IMPLEMENTASI ALAT PENGUKUR JARAK MENGGUNAKAN SENSOR ULTRASONIK HC-SR04 BERBASIS PROTOKOL MQTT MENGGUNAKAN HIVEMQ

Dosen Pengampu :

Sayyidul Aulia Alamsyah, S.T., M.T.

Nama anggota kelompok :

1. Tri Rejeki Andani (21050874017)
2. Taj Hakam Ikhsan (21050874033)

Mata Kuliah :
Sistem IoT

TEKNIK ELEKTRO 2021 B
Konsentrasi Elektronika

UNIVERSITAS NEGERI SURABAYA
FAKULTAS TEKNIK
PRODI S1 TEKNIK ELEKTRO
SURABAYA

2023

ABSTRAK

Pada alat pengukur jarak berbasis *Internet of Things* (IoT) menggunakan mikrokontroler NodeMCU ESP8266 sebagai pengontrol rangkaian elektronik ini dan dihubungkan dengan protokol komunikasi data (Message Queuing Telemetry Transport) MQTT dengan memanfaatkan *Hivemq JS* sebagai *web publisher* dan *Express JS* sebagai *web* tampilan client. Sensor ultrasonik HC-SR04 dimanfaatkan sebagai pengukur jarak dari suatu benda. Kelebihan dari sistem ini dapat dipantau secara *real time* menggunakan tampilan *web server* pengguna dengan *Express JS*. Pengguna dapat melihat data pengukuran sensor ultrasonic HCSR-04 yang dikirimkan di tampilan *web Express JS*, sehingga sangat memudahkan pengguna dalam memantau data dari jarak jauh.

Kata Kunci : NodeMCU ESP8266, MQTT, Internet Of Things.

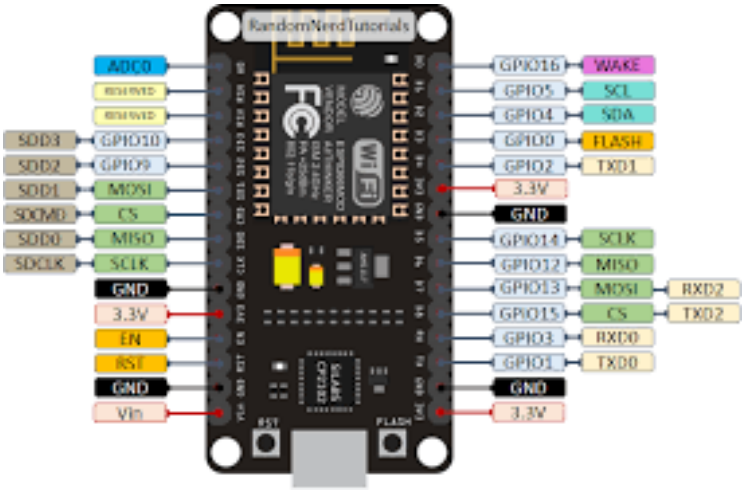
ABSTRACT

The Internet of Things (IoT)-based distance measuring device uses a NodeMCU microcontroller ESP8266 as the controller of this electronic circuit and is connected to the MQTT data communication protocol (Message Queuing Telemetry Transport) by utilizing Havemq JS as a web publisher and Express JS as a web display client. The HC-SR04 ultrasonic sensor is used as a distance meter from an object. The advantages of this system can be monitored in real time using the user's server web view with Express JS. Users can view the HCSR-04 ultrasonic sensor measurement data transmitted in the Express JS web display, making it very easy for users to monitor the data remotely.

Keywords: NodeMCU ESP8266, MQTT, Internet Of Things.

BAB II
TINJAUAN PUSTAKA

2.1.1 Node MCU ESP8266



Gambar 2. 1 Node MCU ESP8266

Sumber : https://www.electronicaembajadores.com

Node MCU ESP8266 merupakan SoC yang memiliki module wifi sebagai perangkat tambahan mikrokontroller agar dapat terhubung dengan wifi dan membuat koneksi TCP/IP. NodeMCU merupakan sebuah platform IoT yang bersifat open source dan juga include dengan module ESP 12, dan berjalan pada firmware esp8266 yang menjadikan NodeMCU sebuah mikrokontroller yang telah dilengkapi dengan module Wifi didalamnya. dll.

2.1.2 Sensor Ultrasonik HC-SR04

Gambar 2. 2 Sensor Ultrasonik HC-SR04

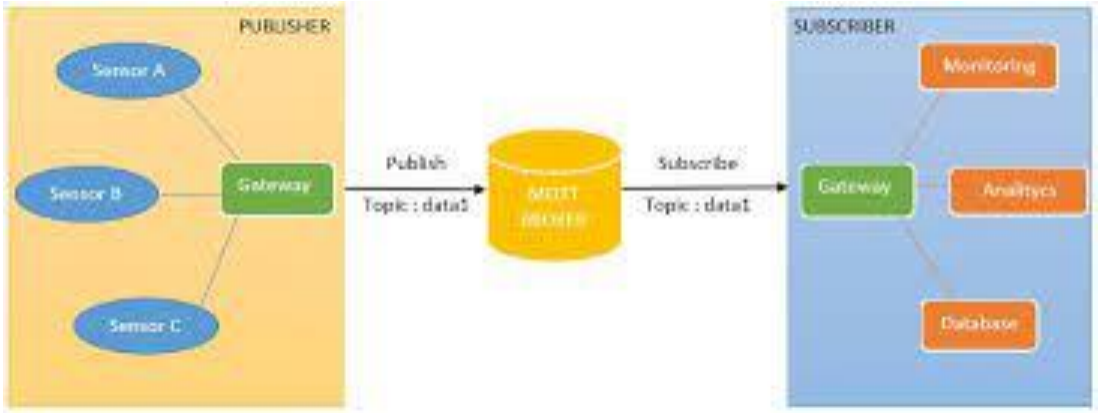
Sumber : https://www.easyelectronics.in

Sensor ultrasonik merupakan sensor yang menggunakan gelombang ultrasonik. Gelombang ultrasonik yaitu gelombang yang umum digunakan untuk mendeteksi keberadaan suatu benda dengan memperkirakan jarak antara sensor dan benda tersebut. Sensor ini berfungsi untuk mengubah besaran fisis (bunyi) menjadi besaran listrik begitu pula sebaliknya. Gelombang ultrasonik memiliki frekuensi sebesar 20.000 Hz.

Cara kerja sensor ultrasonik berdasarkan prinsip dari pantulan suatu gelombang suara sehingga dapat dipakai untuk menafsirkan jarak suatu benda menggunakan frekuensi tertentu. Gelombang ultrasonik dibangkitkan melalui sebuah alat yang disebut dengan piezoelektrik menggunakan frekuensi tertentu. Piezoelektrik tersebut akan menghasilkan gelombang ultrasonik dengan frekuensi 40KHz ketika sebuah osilator diterapkan pada benda tersebut.

Pada umumnya, alat tersebut akan menembakkan gelombang ultrasonik menuju suatu area atau suatu target. Ketika gelombang sudah menyentuh permukaan target, maka target akan memantulkan kembali gelombang tersebut. Kemudian gelombang pantulan yang dihasilkan dari target akan ditangkap oleh sensor. Setelah itu, sensor akan menghitung selisih antara waktu pengiriman gelombang dan waktu gelombang pantul yang diterima.

2.1.3 MQTT Java Script



Gambar 2. 3 Cara Kerja MQTT Java Script

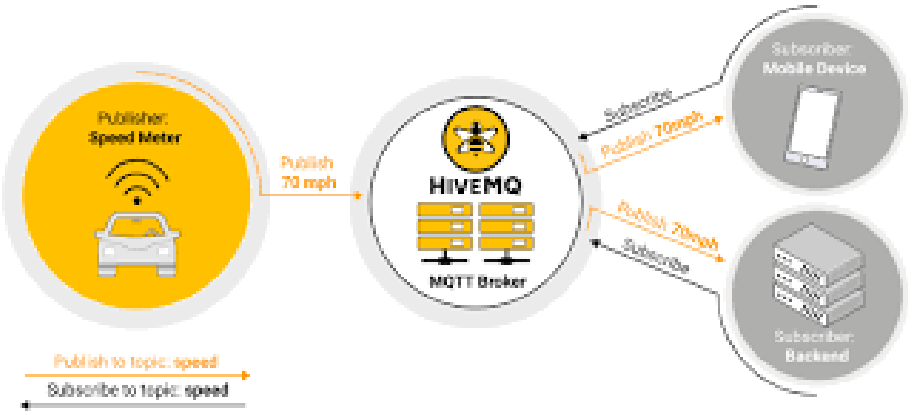
Sumber : tpsearchtool.com

Message Queuing Telemetry Transport, adalah sebuah protokol komunikasi ringan yang dirancang untuk mendukung pertukaran pesan antara perangkat di lingkungan Internet of Things (IoT) atau sistem terdistribusi. Protokol ini mengadopsi

model komunikasi publish-subscribe, di mana perangkat atau aplikasi yang bertindak sebagai penerbit (publisher) mengirimkan pesan ke suatu topik, dan perangkat atau aplikasi yang bertindak sebagai pelanggan (subscriber) menerima pesan dari topik tersebut. MQTT dirancang untuk beroperasi secara efisien di bawah kondisi jaringan yang tidak stabil atau berkecepatan rendah, dan menawarkan fleksibilitas dalam menangani skenario dengan keterbatasan sumber daya.

Cara kerja MQTT melibatkan beberapa komponen utama. Penerbit (publisher) mengirimkan pesan ke broker MQTT, yang bertindak sebagai perantara pusat. Broker ini memastikan pengiriman pesan ke pelanggan (subscriber) yang telah berlangganan pada topik yang sesuai. Pesan-pesan dapat dikirim dengan tingkat kualitas layanan (QoS) yang berbeda, memungkinkan untuk menyesuaikan kehandalan pengiriman. Selain itu, MQTT mendukung fitur-fitur seperti Last Will and Testament (LWT) dan retained messages untuk meningkatkan keandalan dan manajemen pesan. Kesederhanaan dan efisiensi protokol ini membuatnya sangat cocok untuk aplikasi IoT dan lingkungan di mana perangkat memiliki keterbatasan sumber daya.

2.1.4 Hive MQ



Gambar 2. 4 Pengertian HiveMQ

Sumber : IoT: HiveMQ startet Cloud-Angebot für MQTT | heise online

HiveMQ adalah sebuah platform MQTT (Message Queuing Telemetry Transport) yang canggih dan dapat diandalkan, dirancang untuk mendukung komunikasi di lingkungan Internet of Things (IoT) dan aplikasi terdistribusi. MQTT sendiri adalah protokol komunikasi ringan yang memungkinkan perangkat atau aplikasi untuk bertukar

pesan dengan efisien di bawah kondisi jaringan yang berbeda, dan HiveMQ berfungsi sebagai broker MQTT yang memfasilitasi pertukaran pesan antara penerbit (publisher) dan pelanggan (subscriber). Platform ini menawarkan keandalan tinggi, skalabilitas, serta fitur-fitur keamanan dan manajemen yang kuat, menjadikannya solusi yang sangat sesuai untuk mengelola komunikasi dalam ekosistem IoT yang kompleks.

Cara kerja HiveMQ mirip dengan protokol MQTT secara umum. Perangkat atau aplikasi yang bertindak sebagai penerbit mengirimkan pesan ke broker HiveMQ dengan menyertakan informasi tentang topik pesan. Selanjutnya, perangkat atau aplikasi yang berperan sebagai pelanggan dapat berlangganan pada topik tertentu, dan HiveMQ memastikan pengiriman pesan tersebut kepada pelanggan yang sesuai. HiveMQ mendukung tingkat kualitas layanan (QoS) yang berbeda, menjadikannya fleksibel dalam menangani persyaratan keandalan berbeda. Dengan keandalan tinggi, dukungan untuk skenario ketersediaan tinggi, dan kemampuan untuk berintegrasi dengan berbagai infrastruktur, HiveMQ menjadi pilihan yang populer dalam pengelolaan komunikasi di lingkungan IoT dan aplikasi terdistribusi.

2.1.5 Express Java Script.

Gambar 2.5 E*xpress Java Script*

Sumber : 3 Express.js features you need to know | by Louis Petrik | JavaScript in Plain English

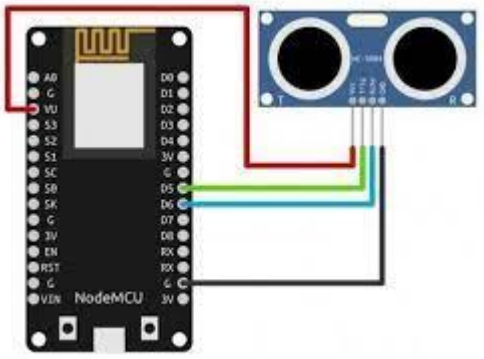
Adalah framework web app untuk Node.js yang ditulis dengan bahasa pemrograman JavaScript. Framework open source ini dibuat oleh TJ Holowaychuk pada tahun 2010 lalu. Express.js adalah framework back end. Artinya, ia bertanggung jawab untuk mengatur fungsionalitas website, seperti pengelolaan routing dan session, permintaan HTTP, penanganan error, serta pertukaran data di server.

Express.js di JavaScript adalah framework yang menggunakan pendekatan Unopinionated dalam proses pengembangan. Artinya, pengguna punya kebebasan dalam menentukan metode yang akan digunakan untuk mengeksekusi suatu perintah. Unopinionated ini punya beberapa kelebihan. Misalnya Anda bisa menentukan sendiri arsitektur yang akan dikembangkan. Di samping itu, ukuran framework juga cenderung lebih kecil, sehingga performanya jauh lebih cepat.

BAB III

METODOLOGI

3.1 Rancangan Hardware



Gambar 3. 1 Skematik Rangkaian Alat Pengukur Jarak

Sumber : Foto Pribadi Praktikum

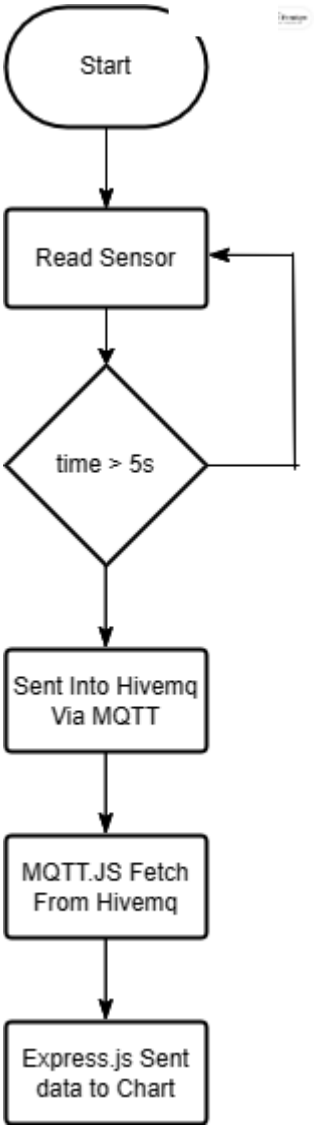
Berikut ini adalah tabel konfigurasi pin dari sensor – sensor yang digunakan pada skematik rangkaian alat pengukur jarak berbasis IoT menurut gambar 3.1di atas. Hal ini penting untuk dipastikan bahwa sambungan konfigurasi pin sensor sesuai agar dapat mendeteksi dan bekerja sesuai yang diinginkan.

Tabel 3. 1 Konfigurasi Pin Sensor Ultrasonik HC-SR04

Konfigurasi Pin Sensor Ultrasonik	
VCC	3V
Trigg	D4
Echo	D3
GND	GND

3.1 Rancangan Software

Berikut merupakan diagram alir dari rangkaian alat pengukur jarak menggunakan sensor ultrasonik hc-sr04 berbasis protokol MQTT menggunakan Hivemq



Gambar 3. 1 Skematik Rangkaian Alat Pengukur Jarak

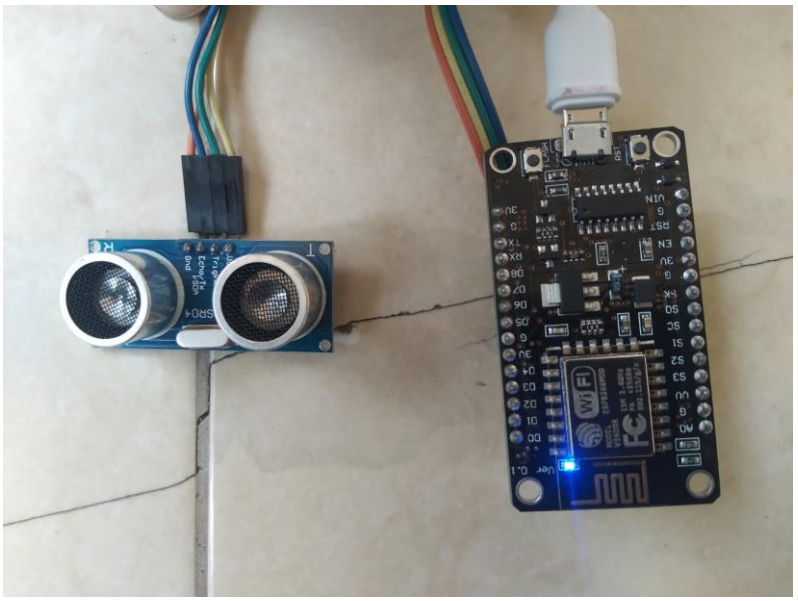
Sumber : Foto Pribadi Praktikum

BAB IV

HASIL DAN PEMBAHASAN

4.1 Cara Kerja

Alat ini dirancang dengan tujuan untuk dapat memonitoring jarak di suatu tempat secara *realtime* melalui Sensor ultrasonic HC-SR04. Ketika telah melewati 5 detik Esp8266 akan mengirimkan data jarak dengan protokol MQTT pada Broker MQTT Hivemq, Monitoring dapat dilakukan dimana saja selama memiliki internet dengan cara memasuki Web dengan MQTT js dan Express.js yang telah dibuat Data yang diterima akan ditampilkan pada Grafik dan akan terus bertambah ketika terdapat publish dari Esp8266.

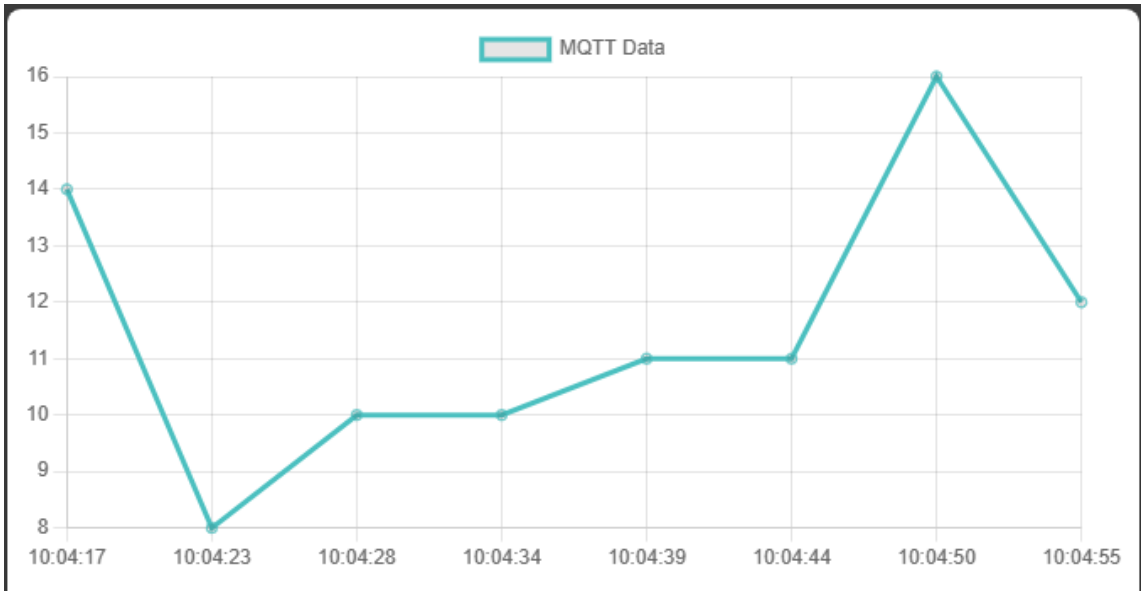


Gambar 4. 1 Rangkaian Hardware Pengukur Jarak

Sumber : Foto Pribadi Praktikum

4.2 Hasil

Grafik dapat dilihat dari web berikut MQTT Graph (adaptable.app) / https://mqttexpress.adaptable.app/ Nilai Grafik X-Axis adalah waktu Ketika terdapat data yang masuk dan Y-Axis adalah nilai jarak yang telah diukur oleh Sensor Ultrasonic dan ESP8266



Gambar 4. 2 Tampilan Grafik pada Web

Sumber : Foto Pribadi Praktikum

4.3 Program MQTT.js

```
const express = require('express');
const mqtt = require('mqtt');
const http = require('http');
const path = require('path');
const app = express();
const server = http.createServer(app);
const io = require('socket.io')(server);
const mqttOptions = {
  username: 'DATA SISTEM IOT',
  password: 'Datasistemiot123',
};
const mqttClient =
mqtt.connect('mqtts://34e4af2c39f947029e4d6ac853af4421.s2.eu.hivemq.cloud:888
3/', mqttOptions);
mqttClient.on('message', (topic, message) => {
  const data = String.fromCharCode.apply(null, message);
  io.emit('mqttData', data);
});
mqttClient.on('connect', () => {
  console.log('Connected to MQTT!');
  mqttClient.subscribe('esp8266_data');
});
mqttClient.on('error', (error) => {
  console.log('MQTT Error:', error);
```

```
app.use(express.static(path.join(__dirname, 'public')));
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public/index.html'));
});
io.on('connection', (socket) => {
  console.log('Socket.io connected');
});
const PORT = process.env.PORT || 3000;
server.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

4.4 Program Index

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>MQTT Graph</title>
  <script src="https://cdn.socket.io/4.1.3/socket.io.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <canvas id="mqttChart" width="400" height="200"></canvas>
  <script>
    const socket = io();
    const ctx = document.getElementById('mqttChart').getContext('2d');
    const mqttChart = new Chart(ctx, {
      type: 'line',
      data: {
        labels: [],
        datasets: [{
          label: 'MQTT Data',
          borderColor: 'rgb(75, 192, 192)',
          data: [],
        }],
      },
    });
    socket.on('mqttData', (data) => {
      const time = new Date().toLocaleTimeString();
      mqttChart.data.labels.push(time);
      console.log(data);
      mqttChart.data.datasets[0].data.push(data);
      mqttChart.update();
    });
  </script>
</body>
```

4.5 Program ESP8266

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>

const int trigPin = 2;
const int echoPin = 0;
long duration;
int distance;
const char* ssid = "Andani";
const char* password = "ayamgoreng";
const char* mqtt_server = "34e4af2c39f947029e4d6ac853af4421.s2.eu.hivemq.cloud";
const char* mqtt_username = "DATA SISTEM IOT";
const char* mqtt_password = "Datasistemiot123";
const int mqtt_port =8883;
WiFiClientSecure espClient;
PubSubClient client(espClient);

unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
static const char *root_ca PROGMEM = R"EOF(

----- CERTIFICATE-----

)EOF";
void setup_wifi() {
  delay(10);
  Serial.print("\nConnecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("\nWiFi connected\nIP address: ");
  Serial.println(WiFi.localIP());
}
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";   // Create a random client ID
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str(), mqtt_username, mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
```

```
void publishMessage(const char* topic, String payload , boolean retained){
  if (client.publish(topic, payload.c_str(), true))
      Serial.println("Message publised ["+String(topic)+"]: "+payload);
}

void setup() {
   Serial.begin(9600);
  while (!Serial) delay(1);
  setup_wifi();
  #ifdef ESP8266
    espClient.setInsecure();
  #else
    espClient.setCACert(root_ca);
  #endif
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  client.setServer(mqtt_server, mqtt_port);
}

void loop() {
  if (!client.connected()) reconnect();
  client.loop();
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;
  String data = String(distance);
  publishMessage("esp8266_data", data, true);

  delay(5000);
}
```

DAFTAR PUSTAKA

Steve. March 1, 2023. Using The JavaScript MQTT Client With Websockets. Using The JavaScript MQTT Client With Websockets (steves-internet-guide.com)

HiveMQ Team. September 20, 2023. Implementing MQTT in JavaScript. Implementing MQTT in JavaScript (hivemq.com)

Edouard Bonlieu. April 24, 2023. Using WebSockets with Socket.io and Node.js on Koyeb. Using WebSockets with Socket.io and Node.js on Koyeb - Koyeb

Koçfinans Tech. Dec 16, 2021. Socket.IO with Node.Js + Express. Socket.IO with Node.Js + Express. Nowadays, most web developers want to… | by Sude Kılıç | Koçfinans Tech | Medium