Smart Comp: Jurnalnya Orang Pintar Komputer
Volume 15, Nomor 3, Juli 2026, hlm. 589-601
Terakreditasi Sinta, Peringkat 4, SK No. 295/C/C3/KPT/2026
DOI : 10.30591/smartcomp.v15i3.10308

P-ISSN : 2089-676X
E-ISSN : 2549-0796

Rancang Bangun Sistem Manajemen Perangkat IoT Berbasis *Cloud* dengan Fitur *Self-Diagnostic* dan *Over-The-Air Update*

Muhammad Yusuf Abdurrahman*[1], Alauddin Maulana Hirzan[2]
[1,2)] Teknik Informatika, Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang
Email: *[1]myusufa120@gmail.com, [2]maulanahirzan@usm.ac.id

(Naskah masuk: *4 Mei 2026*, diterima untuk diterbitkan: *2 Juli 2026*)

Abstrak: *Implementasi perangkat Internet of Things (IoT) sering menghadapi kendala pemeliharaan pada lokasi dengan akses fisik terbatas. Penelitian ini bertujuan merancang sistem manajemen perangkat IoT berbasis cloud yang mengintegrasikan fitur Self-Diagnostic dan pembaruan Over-The-Air (OTA). Sistem dikembangkan menggunakan mikrokontroler ESP32 dengan infrastruktur Cloud Hybrid yang menggabungkan Thinger.io, Firebase, dan GitHub. Metode penelitian yang digunakan adalah Prototyping yang meliputi tahap analisis kebutuhan hingga implementasi sistem kecerdasan buatan berbasis Fuzzy Logic dan Polynomial Regression. Fitur Self-Diagnostic berperan mendeteksi integritas perangkat keras melalui pemindaian jalur I2C secara otomatis, sementara fitur Tri-Method OTA (Local, Chunking, dan Cloud) menyediakan pembaruan firmware yang andal. Hasil pengujian menunjukkan tingkat keberhasilan tinggi pada berbagai kondisi jaringan. Validasi algoritma prediksi menunjukkan tingkat presisi yang lebih tinggi untuk titik koordinat spesifik dibandingkan dengan data Weather API, dengan selisih suhu rata-rata pada STATION-02 hanya sebesar 0,1°C hingga 0,3°C. Penelitian ini membuktikan bahwa integrasi manajemen jarak jauh dan analisis data cerdas mampu mentransformasi data mentah menjadi narasi informasi yang preventif. Sistem ini memberikan solusi berkelanjutan bagi infrastruktur IoT melalui otomatisasi diagnosis dan fleksibilitas pemeliharaan firmware tanpa memerlukan kehadiran fisik di lokasi.*

Kata Kunci – *Internet of Things; Device Management; Over-The-Air Update; Fuzzy Logic; ESP32*

Design and Development of Cloud-Based IoT Management with Self-Diagnostics and OTA Updates

Abstract: *The rapid expansion of Internet of Things (IoT) deployment often encounters significant challenges regarding maintenance efficiency and system reliability, particularly for devices installed in areas with limited physical access. This research develops a cloud-based IoT device management system integrated with self-diagnostic capabilities and Over-The-Air (OTA) update features to enhance operational sustainability. The system is built using ESP32 microcontrollers and a hybrid cloud infrastructure connecting Thinger.io, Firebase, and GitHub. Utilizing a Prototyping methodology, this study implements a dual-intelligent system: Polynomial Regression for atmospheric trend forecasting and Fuzzy Logic for environmental risk assessment and human-centric recommendations. The self-diagnostic feature ensures hardware integrity by automatically scanning the I2C bus, while a Tri-Method OTA approach provides robust firmware update redundancy. Experimental results indicate a successful update across various network conditions. Validation against local ground truth and global Weather API shows a high correlation, with the local AI model providing more precise predictions for specific coordinates. This research concludes that integrating remote management with embedded intelligence significantly improves the maintenance process, transforming raw data into preventive informational narratives and ensuring autonomous diagnostic capabilities for long-term monitoring.*

Keywords – *Internet of Things; Device Management; Over-The-Air Update; Fuzzy Logic; ESP32*

1. PENDAHULUAN

Implementasi teknologi *Internet of Things (IoT)* telah memberikan solusi transformatif dalam sistem akuisisi data lingkungan secara *real-time* [1][2]. Dalam pengembangannya, mikrokontroler berbasis ESP32 telah menjadi standar industri untuk stasiun cuaca mandiri karena efisiensi konsumsi daya yang optimal serta dukungan integrasi konektivitas nirkabel yang stabil melalui

protokol Wi-Fi dan Bluetooth [3][4]. Namun, kendala krusial yang sering dihadapi dalam operasional jangka panjang adalah kompleksitas pemeliharaan perangkat yang terinstalasi pada lokasi dengan akses fisik terbatas, seperti pada ketinggian tertentu atau area terpencil [5][6]. Isu teknis seperti kebutuhan pembaruan *firmware* (perangkat lunak) dan deteksi kerusakan sensor sering kali memerlukan pengecekan manual di lokasi, yang memicu tingginya biaya operasional serta risiko teknis yang signifikan di lapangan [7][8].

Beberapa studi terdahulu telah mengeksplorasi sistem pemantauan lingkungan, mulai dari optimalisasi suhu pada inkubator penetasan telur [9], deteksi dini kebakaran hutan [10] hingga otomatisasi pada sistem *smart indoor farming* [5][11]. Meskipun demikian, mayoritas penelitian tersebut masih menitikberatkan pada mekanisme transmisi data sensor ke *dashboard* [12][13] tanpa memperhatikan aspek manajemen kesehatan perangkat secara mandiri. Teknologi *Over-The-Air (OTA)*, sebuah metode pembaruan perangkat lunak secara nirkabel telah diperkenalkan sebagai solusi pembaruan jarak jauh [14]. Namun, aplikasinya dalam penelitian-penelitian sebelumnya cenderung terbatas pada satu metode tunggal yang rentan terhadap kegagalan transmisi berkas *biner* dalam skala besar atau kendala limitasi memori dinamis pada mikrokontroler [6][13].

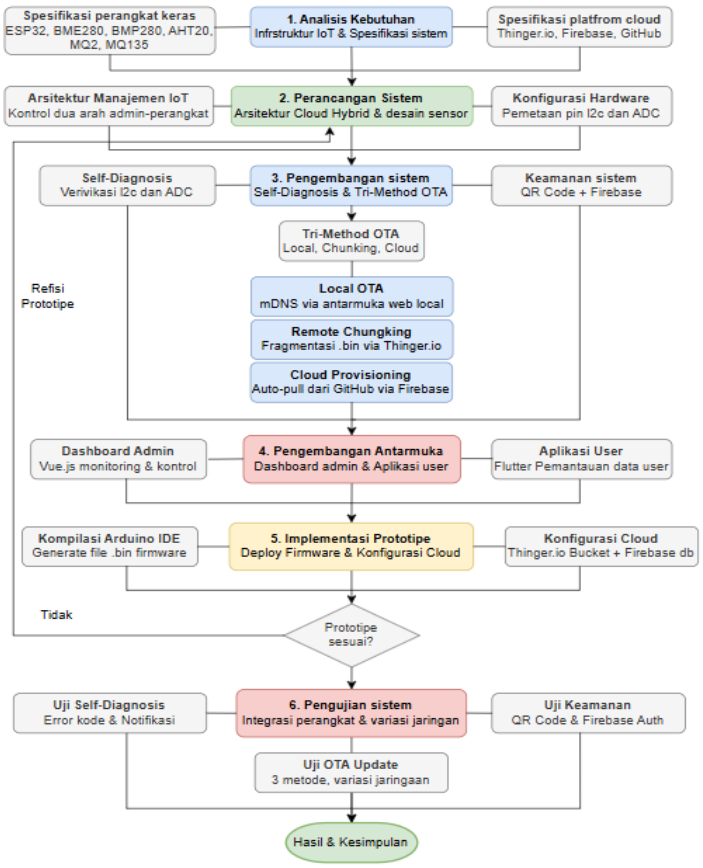
Fokus utama dari penelitian ini terletak pada pengembangan ekosistem manajemen *IoT* secara luas yang mengintegrasikan tiga metode pembaruan *OTA* sekaligus, yakni *Local OTA* melalui jaringan lokal, *Remote Chunking* untuk pengiriman data yang besar di ubah menjadi potongan-potongan kecil (*fragment* atau *chunks*) sebelum di kirim, dan *Cloud Repository* berbasis GitHub. Untuk menjamin keberhasilan sistem, penelitian ini mengintegrasikan *platform Cloud Hybrid* yang menggabungkan Thinger.io dan Firebase. Thinger.io berperan sebagai *platform cloud IoT* sumber terbuka yang menyediakan *broker* data dan manajemen perangkat yang luas, sementara Firebase digunakan untuk menangani autentikasi tingkat tinggi dan penyimpanan metadata sistem. Berbeda dengan sistem konvensional, penelitian ini mengusulkan fitur *System Diagnosis* untuk memantau *hardware integrity* (keadaan perangkat keras) secara otomatis guna mendeteksi kegagalan sensor [15][8]. Selain itu, terdapat inovasi keamanan berupa mekanisme *Device-to-User Affinity* melalui fitur *Claiming Device* pada aplikasi mobile berbasis Flutter untuk memproteksi hak akses perangkat secara eksklusif.

Pemanfaatan kecerdasan buatan berbasis *Fuzzy Logic* dan *Polynomial Regression* juga diintegrasikan untuk mentransformasi data mentah menjadi narasi prediksi cuaca yang lebih presisi dibandingkan data cuaca umum [4]. Masalah keterbatasan akses fisik untuk pemeliharaan diselesaikan melalui fitur *remote management* yang memungkinkan administrator melakukan *remote reboot*, pembersihan memori (*RAM clear*), serta Pembaruan *firmware* secara nirkabel melalui penyimpanan *cloud* [6][7]. Dengan demikian, penelitian ini bertujuan untuk merancang platform pemeliharaan yang tidak hanya berfungsi sebagai alat pemantauan data lingkungan, tetapi juga sebagai solusi manajemen infrastruktur IoT yang cerdas, aman, dan berkelanjutan.

2. METODE PENELITIAN

2.1. Jenis Metode Penelitian

Penelitian ini menerapkan metodologi pengembangan sistem Prototyping yang bersifat iteratif (berulang). Pemilihan metode ini didasarkan pada kompleksitas integrasi antara perangkat keras *embedded system* dengan berbagai layanan *Cloud Hybrid*. Berbeda dengan model Waterfall yang linear, Prototyping memungkinkan adanya siklus evaluasi pada setiap tahapan pengembangan guna memastikan fitur manajemen firmware dan akurasi sensor berfungsi optimal sebelum sistem diimplementasikan secara final. Alur kerja sistematis dimulai dari analisis kebutuhan, perancangan sistem, pengembangana sistem dengan membuat fitur *self-diagnosis* dan *tri-method OTA*, dilanjutkan dengan pengembangan antarmuka, implementasi prototipe, dan pengujian pada sistem, hingga perumusan hasil, seperti yang ditunjukkan pada Gambar 1



Gambar 1. Alur Pengembangan Sistem

2.2. *Analisis Kebutuhan Sistem*

Tahap awal dilakukan melalui analisis mendalam terhadap kebutuhan fungsional dan non-fungsional guna mengatasi kendala pemeliharaan perangkat IoT pada lokasi dengan akses fisik terbatas. Pada aspek perangkat keras, unit sensor dirancang dalam dua konfigurasi stasiun berbeda; ESP32-STATION-01 difokuskan pada pemantauan parameter cuaca menggunakan sensor BME280 dan MQ2, sedangkan ESP32-STATION-02 mengintegrasikan sensor AHT20, BMP280, serta MQ135 untuk kebutuhan akurasi kualitas udara yang lebih tinggi. Seluruh unit dilengkapi dengan indikator visual berupa LED dan peringatan audio melalui buzzer untuk fungsi peringaatan dini, dengan detail spesifikasi teknis masing-masing komponen yang dirangkum pada Tabel 1.

Tabel 1. Daftar Komponen Hardware dan Spessifikasi Fungsi

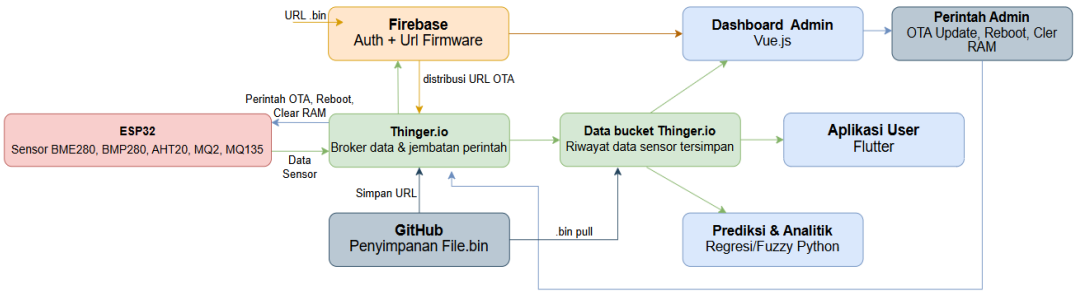
No	Nama Hardware	Spesifikasi	Fungsi
1	ESP 32 DEV KIT 4	Dual-core 32-bit MCU, Wi-Fi & BT, 4MB Flash	Pengendali utama (MCU), pemroses data sensor, dan pengelola protokol *OTA*.
2	BME 280	Protokol I2C, Range: -40 to +85°C, 0-100% Humidity	Mengukur suhu, kelembapan, dan tekanan udara secara berulang pada Unit 01.
3	BMP 280	Protokol I2C, Pressure range 300-1100 hPa	Mengukur tekanan atmosfer dan suhu dengan presisi tinggi pada Unit 02.
4	AHT 20	Protokol I2C, Akurasi Kelembapan ±2% RH	Sensor digital untuk pemantauan suhu dan kelembapan lingkungan pada Unit 02.
5	MQ2	Analog (ADC), Deteksi LPG, Asap, Alkohol	Mendeteksi keberadaan gas mudah terbakar dan asap sebagai input diagnosis lingkungan.
6	MQ135	Analog (ADC), Deteksi Benzen, Alkohol, NH3	Memantau kualitas udara dan mendeteksi gas pencemar (CO2) di sekitar perangkat.

No	Nama Hardware	Spesifikasi	Fungsi
7	Buzzer	Active Low/High 5V DC	Memberikan peringatan audio jika sistem mendeteksi kegagalan sensor atau ambang batas gas terlampaui.
8	LED	3mm/5mm Diffused (Red/Green/Blue)	(Red/Green/Blue)Indikator visual untuk status koneksi jaringan dan status aktif sistem *Self-Diagnosis*.
9	Antena 2,4 hz	Gain 2-3 dBi, Impedansi 50 Ohm	Memperkuat dan memperluas jangkauan sinyal Wi-Fi untuk memastikan stabilitas pengiriman data dan proses *OTA*.

Untuk mendukung operasional sistem, infrastruktur *cloud* memanfaatkan integrasi antara platform Thinger.io sebagai *broker* data *real-time* serta jalur transmisi *chunking biner*, dan Firebase *Realtime Database* sebagai pengelola metadata jalur pembaruan serta autentikasi pengguna. Seluruh berkas *biner firmware* yang telah dikompilasi disimpan secara terpusat pada repositori GitHub yang berperan sebagai penyedia penyimpanan *cloud*.

2.3. Perancangan Sistem Cloud

Perancangan sistem mencakup desain arsitektur jaringan dan pemetaan jalur fisik pada mikrokontroler. Arsitektur dikembangkan dengan fokus pada kemampuan kontrol dua arah, di mana administrator dapat memberikan instruksi teknis seperti *Remote Reboot* dan *Clear RAM* secara langsung melalui *dashboard*. Jalur komunikasi ini dijembatani oleh access token yang menjamin keamanan transmisi data antara Thinger.io sebagai pusat penyimpanan data sensor dan Firebase sebagai pengelola autentikasi serta *URL (Uniform Resourch Locator)* pembaruan. Detail arsitektur *cloud* ini dapat dilihat pada Gambar 2.



Gambar 2. Alur data pada *Arsitekture Cloud*

Pada tingkat perangkat keras, pemetaan pin dilakukan secara teliti guna menunjang fungsionalitas *Self-Diagnosis*. Penggunaan protokol *I2C* pada pin GPIO25 (SDA) dan GPIO26 (SCL) digunakan untuk integrasi sensor digital, sementara jalur analog (ADC) digunakan untuk akuisisi data dari sensor gas. Konfigurasi ini memungkinkan sistem melakukan pemindaian integritas jalur data secara mandiri, dengan detail pemetaan pin yang tersaji pada Tabel 2.

Tabel 2. Konfigurasi *Pinout* Sensor pada ESP32

No	Nama Sensor	Pin ESP32	Jenis Protokol/Sinyal	Keterangan
1	BME 280	GPIO25 (SDA), GPIO26 (SCL)	I2C (Digital)	Jalur komunikasi data cuaca dan lingkunagn pada STATION-01.
2	BMP280 + AHT20	GPIO25 (SDA), GPIO26 (SCL)	I2C (Digital)	Jalur komunikasi sensor pada STATION-02.
3	MQ-2	GPIO33	Analog (ADC)	Input level tegangan untuk deteksi asap/gas pada STATION-01.

No	Nama Sensor	Pin ESP32	Jenis Protokol/Sinyal	Keterangan
4	MQ-135	GPIO33	Analog (ADC)	Input level tegangan untuk kualitas udara pada STATION-02.
5	Buzzer	GPIO27	Digital Output	Sebagai sistem peringatan dini melalui suara/audio yang dihasilkan.
6	LED Indikator	GPIO13, GPIO12, GPIO14	Digital Output (PWM)	Indikator visual status sistem (R/G/B) dan status diagnosis.

2.4. Pengembangan Sistem

Fase pengembangan melibatkan penulisan kode sumber dan implementasi logika pemeliharaan mandiri. Mikrokontroler ESP32 diprogram untuk menjalankan fungsi hardware *integrity check* yang secara otomatis memindai alamat *I2C* setiap sensor saat perangkat dinyalakan. Apabila ditemukan kegagalan respons dari sensor, sistem akan membangkitkan kode error (*true/false*) ke *dashboard* admin sebagai bentuk peringatan *instan* tanpa memerlukan pengecekan fisik di lapangan. Selain itu, keamanan hak akses diperkuat melalui mekanisme *Device-to-User Affinity* menggunakan Firebase *Authentication*. Fitur *Claiming Device* mengintegrasikan identitas unik perangkat dalam format JSON pada *QR Code* (Quick Response Code), memastikan bahwa hanya akun pertama atau Master User yang memiliki otoritas eksklusif dalam manajemen perangkat.

Sebagai fitur utama, teknologi *Over-The-Air (OTA)* diimplementasikan untuk pengiriman firmware terbaru secara nirkabel melalui tiga skema berbeda guna menjamin ketersediaan pembaruan pada berbagai kondisi jaringan. Metode *Local OTA* memanfaatkan protokol *mDNS (Multicast Domain Name System)* melalui domain *alatcuaca.local* untuk pembaruan di jaringan lokal tanpa akses internet. Sementara itu, metode *Remote Chunking* menangani keterbatasan *RAM* dengan memecah berkas *.bin* menjadi potongan kecil melalui protokol *WebSocket* Thinger.io untuk dirakit kembali oleh mikrokontroler. Terakhir, metode *Cloud Repository* memungkinkan perangkat melakukan pengunduhan mandiri dari GitHub berdasarkan distribusi *URL* dari Firebase.

2.5. Perancangan Antarmuka dan Implementasi Prototipe

Pengembangan antarmuka dirancang menggunakan framework Vue.js dengan build tool Vite untuk menghasilkan *dashboard* admin yang reaktif dalam mengelola integrasi perangkat dan data historis dari Thinger.io. Di sisi pengguna, aplikasi seluler dibangun dengan framework Flutter untuk menyediakan pemantauan lintas platform yang dilengkapi fitur Scan Barcode untuk keamanan kepemilikan perangkat.

Fase implementasi teknis dilakukan dengan menuliskan seluruh logika program pada Arduino IDE menggunakan bahasa C++. Proses kompilasi mengubah kode sumber menjadi berkas *biner* tingkat rendah yang kemudian diunggah ke sistem manajemen guna divalidasi efektivitas transmisinya melalui jalur *OTA*. Secara paralel, konfigurasi lapisan middleware pada Thinger.io dan Firebase dilakukan untuk menyinkronkan jalur transmisi data dan metadata *firmware* secara *real-time*. Tahapan ditutup dengan analisis data tingkat lanjut menggunakan bahasa Python, di mana sistem secara terus-menerus mengambil 60 data historis terakhir untuk diproses melalui algoritma *Fuzzy Logic* dan *Polynomial Regression,* guna menghasilkan narasi prediksi cuaca yang presisi.

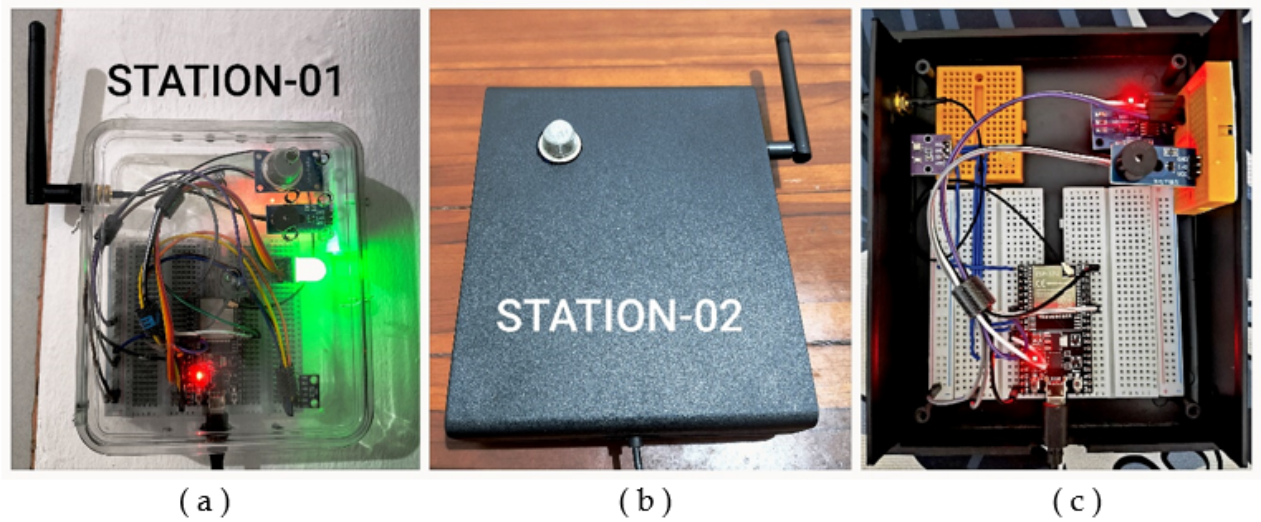
3. HASIL DAN PEMBAHASAN

3.1. Hasil Perakitan Perangkat Keras

Penyajian hasil penelitian diawali dengan demonstrasi alat fisik dari stasiun pemantauan yang telah berhasil dirakit menjadi unit fungsional. Proses perakitan ini mengintegrasikan seluruh komponen elektronik dan sensor yang telah disiapkan ke dalam dua unit prototipe, yaitu ESP32-STATION-01 dan ESP32-STATION-02, yang secara visual ditunjukkan pada Gambar 3.

Unit pertama, sebagaimana terlihat pada Gambar 3 (a), menggunakan *casing* transparan untuk mempermudah monitoring status modul secara visual, di mana sistem menggabungkan sensor BME280 dan MQ2. Sementara itu, unit kedua yang ditunjukkan pada Gambar 3 (b) menggunakan material *casing* yang lebih solid dengan sensor eksternal yang menonjol untuk akurasi pembacaan data cuaca dan lingkungan, mengintegrasikan sirkuit sensor AHT20, BMP280, serta MQ135. Detail instalasi internal perangkat, termasuk penataan *breadboard*, mikrokontroler ESP32, dan manajemen perkabelan jumper yang presisi, ditunjukan pada Gambar 3 (c). Penggunaan *breadbord* digunakan untuk memudahkan dalam proses uji coba saat penelitian.

Pemasangan antena eksternal pada setiap unit secara nyata meningkatkan stabilitas jangkauan sinyal Wi-Fi, yang menjadi faktor kunci dalam menjamin kelancaran transmisi data ke platform cloud Thinger.io. Implementasi indikator visual berupa LED RGB dan audio melalui buzzer juga telah terpasang secara presisi guna memberikan umpan balik langsung terhadap status operasional sistem serta notifikasi hasil diagnosis internal.

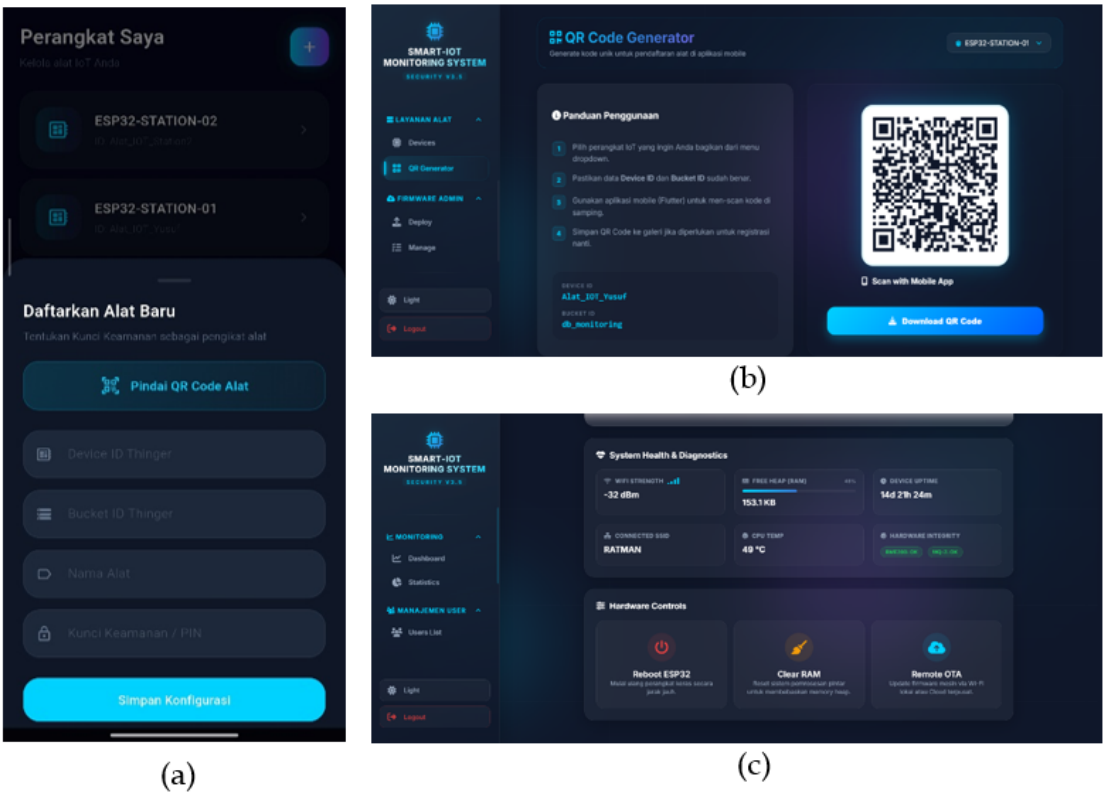


Gambar 3. Hasil Perakitan Prototipe: (a) Unit ESP32-STATION-01; (b) Unit ESP32-STATION-02; (c) Detail Instalasi Komponen Internal.

3.2. Implementasi Antarmuka Sistem dan Validasi Data hasil Prediksi

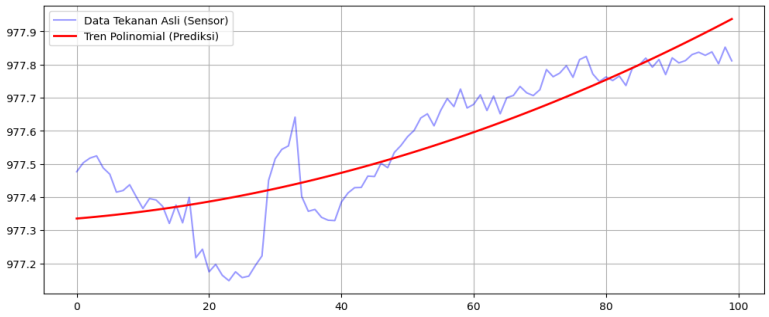
Validasi terhadap antarmuka manajemen menunjukkan bahwa *dashboard* admin berbasis Vue.js mampu menyajikan data secara reaktif dan sinkron dengan kondisi perangkat di lapangan, sebagaimana ditampilkan pada panel diagnostik dan kendali di Gambar 4 (c). Fitur *QR Generator* pada *dashboard* admin berfungsi secara optimal dalam memproduksi identitas unik perangkat yang ditunjukkan pada Gambar 4 (b), yang kemudian digunakan untuk pendaftaran melalui aplikasi mobile berbasis Flutter.

Proses pendaftaran ini dilakukan melalui fitur pindai pada aplikasi mobile seperti yang terlihat pada Gambar 4 (a). Dalam pengujian fitur keamanan *Claiming Device*, sistem secara konsisten berhasil menjalankan logika *Device-to-User Affinity* melalui Firebase *Authentication*. Protokol ini memastikan bahwa setelah sebuah perangkat terdaftar di bawah otoritas Master User, upaya klaim ulang oleh akun lain akan ditolak secara sistematis. Dengan demikian, hak eksklusif dalam manajemen perangkat, seperti akses terhadap panel kendali di Gambar 4 (c) serta otoritas penghapusan atau penggantian kata sandi, tetap terlindungi.



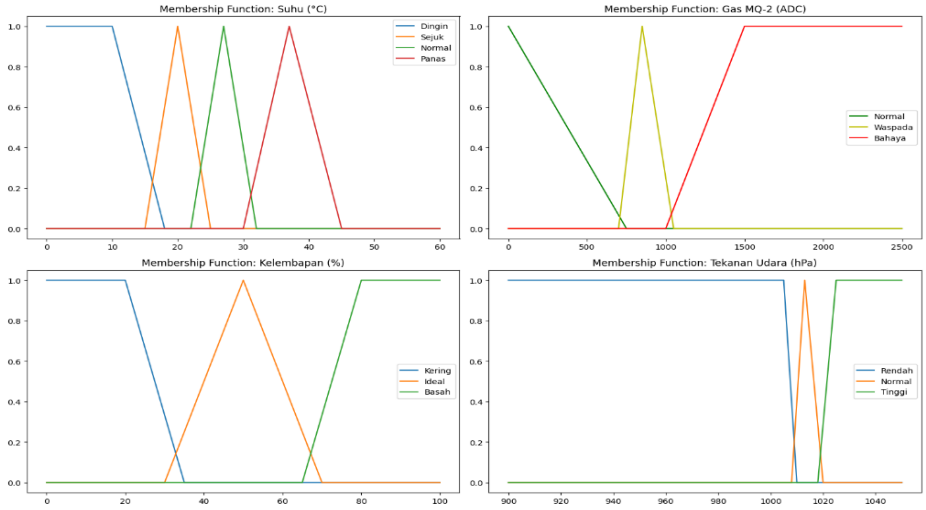
Gambar 4. Antarmuka Sistem: (a) Pendaftaran pada Mobile; (b) *QR Code Generator*; (c) Diagnostik dan panel kendali.

Sebagai bagian dari pemrosesan data tingkat lanjut yang disajikan pada antarmuka dashboard, sistem kecerdasan buatan berbasis Python secara terus-menerus mengambil 60 data terbaru setiap satu menit, memungkinkan algoritma *Polynomial Regression* untuk memetakan tren fluktuasi lingkungan, sebagaimana ditunjukkan pada kurva prediksi di Gambar 5. Pola tren ini menjadi dasar dalam memproyeksikan kondisi cuaca beberapa menit ke depan melalui analisis kemiringan kurva (slope) pada parameter tekanan udara. Meskipun data sensor (garis biru) mengalami fluktuasi akibat noise lokal, algoritma berhasil melakukan smoothing untuk menangkap tren makro secara akurat.



Gambar 5. Grafik *Polynomial Regression*

Secara paralel, data sensor lingkungan diklasifikasikan menggunakan *Fuzzy Logic* untuk menentukan level keamanan dan narasi rekomendasi berdasarkan fungsi keanggotaan (*Membership Function*) yang dirancang khusus untuk kondisi cuaca lokal, seperti terlihat pada Gambar 6. Melalui proses *defuzzification*, sistem mampu mentransformasi data mentah menjadi narasi keputusan yang informatif. Penggunaan kurva representasi Triangular dan Trapezoid memungkinkan sistem menangani parameter yang saling tumpang tindih secara presisi.



Gambar 6. Grafik Membership Function

Untuk menguji keandalan sistem, dilakukan validasi melalui komparasi antara prediksi sistem, data pihak ketiga (Weather API), dan observasi kondisi lapangan secara langsung (Ground Truth). Hasil pengujian tersebut dirangkum dalam Tabel 3.

Tabel 3. Sample Data Hasil Perbandingan Prediksi Sistem dengan Kondisi Lapangan

Tanggal 26/04/2026	**Nama Alat**	**Prediksi Sistem**	**Weather API**	**Kondisi Lapangan**	**Status Sistem Vs Lapangan**	**Selisih Suhu dengan Weather API**
16:33	*STATION-01*	Cerah (28%)	*Patchy rain nearby*	Cerah	*MATCH*	3.6°C
16:33	*STATION-02*	Cerah (37%)	*Patchy rain nearby*	Cerah	*MATCH*	0.3°C
16:35	*STATION-01*	Cerah (28%)	*Patchy rain nearby*	Cerah	*MATCH*	3.7°C
16:35	*STATION-02*	Cerah (37%)	*Patchy rain nearby*	Cerah	*MATCH*	0.1°C
16:37	*STATION-01*	Cerah (28%)	*Patchy rain nearby*	Cerah	*MATCH*	3.7°C
16:37	*STATION-02*	Cerah (37%)	*Patchy rain nearby*	Cerah	*MATCH*	0.1°C
16:39	*STATION-02*	Cerah (37%)	*Patchy rain nearby*	Cerah	*MATCH*	0.1°C
16:41	*STATION-02*	Cerah (37%)	*Patchy rain nearby*	Cerah	*MATCH*	0.1°C
16:42	*STATION-01*	Cerah (28%)	*Patchy rain nearby*	Cerah	*MATCH*	3.6°C
16:42	*STATION-02*	Cerah (37%)	*Patchy rain nearby*	Cerah	*MATCH*	0.2°C

Berdasarkan data pada Tabel 3, ditemukan temuan penting di mana Weather API melaporkan indikasi hujan di sekitar lokasi *(Patchy rain nearby*), namun sistem Prediksi lokal tetap konsisten memberikan prediksi Cerah. Validasi faktual di lapangan menunjukkan kondisi cuaca yang memang cerah, sehingga dapat disimpulkan bahwa sistem berbasis sensor fisik memiliki tingkat presisi yang lebih tinggi untuk titik koordinat spesifik dibandingkan data satelit global.

Tingkat keyakinan (*confidence level*) pada kategori Cerah yang berada pada rentang 28% hingga 37% merupakan bentuk Distribusi Probabilitas Kumulatif. Sistem Prediksi dalam penelitian ini mengklasifikasikan kondisi cuaca ke dalam lima kategori dengan total bobot kumulatif 100%. Nilai 37% menandakan bahwa kategori Cerah memiliki bobot paling dominan dibandingkan kategori lainnya. Sisa persentase didistribusikan ke kategori alternatif yang memiliki parameter serupa (seperti Berawan).

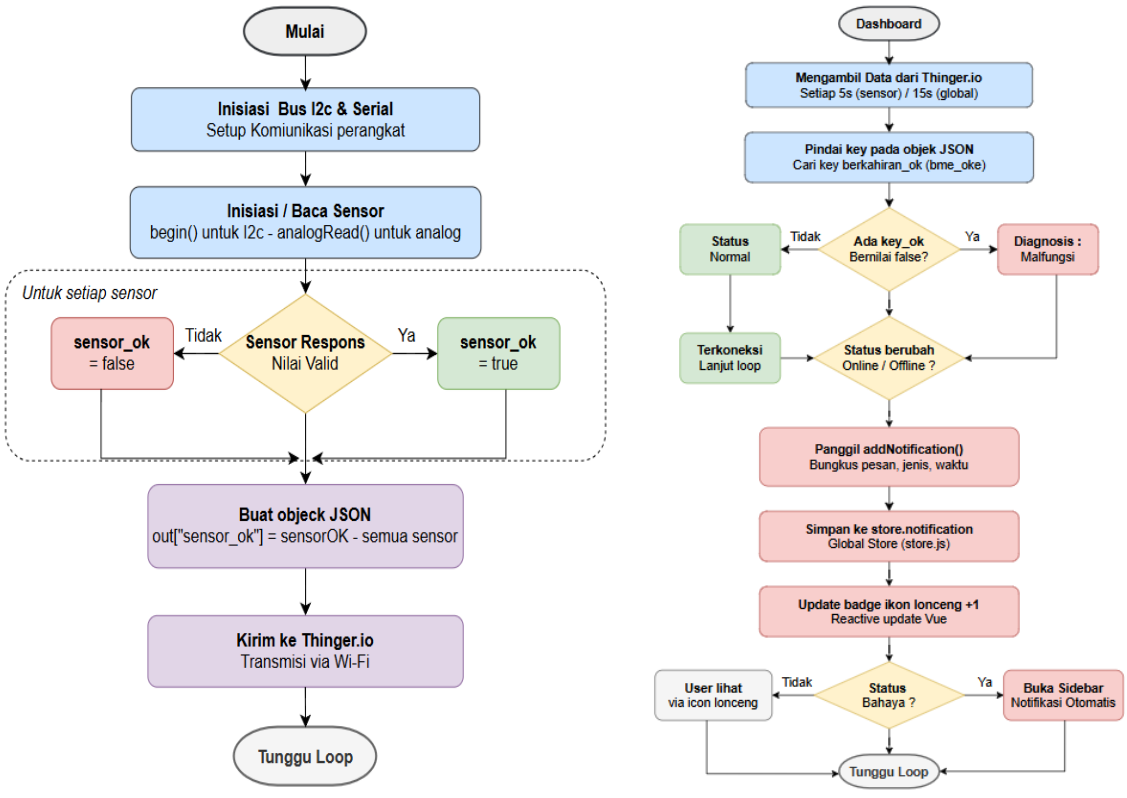
Munculnya angka yang moderat ini mencerminkan sifat alami atmosfer yang dinamis dan penuh ketidakpastian (*uncertainty*), di mana transisi antar kondisi cuaca sering kali memiliki parameter yang saling tumpang tindih (*overlapping*). Penggunaan metode *Weighted Scoring* menjamin bahwa

sistem tidak hanya sekadar menebak secara mutlak, melainkan menyajikan klasifikasi paling logis berdasarkan data sensor fisik. Hal ini menegaskan bahwa integrasi algoritma *Polynomial Regression* dan *Fuzzy Logic* terbukti mampu mengubah data mentah *IoT* menjadi instrumen pemeliharaan lingkungan yang preventif, cerdas, dan andal.

3.3. Evaluasi Mekanisme Self-Diagnosis

Fungsionalitas utama yang menjadi keunggulan teknis dalam penelitian ini adalah kemampuan perangkat untuk melakukan evaluasi kesehatan internal secara mandiri melalui fitur *Self-Diagnosis*. Mekanisme ini dirancang sebagai sistem *end-to-end* yang komprehensif, mulai dari deteksi fisik pada perangkat keras hingga penyajian informasi pada antarmuka pengguna, sebagaimana diilustrasikan pada Gambar 7.

Berdasarkan hasil pengujian, mikrokontroler ESP32 terbukti mampu menjalankan fungsi *hardware integrity check* dengan sangat responsif sesuai dengan alur logika pada Gambar 7 (a). Saat perangkat mulai beroperasi, sistem secara otomatis melakukan pemindaian alamat unik pada *bus I2C* serta memeriksa status inisialisasi sensor BME280. Selain itu, untuk sensor analog seperti MQ2, sistem menerapkan logika ambang batas pada nilai ADC untuk memastikan koneksi fisik sensor. Jika ditemukan sensor yang tidak memberikan respons (bme_OK == false) atau mengalami gagal inisialisasi, mikrokontroler segera membangkitkan flag kesalahan unik (misalnya: bme_ok: false) yang diteruskan ke *platform cloud*. Di sisi lain, *dashboard* administrator secara aktif melakukan pengambilan data setiap 5 detik untuk memeriksa status flag kesehatan tersebut, sesuai dengan alur pada Gambar 7 (b). Jika ditemukan flag berakhiran _ok bernilai false, sistem dashboard Vue.js mendiagnosa adanya malfungsi sensor dan secara otomatis memicu fungsi addNotification.



(a) (b)

Gambar 7. Flowchart Diagram: (a) Alur Logika pada Perangkat (ESP32); (b) Alur Diagnosis di Dashboard Web.

Bukti keberhasilan pengiriman peringatan ini dapat dilihat pada panel histori notifikasi di Gambar 8, yang menampilkan rincian perangkat serta waktu terdeteksinya

status offline secara presisi. Mekanisme ini memastikan integritas data tinggi dan memungkinkan penanganan cepat melalui perintah *Remote Reboot* atau *Clear RAM* dari jauh.

Gambar 8. Histori Notifikasi Sistem

3.4. Analisis Pengujian Kinerja Tri-Method OTA

Aspek krusial terakhir yang dievaluasi dalam penelitian ini adalah efektivitas pengiriman firmware melalui tiga metode Over-The-Air (OTA) yang terintegrasi. Sistem ini dirancang untuk menjaga keberlanjutan pembaruan perangkat dalam berbagai kondisi operasional. Guna menguji keandalan awal (initial reliability) dari arsitektur tersebut, dilakukan eksperimen pengujian secara sekuensial sebanyak 20 kali percobaan untuk masing-masing metode (n = 20 per metode, dengan total 60 kali pengujian akumulatif) pada kondisi jaringan lokal dan internet standar yang stabil. Pengujian dilakukan menggunakan berkas biner (.bin) terkompilasi dengan ukuran tetap sebesar 1.19 MB. Data kuantitatif mengenai perbandingan performa ketiga metode tersebut dirangkum secara mendalam pada Tabel 4.

Tabel 4. Hasil Pengujian Kecepatan dan Keberhasilan Update OTA

Metode OTA	Ukuran File(.bin)	Rata-rata Waktu (Detik)	Jumlah Percobaan (n)	Status Keberhasilan
Local OTA	1.19 MB	20 – 25	20	Berhasil
Remote Chunking	1.19 MB	450 – 500	20	Berhasil
Cloud Repository	1.19 MB	45 – 50	20	Berhasil

Dalam rangkaian eksperimen batasan 20 kali percobaan ini menunjukkan efisiensi pemanfaatan arsitektur perangkat lunak pada kondisi pengujian yang terkontrol. Pada lapisan transportasi, reliabilitas pengiriman paket didukung oleh penggunaan protokol komunikasi Stateful WebSocket dan HTTPS/TLS yang berjalan di atas TCP. Secara inheren, mekanisme Error Detection melalui Cyclic Redundancy Check (CRC) dan algoritma Automatic Repeat Request (ARQ) pada TCP bertugas melakukan retransmisi otomatis jika terjadi gangguan transmisi minor (packet loss). Karakteristik kontrol aliran data ini sejalan dengan tinjauan menyeluruh oleh Arakadakis dkk[16]. mengenai berbagai teknik Firmware Over-the-air Programming (OTAP), yang menegaskan bahwa pemilihan protokol transport yang tepat sangat krusial untuk mengatasi kendala konektivitas dan jangkauan pada jaringan Internet of Things (IoT).

Perlindungan tersebut diperkuat oleh verifikasi lokal tingkat perangkat keras menggunakan fungsi hash MD5 dari pustaka Update.h pada skema partisi Virtual OTA (OTA0/OTA1) ESP32. Pendekatan manajemen partisi ini mengacu pada prinsip dasar sistem manajemen firmware berbasis ESP32 seperti yang dievaluasi pada platform IOTAfy oleh Panagou dkk. [17], di mana validasi kriptografi atau checksum tingkat lokal diterapkan untuk memastikan integritas berkas biner dan mencegah eksekusi data yang korup akibat fenomena bit-flip selama proses transmisi

berjalan. Jika ditemukan ketidakcocokan nilai hash, sistem secara otomatis membatalkan proses penulisan dan melakukan pemulihan (rollback) ke versi firmware lama yang stabil.

Faktor penentu berikutnya yang menjaga stabilitas pengujian pada metode Remote Chunking adalah penerapan strategi manajemen beban buffer memori dengan memecah berkas biner utuh menjadi potongan-potongan kecil berukuran masing-masing 1 KB melalui jalur WebSocket Thinger.io. Prinsip pembagian beban biner secara bertahap ini memiliki kesamaan konsep dengan pengiriman komponen biner sekuensial yang diterapkan oleh Sudharsan dkk. [18] pada riset OTA-TinyML, yang membuktikan bahwa pengiriman berbasis blok terstruktur efektif mengeliminasi risiko lonjakan memori (Out of Memory) pada perangkat dengan limitasi RAM dinamis. Kemampuan komputasi ESP32 dalam mengeksekusi rangkaian tugas hibrida ini secara berulang tanpa mengalami malafungsi juga didukung oleh hasil analisis fungsionalitas Industrial IoT Gateway berbasis ESP32 oleh Devya dan Das [19], yang mengonfirmasi durabilitas dan ketahanan arsitektur mikrokontroler ini dalam mengolah lalu lintas data yang padat pada lingkungan pemantauan aktif.

Meskipun pengujian statis sebanyak 20 kali percobaan ini mencatatkan tingkat keberhasilan yang tinggi, perlu digarisbawahi bahwa metrik tersebut merepresentasikan performa perangkat pada kondisi lingkungan pengujian dengan stabilitas sinyal yang terjaga dan stabil. Di bawah implementasi skala nyata di area terpencil, keberhasilan proses OTA akan tetap dipengaruhi oleh fluktuasi ekstrem bandwidth operator seluler dan potensi pemutusan koneksi secara permanen di tengah jalan. Namun demikian, integrasi kendali alur TCP, validasi MD5 tingkat hardware, serta pembatasan ukuran chunk memori dalam penelitian ini telah memberikan landasan arsitektur yang kuat dan tervalidasi oleh berbagai riset terdahulu untuk meminimalisir risiko kegagalan firmware deployment. Karakteristik mendalam dari tiap metode disajikan pada Tabel 5.

Table 5. Komparasi Keunggulan, Kekurangan, dan Fungsi *Tri-Method OTA*

Metode OTA	Fungsi Utama	Keunggulan Utama	Kekurangan/Limitasi
Local OTA	Untuk melakukan pemeliharaan di Lokasi oleh teknisi lapangan tanpa internet	-Waktu eksekusi paling cepat (20-25 detik). -Tidak membutuhkan kuota internet. -kebal terhadap gangguan cloud server	Jarak jangkauan terbatas pada radius sinyal Wi-Fi local (Local Area Network).
Remote Chunking	Update jarak jauh (remote) berskala massal saat perangkat mengalami limitasi memori (RAM)	-Sangat stabil karena memori ESP32 tidak terbebani berkas penuh. -Proses perakitan firmware terstrukture lewat WebSocket.	Waktu eksekusi sangat lambat (450-500 detik) karena adanya overhead pemisahan dan penggabungan paket data.
Cloud Repository	Pembaruan otomatis berbasis siklus update (Automated Deployment)	-Waktu eksekusi relative cepat (45-50 detik) untuk skala remote. -Menggunakan URL dinamis Firebase yang fleksibel.	Sangat bergantung pada stabilitas internet luar dan ketersedian server pihak ketiga (Github & Firebase)

Secara keseluruhan, hasil analisis performa ini menegaskan bahwa setiap metode dalam Tri-Method OTA memiliki peran strategis yang saling melengkapi dalam siklus pemeliharaan operasional. Metode Local OTA Update menjadi solusi paling efisien dengan waktu eksekusi tersingkat karena memanfaatkan fungsionalitas mDNS pada jaringan lokal tanpa terpengaruh oleh latensi routing internet luar. Dalam implementasi praktisnya, metode lokal ini dirancang khusus untuk mengakomodasi kebutuhan teknisi lapangan pada saat melakukan instalasi awal, kalibrasi rutin, ataupun penanganan darurat di lokasi pemasangan alat ketika koneksi internet publik mengalami gangguan total.

Di sisi lain, aspek fleksibilitas dan kemudahan manajemen sistem bertumpu pada metode Cloud Repository Update serta Remote Chunking. Metode Cloud Repository dikembangkan untuk mempermudah operasional dari sisi client atau administrator pusat, di mana pembaruan skrip firmware berskala massal dapat dideploy secara simultan langsung dari repositori GitHub hanya

melalui pembaruan instruksi parameter URL dinamis pada Firebase. Pendekatan jarak jauh ini memangkas kebutuhan mobilisasi kru teknis ke area terpencil.

Integrasi ketiga jalur ini secara terstruktur membentuk sebuah ekosistem pemeliharaan dengan toleransi kesalahan yang tinggi (fault-tolerant system). Melalui arsitektur ini, jika salah satu jalur distribusi mengalami kendala jaringan atau kegagalan total (single point of failure), administrator masih memiliki opsi jalur alternatif lain sebagai failover untuk memastikan perangkat stasiun cuaca tetap dapat diperbarui secara aman, adaptif, dan berkelanjutan tanpa memerlukan kehadiran fisik sama sekali di lapangan.

4. KESIMPULAN

Penelitian ini berhasil membuktikan bahwa integrasi arsitektur *Cloud Hybrid* menggunakan Thinger.io, Firebase, dan GitHub mampu menciptakan ekosistem manajemen IoT yang tangguh dan mandiri. Mekanisme *Self-Diagnosis* yang dikembangkan menunjukkan efektivitas tinggi dalam mendeteksi kegagalan perangkat keras melalui pemindaian *bus I2C* secara otomatis, sehingga memungkinkan administrator mengidentifikasi kerusakan sensor tanpa perlu melakukan inspeksi fisik di lokasi. Selain itu, penerapan strategi *Tri-Method OTA Update* memberikan alternatif pembaruan sistem yang sangat andal, dengan tingkat keberhasilan yang tinggi pada berbagai kondisi jaringan. Penggabungan metode lokal, *chunking*, dan *cloud repository* memastikan integritas *firmware* tetap terjaga serta mengatasi keterbatasan memori pada perangkat keras.

Kelebihan utama dari sistem ini terletak pada kemampuan manajemen jarak jauh yang menyeluruh, mulai dari keamanan hak akses melalui fitur *Claiming Device* hingga analisis data cerdas yang mampu mentransformasi data cuaca mentah menjadi narasi prediksi yang informatif bagi pengguna. Namun, terdapat keterbatasan pada aspek akurasi prediksi cuaca jangka panjang karena model *Polynomial Regression* dan *Fuzzy Logic* yang digunakan saat ini masih sangat bergantung pada variasi data historis lokal yang terbatas. Hal ini membuka peluang bagi adanya sedikit potensi galat (*error*) atau misprediksi jika terjadi perubahan cuaca ekstrem yang tiba-tiba.

Sebagai langkah pengembangan selanjutnya, integrasi algoritma kecerdasan buatan yang lebih kompleks, seperti pemanfaatan *Machine Learning* dengan *model Long Short-Term Memory (LSTM)* atau *Deep Learning*, sangat direkomendasikan untuk meningkatkan presisi prediksi. Peningkatan pada sisi pengolahan *Fuzzy Logic* dan optimalisasi *regresi polinomial* perlu dilakukan guna meminimalisir tingkat *error* sehingga dihasilkan prediksi yang lebih tinggi tingkat kepercayaannya. Pengembangan di masa depan juga dapat diarahkan pada penambahan parameter sensor yang lebih beragam serta perluasan integrasi sistem keamanan berbasis blockchain untuk menjamin integritas data yang lebih absolut.

DAFTAR PUSTAKA

[1] H. J. El-Khozondar *et al.*, "A smart energy monitoring system using ESP32 microcontroller," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 9, Sep. 2024, doi: 10.1016/j.prime.2024.100666.

[2] Md. H. Rahman, M. Shihab, Md. A. Rahman, and M. Naderuzzaman, "ESP32 Microcontroller: A Review of Architecture, Communication Protocols, Applications and Research Challenges," *Open Access Journal on Engineering Applications*, vol. 2, no. 1, p. 19, Feb. 2026, doi: 10.64886/oajea.0102.003.

[3] E. Gupta, S. Bansal, V. Choudhary, V. Gupta, and R. K. Pachauri, "ESP32-Based Dual-Connectivity Data Logger for Continuous Environmental Monitoring for AI Applications," Oct. 29, 2025. doi: 10.21203/rs.3.rs-7656817/v1.

[4] M. A. R. Kiran, M. Santhoh Kumar, M. Pavan Balaji, T. Raja, and V. Sivaram, "Smart IoT-Based Weather Monitoring Ecosystem with Cloud Integration and Telegram Alert System," 2026. [Online]. Available: https://ijerst.org/index.php/ijerst

[5] T. Aditya and O. A. Dhewa, "Design and Implementation of Update Script in the IoT-Based Smart Indoor Farming System Module at PT Inastek Using Over-the-Air Programming," *Media of Computer Science*, vol. 1, no. 2, pp. 129–138, Dec. 2024, doi: 10.69616/mcs.v1i2.201.

[6] L. Formanek, M. Kubascik, O. Karpis, and P. Kolok, "Advanced System for Remote Updates on ESP32-Based Devices Using Over-the-Air Update Technology," *Computers*, vol. 14, no. 12, Dec. 2025, doi: 10.3390/computers14120531.
[7] I. C. Panagou, S. Katsoulis, E. Nannos, F. Zantalis, and G. Koulouras, "IOTAfy: An ESP32-Based OTA Firmware Management Platform for Scalable IoT Deployments," in *The 6th International Electronic Conference on Applied Sciences*, Basel Switzerland: MDPI, Feb. 2026, p. 40. doi: 10.3390/engproc2026124040.
[8] Noprianto, N. Funabiki, H. H. S. Kyaw, K. C. Brata, and I. N. D. Kotama, "A Proposal of Secure and Automated Over-the-Air Firmware Update Mechanism for IoT Devices Using Continuous Integration and Continuous Delivery," *Sensors*, vol. 26, no. 5, Mar. 2026, doi: 10.3390/s26051535.
[9] A. Yoga Ashidiqi and A. Ichsan Pradana, "Sistem Monitoring Suhu Penetasan Telur Ayam Berbasis Internet Of Things (IoT) Internet Of Things (IoT) Based Chicken Egg Hatching Temperature Monitoring System," *Jurnalnya Orang Pintar Komputer*, vol. 14, no. 2, 2025, doi: 10.30591/smartcomp.v13i1.7557.
[10] B. Kusumo and T. Ardiansyah, "RANCANG BANGUN SISTEM PENDETEKSI KEBAKARAN BERBASIS MIKROKONTROLER ESP32," 2024.
[11] S. Kurnia, A. Khaidar, and P. Studi Magister, "Rancang Bangun Sistem Pemantauan Suhu dan Kelembapan Berbasis Internet of Things," vol. 17, no. 2, 2025.
[12] W. He, M. J. A. Baig, and M. T. Iqbal, "An Internet of Things—Supervisory Control and Data Acquisition (IoT-SCADA) Architecture for Photovoltaic System Monitoring, Control, and Inspection in Real Time," *Electronics (Switzerland)*, vol. 14, no. 1, Jan. 2025, doi: 10.3390/electronics14010042.
[13] C. Silva, V. A. Cunha, J. P. Barraca, and R. L. Aguiar, "Hands-on evaluation of the cryptographic overhead on wireless sensor networks," 2020.
[14] S. Palm and R. Bui, "Bachelor Degree Project Over-The-Air update techniques and how to evaluate them-A comparison of Over-The-Air updates for type ESP-32," 2022.
[15] D. Arregui Almeida, J. Chafla Altamirano, M. Román Cañizares, P. P. Játiva, J. Guaña-Moya, and I. Sánchez, "Gateway-Free LoRa Mesh on ESP32: Design, Self-Healing Mechanisms, and Empirical Performance," *Sensors*, vol. 25, no. 19, Oct. 2025, doi: 10.3390/s25196036.
[16] 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, Dec. 2022, doi: 10.1145/3472292.
[17] I. C. Panagou, S. Katsoulis, E. Nannos, F. Zantalis, and G. Koulouras, "IOTAfy: An ESP32-Based OTA Firmware Management Platform for Scalable IoT Deployments," in *The 6th International Electronic Conference on Applied Sciences*, Basel Switzerland: MDPI, Feb. 2026, p. 40. doi: 10.3390/engproc2026124040.
[18] B. Sudharsan *et al.*, "OTA-TinyML: Over the Air Deployment of TinyML Models and Execution on IoT Devices," *IEEE Internet Comput.*, vol. 26, no. 3, pp. 69–78, 2022, doi: 10.1109/MIC.2021.3133552.
[19] Devya and Shib Sankar Das, "Design and Implementation of a Low-Cost, High-Density IIoT Gateway for Smart Factory Monitoring Using ESP32," *International Journal of Innovations in Science, Engineering And Management*, pp. 364–375, Jun. 2026, doi: 10.69968/ijisem.2026v5i2364-375.