BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kerja magang di Badan Meteorologi, Klimatologi, dan Geofisika (BMKG), kegiatan dilakukan di Bidang Instrumentasi pada Direktorat Gempa Bumi dan Tsunami dengan bimbingan dari Supervisor Lapangan. Aktivitas yang dilakukan meliputi pembelajaran dan pelaksanaan instrumentasi, akuisisi, serta pengolahan data sensor seismik, termasuk perancangan rangkaian sederhana, pengujian sensor, dan integrasi data ke dalam sistem pemantauan real-time. Koordinasi rutin dengan supervisor dan staf teknis dilakukan untuk memastikan setiap kegiatan berjalan sesuai rencana dan pemahaman praktis mengenai sistem monitoring gempa bumi serta tsunami diperoleh.

3.2 Tugas Kerja Magang

Selama proses kerja magang di BMKG, terdapat kegiatan dan tugas yang dilakukan. Kegiatan dan tugas tersebut yaitu dapat dilihat pada Tabel 3.1 berikut.

Tabel 3.1 Waktu Pelaksanaan Magang Perusahaan

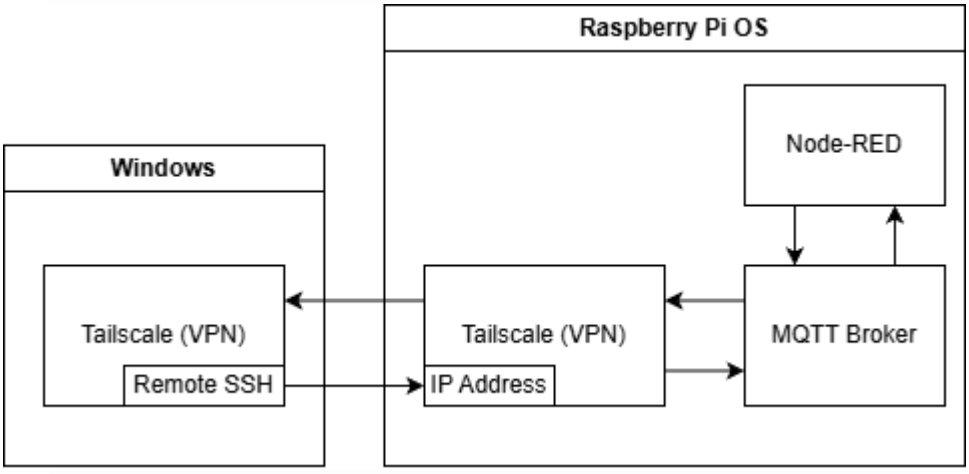
Bulan	Kegiatan
Juli	Perkenalan dengan pihak DGT serta rekan magang dari kampus lain Mempelajari SeiscomP Mempresentasikan rencana pembuatan proyek
Agustus	Mempelajari Digitizer sensor Trillium Compact Mempelajari Digitizer sensor Lennartz 3D Lite Mk II Percobaan ADS1115 dengan sensor Geophone Percobaan ADS1115 dengan Accelerometer
September	Percobaan Penggunaan API untuk mengirim notifikasi ke WhatsApp Percobaan ADS1256 dengan Sensor Geophone

	Percobaan ADS1256 dengan Accelerometer GY-61
Oktober	Mempelajari cara kerja Remote SSH Mempelajari cara kerja Tailscale Pembuatan Dashboard Node-RED

Proyek yang dilaksanakan selama kegiatan magang adalah perancangan sistem seismograf berbasis Raspberry Pi 4B yang memanfaatkan sensor GY-61 dan sensor geophone, dengan modul ADS1256 sebagai komponen pendukung untuk akuisisi data beresolusi tinggi. Sistem ini berfungsi untuk mendeteksi, merekam, dan memvisualisasikan getaran tanah secara *real-time*, sehingga dapat digunakan sebagai alat pemantauan aktivitas seismik dalam skala laboratorium maupun lapangan.

3.3 Uraian Kerja Magang

Pada proyek yang dilaksanakan selama kegiatan magang, kegiatan difokuskan pada perancangan dan pengembangan dashboard pemantauan data sensor secara *real-time*, yang berfungsi untuk merepresentasikan hasil akuisisi getaran tanah dari sistem seismograf secara visual dan interaktif.



Gambar 3.1 Diagram Blok Sistem

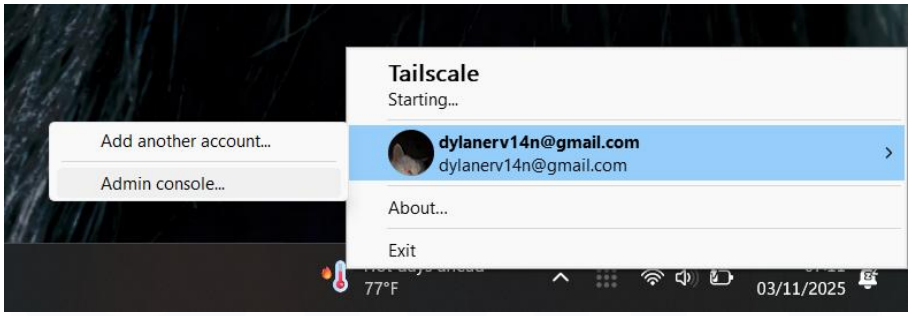
Pada Gambar 3.1 ditunjukkan diagram blok sistem yang menggambarkan bagaimana Raspberry Pi dengan sistem operasi Raspberry Pi OS dapat dikendalikan dari laptop berbasis Windows secara jarak jauh. Koneksi antara kedua perangkat difasilitasi oleh Tailscale sebagai jaringan VPN, di mana Tailscale memberikan alamat

IP virtual (Tailscale IP) untuk masing-masing perangkat sehingga keduanya dapat saling terhubung meskipun berada di jaringan yang berbeda. Melalui alamat IP tersebut, laptop Windows dapat mengakses Raspberry Pi menggunakan protokol Remote SSH (Secure Shell) untuk menjalankan perintah, memantau proses, serta melakukan konfigurasi sistem secara langsung.

Selain fungsi kendali jarak jauh, Raspberry Pi juga menjalankan Node-RED dan MQTT Broker untuk pengelolaan serta pemantauan data sensor. MQTT Broker berperan sebagai pusat komunikasi data, di mana sensor mengirimkan data pembacaan ke broker agar dapat diteruskan ke Node-RED. Node-RED kemudian memproses data tersebut dan menampilkannya pada dashboard secara real-time sehingga pengguna dapat memantau kondisi sensor dengan mudah. Dengan integrasi antara Node-RED dan MQTT Broker.

3.3.1 Konfigurasi Koneksi Jaringan Menggunakan Tailscale

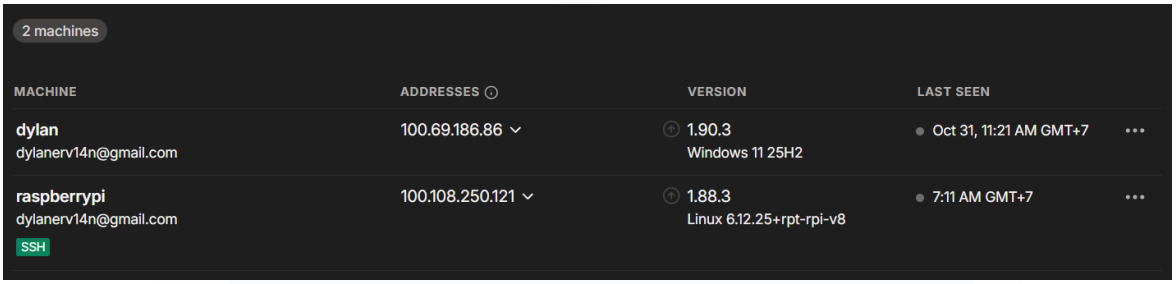
Setelah proses instalasi Tailscale pada laptop selesai dan perangkat berhasil terhubung dengan akun yang terdaftar, ikon Tailscale akan muncul *taskbar* sebagaimana ditunjukkan pada Gambar 3.2



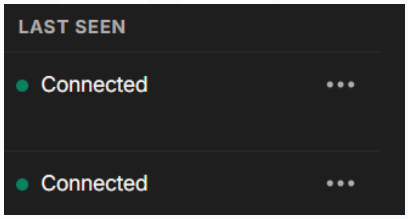
Gambar 3.2 Tampilan Taskbar Tailscale

Pada Gambar 3.2 ditampilkan beberapa opsi yang muncul ketika ikon Tailscale diklik kanan. Selanjutnya, ketika kursor diarahkan (*hover*) pada bagian akun, akan muncul opsi *Admin Console* yang kemudian dapat dipilih untuk membuka halaman pengaturan.

A



B



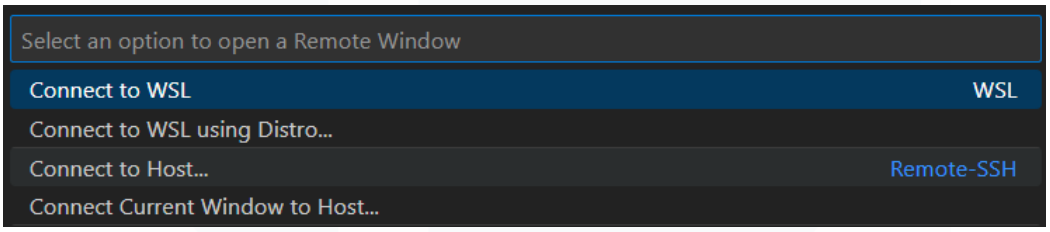
Gambar 3.3 A. Tampilan *Admin Console,* B. Tampilan Kolom *Last Seen*

Pada Gambar 3.3 (A) ditampilkan daftar machine yang terhubung ke akun, yaitu perangkat "dylan" dengan sistem operasi Windows dan perangkat "raspberrypi" dengan sistem operasi Raspberry Pi OS. Pada Gambar 3.2 (B) Keduanya menunjukkan status *Connected* pada kolom *Last Seen* di Admin Console, menandakan bahwa masing-masing perangkat sudah aktif dan berhasil tersambung ke jaringan VPN Tailscale. Kondisi ini menunjukkan bahwa kedua perangkat telah berada dalam satu jaringan dan siap berkomunikasi, meskipun koneksi langsung antarperangkat baru akan terjadi ketika salah satu perangkat mengakses alamat IP Tailscale milik perangkat lainnya.

Dalam proyek ini, perangkat lunak tambahan berupa *Visual Studio Code* digunakan untuk mempermudah proses pengembangan program. Melalui *Visual Studio Code* (VS Code), koneksi SSH dapat dilakukan secara langsung dari aplikasi tersebut.

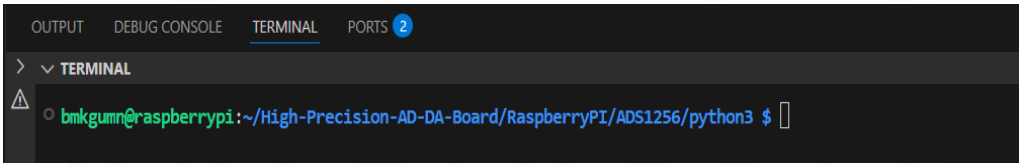
A

B



Gambar 3.4 A. Tampilan ikon *Remote Window* pada VS Code, B. Tampilan opsi setelah ikon *Remote WIndow* di klik

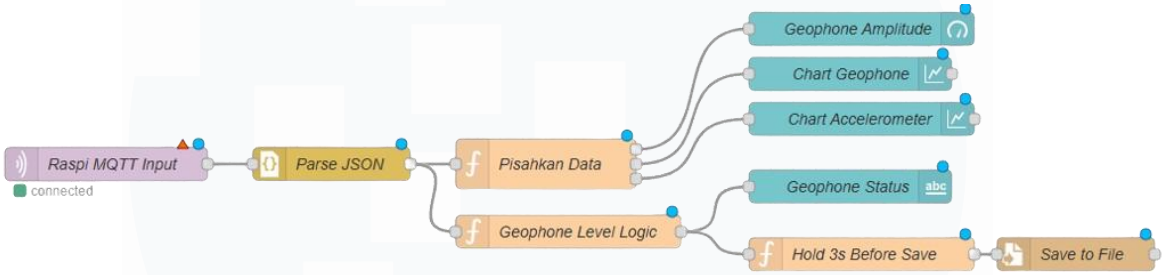
Pada Gambar 3.4 (A) ditampilkan ikon *Remote Window* yang berada di pojok kiri bawah tampilan VS Code. Setelah ikon tersebut diklik, muncul beberapa opsi di bagian atas jendela, seperti terlihat pada gambar 3.4 (B). Pada Gambar 3.4 (B) ditunjukkan opsi *Connect to Host... (Remote-SSH)*. Setelah opsi ini dipilih, terdapat perintah untuk memasukkan "ssh nama@ip_address". Ketika perintah tersebut dijalankan, terminal di VS Code akan menggunakan username yang ada pada Raspberry Pi, seperti terlihat pada Gambar 3.5.



Gambar 3. 5 Tampilan Terminal VS Code setelah koneksi SSH berhasil

Pada Gambar 3.5 terlihat tampilan terminal pada Visual Studio Code dengan keterangan "bmkgumn@raspberrypi" beserta *file path* tempat program tersimpan. Hal ini menunjukkan bahwa Raspberry Pi telah berhasil terhubung dan dapat dikendalikan secara langsung melalui perangkat Windows

3.3.2 Struktur Alur Node-RED



Gambar 3.6 Diagram *Flow* Sistem Monitoring Getaran pada Node-RED

Gambar 3.6 menunjukkan flow Node-RED yang menggambarkan alur pemrosesan data getaran dari Raspberry Pi. Data dari sensor geophone dan akselerometer dikirim melalui MQTT dan diterima oleh node "Raspi MQTT Input", kemudian diubah menjadi format objek oleh "*Parse JSON*". Data yang sudah diparsing diproses oleh dua node utama: "Pisahkan Data", yang memisahkan nilai amplitudo serta percepatan sumbu X, Y, dan Z untuk ditampilkan melalui "*Geophone Amplitude*", "*Chart Geophone*", dan "*Chart Accelerometer*"; serta "*Geophone Level Logic*", yang menganalisis tingkat getaran dan menampilkan status secara real-time melalui "*Geophone Status*". Setelah pemrosesan pada Geophone Level Logic, sistem melakukan penyimpanan data ketika indikator berada pada level *waspada* atau *bahaya* selama tiga detik.


```
{
    "id": "68544cc2784fcbee",
    "type": "function",
    "z": "269e9ae46e1b011f",
    "name": "Pisahkan Data",
    "func": "let d = msg.payload;\nif (!d) return null;\n\n// Output 1: Gauge
        Geophone Amplitude\nlet geoAmp = { payload: d.geo_amp };\n\n//
```

```
            Output 2: Chart Geophone Signal\nlet geoChart = { topic:
            'Geophone', payload: d.geo_amp };\n\n// Output 3: Chart
            Accelerometer (multi-axis)\nlet ax = { topic: 'Accel_X', payload:
            d.accel_x };\nlet ay = { topic: 'Accel_Y', payload: d.accel_y };\nlet
            az = { topic: 'Accel_Z', payload: d.accel_z };\n\nreturn [geoAmp,
            geoChart, [ax, ay, az]];",
        "outputs": 3,
        "noerr": 0,
        "x": 640,
        "y": 260,
        "wires": [
          ["467a90878fd29109"],["121317b8ef6caa45"], ["9262a6094b90c80b"]
          ]
}
```


Kode JSON Flow 1. Node Pisahkan Data

Kode JSON Flow 1 adalah kode untuk node “Pisahkan Data” dan berfungsi memproses data getaran yang diterima melalui msg.payload. Node ini memeriksa apakah data tersedia, kemudian memisahkannya menjadi tiga output. Output pertama mengirim nilai amplitudo geophone (geo_amp) untuk ditampilkan pada *gauge*. Output kedua mengirim nilai amplitudo yang sama dengan topik “Geophone” untuk keperluan grafik sinyal. Output ketiga menghasilkan tiga objek terpisah berisi data percepatan akselerometer pada sumbu X, Y, dan Z (accel_x, accel_y, accel_z) dengan topik masing-masing. Ketiga Output ini dialirkan ke node-node visualisasi yang berbeda sesuai dengan struktur flow yang telah dirancang.


```
{
        "id": "467a90878fd29109",
        "type": "ui_gauge",
        "z": "269e9ae46e1b011f",
        "name": "Geophone Amplitude",
        "group": "ui_group_geo",
        "order": 1,
        "width": 6,
        "height": 6,
        "gtype": "gage",
        "title": "Geophone Amplitude",
```

```
    "label": "Amp",
    "format": "{{value}}",
    "min": 0,
    "max": "10000",
    "colors": ["#00b500","#e6e600","#ca3838"],
    "x": 940,
    "y": 180,
    "wires": []
}
```


Kode JSON Flow 2. Node *Geophone Amplitude*

Kode JSON Flow 2 adalah kode untuk node "*Geophone Amplitude*" pada kode tersebut adalah sebuah *ui_gauge* yang digunakan untuk menampilkan nilai amplitudo geophone secara real-time pada dashboard Node-RED. Gauge ini berada dalam grup antarmuka bernama ui_group_geo, memiliki ukuran 6 x 6, dan menampilkan nilai dengan label "*Amp*". Batas bawah gauge ditetapkan pada 0 dan batas atas pada 10.000, dengan skema warna hijau–kuning–merah untuk menunjukkan rentang aman hingga berbahaya. Node ini tidak memiliki keluaran karena fungsinya hanya untuk menampilkan data yang diterima dari node sebelumnya.


```
{
    "id": "121317b8ef6caa45",
    "type": "ui_chart",
    "z": "269e9ae46e1b011f",
    "name": "Chart Geophone",
    "group": "ui_group_geo",
    "order": 2,
    "width": "12",
    "height": "6",
    "label": "Geophone Amplitude",
    "chartType": "line",
    "legend": "false",
    "xformat": "HH:mm:ss",
    "interpolate": "linear",
    "nodata": "No data",
    "dot": true,
    "ymin": "0",
    "ymax": "50000",
    "removeOlder": "60",
```

```
    "removeOlderPoints": "",
    "removeOlderUnit": "1",
    "cutout": "",
    "useOneColor": true,
    "useUTC": false,
    "colors": ["#1f77b4","#000000","#000000","#000000","#000000",
               "#000000","#000000","#000000","#000000"],
    "outputs": 1,
    "useDifferentColor": false,
    "className": "",
    "x": 940,
    "y": 240,
    "wires": [
      [ ]
    ]
}
```


Kode JSON Flow 3. Node *Chart Geophone*

Kode JSON Flow 3 adalah kode untuk node "*Chart Geophone*" yang merupakan *ui_chart* yang digunakan untuk menampilkan grafik garis (*line chart*) dari amplitudo geophone secara real-time pada dashboard Node-RED. Grafik ini ditempatkan di grup ui_group_geo dengan lebar 12 grid dan tinggi 6 grid. Label grafik ditentukan sebagai "*Geophone Amplitude*", dengan sumbu Y dibatasi pada rentang 0 hingga 50.000. Format waktu pada sumbu X menggunakan jam-menit-detik, sedangkan titik-titik data dihubungkan secara linear. Grafik hanya menampilkan satu warna garis dan akan menghapus data yang lebih lama dari 60 detik untuk menjaga tampilan tetap ringan. Node ini tidak meneruskan data lebih lanjut karena berfungsi sebagai komponen visualisasi.


```
{
    "id": "9262a6094b90c80b",
    "type": "ui_chart",
    "z": "269e9ae46e1b011f",
    "name": "Chart Accelerometer",
    "group": "ui_group_accel",
    "order": 3,
    "width": 12,
    "height": 6,
    "label": "Accelerometer X/Y/Z",
    "chartType": "line",
```

```
        "legend": "true",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "No data",
        "dot": true,
        "ymin": "-100000",
        "ymax": "100000",
        "removeOlder": "60",
        "removeOlderPoints": "",
        "removeOlderUnit": "1",
        "cutout": "",
        "useOneColor": false,
        "useUTC": false,
        "colors": ["#ff7f0e","#2ca02c","#d62728","#000000","#000000","#000000",
                   "#000000","#000000","#000000"],
        "outputs": 1,
        "useDifferentColor": false,
        "className": "",
        "x": 940,
        "y": 300,
        "wires": [
            [ ]
        ]
}
```


Kode JSON Flow 4. Node *Chart Accelerometer*

Kode JSON Flow 4 adalah kode untuk node "*Chart Accelerometer*" yang merupakan *ui_chart* yang digunakan untuk menampilkan grafik percepatan akselerometer pada tiga sumbu X, Y, dan Z secara real-time di dashboard Node-RED. Grafik ini ditempatkan dalam grup ui_group_accel, dengan ukuran 12 grid lebar dan 6 grid tinggi. Label grafik ditampilkan sebagai "*Accelerometer X/Y/Z*", menggunakan jenis grafik garis yang dapat dikenali melalui warna berbeda. Rentang sumbu Y ditetapkan dari -100.000 hingga 100.000 untuk mengakomodasi variasi percepatan dari ketiga sumbu. Format waktu pada sumbu X menggunakan jam-menit-detik, dan data lama akan dibersihkan setelah 60 detik agar tampilan tetap responsif. Node ini tidak meneruskan data lebih lanjut karena berfungsi sebagai komponen visualisasi.

```
{
    "id": "0aa682e8bf693742",
    "type": "function",
    "z": "269e9ae46e1b011f",
    "name": "Geophone Level Logic",
    "func": "let amp = msg.payload.geo_amp;\n\nif (amp < 5000)
        {\n    msg.color = 'green';\n    msg.status = 'Normal';\n} else if (amp
        < 15000) {\n    msg.color = 'yellow';\n    msg.status = 'Waspada';\n}
        else {\n    msg.color = 'red';\n    msg.status =
        'Bahaya';\n}\n\nmsg.payload = {\n    amp: amp,\n    status:
        msg.status\n};\n\nreturn msg;",
    "outputs": 1,
    "timeout": "",
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 660,
    "y": 400,
    "wires": [
      ["df554866ee685cc9","hold3s_func"]
    ]
}
```


Kode JSON Flow 5. Node *Geophone Level Logic*

Kode JSON Flow 5 merupakan kode untuk node “Geophone Level Logic”, yaitu sebuah *function node* yang digunakan untuk menentukan level kondisi getaran berdasarkan nilai amplitudo geophone. Node ini membaca geo_amp dari msg.payload, kemudian mengelompokkannya ke dalam tiga kategori, yaitu Normal jika amplitudo berada di bawah 5000 *ADC counts* Waspada jika amplitudo berada di bawah 15.000 *ADC counts*, dan Bahaya jika amplitudo melebihi nilai tersebut. Setiap kategori ditandai dengan warna indikator hijau, kuning, atau merah yang mencerminkan tingkat getaran. Selanjutnya, node ini membentuk kembali msg.payload yang berisi nilai amplitudo dan status level getaran, sebelum dikirimkan ke node berikutnya untuk ditampilkan atau diproses lebih lanjut.

Nilai yang dihasilkan oleh geophone pada tahap ini masih berupa data mentah (*ADC counts*), sehingga diperlukan proses konversi agar diperoleh satuan yang valid untuk merepresentasikan getaran tanah. Konversi pertama dilakukan

dengan mengubah nilai ADC menjadi tegangan (V) menggunakan persamaan berikut:

$$V_{geo}(V) = \frac{ADC\ Count}{2^{23} - 1} \times V_{ref} \times 1000$$

Keterangan:

ADC 24-bit bipolar : $2^{23} - 1$

Raw Data : ADC Count

ADS1256 V_{ref}: 2.5V

Setelah diperoleh nilai tegangan, data tersebut kemudian dikonversi kembali ke dalam satuan kecepatan getaran tanah (mm/s) menggunakan persamaan berikut:

$$v\ (mm/s) = \frac{V_{geo}}{28.8}$$

Keretangan:

Sensitifitas Geophone : 28.8 mV/(mm/s)

Melalui dua tahap konversi tersebut, keluaran geophone tidak lagi berupa data mentah, melainkan telah dinyatakan dalam satuan fisik yang valid sehingga dapat digunakan untuk mendeteksi dan menganalisis getaran tanah.


```
{
    "id": "hold3s_func",
    "type": "function",
    "z": "269e9ae46e1b011f",
    "name": "Hold 3s Before Save",
    "func":"let amp= msg.payload.amp;\nlet status = msg.payload.status;\n\nconst
        now = Date.now();\nlet start = context.get('start') || null;\n\n// Reset
        jika normal\nif (status=== \"Normal\") {\n  context.set('start', null);\n
        return null;\n}\n\n// Jika baru masuk Waspada/Bahaya, mulai
        timer\nif (!start) {\n  context.set('start', now);\n  return null;\n}\n\n//
        Hitung durasi\nlet dur = now - start;\n\n// Jika lebih dari 3 detik,
        simpan\nif (dur >= 3000) {\n  let data = {\n    timestamp: new
        Date().toISOString(),\n    amplitude: amp,\n    status: status,\n
        duration_ms: dur\n  };\n\n  // Agar tidak spam log → reset timer
        setelah disimpan\n  context.set('start', null);\n\n  return { payload:
        JSON.stringify(data) + \"\\n\" };\n}\n\nreturn null;",
    "outputs": 1,
```

```
    "timeout": "",
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 900,
    "y": 340,
    "wires": [ ["file_log_node"]]
}
```


Kode JSON Flow 6. Node *Hold 3s Before Save*

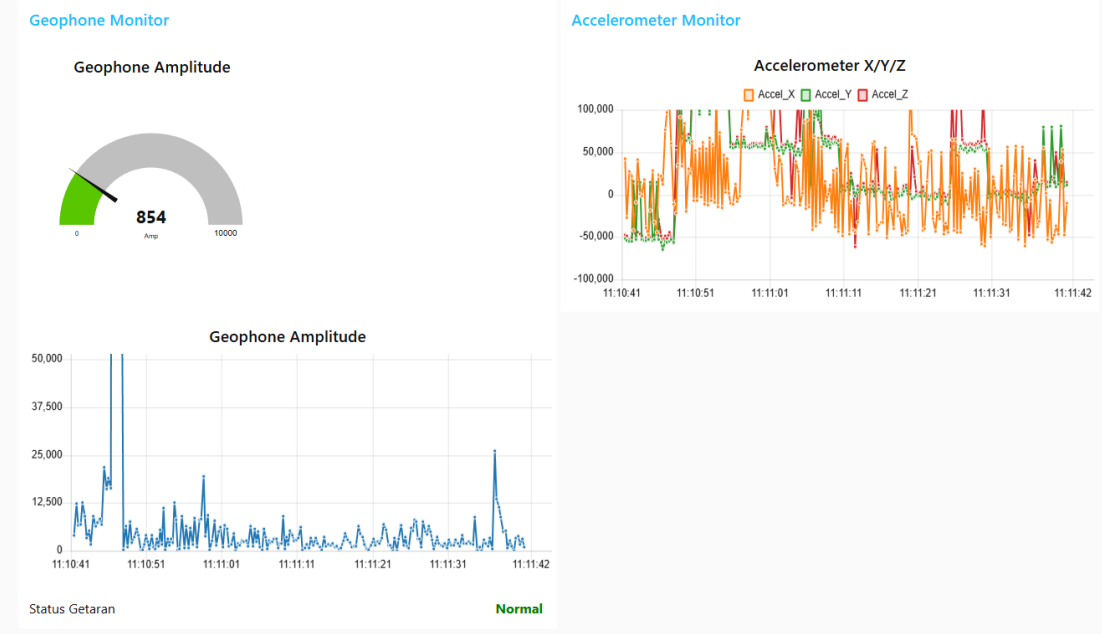
Kode JSON Flow 6 adalah kode untuk node "*Hold 3s Before Save*" yang merupakan *function node* yang bertugas menahan proses penyimpanan data selama tiga detik untuk memastikan kondisi *Waspada* atau *Bahaya* benar-benar stabil sebelum dicatat. Node ini membaca nilai amplitudo (amp) dan status level getaran dari pesan masuk, lalu menggunakan *context storage* untuk menyimpan waktu mulai ketika status pertama kali berubah menjadi Waspada atau Bahaya. Jika status kembali ke Normal, timer di-reset dan proses penyimpanan dibatalkan. Ketika status tetap berada pada Waspada atau Bahaya, node menghitung berapa lama kondisi tersebut bertahan. Jika durasinya mencapai tiga detik atau lebih, node membuat objek berisi timestamp, amplitudo, status, dan durasi lalu mengirimkannya sebagai payload untuk disimpan oleh node file. Setelah data disimpan, timer direset agar log tidak berulang-ulang tercatat dalam waktu singkat.


```
{
    "id": "file_log_node",
    "type": "file",
    "z": "269e9ae46e1b011f",
    "name": "Save to File",
    "filename":"/home/bmkgumn/High-Precision-AD-DA
                Board/RaspberryPI/ADS1256/python3/logs/geophone_log.txt
                ",
    "filenameType": "str",
    "appendNewline": true,
    "createDir": false,
    "overwriteFile": "false",
    "encoding": "none",
```

```
        "x": 1090,
        "y": 340,
        "wires": [ [ ] ]
}
```

Kode JSON Flow 7. Node *Save to File*

Kode JSON Flow 7 adalah kode untuk node "Save to File" yang merupakan *file node* yang berfungsi menyimpan data log ke sebuah berkas teks pada Raspberry Pi. Setiap pesan yang diterima akan ditambahkan ke file bernama geophone_log.txt. Node ini menggunakan mode *append*, sehingga data baru selalu ditambahkan ke baris berikutnya tanpa menimpa isi sebelumnya. Node tidak membuat folder baru secara otomatis karena opsi *createDir* diset ke *false*. Node ini menjadi tahap akhir dalam proses pencatatan sehingga tidak memiliki keluaran ke node lain.



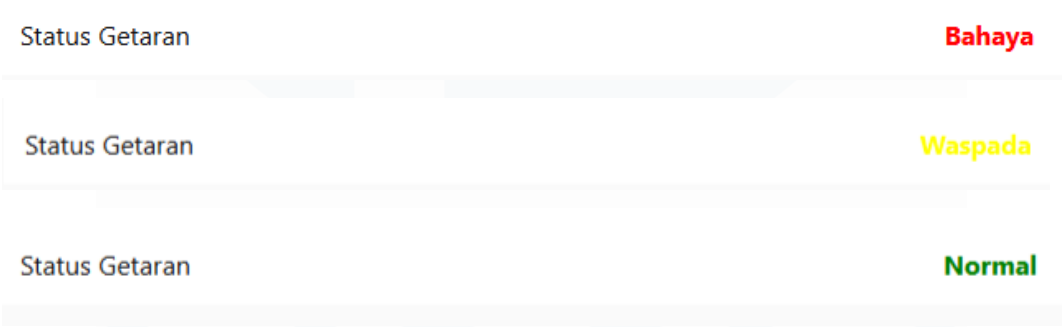
Gambar 3. 7 Tampilan Dashboard Pembacaan Sensor Secara *Real-Time*

Gambar 3.7 menampilkan Dashboard pembacaan sensor secara *real-time*. *Geophone Gauge* menunjukkan amplitudo getaran geophone, sementara grafik garis memvisualisasikan perubahan amplitudo tersebut terhadap waktu. Grafik

accelerometer menampilkan getaran pada sumbu X, Y, dan Z dengan warna berbeda untuk memudahkan identifikasi. Status getaran ditampilkan melalui teks berwarna yang berubah sesuai level normal, waspada, atau bahaya berdasarkan logika ambang batas pada node function. Jika status waspada atau bahaya bertahan selama 3 detik, sistem otomatis menyimpan data waktu dan detail terkait dalam file teks seperti pada gambar 3.8 berikut.


```
{"timestamp":"2025-11-26T23:28:30.425Z","amplitude":188034,"status":"Bahaya","duration_ms":3117}
{"timestamp":"2025-11-26T23:28:33.854Z","amplitude":18166,"status":"Bahaya","duration_ms":3153}
{"timestamp":"2025-11-26T23:34:16.110Z","amplitude":11564,"status":"Waspada","duration_ms":3046}
{"timestamp":"2025-11-26T23:34:23.857Z","amplitude":8976,"status":"Waspada","duration_ms":3112}
{"timestamp":"2025-11-26T23:34:27.270Z","amplitude":8209,"status":"Waspada","duration_ms":3137}
```


Gambar 3.8 detail data getaran pada file teks



Gambar 3. 9 Indikator Status Getaran berdasarkan perubahan warna

Gambar 3.9 menampilkan indikator status getaran yang berubah warna secara otomatis sesuai dengan amplitudo geophone. Nilai di bawah 5000 ADC counts (≈ 0,052 mm/s) ditandai sebagai Normal dengan indikator hijau, nilai 5000–14.999 ADC counts (≈ 0,052–0,156 mm/s) dikategorikan sebagai Waspada dengan indikator kuning, dan nilai di atas 15.000 ADC counts (> 0,156 mm/s) diklasifikasikan sebagai Bahaya dengan indikator merah. Indikator ini memudahkan pengguna dalam memantau kondisi getaran secara cepat melalui perubahan warna pada dashboard.

3.3.3 Akuisisi dan Monitoring Data PGA Real-Time Berbasis Node-RED

Pada bagian ini dibahas proses akuisisi data *Peak Ground Acceleration* (PGA) yang diperoleh dari sensor accelerometer pada modul MPU9250, mulai dari pembacaan sinyal, pengolahan awal, hingga pengiriman data secara *real-time*. Data PGA yang telah diproses kemudian dimonitor melalui website berbasis Node-RED sehingga dapat diakses secara langsung tanpa bergantung pada sistem *localhost*. Sistem ini dirancang untuk menampilkan nilai PGA secara kontinu dalam bentuk grafik dan data numerik guna memudahkan pemantauan kondisi getaran tanah, di mana integrasi antara perangkat akuisisi data, protokol komunikasi, dan antarmuka web memungkinkan informasi PGA disajikan secara *real-time* sebagai bagian dari sistem monitoring gempa.

Berikut Persamaan yang digunakan pada proses akuisisi data PGA:

1. Persamaan Kalibrasi Accelerometer (Menentukan *Offset*)

$$a'_{Axis} = a_{Axis} - a_{Axis\_off}$$

Keterangan:

a'_{Axis} : Percepatan sumbu (X, Y, atau Z) setelah koreksi offset (g)

a_{Axis} : Percepatan mentah sumbu (X, Y, atau Z), masih mengandung bias sensor, gravitasi, dan noise (g)

a_{Axis_off} : Offset ssumbu (X, Y, atau Z), nilai rata-rata dari a_{Axis} (g)

2. Resultan *Acceleration*

$$a_{tot} = \sqrt{{a'_x}^2 + {a'_y}^2 + {a'_z}^2}$$

Keterangan:

a_{tot} : Percepatan total (g)

a'_{Axis} : Percepatan sumbu (X, Y, atau Z) setelah koreksi offset (g)

3. *Peak Ground Acceleration* (PGA)

$$PGA = \max\left(a_{tot}(t)\right)$$

Keterangan:

PGA : Nilai percepatan maksimum (g)

$a_{tot}(t)$: Percepatan total sebagai fungsi waktu (1 detik)

4. Konversi ke satuan percepatan (m/s^2)

$$PGA\ (\ m/s^2\ ) = \ PGA(g)\ \times\ 9{,}80665$$

Keterangan:

PGA(m/s^2) : PGA dalam satuan meter per detik kuadrat

PGA(g) : PGA dalam satuan gravitasi

9,80665 : Konstanta percepatan gravitasi bumi (m/s^2)

5. Konversi ke satuan Galileo (Gal)

$$PGA\ (Gal) = \ PGA(m/s^2)\ \times\ 100$$

Keterangan:

PGA (Gal) : PGA dalam satuan Galileo atau (cm/s^2)

6. Estimasi magnitudo berbasis PGA

$$M\ \approx\ \frac{log_{10}(PGA_{Gal}) + 1.5}{1.5}$$

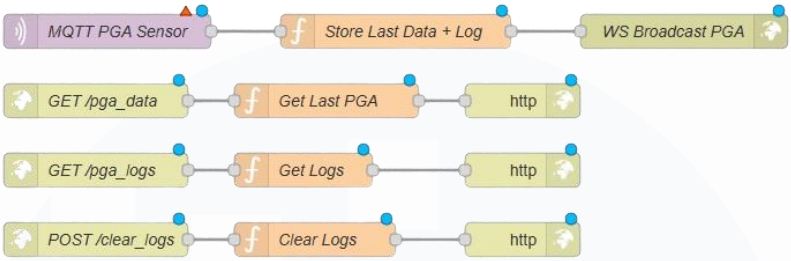
Keterangan:

M : magnitudo gempa hasil estimasi

PGA_{GAL} : Peak Ground Acceleration dalam satuan Galileo

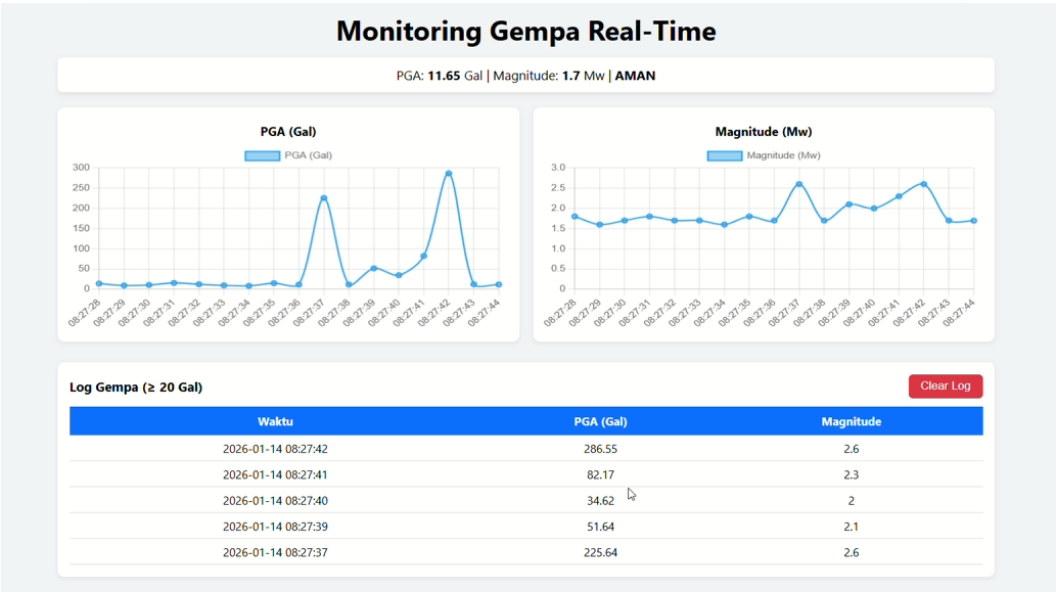
Log_{10} : fungsi logaritma basis 10

1.5 : konstanta empiris untuk penskalaan magnitudo



Gambar 3. 10 Diagram Flow Sistem Monitoring data PGA pada Node-RED

Pada Gambar 3.10 Monitoring data *Peak Ground Acceleration* (PGA) dilakukan secara *real-time* berbasis Node-RED dengan data sensor yang dikirim melalui protokol MQTT. Sensor accelerometer membaca getaran tanah, kemudian data tersebut diolah pada Raspberry Pi untuk menghasilkan nilai PGA dan estimasi magnitudo. Data hasil pengolahan dikirim secara periodik ke broker MQTT (HiveMQ) dalam format JSON dan diterima secara *real-time* oleh Node-RED melalui node *MQTT PGA Sensor* (*mqtt in*) yang melakukan *subscribe*. Selanjutnya, Node-RED memproses data menggunakan function node untuk menyimpan nilai PGA terbaru serta mencatat log kejadian apabila nilai PGA melebihi ambang batas 20 Gal. Node-RED juga menyediakan layanan API HTTP yang dapat diakses melalui website, sehingga data sensor terbaru dan log gempa dapat ditampilkan.



Gambar 3.11 Tampilan Website Monitoring PGA Real-Time

Pada Gambar 3.11 Menunjukan tampilan website yang mengambil data dari API Node-RED menggunakan JavaScript secara berkala dan menampilkannya dalam bentuk teks serta grafik interaktif berbasis Chart.js yang diperbarui secara *real-time*. Untuk memungkinkan akses dari jaringan luar, sistem dipublikasikan menggunakan NGROK sehingga website dan layanan Node-RED dapat diakses oleh perangkat lain dan digunakan sebagai media pemantauan kondisi getaran tanah secara langsung.

3.4 Kendala yang Ditemukan

Dalam pelaksanaan kegiatan ini, terdapat beberapa kendala yang dihadapi. Salah satunya adalah minimnya referensi yang secara khusus membahas sensor geophone, sehingga proses pemahaman karakteristik, prinsip kerja, dan penerapannya memerlukan waktu lebih lama. Selain itu, keterbatasan alat uji menyebabkan tidak dapat dihasilkannya gaya getaran yang setara dengan kondisi gempa bumi sebenarnya pada geophone, sehingga pengujian dilakukan dalam skala dan kondisi yang lebih sederhana.

3.5 Solusi atas Kendala yang Ditemukan

Sebagai solusi terhadap kendala tersebut, digunakan bantuan teknologi berupa AI dan referensi dari video YouTube untuk menambah pemahaman mengenai sensor geophone dan penerapannya. Selain itu, simulasi getaran dilakukan dengan cara menggetarkan meja sebagai pengganti kondisi gempa bumi saat proses pengambilan data, sehingga pengujian tetap dapat dilaksanakan meskipun dalam keterbatasan alat.