

**PERANCANGAN DAN IMPLEMENTASI *DATA ACQUISITION*
BERBASIS *REAL TIME OPERATING* UNTUK MASUKAN PEMODELAN
DAN SIMULASI KONSUMSI BATERAI KENDARAAN LISTRIK**

TUGAS AKHIR

**Karya tulis ilmiah sebagai salah satu syarat untuk memperoleh gelar Sarjana
Teknik dari Program Studi Teknik Elektro
Universitas Jenderal Achmad Yani**

Oleh

**WANDA ADI BUKHARI
2212212023**



**PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS JENDERAL ACHMAD YANI
2023**

LEMBAR PENGESAHAN

PERANCANGAN DAN IMPLEMENTASI DAQ BERBASIS REAL TIME OPERATING UNTUK MASUKAN PEMODELAN DAN SIMULASI KONSUMSI BATERAI KENDARAAN LISTRIK

Oleh

WANDA ADI BUKHARI

2212212023

Konsentrasi Teknik Kendali dan Instrumentasi



Tugas Akhir telah disetujui

Cimahi, 8 September 2023

Menyetujui,

Pembimbing

Dede Irawan Saputra, S.Pd., M.T.
NID. 412197591

Mengetahui,
Ketua Program Studi Teknik Elektro

Sofyan Basuki, S.T., M.T.
NID. 412178367

LEMBAR PERNYATAAN BEBAS PLAGIASI

Yang bertanda tangan di bawah ini

Nama Mahasiswa : Wanda Adi Bukhari

NIM : 2212212023

Program Studi : Teknik Elektro

Judul Tugas Akhir : PERANCANGAN DAN IMPLEMENTASI DAQ

BERBASIS REAL TIME OPERATING UNTUK MASUKAN PEMODELAN

DAN SIMULASI KONSUMSI BATERAI KENDARAAN LISTRIK

Tugas Akhir (skripsi) tersebut di atas telah dicek plagiasi oleh tim turnitin program studi, disetujui oleh pembimbing, serta diperkenankan untuk dilanjutkan proses cetak dan unggah mandiri.



Cimahi, 8 September 2023

Materai Rp. 10.000

Wanda Adi Bukhari

2212212023

LEMBAR PERNYATAAN PUBLIKASI KARYA ILMIAH

Yang bertanda tangan di bawah ini

Nama Mahasiswa : Wanda Adi Bukhari
Tempat, tgl lahir : Bandung, 10 Oktober 1998
NIM : 2212212023
Fakultas/Program Studi : Fakultas Teknik/ Teknik Elektro
Judul Tugas Akhir : PERANCANGAN DAN IMPLEMENTASI DAQ
BERBASIS REAL TIME OPERATING UNTUK MASUKAN PEMODELAN
DAN SIMULASI KONSUMSI BATERAI KENDARAAN LISTRIK

Demi pengembangan ilmu pengetahuan, kami memberikan ijin kepada **Universitas Jenderal Achmad Yani**, Hak Bebas Royalti Non-ekslusif (*Non-exclusive Royalty-Free Right*) atas karya ilmiah yang berjudul : “**Perancangan dan Implementasi DAQ Berbasis Real Time Operating untuk Masukan Pemodelan dan Simulasi Konsumsi Baterai Kendaraan Listrik**“.

Dengan Hak Bebas Royalti Non-ekslusif ini, **Universitas Jenderal Achmad Yani** berhak menyimpan, mengalih-mediakan, mengelolanya dalam bentuk pangkalan data (*database*), mendistribusikannya, menampilkan/mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin kami selama tetap mencantumkan nama-nama kami sebagai penulis/pemilik karya ilmiah ini.

Segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah ini menjadi tanggung jawab penulis.

Demikian pernyataan ini dibuat dengan sebenarnya.

Cimahi, 8 September 2023

Wanda Adi Bukhari
2212212023

ABSTRAK

Simulasi kendaraan listrik dibuat untuk memungkinkan para insinyur dan pengembang untuk menguji dan memperbaiki kinerja kendaraan listrik sebelum dibuat secara fisik. Dalam simulasi, para pengembang dapat mengevaluasi efisiensi baterai, kecepatan maksimum, kemiringan jalan, daya tahan dan berbagai parameter teknis lainnya. Input simulasi dapat dihasilkan berdasarkan pembacaan dan pengukuran kondisi yang sebenarnya berdasarkan pembacaan DAQ. Hal ini berugna untuk mempercepat pengembangan kendaraan listrik karena setiap cacat atau kekurangan dapat diidentifikasi dan diperbaiki sebelum kendaraan benar-benar dibuat.

Perancangan yang dibuat dengan menggunakan simulink MATLAB merupakan mobil listrik dengan beberapa bagian komponen seperti bagian masukan yang di representasikan dengan bagian masukan, batrai, kontrol motor, dan bagian bodi mobil. Simulasi akan berjalan secara sekuensial dimulai dari bagian masukan sampai bodi mobil dengan masing-masing bagian dapat diidentifikasi besaran yang bisa diambil seperti SOC(State of Charge) dari batrai, arus yang mengalir pada beban dan parameter lainnya. Masukan yang akan diolah oleh sistem hanya besaran kecepatan yang didapat dari longitude dan latitude, kecepatan umpan balik setelah satu proses dan kemiringan jalan.

Perancangan DAQ memudahkan pengambilan data GPS dan kemiringan untuk parameter masukan. Besaran yang dapat diambil seperti kecepatan dalam pengambilan data lebih baik dan data dapat disimpan didalam database server. Sistem yang dijalankan berbasis RTOS(Real Time Operating System) sehingga memudahkan dalam monitoring secara langsung. Sampel data diambil dari lokasi awal perumahan Cikarang (Meadow Green) dan titik akhir di Cifest Cikarang dengan kecepatan rata-rata 2.17 (m/s) hasil pembacaan DAQ.

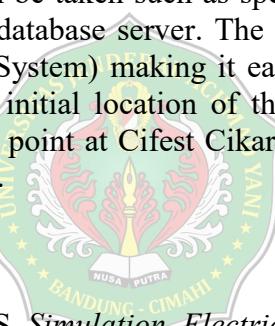
Kata kunci: DAQ, Batrai, GPS, Simulasi, Kendaraan Listrik

ABSTRACT

Electric vehicle simulations are created to allow engineers and developers to test and improve the performance of electric vehicles before they are physically built. In the simulation, the developers were able to evaluate battery efficiency, maximum speed, road gradient, endurance and various other technical parameters. Simulation input can be generated based on actual condition readings and measurements based on DAQ readings. This is useful for speeding up the development of electric vehicles because any defects or shortcomings can be identified and corrected before the vehicle is actually built.

The design created using MATLAB simulink is an electric car with several component parts such as the input part which is represented by the input part, battery, motor controls, and car body parts. The simulation will run sequentially starting from the input part to the car body with each part identifying quantities that can be taken such as the SOC (State of Charge) of the battery, the current flowing in the load and other parameters. The input that will be processed by the system is only the speed obtained from longitude and latitude, feedback speed after one process and the slope of the road.

The DAQ design makes it easy to retrieve GPS and slope data for input parameters. Quantities that can be taken such as speed in data retrieval are better and data can be stored in the database server. The system being run is based on RTOS (Real Time Operating System) making it easier to monitor directly. Data samples were taken from the initial location of the Cikarang housing complex (Meadow Green) and the final point at Cifest Cikarang with an average speed of 2.17 (m/s) from DAQ readings.



Keywords: DAQ, Battery, GPS, Simulation, Electric Vehicle

KATA PENGANTAR

Alhamdulillah, puji syukur penulis panjatkan kepada Allah SWT atas karunia dan kemampuan yang telah diberikan sehingga dapat menyelesaikan tugas akhir yang hasilnya dilaporkan dalam karya tulis. Penulis ingin menyampaikan ucapan terima kasih kepada pihak yang secara langsung maupun tidak langsung memberikan bantuan dalam penyelesaian tugas akhir. Penulis sangat berterima kasih kepada:

1. Orang tua yang telah membantu secara moril maupun materil dan tidak lelah mendukung dengan doa yang tidak pernah henti sehingga penulis dapat menyelesaikan studi di Jurusan Teknik Elektro Unjani.
2. Penulis sangat berterima kasih kepada Bapak Dede Irawan Saputra, S.Pd., M.T. sebagai pembimbing utama.
3. Penulis berterima kasih kepada Bapak Een Taryana, ST., MT. sebagai ketua Jurusan Program Studi Teknik Elektro Universitas Jenderal Achmad Yani.
4. Penulis juga berterima kasih kepada Ibu Nivika Tiffany Somantri, ST., MT. sebagai dosen wali akademik yang telah memberikan bimbingan dalam setiap proses akademik.



Semoga Tugas Akhir ini dapat bermanfaat untuk pengembangan ilmu pengetahuan.

Cimahi, 1, September 2023

Wanda Adi Bukhari
22122212023

DAFTAR ISI

LEMBAR PERNYATAAN BEBAS PLAGIASI	iii
LEMBAR PERNYATAAN PUBLIKASI KARYA ILMIAH	iv
ABSTRAK	v
ABSTRACT	vi
KATA PENGANTAR	vii
DAFTAR ISI	viii
DAFTAR LAMPIRAN	x
DAFTAR GAMBAR	xi
DAFTAR TABEL	xii
DAFTAR SINGKATAN	xiii
BAB I	1
PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Tujuan Penelitian	2
1.3. Batasan Masalah	3
1.4. Sistematika Penulisan	3
BAB II	4
TEORI PENUNJANG	4
2.1 Studi Penelitian Terdahulu	4
2.2 BMS (Battery Management System)	5
2.3 State of Charge dan Depth of Discharge	7
2.4 Akuisisi Data	8
2.5 Protokol MQTT (Message Queuing Telemetry Transport)	9
2.6 Raspberry Pi 4	10
2.7 Node Red	12
2.8 Database SQLITE	13
BAB III	21
METODE PENELITIAN	21
3.1 Arsitektur Umum Sistem Perancangan	21

3.2 Perancangan Software Akuisisi Data	26
3.2.1 Node-red	26
3.2.2 Databse SQLITE	29
3.3 Perancangan Alat Akuisisi Data	31
3.3.1 Skematik Rangkaian	31
3.3.2 Perancangan Hardware	32
BAB V	60
KESIMPULAN DAN SARAN	60
5.1 Kesimpulan	60
5.2 Saran	60
DAFTAR PUSTAKA	62



DAFTAR LAMPIRAN

LAMPIRAN A SPESIFIKASI SENSOR ULTRASONIK.....	A-1
LAMPIRAN B DIAGRAM SKEMATIK RANGKAIAN DAN HASIL PENGUKURAN.....	B-1
LAMPIRAN C KETENTUAN UNGGAH MANDIRI	B-3
LAMPIRAN D TIM PENYUSUN DRAFT TA	B-4



DAFTAR GAMBAR

Gambar 2.1 Diagram blok BMS	6
Gambar 2.2 SOC dan DOD	8
Gambar 2.3 Arsitektur sistem MQTT	9
Gambar 2.4 Raspberry Pi 4 model B [8]	10
Gambar 2.5 Node-Red [9]	12
Gambar 2.6 database SQLite[10]	13
Gambar 3.2 Metodologi VDI2206	21
Gambar 3.5 <i>Node</i> akuisisi data Node-RED	26
Gambar 3.6 Flow process dari MQTT sampai database	27
Gambar 3.7 Flow process dari node MQTT	27
Gambar 3.8 Proses node Joint	27
Gambar 3.9 Proses node function query	28
Gambar 3.10 Proses node database SQLITE	28
Gambar 3.11 Proses node debug	29
Gambar 3.12 Database SQLite	29
Gambar 3.13 lokasi penyimpanan database	30
Gambar 3.14 Skematik Rangkaian DAQ	31
Gambar 3.15 Rancangan Komponen Hardware	32
Gambar 3.16 Diagram alir pengambilan data	33

DAFTAR TABEL

Tabel 1.1 Penelitian terdahulu	4
--------------------------------------	---



DAFTAR SINGKATAN

Singkatan	Arti	Pemakaian pertama kali pada halaman
I	Arus listrik [Ampere]	1
V	Beda potensial [Volt]	1
BMS	Battery Management System	1
IoT	Internet of Things	1
MQTT	Message Queuing Telemetry Transport	1
SOC	State of Charge	5
DOD	Depth of Discharge	4
JSON	JavaScript Object Notation	1
OS	Operating System	1



BAB I

PENDAHULUAN

1.1. Latar Belakang

Kendaraan listrik merupakan kendaraan yang dapat bekerja dengan bantuan energi listrik. Tidak seperti kendaraan yang menggunakan minyak untuk dapat bekerja kendaraan listrik menggunakan batrai sebagai penyimpan energi. Kendaraan listrik juga memiliki beberapa keterbatasan seperti jarak tempuh yang relatif tidak jauh, durasi pengisian kembali batrai yang relatif lebih lama, metode pengisian batrai yang hanya memiliki beberapa tempat untuk pengisian nya dan juga keterbatasan lainnya. Ada beberapa faktor yang mempengaruhi konsumsi batrai pada kendaraan listrik baik itu kemiringan jalan, pengaturan pedal gas untuk kendali motor penggerak. Sistem pengambilan data secara *real time* dapat memudahkan dalam melakukan pengecekan dan menjadi sumber data apabila akan meentukan sebuah kesimpulan. Oleh karena itu, riset yang dilakukan secara konsisten dalam memberikan kontribusi untuk memudahkan pemecahan masalah yang terkait dengan parapran diatas. Beberapa contoh dalam riset pengaplikasian nya seperti sumber energi listrik tambahan dapat dimanfaatkan untuk memperpanjang lifetime baterai. Seperti contoh, dengan hanya menambahkan energisuperkapasitor, maka energi baterai dapat direduksi 23,23% [1].

Kendaraan listrik dalam penggunaan nya ramah untuk lingkungan serta mengurangi dampak bagi polusi udara. Indonesia secara umum memiliki udara yang relatif tidak sehat sebab pencemaran dari beberapa lini dan termasuk yang terbesar disebabkan oleh transportasi umum. Kendaraan listrik juga dapat digunakan dengan jarak yang cukup jauh. Jarak tempuh mobil listrik dengan kondisi baterai penuh terus mengalami peningkatan dari tahun ke tahun. Pada tahun 2021, IEA mencatat bahwa rata-rata jarak tempuh mobil listrik dengan kondisi baterai penuh adalah 349 kilometer. Jarak tersebut lebih panjang dari perjalanan Jakarta-Bandung pulang pergi dengan mobil[2]. Dan penggunaan nya tidak hanya dapat digunakan dalam perkotaan tapi juga dapat digunakan dalam jalanan pedesaan. Solusi preventif untuk mengoptimalkan adalah dengan melakukan simulasi dan identifikasi berdasarkan perilaku berkendara seperti

kecepatan dan medan seperti jalan. Medan jalan yang dilalui oleh mobil listrik tersebut berupa tanjakan dan turunan yang dapat memengaruhi konsumsi batrai. Hal ini tercantum dalam Instruksi Presiden (INPRES) Nomor 7 Tahun 2022 tentang Penggunaan Kendaraan Bermotor Listrik Berbasis Baterai (*Battery Electric Vehicle*) Sebagai Kendaraan Dinas Operasional dan/atau Kendaraan Perorangan Dinas Instansi Pemerintah Pusat dan Pemerintahan Daerah[3]. Peraturan Presiden (PERPRES) Nomor 55 Tahun 2019 tentang Percepatan Program Kendaraan Bermotor Listrik Berbasis Baterai (*Battery Electric Vehicle*) untuk Transportasi Jalan.

Selain itu, permasalahan yang masih menjadi perhatian antara lain seperti jarak mengemudi yang terbatas, waktu pengisian yang lama, dan di beberapa daerah jumlah infrastruktur pengisian daya yang terbatas masih menjadi hambatan utama untuk beralih menggunakan kendaraan listrik. Kombinasi dari faktor-faktor pembatas tersebut menyebabkan fenomena yang disebut ‘*range anxiety*’ yang menjadikan pengendara merasa khawatir terkait jarak tempuh apabila tidak dapat sampai pada tujuan. Meskipun kecemasan terhadap jarak berkendara dapat dikurangi dengan meningkatkan kapasitas paket baterai, meningkatkan daya pengisian yang diizinkan, dan membangun lebih banyak stasiun pengisian daya. Solusi tersebut membutuhkan waktu dalam membangun infrastruktur yang memadai dan tidak secara langsung meningkatkan kepercayaan pengemudi pada jarak berkendara yang tersisa. Pengguna kendaraan listrik sering mencadangkan buffer sekitar 20% dari kapasitas baterai untuk menghindari kehabisan daya di perjalanan [4].

Meningkatnya kebutuhan akan informasi, monitoring, dan pengendalian jarak jauh terhadap peralatan, industri, dan otomotif membuat teknologi RTOS sistem monitoring kendaraan digunakan untuk memudahkan pengguna dalam mendapatkan informasi disekitar kendaraan secara real-time.

1.2. Tujuan Penelitian

1. Merancang sistem DAQ(*Data Acquasition*) untuk masukan simulasi konsumsi kendaraan listrik berbasis RTOS(*Real Time Operating System*).
2. Merancang model simulasi kendaraan listrik menggunakan simulink.

3. Menganalisis konsumsi baterai berdasarkan kecepatan, kemiringan jalan dan jarak tempuh.

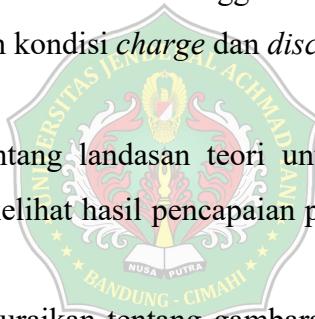
1.3. Batasan Masalah

Adapun batasan masalah pada tugas akhir meliputi.

1. Parameter masukan yang diobservasi hanya meliputi koordinat dan kondisi terhadap kemiringan suatu kondisi jalan.
2. Jenis sensor yang digunakan u-bok NEO-8M untuk menentukan koordinat longitude dan latitude.
3. Sensor u-bok NEO-8M untuk menentukan besaran altitude.
4. Proses simulasi menggunakan software MATLAB untuk menjalankan pemodelan kendaraan listrik.
5. Motor untuk menggerakan roda pada simulasi kendaraan listrik menggunakan motor DC.
6. Batria yang digunakan untuk simulasi menggunakan jenis batria tipe *lithium-ion* supaya memungkinkan kondisi *charge* dan *discharge*.

1.4. Sistematika Penulisan

Bab I Pendahuluan berisi tentang landasan teori untuk menjelaskan beberapa istilah dan ilmu terkait serta melihat hasil pencapaian penelitian terdahulu dengan kajian yang sama.



Bab II Teori Penunjang menguraikan tentang gambaran umum tentang landasan teori untuk menjelaskan beberapa istilah dan ilmu terkait serta melihat hasil pencapaian penelitian terdahulu dengan kajian yang sama.

Bab III Metode Penelitian memuat tentang langkah-langkah penyelesaian tugas akhir berupa gambaran umum sistem serta perancangan sistem.

Bab IV Hasil Pengujian dan Analisis menjelaskan tentang rancangan jadwal kegiatan TA dan rincian anggaran biaya untuk penyelesaian TA.

Bab V Kesimpulan dan Saran saran.

BAB II

TEORI PENUNJANG

2.1 Studi Penelitian Terdahulu

Berdasarkan hasil studi yang dilakukan terhadap beberapa penelitian terdahulu, diperoleh berbagai informasi mengenai sistem yang telah dibuat. Berikut adalah hasil studi terdahulu pada Tabel 1.1.

Tabel 1.1 Penelitian terdahulu

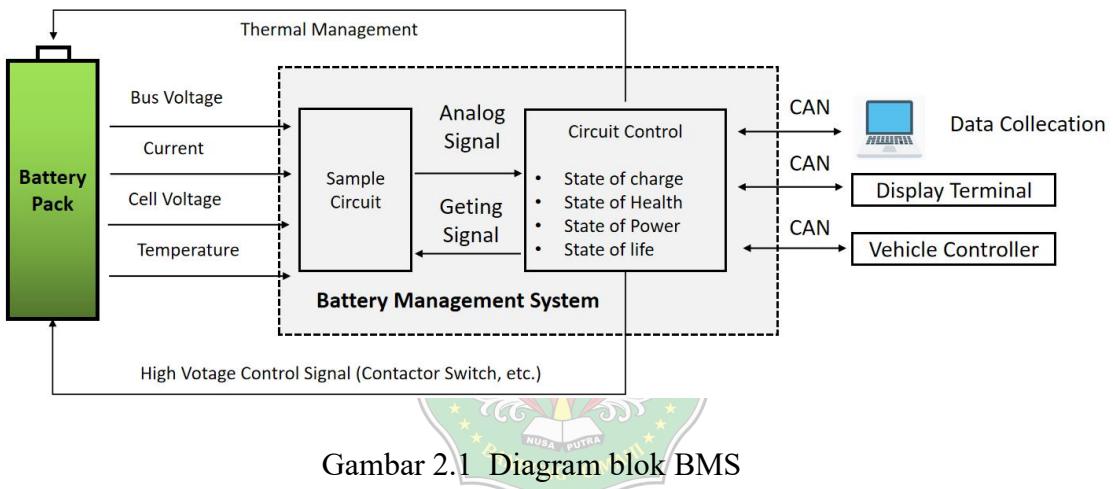
No	Judul Penelitian	Nama Peneliti	Hasil Penelitian/Kesimpulan
1	Implementasi Sistem <i>Monitoring</i> Jarak Tempuh pada Sepeda Motor Listrik [23]	Ricky Bagas Setiawan, Dr. Muhammad Reza.,S.T., M.Sc. , dan Sigit Yuwono.,S.T., MSc., PhD. (2019)	<ul style="list-style-type: none">➤ Dalam penelitian ini dibuat sistem <i>monitoring</i> jarak tempuh dengan data <i>logging</i> berbasis <i>SOC</i> dan <i>SOH</i>.➤ Model data <i>logging</i> yang digunakan berdasarkan parameter <i>tracking</i> jarak tempuh melalui gps yang akan <i>Li-ion</i> tegangan dan kapasitas baterai (<i>SOC</i> dan <i>SOH</i>) setiap 50 meter.➤ Sistem <i>monitoring</i> hanya merekam <i>SOC</i> dan <i>SOH</i> dengan komunikasi menggunakan <i>bluetooth</i> dengan <i>delay</i> selama 6 detik
2	Desain dan Implementasi Sistem <i>Monitoring</i> dan Manajemen Baterai Mobil Listrik [24]	Bayu Segara Putra, Angga Rusdinae, dan Ekki Kurniawan. (2015)	<ul style="list-style-type: none">➤ Dalam penelitian ini dibuat sistem monitoring baterai dengan parameternya adalah arus dan tegangan.➤ Menggunakan mikrokontroler Arduino Due dan ditampilkan pada <i>LCD</i> secara <i>real-time</i>.➤ Sistem <i>monitoring</i> baterai dapat memprediksi sisa waktu dan jarak mobil melaju dengan baik.➤ Sistem <i>monitoring</i> masih lokal

			hanya pada <i>dashboard</i> kendaraan.
3	Desain dan Implementasi untuk <i>Monitoring</i> dan Manajemen Energi pada <i>Charging Station</i> Kendaraan Listrik Berbasis CAN Bus [25]	Giri Sasongko S S	<ul style="list-style-type: none"> ➤ Sistem monitoring menggunakan topologi CAN Bus yang dapat melakukan komunikasi secara serial yang ditampilkan pada suatu <i>graphical user interface</i> (GUI). ➤ Sistem masih belum diterapkan langsung pada <i>charging station</i>.
4	Sistem <i>Monitoring</i> dan Manajemen Baterai pada Mobil Listrik [26]	M. Arya Harisa Ashari, Angga Rusdinar, dan Porman Pangaribuan. (2018)	<ul style="list-style-type: none"> ➤ Sistem monitoring menampilkan informasi kapasitas baterai, kecepatan mobil, serta estimasi jarak tempuh maksimal. ➤ Sistem monitoring masih lokal hanya pada <i>dashboard</i> kendaraan.
5	Analisa Perbandingan Protokol <i>MQTT</i> dengan <i>HTTP</i> pada <i>IoT</i> Platform Patriot [27]	Ninditha Putri Windryani, Dr. Nyoman Bogi A. K, S.T., MSEE., dan Ratna Mayasari, S.T., M.T. (2019)	<ul style="list-style-type: none"> ➤ Protokol <i>MQTT</i> dapat lebih handal berjalan pada keadaan bandwidth rendah atau <i>latency</i> tinggi dibandingkan protokol <i>HTTP</i>. ➤ Penerapan protokol <i>MQTT</i> sebagai pendukung platform <i>IoT</i> masih berbasis web server.

2.2 BMS (Battery Management System)

Battery Management System merupakan sebuah sistem yang dirancang untuk mengelola dan memantau kinerja baterai dalam aplikasi nya untuk peralatan listrik seperti kendaraan listrik. BMS memiliki tujuan untuk

meningkatkan umur baterai, efisiensi, keamanan, dan kinerja keseluruhan baterai. Besaran-besaran fisika yang dapat diamati antara lain seperti tegangan, arus, daya, temperature dapat diamati dengan suatu alat tertentu. Akuisisi data secara simultan dilakukan dengan laju pengambilan sampel yang tinggi untuk meningkatkan akurasi dan mencapai kemampuan *monitoring* yang diinginkan tanpa propagasi kesalahan *PoE* (Propagation of Error). Bagian *software* bertanggung jawab untuk memantau variabel secara aktif, memberikan peringatan/perlindungan, melakukan komunikasi internal/eksternal, melakukan berbagai jenis estimasi seperti *State of Charge* dan melakukan penyesuaian sel pada paket baterai untuk menghindari degradasi baterai yang cepat.



Gambar 2.1 Diagram blok BMS

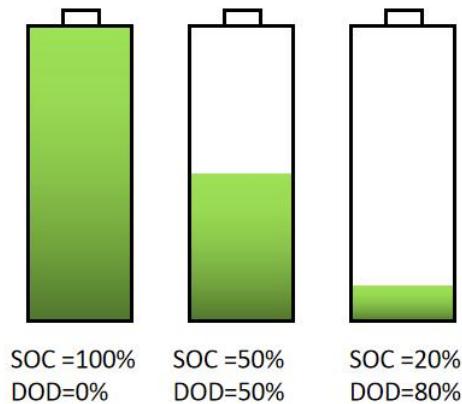
Kerangka kerja ini dikenal sebagai *BMS* (Battery Management System) dan dipandang sebagai otak pada paket baterai. Pada (Gambar 2.1) menunjukan kerangka kerja *hardware* dan *software* dari *BMS*. *BMS* yang kuat harus memberikan kontrol baterai yang lebih baik, menjamin aktivitas baterai yang aman, memberikan kekuatan paling ekstrem yang diperlukan, dan memperpanjang masa pakai baterai. *BMS* juga harus berkomunikasi dengan sistem *onboard* lainnya, seperti pengontrol mesin, pengontrol temperatur, sistem *monitoring*, dan komputer kendaraan untuk memelihara dan melakukan fungsi yang berbeda [5].

2.3 State of Charge dan Depth of Discharge

SOC (State of Charge) menggambarkan kemampuan baterai yang tersisa sebagai proporsi dari total kapasitas dalam kondisi yang sama. Ini dapat dianggap setara dengan pengukur bahan bakar di kendaraan mesin pembakaran. Estimasi *SOC* sangat penting untuk memantau dan mengoptimalkan kinerja baterai dengan mengontrol pengisian dan pemakaianya. *SOC* mengacu pada persentase berapa banyak kapasitas baterai yang masih tersisa dibandingkan dengan kapasitas penuhnya. Dalam konteks manajemen baterai, *SOC* adalah ukuran yang penting untuk mengetahui seberapa banyak energi yang masih tersedia dalam baterai pada suatu waktu tertentu. Estimasi *SOC* yang akurat dapat mengetahui berapa lama kendaraan listrik dapat mengemudi sebelum mengisi daya. Umumnya kapasitas yang dapat digunakan berkisar dari 100% hingga 5% *SOC* dan waktu yang dibutuhkan untuk mengosongkan baterai dari 100% hingga 5% didefinisikan sebagai waktu pengoperasian baterai (discharging). Karakteristik non-linear dan non-stasioner baterai mempengaruhi penurunan kapasitasnya dari waktu ke waktu dan fungsi *BMS* adalah untuk menjaga baterai di atas *SOC* tertentu dengan mencegahnya dari *overdischarging*. *SOC* tidak dapat diperkirakan secara langsung. Jika $I_b(\tau)$ adalah arus *charging*, maka muatan yang dikirim ke baterai didefinisikan sebagai $\int_{t_0}^t I_b(\tau)d\tau$. Maka perhitungan untuk nilai *SOC* adalah:

$$SOC_{(t)} = \frac{\int_{t_0}^t I_b(\tau)d\tau}{Q_0} \times 100\%$$

Dimana kapasitas baterai didefinisikan sebagai Q_0 pada waktu t. Umumnya, estimasi langsung (*coulomb counting*, berbasis *open circuit voltage* (OCV), dll.), berbasis model (*equivalent circuit* dan elektrokimia, dll.) [6].



Gambar 2.2 SOC dan DOD

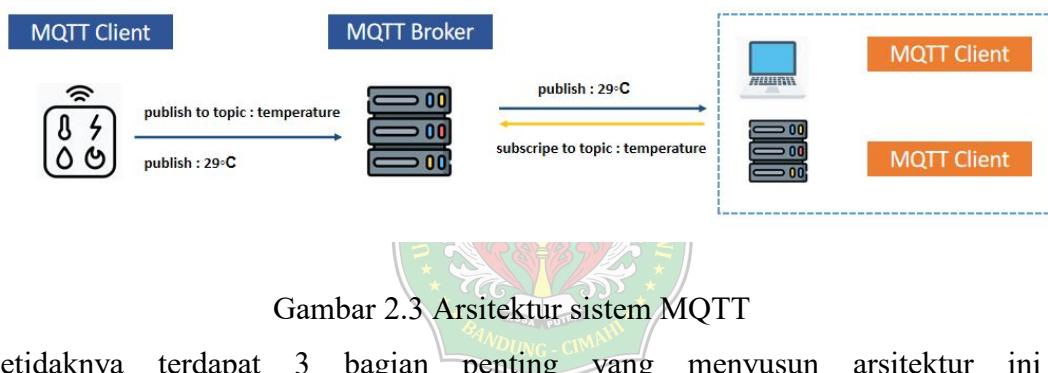
Pada gambar ditunjukan SOC ketika memiliki nilai maksimal ada pada kondisi 100% dan setengah dari kondisi maksimal berapa pada 50% sedangkan posisi rawan sudah berapa pada 20%. Konsep ini memiliki kemiripan dengan konsumsi bensin pada motor dengan sumber daya fosil. Dept of Discharge (DOD) merupakan metode alternatif untuk menunjukan status SOC. DOD adalah pelengkap SOC. Sementara nilai SOC menunjukan nilai persen (0% = kosong; 100% = penuh, DOD dapat menggunakan unit Ah (0 Ah = penuh; 50 Ah = kosong) atau nilai persen (100% = kosong; 0% = penuh).

2.4 Akuisisi Data

Sebuah sistem akuisisi data atau biasa dikenal *Data Acquisition System* (DAS) merupakan sistem instrumentasi elektronik. Terdiri dari sejumlah elemen yang secara bersama-sama bertujuan untuk melakukan pengukuran, menyimpan, dan mengolah hasil pengukuran. Secara aktual DAS berupa *interface* antara lingkungan analog dengan lingkungan digital. Lingkungan analog meliputi transduser dan pengondisian sinyal dengan segala kelengkapannya, sedangkan lingkungan digital meliputi *analog to digital converter* (ADC) dan selanjutnya pemrosesan digital yang dilakukan oleh mikroprosesor atau sistem berbasis mikroprosesor. Komputer yang digunakan untuk sistem akuisisi data dapat mempengaruhi kecepatan akuisisi data [7].

2.5 Protokol MQTT (Message Queuing Telemetry Transport)

Protokol *MQTT* (Message Queue Telemetry Transport) adalah protokol pesan ringan (lightweight) berbasis *publish-subscribe* (Gambar II.7) digunakan di atas protokol *TCP/IP*. *MQTT* (Message Queuing Telemetry Transport) adalah protokol komunikasi ringan yang dirancang khusus untuk pertukaran pesan antara perangkat-perangkat dalam jaringan. Sistem ini memiliki keterbatasan sumber daya, seperti perangkat Internet of Things (IoT), dan koneksi internet sesuai dengan kebutuhan *device*. *MQTT* digunakan untuk mentransmisikan data melalui jaringan dengan overhead yang rendah, membuatnya cocok untuk aplikasi yang membutuhkan efisiensi dan respons cepat, terutama dalam lingkungan dengan koneksi yang tidak stabil atau bandwidth terbatas.



Gambar 2.3 Arsitektur sistem MQTT

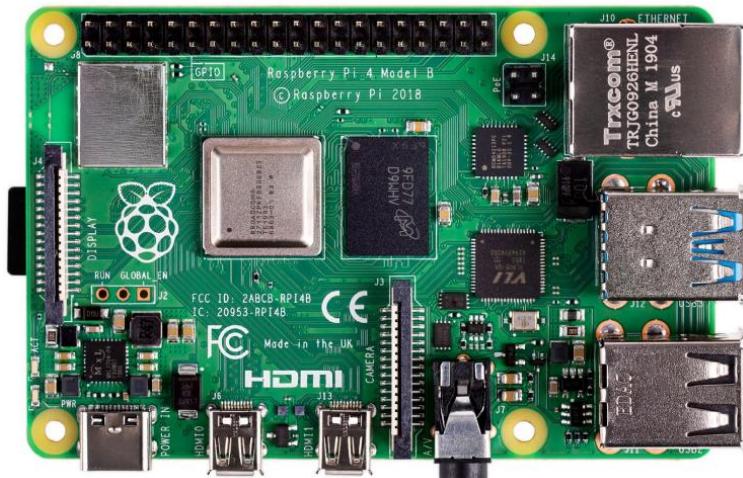
Setidaknya terdapat 3 bagian penting yang menyusun arsitektur ini dengan bagian masing-masing antara lain *MQTT Client*, *MQTT broker*, dan bagian *customer* atau bagian yang memanfaatkan sistem tersebut. Protokol ini mempunyai ukuran paket data *low overhead* kecil (minimal 2 gigabytes) dengan konsumsi catu daya kecil. *MQTT* bersifat terbuka, ringkas, dan didesain agar mudah untuk diimplementasikan, yang mampu menangani ribuan *client* jarak jauh hanya dengan satu server. Karakteristik ini membuatnya ideal untuk digunakan dalam banyak situasi, termasuk lingkungan terbatas seperti dalam komunikasi *Machine to Machine* (M2M) dan konteks *Internet of Things* (IoT) dimana dibutuhkan kode *footprint* yang kecil dan/atau jaringan yang terbatas. Pola pesan *publish-subscribe* membutuhkan *broker* pesan. *Broker*

bertanggung jawab untuk mendistribusikan pesan ke *client* tertarik berdasarkan topik pesan [6].

Berikut merupakan fitur protokol *MQTT*:

- 1) *Publish/subscribe message pattern* yang menyediakan distribusi *message* dari satu ke banyak dan *decoupling* aplikasi.
- 2) *Messagging transport* yang *agnostic* dengan isi dari *payload*.
- 3) Menggunakan *TCP/IP* sebagai koneksi dasar jaringan.
- 4) Terdapat tiga *level Qualities of Service* (QoS) dalam penyampaian pesan :
 - a) “*At most once*”, di mana pesan dikirim dengan upaya terbaik dari jaringan *TCP/IP*. Kehilangan pesan atau terjadi duplikasi dapat terjadi.
 - b) “*At least once*”, dapat dipastikan pesan tersampaikan walaupun duplikasi dapat terjadi.
 - c) “*Exactly once*”, dimana pesan dapat dipastikan tiba tepat satu kali.

2.6 Raspberry Pi 4



Gambar 2.4 Raspberry Pi 4 model B [8]

Raspberry Pi adalah sebuah komputer papan tunggal (single-board computer) yang memiliki fungsi utama sebagai CPU (Central Processing Unit). Raspberry Pi memiliki dimensi yang jauh lebih kecil dibandingkan dengan komputer biasa, tetapi tetap mampu menjalankan fungsi komputasi yang kompleks dan beragam. Board ini pertama kali dirancang dan dikembangkan oleh Raspberry Pi Foundation di Inggris untuk tujuan pendidikan dalam rangka meningkatkan keterampilan komputasi, terutama bagi pelajar di seluruh dunia.

Raspberry Pi sangat fleksibel dan serbaguna dalam berbagai aplikasi, mulai dari pengembangan perangkat lunak, robotika, IoT (Internet of Things), hingga server mini untuk berbagai tugas jaringan. Pada dasarnya, Raspberry Pi dapat beroperasi seperti komputer desktop biasa, termasuk menjalankan sistem operasi berbasis Linux, menjelajah internet melalui browser, serta mengakses dan memanipulasi database.

2.6.1 Keunggulan Raspberry Pi dalam Sistem Embedded

Salah satu keunggulan utama Raspberry Pi dibandingkan dengan sistem embedded lain adalah adanya General Purpose Input/Output (GPIO) pins yang tersedia pada board. GPIO adalah pin yang dapat diprogram dan digunakan untuk berinteraksi dengan perangkat eksternal seperti sensor, aktuator, motor, dan modul komunikasi lainnya. Pin GPIO pada Raspberry Pi memungkinkan board ini digunakan untuk aplikasi yang melibatkan kontrol langsung terhadap perangkat keras (hardware control), yang membuatnya sangat ideal untuk pengembangan proyek berbasis sistem kontrol, otomatisasi, dan robotika.

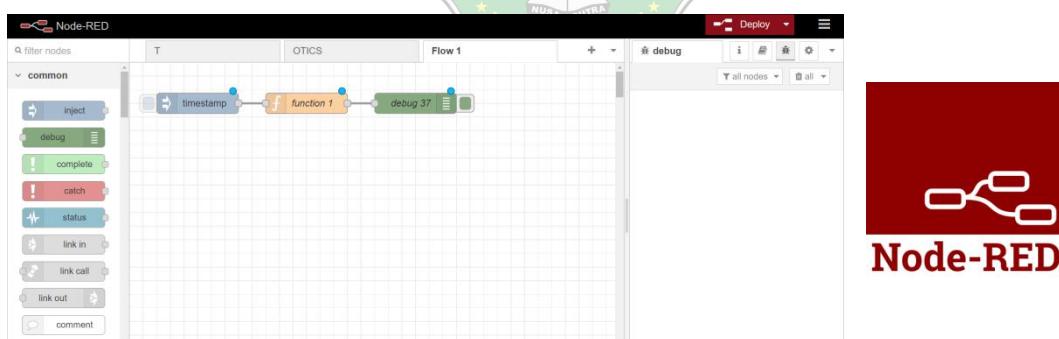
Raspberry Pi beroperasi pada tegangan 3.3V yang relatif rendah, sehingga memastikan keamanan dalam aplikasi elektronik yang melibatkan microcontroller. Namun, untuk mencegah terjadinya short circuit atau korsleting, terutama ketika Raspberry Pi dihubungkan dengan perangkat eksternal yang bekerja pada tegangan lebih tinggi seperti 5V atau 12V, diperlukan rangkaian safety, seperti penggunaan resistor, dioda, transistor, atau modul relay untuk menghindari kerusakan permanen pada board.

2.6.2 Spesifikasi Raspberry Pi Model B

Model Raspberry Pi yang banyak digunakan dalam berbagai proyek, termasuk proyek akademik dan industri, adalah Raspberry Pi 4 Model B. Board ini menyediakan kinerja yang cukup baik untuk menjalankan aplikasi yang relatif kompleks, seperti browser, server web, media center, hingga aplikasi IoT. LAN *nirkabel* dual-band dan Bluetooth memiliki sertifikasi kepatuhan modular, memungkinkan board dirancang menjadi produk akhir berstandar industri namun tetap memiliki nilai ekonomis untuk pengembangan lebih lanjut [8].

2.7 Node Red

Node-RED adalah sebuah aplikasi pengembangan visual yang dirancang untuk menghubungkan berbagai perangkat. Perangkat yang dapat dihubungkan antara lain melalui layanan, dan API dalam lingkungan IoT(*Internet of Things*) dan pengembangan aplikasi berbasis aliran data *flow-based programming*. Dikembangkan oleh IBM, Node-RED memberikan antarmuka grafis yang memungkinkan pengguna untuk membuat alur kerja (flows) dengan menggabungkan berbagai node yang mewakili fungsi-fungsi berbeda.



Gambar 2.5 Node-Red [9]

Dalam pengaplikasian nya node-red menjadi jembatan penghubung antara device dengan server utama sebagai pengolah data. Metode pegambilan data yang dilakukan menggunakan beberapa metode komunikasi seperti serial, pararel, TCP/IP dan metode komunikasi lain-lainnya.

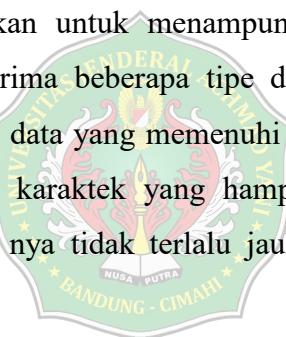
2.8 Database SQLITE

SQLite adalah sebuah sistem manajemen basis data relasional (RDBMS) yang bersifat self-contained atau mandiri. Artinya, semua komponen SQLite termasuk dalam satu file tunggal, tanpa perlu konfigurasi server eksternal atau pengaturan tambahan. SQLite dirancang untuk menyimpan, mengelola, dan mengakses data dalam aplikasi, baik itu aplikasi desktop, mobile, atau embedded.



Gambar 2.6 database SQLite[10]

Database sqlite relatif lebih ringan ketika beroperasi jika dibandingkan dengan database sejenis. Oleh karenanya, dapat menjadi lebih baik ketika beroperasi pada raspberrypi yang akan digunakan untuk menampung data *longging*, maupun *realtime*. Database dapat menerima beberapa tipe data seperti *string*, *integer*, *boolean*, *char* dan beberapa tipe data yang memenuhi syarat operasi pada *syntax query* SQL. SQLITE memiliki karakter yang hampir mirip dengan MYSQL sehingga dalam pengaplikasian nya tidak terlalu jauh baik itu dijalankan oleh program PHP, Python dll.



2.9 Sensor Ublox GPS

Sensor GPS u-blox adalah salah satu sensor GPS yang terkenal karena keakuratan dan kinerjanya yang andal dalam berbagai aplikasi, termasuk penelitian dan proyek akademik seperti skripsi. Modul u-blox, seperti seri NEO atau ZED, sering digunakan dalam berbagai bidang seperti navigasi, robotika, pemetaan, dan pelacakan kendaraan. Untuk sebuah skripsi, penggunaan sensor u-blox menawarkan sejumlah keunggulan teknis yang relevan untuk penelitian yang melibatkan lokasi atau data geospasial.



Gambar 2.7 Sensor GPS

Senosr GPS di setting untuk mendapatkan signal koordinat untuk menentukan dimana posisi kondaraan sedang berada. Dan setiap beberapa saat koordinat akan update setiap beberapa saat. GPS u-blox bekerja dengan menangkap sinyal dari beberapa satelit yang mengorbit bumi. Sinyal ini berisi informasi waktu dan posisi satelit. Sensor kemudian menghitung jarak dari satelit tersebut menggunakan perbedaan waktu yang ditempuh oleh sinyal untuk mencapai perangkat. Dengan data dari minimal empat satelit, sensor GPS dapat menghitung posisi tiga dimensi (lintang, bujur, dan ketinggian) serta waktu yang sangat akurat.

2.10 Sensor Gyroscope

Sensor GPS u-blox adalah salah satu sensor GPS yang terkenal karena keakuratan dan kinerjanya yang andal dalam berbagai aplikasi, termasuk penelitian dan proyek akademik seperti skripsi. Modul u-blox, seperti seri NEO atau ZED, sering digunakan dalam berbagai bidang seperti navigasi, robotika, pemetaan, dan pelacakan kendaraan. Untuk sebuah skripsi, penggunaan sensor u-blox menawarkan sejumlah keunggulan teknis yang relevan untuk penelitian yang melibatkan lokasi atau data geospasial.



Gambar 2.8 Sensor Gyroscope

Modul GPS u-blox dapat dengan mudah diintegrasikan dengan mikrokontroler atau sistem komputer seperti Raspberry Pi, Arduino, dan ESP32. Antarmuka standar seperti UART, SPI, dan I2C yang disediakan oleh u-blox mempermudah proses penghubungan dengan perangkat lain. Untuk proyek skripsi, hal ini memungkinkan penggunaan sensor GPS dengan berbagai platform, seperti pemrograman dengan Python, C++, atau Node.js untuk mengelola data lokasi.

Modul u-blox dibangun dengan teknologi GPS dan GNSS (Global Navigation Satellite System) yang canggih, memungkinkan pengguna untuk menerima sinyal dari berbagai sistem satelit seperti GPS, GLONASS, Galileo, dan BeiDou. Keuntungan ini membuat modul u-blox lebih akurat dan lebih cepat dalam menentukan posisi, terutama di lingkungan yang sulit seperti perkotaan dengan banyak gedung tinggi. Multi-GNSS: Mendukung berbagai sistem satelit untuk meningkatkan akurasi dan ketersediaan sinyal. Akurasi Tinggi Modul seperti ZED-F9P dapat mencapai tingkat akurasi hingga beberapa sentimeter melalui teknologi Real-Time Kinematic (RTK). Ideal untuk aplikasi berbasis baterai karena konsumsi daya yang rendah. Kemampuan Navigasi Dinamis: Mampu memberikan data yang stabil bahkan ketika objek bergerak cepat, seperti dalam pelacakan kendaraan atau drone. Kemampuan *Dead Reckoning* beberapa modul mendukung navigasi meskipun sinyal GPS hilang sementara, dengan menggunakan sensor inersia untuk menghitung posisi berdasarkan gerakan terakhir. Data yang dihasilkan dari modul GPS u-blox bisa dimanfaatkan untuk berbagai analisis dalam skripsi atau penelitian. Contoh data yang sering digunakan meliputi Posisi (Latitude, Longitude) menentukan lokasi objek pada permukaan bumi. Kecepatan menentukan kecepatan gerakan objek, misalnya kendaraan. Waktu memberikan waktu presisi yang disinkronkan dengan waktu satelit. Elevasi menentukan ketinggian objek dari permukaan laut, berguna dalam penelitian yang melibatkan topografi atau pemantauan lingkungan.

Selain itu, sensor ini dapat diintegrasikan dengan sensor lain seperti akselerometer, giroskop, atau magnetometer untuk membangun sistem navigasi yang lebih

kompleks, yang bisa sangat berguna dalam aplikasi seperti kendaraan otonom atau robotika. Modul GPS u-blox merupakan pilihan ideal untuk proyek akademik, khususnya skripsi yang melibatkan pelacakan lokasi dan navigasi. Keandalannya dalam memberikan data yang akurat, serta kemudahan integrasi dengan sistem lainnya, membuatnya banyak digunakan dalam berbagai bidang penelitian seperti robotika, transportasi, pemetaan, dan survei lapangan. Penggunaan modul ini dalam skripsi tidak hanya menawarkan keunggulan teknis, tetapi juga menyediakan data penting yang relevan untuk analisis geospasial dan pengembangan teknologi berbasis lokasi.

2.11 LCD OLED

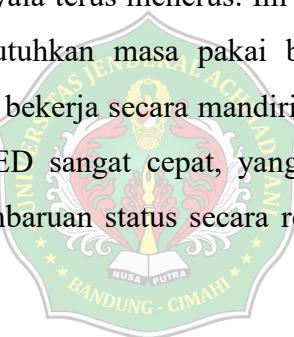
LCD OLED (Organic Light Emitting Diode) adalah jenis layar yang semakin populer dalam aplikasi mikrokontroler, termasuk ESP32, karena kemampuannya menampilkan informasi dengan visual yang jelas dan terang. OLED adalah layar self-emissive, yang berarti setiap piksel menghasilkan cahayanya sendiri tanpa memerlukan lampu latar, berbeda dengan layar LCD konvensional. Hal ini memungkinkan OLED menghasilkan kontras yang sangat tinggi, di mana warna hitam benar-benar hitam, dan konsumsi daya yang lebih rendah, terutama saat menampilkan gambar dengan banyak area gelap. Ukuran layar OLED yang sering digunakan pada proyek mikrokontroler berkisar antara 0,96 hingga 1,3 inci, dengan resolusi umum seperti 128x64 piksel



Gambar 2.9 LCD OLED

Salah satu keunggulan utama OLED adalah kemudahan integrasinya dengan mikrokontroler seperti ESP32 melalui antarmuka komunikasi I2C atau SPI. Banyak modul OLED yang mendukung protokol komunikasi ini, sehingga memudahkan pengembang untuk menghubungkannya hanya dengan beberapa pin. Dalam proyek ESP32, OLED sering digunakan untuk menampilkan data sensor, status sistem, atau bahkan grafik sederhana. Driver seperti Adafruit SSD1306 dan U8g2 adalah library populer yang mempermudah pengembangan aplikasi tampilan untuk OLED, karena menyediakan API sederhana untuk menggambar teks, garis, bentuk, dan gambar pada layar.

Layar OLED juga dikenal karena efisiensi daya yang baik. Karena setiap piksel memancarkan cahayanya sendiri, layar hanya menggunakan daya untuk piksel yang aktif, sehingga sangat hemat energi dibandingkan dengan LCD yang membutuhkan lampu latar menyala terus menerus. Ini menjadikannya ideal untuk aplikasi portabel yang membutuhkan masa pakai baterai yang lama, seperti perangkat IoT atau sensor yang bekerja secara mandiri dalam jangka waktu lama. Selain itu, waktu respons OLED sangat cepat, yang membuatnya ideal untuk menampilkan animasi atau pembaruan status secara real-time tanpa delay visual yang terlihat.



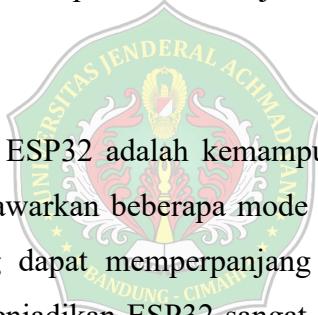
Kemampuan tampilan yang tajam dan jernih juga didukung oleh teknologi OLED yang memungkinkan sudut pandang lebar hingga hampir 180 derajat tanpa distorsi warna atau kehilangan kecerahan. Ini memastikan bahwa informasi yang ditampilkan tetap mudah dibaca meskipun dilihat dari sudut yang tidak biasa. Dalam proyek berbasis ESP32, OLED biasanya digunakan untuk menampilkan informasi sensor, status jaringan, atau kontrol antarmuka sederhana, dan dapat diintegrasikan dengan program-program berbasis event atau real-time monitoring.

2.12 ESP32

ESP32 adalah mikrokontroler yang dikembangkan oleh Espressif Systems, yang terkenal dalam dunia Internet of Things (IoT) karena kemampuannya yang canggih dan serbaguna. ESP32 menawarkan dua inti prosesor Xtensa LX6 yang

dapat beroperasi pada kecepatan hingga 240 MHz, memungkinkan pengolahan data secara cepat dan efisien. Selain itu, ESP32 juga dilengkapi dengan modul Wi-Fi dan Bluetooth terintegrasi, menjadikannya sangat ideal untuk aplikasi IoT yang memerlukan koneksi nirkabel. Kombinasi ini membuat ESP32 menjadi pilihan populer bagi pengembang yang ingin mengimplementasikan proyek dengan kebutuhan komunikasi yang luas.

Selain kemampuannya dalam koneksi, ESP32 juga mendukung berbagai antarmuka dan modul periferal seperti I2C, SPI, UART, PWM, dan ADC/DAC, sehingga bisa digunakan untuk mengontrol sensor, aktuator, atau berbagai perangkat elektronik lainnya. Dengan lebih dari 30 GPIO, ESP32 memungkinkan kontrol yang fleksibel dan konfigurasi beragam. Mikrokontroler ini juga dilengkapi dengan memori RAM sebesar 520 KB dan flash hingga 4 MB, memberikan kapasitas yang cukup untuk menjalankan aplikasi yang lebih kompleks.



Salah satu keunggulan utama ESP32 adalah kemampuannya dalam pengelolaan daya. Mikrokontroler ini menawarkan beberapa mode hemat energi seperti Deep Sleep dan Light Sleep, yang dapat memperpanjang masa pakai baterai pada perangkat portabel. Hal ini menjadikan ESP32 sangat sesuai untuk aplikasi yang membutuhkan konsumsi daya rendah, seperti sensor nirkabel atau perangkat yang bekerja dalam jaringan IoT terdistribusi. Selain itu, dukungan untuk pembaruan firmware melalui Over-the-Air (OTA) membuatnya semakin fleksibel dan mudah dikelola dalam skala besar.

Dengan semua fitur yang ditawarkan, ESP32 memberikan solusi yang kuat dengan harga yang relatif terjangkau. Kemampuan untuk menangani keamanan tingkat tinggi, seperti enkripsi AES, RSA, dan HMAC, membuatnya cocok untuk aplikasi yang memerlukan perlindungan data yang ketat. Selain itu, didukung oleh komunitas pengembang yang besar dan dokumentasi yang komprehensif, pengembangan aplikasi menggunakan ESP32 menjadi lebih mudah dan cepat.

Referensi dan dukungan yang luas menjadikannya salah satu platform pilihan utama dalam pengembangan sistem IoT di berbagai bidang.



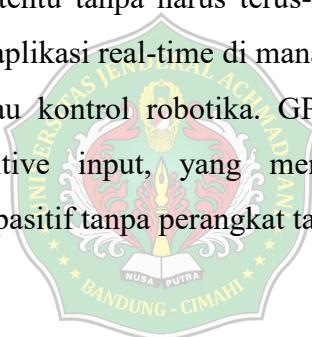
GPIO (General Purpose Input/Output) pada ESP32 merupakan salah satu fitur kunci yang memberikan fleksibilitas tinggi untuk mengontrol perangkat eksternal seperti sensor, aktuator, LED, dan komponen lain. ESP32 dilengkapi dengan lebih dari 30 pin GPIO yang dapat dikonfigurasi baik sebagai input maupun output, memungkinkan interaksi dengan berbagai jenis perangkat keras. Pin-pin ini bisa digunakan untuk membaca sinyal dari sensor atau mengirimkan sinyal untuk mengontrol perangkat seperti relay atau motor. Keunggulan utama dari GPIO pada ESP32 adalah kemampuannya untuk digunakan dalam berbagai mode dan fungsi, termasuk PWM (Pulse Width Modulation), I2C, SPI, dan UART.

Sebagai input, pin GPIO dapat digunakan untuk membaca data dari perangkat eksternal seperti tombol, sensor suhu, atau sensor jarak. Pin ini juga mendukung fitur internal pull-up dan pull-down resistors, yang berguna untuk menjaga sinyal tetap stabil ketika tidak ada input yang terhubung secara langsung. Untuk mendeteksi sinyal digital atau analog, ESP32 menyediakan konverter ADC (Analog to Digital Converter) dengan resolusi hingga 12-bit, yang dapat membaca

tegangan input dan mengonversinya menjadi data digital yang dapat diproses oleh program.

Sebagai output, GPIO pada ESP32 mampu mengontrol berbagai perangkat seperti LED, buzzer, atau relay untuk aplikasi otomatisasi. Dengan dukungan PWM, pin ini dapat menghasilkan sinyal dengan siklus kerja yang bervariasi, yang berguna untuk mengatur kecerahan LED atau kecepatan motor. Selain itu, pin GPIO ESP32 juga bisa dikonfigurasi untuk bekerja dengan protokol komunikasi seperti SPI, I2C, dan UART, sehingga memungkinkan komunikasi antar mikrokontroler atau dengan sensor dan modul lain.

Salah satu keunggulan dari GPIO ESP32 adalah kemampuannya untuk bekerja dalam mode interrupt, yang memungkinkan perangkat untuk segera merespons perubahan sinyal pada pin tertentu tanpa harus terus-menerus memeriksa status pin. Ini sangat berguna dalam aplikasi real-time di mana respons cepat diperlukan, seperti pada sistem alarm atau kontrol robotika. GPIO ESP32 juga memiliki dukungan untuk touch-sensitive input, yang memungkinkan pengembang membuat antarmuka sentuh kapasitif tanpa perangkat tambahan.

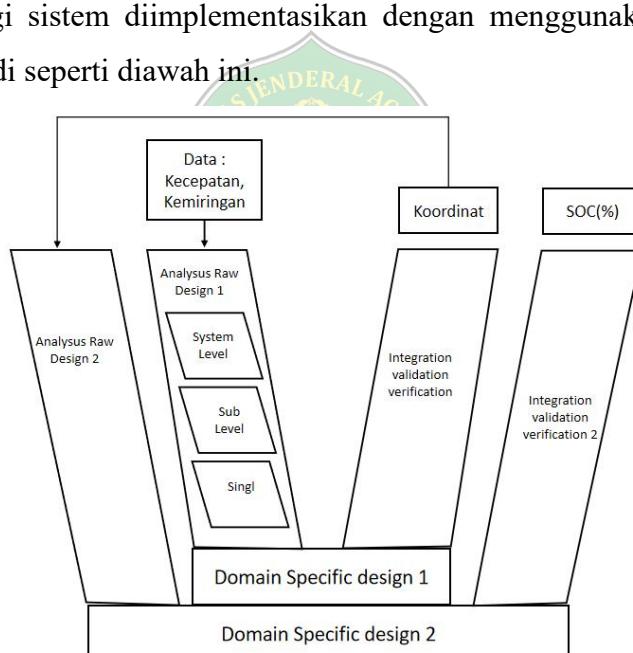


BAB III

METODE PENELITIAN

Metode penelitian mengenai perancangan dan implementasi DAQ berbasis *realtime operating* untuk masukan pemodelan dan simulasi konsumsi baterai kendaraan listrik. Berdasarkan landasan teori dari bab sebelumnya dan latar belakang dari tujuan penelitian ini maka metode yang digunakan pada tugas akhir. Tahapan awal mengumpulkan sumber literatur yang terkait dengan pembahasan yang sedang diteliti. Setelah sumber sudah terkumpul maka dibuatkan untuk sistem pemodelan baik secara matematis maupun secara langsung melalui simulink pada matlab. Setelah sistem tergambar dengan rinci selanjutnya perancangan terkait kebutuhan untuk software dan integrasi pada peralatan. Kedua sistem sudah berjalan dengan baik maka melakukan kolaborasi antara kedua sistem tersebut. Setelah itu monioring sistem dan melakukan *trial and error*.

Apabila metodologi sistem diimplementasikan dengan menggunakan VD12206 akan menjadi seperti diawah ini.

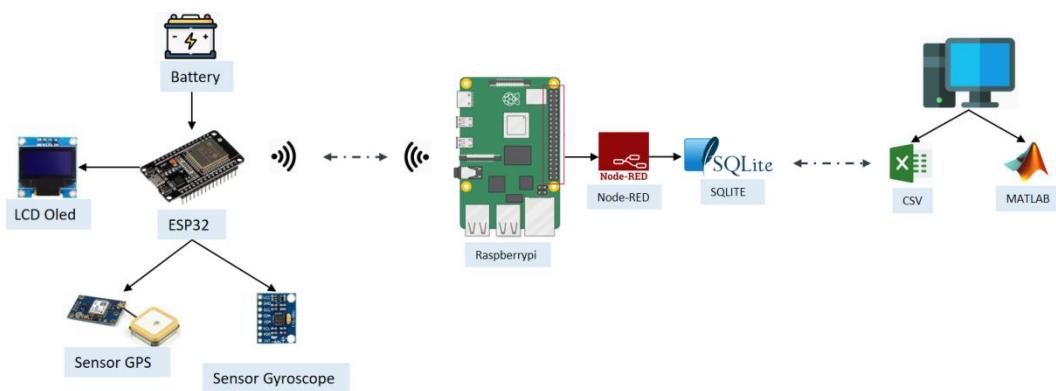


Gambar 3.2 Metodologi VDI2206

3.1 Arsitektur Umum Sistem Perancangan

Secara umum bagian utama arsitektur perancangan terbagi menjadi tiga bagian diantaranya bagian pengambilan data dengan mikrokontroler ESP32 sebagai

kendali untuk sistem. Lalu bagian akuisisi data pada bagian raspberrypi dengan menggunakan database SQLITE melalui sebuah aplikasi Node-RED. Setelah bagian pengambilan data melalui sensor dan data berhasil diambil oleh raspberrypi kedalam database maka data tersebut akan dijadikan sebagai masukan untuk sistem simulasi yang dijalankan pada *software* MATLAB. Simulasi pada MATLAB menggunakan metode simulink untuk mendapatkan pemodelan yang dapat divisualkan. Blok diagram untuk sistem ditunjukkan pada gambar 3.xx.



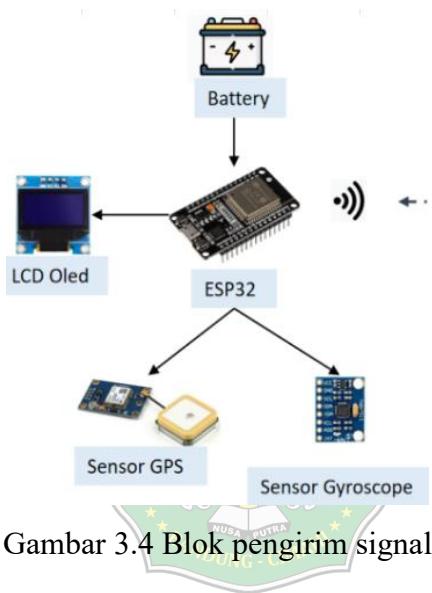
Gambar 3.3 Arsitektur umum sistem perancangan

Sistem yang dijalankan membutuhkan koneksi internet untuk dapat mengakses link MQTT Broker (broker.mqtt-dashboard.com) baik ESP32 maupun raspberrypi membutuhkan koneksi internet. Sensor yang digunakan hanya dapat berfungsi apabila berada diluar ruangan karena keterbatasan pada sensor GPS yang digunakan. Raspberrypi menjadi server utama ketika alat tersebut memberikan atau mempublish sebuah data. Data yang nantinya akan dikirimkan antara lain seperti :

1. Latitude
2. Longitude
3. Altitude
4. Kecepatan
5. Time
6. *Gradien* (Kemiringan jalan)
7. Akselerasi x,y,z
8. Rotasi x,y,z

3.1.1 Blok Pengirim Signal

Pada bagian blok pengirim signal menggunakan esp32 sebagai kontroler utama untuk mengatur segala dari perubahan yang terjadi pada sensor Gyroscope dan sensor GPS. Sensor dan kontroler mendapatkan tegangan sumber sebesar 5 Volt dari tegangan baterai luar menggunakan *powerbank*. Dengna menggunakan fitur wifi yang sudah tersedia secara buildin didalam esp32 mengirimkan data secara *realtime* kedalam server yang terhubung dengan internet. Dengna interface menggunakan LCD Oled sebagai indikator keberhasilan dari sensor yang mengirimkan data kedalam server.

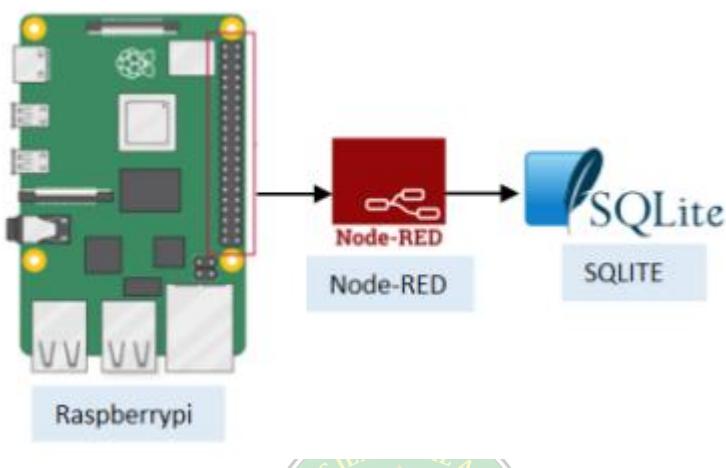


Gambar 3.4 Blok pengirim signal

Besasran-besaran tambahan lain masih dapat diambil oleh sensor dengan bantuan sebuah library pada mikrokontroler ESP32. Semua *sample* data yang diambil memiliki interval waktu masing-masing, tetapi setiap besaran memiliki momen yang sama ketika waktu pengambilan datanya. Metode pengambilan data dengan ESP32 dengan mengambil data dari sensor yang terdiri dari sensor *Gyro* untuk mendapatkan nilai kemiringan dari jalanan, sensor GPS Ublox NEO-6M untuk mendapatkan beberapa parameter koordinat untuk medapatkan posisi latitude dan longitude. Besaran ini tersebut yang nantinya akan digunakan untuk menentukan beberapa parameter. Sensor *gyro* digunakan untuk menentukan kemiringan dari jalan yang dilalui oleh kendaraan listrik.

3.1.2 Blok Penerima data

Pada bagian penerima data menggunakan raspberrypi untuk mengolah data yang sudah dikirimkan dengan protocol MQTT. Data yang dikirimkan diolah menggunakan node-red dan database sqlite sebagai penampung dari datanya. Masing - masing data yang diterima oleh raspi disimpan dalam table dan semua masuk dalam kolom yang sama baik sensor GPS maupun sensor Gyroscope.

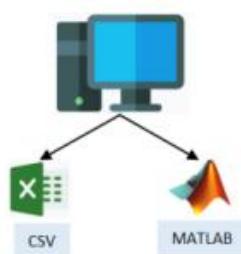


Gambar 3.4 Blok pengirim signal

Raspberrypi sebagai kontrol utama mengolah data dengan bantuan aplikasi yang bernama Node-RED untuk program data akuisisinya dan pemilahan data apa saja yang akan dimasukan kedalam database SQLite.

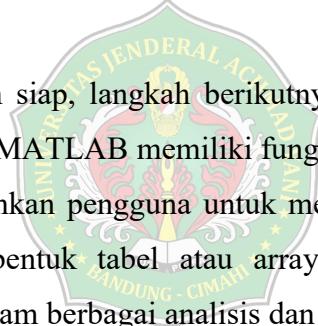
3.1.3 Blok Simulasi Dengan MATLAB

Data yang sudah disimpan dalam database lalu dimasukan kedalam simulasi matlab dengan format dasar csv. Data yang dimasukan menghubungkan antara pengaruh dari jarak dan kemiringan yang ditempuh oleh kendaraan mobil listrik terhadap konsumsi energi listrik.



Data yang diperlukan untuk simulasi ini dikumpulkan dari berbagai sensor dan perangkat pengukuran yang dipasang pada kendaraan mobil listrik. Data ini mencakup informasi tentang jarak yang ditempuh dan kemiringan jalan yang dilalui. Setelah data dikumpulkan, data tersebut disimpan dalam database untuk memudahkan pengelolaan dan aksesibilitas. Format penyimpanan yang umum digunakan adalah CSV (Comma-Separated Values), karena format ini sederhana dan mudah diintegrasikan dengan berbagai aplikasi analisis data.

Setelah data disimpan dalam database, langkah selanjutnya adalah mengekspor data tersebut ke dalam format CSV. Proses ini melibatkan pemilihan data yang relevan dari database dan menyimpannya dalam file CSV. File CSV ini akan berisi kolom-kolom yang mewakili berbagai parameter seperti jarak, kemiringan, dan konsumsi energi listrik. Setiap baris dalam file CSV mewakili satu set data yang diukur pada waktu tertentu.



Dengan file CSV yang sudah siap, langkah berikutnya adalah mengimpor data tersebut ke dalam MATLAB. MATLAB memiliki fungsi bawaan seperti `readtable` atau `csvread` yang memungkinkan pengguna untuk membaca data dari file CSV dan menyimpannya dalam bentuk tabel atau array. Data yang diimpor ini kemudian dapat digunakan dalam berbagai analisis dan simulasi.

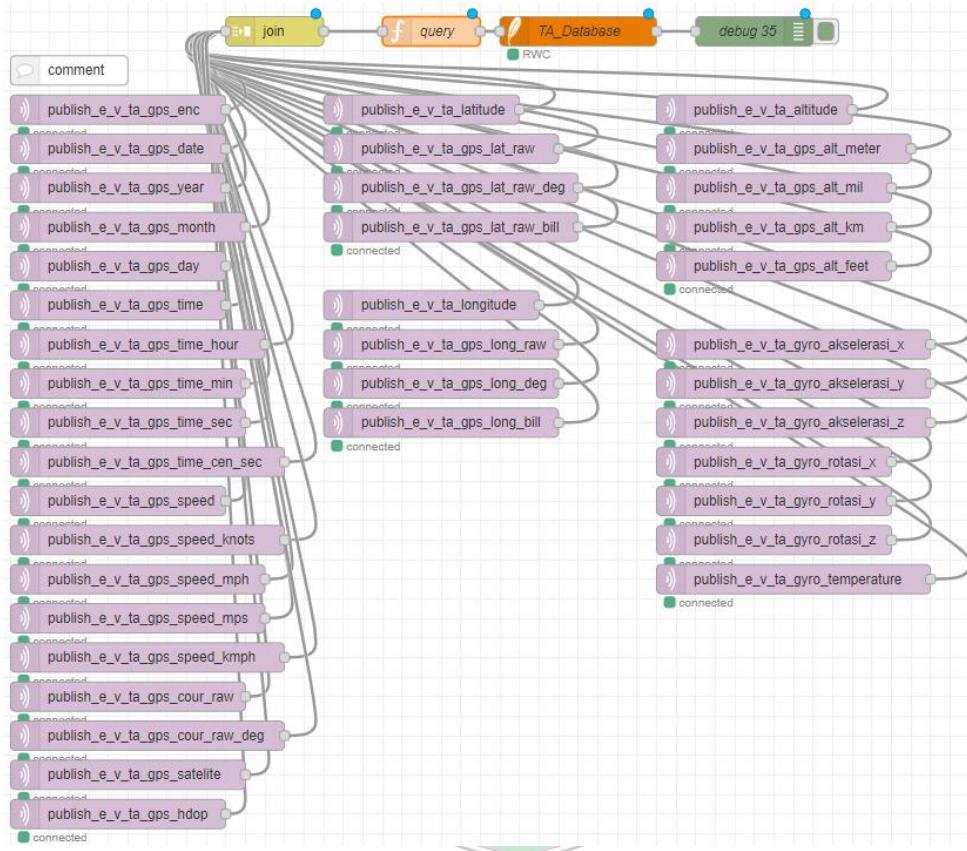
Dalam simulasi MATLAB, data yang diimpor digunakan untuk menganalisis bagaimana jarak dan kemiringan jalan mempengaruhi konsumsi energi listrik pada kendaraan mobil listrik. Simulasi ini dapat melibatkan pembuatan model matematis yang menggambarkan hubungan antara variabel-variabel tersebut. Misalnya, model dapat menunjukkan bahwa konsumsi energi meningkat seiring dengan peningkatan jarak dan kemiringan jalan.

Setelah simulasi dijalankan, hasilnya dianalisis untuk mendapatkan wawasan tentang efisiensi energi kendaraan. Analisis ini dapat mencakup pembuatan grafik yang menunjukkan hubungan antara jarak, kemiringan, dan konsumsi energi.

Hasil analisis ini dapat digunakan untuk mengoptimalkan desain kendaraan dan strategi pengemudian, sehingga konsumsi energi dapat diminimalkan.

3.2 Perancangan Software Akuisisi Data

3.2.1 Node-red



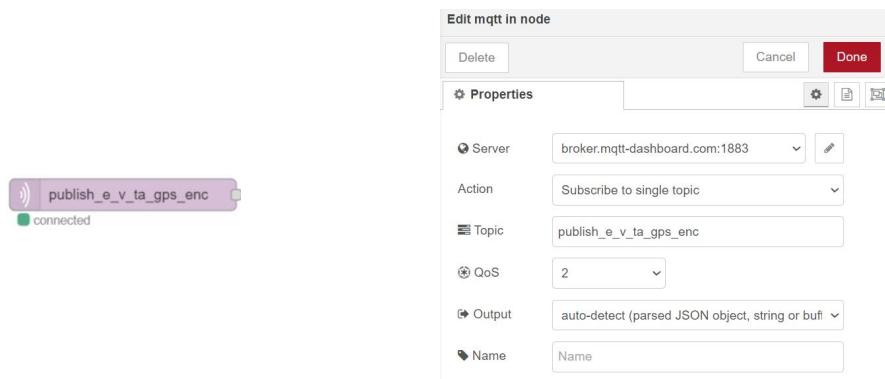
Gambar 3.5 Node akuisisi data Node-RED

Node dalam Node-RED memiliki keunikan dan fungsinya masing-masing. *Node MQTT broker* memiliki fungsi untuk menghubungkan beberapa sistem yang ada di plant kedalam sebuah server. Plant dalam hal ini bisa berupa *publisher* data yang dikirimkan oleh ESP32 atau *subscriber* yang dikendalikan oleh ESP32. *Node join* memiliki fungsi untuk mengumpulkan semua data dari *MQTT broker* menjadi satu dalam tipe data JSON(JavaScript Object Notation). *Node SQLITE* memiliki fungsi untuk mengolah JSON yang diberikan dan menjalankan query agar dapat memasukan data kedalam database.



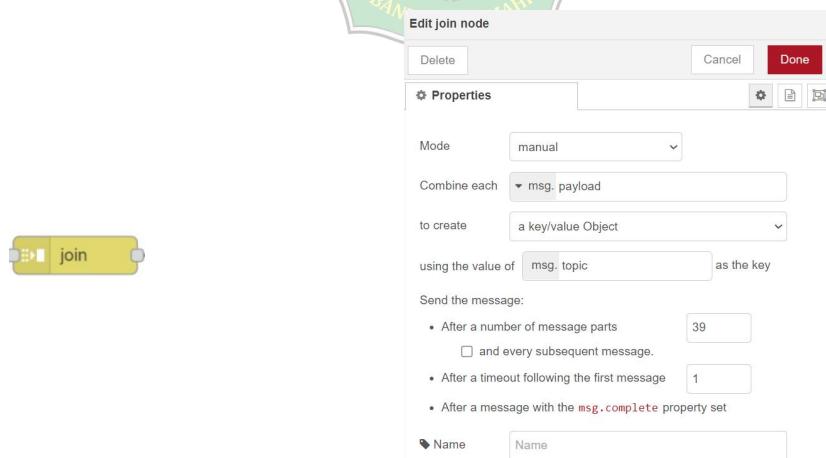
Gambar 3.6 Flow process dari MQTT sampai database

Node-RED diinstal di dalam Raspberrypi dan dijalankan didalam sebuah web browser. Untuk menjalankannya melalui sebuah perintah node-red didalam terminal lalu membuka nya didalam link localhost:1880.



Gambar 3.7 Flow process dari node MQTT

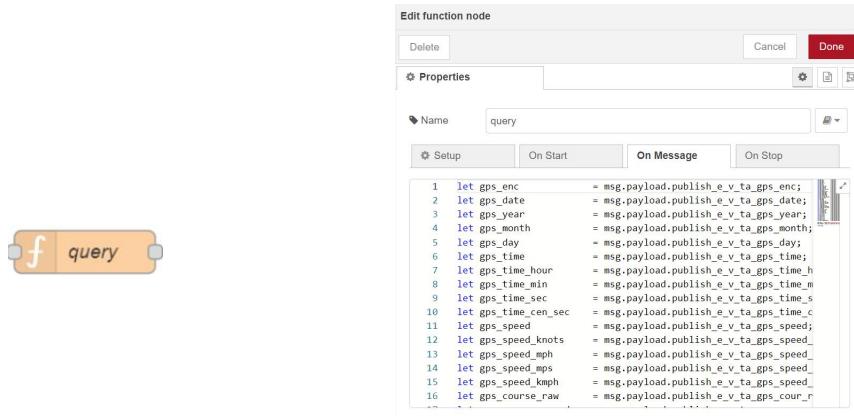
Proses pertama pada node MQTT dilakukan proses pembacaan untuk *subscribe* sebuah topic yang sudah di *publish* oleh ESP32 dan setiap topic memiliki penyimpanan yang unik agar tidak tertukar dengan proses *publish* yang lain. Karena node MQTT yang digunakan tidak hanya satu. Setiap node yang digunakan menampung *value* dari masing-masing besaran.



Gambar 3.8 Proses node Joint

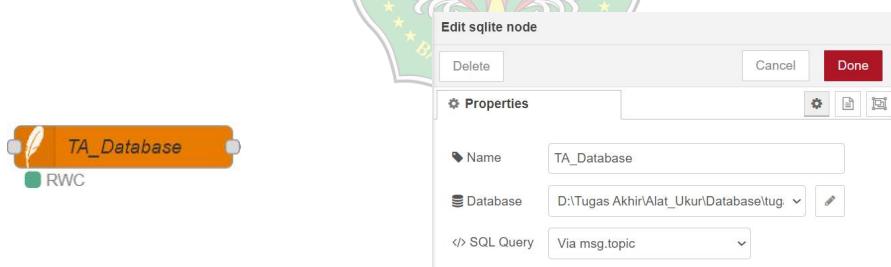
Node join berfungsi untuk menggabungkan *value* yang dimasukan sebagai inputnya. Input yang dapat dimasukan memiliki topic sebagai key untuk identitas dari masing-masing variabel. Variabel yang ada sudah didefinisikan sebelumnya

oleh mikrokontroler ESP32. Data hasil olahan node ini bisa berupa JSON atau sebuah objek untuk memudahkan ketika proses query.



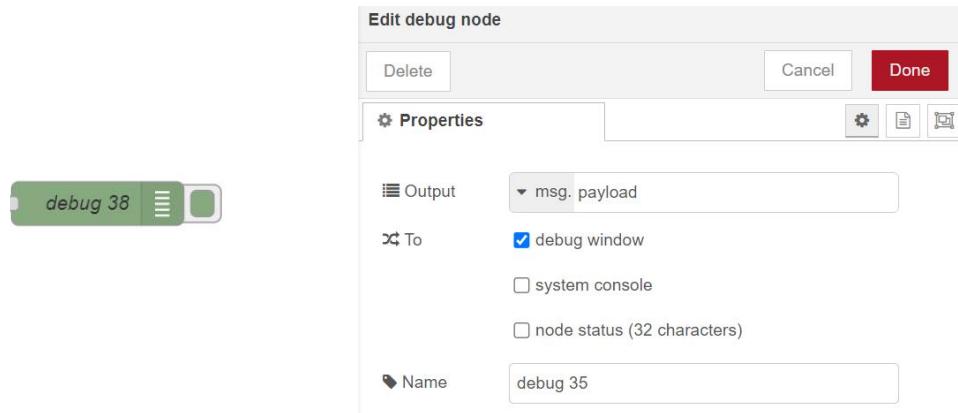
Gambar 3.9 Proses node function query

Node function memiliki fungsi untuk membuat sebuah sintak yang memiliki bahasa program Javasctipt. Bahasa ini tempat algoritma pemilihan dan query untuk memasukan nilai pada database. Proses masukan data pada database dilakukan dengan sebuah sintak pada node *function* tersebut. Dalam sintak mengikuti kaidan algoritma secara umum diantaranya ada bagian deklarasi, bagian *setup*, bagian body sintak.



Gambar 3.10 Proses node database SQLITE

Node database SQLITE untuk menghubungkan antara sintak pada node function Node-RED dengan database. Dalam hal ini harus didefinisikan dimana letak folder database tersebut disimpan. Pada bagian database alamat yang disetting seperti (/home/pi/database/tugas_akhir.db) database SQLITE harus menempati sebuah folder yang menjadi acuan untuk *environment* pada pemrograman.



Gambar 3.11 Proses node debug

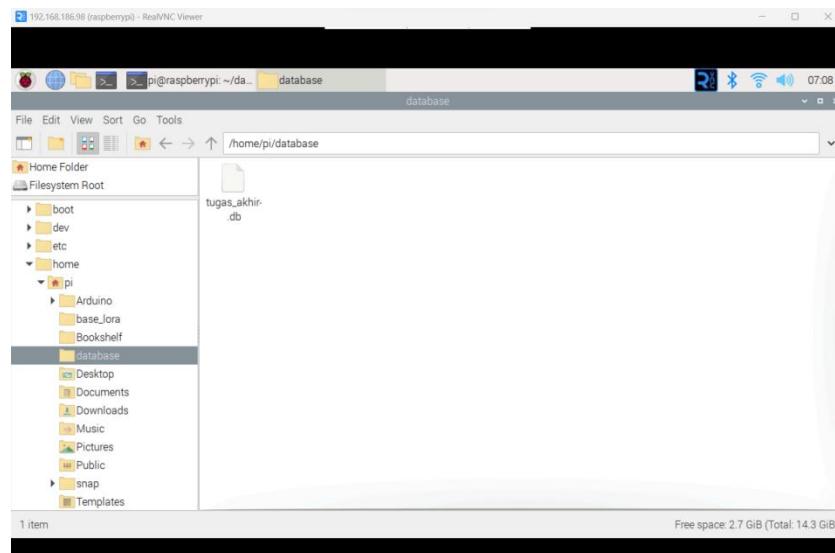
Bagian node debug berfungsi untuk menampilkan pesan dari msg.payload. Apabila ada error maka pesan ini akan ditampilkan sebagai indikator dan dimana lokasi node *error* tersebut berada. Sistem debug menampilkan data yang bisa dilihat pada jendela *sidebar*. Node debug ini memudahkan untuk melacak letak error dengan menyusuri sumber kesalahan nya, pesan yang ditampilkan akan berwarna biru apabila pesan yang disampaikan benar, sedangkan pesan akan berwarna merah apabila pesan yang disampaikan keliru.

3.2.2 Database SQLITE

Gambar 3.12 Database SQLite

Database SQLITE memiliki fungsi untuk menampung data yang diterima oleh MQTT *broker*. Beberapa data yang digunakan seperti kecepatan, longitude dan latitdu. Data yang diterima akan dimasukan kedalam sistem simulasi dan dilakukan pengolahan data untuk mencari hubungan antara konsumsi batrai terhadap kecepatan dan kemiringan jalan. CRUD(*Create Read Update Delete*)

dapat diimplementasikan untuk database SQLITE karena bahasa *query* yang digunakan sama. Variabel yang akan dimasukan kedalam sistem CRUD seperti latitude, longitude, altitude dari sensor GPS dan beberapa besaran akselerasi koordinat kartesius. Nilai dari setiap variabel tersebut akan tersimpan didalam folder raspi.

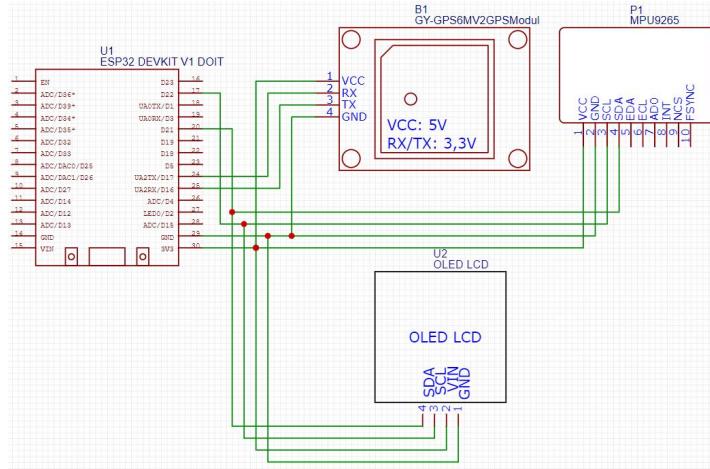


Gambar 3.13 lokasi penyimpanan database

Untuk dapat mengakses database yang sudah tersimpan harus menggunakan sebuah bahasa program yang dapat menjalankan instruksi *query language*. Semua sintak CRUD dijalankan menggunakan bahasa program JavaScript melalui Node-RED.

3.3 Perancangan Alat Akuisisi Data

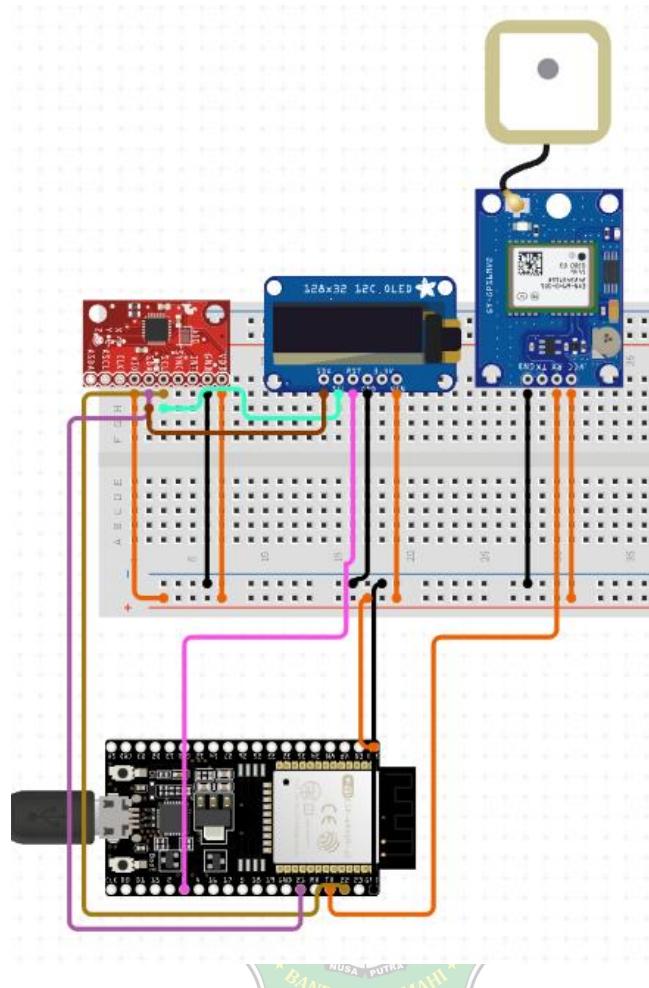
3.3.1 Skematik Rangkaian



Gambar 3.14 Skematik Rangkaian DAQ

Skematik berupa modul modul komponen elektronika yang sudah tertanam, sehingga dalam penggunaannya relatif lebih mudah karena sudah tersedia pin-pin untuk catu daya, ground, dan sinyal data. Perlu dibuatkan rangkaian skematik dengan tujuan membuat gambaran dan sketsa awal wiring untuk sistem elektrik. Setelah proses wiring skematik selesai maka selanjutnya tahapan pembuatan board PCB . PCB memudahkan dalam menyimpan dan menjaga ke stabilan sistem karena wiring yang sudah kokoh tidak ada jumper kabel untuk koneksinya. Sumber tegangan yang digunakan pada sistem skematik

3.3.2 Perancangan Hardware



Gambar 3.15 Rancangan Komponen Hardware

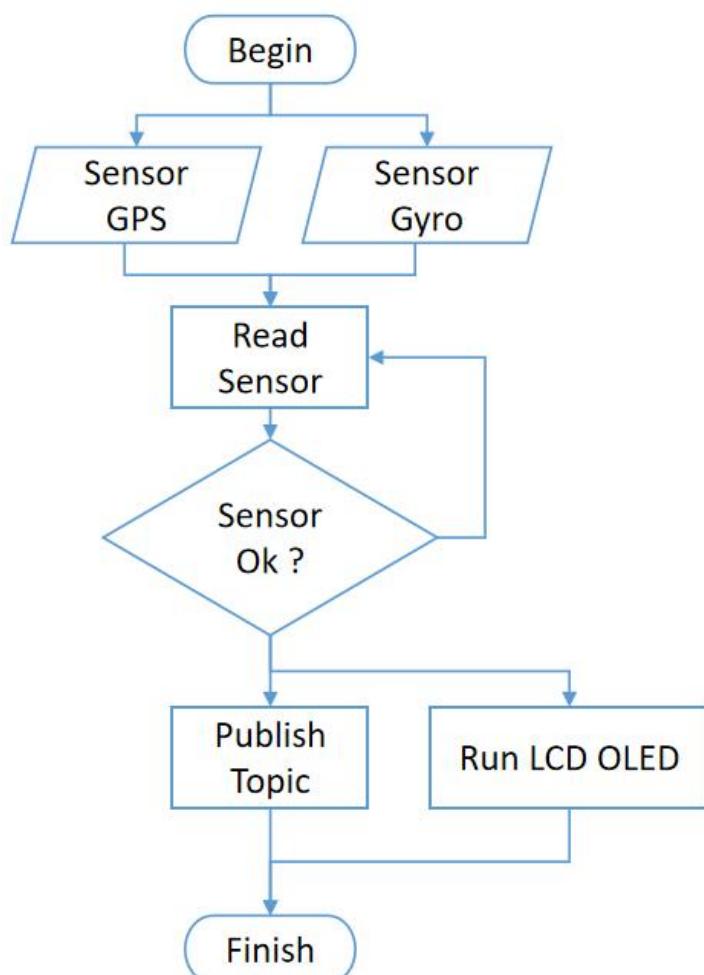
Rangkaian elektrik yang dihubungkan satu sama lain dengan tampilan 2D untuk memudahkan pemahaman terkait metode pengambilan data melalui sensor GPS. Hardware yang digunakan meliputi mikrokontroler ESP32 dan beberapa sensor dengan satu interface yaitu LCD OLED. Sensor Gyro menggunakan pin SDA, SCL untuk metode pengambilan data dengan mikrokontrolernya dan LCD OLED juga sama menggunakan SDA, SCL untuk dapat menampilkan grafik yang sudah diprogram sesuai dengan kebutuhan. Sensor GPS menggunakan metode serial komunikasi dalam hal ini menggunakan pin RX dan pin TX.

3.3.4 Perancangan Sensor Gyroscope



Gambar 3.16 Perancangan Sensor Gyroscope

Dalam perancangan pengambilan datanya sensor Gyroscope menggunakan pin 5Vdc dan GND. Sedangkan dalam pengambilan datanya menggunakan pin RX TX yang dihubungkan dengan pin ESP32. Sensor *Gyroscope* memiliki fungsi untuk mendapatkan signal kemiringan dari kendaraan. Dengan bantuan sensor ini dapat memudahkan untuk mendapatkan signal kemiringan jalan. Diagram alir untuk sistem tersebut seperti ditunjukkan pada gambar dibawah.

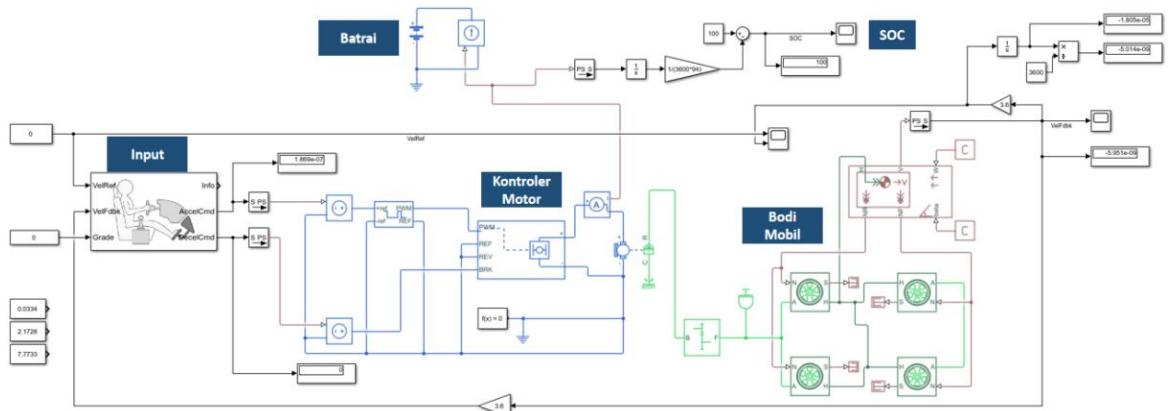


Gambar 3.16 Diagram alir pengambilan data

Peroses pertama ketika sistem berjalan akan membaca masukan dari sensor GPS dan sensor *gyroscope*. Sensor GPS dan *gyroscope* berjalan bersamaan dan ketika proses pembacaan berhasil maka sistem akan mempublish variabel-variabel yang telah di deklarasikan. Proses deklarasi bersamaan dengan ditampilkan nya besaran tersebut pada interface LCD OLED.

3.4 Perencanaan Simulasi Kendaraan Listrik

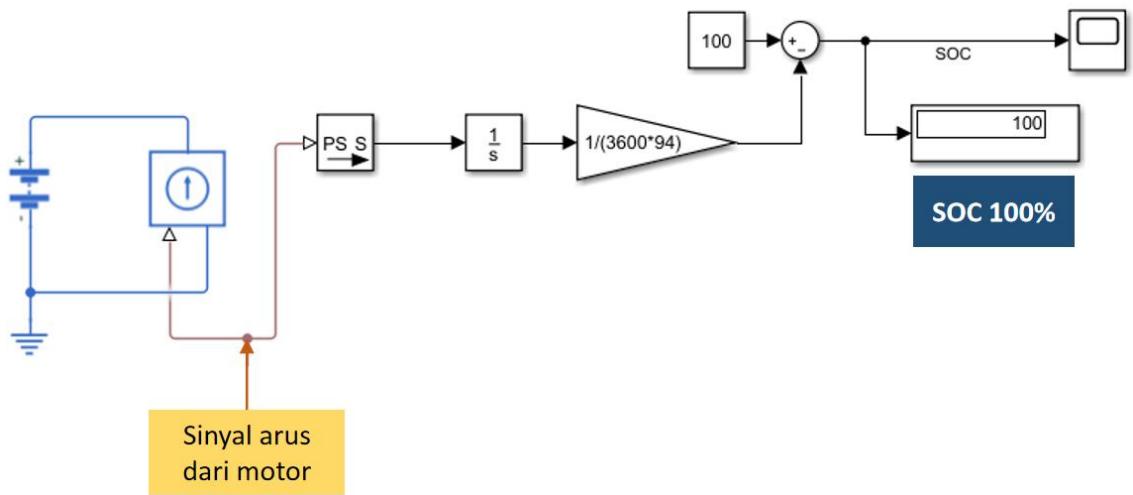
Simulasi kendaraan listrik menggunakan metode simulink dan terdiri dari beberapa bagian sub sistem seperti bagian sistem batrai, *longitudinal driver*, kontroler motor, bodi mobil dan bagian wheel. Keseluruhan sistem simulasi ditunjukan pada Gambar 3.17.



Gambar 3.17 Simulasi mobil listrik dengan simulink

Nilai masukan yang dijadikan parameter menggunakan blok constant dengan nilai yang sebelumnya sudah didapatkan pada proses DAQ. Semua sistem saling terhubung menjadi satu bagian untuk mendapatkan nilai SOC dengan inputan dari profile jalan.

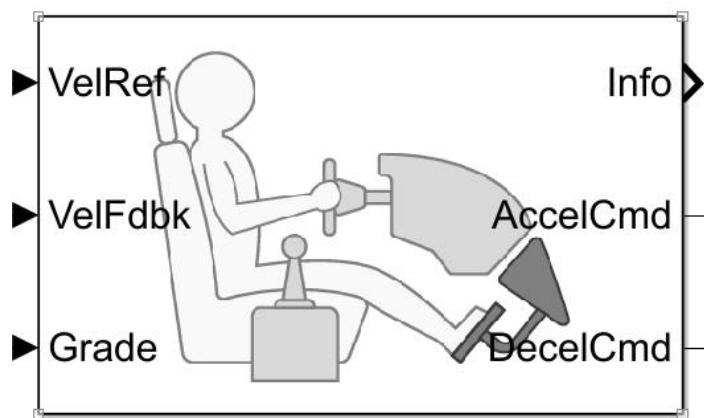
3.4.1 Perancangan Model Batrai



Gambar 3.18 Perancangan Model Batrai

Pada blok perancangan model batrai ketika beroperasi membutuhkan sinyal dari arus motor penggerak utama. Sinyal dari simulasi batrai harus dikonversi dari PS-Simulink agar dapat dilakukan proses aritmatika dan konversi nilai SOC. Sinyal arus akan melalui proses integrasi dan operasi aritmatika untuk mendapatkan nilai sebenarnya dari SOC. Dalam gambar di atas apabila sinyal arus yang dibawa oleh moto sebesar 0A maka nilai dari SOC menjadi 100%. nilai SOC ketika 100% menunjukkan kondisi batrai dalam keadaan penuh.

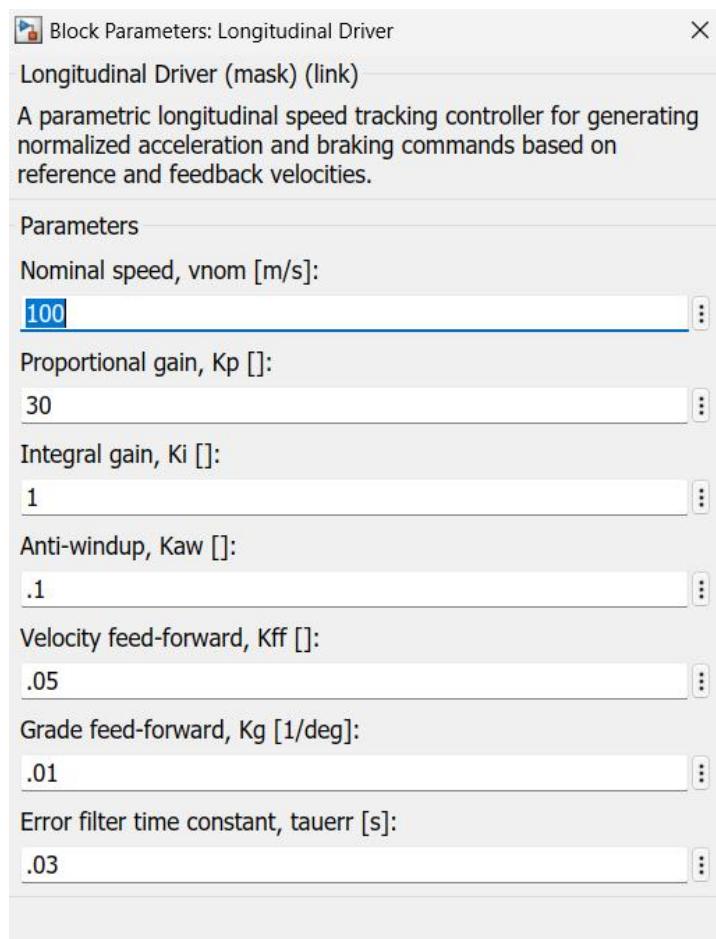
3.4.2 Perancangan Model Input



Gambar 3.19 Perancangan Model Input Longitudinal Driver

Parameter yang dijadikan inputan untuk simulasi kendaraan listrik terdiri dari 3 bagian diantaranya bagian VelRef untuk menentukan setpoint dari kecepatan yang

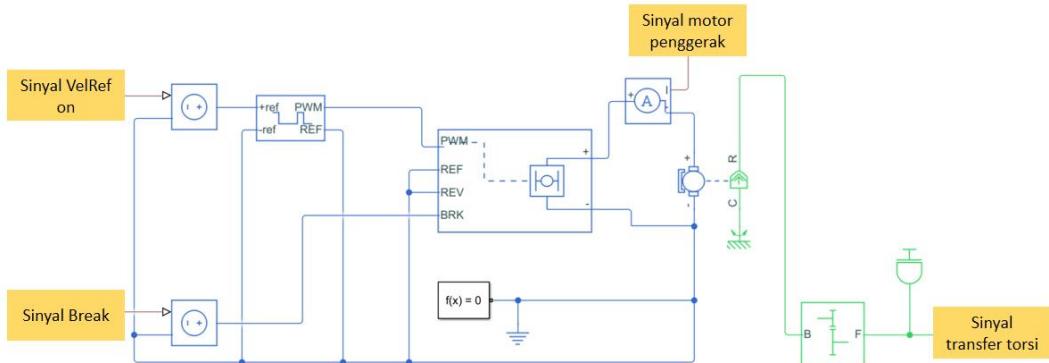
ditentukan. Variabel VelFdbk merupakan variabel kecepatan dari umpan balik sistem untuk variabel pembanding nilai *error*. Kemiringan jalan dinotasikan dengan Grade sebagai variabel yang menampung data dari profile kemiringan jalan dengan satuan derajat.



Gambar 3.20 Spesifik data longitudinal driver

Parameter kecepatan menggunakan satuan m/s karena dalam pengambilan data relatif tidak terlalu cepat, sehingga besaran kecepatan akan lebih mudah untuk dianalisa ketika menggunakan satuan m/s.

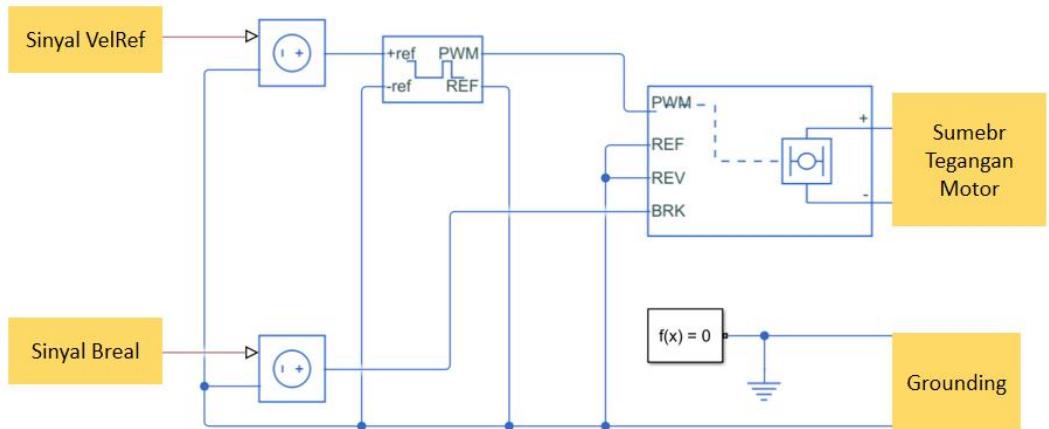
3.4.3 Perancangan Model Penggerak



Gambar 3.21 Model sistem penggerak

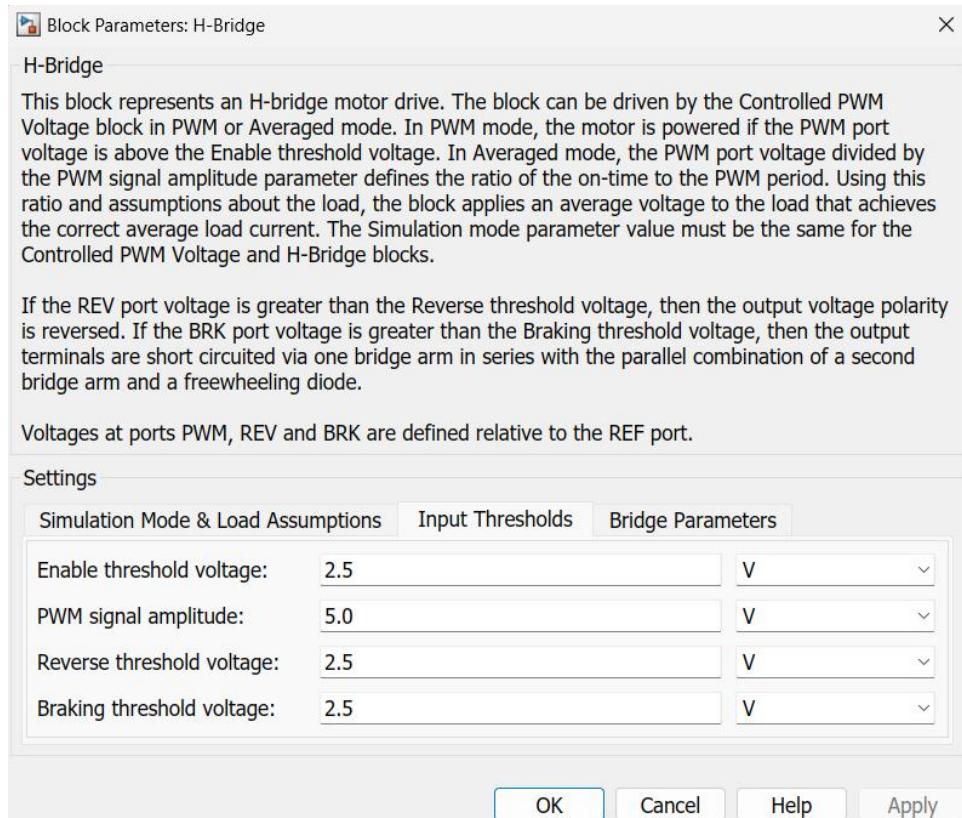
Tiga bagian utama dalam sistem simulasi ada pada bagian penerima sinyal dari input sebagai kondisi untuk PWM. Bagian kontroler utama *duty cycle* untuk kontrol kecepatan pada bagian blok PWM dan yang terakhir pada bagian motor untuk menggerakan roda mobi. Dalam transfer pergerakan menggunakan blok gear yang dihubungkan dengan motor DC dan roda kendaraan listrik.

3.4.4 Perancangan Model Kontroler Motor



Gambar 3.22 Model sistem penggerak

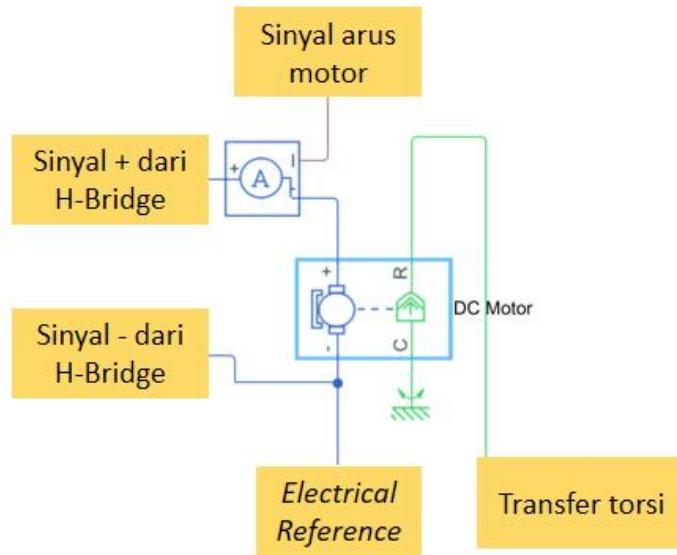
Sinyal VelRef memiliki dua kondisi yaitu kondisi on dan kondisi off. Masing-masing kondisi ditentukan dari nilai input yang sebelumnya diberikan, sehingga apabila nilai masukan bernilai logika atau off sistem kontroler motor tidak akan membuka sumber tegangan untuk motor.



Gambar 3.23 Model sistem penggerak

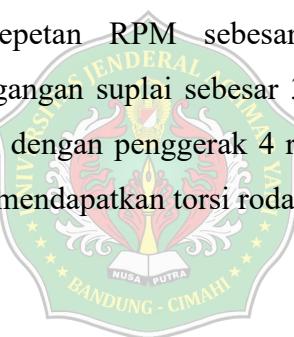
Sinyal H-Brigde beroperasi dalam tegangan 5V didalam kontrolernya. Sistem berjalan dengan menggunakan prinsip pengkondisian switch. Dengan menggunakan blok H-Brigde memungkinkan motor memiliki kondisi dipercepat, dierlambat, dan berhenti. Module signal H-bridge sebagai kontroler utama pada motor berfungsi untuk mengatur segala kondisi dari roda mobil dalam hal ini dapat mengatur kecepatan putaran dan arah putaran dari mobil. Switch yang mengatur perubahan dari putaran roda mobil berjalan kearah kiri secara terus menerus atau mobil berjalan kearah kana secara terus menerus.

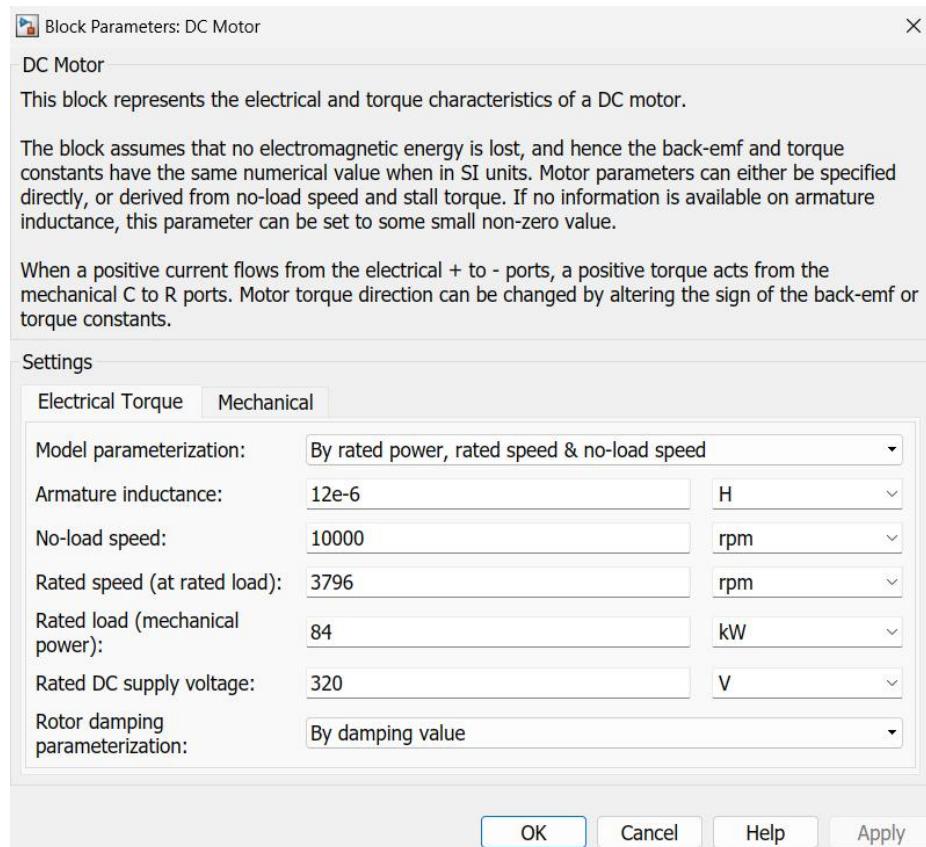
3.4.5 Perancangan Model dan Motor



Gambar 3.24 Model sistem penggerak

Motor DC memberikan kecepatan RPM sebesar 10.000 rpm kecepatan maksimalnya dengan rentang tegangan suplai sebesar 320 V. Spesifikasi tersebut berdasarkan pada sebuah mobil dengan penggerak 4 roda. Lalu rated DC supply voltage sebesar 320 Volt untuk mendapatkan torsi roda yang besar.

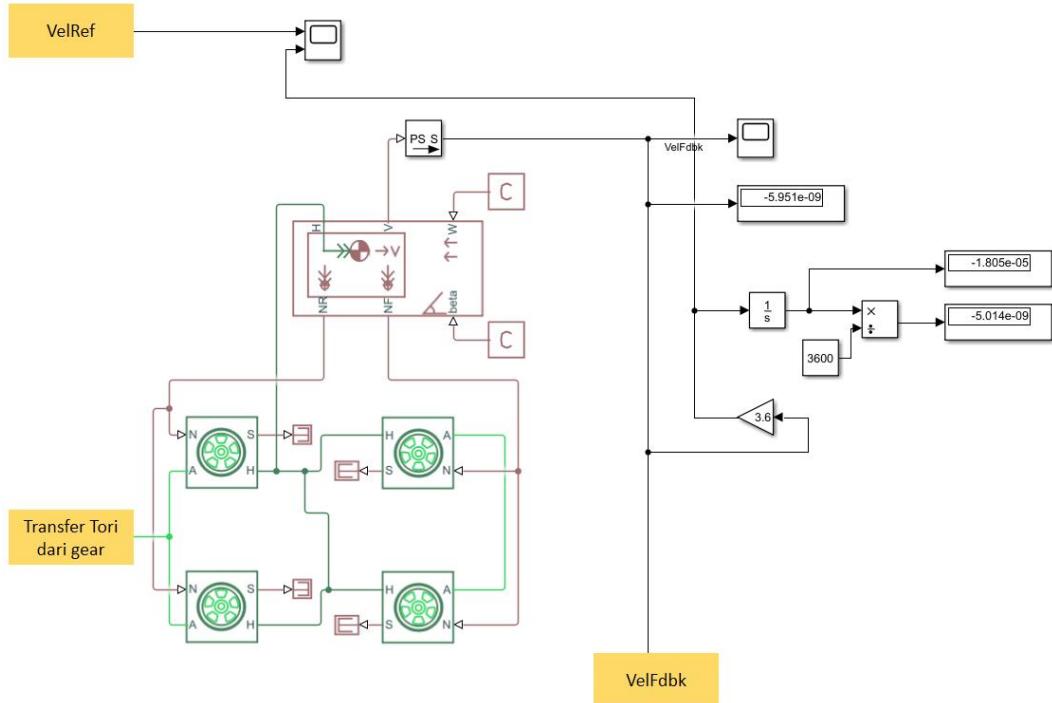




Gambar 3.25 Model sistem penggerak spesifikasi

Area blok ini mendapatkan signal dari H-Bridge untuk memberikan arus pada motor roda mobil. Terjadi perubahan energi dari energi listrik menjadi energi mekanik dalam kasus ini rotasi roda moil.

3.4.6 Perancangan Body dan Wheel



Gambar 3.5 Model sistem bodi dan roda

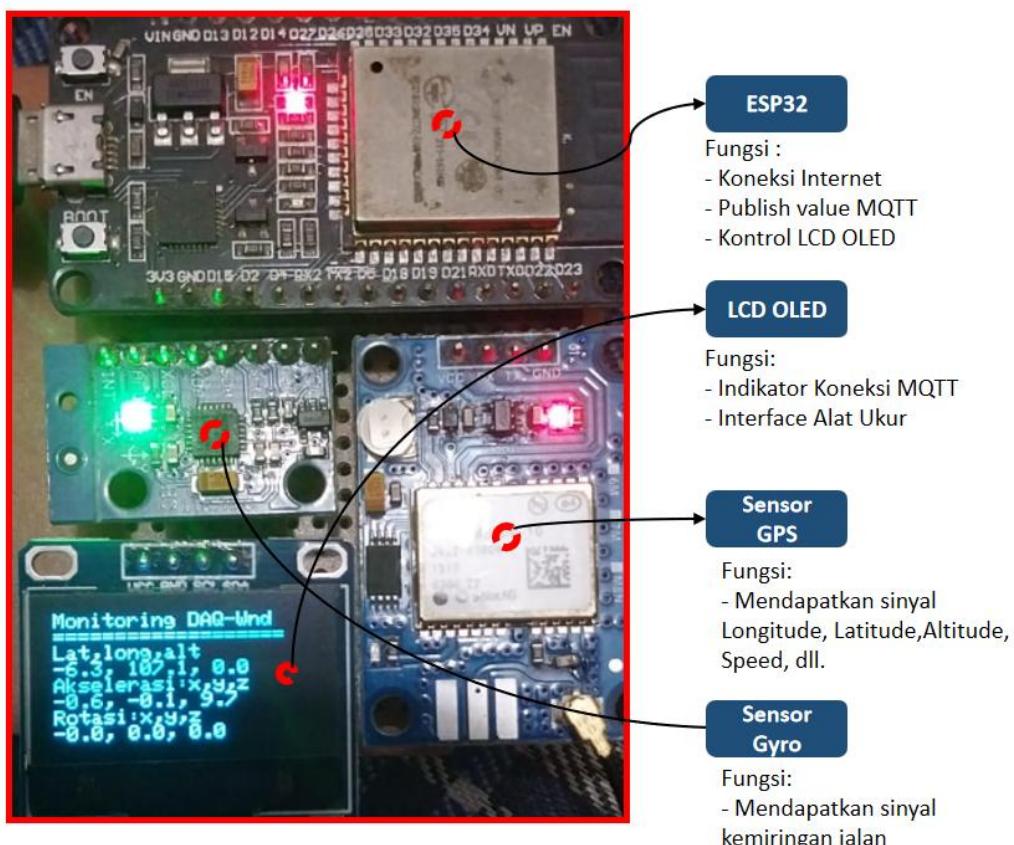
Sistem transfer yang sebelumnya diolah oleh motor DC dijadikan masukan untuk keempat roda yang terhubung dengan bodi mobil. Pengaruh dari berat bodi mobil dan roda menjadi pengaruh terhadap konsumsi baterai. Blok bodi mesin memberikan output kecepatan umpan balik untuk menjadi parameter input pembanding dengan setpoint. Area transfer torsi dari gear pada roda masing-masing keempat bagian memberikan signal untuk seberapa besar torsi yang harus diberikan. Masing-masing roda akan berputar sesuai dengan kondisi yang diperintahkan.

BAB IV

HASIL PENGUJIAN DAN ANALISIS

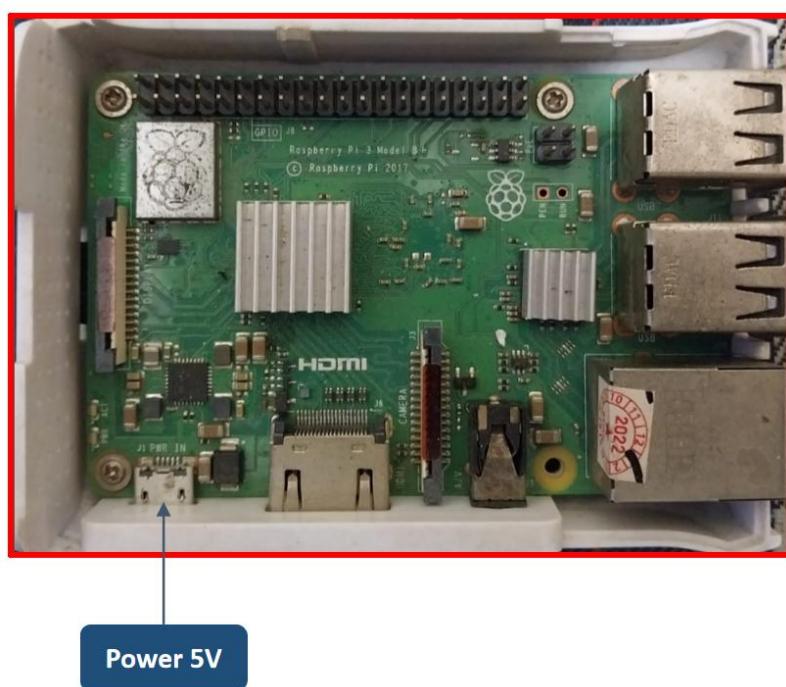
4.1 Hasil Pengujian Alat

Pengujian alat dilakukan secara langsung dijalan dan sumber yang digunakan untuk menyalakan alat menggunakan *powerbank* dengan tegangan 5V DC. Alat tersebut harus mendapatkan koneksi internet ketika beroperasi, sehingga membutuhkan peralatan lain dari luar sebagai sumber internet dari *smartphone*. Alat yang digunakan untuk medapatkan besaran terkait dengan simulasi pada kendaraan listrik simulink MATLAB. Beberapa besaran dapat dilihat secara langsung nilainya sedangkan beberapa nilai masih harus dilakukan pengolahan. Beberapa nilai tersebut seperti kecepatan, kemiringan profil jalan. Kecepatan dapat ditemukan dengan menggunakan besaran longitude dan latitude, sedangkan kemiringan didapatkan dengan menggunakan besaran altitude.



Gambar 4.1 Alat ukur pengambilan data

Esp32 memiliki fungsi untuk mengirimkan data yang terima oleh sensor untuk dapat di upload dengan metode publish kepada MQTT Broker untuk bisa di subscribe dengan device lain yang terkait. Pada Gambar 4.1 ditunjukan LCD OLED sebagai interface untuk menampilkan beberapa besaran yang dapat di ambil. Semua data yang diambil dengan sensor GPS dan sensor Gyro scope.Untuk menentukan sebuah koordinat dapat menggunakan besaran yang sudah diketahui dengan menggunakan bantuan sensor ublox GPS.

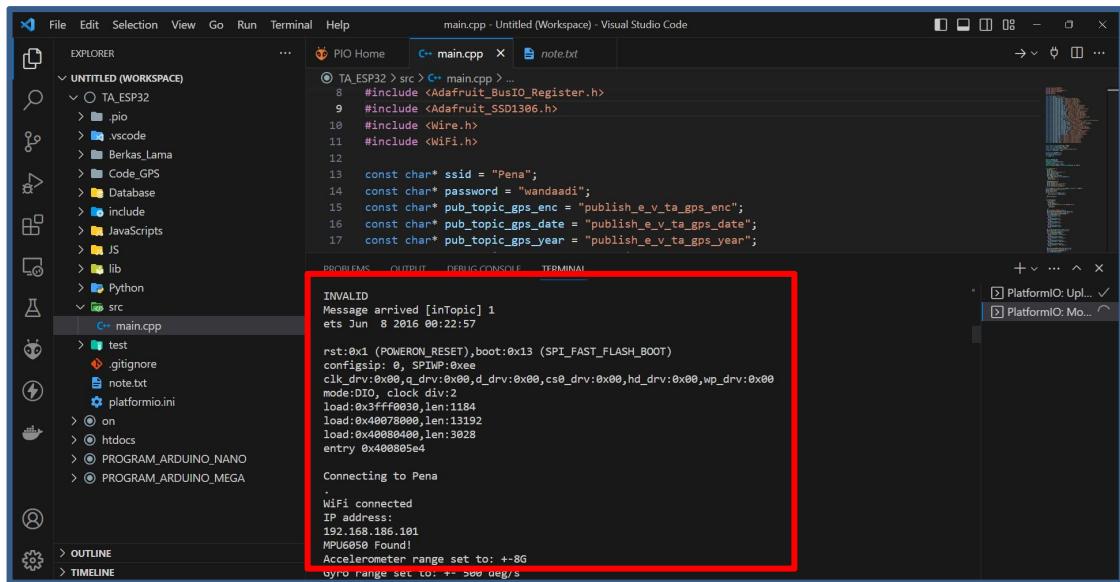


Gambar 4.2 Raspberrypi

Raspberrypi beroperasi dalam tegangan 5V dan memiliki adaptor yang khusus dengan arus 3A. Raspberripy memiliki OS(*Operating System*) dengan tipe *unix* debian sehingga memungkinkan untuk menginstal aplikasi untuk kebutuhan RTOS(*Real Time Operating System*) seperti Node-RED, database SQLITE dan Arduino IDE. Raspberrypi memiliki fungsi untuk akuisisi data dan penyimpanan data didalam database.

4.2 Pengujian Pembacaan Sensor

Tampilan pengujian sensor ditunjukan pada Gambar 4.3 menggunakan software Sisual Studio Code dengan bagian tampilan nya pada *serial monitor*. Baudrate yang digunakan untuk menampilkan data serial adalah 115200 sesuai dengan settingan pada program ESP 32 yang telah diupload pada boardnya.



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure under "UNTITLED (WORKSPACE)".
- main.cpp - Untitled (Workspace) - Visual Studio Code**: The main code editor window.
- TERMINAL**: The serial monitor terminal window.
- Output:**

```
INVALID
Message arrived [inTopic] 1
ets Jun 8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPINP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1184
load:0x40007800,len:13192
load:0x40008840,len:3028
entry 0x40008054

Connecting to Pena
.
WiFi connected
IP address:
192.168.186.101
MPU6050 Found!
Accelerometer range set to: +/- 8G
Gyro range set to: +/- 500 deg/s
```

Gambar 4.3 Pengujian pembacaan sensor

Proses pengujian pembacaan sensor dilakukan dengan menggunakan alat pendekripsi sinyal GPS dan kemiringan jalan. Mendekripsi pembacaan sensor dengan bantuan server dan teks editor (visual studio code). Sinyal dapat terdeteksi didalam terminal *serial monitor* karena ESP32 terhubung dengan server raspberrypi. Dalam proses pengujian nya sensor.

4.2.1 Pengujian ESP32 Dengan Internet

```
Message arrived [inTopic] 1
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1184
load:0x40078000,len:13192
load:0x40080400,len:3028
entry 0x400805e4

Connecting to Pena
.
WiFi connected
IP address:
192.168.186.101
MPU6050 Found!
Accelerometer range set to: +-8G
Gyro range set to: +- 500 deg/s
Filter bandwidth set to: 21 Hz
```

Gambar 4.4 Pengujian pembacaan sensor

Pengujian sensor ESP32 mencoba untuk terhubung dengan koneksi jaringan internet. Apabila berhasil terkoneksi dengan jaringan internet maka program akan menghubungkan ke server MQTT *broker*. Apabila wifi berhasil terkoneksi dengan ESP32 maka akan tampil seperti ditunjukkan pada gambar 4.3 (Wifi connected) dan menampilkan IP *Address* yang terhubung. Sedangkan apabila kondisi alat tidak bisa terkoneksi dengan wifi maka tidak akan menampilkan IP *Address*.

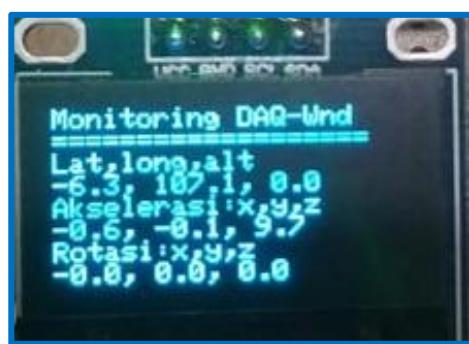
4.2.2 Pengujian Koneksi dengan MQTT Broker

Gambar 4.5 Pengujian pembacaan sensor

Setelah ESP32 berhasil terkoneksi dengan jaringan internet maka program akan berusaha untuk menghubungkan dengan server MQTT broker, sedangkan apabila koneksi tidak berhasil maka serial monitor akan menampilkan pesan (INVALID) dan proses tersebut terus berlangsung sampai ESP32 mendapatkan koneksi dengan MQTT broker.

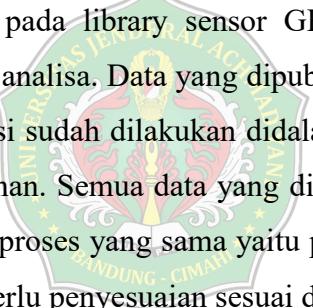
4.2.3 Pengujian Data Publisher ke Server MQTT Broker

Apabila ESP32 berhasil mendapatkan jaringan internet dan menjangkau server MQTT *broker* maka tampilan serial monitor akan menampilkan semua data yang berhasil dipublish seperti ditunjukkan pada Gambar 4.4.



Gambar 4.6 Pengujian pembacaan sensor

Pembacaan nilai sensor pada interface LCD OLED menunjukkan koordinat untuk latitude, longitude dan beberapa besaran dari sensor gyroscope. Hasil pengujian menunjukkan alat dapat berfungsi dengan baik dan dapat melakukan proses DAQ. LCD Oleh menunjukkan perubahan pada nilai dari masing-masing sensor. Longitude, latitude, akselesari x y z, rotasi x, y , z. Dan pembacaan pada sernial monitor ditunjukkan pada Gambar 4.3 dibawah ini.



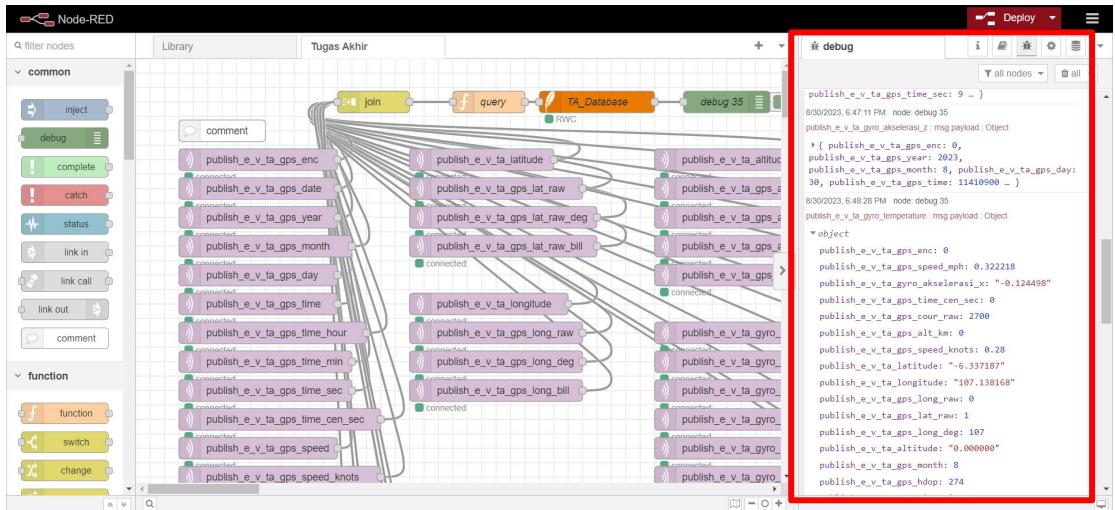
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Publish message gps-time-min: 41.000000
Publish message gps-time-sec: 9.000000
Publish message gps-time-cen-min: 0.000000
Publish message gps-speed: 205.000000
Publish message gps-speed-knots: 2.050000
Publish message gps-speed-mph: 2.359098
Publish message gps-speed-mps: 1.054611
Publish message gps-speed-kmph: 3.796600
Publish message gps-course-raw: 0.000000
Publish message gps-course-raw-deg: 0.000000
Publish message gps-satelite: 4.000000
Publish message gps-hdop: 274.000000
Publish message gps-lat: -6.335319
Publish message gps-lat-raw: 1.000000
Publish message gps-lat-raw-deg: 6.000000
Publish message gps-lat-raw-bill: 335319167.000000
Publish message gps-long: 107.143102
Publish message gps-long-raw: 0.000000
Publish message gps-lat-raw-deg: 107.000000
Publish message gps-lat-raw-bill: 143102000.000000
Publish message gps-alt: 0.000000
Publish message gps-alt-meter: 0.000000
Publish message gps-alt-mil: 0.000000
Publish message gps-alt-km: 0.000000
Publish message gps-alt-feet: 0.000000
Publish message akselerasi-x: -0.153229
Publish message akselerasi-y: 0.816423
Publish message akselerasi-z: 9.665392
Publish message rotasi-x: -0.037571
Publish message rotasi-y: 0.007727
```

Gambar 4.7 Pengujian pembacaan sensor

Semua besaran yang tersedia pada library sensor GPS, *Gyroscope*, dipublish untuk memudahkan data dalam analisa. Data yang dipublish memiliki satuan yang berbeda beda algoritma konversi sudah dilakukan didalam library, sehingga tidak perlu membuat program tambahan. Semua data yang didapatkan dari sensor GPS dan *Gyro* dijadikan dalam satu proses yang sama yaitu pembacaan selama 5 detik satu kali. Beberapa data masih perlu penyesuaian sesuai dengan kebutuhan.

4.3 Komunikasi Dengan MQTT



Gambar 4.8 Aplikasi node-red MQTT broker.

Komunikasi melalui MQTT *broker* dilakukan dengan menggunakan node-red dari sisi server untuk mengumpulkan datanya. Data yang diambil melalui sebuah *node* MQTT yang tersedia didalam node-red. Dengan menyesuaikan dari sisi alamat broker dan key untuk dapat mengakses nilai dari node yang dibawa oleh *subscriber*. Nilai yang diterima oleh *subscriber* bertipe data *string* atau tipe data yang bisa diolah oleh Javascripts.

```

8/30/2023, 6:48:28 PM node: debug 35
publish_e_v_ta_gyro_temperature : msg.payload : Object
  ▼ object
    publish_e_v_ta_gps_enc: 0
    publish_e_v_ta_gps_speed_mph: 0.322218
    publish_e_v_ta_gyro_akselerasi_x: "-0.124498"
    publish_e_v_ta_gps_time_cen_sec: 0
    publish_e_v_ta_gps_cour_raw: 2700
    publish_e_v_ta_gps_alt_km: 0
    publish_e_v_ta_gps_speed_knots: 0.28
    publish_e_v_ta_latitude: "-6.337187"
    publish_e_v_ta_longitude: "107.138168"
    publish_e_v_ta_gps_long_raw: 0
    publish_e_v_ta_gps_lat_raw: 1
    publish_e_v_ta_gps_long_deg: 107
    publish_e_v_ta_altitude: "0.000000"
    publish_e_v_ta_gps_month: 8
    publish_e_v_ta_gps_hdop: 274
  
```

Gambar 4.9 Aplikasi Node-RED MQTT broker.

Masing-masing data berhasil ditampilkan pada *sidebar* proses pengiriman data terjadi delay selama 5 detik. Setiap satu *cycle time* variabel pada object tersebut akan terus diupdate.

```
8/30/2023, 6:47:11 PM node: debug 35
publish_e_v_ta_gyro_akselerasi_z : msg.payload : Object
  ▶ { publish_e_v_ta_gps_enc: 0,
    publish_e_v_ta_gps_year: 2023,
    publish_e_v_ta_gps_month: 8, publish_e_v_ta_gps_day:
    30, publish_e_v_ta_gps_time: 11410900 ... }
```

Gambar 4.10 Aplikasi node-red MQTT broker.

Semua data disimpan dalam tipe data *object*, sehingga masing-masing besaran memiliki *key* untuk dapat diakses. Topik digunakan sebagai *key* ketika proses DAQ dengan aplikasi Node-RED. Karena semua data yang dipublish berupa *string* maka untuk besaran lain perlu penyesuaian seperti tanggal dan waktu. Data subcribe tersebut disimpan dalam database lalu dilakukan

Tabel 4.1 Variabel data *publish*

No	Variabel	Fungsi subscribe
1	gps_enc	msg.payload.publish_e_v_ta_gps_enc
2	gps_date	msg.payload.publish_e_v_ta_gps_date
3	gps_year	msg.payload.publish_e_v_ta_gps_year
4	gps_month	msg.payload.publish_e_v_ta_gps_month
5	gps_day	msg.payload.publish_e_v_ta_gps_day
6	gps_time	msg.payload.publish_e_v_ta_gps_time
7	gps_time_hour	msg.payload.publish_e_v_ta_gps_time_hour
8	gps_time_min	msg.payload.publish_e_v_ta_gps_time_min
9	gps_time_sec	msg.payload.publish_e_v_ta_gps_time_sec

10	gps_time_cen_sec	msg.payload.publish_e_v_ta_gps_time_cen_sec
11	gps_speed	msg.payload.publish_e_v_ta_gps_speed
12	gps_speed_knots	msg.payload.publish_e_v_ta_gps_speed_knots
13	gps_speed_mph	msg.payload.publish_e_v_ta_gps_speed_mph
14	gps_speed_mps	msg.payload.publish_e_v_ta_gps_speed_mps
15	gps_speed_kmph	msg.payload.publish_e_v_ta_gps_speed_kmph
16	gps_course_raw	msg.payload.publish_e_v_ta_gps_course_raw
17	gps_course_raw_deg	msg.payload.publish_e_v_ta_gps_course_raw_deg
18	gps_satelite	msg.payload.publish_e_v_ta_gps_satelite
19	gps_hdop	msg.payload.publish_e_v_ta_gps_hdop
20	latitude	msg.payload.publish_e_v_ta_latitude

4.4 Penyimpanan Data pada Database



DB Browser for SQLite - /home/pi/Tugas_Aakhir/database/ta.db

The screenshot shows the DB Browser for SQLite interface. A red box highlights the first 12 rows of the 'table_data' table, which contains GPS data. The table has columns: id, gps_encode, gps_date, gps_year, gps_month, gps_day, gps_time, gps_time_hour, gps_time_minute, gps_time_second, and gps_time_cen_sec.

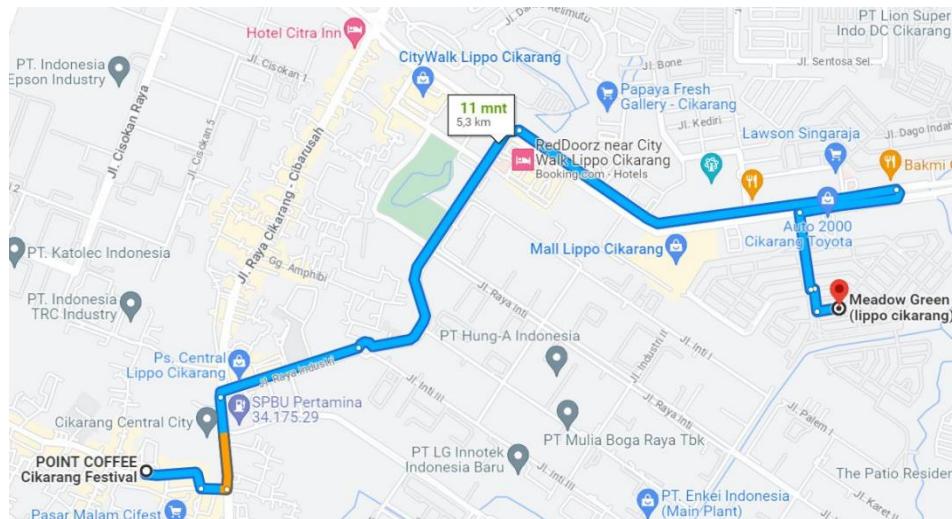
1	1 0	300823	2023	8	30	114109...	11	41	9	0
2	2 0	300823	2023	8	30	114109...	11	41	9	0
3	3 0	300823	2023	8	30	114109...	11	41	9	0
4	4 0	300823	2023	8	30	114109...	11	41	9	0
5	5 0	300823	2023	8	30	114109...	11	41	9	0
6	6 0	300823	2023	8	30	114109...	11	41	9	0
7	7 0	300823	2023	8	30	114109...	11	41	9	0
8	8 0	300823	2023	8	30	114109...	11	41	9	0
9	9 0	300823	2023	8	30	114827...	11	48	27	0
10	... 0	300823	2023	8	30	114827...	11	48	27	0
11	... 0	300823	2023	8	30	114827...	11	48	27	0
12	... 0	300823	2023	8	30	115102...	11	51	2	0

Gambar 4.11 Hasil penyimpanan data di dalam SQLITE

Database SQLITE dapat dibuka dengan aplikasi tambahan DB Browser yang perlu diinstal dulu dalam raspberrypi. Selain dapat dijalankan menggunakan aplikasi lain seperti SQLITE Studio SQLITE juga dapat dibuka pada terminal . Data yang berhasil disimpan pada Gambar 4.11 menujukan beberapa baris dan kolom data yang sebelumnya telah di *subscribe* oleh Node-RED. Data yang telah berhasil disimpan ini akan di *export* kedalam format lain CSV untuk dapat menjadi masukan pemodelan simulink MATLAB.

4.5 Pengujian Data Hasil DAQ

4.5.1 Rute Pengambilan Data DAQ



Gambar 4.12 Rute Pengambilan data DAQ

Rute pengambilan data berjarak 5,3 km dimulai dari perumahan meadow greean menuju cifest kukun cikarang. Secara umum perjalan berjalan lancar karena tidak adanya rute yang macet karena kendaraan penuh atau penyebab macet lain nya.

4.5.2 Data Hasil DAQ

Data yang diperoleh selama proses pengambilan secara langsung terdapat beberapa data yang memiliki karakteristik sesuai dengan sistem simulasi dan menunjukkan pengaruh terhadap konsumsi batrai. Data kecepatan pada Tabel 4.2 diperoleh secara langsung dari kecepatan kendaraan yang sedang dalam perjalanan. Satuan kecepatan menggunakan (m/s) karena ketika proses pengambilan data kendaraan melaju dengan kecepatan yang relatif pelan dan jarak yang kurang dari 10 km.

Table 4.2 Tabel Hasil DAQ kecepatan

No	Kecepatan (m/s)	Waktu
1	0.33	14:20:16
2	0.03	14:19:25
3	0.33	14:18:34
4	0.28	14:17:41
5	0.42	14:16:50
6	0.42	14:15:59
7	0.07	14:15:07
8	0	14:14:40
9	0.07	14:13:49
10	0.14	14:12:57
11	0.14	14:12:06
12	0.19	14:11:14
13	0.19	14:10:23
14	4.2	14:09:31
15	4.2	14:08:40
16	2.69	14:07:48

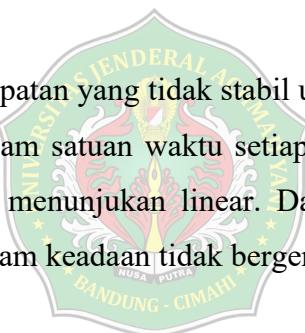
17		2.69	14:06:57
18		2.69	14:05:40
19		2.69	14:04:23
20		2.69	14:03:31
21		0.63	14:03:05
22		0.63	14:02:35
23		0.33	14:00:57
24		0.33	14:00:05
25		7.77	13:59:14
26		7.77	13:58:22
27		6.42	13:57:56
28		6.42	13:57:05
29		6.22	13:56:13
30		5	13:55:20
31		5	13:54:29
32		5	13:53:38
33		5	13:51:54
34		0.34	13:51:02
35		0.12	13:50:11
36		0.12	13:49:18
37		0.12	13:48:27
38		0.09	13:47:35
39		0.09	13:46:44

Selain data kecepatan ada beberapa data lain sebagai pendukung seperti ditunjukan pada Tabel 4.3. Tabel pengambilan data dapat divisualkan dengan bentuk grafik line seperti ditunjukan pada Gambar4.12.

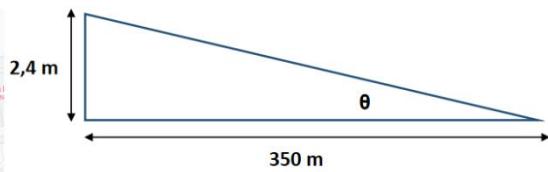
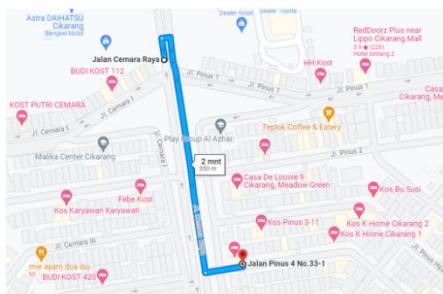


Gambar 4.13 Hasil penyimpanan data di dalam SQLITE

Terlihat pada Gambar4.12 kecepatan yang tidak stabil untuk masukan pada sistem. Data kecepatan ini diambil dalam satuan waktu setiap satu menit satu kali. Dan karakter data kecepatan tidak menunjukan linear. Data yang mendekati 0 m/s memiliki kondisi kendaraan dalam keadaan tidak bergerak.



4.5.3 Menentukan Gradien Profile Kemiringan Jalan



$$\text{arc tan}(\theta) = \frac{\text{sisi depan}}{\text{sisi samping}}$$

$$\text{arc tan}(\theta) = \frac{2,4}{350}$$

$$\text{arc tan}(\theta) = 0,00685714$$

$$\theta = 0,392879^\circ$$

Gambar 4.14 Menentukan kemiringan Gradien

Dalam proses menentukan kemiringan jalan menggunakan persamaan Pythagoras seperti ditunjukkan pada Gambar 4.12. Gradien dinyatakan dalam satuan derajat yang akan dimasukan kedalam bagian masukan sistem simulasi. Berdasarkan hasil pengukuran DAQ dan perhitungan diatas diperoleh tabel pengukuran seperti ditunjukkan pada Tabel 4.3.

Tabel 4.3 Menentukan kemiringan pada profile jalan

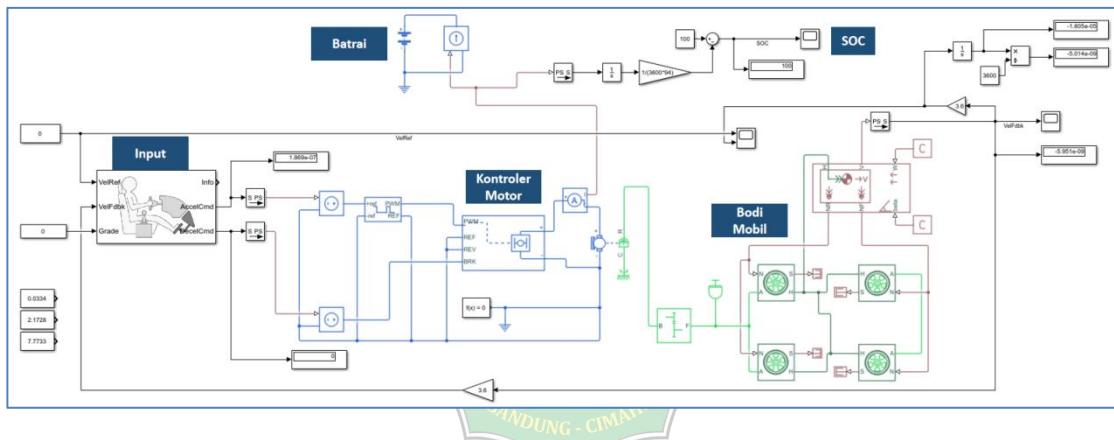
Latitude	Longitude	Altitude (m)	Gradien (%)	Average (m/s)	Waktu
-6.342.129	107.117.137	24.2	0	2,20	14:20:16
-6.342.129	107.117.137	24.2	0		14:19:25
-6.342.129	107.117.137	24.2	0		14:18:34
-6.342.184	107.117.210	24.2	0		14:17:41
-6.342.138	107.117.230	24.2	0		14:16:50
-6.342.138	107.117.230	24.2	0		14:15:59
-6.342.238	107.117.196	24.2	0		14:15:07
-6.342.238	107.117.196	24.2	0		14:14:40
-6.342.238	107.117.196	24.2	0		14:13:49
-6.342.209	107.117.255	24.2	0		14:12:57
-6.342.209	107.117.255	24.2	0		14:12:06
-6.342.219	107.117.251	24.2	0		14:11:14
-6.342.219	107.117.251	24.2	0		14:10:23
-6.342.330	107.117.305	24.2	0		14:09:31
-6.342.330	107.117.305	24.2	0		14:08:40

-6.331.839	107.128.656	24.2	0		14:07:48
-6.331.839	107.128.656	24.2	0		14:06:57
-6.331.839	107.128.656	24.2	0		14:05:40
-6.331.839	107.128.656	24.2	0		14:04:23
-6.331.839	107.128.656	24.2	0		14:03:31
-6.331.772	107.128.820	24.2	0		14:03:05
-6.331.772	107.128.820	24.2	0		14:02:13
-6.331.847	107.128.818	24.2	0		14:00:57
-6.331.847	107.128.818	24.2	0		14:00:05
-6.330.414	107.131.800	24.2	0		13:59:14
-6.330.414	107.131.800	24.2	0		13:58:22
-6.333.140	107.136.424	24.2	0		13:57:56
-6.333.140	107.136.424	24.2	0		13:57:05
-6.332.771	107.139.357	24.2	0		13:56:13
-6.332.648	107.140.516	24.2	0		13:55:20
-6.333.298	107.141.573	42.5	2.4	5	13:54:29
-6.333.298	107.141.573	42.5	2.4		13:53:38
-6.335.246	107.142.331	44.9	2.4	1.36	13:52:46
-6.335.246	107.142.331	44.9	2.4		13:51:54
-6.335.110	107.143.083	44.9	2.4		13:51:02
-6.335.345	107.142.990	44.9	2.4		13:50:11
-6.335.345	107.142.990	44.9	2.4		13:49:18
-6.335.345	107.142.990	44.9	2.4		13:48:27

-6.335.331	107.143.075	44.9	2.4		13:47:35
-6.335.331	107.143.075	44.9	2.4		13:46:44

Dari data altitude dapat diperoleh besaran dari kemiringan jalan dengan memanfaatkan perhitungan matematika. *Sample* kemiringan jalan dibuat sama pada beberapa titik karena memiliki ketinggian diatas permukaan laut yang relatif tidak terlalu tinggi.

4.5.4 Simulasi dengan Model Simulink Mobil Listrik



Gambar 4.12 Simulasi mobil listrik dengan simulink

Masukan pada bagian blok *longitudinal driver* merupakan data yang sebelumnya diperoleh ketika proses DAQ. Ada tiga data utama yang akan menjadi masukan pada sistem simulasi diantaranya kecepatan, kemiringan jalan, dan kecepatan hasil umpan balik. Dari ketiga besaran ini akan diperoleh hasil dari SOC(*State of Charge*) atau kapasitas dari batari yang digunakan telah menghabiskan berapa banyak energi dari keseluruhan kapasitas maksimalnya.

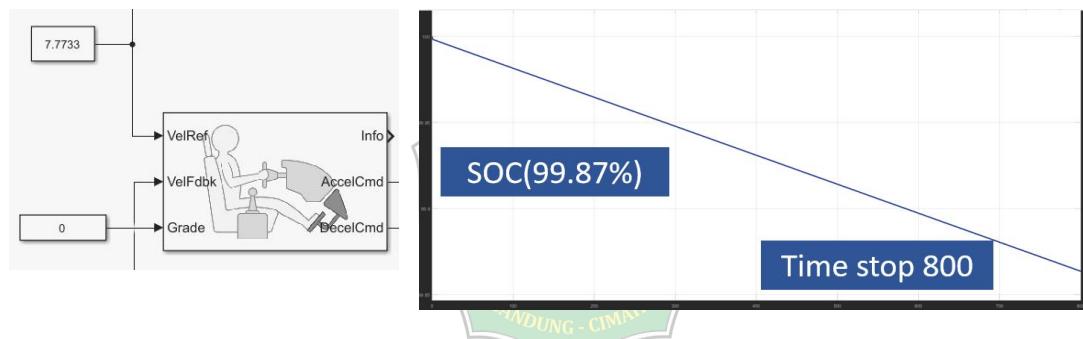
a. Untuk kecepatan 2,1728 m/s dan kemiringan 0



Gambar 4.13 Simulasi mobil listrik dengan simulink

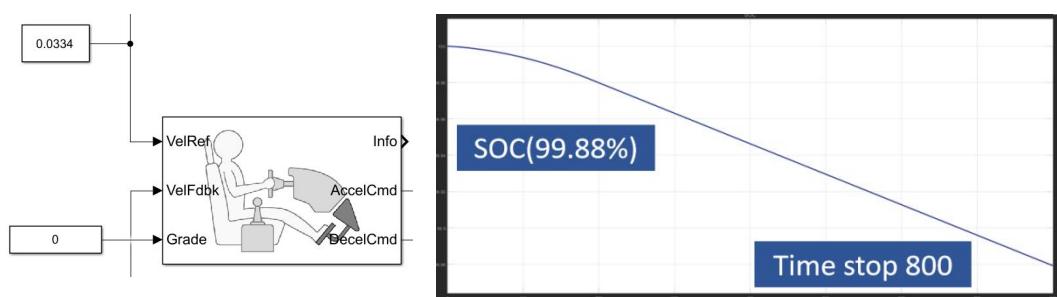
Ketika kemiringan 0 derajat dan kecepatan 2.1728 maka SOC = 99.87% .
kecepatan masukan menggunakan kecepatan rata-rata.

b. Untuk kecepatan 7,773 m/s dan kemiringan 0



Gambar 4.14 Simulasi mobil listrik dengan simulink

Ketika kemiringan 0 derajat dan kecepatan 7.773 m/s maka SOC = 99.87%.
kecepatan tertinggi ketika proses sample data diambil.



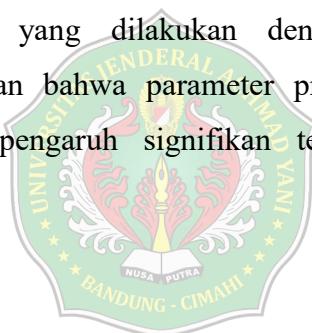
Gambar 4.15 Simulasi mobil listrik dengan simulink

Ketika kemiringan 0 derajat dan kecepatan 0.0334 m/s maka SOC = 99.88% . kecepatan masukan menggunakan kecepatan terendah. Berdasarkan hasil simulasi dengan masukan DAQ tersebut diperoleh data seperti ditunjukkan pada tabel 4.4.

Tabel 4.4 Hasil simulasi mobil listrik dengan masukan DAQ

No	Kecepatan (m/s)	SOC (%)
1	0	100
2	2.1728	99.87
3	7.7733	99.87
4	0.0334	99.88
5	30	99.85

Berdasarkan hasil simulasi yang dilakukan dengan menggunakan Data Acquisition (DAQ), ditemukan bahwa parameter profil jalan dan kecepatan kendaraan listrik memiliki pengaruh signifikan terhadap konsumsi energi kendaraan listrik.



BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penelitian didapatkan beberapa point kesimpulan yang dicantumkan dibawah ini antara lain:

1. Sistem DAQ(*Data Acquisition*) untuk masukan simulasi dapat mengambil *sample* sinyal yang dibutuhkan seperti longitude, latitude dan besaran kemiringan sesuai dengan satunya.
2. Data hasil pengambilan *sample* berhasil tersimpan didalam database SQLITE untuk memudahkan dalam analisis lanjutan dalam penelitian lebih lanjut.
3. Metode RTOS(*Real Time Oeprating System*) memudahkan dalam pengambilan data karena dapat dimonitoring secara langsung proses pengambilan data ketika kendaraan listrik digunakan.
4. Simulasi kendaraan listrik dengan menggunakan software MATLAB menggunakan metode simulink berhasil menampilkan pengaruh dari profil jalan seperti kemiringan, kecepatan jalan terhadap konsumsi batrai.
5. Pemodelan simulasi kendaraan listrik dapat digunakan unutk membandingkan antara spesifikasi batrai tipe apa saja yang digunakan.
6. Profile kemiringan jalan dan kecepatan kendaraan litrik berperngatuh terhadap konsumsi batrai.
7. Pemodelan simulasi kendaraan listrik dapat digunakan unutk membandingkan antara spesifikasi mobil listrik tipe apa saja yang digunakan.
8. Penggunaan protokol komunikasi *MQTT* menghasilkan pengiriman data yang baik untuk sistem DAQ.

5.2 Saran

Ketika melakukan proses pengambilan data pada penelitian yang melibatkan kendaraan listrik, ada kebutuhan untuk menggunakan dua koneksi internet terpisah. Koneksi pertama digunakan untuk alat yang dipasangkan pada kendaraan listrik, dan koneksi kedua digunakan untuk alat yang terhubung ke server. Kedua

koneksi ini harus bekerja secara sinkron untuk memastikan bahwa data yang diambil dari kendaraan dapat segera dikirim dan disimpan di server tanpa ada kehilangan data. Penggunaan dua koneksi ini sangat penting terutama pada penelitian atau eksperimen di mana keakuratan dan kelancaran pengambilan data merupakan faktor kunci keberhasilan.

Dalam simulasi yang menggunakan perangkat lunak seperti MATLAB, yang membutuhkan pemrosesan intensif, spesifikasi perangkat keras komputer memainkan peran yang sangat penting. Simulasi MATLAB yang melibatkan pemodelan sistem kompleks atau analisis data besar memerlukan daya komputasi yang signifikan. Jika perangkat komputer yang digunakan tidak memiliki spesifikasi yang cukup tinggi, proses simulasi akan berjalan lebih lama dan dapat mengakibatkan masalah serius, seperti overheating.



DAFTAR PUSTAKA

- [1] “Instruksi Presiden (INPRES) Nomor 7 Tahun 2022 tentang Penggunaan Kendaraan Bermotor Listrik Berbasis Baterai (Battery Electric Vehicle) Sebagai Kendaraan Dinas Operasional dan/atau Kendaraan Perorangan Dinas Instansi Pemerintah Pusat dan Pemerintahan Daerah.” 2022.
- [3] A. Mulay, Y. V. Pant, R. Mangharam, “Protodrive: Rapid Prototyping Platform for Electric Vehicle Powertrain”, 2012.
- [4] L. Mao, A. Fotouhi, N. Shateri, and N. Ewin, “A multi-mode electric vehicle range estimator based on driving pattern recognition,” *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2021, doi: 10.1177/09544062211032994
- [5] M. Uzair, G. Abbas, and S. Hosain, “Characteristics of Battery Management Systems of Electric Vehicles with Consideration of the Active and Passive Cell Balancing Process,” in *World Electric Vehicle Journal*, Aug. 2021, vol. 12, no. 3. doi: 10.3390/wevj12030120.
- [6] D. Kusanto and K. Indriawati, “Perancangan Sistem Akuisisi Data Sebagai Alternatif Modul DAQ LabVIEW Menggunakan Mikrokontroler ATMEGA8535,” 2010.
- [7] G. Yudha Saputra, A. Denhas Afrizal, F. Khusnu Reza Mahfud, F. Angga Pribadi, and F. Jati Pamungkas, “Penerapan Protokol MQTT pada Teknologi Wan (Studi Kasus Sistem Parkir Univeristas Brawijaya),” 2017.
- [8] “Raspberry Pi 4.” <https://www.raspberrypi.org> (accessed Feb. 02, 2022).
- [9] “Node-RED.” <https://nodered.org> (accessed Aug. 30, 2022).
- [10] “SQLITE.” <https://www.sqlite.org> (accessed Aug. 30, 2022)

LAMPIRAN A

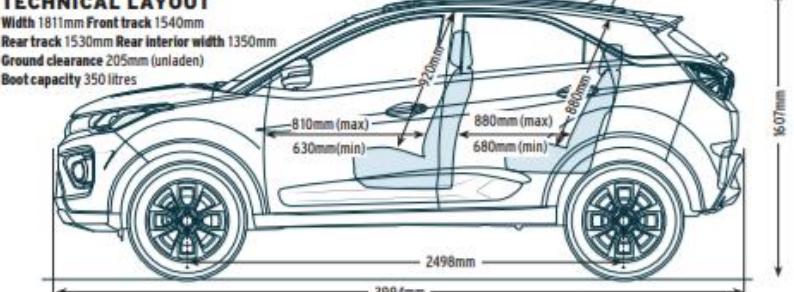
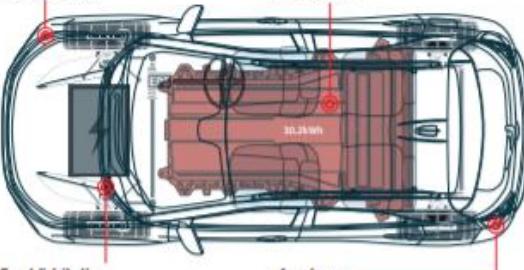
SPESIFIKASI SENSOR GPS Neo-Ublox-M8

Parameter	Symbol	Module	Min	Typ	Max	Units	Condition
Power supply voltage	VCC	NEO-6G	1.75	1.8	1.95	V	
		NEO-6Q/M NEO-6P/V/T	2.7	3.0	3.6	V	
Supply voltage USB	VDDUSB	All	3.0	3.3	3.6	V	
Backup battery voltage	V_BCKP	All	1.4		3.6	V	
Backup battery current	I_BCKP	All		22		µA	V_BCKP = 1.8 V, VCC = 0V
Input pin voltage range	Vin	All	0		VCC	V	
Digital IO Pin Low level input voltage	Vil	All	0		0.2*VCC	V	
Digital IO Pin High level input voltage	Vih	All	0.7*VCC		VCC	V	
Digital IO Pin Low level output voltage	Vol	All			0.4	V	IoI=4mA
Digital IO Pin High level output voltage	Voh	All	VCC -0.4		V		IoH=4mA
USB_DM, USB_DP	VinU	All					Compatible with USB with 22 Ohms series resistance
VCC_RF voltage	VCC_RF	All			VCC-0.1	V	
VCC_RF output current	ICC_RF	All			50	mA	
Antenna gain	Gant	All			50	dB	
Receiver Chain Noise Figure	NFtot	All		3.0		dB	
Operating temperature	Topr	All	-40		85	°C	



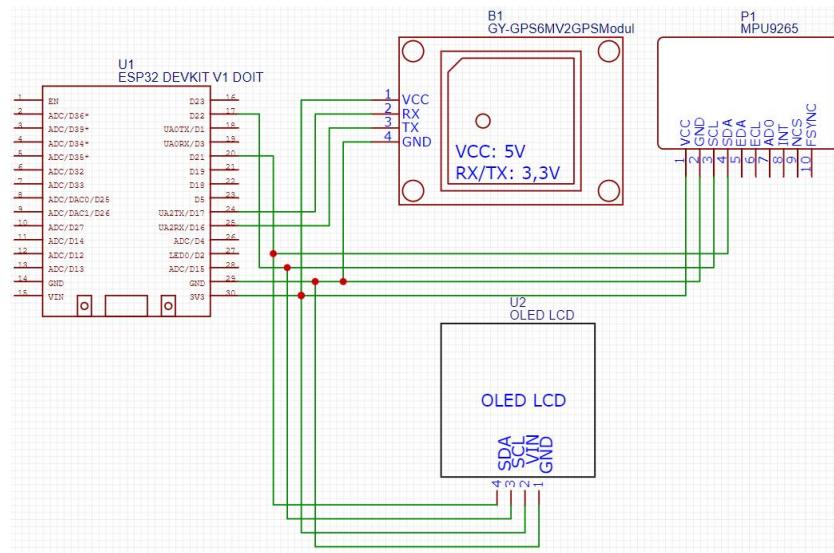
LAMPIRAN B

SPESIFIKASI KENDARAAN MOBIL LISTRIK

DATA LOG			
TATA NEXON EV			
Price Rs 16.25 lakh Warranty 3 years or 1,25,000km* 8 years or 1,60,000km on battery and motor			
BATTERY			
Rating 30.2kWh Type Lithium-ion Voltage 320V Motor type Permanent magnet synchronous motor Power 129hp Torque 245Nm Power to weight 92hp per tonne Torque to weight 175Nm per tonne			
TRANSMISSION			
Type Single-speed automatic Final drive ratio 9.1:1			
CHASSIS & BODY			
Construction Five-door, monocoque SUV Weight 1400kg Tires 215/60 R16 Spare 215/60 R16			
SUSPENSION			
Front Independent, MacPherson strut with coil springs Rear Torsion beam with hydraulic shock absorbers, coil springs			
STEERING			
Type Rack and pinion Type of power assist Electric Turning circle 10.2m			
BRAKES			
Front Disc Rear Drum Anti-lock Yes			
RANGE			
TEST City 216km per charge Highway 201km per charge			
EQUIPMENT CHECK LIST			
ESP NA Airbags 2 Sunroof ■ Cruise control NA Touchscreen ■ Android Auto ■ Apple CarPlay ■ LED headlamps NA Climate control ■ Rear AC vents ■ Wireless phone charging NA			
ANNUAL SERVICE COSTS			
Year 1 3279 Year 2 6046 Year 3 4250			
EV vs DIESEL AUTOMATIC, OWNERSHIP COST FOR 3 YEARS			
NEXON EV NEXON D AMT COMMENTS			
VARIANT	X2+ Lux	XZA+ (0)	
Ex-showroom Delhi	1625000	1250000	
0.75% Tax Collected at Source	12188	9375	
Registration + road tax	5000	163100	Full waiver of road tax for EV
Insurance	50863	39125	
On-road price	1693050	1461600	
Delhi state govt. subsidy	-150000	-	To be availed from the state govt. directly, not from the dealer.
Final buy price (A)	1543050	1461600	
Electricity / fuel cost per unit	8.00	73.87	Rs. 8 per unit considered as average cost of electricity Pan India.
Running cost per km	1.16	4.62	For EV - 240 (i.e. 8*30kWh) divide by 208 (range); For diesel - 73.87/16(kg)
Running cost for 10000km	11615	46200	Cost per km* 10000
Running cost after 3 years (B)	34846	138600	
TECHNICAL LAYOUT			
			
Five starer It's a modified version of a platform that has secured a solid five stars in Global NCAP crash tests.			
IP67 rated Its battery pack has been submerged one metre underwater for half an hour without any ill effects.			
			
Equal distribution With a 50:50 weight distribution across its axles, this one feels very sporty from behind the wheel.			
4mm lower The battery is tucked beneath the cabin, yet the ground clearance is only 4mm lower than the ICE version.			

LAMPIRAN C

DIAGRAM SKEMATIK DAN PROGRAM



Gambar B.1 Diagram skematik rangkaian



Program ESP32

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BusIO_Register.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include <WiFi.h>

const char* ssid = "Pena";
const char* password = "wandaadib";
const char* pub_topic_gps_enc = "publish_e_v_ta_gps_enc";
const char* pub_topic_gps_date = "publish_e_v_ta_gps_date";
const char* pub_topic_gps_year = "publish_e_v_ta_gps_year";
const char* pub_topic_gps_month = "publish_e_v_ta_gps_month";
const char* pub_topic_gps_day = "publish_e_v_ta_gps_day";
const char* pub_topic_gps_time = "publish_e_v_ta_gps_time";
const char* pub_topic_gps_time_hour = "publish_e_v_ta_gps_time_hour";
const char* pub_topic_gps_time_min = "publish_e_v_ta_gps_time_min";
const char* pub_topic_gps_time_sec = "publish_e_v_ta_gps_time_sec";
const char* pub_topic_gps_time_cen_sec = "publish_e_v_ta_gps_time_cen_sec";
const char* pub_topic_gps_speed = "publish_e_v_ta_gps_speed";
const char* pub_topic_gps_speed_knots = "publish_e_v_ta_gps_speed_knots";
const char* pub_topic_gps_speed_mph = "publish_e_v_ta_gps_speed_mph";
const char* pub_topic_gps_speed_mps = "publish_e_v_ta_gps_speed_mps";
const char* pub_topic_gps_speed_kmph = "publish_e_v_ta_gps_speed_kmph";
```

```

const char* pub_topic_gps_course_raw = "publish_e_v_ta_gps_cour_raw";
const char* pub_topic_gps_course_raw_deg = "publish_e_v_ta_gps_cour_raw_deg";
const char* pub_topic_gps_satelite = "publish_e_v_ta_gps_satelite";
const char* pub_topic_gps_hdop = "publish_e_v_ta_gps_hdop";
const char* pub_topic_gps_latitude = "publish_e_v_ta_latitude"; // publish/username/apiKeyIn
const char* pub_topic_gps_latitude_raw = "publish_e_v_ta_gps_lat_raw";
const char* pub_topic_gps_latitude_raw_deg = "publish_e_v_ta_gps_lat_raw_deg";
const char* pub_topic_gps_latitude_raw_bill = "publish_e_v_ta_gps_lat_raw_bill";
const char* pub_topic_gps_longitude = "publish_e_v_ta_longitude";
const char* pub_topic_gps_longitude_raw = "publish_e_v_ta_gps_long_raw";
const char* pub_topic_gps_longitude_raw_deg = "publish_e_v_ta_gps_long_deg";
const char* pub_topic_gps_longitude_raw_bill = "publish_e_v_ta_gps_long_bill";
const char* pub_topic_gps_altitude = "publish_e_v_ta_altitude";
const char* pub_topic_gps_altitude_meter = "publish_e_v_ta_gps_alt_meter";
const char* pub_topic_gps_altitude_mil = "publish_e_v_ta_gps_alt_mil";
const char* pub_topic_gps_altitude_km = "publish_e_v_ta_gps_alt_km";
const char* pub_topic_gps_altitude_feet = "publish_e_v_ta_gps_alt_feet";
const char* pub_topic_aks_x = "publish_e_v_ta_gyro_akselerasi_x";
const char* pub_topic_aks_y = "publish_e_v_ta_gyro_akselerasi_y";
const char* pub_topic_aks_z = "publish_e_v_ta_gyro_akselerasi_z";
const char* pub_topic_rts_x = "publish_e_v_ta_gyro_rotasi_x";
const char* pub_topic_rts_y = "publish_e_v_ta_gyro_rotasi_y";
const char* pub_topic_rts_z = "publish_e_v_ta_gyro_rotasi_z";
const char* pub_topic_temp = "publish_e_v_ta_gyro_temperature";

```

```

const unsigned int writeInterval = 25000;
static const int RXPin = 16, TXPin = 17;
static const uint32_t GPSBaud = 9600;
const char* mqtt_server = "broker.mqtt-dashboard.com";
unsigned int mqtt_port = 1883;
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE      (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;

```

```

Adafruit_MPU6050 mpu;
WiFiClient espClient;
PubSubClient client(espClient);
TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire);

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    randomSeed(micros());
    Serial.println("");
}

```



```

Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i = 0; i < length; i++) {
Serial.print((char)payload[i]);
}
Serial.println();
}

```

```

void main_gyro(){
while (!Serial)
delay(10);
if (!mpu.begin()) {
Serial.println("Failed to find MPU6050 chip");
while (1) {
delay(10);
}
Serial.println("MPU6050 Found!");
mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
Serial.print("Accelerometer range set to: ");
switch (mpu.getAccelerometerRange()) {
case MPU6050_RANGE_2_G:
Serial.println("+-2G");
break;
case MPU6050_RANGE_4_G:

```



```

Serial.println("+-4G");
break;

case MPU6050_RANGE_8_G:
Serial.println("+-8G");
break;

case MPU6050_RANGE_16_G:
Serial.println("+-16G");
break;

}

mpu.setGyroRange(MPU6050_RANGE_500_DEG);
Serial.print("Gyro range set to: ");
switch (mpu.getGyroRange()) {
case MPU6050_RANGE_250_DEG:
Serial.println("+- 250 deg/s");
break;

case MPU6050_RANGE_500_DEG:
Serial.println("+- 500 deg/s");
break;

case MPU6050_RANGE_1000_DEG:
Serial.println("+- 1000 deg/s");
break;

case MPU6050_RANGE_2000_DEG:
Serial.println("+- 2000 deg/s");
break;

}

mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
Serial.print("Filter bandwidth set to: ");
switch (mpu.getFilterBandwidth()) {
case MPU6050_BAND_260_HZ:
Serial.println("260 Hz");
break;
}

```



```

case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
}
Serial.println("");
delay(100);
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        String clientId = "ESP8266Client-";
        clientId += String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println("connected");
            client.publish("outTopic", "hello world");
            client.subscribe("inTopic");

```

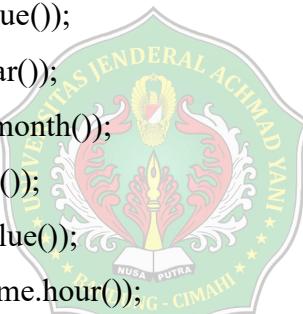


```

} else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    delay(5000);
}
}

void control_program() {
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);
    if (gps.location.isValid()){
        double gps_enc = (gps.encode(ss.read()));
        double gps_date = (gps.date.value());
        double gps_year = (gps.date.year());
        double gps_month = (gps.date.month());
        double gps_day = (gps.date.day());
        double gps_time = (gps.time.value());
        double gps_time_hour = (gps.time.hour());
        double gps_time_min = (gps.time.minute());
        double gps_time_sec = (gps.time.second());
        double gps_time_cen_sec = (gps.time.centisecond());
        double gps_speed = (gps.speed.value());
        double gps_speed_knots = (gps.speed.knots());
        double gps_speed_mph = (gps.speed.mph());
        double gps_speed_mps = (gps.speed.mps());
        double gps_speed_kmph = (gps.speed.kmph());
        double gps_course_raw = (gps.course.value());
        double gps_course_raw_deg = (gps.course.deg());
        double gps_satelite = (gps.satellites.value());
    }
}

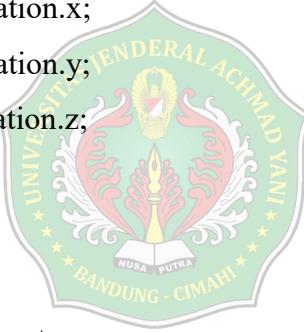
```



```

double gps_hdop = (gps.hdop.value());
double latitude = (gps.location.lat());
double latitude_raw = (gps.location.rawLat().negative());
double latitude_raw_deg = (gps.location.rawLat().deg);
double latitude_raw_bill = (gps.location.rawLat().billions());
double longitude = (gps.location.lng());
double longitude_raw = (gps.location.rawLng().negative());
double longitude_raw_deg = (gps.location.rawLng().deg);
double longitude_raw_bill = (gps.location.rawLng().billions());
double altitude = (gps.altitude.value());
double altitude_meter = (gps.altitude.meters());
double altitude_mil = (gps.altitude.miles());
double altitude_km = (gps.altitude.kilometers());
double altitude_feet = (gps.altitude.feet());
double akselerasi_x = a.acceleration.x;
double akselerasi_y = a.acceleration.y;
double akselerasi_z = a.acceleration.z;
double rotasi_x = g.gyro.x;
double rotasi_y = g.gyro.y;
double rotasi_z = g.gyro.z;
double temp_gyro = temp.temperature;
// _____

```



```

char mqtt_payload_gps_enc[50] = "";
char mqtt_payload_gps_date[50] = "";
char mqtt_payload_gps_year[50] = "";
char mqtt_payload_gps_month[50] = "";
char mqtt_payload_gps_day[50] = "";
char mqtt_payload_gps_time[50] = "";
char mqtt_payload_gps_time_hour[50] = "";
char mqtt_payload_gps_time_min[50] = "";

```

```
char mqtt_payload_gps_time_sec[50] = "";
char mqtt_payload_gps_time_cen_sec[50] = "";
char mqtt_payload_gps_speed[50] = "";
char mqtt_payload_gps_speed_knots[50] = "";
char mqtt_payload_gps_speed_mph[50] = "";
char mqtt_payload_gps_speed_mps[50] = "";
char mqtt_payload_gps_speed_kmph[50] = "";
char mqtt_payload_gps_course_raw[50] = "";
char mqtt_payload_gps_course_raw_deg[50] = "";
char mqtt_payload_gps_satelite[50] = "";
char mqtt_payload_gps_hdop[50] = "";
char mqtt_payload_gps_latitude[50] = "";
char mqtt_payload_gps_latitude_raw[50] = "";
char mqtt_payload_gps_latitude_raw_deg[50] = "";
char mqtt_payload_gps_latitude_raw_bill[50] = "";
char mqtt_payload_gps_longitude[50] = "";
char mqtt_payload_gps_longitude_raw[50] = "";
char mqtt_payload_gps_longitude_raw_deg[50] = "";
char mqtt_payload_gps_longitude_raw_bill[50] = "";
char mqtt_payload_gps_altitude[50] = "";
char mqtt_payload_gps_altitude_meter[50] = "";
char mqtt_payload_gps_altitude_mil[50] = "";
char mqtt_payload_gps_altitude_km[50] = "";
char mqtt_payload_gps_altitude_feet[50] = "";
char mqtt_payload_akselerasi_x[50] = "";
char mqtt_payload_akselerasi_y[50] = "";
char mqtt_payload_akselerasi_z[50] = "";
char mqtt_payload_rotasi_x[50] = "";
char mqtt_payload_rotasi_y[50] = "";
char mqtt_payload_rotasi_z[50] = "";
char mqtt_payload_temp[50] = "";
```

```
//

---



---

  
snprintf (mqtt_payload_gps_enc, 50, "%lf", gps_enc);  
Serial.print("Publish message gps-enc: ");  
Serial.println(mqtt_payload_gps_enc);  
client.publish(pub_topic_gps_enc, mqtt_payload_gps_enc);  
  
snprintf (mqtt_payload_gps_date, 50, "%lf", gps_date);  
Serial.print("Publish message gps-date: ");  
Serial.println(mqtt_payload_gps_date);  
client.publish(pub_topic_gps_date, mqtt_payload_gps_date);  
  
snprintf (mqtt_payload_gps_year, 50, "%lf", gps_year);  
Serial.print("Publish message gps-year: ");  
Serial.println(mqtt_payload_gps_year);  
client.publish(pub_topic_gps_year, mqtt_payload_gps_year);  
  
snprintf (mqtt_payload_gps_month, 50, "%lf", gps_month);  
Serial.print("Publish message gps-month: ");  
Serial.println(mqtt_payload_gps_month);  
client.publish(pub_topic_gps_month, mqtt_payload_gps_month);  
  
snprintf (mqtt_payload_gps_day, 50, "%lf", gps_day);  
Serial.print("Publish message gps-day: ");  
Serial.println(mqtt_payload_gps_day);  
client.publish(pub_topic_gps_day, mqtt_payload_gps_day);  
  
snprintf (mqtt_payload_gps_time, 50, "%lf", gps_time);  
Serial.print("Publish message gps-time: ");  
Serial.println(mqtt_payload_gps_time);
```

```

client.publish(pub_topic_gps_time, mqtt_payload_gps_time);

snprintf(mqtt_payload_gps_time_hour, 50, "%lf", gps_time_hour);
Serial.print("Publish message gps-time-hour: ");
Serial.println(mqtt_payload_gps_time_hour);
client.publish(pub_topic_gps_time_hour, mqtt_payload_gps_time_hour);

snprintf(mqtt_payload_gps_time_min, 50, "%lf", gps_time_min);
Serial.print("Publish message gps-time-min: ");
Serial.println(mqtt_payload_gps_time_min);
client.publish(pub_topic_gps_time_min, mqtt_payload_gps_time_min);

snprintf(mqtt_payload_gps_time_sec, 50, "%lf", gps_time_sec);
Serial.print("Publish message gps-time-sec: ");
Serial.println(mqtt_payload_gps_time_sec);
client.publish(pub_topic_gps_time_sec, mqtt_payload_gps_time_sec);

snprintf(mqtt_payload_gps_time_cen_sec, 50, "%lf", gps_time_cen_sec);
Serial.print("Publish message gps-time-cen-min: ");
Serial.println(mqtt_payload_gps_time_cen_sec);
client.publish(pub_topic_gps_time_cen_sec, mqtt_payload_gps_time_cen_sec);

snprintf(mqtt_payload_gps_speed, 50, "%lf", gps_speed);
Serial.print("Publish message gps-speed: ");
Serial.println(mqtt_payload_gps_speed);
client.publish(pub_topic_gps_speed, mqtt_payload_gps_speed);

snprintf(mqtt_payload_gps_speed_knots, 50, "%lf", gps_speed_knots);
Serial.print("Publish message gps-speed-knots: ");
Serial.println(mqtt_payload_gps_speed_knots);
client.publish(pub_topic_gps_speed_knots, mqtt_payload_gps_speed_knots);

```

```

snprintf (mqtt_payload_gps_speed_mph, 50, "%lf", gps_speed_mph);
Serial.print("Publish message gps-speed-mph: ");
Serial.println(mqtt_payload_gps_speed_mph);
client.publish(pub_topic_gps_speed_mph, mqtt_payload_gps_speed_mph);

snprintf (mqtt_payload_gps_speed_mps, 50, "%lf", gps_speed_mps);
Serial.print("Publish message gps-speed-mps: ");
Serial.println(mqtt_payload_gps_speed_mps);
client.publish(pub_topic_gps_speed_mps, mqtt_payload_gps_speed_mps);

snprintf (mqtt_payload_gps_speed_kmph, 50, "%lf", gps_speed_kmph);
Serial.print("Publish message gps-speed-kmph: ");
Serial.println(mqtt_payload_gps_speed_kmph);
client.publish(pub_topic_gps_speed_kmph, mqtt_payload_gps_speed_kmph);

snprintf (mqtt_payload_gps_course_raw, 50, "%lf", gps_course_raw);
Serial.print("Publish message gps-course-raw: ");
Serial.println(mqtt_payload_gps_course_raw);
client.publish(pub_topic_gps_course_raw, mqtt_payload_gps_course_raw);

snprintf (mqtt_payload_gps_course_raw_deg, 50, "%lf", gps_course_raw_deg);
Serial.print("Publish message gps-course-raw-deg: ");
Serial.println(mqtt_payload_gps_course_raw_deg);
client.publish(pub_topic_gps_course_raw_deg,
mqtt_payload_gps_course_raw_deg);

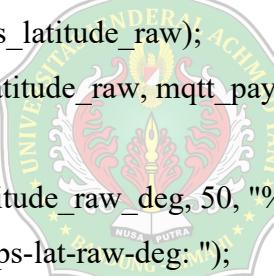
snprintf (mqtt_payload_gps_satelite, 50, "%lf", gps_satelite);
Serial.print("Publish message gps-satelite: ");
Serial.println(mqtt_payload_gps_satelite);
client.publish(pub_topic_gps_satelite, mqtt_payload_gps_satelite);

```

```
snprintf (mqtt_payload_gps_hdop, 50, "%lf", gps_hdop);
Serial.print("Publish message gps-hdop: ");
Serial.println(mqtt_payload_gps_hdop);
client.publish(pub_topic_gps_hdop, mqtt_payload_gps_hdop);
//_____
```

```
snprintf (mqtt_payload_gps_latitude, 50, "%lf", latitude);
Serial.print("Publish message gps-lat: ");
Serial.println(mqtt_payload_gps_latitude);
client.publish(pub_topic_gps_latitude, mqtt_payload_gps_latitude);
```

```
snprintf (mqtt_payload_gps_latitude_raw, 50, "%lf", latitude_raw);
Serial.print("Publish message gps-lat-raw: ");
Serial.println(mqtt_payload_gps_latitude_raw);
client.publish(pub_topic_gps_latitude_raw, mqtt_payload_gps_latitude_raw);
```



```
snprintf (mqtt_payload_gps_latitude_raw_deg, 50, "%lf", latitude_raw_deg);
Serial.print("Publish message gps-lat-raw-deg: ");
Serial.println(mqtt_payload_gps_latitude_raw_deg);
client.publish(pub_topic_gps_latitude_raw_deg,
mqtt_payload_gps_latitude_raw_deg);
```

```
snprintf (mqtt_payload_gps_latitude_raw_bill, 50, "%lf", latitude_raw_bill);
Serial.print("Publish message gps-lat-raw-bill: ");
Serial.println(mqtt_payload_gps_latitude_raw_bill);
client.publish(pub_topic_gps_latitude_raw_bill,
mqtt_payload_gps_latitude_raw_bill);
```

```
//_____
```

```
snprintf (mqtt_payload_gps_longitude, 50, "%lf", longitude);
Serial.print("Publish message gps-long: ");
Serial.println(mqtt_payload_gps_longitude);
client.publish(pub_topic_gps_longitude, mqtt_payload_gps_longitude);

snprintf (mqtt_payload_gps_longitude_raw, 50, "%lf", longitude_raw);
Serial.print("Publish message gps-long-raw: ");
Serial.println(mqtt_payload_gps_longitude_raw);
client.publish(pub_topic_gps_longitude_raw,
mqtt_payload_gps_longitude_raw);

snprintf      (mqtt_payload_gps_longitude_raw_deg,      50,      "%lf",
longitude_raw_deg);
Serial.print("Publish message gps-lat-raw-deg: ");
Serial.println(mqtt_payload_gps_longitude_raw_deg);
client.publish(pub_topic_gps_longitude_raw_deg,
mqtt_payload_gps_longitude_raw_deg);

snprintf      (mqtt_payload_gps_longitude_raw_bill,      50,      "%lf",
longitude_raw_bill);
Serial.print("Publish message gps-lat-raw-bill: ");
Serial.println(mqtt_payload_gps_longitude_raw_bill);
client.publish(pub_topic_gps_longitude_raw_bill,
mqtt_payload_gps_longitude_raw_bill);

//
```

```
snprintf (mqtt_payload_gps_altitude, 50, "%lf", altitude);
Serial.print("Publish message gps-alt: ");
```

```

Serial.println(mqtt_payload_gps_altitude);
client.publish(pub_topic_gps_altitude, mqtt_payload_gps_altitude);

snprintf (mqtt_payload_gps_altitude_meter, 50, "%lf", altitude_meter);
Serial.print("Publish message gps-alt-meter: ");
Serial.println(mqtt_payload_gps_altitude_meter);
client.publish(pub_topic_gps_altitude_meter,
mqtt_payload_gps_altitude_meter);

snprintf (mqtt_payload_gps_altitude_mil, 50, "%lf", altitude_mil);
Serial.print("Publish message gps-alt-mil: ");
Serial.println(mqtt_payload_gps_altitude_mil);
client.publish(pub_topic_gps_altitude_mil, mqtt_payload_gps_altitude_mil);

snprintf (mqtt_payload_gps_altitude_km, 50, "%lf", altitude_km);
Serial.print("Publish message gps-alt-km: ");
Serial.println(mqtt_payload_gps_altitude_km);
client.publish(pub_topic_gps_altitude_km, mqtt_payload_gps_altitude_km);

snprintf (mqtt_payload_gps_altitude_feet, 50, "%lf", altitude_feet);
Serial.print("Publish message gps-alt-feet: ");
Serial.println(mqtt_payload_gps_altitude_feet);
client.publish(pub_topic_gps_altitude_feet, mqtt_payload_gps_altitude_feet);

// _____

```

```

snprintf (mqtt_payload_akselerasi_x, 50, "%lf", akselerasi_x);
Serial.print("Publish message akselerasi-x: ");
Serial.println(mqtt_payload_akselerasi_x);
client.publish(pub_topic_aks_x, mqtt_payload_akselerasi_x);
snprintf (mqtt_payload_akselerasi_y, 50, "%lf", akselerasi_y);

```

```

Serial.print("Publish message akselerasi-y: ");
Serial.println(mqtt_payload_akselerasi_y);
client.publish(pub_topic_aks_y, mqtt_payload_akselerasi_y);
snprintf (mqtt_payload_akselerasi_z, 50, "%lf", akselerasi_z);
Serial.print("Publish message akselerasi-z: ");
Serial.println(mqtt_payload_akselerasi_z);
client.publish(pub_topic_aks_z, mqtt_payload_akselerasi_z);
snprintf (mqtt_payload_rotasi_x, 50, "%lf", rotasi_x);
Serial.print("Publish message rotasi-x: ");
Serial.println(mqtt_payload_rotasi_x);
client.publish(pub_topic_rts_x, mqtt_payload_rotasi_x);
snprintf (mqtt_payload_rotasi_y, 50, "%lf", rotasi_y);
Serial.print("Publish message rotasi-y: ");
Serial.println(mqtt_payload_rotasi_y);
client.publish(pub_topic_rts_y, mqtt_payload_rotasi_y);
snprintf (mqtt_payload_rotasi_z, 50, "%lf", rotasi_z);
Serial.print("Publish message rotasi-z: ");
Serial.println(mqtt_payload_rotasi_z);
client.publish(pub_topic_rts_z, mqtt_payload_rotasi_z);
snprintf (mqtt_payload_temp, 50, "%lf", temp_gyro);
Serial.print("Publish message temp-gyro: ");
Serial.println(mqtt_payload_temp);
client.publish(pub_topic_temp, mqtt_payload_temp);

Serial.println("> MQTT data published");

// _____
display.clearDisplay();
display.setCursor(0, 0);
display.println("Monitoring DAQ-Wnd");
display.println("=====");

```

```

display.println("Lat,long,alt");
display.print(latitude, 1);
display.print(", ");
display.print(longitude, 1);
display.print(", ");
display.print(altitude, 1);
display.println("");
display.display();
display.println("Akselerasi:x,y,z");
display.print(akselerasi_x, 1);
display.print(", ");
display.print(akselerasi_y, 1);
display.print(", ");
display.print(akselerasi_z, 1);
display.println("");
display.display();
display.println("Rotasi:x,y,z");
display.print(rotasi_x, 1);
display.print(", ");
display.print(rotasi_y, 1);
display.print(", ");
display.print(rotasi_z, 1);
display.println("");
display.display();
delay(writeInterval);// delay
}
else{
Serial.println(F("INVALID"));
}
}

```



```

// setup
void setup() {
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
    ss.begin(GPSBaud);
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)){
        Serial.println(F("SSD1306 allocation failed"));
        for (;;)
            ;
    }
    delay(500);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setRotation(0);
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
    if (!mpu.begin()) {
        Serial.println("Sensor init failed");
        while (1)
            yield();
    }
    main_gyro();
}

void loop() {
    if (!client.connected())
        reconnect();
    client.loop();
    while (ss.available() > 0)
        if(gps.encode(ss.read()))

```

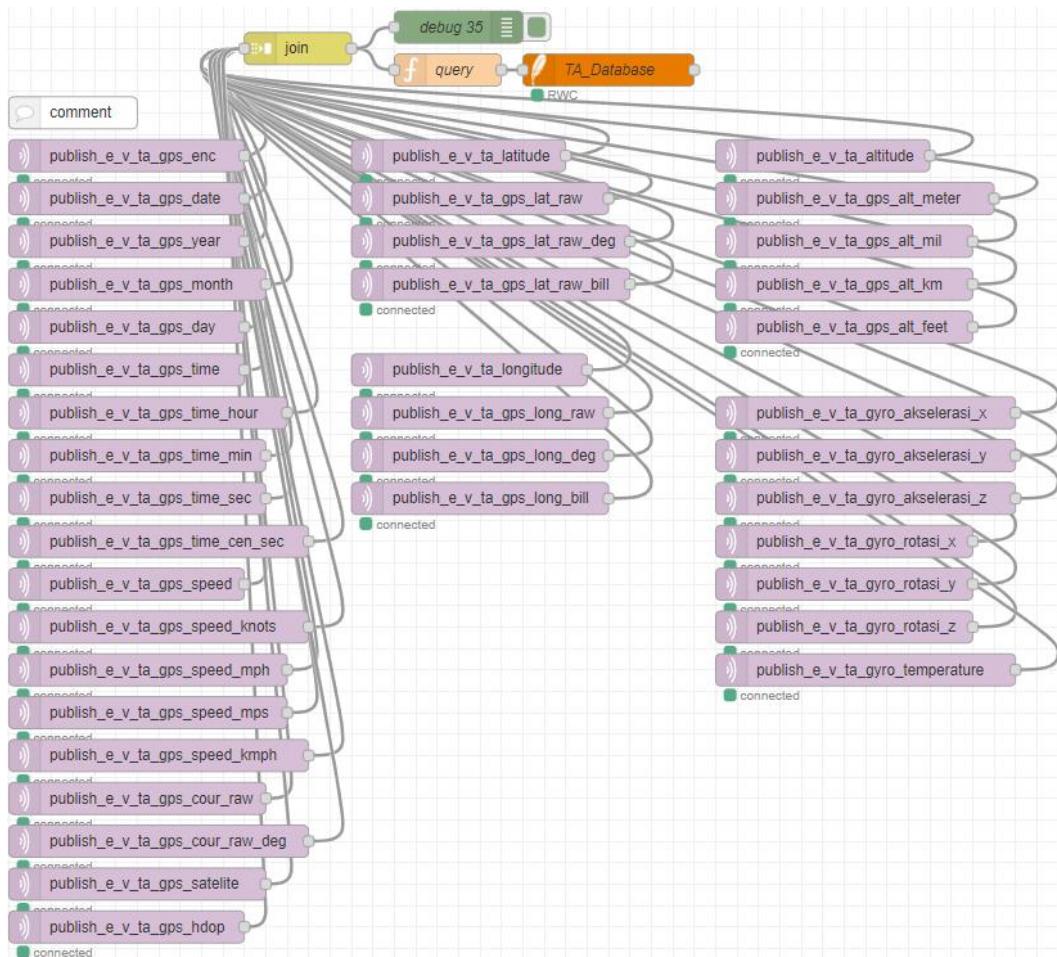


```

control_program();
if (millis() > 5000 && gps.charsProcessed() < 10){
    Serial.println(F("No GPS detected: check wiring."));
    while(true);
}
}

```

Node pada Node-RED



Program Node Query

```

let gps_enc      = msg.payload.publish_e_v_ta_gps_enc;
let gps_date     = msg.payload.publish_e_v_ta_gps_date;

```

```

let gps_year      = msg.payload.publish_e_v_ta_gps_year;
let gps_month     = msg.payload.publish_e_v_ta_gps_month;
let gps_day       = msg.payload.publish_e_v_ta_gps_day;
let gps_time      = msg.payload.publish_e_v_ta_gps_time;
let gps_time_hour = msg.payload.publish_e_v_ta_gps_time_hour;
let gps_time_min  = msg.payload.publish_e_v_ta_gps_time_min;
let gps_time_sec  = msg.payload.publish_e_v_ta_gps_time_sec;
let gps_time_cen_sec = msg.payload.publish_e_v_ta_gps_time_cen_sec;
let gps_speed     = msg.payload.publish_e_v_ta_gps_speed;
let gps_speed_knots = msg.payload.publish_e_v_ta_gps_speed_knots;
let gps_speed_mph  = msg.payload.publish_e_v_ta_gps_speed_mph;
let gps_speed_mps  = msg.payload.publish_e_v_ta_gps_speed_mps;
let gps_speed_kmph = msg.payload.publish_e_v_ta_gps_speed_kmph;
let gps_course_raw = msg.payload.publish_e_v_ta_gps_cour_raw;
let gps_course_raw_deg = msg.payload.publish_e_v_ta_gps_cour_raw_deg;
let gps_satelite   = msg.payload.publish_e_v_ta_gps_satelite;
let gps_hdop       = msg.payload.publish_e_v_ta_gps_hdop;
let latitude       = msg.payload.publish_e_v_ta_latitude;
let latitude_raw    = msg.payload.publish_e_v_ta_gps_lat_raw;
let latitude_raw_deg = msg.payload.publish_e_v_ta_gps_lat_raw_deg;
let latitude_raw_bill = msg.payload.publish_e_v_ta_gps_lat_raw_bill;
let longitude      = msg.payload.publish_e_v_ta_longitude;
let longitude_raw   = msg.payload.publish_e_v_ta_gps_long_raw;
let longitude_raw_deg = msg.payload.publish_e_v_ta_gps_long_deg;
let longitude_raw_bill = msg.payload.publish_e_v_ta_gps_long_bill;
let altitude        = msg.payload.publish_e_v_ta_altitude;
let altitude_meter   = msg.payload.publish_e_v_ta_gps_alt_meter;
let altitude_mil     = msg.payload.publish_e_v_ta_gps_alt_mil;
let altitude_km      = msg.payload.publish_e_v_ta_gps_alt_km;
let altitude_feet    = msg.payload.publish_e_v_ta_gps_alt_feet;
let gyro_aks_x      = msg.payload.publish_e_v_ta_gyro_akselerasi_x;

```

```

let gyro_aks_y      = msg.payload.publish_e_v_ta_gyro_akselerasi_y;
let gyro_aks_z      = msg.payload.publish_e_v_ta_gyro_akselerasi_z;
let gyro_rts_x      = msg.payload.publish_e_v_ta_gyro_rotasi_x;
let gyro_rts_y      = msg.payload.publish_e_v_ta_gyro_rotasi_y;
let gyro_rts_z      = msg.payload.publish_e_v_ta_gyro_rotasi_z;
let gyro_temperature = msg.payload.publish_e_v_ta_gyro_temperature;

msg.topic = 'INSERT INTO table_data(id, gps_encode, gps_date, gps_year,
gps_month,   gps_day,   gps_time,   gps_time_hour,   gps_time_minute,
gps_time_second,   gps_time_centi_second,   gps_speed,   gps_speed_knots,
speed_mph,   speed_mps,   speed_kmph,   gps_course_raw,   gps_course_raw_deg,
gps_satelite,   gps_hdop,   latitude,   latitude_raw,   latitude_raw_deg,
latitude_raw_bill,   longitude,   longitude_raw,   longitude_raw_deg,
longitude_raw_bill,   altitude,   altitude_meter,   altitude_mil,   altitude_km,
altitude_feet,   akseleration_x,   akseleration_y,   akseleration_z,   rotation_x,
rotation_y,   rotation_z,   gyro_temp,   date,   time,   device) values(null, $val1, $val2,
$val3, $val4, $val5, $val6, $val7, $val8, $val9, $val10, $val11, $val12, $val13,
$val14, $val15, $val16, $val17, $val18, $val19, $val20, $val21, $val22, $val23,
$val24, $val25, $val26, $val27, $val28, $val29, $val30, $val31, $val32, $val33,
$val34, $val35, $val36, $val37, $val38, $val39, date("now"),time("now"), "Alat-
Params");'

msg.payload = [gps_enc, gps_date, gps_year, gps_month, gps_day, gps_time,
gps_time_hour, gps_time_min, gps_time_sec, gps_time_cen_sec, gps_speed,
gps_speed_knots,   gps_speed_mph,   gps_speed_mps,   gps_speed_kmph,
gps_course_raw,   gps_course_raw_deg,   gps_satelite,   gps_hdop,   latitude,
latitude_raw,   latitude_raw_deg,   latitude_raw_bill,   longitude,   longitude_raw,
longitude_raw_deg,   longitude_raw_bill,   altitude,   altitude_meter,   altitude_mil,
altitude_km,   altitude_feet,   gyro_aks_x,   gyro_aks_y,   gyro_aks_z,   gyro_rts_x,
gyro_rts_y, gyro_rts_z, gyro_temperature]

return msg;

```

LAMPIRAN D

Surat Edaran WR I Nomor :SE/12/UNJANI/I/2021

Tentang

Ketentuan Unggah Mandiri Tugas Akhir bagi Mahasiswa Unjani

- Memakai Watermark logo Unjani disetiap halaman (**ukuran 12 cm Berwarna**)
 - Cover berwarna dan di simpan dalam format JPG ukuran maksimum 500 Kb
 - Pindai Lembar Pernyataan Bebas Plagiasi yang sudah ditandatangani diatas Materai Rp 10.000 oleh Dosen Pembimbing dan Mahasiswa dalam format PDF ukuran maksimum file 1 MB
 - Pindai Lembar Izin Publikasi yang sudah ditandai tangan oleh Dosen dan Mahasiswa
 - Abstrak dengan dua bahasa (Bahasa Indonesia dan Bahasa Inggris dalam bentuk PDF ukuran masimum file 500 Kb)
 - Isi BAB I - V Maksimum ukuran file 5 MB
 - Daftar Pustaka dan Lampiran-lampiran dalam bentuk PDF ukuran maksimum 1 MB
 - Soft file tugas Akhir di simpan dalam Compack Disk (CD) dan tempat CD berbentuk kotak yang sudah ditanda tangani oleh Dosem Pembimbing dan Mahasiswa
- 4 Apabila mahasiswa sudah unggah Laporan Tugas Akhir ke Website Perpustakaan Pusat Unjani dan Perpustakaan Fakultas Teknik, mahasiswa tersebut akan mendapatkan Report berupa Surat Keterangan Penyerahan Tugas Akhir ke email masing-masing.
- 5 Surat Keterangan Penyerahan Tugas Akhir harap dibawa saat penyerahan CD Tugas ke Perpustakaan Pusat Unjani saat verifikasi Data.

LAMPIRAN E

Surat Edaran WR I Nomor :SE/33/UNJANI/I/2021

Tentang

Revisi Peraturan Pengumpulan Tugas Akhir di Perpustakaan Pusat bagi Mahasiswa Unjani Nomor :SE/10/UNJANI/2021

- Memakai Watermark logo Unjani disetiap halaman **ukuran 4x4 cm Berwarna)**
 - Cover berwarna dan di simpan dalam format JPG ukuran maksimum 500 Kb
 - Pindai Lembar Pernyataan Bebas Plagiasi yang sudah ditanda tangani diatas Materai Rp 10.000 oleh Mahasiswa dalam format PDF ukuran maksimum file 1 MB
 - Pindai Lembar Izin Publikasi yang bersifat opsional (apabila dosen pembimbing tidak berkenan untuk dipublikasikan, maka yang mendatangi hanya mahasiswa bersangkutan.)
 - Abstrak dengan dua bahasa (Bahasa Indonesia dan Bahasa Inggris dalam bentuk PDF ukuran masimum file 500 Kb)
 - Isi BAB I - V Maksimum ukuran file 5 MB
 - Daftar Pustaka dan Lampiran-lampiran dalam bentuk PDF ukuran maksimum 1 MB
 - Soft file tugas Akhir di simpan dalam Compack Disk (CD) dan tempat CD berbentuk kotak yang sudah ditanda tangani oleh Dosem Pembimbing dan Mahasiswa
- 4 Apabila mahasiswa sudah unggah Laporan Tugas Akhir ke Website Perpustakaan Pusat Unjani dan Perpustakaan Fakultas Teknik, mahasiswa tersebut akan mendapatkan Report berupa Surat Keterangan Penyerahan Tugas Akhir ke email masing-masing.

- 5 Surat Keterangan Penyerahan Tugas Akhir harap dibawa saat penyerahan CD Tugas ke Perpustakaan Pusat Unjani saat verifikasi Data.



LAMPIRAN F

Surat Edaran WR I Nomor :SE/12/UNJANI/I/2021

Tentang

Ketentuan Unggah Mandiri Tugas Akhir bagi Mahasiswa Unjani

- Memakai Watermark logo Unjani disetiap halaman (**ukuran 12 cm Berwarna**)
 - Cover berwarna dan di simpan dalam format JPG ukuran maksimum 500 Kb
 - Pindai Lembar Pernyataan Bebas Plagiasi yang sudah ditandatangani diatas Materai Rp 10.000 oleh Dosen Pembimbing dan Mahasiswa dalam format PDF ukuran maksimum file 1 MB
 - Pindai Lembar Izin Publikasi yang sudah ditandai tangan oleh Dosen dan Mahasiswa
 - Abstrak dengan dua bahasa (Bahasa Indonesia dan Bahasa Inggris dalam bentuk PDF ukuran masimum file 500 Kb)
 - Isi BAB I - V Maksimum ukuran file 5 MB
 - Daftar Pustaka dan Lampiran-lampiran dalam bentuk PDF ukuran maksimum 1 MB
 - Soft file tugas Akhir di simpan dalam Compack Disk (CD) dan tempat CD berbentuk kotak yang sudah ditanda tangani oleh Dosem Pembimbing dan Mahasiswa
- 4 Apabila mahasiswa sudah unggah Laporan Tugas Akhir ke Website Perpustakaan Pusat Unjani dan Perpustakaan Fakultas Teknik, mahasiswa tersebut akan mendapatkan Report berupa Surat Keterangan Penyerahan Tugas Akhir ke email masing-masing.
- 5 Surat Keterangan Penyerahan Tugas Akhir harap dibawa saat penyerahan CD Tugas ke Perpustakaan Pusat Unjani saat verifikasi Data.

LAMPIRAN D Tim Penyusun Standarisasi Pedoman Draft Tugas Akhir

Penanggung jawab	: Ketua Prodi
	: Sekretaris Prodi
Ketua Tim	: Udin Komarudin, S.T., M.T.
Anggota	: Sunubroto, S.T., M.T.
	: Ahmad Daelami, S.T., M.M.
	: Ade Sena Permana, S.T., M.T.
	: M. Reza Hidayat, S.T., M.T.
	: Giri Angga Setia, S.T., M.T.
	: Fauzia Haz, S.T., M.T.

