# Rancang Bangun Sistem Aplikasi Deteksi Jatuh Pada Lanjut Usia (Lansia) Berbasis *Internet of Things* (IoT)

1st Adinda Zahra Febriani
*School of Electrical Engineering*
*Telkom University*
Bandung, Indonesia
adindazahra@student.telkomuniversity.ac.id

2nd Ainul Mardiyah
*School of Electrical Engineering*
*Telkom University*
Bandung, Indonesia
ainulrtg@student.telkomuniversity.ac.id

3rd Salma Pebrina Firdatunnisa
*School of Electrical Engineering*
*Telkom University*
Bandung, Indonesia
salmapebrina@student.telkomuniversity.ac.id

4th Zulfa Nurmala
*School of Electrical Engineering*
*Telkom University*
Bandung, Indonesia
zulfanurmala@student.telkomuniversity.ac.id

5th Sofia Naning Hertiana
*School of Electrical Engineering*
*Telkom University*
Bandung, Indonesia
sofiananing@student.telkomuniversity.ac.id

6th Inung Wijayanto
*School of Electrical Engineering*
*Telkom University*
Bandung, Indonesia
iwijayanto@telkomuniversity.ac.id

***Abstrak* — Lansia memiliki risiko tinggi mengalami jatuh akibat penurunan fungsi motorik dan keseimbangan tubuh. Kejadian jatuh yang tidak segera ditangani dapat berdampak fatal, terutama bagi lansia dengan kondisi medis tertentu seperti penderita penyakit stroke. Untuk mengatasi masalah tersebut, dirancang sistem deteksi jatuh berbasis *Internet of Things* (IoT) yang terintegrasi dengan aplikasi *mobile*. Sistem ini memanfaatkan sensor MPU6050 dan mikrokontroler ESP32-C6 untuk memantau gerakan tubuh secara *real-time*. Metode yang digunakan melibatkan pemrosesan data akselerometer dan *gyroscope* melalui algoritma *threshold* untuk mengklasifikasikan tiga kondisi: diam, berjalan, dan jatuh. Data dikirim secara nirkabel ke *Firebase Realtime Database* dan ditampilkan dalam aplikasi *mobile*, yang akan menampilkan status dan mengirimkan notifikasi jika terdeteksi jatuh. Berdasarkan hasil pengujian, sistem menunjukkan tingkat akurasi sebesar 92% untuk pengujian inter-subjek, dengan rata-rata *delay* dari perangkat ke firebase di bawah 5 detik. Selain itu, evaluasi kualitas aplikasi melalui survei menunjukkan respons positif dari pengguna dengan menghasilkan skor *Mean Opinion Score* (MOS) sebesar 4.3 dari 5. Diskusi hasil menunjukkan bahwa sistem masih perlu pengembangan lebih lanjut untuk mendeteksi kejadian jatuh pada lansia, karena masih terdapat sedikit gap terhadap spesifikasi ideal (95% akurasi). Potensi pengembangan selanjutnya adalah integrasi *machine learning* untuk meningkatkan akurasi klasifikasi aktivitas dan pengurangan *false positive*.**

***Kata kunci*— lansia, deteksi jatuh, IoT, ESP32-C6, MPU6050, akurasi, *mobile app*.**

## I. PENDAHULUAN

Di Indonesia, menurut laporan dari *World Health Organization* (WHO) pada tahun 2023, sekitar 28-35% dari populasi yang berusia 65 tahun ke atas mengalami kejadian jatuh setiap tahunnya. Angka ini meningkat menjadi 32-42% untuk lansia yang berusia di atas 70 tahun [1]. Pada lansia yang mengidap stroke, adanya *golden time* atau periode emas sangat krusial. *Golden time* merujuk pada waktu kritis segera setelah terjadinya stroke atau cedera serius, di mana intervensi medis cepat dapat sangat mempengaruhi hasil pemulihan. *Golden time* untuk penanganan stroke adalah sekitar 3 hingga 5 jam setelah gejala pertama kali muncul [2]. Selain itu, lansia yang mengalami jatuh dan tidak segera mendapatkan perawatan medis memiliki risiko cedera dan kematian yang lebih tinggi. Waktu yang kritis antara terjadinya jatuh dan mendapatkan bantuan medis adalah faktor penentu utama dalam tingkat kelangsungan hidup dan kualitas pemulihan [3].

Dalam era digital saat ini, teknologi *Internet of Things* (IoT) telah menawarkan beragam peluang untuk meningkatkan kualitas hidup, terutama bagi lansia. Sistem ini diharapkan dapat memberikan respons cepat dalam situasi darurat, sehingga mengurangi waktu yang dibutuhkan untuk memberikan pertolongan. Dengan adanya teknologi *Internet of Things* (IoT), sistem deteksi jatuh dapat dirancang untuk memantau kondisi lansia secara *real-time* dan mengirimkan notifikasi kepada pengasuh atau keluarga melalui aplikasi [4]. Hal ini tidak hanya meningkatkan keselamatan lansia, tetapi juga memberikan ketenangan pikiran dan rasa aman bagi keluarga dan pengasuh. Namun, teknologi yang telah dikembangkan sebelumnya masih memiliki sejumlah keterbatasan, seperti desain perangkat yang kurang fleksibel dan tidak cukup ringan, sehingga terasa tidak nyaman saat digunakan oleh lansia dalam aktivitas sehari-hari.

Untuk mengatasi keterbatasan tersebut, kami merancang sistem deteksi jatuh berbasis *Internet of Things* (IoT) yang mempertimbangkan kenyamanan dan keamanan bagi lansia. Sistem ini menggunakan sensor percepatan dan giroskop untuk mendeteksi perubahan gerakan yang signifikan.

Permasalahan yang mendasari penelitian ini adalah bagaimana merancang perangkat deteksi jatuh yang ringan, nyaman, dan tidak mengganggu aktivitas pengguna, serta bagaimana membangun sistem deteksi *real-time* yang terintegrasi dengan aplikasi *mobile* agar dapat mengirimkan

respons kondisi darurat secara cepat dan efisien.

Berdasarkan permasalahan tersebut, tujuan dari penelitian ini adalah untuk merancang dan mengimplementasikan perangkat deteksi jatuh bagi lansia yang ringan, nyaman, dan mudah digunakan dalam aktivitas sehari-hari, serta membangun sistem pendukung berbasis aplikasi *mobile* yang mampu merespons kondisi darurat secara *real-time*.

## II. KAJIAN TEORI

Lanjut usia (lansia) adalah kelompok usia ≥60 tahun yang rentan mengalami penurunan fungsi tubuh. Salah satu risiko serius adalah kejadian jatuh, yang dapat menyebabkan cedera berat hingga kematian [5]. Di Amerika Serikat, angka kematian akibat jatuh pada lansia mencapai 78 per 100.000 jiwa pada tahun 2021, meningkat 41% dalam 9 tahun terakhir [6]. Hal ini menunjukkan bahwa jatuh pada lansia merupakan isu global.

### A. Studi Terkait Sistem Deteksi Jatuh

Penelitian deteksi jatuh telah menggunakan berbagai kombinasi sensor dan platform. Salah satunya menggunakan MPU6050 dan NodeMCU ESP8266, dengan notifikasi via Telegram bot dan waktu respons 0,21 detik [7]. Studi lain memanfaatkan MPU6050 dan GPS Neo 6M dengan notifikasi melalui aplikasi Flutter dan Firebase, rata-rata *delay* 4,71 detik [3]. Penelitian terbaru menggabungkan MPU6050 dan MAX30100 dengan aplikasi MIT Inventor, mampu mendeteksi jatuh dan memantau detak jantung serta $SpO_2$ dengan akurasi tinggi [4].

### B. Komponen Sistem

Sistem ini memanfaatkan sensor MPU6050 untuk mendeteksi orientasi gerakan, serta mikrokontroler Beetle ESP32-C6 untuk mengirimkan data secara *real-time* ke Firebase. Catu daya menggunakan baterai LiPo 1300 mAh. Perakitan dilakukan di perfboard untuk efisiensi ruang.

Pengembangan perangkat lunak dilakukan di Arduino IDE menggunakan bahasa C++, menerapkan logika *threshold*. Untuk notifikasi otomatis digunakan Node.js, dengan pengembangan *backend* di Visual Studio Code. Aplikasi Android dikembangkan menggunakan Kotlin di Android Studio, dan desain UI dibuat di Figma. Firebase digunakan sebagai platform *cloud* untuk penyimpanan dan pengiriman notifikasi (melalui FCM).

### C. Batasan dan Spesifikasi

Sistem deteksi jatuh yang dikembangkan memiliki batasan dan spesifikasi pada dua aspek utama, yaitu perangkat IoT dan aplikasi *mobile application*.

1. Pada perangkat IoT, sistem dilengkapi sensor untuk mendeteksi aktivitas diam, berjalan, dan jatuh dengan definisi spesifik mengenai perubahan posisi tubuh yang tidak terkendali diikuti dengan kontak kaki dan badan ke permukaan. Sistem ini harus memiliki akurasi minimal 95% dan presisi minimal 90%, sesuai standar alat kesehatan nasional dan internasional. Dari sisi ergonomi, perangkat dirancang ringan, nyaman, dan mudah digunakan oleh lansia. Selain itu, waktu tunda (*delay*) pengiriman data harus berada dalam batas yang memungkinkan respons cepat, terutama dalam situasi darurat. Penanganan medis idealnya dilakukan dalam *golden period* 3–5 jam pascakejadian. Untuk *delay* jaringan, standar TIPHON dan ITU-T menetapkan batas maksimal 150 ms agar komunikasi data tetap responsif dan tidak mengganggu kualitas layanan.
2. Pada aplikasi *mobile*, spesifikasi mencakup antarmuka yang ramah pengguna, mampu mengirimkan notifikasi *real-time* jika terdeteksi jatuh, serta menyimpan riwayat status pengguna. Aplikasi harus memiliki waktu muat (*load time*) cepat, dengan target kurang dari 2 detik, dan mendukung konektivitas stabil dengan perangkat IoT melalui jaringan internet. Semua komponen sistem ini dirancang untuk memastikan keandalan, kenyamanan, serta keamanan pengguna lanjut usia.

### D. Metode Uji Pengukuran Spesifikasi

Pengujian dilakukan untuk memastikan perangkat IoT dan aplikasi *mobile* bekerja sesuai spesifikasi. Aspek yang diuji meliputi deteksi gerakan, akurasi, presisi, kenyamanan alat, serta performa aplikasi. Berikut rincian pengukuran yang dilakukan:

1. Perangkat IoT

TABEL 1
Metode Pengukuran Perangkat IoT

| Parameter | Penjelasan |
|---|---|
| Perangkat IoT | Pengujian dilakukan dengan menjepitkan perangkat pada pakaian pengguna pada bagian tengah dan samping kanan kiri dada, lalu mensimulasikan gerakan jatuh ke depan, ke samping kanan/kiri, dan ke belakang. Selain itu, diuji juga kondisi lain seperti diam, berjalan, serta transisi posisi dari diam ke berjalan, berjalan ke jatuh, diam ke jatuh, dan sebaliknya. |
| Akurasi | Pengujian dilakukan sebanyak 20 kali untuk masing-masing kondisi oleh empat responden yang berbeda. Data dicatat untuk mengevaluasi akurasi dan presisi, menggunakan kategori:<br>- *True Positive* (TP): Sistem benar mendeteksi kejadian positif secara benar.<br>- *True Negative* (TN): Sistem benar tidak mendeteksi kejadian positif.<br>- *False Positive* (FP): Sistem salah mendeteksi kejadian positif.<br>- *False Negative* (FN): Sistem gagal mendeteksi kejadian positif yang sebenarnya terjadi. |
| Presisi | Pengujian dilakukan sebanyak 20 kali untuk tiga kondisi: diam, berjalan, dan jatuh dan melibatkan 4 responden.<br>Hasil diklasifikasikan menjadi:<br>- *True Positive* (TP): Sistem benar mendeteksi kejadian positif secara benar. |

| Parameter | Penjelasan |
|---|---|
| | - *True Negative* (TN): Sistem benar tidak mendeteksi kejadian positif.<br>- *False Positive* (FP): Sistem salah mendeteksi kejadian positif.<br>- *False Negative* (FN): Sistem gagal mendeteksi kejadian positif yang sebenarnya terjadi. |
| Ergonomis | Melakukan riset kepada responden di panti jompo dengan membawa contoh bentuk fisik yang mirip dengan alat deteksi jatuh yang sudah ada sebelumnya. |
| *Delay* | Data dikirim dari perangkat dan diterima di Firebase untuk mengukur selisih waktu pengiriman dan penerimaan. Perhitungan *delay* menggunakan rumus:<br>$Delay = Waktu\ Menerima - Waktu\ Mengirim$ (1)<br>Untuk mengukur *delay* jaringan, digunakan Wireshark dengan merekam lalu lintas paket saat pengguna dalam kondisi diam, berjalan, dan jatuh. *Delay* dihitung dari selisih waktu antar paket, lalu dirata-rata untuk mengetahui estimasi waktu tunda selama transmisi data. |

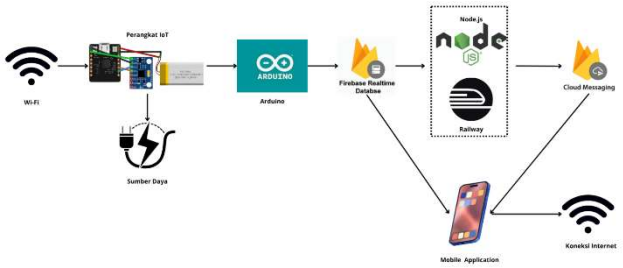
*2. Mobile Application*

TABEL 2
Metode Pengukuran *Mobile Application*

| Parameter | Penjelasan |
|---|---|
| Antarmuka Pengguna | Melakukan percobaan pada aplikasi secara eksternal melalui survei untuk memastikan aplikasi yang dirancang berfungsi dengan baik. |
| *Time to Load* | Mengukur durasi pemuatan halaman hingga seluruh data dan elemen antarmuka muncul secara lengkap dan dapat digunakan oleh pengguna. |
| Konektivitas dengan Perangkat IoT dan Sensor | Mengukur respons waktu dan stabilitas koneksi dengan berbagai perangkat dan sensor IoT. |

## III. METODE

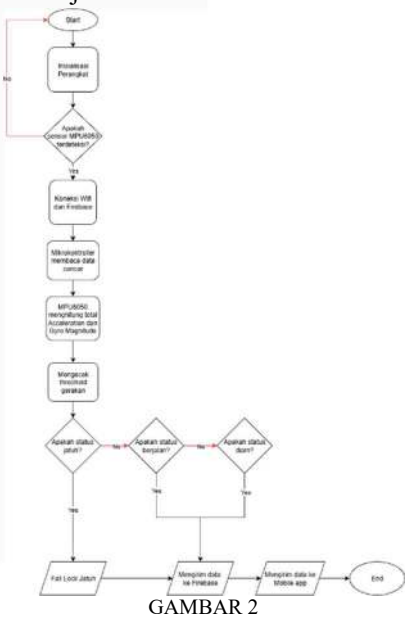
Penelitian ini menggunakan pendekatan rekayasa perangkat lunak untuk merancang sistem deteksi jatuh berbasis *Internet of Things* (IoT) yang nyaman dan responsif bagi lansia. Metode pelaksanaannya dibagi ke dalam dua komponen utama, yaitu implementasi perangkat keras (*hardware*) dan perangkat lunak (*software*). Gambar 1 berikut menunjukkan arsitektur umum sistem yang diimplementasikan:




GAMBAR 1
Arsitektur Sistem Deteksi Jatuh Berbasis IoT

Dari sisi *hardware*, sistem dikembangkan menggunakan Beetle ESP32-C6 sebagai mikrokontroler dan MPU6050 sebagai sensor utama. MPU6050 menggabungkan akselerometer dan *gyroscope* untuk memantau pergerakan tubuh lansia. Data gerakan diproses dengan algoritma *threshold* untuk menentukan status aktivitas pengguna berdasarkan nilai percepatan dan kecepatan sudut, lalu diteruskan secara *real-time* ke *Firebase Realtime Database* melalui koneksi Wi-Fi. Firebase menyimpan dan memperbarui data, serta memicu *backend* Node.js (di*hosting* di Railway) untuk mendeteksi kondisi jatuh. Jika terdeteksi, notifikasi dikirim ke aplikasi *mobile* melalui *Firebase Cloud Messaging* (FCM). Aplikasi kemudian menampilkan data aktivitas pengguna secara *real-time*.

### A. Cara Kerja Sistem




GAMBAR 2
Diagram Alir Sistem

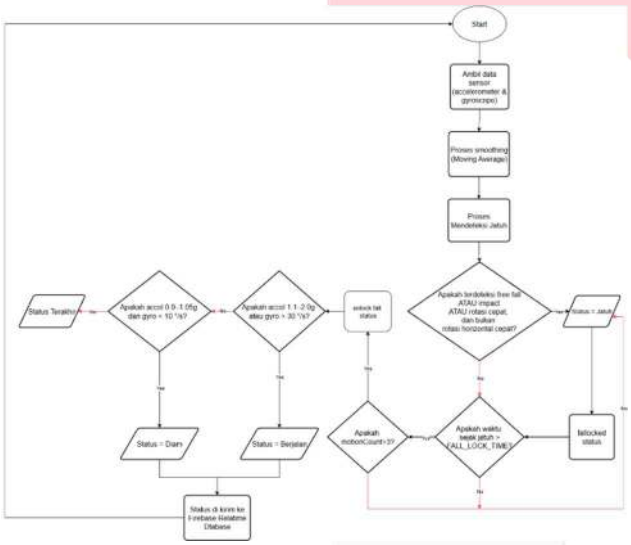
Gambar 2 menunjukkan alur kerja sistem deteksi jatuh yang menggunakan sensor MPU6050 dan mikrokontroler ESP32-C6. Proses dimulai dari inisialisasi perangkat dan pembacaan data akselerasi serta *gyro* dari sumbu x, y, dan z. Nilai ini dihitung menjadi total akselerasi dan kecepatan sudut, lalu difilter menggunakan *Moving Average Filter* untuk mengurangi *noise*. Data yang telah diproses dibandingkan dengan *threshold* untuk mengklasifikasikan status pengguna menjadi diam, berjalan, atau jatuh. Data kondisi dikirim ke *Firebase Realtime Database* secara *real-time*. Jika status berubah atau bertahan

selama 5 menit, sistem menyimpannya ke *history*. Jika terdeteksi jatuh dan tidak ada gerakan lanjutan, sistem mengaktifkan *fallLock* dan mengirimkan notifikasi ke aplikasi *mobile*. Nilai total akselerasi dihitung dengan rumus:

$$Total\ Akselerasi = \sqrt{(a_x^2 + a_y^2 + a_z^2)} \qquad (2)$$

ax, ay, az adalah nilai percepatan pada masing – masing sumbu dalam satuan ($m/s^2$). Hasil total akselerasi dibagi dengan 9,81 untuk mendapatkan nilai dalam satuan gravitasi (g). Selain data akselerasi, sistem juga membaca nilai kecepatan sudut dari *gyro*. Nilai sensor dikonversi dari radian per detik ($rad/s$) menjadi derajat per detik ($°/s$) menggunakan rumus:

$$Gyro(x, y, z) = rad/s \times \frac{180}{\pi} \qquad (3)$$



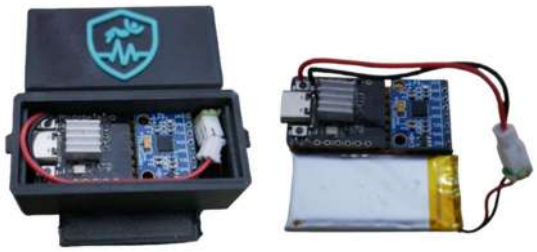

GAMBAR 3
Diagram Alir Penentuan *Threshold*

Pada Gambar 3 menunjukkan bahwa sistem menentukan status aktivitas pengguna (diam, berjalan, atau jatuh) berdasarkan evaluasi terhadap nilai akselerasi dan *gyro* yang telah diproses menggunakan *Moving Average Filter*. Logika deteksi mencakup pemeriksaan terhadap kondisi *freefall, impact*, dan rotasi tubuh. Jika salah satu terpenuhi, sistem akan mengunci status jatuh (*fallLock*) dan memeriksa pergerakan pascajatuh. Bila tidak ada gerakan signifikan, status jatuh tetap dikunci dan dikirimkan ke *Firebase Realtime Database*. Sementara itu, status diam atau berjalan ditentukan berdasarkan rentang *threshold* tertentu, sebagaimana dirangkum pada Tabel 3.

TABEL 3
Kriteria Penentuan Status Gerakan Berdasarkan *Threshold* Sensor

| Status | Total Percepatan (g) | Kecepatan Sudut (°/s) |
|---|---|---|
| Diam | 0,9 – 1,05 | < 10 |
| Berjalan | 1,1 – 2,0 | > 30 |
| Jatuh | < 0,4 (*free fall*) atau > 3,2 (*impact*) | > 150 |

B. Detail Implementasi

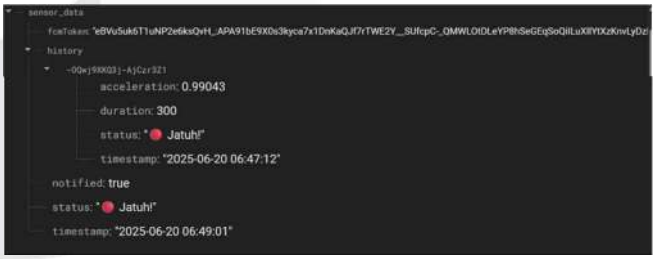


GAMBAR 4
Implementasi *Hardware*

Sistem deteksi jatuh lansia berbasis IoT terdiri dari sensor MPU6050, mikrokontroler Beetle ESP32-C6, dan baterai Lipo 1300 mAh. Semua komponen dirakit ke dalam *case box* berbahan PLA+ berukuran 53 × 21,7 × 21 mm untuk perlindungan fisik. Koneksi antar komponen menggunakan *flat cable* dengan konfigurasi pin sebagai berikut:

1. Pin 3V3 pada Beetle ESP32-C6 dihubungkan dengan VCC pada MPU6050 sebagai sumber daya.
2. Pin GND pada Beetle ESP32-C6 dihubungkan dengan GND pada MPU6050 untuk menyinkronkan referensi tegangan.
3. Pin 19 pada Beetle ESP32-C6 dihubungkan dengan SDA pada MPU6050 untuk pengiriman data.
4. Pin 20 pada Beetle ESP32-C6 dihubungkan dengan SCL pada MPU6050 untuk sinkronisasi pengiriman dan penerimaan data.

Pada sistem yang dikembangkan, Firebase digunakan sebagai media penyimpanan data dan pengirim notifikasi. Dua layanan utama dari Firebase yang dimanfaatkan yaitu *Realtime Database* dan *Firebase Cloud Messaging* (FCM).



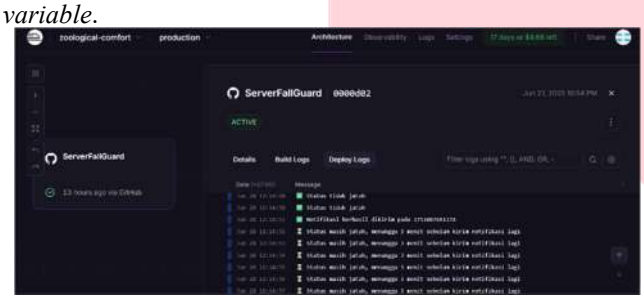

GAMBAR 5
Implementasi *Firebase Realtime Database*

Struktur *database* terdiri dari tiga bagian utama:

1. sensor_data
   Menyimpan status *real-time* pengguna (diam, berjalan, jatuh) beserta *timestamp* untuk memantau kondisi terkini.
2. fcmToken
   Menyimpan *token* FCM dari perangkat Android yang digunakan untuk mengirim *push notification* saat terdeteksi jatuh.
3. *history*
   Menyimpan riwayat status, baik saat status berubah

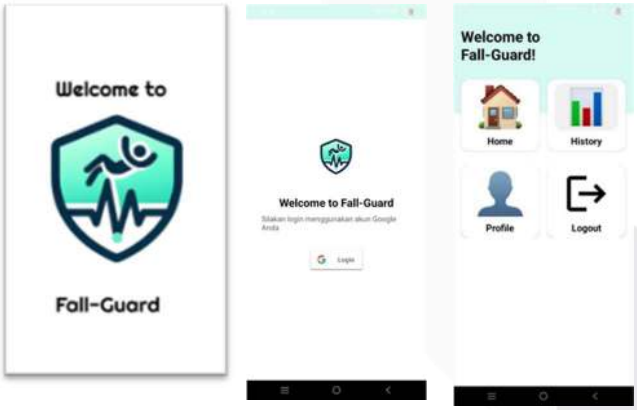
maupun saat status tetap selama 5 menit. Setiap entri memiliki ID unik (*timestamp*), status, waktu, dan status notifikasi.

Server *back-end* dikembangkan menggunakan Node.js untuk memantau status pengguna pada *Firebase Realtime Database* secara berkala dan mengirimkan notifikasi darurat jika terdeteksi kondisi jatuh. Dengan memanfaatkan *Firebase Admin SDK*, sistem membaca status terbaru setiap detik. Jika status "jatuh" terdeteksi dan memenuhi syarat (misalnya jatuh pertama atau sudah lebih dari 3 menit sejak notifikasi sebelumnya), maka sistem mengambil *FCM token* dan mengirim notifikasi ke perangkat Android melalui *Firebase Cloud Messaging*. Sistem ini juga menandai status *notified* di Firebase untuk menghindari pengiriman berulang. Seluruh layanan dijalankan secara *always-on* menggunakan Railway untuk memastikan monitoring berjalan 24/7, sekaligus mempermudah *deployment* dan pengelolaan *environment variable*.



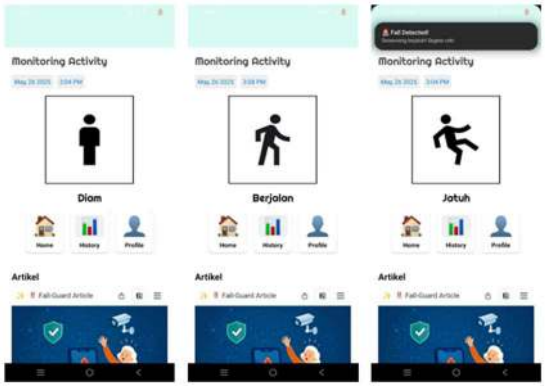

GAMBAR 6
Implementasi Node.js dengan Railway

Adapun aplikasi Fall-Guard memiliki beberapa menu utama yang memudahkan pengguna dalam memantau kondisi lansia, di antaranya:



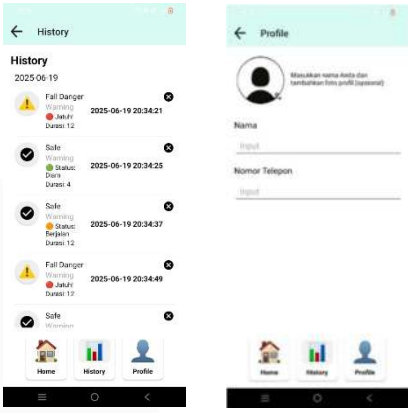

GAMBAR 7
Implementasi *Splash Screen, Login Page*, dan *Landing Page*

1. *Splash Screen* pada Gambar 7 bagian kiri menampilkan logo aplikasi Fall-Guard saat pertama kali membuka aplikasi.
2. *Login Page* pada Gambar 7 bagian tengah yang mengarahkan pengguna untuk masuk dengan menggunakan akun Google.
3. *Landing Page* pada Gambar 7 bagian kanan memuat empat menu penting:




GAMBAR 8
Implementasi Menu *Home*

a. *Home*: Menampilkan status beserta tanggal dan waktu kejadian, serta artikel-artikel relevan. Menampilkan tiga kondisi, yaitu: diam, berjalan, dan jatuh. Pada saat terdeteksi kondisi jatuh maka akan muncul *pop up* notifikasi seperti pada Gambar 8 bagian kanan.




GAMBAR 9
Implementasi Menu *History* dan *Profile*

b. *History*: Tampilan *History* pada Gambar 9 bagian kiri, menyajikan riwayat perubahan kondisi lansia, disertai dengan tanggal, waktu, dan durasi masing-masing kejadian serta pengguna dapat menghapus riwayat tersebut.
c. *Profile*: Pada Gambar 9 bagian kanan terdapat menu *Profile* yang menyimpan informasi pribadi seperti nama dan nomor telepon pihak yang bersangkutan serta foto profil yang bisa diisi oleh pihak terkait.

## IV. HASIL DAN PEMBAHASAN

Pengujian sistem dilakukan untuk menilai performa alat deteksi jatuh lansia berbasis IoT, yang mengirimkan data ke Firebase dan memberikan notifikasi ke aplikasi. Pengujian mencakup tiga aspek utama berikut:

1. Pengujian *Hardware*
   Bertujuan memastikan seluruh komponen utama seperti sensor MPU6050 dan mikrokontroler ESP32-C6 berfungsi stabil dan terintegrasi. Pengujian meliputi pengecekan koneksi, pembacaan sensor gerak, kestabilan daya, serta konektivitas Wi-Fi untuk pengiriman data ke Firebase.

2. Pengujian Sistem Deteksi Jatuh
   Dilakukan pada empat subjek dengan posisi alat berbeda (tengah, kiri, kanan), dalam tiga kondisi utama: diam, berjalan, dan jatuh. Data sensor diamati untuk memetakan pola gerakan, lalu dievaluasi akurasi sistem melalui *confusion matrix* (akurasi, presisi, *recall*, dan *specificity*). Selain itu, diuji pula masa hidup baterai dan *delay* pengiriman data ke Firebase.
3. Pengujian Aplikasi *Mobile* & *User Experience*
   Menilai waktu respons UI dan kecepatan notifikasi setelah kejadian jatuh. Evaluasi juga dilakukan melalui survei pengguna untuk menilai aspek kenyamanan, keterbacaan, navigasi menu, dan keakuratan notifikasi. Hasil pengujian ini menggabungkan pendekatan teknis (Firebase + *Logcat*) dan subjektif (kuesioner).

TABEL 4
Hasil Pengujian *Hardware*

| Tahapan | Deskripsi Pengujian | Keterangan |
|---|---|---|
| Pemeriksaan Koneksi | Memeriksa semua komponen elektronik terpasang sesuai dengan skema rangkaian. | Berhasil |
| Uji Sensor MPU6050 | Memeriksa sensor mencakup pembacaan data akselerasi dan *gyroscope* pada kondisi diam, berjalan, dan jatuh. | Berhasil |
| Pengujian *Power Supply* | Menghubungkan sumber daya ke sistem dan memastikan perangkat menyala dengan stabil. | Berhasil |
| Uji Koneksi Wi-Fi | Menguji kestabilan ESP32-C6 dalam menghubungkan ke jaringan Wi-Fi dan mengirim data ke *Firebase Realtime Database*. | Berhasil |

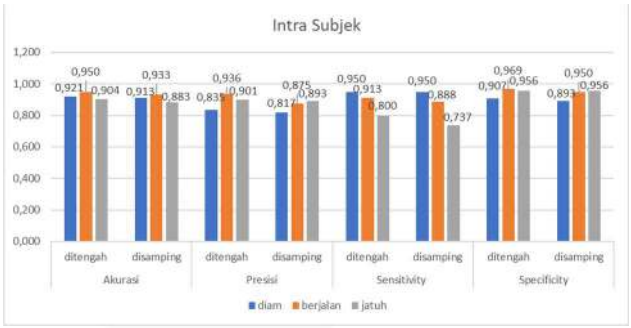
Tabel 4 menyajikan hasil pengujian *hardware* untuk memastikan kesesuaian antara desain sistem dengan fungsionalitas perangkat keras dan perangkat lunak. Pengujian tersebut mencakup pengecekan koneksi komponen (ESP32-C6, MPU6050, baterai), pembacaan data sensor MPU6050 pada tiga kondisi (diam, berjalan, jatuh), pengujian *power supply*, serta konektivitas ESP32-C6 ke Wi-Fi dan *Firebase Realtime Database.* Hasilnya menunjukkan bahwa seluruh komponen berfungsi dengan baik, sensor dapat membaca perubahan gerakan, pasokan daya stabil, dan perangkat berhasil mengirimkan data secara *real-time*.

Pada penelitian ini, pengujian sistem deteksi jatuh dilakukan pada 4 subjek dengan posisi alat di tengah dan di samping (kiri dan kanan). Setiap subjek melakukan tiga skenario aktivitas yaitu diam, berjalan, dan jatuh. Seluruh data hasil pengujian dicatat dan diolah ke dalam bentuk *confusion matrix* dan dihitung nilai parameter evaluasi yaitu akurasi, presisi, *sensitivity*, dan *specificity*. Dilakukan dua jenis pengujian, yaitu intra subjek dan inter subjek. Pada pengujian intra subjek, parameter evaluasi dihitung untuk setiap kondisi pada masing-masing subjek untuk mengetahui konsistensi sistem dalam mendeteksi pola aktivitas yang sama pada subjek yang berbeda. Sementara pada pengujian inter subjek, parameter dihitung berdasarkan gabungan seluruh data dari keempat subjek untuk semua kondisi aktivitas untuk menilai kinerja sistem secara keseluruhan.

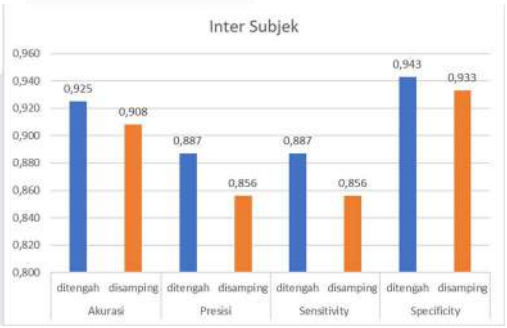
TABEL 5
Hasil Evaluasi Model per Kondisi (Posisi Alat di Tengah dan Samping)

| Posisi Alat | Kondisi | TP | FP | FN | TN |
|---|---|---|---|---|---|
| Tengah | Diam | 76 | 15 | 4 | 146 |
| | Berjalan | 73 | 5 | 7 | 155 |
| | Jatuh | 64 | 7 | 16 | 153 |
| Samping | Diam | 76 | 17 | 4 | 143 |
| | Berjalan | 73 | 8 | 7 | 152 |
| | Jatuh | 59 | 7 | 21 | 153 |

Tabel 5 menyajikan nilai TP, FP, FN, dan TN yang diperoleh dari perhitungan *confusion matrix* untuk masing-masing kondisi pada posisi pemakaian alat di tengah dan samping. Nilai-nilai tersebut digunakan untuk menghitung parameter akurasi, presisi, *sensitivity*, dan *specificity*.




GAMBAR 10
Grafik Hasil Parameter Intra Subjek




GAMBAR 11
Grafik Hasil Parameter Inter Subjek

Gambar 10 pengujian intra subjek menunjukkan nilai akurasi tertinggi tercapai pada kondisi jatuh ketika alat dipasang di tengah. Terdapat perbedaan nilai parameter akurasi, presisi, *sensitivity*, dan *specificity* antara posisi alat di tengah dan di samping, dengan kecenderungan nilai lebih tinggi saat alat dipasang di tengah. Hal ini mengindikasikan bahwa letak pemasangan alat sangat memengaruhi kinerja sistem terutama dalam mengenali kondisi kritis seperti kejadian jatuh.

Nilai parameter untuk pengujian inter subjek juga mengalami peningkatan ketika alat dipasang di tengah.

Meskipun demikian, nilai akurasi sistem hanya mencapai 92%, dengan selisih ±3% di bawah target spesifikasi yang ditetapkan yaitu 95%. Selain itu, presisi sistem hanya mencapai 88%, berada ±2% di bawah spesifikasi yang ditetapkan yaitu 90%.

Ketidaksesuaian ini kemungkinan disebabkan oleh penggunaan logika *threshold* yang masih belum sepenuhnya andal dalam mengidentifikasi kejadian jatuh, serta keterbatasan jumlah sensor yang digunakan. Selain itu, meskipun *moving average filter* telah digunakan untuk mengurangi *noise*, karakteristiknya statis yang mengakibatkan kurang responsif terhadap perubahan mendadak dalam data sensor.

Pengujian *delay* dilakukan untuk mengukur waktu respons sistem dalam mendeteksi kejadian dan mengirimkan data ke Firebase. Berdasarkan 10 kali pengujian pada empat skenario aktivitas (Diam–Berjalan, Berjalan–Diam, Berjalan–Jatuh, dan Diam–Jatuh), diperoleh *delay* tertinggi pada aktivitas D–B (3,4 detik) dan terendah pada D–J (1,8 detik), dengan rata-rata keseluruhan sebesar 2,85 detik. Nilai ini menunjukkan responsivitas sistem yang baik dalam mengirimkan data ke Firebase untuk mendukung notifikasi peringatan jatuh secara tepat waktu.

TABEL 6
Hasil Pengujian QoS Menggunakan Wireshark

| Kategori | Nilai |
|---|---|
| *Delay* | 26,84 ms |
| *Jitter* | 27,25 ms |
| *Throughput* | 218,63 kbps |
| *Packet Loss* | 0 % |

Sementara itu, pengujian *delay* jaringan menggunakan Wireshark menghasilkan estimasi *delay* rata-rata sebesar 26,84 ms, jauh di bawah batas standar maksimum 150 ms menurut TIPHON dan ITU-T. Hal ini mengindikasikan bahwa kualitas transmisi data jaringan berada dalam kategori sangat baik dan tidak menjadi hambatan dalam proses komunikasi aplikasi. Selain itu, Menurut standar TIPHON, *throughput* yang didapatkan termasuk kategori "*Bad*". Meskipun jumlah paket yang dikirim cukup banyak, ukuran masing-masing paket relatif kecil karena hanya memuat data sensor sederhana. Akibatnya, kecepatan rata-rata transfer data (*throughput*) menjadi rendah.

Pengukuran *Mean Opinion Score* (MOS) dilakukan berdasarkan standar ITU-T G.107 (E-model) yang juga digunakan oleh perangkat lunak Vivinet Diagnostics [8]. Pengukuran ini melibatkan 42 responden yang terdiri dari mahasiswa Telkom University dan pengguna simulasi aplikasi lainnya, baik dari kalangan umum maupun yang memiliki pengalaman merawat lansia melalui Google Form. Mayoritas responden (73,17%) memiliki hubungan langsung dengan lansia, sehingga penilaian yang diberikan sangat relevan. Sebagian besar aspek mencatat nilai rata-rata di atas 4,5 dan median pada angka 5 (skala 5). Akses menu utama, kenyamanan bagi lansia, dan kemudahan penggunaan umum menjadi aspek dengan skor tertinggi (≥ 4,6) dan deviasi standar yang rendah (< 0,7), mencerminkan kepuasan yang merata. Ukuran *font* & ikon serta kecepatan aplikasi juga dinilai sangat memuaskan, menunjukkan desain yang sesuai dengan kebutuhan lansia. Hanya desain tampilan yang memperoleh skor sedikit lebih rendah (4,19) dan standar deviasi 0,76, namun tetap masuk kategori *satisfied*, mengindikasikan ruang untuk peningkatan visual. Tidak ada responden yang merasa aplikasi ini tidak membantu. Sebaliknya, sebagian besar memberikan masukan positif, termasuk usulan fitur seperti GPS, pengingat obat, dan integrasi data medis. Secara keseluruhan, Fall-Guard dinilai fungsional, responsif, dan ramah lansia, dengan potensi besar untuk meningkatkan keamanan serta ketenangan bagi pengguna dan keluarga mereka.

Selain pengujian berbasis pengguna, evaluasi teknis dilakukan untuk mengukur performa sistem dalam merespons fitur dan notifikasi, menggunakan dua pendekatan: *Firebase Performance Monitoring* (melalui *custom trace* dan *trace* otomatis seperti *screen rendering* serta *network*), serta *logcat* dengan pencatatan waktu manual *(timestamp*). Pengujian ini mengacu pada ambang batas dari *PageSpeed Insights*, yang menggunakan metrik FCP (*First Contentful Paint*) mengukur waktu dari awal pemuatan aplikasi hingga konten pertama muncul di layar. INP (*Interaction to Next Paint*) mencerminkan kecepatan aplikasi merespons interaksi pengguna, seperti klik atau input. TTFB (*Time to First Byte*) menunjukkan durasi dari permintaan pengguna hingga server mengirimkan respons pertama [9].

Selama pengujian pada 25–27 Juni 2025, seluruh fitur diuji langsung di perangkat Android fisik. Analisis performa dilakukan melalui Firebase *Console* menggunakan fitur "*Last 24 hours*" dan persentil 90%. Pengujian performa dilakukan dengan menganalisis *network request traces*, yaitu pengukuran waktu respons dari permintaan jaringan yang dilakukan oleh aplikasi. Hasil pengujian *Network Request Trace* menunjukkan sebagian besar *endpoint* eksternal seperti s.yimg.com, static.foxnews.com, dan s3files.core77.com memiliki waktu respons cepat (< 210 ms) dengan masing-masing terdapat 6 *sample*. Namun, *endpoint* utama newsapi.org yang digunakan di menu *Home* menunjukkan rata-rata waktu respons sebesar 2,41 detik dalam 18 *sample*, tergolong lambat dan berpotensi menurunkan kenyamanan pengguna. Oleh karena itu, optimasi pemanggilan API direkomendasikan agar pengalaman pengguna tetap responsif.

Sedangkan hasil pengujian menggunakan *custom trace* Firebase, sebagian besar proses dalam aplikasi menunjukkan performa yang baik dengan waktu respons di bawah ambang ideal. Beberapa *trace* seperti starter_fragment_load, home_fragment_load, profile_fragment_load, splash_screen_load, dan app_start mencatat durasi antara 62 ms hingga 1,58 detik, yang tergolong baik karena berada di bawah batas FCP 1800 ms. *Trace* metrics_ui_render juga menunjukkan hasil yang sangat baik, yaitu 129 ms, sesuai dengan batas INP ≤ 200 ms. Namun, beberapa proses seperti notion_card_click (397 ms) dan profile_data_load (312 ms) tergolong perlu peningkatan karena melebihi 200 ms yang disebabkan oleh beban di UI *thread* dan interaksi antar aplikasi. Proses login_flow_duration memiliki hasil paling lambat yaitu 11,21 detik, dan masuk kategori buruk karena melebihi batas *Time to First Byte* (TTFB ≥ 1800 ms). Hal ini dikarenakan mencakup seluruh interaksi autentikasi pengguna, sementara splash_screen_load yang mencatat 1,50 detik tampak lama akibat adanya penundaan yang ditambahkan secara manual. Proses _app_start (758 ms), firebase_auth_duration (750 µs), dan metrics_fragment_load (132 ms) mencatat hasil di bawah 800 ms, yang mencerminkan efisiensi baik, terutama untuk aktivitas awal dan autentikasi. Secara umum, performa antarmuka aplikasi

dinilai optimal, namun proses tertentu masih perlu dioptimalkan untuk meningkatkan kenyamanan pengguna.

Fokus utama dari pengujian ini adalah mengamati *frozen frames*, yaitu *frame* yang membutuhkan waktu lebih dari 700 ms untuk dirender. *Frozen frames* sering kali menyebabkan tampilan aplikasi terasa patah atau *lag* bagi pengguna. Sebagian besar aktivitas seperti *Login, Home*, *Metric, Starter, Splash, Sign In,* dan *Profile* menunjukkan 0% *frozen frames*, menandakan performa UI sangat baik. Namun, *MainActivity* mengalami 2,38% *frozen frames* dalam 75 *sample*, yang berpotensi mengganggu transisi awal. Hal ini kemungkinan disebabkan oleh beban proses awal seperti inflasi tampilan dan pemuatan data besar. Optimalisasi dengan menunda proses berat ke *background* direkomendasikan untuk meningkatkan responsivitas awal aplikasi.

Di sisi lain, pengujian dengan *Logcat* menunjukkan sebagian besar komponen aplikasi memiliki performa baik sesuai standar FCP dan INP, seperti *Starter* (45 ms)*, Metrics* (1–33 ms), dan *Profile Fragment* (10–90 ms ) yang memuat di bawah 100 ms. *Splash Activity* masuk dalam kategori *perlu peningkatan* karena adanya *delay* eksplisit 1500 ms, bukan karena lambatnya proses *render*. *Login Activity* tercatat cepat menurut standar TTFB (623 ms). *Home Fragment* menunjukkan waktu UI (78-100 ms) yang tergolong sangat baik menurut FCP dan waktu *fetch* API (1799–2415 ms) yang lambat dalam standar TTFB, sehingga masuk ke dalam kategori *buruk* hingga *perlu peningkatan*. Hal ini disebabkan oleh lokasi server NewsAPI di luar negeri. Bermbach dan Wittern menyatakan bahwa latensi API sangat dipengaruhi oleh lokasi fisik dan jalur komunikasi antara klien dan server; semakin jauh jaraknya, semakin tinggi latensinya [10]. Selain itu, penggunaan *query* kompleks seperti *"elderly fall OR fall prevention"* dapat memperpanjang waktu pemrosesan karena melibatkan evaluasi logika *boolean* ganda [11]. Lalu, pengujian pengiriman notifikasi jatuh menunjukkan respons 0–870 ms. Sesuai standar ITU-T G.1010, nilai di bawah 400 ms tergolong sangat baik, sementara di atasnya dapat menurunkan kualitas interaktif tergantung jaringan. Secara umum, performa antarmuka aplikasi sudah optimal, namun masih terdapat hambatan pada proses pengambilan data eksternal dari API pihak ketiga. Oleh karena itu, disarankan dilakukan optimisasi pada sisi *back-end* untuk meningkatkan waktu respons aplikasi.

## V. KESIMPULAN

Penelitian ini telah berhasil merancang dan mengimplementasikan sistem deteksi jatuh berbasis *Internet of Things* (IoT) yang ditujukan untuk mendukung keamanan lansia. Perangkat *wearable* yang dikembangkan memiliki dimensi kecil dan bobot ringan, sehingga nyaman digunakan dalam aktivitas sehari-hari oleh lansia. Sistem deteksi menggunakan sensor MPU6050 dan mikrokontroler ESP32-C6 terbukti mampu mengenali tiga kondisi utama, yaitu diam, berjalan, dan jatuh, dengan akurasi sebesar 92%, presisi 88%, sensitivitas 88%, dan spesifisitas 94%. Meski nilai akurasi dan presisi masih sedikit di bawah target ideal yang ditetapkan, hasil ini menunjukkan bahwa sistem telah berjalan secara fungsional. Rata-rata *delay* pengiriman data ke Firebase sebesar 2,85 detik serta *delay* jaringan sebesar 26,84 ms menunjukkan bahwa sistem mampu merespons kejadian secara cepat dan mendukung kebutuhan akan notifikasi *real-time*. Aplikasi *mobile* Fall-Guard yang terhubung dengan sistem IoT mampu menampilkan status aktivitas secara *real-time*, menyediakan riwayat aktivitas, serta mengirimkan notifikasi otomatis jika deteksi jatuh terjadi. Evaluasi usability dengan metode *Mean Opinion Score* (MOS) memberikan skor rata-rata 4,3 dari 5, mengindikasikan kepuasan tinggi dari pengguna terhadap performa aplikasi. Dengan demikian, sistem ini dapat memberikan kontribusi nyata terhadap peningkatan keselamatan lansia dan dapat dikembangkan lebih lanjut dengan integrasi metode kecerdasan buatan untuk peningkatan akurasi.

## REFERENSI


[1] Salsabila, S. (2024). *Gambaran tingkat risiko jatuh pada lansia berdasarkan Berg Balance Scale di Puskesmas Muara Dua Kota Lhokseumawe 2023* (Doctoral dissertation, Universitas Malikussaleh).

[2] Kamesyworo, K., Haryanti, E., Hartati, S., & Elviani, Y. (2024). Pelatihan Deteksi Dini Terserang Stroke Dengan Metode Fast Pada Lansia Di Kelurahan Sari Bunga Mas Kecamatan Lahat. *Jurnal Abdi Kesehatan dan Kedokteran*, *3*(2), 133-139.

[3] Fitriandini, L., Suhardi, S., & Sari, K. (2025). Fall Detector pada Lansia berbasis IoT Menggunakan Sensor MPU-6050 dan Sensor GPS Neo 6M. *Journal of Telecommunication Electronics and Control Engineering (JTECE)*, *7*(1), 10-22.

[4] Eriska, S., Arradea, M. R. M., Saputra, Z., & Khasanah, N. (2025). Alat Pendeteksi Jatuh pada Lansia dalam Keadaan Rawat Jalan Berbasis Internet of Things (IoT). *Jurnal Inovasi Teknologi Terapan*, *3*(1), 199-206.

[5] M. R. Ahmad, V. Fatmawati, and A. Ariyanto, "Faktor-faktor yang mempengaruhi resiko jatuh pada lansia di PCA Pajangan, Yogyakarta," in *Prosiding Seminar Nasional Penelitian dan Pengabdian Kepada Masyarakat LPPM Universitas 'Aisyiyah Yogyakarta*, vol. 2, pp. 408–416, Oct. 2024.

[6] Centers for Disease Control and Prevention. (2024, May 10). *Important facts about falls*. https://www.cdc.gov/falls/data-research/index.html

[7] Purnomo, C. F., & Adriansyah, A. (2022). Rancang bangun fall detector system untuk pasien stroke dengan metode WSN (Wireless Sensor Network). *Jurnal Teknologi Elektro, 13*(1), 29–34.

[8] *Micro Focus. (2015). NetIQ Vivinet Diagnostics User Guide: Reviewing diagnosis factors.*https://www.netiq.com/documentation/appmanager-vivinet/vdiaguserguide/data/reviewing_diagnosis_factors.html#one_way_delay

[9] Google Developers. (n.d.). *PageSpeed Insights – Overview*. https://developers.google.com/speed/docs/insights/v5/about

[10] Bermbach, D., & Wittern, E. (2020). Benchmarking Web API quality – Revisited. *Journal of Web Engineering, 19*(5–6), 603–646. https://doi.org/10.13052/jwe1540-9589.19563

[11] MongoDB Inc. (n.d.). *Query optimization*. https://www.mongodb.com/docs/manual/core/query-optimization/