

## BAB 3

### METODOLOGI PENELITIAN

#### 3.1 Jenis dan Pendekatan Penelitian

Penelitian ini termasuk dalam kategori penelitian terapan (*applied research*) dengan jenis penelitian rekayasa sistem (*system development research*). Penelitian terapan dilakukan untuk menghasilkan solusi konkret terhadap masalah nyata melalui pengembangan sistem atau artefak teknologi yang dapat diimplementasikan langsung di lapangan (Kothari & Garg, 2019) [36].

Pendekatan yang digunakan adalah pendekatan rekayasa sistem, yang menekankan tahapan analisis kebutuhan, perancangan, implementasi, dan pengujian sistem. Pendekatan ini sesuai dengan tujuan penelitian, yaitu merancang dan mengembangkan sistem diagnostik kendaraan berbasis *Controller Area Network* (CAN Bus) serta melakukan translasi terhadap data yang didapatkan melalui CANBus.

Secara filosofis, penelitian ini berlandaskan pada paradigma postpositivistik, yang memandang realitas sebagai sesuatu yang dapat diamati dan diukur secara objektif. Paradigma ini menekankan pentingnya pengujian empiris terhadap hipotesis dan hasil implementasi (Creswell & Poth, 2018) [37]. Dalam konteks ini, pengujian dilakukan terhadap kinerja sistem dan akurasi model prediksi perawatan untuk memperoleh kesimpulan berbasis data.

Selain itu, pendekatan penelitian ini juga selaras dengan kerangka *Design Science Research* (DSR) yang banyak digunakan dalam penelitian teknologi dan sistem informasi modern (De Sordi, 2021) [38]. Dalam kerangka DSR, penelitian berfokus pada pengembangan artefak (sistem) serta evaluasi efektivitas dan keandalannya di konteks nyata.

Secara filosofis, penelitian ini berlandaskan pada paradigma postpositivistik. Paradigma ini dipilih karena memandang bahwa efektivitas sebuah artefak teknologi harus dibuktikan melalui observasi dan pengukuran yang ketat. Berbeda dengan positivisme murni yang mengklaim kebenaran mutlak, postpositivisme menekankan pada pengujian empiris untuk meminimalkan bias dan membuktikan reliabilitas sistem melalui serangkaian eksperimen terkontrol (Creswell & Creswell, 2018) [39]. Dalam penelitian ini, paradigma tersebut diimplementasikan melalui proses validasi akurasi pembacaan data CAN Bus dan evaluasi performa model

prediksi perawatan berdasarkan data numerik yang objektif.

Dengan demikian, penelitian ini menggabungkan tiga elemen utama: orientasi terapan, pendekatan rekayasa sistem, dan paradigma postpositivistik berbasis bukti, sehingga menghasilkan metodologi yang ilmiah, terukur, dan relevan dengan konteks pengembangan sistem diagnostik berbasis data.

### 3.2 Lokasi dan Waktu Penelitian

Penelitian ini dilakukan di Laboratorium Universitas Multimedia Nusantara, serta sebagian kegiatan dilakukan di kendaraan uji (*test vehicle*) untuk proses pengambilan data melalui jaringan *Controller Area Network* (CAN bus). Pemilihan lokasi didasarkan pada ketersediaan perangkat pendukung seperti mikrokontroler (ESP32), *CAN controller* (MCP2515), serta infrastruktur jaringan yang memungkinkan pengujian sistem diagnostik secara langsung. Selain itu, proses pengembangan sistem perangkat lunak dilakukan pada lingkungan *server backend* dan *frontend* berbasis web yang dijalankan secara lokal untuk memantau performa sistem secara *real-time*.

Penelitian ini dilaksanakan dalam rentang waktu bulan Juli hingga Desember 2025. Secara umum, tahapan penelitian dibagi menjadi fase-fase berikut:

1. Juli 2025: Perencanaan penelitian dan studi literatur.
2. Agustus 2025: Perancangan dan penyusunan sistem perangkat keras dan perangkat lunak.
3. September 2025: Pengumpulan dan translasi data CAN bus melalui *port* OBD-II pada kendaraan uji.
4. Oktober - November 2025: Implementasi dan integrasi perangkat keras dengan sistem server dan dashboard visualisasi.
5. Desember 2025: Evaluasi dan optimasi sistem.

### 3.3 Prosedur Penelitian

Prosedur penelitian menjelaskan tahapan-tahapan yang dilakukan dalam pelaksanaan penelitian, mulai dari tahap perencanaan hingga tahap evaluasi hasil. Prosedur ini disusun agar proses penelitian berjalan secara sistematis, terarah, dan sesuai dengan tujuan penelitian yang telah ditetapkan.

Secara umum, prosedur penelitian ini terdiri atas enam tahapan utama sebagai berikut:

1. Studi Literatur

Pada tahap ini dilakukan pengumpulan referensi yang relevan terkait dengan sistem komunikasi kendaraan menggunakan *Controller Area Network* (CAN Bus) dan sistem diagnostik kendaraan. Studi literatur ini bertujuan untuk memperoleh dasar teori dan identifikasi penelitian terdahulu yang menjadi acuan dalam perancangan sistem.

2. Analisis Kebutuhan Sistem

Tahap ini bertujuan untuk menentukan kebutuhan sistem yang akan dikembangkan, baik dari sisi perangkat keras maupun perangkat lunak. Analisis dilakukan terhadap parameter-parameter kendaraan yang akan diambil melalui CAN Bus, jenis mikrokontroler yang digunakan, serta kebutuhan penyimpanan dan visualisasi data.

3. Perancangan Sistem

Berdasarkan hasil analisis kebutuhan, dilakukan perancangan sistem yang meliputi perancangan arsitektur umum, perancangan perangkat keras, dan perancangan perangkat lunak. Hasil perancangan ini menjadi dasar dalam implementasi sistem diagnostik berbasis CAN Bus.

4. Implementasi dan Integrasi Sistem

Tahap ini melibatkan proses perakitan dan pengujian awal perangkat keras, serta pengembangan perangkat lunak untuk komunikasi data antara mikrokontroler dan server. Selain itu, dilakukan integrasi antara sistem pengambilan data, penyimpanan data, dan *dashboard* visualisasi berbasis web.

5. Pengumpulan dan Pengolahan Data

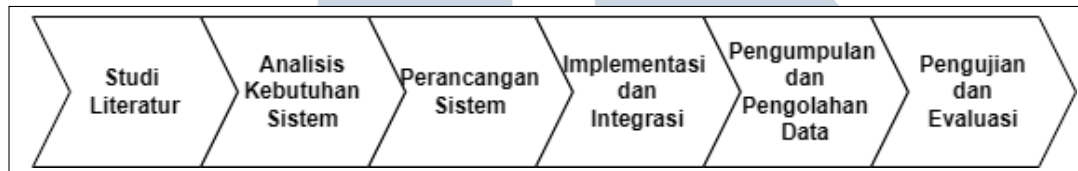
Setelah sistem berjalan, dilakukan pengumpulan data CAN Bus dari kendaraan uji melalui antarmuka OBD-II. Data yang dikumpulkan disimpan dalam format CSV untuk kemudian diolah dan dianalisis dengan tujuan menghasilkan formula translasi atau penerjemahan data CANBus.

6. Pengujian dan Evaluasi Sistem

Tahap akhir meliputi pengujian fungsional dan pengujian performa sistem

untuk memastikan bahwa sistem bekerja sesuai dengan spesifikasi yang telah dirancang.

Setiap tahapan pada prosedur penelitian ini dilakukan secara berurutan dan saling berkaitan, sebagaimana ditunjukkan pada Gambar 3.1.

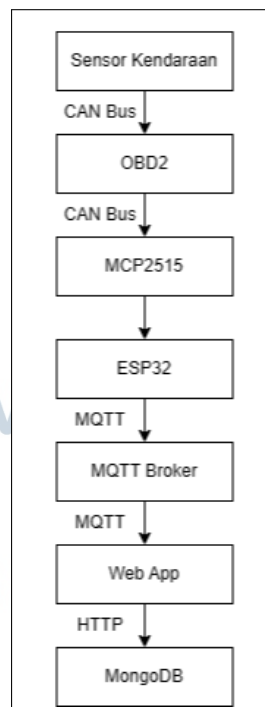


Gambar 3.1. Diagram alur penelitian

### 3.4 Perancangan Sistem

#### 3.4.1 Arsitektur Umum Sistem

Arsitektur umum sistem pada penelitian ini dirancang untuk memungkinkan proses akuisisi, pengiriman, penyimpanan, serta analisis data kendaraan secara otomatis dan terintegrasi. Sistem ini menghubungkan perangkat keras kendaraan dengan platform perangkat lunak berbasis web dan model *machine learning* untuk mendukung proses diagnostik dan prediksi perawatan kendaraan.



Gambar 3.2. Diagram arsitektur umum sistem

Sebagaimana ditunjukkan pada Gambar 3.2, data diperoleh dari sensor kendaraan yang terhubung melalui jaringan *Controller Area Network* (CAN Bus). Data CAN dikirim melalui port *On-Board Diagnostics II* (OBD-II) menuju modul *CAN Controller* (MCP2515) yang berfungsi sebagai antarmuka komunikasi antara jaringan CAN dengan mikrokontroler ESP32.

*Microcontroller* ESP32 berperan sebagai *data acquisition node* yang membaca frame data dari MCP2515, kemudian mengirimkan data tersebut ke jaringan melalui protokol *Message Queuing Telemetry Transport* (MQTT). Protokol MQTT dipilih karena sifatnya yang ringan dan sesuai untuk komunikasi *Internet of Things* (IoT) [40, 9].

Data yang dikirim oleh ESP32 diterima oleh *MQTT Broker*, yang bertugas mengelola dan mendistribusikan data ke aplikasi web (*Web App*). Aplikasi web berfungsi sebagai antarmuka pengguna untuk menampilkan data kendaraan secara *real-time*, melakukan visualisasi grafik, serta menyimpan data ke dalam *database MongoDB* melalui protokol HTTP.

Secara keseluruhan, arsitektur sistem ini terdiri dari tiga lapisan utama:

1. Lapisan Akuisisi Data: Terdiri atas sensor kendaraan, jalur CAN Bus, modul MCP2515, dan mikrokontroler ESP32 yang bertugas membaca serta mengirimkan data kendaraan.
2. Lapisan Komunikasi dan Integrasi: Mencakup protokol MQTT dan *MQTT Broker* yang berfungsi sebagai penghubung antara perangkat keras dan sistem berbasis web.
3. Lapisan Aplikasi dan Analisis: Terdiri dari aplikasi web dan *database MongoDB*.

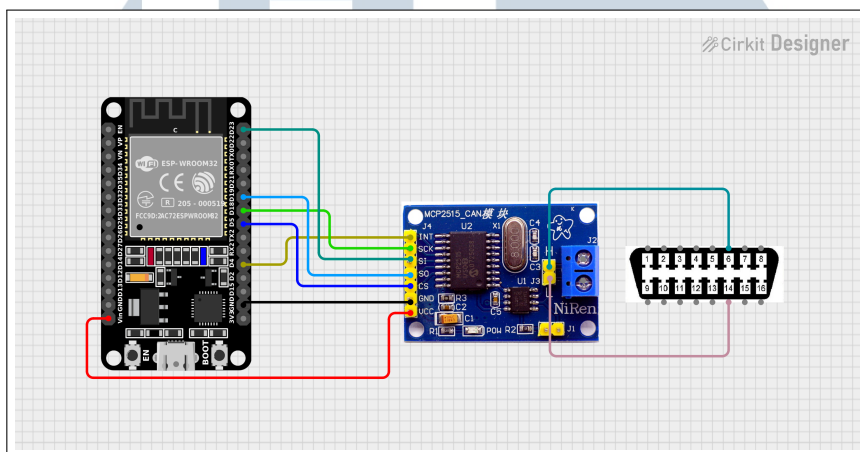
Arsitektur ini dirancang agar sistem dapat beroperasi secara modular, sehingga setiap komponen baik perangkat keras maupun perangkat lunak dapat dikembangkan atau diubah tanpa mempengaruhi keseluruhan sistem.

### **3.4.2 Perancangan Perangkat Keras**

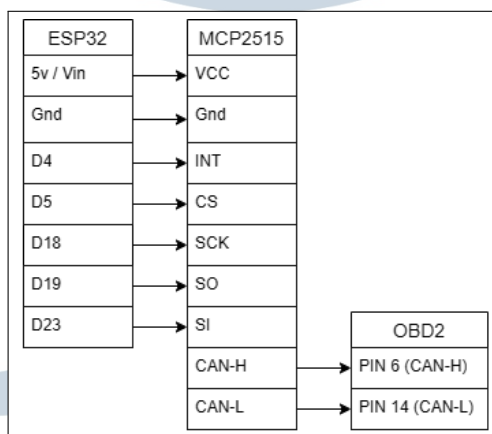
Perancangan perangkat keras pada penelitian ini bertujuan untuk membangun sistem akuisisi data yang mampu membaca komunikasi *Controller Area Network* (CAN Bus) dari kendaraan dan mengirimkan data tersebut ke sistem server melalui jaringan nirkabel. Perangkat keras utama yang digunakan terdiri

dari modul *microcontroller* ESP32, modul *CAN Controller* MCP2515, dan konektor OBD-II sebagai antarmuka ke jaringan kendaraan.

Gambar 3.3 menunjukkan rancangan fisik sambungan antar komponen perangkat keras, sedangkan Gambar 3.4 memperlihatkan diagram koneksi antar pin secara logis.



Gambar 3.3. Rancangan fisik koneksi perangkat keras ESP32, MCP2515, dan OBD-II



Gambar 3.4. Diagram koneksi pin antara ESP32, MCP2515, dan OBD-II

Modul ESP32 berfungsi sebagai pusat pemrosesan utama dan pengendali komunikasi data. ESP32 dipilih karena memiliki kemampuan komunikasi Wi-Fi serta antarmuka SPI yang kompatibel dengan modul MCP2515. Modul MCP2515 berperan sebagai pengendali komunikasi CAN yang menerjemahkan sinyal dari jaringan CAN kendaraan agar dapat dibaca oleh mikrokontroler.

Koneksi antara ESP32 dan MCP2515 dilakukan menggunakan antarmuka SPI dengan rincian pin seperti pada Tabel 3.1.

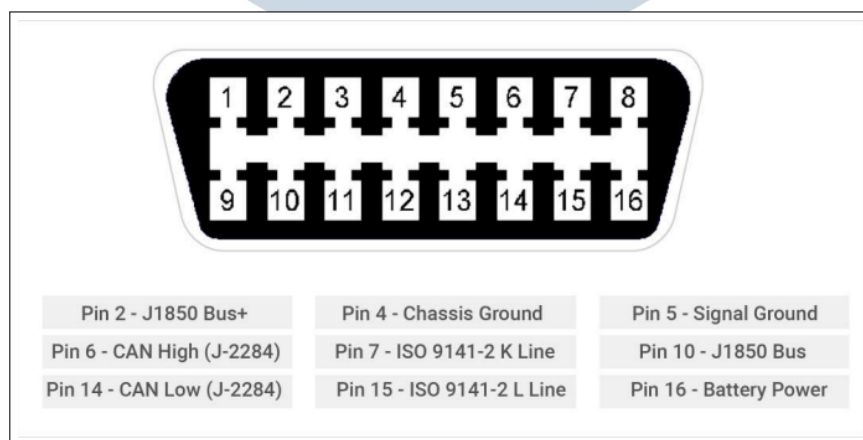


Tabel 3.1. Koneksi pin antara ESP32 dan MCP2515

| ESP32    | MCP2515 |
|----------|---------|
| 5V / Vin | VCC     |
| GND      | GND     |
| D4       | INT     |
| D5       | CS      |
| D18      | SCK     |
| D19      | SO      |
| D23      | SI      |

Modul MCP2515 kemudian dihubungkan ke port OBD-II kendaraan seperti pada Gambar 3.5 [1] menggunakan dua jalur utama, yaitu:

1. CAN-H (Pin 6): Jalur sinyal diferensial positif dari jaringan CAN.
2. CAN-L (Pin 14): Jalur sinyal diferensial negatif dari jaringan CAN.



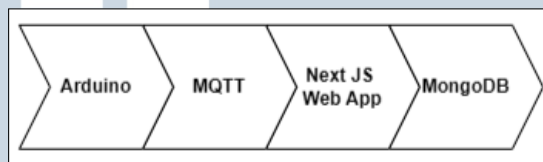
Gambar 3.5. Diagram konektor OBD-II [1]

Kedua jalur tersebut digunakan untuk membaca sinyal komunikasi antar modul elektronik kendaraan yang dikirim melalui jaringan CAN Bus. Hasil pembacaan tersebut kemudian diteruskan oleh ESP32 melalui protokol MQTT untuk diproses pada lapisan aplikasi dan analisis data. Dengan rancangan ini, sistem mampu melakukan pengambilan data kendaraan secara langsung.

### 3.4.3 Perancangan Perangkat Lunak

Perancangan perangkat lunak pada penelitian ini berfokus pada pengembangan sistem yang mampu menerima, menyimpan, memvisualisasikan, dan menganalisis data hasil pembacaan jaringan *Controller Area Network* (CAN Bus). Sistem perangkat lunak dirancang agar dapat bekerja secara terintegrasi dengan perangkat keras melalui protokol komunikasi *Message Queuing Telemetry Transport* (MQTT), serta mendukung analisis berbasis *machine learning*.

Gambar 3.6 menunjukkan alur umum arsitektur perangkat lunak yang dikembangkan dalam penelitian ini.



Gambar 3.6. Diagram alur perancangan perangkat lunak sistem diagnostik kendaraan

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

1. Arduino/ESP32

Bertugas membaca data dari jaringan CAN Bus melalui modul MCP2515, kemudian mengirimkan data tersebut dalam format JSON melalui protokol MQTT ke broker. Mikrokontroler ini juga dapat melakukan *preprocessing* sederhana seperti konversi nilai atau penandaan waktu sebelum pengiriman data.

2. MQTT Broker

Berfungsi sebagai perantara komunikasi antara mikrokontroler dan aplikasi web. Broker menerima pesan dari ESP32 dan mendistribusikannya ke klien lain (misalnya server atau aplikasi monitoring) yang berlangganan pada topik tertentu. Protokol MQTT dipilih karena sesuai untuk komunikasi data *Internet of Things (IoT)* [40].

3. Aplikasi Web (Next.js)

Aplikasi berbasis *framework* Next.js dikembangkan sebagai antarmuka utama sistem. Aplikasi ini menampilkan data kendaraan secara *real-time*, menampilkan grafik historis, serta menyediakan halaman untuk analisis hasil



model *machine learning*. Data yang diterima melalui MQTT diproses dan dikirim ke basis data menggunakan protokol HTTP.

#### 4. Basis Data (MongoDB)

Data yang dikirim dari aplikasi web disimpan dalam basis data non-relasional MongoDB. Struktur data yang fleksibel dari MongoDB memungkinkan penyimpanan data sensor dengan format dinamis dan waktu akses yang cepat.

Aliran data dalam sistem dimulai dari pembacaan sinyal CAN oleh mikrokontroler, dilanjutkan dengan pengiriman data melalui MQTT ke aplikasi web, hingga penyimpanan data di MongoDB. Dengan rancangan ini, sistem perangkat lunak mampu melakukan pemantauan kondisi kendaraan secara *real-time* dan menyediakan data historis untuk analisis

### 3.5 Pengumpulan Data

Pengumpulan data pada penelitian ini dilakukan secara langsung pada kendaraan uji untuk memperoleh data komunikasi dari sistem kendaraan melalui jalur *Controller Area Network* (CAN Bus). Kendaraan uji yang digunakan adalah mobil berjenis Honda BR-V tahun 2017. Pemilihan kendaraan ini didasarkan pada ketersediaannya untuk digunakan secara rutin selama proses penelitian, serta kesesuaiannya dengan kebutuhan pengambilan data pada sistem komunikasi kendaraan modern.

Tahap pengumpulan data awal difokuskan pada proses identifikasi dan *mapping* data CAN Bus terhadap parameter-parameter kendaraan yang relevan, seperti putaran mesin (RPM), kecepatan kendaraan, suhu mesin, dan parameter lainnya. Data diambil melalui antarmuka *On-Board Diagnostics II* (OBD-II) yang terhubung dengan jaringan CAN kendaraan.

Proses akuisisi data dilakukan menggunakan dua konfigurasi perangkat utama, yaitu:

1. *Microcontroller* Arduino Mega 2560 yang dipasangkan dengan modul *CAN Controller* MCP2515 atau TJA1050.
2. *Microcontroller* ESP32 yang dipasangkan dengan modul *CAN Controller* MCP2515.

Kedua konfigurasi tersebut dihubungkan ke komputer melalui antarmuka USB untuk keperluan perekaman data. Data hasil pembacaan dari mikrokontroler

direkam menggunakan *script* Python yang membaca keluaran serial, kemudian menyimpannya dalam format CSV untuk proses pengolahan dan analisis lebih lanjut.

### 3.6 Pengolahan dan Analisis Data

#### 3.6.1 Pengolahan Data

Pengolahan data pada penelitian ini dilakukan melalui beberapa tahapan bertahap yang mencakup proses pemrosesan data mentah dari jaringan *Controller Area Network* (CAN Bus), penyaringan data relevan, konversi nilai, hingga persiapan data untuk kebutuhan pelatihan model *machine learning*. Tahapan pengolahan data dapat dibedakan menjadi tiga bagian utama, yaitu pengolahan data awal, pengolahan data pada sistem *embedded*, dan pengolahan data untuk analisis prediktif.

##### 1. Pengolahan Data Awal (Offline Processing)

Pada tahap ini, data hasil perekaman komunikasi CAN Bus kendaraan diperoleh melalui antarmuka OBD-II dan disimpan dalam format *Comma-Separated Values* (CSV). Proses perekaman dilakukan menggunakan *script* Python yang membaca data dari port serial mikrokontroler.

Data yang telah direkam kemudian diolah menggunakan *Jupyter Notebook* untuk mengidentifikasi *CAN Identifier* (CAN ID) yang relevan dengan parameter kendaraan, seperti kecepatan, putaran mesin (RPM), dan suhu mesin. Proses identifikasi dilakukan melalui visualisasi data dalam bentuk grafik untuk mengamati pola perubahan nilai, serta dengan mencocokkan hasilnya terhadap perilaku pengemudi yang direkam secara manual.

Setelah CAN ID yang relevan berhasil diidentifikasi, data tersebut dianalisis untuk menentukan rumus atau faktor konversi yang dapat mengubah nilai mentah dari frame CAN menjadi nilai parameter yang bermakna. Hasil pengolahan ini menjadi dasar bagi proses perhitungan di tahap berikutnya.

##### 2. Pengolahan Data *Live* pada Sistem Embedded

Tahap ini dilakukan secara *real-time* pada perangkat ESP32 yang telah diintegrasikan dengan modul MCP2515. ESP32 menyaring data CAN berdasarkan daftar CAN ID relevan yang telah diidentifikasi sebelumnya. Nilai dari masing-masing CAN ID dikalkulasikan menggunakan formula

konversi hasil tahap pertama untuk menghasilkan parameter kendaraan yang dapat dibaca oleh pengguna (misalnya, RPM atau kecepatan).

Data yang telah diolah kemudian dikirim ke aplikasi web melalui protokol MQTT dalam format JSON. Aplikasi web menampilkan data tersebut secara *live* melalui *dashboard* visualisasi.

Secara keseluruhan, proses pengolahan data pada penelitian ini mencakup alur penuh mulai dari pengambilan data mentah, penyaringan, konversi nilai, dan penyimpanan. Alur pengolahan ini memastikan bahwa data yang dihasilkan memiliki kualitas dan konsistensi yang memadai untuk keperluan visualisasi dan diagnostik.

### **3.7 Pengujian Sistem**

Pengujian sistem bertujuan untuk memastikan bahwa sistem yang dikembangkan berfungsi sesuai dengan rancangan serta mampu beroperasi dalam kondisi nyata. Pengujian dilakukan untuk menilai baik aspek fungsionalitas maupun performa keseluruhan sistem, termasuk komponen *hardware*, komunikasi data, dan aplikasi web baik *frontend* maupun *backend*.

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

#### **3.7.1 Pengujian Fungsional**

Pengujian fungsional bertujuan untuk memastikan bahwa setiap komponen sistem bekerja sesuai dengan fungsi yang telah ditentukan dalam tahap perancangan. Pengujian dilakukan untuk memverifikasi jalur komunikasi data dari sumber hingga visualisasi, serta memastikan integrasi antar modul berjalan dengan baik.

Pengujian dilakukan secara langsung di lapangan dengan kondisi sebagai berikut:

1. Kendaraan uji: Honda BR-V 2017 yang terhubung ke sistem melalui port OBD-II.
2. Perangkat keras: ESP32 dan modul MCP2515 yang berfungsi membaca data CAN Bus.

3. Jaringan: seluruh perangkat (ESP32, *MQTT broker*, dan *web app*) berada dalam jaringan Wi-Fi lokal yang sama.
4. Perangkat lunak: aplikasi *web app* berjalan di laptop sebagai server lokal.

Prosedur pengujian dilakukan dari hulu ke hilir untuk memastikan bahwa seluruh rantai data berfungsi sebagaimana mestinya:

1. Pengujian konektivitas perangkat keras dengan kendaraan melalui port OBD-II.
2. Verifikasi pembacaan data CAN Bus oleh ESP32.
3. Pengujian pengiriman data dari ESP32 ke broker MQTT.
4. Pengujian penerimaan data oleh aplikasi *web app*.
5. Validasi tampilan data di *dashboard* sesuai dengan nilai aktual kendaraan.
6. Verifikasi pengiriman data ke basis data (MongoDB).

Kriteria keberhasilan pengujian fungsional adalah apabila seluruh data dari kendaraan dapat terbaca, diteruskan, dan divisualisasikan tanpa kesalahan komunikasi atau kehilangan data.

### 3.7.2 Pengujian Performa Sistem

Pengujian performa dilakukan untuk mengevaluasi kinerja sistem dalam memproses, mentransmisikan, dan menampilkan data secara *real-time*. Pengujian ini juga digunakan untuk mengidentifikasi potensi *bottleneck* yang dapat memengaruhi kecepatan atau stabilitas sistem.

Aspek yang diuji pada tahap ini meliputi:

1. Tingkat *Throughput* Data: Mengukur kecepatan perekaman dokumen telemetri ke dalam basis data per unit waktu.
2. Efisiensi Transmisi dan *Loss Rate*: Menganalisis stabilitas pengiriman paket data melalui protokol MQTT dari mikrokontroler ke *backend*. Pengujian ini bertujuan untuk menghitung persentase kehilangan data selama periode operasional tertentu.

3. Efisiensi Penyimpanan Data: Mengevaluasi beban penyimpanan pada database MongoDB dengan menghitung volume data (dalam MB) dan ukuran indeks yang dihasilkan dari puluhan ribu dokumen telemetry berformat JSON.
4. Validasi Akurasi Penerjemahan (*Decoding*): Melakukan uji silang (cross-validation) antara nilai parameter fisik yang dihasilkan oleh formula bitwise sistem (seperti kecepatan, RPM, posisi gigi, dan tekanan pedal) dengan indikator standar pada dasbor kendaraan serta data referensi dari aplikasi GPS pihak ketiga.
5. Kontinuitas Estimasi Odometer: Menguji mekanisme *handshake* dalam menjaga sinkronisasi nilai jarak tempuh antara mikrokontroler dan basis data agar perhitungan odometer tetap berlanjut dan persisten setelah proses *restart* perangkat.

Seluruh pengujian dilakukan dengan kondisi jaringan lokal yang stabil dan spesifikasi perangkat keras yang seragam..

