REVOLUSI IOT:

Koneksi Cerdas dengan ESP32 dan Data Real-Time
[Studi Kasus Deteksi Dini Longsor Tanah]

Ariman
Riadi Marta Dinata
Muhammad Ikrar Yamin

Revolusi IoT: Koneksi Cerdas dengan ESP32 dan Data Real-Time

ARIMAN
RIADI MARTA DINATA
MUHAMMAD IKRAR YAMIN

Diterbitkan oleh:

KAMPUS ISTN
Jl. Moh Kahfi II, Srengseng Sawah, Jagakarsa, Jakarta Selatan 12640

Terbit: September 2024

ISBN: XYZ-623-7839-XY-Z

Kata Pengantar

Dalam era teknologi yang terus berkembang, perangkat mikrokontroler seperti ESP32 telah membuka peluang baru dalam pengembangan sistem berbasis Internet of Things (IoT). ESP32 tidak hanya dikenal karena kemampuannya dalam menghubungkan berbagai sensor dan perangkat, tetapi juga karena dukungan integrasinya terhadap platform cloud seperti ThingSpeak.

Proses akuisisi data yang efisien menjadi kunci dalam memanfaatkan potensi IoT. Dengan ESP32, data dapat diambil dari sensor, seperti gyroscope dan LCD, kemudian dikirimkan secara langsung ke ThingSpeak untuk dipantau dan dianalisis. Melalui penggunaan protokol komunikasi yang sederhana, ESP32 memungkinkan pengembangan aplikasi yang dapat mengontrol dan memantau berbagai aspek lingkungan secara real-time, memberikan kemudahan dan efisiensi yang belum pernah ada sebelumnya.

Tulisan ini bertujuan untuk memberikan pemahaman yang lebih baik tentang bagaimana ESP32 dan ThingSpeak bekerja bersama dalam menciptakan solusi IoT yang inovatif. Semoga bermanfaat.

Jakarta, Oktober 2024

Penyusun

Daftar Isi

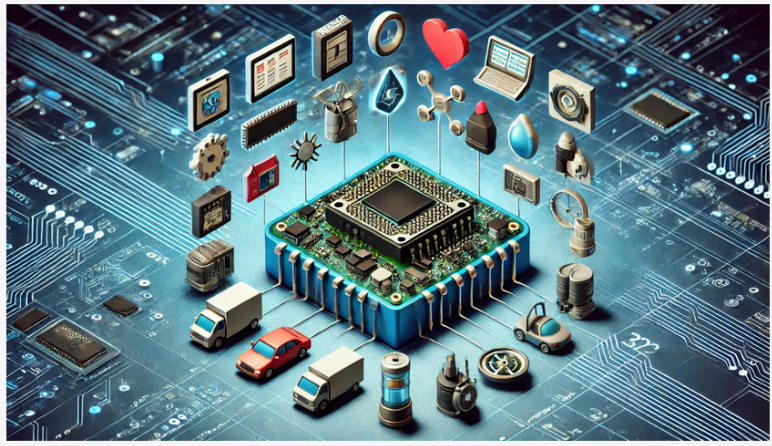
BAGIAN 1

Embedeed System, IOT, JSON Database

1.1 Embedeed System

Sistem embedded atau sistem tertanam adalah sebuah sistem komputer yang dirancang untuk melakukan fungsi tertentu dalam konteks yang lebih besar. Sistem ini biasanya terdiri dari perangkat keras dan perangkat lunak yang diintegrasikan ke dalam perangkat atau sistem lainnya untuk menjalankan tugas yang spesifik, sering kali dalam

lingkungan yang real-time. Contoh umum dari sistem embedded termasuk perangkat elektronik rumah, kendaraan, peralatan medis, dan berbagai aplikasi industri.



Arduino merupakan salah satu platform yang paling terkenal dalam kategori sistem embedded. Dengan kemudahan penggunaan dan komunitas yang besar, Arduino memungkinkan pengguna dari berbagai latar belakang untuk belajar dan berinovasi dalam bidang elektronik dan pemrograman. Arduino menyediakan berbagai jenis papan mikrokontroler yang dapat diprogram untuk berinteraksi dengan sensor, aktuator, dan perangkat lain. Ini menjadikannya pilihan yang ideal untuk prototyping dan pengembangan aplikasi embedded.

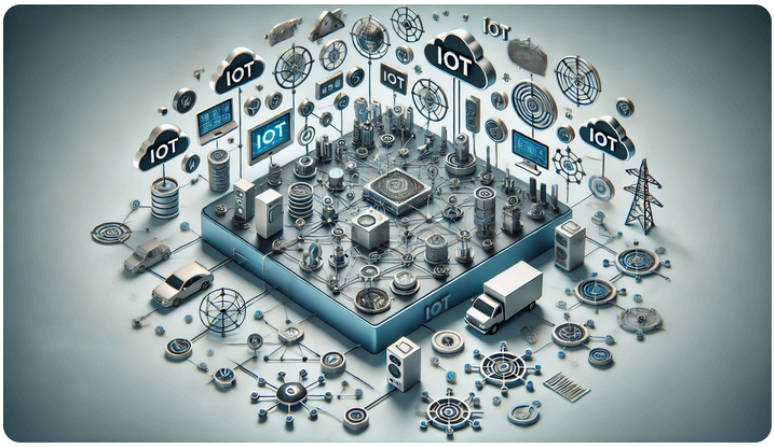
Seiring dengan perkembangan teknologi, munculnya ESP32 membawa kemampuan yang lebih lanjut untuk sistem embedded. ESP32 adalah mikrokontroler yang dilengkapi dengan fitur Wi-Fi dan Bluetooth, sehingga sangat mendukung pengembangan aplikasi Internet of Things (IoT). Dengan ESP32, pengguna dapat menghubungkan perangkat mereka ke internet dan berkomunikasi dengan sistem cloud, membuka kemungkinan baru untuk pengumpulan data, kontrol jarak jauh, dan automasi.

Ilmu tentang mikrokontroler menjadi semakin penting dalam dunia teknologi saat ini, terutama dengan meningkatnya popularitas sistem IoT. Sensor-sensor seperti sensor suhu, kelembaban, gerakan, dan banyak lagi, dapat digunakan bersama mikrokontroler untuk mengumpulkan data dari lingkungan. Aktuator, seperti motor dan relay, memungkinkan pengguna untuk mengontrol perangkat fisik berdasarkan data yang diterima dari sensor. Sistem IoT, yang mengintegrasikan semua ini, memungkinkan perangkat untuk saling berkomunikasi dan berinteraksi dengan manusia melalui aplikasi berbasis cloud, memberikan efisiensi dan kenyamanan dalam kehidupan sehari-hari.

Dengan demikian, pemahaman tentang sistem embedded, mikrokontroler seperti Arduino dan ESP32, serta kemampuan untuk memanfaatkan sensor dan aktuator dalam pengembangan sistem IoT, sangat penting untuk mempersiapkan diri menghadapi tantangan teknologi di masa depan. Pengetahuan ini tidak hanya relevan bagi insinyur dan pengembang, tetapi juga bagi siapa saja yang ingin memahami bagaimana teknologi dapat diintegrasikan ke dalam kehidupan sehari-hari.

1.2 Perkembangan IOT

Internet of Things (IoT) adalah konsep yang menggambarkan jaringan perangkat fisik yang saling terhubung dan dapat berkomunikasi satu sama lain melalui internet. Perangkat-perangkat ini dapat mencakup berbagai hal, mulai dari sensor dan aktuator hingga perangkat rumah tangga, kendaraan, dan sistem industri.



Ilustrasi 3D yang menggambarkan konsep Internet of Things (IoT). Gambar ini menunjukkan jaringan perangkat yang saling terhubung seperti sensor, aktuator, perangkat rumah tangga, kendaraan, dan sistem industri.

IoT memungkinkan pengumpulan, pertukaran, dan analisis data secara real-time, memberikan kemampuan untuk mengontrol dan memantau perangkat dari jarak jauh. Dengan demikian, IoT membuka peluang baru untuk efisiensi, otomatisasi, dan inovasi di berbagai sektor.

Peranan Arduino dan IoT

Arduino, khususnya varian seperti ESP32, telah menjadi salah satu platform yang paling populer dalam pengembangan aplikasi IoT. ESP32 adalah mikrokontroler

yang dilengkapi dengan kemampuan Wi-Fi dan Bluetooth, memungkinkan perangkat untuk terhubung ke internet dengan mudah. Berikut adalah beberapa peranan penting Arduino dalam IoT:

1. Kontrol dan Monitoring, Dengan menggunakan Arduino, pengguna dapat merancang sistem yang dapat mengontrol perangkat dari jarak jauh, serta memantau kondisi lingkungan secara real-time. Misalnya, dengan sensor suhu dan kelembaban yang terhubung ke ESP32, pengguna dapat mengawasi kondisi ruangan melalui aplikasi mobile atau web.
2. Automatisasi, Arduino memungkinkan pengguna untuk mengatur perangkat secara otomatis berdasarkan data yang dikumpulkan. Misalnya, dalam sistem pertanian pintar, sensor kelembaban tanah dapat mengontrol pompa air, sehingga irigasi terjadi secara otomatis saat tanah kering.
3. Interaksi Pengguna, Arduino dapat menghubungkan berbagai perangkat dan aplikasi, memungkinkan pengguna untuk mengontrol perangkat melalui smartphone atau antarmuka web. Hal ini

meningkatkan kenyamanan dan kemudahan penggunaan.

4. Pengumpulan Data, Dengan IoT, Arduino dapat digunakan untuk mengumpulkan dan menganalisis data dari berbagai sensor. Data ini dapat disimpan di cloud dan digunakan untuk analisis lebih lanjut, sehingga memberikan wawasan yang berharga.

Pemanfaatan IoT di Era Modern

Di zaman sekarang, pemanfaatan IoT semakin luas dan memberikan banyak peluang, antara lain :

Penggunaan IoT dalam sistem rumah pintar memungkinkan pengguna untuk mengontrol pencahayaan, suhu, keamanan, dan perangkat lainnya dari jarak jauh, meningkatkan kenyamanan dan efisiensi energi.

Di bidang kesehatan dan kebugaran, IoT juga berperan dalam pemantauan kesehatan dengan perangkat yang dapat melacak aktivitas fisik, detak jantung, dan parameter kesehatan lainnya, memberikan data yang berharga kepada pengguna dan penyedia layanan kesehatan. Dalam industri dan manufaktur, IoT membantu dalam pemantauan peralatan, manajemen rantai pasokan, dan otomatisasi

proses, yang dapat meningkatkan produktivitas dan mengurangi biaya.

Dalam sektor pertanian, IoT memungkinkan petani untuk memantau kondisi tanah, cuaca, dan kesehatan tanaman secara real-time, sehingga dapat meningkatkan hasil panen dan mengoptimalkan penggunaan sumber daya.

IoT juga mendukung sistem transportasi pintar, seperti pengelolaan lalu lintas dan pelacakan kendaraan, yang dapat meningkatkan efisiensi dan mengurangi kemacetan.

Dengan perkembangan teknologi yang cepat dan semakin banyaknya perangkat yang terhubung, peranan Arduino dalam IoT akan terus berkembang, menawarkan peluang inovasi yang lebih besar di berbagai sektor. Keberadaan platform seperti Arduino dan ESP32 memberikan kemudahan bagi pengembang, pelajar, dan penggemar untuk mengeksplorasi potensi IoT dan menciptakan solusi yang dapat memberikan dampak positif di masyarakat.

1.3 Database (FREE dan ONLINE)

Database adalah sistem yang digunakan untuk menyimpan, mengelola, dan mengakses data secara

terstruktur. Data dalam database disimpan dalam tabel-tabel yang terdiri dari baris dan kolom, yang memudahkan pengguna untuk melakukan pencarian, penyortiran, dan pengolahan informasi. Database memungkinkan pengguna untuk melakukan operasi dasar seperti menambah, mengubah, menghapus, dan mengambil data, serta memberikan keamanan dan integritas data yang lebih baik.

Beragam Database Gratis

Saat ini, sudah ada banyak pilihan database gratis yang dapat digunakan oleh pengguna perangkat seperti Arduino UNO dan ESP32 dalam konteks Internet of Things (IoT). Salah satu contoh yang populer adalah ThingSpeak, yang merupakan platform berbasis cloud yang dirancang untuk mengumpulkan, menyimpan, dan menganalisis data dari berbagai sensor IoT. ThingSpeak memungkinkan pengguna untuk:

- Menyimpan Data, Pengguna dapat dengan mudah mengirimkan data dari perangkat seperti Arduino atau ESP32 ke ThingSpeak, di mana data tersebut akan disimpan di cloud.

- Melihat Data, Platform ini menyediakan antarmuka visual yang memungkinkan pengguna untuk melihat data yang telah disimpan dalam bentuk grafik dan tabel, sehingga memudahkan dalam pemantauan tren dan analisis.
- Ekspor Data, ThingSpeak memungkinkan pengguna untuk mengekspor data yang telah dikumpulkan ke dalam format yang dapat digunakan dengan alat eksternal lainnya, seperti spreadsheet atau perangkat lunak analisis data.
- Analisis Data, Data yang disimpan di ThingSpeak dapat dianalisis menggunakan alat eksternal, termasuk Tools Open Source, yang terintegrasi dengan platform ini. Hal ini memberikan kemampuan untuk melakukan analisis lebih lanjut pada data yang telah dikumpulkan.

Manfaat Database dalam Pengolahan Big Data dan Machine Learning

Penggunaan database dalam konteks IoT dan analisis data memiliki banyak manfaat, terutama dalam pengolahan big data dan pengembangan model machine learning:

Database memiliki peran penting sebagai arsip data yang dapat diakses dan dianalisis di masa depan. Data historis yang disimpan dalam database dapat dimanfaatkan untuk analisis tren, pola, dan perilaku yang sangat berharga dalam pengambilan keputusan. Seiring dengan kemajuan teknologi, data yang dihasilkan oleh perangkat IoT dapat mencapai volume yang sangat besar. Dalam konteks ini, database memungkinkan penyimpanan dan pengelolaan data tersebut secara efisien, sehingga memudahkan dalam pengolahan big data untuk mendapatkan wawasan yang berguna.

Ilustrasi yang menggambarkan konsep database dalam konteks IoT. Gambar ini menunjukkan database cloud dengan aliran data dari berbagai perangkat IoT, serta

elemen visual seperti grafik dan diagram yang mewakili analisis data dan pemantauan waktu nyata.

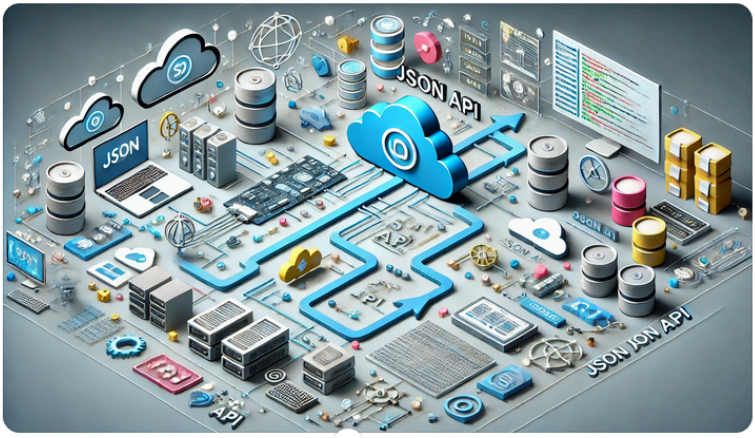
Selain itu, data yang disimpan dalam database dapat digunakan untuk melatih model machine learning. Dengan memiliki data yang terstruktur dan terorganisir, proses pelatihan model menjadi lebih efisien dan akurat. Hal ini pada akhirnya dapat meningkatkan performa model dalam membuat prediksi atau klasifikasi.

Banyak database modern juga memungkinkan analisis data secara real-time, yang sangat penting dalam aplikasi IoT. Dengan kemampuan ini, pengguna dapat mengambil tindakan cepat berdasarkan data yang diterima tanpa harus menunggu analisis mendalam yang memakan waktu. Database sering kali dapat diintegrasikan dengan berbagai alat analisis dan visualisasi, memberikan fleksibilitas bagi pengguna untuk memilih alat yang paling sesuai dengan kebutuhan mereka.

Dengan berbagai database gratis yang tersedia, pengguna perangkat IoT seperti Arduino dan ESP32 kini memiliki akses lebih mudah untuk menyimpan dan menganalisis data mereka. Ini membuka peluang baru untuk eksplorasi dan inovasi dalam bidang teknologi dan ilmu data.

1.4 Akuisisi Data/ JSON API

Kini telah memasuki era modern di mana data dapat dipindahkan secara fleksibel antara berbagai platform dan perangkat lunak. Proses ini dikenal sebagai akuisisi data, yang memungkinkan aliran informasi dari satu sistem ke sistem lainnya. Misalnya, data dapat dialihkan dari software X ke software Y, atau dari program A ke program B. Hal ini memberikan kebebasan bagi pengguna untuk mengintegrasikan berbagai aplikasi dan sistem tanpa batasan yang signifikan.



Ilustrasi 3D yang menggambarkan konsep akuisisi data dan JSON API di era modern, di mana data dapat dipindahkan dengan fleksibel antara berbagai platform dan perangkat lunak. Gambar ini menunjukkan berbagai perangkat dan

aplikasi perangkat lunak yang saling terhubung dengan aliran data yang diwakili oleh panah atau garis. Termasuk elemen-elemen seperti penyimpanan cloud, database, dan ikon perangkat lunak yang berbeda untuk menunjukkan integrasi data. Warna dan desain gambar mencerminkan tema modern dan teknologi, dengan penekanan pada konektivitas dan aliran data.

Akuisisi data melibatkan pengambilan, pengolahan, dan pemindahan data dari satu sumber ke sumber lainnya. Ini dapat dilakukan dalam berbagai format, termasuk XML dan JSON API. Kedua format ini sangat berguna dalam konteks pertukaran data antar aplikasi, tetapi JSON API semakin populer karena kemudahan penggunaannya.

JSON API

JSON (JavaScript Object Notation) adalah format yang ringan dan mudah dibaca untuk pertukaran data. JSON API secara khusus digunakan untuk mengakses dan menyimpan data melalui protokol HTTP.

JSON memiliki sejumlah keunggulan yang membuatnya populer dalam pengembangan perangkat lunak. Salah satu keunggulannya adalah struktur yang sederhana dan mudah

dipahami. Dengan format yang dapat dibaca oleh manusia, JSON memudahkan pengembang dalam memahami dan menggunakan data yang dipertukarkan antara sistem. Keunggulan lainnya adalah kompatibilitas luas, di mana JSON API didukung oleh hampir semua bahasa pemrograman modern, sehingga dapat digunakan pada berbagai platform tanpa memerlukan konversi yang rumit.

Selain itu, JSON dikenal karena kecepatannya. Formatnya yang lebih ringan dibandingkan dengan XML memungkinkan transfer data yang lebih cepat, hal ini menjadi sangat penting dalam aplikasi real-time yang membutuhkan respons cepat. JSON juga mendukung pengorganisasian data yang kompleks, termasuk kemampuan untuk menyimpan array dan objek bersarang, menjadikannya ideal untuk aplikasi yang memerlukan struktur data yang kaya.

Dalam konteks penyimpanan dan pemantauan data, JSON sudah umum digunakan, baik untuk aplikasi web maupun aplikasi berbasis IoT. Format ini memudahkan pengiriman dan penerimaan data antar perangkat, serta memfasilitasi pemantauan status atau kinerja sistem secara real-time. Selain itu, JSON juga mendukung integrasi yang mudah antara berbagai layanan dan alat analisis, sehingga

memungkinkan penggabungan data dari sumber yang berbeda untuk mendapatkan wawasan yang lebih mendalam dan mendukung pengambilan keputusan berbasis data.

Akuisisi Data JSON API

Penggunaan JSON API dalam akuisisi data memberikan banyak manfaat, antara lain:

- Interoperabilitas, JSON API meningkatkan interoperabilitas antar sistem yang berbeda, memungkinkan pengembang untuk menghubungkan berbagai aplikasi dengan mudah.
- Skalabilitas, Dengan kemampuan untuk menangani berbagai jenis data dan struktur yang kompleks, JSON API dapat dengan mudah diadaptasi untuk kebutuhan aplikasi yang berkembang.
- Dukungan untuk Mobile dan Web, JSON API sangat sesuai untuk aplikasi web dan mobile, memberikan pengalaman pengguna yang responsif dan cepat.

Dengan semua keuntungan ini, jelas bahwa akuisisi data melalui JSON API tidak hanya menyederhanakan pertukaran data tetapi juga membuka banyak peluang baru untuk inovasi dan pengembangan di berbagai bidang,

termasuk Internet of Things (IoT), analisis data, dan aplikasi berbasis cloud.

1.5 Gyroscope MEMS

Gyroscope MEMS adalah evolusi modern yang menggunakan elemen mikroskopis untuk mendeteksi rotasi. Berbeda dengan gyroscope tradisional yang mengandalkan roda berputar, gyroscope MEMS bekerja dengan prinsip efek Coriolis.

Komponen Utama Gyroscope MEMS:

1. Massa Bergetar (Vibrating Mass):
    - Elemen mikroskopis yang dirancang untuk bergetar secara konstan pada frekuensi tertentu.
2. Elektroda Kapasitif:
    - Digunakan untuk mendeteksi perubahan posisi atau gerakan massa akibat rotasi.
3. Sensor Elektronik:
    - Mengukur perubahan kapasitansi dan mengubahnya menjadi data rotasi pada sumbu X, Y, dan Z.

Prinsip Efek Coriolis:

- Ketika massa bergetar digerakkan dalam arah tertentu dan alat mengalami rotasi, massa tersebut akan merasakan gaya Coriolis. Gaya ini menggeser massa dalam arah tegak lurus terhadap getarannya.

$$F_c = 2m(v \times \omega)$$

- Rumus Efek Coriolis:
    - Fc: Gaya Coriolis
    - m: Massa
    - v: Kecepatan getaran
    - ω: Kecepatan sudut (rotasi)

Cara Kerja Internal Gyroscope MEMS

1. Getaran Awal:
    - Massa bergetar secara konstan dalam arah tertentu dengan frekuensi tertentu yang dihasilkan oleh elektroda internal.
2. Rotasi Sensor:
    - Ketika alat mengalami rotasi, gaya Coriolis menyebabkan massa bergetar bergeser ke arah tegak lurus terhadap arah getaran awal.
3. Pengukuran Pergeseran:
    - Pergeseran massa diukur oleh elektroda kapasitif sebagai perubahan kapasitansi.

4. Pengolahan Data:
    - Data kapasitansi mentah diproses oleh ASIC untuk menentukan kecepatan sudut (ω\omegaω) pada masing-masing sumbu (X, Y, Z).
5. Pengeluaran Data:
    - Data rotasi dalam bentuk digital atau analog dikirimkan ke perangkat pengolah data, seperti microcontroller (Arduino atau ESP32).

Contoh Sensor Gyroscope MEMS

1. MPU6050:
    - Gyroscope dan accelerometer 6 sumbu dalam satu chip.
    - Rentang gyroscope: ±250°/s hingga ±2000°/s.
2. MPU9250:
    - Kombinasi gyroscope, accelerometer, dan magnetometer 9 sumbu.
    - Cocok untuk aplikasi yang membutuhkan navigasi presisi tinggi.
3. L3GD20H:
    - Gyroscope 3 sumbu dengan sensitivitas tinggi.

- Digunakan dalam drone dan perangkat stabilisasi.

Gyroscope menjadi lebih populer karena ukurannya yang kecil, efisiensi biaya, dan fleksibilitas dalam berbagai aplikasi modern. Juga dengan efek Coriolisnya, gyroscope MEMS dapat mendeteksi rotasi pada tiga sumbu dengan akurasi tinggi.

Nilai X, Y, Z pada accelerometer dan gyroscope memungkinkan perangkat untuk memahami gerakan dan orientasi dalam ruang tiga dimensi. Accelerometer berfokus pada gerakan linier seperti perubahan posisi dan percepatan, sedangkan gyroscope mendeteksi rotasi sudut. Kombinasi kedua sensor ini telah membuka pintu untuk berbagai inovasi teknologi, mulai dari stabilisasi drone hingga perangkat wearable untuk kesehatan.

1.6 Prinsip Deteksi Gempa

Meski gyroscope memiliki banyak keunggulan, penggunaannya menjadi lebih efektif bila dikombinasikan dengan accelerometer. Berikut adalah alasan mengapa kedua sensor ini saling melengkapi:

- Accelerometer: Mengukur percepatan linier akibat gelombang gempa.

- Gyroscope: Mengukur rotasi sudut akibat gerakan melingkar atau rotasi selama gempa.
- Hasil Gabungan: Data yang lebih akurat dan lengkap untuk mendeteksi arah, pola, dan intensitas gempa.

Gyroscope dan accelerometer digunakan bersama untuk mendeteksi gempa dengan mengukur percepatan linier dan rotasi. Prinsip ini melibatkan pengolahan data sensor untuk menghitung magnitudo total dari kedua parameter, membandingkan hasil dengan ambang batas tertentu, dan menganalisis pola gerakan untuk menentukan tingkat gempa.

1. Menggunakan Percepatan Linier dan Rotasi untuk Mendeteksi Getaran

1.1 Percepatan Linier (Accelerometer)

- Accelerometer mengukur percepatan linier pada tiga sumbu (X, Y, Z).
- Magnitudo percepatan total dihitung menggunakan rumus:

$$\text{accel\_magnitude} = \sqrt{ax^2 + ay^2 + az^2}$$

1.2 Rotasi Sudut (Gyroscope)

- Gyroscope mengukur kecepatan sudut pada tiga sumbu (X, Y, Z).
- Magnitudo rotasi total dihitung menggunakan rumus:

$$\text{gyro\_magnitude} = \sqrt{gx^2 + gy^2 + gz^2}$$

1.3 Gabungan Data dari Dua Sensor

- Untuk meningkatkan akurasi, dua sensor gyroscope dan accelerometer digunakan dan hasilnya dirata-rata:

$$\text{average\_value} = \frac{\text{sensor1\_value} + \text{sensor2\_value}}{2}$$

- Sebelum digunakan, sensor harus dikalibrasi untuk menghilangkan bias bawaan. Berikut adalah langkah-langkah kalibrasi:
- Mengukur Offset Sensor:
    - Tempatkan sensor dalam posisi statis (tanpa gerakan).
    - Catat rata-rata pembacaan selama beberapa iterasi sebagai offset.
- Mengurangi Offset:

- Gunakan nilai offset untuk mengoreksi pembacaan sensor:

$$\text{calibrated\_value} = \text{raw\_value} - \text{offset}$$

Penentuan Ambang Batas Gempa

- Percepatan Linier (accel_magnitude):
  - Normal: <2.0 g< 2.0 \, g<2.0g
  - Warning: 2.0–5.0 g2.0 - 5.0 \, g2.0–5.0g
  - Danger: >5.0 g> 5.0 \, g>5.0g
- Rotasi Sudut (gyro_magnitude):
  - Normal: <3.0°/s< 3.0°/s<3.0°/s
  - Warning: 3.0–6.0°/s3.0 - 6.0°/s3.0–6.0°/s
  - Danger: >6.0°/s> 6.0°/s>6.0°/s

Analisis Pola Gerakan

1. Normal:
   - Tidak ada perubahan signifikan pada percepatan atau rotasi.
2. Warning:
   - Perubahan moderat pada percepatan linier atau rotasi, menunjukkan gempa kecil.
3. Danger:
   - Perubahan besar pada percepatan dan rotasi, menunjukkan gempa kuat.

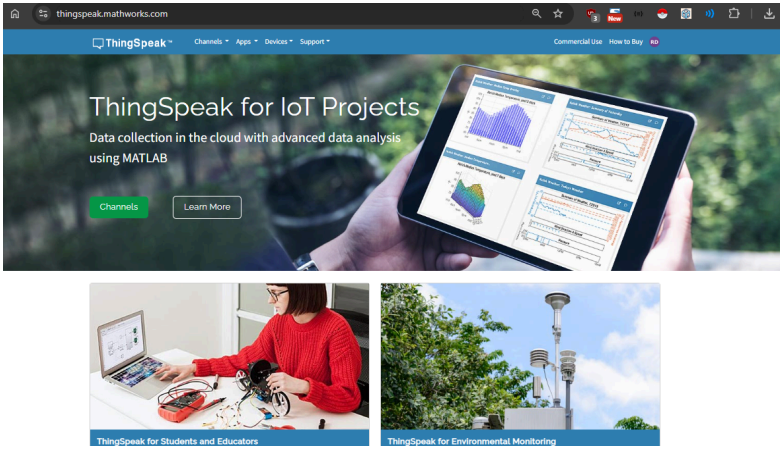
BAGIAN 2

DATABASE ONLINE + JSON API

2.1 Thinkspeak

ThingSpeak adalah platform berbasis cloud yang disediakan oleh MathWorks untuk pengumpulan, penyimpanan, analisis, dan visualisasi data dari perangkat Internet of Things (IoT). Platform ini memungkinkan pengguna untuk mengirimkan data dari berbagai sensor atau perangkat IoT secara real-time, serta menyimpan dan mengolah data tersebut dengan berbagai alat analitik yang tersedia, termasuk MATLAB, PYTHON, JAVA dan sebagainya.

Platform ThingSpeak menyediakan fitur untuk memantau dan mengelola proyek IoT, memungkinkan pengembang, peneliti, atau pengguna umum untuk memantau kondisi fisik dari perangkat sensor yang mereka gunakan. Selain itu, ThingSpeak dapat memfasilitasi interaksi antar perangkat IoT melalui cloud, sehingga data dapat dianalisis dan dilihat secara real-time dari mana saja.



Fitur Dan Kelebihan ThingSpeak:

ThingSpeak memiliki berbagai fitur utama yang menjadikannya platform yang andal untuk proyek IoT. Salah satu fitur utamanya adalah kemampuannya untuk mengumpulkan dan menyimpan data secara real-time dari berbagai sensor IoT. Data dapat dikirimkan melalui protokol HTTP atau MQTT dari perangkat seperti sensor suhu, kelembaban, gerak, atau perangkat lainnya.

Selain itu, ThingSpeak menyediakan fitur visualisasi data dalam bentuk grafik yang dapat dilihat langsung di dashboard. Visualisasi ini memudahkan pemantauan tren atau perubahan kondisi sensor yang digunakan. ThingSpeak juga terintegrasi dengan MATLAB, yang

memungkinkan analisis lebih lanjut pada data yang dikumpulkan. Pengguna dapat menggunakan MATLAB untuk melakukan perhitungan statistik, pemodelan, atau analisis data secara otomatis.

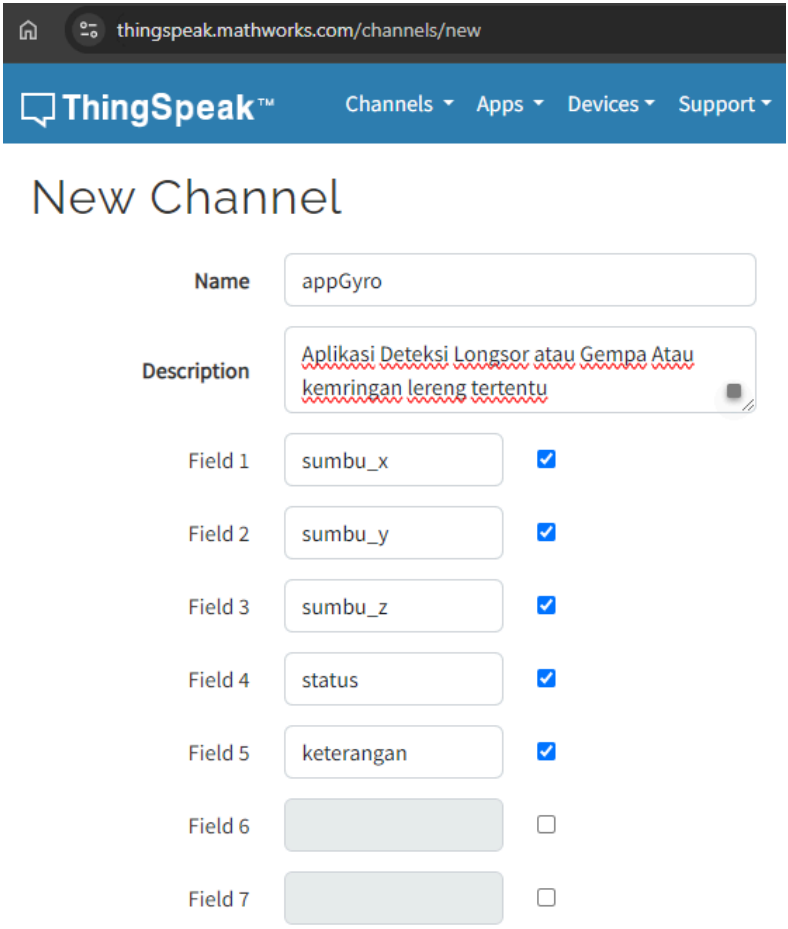
ThingSpeak menyediakan akses melalui API JSON, yang memungkinkan data untuk diambil secara online dalam format terstruktur. Ini memudahkan integrasi dengan aplikasi lain. Data yang dikirimkan ke ThingSpeak disimpan di cloud, memberikan fleksibilitas penyimpanan online gratis untuk penggunaan non-komersial. Setiap akun memiliki batasan penggunaan yang cukup untuk pengembangan kecil atau uji coba IoT sederhana.

Platform ini juga memungkinkan interaksi antar perangkat IoT melalui aturan yang dapat diatur oleh pengguna. Automasi ini memungkinkan satu perangkat mengirimkan perintah ke perangkat lain berdasarkan kondisi tertentu. ThingSpeak memiliki beberapa kelebihan, di antaranya adalah kemudahan penggunaan berkat antarmuka yang intuitif dan sederhana, membuatnya ideal untuk pemula atau pengguna berpengalaman.

Contoh Penggunaan ThingSpeak:

- Pemantauan Kualitas Udara, Data dari sensor kualitas udara dapat dikirimkan secara real-time ke ThingSpeak, di mana pengguna dapat memvisualisasikan data dan menganalisis tingkat polusi menggunakan Tools External.
- Sistem Pemantauan Cuaca, Pengguna dapat menggunakan ThingSpeak untuk mengirimkan data cuaca seperti suhu, kelembaban, dan tekanan dari stasiun cuaca ke cloud. Data ini bisa diakses dan dianalisis untuk memprediksi cuaca.
- Pemantauan Energi, ThingSpeak dapat digunakan untuk memantau penggunaan energi pada perangkat rumah tangga atau kantor secara real-time, sehingga membantu manajemen energi yang lebih efisien;

Daftar dan buat channel baru di **https://thingspeak.mathworks.com/channels/new** untuk memulai proyek IoT.



Berikut adalah penjelasan terkait kegunaan dari setiap fitur isian pada ThingSpeak:

Channel ID:

ID yang dihasilkan secara otomatis ini berfungsi sebagai

identitas unik dari sebuah channel di ThingSpeak. Aplikasi menggunakan Channel ID ini untuk membaca atau mengakses data yang ada dalam channel tersebut. Karena ID ini bersifat unik dan berkaitan langsung dengan channel yang dibuat, nilainya tidak bisa diubah setelah channel tersebut dibuat.

Name:

Nama yang diberikan untuk channel di ThingSpeak membantu dalam mengidentifikasi channel tersebut, terutama jika terdapat banyak channel. Pemilihan nama yang deskriptif dan jelas sangat penting agar memudahkan identifikasi channel di antara berbagai channel yang dikelola.

Description:

Deskripsi berfungsi sebagai kolom informasi untuk menjelaskan secara singkat tujuan atau fungsi dari channel yang telah dibuat. Deskripsi ini memudahkan pengguna atau pengunjung channel untuk memahami jenis data yang dikumpulkan oleh channel tersebut.

Field#:

ThingSpeak memungkinkan hingga delapan fields per channel. Centang opsi untuk mengaktifkan field yang akan

digunakan, kemudian masukkan nama untuk setiap field. Field ini biasanya menyimpan data dari berbagai perangkat IoT atau sensor yang berbeda, seperti sensor suhu, kelembaban, atau tekanan.

Metadata:

Kolom metadata berfungsi untuk memasukkan informasi tambahan mengenai data yang dikumpulkan di dalam channel, yang dapat berupa format **JSON**, **XML**, atau **CSV**. Metadata memberikan konteks lebih lanjut tentang data yang disimpan, sehingga data lebih mudah dipahami atau digunakan di aplikasi lain.

Tags:

Tags berguna untuk mengidentifikasi dan mengelompokkan channel berdasarkan kata kunci yang relevan dengan fungsinya. Tags berupa kata atau frasa yang dipisahkan oleh koma dapat ditambahkan untuk memudahkan pencarian atau penyortiran channel di masa mendatang.

Link to External Site:

Jika terdapat situs web eksternal yang terkait dengan channel di ThingSpeak, URL situs tersebut dapat dimasukkan di bagian ini. Link ini akan ditampilkan di

bagian **More Information** pada tampilan publik atau privat channel.

Elevation:

Elevation digunakan untuk mencatat ketinggian atau posisi sensor dari permukaan laut dalam meter. Informasi ini berguna dalam analisis data yang terkait dengan kondisi lingkungan, terutama untuk sensor yang memerlukan informasi ketinggian.

Show Channel Location:

Dengan mencentang kotak ini, informasi lokasi channel dapat dimasukkan. Ketika diaktifkan, peta yang menunjukkan lokasi sensor akan ditampilkan di tampilan channel. Lokasi ini bersifat statis, namun data individu dalam channel juga dapat menyertakan informasi lokasi tambahan.

Latitude dan Longitude:

Bagian ini digunakan untuk memasukkan informasi geografis dari lokasi sensor atau perangkat dalam bentuk derajat desimal. Latitude dan longitude membantu menandai posisi fisik sensor yang digunakan di channel.

Show Video:

Jika channel memerlukan visualisasi video, seperti video

real-time dari sensor atau panduan, video dari YouTube atau Vimeo dapat ditambahkan. Cukup masukkan URL video di kolom yang tersedia untuk menampilkan video di tampilan channel.

Link to GitHub:
Jika kode yang terkait dengan channel ThingSpeak disimpan di GitHub, URL repositori GitHub dapat dimasukkan di bagian ini. Ini berguna untuk mendokumentasikan atau berbagi kode sumber dengan pengguna lain.

Show Status:
Dengan mencentang kotak ini, jendela status akan ditampilkan di tampilan channel. Jendela ini dapat menampilkan informasi tentang perintah yang dieksekusi atau status dari API yang terhubung ke channel. Status juga dapat diperbarui melalui parameter status di API.

Pengaturan Privasi Channel:
Secara default, channel di ThingSpeak bersifat privat sehingga hanya bisa diakses menggunakan Read API Key. Namun, channel dapat diubah menjadi publik agar pengguna lain bisa melihat feed tanpa memerlukan API Key. Ada juga opsi Shared Channel untuk berbagi channel

dengan pengguna tertentu yang hanya dapat melihat tampilan privat tanpa akses tambahan.

Plugin dan Visualisasi:
ThingSpeak mendukung penggunaan plugin untuk visualisasi data dan analisis lebih lanjut. Plugin ini bisa diaktifkan di tampilan publik maupun privat dari channel, untuk memudahkan interpretasi data.

Tags for Sorting:
Tags juga berfungsi untuk menyortir dan mengatur channel publik dan privat. Saat mencari channel, tag dapat digunakan untuk memfilter channel berdasarkan kata kunci yang ditentukan.

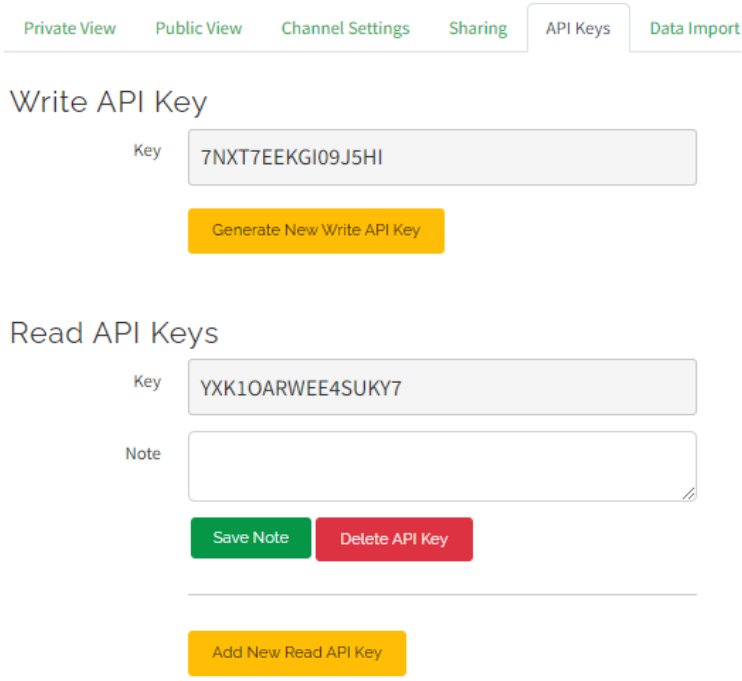
2.2 Fitur

ThingSpeak sebagai platform yang kuat dan fleksibel untuk pengelolaan data IoT adalah mampu menggabungkan pengumpulan, penyimpanan, analisis, dan visualisasi dalam satu sistem. Keunggulan lainnya adalah kemampuan integrasi dengan TOOLS External lainnya, penyimpanan data secara gratis, dan akses data melalui JSON API. Semua ini menjadikannya alat yang ideal untuk pengembang IoT, pelajar, dan peneliti yang ingin

memanfaatkan data sensor dan perangkat IoT dalam proyek mereka.

API Keys dalam platform ThingSpeak digunakan untuk memberikan akses kontrol pada pengguna untuk membaca atau menulis data dari sebuah channel. Ketika membuat channel baru, API Keys ini akan otomatis dihasilkan, dan dapat digunakan untuk mengakses data yang ada dalam channel tersebut. API Keys memastikan bahwa akses terhadap data dapat dikendalikan dengan aman, baik untuk keperluan membaca data yang bersifat pribadi maupun menulis data baru ke channel.

Artinya API Keys ini berfungsi sebagai alat otorisasi saat menulis atau membaca data dari ThingSpeak. Write API Key memberikan akses kepada pengguna untuk memperbarui data dalam channel, sementara Read API Key memungkinkan pengguna atau pihak lain untuk membaca data dari channel yang bersifat pribadi. ThingSpeak mendukung API berbasis JSON, yang memudahkan pengguna atau aplikasi lain untuk mengambil data secara online dalam format yang terstruktur. Keamanan menjadi aspek yang penting, dan ThingSpeak menyediakan opsi bagi pengguna untuk menghasilkan kunci baru jika kunci yang lama dirasa telah terkompromikan.



Bagian detail API Keys:

Write API Key berfungsi sebagai kunci yang digunakan untuk menulis atau mengirim data ke channel. Setiap kali pengguna ingin memperbarui data dalam channel, mereka perlu menggunakan Write API Key. Dalam hal keamanan, jika pengguna merasa bahwa kunci ini telah diketahui oleh orang yang tidak berhak, mereka dapat menghasilkan kunci baru dengan mengganti kunci yang lama melalui opsi Generate New Write API Key.

Read API Key digunakan untuk membaca data dari channel yang bersifat pribadi. Pengguna dapat memberikan Read API Key ini kepada pihak lain yang ingin melihat data tanpa mengubahnya. Selain itu, pengguna juga memiliki opsi untuk membuat kunci tambahan bagi pengguna lain dengan mengklik Generate New Read API Key. Mereka dapat menambahkan catatan untuk memantau siapa saja yang memiliki akses terhadap channel tersebut.

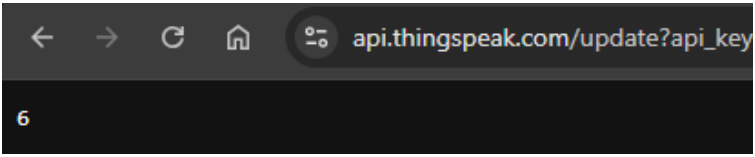
2.3 Pengujian

Berikut adalah beberapa contoh API Requests yang bisa digunakan dalam ThingSpeak untuk mengelola dan mengakses data:

1.Menulis Data ke Channel (Write a Channel Feed), Contoh Request:

```
GET
https://api.thingspeak.com/update?api_key=7NXT7EEKG
I09J5HI&field1=0
```

Di sini, Write API Key digunakan untuk mengirim atau memperbarui data pada channel di field1 dengan nilai 0. Setiap kali sistem mengirim data baru, API Key yang sesuai harus digunakan untuk otorisasi.



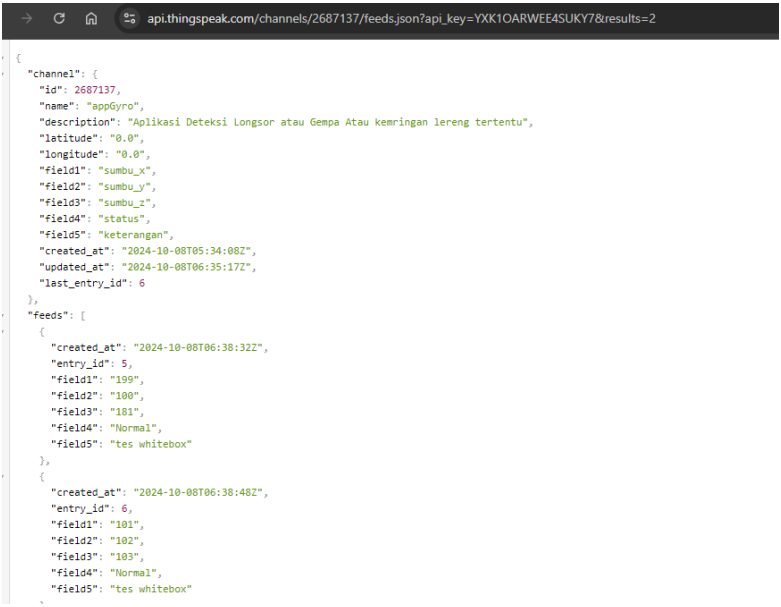
https://api.thingspeak.com/update?api_key=7NXT7EEKGI09J5HI&field1=101&&field2=102&&field3=103&field4=Normal&field5=tes%20whitebox

Hasil out dengan kembalian data "6" menunjukkan data sudah ada di data thinkspeak adalah sukses masuk data index 6.

2.Membaca Data dari Channel (Read a Channel Feed), Contoh Request:

```
GET
https://api.thingspeak.com/channels/2687137/feeds.json
?api_key=YXK1OARWEE4SUKY7&results=2
```

Read API Key digunakan untuk membaca data dari channel yang bersifat pribadi. Dalam hal ini, hanya dua hasil terakhir dari channel yang diminta (results=2), dan data akan dikembalikan dalam format JSON.



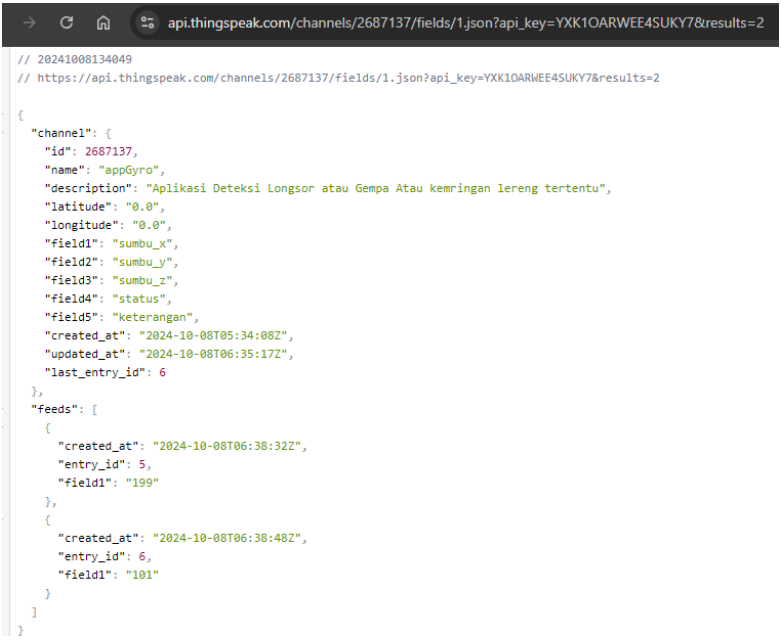
3.Membaca Data dari Field Spesifik (Read a Channel Field); Contoh Request:

GET https://api.thingspeak.com/channels/2687137/fields/1.json?api_key=YXK1OARWEE4SUKY7&results=2

Pengguna bisa membaca data dari field tertentu (misalnya field1) di channel yang bersifat pribadi. Dalam contoh ini, data dari field 1 akan diambil dalam format JSON, dan hanya dua hasil terakhir yang akan ditampilkan.

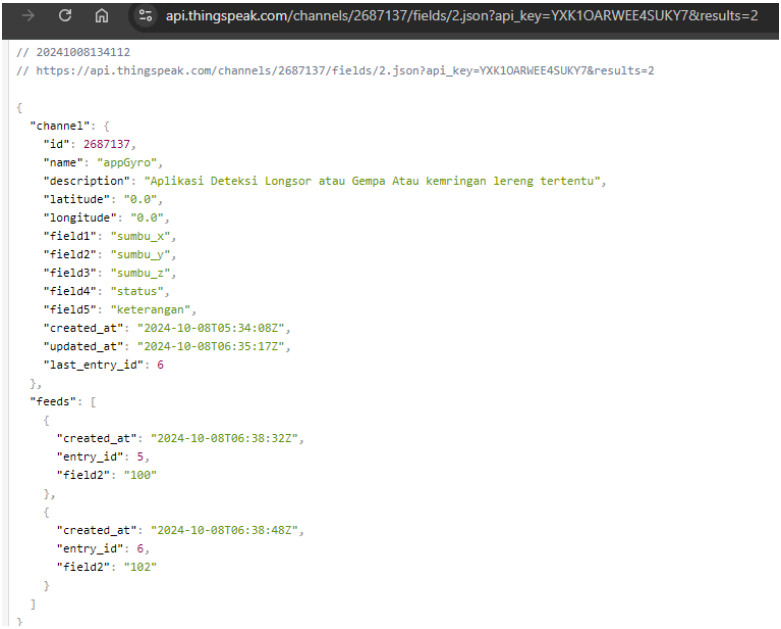
https://api.thingspeak.com/channels/2687137/fields/1.json?api_key=YXK1OARWEE4SUKY7&results=2

Hanya menampilkan field1 sebanyak 2 data terakhir :

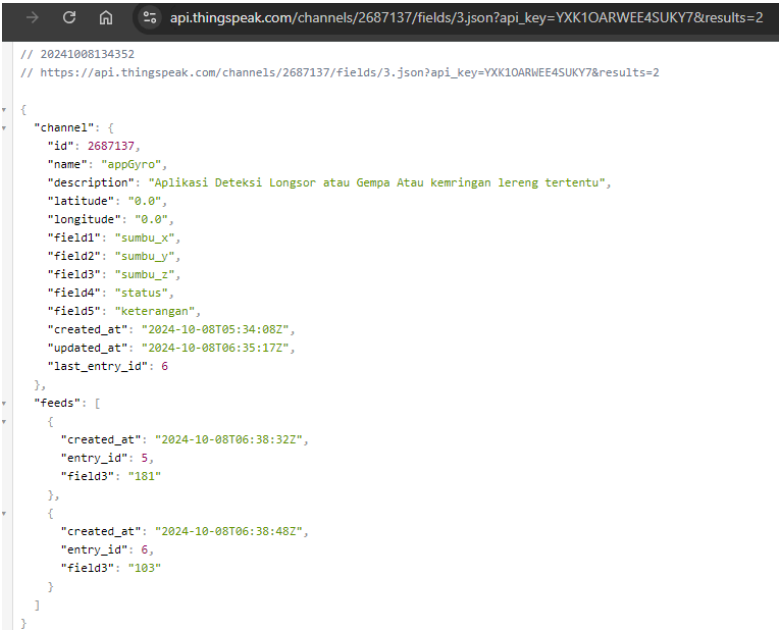


https://api.thingspeak.com/channels/2687137/fields/2.json?api_key=YXK1OARWEE4SUKY7&results=2

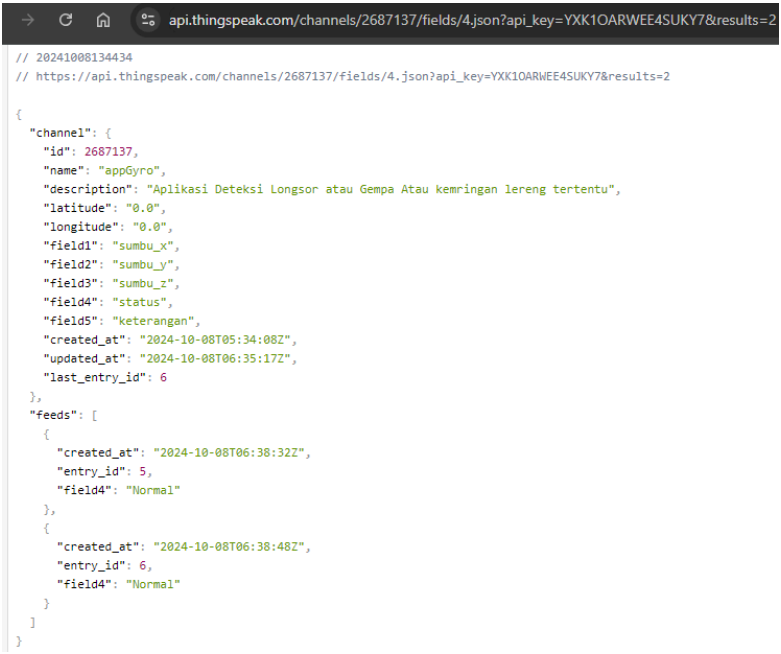
Hanya menampilkan field2 sebanyak 2 data terakhir:



https://api.thingspeak.com/channels/2687137/fields/3.json?api_key=YXK1OARWEE4SUKY7&results=2
Hanya menampilkan field3 sebanyak 2 data terakhir:



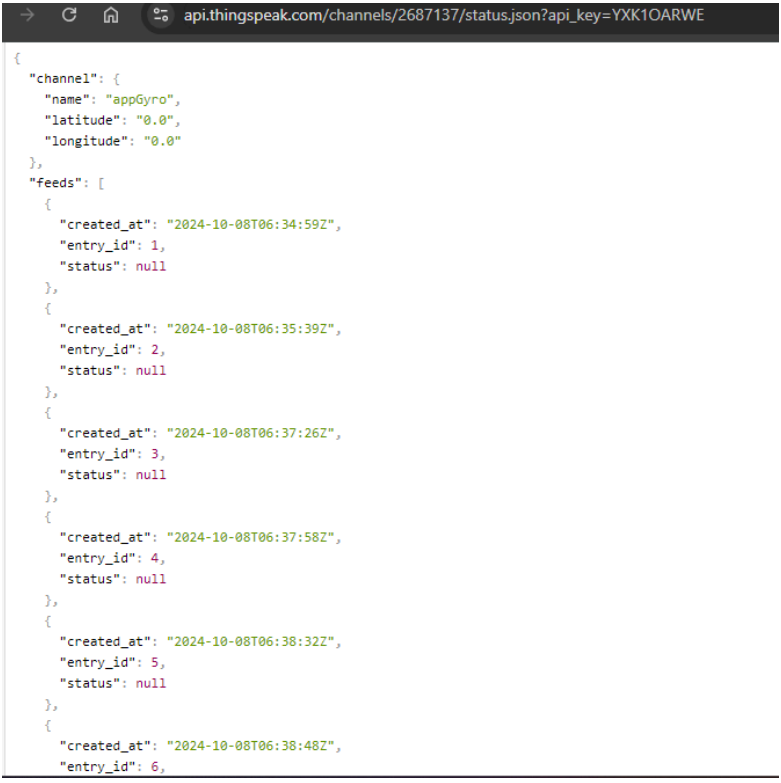
https://api.thingspeak.com/channels/2687137/fields/4.json?api_key=YXK1OARWEE4SUKY7&results=2
Hanya menampilkan field4 sebanyak 2 data terakhir:



4.Membaca Pembaruan Status Channel (Read Channel Status Updates); Contoh Request:

```
GET
https://api.thingspeak.com/channels/2687137/status.json?api_key=YXK1OARWE
```

Request ini digunakan untuk membaca status pembaruan dari channel yang bersifat pribadi, misalnya untuk mengetahui apakah ada perubahan status atau informasi terkini terkait channel tersebut.



2.4 Uji JSON

Dalam pengujian blackbox, fokus utamanya adalah menguji fungsi eksternal dari sistem tanpa memperhatikan bagaimana kode atau logika internalnya bekerja. Dalam hal ini, pengujian dilakukan untuk memastikan bahwa data dari sensor gyroscope pada sistem UNO dapat dikirim dan diterima dengan benar oleh server IoT (ThingSpeak).

Sistem ini menggunakan sensor gyroscope untuk mengukur pergerakan pada sumbu x, y, dan z. Setelah memperoleh data tersebut, sistem UNO melakukan perhitungan rata-rata nilai dari masing-masing sumbu dan mengirimkannya ke server IoT (ThingSpeak) melalui protokol HTTP GET request.

Kode PHP yang disiapkan berperan untuk mensimulasikan atau melakukan blackbox testing terhadap sistem IoT ini, dengan asumsi bahwa sistem UNO terhubung dengan ThingSpeak dan pengiriman data dilakukan melalui endpoint ThingSpeak. Kode ini mengambil nilai acak sebagai pengganti data gyroscope x, y, dan z rata-ratanya, kemudian mengirimkan nilai tersebut ke ThingSpeak melalui API yang telah ditentukan.

send.php

```
<?php
// URL ThingSpeak API
$url = 'https://api.thingspeak.com/update';

// API Key dari ThingSpeak
$api_key = '7NXT7EEKGI09J5HI';

// Generate nilai acak (random) antara 100 dan 200
$valuex = rand(100, 200);
$valuey = rand(100, 200);
$valuez = rand(100, 200);
```

```
// Parameter yang akan dikirim ke ThingSpeak
$data = array(
    'api_key' => $api_key,
    'field1' => $valuex,
            'field2' => $valuey,
            'field3' => $valuez,
            'field4' => 'Normal',
            'field5' => 'tes whitebox'
);

// Mengubah array data menjadi query string
$query_string = http_build_query($data);

// URL lengkap dengan parameter
$full_url = $url . '?' . $query_string;

// Mengirim permintaan GET ke ThingSpeak
$response = file_get_contents($full_url);

// Memeriksa respons dari ThingSpeak
if ($response === FALSE) {
    echo 'Gagal mengirim data ke ThingSpeak';
} else {
    echo 'Data berhasil dikirim: ' . $response;
}
?>
```

Fungsi code perbagian adalah sebagai berikut:

1. rand(100, 200) digunakan untuk menghasilkan nilai acak antara 100 dan 200.
2. Menggunakan http_build_query untuk membentuk query string yang akan dikirim melalui URL.
3. file_get_contents digunakan untuk mengirim request GET ke ThingSpeak.

4. $response berisi respons dari server ThingSpeak, yang bisa digunakan untuk memverifikasi apakah data berhasil dikirim atau tidak.

Setiap kali kode ini dijalankan, data acak antara 100 hingga 200 akan dikirim ke ThingSpeak pada field1, field2 dan field3. Alur pengujian adalah dimulai dengan input yang diuji berupa data gyroscope yang diperoleh dari sistem UNO. Dalam pengujian ini, nilai acak dihasilkan menggunakan fungsi rand(100, 200) untuk mensimulasikan nilai x, y, dan z dari gyroscope. Nilai acak ini dianggap mewakili rata-rata nilai gyroscope yang biasanya diperoleh dari sensor fisik pada perangkat.

Selanjutnya, proses pengujian melibatkan kode PHP yang ditulis untuk mengirimkan data acak tersebut menggunakan metode HTTP GET ke alamat ThingSpeak API. Setiap kali kode dijalankan, nilai rata-rata gyroscope x, y, dan z disimulasikan, kemudian dikirimkan ke URL endpoint ThingSpeak. Kode ini menggunakan Write API Key yang valid untuk otorisasi pengiriman data ke channel yang sesuai.

Pada tahap output, hasil yang diharapkan adalah bahwa ThingSpeak berhasil menerima data yang dikirim dan

menyimpannya di channel yang telah ditentukan. Setelah data diterima, ThingSpeak akan mengembalikan respons sebagai konfirmasi bahwa pengiriman data telah berhasil atau gagal.

Verifikasi dilakukan dengan pendekatan blackbox, di mana tidak perlu memperhatikan detail cara kerja internal kode PHP. Fokus pengujian adalah memeriksa apakah ThingSpeak telah menerima data yang dikirim. Verifikasi bisa dilakukan dengan memantau data yang muncul di dashboard ThingSpeak atau dengan menggunakan API endpoint ThingSpeak untuk membaca data yang sudah tersimpan dan memastikan pengiriman data berlangsung sesuai harapan.Adapun Tujuan Pengujian ini adalah:

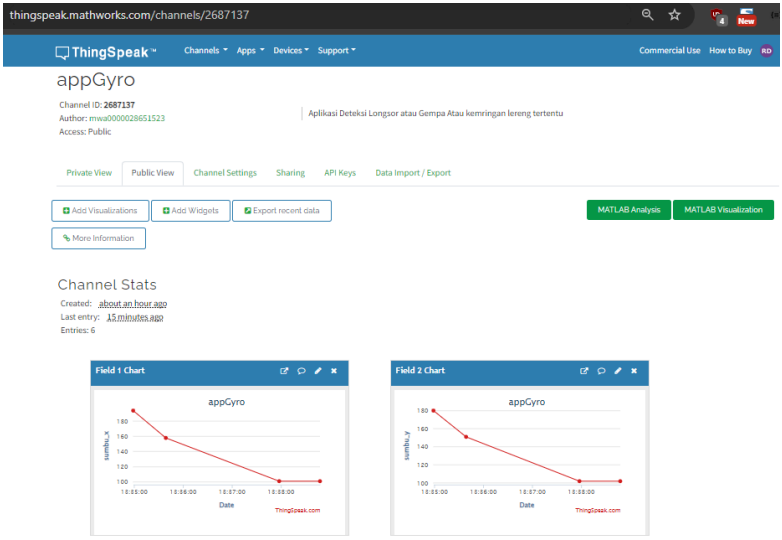
- Fungsi Utama yang Diuji, Memastikan bahwa data rata-rata gyroscope x, y, z dari sistem UNO dapat dikirimkan dengan benar ke ThingSpeak dan tersimpan di sana.
- Pengiriman Data, Memverifikasi bahwa sistem dapat melakukan pengiriman data menggunakan GET request dan menerima respons yang sesuai dari server ThingSpeak.
- Keberhasilan Pengujian, Pengujian dinyatakan berhasil jika data acak (sebagai pengganti

gyroscope x, y, z) berhasil dikirim ke ThingSpeak, disimpan, dan bisa dilihat melalui dashboard ThingSpeak atau diambil kembali melalui API Read.

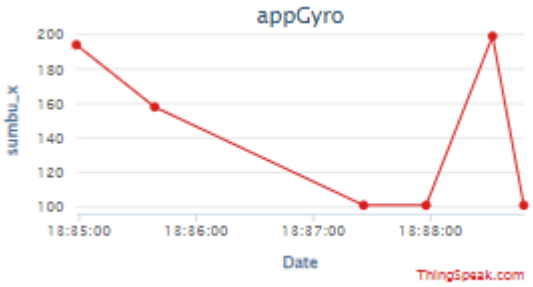
Pengujian blackbox menggunakan kode PHP ini berfokus pada aspek eksternal dari sistem UNO yang terhubung ke server IoT (ThingSpeak). Dalam pengujian ini, data rata-rata gyroscope x, y, z dikirimkan secara otomatis ke server IoT melalui API ThingSpeak, dan proses pengiriman serta penerimaan data dapat diverifikasi tanpa perlu mengetahui detail internal logika sistem yang berjalan.

2.5 View Data

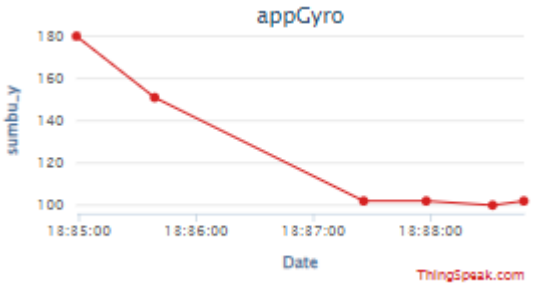
ThingSpeak tidak hanya menyediakan akses data melalui JSON API, tetapi juga menawarkan fitur visualisasi data dalam bentuk grafik yang dapat diakses melalui tautan *https://thingspeak.mathworks.com/channels/2687137*.
Melalui grafik ini, pengguna dapat melihat data yang dikumpulkan secara real-time atau data historis dari sensor yang terhubung dengan channel.



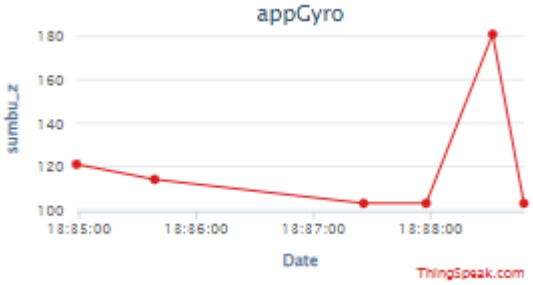
Untuk aksesibilitas, pengguna dapat mengatur konfigurasi channel agar bersifat public, memungkinkan siapa pun untuk melihat data yang ada tanpa memerlukan izin khusus atau Read API Key. Dengan pengaturan ini, channel menjadi terbuka dan siap diakses oleh publik. Namun, jika pengguna menginginkan kontrol akses yang lebih ketat, tersedia opsi private view, di mana hanya pengguna tertentu yang memiliki akses ke data dan tampilan channel. Pengaturan ini memberikan fleksibilitas bagi pengguna untuk memilih siapa yang bisa melihat dan menggunakan data yang tersedia di channel ThingSpeak.



Data detail field1

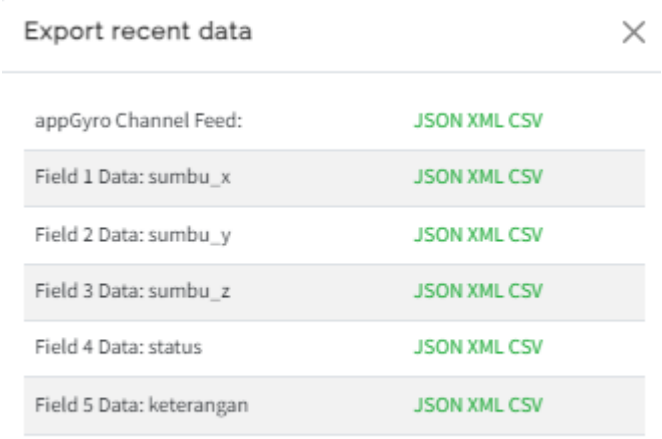


Data detail field2



Data detail field3

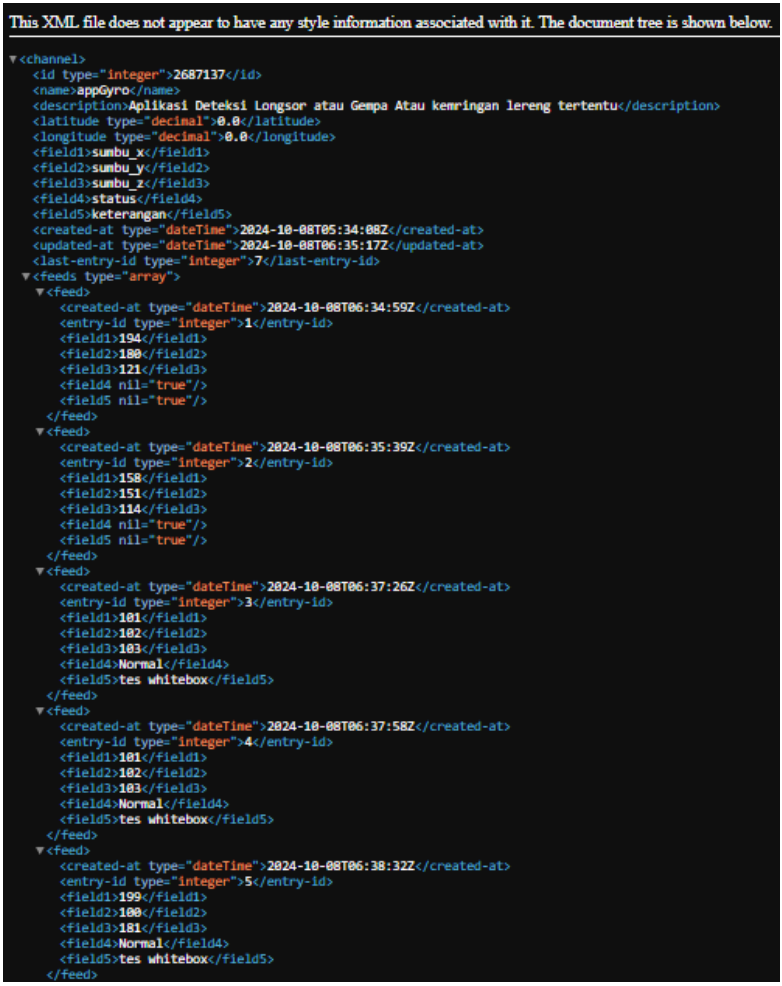
Selain itu, ThingSpeak memungkinkan pengguna untuk mengekspor data ke dalam format Excel, sehingga data yang dikumpulkan dapat diolah lebih lanjut atau dianalisis menggunakan perangkat lunak lain. Kemudahan ini mempermudah pengelolaan data dan integrasi dengan berbagai platform.



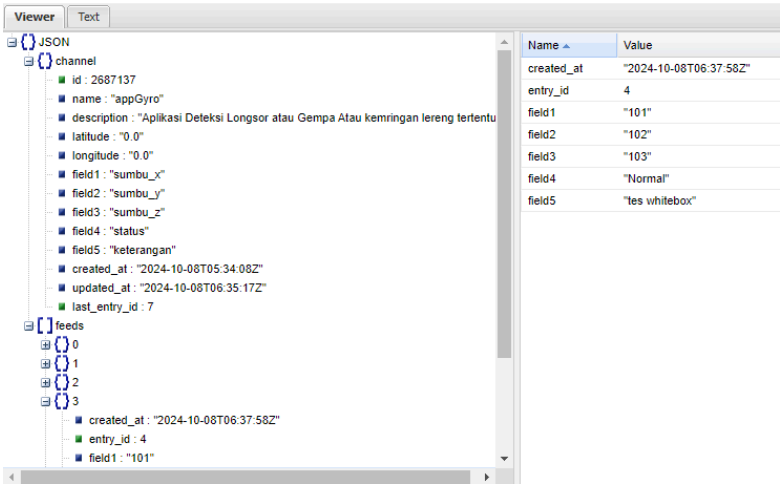
Gambar Memilih Jenis Keluaran data

A	B	C	D	E	F	G	H
created_at	entry_id	field1	field2	field3	field4	field5	
2024-10-08 06:34:59 UTC	1	194	180	121	Normal	tes whitebox	
2024-10-08 06:35:39 UTC	2	158	151	114	Normal	tes whitebox	
2024-10-08 06:37:26 UTC	3	101	102	103	Normal	tes whitebox	
2024-10-08 06:37:58 UTC	4	101	102	103	Normal	tes whitebox	
2024-10-08 06:38:32 UTC	5	199	100	181	Normal	tes whitebox	
2024-10-08 06:38:48 UTC	6	101	102	103	Normal	tes whitebox	
2024-10-08 07:03:20 UTC	7	111	112	113	Normal	tes whitebox	

Gambar Keluaran data CSV



Gambar Keluaran data XML



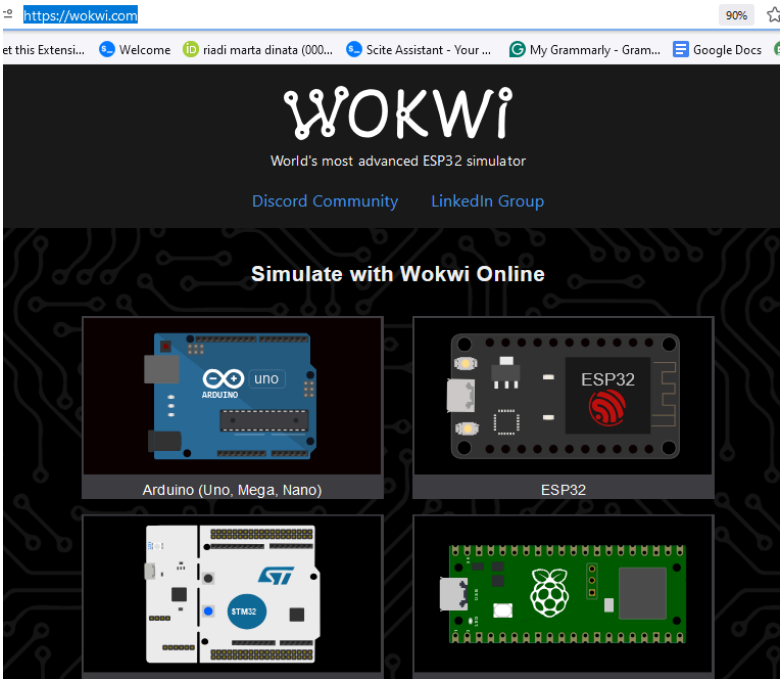
Gambar Keluaran data JSON

BAGIAN 3

RANGKAIAN SKEMATIK DAN UNO

3.1 Arduino Simulator

Sebelum menerapkan proyek pada perangkat fisik seperti Arduino UNO atau ESP32, sangat disarankan untuk melakukan uji coba menggunakan simulator.



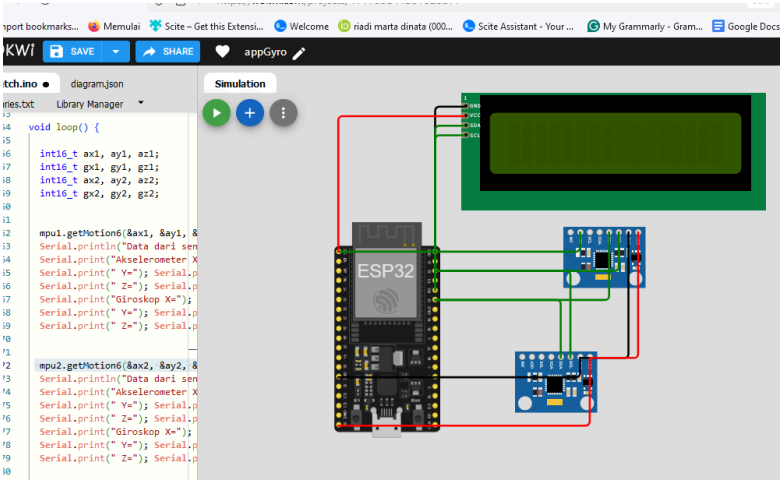
Hal ini memungkinkan pengujian kode dan memahami perilaku sistem tanpa risiko merusak komponen fisik. Dengan simulator seperti Tinkercad untuk Arduino atau Wokwi untuk ESP32, kesalahan dapat diidentifikasi lebih awal, sehingga perbaikan bisa dilakukan sebelum proyek diterapkan secara nyata.

Melalui simulasi, waktu dan biaya dapat dihemat, karena memastikan program bekerja sesuai harapan sebelum membeli atau merakit komponen. Jika semua telah berjalan dengan lancar di dalam simulasi dan semua persiapan dianggap aman, maka implementasi ke perangkat fisik dapat dilakukan dengan lebih percaya diri, mengurangi potensi kesalahan atau kerusakan yang fatal pada komponen nyata.

Langkah ini merupakan pendekatan yang efisien, karena simulasi terlebih dahulu meminimalkan risiko serta meningkatkan keberhasilan saat proyek diterapkan di dunia nyata.Untuk ESP32 silakan daftar di https://wokwi.com/

Wokwi adalah sebuah simulator online yang memungkinkan pengguna untuk memprogram dan mensimulasikan berbagai jenis mikrokontroler, termasuk ESP32, tanpa memerlukan perangkat keras fisik. Ini sangat berguna bagi pengembang atau pelajar yang ingin menguji kode dan

komponen elektronik dalam lingkungan virtual sebelum menerapkannya pada perangkat nyata. Dengan Wokwi, pengguna dapat menghubungkan berbagai sensor, perangkat I/O seperti LCD, dan komponen lainnya ke mikrokontroler, lalu memprogramnya secara langsung di browser.



ESP32

Selanjutnya, Untuk memudahkan pembuatan alat dalam komunikasi ke IoT, ESP32 adalah alternatif yang terbaik karena mikrokontroler ini menawarkan berbagai fitur unggulan yang sangat mendukung pengembangan proyek IoT dengan efisiensi dan kemudahan.

ESP32 adalah mikrokontroler yang populer karena dilengkapi dengan fitur Wi-Fi dan Bluetooth terintegrasi menjadikannya pilihan ideal untuk proyek IoT (Internet of Things). Beberapa alasan mengapa ESP32 sering digunakan untuk IoT antara lain konektivitas yang mudah, dengan Wi-Fi dan Bluetooth bawaan, ESP32 memungkinkan perangkat untuk terhubung langsung ke internet atau jaringan lokal, memungkinkan pengiriman dan penerimaan data tanpa perangkat tambahan.

Selain itu, biaya terjangkau ESP32 adalah perangkat yang kuat tetapi relatif murah, membuatnya cocok untuk proyek

IoT skala besar atau kecil. Dukungan multi-komponen, ESP32 mendukung banyak sensor dan modul termasuk gyroscope IMU, LCD, dan perangkat lainnya yang memudahkan untuk membuat sistem yang kompleks dalam satu proyek. Efisiensi energi, ESP32 memiliki mode daya rendah yang dapat digunakan dalam proyek IoT di mana daya baterai menjadi pertimbangan utama.

Mengapa Harus Simulasi Dahulu?

Simulasi di Wokwi sangat bermanfaat karena pengguna tidak memerlukan perangkat fisik. Hal ini memungkinkan pengujian kode untuk ESP32 secara langsung tanpa harus memiliki perangkat keras yang sebenarnya. Selain itu, Wokwi menyediakan integrasi komponen yang mudah, termasuk berbagai komponen simulasi seperti gyroscope IMU, LCD I2C, dan mikrokontroler lainnya, yang memungkinkan simulasi kompleks tanpa perangkat nyata. Penggunaan Wokwi juga meningkatkan efisiensi waktu dan biaya, karena pengujian dapat dilakukan lebih cepat dan murah tanpa perlu membeli dan memasang komponen fisik untuk setiap iterasi pengembangan.

Contoh Code:

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu1(0x68);
MPU6050 mpu2(0x69);

void setup() {
  Serial.begin(9600);
  Wire.begin();

  mpu1.initialize();
  if (mpu1.testConnection()) {
    Serial.println("MPU6050 pertama (0x68) terhubung.");
  } else {
    Serial.println("MPU6050 pertama (0x68) tidak
terhubung.");
  }


  mpu2.initialize();
  if (mpu2.testConnection()) {
    Serial.println("MPU6050 kedua (0x69) terhubung.");
  } else {
    Serial.println("MPU6050 kedua (0x69) tidak
terhubung.");
  }
}

void loop() {
  int16_t ax1, ay1, az1;
  int16_t gx1, gy1, gz1;
  int16_t ax2, ay2, az2;
  int16_t gx2, gy2, gz2;


  mpu1.getMotion6(&ax1, &ay1, &az1, &gx1, &gy1, &gz1);
  Serial.println("Data dari sensor 1 (0x68):");
  Serial.print("Akselerometer X="); Serial.print(ax1);
```

```
  Serial.print(" Y="); Serial.print(ay1);
  Serial.print(" Z="); Serial.println(az1);
  Serial.print("Giroskop X="); Serial.print(gx1);
  Serial.print(" Y="); Serial.print(gy1);
  Serial.print(" Z="); Serial.println(gz1);


  mpu2.getMotion6(&ax2, &ay2, &az2, &gx2, &gy2, &gz2);
  Serial.println("Data dari sensor 2 (0x69):");
  Serial.print("Akselerometer X="); Serial.print(ax2);
  Serial.print(" Y="); Serial.print(ay2);
  Serial.print(" Z="); Serial.println(az2);
  Serial.print("Giroskop X="); Serial.print(gx2);
  Serial.print(" Y="); Serial.print(gy2);
  Serial.print(" Z="); Serial.println(gz2);

  delay(1000);
}
```

Artinya dengan memanfaatkan Wokwi sebagai simulator ESP32, pengguna bisa bereksperimen dengan pengambilan data sensor, integrasi perangkat, dan pengiriman data IoT secara mudah dan efisien, tanpa harus menghadapi keterbatasan perangkat keras.

3.2 Arduino Rangkaian

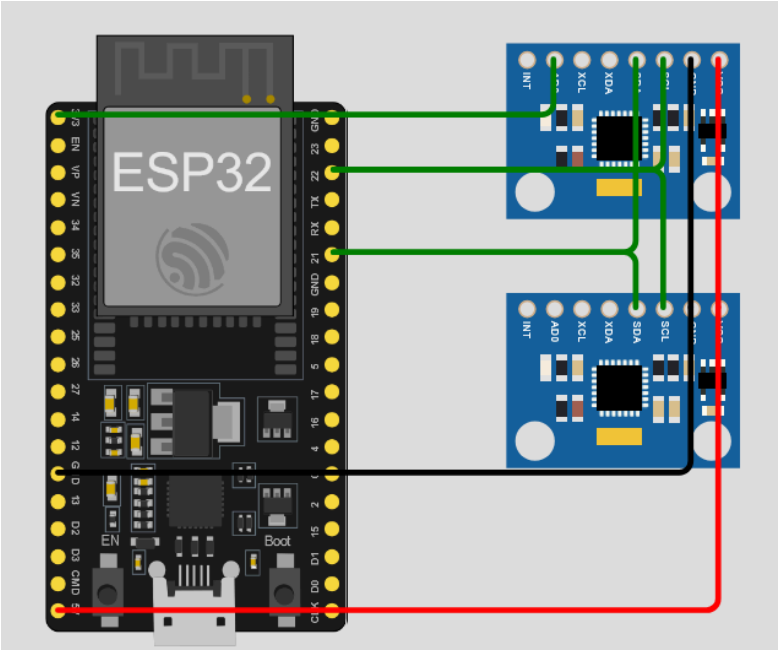
Dalam rangkaian ini, baik gyroscope dan LCD I2C memanfaatkan pin yang sama, yaitu pin SDA (21) dan SCL (22), pada mikrokontroler ESP32. Penggunaan pin yang sama dimungkinkan berkat protokol komunikasi I2C (Inter-Integrated Circuit), yang dirancang untuk

memungkinkan komunikasi antara beberapa perangkat melalui satu bus data.

I2C

adalah protokol komunikasi serial yang menggunakan dua jalur untuk komunikasi data: jalur data serial (SDA) dan jalur clock serial (SCL). I2C mendukung penggunaan banyak perangkat pada bus yang sama melalui pengalamatan perangkat yang unik. Setiap perangkat I2C memiliki alamat yang unik, memungkinkan banyak perangkat untuk terhubung ke bus yang sama tanpa konflik.

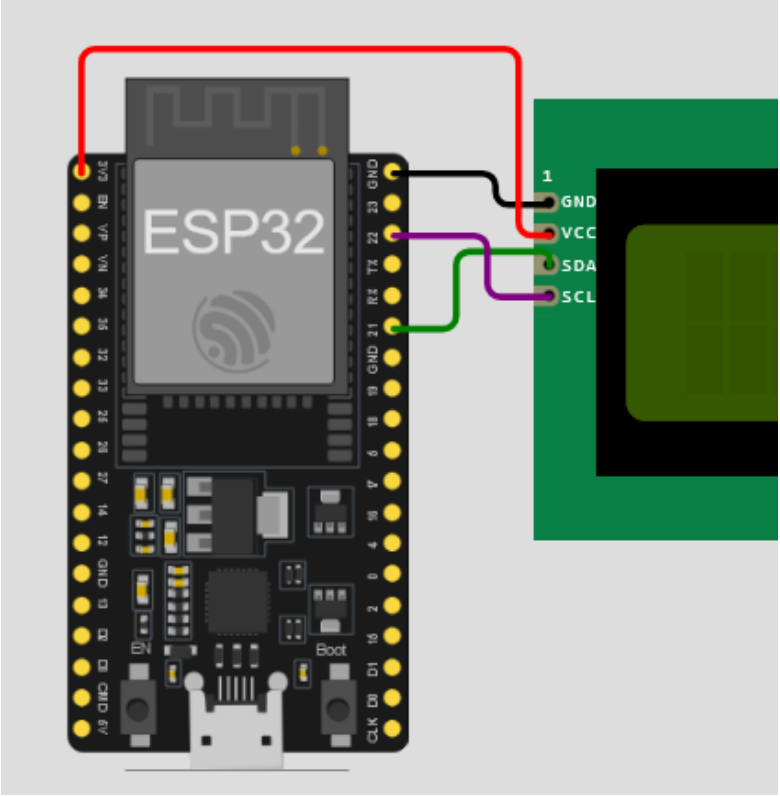
Pada rangkaian ini, pin SDA (21) dan SCL (22) digunakan untuk menghubungkan ESP32 dengan gyroscope dan LCD I2C. Berikut adalah alasan dan keunggulan penggunaan I2C dalam konteks ini:



Kemudahan Integrasi, I2C memungkinkan integrasi berbagai perangkat hanya dengan dua jalur komunikasi (SDA dan SCL), menghemat pin GPIO pada ESP32 untuk keperluan lain. Protokol ini juga memungkinkan penambahan lebih banyak perangkat ke bus yang sama tanpa memerlukan jalur tambahan.

Pengalamatan Unik, Setiap perangkat I2C memiliki alamat yang unik, memungkinkan ESP32 untuk berkomunikasi dengan setiap perangkat secara individual tanpa konflik.

Sederhana dan Hemat Pin, Dengan hanya dua jalur komunikasi, I2C memudahkan koneksi beberapa perangkat, menghemat pin GPIO yang berharga pada mikrokontroler. Penghematan pin ini sangat penting pada mikrokontroler dengan jumlah pin terbatas, memungkinkan penambahan lebih banyak perangkat atau sensor pada sistem yang sama.



Kompatibilitas dengan Banyak Perangkat, I2C adalah standar yang diadopsi luas, didukung oleh berbagai sensor dan modul, termasuk gyroscope dan LCD I2C. Standar ini memudahkan penambahan berbagai jenis perangkat tanpa perlu perubahan besar pada rangkaian atau kode.

Efisiensi dalam Pengembangan dan Pemeliharaan, Protokol I2C memungkinkan pengembangan yang lebih cepat dan pemeliharaan yang lebih mudah karena hanya perlu menangani dua jalur komunikasi. I2C juga mendukung kecepatan komunikasi yang cukup untuk banyak aplikasi, termasuk pembacaan sensor dan tampilan data.

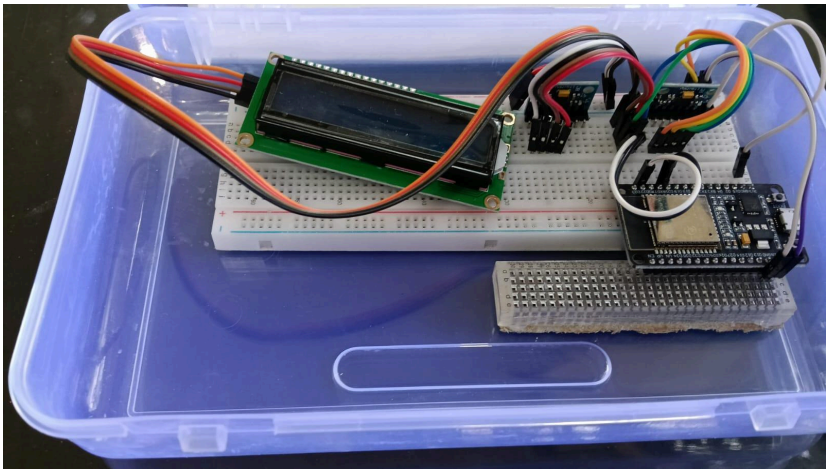
Stabilitas dan Keandalan, I2C dirancang untuk komunikasi jarak pendek dan memiliki mekanisme untuk mendeteksi dan mengatasi kesalahan komunikasi, meningkatkan keandalan sistem. I2C mendukung komunikasi multi-master dan multi-slave, memungkinkan fleksibilitas dalam desain sistem.

3.3 Implementasi pada Rangkaian

Pada rangkaian ini, gyroscope dan LCD I2C dihubungkan ke pin SDA (21) dan SCL (22) pada ESP32. Berikut adalah beberapa langkah implementasi:

1. Inisialisasi Bus I2C, Menginisialisasi bus I2C pada pin 21 (SDA) dan 22 (SCL) menggunakan Wire.begin(21, 22).
2. Inisialisasi Perangkat, Menginisialisasi LCD dengan alamat I2C yang sesuai, misalnya 0x27, dan gyroscope, misalnya 0x68.
3. Komunikasi dengan Perangkat, Menggunakan alamat I2C unik untuk berkomunikasi dengan masing-masing perangkat, membaca data dari gyroscope dan menampilkan data pada LCD.

Dengan memanfaatkan protokol I2C, sistem ini dapat dengan mudah menambahkan lebih banyak perangkat di masa depan, hanya dengan memastikan setiap perangkat memiliki alamat I2C yang unik. Ini memungkinkan fleksibilitas dan skalabilitas dalam pengembangan proyek IoT yang lebih besar dan kompleks.



3.4 Mencari Nilai Rata-Rata Gyroscope

Untuk mendapatkan nilai rata-rata dari data gyroscope pada sumbu x, y, dan z dari dua sensor (dalam hal ini mpu1 dan mpu2), Lalu dapat dijumlahkan nilai dari masing-masing sumbu (x, y, z) yang diperoleh dari kedua sensor dan kemudian membaginya dengan 2 (karena ada dua sensor).

Misalkan variabel gx1, gy1, dan gz1 mewakili nilai gyroscope dari sensor pertama (mpu1), dan gx2, gy2, dan gz2 mewakili nilai gyroscope dari sensor kedua (mpu2). Untuk menghitung rata-rata dari setiap sumbu gyroscope, langkahnya adalah sebagai berikut:

Rumus untuk menghitung rata-rata gyroscope:

1. Rata-rata gyroscope sumbu x:

$$\text{gx\_avg} = \frac{gx1 + gx2}{2}$$

2. Rata-rata gyroscope sumbu y:

$$\text{gy\_avg} = \frac{gy1 + gy2}{2}$$

3. Rata-rata gyroscope sumbu z:

$$\text{gz\_avg} = \frac{gz1 + gz2}{2}$$

```
// Mengambil data gyroscope dari dua sensor MPU
mpu1.getMotion6(&ax1, &ay1, &az1, &gx1, &gy1, &gz1);
mpu2.getMotion6(&ax2, &ay2, &az2, &gx2, &gy2, &gz2);

// Menghitung rata-rata gyroscope pada sumbu x, y, dan
z
float gx_avg = (gx1 + gx2) / 2.0;
float gy_avg = (gy1 + gy2) / 2.0;
float gz_avg = (gz1 + gz2) / 2.0;

// Menampilkan hasil rata-rata
Serial.print("Rata-rata gyroscope X: ");
Serial.println(gx_avg);
```

```
Serial.print("Rata-rata gyroscope Y: ");
Serial.println(gy_avg);
Serial.print("Rata-rata gyroscope Z: ");
Serial.println(gz_avg);
```

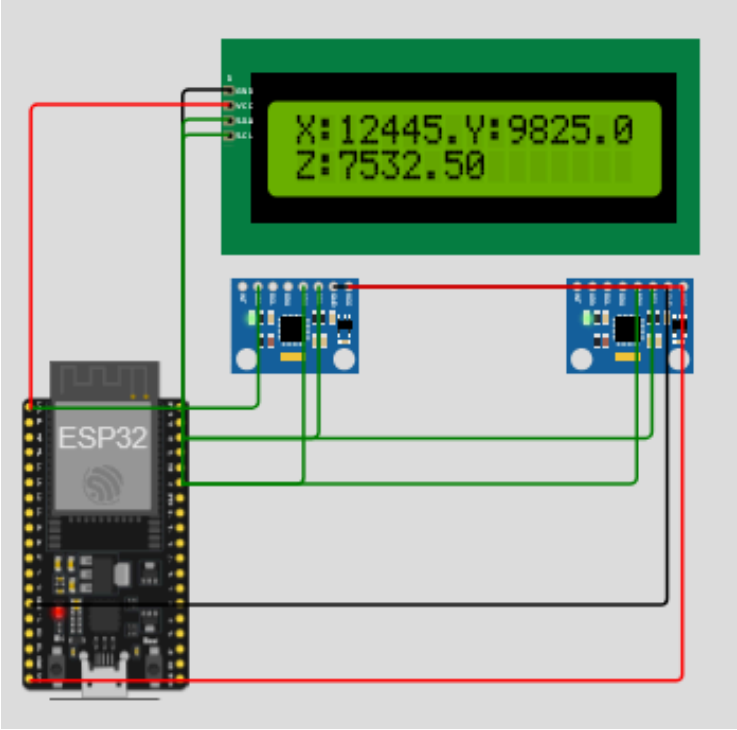
Penjelasan Fungsi Kode terpilih:

- mpu1.getMotion6 dan mpu2.getMotion6, Fungsi ini digunakan untuk mendapatkan nilai accelerometer dan gyroscope dari sensor MPU6050 pada sumbu x, y, z.
- gx1, gy1, gz1, Nilai gyroscope dari sensor pertama (mpu1) untuk sumbu x, y, z.
- gx2, gy2, gz2, Nilai gyroscope dari sensor kedua (mpu2) untuk sumbu x, y, z.
- Nilai rata-rata untuk setiap sumbu dihitung dengan menjumlahkan nilai gyroscope dari kedua sensor pada sumbu yang sama, kemudian membaginya dengan 2.

Hasil pengujian di Wokwi menunjukkan bahwa data dari dua sensor gyroscope berhasil dikirimkan dan ditampilkan secara real-time pada LCD serta melalui serial monitor. Pada layar LCD, hasil rata-rata nilai gyroscope untuk sumbu X, Y, dan Z ditampilkan dengan jelas. Nilai yang ditampilkan

adalah hasil pengolahan data dari kedua sensor yang terhubung ke ESP32.

Secara spesifik, pada tampilan LCD terlihat nilai rata-rata gyroscope sebagai berikut: X: 12445.0, Y: 9825.0, dan Z: 7532.50. Hal ini menunjukkan bahwa sistem telah berhasil menghitung dan menampilkan rata-rata data dari kedua sensor gyroscope.



Selain itu, data yang sama juga muncul pada serial monitor dengan format yang sesuai, memudahkan pemantauan dan verifikasi data secara langsung. Pesan-pesan di serial monitor menunjukkan data mentah dari masing-masing sensor dan hasil rata-rata yang dihitung:

- Giroskop X: 24890 Y: 19650 Z: 15065
- Rata-rata gyroscope X: 12445.00
- Rata-rata gyroscope Y: 9825.00
- Rata-rata gyroscope Z: 7532.50

Dan sebagai berikut adalah data keluarannya pada mode serial:

```
Rata-rata gyroscope X: 12445.00
Rata-rata gyroscope Y: 9825.00
Rata-rata gyroscope Z: 7532.50
++++++++++++++++++++++++++++++++++++++
Data dari sensor 1 (0x68):
Akselerometer X=0 Y=0 Z=16384
Giroskop X=0 Y=0 Z=0
Data dari sensor 2 (0x69):
Akselerometer X=21299 Y=18022 Z=16384
Giroskop X=24890 Y=19650 Z=15065
Rata-rata gyroscope X: 12445.00
Rata-rata gyroscope Y: 9825.00
Rata-rata gyroscope Z: 7532.50
++++++++++++++++++++++++++++++++++++++
Data dari sensor 1 (0x68):
Akselerometer X=0 Y=0 Z=16384
Giroskop X=0 Y=0 Z=0
Data dari sensor 2 (0x69):
Akselerometer X=21299 Y=18022 Z=16384
Giroskop X=24890 Y=19650 Z=15065
```

Keberhasilan ini menegaskan bahwa sistem integrasi antara ESP32 dengan LCD I2C dan sensor gyroscope melalui I2C berjalan dengan baik. Baik data yang muncul di LCD maupun di serial monitor menunjukkan hasil yang akurat dan sesuai dengan yang diharapkan, membuktikan bahwa penggunaan I2C untuk komunikasi antara komponen dalam rangkaian ini adalah solusi yang efisien dan efektif.

Atau secara rinci hasilnya adalah sebagai berikut:

```
WiFi connected
IP address: 10.10.0.2
Data dari sensor 1 (0x68):
Akselerometer X=0 Y=0 Z=16384
Giroskop X=0 Y=0 Z=0
Data dari sensor 2 (0x69):
Akselerometer X=21299 Y=18022 Z=16384
Giroskop X=24890 Y=19650 Z=15065
Rata-rata gyroscope X: 12445.00
Rata-rata gyroscope Y: 9825.00
Rata-rata gyroscope Z: 7532.50

++++++++++++++++ +++++++++++++++

Data dari sensor 1 (0x68):
Akselerometer X=0 Y=0 Z=16384
Giroskop X=0 Y=0 Z=0
Data dari sensor 2 (0x69):
Akselerometer X=21299 Y=18022 Z=16384
Giroskop X=24890 Y=19650 Z=15065
Rata-rata gyroscope X: 12445.00
Rata-rata gyroscope Y: 9825.00
Rata-rata gyroscope Z: 7532.50
```

```
++++++++++++++++ ++++++++++++++++++

Data dari sensor 1 (0x68):
Akselerometer X=0 Y=0 Z=16384
Giroskop X=0 Y=0 Z=0
Data dari sensor 2 (0x69):
Akselerometer X=21299 Y=18022 Z=16384
Giroskop X=24890 Y=19650 Z=15065
Rata-rata gyroscope X: 12445.00
Rata-rata gyroscope Y: 9825.00
Rata-rata gyroscope Z: 7532.50

++++++++++++++ +++++++++++++++++

Data dari sensor 1 (0x68):
Akselerometer X=0 Y=0 Z=16384
Giroskop X=0 Y=0 Z=0
Data dari sensor 2 (0x69):
Akselerometer X=21299 Y=18022 Z=16384
Giroskop X=24890 Y=19650 Z=15065
Rata-rata gyroscope X: 12445.00
Rata-rata gyroscope Y: 9825.00
Rata-rata gyroscope Z: 7532.50
++++++++++++++ +++++++++++++++++
…
```

Dengan kode ini, nilai rata-rata gyroscope dari kedua sensor akan diperoleh dan dapat digunakan untuk analisis lebih lanjut atau dikirim ke platform seperti ThingSpeak.

Untuk mendeteksi gempa dengan akurasi tinggi menggunakan **a**ccelerometer dan gyroscope, kita dapat menggabungkan data dari keduanya untuk mendeteksi baik

percepatan linier maupun rotasi mendadak. Berikut adalah langkah dan kode untuk mengimplementasikan perhitungan deteksi gempa dengan Keluaran gyroscope adalah nilai

Accelerometer:

- Cocok untuk mendeteksi getaran linier (amplitudo gerakan).
- Lebih banyak digunakan dalam mendeteksi gempa karena lebih langsung mengukur perubahan posisi.

Gyroscope:

- Cocok untuk mendeteksi perubahan rotasi yang tajam.
- Bisa digunakan sebagai pelengkap accelerometer untuk mendeteksi pola rotasi mendadak selama gempa.

1. Percepatan Linier untuk Mendeteksi Gempa

Apa itu percepatan linier?

- Percepatan linier adalah perubahan kecepatan suatu benda yang bergerak dalam garis lurus.

- Dalam konteks gempa, ini berarti gerakan naik-turun, maju-mundur, atau ke samping akibat getaran tanah.

Contoh sehari-hari:

- Bayangkan Anda sedang berdiri di dalam lift. Saat lift bergerak ke atas atau ke bawah, tubuh Anda merasakan tekanan atau ringan. Itu adalah percepatan linier.
- Dalam gempa, fenomena ini terjadi secara horizontal (samping) atau vertikal (atas-bawah), yang membuat lantai terasa bergoyang.

Mengapa percepatan linier relevan untuk mendeteksi gempa?

- Saat tanah bergoyang akibat gempa, gerakan tersebut menghasilkan percepatan linier yang terukur oleh accelerometer. Semakin besar guncangan, semakin besar percepatan linier yang tercatat.

Cara mendeteksi:

- Sensor accelerometer pada perangkat Anda membaca perubahan percepatan di sumbu X, Y, dan Z (depan-belakang, kiri-kanan, atas-bawah).
- Jika magnitudo percepatan ini melebihi ambang batas tertentu, maka kita bisa mengklasifikasikan tingkat gempa, seperti:
    - Normal: Getaran sangat kecil, hampir tidak terasa.
    - Warning: Getaran terasa, tapi tidak terlalu kuat.
    - Danger: Getaran kuat yang bisa menyebabkan kerusakan.

2. Pola Rotasi atau Kombinasi Gerakan yang Kompleks

Apa itu rotasi?

- Rotasi adalah perputaran suatu benda di sekitar sumbunya. Gyroscope mendeteksi gerakan ini dengan mengukur kecepatan sudut (seberapa cepat benda berputar).

Contoh sehari-hari:

- Saat Anda memegang gelas air dan memutarnya perlahan, gerakan memutar gelas itu adalah rotasi. Jika Anda menggoyangkan gelas sambil memutar,

itu adalah kombinasi gerakan (rotasi dan percepatan linier).

- Dalam konteks gempa, ini terjadi saat benda-benda di dalam ruangan, seperti lampu gantung, tidak hanya bergerak maju-mundur, tetapi juga bergoyang dalam pola melingkar.

Mengapa rotasi relevan untuk mendeteksi gempa?

- Selama gempa kuat, tanah tidak hanya bergoyang ke satu arah, tetapi juga bisa bergoyang dengan pola yang kompleks, menciptakan rotasi pada benda-benda di atasnya.
- Misalnya, meja mungkin bergerak maju-mundur sambil sedikit berputar pada sumbunya. Gyroscope mendeteksi pola ini.

Cara mendeteksi:

- Gyroscope mengukur perubahan rotasi pada sumbu X, Y, dan Z. Jika pola rotasi tiba-tiba meningkat, ini bisa menjadi tanda getaran yang kompleks atau gempa kuat.

Perbandingan Percepatan Linier dan Rotasi

Karakteristik	Percepatan Linier (Accelerometer)	Rotasi (Gyroscope)
Jenis Gerakan	Maju-mundur, kiri-kanan, atas-bawah (getaran lurus)	Memutar atau bergoyang melingkar
Contoh	Bangunan bergoyang ke samping	Lampu gantung berayun dalam pola melingkar
Sensor yang Digunakan	Accelerometer	Gyroscope
Relevansi pada Gempa	Utama (deteksi getaran tanah)	Pelengkap (deteksi rotasi mendadak)

3. Kombinasi Gerakan: Percepatan + Rotasi

Apa itu kombinasi gerakan?

- Kombinasi gerakan adalah ketika percepatan linier dan rotasi terjadi bersamaan.
- Misalnya, selama gempa, tanah bergoyang ke samping (percepatan linier), dan lampu gantung mulai berputar karena pola gelombang gempa yang tidak beraturan (rotasi).

Contoh sehari-hari:

- Bayangkan Anda mengaduk kopi dengan sendok. Gerakan tangan Anda ke depan-belakang adalah percepatan linier, sementara gerakan melingkar sendok adalah rotasi.
- Selama gempa besar, lantai mungkin tidak hanya bergerak maju-mundur, tetapi juga menciptakan

pola gelombang yang menyebabkan benda di atasnya berputar sedikit.

Mengapa kombinasi ini penting?

- Dalam gempa yang sangat kuat, kedua pola ini sering terjadi bersamaan. Menggabungkan data accelerometer dan gyroscope dapat memberikan gambaran yang lebih lengkap tentang tingkat kekuatan gempa.

Kesimpulan

- Percepatan linier (accelerometer): Cocok untuk mendeteksi getaran tanah sederhana seperti maju-mundur atau atas-bawah.
- Rotasi (gyroscope): Berguna untuk mendeteksi pola bergoyang melingkar atau rotasi mendadak.
- Kombinasi keduanya: Berguna untuk analisis yang lebih akurat dalam mendeteksi gempa yang kompleks.

Jika Anda hanya ingin mendeteksi getaran gempa sederhana, fokuslah pada accelerometer. Namun, jika ingin mendeteksi pola yang lebih rumit, tambahkan gyroscope sebagai pelengkap

3.5 Kalibrasi Getaran

Untuk mendeteksi gempa dengan akurasi tinggi menggunakan accelerometer dan gyroscope, kita dapat menggabungkan data dari keduanya untuk mendeteksi baik percepatan linier maupun rotasi mendadak. Berikut adalah langkah dan kode untuk mengimplementasikan perhitungan deteksi gempa:

Langkah Implementasi

1. **Baca Data dari Sensor:**
    - Gunakan **accelerometer** untuk mendeteksi percepatan linier pada sumbu X, Y, dan Z.
    - Gunakan **gyroscope** untuk mendeteksi kecepatan sudut pada sumbu X, Y, dan Z.
2. **Hitung Magnitudo Total:**
    - Magnitudo total percepatan linier dari accelerometer:

$$\text{accel\_magnitude} = \sqrt{ax^2 + ay^2 + az^2}$$

- Magnitudo total kecepatan sudut dari gyroscope:

$$\text{gyro\_magnitude} = \sqrt{gx^2 + gy^2 + gz^2}$$

3. **Tentukan Ambang Batas:**
   - **Accelerometer:**
     - Normal: <2.0 m/s2
     - Warning: 2.0 m/s2 – 5.0 m/s2
     - Danger: >5.0 m/s2
   - **Gyroscope:**
     - Normal: <3.0 rad/s
     - Warning: 3.0 rad/s – 6.0 rad/s
     - Danger: >6.0 rad/s
4. **Kombinasikan Data:**
   - Jika salah satu magnitudo (accelerometer atau gyroscope) berada di level "Warning" atau "Danger," tingkatkan sensitivitas deteksi.

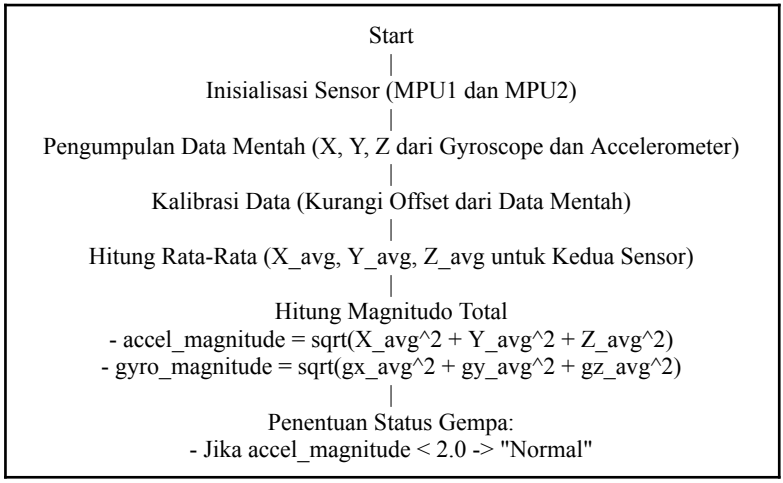
```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu1(0x68); // Sensor pertama
MPU6050 mpu2(0x69); // Sensor kedua

float ax1_offset = 0, ay1_offset = 0, az1_offset = 0;
float gx1_offset = 0, gy1_offset = 0, gz1_offset = 0;
float ax2_offset = 0, ay2_offset = 0, az2_offset = 0;
float gx2_offset = 0, gy2_offset = 0, gz2_offset = 0;
```

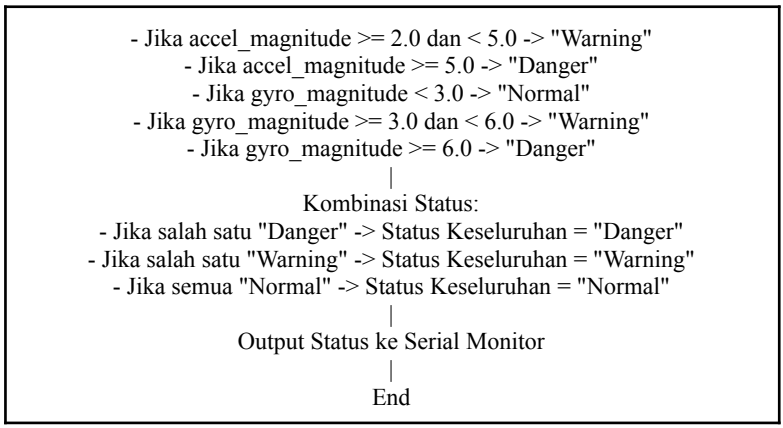
```
void setup() {
  Serial.begin(115200);
  Wire.begin();
  mpu1.initialize();
  mpu2.initialize();

  // Kalibrasi sensor
      calibrateSensor(mpu1,  ax1_offset,  ay1_offset,  az1_offset,
gx1_offset, gy1_offset, gz1_offset);
      calibrateSensor(mpu2,  ax2_offset,  ay2_offset,  az2_offset,
gx2_offset, gy2_offset, gz2_offset);
}

void loop() {
  // Data mentah dari sensor
  int16_t ax1, ay1, az1, gx1, gy1, gz1;
  int16_t ax2, ay2, az2, gx2, gy2, gz2;

  mpu1.getMotion6(&ax1, &ay1, &az1, &gx1, &gy1, &gz1);
  mpu2.getMotion6(&ax2, &ay2, &az2, &gx2, &gy2, &gz2);

  // Koreksi offset
  float ax1_cal = ax1 - ax1_offset;
  float ay1_cal = ay1 - ay1_offset;
  float az1_cal = az1 - az1_offset;
  float gx1_cal = gx1 - gx1_offset;
  float gy1_cal = gy1 - gy1_offset;
  float gz1_cal = gz1 - gz1_offset;

  float ax2_cal = ax2 - ax2_offset;
  float ay2_cal = ay2 - ay2_offset;
  float az2_cal = az2 - az2_offset;
  float gx2_cal = gx2 - gx2_offset;
  float gy2_cal = gy2 - gy2_offset;
  float gz2_cal = gz2 - gz2_offset;

  // Rata-rata accelerometer dan gyroscope
  float ax_avg = (ax1_cal + ax2_cal) / 2.0;
  float ay_avg = (ay1_cal + ay2_cal) / 2.0;
  float az_avg = (az1_cal + az2_cal) / 2.0;

  float gx_avg = (gx1_cal + gx2_cal) / 2.0;
  float gy_avg = (gy1_cal + gy2_cal) / 2.0;
  float gz_avg = (gz1_cal + gz2_cal) / 2.0;
```

```
  // Hitung magnitudo total
  float accel_magnitude = sqrt(ax_avg * ax_avg + ay_avg * ay_avg +
az_avg * az_avg);
  float gyro_magnitude = sqrt(gx_avg * gx_avg + gy_avg * gy_avg +
gz_avg * gz_avg);

  // Tentukan status gempa
  detectEarthquake(accel_magnitude, gyro_magnitude);
}

void calibrateSensor(MPU6050 &mpu, float &ax_offset, float
&ay_offset, float &az_offset,
                float &gx_offset, float &gy_offset, float &gz_offset) {
  long ax_sum = 0, ay_sum = 0, az_sum = 0;
  long gx_sum = 0, gy_sum = 0, gz_sum = 0;

  for (int i = 0; i < 100; i++) {
    int16_t ax, ay, az, gx, gy, gz;
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    ax_sum += ax;
    ay_sum += ay;
    az_sum += az;
    gx_sum += gx;
    gy_sum += gy;
    gz_sum += gz;
    delay(10);
  }

  ax_offset = ax_sum / 100.0;
  ay_offset = ay_sum / 100.0;
  az_offset = az_sum / 100.0;
  gx_offset = gx_sum / 100.0;
  gy_offset = gy_sum / 100.0;
  gz_offset = gz_sum / 100.0;
}

void detectEarthquake(float accel_magnitude, float
gyro_magnitude) {
  String accel_status;
  String gyro_status;

  // Penentuan status berdasarkan magnitudo
  if (accel_magnitude < 2.0) accel_status = "Normal";
```

```
  else if (accel_magnitude < 5.0) accel_status = "Warning";
  else accel_status = "Danger";

  if (gyro_magnitude < 3.0) gyro_status = "Normal";
  else if (gyro_magnitude < 6.0) gyro_status = "Warning";
  else gyro_status = "Danger";

  // Kombinasi status
  String overall_status = "Normal";
     if (accel_status == "Danger" || gyro_status == "Danger")
overall_status = "Danger";
   else if (accel_status == "Warning" || gyro_status == "Warning")
overall_status = "Warning";

  // Output status
  Serial.print("Accel Magnitude: "); Serial.print(accel_magnitude);
  Serial.print(" | Gyro Magnitude: "); Serial.print(gyro_magnitude);
  Serial.print(" | Overall Status: "); Serial.println(overall_status);
}
```

Berikut adalah flowchart alur kerja sistem deteksi gempa menggunakan minimal dua sensor (gyroscope dan accelerometer):





Penjelasan Flowchart:

1. Mulai (Start)

   Proses dimulai dengan inisialisasi sistem dan sensor.

2. Inisialisasi Sensor

   Kedua sensor gyroscope dan accelerometer diinisialisasi untuk pengumpulan data (MPU1 dan MPU2).

3. Pengumpulan Data Mentah

   Sensor membaca nilai X, Y, Z untuk gyroscope dan accelerometer dari kedua sensor.

4. Kalibrasi Data

   Offset dikurangkan dari data mentah untuk setiap

sumbu (X, Y, Z) dari kedua sensor untuk mendapatkan data yang dikalibrasi.

5. Penghitungan Rata-Rata

   Data dari kedua sensor dihitung rata-rata untuk setiap sumbu:

   - X_avg = (X1 + X2) / 2
   - Y_avg = (Y1 + Y2) / 2
   - Z_avg = (Z1 + Z2) / 2

6. Kalkulasi Magnitudo Total

   Hitung magnitudo total untuk accelerometer dan gyroscope menggunakan formula:

   - accel_magnitude = sqrt(X_avg^2 + Y_avg^2 + Z_avg^2)
   - gyro_magnitude = sqrt(gx_avg^2 + gy_avg^2 + gz_avg^2)

7. Penentuan Status

   Tentukan status berdasarkan magnitudo:

   - Normal: accel_magnitude < 2.0 dan gyro_magnitude < 3.0
   - Warning: 2.0 <= accel_magnitude < 5.0 atau 3.0 <= gyro_magnitude < 6.0
   - Danger: accel_magnitude >= 5.0 atau gyro_magnitude >= 6.0

8. Output Status

   Status keseluruhan dicetak ke serial monitor:

   - Jika ada salah satu status "Danger", maka status keseluruhan adalah "Danger".
   - Jika ada salah satu status "Warning", maka status keseluruhan adalah "Warning".
   - Jika semua status "Normal", maka status keseluruhan adalah "Normal".

9. Ulangi Proses

   Kembali ke langkah pengumpulan data mentah untuk loop berikutnya.

10. Selesai (End)

    Proses dihentikan jika diperlukan.

Catatan Sumber referensi:

1. Literatur Akademik dan Artikel Ilmiah

1. "An Earthquake Detection System Based on MEMS Accelerometer Sensor and IoT"
   - Penulis: S. Sharma, A. Kumar
   - Sumber: IEEE Xplore Digital Library
   - Artikel ini membahas penggunaan sensor MEMS (Micro-Electro-Mechanical Systems) accelerometer untuk mendeteksi gempa bumi dan pengiriman data menggunakan IoT.

- Link: IEEE Xplore : https://ieeexplore.ieee.org

2. "Earthquake Detection using MEMS Sensors: A Comprehensive Study"
    - Penulis: G. Yu, L. Wang
    - Sumber: Springer Link
    - Penelitian ini meninjau metode deteksi gempa menggunakan kombinasi accelerometer dan gyroscope serta teknik pemrosesan data.
    - Link: Springer Link : https://link.springer.com
3. "Earthquake Early Warning System Using Low-Cost Sensors"
    - Penulis: K. Nakamura, S. Kanai
    - Sumber: Earthquake Engineering Journal
    - Membahas pengembangan sistem peringatan dini gempa dengan menggunakan sensor accelerometer murah untuk keperluan publik.
    - Link: Earthquake Engineering Journal : https://earthquakeengineering.org

2. Datasheet Sensor

1. MPU6050 Datasheet
    - Datasheet resmi untuk sensor MPU6050 yang digunakan dalam kode. Ini mencakup informasi tentang sensitivitas accelerometer dan gyroscope.
    - Link: MPU6050 Datasheet
2. Bosch BMA280 (Alternatif Accelerometer)

- Datasheet sensor accelerometer BMA280 yang sering digunakan untuk deteksi getaran.
- Link: BMA280 Datasheet : https://www.bosch-sensortec.com

3. Dokumentasi dan Panduan Pengembangan

1. Arduino MPU6050 Library Documentation
   - Dokumentasi resmi library untuk menggunakan sensor MPU6050 dengan Arduino.
   - Link: MPU6050 Arduino Library : https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050
2. Google Scholar
   - Banyak penelitian terkait deteksi gempa dapat ditemukan dengan kata kunci seperti "earthquake detection using accelerometer and gyroscope."
   - Link: Google Scholar : https://scholar.google.com

4. Tutorial dan Blog Teknologi

1. "Detecting Earthquake Vibration Using MPU6050 with Arduino"
   - Tutorial praktis yang menjelaskan cara membaca data accelerometer dan gyroscope untuk deteksi gempa.

- Link: Electronics Hub : https://www.electronicshub.org

2. "Earthquake Simulation with Arduino and MPU6050"
    - Blog yang membahas simulasi gempa menggunakan Arduino dan MPU6050 untuk mendeteksi getaran.
    - Link: Hackster.io : https://www.hackster.io

---

5. Organisasi Terkait

1. US Geological Survey (USGS)
    - USGS memiliki penelitian mendalam tentang deteksi gempa dan metode pengukuran getaran.
    - Link: USGS Earthquake Monitoring
2. Earthquake Engineering Research Institute (EERI)
    - Organisasi yang berfokus pada penelitian dan pengembangan teknologi deteksi gempa.
    - Link: EERI : https://www.eeri.org

KALIBRASI

Angka-angka minimum dan maksimum yang digunakan dalam deteksi gempa adalah bergantung pada kondisi fisik lingkungan, jenis perangkat keras, dan kebutuhan spesifik aplikasi Anda. Jika Anda menggunakan library #include

<MPU6050.h>, ada beberapa poin penting yang harus dipertimbangkan terkait sensitivitas sensor dan kebutuhan kalibrasi.

1. Apakah Angka Min-Max Sudah Sesuai?

Tidak sepenuhnya. Angka yang diberikan sebagai ambang batas (2.0, 5.0, dll.) adalah nilai umum yang dapat digunakan sebagai referensi awal. Namun, sensor MPU6050 memiliki karakteristik unik, dan nilai ini dapat bervariasi berdasarkan:

- Lingkungan Aplikasi: Getaran gempa di laboratorium atau simulasi mungkin berbeda dari getaran nyata.
- Sensitivitas Sensor: MPU6050 memiliki pengaturan skala penuh untuk accelerometer dan gyroscope, yang memengaruhi nilai yang terbaca.
    - Accelerometer: ±2g, ±4g, ±8g, atau ±16g
    - Gyroscope: ±250°/s, ±500°/s, ±1000°/s, atau ±2000°/s
- Tingkat Noise: MPU6050 memiliki noise kecil yang dapat memengaruhi pembacaan, terutama dalam lingkungan statis.

Jika nilai ambang batas terlalu tinggi atau terlalu rendah, hasil deteksi bisa tidak akurat. Oleh karena itu, kalibrasi diperlukan.

2. Apakah MPU6050 Perlu Dikoreksi (Kalibrasi)?

Ya, kalibrasi sangat disarankan karena sensor MPU6050 mungkin memiliki bias bawaan (*offset*) dan variasi kecil dalam sensitivitas antara unit. Kalibrasi akan memastikan bahwa pembacaan akurat dan konsisten.

2.1 Bias (Offset)

- Bias adalah nilai tetap yang terbaca saat sensor dalam posisi statis.
- Contoh:
    - Saat accelerometer dalam posisi diam pada sumbu X, nilai seharusnya 0 g, tetapi sensor mungkin membaca +0.05 g atau -0.03 g.

2.2 Cara Kalibrasi MPU6050

1. Kalibrasi Manual:
    - Tempatkan sensor dalam posisi statis di permukaan datar.
    - Catat nilai rata-rata untuk setiap sumbu (X, Y, Z).

- Gunakan nilai tersebut untuk mengoreksi pembacaan sensor.
- Contoh

```
float ax_offset = 0.05;
// Offset dari kalibrasi manual
float ay_offset = -0.02;
float az_offset = 0.01;

float ax_calibrated = ax - ax_offset;
float ay_calibrated = ay - ay_offset;
float az_calibrated = az - az_offset;
```

Kalibrasi Otomatis:

- Penggunaan library seperti **MPU6050** atau **I2Cdevlib** yang memiliki fungsi bawaan untuk kalibrasi.
- Contoh dengan library **MPU6050.h**:

```
#include <MPU6050.h>
MPU6050 mpu;

void setup() {
  Serial.begin(115200);
  Wire.begin();
  mpu.initialize();

  Serial.println("Kalibrasi sensor...");
  mpu.CalibrateAccel(6);
```

```
// Kalibrasi accelerometer
   mpu.CalibrateGyro(6);
// Kalibrasi gyroscope
   Serial.println("Kalibrasi selesai.");
}
```

3. Langkah untuk Menyesuaikan Angka Min-Max

1. Setel Skala Penuh: Pastikan menggunakan skala penuh (full scale range) yang sesuai untuk aplikasi:
    - Untuk accelerometer, gunakan ±4g atau ±8g untuk deteksi gempa.
    - Untuk gyroscope, gunakan ±500°/s untuk deteksi rotasi mendadak.
2. Contoh pengaturan:

```
mpu.setFullScaleAccelRange(MPU6050_ACCEL_FS_4);
mpu.setFullScaleGyroRange(MPU6050_GYRO_FS_500);
```

Pengujian Nyata:

- Rekam data accelerometer dan gyroscope dalam kondisi normal (tanpa gempa), simulasi gempa ringan, dan simulasi gempa kuat.
- Analisis data untuk menentukan angka ambang batas yang tepat.

Gunakan Nilai Rata-Rata:

- Jika ada noise pada data, hitung rata-rata beberapa pembacaan untuk mendapatkan nilai yang lebih stabil:

```
float ax_avg = (ax1 + ax2 + ax3) / 3.0;
// Contoh rata-rata
```

4. Referensi Nilai Ambang Batas

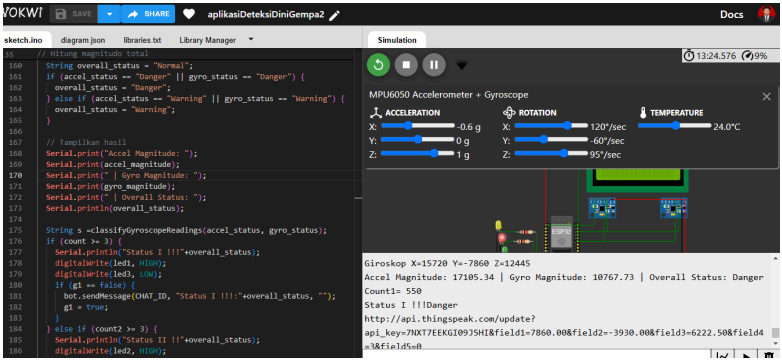
Berikut adalah nilai awal yang dapat digunakan sebagai referensi setelah kalibrasi:

- Accelerometer (Percepatan Linier):
    - Normal: < 1.5g
    - Warning: 1.5g - 3.0g
    - Danger: > 3.0g
- Gyroscope (Kecepatan Sudut):
    - Normal: < 100°/s
    - Warning: 100°/s - 200°/s
    - Danger: > 200°/s

Simpulan:

1. Angka min-max awal (misalnya, 2.0, 5.0) dapat digunakan sebagai referensi tetapi harus diuji dan disesuaikan melalui kalibrasi.

2. Kalibrasi sangat penting untuk memperbaiki bias bawaan sensor dan memastikan akurasi pembacaan.
3. Gunakan library MPU6050.h untuk mempermudah kalibrasi otomatis dan pengaturan skala penuh.
4. Setelah kalibrasi, rekam data pada berbagai skenario untuk menetapkan ambang batas yang tepat.



https://wokwi.com/projects/419206124340045825

```
 #include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu1(0x68); // Sensor pertama dengan alamat I2C default
MPU6050 mpu2(0x69); // Sensor kedua dengan alamat I2C yang dimodifikasi

// Variabel untuk menyimpan offset kalibrasi
float ax1_offset = 0, ay1_offset = 0, az1_offset = 0;
```

```
float gx1_offset = 0, gy1_offset = 0, gz1_offset = 0;

float ax2_offset = 0, ay2_offset = 0, az2_offset = 0;
float gx2_offset = 0, gy2_offset = 0, gz2_offset = 0;

void setup() {
  Serial.begin(115200);
  Wire.begin();

  // Inisialisasi kedua sensor
  mpu1.initialize();
  mpu2.initialize();

  if (!mpu1.testConnection() || !mpu2.testConnection()) {
    Serial.println("Salah satu sensor tidak terhubung!");
    while (1);
  }

  // Kalibrasi sensor
  calibrateMPU(mpu1, ax1_offset, ay1_offset, az1_offset,
gx1_offset, gy1_offset, gz1_offset);
  calibrateMPU(mpu2, ax2_offset, ay2_offset, az2_offset,
gx2_offset, gy2_offset, gz2_offset);

  Serial.println("Kalibrasi selesai!");
}

void loop() {
  int16_t ax1, ay1, az1, gx1, gy1, gz1;
  int16_t ax2, ay2, az2, gx2, gy2, gz2;

  // Baca data dari sensor 1
  mpu1.getMotion6(&ax1, &ay1, &az1, &gx1, &gy1, &gz1);
  float ax1_cal = ax1 - ax1_offset;
  float ay1_cal = ay1 - ay1_offset;
  float az1_cal = az1 - az1_offset;
  float gx1_cal = gx1 - gx1_offset;
  float gy1_cal = gy1 - gy1_offset;
  float gz1_cal = gz1 - gz1_offset;
```

```
  // Baca data dari sensor 2
  mpu2.getMotion6(&ax2, &ay2, &az2, &gx2, &gy2, &gz2);
  float ax2_cal = ax2 - ax2_offset;
  float ay2_cal = ay2 - ay2_offset;
  float az2_cal = az2 - az2_offset;
  float gx2_cal = gx2 - gx2_offset;
  float gy2_cal = gy2 - gy2_offset;
  float gz2_cal = gz2 - gz2_offset;

  // Hitung rata-rata accelerometer
  float ax_avg = (ax1_cal + ax2_cal) / 2.0;
  float ay_avg = (ay1_cal + ay2_cal) / 2.0;
  float az_avg = (az1_cal + az2_cal) / 2.0;

  // Hitung rata-rata gyroscope
  float gx_avg = (gx1_cal + gx2_cal) / 2.0;
  float gy_avg = (gy1_cal + gy2_cal) / 2.0;
  float gz_avg = (gz1_cal + gz2_cal) / 2.0;

  // Hitung magnitudo total
  float accel_magnitude = sqrt(ax_avg * ax_avg + ay_avg *
ay_avg + az_avg * az_avg);
  float gyro_magnitude = sqrt(gx_avg * gx_avg + gy_avg *
gy_avg + gz_avg * gz_avg);

  // Tentukan level gempa berdasarkan accelerometer
  String accel_status;
  if (accel_magnitude < 2.0) {
    accel_status = "Normal";
  } else if (accel_magnitude < 5.0) {
    accel_status = "Warning";
  } else {
    accel_status = "Danger";
  }

  // Tentukan level gempa berdasarkan gyroscope
  String gyro_status;
  if (gyro_magnitude < 3.0) {
    gyro_status = "Normal";
  } else if (gyro_magnitude < 6.0) {
```

```
    gyro_status = "Warning";
  } else {
    gyro_status = "Danger";
  }

  // Kombinasi status accelerometer dan gyroscope
  String overall_status = "Normal";
  if (accel_status == "Danger" || gyro_status == "Danger") {
    overall_status = "Danger";
  } else if (accel_status == "Warning" || gyro_status ==
"Warning") {
    overall_status = "Warning";
  }

  // Output status
  Serial.print("Accel Magnitude: ");
Serial.print(accel_magnitude);
  Serial.print(" | Gyro Magnitude: ");
Serial.print(gyro_magnitude);
  Serial.print(" | Overall Status: ");
Serial.println(overall_status);

  delay(500);
}

void calibrateMPU(MPU6050 &mpu, float &ax_offset, float
&ay_offset, float &az_offset,
                  float &gx_offset, float &gy_offset, float
&gz_offset) {
  Serial.println("Mulai kalibrasi sensor...");
  long ax_sum = 0, ay_sum = 0, az_sum = 0;
  long gx_sum = 0, gy_sum = 0, gz_sum = 0;

  for (int i = 0; i < 100; i++) {
    int16_t ax, ay, az, gx, gy, gz;
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    ax_sum += ax;
    ay_sum += ay;
    az_sum += az;
```

```
    gx_sum += gx;
    gy_sum += gy;
    gz_sum += gz;

    delay(10); // Delay untuk stabilisasi pembacaan
  }

  ax_offset = ax_sum / 100.0;
  ay_offset = ay_sum / 100.0;
  az_offset = az_sum / 100.0;
  gx_offset = gx_sum / 100.0;
  gy_offset = gy_sum / 100.0;
  gz_offset = gz_sum / 100.0;

  Serial.println("Kalibrasi selesai.");
  Serial.print("Offsets - AX: "); Serial.print(ax_offset);
  Serial.print(" AY: "); Serial.print(ay_offset);
  Serial.print(" AZ: "); Serial.print(az_offset);
  Serial.print(" GX: "); Serial.print(gx_offset);
  Serial.print(" GY: "); Serial.print(gy_offset);
  Serial.print(" GZ: "); Serial.println(gz_offset);
}
```

https://wokwi.com/projects/419210267490454529

Dengan kode ini, pembacaan sensor lebih akurat karena sudah melalui proses kalibrasi awal:

1. Fungsi Kalibrasi:
    - Kalibrasi dilakukan dengan membaca rata-rata 100 pembacaan sensor untuk setiap sumbu (X, Y, Z) dalam keadaan diam.

- Offset yang dihasilkan digunakan untuk mengoreksi pembacaan sensor saat digunakan.

2. Koreksi Data:
    - Offset yang diperoleh diterapkan ke pembacaan sensor untuk mendapatkan nilai yang sudah dikalibrasi (ax_cal, ay_cal, dll.).
3. Rata-Rata Dua Sensor:
    - Setelah data dikalibrasi, rata-rata dari kedua sensor dihitung untuk memperbaiki akurasi.
4. Magnitudo dan Status:
    - Hitungan magnitudo total (accel_magnitude dan gyro_magnitude) digunakan untuk menentukan level gempa berdasarkan ambang batas.

3.6 Akuisisi Data Ke IOT

Untuk mengirimkan data rata-rata gyroscope (X, Y, Z) ke ThingSpeak secara periodik setelah WiFi terhubung, perlu dibuat suatu fungsi khusus dalam program Arduino. Fungsi ini akan menggunakan library WiFiClient untuk mengirimkan data ke server ThingSpeak melalui

HTTP GET request. Berikut adalah implementasi lengkapnya:

1. Tambahkan library yang diperlukan:

```
#include <WiFi.h>
#include <HTTPClient.h>
```

2. Deklarasi variabel global:

```
const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* apiKey = "7NXT7EEKGI09J5HI";
const char* server = "http://api.thingspeak.com/update";
```

3. Buat fungsi untuk mengirim data ke ThingSpeak:

```
void sendDataToThingSpeak(float gx_avg, float gy_avg, float
gz_avg) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = String(server) + "?api_key=" + apiKey +
                 "&field1=" + String(gx_avg) +
                 "&field2=" + String(gy_avg) +
                 "&field3=" + String(gz_avg);
    http.begin(url);
    int httpResponseCode = http.GET();
    if (httpResponseCode > 0) {
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
    } else {
      Serial.print("Error code: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  } else {
    Serial.println("WiFi Disconnected");
  }
```

```
}
```

4. Modifikasi `setup()` untuk menghubungkan WiFi:

```
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password, 6);
  while (WiFi.status() != WL_CONNECTED) {
    delay(250);
    Serial.print(".");
  }
  Serial.println("Connected to WiFi");
}
```

5. Panggil `sendDataToThingSpeak()` secara periodik di `loop()`:

```
void loop() {
  // Contoh nilai rata-rata gyroscope
  float gx_avg = 12445.0; //Misal
  float gy_avg = 9825.0; //Misal
  float gz_avg = 7532.5; //Misal

  // Kirim data rata-rata gyroscope ke ThingSpeak
  sendDataToThingSpeak(gx_avg, gy_avg, gz_avg);

  // Tampilkan hasil rata-rata di serial monitor
  Serial.print("Rata-rata gyroscope X: ");
  Serial.println(gx_avg);
  Serial.print("Rata-rata gyroscope Y: ");
  Serial.println(gy_avg);
  Serial.print("Rata-rata gyroscope Z: ");
  Serial.println(gz_avg);
```

```
  Serial.println("+++++++++++++++");

  // Tunggu selama 15 detik sebelum mengirim data berikutnya
  delay(15000);
}
```

Maksud fungsi pada kode ini adalah:

- sendDataToThingSpeak() adalah fungsi yang mengirim data rata-rata gyroscope ke ThingSpeak menggunakan HTTP GET request.
- Fungsi setup() menghubungkan ESP32 ke WiFi.
- Fungsi loop() secara periodik mengirimkan data ke ThingSpeak setiap 15 detik dan menampilkan hasil rata-rata gyroscope di serial monitor.

Dengan implementasi ini, data gyroscope akan dikirim ke ThingSpeak secara berkala, dan status pengiriman akan ditampilkan di serial monitor.

Untuk menentukan kategori dari nilai rata-rata gyroscope gx_avg, gy_avg dan gz_avg yang masuk dalam kategori DANGER, WARNING, dan NORMAL, artinya pengguna bisa menggunakan ambang batas tertentu. Ambang batas ini biasanya didasarkan pada karakteristik dan kebutuhan spesifik dari aplikasi yang menggunakan sensor gyroscope. Berikut adalah contoh sederhana untuk menetapkan ambang batas ini:

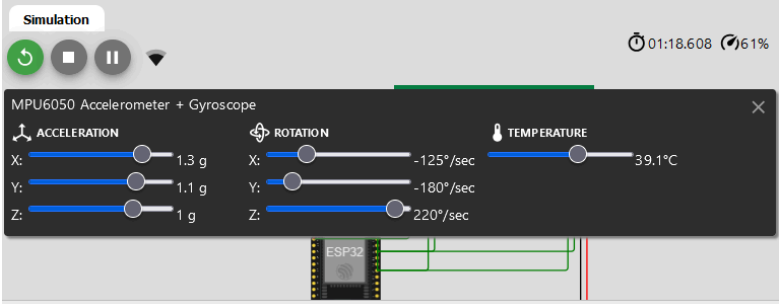
1. DANGER, Nilai rata-rata sangat tinggi, yang menunjukkan adanya gerakan atau getaran yang ekstrem.
2. WARNING, Nilai rata-rata berada di atas normal tetapi belum mencapai tingkat bahaya.
3. NORMAL, Nilai rata-rata berada dalam rentang yang diharapkan untuk kondisi normal.

Sebagai contoh, berikut adalah ambang batas yang dapat digunakan (dalam satuan yang sesuai, misalnya derajat per detik untuk gyroscope). Berikut adalah implementasi untuk mengklasifikasikan nilai rata-rata gyroscope:

- **DANGER**: $|gx\_avg| \geq 15000$ atau $|gy\_avg| \geq 15000$ atau $|gz\_avg| \geq 15000$
- **WARNING**: $|gx\_avg| \geq 10000$ atau $|gy\_avg| \geq 10000$ atau $|gz\_avg| \geq 10000$ tetapi kurang dari 15000
- **NORMAL**: $|gx\_avg| < 10000$ dan $|gy\_avg| < 10000$ dan $|gz\_avg| < 10000$

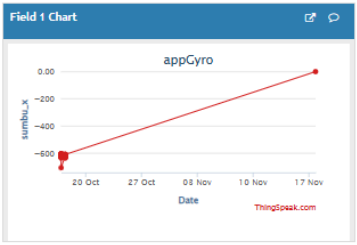
```
String classifyGyroscopeReadings(float gx_avg, float gy_avg, float
gz_avg) {
  if (abs(gx_avg) >= 15000 || abs(gy_avg) >= 15000 || abs(gz_avg) >=
15000) {
    return "DANGER";
  } else if (abs(gx_avg) >= 10000 || abs(gy_avg) >= 10000 ||
abs(gz_avg) >= 10000) {
    return "WARNING";
  } else {
    return "NORMAL";
  }
}
```

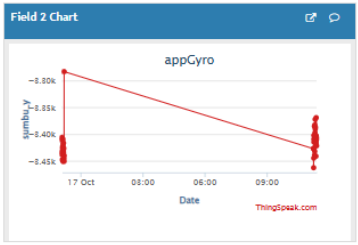
Dengan adanya Fungsi *classifyGyroscopeReadings* maka sistem akan mampu automatic mengklasifikasikan nilai rata-rata gyroscope ke dalam kategori DANGER, WARNING, atau NORMAL berdasarkan ambang batas yang telah ditentukan. Dalam penenmpatan di fungsi loop, nilai rata-rata gyroscope dihitung, diklasifikasikan, dan ditampilkan pada serial monitor. Data gyroscope juga dikirim ke ThingSpeak secara periodik setiap 15 detik.

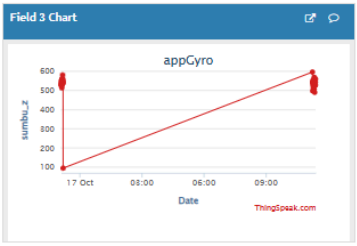


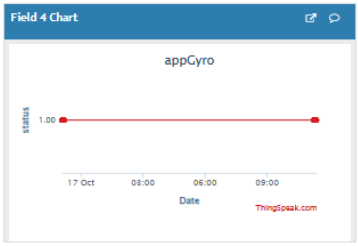

```
Data dari sensor 2 (0x69):
Akselerometer X=21299 Y=18022 Z=16384
Giroskop X=-16375 Y=-23580 Z=28820
Rata-rata gyroscope X: 327.50
Rata-rata gyroscope Y: -21615.00
Rata-rata gyroscope Z: 23252.50
++++++++++++++++++++++++++++++++++++++
http://api.thingspeak.com/update?
api_key=7NXT7EEKGI09J5HI&field1=327.50&field2=-21615.00&field3=23252.50&field4=DANGER&field5=
Via IOT
```

Dengan demikian, ini memberikan gambaran umum tentang bagaimana mengklasifikasikan data sensor gyroscope untuk mendeteksi kondisi yang berpotensi berbahaya, serta bagaimana mengirimkan data tersebut ke ThingSpeak.









BAGIAN 4

Android Studio Kotlin – Java

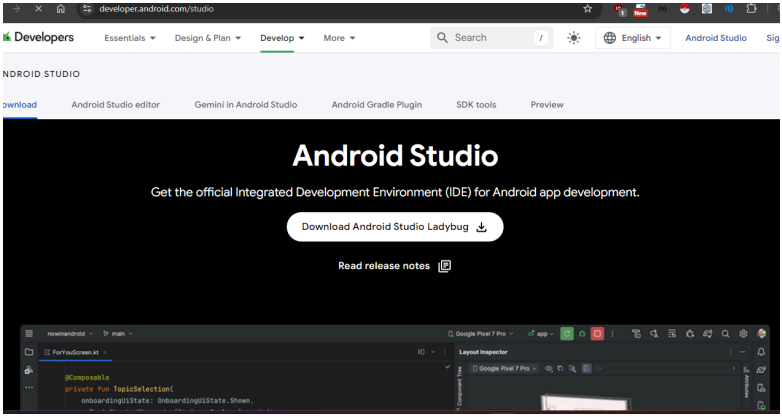
4.1 Android Studio

Android Studio adalah Integrated Development Environment (IDE) resmi yang dikembangkan oleh Google untuk membangun aplikasi Android. Android Studio didasarkan pada IntelliJ IDEA dan dirancang khusus untuk pengembangan aplikasi Android, menyediakan berbagai alat seperti editor kode, debugger, emulator, serta integrasi dengan sistem build berbasis Gradle.

Dalam versi terbaru, Android Studio mendukung dua bahasa pemrograman utama untuk pengembangan aplikasi Android: Kotlin dan Java. Kotlin telah menjadi bahasa pemrograman resmi Android sejak 2017, dan Android Studio sekarang menggunakan Kotlin secara default dalam konfigurasi Gradle, yang merupakan sistem build yang digunakan untuk mengatur dependensi, plugin, dan pengaturan proyek. Meskipun Kotlin digunakan di Gradle, Android Studio tetap mendukung penggunaan Java sebagai bahasa pemrograman utama untuk pengembangan aplikasi.

Alamat Download Android Studio

Pengguna dapat mengunduh Android Studio dari situs resmi Google dengan url ini: https://developer.android.com/studio.



Cara Instalasi Android Studio

Berikut langkah-langkah untuk menginstal Android Studio pada suatu sistem operasi :

1. Download Android Studio:
    - Kunjungi situs resmi Android Studio di https://developer.android.com/studio.
    - Pilih versi Android Studio yang sesuai dengan sistem operasi yang dimiliki (Windows, macOS, atau Linux).
    - Klik tombol Download untuk memulai pengunduhan.
2. Instal Android Studio (Langkah untuk Windows, macOS, dan Linux):
    - Windows:

        - Buka file yang diunduh (.exe) dan ikuti petunjuk instalasi.
        - Pilih lokasi instalasi dan komponen tambahan yang diinginkan (misalnya, Android Virtual Device).
        - Setelah instalasi selesai, klik Finish untuk memulai Android Studio.
    - macOS:
        - Buka file .dmg yang diunduh.
        - Seret dan jatuhkan ikon Android Studio ke folder Applications.
        - Setelah selesai, buka Android Studio dari folder Applications.
    - Linux:
        - Ekstrak file .zip yang diunduh ke direktori pilihan Anda.
        - Buka terminal dan jalankan file studio.sh yang ada di dalam direktori bin untuk memulai Android Studio.
3. Konfigurasi Awal:
    - Setelah Android Studio dijalankan, ikuti wizard konfigurasi untuk mengatur SDK Android dan menginstal emulator jika diperlukan.

- Pilih mode standar saat diminta untuk memilih pengaturan awal.
- Tunggu hingga komponen yang dibutuhkan diunduh, termasuk SDK Android dan alat pengembang lainnya.

4. Siapkan SDK Manager:
    - Buka SDK Manager melalui menu File > Settings > Appearance & Behavior > System Settings > Android SDK.
    - Pastikan versi SDK yang diperlukan diinstal (misalnya, SDK versi terbaru dan Android API level tertentu).

Langkah Membuat Halaman Baru (New Project)

Setelah Android Studio terinstal, Pengguna dapat mulai membuat proyek baru dengan mengikuti langkah-langkah berikut:

1. Buka Android Studio:
    - Luncurkan Android Studio setelah instalasi selesai.
    - Pada layar pembuka, klik New Project.
2. Konfigurasi Proyek Baru:

- Select Project Template: Pilih template yang ingin Pengguna gunakan untuk memulai proyek, seperti Empty Activity atau Basic Activity. Untuk memulai sederhana, pilih Empty Activity.
- Klik Next untuk melanjutkan.

3. Isi Detail Proyek:
    - Name: Masukkan nama proyek Anda. Ini akan menjadi nama aplikasi.
    - Package Name: Secara otomatis diisi berdasarkan nama proyek, dan silakan untuk bisa menyesuaikannya. Package name harus unik.
    - Save Location: Pilih folder tempat proyek akan disimpan di komputer Anda.
    - Language: Pilih bahasa pemrograman yang akan digunakan untuk proyek. Pilih Java jika ingin menggunakan Java, atau pilih Kotlin jika ingin menggunakan Kotlin.
    - Minimum API Level: Pilih versi minimum SDK Android yang akan didukung aplikasi Anda. Semakin rendah API level yang dipilih, semakin banyak perangkat yang bisa menggunakan aplikasi Anda, tetapi adalah

mungkin perlu menyesuaikan dengan fitur-fitur yang ada di versi tersebut.

- Klik Finish untuk memulai proyek.

4. Proyek Siap Digunakan:
    - Setelah klik Finish, Android Studio akan membuat struktur proyek Anda, termasuk file MainActivity.java (atau MainActivity.kt jika menggunakan Kotlin), file XML untuk UI, dan konfigurasi Gradle.
    - Dan sekarang dapat mulai mengedit kode di editor Android Studio.
5. Menambahkan Halaman Baru (Activity Baru):
    - Jika ingin menambahkan halaman baru (Activity) ke proyek, klik kanan pada folder app > java > [package name] di dalam Project Explorer.
    - Pilih New > Activity > Empty Activity.
    - Beri nama activity baru (misalnya SecondActivity) dan klik Finish.
    - Android Studio akan membuat file SecondActivity.java (atau SecondActivity.kt) serta layout XML untuk activity baru ini.

- Pengguna sekarang dapat menambahkan kode dan elemen UI untuk halaman baru tersebut.

Dengan mengikuti langkah-langkah ini, maka akan berhasil dalam menginstal Android Studio, membuat proyek baru, dan menambahkan halaman tambahan ke aplikasi Android yang sedang dikembangkan.

4.2 Layout Aplikasi

Di dalam Android Studio, tampilan atau layout aplikasi Android dibuat menggunakan XML (eXtensible Markup Language). Layout ini berfungsi untuk mendesain antarmuka pengguna (UI) yang akan ditampilkan pada perangkat Android. Di dalam XML, Android menyediakan banyak sekali widget atau elemen UI, seperti Button, TextView, EditText, dan lain-lain, yang bisa diatur dengan berbagai properti seperti ukuran font, warna teks, warna latar belakang, dan tata letak.

Dalam pembuatan aplikasi client monitoring data gyroscope dari database, layout yang didesain dengan XML akan memainkan peran penting dalam menampilkan data yang diperoleh dari server atau database IoT. Berikut adalah

penjelasan tentang komponen umum dan properti yang sering digunakan dalam desain layout di Android Studio:

Widget Umum dalam Layout Android:

1. TextView:
    - Digunakan untuk menampilkan teks statis kepada pengguna. Misalnya, untuk menampilkan hasil data gyroscope seperti nilai Sumbu X, Sumbu Y, dan Sumbu Z.
    - Contoh kode XML:

```
<TextView
   android:id="@+id/gyroscope_x"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Sumbu X: 0"
   android:textSize="16sp"
   android:textColor="#000000"
   android:background="#FFFFFF" />
```

- Properti penting:
    - android:text: Menentukan teks yang akan ditampilkan.
    - android:textSize: Mengatur ukuran font (misalnya, 16sp).

- android:textColor: Mengatur warna teks (misalnya, #000000 untuk warna hitam).
- android:background: Mengatur warna latar belakang (misalnya, #FFFFFF untuk putih).

2. Button:

- Digunakan untuk aksi seperti mengirim data atau mengambil data terbaru dari server.
- Contoh kode XML:

```
<Button
    android:id="@+id/refresh_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Refresh Data"
    android:textSize="18sp"
    android:backgroundTint="#3F51B5"
    android:textColor="#FFFFFF" />
```

- Properti penting:
  - android:text: Menentukan teks pada button (misalnya, "Refresh Data").
  - android:backgroundTint: Mengatur warna latar belakang tombol.

- android:textColor: Mengatur warna teks di tombol.

3. EditText:

- Digunakan untuk menerima input dari pengguna. Misalnya, jika ada kebutuhan untuk input manual dari nilai ambang batas (threshold) untuk gyroscope.
- Contoh kode XML:

```
<EditText
    android:id="@+id/input_threshold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Enter threshold"
    android:textColor="#000000"
    android:background="#EEEEEE"
    android:inputType="number" />
```

Properti penting:

- android:hint: Memberikan petunjuk kepada pengguna mengenai data yang harus dimasukkan.
- android:textColor: Mengatur warna teks yang dimasukkan.

- android:background: Mengatur warna latar belakang kotak input.
- android:inputType: Menentukan jenis input, seperti teks, angka, email, dl

Membuat Aplikasi Monitoring Data Gyroscope:

Dalam aplikasi Client Monitoring Data Gyroscope yang menggunakan Android Studio, elemen-elemen TextView dapat digunakan untuk menampilkan data gyroscope yang diperoleh dari database. Sebagai contoh, nilai Sumbu X, Sumbu Y, dan Sumbu Z dari data gyroscope dapat diambil dari API server dan ditampilkan di aplikasi. Selain itu, Button dapat digunakan untuk mengambil data terbaru dari server dengan cara mengirim request ke API.

Dengan memanfaatkan TextView, Button, dan layout seperti LinearLayout atau ConstraintLayout, serta memodifikasi properti yang diperlukan seperti ukuran font, warna teks, dan tata letak, Pengguna dapat membuat antarmuka yang informatif dan mudah digunakan untuk memantau data gyroscope dalam aplikasi Android.

4.3 Aplikasi Monitoring

Setelah layout XML di setiap halaman selesai, setiap halaman seperti halaman loading, menu utama, list arsip, detail info data monitoring, halaman about, dan halaman view real-time sensor dibuatkan masing-masing file Java. Dalam file Java tersebut, dilakukan penyesuaian layout dengan cara mengakses ID elemen-elemen layout yang telah didefinisikan di XML melalui findViewById(). Elemen-elemen seperti TextView, Button, RecyclerView, atau ListView akan dipetakan ke variabel di Java sehingga dapat diakses dan dimodifikasi sesuai kebutuhan. Dengan ini, aplikasi bisa mengatur interaksi antara logika bisnis dan antarmuka pengguna, seperti menampilkan data yang didapatkan dari server IoT, mengupdate status real-time, atau menavigasi antar halaman di aplikasi

Contoh Implementasi:

Untuk halaman real-time sensor, misalnya:

```
<!-- Layout XML: activity_realtime.xml -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
```

```
        android:id="@+id/textGyroscopeX"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sumbu X: 0" />

    <TextView
        android:id="@+id/textGyroscopeY"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sumbu Y: 0" />

    <TextView
        android:id="@+id/textGyroscopeZ"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sumbu Z: 0" />

    <Button
        android:id="@+id/btnRefresh"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Refresh Data" />
</LinearLayout>
```

Lalu kode Java nya adalah:

```
// RealtimeSensorActivity.java
package com.example.gyroscopemonitoring;

import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class RealtimeSensorActivity extends
AppCompatActivity {

    private TextView textGyroscopeX, textGyroscopeY,
textGyroscopeZ;
    private Button btnRefresh;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_realtime); //
Menghubungkan ke layout XML

        // Menghubungkan ID elemen di XML dengan variabel
Java
        textGyroscopeX = findViewById(R.id.textGyroscopeX);
        textGyroscopeY = findViewById(R.id.textGyroscopeY);
        textGyroscopeZ = findViewById(R.id.textGyroscopeZ);
        btnRefresh = findViewById(R.id.btnRefresh);

        // Misalnya: Set OnClickListener untuk tombol Refresh
        btnRefresh.setOnClickListener(view -> {
            // Logika untuk mengambil data baru dari sensor
atau server IoT
            fetchGyroscopeData();
        });
    }

    private void fetchGyroscopeData() {
        // Contoh logika untuk mengupdate nilai gyroscope
        textGyroscopeX.setText("Sumbu X: 101");
        textGyroscopeY.setText("Sumbu Y: 102");
        textGyroscopeZ.setText("Sumbu Z: 103");
    }
}
```

Pada contoh ini, elemen-elemen TextView dan Button yang telah didefinisikan di dalam XML dihubungkan ke variabel Java menggunakan metode findViewById(). Setelah itu, interaksi seperti memperbarui nilai gyroscope atau menambahkan event listener (seperti klik tombol) dapat dilakukan dalam logika program Java. Proses ini dilakukan untuk setiap halaman aplikasi seperti halaman loading, menu utama, list arsip, detail info data monitoring, dan about page, sehingga elemen-elemen dari layout dapat dimanipulasi dengan logika aplikasi.

Hal ini menciptakan interaksi yang dinamis antara tampilan (UI) dan kode bisnis aplikasi, yang mendukung pengembangan aplikasi yang lebih interaktif dan fungsional.

4.4 Display Aplikasi

Dalam aplikasi monitoring data sensor, terutama untuk data dari gyroscope, setiap halaman menu memiliki maksud dan tujuan tertentu yang mendukung fungsionalitas aplikasi secara keseluruhan. Berikut penjelasan mengenai maksud dan tujuan dari masing-masing halaman yang telah dibuat:

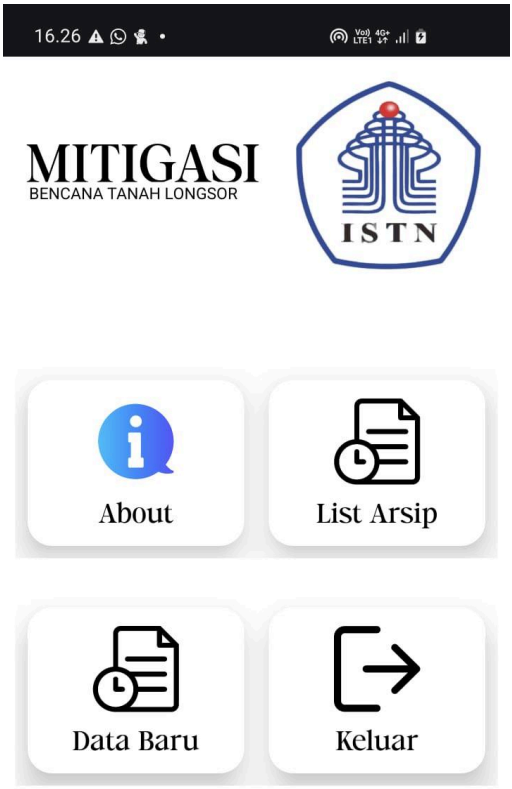
1. Halaman Loading,

- Maksud: Halaman ini ditampilkan ketika aplikasi pertama kali dibuka. Biasanya, halaman ini memuat logo atau nama aplikasi sementara aplikasi sedang mempersiapkan data yang diperlukan atau melakukan inisialisasi awal.
- Tujuan: Memberikan indikasi kepada pengguna bahwa aplikasi sedang mempersiapkan sumber daya (seperti koneksi ke database atau memuat data dari server). Hal ini juga memberikan pengalaman pengguna yang lebih mulus dengan menunggu hingga aplikasi siap digunakan.

2. Menu Utama,

- Maksud: Menu utama adalah halaman pertama yang diakses setelah loading selesai. Di halaman ini, pengguna dapat memilih opsi untuk melanjutkan ke berbagai fitur aplikasi seperti melihat data arsip, mengakses informasi real-time, atau membaca informasi tentang aplikasi.
- Tujuan: Memberikan pengguna navigasi yang mudah dan cepat ke berbagai fitur utama aplikasi. Di sinilah semua fitur penting aplikasi disajikan dalam bentuk tombol atau menu, sehingga

pengguna dapat memilih apa yang ingin mereka lakukan selanjutnya.



3. List Arsip,

- Maksud: Halaman ini berisi daftar arsip data monitoring yang telah disimpan dari waktu ke waktu. Setiap entri dalam arsip biasanya mencakup

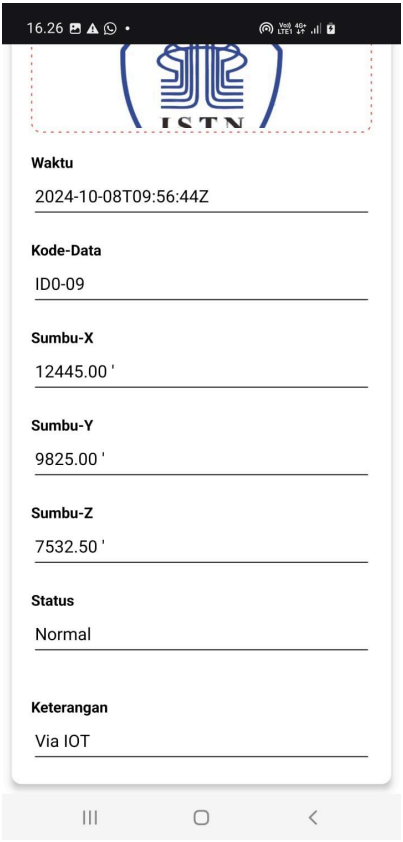
waktu pengambilan data, nilai gyroscope, serta status (Normal, Warning, Danger).

- Tujuan: Memberikan akses kepada pengguna untuk melihat data historis yang telah dikumpulkan oleh sistem. Arsip ini berguna untuk menganalisis tren data dari waktu ke waktu atau untuk memeriksa kondisi pada waktu tertentu yang dapat membantu dalam memahami pola dan deteksi anomali.

4. Detail Info Data Monitoring dari Arsip,

- Maksud: Ketika pengguna memilih salah satu entri arsip, mereka diarahkan ke halaman ini yang berisi rincian lengkap dari data yang dipilih. Rincian ini bisa mencakup nilai pada setiap sumbu gyroscope (X, Y, Z), waktu pengambilan data, status (Normal, Warning, Danger), dan informasi tambahan jika ada.

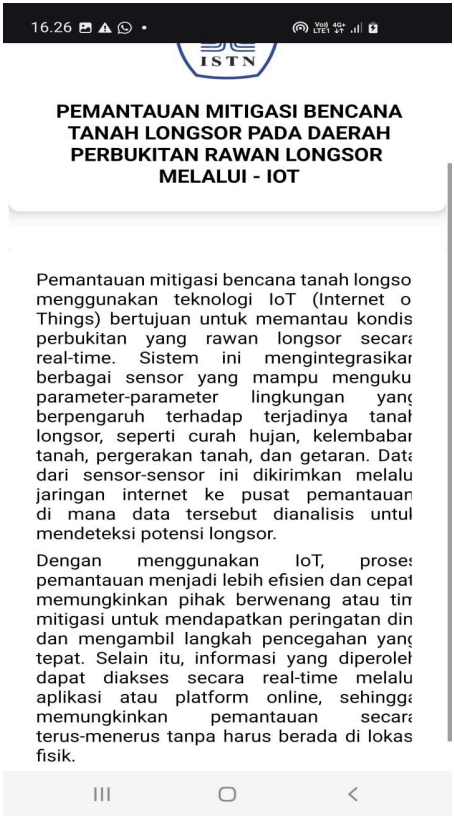
- Tujuan: Memberikan pengguna kemampuan untuk melihat detail data lebih dalam dari catatan monitoring yang disimpan di arsip. Ini membantu pengguna untuk memahami secara spesifik bagaimana kondisi pada waktu tertentu, terutama dalam hal pemantauan kondisi tanah atau kemiringan (misalnya untuk deteksi dini bencana).



5. Halaman About (Tentang),

- Maksud: Halaman About atau Tentang berisi informasi mengenai aplikasi, seperti siapa pengembangnya, versi aplikasi, teknologi yang digunakan, dan informasi kontak jika pengguna membutuhkan bantuan lebih lanjut.

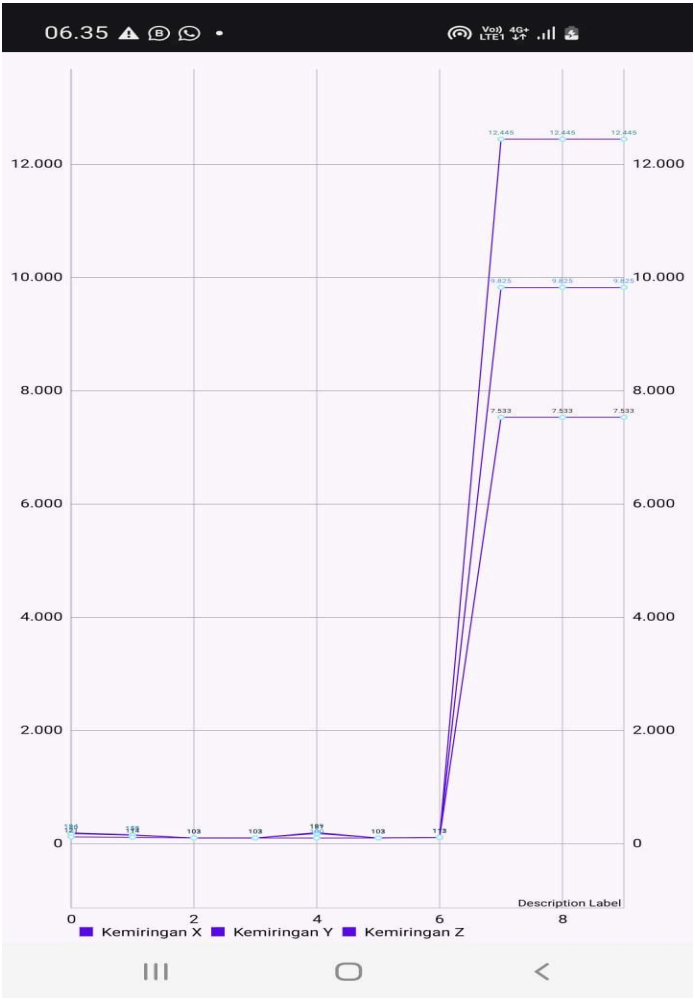
- Tujuan: Memberikan informasi latar belakang tentang aplikasi kepada pengguna, serta memberikan akses untuk mendapatkan bantuan atau dukungan jika diperlukan. Halaman ini juga berfungsi untuk membangun kepercayaan pengguna dengan mengetahui bahwa aplikasi ini dikelola dan dikembangkan secara resmi.



6. Halaman View Real-Time Sensor,

- Maksud: Halaman ini menampilkan data sensor secara real-time yang diperoleh langsung dari sensor gyroscope yang dihubungkan ke server IoT atau database ThingSpeak. Data yang ditampilkan meliputi nilai Sumbu X, Y, Z, serta status kondisi saat ini (Normal, Warning, Danger).

- Tujuan: Halaman ini adalah inti dari aplikasi monitoring, di mana pengguna dapat memantau data real-time yang terus diperbarui. Fungsinya adalah untuk memberikan informasi langsung mengenai kondisi yang dipantau, sehingga pengguna dapat merespons lebih cepat jika terjadi kondisi yang berbahaya, seperti potensi tanah longsor.



Pada aplikasi monitoring ini, pada masing-masing maneu/page/halaman memiliki peran penting dalam mendukung keseluruhan fungsi aplikasi monitoring ini. Mulai dari menyediakan navigasi yang mudah, menyimpan

dan menampilkan data arsip, hingga menampilkan data real-time yang penting, setiap halaman mendukung pengalaman pengguna yang informatif dan interaktif. Hal ini memastikan bahwa pengguna dapat memantau kondisi secara efektif, memahami data yang disimpan, dan mendapatkan informasi yang relevan mengenai aplikasi yang digunakan.

BAGIAN 5

Pengujian Blackbox dan Whitebox Sistem

5.1 Pengujian Alat

Untuk pengujian sistem pemantauan mitigasi bencana tanah longsor berbasis ESP32 yang mengirimkan data dari beberapa sensor gyroscope melalui Wi-Fi ke database IoT dengan status Danger/Warning/Normal, berikut adalah beberapa aktivitas yang bisa dilakukan dalam pengujian Blackbox dan Whitebox.

Pengujian Blackbox

Blackbox testing berfokus pada pengujian fungsionalitas sistem tanpa mengetahui detail implementasi kode atau arsitektur internalnya. Pengujian ini dilakukan dari perspektif pengguna akhir untuk memastikan bahwa sistem bekerja sesuai dengan persyaratan yang telah ditentukan. Beberapa aktivitas yang dapat dilakukan dalam pengujian ini meliputi:

1. Pengujian Pengiriman Data Sensor:
    - Pastikan bahwa data dari beberapa sensor gyroscope dikirimkan secara real-time ke platform IoT (ThingSpeak) melalui sinyal Wi-Fi.
    - Verifikasi bahwa data yang dikirim adalah rata-rata (average) dari data gyroscope yang diperoleh.
2. Pengujian Status (Danger/Warning/Normal):
    - Uji apakah sistem mampu menentukan status berdasarkan nilai threshold tertentu (misalnya, kondisi Danger jika kemiringan melebihi 20 derajat, Warning jika antara 10-20 derajat, dan Normal jika di bawah 10 derajat).

- Uji setiap kondisi threshold dengan input nilai yang berbeda-beda dan pastikan status yang ditampilkan benar.

3. Pengujian Pengiriman Status ke IoT:
    - Verifikasi bahwa status yang dihitung (Danger/Warning/Normal) dikirimkan ke platform ThingSpeak dan tersimpan dengan baik dalam database IoT.
    - Pastikan status ini dapat diakses melalui dashboard IoT dan juga dilihat melalui aplikasi Android yang dikembangkan untuk memonitoring kondisi.
4. Pengujian Wi-Fi Connectivity:
    - Uji koneksi Wi-Fi pada berbagai skenario, seperti sinyal Wi-Fi lemah, kehilangan koneksi, atau kondisi jaringan yang tidak stabil, untuk memastikan bahwa data tetap terkirim ketika koneksi kembali stabil.
    - Uji apa yang terjadi jika Wi-Fi tidak tersedia, apakah sistem dapat menyimpan data sementara dan mengirimkan setelah koneksi kembali tersedia (retry mechanism).
5. Pengujian Antarmuka Aplikasi Android:

- Uji aplikasi Android untuk memastikan bahwa data gyroscope dan status (Danger/Warning/Normal) dapat dipantau dengan benar.
- Verifikasi bahwa aplikasi Android mampu menerima update status secara real-time dari IoT platform, dengan tampilan yang benar sesuai kondisi aktual.

6. Pengujian Threshold Adjustment:
    - Uji apakah ada kemampuan untuk mengatur nilai threshold secara dinamis melalui aplikasi atau server, dan verifikasi apakah perubahan ini mempengaruhi pengambilan keputusan status (Danger/Warning/Normal) secara real-time.

Pengujian Whitebox:

Whitebox testing dilakukan dengan pengetahuan penuh tentang arsitektur dan implementasi internal sistem. Tujuannya adalah untuk menguji logika dan struktur kode, serta memastikan bahwa setiap bagian sistem berfungsi sebagaimana mestinya. Beberapa aktivitas dalam pengujian Whitebox meliputi:

1. Pengujian Pengambilan Data dari Sensor Gyroscope:
    - Uji fungsi pembacaan data gyroscope secara langsung pada tingkat kode untuk memastikan bahwa data yang diperoleh benar dan akurat.
    - Lakukan debugging pada kode pengambilan data dari setiap sensor untuk memastikan bahwa nilai yang diambil merupakan data yang valid dan sesuai.
2. Pengujian Algoritma Perhitungan Rata-rata :
    - Uji algoritma yang digunakan untuk menghitung rata-rata data dari beberapa sensor gyroscope.
    - Pastikan tidak ada kesalahan dalam logika perhitungan, seperti pengabaian nilai tertentu atau penanganan kondisi ekstrem seperti data yang hilang atau rusak.
3. Pengujian Kondisi Threshold Status :
    - Uji logika pemrosesan kondisi threshold yang menentukan status berdasarkan kemiringan tanah. Verifikasi bahwa setiap kondisi yang memicu status Danger, Warning, atau Normal sudah diimplementasikan dengan benar.

- Uji apakah perubahan pada nilai threshold diimplementasikan dengan benar dalam kode dan mempengaruhi pengambilan keputusan secara real-time.

4. Pengujian Pengiriman Data ke IoT JSON API:
    - Uji fungsi dalam kode yang bertanggung jawab untuk mengirimkan data ke server IoT (ThingSpeak) melalui JSON API. Pastikan bahwa format data yang dikirim sudah benar dan sesuai dengan spesifikasi API.
    - Verifikasi apakah ada mekanisme penanganan error jika pengiriman data gagal, dan apakah ada retry logic jika Wi-Fi mengalami gangguan.
5. Pengujian Wi-Fi Module pada ESP32:
    - Uji modul Wi-Fi pada ESP32 untuk memastikan bahwa koneksi ke jaringan stabil dan dapat memulihkan koneksi dengan benar ketika terjadi gangguan.
    - Lakukan stress test untuk menguji performa koneksi Wi-Fi dalam jangka waktu panjang dan dalam kondisi jaringan yang berubah-ubah.
6. Pengujian I2C Bus:

- Uji implementasi protokol I2C yang menghubungkan sensor gyroscope dan LCD 16x2 untuk memastikan bahwa data dikomunikasikan dengan benar melalui jalur SDA dan SCL.
- Lakukan pengecekan pada timing dan addressing untuk memastikan bahwa tidak terjadi konflik antar perangkat yang terhubung melalui I2C bus.

7. Pengujian Penanganan Data di ESP32:
    - Uji bagaimana ESP32 menangani data yang diperoleh dari sensor, termasuk cara penyimpanan sementara dan penjadwalan pengiriman data ke server IoT.
    - Verifikasi manajemen memori dan proses buffering agar tidak terjadi penumpukan data atau crash ketika data yang diambil melebihi kapasitas memori yang tersedia.

Dengan kombinasi pengujian blackbox dan whitebox, pengujian menyeluruh dapat dilakukan baik dari sisi fungsionalitas maupun struktur internal sistem. Ini akan memastikan bahwa sistem pemantauan mitigasi bencana tanah longsor ini berfungsi dengan baik, stabil, dan dapat

diandalkan dalam memberikan data yang akurat serta peringatan dini yang tepat.

5.2 Pengujian Android

Untuk pengujian secara Blackbox dan Whitebox dari sisi client (smartphone berbasis Android) yang berfungsi untuk memantau data dari ESP32, berikut adalah aktivitas pengujian yang dilakukan untuk memastikan aplikasi berfungsi dengan baik pada setiap fitur dan menu yang tersedia.

Pengujian Blackbox:

Pengujian Blackbox fokus pada pengujian fungsionalitas tanpa melihat implementasi kode. Pengujian ini dilakukan dari perspektif pengguna akhir untuk memastikan bahwa semua fungsi dalam aplikasi bekerja sesuai harapan.

1. Pengujian Halaman Loading:
    - Uji apakah aplikasi menampilkan halaman loading saat pertama kali dibuka.
    - Verifikasi bahwa halaman loading muncul dengan benar dan mengarahkan pengguna

ke menu utama setelah beberapa saat atau setelah data awal telah dimuat.

2. Pengujian Navigasi dari Menu Utama:
    - Uji apakah menu utama memuat dengan benar dan terdapat empat tombol yang dapat diklik:
        - Tombol ke halaman bantuan atau profil aplikasi: Verifikasi bahwa ketika tombol ini diklik, pengguna diarahkan ke halaman yang berisi informasi bantuan atau profil aplikasi.
        - Tombol ke menu arsip data average gyro: Uji apakah tombol ini membuka halaman arsip data yang berisi daftar data historis dari sensor gyroscope.
        - Tombol ke halaman informasi realtime: Uji apakah tombol ini membuka halaman yang menampilkan data real-time dari sensor gyroscope.
3. Pengujian Tampilan Data Arsip:
    - Verifikasi bahwa halaman arsip menampilkan daftar data average gyroscope yang

tersimpan, termasuk waktu pembuatan dan status (Normal, Warning, Danger).

- Klik pada salah satu item di arsip dan pastikan bahwa aplikasi menampilkan halaman rincian data yang berisi detail seperti:
  - Tanggal dan waktu data.
  - Kode Data.
  - Nilai Sumbu-X, Y, dan Sumbu-Z.
  - Status (Normal, Warning, Danger).
  - Keterangan tambahan jika ada.

4. Pengujian Halaman Data Realtime:
   - Uji apakah halaman data real-time menampilkan data average gyroscope secara real-time yang diambil dari server (ThingSpeak atau database IoT lainnya).
   - Pastikan data terus diperbarui dalam interval waktu tertentu dan sesuai dengan data yang dikirim dari server.
5. Pengujian Kesesuaian Tampilan Status:
   - Verifikasi apakah status seperti Normal, Warning, dan Danger ditampilkan dengan benar pada halaman arsip dan halaman

real-time, sesuai dengan ambang batas yang ditentukan.

- Uji bagaimana aplikasi menampilkan informasi status saat kondisi data berubah, misalnya dari Normal ke Warning.

6. Pengujian Responsivitas dan Performa Aplikasi:
    - Uji responsivitas aplikasi saat berpindah dari satu halaman ke halaman lain, terutama saat membuka arsip data atau halaman real-time yang memuat data dari server.
    - Verifikasi performa aplikasi dengan memuat banyak data historis di halaman arsip, untuk memastikan aplikasi tidak lambat atau crash.

Pengujian Whitebox:

Pengujian Whitebox fokus pada pengujian struktur internal aplikasi, termasuk logika kode dan cara aplikasi memproses data. Berikut adalah beberapa aktivitas yang dapat dilakukan dalam pengujian ini:

1. Pengujian Logika Navigasi:
    - Verifikasi logika di balik navigasi antar halaman, seperti dari halaman loading ke menu utama, dan dari menu utama ke

halaman lain (bantuan, arsip data, dan halaman data real-time).

- Uji apakah ada error handling untuk situasi ketika halaman gagal dimuat, misalnya ketika data gagal diambil dari server.

2. Pengujian Fungsi Pengambilan Data dari Server:
    - Uji fungsi yang bertanggung jawab untuk mengambil data dari server IoT (ThingSpeak) menggunakan JSON API.
    - Pastikan bahwa setiap request yang dilakukan berhasil, dan respon yang diterima dari server diolah dengan benar oleh aplikasi.
    - Verifikasi apakah ada mekanisme retry atau error handling ketika koneksi internet terputus atau data gagal diambil.
3. Pengujian Pengolahan Data Gyroscope:
    - Uji bagaimana aplikasi mengolah data JSON yang diterima dari server, khususnya untuk menampilkan average data gyroscope dan status (Normal, Warning, Danger).
    - Verifikasi apakah logika di balik pengolahan data ini sudah sesuai, seperti perhitungan

average atau menampilkan nilai yang benar pada setiap sumbu (X, Y, Z).

4. Pengujian Penanganan Status:
    - Uji bagaimana aplikasi mengidentifikasi status berdasarkan nilai threshold yang diterima dari server, dan apakah logika yang digunakan sesuai dengan ambang batas yang sudah diatur.
    - Pastikan bahwa aplikasi menggunakan mekanisme validasi status yang benar untuk memutuskan status yang ditampilkan di halaman arsip dan real-time.
5. Pengujian Penyimpanan dan Pengelolaan Arsip:
    - Uji cara aplikasi menyimpan dan memuat data historis (arsip data) dari sensor gyroscope. Verifikasi bahwa data tersebut tersimpan dengan format yang benar, serta dapat diambil dan ditampilkan saat diperlukan.
    - Pastikan bahwa data yang ditampilkan pada halaman arsip benar-benar berasal dari sumber yang valid, dan tidak ada kesalahan dalam pengambilan atau penempatan data.
6. Pengujian Error Handling:

- Verifikasi apakah aplikasi dapat menangani skenario error seperti kegagalan mengambil data dari server, timeout koneksi, atau masalah pada jaringan Wi-Fi.
- Uji apakah ada notifikasi atau pesan error yang diberikan kepada pengguna saat terjadi masalah, dan apakah aplikasi dapat memulihkan diri setelah error terjadi.

7. Pengujian Efisiensi Penggunaan Memori dan CPU:
   - Uji performa aplikasi dengan memantau penggunaan memori dan CPU saat aplikasi memuat data real-time dan data arsip dalam jumlah besar.
   - Pastikan aplikasi tidak mengalami leak memori atau masalah performa yang dapat menyebabkan aplikasi menjadi lambat atau crash.

Dengan pengujian Blackbox dan Whitebox ini, semua aspek dari aplikasi Android yang berfungsi sebagai client untuk memantau data dari sensor gyroscope dapat diperiksa secara menyeluruh. Pengujian ini akan memastikan bahwa aplikasi dapat berfungsi dengan baik, responsif, dan stabil

dalam mengolah data dan menampilkan informasi yang diperlukan oleh pengguna.

5.3 Pengujian Database

Untuk pengujian Blackbox dan Whitebox dari sisi database ThingSpeak serta pengaturan JSON API yang digunakan untuk menyimpan dan menarik data dari sensor, pengujian perlu dilakukan secara menyeluruh untuk memastikan bahwa komunikasi antara sistem pemantauan (ESP32) dan server ThingSpeak berjalan dengan baik. Berikut adalah aktivitas pengujian yang dapat dilakukan untuk memastikan bahwa sistem berfungsi sesuai dengan spesifikasi.

Pengujian Blackbox:

Pengujian Blackbox berfokus pada pengujian fungsionalitas API dan database ThingSpeak tanpa melihat detail implementasi internal dari sistem ThingSpeak itu sendiri. Fokusnya adalah untuk memastikan bahwa data dapat ditulis dan dibaca dengan benar.

1. Pengujian Pengiriman Data (Write a Channel Feed), yaitu dengan melakukan request API untuk menulis data ke channel ThingSpeak menggunakan format berikut:

https://api.thingspeak.com/update?api_key=7NXT7EEKGI09J5HI&field1=0

Lalu Uji apakah data dapat dikirimkan dari perangkat ESP32 ke ThingSpeak.Dan verifikasi bahwa data yang dikirim (nilai dari gyroscope, status danger/warning/normal) tersimpan di ThingSpeak dan muncul di channel yang tepat.

2. Pengujian Pembacaan Data (Read a Channel Feed), Lakukan request API untuk membaca feed dari channel ThingSpeak:

GET https://api.thingspeak.com/channels/2687137/feeds.json?results=2

Uji apakah data yang disimpan dapat dibaca kembali dari channel ThingSpeak dengan hasil yang sesuai.Dan verifikasi apakah hasilnya mencakup semua field yang diperlukan seperti nilai sumbu X, Y, Z, status, dan waktu pembuatan (created_at).

3. Pengujian Pembacaan Field Spesifik (Read a Channel Field), Uji API yang digunakan untuk membaca data dari field spesifik di ThingSpeak:

```
GET
https://api.thingspeak.com/channels/2687137/fields/1.json?results=2
```

Verifikasi bahwa data yang dikembalikan adalah nilai untuk field yang benar (misalnya, nilai sumbu X jika field 1 berisi nilai dari gyroscope X).Dan uji dengan beberapa request untuk field yang berbeda (sumbu Y, sumbu Z) untuk memastikan semua data dapat diambil.

4. Pengujian Pembacaan Status Channel (Read Channel Status Updates), dan lakukan pengujian untuk membaca status pembaruan channel:

```
GET
https://api.thingspeak.com/channels/2687137/status.json
```

Verifikasi bahwa status channel dan informasi terkini ditampilkan dengan benar, seperti waktu pembaruan terakhir atau informasi status sistem.

5. Pengujian Ketahanan API dalam Kondisi Jaringan yang Tidak Stabil:
    - Simulasikan gangguan jaringan untuk menguji bagaimana API berperilaku saat koneksi terganggu. Verifikasi apakah sistem dapat memulihkan koneksi dan melanjutkan pengiriman atau pengambilan data setelah jaringan stabil kembali.
    - Uji apakah API menghasilkan pesan kesalahan yang sesuai jika ada masalah dalam proses pengiriman atau pembacaan data.
2. Pengujian Performa API dengan Banyak Data:
    - Uji sistem dengan mengirimkan dan membaca data dalam jumlah besar dari ThingSpeak untuk memverifikasi apakah API tetap responsif dan dapat menangani beban yang tinggi tanpa kehilangan data atau waktu respons yang terlalu lama.

Pengujian Whitebox:

Pengujian Whitebox berfokus pada detail implementasi dan logika internal terkait penggunaan JSON API untuk komunikasi antara ESP32 dan ThingSpeak. Ini termasuk memastikan bahwa setiap request dan response diproses dengan benar dalam sistem.

1. Pengujian Kode API Request pada ESP32:
    - Uji logika dalam kode ESP32 yang bertanggung jawab untuk melakukan request API ke ThingSpeak, khususnya saat menggunakan fungsi seperti WiFiClient atau HTTPClient untuk mengirim data.
    - Verifikasi bahwa API Key (seperti api_key=7NXT7EEKGI09J5HI) digunakan dengan benar untuk mengotorisasi pengiriman data.
2. Pengujian Format Data JSON:
    - Verifikasi bahwa data yang dikirim dalam format JSON telah disusun dengan benar, termasuk nilai dari setiap field (seperti Sumbu X, Y, Z dan status).
    - Uji apakah semua field yang dibutuhkan telah terisi dengan benar sebelum dikirim ke

ThingSpeak, dan pastikan tidak ada data kosong atau data rusak yang terkirim.

3. Pengujian Penanganan Response dari ThingSpeak:
    - Uji cara ESP32 memproses response dari server ThingSpeak setelah pengiriman data berhasil. Verifikasi apakah sistem menangkap dan mengolah kode status HTTP, seperti 200 OK atau error code, dan menampilkan pesan yang relevan jika terjadi kegagalan.
    - Pastikan bahwa jika server ThingSpeak mengembalikan kode error, perangkat dapat menangani error tersebut dengan benar, seperti melakukan retry atau menampilkan notifikasi kesalahan.
4. Pengujian Pengambilan Data JSON dari Server :
    - Uji logika pada aplikasi Android atau client lain yang bertanggung jawab untuk mengambil data dari ThingSpeak menggunakan GET request. Pastikan bahwa data JSON yang diterima diolah dengan benar oleh client, termasuk parsing data yang sesuai.

- Uji apakah aplikasi dapat menampilkan status (Danger/Warning/Normal) dan data gyroscope (Sumbu X, Y, Z) berdasarkan data JSON yang diterima dari ThingSpeak.

5. Pengujian Rate Limit dan Error Handling API:
    - Uji bagaimana sistem menangani rate limit dari ThingSpeak, yang biasanya membatasi jumlah request API dalam waktu tertentu. Verifikasi bahwa jika batas tercapai, sistem dapat menghentikan permintaan sementara atau memberikan pemberitahuan ke pengguna.
    - Verifikasi apakah ada mekanisme untuk menangani request timeout atau server overload yang mungkin terjadi ketika ThingSpeak tidak merespon dalam waktu yang diharapkan.
6. Pengujian Data Retention dan Backup:
    - Uji apakah data yang dikirim ke ThingSpeak dapat di-backup atau disimpan dalam database lokal (misalnya di ESP32) sebelum pengiriman ke server. Ini penting untuk memastikan bahwa data tidak hilang jika

ada gangguan jaringan sebelum pengiriman data selesai.

Dengan pengujian Blackbox dan Whitebox ini, semua aspek dari API ThingSpeak dan komunikasi data antara ESP32 dan database IoT dapat dipastikan berfungsi dengan baik. Aktivitas pengujian ini akan memastikan bahwa data dapat dikirimkan dan diambil dari ThingSpeak secara akurat, serta bahwa sistem dapat menangani kesalahan dan gangguan dalam proses komunikasi data.

BAGIAN 6

Kesimpulan, Saran, Whats'Next

6.1 Kesimpulan

Pengembangan sistem pemantauan mitigasi bencana tanah longsor berbasis IoT menggunakan ESP32 sebagai mikrokontroler untuk membaca dan mengakuisisi data dari beberapa sensor gyroscope telah memberikan solusi inovatif dalam mendeteksi potensi bencana secara real-time. Sistem ini mampu melakukan pengukuran kemiringan tanah menggunakan sensor gyroscope, menghitung rata-rata (average) dari data sensor, dan mengirimkan data tersebut ke platform ThingSpeak untuk pemantauan lebih lanjut.

Sistem ini juga memanfaatkan protokol komunikasi I2C untuk integrasi beberapa komponen seperti LCD 16x2 untuk menampilkan informasi lokal secara langsung dan sensor gyroscope untuk mendeteksi perubahan kemiringan tanah. Data yang diperoleh dari sensor dikirim ke server IoT melalui penggunaan JSON API, sehingga dapat diakses dan dipantau oleh pengguna melalui aplikasi Android. Aplikasi ini memberikan informasi status yang

relevan, seperti Danger, Warning, dan Normal, berdasarkan ambang batas data gyroscope yang telah ditentukan sebelumnya.Keunggulan sistem ini meliputi:

1. Pemantauan Real-Time, Data kemiringan tanah dapat dipantau secara real-time dari lokasi jarak jauh menggunakan aplikasi Android, sehingga memungkinkan pengambilan keputusan yang cepat dan tepat.
2. Efisiensi Komunikasi Data, Dengan menggunakan platform ThingSpeak dan JSON API, data dapat diolah dan dikirim dengan efisien ke server IoT untuk analisis lebih lanjut.
3. Integrasi yang Fleksibe, Penggunaan protokol I2C memungkinkan sistem untuk menghubungkan berbagai sensor dan perangkat secara efisien dengan hanya menggunakan dua jalur komunikasi (SDA dan SCL), sehingga memudahkan pengembangan sistem.
4. Peringatan Dini, Sistem memberikan status deteksi tanah longsor secara otomatis berdasarkan nilai rata-rata dari sensor gyroscope, membantu masyarakat dan otoritas terkait dalam memitigasi risiko sebelum terjadi bencana yang lebih besar.

Dengan demikian, implementasi sistem IoT ini menawarkan pendekatan yang efektif dan efisien dalam upaya mitigasi bencana tanah longsor, terutama di daerah rawan longsor. Sistem ini dapat dikembangkan lebih lanjut dengan menambahkan sensor lain, memperluas cakupan pemantauan, serta menyempurnakan analisis data untuk meningkatkan akurasi dan ketepatan dalam deteksi bencana.

6.2 Saran

Untuk pengembangan ke depannya, ada beberapa saran yang dapat diterapkan dalam sistem pemantauan mitigasi bencana tanah longsor berbasis IoT ini. Pertama, penting untuk meningkatkan akurasi dengan menambahkan sensor-sensor lain yang relevan seperti sensor kelembaban tanah, sensor getaran seismik, atau sensor curah hujan. Dengan data tambahan dari berbagai sensor tersebut, sistem akan mampu memberikan peringatan dini yang lebih tepat mengenai potensi longsor.

Selain itu, algoritma prediksi yang lebih canggih dapat diintegrasikan ke dalam sistem. Penggunaan machine learning atau kecerdasan buatan (AI) untuk menganalisis pola data jangka panjang dapat membantu memprediksi terjadinya bencana berdasarkan data historis, sehingga

deteksi tidak hanya dilakukan berdasarkan kondisi saat ini tetapi juga memprediksi kemungkinan longsor di masa mendatang.

Sistem peringatan berlapis juga akan sangat bermanfaat. Pengembangan notifikasi yang mencakup SMS, email, atau bahkan alarm sirene lokal dapat mempercepat penyebaran informasi peringatan kepada masyarakat yang tinggal di wilayah rawan longsor. Hal ini akan membantu mereka mempersiapkan diri dengan lebih baik dan merespons potensi bencana secara cepat.

Pengembangan aplikasi mobile untuk monitoring juga perlu ditingkatkan. Aplikasi Android dapat ditambah dengan fitur visualisasi data yang lebih baik, seperti grafik tren kemiringan tanah atau peta interaktif yang menunjukkan lokasi sensor dan tingkat risiko. Selain itu, fitur log histori data dapat memberikan gambaran yang lebih lengkap kepada pengguna mengenai kondisi tanah dalam jangka waktu tertentu.

Karena di daerah terpencil sering terjadi masalah dengan koneksi internet, penggunaan teknologi komunikasi alternatif seperti LoRa atau mesh network dapat memastikan bahwa data tetap terkirim ke server

ThingSpeak meskipun terjadi gangguan pada jaringan internet utama. Hal ini akan memperkuat keandalan sistem dalam situasi kritis.

Sumber daya cadangan juga menjadi faktor penting untuk dipertimbangkan. Untuk memastikan sistem tetap berjalan selama keadaan darurat seperti hujan deras atau pemadaman listrik, penggunaan baterai atau panel surya dapat menjadi solusi agar sistem terus beroperasi tanpa gangguan.

Kerja sama dengan pihak pemerintah dan otoritas lokal juga diperlukan untuk meningkatkan efektivitas sistem ini. Dengan adanya integrasi antara data yang dikumpulkan oleh sistem IoT ini dengan jaringan pemantauan bencana nasional, keputusan tanggap darurat dapat diambil lebih cepat dan lebih tepat.

Selain itu, penambahan kemampuan analisis data berbasis cloud dapat mempercepat pengolahan data dan penyediaan informasi yang lebih komprehensif. Dengan menggunakan platform seperti Google Cloud atau AWS IoT, data dalam jumlah besar dapat dianalisis secara cepat dan hasilnya dapat diakses melalui dashboard yang lebih detail.

Langkah terakhir yang penting untuk diambil adalah melakukan uji coba sistem ini di berbagai wilayah dengan karakteristik tanah yang berbeda. Dengan begitu, sistem bisa memastikan bahwa sistem ini dapat berfungsi dengan baik di berbagai kondisi lingkungan, sehingga dapat memberikan deteksi yang andal di banyak lokasi rawan longsor.

Dengan berbagai pengembangan ini, sistem pemantauan mitigasi bencana tanah longsor akan semakin andal dan efektif dalam memberikan peringatan dini yang lebih akurat. Sistem ini dapat membantu upaya mitigasi bencana yang lebih baik, serta melindungi masyarakat dari risiko tanah longsor di masa yang akan datang.

6.3 What's Next

Dalam pengembangan aplikasi pemantauan mitigasi bencana tanah longsor berbasis IoT ini, ada banyak opportunity dan ide novelty yang bisa diimplementasikan untuk memperkuat daya tarik dan potensi komersial aplikasi ini ke depannya. Dengan berbagai perkembangan teknologi dan kebutuhan mitigasi bencana yang terus meningkat, berikut adalah beberapa peluang dan inovasi yang bisa menjadi keunggulan dan nilai jual utama:

1. Integrasi dengan Kecerdasan Buatan (AI)

Salah satu peluang terbesar adalah integrasi sistem ini dengan AI dan machine learning untuk analisis data yang lebih dalam. Dengan menggunakan pembelajaran mesin, aplikasi ini dapat belajar dari data historis mengenai tanah longsor, pola cuaca, dan perubahan kemiringan tanah. Ini akan memberikan kemampuan prediktif yang sangat kuat, sehingga aplikasi tidak hanya mendeteksi kondisi berbahaya saat ini, tetapi juga memprediksi potensi longsor di masa depan berdasarkan pola yang teridentifikasi. Novelty ini akan memberikan nilai tambah yang sangat besar, terutama bagi pemerintah atau organisasi yang bertanggung jawab atas mitigasi bencana.

2. Ekosistem IoT yang Terkoneksi secara Luas

Membuat ekosistem IoT yang lebih luas adalah langkah penting. Sistem pemantauan ini bisa dihubungkan dengan berbagai sensor di berbagai wilayah rawan bencana, menciptakan jaringan deteksi yang masif. Penggunaan teknologi mesh network atau LoRaWAN akan memungkinkan pengiriman data dari lokasi terpencil tanpa bergantung pada koneksi internet konvensional. Dengan jaringan luas ini, aplikasi dapat digunakan oleh pemerintah,

perusahaan tambang, hingga lembaga penyelamat untuk pemantauan secara real-time di seluruh daerah yang rentan. Peluang ini bisa membuka pasar baru di sektor infrastruktur, pertambangan, dan pengelolaan lahan yang rawan longsor.

3. Integrasi Sistem Penanganan Bencana Nasional

Salah satu peluang emas untuk aplikasi ini adalah menjalin kerja sama dengan pemerintah atau Badan Penanggulangan Bencana (BNPB di Indonesia) atau lembaga serupa di negara lain. Aplikasi ini dapat diintegrasikan langsung dengan sistem nasional untuk memberikan peringatan dini kepada masyarakat. Novelty di sini adalah kemampuan aplikasi untuk terhubung dengan sistem pemerintah, baik melalui dashboard monitoring yang dapat diakses oleh otoritas lokal maupun melalui layanan publik seperti SMS peringatan. Hal ini akan memberikan nilai tambah yang besar karena memperkuat kapabilitas pemerintah dalam menanggulangi bencana.

4. Fitur Pemantauan Multidimensi

Aplikasi dapat dikembangkan lebih lanjut dengan mengintegrasikan berbagai jenis sensor, seperti sensor

kelembaban tanah, seismometer, sensor getaran, dan sensor curah hujan. Dengan memantau berbagai parameter sekaligus, aplikasi ini dapat memberikan pemantauan multidimensi terhadap kondisi lingkungan. Ini akan menjadi keunggulan tersendiri dibandingkan dengan solusi lain yang hanya mengandalkan satu jenis sensor. Novelty ini membuat aplikasi mampu memberikan gambaran komprehensif terhadap risiko bencana alam, sehingga pengguna dapat memahami risiko secara lebih holistik.

5. Penggunaan Teknologi Blockchain

Penerapan blockchain untuk pengelolaan dan penyimpanan data IoT dapat meningkatkan kepercayaan dan keamanan aplikasi ini, terutama dalam konteks yang melibatkan berbagai lembaga publik dan swasta. Teknologi ini memungkinkan data sensor yang dikirimkan dari lapangan untuk terenkripsi dan tersimpan dengan aman, sehingga tidak dapat dimanipulasi. Novelty ini dapat memberikan jaminan keamanan bagi data yang dikirim dari sensor ke server IoT, sekaligus memberikan transparansi yang lebih baik kepada publik dan pemerintah.

6. Platform Crowdsourcing

Mengembangkan aplikasi dengan fitur crowdsourcing di mana masyarakat dapat berkontribusi dalam pemantauan kondisi tanah di lingkungan mereka juga bisa menjadi terobosan inovatif. Warga dapat melaporkan kondisi fisik tanah atau perubahan lingkungan yang mereka amati secara langsung ke sistem, yang kemudian divalidasi oleh data sensor. Hal ini dapat memperluas cakupan pemantauan tanpa harus mengandalkan hanya pada sensor, dan menciptakan ekosistem pemantauan yang lebih partisipatif. Novelty ini tidak hanya memperluas penggunaan aplikasi tetapi juga meningkatkan keterlibatan masyarakat.

7. Skalabilitas Aplikasi untuk Pasar Global

Karena tanah longsor merupakan masalah global yang tidak hanya terjadi di Indonesia, aplikasi ini memiliki potensi pasar internasional yang besar. Dengan mengembangkan versi internasional aplikasi, yang mendukung berbagai bahasa dan terhubung dengan platform global, aplikasi ini dapat diadopsi di berbagai negara yang rawan bencana seperti Jepang, India, dan negara-negara di Amerika Selatan. Peluang komersial dari skala global ini akan memberikan aplikasi ini peluang besar untuk pertumbuhan

bisnis, terutama dengan fokus pada kemitraan internasional dan adaptasi lokal.

8. Aplikasi dengan Augmented Reality (AR)

Untuk membuat aplikasi lebih menarik dan interaktif, fitur visualisasi real-time data dan augmented reality (AR) dapat ditambahkan. Pengguna dapat melihat data gyro dan kondisi kemiringan tanah dalam bentuk grafik atau bahkan simulasi visual 3D melalui teknologi AR. Ini akan menjadi nilai jual yang menarik, terutama bagi pengguna teknis seperti insinyur, ilmuwan lingkungan, dan tim tanggap bencana, yang membutuhkan pemahaman yang mendalam mengenai kondisi lapangan.

Dengan peluang dan inovasi ini, aplikasi pemantauan mitigasi bencana tanah longsor berbasis IoT ini tidak hanya akan menjadi solusi yang sangat menjual, tetapi juga dapat memberikan dampak besar dalam skala lokal dan global. Inovasi dalam teknologi prediksi, keamanan, partisipasi publik, dan penggunaan AI akan menjadikan aplikasi ini sebagai terobosan penting di bidang mitigasi bencana dan pemantauan lingkungan berbasis IoT.

DAFTAR PUSTAKA

Broto, P. E. (2023). *Sistem monitoring suhu dan kelembaban portable berbasis IoT menggunakan Arduino Mega dan ESP32*. Insypro: Jurnal Informatika Sistem dan Proses, 8(1). https://journal.uin-alauddin.ac.id/index.php/insypro/article/view/37466

Samosir, A. S., & Widodo, N. S. (2020). *Gyroscope and accelerometer sensor on the Lanange Jagad dance robot balance system*. Buletin Ilmiah Sarjana Teknik Elektro, 2(2). https://journal2.uad.ac.id/index.php/biste/article/view/922

Sari, R. S., Nuryanto, N., & Widiyanto, A. (2021). *Implementasi IoT pada sistem monitoring suhu dan kelembaban ruangan berbasis web*. Jurnal Teknik Elektro dan Komputer, 7(3). https://repository.teknokrat.ac.id/5561/5/skripsi19316011.pdf

Apri, J. (2015). *Monitoring suhu dan kelembaban berbasis Internet of Things (IoT)*. Jurnal Ilmiah Teknologi Informasi Terapan (JITTER), 1(3). https://jurnal.unprimdn.ac.id/index.php/JUTIKOMP/article/view/1676

Supriyadi, E., & Ariman. (2025). *Sistem deteksi gempa berbasis IoT dengan visualisasi real-time dan notifikasi cerdas*. Jurnal INTI Nusa Mandiri, 19(2), 288–294.

Widodo, N. S., & Samosir, A. S. (2020). *Penggunaan sensor gyroscope dan accelerometer untuk kendali keseimbangan robot humanoid*. Buletin Ilmiah Sarjana Teknik Elektro, 2(2). https://journal2.uad.ac.id/index.php/biste/article/view/922

Ruliyanta, R., Marhaeni, M., & Arrang, H. (2022). *Review of the access network selection algorithms on Wi-Fi offloading in LTE networks*. Journal of Advanced Computing Technology and Application (JACTA), 1–12.

Prasetyo, D. R., & Marhaeni, M. (2022). *Design and build motor parts sales application at Calvin Jaya Motor Workshop Pengasinan Sawangan*. Jurnal Rekayasa Informasi, 11(2), 127–133.

TENTANG PENULIS-1

Ariman adalah dosen tetap di Program Studi Teknik Elektro, Institut Sains dan Teknologi Nasional (ISTN) Jakarta, dengan keahlian di bidang elektronika, sistem tertanam, Internet of Things (IoT), dan komunikasi data. Ia aktif dalam riset dan pengembangan teknologi terapan, khususnya pada sistem monitoring, automasi berbasis mikrokontroler, dan integrasi perangkat berbasis sensor untuk kebutuhan industri dan masyarakat.

Selain mengajar dan membimbing mahasiswa, Ariman juga terlibat dalam berbagai proyek sebagai tenaga ahli serta rutin mempublikasikan hasil penelitian di jurnal nasional terakreditasi. Beberapa fokus risetnya mencakup deteksi gempa berbasis IoT, sistem monitoring kesehatan, serta pengembangan teknologi hemat energi dan infrastruktur digital. Ia juga berperan aktif dalam pelatihan dan seminar di bidang teknik elektro dan inovasi teknologi.

Untuk kolaborasi akademik dan profesional, penulis dapat dihubungi di ariman@istn.ac.id.

TENTANG PENULIS–2

Riadi Marta Dinata adalah dosen tetap di Program Studi Teknik Informasi, Fakultas Sains dan Teknologi Informasi, ISTN Jakarta. Ia menyelesaikan pendidikan di bidang Elektronika, Teknik Informatika, dan Ilmu Komputer (S2), serta kini tengah menempuh studi doktoral di Universitas Lampung (UNILA) dengan fokus riset pada sampling data, kecerdasan buatan, dan sistem informasi.

Sejak awal 2000-an, ia aktif di dunia teknologi informasi, dengan pengalaman dalam pemrograman, jaringan komputer, sistem tertanam, dan AI terapan. Riadi juga mendirikan dan mengajar di komunitas *StartUp IT Lp2maray (From Zero to Hero)* sejak 2001, yang fokus membina talenta digital dari dasar.

Selain mengajar dan membimbing mahasiswa, ia terlibat dalam pelatihan, workshop, serta seminar IT. Karya risetnya mencakup quick count sampling, sistem pakar, IoT, data mining, dan geospasial. Ia juga menulis buku *Data Mining: Memahami Pola di Balik Angka*, yang diapresiasi oleh akademisi dan praktisi.

Kontak: adiarray@istn.ac.id

TENTANG PENULIS–3

Muhammad Ikrar Yamin adalah dosen dan peneliti di Institut Sains dan Teknologi Nasional (ISTN) Jakarta, dengan keahlian di bidang robotika, otomasi, telekomunikasi, IoT, dan elektronika. Ia aktif mengembangkan sistem komunikasi cerdas, perangkat pintar, serta jaringan kontrol robotik untuk kebutuhan kebencanaan, industri, dan otomasi modern.

Salah satu karyanya yang menonjol adalah riset tentang evaluasi protokol komunikasi MQTT dan COAP dalam sistem robot penanggulangan bencana. Ia juga terlibat dalam pengembangan sistem kontrol drone berbasis PID, sistem absensi QR code lokal, integrasi WebGIS untuk pemasaran, serta optimalisasi jaringan UMTS menggunakan backup link dan BFD.Dengan pendekatan interdisipliner dan basis teknik yang kuat, ia berkontribusi pada pengembangan solusi teknologi yang aplikatif bagi masyarakat dan industri.

Kontak: ikrar@istn.ac.id