RANCANG BANGUN APLIKASI BERBASIS WEB UNTUK MANAJEMEN PEMBARUAN FIRMWARE DENGAN METODE OVER-THE-AIR

Alvaro Ariel Ilanunu[1].
[1,2,3] Prodi S1 Teknik Komputer, Fakultas Teknik Elektro, Universitas Telkom, Surabaya
[1]**alvaroarielz@.telkomuniversity.ac.id**

Abstrak

***Internet of Things* (IoT) telah menjadi bagian dari kehidupan sehari-hari dengan penerapan yang luas di berbagai industri. Salah satu elemen penting dari perangkat IoT adalah pembaruan *firmware*, yang diperlukan untuk meningkatkan fungsionalitas dan memperbaiki bug. Namun, memperbarui banyak perangkat secara manual memerlukan biaya dan waktu yang tidak sedikit. Penelitian ini merancang dan membangun aplikasi berbasis *web* untuk manajemen pembaruan *firmware* perangkat IoT menggunakan metode *Over-the-Air* (OTA). Aplikasi ini dikembangkan menggunakan Flask dan MySQL, dengan antarmuka berbasis HTML, CSS, dan JavaScript. *Firmware* diperbarui dengan mendaftarkan MAC *address* perangkat ke aplikasi, yang secara otomatis mengunduh dan menginstal *firmware* terbaru jika tersedia. Pengujian dilakukan dalam dua skenario: pembaruan satu perangkat dan multi perangkat. Hasil pengujian menunjukkan waktu pembaruan rata-rata berkisar antara 9,398 hingga 17,750 detik untuk satu perangkat, dan 11,317 hingga 13,108 detik untuk multi perangkat. Pengujian dilakukan 20 kali pada setiap perangkat untuk memastikan konsistensi keberhasilan pembaruan.**

Kata kunci : *Internet of Things*, *Over-the-Air*, *Firmware*, Aplikasi *Web*, Flask.

Abstract

The Internet of Things (IoT) has become a part of everyday life with wide applications in various industries. One of the essential elements of IoT devices is firmware updates, which are required to improve functionality and fix bugs. However, manually updating many devices is costly and time-consuming. This research designs and builds a web-based application for firmware update management of IoT devices using the Over-the-Air (OTA) method. The application is developed using Flask and MySQL, with an HTML, CSS, and JavaScript-based interface. Firmware is updated by registering the MAC address of the device to the application, which automatically downloads and installs the latest firmware when available. Tests were conducted in two scenarios: single-device and multi-device updates. The test results showed that the average update time ranged from 9.398 to 17.750 seconds for a single device, and 11.317 to 13.108 seconds for multiple devices. Tests were conducted 20 times on each device to ensure consistency of update success.

Keywords: Internet of Things, Over-the-Air, Firmware, Web Application, Flask.

1. Pendahuluan

Internet of Things (IoT) sekarang telah menjadi sangat umum karena menjadi bagian dari kehidupan sehari-hari. IoT berkembang begitu pesat dan penerapannya sangat masif karena diterapkan di berbagai tempat seperti rumah, bangunan, sampai perkotaan dan di berbagai bidang seperti transportasi, kesehatan, dan industri [1]. Ini karena IoT merevolusi cara manusia untuk berinteraksi dengan objek di sekitarnya yang digunakan sehari-hari, atau dapat dikatakan IoT sedang mendigitalkan fungsionalitas, meningkatkan efisiensi, dan meningkatkan kualitas hidup manusia. Banyak benda-benda yang dapat terhubung ke internet dan dapat dikontrol, ini berkat teknologi kecil yang diintegrasikan dengan benda-benda tersebut [2].

Dalam konteks perangkat IoT, salah satu elemen penting adalah kapasitasnya dalam pembaruan perangkat lunak atau *firmware*. Perangkat IoT dikontrol melalui jaringan internet untuk mengatur fungsionalitasnya, dimana fungsionalitas ini ditentukan oleh *firmware* yang tertanam di dalamnya. Pembaruan *firmware* merupakan proses vital dengan tujuan meningkatkan fungsionalitas dan memperbaiki bug sebagai upaya untuk memaksimalkan fungsinya dan menjaga perangkat tetap relevan dengan protokol dan standar baru [3]. Namun kesulitan muncul ketika diperlukan pembaruan pada sejumlah besar perangkat yang seringkali sulit dijangkau secara fisik. Situasi ini mengakibatkan proses pembaruan membutuhkan biaya dan memakan waktu. Untuk mengatasi hambatan ini, solusinya terletak pada penerapan pembaruan *Over-the-Air* (OTA). Pendekatan ini memungkinkan proses

pembaruan secara nirkabel, memfasilitasi penyebaran perangkat lunak atau *firmware* baru ke banyak perangkat tanpa perlu melakukan intervensi langsung pada masing-masing perangkat [4].

Proses pembaruan *firmware* dengan metode OTA akan menggunakan platform yang menghubungkan perangkat IoT dengan perangkat yang akan mengirim berkas pembaruan [5]. Tetapi, untuk memungkinkan pembaruan *firmware* pada banyak perangkat atau dengan skala yang ditingkatkan agar lebih fungsional dapat didukung dengan kemampuan manajemen aplikasinya [6].

Berdasar kondisi ini, maka pada penelitian ini penulis mengangkat tema "Rancang Bangun Aplikasi Berbasis *Web* untuk Manajemen Pembaruan *Firmware* dengan Metode *Over-the-Air*". Ini adalah sebuah platform berupa aplikasi yang berfungsi untuk memanajemen pembaruan *firmware* perangkat IoT dengan metode *Over-the-Air*. Aplikasi yang dibangun terdiri dari *script* yang ditulis menggunakan HTML, CSS, Javascript, dan *framework* bahasa pemrograman Python yaitu Flask. Untuk melakukan pembaruan *firmware* melalui aplikasi ini, MAC *address* perangkat IoT perlu didaftarkan terlebih dahulu pada aplikasi. Perangkat IoT yang akan diperbarui *firmware*-nya biasanya berupa mikrokontroler. Mikrokontroler yang MAC *address*-nya telah terdaftar di basis data aplikasi dapat melakukan pembaruan *firmware* secara nirkabel. Pengguna hanya perlu mengunggah berkas *firmware* pada mikrokontroler yang dipilih. Selanjutnya, mikrokontroler akan mengecek ketersediaan berkas *firmware* versi baru dengan mengirim *request* setiap waktu ke *server* aplikasi. Penamaan berkas *firmware* dibuat mengandung angka yang difungsikan sebagai versi *firmware* tersebut, karena mikrokontroler diprogram untuk membaca versi *firmware* dari nama berkasnya. Jika versi pada *server* aplikasi lebih kecil atau sama, maka mikrokontroler tidak merespon. Namun jika ditemukan lebih besar dari versi yang sedang berjalan di mikrokontroler, maka mikrokontroler akan mengunduh berkas *firmware* baru secara otomatis dan melakukan *self-flashing* untuk menjalankan programnya. Untuk membuktikan apakah mikrokontroler benar-benar menjalankan *firmware* versi terbaru di dalamnya, dapat diperiksa pada output yang tampil di serial monitor lingkungan pengembangan terintegrasi atau Integrated Development Environment (IDE).

2. Dasar Teori

2.1 Internet-of-Things (IoT)

Internet of Things atau IoT merujuk pada jaringan perangkat fisik yang terhubung secara digital dan saling berkomunikasi melalui internet. IoT mencakup berbagai objek seperti RFID, NFC, sensor, aktuator, ponsel, dan banyak lagi [7]. Solusi yang diberikan IoT adalah meningkatkan kemampuan berbagai industri, mulai dari skala kecil hingga besar di bidang apapun. Karena IoT memungkinkan pengumpulan data dan analisis agar pekerjaan lebih efektif dan efisien serta membantu dalam pengambilan keputusan yang tepat. Dengan IoT, perangkat dapat saling berkomunikasi dan berkolaborasi untuk memberikan layanan yang lebih cerdas dan terkoneksi [8].

2.2 Over-the-Air (OTA)

Over-the-Air atau OTA merujuk pada proses pengiriman data atau perangkat lunak secara nirkabel. Dengan menggunakan metode ini, pengguna dapat memperbarui, mengunduh, atau mentransfer data atau perangkat lunak tanpa harus menggunakan kabel atau koneksi fisik lainnya [3]. Pada konteks pembaruan *firmware*, metode ini memungkinkan pembaruan dilakukan secara jarak jauh atau nirkabel. Sehingga pengguna tidak perlu membawa produk ke pusat layanan untuk perubahan perangkat lunak [9]. Metode ini banyak diterapkan karena dianggap lebih cepat, memudahkan akses, dan tidak begitu rumit dalam penerapannya [10]. Sehingga membebaskan pengguna dari keterbatasan fisik dan memberi keleluasaan dalam mengelola perangkat IoT mereka.

2.3 *Firmware*

Firmware adalah perangkat lunak berbasis *read-only-memory* (ROM) yang disematkan pada perangkat keras untuk mengontrol fungsionalitas dan operasi perangkat tersebut. *Firmware* berfungsi sebagai jembatan antara perangkat keras dan perangkat lunak yang dijalankan oleh perangkat tersebut, memungkinkan perangkat keras untuk beroperasi sesuai dengan kebutuhan dan instruksi yang diberikan oleh pengguna atau oleh perangkat lunak yang berjalan di atasnya. *Firmware* diperbarui secara berkala, biasanya bertujuan untuk memperbaiki bug, memperbaiki masalah keamanan, meningkatkan fungsionalitas, atau bahkan mengubah cara kerja perangkat secara keseluruhan [11].

Prosedur pembaruan dimulai ketika pengembang membuat *firmware image* yang baru dan mendistribusikannya ke perangkat. *Flash memory* pada perangkat IoT terbagi atas *memory regions* (*slots*), yang berisi *bootloader* dan gambar-gambar (*image*) *firmware*. *Firmware* baru disimpan terlebih dahulu ke ruang yang tersedia. Lalu perangkat yang menerima pembaruan *firmware* akan melakukan *rebooting*, selanjutnya berkas

firmware baru akan menimpa berkas pada internal *flash memory* (memori program) dan perangkat mulai menjalankan *firmware* baru [12].

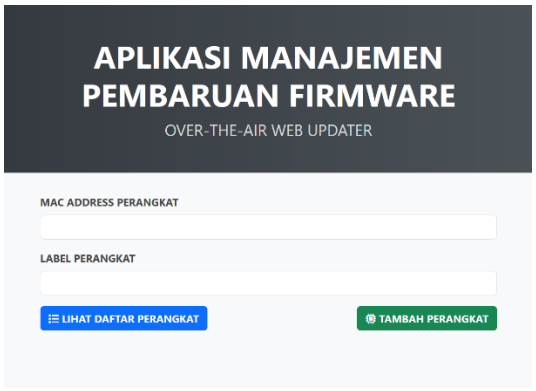
2.4 Mikrokontroler

Mikrokontroler (MCU) adalah sebuah perangkat semikonduktor yang dapat diprogram khusus untuk sistem tertanam. Perangkat ini terintegrasi dalam satu chip yang terdiri dari unit pemrosesan pusat atau *Central Processing Unit* (CPU), jam sistem, memori, dan berbagai periferal [13]. Mikrokontroler sering digunakan dalam berbagai aplikasi yang memerlukan kontrol otomatis atau interaksi dengan lingkungan. Perangkat-perangkat elektronik tersebut memerlukan mikrokontroler sebagai "otak" nya. Namun, mikrokontroler tidak dapat bekerja hanya dengan terhubung ke pencatu daya (*power supply*). Tetapi memerlukan program seperti *firmware* di dalamnya untuk menjalankan fungsinya, *firmware* biasanya ditulis dalam bahasa pemrograman C dan membutuhkan kompiler khusus untuk mengubah bahasa pemrograman ke kode mesin [14].

3. Perancangan Sistem

3.1 Desain *Front-End*

Front-end mengacu pada antarmuka pengguna (*user interface*) yang berinteraksi langsung dengan pengguna. *Front-end* aplikasi disusun menggunakan skrip HTML, CSS, dan Javascript. Aplikasi *web* yang dibangun terdiri dari 3 halaman (*pages*). Halaman pertama digunakan untuk mendaftarkan perangkat, halaman kedua untuk menampilkan perangkat terdaftar dan mengunggah firmware, dan halaman ketiga untuk menampilkan riwayat pembaruan *firmware* beserta informasi dari *firmware* (tanggal, waktu, nama berkas, dan versi).

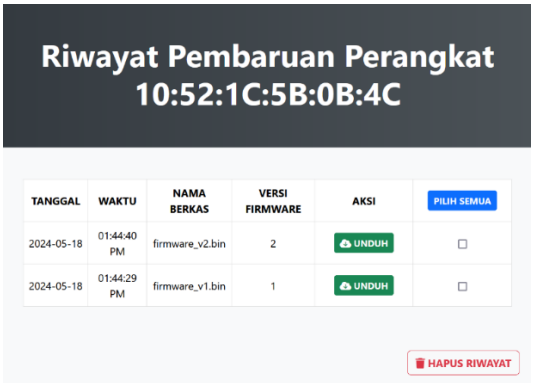


Gambar 3. 1 Halaman Registrasi Perangkat

Di halaman pertama, pengguna akan mendaftarkan MAC *address* dari mikrokontroler perangkat IoT dan memberikan label sebagai deskripsi dari perangkat tersebut.

Gambar 3. 2 Halaman Perangkat Terdaftar

Halaman kedua menampilkan perangkat IoT yang MAC *address*-nya dari mikrokontrolernya telah terdaftar. Pada halaman ini terdapat opsi untuk memperbarui *firmware* perangkat dan menghapus perangkat. Pembaruan dapat dilakukan pada satu atau beberapa perangkat sekaligus, untuk melakukannya pengguna perlu memberi centang pada perangkat yang dipilih.

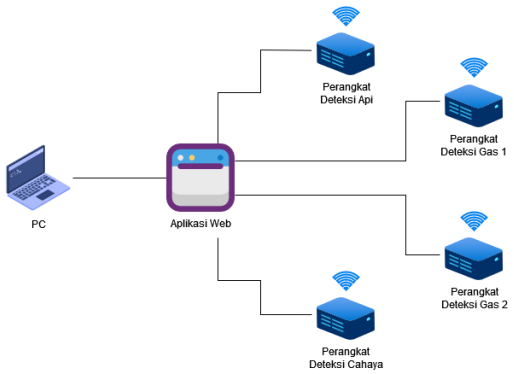


Gambar 3. 3 Halaman Riwayat Pembaruan

Riwayat berkas *firmware* yang sebelumnya telah diunggah akan ditampilkan pada halaman ini, beserta dengan tanggal dan waktu pengunggahan serta nama berkas dan versi dari *firmware* tersebut.

3.2 Cara Kerja Prototipe

Prototipe mencakup aplikasi sebagai komponen utama dan beberapa perangkat IoT seperti perangkat sensor api, perangkat sensor cahaya, dan 2 perangkat sensor gas untuk pengujian metode pembaruan. Berikut adalah skema dari prototipe.



Gambar 3. 4 Skema Prototipe

Alur kerja prototipe dimulai ketika pengguna mengakses aplikasi *web* dan mendaftarkan MAC *address* perangkat IoT. Saat pengguna memasukkan MAC *address* dan label perangkat, permintaan POST dikirim ke *endpoint /register* di *server*. Lalu, *server* menerima permintaan tersebut dan mengekstrak MAC *address* dan label perangkat dari data yang diterima. Informasi perangkat disimpan ke dalam tabel *Device* di *database*. Pengguna kemudian memilih perangkat yang ingin diperbarui *firmware*-nya dan mengunggah berkas *firmware* yang sesuai. Ketika berkas diunggah, permintaan POST dikirim ke *endpoint /upload* di *server*. Setelah menerima permintaan, *server* menangani berkas yang diunggah. Di sini, versi *firmware* diekstrak dari nama berkasnya dan tanggal serta waktu saat pengunggahannya juga dicatat. Versi ini diekstrak agar memungkinkan perangkat untuk memeriksa pembaruan dengan membandingkan versi *firmware* yang berjalan di dalamnya dengan versi terbaru yang tersedia. Lalu informasi tentang pembaruan tersebut disimpan dalam tabel *FirmwareHistory* di *database*, informasi tersebut mencakup versi, *timestamp* (tanggal dan waktu), dan nama berkas.

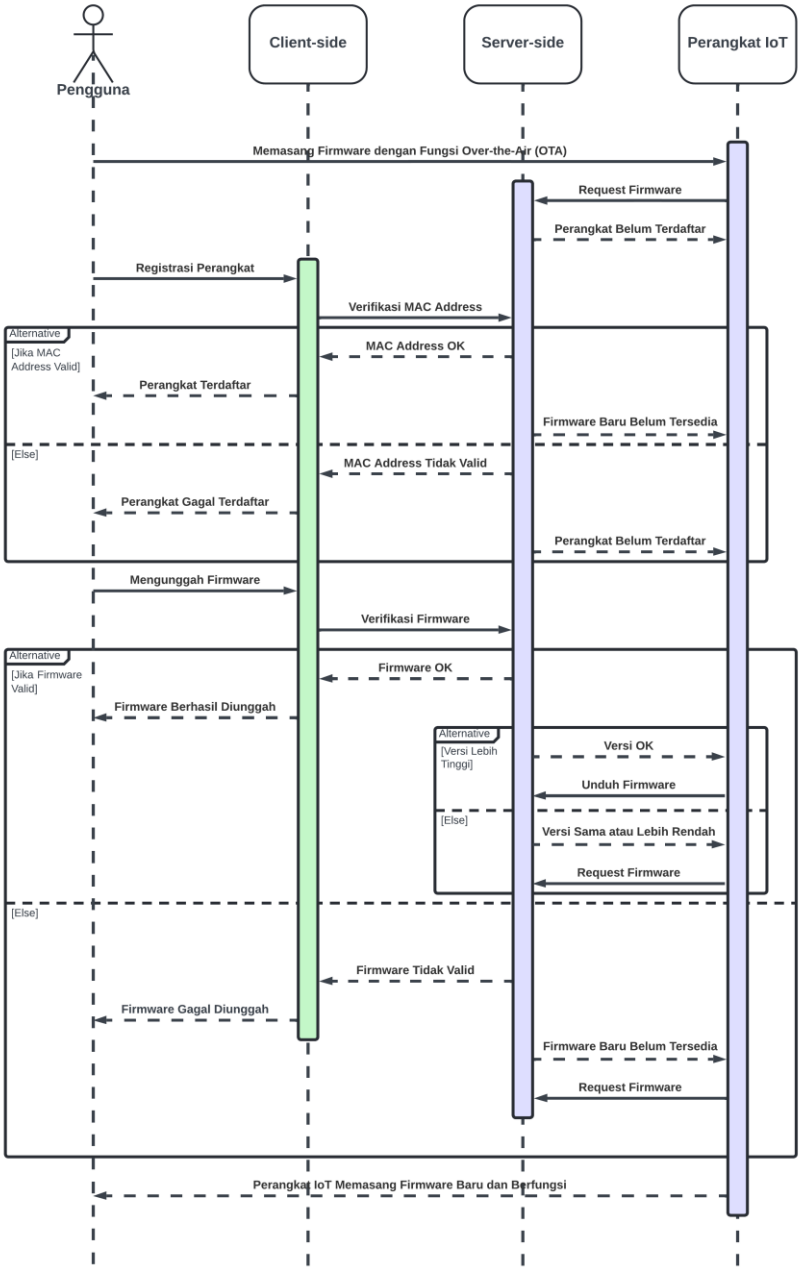
Perangkat IoT yang terkoneksi dengan jaringan Wi-Fi menjalankan fungsi *loop* secara terus menerus berdasarkan *firmware* yang telah dipasang. Jadi sebenarnya, perangkat IoT tidak bisa menjalankan fungsi OTA begitu saja. Itulah mengapa perangkat IoT perlu dipasangi *firmware* yang menjalankan fungsi OTA terlebih dahulu agar dapat melakukan pembaruan dengan metode tersebut. Sebagai tambahan, setiap kali *firmware* perangkat IoT ingin diperbarui, fungsi OTA harus selalu disertakan pada berkas *firmware* baru yang akan dipasang, agar pembaruan berikutnya dapat menggunakan metode yang sama.

Dalam setiap iterasi loop, perangkat mengirimkan permintaan GET ke *endpoint /update* di *server* dengan mencantumkan MAC *address* perangkat sebagai parameter. *Server* menerima permintaan tersebut dan memeriksa riwayat *firmware* dari perangkat tersebut. Jika ada berkas *firmware* yang tersedia (dengan versi yang lebih tinggi dari yang berjalan di perangkat), *server* akan memberikan respon yang menyertakan versi

dan URL unduhan. Lalu perangkat mengunduh *firmware* baru dari URL tersebut dan melakukan *self-flashing* untuk menerapkan *firmware* baru. Setiap kali ada pembaruan *firmware*, entri baru dibuat dalam tabel *FirmwareHistory* di *database*. Informasi seperti MAC *address* perangkat, versi *firmware*, *timestamp*, dan nama berkas *firmware* disimpan dalam entri tersebut. Ini memungkinkan aplikasi melacak riwayat pembaruan *firmware* dari perangkat yang terdaftar. Terdapat juga opsi untuk mengunduh berkas pada riwayat jika sewaktu-waktu berkas tersebut dibutuhkan kembali.

3.3 Diagram Urutan (*Sequence* Diagram)

Diagram urutan (*sequence* diagram) disusun untuk menunjukkan interaksi antara objek atau komponen sistem dalam urutan waktu.

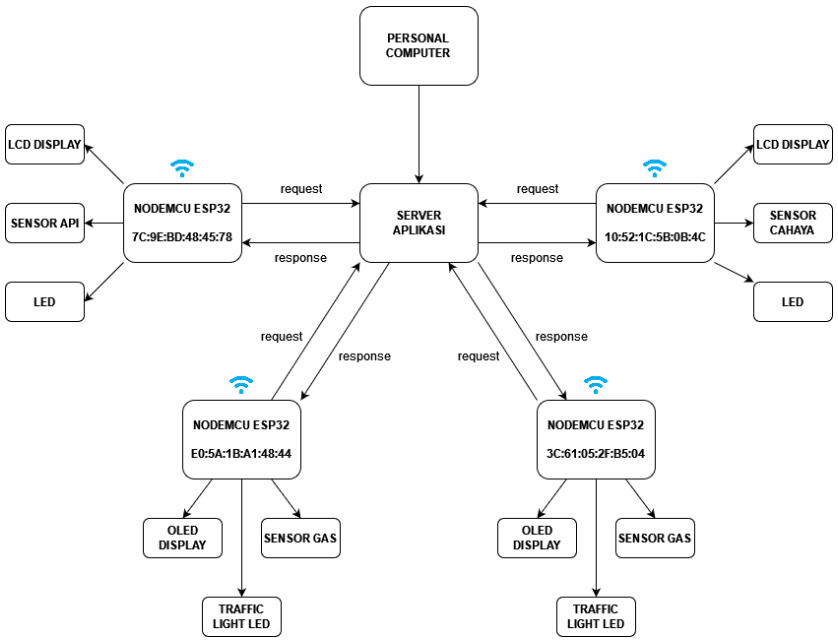


Gambar 3. 5 Diagram Urutan (*Sequence* Diagram)

Diagram terdiri dari pengguna (*actor*) dan 3 objek (*client-side*, *server-side*, dan perangkat IoT) yang memberi interaksi satu sama lain. Di sini dapat dilihat bahwa perangkat IoT adalah objek paling aktif dibanding objek lain dalam urutan ini, karena memiliki peran atau aktivitas paling banyak.

4. Implementasi dan Analisis

Implementasi merupakan tahap penerapan sistem yang sudah dirancang sebelumnya. Tahap ini merupakan aktivitas untuk menguji prototipe dan memperoleh data hasil pengujiannya.



Gambar 4. 1 Diagram Blok Sistem

Diagram tersebut menunjukkan setiap perangkat IoT memiliki NodeMCU ESP32 sebagai mikrokontrolernya. Mikrokontroler inilah yang akan terhubung dengan *server* aplikasi untuk mengirim permintaan (*request*), menerima respon (*response*), dan melakukan pembaruan.

4.1 Pemasangan Firmware

Pada bab sebelumnya telah diurai dan dijelaskan mengenai macam-macam pustaka dan fungsi yang ada di dalam kode sumber dari *firmware* yang akan dipasang terlebih dahulu pada perangkat IoT.

```
#include <Arduino.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <ESP32httpUpdate.h>

#define VERSI 1
#define ALAMAT_IP_SERVER ""
#define JALUR_PEMBARUAN "/update"

String SSID_Wifi = "";
String sandi_Wifi = "";

void koneksiWiFi();
void periksaServer();
t_httpUpdate_return pembaruanFirmware(String url_pembaruan);

void setup()
{
  Serial.begin(9600);
  koneksiWiFi();
}

void loop()
{
  periksaServer();
  delay(1000);
}
```

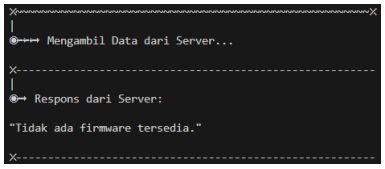
Gambar 4. 2 Konfigurasi *Firmware*

Pengguna memasang *firmware* terlebih dahulu pada perangkat IoT, tetapi sebelum itu perlu mengonfigurasinya agar bisa terhubung dengan *server* aplikasi. Caranya relatif mudah, pengguna hanya perlu menyesuaikan alamat IP dari *server* aplikasi, SSID Wi-Fi, dan kata sandi Wi-Fi.

4.2 Pembaruan Perangkat

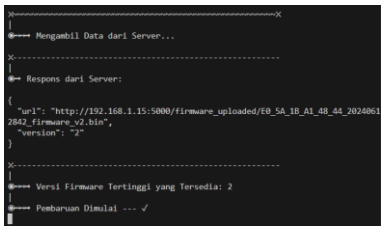
Firmware yang sebelumnya telah dipasang hanya menjalankan fungsi *over-the-air* saja. Pada tahap ini fungsi keseluruhan dari perangkat IoT belum berjalan. Sehingga *firmware* sebelumnya dimodifikasi dengan menambahkan fungsi-fungsi yang diperlukan untuk menjalankan perangkat IoT di dalamnya.

Pembaruan perangkat dimulai dengan mendaftarkan perangkat agar terhubung dengan *server*. Registrasi perangkat dilakukan dengan menambahkan MAC *address* pada halaman registrasi yang ditunjukkan pada Gambar 3.1. Perangkat yang sudah terdaftar akan menampilkan respon JSON yang menunjukkan permintaan (*request*) ke *server* aplikasi telah berhasil, respon ini dapat dipantau pada *serial monitor*.



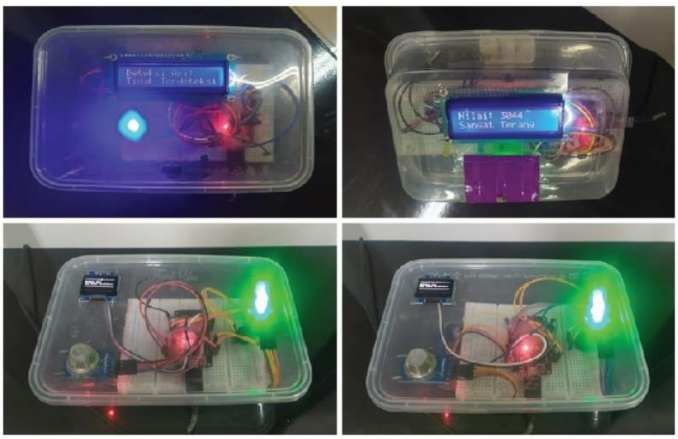
Gambar 4. 3 Respon JSON Pada Perangkat yang Terdaftar

Setelah perangkat IoT mendeteksi versi yang lebih tinggi, perangkat akan memulai pembaruan. Respon JSON yang diberikan oleh *server* menampilkan sebuah URL dan versi. URL tersebut adalah alamat pengunduhan *firmware* atau disebut *endpoint*. *Endpoint* adalah titik akhir dari sebuah URL yang menunjukkan sumber daya tertentu di dalam sebuah aplikasi atau layanan *web*. Sedangkan versi adalah versi dari *firmware* tersebut, yang sebelumnya diekstrak dari nama berkas. Jika dipantau pada *serial monitor*, seperti ini *output*-nya:



Gambar 4. 6 Perangkat Memulai Pembaruan

Selanjutnya pembaruan akan berlangsung, pembaruan yang berhasil ditandai dengan aktifnya fungsi keseluruhan (OLED dan LCD display, LED, sensor pendeteksi api, cahaya, dan gas) perangkat IoT.



Gambar 4. 7 Perangkat Deteksi Api

Keempat gambar tersebut menunjukkan bahwa pembaruan yang dilakukan pada empat perangkat berhasil.

4.3 Hasil Uji Pembaruan

Hasil uji pembaruan menghitung berapa lama durasi pembaruan dan status keberhasilan pembaruan dari semua perangkat. Durasi pembaruan diperoleh dengan menghitung selisih waktu yang dimulai saat perangkat IoT mendeteksi adanya *firmware* dengan versi yang lebih tinggi pada *server* dan mengirimkan permintaan GET ke *endpoint* berkas *firmware* untuk mengunduhnya hingga proses *self-flashing* selesai dilakukan dan perangkat IoT kembali memulai *loop request* ke *server*. Durasi tersebut dipantau menggunakan perangkat lunak Wireshark. Cara kerja perangkat IoT yang telah berubah menandakan keberhasilan pembaruan. Hasil uji pembaruan dibagi menjadi dua, yaitu pembaruan pada satu perangkat dan pembaruan pada multi perangkat (lebih dari satu sekaligus).

4.4 Pembaruan Satu Perangkat

Pembaruan satu perangkat adalah pembaruan yang dilakukan secara satu-persatu (tunggal). Berikut adalah hasil uji pembaruan pada perangkat deteksi api:

Tabel 1. 1 Hasil Uji Pembaruan Perangkat Deteksi Api

Pengujian	Status Pembaruan	Durasi Pembaruan (s)
1	Berhasil	8,498
2	Berhasil	9,128
3	Berhasil	9,610
4	Berhasil	9,161
5	Berhasil	10,125
6	Berhasil	8,987
7	Berhasil	8,749
8	Berhasil	9,538
9	Berhasil	9,523
10	Berhasil	9,760
11	Berhasil	9,239
12	Berhasil	9,585
13	Berhasil	8,965
14	Berhasil	8,792
15	Berhasil	10,710
16	Berhasil	10,270
17	Berhasil	8,490
18	Berhasil	8,948
19	Berhasil	8,920
20	Berhasil	10,967
Rata-Rata Durasi		9,398

Durasi pembaruan yang dibutuhkan perangkat ini cukup bervariasi, durasinya berkisar di rentang 8 hingga 10 detik. Jika menghitung rata-rata durasi yang dibutuhkan setelah 20 kali pengujian, rata-rata durasinya adalah 9,398 detik. Selanjutnya adalah hasil uji pembaruan dari perangkat deteksi cahaya:

Tabel 1. 2 Hasil Uji Pembaruan Perangkat Deteksi Cahaya

Pengujian	Status Pembaruan	Durasi Pembaruan (s)
1	Berhasil	12,431
2	Berhasil	11,348
3	Berhasil	11,177
4	Berhasil	10,170
5	Berhasil	10,902
6	Berhasil	11,389
7	Berhasil	11,302
8	Berhasil	10,755
9	Berhasil	10,754
10	Berhasil	10,611
11	Berhasil	10,488
12	Berhasil	10,521
13	Berhasil	10,827
14	Berhasil	10,938
15	Berhasil	10,648
16	Berhasil	11,115
17	Berhasil	11,503
18	Berhasil	11,254
19	Berhasil	11,652
20	Berhasil	10,556
Rata-Rata Durasi		11,017

Durasi pembaruan yang dibutuhkan perangkat ini berkisar di rentang 10 hingga 12 detik, sedikit lebih lama dibanding perangkat deteksi api. Jika menghitung rata-rata durasi yang dibutuhkan setelah 20 kali pengujian, rata-rata durasinya adalah 11,017 detik.

4.5 Pembaruan Multi Perangkat

Pembaruan multi perangkat adalah pembaruan yang dilakukan pada lebih dari satu perangkat sekaligus. Pada pengujian ini, perangkat yang menggunakan *firmware* yang sama adalah perangkat deteksi gas 1 dan perangkat deteksi gas 2. Berikut adalah hasil uji pembaruan dari perangkat deteksi gas 1:

Tabel 1. 3 Hasil Uji Pembaruan Perangkat Deteksi Gas 1

Pengujian	Status Pembaruan	Durasi Pembaruan (s)
1	Berhasil	27,817
2	Berhasil	11,634
3	Berhasil	11,900
4	Berhasil	11,827
5	Berhasil	12,931
6	Berhasil	10,658

7	Berhasil	12,155
8	Berhasil	11,048
9	Berhasil	10,908
10	Berhasil	11,260
11	Berhasil	11,641
12	Berhasil	12,231
13	Berhasil	11,555
14	Berhasil	11,255
15	Berhasil	10,934
16	Berhasil	11,445
17	Berhasil	14,171
18	Berhasil	23,301
19	Berhasil	12,966
20	Berhasil	10,527
Rata-Rata Durasi		13,108

Durasi pembaruan yang dibutuhkan perangkat ini berkisar di rentang 10 hingga 27 detik. Jika menghitung rata-rata durasi yang dibutuhkan setelah 20 kali pengujian, rata-rata durasinya adalah 13,108 detik. Selanjutnya adalah hasil uji pembaruan dari perangkat deteksi gas 2:

Tabel 1. 4 Hasil Uji Pembaruan Perangkat Deteksi Gas 2

Pengujian	Status Pembaruan	Durasi Pembaruan (s)
1	Berhasil	11,861
2	Berhasil	9,203
3	Berhasil	11,571
4	Berhasil	10,689
5	Berhasil	13,348
6	Berhasil	9,841
7	Berhasil	10,586
8	Berhasil	10,045
9	Berhasil	9,952
10	Berhasil	13,076
11	Berhasil	10,397
12	Berhasil	10,622
13	Berhasil	9,773
14	Berhasil	11,918
15	Berhasil	10,654
16	Berhasil	9,908
17	Berhasil	12,766
18	Berhasil	15,088

19	Berhasil	14,351
20	Berhasil	10,692
Rata-Rata Durasi		11,317

Durasi pembaruan yang dibutuhkan perangkat ini berkisar di rentang 9 hingga 15 detik. Jika menghitung rata-rata durasi yang dibutuhkan setelah 20 kali pengujian, rata-rata durasinya adalah 11,317 detik.

4.6 Pengujian Perangkat Keras

Hasil pengujian fungsi-fungsi dari perangkat keras ditampilkan pada tabel di bawah.

Tabel 1. 5 Hasil Pengujian Perangkat Keras

No	Fungsi	Status
1.	Terhubung Wi-Fi	Berhasil
2.	Mengirim permintaan (*request*) dan menerima respon (*response*)	Berhasil
3.	Mengunduh *firmware*	Berhasil
4.	*Self-flashing firmware* baru	Berhasil
5.	Sensor dapat mendeteksi api, cahaya, dan gas	Berhasil
6.	LCD dan OLED dapat menampilkan *output*	Berhasil
7.	LED menyala jika mencapai tingkat deteksi tertentu	Berhasil

Fungsi-fungsi dari perangkat keras telah diuji dan dipantau, semua fungsi berjalan dengan baik.

4.1 Pengujian Perangkat Lunak

Hasil pengujian fungsi-fungsi dari perangkat lunak ditampilkan pada tabel di bawah.

Tabel 1. 6 Hasil Pengujian Perangkat Lunak

No	Fungsi	Status
1.	Registrasi perangkat	Berhasil
2.	Menampilkan perangkat terdaftar	Berhasil
3.	Mengunggah berkas *firmware* pada perangkat	Berhasil
4.	Mengunggah berkas *firmware* pada perangkat IoT lebih dari 1 sekaligus (*checklist*)	Berhasil
5.	Menghapus perangkat	Berhasil
6.	Menampilkan riwayat pembaruan	Berhasil
7.	Mengunduh berkas *firmware* pada riwayat pembaruan	Berhasil

Fungsi-fungsi dari perangkat lunak telah diuji dan dipantau, semua fungsi menunjukkan keberhasilan.

5.1 Kesimpulan

Aplikasi berbasis *web* untuk manajemen pembaruan *firmware* dengan metode *over-the-air* adalah aplikasi yang bertujuan memudahkan proses distribusi perangkat lunak (*firmware*) secara *wireless* ke perangkat IoT. Aplikasi *web* ini dibangun menggunakan HTML, CSS, dan JavaScript sebagai *front-end* atau antarmuka penggunanya (*user interface*) dan *framework* Flask sebagai *back-end* dengan MySQL sebagai *database*-nya. Aplikasi *web* terdiri dari 3 halaman, yakni halaman registrasi, halaman perangkat terdaftar, dan halaman riwayat pembaruan. Sedangkan pada *back-end* tersusun dari rute-rute dalam skrip Flask yang menangani setiap fungsi yang bekerja pada aplikasi web. Fungsi-fungsi dasar dari aplikasi web seperti registrasi perangkat, menampilkan perangkat yang terdaftar, mengunggah *firmware*, menjalankan pembaruan, dan menampilkan riwayat pembaruan memiliki masing-masing rute yang menjalankan fungsinya. Untuk melakukan pembaruan *firmware* menggunakan aplikasi ini, pengguna perlu menghubungkan perangkat yang akan diperbarui dengan *server* aplikasi *web* melalui jaringan internet. Pengguna perlu mengonfigurasi kode *firmware* untuk mengatur SSID, kata sandi, dan alamat IP aplikasi di dalamnya. Jika pengguna ingin mengubah fungsi pada perangkat IoT, pengguna dapat memodifikasi kode *firmware* dengan tetap menyertakan fungsi *over-the-air* ke dalam *firmware* yang akan dipasang. Hal ini dilakukan agar pengguna tetap dapat melakukan pembaruan secara *wireless* pada pembaruan berikutnya.

Hasil pengujian pada penelitian ini dibagi menjadi dua, yaitu pembaruan pada satu perangkat dan multi perangkat. Pada pengujian pembaruan satu perangkat, hasilnya menunjukkan bahwa setiap perangkat melakukan pembaruan dengan relatif cepat atau tidak memerlukan waktu yang lama. Durasi rata-rata masing-masing perangkat dalam melakukan pembaruan adalah 9,398 detik pada perangkat deteksi api, 11,017 detik pada perangkat deteksi cahaya, 17,750 detik pada perangkat deteksi gas 1, 14,050 detik pada perangkat deteksi gas 2. Pada pengujian pembaruan multi perangkat, hasilnya menunjukkan perangkat deteksi gas 1 memerlukan waktu rata-rata 13,108 detik dan perangkat deteksi gas 2 memerlukan waktu rata-rata 11,317 detik. Pengujian dilakukan berulang kali pada masing-masing perangkat, setiap perangkat diuji sebanyak 20 kali. Hal ini bertujuan untuk melihat sejauh mana konsistensi keberhasilan pembaruan dan meminimalisir kegagalan yang terjadi dengan memperbaiki setiap kesalahan (*error*) yang muncul. Cepat atau lamanya durasi pembaruan juga salah satunya dapat bergantung dari koneksi jaringan. Koneksi jaringan internet yang stabil membantu proses pembaruan dapat berjalan lebih cepat. Sebaliknya, koneksi jaringan internet yang tidak stabil dapat menyebabkan proses pembaruan berjalan lebih lama atau bahkan berpotensi menyebabkan kegagalan pembaruan.

5.2 Saran

Pada bagian ini peneliti bermaksud memberi saran yang diharapkan dapat memberi manfaat bagi pihak lembaga dan peneliti selanjutnya:

1. **Bagi Pihak Lembaga**

   Menyediakan fasilitas bagi mahasiswa agar memiliki akses ke jurnal-jurnal internasional.

2. **Bagi Peneliti Selanjutnya**

   Pengembangan terhadap penelitian ini dapat dilakukan dengan menambah fitur-fitur seperti multi berkas pada pengunggahan *firmware* untuk sistem perangkat IoT yang kompleks, pembaruan yang dapat dijalankan dari jaringan yang berbeda, menerima informasi berupa versi *firmware* yang sedang berjalan pada perangkat IoT dan menampilkannya pada halaman *web*, dan mengambil informasi dalam *firmware* yang diunggah untuk proteksi perangkat IoT dari perangkat lunak jahat.

Daftar Pustaka:

[1] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, and M. Guizani, "Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges," *IEEE Wirel Commun*, vol. 23, no. 5, pp. 10–16, Oct. 2016, doi: 10.1109/MWC.2016.7721736.

[2] J. Bauwens, P. Ruckebusch, S. Giannoulis, I. Moerman, and E. De Poorter, "Over-the-Air Software Updates in the Internet of Things: An Overview of Key Principles," *IEEE Communications Magazine*, vol. 58, no. 2, pp. 35–41, Feb. 2020, doi: 10.1109/MCOM.001.1900125.

[3] X. He, S. Alqahtani, R. Gamble, and M. Papa, "Securing Over-The-Air IoT Firmware Updates using Blockchain," in *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, New York, NY, USA: ACM, May 2019, pp. 164–171. doi: 10.1145/3312614.3312649.

[4] M. J. B. de Sousa and J. F. Borin, "Design and Evaluation of a Method for Over-The-Air Firmware Updates for IoT Devices," in *Anais do XXXVI Concurso de Teses e Dissertações (CTD 2023)*, Sociedade Brasileira de Computação - SBC, Aug. 2023, pp. 80–87. doi: 10.5753/ctd.2023.230128.

[5] S. Halder, A. Ghosal, and M. Conti, "Secure over-the-air software updates in connected vehicles: A survey," *Computer Networks*, vol. 178, p. 107343, Sep. 2020, doi: 10.1016/j.comnet.2020.107343.

[6] A. Kolehmainen, "Secure Firmware Updates for IoT: A Survey," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, IEEE, Jul. 2018, pp. 112–117. doi: 10.1109/Cybermatics_2018.2018.00051.

[7] S. H. Shah and I. Yaqoob, "A survey: Internet of Things (IOT) technologies, applications and challenges," in *2016 IEEE Smart Energy Grid Engineering (SEGE)*, IEEE, Aug. 2016, pp. 381–385. doi: 10.1109/SEGE.2016.7589556.

[8] L. Da Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Trans Industr Inform*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014, doi: 10.1109/TII.2014.2300753.

[9] N. Nikolov, "Research Firmware Update Over the Air from the Cloud," in *2018 IEEE XXVII International Scientific Conference Electronics - ET*, IEEE, Sep. 2018, pp. 1–4. doi: 10.1109/ET.2018.8549628.

[10] H. Chandra, E. Anggadjaja, P. S. Wijaya, and E. Gunawan, "Internet of Things: Over-the-Air (OTA) firmware update in Lightweight mesh network protocol for smart urban development," in *2016 22nd Asia-Pacific Conference on Communications (APCC)*, IEEE, Aug. 2016, pp. 115–118. doi: 10.1109/APCC.2016.7581459.

[11] K. Arakadakis, P. Charalampidis, A. Makrogiannakis, and A. Fragkiadakis, "Firmware Over-the-air Programming Techniques for IoT Networks - A Survey," *ACM Comput Surv*, vol. 54, no. 9, pp. 1–36, Dec. 2022, doi: 10.1145/3472292.

[12] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, and E. Baccelli, "Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check," *IEEE Access*, vol. 7, pp. 71907–71920, 2019, doi: 10.1109/ACCESS.2019.2919760.

[13] C. Gao, L. Luo, Y. Zhang, B. Pearson, and X. Fu, “Microcontroller Based IoT System Firmware Security: Case Studies,” in *2019 IEEE International Conference on Industrial Internet (ICII)*, IEEE, Nov. 2019, pp. 200–209. doi: 10.1109/ICII.2019.00045.

[14] S. Nuratch, “A universal microcontroller circuit and firmware design and implementation for IoT-based realtime measurement and control applications,” in *2017 International Electrical Engineering Congress (iEECON)*, IEEE, Mar. 2017, pp. 1–4. doi: 10.1109/IEECON.2017.8075906.