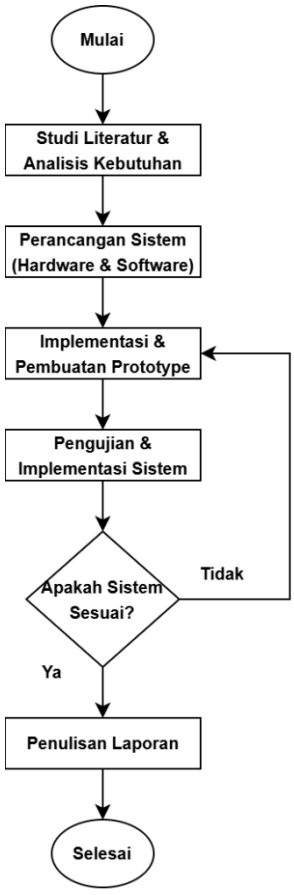
BAB III

METODOLOGI PENELITIAN

3.1 Desain Penelitian

Penelitian ini menggunakan desain Penelitian dan Pengembangan atau *Research and Development* (R&D). Desain R&D dipilih karena tujuan utama penelitian ini adalah untuk menghasilkan dan menguji sebuah produk teknologi baru, yaitu prototipe sistem deteksi jatuh berbasis *edge computing*, serta untuk mengevaluasi efektivitas dan kinerjanya secara empiris. Pendekatan R&D bersifat siklis dan berorientasi pada produk, yang mencakup serangkaian tahapan mulai dari analisis kebutuhan, perancangan produk, pengembangan, hingga pengujian di lingkungan yang relevan. Pada gambar 3.1 dijelaskan alur diagram tahapan metode penelitian.



Gambar 3.1 Kerangka Alur Penelitian

Penelitian ini diawali dengan analisis permasalahan, di mana kelemahan sistem deteksi jatuh konvensional seperti latensi tinggi dan ketergantungan internet diidentifikasi. Sebagai solusinya, diusulkan sistem berbasis *edge computing* dengan arsitektur *hybrid*, yang didukung oleh studi literatur mendalam mengenai teknologi AIoT, ESP32-S3, MPU6050, dan algoritma *Random Forest*.

Selanjutnya, dilakukan tahap perancangan untuk membuat prototipe yang ringkas dan nyaman dikenakan, dengan alur kerja yang jelas: *input* data gerak dari MPU6050, proses algoritma *hybrid* di perangkat, dan *output* berupa notifikasi darurat.

Pada tahap pengembangan, prototipe dibangun menggunakan perangkat keras berkinerja tinggi dan terjangkau, yaitu mikrokontroler ESP32-S3 sebagai unit pemrosesan utama dan sensor MPU6050 untuk akuisisi data gerak. Pengembangan perangkat lunak melibatkan Python dengan Scikit-learn untuk melatih model dan Arduino IDE untuk memprogram perangkat.

Kemudian, dilakukan tahap uji coba secara iteratif untuk memastikan prototipe dapat bekerja optimal dalam membedakan antara jatuh dan aktivitas harian (ADL). Jika ditemukan alarm palsu atau kegagalan deteksi, dilakukan perbaikan pada algoritma dan pengujian diulang hingga mencapai performa yang diinginkan.

Terakhir, pada tahap pelaporan, seluruh data hasil pengujian, seperti akurasi dan kecepatan respons, dianalisis dan disajikan. Kesimpulan ditarik berdasarkan temuan tersebut untuk menjawab rumusan masalah dan tujuan penelitian yang telah ditetapkan.

Melalui desain R&D ini, penelitian tidak hanya bertujuan untuk menguji hipotesis, tetapi juga untuk menghasilkan sebuah prototipe yang fungsional dan tervalidasi, yang dapat menjadi solusi konkret untuk permasalahan yang telah diidentifikasi.

3.2 Instrumen Penelitian

Instrumen yang digunakan dalam penelitian ini terdiri dari komponen perangkat keras untuk membangun prototipe dan perangkat lunak untuk pengembangan algoritma dan analisis data.

3.2.1 Perangkat Keras

Perangkat keras yang dirancang untuk prototipe sistem deteksi jatuh ini terdiri dari beberapa komponen utama yang dipilih berdasarkan kriteria performa, ketersediaan, dan biaya yang terjangkau. Komponen-komponen tersebut diintegrasikan untuk membentuk sebuah perangkat *wearable* yang ringkas dan fungsional.

1) Mikrokontroler ESP32-S3

   Komponen ini berfungsi sebagai unit pemrosesan pusat (*Central Processing Unit* / CPU) dari sistem. Model ESP32-S3 dipilih secara spesifik karena keunggulannya dalam aplikasi AIoT di level *edge*. ESP32-S3 dilengkapi dengan prosesor *dual-core* Tensilica Xtensa LX7, konektivitas Wi-Fi dan Bluetooth 5 (LE) terintegrasi, serta yang terpenting, memiliki akselerasi perangkat keras untuk operasi AI dan *machine learning*. Kemampuan ini sangat krusial untuk menjalankan model *Random Forest* secara efisien langsung di perangkat (Noor Mohammad Rafee, t.t.)

2) Sensor *Inertial Measurement Unit* (IMU) MPU6050

   Modul ini digunakan untuk akuisisi data gerak. MPU6050 adalah sensor terintegrasi yang menggabungkan akselerometer 3-sumbu dan giroskop 3-sumbu. Akselerometer bertugas mengukur percepatan linear untuk mendeteksi benturan, sementara giroskop mengukur kecepatan sudut untuk mendeteksi perubahan orientasi tubuh yang cepat. Sensor ini dipilih karena merupakan pilihan yang efisien dan efektif untuk perangkat *wearable* berbiaya rendah dan telah terbukti andal dalam berbagai penelitian deteksi jatuh (Britto Filho & Lubaszewski, 2020; Moreira dkk., 2024)

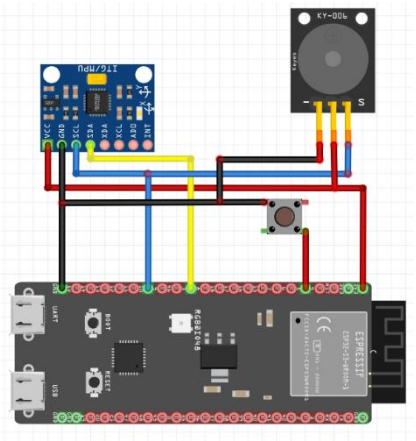
3) *Buzzer*

Komponen ini berfungsi sebagai aktuator yang memberikan umpan balik auditori secara langsung kepada pengguna atau orang di sekitarnya. Ketika sistem mendeteksi dan memvalidasi sebuah insiden jatuh, *buzzer* akan mengeluarkan suara alarm yang nyaring untuk memberikan peringatan darurat secara instan.

4) *Push Button*

Tombol ini diimplementasikan sebagai *panic button*. Fitur ini memungkinkan pengguna untuk secara manual memanggil bantuan dengan menekan tombol, terlepas dari apakah sistem mendeteksi jatuh secara otomatis atau tidak. Ini memberikan lapisan keamanan tambahan bagi pengguna dalam situasi darurat lainnya.

Seluruh komponen ini dirakit pada sebuah *breadboard* yang dirancang agar ringan untuk dikenakan di tubuh. Untuk memperjelas arsitektur perangkat keras, Gambar 3.1 menyajikan diagram skematis dari rangkaian prototipe yang digunakan dalam penelitian ini.



Gambar 3.2 Skematis Rangkaian Sistem

3.2.2 Perangkat Lunak

Pengembangan perangkat lunak untuk sistem ini melibatkan beberapa platform dan *library* yang berbeda, yang masing-masing memiliki peran spesifik dalam alur kerja penelitian.

1) Arduino IDE

   Integrated Development Environment (IDE) ini digunakan untuk memprogram mikrokontroler ESP32-S3. Bahasa pemrograman yang digunakan adalah C/C++, yang dioptimalkan untuk sistem *embedded*. Arduino IDE dipilih karena kemudahan penggunaannya, dukungan komunitas yang luas, dan ketersediaan banyak *library* untuk mengontrol sensor MPU6050, mengelola koneksi Wi-Fi, dan mengoperasikan komponen lainnya.

2) Python dengan Scikit-learn

   Bahasa pemrograman Python digunakan untuk seluruh tahap pengembangan model *machine learning* di luar perangkat. *Library* Scikit-learn secara khusus digunakan untuk melatih, menguji, dan mengoptimalkan *classifier Random Forest*. Scikit-learn dipilih karena merupakan standar industri untuk *machine learning* klasik, menyediakan implementasi algoritma yang efisien dan antarmuka yang mudah digunakan untuk evaluasi model (Noor Mohammad Rafee, t.t.).

3) Micromlgen

   Ini adalah *library* Python yang berfungsi sebagai jembatan antara model *machine learning* yang dilatih di Python dan implementasinya pada mikrokontroler. Setelah model *Random Forest* dilatih dan dioptimalkan menggunakan Scikit-learn, micromlgen digunakan untuk mengonversi model tersebut menjadi kode C++ yang portabel. Kode hasil konversi ini kemudian diintegrasikan ke dalam proyek Arduino IDE untuk dijalankan secara lokal di ESP32-S3.

4) Telegram Bot API

   Untuk sistem notifikasi darurat, digunakan *Application Programming Interface* (API) dari platform Telegram. Sebuah bot Telegram diprogram

untuk menerima perintah dari ESP32-S3 melalui koneksi internet. Ketika jatuh terdeteksi, ESP32 akan mengirim permintaan HTTP ke API Telegram, yang kemudian memicu bot untuk mengirimkan pesan peringatan ke pengguna atau grup yang telah ditentukan (Moreira dkk., 2024).

3.3 Prosedur Penelitian

Prosedur penelitian ini dilaksanakan secara sistematis mengikuti tahapan-tahapan dalam desain R&D. Alur kerja penelitian, mulai dari pengumpulan data hingga evaluasi akhir, dirancang untuk memastikan bahwa setiap komponen sistem dikembangkan dan diuji secara cermat.

3.3.1 Pengumpulan dan Pra-pemrosesan Data

Tahap pertama berfokus pada penyediaan data yang akan digunakan untuk melatih dan menguji model *machine learning*. Mengingat kesulitan dan risiko dalam mengumpulkan data jatuh dari lansia secara langsung, penelitian ini akan memanfaatkan *dataset* publik yang sudah tersedia. *Dataset* yang dipilih adalah SisFall Dataset (Sucerquia dkk., 2017), yang merupakan salah satu *dataset* paling komprehensif dan sering digunakan dalam penelitian deteksi jatuh. *Dataset* ini berisi rekaman data dari sensor akselerometer dan giroskop yang dikenakan oleh 38 subjek saat melakukan 15 jenis skenario jatuh yang berbeda dan 19 jenis Aktivitas Kehidupan Sehari-hari (ADLs).

Data mentah dari *dataset* SisFall kemudian akan melalui tahap pra-pemrosesan yang meliputi:

1) Pembersihan Data

   Menghapus data yang tidak lengkap atau anomali.

2) Segmentasi Data

   Memotong data kontinu menjadi jendela-jendela waktu (*time windows*) yang lebih kecil untuk dianalisis sebagai unit data individual.

3) Ekstraksi Fitur

Menghitung fitur-fitur statistik dari setiap jendela waktu dari data akselerometer dan giroskop. Proses ini bertujuan untuk mengubah data mentah menjadi representasi yang lebih informatif bagi model *machine learning* (Noor Mohammad Rafee, t.t.).

3.3.2 Pengembangan dan Pelatihan Model *Machine Learning*

Pada tahap ini, model klasifikasi *Random Forest* akan dikembangkan dan dilatih menggunakan data yang telah diproses. Proses ini dilakukan menggunakan bahasa pemrograman Python dengan *library* Scikit-learn.

1) Pembagian Data

Dataset yang telah diekstraksi fiturnya akan dibagi menjadi dua set: *training set* sebesar 80% dan *testing set* sebesar 20%.

2) Pelatihan Model

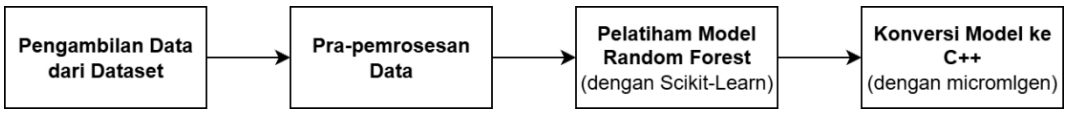
Model *Random Forest* akan dilatih menggunakan data latih untuk mempelajari pola yang membedakan antara kejadian jatuh dan ADLs.

3) Optimasi Model

Dilakukan *hyperparameter tuning* untuk menemukan konfigurasi model terbaik yang memberikan akurasi tertinggi, seperti menyesuaikan jumlah pohon (*n_estimators*) dan kedalaman maksimum pohon (*max_depth*).

4) Konversi Model

Setelah model final didapatkan, model tersebut akan dikonversi ke dalam format kode C/C++ menggunakan micromlgen agar dapat diimplementasikan pada mikrokontroler ESP32-S3.



Gambar 3.3 *Flowchart* Pengembangan Model *Random Forest*

Flowchart pada gambar 3.3 mengilustrasikan proses pelatihan model *random forest*

3.3.3 Implementasi Sistem pada Perangkat Keras

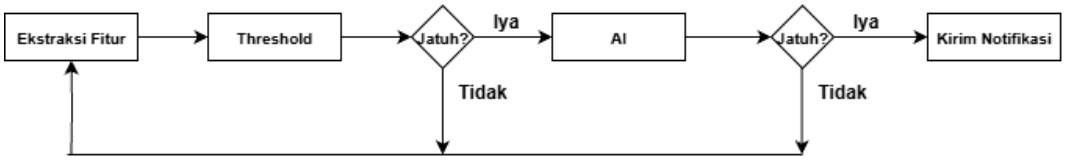
Tahap ini adalah implementasi algoritma *hybrid* pada prototipe perangkat keras yang telah dirancang.

1) Pemrograman Mikrokontroler

   Kode program akan ditulis menggunakan Arduino IDE. Kode program dirancang untuk menjalankan beberapa fungsi utama, yaitu: (1) melakukan akuisisi data dari sensor MPU6050, (2) mengimplementasikan algoritma *hybrid* yang dimulai dengan pengecekan *threshold* dan dilanjutkan dengan validasi menggunakan model *Random Forest*, (3) mengaktifkan buzzer sebagai alarm auditori, (4) mengirimkan notifikasi ke API Telegram, dan (5) menangani interupsi dari *push button*.

2) Integrasi Komponen

   Seluruh komponen perangkat keras akan dirakit dan diintegrasikan menjadi satu unit prototipe *wearable* yang fungsional. Tujuannya adalah untuk memastikan setiap komponen dapat berkomunikasi dan bekerja sama dengan baik, sehingga menghasilkan sebuah prototipe yang siap untuk diuji coba.



Gambar 3. 4 *Flowchart* Sistem

Flowchart pada Gambar 3.4 mengilustrasikan proses pengambilan keputusan sistem, mulai dari ekstraksi fitur hingga pengiriman notifikasi.

3.3.4 Pengujian dan Evaluasi Sistem

Tahap terakhir adalah pengujian prototipe secara menyeluruh untuk memvalidasi kinerjanya. Pengujian dilakukan di lingkungan yang terkontrol oleh peneliti sendiri sebagai subjek uji untuk memastikan keamanan dan konsistensi. Skenario pengujian dirancang untuk menyimulasikan kejadian jatuh dan berbagai

ADLs secara realistis. Untuk menjamin keselamatan selama simulasi jatuh, sebuah matras tebal diletakkan di lantai.

1) Persiapan

Peneliti mengenakan prototipe perangkat *wearable* pada posisi pinggang. Perangkat diaktifkan dan dipastikan terhubung ke jaringan Wi-Fi. Sebuah komputer dan ponsel disiapkan untuk memonitor *output* serial dari perangkat dan menerima notifikasi Telegram.

2) Pelaksanaan Skenario

Peneliti melakukan gerakan sesuai skenario yang telah ditentukan.

- Untuk Skenario Jatuh: Peneliti menyimulasikan jatuh ke empat arah: depan; belakan;, kanan; dan kiri di atas matras. Setiap jenis jatuh diulang beberapa kali untuk memastikan konsistensi data.
- Untuk Skenario ADL: Peneliti melakukan berbagai aktivitas sehari-hari yang berpotensi menghasilkan alarm palsu, seperti duduk dengan cepat, jongkok, berbaring di tempat tidur, dan berjalan. Pemilihan skenario ADL ini didasarkan pada gerakan-gerakan fundamental yang sering dievaluasi dalam asesmen risiko jatuh klinis, seperti tes *Timed Up & Go* dan tes berdiri dari kursi, yang menilai kemampuan mobilitas dasar dan kekuatan ekstremitas bawah (Giovannini dkk., 2022; Yi dkk., 2022).

3) Pencatatan Data

Selama pengujian, data berikut dicatat secara cermat:

- Apakah sistem mendeteksi gerakan tersebut sebagai 'Jatuh' atau 'ADL'.
- Untuk deteksi jatuh yang berhasil, waktu diukur mulai dari saat benturan terjadi hingga notifikasi diterima di ponsel.
- Waktu yang dibutuhkan ESP32 untuk melakukan inferensi model *machine learning* dicatat melalui *output* serial monitor untuk setiap prediksi.

4) Analisis

Data yang terkumpul kemudian dianalisis untuk menghitung metrik performa. Jika hasil pengujian belum optimal, dilakukan iterasi perbaikan pada algoritma dan pengujian diulang.

3.4 Analisis Data

Analisis data dalam penelitian ini bersifat kuantitatif dan bertujuan untuk mengevaluasi kinerja sistem dari dua aspek utama: keandalan model klasifikasi dan efisiensi sistem secara keseluruhan.

3.4.1 Analisis Kinerja Model

Untuk mengevaluasi seberapa baik model *Random Forest* yang telah dilatih dapat membedakan antara jatuh dan ADL, akan digunakan *confusion matrix*. Matriks ini akan memvisualisasikan performa model pada data uji dengan membandingkan hasil prediksi dengan label sebenarnya. Dari matriks ini, akan dihitung empat metrik statistik utama yang umum digunakan dalam evaluasi FDS (Amir dkk., 2024; Saleh & Jeannes, 2019):

1) Akurasi

Akurasi merupakan metrik yang digunakan untuk mengukur proporsi total prediksi yang benar (baik jatuh maupun ADL) dari keseluruhan data uji, yang perhitungannya dirumuskan dalam Persamaan (1).

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \quad (1)$$

2) Presisi

Perhitungan presisi, yang mengukur proporsi prediksi jatuh yang benar dari semua prediksi yang diklasifikasikan sebagai jatuh sehingga penting untuk meminimalkan alarm palsu, ditunjukkan pada Persamaan (2).

$$Presisi = \frac{TP}{TP + FP} \times 100\% \quad (2)$$

3) Sensitivitas

Perhitungan sensitivitas, yang mengukur proporsi kejadian jatuh yang sebenarnya berhasil terdeteksi oleh sistem dan bersifat krusial untuk

memastikan tidak ada insiden yang terlewat, ditunjukkan pada Persamaan (3).

$$Sensitivitas = \frac{TP}{TP + FN} \times 100\% \quad (3)$$

4) F1-Score

Perhitungan F1-Score, yang merupakan rata-rata harmonik dari presisi dan sensitivitas untuk memberikan skor tunggal yang menyeimbangkan kedua metrik tersebut, dirumuskan dalam Persamaan (4).

$$F1 - Score = 2 \times \frac{Presisi \times Sensitivitas}{Presisi + Sensitivitas} \times 100\% \quad (4)$$

Di mana:

- TP (*True Positive*): Jumlah kejadian jatuh yang diprediksi dengan benar sebagai jatuh.
- TN (*True Negative*): Jumlah ADL yang diprediksi dengan benar sebagai bukan jatuh.
- FP (*False Positive*): Jumlah ADL yang salah diprediksi sebagai jatuh (alarm palsu).
- FN (*False Negative*): Jumlah kejadian jatuh yang gagal terdeteksi.

3.4.2 Analisis Kinerja Sistem

Analisis ini berfokus pada evaluasi performa prototipe fisik.

Waktu komputasi akan diukur sebagai durasi yang dibutuhkan oleh mikrokontroler ESP32 untuk menjalankan seluruh proses deteksi, mulai dari ekstraksi fitur data sensor hingga menghasilkan *output* klasifikasi dari model *Random Forest.* Pengukuran ini akan dilakukan dalam satuan milidetik (ms) untuk menunjukkan efisiensi pemrosesan di perangkat.

Waktu respons akan diukur sebagai total latensi *end-to-end,* yaitu waktu yang dibutuhkan dari saat kejadian jatuh disimulasikan hingga notifikasi darurat berhasil diterima di aplikasi Telegram. Pengukuran ini akan diulang beberapa kali

untuk mendapatkan nilai rata-rata dan standar deviasi, yang akan disajikan dalam satuan detik (s).