Quad-Drive Speed Control berbasis PID pada *Barelang Crab_Bot 2023*

Tugas Akhir

Oleh:
Muhammad Akbar (4212001036)

Program Studi Teknik Mekatronika
Jurusan Teknik Elektro
Politeknik Negeri Batam
2024

Pernyataan Keaslian Tugas Akhir

Saya yang bertandatangan dibawah ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya yang berjudul : "*Quad-Drive Speed Control* berbasis PID pada *Barelang Crab_Bot 2023*" adalah **hasil karya sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan**, dan **bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.** Semua referensi yang dikutip atau dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan saya ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Batam, 14 Mei 2024

Muhammad Akbar
NIM: 4212001036

Lembar Pengesahan

Tugas Akhir disusun untuk memenuhi salah satu syarat memperoleh gelar
Sarjana Terapan Teknik (S.Tr.T)
di
Politeknik Negeri Batam

Oleh:
Muhammad Akbar (4212001036)

Tanggal Sidang: 7 Mei, 2024

Disetujui oleh :

1. Indra Hardian Mulyadi, S.T., M.Eng., Pd.D.
   NIK: 117179

2. Eka Mutia Lubis, S.Pd., M.Pd.
   NIK: 117186

1. Dr. Abdurahman Dwijotomo, S.ST., M.Sc.
   NIK: 122257

Quad-Drive Speed Control berbasis PID pada *Barelang Crab_Bot 2023*

Abstrak

Kontes Robot Tematik Indonesia merupakan ajang perlombaan nasional yang dilaksanakan sejak tahun 2019. Salah satu tugas pada perlombaan harus dapat memindahkan barang pada titik-titik tertentu di lapangan. Implementasi *quad-drive mecanum wheels* pada *Barelang Crab_Bot* menjadi platform penggerak yang menggabungkan empat sumber daya untuk mencapai pergerakan presisi dan mobilitas yang unggul. Sistem penggerak robot ini umumnya menghasilkan gerak yang tidak lurus dan tidak stabil dengan arah hadapnya tidak konsisten melalui pengendali terbuka, dikarenakan karakter tiap motor memiliki kecepatan yang tidak sama dengan karaktristik yang tidak identik. Penelitian ini bertujuan untuk meningkatkan sistem *Quad-drive* dengan penggunaan algoritma pengendalian *Proportional, Integral, Derivative* (PID). Sistem kendali PID ini bekerja dengan cara memproses perhitungan berdasarkan variable kendali Kp, Ki, dan Kd untuk mencapai kondisi sesuai setpoint yang diharapkan. Pada praktiknya variabel yang akan di kontrol pada penelitian adalah kendali kecepatan berupa RPM dari pembacaan sensor *encoder* sebagai *feedback*. Dengan alat pengendali Arduino Mega, driver motor sebagai pengontrol kecepatan dan putaran motor DC, *software* yang digunakan adalah Arduino IDE. Yang mengasilkan hadap pergerakan robot dengan kemungkinan *error* sebesar 4^{o} dari hasi pengujain pergerakan maju. dengan parameter PID yang ditentukan melalui *trial and error* metode manual *tunning* didapat nilai Kp=1.1, Ki=3.5, Kd=0.01.

Kata kunci: motor DC, PID *(proposional, integral, derivative*), *Respon system*, *mecannum wheels.*

PID-based Quad-Drive Speed Control for Barelang Crab_Bot 2023

Abstract

The Indonesian Thematic Robot Contest is a national competition held since 2019. One of the tasks in the race must be able to move goods at certain points on the field. The implementation of quad-drive mecanum wheels on Barelang Crab_Bot becomes a drive platform that combines four power sources to achieve precision movement and superior mobility. This robot drive system generally produces motion that is not straight and unstable with inconsistent facing direction through open controllers, because the character of each motor has an unequal speed with non-identical characteristics. This research aims to improve the Quad-drive system with the use of Proportional, Integral, Derivative (PID) control algorithms. This PID control system works by processing calculations based on the control variables Kp, Ki, and Kd to achieve conditions according to the expected setpoint. In practice, the variable that will be controlled in the research is speed control in the form of RPM from the encoder sensor reading as feedback. With Arduino Mega controller, motor driver as speed controller and DC motor rotation, the software used is Arduino IDE. Which produces a robot movement face with a possible error of 4^o from the forward movement test results, with PID parameters determined through trial and error, the manual tunning method obtained the value of Kp = 1.1, Ki = 3.5, Kd = 0.01.

Keywords: DC motor, PID (proportional, integral, derivative), Response system, mecannum wheels

Kata Pengantar

Segala puji dan syukur penulis panjatkan kepada Allah SWT sang Maha Segalanya, atas seluruh curahan rahmat dan hidayatnya sehingga penulis mampu menyelesaikan skripsi yang berjudul *" Quad-Drive Speed Control* berbasis PID pada *Barelang Crab_Bot 2023*" ini tepat pada waktunya. Skripsi ini ditulis dalam rangka memenuhi syarat untuk mencapai gelar Sarjana pada Program Studi Teknik Mekatronika Politeknik Negeri Batam.

Dalam penyelesaian studi dan penulisan skripsi ini, penulis banyak memperoleh bantuan baik pengajaran, bimbingan dan arahan dari berbagai pihak baik secara langsung maupun tidak langsung. Untuk itu penulis menyampaikan penghargaan dan terima kasih yang tak terhingga kepada:

1. Kedua orang tua dan keluarga yang telah memberikan dukungan secara doa, moral, motivasi serta nasihat yang sangan membantu bagi penulis.
2. Bapak Uuf Brajawidagda, S.T., M.T., Ph.D. selaku Direktur Politeknik Negeri Batam.
3. Bapak Budi Sugandi, S.T., M.Eng selaku kepala jurusan teknik elektro.
4. Bapak Indra Hardian Mulyadi, S.T., M.Eng., Ph.D selaku kepala prodi teknik mekatronika.
5. Bapak Dr. Abdurahman Dwijotomo, S.ST., M.Sc. Selaku dosen pembimbing yang telah menyediakan waktu, tenaga dan pikiran untuk memberikan arahan kepada penulis dalam penyusunan dan menyelesaikan Buku Tugas Akhir ini.
6. Bapak Indra Hardian Mulyadi, S.T., M.Eng., Ph.D, dan ibu Eka Mutia Lubis, S.Pd., M.Pd. selaku dosen penguji.
7. Bapak Senanjung Prayoga, S.Pd., M.T selaku kepala prodi teknik robotika yang telah memberikan akses pembuatan alat.
8. Teman-Teman dari prodi robotika yang sangat membantu memberikan semangat, saran terkait pembuatan alat.
9. Seluruh teman-teman kelas Teknik Mekatronika angkatan 2020 yang memberikan dukungan bagi penulis untuk menyelesaikan tugas akhir.

Batam, 1 April 2024

Muhammad Akbar

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1. Pendahuluan

1.1. Latar Belakang

Perkembangan dunia teknologi sangat pesat, teknologi robotika dan sistem otomasi banyak merubah aspek industri dan kehidupan manusia saat ini. Robot ialah kumpulan rangkaian-rangkaian elektronika yang bekerja secara otomatis sesuai dengan perintah yang diberikan oleh pembuatnya, dalam berkembangan jaman robot semakin dikenal di hampir semua kalangan, salah satunya sang sangat populer yaitu KRI (Kontes Robot Indonesia), KRI sendiri adalah kontes nasional, yang dibagi atas 7 divisi yang dilombakan, salah satunya yaitu Kontes Robot Tematik Indonesia (KRTMI) yang diadakan oleh Direktorat Kementrian Pendidikan, Kebudayaan, Riset, dan Teknologi.

Kontes Robot Tematik Indonesia (KRTMI) yang bertema "*Robo Game-Digital Twin*", tema yang merupakan komponen utama pada industri 4.0, yang mencakup otomasi, pertukaran data, dan proses manufaktur yang mana dapat menghasilkan peluang tak terbatas bagi industri untuk tumbuh. Pada rancangan kontes, kontes ini mengadu kecekatan dan keakuratan antara kedua tim dalam menggerakan robot, mengambil dan meletakan koin (roda gigi) pada posisi tertentu. *Barelang Crab_Bot* dibuat dengan aturan yang telah ditetapkan DIKTI, robot dengan 4 roda yang di kombinasikan dengan griper dibagian depan untuk megambil dan meletakan koin[1].

Namun, robot dan sistem otomasi bukanlah tanpa tantangan, salah satu aspek yang paling penting dalam robotika dan sistem otomasi adalah sistem kontrol, yang memiliki peran penting dalam kelancaran sistem. Sistem kontrol adalah sekumpulan komponen yang bekerja sama untuk mengontrol dan memantau kinerja robot dan sistem otomasi. Dari penelitian yang diterbitkan dalam *jurnal of Intelligent Manufacturing*, perkembangan sistem kontrol robotika dan otomasi merupakan aspek penting untuk memastikan keandalan dan efektivitas sistem[2].

Dari penelitiannya penulis menggunakan *mecanum wheel.* Roda ini didesain dengan disekelilingi roller yang membentuk sudut 45^{o}, dengan desain ini memberikan keuntungan kinematik tambahan untuk roda *mecannum* di bandingkan dengan roda konvensional yang mengarah pada kendala non-holonomis pada desain pergerakan *mobile* robot *differential drive*[3]. Keuntungan dan kelemahan sistem penggerak robot menggunakan mecanum wheel adalah salah satunya yaitu mampu bergerak cepat ke berbagai arah tanpa harus mengubah arah hadapnya robot. Sejauh ini banyak penelitian tentang kontrol pergerakkan mobile robot. Banyak tantangan yang dihadapi, diantaranya pada aktuatornya sendiri yaitu Motor DC.

Motor DC memiliki karakteristik dari variable – variabel yang berperan sebagai fungi waktu pada kondisi transien merupakan hal yang perlu diperhatikan selama

pengendalian. Suatu penyearah pasti memiliki karakteristik yang berbeda – beda dan karakteristik ini mempengaruhi tegangan masukan pada motor DC sehingga juga akan mempengaruhi bentuk gelombang arus yang dihasilkan dan hal ini juga mempengaruhi torsi dan kecepatan rotor yang membuat ketidak stabilan pergerakan pada robot[4].

Dalam pengendalian motor DC dibutuhkannya sensor untuk membaca kecepatan, agar tujuan yang diinginkan tercapai maka diperlukan controller untuk mengatur kenerja motor DC dengan menggunakan metode *Proportional, Integral, Derivative* atau PID. Metode PID merupakan metode yang bekerja dengan cara menganalisis kesalahan yang ada untuk dijadikan koreksi saat sistem bekerja[5].

Menurut (Kusuma, Panca Agung dan Dharmawan, Adhi, 2017), PID populer dikarenakan pemrosesannya yang ringan namun kemampuannya fleksibel untuk setiap sistem dengan melakukan pengaturan nilai gain untuk *Konstanta proportional (Kp), Konstanta integral (Ki),* dan *Konstanta derivative (Kd).* Nilai *gain Kp, Ki, Kd* merupakan nilai tetap yang dijadikan sebagai penentu kinerja sistem, nilai *gain* ini tidak bisa digunakan secara universal yang artinya setiap sistem tidak memiliki nilai *gain* yang sama meskipun komponen fisik yang digunakan identik. *Kp* berlaku sebagai *Gain* (penguat) tanpa memberikan efek dinamik kepada kinerja sistem, *Ki* dapat memperbaiki respon dari sistem, sedangkan *Kd* dapat mengurangi efek berlebihan yang dihasilkan oleh sistem[6].

Sistem PID menjadi pilihan yang tepat untuk memenuhi kondisi guna mendapatkan performa yang baik. PID adalah salah satu contoh metode untuk mengontrol kecepatan motor dc untuk menghindari slip pada roda. Pengendali PID memiliki respon yang cepat terhadap suatu sistem[7]. Huang G. dan Lee S. merancang pengontrol PID berbasis LabVIEW untuk kontrol kecepatan motor DC dan responnya menggunakan simulasi perangkat lunak VisSim untuk analisis. Pertama, dengan mendesain rangkaian driver motor menggunakan sensor foto dan modul 8051 untuk informasi umpan balik. Hasil simulasi yang diperoleh menyimpulkan bahwa hal itu sesuai dengan prediksi teoritis [8]. Teknik kontrol PID digunakan dalam berbagai aplikasi industri karena kesederhanaannya dalam pengaplikasiannya. Aplikasi awal adalah dalam sistem pneumatik, peralatan vakum dan elektronik analog solid-state. Kemudian, implementasi digital dari proses mikro mulai digunakan[9].

Berdasarkan hal tersebut, dari penelitian ini Metode PID diharapkan dapat meningkatkan akurasi pada motor DC, yang membuat pergerakan pada robot dapat bergerak dengan presisi sesuai dengan set point yang diberikan.

1.2. Rumusan Masalah

1. Bagaimana membuat sistem kendali yang digunakan untuk menentukan presisi pada pergerakan robot?
2. Apa pengaruh sistem kontrol PID pada robot?
3. Bagaimana menentukan parameter PID(*proportional, integral, derivative)* yang optimal pada motor DC?

1.3. Tujuan

1. Merancang sistem kendali kecepatan motor DC menggunakan metode PID.
2. Memberikan kecepatan kontrol yang efektif serta pergerakan dan perpindahan robot stabil dan presisi.
3. Melakukan *tunning Kp, Ki, Kd* secara manual dengan melihat ketentuan parameter secara spesifik dengan respon yang baik.

1.4. Manfaat

Bertujuan untuk mendapatkan kestabilan kecepatan pada tiap motor untuk mengasilkan pergerakan yang stabil dan presisi. Memberikan efektivitas pada pergerakan untuk memudahkan pengambilan dan peletakan koin yang ada di arena lapangan.

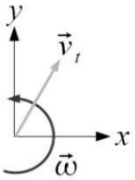
1.5. Batasan

1. Menggunakan metode PID sebagai kendali kecepatan motor DC pada robot *mecanum wheel*.
2. Menggunakan *software* Arduino IDE untuk membuat program dan menampilkan data.
3. Merakit komponen yang digunakan untuk mengontrol kecepatan motor dc.

Bab 2. Tinjauan Pustaka

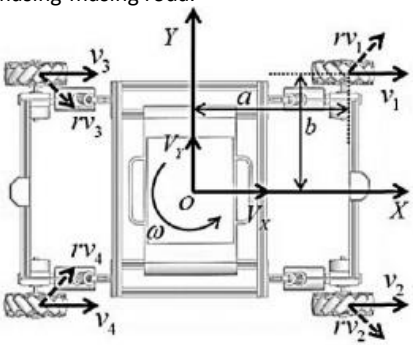
2.1. *Kinematic Mecanum Wheels*

Persamaan kinematik mendefinisikan pergerakan suatu roda untuk memudahkan mengontrol pergerakan yang mudah dilakukan. Diantaranya, V_{xy} V_y dan ω , Resultan diantarannya V_{xy} V_y adalah V_t. Persamaan pergerakan roda ditunjukan pada gambar 2.1.



Gambar 2.1. *Vektor* pergerakan robot[10]

Persamaan invers kinematik digunakan untuk menjelaskan pergerakan robot menjadi persamaan masing-masing roda.



Gambar 2.2. *Vektor* pergerakan *Mobile Robot* roda *mecannum*[10]

Dari gambar 2.2 menjelaskan bahwa a adalah jarak antara roda depan dan titik pusat robot. b adalah jarak roda samping dengan titik pusat. $v_{1,2,3,4,}$ merepresentasikan kecepatan roda, $rv_{1,2,3,4}$ merepresentasikan kecepatan sudut dalam titik pusat[10].

2.2. Kendali Kecepatan Motor DC

Pengendali fluksi medan magnet atau mengatur pada tahanan geser medan yang dihubungkan secara seri dengan medan *shnut*. Dengan menaikan tahanan dalam rangkaian medan menyebabkan fluksi medan yang mengecil (menurun)

sehingga kecepatan bertambah. Dengan tahanan seri dinaikan, tegangan pada jangkar berkurang degan kecepatan motor turun. Pengendalian tegangan terminal, dengan memberikan tegangan variable pada jangkar, akan menyebabkan kecepatan pada motor berubah-ubah[11].

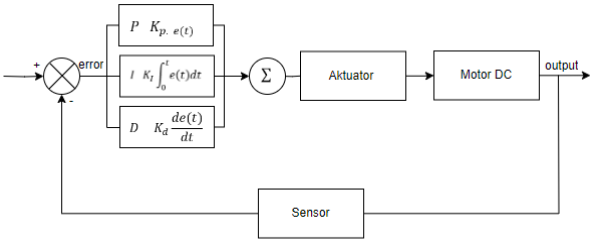
2.3. Kontrol PID

Sistem Kontrol PID (*Propotional Integral Derivative*) merupakan kontrol mekanisme yang digunakan untuk menentukan presisi suatu sistem instrumentasi dengan karakteristik adanya umpan balik (*feedback*). Parameter Kontrol PID terdiri dari atas kontrol P (*Proportional*), I (*Integral*), D (*Derivative*). Yang dimana nilai dari parameter tersebut bertujuan untuk mempercepat reaksi sistem, menghilangkan *offset* dan menghasilkan perubahan yang besar. Masukan sistem (set point), dengan nilai keluaran (RPM), sedangkan nilai *error* yang dilambangkan (e) yang dihasilkan sistem dengan dihitungnya (set point) – (RPM). Nilai *error* ini yang nantinya akan di proses oleh Kontrol PID akan menghasilkan output sesuai keinginan[11].

Keterangan :

a. *Proportional* (P) : Kontrol P sebagai penguat yang mampu mengubah output dari sistem secara proporsional tanpa memberikan efek dinamik.
b. *Integral* (I) : Kontrol I merupakan pengendali yang berfungsi untuk memperbaiki respon / stady state dari sistem sehingga pengendali ini mampu memperkecil *error system.*
c. *Derivarative* (D) : Kontrol D tergantung laju perubahan kesalahan. Pengendali ini merupakan suatu pengendali yang berfungsi untuk memperbaiki respon *transien* dari *system.*
d. *Error* : merupakan nilai jumlah yang tidak melakukan dengan benar.
e. *RPM:* berupa output dari motor DC.

PID blok diagram dapat dilihat pada gambar 2.1.



Gambar 2.3. Blok diagram sistem kendali PID

Adapun persamaan pengontrol PID:

$$cv = cv1 + \left(Kp + \frac{Kd}{Ts}\right) e(n) + \left(-Kp + KiTs - 2\frac{Kd}{Ts}\right) e(n-1) + \frac{Kd}{Ts} e(n-2)$$

Keterangan :
cv = Ouput pengontrol PID
Kp = Konstanta Proportional
Ki = Konstanta Integral
Kd = Konstanta Derivative
Ts = Time
e(n) = error (selisih dari set point dengan nilai level actual)

Berdasarkan dari pengontrol ini pengendali *proportional* bertujuan meningkan *offset*, hal ini disebabkan oleh sifat dasar pengendali proporsional yang masih tetap membentuk *error* untuk menghasilkan output. Oleh karena itu untuk menghilangkan *offset* diperlukan pengendali lain yang dapat menghasilkan output walaupun sudah tidak ada input, sifat unik inilah yang hanya dimiliki oleh pengendali *integral*. Tetapi kemampuan pengendali *integral* menghilangkan *offset* tidak disertai kemampuan bereaksi secara cepat. Untuk mempercepat kemampuan bereaksi maka diperlukan pengendali *derivative*, sehingga kekurangan yang ada pada pengendali *integral* dapat ditutupi. Keluaran kontroler PID merupakan jumlah dari keluaran kontroler *proportional*, *integral, derivative.* [11].

Efek dari setiap pengontrol *Proportional, Integral, Derivative* pada sistem tertutup dilihat pada table 2.1.

Table 2.1. Efek parameter PID

Parameter	*Rise time*	*Overshoot*	*Settling time*	*Error*
Kp	Bekurang	Bertambah	Minor Change	Berkurang
Ki	Bekurang	Bertambah	Bertambah	Menghilangkan
Kd	Minor change	Berkurang	Berkurang	Minor Change

2.4. Motor DC

Motor DC (*direct current*) adalah peralatan elektromekanik dasar yang berfungsi untuk mengubah tenaga listrik menjadi tenaga mekanik. Motor dc merupakan jenis motor yang menggunakan tegangan searah sebagai sumber tenaganya. Dengan memberikan beda tegangan pada kedua terminal tersebut, motor akan berputar pada satu arah, dan bila polaritas dari tegangan tersebut dibalik maka arah putaran motor akan terbalik pula. Polaritas dari tegangan yang diberikan pada dua terminal menentukan arah putaran motor sedangkan besar dari beda tegangan pada kedua terminal menentukan kecepatan motor[12].

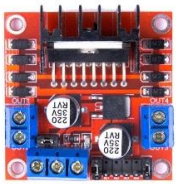
Gambar 2.4. Motor DC *encoder*
(Sumber : www.5v 600 rpm Encoder Motor DC)

Konstruksi motor dc secara umum :

1. Stator
   Stator merupakan bagian yang diam pada motor DC, di bagian stator terdapat magnet permanen yang menghasilkan medan magnet untuk memutar rotor yang berada diantara kedua kutub magnet, pada bagan stator ini diberi catu DC pada kumparan medan (field windings) sehingga muncul medan magnet konstan, oleh karena itu, stator pada mesin DC disebut sebagai penghasil medan magnet utama[13].
2. Rotor
   Rotor merupakan bagian yang bergerak pada motor DC, rotor terdiri dari inti besi yang dililitkan kumparan. Kumparan pada rotor ini disebut sebagai kumparan jangkar dimana GGL induksi dihasilkan. Rotor ini yang menggerakan roda pada robot sehingga perputarannya dapat membantu pergerakan pada lintasan, kecepatan putaran rotor ini tergantung dari tegangan yang masuk dan gaya medan magnet yang dihasilkan, semakin besar tegangan yang masuk, maka perputarannya semakin cepat[13].
3. Komutator
   Komutator adalah sebuah peralatan penyearah arus listrik yang bekerja secara mekanis. Fungsi dari komutator ialah menghasilkan tegangan listrik dalam suatu siklus putaran dengan jenis arus searah[13].

2.5. Driver motor L298N

Driver motor digunakan sebagai penghubung antara mikrokontroler ke motor DC. Driver motor ini digunakan karena arus yang keluar dari mikrokontroler tidak mampu mencukupi kebutuhan dari motor DC. IC L298N dapat digunakan untuk menggerakan motor DC *half-bridge* sebanyak empat buah atau dua motor DC *fullbridge*. IC ini mempunyai 4 pin input yang bersesuaian dengan 4 pin outputnya. Selain itu juga terdapat 2 pin enable[13].



Gambar 2.5. L298N
(Sumber : http://t0.gstatic.com)

2.5. Arduino Mega 2560

Arduino Mega 2560 merupakan board elektronik yang menggunakan mikrokontroller ATmega2560. Arduino ini memiliki 16 pin analog dan 54 pin I/O digital (15 pin diantaranya digunakan untuk output PWM), 16 pin input analog, 4 UART(Port Serial perangkat keras), Osilator kristal 16Mhz, koneksi USB, colokan listrik, header ICSP, dan tombol Reset. Cukup sambungkan ke computer dengan kabel USB atau nyalakan dengan adaptor AC ke DC atau batrai untuk memulai. Arduino Mega 2560 dapat memuat segala kebutukan dari projek ini. Selain itu juga bisa dapat terhubung dengan USB host shield yang digunakan untuk control *wireless*. Tampak atas dari Arduino Mega 2560[14].

Gambar 2.6. Arduino Mega 2560.
(Sumber : www.arduino.cc)

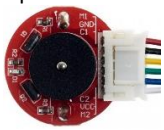
Adapun Spesifikasi board Arduino Mega 2560 adalah sebagai berikut :

1. Mikrokontroller ATmega2560
2. USB to Serial Chip ATmega AT16U2
3. Tegangan Operasi 5V
4. Tegangan Input (recommended) : 7-9V
5. Pin Digital I/O 54 Pin (15 PWM Output)
6. Pin Analog Input 16
7. Arus DC per pin I/O 40mA
8. Arus DC pin 3.3V 50mA
9. Flash Memory 256 KB dengan 8 KB digunakan untuk bootloader
10. SRAM 8 KB

11. EEPROM 4 KB
12. Clock Speed 16 MHz

2.6. Rotary Encoder

Rotary encoder adalah perangkat elektromagnetik yang dapat memonitoring gerak dan posisi. Rotary encoder tersusun dari suatu piringan tipis yang memiliki lubang-lubabg dibagian piringan. Di ujung kutub motor terdapat magnet yang akan dibaca oleh sensor hall magnetic Ketika berputar. Encoder digunakan untuk mengkur kecepatan dengan mengubah gerak putar menjadi sinyal digital yang kemudian penjadi parameter kecepatan motor DC[15].



Gambar 2.7. *encoder* motor DC
(Sumber : http://naylampmechatronics.com)

2.7. Arduino IDE

Arduino IDE (*Integrated Development Environment*) merupakan pengendali micro *single-board* yang bersifat *open source*, yang dirancang untuk mempermudah penggunaaan barang elektronik sebagai bidang. *Software* ini juga di *support* dengan library bahasa C/C++. Karena sifatnya yang terbuka maka siapa saja bisa mengunduhnya dan menggunakannya[16].

Gambar 2.8. *Software* Arduino IDE
(Sumber : www.arduino.cc)

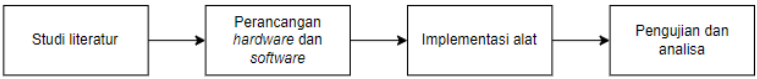
Bab 3. Metodologi Penelitian

3.1. Perancangan

Pada proses ini ada beberapa perancangan yang dilakukan yaitu peracangan penelitian, perancangan sistem, perancangan *hardware*, perancangan wirring keseluruhan, dan perancangan *software*, untuk memudahkan pengerjaan alat.

3.1.1. Perancangan Penelitian

Pada Pelaksanaan penelitian, digunakan metode-metode sistematis yang digunakan dalam menyelesaikan masalah dalam penelitian. Adapun motode yang akan dilakukan tersebut.

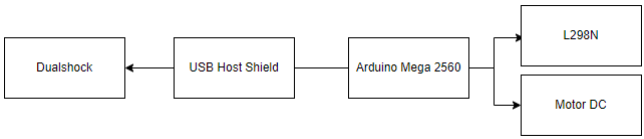


Gambar 3.1. Diagram blok perancangan penelitian

1. Studi literatur
   Mempelajari dan mengumpulkan teori keterkaitan dengan PID, Motor DC, dan tata cara melakukan logika pemrograman, serta melakukan pencarian berbagai referensi dari sumber-sumber yang relevan untuk membantu memudahkan pegerjaan tugas akhir.
2. Perancangan *hardware* dan *software.*
   Merancang perangkat keras dan perangkat lunak, yang berupa *prototype* dengan mempertimbangkan alat yang diperlukan. pada perancangan software berupa code program yang digunakan untuk pengendalian *hardware*.
3. Implementasi alat
   Pada tahap ini dilakukan pembuatan alat berdasarkan hasil perancangan alat yang sudah dilakukan sebelumnya.
4. Pengujian dan analisa
   Melakukan pengujian dan analisa pada sistem secara keseluruhan baik dari perangkat keras maupun perangkat lunak, dan melakukan pemeriksaan pada *hardware* dan code program pada *software*. Dari hasil pengujian, mengumpulkan data-data yang akan dianalisis untuk memahami dan dapat diidentifikasi.

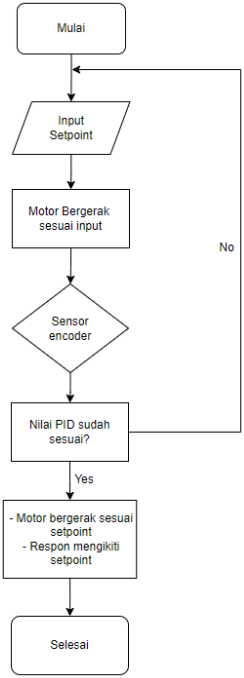
3.1.2. Perancangan Sistem

Perancangan sistem secara blok diagram yaitu untuk mempermudah dalam menganalisa rangkaian secara keseluruhan.



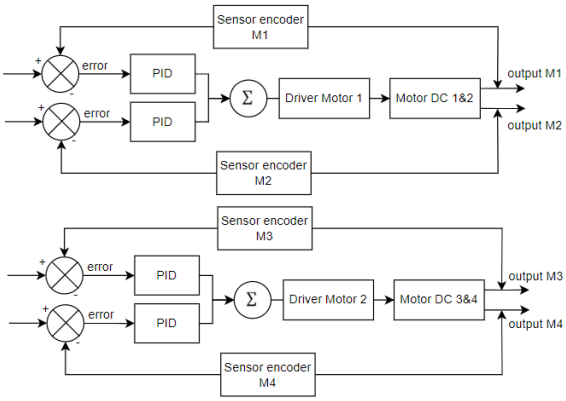
Gambar 3.2 Diagram blok sistem

Pada gambar 3.2 perancangan robot sudah dapat dikontrol dengan dualshoock yang terkomunikasi dengan USB host shield, yang selanjutnya akan diproses Arduino yang dikirimkan ke motor dc. Pergerakan motor DC dengan pengontrol motor menggunakan driver motor L298N. Pada sistem menggunakan motor DC yang di coupling dan salah satu keluaran motor DC yang akan bekerja sebagai *feedback.*



Gambar 3.3. *Flowchart* sistem kendali kecepatan motor

Pada gambar 3.3 menunjukan cara kerja kontrol kecepatan motor. Dengan memasukan nilai setpoint, lalu motor akan berputar dengan nilai yang sesuai dengan nilai yang diberikan set point tersebut. Sensor *encoder* yang terintegrasi pada motor dc dapat membaca nilai kecepatan motor lalu dari pembacaan tersebut akan dibandingkan dengan nilai referensi (nilai setpoint) yang nantinya akan mendapatkan nilai *error* yang akan digunakan sebagai masukan pengendali PID agar dapat mengontrol kecepatan motor DC sehingga mencapai nilai referensi yang diinginkan.



Gambar 3.4. Blok diagram sistem kendali PID *Crab_Bot 2023*

Pada gambar 3.4 dilakukan perancangan sistem PID pada robot *mecanum wheel* bertujuan untuk mempermudah merealisasikan *prototype* pengendali kecepatan motor yang akan dibuat. Cara kerja dari sistem pengendali kalam tertutup ini mengeluarkan sinyal *pulse width modulation*(PWM) untuk motor driver. Sinyal PWM berfungsi untuk mengatur tegangan yang keluar sesuai dengan set point. Pada bagian motor dc telah terpasang sensor *encoder* yang membaca *pulse* tiap putaran yang akan diberi nilai PID, nilai yang digunakan sebagai referensi untuk *looping* menuju set point.

3.1.3. Perancangan Hardware

1. Kontroller
   Kontroller merupakan perangkat keras utama yang digunakan sebagai alat pemrograman. Microcontroller yang digunakan adalah ATmega2560 yang terintegrasi dengan komponen penunjang lainnya pada board Arduinomega2560 dengan spesifikasi pada Tabel 3.1.

Tabel 3.1 Spesifikasi Arduino Mega 2560

No	Spesifikasi	Keterangan
1	Chipset	ATmega2560
2	Tegangan Operasional	5 Volt
3	Tegangan Rekomendasi	7-12 Volt
4	Batas Tegangan	6-20 Volt
5	Pin Input/Output Digital	56
6	Pin PWM	15
7	Pin Input Analog	16
8	Arus untuk Pin Digital	40 mA
9	Arus Untuk Pin 3.3 V	50 mA
10	Memori Flash	256 KB (8 KB Untuk *bootloader*)
11	SRAM	8 KB
12	EEPROM	4 KB
13	Clock Speed	16 Mhz
14	Panjang	10.1 cm
15	Lebar	5.3 cm
16	Berat	37 gram

2. Aktuator

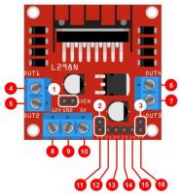
Aktuator adalah sebuah alat penggerak, Motor DC digunakan sebagai actuator pada robot, berikut spesifikasi pada motor DC pada Tabel 3.2.

Tabel 3.2 Spesifikasi Motor DC

NO	Spesifikasi	Keterangan
1	Speed	600rpm
2	Torsi	2.0Kg.cm
3	Arus	1.3 A
4	Tegangan Suplay	6VDC

3. Driver Motor

Driver Motor L298N adalah komponen elektronik yang digunakan untuk mengontrol kecepatan serta putaran motor DC.



Gambar 3.5. Driver motor L298N
(Sumber : http://mahirelektro.com)

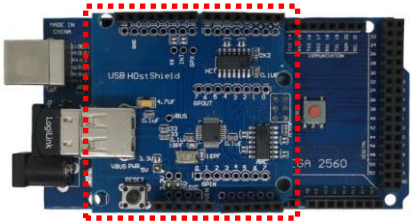
Spesifikasi driver motor L298N ditunjukan pada Table 3.3.

Tabel 3.3 Spesifikasi Driver motor L298N

NO	Keterangan
1	Jumper 12v
2	Jumper enable Motor 1
3	Jumper enable Motor 2
4	Output1 Motor 1(+)
5	Output2 Motor 1(-)
6	Output2 Motor 3(+)
7	Output2 Motor 4(-)
8	Sumber tegangan Motor 12V
9	Ground
10	Sumber Tegangan 5V (bisa digunakan untuk IC)
11	ENA, dihubungkan dengan pin PWM pada Arduino untuk mengontrol speed Motor 1
12	IN1
13	IN2
14	IN3
15	IN4
16	ENB, dihubungkan dengan pin PWM pada Arduino untuk mengontrol Speed Motor 4

4. USB Host Shield

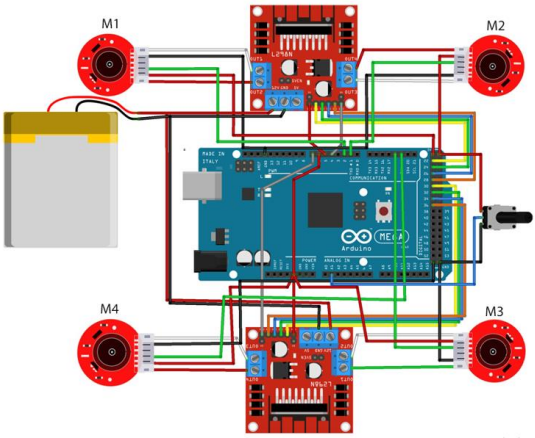
Usb Host Shield adalah *board add-on* sebagai penghubung antara Arduino dengan Dualshock mengunakan media USB *Bluetooth*, *dualshock* tersebut yang nantinya digunakan sebagai pengontrol robot. Shield ini bisa digunakan untuk dengan jarak jauh tanpa kabel *(wireless).*Tampilan atas USB Host Shield to Arduino Mega 2560 dilihat pada gambar 3.5.



Gambar 3.6. USB Host Shield
(Sumber : http://printed-droid.com)

3.1.4. Perancangan Wirring Keseluruhan

Perancangan keseluruhan komponen, menggunakan Arduino Mega 2560, motor DC encoder, driver motor L298N, Batrai, dan tambahan potensio meter untuk mempermudah menentukan nilai set point. *Scematic* keseluruhan dapat dilihat pada gambar 3.6.



Gambar 3.7. Wirring komponen

3.1.5. Perancangan Software

Perancangan software digunakan untuk mengintegrasikan seluruh perangkat keras dan menjadi pusat kendali dari alat. Program yang digunakan menggunakan Bahasa Arduino yaitu bahaca C/C++ yang bisa digunakan Arduino mega 2560.

Gambar 3.8. Rancangan program

3.2. Alat dan Bahan

Alat dan bahan ini disediakan oleh pihak Kampus dan ada beberapa komponen yang dibeli dengan dana pribadi, Tempat pengerjaan alat disediakan Kampus.

Tabel 3.4 Estimasi biaya

No.	Alat/bahan	Harga Satuan (Rp.)	Jumlah	Total (Rp.)
1	Motor DC	Rp. 156.000	4	Rp. 624.000
2	Roda Mecanum	Rp. 80.000	4	Rp. 320.000
3	Driver Motor	Rp. 25.000	2	Rp. 50.000
4	Arduino Mega	Rp. 250.000	1	Rp. 250.000
5	USB Host Shield	Rp. 165.000	1	Rp. 165.000
6	Dualshock 3	Rp. 55.000	1	Rp. 55.000
7	Batrai	Rp. 350.000	1	Rp. 350.000
8	Siku	Rp. 16.000	4	Rp. 64.000
9	Plat Besi	Rp. 10.000	1	Rp. 10.000
10	Couple	Rp. 50.000	4	Rp. 200.000
11	Donggle V4	Rp. 80.000	1	Rp. 80.000
12	Kabel Jumper	Rp.1.000		Rp. 20.000
	Total			Rp. 2.188.000

3.3. Pengujian

Pengujian dilakukan untuk memastikan bahwa sItem yang dibuat berfungsi dengan baik sesuai dengan tujuan yang diinginkan. Untuk pengerjaan tugas akhir ini, pengujian dilakukan dengan metode manual *tunning* PID.

Ada beberapa pengujian yang dilakukan sebelum proses pengujian PID, Diantaranya, Kalibrasi RPM untuk melihat hasil kecepatan motor DC. Pengujian gerak robot untuk mengontrol perpindaha robot, serta pengujian kontrol open loop untuk membandingkan hasil yang didapat dengan kontrol PID.

3.3.1. Kalibrasi RPM

Pada proses ini, perlunya dilakukan pengkalibrasin pada motor DC yang dikarenakan pada karaketer setiap motor memiliki karakteristik yang berbeda-beda yang dapat mempengaruhi performa kecepatannya, seperti non-linieritas, encoder offset, histeresis, dan reaksi balik. Dengan dikalibrasi, motor DC dapat meningkatkan keakuratan pengukuran kecepatan motor, kestabilan sistem kendali, mengurangi kesalahan dan osislasi pada sistem, dan meningkatkan kinerja keseluruhan sistem. seperti yang ada di standar internasional IEC 60034-1, ini banyak digunakan yang menetapkan persyaratan untuk motor DC. Menurut

standar ini kecepatan motor DC biasanya diukur dalam putaran per menit (RPM) atau radian per detik *(rad/s).*

Tabel 3.5 Pengukuran RPM Motor DC

Kecepatan yang diset	Motor 1(E_1)	Motor 2(E_2)	Motor 3(E_3)	Motor 4(E_4)

3.3.2. Pengujian Putaran Motor Pada Gerak Robot

Pengujian putaran motor untuk gerak perpindahan dilakukan untuk menguji arah putaran tiap-tiap motor yang terpasang roda *mecanum* yang dihasilkan dari putaran tersebut.

Table 3.6 Pengujian arah putaran robot

Arah Robot	Motor 1	Motor 2	Motor 3	Motor 4
Maju				
Mundur				
Kiri				
Kanan				
Putar Kiri				
Putar Kanan				

3.3.3. Pengujian Kecepatan Motor dengan Kontrol Open Loop

Dalam pengengujian sistem kendali *open loop* yang mana output tidak memiliki pengaruh terhadap sistem, dari data yang dikeluarkan ketika di beri tengangan atau satuan dengan kecepatan tertentu, tidak sesuai. Karakteristik pada setiap motor yang berbeda-beda menjadikan perbedaan setiap kecepatan, perbedaan tersebut yang mempengaruhi pergerakan pada robot.

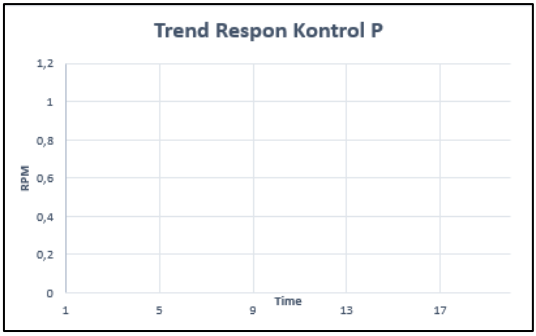
Table 3.7 Pengujian kecepatan motor dengan kontrol *open loop*

NO	Kecepatan yang diset	Motor 1(E_1)	Motor 2(E_2)	Motor 3(E_3)	Motor 4(E_4)
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

3.3.4. Pengujian Kecepatan Motor dengan Kontrol PID

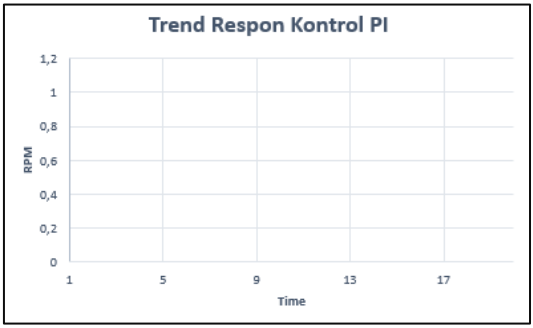
Pada proses ini, pengujian dilakukan dengan 3 tahapan yaitu, variable tunning *Proportional* (P), *Integral* (I), dan *Derivative* (D). tahap pertama dengan melakukan kontrol P dengan mengkonversikan ke gein Kp, setelah mendapatkan nilai untuk Kp, dilanjutkan dengan kontrol Ki, dan Kd.

A. *Tunning* P



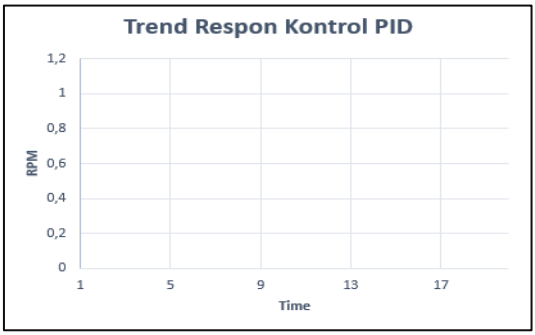
Gambar 3.9. *Trend* respon kontrol P

B. *Tunning* PI



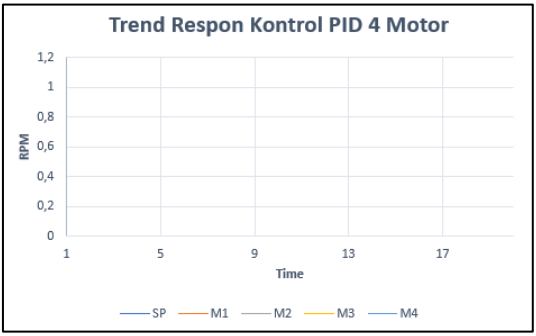
Gambar 3.10 *Trend* respon kontrol PI

C. *Tunning* PID



Gambar 3.11 *Trend* respon kontrol PID

D. Respon Kontrol PID 4 Motor



Gambar 3.12. *Trend* respon kontrol PID 4 Motor

Bab 4. Hasil dan Pembahasan

Tahapan ini terdiri dari hasil pengujian hasil kalibrasi RPM, Pengujian putaran motor pada gerak robot, pengujian kontrol *open loop*, Pengujian kontrol PID, perbandingan kontrol PID dan *open loop*, dan hasil pergerakan robot, hasil pengujian ini dapat di analisis sebagai untuk kesimpulan yang pada setiap data yang diperoleh.

4.1. Hasil Kalibrasi RPM

Pada proses ini, pengkalibrasian dilakukan dengan menghitung kecepatan motor DC dengan pembacaan sensor *rotary encoder*, keempat motor akan dilakukan pengujian secara bersaaman untuk melihat perbandingan motor.

Tabel 4.1 Hasil kalibrasi Motor DC

Kecepatan yang diset (PWM)	Motor 1(E_1) (RPM)	Motor 2(E_2) (RPM)	Motor 3(E_3) (RPM)	Motor 4(E_4) (RPM)
0	0	0	0	0
20	0	0	0	0
30	0	10.0	12.5	12.5
50	84.5	175.5	182	182
50	97.5	175.5	182	182
50	91	175.5	182	188.5
50	97.5	175.5	182	182
50	91	175.5	175.5	182
50	97.5	175.5	182	182
50	97.5	169	182	182
50	97.5	175.5	182	182
50	91	175.5	182	182
50	97.5	175.5	182	182
50	97.5	175.5	182	182
50	97.5	175.5	182	188.5
100	344.5	435.5	422.5	429
100	351	435.5	429	435.5
100	351	429	422.5	435.5
100	351	429	429	429
100	351	435.5	422.5	435.5
100	351	435.5	429	429
100	351	435.5	422.5	435.5
100	351	435.5	429	435.5

100	351	435.5	422.5	429
100	351	435.5	429	435.5
100	351	435.5	422.5	435.5
100	351	435.5	429	429
200	539.5	572	546	572
200	533	572	546	565.5
200	533	572	546	572
200	533	572	546	565.5
200	533	578.5	546	572
200	533	572	546	565.5
200	533	572	546	565.5
200	533	572	546	565.5
200	533	572	546	572
200	533	578.5	546	565.5
200	539.5	572	546	572
255	611	617.5	585	617.5
255	611	611	591.5	617.5
255	604.5	617.5	585	617.5
255	611	617.5	585	611
255	611	611	585	617.5
255	611	611	585	617.5
255	604.5	617.5	585	611
255	611	617.5	585	617.5
255	604.5	617.5	591.5	611
255	604.5	617.5	585	617.5
255	611	611	585	617.5

Pada prosesnya motor diberi tegangan 1-5V yang dikonversikan ke PWM 0-255 dengan keluaran pada code arduino dituliskan *analogWrite(PinA, speed_data*(read_encoder/int offset));* sensor akan membaca putaran motor radian per detik *(rad/s)* dengan keluaran RPM. Pada tabel 4.1 terlihat bahwa hasil yang didapat bahwasanya kecepatan tiap motor berbeda-beda, motor 1 dapat berputar jika diberi tegangan minimal 50PWM, sedangkan motor 2, 3, dan 4 dapat berputar jika diberi tegangan minimal 30PWM. perbedaan keluaran motor tersebut yang akan diperbaiki oleh sistem PID.

4.2. Pengujian Putaran Motor Pada Gerak Robot

Pengujian putaran motor dilakukan secara langsung dengan memasang roda *mecannum* untuk melihat gerak perpindahan robot dengan hasil yang dapat dilihat pada tabel 4.2.

Tabel 4.2 Konfigurasi Puaran Motor untuk Gerak Robot

Arah Robot	Motor 1	Motor 2	Motor 3	Motor 4
Maju	CW	CW	CW	CW
Mundur	CCW	CCW	CCW	CCW
Kiri	CCW	CW	CCW	CW
Kanan	CW	CCW	CW	CCW
Putar Kiri	CCW	CCW	CW	CW
Putar Kanan	CW	CW	CCW	CCW

4.3. Hasil Pengujian Kecepatan Motor dengan Kontrol Open Loop

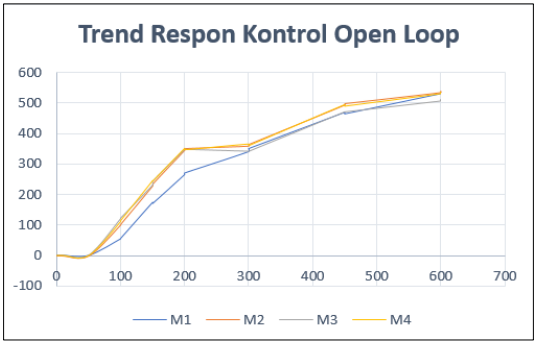
Dari hasil pengujian kontrol *open loop (without feedback)* nilai pada setiap motor berbeda beda. Sehingga dapat mempengaruhi kecepatan dan pergerakan pada robot, sehingga tidak efisien untuk digunakan untuk kecekatan pada robot untuk presisi. *Stedy state* pada motor memiliki perbedaan respon yang cepat, yang memiliki *error*.

Tabel 4.3 Hasil Pengujian Kecepatan Motor dengan Kontrol Open Loop.

SP	M1	M2	M3	M4
0	0	0	0	0
10	0	0	0	0
50	0	0	0	0
100	54.5	102	120	112
100	54.5	104	120	115
100	56.1	102	122	115
100	56.1	104	122	115
150	174.3	230.1	230.1	245.9
150	170.5	235.8	242	243.8
200	265.2	346.8	346.8	351.9
200	265.2	346.8	346.8	351.9
200	270.3	351.9	346.8	346.8
300	340.5	360.1	340.5	365.5
300	350.4	365.5	340.1	360.1

300	350.4	365.5	340.1	360.1
450	469	494.7	469.2	494.7
450	464.1	499.5	469.2	489.6
450	464.1	499.5	469.2	489.6
600	530.4	535.5	504.9	530.4
600	530.4	540.4	504.9	530.4
600	535.5	535.5	510	535.5

Pada proses pengujian di Arduino sinyal PWM 0-255 dikonversikan menjadi RPM untuk memudahkan pembacaan hasil pengujian kecepatan motor yang terlihat pada gambar 4.1.



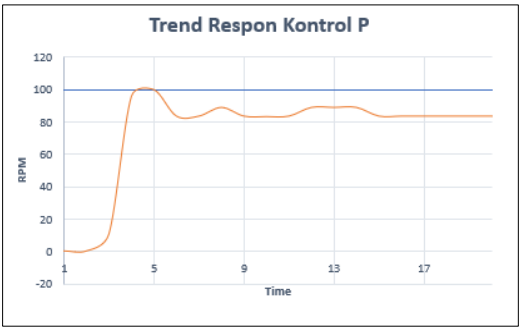
Gambar 4.1. *Trend* respon kontrol *open loop*

4.4. Hasil Pengujian Kecepatan Motor dengan Kontrol PID

Proses pengujian ini dilakukan dengan metode manual *tunning,* adapun tahapannya diantaranya yaitu:

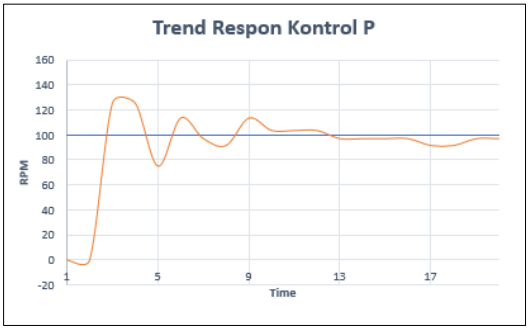
1. ***Tunning* P**

   Pada percobaanya *trial and error*, percobaan pertama *tunning* parameter P dengan Kp = 2.5. Terlihat pada gambar 4.2 hasil yang didapatkan memiliki respon sistem yang baik, sistem dapat mencapai set-point dan tidak ada *overshoot*, akan tetapi ada nilai *stady state error* yang cukup jauh antara set-point yang membuat sistem tidak bekerja dengan maksimal.



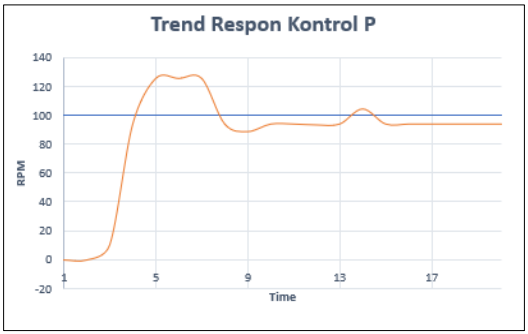
Gambar 4.2 *Tunning* kontrol P pertama

Percobaan *tunning* kedua dilakukan dengan tujuan untuk memperkecil *error stady state* dengan nilai Kp = 5.1, terlihat pada gambar 4.3 bahwa semakin memperbesar nilai maka akan terjadi sistem osilasi yang naik turun membuat sistem tidak stabil.



Gambar 4.3 *Tunning* kontrol P kedua

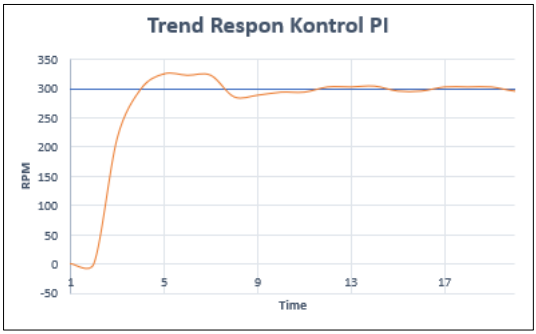
Dilakukan percobaan *tunning* ketiga dengan menurunkan nilai Kp = 1.1, pada hasil tersebut sudah mendapatkan respon sistem yang cukup baik, walaupun ada terjadi getaran naik pada sistem pada awalannya. Dengan terdapat nilai *error stasy state* 6,4. Terlihat pada gambar 4.4 bahwasanya ada respon *overshoot* yang membuat respon yang kurang baik, jika melihat polanya semakin besar nilai Kp, tidak akan mengurangi nilai *overshoot* maka dari ini dubutuhkannya penggali tambahan lainnya.



Gambar 4.4 *Tunning* kontrol P ketiga

2. *Tunning PI*

Pada kali ini ditambahkannya kontrol *integral,* karakternya pada kontrol ini cukup sama seperti kontrol *proportional* yang memberikan *step respon* yang baik, pengendali ini bersifat untuk menghilangkan nilai *error stady state* yang sudah dikalkukasikan dengan kontrol *proportional.*

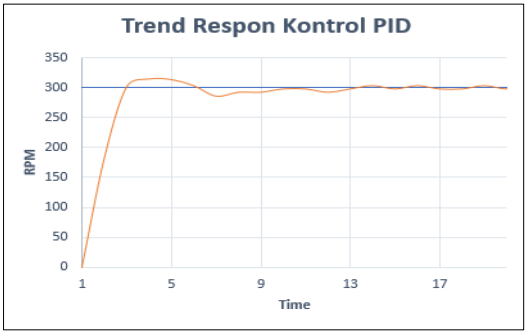


Gambar 4.5 *Tunning* kontrol PI

Pada gambar 4.5 terlihat bahwa sistem yang diberikan oleh kontrol *integral* sangat signifikan, bahwa tidak ada lagi nilai *error stady state* pada sistem, dengan nilai Ki = 3.5 nilai ini sudah dioptimalkan dan lebih tinggi dari nilai Kp sebab kestabilan rotasi (*inersia*) lebih penting dari kekuatan mekanis dalam menentukan stabilitas kecepatan.

3. Tunning PID

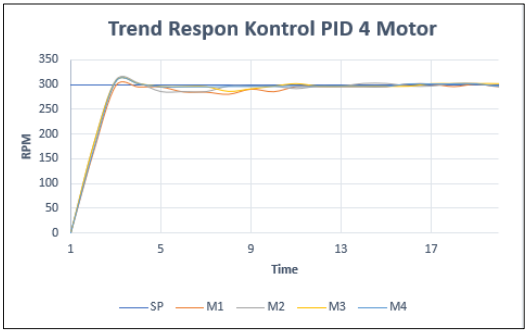
Pada proses ini adalah tahapan terakhir yaitu menambahkan kontrol *derivative,* proses ini penggabungan gain Kp, Ki, Kd.



Gambar 4.6. *Tunning* PID.

Pada kontrol ini untuk meng-optimalkan sistem, terlihat pada gambar 4.6 kontrol *derivative* tidak merubah *error stady state*(2,5) akan tetapi memperkecil overshoot(12,5) yang memperbaiki sistem dengan optimal, dengan nilai Kp = 0.1.

4. Respon Kontrol PID 4 Motor

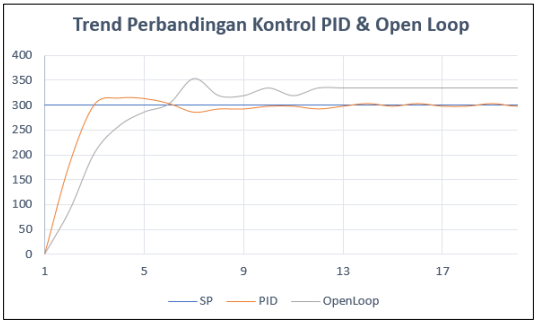


Gambar 4.7 Trend respon kontrol PID 4 Motor

Pada pengujian ini nilai PID yang digunakan sama pada keempat motor, dikarenakan spesifikaksi motor dan hasilnya respon tiap motor cukup sama dan tidak mendapatkan kendala teknis jika dibandingkan dengan respon tiap motor pada kontrol *open loop*.

4.5. Perbandingan Kontrol PID dan Open Loop

Dari analisa yang dilakukan bisa dilihat pada gambar 4.8 dibawah, pengujian dilakukan dengan memberikan input *set-point* yang sama antara kontrol PID dengan kontrol *open loop,* sistem *open loop* memiliki *respon time* yang cukup jauh dibandingkan dengan sistem PID, dengan nilai *overshoot 58.5* dan *error stady state 34.5* dibandingkan dengan kontrol PID dengan nilai *overshoot 12.5* dan *error stady state 2.5*

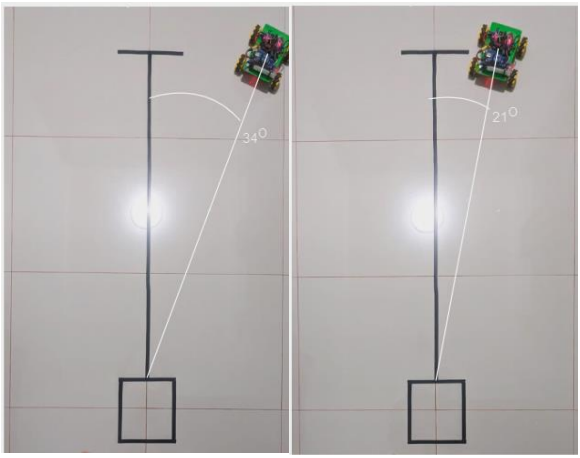


Gambar 4.8 Trend Perbandingan Kontrol PID dan Open Loop

4.6. Hasil Pergerakan Robot

Pengujian ini dilakukan dengan kedua kontrol yang berbeda untuk menguji ketepatan perpindahan robot. Dilakukan dipermukaan lantai pada gerak maju dengan jarak 1,5m.

1. **Pengujian Gerak Robot Dengan Kontrol Open Loop**
   Dari percobaan yang dilakukan menjalankan robot dengan controller dengan gerak maju bahwa kontrol *open loop* tidak presisi dengan nilai rata-rata kemiringan 27,5^{o} kekanan. Dikarenakan pada motor 1 lebih lambat dibandingkan dengan motor lain, sehingga kinematik pada roda *mecanum* yang dihasilkan pada kecepatan pada motor DC mempengaruhi pergerakan pada robot.



Gambar 4.9 Perpindahan robot dengan kontrol *open loop*

2. Pengujian Gerak Robor Dengan Kontrol PID

Pengujian ini dilakukan sama seperti kontrol open loop, dengan sistem PID. Robot bergerak stabil dengan mempertahankan kecepatan tiap-tiap motor dengan *error* 4^{o}.

Gambar 4.10 Perpindahan robot dengan kontrol PID

Bab 5. Kesimpulan dan Saran

5.1. Kesimpulan

Berdasarkan dari pengujian dan analisa dari penelitian yang berjudul "*Quad-Drive Speed Control* berbasis PID pada *Barelang Crab_Bot 2023*" ini maka dapat disimpulkan sebagai berikut:

1. Pengujian arah perpindahan gerak pada robot, setelah diuji dan dibandingkan dengan kedua kontrol, kontrol PID dengan *error* kemiringan 4^{o} lebih baik dibandingankan kontrol *open loop* dengan *error* kemiringan 27,5^{o}.
2. Dengan menggunakan metode PID untuk mengontrol masing masing motor DC dengan roda *mecannum*, respon sistem pada motor DC cukup stabil, nilai *overshoot* dari *open loop* 58.5 menjadi 12.5 dan *error steady state* 34.5 menjadi 2.5.

5.2. Saran

Kontrol pada robot dalam penelitian mampu bergerak dengan hadap error yang relative kecil. Dibutuhkannya perhitungan *center of gravity* dari robot, agar beban yang diterima masing-masing roda sama, untuk mendapatkan hasil yang lebih masksimal. Penempatan roda *mecanum* pada motor DC harus tepat jika terjadinya kemiringan maka akan berpengaruh pada pegerakan robot yang tidak tepat.

Penambahan sistem pengendali tambahan seperti *logic position fuzzy controll* yang lebih kompleks, serta memiliki toleransi terhadapt data-data yang tidak tepat.

Daftar Pustaka

[1] DIKTI Puspesnas, “Buku panduan kontes robot Indonesia 2023,” Kementrian Pendidikan dan Kebudayaan, 2023.

[2] Kumar, S., Kumar, V., & Kumar, P. (2020). Control systems for robotics and automation: A review. Journal of Intelligent Manufacturing, 31(2), 347-356.

[3] I. Zeidis, K. Zimmermann. *Dynamics of a four-wheeled mobile robot with Mecanum wheels. Z Angew Math Mech*. 2019;99:e201900173.

[4] R. Harahap *and* S. Nofriadi, “Analisa Perbandingan Efisiensi dan Torsi dengan menggunakan Metode Peyadapan Sejajar Terhadap Kompon Pendek Dengan Kutub Bantu,” J. Electr. Technol., vol. 4, no. 3, pp. 105-111, 2019.

[5] Mahendra B N, Rades S, “Penerapan Sistem Kendali PID pada Antena Pendeteksi Kordinat Posisi UAV”, Universitas Gajah Mada, 2015.

[6] Panca Agung K , Andi D, “Pengendalian Kestabilan Ketinggian pada Penerbangan Quadrotor dengan Metode PID Fuzzy,” Universitas Gajah Mada, 2017.

[7] Andrean George W, Abdul Rahman, “Straight-Move Robot Control System with LabView-Based Proportional Integral Derivative (PID) Control,” 2015.

[8] Huang G., Lee S., *PC-based PID Speed Control in DC Motor*, IEEE, 978-1-4244-1724-7/08/$25.00, 2008.

[9] Rahul Malhotra, Tejbeer Kaur, Gurpreet Singh Deol, ”*DC motor control using fuzzy logic controller,” international journal of advanced engineering sciences and Technologies*, 2011.

[10] Park,J, Kim,S, Driving Control of Mobile Robot with Mecanum Wheel using Fuzzy Inference System. Word Academy of Science Engineering Technology, Vol.6, pp2519-2523, 2010.

[11] Thahja Odinanto, Bambang Suprijono dan Winda Andrianta Widya Natasari. “*Perancangan pengendali kecepatan motor arus searah 1 HP volt dengan kontrol PID berbasis mikrokontroller,*“ Universitas ITATS, Surabaya, 2015.

[12] Arif, Muhamad Faishol, “*Sistem Kontrol Kecepatan Motor DC D-6759 Berbasis Arduino Mega 2560,” 2018.*

[13] Fatoni, Rinal Achma, “*Sistem Navigasi Robot Beroda Berbasis Teknik Kendali PID Menggunakan Mikrokontroller Arduino Mega2560*,” Cimahi UNJANI, 2017.

[14] Eda. (2017). spesifikasi arduino mega 2560 rev3, eda channel.com

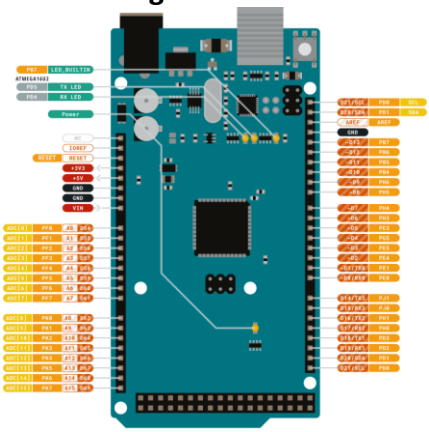
[15] Muhammad Reza Aditya Nurkholis Putera1, Rahmat Hidayat, “Kendali kecepatan motor dc menggunakan pengendali PID dengan encoder sebagai feedback”, 2022.
[16] Arduino.(2023).ArduinoSoftware(IDE).https.//www.arduino.cc/en/Guide/Environtment.
[17] Nida Nur Rokhmah, “*Kendali kecepatan motor dc dengan metode pid berbasis miktokontroller“,* Universitas Jendral Ahmad Yani, Yogyakarta, 2018.

Biodata

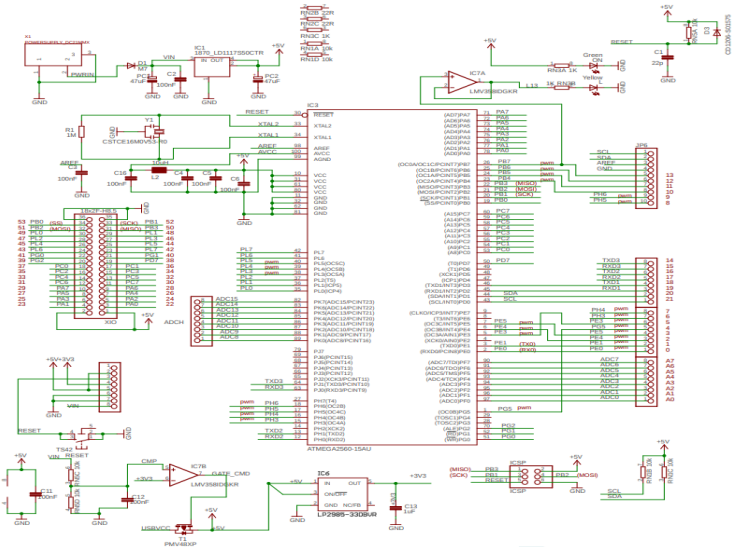
Nama	: Muhammad Akbar
TTL	: Palembang, 20 September 2002
Agama	: Islam
Alamat	: Sei Pancur Blok D No.17 RT001 RW003, Tanjung Piayu, Batam.
Email	: muhammadakbar0121@gmail.com
Riwayat Pendidikan	: SMK : SMKN 2 Bandar Lampung. SMP : SMPN 29 Kota Sepang.

Lampiran

1. Datasheet Arduino Mega 2560



Gambar 1 *Datasheet* Arduino Mega 2560
(Sumber : Arduino Mega 2560 Rev3.com)



Gambar 2 *Schematic* Arduino Mega 2560
(Sumber : Arduino Mega 2560 Rev3.com)

2. Datasheet Motor DC

Tabel 1. Spesifikasi Motor DC

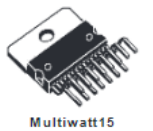
SPECIFICATIONS	
System Volatage	12V
Model	JGA25-370-CE
Type	DC Encoder
Current	2.0A
Speed	600rpm
Stall Torque	1kg.cm
Holding Torque	2.0kg.cm
Power Consumption Standard	0.45A
Power Consumption Max Load	1.3A
Variable L pada gambar dimensi	21mm
Saft	4mm
Outer diameter	25mm length dimension based on different parameters of different length
Long	axis length 8mm 12mm D
Weight	100g different than

3. Datasheet L298N

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

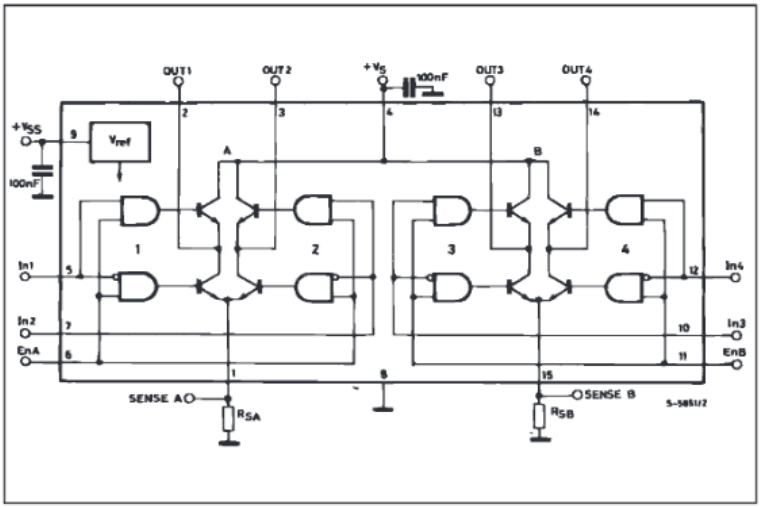


ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM



Gambar 3 *datasheet* L298N
(Sumber : alldatasheet/l298n+datasheet.com)

4. Datasheet USB Host Shield

Genaral Description

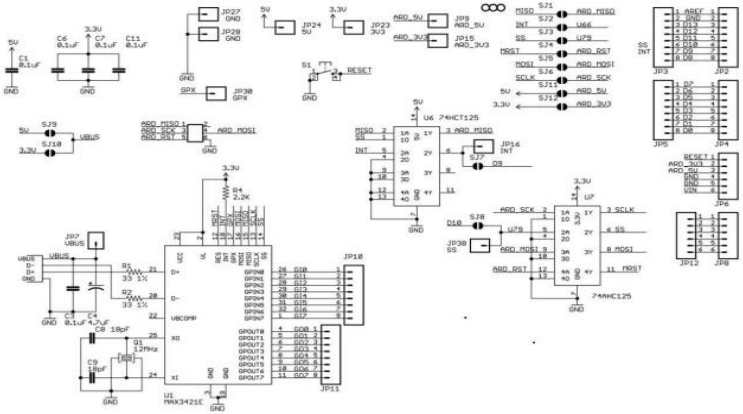
The Keyes USB Host Shield allows you to connect a USB device to your Arduino board. It is based on the MAX3421E, which is a USB peripheral/host controller containing the digital logic and analog circuitry necessary to implement a full-speed USB peripheral or a full-/low-speed host compliant to USB specification rev 2.0.

This is based on revision 2.0 of USB Host Shield. Thanks to new interface layout it is now compatible with more Arduinos - not only UNO and Duemilanove, but also Mega and Mega 2560 work with Standard variant of this shield out of the box. No more SPI re-wiring and code modifications - just solder included stackable connectors (2x3 ICSP connector's female side should be facing down), plug and play!

Specifications

- Works with standard (dual 5/3.3V) and 3.3V-only (for example, Arduino Pro) boards.
- Operates over the extended -40°C to +85°C temperature range
- Complies with USB Specification Revision 2.0 (Full-Speed 12Mbps Peripheral, Full-/LowSpeed 12Mbps/1.5Mbps Host)
- The following device classes are currently supported by the shield:
- HID devices, such as keyboards, mice, joysticks, etc.
- Mass storage devices, such as USB sticks, memory card readers, external hard drives (FAT32 Type File System - Arduino Mega only)

Schematics



Gambar 4 schematics USB Host Shield
(Sumber : arduino/usbhostshiels-rev2.com)

5. Coding Program Ardino IDE

```
#include <PS3BT.h>
#include <usbhub.h>

#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>

USB Usb;
BTD Btd(&Usb);
PS3BT PS3(&Btd);
bool printTemperature, printAngle;

float sp1=0.0;
float sp2=0.0;
float sp3=0.0;
float sp4=0.0;

int IN1 = 22;//motor_driver_kiri
int IN2 = 24;//motor_driver_kiri
int IN3 = 26;//motor_driver_kiri
int IN4 = 28;//motor_driver_kiri
int IN11 = 30;//motor_driver_kanan
int IN12 = 32;//motor_driver_kanan
int IN13 = 34;//motor_driver_kanan
int IN14 = 36;//motor_driver_kanan
int EN1= 4;
int EN2= 5;
int EN3= 6;
int EN4= 7;
float pv1;
float pv2;
float pv3;
float pv4;

int switch_kecepatan=1;
int enc1 = 2;
```

```
int enc2 = 3;
int enc3 = 18;
int enc4 = 19;
volatile int counter1 = 0;
volatile int counter2 = 0;
volatile int counter3 = 0;
volatile int counter4 = 0;
unsigned long previousMillis = 0;
long interval = 100;
unsigned int delayer = 0;

float cv_motor1;
float cv1_motor1;
float cv_motor2;
float cv1_motor2;
float cv_motor3;
float cv1_motor3;
float cv_motor4;
float cv1_motor4;
float error_motor1;
float error1_motor1;
float error2_motor1;
float error_motor2;
float error1_motor2;
float error2_motor2;
float error_motor3;
float error1_motor3;
float error2_motor3;
float error_motor4;
float error1_motor4;
float error2_motor4;
float Kp = 1.1;
float Ki = 3.5;
float Kd = 0.01;
float Tm = 0.1;
float kr = 250;

float EN1_maju= 0.0;
float EN1_mundur=0.0;
float EN2_maju=0.0;
float EN2_mundur=0.0;
float EN3_maju=0.0;
```

```
float EN3_mundur=0.0;
float EN4_maju=0.0;
float EN4_mundur=0.0;
float pwm1,pwm2,pwm3,pwm4,pwm5,pwm6,pwm7,pwm8;

void pid_speed_control_task(void)
{
 pv1 = 120*counter1*(60.0/600.0);
 counter1 = 0;
 pv2 = 120*counter2*(60.0/600.0);
 counter2 = 0;
 pv3 = 120*counter3*(60.0/600.0);
 counter3 = 0;
 pv4 = 120*counter4*(60.0/600.0);
 counter4 = 0;

 error_motor1 = sp1 - pv1;
 error_motor2 = sp2 - pv2;
 error_motor3 = sp3 - pv3;
 error_motor4 = sp4 - pv4;

 cv_motor1 = cv1_motor1 + (Kp + Kd/Tm)*error_motor1 + (-Kp + Ki*Tm -
2*Kd/Tm)*error1_motor1 + (Kd/Tm)*error2_motor1;
 cv1_motor1 = cv_motor1;

 cv_motor2 = cv1_motor2 + (Kp + Kd/Tm)*error_motor2 + (-Kp + Ki*Tm -
2*Kd/Tm)*error1_motor2 + (Kd/Tm)*error2_motor2;
 cv1_motor2 = cv_motor2;

 cv_motor3 = cv1_motor3 + (Kp + Kd/Tm)*error_motor3 + (-Kp + Ki*Tm -
2*Kd/Tm)*error1_motor3 + (Kd/Tm)*error2_motor3;
 cv1_motor3 = cv_motor3;

 cv_motor4 = cv1_motor4 + (Kp + Kd/Tm)*error_motor4 + (-Kp + Ki*Tm -
2*Kd/Tm)*error1_motor4 + (Kd/Tm)*error2_motor4;
 cv1_motor4 = cv_motor4;

 error2_motor1 = error1_motor1;
 error1_motor1 = error_motor1;

 error2_motor2 = error1_motor2;
 error1_motor2 = error_motor2;
```

```
error2_motor3 = error1_motor3;
error1_motor3 = error_motor3;

error2_motor4 = error1_motor4;
error1_motor4 = error_motor4;

if (cv_motor1 > 700.0){
  cv_motor1 = 700.0;
  }
if (cv_motor1 < 30.0){
  cv_motor1 = 30.0;
  }

if (cv_motor2 > 700.0){
  cv_motor2 = 700.0;
  }
if (cv_motor2 < 30.0){
  cv_motor2 = 30.0;
  }

if (cv_motor3 > 700.0){
  cv_motor3 = 700.0;
  }
if (cv_motor3 < 30.0){
  cv_motor3 = 30.0;
  }

if (cv_motor4 > 700.0){
  cv_motor4 = 700.0;
  }
if (cv_motor4 < 30.0){
  cv_motor4 = 30.0;
  }
pwm1 = cv_motor1*(255.0/700.0);
pwm2 = cv_motor2*(255.0/700.0);
pwm3 = cv_motor3*(255.0/700.0);
pwm4 = cv_motor4*(255.0/700.0);
analogWrite(EN1, pwm1);// pwm motor depan kiri
analogWrite(EN2, pwm2); //pwm motor belakang kiri
analogWrite(EN3, pwm3); // pwm motor belakang kanan
analogWrite(EN4, pwm4); // pwm motor depan kanan
```

```
}

void setup() {
  Serial.begin(115200);
  pinMode(enc1, INPUT);
  pinMode(enc2, INPUT);
  pinMode(enc3, INPUT);
  pinMode(enc4, INPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(IN11, OUTPUT);
  pinMode(IN12, OUTPUT);
  pinMode(IN13, OUTPUT);
  pinMode(IN14, OUTPUT);
  pinMode(EN1, OUTPUT);
  pinMode(EN2, OUTPUT);
  pinMode(EN3, OUTPUT);
  pinMode(EN4, OUTPUT);
  attachInterrupt(0, interruption1, RISING);
  attachInterrupt(1, interruption2, RISING);
  attachInterrupt(5, interruption3, RISING);
  attachInterrupt(4, interruption4, RISING);
#if !defined(__MIPSEL__)
  while (!Serial);
#endif
  if (Usb.Init() == -1) {
    Serial.print(F("\r\nOSC did not start"));
    while (1);
  }
  Serial.print(F("\r\nPS3 Bluetooth Library Started"));
}
void loop() {
  Usb.Task();

  if (PS3.PS3Connected || PS3.PS3NavigationConnected) {
    if (PS3.getAnalogHat(LeftHatX) > 137)
    {
      KANAN();
    }
    esle if (PS3.getAnalogHat(LeftHatX) < 117)
```

```
{
  KIRI();
}
else if (PS3.getAnalogHat(LeftHatY) > 137)
{
  MAJU();
}
if (PS3.getAnalogHat(LeftHatY) < 117)
{
  MUNDUR();
}
else if (PS3.getAnalogButton(L2) > 137)
{
  PUTARKIRI();
}
else if (PS3.getAnalogButton(R2) > 137)
{
  PUTARKANAN();
}
else
{
  STOP();
}

if (PS3.getButtonClick(PS)) {
  Serial.print(F("\r\nPS"));
  PS3.disconnect();
}
if (PS3.getButtonClick(L1))
{
  switch_kecepatan++;
  if (switch_kecepatan>1)
  {
    switch_kecepatan = 1;
  }
  if (switch_kecepatan == 0)
  {
    kr = 100;
  }
  else if (switch_kecepatan == 1)
  {
    kr = 250;
```

```
    }
   }
   if (PS3.getButtonClick(R1))
   {
    switch_kecepatan--;
    if (switch_kecepatan<0)
    {
     switch_kecepatan =0;
    }
    if (switch_kecepatan == 0)
    {
     kr = 100;
    }
    else if (switch_kecepatan == 1)
    {
     kr = 250;
    }
   }

   unsigned long currentMillis = millis();
    if (currentMillis - previousMillis > 50 )
    {
//     Serial.println(currentMillis - previousMillis );
      pid_speed_control_task();
      previousMillis = currentMillis;
    }
 }
}
#if 0 // Set this to 1 in order to enable support for the Playstation Move controller
  else if (PS3.PS3MoveConnected) {
   if (PS3.getAnalogButton(T)) {
    Serial.print(F("\r\nT: "));
    Serial.print(PS3.getAnalogButton(T));
   }
   if (PS3.getButtonClick(PS)) {
    Serial.print(F("\r\nPS"));
    PS3.disconnect();
   }
 }
#endif
}
////////////////////////////////////MOVEMENT/////////////////////////////////
```

```
void MAJU()
{
  sp1 = kr;
  sp2 = kr;
  sp3 = kr;
  sp4 = kr;
  digitalWrite(IN1, High);
  digitalWrite(IN2, LOW);
  digitalWrite(IN4, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN11, HIGH);
  digitalWrite(IN12, LOW);
  digitalWrite(IN14, HIGH);
  digitalWrite(IN13, LOW);
}
void MUNDUR()
{
  sp1 = kr;
  sp2 = kr;
  sp3 = kr;
  sp4 = kr;
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN4, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN11, LOW);
  digitalWrite(IN12, HIGH);
  digitalWrite(IN14, LOW);
  digitalWrite(IN13, HIGH);
}
void KIRI()
{
  sp1 = kr;
  sp2 = kr;
  sp3 = kr;
  sp4 = kr;
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN4, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN11, LOW);
  digitalWrite(IN12, HIGH);
```

```
  digitalWrite(IN14, HIGH);
  digitalWrite(IN13, LOW);
}
void KANAN()
{
  sp1 = kr;
  sp2 = kr;
  sp3 = kr;
  sp4 = kr;
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN4, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN11, HIGH);
  digitalWrite(IN12, LOW);
  digitalWrite(IN14, LOW);
  digitalWrite(IN13, HIGH);
}
void PUTARKANAN()
{
  sp1 = kr;
  sp2 = kr;
  sp3 = kr;
  sp4 = kr;
  digitalWrite(IN1,LOW );
  digitalWrite(IN2, HIGH);
  digitalWrite(IN4, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN11, HIGH);
  digitalWrite(IN12, LOW);
  digitalWrite(IN14, HIGH);
  digitalWrite(IN13, LOW);
}
void PUTARKIRI()
{
  sp1 = kr;
  sp2 = kr;
  sp3 = kr;
  sp4 = kr;
  digitalWrite(IN1,HIGH );
  digitalWrite(IN2, LOW);
  digitalWrite(IN4, HIGH);
```

```
  digitalWrite(IN3, LOW);
  digitalWrite(IN11, LOW);
  digitalWrite(IN12, HIGH);
  digitalWrite(IN14, LOW);
  digitalWrite(IN13, HIGH);
}
void STOP()
{
  sp1 = 0;
  sp2 = 0;
  sp3 = 0;
  sp4 = 0;
  digitalWrite(IN1,LOW );
  digitalWrite(IN2, LOW);
  digitalWrite(IN4, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN11, LOW);
  digitalWrite(IN12, LOW);
  digitalWrite(IN14, LOW);
  digitalWrite(IN13, LOW);
}
////////////////////////////////Encoder Interrupt////////////////////////////////
void interruption1()
{
  counter1++;
}

void interruption2()
{
  counter2++;
}

void interruption3()
{
  counter3++;
}

void interruption4()
{
  counter4++;
}
```