



## **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

-

NIP -

**Program Studi Strata 1 (S1) Teknik Komputer**

Departemen Teknik Komputer

Fakultas Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023



## **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

-

NIP -

**Program Studi Strata 1 (S1) Teknik Komputer**

Departemen Teknik Komputer

Fakultas Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023

*[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

-

NIP -

**Undergraduate Study Program of Computer Engineering**

Department of Computer Engineering

Faculty of Faculty of Intelligent Electrical and Informatics Technology

Sepuluh Nopember Institute of Technology

Surabaya

2023

*[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 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.

(Pembimbing I)

NIP: 19680601 199512 1 009

.....  
(Pembimbing II)

NIP: -

.....  
(Penguji I)

NIP: -

.....  
(Penguji II)

NIP: -

.....  
(Penguji III)

NIP: -

Mengetahui,  
Kepala Departemen Teknik Komputer FTEIC - ITS

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

**SURABAYA**  
**Desember, 2023**

*[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 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.

(Advisor I)

NIP: 19680601 199512 1 009

.....

-

(Co-Advisor II)

NIP: -

.....

-.

(Examiner I)

NIP: -

.....

-.

(Examiner II)

NIP: -

.....

-.

(Examiner III)

NIP: -

.....

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

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

**SURABAYA**  
**December, 2023**

*[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, December 2023

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, December 2023

Acknowledged

Advisor

Student

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]*

## **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.  
2. -

Pada penelitian ini kami mengajukan Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Kata Kunci: Roket, *Anti-gravitas*, Energi, Angkasa.

*[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.

2. -

*In this research, we proposed Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.*

*Keywords:* Rocket, Anti-gravity, Energy, Space.

*[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 Prediksi Jumlah Kalori Yang Terbakar Saat Olahraga *Pull-Up* Berbasis CNN Dengan Menggunakan Jetson Nano.

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 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 Arief Kurniawan, S.T., M.T selaku dosen penguji I dan Ibu Dr. Susi Juniastuti, S.T., M.Eng selaku dosen penguji II 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. Farel Jevon, S.T., Paschalis Seto Wicaksono, S.T., Felix Titus Setiawan, S.T., dan I Gusti Komang Agung Wiguna, S.T. yang telah memberikan inspirasi sehingga penelitian ini dilaksanakan
6. I Putu Krisna Erlangga, Muh. Khaeral Azzam, Moh. Iqbal Fatchurozi, Batrisyia Zahrani Ananto, Evandrew Reynald Collin, Dimas Triananda Murti Putra, Ruky Augusta Gautama, Muh. Rezky Firdaus Irwan, Raka Zein Akbar, Priansa Putra Jaya Wardana dan teman - teman lab B300 lainnya yang telah menemani setiap kegiatan penelitian
7. Teman - teman Departemen Teknik Komputer

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, Desember 2023

I Putu Haris Setiadi Ekatama

*[Halaman ini sengaja dikosongkan]*

# **DAFTAR ISI**

<b>ABSTRAK</b>	<b>i</b>
<b>ABSTRACT</b>	<b>iii</b>
<b>KATA PENGANTAR</b>	<b>v</b>
<b>DAFTAR ISI</b>	<b>vii</b>
<b>DAFTAR GAMBAR</b>	<b>xii</b>
<b>DAFTAR TABEL</b>	<b>xiii</b>
<b>Program</b>	<b>xv</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Permasalahan . . . . .	2
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Manfaat . . . . .	2
<b>2 TINJAUAN PUSTAKA</b>	<b>3</b>
2.1 Penelitian Terdahulu . . . . .	3
2.1.1 Kontrol Kursi Roda Menggunakan Sinyal Suara Melalui Bluetooth . . . . .	3
2.1.2 Rancang Bangun Kursi Roda Elektrik Dengan Sistem Kontrol Joystick Dan <i>Smartphone</i> Android . . . . .	3
2.1.3 <i>Wheelchair Control Using Bluetooth-Based Electromyography Signals</i> . . . . .	3
2.1.4 Prototipe Kursi Roda Elektrik Dengan Kendali Joystick dan <i>Smartphone</i> . . . . .	4
2.1.5 Penelitian Terdahulu 5 . . . . .	4
2.2 Teori/Konsep Dasar . . . . .	4
2.2.1 <i>Convolutional Neural Network (CNN)</i> . . . . .	4
2.2.2 NVIDIA® Jetson Nano™ . . . . .	6
<b>3 METODOLOGI</b>	<b>7</b>

3.1	Deskripsi Sistem . . . . .	7
3.1.1	Estimasi Pose . . . . .	8
3.1.2	Klasifikasi Pose . . . . .	9
3.1.3	Paket Data . . . . .	11
3.1.4	Memecahkan Paket Data . . . . .	11
3.1.5	Kontrol Navigasi . . . . .	11
3.2	Implementasi Alat . . . . .	12
3.2.1	<i>Hardware</i> dan <i>Software</i> yang digunakan . . . . .	12
3.2.2	Skematik Alat . . . . .	12
3.3	Kode Program . . . . .	13
3.3.1	Program Untuk Memeriksa MAC Address Pada ESP32 . . . . .	13
3.3.2	Program Untuk Menerima Data String Melalui Bluetooth Pada ESP32 . . . . .	14
3.3.3	Program Untuk Menerima Data JSON Melalui Bluetooth Pada ESP32 . . . . .	15
3.3.4	Program Untuk Menerima Data String Melalui <i>Access Point WiFi</i> Pada ESP32 . . . . .	17
3.3.5	Program Untuk Menerima Data JSON Melalui <i>Access Point WiFi</i> Pada ESP32 . . . . .	19
3.3.6	Program Untuk Menguji Rangkaian Kontrol ESP32 . . . . .	21
3.3.7	Program Kontrol Motor Kursi Roda Melalui Bluetooth . . . . .	26
3.3.8	Program Kontrol Motor Kursi Roda Melalui <i>Access Point WiFi</i> . . . . .	30
3.3.9	Program Untuk Mengirim Data String Melalui Bluetooth . . . . .	34
3.3.10	Program Untuk Mengirim Data String Melalui WiFi . . . . .	35
3.3.11	Program Untuk Mengirim Data JSON Melalui Bluetooth . . . . .	36
3.3.12	Program Untuk Mengirim Data JSON Melalui WiFi . . . . .	37
<b>4</b>	<b>PENGUJIAN DAN ANALISIS</b>	<b>39</b>
4.1	Pengujian Waktu <i>Delay</i> Pengiriman Data String Melalui Bluetooth . . . . .	39
4.2	Pengujian Waktu <i>Delay</i> Pengiriman Data JSON Melalui Bluetooth . . . . .	42
4.3	Pengujian Waktu <i>Delay</i> Pengiriman Data String Melalui <i>Access Point WiFi</i> . . . . .	44
4.4	Pengujian Waktu <i>Delay</i> Pengiriman Data JSON Melalui <i>Access Point WiFi</i> . . . . .	47
<b>5</b>	<b>PENUTUP</b>	<b>51</b>
5.1	Kesimpulan . . . . .	51
5.2	Saran . . . . .	51
<b>DAFTAR PUSTAKA</b>		<b>53</b>



*[Halaman ini sengaja dikosongkan]*

## DAFTAR GAMBAR

2.1	Arsitektur pada <i>Convolutional Neural Network</i> . . . . .	4
2.2	<i>Aktivasi ReLU</i> . . . . .	5
2.3	<i>Max Pooling</i> . . . . .	5
2.4	Proses <i>Flattening</i> . . . . .	6
2.5	Perangkat Jetson Nano . . . . .	6
3.1	Blok Diagram Penelitian . . . . .	7
3.2	Contoh citra yang telah diestimasi pose . . . . .	9

*[Halaman ini sengaja dikosongkan]*

## DAFTAR TABEL

3.1	Tabel titik <i>keypoints</i> yang relevan pada tahap estimasi pose . . . . .	8
3.2	Tabel contoh klasifikasi citra . . . . .	10
3.3	Kode instruksi dari hasil klasifikasi . . . . .	11
4.1	Pengujian Waktu <i>Delay</i> Pengiriman Data String Berisi 2 Nilai Melalui Bluetooth	39
4.2	Pengujian Waktu <i>Delay</i> Pengiriman Data String Berisi 1 Nilai Melalui Bluetooth	40
4.3	Pengujian Waktu <i>Delay</i> Pengiriman Data JSON Berisi 2 Nilai Melalui Bluetooth	42
4.4	Pengujian Waktu <i>Delay</i> Pengiriman Data JSON Berisi 1 Nilai Melalui Bluetooth	43
4.5	Pengujian Waktu <i>Delay</i> Pengiriman Data String Berisi 2 Nilai Melalui <i>Access Point WiFi</i> . . . . .	45
4.6	Pengujian Waktu <i>Delay</i> Pengiriman Data String Berisi 1 Nilai Melalui <i>Access Point WiFi</i> . . . . .	46
4.7	Pengujian Waktu <i>Delay</i> Pengiriman Data JSON Berisi 2 Nilai Melalui <i>Access Point WiFi</i> . . . . .	47
4.8	Pengujian Waktu <i>Delay</i> Pengiriman Data JSON Berisi 1 Nilai Melalui <i>Access Point WiFi</i> . . . . .	48

*[Halaman ini sengaja dikosongkan]*

## **DAFTAR PROGRAM**

3.1	Program Untuk Memeriksa MAC Address Pada ESP32 . . . . .	13
3.2	Program Untuk Menerima Data String Melalui Bluetooth Pada ESP32 . . . . .	14
3.3	Program Untuk Menerima Data JSON Melalui Bluetooth Pada ESP32 . . . . .	15
3.4	Program Untuk Menerima Data String Melalui <i>Access Point WiFi</i> Pada ESP32	17
3.5	Program Untuk Menerima Data JSON Melalui Access Point WiFi Pada ESP32	19
3.6	Program Untuk Menguji Rangkaian Kontrol Pada ESP32 . . . . .	21
3.7	Program Kontrol Motor Kursi Roda Melalui Bluetooth . . . . .	26
3.8	Program Kontrol Motor Kursi Roda Melalui <i>Access Point WiFi</i> . . . . .	30
3.9	Program Untuk Mengirim Data String Melalui Bluetooth . . . . .	34
3.10	Program Untuk Mengirim Data String Melalui WiFi . . . . .	35
3.11	Program Untuk Mengirim Data JSON Melalui Bluetooth . . . . .	36
3.12	Program Untuk Mengirim Data JSON Melalui WiFi . . . . .	37

*[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 (Darling, 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 akan terjadi kelumpuhan, 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 (Pan-sawira, 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 atau Jetson Nano ke ESP32 yang menggunakan WiFi dengan yang menggunakan Bluetooth, serta membandingkan performa FPS dari laptop maupun Jetson Nano dalam menjalankan sistem visi komputer.

## 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 & Hindarto, n.d.).

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.

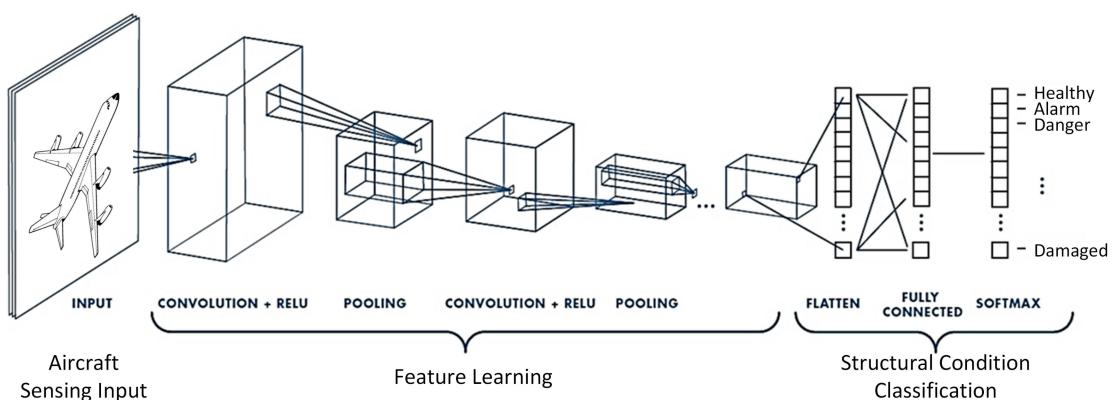
## 2.1.5 Penelitian Terdahulu 5

### 2.2 Teori/Konsep Dasar

#### 2.2.1 Convolutional Neural Network (CNN)

*Convolutional Neural Network* (CNN) telah mencapai hasil yang luar biasa selama beberapa dekade terakhir dalam berbagai bidang yang terkait dengan pengenalan pola, mulai dari pemrosesan gambar hingga pengenalan suara. Aspek paling bermanfaat dari CNN adalah mengurangi jumlah parameter dalam *Artificial Neural Network*. Pencapaian ini telah membantu banyak peneliti dan pengembang dalam mengembangkan model yang lebih besar guna mengetasi tugas-tugas yang kompleks yang tidak mungkin untuk diselesaikan dengan menggunakan *Artificial Neural Network* klasik. Aspek penting dari CNN adalah untuk mendapatkan fitur-fitur abstrak ketika input menyebar menuju lapisan-lapisan yang lebih dalam (Albawi et al., 2017).

Arsitektur pada CNN terdiri dari tiga bagian, yaitu input, *feature learning*, dan *classification*. *Feature Learning* terdiri dari dua buah *convolution layer* dan dua buah *pooling layer*. Pada *classification* terdiri dari dua *hidden layer* dan satu *output layer*. Arsitektur CNN dapat digambarkan seperti pada Gambar 2.1.



Gambar 2.1: Arsitektur pada Convolutional Neural Network

Input CNN merupakan array tiga dimensi dengan ukuran seperti pada Persamaan 2.1. Apabila input merupakan suatu citra maka citra tersebut harus diubah menjadi array dua dimensi.

$$\text{Baris} * \text{Kolom} * \text{Depth}$$

(2.1)

*Convolution Layer* digunakan untuk menyaring (*filter*) matriks dari citra *input*. *Zero Padding* akan diperlukan untuk mempertahankan ukuran matriks dari citra (Dwitama, 2019). Ukuran kernel yang digunakan pada layer konvolusi adalah  $3 \times 3$  dan  $5 \times 5$ . *Output* dari lapisan konvolusi ini akan digunakan sebagai *input* pada *Pooling Layer* (Hakim et al., 2018). Apabila output dari *Convolution Layer* bernilai negatif maka akan dilakukan perhitungan tambahan berupa aktifasi ReLU. Fungsi aktivasi ReLU akan nilai matriks yang bernilai negatif menjadi nol. Contoh penerapan dari aktivasi ReLU dapat dilihat pada Gambar 2.2.

2	1	-3
1	2	1
2	1	-1

2	1	0
1	2	1
2	1	0

Gambar 2.2: Aktivasi ReLU

*Pooling Layer* digunakan untuk mengurangi jumlah parameter ketika ukuran citra terlalu besar dengan cara mengurangi dimensi setiap fitur. Karena ukuran citra menjadi lebih kecil maka proses *feature map* akan menjadi lebih cepat (Hakim et al., 2018). *Max Pooling* dilakukan dengan cara mengambil nilai dengan elemen terbesar sesuai dengan ukuran filter. Sebagai contoh pada Gambar 2.3 merupakan *max pooling* dengan filter  $2 \times 2$  dengan *stride* sebesar 2.

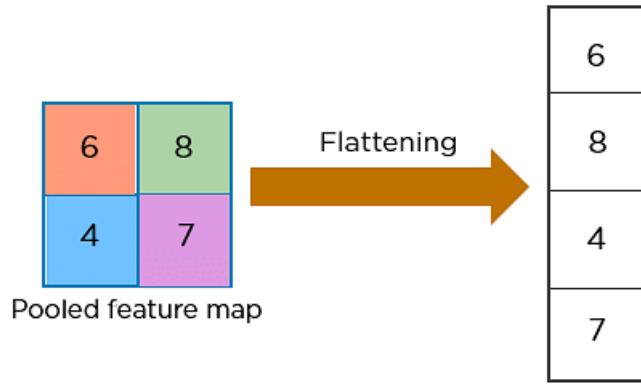
12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

20	30
112	37

Gambar 2.3: Max Pooling

Flatten merupakan suatu proses dimana hasil dari *Feature Learning* diubah menjadi vektor yang selanjutnya akan menjadi input pada proses klasifikasi dengan arsitektur *fully connected layer*. Flatten digunakan untuk mengubah matriks menjadi vektor dengan menyesuaikan sesuai format *input* pada *neural network layer*. Flatten dapat digambarkan seperti pada Gambar 2.4.



Gambar 2.4: Proses *Flattening*

### 2.2.2 NVIDIA® Jetson Nano™



Gambar 2.5: 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 diemas 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.

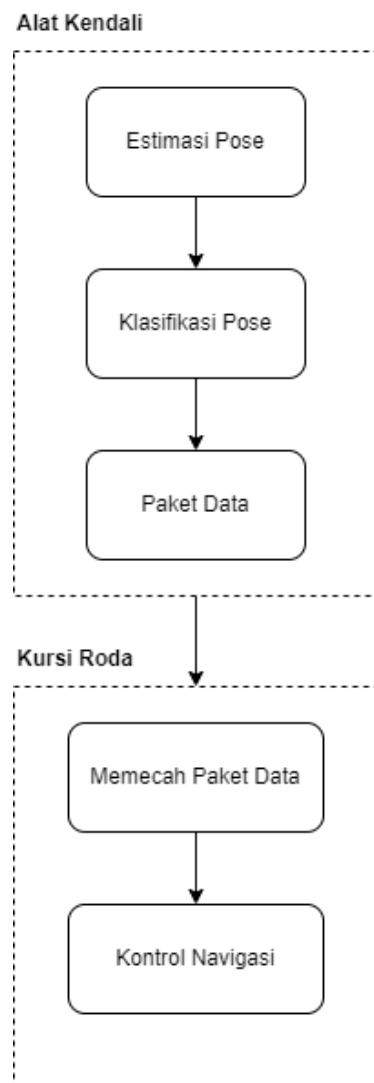
## BAB III

## 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 Estimasi Pose

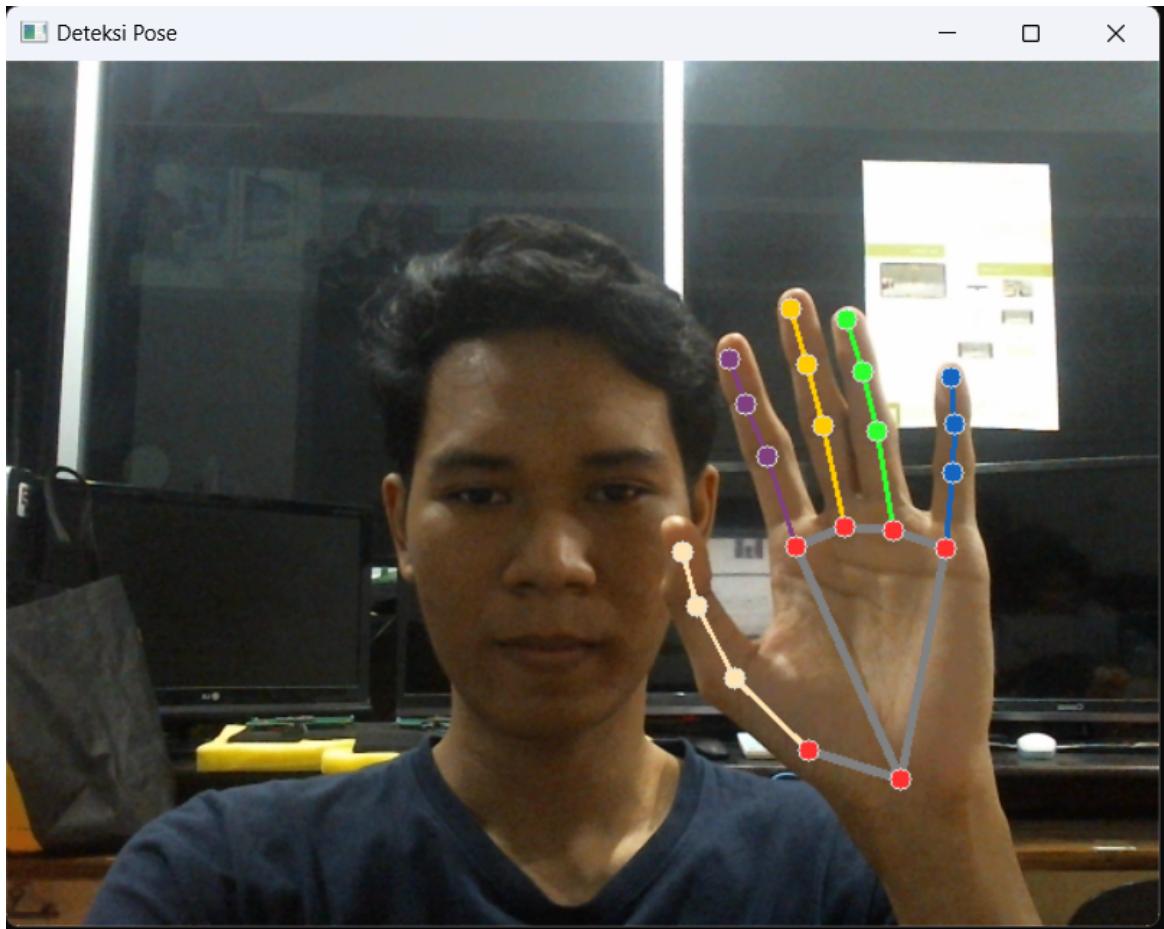
Deteksi pose merupakan suatu proses yang melibatkan penggunaan bahasa pemrograman Python bersama dengan *library* OpenCV dan *framework* Mediapipe. Dalam konteks ini, Mediapipe berperan penting dalam mendapatkan informasi titik-titik *landmark* yang signifikan pada objek yang diidentifikasi. *Landmark* ini kemudian menjadi dasar untuk membentuk suatu representasi visual yang memvisualisasikan pose tersebut. Proses selanjutnya melibatkan penghubungan titik-titik *landmark* yang telah ditentukan, di mana garis-garis dibuat untuk menggambarkan relasi spasial antar titik-titik tersebut. Dengan demikian, prosedur ini tidak hanya mengandalkan Mediapipe sebagai *framework* utama, tetapi juga memanfaatkan OpenCV sebagai alat bantu untuk analisis citra dan manipulasi visual yang diperlukan dalam deteksi pose.

Pada penelitian kali ini akan memanfaatkan teknologi *hand pose* dari Mediapipe untuk mengontrol gerak kursi roda. Terdapat beberapa titik *keypoints* yang akan digunakan pada estimasi pose ini. Titik *keypoints* yang akan digunakan pada estimasi pose ini dapat dilihat pada Tabel 3.1.

Tabel 3.1: Tabel titik *keypoints* yang relevan pada tahap estimasi pose

Nomor Keypoint	Nama Keypoint
0	Pergelangan Tangan
1	CMC Ibu Jari
2	MPM Ibu Jari
3	IP Ibu Jari
4	TIP Ibu Jari
5	MCP Telunjuk
6	PIP Telunjuk
7	DIP Telunjuk
8	TIP Telunjuk
9	MCP Jari Tengah
10	PIP Jari Tengah
11	DIP Jari Tengah
12	TIP Jari Tengah
13	MCP Jari Manis
14	PIP Jari Manis
15	DIP Jari Manis
16	TIP Jari Manis
17	MCP Kelingking
18	PIP Kelingking
19	DIP Kelingking
20	TIP Kelingking

Setiap titik *landmark* yang terdapat pada peraga akan diwarnai dengan warna yang unik untuk membedakan setiap jarinya. Secara spesifik, warna yang diberikan pada setiap titik *landmark* mencerminkan asosiasi dengan jari tertentu, sehingga menciptakan representasi visual yang lebih terperinci dan informatif. Untuk memberikan gambaran yang lebih konkret maka berikut ini contoh citra yang telah diestimasi pose yang dapat dilihat pada Gambar 3.2.



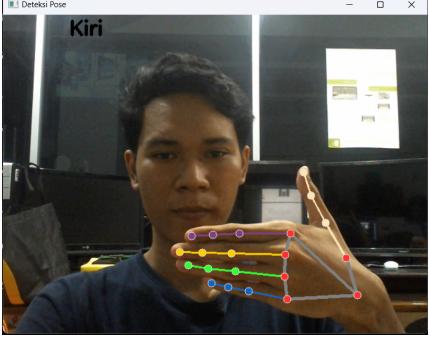
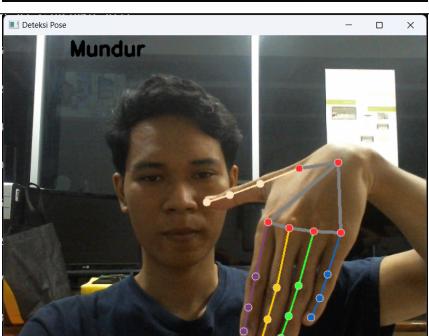
Gambar 3.2: Contoh citra yang telah diestimasi pose

### 3.1.2 Klasifikasi Pose

Setelah proses estimasi pose tangan selesai, maka langkah selanjutnya ada mengelompokkan citra-citra hasil estimasi menjadi suatu dataset. Dataset ini akan memiliki 5 kelas berbeda yang masing-masing merepresentasikan perintah untuk maju, mundur, bergerak ke kanan, bergerak ke kiri, dan berhenti. Kelas ini mewakili perintah dasar untuk menggerakkan kursi roda.

Untuk meningkatkan kinerja dan akurasi maka dataset ini kemudian akan melewati proses *training* menggunakan algoritma *Convolutional Neural Network* (CNN). Penggunaan CNN dalam *training* dataset diharapkan dapat menghasilkan model yang mampu mengenali pola dan fitur yang kompleks, sehingga memungkinkan sistem untuk merespon dengan tepat terhadap variasi perintah yang mungkin diberikan oleh peraga. Hasil dari model prediksi yang telah dibuat dapat dilihat pada Tabel 3.2.

Tabel 3.2: Tabel contoh klasifikasi citra

Pose	Citra
Kiri	
Maju	
Stop	
Mundur	
Kanan	

### 3.1.3 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.

$$\text{Arah}(\text{char}), \text{Kecepatan}(\text{integer}) \quad (3.1)$$

Variabel arah memiliki tipe data *char* yang akan menentukan gerak dari motor kursi roda, serta variabel kecepatan memiliki tipe data *integer* yang akan menentukan kecepatan maksimal dari kursi roda. Untuk memperkecil ukuran data maka kode instruksi untuk menentukan arah gerak menggunakan satu huruf untuk mewakili setiap gerakan. Kode instruksi dapat dilihat pada Tabel 3.3.

Tabel 3.3: Kode instruksi dari hasil klasifikasi

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

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

### 3.1.4 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.

### 3.1.5 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 3.3. 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 kecepatan maksimal yang sebelumnya telah ditetapkan oleh pemguna. 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.

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 3.1.

Program 3.1: Program Untuk Memeriksa MAC Address Pada ESP32

---

```
1 #include <BluetoothSerial.h>
2 BluetoothSerial SerialBT;
3
4 void setup() {
5   Serial.begin(115200);
6   SerialBT.begin("ESP32_Haris"); // Bluetooth device name
7   delay(1000); // Give some time for the Bluetooth module to←
                  initialize
8 }
9
10 void loop() {
11   // Get the ESP32 Bluetooth address
12   uint8_t esp32Address[6];
13   esp_efuse_mac_get_default(esp32Address);
14
15   // Print the address in a readable format
16   Serial.printf("ESP32 Bluetooth Address: %02X:%02X:%02X:%02X←
                  :%02X:%02X\n",
17           esp32Address[0], esp32Address[1], ←
                   esp32Address[2],
18           esp32Address[3], esp32Address[4], ←
                   esp32Address[5]);
19 }
```

---

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 den-

gan 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 3.2 yang digunakan untuk menguji kemampuan ESP32 dalam menerima data string melalui Bluetooth.

Program 3.2: Program Untuk Menerima Data String Melalui Bluetooth Pada ESP32

```
1 #include <BluetoothSerial.h>
2 BluetoothSerial SerialBT;
3
4 int maxspeed;
5
6 void setup() {
7     Serial.begin(115200);
8     SerialBT.begin("ESP32_Haris");
9 }
10
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
25         if (maxspeed < 20) {
26             Serial.println("TRUE");
27         }
28         else{
29             Serial.println("FALSE");
30         }
31 }
```

```
31     }
32 }
```

---

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(). Pertama-tama 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*.

### 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 3.3.

Program 3.3: 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');
```

```

16 // Deserialize the JSON
17 DynamicJsonDocument doc(1024);
18 DeserializationError error = deserializeJson(doc, ←
19     json_data);
20
21 if (error) {
22     Serial.println("Failed to parse JSON");
23 } else {
24     // Extract values from JSON
25     const char* arah = doc["arah"];
26     const char* kecepatan = doc["kecepatan"];
27
28     // Perform actions based on the received data
29     Serial.print("Received Arah: ");
30     Serial.println(arah);
31     Serial.print("Received Kecepatan: ");
32     Serial.println(kecepatan);
33 }
34 }
```

---

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*.

### 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 3.4 yang digunakan untuk menerima data string melalui Access Point WiFi pada ESP32.

Program 3.4: 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 (↔
18   // 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()) {
```

```

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 client.stop();
61
62
63 }
64 }
```

---

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 3.5 yang digunakan untuk menerima data JSON melalui *Access Point WiFi* pada ESP32.

Program 3.5: 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 (←
19   // Access Point) to be open
20   WiFi.softAP(ssid, password);
21
22   IPAddress IP = WiFi.softAPIP();
23   Serial.print("ESP32 AP IP Address : ");
24   Serial.println(IP);
25
26   // Start the server
27   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, ←
44                     json_data);
45
46                 if(error) {
47                     Serial.println("Failed to parse JSON");
48                 }
49                 else{
50                     const char* arah = doc["arah"];
51                     int kecepatan = doc["kecepatan"];
52
53                     Serial.print("Received Arah : ");
54                     Serial.print(arah);
55                     Serial.print(" Received Kecepatan : ");
56                     Serial.println(kecepatan);
57                 }
58             }
59
60             client.stop();
61
62     }
63 }
```

---

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.

### 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 3.6 yang digunakan untuk menguji rangkaian kontrol ESP32.

Program 3.6: 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     delay(1000);
96
97 //BELOK KANAN
98     while (PWM1_DutyCycle < maxspeed)
99     {
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 }
```

---

*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.

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 mengkonfigurasi 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 se-

cara bertahap hingga mencapai nilai maxspeed yang telah ditentukan. Selama itu juga kecepatan putar motor akan bertambah secara bertahap. Penambahan nilai PWM1\_DutyCycle akan ditunda 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 3.7 yang digunakan untuk mengontrol motor kursi roda melalui Bluetooth.

Program 3.7: 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.←
49       indexOf(','));
50     String kecepatan = receivedData.substring(receivedData.←
51       indexOf(',') + 1);
52
53     maxspeed = kecepatan.toInt();
54     turnspeed = maxspeed / 2;
55
56     Serial.print("Arah : ");
57     Serial.println(arah);
58     Serial.print("Kecepatan : ");
59     Serial.println(kecepatan);
60
61     if (arah == "A") {
62       while (PWM1_DutyCycle <= turnspeed) {
63         stdir[0] = LOW;
64         stdir[1] = LOW;
65         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 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.

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.available(). 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 3.8 yang digunakan untuk mengontrol motor kursi roda melalui WiFi.

Program 3.8: 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;
30 int turnspeed = 35;
31
32 void setup() {
33   Serial.begin(115200);
34   Serial.print("Setting AP (Access Point) ...");
35   WiFi.softAP(ssid, password);
36
37   IPAddress IP = WiFi.softAPIP();
38   Serial.print("ESP32 AP IP Address : ");
39   Serial.println(IP);
40
41   server.begin();
42
43   pinMode(dir1, OUTPUT);
44   pinMode(dir2, OUTPUT);
45   pinMode(dir3, OUTPUT);
46   pinMode(dir4, OUTPUT);
47
48   ledcSetup(pwmChannel1, freq, res);
49   ledcSetup(pwmChannel2, freq, res);
50
51   ledcAttachPin(pwmpin1, pwmChannel1);
52   ledcAttachPin(pwmpin2, pwmChannel2);
53 }
54
55 void loop() {
56   WiFiClient client = server.available();
57   if(client){
58     while(client.connected()){
59       if(client.available()){
60
61         String arah = client.readStringUntil('\n');
62         Serial.print("Arah : ");
63         Serial.println(arah);
64
65         if(arah == "A"){

```

```

66     while(PWM1_DutyCycle <= turnspeed) {
67         stdir[0] = LOW;
68         stdir[1] = LOW;
69         stdir[2] = HIGH;
70         stdir[3] = LOW;
71
72         digitalWrite(dir1, stdir[0]);
73         digitalWrite(dir2, stdir[1]);
74         digitalWrite(dir3, stdir[2]);
75         digitalWrite(dir4, stdir[3]);
76         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 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 di-

gunakan 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.

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 3.9 yang digunakan untuk mengirim data string melalui Bluetooth.

Program 3.9: Program Untuk Mengirim Data String Melalui Bluetooth

```
1 import bluetooth
2 import datetime
3
4 # ESP32's Bluetooth MAC address
5 esp32_mac_address = "30:C6:F7:2F:31:86" # Replace with your ←
      ESP32's MAC address
6
7 # Establish a Bluetooth connection
8 sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
9 sock.connect((esp32_mac_address, 1)) # Channel 1 is commonly←
      used for SPP (Serial Port Profile)
```

```

11 while True:
12     # Send two data values
13     arah = input("Enter arah : ")
14     kecepatan = input("Enter kecepatan : ")
15     message = f"{arah},{kecepatan}"
16     date = datetime.datetime.now()
17     print(date)
18
19     if arah == "q" or kecepatan == "q":
20         break
21
22     sock.send(message)
23
24 # Close the Bluetooth connection
25 sock.close()

```

---

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 3.10 yang digunakan untuk mengirimkan data string melalui WiFi.

Program 3.10: 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
21     if arah == "q" or kecepatan == "q":
22         break
23
24     s.send(message.encode('utf-8'))
25
26 s.close()

```

---

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 3.11 yang digunakan untuk mengirim data JSON melalui Bluetooth.

Program 3.11: Program Untuk Mengirim Data JSON Melalui Bluetooth

---

```

1 import bluetooth
2 import json
3
4 # ESP32 Bluetooth address
5 esp32_address = "EC:62:60:9B:E4:92" # Replace with your ←
6           ESP32's Bluetooth address
7 # Create a Bluetooth socket

```

```

8 sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
9
10 # Connect to the ESP32
11 sock.connect((esp32_address, 1))
12
13 while True:
14     # Get keyboard input for two values
15     arah = input("Masukkan Arah: ")
16     kecepatan = input("Masukkan Kecepatan: ")
17
18     # Create a JSON object
19     json_data = {"arah": arah, "kecepatan": kecepatan}
20
21     # Serialize the JSON data
22     json_string = json.dumps(json_data)
23
24     if arah == "q" or kecepatan == "q":
25         break
26
27     # Send the serialized JSON over Bluetooth
28     sock.send(json_string)
29
30 # Close the Bluetooth socket
31 sock.close()

```

---

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 3.12 yang digunakan untuk mengirimkan data JSON melalui WiFi.

Program 3.12: Program Untuk Mengirim Data JSON Melalui WiFi

---

```
1 import socket
```

```

2 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()

```

---

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*

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.

$$Arah(char), Kecepatan(integer) \quad (4.1)$$

Variabel arah memiliki tipe data *char* yang digunakan untuk menentukan arah gerak dari motor kursi roda serta variabel kecepatan memiliki tipe data *integer* 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
C,40	22:02:06.857715	22:02:07.892	1.034285
A,226	22:02:08.358496	22:02:09.390	1.031504
E,21	22:02:09.859346	22:02:10.921	1.061654
C,168	22:02:11.360115	22:02:12.405	1.044885
B,247	22:02:12.860932	22:02:13.900	1.039068
B,72	22:02:14.361692	22:02:15.400	1.038308
E,129	22:05:28.232092	22:05:29.283	1.050908
D,117	22:05:29.732554	22:05:30.765	1.032446
C,31	22:05:31.233293	22:05:32.264	1.030707
B,248	22:05:32.734151	22:05:33.779	1.044849

D,126	22:05:34.235059	22:05:35.271	1.035941
C,2	22:07:30.543505	22:07:31.599	1.055495
B,29	22:07:32.044303	22:07:33.077	1.032697
C,247	22:07:33.545370	22:07:34.595	1.04963
A,190	22:07:35.046573	22:07:36.085	1.038427
A,49	22:07:36.547315	22:07:37.575	1.027685
B,25	22:09:44.370413	22:09:45.405	1.034587
D,72	22:09:45.871428	22:09:46.921	1.049572
C,63	22:09:47.372407	22:09:48.415	1.042593
A,245	22:09:48.873310	22:09:49.885	1.01169
C,70	22:09:50.374135	22:09:51.406	1.031865
D,240	22:09:51.874944	22:09:52.929	1.054056
A,46	22:11:46.717729	22:11:47.744	1.026271
B,245	22:11:48.218593	22:11:49.263	1.044407
B,68	22:11:49.719513	22:11:50.763	1.043487
E,204	22:11:51.219962	22:11:52.271	1.051038
B,26	22:11:52.720738	22:11:53.746	1.025262
D,113	22:13:39.843434	22:13:40.871	1.027566
B,195	22:13:41.344420	22:13:42.389	1.04458
A,242	22:13:42.845419	22:13:43.876	1.030581
A,17	22:13:44.346312	22:13:45.360	1.013688
A,140	22:13:45.847280	22:13:46.915	1.06772
Average Delay Time		1.038982875	

Tabel 4.1 menampilkan pengiriman data string yang terdiri dari 2 nilai dan dipisahkan dengan simbol koma (”, ”). Dalam pengujian kali ini, data dikirimkan sebanyak 32 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 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 1.038982875 detik.

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

Data	Timestamp Sent	Timestamp Received	Delay Time
D	17:56:13.163165	17:56:14.352	1.189
E	17:56:14.260985	17:56:14.491	0.230
D	17:56:14.442510	17:56:14.708	0.265
E	17:56:14.649861	17:56:14.878	0.228
D	17:56:14.782992	17:56:15.017	0.234
C	17:56:14.957997	17:56:15.188	0.230
A	17:56:15.096377	17:56:15.311	0.215
D	17:56:15.255594	17:56:15.483	0.227
D	17:56:15.406847	17:56:15.745	0.338

E	17:56:15.663510	17:56:15.930	0.266
D	17:56:15.893686	17:56:16.195	0.301
C	17:56:16.118896	17:56:16.320	0.201
A	17:56:16.287623	17:56:16.458	0.170
A	17:56:16.416326	17:56:16.674	0.258
A	17:56:16.600528	17:56:16.862	0.261
C	17:56:16.800782	17:56:17.080	0.279
A	17:56:16.972893	17:56:17.360	0.387
C	17:56:17.250136	17:56:17.641	0.391
B	17:56:17.544344	17:56:17.779	0.235
C	17:56:17.728154	17:56:18.012	0.284
A	17:56:17.945683	17:56:18.151	0.205
D	17:56:18.097472	17:56:18.382	0.285
A	17:56:18.302853	17:56:18.614	0.311
D	17:56:18.550624	17:56:18.818	0.267
B	17:56:18.745921	17:56:18.990	0.244
A	17:56:18.932188	17:56:19.162	0.230
B	17:56:19.129604	17:56:19.596	0.466
C	17:56:19.482713	17:56:19.937	0.454
C	17:56:19.876232	17:56:20.341	0.465
C	17:56:20.216948	17:56:20.649	0.432
E	17:56:20.588913	17:56:21.052	0.463
B	17:56:20.913091	17:56:21.468	0.555
C	17:56:21.323255	17:56:21.873	0.550
C	17:56:21.820789	17:56:22.278	0.457
B	17:56:22.149208	17:56:22.807	0.658
Average Delay Time		0.3495228571	

Tabel 4.2 menampilkan pengiriman data string yang terdiri dari 1 nilai, yaitu arah. Dalam pengujian kali ini, data dikirimkan sebanyak 35 kali secara berturut-turut tanpa penambahan waktu *delay* setiap kali mengirimkan data. Hasil dari pengujian ini menunjukkan bahwa waktu pengiriman rata-rata dari laptop menuju ESP32 melalui Bluetooth adalah sebesar 0.3495228571 detik.

Ditemukan perbedaan waktu *delay* yang cukup 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 1.038982875 detik. Dalam perbandingan dengan pengujian yang melibatkan string 1 nilai, waktu *delay* yang tercatat hanya sekitar 0.3495228571 detik. Penting untuk diperhatikan bahwa terdapat kekurangan saat mengirimkan data string yang mengandung 2 nilai, yaitu adanya penambahan *delay* sebesar 1.5 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 Bluetooth, lebih disarankan untuk mengirimkan string dengan hanya 1 nilai guna menghindari *delay* tambahan yang dapat mempengaruhi efisiensi dan kecepatan transmisi data secara keseluruhan.

## 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

$$\{ \text{'arah'} : \text{arah(char)}, \text{'kecepatan'} : \text{kecepatan(integer)} \} \quad (4.2)$$

$$\{ \text{'arah'} : \text{arah(char)} \} \quad (4.3)$$

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 integer 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': 'C', 'kecepatan': 73}	20:50:22.830017	20:50:23.901	1.070983
{'arah': 'D', 'kecepatan': 69}	20:50:23.931177	20:50:24.975	1.043823
{'arah': 'D', 'kecepatan': 140}	20:50:25.031644	20:50:26.109	1.077356
{'arah': 'A', 'kecepatan': 112}	20:50:26.132228	20:50:27.177	1.044772
{'arah': 'B', 'kecepatan': 125}	20:50:27.232932	20:50:28.265	1.032068
{'arah': 'A', 'kecepatan': 141}	20:50:28.333797	20:50:29.402	1.068203
{'arah': 'D', 'kecepatan': 210}	20:50:29.434256	20:50:30.531	1.096744
{'arah': 'E', 'kecepatan': 48}	20:50:30.534885	20:50:31.585	1.050115
{'arah': 'D', 'kecepatan': 247}	20:50:31.635389	20:50:32.671	1.035611
{'arah': 'E', 'kecepatan': 48}	20:50:32.736202	20:50:33.818	1.081798
{'arah': 'A', 'kecepatan': 117}	20:50:33.837167	20:50:34.942	1.104833
{'arah': 'B', 'kecepatan': 175}	20:50:34.937984	20:50:35.980	1.042016
{'arah': 'C', 'kecepatan': 121}	20:50:36.039064	20:50:37.066	1.026936
{'arah': 'E', 'kecepatan': 55}	20:50:37.140106	20:50:38.211	1.070894
{'arah': 'A', 'kecepatan': 134}	20:50:38.240672	20:50:39.338	1.097328
{'arah': 'A', 'kecepatan': 241}	20:50:39.341402	20:50:40.380	1.038598
{'arah': 'C', 'kecepatan': 8}	20:50:40.442061	20:50:41.484	1.041939
{'arah': 'C', 'kecepatan': 242}	20:50:41.542481	20:50:42.604	1.061519
{'arah': 'E', 'kecepatan': 251}	20:50:42.642951	20:50:43.741	1.098049
{'arah': 'C', 'kecepatan': 65}	20:50:43.743514	20:50:44.754	1.010486
{'arah': 'B', 'kecepatan': 184}	20:50:44.844210	20:50:45.889	1.044790
{'arah': 'D', 'kecepatan': 244}	20:50:45.944934	20:50:47.010	1.065066
{'arah': 'D', 'kecepatan': 216}	20:50:47.045304	20:50:48.138	1.092696
{'arah': 'E', 'kecepatan': 209}	20:50:48.145895	20:50:49.194	1.048105
{'arah': 'C', 'kecepatan': 131}	20:50:49.246447	20:50:50.286	1.039553

{'arah': 'D', 'kecepatan': 172}	20:50:50.346836	20:50:51.407	1.060164
{'arah': 'C', 'kecepatan': 166}	20:50:51.447450	20:50:52.510	1.062550
{'arah': 'E', 'kecepatan': 47}	20:50:52.548494	20:50:53.596	1.047506
{'arah': 'E', 'kecepatan': 193}	20:50:53.648852	20:50:54.701	1.052148
{'arah': 'A', 'kecepatan': 54}	20:50:54.749663	20:50:55.812	1.062337
{'arah': 'B', 'kecepatan': 158}	20:50:55.850048	20:50:56.917	1.066952
Average Delay Time			1.059223806

Tabel 4.3 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 1.1 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 1.059223806 detik.

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

Data	Timestamp Sent	Timestamp Received	Delay Time
{'arah': 'B'}	20:18:19.920784	20:18:20.984	1.063216
{'arah': 'D'}	20:18:21.040387	20:18:22.091	1.050613
{'arah': 'E'}	20:18:22.093848	20:18:23.150	1.056152
{'arah': 'A'}	20:18:23.185556	20:18:24.238	1.052444
{'arah': 'C'}	20:18:24.241774	20:18:25.281	1.039226
{'arah': 'C'}	20:18:25.387649	20:18:26.526	1.138351
{'arah': 'B'}	20:18:26.514085	20:18:27.550	1.035915
{'arah': 'D'}	20:18:27.626023	20:18:28.683	1.056977
{'arah': 'C'}	20:18:28.757853	20:18:29.807	1.049147
{'arah': 'D'}	20:18:29.905233	20:18:31.037	1.131767
{'arah': 'C'}	20:18:31.117422	20:18:32.255	1.137578
{'arah': 'B'}	20:18:32.229935	20:18:33.284	1.054065
{'arah': 'E'}	20:18:33.367298	20:18:34.424	1.056702
{'arah': 'E'}	20:18:34.474402	20:18:35.527	1.052598
{'arah': 'E'}	20:18:35.612564	20:18:36.661	1.048436
{'arah': 'C'}	20:18:36.740207	20:18:37.795	1.054793
{'arah': 'A'}	20:18:37.857143	20:18:38.917	1.059857
{'arah': 'E'}	20:18:38.997421	20:18:40.052	1.054579
{'arah': 'E'}	20:18:40.134763	20:18:41.264	1.129237
{'arah': 'E'}	20:18:41.242444	20:18:42.274	1.031556
{'arah': 'D'}	20:18:42.457488	20:18:43.520	1.062512
{'arah': 'C'}	20:18:43.614871	20:18:44.749	1.134129
{'arah': 'D'}	20:18:44.738330	20:18:45.852	1.11367
{'arah': 'A'}	20:18:45.857615	20:18:46.909	1.051385
{'arah': 'E'}	20:18:46.987464	20:18:48.108	1.120536

{'arah': 'B'}	20:18:48.117471	20:18:49.234	1.116529
{'arah': 'A'}	20:18:49.314138	20:18:50.375	1.060862
{'arah': 'D'}	20:18:50.440545	20:18:51.498	1.057455
{'arah': 'C'}	20:18:51.667602	20:18:52.728	1.060398
{'arah': 'C'}	20:18:52.802780	20:18:53.846	1.04322
{'arah': 'C'}	20:18:53.935965	20:18:54.985	1.049035
{'arah': 'C'}	20:18:55.069708	20:18:56.171	1.101292
{'arah': 'D'}	20:18:56.275415	20:18:57.292	1.016585
{'arah': 'B'}	20:18:57.399101	20:18:58.445	1.045899
{'arah': 'D'}	20:18:58.536479	20:18:59.581	1.044521
Average Delay Time			1.069463914

Tabel 4.4 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 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 1.069463914 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 1.059223806 detik. Jika dibandingkan dengan pengujian yang mengirimkan JSON dengan 1 *key-value*, waktu *delay* yang tercatat adalah sebesar 1.069463914 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.

### 4.3 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

$$\text{Arah}(\text{char}), \text{Kecepatan}(\text{integer}) \quad (4.4)$$

Variabel arah memiliki tipe data char yang digunakan untuk menentukan arah gerak dari motor kursi roda serta variabel kecepatan yang memiliki tipe data integer 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
D,175	22:34:36.127972	22:34:37.159	1.031028
D,220	22:34:37.428370	22:34:38.433	1.00463
E,164	22:34:38.728955	22:34:39.760	1.031045
E,109	22:34:40.029352	22:34:41.068	1.038648
D,187	22:34:41.329739	22:34:42.371	1.041261
A,174	22:34:42.630322	22:34:43.649	1.018678
A,176	22:34:43.931092	22:34:44.957	1.025908
C,245	22:34:45.231675	22:34:46.276	1.044325
E,189	22:34:46.532305	22:34:47.821	1.288695
B,149	22:34:47.832930	22:34:48.865	1.03207
D,113	22:34:49.133646	22:34:50.179	1.045354
A,150	22:34:50.434051	22:34:51.482	1.047949
A,216	22:34:51.734834	22:34:52.776	1.041166
B,21	22:34:53.035137	22:34:54.210	1.174863
B,153	22:34:54.335986	22:34:55.466	1.130014
E,214	22:34:55.636653	22:34:56.679	1.042347
C,231	22:34:56.937379	22:34:57.974	1.036621
E,41	22:34:58.237752	22:34:59.514	1.276248
B,210	22:34:59.538330	22:35:00.582	1.04367
C,38	22:35:00.838618	22:35:01.871	1.032382
E,83	22:35:02.139088	22:35:03.182	1.042912
A,52	22:35:03.439819	22:35:04.441	1.001181
A,106	22:35:04.740350	22:35:05.764	1.02365
C,255	22:35:06.040967	22:35:07.069	1.028033
B,205	22:35:07.341426	22:35:08.380	1.038574
C,4	22:35:08.642326	22:35:09.764	1.121674
C,158	22:35:09.942969	22:35:10.994	1.051031
A,64	22:35:11.243758	22:35:12.268	1.024242
E,132	22:35:12.544406	22:35:13.581	1.036594
D,248	22:35:13.844881	22:35:14.871	1.026119
B,194	22:35:15.145789	22:35:16.175	1.029211
Average Delay Time			1.059681387

Tabel 4.5 menampilkan pengiriman data string yang terdiri dari 2 nilai dan dipisahkan dengan simbol koma (”, ”). Dalam pengujian kali ini, data dikirimkan sebanyak 31 kali secara berturut-turut dengan penambahan waktu *delay* sebesar 1.3 detik setiap kali mengirimkan data. 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 1.059681387 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	21:05:44.335642	21:05:44.386	0.050358
E	21:05:44.380617	21:05:44.433	0.052383
B	21:05:44.474109	21:05:44.526	0.051891
E	21:05:44.544724	21:05:44.572	0.027276
E	21:05:44.674358	21:05:44.682	0.007642
E	21:05:44.684230	21:05:44.728	0.04377
E	21:05:44.787668	21:05:44.822	0.034332
E	21:05:44.834278	21:05:44.854	0.019722
C	21:05:44.847274	21:05:44.899	0.051726
A	21:05:44.855957	21:05:44.899	0.043043
C	21:05:44.968575	21:05:44.977	0.008425
D	21:05:44.988174	21:05:45.024	0.035826
C	21:05:45.018276	21:05:45.062	0.043724
A	21:05:45.054604	21:05:45.100	0.045396
A	21:05:45.069057	21:05:45.100	0.030943
C	21:05:45.123548	21:05:45.147	0.023452
A	21:05:45.144644	21:05:45.194	0.049356
E	21:05:45.164004	21:05:45.194	0.029996
A	21:05:45.218556	21:05:45.270	0.051444
B	21:05:45.245502	21:05:45.270	0.024498
C	21:05:45.347689	21:05:45.373	0.025311
C	21:05:45.355225	21:05:45.373	0.017775
E	21:05:45.366221	21:05:45.410	0.043779
A	21:05:45.413087	21:05:45.456	0.042913
C	21:05:45.458212	21:05:45.492	0.033788
C	21:05:45.482171	21:05:45.534	0.051829
A	21:05:45.529777	21:05:45.579	0.049223
C	21:05:45.603062	21:05:45.626	0.022938
E	21:05:45.670848	21:05:45.710	0.039152
D	21:05:45.682481	21:05:45.710	0.027519
B	21:05:45.703240	21:05:45.749	0.04576
D	21:05:45.800390	21:05:45.843	0.04261
B	21:05:45.809048	21:05:45.843	0.033952
E	21:05:45.858800	21:05:45.875	0.0162
D	21:05:45.868054	21:05:45.875	0.006946
Average Delay Time		0.03499708571	

Tabel 4.6 menampilkan pengiriman data string yang terdiri dari 1 nilai, yaitu arah. Dalam pengujian kali ini, data dikirimkan sebanyak 35 kali secara berturut-turut tanpa penambahan waktu *delay* setiap kali mengirimkan data. Hasil dari pengujian ini menunjukkan bahwa waktu pengiriman rata-rata dari laptop menuju ESP32 melalui WiFi adalah sebesar 0.03499708571 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 transmisi 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.

$$\{ \text{'arah'} : \text{arah(char)}, \text{'kecepatan'} : \text{kecepatan(integer)} \} \quad (4.5)$$

$$\{ \text{'arah'} : \text{arah(char)} \} \quad (4.6)$$

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 integer 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.

*[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 diapaparkan 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**

Untuk pengembangan lebih lanjut pada Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum

gravida mauris. antara lain:

1. Memperbaiki Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus.
2. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa.
3. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna.

## DAFTAR PUSTAKA

- Abrahamsen, H. (2023, October). Bluetoothserial. <https://www.arduino.cc/reference/en/libraries/bluetoothserial/>
- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6. <https://doi.org/10.1109/ICEngTechnol.2017.8308186>
- 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>
- Developer, N. (2023). <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>
- Dwitama, A. P. J. (2019). *Klasifikasi tingkat retakan pada bangunan berbasis citra menggunakan metode convolution neural network* [Doctoral dissertation, Universitas Mataram].
- Hakim, R. F., et al. (2018). Penerapan deep learning menggunakan convolutional neural network untuk klasifikasi citra wayang punakawan.
- 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.
- 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. (n.d.). Wheelchair control using bluetooth-based electromyography signals [kontrol kursi roda menggunakan sinyal elektromiografi berbasis bluetooth].
- 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 melewati bluetooth. *Prosiding Seminar Nasional Penelitian dan Pengabdian Kepada Masyarakat*, 1(1), 26–30.

*[Halaman ini sengaja dikosongkan]*

## BIOGRAFI PENULIS



I Putu Haris Setiadi Ekatama, lahir pada Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

*[Halaman ini sengaja dikosongkan]*