuk dapat memvisualisasikan data dalam bentuk grafik. Berikut cara menambahkan palette tersebut.





Gambar 18. Langkah-langkah menginstall palette ui dashboard

Setelah proses installasi selesai maka dapat kita jumpai komponen baru paling bawah berupa komponen dashboard seperti berikut ini.

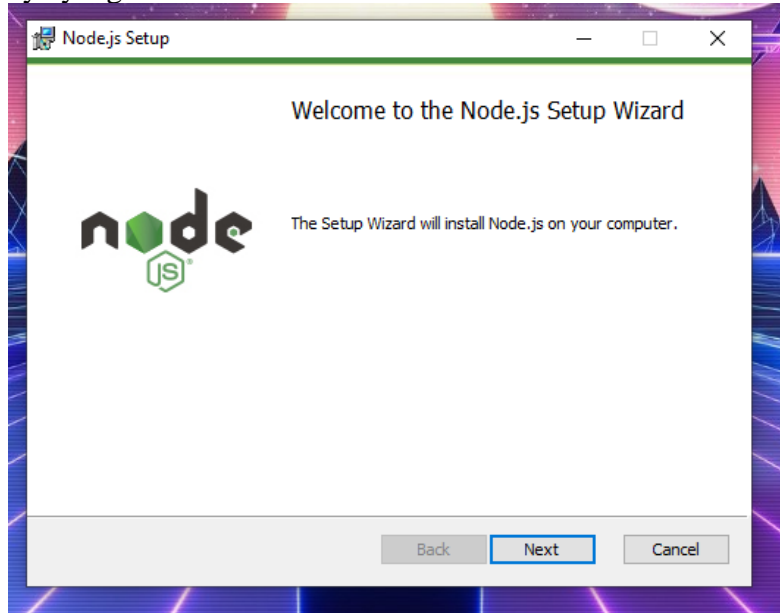


Gambar 19. Komponen dashboard

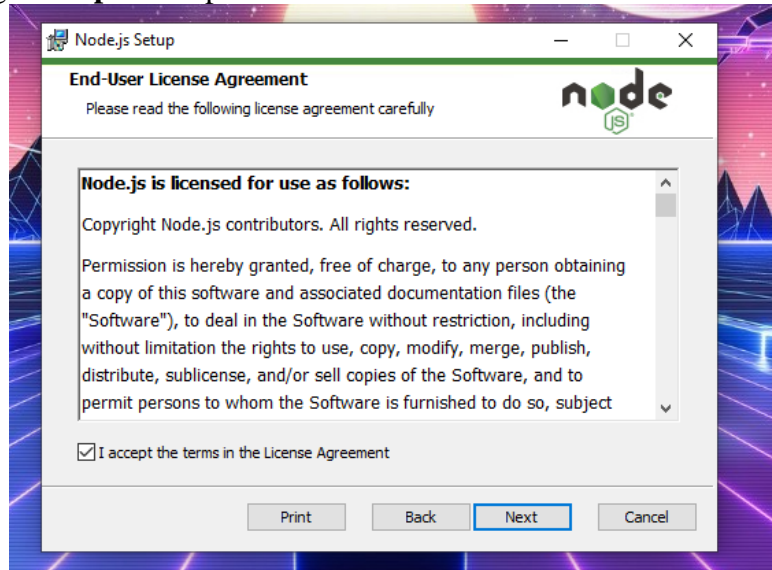
1.1.Tugas

1. Dokumentasikan setiap langkah-langkah instalasi Node-Red berserta menjalankan Node-Red. Install

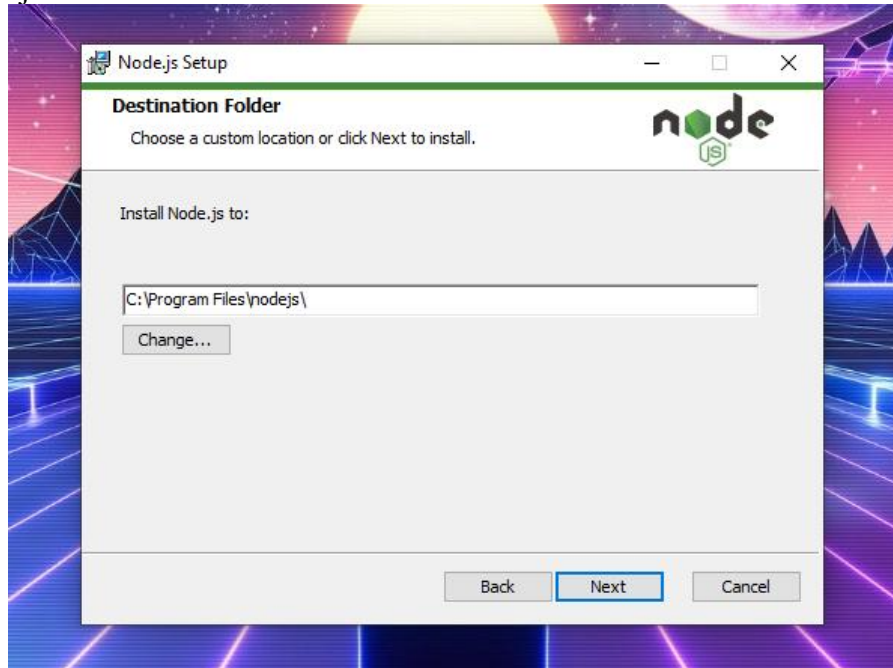
- Pertama Download NodeJs terlebih dahulu, pilih versi yang LTS. Lalu klik 2x di file installernya yang sudah didownload. Pilih next.



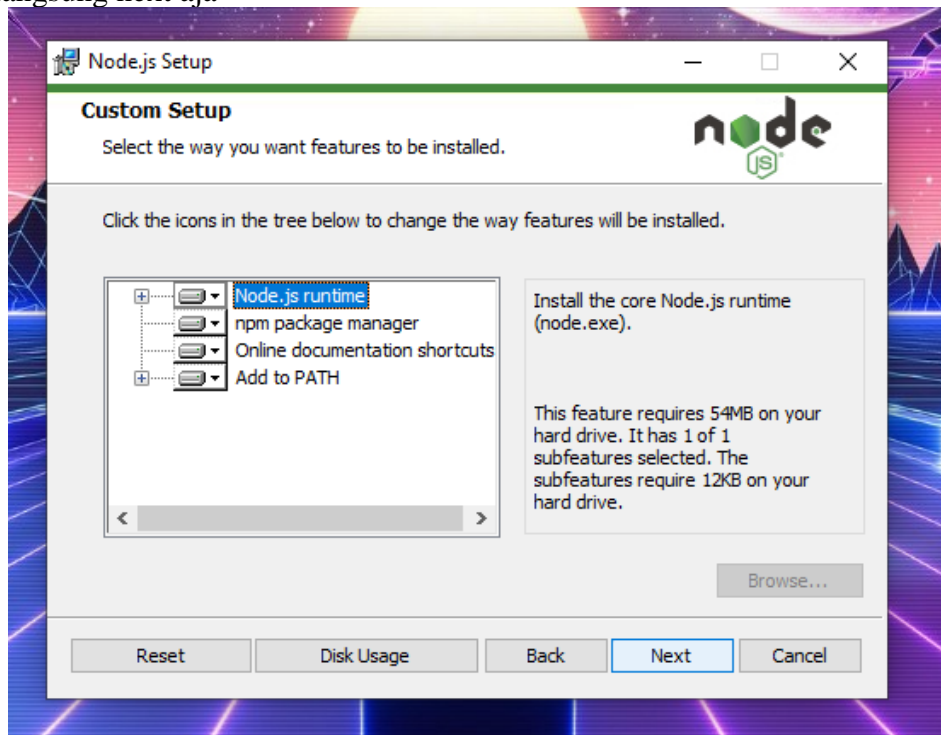
- Centang **I accept**. Lalu pilih next



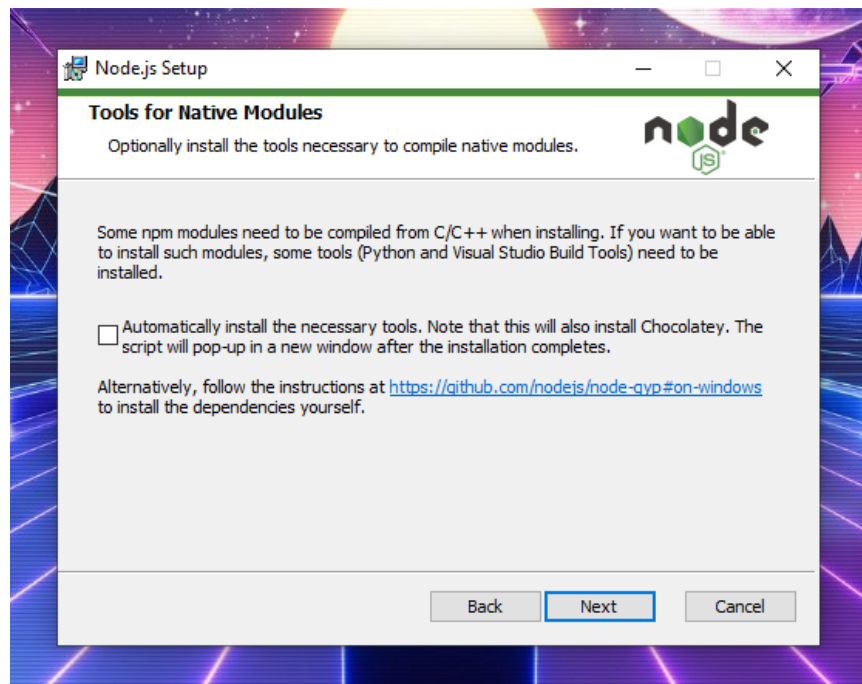
- Langsung pilih next, disini kami menggunakan letak default yang dianjurkan oleh nodejs



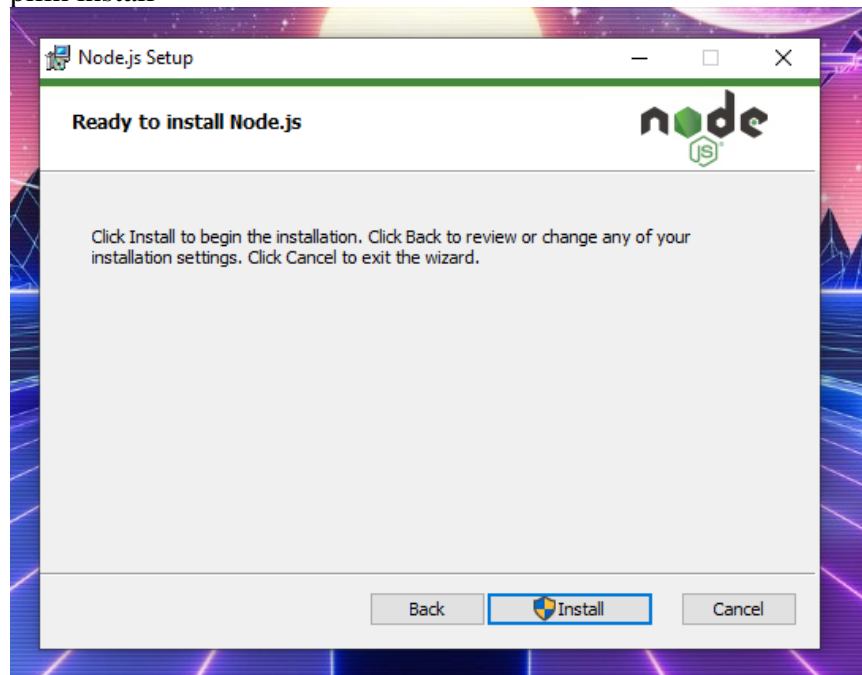
- Langsung next aja

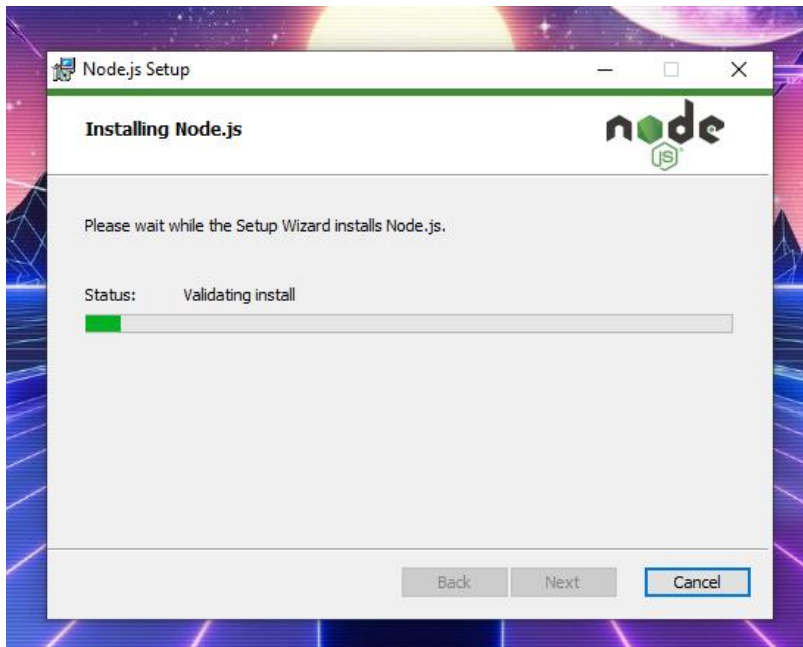


- Langsung next

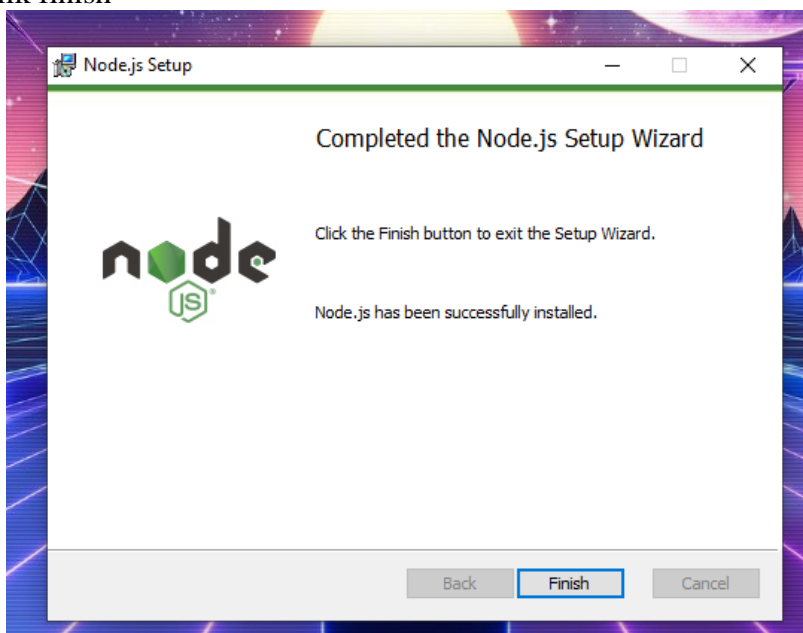


- Lalu pilih install





- Lalu klik finish



- Lalu buka terminal komputer, lalu kita install node-red menggunakan npm

```
MINGW64:/c/Users/dell 7270
dell 7270@DESKTOP-9MU0E00 MINGW64 ~
$ npm install -g --unsafe-perm node-red
C:\Users\dell 7270\AppData\Roaming\npm\node-red -> C:\Users\dell 7270\AppData\Roaming\npm\node_modules\node-red
C:\Users\dell 7270\AppData\Roaming\npm\node-red-pi -> C:\Users\dell 7270\AppData\Roaming\npm\node_modules\node-red-pi

> bcrypt@5.0.1 install C:\Users\dell 7270\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

[bcrypt] Success: "C:\Users\dell 7270\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt\lib\
+ node-red@2.1.1
added 290 packages from 371 contributors in 56.25s

dell 7270@DESKTOP-9MU0E00 MINGW64 ~
$
```

- Lalu jalankan node-red menggunakan perintah **node-red**

```
de11 7270@DESKTOP-9MUEEO MINGW64 ~
$ node-red
25 Oct 13:26:26 - [info]

Welcome to Node-RED
=====

25 Oct 13:26:26 - [info] Node-RED version: v2.1.1
25 Oct 13:26:26 - [info] Node.js version: v14.18.1
25 Oct 13:26:26 - [info] Windows_NT 10.0.19043 x64 LE
25 Oct 13:26:27 - [info] Loading palette nodes
25 Oct 13:26:29 - [info] Settings file : C:\Users\de11 7270\.node-red\settings.js
25 Oct 13:26:29 - [info] Context store : 'default' [module=memory]
25 Oct 13:26:29 - [info] User directory : C:\Users\de11 7270\.node-red
25 Oct 13:26:29 - [warn] Projects disabled : editorTheme.projects.enabled=false
25 Oct 13:26:29 - [info] Flows file : C:\Users\de11 7270\.node-red\flows.json
25 Oct 13:26:29 - [info] Creating new flow file
25 Oct 13:26:29 - [warn]

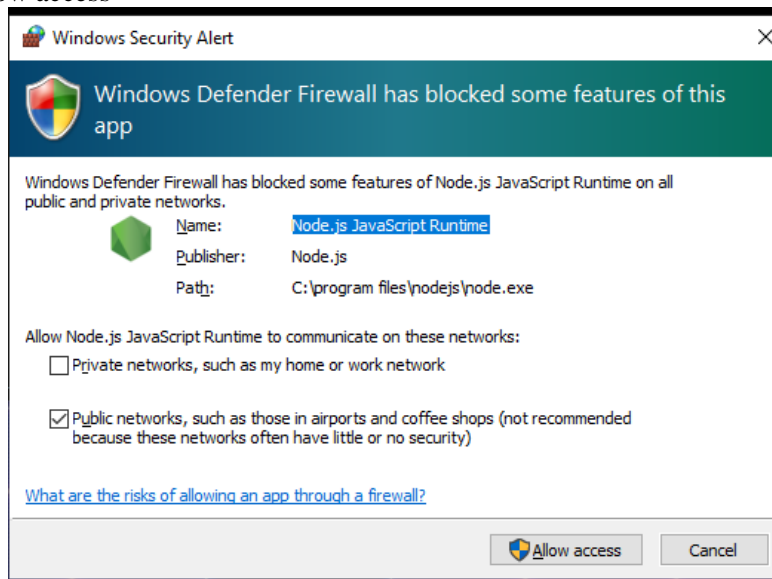
-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

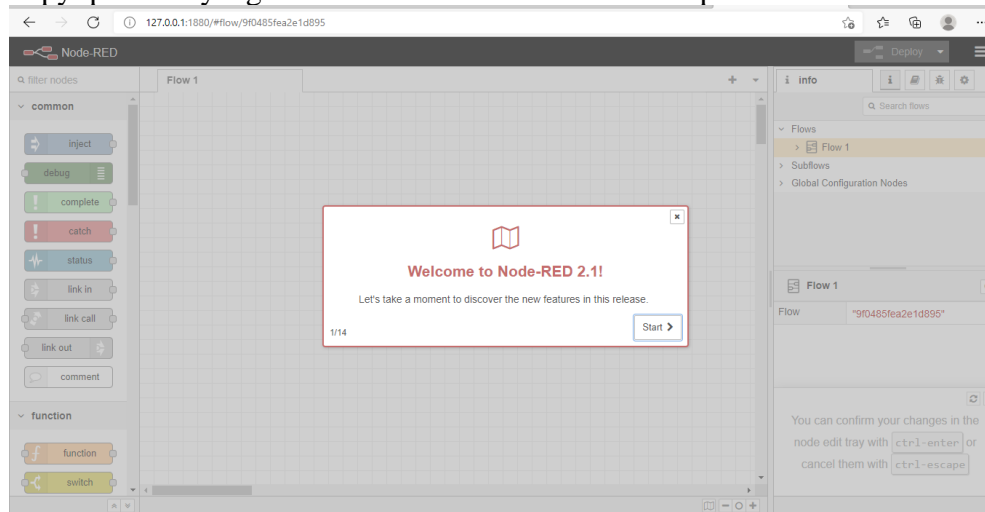
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

25 Oct 13:26:29 - [info] Server now running at http://127.0.0.1:1880/
25 Oct 13:26:29 - [info] Starting flows
25 Oct 13:26:29 - [info] Started flows
```

- Pilih allow access



- Copy ip server yang diberikan dari node-red. Yaitu <http://127.0.0.1:1880/>



2. Menampilkan data dalam bentuk grafik Gauge untuk masing-masing paramter sensor yaitu Humidity, Temperature dan LDR.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

WiFiClient espClient;
PubSubClient client(espClient);

#define WIFI_SSID "Kusen_Hotspot"
#define WIFI_PASS ""
#define MQTT_SERVER "broker.hivemq.com"
#define MQTT_PORT 1883
#define DHTPIN D2
#define DHTTYPE DHT11
#define PIN_LDR A0

unsigned long _waiting = millis();
unsigned long _now;
int value = 0;
char data[50];

DHT dht(DHTPIN, DHTTYPE);

String buffData;

void setup_wifi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length){ }
```

```

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "Kelompok-5-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.publish("sukses_konek", "Yess... saya terkoneksi");
      client.subscribe("kelompok-5/act/led");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(MQTT_SERVER, MQTT_PORT);
  client.setCallback(callback);
  dht.begin();
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  kirimPer2Detik();
}

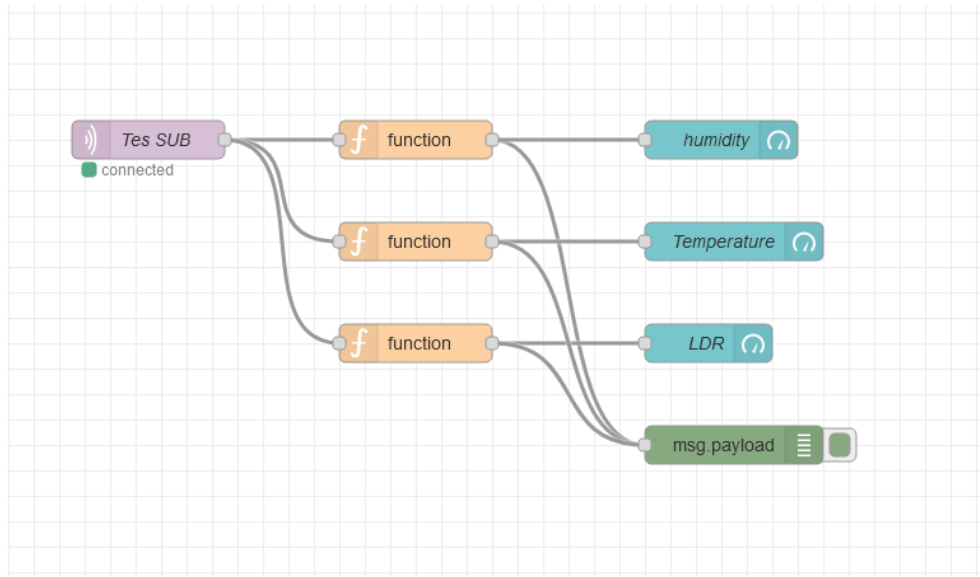
void kirimPer2Detik(){
  _now = millis();
  if(millis() - _waiting > 2000){
    _waiting = _now;

    sprintf(data, "%g#%g#%i",
      dht.readTemperature(),
      dht.readHumidity(),
      analogRead(PIN_LDR)
    );

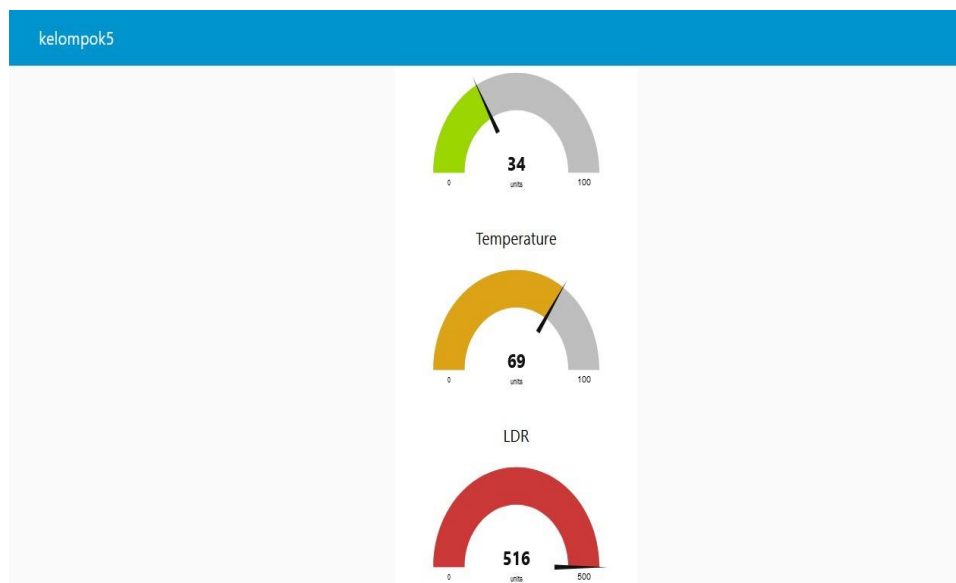
    Serial.print("Publish message: ");
    Serial.println(data);
    client.publish("kelompok-5/sensor/temp", data);
  }
}

```

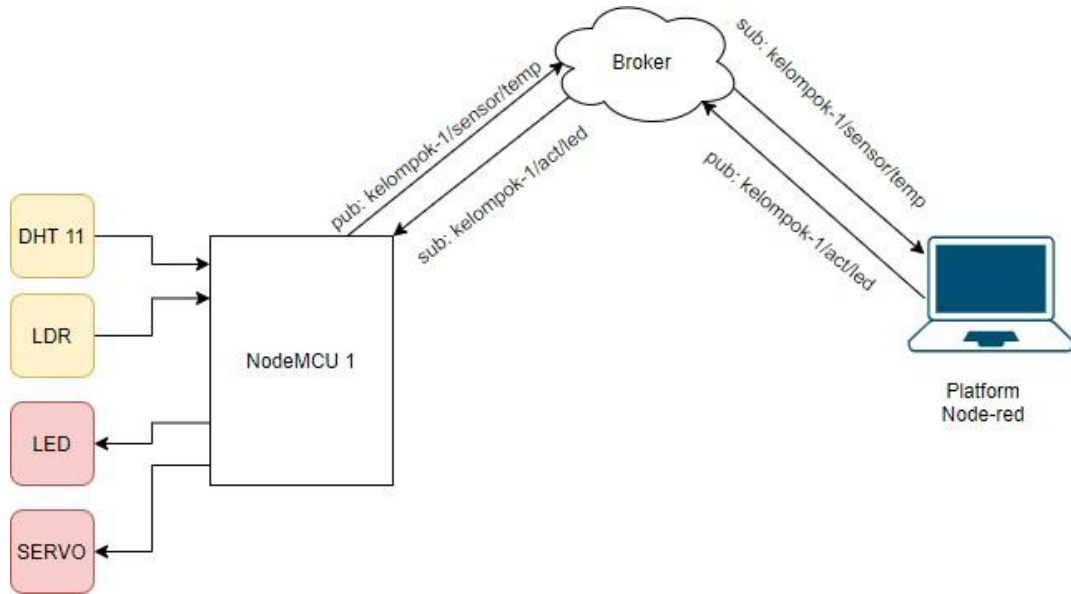
- Node-RED



- Grafik Gaug



3. Melakukan hal yang sama seperti praktikum sebelumnya namun menggantikan kontrol sistemnya menggunakan platform Node-Red dengan ketentuan berikut ini.



NodeMCU 1 bertugas sebagai pengambil data sensor dan terhubung dengan LED. Sedangkan Device Node-Red bertugas sebagai pemroses data yang dikirimkan oleh NodeMCU 1 melalui jaringan dengan protokol MQTT. Berikut ini adalah Rule dari kasus tersebut.

No	Sensor	Servo	LED
1	Temperature > 29	90 derajat	-
2	Temperature <= 29	0 derajat	-
3	LDR mendeteksi cahaya		ON
4	LDR tidak mendeteksi cahaya		OFF

- **Kode Program**

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
#include <Servo.h>
#include <ArduinoJson.h>

Servo myservo;
WiFiClient espClient;
PubSubClient client(espClient);

#define WIFI_SSID "Kusen_Hotspot"
#define WIFI_PASS ""
#define MQTT_SERVER "broker.hivemq.com"
#define MQTT_PORT 1883
#define DHTPIN D2
#define DHTTYPE DHT11
#define PIN_LDR A0

unsigned long _waiting = millis();
unsigned long _now;
int value = 0;
char data[50];

DHT dht(DHTPIN, DHTTYPE);

String buffData;

void setup_wifi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length){
  buffData = "";
  for(int i=0; i<length; i++){
    buffData += (char) payload[i];
  }
}
```



```

Serial.print(topic);
Serial.print(" ==> ");
Serial.println(buffData);

StaticJsonDocument<200> doc;
deserializeJson(doc,payload);

int rotate = doc["servo"];

myservo.write(rotate);

if(doc["led"] == "HIGH")
    digitalWrite(D3, HIGH);
else
    digitalWrite(D3, LOW);
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        String clientId = "Kelompok-5-";
        clientId += String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println("connected");
            client.publish("sukses_konek", "Yess... saya terkoneksi");
            client.subscribe("kelompok-5/act/led");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(57600);
    setup_wifi();
    pinMode(D3, OUTPUT);
    myservo.attach(D1);
    client.setServer(MQTT_SERVER, MQTT_PORT);
    client.setCallback(callback);
    dht.begin();
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    kirimPer2Detik();
}

```

```

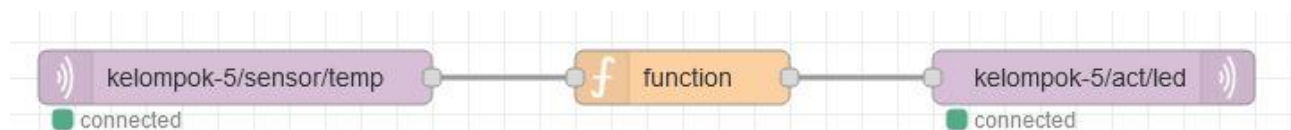
void kirimPer2Detik(){
  _now = millis();
  if(millis() - _waiting > 2000){
    _waiting = _now;

    sprintf(data, "%g#%i",
      dht.readTemperature(),
      analogRead(PIN_LDR)
    );

    Serial.print("Publish message: ");
    Serial.println(data);
    client.publish("kelompok-5/sensor/temp", data);
  }
}

```

- **Node –RED**



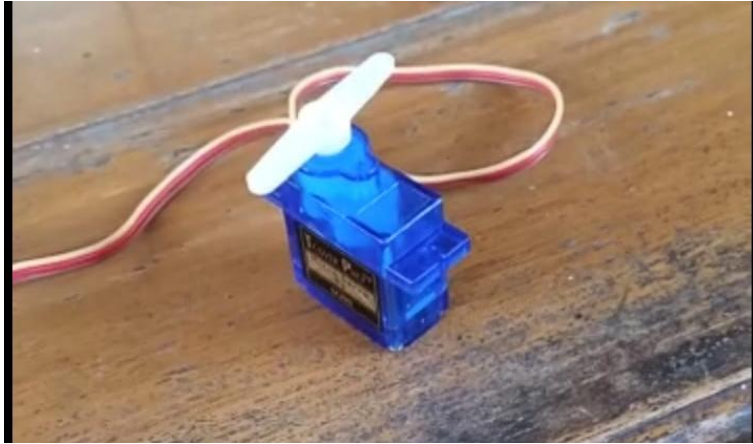
- **Hasil**

```

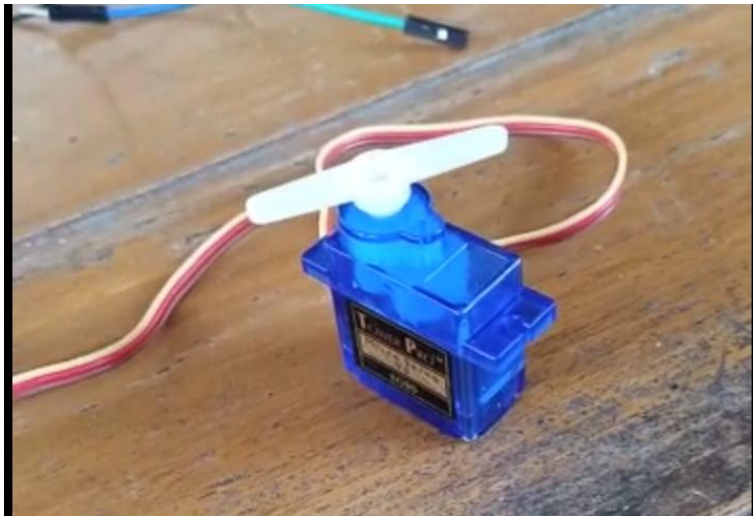
kelompok-5/act/led ==> {"led":"HIGH","servo":90}
kelompok-5/act/led ==> {"led":"HIGH","servo":90}
kelompok-5/act/led ==> {"led":"HIGH","servo":90}
kelompok-5/act/led ==> {"led":"HIGH","servo":90}
kelompok-5/act/led ==> {"led":"HIGH","servo":90}
kelompok-5/act/led ==> {"led":"HIGH","servo":90}
Publish message: 33#56
kelompok-5/act/led ==> {"led":"LOW","servo":90}
Publish message: 33#60
kelompok-5/act/led ==> {"led":"LOW","servo":90}
Publish message: 33#51
kelompok-5/act/led ==> {"led":"LOW","servo":90}
Publish message: 33#51
kelompok-5/act/led ==> {"led":"LOW","servo":90}
Publish message: 33#52

```

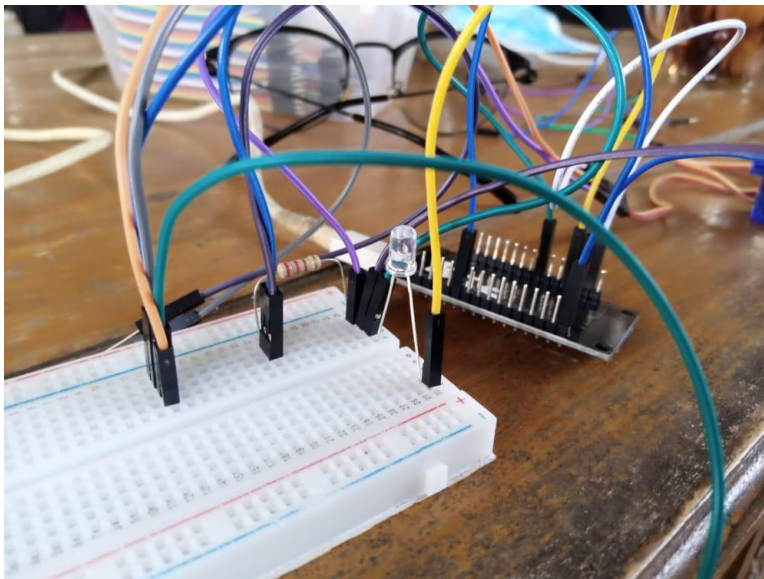
- **Posisi Servo 90**



- **Posisi Servo 0**



- **Led Off**



- **Led On**

