PROTOTYPE PENGATURAN KECEPATAN DAN KENDALI JARAK OTOMATIS PADA MOBIL LISTRIK TERHADAP BAHAYA KECELAKAAN MENGGUNAKAN METODE *FUZZY LOGIC CONTROLLER* (FLC)

PROYEK AKHIR

Laporan akhir ini dibuat dan diajukan untuk memenuhi salah satu syarat kelulusan
Diploma III Politeknik Manufaktur Negeri Bangka Belitung

Disusun oleh:

Tegar Prayogi NIM: 0032030

Sri Agustini NIM: 0032028

POLITEKNIK MANUFAKTUR NEGERI

BANGKA BELITUNG

TAHUN 2023

LEMBAR PENGESAHAN

JUDUL PROYEK AKHIR

PROTOTYPE PENGATURAN KECEPATAN DAN KENDALI JARAK OTOMATIS PADA MOBIL LISTRIK TERHADAP BAHAYA KECELAKAAN MENGGUNAKAN METODE *FUZZY LOGIC CONTROLLER* (FLC)

Oleh:

Tegar Prayogi/0032030

Sri Agustini/0032028

Laporan akhir ini telah disetujui dan disahkan sebagai salah satu syarat kelulusan Program Diploma III Politeknik Manufaktur Negeri Bangka Belitung

Menyetujui,

Pembimbing 1	Pembimbing 2
Eko Sulistyo, M.T.	I Made Andik Setiawan, M.Eng.Ph.D.
Penguji 1	Penguji 2
Irwan, M.Sc., Ph.D.	Indah Riezky Pratiwi, M.Pd.

PERNYATAAN BUKAN PLAGIAT

Yang bertanda tangan di bawah ini:

Nama Mahasiswa 1: Tegar Prayogi NIM: 0032030
Nama Mahasiswa 2: Sri Agustini NIM: 0032028

Dengan Judul: *PROTOTYPE* PENGATURAN KECEPATAN DAN KENDALI JARAK OTOMATIS PADA MOBIL LISTRIK TERHADAP BAHAYA KECELAKAAN MENGGUNAKAN METODE *FUZZY LOGIC CONTROLLER* (FLC)

Menyatakan bahwa laporan akhir ini adalah hasil kerja kami sendiri dan bukan merupakan plagiat. Pernyataan ini kami buat dengan sebenarnya dan bila ternyata di kemudian hari melanggar pernyataan ini, kami bersedia menerima sanksi yang berlaku.

Sungailiat, 24 Juli 2023

Nama Mahasiswa	Tanda Tangan
1. Tegar Prayogi	.....
2. Sri Agustini	……..

ABSTRAK

Mobil listrik merupakan sebuah mobil yang akan bergerak dengan motor listrik DC dan memanfaatkan energi listrik. Pada mobil listrik saat ini pengereman mobil listrik hanya bisa dilakukan secara manual dengan perkiraan pengemudi saja. Oleh karena itu, untuk mencegah terjadinya kecelakaan terutama pada mobil listrik diperlukan sebuah sistem pengontrolan kecepatan yang akan mengurangi kecepatan dengan melakukan pengereman secara otomatis. Penelitian ini bertujuan untuk mengontrol kecepatan dan pengereman mobil listrik jika terdapat objek di depan mobil listrik tersebut menggunakan sensor ultrasonik dengan metode Fuzzy Logic Controller (FLC) dan membuat kontrol jarak mobil dengan objek yang ada di kiri, kanan dan belakang. Metode penelitan yang digunakan yaitu pengontrolan motor DC yang dilakukan menggunakan Fuzzy Logic Controller (FLC) dengan inputan kecepatan motor dan jarak ke objek yang ada di depan mobil listrik. Kemudian dilakukan difuzzifikasi untuk menentukan aksi output dari motor berupa melambat atau berhenti. Pengujian dilakukan dengan meletakkan objek di depan mobil listrik dengan jarak sangat dekat (10 cm) sampai dengan sangat jauh (400 cm). Hasil pengujian prototype jika tanpa beban yaitu mobil akan berhenti atau kecepatan 0 RPM jika jarak terhadap objek < 50 cm dengan rata-rata delay 2,9 detik. Sedangkan, jika menggunakan beban pada pengujian mulai dari 100 cm, mobil akan berhenti rata-rata pada jarak 70 cm dikarenakan torsi motor tidak kuat. Jika pengujian mulai dari 200 cm mobil kebanyakan menabrak dikarenakan ada delay pada pembacaan sensor.

Kata kunci: Fuzzy Logic Controller (FLC), kendali jarak, pengaturan kecepatan

ABSTRACT

An electric car is a car that will move with a DC electric motor and utilize electrical energy. In electric cars, currently braking electric cars can only be done manually with the driver's estimate. Therefore, to prevent accidents, especially in electric cars, a speed control system is needed that will reduce speed by braking automatically. This study aims to control the speed and braking of the electric car if there is an object in front of the electric car using ultrasonic sensors with the Fuzzy Logic Controller (FLC) method and make a control of the car's distance with objects on the left, right and rear. The research method used is DC motor control which is carried out using a Fuzzy Logic Controller (FLC) with motor speed input and distance to objects in front of the electric car. Then difuzzification is carried out to determine the output action of the motor in the form of slowing down or stopping. The test was carried out by placing the object in front of the electric car with a very close distance (10 cm) to very far (400 cm). The results of prototype testing if without load, namely the car will stop or speed 0 RPM if the distance to the object < 50 cm with an average delay of 2.9 seconds. Meanwhile, if you use a load on the test starting from 100 cm, the car will stop on average at a distance of 70 cm because the motor torque is not strong. If the test starts at 200 cm, most cars crash because there is a delay in sensor readings.

Keywords: Fuzzy Logic Controller (FLC), distance control, speed regulation

KATA PENGANTAR

Bismillahirrahmanirrahim

Penulis ucapkan puji dan syukur atas kehadirat Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya Penulis dapat menyelesaikan makalah proyek akhir ini dengan baik dan sesuai dengan waktu yang telah ditentukan.

Untuk memenuhi salah satu syarat wajib kelulusan Diploma III di Politeknik Manufaktur Negeri Bangka Belitung maka diperlukan untuk membuat makalah proyek akhir ini. Melalui makalah ini bertujuan untuk membuat pembaca dapat mengetahui gambaran proyek akhir yang dibuat oleh penulis. Makalah proyek akhir ini dibuat berdasarkan pengembangan dari jurnal-jurnal penelitian yang sudah pernah dilakukan sebelumnya. Dalam proyek akhir ini, penulis mencoba menerapkan ilmu pengetahuan yang sudah didapat selama 3 tahun menempuh pendidikan di Politeknik Manufaktur Negeri Bangka Belitung. Selain itu, penulis juga menjadikan makalah-makalah proyek akhir mahasiswa Politeknik Manufaktur Negeri Bangka Belitung pada tahun-tahun sebelumnya untuk dijadikan informasi mengenai data-data pendukung dari proyek akhir ini.

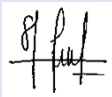
Dalam penyusunan makalah proyek akhir ini, penulis mendapatkan bantuan dan bimbingan dari berbagai pihak yang membuat proses pembuatan makalah ini dapat diselesaikan dengan baik. Oleh sebab itu, penulis ingin mengucapkan terima kasih kepada:

1. Orangtua dan keluarga tercinta yang selalu memberikan dukungan baik secara moral maupun materil sehingga penulis dapat memberikan hasil yang terbaik.
2. Bapak I Made Andik Setiawan, M.Eng., Ph.D. selaku Direktur Politeknik Manufaktur Negeri Bangka Belitung sekaligus dosen pembimbing 2 dalam proyek akhir ini.
3. Bapak Eko Sulistyo, M.T. selaku Wakil Direktur III Politeknik Manufaktur Negeri Bangka Belitung sekaligus dosen pembimbing 1 dalam proyek akhir ini.

4. Bapak Zanu Saputra M.Tr.T selaku Ketua Jurusan Teknik Elektro dan Informatika Politeknik Manufaktur Negeri Bangka Belitung.
5. Bapak Ocsirendi, M.T. selaku Ketua Program Studi DIII Teknik Elektronika Politeknik Manufaktur Negeri Bangka Belitung.
6. Seluruh tenaga pendidik dan kependidikan Politeknik Manufaktur Negeri Bangka Belitung.
7. Teman-teman mahasiswa Politeknik Manufaktur Negeri Bangka Belitung yang telah membantu dalam penyelesaian proyek akhir ini.
8. Seluruh pihak yang ikut terlibat baik secara langsung maupun tidak langsung dalam penyelesaian proyek akhir ini yang tidak bisa disebutkan satu persatu.

Dalam penulisan laporan ini, penulis menyadari masih terdapat kekurangan dikarenakan masih terbatasnya pengetahuan dan kemampuan penulis. Oleh karena itu, untuk perbaikan dan pengembangan penulisan makalah ini di masa yang akan dating, penulis mengharapkan kritik dan saran dari semua pihak. Penulis berharap makalah ini dapat bermanfaat bagi pihak yang berkepentingan pada khususnya dan bagi perkembangan ilmu pengetahuan dan teknologi.

Sungailiat, 24 Juli 2023



Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Mobil listrik adalah jenis kendaraan yang menggunakan energi listrik sebagai sumber utama untuk menggerakkan mesin listriknya. Energi listrik yang digunakan oleh mobil ini disimpan dalam baterai atau wadah penyimpanan energi lainnya. [1]. Mobil listrik banyak digemari terbukti dengan persentase pencapaian penggunaan mobil listrik pada tahun 2020 saja sebesar 0,15% yaitu 230/150.000 unit [2]. Namun, dibalik kepraktisan mobil listrik, masih banyak kecelakaan yang melibatkan mobil tersebut di mana pada mobil listrik Tesla saja sudah melibatkan 356 kecelakaan dari tahun 2013 sampai 2023 [3]. Sebenarnya, sulit untuk memprediksi kapan dan di mana kecelakaan akan terjadi [4]. Dapat dipahami bahwa faktor manusia memiliki peran besar dalam terjadinya kecelakaan di jalan raya. Dengan menggunakan metode ANP, diperoleh urutan prioritas dengan bobot nilai, di mana faktor manusia memiliki bobot sebesar 63,05%, faktor kendaraan memiliki bobot sebesar 20,15%, dan faktor jalan memiliki bobot sebesar 16,80% [5]. Kecelakaan lalu lintas disebabkan oleh faktor manusia dapat terjadi karena beberapa alasan, seperti pengemudi yang tidak mengemudi dengan hati-hati, kurangnya pengetahuan pengemudi tentang peraturan lalu lintas yang berlaku, kurangnya keterampilan pengemudi dalam mengemudi, kurangnya kesadaran pengendara, dan juga seringkali kecelakaan lalu lintas terjadi karena pengemudi mengemudi dalam kondisi mengantuk, dalam keadaan mabuk, atau sambil menggunakan perangkat gawai [6]. Kecelakaan memang tidak bisa diprediksi, saat kecelakaan sudah terjadi maka selalu menyebabkan kerugian baik materil maupun immaterial [7].

Pada mobil listrik saat ini, kebanyakan pengereman masih dilakukan secara manual dengan perkiraan pengemudi saja [8]. Di samping itu, pada mobil listrik saat ini masih terbatasnya pemberitahuan mengenai jarak antara mobil dan objek yang terletak di depan, di sisi kiri, di sisi kanan, dan di belakang mobil listrik

tersebut. Padahal dengan mengetahui jarak antara mobil dengan objek tersebut pengemudi dapat melakukan parkir atau berkendara dengan aman sehingga kecelakaan dapat dihindari.

Oleh karena itu, untuk mencegah terjadinya kecelakaan terutama pada mobil listrik yang sangat merugikan jika kecelakaan terjadi diperlukan sebuah sistem pengontrolan kecepatan mobil listrik yang akan mengurangi kecepatan dengan melakukan pengereman secara otomatis apabila di depan mobil terdapat objek yang sudah mendekati setting jarak. Sudah ada penelitian sebelumnya yang telah menguji pengontrolan kecepatan mobil listrik. Salah satu penelitian tersebut dilakukan oleh Hendrik Iswanto, yang menggunakan teknologi komunikasi dalam pengendalian kecepatan mobil listrik. Metode tersebut melibatkan pengendalian perangkat keras dan pemantauan data melalui telepon seluler yang menggunakan Fuzzy Logic Controller (FLC) [9]. Sementara itu pada penelitian Afwan Ramadhan, dkk. yang merancang kendali kecepatan mobil listrik dengan metode PID menggunakan remote control [10], sedangkan pada penelitian Dwi Aji Saputra, dkk. merancang simulasi pengendali PWM kecepatan dengan Arduino menggunakan sensor ultrasonic untuk mendeteksi jarak [11]. Namun, penelitian yang telah dilakukan di atas hanya mengontrol kecepatan mobil listrik saat ada objek di depannya saja. Belum ada penelitian yang me-monitoring objek di kanan, kiri, dan belakang mobil listrik untuk memberikan peringatan kepada pengemudi jika objek terlalu dekat.

Berdasarkan penelitian yang telah dilakukan sebelumnya, proyek akhir ini akan mengembangkan sebuah perangkat pengendalian kecepatan dan jarak untuk mobil listrik. Perangkat tersebut akan mengatur kecepatan mobil secara otomatis dan melakukan pengereman otomatis jika terdapat halangan di depannya. Pengereman ini dilakukan dengan pengaturan PWM oleh *Fuzzy Logic Controller*. Sedangkan pada kiri, kanan dan belakang mobil menggunakan sensor ultrasonik untuk mengukur jarak terhadap obyek yang datanya akan ditampilkan di LCD sebagai *monitoring* pengemudi terhadap jarak mobil dengan objek. Jika jarak mobil dengan objek terlalu dekat, maka akan diberikan suara peringatan buzzer dan LCD akan menampilkan jarak mobil terhadap objek. Dari konsep tersebut,

maka pada proyek akhir ini, diajukan judul " *Prototype* Pengaturan Kecepatan dan Kendali Jarak Otomatis pada Mobil Listrik terhadap Bahaya Kecelakaan Menggunakan Metode *Fuzzy Logic Controller* (FLC)". Harapannya, keberadaan perangkat ini akan mengurangi insiden tabrakan antara kendaraan dan objek lainnya, sehingga dapat mencegah terjadinya kecelakaan.

1.2 Rumusan Masalah

Rumusan masalah yang dibuat dalam proyek akhir ini adalah:

1. Bagaimana cara mengontrol kecepatan dan pengereman mobil listrik jika terdapat objek di depan mobil listrik tersebut menggunakan sensor ultrasonik dengan metode *Fuzzy Logic Controller (FLC)* ?
2. Bagaimana cara membuat kontrol jarak mobil dengan objek yang ada di kiri, kanan dan belakang dan menampilkannya ke LCD?

1.3 Batasan Masalah

Untuk memperjelas arah pembahasan dari proyek akhir ini, maka diberikan batasan masalah sebagai berikut.

1. Sistem pengaturan kecepatan dan kendali jarak otomatis pada mobil listrik dibuat dalam bentuk *prototype* menggunakan mobil *remote control* dengan ukuran 45 cm x 20 cm x 12 cm.
2. *Delay* untuk pembacaan sensor sedikit lama.
3. Menggunakan metode *fuzzy logic* Mamdani untuk menentukan nilai PWM.

1.4 Tujuan

Tujuan yang ingin dicapai dalam pembuatan sistem pengontrolan kecepatan dan kendali jarak mobil listrik ini adalah sebagai berikut.

1. Mengontrol kecepatan dan pengereman mobil listrik jika terdapat objek di depan mobil listrik tersebut menggunakan sensor ultrasonik dengan metode *Fuzzy Logic Controller (FLC)*.
2. Membuat kontrol jarak mobil dengan objek yang ada di kiri, kanan dan belakang dan menampilkannya ke LCD.

BAB II
LANDASAN TEORI

2.1 Mobil Listrik

Mobil listrik adalah jenis kendaraan yang menggunakan motor listrik DC sebagai sumber tenaga untuk bergerak, dengan memanfaatkan energi listrik. Energi listrik yang diperlukan untuk menggerakkan motor listrik tersebut disimpan dalam baterai atau dalam wadah penyimpanan energi lainnya. Dibandingkan dengan mobil yang memanfaatkan bahan bakar minyak (BBM), mobil listrik memiliki beberapa kelebihan, yaitu tidak menyebabkan polusi udara dan mengurangi efek rumah kaca. Hal ini dikarenakan mobil listrik tidak menggunakan sumber energi dari bahan bakar fosil sebagai penggeraknya [12].

2.2 Batas Jarak Aman Berkendara

Dalam berkendara, hal terpenting yang dilakukan untuk keselamatan adalah dengan menjaga jarak aman. Jarak aman ini maksudnya adalah jarak yang harus dijaga atau jarak minimal antara satu kendaraan dengan kendaraan di depan atau dibelakangnya. Untuk menjaga keselamatan, pengendara harus berkendara dalam posisi minimum setidaknya dalam dua detik. Detik pertama digunakan untuk reaksi pengendara dan detik kedua digunakan untuk proses pengereman. Dalam penentuan jarak aman ini sangat dipengaruhi oleh faktor kondisi dan situasi jalan. Hal ini dikarenakan, jalan dalam kondisi licin dan kering akan menghasilkan efek pengereman yang berbeda [13]. Tabel 2.1 berikut ini menampilkan jarak yang umumnya dianggap aman dalam situasi berkendara.

Tabel 2. 1 Jarak Aman Berkendara [13]

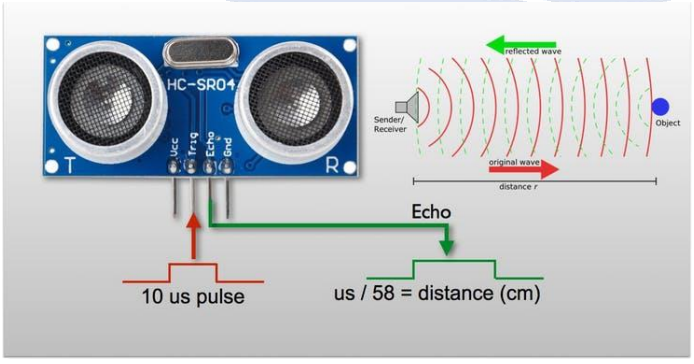
Kecepatan	Jarak Minimal	Jarak Aman
30 km/jam	15 m	30 m
40 km/jam	20 m	40 m
50 km/jam	25 m	50 m

Kecepatan	Jarak Minimal	Jarak Aman
60 km/jam	40 m	60 m
70 km/jam	50 m	70 m
80 km/jam	60 m	80 m
90 km/jam	70 m	90 m
100 km/jam	80 m	100 m
120 km/jam	100 m	120 m

2.3 Kendali Jarak Mobil Lisrik

Kendali jarak pada mobil listrik diperlukan agar mobil listrik dapat bergerak tanpa melewati batas aman berkendara agar menghindari kecelakaan. Untuk mengetahui jarak antar mobil listrik dengan objek di depannya, diperlukanlah sebuah sensor untuk mendeteksi jarak. Dalam *prototype* ini, sensor ultrasonik digunakan untuk mendeteksi objek yang berada di depan, di sisi kanan, di sisi kiri, dan di belakang mobil.

Sensor ultrasonik HC-SR04 adalah sebuah perangkat yang dirancang untuk mengukur jarak suatu objek. Perangkat ini memiliki kemampuan untuk mendeteksi objek dalam rentang 2 cm hingga 450 cm (0,78 inci hingga 14 inci) [14]. Proses kerja dari sensor ultrasonik HC-SR04 dapat diilustrasikan melalui Gambar 2.1 berikut ini.



Gambar 2. 1 Proses Kerja Sensor Ultrasonik [14]

Sensor ultrasonik HC-SR04 memiliki empat pin yang terdiri dari Vcc, Gnd, Trigger, dan Echo. Setiap pin memiliki peran dan fungsi yang berbeda. Pin Vcc

digunakan sebagai sumber listrik positif, sedangkan pin Gnd berfungsi sebagai ground [15]. Pin Trigger berperan sebagai pemicu untuk menghasilkan sinyal dari sensor, sementara pin Echo berfungsi untuk menangkap sinyal pantulan dari objek di depannya.

Sensor ultrasonik HC-SR04 dilengkapi dengan sepasang transduser ultrasonik yang berfungsi sebagai pemancar gelombang dan penerima pantulan gelombang. Prinsip kerja sensor ultrasonik ini dimulai dengan mengirimkan pulsa ultrasonik sekitar 40 KHz (sesuai isolator), yang kemudian memantulkan pulsa echo kembali. *Receiver piezoelektrik* akan menerima pantulan pulsa tersebut untuk menghitung waktu pengiriman dan waktu penerimaan pulsa echo yang dikembalikan, sebagaimana dijelaskan dalam Gambar 2.1. Hasil kalkulasi ini kemudian dijadikan sebagai nilai jarak yang diukur. Ukuran objek yang dapat dideteksi oleh sensor ultrasonik ini dalam ukuran apapun selama objek tersebut dapat memantulkan gelombang ultrasonik kembali ke sensor [14].

Sensor ultrasonik sudah banyak digunakan dalam beberapa penelitian karena keakuratannya dan kemampuan deteksi yang lumayan jauh. Pada penelitian Akbar Sugih Miftahul Huda, dkk. Yang melakukan pengujian sensor ultrasonik HC-SR04 dengan mengambil data yang bisa membaca jarak mulai dari 3 cm- 440 cm. Berdasarkan hasil penelitian tersebut, didapatkan hasil bahwa sensor ultrasonik mengalami sedikit kesalahan dalam pembacaan jarak setelah mencapai jarak lebih dari 60 cm [16].

Penelitian yang dilakukan oleh Alimuddin Mappa dkk. menggunakan sensor jarak ultrasonik HC-SR04 untuk mengembangkan sistem pengontrolan kecepatan dan pengereman. Data jarak yang diperoleh dari sensor tersebut ditampilkan pada layar LCD. Dalam penelitian ini, dilakukan pengujian akurasi sensor jarak ultrasonik HC-SR04 dengan menghitung tingkat kesalahannya pada jarak tertentu. Tabel 2.2 berikut ini menyajikan hasil uji sensor ultrasonik sebagai pendeteksi jarak, yang dibandingkan dengan jarak yang sebenarnya serta tingkat kesalahannya [17].

Tabel 2. 2 Hasil Perhitungan Persentase *Error* Pengukuran Jarak [17]

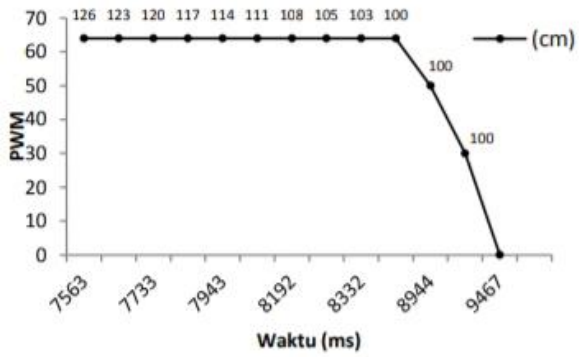
No	Jarak Seharusnya (cm)	Jarak Terukur (cm)	*Error*
1	10	10	0 %
2	22,1	22	0,45 %
3	40,2	40	0,5 %
4	65,2	65	0,31 %
5	80,5	80	0,62 %
6	101	100	0,99 %
7	121	121	0
8	151	150	0,66 %

Dari data tersebut diketahui bahwa jarak yang terukur dengan jarak seharusnya tidak jauh berbeda, sehingga nilai *error* yang dihasilkan kecil. Penelitian tersebut juga mengindikasikan bahwa sensor ultrasonik HC-SR04 beroperasi dengan kinerja yang baik dan tingkat akurasi yang tinggi.

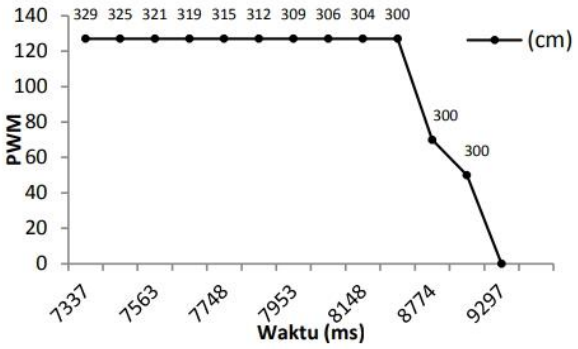
2.4 Pengaturan Kecepatan Mobil Listrik

Untuk mengatur kecepatan mobil listrik dilakukan dengan pengaturan atau pengendalian PWM. Proses pengereman dengan pengendalian PWM ini agar mobil listrik dapat berhenti secara otomatis sudah pernah dilakukan pada penelitian yang dilakukan oleh Dwi Aji Saputra, dkk. Penelitian ini mensimulasikan pemrograman pengendali PWM kecepatan dengan mikrokontroler pada mobil listrik. Dalam penelitian ini, dirancang suatu sistem pengatur kecepatan otomatis untuk mobil RC. Sistem ini bekerja dengan cara mengenali saat mobil RC bergerak dengan kecepatan tertentu dan mendeteksi keberadaan penghalang di depannya. Kemudian, melalui penggunaan mikrokontroler dan PWM, kecepatan mobil secara otomatis dan bertahap akan dikurangi. Pengurangan kecepatan akan terus dilakukan hingga mobil RC berhenti saat jarak antara mobil dan objek penghalang semakin dekat [11].

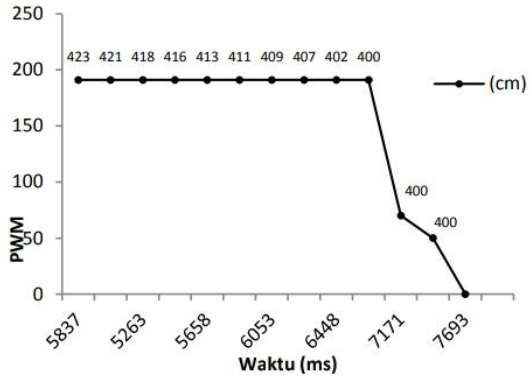
Penelitian ini dilakukan dengan tujuan untuk menguji data yang optimal agar mobil listrik dapat berhenti dengan sempurna. Dalam rangkaian pengujian, beberapa kecepatan dan jarak telah ditentukan, dan hasilnya menghasilkan empat kombinasi yang relevan dalam penelitian ini. Salah satunya adalah kombinasi pertama, di mana dengan menggunakan PWM 64 (25%), mobil berhasil berhenti dengan sempurna pada jarak 100 cm. Informasi lebih lanjut dapat ditemukan dalam Gambar 2.2. Kombinasi kedua pada PWM 127 (50%) mobil dapat berhenti sempurna pada jarak 300 cm dapat dilihat prosesnya pada Gambar 2.3. Kombinasi ketiga di mana dengan menggunakan PWM 191 (75%), mobil dapat berhenti dengan sempurna pada jarak 400 cm. Proses ini dapat dilihat dalam Gambar 2.4. Selanjutnya, terdapat pula kombinasi keempat di mana dengan menggunakan PWM 255 (100%), mobil dapat berhenti dengan sempurna pada jarak 500 cm. Proses ini dapat dilihat dalam Gambar 2.5.



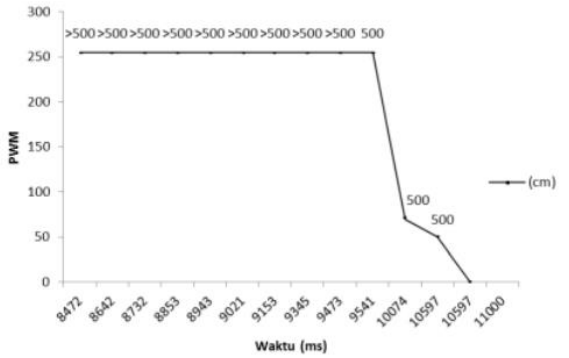
Gambar 2. 2 Waktu yang Diperlukan untuk Berhenti Sempurna pada Kombinasi Pertama



Gambar 2. 3 Waktu yang Diperlukan untuk Berhenti Sempurna pada Kombinasi Kedua



Gambar 2. 4 Waktu yang Diperlukan untuk Berhenti Sempurna pada Kombinasi Ketiga



Gambar 2. 5 Waktu yang Diperlukan untuk Berhenti Sempurna pada Kombinasi Keempat

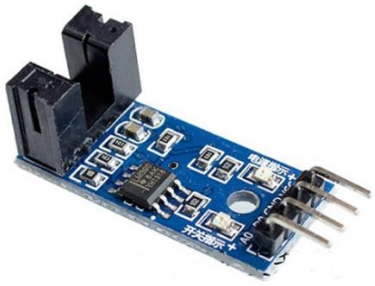
Dari grafik di atas, waktu dan jarak yang diperlukan untuk mobil listrik berhenti sempurna pada 4 kombinasi PWM tertera pada Tabel 2.3 berikut.

Tabel 2. 3 Perbandingan Waktu Mulai Pengereman

PWM %	Waktu Mulai Pengereman (millisekon)	Jarak (cm)
64	8.700	100
127	8.500	300
191	6.900	400
255	9.700	500

Berdasarkan data tersebut, kesimpulan yang dapat diambil bahwa sistem pengaturan kecepatan pada mobil listrik tidak mampu menghentikan kecepatan secara instan atau tiba-tiba. Ketika diterapkan pada mobil listrik yang sebenarnya, sistem pengendali kecepatan ini berfungsi mirip dengan penginjak pedal rem secara perlahan. Sistem ini akan beroperasi ketika sensor ultrasonik mendeteksi adanya objek penghalang [11]. Dari penelitian tersebut, pada proyek akhir ini untuk mengatur kecepatan mobil listrik dengan cara mengatur atau mengendalikan PWM dapat digunakan.

Kemudian, untuk mendeteksi kecepatan pada mobil listrik digunakan sensor kecepatan *Opto-Interrupter*. Sensor kecepatan *Opto-Interrupter* adalah salah satu komponen elektronika yang digunakan sebagai penghubung yang berdasarkan cahaya optik. Sensor kecepatan *Opto-Interrupter* ini terbuat dari bahan semi-konduktor dan terdiri dari LED (*Light Emitting Diode)* infra merah dan phototransitor. Sensor *Opto-Interrupter* ini mempunyai 2 bagian utama yaitu *transmitter* dan *receiver*. Dimana LED pada sensor ini berfungsi sebagai *transmitter* yaitu untuk menstandarkan sinyal keluaran dari sensor, serta phototransistor berfungsi sebagai *receiver* yaitu untuk menangkap sinyal dari *transmitter* [18]. Gambar 2.6 berikut ini adalah bentuk fisik dari sensor kecepatan *Opto-Interrupter*.

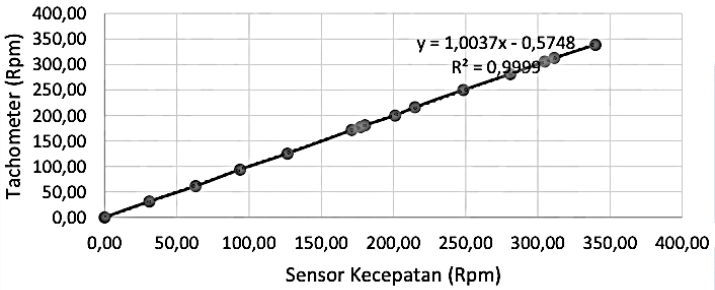


Gambar 2. 6 Sensor *Opto-Interrupter* [19]

Sensor kecepatan *Opto-Interrupter* ini bekerja apabila ada arus yang mengalir melalui LED dan menyebabkan LED memancarkan sinyal cahaya. Sinyak cahaya inilah yang akan ditangkap oleh phototransistor. Apabila sinyal cahaya yang dikirim oleh LED dapat diterima oleh phototransistor, maka

indikator sensor akan menyala (HIGH). Namun, apabila saat sinyal cahaya yang dikirim oleh LED tidak diterima oleh phototransistor, maka indikator sensor akan padam (LOW) [18].

Penggunaan sensor kecepatan *Opto-Interrupter* sudah pernah dilakukan pada beberapa penelitian karena sensor ini dapat mendeteksi kecepatan motor secara akurat. Pada penelitian oleh Hendrik Iswanto dilakukan pengujian untuk membandingkan pembacaan sensor kecepatan *Opto-Interrupter* dengan *tachometer* digital untuk mengetahui tingkat keakuratan terhadap pembacaan kecepatan motor DC. Pada Gambar 2.7 merupakan hasil 15 kali pengujian terhadap sensor kecepatan *Opto-Interrupter* dengan *tachometer* sebagai berikut.



Gambar 2. 7 Grafik Kalibrasi Sensor Kecepatan

Dari data tersebut diperoleh nilai persamaan y = 1,0037x – 0,5748 dan nilai regresi sebesar 0,9999. Persamaan tersebut digunakan untuk kalibrasi sensor kecepatan yang dimasukan ke dalam program Arduino DUE agar diperoleh hasil pengukuran sensor dengan persentase *error* sekecil mungkin. Nilai regresi menyatakan tingkat kemiripan atau ketelitian sensor kecepatan mendekati 1 sehingga sensor kecepatan *Opto-Interrupter* layak digunakan untuk penelitian [9].

2.4.1 *Pulse Width Modulation* (PWM)

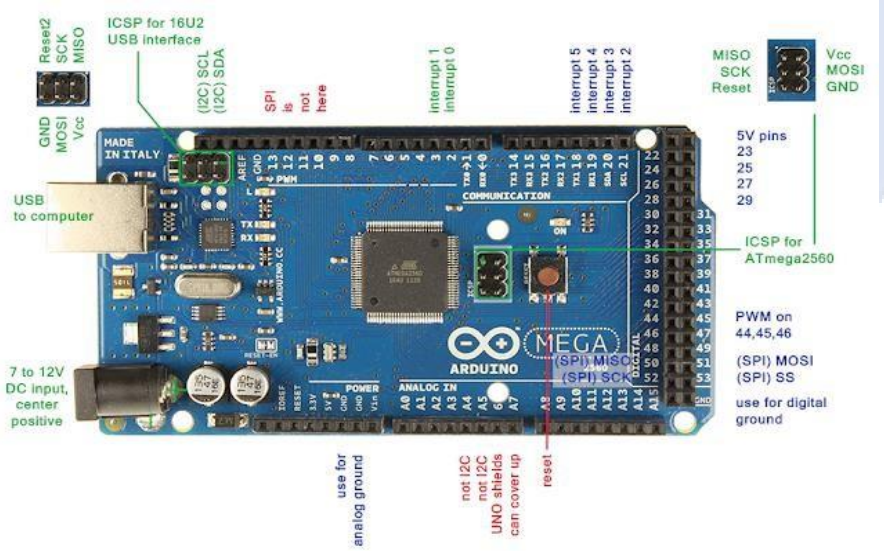
Pulse Width Modulation (PWM) adalah metode yang digunakan untuk mengatur lebar sinyal (dinyatakan dalam pulsa) dalam satu periode. PWM memiliki berbagai aplikasi, seperti pengendalian motor DC, pengendalian motor servo, mengatur tingkat kecerahan LED, dan banyak lagi. Sinyal PWM memiliki lebar pulsa yang dapat berbeda-beda, tetapi amplitudo dan frekuensi tetap. PWM

digunakan untuk menghasilkan sinyal analog dari perangkat digital. PWM dapat dibangkitkan melalui dua metode, yaitu metode analog dan metode digital [20] .

2.5 Pengontrolan Sistem Menggunakan Arduino Mega 2560

Pengontrolan sistem ini menggunakan mikrokontroler Arduino Mega 2560. Sebenarnya, banyak jenis Arduino yang bisa digunakan dalam pengontrolan secara umum, seperti Arduino Uno, Arduino Nano, dan lain sebagainya. Namun, Arduino Mega 2560 ini dipilih karena Arduino ini memiliki lebih banyak pin *input* dan *output*-nya. Arduino Mega 2560 memiliki total 54 pin *input/output* yaitu pada pin 0-53. Dimana, pin *input* mikrokontroler ini ada 16 *analog input* yaitu A0 sampai dengan A15, 15 *output Pulse Wide Modulation* (PWM) yaitu pin 2-13 dan pin 44-46, serta 4 UART atau port serial perangkat keras. Mikrokontroler ini juga dilengkapi osilator kristal 16 Mhz, *header* ICSP, dan tombol reset [21]. Gambar 2.8 berikut ini menampilkan tampilan fisik dari Arduino Mega 2560 yang dapat diamati.

.



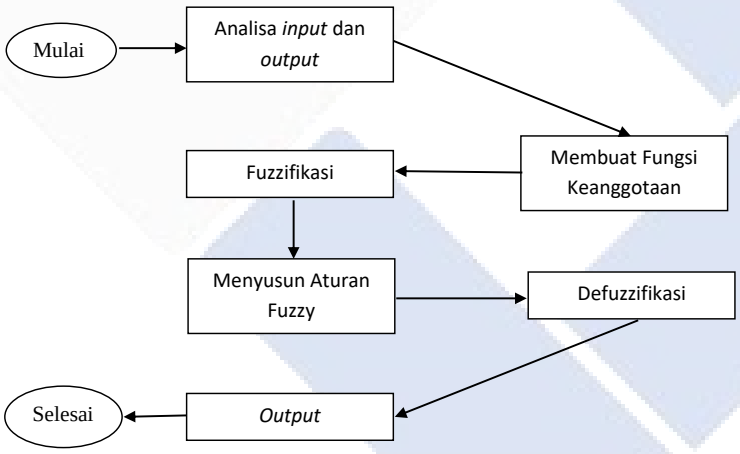
Gambar 2. 8 Arduino Mega 2560 [22]

2.6 *Fuzzy Logic Controller* (FLC)

Pada prototipe pengaturan kecepatan dan kendali jarak mobil listrik ini, digunakan metode Fuzzy Logic Controller (FLC). Metode Fuzzy Logic Controller (FLC) telah banyak diterapkan dalam berbagai sistem dinamik, mulai dari yang

sederhana hingga yang kompleks. Keunggulan Fuzzy Logic Controller (FLC) adalah tidak memerlukan model matematis dari sistem yang akan dikendalikan. Mekanisme pengambilan keputusan diimplementasikan dalam pengendali berdasarkan aturan dasar selama proses pengendalian berlangsung [23]. Selain itu, Fuzzy Logic Controller (FLC) juga dikategorikan sebagai salah satu bentuk kontrol cerdas atau intelligent control. Metode Fuzzy Logic Controller (FLC) digunakan untuk menyelesaikan masalah pada sistem yang memiliki kriteria kompleks. Dalam metode ini, tidak ada batas yang tegas, melainkan terdapat transisi antara satu himpunan ke himpunan lainnya [24].

Logika *Fuzzy Logic Controller* (FLC) merupakan sebuah logika yang nilainya di antara 0 dan 1, disebut juga sebagai logika yang memiliki presentasi nilai samar, ketidakpastian, dan kebenaran sebagian. Gambar 2.9 berikut ini menampilkan diagram *Fuzzy Logic Controller* (FLC) yang dapat diamati.



Gambar 2. 9 Blok Diagram *Fuzzy Logic Controller* (FLC)

Penerapan metode *Fuzzy Logic Controller* (FLC) sudah banyak digunakan pada beberapa penelitian. Pada penelitian oleh Atha'llah dkk. yang membuat *prototype* sistem pengereman dengan menerapkan metode *Fuzzy Logic Controller* (FLC) ini. *Prototype* ini akan melakukan pengereman pada saat kecepatan dan jarak tertentu. Tabel 2.4 berikut ini menunjukkan hasil pengujian dari penelitian ini.

Tabel 2. 4 Hasil Pengujian Simulasi Matlab dan *Prototype* [25]

Pengujian Ke	Masukan		Keluaran (Tingkat Pengereman)		*Error %*
	Kecepatan	Jarak	Simulasi	*Prototype*	
1	19	5	28	20	0.8
2	21	18	43.2	34	1.4
3	26	37	82.8	81	0.3
4	20	30	60	43	1.7
5	16	26	42	27	2.1
6	17	34	48	28	0.6
7	34	12	76.8	96	1.3
8	19	40	60	38	2.8
9	34	40	100	100	0
10	22	8	37.6	38	0.2
11	20	15	40	26	5
12	30	10	60	81	2.2
13	18	26	47.2	34	1.9
14	25	5	40	60	6.7
15	23	37	72	64	0.8
Rata-Rata					2.5

Dari penelitian di atas terlihat bahwa *prototype* mampu mengendalikan pengereman dengan nilai yang tidak jauh berbeda dengan nilai simulasi Matlab dengan metode FLC. Sehingga dapat disimpulkan pada penelitian tersebut bahwa sistem *Fuzzy Logic Controller* (FLC) pada *prototype* ini berhasil mengimplementasikan hasil yang diinginkan. Hasil pengujian menunjukkan bahwa simulasi MATLAB dan *prototype* yang telah dirancang dapat mengendalikan pengereman dengan rata-rata kesalahan 2,5% dan sistem kendali pengereman ini dapat bekerja ketika *prototype* dikendalikan dengan kecepatan dan jarak, dengan rata-rata tingkat keberhasilan mencapai 95% [25]. Oleh sebab itu, pada proyek akhir ini metode *Fuzzy Logic Controller* (FLC) dapat digunakan.

2.6.1 Variabel dan Himpunan *Fuzzy*

Variabel *fuzzy* adalah variabel yang digunakan dalam sistem *fuzzy*. Contohnya termasuk umur, kecepatan, suhu, tinggi badan, penghasilan, dan lain sebagainya. Di sisi lain, himpunan fuzzy merupakan himpunan yang memiliki derajat keanggotaan dan menggunakan bahasa linguistik atau bahasa numerik sebagai representasinya. Contoh jika menggunakan bahasa linguistik, yaitu dengan variabel suhu maka himpunan *fuzzy*-nya dapat berupa panas, sedang, dan dingin. Kemudian jika variabel tinggi badan, maka himpunan *fuzzy*-nya pendek, sedang, dan tinggi. Berbeda dengan bahasa linguistik, bahasa numerik digunakan dengan angka. Misalnya, 1, 30, 50, dan seterusnya [26].

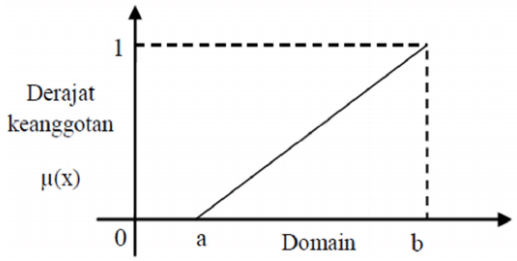
2.6.2 Semesta Pembicaraan *Fuzzy*

Semesta pembicaraan *fuzzy* merupakan kumpulan semua nilai yang digunakan dalam operasi dengan variabel *fuzzy*. Semesta pembicaraan ini terdiri dari himpunan bilangan *real* yang nilainya meningkat secara bertahap dari sisi kiri ke sisi kanan. Semesta pembicaraan mencakup nilai-nilai negatif dan positif, tanpa adanya batasan atau terhingga pada nilai maksimum. Contoh : Umur [0, ∞] atau Suhu [0, 100] [27].

2.6.3 Fungsi Keanggotaan *Fuzzy*

Fungsi keanggotaan adalah representasi grafik atau kurva yang menggambarkan pemetaan titik-titik data input ke dalam nilai-nilai keanggotaan yang berada dalam rentang antara 0 dan 1 [28].

2.6.3.1 Representasi Linear Naik



Gambar 2. 10 Representasi Linear Naik [28]

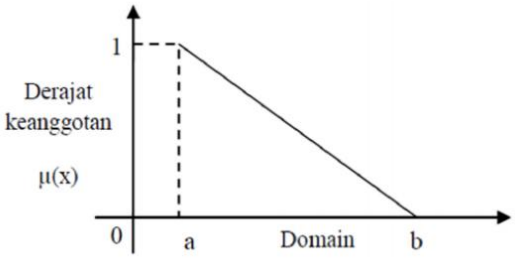
Fungsi Keanggotaan :

μ [x] = 0; x ≤ a

(x-a)(x-b); a ≤ x ≤ b

1; x ≥ b

2.6.3.2 Representasi Linear Turun



Gambar 2. 11 Representasi Linear Turun [28]

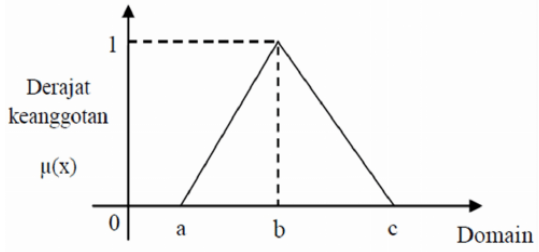
Fungsi Keanggotaan :

μ [x] = 0; x ≥ b

$\frac{b-x}{b-a}$; a ≤ x ≤ b

1; x ≤ a

2.6.3.3 Representasi Kurva Segitiga



Gambar 2. 12 Representasi Kurva Segitiga [28]

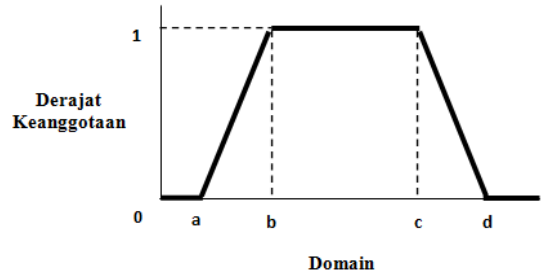
Fungsi Keanggotaan :

μ [x] = 0; x ≤ a atau x ≥ c

$\frac{x-a}{b-a}$; a ≤ x ≤ b

$\frac{c-x}{c-b}$; b ≤ x ≤ c

2.6.3.3 Representasi Kurva Trapesium



Gambar 2. 13 Representasi Kurva Trapesium [28]

Fungsi Keanggotaan :

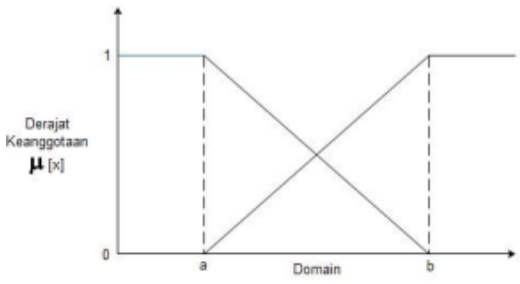
μ [x] = 0; x ≤ a atau x ≥ d

$\frac{x-a}{b-a}$; a ≤ x ≤ b

$\frac{d-x}{d-c}$; c ≤ x ≤ d

1; b ≤ x ≤ c

2.6.3.4 Representasi Kurva Bahu Kiri atau Bahu Kanan



Gambar 2. 14 Representasi Kurva Bahu Kiri atau Bahu Kanan [28]

Fungsi Keanggotaan :

μ [x] = 0; x ≤ a atau x ≥ b

$\frac{b-x}{b-a}$; a ≤ x ≤ b

$\frac{x-a}{b-a}$; a ≤ x ≤ b

1; x ≤ a atau x ≥ b

2.6.4 Metode *Fuzzy*

Terdapat tiga metode yang dapat digunakan dalam *fuzzy logic controller*, yaitu metode Tsukamoto, Mamdani, dan Sugeno. Ketiga metode ini telah terbukti efektif ketika diimplementasikan dalam sistem. Namun, pada proyek akhir ini digunakan metode mamdani dalam pelaksanaannya. Hal ini dikarenakan metode mamdani mudah diaplikasikan dan tidak terlalu banyak infomasi awal dari sistem serta dikenal sederhana sehingga mudah untuk sistem yang memiliki sifat non liniear. Metode mamdani juga dapat mempermudah pekerjaan karena penarikan kesimpulan pada metode ini sesuai dengan naluri manusia [29].

2.6.5 Prosedur Metode *Fuzzy* Mamdani

Dalam metode ini, terdapat beberapa tahapan yang perlu dilakukan hingga mencapai kesimpulan. Tahapan-tahapan tersebut meliputi pembentukan himpunan *fuzzy*, aplikasi fungsi implikasi, komposisi aturan, dan *defuzzifikasi*.

2.6.5.1 Pembentukan Himpunan *Fuzzy*

Fuzzifikasi yang juga dikenal sebagai pembentukan himpunan *fuzzy*, adalah proses mengkonversi nilai-nilai *real* menjadi nilai-nilai *fuzzy*. Tujuan dari proses ini adalah agar kendali logika *fuzzy* dapat memproses nilai-nilai tersebut. Nilai-nilai yang dinyatakan dalam bentuk data *fuzzy* akan terdiri dari dua aspek, yaitu himpunan *fuzzy* dan nilai keanggotaannya. Pada metode Mamdani, tipe fungsi keanggotan yang dapat digunakan yaitu, tipe trapezium yang digunakan jika ada 2 titik himpunan, tipe segitiga yang digunakan jika hanya ada 1 titik himpunan, dan tipe bahu kiri atau kanan yang digunakan jika diawali dan diakhiri variabel *fuzzy* [30] .

2.6.5.2 Aplikasi Fungsi Implikasi

Tahapan ini dilakukan penyusunan basis aturan berupa implikasi *fuzzy* yang menyatakan relasi antara variabel input dengan variabel output. Bentuk dari fungsi implikasi ini yaitu, *IF* x is *A THEN* y is *B*, dengan x dan y adalah saklar, serta *A ˆdan B* adalah himpunan *fuzzy*. Aturan *fuzzy* memiliki bentuk sebagai berikut.

IF (X_1 is A_1) AND (X_2 is A_2) AND … AND (X_n is A_n) *THEN* y is *B*

Dimana, banyaknya n ditentukan berdasarkan dari jumlah variabel *input fuzzy* yang digunakan [30] . Nilai keanggotaan berdasarkan aturan *fuzzy* yang dibentuk mengunakan fungsi implikasi Min karena menggunakan operator *AND*. Adapun fungsi implikasi Min dapat didefiniskan sebagai berikut.

$A \cap B \rightarrow \mu_{A \cap B} = \mu_A$ (x) $\wedge$ μ_B (x) = Min (μ_A (x), μ_B (x))

2.6.5.3 Komposisi Aturan

Komposisi aturan dimaksudkan untuk menggabungkan fungsi keanggotaan dari aturan aplikasi fungsi implikasi menggunakan metode Max. Proses penggabungan fungsi keanggotaan menggunakan metode Max dilakukan dengan perumusan sebagai berikut.

$\mu_{sf}\ (x_i)$= Max (($\mu_{sf}\ (x_i)$, $\mu_{kf}\ (x_i)$)

Dengan $\mu_{sf}\ (x_i)$ menyatakan nilai keanggotaan solusi *fuzzy* sampai aturan ke-i dan $\mu_{kf}\ (x_i)$ menyatakan nilai keanggotaan konsekuensi *fuzzy* aturan ke-i.

2.6.5.4 *Difuzzifikasi*

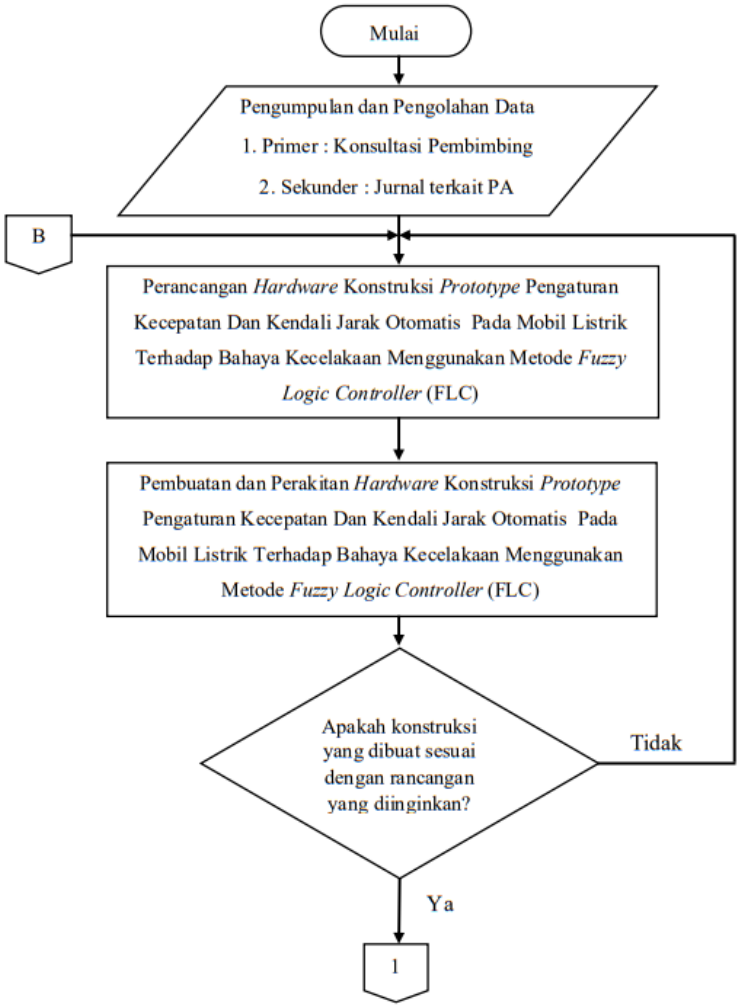
Tahap ini memiliki *input* berupa komposisi aturan *fuzzy* dan memiliki *output* bilangan *real* yang tegas. Atau dapat dikatakan tahapan ini merupakan proses mengubah hasil dari tahap inferensi menjadi *output* yang bernilai tegas menggunakan fungsi keanggotaan yang telah ditetapkan . Proses menentukan titik pusat daerah *fuzzy* dilakukan dengan menggunakan rumus :

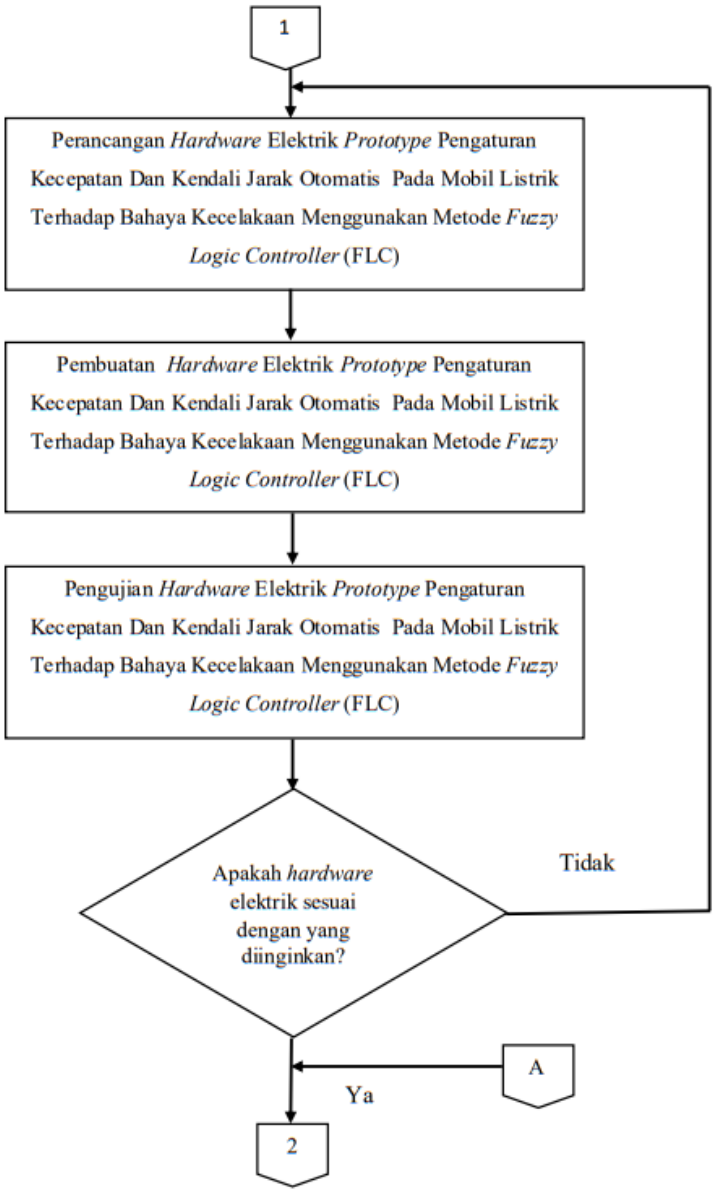
$$Z_O = \frac{\int_a^b z.\mu_{(z)} dz\ (momen)}{\int_a^b \mu_{(z)} dz\ (luas\ daeraḥ)}$$

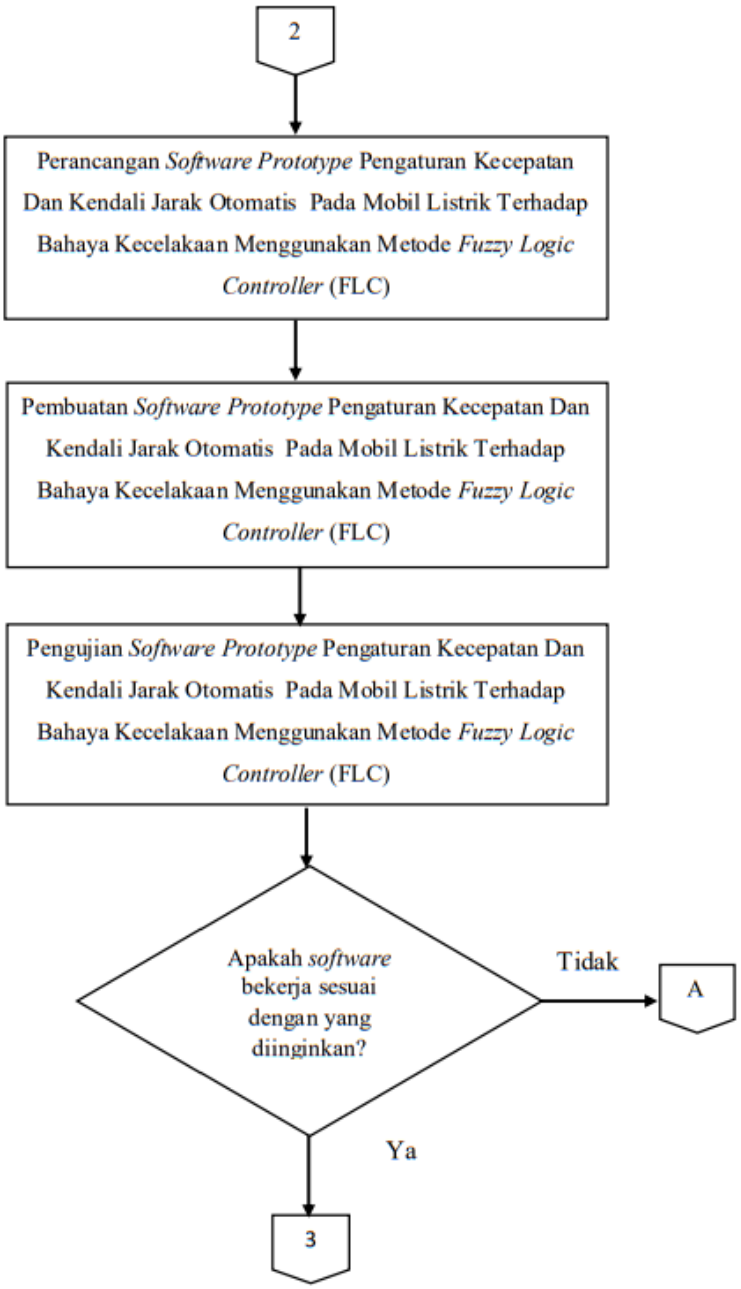
Dengan Z_O adalah nilai hasil defuzzifikasi dan $\mu_{(z)}$adalah derajat keanggotaan titik tersebut, sedangkan z adalah nilai domain ke-I [30].

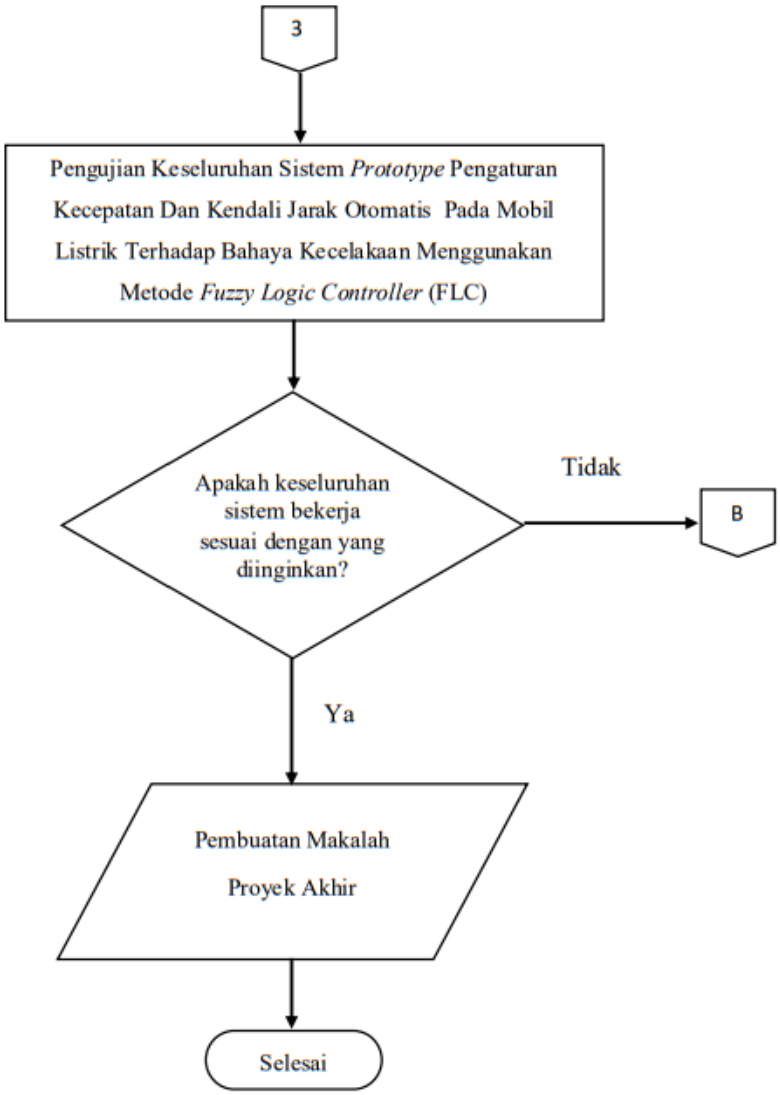
BAB III
METODE PELAKSANAAN

Tahapan yang dilakukan dalam menjalankan proyek akhir ini dirancang untuk membantu proses pembuatan proyek akhir. Rincian mengenai tahapan pelaksanaan proyek akhir ini dapat diilustrasikan melalui *flowchart* berikut.




1
Perancangan *Hardware* Elektrik *Prototype* Pengaturan Kecepatan Dan Kendali Jarak Otomatis Pada Mobil Listrik Terhadap Bahaya Kecelakaan Menggunakan Metode *Fuzzy Logic Controller* (FLC)
Pembuatan *Hardware* Elektrik *Prototype* Pengaturan Kecepatan Dan Kendali Jarak Otomatis Pada Mobil Listrik Terhadap Bahaya Kecelakaan Menggunakan Metode *Fuzzy Logic Controller* (FLC)
Pengujian *Hardware* Elektrik *Prototype* Pengaturan Kecepatan Dan Kendali Jarak Otomatis Pada Mobil Listrik Terhadap Bahaya Kecelakaan Menggunakan Metode *Fuzzy Logic Controller* (FLC)
Apakah *hardware* elektrik sesuai dengan yang diinginkan?
Tidak
Ya
A
2


2
Perancangan *Software Prototype* Pengaturan Kecepatan Dan Kendali Jarak Otomatis Pada Mobil Listrik Terhadap Bahaya Kecelakaan Menggunakan Metode *Fuzzy Logic Controller* (FLC)
Pembuatan *Software Prototype* Pengaturan Kecepatan Dan Kendali Jarak Otomatis Pada Mobil Listrik Terhadap Bahaya Kecelakaan Menggunakan Metode *Fuzzy Logic Controller* (FLC)
Pengujian *Software Prototype* Pengaturan Kecepatan Dan Kendali Jarak Otomatis Pada Mobil Listrik Terhadap Bahaya Kecelakaan Menggunakan Metode *Fuzzy Logic Controller* (FLC)
Apakah *software* bekerja sesuai dengan yang diinginkan?
Tidak
A
Ya
3



Gambar 3. 1 *Flowchart* Tahapan Pelaksanaan

3.1 Pengumpulan dan Pengolahan Data

Pengumpulan data dilakukan untuk mendapatkan informasi dan pengetahuan mengenai judul proyek akhir ini dan dapat dijadikan referensi dalam pembuatan alat ataupun pembuatan makalah proyek akhir. Dari referensi ini dapat diketahui gambaran mengenai pengaturan kecepatan dan kendali jarak otomatis pada mobil listrik menggunakan metode *Fuzzy Logic Controller* (FLC) yang

pernah dilakukan pada penelitian sebelumnya dan akan dikembangkan dalam proyek akhir ini.

Dalam pengumpulan data untuk dijadikan referensi ini, tahap yang dilakukan yaitu pengumpulan data primer dengan konsultasi kepada dosen pembimbing berkaitan dengan pengerjaan proyek akhir dan pengumpulan data sekunder yang dilakukan dengan mengumpulkan artikel-artikel yang berhubungan dengan proyek akhir.

Setelah data berhasil dikumpulkan, langkah selanjutnya yaitu merumuskan data tersebut untuk dilakukan pengembangan dan membuat sebuah ide baru dalam pelaksanaan pembuatan proyek akhir ini.

3.2 Rancangan *Hardware* dan *Software*

Perancangan *hardware* dilakukan dengan tujuan untuk menentukan bentuk fisik dari *prototype* pengaturan kecepatan dan kendali jarak otomatis serta penempatan sensor pada *prototype* tersebut. Sementara perancangan *software* dilakukan dengan tujuan untuk merancang sistem kontrol *prototype* melalui aplikasi *Bluetooth RC Car* pada *smartphone*.

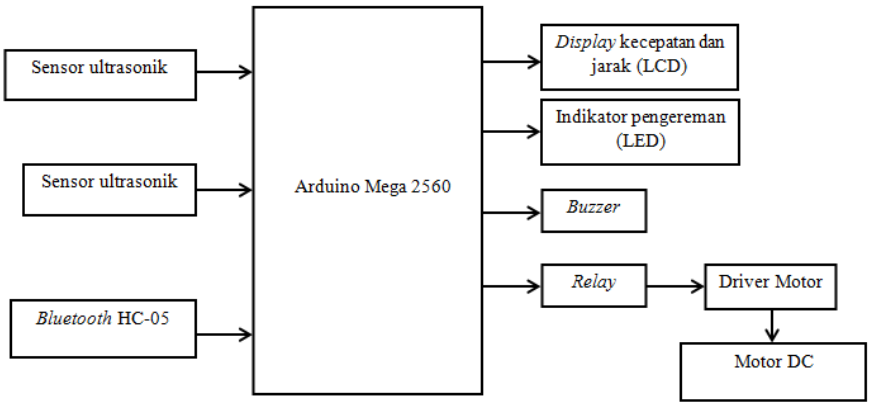
3.2.1 Rancangan *Hardware*

Rancangan *hardware* pada *prototype* pengaturan kecepatan dan kendali jarak otomatis pada mobil listrik ini terdiri dari *hardware* mekanik dan *hardware* elektrik. Pada perancangan *hardware* ini ditentukan komponen yang akan digunakan dan ukuran dari *prototype.* Perancangan bentuk fisik (konstruksi) dari mobil listrik ini didesain dengan menggunakan aplikasi *sketch up.*

Adapun rangkaian *hardware* secara mekanik yang akan dibuat sebagai berikut.

1. Dalam pembuatan *prototype* ini menggunakan mobil *remote control* dengan ukuran 45 cm x 20 cm x 12 cm.
2. Pemasangan sensor ultrasonik HC-SR04, sensor kecepatan *Opto-Interrupter*, mikrokontroller (Arduino Mega 2560), driver motor, LCD, LED, *buzzer*, dan *bluetooth* HC-05 pada mobil *remote control.*

Kemudian dalam tahap perancangan *hardware* secara elektrik yaitu membuat skema *wiring* kontrol menggunakan Arduino Mega 2560 untuk mengontrol *prototype* pengaturan kecepatan dan kendali jarak otomatis terhadap bahaya kecelakaan dengan menggunakan metode *Fuzzy Logic Controller* (FLC).



Gambar 3. 2 Rancangan *Hardware* Elektrik

3.2.2 Rancangan *Software*

Perancangan *software* yang dilakukan dalam proyek akhir ini untuk menentukan perangkat apa saja yang akan digunakan. Perancangan *software* yaitu pemrograman Arduino yang digunakan untuk mengontrol sistem secara keseluruhan untuk mengatur kecepatan dan kendali jarak pada mobil listrik menggunakan *software* Arduino IDE, membuat logika *fuzzy* dengan menggunakan aplikasi Matlab, serta mengontrol laju *prototype* mobil listrik dengan aplikasi *Bluetooth RC Car* melalui *smartphone.*

3.3 Pembuatan *Hardware* dan *Software*

Pembuatan *hardware* dan *software* adalah tahap untuk mewujudkan rancangan alat yang sudah dibuat pada tahap sebelumnya.

3.3.1 Pembuatan *Hardware*

Pada tahap konstruksi mekanik, dilakukan pemasangan komponen yang telah ditentukan sebelumnya ke dalam mobil *remote control* sesuai dengan desain

yang telah disusun sebelumnya. Berikut ini adalah langkah-langkah dalam proses pembuatan hardware secara mekanik:

1. Melubangi *body* mobil *remote control* dengan menyesuaikan bentuk sensor ultrasonik, LCD, dan LED yang akan dipasang.
2. Memasang sensor ultrasonik sebanyak dua buah di masing-masing sisi kanan, kiri, depan, dan belakang, LED sebanyak dua buah di belakang dan di depan, LCD di atas *body* mobil, serta Arduino Mega 2560, sensor *Opto-Interrupter*, driver motor, *bluetooth* HC-05, dan *buzzer* yang dipasang di bagian dalam mobil *remote control.*

3.3.2 Pembuatan *Software*

Software dibuat sesuai dengan rancangan yang sudah ditentukan sebelumnya. Berikut adalah langkah-langkah dalam proses pembuatan *software*.

1. Pembuatan logika *fuzzy* berupa fungsi keanggotaan *input* dan *output*, serta aturan *rules*-nya untuk mengatur kecepatan dan kendali jarak pada mobil listrik.
2. Pemrograman Arduino Mega untuk menampilkan data yang diperoleh dari sensor jarak (sensor ultrasonik) dan sensor kecepatan (sensor *Opto-Interrupter*) ke layar LCD.
3. Pemrograman Arduino Mega untuk mengaktifkan *buzzer*, driver motor L298N, motor DC dan LED sebagai tindakan pengontrolan kecepatan dan kendali jarak otomatis pada mobil.
4. Pemorgraman Arduino Mega untuk mengaktifkan *bluetooth* HC-05 agar dapat mengontrol mobil bergerak maju, mundur, ke kiri, dan ke kanan melalui aplikasi *Bluetooth RC Car* di *smartphone.*

3.4 Pengujian *Hardware* dan *Software*

Pengujian terhadap *hardware* dan *software* dalam *prototype* pengaturan kecepatan dan kendali jarak otomatis pada mobil listrik terhadap bahaya kecelakaan menggunakan metode *Fuzzy Logic Controller* (FLC) dilakukan untuk mengetahui apakah *prototype* pengaturan kecepatan dan kendali jarak otomatis

pada mobil listrik terhadap bahaya kecelakaan menggunakan metode *Fuzzy Logic Controller* (FLC) dapat bekerja sesuai dengan target yang diinginkan.

3.4.1 Pengujian *Hardware*

1. Pengujian dilakukan pada Arduino Mega dengan menggunakan sensor ultrasonik untuk memahami kemampuan deteksi sensor terhadap jarak objek.
2. Pengujian Arduino Mega dengan sensor *Opto-Interrupter* dengan tujuan agar mengetahui kemampuan deteksi sensor *Opto-Interrupter* terhadap kecepatan motor.
3. Pengujian Arduino Mega dengan *Bluetooth* HC-05 dengan tujuan agar mengetahui kemampuan kerja *bluetooth* yang dapat menggerakan mobil listrik yang dikontrol melalui *smartphone.*

3.4.2 Pengujian *Software*

Pengujian *software* pada proyek akhir ini dilakukan dengan menguji *output* yang dihasilkan pada perancangan dan pembuatan logika *fuzzy* apakah sesuai dengan *output* yang dihasilkan pada pemrograman Arduino.

3.5 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem merupakan pengujian secara keseluruhan terhadap *prototype* pengaturan kecepatan dan kendali jarak otomatis pada mobil listrik terhadap bahaya kecelakaan menggunakan metode *Fuzzy Logic Controller* (FLC) dengan tujuan agar mengetahui jika sistem dapat berfungsi dengan baik atau tidak. Pengujian keseluruhan sistem ini dilakukan dengan meletakan objek pada bagian kiri, kanan, dan belakang mobil untuk mengetahui kondisi *buzzer* yang akan aktif sebagai peringatan jika jarak sudah mencapai batas.

Selain itu, pengujian juga dilakukan dengan cara meletakan suatu objek di depan mobil mulai dari jarak sangat dekat sampai dengan sangat jauh sesuai inputan pada *Fuzzy Logic controller* (FLC), serta memberikan kecepatan mulai dari lambat sampai dengan cepat untuk melihat *output* PWM yang dihasilkan.

Selain itu, pengujian dilakukan dengan melihat *buzzer*, kondisi, dan LED dalam keadaan apa saat diberi *input* dengan nilai tertentu. Selanjutnya, data hasil pengujian tersebut akan dicatat dalam tabel pengujian.

3.6 Analisis Data

Proses analisis data dilakukan berdasarkan hasil pengujian yang mencakup pengujian *hardware* dan *software* yang digunakan dalam sistem pengaturan kecepatan dan kendali jarak otomatis pada mobil listrik, serta pengujian keseluruhan sistem. Jika terdapat kekurangan dalam data yang diperoleh, tahap ini akan melibatkan pencarian solusi dan perbaikan guna mencapai hasil yang optimal.

3.7 Pembuatan Makalah Proyek Akhir

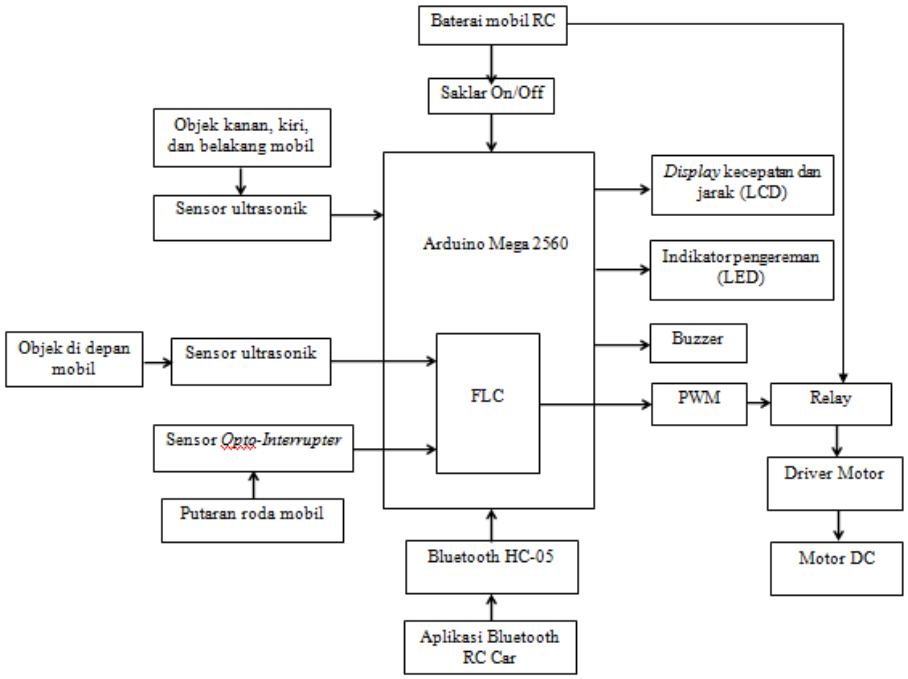
Makalah proyek akhir ini disusun dengan mempertimbangkan semua aspek yang terkait dengan pembuatan proyek akhir, mulai dari pendahuluan, teori, tahap pelaksanaan, hasil pengujian dan pembahasan, hingga kesimpulan dan saran.

BAB IV
PEMBAHASAN

Bab 4 ini akan mengulas tentang langkah-langkah dalam pembuatan proyek akhir, termasuk perancangan dan pembuatan prototipe untuk pengaturan kecepatan dan kendali jarak otomatis pada mobil listrik dengan menggunakan metode *Fuzzy Logic Controller* (FLC), serta pengujian alat tersebut.

4.1 Deskripsi Alat

Prototype dirancang supaya dapat mengatur kecepatan dan kendali jarak otomatis jika terdapat objek di depannya dengan jarak tertentu. Gambar 4.1 berikut ini menampilkan blok diagram yang menjelaskan prinsip kerja dari prototype ini.



Gambar 4. 1 Blok Diagram Prinsip Kerja *Prototype*

Berikut adalah penjelasan mengenai blok diagram di atas:

- *Battery* 9 V, sebagai sumber tegangan mobil dan sistem kontrol.

- Sensor Ultrasonik, sebagai pendeteksi jarak mobil dengan objek yang ada di depan, sisi kiri, sisi kanan, dan belakang. Sensor ultrasonik yang digunakan sebanyak 8 buah yang peletakannya ada dua sensor di depan, dua sensor di kanan, dua sensor di kiri, dan dua sensor di belakang mobil.
- Sensor kecepatan *Opto-Interrupter*, sebagai pengukur rotasi per menit (rpm) motor DC pada mobil listrik.
- Saklar on/off, sebagai pemutus dan penyambung aliran listrik ke Arduino dan ke relay.
- *Buzzer*, sebagai alarm peringatan bahaya jika mobil terlalu dekat dengan objek.
- LED, sebagai indikator nilai FLC sebagai output aktif.
- Relay, sebagai pemutus antara nilai *set point* dan nilai *output* ke motor driver.
- Motor DC, sebagai penggerak mobil.
- Driver Motor, sebagai pengontrol kecepatan dan arah putaran motor DC.
- LCD, sebagai *display* jarak mobil dengan objek, kecepatan mobil, dan kondisi mobil dengan objek.
- *Bluetooth* HC-05, sebagai penghubung untuk mengontrol mobil dari *smartphone.*
- Aplikasi *Bluetooth RC Car,* untuk mengendalikan mobil agar dapat bergerak maju, mundur, ke kiri, dan ke kanan.
- Arduino Mega 2560, sebagai pengendali dan proses data input ke output.

Cara kerja sistem pengontrolan kecepatan dan kendali jarak mobil listrik dapat diterangkan sebagai berikut:

- Sumber tegangan peralatan kontrol berasal dari baterai mobil listrik. Saklar *on/off* digunakan untuk menghidupkan dan mematikan rangkaian kontrol dan relay.
- Untuk menjalankan mobil listrik menggunakan Aplikasi *Bluetooth RC Car*.

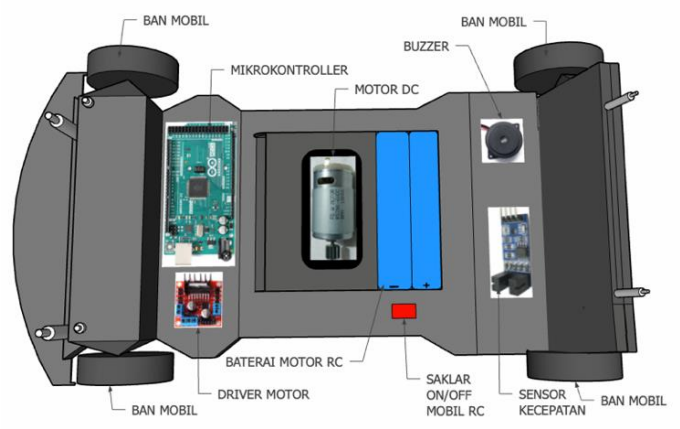
- Input rangkaian kontrol berasal dari sensor jarak (ultrasonik) dan sensor kecepatan (*Opto-Interrupter*).
- Pengaturan sensor ultrasonik yang berada di sisi depan mobil menggunakan *Fuzzy Logic Controller* (FLC) yang akan mengatur nilai dari *Pulse Width Modulation* (PWM). Hasil keluaran PWM akan mengontrol RPM dari motor DC. Untuk sensor jarak pada di sisi kanan, kiri, dan belakang mobil akan mendeteksi objek dan hasilnya akan ditampilkan pada LCD.
- Jarak pendeteksian objek di depan ditentukan dengan pengaturan jarak yang sudah ditentukan mulai dari 0-400 cm.
- Jika kecepatan dan jarak mobil terhadap objek di depan melebih nilai yang ditentukan sebelumnya maka Arduino akan mengaktifkan buzzer sebagai alarm peringatan bahaya dan mobil akan melakukan pengereman bertahap secara otomatis.
- Pengereman ini dilakukan dengan cara pengaturan PWM oleh FLC lalu diteruskan ke driver motor dan motor DC.
- LED akan menyala saat FLC aktif.

4.2 Perancangan dan Pembuatan *Hardware Prototype* Pengaturan Kecepatan dan Kendali Jarak Otomatis pada Mobil Listrik

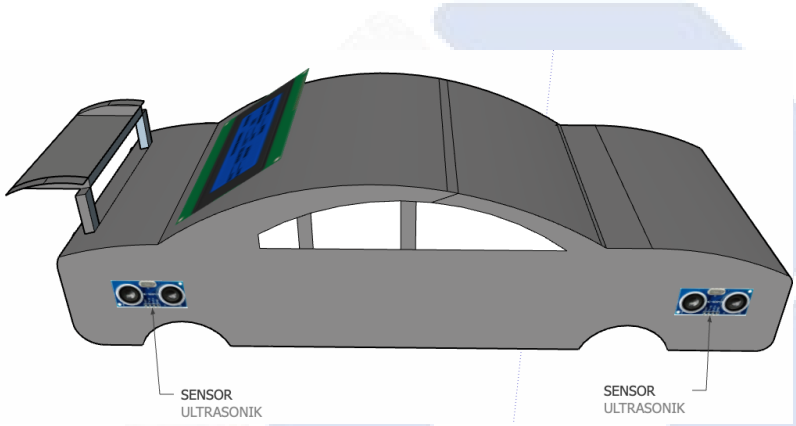
Ada dua tahap dalam perancangan dan pembuatan *hardware* untuk *prototype* pengaturan kecepatan dan kendali jarak otomatis pada mobil listrik ini, yaitu tahap mekanik dan elektrik. Berikut adalah langkah-langkah dalam tahap perancangan dan pembuatan hardware.

4.2.1 Perancangan *Hardware* secara Mekanik

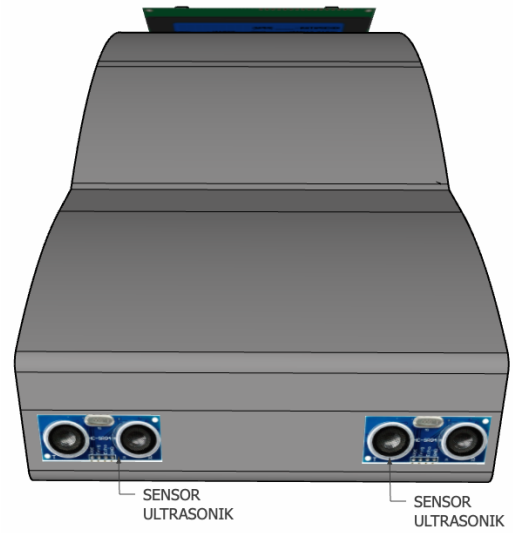
Pada tahap perancangan *hardware* secara mekanik, dilakukan penyusunan desain fisik dari *prototype* pengaturan kecepatan dan kendali jarak otomatis menggunakan aplikasi SketchUp. Dalam pembuatan *prototype* ini kami menggunakan mobil *remote control* dengan ukuran 45 cm x 20 cm x 12 cm. Berikut adalah desain *prototype* yang akan kami buat.



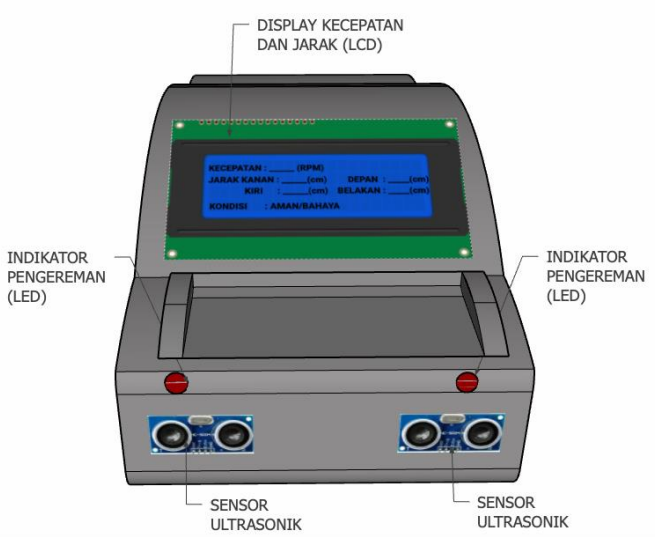
Gambar 4. 2 Tampak Dalam *Prototype* Pengaturan Kecepatan dan Kendali Jarak



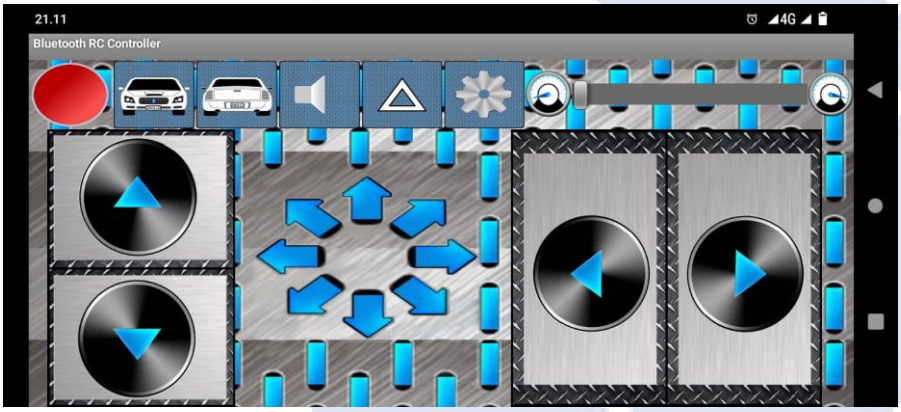
Gambar 4. 3 Tampak Samping *Prototype* Pengaturan Kecepatan dan Kendali Jarak



Gambar 4. 4 Tampak Depan *Prototype* Pengaturan Kecepatan dan Kendali Jarak



Gambar 4. 5 Tampak Belakang *Prototype* Pengaturan Kecepatan dan Kendali Jarak

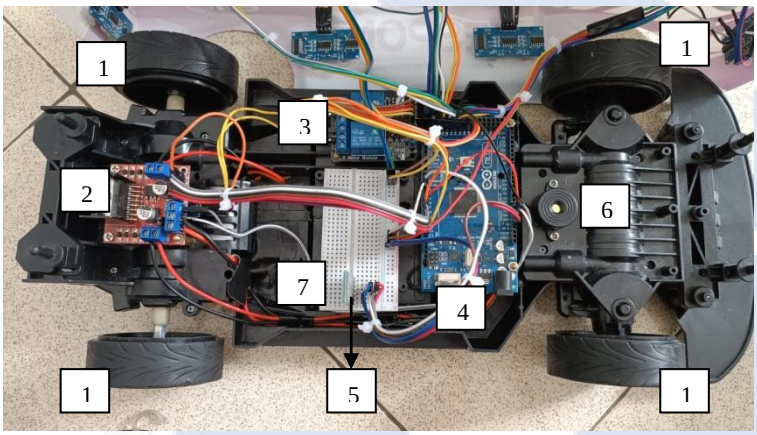


Gambar 4. 6 *Remote Control* pada *Smartphone*

Gambar 4. 7 Tampilan pada LCD

4.2.2 Pembuatan *Hardware* secara Mekanik

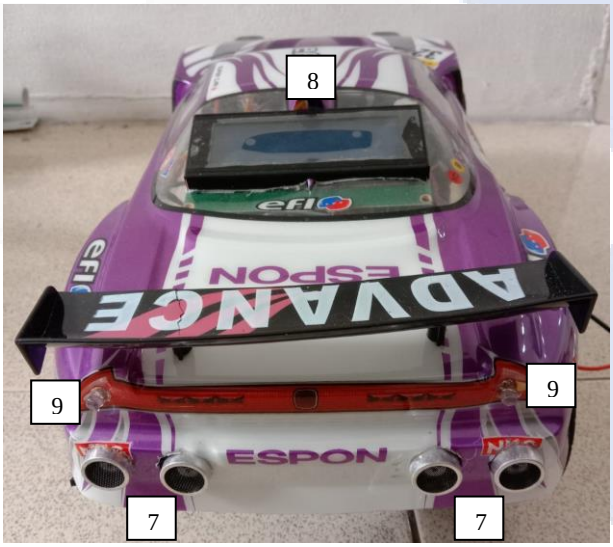
Prototype ini menggunakan mobil *remote control* dengan dimensi 40 cm x 20 cm x 12 cm. Pada body *mobil remote control* tersebut, terdapat 8 sensor ultrasonik yang dipasang dengan rincian 2 sensor pada setiap sisi yaitu kiri, kanan, depan, dan belakang mobil. Kemudian akan dipasang Arduino Mega 2560, sensor *Opto-Interrupter*, *buzzer*, driver motor, motor DC, modul *bluetooth* HC-05,dan baterai pada bagian dalam mobil. LCD 20x4 akan dipasang pada bagian atas mobil. Berikut ini merupakan hasil akhir dari pembuatan *prototype* pengaturan kecepatan dan kendali jarak otomatis.



Gambar 4. 8 Tampak Dalam *Prototype* Pengaturan Kecepatan dan Kendali Jarak

Gambar 4. 9 Tampak Samping *Prototype* Pengaturan Kecepatan dan Kendali Jarak

Gambar 4. 10 Tampak Depan *Prototype* Pengaturan Kecepatan dan Kendali Jarak



Gambar 4. 11 Tampak Belakang *Prototype* Pengaturan Kecepatan dan Kendali Jarak

Keterangan :

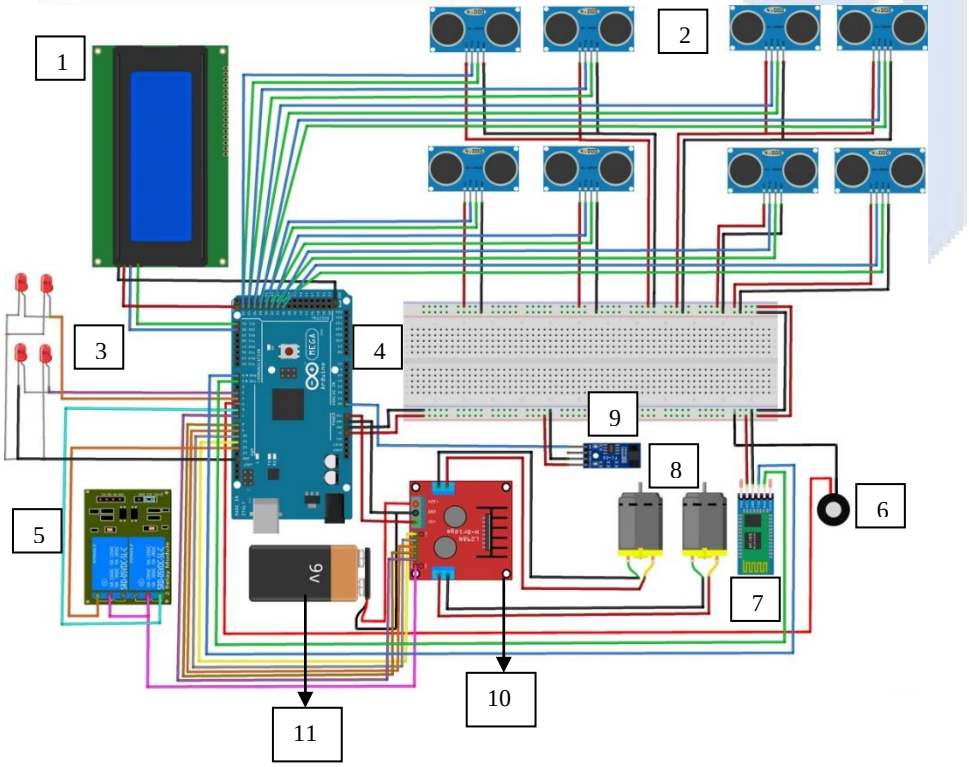
1. Ban Mobil
2. Driver Motor
3. Relay
4. Arduino Mega 2560
5. *Bluetooth* HC-05
6. *Buzzer*
7. Sensor HC-SR05
8. LCD
9. LED

4.2.3 Perancangan dan Pembuatan *Hardware* Elektrik

Pada tahap ini, dilakukan perancangan sistem yang bertujuan mengatur kecepatan dan jarak pada mobil. Pada perancangan ini terdapat 2 data input yaitu input dari sensor ultrasonik berupa jarak dan juga input dari sensor *Opto-Interrupter* berupa kecepatan. Dari kedua data tersebut akan diproses oleh Arduino Mega untuk mengontrol kecepatan, *buzzer*, dan LED.

Perancangan dan pembuatan sistem untuk mengontrol sistem pengaturan kecepatan dan kendali jarak meliputi tahap berikut ini.

1. Membuat *wiring* atau skema pengkabelan menggunakan aplikasi Fritzing. Pembuatan skema pengkabelan ini bertujuan untuk mempermudah dalam membuat dan memperbaiki pada pemasangan *wiring* yang sebenarnya. Gambar 4.12 berikut ini menampilkan skema rangkaian dari komponen yang digunakan dalam proyek ini.

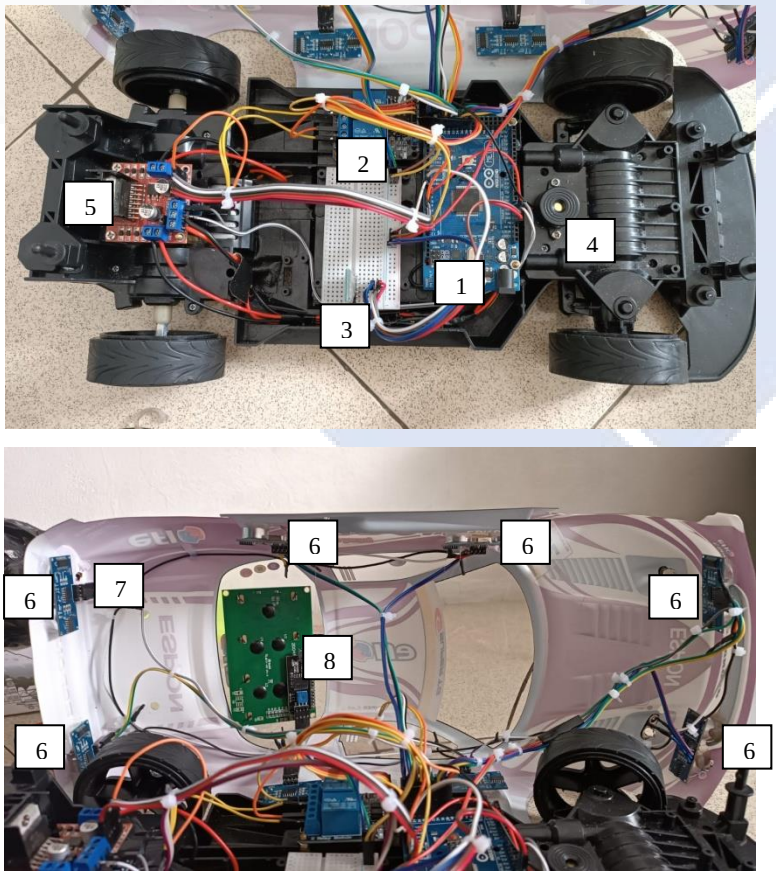


Gambar 4. 12 Skema Rangkaian Sistem Kontrol

Keterangan :

1. LCD
2. Sensor HC-SR04
3. LED
4. Arduino Mega 2560
5. Relay
6. *Buzzer*
7. *Bluetooth* HC-05
8. Motor DC
9. Sensor *Opto-Interrupter*
10. Driver Motor L298N
11. Baterai 9V

2. Pemasangan kabel pada komponen dengan menghubungkan sensor HC-SR04, sensor *Opto-Interrupter*, *buzzer*, LED, LCD, modul *Bluetooth* HC-05, driver motor, dan relay ke pin-pin Arduino Mega. Berikut adalah hasil akhir dari pembuatan sistem kontrol yang dipasang pada *prototype.*



Gambar 4. 13 Rangkaian Sistem Kontrol *Prototype* Pengaturan Kecepatan dan Kendali Jarak

Keterangan :

1. Arduino Mega 2560
2. Relay
3. *Bluetooth* HC-05
4. *Buzzer*
5. Driver Motor L298N
6. Sensor HC-SR04
7. LED
8. LCD

4.3 Pengujian *Hardware* Elektrik *Prototype* Pengaturan Kecepatan dan Kendali Jarak Otomatis pada Mobil Listrik

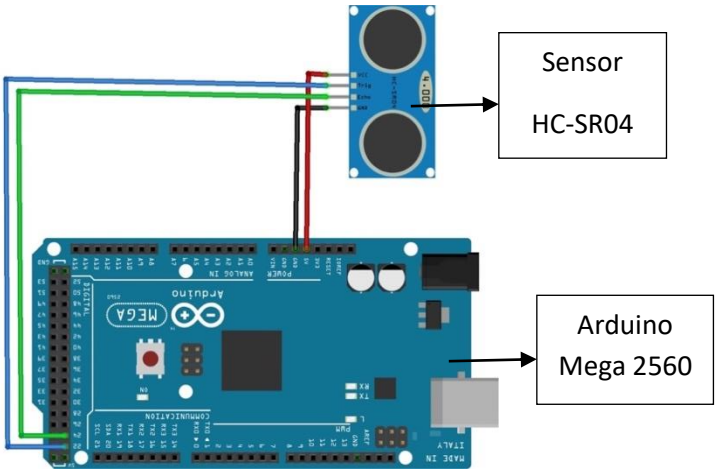
Pengujian *hardware* elektrik dilakukan untuk menguji kemampuan komponen-komponen yang digunakan dalam proyek akhir. Hal ini bertujuan untuk mengetahui sejauh mana kinerja komponen-komponen tersebut dalam implementasi proyek.

4.3.1 Pengujian Sensor HC-SR04

Pengujian terhadap sensor ultrasonik HC-SR04 dilakukan untuk memahami kemampuan sensor tersebut dalam mendeteksi jarak dengan objek. Dengan mengetahui batas deteksi sensor ultrasonik HC-SR04 maka akan lebih mudah untuk memperkirakan peletakkan objek dengan jarak dekat, sedang, dan jauh. Pengujian sensor ultrasonik HC-SR04 ini melibatkan serangkaian tahapan sebagai berikut.

4.3.1.1 Perancangan dan Pembuatan Rangkaian Pengujian Sensor HC-SR04

Tahap ini merupakan perancangan dan pembuatan skema rangkaian menggunakan aplikasi Fritzing untuk menghubungkan pin pada sensor ultrasonik HC-SR04 ke Arduino Mega. Skema rangkaian tersebut dapat dilihat pada Gambar 4.14 berikut ini.

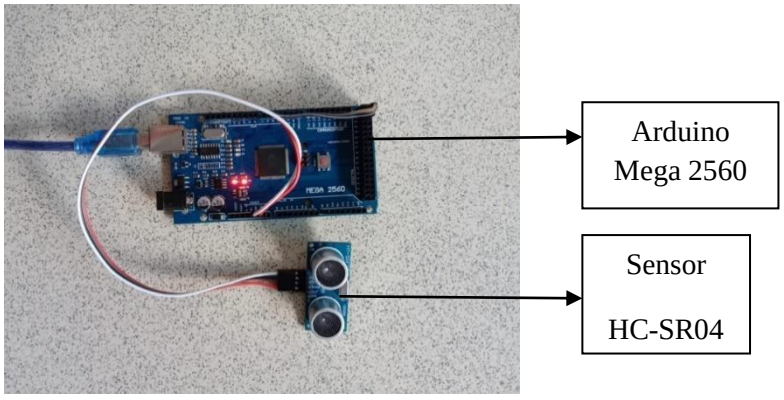


Gambar 4. 14 Skema Rangkaian Sensor HC-SR04 dengan Arduino Mega 2560

Berdasarkan Gambar 4.15, dapat disimpulkan bahwa penghubung antara sensor HC-SR04 dan Arduino Mega dilakukan melalui pin-pin berikut:

- Pin Vcc sensor HC-SR04 dihubungkan ke pin 5 V pada Arduino Mega
- Pin Gnd sensor HC-SR04 dihubungkan ke pin Gnd pada Arduino Mega
- Pin Trig sensor HC-SR04 dihubungkan ke pin 22 pada Arduino Mega
- Pin Echo sensor HC-SR04 dihubungkan ke pin 24 pada Arduino Mega

Gambar 4.15 berikut ini menunjukkan rangkaian antara sensor HC-SR04 dan Arduino Mega



Gambar 4. 15 Rangkaian Sensor HC-SR04 dengan Arduino Mega 2560

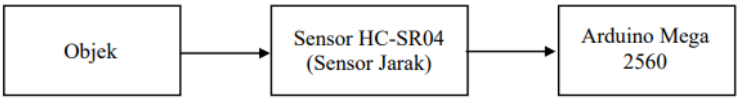
Selain merancang dan skema rangkaian sensor HC-SR04 dengan Arduino Mega, tahap ini juga perlu dibuat program yang digunakan dalam pengujian. Program ini dibuat pada *software* Arduino IDE dengan program sebagai berikut.

```
// Konstanta
const int trigPin = 22;  // Pin Trig pada sensor ultrasonik
const int echoPin = 24;  // Pin Echo pada sensor ultrasonik

void setup() {
  // Inisialisasi komunikasi Serial
  Serial.begin(9600);

  // Mengatur pin sensor ultrasonik
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Mengirimkan sinyal trigger ke sensor ultrasonik selama 10µs
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Mengukur durasi pulsa pada pin echo
  long duration = pulseIn(echoPin, HIGH);

  // Menghitung jarak berdasarkan kecepatan suara
  // (dianggap 343 meter per detik)
  float distance = duration * 0.0343 / 2;

  // Menampilkan jarak ke Serial Monitor
  Serial.print("Jarak: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Delay sebelum mengukur jarak berikutnya
  delay(100);
}
```

4.3.1.2 Prosedur Pengujian Sensor HC-SR04

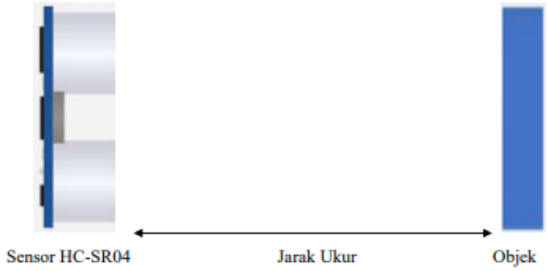
Pengujian ini menggunakan input berupa objek yang ada di depan sensor ultrasonik HC-SR04 dengan jarak tertentu. Setelah didapat hasil pengujian dari

sensor ultrasonik HC-SR04 ini, maka hasil tersebut akan dicatat dalam Tabel 4.1. Jarak yang dihasilkan oleh sensor ultrasonik HC-SR04 ini akan dibandingkan dengan jarak sebenarnya. Berikut ini adalah diagram blok pengujian sensor ultrasonik terhadap objek dengan jarak tertentu.



Gambar 4. 16 Diagram Blok Pengujian Sensor HC-SR04

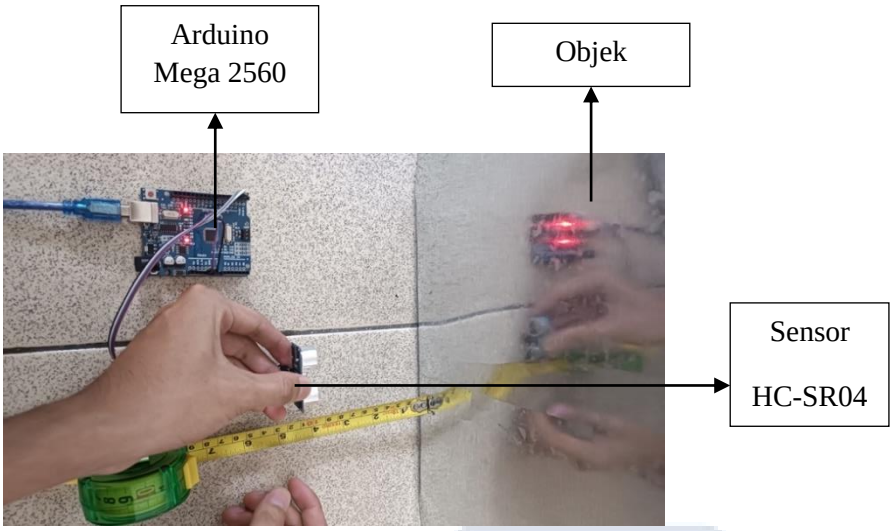
Diagram pengujian sensor HC-SR04 terhadap objek dapat dilihat pada Gambar berikut.



Gambar 4. 17 Diagram Pengujian Sensor HC-SR04 terhadap Objek

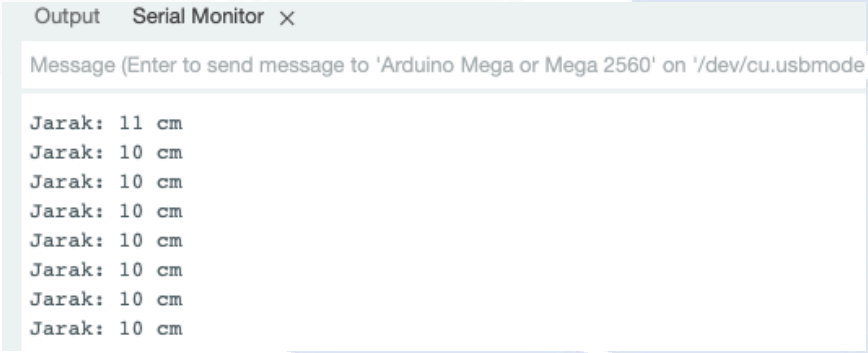
4.3.1.3 Hasil Pengujian Sensor HC-SR04

Pengujian sensor HC-SR04 dilakukan dengan mengukur jarak objek pada posisi tertentu dan membandingkan hasil pengukuran dengan jarak sebenarnya. Gambar 4.18 berikut ini menunjukkan hasil pengujian sensor HC-SR04 pada jarak 10 cm.



Gambar 4. 18 Pengujian Sensor HC-SR04 pada Jarak 10 cm

Berikut merupakan hasil pengujian sensor pada jarak 10 cm.



Gambar 4. 19 Hasil Pengujian Sensor HC-SR04 pada Jarak 10 cm

Pengujian dilakukan mulai dari jarak minimum sensor ultrasonik dapat membaca yaitu 3 cm sampai objek tidak terdeteksi lagi oleh sensor yaitu pada jarak 400 cm. Berikut ini adalah tabel yang menampilkan hasil pengujian sensor HC-SR04 terhadap objek pada beberapa jarak tertentu.

Tabel 4. 1 Hasil Pengujian Sensor HC-SR04 terhadap Objek

Input	Jarak Sensor terhadap Objek	Output	Selisih	*Error*	Rata-Rata *Error*
	3	3,07	0,07	2,33%	

Input	Jarak Sensor terhadap Objek	Output	Selisih	*Error*	Rata-Rata *Error*
Jarak	5	5,23	0,23	4,60%	1,16%
	10	10	0	0%	
	20	20,2	0,2	1,00%	
	30	30,3	0,3	1,00%	
	50	49,43	0,57	1,14%	
	75	74,12	0,88	1,17%	
	100	99,25	0,75	0,75%	
	125	124,65	0,35	0,28%	
Jarak	150	148,78	1,22	0,81%	1,16%
	175	173,95	1,05	0,60%	
	200	198,19	1,81	0,91%	
	250	248,86	1,14	0,46%	
	300	297,67	2,33	0,78%	
	350	348,19	1,81	0,52%	
	400	398,65	1,35	0,34%	

Perhitungan persentase *error* (%) yang didapatkan pada pengujian sensor HC-SR04 dihitung dengan rumus berikut ini.

$$\%\ error = \frac{\text{Jarak sensor terhadap obje}k - output}{\text{Jarak sensor terhadap objek}} \text{ x } 100\%$$

$$\%\ error = \frac{100-99{,}25}{100} \text{ x } 100\%$$

$$\%\ error = 0{,}75\%$$

Berdasarkan data yang terdapat dalam tabel pengujian sensor HC-SR04 dan jarak sebenarnya terhadap objek, terdapat rata-rata persentase kesalahan sebesar

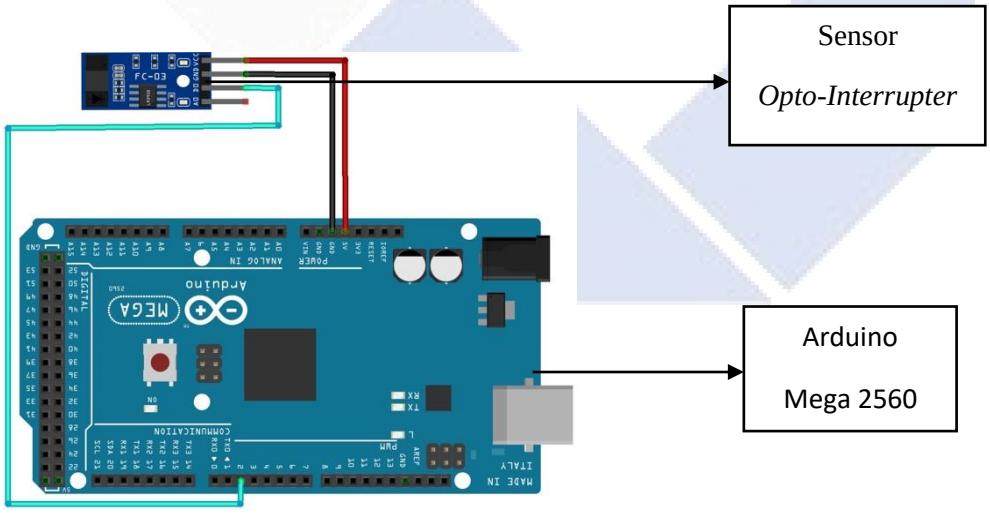
1,16%. Dari nilai rata-rata *error* pada pengujian tersebut dapat disimpulkan bahwa kemampuan deteksi sensor HC-SR04 dapat berfungsi dengan baik sehingga pembuatan proyek akhir digunakan sensor ini sebagai pendeteksi jarak.

4.3.2 Pengujian Sensor *Opto-Interrupter*

Pengujian pada sensor *Opto-Interrupter* dilakukan untuk mengetahui kemampuan deteksi sensor *Opto-Interrupter* terhadap kecepatan motor DC agar dapat lebih mudah untuk menentukan penempatan posisi sensor agar dapat mendeteksi kecepatan motor DC dengan baik. Adapun pengujian sensor *Opto-Interrupter* meliputi tahapan berikut.

4.3.2.1 Perancangan dan Pembuatan Rangkaian Pengujian Sensor *Opto-Interrupter*

Perancangan dan pembuatan dilakukan dengan menyusun skema rangkaian yang menghubungkan pin-pin dari sensor *Opto-Interrupter* ke Arduino Mega. Berikut ini adalah skema rangkaian antara sensor *Opto-Interrupter* dan Arduino Mega menggunakan aplikasi Fritzing.

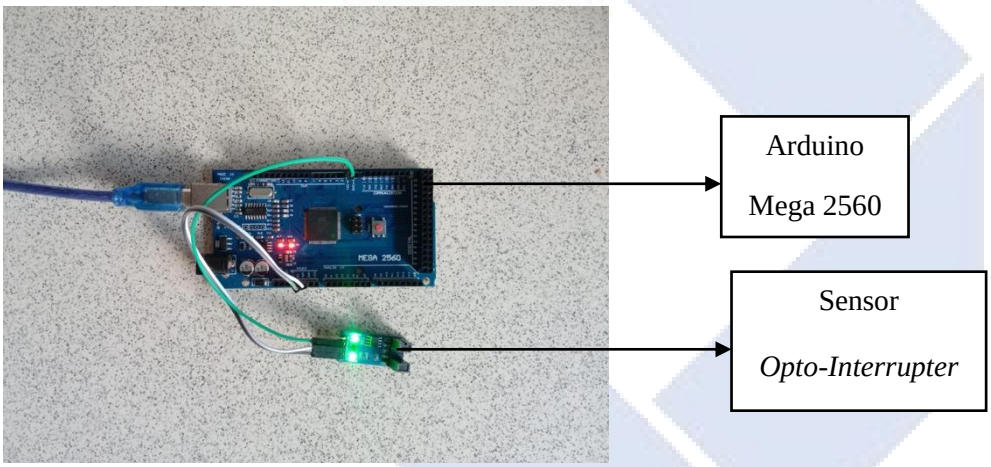


Gambar 4. 20 Skema Rangkaian Sensor *Opto-Interrupter* dengan Arduino Mega 2560

Berdasarkan Gambar 4.20, dapat disimpulkan bahwa penghubung antara sensor Opto-Interrupter dan Arduino Mega 2560 dilakukan melalui pin-pin berikut:

- Pin Vcc sensor *Opto-Interrupter* dihubungkan ke pin 5 V pada Arduino Mega 2560
- Pin A0 sensor *Opto-Interrupter* dihubungkan ke pin 2 pada Arduino Mega 2560
- Pin Gnd sensor *Opto-Interrupter* dihubungkan ke pin Gnd pada Arduino Mega 2450

Hasil dari rangkaian antara sensor *Opto-Interrupter* dan Arduino Mega 2560 dapat dilihat pada Gambar 4.21 berikut ini



Gambar 4. 21 Rangkaian Sensor *Opto-Interrupter* dengan Arduino Mega 2560

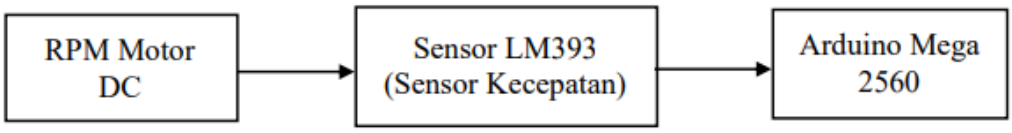
Selain merancang dan skema rangkaian sensor HC-SR04 dengan Arduino Mega 2560, pada tahap ini juga perlu dibuat program untuk pengujian. Program ini dibuat pada *software* Arduino IDE dengan program sebagai berikut.

```
void setup(){
lcd.init();
  lcd.backlight();

pinMode(encoderPin, INPUT);
```

```
  attachInterrupt(digitalPinToInterrupt(encoderPin), countPulse,
RISING);
}
void loop(){
unsigned long currentTime = millis();

  if (currentTime - prevTime >= 1000) {  // Menghitung RPM setiap
detik
    rpm = (count * 60) / 20;  // Menghitung RPM dengan asumsi 20
pulsa per putaran
    count = 0;                // Reset jumlah pulsa

    prevTime = currentTime;
  lcd.setCursor(0,0);
  lcd.print("Kec:");
  lcd.print(rpm);
}
```

4.3.2.2 Prosedur Pengujian Sensor *Opto-Interrupter*

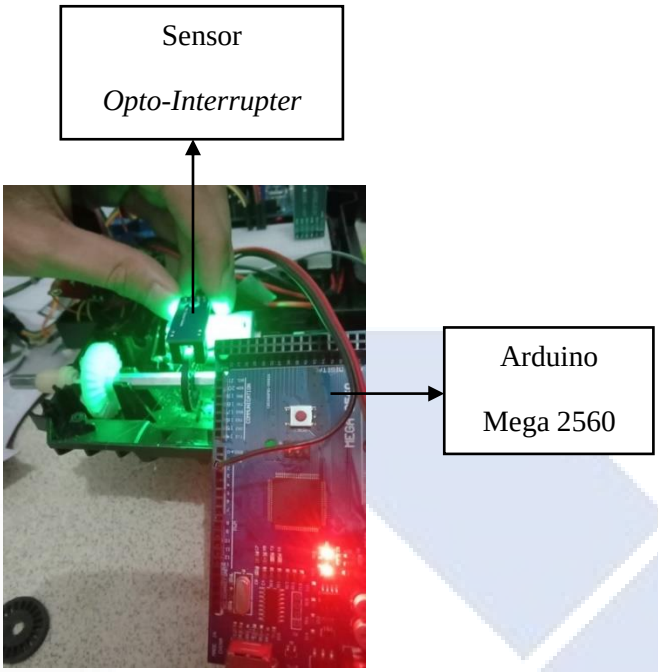
Prosedur pengujian yaitu dengan menggunakan input berupa RPM dari motor DC. Kemudian hasil pengujian dari sensor kecepatan *Opto-Interrupter* ini dicatat dalam Tabel 4.2. Perlu diketahui bahwa prinsip kerja dari sensor Kecepatan *Opto-Interrupter* ini adalah mendeteksi kecepatan yang dihasilkan oleh motor DC. Kecepatan yang dihasilkan oleh sensor *Opto-Interrupter* ini akan dibandingkan dengan pengukuran menggunakan *tachometer* dengan mengatur PWM pada nilai tertentu. Berikut ini adalah diagram blok pengujian sensor kecepatan *Opto-Interrupter* terhadap motor DC.



Gambar 4. 22 Diagram Blok Pengujian Sensor *Opto-Interrupter*

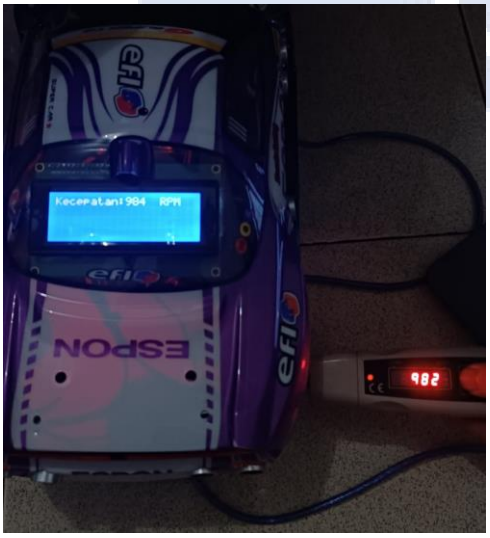
4.3.2.3 Hasil Pengujian Sensor *Opto-Interrupter*

Pengujian sensor *Opto-Interrupter* terhadap RPM motor DC dapat dilihat pada Gambar 4.23 berikut ini.



Gambar 4. 23 Pengujian Sensor *Opto-Interrupter* terhadap RPM Motor DC

Pengujian sensor *Opto-Interrupter* terhadap RPM motor DC dengan pengaturan PWM sebesar 150% diperoleh hasil pada LCD sebagai berikut.



Gambar 4. 24 Hasil Pengujian Sensor *Opto-Interrupter* terhadap RPM Motor DC

Pengujian dilakukan mulai dari PWM 75 sampai dengan PWM 255. Untuk membandingkan tingkat keakuratan sensor *Opto-Interrupter* , maka digunakan *tachometer* sebagai pembanding nilai yang dihasilkan. Berikut adalah tabel hasil pengujian sensor *Opto-Interrupter* terhadap RPM Motor DC.

Tabel 4. 2 Hasil Pengujian Sensor *Opto-Interrupter* terhadap RPM Motor DC

PWM	RPM		Selisih	*Error*	Rata-Rata *Error*
	Sensor *Opto-Interrupter*	*Tachometer*			
75	528	554	26	4,92 %	
100	765	760	5	0,65 %	
125	897	898	1	0,11 %	
127	916	918	2	0,22 %	
150	984	982	2	0,20 %	1,14 %
175	1032	1013	19	1,84 %	
191	1080	1076	4	0,37 %	
200	1095	1082	13	1,19 %	
225	1134	1128	6	0,53 %	
255	1170	1154	16	1,37 %	

Kemudian perhitungan persentase *error* (%) yang didapatkan pada pengujian sensor HC-SR04 dihitung dengan rumus berikut ini.

$$\%\ error = \frac{Tachometer - \text{Sensor } Opto-Interrupter}{Tachometer} \text{ x } 100\%$$

$$\%\ error = \frac{1080-1076}{1076} \text{ x } 100\%$$

$$\%\ error = 0{,}37\%$$

Berdasarkan tabel hasil pengujian sensor *Opto-Interrupter* dan *tachometer* terhadap RPM Motor DC, didapat bahwa *persentase* rata-rata *error* adalah 1,14%. Jadi, dapat disimpulkan bahwa sensor HC-SR04 dapat berfungsi dengan

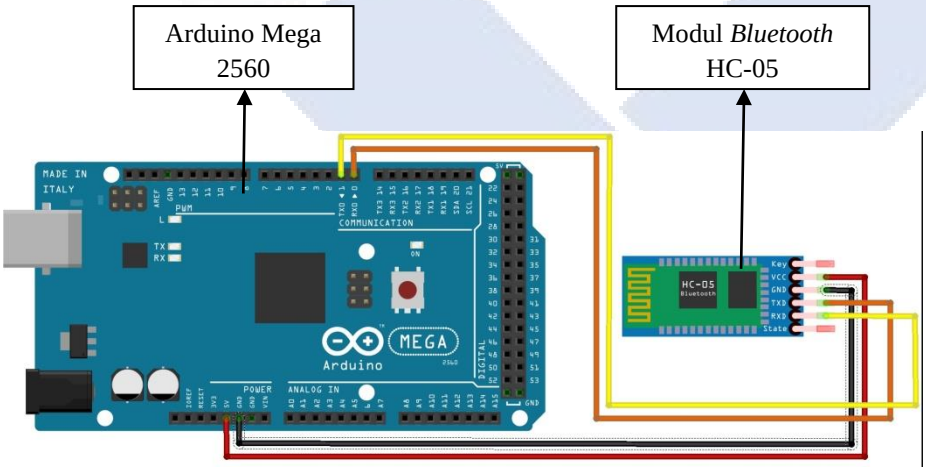
baik sehingga dapat digunakan dalam pembuatan proyek akhir sebagai pendeteksi kecepatan.

4.3.3 Pengujian Modul *Bluetooth* HC-05

Pengujian pada modul *Bluetooth* HC-05 dilakukan untuk menguji kemampuan modul ini sebagai perangkat komunikasi serial yang mengkonversi *port serial* menjadi *Bluetooth.* Pengujian dilakukan dengan mengontrol pergerakan mobil pada jarak tertentu, baik dengan penghalang maupun tanpa penghalang. Berikut adalah tahapan-tahapan dalam pengujian modul *Bluetooth* HC-05.

4.3.3.1 Perancangan dan Pembuatan Rangkaian Pengujian Modul *Bluetooth* HC-05

Perancangan dan pembuatan dilakukan dengan menyusun skema rangkaian yang menghubungkan pin-pin dari modul *Bluetooth* HC-05 ke Arduino Mega 2560. Berikut ini adalah skema rangkaian antara modul Bluetooth HC-05 dan Arduino Mega 2560 menggunakan aplikasi Fritzing.

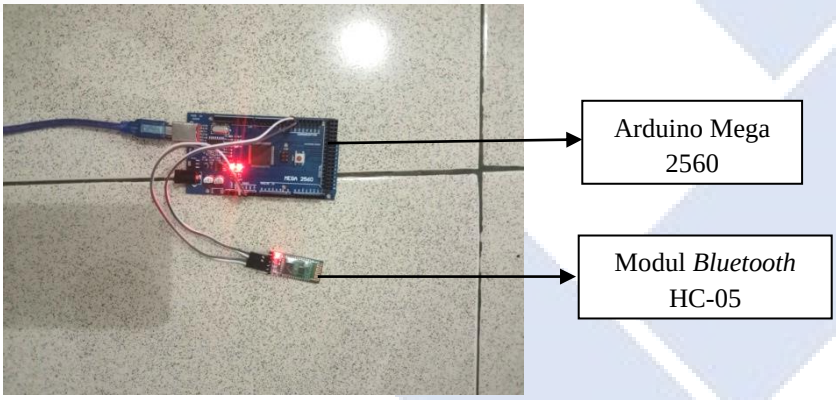


Gambar 4. 25 Skema Rangkaian Modul *Bluetooth* HC-05 dengan Arduino Mega 2560

Dari Gambar 4.25 di atas dapat diketahui bahwa pin yang digunakan pada modul *bluetooth* HC-05 ke Arduino Mega 2560 sebagai berikut.

- Pin Vcc *bluetooth* HC-05 dihubungkan ke pin 5 V pada Arduino Mega 2560
- Pin Gnd *bluetooth* HC-05 dihubungkan ke pin Gnd pada Arduino Mega 2450
- Pin Rx *bluetooth* HC-05 dihubungkan ke pin 1 Tx 0 pada Arduino Mega 2560
- Pin Tx *bluetooth* HC-05 dihubungkan ke pin 0 Rx 0 pada Arduino Mega 2560

Hasil dari rangkaian modul *bluetooth* HC-05 dengan Arduino Mega 2560 dapat dilihat pada Gambar 4.26 berikut ini.



Gambar 4. 26 Rangkaian Modul *Bluetooth* HC-05 dengan Arduino Mega 2560

Selain merancang dan skema rangkaian modul *bluetooth* HC-05 dengan Arduino Mega 2560, pada tahap ini juga perlu dibuat program untuk pengujian. Program ini dibuat pada *software* Arduino IDE dengan program sebagai berikut.

```
//// BLUETOOTH//////
#define E1 11       // Enable Pin Untuk motor 1 LR
#define E2 6        // Enable Pin Untuk motor 2 FB
#define in1 7       // Motor 1 F
```

```
#define in2 8       // Motor 1 B
#define in3 9           // Motor 2 L
#define in4 10         // Motor 2 R
#define buzzer 5
#define Ldepan 3       // Lampu Depan
#define Lbelakang 4   // Lampu Belakang

void setup() {
  Serial.begin(9600);
  /// Inisialisasi pin BLUETOOTH
  pinMode(E1, OUTPUT);
  pinMode(E2, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(Ldepan, OUTPUT);
  pinMode(Lbelakang, OUTPUT);

// Mengirim data melalui modul Bluetooth
void loop() {
while (Serial.available() > 0) {
    command = Serial.read();
    // berhenti(); //Initialize with motors stoped.

    switch (command) {
      case 'F':
        maju();
       break;
      case 'B':
        mundur();
        break;
      case 'L':
        Kiri();
        break;
      case 'R':
        Kanan();
```

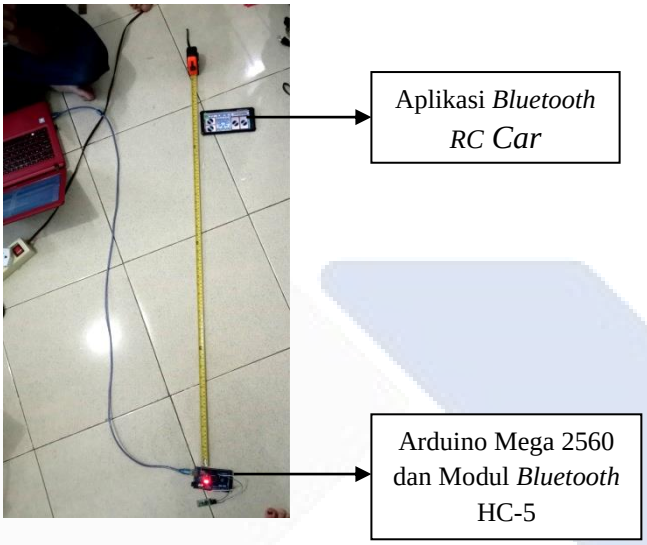
```
   break;
case 'G':
  majukiri();
   break;
case 'I':
  majukanan();
   break;
case 'H':
  mundurkiri();
  break;
case 'J':
  mundurkanan();
  break;
case 'S':
  berhenti();
  break;
case '0':
  Speed = 50;
 break;
case '1':
  Speed = 75;
 break;
case '2':
  Speed = 80;
break;
case '3':
  Speed = 100;
 break;
case '4':
  Speed = 120;
  break;
case '5':
  Speed = 130;
  break;
case '6':
  Speed = 140;
  break;
case '7':
```

```
      Speed = 140;
     break;
    case '8':
      Speed = 160;
      break;
    case '9':
      Speed = 180;
       break;
    case 'q':
      Speed = 200 ;
      break;
    case 'V':  // horn On
      digitalWrite(buzzer, HIGH);
      break;
    case 'v':  // horn Off
      digitalWrite(buzzer, LOW);
      break;
     case 'W':  // Front Light On
      digitalWrite(Ldepan, HIGH);
      break;
    case 'w':  // Front Light Off
      digitalWrite(Ldepan, LOW);
      break;
    case 'U':  // Back Light On
      digitalWrite(Lbelakang, HIGH);
      break;
    case 'u':  // Back Light Off
      digitalWrite(Lbelakang, LOW);
      break;
    case 'X':  // Hazard On
      digitalWrite(Lbelakang && Ldepan, HIGH);
      break;
    case 'x':  // Hazard Off
      digitalWrite(Lbelakang && Ldepan, LOW);
      break;
  }
}
  analogWrite(E1, Speed2);
```

```
    analogWrite(E2, Speed);
    analogWrite(outputfuzzy, output);
}
void maju() {
  digitalWrite(in1, HIGH);
}
void mundur() {
  digitalWrite(in2, HIGH);
}
void Kiri() {
  digitalWrite(in4, HIGH);
}
void Kanan() {
  digitalWrite(in3, HIGH);
}
void majukiri() {
  digitalWrite(in1, HIGH);
  digitalWrite(in4, HIGH);
}
void majukanan() {
  digitalWrite(in1, HIGH);
  digitalWrite(in3, HIGH);
}
void mundurkanan() {
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
}
void mundurkiri() {
  digitalWrite(in2, HIGH);
  digitalWrite(in4, HIGH);
}
void berhenti() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
```

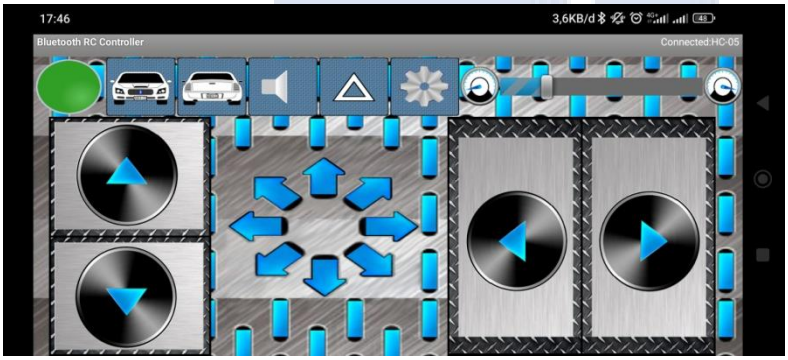
4.3.3.2 Hasil Pengujian Modul *Bluetooth* HC-05

Pengujian dilakukan dengan menghubungkan koneksi modul *Bluetooth* HC-5 dengan aplikasi *Bluetooth RC Car* mulai dari jarak 1 meter sampai koneksi terputus. Berikut adalah gambar pengujian modul *bluetooth* HC-05 dengan jarak tertentu.

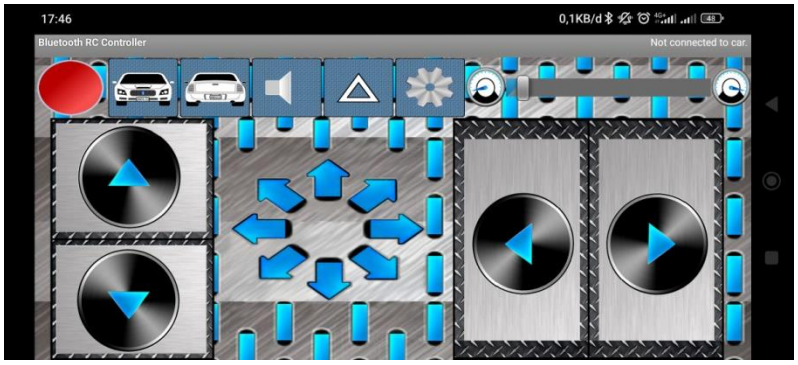


Gambar 4. 27 Pengujian Modul *Bluetooth* HC-05

Untuk melihat koneksi terhubung atau terputus antara modul *Bluetooth* dengan aplikasi untuk pengontrolan dapat dilihat perbedaannya pada Gambar 4.28 dan 4.29 berikut ini.



Gambar 4. 28 Koneksi Terhubung



Gambar 4. 29 Koneksi Terputus

Tabel 4. 3 Hasil Pengujian Modul *Bluetooth* HC-05

Tanpa Penghalang		Dengan Penghalang	
Jarak	Keterangan	Jarak	Keterangan
1 m	Tersambung	1 m	Tersambung
2 m	Tersambung	2 m	Tersambung
3 m	Tersambung	3 m	Tersambung
4 m	Tersambung	4 m	Tersambung
5 m	Tersambung	5 m	Tersambung
6 m	Tersambung	6 m	Tersambung
7 m	Tersambung	7 m	Tersambung
8 m	Tersambung	8 m	Terputus
9 m	Tersambung	9 m	Terputus
10 m	Tersambung	10 m	Terputus

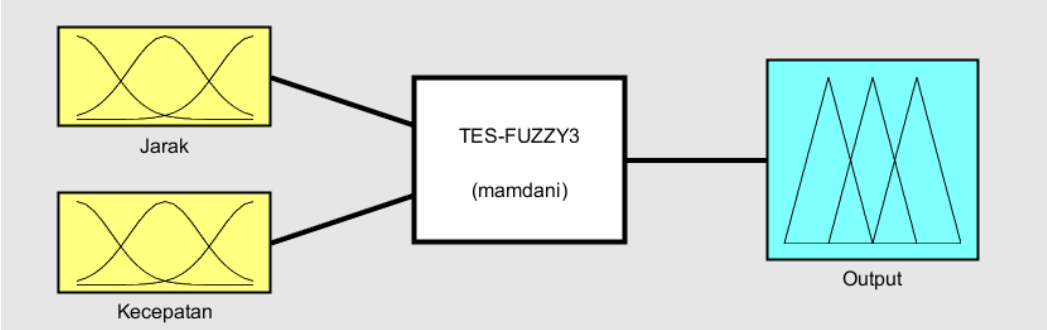
Berdasarkan Tabel 4.3 dapat dilihat bahwa pengujian modul *bluetooth* HC-05 dapat terhubung sampai jarak 9 meter jika tanpa halangan. Namun, jika dihalangi oleh tembok maka modul *bluetooth* HC-05 hanya akan terhubung sampai dengan jarak 7 meter.

4.4 Perancangan dan Pembuatan *Software Prototype* Pengaturan Kecepatan dan Kendali Jarak Otomatis pada Mobil Listrik

Tahapan ini bertujuan untuk merancang, mendesain, dan membuat sistem agar mempermudah dalam proses pembuatan sistemnya. Pada perancangan dan pembuatan *software* akan dibuat rancangan mengenai sistem kontrol menggunakan aplikasi Matlab dan menerapkan metode Mamdani pada *Fuzzy Logic Controller* (FLC).

4.4.1 Perancangan *Software*

Perancangan *software* dilakukan dengan merancang logika *fuzzy* yang diterapkan untuk menentukan pengurangan kecepatan pada mobil listrik berdasarkan *input* 1 yaitu jarak dengan rentang nilai 0-400 cm. kemudian, *input* 2 yaitu kecepatan dari motor DC dengan rentang nilai 0-1000 RPM. Perancangan logika *fuzzy* ini berdasarkan penalaran dan pembacaan sensor yang digunakan, karena konsep logika *fuzzy* ini *fleksibel* dan sama dengan penalaran manusia. Desain *Fuzzy Logic Control* (FLC) menggunakan 2 *input* dan 1 *output* dan metode yang digunakan adalah metode Mamdani. Pada Gambar 4.30 ini merupakan desain blok diagram *Fuzzy Logic Control* (FLC).



Gambar 4. 30 Blok Diagram *Fuzzy Logic Control* (FLC)

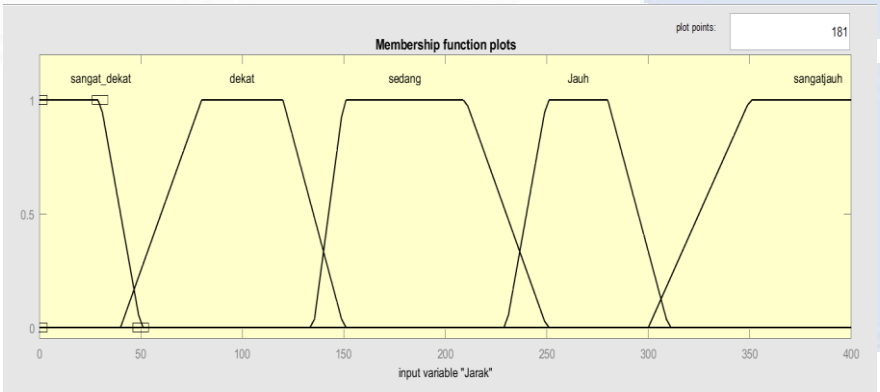
4.4.2 Pembuatan *Software*

Pembuatan *software* sistem pengaturan kecepatan dan kendali jarak otomatis ini dilakukan dengan pembuatan logika *fuzzy* beserta *rules*-nya serta pemrograman menggunakan *software* Arduino IDE dengan menerapkan logika

fuzzy. Adapun tahapan dari pengambilan kesimpulan atau keputusan dengan menggunakan metode *fuzzy* Mamdani yaitu:

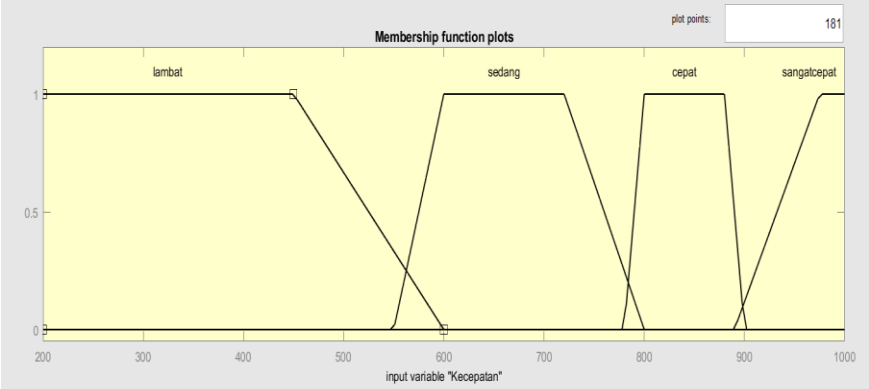
4.4.2.1 Fungsi Keanggotaan Input

Ada 2 *input* yang digunakan pada sistem ini, *input* pertama adalah jarak terhadap objek. Jarak terhadap objek digunakan dalam satuan sentimeter (cm) dan dibagi menjadi 5 fungsi keanggotaan yaitu sangat dekat, dekat, sedang, jauh, dan sangat jauh. Jarak dapat dikategorikan sangat dekat jika bernilai 0-50 cm. Jarak dapat dikategorikan dekat jika bernilai 40-150 cm. Jarak dikategorikan sedang jika bernilai 135-250 cm. Jarak dikategorikan jauh jika bernilai 230-310 cm. Jarak dikategorikan sangat jauh jika bernilai 300-400 cm. Gambar fungsi keanggotaan jarak dapat dilihat pada Gambar 4.31 berikut ini.



Gambar 4. 31 Fungsi Keanggotaan Jarak

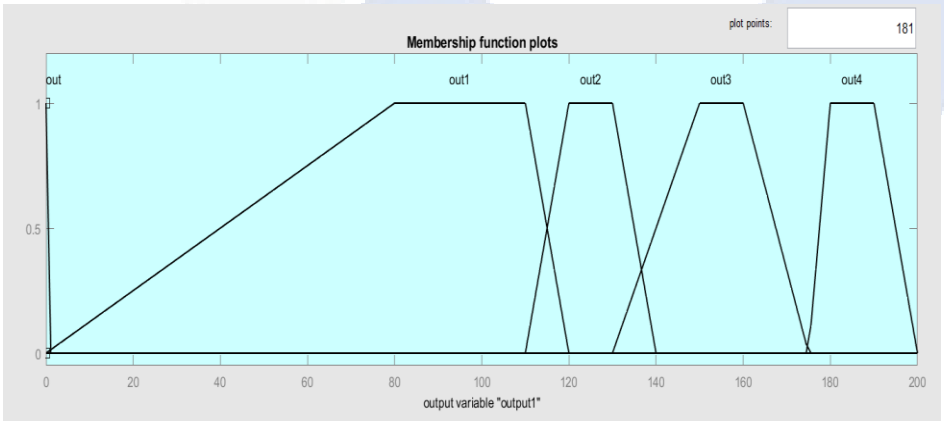
Input yang kedua adalah kecepatan. Kecepatan *prototype* digunakan dalam satuan *Revolution per Minute* (RPM) dan dibagi menjadi 4 fungsi keanggotaan yaitu lambat, sedang, cepat, dan sangat cepat. Kecepatan dapat dikategorikan lambat jika bernilai 200-600 RPM. Kecepatan dikategorikan sedang jika bernilai 550-800 RPM. Kecepatan dikategorikan cepat jika bernilai 780-900 RPM. Kecepatan dikategorikan sangat cepat jika bernilai 890-1000 RPM. Gambar fungsi keanggotaan *input* tekanan dapat dilihat pada Gambar 4.32 berikut ini.



Gambar 4. 32 Fungsi Keanggotaan Kecepatan

4.4.2.2 Fungsi Keanggotaan *Output*

Output pada sistem ini adalah kecepatan berupa PWM yang terbagi menjadi 5 fungsi keanggotaan yaitu *output*, *output* 1, *output* 2, *output* 3, dan *output* 4. Nilai kondisi *output* direpresentasikan dengan angka 0-200. *Output* bernilai 0, *Output* 1 bernilai 0-120, *Output* 2 bernilai 110-140, *Output* 3 bernilai 130-180, dan *Output* 4 bernilai 175-200. Gambar fungsi keanggotaan *output* kecepatan dapat dilihat pada Gambar 4.33 berikut ini.



Gambar 4. 33 Fungsi Keanggotaan *Output*

4.4.2.3 Rules Fuzzy

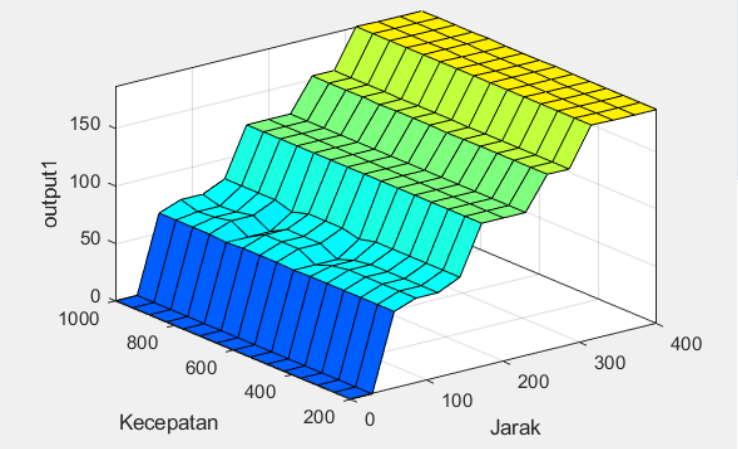
Aturan *fuzzy* ini memiliki 2 *input* yaitu jarak yang memiliki 5 fungsi keanggotaan dan kecepatan yang memiliki 4 fungsi keanggotaan. Setiap fungsi keanggotaan pada *input* lainnya sehingga secara keseluruhan kemungkinan yang

akan didapatkan adalah sebanyak 20 aturan *fuzzy*. Setiap aturan *fuzzy* inilah yang kemudian menentukan *output* sistem. Berikut ini merupakan aturan *fuzzy* dapat dilihat pada Tabel 4.4 berikut ini.

Tabel 4. 4 Tabel Aturan Logika *Fuzzy*

	Sangat Dekat	Dekat	Sedang	Jauh	Sangat Jauh
Lambat	*Output*	*Output* 1	*Output* 2	*Output* 3	*Output* 4
Sedang	*Output*	*Output* 1	*Output* 2	*Output* 3	*Output* 4
Cepat	*Output*	*Output* 1	*Output* 2	*Output* 3	*Output* 4
Sangat Cepat	*Output*	*Output* 1	*Output* 2	*Output* 3	*Output* 4

Hubungan antara jarak dan kecepatan terhadap *output* dibuatkan dalam bentuk *plot surface* maka akan menghasilkan grafik pada Gambar 4.34 berikut ini.



Gambar 4. 34 *Plot Surface* Jarak, Kecepatan, dan *Output*

4.5 Pengujian *Software Prototype* Pengaturan Kecepatan dan Kendali Jarak Otomatis pada Mobil Listrik

Pengujian *software* merupakan pengujian logika *fuzzy* yang dilakukan dengan mengkombinasikan kemungkinan-kemungkinan yang ada. Pada pengujian logika *fuzzy* secara umum dilakukan dengan memasukan nilai *input* 1 tetap dan *input 2* nilainya berubah-ubah. Pada pengujian pertama, input yang nilainya tetap adalah jarak dan input kecepatan nilainya berubah-ubah. Pengujian dilakukan dengan membandingkan simulasi pada aplikasi Matlab dengan hasil yang didapatkan pada pemrograman Arduino. Pengujian ini bertujuan untuk mengetahui apakah metode *Fuzzy Logic Controller* (FLC) pada proyek akhir ini layak digunakan atau tidak. Adapun hasil pengujian pada perubahan nilai *input* kecepatan dapat dilihat pada Tabel 4.5 berikut ini.

Tabel 4. 5 Hasil Pengujian *Fuzzy Logic Controller* (FLC) ketika Perubahan Nilai *Input* Kecepatan

Jarak	Kecepatan	Matlab	Arduino	*Error %*
200	720	125	125	0
200	810	125	125	0
200	900	125	125	0
200	996	125	125	0
Rata-Rata *Error*				0

Dari Tabel 4.5 dimana pengujian dilakukan dengan membandingkan nilai yang dihasilkan pada simulasi Matlab dan pemrograman Arduino. *Error* yang dihasilkan pada pengujian ini senilai 0 %. Dari nilai *error* ini dapat disimpulkan bahwa sistem kontrol menggunakan metode *Fuzzy Logic Controller* (FLC) dapat dilakukan dan layak untuk digunakan. Dari hasil percobaan, dapat dilihat bahwa perubahan nilai kecepatan jika nilai jarak tetap sama, maka *output* yang dihasilkan tidak akan berubah dan tetap pada nilai 125 jika nilai jarak yang diberikan adalah 200 cm.

Pengujian kedua untuk logika *fuzzy* dilakukan dengan merubah nilai pada *input* jarak, sedangkan nilai *input* kecepatan bernilai tetap. Adapun hasil pengujiannya dapat dilihat pada Tabel 4.6 berikut ini.

Tabel 4. 6 Hasil Pengujian *Fuzzy Logic Controller* (FLC) ketika Perubahan Nilai *Input* Jarak

Jarak	Kecepatan	Matlab	Arduino	*Error %*
30	730	0	0	0
50	730	63	64.2	1,86
100	730	72	72.8	1,09
200	730	125	125	0
300	730	152	153	0,65
400	730	186	186	0
Rata-Rata *Error*				0,6

Berdasarkan hasil pengujian pada Tabel 4.6 yang dilakukan dengan membandingkan nilai yang dihasilkan menggunakan simulasi Matlab dan pemrograman Arduino. *Error* terkecil pada hasil pengujian bernilai 0 % dan *error* terbesar bernilai 1,86 %, serta rata-rata *error* bernilai 0,6 %. Dari nilai *error* ini dapat disimpulkan bahwa sistem kontrol menggunakan metode *Fuzzy Logic Controller* (FLC) dapat dilakukan dan layak untuk digunakan. Dari hasil percobaan, dapat dilihat bahwa perubahan nilai jarak jika nilai kecepatan tetap sama, maka *output* yang dihasilkan akan berubah sesuai dengan *rules* yang telah dibuat.

4.7 Pengujian Keseluruhan Alat

Pengujian keseluruhan alat dilakukan untuk memastikan bahwa alat dapat bekerja dengan benar dan sesuai dengan yang diinginkan.

4.7.1 Pengujian Kontrol Jarak Kiri, Kanan, dan Belakang Mobil

Pengujian dilakukan dengan cara meletakan suatu objek pada bagian kiri, kanan, dan belakang mobil untuk mengetahui kondisi *buzzer* yang akan aktif sebagai peringatan jika jarak sudah mencapai batas.

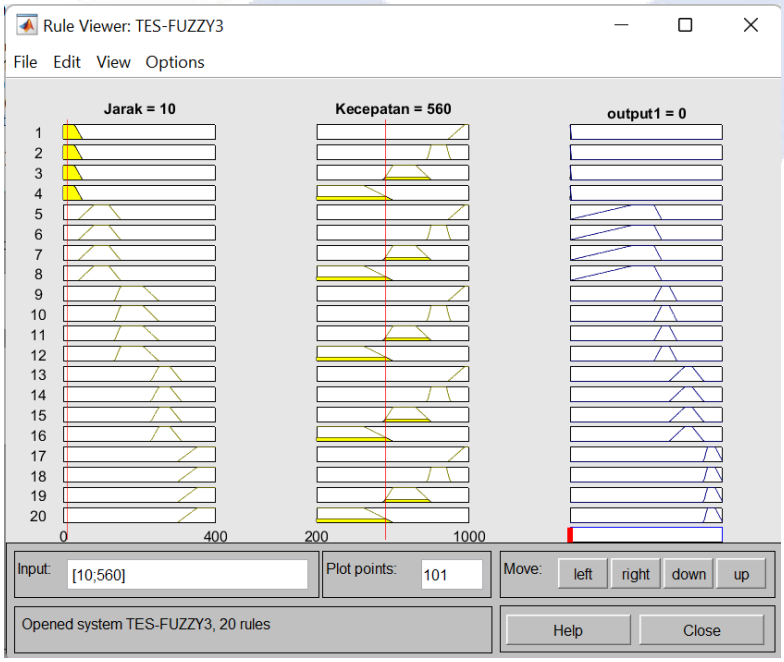
Tabel 4. 7 Pengujian Kontrol Jarak Kiri, Kanan, dan Belakang Mobil

Posisi	Jarak	*Buzzer*
Kanan	175	Aktif
	150	Mati
	100	Mati
	75	Aktif
	50	Aktif
	25	Aktif
	0	Aktif
Kiri	175	Mati
	150	Mati
	100	Mati
	75	Aktif
	50	Aktif
	25	Aktif
	0	Aktif
Belakang	175	Mati
	150	Aktif
	100	Aktif
	75	Aktif
	50	Aktif
	25	Aktif
	0	Aktif

Berdasarkan tabel di atas, dapat dilihat bahwa *buzzer* akan berbunyi jika objek sudah melewati batas aman. Dimana, pada bagian kiri dan kanan mobil, *buzzer* akan menyala saat jarak objek ≤ 75 cm. Sedangkan, pada bagian belakang mobil, *buzzer* akan menyala jika jarak objek ≤ 150 cm.

4.7.2 Pengujian *Prototype* Tanpa Beban

Pengujian dilakukan dengan cara mengangkat *prototype* mobil listrik sehingga *prototype* tidak menyentuh lantai, kemudian memberikan nilai pada *input* jarak dan *input* kecepatan. Nilai yang diberikan pada *input* jarak mulai dari sangat dekat sampai dengan sangat jauh, serta *input* kecepatan diberi nilai mulai dari lambat sampai dengan sangat cepat. Kemudian, data yang dihasilkan dengan nilai *input* tersebut mempengaruhi kecepatan, LED, dan juga *buzzzer* seperti apa, serta *output* yang dihasilkan sama atau tidak jika menggunakan simulasi Matlab. Simulasi pengendalian logika *Fuzzy Logic Control* (FLC) menggunakan *software* Matlab dilakukan dengan memberikan nilai pada *input* yang akan menghasilkan nilai *output*. Simulasi Matlab ini dapat dilihat dalam Gambar 4.35 hingga 4.37 sebagai contohnya.



Gambar 4. 35 Simulasi Matlab dengan Nilai *Input* Jarak 10 dan *Input* Kecepatan 560

Gambar 4. 36 Simulasi Matlab dengan Nilai *Input* Jarak 150 dan *Input* Kecepatan 560

Gambar 4. 37 Simulasi Matlab dengan Nilai *Input* Jarak 400 dan *Input* Kecepatan 975

Simulasi pada aplikasi Matlab ini dilakukan sebanyak 21 kali sama dengan pengujian pada *prototype* ini. Adapun hasil pengujian alat ini dapat dilihat pada Tabel 4.8 berikut ini.

Tabel 4. 8 Pengujian Keseluruhan Alat Tanpa Beban

Uji Ke-	*Input*		*Output*		*Delay*	LED	Kondisi	*Buzzer*
	Jarak (cm)	Kecepatan (RPM)	Matlab	Arduino				
1.	10	560	0	0	2.47	Aktif	Bahaya	Aktif
2.	30	600	0	0	2.9	Aktif	Bahaya	Aktif
3.	50	770	64.2	64.22	2.54	Aktif	Bahaya	Aktif
4.	70	900	61.8	61.83	3.1	Aktif	Bahaya	Aktif
5.	80	560	64.4	65.87	3	Aktif	Bahaya	Aktif
6.	100	600	74	73	2.98	Aktif	Bahaya	Aktif
7.	110	810	74	73	3.28	Mati	Bahaya	Aktif
8.	130	951	70.3	70	2.6	Mati	Bahaya	Aktif
9.	150	560	125	125	2.78	Mati	Bahaya	Aktif
10.	170	665	125	125	2.56	Mati	Aman	Mati
11.	200	870	125	125	3.22	Aktif	Aman	Mati
12.	220	900	125	125	3.43	Aktif	Aman	Mati
13.	230	906	125	125	2.67	Aktif	Aman	Mati
14.	250	560	153	153	2.85	Mati	Aman	Mati
15.	270	600	154	153	3.2	Mati	Aman	Mati
16.	300	870	153	153	3.3	Mati	Aman	Mati
17.	310	951	187	186	3.26	Mati	Aman	Mati
18.	330	560	187	187.18	2.7	Mati	Aman	Mati
19.	350	770	187	186	2.45	Mati	Aman	Mati

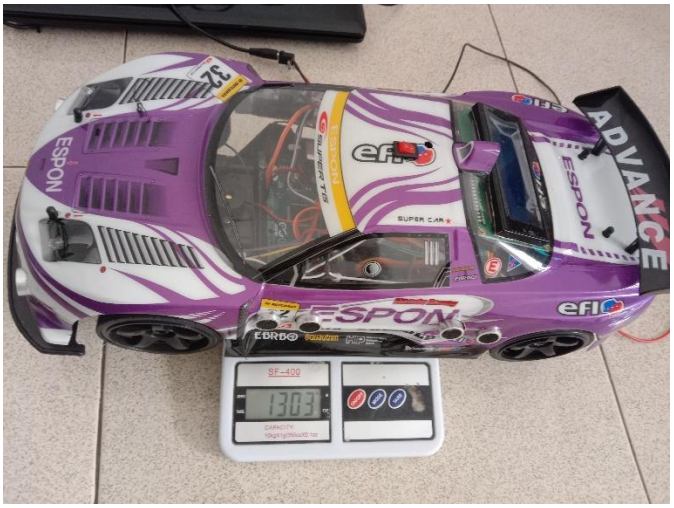
Uji Ke-	*Input*		*Output*		*Delay*	LED	Kondisi	*Buzzer*
	Jarak (cm)	Kecepatan (RPM)	Matlab	Arduino				
20.	370	882	186	186	2.9	Mati	Aman	Mati
21.	400	975	186	186	3.2	Aktif	Aman	Mati
Rata-Rata *Delay*					2.92			

Dari Tabel 4.8 di atas, pada simulasi ini dapat dilihat bahwa mobil akan berhenti jika jarak objek kurang dari 50 cm. Hal ini menyesuaikan keadaan mobil listrik yang akan berhenti jika objek di depannya terlalu dekat untuk menghindari tabrakan. Kemudian, semakin jauh jarak dengan objek, maka kecepatan yang dihasilkan akan semakin besar. Untuk LED akan aktif jika nilai kecepatan yng dihasilkan oleh *output fuzzy* lebih kecil dari kecepatan *set point.* Selanjutnya, kondisi dikatakan aman jika jarak > 150 cm. Kondisi dikatakan bahaya jika jarak ≤ 150 cm yang akan ditandai dengan aktifnya *buzzer*.

Namun, pada simulasi di atas masih mengalami *delay* dengan rata-rata *delay* selama 2.92 detik. *Delay* ini disebabkan karena pembacaan sensor dan pergantian pada *on/off* relay atau *switching time* dan respon pada kontrol *bluetooth.*

4.7.3 Pengujian *Prototype* dengan Beban

Pengujian ini dilakukan dengan meletakkan *prototype* di lantai kemudian memberikan nilai pada *input* jarak dan *input* kecepatan. Nilai yang diberikan pada *input* jarak sebesar 100 cm dan 200 cm, serta *input* kecepatan diberi nilai sebesar 150 RPM. Pengujian *prototype* dengan beban ini dilakukan dengan berat sebesar 1,3 kg dan dapat dilihat pada Gambar 4.38 berikut ini.



Gambar 4.38 *Prototype* dengan Beban

Adapun hasil pengujian alat ini dapat dilihat pada Tabel 4.9 dan Tabel 4.10 berikut ini.

Tabel 4.9 Pengujian Mulai dari Jarak 100 cm

Pengujian	Jarak	*Output* FLC	Jarak Berhenti
1		73	76
2		73	78
3	100 cm	73	69
4		73	67
5		73	60

Pada Tabel 4.9 ini *prototype* akan berhenti bukan pada jarak 50 cm. Hal ini dikarenakan torsi pada motor DC tidak kuat untuk menggerakkan mobil seberat 1,3 kg. Oleh karena itu, mobil akan berhenti pada jarak mulai dari 60 cm sampai dengan 78 cm sebanyak 5 kali pengujian.

Tabel 4.10 Pengujian Mulai dari Jarak 200 cm

Pengujian	Jarak	*Output* FLC	Jarak Berhenti
1		125	Menabrak
2	200 cm	125	Menabrak

Pengujian	Jarak	*Output* FLC	Jarak Berhenti
3		125	5
4	200 cm	125	Menabrak
5		125	Menabrak

Pada Tabel 4.10 di atas, dalam 5 kali pengujian *prototype* kebanyakan tidak berhenti dan menabrak. Hal ini dikarenakan sistem kontrol mengalami *delay* yang disebabkan karena pergantian pada *on/off* relay atau *switching time.* Sistem kontrol ini lambat merespon yang menyebabkan *delay* sehingga kendaraan tidak bisa menyesuaikan kecepatan dengan cepat dan mengakibatkan resiko menabrak. Selain pada penelitian ini, delay juga terjadi pada penelitian Gusti Alga Maulana dan Syah Alam yang menyebabkan kondisi mobil akan menabrak karena terjadi *error. Error* ini terjadi dikarenakan kondisi jalan mobil yang licin dan tidak sempurnanya pembacaan sensor ultrasonik (pembacaan terbatas hanya disudut 15°) [31].

BAB V
PENUTUP

5.1 Kesimpulan

Setelah pembuatan alat selesai dan dilakukan pengujian, maka dari *prototype* pengaturan kecepatan dan kendali jarak otomatis ini dapat disimpulkan bahwa :

1. Pengontrolan pengaturan kecepatan dan pengereman mobil listrik jika terdapat objek di depan mobil dilakukan dengan pemrograman pada mikrokontroler yang akan membuat mobil berhenti atau kecepatan 0 RPM jika jarak terhadap objek < 50 cm. Dan semakin jauh jarak mobil terhadap objek maka akan semakin besar nilai kecepatan yang dapat dihasilkan pada *prototype* ini. Begitupun sebaliknya, jika jarak objek semakin dekat, maka kecepatan dari mobil akan semakin pelan.
2. Kontrol jarak mobil dengan objek yang berada di sebelah kiri dan kanan mobil dilakukan dengan pemrograman pada mikrokontroler yang akan membuat aktifnya *buzzer* jika jarak objek ≤ 75 cm. Objek yang berada di belakang mobil akan mengaktifkan *buzzer* jika jarak objek ≤ 150 cm.
3. Pengujian sensor HC-SR04 dilakukan dengan membandingkan nilai yang dihasilkan oleh sensor dan pengukuran jarak sebenarnya didapatkan nilai *error* rata-rata sebesar 1,16 %. Sensor HC-SR04 dapat membaca jarak maksimal 400 cm.
4. Pengujian sensor *Opto-Interrupter* dilakukan dengan membandingkan nilai yang dihasilkan oleh sensor dan pengukuran menggunakan *tachometer* didapatkan nilai *error* rata-rata sebesar 1,14 %.
5. Pengujian modul *bluetooth* HC-05 menyatakan bahwa pengontrolan menggunakan modul ini dapat dilakukan ingga jarak 9 meter tanpa halangan dan jarak 7 meter jika dihalangi oleh tembok.

6. Pengujian logika *fuzzy* untuk menguji kelayakan penggunaan metode *Fuzzy Logic Controller* (FLC) didapatkan nilai *error* hanya sebesar 0,6 % jika dibandingkan pada pemrograman Arduino.
7. Simulasi masih mengalami *delay* dengan rata-rata *delay* selama 2.92 detik. *Delay* ini disebabkan karena pergantian pada *on/off* relay atau *switching time* dan respon pada kontrol *bluetooth.*
8. Pengujian *prototype* dengan beban mulai dari jarak 100 cm akan berhenti bukan pada jarak 50 cm. Hal ini dikarenakan torsi pada motor DC tidak kuat untuk menggerakkan mobil seberat 1,3 kg. Oleh karena itu, mobil akan berhenti pada jarak mulai dari 60 cm sampai dengan 78 cm sebanyak 5 kali pengujian.
9. Dalam 5 kali pengujian *prototype* dengan beban mulai dari jarak 100 cm kebanyakan tidak berhenti dan menabrak. Hal ini dikarenakan sistem kontrol mengalami *delay* yang disebabkan karena pergantian pada *on/off* relay atau *switching time.* Sistem kontrol ini lambat merespon yang menyebabkan *delay* sehingga kendaraan tidak bisa menyesuaikan kecepatan dengan cepat dan mengakibatkan resiko menabrak.

5.2 Saran

Berikut adalah beberapa saran untuk pengembangan proyek akhir ini:

1. Pada simulasi *prototype* pengaturan kecepatan ini masih mengalami *delay* pada pembacaan sensor sehingga mobil masih menabrak. Pada perkembangan selanjutnya, sebaiknya dilakukan pengurangan *delay* dengan mengganti sensor yang memiliki respon yang lebih cepat.
2. Memperluas basis aturan *fuzzy* pada variabel *fuzzy* agar nilai kecepatan menjadi lebih luas dan akurat. Hal ini akan menghasilkan implementasi pengendalian yang lebih baik.

DAFTAR PUSTAKA

[1] A. I. Maulana, "Tugas Akhir Perancangan Perancangan Fasilitas Penelitian Mobil Listrik di Surabaya.," Universitas 17 Agustus 1945, Surabaya, 2018.

[2] I. Utami, D. Yoesgiantoro and N. A. Sasongko, "Implementasi Kebijakan Kendaraan Listrik Indonesia Untuk Mendukung Ketahanan Energi Nasional," *Jurnal Ketahanan Energi,* vol. 8, no. 1, pp. 49-65, 26 Agustus 2022.

[3] Tesladeaths, "Tesla Deaths," 2023. [Online]. Available: https://www.tesladeaths.com/#totals. [Accessed 23 Februari 2023].

[4] A. D. Saputra, "Studi Tingkat Kecelakaan Lalu Lintas Jalan di Indonesia Berdasarkan Data KNKT (Komite Nasional Keselamatan Transportasi) dari Tahun 2007-2016," *Jurnal Warta Penelitian Perhubungan,* vol. 29, no. 2, pp. 183-190, 2017.

[5] R. B. Rajasa, "Kajian Faktor–Faktor Penyebab Kecelakaan Lalu Lintas Di Kota Bandung.," Institut Teknologi Nasional Bandung, Bandung, 2017.

[6] E. Sugiharti, M. Saleh and M. , "Analisis Faktor Potensi Penyebab Kecelakaan Lalu Lintas Kendaraan Bermotor," *Jurnal Manajemen Bisnis Transportasi dan Logistik (JMBTL),* vol. 5, no. 3, pp. 367-374, 2019.

[7] Irayuana, "Ganti Rugi terhadap Korban Kecelakaan Lalu Lintas," Universitas Islam Negeri Ar-Raniry Darussalam, Banda Aceh, 2019.

[8] A. R. Dayus, J. E. Hutagalung and I. R. Harahap, "Penerapan Sistem Pengereman dan Parkir Mobil Listrik Menggunakan Sensor Ultrasonik Berbasis Arduino UNO," *J-Com (Journal of Computer),* vol. 2, no. 2, pp. 101-106, 2022.

[9] H. Iswanto, "Sistem Kontrol Kecepatan Dan Monitoring Mobil Listrik Berbasis Logika Fuzzy Dan Internet Of Things," Universitas Jember, Jawa Timur, 2020.

[10] A. Ramadan, A. Rusdinar and M. R. Rosa, "Perancangan Kendali Kecepatan Mobil Listrik Dengan Metode PID Berbasis Remot Kontrol," *e-Proceeding of Engineering,* vol. 9, no. 2, pp. 152-161, 2022.

[11] D. A. Saputra, B. Handaga, M. Effendy and D. H. Alim, "Simulasi Pemograman Pengendali PWM Kecepatan dengan Mikrokontroler Arduino berbasis Sensor Ultrasonik HC-SR04 pada Purwarupa Mobil Listrik," *Journal of Mechanical Engineering and Science,* vol. 1, no. 2, pp. 19-25, 2020.

[12] R. Mulyadi, K. D. Artika and M. Khalil, "Perancangan Sistem Kelistrikan Perangkat Elektronik pada Mobil Listrik," *Jurnal Elemen* , vol. 6, no. 1, pp. 7-12, 2019.

[13] P. M. Widiasaputra, I. K. A. Bawa, P. E. Suartawan, D. Fitriani, M. Aryuni and R. R. O. Sasue, "Pengembangan Sistem Lidar Pendeteksi Jarak Aman Berkendara," *Jurnal Rekayasa Sipil dan Lingkungan*, vol. 6, no. 2, pp. 190-199, 2022.

[14] A. Soni and A. Aman, "Distance Measurement of an Object by using Ultrasonic Sensors with Arduino and GSM Module," *International Journal of Science Technology & Engineering*, vol. 4, no. 11, pp. 23-28, 2018.

[15] A. H. Lutfiyanto, "Rancang Bangun Pintu Wahana Otomatis Menggunakan Sensor Ultrasonik Hc-Sr04 Sebagai Pengukur Tinggi Badan Dan Sensor Load Cell Dilengkapi Dengan Hx711 Sebagai Pengukur Berat Badan Berbasis Arduino Mega 2560," Universitas Diponegoro, Semarang, 2017.

[16] A. S. M. Huda, T. A. Zuraiyah and F. L. Hakim, "Prototype Alat Pengukur Jarak Dan Sudut Kemiringan Digital Menggunakan Sensor Ultrasonik dan Accelerometer Berbasis Arduino Nano," *Bina Insani Ict Journal,* vol. 6, no. 2, pp. 185-194, 2019.

[17] A. Mappa and M. D. T. Sogen, "Prototype Sistem Pengendalian Kecepatan dan Pengereman Menggunakan Sensor Jarak," *Jurnal Elektro Luceat,* vol. 5, no. 2, pp. 48-61, 2019.

[18] A. A. Putra, E. Susanto and N. P. Prihat, "Sistem Perekam Kecepatan Sepeda Motor Saat Kecelakaan Menggunakan MICROSD," *e-Proceeding of Engineering,* vol. 8, no. 6, pp. 11479-11484, 2021.

[19] Megatronics, "LM393 Infrared Velometer Optical Encoder Speed Sensor Optocoupler Tacho Sensor 5," [Online]. Available: https://www.arduinopak.com/Prd_Detail.aspx?Prd_ID=20039&Title=LM393%20Infrared%20Velometer%20optical%20encoder%20Speed%20Sensor%2

0Optocoupler%20Tacho%20Sensor%20S. [Accessed 7 Juni 2023].

[20] F. B. Lubis and A. Yanie, "Implementasi Pulse Width Modulation (PWM) Pada Penyaluran Limbah Cair Pupuk Kelapa Sawit Berbasis Arduino," *Jurnal of Electrical Technology,* vol. 7, no. 2, pp. 39-46, 2022.

[21] Alimudin, "Sistem Parkir Cerdas Sederhana Berbasis ARDUINO MEGA 2560 Rev3," *Jurnal Electro Luceat,* vol. 4, no. 1, pp. 1-12, 2018.

[22] L. Elektronika, "Arduino Mega 2560 Mikrokontroler ATMrga 2560," 28 Februari 2017. [Online]. Available: http://www.labelektronika.com/2017/02/arduino-mega-2560-mikrokontroler.html. [Accessed 12 Juni 2023].

[23] F. Wahab, A. Sumardiono, A. R. A. Tahtawi and A. F. A. Mulayari, "Desain dan Purwarupa Fuzzy Logic Control untuk Pengendalian Suhu Ruangan," *Jurnal Teknologi Rekayasa,* vol. 2, no. 1, pp. 1-8, 2017.

[24] Iradiratu and T. W. Saputra, "Penerapan Metode Fuzzy LogicSebagai Pengatur Kecepatan Pada Motor BLDC," *SinarFe7,* vol. 1, no. 1, p. 318–326, 2018.

[25] M. I. Atha'llah and R. Pradana, "Penerapan Metode Fuzzy Logic Sugeno pada Prototype Sistem Kendali Pengereman dengan Menggunakan Arduino," *Seminar Nasional Mahasiswa Fakultas Teknologi Informasi,* vol. 17, no. 1, pp. 107-115, 2022.

[26] B. Davvaz, I. Mukhlash and S. , "Himpunan Fuzzy dan Rough Sets," *Journal of Mathematics and Its Applications,* vol. 18, no. 1, pp. 79-94, 2021.

[27] S. Hartanto, "Implementasi Fuzzy Rule Based System Untuk Klasifikasi Buah Mangga," *e-Journal Techsi,* vol. 9, no. 2, pp. 104-117, 2017.

[28] A. Ikhwan, L. T. Hsb, A. W. Pratiwi and A. Raynaldi, "Penerapan Fuzzy Mamdani untuk Sistem Keputusan Pemilihan Laptop," *Jurnal FASILKOM,* vol. 9, no. 2, pp. 476-483, 2019.

[29] S. Widaningsih, "Analisis Perbandingan Metode Fuzzy Tsukamoto, Mamdani dan Sugeno dalam Pengambilan Keputusan Penentuan Jumlah Distribusi Raskin di Bulog Sub. Divisi Regional (Divre) Cianjur," *Jurnal Informatika dan Manajemen STMIK,* vol. 11, no. 1, pp. 51-65, 2017.

[30] A. I. Ikhwan, M. Badri, M. Andriani and N. Saragih, "Analisis Tingkat Kepuasan Pelanggan Menggunakan Fuzzy Mamdani (Studi Kasus: Busrain Bakery)," *Jurnal Sains dan Komputer*, vol. 18, no. 2, pp. 147-153, 2019.

[31] G. A. Maulana and S. Alam, "Rancang Bangun Sistem Pengereman Otomatis pada Prototype Mobil Berbasis Mikrokontroler Arduino," *E-journal Kajian Teknik Elektro*, vol. 5, no. 1, pp. 42-50, 2020.

LAMPIRAN 1

RIWAYAT HIDUP

DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama Lengkap	: Tegar Prayogi
Tempat Tanggal Lahir	: Simpangkatis, 11 Februari 2002
Alamat Rumah	: Desa Simpangkatis
No. HP	: 085609920588
Email	: tegarprayogi@gmail.com
Jenis Kelamin	: Laki-laki
Agama	: Islam

2. Riwayat Pendidikan

1. SD Negeri 1 Simpang Katis	Lulus 2014
2. SMP Negeri 1 Simpang Katis	Lulus 2017
3. SMA Negeri 3 Pangkalpinang	Lulus 2020
4. Politeknik Manufaktur Negeri Bangka Belitung	2020 - Sekarang

3. Pendidiikan Non Formal

-

Sungailiat, 20 Juli 2023

Tegar Prayogi

DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama Lengkap	: Sri Agustini
Tempat Tanggal Lahir	: Belinyu, 11 Agustus 2002
Alamat Rumah	: JL. YOS SUDARSO PMD
No. HP	: 082372343506
Email	: srgustini11@gmail.com
Jenis Kelamin	: Perempuan
Agama	: Islam

2. Riwayat Pendidikan

1. SD Negeri 1 Belinyu	Lulus 2014
2. SMP Negeri 1 Belinyu	Lulus 2017
3. SMA Negeri 1 Belinyu	Lulus 2020
4. Politeknik Manufaktur Negeri Bangka Belitung	2020 - Sekarang

3. Pendidiikan Non Formal

-

Sungailiat, 20 Juli 2023

Sri Agustini

LAMPIRAN 2

PROGRAM

```
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>  // Library untuk LCD
#include <Fuzzy.h>              // Library FUZZY
LiquidCrystal_I2C lcd(0x27,20,4);  //inisialisasi Library I2C

//// FUZZY ////
Fuzzy *fuzzy = new Fuzzy();
int output;
int outputfuzzy = 12;        //  Pin Untuk motor

//// BLUETOOTH//////
SoftwareSerial modul_bt(0,1);
#define E1 11           //  Pin Untuk motor 1 LR
#define E2 6            //  Pin Untuk motor 2 FB

#define in1 7           // Motor 1 F
#define in2 8           // Motor 1 B
#define in3 9           // Motor 2 L
#define in4 10          // Motor 2 R

#define buzzer 5
#define buzzer2 13
#define Ldepan 3        // Lampu Depan
#define Lbelakang 4     // Lampu Belakang
#define encoderPin 2    // Pin input sensor encoder

char command;           // Int to store app command state.
int Speed = 0;          // kecepatan motor saat di jalankan
int Speed2 = 200;       // kemudi


// RPOGRAM RPM
unsigned int rpm = 0;   // Nilai RPM yang dihitung

// Array untuk menyimpan pin trigger dan echo dari sensor ultrasonik
const int U = 8;
const int triggerPins[U] = {23, 25, 27, 29, 31, 33, 35, 37};
const int echoPins[U] = {22, 24, 26, 28,30, 32, 34, 36};

// untuk mendeteksi halangan
bool terdeteksi_halangan1 = false;
```

```
bool terdeteksi_halangan2 = false;
bool terdeteksi_halangan3 = false;
bool terdeteksi_halangan4 = false;
bool terdeteksi_halangan5 = false;
bool terdeteksi_halangan6 = false;

// Variabel untuk menyimpan nilai terkecil dari dua sensor ultrasonik di bagian depan
  int mindepan = 0;
  int minbelakang = 0;
  int minkanan = 0;
  int minkiri = 0;

 /// RELAY
const int relay1Pin = 38;
const int relay2Pin = 39;
const int saklarPin = 40;
int lampu = 13;

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

  /// Inisialisasi pin LCD
  lcd.init();
  lcd.backlight();

  /// Inisialisasi pin BLUETOOHT
  pinMode(E1, OUTPUT);
  pinMode(E2, OUTPUT);

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  pinMode(buzzer, OUTPUT);
  pinMode(buzzer2, OUTPUT);
  pinMode(Ldepan, OUTPUT);
  pinMode(Lbelakang, OUTPUT);

  // Inisialisasi pin Sensor Kecepatan (RPM)
  pinMode(encoderPin, INPUT_PULLUP);
```

```
 // PROGRAM JARAK
   // Inisialisasi pin sensor ultrasonik
   for (int i = 0; i < 8; i++) {
   pinMode(triggerPins[i], OUTPUT);
   pinMode(echoPins[i], INPUT);
  }
    for (int j = 4 ; j < 8; j++) {
   pinMode(triggerPins[j], OUTPUT);
   pinMode(echoPins[j], INPUT);
  }
    if(!Serial);

  /// RELAY
  // Inisialisasi pin relay sebagai output
  pinMode(relay1Pin, OUTPUT);
  pinMode(relay2Pin, OUTPUT);
  pinMode(saklarPin, INPUT_PULLUP);

  // Set awal kondisi relay (hidup)
  digitalWrite(relay1Pin, LOW);
  digitalWrite(relay2Pin, LOW);

  //// FUZZY
   logika_fuzzy();
   pinMode (outputfuzzy, OUTPUT);
pinMode (lampu, OUTPUT);
}

void loop() {
      baca_Jarak_Ultrasonik();
     jaraktampilLCD();
    baca_kecepatan();
    BluetoothData();
    defuzzify();
    relay();
  }

// Mengirim data melalui modul Bluetooth
void BluetoothData() {
```

```
while (Serial.available() > 0) {
    command = Serial.read();
    // berhenti(); //Initialize with motors stoped.

    switch (command) {
      case 'F':
        maju();
       break;
      case 'B':
        mundur();
        break;
      case 'L':
        Kiri();
        break;
      case 'R':
        Kanan();
         break;
      case 'G':
        majukiri();
         break;
      case 'I':
        majukanan();
         break;
      case 'H':
        mundurkiri();
        break;
      case 'J':
        mundurkanan();
        break;
      case 'S':
        berhenti();
        break;
      case '0':
        Speed = 50;
       break;
      case '1':
        Speed = 75;
       break;
      case '2':
        Speed = 80;
```

```
break;
case '3':
  Speed = 100;
 break;
case '4':
  Speed = 120;
  break;
case '5':
  Speed = 130;
  break;
case '6':
  Speed = 140;
  break;
case '7':
  Speed = 140;
 break;
case '8':
  Speed = 160;
  break;
case '9':
  Speed = 180;
   break;
case 'q':
  Speed = 200 ;
  break;
case 'V':  // horn On
  digitalWrite(buzzer, HIGH);
  break;
case 'v':  // horn Off
  digitalWrite(buzzer, LOW);
  break;
 case 'W':  // Front Light On
  digitalWrite(Ldepan, HIGH);
  break;
case 'w':  // Front Light Off
  digitalWrite(Ldepan, LOW);
  break;
case 'U':  // Back Light On
  digitalWrite(Lbelakang, HIGH);
  break;
case 'u':  // Back Light Off
```

```
      digitalWrite(Lbelakang, LOW);
      break;
    case 'X':  // Hazard On
      digitalWrite(Lbelakang && Ldepan, HIGH);
      break;
    case 'x':  // Hazard Off
      digitalWrite(Lbelakang && Ldepan, LOW);
      break;
   }
 }
  analogWrite(E1, Speed2);
  analogWrite(E2, Speed);
  analogWrite(outputfuzzy, output);
}

void maju() {
 digitalWrite(in1, HIGH);
}

void mundur() {
 digitalWrite(in2, HIGH);
}

void Kiri() {
 digitalWrite(in4, HIGH);
}

void Kanan() {
 digitalWrite(in3, HIGH);
}

void majukiri() {
 digitalWrite(in1, HIGH);
 digitalWrite(in4, HIGH);
}

void majukanan() {
 digitalWrite(in1, HIGH);
 digitalWrite(in3, HIGH);
}
```

```
void mundurkanan() {
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
}
void mundurkiri() {
  digitalWrite(in2, HIGH);
  digitalWrite(in4, HIGH);
}

void berhenti() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void logika_fuzzy(){

  FuzzyInput * jarak = new FuzzyInput(1);            // Instansiasi Objek FuzzyInput

   FuzzySet * sangatdekat1 = new FuzzySet(0,0,30,60);     // Instansiasi Objek FuzzySet
  jarak->addFuzzySet(sangatdekat1);
  FuzzySet * dekat1 = new FuzzySet(50,80,120,150);     // Instansiasi Objek FuzzySet
  jarak->addFuzzySet(dekat1);                       // Memasukkan FuzzySet ke dalam FuzzyInput
  FuzzySet *sedang1 = new FuzzySet(135,150,210,250);       // Instansiasi Objek FuzzySet
  jarak->addFuzzySet(sedang1);                      // Memasukkan FuzzySet ke dalam FuzzyInput
  FuzzySet *jauh1 = new FuzzySet(230,250,280,310);       // Instansiasi Objek FuzzySet
  jarak->addFuzzySet(jauh1);                        // Memasukkan FuzzySet ke dalam FuzzyInput
  FuzzySet *sangatjauh1 = new FuzzySet(300,350,400,400);       // Instansiasi Objek FuzzySet
  jarak->addFuzzySet(sangatjauh1);                      // Memasukkan FuzzySet ke dalam FuzzyInput

  fuzzy->addFuzzyInput(jarak);


  FuzzyInput * kecepatan = new FuzzyInput(2);           // Instansiasi Objek FuzzyInput

  FuzzySet *lambat2 = new FuzzySet(200,200,450,600);         // Instansiasi Objek FuzzySet
  kecepatan->addFuzzySet(lambat2);                    // Memasukkan FuzzySet ke dalam FuzzyInput
  FuzzySet *sedang2 = new FuzzySet(550,600,720,800);       // Instansiasi Objek FuzzySet
  kecepatan->addFuzzySet(sedang2);                    // Memasukkan FuzzySet ke dalam FuzzyInput
  FuzzySet *cepat2 = new FuzzySet(780,800,880,900);        // Instansiasi Objek FuzzySet
  kecepatan->addFuzzySet(cepat2);                     // Memasukkan FuzzySet ke dalam FuzzyInput
```

```
FuzzySet *sangatcepat2 = new FuzzySet(890,975,1000,1000);  // Instansiasi Objek FuzzySet
kecepatan->addFuzzySet(sangatcepat2);                // Memasukkan FuzzySet ke dalam FuzzyInput

fuzzy->addFuzzyInput(kecepatan);                  // Memasukkan FuzzyInput ke dalam Fuzzy

FuzzyOutput *output = new FuzzyOutput(1);          // Instansiasi Objek FuzzyOutput

FuzzySet *out = new FuzzySet(0,0,0,0);          // Instansiasi Objek FuzzySet
output->addFuzzySet(out);
FuzzySet *out1 = new FuzzySet(0,80,110,120);          // Instansiasi Objek FuzzySet
output->addFuzzySet(out1);                     // Memasukkan FuzzySet ke dalam FuzzyOutput
FuzzySet *out2 = new FuzzySet(110,120,130,140);        // Instansiasi Objek FuzzySet
output->addFuzzySet(out2);                     // Memasukkan FuzzySet ke dalam FuzzyOutput
FuzzySet *out3 = new FuzzySet(130,150,160,175);       // Instansiasi Objek FuzzySet
output->addFuzzySet(out3);                     // Memasukkan FuzzySet ke dalam FuzzyOutput
FuzzySet *out4 = new FuzzySet(175,180,190,200);       // Instansiasi Objek FuzzySet
output->addFuzzySet(out4);                     // Memasukkan FuzzySet ke dalam FuzzyOutput

fuzzy->addFuzzyOutput(output);                  // Memasukkan FuzzyOutput ke dalam Fuzzy

 //// FuzzyRule 1
FuzzyRuleAntecedent *sangatdekat1_sangatcepat2 = new FuzzyRuleAntecedent();
sangatdekat1_sangatcepat2->joinWithAND(sangatdekat1, sangatcepat2);
FuzzyRuleConsequent *kecepatan_out11 = new FuzzyRuleConsequent();
kecepatan_out11->addOutput(out);
FuzzyRule *fuzzyRule1 = new FuzzyRule(1, sangatdekat1_sangatcepat2, kecepatan_out11);
fuzzy->addFuzzyRule(fuzzyRule1);

//// FuzzyRule2
FuzzyRuleAntecedent *sangatdekat1_cepat2 = new FuzzyRuleAntecedent();
sangatdekat1_cepat2->joinWithAND(sangatdekat1,cepat2);
FuzzyRuleConsequent *kecepatan_out12 = new FuzzyRuleConsequent();
kecepatan_out12->addOutput(out);
FuzzyRule *fuzzyRule2 = new FuzzyRule(2, sangatdekat1_cepat2, kecepatan_out12);
fuzzy->addFuzzyRule(fuzzyRule2);

//// FuzzyRule 3
FuzzyRuleAntecedent *sangatdekat1_sedang2 = new FuzzyRuleAntecedent();
sangatdekat1_sedang2->joinWithAND(sangatdekat1,sedang2);
FuzzyRuleConsequent *kecepatan_out13 = new FuzzyRuleConsequent();
kecepatan_out13->addOutput(out);
```

```
FuzzyRule *fuzzyRule3 = new FuzzyRule(3, sangatdekat1_sedang2, kecepatan_out13);
fuzzy->addFuzzyRule(fuzzyRule3);

//// FuzzyRule 4
FuzzyRuleAntecedent *sangatdekat1_lambat2 = new FuzzyRuleAntecedent();
sangatdekat1_lambat2->joinWithAND(sangatdekat1,lambat2);
FuzzyRuleConsequent *kecepatan_out14 = new FuzzyRuleConsequent();
kecepatan_out14->addOutput(out);
FuzzyRule *fuzzyRule4 = new FuzzyRule(4, sangatdekat1_lambat2, kecepatan_out14);
fuzzy->addFuzzyRule(fuzzyRule4);


//// FuzzyRule 6
FuzzyRuleAntecedent *dekat1_sangatcepat2 = new FuzzyRuleAntecedent();
dekat1_sangatcepat2->joinWithAND(dekat1,sangatcepat2);
  FuzzyRuleConsequent *kecepatan_out21 = new FuzzyRuleConsequent();
kecepatan_out21->addOutput(out1);
FuzzyRule *fuzzyRule6 = new FuzzyRule(6, dekat1_sangatcepat2, kecepatan_out21);
fuzzy->addFuzzyRule(fuzzyRule6);


//// FuzzyRule 7
FuzzyRuleAntecedent *dekat1_cepat2 = new FuzzyRuleAntecedent();
dekat1_cepat2->joinWithAND(dekat1,cepat2);
FuzzyRuleConsequent *kecepatan_out22 = new FuzzyRuleConsequent();
kecepatan_out22->addOutput(out1);
FuzzyRule *fuzzyRule7 = new FuzzyRule(7, dekat1_cepat2, kecepatan_out22);
fuzzy->addFuzzyRule(fuzzyRule7);

//// FuzzyRule 8
FuzzyRuleAntecedent *dekat1_sedang2 = new FuzzyRuleAntecedent();
dekat1_sedang2->joinWithAND(dekat1,sedang2);
FuzzyRuleConsequent *kecepatan_out23 = new FuzzyRuleConsequent();
kecepatan_out23->addOutput(out1);
FuzzyRule *fuzzyRule8 = new FuzzyRule(8, dekat1_sedang2,kecepatan_out23);
fuzzy->addFuzzyRule(fuzzyRule8);

//// FuzzyRule 9
FuzzyRuleAntecedent *dekat1_lambat2 = new FuzzyRuleAntecedent();
dekat1_lambat2->joinWithAND(dekat1,lambat2);
FuzzyRuleConsequent *kecepatan_out24 = new FuzzyRuleConsequent();
```

```
kecepatan_out24->addOutput(out1);
FuzzyRule *fuzzyRule9 = new FuzzyRule(9, dekat1_lambat2, kecepatan_out24);
fuzzy->addFuzzyRule(fuzzyRule9);

 //// FuzzyRule 11
FuzzyRuleAntecedent *sedang1_sangatcepat2 = new FuzzyRuleAntecedent();
sedang1_sangatcepat2->joinWithAND(sedang1,sangatcepat2);
FuzzyRuleConsequent *kecepatan_out31 = new FuzzyRuleConsequent();
kecepatan_out31->addOutput(out2);
FuzzyRule *fuzzyRule11 = new FuzzyRule(11, sedang1_sangatcepat2, kecepatan_out31);
fuzzy->addFuzzyRule(fuzzyRule11);

 //// FuzzyRule 12
FuzzyRuleAntecedent *sedang1_cepat2 = new FuzzyRuleAntecedent();
sedang1_cepat2->joinWithAND(sedang1,cepat2);
FuzzyRuleConsequent *kecepatan_out32 = new FuzzyRuleConsequent();
kecepatan_out32->addOutput(out2);
FuzzyRule *fuzzyRule12 = new FuzzyRule(12, sedang1_cepat2, kecepatan_out32);
fuzzy->addFuzzyRule(fuzzyRule12);

 //// FuzzyRule 13
FuzzyRuleAntecedent *sedang1_sedang2 = new FuzzyRuleAntecedent();
sedang1_sedang2->joinWithAND(sedang1,sedang2);
FuzzyRuleConsequent *kecepatan_out33 = new FuzzyRuleConsequent();
kecepatan_out33->addOutput(out2);
FuzzyRule *fuzzyRule13 = new FuzzyRule(13, sedang1_sedang2, kecepatan_out33);
fuzzy->addFuzzyRule(fuzzyRule13);

 //// FuzzyRule 14
FuzzyRuleAntecedent *sedang1_lambat2 = new FuzzyRuleAntecedent();
sedang1_lambat2->joinWithAND(sedang1,lambat2);
FuzzyRuleConsequent *kecepatan_out34 = new FuzzyRuleConsequent();
kecepatan_out34->addOutput(out2);
FuzzyRule *fuzzyRule14 = new FuzzyRule(14, sedang1_lambat2, kecepatan_out34);
fuzzy->addFuzzyRule(fuzzyRule14);


//// FuzzyRule 16
FuzzyRuleAntecedent *jauh1_sangatcepat2 = new FuzzyRuleAntecedent();
jauh1_sangatcepat2->joinWithAND(jauh1,sangatcepat2);
FuzzyRuleConsequent *kecepatan_out41 = new FuzzyRuleConsequent();
```

```
kecepatan_out41->addOutput(out3);
FuzzyRule *fuzzyRule16 = new FuzzyRule(16, jauh1_sangatcepat2, kecepatan_out41);
fuzzy->addFuzzyRule(fuzzyRule16);

 //// FuzzyRule 17
FuzzyRuleAntecedent *jauh1_cepat2 = new FuzzyRuleAntecedent();
jauh1_cepat2->joinWithAND(jauh1,cepat2);
FuzzyRuleConsequent *kecepatan_out42 = new FuzzyRuleConsequent();
kecepatan_out42->addOutput(out3);
FuzzyRule *fuzzyRule17 = new FuzzyRule(17, jauh1_cepat2, kecepatan_out42);
fuzzy->addFuzzyRule(fuzzyRule17);

 //// FuzzyRule 18
FuzzyRuleAntecedent *jauh1_sedang2 = new FuzzyRuleAntecedent();
jauh1_sedang2->joinWithAND(jauh1,sedang2);
FuzzyRuleConsequent *kecepatan_out43 = new FuzzyRuleConsequent();
kecepatan_out43->addOutput(out3);
FuzzyRule *fuzzyRule18 = new FuzzyRule(18, jauh1_sedang2, kecepatan_out43);
fuzzy->addFuzzyRule(fuzzyRule18);

 //// FuzzyRule 19
FuzzyRuleAntecedent *jauh1_lambat2 = new FuzzyRuleAntecedent();
jauh1_lambat2->joinWithAND(jauh1,lambat2);
FuzzyRuleConsequent *kecepatan_out44 = new FuzzyRuleConsequent();
kecepatan_out44->addOutput(out3);
FuzzyRule *fuzzyRule19 = new FuzzyRule(19, jauh1_lambat2, kecepatan_out44);
fuzzy->addFuzzyRule(fuzzyRule19);


 //// FuzzyRule 21
FuzzyRuleAntecedent *sangatjauh1_sangatcepat2 = new FuzzyRuleAntecedent();
sangatjauh1_sangatcepat2->joinWithAND(sangatjauh1,sangatcepat2);
FuzzyRuleConsequent *kecepatan_out51 = new FuzzyRuleConsequent();
kecepatan_out51->addOutput(out4);
FuzzyRule *fuzzyRule21 = new FuzzyRule(21, sangatjauh1_sangatcepat2, kecepatan_out51);
fuzzy->addFuzzyRule(fuzzyRule21);

 //// FuzzyRule 22
FuzzyRuleAntecedent *sangatjauh1_cepat2 = new FuzzyRuleAntecedent();
sangatjauh1_cepat2->joinWithAND(sangatjauh1,cepat2);
FuzzyRuleConsequent *kecepatan_out52 = new FuzzyRuleConsequent();
```

```
  kecepatan_out52->addOutput(out4);
  FuzzyRule *fuzzyRule22 = new FuzzyRule(22, sangatjauh1_cepat2, kecepatan_out52);
  fuzzy->addFuzzyRule(fuzzyRule22);

   //// FuzzyRule 23
  FuzzyRuleAntecedent *sangatjauh1_sedang2 = new FuzzyRuleAntecedent();
  sangatjauh1_sedang2->joinWithAND(sangatjauh1,sedang2);
  FuzzyRuleConsequent *kecepatan_out53 = new FuzzyRuleConsequent();
  kecepatan_out53->addOutput(out4);
  FuzzyRule *fuzzyRule23 = new FuzzyRule(23, sangatjauh1_sedang2, kecepatan_out53);
  fuzzy->addFuzzyRule(fuzzyRule23);

   //// FuzzyRule 24
  FuzzyRuleAntecedent *sangatjauh1_lambat2 = new FuzzyRuleAntecedent();
  sangatjauh1_lambat2->joinWithAND(sangatjauh1,lambat2);
  FuzzyRuleConsequent *kecepatan_out54 = new FuzzyRuleConsequent();
  kecepatan_out54->addOutput(out4);
  FuzzyRule *fuzzyRule24 = new FuzzyRule(24, sangatjauh1_lambat2, kecepatan_out54);
  fuzzy->addFuzzyRule(fuzzyRule24);

}

void defuzzify(){
  fuzzy->setInput(1,mindepan);
  fuzzy->setInput(2,rpm);
  // Running the Fuzzification
  fuzzy->fuzzify();

  output = fuzzy->defuzzify(1);
  // Printing something
//  Serial.println("Result: ");
//  Serial.print("\tkecepatan: ");
//  Serial.println(output);
  lcd.setCursor(0,3);
  lcd.print("FLC:");
  lcd.setCursor(4,3);
  lcd.print(output);
  lcd.print("  ");
  lcd.setCursor(8,3);
  lcd.print("PWM");
}
```

```
void jaraktampilLCD(){

int jarakdpn1;
int jarakdpn2;
int jarakblk1;
int jarakblk2;
int jarakknn1;
int jarakknn2;
int jarakkri1;
int jarakkri2;

 // Mencari nilai terkecil dari dua sensor ultrasonik di bagian depan
 for (int i = 0; i < 8; i++) {
  jarakdpn1 = hitungjarak(triggerPins[0], echoPins[0]);
  jarakdpn2 = hitungjarak(triggerPins[1], echoPins[1]);
  jarakblk1 = hitungjarak(triggerPins[2], echoPins[2]);
  jarakblk2 = hitungjarak(triggerPins[3], echoPins[3]);
  jarakknn1 = hitungjarak(triggerPins[4], echoPins[4]);
  jarakknn2 = hitungjarak(triggerPins[5], echoPins[5]);
  jarakkri1 = hitungjarak(triggerPins[6], echoPins[6]);
  jarakkri2 = hitungjarak(triggerPins[7], echoPins[7]);
 }
 if (jarakdpn1 > 0 || jarakdpn2 > 0 || jarakblk1 > 0 || jarakblk2 > 0
 || jarakknn1 > 0 || jarakknn1 > 0 || jarakkri1 > 0 || jarakkri2 > 0) {

 mindepan = min(jarakdpn1, jarakdpn2);
 minbelakang = min(jarakblk1, jarakblk2);
 minkanan = min(jarakknn1, jarakknn2);
 minkiri = min(jarakkri1, jarakkri2);

 lcd.setCursor(1,1);
 lcd.print("knn:");
 lcd.setCursor(5,1);
 lcd.print(minkanan);
 lcd.print("  ");
 lcd.setCursor(8,1);
 lcd.print("cm");

 lcd.setCursor(11,1);
 lcd.print("kri:");
 lcd.setCursor(15,1);
```

```
  lcd.print(minkiri);
  lcd.print("  ");
  lcd.setCursor(18,1);
  lcd.print("cm");

  lcd.setCursor(11,2);
  lcd.print("blk:");
  lcd.setCursor(15,2);
  lcd.print(minbelakang);
  lcd.print("  ");
  lcd.setCursor(18,2);
  lcd.print("cm");

  lcd.setCursor(1,2);
  lcd.print("dpn:");
  lcd.setCursor(5,2);
  lcd.print(mindepan);
  lcd.print("  ");
  lcd.setCursor(8,2);
  lcd.print("cm");
}
}
 int hitungjarak(int triggerPins, int echoPins) {
  // Mengirim sinyal ultrasonik
  digitalWrite(triggerPins, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPins, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPins, LOW);

  // Menerima echo
  unsigned long durasilcd = pulseIn(echoPins, HIGH);

  // Menghitung jarak berdasarkan waktu
  int jarak_lcd = durasilcd * 0.034 / 2;  // Rumus: waktu (us) * kecepatan suara (cm/us) / 2

  return jarak_lcd;
}

void baca_Jarak_Ultrasonik() {
```

```
int Jarak[U];
int Jarak1[U];

// Mengukur jarak menggunakan setiap sensor
for (int j = 4; j < 8; j++) {
  digitalWrite(triggerPins[j], LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPins[j], HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPins[j], LOW);
  long durasi1 = pulseIn(echoPins[j], HIGH);
  Jarak1[j] = durasi1 * 0.034 / 2;

  // Menambahkan nilai jarak ke variabel rata-rata yang sesuai
}

// Mengukur jarak menggunakan setiap sensor
for (int i = 0; i < 4; i++) {
  digitalWrite(triggerPins[i], LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPins[i], HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPins[i], LOW);
  long durasi = pulseIn(echoPins[i], HIGH);
  Jarak[i] = durasi * 0.034 / 2;        // Rumus: waktu * kecepatan suara (cm/us) / 2
}

// Memeriksa apakah ada sensor yang mendeteksi jarak yang di sisi kiri dan kanan
for (int j = 4; j < 7; j++) {
  if (Jarak1[j] <= 30) { // Jarak yang terlalu dekat (misalnya, < 20 cm)
    terdeteksi_halangan4 = true;
    digitalWrite(buzzer,HIGH);  // Berkedip buzzer terus
    delay(5000);
    digitalWrite(buzzer,LOW);
    break;
  }
  else if (Jarak1[j] <= 50) { // Jarak
    terdeteksi_halangan5 = true;
    blinkBuzzer(100); // Berkedip buzzer dengan delay 100 ms
    break;
  }
```

```
  else if (Jarak1[j] <= 75) { // Jarak
    terdeteksi_halangan6 = true;
    blinkBuzzer(300); // Berkedip buzzer dengan delay 300 ms
    break;
  } else if (Jarak1[j] >= 75) { // Jarak
    terdeteksi_halangan4 = false;
    terdeteksi_halangan5 = false;
    terdeteksi_halangan6 = false;
    digitalWrite(buzzer,LOW);
  }
 }
 // Memeriksa apakah ada sensor yang mendeteksi jarak yang di sisi depan dan belakang
 for (int i = 0; i < 3; i++) {
  if (Jarak[i] <= 50) { // Jarak yang terlalu dekat (misalnya, < 20 cm)
    terdeteksi_halangan1 = true;
    digitalWrite(buzzer,HIGH);  // Berkedip buzzer terus
    break;
  }
  else if (Jarak[i] <= 100) { // Jarak
    terdeteksi_halangan2 = true;
    blinkBuzzer(100); // Berkedip buzzer dengan delay 100 ms
    break;
  }
  else if (Jarak[i] <= 150) { // Jarak
    terdeteksi_halangan3 = true;
    blinkBuzzer(300); // Berkedip buzzer dengan delay 300 ms
    break;
  } else if (Jarak[i] >= 150) { // Jarak
    terdeteksi_halangan1 = false;
    terdeteksi_halangan2 = false;
    terdeteksi_halangan3 = false;
    digitalWrite(buzzer,LOW);
  }
 }


if ( terdeteksi_halangan1 || terdeteksi_halangan2 || terdeteksi_halangan3
|| terdeteksi_halangan4 || terdeteksi_halangan5 || terdeteksi_halangan6){
    lcd.setCursor(12,3);
    lcd.print("BAHAYA");
    lcd.print("!!");
```

```
  } else if ( !terdeteksi_halangan1 && !terdeteksi_halangan2 && !terdeteksi_halangan3
  && !terdeteksi_halangan4 && !terdeteksi_halangan5 && !terdeteksi_halangan6 ) {
     lcd.setCursor(12,3);
     lcd.print("AMAN");
     lcd.print("..");
     lcd.print("  ");
  }
}

 // Fungsi untuk mengatur berkedipnya buzzer dengan delay tertentu
void blinkBuzzer(int delayTime){
  digitalWrite(buzzer,LOW);
  delay(delayTime);
  digitalWrite(buzzer,HIGH);
  delay(delayTime);
  digitalWrite(buzzer,LOW);
  delay(delayTime);
  digitalWrite(buzzer,HIGH);
  delay(delayTime);
  digitalWrite(buzzer,LOW);
  delay(delayTime);
}

 //// RELAY

               //// RELAY
                void relay(){

               int Status_saklar = digitalRead(saklarPin);
              //  //// Membandingkan nilai a dan b
              if(Status_saklar == 0) {
                 if ( Speed > output) {
                 digitalWrite(relay1Pin, HIGH);  // Matikan relay 1
                 digitalWrite(relay2Pin, LOW);   // Nyalakan relay 2
                 digitalWrite(Lbelakang, HIGH);
                  if (output == 0){
                 digitalWrite(in2, HIGH);
                 digitalWrite(in1, LOW);
                 analogWrite(E2,50);
                 delay(500);
                 digitalWrite(in2, LOW);
```

```
    digitalWrite(in1, LOW);
   }
    } else {
    digitalWrite(relay1Pin, LOW);   // Nyalakan relay 1
    digitalWrite(relay2Pin, HIGH);   // Nyalakan relay 2
    digitalWrite(Lbelakang, LOW);
   }
   digitalWrite(lampu, HIGH);
   } else {
   digitalWrite(relay1Pin, LOW);   // Nyalakan relay 1
   digitalWrite(relay2Pin, HIGH);   // Nyalakan relay 2
   digitalWrite(lampu, LOW);
   digitalWrite(Lbelakang, LOW);
   }

}
//// KECEPATAN (RPM)
void baca_kecepatan() {

 unsigned long waktu_mulai = 0;   //   mengukur waktu saat ini dan menghitung durasi perhitungan langkah selama 1 detik
 unsigned long waktu_akhir = 0;   //   mengukur waktu saat ini dan menghitung durasi perhitungan langkah selama 1 detik
  int langkah = 0 ;              //   menyimpan jumlah langkah yang telah dihitung
  int langkah_akhir = 0;             //   menyimpan jumlah langlah sebelumnya untuk menghitung perubahan langkah dalam 1 detik
  int waktu = 0;                //   menyimpan perubahan langkah dalam 1 detik
  int rpm = 0;

waktu_mulai = millis();
waktu_akhir = waktu_mulai + 1000;  /// menunjukkan waktu akhir penghitungan selama satu detik.
while(millis() < waktu_akhir ){
  if(digitalRead(encoderPin)){    /// Jika sensor mendeteksi putaran atau gerakan (dalam kondisi HIGH),
    langkah = langkah + 1;     /// variabel "langkah" diincrement (ditambah satu) dari "langkah_akhir".
    while(digitalRead(encoderPin)){   /// Program juga menunggu hingga sensor kembali ke kondisi LOW sebelum melanjutkan penghitungan langkah selanjutnya.
      if (millis()-waktu_mulai > 1000){     ///  untuk memastikan bahwa loop penghitungan langkah tidak berjalan lebih dari satu detik.
        break;
      }
    }
```

```
  }
}

waktu = langkah - langkah_akhir;
langkah_akhir = langkah ;
rpm = ((waktu * 60) / 20);   //waktu dikali 60 detik dibagi dengan jumlah lubang piringan encoder


  lcd.setCursor(0,0);
  lcd.print("Kec:");
  lcd.print(rpm);
  lcd.print("  ");
  lcd.setCursor(8,0);
  lcd.print("RPM");
  lcd.setCursor(12,0);
  lcd.print("PWM:");
  lcd.print(Speed);
  lcd.print("  ");

}
```