METROTECH
Journal of Mechanical and Electrical Technology
Vol. 4 No. 1, Januari 2025, E-ISSN 2809-9605 | P-ISSN 2827-8097

Implementasi Modular Papan Kontroler *Swerve drive*: Komunikasi I2C

Rian Egar Pramanta[1*], Indrazno Siradjuddin[2], Totok Winarno[3]
[1,2,3]Politeknik Negeri Malang, Malang
[*]Penulis Korespondensi, email: rianpramanta@gmail.com

Received:01/09/2024 *Revised:08/09/2024* *Accepted:04/10/2024*

Abstract. *The drive system for robots, especially mobile robots, is the most important thing. In mobile robots that use a swerve drive system, an innovation has been found to overcome the problem of being able to control the swerve drive wheel in real-time via I2C communication. This system consists of a master and slave microcontroller, where the master microcontroller is responsible for sending RPM and position data to the slave microcontroller. The slave microcontroller then sends the data to the VESC to regulate the steering and driving of the swerve drive wheel. The development process involves designing hardware and software that supports I2C communication as well as implementing PID control to achieve stability and accuracy in controlling speed and wheel position. The PID values used in steering are Kp=0.03, Ki=0, and Kd=0.0007. Meanwhile, the PID values used in driving are Kp=0.022, Ki=0.03, and Kd=0.00012. Test results show that this system is able to provide a fast and stable response to changes in set point with minimal steady-state error. The system also shows that I2C communication between microcontrollers can also run in real-time, the total transfer time between master-slave and slave-master is 0.002 seconds and the transfer speed is 8000 bytes per second.*

Keywords: *Swerve drive, I2C, PID control, Microcontroller, Modular*

Abstrak. Sistem penggerak pada robot khususnya *mobile* robot merupakan hal paling penting. Pada *mobile robot* yang menggunakan sistem penggerak *swerve drive* ditemukan inovasi mengatasi masalah agar dapat mengendalikan roda *swerve drive* secara real-time melalui komunikasi I2C. Sistem ini terdiri dari mikrokontroler master dan *slave*, di mana mikrokontroler master bertanggung jawab mengirimkan data RPM dan posisi ke mikrokontroler *slave*. Mikrokontroler *slave* kemudian mengirimkan data tersebut ke VESC untuk mengatur *steering* dan *driving* roda *swerve drive*. Proses pengembangan melibatkan perancangan perangkat keras dan perangkat lunak yang mendukung komunikasi I2C serta implementasi kontrol PID untuk mencapai stabilitas dan akurasi dalam pengendalian kecepatan dan posisi roda. Nilai PID pada *steering* adalah Kp=0.03, Ki=0, dan Kd=0.0007. Sedangkan nilai PID pada *driving* adalah Kp=0.022, Ki=0.03, dan Kd=0.00012. Hasil pengujian menunjukkan bahwa sistem ini mampu memberikan respon cepat dan stabil terhadap perubahan *set point* dengan *error* steady-state yang minimal. Dalam sistem juga menunjukkan bahwa komunikasi I2C antar mikrokontroler juga dapat berjalan secara real-time, total waktu transfer antar master-*slave* dan *slave*-master adalah 0.002 *seconds* dan kecepatan transfernya 8000 *bytes per seconds.*

Kata Kunci: *Swerve drive*, I2C, Kontrol PID, Mikrokontroler, Modular

I. PENDAHULUAN

Penelitian robot semakin menarik karena kemajuan teknologi, memungkinkan penggunaan robot di berbagai bidang seperti kesehatan, bisnis, eksplorasi lingkungan, penyelamatan, pendidikan, dan militer. Salah satu studi yang menonjol adalah tentang *mobile robot*, yaitu robot yang bergerak menggunakan roda untuk berpindah dari satu titik ke titik lain [1] [2] [3] [4]. Robot dapat dibagi menjadi dua jenis berdasarkan pergerakannya yaitu *holonomic* dan *non-holonomic*. *Wheeled Mobile Robot* (WMR) konvensional adalah *non-holonomic*, karena tidak dapat bergerak ke samping tanpa manuver awal, berbeda dengan *holonomic* robot yang dapat bergerak ke segala arah [5]. Salah satu tipe penggerak *holonomic* yang menonjol adalah *swerve drive*, yang memiliki kelebihan dalam fleksibilitas dan akurasi. *Swerve drive* terdiri dari dua motor per modul untuk *steering* dan *driving*, serta dilengkapi dengan *gearbox*, *encoder*, dan roda, memungkinkan robot bergerak dengan mudah ke segala arah [6].

Papan kontroler *swerve drive* merupakan komponen penting untuk mengendalikan kecepatan

dan arah putaran roda. Namun, papan kontroler *swerve drive* yang tersedia di pasaran jarang ditemukan dan umumnya tidak modular, sehingga sulit diintegrasikan dengan kontroler dan komponen lain seperti sensor dan aktuator. Sebagian besar papan kontroler *swerve drive* menggunakan komunikasi serial atau USB, yang memiliki keterbatasan pada modul mikrokontroler dan kurangnya fitur seperti indikator. Diperlukan desain papan kontroler yang modular dengan komunikasi I2C, karena I2C memiliki kecepatan transfer data tinggi, konsumsi daya rendah, dan kompatibilitas luas dengan mikrokontroler, sehingga cocok untuk sistem robotik.

Penelitian ini menggunakan ESP32 karena memiliki prosesor *dual-core* 240 MHz, memungkinkan multitasking, serta konsumsi daya rendah yang cocok untuk sistem robotik dengan daya tahan baterai lama. ESP32 juga dilengkapi dengan Wi-Fi, Bluetooth, sensor *in-built*, dan berbagai port komunikasi seperti I2C, UART, dan SPI, membuatnya ideal untuk aplikasi robotika. Komunikasi I2C menawarkan kecepatan transfer data tinggi, konsumsi daya rendah, dan kemudahan implementasi. I2C dapat digunakan untuk mengontrol mekanisme *swerve drive* dengan konfigurasi *half-duplex*, membutuhkan satu master dan beberapa *slave*, serta menggunakan pin SDA dan SCL untuk koneksi [7] [8].

Modular papan kontroler *swerve drive* yang dibuat memiliki 7 tombol dan layar *OLED* untuk tampilan antarmuka, memudahkan pemantauan dan navigasi. Tombol-tombol ini berfungsi sebagai *cursor* untuk mengatur tampilan, memberikan kontrol yang jelas terhadap gerakan motor *swerve drive*. Sistem ini mendukung pengaturan kecepatan pada *Electronic Speed Controller* (ESC) melalui komunikasi I2C, memungkinkan pengguna untuk mengontrol kecepatan motor dan posisi *steering* dengan mudah. Sensor *encoder* juga dapat diatur untuk mengukur putaran *steering* dan memperoleh informasi posisi yang akurat. Modular ini mempermudah pengembang dalam menyesuaikan parameter sesuai kebutuhan robot, dengan *VESC* yang sudah terintegrasi dengan PID. Parameter seperti kecepatan dan posisi *steering* dapat diubah melalui *library* pada ESC FSESC 4.12, memungkinkan optimalisasi kinerja *swerve drive*. Dengan kontrol PID terintegrasi, robot dapat mencapai stabilitas dan respon yang tinggi.

Pada penelitian yang dilakukan oleh Bagus dkk yang berfokus pada desain dan kontrol aktuator robot *mobile* modular dengan sistem *swerve drive*, menggunakan kontrol PID untuk mengatur kecepatan penggerak dan posisi kemudi. Hasilnya menunjukkan bahwa kontrol PID secara signifikan meningkatkan kinerja dan akurasi sistem, dengan tingkat kesalahan rata-rata hanya 2.54% untuk kecepatan penggerak dan 0.91% untuk posisi kemudi. Implementasi kontrol PID memberikan respon yang akurat dan bebas *overshoot*, serta meningkatkan kinerja robot dalam berbagai pengaturan dibandingkan dengan kontrol P atau PI [9]. Kemudian, pada penelitian oleh Nurul Achmadiah dkk yang bertujuan meningkatkan kinerja sistem *swerve drive* dengan mengurangi kesalahan dan meningkatkan respons pengendalian dan kemudi melalui penggunaan kontrol *FOC*. Penelitian mencakup desain skematik *main board* modul *swerve drive*, konfigurasi elektronik, dan perancangan mekanik. Hasil pengujian menunjukkan bahwa kontrol *FOC* mengurangi *error* sebesar 2% pada motor *driving* dan 0.7% pada motor *steering*, serta memperbaiki respons *driving* dan *steering*, sehingga meningkatkan performa sistem secara keseluruhan [10].

Sehingga sistem *swerve drive* ini bertujuan untuk memberikan pengalaman pengguna yang lebih baik dan kontrol yang efisien dalam mengoperasikan roda *swerve drive* melalui komunikasi I2C. Modularitasnya meningkatkan fleksibilitas dan adaptabilitas sistem kendali, menjadikannya solusi ideal bagi pengembang robot untuk mencapai kinerja optimal dalam berbagai aplikasi robotika.

II. METODOLOGI

A. Diagram Blok Sistem

Diagram blok kontrol penelitian akan menjelaskan mengenai prinsip kerja dari sistem

Modular Papan Kontroler *swerve drive*. Berikut merupakan diagram blok sistem yang mencakup perangkat *Input*, Proses, dan *Output*.

1. Input

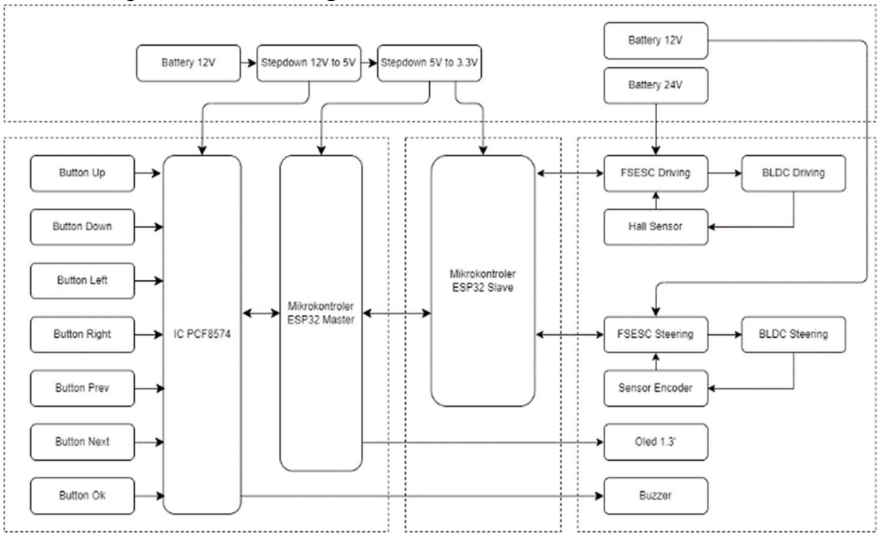
Mikrokontroler *master* ESP32 menggunakan I2C untuk berkomunikasi dengan mikrokontroler *slave*, mengendalikan *steering* dan *driving* pada *swerve drive*. Terdapat 7 *push button* (PB UP, PB DOWN, PB LEFT, PB RIGHT, PB PREV, PB NEXT, PB OK) untuk navigasi *interface*, dan sensor rotary *encoder* untuk mengontrol posisi *steering*.

2. Process

Mikrokontroler *slave* ESP32 menggunakan tegangan 3V untuk komputasi data, mengontrol posisi *steering* dan *driving* berdasarkan perintah dari mikrokontroler *master* melalui I2C. *Slave* juga menerima data RPM, POS, dan Volt dari *FSESC*.

3. Output

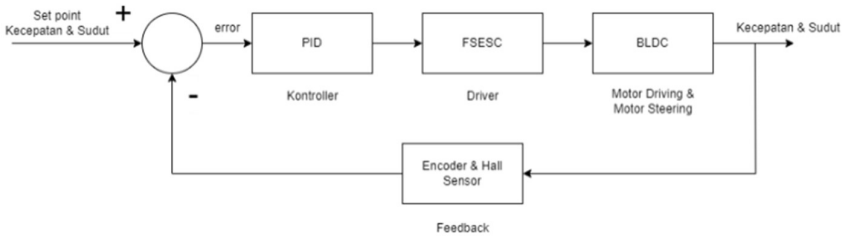
FSESC untuk *driving* dan *steering* berfungsi sebagai *driver* motor BLDC, berkomunikasi dengan mikrokontroler *slave* melalui UART. Motor BLDC digunakan sebagai aktuator, *encoder* sebagai sensor posisi sudut *steering*, dan *hall sensor* untuk kecepatan *driving*. Data sensor dikirim ke modul *slave*, lalu ke modul *master* untuk ditampilkan pada *OLED display*, yang menampilkan informasi kecepatan, sudut motor, dan tegangan.



Gambar 1. Diagram blok sistem

B. Perancangan Kontrol PID

Perancangan PID kontrol motor BLDC dilakukan pada 2 tahap yaitu PID untuk motor *steering* (Motor BLDC 5045) dan PID motor *driving* (Motor BLDC 70mm). Perancangan melalui proses simulasi dibuat dengan menggunakan *FOC* (*Field Oriented Control*) BLDC dan *state space* BLDC. Kemudian hasil respon sistem akan dikontrol menggunakan PID secara *trial and error*. kemudian nilai Ki, Kp, dan Kd tersebut akan dijadikan referensi nilai PID yang akan di*Input*kan pada motor secara *real-time* pada sofware *VESC-tool*.



Gambar 2. Diagram blok sistem perancangan kontrol PID

C. Perancangan Mekanik

Perancangan mekanik terdapat 2 motor yaitu pada *steering* sebagai penggerak rotasi dan *driving* sebagai penggerak translasi. Terdapat beberapa komponen pada perancangan mekanik yaitu motor BLDC, rotary *encoder*, dan *gear*.

1. Perancangan Rasio *Gear*

Pada pembuatan *gear steering*, metode *spiral bevel gear* digunakan karena memiliki kelebihan dalam transmisi daya, efisiensi, dan kebisingan yang rendah. Modular *swerve drive* ini menerapkan rasio *gear* pada bagian *steering* dan *encoder* untuk memastikan keakuratan pembacaan sudut. *Steering* memiliki rasio 1:4 dengan 120 gigi pada *gear* utama dan 30 gigi pada *gear* motor *steering*, sedangkan *encoder* memiliki rasio 1:3 dengan 120 gigi pada *gear* utama dan 40 gigi pada *shaft encoder*.

$$rasio\ gear\ steering = \frac{n_{output}}{n_{input}} \quad (1)$$

$$rasio\ gear\ steering = \frac{120}{30} \quad (2)$$

$$rasio\ gear\ steering = 4 \quad (3)$$

Persamaan di atas memiliki arti bahwa empat putaran pada *gear* motor sama dengan 1 putara pada *gear* modul. Hal ini bisa membuat bertambahnya torsi pada *steering*. Berikut persamaan torsi pada *steering*.

$$torsi\ steerin = rasio\ gear\ steering \times torsi\ motor \quad (4)$$

$$torsi\ steering = 4 \times 0.85 \quad (5)$$

$$torsi\ steering = 3.4 \quad (6)$$

Pada *encoder* juga diberi rasio *gear*. Tujuan pemberian rasio *gear* pada *encoder* adalah ketidakmungkinan pembacaan sudut *steering* langsung dari *gear* modul. Oleh karena itu dibutuhkan *gear* pada *encoder* agar bisa terbaca dengan baik. Berikut perhitungan rasio *gear* pada *encoder*.

$$rasio\ gear\ encoder = \frac{n_{output}}{n_{input}} \quad (7)$$

$$rasio\ gear\ encoder = \frac{120}{40} \quad (8)$$

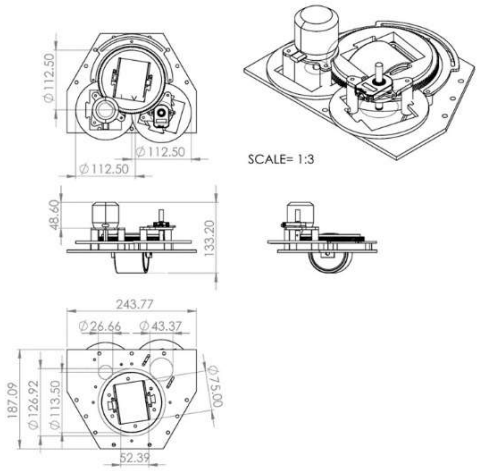
$$rasio\ gear\ encoder = 3 \quad (9)$$

Persamaan di atas memiliki arti bahwa tiga putaran pada *gear encoder* sama dengan 1 putaran pada *gear* modul. Pada *encoder* menggunakan rasio *gear* 1:3 dengan tujuan mempermudah perhitungan *encoder* dan mengantisipasi data *losing* pada saat pembacaan *encoder* oleh *ESC*.

2. Spesifikasi Mekanik

Berikut adalah spesifikasi dari mekanik *Drivetrain*:

a. Panjang : 19cm
b. Lebar : 21cm
c. Tinggi : 30 cm
d. Berat *Drivetrain* : 1kg
e. Jenis Roda : Motor BLDC 5045 (*Steering*), Motor BLDC 70mm (*Driving*)
f. Bahan *Base* : Plat Alumunium
g. Bahan *Gear* : PETG



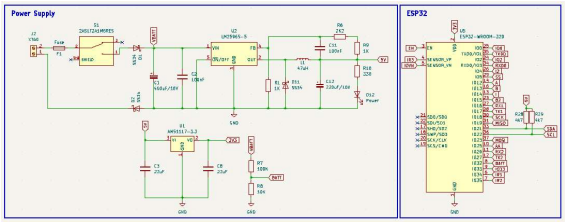
Gambar 3. Spesifikasi mekanik (a & c Tampak atas, b. Tampak depan, d. Tampak samping atas, e. Tampak samping)

D. Perancangan Elektronik

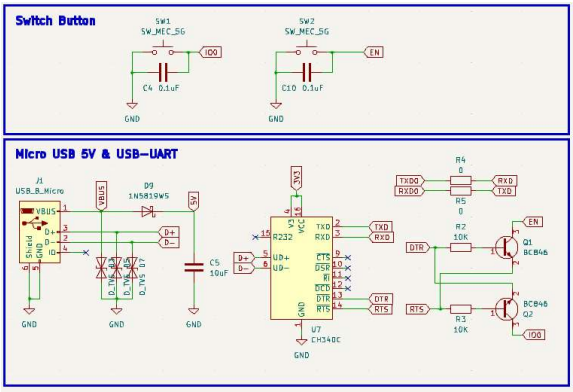
Perancangan elektrik merupakan perancangan rangkaian elektrik dari *main board* modul *master* dan *slave* yaitu *power management, interface, push button, buzzer*, mikrokontroler ESP32, dan pin pin GPIO untuk penunjang komunikasi dan lainnya.

Perancangan modul *master* dan *slave* mencakup pengiriman data melalui *interface* yang tersedia, dengan skematik mencakup mikrokontroler utama dan manajemen daya. Sistem *power management* terdiri dari konektor baterai, *fuse* 3A untuk proteksi arus berlebih, saklar *on/off*, rangkaian *step-down* 5V 3A untuk menurunkan tegangan dari baterai 12V,

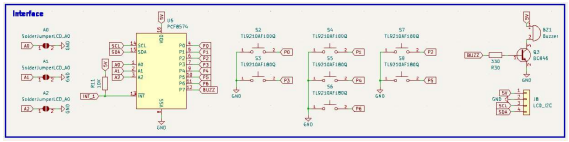
dan AMS117 3.3V untuk memenuhi kebutuhan tegangan 3V ESP32.



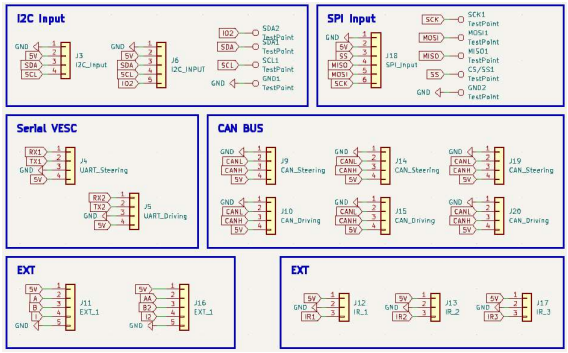
Gambar 4. Skematik MCU, *power supply master* dan *slave*



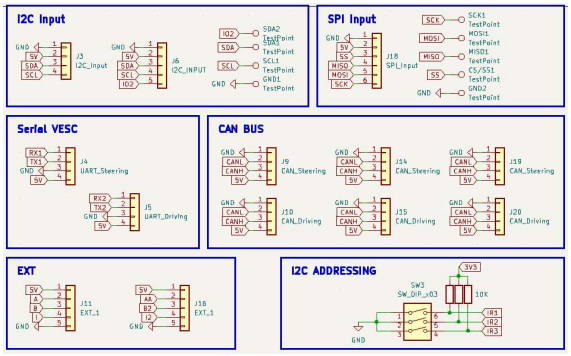
Gambar 5. Skematik USB-UART *master* dan *slave*



Gambar 6. Skematik *interface* dan tombol master



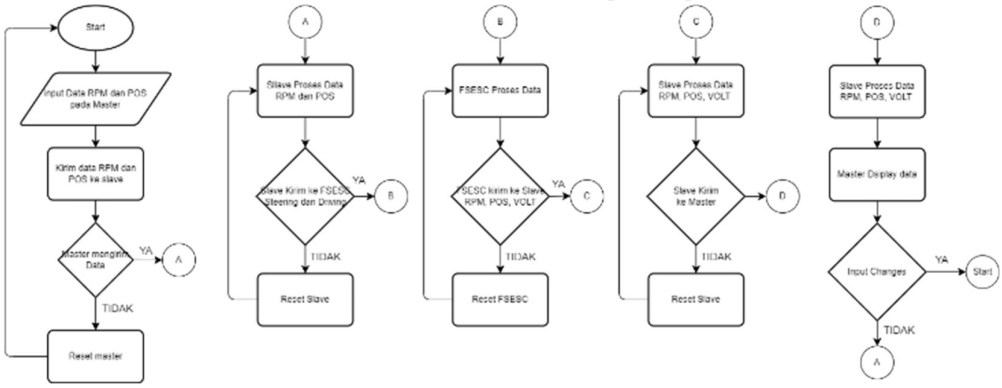
Gambar 7. Skematik pinout GPIO *master*



Gambar 8. Skematik *addressing* dan *pinout slave*

E. Perancangan Software

Agar *swerve drive* mudah dikendalikan, komunikasi I2C digunakan antara mikrokontroler *master* dan *slvae* modular papan kontroler. I2C dipilih karena kecepatan transfer data tinggi, konsumsi daya rendah, dan kompatibilitas luas. Mikrokontroler *master* mengirim dan menerima *Input* seperti arah dan kecepatan, sementara mikrokontroler *slave* mengontrol pergerakan roda *swerve drive* dan mengembalikan data dari *FSESC* ke *master*. Komunikasi I2C menghubungkan perangkat secara paralel untuk mengatur dan memproses perintah *swerve drive*.



Gambar 9. *Flowchart* perancangan *software*

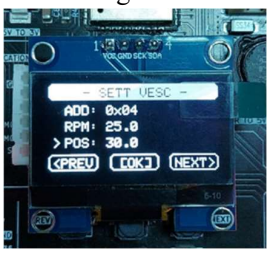
Berikut adalah *Flowchart* perancangan *software* untuk sistem kontroler *swerve drive* dengan komunikasi I2C:

1. Mulai untuk inisialisasi sistem.
2. *Input* data RPM dan POS pada mikrokontroler *master*. Pengguna dapat memasukkan nilai RPM dan POS pada mikrokontroler *master* sesuai yang diinginkan.
3. Kemudian *master* mengirimkan data RPM dan POS ke mikrokontroler *slave* melalui komunikasi I2C.
   a. Jika Ya, proses data RPM dan POS pada *slave*
   b. Jika Tidak, reset *master* atau ulangi dari *start*
4. *Slave* kirim data RPM dan POS ke masing – masing *FSESC*?
   a. Jika Ya, proses data RPM dan POS pada *FSESC*.
   b. Jika Tidak, reset *slave* dan ulangi ke proses data RPM dan POS pada *slave*
5. *FSESC* mengirim data RPM, POS, dan Volt pada *slave*?
   a. Jika iya, *slave* proses data RPM, POS dan Volt dari FSESC
   b. Jika tidak *FSESC* akan terus mengirim data pada *slave*
6. *Slave* mengirim data RPM, POS, dan Volt ke *master*
   a. Jika Ya, *master* akan proses data RPM, POS, dan Volt ke *FSESC*
   b. Jika Tidak, *slave* akan terus mengirim data terakhir yang diterima dari *FSESC*
7. *Master display* data RPM, POS, dan Volt pada OLED
8. *Input* RPM dan POS berubah?
   a. Jika Ya, kembali *Input* data RPM dan POS pada *master* dari awal kondisi *start*
   b. Jika Tidak, *slave* akan memproses data RPM dan POS terakhir yang diterima dari *master*.

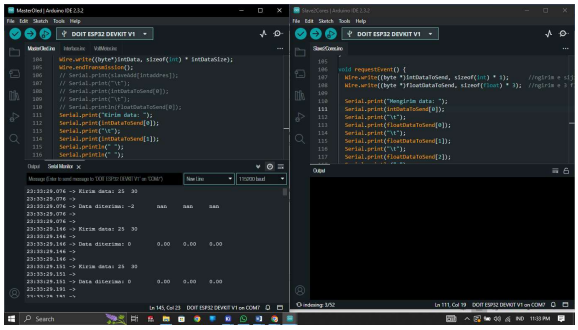
III. HASIL DAN PEMBAHASAN

A. Pengujian Rangkaian Hardware Modular (Hardware Master dan Hardware Slave)

Pengujain hardware *master* dilakukan untuk mengetahui apakah mikrokontroler *master* dapat digunakan dengan optimal atau tidak. Prosedur pengujian dilakukan dengan mencoba mengupload program I2C untuk mengirim data.

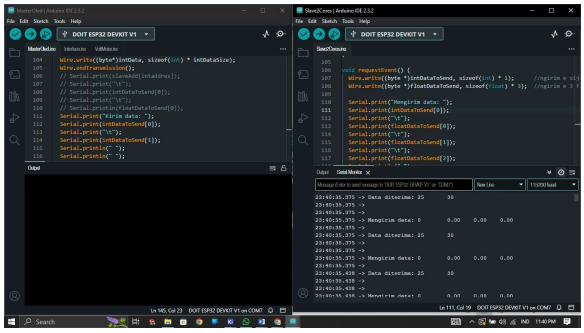


Gambar 10. Tampilan nilai pengiriman data pada *display* OLED master



Gambar 11. Tampilan pengiriman data pada serial monitor master

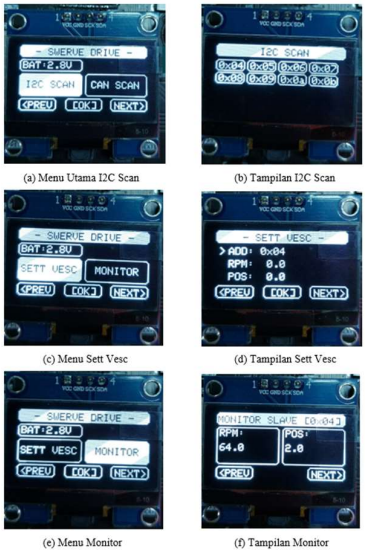
Pengujain hardware *slave* dilakukan untuk mengetahui apakah mikrokontroler *master* dapat digunakan dengan optimal atau tidak. Prosedur pengujian dilakukan dengan mencoba mengupload program I2C untuk menerima data dari *master*.



Gambar 12. Tampilan penerimaan data pada serial monitor *slave*

B. Pengujian Tampilan OLED

Pengujian tampilan *OLED* dilakukan untuk memastikan kualitas, keandalan, dan kinerja optimal dari teknologi layar ini sebelum digunakan.



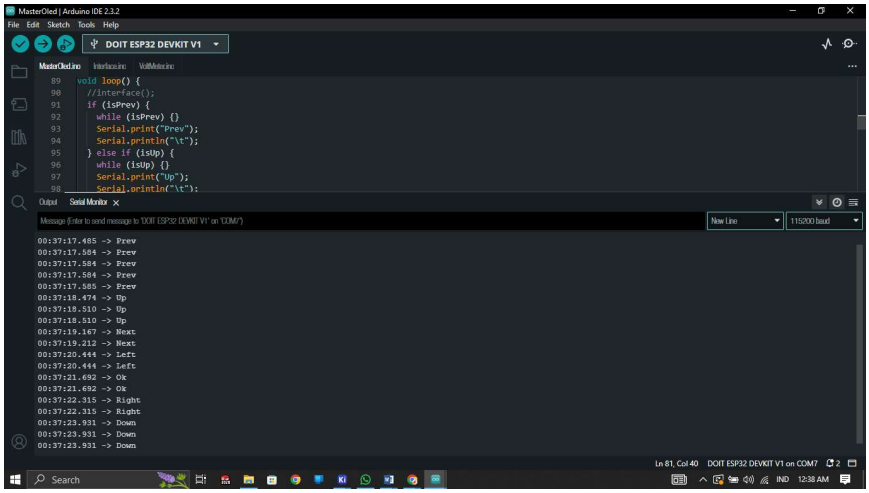
(a) Menu Utama I2C Scan (b) Tampilan I2C Scan
(c) Menu Sett Vesc (d) Tampilan Sett Vesc
(e) Menu Monitor (f) Tampilan Monitor

Gambar 13. Hasil pengujian tampilan OLED

Menu "I2C SCAN" digunakan untuk menampilkan alamat dan jumlah *device slave* yang terhubung pada *master*. Kemudian untuk "SPI SCAN" tidak digunakan. Menu "SETT VESC" digunakan untuk mengirim data nilai RPM dan POS ke *slave* yang terhubung. Terdapat juga pemilihan alamat *slave* yang ingin dikirimkan datanya. Kemudian menu "MONITOR" terdapat 2 bagian yaitu RPM dan POS. RPM merupakan hasil *read* ERPM pada VESC dan POS merupakan hasil *read position*

C. Pengujian Push button dan IC PCF8574

Pengujian *push button* dilakukan agar mengetahui apakah dapat bekerja sesuai dengan fungsinya yaitu apakah dapat ditekan dengan baik.



Gambar 14. Hasil serial monitor saat *push button* ditekan pada *master*

Ketika *push button* tidak ditekan maka akan berlogika 1 atau "HIGH" hal ini terjadi karena *push button* diberi *pull up* internal pada IC PCF8574 melalui program sehingga *push button* merupakan aktif *low*. Sedangkan ketika ditekan akan berlogika 0 atau "LOW" hal ini terjadi pin IC PCF8574 terhubung dengan *ground*. Tampilan pada hasil serial monitor adalah ketika *push button* ditekan atau berlogika LOW akan menampilkan tulisann sesuai dengan tombol yang ditekan.

D. Pengujian Komunikasi I2C

Pengujian komunikasi dilakukan dengan cara mengirim data dari *master* ke *slave*. Mikrokontroler *master* akan mengirim data dan *slave* akan menerima data dari *master* kemudian *slave* akan mengirimkan ke masing masing *FSESC*. Setelah itu *slave* juga akan menerima data dari masing masing *FSESC* yang nanti akan dikembalikan ke *master* untuk ditampilkan pada *master*. Untuk pengujian dilakukan pada serial monitor.

Pengujian komunikasi antar kontroler dilakukan untuk mengetahui tingkat akurasi data yang dikirim. Pengujian ini melibatkan pengiriman data dari *master* ke *slave*. Dalam komunikasi data ini, target *error* data adalah 0%, yang berarti semua data yang dikirim dan diterima harus identik.

Table 1. Hasil pengujian komunikasi I2C antara *master* dan *slave*

Data yang dikirim (master)	Data yang diterima (*Slave*)	Waktu transfer	*Error* (%)
Rpm: 705 Pos: 90	Rpm: 705 Pos: 90	1 ms	**0**
Rpm: 600 Pos: 90	Rpm: 600 Pos: 90	1 ms	**0**
Rpm: 650 Pos: 45	Rpm: 650 Pos: 45	1 ms	**0**
Rpm: 400 Pos: 30	Rpm: 400 Pos: 30	1 ms	**0**
Rpm: 1000 Pos: 0	Rpm: 1000 Pos: 0	1 ms	**0**
		Rata rata *Error*	**0**

Untuk menghitung kecepatan transfer (*data rate*) dihitung dengan membagi ukuran data dengan waktu yang dibutuhkan untuk mentransfer data. Pada percobaan digunakan 2 data dengan tipe data *integer* yang masing – masing data memiliki ukuran 4 *bytes* sehingga total data yang dikirim adalah 8 *bytes* dengan waktu 1 ms. Rumus dari kecepatan transfer adalah pada persamaan di bawah ini.

$$Kec.\ Transfer = \frac{Ukuran\ Data\ (bytes)}{Waktu\ Transfer\ (seconds)} \quad (10)$$

$$Kec.\ Transfer = \frac{8\ bytes}{0.001\ seconds} \quad (11)$$

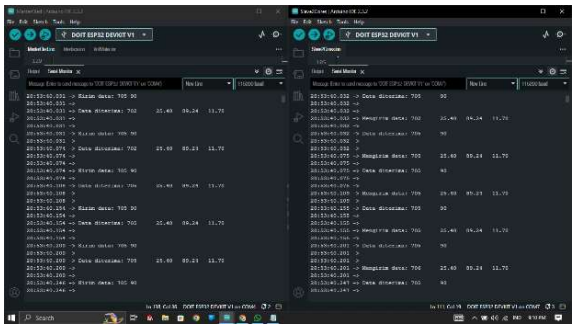
$$Kec.\ Transf = 8000\ bytes\ per\ seconds \quad (12)$$

Dengan begitu dapat disimpulkan bahwa kecepatan untuk mengirim data dari *master* ke *slave* dan pengiriman kembali data dari *slave* ke master adalah dengan persamaan di bawah ini.

$$Kec.\ Total = waktu\ transfer1 + waktu\ transfer2 \quad (13)$$

$$Kec.\ Total = 0.001\ seconds + 0.001\ seconds \quad (14)$$

$$Kec.\ Total = 0.002\ seconds \quad (15)$$



Gambar 15. Waktu transfer data

E. Pengujian Kontrol Posisi Steering

Pada pengujian ini dilakukan dengan cara memberikan nilai posisi dalam satuan derajat mulai dari 0 sampai 180. Pengujian dilakukan dengan melihat *error* dimana jika *error* masih besar dan presisi masih jauh dari kata sempurna, nanti di setting kembali PID pada *VESC tool*. Untuk menghitung nilai *error* dari skala penuh menggunakan rumus (16.

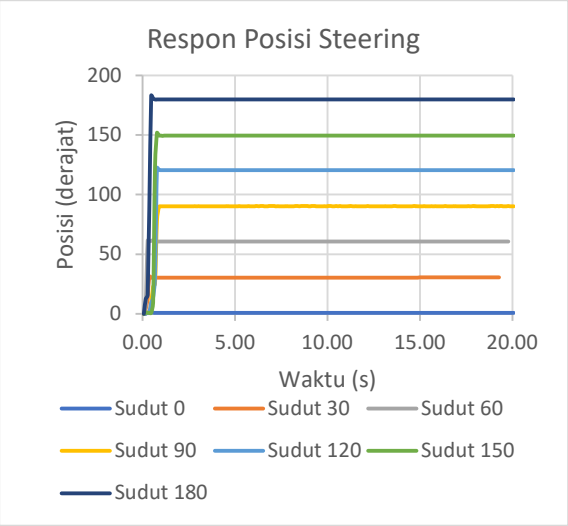
$$error = \frac{Set\ Point - Data\ aktual}{360\ derajat} \times 100\% \quad (16)$$

Pengukuran posisi aktual pada motor *steering* dilakukan dengan cara mengukur menggunakan penggaris busur.

Tabel 2. Hasil pengujian kontrol posisi *steering*

***Set point* Posisi (derajat)**	**Nilai Posisi VESC**	**Data Aktual (derajat)**	***Error* (%)**
0	0	0.5	**0.14**
30	30	30.5	**0.14**
60	60	60	**0.00**
90	90	90	**0.00**
120	120	119.5	**0.14**
150	150	151	**0.28**
180	180	179.5	**0.14**
		Rata rata Error	**0.12**

Pada pengujian ini bertujuan untuk melihat respon posisi *steering* menggunakan kontrol PID yang ada pada *VESC tool*. **Gambar** 6 merupakan hasil respon posisi *steering*. Sumbu X menunjukkan waktu dalam detik (s), dari 0 hingga 20 detik. Ini berarti pengukuran dilakukan selama periode 20 detik. Sumbu Y menunjukkan posisi dalam derajat, dari 0 hingga 200 derajat. Kontrol PID digunakan untuk mencapai dan mempertahankan sudut *set point* dengan cepat dan akurat. Pengaturan parameter PID yang dilakukan adalah men-*setting* nilai Kp=0.03, Ki=0, dan Kd=0.0007.



Gambar 16. Respon posisi *steering*

Gambar 6 menunjukkan bahwa grafik sistem kontrol PID dengan Kp=0.03, Ki=0, dan Kd=0.0007 memberikan respon yang cepat dan stabil untuk berbagai sudut *set point*. Pada awalnya, sistem menunjukkan kenaikan tajam menuju sudut *set point* berkat komponen *proportional* (Kp). Setelah itu, sistem mendatar pada nilai *set point* yang diinginkan, menunjukkan bahwa posisi dapat dipertahankan dengan baik. Komponen *derivative* (Kd) membantu mengurangi osilasi dan mempercepat stabilisasi, dengan sedikit atau tanpa *overshoot* dan *settling time* yang singkat. Meskipun komponen integral (Ki) diatur ke 0, *error steady-state* sangat kecil, menandakan bahwa kombinasi Kp dan Kd efektif. Kesimpulannya, kontrol PID yang digunakan mampu mengendalikan posisi *steering* dengan cepat, stabil, dan akurat.

F. Pengujian Kontrol Kecepatan Driving

Pada pengujian ini dilakukan dengan cara memberikan nilai RPM mulai dari 500 sampai 1000. Pengujian dilakukan dengan melihat *error* yang dimana jika *error* masih tinggi dan jauh dari kata sempurna, di *setting* kembali PID pada *VESC tool*. Untuk menghitung nilai *error* menggunakan rumus (17.

$$error = \frac{Set\ Point - Data\ aktual}{Max\ RPM} \times 100\% \qquad (17)$$

Maksimal RPM motor *driving* yang digunakan adalah 1440. Nilai maksimal RPM dihitung dari dengan rumus (18.

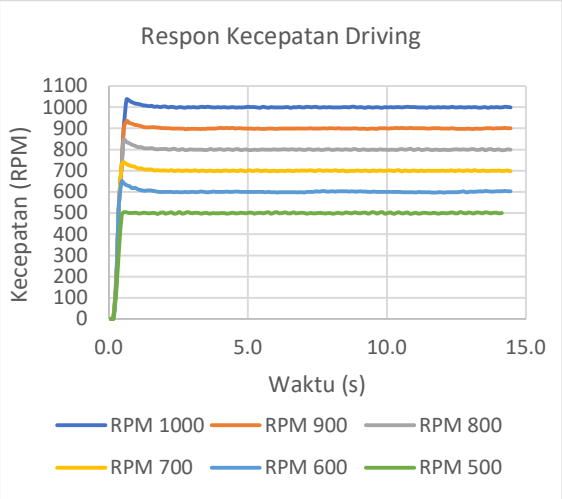
$$Max\ RPM = KV\ \times Tegangan\ (V) \qquad (18)$$

KV adalah konstanta kecepatan motor yang menyatakan berapa RPM yang dihasilkan per Volt. Motor BLDC *driving* yang digunakan memiliki spesifikasi 60 KV. Tegangan (V) merupakan tegangan yang digunakan pada motor tersebut yaitu 24V. Jadi $Max\ RPM = 60 \times 24$, yaitu 1440 Pengukuran RPM pada motor *driving* dilakukan dengan cara mengukur menggunakan alat ukut *tachometer*. Pada motor BLDC nilai kecepatan berupa *ERPM* yang merupakan *electrical RPM* dimana nilai *RPM* akan dikalikan dengan jumlah *pole pair* pada motor BLDC. Pada motor BLDC ini mempunyai spesifikasi 14 *pole* atau 7 *pole pairs* sehingga untuk menghitung *ERPM* adalah *RPM* x 14.

Tabel 3. Hasil pengujian kontrol kecepatan *driving*

Set point **Kecepatan (RPM)**	**ERPM**	**Data Aktual (RPM)**	***Error*** **(%)**
500	3500	499	**0.069**
600	4200	602	**0.138**
700	4900	701	**0.069**
800	5600	801	**0.069**
900	6300	901	**0.069**
1000	7000	1002	**0.138**
		Rata rata Error	**0.092**

Pada pengujian ini bertujuan untuk melihat respon kecepatan *driving* menggunakan kontrol PID yang ada pada *VESC tool*. **Gambar** merupakan hasil respon kecepatan *driving*. Sumbu X menunjukkan waktu dalam detik (s), dari 0 hingga 14 detik. Ini berarti pengukuran dilakukan selama periode 14 detik. Sumbu Y menunjukkan kecepatan dalam satuan RPM, dari 500 hingga 1000 *RPM*. Kontrol PID digunakan untuk mencapai dan mempertahankan kecepatan *RPM* dengan cepat dan akurat. Pengaturan parameter PID yang dilakukan adalah men-*setting* nilai Kp=0.022, Ki=0.03, dan Kd=0.00012.



Gambar 17. Respon kecepatan *driving*

Dalam pengujian kontrol PID untuk kecepatan motor BLDC, komponen *proportional* (Kp) memberikan respon cepat terhadap *error* awal, mencapai *set point* dalam waktu kurang dari 0.2 detik. Komponen integral (Ki) mengeliminasi *error steady-state* dengan akumulasi *error* dari waktu ke waktu, dan komponen *derivative* (Kd) mengurangi osilasi serta mempercepat stabilisasi. Grafik menunjukkan sedikit *overshoot* dan waktu stabilisasi singkat, dengan *error steady-state* yang kecil. Parameter PID Kp=0.022, Ki=0.03, dan Kd=0.00012 terbukti efektif, menghasilkan kontrol kecepatan motor yang cepat, stabil, dan akurat.

IV. KESIMPULAN DAN SARAN

Berdasarkan perancangan dapat diambil kesimpuan dari komunikasi antara mikrokontroler *master* dan *slave* menggunakan protokol I2C berjalan dengan baik, memungkinkan pengiriman dan penerimaan data *RPM* serta posisi *steering* secara *real-time*.

Pengujian menunjukkan bahwa sistem ini mampu mengurangi *error steady-state* hingga tingkat minimum dan mencapai stabilitas dalam waktu singkat. Implementasi ini membuktikan bahwa desain modular dapat diadaptasi untuk berbagai aplikasi kontrol robotik yang memerlukan komunikasi data secara *real-time.*

Saran dari penelitian yang telah penulis lakukan adalah meningkatkan antarmuka pengguna pada mikrokontroler master, seperti menggunakan layar sentuh atau antarmuka grafis yang lebih intuitif, akan memudahkan operator dalam mengontrol dan memantau sistem.

UCAPAN TERIMAKASIH

Penulis mengucapkan banyak terima kasih kepada dosen pembimbing Bapak Indrazno Siradjuddin, ST., MT., PhD. dan Bapak Ir. Totok Winarno, M.T. yang sudah bersedia meluangkan waktu untuk mengarahkan serta memberi saran dalam proses pengerjaan jurnal dan artikel ini.

REFERENSI

[1] I. Siradjuddin, G. Al Azhar, A. Murdani, and M. L. M. Faizin, “Desain dan pemodelan kontrol kinematik pergerakan robot beroda dengan menggunakan 6 roda omni-wheels,” *JURNAL ELTEK*, vol. 18, no. 1, p. 116, Apr. 2020, doi: 10.33795/eltek.v18i1.226.

[2] V. K. Pranav, K. S. Kumar, A. R. Nair, and G. Udupa, “Design and Manufacture of Octagonal Rover for Space Exploration,” in *2020 4th International Conference on Automation, Control and Robots, ICACR 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 48–52. doi: 10.1109/ICACR51161.2020.9265497.

[3] R. Dhelika, A. F. Hadi, and P. A. Yusuf, “Development of a motorized hospital bed with *swerve drive* modules for holonomic mobility,” *Applied Sciences (Switzerland)*, vol. 11, no. 23, pp. 1–9, Dec. 2021, doi: 10.3390/app112311356.

[4] R. Zamronnan, T. Winarno, and E. Mandayatma, “Kontrol Posisi Sistem Pergerakan *Mobile* Robot Berbasis Analisa Kinematik,” *Jurnal Elektronika dan Otomasi Industri*, vol. 07, no. 1, pp. 77–84, May 2020, Accessed: Dec. 15, 2023. [Online]. Available: https://jurnal.polinema.ac.id/index.php/elkolind/article/view/4321

[5] M. A. Al Mamun, M. T. Nasir, and A. Khayyat, “Embedded system for motion control of an omnidirectional *mobile* robot,” *IEEE Access*, vol. 14, no. 8, pp. 6722–6739, Jan. 2018, doi: 10.1109/ACCESS.2018.2794441.

[6] J. Carothers, "Design of a Triple Singularity Drive for *Mobile* Wheeled Robots," Cambridge, Jul. 2014.

[7] G. Al Azhar, T. Winarno, S. Izza, P. N. Malang, P. Unisma, and P. Korespondensi, "Sistem Distribusi Data Kontrol Pada Differential Drive *Mobile* Robot Menggunakan Robot Operating System," *Journal of Mechanical and Electrical Technology*, vol. 1, no. 3, 2022.

[8] G. Al Azhar, S. Sungkono, M. N. Achmadiyah, and S. Izza, "Peningkatan Kestabilan Sistem Kontrol UGV melalui Optimalisasi Manajemen Core dan Free-RTOS pada ESP32," *Jurnal Elektronika dan Otomasi Industri*, vol. 10, no. 2, pp. 253–263, Jul. 2023, doi: 10.33795/elkolind.v10i2.3720.

[9] B. Bagus, I. Adinugraha, I. Siradjuddin, L. Kamajaya, P. N. Malang, and P. Korespondensi, "Desain dan Kontrol Modular Independent Drive Independent *Steering Mobile* Robot Aktuator," *Journal of Mechanical and Electrical Technology*, vol. 2, no. 3, pp. 144–150, Oct. 2023, Accessed: Dec. 15, 2023. [Online]. Available: https://ejournal.uniramalang.ac.id/index.php/metrotech/article/view/3349

[10] M. Nurul Achmadiah, A. anwar Rosyidin, A. Pracoyo, I. Siradjuddin, D. A. Permatasari, and G. Al Azhar, "Desain permodelan dan simulasi Field Oriented Control (FOC) menggunakan motor BLDC : Aplikasi pada *Drivetrain* - SwerveDrive," *Jurnal Elektronika dan Otomasi Industri*, vol. 10, no. 3, pp. 361–368, Sep. 2023, doi: 10.33795/elkolind.v10i3.4416.