BAB 3
METODOLOGI PENELITIAN

3.1 Desain Penelitian

Desain penelitian ini menerapkan metode *prototyping* yang dilakukan secara bertahap, dimulai dari perancangan skema perangkat keras, pengembangan logika *firmware*, hingga integrasi antarmuka pengguna. Pendekatan ini dipilih untuk memvalidasi stabilitas komunikasi data antara protokol CANBus dan MQTT pada setiap tahapan pengembangan sebelum dilakukan pengujian skala penuh.

3.1.1 Jenis dan Pendekatan Penelitian

Penelitian ini termasuk ke dalam jenis penelitian *experimental research* dengan pendekatan rekayasa sistem. Penelitian dilakukan dengan merancang dan mengimplementasikan BEMS berbasis mikrokontroler yang berfungsi untuk memantau kondisi lingkungan secara *real-time* serta menyediakan data yang terstruktur untuk keperluan evaluasi dan analisis sistem.

Sistem dibangun menggunakan kombinasi perangkat keras berupa Arduino Mega dan ESP32 yang saling berkomunikasi melalui protokol CANBus menggunakan modul MCP2515. Data dari berbagai sensor dikumpulkan terlebih dahulu oleh Arduino Mega, kemudian diteruskan melalui modul MCP2515 pertama menuju modul MCP2515 kedua yang terhubung ke ESP32. Selanjutnya, data dikirimkan ke MQTT broker menggunakan Mosquitto dan divisualisasikan secara *real-time* melalui *dashboard* berbasis Node.js. Selain itu, data sensor disimpan pada basis data MongoDB untuk keperluan pencatatan historis, analisis tren, serta evaluasi performa sistem pemantauan.

Pendekatan penelitian ini bersifat kuantitatif karena melibatkan proses pengumpulan, pengolahan, dan analisis data numerik yang diperoleh dari sensor lingkungan dan keselamatan. Analisis dilakukan untuk mengevaluasi kestabilan sistem, keandalan komunikasi data, serta respons sistem terhadap perubahan kondisi lingkungan. Dengan demikian, penelitian ini memadukan pengembangan perangkat keras, integrasi sistem IoT, dan pengolahan data berbasis sistem pemantauan untuk menghasilkan solusi pemantauan lingkungan yang andal dan terstruktur.

3.1.2 Lokasi dan Waktu Penelitian

Penelitian ini dilakukan di Laboratorium IoT, Program Studi Informatika, Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara. Laboratorium ini dipilih karena memiliki fasilitas yang mendukung pengembangan sistem pemantauan lingkungan, seperti Arduino Mega, ESP32, modul MCP2515 CANBus, serta infrastruktur jaringan untuk pengujian sistem IoT dan integrasi dengan MQTT *broker*.

Sebagian pengujian dilakukan pada lingkungan simulasi untuk memastikan akurasi pembacaan sensor dan keandalan komunikasi antar perangkat. Proses pengembangan perangkat lunak, termasuk pembuatan *dashboard* berbasis Node.js dan integrasi dengan Mosquitto broker, dilakukan pada server lokal untuk memastikan sistem dapat divisualisasikan secara *real-time*.

Penelitian dilaksanakan pada bulan Agustus hingga Desember 2025 dengan rincian berikut:

- Agustus 2025: Perancangan sistem, studi literatur, penyusunan arsitektur pemantauan lingkungan, desain rangkaian perangkat keras, dan konfigurasi komunikasi CANBus.
- September 2025: Implementasi perangkat keras dan integrasi sensor melalui jaringan CANBus. Data dikirimkan ke ESP32 dan diteruskan ke MQTT broker untuk ditampilkan pada *dashboard*.
- Oktober 2025: Pengembangan dan penyempurnaan perangkat lunak, optimasi pengiriman data, pengujian komunikasi antar mikrokontroler, serta pengembangan *dashboard*.
- November 2025: Pengumpulan dan pencatatan data historis lingkungan serta konsumsi energi, disertai analisis tren perubahan kondisi lingkungan.
- Desember 2025: Evaluasi akurasi sensor, kestabilan komunikasi CANBus dan MQTT, serta performa sistem pemantauan dan *dashboard*.

3.1.3 Prosedur Penelitian

Prosedur penelitian menjelaskan tahapan yang dilakukan mulai dari perencanaan hingga evaluasi sistem. Prosedur ini disusun agar penelitian berjalan

sistematis dan sesuai tujuan pengembangan *Building Energy Management System* berbasis IoT.

Tahapan prosedur penelitian terdiri dari enam langkah utama:

1. **Studi Literatur**
   Mengumpulkan referensi terkait sistem pemantauan lingkungan, arsitektur BEMS, komunikasi CANBus, integrasi Arduino Mega dan ESP32, serta teknologi IoT untuk pengiriman dan visualisasi data sensor.

2. **Analisis Kebutuhan Sistem**
   Mengidentifikasi kebutuhan perangkat keras dan perangkat lunak, meliputi jenis sensor lingkungan dan keselamatan, konfigurasi komunikasi CANBus, mekanisme pengiriman data melalui MQTT, kebutuhan basis data, serta spesifikasi *dashboard* pemantauan.

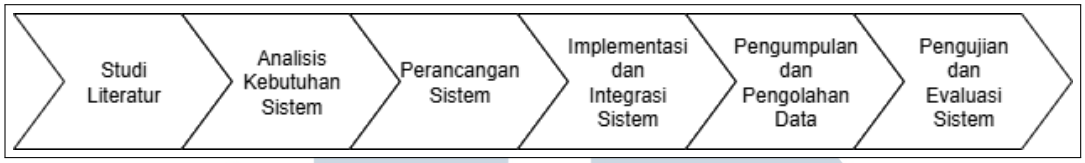
3. **Perancangan Sistem**
   Merancang arsitektur sistem secara keseluruhan, termasuk alur komunikasi data dari sensor ke *dashboard*, skema koneksi perangkat keras, konfigurasi protokol CANBus, serta perancangan perangkat lunak untuk pengiriman data melalui MQTT. Struktur basis data dan tampilan *dashboard* juga dirancang pada tahap ini.

4. **Implementasi dan Integrasi Sistem**
   Melakukan perakitan perangkat keras, pemrograman mikrokontroler, pengujian awal komunikasi CANBus, serta integrasi dengan broker MQTT Mosquitto dan *dashboard* interaktif berbasis kerangka kerja Next.js yang memanfaatkan koneksi WebSocket untuk memvisualisasikan data sensor secara *real-time*.

5. **Pengumpulan dan Pengolahan Data**
   Mengumpulkan data sensor melalui jaringan CANBus secara berkelanjutan. Data yang diterima dari MQTT broker disimpan ke dalam basis data untuk keperluan pencatatan historis, visualisasi, dan analisis tren kondisi lingkungan serta konsumsi energi.

6. **Pengujian dan Evaluasi Sistem**
   Melakukan evaluasi terhadap akurasi pembacaan sensor, kestabilan komunikasi CANBus dan MQTT, latensi pengiriman data, serta keandalan *dashboard* dalam menampilkan informasi pemantauan secara *real-time*.

Diagram alur prosedur penelitian ditunjukkan pada Gambar 3.1.



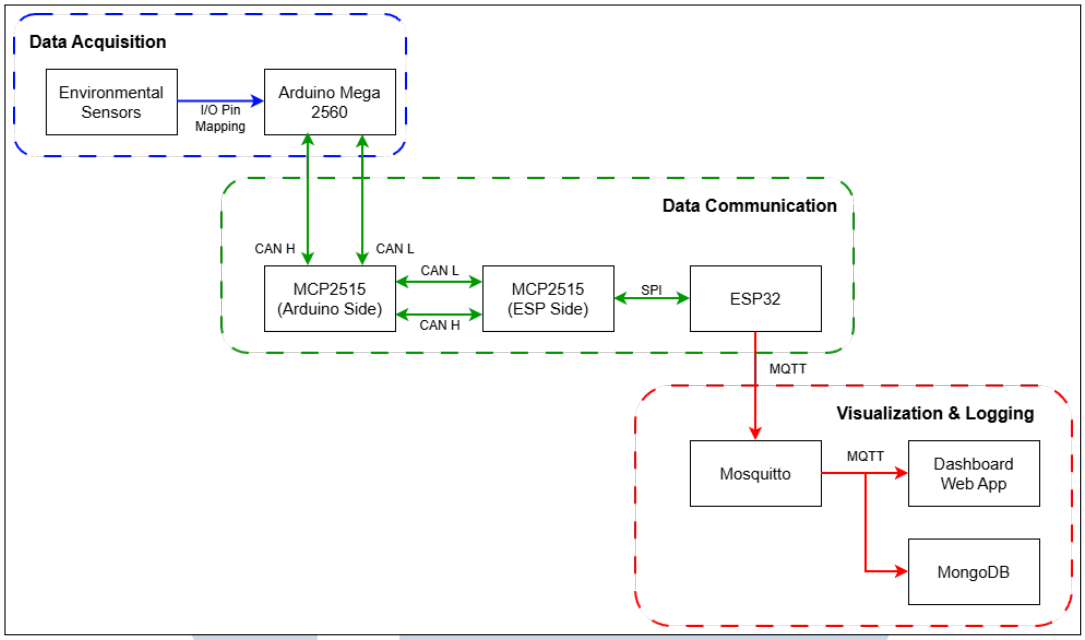
Gambar 3.1. Diagram alur prosedur penelitian.

3.2 Perancangan Sistem

Proses perancangan sistem dilakukan melalui pendekatan *bottom-up*, dimulai dari pemilihan sensor yang memiliki antarmuka digital untuk meminimalkan *noise*, kemudian merancang skema komunikasi *wired* menggunakan CANBus untuk mengatasi keterbatasan jarak I2C/SPI. Perancangan dilanjutkan dengan mendesain struktur data *Unified Payload* agar sinkronisasi waktu antar sensor tetap terjaga saat dikirimkan ke *gateway*.

3.2.1 Arsitektur Umum Sistem

Arsitektur umum sistem dirancang untuk mendukung proses akuisisi, pengiriman, penyimpanan, serta visualisasi data energi dan lingkungan gedung secara *real-time* sebagai bagian dari BEMS. Sistem ini mengintegrasikan perangkat keras berbasis mikrokontroler dengan infrastruktur komunikasi data dan *web dashboard* untuk pemantauan terpusat.



Gambar 3.2. Arsitektur *dataflow* sistem BEMS.

Data diperoleh dari sensor lingkungan dan energi yang terhubung ke Arduino Mega 2560 sebagai node akuisisi data. Data sensor dikumpulkan dan dikirimkan melalui modul MCP2515 pada sisi Arduino Mega menuju modul MCP2515 pada sisi ESP32 menggunakan protokol CANBus yang bersifat deterministik dan stabil.

ESP32 berperan sebagai *gateway* IoT yang meneruskan data hasil akuisisi ke jaringan menggunakan protokol MQTT melalui *Mosquitto Broker*. Data yang diterima selanjutnya dipublikasikan ke aplikasi *dashboard* berbasis Node.js untuk ditampilkan secara *real-time* kepada pengguna.

Selain ditampilkan pada *dashboard*, data energi dan lingkungan juga disimpan ke dalam basis data MongoDB sebagai data historis. Data historis ini digunakan untuk kebutuhan pencatatan, visualisasi tren, serta evaluasi performa sistem BEMS dan konsumsi energi gedung dalam periode tertentu.

Secara umum, arsitektur sistem BEMS ini terdiri atas tiga lapisan utama sebagai berikut:

1. **Lapisan Akuisisi Data**
   Terdiri dari sensor lingkungan dan energi, Arduino Mega 2560, serta modul MCP2515 yang berfungsi untuk membaca, mengolah awal, dan mengirimkan data melalui jaringan CANBus.

2. **Lapisan Komunikasi dan Integrasi**
   Mencakup ESP32 sebagai *gateway*, protokol CANBus untuk komunikasi lokal, MQTT sebagai protokol pengiriman data ke jaringan, serta *Mosquitto Broker* sebagai pengelola pertukaran data.

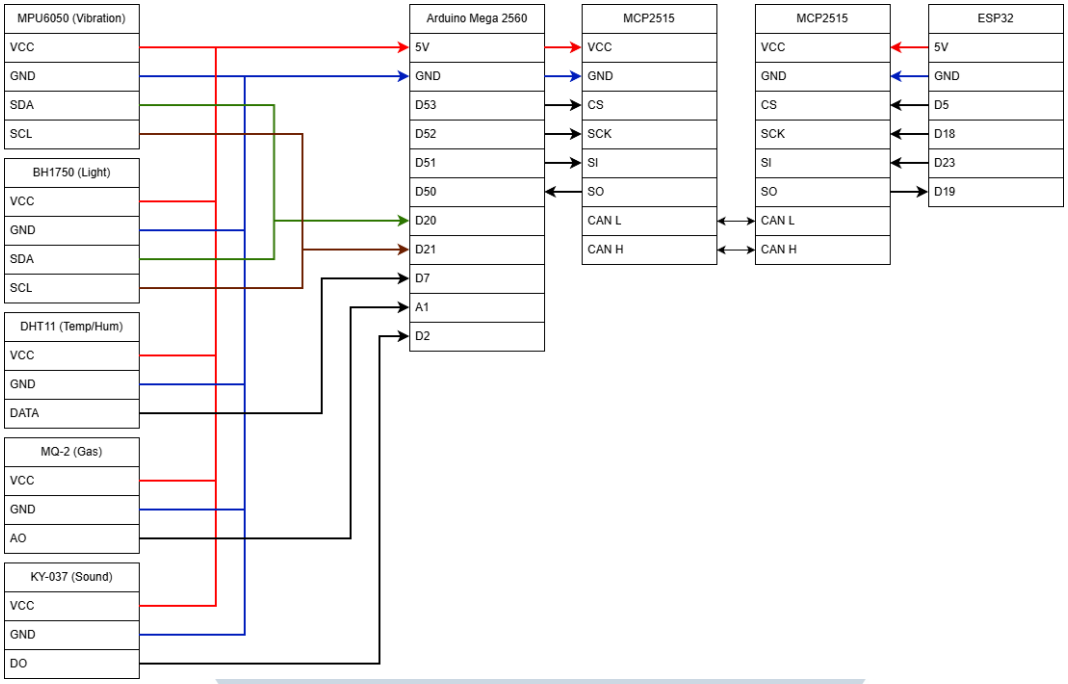
3. **Lapisan Aplikasi dan Visualisasi**
   Lapisan ini terdiri dari *Web Dashboard* berbasis Next.js untuk visualisasi data secara *real-time* dan layanan *backend* berbasis Node.js yang terintegrasi dengan Prisma ORM. Layanan *backend* bertanggung jawab untuk mengelola logika penyimpanan data historis ke dalam basis data MongoDB, menjamin integritas data, serta menyediakan API.

3.2.2 Perancangan Perangkat Keras

Perangkat keras pada sistem ini dirancang berdasarkan empat representasi gambar yang menunjukkan tahapan detail dari integrasi sensor, mikrokontroler, dan modul komunikasi CANBus. Setiap gambar memiliki fungsi penjelasan yang berbeda, mulai dari pemetaan logis antar pin hingga tampilan rangkaian fisik yang lengkap.

A Diagram Koneksi Pin

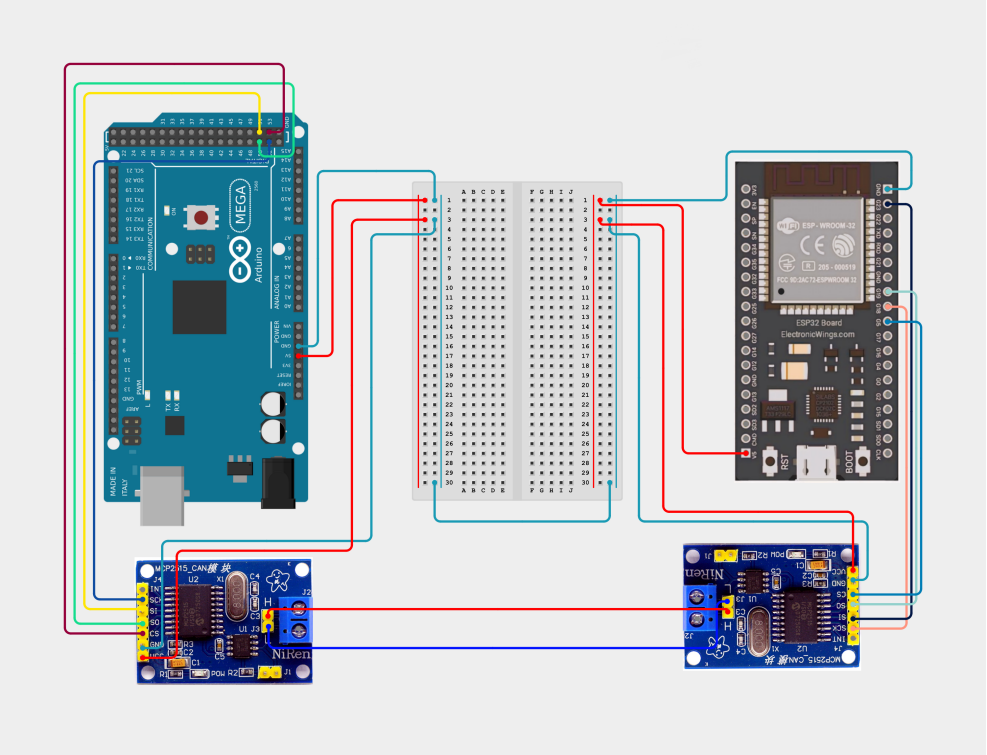
Gambar 3.3 menampilkan hubungan logis antara seluruh sensor, Arduino Mega 2560, dua modul MCP2515, dan ESP32. Diagram ini memperlihatkan jalur koneksi setiap pin, termasuk jalur I^2C (SDA dan SCL), jalur digital, jalur analog, serta jalur SPI yang digunakan untuk menghubungkan Arduino dan ESP32 dengan modul MCP2515. Diagram ini menjadi acuan utama untuk memastikan tidak ada konflik pin dan seluruh perangkat terhubung dengan konfigurasi yang benar.



Gambar 3.3. Diagram koneksi pin antara sensor, Arduino Mega, MCP2515, dan ESP32.

B Rangkaian CANBus antara Arduino dan ESP32

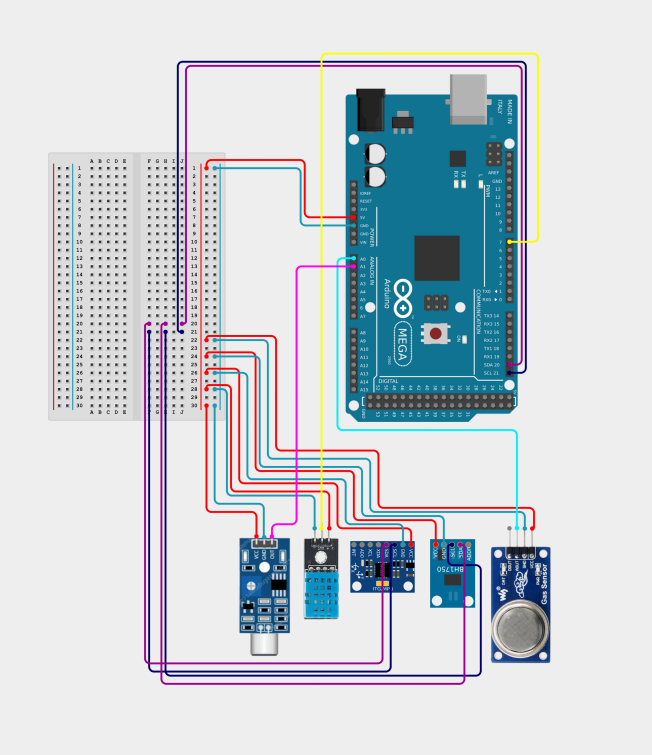
Gambar 3.4 menunjukkan sambungan fisik antara Arduino Mega, dua modul MCP2515, dan ESP32 menggunakan protokol CANBus. Diagram ini memperjelas bagaimana kedua modul MCP2515 dihubungkan melalui jalur CAN_H dan CAN_L, serta bagaimana masing-masing mikrokontroler mengakses modul tersebut melalui jalur SPI. Gambar ini menggambarkan fungsi CANBus sebagai penghubung utama antara *node* akuisisi data dan *node gateway*.



Gambar 3.4. Sambungan fisik jaringan CANBus antara Arduino Mega dan ESP32 melalui modul MCP2515.

C Integrasi Sensor Lengkap dengan Arduino Mega

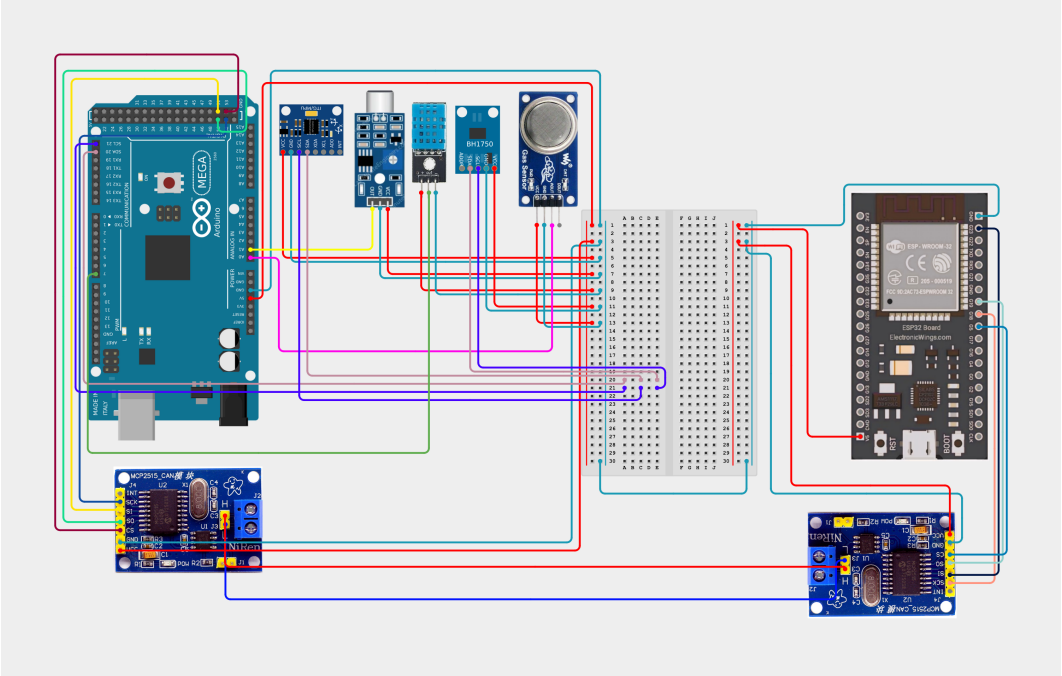
Gambar 3.5 memperlihatkan integrasi seluruh sensor ke Arduino Mega 2560. Setiap sensor ditampilkan dengan jalur koneksi yang terpisah untuk mempermudah identifikasi. Diagram ini menjelaskan bagaimana sensor MPU6050, BH1750, DHT11, MQ-2, dan KY-037 dihubungkan ke pin Arduino sesuai dengan karakteristik antarmuka masing-masing (I^2C, digital, atau analog). Gambar ini menjadi referensi penting dalam proses perakitan perangkat keras.



Gambar 3.5. Integrasi seluruh sensor lingkungan dengan Arduino Mega 2560.

D Rangkaian Fisik Keseluruhan Sistem

Gambar 3.6 menampilkan keseluruhan rangkaian fisik yang mencakup integrasi sensor, Arduino Mega, dua modul MCP2515, *breadboard* sebagai jalur distribusi, dan ESP32 sebagai *gateway*. Rangkaian ini menunjukkan susunan komponen sebagaimana diimplementasikan secara nyata, termasuk jalur suplai daya 5V, jalur *ground* bersama, jalur komunikasi SPI, serta jalur CANBus. Gambar ini menggambarkan bentuk akhir perangkat keras yang digunakan dalam penelitian.



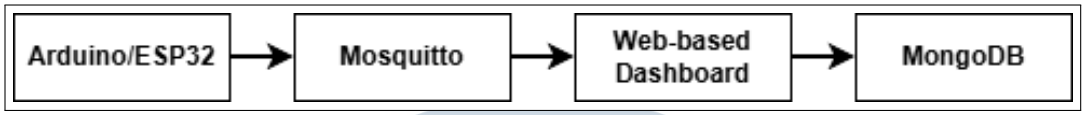
Gambar 3.6. Rangkaian fisik lengkap sistem pemantauan lingkungan berbasis Arduino Mega, MCP2515, dan ESP32.

Dengan keempat representasi gambar tersebut, seluruh proses perancangan perangkat keras dapat dijelaskan secara sistematis mulai dari hubungan logis antar pin hingga implementasi fisik pada rangkaian yang sesungguhnya. Pendekatan ini memastikan bahwa integrasi sensor, mikrokontroler, dan modul komunikasi berjalan konsisten dan sesuai dengan rancangan awal.

3.2.3 Perancangan Perangkat Lunak

Perancangan perangkat lunak pada penelitian ini berfokus pada pengembangan sistem yang mampu menerima, menyimpan, memvisualisasikan, dan mengelola data sensor yang dikirim melalui jaringan CANBus. Sistem perangkat lunak dirancang agar dapat bekerja secara terintegrasi dengan perangkat keras melalui protokol MQTT, serta mendukung proses analisis lanjutan berbasis data.

Gambar 3.7 menunjukkan alur umum arsitektur perangkat lunak yang dikembangkan pada penelitian ini.



Gambar 3.7. Diagram alur perancangan perangkat lunak sistem *monitoring* lingkungan

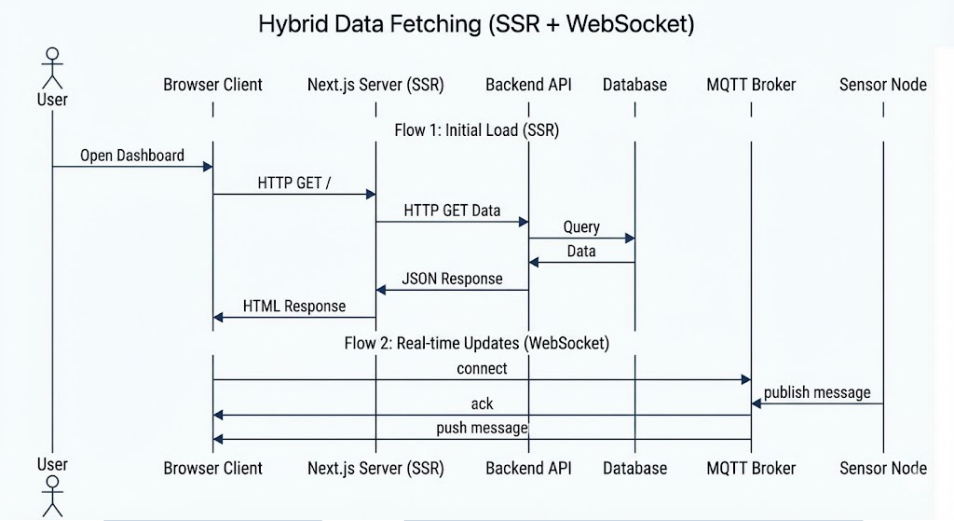
Secara keseluruhan, sistem perangkat lunak terdiri atas empat komponen utama yang saling terhubung, yaitu:

1. **Arduino Mega dan ESP32**
   Arduino Mega berfungsi sebagai *node* akuisisi data yang membaca seluruh sensor lingkungan. Data kemudian dikemas menjadi *frame* CAN dan dikirim melalui modul MCP2515 menuju ESP32. ESP32 menerima *frame* tersebut, melakukan *parsing* data, dan mengirimkannya ke jaringan menggunakan protokol MQTT dalam format JSON. ESP32 juga dapat melakukan *preprocessing* seperti perataan nilai atau penambahan waktu pengambilan data.

2. **MQTT Broker (Mosquitto)**
   Mosquitto berfungsi sebagai penghubung antara ESP32 dan aplikasi web. Broker ini menerima data sensor yang dikirim oleh ESP32 dan mendistribusikannya kepada *client* yang *subscribe* pada topik tertentu. Penggunaan Mosquitto dipilih dikarenakan oleh implementasi ringan, latensi rendah, dan sesuai untuk kebutuhan komunikasi perangkat IoT.

3. **Web-based Dashboard (Next.js)**
   Web-based dashboard berbasis Next.js bertindak sebagai antarmuka visual utama sistem monitoring. Aplikasi ini mengadopsi arsitektur komunikasi hibrida: untuk visualisasi *real-time*, dashboard berlangganan langsung ke topik MQTT melalui protokol WebSocket. Sementara itu, untuk kebutuhan analisis historis, aplikasi mengakses data yang tersimpan di MongoDB melalui layanan *backend* (API) yang dikelola oleh Prisma ORM, memastikan pemisahan yang jelas antara jalur data cepat dan jalur pengarsipan data.

   Secara visual, alur distribusi data mulai dari permintaan awal pengguna hingga pembaruan angka secara langsung diilustrasikan pada Gambar 3.8. Diagram ini memperlihatkan bagaimana jalur HTTP digunakan untuk inisialisasi data historis, sementara jalur WebSocket digunakan untuk aliran data sensor berkelanjutan.



Gambar 3.8. Arsitektur pengambilan data *hybrid* (SSR dan WebSocket).

4. **Database (MongoDB)**

   Database MongoDB digunakan untuk menyimpan seluruh data sensor yang diterima dari aplikasi web. Struktur dokumen yang fleksibel mendukung penyimpanan parameter lingkungan yang bervariasi dan akses cepat untuk kebutuhan analisis. Basis data ini juga menyediakan *dataset* untuk pengembangan model analisis pada tahap berikutnya.

Alur perangkat lunak dimulai dari akuisisi data sensor yang dikirim melalui CANBus, diterima dan diproses oleh ESP32, lalu dikirim ke MQTT Broker. Aplikasi web mengambil data tersebut untuk ditampilkan secara *real-time* dan disimpan ke MongoDB. Data historis dapat dianalisis lebih lanjut untuk mendukung pemantauan kondisi lingkungan dan sistem peringatan dini.

Dengan rancangan ini, perangkat lunak mampu menyediakan pemantauan lingkungan secara *real-time*, visualisasi berbasis web, penyimpanan data jangka panjang, dan fondasi untuk analisis berbasis data pada tahap penelitian berikutnya.

3.3 Pengumpulan & Pengolahan Data

Bagian ini menjelaskan tahapan pengumpulan dan pengolahan data yang dilakukan dalam penelitian untuk mendukung sistem BEMS yang dikembangkan. Proses ini dirancang untuk memastikan data lingkungan dan energi yang diperoleh dari jaringan sensor bersifat akurat, konsisten, dan dapat digunakan secara andal untuk pemantauan kondisi gedung secara *real-time* maupun analisis historis.

Seluruh data diperoleh melalui sistem akuisisi berbasis CANBus, diteruskan melalui protokol MQTT, dan diolah sebelum ditampilkan pada *dashboard* serta disimpan ke dalam basis data.

3.3.1 Pengumpulan Data

Pengumpulan data pada sistem dilakukan melalui integrasi berbagai sensor lingkungan dengan jaringan CANBus. Arduino Mega 2560 berperan sebagai *node* utama yang membaca seluruh sensor dan mengemas hasil pembacaan ke dalam *CAN frame* berukuran 8 byte sesuai standar CAN 2.0B. Setiap jenis sensor diberikan *CAN ID* unik untuk memastikan bahwa ESP32 dapat mengidentifikasi isi *payload* secara deterministik. Tabel 3.1 menunjukkan struktur data yang dikirimkan melalui CANBus:

Tabel 3.1. Struktur Data Sensor pada CANBus

ID CAN (Hex)	Data Sensor	Tipe dan Panjang Data
0x12	Lux Level (BH1750)	1x float (4 bytes)
0x13	Temperature & Humidity (DHT11)	2x float (8 bytes)
0x14	Gas, Sound, Vibration, UV Status	1x uint16_t (2 bytes) + 3x uint8_t (3 bytes)

Arduino melakukan akuisisi data sensor secara periodik dengan siklus 1 detik, kemudian mengonversi nilai pembacaan ke dalam bentuk *byte array* menggunakan fungsi *memcpy()*. *Payload* yang telah dikemas dikirimkan melalui modul MCP2515 sehingga menghasilkan *frame* CAN yang konsisten.

Untuk memastikan akurasi penanda waktu (*timestamp*), sistem menerapkan mekanisme sinkronisasi berbasis *Request-Response*. ESP32 menginisiasi komunikasi dengan mengirimkan paket permintaan ke Arduino. Setelah Arduino merespons dengan mengirimkan data sensor, ESP32 menghitung durasi total perjalanan data atau *Round Trip Time* (RTT). Estimasi waktu tempuh dari Arduino ke ESP32 diperoleh dengan membagi nilai RTT tersebut dengan dua ($\frac{RTT}{2}$). Hasil perhitungan ini ditambahkan ke waktu sistem saat ini untuk menghasilkan *timestamp* data yang presisi.

Pada sisi penerima, ESP32 memantau seluruh CAN ID aktif dan melakukan identifikasi *payload* saat *frame* diterima. Nilai tersebut disimpan sebagai data mentah dalam *buffer* untuk diproses lebih lanjut. Pada tahap ini, data belum mengalami pembersihan ataupun perataan nilai, sehingga fokus utama adalah menjaga integritas akuisisi sensor melalui jaringan CANBus.

Setelah data berhasil diakuisisi oleh ESP32, proses transmisi menuju server (*broker*) dilakukan menggunakan protokol MQTT dengan konfigurasi *Quality of Service* (QoS) Level 0. Mekanisme ini dikenal sebagai *"at most once"* atau *fire-and-forget*, di mana paket data dikirimkan tanpa menunggu konfirmasi penerimaan (*acknowledgment*) dari penerima.

Pemilihan QoS 0 didasarkan pada karakteristik sistem pemantauan lingkungan yang membutuhkan latensi rendah dan efisiensi *bandwidth* tinggi. Mengingat data sensor diperbarui secara kontinu dengan frekuensi 1 Hz (satu data per detik), kehilangan satu paket data (*packet loss*) dianggap dapat ditoleransi karena akan segera tergantikan oleh data terbaru pada siklus berikutnya. Pendekatan ini meminimalkan beban komputasi pada mikrokontroler dan mengurangi *overhead* jaringan, sehingga menjamin aliran data yang responsif untuk visualisasi *real-time*.

3.3.2 Pengolahan Data

Pengolahan data pada penelitian ini dilakukan secara bertahap untuk memastikan data lingkungan dan energi yang diperoleh dari sistem BEMS memiliki kualitas, konsistensi, dan keterandalan yang memadai. Proses pengolahan mencakup akuisisi data dari jaringan CANBus, pemrosesan pada sistem *embedded*, serta penyimpanan data untuk keperluan visualisasi dan analisis historis.

Tahapan pengolahan data dibagi menjadi tiga bagian utama sebagai berikut.

1. **Pengolahan Data Awal**

   Data lingkungan dan konsumsi energi diperoleh dari sensor yang terhubung ke Arduino Mega dan dikirimkan dalam bentuk *frame* CANBus. Setiap sensor memiliki *identifier* CAN yang berbeda sehingga memungkinkan proses pemetaan data berdasarkan jenis parameter lingkungan yang dipantau.

   Data mentah yang diterima berupa nilai digital dari masing-masing sensor. Nilai tersebut kemudian dianalisis untuk menentukan hubungan antara data mentah dan satuan fisis yang sesuai, seperti derajat Celcius, persen kelembapan, lux, atau satuan energi listrik. Proses ini bertujuan memastikan bahwa data yang dikirimkan telah memiliki makna fisis yang representatif terhadap kondisi lingkungan gedung.

2. **Pengolahan Data *Real-time* pada Sistem Embedded**

   Pengolahan data secara *real-time* dilakukan pada sisi mikrokontroler dan *gateway*. Arduino Mega membaca data sensor, melakukan pengolahan dasar

seperti perhitungan rata-rata dan validasi nilai, kemudian mengirimkan data melalui CANBus ke ESP32.

ESP32 berfungsi sebagai *gateway* yang menyaring data CANBus berdasarkan *identifier* yang telah ditentukan, mengemas data dalam format JSON, dan mengirimkannya ke MQTT broker.

Perancangan struktur data pengiriman menggunakan format *Unified Payload* agar seluruh parameter sensor yang diterima dari Arduino Mega dapat dikirimkan dalam satu paket transmisi yang sinkron. Berdasarkan desain *firmware*, struktur JSON yang dirancang ditunjukkan pada Kode 3.1.

```
{
  "device_id": "String",    // ID Perangkat
  "lux": "Float",           // Sensor Cahaya
  "temperature": "Float",   // Sensor Suhu
  "humidity": "Float",      // Sensor Kelembapan
  "mq2_adc": "Integer",     // Sensor Gas (ADC)
  "sound": "Integer",       // Status Suara (0/1) atau ADC
  "vibration": "Integer",   // Status Getaran (0/1)
  "uv": "Integer",          // Status Api / UV (0/1)
  "latency_ms": "Float",    // Latensi Jaringan (RTT)
  "timestamp": "Long"       // Epoch Time
}
```

Kode 3.1: Rancangan struktur data JSON (Unified Payload)

Data yang diterima oleh aplikasi *web dashboard* selanjutnya akan diurai (*parsed*), ditampilkan secara *real-time*, dan disimpan ke dalam basis data MongoDB sebagai data historis.

3. **Penentuan Ambang Batas (*Threshold*) Status Sistem**

Untuk memvisualisasikan status indikator biner (Aman/Bahaya) pada *dashboard*, sistem melakukan pengolahan data mentah menggunakan logika komparasi terhadap nilai ambang batas (*threshold*) yang telah dikalibrasi pada sisi mikrokontroler Arduino Mega. Penentuan nilai *threshold* ini didasarkan pada karakteristik fisik sensor dan pengujian empiris sebagai berikut:

(a) **Deteksi Stabilitas Struktur (Sensor Getaran)**

Sensor MPU6050 mengukur percepatan total (a_{total}) yang mencakup gaya gravitasi bumi ($g \approx 9.8\ m/s^2$). Nilai ambang batas ditetapkan

sebesar 10.5 m/s^2 dengan logika penentuan status sebagaimana ditunjukkan pada Persamaan 3.1.

$$Status = \begin{cases} 1 \text{ (BAHAYA)} & \text{jika } a_{total} > 10.5\ m/s^2 \\ 0 \text{ (AMAN)} & \text{jika } a_{total} \leq 10.5\ m/s^2 \end{cases} \tag{3.1}$$

Nilai 10.5 dipilih untuk memberikan toleransi *noise* sensor sebesar $\pm 0.7\ m/s^2$ dari gaya gravitasi normal, sehingga getaran kecil tidak memicu alarm palsu pada *dashboard*.

(b) **Deteksi Kebakaran (Sensor Api)**

Sensor api berbasis spektrum inframerah/UV memiliki karakteristik *active-low* pada pembacaan ADC (0-1023), dimana intensitas cahaya api yang kuat akan menghasilkan nilai tegangan output yang rendah. Berdasarkan kalibrasi menggunakan sumber api standar pada jarak 50 cm, ditetapkan ambang batas ADC < 300 sebagai indikator kondisi kebakaran.

(c) **Deteksi Kebisingan Suara**

Ambang batas intensitas suara diatur secara perangkat keras (*hardware adjustment*) menggunakan potensiometer pada modul sensor LM393. Sensitivitas dikalibrasi untuk memicu sinyal digital `HIGH` hanya apabila intensitas suara melampaui 60 dB (setara dengan kebisingan percakapan keras), sehingga suara latar belakang (*background noise*) tidak dianggap sebagai gangguan.

Secara keseluruhan, alur pengolahan data dirancang untuk memastikan bahwa data lingkungan dan energi yang ditampilkan pada sistem BEMS bersifat akurat, stabil, dan dapat diandalkan untuk pemantauan kondisi gedung secara berkelanjutan.

3.4 Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa sistem BEMS yang dikembangkan mampu beroperasi sesuai dengan perancangan serta memenuhi kebutuhan pemantauan lingkungan dan energi secara *real-time*. Pengujian difokuskan pada aspek fungsionalitas dan performa sistem, mencakup perangkat keras, komunikasi data, serta aplikasi *web dashboard*.

Secara umum, pengujian sistem dibagi menjadi dua tahap utama, yaitu pengujian fungsional dan pengujian performa.

3.4.1 Pengujian Fungsional

Pengujian fungsional bertujuan untuk memverifikasi bahwa seluruh komponen sistem BEMS dapat berfungsi dengan baik dan saling terintegrasi. Pengujian ini dilakukan dengan mengamati aliran data secara menyeluruh mulai dari proses akuisisi data sensor hingga visualisasi data pada *dashboard*.

Pengujian dilakukan dengan konfigurasi sebagai berikut:

- Perangkat keras: Arduino Mega, ESP32, dan modul MCP2515.
- Sensor lingkungan: suhu, kelembapan, intensitas cahaya, gas, kebisingan, api, dan getaran.
- Jaringan: komunikasi lokal berbasis Wi-Fi untuk pengiriman data menggunakan protokol MQTT.
- Perangkat lunak: MQTT broker dan aplikasi *web dashboard*.

Tahapan pengujian fungsional dilakukan secara berurutan untuk memastikan setiap bagian sistem bekerja sesuai dengan fungsi yang dirancang:

1. Verifikasi pembacaan data sensor oleh Arduino Mega.
2. Pengujian pengiriman data sensor melalui jaringan CANBus.
3. Verifikasi penerimaan dan pemrosesan data oleh ESP32 sebagai *gateway*.
4. Pengujian pengiriman data dari ESP32 ke MQTT broker.
5. Validasi tampilan data pada *dashboard* secara *real-time*.
6. Verifikasi penyimpanan data ke dalam basis data.

Pengujian fungsional dinyatakan berhasil apabila seluruh data sensor dapat terbaca, dikirimkan, dan divisualisasikan secara konsisten tanpa terjadi kesalahan komunikasi maupun kehilangan data.

3.4.2 Pengujian Performa Sistem

Pengujian performa dilakukan untuk mengevaluasi kemampuan sistem BEMS dalam memproses, mentransmisikan, dan menampilkan data secara *real-time*. Pengujian ini bertujuan untuk menilai efisiensi sistem serta mengidentifikasi potensi hambatan yang dapat memengaruhi stabilitas dan responsivitas sistem.

Parameter performa yang diuji meliputi:

1. **Waktu Akuisisi Data**: waktu yang dibutuhkan sistem untuk membaca dan memproses data dari sensor lingkungan.
2. **Latensi Komunikasi**: waktu tunda pengiriman data dari Arduino Mega hingga data diterima dan ditampilkan pada *dashboard* melalui jaringan CANBus dan MQTT.
3. **Stabilitas Sistem**: konsistensi pengiriman dan penerimaan data selama periode pengujian.
4. **Performa *Dashboard***: kecepatan pembaruan data dan grafik pada antarmuka pengguna secara *real-time*.

3.4.3 Metrik dan Skenario Pengujian

Untuk memperoleh hasil evaluasi yang terukur dan objektif, pengujian performa sistem dilakukan menggunakan beberapa metrik kuantitatif. Seluruh pengujian dilakukan pada kondisi jaringan lokal yang stabil dengan interval pengambilan data sensor sebesar 1 detik atau setara dengan frekuensi 1 Hz.

Metrik performa yang digunakan pada penelitian ini dirangkum pada Tabel 3.2.

Tabel 3.2. Metrik Pengujian Performa Sistem

No	Parameter Uji	Satuan	Deskripsi
1	Waktu akuisisi data sensor	ms	Waktu pembacaan data sensor oleh Arduino Mega
2	Latensi CANBus	ms	Waktu pengiriman data dari Arduino Mega ke ESP32
3	Latensi MQTT	ms	Waktu pengiriman data dari ESP32 ke *dashboard*
4	Frekuensi pembaruan data	Hz	Jumlah pembaruan data pada *dashboard* per detik
5	Packet loss	%	Persentase data yang hilang selama transmisi

Setiap parameter diuji selama periode pengamatan tertentu untuk mengevaluasi kestabilan sistem dalam kondisi operasi berkelanjutan. Hasil

pengukuran dianalisis untuk menilai kelayakan arsitektur komunikasi CANBus–MQTT sebagai solusi pemantauan lingkungan dan energi yang stabil dan andal dalam sistem BEMS.