TUGAS AKHIR - EC234801

PERANCANGAN SISTEM KONTROL MOTOR KURSI RODA SECARA NIRKABEL BERBASIS ESP32

I Putu Haris Setiadi Ekatama
NRP 0721 19 4000 0046

Dosen Pembimbing

Dr. Eko Mulyanto Yuniarno, S.T., M.T.

NIP 19680601 199512 1 009

Program Studi Strata 1 (S1) Teknik Komputer
Departemen Teknik Komputer
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2024

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - EC234801

PERANCANGAN SISTEM KONTROL MOTOR KURSI RODA SECARA NIRKABEL BERBASIS ESP32

I Putu Haris Setiadi Ekatama
NRP 0721 19 4000 0046

Dosen Pembimbing

Dr. Eko Mulyanto Yuniarno, S.T., M.T.

NIP 19680601 199512 1 009

Program Studi Strata 1 (S1) Teknik Komputer
Departemen Teknik Komputer
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2024

[Halaman ini sengaja dikosongkan]

FINAL PROJECT - EC234801

DESIGNING A WIRELESS CONTROL SYSTEM FOR WHEELCHAIR MOTORS BASED ON ESP32

I Putu Haris Setiadi Ekatama
NRP 0721 19 4000 0046

Advisor

Dr. Eko Mulyanto Yuniarno, S.T., M.T.

NIP 19680601 199512 1 009

Undergraduate Study Program of Computer Engineering
Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya
2024

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PERANCANGAN SISTEM KONTROL MOTOR KURSI RODA SECARA NIRKABEL BERBASIS ESP32

TUGAS AKHIR
Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Teknik pada
Program Studi S-1 Teknik Komputer
Departemen Teknik Komputer
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh: **I Putu Haris Setiadi Ekatama**
NRP. 0721 19 4000 0046

Disetujui oleh Tim Penguji Tugas Akhir:

Dr. Eko Mulyanto Yuniarno, S.T., M.T. NIP: 19680601 199512 1 009	(Pembimbing I) .....
Dion Hayu Fandiantoro, S.T., M.Eng.. NIP: 19942020 11064	(Penguji I) .....
Dr. Arief Kurniawan, S.T., M.T.. NIP: 19740907 200212 1 001	(Penguji II) .....
Arta Kusuma Hernanda, S.T., M.T.. NIP: 19962023 11024	(Penguji III) .....

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T..
NIP. 19700313 199512 1 001

SURABAYA
Januari, 2024

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

DESIGNING A WIRELESS CONTROL SYSTEM FOR WHEELCHAIR MOTORS BASED ON ESP32

FINAL PROJECT
Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Engineering at
Undergraduate Study Program of Computer Engineering
Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology

By: **I Putu Haris Setiadi Ekatama**
NRP. 0721 19 4000 0046

Approved by Final Project Examiner Team:

Dr. Eko Mulyanto Yuniarno, S.T., M.T. NIP: 19680601 199512 1 009	(Advisor I) .....
Dion Hayu Fandiantoro, S.T., M.Eng.. NIP: 19942020 11064	(Examiner I) .....
Dr. Arief Kurniawan, S.T., M.T.. NIP: 19740907 200212 1 001	(Examiner II) .....
Arta Kusuma Hernanda, S.T., M.T.. NIP: 19962023 11024	(Examiner III) .....

Acknowledged,
Head of Computer Engineering Department F-ELECTICS - ITS

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T..
NIP. 19700313 199512 1 001

SURABAYA
January, 2024

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa / NRP	:	I Putu Haris Setiadi Ekatama / 0721 19 4000 0046
Departemen	:	Teknik Komputer
Dosen Pembimbing / NIP	:	Dr. Eko Mulyanto Yuniarno, S.T., M.T. / 19680601 199512 1 009

Dengan ini menyatakan bahwa Tugas Akhir dengan judul ”PERANCANGAN SISTEM KONTROL MOTOR KURSI RODA SECARA NIRKABEL BERBASIS ESP32” adalah hasil karya sendiri, berfsifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, January 2024

Mengetahui Dosen Pembimbing	Mahasiswa
Dr. Eko Mulyanto Yuniarno, S.T., M.T. NIP. 19680601 199512 1 009	I Putu Haris Setiadi Ekatama NRP. 0721 19 4000 0046

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP	:	I Putu Haris Setiadi Ekatama / 0721 19 4000 0046
Department	:	Computer Engineering
Advisor / NIP	:	Dr. Eko Mulyanto Yuniarno, S.T., M.T. / 19680601 199512 1 009

Hereby declared that the Final Project with the title of "*DESIGNING A WIRELESS CONTROL SYSTEM FOR WHEELCHAIR MOTORS BASED ON ESP32*" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with provisions that apply at Sepuluh Nopember Institute of Technology.

Surabaya, January 2024

Acknowledged
Advisor

Dr. Eko Mulyanto Yuniarno, S.T., M.T.
NIP. 19680601 199512 1 009

Student

I Putu Haris Setiadi Ekatama
NRP. 0721 19 4000 0046

[Halaman ini sengaja dikosongkan]

ABSTRAK

Nama Mahasiswa : I Putu Haris Setiadi Ekatama
Judul Tugas Akhir : PERANCANGAN SISTEM KONTROL MOTOR KURSI RODA SECARA NIRKABEL BERBASIS ESP32
Pembimbing : 1. Dr. Eko Mulyanto Yuniarno, S.T., M.T.

Kelumpuhan merupakan suatu keadaan dimana penderita mengalami pelemahan fungsi pada anggota tubuhnya sehingga penderita tidak bertenaga atau tidak menggerakkan anggota tubuh sebagaimana mestinya. Terdapat beberapa kondisi yang dapat mengakibatkan kelumpuhan, mulai dari penyakit seperti stroke hingga kecelakaan. Seseorang yang mengalami kelumpuhan sering kali memiliki permasalahan dalam hal mobilitas sehari-hari. Mereka memerlukan alat tambahan agar dapat beraktivitas, salah satunya adalah kursi roda. Hingga saat ini telah terdapat kursi roda elektrik yang dikendalikan dengan menggunakan *joystrick*. Akan tetapi penggunaan *joystick* belum dapat menjawab permasalahan dari seseorang yang mengalami kelumpuhan. Karena bagi orang yang mengalami kelumpuhan pada bagian lengan akan kekusahan dalam mengendalikan kursi roda elektrik berjenis ini. Pada penelitian ini telah dikembangkan kontroler kursi roda elektrik yang dapat digerakkan melalui teknologi visi komputer, baik dengan pose tangan maupun gestur kepala. Integrasi teknologi ini dapat menjadi solusi yang inovatif terhadap permasalahan yang sedang dihadapi. Pemilihan ESP32 sebagai mikrokontroler urama menjadi langkah strategis, karena kemampuanya dalam mengatur kerja motor dengan presisi. Selain berfungsi sebagai kontroler motor, ESP32 juga berperan sebagai perangkat penerima data dari komputer yang dilengkapi dengan teknologi visi komputer. Dari hasil pengujian, didapatkan suatu kesimpulan bahwa pengiriman dari visi komputer sebaiknya dianalogikan dengan 1 karakter huruf dan ditransmisikan menggunakan WiFi. Hal ini dilakukan karena transmisi data yang berisikan 1 huruf dan ditransmisikan melalui WiFi memiliki waktu delay yang terbaik, yaitu 0.03499708571 detik. Melalui integrasi ini, diharapkan bahwa kontroler motor dapat beroperasi secara sinergis dengan informasi yang diterima dari komputer dan menciptakan sebuah sistem yang efisien dan responsif.

Kata Kunci: Kursi Roda, *Convolutional Neural Network*, Mediapipe, ESP32, WiFi.

[Halaman ini sengaja dikosongkan]

ABSTRACT

Name : I Putu Haris Setiadi Ekatama
Title : *DESIGNING A WIRELESS CONTROL SYSTEM FOR WHEELCHAIR MOTORS BASED ON ESP32*
Advisors : 1. Dr. Eko Mulyanto Yuniarno, S.T., M.T.

Paralysis is a condition where a person experiences a weakening of the functions in their body parts, causing them to lack energy or be unable to move their limbs as they should. There are several conditions that can lead to paralysis, ranging from diseases such as stroke to accidents. Individuals experiencing paralysis often face challenges in their daily mobility. They require additional tools to be able to carry out activities, one of which is a wheelchair. Electric wheelchairs controlled by a joystick have been developed to date. However, the use of a joystick may not address the issues faced by someone experiencing paralysis. This is because individuals with paralysis in their arms may struggle to control this type of electric wheelchair. In this research, a controller for an electric wheelchair has been developed that can be operated through computer vision technology, either with hand poses or head gestures. The integration of this technology can be an innovative solution to the problems being faced. Choosing ESP32 as the main microcontroller is a strategic step due to its ability to precisely control the wheelchair's motor. In addition to functioning as a motor controller, ESP32 also serves as a data receiver from the computer equipped with computer vision technology. From the test results, it is concluded that the transmission from computer vision should be analogized to 1 character letter and transmitted using WiFi. This is done because transmitting data containing 1 letter and using WiFi has the best delay time, which is 0.03499708571 seconds. Through this integration, it is expected that the motor controller can operate synergistically with the information received from the computer, creating an efficient and responsive system.

Keywords: *Wheelchair*, *Convolutional Neural Network*, *Mediapipe*, *ESP32*, *WiFi*.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kehadirat Tuhan Yang Maha Esa, atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penelitian ini yang berjudul ”PERANCANGAN SISTEM KONTROL MOTOR KURSI RODA SECARA NIRKABEL BERBASIS ESP32”.

Penelitian ini disusun dalam rangka pemenuhan Tugas Akhir sebagai syarat kelulusan Mahasiswa ITS. Oleh karena itu, penulis mengucapkan banyak terima kasih kepada

1. Bapak Dr.Supeno Mardi Susiko Nugroho, ST.,MT, selaku Kepala Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember
2. Bapak Dr. Eko Mulyanto Yuniarno, S.T., M.T. selaku Dosen Pembimbing telah memberikan arahan selama pengerjaan tugas akhir ini
3. Bapak Dion Hayu Fandiantoro, S.T., M.Eng. selaku dosen penguji I, Bapak Dr. Arief Kurniawan, S.T., M.T. selaku dosen penguji II dan Bapak Arta Kusuma Hernanda, S.T., M.T. selaku dosen penguji III yang telah memberikan saran dan revisi agar pengerjaan Buku Tugas Akhir ini dapat menjadi lebih baik
4. Bapak-Ibu dosen pengajar Departemen Teknik Komputer, atas ilmu dan pengajaran yang telah diberikan kepada penulis selama ini
5. Nenek, bibi dan ibu yang senantiasa membantu serta membiayai pendidikan sedari kecil hingga sarjana
6. Teman - teman lab B300 dan B201 serta teman - teman Departemen Teknik Komputer lainnya

Akhir kata, semoga penelitian ini dapat memberikan manfaat kepada banyak pihak, penulis menyadari jika skripsi ini masih belum sempurna, dikarenakan keterbatasan ilmu yang dimiliki. Untuk itu penulis mengharapkan saran dan kritik yang bersifat membangun kepada penulis untuk menuai hasil yang lebih baik lagi.

Surabaya, Januari 2024

I Putu Haris Setiadi Ekatama

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR PROGRAM

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Menurut Kamus Besar Bahasa Indonesia, lumpuh merupakan melemahnya fungsi anggota badan sehingga tidak bertenaga atau tidak dapat digerakkan lagi sebagaimana mestinya (Daring, 2016). Otot beserta tulang, saraf, serta jaringan penghubung antara otot, tulang dan saraf memiliki peran yang penting dalam mengendalikan gerak tubuh manusia. Apabila salah satu jaringan mengalami gangguan makan akan terjadi kelumpulan, baik kelumpuhan sementara maupun kelumpuhan permanen.

Terdapat beberapa kondisi yang dapat mengakibatkan kelumpuhan, seperti penyakit stroke yang dapat menyebabkan kelumpuhan pada salah satu sisi wajah, lengan serta tungkai, *Bell's Palsy* yang dapat menyebabkan kelumpuhan pada salah satu sisi wajah tanpa disertai kelumpuhan pada anggota tubuh yang lain, cedera otak yang dapat memicu kelumpuhan pada setiap bagian tubuh sesuai bagian otak yang rusak, polio yang menyebabkan kelumpuhan pada lengan, tungkai, serta otot pernapasan, dan masih banyak kondisi yang menyebabkan kelumpuhan (Pansawira, 2022).

Seseorang yang mengalami kelumpuhan sering kali mengalami permasalahan dalam hal mobilitas sehari-hari. Mereka memerlukan alat tambahan untuk dapat beraktivitas sehari-hari, salah satunya adalah kursi roda. Hingga saat ini sudah terdapat kursi roda elektrik yang dikendalikan dengan menggunakan *joystick* (Choi et al., 2019). Akan tetapi penggunaan *joystick* belum dapat menjawab permasalahan dari seseorang yang mengalami kelumpuhan. Karena bagi orang yang mengalami kelumpuhan pada bagian lengan akan kekusahan dalam mengendalikan kursi roda elektrik berjenis ini.

Dalam menghadapi permasalahan kelumpuhan, sangat penting untuk mencari solusi yang dapat meningkatkan kemandirian para penderita. Salah satu pendekatan yang menjanjikan adalah memanfaatkan teknologi canggih, seperti visi komputer yang dapat diintegrasikan dengan sistem tertanam. Dengan menggabungkan kedua teknologi ini, diharapkan dapat diciptakan solusi inovatif yang memungkinkan para penderita kelumpuhan untuk tetap dapat bermobilitas secara mandiri.

Visi komputer merupakan bidang keilmuan yang memungkinkan komputer dapat "melihat" (Tian et al., 2020). Teknologi ini menggunakan kamera untuk mengidentifikasi, melacak, hingga mengukur target untuk pemrosesan citra lebih lanjut. Visi komputer memberikan kemampuan untuk mengenali dan memahami lingkungan sekitar. Sedangkan sistem tertanam dapat diatur secara personal untuk memenuhi kebutuhan spesifik sesuai dengan permasalahan yang dihadapi.

Integrasi teknologi ini dapat menjadi solusi inovatif terhadap permasalahan yang dihadapi. Dalam rangka mengatasi tantangan ini, penelitian akan difokuskan pada pengembangan kontroler motor yang dapat secara optimal berinteraksi dengan teknologi visi komputer. Pemilihan ESP32 sebagai mikrokontroler utama menjadi langkah strategis, karena kemampuannya dalam

mengatur dengan presisi kerja motor. Tidak hanya berfungsi sebagai kontroler motor, ESP32 juga akan berperan sebagai perangkat penerima data dari komputer yang dilengkapi dengan teknologi visi komputer. Melalui integrasi ini, diharapkan bahwa kontroler motor dapat beroperasi secara sinergis dengan informasi yang diterima dari komputer dan menciptakan sebuah sistem yang efisien dan responsif.

1.2 Permasalahan

Berdasarkan hal yang telah dipaparkan pada latar belakang, untuk dapat mengatur kerja motor dari kursi roda yang terintegrasi dengan teknologi visi komputer maka diperlukan kontroler yang dapat digunakan sebagai perantara antara keduanya.

1.3 Tujuan

Tujuan dari tugas akhir ini adalah untuk mengembangkan kontroler motor kursi roda yang dapat terintegrasi dengan teknologi visi komputer.

1.4 Batasan Masalah

Untuk memfokuskan permasalahan yang diangkat maka dilakukan pembatasan masalah. Batasan-batasan masalah tersebut diantaranya:

1. Mikrokontroler yang digunakan adalah ESP32 Devkit V1.
2. Laptop atau Jetson Nano digunakan sebagai pengolah data visi komputer.
3. Pengiriman data visi komputer ke ESP32 menggunakan WiFi maupun Bluetooth.
4. Pengujian yang dilakukan adalah membandingkan tingkat delay pengiriman data dari laptop ke ESP32 yang menggunakan WiFi dengan yang menggunakan Bluetooth.

1.5 Manfaat

Manfaat dari penelitian ini adalah untuk memungkinkan teknologi visi komputer agar dapat mengontrol gerak dari kursi roda. Sehingga para pengembang dapat mengembangkan model dari *machine learning* mereka untuk diaplikasikan sebagai kontrol kursi roda.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

2.1.1 Kontrol Kursi Roda Menggunakan Sinyal Suara Melalui Bluetooth

Pada tahun 2023 telah dilakukan penelitian yang berjudul ”Kontrol Kursi Roda Menggunakan Sinyal Suara Melalui Bluetooth” oleh Arief Wisaksono, Rachmad Aditya Pratama, dan Hindarto hindarto dari Departemen Teknik Elektro, Fakultas Sains dan Teknologi Universitas Muhammadiyah Sidoarjo (Wisaksono, Pratama, et al., 2023).

Pada penelitian ini dapat disimpulkan bahwa pengujian koneksi Bluetooth dan Android dapat berjalan secara optimal. Sehingga input dari Android bisa terkirim ke rangkaian Arduino Uno. Hasil pengujian koneksi memiliki waktu delay selama 4 detik hingga 6 detik. Pengujian baterai 12 Volt memiliki deviasi sebesar 0,43 serta akurasi sebesar 96,7%. Hal ini disebabkan karena hasil dari pengukuran lebih besar daripada tegangan yang diperlukan. Akan tetapi hal tersebut tidak mempengaruhi sistem kerja alat karena tegangan 12 Volt merupakan tegangan minimum alat.

2.1.2 Rancang Bangun Kursi Roda Elektrik Dengan Sistem Kontrol *Joystick* Dan *Smartphone* Android

Pada tahun 2023 telah dilakukan penelitian yang berjudul ”Rancang Bangun Kursi Roda Elektrik Dengan Sistem Kontrol *Joystick* dan *Smartphone* Android” oleh Bayu Ahityanto Wicaksono dari Program Studi Diploma IV Rekayasa Perancangan Mekanik, Sekolah Vokasi Universitas Diponegoro (Wicaksono, 2023).

Pada penelitian ini didapatkan kesimpulan bahwa kursi roda konvensional yang dijadikan kursi roda elektrik berhasil dijalankan dengan kecepatan maksimal 2 km/h sesuai perencanaan. Kursi roda elektrik dapat dikontrol dengan *joystick* maupun dari aplikasi yang berada di *smartphone* android. Kursi roda elektrik dapat berjalan dengan beban maksimal 80 kg. Terdapat beberapa saran dari penulis seperti menambahkan sandaran kepala agar pengguna lebih nyaman di kursi roda elektrik, serta pembuatan sistem aplikasi untuk pengguna *smartphone* dari Apple.

2.1.3 *Wheelchair Control Using Bluetooth-Based Electromyography Signals*

Telah dilakukan penelitian yang berjudul *”Wheelchair Control Using Bluetooth-Based Electromyography Signals”* oleh Yoga Eko Prasetyo dari Program Studi Teknik Elektro dan Hindarto Hindarto dari Program Studi Informatika Universitas Muhammadiyah Sidoarjo (Prasetyo et al., 2023).

Pada penelitian ini didapatkan kesimpulan bahwa durasi tunggu dari bluetooth master dengan bluetooth slave sebesar 4 detik hingga 5 detik. Pengujian sensor elektromiografi dapat berjalan dengan normal dan menghasilkan nilai yang berbeda ketika otot berkontraksi maupun relaksasi.

2.1.4 Prototipe Kursi Roda Elektrik Dengan Kendali *Joystick* dan *Smartphone*

Pada tahun 2019 telah dilakukan penelitian yang berjudul "Prototipe Kursi Roda Elektrik Dengan Kendali *Joystick* dan *Smartphone*" oleh Andy Sadewa Junior dan Fatchul Arifin dari Program Studi Teknik Elektronika, Fakultas Teknik Universitas Negeri Yogyakarta (Junior & Arifin, 2019).

Pada penelitian ini didapatkan kesimpulan bahwa total *error* yang dihasilkan dari pengujian tegangan motor kiri dan kanan adalah sebesar 0,144% dan rata-rata *error* yang didapatkan adalah sebesar 0,024% pada keseluruhan pengujian yang dilakukan. Pada pengujian bluetooth didapatkan kesimpulan bahwa jangkauan pengiriman optimal dari bluetooth apabila tidak ada penghalang adalah sebesar 1 meter hingga 10 meter.

2.1.5 *Vision-based Head Posture Control Wheelchair System Research*

Pada tahun 2023 telah dilakukan penelitian yang berjudul *"Vision-based Head Posture Control Wheelchair System Research"* oleh Pengyu Gao dari *School of Information Engineering*, *Shenyang University of Chemical Technology* bersama Haitao Luo dan Yuxin Li dari *Department of Space Automation Technology*, *Shenyang Institute of Automation*, *Chinese Academy of Sciences* (Gao et al., 2023).

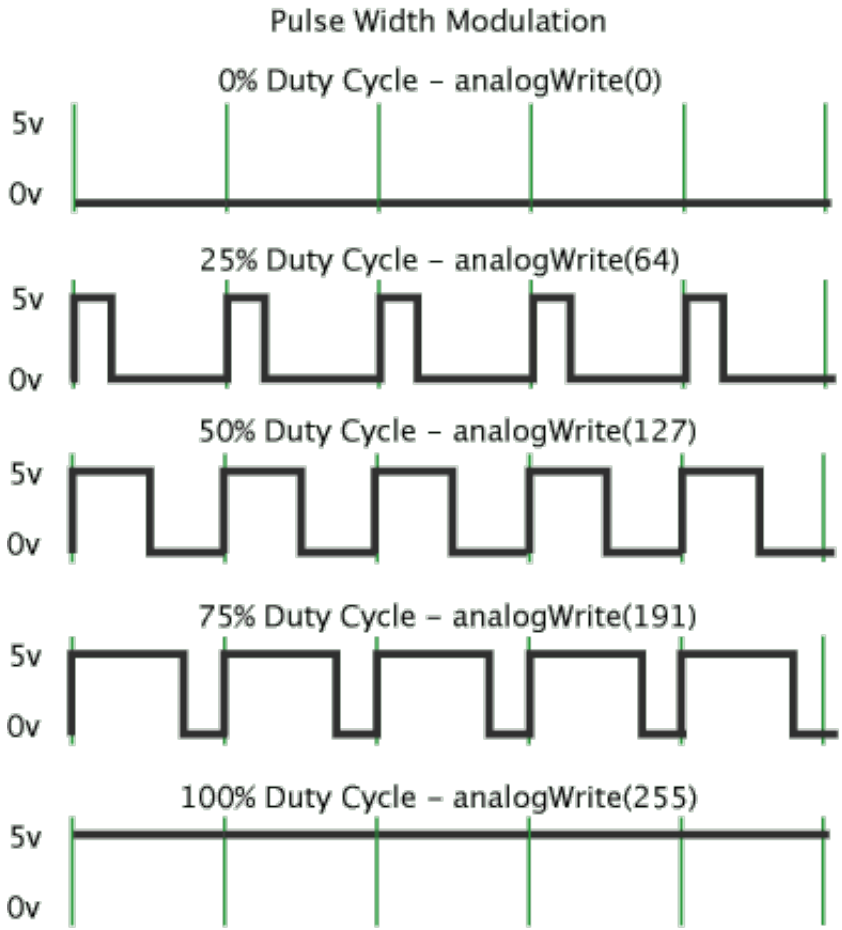
Pada penelitian ini didapatkan kesimpulan bahwa sistem pendeteksi pose kepada berbasis Mediapipe berhasil dilakukan menggunakan metode pemodelan matematis yang dikombinasikan dengan fungsi trigonometri untuk memperkirakan sudut wajah dan arah pose kepala. Namun masih terdapat beberapa *error* pada pengenalan citra karena transmisi input citra bersifat *realtime*.

2.2 Teori/Konsep Dasar

2.2.1 *Pulse Width Modulation* (PWM)

Pulse Width Modulation, atau PWM, adalah teknik untuk menghasilkan sinyal analog menggunakan metode digital (Hirzel, 2024). Dalam metode ini, kontrol digital digunakan untuk menghasilkan sinyal yang beralih antara kondisi *on* dan *off*. Sinyal ini kemudian dapat disesuaikan untuk mensimulasikan tegangan di antara nilai VCC penuh (misalnya, 5V pada *board* Arduino UNO, atau 3.3 V pada *board* MKR) dan VCC rendah (0V) dengan mengatur proporsi waktu sinyal dalam kondisi *on* dan *off*. Durasi sinyal dalam kondisi hidup disebut sebagai lebar pulsa (*pulse widht*). Untuk mendapatkan nilai analog yang bervariasi, lebar pulsa tersebut dapat diubah atau dimodulasi. Misalnya, dengan mengulangi pola *on-off* dengan cepat pada sebuah LED, hasilnya akan terlihat seperti tegangan yang stabil antara 0V dan VCC yang mengatur kecerahan LED tersebut.

Pada Gambar 2.1 garis hijau mewakili periode waktu yang teratur. Durasi atau periode ini adalah kebalikan dari frekuensi PWM. Dengan kata lain, dengan frekuensi PWM Arduino sekitar 500HZ maka garis hijau akan memiliki panjang 2 milidetik setiap satunya. Pemanggilan analogWrite() berkisar antara 0-255, sehingga analogWrite(255) akan meminta siklus tugas 100%(selalu menyala) dan analogWrite(127) akan meminta siklus tugas 50% (menyala setengah waktu).



Gambar 2.1: *Pulse Width Modulation*(PWM)

Pada beberapa mikrokontroler, PWM hanya tersedia pada pin tertentu. Silakan pertimbangkan diagram *pinout* pada *board* yang akan digunakan untuk mengetahui pin mana saja yang dapat digunakan sebagai PWM. Pin yang dapat digunakan sebagai PWM biasanya dilambangkan dengan tanda tilde (~).

2.2.2 JSON

JavaScript Object Notation (JSON) merupakan format pertukaran data yang ringan dan mudah digunakan (Data, n.d.). JSON mudah dipahami oleh manusia dan mesin. JSON didasarkan pada sebagian kecil dari Standar Bahasa Pemrograman JavaScript ECMA-262 Edisi ke-3 yang dirilis pada bulan Desember 1999. JSON menggunakan konvensi yang familiar bagi para programmer dengan berbagai latar bahasa pemrograman, termasuk C, C++, C#, Java, JavaScript, Perl, hingga Python (Org, 2017). Karena sifatnya yang fleksibel ini membuat JSON menjadi bahasa pertukaran data yang sangat ideal.

JSON terdiri dari 2 struktur utama, yaitu *key* dan *value*. Sekumpulan *key-value* dapat dianggap sebagai objek pada beberapa bahasa pemrograman. Selain itu JSON juga dapat terdiri dari sebuah daftar nilai yang diurutkan (*ordered list of values*). Struktur ini dapat diwakili sebagai *array, vector, list,* maupun *sequence* pada berbagai bahasa pemrograman.

2.2.3 Bluetooth

Bluetooth merupakan sebuah standar terbuka untuk konektivitas nirkabel dengan dukungan tinggi dari industri komputer dan perangkat seluler (Sairam et al., 2002). Bluetooth diciptakan pada tahun 1994 oleh L. M. Ericsson dari Swedia.

Bluetooth Classic radio, juga dikenal sebagai Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR), adalah jenis radio dengan konsumsi daya rendah yang mengirimkan data melalui 79 saluran di band frekuensi 2.4 GHz yang tidak berlisensi untuk penggunaan industri, ilmiah, dan medis (ISM) (SIG, 2024). Fitur komunikasi antar-perangkat titik ke titik, Bluetooth Classic umumnya digunakan untuk mengaktifkan streaming audio nirkabel dan telah menjadi standar protokol radio untuk perangkat seperti speaker nirkabel, headphone, dan sistem hiburan di mobil. Selain itu, radio Bluetooth Classic juga memungkinkan aplikasi transfer data, termasuk pencetakan melalui perangkat seluler.

Radio Bluetooth Low Energy (BLE) dirancang untuk beroperasi dengan sangat efisien dalam penggunaan daya. Dengan mentransmisikan data melalui 40 saluran di band frekuensi ISM 2.4 GHz yang tidak berlisensi, radio Bluetooth LE memberikan fleksibilitas yang besar bagi para pengembang untuk membuat produk yang sesuai dengan kebutuhan konektivitas pasar mereka. Bluetooth LE mendukung berbagai topologi komunikasi, mulai dari titik ke titik hingga siaran dan, yang terbaru, mesh, sehingga teknologi Bluetooth dapat mendukung pembuatan jaringan perangkat yang andal dan luas. Awalnya dikenal karena kemampuannya dalam komunikasi perangkat, Bluetooth LE kini juga banyak digunakan sebagai teknologi penentuan posisi perangkat untuk mengatasi permintaan yang semakin meningkat untuk layanan lokasi dalam ruangan yang akurat. Bluetooth LE sekarang dilengkapi dengan fitur yang memungkinkan satu perangkat menentukan keberadaan, jarak, dan arah perangkat lainnya. Perbandingan antara Bluetooth Low Energy dengan Bluetooth Classic dapat dilihat pada Tabel 2.1

Tabel 2.1: Tabel Perbandingan Bluetooth Low Energy dengan Bluetooth Classic

	Bluetooth Low Energy (BLE)	Bluetooth Classic
Frequency Band	2.4 GHz ISM Band	2.4 GHz ISM band
Channels	40 channels with 2 MHz spacing (3 advertising channels/37 data channels)	79 channels with 1 MHz spacing
Channel Usage	FHSS	FHSS
Modulation	GFSK	GFSK, π/4 DQPSK, 8 DPSK
Data Rate	LE 2M PHY: 2 Mb/s LE 1M PHY: 1 Mb/s LE Coded PHY (S=2): 500 Kb/s LE Coded PHY (S=8): 125 Kb/s	EDR PHY (8DPSK): 3 Mb/s EDR PHY (π/4 DQPSK): 2 Mb/s BR PHY (GFSK): 1Mb/s
Tx Power*	$\leq$ 100mW (+20 dBm)	$\leq$ 100mW (+20 dBm)
Rx Sensitivity	LE 2M PHY: $\leq$ -70 dBm LE 1M PHY: $\leq$ -70 dBm LE Coded PHY (S=2): $\leq$ -75 dBm LE Coded PHY (S=8): $\leq$ -82 dBm	$\leq$ -70 dBm

Data Transports	Asynchronous Connection-oriented Isochronous Connection-oriented Asynchronous Connectionless Synchronous Connectionless Isochronous Connectionless	Asynchronous Connection-oriented Synchronous Connection-oriented
Communication Topologies	Point-to-Point (including piconet) Broadcast Mesh	Point-to-Point (including piconet)
Positioning Features	Presence: Advertising Direction: Direction Finding(AoA/AoD) Distance: RSSI, HADM(Coming)	None

2.2.4 WiFi

WiFi merupakan teknologi jaringan nirkabel yang dapat menghubungkan perangkat dengan perangkat lainnya maupun terhubung dengan internet. WiFi terstandarisasi sebagai IEEE 802.11. Keuntungan terbesar dari WiFi adalah kesederhanaannya. Komputer dapat dihubungkan tanpa kabel baik dengan internet maupun dengan perangkat lainnya. Komputer terhubung ke jaringan menggunakan sinyal radio dengan radius sejauh 100 kaki.

Radio WiFi yang bekerja pada standar 802.11b dan 802.11g mengirimkan sinyal pada frekuensi 2.4GHz, sementara WiFi dengan standar 802.11a mengirimkan sinyal pada frekuensi 5GHz. Frekuensi yang lebih tinggi akan memungkinkan pengiriman data yang lebih tinggi.

Radio WiFi menggunakan teknik pengkodean yang jauh lebih efisien. Untuk standar 802.11a dan 802.11 g, teknik ini dikenal sebagai *Orthogonal Frequency-Division Multiplexing*(OFDM). Sedangkan untuk standar 802.11b, teknik ini disebut *Complementary Code Keying*(CCK).

Kartu WiFi 802.11b memiliki kemampuan untuk mentransmisikan sinyal pada tiga frekuensi yang berbeda secara langsung. 802.11b juga dapat membagi lebar pita radio yang tersedia menjadi banyak saluran dan dengan cepat berpindah-pindah frekuensi diantara mereka. Metode ini lebih tahan terhadap gangguan dan dapat memungkinkan banyak kartu WiFi beroperasi secara bersamaan tanpa saling mengganggu.

Radio WiFi dapat mengirimkan data yang besar pada setiap detiknya. Standar 802.11b dapat menangani hingga 11 Mb/detik. Sedangkan standar 802.11a dan 802.11g mampu menangani hingga 54 Mb/detik.

The Instituteof Electrical and Electronics Engineers (IEEE) telah membuat standar penamaan dan penomoran yang unik pada teknologi WiFi. Standar 802.11 berkaitan erat dengan jaringan nirkabel. Notasi a, b, dan g mengidentifikasikan variasi yang berbeda dari standar 802.11.

2.2.5 Arduino IDE

Arduino Integrated Development Environment (IDE) merupakan perangkat lunak sumber terbuka yang digunakan untuk menulis dan mengunggah kode ke mikrokontroler seperti Arduino maupun ESP buatan Espressif System. Arduino IDE berisikan editor teks untuk menulis kode program, toolbar dengan tombol untuk fungsi umum serta serangkaian menu. Program

yang ditulis menggunakan Arduino IDE disebut sketches. Sketches ditulis pada teks editor dan disimpan pada file dengan ekstensi .ino. Editor memiliki fitur untuk memotong dan menempel serta untuk mencari dan mengganti teks. Area pesan akan memberikan umpan balik ketika menyimpan dan mengekspor serta akan menampilkan error. Konsol menampilkan keluaran teks oleh Arduino IDE termasuk pesan error yang lengkap serta informasi lainnya (Söderby & Hylén, 2023a).

Gambar 2.2: Tampilan Arduino IDE 2

Gambar 2.2 merupakan tampilan dari Arduino IDE. Arduino IDE 2 dilengkapi dengan *sidebar* baru yang memudahkan pengguna dalam mengakses *tools* yang umum digunakan. *Verify/Upload* digunakan untuk *compile* dan *upload* program ke *Board* Arduino. *Select Board & Port* digunakan untuk mendeteksi *Board* Arduino beserta nomor port yang terhubung dengan Arduino. *Sketchbook* merupakan tempat untuk mencari semua *sketchbook* lokal yang disimpan pada komputer pengguna. Sebagai tambahan, pengguna dapat menguhubungkan Arduino IDE dengan Arduino Cloud sehingga pengguna dapat menyimpan *sketch* pada lingkungan daring. *Board Manager* merupakan kumpulan *board* arduino serta *board packages* pihak ketiga yang dapat diinstal (Söderby & Hylén, 2023b). *Library Manager* merupakan tempat untuk menginstal berbagai *library* yang mendukung pengembangan Arduino, baik yang dibuat oleh Arduino maupun dari komunitas (Söderby & Hylén, 2023c). *Debugger* berguna untuk menguji dan men-*debug* program secara *realtime*. *Search* digunakan untuk mencari kata kunci pada program yang sedang dimuat. *Open Serial Monitor* berguna untuk memuat Serial Monitor yang sangat berguna untuk pengembangan (Söderby, 2023).

Sketchbook merupakan tempat dimana file program akan disimpan. *Sketches* Arduino akan disimpan dengan tipe file .ino dan harus disimpan pada folder dengan nama folder yang sama dengan nama file. Sebagai contoh, file my_sketch.ino harus disimpan pada folder dengan nama folder my_sketch.

Dengan menggunakan *Library Manager*, pengguna dapat menemukan dan menginstal

ribuan *library* yang dapat membantu dalam mengembangkan proyek menggunakan . *Library* merupakan *extension* dari Arduino API yang dapat memudahkan pengembang. Sebagai contoh, para pengembang dapat mengontrol motor servo, membaca sensor tertentu, hingga menggunakan modul WiFi hanya dengan memanggil fungsi yang terdapat pada *library* (Söderby & Hylén, 2023c).

Serial Monitor merupakan alat yang memungkinkan pengguna untuk melihat *streaming* data pada *Board* Arduino yang terhubung. Pada Arduino versi 1, *tool* ini ditempatkan pada jendela yang terpisah, namun untuk memudahkan pengguna maka *tool* ini kini terintegrasi dengan editor (Söderby, 2023).

2.2.6 NVIDIA® Jetson Nano™

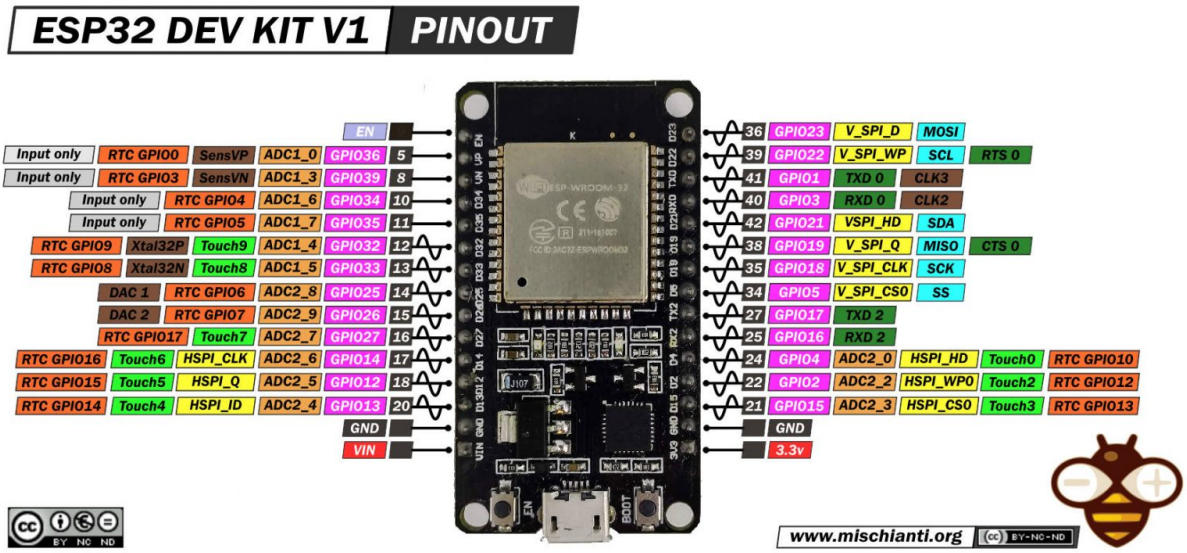
Gambar 2.3: Perangkat Jetson Nano

NVIDIA® Jetson Nano™ Developer Kit adalah komputer kecil dan kuat yang dapat digunakan untuk menjalankan beberapa *neural network* secara paralel untuk berbagai penerapan seperti klasifikasi gambar, deteksi objek, segmentasi, dan pemrosesan ucapan. Semuanya dikemas dalam platform yang mudah digunakan dan hanya membutuhkan daya 5 watt (Developer, 2023).

Perangkat ini memiliki pin *input* serta *output* yang berlimpah, mulai dari GPIO hingga pin CSI. Jumlah pin yang berlimpah ini sangat memudahkan para pengembang dalam menghubungkan berbagai perangkat tambahan seperti sensor untuk keperluan pengembangan aplikasi *Artificial Intelligence*. NVIDIA® Jetson Nano™ Developer Kit juga didukung dengan NVIDIA JetPack yang mencakup berbagai perangkat lunak seperti Sistem Operasi Linux, cuDNN, NVIDIA CUDA, TensorRT, dan juga *Board Support Package* (BSP) yang digunakan untuk keperluan *Deep Learning* serta visi komputer.

2.2.7 ESP32 Devkit V1

ESP32 Devkit V1 merupakan salah satu *development board* yang dibuat oleh DOIT untuk menjalankan modul ESP-WROOM-32 buatan Espressif (SmartArduino, 2022). *Development Board* ini berisi WiFi, Bluetooth, serta berdaya rendah hanya dengan 1 chip. Setiap pin dari ESP32 Devkit V1 dapat dilihat pada Gambar 2.4. Flash internal modul ESP32 disusun dalam satu area flash dengan 4096 *byte* pada masing-masing halaman. Alamat flash dimulai dari



Gambar 2.4: Perangkat ESP32 Devkit V1

0x00000.

Daya untuk menjalankan ESP32 disuplai melalui konektor Micro USB Tipe B atau dapat langsung melalui pin ”VIN”. Perangkat dapat beroperasi pada tegangan antara 6 Volt hingga 20 Volt. Jika menggunakan daya lebih dari 12 Volt maka regulator akan menjadi panas sehingga dapat merusak perangkat.

2.2.8 *Motor Driver H-Bridge*

H-Bridge merupakan salah satu jenis *driver motor* yang paling sering digunakan untuk mengendalikan motor listrik. Perangkat ini dapat digunakan untuk mengendalikan arah serta kecepatan putar motor. *Driver Motor H-Bridge* bekerja seperti saklar pada transistor. Transistor merupakan bagian utama dari perangkat ini. *Driver Motor H-Bridge* tersusun oleh sekumpulan transistor yang berfungsi sebagai pengendali motor, terutama yang memerlukan arus serta tegangan yang cukup besar (Muhammad, 2018).

Gambar 2.5: *H-Bridge Motor Driver*

Penelitian ini menggunakan *Motor Driver H-Bridge* dari MRI. Perangkat ini dapat digunakan untuk mengatur kecepatan dan arah putar motor *brushed* dengan kapasitas maksimal 50 A pada tegangan 48 V. Driver motor ini dilengkapi dengan satu kanal PWM yang dapat digunakan untuk mengatur kecepatan putar motor dengan cara mengubah lebar pulsa yang dikirimkan ke motor. Lebar pulsa PWM dapat diatur menggunakan potensiometer maupun melalui kontrol sinyal eksternal. Perangkat ini juga memiliki kemampuan untuk mengatur arah putar motor melalui kedua pin dir. Dengan mengganti keadaan *input* kontrol maka pengguna dapat mengubah arah putaran motor secara mudah. Gambar 2.5 merupakan perangkat yang digunakan pada penelitian ini.

2.2.9 Kursi Roda Elektrik KY-123

Gambar 2.6: Kursi Roda Elektrik KY-123

Kursi roda elektrik sebagian besar terdiri dari rangka kursi roda, alat kontrol gerak kursi roda, motor listrik, serta unit baterai. Alat ini dapat dioperasikan dengan fleksibel, mudah, dan sederhana sehingga tidak memerlukan tenaga yang besar dari pengguna jika dibandingkan dengan kursi roda biasa. Kursi roda ini dapat dioperasikan dengan satu tangan oleh pengguna yang mengalami kelumpuhan hemiplegia. Alat ini juga dapat membantu manula yang susah dalam melakukan mobilisasi.

Perusahaan MySella telah mengembangkan beberapa model kursi roda elektrik sesuai dengan skenario dan syarat pengaplikasian yang berbeda. Gambar 2.6 merupakan salah satu kursi roda elektrik buatan MySella yang digunakan pada penelitian ini. Kursi roda ini sudah memenuhi persyaratan standar nasional GB 12996-2012. Berikut Tabel 2.2 yang menjadi persyaratan standar nasional GB12996-2012.

Tabel 2.2: Standar Nasional GB 12996-2012

Parameter	Indikator Kinerja
Kecepatan Maksimal	≤ 6 Km/Jam
Derajat Kemiringan	6°- 8°
Ketinggian Penghalang	≤ 40 mm

Jarak Tempuh	≤ 20 Km
Radius Putar Minimal	1.2 m
Kemampuan Docking	9°
Lebar Parit	100 mm
Suhu Kerja Optimal	-5°C - 40°C

2.2.10 Motor DC MY1016Z

Gambar 2.7: Motor DC MY1016Z

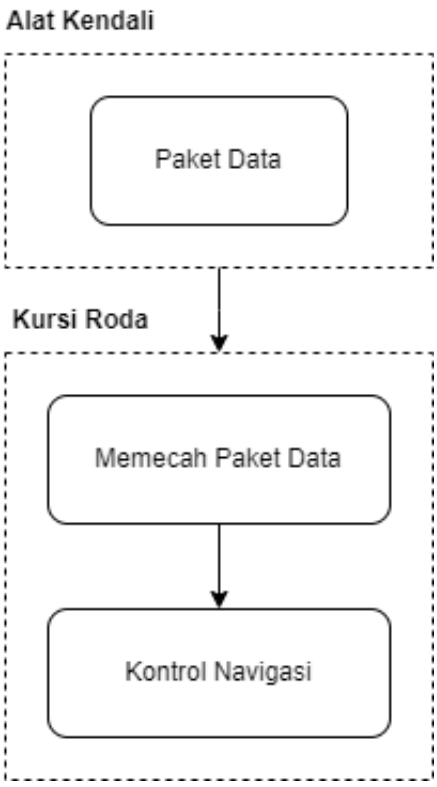
Motor DC merupakan jenis dari motor listrik yang menggunakan arus searah untuk menghasilkan gerakan. MY1016Z merupakan *geared* motor DC yang menggunakan mekanisme roda gigi didalamnya. Motor DC ini sering digunakan pada sepeda listrik serta kendaraan sejenisnya. Kursi roda buatan MySella yang bertipe KY-123 juga menggunakan motor DC MY1016Z seperti pada Gambar 2.7.

BAB III
METODOLOGI

Penelitian ini dilaksanakan sesuai dengan desain sistem berikut ini beserta implementasinya. Desain sistem merupakan konsep dari pembuatan dan perencangan infrastruktur dan kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan.

3.1 Deskripsi Sistem

Tugas akhir ini merupakan penelitian yang mengintegrasikan teknologi visi komputer agar dapat mengontrol gerak kursi roda. Secara umum penelitian kali ini akan menggunakan desain sistem sesuai dengan Gambar 3.1.



Gambar 3.1: Blok Diagram Penelitian

3.1.1 Paket Data

Untuk dapat menggerakkan kursi roda maka perlu mengirimkan perintah ke kontroler kursi roda. Pada tahap klasifikasi pose telah didapatkan perintah dasar untuk menggerakkan kursi roda, seperti maju, mundur, kanan, kiri, maupun stop. Perintah ini kemudian akan digabungkan dengan kecepatan maksimal menjadi satu *command* atau paket data seperti yang dilihat pada Persamaan 3.1.

Arah, *Kecepatan* (3.1)

Keterangan

Arah : Variabel dengan tipe data *char* yang digunakan untuk mengatur arah gerak motor kursi roda

Kecepatan : Variabel dengan tipe data *char* yang digunakan untuk mengatur kecepatan putar motor kursi roda

Variabel arah memiliki tipe data *char* yang akan menentukan gerak dari motor kursi roda, serta variabel kecepatan memiliki tipe data *char* yang akan menentukan kecepatan maksimal dari kursi roda. Untuk memperkecil ukuran data maka kode instruksi untuk menentukan arah gerak dan kecepatan maksimal akan diwakili oleh satu huruf. Kode instruksi dapat dilihat pada Tabel 3.1 dan Tabel 3.2.

Tabel 3.1: Kode instruksi dari hasil klasifikasi

Klasifikasi Pose	Kode Instruksi
Kiri	A
Maju	B
Stop	C
Mundur	D
Kanan	E

Tabel 3.2: Kode instruksi untuk mengatur tingkat PWM

Kecepatan Maksimal	Kode Instruksi
0	O
31	P
63	Q
95	R
127	S
159	T
191	U
223	V
255	W

Setelah kedua variabel tersebut digabungkan maka akan dikirim secara nirkabel, baik menggunakan Bluetooth maupun WiFi dari laptop atau Jetson Nano ke ESP32.

3.1.2 Memecahkan Paket Data

Paket data yang telah dikirimkan melalui laptop maupun Jetson Nano akan diterima oleh ESP32 menggunakan Bluetooth maupun WiFi. Saat diterima oleh ESP32, data tersebut akan menjalani serangkaian proses yang melibatkan pemecahan paket data dan penyesuaian sesuai dengan variabel yang telah ditentukan sebelumnya. Pemecahan paket data ini memungkinkan ESP32 untuk mendekomposisi informasi yang terkandung dalam setiap paket dan memastikan bahwa setiap variabel terpisah dengan akurat. Dengan demikian proses ini akan mengorganisir

dan menyusun kembali informasi serta memastikan bahwa setiap variabel telah benar sesuai dengan nama variabel dan tipe data yang disediakan.

3.1.3 Kontrol Navigasi

Kedua variabel yang didapatkan dari pemecahan paket data akan diproses pada ESP32. Variabel arah akan berperan untuk menentukan arah gerak dari motor, sedangkan variabel kecepatan akan digunakan untuk menetapkan kecepatan maksimal dari pergerakan motor tersebut. Terdapat serangkaian logika *if* berantai pada kontrol navigasi, dimana empat variabel dir akan menentukan arah putaran motor. Selain itu, nilai PWM maksimal dikonfigurasi dengan menggunakan variabel kecepatan sehingga pengguna dapat menyesuaikan kecepatan maksimal motor yang pengguna inginkan. Dengan demikian pada tahap ini ESP32 dapat secara efektif memproses data yang diterima melalui sistem nirkabel dan menghasilkan instruksi kontrol yang sesuai untuk menggerakkan kursi roda dengan arah dan kecepatan yang diinginkan.

3.2 Implementasi Alat

Pada penelitian ini dikembangkan suatu alat kontrol yang dapat menerima perintah melalui perangkat lain seperti laptop maupun Jetson Nano secara nirkabel. Pada sub bab ini akan dijabarkan implementasi dari alat yang dikembangkan pada penelitian ini.

3.2.1 *Hardware* dan *Software* yang digunakan

Berikut ini dijabarkan beberapa *Hardware* dan juga *Software* yang digunakan pada penelitian ini seperti berikut:

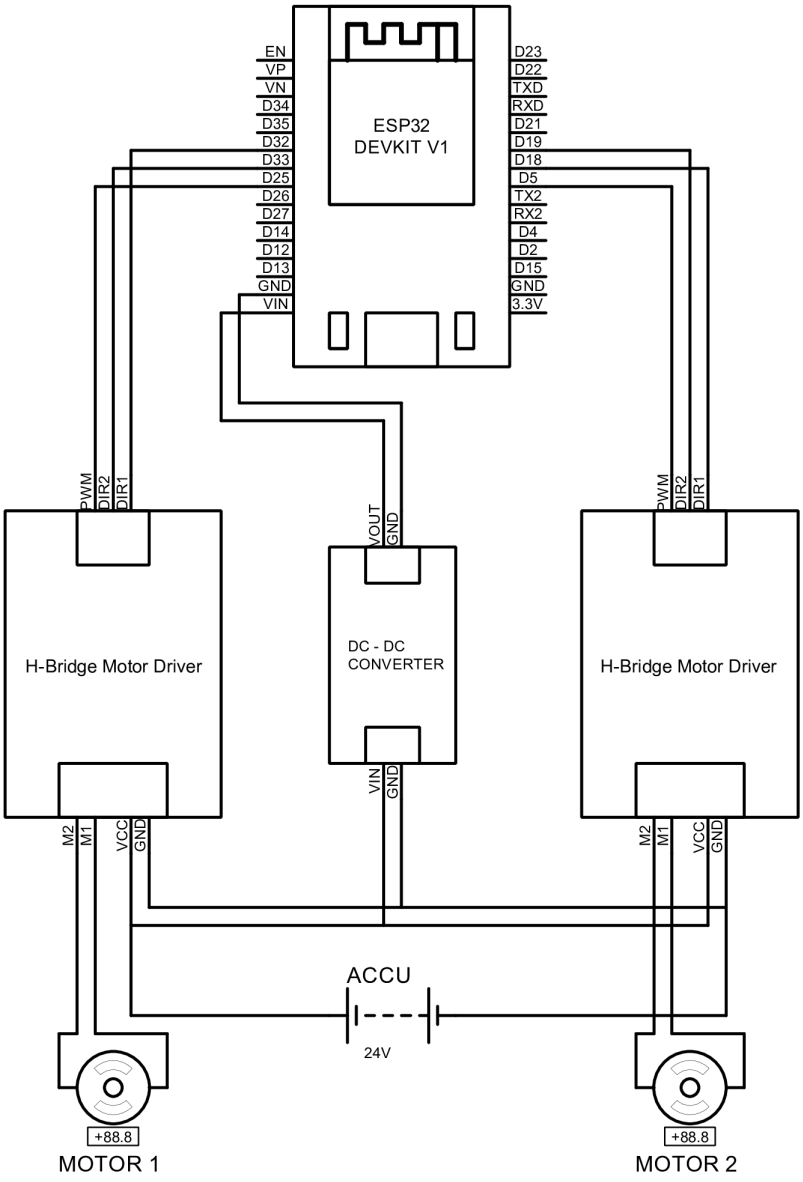
1. Anaconda Navigator
2. Arduino IDE
3. Laptop
4. Jetson Nano
5. Kamera
6. ESP32 Devkit V1
7. 2 Kontroller Motor
8. 2 DC-DC Voltage Regulator
9. 2 DC Motor
10. Baterai 24V

3.2.2 Skematik Alat

Skematik pada alat ini diilustrasikan secara rinci pada Gambar 3.2. Sistem ini menggunakan kamera yang dihubungkan dengan Laptop atau Jetson Nano sebagai perangkat utama dalam menangkap citra. Proses kerja dimulai saat kamera menangkap citra objek. Citra yang telah ditangkap ini lantas diproses oleh Laptop atau Jetson Nano. Di dalam sistem ini, model klasifikasi yang telah diprogram sebelumnya memainkan peran penting dalam menginterpretasikan data citra tersebut. Hasil dari proses klasifikasi ini sangat krusial karena menjadi dasar dalam penentuan kode instruksi yang akan diimplementasikan.

Kode instruksi tersebut kemudian akan dikombinasikan dengan parameter kecepatam maksimal yang sebelumnya telah ditetapkan oleh pemngguna. Gabungan dari kode instruksi dan parameter kecepatan ini akan menjadi satu paket data sebagai kontrol gerak kursi roda. Paket data ini kemudian akan ditransmisikan secara nirkabel, baik dengan Bluetooth maupun WiFi,

ke modul ESP32 Devkit V1.



Gambar 3.2: Skematik kontrol motor kursi roda

ESP32 memiliki peranan penting dalam kontrol motor kursi roda. ESP32 akan digunakan sebagai pusat pengendalian yang menerima paket data yang telah dikirimkan oleh pengguna secara nirkabel. Kemudian ESP32 akan melakukan pemecahan paket data dan menyesuaikan data tersebut kedalam variabel-variabel yang telah ditentukan. Proses pemecahan paket data ini akan menghasilkan dua data utama yang kemudian akan diproses lebih lanjut oleh ESP32.

Variabel pertama merupakan variabel arah yang memiliki fungsi krusial untuk menentukan arah gerak kedua motor pada kursi roda. Variabel ini akan memastikan bahwa motor bergerak sesuai dengan arah yang diinginkan sesuai dengan data yang diterima. Selain itu terdapat variabel kecepatan yang digunakan untuk menetapkan kecepatan maksimal pergerakan motor.

Dalam implementasinya terdapat serangkaian logika *if* berantai yang akan dijelaskan secara terperinci pada sub bab program. Sederhananya logika *if* berantai ini memiliki peran yang

penting dalam pengambilan keputusan baik arah dan kecepatan motor sesuai dengan data yang telah diterima. Hasil dari logika ini akan memberikan *trigger* berupa tegangan 5V ataupun 0V. Tegangan ini kemudian akan mempengaruhi arah putar motor.

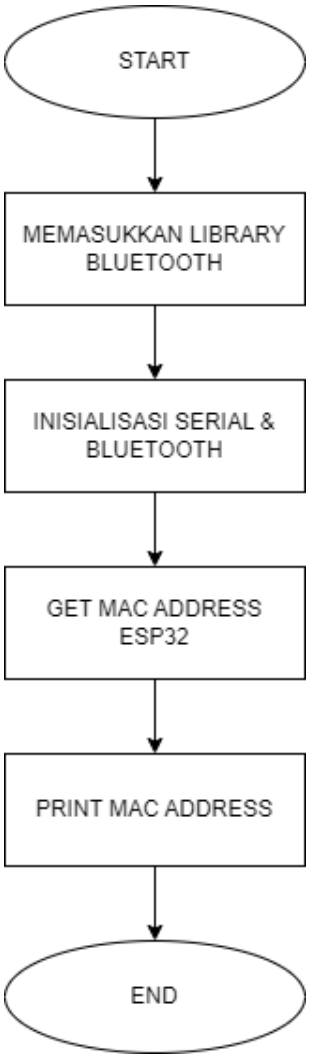
Selanjutnya variabel kecepatan akan digunakan untuk mengatur tingkat *Pulse Width Modulation* pada kontroler motor. Pengaturan PWM ini sangatlah penting guna mengatur kecepatan putar motor. Dengan mengatur tingkat PWM maka kecepatan motor maksimal dapat disesuaikan sesuai dengan kebutuhan.

3.3 Kode Program

Pada sub bab ini akan dijabarkan kode program yang digunakan pada penelitian ini.

3.3.1 Program Untuk Memeriksa MAC Address Pada ESP32

Agar dapat terhubung dengan Bluetooth pada ESP32 maka kita harus mengetahui MAC Address dari ESP32. Berikut merupakan program untuk memeriksa MAC Address Pada ESP32 dapat dilihat pada Program 5.1 beserta *flowchart* sesuai Gambar 3.3.



Gambar 3.3: *Flowchart* Memeriksa MAC Address Pada ESP32

Program ini menggunakan *library* BluetoothSerial dari Henry Abrahamsen. *Library* ini menyediakan fungsionalitas untuk mengontrol modul Bluetooth pada ESP32 (Abrahamsen, 2023). Selanjutnya akan dibuat objek SerialBT dari kelas BluetoothSerial. Objek ini digunakan untuk berkomunikasi melalui Bluetooth.

Pada void setup() terdapat Serial.begin() yang digunakan untuk menginisialisasi komunikasi serial dengan kecepatan 115200 bps. Selanjutnya modul Bluetooth diinisialisasikan dengan nama perangkat ESP32_Haris. Berikan delay selama 1 detik agar Bluetooth dapat diinisialisasikan dengan baik.

Array esp32Address dideklarasi pada awal fungsi void loop(). Array ini akan digunakan untuk menyimpan alamat dari Bluetooth ESP32. fungsi esp_efuse_mac_get_default() digunakan untuk mendapatkan alamat Bluetooth dan menyimpannya dalam array esp32Address (Systems, 2023). Alamat Bluetooth kemudian akan dicetak ke Serial Monitor menjadi bentuk yang dapat dibaca dengan menggunakan kode Serial.printf().

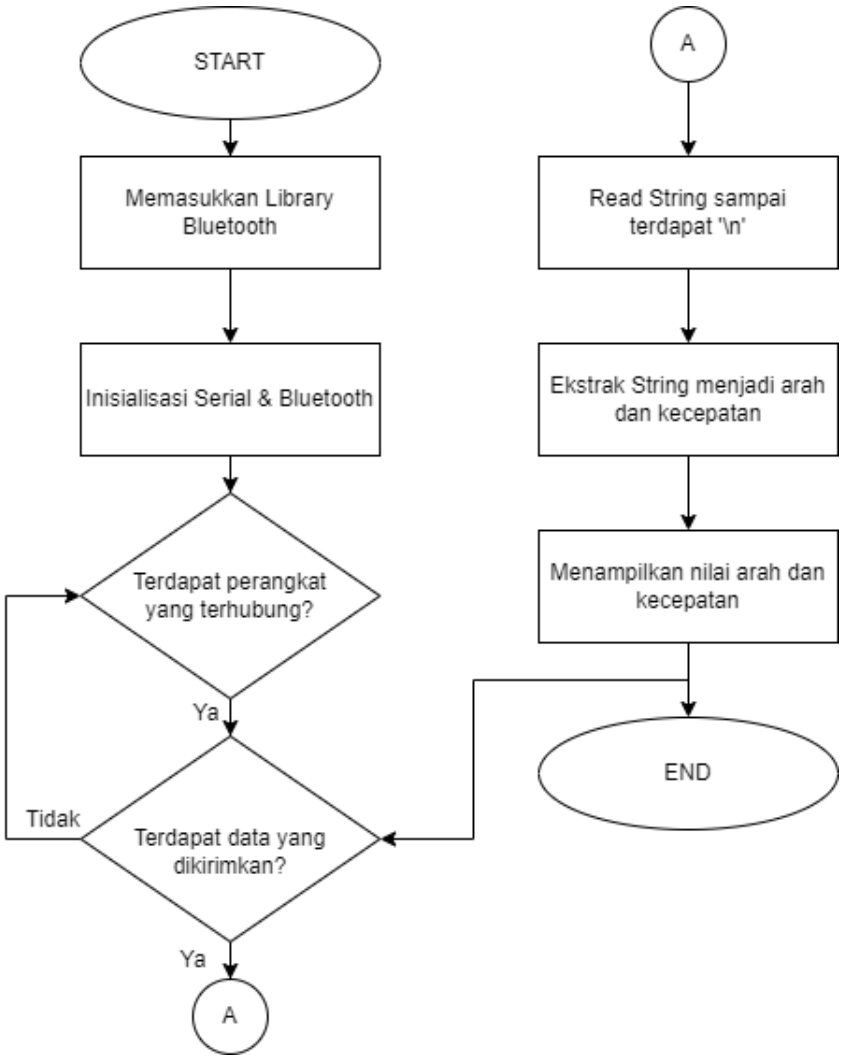
3.3.2 Program Untuk Menerima Data String Melalui Bluetooth Pada ESP32

Program ini dirancang untuk menerima data string melalui Bluetooth dan memisahkannya menjadi 2 bagian. Kemudian terdapat data yang dievaluasi untuk mengetahui apakah data tersebut sudah berubah menjadi integer. Berikut ini merupakan Program 5.2 yang digunakan untuk menguji kemampuan ESP32 dalam menerima data string melalui Bluetooth beserta *flowchart* sesuai Gambar 3.4.

Program ini menggunakan *library* BluetoothSerial dari Henry Abrahamsen. *Library* ini menyediakan fungsionalitas untuk menerima data melalui koneksi Bluetooth dan mengolahnya (Abrahamsen, 2023). Selanjutnya akan dibuat objek SerialBT dari kelas BluetoothSerial. Objek ini selanjutnya digunakan untuk berkomunikasi melalui Bluetooth. Terdapat juga variabel maxspeed yang dideklarasikan sebagai suatu variabel yang menyimpan nilai kecepatan maksimum bertipe data integer.

Pada void setup() terdapat Serial.begin() yang digunakan untuk menginisialisasi komunikasi serial dengan kecepatan 115200 bps. Selanjutnya modul Bluetooth diinisialisasikan dengan nama perangkat ESP32_Haris.

Pada fungsi void loop() akan berjalan terus-menerus setelah fungsi void setup(). Pertamatama akan diperiksa apakah terdapat data yang tersedia untuk dibaca dari koneksi Bluetooth dengan menggunakan SerialBT.available(). Apabila terdapat data yang diterima maka data tersebut akan dimasukkan kedalam variabel receivedData yang bertipe string dengan menggunakan fungsi SerialBT.readStringUntil(). Data tersebut akan dibaca hingga menemukan karakter *newline* ('\n'). Data kemudian akan dipisahkan menjadi arah dan kecepatan. arah akan memisahkan data sebelum koma (',') dari receivedData. kecepatan akan memisahkan data setelah (',') hingga akhir receivedData. Tipe data dari variabel kecepatan ini kemudian akan diubah menjadi integer dan nilai tersebut dimasukkan ke dalam variabel maxspeed. Kemudian nilai pada variabel arah dan kecepatan akan ditampilkan ke *Serial Monitor*. fungsi *if* ditambahkan untuk menguji apakah variabel maxspeed sudah bertipe integer. Apabila nilai maxspeed kurang dari 20 maka akan ditampilkan *boolean True*, apabila lebih dari atau sama dengan 20 maka akan menampilkan *boolean False*.



Gambar 3.4: *Flowchart* Menerima Data String Melalui Bluetooth Pada ESP32

3.3.3 Program Untuk Menerima Data JSON Melalui Bluetooth Pada ESP32

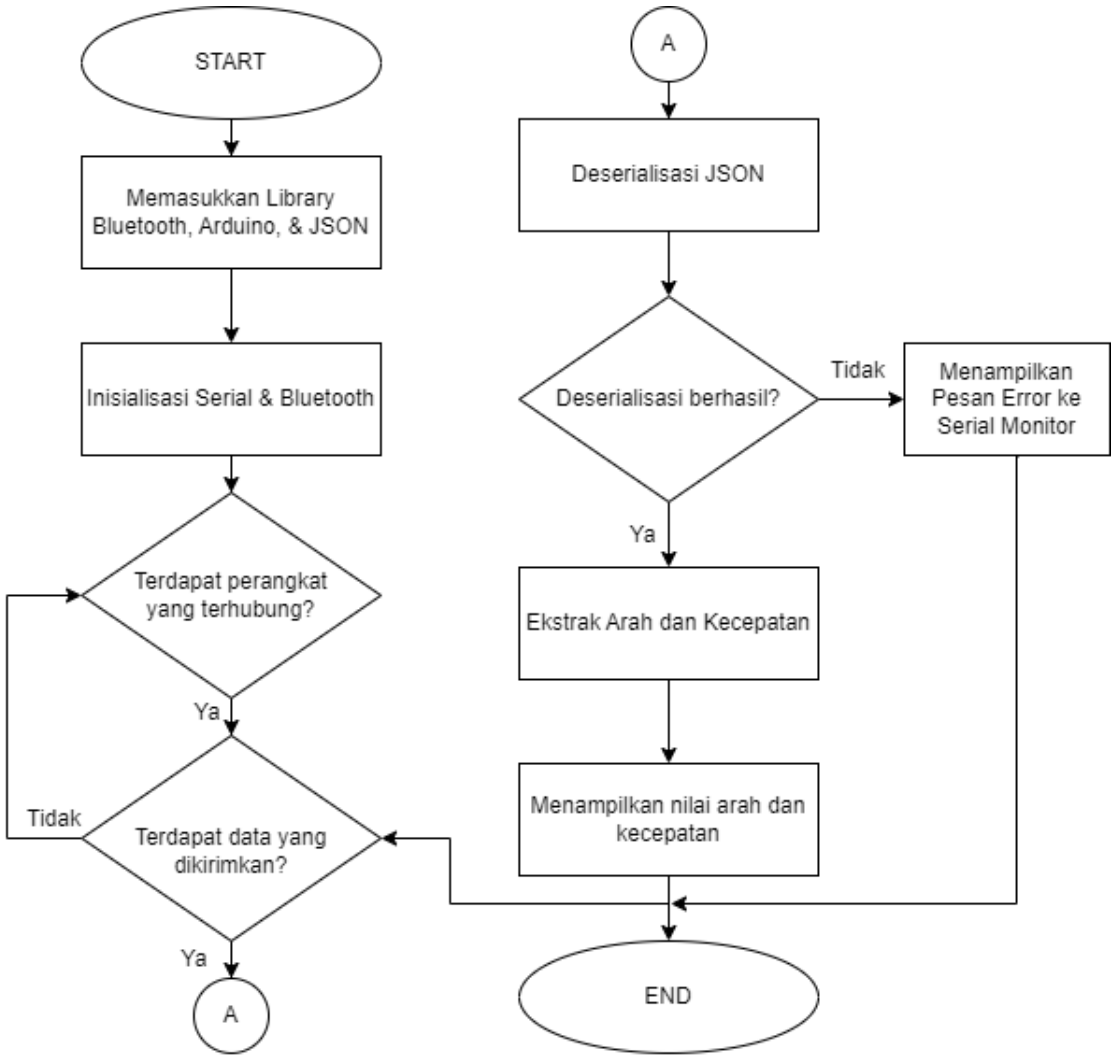
Program ini dirancang untuk menerima data JSON melalui Bluetooth dan memisahkannya menjadi 2 bagian. Berikut merupakan program untuk menguji kemampuan ESP32 dalam menerima data JSON melalui Bluetooth yang dapat dilihat pada Program 5.3 beserta *flowchart* sesuai Gambar 3.5.

Program ini menggunakan *library* BluetoothSerial dari Henry Abrahamsen dan ArduinoJson dari Benoît Blanchon. *Library* BluetoothSerial menyediakan fungsionalitas untuk menerima data melalui koneksi Bluetooth (Abrahamsen, 2023). Sedangkan *library* ArduinoJson digunakan untuk menguraikan *deserialize* data JSON yang diterima (Blanchon, 2021). Selanjutnya akan dibuat objek SerialBT dari kelas BluetoothSerial. Objek ini akan digunakan untuk berkomunikasi melalui Bluetooth.

Pada void setup() terdapat Serial.begin() untuk menginisialisasi komunikasi serial dengan kecepatan 115200 bps. Selanjutnya modul Bluetooth diinisialisasi dengan nama perangkat ESP_Haris.

Pada fungsi void loop() akan berjalan terus-menerus setelah fungsi void setup(). Pertama-

tama akan diperiksa apakah terdapat data yang tersedia untuk dibaca dari koneksi Bluetooth dengan menggunakan SerialBT.available(). Kemudian data akan dibaca dari koneksi Bluetooth hingga terdapat karakter *newline* ('\n') dan menyimpannya dalam string json_data. Objek DynamicJsonDocument selanjutnya akan dibuat dengan kapasitas 1024 byte yang berguna untuk menampung dokumen JSON yang akan diuraikan. Data JSON yang telah ditampung kemudian hasilnya disimpan dalam objek doc. Jika terdapat kesalahan dalam proses deserialisasi maka pesan kesalahan akan ditampilkan. Lalu terdapat fungsi *if* yang digunakan untuk memberikan pesan *"Failed to parse JSON"* apabila terdapat kesalahan pada proses deserialisasi. Apabila tidak terdapat kesalahan maka setiap nilai dari atribut JSON akan diekstrak dengan nama arah dan kecepatan serta menyimpannya dalam variabel bertipe string. Nilai yang diterima dari atribut JSON akan dicetak ke *Serial Monitor*.

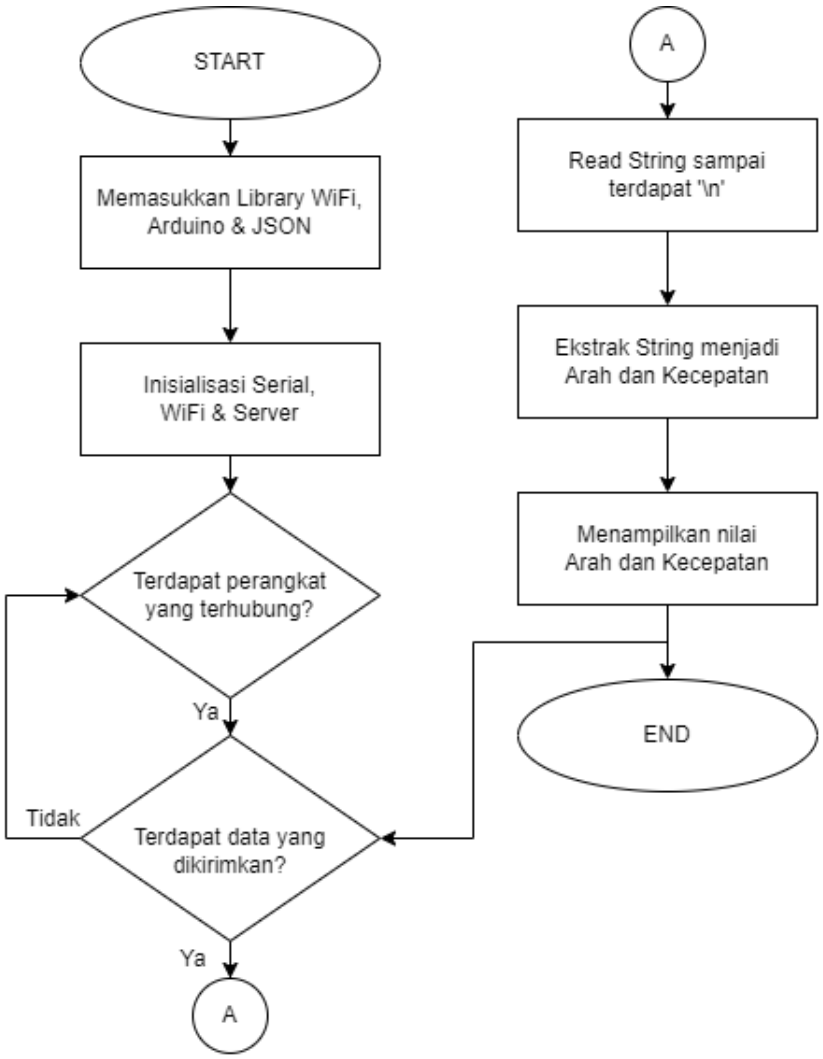


Gambar 3.5: *Flowchart* Menerima Data JSON Melalui Bluetooth Pada ESP32

3.3.4 Program Untuk Menerima Data String Melalui *Access Point* WiFi Pada ESP32

Program ini dirancang sebagai server WiFi yang dapat menerima data dari *client* yang terhubung dan mengolahnya. Data yang diterima akan diproses dan dipisahkan menjadi 2 variabel, yaitu variabel arah dan kecepatan. Kemudian nilai dari variabel kecepatan akan dimasukkan ke

variabel maxspeed dan tipe datanya dievaluasi. Berikut merupakan Program 5.4 yang digunakan untuk menerima data string melalui *Access Point* WiFi pada ESP32 beserta *flowchart* sesuai Gambar 3.6.



Gambar 3.6: *Flowchart* Menerima Data String Melalui *Access Point* WiFi Pada ESP32

Program ini menggunakan *library* WiFi yang menyediakan fungsionalitas untuk mengonfigurasi dan mengelola koneksi WiFi pada ESP32. Selain itu *library* Arduino juga digunakan pada program ini yang menyediakan fungsi-fungsi dasar untuk pemrograman mikrokontroler. Variabel maxspeed dideklarasikan sebagai variabel yang menyimpan nilai integer yang kemudian digunakan untuk menyimpan nilai kecepatan maksimum. Variabel ssid akan digunakan untuk menyimpan nama jaringan WiFi yang akan dibuat dan variabel *password* akan menyimpan nilai *password* dari WiFi Server yang dibuat. Kedua variabel ini sangat penting agar ESP32 dapat dikonfigurasikan sebagai *Access Point*. Selanjutnya objek server dibuat dari kelas WiFiServer untuk menangani koneksi pada port 80.

Pada void setup() terdapat Serial.begin() untuk menginisialisasi komunikasi serial dengan kecepatan 115200 bps. Lalu mengonfigurasi ESP32 sebagai *Access Point* sesuai dengan SSID dan *password* yang telah ditentukan. Apabila *Access Point* telah dikonfigurasi maka IP *Address* dari ESP32 akan disimpan pada variabel IP dan dicetak ke *Serial Monitor*. server.begin()

digunakan untuk memulai server pada port 80.

Fungsi void loop() akan berjalan terus-menerus setelah fungsi void setup() dijalankan. Pada fungsi ini terdapat WiFiClient yang digunakan untuk menguji apakah terdapat *client* yang terhubung ke server. Jika terdapat *client* yang terhubung maka akan masuk ke fungsi *if*. Data yang dikirimkan akan dibaca secara terus-menerus melalui fungsi *while*. Data yang diterima akan dimasukkan ke variabel receivedData hingga *client* mengirimkan karakter *newline* ('\n'). Data kemudian akan dipisahkan menjadi arah dan kecepatan. arah akan memisahkan data sebelum koma (',') dari receivedData. kecepatan akan memisahkan data setelah (',') hingga akhir receivedData. Tipe data dari variabel kecepatan ini kemudian akan diubah menjadi integer dan nilai tersebut dimasukkan ke dalam variabel maxspeed. Kemudian nilai pada variabel arah dan kecepatan akan ditampilkan ke *Serial Monitor*. fungsi *if* ditambahkan untuk menguji apakah variabel maxspeed sudah bertipe integer. Apabila nilai maxspeed kurang dari 20 maka akan ditampilkan *boolean True*, apabila lebih dari atau sama dengan 20 maka akan menampilkan *boolean False*. Lalu koneksi akan ditutup setelah selesai membaca data dari *client*.

3.3.5 Program Untuk Menerima Data JSON Melalui *Access Point* WiFi Pada ESP32

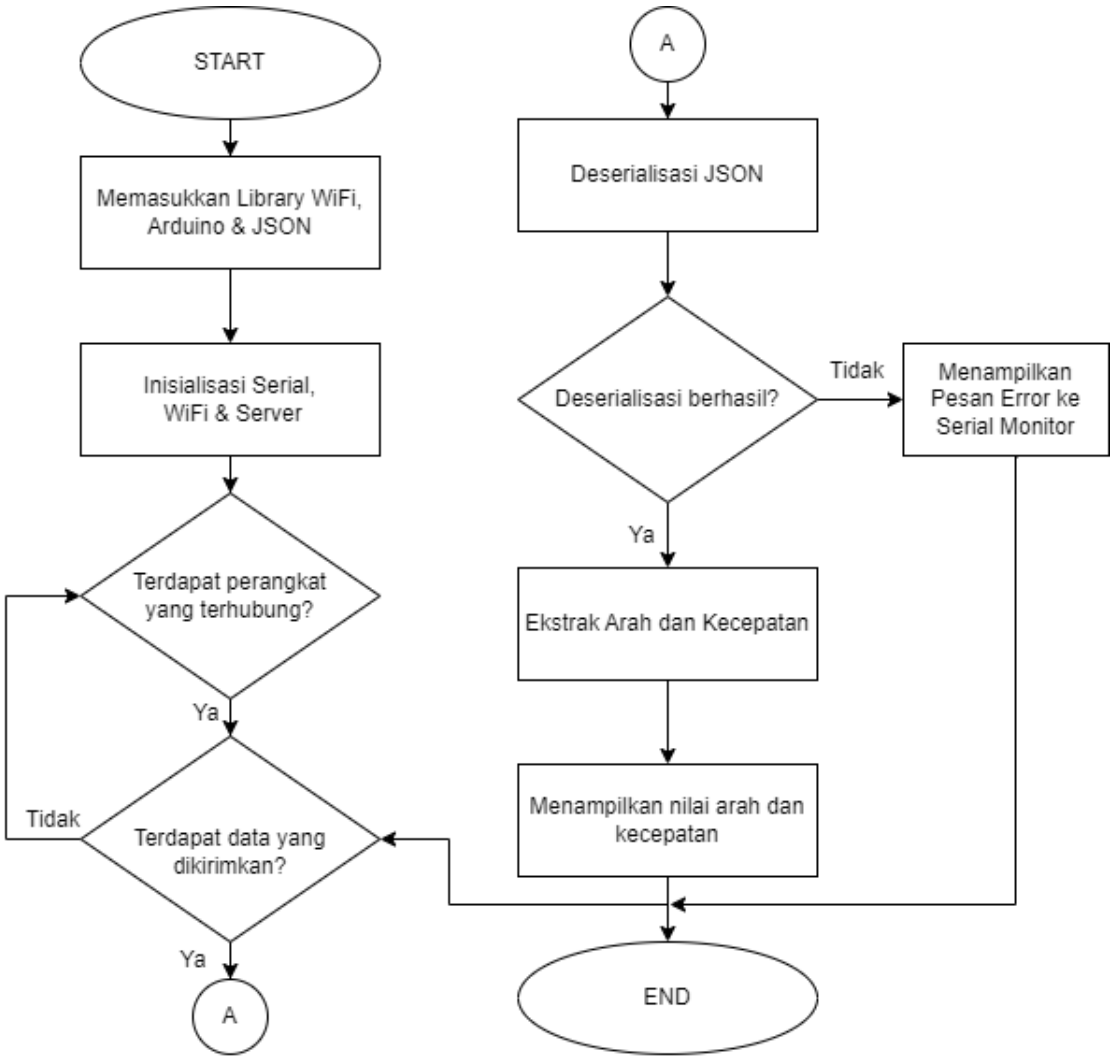
Program ini dirancang untuk menerima data JSON melalui WiFi. ESP32 berperan sebagai *Access Point* WiFi yang dapat menerima data JSON dari *client* dan mencetaknya ke dalam *Serial Monitor*. Berikut merupakan Program 5.5 yang digunakan untuk menerima data JSON melalui *Access Point* WiFi pada ESP32 beserta *flowchart* sesuai Gambar 3.7.

Program ini menggunakan *library* WiFi yang menyediakan fungsionalitas untuk mengkonfigurasi dan mengelola koneksi WiFi pada ESP32. Selain itu *library* Arduino juga digunakan pada program ini yang menyediakan fungsi-fungsi dasar untuk pemrograman mikrokontroler. *Library* ArduinoJson dari Benoît Blanchon digunakan untuk mengolah data JSON. Variabel maxspeed yang dideklarasikan sebagai suatu variabel yang menyimpan nilai kecepatan maksimum yang diterima dari data JSON. Variabel ssid dan password digunakan untuk menyimpan nilai nama dan *password* untuk *access point* yang akan dibuat oleh ESP32. Lalu objek server dari kelas WiFiServer dibuat untuk menangani koneksi pada port 80.

Pada void setup() terdapat Serial.begin() yang digunakan untuk menginisialisasi komunikasi serial dengan kecepatan 115200 bps. Selanjutnya *access point* akan dibuat sesuai dengan ssid dan password yang telah ditentukan sebelumnya. IP *Address* kemudian akan ditampilkan pada *Serial Monitor*. Akhirnya server akan dimulai pada port 80.

Fungsi void loop() akan berjalan terus-menerus setelah fungsi void setup() dijalankan. ESP32 akan mencoba untuk menerima koneksi dari *client* melalui fungsi server.available(). Jika terdapat *client* yang terhubung maka ESP32 akan memeriksa apakah *client* masih tetap terhubung. Jika terdapat data yang tersedia maka data akan dibaca hingga terdapat karakter *newline*('\n'). Selanjutnya objek doc akan dibuat dari kelas DynamicJsonDocument dengan kapasitas 1024 *byte*. Data yang didapatkan kemudian akan dipecah dengan menggunakan fungsi deserializeJson.

Jika terjadi kesalahan saat mengurai JSON maka program akan mencetak pesan *error*. Jika *parsing* JSON berhasil dilakukan, maka nilai dari JSON akan diakses. Data JSON akan diurai dan disimpan kedalam variabel arah serta kecepatan. Selanjutnya program akan menampilkan variabel tersebut ke *Serial Monitor*. Setelah data JSON selesai diproses maka koneksi dengan *client* akan ditutup.



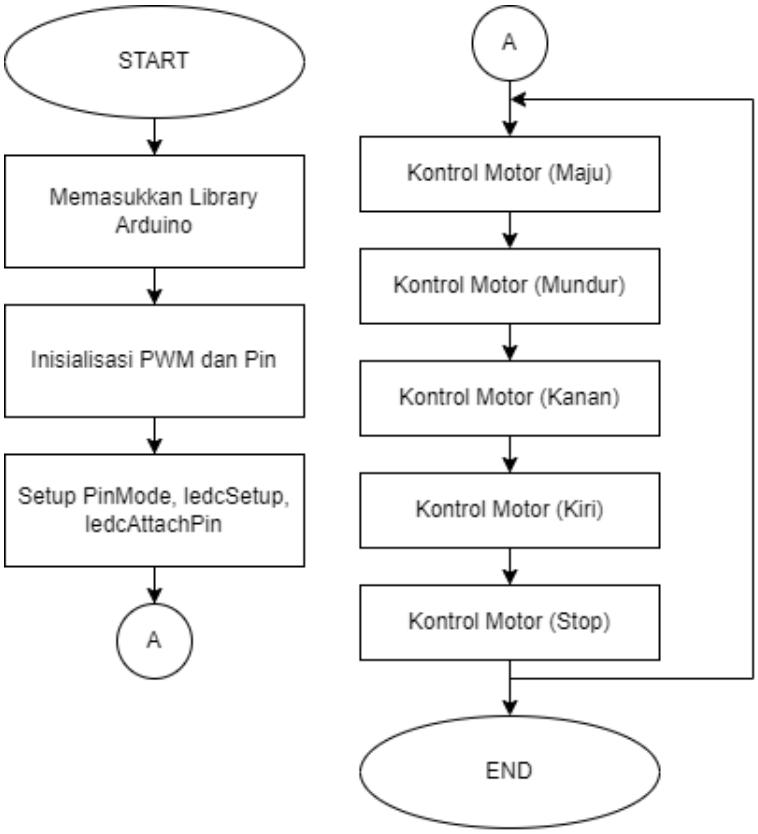
Gambar 3.7: *Flowchart* Menerima Data JSON Melalui *Access Point* WiFi Pada ESP32

3.3.6 Program Untuk Menguji Rangkaian Kontrol ESP32

Program ini dirancang untuk menguji rangkaian kontrol ESP32. Kedua motor akan diuji dan dikendalikan melalui modul ESP32 dan driver motor H-Bridge. Berikut program 5.6 yang digunakan untuk menguji rangkaian kontrol ESP32 beserta *flowchart* sesuai Gambar 3.8.

Library Arduino digunakan pada program ini. *Library* ini menyediakan berbagai fungsi-fungsi dasar untuk pemrograman mikrokontroler. pwmPin1, dir1, dan dir2 merupakan variabel untuk mengatur kerja dari motor kiri. pwmPin1 didefinisikan dengan nilai 5 yang kemudian akan terhubung dengan GPIO5 pada ESP32, dir1 didefinisikan dengan nilai 18 yang kemudian akan terhubung dengan GPIO18 pada ESP32, dan dir2 yang didefinisikan dengan nilai 19 yang kemudian akan terhubung dengan GPIO19 pada ESP32. Selanjutnya terdapat pwmPin2, dir3, dan dir4 yang merupakan variabel yang digunakan untuk mengatur kerja dari motor kanan. pwmPin2 didefinisikan dengan nilai 25 yang kemudian akan terhubung dengan GPIO25 pada ESP32, dir3 didefinisikan dengan nilai 32 yang kemudian akan terhubung dengan GPIO32 pada ESP32, dan dir4 yang didefinisikan dengan nilai 33 yang kemudian akan terhubung dengan GPIO33 pada ESP32. Lalu variabel pwmChannel1, pwmChannel2, freq, dan res dideklarasikan. pwmChannel1 dan pwmChannel2 merupakan nomor saluran PWM yang

digunakan. Variabel freq merupakan frekuensi PWM yang diatur menjadi 15kHz untuk meminimalisir *noise* pada motor ketika bekerja. Variabel res digunakan untuk mengatur resolusi dari PWM yang diatur menjadi 8-bit. Variabel PWM1_DutyCycle akan menyimpan siklus tugas yang kemudian akan digunakan untuk mengendalikan kecepatan motor dan variabel maxspeed merupakan nilai dari kecepatan motor maksimal.



Gambar 3.8: *Flowchart* Menguji Rangkaian Kontrol ESP32

Terdapat fungsi void setup() yang dieksekusi satu kali pada saat program pertama kali dijalankan. Pada fungsi ini terdapat dir1, dir2, dir3, dan dir4 yang diatur sebagai pin *output*. ledcSetup merupakan fungsi dari *library* ledc yang digunakan untuk mengonfigurasi saluran PWM. Fungsi ini memerlukan 3 variabel, yaitu pwmChannel, freq, dan res. Selanjutnya terdapat fungsi ledcAttachPin yang merupakan fungsi dari *library* ledc yang digunakan untuk menghubungkan suatu pin dengan saluran PWM tertentu. Fungsi ini memerlukan 2 variabel, yaitu pin PWM dan juga PWM Channel. Secara keseluruhan, blok kode ini digunakan untuk melakukan konfigurasi pin-pin pada mikrokontroler ESP32 sehingga motor dapat dikendalikan. Pin-pin yang diatur sebagai *output* akan digunakan untuk mengatur arah putar motor. Sedangkan saluran PWM akan digunakan untuk mengatur kecepatan motor.

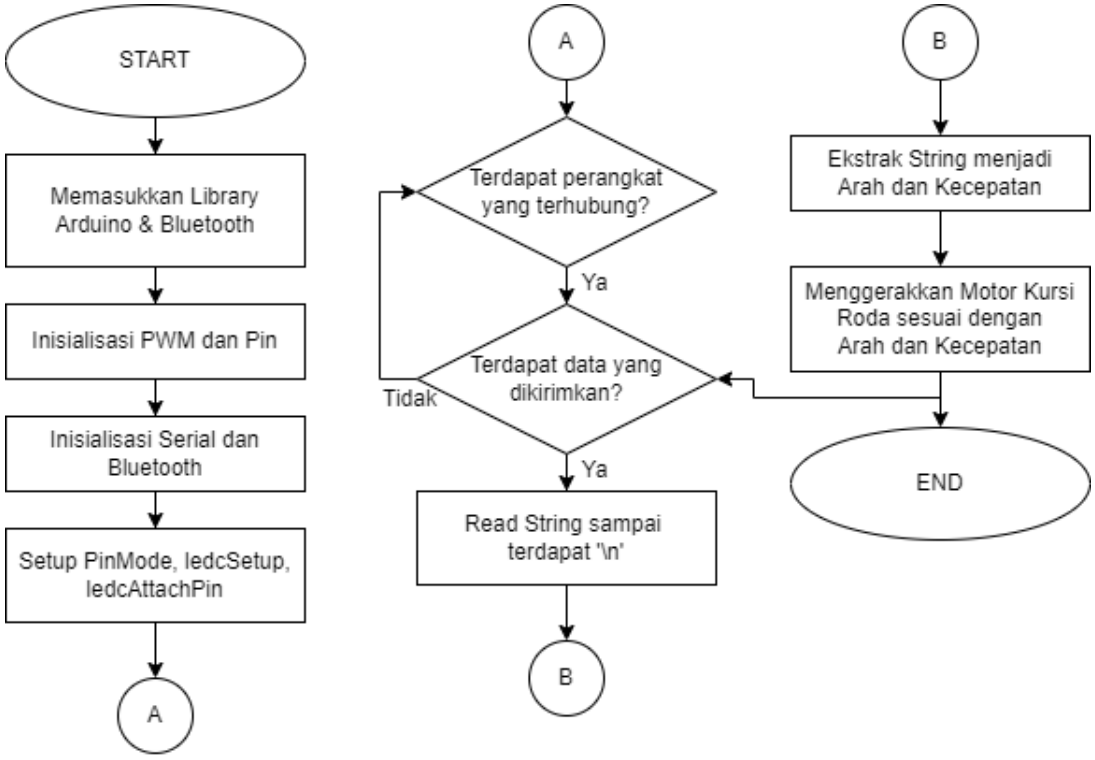
Pada fungsi void loop() terdapat serangkaian perintah yang mengatur gerakan kursi roda dengan menggunakan motor DC. Gerakan ini akan diulang secara terus-menerus. Hal ini berguna untuk menguji rangkaian kontrol pada ESP32. Setiap gerakan diawali dengan *softstart* dan diakhiri dengan *softstop*. Pada gerakan maju, kedua motor akan bergerak maju. Selama nilai PWM1_DutyCycle kurang dari maxspeed, maka nilai PWM1_DutyCycle akan ditambahkan secara bertahap hingga mencapai nilai maxspeed yang telah ditentukan. Selama itu juga kecepatan putar motor akan bertambah secara bertahap. Penambahan nilai PWM1_DutyCycle akan di-

tunda selama 10 milidetik. Setelah PWM1_DutyCycle mencapai nilai maxspeed maka kode selanjutnya akan ditunda selama 5 detik yang mengakibatkan motor akan berputar maju dengan kecepatan maksimum selama 5 detik. Setelah 5 detik, apabila nilai dari PWM1_DutyCycle lebih dari 0 maka akan dilakukan *softstop*. Arah putar kedua motor tetap diatur maju. Nilai PWM1_DutyCycle akan diturunkan secara bertahap. Pengurangan nilai PWM1_DutyCycle akan ditunda selama 10 milidetik. Apabila nilai PWM1_DutyCycle sudah mencapai nilai 0 maka kode selanjutnya akan ditunda selama 1 detik. Langkah yang serupa akan dilakukan untuk setiap gerakan. Setiap gerakan memiliki fase *softstart* dan *softstop* yang bertujuan untuk memberikan perubahan kecepatan yang halus dan menghindari perubahan secara tiba-tiba. Penggunaan variabel PWM1_DutyCycle digunakan untuk mengatur siklus kerja PWM untuk sebagai kontrol kecepatan yang fleksibel. fungsi *delay* digunakan untuk memberikan jeda waktu antara setiap gerakan untuk mengatur durasi gerakan dan memberikan waktu bagi motor kursi roda untuk menyelesaikan setiap gerakan sebelum beralih ke gerakan berikutnya. Pada fungsi mundur, kedua roda akan berputar mundur. Pada fungsi belok kanan, roda kiri akan berputar maju sedangkan roda kanan tidak berputar sehingga kursi roda dapat berbelok ke arah kanan. Pada fungsi belok kiri, motor kiri tidak berputar namun roda kanan akan berputar maju sehingga kursi roda dapat berbelok ke arah kiri. Pada fungsi stop, kedua roda tidak akan berputar. Proses ini terus berulang didalam *loop*, sehingga kursi roda akan melakukan gerakan berulang sesuai dengan logika yang telah diatur.

3.3.7 Program Kontrol Motor Kursi Roda Melalui Bluetooth

Program ini dirancang untuk mengendalikan motor DC dengan memproses perintah yang diterima melalui koneksi Bluetooth. Setelah menerima data arah dan kecepatan, maka program ini akan mengatur motor sesuai dengan perintah yang diterima. Berikut merupakan Program 5.7 yang digunakan untuk mengontrol motor kursi roda melalui Bluetooth beserta *flowchart* sesuai Gambar 3.9.

Program ini menggunakan *library* BluetoothSerial dari Henry Abrahamsen dan juga *library* Arduino. *Library* BluetoothSerial menyediakan fungsionalitas untuk menerima data melalui koneksi Bluetooth (Abrahamsen, 2023). *Library* Arduino menyediakan berbagai fungsi dasar untuk pemrograman mikrokontroler. pwmPin1, dir1, dan dir2 merupakan variabel yang digunakan untuk mengatur kerja dari motor kiri. pwmPin1 didefinisikan dengan nilai 5 yang kemudian akan terhubung dengan GPIO5 pada ESP32, dir1 didefinisikan dengan nilai 18 yang kemudian akan terhubung dengan GPIO18 pada ESP32, dan dir2 didefinisikan dengan nilai 19 yang kemudian akan terhubung dengan GPIO19 pada ESP32. Selanjutnya terdapat pwmPin2, dir3, dan dir4 merupakan variabel yang akan digunakan untuk mengatur kerja dari motor kanan. pwmPin2 didefinisikan dengan nilai 25 yang kemudian akan terhubung dengan GPIO25 pada ESP32, dir3 didefinisikan dengan nilai 32 yang kemudian akan terhubung dengan GPIO32, dan dir4 yang didefinisikan dengan nilai 33 yang kemudian akan terhubung dengan GPIO33 pada ESP32. Array stdir digunakan untuk menyimpan keadaan arah motor pada setiap langkah. Lalu variabel pwmChannel1, pwmChannel2. req, dan res dideklarasikan. pwmChannel1 dan pwmChannel2 merupakan nomor saluran PWM yang digunakan. Variabel freq merupakan frekuensi PWM yang diatur menjadi 15 kHz untuk meminimalisir *noise* pada motor ketika bekerja. Variabel res digunakan untuk mengatur resolusi dari PWM yang diatur menjadi 8-bit. Variabel PWM1_DutyCycle akan menyimpan siklus tugas yang kemudian akan digunakan untuk mengendalikan kecepatan motor. Variabel maxspeed digunakan untuk mengatur kecepatan maksimal dari motor dan variabel turnspeed digunakan untuk kecepatan maksimal ketika kursi roda berbelok.



Gambar 3.9: *Flowchart* Kontrol Motor Kursi Roda Melalui Bluetooth

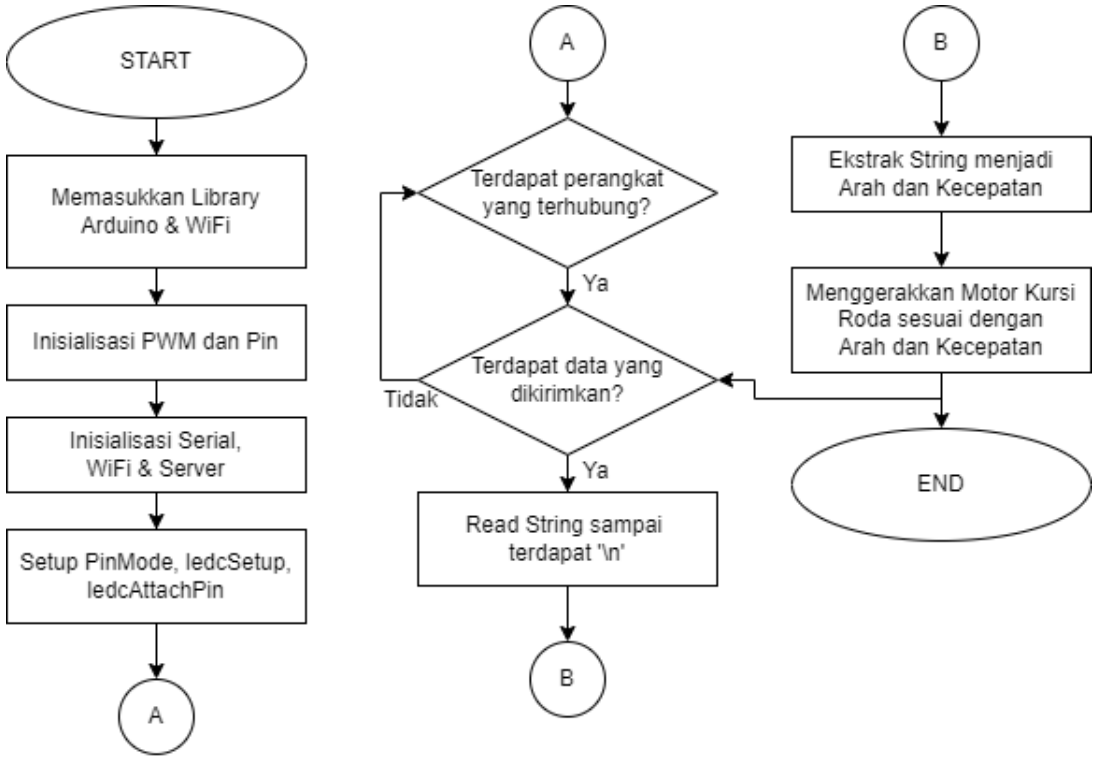
Fungsi void setup() dieksekusi satu kali pada saat program pertama kali dijalankan. Pada fungsi ini objek SerialBT diinisialisasi sebagai Bluetooth Serial dengan nama ESP32_Haris. dir1, dir2, dir3, dan dir4 diatur sebagai *output*. ledcSetup merupakan fungsi dari *library* ledc yang digunakan untuk menghubungkan suatu pin dengan saluran PWM tertentu. Fungsi ini memerlukan 2 variabel, yaitu pin PWM dan juga PWM *Channel*. Secara keseluruhan blok kode ini digunakan untuk melakukan konfigurasi pin-pin pada mikrokontroler ESP32 sehingga motor dapat dikendalikan. Pin-pin yang diatur sebagai *output* akan digunakan untuk mengatur arah putar motor. Sedangkan saluran PWM akan digunakan untuk mengatur kecepatan putar motor.

Pada fungsi void loop() terdapat perintah untuk memeriksa ketersediaan data dari Bluetooth dengan SerialBT.acailable(). Jika terdapat *client* yang terhubung maka data akan dibaca hingga *newline* ('\n'). Data yang diterima kemudian akan diurai dan dimasukkan kedalam variabel arah dan kecepatan. Tipe data dari variabel kecepatan akan diubah dari string menjadi integer dengan menggunakan fungsi toInt(). Nilai dari maxspeed dan turnspeed kemudian akan diatur sesuai dengan perintah. Kontrol arah gerak motor menggunakan pernyataan kondisional (*if-else*) berdasarkan variabel arah yang diterima. Pada setiap kondisi terdapat perulangan *while*. Hal ini dilakukan agar perubahan kecepatan putar motor tidak terjadi secara tiba-tiba. digitalWrite digunakan untuk menetapkan arah putar motor. ledcWrite digunakan untuk mengendalikan kecepatan motor. Perulangan ini terus dilakukan selama ESP32 masih menerima data melalui Bluetooth.

3.3.8 Program Kontrol Motor Kursi Roda Melalui *Access Point* WiFi

Program ini dirancang untuk mengendalikan motor DC dengan memproses perintah yang diterima melalui koneksi WiFi. ESP32 bertindak sebagai *Access Point*. Setelah menerima data

arah maka program ini akan mengatur motor sesuai dengan perintah yang diterima. Berikut merupakan Program 5.8 yang digunakan untuk mengontrol motor kursi roda melalui WiFi beserta *flowchart* sesuai Gambar 3.10.



Gambar 3.10: *Flowchart* Kontrol Motor Kursi Roda Melalui *Access Point* WiFi

Program ini menggunakan *library* WiFi yang menyediakan fungsionalitas untuk mengkonfigurasi dan mengelola koneksi WiFi pada ESP32. Selain itu *library* Arduino juga digunakan pada program ini yang menyediakan fungsi-fungsi dasar untuk pemrograman mikrokontroler. Variabel ssid dan password digunakan untuk menyimpan nilai nama dan *password* untuk *access point* yang akan dibuat oleh ESP32. Lalu objek server dari kelas WiFiServer dibuat untuk menangani koneksi pada port 80.

pwmPin1, dir1, dan dir2 merupakan variabel yang digunakan untuk mengatur kerja dari motor kiri. pwmPin1 didefinisikan dengan nilai 5 yang kemudian akan terhubung dengan GPIO5 pada ESP32, dir1 didefinisikan dengan nilai 18 yang kemudian akan terhubung dengan GPIO18 pada ESP32, dan dir2 didefinisikan dengan nilai 19 yang kemudian akan terhubung dengan GPIO19 pada ESP32. Selanjutnya terdapat pwmPin2, dir3, dan dir4 merupakan variabel yang akan digunakan untuk mengatur kerja dari motor kanan. pwmPin2 didefinisikan dengan nilai 25 yang kemudian akan terhubung dengan GPIO25 pada ESP32, dir3 didefinisikan dengan nilai 32 yang kemudian akan terhubung dengan GPIO32, dan dir4 yang didefinisikan dengan nilai 33 yang kemudian akan terhubung dengan GPIO33 pada ESP32. Array stdir digunakan untuk menyimpan keadaan arah motor pada setiap langkah. Lalu variabel pwmChannel1, pwmChannel2. req, dan res dideklarasikan. pwmChannel1 dan pwmChannel2 merupakan nomor saluran PWM yang digunakan. Variabel freq merupakan frekuensi PWM yang diatur menjadi 15 kHz untuk meminimalisir *noise* pada motor ketika bekerja. Variabel res digunakan untuk mengatur resolusi dari PWM yang diatur menjadi 8-bit. Variabel PWM1_DutyCycle akan menyimpan siklus tugas yang kemudian akan digunakan untuk mengendalikan kecepatan mo-

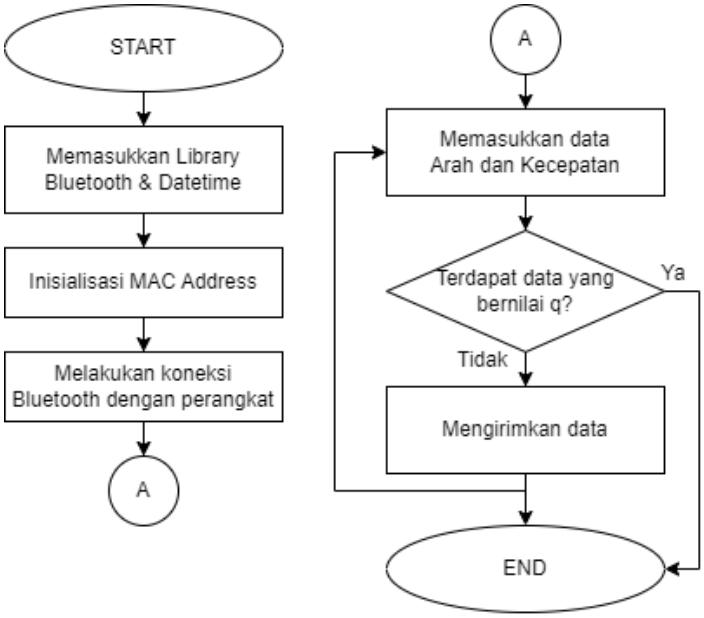
tor. Variabel maxspeed digunakan untuk mengatur kecepatan maksimal dari motor dan variabel turnspeed digunakan untuk kecepatan maksimal ketika kursi roda berbelok.

Fungsi void setup() dieksekusi satu kali pada saat program pertama kali dijalankan. Pada fungsi ini terdapat Serial.begin() yang digunakan untuk menginisialisasi komunikasi serial dengan kecepatan 115200 bps. Selanjutnya *access point* akan dibuat sesuai dengan ssid dan *password* yang telah ditentukan. IP *Address* kemudian akan ditampilkan pada *Serial Monitor*. Lalu server akan dimulai pada port 80. dir1, dir2, dir3, dan dir4 diatur sebagai *output*. ledcSetup merupakan fungsi dari *library* ledc yang digunakan untuk menghubungkan suatu pin dengan saluran PWM tertentu. Fungsi ini memerlukan 2 variabel, yaitu pin PWM dan juga PWM *Channel*. Secara keseluruhan blok kode ini digunakan untuk melakukan konfigurasi pin-pin pada mikrokontroler ESP32 sehingga motor dapat dikendalikan. Pin-pin yang diatur sebagai *output* akan digunakan untuk mengatur arah putar motor. Sedangkan saluran PWM akan digunakan untuk mengatur kecepatan putar motor.

Pada fungsi void loop() terdapat perintah untuk memeriksa ketersediaan koneksi *client*. Jika terdapat *client* yang terhubung maka ESP32 akan memeriksa apakah *client* masih tetap terhubung. Jika terdapat data yang tersedia maka data tersebut akan dibaca hingga terdapat karakter *newline* ('\n'). Data yang diterima kemudian akan diurai dan dimasukkan kedalam variabel arah. Kontrol arah gerak motor menggunakan pernyataan kondisional *if-else* berdasarkan variabel arah yang diterima. Pada setiap kondisi terdapat perulangan *while*. Hal ini dilakukan agar perubahan kecepatan motor tidak terjadi secara tiba-tiba. digitalWrite digunakan untuk menetapkan arah putar motor. ledcWrite digunakan untuk mengendalikan kecepatan motor. Perulangan ini terus dilakukan selama ESP32 masih menerima data melalui WiFi.

3.3.9 Program Untuk Mengirim Data String Melalui Bluetooth

Program ini menggunakan bahasa pemrograman Python. Program ini dirancang untuk mengirim data string ke perangkat ESP32 melalui Bluetooth. Berikut merupakan Program 5.9 yang digunakan untuk mengirim data string melalui Bluetooth beserta *flowchart* sesuai Gambar 3.11.



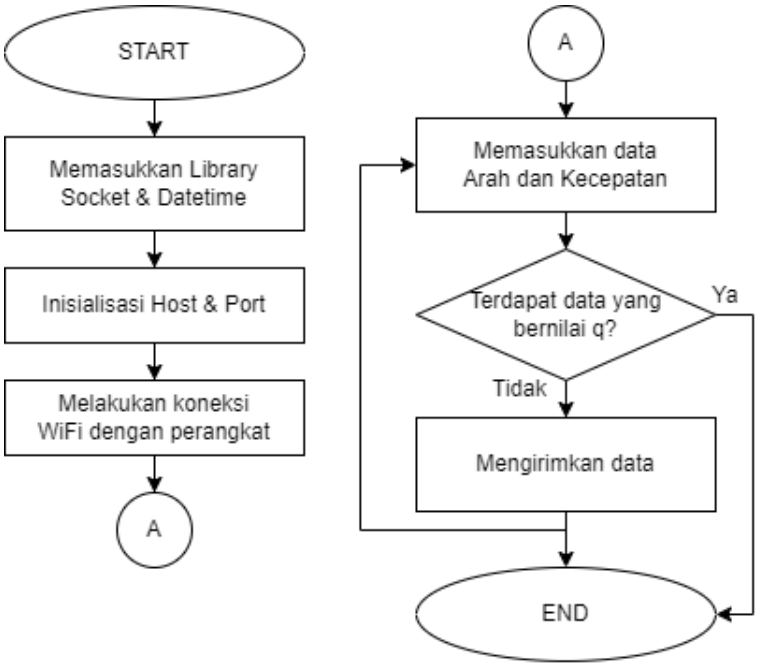
Gambar 3.11: *Flowchart* Mengirim Data String Melalui Bluetooth

Program ini menggunakan *library* Bluetooth untuk berinteraksi dengan perangkat lain melalui Bluetooth dan juga *library* datetime yang digunakan untuk mendapatkan informasi waktu secara *realtime*. Lalu menginisialisasi MAC *Address* dari perangkat ESP32 yang akan dikendalikan. Objek soket Bluetooth dibuat menggunakan fungsi BluetoothSocket lalu menghubungkannya dengan perangkat ESP32 melalui MAC *Address* dan nomor *channel*. Pada umumnya, *channel* 1 digunakan untuk profil *Serial Port Profile* (SPP)

Kemudian program berjalan pada perulangan yang tak terbatas. Pengguna diminta untuk memasukkan nilai arah dan kecepatan melalui *input*. Data tersebut kemudian akan digabungkan menjadi satu pesan dan dikirimkan ke perangkat ESP32 melalui soket Bluetooth. Apabila pengguna memasukkan ”q” sebagai nilai arah maupun kecepatan, maka perulangan akan dihentikan dan soket Bluetooth akan ditutup.

3.3.10 Program Untuk Mengirim Data String Melalui WiFi

Program ini menggunakan bahasa pemrograman Python. Program ini dirancang untuk mengirim data string ke perangkat ESP32 melalui WiFi. ESP32 akan bertindak sebagai *Access Point*. Berikut merupakan Program 5.10 yang digunakan untuk mengirimkan data string melalui WiFi beserta *flowchart* sesuai Gambar 3.12.



Gambar 3.12: *Flowchart* Mengirim Data String Melalui WiFi

Program ini menggunakan *library* socket, time, dan datetime. *Library* socket menyediakan antarmuka untuk membuat dan berinteraksi dengan soket sehingga dapat membuat koneksi dan mengirim ataupun menerima data melalui jaringan. *Library* time menyediakan fungsi-fungsi terkait waktu dan pengukuran waktu. Fungsi yang sering digunakan adalah time.sleep() yang berguna untuk menghentikan eksekusi program selama jumlah waktu tertentu. *Library* datetime digunakan untuk mendapatkan informasi waktu secara *realtime*. Selanjutnya IP *Address* dari *Access Point* ESP32 dimasukkan kedalam variabel host. Nomor port yang digunakan untuk koneksi akan dimasukkan kedalam variabel port. Umumnya menggunakan port 80.

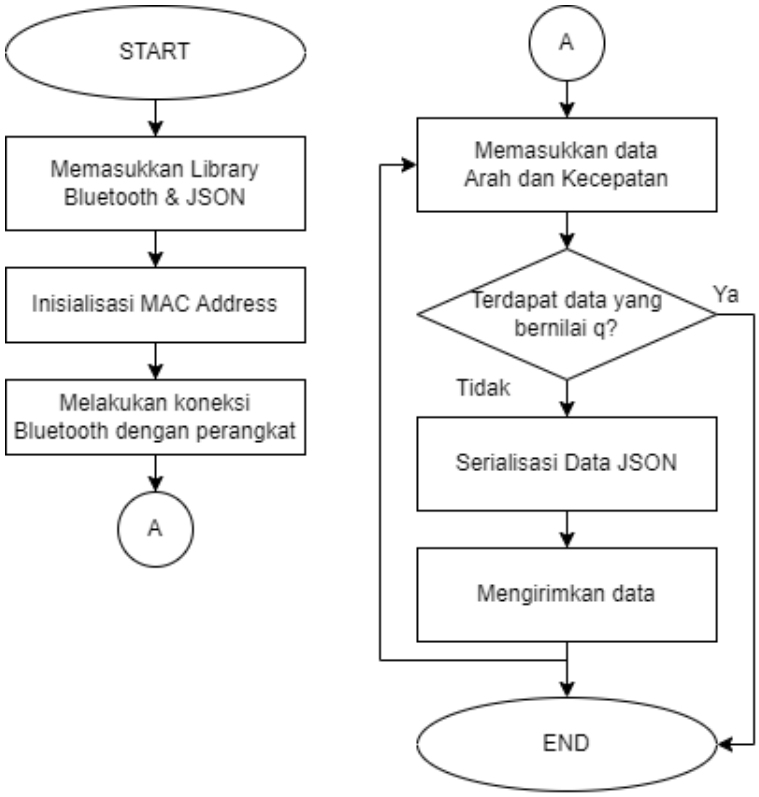
Objek soket kemudian dibuat menggunakan socket.socket() dengan alamat IPv4 dan tipe

soket streaming. Ini menunjukkan bahwa program ini akan menggunakan protokol TCP/IP untuk koneksi. Kemudian mencoba untuk membuat koneksi ke server yang telah ditentukan sesuai IP *Address* host dan nomor port.

Kemudian program akan berjalan pada perulangan yang tak terbatas. Pengguna diminta untuk memasukkan nilai arah dan kecepatan melalui *input*. Data tersebut kemudian digabungkan menjadi 1 pesan. Pesan kemudian dikonversi menjadi *byte* menggunakan *encoding* UTF-8. Setelah dikonversi maka pesan tersebut akan dikirimkan ke server. Apabila pengguna memasukkan ”q” sebagai nilai arah maupun kecepatan, maka perulangan akan dihentikan dan program akan menutup koneksi soket.

3.3.11 Program Untuk Mengirim Data JSON Melalui Bluetooth

Program ini menggunakan bahasa pemrograman Python. Program ini dirancang untuk mengirimkan data JSON melalui Bluetooth. Berikut merupakan Program 5.11 yang digunakan untuk mengirim data JSON melalui Bluetooth beserta *flowchart* sesuai Gambar 3.13.



Gambar 3.13: *Flowchart* Mengirim Data JSON Melalui Bluetooth

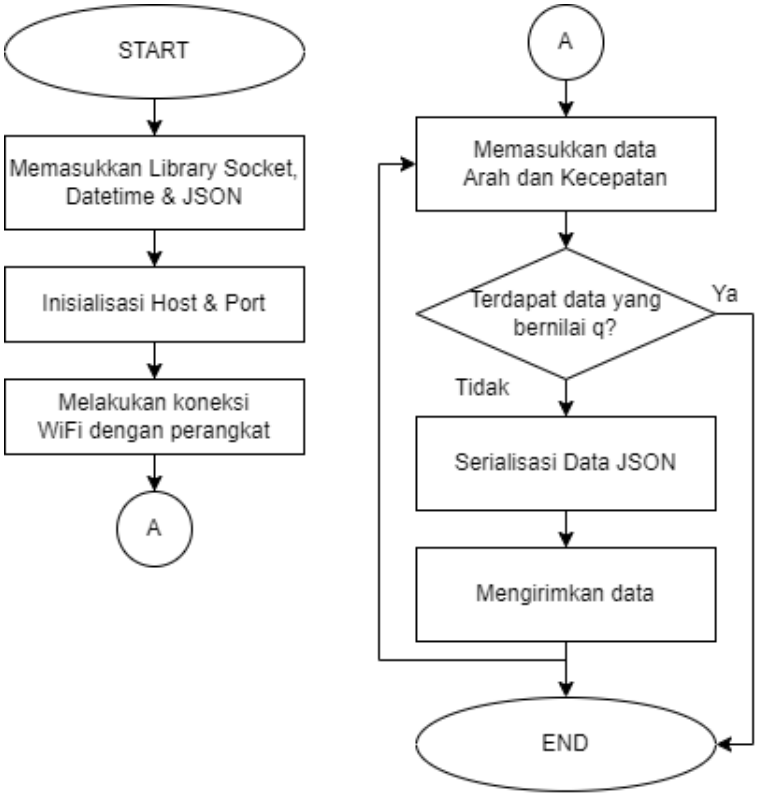
Program ini menggunakan *library* Bluetooth untuk berinteraksi dengan perangkat lain melalui Bluetooth dan *library* JSON untuk mengelola data dalam format JSON. Alamat Bluetooth dari perangkat ESP32 yang akan dikendalikan kemudian dimasukkan kedalam suatu variabel. Objek soket Bluetooth kemudian dibuat dengan menggunakan protokol *Radio Frequency Communication*(RFCOMM). RFCOMM merupakan protokol yang umum digunakan untuk komunikasi serial melalui Bluetooth. Soket Bluetooth kemudian dihubungkan ke perangkat ESP32 menggunakan alamat Bluetooth dan nomor *channel*. Pada umumnya *channel* 1 digunakan untuk profil SPP (*Serial Port Profile*).

Kemudian program akan berjalan dalam perulangan yang tak terbatas. Pengguna diminta

untuk memasukkan nilai arah dan kecepatan melalui *input*. Objek JSON kemudian dibuat untuk menyimpan data yang telah dimasukkan. Objek JSON kemudian dikonversi menjadi string menggunakan json.dumps(). String JSON kemudian akan dikirimkan melalui koneksi Bluetooth menggunakan sock.send(). Apabila pengguna memasukkan ”q” sebagai nilai arah maupun kecepatan, maka perulangan akan dihentikan dan program akan menutup koneksi soket.

3.3.12 Program Untuk Mengirim Data JSON Melalui WiFi

Program ini dibuat dengan menggunakan bahasa pemrograman Python. Program ini dirancang untuk mengirim data JSON ke perangkat ESP32 melalui WiFi. ESP32 akan bertindak sebagai *Access Point*. Berikut merupakan Program 5.12 yang digunakan untuk mengirimkan data JSON melalui WiFi beserta *flowchart* sesuai Gambar 3.14.



Gambar 3.14: *Flowchart* Mengirim Data JSON Melalui WiFi

Program ini menggunakan *library* socket, time, json, dan datetime. *Library* socket menyediakan antarmuka untuk membuat dan berinteraksi dengan soket sehingga dapat membuat koneksi dan mengirim ataupun menerima data melalui jaringan. *Library* time menyediakan fungsi-fungsi terkait waktu dan pengukuran waktu. Fungsi yang sering digunakan adalah time.sleep() yang berguna untuk menghentikan eksekusi program selama jumlah waktu tertentu. *Library* datetime digunakan untuk mendapatkan informasi waktu secara *realtime*. *Library* JSON digunakan untuk mengelola data dalam bentuk JSON. Selanjutnya IP *Address* dari *Access Point* ESP32 dimasukkan kedalam variabel host. Nomor port yang digunakan untuk koneksi akan dimasukkan kedalam variabel port. Umumnya menggunakan port 80.

Kemudian program akan berjalan dalam perulangan yang tak terbatas. Pengguna diminta untuk memasukkan nilai arah dan kecepatan melalui *input*. Objek JSON kemudian dibuat untuk menyimpan data yang telah dimasukkan. Objek JSON kemudian dikonversi menjadi string

menggunakan json.dumps(). String JSON kemudian akan dikonversi menjadi *byte* menggunakan *encoding* UTF-8. Setelah dikonversi maka akan dikirimkan melalui koneksi WiFi menggunakan s.send(). Apabila pengguna memasukkan ”q” sebagai nilai arah maupun kecepatan, maka perulangan akan dihentikan dan program akan menutup koneksi soket.

BAB IV
PENGUJIAN DAN ANALISIS

Pada bab ini akan dipaparkan mengenai beberapa skenario pengujian sesuai dengan telah dijelaskan pada metodologi. Skenario pengujian ini dilakukan guna untuk mengetahui waktu *delay* yang dibutuhkan untuk mentransmisikan data dari laptop menuju ESP32. Skenario yang nantinya akan diterapkan pada pengujian meliputi beberapa poin sebagai berikut:

1. Pengujian waktu *delay* pengiriman data String melalui Bluetooth
2. Pengujian waktu *delay* pengiriman data JSON melalui Bluetooth
3. Pengujian waktu *delay* pengiriman data String melalui *Access Point* WiFi
4. Pengujian waktu *delay* pengiriman data JSON melalui *Access Point* WiFi
5. Pengujian kestabilan Motor Kursi Roda

Pelaksanaan metodologi serta skenario pengujian yang akan dipaparkan dalam bab ini diharapkan dapat memberikan pemahaman mengenai hasil dan pembahasan sehingga dapat ditarik kesimpulan dari Tugas Akhir yang telah dilaksanakan.

4.1 Pengujian Waktu *Delay* Pengiriman Data String Melalui Bluetooth

Pengujian waktu *delay* pengiriman data ini dilakukan dengan cara mengirimkan data berupa string dari laptop menuju ESP32 melalui Bluetooth. Data yang dikirimkan adalah data arah dan kecepatan yang dipisahkan dengan koma seperti yang dapat dilihat pada Persamaan 4.1.

$$Arah, Kecepatan \tag{4.1}$$

Keterangan

Arah : Variabel dengan tipe data *char* yang digunakan untuk mengatur arah gerak motor kursi roda

Kecepatan : Variabel dengan tipe data *char* yang digunakan untuk mengatur kecepatan putar motor kursi roda

Variabel arah memiliki tipe data *char* yang digunakan untuk menentukan arah gerak dari motor kursi roda serta variabel kecepatan memiliki tipe data *char* yang akan menentukan kecepatan maksimal dari rotasi motor kursi roda. Hasil dari pengujian waktu *delay* pengiriman data string melalui bluetooth dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1: Pengujian Waktu *Delay* Pengiriman Data String Berisi 2 Nilai Melalui Bluetooth

Data	Timestamp Sent	Timestamp Received	Delay Time
A,S	22:35:51.523626	22:35:52.588	1,064374
E,T	22:35:52.552591	22:35:53.591	1,038409
E,P	22:35:53.565132	22:35:54.610	1,044868
B,S	22:35:54.579935	22:35:55.602	1,022065

D,U	22:35:55.593733	22:35:56.619	1,025267
C,P	22:35:56.609782	22:35:57.624	1,014218
A,R	22:35:57.618772	22:35:58.666	1,047228
D,Q	22:39:03.614853	22:39:04.672	1,057147
C,O	22:39:04.651688	22:39:05.690	1,038312
B,T	22:39:05.666397	22:39:06.679	1,012603
B,S	22:39:06.676370	22:39:07.711	1,03463
B,T	22:39:07.686293	22:39:08.724	1,037707
B,T	22:39:08.692595	22:39:09.729	1,036405
C,O	22:41:01.427643	22:41:02.485	1,057357
D,U	22:41:02.443022	22:41:03.486	1,042978
A,U	22:41:03.449566	22:41:04.485	1,035434
B,O	22:41:04.463072	22:41:05.507	1,043928
D,O	22:41:05.470721	22:41:06.493	1,022279
C,T	22:41:06.477458	22:41:07.512	1,034542
B,U	22:41:07.492309	22:41:08.516	1,023691
D,R	22:41:08.499671	22:41:09.532	1,032329
E,Q	22:41:09.509928	22:41:10.547	1,037072
E,V	22:41:10.523644	22:41:11.563	1,039356
C,U	22:43:11.001438	22:43:12.026	1,024562
C,P	22:43:12.028891	22:43:13.072	1,043109
A,T	22:43:13.035252	22:43:14.066	1,030748
D,Q	22:43:14.044212	22:43:15.086	1,041788
D,O	22:43:15.050093	22:43:16.087	1,036907
C,V	22:44:38.377450	22:44:39.438	1,06055
B,P	22:44:39.427390	22:44:40.461	1,03361
Average Delay Time			1,037115767

Tabel 4.1 menampilkan pengiriman data string yang terdiri dari 2 nilai dan dipisahkan dengan simbol koma (”,”). Dalam pengujian kali ini, data dikirimkan sebanyak 30 kali secara berturut-turut. Setelah ESP32 menerima dan mengolah data maka Flag akan dikirimkan ke Laptop. Hal ini dilakukan agar ESP32 dapat menerima data dengan baik dan berhasil memisahkan dan memasukkan kedua nilai tersebut sesuai dengan variabel yang telah ditentukan. Hasil dari pengujian ini menunjukkan bahwa waktu pengiriman rata-rata dari laptop menuju ESP32 melalui Bluetooth adalah sebesar 1,037115767 detik.

Tabel 4.2: Pengujian Waktu *Delay* Pengiriman Data String Berisi 1 Nilai Melalui Bluetooth

Data	Timestamp Sent	Timestamp Received	Delay Time
B	22:17:58.360200	22:17:59.417	1,0568
B	22:17:59.396644	22:18:00.426	1,029356
B	22:18:00.411642	22:18:01.449	1,037358
C	22:18:01.419172	22:18:02.461	1,041828
E	22:18:02.431706	22:18:03.436	1,004294

B	22:18:03.441714	22:18:04.487	1,045286
A	22:18:04.452937	22:18:05.481	1,028063
B	22:18:05.468058	22:18:06.502	1,033942
C	22:18:06.475522	22:18:07.504	1,028478
A	22:18:07.483055	22:18:08.508	1,024945
C	22:19:35.549304	22:19:36.610	1,060696
A	22:19:36.580394	22:19:37.624	1,043606
B	22:19:37.587334	22:19:38.612	1,024666
B	22:19:38.593528	22:19:39.645	1,051472
E	22:19:39.605674	22:19:40.634	1,028326
E	22:19:40.621872	22:19:41.643	1,021128
A	22:19:41.633767	22:19:42.668	1,034233
D	22:19:42.641717	22:19:43.675	1,033283
E	22:19:43.650596	22:19:44.682	1,031404
D	22:19:44.659460	22:19:45.698	1,03854
A	22:20:22.820901	22:20:23.885	1,064099
E	22:20:23.850953	22:20:24.900	1,049047
B	22:20:24.867989	22:20:25.873	1,005011
E	22:20:25.879869	22:20:26.919	1,039131
C	22:20:26.890743	22:20:27.925	1,034257
D	22:20:27.899877	22:20:28.944	1,044123
B	22:20:28.911163	22:20:29.917	1,005837
A	22:20:29.918844	22:20:30.969	1,050156
E	22:20:30.945381	22:20:31.991	1,045619
E	22:20:31.972686	22:20:32.994	1,021314
Average Delay Time			1,035209933

Tabel 4.2 menampilkan pengiriman data string yang terdiri dari 1 nilai, yaitu arah. Dalam pengujian kali ini, data dikirimkan sebanyak 30 kali secara berturut-turut tanpa penambahan waktu *delay* setiap kali mengirimkan data. Setelah ESP32 menerima dan mengolah data maka Flag akan dikirimkan ke Laptop. Hal ini dilakukan agar ESP32 dapat menerima data dengan baik dan berhasil memisahkan dan memasukkan kedua nilai tersebut sesuai dengan variabel yang telah ditentukan. Hasil dari pengujian ini menunjukkan bahwa waktu pengiriman rata-rata dari laptop menuju ESP32 melalui Bluetooth adalah sebesar 1,035209933 detik.

Ditemukan perbedaan waktu *delay* yang tidak terlalu signifikan antara proses pengiriman string yang mengandung 2 nilai dibandingkan dengan string yang hanya berisi 1 nilai. Saat mengirimkan string yang memuat 2 data, pengujian menunjukkan adanya waktu *delay* sekitar 1,037115767 detik. Dalam perbandingan dengan pengujian yang melibatkan string 1 nilai, waktu *delay* yang tercatat hanya sekitar 1,035209933 detik. Setelah ESP32 menerima dan mengolah data maka Flag akan dikirimkan ke Laptop. Hal ini dilakukan agar ESP32 dapat menerima data secara optimal. Oleh karena itu, dapat disimpulkan bahwa dalam konteks pengiriman data melalui Bluetooth, pengguna dapat mengirim data string baik yang berisi 1 nilai maupun data string yang berisi 2 nilai, karena perbedaan waktu delay yang tidak terlalu berbeda.

4.2 Pengujian Waktu *Delay* Pengiriman Data JSON Melalui Bluetooth

Pengujian waktu *delay* pengiriman data ini dilakukan dengan cara mengirimkan data berupa JSON dari laptop menuju ESP32 melalui Bluetooth. Data yang dikirimkan adalah data arah dan kecepatan yang dirangkum menjadi JSON seperti yang dapat dilihat pada Persamaan 4.2 dan juga JSON yang hanya berisikan data arah seperti pada Persamaan 4.3

$$\{'arah' : kode - arah,' kecepatan' : level - kecepatan\} \tag{4.2}$$

Keterangan
arah : Key yang digunakan untuk menyimpan nilai kode-arah
kecepatan : Key yang digunakan untuk menyimpan nilai level-kecepatan
kode-arah : Value dengan tipe data *char* yang digunakan untuk mengatur arah gerak motor kursi roda
level-kecepatan : Value dengan tipe data *char* yang digunakan untuk mengatur kecepatan putar motor kursi roda

$$\{'arah' : kode - arah\} \tag{4.3}$$

Keterangan
arah : Key yang digunakan untuk menyimpan nilai kode-arah
kode-arah : Value dengan tipe data *char* yang digunakan untuk mengatur arah gerak motor kursi roda

Data JSON terdiri dari 2 bagian, yaitu *key* dan *value*. Terdapat 2 *key*, yaitu arah dan kecepatan. *Key* arah berisi *value* dengan tipe data char yang digunakan untuk menentukan arah gerak dari motor kursi roda dan *key* kecepatan berisi *value* dengan tipe data *char* yang digunakan untuk menentukan kecepatan maksimal dari rotasi motor kursi roda. Hasil dari pengujian waktu *delay* pengiriman data JSON dapat dilihat pada Tabel 4.3 dan Tabel 4.4.

Tabel 4.3: Pengujian Waktu *Delay* Pengiriman Data JSON Berisi 2 Nilai Melalui Bluetooth

Data	Timestamp Sent	Timestamp Received	Delay Time
{'arah': 'A', 'kecepatan': 'W'}	00:15:50.714083	00:15:51.776	1,061917
{"arah": "D", "kecepatan": "T"}	00:15:51.747698	00:15:52.796	1,048302
{"arah": "E", "kecepatan": "S"}	00:15:52.756331	00:15:53.766	1,009669
{"arah": "D", "kecepatan": "V"}	00:15:53.768717	00:15:54.807	1,038283
{"arah": "B", "kecepatan": "W"}	00:15:54.784688	00:15:55.824	1,039312
{"arah": "D", "kecepatan": "Q"}	00:15:55.792741	00:15:56.841	1,048259
{"arah": "A", "kecepatan": "O"}	00:15:56.803679	00:15:57.848	1,044321
{"arah": "E", "kecepatan": "Q"}	00:15:57.809707	00:15:58.854	1,044293
{"arah": "A", "kecepatan": "T"}	00:15:58.828668	00:15:59.840	1,011332
{"arah": "B", "kecepatan": "U"}	00:15:59.837626	00:16:00.875	1,037374
{"arah": "A", "kecepatan": "V"}	00:16:00.852674	00:16:01.888	1,035326
{"arah": "C", "kecepatan": "Q"}	00:16:01.867249	00:16:02.895	1,027751
{"arah": "A", "kecepatan": "O"}	00:16:02.885202	00:16:03.932	1,046798
{"arah": "A", "kecepatan": "V"}	00:16:03.910200	00:16:04.953	1,0428

{”arah”: ”B”, ”kecepatan”: ”T”}	00:16:04.917273	00:16:05.962	1,044727
{”arah”: ”D”, ”kecepatan”: ”W”}	00:16:05.925647	00:16:06.947	1,021353
{”arah”: ”E”, ”kecepatan”: ”P”}	00:16:06.942693	00:16:07.981	1,038307
{”arah”: ”E”, ”kecepatan”: ”R”}	00:16:07.955136	00:16:09.004	1,048864
{”arah”: ”A”, ”kecepatan”: ”T”}	00:16:08.966673	00:16:10.013	1,046327
{”arah”: ”E”, ”kecepatan”: ”P”}	00:16:09.977653	00:16:11.029	1,051347
{”arah”: ”E”, ”kecepatan”: ”O”}	00:16:10.985241	00:16:12.033	1,047759
{”arah”: ”C”, ”kecepatan”: ”R”}	00:16:11.997910	00:16:13.010	1,01209
{’arah’: ’A’, ’kecepatan’: ’V’}	02:33:35.225933	02:33:36.305	1,079067
{”arah”: ”D”, ”kecepatan”: ”U”}	02:33:36.263807	02:33:37.288	1,024193
{”arah”: ”A”, ”kecepatan”: ”R”}	02:33:37.274213	02:33:38.297	1,022787
{”arah”: ”C”, ”kecepatan”: ”W”}	02:33:38.309262	02:33:39.356	1,046738
{”arah”: ”B”, ”kecepatan”: ”U”}	02:33:39.324412	02:33:40.382	1,057588
{”arah”: ”A”, ”kecepatan”: ”O”}	02:33:40.394353	02:33:41.424	1,029647
{”arah”: ”B”, ”kecepatan”: ”T”}	02:33:41.411566	02:33:42.433	1,021434
Average Delay Time			1,038895345

Tabel 4.3 menampilkan pengiriman JSON yang terdiri dari 2 *key-value*, yaitu arah dan kecepatan. Dalam pengujian kali ini, data dikirimkan sebanyak 29 kali secara berturut-turut. Setelah ESP32 menerima dan mengolah data maka Flag akan dikirimkan ke Laptop. Hal ini dilakukan agar ESP32 dapat menerima data dengan baik dan berhasil memisahkan (*deserialize*) serta memasukkan kedua nilai tersebut sesuai dengan variabel yang telah ditentukan. Hasil dari pengujian ini menunjukkan bahwa waktu pengiriman rata-rata dari laptop menuju ESP32 melalui Bluetooth adalah sebesar 1,038895345 detik.

Tabel 4.4: Pengujian Waktu *Delay* Pengiriman Data JSON Berisi 1 Nilai Melalui Bluetooth

Data	Timestamp Sent	Timestamp Received	Delay Time
{’arah’: ’D’}	23:51:13.043594	23:51:14.086	1,042406
{”arah”: ”A”}	23:51:14.076700	23:51:15.105	1,0283
{”arah”: ”D”}	23:51:15.086795	23:51:16.117	1,030205
{”arah”: ”E”}	23:51:16.099483	23:51:17.140	1,040517
{”arah”: ”C”}	23:51:17.105868	23:51:18.149	1,043132
{”arah”: ”C”}	23:51:18.117818	23:51:19.166	1,048182
{”arah”: ”E”}	23:51:19.127469	23:51:20.146	1,018531
{”arah”: ”B”}	23:51:20.135577	23:51:21.166	1,030423
{”arah”: ”C”}	23:51:21.145479	23:51:22.190	1,044521
{”arah”: ”C”}	23:51:22.158915	23:51:23.176	1,017085
{”arah”: ”E”}	23:51:23.168461	23:51:24.180	1,011539
{”arah”: ”D”}	23:51:24.180958	23:51:25.207	1,026042
{”arah”: ”A”}	23:51:25.194500	23:51:26.199	1,0045
{”arah”: ”C”}	23:51:26.203449	23:51:27.244	1,040551
{”arah”: ”A”}	23:51:27.216460	23:51:28.254	1,03754
{”arah”: ”A”}	23:51:28.224202	23:51:29.260	1,035798

{”arah”: ”C”}	23:51:29.236461	23:51:30.280	1,043539
{”arah”: ”A”}	23:51:30.243411	23:51:31.287	1,043589
{”arah”: ”D”}	23:51:31.253768	23:51:32.293	1,039232
{”arah”: ”A”}	23:51:32.266472	23:51:33.301	1,034528
{”arah”: ”C”}	23:51:33.279736	23:51:34.327	1,047264
{”arah”: ”D”}	23:51:34.288698	23:51:35.332	1,043302
{”arah”: ”C”}	23:51:35.303347	23:51:36.345	1,041653
{”arah”: ”E”}	23:51:36.323448	23:51:37.351	1,027552
{”arah”: ”E”}	23:51:37.331450	23:51:38.357	1,02555
{”arah”: ”C”}	23:51:38.341408	23:51:39.383	1,041592
{”arah”: ”A”}	23:51:39.351443	23:51:40.356	1,004557
{”arah”: ”B”}	23:51:40.366713	23:51:41.412	1,045287
{”arah”: ”D”}	23:51:41.377986	23:51:42.415	1,037014
{”arah”: ”E”}	23:51:42.392035	23:51:43.421	1,028965
{”arah”: ”B”}	23:51:43.406513	23:51:44.431	1,024487
{”arah”: ”C”}	23:51:44.424451	23:51:45.464	1,039549
{”arah”: ”C”}	23:51:45.430967	23:51:46.477	1,046033
{”arah”: ”C”}	23:51:46.438140	23:51:47.486	1,04786
{”arah”: ”B”}	23:51:47.449801	23:51:48.476	1,026199
{”arah”: ”C”}	23:51:48.457089	23:51:49.496	1,038911
{”arah”: ”E”}	23:51:49.471782	23:51:50.496	1,024218
{”arah”: ”B”}	23:51:50.487487	23:51:51.525	1,037513
{”arah”: ”D”}	23:51:51.496917	23:51:52.538	1,041083
{”arah”: ”D”}	23:51:52.504746	23:51:53.528	1,023254
{”arah”: ”B”}	23:51:53.519392	23:51:54.569	1,049608
{”arah”: ”D”}	23:51:54.526676	23:51:55.555	1,028324
{”arah”: ”B”}	23:51:55.539415	23:51:56.571	1,031585
{”arah”: ”E”}	23:51:56.552679	23:51:57.576	1,023321
Average Delay Time			1,033746386

Tabel 4.4 menampilkan pengiriman JSON yang terdiri dari 1 *key-value*, yaitu arah. Dalam pengujian kali ini, data dikirimkan sebanyak 44 kali secara berturut-turut. Setelah ESP32 menerima dan mengolah data maka Flag akan dikirimkan ke Laptop. Hal ini dilakukan agar ESP32 dapat menerima data dengan baik dan berhasil memisahkan (*deserialize*) serta memasukkan nilai tersebut sesuai dengan variabel yang telah ditentukan. Hasil dari pengujian ini menunjukkan waktu pengiriman rata-rata dari laptop menuju ESP32 melalui Bluetooth sebesar 1,033746386 detik.

Dari kedua cara pengujian tersebut, didapatkan bahwa waktu *delay* antara proses pengiriman JSON yang mengandung 2 *key-value* dibandingkan dengan JSON yang hanya berisikan 1 *key-value* tidak terlalu signifikan. Saat mengirimkan JSON yang berisikan 2 *key-value*, pengujian menunjukkan adanya *delay* sebesar 1,038895345 detik. Jika dibandingkan dengan pengujian yang mengirimkan JSON dengan 1 *key-value*, waktu *delay* yang tercatat adalah sebesar 1,033746386 detik. Dapat disimpulkan bahwa dalam konteks pengiriman data melalui Bluetooth, maka pengguna dapat mengirimkan data JSON baik yang berisi 1 *key-value* maupun yang berisi 2 *key-value*, karena perbedaan *delay* kedua teknik tersebut tidak terlalu berbeda.

4.3 Pengujian Waktu *Delay* Pengiriman Data String Melalui *Access Point* WiFi

Pengujian waktu *delay* pengiriman data ini dilakukan dengan cara mengirimkan data berupa String dari laptop menuji ESP32 melalui WiFi. ESP32 diatur sebagai *Access Point*. Data yang dikirimkan adalah data arah dan kecepatan yang dipisahkan dengan simbol koma seperti yang dapat dilihat pada Persamaan 4.4

$$Arah, Kecepatan \tag{4.4}$$

Keterangan

Arah : Variabel dengan tipe data *char* yang digunakan untuk mengatur arah gerak motor kursi roda

Kecepatan : Variabel dengan tipe data *char* yang digunakan untuk mengatur kecepatan putar motor kursi roda

Variabel arah memiliki tipe data char yang digunakan untuk menentukan arah gerak dari motor kursi roda serta variabel kecepatan yang memiliki tipe data *char* yang akan menentukan kecepatan maksimal dari rotasi motor kursi roda. Hasil dari pengujian waktu *delay* pengiriman data String melalui WiFi dapat dilihat pada Tabel 4.5 dan Tabel 4.6

Tabel 4.5: Pengujian Waktu *Delay* Pengiriman Data String Berisi 2 Nilai Melalui *Access Point* WiFi

Data	Timestamp Sent	Timestamp Received	Delay Time
A,Q	03:49:59.217022	03:50:00.256	1,038978
D,S	03:50:00.232977	03:50:01.260	1,027023
A,O	03:50:01.256289	03:50:02.301	1,044711
B,Q	03:50:02.278635	03:50:03.317	1,038365
A,P	03:50:03.303124	03:50:04.336	1,032876
E,U	03:50:04.326652	03:50:05.374	1,047348
B,U	03:50:05.365324	03:50:06.411	1,045676
D,R	03:50:06.374651	03:50:07.413	1,038349
B,U	03:50:07.403115	03:50:08.432	1,028885
E,Q	03:50:08.422259	03:50:09.469	1,046741
D,S	03:50:09.452763	03:50:10.486	1,033237
A,Q	03:50:10.470206	03:50:11.489	1,018794
B,V	03:50:11.513787	03:50:12.556	1,042213
D,S	03:50:12.519142	03:50:13.559	1,039858
B,U	03:50:13.547184	03:50:14.595	1,047816
C,S	03:50:14.566997	03:50:15.588	1,021003
D,R	03:50:15.600210	03:50:16.638	1,03779
C,W	03:50:16.616353	03:50:17.645	1,028647
E,O	03:50:17.628725	03:50:18.664	1,035275
C,Q	03:50:18.640125	03:50:19.654	1,013875
D,U	03:50:19.645720	03:50:20.695	1,04928
E,T	03:50:20.652242	03:50:21.674	1,021758

A,S	03:50:21.664690	03:50:22.708	1,04331
B,T	03:50:22.668332	03:50:23.705	1,036668
E,P	03:50:23.671113	03:50:24.716	1,044887
E,S	03:50:24.677001	03:50:25.719	1,041999
A,V	03:50:25.683752	03:50:26.717	1,033248
E,U	03:50:26.692210	03:50:27.719	1,02679
C,V	03:50:27.696829	03:50:28.738	1,041171
C,U	03:50:28.715728	03:50:29.738	1,022272
D,R	03:50:29.719687	03:50:30.740	1,020313
B,U	03:50:30.725229	03:50:31.760	1,034771
B,R	03:50:31.731151	03:50:32.778	1,046849
Average Delay Time			1,035478061

Tabel 4.5 menampilkan pengiriman data string yang terdiri dari 2 nilai dan dipisahkan dengan simbol koma (”,”). Dalam pengujian kali ini, data dikirimkan sebanyak 33 kali secara berturut-turut. Setelah ESP32 menerima dan mengolah data maka Flag akan dikirimkan ke Laptop. Hal ini dilakukan agar ESP32 dapat menerima data dengan baik dan berhasil memisahkan dan memasukkan kedua nilai tersebut sesuai dengan variabel yang telah ditentukan. Hasil dari pengujian ini menunjukkan bahwa waktu pengiriman rata-rata dari laptop menuju ESP32 melalui WiFi adalah sebesar 1,035478061 detik.

Tabel 4.6: Pengujian Waktu *Delay* Pengiriman Data String Berisi 1 Nilai Melalui *Access Point* WiFi

Data	Timestamp Sent	Timestamp Received	Delay Time
C	03:40:01.384466	03:40:02.417	1,032534
B	03:40:02.390604	03:40:03.432	1,041396
E	03:40:03.395418	03:40:04.429	1,033582
A	03:40:04.399194	03:40:05.446	1,046806
D	03:40:05.404434	03:40:06.444	1,039566
D	03:40:06.408148	03:40:07.414	1,005852
D	03:40:07.413181	03:40:08.460	1,046819
E	03:40:08.417232	03:40:09.449	1,031768
E	03:40:09.423408	03:40:10.467	1,043592
E	03:40:10.428216	03:40:11.449	1,020784
A	03:40:11.431451	03:40:12.495	1,063549
E	03:40:12.453247	03:40:13.477	1,023753
D	03:40:13.456520	03:40:14.496	1,03948
E	03:40:14.463187	03:40:15.480	1,016813
B	03:40:15.480592	03:40:16.512	1,031408
E	03:40:16.485499	03:40:17.507	1,021501
D	03:40:17.488464	03:40:18.512	1,023536
C	03:40:18.493367	03:40:19.521	1,027633
C	03:40:19.497250	03:40:20.543	1,04575
A	03:40:20.502595	03:40:21.527	1,024405

C	03:40:21.506182	03:40:22.547	1,040818
D	03:40:22.511194	03:40:23.540	1,028806
C	03:40:23.515176	03:40:24.547	1,031824
C	03:40:24.520190	03:40:25.554	1,03381
E	03:40:25.525565	03:40:26.574	1,048435
E	03:40:26.529479	03:40:27.580	1,050521
D	03:40:27.558203	03:40:28.596	1,037797
B	03:40:28.564173	03:40:29.588	1,023827
B	03:40:29.568581	03:40:30.600	1,031419
B	03:40:30.571593	03:40:31.619	1,047407
A	03:40:31.576199	03:40:32.612	1,035801
C	03:40:32.579504	03:40:33.619	1,039496
Average Delay Time			1,03470275

Tabel 4.6 menampilkan pengiriman data string yang terdiri dari 1 nilai, yaitu arah. Dalam pengujian kali ini, data dikirimkan sebanyak 32 kali secara berturut-turut. Setelah ESP32 menerima dan mengolah data maka Flag akan dikirimkan ke Laptop. Hasil dari pengujian ini menunjukkan bahwa waktu pengiriman rata-rata dari laptop menuju ESP32 melalui WiFi adalah sebesar 1,03470275 detik.

Ditemukan perbedaan waktu *delay* yang cukup signifikan antara proses pengiriman string yang mengandung 2 nilai dibandingkan dengan string yang hanya berisikan 1 nilai. Saat mengirimkan string yang memuat 2 data, pengujian menunjukkan adanya waktu *delay* sekitar 1.059681387 detik. Dalam perbandingan dengan pengujian yang melibatkan string dengan 1 nilai, waktu yang *delay* yang tercatat hanya 0.03499708571 detik. Penting untuk diperhatikan bahwa terdapat kekurangan saat mengirimkan data string yang mengandung 2 nilai, yaitu adanya penambahan *delay* sebesar 1.3 detik setiap kali pengiriman data dilakukan. Hal ini dilakukan agar ESP32 dapat menerima data secara optimal. Oleh karena itu, dapat disimpulkan bahwa dalam konteks pengiriman data melalui WiFi, lebih disarankan untuk mengirimkan string dengan hanya 1 nilai guna menghindari *delay* tambahan yang dapat mempengaruhi efisiensi dan kecepatan transmiri data secara keseluruhan.

4.4 Pengujian Waktu *Delay* Pengiriman Data JSON Melalui *Access Point* WiFi

Pengujian waktu *delay* pengiriman data ini dilakukan dengan cara mengirimkan data berupa JSON dari laptop menuju ESP32 melalui WiFi. Data yang dikirimkan adalah data arah dan kecepatan yang dirangkum menjadi JSON seperti yang dapat dilihat pada Persamaan 4.5 dan Persamaan 4.6.

$$\{'arah' : kode - arah,' kecepatan' : level - kecepatan\} \tag{4.5}$$

Keterangan
arah : Key yang digunakan untuk menyimpan nilai kode-arah
kecepatan : Key yang digunakan untuk menyimpan nilai level-kecepatan
kode-arah : Value dengan tipe data *char* yang digunakan untuk mengatur arah gerak motor kursi roda

level-kecepatan : Value dengan tipe data *char* yang digunakan untuk mengatur kecepatan putar motor kursi roda

$$\{'arah' : kode - arah\} \tag{4.6}$$

Keterangan

arah : Key yang digunakan untuk menyimpan nilai kode-arah

kode-arah : Value dengan tipe data *char* yang digunakan untuk mengatur arah gerak motor kursi roda

Data JSON terdiri dari 2 bagian, yaitu *key* dan *value*. Terdapat 2 *key*, yaitu arah dan kecepatan. *Key* arah berisi *value* dengan tipe data char yang digunakan untuk menentukan arah gerak dari motor kursi roda dan *key* kecepatan berisi *value* dengan tipe data *char* yang digunakan untuk menentukan kecepatan maksimal dari rotasi motor kursi roda. Hasil dari pengujian waktu *delay* pengiriman data JSON dapat dilihat pada Tabel 4.7 dan Tabel 4.8.

Tabel 4.7: Pengujian Waktu *Delay* Pengiriman Data JSON Berisi 2 Nilai Melalui *Access Point* WiFi

Data	Timestamp Sent	Timestamp Received	Delay Time
{'arah': 'C', 'kecepatan': 89}	18:58:14.500967	18:58:15.534	1.033033
{'arah': 'C', 'kecepatan': 103}	18:58:16.001310	18:58:17.091	1.08969
{'arah': 'C', 'kecepatan': 164}	18:58:17.502196	18:58:18.550	1.047804
{'arah': 'A', 'kecepatan': 101}	18:58:19.002654	18:58:20.026	1.023346
{'arah': 'A', 'kecepatan': 152}	18:58:20.502992	18:58:21.537	1.034008
{'arah': 'A', 'kecepatan': 19}	18:58:22.003740	18:58:23.042	1.03826
{'arah': 'A', 'kecepatan': 198}	18:58:23.504455	18:58:24.539	1.034545
{'arah': 'A', 'kecepatan': 211}	18:58:25.005622	18:58:26.045	1.039378
{'arah': 'E', 'kecepatan': 238}	18:58:26.506072	18:58:27.543	1.036928
{'arah': 'D', 'kecepatan': 30}	18:58:28.006535	18:58:29.175	1.168465
{'arah': 'D', 'kecepatan': 112}	18:58:29.507104	18:58:30.532	1.024896
{'arah': 'D', 'kecepatan': 7}	18:58:31.007617	18:58:32.011	1.003383
{'arah': 'C', 'kecepatan': 66}	18:58:32.508185	18:58:33.547	1.038815
{'arah': 'A', 'kecepatan': 105}	18:58:34.008584	18:58:35.051	1.042416
{'arah': 'B', 'kecepatan': 117}	18:58:35.509099	18:58:36.526	1.016901
{'arah': 'D', 'kecepatan': 223}	18:58:37.009885	18:58:38.037	1.027115
{'arah': 'A', 'kecepatan': 245}	18:58:38.510667	18:58:39.515	1.004333
{'arah': 'A', 'kecepatan': 15}	18:58:40.011508	18:58:41.050	1.038492
{'arah': 'D', 'kecepatan': 156}	18:58:41.512148	18:58:42.543	1.030852
{'arah': 'C', 'kecepatan': 118}	18:58:43.013213	18:58:44.048	1.034787
{'arah': 'B', 'kecepatan': 200}	18:58:44.514015	18:58:45.551	1.036985
{'arah': 'E', 'kecepatan': 159}	18:58:46.014630	18:58:47.027	1.01237
{'arah': 'D', 'kecepatan': 39}	18:58:47.515394	18:58:48.522	1.006606
{'arah': 'C', 'kecepatan': 200}	18:58:49.015871	18:58:50.056	1.040129
{'arah': 'B', 'kecepatan': 179}	18:58:50.516719	18:58:51.543	1.026281
{'arah': 'E', 'kecepatan': 55}	18:58:52.017064	18:58:53.062	1.044936
{'arah': 'E', 'kecepatan': 228}	18:58:53.518106	18:58:54.544	1.025894

{’arah’: ’C’, ’kecepatan’: 38}	18:58:55.018428	18:58:56.059	1.040572
{’arah’: ’D’, ’kecepatan’: 38}	18:58:56.518869	18:58:57.560	1.041131
{’arah’: ’C’, ’kecepatan’: 192}	18:58:58.019668	18:58:59.065	1.045332
{’arah’: ’E’, ’kecepatan’: 56}	18:58:59.520195	18:59:00.557	1.036805
Average Delay Time			1.037564129

Tabel 4.7 menampilkan pengiriman JSON yang terdiri dari 2 *key-value*, yaitu arah dan kecepatan. Dalam pengujian kali ini, data dikirimkan sebanyak 31 kali secara berturut-turut dengan penambahan waktu *delay* sebesar 1.5 detik setiap kali mengirimkan data. Hal ini dilakukan agar ESP32 dapat menerima data dengan baik dan berhasil memisahkan (*deserialize*) serta memasukkan kedua nilai tersebut sesuai dengan variabel yang telah ditentukan. Hasil dari pengujian ini menunjukkan bahwa waktu pengiriman rata-rata dari laptop menuju ESP32 melalui Bluetooth adalah sebesar 1,037564129 detik.

Tabel 4.8: Pengujian Waktu *Delay* Pengiriman Data JSON Berisi 1 Nilai Melalui *Access Point* WiFi

Data	Timestamp Sent	Timestamp Received	Delay Time
{’arah’: ’C’}	22:27:11.678759	22:27:12.710	1.031241
{’arah’: ’B’}	22:27:13.184177	22:27:14.233	1.048823
{’arah’: ’C’}	22:27:14.698806	22:27:15.748	1.049194
{’arah’: ’A’}	22:27:16.203611	22:27:17.215	1.011389
{’arah’: ’C’}	22:27:17.715604	22:27:18.746	1.030396
{’arah’: ’A’}	22:27:19.222435	22:27:20.228	1.005565
{’arah’: ’C’}	22:27:20.730434	22:27:21.771	1.040566
{’arah’: ’D’}	22:27:22.239679	22:27:23.285	1.045321
{’arah’: ’C’}	22:27:23.750695	22:27:24.788	1.037305
{’arah’: ’E’}	22:27:25.265167	22:27:26.301	1.035833
{’arah’: ’C’}	22:27:26.769275	22:27:27.804	1.034725
{’arah’: ’C’}	22:27:28.276680	22:27:29.317	1.04032
{’arah’: ’D’}	22:27:29.789932	22:27:30.831	1.041068
{’arah’: ’B’}	22:27:31.342439	22:27:32.400	1.057561
{’arah’: ’B’}	22:27:32.848560	22:27:33.882	1.03344
{’arah’: ’A’}	22:27:34.369849	22:27:35.396	1.026151
{’arah’: ’C’}	22:27:35.877856	22:27:36.924	1.046144
{’arah’: ’B’}	22:27:37.525218	22:27:38.531	1.005782
{’arah’: ’A’}	22:27:39.043576	22:27:40.095	1.051424
{’arah’: ’E’}	22:27:40.637793	22:27:41.671	1.033207
{’arah’: ’B’}	22:27:42.143229	22:27:43.168	1.024771
{’arah’: ’E’}	22:27:43.677645	22:27:44.714	1.036355
{’arah’: ’D’}	22:27:45.218328	22:27:46.257	1.038672
{’arah’: ’C’}	22:27:46.723402	22:27:47.758	1.034598
{’arah’: ’B’}	22:27:48.296809	22:27:49.347	1.050191
{’arah’: ’D’}	22:27:49.898938	22:27:50.940	1.041062
{’arah’: ’B’}	22:27:51.405449	22:27:52.453	1.047551

{'arah': 'D'}	22:27:52.914479	22:27:53.965	1.050521
{'arah': 'E'}	22:27:54.458062	22:27:55.497	1.038938
{'arah': 'E'}	22:27:55.966855	22:27:57.010	1.043145
{'arah': 'E'}	22:27:57.472623	22:27:58.518	1.045377
{'arah': 'D'}	22:28:01.988199	22:28:03.032	1.043801
{'arah': 'C'}	22:28:03.502648	22:28:04.548	1.045352
{'arah': 'B'}	22:28:05.007693	22:28:06.048	1.040307
{'arah': 'C'}	22:28:07.635244	22:28:08.708	1.072756
Average Delay Time			1.038824343

Tabel 4.8 menampilkan pengiriman JSON yang terdiri dari 1 *key-value*, yaitu arah. Dalam pengujian kali ini, data dikirimkan sebanyak 35 kali secara berturut-turut dengan penambahan waktu *delay* sebesar 1.5 detik setiap kali mengirimkan data. Hal ini dilakukan agar ESP32 dapat menerima data dengan baik dan berhasil memisahkan (*deserialize*) serta memasukkan nilai tersebut sesuai dengan variabel yang telah ditentukan. Hasil dari pengujian ini menunjukkan waktu pengiriman rata-rata dari laptop menuju ESP32 melalui Bluetooth sebesar 1.038824343 detik.

Dari kedua cara pengujian tersebut, didapatkan bahwa waktu *delay* antara proses pengiriman JSON yang mengandung 2 *key-value* dibandingkan dengan JSON yang hanya berisikan 1 *key-value* tidak terlalu signifikan. Saat mengirimkan JSON yang berisikan 2 *key-value*, pengujian menunjukkan adanya *delay* sebesar 1,037564129 detik. Jika dibandingkan dengan pengujian yang mengirimkan JSON dengan 1 *key-value*, waktu *delay* yang tercatat adalah sebesar 1.038824343 detik. Terdapat anomali pada pengujian ini, karena waktu delay pada saat mengirimkan data JSON yang berisikan 2 *key-value* lebih kecil jika dibandingkan dengan saat mengirimkan data JSON yang berisikan 1 *key-value* walaupun perbedaan waktu *delay*-nya tidak terlalu signifikan.

4.4.1 Pengujian Kestabilan Motor Kursi Roda

Pengujian kestabilan motor dilakukan untuk mendeteksi derajat penyimpangan dari setiap motor. Pada pengujian ini akan dilakukan dengan cara menggerakkan kursi roda kearah depan (maju) dan juga kearah belakang (mundur). Perbedaan antara posisi akhir roda dengan titik pusat akan diukur untuk menentukan sudut penyimpangan seperti pada Gambar 5.1. Sudut Penyimpangan dapat dihitung menggunakan Persamaan 4.7. Hasil dari pengujian ini dapat dilihat pada Tabel 4.9 dan Tabel 4.10.

$$SudutPenyimpangan = \arctan\left(\frac{JarakPenyimpangan}{JarakTempuh}\right) \tag{4.7}$$

Keterangan
Sudut Penyimpangan : Besar sudut penyimpangan gerak motor kursi roda (°)
Jarak Penyimpangan : Jarak penyimpangan posisi akhir roda yang diukur dari posisi garis lurus jarak tempuh (cm)
Jarak Tempuh : Lintasan garis lurus yang harus ditempuh oleh kursi roda (m)

Tabel 4.9: Pengujian Kestabilan Motor Kursi Roda Dengan Gerak Maju

Jarak Tempuh	Jarak Penyimpangan	Sudut Penyimpangan	Arah Penyimpangan
10,34	47	2,60°	Kiri
10,50	49	2,67°	Kiri
10,74	53	2,83°	Kiri
10,53	51	2,77°	Kiri
10,62	57	3,07°	Kiri
Sudut Penyimpangan Rata - Rata			2,788°

Tabel 4.9 menampilkan hasil pengujian kestabilan motor kursi roda dengan gerak maju. Pengujian ini dilakukan sebanyak 5 kali dengan jarak tempuh bervariasi dan lebih dari 10 meter. Hasil dari pengujian ini menunjukkan bahwa kursi roda cenderung bergerak kearah kiri dengan rata-rata sudut penyimpangan sebesar 2,788°.

Tabel 4.10: Pengujian Kestabilan Motor Kursi Roda Dengan Gerak Mundur

Jarak Tempuh	Jarak Penyimpangan	Sudut Penyimpangan	Arah Penyimpangan
10,17	102	5,73°	Kanan
10,33	96	5.31°	Kanan
10,24	91	5,08°	Kanan
10,67	107	5,73°	Kanan
10,46	103	5,62°	Kanan
Sudut Penyimpangan Rata - Rata			5,494°

Tabel 4.10 menampilkan hasil pengujian kestabilan motor kursi roda dengan gerak mundur. Pengujian ini dilakukan sebanyak 5 kali dengan jarak tempuh bervariasi dan lebih dari 10 meter. Hasil dari pengujian ini menunjukkan bahwa kursi roda cenderung bergerak kearah kanan dengan rata-rata sudut penyimpangan sebesar 5,494°.

[Halaman ini sengaja dikosongkan]

BAB V
PENUTUP

Pada bab ini akan dipaparkan kesimpulan dari hasil pengujian yang akan menjawab dari permasalahan yang diangkat dari pelaksanaan tugas akhir ini. Pada bab ini juga diapaprkan saran mengenai hal yang dapat dilakukan untuk mengembangkan penelitian kedepannya.

5.1 Kesimpulan

Berdasarkan hasil pengujian yang dilakukan selama pelaksanaan tugas akhir ini adalah sebagai berikut:

1. Waktu *delay* rata-rata terendah terdapat pada pengujian dengan mengirimkan string dengan 1 nilai dan ditransmisikan melalui WiFi. Waktu *delay* rata-ratanya adalah sebesar 0,03499708571 detik.
2. Waktu *delay* rata-rata tertinggi terdapat pada pengujian dengan mengirimkan string dengan 2 nilai dan ditransmisikan melalui WiFi. Waktu *delay* rata-ratanya adalah sebesar 1,059681387 detik.
3. Waktu *delay* tambahan pada program pengirim data diperlukan untuk beberapa pengujian. Hal ini diakibatkan agar tidak terjadi penumpukan (*flooding*) data yang dapat mengganggu kinerja dari ESP32.
4. Terdapat beberapa pengujian yang memerlukan waktu *delay* tambahan pada program pengirim data seperti pada pengiriman data String berisi 2 nilai melalui Bluetooth yang memerlukan waktu *delay* tambahan sebesar 1,5 detik, pengiriman data JSON dengan 2 nilai melalui Bluetooth yang memerlukan waktu *delay* tambahan sebesar 1,1 detik, pengiriman data String berisi 2 nilai melalui WiFi yang memerlukan waktu *delay* tambahan sebesar 1,3 detik, pengiriman data JSON yang berisi 2 nilai melalui WiFi yang memerlukan waktu *delay* tambahan sebesar 1,5 detik, pengiriman data JSON yang berisi 1 nilai melalui Bluetooth yang memerlukan waktu *delay* tambahan sebesar 1 detik, pengiriman data JSON yang berisi 1 nilai melalui WiFi yang memerlukan waktu *delay* tambahan sebesar 1,5 detik.
5. Hanya 2 pengujian yang tidak memerlukan waktu *delay* tambahan pada program pengirim data, yaitu pada pengiriman data String yang berisi 1 nilai melalui Bluetooth dan pengiriman data String yang berisi 1 nilai melalui WiFi.
6. Data yang ditransmisikan untuk mengontrol kursi roda melalui visi komputer lebih baik menggunakan String yang hanya berisi 1 nilai, yaitu variabel arah yang dianalogikan dengan 1 karakter huruf.
7. Transmisi data yang digunakan untuk mengontrol kursi roda melalui visi komputer lebih baik menggunakan WiFi dengan ESP32 sebagai *Access Point*-nya.

5.2 Saran

Berdasarkan hasil yang diperoleh dari penelitian ini maka saran yang dapat dipertimbangkan untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Kecepatan putar motor diatur melalui ESP32 dengan menggunakan *button* ataupun potensiometer.
2. Mencoba menggunakan gestur lain untuk mengontrol gerak kursi roda.

DAFTAR PUSTAKA

Abrahamsen, H. (2023, October). Bluetoothserial. https://www.arduino.cc/reference/en/libraries/bluetoothserial/

Blanchon, B. (2021). *Mastering arduinojson 6: Efficient json serialization for embedded c++*. Benoit Blanchon. https://books.google.co.id/books?id=zuCgzwEACAAJ

Choi, J. H., Chung, Y., & Oh, S. (2019). Motion control of joystick interfaced electric wheelchair for improvement of safety and riding comfort. *Mechatronics*, *59*, 104–114.

Daring, K. (2016). Kbbi vi daring. https://kbbi.kemdikbud.go.id/entri/lumpuh

Data, R. (n.d.). What is json? https://www.w3schools.com/whatis/whatis_json.asp

Developer, N. (2023). https://developer.nvidia.com/embedded/jetson-nano-developer-kit

Gao, P., Luo, H., & Li, Y. (2023). Vision-based head posture control wheelchair system research. *2023 2nd International Symposium on Sensor Technology and Control (ISSTC)*, 232–237. https://doi.org/10.1109/ISSTC59603.2023.10280784

Hirzel, T. (2024, January). Basics of pwm (pulse width modulation). https://docs.arduino.cc/learn/microcontrollers/analog-output/

Junior, A. S., & Arifin, F. (2019). Prototipe kursi roda elektrik dengan kendali joystick dan smartphone. *Elinvo (Electronics, Informatics, and Vocational Education)*, *4*(1), 62–68.

Muhammad. (2018, December). Analisis driver h-bridge menggunakan relay pada motor bldc konstruksi axial flux (celah ganda). https://repository.unej.ac.id/handle/123456789/89217

Org, J. (2017). Introducing json. https://www.json.org/json-en.html

Pansawira, P. (2022, April). Kelumpuhan - gejala, penyebab, dan mengobati - alodokter. https://www.alodokter.com/kelumpuhan#:~:text=Kondisi%20ini%20dapat%20disebabkan%20oleh,Penanganan%20kelumpuhan%20tergantung%20pada%20penyebabnya.

Prasetyo, Y. E., Hindarto, H., Syahrorini, S., & Wisaksono, A. (2023). Wheelchair control using bluetooth-based electromyography signals. *Journal of Computer Networks, Architecture and High Performance Computing*, *5*(1), 148–159.

Sairam, K., Gunasekaran, N., & Redd, S. R. (2002). Bluetooth in wireless communication. *IEEE Communications Magazine*, *40*(6), 90–96.

SIG, B. (2024). Bluetooth technology overview. https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

SmartArduino, Y. (2022). Esp8266 esp32. https://github.com/SmartArduino/SZDOITWiKi/wiki/ESP8266---ESP32

Söderby, K. (2023). Using the serial monitor tool. https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-monitor

Söderby, K., & Hylén, J. (2023a). Getting started with arduino ide 2. https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2

Söderby, K., & Hylén, J. (2023b). Installing a board package in the ide 2. https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-board-manager

Söderby, K., & Hylén, J. (2023c). Installing libraries. https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-installing-a-library

Systems, E. (2023, December). https://docs.espressif.com/projects/esp-idf/en/latest/esp32/esp-idf-en-master-esp32.pdf

Tian, H., Wang, T., Liu, Y., Qiao, X., & Li, Y. (2020). Computer vision technology in agricultural automation —a review. *Information Processing in Agriculture*, *7*(1), 1–19. https://doi.org/https://doi.org/10.1016/j.inpa.2019.09.006

Wicaksono, B. A. (2023). *Rancang bangun kursi roda elektrik dengan sistem kontrol joystick dan smartphone android* [Doctoral dissertation, Universitas Diponegoro].

Wisaksono, A., Pratama, R. A., et al. (2023). Kontrol kursi roda menggunakan sinyal suara melelui bluetooth. *Prosiding Seminar Nasional Penelitian dan Pengabdian Kepada Masyarakat*, *1*(1), 26–30.

LAMPIRAN

Kode Program

Program Untuk Memeriksa MAC Address Pada ESP32

Program 5.1: Program Untuk Memeriksa MAC Address Pada ESP32

```
#include <BluetoothSerial.h>
BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32_Haris"); // Bluetooth device name
  delay(1000);  // Give some time for the Bluetooth module to initialize
}

void loop() {
  // Get the ESP32 Bluetooth address
  uint8_t esp32Address[6];
  esp_efuse_mac_get_default(esp32Address);

  // Print the address in a readable format
  Serial.printf("ESP32 Bluetooth Address: %02X:%02X:%02X:%02X:%02X:%02X\n",
                esp32Address[0], esp32Address[1], esp32Address[2],
                esp32Address[3], esp32Address[4], esp32Address[5]);
}
```

Program Untuk Menerima Data String Melalui Bluetooth Pada ESP32

Program 5.2: Program Untuk Menerima Data String Melalui Bluetooth Pada ESP32

```
#include <BluetoothSerial.h>
BluetoothSerial SerialBT;

int maxspeed;

void setup(){
  Serial.begin(115200);
  SerialBT.begin("ESP32_Haris");
}

```

```
11 void loop(){
12   if (SerialBT.available()){
13     String receivedData = SerialBT.readStringUntil('\n');
14
15     String arah = receivedData.substring(0, receivedData.↩
         indexOf(','));
16     String kecepatan = receivedData.substring(receivedData.↩
         indexOf(',') + 1);
17
18     maxspeed = kecepatan.toInt();
19
20     Serial.print("Arah : ");
21     Serial.println(arah);
22     Serial.print("Kecepatan : ");
23     Serial.println(kecepatan);
24     SerialBT.println("1");
25
26     if(maxspeed < 20){
27       Serial.println("TRUE");
28     }
29     else{
30       Serial.println("FALSE");
31     }
32   }
33 }
```

Program Untuk Menerima Data JSON Melalui Bluetooth Pada ESP32

Program 5.3: Program Untuk Menerima Data JSON Melalui Bluetooth Pada ESP32

```
1 #include <BluetoothSerial.h>
2 #include <ArduinoJson.h>
3
4 BluetoothSerial SerialBT;
5
6 void setup() {
7   Serial.begin(115200);
8   SerialBT.begin("ESP32_Haris");
9 }
10
11 void loop() {
12   if (SerialBT.available()) {
13     // Read the incoming data
14     String json_data = SerialBT.readStringUntil('\n');
15
16     // Deserialize the JSON
17     DynamicJsonDocument doc(1024);
```

```
18      DeserializationError error = deserializeJson(doc, ↩
           json_data);
19
20      if (error) {
21        Serial.println("Failed to parse JSON");
22      } else {
23        // Extract values from JSON
24        const char* arah = doc["arah"];
25        const char* kecepatan = doc["kecepatan"];
26
27        // Perform actions based on the received data
28        Serial.print("Received Arah: ");
29        Serial.println(arah);
30        Serial.print("Received Kecepatan: ");
31        Serial.println(kecepatan);
32      }
33    }
34 }
```

Program Untuk Menerima Data String Melalui *Access Point* WiFi Pada ESP32

Program 5.4: Program Untuk Menerima Data String Melalui *Access Point* WiFi Pada ESP32

```
1  #include <WiFi.h>
2  #include <Arduino.h>
3
4  int maxspeed;
5
6  const char* ssid = "Haris-Access-Point";
7  const char* password = "123456789";
8
9  WiFiServer server(80);
10
11 void setup(){
12   Serial.begin(115200);
13
14
15   // Connect to Wi-Fi network with SSID and password
16   Serial.print("Setting AP Access Point");
17   // Remove the password parameter, if you want the AP (↩
        Access Point) to be open
18   WiFi.softAP(ssid, password);
19
20   IPAddress IP = WiFi.softAPIP();
21   Serial.print("ESP32 AP IP Address : ");
22   Serial.println(IP);
23
```

```
24   // Start the server
25   server.begin();
26 }
27
28 void loop(){
29     // Check if a client has connected
30   WiFiClient client = server.available();
31
32   if (client) {
33     //Serial.println("New client connected");
34
35     // Read the data from the client
36     while (client.connected()) {
37       if (client.available()) {
38
39         String receivedData = client.readStringUntil('\n');
40
41         String arah = receivedData.substring(0, receivedData.↩
              indexOf(','));
42         String kecepatan = receivedData.substring(↩
              receivedData.indexOf(',') + 1);
43
44         maxspeed = kecepatan.toInt();
45
46         Serial.print("Arah : ");
47         Serial.println(arah);
48         Serial.print("Kecepatan : ");
49         Serial.println(kecepatan);
50
51         if(maxspeed < 20){
52           Serial.println("TRUE");
53         }
54         else{
55           Serial.println("FALSE");
56         }
57
58       }
59     }
60
61     client.stop();
62
63   }
64 }
```

Program Untuk Menerima Data JSON Melalui *Access Point* WiFi Pada ESP32

Program 5.5: Program Untuk Menerima Data JSON Melalui Access Point WiFi Pada ESP32

```
1 #include <WiFi.h>
2 #include <Arduino.h>
3 #include <ArduinoJson.h>
4
5 int maxspeed;
6
7 const char* ssid = "B300-Access-Point";
8 const char* password = "123456789";
9
10 WiFiServer server(80);
11
12 void setup(){
13   Serial.begin(115200);
14
15
16   // Connect to Wi-Fi network with SSID and password
17   Serial.print("Setting AP (Access Point)");
18   // Remove the password parameter, if you want the AP (↩
       Access Point) to be open
19   WiFi.softAP(ssid, password);
20
21   IPAddress IP = WiFi.softAPIP();
22   Serial.print("ESP32 AP IP Address : ");
23   Serial.println(IP);
24
25   // Start the server
26   server.begin();
27 }
28
29 void loop(){
30     // Check if a client has connected
31   WiFiClient client = server.available();
32
33   if (client) {
34     //Serial.println("New client connected");
35
36     // Read the data from the client
37     while (client.connected()) {
38       if (client.available()) {
39
40         String json_data = client.readStringUntil('\n');
41
42         DynamicJsonDocument doc(1024);
43         DeserializationError error = deserializeJson(doc, ↩
             json_data);
44
```

```
45          if(error){
46            Serial.println("Failed to parse JSON");
47          }
48          else{
49            const char* arah = doc["arah"];
50            int kecepatan = doc["kecepatan"];
51
52            Serial.print("Received Arah : ");
53            Serial.print(arah);
54            Serial.print("   Received Kecepatan : ");
55            Serial.println(kecepatan);
56          }
57        }
58      }
59
60      client.stop();
61
62    }
63 }
```

Program Untuk Menguji Rangkaian Kontrol ESP32

Program 5.6: Program Untuk Menguji Rangkaian Kontrol Pada ESP32

```
1  #include <Arduino.h>
2
3  //Motor Kiri
4  #define pwmpin1 5
5  #define dir1  18
6  #define dir2  19
7
8  //Motor kanan
9  #define pwmpin2 25
10 #define dir3 32
11 #define dir4 33
12
13 #define pwmChannel1  0
14 #define pwmChannel2  1
15 #define freq  15000
16 #define res  8
17
18 int PWM1_DutyCycle = 0;
19 int maxspeed = 127;
20
21
22 void setup() {
23   // put your setup code here, to run once:
24
```

```
25   pinMode(dir1,OUTPUT);
26   pinMode(dir2,OUTPUT);
27   pinMode(dir3,OUTPUT);
28   pinMode(dir4,OUTPUT);
29   //pinMode(pwmpin,OUTPUT);
30
31   ledcSetup(pwmChannel1, freq, res);
32   ledcSetup(pwmChannel2, freq, res);
33
34   ledcAttachPin(pwmpin1, pwmChannel1);
35   ledcAttachPin(pwmpin2, pwmChannel2);
36
37   //ledcWrite(pwmChannel, 127);
38 }
39
40 void loop() {
41   // put your main code here, to run repeatedly:
42
43 //MAJU
44   //SOFTSTART
45   while(PWM1_DutyCycle < maxspeed)
46   {
47     digitalWrite(dir1, HIGH);
48     digitalWrite(dir2, LOW);
49     digitalWrite(dir3, HIGH);
50     digitalWrite(dir4, LOW);
51     ledcWrite(pwmChannel1, PWM1_DutyCycle++);
52     ledcWrite(pwmChannel2, PWM1_DutyCycle++);
53     delay(10);
54   }
55   delay(5000);
56
57   //SOFTSTOP
58   while(PWM1_DutyCycle > 0)
59   {
60     digitalWrite(dir1, HIGH);
61     digitalWrite(dir2, LOW);
62     digitalWrite(dir3, HIGH);
63     digitalWrite(dir4, LOW);
64     ledcWrite(pwmChannel1, PWM1_DutyCycle--);
65     ledcWrite(pwmChannel2, PWM1_DutyCycle--);
66     delay(10);
67
68   }
69   delay(1000);
70
71 //MUNDUR
```

```
72   while(PWM1_DutyCycle < maxspeed)
73   {
74     digitalWrite(dir1, LOW);
75     digitalWrite(dir2, HIGH);
76     digitalWrite(dir3, LOW);
77     digitalWrite(dir4, HIGH);
78     ledcWrite(pwmChannel1, PWM1_DutyCycle++);
79     ledcWrite(pwmChannel2, PWM1_DutyCycle++);
80     delay(10);
81   }
82   delay(5000);
83
84   //SOFTSTOP
85   while(PWM1_DutyCycle > 0)
86   {
87     digitalWrite(dir1, LOW);
88     digitalWrite(dir2, HIGH);
89     digitalWrite(dir3, LOW);
90     digitalWrite(dir4, HIGH);
91     ledcWrite(pwmChannel1, PWM1_DutyCycle--);
92     ledcWrite(pwmChannel2, PWM1_DutyCycle--);
93     delay(10);
94
95   }
96   delay(1000);
97
98 //BELOK KANAN
99  while(PWM1_DutyCycle < maxspeed)
100  {
101    digitalWrite(dir1, HIGH);
102    digitalWrite(dir2, LOW);
103    digitalWrite(dir3, LOW);
104    digitalWrite(dir4, LOW);
105    ledcWrite(pwmChannel1, PWM1_DutyCycle++);
106    ledcWrite(pwmChannel2, PWM1_DutyCycle++);
107    delay(10);
108  }
109  delay(5000);
110
111  //SOFTSTOP
112  while(PWM1_DutyCycle > 0)
113  {
114    digitalWrite(dir1, HIGH);
115    digitalWrite(dir2, LOW);
116    digitalWrite(dir3, LOW);
117    digitalWrite(dir4, LOW);
118    ledcWrite(pwmChannel1, PWM1_DutyCycle--);
```

```
119     ledcWrite(pwmChannel2, PWM1_DutyCycle--);
120     delay(10);
121 
122   }
123   delay(1000);
124 
125 //BELOK KIRI
126  while(PWM1_DutyCycle < maxspeed)
127   {
128     digitalWrite(dir1, LOW);
129     digitalWrite(dir2, LOW);
130     digitalWrite(dir3, HIGH);
131     digitalWrite(dir4, LOW);
132     ledcWrite(pwmChannel1, PWM1_DutyCycle++);
133     ledcWrite(pwmChannel2, PWM1_DutyCycle++);
134     delay(10);
135   }
136   delay(5000);
137 
138   //SOFTSTOP
139   while(PWM1_DutyCycle > 0)
140   {
141     digitalWrite(dir1, LOW);
142     digitalWrite(dir2, LOW);
143     digitalWrite(dir3, HIGH);
144     digitalWrite(dir4, LOW);
145     ledcWrite(pwmChannel1, PWM1_DutyCycle--);
146     ledcWrite(pwmChannel2, PWM1_DutyCycle--);
147     delay(10);
148 
149   }
150   delay(1000);
151 
152 //STOP
153  while(PWM1_DutyCycle < maxspeed)
154   {
155     digitalWrite(dir1, LOW);
156     digitalWrite(dir2, LOW);
157     digitalWrite(dir3, LOW);
158     digitalWrite(dir4, LOW);
159     ledcWrite(pwmChannel1, PWM1_DutyCycle++);
160     ledcWrite(pwmChannel2, PWM1_DutyCycle++);
161     delay(10);
162   }
163   delay(5000);
164 
165   //SOFTSTOP
```

```
166   while(PWM1_DutyCycle > 0)
167   {
168     digitalWrite(dir1, LOW);
169     digitalWrite(dir2, LOW);
170     digitalWrite(dir3, LOW);
171     digitalWrite(dir4, LOW);
172     ledcWrite(pwmChannel1, PWM1_DutyCycle--);
173     ledcWrite(pwmChannel2, PWM1_DutyCycle--);
174     delay(10);
175
176   }
177   delay(1000);
178 }
```

Program Kontrol Motor Kursi Roda Melalui Bluetooth

Program 5.7: Program Kontrol Motor Kursi Roda Melalui Bluetooth

```
1   #include <BluetoothSerial.h>
2   #include <Arduino.h>
3
4   //Motor Kiri
5   #define pwmpin1 5
6   #define dir1 18
7   #define dir2 19
8
9   //Motor kanan
10  #define pwmpin2 25
11  #define dir3 32
12  #define dir4 33
13
14  //STATE Motor
15  int stdir[4];
16
17  #define pwmChannel1 0
18  #define pwmChannel2 1
19  #define freq 15000
20  #define res 8
21
22  int PWM1_DutyCycle = 0;
23  int maxspeed = 0;
24  int turnspeed = 0;
25
26  BluetoothSerial SerialBT;
27
28  void setup() {
29    Serial.begin(115200);
30    SerialBT.begin("ESP32_Haris");
```

```
31
32      pinMode(dir1, OUTPUT);
33      pinMode(dir2, OUTPUT);
34      pinMode(dir3, OUTPUT);
35      pinMode(dir4, OUTPUT);
36
37      ledcSetup(pwmChannel1, freq, res);
38      ledcSetup(pwmChannel2, freq, res);
39
40      ledcAttachPin(pwmpin1, pwmChannel1);
41      ledcAttachPin(pwmpin2, pwmChannel2);
42    }
43
44    void loop() {
45      if (SerialBT.available()) {
46        String receivedData = SerialBT.readStringUntil('\n');
47
48        String arah = receivedData.substring(0, receivedData.↩
            indexOf(','));
49        String kecepatan = receivedData.substring(receivedData.↩
            indexOf(',') + 1);
50
51        maxspeed = kecepatan.toInt();
52        turnspeed = maxspeed / 2;
53
54        Serial.print("Arah : ");
55        Serial.println(arah);
56        Serial.print("Kecepatan : ");
57        Serial.println(kecepatan);
58
59        if (arah == "A") {
60          while (PWM1_DutyCycle <= turnspeed) {
61            stdir[0] = LOW;
62            stdir[1] = LOW;
63            stdir[2] = HIGH;
64            stdir[3] = LOW;
65
66            digitalWrite(dir1, stdir[0]);
67            digitalWrite(dir2, stdir[1]);
68            digitalWrite(dir3, stdir[2]);
69            digitalWrite(dir4, stdir[3]);
70            ledcWrite(pwmChannel1, PWM1_DutyCycle++);
71            ledcWrite(pwmChannel2, PWM1_DutyCycle++);
72            delay(10);
73          }
74
75        } else if (arah == "B") {
```

```
76          while (PWM1_DutyCycle <= maxspeed) {
77            stdir[0] = HIGH;
78            stdir[1] = LOW;
79            stdir[2] = HIGH;
80            stdir[3] = LOW;
81
82            digitalWrite(dir1, stdir[0]);
83            digitalWrite(dir2, stdir[1]);
84            digitalWrite(dir3, stdir[2]);
85            digitalWrite(dir4, stdir[3]);
86            ledcWrite(pwmChannel1, PWM1_DutyCycle++);
87            ledcWrite(pwmChannel2, PWM1_DutyCycle++);
88            delay(10);
89          }
90
91        } else if (arah == "C") {
92          while (PWM1_DutyCycle >= 0) {
93
94            digitalWrite(dir1, stdir[0]);
95            digitalWrite(dir2, stdir[1]);
96            digitalWrite(dir3, stdir[2]);
97            digitalWrite(dir4, stdir[3]);
98            ledcWrite(pwmChannel1, PWM1_DutyCycle--);
99            ledcWrite(pwmChannel2, PWM1_DutyCycle--);
100           delay(10);
101         }
102       } else if (arah == "D") {
103         while (PWM1_DutyCycle <= turnspeed) {
104           stdir[0] = LOW;
105           stdir[1] = HIGH;
106           stdir[2] = LOW;
107           stdir[3] = HIGH;
108
109           digitalWrite(dir1, stdir[0]);
110           digitalWrite(dir2, stdir[1]);
111           digitalWrite(dir3, stdir[2]);
112           digitalWrite(dir4, stdir[3]);
113           ledcWrite(pwmChannel1, PWM1_DutyCycle++);
114           ledcWrite(pwmChannel2, PWM1_DutyCycle++);
115           delay(10);
116         }
117
118       } else if (arah == "E") {
119         while (PWM1_DutyCycle <= turnspeed) {
120           stdir[0] = HIGH;
121           stdir[1] = LOW;
122           stdir[2] = LOW;
```

```
123             stdir[3] = LOW;
124
125             digitalWrite(dir1, stdir[0]);
126             digitalWrite(dir2, stdir[1]);
127             digitalWrite(dir3, stdir[2]);
128             digitalWrite(dir4, stdir[3]);
129             ledcWrite(pwmChannel1, PWM1_DutyCycle++);
130             ledcWrite(pwmChannel2, PWM1_DutyCycle++);
131             delay(10);
132           }
133
134       }
135     }
136   }
```

Program Kontrol Motor Kursi Roda Melalui *Access Point* WiFi

Program 5.8: Program Kontrol Motor Kursi Roda Melalui *Access Point* WiFi

```
1   #include <WiFi.h>
2   #include <Arduino.h>
3
4   //WiFi Configuration
5   const char* ssid = "Haris-Access-Point";
6   const char* password = "123456789";
7
8   WiFiServer server(80);
9
10  //Motor Kiri
11  #define pwmpin1 5
12  #define dir1 18
13  #define dir2 19
14
15  //Motor kanan
16  #define pwmpin2 25
17  #define dir3 32
18  #define dir4 33
19
20  //STATE Motor
21  int stdir[4];
22
23  #define pwmChannel1 0
24  #define pwmChannel2 1
25  #define freq 15000
26  #define res 8
27
28  int PWM1_DutyCycle = 0;
29  int maxspeed = 70;
```

```
int turnspeed = 35;

void setup() {
  Serial.begin(115200);
  Serial.print("Setting AP (Access Point) ...");
  WiFi.softAP(ssid, password);

  IPAddress IP = WiFi.softAPIP();
  Serial.print("ESP32 AP IP Address : ");
  Serial.println(IP);

  server.begin();

  pinMode(dir1, OUTPUT);
  pinMode(dir2, OUTPUT);
  pinMode(dir3, OUTPUT);
  pinMode(dir4, OUTPUT);

  ledcSetup(pwmChannel1, freq, res);
  ledcSetup(pwmChannel2, freq, res);

  ledcAttachPin(pwmpin1, pwmChannel1);
  ledcAttachPin(pwmpin2, pwmChannel2);
}

void loop() {
  WiFiClient client = server.available();
  if(client){
    while(client.connected()){
      if(client.available()){

        String arah = client.readStringUntil('\n');
        Serial.print("Arah : ");
        Serial.println(arah);

        if(arah == "A"){
          while(PWM1_DutyCycle <= turnspeed){
            stdir[0] = LOW;
            stdir[1] = LOW;
            stdir[2] = HIGH;
            stdir[3] = LOW;

            digitalWrite(dir1, stdir[0]);
            digitalWrite(dir2, stdir[1]);
            digitalWrite(dir3, stdir[2]);
            digitalWrite(dir4, stdir[3]);
            ledcWrite(pwmChannel1, PWM1_DutyCycle++);
```

```
77              ledcWrite(pwmChannel2, PWM1_DutyCycle++);
78              delay(10);
79            }
80          }
81          else if(arah == "B"){
82            while(PWM1_DutyCycle <= maxspeed){
83              stdir[0] = HIGH;
84              stdir[1] = LOW;
85              stdir[2] = HIGH;
86              stdir[3] = LOW;
87
88              digitalWrite(dir1, stdir[0]);
89              digitalWrite(dir2, stdir[1]);
90              digitalWrite(dir3, stdir[2]);
91              digitalWrite(dir4, stdir[3]);
92              ledcWrite(pwmChannel1, PWM1_DutyCycle++);
93              ledcWrite(pwmChannel2, PWM1_DutyCycle++);
94              delay(10);
95            }
96          }
97          else if(arah == "C"){
98            while(PWM1_DutyCycle >= 0){
99              digitalWrite(dir1, stdir[0]);
100             digitalWrite(dir2, stdir[1]);
101             digitalWrite(dir3, stdir[2]);
102             digitalWrite(dir4, stdir[3]);
103             ledcWrite(pwmChannel1, PWM1_DutyCycle--);
104             ledcWrite(pwmChannel2, PWM1_DutyCycle--);
105             delay(10);
106           }
107         }
108         else if(arah == "D"){
109           while(PWM1_DutyCycle <= turnspeed){
110             stdir[0] = LOW;
111             stdir[1] = HIGH;
112             stdir[2] = LOW;
113             stdir[3] = HIGH;
114
115             digitalWrite(dir1, stdir[0]);
116             digitalWrite(dir2, stdir[1]);
117             digitalWrite(dir3, stdir[2]);
118             digitalWrite(dir4, stdir[3]);
119             ledcWrite(pwmChannel1, PWM1_DutyCycle++);
120             ledcWrite(pwmChannel2, PWM1_DutyCycle++);
121             delay(10);
122           }
123         }
```

```
124           else if(arah == "E"){
125             while(PWM1_DutyCycle <= turnspeed){
126               stdir[0] = HIGH;
127               stdir[1] = LOW;
128               stdir[2] = LOW;
129               stdir[3] = LOW;
130 
131               digitalWrite(dir1, stdir[0]);
132               digitalWrite(dir2, stdir[1]);
133               digitalWrite(dir3, stdir[2]);
134               digitalWrite(dir4, stdir[3]);
135               ledcWrite(pwmChannel1, PWM1_DutyCycle++);
136               ledcWrite(pwmChannel2, PWM1_DutyCycle++);
137               delay(10);
138             }
139           }
140         }
141       }
142     }
143   }
```

Program Untuk Mengirim Data String Melalui Bluetooth

Program 5.9: Program Untuk Mengirim Data String Melalui Bluetooth

```
1  import bluetooth
2  import random
3  import datetime
4  import time
5 
6  # Cari perangkat Bluetooth
7  nearby_devices = bluetooth.discover_devices()
8 
9  # Ambil alamat MAC dari ESP32 (pastikan sudah dipasangkan ↩
       sebelumnya)
10 esp32_address = "C0:49:EF:E7:BD:EA"  # Ganti dengan alamat ↩
       MAC ESP32 yang sesuai
11 
12 # Lakukan koneksi
13 sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
14 sock.connect((esp32_address, 1))
15 
16 arah = random.choice('ABCDE')
17 kecepatan = random.choice('OPQRSTUVW')
18 pesan = f"{arah},{kecepatan}"
19 sock.send(pesan)
20 
21 date = datetime.datetime.now()
22 print(f"{date} -> {pesan}")
```

```
23
24 i = 0
25
26 # Terima data
27 while True:
28     data = sock.recv(8)
29     flag = data.decode("utf-8")
30     #print("Received:", flag)
31
32     if flag == '1':
33         arah = random.choice('ABCDE')
34         kecepatan = random.choice('OPQRSTUVW')
35         pesan = f"{arah},{kecepatan}"
36         i = i + 1
37
38         if arah == 'q' or i == 10:
39             break
40
41         sock.send(pesan)
42         date = datetime.datetime.now()
43         print(f"{date} -> {pesan}")
44
45 # Tutup koneksi
46 sock.close()
```

Program Untuk Mengirim Data String Melalui WiFi

Program 5.10: Program Untuk Mengirim Data String Melalui WiFi

```
1  import socket
2  import time
3  import datetime
4
5  host = "192.168.4.1" # Set to ESP32 Access Point IP Address
6  port = 80
7
8  # Create a socket connection
9  with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s↩
       :
10     # Connect to the ESP32 server
11     s.connect((host, port))
12
13     while True:
14         # Send two data values
15         arah = input("Enter arah : ")
16         kecepatan = input("Enter kecepatan : ")
17         message = f"{arah},{kecepatan}"
18         date = datetime.datetime.now()
19         print(date)
20
```

```
            if arah == "q" or kecepatan == "q":
                break

            s.send(message.encode('utf-8'))

    s.close()
```

Program Untuk Mengirim Data JSON Melalui Bluetooth

Program 5.11: Program Untuk Mengirim Data JSON Melalui Bluetooth

```
import bluetooth
import json

# ESP32 Bluetooth address
esp32_address = "EC:62:60:9B:E4:92"  # Replace with your ESP32's Bluetooth address

# Create a Bluetooth socket
sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)

# Connect to the ESP32
sock.connect((esp32_address, 1))

while True:
    # Get keyboard input for two values
    arah = input("Masukkan Arah: ")
    kecepatan = input("Masukkan Kecepatan: ")

    # Create a JSON object
    json_data = {"arah": arah, "kecepatan": kecepatan}

    # Serialize the JSON data
    json_string = json.dumps(json_data)

    if arah == "q" or kecepatan == "q":
        break

    # Send the serialized JSON over Bluetooth
    sock.send(json_string)

# Close the Bluetooth socket
sock.close()
```

Program Untuk Mengirim Data JSON Melalui WiFi

Program 5.12: Program Untuk Mengirim Data JSON Melalui WiFi

```
  import socket
  import time
```

```
3  import json
4  import datetime
5
6  host = "192.168.4.1" # Set to ESP32 Access Point IP Address
7  port = 80
8
9  # Create a socket connection
10 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
11     # Connect to the ESP32 server
12     s.connect((host, port))
13
14     while True:
15         arah = input("Masukkan Arah: ")
16         kecepatan = input("Masukkan Kecepatan: ")
17
18         json_data = {"arah": arah, "kecepatan": kecepatan}
19         date = datetime.datetime.now()
20
21         if arah == q or kecepatan == q:
22             break
23         else:
24             json_string = json.dumps(json_data)
25             s.send(json_string.encode('utf-8'))
26             print(f"{date} -> {json_data}")
27
28 s.close()
```

Pengujian Kestabilan Motor Kursi Roda

Gambar 5.1: Dokumentasi Perhitungan Jarak Penyimpangan

BIOGRAFI PENULIS

I Putu Haris Setiadi Ekatama, atau yang biasa dikenal dengan Haris, lahir pada tanggal 21 Maret 2000 di kota Denpasar. Penulis merupakan anak pertama dari tiga bersaudara yang tinggal dan besar di Denpasar, Bali. Setelah lulus dari SMA Negeri 2 Denpasar, penulis kemudian melanjutkan pendidikan ke jenjang strata satu di Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember mulai tahun 2019.

Penulis merupakan orang yang aktif berorganisasi, dan memiliki berbagai *softskill* serta *hardskill*. Hal ini dibuktikan dengan rekam jejak organisasi dan kepanitiaan dari penulis seperti, Wakil Kepala Departemen Pengabdian Masyarakat Tim Pembina Kerohanian Hindu (TPKH-ITS), Wakil Ketua Internal pada TPKH Festival 2021, Staff Generasi Baru Indonesia (GenBI ITS), Staff Himpunan Mahasiswa Teknik Komputer (HIMATEKKOM ITS), hingga menjadi *Product Manager* salah satu kelompok pada Bootcamp MyDigitalAcademy yang diselenggarakan oleh Bank Mandiri.

[Halaman ini sengaja dikosongkan]