

# **UTS IF4051 Pengembangan Sistem IoT**



Disusun oleh:

Dhanika Novlisariyanti

13521132

**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2025**

## DAFTAR ISI

<b>I. Pendahuluan</b>	<b>3</b>
A. Latar Belakang	3
B. Deskripsi Tugas	3
<b>II. Deskripsi Sistem</b>	<b>4</b>
A. Arsitektur End-to-End	4
Gambar I.A Arsitektur IoT End-to-End	4
B. Rancangan Hardware	5
Gambar I.B ESP32 DEVKIT V1	5
C. Software	5
Gambar I.B Alur Software	5
<b>III. Pengujian dan Analisis</b>	<b>7</b>
Gambar III.A Serial Monitor Arduino Sketch	7
Gambar III.B Log Kontainer Subscriber	7
Gambar III.C Aplikasi Dashboard I	8
Gambar III.C Aplikasi Dashboard II	8
<b>IV. Kesimpulan</b>	<b>9</b>
<b>V. Lampiran</b>	<b>9</b>

## I. Pendahuluan

### A. Latar Belakang

Di era digital saat ini, jumlah perangkat yang terhubung ke internet terus mengalami peningkatan. Perangkat-perangkat ini mampu berkomunikasi satu sama lain dan menghasilkan volume data yang sangat besar, mendukung berbagai kebutuhan digitalisasi. Seiring dengan itu, kebutuhan akan pengelolaan data yang cepat, aman, dan efisien menjadi semakin krusial. Konsep ini dikenal dengan istilah Internet of Things (IoT).

Seiring dengan pesatnya perkembangan Artificial Intelligence (AI), teknologi IoT tidak hanya berfungsi untuk mengumpulkan dan mentransmisikan data, tetapi dikombinasikan dengan AI untuk melakukan pemrosesan langsung di perangkat edge. Integrasi ini memungkinkan perangkat untuk melakukan analisis dan pemrosesan data secara lokal tanpa bergantung pada server pusat. Namun, kombinasi antara IoT dan AI membawa tantangan baru, terutama dalam hal kebutuhan data. Volume data yang dihasilkan akan jauh lebih besar dan kompleks, sehingga memerlukan sistem pemrosesan yang lebih efisien dan skalabel.

Pada tugas ini, akan dilakukan perancangan dan implementasi sebuah sistem berbasis Internet of Things (IoT), di mana perangkat end devices bertugas untuk mengirimkan data berupa gambar ke sistem cloud. Tujuan dari perancangan sistem ini untuk memastikan pengiriman data gambar dari perangkat ke cloud dapat dilakukan secara efisien serta memungkinkan data untuk diolah dan divisualisasikan melalui aplikasi dashboard.

### B. Deskripsi Tugas

Sistem IoT yang dirancang akan berbasis sensor image (video camera) dengan spesifikasi sebagai berikut:

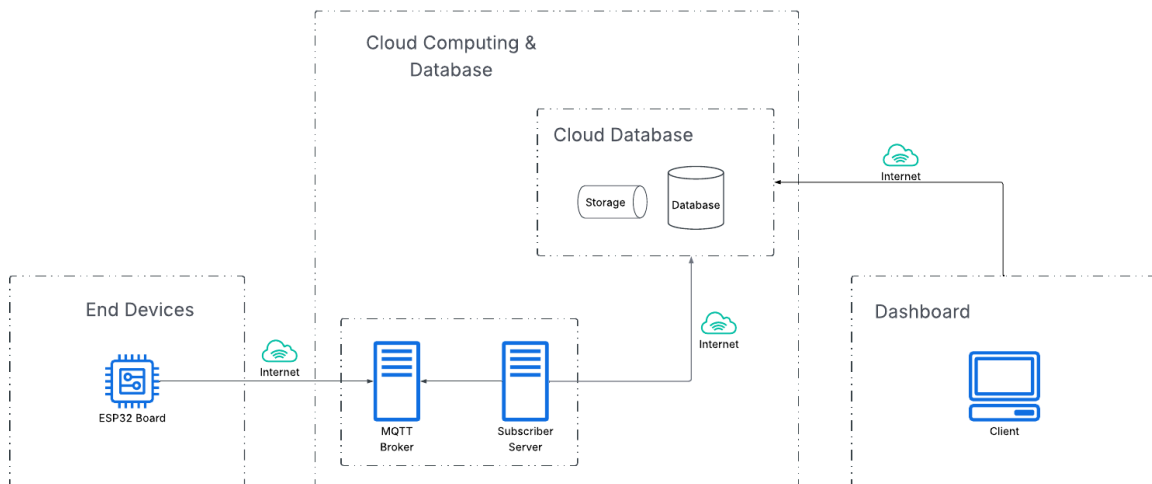
1. Unit data akuisisi dapat mengambil data berupa gambar, tetapi pada tugas ini data gambar sudah tersedia di memory buffer microprocessor.
2. Microprocessor akan mengirimkan data berupa:
  - a. Image berukuran resolusi 640x480 pixel,
  - b. Metadata image seperti timestamp.
3. Pengiriman data gambar dilakukan melalui modul komunikasi WiFi dengan protokol MQTT. Pengiriman data dilakukan dengan interval **T** second selama **K** kali pengiriman
  - a. Nilai T diambil dari  $\text{mod}(xyz, 10) + 1$

- b. K menggunakan nilai-nilai berikut {10, 20, 100}
- c. *xyz* adalah 3 digit terakhir NIM
- 4. Data yang diterima akan disimpan dalam database.
- 5. Data akan melalui pemrosesan dan dapat diakses oleh user dan ditampilkan pada sebuah aplikasi/dashboard.

## II. Deskripsi Sistem

Berikut adalah sistem yang dikembangkan berdasarkan deskripsi tugas yang diberikan.

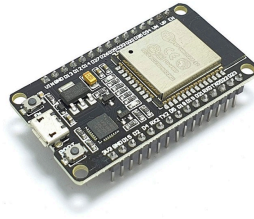
### A. Arsitektur End-to-End



Gambar I.A Arsitektur IoT End-to-End

Arsitektur IoT secara end-to-end dapat dilihat pada Gambar I.A. Arsitektur terdiri dari tiga layer, yaitu **layer end-devices/edge**, **layer cloud**, dan **layer applications**. Pada **layer end devices**, terdapat ESP32 Board yang akan mengirimkan data gambar melalui jaringan WiFi dengan protokol MQTT. Data tersebut diproses oleh **layer cloud** untuk disimpan ke dalam database dan storage. Selanjutnya, pada **lapisan applications**, data tersebut diolah dan disajikan melalui **dashboard**.

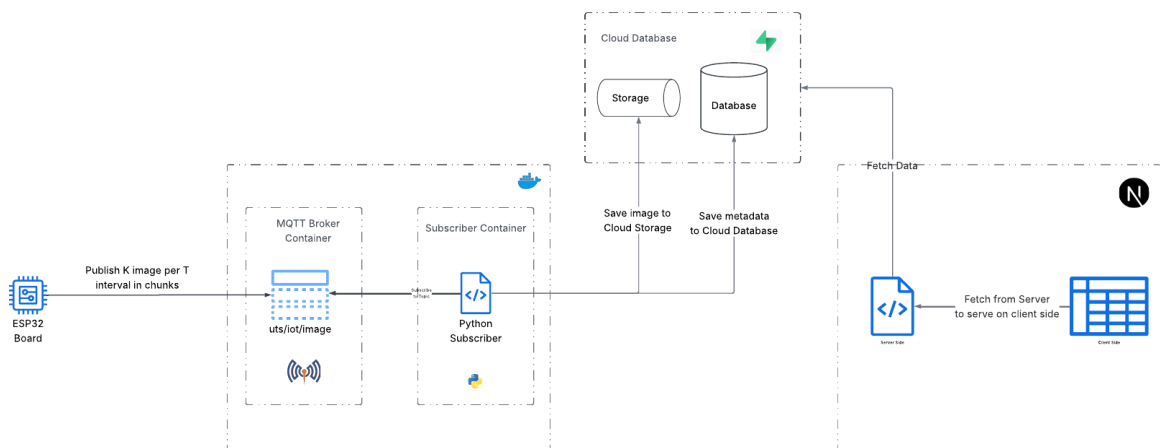
## B. Hardware



Gambar I.B ESP32 DEVKIT V1

Hardware yang digunakan merupakan ESP32 DEVKIT V1 dan tidak ada sensor yang terhubung. Data gambar diasumsikan sudah berada pada memori buffer. Board hanya dihubungkan ke *power source* untuk mengirimkan data gambar melalui jaringan WiFi dengan publish ke topic MQTT “*uts/iot/image*”.

## C. Software



Gambar I.B Alur Software

Saat ESP32 terhubung ke sumber daya listrik, perangkat akan secara otomatis mengirimkan gambar selama koneksi ke jaringan WiFi dan broker MQTT tersedia. Kontainer MQTT broker dan subscriber dijalankan secara lokal di masing-masing komputer untuk menangani pengiriman dan penerimaan data. Sementara itu, aplikasi dashboard telah di deploy melalui platform Vercel dan terhubung langsung ke cloud database untuk mengambil dan menampilkan data secara real-time.

### a. ESP32 Board

ESP32 menggunakan beberapa pustaka tambahan untuk mendukung proses komputasi seperti WiFi.h untuk modul WiFi, PubSubClient.h untuk MQTT, dan

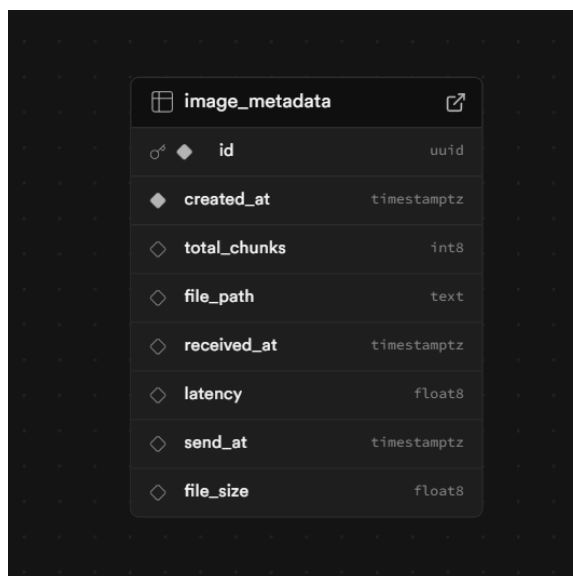
time.h untuk waktu. ESP32 akan melakukan **setup** terlebih dahulu untuk melakukan koneksi WiFi, MQTT, dan memulai waktu. Setelah itu, ESP32 akan melakukan **loop** untuk mengirim **K** data gambar per **T** interval waktu. Variabel K dan T di hard code dan tidak ada input untuk mengubahnya secara dinamis sehingga jika diubah harus melakukan upload kode kembali ke ESP32.

Data gambar diambil dari file **image\_array.h** dan dikirimkan dalam beberapa bagian. Setiap bagian memiliki ukuran **8192 bytes**, dengan batas maksimum payload pada pustaka **PubSubClient** disesuaikan menjadi **20480**. Pengaturan ini bertujuan untuk menghindari beban komputasi yang terlalu besar pada **ESP32**, yang dapat berisiko menyebabkan kegagalan dalam proses pengiriman gambar.

b. Computing

Pengiriman data dilakukan oleh ESP32 melalui jaringan WiFi menggunakan protokol MQTT. Implementasi MQTT broker dijalankan dalam lingkungan container menggunakan Docker.

Sementara itu, subscriber yang berlangganan pada topik yang sama juga di kontainerisasi menggunakan Python, dan berfungsi untuk memproses serta menyimpan data ke dalam database dan storage. Untuk schema database sendiri seperti gambar di bawah.



image_metadata	
id	uuid
created_at	timestampz
total_chunks	int8
file_path	text
received_at	timestampz
latency	float8
send_at	timestampz
file_size	float8

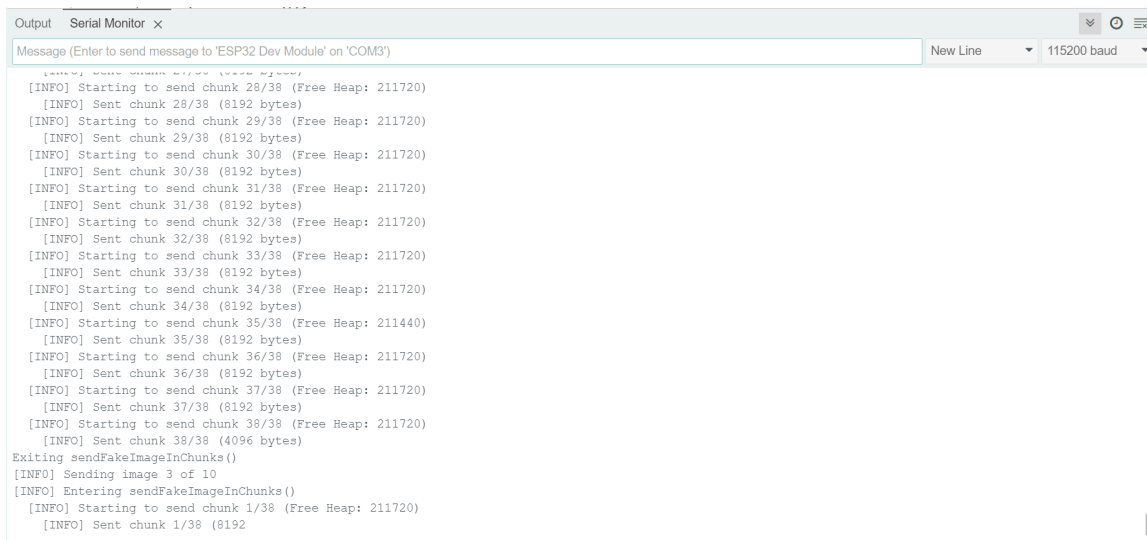
Gambar 1.B Skema Database

c. Aplikasi Dashboard

Aplikasi dashboard dikembangkan menggunakan NextJS 15 menggunakan tambahan pustaka tailwind dan ShadCN untuk *styling*. Data yang ditampilkan pada dashboard dipanggil menggunakan server action yang disediakan oleh NextJS sehingga tidak perlu set up *back-end* kembali.

Aplikasi hanya terdiri dari satu halaman yang menunjukkan tabel berisi data gambar yang sudah dikirim. Aplikasi dashboard di deploy menggunakan vercel dan dapat diakses pada [halaman ini](#) .

### III. Pengujian dan Analisis

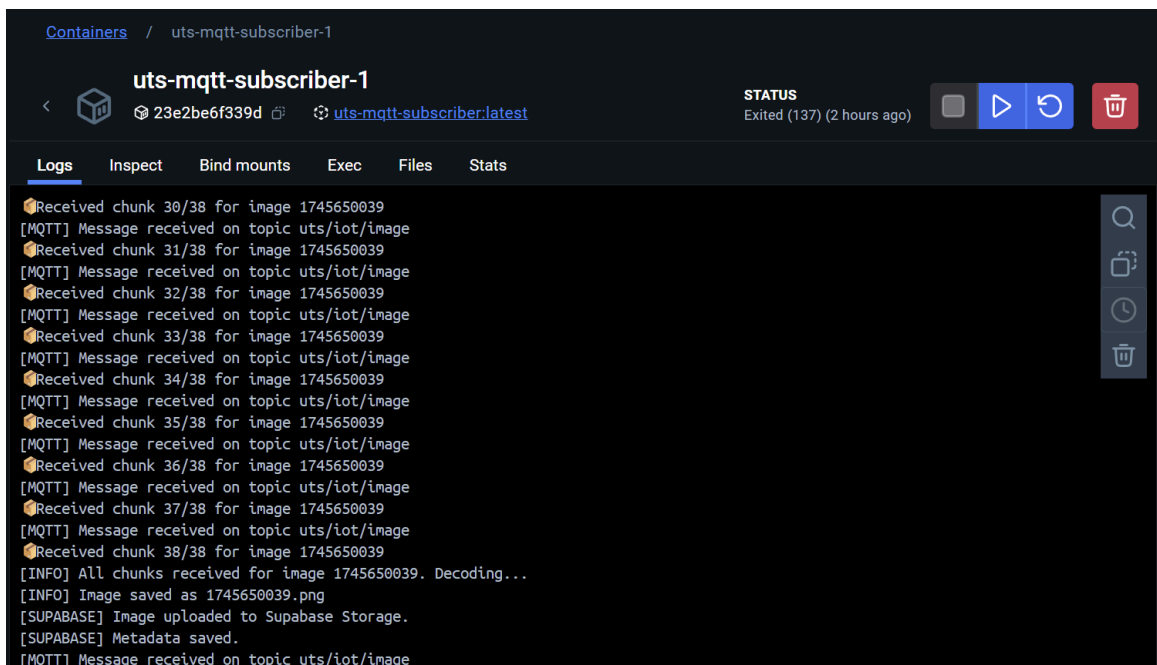


```
Output Serial Monitor x
Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')
New Line 115200 baud

[INFO] Starting to send chunk 28/38 (Free Heap: 211720)
[INFO] Sent chunk 28/38 (8192 bytes)
[INFO] Starting to send chunk 29/38 (Free Heap: 211720)
[INFO] Sent chunk 29/38 (8192 bytes)
[INFO] Starting to send chunk 30/38 (Free Heap: 211720)
[INFO] Sent chunk 30/38 (8192 bytes)
[INFO] Starting to send chunk 31/38 (Free Heap: 211720)
[INFO] Sent chunk 31/38 (8192 bytes)
[INFO] Starting to send chunk 32/38 (Free Heap: 211720)
[INFO] Sent chunk 32/38 (8192 bytes)
[INFO] Starting to send chunk 33/38 (Free Heap: 211720)
[INFO] Sent chunk 33/38 (8192 bytes)
[INFO] Starting to send chunk 34/38 (Free Heap: 211720)
[INFO] Sent chunk 34/38 (8192 bytes)
[INFO] Starting to send chunk 35/38 (Free Heap: 211440)
[INFO] Sent chunk 35/38 (8192 bytes)
[INFO] Starting to send chunk 36/38 (Free Heap: 211720)
[INFO] Sent chunk 36/38 (8192 bytes)
[INFO] Starting to send chunk 37/38 (Free Heap: 211720)
[INFO] Sent chunk 37/38 (8192 bytes)
[INFO] Starting to send chunk 38/38 (Free Heap: 211720)
[INFO] Sent chunk 38/38 (4096 bytes)
Exiting sendFakeImageInChunks()
[INFO] Sending image 3 of 10
[INFO] Entering sendFakeImageInChunks()
[INFO] Starting to send chunk 1/38 (Free Heap: 211720)
[INFO] Sent chunk 1/38 (8192
```

Gambar III.A Serial Monitor Arduino Sketch

ESP32 berhasil mengirimkan data gambar melalui jaringan WiFi dengan protokol MQTT.



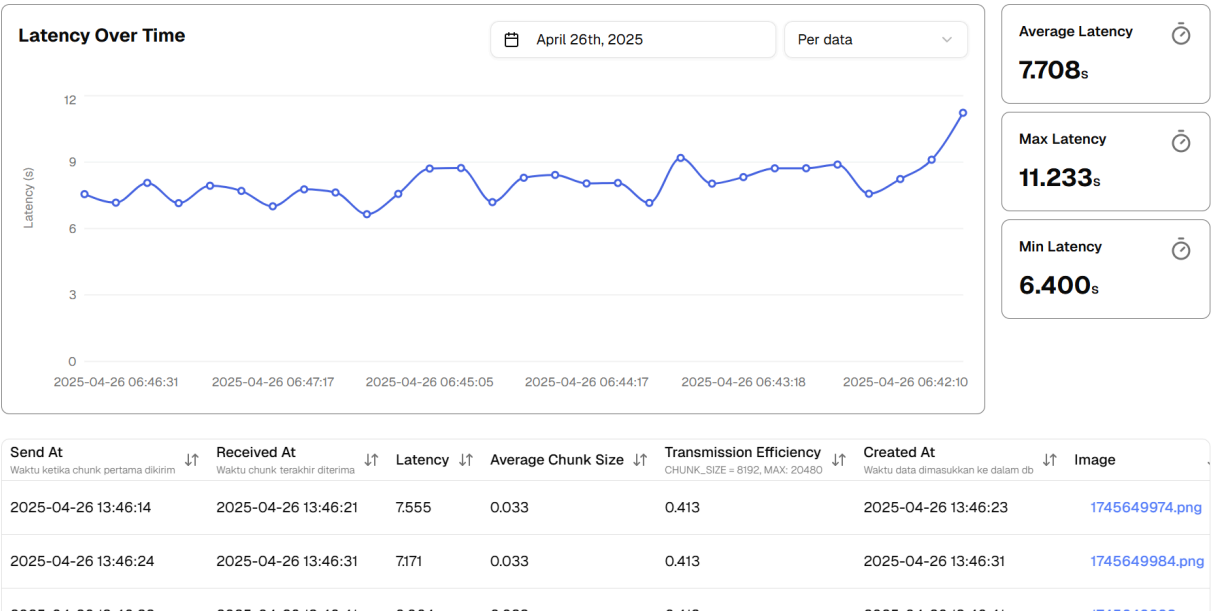
```
Containers / uts-mqtt-subscriber-1
uts-mqtt-subscriber-1
23e2bef6339d uts-mqtt-subscriber:latest
STATUS
Exited (137) (2 hours ago)
Logs Inspect Bind mounts Exec Files Stats
Received chunk 30/38 for image 1745650039
[MQTT] Message received on topic uts/iot/image
Received chunk 31/38 for image 1745650039
[MQTT] Message received on topic uts/iot/image
Received chunk 32/38 for image 1745650039
[MQTT] Message received on topic uts/iot/image
Received chunk 33/38 for image 1745650039
[MQTT] Message received on topic uts/iot/image
Received chunk 34/38 for image 1745650039
[MQTT] Message received on topic uts/iot/image
Received chunk 35/38 for image 1745650039
[MQTT] Message received on topic uts/iot/image
Received chunk 36/38 for image 1745650039
[MQTT] Message received on topic uts/iot/image
Received chunk 37/38 for image 1745650039
[MQTT] Message received on topic uts/iot/image
Received chunk 38/38 for image 1745650039
[INFO] All chunks received for image 1745650039. Decoding...
[INFO] Image saved as 1745650039.png
[SUPABASE] Image uploaded to Supabase Storage.
[SUPABASE] Metadata saved.
[MQTT] Message received on topic uts/iot/image
```

Gambar III.B Log Kontainer Subscriber

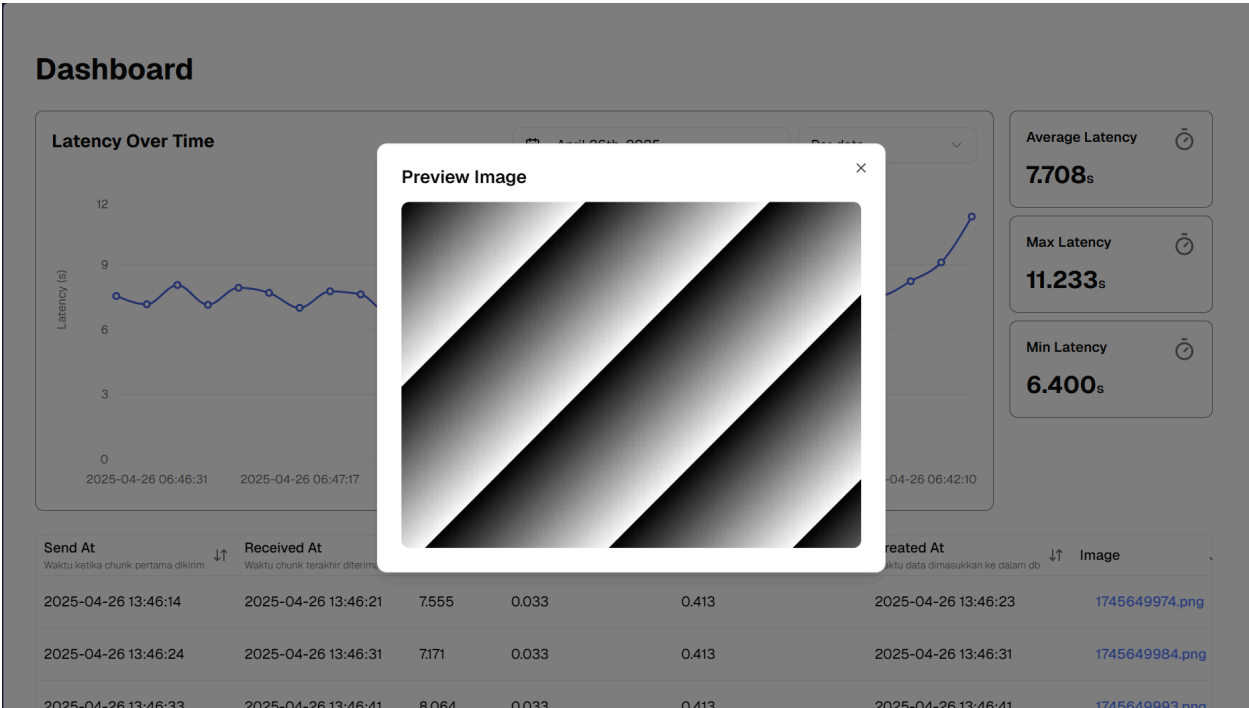
Data berhasil diterima dan disimpan ke dalam database.



Dashboard



Gambar III.C Aplikasi Dashboard I



Gambar III.C Aplikasi Dashboard II

Data yang disimpan ke dalam database berhasil ditampilkan melalui aplikasi dashboard. Berdasarkan hasil data yang diterima, latensi pengiriman gambar dengan ukuran 8120 bytes per bagian berada pada kisaran 6 hingga 7 detik hingga seluruh gambar berhasil diterima.

Waktu latensi ini masih dapat dioptimalkan lebih lanjut melalui beberapa eksperimen, seperti dengan menyesuaikan ukuran potongan data (chunk size) atau dengan melakukan kompresi pada file gambar agar ukuran data yang dikirim menjadi lebih kecil, sehingga mempercepat proses transmisi.

#### IV. Kesimpulan dan Saran

Berdasarkan hasil perancangan sistem dan pengujian yang telah dilakukan, beberapa kesimpulan yang dapat diambil adalah sebagai berikut:

1. Sistem berhasil berfungsi dengan baik dan memiliki sifat scalable, karena setiap komponen dikembangkan secara independen tanpa saling bergantung satu sama lain.
2. Proses **pengiriman data** berjalan dengan lancar, dan data yang diterima dapat disimpan di database dan storage dengan baik.

Sistem dapat diperbaiki dan dikembangkan lebih baik melalui beberapa saran berikut:

1. Meningkatkan tingkat keamanan pada broker MQTT dengan menerapkan mekanisme autentikasi, sehingga hanya pengguna yang telah terverifikasi yang diizinkan untuk melakukan koneksi ke broker.
2. Meningkatkan keamanan komunikasi pengiriman data dengan mengenkripsi data.
3. Melakukan deployment kontainer MQTT broker dan subscriber ke server cloud agar koneksi dapat berjalan secara persisten. Dengan demikian, ketika ESP32 mengirimkan gambar, perangkat dapat langsung terhubung ke broker, data dapat segera diterima dan disimpan ke dalam database, serta dashboard dapat langsung mengambil dan menampilkan data secara real-time.
4. Menambahkan fitur pengaturan interval dan jumlah image yang dikirim dari aplikasi dashboard dengan mempublish topic ke ESP32.

## **V. Lampiran**

Kode: [https://github.com/dhanikanovlisa/if4051\\_13521132\\_uts\\_iot.git](https://github.com/dhanikanovlisa/if4051_13521132_uts_iot.git)

Dashboard: <https://if4051-13521132-uts-iot.vercel.app/>