**SISTEM IOT UNTUK PENGIRIMAN DATA GAMBAR**

**Oleh**

**VIERI FAJAR FIRDAUS**

**NIM : 13521099**

**JURUSAN : TEKNIK INFORMATIKA**

A black and white drawing of an elephant

AI-generated content may be incorrect.

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO & INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**April 2025**

Latar Belakang

Perkembangan Internet of Things (IoT) yang pesat dalam beberapa tahun terakhir telah membawa perubahan signifikan dalam berbagai sektor, mulai dari rumah pintar (smart homes), pertanian pintar (smart farming), hingga pemantauan kesehatan jarak jauh (telemedicine). IoT memungkinkan perangkat yang terhubung untuk saling bertukar data dan berinteraksi secara otomatis melalui jaringan internet. Salah satu aplikasi penting IoT adalah dalam pengolahan data berbasis gambar, yang dapat digunakan dalam berbagai bidang seperti pengawasan keamanan, analisis citra medis, dan pemantauan lingkungan.

Salah satu tantangan utama dalam aplikasi IoT yang berbasis gambar adalah bagaimana menangani pengiriman data gambar secara efisien dan aman. Data gambar berukuran besar memerlukan bandwidth yang cukup besar, dan pengiriman data secara terus-menerus, seperti dalam streaming video, bisa sangat membebani jaringan jika tidak dikelola dengan baik. Oleh karena itu, pengiriman gambar melalui protokol komunikasi yang efisien, seperti MQTT (Message Queuing Telemetry Transport), sangat penting untuk memastikan bahwa data dapat dikirimkan dengan latensi yang rendah dan menggunakan bandwidth yang efisien.

Rancangan Sistem

### Kebutuhan Sistem

Sistem IoT yang dibuat memiliki beberapa fitur dibawah ini :

* Mengirimkan gambar secara real time melalui MQTT
* Melakukan enkripsi terhadap gambar
* Melihat gambar melalui dashboard

Selain itu, Penulis menggunakan ESP32 CAM yang dapat mengirimkan gambar berdasarkan hasil yang ditangkap.

### Arsitektur Sistem

Arsitektur sistem ini dibangun dengan pendekatan modular, yang terdiri dari beberapa komponen utama yang saling berinteraksi untuk mencapai tujuan pengiriman gambar secara efisien dan aman. Komponen utama dalam arsitektur ini adalah perangkat **ESP32** sebagai unit pengambil data, **MQTT** sebagai protokol komunikasi, **backend server** untuk pengelolaan data, dan **frontend** sebagai antarmuka pengguna untuk menampilkan gambar. Untuk mengelola service yang ada digunakan containerization menggunakan docker.

A screenshot of a computer

AI-generated content may be incorrect.

1. **ESP32 (Sensing System)**: Pada bagian perangkat keras, **ESP32** digunakan untuk mengambil gambar menggunakan kamera **ESP32-CAM**. Gambar yang diambil disimpan dalam buffer memori dan dikirimkan ke server melalui jaringan **WiFi** menggunakan **MQTT**. ESP32 juga menyertakan **timestamp** untuk setiap pengiriman gambar, yang memungkinkan server untuk melacak waktu pengambilan gambar dan pengaturannya.
2. **MQTT Broker**: Di sisi server, **MQTT** bertindak sebagai protokol komunikasi yang mengelola pengiriman data gambar. **MQTT broker** menerima pesan dari perangkat ESP32 yang berisi gambar yang telah dienkode dalam format **Base64**, yang kemudian diteruskan ke aplikasi backend. Dengan menggunakan MQTT, pengiriman data dapat dilakukan dengan efisien dan dalam mode **publish-subscribe**, sehingga mengurangi overhead komunikasi dan memungkinkan sistem untuk skalabilitas lebih tinggi.
3. **Backend Server (Flask API)**: Server backend dibangun menggunakan framework **Flask**. Server ini menerima gambar yang dikirimkan oleh ESP32, kemudian melakukan **enkripsi** menggunakan algoritma **AES** untuk memastikan keamanan data. Gambar yang telah dienkripsi kemudian disimpan dalam **database MySQL**. Backend juga menyediakan API untuk mengakses gambar yang telah disimpan, termasuk endpoint untuk mendekripsi gambar saat diminta oleh pengguna. API ini memungkinkan pengguna untuk melakukan **pagination** dan **sorting** gambar berdasarkan timestamp, serta mendownload gambar yang telah didekripsi.
4. **MySQL Database**: Gambar yang dikirim oleh ESP32 dan dienkripsi di backend disimpan dalam **database MySQL**. Database ini menyimpan gambar dalam bentuk binary dan menyertakan metadata seperti timestamp untuk memudahkan pencarian dan pengelolaan gambar yang diterima. Dengan menggunakan database yang terstruktur, sistem dapat dengan mudah mengelola gambar yang telah dikirimkan oleh banyak perangkat IoT secara efisien.
5. **Frontend (React Dashboard)**: Aplikasi **frontend** dibangun menggunakan **React.js** untuk menyediakan antarmuka pengguna yang interaktif dan responsif. Pengguna dapat melihat gambar yang telah didekripsi melalui dashboard yang menampilkan gambar-gambar dalam bentuk galeri. Frontend juga menyediakan fitur **pagination** dan **sorting** berdasarkan timestamp, memungkinkan pengguna untuk menavigasi koleksi gambar yang lebih besar dengan mudah. Gambar yang diambil melalui **React Router** diambil dari backend yang mendekripsinya sebelum ditampilkan ke pengguna.

Membuat program pada ESP32 untuk:

* Menyalakan LED dengan frekuensi awal 10 Hz.
* Menambah frekuensi saat tombol ditekan.
* Mengirim data [NIM] : [Frekuensi LED] ke MQTT server secara periodik.
* Menampilkan data tersebut melalui MQTT subscriber.

## Alat dan Bahan

* Board: ESP32
* IDE: Arduino IDE
* Library: WiFi.h, PubSubClient.h

Komponen:

* LED
* Resistor
* Push Button
* Kabel jumper
* Breadboard
* MQTT Server: Mosquitto (Windows), Subscriber menggunakan Postman

## Rangkaian

A circuit board with wires

AI-generated content may be incorrect.

Untuk komponen :

* LED di GPIO 23
* Push Button di GPIO 22

## Program

|  |
| --- |
| #include <WiFi.h>  #include <PubSubClient.h>  #define LED\_PIN 23  #define BUTTON\_PIN 22  #define NIM "13521099"  const char\* ssid = "\*";  const char\* password = "\*";  const char\* mqtt\_server = "\*";  const int mqtt\_port = 1883;  const char\* mqtt\_user = "\*";  const char\* mqtt\_password = "\*";  const char\* mqtt\_topic = "tugas1/led\_freq";  WiFiClient espClient;  PubSubClient client(espClient);  int frequency = 10;  unsigned long lastPublishTime = 0;  bool lastButtonState = HIGH;  void reconnect() {    while (!client.connected()) {      Serial.print("Menghubungkan ke MQTT...");      if (client.connect("ESP32\_Client", mqtt\_user, mqtt\_password)) {        Serial.println(" Terhubung!");      } else {        Serial.print(" Gagal, rc=");        Serial.print(client.state());        Serial.println(" Coba lagi dalam 5 detik...");        delay(5000);      }    }  }  void setup() {    Serial.begin(115200);    pinMode(LED\_PIN, OUTPUT);    pinMode(BUTTON\_PIN, INPUT\_PULLUP);    Serial.print("Menghubungkan ke WiFi...");    WiFi.begin(ssid, password);    while (WiFi.status() != WL\_CONNECTED) {      delay(500);      Serial.print(".");    }    Serial.println(" Terhubung!");    client.setServer(mqtt\_server, mqtt\_port);  }  void loop() {    if (!client.connected()) {      reconnect();    }    client.loop();    bool buttonState = digitalRead(BUTTON\_PIN);    if (buttonState == LOW && lastButtonState == HIGH) {      frequency++;      Serial.print("Frekuensi LED: ");      Serial.println(frequency);    }    lastButtonState = buttonState;    static unsigned long lastToggleTime = 0;    unsigned long interval = 1000 / (frequency \* 2);    if (millis() - lastToggleTime >= interval) {      lastToggleTime = millis();      digitalWrite(LED\_PIN, !digitalRead(LED\_PIN));    }    if (millis() - lastPublishTime > 5000) {      lastPublishTime = millis();      String message = String(NIM) + " : " + String(frequency);      client.publish(mqtt\_topic, message.c\_str());      Serial.println("Data dikirim ke MQTT: " + message);    }  } |

Penjelasan

* **Frekuensi awal** LED adalah 10 Hz.
* **Jika tombol ditekan**, frekuensi naik 1 Hz.
* Setiap 5 detik, data NIM : Frekuensi dikirim ke **MQTT topic** tugas1/led\_freq.

## Hasil Percobaan

### Awal Koneksi

|  |
| --- |
| 13:24:27.438 -> .... Terhubung!  13:24:28.941 -> Menghubungkan ke MQTT... Terhubung!  13:24:31.346 -> Data dikirim ke MQTT: 13521099 : 10  13:24:36.349 -> Data dikirim ke MQTT: 13521099 : 10  13:24:41.335 -> Data dikirim ke MQTT: 13521099 : 10  13:24:46.318 -> Data dikirim ke MQTT: 13521099 : 10  13:24:51.334 -> Data dikirim ke MQTT: 13521099 : 10 |

Hasil dari subscriber

A screenshot of a computer

AI-generated content may be incorrect.

### Percobaan 1

Ketika awal set up koneksi dan dicoba dengan 3 kali tekan

* Tekanan frekuensi 11 Hz di pukul 13:27:57 dan dipublish melalui ESP pada pukul 13:28:01 diterima oleh subscriber pada pukul 13:28:01
* Tekanan frekuensi 12 Hz di pukul 13:28:06 dan dipublish melalui ESP pada pukul 13:28:06 diterima oleh subscriber pada pukul 13:28:06
* Tekanan frekuensi 13 Hz di pukul 13:28:11 dan dipublish melalui ESP pada pukul 13:28:11 diterima oleh subscriber pada pukul 13:28:11

Console dari Arduino IDE

### A white screen with black text AI-generated content may be incorrect.

Hasil dari subscriber

A screenshot of a computer

AI-generated content may be incorrect.

Hasil pada ESP32

A circuit board with wires and a red light

AI-generated content may be incorrect.

### Percobaan 2

Ketika awal set up koneksi dan dicoba dengan 2 kali tekan

* Tekanan frekuensi 14 Hz di pukul 13:30:56 dan dipublish melalui ESP pada pukul 13:30:56 diterima oleh subscriber pada pukul 13:30:56
* Tekanan frekuensi 15 Hz di pukul 13:31:02 dan dipublish melalui ESP pada pukul 13:31:02 diterima oleh subscriber pada pukul 13:31:02

Console dari Arduino IDE

A screen shot of a computer

AI-generated content may be incorrect.

Hasil pada Subscriber

A screenshot of a computer

AI-generated content may be incorrect.

## Kesimpulan

* ESP32 dapat mengendalikan LED dengan frekuensi dinamis.
* Data dapat dikirim ke MQTT server menggunakan format [NIM] : [Frekuensi].
* Kombinasi sensor input (button), kontrol output (LED), dan IoT (MQTT) dapat diintegrasikan dengan baik.