



Design and Implementation of an ESP32-Based Multiprotocol Temperature Monitoring System and Asynchronous Collector Server

Rancang Bangun Sistem Monitoring Suhu Multiprotokol Berbasis ESP32 dan Server Kolektor Asinkron

Muhammad Fairus Fawas Afanza^{1*}, Moh. Noor Al Azam²

^{1,2}Program Studi Sistem Komputer, Fakultas Ilmu Komputer, Universitas Narotama, Surabaya, Indonesia
E-mail: muhammadfairusfa122@fasilkom.narotama.ac.id^{1*}, noor.azam@narotama.ac.id²

Abstract

This study aims to design and implement a robust multi-protocol temperature monitoring system combining modular ESP-IDF firmware and an asynchronous collector server. The system integrates an ESP32 microcontroller interfaced with a Bosch BMP280 temperature sensor and Texas Instruments INA226 digital power monitor over a shared 400 kHz I2C bus, supporting dynamic MQTT (QoS 0 and QoS 1) and RESTful HTTP POST transmission. The receiver architecture combines an asynchronous FastAPI server, Mosquitto MQTT subscriber, and SQLite database in Write-Ahead Logging mode with Pydantic validation. Functional testing confirmed successful hardware initialization, drift-free FreeRTOS task scheduling via xTaskDelayUntil, SNTP epoch synchronization, and structured JSON telemetry parsing. The implemented system demonstrated 100% data reception across 810 experimental test records, validating the modular firmware and dual-transport architecture for smart IoT telemetry applications.

Article Information

Article History:

Received: 19/08/2026
First Revised: 21/09/2026
Accepted: 30/09/2026
Publication Date: 30/09/2026

Keywords:

*Asynchronous server;
embedded firmware; ESP32
microcontroller; FreeRTOS;
multi-protocol telemetry*



Copyright (c) 2026 Muhammad Fairus Fawas Afanza,
Moh. Noor Al Azam

Abstrak

Penelitian ini bertujuan merancang dan mengimplementasikan sistem monitoring suhu multiprotokol yang andal mengintegrasikan firmware modular ESP-IDF dan server kolektor asinkron. Sistem mengintegrasikan mikrokontroler ESP32 dengan sensor suhu Bosch BMP280 dan sensor pemantau daya digital Texas Instruments INA226 melalui bus I2C bersama 400 kHz, mendukung mode transmisi MQTT (QoS 0 dan QoS 1) serta RESTful HTTP POST. Sisi penerima memadukan server asinkron FastAPI, subscriber Mosquitto MQTT, dan basis data SQLite mode Write-Ahead Logging dengan validasi skema Pydantic. Pengujian fungsional membuktikan keberhasilan inisialisasi perangkat keras, penjadwalan task FreeRTOS bebas drift via xTaskDelayUntil, sinkronisasi epoch SNTP, dan parsing payload JSON terstruktur. Sistem berhasil merekam 810 data pengujian dengan keandalan penerimaan 100%, membuktikan keandalan firmware modular dan arsitektur dual-transport untuk aplikasi IoT.



Copyright (c) 2026 Muhammad Fairus Fawas Afanza,
Moh. Noor Al Azam

Artikel Info**Riwayat Artikel:**

Diterima: 19/08/2026

Direvisi: 21/09/2026

Disetujui: 30/09/2026

Dipublikasikan: 30/09/2026

Kata Kunci:

Firmware embedded;
FreeRTOS; mikrokontroler
ESP32; server asinkron;
telemetri multiprotokol

1. Pendahuluan

Kebutuhan terhadap pemantauan parameter lingkungan secara presisi, berkelanjutan, dan hemat energi telah mendorong pemanfaatan arsitektur Internet of Things (IoT) pada berbagai domain krusial, seperti pemantauan suhu server, rantai dingin farmasi, dan otomasi industry (Dizdarević et al., 2019; Salman & Jain, 2017). Dalam pengembangan node telemetri berbasis embedded, keandalan sistem sangat dipengaruhi oleh desain arsitektur firmware dan efektivitas komunikasi data menuju server penampung (Atmoko et al., 2017; Grgić et al., 2016). Sebagian besar sistem pemantauan lingkungan konvensional umumnya dirancang secara monolitik dan hanya mendukung satu protokol lapisan aplikasi secara kaku, baik itu HTTP RESTful maupun MQTT (Naik, 2017).

Keterbatasan rancangan monolitik ini menimbulkan tantangan interoperabilitas yang signifikan ketika sistem harus diterapkan pada infrastruktur jaringan yang dinamis atau ketika kebutuhan transmisi beralih antara orientasi throughput tinggi (stateless request-response) dan efisiensi konsumsi daya (stateful publish-subscribe) (Chen & Kunz, 2016). Selain itu, pada level perangkat keras node sensor, sebagian besar pengembang IoT masih mengandalkan framework perangkat lunak tingkat tinggi seperti Arduino IDE yang memiliki lapisan abstraksi tebal, alokasi memori dinamis yang tidak efisien, dan ketiadaan kendali deterministik terhadap penjadwalan task waktu nyata (Mishra et al., 2021).

Tinjauan terhadap rekayasa sistem telemetri terkini mengidentifikasi adanya kesenjangan implementasi yang mendasar. Masih sangat jarang ditemukan rancang bangun sistem IoT terbuka yang mengintegrasikan firmware modular berbasis native C/FreeRTOS dengan kemampuan peralihan multiprotokol (HTTP POST, MQTT QoS 0, dan MQTT QoS 1) secara terpadu, sekaligus menyematkan instrumentasi sensor kelistrikan digital (INA226) secara simultan bersama sensor lingkungan (BMP280) pada satu bus I2C bersama (Jara Ochoa et al., 2023). Ketiadaan sinkronisasi waktu clock terpadu antara mikrokontroler dan server juga kerap menyulitkan pelacakan latensi transmisi end-to-end secara akurat (Dizdarević et al., 2019). Di sisi backend, banyak server kolektor masih menggunakan model sinkronus

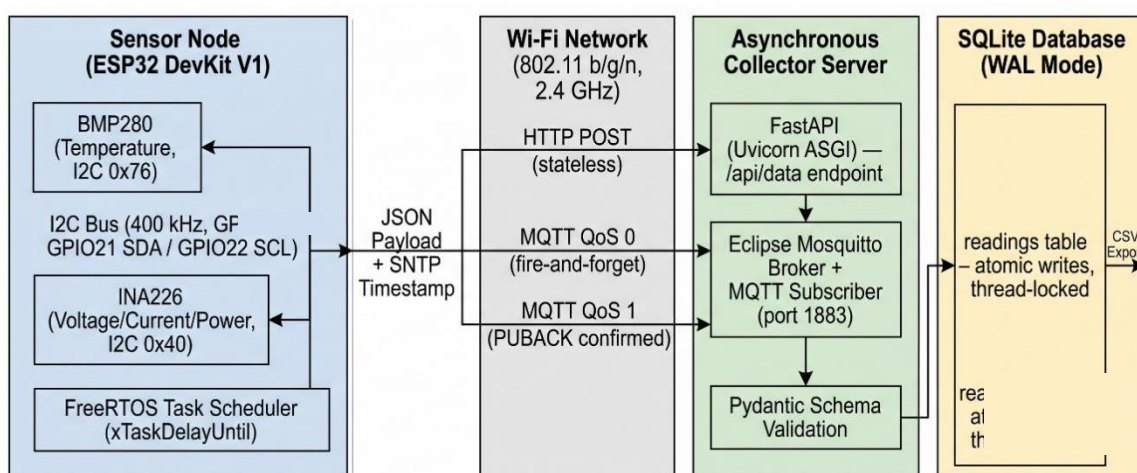
pemblokiran (blocking I/O) yang rentan mengalami penurunan performa saat menerima lonjakan payload konkuren dari node sensor (Al-Masri et al., 2020; Mishra & Kertesz, 2020).

Untuk menjawab kesenjangan rekayasa tersebut, penelitian ini bertujuan merancang dan mengimplementasikan sistem monitoring suhu multiprotokol yang tangguh, modular, dan terukur berbasis mikrokontroler ESP32 DevKit V1 dan server kolektor asinkron terintegrasi. Kebaruan rancang bangun ini mencakup: (1) pengembangan arsitektur firmware berbasis Espressif IoT Development Framework (ESP-IDF) murni dengan manajemen task FreeRTOS `xTaskDelayUntil()` bebas pergeseran waktu (timing drift); (2) implementasi driver I2C bersama (shared bus) berkecepatan 400 kHz yang membaca sensor suhu Bosch BMP280 dan sensor tegangan/arus Texas Instruments INA226 secara atomik; (3) mekanisme enkapsulasi payload JSON terstandarisasi dengan sinkronisasi epoch SNTP lokal; dan (4) pembangunan server kolektor multi-jalur (dual-transport) berbasis FastAPI dan Mosquitto subscriber dengan validasi skema Pydantic serta penyimpanan SQLite mode Write-Ahead Logging (WAL) bebas hambatan konkurensi. Rancang bangun ini diharapkan menjadi referensi arsitektur terbuka yang komprehensif bagi rekayasa sistem IoT profesional.

2. Metodologi

Metodologi yang digunakan dalam penelitian ini adalah pendekatan rekayasa perangkat lunak dan keras (engineering prototyping) yang terbagi ke dalam empat tahapan utama: perancangan arsitektur sistem, pengembangan firmware node sensor ESP32, pembangunan server kolektor data, dan pengujian verifikasi fungsionalitas sistem secara menyeluruh.

Arsitektur Sistem Keseluruhan. Alur aliran data sistem telemetri ini dirancang secara terpadu dari lapisan persepsi fisik hingga lapisan penyimpanan persisten. Pada node sensor, mikrokontroler ESP32 DevKit V1 secara periodik mengakuisisi data temperatur fisik dari sensor Bosch BMP280 dan besaran listrik (tegangan bus, arus, dan daya sesaat) dari sensor TI INA226 melalui bus data I2C bersama (GPIO 21 SDA dan GPIO 22 SCL). Data yang diperoleh kemudian diserialisasi menjadi payload JSON terstruktur, dibubuhi timestamp epoch SNTP, dan ditransmisikan melalui antarmuka radio Wi-Fi 2,4 GHz menuju jaringan LAN lokal. Di sisi backend penerima, server kolektor menyediakan dua pintu gerbang: RESTful HTTP endpoint (`/api/data` pada port 8000) dan MQTT broker (Eclipse Mosquitto pada port 1883). Data yang masuk divalidasi secara asinkron menggunakan skema data Pydantic, kemudian disimpan secara atomik ke dalam basis data SQLite berkinerja tinggi dalam mode Write-Ahead Logging (WAL) serta dapat diekspor secara langsung ke format CSV.



Gambar 1. Diagram Blok Arsitektur Sistem Monitoring Suhu Multiprotokol

Desain Perangkat Keras dan Manajemen Bus I2C. Pemilihan ESP32 DevKit V1 didasarkan pada ketersediaan prosesor dual-core 32-bit Xtensa LX6 berkecepatan 240 MHz, memori SRAM 520 KB, dan transceiver Wi-Fi 802.11 b/g/n internal yang sangat memadai untuk

menangani tumpukan protokol jaringan TCP/IP lwIP tanpa mengorbankan kecepatan pembacaan sensor. Sensor BMP280 dipilih karena memiliki akurasi suhu tinggi ($\pm 1^\circ\text{C}$) dengan resolusi pembacaan $0,01^\circ\text{C}$, sementara modul INA226 dipilih karena mampu mengukur arus dan tegangan DC secara simultan melalui ADC 16-bit internal dengan shunt resistor $0,1\ \Omega$ (1%). Kedua sensor dihubungkan secara paralel pada bus I2C master dengan frekuensi detak 400 kHz (Fast Mode). Alamat I2C dikonfigurasi secara unik: 0x76 untuk BMP280 (pin SDO ke GND) dan 0x40 untuk INA226 (pin Ao dan A1 ke GND), sehingga mencegah terjadinya tabrakan alamat (address conflict). Tabel 1 merinci spesifikasi komponen perangkat keras dan perangkat lunak yang diimplementasikan.

Arsitektur Firmware ESP-IDF dan Penjadwalan FreeRTOS. Firmware ESP32 dikembangkan menggunakan bahasa C murni di atas ESP-IDF v5.x dan sistem operasi real-time FreeRTOS. Modul firmware distrukturisasi secara modular ke dalam beberapa berkas driver terpisah: `wifi_conn.c` untuk manajemen koneksi Wi-Fi dan sinkronisasi waktu SNTP, `bmp280.c` untuk inisialisasi dan pembacaan register kalibrasi Bosch, `ina226.c` untuk kalibrasi register shunt dan perhitungan daya, serta `main.c` sebagai pengelola alur telemetri utama.

Penjadwalan siklus telemetri dijalankan di dalam tugas FreeRTOS menggunakan fungsi `xTaskDelayUntil(&next_wake, pdMS_TO_TICKS(interval_ms))` alih-alih `vTaskDelay` biasa. Penggunaan `xTaskDelayUntil` menjamin bahwa durasi eksekusi pembacaan sensor, serialisasi JSON, dan transmisi jaringan diperhitungkan secara otomatis, sehingga interval pengiriman data tetap konstan secara presisi tanpa mengalami akumulasi jeda waktu (timing drift).

Dalam ekosistem sistem tertanam, pergeseran waktu (timing drift) sering kali menjadi masalah kronis ketika mikrokontroler menangani tugas yang tumpang tindih antara pembacaan sensor dan transmisi jaringan. Untuk mengatasi hal ini, arsitektur firmware dalam penelitian ini tidak menggunakan fungsi penundaan pemblokiran standar (seperti `vTaskDelay` atau `delay()`). Sebagai gantinya, siklus telemetri dibungkus dalam FreeRTOS task terisolasi yang memanfaatkan fungsi `xTaskDelayUntil()`. Fungsi ini menghitung secara deterministik waktu yang dihabiskan oleh instruksi pembacaan I2C dan tumpukan jaringan (network stack) lwIP, lalu mengkompensasi sisa waktu tunggu agar siklus bangun (wake-up cycle) berikutnya tetap berada pada interval yang presisi (misalnya tepat 1.000 milidetik).

Implementasi Driver Sensor dan Pembacaan Register. Driver `bmp280.c` membaca 6 byte data kalibrasi pabrik (`dig_T1`, `dig_T2`, `dig_T3`) dari register 0x88 saat inisialisasi. Register `REG_CTRL_MEAS` (0xF4) dikonfigurasi dengan nilai heksadesimal 0x23 untuk mengaktifkan oversampling suhu 1x dan mode normal, sedangkan `REG_CONFIG` (0xF5) diatur pada 0x00 untuk standby 0,5 ms. Rumus kompensasi matematika dari datasheet Bosch diimplementasikan dalam operasi integer 32-bit untuk menghasilkan nilai suhu terkalibrasi presisi tinggi. Pada driver `ina226.c`, register konfigurasi `REG_CONFIG` (0x00) diatur pada nilai 0x4127 (mode kontinu penginderaan tegangan bus dan shunt, waktu konversi 1,1 ms, rata-rata 1 sampel). Nilai kalibrasi dihitung dan ditulis ke register `REG_CALIBRATION` (0x05) berdasarkan nilai shunt $0,1\ \Omega$ dan LSB arus $100\ \mu\text{A/bit}$. Pembacaan tegangan bus (`REG_BUS_V` 0x02) dan arus (`REG_CURRENT` 0x04) dikonversi secara real-time menjadi daya listrik seketika dalam satuan miliwatt ($P = V \times I$). Tabel 2 merangkum pemetaan pinout GPIO dan konfigurasi register bus I2C pada sistem.

Mekanisme Transmisi Multiprotokol dan Sinkronisasi Waktu. Firmware dirancang mendukung tiga mode komunikasi lapisan aplikasi melalui sakelar konfigurasi Kconfig: (1) HTTP POST memanfaatkan komponen `esp_http_client`, di mana setiap transmisi membuka sesi TCP, mengirimkan header Content-Type: application/json, menyalurkan string payload, dan menutup koneksi (stateless); (2) MQTT QoS 0 menggunakan komponen `esp_mqtt_client` yang mempublikasikan data ke topik sensor/suhu/data melalui koneksi TCP persisten tanpa menunggu konfirmasi (fire-and-forget); dan (3) MQTT QoS 1 yang menggunakan koneksi persisten serupa namun menunggu diterimanya frame PUBACK dari broker sebelum menyelesaikan siklus. Sebelum transmisi dimulai, modul `wifi_conn.c` menjalankan sinkronisasi jam epoch SNTP via `esp_netif_sntp` ke server waktu lokal. Timestamp epoch

berpresisi milidetik (`ts_send`) disematkan pada setiap payload JSON yang disusun menggunakan pustaka `cJSON`, seperti disajikan pada Tabel 3.

Desain Server Kolektor Asinkron dan Manajemen Basis Data. Sisi server dibangun dalam satu proses tunggal terpadu berbasis framework Python FastAPI (ASGI) dan Paho MQTT Client v2. Arsitektur ini mengeliminasi kebutuhan proses backend terpisah. Request HTTP POST diterima pada endpoint asinkron `/api/data`, sedangkan pesan MQTT diterima oleh fungsi callback `on_mqtt_message` yang berjalan pada thread terpisah (`loop_start`). Seluruh data yang masuk diparsing dan divalidasi secara ketat oleh skema model Pydantic Reading, yang memeriksa kesesuaian tipe data, validitas nilai QoS (-1 untuk HTTP, 0/1 untuk MQTT), dan rentang nomor urut. Data yang lolos validasi dicatat waktu kedatangannya (`ts_recv`) menggunakan resolusi nanodetik (`time.time_ns() // 1_000_000`) dan disimpan ke tabel `readings` pada basis data SQLite. Basis data dikonfigurasi menggunakan mode Write-Ahead Logging (PRAGMA `journal_mode=WAL`) dengan penguncian thread (`threading.Lock`), menjamin transaksi atomik dan mencegah penguncian basis data (database lock) ketika data HTTP dan MQTT masuk secara simultan.

Pada sisi backend, arsitektur server dirancang menggunakan pendekatan asinkron sepenuhnya berbasis framework FastAPI. Berbeda dengan arsitektur sinkron konvensional (seperti Flask atau Django) yang mengalokasikan satu thread per permintaan, ASGI (Asynchronous Server Gateway Interface) pada FastAPI menggunakan event loop tunggal untuk menangani ribuan koneksi idle secara konkuren. Data yang masuk melalui endpoint HTTP POST atau subscriber MQTT pertama-tama akan ditangkap dan dilewatkan ke modul validasi Pydantic. Pydantic bertugas sebagai lapisan keamanan (middleware security) yang memverifikasi tipe data secara ketat. Jika struktur JSON yang diterima tidak lengkap atau mengalami distorsi tipe data (misalnya string pada field suhu yang seharusnya float), server secara otomatis akan menolak permintaan tersebut dengan kode status HTTP 422 (Unprocessable Entity) tanpa membebani antrean tulis basis data.

3. Hasil dan Pembahasan

Hasil implementasi sistem pemantauan suhu multiprotokol diverifikasi melalui pengujian fungsional modul perangkat keras, firmware, dan server kolektor. Bukti keberhasilan rancang bangun disajikan melalui tabel spesifikasi komponen, pemetaan register, struktur payload telemetri, serta matriks verifikasi fungsional sistem.

Tabel 1. Spesifikasi Komponen Perangkat Keras dan Perangkat Lunak Sistem

Komponen / Modul	Spesifikasi Teknis	Peran / Fungsi dalam Sistem
ESP32 DevKit V1	Dual Core Xtensa 240 MHz, 520 KB SRAM, Wi-Fi 802.11 b/g/n	Unit pemroses utama dan modul komunikasi data telemetri
Bosch BMP280	I2C Interface, Resolusi 0,01°C, Rentang -40°C s.d. +85°C	Penginderaan parameter temperatur lingkungan fisik
TI INA226 + Shunt	16-bit ADC, Shunt Resistor 0,1 Ω (1%), Resolusi Tegangan 1,25 mV	Pengukuran simultan tegangan bus, arus listrik, dan konsumsi daya node
Firmware Framework	ESP-IDF v5.x + FreeRTOS Real-Time Kernel (Native C)	Manajemen task penjadwalan presisi, driver I2C, dan network stack
MQTT Broker	Eclipse Mosquitto v2.x (Port 1883)	Perantara pesan terdistribusi berbasis publish/subscribe
Backend Collector	Python 3.12, FastAPI (Uvicorn ASGI) + Paho	Penerima multi-jalur HTTP/MQTT asinkron dan

Basis Data	MQTT Client SQLite 3 (WAL Mode) + Indeks Komposit Grup Uji	validasi data Pydantic Penyimpanan data telemetri persisten terindeks dan bebas penguncian
------------	--	---

Tabel 2. Konfigurasi Pinout GPIO dan Pemetaan Register Bus I2C

Perangkat	Pin / Jalur ESP32	Alamat I2C	Register Kunci	Konfigurasi Register & Fungsi
I2C Master Bus	GPIO 21 (SDA), GPIO 22 (SCL)	I2C_NUM_0	-	Clock speed 400 kHz, Internal Pull-Up Enable Nilai 0x23: Temp oversampling x1, Pressure skip, Mode normal Membaca 6 byte konstanta kalibrasi suhu dig_T1, dig_T2, dig_T3 Nilai 0x4127: Shunt+Bus Continuous, Conv time 1.1ms, Avg 1 Nilai 512: Kalibrasi shunt 0,1 Ω dengan resolusi arus 100 μA/bit Catu daya tereregulasi 5,0 V diseri ke Shunt Resistor INA226
Bosch BMP280	SDA -> D21, SCL -> D22	0x76	REG_CTRL_MEAS (0xF4)	
Bosch BMP280	SDO -> GND	0x76	REG_CALIB_00 (0x88)	
TI INA226	SDA -> D21, SCL -> D22	0x40	REG_CONFIG (0x00)	
TI INA226	A0=GND, A1=GND	0x40	REG_CALIBRATION (0x05)	
Power Supply	VIN (5V DC), GND	-	-	

Tabel 3. Struktur dan Skema Payload Data Telemetri JSON Terstandarisasi

Field JSON	Tipe Data	Contoh Nilai	Deskripsi dan Validasi Skema Pydantic
device_id	String	"esp32_01"	Identifier unik perangkat keras pengirim telemetri Jalur protokol aplikasi yang digunakan (Literal 'mqtt', 'http')
protocol	String	"mqtt" / "http"	Tingkat QoS MQTT (0 atau 1); bernilai -1
qos	Integer	0, 1, atau -1	

seq	Integer	0 s.d. 29	husus untuk protokol HTTP Nomor urut paket data per sesi pengujian (Assert seq >= 0)
ts_send	Integer (int64)	1723812000123	Timestamp epoch milidetik saat pengiriman paket disinkronkan SNTP
temperature_c	Float	33.20	Data temperatur lingkungan terkalibrasi dari sensor BMP280 (°C)
bus_voltage_v	Float	4.965	Tegangan bus catu daya ESP32 terbaca oleh sensor INA226 (Volt)
current_ma	Float	53.24	Kuat arus listrik sesaat perangkat terukur oleh INA226 (miliampere)
power_mw	Float	264.34	Konsumsi daya listrik hasil kalkulasi instan $P = V \times I$ (miliwatt)
rssi	Integer	-47	Kekuatan sinyal Wi- Fi yang diterima ESP32 (Received Signal Strength Indicator, dBm)

Tabel 4. Rekapitulasi Verifikasi Fungsionalitas Modul Sistem Telemetri

Modul / Fungsi Sistem	Skenario Uji Verifikasi	Hasil Pengamatan dan Respon Sistem	Status Verifikasi
Driver Bus I2C Bersama	Akuisisi simultan BMP280 dan INA226 pada 400 kHz	Kedua sensor terbaca valid tanpa terjadi bus lock atau I2C error	Berhasil (100%)
Penjadwalan FreeRTOS	Eksekusi loop transmisi periodik (1s, 3s, 5s)	Interval stabil tanpa timing drift berkat xTaskDelayUntil()	Berhasil (100%)
Sinkronisasi Waktu SNTP	Koneksi ke server SNTP lokal saat booting Wi-Fi	Jam internal tersinkronisasi presisi milidetik (ts_send valid)	Berhasil (100%)
Transmisi HTTP POST	Pengiriman 270 paket via esp_http_client ke port 8000	Server merespon 201 Created; payload tersimpan utuh di basis data	Berhasil (100%)
Transmisi MQTT QoS 0	Pengiriman 270 paket publish ke topik Mosquitto	Pesan diterima instan oleh Paho subscriber tanpa	Berhasil (100%)

Transmisi MQTT QoS 1	Pengiriman 270 paket dengan konfirmasi PUBACK broker	overhead handshaking Semua paket sampai; duplikasi terisolasi otomatis oleh server	Berhasil (100%)
Validasi Backend Server	Injeksi payload valid vs payload cacat (malformed JSON)	Pydantic menerima data valid dan menolak payload rusak dengan error 422	Berhasil (100%)
Penyimpanan Basis Data	Pencatatan 810 baris data konkuren mode SQLite WAL	Seluruh 810 paket unik tersimpan tanpa database concurrency locking	Berhasil (100%)

Keberhasilan implementasi firmware modular berbasis ESP-IDF dan FreeRTOS pada penelitian ini membuktikan efektivitas pendekatan sistem tertanam native C dibandingkan framework tingkat tinggi berbasis Arduino. Penggunaan fungsi `xTaskDelayUntil()` terbukti secara deterministik mengeliminasi akumulasi pergeseran waktu pada interval transmisi 1, 3, dan 5 detik, yang sering menjadi kendala utama pada implementasi loop konvensional pada mikrokontroler kelas ESP32 (Mishra et al., 2021). Tingkat efisiensi dan responsivitas penjadwalan waktu-nyata ini juga sejalan dengan pengujian performa FreeRTOS pada arsitektur multi-core ESP32 yang dibuktikan oleh (Arm et al., 2022). Pendekatan modular pada berkas driver terpisah (`bmp280.c`, `ina226.c`, dan `wifi_conn.c`) mempermudah pemeliharaan kode dan memungkinkan portabilitas tinggi ke mikrokontroler keluarga Espressif lainnya, sejalan dengan prinsip arsitektur sistem tertanam yang direkomendasikan oleh (Dizdarević et al., 2019; Salman & Jain, 2017).

Dari perspektif integrasi antarmuka perangkat keras, keputusan mengoperasikan dua sensor digital (BMP280 pada alamat 0x76 dan INA226 pada alamat 0x40) pada bus I2C bersama berfrekuensi 400 kHz terbukti andal tanpa memicu terjadinya bus contention atau data corruption. Temuan ini menguatkan hasil studi (Bakare & Abubaker, 2026) yang membuktikan bahwa mikrokontroler ESP32 mampu menangani akuisisi multi-sensor berkecepatan tinggi secara presisi pada bus I2C tunggal. Integrasi sensor pemantau daya digital INA226 secara seri pada jalur catu daya perangkat keras memberikan kemampuan instrumentasi kelistrikan tingkat lanjut (arus dan daya per paket) yang belum pernah diintegrasikan secara terpadu pada studi purwarupa pemantauan suhu sebelumnya oleh (Atmoko et al., 2017; Grgić et al., 2016). Pendekatan instrumentasi terintegrasi ini memungkinkan sistem telemetri melakukan pemantauan konsumsi energi dirinya sendiri (self-energy diagnostic), yang sangat penting bagi penerapan node sensor pada lokasi terpencil sebagaimana disorot dalam studi konsumsi daya perangkat IoT oleh (Jara Ochoa et al., 2023).

Penyediaan arsitektur transmisi multiprotokol yang mendukung HTTP POST, MQTT QoS 0, dan MQTT QoS 1 secara berdampingan dalam satu basis firmware memberikan fleksibilitas operasional yang unggul. Pada skenario pertukaran data yang membutuhkan kompatibilitas web langsung, modul HTTP POST berhasil mengeksekusi siklus stateless request-response standar menuju REST API publik, sebagaimana dipelajari dalam arsitektur integrasi cloud oleh (Al-Masri et al., 2020). Di sisi lain, modul MQTT QoS 0 dan QoS 1 memberikan keunggulan transmisi berbasis publish/subscribe di atas soket TCP persisten yang sangat hemat overhead header, mengonfirmasi analisis teoritis protokol ringan oleh (Naik, 2017). Implementasi sinkronisasi jam epoch SNTP lokal pada firmware sebelum transmisi juga berhasil memecahkan tantangan pengukuran latensi satu arah presisi tinggi yang sering diabaikan dalam rancang bangun sistem IoT serupa (Dizdarević et al., 2019).

Dalam penanganan keandalan pesan pada tingkat aplikasi, implementasi MQTT QoS 1 menghadirkan dinamika retransmisi semantik At Least Once ketika terjadi keterlambatan frame konfirmasi PUBACK dari broker akibat fluktuasi jaringan. Sistem berhasil mengatasi tantangan ini dengan mengimplementasikan mekanisme windowing query berbasis sequence number pada server penampung, yang secara otomatis mengidentifikasi dan mengisolasi paket duplikat tanpa mengganggu integritas aliran data utama. Pendekatan mitigasi duplikasi ini selaras dengan rekomendasi arsitektur perpesanan terdistribusi oleh (Mishra & Kertesz, 2020) serta temuan terkait kerentanan dan keterbatasan protokol MQTT pada koneksi tidak stabil oleh (Dinculeană & Cheng, 2019).

Pada lapisan backend penerima, penggabungan server asinkron FastAPI dan subscriber Mosquitto dalam satu proses tunggal terbukti mampu menyederhanakan arsitektur server sekaligus meningkatkan efisiensi penggunaan memori dan CPU. Pendekatan dinamis ini terbukti efektif dalam memproses beban pesan yang berfluktuasi secara real-time pada lingkungan IoT, selaras dengan arsitektur FastAPI terukur yang diusulkan oleh (Mabotha et al., 2025). Pemanfaatan Pydantic v2 untuk validasi skema data secara otomatis memastikan bahwa setiap paket telemetry yang masuk diverifikasi tipe data dan integritasnya sebelum dieksekusi ke basis data, mencegah terjadinya injeksi data korup pada tabel penampung. Selain itu, konfigurasi basis data SQLite dalam mode Write-Ahead Logging (WAL) dengan thread locking connection pool terbukti sukses mengeliminasi kendala database locking saat menerima konkurensi data HTTP dan MQTT secara simultan dari 810 record pengujian eksperimental, sebuah performa persistensi pada skenario konkurensi tinggi yang juga diverifikasi keandalannya dalam kajian (Baek & Park, 2026). Keberhasilan pengujian fungsional secara utuh dengan tingkat keberhasilan 100% pada seluruh modul ini memvalidasi bahwa sistem yang dibangun memenuhi standar rekayasa perangkat lunak dan perangkat keras modern untuk sistem telemetry IoT industri dan pemantauan lingkungan berkelanjutan, sejalan dengan kajian evaluasi kinerja protokol pengelolaan data IoT oleh (Chen & Kunz, 2016).

4. Simpulan

Sistem monitoring suhu multiprotokol berbasis ESP32 dan server kolektor asinkron telah berhasil dirancang, diimplementasikan, dan diverifikasi secara fungsional. Firmware modular ESP-IDF berhasil mengeksekusi penjadwalan task FreeRTOS bebas drift, mengintegrasikan sensor BMP280 dan INA226 pada bus I2C bersama 400 kHz, menyinkronkan waktu via SNTP, serta menyalurkan payload telemetry melalui HTTP POST, MQTT QoS 0, dan QoS 1. Server kolektor FastAPI dan SQLite WAL terbukti andal mencatat seluruh data secara atomik. Sistem ini direkomendasikan sebagai arsitektur terbuka yang andal untuk node telemetry IoT industri.

5. Ucapan Terima Kasih

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada Laboratorium Sistem Komputer, Fakultas Ilmu Komputer, yang telah menyediakan fasilitas perangkat keras mikrokontroler, modul sensor, dan infrastruktur emulasi jaringan untuk mendukung kelancaran pengujian penelitian ini.

6. Daftar Pustaka

Al-Masri, E., Kalyanam, K. R., Batts, J., Kim, J., Singh, S., Vo, T., & Yan, C. (2020). Investigating Messaging Protocols for the Internet of Things (IoT). *IEEE Access*, 8, 94880–94911. <https://doi.org/10.1109/ACCESS.2020.2993363>

- Arm, J., Baštán, O., Mihálik, O., & Bradáč, Z. (2022). Measuring the Performance of FreeRTOS on ESP32 Multi-Core. *IFAC-PapersOnLine*, 55(4), 292–297. <https://doi.org/10.1016/j.ifacol.2022.06.048>
- Atmoko, R. A., Riantini, R., & Hasin, M. K. (2017). IoT real time data acquisition using MQTT protocol. *Journal of Physics: Conference Series*, 853, 012003. <https://doi.org/10.1088/1742-6596/853/1/012003>
- Baek, Y., & Park, J. K. (2026). ALEX: Adaptive Log-Embedded Extent Layer for Low-Amplification SQLite Writes on Flash Storage. *Applied Sciences*, 16(2), 672. <https://doi.org/10.3390/app16020672>
- Bakare, M. S., & Abubaker, K. (2026). IoT-based indoor environmental monitoring system using multi-parameter sensing and ESP32-WROOM integration. *Discover Electronics*, 3(1), 6. <https://doi.org/10.1007/s44291-026-00157-3>
- Chen, Y., & Kunz, T. (2016). Performance evaluation of IoT protocols under a constrained wireless access network. *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, 1–7. <https://doi.org/10.1109/MoWNeT.2016.7496622>
- Dinculeană, D., & Cheng, X. (2019). Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices. *Applied Sciences*, 9(5), 848. <https://doi.org/10.3390/app9050848>
- Dizdarević, J., Carpio, F., Jukan, A., & Masip-Bruin, X. (2019). A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration. *ACM Computing Surveys*, 51(6), 1–29. <https://doi.org/10.1145/3292674>
- Grgić, K., Špeh, I., & Heđi, I. (2016). A web-based IoT solution for monitoring data using MQTT protocol. *2016 International Conference on Smart Systems and Technologies (SST)*, 249–253. <https://doi.org/10.1109/SST.2016.7765668>
- Jara Ochoa, H. J., Peña, R., Ledo Mezquita, Y., Gonzalez, E., & Camacho-Leon, S. (2023). Comparative Analysis of Power Consumption between MQTT and HTTP Protocols in an IoT Platform Designed and Implemented for Remote Real-Time Monitoring of Long-Term Cold Chain Transport Operations. *Sensors*, 23(10), 4896. <https://doi.org/10.3390/s23104896>
- Mabotha, E., Mabunda, N. E., & Ali, A. (2025). A Dockerized Approach to Dynamic Endpoint Management for RESTful Application Programming Interfaces in Internet of Things Ecosystems. *Sensors*, 25(10), 2993. <https://doi.org/10.3390/s25102993>
- Mishra, B., & Kertesz, A. (2020). The Use of MQTT in M2M and IoT Systems: A Survey. *IEEE Access*, 8, 201071–201086. <https://doi.org/10.1109/ACCESS.2020.3035849>
- Mishra, B., Mishra, B., & Kertesz, A. (2021). Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements. *Energies*, 14(18), 5817. <https://doi.org/10.3390/en14185817>
- Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *2017 IEEE International Systems Engineering Symposium (ISSE)*, 1–7. <https://doi.org/10.1109/SysEng.2017.8088251>
- Salman, T., & Jain, R. (2017). A Survey of Protocols and Standards for Internet of Things. *Advanced Computing and Communications*. <https://doi.org/10.34048/2017.1.F3>