IMPLEMENTASI ROBOT *THREE OMNI-DIRECTIONAL* MENGGUNAKAN KONTROLER *PROPORTIONAL INTEGRAL DERIVATIVE* (PID) PADA ROBOT KONTES ROBOT ABU INDONESIA (KRAI)

SKRIPSI

KONSENTRASI TEKNIK ELEKTRONIKA

Diajukan Untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik

DISUSUN OLEH :

RADITYA ARTHA ROCHMANTO

NIM. 0910630017-63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

IMPLEMENTASI ROBOT *THREE OMNI-DIRECTIONAL* MENGGUNAKAN KONTROLER *PROPORTIONAL INTEGRAL DERIVATIVE* (PID) PADA ROBOT KONTES ROBOT ABU INDONESIA (KRAI)

SKRIPSI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

RADITYA ARTHA ROCHMANTO
NIM. 0910630017-63

Telah diperiksa dan disetujui oleh :

Pembimbing I	Pembimbing II
M. Aziz Muslim, ST., MT., Ph.D.	**Moch. Rif'an, ST., MT.**
NIP. 19741203 200012 1 001	**NIP. 19710301 200012 1 001**

LEMBAR PENGESAHAN

IMPLEMENTASI ROBOT *THREE OMNI-DIRECTIONAL* MENGGUNAKAN KONTROLER *PROPORTIONAL INTEGRAL DERIVATIVE* (PID) PADA ROBOT KONTES ROBOT ABU INDONESIA (KRAI)

Disusun oleh:

RADITYA ARTHA ROCHMANTO
NIM. 0910630017-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 21 Januari 2014

DOSEN PENGUJI

Eka Maulana, ST.,MT.,M.Eng	**Rahmadwati,ST.,MT.,Ph.D**
NIK. 841130 06 1 1 0280	**NIP. 19771102 200604 1 001**

Goegoes Dwi Nusontoro,ST.,MT
NIP. 19711013 200604 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

M. Aziz Muslim, ST., MT., Ph.D.
NIP. 19741203 200012 1 001

PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kepada Allah SWT, karena atas segala petunjuk dan nikmat-Nya lah skripsi ini dapat diselesaikan.

Skripsi berjudul "Implementasi Robot *Three Omni-Directional* Menggunakan Kontroler PID Pada Robot KRAI" ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa dalam penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

- Allah SWT atas rahmat dan hidayah yang telah diberikan,
- Rosulullah Muhammad SAW, semoga shalawat serta salam selalu tercurah kepada beliau,
- Ayah dan Ibu atas segala nasehat, kasih sayang, perhatian dan kesabarannya didalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Adikku tersayang atas motivasi dan doanya,
- Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya serta sebagai Dosen Pembimbing I atas segala bimbingan, pengarahan, ide, saran, motivasi, dan masukan yang diberikan,
- Bapak Hadi Suyono, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Moch. Rif'an, ST., MT. selaku Ketua Prodi Strata Satu Jurusan Teknik Elektro Universitas Brawijaya serta sebagai Dosen Pembimbing II atas segala bimbingan, pengarahan, saran, dan kritik yang telah diberikan,
- Bapak Ali Mustofa, ST., MT. selaku dosen Penasehat Akademik,
- Ibu Ir. Nurussa'adah, MT. selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya,
- Seluruh dosen pengajar Teknik Elektro Universitas Brawijaya,
- Staff Recording Jurusan Teknik Elektro,
- Teman – teman Ampere angkatan 2009,

- Rekan seperjuangan dalam skripsi, Jatra, Eky, Juang, Sam, Wito, Somad, Gladi, Aka, Firda, Risma, Bona, Fanti, Lintang, Rafi, Saddam terima kasih atas segala bantuan yang telah diberikan,
- Rekan-rekan Tim Robot Kontes Robot Abu Indonesia, Mas Basori, Veri, Jakawi, Ainun, Irham dan Mirza terima kasih atas segala bantuan dan semangat yang telah diberikan,
- Seluruh Keluarga Besar Tim Robot UB Jurusan Teknik Elektro atas segala bantuan alat, bahan, dan masukan – masukan yang telah diberikan,
- Sekret KRAI, secret KRPAI, secret KRSI, dan Laboratorium Sistem Kontrol yang selama ini telah menyediakan tempat bagi penulis dalam mengerjakan skripsi ini,
- Seluruh teman-teman serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan satu-persatu, terimakasih banyak atas bantuan dan dukungannya.

Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi serta bagi masyarakat.

Malang, Januari 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Kontes Robot Abu Indonesia merupakan ajang perlombaan nasional yang diadakan setiap tahun. Salah satu tugas pada perlombaan ini adalah robot harus dapat memindahkan obyek pada titik – titik tertentu di lapangan perlombaan dengan cara yang cepat dan efisien. Tugas akhir ini merancang dan mengimplementasikan robot *three omni-directional* menggunakan kontroler PID pada robot KRAI. Kelebihan robot *three omni-directional* adalah dapat bergerak ke segala arah tanpa harus mengubah arah hadapnya. Arah gerak pada robot ini bergantung pada perbandingan kecepatan pada tiap roda yang didapat dari perhitungan kinematika robot. Kontroler PID digunakan untuk mengatur kecepatan roda sehingga didapat respon kecepatan yang cepat mencapai *set point* dan stabil. Penentuan parameter kontroler PID menggunakan metode *hand tuning*. Hasil *tuning* parameter PID pada roda 1 diperoleh nilai Kp = 3, Ki = 0,08, dan Kd = 1. Pada roda 2 diperoleh nilai Kp=3, Ki= 0,1, dan Kd=0,5 dan pada roda 3 diperoleh nilai Kp = 2,5, Ki = 0,07, dan Kd = 1. Penggunaan parameter kontroler PID yang tepat membuat robot dapat bergerak sesuai dengan arah yang diinginkan dengan rata –rata kesalahan arah gerak robot sebesar $4{,}375^0$ dan rata – rata kesalahan arah hadap robot sebesar $5{,}75^0$.

Kata kunci : robot *three omni-directional*, kontroler PID, KRAI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kontes Robot Indonesia (KRI) merupakan salah satu kompetisi robotika tingkat nasional yang diadakan secara teratur setiap tahun oleh Direktorat Jendral Pendidikan Tinggi. Kompetisi ini dibagi menjadi beberapa divisi yakni Divisi Kontes Robot Abu Indonesia (KRAI), Divisi Kontes Robot Pemadam Api Indonesia (KRPAI), Kontes Robot Seni Indonesia (KRSI) dan Kontes Robot Sepak Bola Indonesia (KRSBI). Masing-masing divisi memiliki aturan, tugas, dan arena yang berbeda. Pada Kontes Robot Abu Indonesia (KRAI) peraturan yang digunakan selalu berubah-ubah tiap tahun bergantung pada tuan rumah ABU ROBOCON diadakan. Dengan aturan yang berbeda-beda robot tetap memiliki tugas yang sama. Robot harus dapat berpindah tempat pada arena perlombaan yang cukup luas untuk memindahkan obyek-obyek pada tempat yang telah ditentukan. Oleh karena itu, dibutuhkan sistem pergerakan yang tepat agar robot dapat menyelesaikan misi dengan cepat.

Selama ini kebanyakan Robot KRAI menggunakan sistem pergerakan *differential drive. Differential drive* merupakan salah satu pergerakan robot dengan memanfaatkan kecepatan pada roda kiri dan kanannya. Permasalahan utama dari sistem pergerakan *differential drive* adalah terbatasnya pergerakan robot karena hanya memanfaatkan kecepatan pada roda kiri dan kanannya sehingga robot hanya memiliki dua derajat kebebasan, yakni bergerak maju atau mundur dan rotasi tetapi robot tidak mampu bergerak arah lain atau biasa disebut robot *non-holonomic.*

Salah satu solusi metode yang dapat digunakan untuk mengatasi keterbatasan ini adalah dengan menggunakan *three omni-directional drive* yang memiliki lebih banyak derajat kebebasan. *Three omni directional drive* merupakan salah satu pergerakan robot dengan memanfaatkan kecepatan putaran tiga roda omni sehingga robot dapat bergerak ke berbagai arah tanpa harus mengubah arah hadapnya atau biasa disebut *holonomic robot*. Penelitian mengenai robot ini pernah dilakukan oleh A. Salam Al-Amri dan Imam Ahmed

dari *University of Baghdad* yang jurnalnya telah diterbitkan pada tahun 2010. Robot pada penelitian ini hanya menggunakan motor *on off* dengan kecepatan yang tetap, sehingga sudut pergerakan robot terbatas. Oleh karena itu perlu dilakukan pengembangan agar robot dapat bergerak dengan sudut yang lebih banyak dengan cara pengontrolan kecepatan putar motor.

Kecepatan putaran roda berpengaruh sangat penting pada arah gerak robot, maka dari itu dibutuhkan suatu sistem untuk menunjang metode tersebut untuk menstabilkan kecepatan putaran roda pada robot. Dalam skripsi ini akan dibahas Implementasi Robot *Three Omni-Directional* Menggunakan Kontroler *Proportional Integral Derivative* (PID) pada Robot KRAI (Kontes Robot Abu Indonesia). Sistem ini dirancang untuk mendapatkan sistem pergerakan yang cepat dan handal sehingga *mobile robot* dapat dengan cepat menyelesaikan tugasnya.

Perancangan kontroler PID ini menggunakan metode *tuning* parameter kedua Ziegler-Nichols. Beberapa aspek yang sangat penting dalam desain *tuning* parameter kontroler PID metode kedua Ziegler-Nichols untuk pergerakan arah robot ialah penentuan masukan kontroler PID berupa kecepatan putar di masing-masing roda omni. Selanjutnya data berupa kecepatan diolah menggunakan kontroler yang menghasilkan sinyal untuk mengontrol *plant*, sehingga dapat menentukan keluaran kecepatan putaran motor. Keluaran kontroler tersebut akan melalui proses umpan balik, kesalahan ditunjukkan dengan selisih antara input (masukan) dan respon keluaran. Setelah itu baru menentukan parameter kontroler PID supaya sistem *close loop* memenuhi kriteria performansi yang diinginkan.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, dapat disusun rumusan masalah sebagai berikut :

1) Bagaimana merancang, mengimplementasikan, dan menganalisis antarmuka antara *joystick* playstation dengan mikrokontroler
2) Bagaimana merancang, mengimplementasikan, dan menganalisis sistem komunikasi SPI antara mikrokontroler *master* dengan mikrokontroler *slave.*

3) Bagaimana merancang, mengimplementasikan, dan menganilisis sistem akses sensor rotary encoder pada mikrokontroler sebagai sensor kecepatan roda.
4) Bagaimana mengimplementasikan Metode Osilasi Ziegler-Nichols pada proses *tuning* parameter kontrol PID pada robot *three omni-directional*.
5) Bagaimana merancang, mengimplementasikan, dan menganalisis sistem agar didapatkan kecepatan putaran tiap roda yang sesuai dengan hasil perhitungan kinematika robot dan robot dapat bergerak ke arah yang sesuai dengan arah tombol analog pada *joystick.*

1.3 Batasan Masalah

Dengan mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan penelitian akan diberikan batasan sebagai berikut :

1) Sensor yang dipakai adalah sensor rotary encoder untuk mengukur kecepatan putaran motor.
2) Arah pergerakan robot hanya pada sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0.
3) Robot menggunakan roda omni.
4) Menggunakan metode *tuning* parameter kedua Ziegler –Nichols dalam desain kontroler PID.
5) Menggunakan *joystick* playstation sebagai pengendali robot
6) Arena yang digunakan bukanlah arena perlombaan Kontes Robot Abu Indonesia (KRAI), robot hanya diuji untuk bergerak sesuai dengan arah yang diinginkan.

1.4 Tujuan

Tujuan penelitian ini adalah merancang dan membuat robot *three omni-directional* yang mampu bergerak ke segala arah (*holonomic*) menggunakan kontroler PID untuk diterapkan pada robot KRAI.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini sebagai berikut :

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, ruang lingkup, tujuan, dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan alat.

BAB III Metodologi

Berisi tentang metode-metode yang dipakai dalam melakukan perancangan, pengujian, dan analisis data.

BAB IV Perancangan

Perancangan dan perealisasian alat yang meliputi spesifikasi, perencanaan diagram blok, prinsip kerja dan realisasi alat.

BAB V Pengujian dan Analisis

Memuat aspek pengujian meliputi penjelasan tentang cara pengujian dan hasil pengujian. Aspek analisis meliputi penilaian atau komentar terhadap hasil-hasil pengujian. Pengujian dan analisis ini terhadap alat yang telah direalisasikan berdasarkan masing-masing blok dan sistem secara keseluruhan.

BAB VI Kesimpulan dan Saran

Memuat intisari hasil pengujian dan menjawab rumusan masalah serta memberikan rekomendasi untuk perbaikan kualitas penelitian dimasa yang akan datang.

BAB II
TINJAUAN PUSTAKA

Untuk memudahkan dalam memahami prinsip kerja maupun dasar-dasar perancangan alat atau sistem yang akan dibuat, maka perlu adanya penjelasan dan uraian mengenai teori penunjang yang digunakan dalam penelitian ini. Teori penunjang yang akan dijelaskan dalam bab ini adalah:

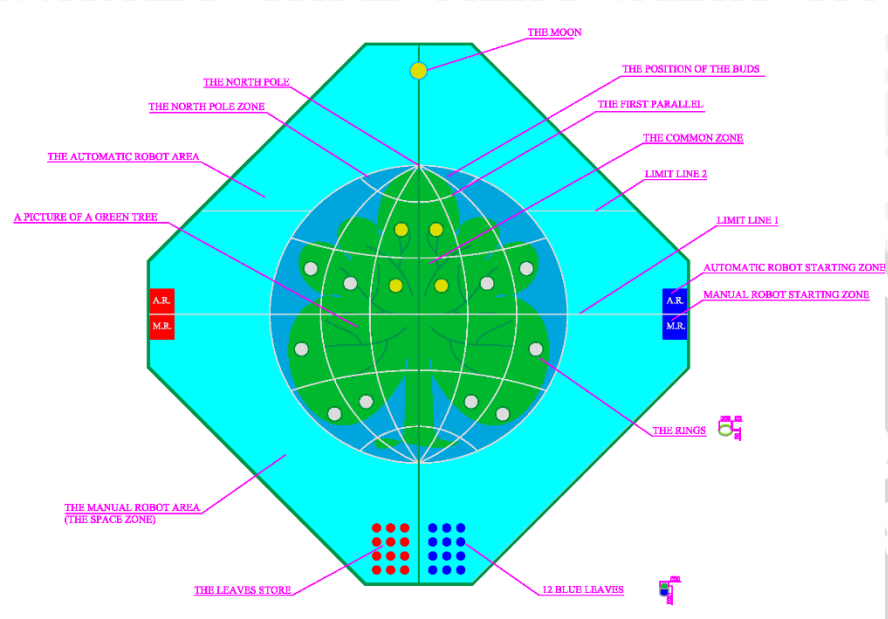
- Kontes Robot Abu Indonesia (KRAI)
- Sensor Rotary Encoder Autonics E40H8-500P/R-6-L-5
- Roda omni
- Kontroler
- Sistem pergerakan robot dengan *three omni-directional drive.*
- Mikrokontroler
- Komunikasi serial
- Komunikasi SPI
- LCD
- *Joystick* Playstation

2.1 Kontes Robot Indonesia (KRAI)

Kontes Robot Abu Indonesia (KRAI) merupakan suatu ajang kompetisi perancangan dan pembuatan robot antar mahasiswa dari seluruh Indonesia dengan aturan dan ketentuan yang berlaku menganut dari aturan yang dikeluarkan oleh ABU Robocon. Aturan yang dikeluarkan tiap tahun berbeda-beda tergantung dari unsur-unsur budaya dan tradisi dari negara yang terpilih menjadi tuan rumah ABU Robocon. Peserta dari kontes ABU Robocon tingkat Internasional adalah tim robot yang telah menjadi juara dari negaranya masing-masing.

Bentuk dan ukuran dari lapangan Kontes Robot Indonesia sendiri juga berbeda tiap tahunnya. Sebagai contoh, ketika Malaysia menjadi tuan rumah ABU Robocon 2007 maka tema yang diambil adalah bangunan yang menjadi simbol Malaysia yaitu Menara Petronas. Dengan demikian semua peserta KRAI diharuskan merancang beberapa robot yang dapat bekerjasama menyusun obyek-obyek perlombaan sehingga membentuk miniatur dari Menara Petronas. Pada tahun 2013 ini Negara yang ditunjuk menjadi tuan rumah adalah Vietnam.

Dengan mengambil tema *The Green Planet* dengan pesan: "Setiap Negara adalah bagian dari dunia, dan tanggung jawab untuk melindungi bumi berada di pundak masing-masing orang yang tinggal di dalamnya". Gambar 2.1 menunjukkan lapangan perlombaan KRAI 2013 jika dilihat dari atas.



Gambar 2.1 Lapangan ABU Robocon 2013
Sumber: Rulebook ABU Robocon 2013, 2013:17

Pada rule tahun 2013 ini robot harus dapat meletakkan *leaf* ke ring. Letak ring yang menyebar membuat robot harus berpindah-pindah tempat. Oleh karena itu dibutuhkan sistem pergerakan yang lincah dan efisien sehingga robot dapat mencapai titik tujuan dengan cepat dan tepat. Selain sistem pergerakan dibutuhkan juga sistem kontrol yang baik agar robot lebih stabil dalam bergerak.

2.2 Sensor Rotary Encoder Autonics E40H8-500P/R-6-L-5

Sensor Rotary encoder yang digunakan pada skripsi ini merupakan produksi dari autonics dengan tipe E40H8-500P/R-6-L-5 seperti dalam Gambar 2.2. Sensor ini memiliki lubang dengan diameter 80 mm sebagai tempat untuk mengkopel antara sensor dengan roda. Keluaran dari sensor ini berupa pulsa dengan tegangan 5V ±5% saat pulsa *high* dan 0V saat pulsa *low*. Sensor ini dapat menyediakan 500 pulsa/ rotasi. Pulsa keluaran dari sensor ini dapat diartikan menjadi gerakan, posisi, dan arah sehingga posisi sudut suatu poros benda berputar dapat diolah menjadi informasi berupa kode digital oleh rotary encoder untuk diteruskan oleh rangkaian kendali.

Gambar 2.2 Rotary Encoder
Sumber : autonics, 2013

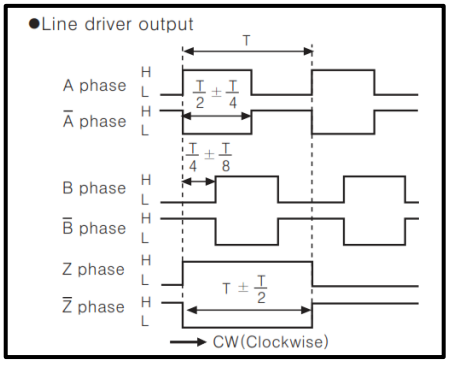
Sensor ini mempunyai 9 kaki atau pin. Konfigurasi dari masing-masing pin ditunjukkan dalam Tabel 2.1.

Tabel 2.1 Konfigurasi Pin Rotary Encoder

No Pin	**Function**	**Warna Kabel**
1	OUT A	hitam
2	OUT A’	merah
3	+V	coklat
4	GND	biru
5	OUT B	Putih
6	OUT B’	Abu – abu
7	OUT Z	jingga
8	OUT Z’	kuning
9	F.G	Pelindung kabel

Sumber : Datasheet Autonics Rotary Encoder

Perbedaan antara masing-masing keluaran dari sensor rotary encoder ini dapat kita lihat dalam Gambar 2.3.



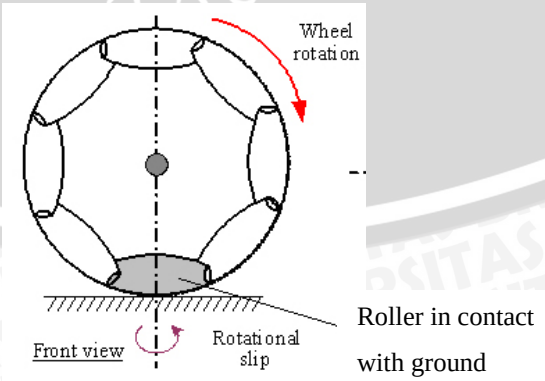
Gambar 2.3 *Line Driver Output* Sensor Rotary Encoder
Sumber : Dataheet autonics rotary encoder

Dalam Gambar 2.3 diketahui bahwa terdapat pergeseran fasa antara OUT A dan OUT B dengan periode pulsa yang sama, sedangkan OUT Z memiliki periode 2 kali lebih lama. Fasa yang berkebalikan juga disediakan oleh rotary ini pada OUT A', B' dan Z'.

2.3 Roda Omni

Roda Omni adalah suatu roda unik karena memiliki tiga derajat kebebasan dalam bergerak. Roda ini berputar seperti roda pada umumnya serta mampu bergeser ke samping menggunakan roda di sepanjang lingkar luar roda. Roda *Omni-directional* memungkinkan robot untuk mengkonversi dari robot *non- holonomic* menjadi robot *holonomic*. Sebuah robot *non-holonomic* yang menggunakan roda normal hanya memiliki dua DOF *(Degree of Freedom)* yang terkendali, yaitu bergerak maju atau mundur dan rotasi. Robot *non-holonomic* tidak memiliki kemampuan untuk bergerak ke samping kiri atau kanan sehingga membuat robot lebih lambat dan kurang efisien dalam mencapai tujuan yang diberikan.

Roda *omni-directional holonomic* mampu mengatasi masalah ini karena roda omni memiliki tiga DOF. Berbeda dengan robot *non-holonomic* normal, robot *omni-directional holonomic* mampu bergerak ke segala arah tanpa mengubah arah roda. Roda *omni-directional holonomic* dapat bergerak maju mundur, geser ke samping, dan berputar pada posisi tetap. Kemampuan ini memungkinkan robot yang menggunakan roda *omni-directional* mampu bermanuver untuk lebih lincah dan lebih efisien. Salah satu bentuk roda *omni-directional* dapat dilihat dalam Gambar 2.4.



Gambar 2.4 Derajat Kebebasan Pada Roda Omni
Sumber : Jae Bok-Song, 2006

2.4 Kontroler

Sistem pengendalian dirancang untuk melakukan dan menyelesaikan tugas tertentu. Syarat utama sistem pengendalian adalah harus stabil. Di samping kestabilan mutlak, maka sistem harus memiliki kestabilan secara relatif, yakni tolak ukur kualitas kestabilan sistem dengan menganalisis sampai sejauh mana batas-batas kestabilan sistem tersebut jika dikenai gangguan (Ogata K.,1997). Selain itu analisis juga dilakukan untuk mengetahui bagaimana kecepatan sistem dalam merespons input, dan bagaimana peredaman terhadap adanya lonjakan *(over shoot).*

Suatu sistem dikatakan stabil jika diberi gangguan maka sistem tersebut akan kembali ke keadaan *steady state* yaitu output berada dalam keadaan tetap seperti tidak ada gangguan. Sistem dikatakan tidak stabil jika outputnya berosilasi terus menerus ketika dikenai suatu gangguan. Suatu sistem pengendalian biasanya melibatkan penyimpanan energi maka output sistem ketika diberi suatu input, tidak dapat mengikuti input secara serentak, tetapi menunjukkan respons transien berupa suatu osilasi teredam sebelum mencapai *steady state*. Dalam sistem pengendalian terdapat dua macam *loop*:

1) Pengendalian dengan *loop* terbuka

Sistem kontrol *loop* terbuka adalah sistem kontrol yang keluarannya tidak berpengaruh pada aksi pengontrolan. Jadi pada sistem kontrol *loop* terbuka, keluaran tidak diukur atau diumpan balikan untuk dibandingkan dengan masukan.

2) Pengendalian dengan *loop* tertutup

Sistem kontrol *loop* tertutup adalah sistem kontrol yang keluarannya mempunyai pengaruh langsung pada aksi pengontrolan. Disebut juga sistem kontrol yang menggunakan umpan balik untuk memperkecil kesalahan sistem.

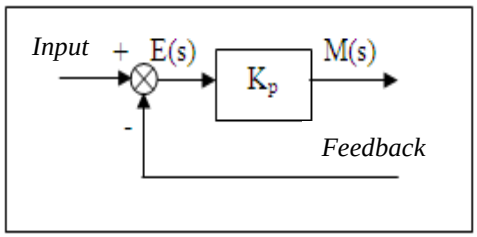
2.4.1 Kontroler Proporsional

Untuk kontroler dengan aksi kontrol proporsional, hubungan antara keluaran kontroler *m(t)* dan sinyal kesalahan penggerak *e(t)* adalah:

$$m(t)=K_p \cdot e(t) \quad \text{.....(2-1)}$$

dengan K_p adalah kepekaan proporsional atau penguatan.

Apapun wujud mekanisme yang sebenarnya dan apapun bentuk daya penggeraknya, kontroler proporsional pada dasarnya merupakan penguat dengan penguatan yang dapat diatur (Ogata K.,1997). Diagram blok kontroler proporsional ditunjukkan dalam Gambar 2.5.



Gambar 2.5 Diagram Blok Kontroler Proposional
Sumber: Ogata K., 1997

2.4.2 Kontroler Integral

Pada kontroler dengan aksi integral, harga keluaran kontroler m(t) diubah dengan laju yang sebanding dengan sinyal kesalahan penggerak *e (t)*, dapat ditulis dalam bentuk :

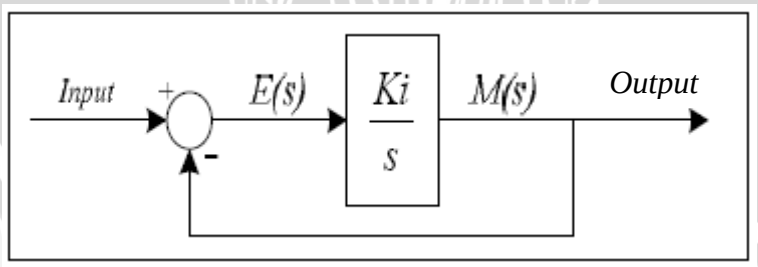
$$m(t) = K_i \int e(t)dt \quad \text{..... (2-2)}$$

rumus *kontroler* integral ini dapat dinyatakan :

$$m(t) = (Ki.Ts) \cdot \sum e(t) \quad \text{..... (2-3)}$$

dengan $\sum e(t) = \sum e(t-1) + e(t)$ ..... (2-4)

Jika harga *e(t)* dikalikan dua, maka harga m(t) berubah dengan laju perubahan menjadi dua kali semula. Jika kesalahan penggerak nol, maka harga m(t) tetap stasioner. Aksi kontrol integral seringkali disebut kontrol *reset* (Ogata K.,1997). Gambar 2.6 menunjukkan diagram blok kontroler integral.



Gambar 2.6 Diagram Blok Kontroler Integral
Sumber: Ogata K., 1997

2.4.3 Kontroler Differensial

Kontroler ini digunakan untuk memperbaiki atau mempercepat respons transien sebuah sistem kontrol dengan cara memperbesar *phase lead* terhadap

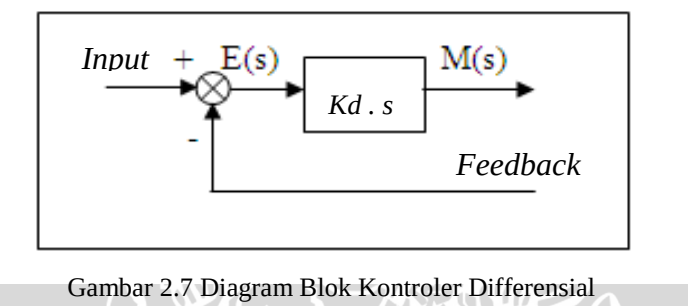
penguatan kontrol dan mengurangi *phase lag* penguatan tersebut (Ogata K.,1997). Kontroler differensial tidak dapat mengeluarkan output bila tidak ada perubahan input, selain itu kontroler differensial tidak dapat digunakan untuk proses yang mengandung *noise*. Hubungan antara keluaran kontroler *m(t)* dan sinyal kesalahan penggerak *e(t)* adalah :

$$m(t) = K_d \frac{d}{dt} e(t) \quad \text{..... (2-5)}$$

rumus kontroler diferensial ini dapat dinyatakan dengan :

$$m(t)=Kd/Ti.(e(t-1)-e(t)) \quad \text{..... (2-6)}$$

Gambar 2.7 menunjukkan diagram blok kontroler differensial.



Gambar 2.7 Diagram Blok Kontroler Differensial
Sumber: Ogata K., 1997

2.4.4 Kontroler Proporsional Integral Differensial (PID)

Gabungan aksi kontrol proporsional, integral, dan differensial mempunyai keunggulan dibandingkan dengan masing-masing dari tiga aksi kontrol tersebut. Persamaan kontroler PID ini dapat dinyatakan sebagai berikut :

$$m(t) = Kp\,.e(t) + \frac{Kp}{Ti}.\int_0^t e(t)dt + Kp.Td\frac{de(t)}{dt} \quad \text{..... (2-7)}$$

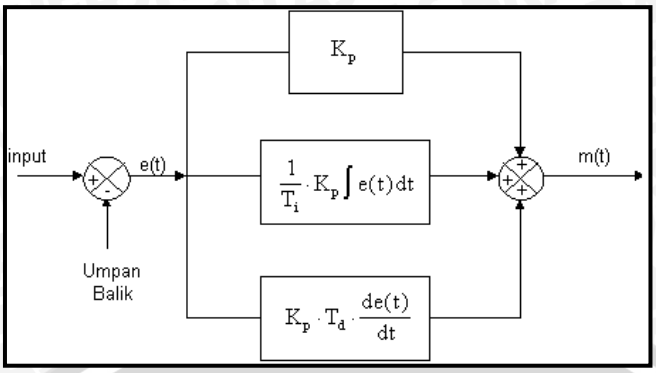
dan dapat didekati dengan persamaan sebagai berikut :

$$m(t)=Kp.error(t)+(Ki.Ts).\sum error(t)+(Kd/Ts).(error(t-1)-error(t)) \quad \text{.... (2-8)}$$

dengan

$$\sum e(t) = \sum e(t-1) + e(t) \quad \text{..... (2-9)}$$

Gambar 2.8 menunjukkan diagram blok kontroler PID

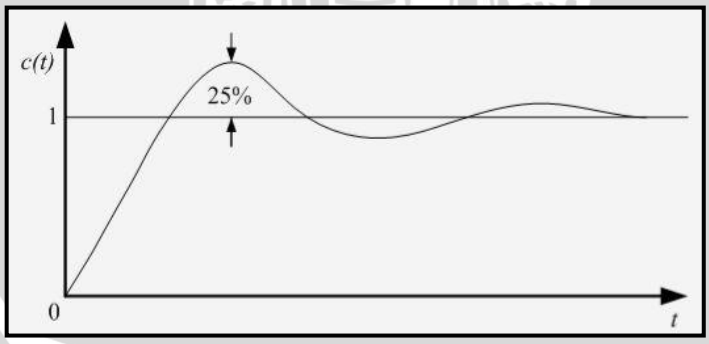


Gambar 2.8 Diagram Blok Kontroler PID
Sumber: Ogata K., 1997

2.4.5 Metode Perancangan Kontroler Proporsional Integral Diferensial (PID) Menggunakan Metode Ziegler-Nichols.

Ziegler dan Nichols mengemukakan aturan-aturan untuk menentukan nilai dari gain proporsional Kp, waktu integral Ti, dan waktu derivatif Td berdasarkan karakteristik respon transien dari *plant* yang diberikan. Penentuan parameter kontroler PID atau penalaan kontroler PID tersebut dapat dilakukan dengan bereksperimen dengan plan.(Ogata, K., 1997)

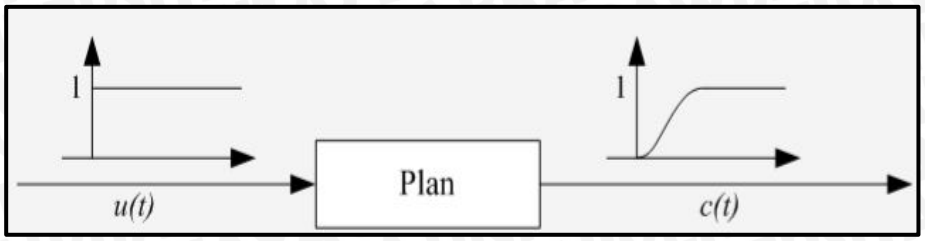
Terdapat dua metode yang disebut dengan aturan penalaan Ziegler-Nichols, pada kedua metode tersebut memiliki tujuan yang sama yaitu untuk mencapai 25% *maximum overshoot* pada respon unit step, seperti ditunjukkan dalam Gambar 2.9.



Gambar 2.9 Kurva Respon Unit Step yang Menunjukkan 25% *Maximum Overshoot*
Sumber: Ogata, K., 1997

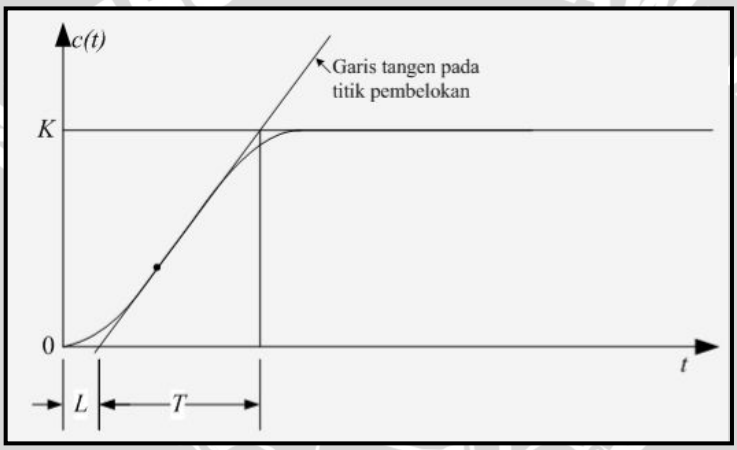
a) Metode Pertama

Metode pertama atau sering disebut metode kurva reaksi, respon dari plan dapat dapat diperoleh secara eksperimental dengan masukan berupa unit step, seperti yang ditunjukkan dalam Gambar 2.10.



Gambar 2.10 Respon Plant Terhadap Masukan Berupa Unit Step
Sumber: Ogata, K. 1997

Jika dalam plan tersebut terdapat integrator atau *dominan complex-conjugate poles*, maka kurva respon unit step berbentuk seperti huruf S, seperti dalam Gambar 2.11, jika respon tidak memberikan bentuk kurva S, maka metode ini tidak berlaku.(Ogata, K., 1997).



Gambar 2.11 Kurva Respon yang Berbentuk S
Sumber: Ogata, K. 1997

Kurva berbentuk S tersebut dapat dikarakteristikkan menjadi dua konstanta yaitu waktu tunda *L* dan konstanta waktu *T*. Waktu tunda dan konstanta waktu ditentukan dengan menggambar sebuah garis tangen pada titik pembelokan dari kurva S, dan menentukan perpotongan antara garis tangen dengan sumbu waktu *t* dan sumbu *c(t) =K*, seperti yang telah ditunjukkan dalam Gambar 2.11.

Fungsi alih *C(s)/U(s)* dapat dilakukan pendekatan dengan sistem orde satu dengan persamaan sebagai berikut:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \quad \text{(2-10)}$$

Ziegler dan Nichols menyarankan untuk menentukan nilai-nilai dari Kp, Ti dan Td berdasarkan pada formula yang ditunjukkan dalam Tabel 2.2 (Ogata, K., 1997).

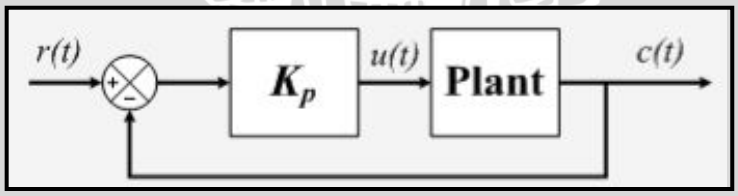
Tabel 2.2 Aturan Penalaan Ziegler-Nichols Berdasarkan Respon Unit Step Dari Plan

Tipe Kontroler	K_p	T_i	Td
P	$\frac{T}{L}$	∞	0
PI	$0{,}9\frac{T}{L}$	$\frac{L}{0{,}3}$	0
PID	$1{,}2\ \frac{T}{L}$	$2L$	$0{,}5\ L$

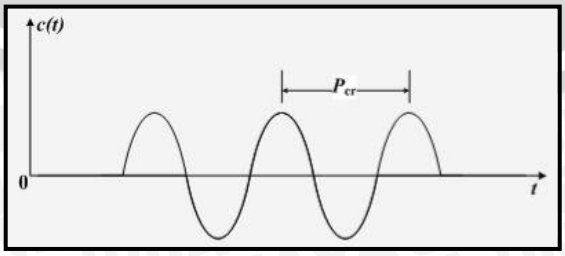
Sumber: Ogata, K. 1997

b) Metode Kedua

Dalam metode kedua ziegler-nichols, mula-mula yang dilakukan adalah membuat Ti = ∞ dan Td = 0. Kemudian hanya dengan menggunakan tindakan kontrol proporsional, harga ditingkatkan dari nol ke suatu nilai kritis *K*cr, disini mula-mula keluaran memiliki osilasi yang berkesinambungan (Jika keluaran tidak memiliki osilasi berkesinambungan untuk nilai Kp manapun yang telah diambil, maka metode ini tidak berlaku). Dari keluaran yang berosilasi secara berkesinambungan, penguatan kritis *K*cr dan periode *P*cr dapat ditentukan. Diagram blok sistem loop tertutup dengan kontroler proporsional dapat dilihat dalam Gambar 2.12 dan untuk osilasi berkesinambungan dengan periode Pcr dapat dilihat dalam Gambar 2.13. Ziegler dan Nichols menyarankan penyetelan nilai parameter Kp,Ti ,Td dan berdasarkan rumus yang diperlihatkan dalam Tabel 2.3. (Ogata, K., 1997).



Gambar 2.12 Sistem Loop Tertutup dengan Kontroler Proporsional
Sumber: Ogata, K., 1997



Gambar 2.13. Osilasi Berkesinambungan dengan Periode *P*cr
Sumber : Ogata, K., 1997

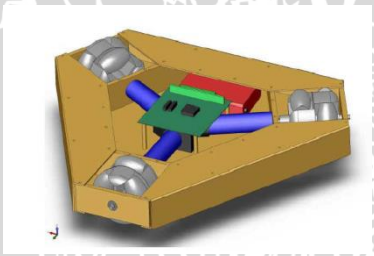
Tabel 2.3 Aturan Dasar Ziegler-Nichols Berdasarkan Critical Gain *Kcr* dan Critical Period *Pcr*

Tipe Kontroler	K_p	T_i	Td
P	0.5 Kcr	∞	0
PI	0.45 Kcr	$\frac{1}{1,2}$ Pcr	0
PID	0.60 Kcr	0.5 Pcr	0.125 Pcr

Sumber: Ogata, K., 1997

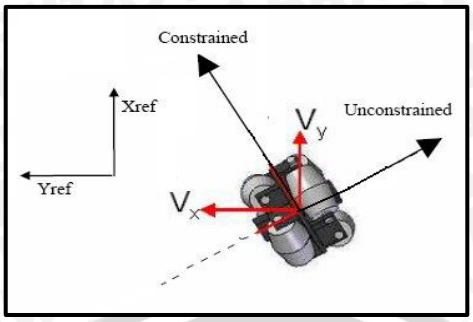
2.5 Sistem Pergerakan Robot Three Omni-directional Drive

Selama ini yang sistem pergerakan yang banyak digunakan adalah sistem pergerakan *differensial drive.* Sistem ini menggunakan dua motor tetap yang diletakkan pada dua roda di belakang, tetapi sistem ini memiliki kekurangan yaitu tidak dapat bergerak ke segala arah yang mungkin atau biasa disebut *non-holonomic.* Oleh karena itu dikembangkan suatu sistem baru yang dapat mendukung pergerakan robot untuk bergerak ke segala arah atau biasa disebut *holonomic robot.* Salah satu yang sedang berkembang pesat adalah robot dengan *three omni-directional drive.* Robot ini berbentuk segitiga sama sisi dengan roda omni diletakkan pada tiap ujungnya. Gambar skematik robot dapat dilihat dalam Gambar 2.14.



Gambar 2.14 Bentuk Skematik Robot *Three Omni-directional*
Sumber : T.A. Baede, 2006

Untuk mengetahui Sistem dari *three omni-directional drive* secara keseluruhan, kinematika dari setiap sistem pergerakan harus dipelajari terlebih dahulu. Gambar 2.15 mengilustrasikan vektor pergerakan yang bebas maupun yang tidak bebas pada pemasangan tiap roda. Pergerakan tidak bebas pada roda dikontrol secara langsung oleh kontrol motor.



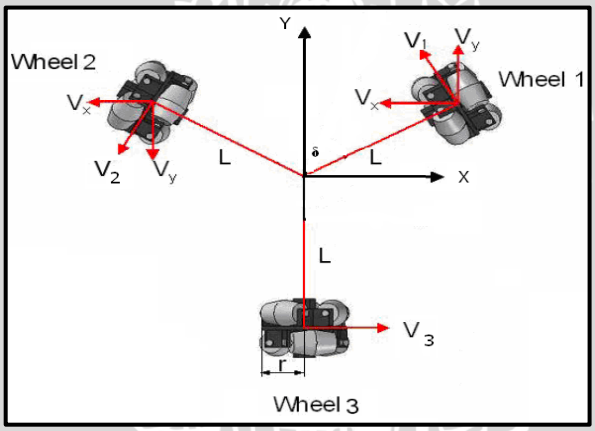
Gambar 2.15 Kinematika Pada Satu Roda
Sumber : Al-Ammri, Salam dan Iman Ahmed, 2010

Melalui Gambar 2.16 persamaan kinematik dari sistem pergerakan ini dapat diperoleh. Persamaan yang digunakan pada sistem kontrol robot adalah:

$$V_x = V_3 - V_1 Cos(\delta) - V_2 Cos(\delta) \quad \text{..... (2-11)}$$

$$V_y = V_1 Sin(\delta) - V_2 Sin(\delta) \quad \text{.....(2-12)}$$

$$V_\emptyset = V_1/L + V_2/L + V_3/L \quad \text{.....(2-13)}$$



Gambar 2.16 Representasi Kinematika dari Sistem Pergerakan *Three Omni-directional*
Sumber : Salam Al-Ammri dan Iman Ahmed, 2010

Setiap roda disusun secara simetris dengan perbedaan sudut tiap roda (120^0) dengan δ = (60^0). Jadi persamaan 2-11, 2-12, dan 2-13 dapat ditulis dengan suatu bentuk matrik :

$$\begin{bmatrix} V_x \\ V_y \\ V_\emptyset \end{bmatrix} = \begin{bmatrix} -cos\delta & -cos\delta & 1 \\ sin\delta & -sin\delta & 0 \\ 1/L & 1/L & 1/L \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \quad \text{.....(2-14)}$$

2.6 Mikrokontroler

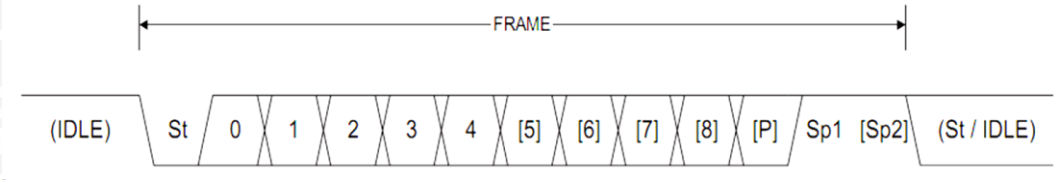
Secara umum, mikrokontroler berfungsi sama dengan komputer. Bedanya adalah mikrokontroler memiliki desain dalam sebuah *single chip* (IC). Mikrokontroler terdapat di hampir semua peralatan elektronik di sekeliling kita, di

dalam tape, TV, radio, telepon genggam (*Hand Phone*). Mikrokontroler memiliki kemampuan yang diperlukan untuk membuat keputusan berdasarkan sinyal dari luar dengan kata lain mikrokontroler merupakan otak dari sebuah perangkat elektronik. Seperti umumnya komputer, mikrokontroler adalah alat yang mengerjakan instruksi-instruksi yang diberikan kepadanya. Artinya, bagian terpenting dan utama dari suatu sistem terkomputerisasi adalah program itu sendiri yang dibuat oleh seorang programmer. Program ini menginstruksikan komputer untuk melakukan jalinan yang panjang dari aksi-aksi sederhana untuk melakukan tugas yang lebih kompleks yang diinginkan oleh programmer.

2.7 Komunikasi Serial

Pada prinsipnya, komunikasi serial ialah komunikasi dengan pengiriman data yang dilakukan per bit, sehingga lebih lambat dibandingkan komunikasi pararel seperti pada port printer yang mampu mengirim 8 bit sekaligus dalam sekali waktu. Devais pada komunikasi serial port dibagi menjadi 2 (dua) kelompok yaitu *Data Communication Equipment* (DCE) dan *Data Terminal Equipment* (DTE). Pengiriman data secara serial dapat berupa sinkron, yaitu pengiriman clock dilakukan bersamaan dengan data, atau berupa asinkron, yaitu pengiriman clock tidak bersamaan dengan data namun secara dua tahap, saat data dikirim dan data diterima. Untuk istilah yang sering digunakan untuk mengirim dan menerima data secara asinkron biasa disebut *Universal Asynchronous Receiver Transmitter* (UART). Komunikasi data serial menggunakan UART sangat umum dan mudah penggunaannya, misalnya pada port serial PC. Pada UART jalur pengiriman dan penerimaan data serial dipisahkan.

Setiap pengiriman data pada UART menggunakan bit tanda *start* bit dan *stop* bit. Jalur data yang digunakan hanya satu untuk setiap pengiriman data. Data-data serial dikirim melewati jalur data satu persatu setiap satuan waktu. Format pengiriman data serial secara asinkron ditunjukkan dalam Gambar 2.17.



Gambar 2.17 Format frame data serial USART
Sumber: Atmel,2007:137

Dengan :

St = *Bit start* selalu berlogika rendah

(n) = Banyaknya data yang dikirim (0-8)

P = *Bit* paritas (ganjil atau genap)

Sp = *Bit stop* selalu berlogika tinggi (*bit* stop bisa berjumlah 1 atau 2

IDLE = Tidak ada data yang ditransfer pada RX dan TX, IDLE selalu berlogika tinggi

2.8 Komunikasi Serial Peripheral Interface (SPI)

Serial Peripheral Interface (SPI) merupakan salah satu mode komunikasi serial *synchronous* kecepatan tinggi yang dimiliki oleh ATMega. Komunikasi SPI membutuhkan 3 jalur (*wire*) yaitu MOSI, MISO dan SCK. Melalui komunikasi SPI ini data dapat saling dikirimkan baik antar mikrokontroler maupun antar mikrokontroler dengan peripheral lain di luar mikrokontroler.

- MOSI : *Master Output Slave Input*

  Artinya jika dikonfigurasi sebagai *master* maka pin MOSI ini sebagai output tetapi jika dikonfigurasi sebagai *slave* maka pin MOSI ini sebagai input.

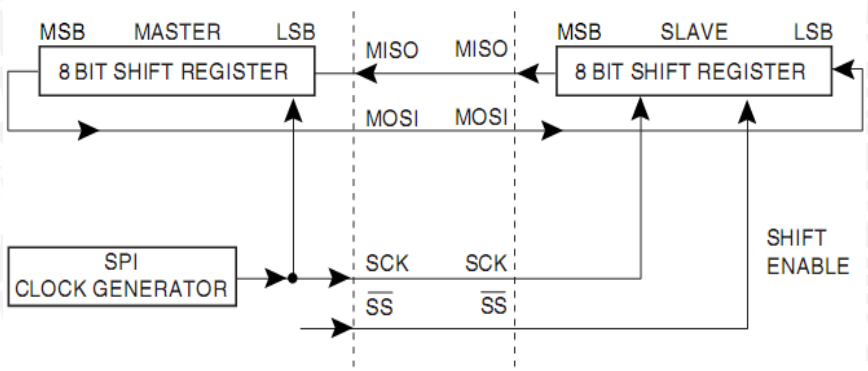
- MISO : *Master Input Slave Output*

  Berkebalikan dengan MOSI di atas, jika dikonfigurasi sebagai *master* maka pin MISO ini sebagai input tetapi dikonfigurasi sebagai *slave* maka pin MISO ini sebagai output.

- CLK : *Clock*

  Jika dikonfigurasi sebagai *master* maka pin CLK berlaku sebagai output (pembangkit *clock*) tetapi jika dikonfigurasi sebagai *slave* maka pin CLK berlaku sebagai input (menerima sumber *clock* dari *master*).

  Gambar 2.18 menunjukkan interkoneksi SPI antara *master* dan *slave*.



Gambar 2.18 Interkoneksi SPI *Master-Slave*
Sumber: Atmel, 2006:133

Pengaturan konfigurasi *master* atau *slave* ditentukan oleh pin SS. Jika pin SS diberi tegangan *high* (‘1’) maka terkonfigurasi sebagai *master* dan jika pin SS diberi tegangan *low* (‘0’) maka terkonfigurasi sebagai *slave*. Untuk mengatur mode kerja komunikasi SPI ini dilakukan dengan menggunakan *register* SPCR, SPSR dan SPDR.

2.9 LCD (Liquid Crystal Display)

LMB162A adalah modul LCD *(liquid crystal display)* matrix dengan konfigurasi 16 karakter dan 2 baris dengan setiap karakternya ditunjukkan dalam Gambar 2.19.

Gambar 2.19 LCD (Liquid Crystal Display) LM162A
Sumber: Baskara, 2013

Liquid Crystal Display atau biasa disebut LCD merupakan alat tampilan yang biasa digunakan untuk menampilkan karakter ASCII sederhana, dan gambar-gambar pada alat-alat digital seperti jam tangan, kalkulator dan lain-lain. Deskripsi sederhana cara kerja dari sebuah LCD matrix adalah sebuah *Twisted Nematic* (TN) *Liquid Crystal Display*, yang terdiri dari 2 material yang terpolarisasi, 2 buah kaca, sebuah bentuk elemen elektroda untuk menentukan pixel, dan Integrated Circuit (IC) untuk mengalamatkan baris dan kolom.
Perlu diketahui fungsi dari setiap kaki yang ada pada komponen tersebut.

1) Kaki 1 (GND) : Kaki ini berhubungan dengan tegangan 0 volt (Ground).

2) Kaki 2 (VCC) : Kaki ini berhubungan dengan tegangan +5 Volt yang merupakan tegangan untuk sumber daya.

3) Kaki 3 (VEE/VLCD) : Tegangan pengatur kontras LCD, kaki ini terhubung pada cermet. Kontras mencapai nilai maksimum pada saat kondisi kaki ini pada tegangan 0 volt.

4) Kaki 4 (RS) : Register Select, kaki pemilih register yang akan diakses. Untuk akses ke Register Data, logika dari kaki ini adalah 1 dan untuk akses ke Register Perintah, logika dari kaki ini adalah 0.

5) Kaki 5 (R/W) : Logika 1 pada kaki ini menunjukkan bahwa modul LCD sedang pada mode pembacaan dan logika 0 menunjukkan bahwa modul LCD sedang pada mode penulisan. Untuk aplikasi yang tidak memerlukan pembacaan data pada modul LCD, kaki ini dapat dihubungkan langsung ke Ground.

6) Kaki 6 (E) : Enable Clock LCD, kaki mengaktifkan clock LCD. Logika 1 pada kaki ini diberikan pada saat penulisan atau pembacaan data.

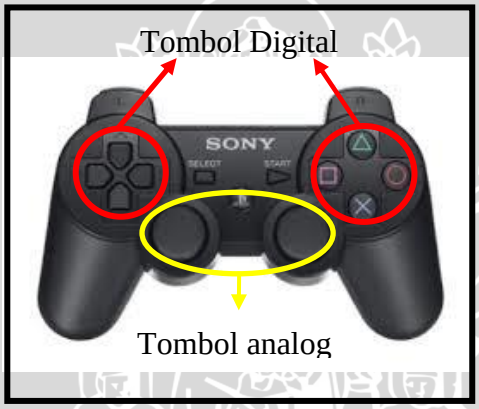
7) Kaki 7 – 14 (D0 – D7) : Data bus, kedelapan kaki LCD ini adalah tempat aliran data sebanyak 4 bit ataupun 8 bit mengalir saat proses penulisan maupun pembacaan data.

8) Kaki 15 (Anoda) : Berfungsi untuk tegangan positif dari *backlight* LCD sekitar 4,5 volt (hanya terdapat untuk LCD yang memiliki back light).

9) Kaki 16 (Katoda) : Tegangan negatif back light LCD sebesar 0 volt (hanya terdapat pada LCD yang memiliki *backlight*).

2.10 *Joystick* Playstation

Joystick playstation merupakan kontroler dari *game console* playstation, baik playstation 1,2 maupun 3. Kontroler ini terdiri atas 16 tombol digital dan 2 tombol analog. Tombol analog ini terdiri atas dua buah variable resistor yang dipasang membentuk tanda plus (+). Satu *variable* resistor diletakkan secara vertikal sebagai penunjuk sumbu Y dan *variable* resistor yang lain diletakkan secara horizontal sebagai penunjuk sumbu X. Desain tombol analog yang seperti ini membuat tombol analog dapat menunjuk ke banyak arah. Jumlah tombol yang cukup banyak dan adanya jalur analog ini membuat *joystick* dapat digunakan sebagai sistem kendali robot. Komunikasi antara joystick dengan mikrokontroler menggunakan komunikasi SPI. Gambar 2.20 menunjukkan gambar *joystick* playstation.



Gambar 2.20 *Joystick* Playstation
Sumber: Jon William, 2006

BAB III
METODOLOGI PENELITIAN

Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan penelitian yang terdapat di bab pendahuluan maka diperlukan metode untuk menyelesaikan masalah tersebut.

3.1 Studi Literatur

Studi literatur dilakukan untuk mempelajari teori penunjang sistem yang dibutuhkan dalam perencanaan dan pembuatan alat. Teori yang diperlukan antara lain berkaitan dengan robot dengan *three omni-directional drive*, metode *tuning* kontrol PID, sensor rotary encoder, komunikasi serial, komunikasi *serial peripheral interface*, mikrokontroler, dan *joystick* playstation.

3.2 Penentuan Spesifikasi Alat

Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan adalah sebagai berikut:

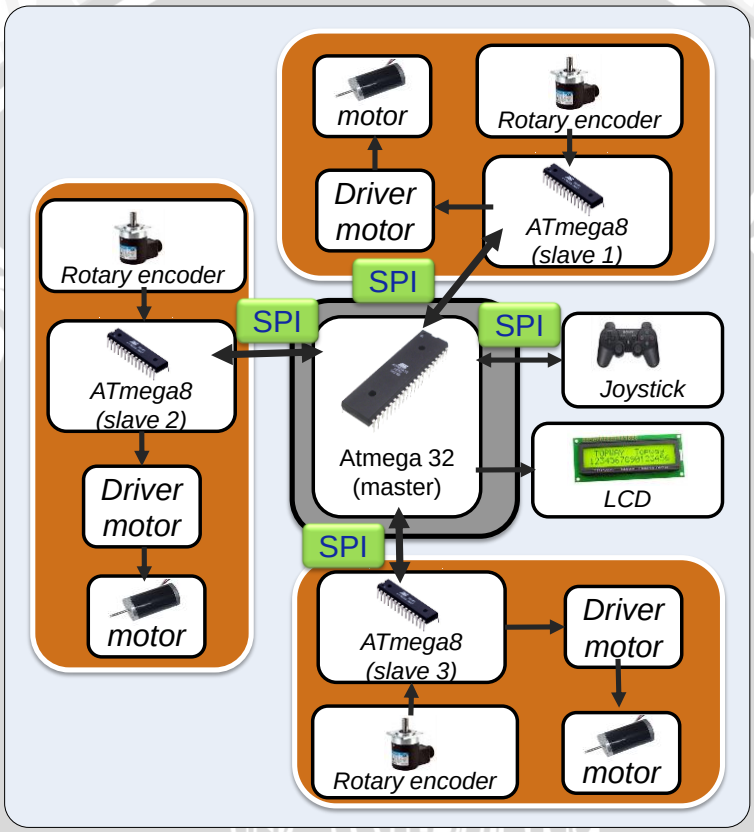
1) Robot berbahan dasar alumunium kotak.
2) Menggunakan tiga buah motor DC magnet permanen sebagai aktuator.
3) Menggunakan tiga roda omni dengan diameter 125 mm dan tebal 1 cm.
4) Pendeteksi kecepatan putaran roda dengan sensor rotary encoder.
5) Sebagai pengendali arah gerak robot digunakan *joystick* playstation
6) LCD berfungsi sebagai tampilan sudut tombol analog yang dikirim oleh *joystick* playstation
7) Menggunakan Mikrokontroler ATMega32 sebagai pengendali utama *mobile robot*.
8) Menggunakan Mikrokontroler ATMega8 sebagai *slave* kontroler PID pada tiap motor.

3.3 Perancangan dan Pembuatan Alat

Perancangan dan pembuatan alat dalam penelitian ini dibagi menjadi perancangan *hardware*, konversi data joystick menjadi sudut, kinematika robot sistem kontrol dan *software*.

3.3.1 Perancangan dan Pembuatan Perangkat Keras (*Hardware*) dan

Secara garis besar, diagram blok perancangan perangkat keras (*hardware*) sistem secara keseluruhan ditunjukkan dalam Gambar 3.1.



Gambar 3.1 Diagram Blok Perancangan Perancangan Keras (*hardware*) Sistem Secara Keseluruhan

Fungsi masing-masing bagian dalam diagram blok ini adalah sebagai berikut:

1) *Joystick* playstation berfungsi untuk memasukkan input sudut arah robot bergerak.
2) Rotary encoder berfungsi sebagai sensor kecepatan roda untuk kemudian dijadikan masukan mikro*kontroler slave.*
3) Perancangan ini menggunakan Mikro*kontroler* ATmega32 sebagai *master* yang berfungsi untuk mengakses input dari tombol analog. Kemudian data

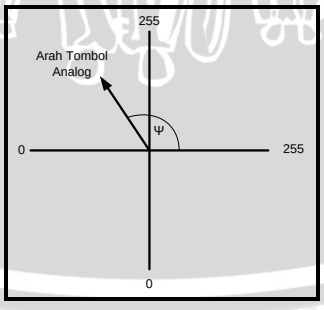
input diolah sehingga menghasilkan output kecepatan masing-masing roda yang dikirim ke masing-masing mikrokontroler *slave*.

4) Mikrokontroler *slave* yang digunakan adalah ATmega8 yang berfungsi untuk mengolah rumus PID. Pada perancangan ini digunakan tiga buah mikrokontroler ATmega8 yang masing-masing mengontrol satu motor. Dengan pembagian tugas ini, diharapkan putaran robot akan lebih stabil karena penghitungan rumus PID akan menjadi lebih cepat bila dibandingkan dengan penggunaan satu mikrokontroler untuk tiga motor.
5) *Driver* motor DC digunakan sebagai antarmuka antara mikrokontroler dengan motor DC.

Prinsip kerja sistem ini adalah awalnya mikrokontroler *master* akan mengolah input dari tombol analog *joystick* playstation untuk menentukan arah gerak robot. Sudut arah gerak robot ini menjadi masukan rumus kinematika robot. Hasil dari rumus ini berupa kecepatan untuk masing – masing roda. Mikrokontroler *master* akan mengirimkan data berupa kecepatan roda ke masing-masing mikrokontroler *slave*. Mikrokontroler *slave* akan mengontorol kecepatan putaran tiap roda dengan *feedback* dari rotary encoder.

3.3.2 Konversi Data *Joystick* Menjadi Sudut Arah Gerak Robot

Data dari *joystick* diterima mikrokontroler dalam bentuk nilai ADC untuk sumbu-X dan sumbu-Y sehingga data ini harus diolah terlebih dahulu agar didapat arah gerak tombol analog. Gambar 3.2 menunjukkan konversi nilai ADC menjadi sudut arah tombol analog.



Gambar 3.2 Konversi Data Nilai ADC menjadi sudut arah tombol analog

Tombol analog digunakan untuk menunjukkan arah 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0. Sudut yang ditunjukkan oleh tombol analog menunjukkan arah robot bergerak.

3.3.3 Kinematika Robot Three Omni Directional

Persamaan kinematika robot telah dijelaskan pada bab 2 (persamaan 2-14) sebagai berikut

$$\begin{bmatrix} V_x \\ V_y \\ V_\emptyset \end{bmatrix} = \begin{bmatrix} -cos\delta & -cos\delta & 1 \\ sin\delta & -sin\delta & 0 \\ 1/L & 1/L & 1/L \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

Pada robot *three omni-directional* setiap roda disusun secara simetris dengan perbedaan sudut tiap roda (120^0) dengan δ = (60^0) sehingga didapat bentuk matrik

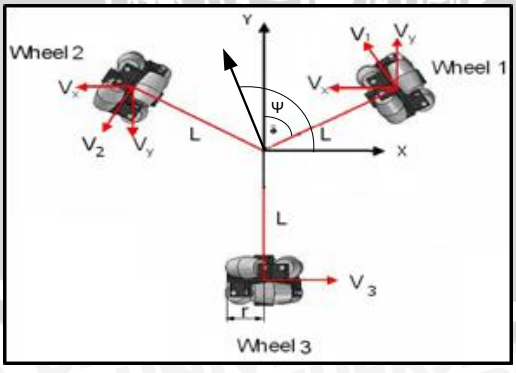
$$\begin{bmatrix} V_x \\ V_y \\ V_\emptyset \end{bmatrix} = \begin{bmatrix} -cos60^0 & -cos60^0 & 1 \\ sin60^0 & -sin60^0 & 0 \\ 1/L & 1/L & 1/L \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \quad \text{.....(3-1)}$$

Dengan menghitung nilai cos 60 dan sin 60 didapat bentuk matrik :

$$\begin{bmatrix} V_x \\ V_y \\ V_\emptyset \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & 1 \\ \frac{1}{3}\sqrt{3} & -\frac{1}{3}\sqrt{3} & 0 \\ 1/L & 1/L & 1/L \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \quad \text{.....(3-2)}$$

Agar robot bergerak ke sudut yang dinginkan maka harus diketahui terlebih dahulu perbandingan kecepatan putaran roda (V_1, V_2, V_3), jadi persamaan (3-2) harus dipindah ruas untuk mendapat nilai V_1, V_2, V_3 dan didapat matriks

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} -\frac{1}{3} & \frac{1}{\sqrt{3}} & L/3 \\ -\frac{1}{3} & -\frac{1}{\sqrt{3}} & L/3 \\ 2/3 & 0 & L/3 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_\emptyset \end{bmatrix} \quad \text{.....(3-3)}$$



Gambar 3.3 Arah Pergerakan Robot

Dari gambar 3.3 menunjukkan Ψ sebagai sudut arah pergerakan robot. Untuk mendapatkan arah orientasi robot (Ψ) digunakan persamaan :

$$\Psi = tan^{-1}\frac{Vy}{V_x} \quad \text{.....(3-4)}$$

$$tan\Psi = \frac{Vy}{V_x} = \frac{sin\Psi}{cos\Psi} \quad \text{.....(3-5)}$$

Dengan vector kecepatan robot = V_R maka,

$$V_y = \sin\Psi \,.\, V_R \quad \text{.....(3-6)}$$

$$V_x = \cos\Psi . V_R \quad \text{.....(3-7)}$$

Nilai V_x dan V_y pada persamaan (3-6) dan (3-7) dimasukkan dalam persamaan (3-3) dengan nilai $V_R = 1$, sehingga didapat bentuk matrik

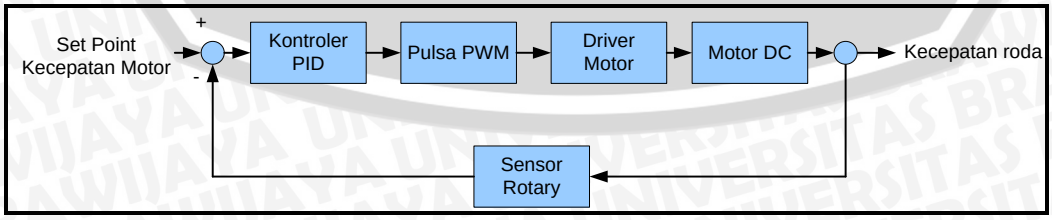
$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} -\frac{1}{3} & \frac{1}{\sqrt{3}} & L/3 \\ -\frac{1}{3} & -\frac{1}{\sqrt{3}} & L/3 \\ 2/3 & 0 & L/3 \end{bmatrix} \begin{bmatrix} cos\Psi . V_R \\ sin\Psi . V_R \\ V_\emptyset \end{bmatrix} \quad \text{.....(3-8)}$$

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} -\frac{1}{3} & \frac{1}{\sqrt{3}} & L/3 \\ -\frac{1}{3} & -\frac{1}{\sqrt{3}} & L/3 \\ 2/3 & 0 & L/3 \end{bmatrix} \begin{bmatrix} cos\Psi . 1 \\ sin\Psi . 1 \\ V_\emptyset \end{bmatrix} \quad \text{.....(3-9)}$$

Nilai V_1, V_2 ,V_3 digunakan sebagai perbandingan kecepatan tiap roda dan dikalikan dengan kecepatan roda tertentu yang nilainya harus kurang dari kecepatan maksimal roda yang paling pelan di antara ketiga roda, sehingga dapat diketahui kecepatan untuk masing-masing roda.

3.3.4 Perancangan Sistem Kontroler PID pada Robot *Three Omni-Directional*

Pada sistem kontroler PID ini dibutuhkan masukan sebagai umpan balik yang diproses oleh rumus PID. Pada perancangan ini, umpan balik diperoleh dari sensor rotary yang diletakkan di ketiga roda untuk mengukur kecepatan putar tiap roda. Gambar 3.4 menunjukkan diagram kontrol PID dari robot *Three Omni-directional.*

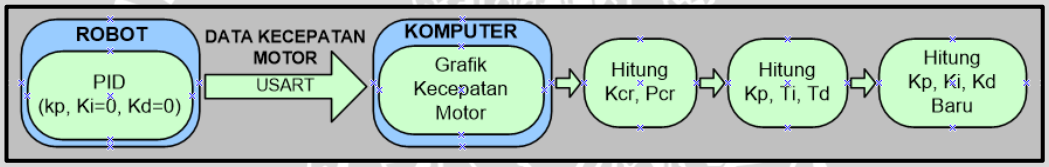


Gambar 3.4 Diagram blok kontrol PID robot *Three Omni-directional.*

Set point dari sistem ini adalah kecepatan putar roda yang dikirim dari mikrokontroler *master*. Kecepatan roda ini berdasarkan dari hasil perhitungan

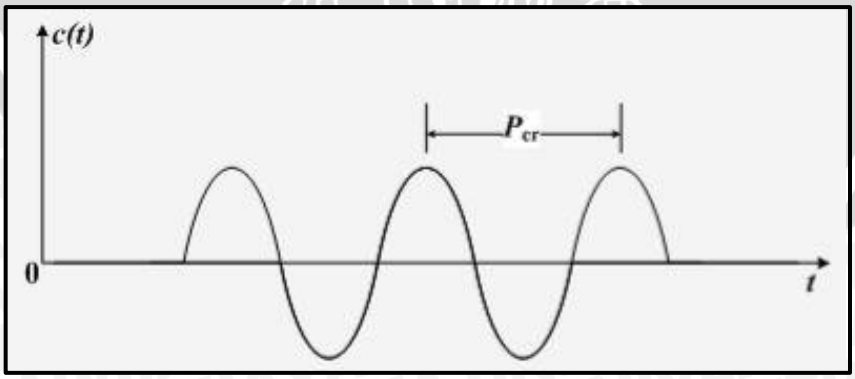
kinematika robot dengan masukan sudut arah gerak robot dan menghasilkan kecepatan untuk masing–masing roda sesuai dengan posisinya pada robot. Untuk melakukan proses pengontrolan dengan PID diperlukan suatu nilai *error* yaitu selisih antara *set point* dengan kecepatan putar roda yang terukur sensor rotary encoder.

Kontroler PID memiliki tiga parameter yang sangat berpengaruh pada kerja kontroler ini yaitu konstanta proporsional (Kp), konstanta integral (Ki), dan konstanta diferensial (Kd). Oleh karena itu dilakukan *tuning* eksperimen untuk mendapatkan nilai Kp, Ki, dan Kd yang tepat sehingga kontroler dapat bekerja secara optimal. Pada perancangan kontroler PID robot *three omni-directional* ini, menggunakan *tuning* parameter Ziegler-Nichols metode kedua. Gambar 3.5 menunjukkan diagram blok *tuning* PID pada robot *three omni-directional* dengan metode kedua Ziegler-Nichols.



Gambar 3.5 Diagram Blok *Tuning* PID dengan Metode kedua Ziegler-Nichols

Pada proses *tuning* kontrol PID dengan menggunakan metode osilasi Ziegler Nichols dimulai dengan memberikan nilai 0 pada parameter Ki dan Kd. Kemudian nilai Kp dinaikkan sedikit demi sedikit hingga didapatkan grafik kecepatan roda yang berkesinambungan. Kesinambungan yang dimaksud adalah saat grafik memiliki amplitudo yang sama pada setiap periodenya seperti yang ditunjukkan dalam Gambar 3.6.



Gambar 3.6 Osilasi Berkesinambungan dengan Periode Pcr
Sumber : Ogata, K., 1997

Setelah didapatkan grafik yang berkesinambungan langkah selanjutnya adalah menghitung nilai Kcr dan Pcr. Kcr adalah nilai Kp saat terjadi osilasi berkesinambungan sedangkan Pcr adalah periode kesinambungan dari grafik. Setelah didapatkan Kcr dan Pcr langkah selanjutnya adalah menghitung nilai Kp, Ti, dan Td sesuai dengan aturan Ziegler-Nichols.

$$Kp = 0.6\ x\ Kcr$$

$$Ti = 0.5 \text{ x } Pcr$$

$$Td = 0.125 \text{ x } Pcr$$

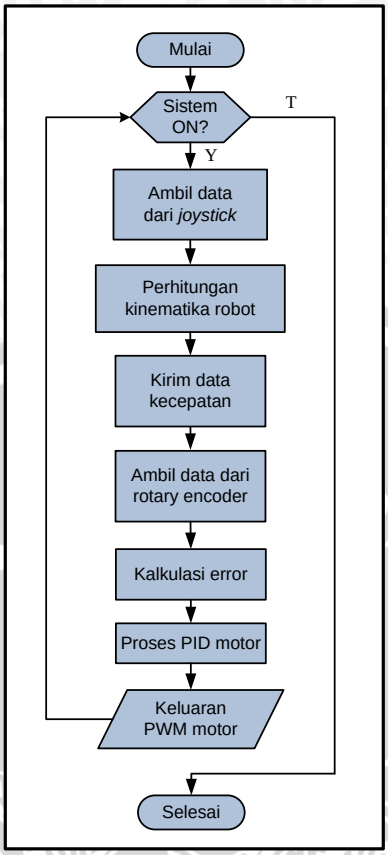
Nilai Ki dan Kd didapatkan dengan menggunakan perhitungan sebagai berikut.

$$Ki = \frac{Kp}{Ti}$$

$$Kd = Kp \text{ x } Td$$

3.3.5 Perancangan dan Pembuatan Perangkat Lunak (*Software*)

Perancangan perangkat lunak dibutuhkan mikrokontroler untuk mengendalikan perangkat keras. Perancangan perangkat lunak dilakukan dengan pembuatan *flowchart* untuk akuisisi data dari sensor, pengolahan data, transmisi data dan pengendalian sistem secara keseluruhan terlebih dahulu kemudian pembuatan programnya. Diagram alir perancangan perangkat lunak pada sistem mikrokontroler ditunjukkan dalam Gambar 3.7.



Gambar 3.7 Diagram Alir Proses Berjalannya Robot

Gambar 3.7 menunjukkan diagram alir perancangan perangkat lunak yang ada pada mikrokontroler robot. Pada saat mulai dinyalakan akan ditentukan terlebih dahulu arah gerak robot sesuai data dari tombol analog. Mikrokontroler *master* akan menghitung kinematika robot dengan masukan sudut gerak robot dari tombol analog sehingga didapat kecepatan masing-masing roda agar didapat arah pergerakan robot yang diinginkan. Setelah itu, data kecepatan roda hasil perhitungan MK *master* dikirim ke masing-masing mikrokontroler *slave*. Mikrokontroler *slave* mengambil data kecepatan putaran motor dari rotary encoder. Data kecepatan roda digunakan sebagai masukan pada MK *slave* untuk menentukan error yaitu selisih antara kecepatan roda yang diinginkan (*set point*) dan kecepatan roda yang terbaca sensor. Nilai error ini digunakan sebagai masukan rumus kontroler PID, dan hasil perhitungan kontroler PID akan dikeluarkan oleh MK *slave* berupa pulsa PWM (Pulse-Width Modulation). Pulsa PWM digunakan sebagai pengendali kecepatan putaran motor. Semakin besar nilai PWM semakin cepat putaran motor, semakin kecil nilai PWM semakin pelan putaran motor.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

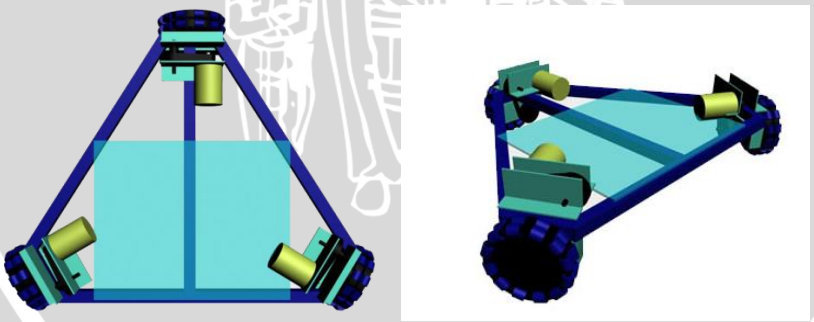
Perancangan robot *three omni-directional* ini dilakukan secara bertahap sehingga akan memudahkan dalam analisis pada setiap bloknya maupun secara keseluruhan. perancangan ini terdiri dari :

- Perancangan perangkat keras terdiri dari dua bagian yaitu sistem mekanik robot dan desain sistem elektronik.
- Perancangan antarmuka *joystick* playstation dengan mikrokontroler ATMega32
- Perancangan Sistem Kinematika Robot *Three Omni Directional*
- Perancangan sistem kontroler PID.
- Perancangan perangkat lunak.

4.1 Perancangan Perangkat Keras

4.1.1 Perancangan Mekanik Robot

Sistem mekanik yang baik berpengaruh besar pada pergerakan robot, oleh karena itu perancangan mekanik dalam hal ini bodi dan rangka robot haruslah dibuat sepresisi mungkin. Gambar 4.1 menunjukkan penampakan mekanik robot.



(a) (b)

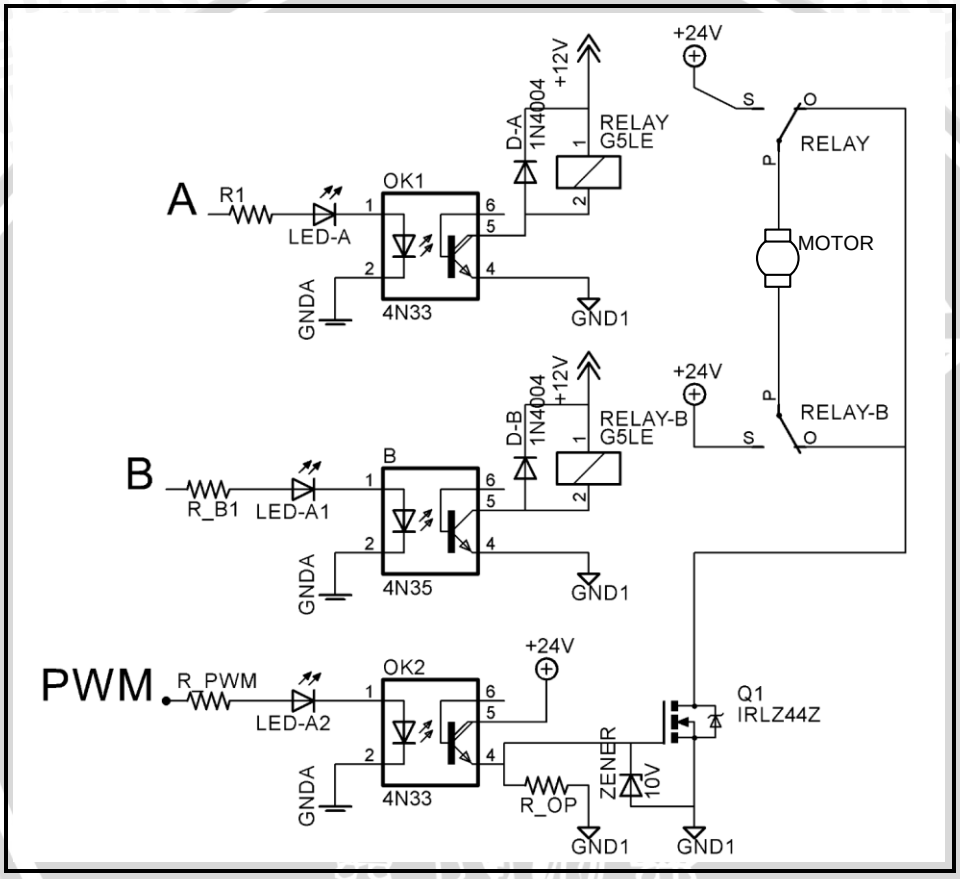
Gambar 4.1 (a) Rancangan robot tampak atas, (b) Rancangan robot tampak perspektif

Badan robot terbuat dari aluminium dengan lebar 2 cm, ketiga roda memakai roda omni berdiameter 12,5 cm dengan tebal 3 cm. Ukuran keseluruhan robot adalah 70 x 70 cm.

4.1.2 Perancangan Desain Rangkaian Elektrik

4.1.2.1 Perancangan Rangkaian Driver Motor

Robot pada perancangan ini menggunakan sistem *driver relay.* Sistem ini terdiri atas sepasang *relay* yang bekerja secara terbalik untuk mengatur arah putaran motor dan komponen E-MOSFET untuk mengatur kecepatan pada motor. Secara keseluruhan rangkaian pengendali arah putaran dan kecepatan motor ditunjukkan dalam Gambar 4.2.



Gambar 4.2 Rangkaian Keseluruhan *Driver* Motor

Dalam proses pengontrolan *driver* motor dibutuhkan suatu komponen yang dapat menghubungkan mikrokontroler yang memiliki keluaran hanya 5V dengan relay dan E-Mosfet yang membutuhkan tegangan lebih dari 5 V. Selain itu komponen ini juga harus dapat memisahkan rangkaian mikrokontroler dengan relay dan E-Mosfet secara elektronis. Hal ini bertujuan agar ketika terjadi perubahan tegangan atau arus pada bagian E-MOSFET atau koil relay tidak akan membebani kerja mikrokontroler. Untuk itu digunakan rangkaian pengendali optik yaitu *optocoupler* 4N33.

Optocoupler 4N33 memiliki spesifikasi sebagai berikut

- V_F = 1,2 V
- I_{Fmax} = 80 mA
- I_{Cmax} = 150 mA
- V_{CEsat} = 1 V
- CTR = 500%

Alasan digunakannya komponen *optocoupler* 4N33 adalah karena 4N33 memiliki CTR_{min} yang besar sehingga dengan nilai I_F yang kecil dapat dikonversi menjadi nilai I_C yang besar (5 kali lipat).

Perencanaan awal relay yang digunakan adalah relay HRS4H-S-12VDC dan E-MOSFET yang digunakan adalah IRFZ44N. Spesifikasi dari relay HRS4H-S-12VDC adalah sebagai berikut :

- *Coil nominal* VDC = 12V
- *Coil operate voltage* VDC = 9V
- *Coil resistance* = 320 Ω ±10%
- *Max switching voltage* = 110VDC / 240 VAC
- *Max Switching Current* = 15 A

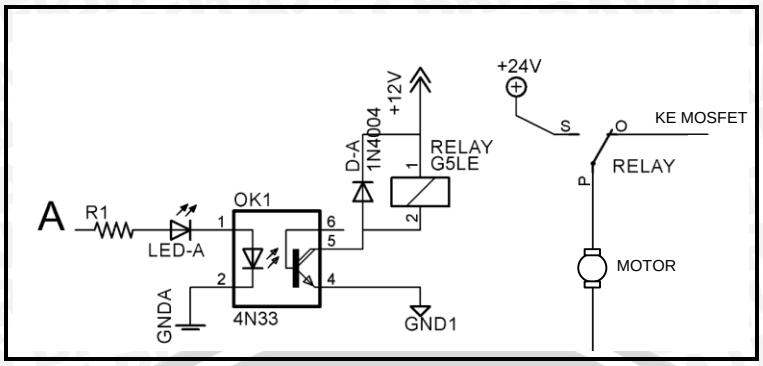
Spesifikasi dari IRFZ44N adalah sebagai berikut

- $V_{DS\ max}$ = 55V
- $I_{D\ max}$ = 49A
- R_{DS} = 17,5 mΩ
- $V_{GS\ max}$ = ±20V
- $V_{GS\ threshold}$ = 2V - 4V

Alasan Pemilihan relay sebagai pengatur arah putaran motor adalah karena relay dapat menghantarkan arus yang besar dan memiliki resistansi kontak yang kecil. Sebagai pengatur kecepatan motor dipilih IRFZ44N karena FET tipe ini memiliki $I_{D\ max}$ cukup besar dan $R_{DS\ on\ max}$ kecil dibanding beberapa jenis E-MOSFET kanal N lain.

4.1.2.2 Perancangan Driver Optik Untuk Relay

Gambar 4.3 menunjukkan rangkaian dari *driver relay*. Rangkaian ini terdiri atas resistor, led 3mm, optocoupler, dioda dan relay.



Gambar 4.3 Rangkaian bagian *Driver Relay*

Pengontrol arah dan kecepatan motor menggunakan ATMega8, dengan spesifikasi:

- I_{OH} = -20 mA
- I_{OL} = 20 mA
- V_{OH} = 4,2 V
- V_{OL} = 0,9 V

Arus minimal yang dibutuhkan agar *coil* relay aktif adalah :

$$I_c = \frac{V_{coil}}{coil\ resistance} \quad \text{.....(4-1)}$$

$$I_c = \frac{9V}{320\Omega + (10\% \times 320)}$$

$$I_c = \frac{9V}{352\ \Omega}$$

$$I_{c\ min} = 25{,}57\ mA$$

Nilai ini masih di bawah nilai arus maksimal keluaran transistor *optocoupler* sebesar 150 mA. Arus I_F dapat dihitung dengan

$$CTR = \frac{I_C}{I_F} x\ 100\% \quad \text{.....(4-2)}$$

$$I_F = \frac{I_C}{CTR} x\ 100\%$$

$$I_F = \frac{25{,}57\ mA}{500\%} x\ 100\%$$

$$I_F = 5{,}11\ mA$$

Jadi arus yang mengalir dalam LED *optocoupler* sebesar 5,11 mA, sedangkan arus yang maksimal dari mikrokontroler lebih besar yaitu 20 mA. Untuk menyesuaikan nilai arus maka dibutuhkan resistor dengan nilai sebagai berikut.

$$R_1 = \frac{V_A - V_F - V_{LED}}{I_F} \quad \text{.....(4-3)}$$

$$= \frac{5V - 1{,}2V - 2{,}5V}{5{,}11\ X\ 10^{-3} A}$$

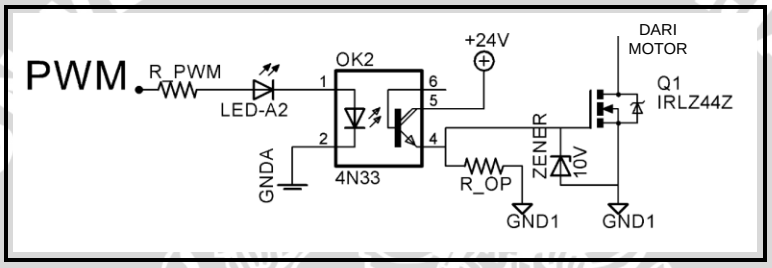
$$= \frac{1{,}3V}{5{,}11\ X\ 10^{-3} A}$$

$$R_1 = 254{,}4\ \Omega$$

R_1 yang digunakan dalam skripsi ini adalah 220 Ω

4.1.2.3 Perancangan Rangkaian Pengontrol Kecepatan Motor

Pada perancangan rangkaian pengontrol kecepatan ini digunakan E-MOSFET kanal N dengan masukan berupa PWM dari mikrokontroler. Gambar 4.4 menunjukkan rangkaian pengontrol kecepatan motor.



Gambar 4.4 Rangkaian Pengontrol Kecepatan dengan Menggunakan E-MOSFET

Untuk aplikasi *driver* motor, FET selalu dikondisikan dalam keadaan saturasi atau *cut off*-nya. Hal ini dimaksudkan agar tidak terlalu banyak daya yang terbuang dalam FET itu sendiri.

V_{GS} yang digunakan adalah 10V. Untuk membatasi tegangan V_{GS} ini digunakan dioda zener 10V.

Perhitungan nilai resistor untuk tegangan masukan E-MOSFET sebagai berikut.

$$R_{op} = \frac{24\ V - V_{CE\ sat}}{I_{C\ max}}$$

$$= \frac{24V - 1\ V}{150\ X\ 10^{-3}\ A}$$

$$R_{op\ min} = 153{,}33\Omega$$

Nilai R_{op} yang digunakan sebesar 270Ω. Resistor ini berfungsi untuk membatasi arus yang melewati transistor dalam *optocoupler*. Dengan nilai R_{op} tersebut, maka :

$$I_C = \frac{24V - V_{CE\ sat}}{R_{op}}$$

$$= \frac{24V - 1V}{270}$$

$$= 0{,}08518\ A$$

$$I_c = 85{,}18\ mA$$

Dari hasil perhitungan diatas didapat nilai I_c maksimal = 85,18 mA saat R_{op} sebesar 270 Ω. Arus masukan *optocoupler* dapat dihitung dengan cara

$$CTR = \frac{I_C}{I_F} x\ 100\%$$

$$I_F = \frac{I_C}{CTR} x\ 100\%$$

$$I_F = \frac{85{,}18\ mA}{500\%} x\ 100\%$$

$$I_F = 17{,}03\ mA$$

Jadi arus yang masuk ke *optocoupler* sebesar 17,03 mA. Nilai arus ini lebih kecil dibandingkan dengan arus keluaran maksimal mikrokontroler yang sebesar 20 mA, sehingga diperlukan resistor pada masukan *optocoupler* untuk menyesuaikan nilai arus. Nilai resistor *input* dapat dihitung dengan cara sebagai berikut

$$R_{PWM} = \frac{V_A - V_F - V_{LED}}{I_F}$$

$$= \frac{5V - 1{,}2V - 2{,}5V}{17{,}03\ X\ 10^{-3}}$$

$$= \frac{1{,}3V}{17{,}03\ X\ 10^{-3}}$$

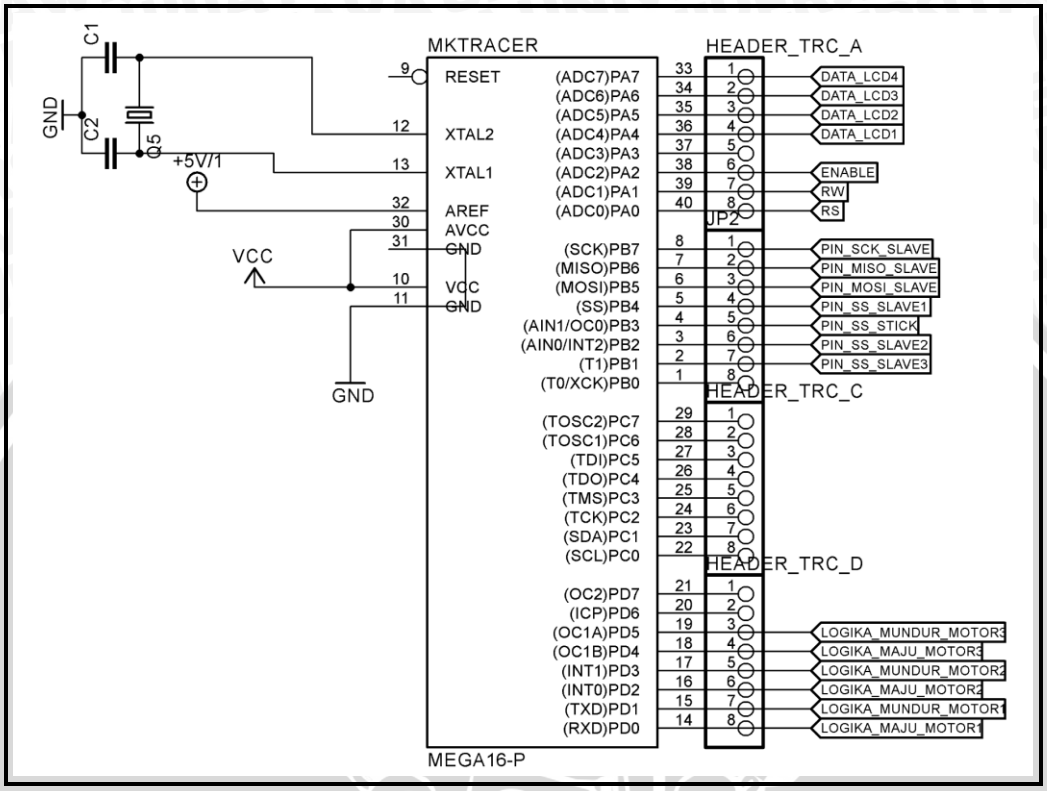
$$R_{PWM} = 76{,}34\ \Omega$$

Berdasarkan perhitungan diatas, R_{PWM} saat R_{op} =270 Ω adalah 76,34Ω. R_{pwm} yang digunakan dalam perancangan ini adalah 100 Ω.

4.1.2.4 Sistem Mikrokontroler ATMega32 dan ATmega8

Pada Perancangan Mikrokontroler Robot *Three Omni Directional* ini menggunakan empat buah mikrokontroler. Sebuah mikrokontroler *master* dan tiga buah mikrokontroler *slave*. Penggunaan empat buah mikrokontroler ini bertujuan agar terdapat pembagian tugas yang merata untuk masing–masing mikrokontroler sehingga proses pengolahan datanya lebih cepat. Mikrokontroler *master* menggunakan ATMega32 dan mikrokontroler *slave* menggunakan ATMega8. Mikrokontroler master bertugas untuk melakukan perhitungan kecepatan tiap

motor sesuai dengan sudut gerak robot yang diinginkan sedangkan mikrokontroler *slave* bertugas untuk mengolah data dari sensor rotary encoder menjadi data *error* dan melakukan perhitungan kontroler PID. Gambar 4.5 menunjukkan rangkaian sistem minimum mikrokontroler ATMega32.



Gambar 4.5 Sistem Minimum Mikrokontroler ATMega32

Mikrokontroler ATMega32 mempunyai 32 jalur I/O yang dapat diprogram menjadi masukan dan keluaran. Ke-32 jalur tersebut dikelompokkan ke dalam empat *port*, yaitu *port* A, B, C, dan D. Pada perancangan ini pin yang digunakan adalah:

PIN A.0 = digunakan sebagai antarmuka dengan LCD sebagai pin *register select*

PIN A.1 = digunakan sebagai antarmuka dengan LCD sebagai pin R/W

PIN A.2 = digunakan sebagai antarmuka dengan LCD sebagai pin *enable*

PIN A.4 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus.

PIN A.5 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus

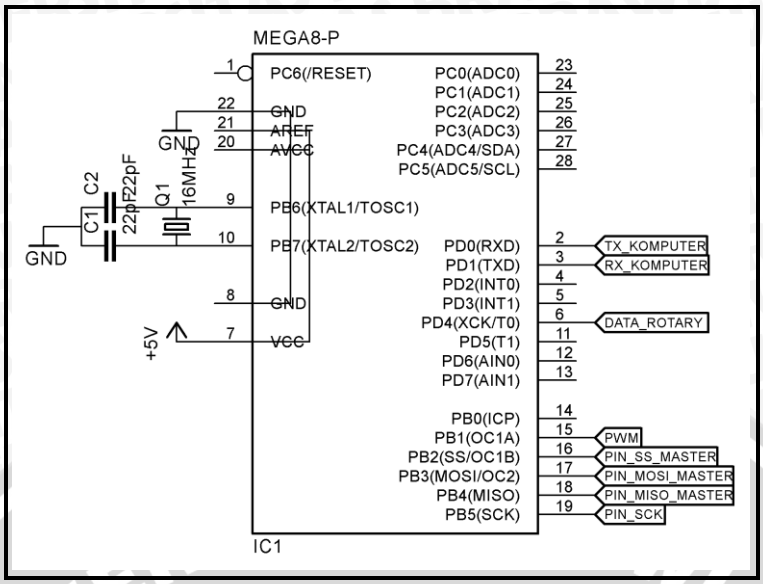
PIN A.6 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus

PIN A.7 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus

PIN B.1 = difungsikan sebagai SS3 (untuk komunikasi SPI dengan *slave* 3)

PIN B.2 = difungsikan sebagai SS2 (untuk komunikasi SPI dengan *slave* 2)
PIN B.3 = difungsikan sebagai SS (untuk komunikasi SPI dengan)
PIN B.4 = pin SS difungsikan sebagai SS1 (untuk komunikasi SPI dengan *slave* 1)
PIN B.5 = pin MOSI (untuk komunikasi SPI dengan *slave*)
PIN B.6 = pin MISO (untuk komunikasi SPI dengan *slave*)
PIN B.7 = pin SCK (untuk komunikasi SPI dengan *slave*)
PIN C.0 = digunakan sebagai antarmuka dengan LCD sebagai pin *register select*
PIN C.1 = digunakan sebagai antarmuka dengan LCD sebagai pin R/W
PIN C.2 = digunakan sebagai antarmuka dengan LCD sebagai pin enable
PIN C.4 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus
PIN C.5 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus
PIN C.6 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus
PIN C.7 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus
PIN D.0 = Rx untuk transmisi UART
PIN D.1 = Tx untuk transmisi UART
XTAL1 = digunakan sebagai masukan dari rangkaian osilator kristal
XTAL2 = digunakan sebagai masukan dari rangkaian osilator kristal

Mikrokontroler *slave* menggunakan mikrokontroler ATMega8. Mikrokontroler ini dipilih karena hanya difungsikan sebagai pembaca sensor *rotary encoder* dan kontroler PID motor sehingga tidak memerlukan pin I/O yang banyak. Gambar 4.6 menunjukkan rangkaian minimum dari ATMega8.



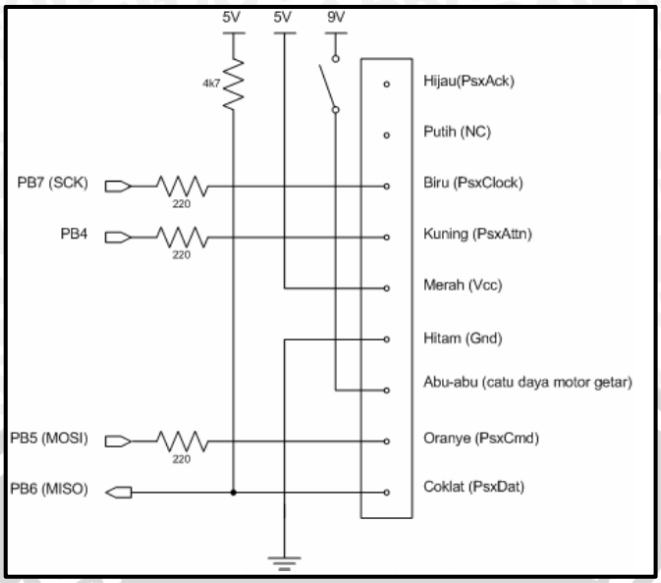
Gambar 4.6 Sistem Minimum Mikrokontroler ATMega8

Mikrokontroler ATMega8 mempunyai 23 jalur I/O yang dapat diprogram menjadi masukan atau keluaran. Ke-23 jalur I/O ini dikelompokkan menjadi 3 kelompok, yaitu *port* B, C, dan D. Pada perancangan ini pin – oin yang digunakan adalah sebagai berikut:

PIN PB.1 = membangkitkan sinyal PWM
PIN PB.2 = pin SS (untuk komunikasi SPI dengan M*aster*)
PIN PB.3 = pin MOSI (untuk komunikasi SPI dengan M*aster*)
PIN PB.4 = pin MISO (untuk komunikasi SPI dengan M*aster*)
PIN PB.5 = pin SCK (untuk komunikasi SPI dengan M*aster*)
PIN PD.0 = Rx untuk transmisi UART
PIN PD.1 = Tx untuk transmisi UART
PIN PD.4 = digunakan sebagai pin masukan sensor *rotary encoder*

4.2 Perancangan Antarmuka *Joystick* Playstation dengan Mikrokontroler ATmega32

Antarmuka antara *joystick* playstation (PS) dengan mikrokontroler ATmega32 menggunakan komunikasi SPI. Kabel *joystick* sendiri memiliki 9 kabel: 4 kabel digunakan untuk komunikasi SPI, 3 kabel untuk catu daya *joystick* dan 2 kabel tidak digunakan. Rangkaian antarmuka antara *joystick* PS dengan mikrokontroler ATMega32 dapat dilihat dalam gambar 4.7.



Gambar 4.7 Rangkaian Antarmuka *Joystick* Playstation dengan Mikrokontroler ATmega32
Sumber : Nugroho,2009:1

Rangkaian yang digunakan pada antarmuka ini membutuhkan beberapa resistor, terdiri dari resistor 220 Ω sebagai pengaman arus dan satu buah resistor 4,7k Ω sebagai *pull up*. Resistor sebagai pengaman diletakkan pada pin PsxCmd, PsxClock, dan PsxAtn berfungsi sebagai pembatas arus. Pada pin PsxDat harus diberi *pull up* karena bersifat *open collector*.

Pertukaran data antara kontroler *joystick* PS analog dan mikrokontroler seperti tercantum pada tabel 4.1. Untuk mengawali komunikasi dengan *joystick* PS, mikrokontroler mengirim byte pertama berupa perintah 0x01, kemudian byte kedua perintah 0x42 (baca data) sekaligus menerima tipe joystick (0x41 untuk digital dan 0x73 untuk analog) dari kontroler *joystick.* Berturut-turut setelah itu mikrokontroler menerima data dari *joystick* berupa data 0x5A (siap), data digital 1, dan data digital 2. Data digital ini bersifat *active low,* jadi jika tidak ada tombol ditekan akan diterima data 0xff (0b11111111). Jika *joystick* dalam kondisi operasi analog maka komunikasi dilanjutkan dengan data analog 1, data analog 2, data analog 3, dan data analog 4. Pada saat kondisi netral data anlog berupa data 0x7f (127) dengan nilai minimal 0 dan nilai maksimal 0xff (255). Saat operasi digital byte 6 – 9 selalu memberikan data 0xff (Nugroho,2009).

Pada Perancangan ini digunakan *joystick* analog, karena terdapat tombol analog sebagai kendali arah gerak robot sedangkan pada *joystick* digital tidak terdapat tombol analog.

Tabel 4.1 Pertukaran Data untuk *Joystick* Analog

Byte	PsxCmd	PsxData							
01	0x01	-							
02	0x42	0x73							
03	-	0x5A	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
04	-	Data Digital 1	←	↓	→	↑	start	Joy-R	Joy-L	select
05	-	Data Digital 2	□	X	O	Δ	R1	L1	R2	L2
06	-	Data Analog 1	*Joystick* Analog Kanan – Sumbu X 0x00-kiri, 0xff-kanan							
07	-	Data Analog 2	*Joystick* Analog Kanan – Sumbu Y 0x00-atas, 0xff-bawah							
08	-	Data Analog 3	*Joystick* Analog Kiri – Sumbu X 0x00-kiri, 0xff-kanan							
09	-	Data Analog 4	*Joystick* Analog Kiri – Sumbu Y 0x00-atas, 0xff-bawah							

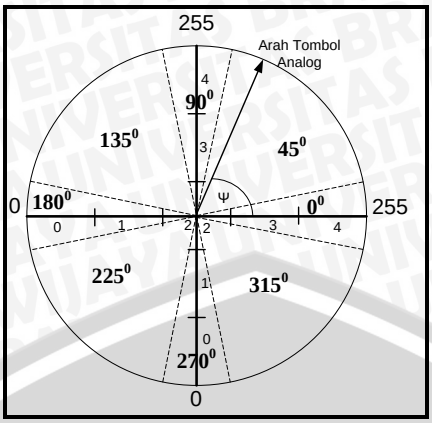
Sumber : Nugroho, 2009

4.3 Perancangan Konversi Data *Joystick* Menjadi Sudut

Data dari Joystick masih berupa nilai ADC dan harus diproses terlebih dahulu agar diperoleh sudut gerak tombol analog yang disimbolkan dengan Ψ. Nilai ADC yang diterima sebesar dari 0-255. Untuk memudahkan pemrosesan ini maka nilai ADC dibagi menjadi 5 kelompok yaitu :

- Kelompok 0 : 0-50
- Kelompok 1 : 51-101
- Kelompok 2 : 102-152
- Kelompok 3 : 153-203
- Kelompok 4 : 204-255

Pembagian nilai ini dilakukan pada nilai ADC sumbu-X dan sumbu-Y yang terbaca mikrokontroler. Besar sudut (Ψ) diperoleh dengan melihat pada kelompok apa nilai ADC yang terbaca. Penentuan nilai sudut dapat dilihat dalam Gambar 4.8



Gambar 4.8 Penentuan sudut arah tombol analog berdasar kelompok nilai ADCnya

4.4 Perancangan Sistem Kinematika Robot *Three Omni Directional*

Pada robot *three omni-directional* untuk bergerak ke sudut yang dinginkan maka harus diketahui terlebih dahulu perbandingan kecepatan motor (V_1, V_2, V_3). Nilai dari V_1, V_2, dan V_3 dapat diperoleh dari penjabaran matriks kinematika robot (persamaan 3-9) menjadi persamaan

$$V_1 = -\frac{1}{3}.\cos\Psi + \frac{1}{\sqrt{3}}.sin\Psi + \frac{L}{3}.V_\emptyset \quad \text{.....(4-4)}$$

$$V_2 = -\frac{1}{3}.\cos\Psi - \frac{1}{\sqrt{3}}.sin\Psi + \frac{L}{3}.V_\emptyset \quad \text{.....(4-5)}$$

$$V_3 = \frac{2}{3}.cos\Psi + \frac{L}{3}.V_\emptyset \quad \text{.....(4-6)}$$

Nilai V_1, V_2 ,V_3 digunakan sebagai perbandingan kecepatan tiap roda dan dikalikan dengan kecepatan tertentu, dalam skripsi ini ditetapkan kecepatan yang digunakan sebagai pengali adalah 300 rpm. Kecepatan yang digunakan ini harus sama atau lebih pelan dibandingkan dengan kecepatan maksimal roda yang paling pelan diantara tiga roda tersebut.

Berikut salah satu contoh perhitungan nilai V_1, V_2 dan V_3 pada sudut 0^0 :

$$V_1 = -\frac{1}{3}.\cos 0^0 + \frac{1}{\sqrt{3}}.sin0^0 + \frac{L}{3}.V_\emptyset$$

$$V_1 = -\frac{1}{3}$$

$$V_2 = -\frac{1}{3}.\cos 0^0 - \frac{1}{\sqrt{3}}.sin0^0 + \frac{L}{3}.V_\emptyset$$

$$V_2 = -\frac{1}{3}$$

$$V_3 = \frac{2}{3}.cos0^0 + \frac{L}{3}.V_\emptyset$$

$$V_3 = \frac{2}{3}$$

Kecepatan roda untuk pergerakan robot ke beberapa sudut lain dapat dilihat dalam tabel 4.2

Tabel 4.2 Kecepatan Tiap Roda

No	Sudut Gerak Robot (^{0})	V1(rpm)	V2(rpm)	V3(rpm
1	0	-100	-100	200
2	45	52	-193	141
3	90	173	-173	0
4	135	193	-52	-141
5	180	100	100	-200
6	225	-52	193	-141
7	270	-173	173	0
8	315	-193	52	141

Nilai negatif menunjukkan arah perputaran roda yang searah dengan arah jarum jam sedangkan nilai positif menunjukkan arah putaran roda yang berlawanan arah jarum jam.

4.5 Perancangan Sistem Tuning Kontroler PID Menggunakan Metode Kedua Ziegler-Nichols pada *Robot Three Omni-Directional*

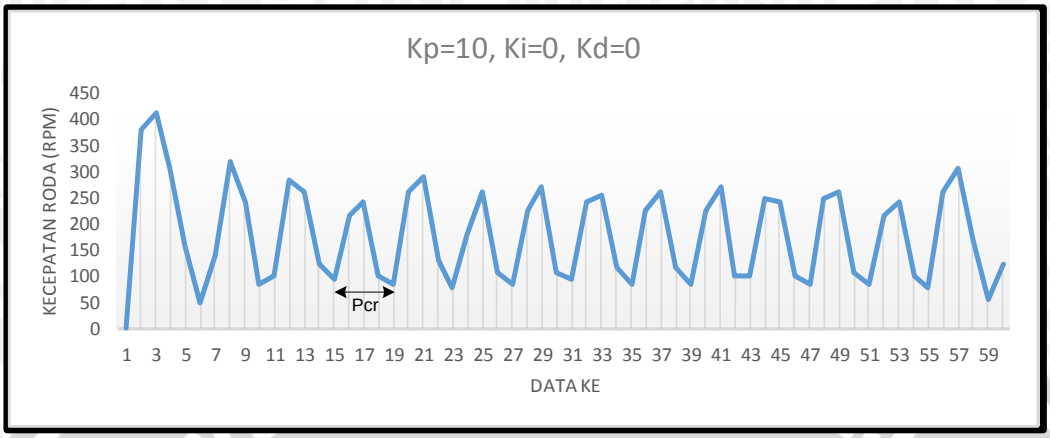
Proses *Tuning* PID pada perancangan ini menggunakan metode kedua Ziegler Nichols sehingga harus dicari terlebih dahulu nilai Kp saat terjadi osilasi berkesinambungan pada masing-masing motor dengan nilai Ki dan Kd adalah 0. Dari nilai Kp saat terjadi osilasi berkesinambungan dapat dicari Kcr dan Pcr. Dengan nilai Kcr dan Pcr, dapat dihitung nilai Ki dan Kd.

Nilai *set point* (kecepatan putaran roda yang diinginkan) yang akan dipakai harus ditentukan terlebih dahulu sebelum mencari grafik osilasi berkesinambungan. Pada robot *three omni-directional, set point* yang digunakan berubah – ubah bergantung pada sudut gerak robot. Oleh karena itu pada perancangan ini ditetapkan set point yang digunakan sebesar 200 rpm untuk mempermudah *tuning* parameter PID.

4.5.1 Perancangan *Tuning* Parameter PID Pada Roda 1

Proses *Tuning* diawali dengan merubah nilai Kp dari 0, 3, 5, 8, 9, dan 10. Grafik respon putaran roda yang berkesinambungan didapat saat nilai kontroler

proporsional 10 (Kcr=10). Hasil pengujian respon putaran roda dengan menggunakan kontroler proporsional dengan nilai 10 (Kcr=10) dapat dilihat dalam Gambar 4.9.



Gambar 4.9 Grafik respon kecepatan roda 1 dengan Kcr=10, Ki=0, Kd=0

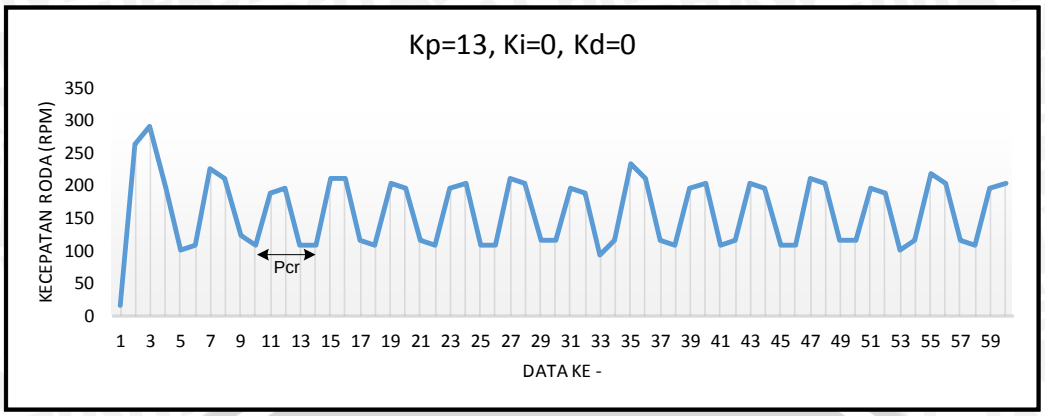
Gambar 4.9 menunjukkan bahwa pada saat kontroler proporsional bernilai 10 kecepatan roda dapat membentuk osilasi berkesinambungan. Respon sistem menampilkan data setiap 32,64 ms sehingga nilai Kcr dan Pcr dapat dihitung. Nilai Ki dan Kd dapat diperoleh dengan perhitungan sebagai berikut

- Kcr = 10
- Pcr = (19 - 15) x *Time Sampling* = 4 x 32,64 x 10^{-3} = 0,130 sekon
- Kp = 0,6 x Kcr = 0,6 x 10 = 6
- Ti = 0,5 x Pcr = 0,5 x 0,13 = 0,065
- Td = 0,125 x Pcr = 0,125 x 0,13 = 0,016
- $Ki = \frac{Kp}{Ti} = \frac{6}{0{,}065} = 92{,}31$
- $Kd = Kp \times Td = 6 \times 0{,}016 = 0{,}096$

Hasil *tuning* parameter PID dengan menggunakan metode kedua Ziegler-Nichols pada roda 1 diperoleh nilai Kp = 6, Ki = 92,31, dan Kd = 0,096.

4.5.2 Perancangan *Tuning* Parameter PID Pada Roda 2

Proses *Tuning* diawali dengan merubah nilai Kp dari 0, 3, 5, 10, 11, 12, dan 13. Grafik respon putaran roda yang berkesinambungan didapat saat kontroler proporsional bernilai 13 (Kcr=13). Hasil pengujian respon putaran roda dengan menggunakan kontroler proporsional dengan nilai 13 (Kcr=13) dapat dilihat dalam Gambar 4.10.



Gambar 4.10 Grafik respon kecepatan roda 2 dengan Kcr=13, Ki=0, Kd=0

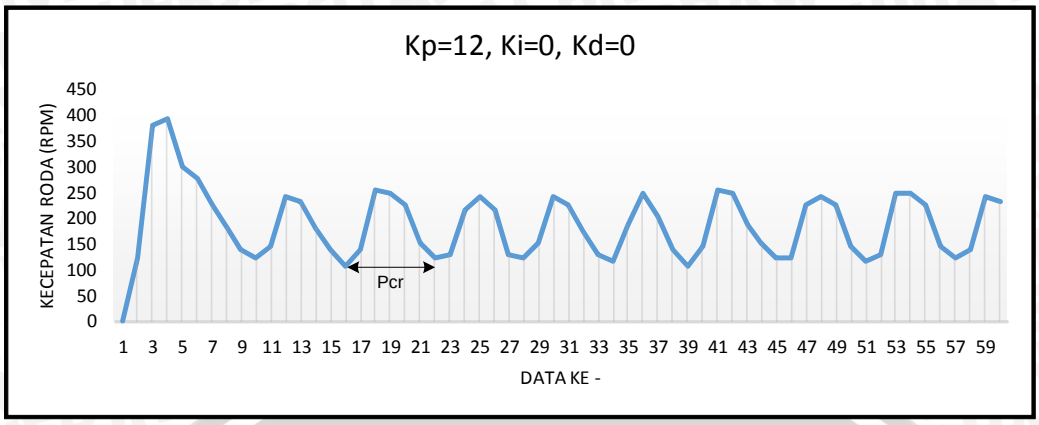
Gambar 4.10 menunjukkan bahwa pada saat kontroler proporsional bernilai 13 kecepatan roda dapat membentuk osilasi berkesinambungan. Respon sistem menampilkan data setiap 32,64 ms sehingga nilai kcr dan Pcr dapat dihitung. Nilai Ki dan Kd dapat diperoleh dengan perhitungan sebagai berikut

- Kcr = 13
- Pcr = (14 - 10) x *Time Sampling* = 4 x 32,64 x 10^{-3} = 0,130 sekon
- Kp = 0,6 x Kcr = 0,6 x 13 = 7,8
- Ti = 0,5 x Pcr = 0,5 x 0,13 = 0,065
- Td = 0,125 x Pcr = 0,125 x 0,13 = 0,016
- $Ki = \frac{Kp}{Ti} = \frac{7,8}{0,065} = 120$
- $Kd = Kp \times Td = 7,8 \times 0,016 = 0,1248$

Hasil *tuning* parameter PID dengan menggunakan metode kedua Ziegler-Nichols pada roda 2 diperoleh nilai Kp = 7,8 , Ki =120, dan Kd = 0,1248.

4.5.3 Perancangan *Tuning* Parameter PID Pada Roda 3

Proses *Tuning* diawali dengan merubah nilai Kp dari 0, 3, 5, 10, 11, dan 12. Grafik respon putaran roda yang berkesinambungan didapat saat kontroler proporsional bernilai 12 (Kcr=12). Hasil pengujian respon putaran roda dengan menggunakan kontroler proporsional dengan nilai 12 (Kcr=12) dapat dilihat dalam Gambar 4.11.



Gambar 4.11 Grafik respon kecepatan roda 3 dengan Kcr=12, Ki=0, Kd=0

Gambar 4.11 menunjukkan bahwa pada saat kontroler proporsional bernilai 12 kecepatan roda dapat memebentuk osilasi berkesinambungan. Respon sistem menampilkan data setiap 32,64 ms sehingga nilai kcr dan Pcr dapat dihitung. Nilai Ki dan Kd dapat diperoleh dengan perhitungan sebagai berikut

- Kcr = 12
- Pcr = (22 - 16) x *Time Sampling* = 6 x 32,64 x 10^{-3} = 0,196 sekon
- Kp = 0,6 x Kcr = 0,6 x 12 = 7,2
- Ti = 0,5 x Pcr = 0,5 x 0,196 = 0,098
- Td = 0,125 x Pcr = 0,125 x 0,196 = 0,0245
- $Ki = \frac{Kp}{Ti} = \frac{7,2}{0,098} = 73,47$
- $Kd = Kp \times Td = 7,2 \times 0,0245 = 0,1764$

Hasil *tuning* parameter PID dengan menggunakan metode kedua Ziegler-Nichols pada roda 3 diperoleh nilai Kp = 7,2 , Ki =73,47 , dan Kd = 0,1764.

4.6 Perancangan Perangkat Lunak

Dalam skripsi ini perancangan perangkat lunak terdiri atas dua bagian, yaitu perancangan perangkat lunak untuk mikrokontroler *master* dan mikrokontroler *slave*. Perancangan perangkat lunak *master* untuk mengolah masukan dari tombol analog dan untuk proses pengolahan rumus kinematika robot *three omni-directional*. Sedangkan perancangan perangkat lunak *slave* untuk mengolah data sensor *rotary encoder* dan proses sistem kontrol PID.

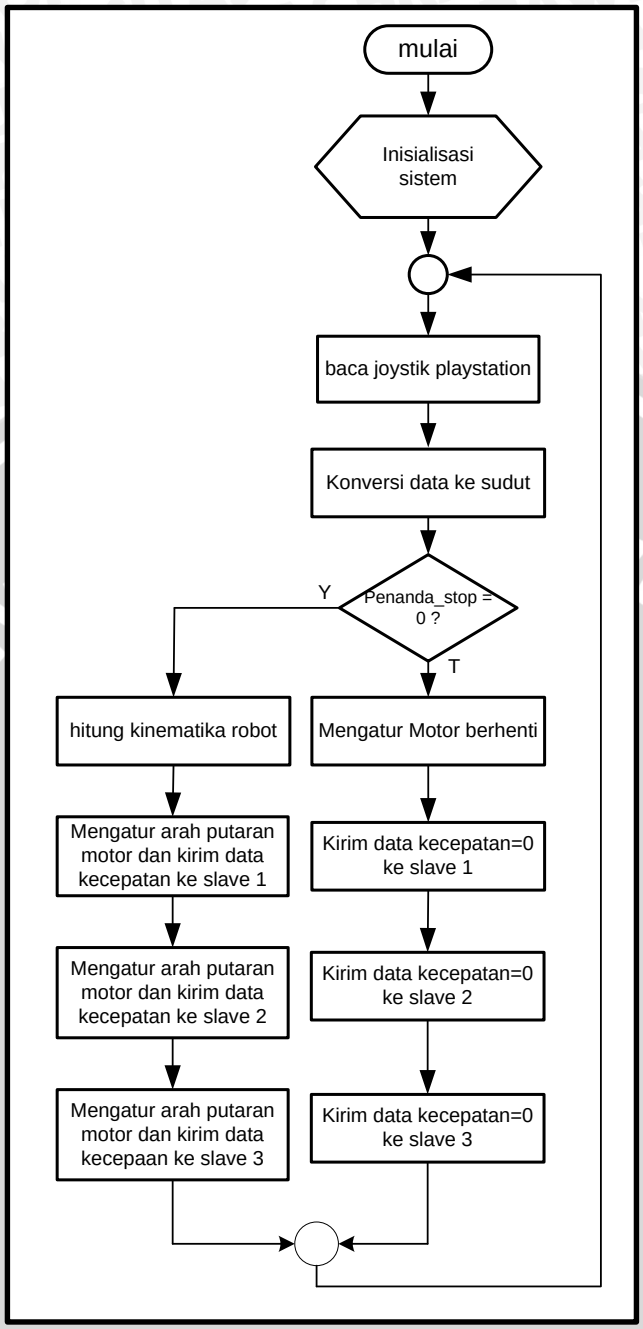
4.6.1 Perancangan Susunan Perangkat Lunak

Tahapan proses yang terdapat pada system ini meliputi, proses pengolahan data dari tombol analog *joystick* via komunikasi SPI, menentukan nilai error kecepatan roda dengan data dari sensor *rotary encoder*, melakukan pengolahan rumus kinematika robot dan pengontrolan motor dengan algoritma PID. Semua proses tersebut dilakukan oleh perangkat lunak yang terdapat dalam mirokontroler. Perangkat lunak ini tersusun dari instruksi – instruksi yang membentuk sebuah *listing* program atau *source code.*

Semua instruksi program disusun secara terstruktur dalam beberapa subrutin yang secara khusus menangani fungsi tertentu. *Software* mikrokontroler dibuat menggunakan *compiler Code Vision AVR* buatan HP infotech. Menggunakan bahasa pemrogaman yaitu bahasa C dan *downloader* menggunakan AVR USB.

4.6.2 Perancangan Program Mikrokontroler *Master*

Program utama mikrokontroler master dirancang untuk melakukan proses antarmuka dengan *joystick* playstation, menghitung kinematika robot dan komunikasi SPI dengan tiga mikrokontroler *slave*. Diagram alir program utama *master* ditunjukkan dalam Gambar 4.12.



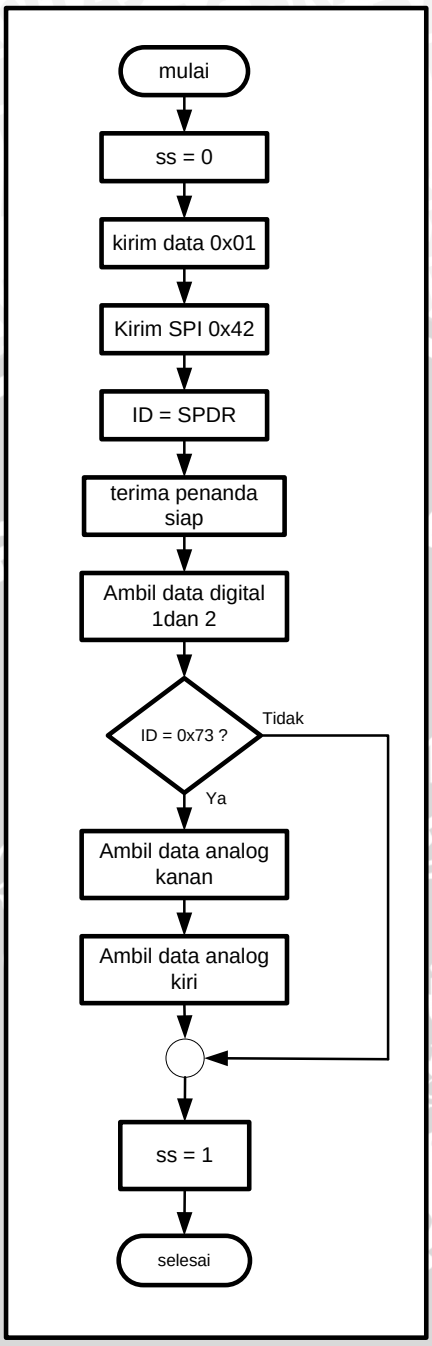
Gambar 4.12 Diagram alir program utama *master*

Langkah pertama dalam diagram alir di atas adalah inisialisasi sistem untuk mengenali port, timer, komunikasi SPI dan lain – lain yang akan digunakan. Setelah itu menjalankan fungsi baca data SPI dari *joystick.* Data *joystick* ini masih berupa data digital nilai ADC dari tombol analog *joystick* kemudian dikonversi menjadi sudut arah gerak tombol analog. Apabila tombol analog digerakkan maka nilai penanda_stop = 0 dan apabila tombol analog tidak digerakkan maka nilai penanda_stop=1. Saat penanda_stop = 0 maka program menjalankan fungsi untuk

menghitung kinematika robot dengan masukan sudut arah gerak tombol analog. Hasil dari perhitungan kinematika ini berupa kecepatan tiap roda yang kemudian dikirim ke tiap *slave* sebagai *set point* kecepatan roda. Arah putaran roda diatur dari logika pada mirokontroler master bergantung pada kecepatan roda hasil perhitungan kinematika. Apabila bernilai negatif maka roda akan berputar ke kanan (searah jarum jam) dan apabila bernilai positif maka roda berputar ke kiri (berlawanan arah jarum jam). Saat penanda_stop = 1 maka logika arah putaran roda akan bernilai 0 semua agar roda berhenti berputar dan mengirim data kecepatan 0 ke mikrokontroler *slave.* Program ini akan terus berjalan hingga mikrokontroler dimatikan.

4.6.3 Diagram Alir Proses Komunikasi dengan *Joystick* Playstation

Program komunikasi dengan *joystick* playstation ini terletak pada mikrokontroler *master*. Diagram alir ini terdiri atas urutan program untuk komunikasi SPI antara mikrokontroler master dengan *joystick* playstation sebagai *slave.* Hasil dari komunikasi ini mikrokontroler mendapat data tombol dari *joystick* playstation. Semua data dari *joystick* diterima oleh mikrokontroler tetapi hanya data tombol analog yang diproses oleh mikrokontroler dan digunakan sebagai input sudut arah gerak robot. Gambar 4.13 menunjukkan diagram alir komunikasi antara mikrokontroler dengan *joystick* playstation.



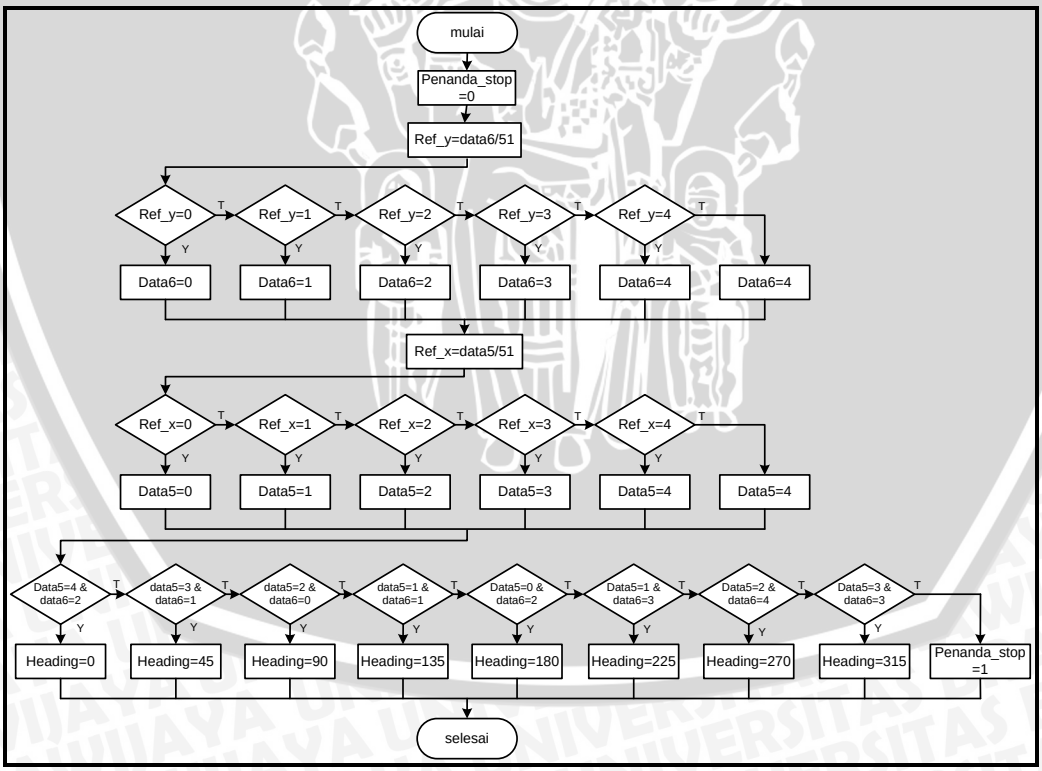
Gambar 4.13 Diagram alir Komunikasi Mikrokontroler Dengan *joystick*

Diagram alir ini diawali dengan memberikan logika 0 pada SS agar *joystick* aktif sebagai *slave*. Protokol komunikasi SPI antara mikrokontroler (MK) dengan *joystick* ditunjukkan pada tabel 4.1. Untuk mengawali komunikasi mikrokontroler mengirim data 0x01, lalu MK mengirim data 0x42 untuk membaca data dari joystick yang menunjukkan tipe *joystick.* Data dari *joystick* disimpan dalam variable ID. *Joystick* kemudian mengirim data penanda siap, data digital1, data digital 2 data ini tidak digunakan. Setelah itu dilakukan proses untuk

mengenali jenis *joystick*. Apabila nilai ID = 0x73 maka program dilanjutkan lagi dengan menerima data analog kanan-sumbuX, data analog kanan-sumbuY, data analog kiri-sumbuX dan data analog kiri-sumbuY dan disimpan dalam variabel tertentu. Apabila nilai ID ≠ 0x73, menandakan jenis *joystick* tidak sesuai dan tidak dilakukan pengambilan data lagi. Diagram alir diakhiri dengan memberikan nilai SS=1 untuk menghentikan komunikasi dengan *joystick.*

4.6.4 Diagram Alir Konversi Data *Joystick* ke Sudut

Data yang diterima dari joystick masih berupa data digital hasil dari proses ADC pada *joystick.* Agar dapat diterjemahkan menjadi sudut arah tombol analog maka data digital ini harus diproses terlebih dahulu. Dari setiap tombol analog diperoleh data digital untuk posisi sumbu x dan y. Data digital yang diterima mulai dari 0 sampai 255. Untuk mempermudah dalam penentuan sudut arah tombol analog maka nilai digital ini dibagi menjadi 5 golongan dengan cara membagi nilai digital tersebut dengan nilai 51. Diagram alir konversi data ke sudut ditunjukkan dalam Gambar 4.14.



Gambar 4.14 Diagram alir Konversi Data *Joystick* ke Sudut

Diagram alir ini diawali dengan memberikan nilai penanda_stop = 0 sebagai penunjuk ada tidaknya tombol yang ditekan. Setelah itu nilai ADC

sumbuY analog kiri (data6) dibagi dengan 51 untuk membagi ADC tersebut menjadi 5 kelompok. Hasil perhitungan disimpan dalam variabel data6. Nilai ADC sumbuX analog kiri (data5) mengalami proses yang sama dengan data6, hanya hasil perhitungan disimpan dalam data5. Perbandingan nilai dari data 5 dan data 6 dijadikan sebagai penunjuk sudut arah gerak robot. Apabila tidak ada perbandingan nilai yang terdeteksi maka nilai penanda_stop=1 sebagai penanda bahwa tidak ada tombol yang ditekan.

4.6.5 Diagram Alir Perhitungan Kinematika Robot

Masukan persamaan perhitungan kinematika ini berupa sudut arah gerak robot dari *joystick* dan hasil dari perhitungan ini berupa kecepatan masing-masing roda. Rumusan kinematika robot *three omni-directional* secara umum adalah sebagai berikut :

$$V_1 = -\frac{1}{3}\cos\Psi + \frac{1}{\sqrt{3}}\sin\Psi + L/3\,V_\emptyset$$

$$V_2 = -\frac{1}{3}\cos\Psi - \frac{1}{\sqrt{3}}\sin\Psi + L/3\,V_\emptyset$$

$$V_3 = 2/3\cos\Psi + L/3\,V_\emptyset$$

Hasil dari perhitungan rumus kinematika ini berupa perbandingan kecepatan tiap-tiap roda. Perbandingan ini dikalikan dengan nilai kecepatan tertentu untuk memperoleh kecepatan roda yang sebenarnya dalam rotasi per menit (rpm). Data kecepatan ini dari rpm dikonversi menjadi pulsa per waktu *interrupt* agar sesuai dengan data kecepatan rotary pada mikrokontroler *slave*. Untuk mengkonversi harus diketahui terlebih dahulu waktu untuk tiap *interrupt* pada mikrokontroler *slave.* Mikrokontroler slave menggunakan *timer2 overflow* interrupt dengan *clock value* = 15.625 KHz sehingga waktu tiap interrupt.

$$t_{interupt} = \frac{1}{15.625} \times 255 = 0{,}01632\ s = 16{,}32\ ms$$

Untuk mengkonversi menit menjadi waktu interrupt maka harus dibagi dengan konstanta tertentu dengan nilai

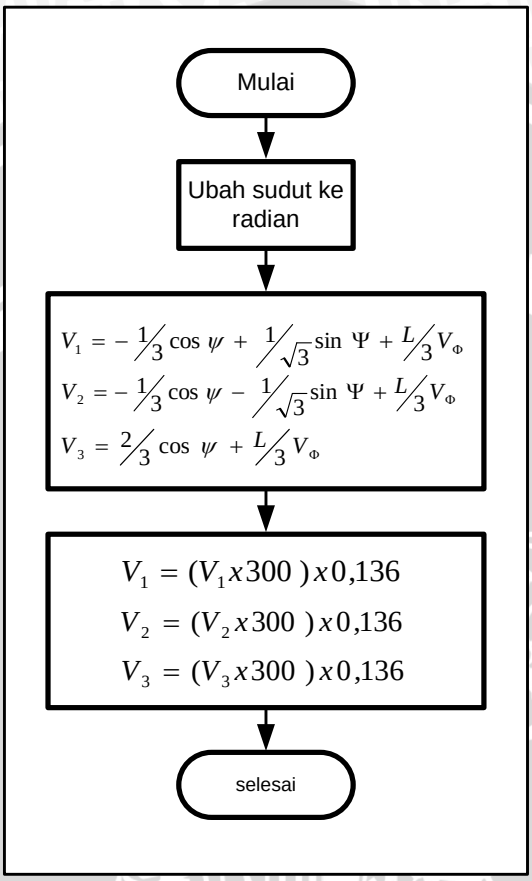
$$K_t = \frac{60\ s}{0{,}01632\ s} = 3676{,}47$$

Untuk mengubah nilai rotasi menjadi pulsa maka harus dikalikan dengan 500, karena sensor rotary encoder yang dipakai memiliki 500 pulsa dalam satu

putaran. Jadi untuk konversi dari rpm menjadi pulsa per *interrupt* harus dikalikan dengan konstanta dengan nilai

$$K = \frac{500}{3676{,}47} = 0{,}136$$

Gambar 4.15 menunjukkan diagram alir perhitungan kinematika robot.



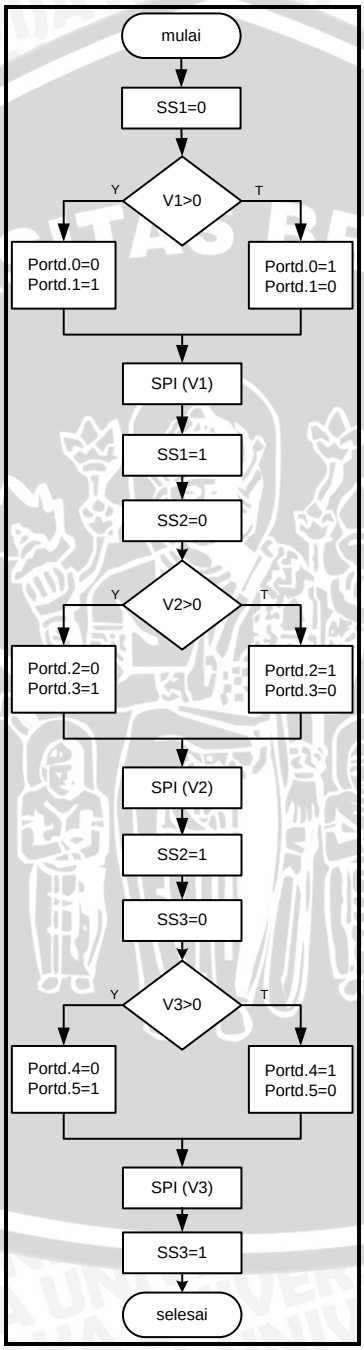
Gambar 4.15 Diagram alir Perhitungan Kinematika Robot

Diagram alir ini diawali dengan mengubah data sudut yang diperoleh menjadi nilai radiannya karena mikrokontroler hanya dapat menghitung dalam bentuk radian. Setelah itu perhitungan rumus kinematika dan menghasilkan perbandingan kecepatan putar tiap roda. Nilai kecepatan roda ini dikalikan dengan 300 rpm kemudian dikonversi menjadi pulsa per *interrupt* sebelum dikirim ke *slave.*

4.6.6 Diagram Alir Pengiriman Data ke Mikrokontroler *Slave*

Diagram alir pengiriman data berfungsi untuk mengatur arah gerak putaran motor dan mengirim data kecepatan tiap roda ke masing–masing *slave* dengan komunikasi SPI. Untuk mengatur arah gerak motor diatur melalui pin

pada mikrokontroler *master*. Saat kecepatan bernilai positif maka motor maju, sedangkan saat kecepatan bernilai negatif maka motor berputar sebaliknya. Tiap SS *slave* dihubungkan dengan pin – pin tertentu pada mikrokontroler *master* untuk menentukan *slave* mana yang menerima data. Gambar 4.16 menunjukkan diagram alir pengiriman data.



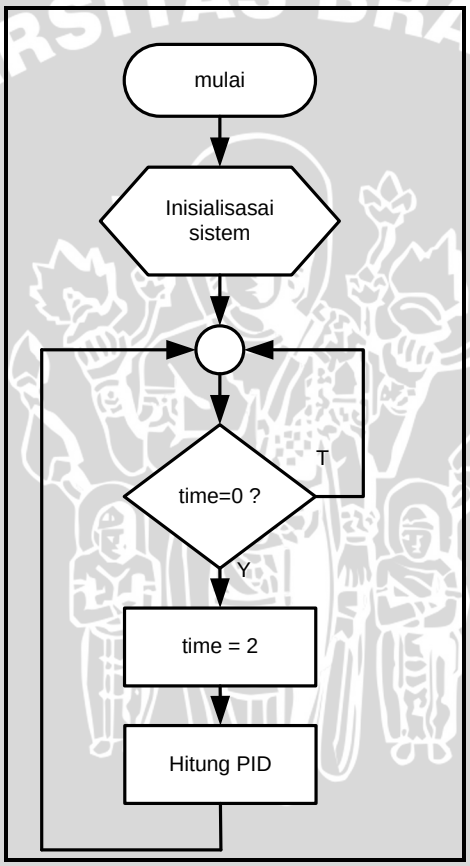
Gambar 4.16 Diagram alir Pengiriman Data ke Mikrokontroler *Slave*

Diagram alir ini diawali dengan memberikan nilai SS1=0 untuk berkomunikasi dengan *slave* 1. Setelah itu diatur port pada MK *master* untuk

mengatur arah gerak motor. Data kecepatan dikirim ke *slave* dan setelah pengiriman data diberikan nilai SS1=1 untuk menghentikan komunikasi dengan slave 1. Proses yang sama dilakukan pada slave 2 dan 3. Untuk *slave* 2 digunakan variabel SS2 dan untuk *slave* 3 digunakan variabel SS3.

4.6.7 Perancangan Program Mikrokontroler S*lave*

Program utama mikrokontroler *slave* dirancang untuk melakukan proses komunikasi dengan mikrokontroler *master* dengan SPI, membaca data kecepatan rotary dan melakukan proses perhitungan PID. Gambar 4.17 menunjukkan diagram alir dari program utama mikrokontroler *slave.*



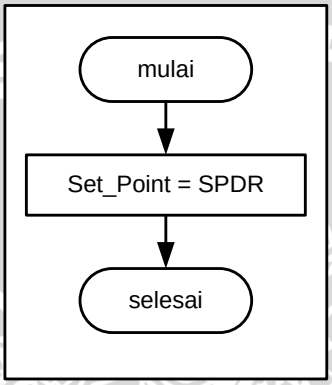
Gambar 4.17 Diagram alir program utama mikrokontroler *slave*

Diagram alir ini diawali dengan inisialisasi sistem untuk mengenali port, timer, komunikasi SPI dan lain – lain yang akan digunakan. Proses selanjutnya adalah melihat nilai pada variabel “time”. Apabila nilai time ≠ 0 maka akan terus dilakukan pengecekan hingga nilai time = 0 untuk masuk ke program berikutnya. Untuk awal nilai time = 0, sehingga proses akan berlanjut dengan memberi nilai

pada variable time = 2 dan menghitung PID. Nilai dari time akan berkurang tiap kali terjadi interrupt.

4.6.8 Diagram Alir *Interrupt* SPI

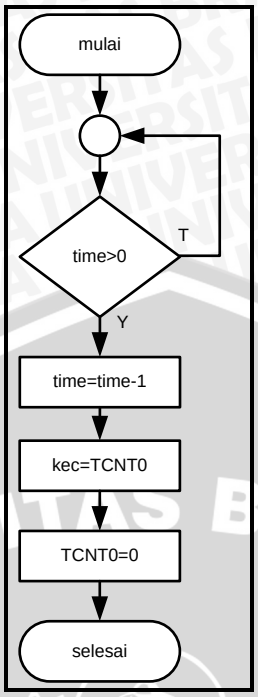
Diagram alir *interrupt* SPI ini terjadi di mikrokontroler *slave*. Diagram alir ini berfungsi untuk mengambil data kecepatan motor yang dikirim oleh mikrokontroler master sebagai set point pada perhitungan PID. Data dari komunikasi SPI ini terletak pada register SDPR. Gambar 4.18 menunjukkan diagram alir *interrupt* SPI.



Gambar 4.18 Diagram alir *interrupt* SPI

4.6.9 Diagram Alir timer2_overflow

Diagram alir ini berfungsi untuk mengambil sampling kecepatan rotary dari TCNT0 tiap kali *interrupt*. Waktu untuk tiap interrupt adalah 16,32 ms. Data kecepatan rotary diambil dari register TCNT0. Gambar 4.19 menunjukkan diagram alir timer2_overflow.



Gambar 4.19 Diagram alir subrutin timer2_overflow

Diagram alir ini diawali dengan melihat nilai variabel time terlebih dahulu. Apabila nilai variabel time lebih dari 0 maka nilai dari variable time mengalami proses pengurangan dan dilanjutkan dengan mengambil data kecepatan rotary dari register TCNT0.

4.6.10 Diagram Alir Perhitungan PID

Diagram Alir perhitungan PID berisi program untuk proses perhitungan kontroler PID. PID adalah suatu perhitungan matematis yang terdiri dari proporsional, integral, dan derivative yang ditambahkan secara bersama-sama dan menghasilkan suatu nilai yang digunakan untuk mengendalikan robot. Rumusan PID secara umum adalah sebagai berikut:

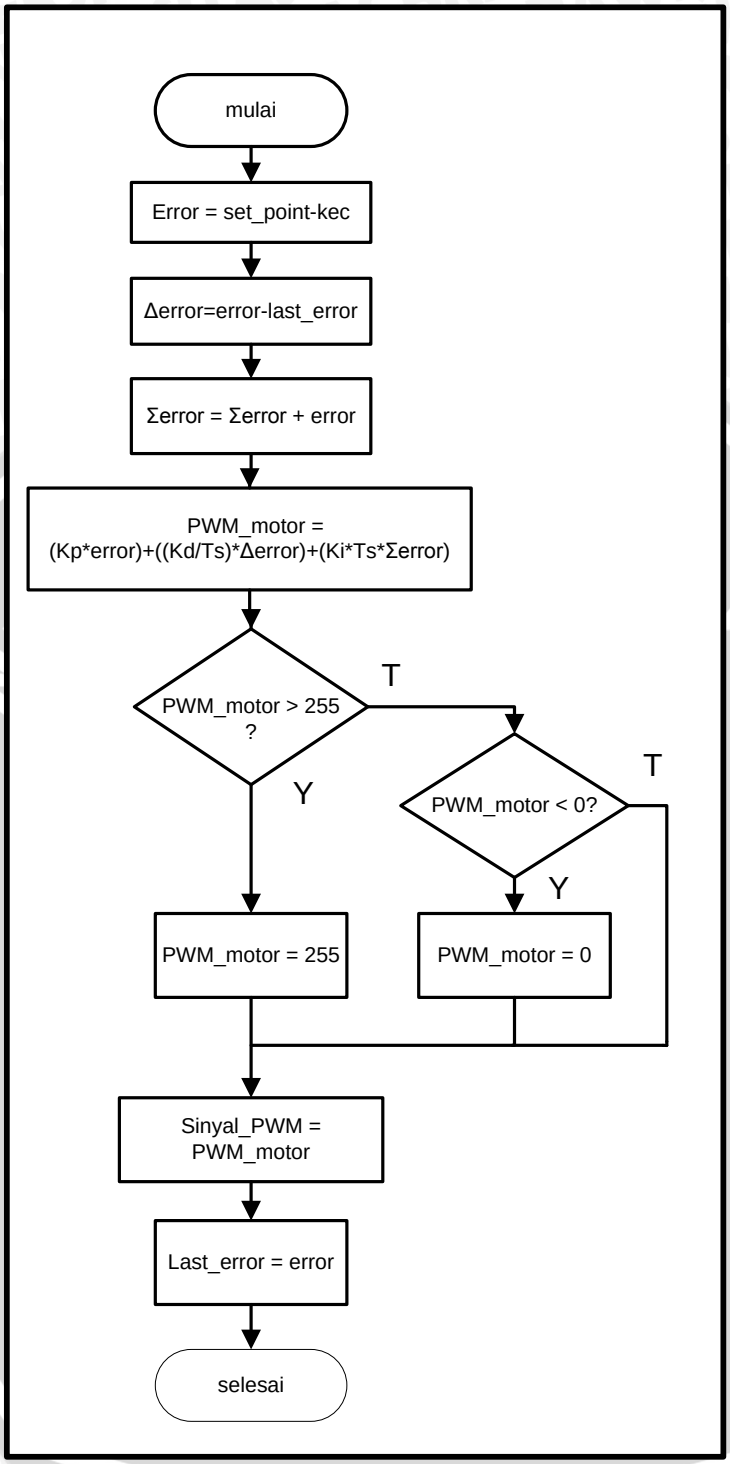
$$\text{Output(t)} = \text{Kp.error(t)} + \int Ki.error(t)dt + \text{Kd.}\frac{d\ error(t)}{dt}$$

Dalam aplikasi PID sebagai kontrol kecepatan, rumusan tersebut didekati dengan:

$$\text{Output(t)=Kp.error(t)+(Ki.Ts).}\sum error(t)\text{+(Kd/Ts).(error(t-1)-error(t))}$$

dengan $$\sum error(t) = \sum error(t-1) + error(t)$$

Nilai s*et point* adalah nilai yang harus dicapai bergantung pada nilai kecepatan yang dikirim oleh mikrokontroler *master*. Gambar 4.20 menunjukkan diagram alir perhitungan PID.



Gambar 4.20 Diagram Alir Perhitungan PID

Diagram alir ini diawali dengan mencari nilai error, ∑error dan Δerror yang digunakan pada perhitungan PID. Hasil dari perhitungan PID ini berupa nilai PWM. Nilai PWM dibatasi antara 0 – 255.

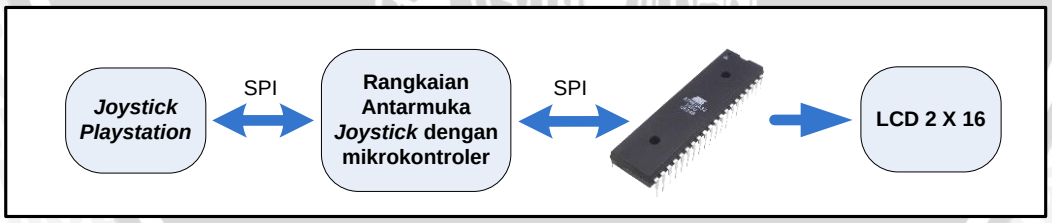
BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui apakah sistem telah bekerja sesuai perancangan yang telah dilakukan. Pengujian dilakukan per blok sistem kemudian secara keseluruhan. Adapun pengujian yang dilakukan sebagai berikut:

1) Pengujian data *Joystick* Playstation
2) Pengujiaan komunikasi SPI antara mikrokontroler *master* dan *Slave*
3) Pengujian sensor Rotary Encoder
4) Pengujian respon kontroler PID
5) Pengujian Kecepatan Putaran Roda
6) Pengujian keseluruhan sistem

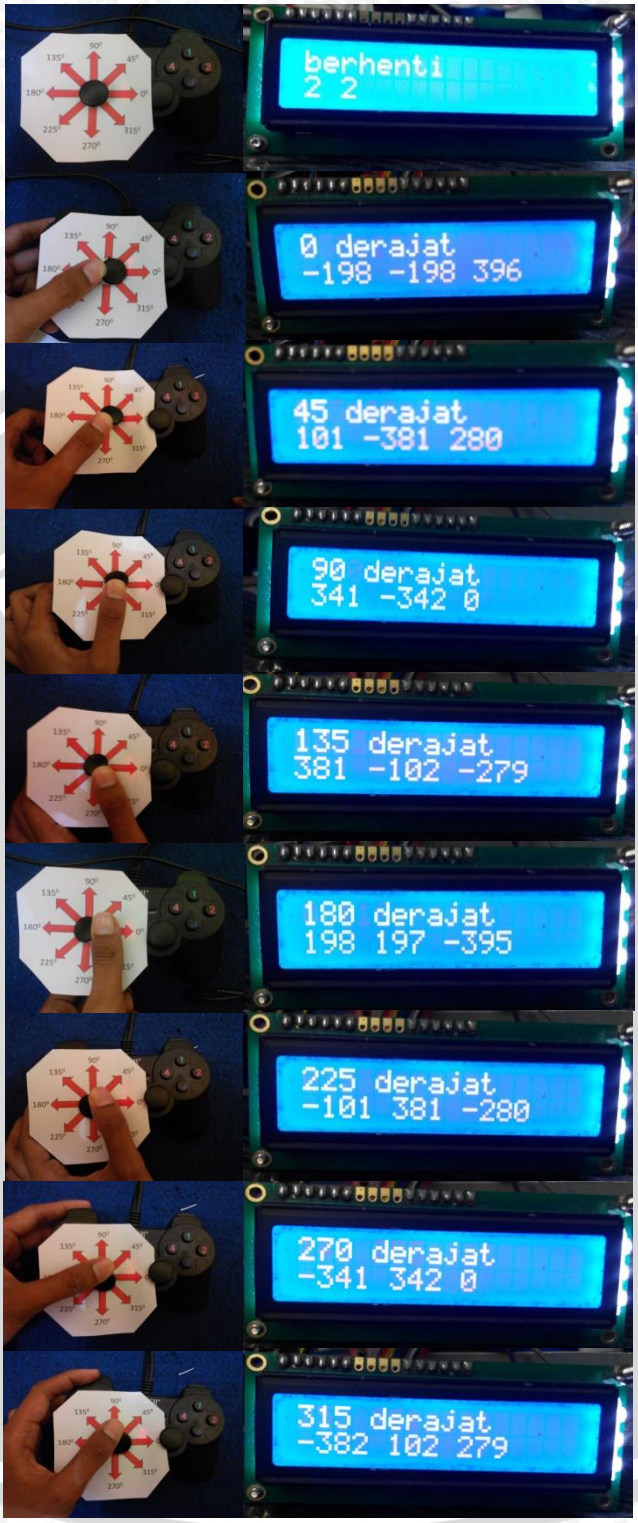
5.1 Pengujian Data *Joystick* Playstation

Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah pembacaan sudut yang dibaca oleh mikrokontroler sesuai dengan sudut pergerakan yang dilakukan pada tombol analog *joystick*. Prosedur pengujian dilakukan dengan menghubungkan *joystick* dengan minimum sistem mikrokontroler ATMega32 dengan menggunakan rangkaian antarmuka *joystick* dengan mikrokontroler. Sudut yang terbaca oleh mikrokontroler ditampilkan pada LCD 2 x 16. Diagram blok pengujian ini ditunjukkan dalam Gambar 5.1.



Gambar 5.1 Diagram Blok Pengujian *Joystick* Playstation

Pada Pengujian ini, tombol analog digerakkan ke sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0. Perangkat *joystick* akan mengirimkan data konversi ADC tombol analog baik sumbu X maupun Y tombol analog melalui komunikasi SPI ke mikrokontroler. Mikrokontroler mengolah data yang didapat menjadi sudut arah tombol analog dan ditampilkan pada LCD. Hasil pengujian ditunjukkan dalam Gambar 5.2.

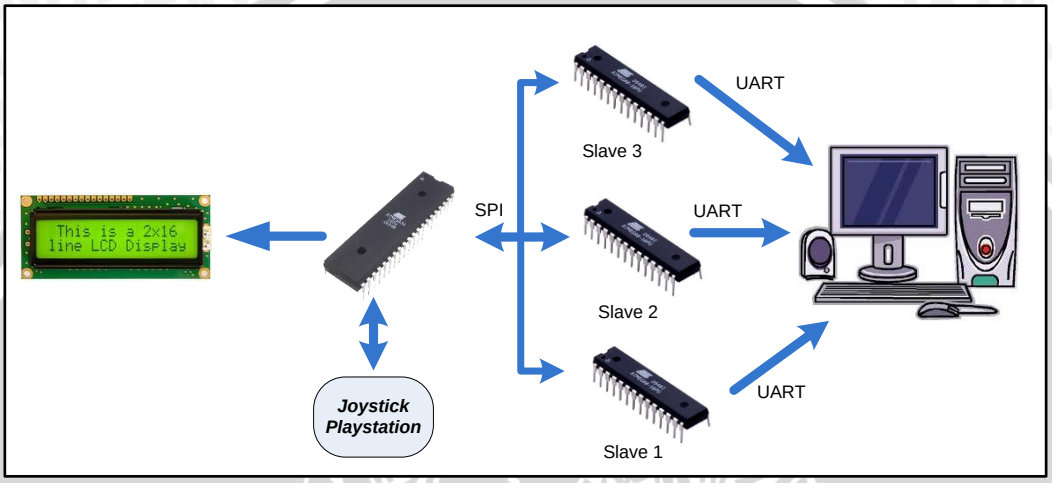


Gambar 5.2 Arah Tombol Analog dan Tampilan Pada LCD

Dari hasil pengujian joystick terlihat bahwa joystick dapat mengirimkan data ke mikrokontroler. Mikrokontroler dapat mengolah data digital yang diperoleh dari *joystick* menjadi sudut sesuai dengan arah tombol analog. Data sudut yang ditampilkan pada LCD sesuai dengan arah tombol analog digerakkan.

5.2 Pengujian Komunikasi SPI Antara Mikrokontroler *Master* dan *slave*

Pengujian ini dilakukan untuk mengetahui dan memeriksa perangkat lunak yang disusun dalam perangkat mikrokontroler apakah sudah dapat menangani komunikasi SPI antara mikrokontroler *master* (ATmega32) dengan mikrokontroler *slave* (ATmega8). Prosedur Pengujian dilakukan dengan menghubungkan LCD, mikrokontroler ATMega32, mikrokontroler ATMega8, *Joystick* Playstation, dan komputer sesuai dengan Gambar 5.3.



Gambar 5.3 Diagram Alir Pengujian SPI MK *master-slave*

Pada pengujian komunikasi SPI ini mikrokontroler *master* akan mengirimkan data sesuai dengan arah tombol analog ditekan. Sudut ini menjadi masukan mikrokontroler *master* untuk menentukan kecepatan tiap roda. Data kecepatan ini kemudian dikirim ke masing-masing *slave* melalui komunikasi SPI secara paralel lalu *slave* mengirim data ini ke computer melalui komunikasi UART. Pengiriman data pada komunikasi UART tidak dapat dilakukan bersama sehingga pengambilan data dilakukan bergantian. Tabel 5.1 menunjukkan hasil pengujian komunikasi SPI dari tiga kali percobaan.

Tabel 5.1 Hasil Pengujian komunikasi SPI *master – slave*

Arah tekan tombol analog	**Nilai pada LCD**			**Nilai pada Komputer**		
	V_1	V_2	V_3	*Slave* 1	*Slave* 2	*Slave* 3
0	99	99	198	99	99	198
90	170	170	0	170	170	0
180	99	99	198	99	99	198

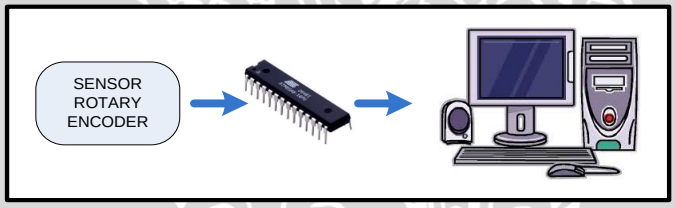
Dari hasil pengujian komunikasi SPI (*Serial Peripheral Interface*) terlihat bahwa mikrokontroler *master* dapat mengirimkan data konstanta ke mikrokontroler *slave*. Data yang ditampilkan pada mikrokontroler *slave di komputer* sama dengan data yang ditampilkan oleh mikrokontroler *master* pada LCD.

5.3 Pengujian Sensor Rotary Encoder

Pengujian ini bertujuan untuk melihat apakah sensor rotary encoder yang dipakai dapat menghasilkan pulsa sesuai dengan datasheet dari rotary encoder dan untuk melihat apakah rotary encoder dapat berfungsi sebagai sensor kecepatan roda.

5.3.1 Pengujian Keluaran Sensor Rotary Encoder

Pada datasheet, rotary encoder dapat menghasilkan 500 pulsa dalam satu putaran. Prosedur pengujian dilakukan dengan menghubungkan antara rotary encoder, mikrokontroler ATMega8 dan komputer sesuai dengan Gambar 5.4



Gambar 5.4 Diagram Blok Pengujian Keluaran Sensor Rotary Encoder

Pada pengujian ini , sensor rotary encoder diputar sebesar ¼, ½, ¾, dan 1 putaran. Pada masing-masing putaran dilihat berapa pulsa yang terbaca oleh mikrokontroler. Jumlah pulsa yang terbaca ditampilkan pada komputer. Hasil pengujian yang diperoleh dari beberapa pengambilan data ditunjukkan pada Tabel 5.2.

Tabel 5.2 Hasil Pengujian data pulsa rotary encoder

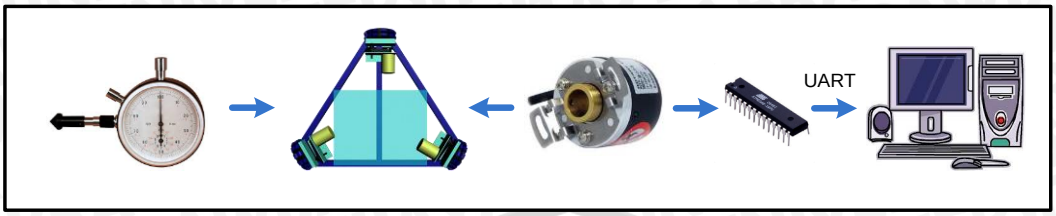
Pengujian ke -	Besar Putaran	Jumlah Pulsa Rotary encoder 1	Jumlah Pulsa Rotary encoder 2	Jumlah Pulsa Rotary encoder 3
1	1/4	125	125	127
		125	125	125
		126	125	126

Pengujian ke -	Besar Putaran	Jumlah Pulsa Rotary encoder 1	Jumlah Pulsa Rotary encoder 2	Jumlah Pulsa Rotary encoder 3
2	1/2	247	250	250
		250	250	249
		248	251	249
3	3/4	375	373	375
		375	174	376
		375	375	375
4	1	500	500	501
		502	498	500
		501	501	500

Berdasarkan Tabel 5.2 dapat diperoleh hasil bahwa kesalahan rata – rata yang terjadi saat pembacaan jumlah pulsa rotary encoder sebesar 0.75 pulsa atau dibulatkan menjadi 1 pulsa. Kesalahan pembacaan terbesar yaitu 3 pulsa. Pada pengujian, kesalahan pembacaan berupa jumlah pulsa yang kurang atau lebih besar dari nilai seharusnya. Kesalahan ini dapat terjadi karena kesalahan dalam memutar rotary encoder. Dari Pengujian ini terlihat bahwa keluaran dari *rotary encoder* dapat dibaca oleh mikrokontroler dan pulsa keluaran dari rotary encoder sesuai dengan *datasheet.*

5.3.2 Pengujian Rotary Encoder Sebagai Sensor Kecepatan Roda

Pengujian dilakukan dengan menghubungkan mikrokontroler *slave* dengan rotary encoder dan komputer melalui UART. Pengukuran kecepatan roda yang sebenarnya menggunakan tachometer analog yang dikopel langsung dengan poros roda. Diagram blok pengujian ditunjukkan dalam Gambar 5.5.



Gambar 5.5 Diagram Blok Pengujian Rotary Encoder Sebagai Sensor Kecepatan Roda

Pengujian dilakukan dengan memberikan nilai PWM tertentu. Kecepatan roda diukur dengan menggunakan rotary encoder dan tachometer. Hasil pengukuran kecepatan roda dengan rotary encoder ditampilkan pada komputer. Hasil pengujian ditunjukkan dalam Tabel 5.3.

Tabel 5.3 Hasil Pengujian Kecepatan Roda dengan Rotary Encoder Dan Tachometer

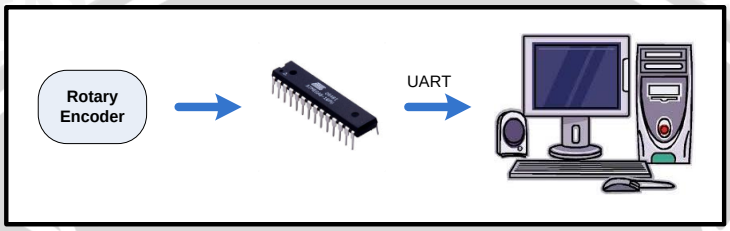
Rotary	PWM	Tacho (rpm)	Rotary (rpm)	kesalahan (%)	Kesalahan Rata-rata (%)
Roda 1	20%	280	260	7,23	1.83
	40%	586	584	0,29	
	60%	748	749	0,07	
	80%	840	844	0,49	
	100%	910	920	1,07	
Roda 2	20%	180	176	2,06	0.8
	40%	390	392	0,51	
	60%	454	454	0,00	
	80%	490	485	1,05	
	100%	510	508	0,33	
Roda 3	20%	280	270	3,57	2.01
	40%	630	618	1,98	
	60%	648	634	2,20	
	80%	700	687	1,86	
	100%	760	757	0,43	

Dari Tabel 5.3 diperoleh kesalahan rata – rata pada roda 1 sebesar 1,83%, roda 2 sebesar 0,8%, dan pada roda 3 sebesar 1,3%. Rotary encoder dapat berfungsi sebagai sensor kecepatan roda dengan kesalahan rata – rata pada semua roda sebesar 1,31%.

5.4 Pengujian Hasil *Tuning* Parameter PID

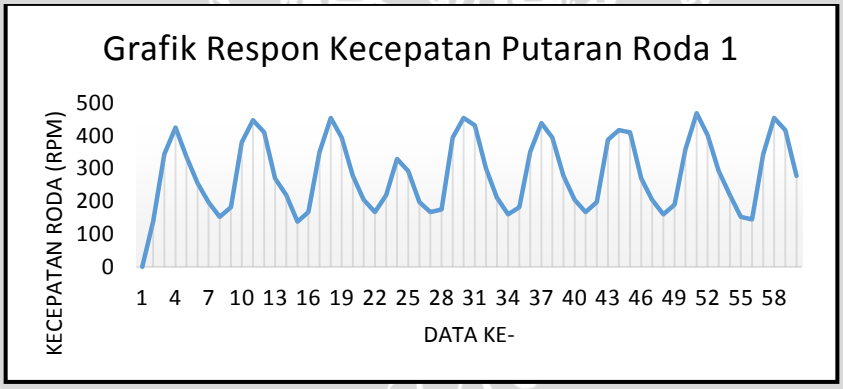
Tujuan dari pengujian ini adalah untuk melihat respon roda dari nilai parameter kontrol PID (Kp, Ki dan Kd) dengan menggunakan metode Osilasi

Ziegler-Nichols yang telah didapat sebelumnya. Prosedur pengujian dilakukan dengan memasukkan nilai parameter yang telah didapat ke persamaan PID pada masing-masing *slave*. Hubungkan mikrokontroler slave ke komputer untuk melihat data kecepatan putaran roda dari sensor rotary encoder yang dikopel langsung dengan roda. Diagram blok pengujian hasil tuning parameter kontroler PID menggunakan metode osilasi Ziegler-Nichols dalam satu sistem mikrokontroler *slave* ditunjukkan dalam gambar 5.6 .



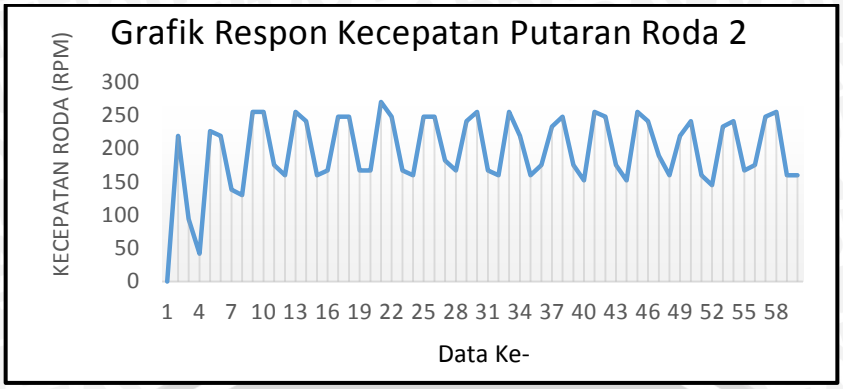
Gambar 5.6 Diagram Blok Pengujian Hasil *Tuning* Parameter Kontroler PID

Dari data yang diterima komputer dapat dilihat grafik respon kecepatan putaran roda. Hasil *tuning* parameter kontroler PID menggunakan metode kedua Ziegler – Nichols pada roda 1 didapat nilai Kp = 6, Ki = 92,31, dan Kd = 0,096. Grafik respon kecepatan putaran roda pada roda 1 ditunjukkan dalam Gambar 5.7.



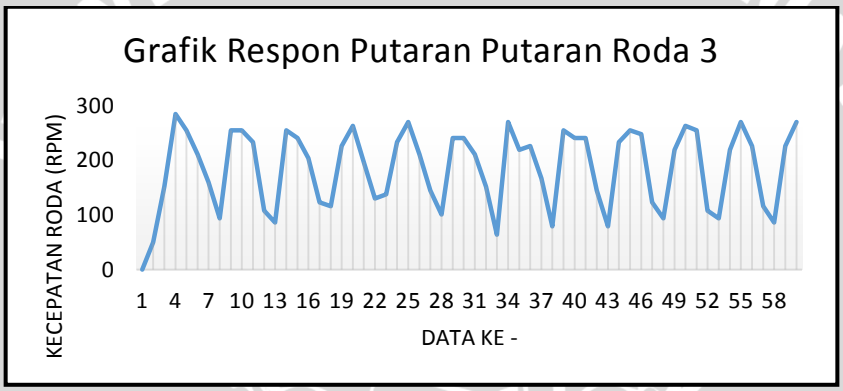
Gambar 5.7 Grafik Respon Kecepatan Putaran Roda 1

Hasil *tuning* parameter kontroler PID pada roda 2 menggunakan metode kedua Ziegler – Nichols didapat nilai Kp = 7,8 , Ki =120, dan Kd = 0,1248. Grafik respon kecepatan putaran roda 2 ditunjukkan dalam Gambar 5.8.



Gambar 5.8 Grafik Respon Kecepatan Putaran Roda 2

Hasil *tuning* parameter kontroler PID pada roda 3 menggunakan metode kedua Ziegler – Nichols didapat nilai Kp = 7,2 , Ki =73,47 , dan Kd = 0,1764. Grafik respon kecepatan putaran roda 3 ditunjukkan dalam Gambar 5.9.



Gambar 5.9 Grafik Respon Kecepatan Putaran Roda 3

Grafik respon pada roda 1, roda 2, dan roda 3 menunjukkan respon yang tidak bisa mencapai stabil. Hal ini membuat nilai Kp, Ki, dan Kd hasil *tuning* parameter PID dengan metode kedua Ziegler – Nichols tidak dapat digunakan. Nilai Kp, Ki dan Kd dicari dengan metode lain yaitu metode *hand tuning.*

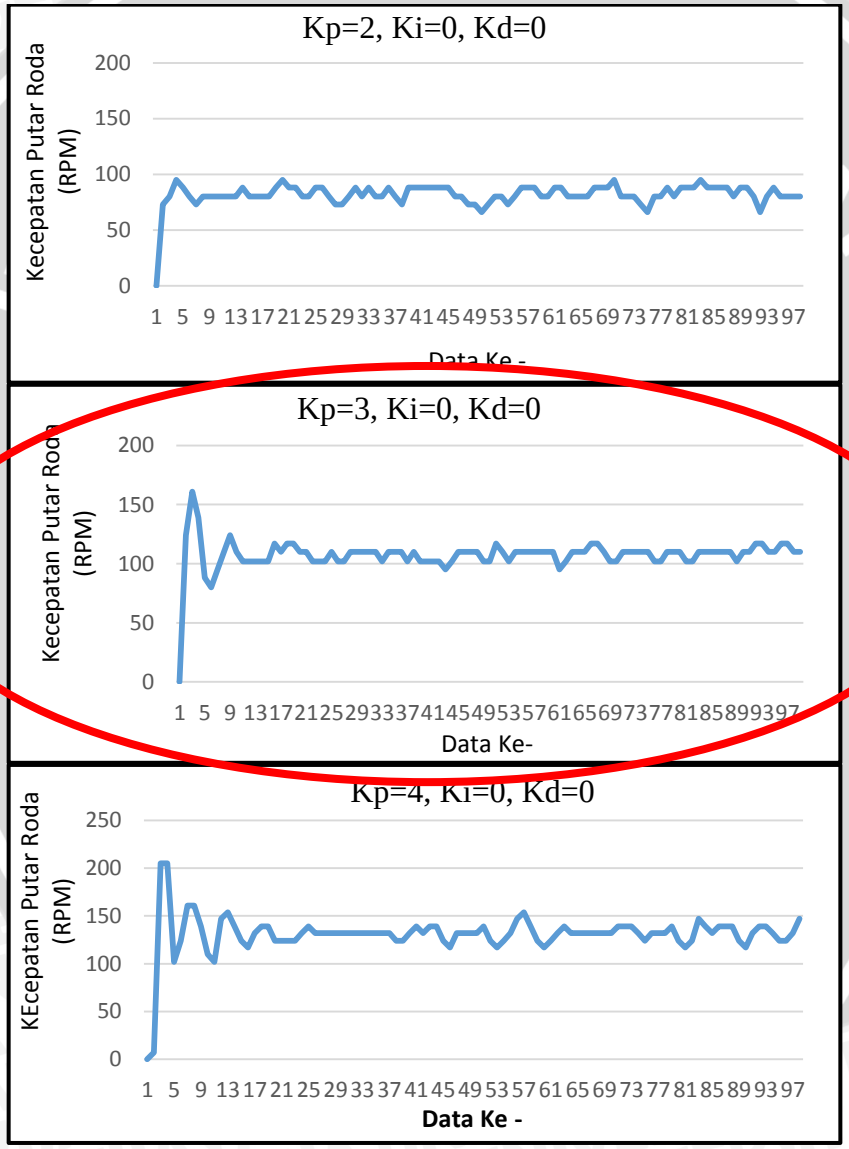
Menurut Smith (1979), untuk melakukan *hand tuning* prosedur yang dilakukan adalah sebagai berikut :

1. Melepaskan kontroler integral dan diferensial dengan memberikan nilai Ki=0 dan Kd=0.
2. Mengatur nilai Kp hingga didapatkan respon yang diinginkan, dengan mengabaikan *offset* dari *setpoint*.
3. Dengan terus menaikkan nilai Kp nilai dari Kd dinaikkan untuk mengurangi *overshoot* yang terjadi.
4. Naikkan nilai Ki untuk mengurangi *offset*.

Dengan menggunakan metode *hand tuning* nilai parameter PID perlu diubah-ubah secara trial dan error agar respon yang diperoleh sesuai dengan harapan. *Set point* yang diinginkan sebesar 200 RPM dan error maksimum yang diharapkan sebesar 20 rpm dari *set point* yang ditetapkan.

5.4.1 Pengujian Tuning Parameter PID Menggunakan Metode *Hand Tuning* pada Roda 1

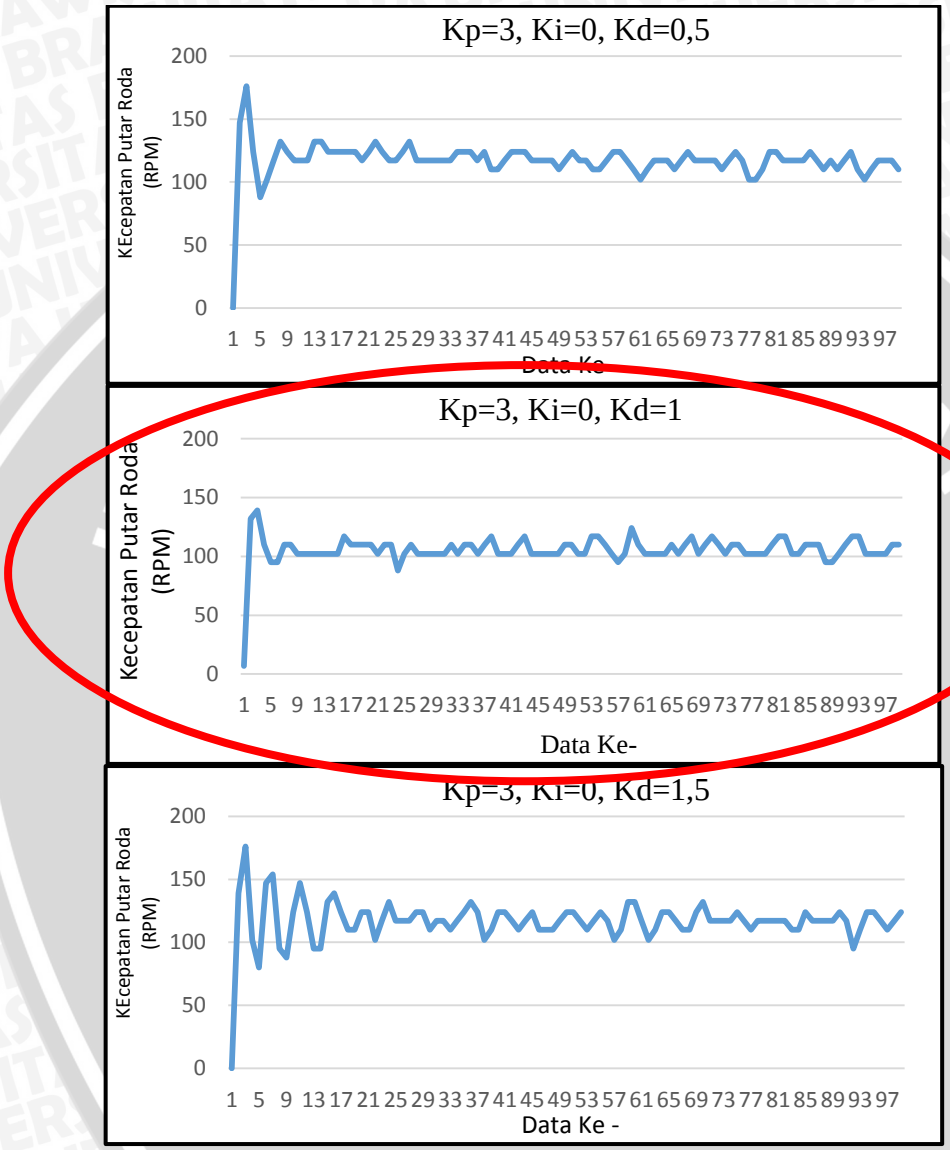
Gambar 5.10 menunjukkan proses penguatan nilai Kp menggunakan metode *hand Tuning* pada roda 1.



Gambar 5.10 Grafik Respon Kecepatan Putaran Roda 1 Untuk Nilai Kp Tertentu

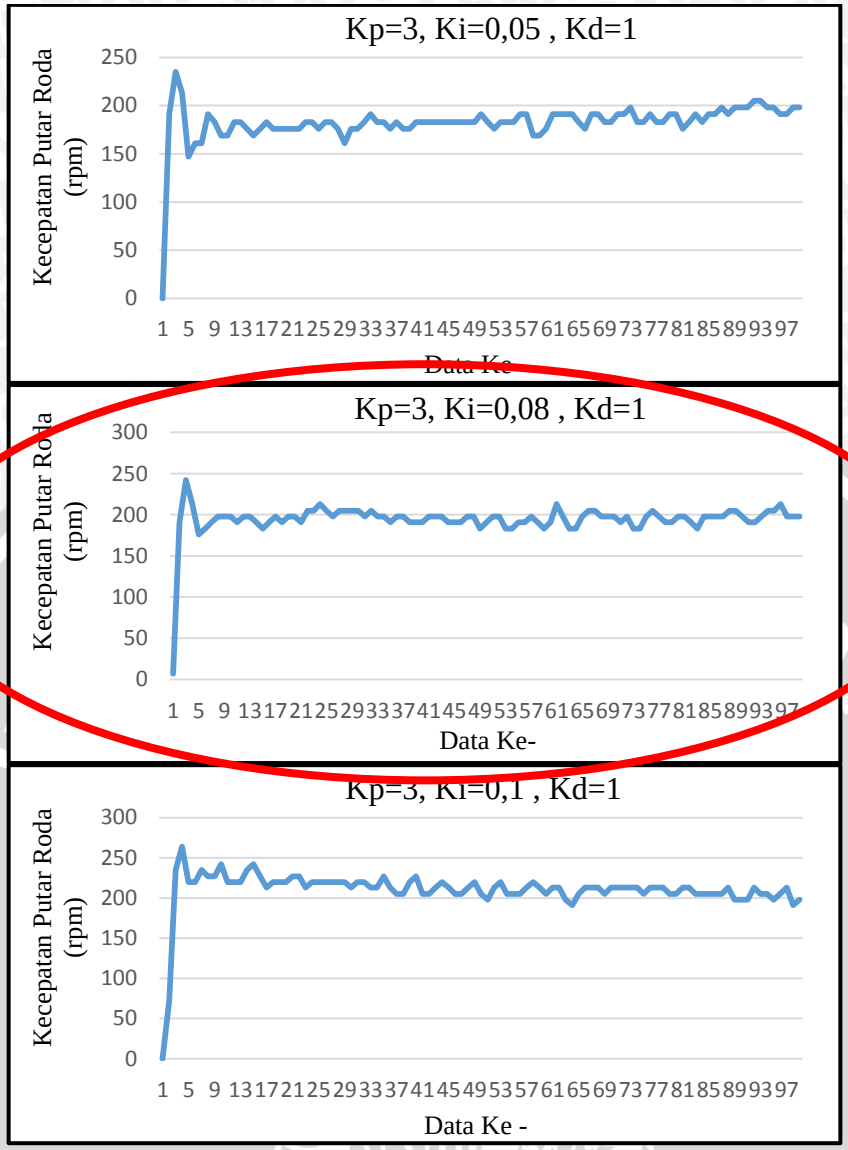
Dari Gambar 5.10 respon untuk Kp = 3 (grafik yang dilingkari warna merah) dapat mencapai stabil walaupun masih terjadi *overshoot* dari titik stabil

sekarang dan akan terjadi osilasi yang lebih besar bila nilai Kp dinaikkan lagi. Untuk mengurangi *overshoot* maka digunakanlah kontroler *derivative*. Grafik respon kecepatan putaran roda 1 dengan kontroler *derivative* ditunjukkan dalam Gambar 5.11.



Gambar 5.11 Grafik Respon Kecepatan Putaran Roda 1 Untuk Nilai Kp dan Kd Tertentu

Grafik respon dalam Gambar 5.11 menunjukkan penggunaan Kd=1 (grafik yang dilingkari warna merah) mengurangi *overshoot* tetapi masih terdapat *offset*. Untuk menghilangkan *offset* yang terjadi maka digunakanlah kontroler integral. Performansi pada penguatan Ki dapat dilihat dalam Gambar 5.12.



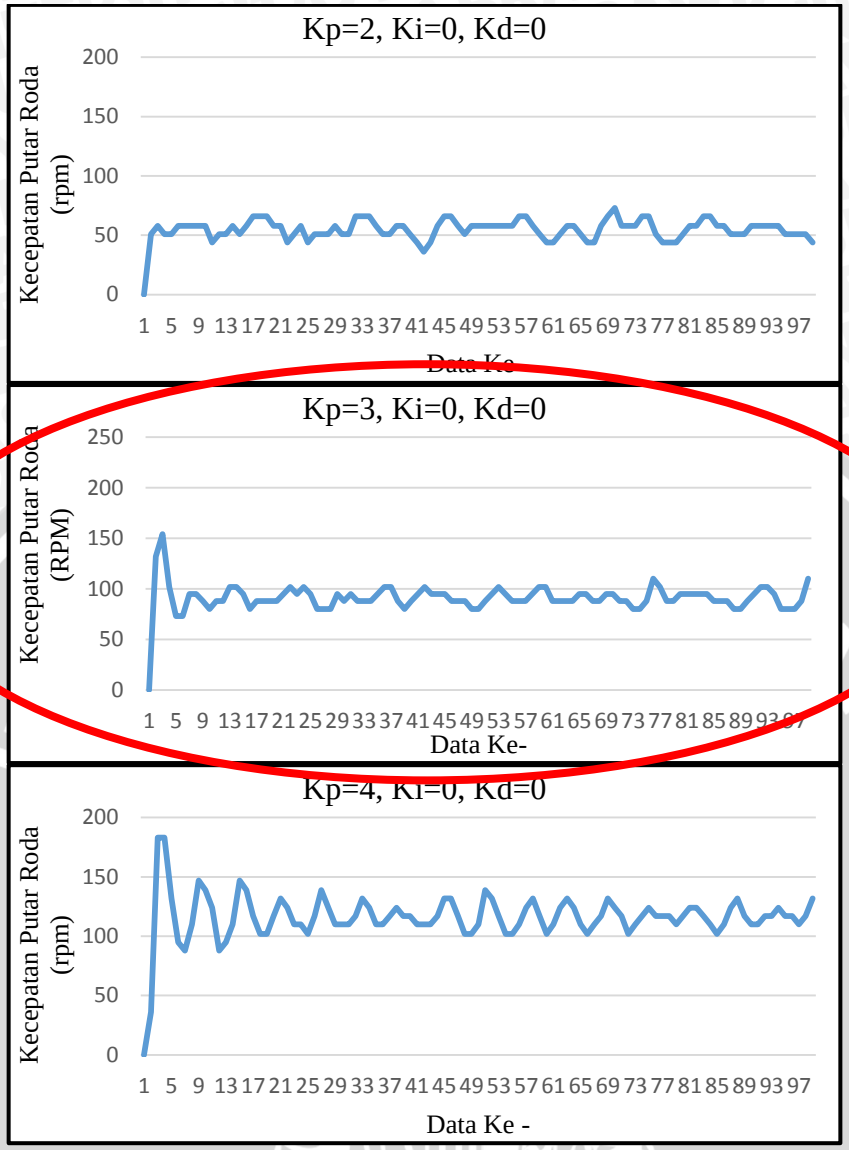
Gambar 5.12 Grafik Respon Kecepatan Putaran Roda 1 untuk nilai Kp, Ki dan Kd tertentu

Gambar 5.12 (pada grafik yang dilingkari warna merah) menunjukkan respon sistem pada roda 1 sudah mampu mencapai *set point* dan *error steady state* yang kurang dari 20 rpm

Dari penentuan nilai penguatan Kp, Ki dan Kd dapat dipastikan nilai penguatan yang digunakan untuk sistem kontrol kecepatan putaran roda 1 robot *three omni-directional* ini adalah Kp = 3, Ki = 0,08, dan Kd = 1.

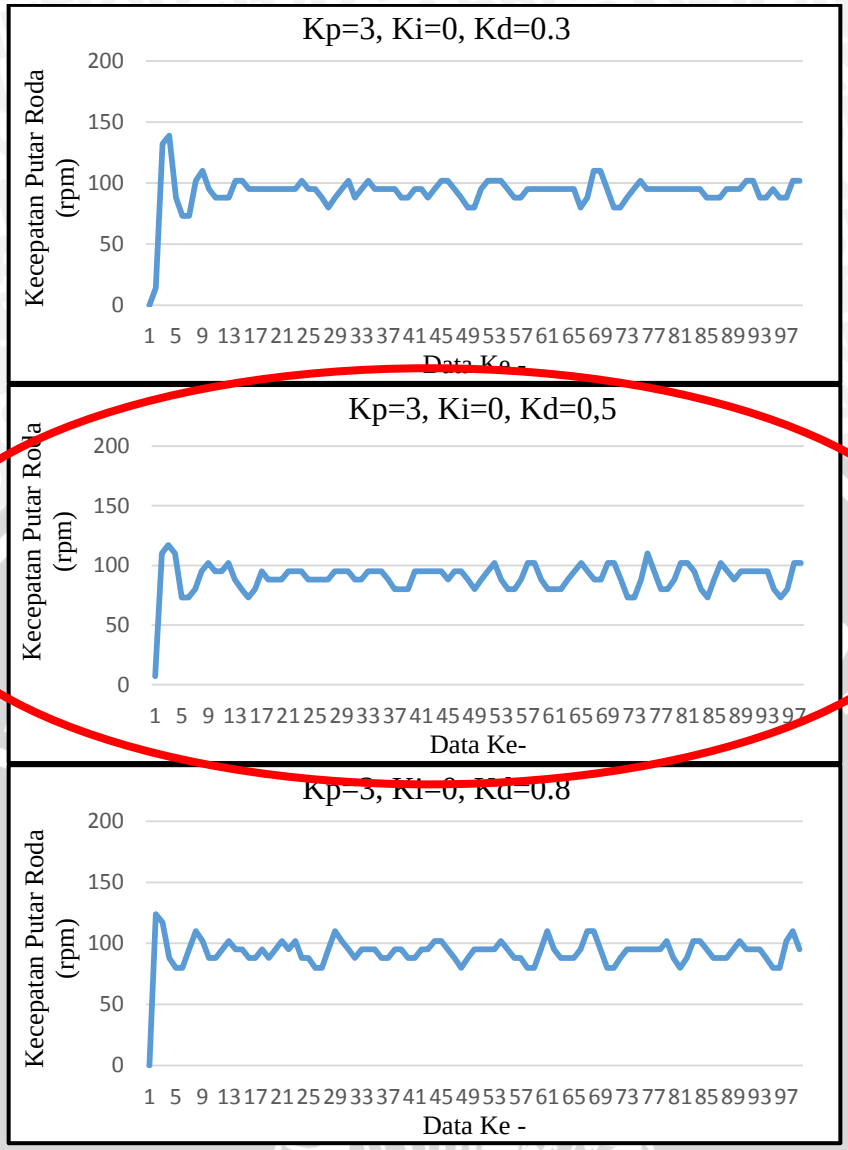
5.4.2 Pengujian Tuning Parameter PID Menggunakan Metode *Hand Tuning* pada Roda 2

Gambar 5.13 menunjukkan proses penguatan nilai Kp menggunakan metode *hand Tuning* pada roda 2.



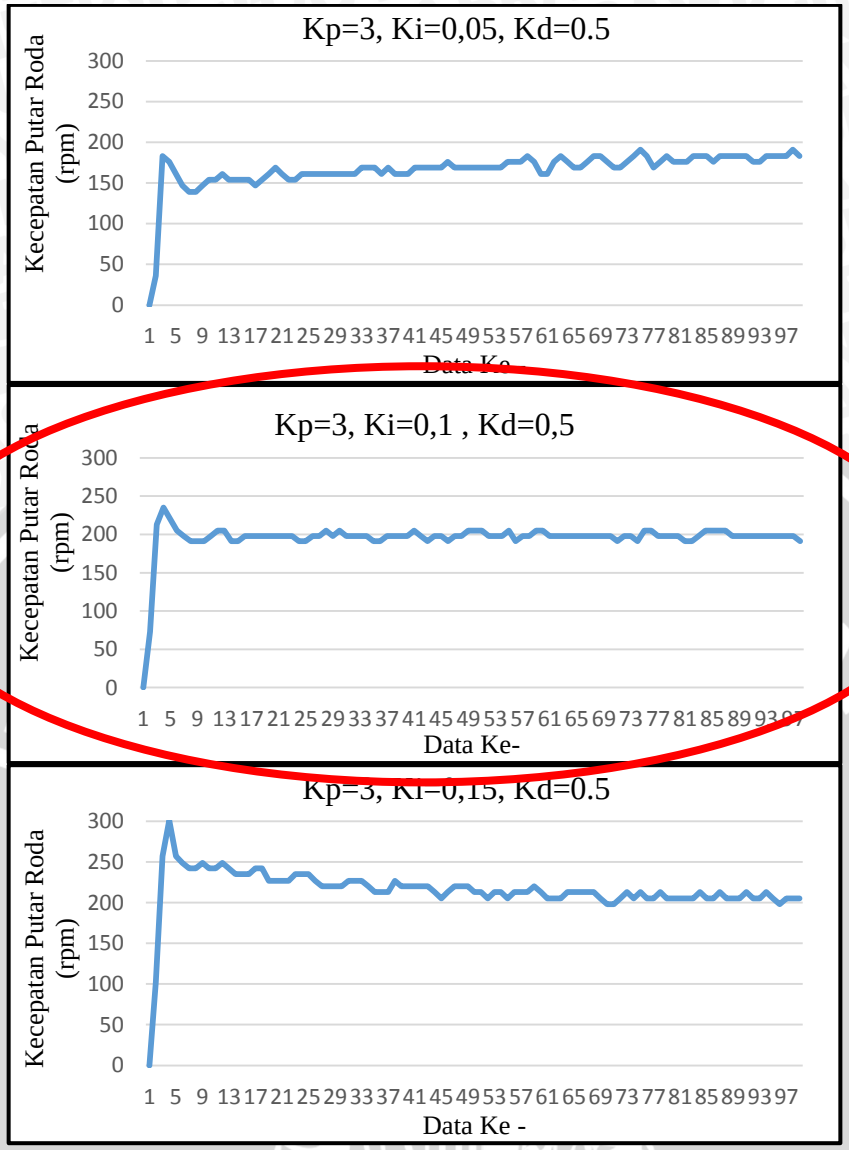
Gambar 5.13 Grafik Respon Kecepatan Putaran Roda 2 untuk nilai Kp Tertentu

Dari Gambar 5.13 respon untuk Kp = 3 (grafik yang dilingkari warna merah) dapat mencapai stabil walaupun masih terjadi *overshoot* dari titik stabil sekarang dan akan terjadi osilasi yang lebih besar bila nilai Kp dinaikkan lagi. Untuk mengurangi *overshoot* maka digunakanlah kontroler *derivative*. Grafik respon kecepatan putaran roda 2 dengan kontroler *derivative* ditunjukkan dalam Gambar 5.14.



Gambar 5.14 Grafik Respon Kecepatan Putaran Roda 2 Untuk Nilai Kp dan Kd Tertentu

Grafik respon dalam Gambar 5.14 menunjukkan penggunaan Kd=0,5 (pada grafik yang dilingkari warna merah) dapat mengurangi *overshoot* tetapi masih terdapat *offset*. Untuk menghilangkan *offset* yang terjadi maka digunakan kontroler integral. Performansi pada penguatan Ki dapat dilihat dalam Gambar 5.15.



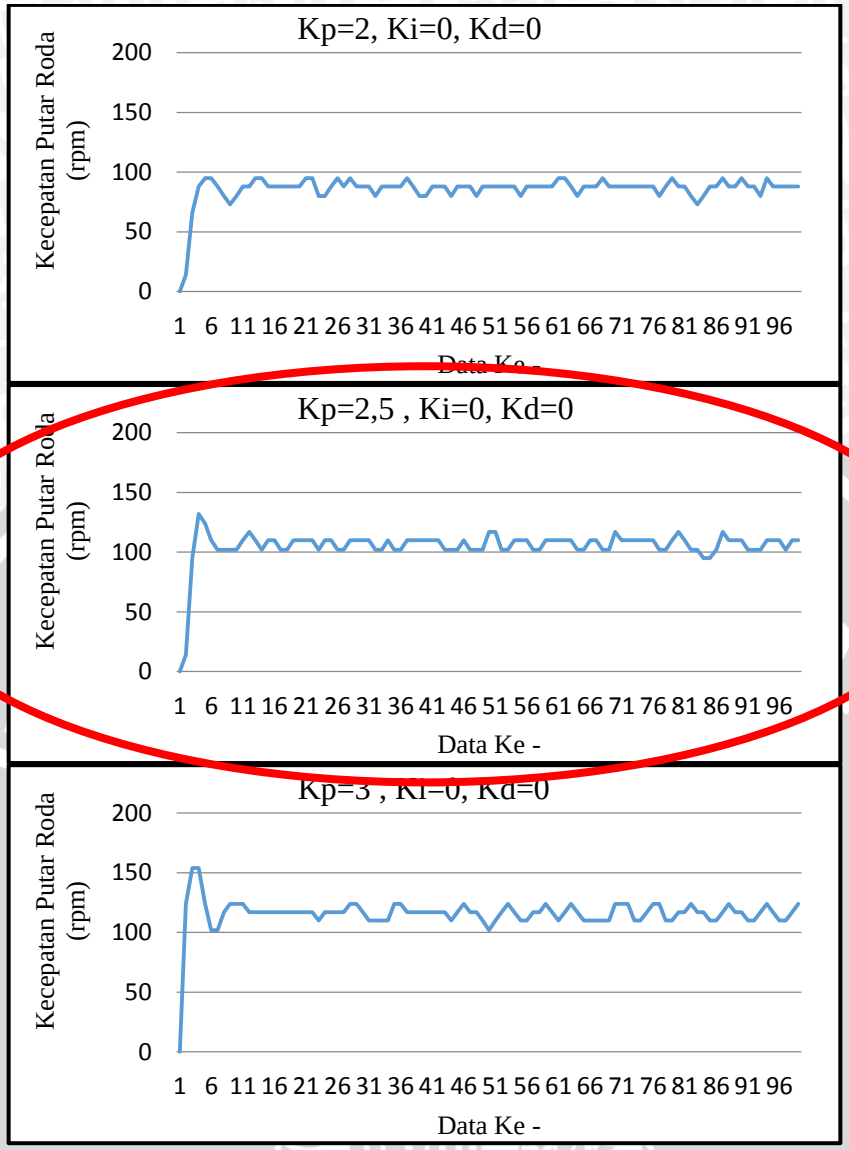
Gambar 5.15 Grafik Respon Kecepatan Putaran Roda 2 Untuk Nilai Kp, Ki dan Kd Tertentu

Gambar 5.15 (pada grafik yang dilingkari warna merah) menunjukkan respon sistem pada roda 2 sudah mampu mencapai *set point* dan *error steady state* yang kurang dari 20 rpm.

Dari penentuan nilai penguatan Kp, Ki dan Kd dapat dipastikan nilai penguatan yang digunakan untuk sistem kontrol kecepatan putaran roda 2 robot *three omni-directional* ini adalah Kp = 3, Ki = 0,1, dan Kd = 0,5.

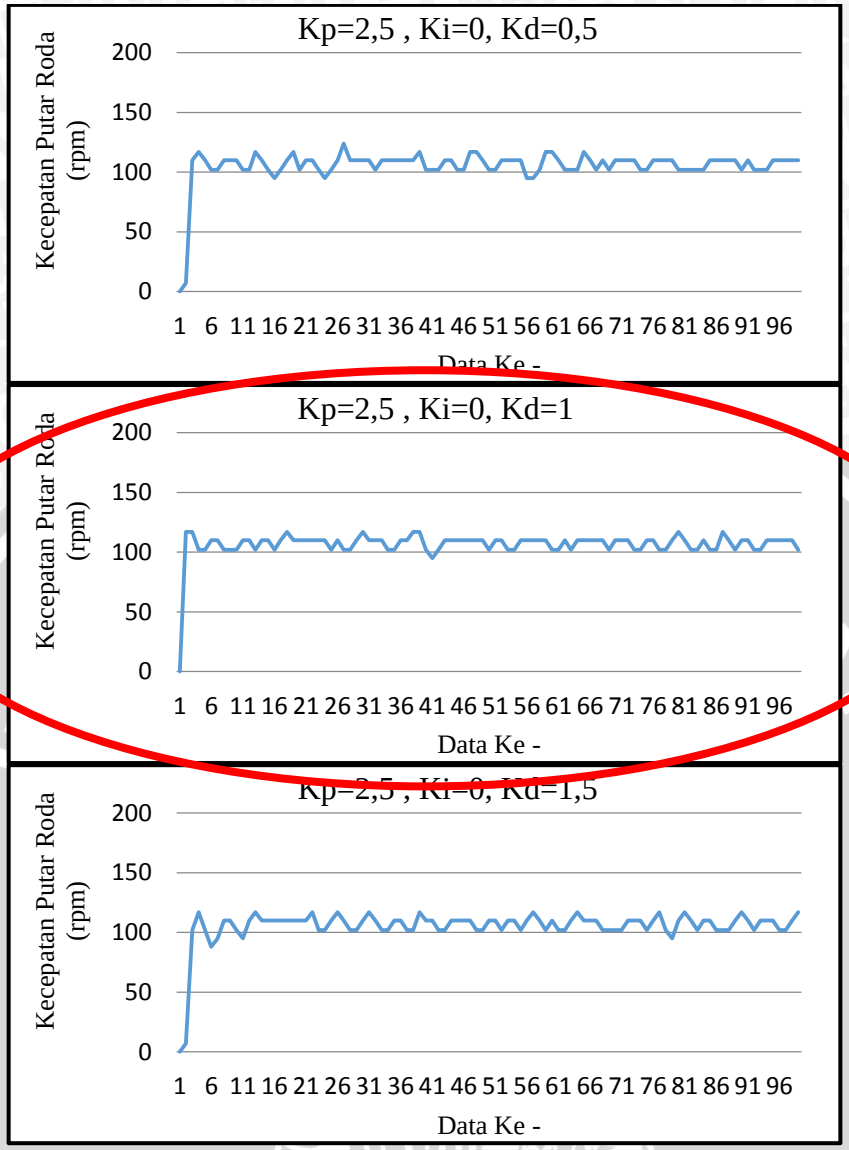
5.4.3 Pengujian Tuning Parameter PID Menggunakan Metode *Hand Tuning* pada Roda 3

Gambar 5.16 menunjukkan proses penguatan nilai Kp menggunakan metode *hand Tuning* pada roda 3.



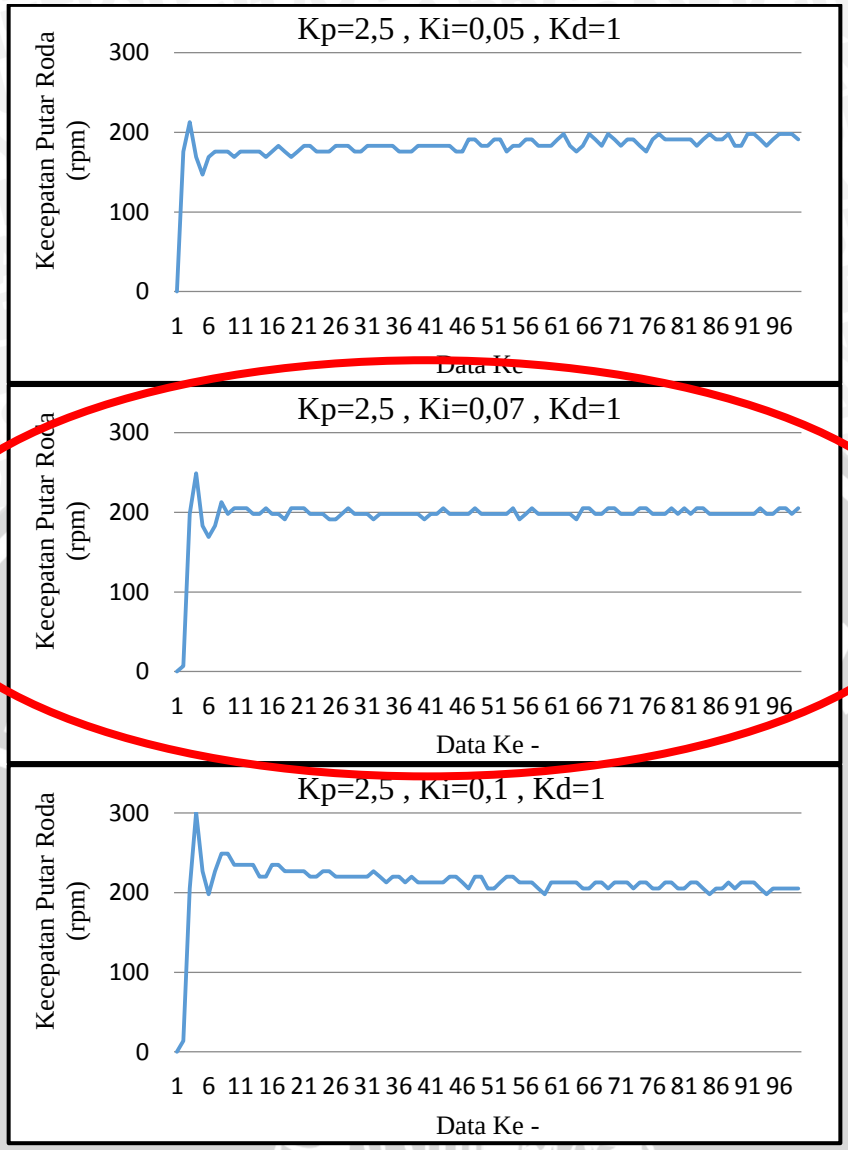
Gambar 5.16 Grafik Respon Kecepatan Putaran Roda 3 untuk nilai Kp Tertentu

Dari Gambar 5.16 respon untuk Kp = 2,5 (pada grafik yang dilingkari warna merah) dapat mencapai stabil walaupun masih terjadi *overshoot* dari titik stabil sekarang dan akan terjadi osilasi yang lebih besar bila nilai Kp dinaikkan lagi. Untuk mengurangi *overshoot* maka digunakanlah kontroler *derivative*. Grafik respon kecepatan putaran roda 3 dengan kontroler *derivative* ditunjukkan dalam Gambar 5.17.



Gambar 5.17 Grafik Respon Kecepatan Putaran Roda 2 untuk nilai Kp dan Kd Tertentu

Grafik respon dalam Gambar 5.17 menunjukkan penggunaan Kd=1 (pada grafik yang dilingkari warna merah) dapat mengurangi *overshoot* dan respon stabil tetapi masih terdapat *offset*. Untuk menghilangkan *offset* yang terjadi maka digunakanlah kontroler integral. Performansi pada penguatan Ki dapat dilihat dalam Gambar 5.18.



Gambar 5.18 Grafik Respon Kecepatan Putaran Roda 3 untuk nilai Kp, Ki dan Kd Tertentu

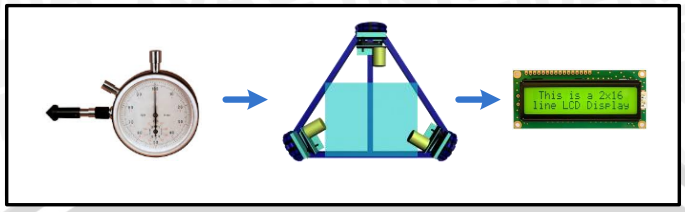
Gambar 5.18 (pada grafik yang dilingkari warna merah) menunjukkan respon sistem pada roda 3 sudah mampu mencapai *set point* dan *error steady state* yang kurang dari 20 rpm.

Dari penentuan nilai penguatan Kp, Ki dan Kd dapat dipastikan nilai penguatan yang digunakan untuk sistem kontrol kecepatan putaran roda 3 robot *three omni-directional* ini adalah Kp = 2,5, Ki = 0,07, dan Kd = 1.

5.5 Pengujian Kecepatan Putaran Roda

Pengujian ini bertujuan untuk melihat apakah roda telah berputar sesuai dengan hasil perhitungan kinematika robot *three omni-directional*. Prosedur pengujian dilakukan dengan memberi perintah pada robot untuk bergerak ke sudut

tertentu. Kecepatan tiap roda hasil perhitungan kinematika robot ditampilkan pada LCD. Ketika roda berputar, ukur kecepatan roda dengan menggunakan tachometer. Diagram blok pengujian kecepatan putaran roda ditunjukkan dalam Gambar 5.19.



Gambar 5.19 Diagram Blok Pengujian Kecepatan Roda

Hasil pengujian kecepatan roda pada sudut – sudut tertentu yang diperoleh melalui beberapa kali percobaan ditunjukkan dalam tabel 5.4

Tabel 5.4 hasil pengujian Kecepatan Roda

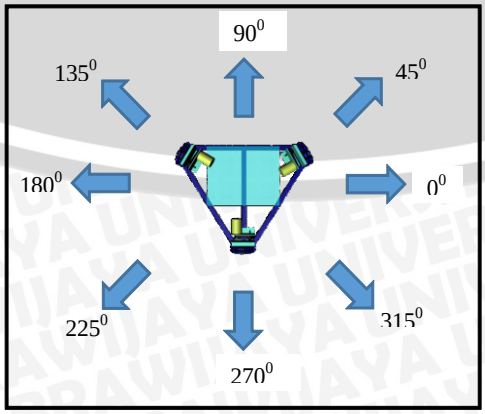
Sudut gerak robot (^{0})	**V_1 (RPM)**		**V_2(RPM)**		**V_3(RPM)**	
	LCD	Tachometer	LCD	Tachometer	LCD	Tachometer
0	99	96	99	99	198	196
		96		100		196
		96		104		196
45	50	40	190	180	140	154
		40		180		134
		48		184		140
90	170	166	170	176	0	0
		170		174		0
		165		170		0
135	190	180	50	52	140	136
		180		56		140
		184		57		140
180	99	90	99	100	198	184
		98		106		194
		96		106		194
225	50	46	190	186	140	136
		42		184		140
		48		186		140

Sudut gerak robot (^{0})	V_1 (RPM)		V_2(RPM)		V_3(RPM)	
	LCD	Tachometer	LCD	Tachometer	LCD	Tachometer
270	170	170	170	174	0	0
		170		176		0
		166		178		0
315	190	182	50	56	140	136
		177		50		136
		180		60		130

Berdasarkan Tabel 5.4, dapat diperoleh hasil bahwa kesalahan rata – rata yang terjadi saat pengukuran dengan tachometer pada roda 1 sebesar 5,37 rpm, pada roda 2 sebesar 5 rpm, dan pada roda 3 sebesar 2,62 rpm. Kesalahan rata – rata total sebesar 4,33 rpm. Kesalahan terbesar pada roda 1 sebesar 10 rpm, pada roda 2 sebesar 9 rpm, dan pada roda 3 sebesar 8 rpm. Kesalahan kecepatan roda terbesar sebesar 10 rpm. Standar devisasi (simpangan baku) kecepatan roda pada roda 1 sebesar 3,897 rpm. Standar devisasi (simpangan baku) kecepatan roda pada roda 2 sebesar 3,107 rpm. Standar devisasi (simpangan baku) kecepatan roda pada roda 3 sebesar 4,21 rpm.

5.6 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem ini bertujuan untuk mengetahui performansi robot *three omni-directional* untuk bergerak ke sudut – sudut tertentu. Prosedur pengujian dilakukan dengan mengendalikan robot dengan menggunakan *joystick* untuk bergerak ke sudut - sudut tertentu seperti yang ditunjukkan dalam gambar 5.20.



Gambar 5.20 Arah Gerak Robot

Hasil pengujian untuk respon robot saat dikendalikan untuk bergerak ke arah tertentu setelah beberapa kali percobaan dapat dilihat dalam tabel 5.5

Tabel 5.5 Hasil Pengujian Arah Gerak Robot

Sudut *Joystick* (^{0})	Sudut gerak robot (^{0})	Kesalahan sudut gerak robot(%)	Perubahan hadap robot(^{0})	Kesalahan arah hadap robot(%)
0	5	1,39	6	1,67
	5	1,39	7	1,94
	5	1,39	6	1,67
45	60	4,17	10	2,78
	50	1,39	14	3,89
	58	3,61	13	3,61
90	90	0,00	0	0,00
	88	0,56	0	0,00
	91	0,28	0	0,00
135	132	0,83	-13	3,61
	128	1,94	-10	2,78
	132	0,83	-10	2,78
180	190	2,78	-5	1,39
	175	1,39	-6	1,67
	181	0,28	0	0,00
225	218	1,94	-5	1,39
	218	1,94	0	0,00
	220	1,39	0	0,00
270	273	0,83	4	1,11
	272	0,56	0	0,00
	268	0,56	0	0,00
315	323	2,22	3	0,83
	319	1,11	-6	1,67
	323	2,22	0	0.00
Kesalahan Rata-rata		1,46		1,37

Berdasarkan tabel diatas dapat diperoleh hasil rata – rata kesalahan sudut gerak robot sebesar $4{,}375^0$ atau 1,46%. Kesalahan arah gerak robot terbesar adalah 11^0 saat sudut 45^0. Standar deviasi (simpangan baku) arah gerak robot sebesar $3{,}7^0$. Hasil pengujian menunjukkan rata – rata *error* hadap robot sebesar $5{,}75^0$ atau 1,37%. *Error* hadap robot terbesar adalah 12^0 terjadi pada saat sudut gerak robot 45^0. Standar deviasi (simpangan baku) arah hadap robot sebesar

$4{,}763^0$. Pada error hadap robot terdapat nilai sudut positif dan negatif, untuk positif menandakan robot berputar berlawanan arah jarum jam, sedangkan nilai negatif menandakan robot berputar searah jarum jam dari posisi awalnya.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil perancangan dan pengujian yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut.

1. Mikrokontroler dapat menerima data dari tombol *joystick* menggunakan komunikasi SPI dan memproses data tersebut menjadi sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0 sesuai dengan arah tombol analog ditekan dengan tingkat keberhasilan 100%.
2. Komunikasi antara mikrokontroler *master* dengan tiga mikrokontroler *slave* dengan menggunakan komunikasi SPI dengan tingkat keberhasilan 100%. Hal ini terlihat dari data yang dikirim dari mikrokontroler *master* sama dengan data yang diterima oleh mikrokontroler *slave*.
3. Mikrokontroler dapat mengakses keluaran pulsa sensor rotary encoder dan menampilkan pulsa keluaran rotary encoder yang diputar dengan besaran tertentu dengan kesalahan rata-rata sebesar 0,75 pulsa atau dibulatkan menjadi 1 pulsa. Kesalahan disebabkan karena pengujian memutar rotary encoder yang kurang presisi. Rotary encoder dapat difungsikan sebagai sensor kecepatan roda dengan kesalahan rata – rata total pada ketiga roda sebesar 1,31%.
4. Hasil Metode osilasi ziegler-nichols diperoleh nilai parameter PID pada roda 1 sebesar nilai Kp = 6, Ki = 92,31, dan Kd = 0,096. Roda 2 sebesar Kp = 7,8 , Ki =120, dan Kd = 0,1248. Roda 3 sebesar Kp = 7,2 , Ki =73,47 , dan Kd = 0,1764 . Parameter PID yang didapat dari metode kedua ziegler-nichols ini tidak dapat digunakan karena respon putaran roda yang masih terdapat osilasi. Parameter kontroler PID diperoleh dengan metode *hand tuning.* Pada roda 1 diperoleh nilai Kp = 3, Ki = 0,08, dan Kd = 1. Pada roda 2 diperoleh nilai Kp = 3, Ki = 0,1, dan Kd = 0,5. Pada roda 3 diperoleh nilai Kp = 2,5, Ki = 0,07, dan

Kd = 1. Dari parameter ini diperoleh respon kecepatan putar roda yang *steady* dengan *error steady state* sebesar 20 rpm.

5. Dengan menggunakan sistem ini mikrokontroler dapat mengatur kecepatan roda sesuai hasil perhitungan kinematika robot dengan kesalahan rata – rata pada roda 1 sebesar 5,37 rpm, roda 2 sebesar 5 rpm dan roda 3 sebesar 2,62 rpm. Kesalahan rata-rata total sebesar 4,33 rpm.
6. Dari hasil pengujian, robot *three omni-directional* dapat bergerak ke sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0 dengan kesalahan rata – rata sudut gerak robot sebesar $4{,}375^0$ atau 1,46%. Kesalahan arah gerak robot terbesar adalah 11^0 saat sudut arah gerak robot 45^0. Terdapat perubahan hadap robot setelah bergerak dengan rata – rata sebesar $5{,}75^0$ atau 1,37%.

6.2 Saran

Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah.

1. Perlu dicari alat pengendali robot yang dapat menunjukkan sudut gerak robot yang lebih presisi.
2. Diperlukan studi lebih lanjut agar robot dapat melakukan metode pemetaan posisi.

DAFTAR PUSTAKA

Al-Ammri, Salam dan Iman Ahmed. 2010. *Control of Omni –Directional Mobile Robot Motion*. http://cdn.intechopen.com/pdfs/465/InTech-Omnidirectional_mobile_robot_design_and_implementation.pdf. Diakses tanggal: 10 April 2013.

Andi Nalwan, Paulus. 2004. *Penggunaan dan Antarmuka Modul LCD M1632*. Jakarta: PT Elek Media Komputindo Kelompok Gramedi.

Atmel. 2006. *ATMEGA8/ATMEGA8L, 8-bit AVR with 8 kbytes in System Programable Flash.* www.atmel.com/literatur. Diakses tanggal: 8 Mei 2013.

Atmel. 2007. *ATMEGA32/ATMEGA32L, 8-bit AVR with 8 kbytes in System Programable Flash.* www.atmel.com/literatur. Diakses tanggal: 8 Mei 2013.

Bejo, Agus. 2008. *C & AVR Rahasia Kemudahan Bahasa C dalam Mikrokontroler ATmega8535*. Yogyakarta: Graha Ilmu.

Braunl, Thomas. 2006. *Embedded Robotics Mobile Robot Design and Applications with Embedded Systems.* Germany:Springer.

Dikti 2013. *Panduan Kontes Robot Indonesia 2013.* Jakarta: DIKTI.

Nugroho Adi, Agung.2009.*Antarmuka Joystick Playstation dengan mikrokontroler AVR menggunkan CVAVR.* http://nugroho.staff.uii.ac.id /files/2009/01/psx.pdf. Diakses tanggal : 10 April 2013.

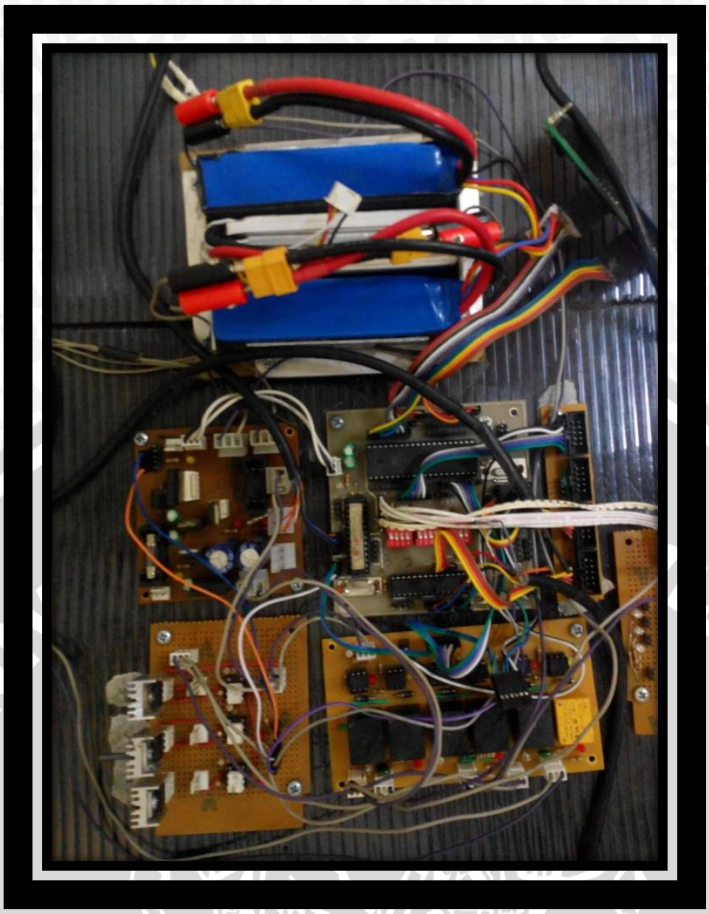
Ogata, Katsuhiko. 1997. *Teknik Kontrol Automatik Jilid 1*. Jakarta: Penerbit Erlangga.

Sudjadi. 2005. *Teori dan Aplikasi Microcontroller.* Yogyakarta: Graha Ilmu.

Sutrisno. 1987. *Elektronika, Teori dan Penerapannya.* Bandung: ITB.

LAMPIRAN I

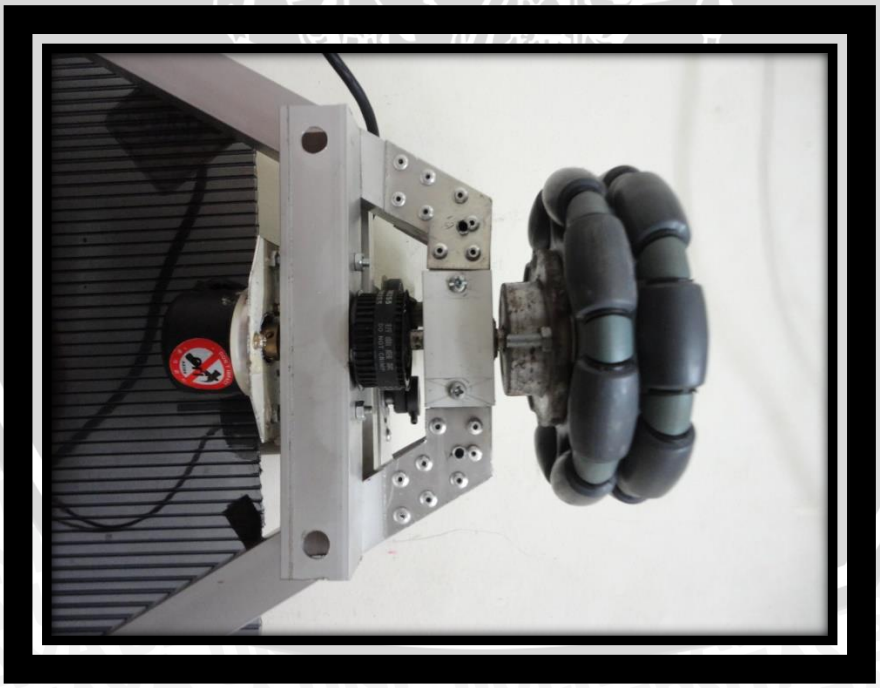
Dokumentasi Alat



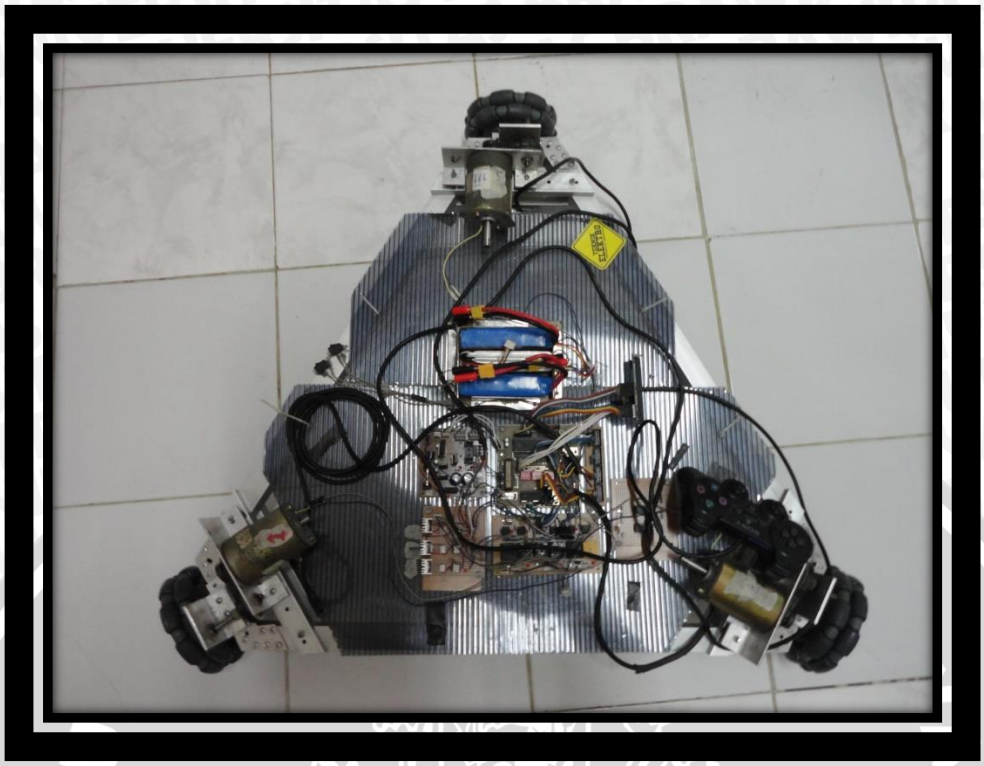
Gambar 1. *Board* Keseluruhan dan Baterai

Gambar 2. *Joystick* Playstation

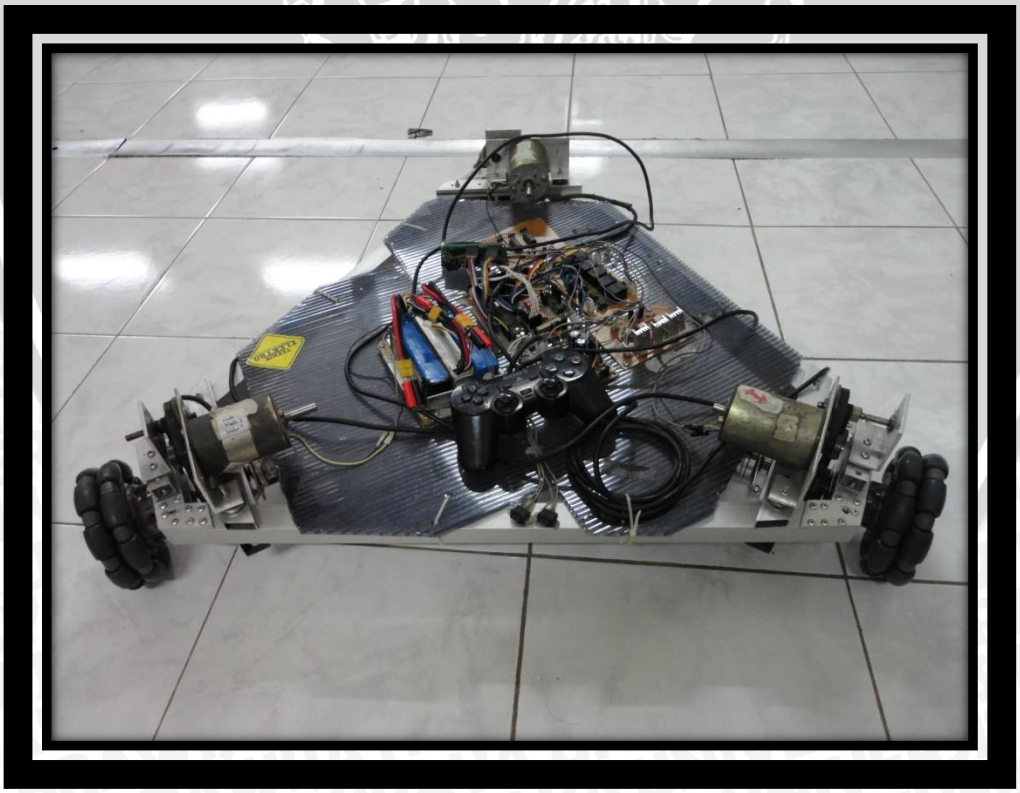
Gambar 3. Roda Omni



Gambar 4. Pemasangan Sensor Rotary Encoder



Gambar 5. Robot Tampak Atas



Gambar 5. Robot Tampak Perspektif

LAMPIRAN II

Listing Program

LISTING PROGRAM MIKROKONTROLER MASTER

```
/*****************************************************
****
This program was produced by the
CodeWizardAVR V1.25.7 beta 5 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech
s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 10/11/2009
Author  : jusuf
Company : jusufcorp
Comments:

Chip type               : ATmega32
Program type            : Application
AVR Core Clock frequency: 16.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 512
*****************************************************
***/
//library
#include <mega32.h>
#include <delay.h>
#include <stdio.h>
#include <math.h>

//LCD
#asm
   .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
// SPI functions
#include <spi.h>
#define ss PORTB.3
#define ss_slave1 PORTB.4
#define ss_slave2 PORTB.1
#define ss_slave3 PORTB.2

// Declare your global variables here
unsigned char ID,data3,data4,data5,data6;
float v1,v2,v3,tetha,;
```

```
int
hasil1,hasil2,hasil3,heading,ref_x,ref_y,v_putar;
bit penanda_stop=0;


char lcd[16];   //array LCD
// Timer 0 overflow interrupt service routine

void stik();
void analog_stik();


// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T
State2=T State1=T State0=T
PORTA=0x00;
DDRA=0xFF;

// Port B initialization
// Func7=Out Func6=In Func5=Out Func4=Out Func3=In
Func2=In Func1=In Func0=In
// State7=0 State6=T State5=0 State4=1 State3=T
State2=T State1=T State0=T
PORTB=0b00011110;
DDRB=0b10111110;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T
State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T
State2=T State1=T State0=T
PORTD=0x00;
DDRD=0xFF;
```

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 15.625 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 15.625 kHz
// Mode: Fast PWM top=0x00FF
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
```

```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// USART disabled
UCSRB=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by
Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 31.250 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: High
// SPI Data Order: LSB First
SPCR=0x7F;
SPSR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

lcd_init(16); //inisialisasi LCD 16x2

// Global enable interrupts
#asm("sei")

while (1)
{
stik();  //ambil  data stik

analog_stik(); //konversi data joystick ke sudut

if(penanda_stop==0)
{

     tetha = heading / 57.32;

     v1 = -(0.33*cos(tetha)) +
(0.57*sin(tetha))+(v_putar);
```

```
     v2 = -(0.33*cos(tetha)) -
(0.57*sin(tetha))+(v_putar);
     v3 = (0.66*cos(tetha))+(v_putar);

     v1 = (v1 * 300) * 0.136;
     v2 = (v2 * 300) * 0.136;
     v3 = (v3 * 300) * 0.136;

lcd_clear();
lcd_gotoxy(0,0);
sprintf(lcd,"%d derajat",heading);
lcd_puts(lcd);
lcd_gotoxy(0,1);
sprintf(lcd,"%d %d %d",(int)v1,(int)v2,(int)v3);
lcd_puts(lcd);
delay_ms(20);

     hasil1=v1;
     hasil2=v2;
     hasil3=v3;
//********pengiriman data ke slave***********
ss_slave1 = 0;
if(hasil1>=0)
{
PORTD.0=0;
PORTD.1=1;
}
else
{
hasil1=hasil1 * -1;
PORTD.0=1;
PORTD.1=0;
}
delay_us(8);
spi(hasil1);
delay_us(8);
ss_slave1 = 1;

ss_slave2 = 0;
if(hasil2>=0)
{PORTD.2=0;
PORTD.3=1;}
else
{
hasil2 = hasil2 * -1;
PORTD.2=1;
PORTD.3=0;;
}
delay_us(8);
spi(hasil2);
```

```
delay_us(8);
ss_slave2 = 1;

ss_slave3 = 0;

if(hasil3>=0)
{PORTD.4=0;
PORTD.5=1;}
else
{
hasil3= hasil3*-1;
PORTD.4=1;
PORTD.5=0;
}
delay_us(8);
spi(hasil3);
delay_us(8);
ss_slave3 = 1;

}


else if(penanda_stop==1)
{
lcd_clear();
```

```
lcd_gotoxy(0,0);
sprintf(lcd,"berhenti");
lcd_puts(lcd);
lcd_gotoxy(0,1);
sprintf(lcd,"%d %d ",data5,data6);
lcd_puts(lcd);
delay_ms(4);

PORTD.0=0;
PORTD.1=0;
PORTD.2=0;
PORTD.3=0;
PORTD.4=0;
PORTD.5=0;

delay_ms(10);
ss_slave1 = 0;
spi(0);
ss_slave1 = 1;

ss_slave2 = 0;
spi(0);
ss_slave2 = 1;

ss_slave3 = 0;
```

```
spi(0);
ss_slave3 = 1;


}
}
}
//*********sub fungsi baca stik***********
void stik()
{
      ss=0;
      spi(0x01);
      ID = spi(0x42);
      spi(0);
      delay_us(8);
      spi(0);
      delay_us(8);
      spi(0);
      delay_us(8);
      if(ID==0x73)
      {
      data3=spi(0);
      delay_us(8);
      data4=spi(0);
      delay_us(8);
      data5=spi(0);
      delay_us(8);
      data6=spi(0);
      delay_us(8);
      }
      ss=1;
}

//*********sub fungsi konversi data joystick ke
sudut*******
void analog_stik()
{
    penanda_stop=0;

ref_y = data6/51;
switch(ref_y)
{
case 0:
{data6=0;}break;

case 1:
{data6=1;}break;
```

```
case 2:
{data6=2;}break;

case 3:
{data6=3;}break;

case 4:
{data6=4;}break;

case 5:
{data6=4;}break;

default : break;
}

ref_x = data5/51;
switch(ref_x)
{
case 0:
{data5=0;}break;

case 1:
{data5=1;}break;

case 2:
{data5=2;}break;

case 3:
{data5=3;}break;

case 4:
{data5=4;}break;

case 5:
{data5=4;}break;

default : break;
}

penanda_stop=0;

if ((data5==4) && (data6==2))heading=0;
else if ((data5==4) && (data6==1))heading=45;
else if ((data5==4) && (data6==0))heading=45;
else if ((data5==2) && (data6==0))heading=90;
else if ((data5==0) && (data6==0))heading=135;
else if ((data5==0) && (data6==2))heading=180;
else if ((data5==0) && (data6==3))heading=225;
else if ((data5==0) && (data6==4))heading=225;
```

```
else if ((data5==1) && (data6==4))heading=225;
else if ((data5==2) && (data6==4))heading=270;
else if ((data5==4) && (data6==4))heading=315;
else if ((data5==4) && (data6==3))heading=0;

else penanda_stop=1;
    }
```

LISTING PROGRAM MIKROKONTROLER SLAVE

```
/*****************************************************
****
This program was produced by the
CodeWizardAVR V1.25.7 beta 5 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech
s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 10/11/2009
Author  : jusuf
Company : jusufcorp
Comments:


Chip type           : ATmega8
Program type        : Application
Clock frequency     : 16.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************
***/

#include <mega8.h>
#include <stdio.h>
#include <delay.h>
#include <math.h>
#define sinyal_pwm OCR1A


unsigned char time=0;
int set_point,view;
float out_motor;
int
error,del_error,last_error=0,sigma_error=0,pwm;
int kec;
float Kp,Ki,Kd;
float Ts=0.03264;

// Timer2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
// Place your code here
  if(time>0)time = time - 1;
  kec=TCNT0;
```

```
  TCNT0=0;
 }

// SPI interrupt service routine
interrupt [SPI_STC] void spi_isr(void)
{
unsigned char data;
data=SPDR;
set_point=data;
}



void main(void)
{

PORTB=0b00000000;
DDRB=0b00010010;

PORTC=0x04;
DDRC=0x07;

PORTD=0b11000000;
DDRD=0b11000000;
```

```
// Timer/Counter 0 initialization
// Clock source: T0 pin Rising Edge
TCCR0=0x07;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 15.625 kHz
// Mode: Fast PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xA1;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
```

```
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 15.625 kHz
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x07;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
//TIMSK=0x40;
TIMSK=0x40;

/*
USART initialization
Communication Parameters: 8 Data, 1 Stop, No
Parity
USART Receiver: Off
USART Transmitter: On
USART Mode: Asynchronous
USART Baud Rate: 9600
*/
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

ACSR=0x80;
SFIOR=0x00;

// SPI initialization
// SPI Type: Slave
// SPI Clock Rate: 4000.000 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: High
// SPI Data Order: LSB First
SPCR=0xEF;
SPSR=0x00;
```

```
// Clear the SPI interrupt flag
#asm
    in   r30,spsr
    in   r30,spdr
#endasm
// Global enable interrupts
#asm("sei")

//****motor1****
//Kp = 3;
//Ki = 2.451;
//Kd = 0.04896;

//****motor 2******
Kp=3;
Ki=1.532;
Kd=0.01632;

//****motor 3****
//Kp=3;
//Ki=2.145;
//Kd=0.03264;

while (1)
                                        {

                                                if(time == 0)
                                                {
                                                 time = 2;
                                                 //*****hitung PID********
                                          error = set_point - kec;
                                                  del_error = error-last_error;
                                                  sigma_error += error;
                                           out_motor = (error * Kp) + (del_error *
                                        (Kd/Ts)) + (sigma_error *   Ki*Ts);
                                                  pwm = out_motor    ;



                                                  if(pwm>150)pwm=150;
                                                  else if(pwm<0)pwm=0;
                                                  sinyal_pwm=pwm;

                                                  view = kec/0.136 ;
                                                  printf("%d \n",view);
                                                  last_error = error;


                                        };
                                    }}
```

LAMPIRAN III

Datasheet