PENGUKURAN KETINGGIAN UNTUK PENYANDANG TUNANETRA DAN LOW VISION MENGGUNAKAN SENSOR BAROMETER

PRAKTIK KERJA LAPANGAN

CLINT JOY PEREZ MELODY MOKOSANDI

NIM: 312210008

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG
MALANG
2025

LEMBAR PENGESAHAN

PRAKTIK KERJA LAPANGAN

PENGUKURAN KETINGGIAN UNTUK PENYANDANG TUNANETRA DAN LOW VISION MENGGUNAKAN SENSOR BAROMETER

Oleh:

CLINT JOY PEREZ MELODY MOKOSANDI

NIM. 312210008

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

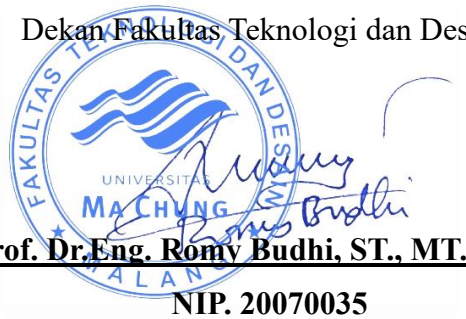
UNIVERSITAS MA CHUNG

Dosen Pembimbing,

Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd.

NIP. 20070035

Dekan Fakultas Teknologi dan Desain,



Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd.

NIP. 20070035

Kata Pengantar

Puji syukur saya panjatkan kehadirat Tuhan Yang Maha Esa karena atas rahmat dan hidayah-Nya sehingga penelitian ini dapat berjalan dengan baik. Laporan ini disusun untuk memenuhi salah satu persyaratan dalam menyelesaikan PKL (Praktik Kerja Lapangan) bagi Mahasiswa dari Prodi Teknik Informatika Universitas Ma Chung Malang.

Pada kesempatan ini, penulis mengucapkan terima kasih kepada seluruh pihak yang telah membantu penulis ketika melaksanakan Praktik Kerja Lapangan (PKL) maupun dalam pembuatan Laporan Praktik Kerja Lapangan, di antaranya:

1. Orang Tua yang selalu mendukung dan menyemangati selama berlangsungnya masa PKL ini.
2. Bapak Dr. Eng. Romy Budhi, ST, MT. selaku kepala peneliti, dosen pembimbing Praktik Kerja Lapangan serta Dekan Fakultas Teknologi dan Desain Universitas Ma Chung,
3. Teman-teman yang ikut serta terlibat untuk mendukung penelitian ini,

Susunan laporan Praktik Kerja Lapangan ini dibuat dengan sebaik-baiknya, namun tidak menutup kemungkinan ditemukan adanya kekurangan dan kesalahan. Semoga laporan Praktik Kerja Lapangan ini dapat bermanfaat bagi pembaca.

Malang, 26 Mei 2025

Clint Joy Perez Melody Mokosandi

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab I
Pendahuluan

1.1 Latar Belakang

Di era saat ini teknologi berkembang sangat pesat, membuat teknologi menjadi bagian tak terpisahkan dalam kehidupan sehari-hari, memberikan kemudahan dalam berbagai aspek, mulai dari komunikasi, pendidikan, hingga pekerjaan. Di tengah kemajuan teknologi saat ini, teknologi juga membuka peluang besar untuk penyandang disabilitas agar dapat menjalani kehidupan dengan lebih mandiri, aman, dan nyaman. Salah satu nya bagi tunanetra dan *low vision impersial*, sering kali menghadapi berbagai tantangan dalam menjalani aktivitas sehari-hari, seperti keterbatasan dalam mobilitas, komunikasi, atau akses informasi. Hal ini dapat menghambat dalam aktivitas sehari-hari. Oleh karena itu, kehadiran teknologi, khusussnya teknologi bantu, menjadi solusi penting untuk mendukung kemandirian dan meningkatkan kualitas hidup penyandang disabilitas, terutama bagi penyandang tuna netra dan *low vision impersial*.

Penyandang tunanetra dan *low vision* mengalami kesulitan dalam hal yang berkaitan dengan orientasi dan mobilitas di ruang publik, termasuk saat mengakses fasilitas di gedung bertingkat. Aktifitas sederhana seperti naik elevator, menggunakan tangga atau berpindah dari satu lantai ke lantai lain sering kali menjadi tantangan besar karena keterbatasa penglihatan membuat mereka sulit mengenali arah, tombol, atau rambu-rambu yang biasanya disediakan untuk pengguna umum.

Di lingkungan gedung bertingkat, keterbatasan fasilitas navigasi bagi penyandang disabilitas penglihatan masih menjadi masalah kritis. Minimnya penanda taktil dan sistem panduan audio di lift atau tangga membuat penyandang tunanetra kesulitan menentukan posisi lantai secara mandiri (Zhang et al., 2016). Dalam upaya meningkatkan navigasi indoor bagi penyandang tunanetra, berbagai solusi berbasis smartphone dan wearable telah dikembangkan dengan pendekatan beragam. Ahmetovic et al. (2016) memperkenalkan sistem *NavCog* yang memanfaatkan jaringan Bluetooth Low Energy (BLE) dan sensor smartphone untuk memberikan panduan audio 3D secara real-time, mencapai akurasi 1–2 meter dengan instalasi *beacon* strategis. Pengembangan lebih mutakhir oleh Murata et al. (2018) menghilangkan ketergantungan pada infrastruktur

eksternal melalui pendekatan *smartphone-only*, di mana kombinasi barometer untuk deteksi lantai dan magnetometer untuk navigasi horizontal mampu beroperasi di kompleks gedung multi-lantai dengan akurasi lantai 98%, didukung algoritma *machine learning* yang memetakan pola medan magnetik unik. Untuk lingkungan terbatas, Silva dan Wimalaratne (2016) mengusulkan solusi hemat daya berbasis *sensor fusion* yang mengintegrasikan IMU smartphone, barometer, dan pemrosesan suara, dirancang khusus untuk navigasi di ruang sempit dengan fitur deteksi *obstacle* melalui getaran *haptik* dan konsumsi daya di bawah 200 mAh. Lebih lanjut, Mai et al. (2023) mengembangkan *Smart Cane* berbasis sensor 2D LiDAR dan kamera RGB-D, yang mampu melakukan navigasi dan pengenalan rintangan secara simultan. Tongkat ini mengombinasikan pemindaian lingkungan secara visual dan spasial untuk menghasilkan umpan balik navigasi yang real-time, akurat, serta meningkatkan kesadaran spasial pengguna tanpa ketergantungan pada *infrastruktur* eksternal tambahan. Selain itu, teknik berbasis fingerprinting Wi-Fi dengan deep neural network juga telah diuji untuk klasifikasi lokasi pada bangunan multi-lantai—Kim et al. (2017) mengembangkan DNN dengan autoencoder dan classifier untuk mengestimasi gedung, lantai, dan koordinat lokasi secara simultan, menghasilkan akurasi deteksi lantai yang mendekati kondisi *state-of-the-art* serta efisiensi energi yang cukup rendah untuk perangkat mobile Namun, sebagian besar solusi tersebut masih membutuhkan infrastruktur tambahan atau perangkat keras khusus yang tidak sederhana, berbeda dengan pendekatan sensor barometer tunggal yang digunakan dalam penelitian ini, yang lebih praktis dan ekonomis.

Untuk mengatasi permasalahan tersebut, diperlukan sebuah solusi berbasis teknologi yang mampu membantu penyandang tunanetra dan low vision dalam mengenali posisi lantai secara mandiri di dalam gedung bertingkat. Salah satu solusi inovatif yang dapat diterapkan adalah dengan memanfaatkan sensor barometrik. Sensor ini mampu mendeteksi perubahan tekanan udara seiring perbedaan ketinggian, sehingga dapat digunakan untuk menentukan posisi lantai di dalam gedung secara *real-time*.tombol elevator berbasis suara, atau panduan arah berbasis smartphone yang terintegrasi dengan sistem gedung pintar (*smart building*).

Dengan adanya sistem berbasis sensor barometer, penyandang disabilitas penglihatan dapat memperoleh informasi yang jelas mengenai posisi lantai tanpa harus

bergantung pada bantuan orang lain. Informasi ini dapat disampaikan melalui sistem suara atau getaran pada perangkat yang digunakan, sehingga memudahkan pengguna dalam mengakses fasilitas lift atau tangga di gedung bertingkat. Salah satu teknologi yang telah digunakan dalam sistem navigasi dalam ruangan adalah sensor barometer. Bai (2015) dalam penelitiannya yang berjudul *A Wearable Indoor Navigation System for Blind and Visually Impaired Individuals* memanfaatkan sensor barometer untuk menentukan ketinggian atau lantai dalam gedung bertingkat sebagai bagian dari sistem lokalization statistik. Sensor barometer digunakan untuk memperkirakan posisi vertikal (ketinggian lantai) karena mampu memberikan informasi tekanan udara yang berkorelasi dengan ketinggian secara real-time.

Pada penelitian ini, dipilih sensor barometer karena memiliki beberapa keunggulan, antara lain biaya yang relatif murah, konsumsi daya rendah, bentuk yang ringkas, serta tingkat keakuratan yang masih dapat diterima untuk aplikasi penentuan lantai di dalam gedung. Selain itu, penggunaan barometer tidak memerlukan infrastruktur tambahan seperti anchor UWB atau beacon BLE, sehingga lebih praktis dan mudah diimplementasikan.

Penerapan teknologi ini diharapkan tidak hanya meningkatkan aksesibilitas dan keamanan bagi penyandang tunanetra dan low vision, tetapi juga menjadi langkah nyata menuju terciptanya lingkungan yang inklusif dan berkeadilan bagi semua kalangan.

1.2 Batasan Masalah

Batasan masalah pada Tugas Akhir ini sebagai berikut,

1. Sistem pengukuran ketinggian ini menggunakan sensor barometer BMP280 sebagai sumber data utama tanpa kombinasi sensor lain seperti GPS atau akselerometer.
2. Penentuan posisi vertikal dibatasi pada penentuan lantai di dalam gedung bertingkat, tidak untuk pengukuran ketinggian absolut dari permukaan laut.
3. Pengujian dilakukan di lingkungan gedung bertingkat minimal 3 lantai, sehingga akurasi alat pada bangunan rendah (1–2 lantai) tidak dianalisis lebih lanjut.

1.3 Rumusan Masalah

Bagaimana merancang dan mengimplementasikan alat pengukur ketinggian berbasis sensor barometer untuk menentukan posisi lantai di gedung bertingkat?

1.4 Tujuan

Tujuan dari penelitian ini adalah:

1. Merancang dan mengembangkan alat pengukur ketinggian berbasis sensor barometer untuk menentukan posisi lantai di gedung bertingkat.
2. Menguji tingkat akurasi dan konsistensi alat dalam membedakan posisi antar lantai di lingkungan gedung bertingkat.

1.5 Manfaat

Manfaat dari Praktik Kerja Lapangan ini adalah:

a. Bagi Mahasiswa:
    1. Menyediakan alat bantu yang dapat memberikan informasi posisi lantai secara otomatis melalui umpan balik suara, sehingga dapat membantu pengguna dalam mengenali lokasi vertikal di gedung tanpa perlu visualisasi tambahan.

b. Bagi Mahasiswa:
    1. Menambah pengetahuan dan pemahaman dalam perancangan serta implementasi sistem pengukuran ketinggian menggunakan sensor barometer berbasis mikrokontroler.
    2. Memberikan pengalaman langsung dalam pengembangan sistem instrumentasi, mulai dari perancangan perangkat keras (hardware), hingga proses kalibrasi dan pengujian alat.
    3. Melatih keterampilan dalam menganalisis data hasil pengukuran dan mengevaluasi tingkat akurasi alat, sehingga dapat meningkatkan kemampuan problem solving dan critical thinking di bidang teknik.
    4. Sebagai sarana untuk menerapkan teori-teori yang diperoleh selama perkuliahan ke dalam proyek nyata (*real project*) yang aplikatif dan bermanfaat di dunia industri maupun riset.

c. Bagi Institusi:

1. Dapat dijadikan sebagai referensi atau bahan ajar bagi mahasiswa lain yang ingin mengembangkan penelitian sejenis di bidang sensor, mikrokontroler, dan sistem navigasi dalam ruangan (*indoor positioning system*).

Bab II
Gambaran Umum Perusahaan

2.1. Universitas Ma Chung

Universitas Ma Chung merupakan universitas swasta di Indonesia yang memiliki empat fakultas yaitu Fakultas Teknologi dan Desain, Fakultas Ilmu Kedokteran, Fakultas Ekonomi dan Bisnis, dan Fakultas Bahasa yang terletak di Villa Puncak Tidar N-01, Karangwidoro, Kecamatan Dau, Kota Malang, Jawa Timur, Indonesia. Universitas Ma Chung didirikan pada tanggal 7 Juli 2007 dan dinaungi oleh Yayasan Harapan Bangsa Sejahtera.

Visi dari Universitas Ma Chung adalah Memuliakan Tuhan melalui akhlak, pengetahuan, dan kontribusi nyata sebagai insan akademik yang berdaya cipta. Sedangkan, misi dari Universitas Ma Chunga yaitu:

1. Menyelenggarakan Tri Dharma Perguruan Tinggi yaitu pendidikan dan pengajaran tinggi, penelitian, dan pengabdian kepada masyarakat secara berkualitas, fokus, dan sesuai dengan kebutuhan masyarakat kini dan akan datang.
2. Membentuk dan mengembangkan angkatan-angkatan motivator dan pemimpin masyarakat yang memiliki potensi dan kapasitas moral yang luhur, berjiwa kepemimpinan dan kewirausahaan yang bertitik berat pada pembentukan akhlak dan kepribadian unggul, rendah hati, melayani, dan berkontribusi sebagai manusia yang utuh
3. Mendorong dan mengembangkan sikap serta pemikiran yang kritis-prinsipil dan kreatif-realistis berdasarkan kepekaan hati nurani yang luhur.
4. Menghasilkan lulusan siap pakai yang berkualitas tinggi yang mampu bersaing di pasar global.
5. Berperan aktif dalam meningkatkan peradaban dunia dengan menghasilkan lulusan yang berwawasan global, toleran, dan cinta damai, serta produktif dalam menghasilkan karya cipta yang mendukung peningkatan martabat manusia globa

6. Melaksanakan pengelolaan perguruan tinggi berdasarkan prinsip ekonomis dan akuntabilitas.

Pada saat ini, Universitas Ma Chung memiliki 11 program studi yaitu Manajemen Bisnis, Akuntansi, Magister Manajemen Inovasi, Sastra Inggris, Pendidikan Bahasa Mandarin, Teknik Informatika, Sistem Informasi, Desain Komunikasi Visual, Teknik Industri, Farmasi dan Optometri.

2.2. Program Studi Teknik Informatika

Program studi Teknik Informatika adalah salah satu studi di Universitas Ma Chung yang berfokus pada pembelajaran teknologi komputer dan praktik penggunaan teknologi itu sendiri untuk memenuhi kebutuhan industri teknologi saat ini, program studi Teknik Informatika Universitas Ma Chung saat ini berakreditasi B berdasarkan Surat Keputusan Badan Akreditasi Nasional Perguruan Tinggi (BANPT) Nomor 0356/SK/BANPT/Akred/S/IV/2016, tanggal 28 April 2016. Program studi ini menawarkan pendidikan yang berkualitas dan didukung oleh fasilitas yang lengkap untuk menunjang kegiatan belajar mengajar. Dalam mengembangkan minat dan kemampuan mahasiswanya program studi ini memberikan kesempatan bagi mahasiswa untuk melakukan praktik kerja lapangan di Perusahaan ataupun di pusat studi kampus serta kegiatan Merdeka Belajar Kampus Merdeka (MBKM). Hal ini bertujuan untuk memberikan pengalaman praktik kepada mahasiswa.

Di program studi Teknik informatika terdapat 2 bidang konsentrasi yang dapat diikuti oleh mahasiswa Teknik informatika yaitu Sistem Cerdas yang berfokus pada pembelajar simulasi kecerdasan buatan dalam berbagai mesin (Artificial Intelligence) dan Sistem Komputer yang berfokus pada perkembangan Interner of Things dan Embedded System dalam kehidupan sehari-hari. Praktik kerja lapangan ini merupakan bagian peminatan sistem komputer di Universitas Ma Chung.

2.3. Pusat Studi Human-Machine Interaction Ma Chung

Pusat studi Human-Machine Interaction Ma Chung dibentuk pada 11 September 2019. Pusat studi ini dinaungi oleh Program Studi Teknik Informatika yang bertujuan untuk menunjang pengembangan teknologi dan penerapanya pada bidang interaksi

manusia dan aspek interaksinya.Pusat Studi Human-Machine Interaction sendiri berlokasi di Gedung RnD lantai 6 Universitas Ma Chung, Beberapa bidang kajian pada pusat studi ini diantaranya:

a) Machine vision for human welfare and human-natural interactions

Bidang ini menjalankan penelitian tentang pengaplikasian visi mesin untuk membantu kesejahteraan manusia ataupun interaksi alami manusia.

b) Human-computer interactions (HCI)

Bidang ini menjalankan penelitian tentang desain, implementasi dan evaluasi perangkat atau User Interface (UI) dan User Experience (UX).

c) Robotics technology and mobile apps for human welfare

Bidang ini menjalankan penelitian tentang aplikasi robotika dalam kesejahteraan manusia.

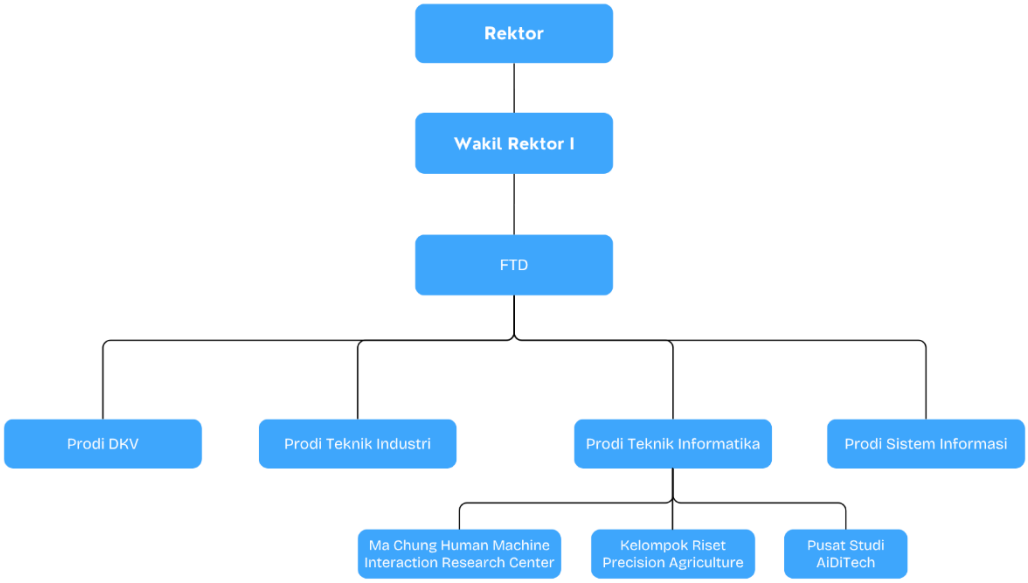
Pusat Studi Human-Machine Interaction memiliki visi dan misi, sebagai berikut:

a) Visi

Pada tahun 2025 menjadi Program Studi Teknik Informatika aras utama di Indonesia Timur yang mendukung eksplorasi sumber daya alam beserta pengelolaan bisnisnya sebagai perwujudan memuliakan Tuhan dan berkontribusi nyata bagi kesejahteraan masyarakat.

b) Misi

1. Menyelenggarakan pengajaran, penelitian dan pengabdian kepada masyarakat yang berfokus pada pengembangan ilmu-ilmu teknik informatika untuk pengelolaan sumberdaya alam dan bisnis.
2. Membentuk dan mengembangkan angkatan-angkatan motivator yang mempunyai jiwa pemimpin dan wirausahawan dengan bertitik berat pada pembentukan akhlak, bersikap rendah hati, dan berjiwa melayani.
3. Membentuk lulusan siap pakai yang berkualitas tinggi dan mampu bersaing pada pasar teknologi informasi global.
4. Menyelenggarakan Program Studi dengan tata kelola yang baik.



Gambar 2. 1 Diagram bidang kajian riset dan pusat studi

Bab III
Tinjauan Pustaka

3.1. Tunanetra dan Low Vision Impersial

Tunanetra adalah keadaan penglihatan seseorang yang mengalami gangguan penglihatan sehingga individu tersebut mengalami hambatan dalam mempersepsi visual atau bahkan tidak bisa melihat sama sekali. Secara umum individu tunanetra dibedakan menjadi dua kategori, yaitu Tunanetra total (total blind atau tidak dapat melihat sama sekali) dan Tunanetra yang masih memiliki penglihatan tetapi tidak sebaik manusia pada umumnya yang dikenal juga sebagai *Low Vision.* Individu dengan tunanetra mengalami kesulitan dalam melihat objek, membaca tulisan, atau mengenali lingkungan sekitarnya tanpa bantuan.

Menurut *World Health Organization* (2019) seseorang dikategorikan sebagai low vision jika ketajaman penglihatannya kurang dari 6/18 namun lebih baik dari 3/60, atau apabila sudut pandangnya kurang dari 20 derajat, meskipun sudah menggunakan koreksi maksimal.

Gambar 3. 1 Penglihatan orang low vision (Sumber: visionservealliance.org)

Seperti ditunjukkan pada Gambar 3.1, penglihatan seorang dengan low vision menunjukkan penurunan ketajaman, kontras, atau penyempitan bidang pandang, yang berdampak pada kemampuan orientasi visual.

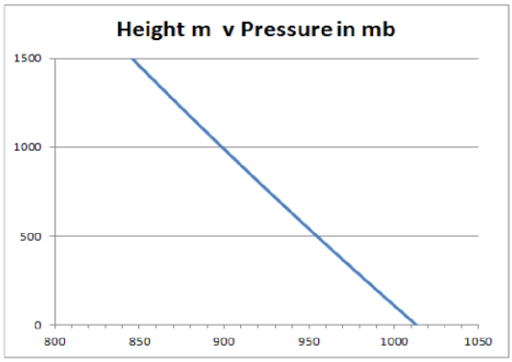
Gangguan penglihatan timbul dari berbagai penyebab. Menurut Flaxman et al. (2017), penyebab utama low vision adalah kesalahan refraksi yang tidak dikoreksi (42%)

diikuti oleh katarak (33%), serta berbagai penyakit mata lain seperti glukoma (2%), degenerasi *macula* (1%), dan *retinopati diabetic* (1%).

Secara global, diperkirakan terdapat sekitar 253 juta orang yang mengalami gangguan penglihatan, dimana 36 juta di antaranya mengalami kebutaan total, dan lebih dari 217 juta lainnya memiliki *low vision* (Holden et al., 2016). Sebagian besar penderita gangguan penglihatan berasal dari negara berkembang.

3.2. Barometer

Barometer adalah instrumen yang digunakan untuk mengukur tekanan atmosfer, yaitu tekanan yang dihasilkan oleh kolom udara yang berada di atas permukaan bumi. Tekanan atmosfer ini berbanding terbalik dengan ketinggian, semakin tinggi suatu tempat dari permukaan laut semakin rendah tekanan atmosfer yang dirasakan (Si et al., 2023).

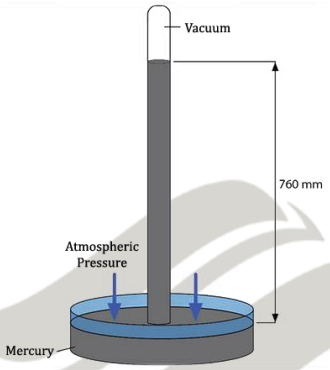


Gambar 3. 2 Pressure vs Height (Sumber: Binghao Li et al., 2013)

Konsep tekanan atmosfer pertama kali dijelaskan oleh Evangelista Torricelli pada tahun 1643, melalui percobaan dengan tabung merkuri yang menghasilkan barometer raksa pertama. Eksperimen tersebut menunjukkan bahwa tekanan udara dapat diestimasi secara tidak langsung dengan mengamati tinggi kolom cairan (Torricelli, 1644 dalam Si et al., 2023). Barometer terbagi dalam beberapa jenis barometer, yaitu barometer raksa, barometer aneroid, dan barometer digital.

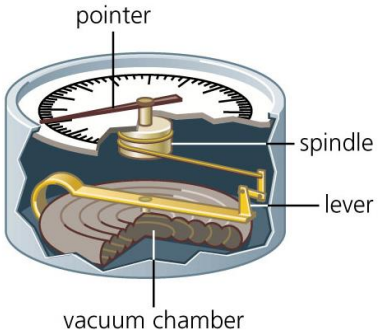
Barometer raksa (Mercury Barometer) bekerja dengan memanfaatkan kolom raksa dalam sebuah tabung kaca yang salah satu ujungnya tertutup. Tekanan udara akan menekan permukaan raksa di dalam wadah, dan menyebabkan raksa naik ke dalam tabung.

Ketinggian kolom raksa ini mencerminkan besarnya tekanan atmosfer. Nilai standar tekanan atmosfer adalah 760 mmHg atau setara dengan 1013,25 hPa. Gambar 3.3 menunjukkan cara kerja dasar barometer raksa, di mana tekanan udara ditentukan berdasarkan tinggi kolom raksa dalam tabung tertutup.



Gambar 3. 3 Barometer Raksa (Sumber: Study.com)

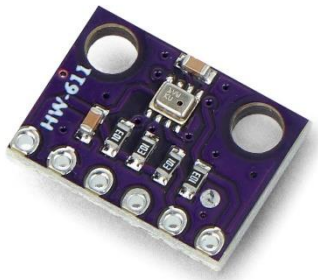
Barometer aneroid merupakan jenis barometer yang mengukur tekanan udara tanpa menggunakan cairan seperti raksa. Istilah "aneroid" berasal dari bahasa Yunani, di mana "a-" berarti "tanpa" dan "neron" berarti "cairan" atau "air", sehingga secara literal berarti "tanpa cairan". Gambar 3.4 menampilkan komponen utama dari barometer aneroid, yang meliputi kapsul hampa udara, pegas, sistem tuas dan roda gigi, serta penunjuk skala. Saat tekanan atmosfer berubah, kapsul akan mengalami deformasi, dan gerakan ini diteruskan melalui mekanisme tersebut untuk menunjukkan nilai tekanan udara.



Gambar 3. 4 Barometer aneroid (Sumber: emaze.com)

Barometer digital merupakan perangkat pengukur tekanan udara yang memanfaatkan teknologi elektronik untuk mendeteksi, memproses, dan menampilkan tekanan atmosfer dalam bentuk angka digital, biasanya dalam satuan hektopascal (hPa)

atau milibar (mb). Berbeda dengan barometer raksa dan aneroid yang mengandalkan cairan atau mekanisme fisik, barometer digital menggunakan sensor mikro dan mikroprosesor guna mengukur tekanan udara secara otomatis. Alat ini banyak diterapkan dalam teknologi modern seperti stasiun cuaca otomatis, perangkat outdoor seperti jam tangan, ponsel pintar, hingga sistem navigasi vertikal di dalam bangunan (Si et al., 2023; Kusuma et al., 2023).



Gambar 3. 5 Modul barometer digital BMP280 (Sumber: botland.com.pl)

Gambar 3.5 menunjukkan contoh modul barometer digital modern, yang mengintegrasikan sensor tekanan mikro, regulator tegangan, level shifter, dan interface komunikasi digital seperti I²C atau SPI, sehingga mudah digunakan dalam rangkaian mikrokontroler.

Barometer dalam penerapannya memiliki beberapa rumus yang digunakan untuk mengubah nilai tekanan atmosfer menjadi informasi lain yang berguna, rumus-rumus ini merupakan turunan dari prinsip fisika dasar, seperti hukum hidrostatik, hukum gas ideal, dan model atmosfer standar internasional (ISA) (Wallace & Hobbs, 2006). Berikut adalah beberapa rumus yang digunakan dalam penerapan teknologi berbasis barometer:

Persamaan barometric umum, digunakan untuk menghitung tekanan udara pada ketinggian tertentu dalam kondisi atmosfer dengan penurunan suhu linier (*lapse rate*).

$$p = p_0 \cdot \left(1 - \frac{T}{L \cdot h}\right)^{\frac{R \cdot L}{g \cdot M}} \tag{3.1}$$

Dimana p adalah tekanan pada ketinggian h (Pa atau hPa), p_0 adalah tekanan referensi (biasanya di permukaan laut), h ketinggian relatif (m), L adalah *lapse rate* atau

laju penurunan suhu (0.0065 K/m, T merujuk pada suhu absolut di permukaan (K), g adalah gravitasi (9.80665 m/s²), kemudian R adalah konstanta gas (8.314 J/mol·K), dan M merujuk pada massa molar udara (0.029 kg/mol).

Kemudian rumus barometric isotermal (Pendekatan Eksponensial) yaitu jika suhu dianggap konstan (isotermal), rumus dapat disederhanakan dalam bentuk eksponensial (Bai, 2014).

$$p \approx p_0 \cdot exp\left(1 - \frac{T}{L \cdot h}\right)^{\frac{R \cdot L}{g \cdot M}} \quad (3.1a)$$

Selanjutnya, rumus Invers digunakan ketika tekanan udara di dua titik tidak diketahui, untuk menghitung ketinggian vertikal antara keduanya. Digunakan pada sistem altimeter dan sensor tekanan digital (Bai, 2014).

$$h = \frac{R \cdot T}{g \cdot M} \cdot \ln\left(\frac{p_0}{p}\right) \quad (3.2)$$

Kemudian, rumus penyederhanaan praktis digunakan untuk konversi tekanan ke ketinggian atau koreksi barometrik. Nilai 44330 muncul dari pengelompokan konstanta ($R \cdot T/(M \cdot g)$) dalam kondisi atmosfer standar (T ≈ 288 K) (Kusuma et al., 2023).

$$h = 44330 \cdot \left(1 - \left(\frac{P}{P_0}\right)^{0.1903}\right) \quad (3.3)$$

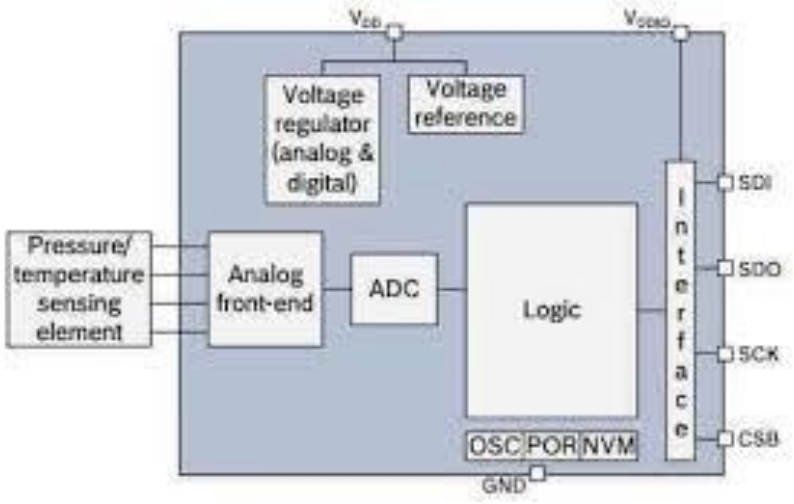
Dalam penelitian ini, peneliti menggunakan rumus persamaan (3.2) yaitu bentuk logaritmik dari persamaan berometrik, yang merujuk pada pendekatan yang digunakan dalam penelitian Bai (2014), yaitu memperkirakan perbedaan ketinggian antar lantai berdasarkan perbedaan tekanan atmosfer secara real time. Bai dalam disertasinya mengembangkan sistem navigasi indoor untuk penyandang tunanetra dengan menggunakan sensor barometer dan menghindari penggunaan ketinggian absolut karena fluktuasi tekanan udara yang besar terhadap waktu.

3.3. Sensor Barometer BMP280

Sensor barometer merupakan salah satu jenis sensor lingkungan yang berfungsi untuk mengukur tekanan atmosfer di sekitar perangkat. Prinsip kerjanya didasarkan pada perubahan tekanan udara pada berbagai ketinggian. Informasi tekanan ini tidak hanya berguna untuk memantau kondisi cuaca, tetapi juga dapat digunakan untuk memperkirakan ketinggian atau perubahan elevasi (altitude), sehingga sensor barometer banyak diterapkan dalam sistem navigasi, pemetaan digital, wearable devices, serta alat bantu bagi penyandang disabilitas.

Salah satu sensor barometer yang banyak digunakan dalam sistem tertanam adalah BMP280, yaitu sensor tekanan digital buatan perusahaan Bosch Sensortec. Sensor ini dirancang khusus untuk aplikasi mobile dan IoT, dengan konsumsi daya rendah dan akurasi tinggi. Selain mengukur tekanan udara, BMP280 juga dapat mengukur suhu lingkungan, sehingga menjadikannya solusi serbaguna untuk pengukuran lingkungan fisik (Bosch, 2020).

Sensor barometer merupakan sensor buatan perusahaan Bosch yang digunakan untuk mengukur suhu dan tekanan atmosfer, sensor ini juga dapat digunakan untuk mengukur altitude berdasarkan tekanan atmosfernya. Sensor barometer BMP280 dapat mengukur relatif akurasi dengan akurasi kurang lebih 0.12 hPa setara dengan perbedaan 1 meter pengukuran altitude (Bosch, 2020), berikut *key parameter* dari BMP280:



Tabel 3. 1 Key Parameter BMP280

Parameter	Nilai/Rentang	Keterangan tambahan
Pressure Range	300 … 1100 hPa	Equiv. to +9000 m *up to* -500 m dpl.
Package	8-pin *LGA metal lid*	*Footprint*: 2.0 × 2.5 mm², *height*: 0.95 mm
Relative Accuracy	±0.12 hPa (equiv. to ±1 m)	*Valid for* 700–900 hPa @25°C
Absolute Accuracy	±1 hPa (*typical*)	*Valid for* 950–1050 hPa, 0–40°C
Temp. Coefficient Offset	1.5 Pa/K (equiv. to 12.6 cm/K)	*Valid for* 25–40°C @900 hPa
Digital Interfaces	I²C (*up to* 3.4 MHz), SPI (3/4 *wire*, 10 MHz)	-
Current Consumption	2.7 µA @ 1 Hz *sampling rate*	-
Temperature Range	-40 ... +85 °C	-
Compliance	*RoHS compliant, halogen-free*	*MSL* 1 (*moisture sensitivity level*)

Gambar 3. 6 Diagram blok BMP280 (Sumber: Arduino, 2024)

Sensor BMP280 merupakan pilihan populer dalam sistem monitoring lingkungan karena bentuknya yang kecil, konsumsi daya rendah, dan kompatibilitas tinggi. Sensor ini sangat efektif digunakan dalam proyek seperti deteksi ketinggian antar lantai, sistem wearable untuk tunanetra, serta alat pemantau tekanan udara berbasis Arduino.

3.4. Arduino Nano

Arduino Nano adalah *board* pengembangan yang dirancang khusus untuk mempercepat pembuatan prototipe dengan dimensi kecil. Arduino nano berbasis

ATmega328P (untuk versi klasik) atau ATmega4809 (untuk versi Nano Every). *Board* sisini dirancang untuk penggunaan breadboard, dengan dimensi hanya sekitar 45mm x 18 mm, sehingga cocok digunakan dalam proyek-proyek sistem tertanam (*embedded system*) yang memerlukan ukuran kecil (Arduino, 2024)

Secara teknis, Arduino nano memiliki spesifikasi sebagai berikut:

Tabel 3. 2 Arduino Nano Spesifikasi

Parameter Spesifikasi	**Parameter Spesifikasi**
Mikrokontroler ATmega328P	Mikrokontroler ATmega328P
Tegangan Operasi 5V	Tegangan Operasi 5V
Tegangan Input (rekomendasi) 7–12V	Tegangan Input (rekomendasi) 7–12V
Jumlah Digital I/O 14 pin (6 PWM)	Jumlah Digital I/O 14 pin (6 PWM)
Jumlah Analog Input 8 pin (A0–A7)	Jumlah Analog Input 8 pin (A0–A7)
Clock Speed 16 MHz	Clock Speed 16 MHz
Flash Memory 32 KB (2 KB digunakan bootloader)	Flash Memory 32 KB (2 KB digunakan bootloader)
SRAM 2 KB	SRAM 2 KB
EEPROM 1 KB	EEPROM 1 KB
Antarmuka Komunikasi UART, SPI, I2C	Antarmuka Komunikasi UART, SPI, I2C
Port USB Mini USB (Tipe B)	Port USB Mini USB (Tipe B)

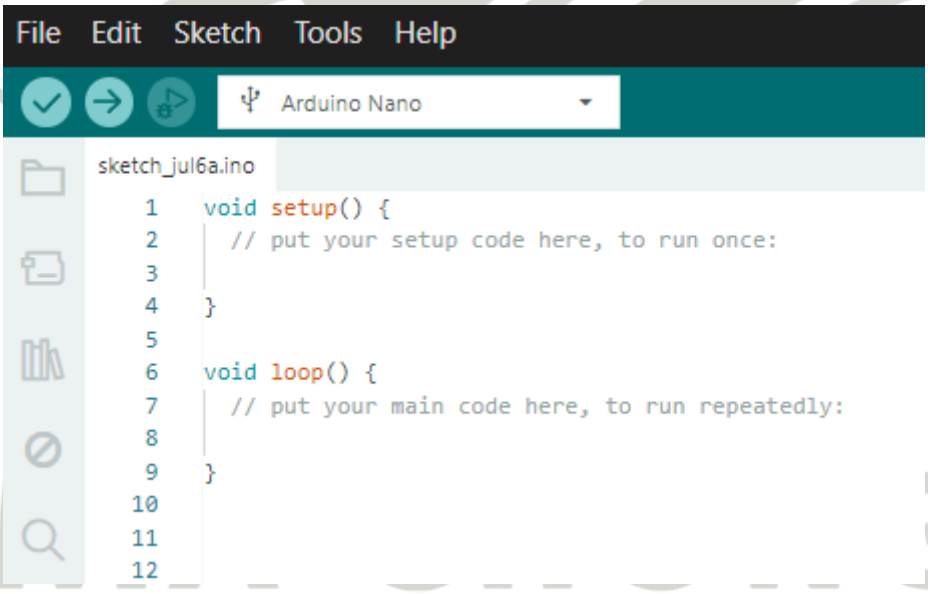
Arduino nano dapat di program menggunakan Arduino IDE melalui koneksi USB *Bootloader* memungkinkan pemrograman tanpa perlu programmer eksternal. Karena kompatibel dengan pustaka (*library*), Arduino Nano mendukung integrasi dengan beragam sensor dan aktuator, menjadikannya sangat fleksibel untuk prototipe maupun implementasi sistem kontrol otomatis dan *Internet of Things* (IoT) (Arduino, 2024).

Selain itu, Arduino Nano memiliki chip USB-to-Serial (CH340) yang menghubungkan antarmuka UART mikrokontroler ke port USB. Dengan adanya chip ini,

pengguna tidak memerlukan converter USB to Serial eksternal untuk melakukan pemrograman atau komunikasi serial dengan komputer. Hal ini menjadikan Arduino Nano lebih praktis dan langsung siap digunakan untuk pemrograman melalui kabel USB Mini-B.

3.5. Arduino IDE

Arduino IDE (Integrated Development Environment) adalah perangkat lunak resmi yang digunakan untuk menulis, mengompilasi, dan menguggah program ke papan pengembangan Arduino, seperti Arduino UNO, Nano, Mega, dan lainnya. Arduino IDE menyediakan antarmuka yang sederhana dan mudah digunakan oleh pemula maupun profesional dalam pengembangan sistem berbasis mikrokontroler.



Gambar 3. 7 Tampilan awal Arduino IDE

Arduino IDE mendukung bahasa pemrograman berbasis C/ C dan dilengkapi dengan berbagai pustaka (library) yang memudahkan integrasi dengan detector, aktuator, dan modul komunikasi. IDE ini menyediakan fitur periodical examiner, board director, dan library director, yang memudahkan debugging dan pengelolaan perangkat keras.

3.6. Library

Dalam ekosistem Arduino, *library* (Pustaka) adalah Kumpulan fungsi dan definisi yang digunakan untuk mempermudah pemrograman perangkat keras seperti sensor, aktuator, dan modul komunikasi. Library Arduino ditulis dalam bahasa C atau C++ dan disusun dalam struktur folder yang berisi file header (.h) dan implementasi (.cpp).

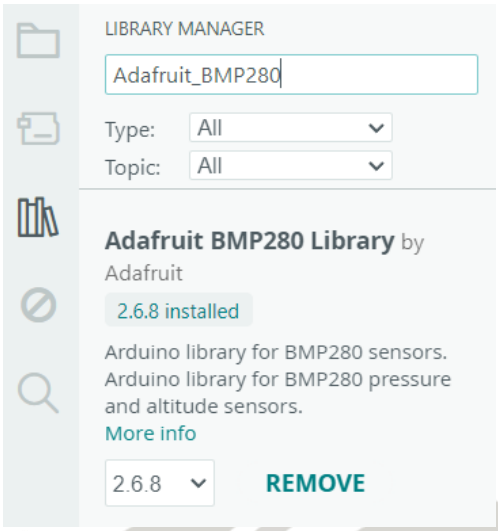
Library sangat membantu dalam menyederhanakan pemrograman. Daripada menulis semua kode dari awal, pengguna cukup memanggil fungsi yang telah disediakan. Misalnya, dalam penelitian ini menggunakan library Adafruit _BMP280, library Wire dan library EEPROM, cukup mengimpor library lalu memanggil metode di dalam library tersebut.

3.6.1. Library AdaFruit_BMP280

Adafruit_BMP280 adalah library yang dikembangkan oleh Adafruit Industries untuk mempermudah penggunaan sensor tekanan dan suhu BMP280 pada papan pengembangan Arduino. Sensor BMP280 yang digunakan untuk mengukur tekanan barometrik dan suhu, dan sering digunakan dalam aplikasi seperti pengukuran ketinggian (Altimeter), cuaca dan sistem navigasi.

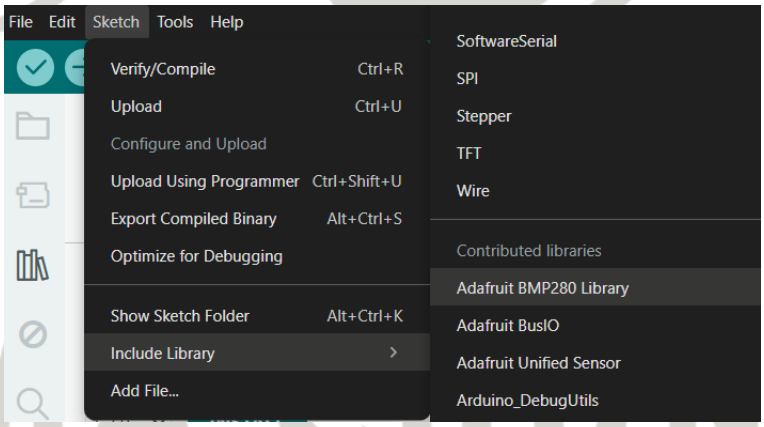
Library ini menyediakan fitur utama seperti menginisialisasi sensor, membaca suhu, dan membaca tekanan dengan mudah. Menggunakan Adafruit_BMP280 lebih praktis dibanding menulis sendiri komunikasi I2C atau SPI, karena bisa mempercepat proses pembuatan program dan mengurangi risiko kesalahan dalam penulisan kode.

Untuk dapat menggunakan library ini maka harus dilakukan tahap instalasi library, dan kemudian impor library ke program, seperti contoh berikut:



Gambar 3. 8 Instalasi library Adafruit_BMP280

Seperti yang terlihat pada gambar 3.8, Langkah-langkah instalasi library yang dilakukan yaitu buka Arduino IDE, kemudian masuk pada bagian Library Manager dan ketikan nama library Adafruit_BMP280 pada kolom pencarian, setelah itu klik install pada versi yang akan digunakan



Gambar 3. 9 Import library Adafruit_bmp280

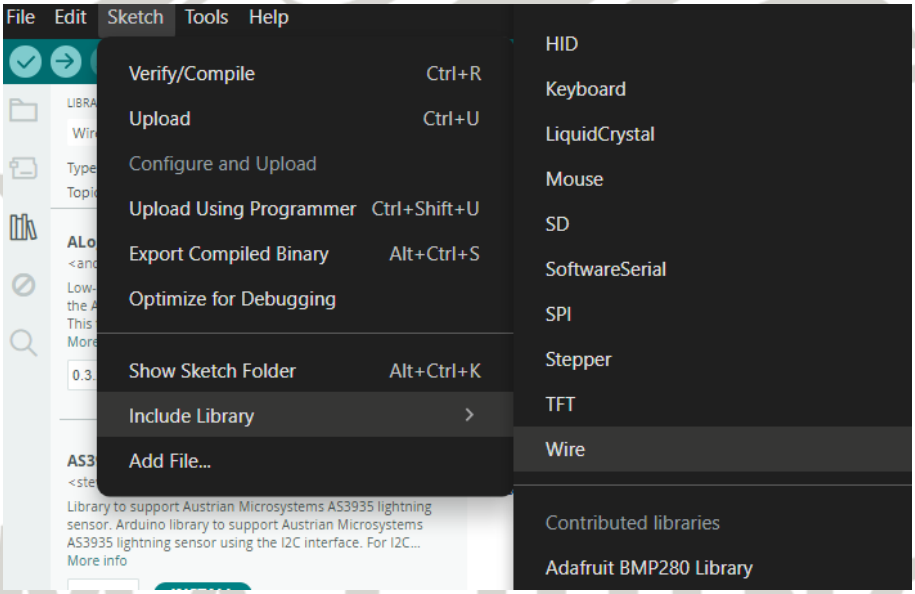
Pada gambar 3.9, buka Sketch, kemudian Include Library, cari Adafurit BMP280 Library, klik maka library diimpor ke dalam program sehingga library dapat digunakan.

3.6.2. Library Wire

Wire.h adalah library standar dari Arduino yang digunakan untuk menghubungkan Arduino dengan perangkat lain menggunakan protokol I²C. Dengan library ini, proses komunikasi data jadi lebih mudah tanpa perlu menulis semua kodenya

dari awal. Protokol ini memungkinkan pertukaran data antara mikrokontroler dan berbagai perangkat seperti sensor, memori, serta modul lainnya menggunakan dua jalur utama, yaitu SDA (data) dan SCL (clock). Pada Arduino Nano, pin A4 berfungsi sebagai SDA dan A5 sebagai SCL. Dengan memanfaatkan library Wire, pengguna dapat dengan mudah melakukan komunikasi data antar perangkat melalui fungsi-fungsi yang telah disediakan, tanpa perlu menulis seluruh protokol secara manual.

Library ini sangat berguna dalam proyek yang melibatkan sensor berbasis I²C seperti BMP280, karena memungkinkan pengiriman dan penerimaan data melalui metode yang efisien dan stabil. Biasanya, Wire.h digunakan bersama dengan library lain seperti Adafruit_BMP280 untuk memproses data sensor lebih lanjut. Library Wire sendiri sudah tersedia tanpa harus mengunduh lagi karena *library* bawaan Arduino IDE.



Gambar 3. 10 Import library Wire

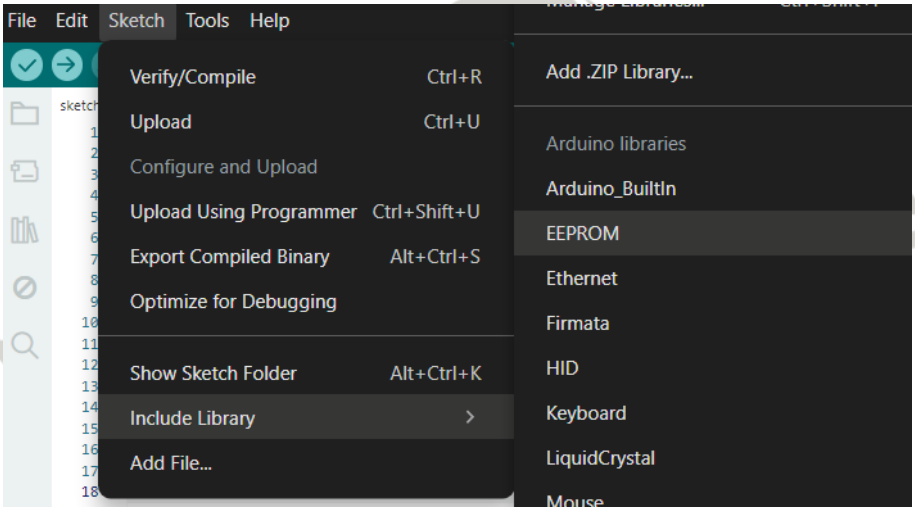
Pada gambar 3.10, buka Sketch, kemudian Include Library, cari *Wire Library*, klik maka library diimpor ke dalam program sehingga library dapat digunakan.

3.6.3. EEPROM

Library EEPROM.h merupakan pustaka bawaan Arduino yang digunakan untuk mengakses dan memanipulasi EEPROM (Electrically Erasable Programmable Read-Only Memory) pada mikrokontroler. EEPROM adalah jenis memori non-volatile, artinya data yang disimpan tidak akan hilang meskipun perangkat dimatikan. Ini sangat berguna

untuk menyimpan data penting seperti konfigurasi, kalibrasi sensor, atau nilai-nilai pengaturan yang ingin dipertahankan antar sesi pemakaian.

Arduino Nano (dengan ATmega328P) memiliki EEPROM internal sebesar 1 KB (1024 byte). Library EEPROM.h menyediakan fungsi-fungsi sederhana untuk membaca dan menulis data byte demi byte pada alamat memori tertentu. Penggunaan EEPROM perlu dilakukan dengan hati-hati karena memiliki batas siklus tulis (sekitar 100.000 kali tulis per lokasi). Oleh karena itu, pengelolaan penulisan data ke EEPROM harus efisien agar tidak memperpendek umur memori.



Gambar 3. 11 Import Library EEPROM

Pada gambar 3.11, buka Sketch, kemudian Include Library, cari EEPROM *Library*, klik maka library diimpor ke dalam program sehingga library dapat digunakan.

3.7. Visual Studio

Visual Studio merupakan salah satu lingkungan pengembangan terintegrasi (IDE) yang dikembangkan oleh Microsoft untuk membangun berbagai jenis aplikasi, termasuk desktop, web, cloud, dan perangkat tertanam. Visual Studio mendukung sejumlah bahasa pemrograman seperti C#, C++, dan VB.NET, serta dilengkapi dengan fitur seperti IntelliSense, debugging interaktif, dan desain antarmuka grafis. Dalam proyek berbasis mikrokontroler seperti Arduino, Visual Studio dapat digunakan untuk membangun antarmuka pengguna (GUI) yang berfungsi sebagai alat pemantau dan pengontrol perangkat. Dalam penelitian ini menggunakan C#, membuat program untuk menerima

bacaan sensor melalui port serial, sehingga komunikasi data antara komputer dan perangkat dapat berlangsung secara real-time dan ditampilkan di terminal.

3.7.1. C#

C# (C- Sharp) adalah bahasa pemrograman tingkat tinggi yang dikembangkan oleh Microsoft dan menjadi bagian dari platform. NET. Bahasa ini punya sintaks yang mirip dengan C dan C, tapi sudah dilengkapi berbagai fitur ultramodern seperti pengelolaan memori otomatis (scrap collection), keamanan tipe data, dan integrasi yang erat dengan pustaka. NET Framework. C# banyak digunakan untuk membuat aplikasi desktop Windows, aplikasi web, backend garçon, sampai aplikasi mobile lewat Xamarin. Karena sifatnya yang berorientasi objek dan didukung banyak tools pengembangan, C# sering dipilih untuk membangun aplikasi yang butuh tampilan antarmuka yang kompleks atau koneksi ke perangkat keras.

Dalam proyek ini, C# digunakan untuk membuat program antarmuka yang bisa membaca data dari detector yang dikirimkan oleh Arduino lewat komunikasi periodical. Data yang diterima ditampilkan secara langsung (real- time) di outstation, dan juga disimpan secara otomatis dalam format. csv. Dengan bantuan kelas SerialPort dari. NET, program ini bisa menangkap data dari Arduino, menampilkannya di layar, dan menyimpannya baris demi baris ke dalam train log yang bisa dipakai untuk dokumentasi atau analisis selanjutnya.

3.8. PuTTY

PuTTY adalah perangkat lunak *open source* untuk koneksi jaringan menggunakan protokol seperti SSH, Telnet, dan terutama Serial (RS232). PuTTY dapat digunakan untuk memantau komunikasi serial antara komputer dan mikrokontroler, seperti Arduino, melalui port COM. Dalam proyek berbasis Arduino, PuTTY digunakan sebagai alat untuk memantau data serial secara langsung, misalnya untuk melihat output dari sensor, proses debugging, atau komunikasi dua arah secara manual. PuTTY memiliki keunggulan dalam tampilannya yang ringan dan kemudahan pengaturan baudrate, COM port, dan *logging* data.

3.9. Laser Distance Meter

Laser Laser Distance Meter adalah metode pengukuran jarak menggunakan sinar laser yang dipantulkan dari suatu objek. Teknologi ini banyak digunakan dalam sistem pengukuran presisi tinggi seperti alat ukur digital, sistem pemetaan, dan navigasi *robotic*. Sensor pengukur jarak berbasis laser bekerja dengan prinsip *Time of Flight* (ToF), yaitu mengukur waktu tempuh pantulan sinar laser untuk menghitung jarak.

Gambar 3. 12 Laser Distance Meter (Sumber: Tokopedia.com)

Penggunaan sensor laser ini sangat efektif dalam sistem otomatisasi, monitoring lingkungan, serta pengukuran ketinggian atau jarak objek dengan tingkat akurasi tinggi. LDM memiliki beberapa mode yaitu *single measurement* (pengukuran tunggal), *continuous measurement* (pengukuran berkelanjutan), *area measurement* (pengukuran luas), *volume measurement* (pengukuran volume), *indirect measurement* or *Pythagoras mode* dan *direct line measurement* (pengukuran lurus).

Gambar 3. 13 Mode Laser Distance Meter (Sumber: Tokopedia.com)

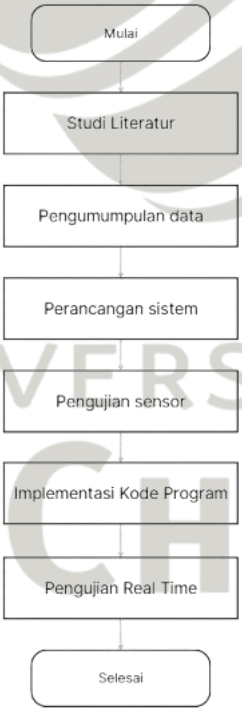
Dalam penelitian ini, digunakan mode pengukuran tidak langsung (*Pythagoras Mode*) untuk menentukan tinggi atau jarak vertikal secara tidak langsung, terutama ketika

sensor tidak dapat diarahkan secara tegak lurus ke objek. Mode ini dipilih karena mampu memberikan estimasi ketinggian dengan bantuan dua sudut pengukuran, yang kemudian dihitung menggunakan rumus Pythagoras. Sebagai verifikasi dan pembanding, mode pengukuran lurus (*Direct Line Measurement*) juga digunakan untuk memastikan keakuratan hasil yang diperoleh dari mode Pythagoras. Hal ini penting agar data ketinggian yang diperoleh memiliki validasi silang, terutama dalam pengukuran antar lantai atau perbedaan elevasi.

Bab IV
Deskripsi Data dan Hasil Praktik Kerja Lapangan

4.1. Tahap pengerjaan

Praktik Kerja Lapangan dilaksanakan di Pusat Studi *Human-Machine Interaction* Universitas Ma Chung. Proyek Praktik Kerja Lapangan ini bertujuan untuk merancang dan mengimplementasikan sistem pengukuran ketinggian berbasis sensor barometrik untuk membantu penyandang tunanetra dan *low vision impaired* dalam mengetahui posisi lantai tempat mereka berada, baik menggunakan tangga maupun elevator di dalam gedung bertingkat. Hasil akhirnya nanti digunakan untuk diterapkan menjadi alat yang dapat digunakan. Gambar 4.1 menunjukkan tahapan pengerjaan yang dilakukan selama Prakti Kerja Lapangan.



Gambar 4. 1 Tahapan Pengerjaan

Proyek ini dilakukan dalam beberapa tahapan, di mulai dari studi literatura untuk mempelajari penelitian serupa tentang penggunaan barometer dalam pengukuran ketinggian, pengumpulan data, perancangan, pengujian sensor, hingga implementasi dan pengujian alat.

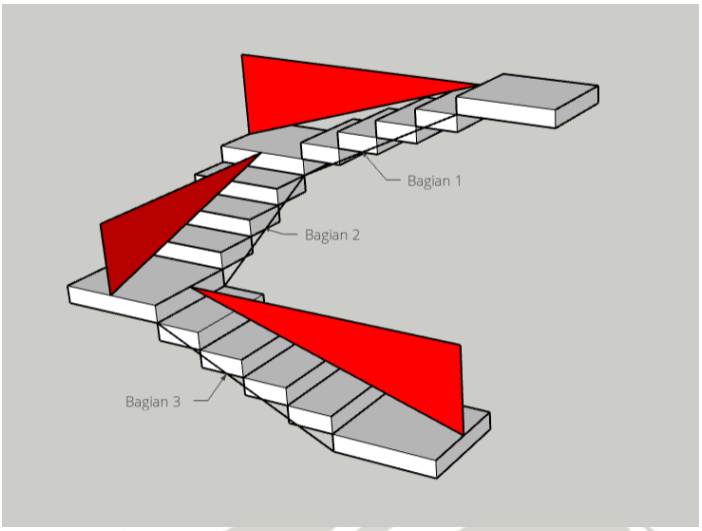
4.2. Pengumpulan Data

Pengumpulan data dilakukan menggunakan sensor BMP280 dan pengukuran langsung menggunakan Laser Distance Meter (LDM). Tujuan utama dari proses ini adalah untuk memperoleh data ground truth ketinggian tiap lantai yang nantinya digunakan sebagai pembanding terhadap hasil pembacaan dari sensor barometrik serta data pendukung lainnya untuk penelitian ini.

4.2.1. Ground Truth Ketinggian Tiap Lantai

Pengukuran tinggi antar lantai dilakukan di Gedung RND Universitas Ma Chung menggunakan alat Laser Distance Meter. Tiap lantai diukur dari lantai dasar hingga lantai enam, dan proses pengukuran diulang sebanyak 5 kali untuk memastikan konsistensi data.

Gambar 4. 2 Contoh pengukuran ketinggian lantai



Gambar 4. 3 Ilustrasi tangga dan pengukuran

Pada proses pengukuran tiap lantai di bagi dalam 3 bagian, karena struktur lantai berbelok seperti pada gambar 4.3 sehingga bagian 1 adalah tangga atas yang menurun, bagian 2 adalah tangga yang turun menyamping dan bagian 3 adalah tangga paling bawah yang menurun berlawan dari arah bagian 1, ditiap bagian lantai tersebut dilakukan pengukuran menggunakan laser distance meter metode phytagoras seperti bagian merah pada gambar 4.3 dan contoh pengukuran dapat dilihat pada gambar 4.2

Tabel 4. 1 Hasil pengukuran Ground Truth ketinggian antar lantai dengan LDM

No Lantai	Bagian Ukur	I (CM)	II (CM)	III (CM)	IV (CM)	V (CM)	Rata-rata (CM)	Standar Deviasi	Tinggi (CM)
Lantai 5	1	148.9	148.9	148.9	148.9	148.9	148.9	0.00	
Ke 6	2	148.7	148.8	148.8	148.8	148.8	148.78	0.45	
	3	130.8	130.6	130.8	130.8	130.7	130.74	0.89	428.42
Lantai 4	1	144.5	144.6	144.6	144.7	144.7	144.62	0.84	
Ke 5	2	148.4	148.6	148.5	148.6	148.5	148.52	0.84	
	3	132.9	133.0	133.1	133.1	133.0	133.02	0.84	426.16
Lantai 3	1	146.0	146.0	146.0	146.1	146.1	146.04	0.55	
Ke 4	2	144.3	144.5	144.2	144.4	144.2	144.32	1.30	

No Lantai	Bagian Ukur	I (CM)	II (CM)	III (CM)	IV (CM)	V (CM)	Rata-rata (CM)	Standar Deviasi	Tinggi (CM)
	3	131.6	131.4	131.5	131.4	131.4	131.46	0.89	421.82
Lantai 2	1	147.2	147.1	147.1	147.3	147.1	147.16	0.90	
Ke 3	2	145.8	145.9	145.7	145.7	145.8	145.78	0.84	
	3	136.0	136.0	136.1	136.3	136.2	136.12	1.30	429.06
Lantai 1	1	148.6	148.7	148.7	148.6	148.8	148.68	0.84	
Ke 2	2	146.0	146.0	145.9	146.0	145.2	145.82	3.42	
	3	132.8	132.8	132.9	132.8	132.8	132.82	0.45	427.32
Lantai 0	1	117.6	117.7	117.8	117.7	117.6	117.68	0.84	
Ke 1	2	98.4	98.6	98.4	98.4	98.5	98.46	0.89	
	3	138.8	138.9	138.8	138.9	138.8	138.84	0.55	354.98

Tabel 4.1 menampilkan hasil pengukuran di setiap bagian ukur yang dilakukan sebanyak lima kali, kemudian dihitung nilai rata-rata dan standar deviasi untuk menilai presisi dan stabilitas data pengukuran. Pengukuran dilakukan dari lantai 6 hingga lantai dasar. Pengukuran dilakukan pada dua lantai yang berdekatan (misalnya, Lantai 6 ke 5), kemudian selisih dari hasil pengukuran tersebut dihitung dan digunakan untuk memperoleh nilai tinggi aktual antar lantai.

Nilai rata-rata menunjukkan hasil pengukuran rata-rata dari kelima titik ukur, sedangkan standar deviasi menunjukkan tingkat variasi antar pengukuran. Semakin kecil nilai standar deviasi, maka semakin konsisten hasil pengukuran yang diperoleh.

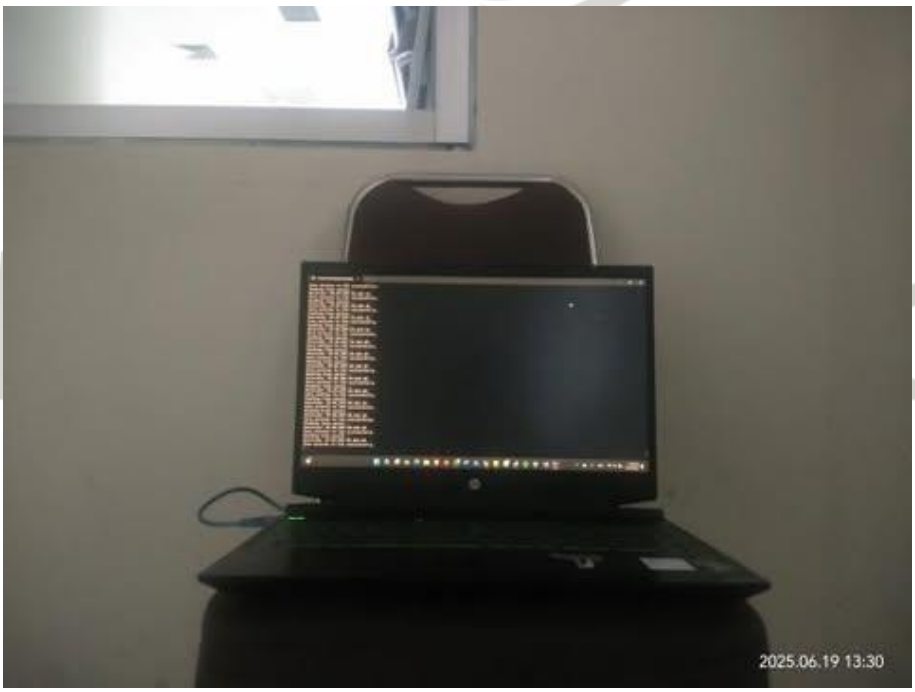
Berdasarakn hasil dari table 4.1 tinggi antar lantai pada bangunan ini berada dalam rentang 421.82 cm hingga 429.06 cm, kecuali untuk lantai 1 ke lantai dasar yang hanya memiliki ketinggian 354.98 cm. Selain itu, hasil juga menunjukkan bahwa sebagian besar lantai bangunan memiliki ketinggian yang relatif seragam, yaitu sekitar 4.2 meter. Hal ini penting untuk memastikan bahwa perancangan sistem deteksi lantai dapat memanfaatkan perbedaan ketinggian ini sebagai parameter utama dan standar deviasi pada setiap pengukuran berkisar antara 0.45 hingga 1.30 cm, menandakan bahwa hasil pengukuran

cukup presisi dan dapat dijadikan ground truth yang valid untuk proses kalibrasi sistem identifikasi lantai berbasis sensor.

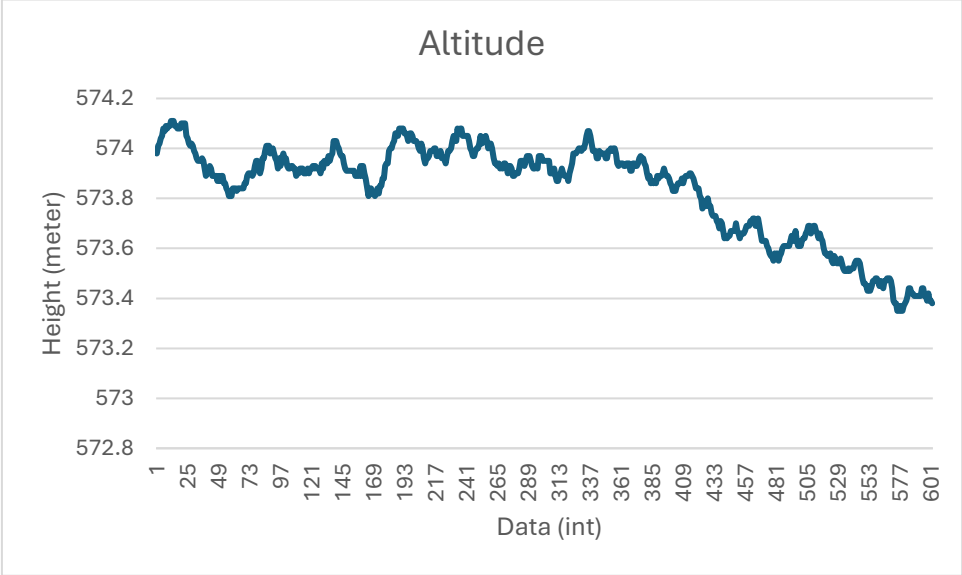
Oleh karena itu dapat disimpulkan bahwa bangunan ini memiliki struktur ketinggian lantai yang stabil dan konsisten, dengan pengecualian pada lantai dasar ke lantai satu yang memiliki tinggi lebih pendek dibandingkan lantai-lantai lainnya. Perbedaan ini perlu diperhatikan dalam pengembangan sistem karena dapat memengaruhi proses identifikasi lantai berbasis perbedaan tekanan atau ketinggian.

4.2.2. Pengamatan Penggunaan Ketinggian Absolut

Selanjutnya dilakukan pengumpulan data dengan mengamati hasil pengukuran ketinggian absolut dari bacaan sensor menggunakan fungsi bawaan library Adafruit_BMP280 yaitu bmp.realAltitude() yang menggunakan rumus standar barometrik yaitu rumus (3.3) dengan tekanan permukaan laut konstan (P0 = 1013.25 hPa). Tujuannya adalah untuk menilai apakah nilai ketinggian absolut dapat digunakan secara langsung untuk estimasi posisi lantai. Gambar 4.4 memperlihatkan grafik bacaan sensor terhadapa ketinggian absolut.



Gambar 4. 4 Contoh Pengambilan Data



Gambar 4. 5 Grafik ketinggian absolut saat diam

Berdasarkan hasil pengamatan, terlihat bahwa nilai ketinggian absolut yang dihasilkan oleh sensor mengalami fluktuasi yang cukup signifikan, meskipun sensor berada dalam posisi diam selama kurang lebih 6 menit waktu pengambilan data.

Ketinggian absolut dalam konteks ini merujuk pada estimasi ketinggian yang dihitung langsung oleh sensor barometrik berdasarkan nilai tekanan udara yang terukur, menggunakan rumus konversi tekanan ke ketinggian berdasarkan standar atmosfer. Nilai ini bersifat langsung dan tidak dikalibrasi terhadap kondisi lingkungan sekitar atau lokasi tertentu.

Namun, selama pengukuran diam ini, nilai ketinggian absolut ternyata tidak stabil. Terjadi fluktuasi, yaitu perubahan nilai naik-turun secara terus-menerus, meskipun posisi sensor tidak bergerak. Fluktuasi ini kemungkinan besar disebabkan oleh berbagai faktor lingkungan seperti:

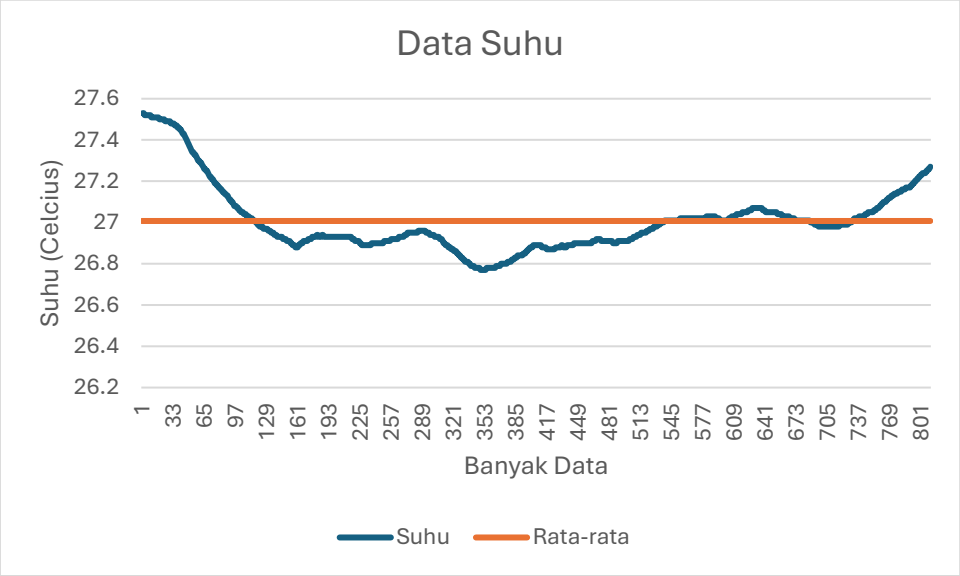
- Perubahan sirkulasi udara di sekitar sensor,
- Variasi suhu ruangan yang memengaruhi densitas udara,
- Ketidakstabilan tekanan lokal, bahkan akibat hembusan angin dari ventilasi atau perubahan kecil di lingkungan indoor.

Kondisi ini menunjukkan bahwa pembacaan ketinggian absolut dari sensor barometer kurang andal jika digunakan secara langsung untuk mendeteksi posisi vertikal (lantai) dalam bangunan.

Oleh karena itu, dalam pengembangan sistem identifikasi lantai ini, digunakan pendekatan ketinggian relatif, yaitu menghitung selisih tekanan antara posisi saat ini dan titik referensi tekanan. Pendekatan ini jauh lebih stabil dan minim fluktuasi, karena mengacu pada perubahan tekanan secara relatif, bukan nilai absolut yang rentan dipengaruhi faktor luar.

4.2.3. Pengamatan perubahan suhu

Dalam pengukuran ini, suhu lingkungan yang terukur oleh sensor BMP280 diamati secara kontinu selama proses pengambilan data. Pemantauan dilakukan untuk memastikan bahwa asumsi suhu konstan pada persamaan barometrik dapat diterapkan dengan valid selama estimasi ketinggian dilakukan. Perubahan suhu dapat dilihat pada gambar 4.6.



Gambar 4. 6 Perubahan suhu selama berjalan dalam satu gedung

Pengukuran suhu dilakukan saat sensor bergerak naik dan turun dari Lantai Dasar (Lantai 0) hingga Lantai 6 di dalam gedung. Hasil pengukuran menunjukkan bahwa suhu berada pada kisaran antara 26.77 °C hingga 27.53 °C, dengan fluktuasi yang terjadi secara bertahap dan lambat. Tidak ditemukan adanya lonjakan atau perubahan tajam yang menandakan gangguan lingkungan secara signifikan.

Tabel 4. 2 Perhitungan Suhu

Parameter	Nilai (°C)
Minimum	26.77
Maksimum	27.53
Rata-rata	27.007
Standar Deviasi	0.23

Nilai standar deviasi yang kecil (0.23 °C) menunjukkan bahwa fluktuasi suhu sangat rendah dan perubahan berlangsung secara perlahan tanpa anomali. Hal ini memperkuat validitas penggunaan asumsi suhu konstan dalam perhitungan ketinggian berdasarkan tekanan udara.

Berdasarkan pola data yang diperoleh, suhu dapat dikatakan stabil secara operasional, sehingga tidak memengaruhi estimasi ketinggian secara signifikan. Oleh karena itu, pada penelitian ini digunakan pendekatan suhu konstan, dengan menggunakan nilai rata-rata suhu sebagai parameter koreksi dalam perhitungan ketinggian relatif antar lantai.

Untuk konversi suhu dari derajat Celcius ke derajat Kelvin sebagai syarat dalam persamaan barometrik, digunakan rumus berikut:

$$\text{Kelvin degree} = \text{Celcius Degree} + 273.15 \quad (3.4)$$

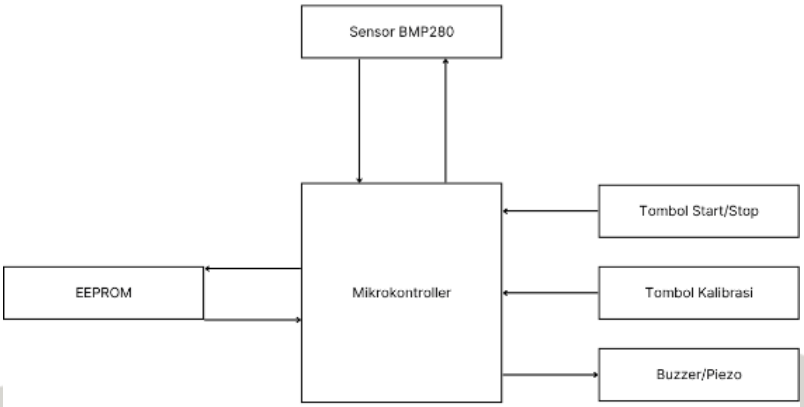
Dengan menggunakan pendekatan ini, perhitungan ketinggian menjadi lebih akurat dan stabil, karena memperhitungkan pengaruh suhu secara konsisten tanpa dipengaruhi fluktuasi yang tidak signifikan.

4.3. Desain Sistem

Sistem yang dikembangkan pada proyek Praktik Kerja Lapangan ini bertujuan untuk membantu penyandang tunanetra dan low vision dalam mengetahui posisi lantai secara otomatis saat berada di dalam gedung bertingkat. Sistem ini bekerja dengan membaca tekanan udara melalui sensor barometrik BMP280 dan mengolah data tersebut untuk memperkirakan perubahan ketinggian secara relatif terhadap lantai dasar.

4.3.1. Komponen Sistem

Blok diagram dan tabel berikut menunjukkan komponen utama dari sistem beserta fungsi masing-masing:



Gambar 4. 7 Blok Diagram Sistem

Tabel 4. 3 Komponen sistem

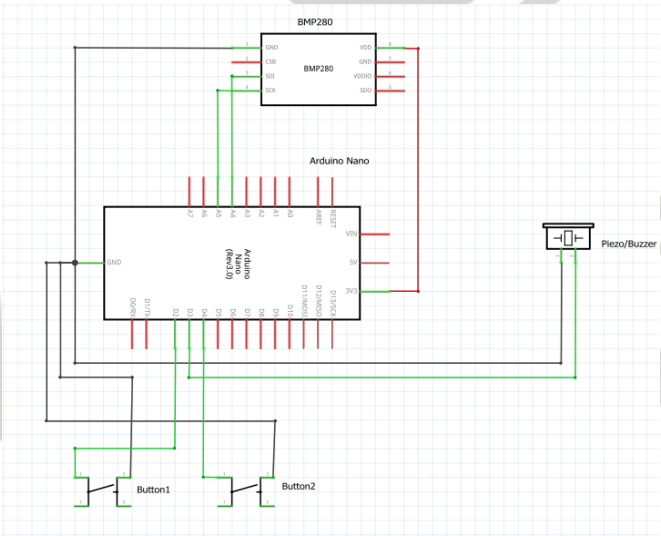
No	Komponen	Fungsi
1	Arduino Nano	Sebagai mikrokontroler utama yang mengontrol seluruh sistem
2	Sensor BMP280	Mengukur tekanan dan suhu lingkungan secara real time
3	EEPROM internal	Menyimpan data tekanan dasar sebagai referensi
4	Push Button	Mengaktifkan mode kalibrasi atau pengukuran
5	Buzzer	Memberikan notifikasi audio ketika lantai terdeteksi
6	Catu daya 5V	Sumber daya sistem dari USB

No	Komponen	Fungsi
7	Kabel Jumper	Menghubungkan berbagai komponen elektronik dalam suatu rangkaian tanpa perlu disolder

4.3.2. Diagram schematic dan alur kerja sistem

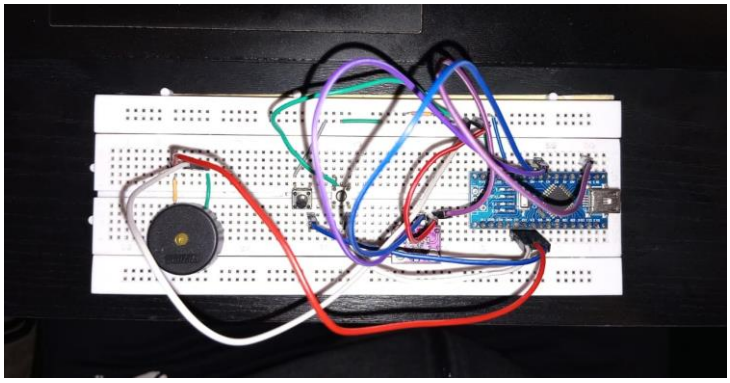
Untuk memberikan Gambaran yang jelas mengenai rancangan perangkat keras, pada bagian ini disajikan diagram skematik dan desain fisik dari alat yang dikembangkan

Diagram skematik pada Gambar 4.7 mengilustrasikan interkoneksi elektrikal antara mikrokontroler Arduino Nano dengan semua komponen. Pada rangkaian ditunjukan bagaimana sensor BMP280 terhubung melalui antarmuka I2C (pin SDA dan SCL), sementara tombol dan buzzer ke pin digital I/O. Skematik ini menjadi acuan utama dalam proses perakitan rangkaian elektronik.



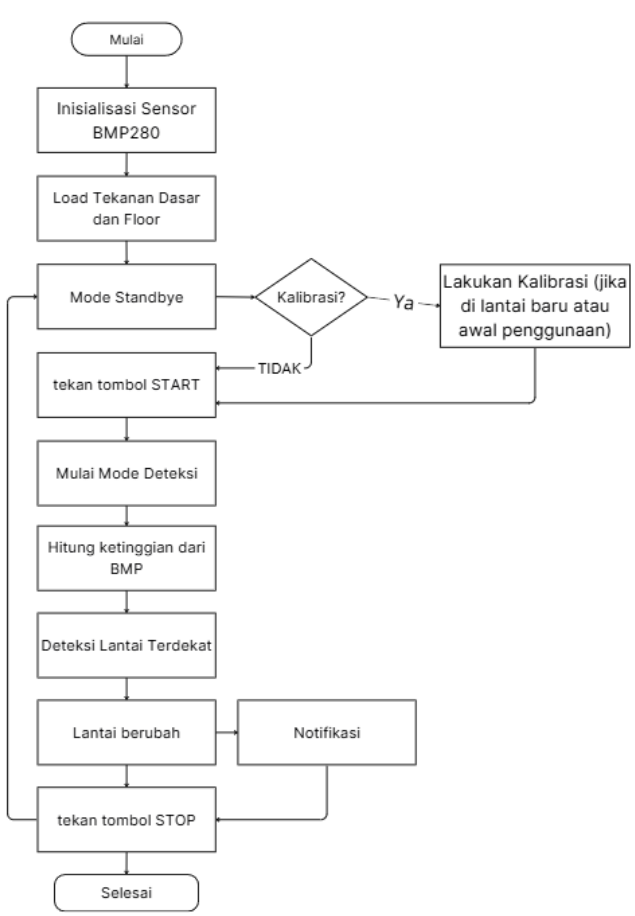
Gambar 4. 8 Schematic diagram

Pada Gambar 4.7 menampilkan desain alat atau prototipe dari alat pengukuran ketinggian. Penempatan tombol dan buzzer dirancang agar mudah diakses dan digunakan pengguna, terutama unruk memastikan fungsionalitas alat.



Gambar 4. 9 Desain alat pengukur ketinggian

Sistem ini dirancang untuk mengenali perubahan ketinggian berdasarkan pembacaan tekanan udara dari sensor barometrik BMP280. Sistem ini memanfaatkan prinsip fisika atmosfer untuk menghitung ketinggian relatif, lalu mencocokannya dengan data ketinggian yang telah ditentukan sebelumnya.



Gambar 4. 10 Diagram Alur Kerja Sistem Pengukur Ketinggian

Alur kerja sistem dimulai dari tahap inisialisasi sensor BMP280, di mana sensor diaktifkan dan dipersiapkan untuk membaca tekanan udara. Setelah sensor berhasil diinisialisasi, sistem akan memuat data tekanan dasar serta informasi lantai saat ini dari penyimpanan (EEPROM). Selanjutnya, sistem memasuki mode *standby*, menunggu perintah dari pengguna. Pada titik ini, pengguna memiliki opsi untuk melakukan proses kalibrasi atau *re-kalibrasi*. Jika tombol kalibrasi ditekan, sistem akan menjalankan proses kalibrasi untuk mencatat nilai tekanan sebagai acuan untuk lantai saat ini. Proses ini juga berlaku saat pengguna berada di lantai yang berbeda dan ingin menetapkan ulang tekanan dasar lantai tersebut.

Jika pengguna tidak memilih kalibrasi, sistem tetap berada di mode *standby* hingga tombol START ditekan. Setelah tombol START ditekan, sistem memasuki mode deteksi aktif. Di dalam mode ini, sistem mulai menghitung ketinggian berdasarkan perubahan tekanan udara yang dibaca oleh sensor BMP280. Berdasarkan data tersebut, sistem mendeteksi lantai terdekat dengan membandingkan nilai tekanan terhadap data referensi dari hasil kalibrasi.

Apabila terdeteksi adanya perubahan lantai, sistem akan memberikan notifikasi kepada pengguna. Proses deteksi terus berlangsung hingga pengguna menekan tombol STOP untuk menghentikan mode deteksi. Setelah proses deteksi dihentikan, sistem akan mengakhiri operasinya dan kembali ke kondisi semula.

4.3.3. Rumus Perhitungan Ketinggian

Dalam menentukan ketinggian untuk mendapatkan posisi maka digunakan ketinggian relatif yang lebih tahan terhadap fluktuasi dengan menggunakan rumus barometrik Invers.

$$h = \frac{R \cdot T}{g \cdot M} \cdot \ln\left(\frac{p_0}{p}\right) \tag{3.2}$$

Keterangan:

h = ketinggian (meter)

R = konstanta gas umum (8,314 J/mol·K)

T = suhu mutlak (Kelvin)

g = percepatan gravitasi bumi (9,80665 m/s²)

M = massa molar udara (0,0289644 kg/mol)

p0 = tekanan udara pada permukaan (hPa atau Pa)

p = tekanan udara pada titik ukur (hPa atau Pa)

Perhitungan ketinggian relatif(h) didasarkan pada perbedaan tekanan udara, dan karena suhu dianggap konstan maka digunakan juga rumus berikut,

$$\text{Kelvin degree} = \text{Celcius Degree} + 273.15 \quad (3.4)$$

Hasil dari implementasi rumus di atas, dilakukan melalui serangkaian pengukuran menggunakan sensor BMP280. Hasilnya kemudian dibandingkan dengan pengukuran dari Laser Distance Meter sebagai estándar acuan.

Tabel 4. 4 Hasil pengukutan ketinggian BMP280

Floor	Real Height (m)	Measured Heights (m)	Average Measured (m)	Average Diff (m)	Std Dev (m)
0	0.00	-0.01, 0.16, -0.18	-0.01	-0.01	0.17
1	3.55	3.50, 3.15, 3.75	3.47	-0.08	0.30
2	7.83	7.73, 7.77, 7.94	7.81	-0.02	0.11
3	12.12	11.84, 12.46, 12.44	12.25	+0.13	0.35
4	16.34	16.72, 16.73, 16.49	16.65	+0.31	0.14
5	20.60	21.06, 21.00, 20.80	20.95	+0.35	0.13
6	24.88	25.05, 25.18, 25.12	25.11	+0.23	0.07

Dapat dilihat pada Tabel 4.5 bahwa meskipun terdapat sedikit selisih antara tinggi aktual dan hasil pengukuran, nilai error tersebut tergolong sangat kecil jika dibandingkan dengan tinggi antar lantai pada gedung umumnya yang berkisar antara 3 hingga 4 meter. Error terbesar terjadi pada lantai 5 dengan selisih sebesar 35 cm. Selain itu, seluruh nilai standar deviasi yang tercatat juga relatif rendah, menunjukkan bahwa hasil pengukuran cukup konsisten. Oleh karena itu, dalam implementasi program digunakan nilai floor tolerance sebesar 50 cm untuk menghindari kesalahan dalam pembacaan dan penentuan lantai berdasarkan data ketinggian.

Floor tolerance dalam konteks pemrograman ini adalah nilai ambang toleransi perbedaan ketinggian yang masih dianggap berada dalam satu lantai yang sama. Artinya, jika nilai ketinggian yang terukur berada dalam ±50 cm dari nilai referensi suatu lantai, maka sistem akan tetap menganggap bahwa posisi pengguna berada pada lantai tersebut. Pendekatan ini penting untuk mengantisipasi fluktuasi tekanan udara atau noise sensor yang dapat menyebabkan variasi kecil dalam pengukuran ketinggian.

4.4. Implementasi dan Penjelasan Kode Program

Implementasi sistem dituangkan ke dalam firmware yang ditulis dalam bahasa C++ dengan kerangka kerja Arduino. Pada bagian ini akan dibahas logika dan fungsi-fungsi kunci yang membangun sistem secara keseluruhan

4.4.1. Inisialisasi dan Konfigurasi

Bagian awal program mendeklarasikan library yang dibutuhkan, seperti Wire.h untuk komunikasi I2C, Adafruit_BMP280.h untuk sensor, dan EEPROM.h untuk penyimpanan data, selain itu pin yang digunakan, serta konstanta penting seperti ketinggian setiap lantai. Array FLOOR_HEIGHTS berisi data ketinggian setiap lantai dalam meter, yang menjadi acuan utama bagi sistem untuk mengenali posisi. FLOOR_TOLERANCE mendefinisikan bata toleransi kesalahan pengukuran. Selain itu ada juga inisialisasi penting lainnya seperti UPDATE_INTERVAL yang mengatur frekuensi pemrosesan data, sementara DEBOUNCE_DELAY berfungsi untuk mencegah pembacaan ganda pada tombol.

```
1 | #include <Wire.h>
2 | #include <Adafruit_BMP280.h>
3 | #include <EEPROM.h>
4 |
5 | // Inisialisasi sensor
6 | Adafruit_BMP280 bmp;
7 |
8 | // --- Pin Konfigurasi ---
9 | const uint8_t BUTTON_PIN = 2;
10 | const uint8_t CALIBRATE_PIN = 4;
11 | const uint8_t BUZZER_PIN = 3;
12 |
13 | // --- Konfigurasi Fisik & Toleransi ---
14 | const uint8_t NUM_FLOORS = 7;
15 | const float FLOOR_HEIGHTS[NUM_FLOORS] = {0.0, 3.55, 7.83, 12.12, 16.34,
16 | 20.6, 24.88};
17 | const float FLOOR_TOLERANCE = 0.5; // Toleransi 50 cm
18 |
19 | // --- Konfigurasi Sistem & Waktu ---
20 | const uint16_t PRESSURE_EEPROM_ADDR = 0;
21 | const uint8_t FLOOR_EEPROM_ADDR = 2;
22 | const uint16_t CALIBRATION_SAMPLES = 50;
23 | const uint16_t UPDATE_INTERVAL = 100;
24 | const uint16_t DEBOUNCE_DELAY = 200;
25 |
26 | // --- Variabel Global ---
27 | float basePressure = 1013.25;
28 | int8_t currentFloor = 0;
29 | bool isDetecting = false;
30 | bool floorChanged = false;
31 | uint32_t stableStartTime = 0;
32 | uint32_t lastUpdateTime = 0;
33 | uint32_t lastStandbyMsg = 0;
34 | uint32_t lastButtonPress = 0;
35 |
36 | // Variabel untuk deteksi arah
37 | enum Direction { STILL, UP, DOWN };
38 | Direction lastDirection = STILL;
39 | float lastAltitude = -100.0; // Inisialisasi dengan nilai yang tidak mungkin
40 |
41 | // Konstanta Fisika
```

```
42 | const float R = 8.31447;
43 | const float g = 9.80665;
44 | const float M = 0.0289644;
```

4.4.2. Alur Kerja Utama

Setelah dilakukan inisialisasi dan konfigurasi, maka selanjutnya masuk pada bagian alur kerja utama pada sistem yang dibuat yaitu bagian setup() dan loop(), keduanya merupakan fungsi utama yang akan dijalankan oleh sistem. Setup() dieksekusi sekali saat perangkat dinyalakan, fungsi ini bertugas menginisialisasi semua perangkat keras yang dibuat dalam fungsi *initializeHardware*() dan memuat status terakhir sistem, seperti tekanan dasar dan lantai terakhir, dari EEPROM dari fungsi *loadSystemState*().

```
44  void setup() {
```


Kemudian fungsi loop(), fungsi ini akan di eksekusi secara berulang setelah setup() selesai. Pada fungsi ini secara konstan memeriksa input dari pengguna pada fungsi handelButtons(). Jika mode deteksi aktif *isDetecting*, fungsi akan memanggil proccesSensorData() secara berkala untuk melakukan pengukuran dan análisis.

```
54  void loop() {
55    handleButtons();
56
57    // Jika sistem dalam mode deteksi, proses data sensor secara berkala
58    if (isDetecting && (millis() - lastUpdateTime >= UPDATE_INTERVAL)) {
59      lastUpdateTime = millis();
60      processSensorData();
```

4.4.3. Akuisisi Data dan Perhitungan Ketinggian

Proses selanjutnya yaitu akuisisi data mentah hingga menjadi informasi ketinggian yang dapat diolah. Dimulai dengan fungsi processSensorData() untuk membaca tekanan dan suhu dari sensor BMP280. Kemudian untuk bagian perhitungan ketinggian menggunakan fungsi *calculateAltitude* untuk mengimplementasikan rumus barometrik (3.2) untuk mengonversi data tekanan menjadi data ketinggian relatif dalam meter. Perhitungan menggunakan tekanan dasar (basePressure) sebagai titik acuan nol.

```
71  void processSensorData() {
72    float rawPressure = bmp.readPressure() / 100.0f;
73    float temperature = bmp.readTemperature();
74
75    if (!isValidPressure(rawPressure) || isnan(temperature)) {
76      Serial.println(F("ERROR: Pembacaan sensor tidak valid!"));
77      return;
78    }
79
80          float currentAltitude = calculateAltitude(rawPressure, basePressure,
temperature);
81
82    // Tentukan arah pergerakan
83    if (lastAltitude > -99.0) {
84        float change = currentAltitude - lastAltitude;
```

```
85        if (change > 0.25) {
86            lastDirection = UP;
87        } else if (change < -0.25) {
88            lastDirection = DOWN;
89        }
90        // Jika perubahan kecil, 'lastDirection' tidak diubah ke STILL
91            // agar sistem tetap mengingat arah pergerakan terakhir (penting saat lift melambat).
92    }
93    lastAltitude = currentAltitude;
94
95    // SESUDAH (Gunakan baris ini)
96    int8_t detectedFloor = detectFloor(currentAltitude);
97
98    // Log data ke serial untuk debugging
99              logSensorData(millis(), rawPressure, rawPressure, currentAltitude, detectedFloor);
100
101    // Tangani perubahan lantai dan notifikasi
102    handleFloorChange(detectedFloor);
103 }
```


4.4.4. Deteksi Lantai

Setelah ketinggian didapatkan, fungsi detectFloor bertugas untuk menentukan posisi lantai. Fungsi bekerja dengan mencari "lantai terdekat" dari posisi saat itu dengan menghitung selisih absolut antara ketinggian terukur (altitude) dengan setiap data ketinggian yang tersimpan di dalam *array* FLOOR_HEIGHTS. Lantai yang menghasilkan selisih terkecil akan ditetapkan sebagai lantai yang terdeteksi sesuai dengan FLOOR_TOLERANCE, maka setelah itu lantai akan berubah yang dijalankan lewat fungsi handleFloorChange yang memastikan notifikasi dari fungsi notifyArrival() hanya diberikan ketika ada perubahan lantai yang sesungguhnya, bukan sekadar fluktuasi kecil.

```
106 int8_t detectFloor(float altitude) {
107   // Prioritaskan lantai terakhir jika melebihi ketinggian maksimum
108   if (altitude > FLOOR_HEIGHTS[NUM_FLOORS-1] + FLOOR_TOLERANCE)
109     return NUM_FLOORS-1; // Kembalikan lantai teratas
110   }
111
112   int8_t closestFloor = -1;
113   float minDifference = FLOOR_TOLERANCE * 1 /*1.5*/; // Toleransi lebih longg
114
115   for (int i = 0; i < NUM_FLOORS; i++) {
116     float difference = abs(altitude - FLOOR_HEIGHTS[i]);
117
118     // Jika dalam toleransi dan lebih dekat dari sebelumnya
119     if (difference < minDifference) {
120       minDifference = difference;
121       closestFloor = i;
122     }
123   }
124
125   return closestFloor;
126 }
```

4.4.5. Sistem Kalibrasi

Untuk menjaga akurasi pengukuran ketinggian terhadap pengaruh perubahan cuaca, suhu, dan tekanan atmosfer jangka panjang, sistem ini dilengkapi fitur kalibrasi manual yang dapat diaktifkan melalui tombol saat perangkat dalam kondisi diam di lantai tertentu. Proses kalibrasi menggunakan fungsi calibrateSensor() yang dirancang untuk menghitung ulang tekanan dasar (basePressure) yang menjadi titik referensi utama dalam konversi tekanan menjadi estimasi lantai.

Proses Kalibrasi:

1. Pengambilan Sampel Tekanan

   Fungsi calibrateSensor() akan mengambil 50 sampel tekanan udara secara berurutan menggunakan sensor BMP280. Data ini disimpan dalam array untuk dianalisis lebih lanjut. Pengambilan sampel dilakukan dengan interval delay kecil untuk menstabilkan pembacaan sensor.

2. Pemrosesan Nilai Median

   Untuk menghindari outlier atau noise akibat fluktuasi pembacaan sensor, nilai tekanan yang dikumpulkan diolah menggunakan metode median, bukan rata-rata, agar lebih representatif terhadap kondisi sebenarnya.

3. Perhitungan Tekanan Dasar (basePressure)

   Sistem menggunakan rumus berikut untuk menghitung nilai tekanan dasar berdasarkan ketinggian lantai saat ini (knownHeight) yang sudah ditentukan secara statis (misalnya lantai dasar = 0 m, lantai 1 = 3 m, dst):

$$P_0 = P_{med} \cdot \exp\left(\frac{gMh}{RT}\right)$$

   Di mana:

   - *Pmed*: median tekanan dari sensor (dalam hPa)
   - g: percepatan gravitasi bumi (9.80665 m/s²)
   - M: massa molar udara (0.0289644 kg/mol)
   - h: ketinggian lantai dalam meter
   - R: konstanta gas umum (8.3144598 J/mol·K)
   - T: suhu absolut dalam Kelvin (diambil dari sensor suhu BMP280)

4. Penyimpanan EEPROM

   Setelah tekanan dasar dihitung, nilainya disimpan secara permanen ke memori EEPROM agar tetap tersedia saat alat dinyalakan ulang. Nilai ini akan digunakan sebagai acuan pengukuran tekanan di sesi berikutnya.

5. Penggunaan Tombol untuk Kontrol Validitas

   Kalibrasi tidak harus dilakukan di lantai dasar, namun sistem mengasumsikan bahwa lantai tempat dilakukan kalibrasi adalah benar sesuai dengan variabel currentFloor saat itu. Oleh karena itu, pengguna diharapkan hanya menekan

tombol kalibrasi ketika yakin bahwa lantai tempat alat berada adalah valid (misalnya alat sudah berhenti diam dan lantai tidak berubah karena drift sensor).

```
250 void calibrateSensor() {
251   Serial.println(F("\nINFO: Kalibrasi dimulai..."));
252   tone(BUZZER_PIN, 1800, 100);
253
254   // 1. Kumpulkan data sensor
255   float pressures[CALIBRATION_SAMPLES];
256   uint8_t validCount = 0;
257
258   for (uint8_t i = 0; i < CALIBRATION_SAMPLES; i++) {
259     float p = bmp.readPressure();
260     if (!isnan(p)) {
261       pressures[validCount++] = p;
262
263       // Feedback progres
264       if (i % 10 == 0) {
265         tone(BUZZER_PIN, 1500, 50);
266         delay(50);
267       }
268     }
269     delay(50);
270   }
271
272   // 2. Hitung median untuk menghilangkan outlier
273   float medianPressure = computeMedian(pressures, validCount);
274
275   // 3. Gunakan lantai saat ini sebagai acuan
276   float knownHeight = FLOOR_HEIGHTS[currentFloor];
277   float temperature = bmp.readTemperature();
278
279   // 4. Hitung base pressure baru
280   float tempK = (isnan(temperature) ? 298.15 : temperature + 273.15);
281       float newBasePressure = (medianPressure/100.0f) * exp((g * M *
knownHeight)/(R * tempK));
282
283   // 5. Simpan dan beri konfirmasi
284   basePressure = newBasePressure;
285   savePressureToEEPROM(basePressure);
```

```
286
287   Serial.print(F("SUKSES: Kalibrasi di Lantai ")); Serial.print(currentFloor);
288   Serial.print(F(". Tekanan dasar: ")); Serial.print(basePressure, 2);
289   Serial.println(F(" hPa"));
290
291   // Bunyi konfirmasi
292   for (uint8_t i = 0; i <= currentFloor; i++) {
293     tone(BUZZER_PIN, 1500 + (i * 200), 150);
294     delay(200);
295   }
296 }
```

4.5. Permasalahan dan Solusi

Selama proses perancangan dan pengujian sistem, beberapa tantangan teknis dihadapi. Bagian ini menguraikan permasalahan yang dihadapi beserta solusi baik yang dapat diselesaikan maupun belum.

1. Perubahan tekanan atmosfer akibat cuaca

   Tekanan udara sebagai acuan dasar dapat berubah setiap hari karena faktor cuaca, hal ini menyebabkan kalibrasi pada satu haru menjadi tidak akurat jika digunakan pada hari berikutnya, sehingga mengakibatkan kesalahan perhitungan.

   Solusi: Ditambahkan fitur kalibrasi manual sehingga pengguna dapat melakukan kalibrasi saat berada di lantai 0 untuk menyesuaikan tekanan awal dari perubahan tekanan yang dapat menyebabkan kesalahan perhitungan

2. Fluktuasi dan Noise pada Sensor

   Sensor barometrik rentan terhadap noise sehingga dapat menyebabkan pembacaan data tekanan sedikir berubah-ubah meskipun alat dalam kondisi diam.

   Solusi: BMP280 memiliki bawaan filter Infinite Impulse Response yang dapat digunakan untuk memfilter noise dan dalam implementasi nya digunakan FILTER_X16 yaitu tertinggi yang digunakan untuk menghaluskan data mentah yang dibaca dari sensor.

3. Penyimpanan data di EPPROM

   Penyimpanan EEPROM yang dapat rusak seteelah penggunaan 100.000 kali harus di optimalkan dalam penggunaannya, awalnya data yang disimpan langsung dari ukuran besar datanya sehingga memakan lebih banyak penyimpanan EEPROM dan dapat mengurangi masa pengunaannya/

   Solusi: Data yang disimpan diperkecil byte nya, seperti data int yang 2 byte disimpan dalam 1 byte, tetapi jika di butuhkan akan dapat dibaca kembali dalam 2 byte atau data awal, sehingga memungkinkan penyimpanan EEPROM lebih awet dan lebih lama.

4. Kesalahan Deteksi lantai

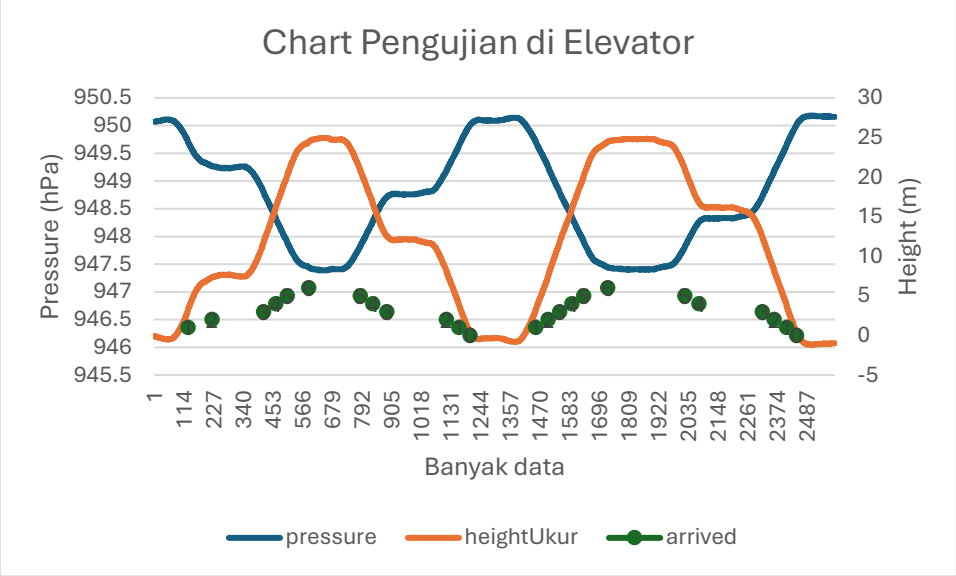
   Deteksi lantai dapat mengalami fluktuasi seiring waktu, baik saat guncangan elevator saat mulai bergerak atau berhenti, serta pergerakan kecil pengguna dapat menyebabkan perubahan ketinggian sessaat yang bisa salah diinterpretasikan sebagain perpindahan lantai

   Solusi: Dalam program ditetapkan sebuah FLOOR_TOLERANCE (50 cm). Sebuah lantai baru dianggap terdeteksi hanya jika ketinggiannya dengan data lantai yang tersimpan melebihi batas toleransi ini.

5. Penggunaan pembacaan dalam waktu lama

   Perubahan tekanan atmosfer harian dan drift sensor menyebabkan akurasi sistem menurun seiring waktu, sehingga memerlukan kalibrasi rutin. Oleh karena itu dibuatlah kalibrasi yang mendukung *multi-floor* kalibrasi, memungkinkan kalibrasi dilakukan di lantai mana pun selama ketinggian dan floornya diketahui, seperti contohnya kalibrasi di lantai 3 (12.12m) yang akan menghitung ulang basePressure secara akurat. Sistem juga menyimpan data tekanan dasar dan lantai terakhir di EEPROM untuk memastikan stabilitas setelah reboot. Untuk meningkatkan akurasi, diterapkan filter hardware FILTER_X16 dan median 50 sample untuk mengurangi noise, serta deteksi arah pergerakan yang mencegah kesalahan deteksi perpindahan lantai. Solusi-solusi ini membuat sistem lebih handal untuk penggunaan jangka panjang.

4.6. Hasil Pengujian dan Analisis



Gambar 4. 11 Hasil pengujian naik turun elevator

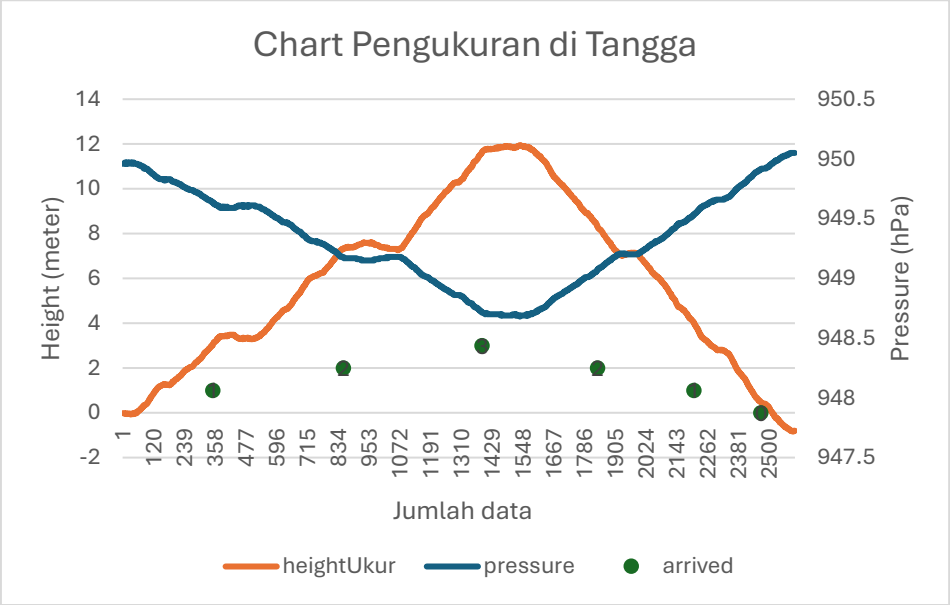
Pengujian pada Gambar 4.11 dilakukan dengan tujuan mengevaluasi konsistensi pembacaan sensor barometrik (BMP280) dalam sistem deteksi lantai secara otomatis berbasis tekanan udara. Dalam skenario ini, sistem di uji di dalam elevator dan dijalankan naik-turun dari lantai 0 hingga lantai 6, lalu kembali ke lantai 0, dan diulang beberapa kali selama sekitar lima menit tanpa melakukan kalibrasi ulang setiap siklus.

Komponen Grafik:

- Sumbu X (Banyak data): Menunjukkan urutan waktu berdasarkan jumlah sampel sensor.
- Sumbu Y kiri (hPa): Menampilkan data tekanan udara.
- Sumbu Y kanan (m): Menampilkan hasil estimasi ketinggian (heightUkur).
- Garis biru (pressure): Tekanan udara yang dibaca sensor BMP280.
- Garis oranye (heightUkur): Perhitungan ketinggian berdasarkan selisih tekanan terhadap base pressure.

- Titik hijau (arrived): Deteksi sistem saat elevator dianggap tiba di suatu lantai (keadaan diam).

Hasil dari grafik menunjukan hubungan antara ketinggian dan tekanan yang juga di validasi dengan arrived saat sistem mendeteksi elevator telah mencapai dan berhenti di suatu lantai, dapat dilihat bahwa nilai tekanan berkurang (garis biru turun) dan nilai ketinggian meningkat (garis oranye naik), dan sebaliknya saat elevator turun, tekanan meningkat dan ketinggian menurun. Ketinggian awal dari pengujian menunjukkan 0 meter dan menunjukkan perubahan ketika ketinggian naik mencapai lantai teratas hampir mencapai ketinggian 25 meter, dan hasil tersebut sesuai dengan FLOOR_HEIGHT yang diketahui bahwa lantai 6 berada di ketinggian 24.88 meter dan perubahan tersebut terdeteksi dengan adanya perubahan arrived (titik hijau), selain itu nilai heightUkur secara periodik kembali mendekati nol meter saat elevator berada di lantai 0, menunjukkan bahwa base pressure yang digunakan tetap valid sepanjang pengujian.



Gambar 4. 12 Hasil pengujian naik turun tangga

Pada gambar 4.22 pengujian juga dilakukan untuk mengevaluasi konsistensi sensor barometrik, tetapi kali ini di uji dengan naik turun tangga dari lantai 0 sampai 3 dan kembali ke lantai 0, sehingga dalam skenario berbeda dari pengujian di gambar 4.21.

Komponen Grafik:

- Sumbu X (Banyak data): Menunjukkan urutan waktu berdasarkan jumlah sampel sensor.
- Sumbu Y kiri (m): Menampilkan hasil estimasi ketinggian (heightUkur).
- Sumbu Y kanan (hPa): Menampilkan data tekanan udara.
- Garis biru (pressure): Tekanan udara yang dibaca sensor BMP280.
- Garis oranye (heightUkur): Perhitungan ketinggian berdasarkan selisih tekanan terhadap base pressure.
- Titik hijau (arrived): Deteksi sistem saat elevator dianggap tiba di suatu lantai (keadaan diam).

Titik-titik hijau pada grafik menandai deteksi perubahan lantai (arrived), menunjukkan bahwa sistem mampu mengenali posisi lantai dengan cukup tepat. Meskipun pergerakan secara manual tidak sehalus elevator, sistem tetap dapat mengenali perpindahan antar lantai secara konsisten, meskipun terdapat sedikit fluktuasi tekanan atau ketinggian akibat langkah kaki atau jeda pergerakan saat pengguna berpindah antar lantai, selain itu perubahan teknana dapat terlihat dengan baik dan ketinggian yang muncul masih berada di kisaran yang wajar, contohnya dipengujian ketika mencapai lantai 3 ketinggian berada di sekitar gridline garis 12 meter dan ini sesuai dengan ketinggian yang diketahui di FLOOR_HEIGHT yaitu 12.12 meter, tetapi pada akhir pengujian ketika kembali ke lantai 0 pengukuran ketinggian tidak kembali tepat di 0 meter dan itulah yang masih menjadi kekurangannya tetapi hal ini masih bisa di selesaikan dengan melakukan kalibrasi bertahap seperti sekali naik kemudian akan turun maka lakukan kalibrasi sehingga nilai tetap sesuai karena kalibrasi ini dapat dilakukan di lantai manapun.

Hasil akhir dari pengujian ini menunjukkan bahwa prototipe yang digunakan dapat bekerja dengan baik mulai dari tombol *start/stop*, tombol kalibrasi, buzzer dan

sensor bmp280, serta sensor bmp280 dapat digunakan dalam pengunaan pengukuran ketinggian dapat dilihat pada gambar 4.5 yang disajikan pada sub-bab sebelumnya yang menampilkan selisih tertinggi yaitu +0.35 meter dalam pembacaan ketinggian tiap lantai.dan nilai standar deviasi yang konsisten menunjukkan bahwa sistem memiliki presisi yang baik

Didalam pengujian baik menggunakan elevator atau tangga pembacaan berjalan dengan baik dan notif dari buzzer berbunyi dengan baik saat sampai di lantai yang dituju, tetapi juga masih ada beberapa kendala yaitu saat menaiki atau menuruni tangga terkadang notif buzzer berbunyi sedikit lebih cepat saat belum sampai di lantai yang dituju.

Bab V
Penutup

5.1 Simpulan

Berdasarkan kegiatan Praktik Kerja Lapangan (PKL) yang telah dilaksanakan, dapat disimpulkan bahwa kegiatan ini memberikan pengalaman langsung dalam merancang, mengimplementasikan, serta menganalisis sistem berbasis mikrokontroler menggunakan sensor tekanan (BMP280) untuk estimasi ketinggian dan deteksi lantai secara otomatis.

Selama pelaksanaan PKL, penulis berhasil membangun alat yang mampu membaca tekanan udara, menghitung ketinggian, serta menentukan posisi lantai berdasarkan toleransi tertentu. Sistem diuji menggunakan elevator dan tangga, dan menunjukkan bahwa pembacaan sensor cukup stabil dan sistem mampu mengenali lantai secara relatif akurat.

Namun, sistem ini masih memiliki beberapa keterbatasan, salah satu keterbatasan utama adalah penggunaan single barometer, yaitu hanya satu sensor tekanan yang terpasang pada unit bergerak (Arduino). Tanpa adanya sensor referensi di posisi tetap (misalnya di lantai dasar), sistem menjadi sangat bergantung pada kestabilan tekanan udara lingkungan. Perubahan tekanan yang terjadi secara alami, misalnya akibat perubahan cuaca atau sirkulasi udara dalam gedung, dapat mempengaruhi hasil pembacaan ketinggian dan menyebabkan pergeseran nilai lantai yang terdeteksi. Hal ini menyebabkan kebutuhan akan kalibrasi ulang secara berkala agar sistem tetap akurat.

5.2 Saran

Berdasarkan hasil pelaksanaan dan pengujian sistem selama kegiatan PKL, terdapat beberapa saran yang dapat diberikan untuk pengembangan dan peningkatan sistem ke depan:

1. Penambahan sensor bantu:

   Disarankan untuk menggunakan sensor seperti sensor accelerometer atau pun menambahkan sensor barometer, dengan penambahan sensor ini dapat membantu pembacaan dan mengurangi fluktuasi menjadi lebih baik kedepannya.

2. Implementasi Filter tambahan:

   Untuk meningkatkan stabilitas pembacaan dapat menggunakan algoritma penyaring data tambahan seperti moving average dan kalman filter, hal ini membantu mengurangi fluktuasi tekanan akibat noise sensor maupun gangguan lingkungan.

3. Uji coba langsung dengan penyandang tunanetra:

   Dalam pengujian ini dilakukan tanpa penyandang tunanetra sehingga disarankan untuk dilakukan pengujian langsung dengan subjek penyandan tunanetra untuk menilai efektivitas alat secara langsung terhadap target pengguna.

Daftar Pusaka

Ahmetovic, D., Gleason, C., Kitani, K., Takagi, H. and Asakawa, C. (2016) 'NavCog: turn-by-turn smartphone navigation assistant for people with visual impairments or blindness', *Proceedings of the 13th Web for All Conference (W4A '16)*, Montreal, Canada, 11-13 April, Article No. 22, pp. 1-2. doi: [10.1145/2899475.2899509](https://doi.org/10.1145/2899475.2899509). Available at: ResearchGate https://www.researchgate.net/publication/305648807

Arduino, 2024. *Arduino Nano*. Arduino Documentation. Available at: https://docs.arduino.cc/hardware/nano [Accessed 28 June 2025].

Bai, Y. 2015, '*A Wearable Indoor Navigation System for Blind and Visually Impaired Individuals*', PhD Dissertation, University of Pittsburgh.

Bosch. 2020. BMP280 - Digital pressure sensor [Datasheet]. Available at: https://www.boschsensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp280-ds001.pdf [Accessed 28 June 2025].

Britannica, 2024. *Mercury barometer*. Encyclopædia Britannica. Available at: https://www.britannica.com/technology/mercury-barometer

Flaxman, S.R., Bourne, R.R.A., Resnikoff, S., Ackland, P., Braithwaite, T., Cicinelli, M.V., et al., 2017. Global causes of blindness and distance vision impairment 1990–2020: a systematic review and meta-analysis. *Lancet Global Health*, 5(12), pp.e1221–e1234.

Holden, B.A., Fricke, T.R., Wilson, D.A., Jong, M., Naidoo, K.S., Sankaridurg, P., Resnikoff, S., and Frick, K.D., 2016. Global prevalence of visual impairment associated with myopic macular degeneration and other causes in 2020 and 2050. *The Lancet Global Health*, 5(9), pp.e888–e897.

Kim, K.S., Lee, S., Wang, R., et al. (2017). *Hierarchical building/floor classification and location estimation using Wi-Fi fingerprinting based on deep neural networks*. arXiv:1710.00951

Kusuma, H. A., Yuliani, Y., Suhendra, T., Devendra, D., and Setyono, D. E. D., 2023. Evaluating the Accuracy of BMP280 and BME280 Sensors for Sea Level in a Coastal Environment: A Field Study at Tanjung Siambang Pier. *ILMU KELAUTAN: Indonesian Journal of Marine Sciences*, [Online] Volume 28(2), pp. 189-202. https://doi.org/10.14710/ik.ijms.28.2.189-202 [Accessed 2 Jul. 2025].

Murata, M., Ahmetovic, D., Sato, D., Takagi, H., Kitani, K. and Asakawa, C. (2018) 'Smartphone-based indoor localization for blind navigation across building complexes', *Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Athens, Greece, 19-23 March, pp. 1-10. doi: [10.1109/PERCOM.2018.8444593](https://doi.org/10.1109/PERCOM.2018.8444593). Available at: ResearchGate https://www.researchgate.net/publication/322807761

Silva, C. and Wimalaratne, P. (2016) 'Sensor fusion for visually impaired navigation in constrained spaces', in *Proceedings of the 2016 IEEE International Conference on Information and Automation for Sustainability (ICIAFS)*. Galle, Sri Lanka, 16-19 December, pp. 1-6. doi: [10.1109/ICIAFS.2016.7946537] (https://doi.org/10.1109/ICIAFS.2016.7946537). Available at: [ResearchGate](https://www.researchgate.net/publication/317640356)

Si, M., Wang, Y., Zhou, N., Seow, C. and Siljak, H., 2023. A Hybrid Indoor Altimetry Based on Barometer and UWB. *Sensors*, 23(9), p.4180. https://doi.org/10.3390/s23094180

Wallace, J.M. and Hobbs, P.V., 2006. *Atmospheric Science: An Introductory Survey*. 2nd ed. Amsterdam: Elsevier.

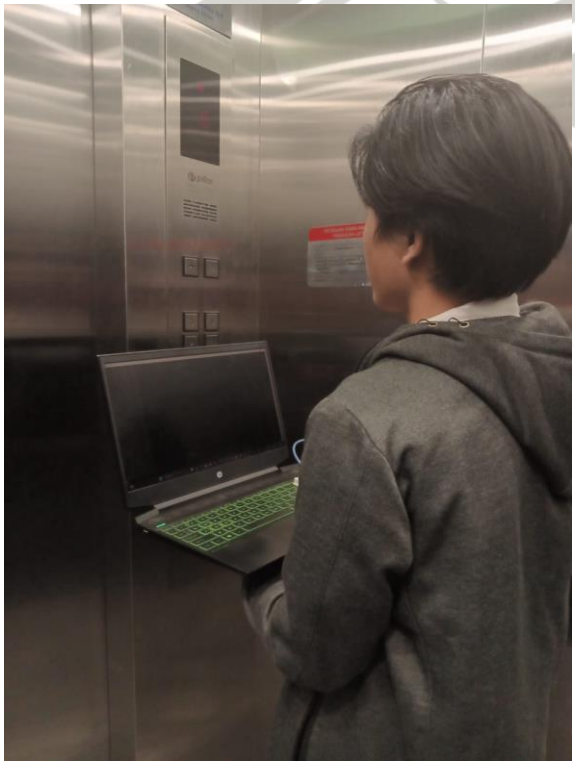
World Health Organization, 2019. *Blindness and visual impairment*. [online] Available at: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

Zhang, J., Wang, H., Wang, Y. and Wang, D. (2016). Assistive technology for visually impaired in public spaces: A review. Journal of Accessibility and Design for All, 6(2), pp.152–169

.

LAMPIRAN

Pengambilan data



Kode Program Arduino:

```
#include <Wire.h>
#include <Adafruit_BMP280.h>
#include <EEPROM.h>

// Inisialisasi sensor
Adafruit_BMP280 bmp;
//  Pin Konfigurasi
const uint8_t BUTTON_PIN = 2;
const uint8_t CALIBRATE_PIN = 4;
const uint8_t BUZZER_PIN = 3;
//  Konfigurasi Fisik & Toleransi
const uint8_t NUM_FLOORS = 7;
const float FLOOR_HEIGHTS[NUM_FLOORS] = {0.0, 3.55, 7.83, 12.12, 16.34,
20.6, 24.88};
const float FLOOR_TOLERANCE = 0.5; // Toleransi 50 cm
//Konfigurasi Sistem & Waktu
const uint16_t PRESSURE_EEPROM_ADDR = 0;
const uint8_t FLOOR_EEPROM_ADDR = 2;
const uint16_t CALIBRATION_SAMPLES = 50;
const uint16_t UPDATE_INTERVAL = 100;
const uint16_t DEBOUNCE_DELAY = 200;
// Variabel Global
float basePressure = 1013.25;
int8_t currentFloor = 0;
bool isDetecting = false;
bool floorChanged = false;
uint32_t stableStartTime = 0;
uint32_t lastUpdateTime = 0;
uint32_t lastStandbyMsg = 0;
uint32_t lastButtonPress = 0;

// Variabel untuk deteksi arah
enum Direction { STILL, UP, DOWN };
Direction lastDirection = STILL;
float lastAltitude = -100.0; // Inisialisasi dengan nilai yang tidak mungkin

// Konstanta Fisika
const float R = 8.31447;
const float g = 9.80665;
const float M = 0.0289644;
```

```
41
42  // --- FUNGSI UTAMA ---
43
44  void setup() {
45    Serial.begin(115200);
46    initializeHardware();
47    loadSystemState();
48
49    Serial.println(F("STATUS: Sistem Deteksi Otomatis Siap"));
50    Serial.print(F("Lantai Terakhir: ")); Serial.println(currentFloor);
51    Serial.print(F("Tekanan Dasar: ")); Serial.print(basePressure, 2); Serial.println(F("
hPa"));
52  }
53
54  void loop() {
55    handleButtons();
56
57    // Jika sistem dalam mode deteksi, proses data sensor secara berkala
58    if (isDetecting && (millis() - lastUpdateTime >= UPDATE_INTERVAL)) {
59      lastUpdateTime = millis();
60      processSensorData();
61    }
62    // Jika tidak, tampilkan pesan standby
63    else if (!isDetecting && (millis() - lastStandbyMsg >= 2000)) {
64      lastStandbyMsg = millis();
65      Serial.println(F("STATUS: Mode Standby [Tekan tombol START]"));
66    }
67  }
68
69  // funsgi pemrosesan dan deteksi
70
71  void processSensorData() {
72    float rawPressure = bmp.readPressure() / 100.0f;
73    float temperature = bmp.readTemperature();
74
75    if (!isValidPressure(rawPressure) || isnan(temperature)) {
76      Serial.println(F("ERROR: Pembacaan sensor tidak valid!"));
77      return;
78    }
79
80    float currentAltitude = calculateAltitude(rawPressure, basePressure, temperature);
81
```

```
82  // Tentukan arah pergerakan
83  if (lastAltitude > -99.0) {
84      float change = currentAltitude - lastAltitude;
85      if (change > 0.25) {
86          lastDirection = UP;
87      } else if (change < -0.25) {
88          lastDirection = DOWN;
89      }
90      // Jika perubahan kecil, 'lastDirection' tidak diubah ke STILL
91      // agar sistem tetap mengingat arah pergerakan terakhir
92  }
93  lastAltitude = currentAltitude;
94
95
96  int8_t detectedFloor = detectFloor(currentAltitude);
97
98  // Log data ke serial untuk debugging
99  logSensorData(millis(), rawPressure, rawPressure, currentAltitude, detectedFloor);
100
101  // Tangani perubahan lantai dan notifikasi
102  handleFloorChange(detectedFloor);
103 }
104
105
106 int8_t detectFloor(float altitude) {
107  // Prioritaskan lantai terakhir jika melebihi ketinggian maksimum
108  if (altitude > FLOOR_HEIGHTS[NUM_FLOORS-1] + FLOOR_TOLERANCE)
{
109    return NUM_FLOORS-1;
110  }
111
112  int8_t closestFloor = -1;
113  float minDifference = FLOOR_TOLERANCE * 1 ; // Toleransi lebih longgar
114
115  for (int i = 0; i < NUM_FLOORS; i++) {
116    float difference = abs(altitude - FLOOR_HEIGHTS[i]);
117
118    // Jika dalam toleransi dan lebih dekat dari sebelumnya
119    if (difference < minDifference) {
120      minDifference = difference;
121      closestFloor = i;
122    }
```

```
123   }
124
125   return closestFloor;
126 }
127
128 void handleFloorChange(int8_t newFloor) {
129   if (newFloor == -1) {
130     // Handle kasus khusus saat di atas lantai teratas
131     if (lastAltitude > FLOOR_HEIGHTS[NUM_FLOORS-1] +
FLOOR_TOLERANCE) {
132       newFloor = NUM_FLOORS-1;
133     } else {
134       return;
135     }
136   }
137
138   // Filter fluktuasi kecil
139   if (abs(FLOOR_HEIGHTS[newFloor] - FLOOR_HEIGHTS[currentFloor]) < 0.3)
{
140     return;
141   }
142
143   // Update status lantai
144   currentFloor = newFloor;
145   stableStartTime = millis();
146   floorChanged = true;
147
148   // Notifikasi langsung tanpa delay stabilisasi
149   notifyArrival(currentFloor);
150   saveFloorToEEPROM(currentFloor);
151   floorChanged = false;
152 }
153
154 void notifyArrival(int8_t floor) {
155   // memastikan floor dalam range yg valid
156   floor = constrain(floor, 0, NUM_FLOORS-1);
157
158   Serial.print(F("INFO: Tiba di Lantai ")); Serial.println(floor);
159
160   // Bunyi berbeda untuk lantai teratas
161   if (floor == NUM_FLOORS-1) {
162     tone(BUZZER_PIN, 2000, 300); // Bunyi khusus lantai teratas
```

```
163     delay(400);
164     tone(BUZZER_PIN, 2000, 300);
165   } else {
166     tone(BUZZER_PIN, 1500 + (floor * 100), 150);
167     delay(250);
168   }
169 }
170
171 // fungsi kalkulasi dan inisialisasi
172
173 float calculateAltitude(float pressure, float reference, float tempC) {
174   if (pressure <= 0 || reference <= 0 || isnan(tempC)) return 0.0;
175   float tempK = tempC + 273.15;
176   return ((R * tempK) / (g * M)) * log(reference / pressure);
177 }
178
179 void initializeHardware() {
180   pinMode(BUTTON_PIN, INPUT_PULLUP);
181   pinMode(CALIBRATE_PIN, INPUT_PULLUP);
182   pinMode(BUZZER_PIN, OUTPUT);
183
184   if (!bmp.begin(0x76)) {
185     Serial.println(F("ERROR: Sensor BMP280 tidak ditemukan!"));
186     while (1) { tone(BUZZER_PIN, 1000, 200); delay(500); }
187   }
188
189   bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
Adafruit_BMP280::SAMPLING_X2,
190                   Adafruit_BMP280::SAMPLING_X16,
Adafruit_BMP280::FILTER_X16,
191                   Adafruit_BMP280::STANDBY_MS_125);
192
193   // Lakukan tes bacaan sensor di awal
194   float testTemp = bmp.readTemperature();
195   float testPress = bmp.readPressure() / 100.0f;
196   if (!isSensorDataValid(testPress, testTemp)) {
197     Serial.println(F("ERROR: Tes sensor awal gagal!"));
198     while (1) { tone(BUZZER_PIN, 1500, 200); delay(300); }
199   }
200 }
201
202 void loadSystemState() {
```

```
203   basePressure = readPressureFromEEPROM();
204   currentFloor = readFloorFromEEPROM();
205
206   if (basePressure < 800.0 || basePressure > 1100.0) {
207     calibrateSensor(); // Lakukan kalibrasi pertama jika data EEPROM tidak valid
208   }
209   if (currentFloor < 0 || currentFloor >= NUM_FLOORS) {
210     currentFloor = 0; // Reset ke lantai 0 jika data EEPROM tidak valid
211   }
212 }
213
214 // fungsi bantu dan tombol
215
216 void handleButtons() {
217   // Tombol untuk START/STOP deteksi otomatis
218   if (digitalRead(BUTTON_PIN) == LOW) {
219     if (millis() - lastButtonPress > DEBOUNCE_DELAY) {
220       lastButtonPress = millis();
221       toggleDetection();
222       while (digitalRead(BUTTON_PIN) == LOW) delay(10);
223     }
224   }
225
226   // Tombol untuk kalibrasi cerdas
227   if (digitalRead(CALIBRATE_PIN) == LOW) {
228     if (millis() - lastButtonPress > DEBOUNCE_DELAY) {
229       lastButtonPress = millis();
230       calibrateSensor();
231       while (digitalRead(CALIBRATE_PIN) == LOW) delay(10);
232     }
233   }
234 }
235
236 void toggleDetection() {
237   isDetecting = !isDetecting;
238   Serial.print(F("STATUS: Deteksi Otomatis "));
239   Serial.println(isDetecting ? F("DIMULAI") : F("DIHENTIKAN"));
240   tone(BUZZER_PIN, isDetecting ? 1200 : 800, isDetecting ? 200 : 300);
241
242   if (!isDetecting) {
243     saveFloorToEEPROM(currentFloor); // Simpan lantai terakhir saat dihentikan
244   } else {
```

```
245     // Reset altitude tracker saat mulai deteksi agar tidak salah mendeteksi arah
246     lastAltitude = -100.0;
247   }
248 }
249
250 void calibrateSensor() {
251   Serial.println(F("\nINFO: Kalibrasi dimulai..."));
252   tone(BUZZER_PIN, 1800, 100);
253
254   //Kumpulkan data sensor
255   float pressures[CALIBRATION_SAMPLES];
256   uint8_t validCount = 0;
257
258   for (uint8_t i = 0; i < CALIBRATION_SAMPLES; i++) {
259     float p = bmp.readPressure();
260     if (!isnan(p)) {
261       pressures[validCount++] = p;
262
263       // Feedback progres
264       if (i % 10 == 0) {
265         tone(BUZZER_PIN, 1500, 50);
266         delay(50);
267       }
268     }
269     delay(50);
270   }
271
272   // Hitung median untuk menghilangkan outlier
273   float medianPressure = computeMedian(pressures, validCount);
274
275   //Gunakan lantai saat ini sebagai acuan
276   float knownHeight = FLOOR_HEIGHTS[currentFloor];
277   float temperature = bmp.readTemperature();
278
279   // Hitung base pressure baru
280   float tempK = (isnan(temperature) ? 298.15 : temperature + 273.15);
281   float newBasePressure = (medianPressure/100.0f) * exp((g * M *
knownHeight)/(R * tempK));
282
283   // Simpan dan beri konfirmasi
284   basePressure = newBasePressure;
285   savePressureToEEPROM(basePressure);
```

```
286
287   Serial.print(F("SUKSES: Kalibrasi di Lantai ")); Serial.print(currentFloor);
288   Serial.print(F(". Tekanan dasar: ")); Serial.print(basePressure, 2);
289   Serial.println(F(" hPa"));
290
291   // Bunyi konfirmasi
292   for (uint8_t i = 0; i <= currentFloor; i++) {
293     tone(BUZZER_PIN, 1500 + (i * 200), 150);
294     delay(200);
295   }
296 }
297
298 // Fungsi bantu untuk menghitung median
299 float computeMedian(float values[], uint8_t n) {
300   // Implementasi sorting sederhana
301   for(uint8_t i = 0; i < n-1; i++) {
302     for(uint8_t j = i+1; j < n; j++) {
303       if(values[j] < values[i]) {
304         float temp = values[i];
305         values[i] = values[j];
306         values[j] = temp;
307       }
308     }
309   }
310
311   if(n % 2 == 0) {
312     return (values[n/2 - 1] + values[n/2]) / 2.0f;
313   } else {
314     return values[n/2];
315   }
316 }
317
318 //Fungsi Lainnya (EEPROM, Cek Validitas, Logging)
319
320 bool isValidPressure(float pressure) {
321   return (pressure >= 800.0 && pressure <= 1100.0 && !isnan(pressure));
322 }
323
324 bool isSensorDataValid(float pressure, float temperature) {
325   return (!isnan(pressure) && !isnan(temperature) &&
326           pressure >= 800.0 && pressure <= 1100.0 &&
327           temperature >= -40.0 && temperature <= 85.0);
```

```
328 }
329
330 void logSensorData(uint32_t timestamp, float rawP, float filteredP, float altitude, int
floor) {
331   Serial.print(timestamp); Serial.print(',');
332   Serial.print(rawP, 2); Serial.print(',');
333   Serial.print(filteredP, 2); Serial.print(',');
334   Serial.print(altitude, 2); Serial.print(',');
335   Serial.println(floor);
336 }
337
338 void savePressureToEEPROM(float pressure) {
339   uint16_t compressed = round(pressure * 10);
340   EEPROM.update(PRESSURE_EEPROM_ADDR, highByte(compressed));
341   EEPROM.update(PRESSURE_EEPROM_ADDR + 1, lowByte(compressed));
342 }
343
344 float readPressureFromEEPROM() {
345   uint16_t compressed = (EEPROM.read(PRESSURE_EEPROM_ADDR) << 8) |
EEPROM.read(PRESSURE_EEPROM_ADDR + 1);
346   return compressed / 10.0f;
347 }
348
349 void saveFloorToEEPROM(int floor) {
350   byte floorByte = floor & 0x0F;
351   EEPROM.update(FLOOR_EEPROM_ADDR, floorByte);
352 }
353
354 int readFloorFromEEPROM() {
355   byte floorByte = EEPROM.read(FLOOR_EEPROM_ADDR);
356   return (floorByte & 0x0F);
357 }
```