METROTECH
Journal of Mechanical and Electrical Technology
Vol. 3 No. 3, September 2024, E-ISSN 2809-9605 | P-ISSN 2827-8097

Implementasi Modular Papan Kontroler *Swerve Drive:* Komunikasi SPI

Naufal Rizki Prayogi[1*], Indrazno Siradjuddin[2], Donny Radianto[3]
[1,2,3] Politeknik Negeri Malang, Malang
[*]Penulis Korespondensi, email: naufalrizkip7@gmail.com

Received:07/08/2024 *Revised:25/08/2024* *Accepted:01/09/2024*

Abstract. *The development of drive technology in mobile robots, especially in wheeled robots is growing so fast. Swerve drive is a drive technology that is designed to be omnidirectional, has the advantage of separate steering and driving (independent) on each wheel. This model allows the robot to move in any direction with high speed and maneuverability. The disadvantages of the swerve drive are weight and cost, as well as a more complex control algorithm in regulating the speed of the steering and driving motors to be controlled simultaneously in moving the robot to the target position and orientation. The ESP32 was chosen as the microcontroller because it is equipped with a dual-core processor and can operate up to 240 MHz, providing sufficient performance to control complex systems such as the swerve drive. The ESP32 also supports the fast and efficient SPI (Serial Peripheral Interface) communication protocol. The solution to control the speed of the steering and driving motors simultaneously in moving the robot to the target position and orientation PID control is very effective with a low error rate, specifically 0.006% in driving control and 0.019%. PID control can help maintain system stability by minimizing steady-state error, overshoot, and settling time. PID control is important to maintain consistent performance of the robot.*

Keywords: *Swerve drive, Mobile robot, SPI, PID*

Abstrak. Perkembangan teknologi penggerak pada *mobile robot*, khususnya pada robot beroda berkembang begitu cepat. *Swerve drive* merupakan teknologi penggerak yang dirancang bersifat *omnidirectional,* memiliki keunggulan yaitu *steering* dan *driving* yang terpisah (*independent*) pada setiap roda. Model ini memungkinkan robot bergerak ke segala arah dengan kecepatan dan kemampuan manuver yang tinggi. Kelemahan dari *swerve drive* yaitu pada bobot dan biaya, serta algoritma kontrol yang lebih komplek dalam mengatur kecepatan motor *steering* dan *driving* untuk selanjutnya di kontrol secara bersamaan dalam menggerakkan robot ke posisi dan orientasi target. ESP32 dipilih sebagai mikrokontroler karena dilengkapi dengan prosesor ganda (*dual-core*) dan dapat beroperasi hingga 240 MHz, memberikan performa yang cukup untuk mengendalikan sistem yang kompleks seperti *swerve drive*. ESP32 juga mendukung protokol komunikasi SPI (*Serial Peripheral Interface*) yang cepat dan efisien. Solusi untuk mengontrol kecepatan motor *steering* dan *driving* secara bersamaan dalam menggerakkan robot ke posisi dan orientasi target kontrol PID sangat efektif dengan tingkat kesalahan yang rendah, khususnya 0.006% pada kontrol *driving* dan 0.019%. Kontrol PID dapat membantu menjaga stabilitas sistem dengan meminimalkan kesalahan *steady-state*, *overshoot*, dan *settling time*. Kontrol PID penting untuk menjaga kinerja yang konsisten dari robot.

Kata Kunci: *Swerve drive*, *Mobile robot*, SPI, PID

I. PENDAHULUAN

Robotika telah menjadi bidang penelitian yang semakin berkembang. Perannya yang dapat membantu dalam meningkatkan kinerja dan mobilitas manusia menjadi alasan utama mengapa penelitian ini masih banyak diminati. Pernyataan ini didukung oleh [1] yang berpendapat bahwa robotika dan otomatisasi merupakan peran yang krusial dalam mengoptimalkan operasi, meningkatkan produktivitas, dan kualitas yang dimiliki oleh suatu perusahaan . Tak hanya itu, penggunaan robot dalam kehidupan sehari-hari juga dapat membantu efektivitas pekerjaan terutama pada segi mobilitas. Dalam konteks ini, penggunaan s*werve drive* pada robot beroda menjadi salah satu pendekatan yang menjanjikan untuk memberikan tingkat mobilitas dan kemampuan manuver yang tinggi.

Swerve drive secara khusus dirancang sebagai *omnidirectional* yakni kemampuan untuk bergerak ke segala arah. Tak hanya itu, *swerve drive* dianggap lebih efisien dari sistem *omnidirectional* lainnya. Hal ini karena prosesnya yang tidak

kehilangan daya yang signifikan pada *drive train* dikarenakan roda yang bekerja berlawanan satu sama lain [2]. Pada dasarnya s*werve drive* adalah modul penggerak dengan konsep sederhana yang dikendalikan secara mandiri. Modul ini terdiri dari dua motor, satu *gearbox*, dua *encoder*, dan roda [3]. Salah satu motor pada modul ini digunakan untuk menggerakkan roda sedangkan motor kedua berfungsi sebagai kontrol kemudi. Modul selanjutnya yaitu *gearbox* berperan untuk mengontrol rotasi roda [4].

Peningkatan presisi gerakan dan daya tanggap robot beroda dapat dilakukan dengan menggunakan ESP32 sebagai mikrokontroler, FSESC 4.12 sebagai kontroler motor BLDC, serta integrasi sensor-sensor seperti *rotary encoder* [5]. BLDC sendiri digunakan karena memiliki keunggulan pada umur operasinya yang panjang, tanpa suara, rentang kecepatannya lebar, suhu rendah dan dapat menahan guncangan serta getaran [6]. Tingginya kecepatan yang dihasilkan oleh BLDC tentunya harus diimbangi dengan kestabilan agar dapat menentukan arah yang dituju dengan akurat. Oleh karena itu, kontroler PID digunakan untuk mengatur kestabilan kecepatan motor BLDC.

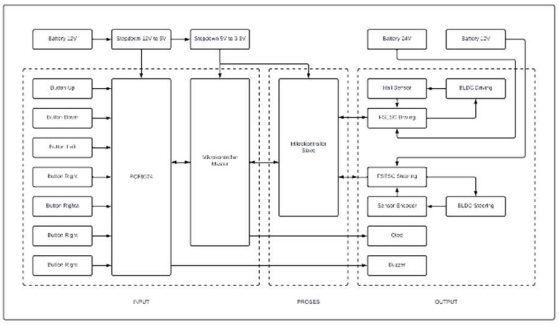
Sistem pengontrol PID memiliki beberapa jenis tindakan kontrol antara lain tindakan kontrol proporsional, tindakan kontrol integral, dan tindakan kontrol turunan [7]. Setiap tindakan memiliki manfaat yang berbeda. Kontrol proposional memiliki keuntungan dalam meningkatkan respons transien. Sedangkan kontrol integral memiliki keuntungan dalam menghasilkan respons sistem yang memiliki kesalahan kondisi tunak nol [8]. Kontroler PID sendiri merupakan gabungan dari hasil penjumlahan *output* kontroler P, *output* kontroler I, dan *output* kontroler D. Karakteristiknya dipengaruhi oleh besaran dari ketiga parameter tersebut [9]. Tentunya pemilihan kontroler PID ini didasari oleh manfaatnya dalam mempercepat proses *steady set* dari kecepatan BLDC dan untuk mengontrol kecepatannya secara otomatis [10].

Penelitian ini dinisiasi oleh kebutuhan akan solusi yang dapat memberikan tingkat mobilitas yang tinggi pada robot mobile, terutama dalam situasi yang memerlukan navigasi presisi dan perubahan arah yang cepat . Dengan demikian, pengembangan modul papan kontroler *swerve drive* dengan komunikasi SPI dan implementasi kontrol PID menjadi langkah kritis menuju peningkatan kemampuan robot *mobile* dalam berbagai aplikasi praktis. Melalui penelitian ini, diharapkan dapat memberikan kontribusi pada pemahaman tentang penggunaan *swerve drive* pada robot *mobile* salah satunya robot beroda dan pengembangan teknologi kendali robotika secara keseluruhan.

II. METODOLOGI

Pendekatan yang diterapkan dalam penelitian ini dapat dibagi menjadi serangkaian tahapan, mencakup diagram blok sistem, perancangan mekanik, perancangan elektronik, perancangan kontrol PID, perancangan kinerja.

A. Diagram Blok Sistem



Gambar 1. Diagram blok sistem

Diagram blok dibuat agar mengetahui alur dari sistem kerja suatu alat yang dirancang. Alur kerja dimulai dari *Input* sampai keseluruhan sesuai dengan spesifikasi sistem kerja alat yang direncanakan. Gambar 1 merupakan diagram blok sistem yang mencakup perangkat *Input,* Proses dan *Output:*

1. Input

Pada bagian input terdapat modul master ESP32 yang menggunakan protocol SPI dengan konfigurasi pin MOSI, MISO, SCK, SS, GND. Konfigurasi ini digunakan untuk komunikasi antara modul mikrokontroler master dan modul mikrokontroler *slave* untuk memberi perintah pada FSESC.

Terdapat 7 *Push button* yang berfungsi sebagai navigator di *interface* yaitu PB UP, PB DOWN, PB LEFT, PB RIGHT, PB PREV, PB NEXT, dan PB OK. *Push button* dirancang menggunakan aktif *(Low),* dimana salah satu pin dihubungkan ke ground dan satunya lagi ke IC PCF8574. Modul *slave* ini menggunakan *supply* tegangan sebesar 3V.

2. Process

Pada bagian proses yaitu pada modul *slave* berfungsi untuk menerima data atau perintah dari modul master yang dikirim melalui komunikasi SPI, kemudian dilakukan komputasi data untuk mengontrol posisi motor *steering* dan kecepatan pada motor *driving*. Kemudian, juga berfungsi sebagai penerima data pembacaan RPM, POS dan Volt pada masing-masing FSESC. Modul *slave* ini menggunakan *supply* tegangan sebesar 3V.

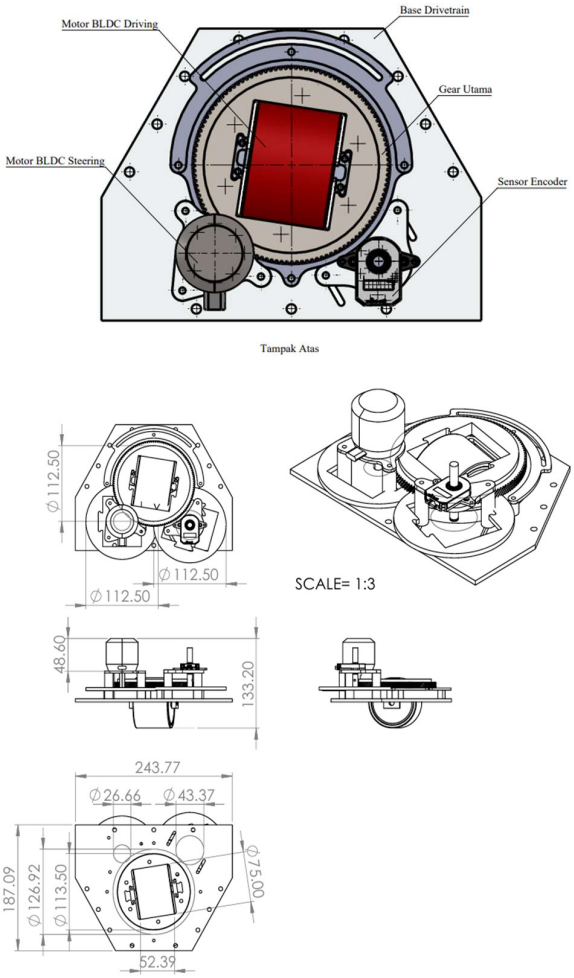
3. Output

Pada bagian *output* terdapat FSESC *(Driving)* dan FSESC *(Steering)* yang berguna sebagai *driver* dari motor BLDC. FSESC berkomunikasi dengan ESP32 melalui UART. Terdapat juga motor BLDC tipe 70mm *(Driving)* dengan *supply* daya sebesar 24V dan BLDC tipe 5045 *(Steering)* dengan *supply* daya sebesar 12V. Motor BLDC digunakan sebagai aktuator atau sebagai penggerak. *Encoder* berfungsi sebagai sensor posisi sudut *steering*. Pengukuran dari sensor *encoder* akan menjadi *feedback* pada FSESC yang akan dikirim ke modul *slave* dan modul *slave* akan mengirim ke modul master untuk ditampilkan pada *interface*. *Oled display* digunakan sebagai *interface* yang isinya adalah informasi tentang kecepatan, sudut motor.

B. Perancangan Mekanik

Perancangan mekanik merupakan salah satu hal yang penting dalam pembuatan suatu alat karena berpengaruh kepada keberhasilan alat. Pada Gambar 2 terdapat 2 motor yaitu *steering* sebagai penggerak rotasi dan *driving* sebagai penggerak translasi. Terdapat beberapa komponen pada perancangan mekanik yaitu motor BLDC, *rotary encoder*, dan *gear*. Berikut adalah spesifikasi mekanik *drivetrain:*

1. Panjang: 19cm
2. Lebar: 21cm
3. Tinggi: 13,32cm
4. Jenis Motor: Motor BLDC 5045 (*Steering*), Motor BLDC 70mm (*Driving*)
5. Bahan Base: Plat Alumunium
6. Bahan *Gear*: PETG



Gambar 2. Desain mekanik *drivetrain*

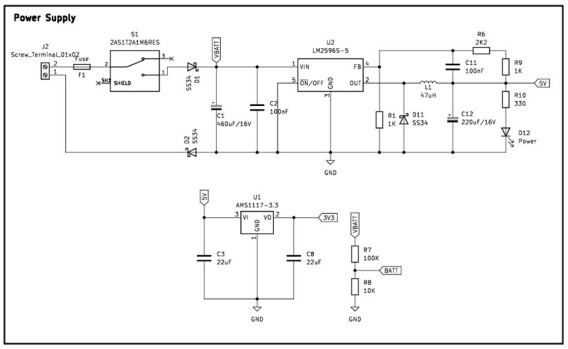
Pembuatan *gear steering* menggunakan metode spiral bevel *gear*, metode spiral bevel *gear* memiliki kelebihan kemampuan transmisi daya dan efisiensi yang lebih besar pada geometri yang sama serta tidak terlalu berisik. Pada *steering* memiliki ukuran rasio dengan perbandingan 1:4 atau 120 grigi pada *gear* utama dan 30 grigi pada *gear* yang terpasang pada motor *steering*. Jadi 4 putaran penuh pada *gear* motor sama dengan 1 putaran pada *gear* module. Dengan rasio gear yang lebih tinggi, torsi yang dihasilkan oleh gear penggerak akan lebih besar karena gear tersebut berputar lebih lambat namun dengan kekuatan yang lebih besar sekitar 3.4 $N.m$.

Pada *encoder* memiliki ukuran rasio dengan perbandingan 1:3 dengan rincian 120 grigi pada *gear* utama dan 40 grigi pada shaft *encoder* sehingga 3 putaran pada *gear encoder* sama dengan 1 putaran pada *gear* modular *swerve drive.* Jika menggunakan rasio *gear* di bawah 1:3 ini sangat tidak memungkinkan karena ukuran gear akan semakin besar dan hal ini akan memakan banyak tempat di modul *swerve drive* dan jika menggunakan rasio gear di atas 1:3 maka akan terjadi data *losing* pada saat ESC membaca *encoder.*

C. Perancangan Elektronik

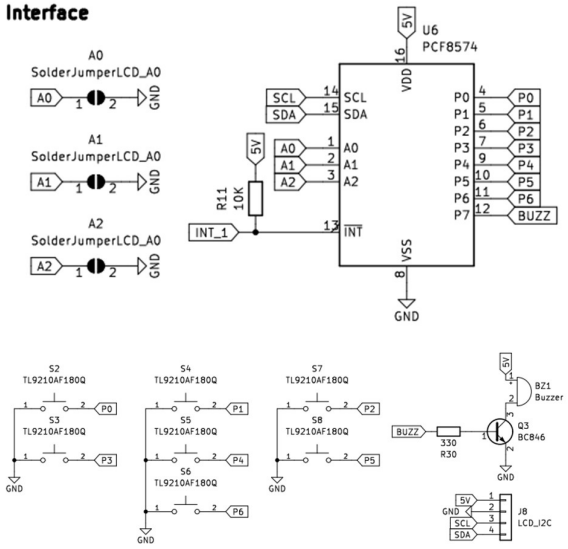
Perancangan elektrik sangat penting untuk penunjang agar alat bekerja dengan baik. Berikut adalah spesfikasi elektronik pada modul *main board swerve drive:*

1. Catu Daya: *Battery Lipo* 11.1V 3S 3300mAh
2. *Step Down*: LM2596
3. Jenis Mikrokontroller: ESP32
4. *Interface*: OLED 1.3 inch



Gambar 3. Perancangan *power management*

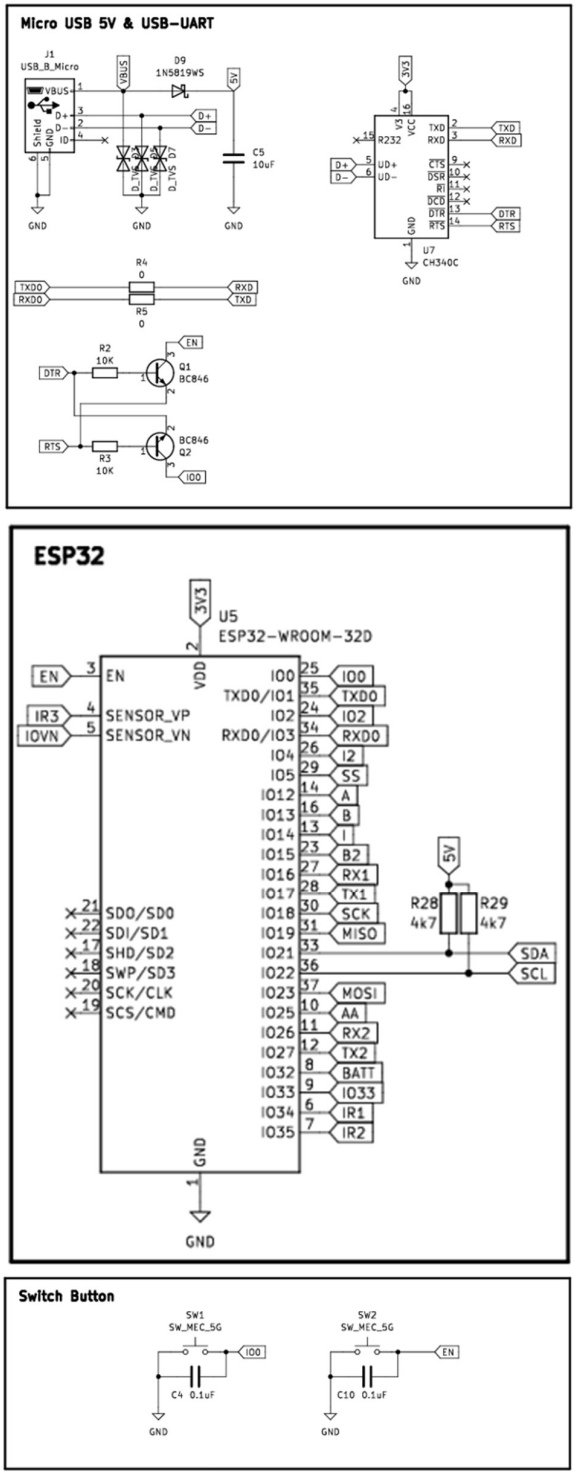
Pada *Power Management* terdapat konektor baterai yang terhubung ke fuse 3A kemudian melalui sakelar. *Fuse* 3A digunakan sebagai pengaman keseluruhan rangkaian jika terjadi arus berlebih. Untuk penurun tegangan dari baterai 12V menggunakan *stepdown* 5V 3A yang digunakan sebagai sumber tegangan dari komponen lain. Karena saya juga menggunakan ESP32-WROOM *chip* untuk mikrokontrolernya dan tegangan *input* yang dibutuhkan yaitu 3.3V jadi fungsi dari AMS1117-3.3 yaitu untuk menurunkan tegangan dari 5V ke 3.3V. Pada rangkaian IC LM2596 *adjustable* digunakan resistor pembagi tegangan pada pin *feedback* IC LM2596 agar yang dihasilkan tegangan 5V 3A.



Gambar 4. Perancangan *interface*

IC PCF8574 pada **Error! Reference source not found.** adalah I/O *expander* 8-bit yang berkomunikasi melalui antarmuka I2C yang berfungsi untuk menambah jumlah pin I/O pada mikrokontroler yang memiliki jumlah pin terbatas. Pada **Error! Reference source not found.** merupakan konfigurasi pin PCF8574, pin A0-A2 digunakan untuk mengatur alamat I2C yaitu dengan mengubah konfigurasi jumper atau solder. Pin P0-P7 berfungsi sebagai *input* dari *push button* yang terhubung ke *Ground* dimana bila tombol ditekan akan menurunkan tegangan mendekati 0V dan mikrokontroler akan membaca logika (*Low*) pada *push button.*

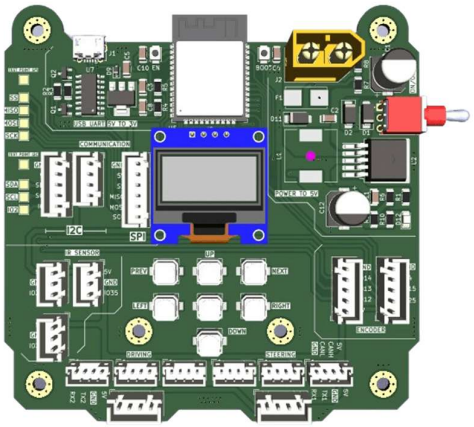
OLED berfungsi untuk menampilkan informasi visual berupa data yang dihasilkan. *Buzzer* berguna untuk indikator suara dari status atau sinyal. Dalam sistem mikrokontroler, *buzzer* dapat diaktifkan atau mematikan melalui perintah yang diberikan.



Gambar 5. Perancangan kontrol ESP32 dengan USB-UART

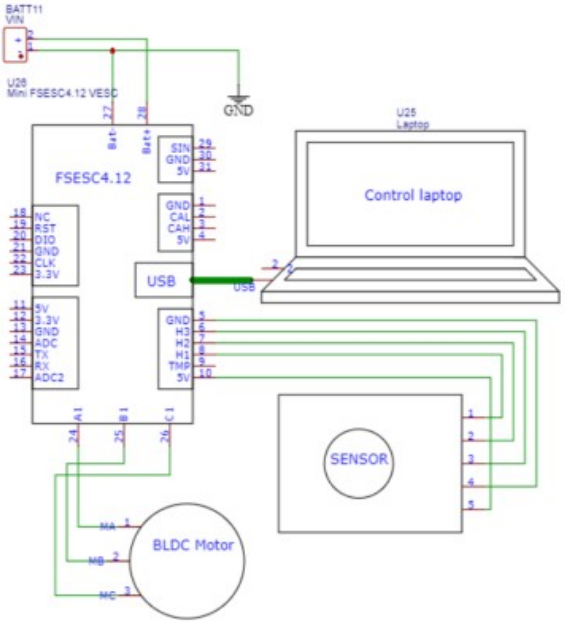
Pada skematik Micro USB5V & USB-UART gambar 5 adalah rangkaian yang berfungsi untuk mengkonversi sinyal USB dari komputer menjadi sinyal UART yang dapat digunakan oleh mikrokontroler, serta menyediakan suplai daya 5V ke rangkaian.

IC CH340C mengkonversi sinyal data USB yang diterima dari komputer melalui D+ dan D- menjadi sinyal serial UART yang bisa diterima oleh ESP32.



Gambar 6. Desain PCB

D. Perancangan PID

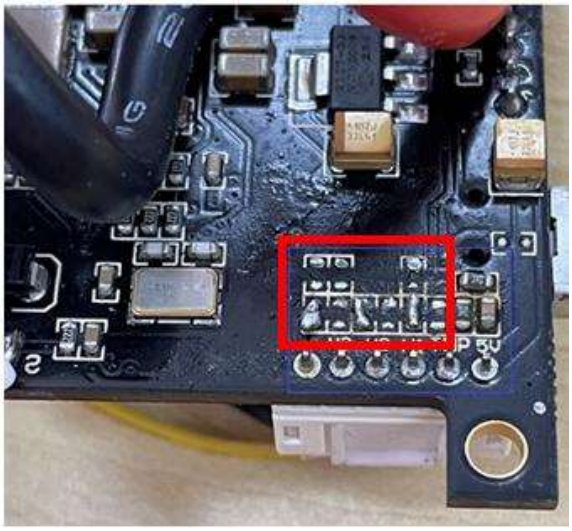


Gambar 7. Rangkaian kontrol PID

Untuk melakukan pengaturan PID motor BLDC baik pada *steering* ataupun *driving* pada perlu merangkai seperti pada gambar 7 dan perlu dilakukannya konfigurasi secara *hardware* dan *software*. Untuk langkah-langkah *setting* PID dapat dilakukan sebagai berikut:

1. Konfigurasi *Hardware*

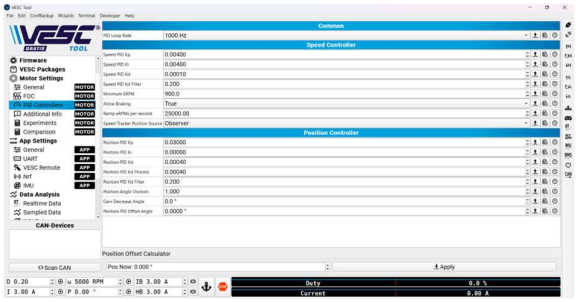
Pada konfigurasi *hardware* diperlukan agar FSESC dapat menggunakan sensor *encoder* selain *Hall* sensor dengan cara menghilangkan komponen *low pass filter* dan menghilangkan komponen *pull up* pada rangkaian *encoder* seperti pada gambar 8.



Gambar 8. Menghilangkan *low pass filter*

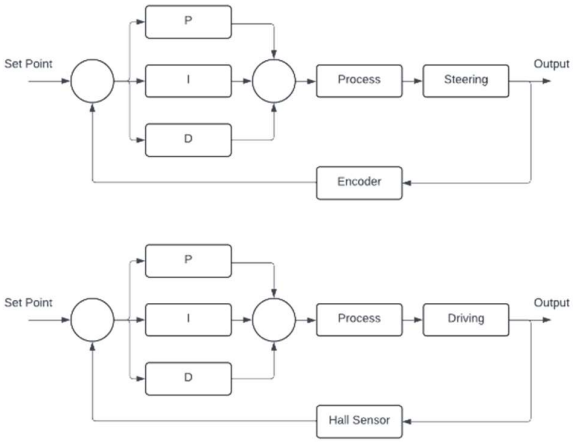
2. Konfigurasi *Software*

VESC *tool* adalah *software open source* yang digunakan untuk konfigurasi FSESC dan motor BLDC sebelum digunakan.



Gambar 9. Tampilan menu *setting* PID

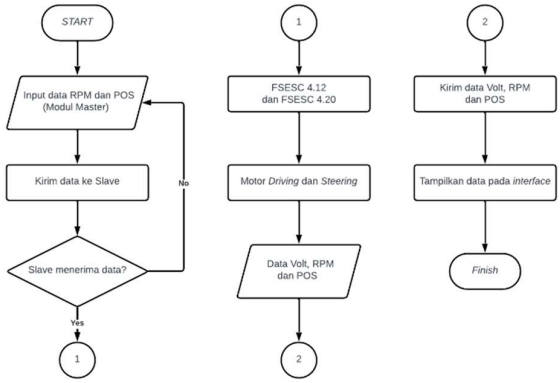
Dimana pada Perancangan PID kontrol motor BLDC dilakukan pada 2 tahap yaitu PID untuk motor s*teering* (Motor BLDC 6834) dan PID motor d*riving* (BLDC hub 70mm). terdapat beberapa parameter yang digunakan agar sistem berjalan optimal yaitu *Proportional* (P), *Integral* (I) dan *Derivatif* (D).



Gambar 10. Diagram blok kontrol PID

Diagram blok kontrol PID pada gambar 10 bertujuan untuk mengembangkan sistem kontrol yang dapat menyesuaikan variabel dengan cepat, stabil dan akurat dengan mengintegrasikan tiga jenis kontrol yaitu Proporsional (P), Integral (I) dan turunan (D). Proporsional digunakan untuk menanggapi kesalahan saat ini, Integral digunakan untuk menyeimbangkan kesalahan dari waktu ke waktu, dan Differensial digunakan untuk menyesuaikan tingkat perubahan kesalahan.

E. Perancangan Kinerja



Gambar 11. Perancangan kinerja

Berikut penjelasan perancangan kinerja:

1. Pertama, Input data nilai RPM dan POS pada modul *master.*
2. Kemudian, modul master akan mengirim data ke modul *slave.*

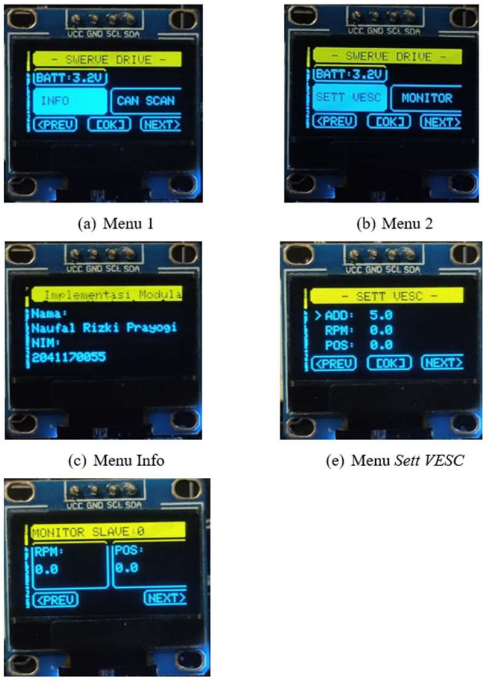
3. Apakah modul *slave* menerima data yang dikirim oleh modul *master?*
4. Jika Iya, modul *slave* akan melanjutkannya untuk dikirim ke FSESC.
5. Jika tidak, coba mengatur kembali pengiriman dari modul *master* ke *slave.*
6. Data yang dikirimkan ke FSESC oleh modul *slave* akan dilanjutkan untuk mengatur motor *steering* (POS) dan *driving* (RPM).
7. Data *realtime* dari pengaturan motor akan dikirim ke modul *slave* yaitu Volt, RPM dan POS.

Yang kemudian oleh modul *slave* akan dikirimkan ke modul *master* untuk di tampilkan pada *interface*.

III. HASIL DAN PEMBAHASAN

A. Pengujian OLED

Pengujian dilakukan untuk memastikan apakah data informasi yang dikirim dan diterima dari modul *slave* yang dikirim dari VESC benar dan sesuai. Terdapat empat pilihan menu yaitu I2C *Scan*, SPI Scan, Sett VESC dan Monitor. Untuk I2C *Scan* saya tidak menggunakannya, pada SPI *Scan* tidak berfungsi dikarenakan pada SPI menggunakan *Pin* bukan *Address*. Pada menu *Sett* VESC didalamnya terdapat yang pertama yaitu pemilihan pin *slave* yang akan dikirimkan *setting VESC,* kemudian *setting* kecepatan motor *driving* (RPM), dan yang ketiga yaitu *setting* posisi motor *steering.* Setelah selesai mensetting klik *next* akan ke menu tampilan RPM dan posisi motor.



(a) Menu 1 (b) Menu 2

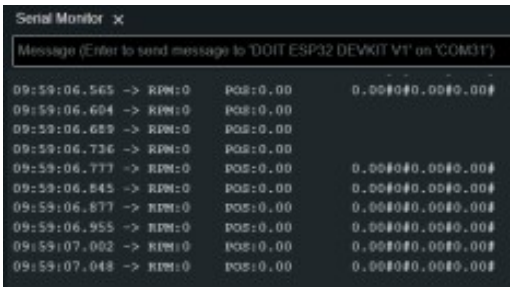
(c) Menu Info (e) Menu *Sett VESC*

(d) Menu Monitor

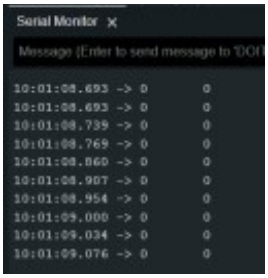
Gambar 12. Pengujian OLED 0,96

B. Pengujian komunikasi SPI

Pada pengujian komunikasi dengan cara mengirim data dari master ke *slave*.



Gambar 13. Modul master



Gambar 14. Modul *slave*

C. Pengujian Kontrol Kecepatan Driving

Pengujian dilakukan dengan cara memberikan nilai RPM mulai dari 100-1000. Pengujian

dilakukan dengan melihat *error* yang dimana jika *error* masih tinggi dan jauh dari kata sempurna, di *setting* kembali PID pada VESC *tool*. Untuk melihat nilai *error* dilakukan dengan cara melihat secara *real-time* melalui *software* VESC *tool*.

Tabel 1. Hasil pengujian kecepatan motor *driving*

Setpoint Kecepatan (RPM)	Nilai Kecepatan (RPM)	Error (%)
100	102	2
150	150.3	0.2
200	200.4	0.2
250	250.4	0.16
300	300.6	0.2
350	350.5	0,142
400	400.7	0.17
450	450.7	0.15
500	501	0.2
550	551	0.18
600	601.2	0.2
650	651.4	0.21
700	701.1	0.15
750	751.3	0.17
800	801.2	0.15
850	851.2	0.14
900	901.6	0.17
950	951.6	0.16
1000	1002	0.2
Rata-rata error		0.27

D. Pengujian Kontrol Kecepatan Steering

Pengujian dilakukan dengan cara memberikan nilai posisi dalam satuan derajat mulai dari 0 sampai 360. Pengujian dilakukan dengan melihat *error* yang dimana jika *error* masih besar dan presisi masih jauh dari kata sempurna, nanti di *setting* kembali PID pada VESC *tool*. Untuk menghitung nilai *error* adalah

$$Error = \frac{setpoint - dataaktu}{setpoint} \times 100\%. \quad (1)$$

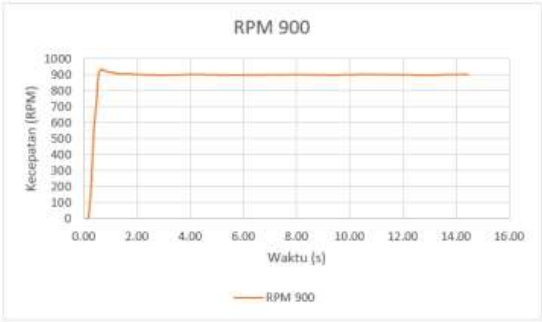
Pengukuran posisi *actual* pada motor *steering* dilakukan dengan cara mengukur menggunakan penggaris busur.

Tabel 2. Hasil pengujian motor s*teering*

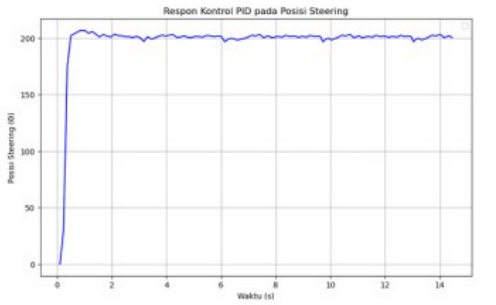
SetPoint Posisi	Nilai Posisi Vesc	Nilai Posisi Busur	Error %
0	0	0	0
30	30	30	0
60	60	60	0
90	90	90	0
120	120	120	0
150	150	150	0
180	180	180	0
210	210	210	0
240	240	241	0.277778
270	270	271	0.277778
300	300	300	0
330	330	329	0.277778
360	360	360	0
		Rata-rata Error	0.064103

E. Respon Kecepatan Driving dan Streering

Hasil dari percobaan dan pengujian didapatkan bahwa kecepatan motor *driving* akan konstan dengan set RPM pada detik ke-2. Terdapat osilasi pada respon kecepatan *driving* yang disebabkan oleh lonjakan arus pada BLDC dan osilasi pada kondisi steady state sangatlah kecil dan menimbulkan grafik yang lebih *smooth.*



Gambar 15. Respon kecepatan motor *driving*



Gambar 16. Respon kecepatan motor *steering*

IV. KESIMPULAN DAN SARAN

Dari hasil pengujain pada penelitian ini, dapat ditarik kesimpulan bahwa:

1. Pembuatan sistem pada skripsi ini dimulai dengan merancang sistem mekanik, elektronik dan *software*. Sistem elektronik meliputi beberapa komponen berupa *stepdown* LM2596, OLED, PCF8574, Buzzer, ESP32. Perancangan mekanik meliputi *drivetrain swerve drive* dan modul *main boar* digunakan, serta perancangan *software* meliputi rancangan komunikasi dan VESC.
2. Dengan pengaturan frekuensi *clock* pada komunikasi SPI di 1MHz (1.000.000Hz) yang dimana frekuensi ini menentukan kecepatan komunikasi, setiap perubahan *clock cycle* mengirimkan satu bit data. Jadi dengan

pengaturan frekuensi di 1MHz = 1.000.000 *clock cycles* per *second* = 1.000.000 bits per *second* (bps).

3. Penambahan kontrol PID (Ki, Kp, dan Kd) memiliki pengaruh yang signifikan terhadap respon kecepatan dan stabilitas pada kontrol motor. Hasil pengujian menunjukkan bahwa kontrol PID memberikan performa yang lebih baik dengan rata-rata nillai *error* saat menggunakan kontrol PID pada kecepatan *driving* dengan pengaturan Kp= 0.022, Ki= 0.03, dan Kd= 0.00012 sebesar 0.006% dan posisi *steering* dengan pengaturan PID Kp= 0.03, Ki= 0, dan Kd= 0.0007 sebesar 0.019%.

Dari penelitian ini, perlu adanya perbaikan dan penyempurnaan kembali pada modul *swerve drive*. Berikut adalah saran setelah melakukan pengujian:

1. Perhitungan rasio pada *gear* sangatlah penting agar nilai posisi akurat sesuai dengan yang diinginkan.
2. Pengecekan kembali kabel komunikasi antara master dan *slave*, karena kabel tersebut sangat berpengaruh dalam pengiriman data.
3. Berhati-hatilah saat awal menggunakan *steering*. Pastikan setelah menggunakan motor *steering* dikembalikan ke posisi 0 kembali.

Perbaiki tuning PID pada kontrol PID agar lebih optimal.

UCAPAN TERIMAKASIH

Penulis mengucapkan banyak terima kasih kepada dosen pembimbing Bapak Indrazno Siradjuddin, ST., MT., PhD. dan Bapak Donny Radianto, S.T., M.Eng. yang sudah bersedia meluangkan waktu untuk mengarahkan serta memberi saran dalam proses pengerjaan skripsi ini.

REFERENSI

[1] M. Huda and E. S. Pudjiarti, "Peran Otomatisasi dan Robotika dalam Era Digital: Trasformasi Bisnis Melalui Otomatisasi dan Robotika dalam Era Digital," *Transform. J. Econ. ...*, vol. 3, no. 1, 2024.

[2] J. Carothers, "Design of a Triple Singularity Drive for Mobile Wheeled Robots," pp. 1–43, 2014.

[3] 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," *J. Elektron. dan Otomasi Ind.*, vol. 10, no. 3, pp. 361–368, 2023.

[4] DeNoma, Benjamin, Kendall, Michael, Poulos, and Nick, "4-wheel Independent Steering 'Swerve Drive,'" pp. 1–73, 2022.

[5] L. B. Setyawan, "Prinsip Kerja dan Teknologi OLED," *Techné J. Ilm. Elektrotek.*, vol. 16, no. 02, pp. 121–132, 2017.

[6] A. A. Obed and A. K. Kadhim, "Speed and Current Limiting Control Strategies for BLDC Motor Drive System: A Comparative Study," *Int. J. Adv. Eng. Res. Sci.*, vol. 5, no. 2, pp. 119–130, 2018.

[7] V. Carev, J. Roháč, S. Tkachenko, and K. Alloyarov, "The Electronic Switch of Windings of a Standard BLDC Motor," *Appl. Sci.*, vol. 12, no. 21, 2022.

[8] R. Kristiyono and Wiyono, "Autotuning fuzzy PID controller for speed control of BLDC motor," *J. Robot. Control*, vol. 2, no. 5, pp. 400–407, 2021.

[9] J. Nilda, G. F. Saunoah, D. P. K. Iradiratu, and B. Y. Dewantara, "Perbaikanfaktor Dayamenggunakancuk Converterberbasis Pid Padamotor Brushless Dc," pp. 168–176, 2020.

[10] B. P. Putri, S. Sutedjo, O. A. Qudsi, and L. S. Mahendra, "Alat Penstabil Kecepatan Motor BLDC Menggunakan Kontrol PID," *Emit. J. Tek. Elektro*, vol. 22, no. 2, pp. 134–140, 2022.