DESAIN SISTEM PERGERAKAN *MOBILE ROBOT* DENGAN METODE EUCLIDEAN PADA KONTES ROBOT ABU INDONESIA 2019

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

DANU PRANANDARU
NIM. 155060300111048

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2019

LEMBAR PENGESAHAN

DESAIN SISTEM PERGERAKAN MOBILE ROBOT DENGAN METODE EUCLIDEAN PADA KONTES ROBOT ABU INDONESIA 2019

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Ditujukan untuk memenuhi persyaratan

memperoleh gelar Sarjana Teknik

DANU PRANANDARU
NIM. 155060300111048

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing
pada tanggal 19 Desember 2019

Dosen Pembimbing I	Dosen Pembimbing II
Goegoes Dwi Nusantoro, S.T., M.T.	Dr. Ir. Erni Yudaningtyas, M.T.
NIP. 19711013 200604 1 001	NIP. 19650913 199002 2 001

Mengetahui,
Ketua Jurusan Teknik Elektro

Ir. Hadi Suyono,S.T., M.T., Ph.D., IPM.
NIP. 19730520 200801 1 013

JUDUL SKRIPSI:

DESAIN SISTEM PERGERAKAN *MOBILE ROBOT* DENGAN METODE EUCLIDEAN PADA KONTES ROBOT ABU INDONESIA 2019

Nama Mahasiswa : DANU PRANANDARU

NIM : 155060300111048

Program Studi : TEKNIK ELEKTRO

Konsentrasi : TEKNIK KONTROL

Dosen Pembimbing 1 : Goegoes Dwi Nusantoro, S.T., M.T. ……

Dosen Pembimbing 2 : Dr. Ir. Erni Yudaningtyas, M.T. ……

Tim Dosen Penguji :

Dosen Penguji 1 : Dr. Ir. Mochammad Rusli, Dipl.Ing. ……

Dosen Penguji 2 : Muhammad Aziz Muslim, S.T., M.T., Ph.D. ……

Dosen Penguji 3 : Rahmadwati, S.T., M.T., Ph.D. ……

Tanggal Ujian : 06 Desember 2019

SK Penguji : 2525 TAHUN 2019

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Skripsi ini adalah asli dari pemikiran saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Skripsi ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Skripsi dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No.20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, 19 Desember 2019
Mahasiswa,

DANU PRANANDARU
NIM. 155060300111048

Teriring Ucapan Terima Kasih kepada:
Ibunda dan Alm. Ayahanda tercinta

RINGKASAN

Danu Pranandaru, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, November 2019, Desain Sistem Pergerakan *Mobile Robot* Dengan Metode Euclidean Pada Kontes Robot Abu Indonesia 2019, Dosen Pembimbing 1: Goegoes Dwi Nusantoro, Dosen Pembimbing 2: Erni Yudaningtyas

Permasalahan yang sering terjadi pada mobile robot khususnya dalam Kontes Robot ABU Indonesia (KRAI) adalah sering terjadi *human error* saat robot dikontrol manual oleh manusia. Untuk mengurangi kesalahan tersebut perlu digunakan metode kontrol gerak robot secara otomatis. Pada penelitian ini digunakan *path planning* metode Euclidean dengan *feedback* berupa gabungan data odometri dan *gyroscope*. Robot yang dibuat dalam penelitian ini menggunakan 4 roda *omni-directional* dengan 4 motor DC sebagai penggerak utama. Kontroler yang digunakan dalam penelitian ini adalah kontroler PI. Sedangkan untuk penentuan parameter kontroler K_p dan K_i dilakukan dengan menggunakan metode kedua Ziegler-Nichols. Dari perhitungan tersebut diperoleh parameter kontroler orientasi robot dengan nilai K_p = 3,15 K_i = 0,125 serta parameter kontroler posisi robot dengan nilai K_p = 6,75 K_i = 0,166. Setelah dilakukan pengujian pada sistem didapatkan nilai *settling time* rata-rata 1,135 detik, *error steady state* rata-rata 0,45%, *overshoot* rata-rata 4,48%, dan recovery time selama 0,5 detik.

Kata kunci: Mobile Robot, Odometri, Euclidean, Kontroler PI, Ziegler-Nichols.

SUMMARY

Danu Pranandaru, *Department of Electrical Engineering, Faculty of Engineering, Brawijaya University, September 2019, Mobile Robot Movement System Design with Euclidean Method in the 2019 Indonesian ABU Robot Contest: Goegoes Dwi Nusantoro, Erni Yudaningtyas,*

The problem that often occurs in mobile robots especially in the Indonesian ABU Robot Contest (KRAI) is human error when the robot is controlled manually by humans. To reduce these error, is is necessary to use automatic robot motion control. In this research, Euclidean method path planning is used with data combination of odometry and gyroscope. The robot made in this research uses 4 omni-directional wheels with 4 DC motors as main actuator. Controller used in this research is PI Controller. Determination of controller parameter K_p and K_i are done using Ziegler-Nichols second method. From these calculations, parameter of robot orientation controller are obtained with value K_p = 3,15 K_i = 0,125 and parameter of robot position controller with valaue K_p = 6,75 K_i = 0,166. After testing the system obtained average settling time 1,135 seconds, average error steady state 0,45%, average overshoot 4,48%, and recovery time is 0,5 seconds.

Keywords: *Mobile Robot, Odometry, Euclidean, PI Controller, Ziegler-Nichols.*

KATA PENGANTAR

Alhamdulillah, puji dan syukur penulis haturkan kehadirat Allah SWT, karena atas berkah rahmat petunjuk dan nikmat-Nya lah skripsi ini dapat diselesaikan. Sholawat serta salam tidak lupa penulis haturkan kepada junjungan Rasulullah Muhammad SAW, semoga kelak mendapatkan syafa'at beliau di yaumul qiyamah.

Skripsi berjudul "Desain Sistem Pergerakan *Mobile Robot* Dengan Metode Euclidean Pada Kontes Robot Abu Indonesia 2019" berikut disusun untuk memenuhi persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah banyak memberikan bantuan sehingga skripsi ini dapat terselesaikan dengan baik, antara lain:

1. Kedua orang tua penulis, Ibu Sri Hartati dan alm. Bapak Suyanto atas segala do'a, pengorbanan, kesabaran, inspirasi, nasihat, motivasi, kasih sayang, dan perhatiannya didalam membesarkan dan mendidik penulis. Mbak Fitria Pranandari dan Dek Aditria Pranandana atas dukungan, doa, semangat, motivasinya dan senantiasa memberi semangat hingga terselesaikannya skripsi ini.
2. Yang Terhormat Bapak Hadi Suyono, ST., MT., Ph.D. selaku Ketua Jurusan Teknik Elektro Universitas.
3. Yang Terhormat Ibu Ir. Nurussa'adah, MT. selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
4. Yang Terhormat Ibu Rahmadwati, ST., MT., Ph.D. selaku Ketua Program Studi Sarjana Teknik Elektro Universitas Brawijaya.
5. Yang Terhormat Bapak M. Aziz Muslim, ST., MT., Ph.D. selaku Ketua Kelompok Jabatan Fungsional Konsentrasi Teknik Kontrol Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
6. Yang Terhormat Bapak Goegoes Dwi Nusantoro, ST., MT. selaku Dosen Pembimbing 1 atas segala bimbingan, masukan, kritik dan saran yang telah diberikan selama proses pengerjaan skripsi.

7. Yang Terhormat Ibu Dr. Ir. Erni Yudaningtyas, M.T. selaku Dosen Pembimbing 2 atas segala bimbingan, masukan, kritik dan saran yang telah diberikan selama proses pengerjaan skripsi.
8. Bapak Dr. Ir. Sholeh Hadi Pramono, MS. selaku Dosen Penasehat Akademik atas segala pengarahan, saran, dan kritik yang telah diberikan selama perkuliahan.
9. Teman-teman SERVO 2015 atas segala bantuan dan kebersamaan yang telah diberikan selama masa studi.
10. Keluarga Besar Tim Robotika Universitas Brawijaya Angkatan 2015 (Godam/Mbah, Syafei/Pekpek, Bertoni/Tooon, Yola/Mamah, Alif/Gundul, Sholikhin/Pacul, Chandra/Bonek, Saidan/Lamis, Nadia/Dragon, Valen/Jo, Cita/Boos, Rahmat/Matt, Andrian/Yan, Ikrar/Ajo, Aby/Abo, Shamsul/Samuel, Rif'al/CEO, Yayak/Kacong, Gerdy/Kpopers) atas bimbingan, pengalaman, perhatian dan semangat yang telah diberikan.
11. Adik-adik tim robot KRAI 2018-2019 (Ashar, Banu, Gregory, Bram, Giffary, Dayat, Daffa, Nisa) atas semangat dan segala bantuannya.
12. Seluruh teman-teman serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan satu-persatu, terimakasih banyak atas bantuan dan dukungannya.

Dalam penyusunan skripsi ini, penulis menyadari bahwa masih terdapat kekurangan karena kendala dan keterbatasan dalam pengerjaan skripsi ini. Oleh karena itu, penulis berharap saran dan kritik yang membangun untuk penyempurnaan tulisan di masa yang akan datang. Semoga tulisan ini dapat bermanfaat dan dapat digunakan untuk pengembangan lebih lanjut.

Malang, 19 Desember 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kontes Robot ABU Indonesia (KRAI) adalah salah satu divisi yang dilombakan pada Kontes Robot Indonesia (KRI). Kontes Robot Indonesia merupakan perlombaan tahunan antar mahasiswa se-Indonesia. Kegiatan ini diselenggarakan oleh Kemenristekdikti guna mengasah kemampuan pada bidang robotika. Pemenang lomba ini akan menjadi perwakilan Indonesia menuju *Asia-Pacific Broadcasting Union Robot Contest* (ABU Robocon).

ABU Robocon tiap tahunnya memiliki tema dan peraturan yang berbeda-beda tergantung negara penyelenggaranya. Tema yang diusung biasanya seputar budaya maupun tradisi yang ada pada negara penyelenggara. ABU Robocon 2019 dilaksanakan di negara Mongolia yang mengusung tema *Great Urtuu: Sharing the knowledge*. Pada peraturan tahun ini tiap tim diwajibkan membuat 2 buah robot, yaitu *Messenger Robot 1* (MR1) dan *Messenger Robot 2* (MR2). MR1 merupakan robot beroda sedangkan MR2 merupakan robot berkaki 4 yang mempunyai tugas berjalan melewati lintasan yang telah ditentukan (ABU Robocon Rule, 2019).

Messenger Robot 1 dalam peraturan perlombaan boleh dikontrol secara manual maupun otomatis. Berhubung pergerakan robot secara manual berpotensi besar terjadi *human error*, diperlukan suatu metode pergerakan otomatis yang memungkinkan robot mengenali posisi serta lintasan yang dilaluinya. Salah satu metode yang bisa digunakan adalah metode odometri. Odometri adalah penggunaan data sensor pergerakan untuk memperkirakan perubahan posisi dari waktu ke waktu. Odometri digunakan untuk memperkirakan posisi relatif terhadap posisi awal (Fernando, 2011).

Berhubung lintasan yang dilalui robot harus sesuai dengan aturan perlombaan, diperlukan metode *path planning* dengan pengendalian yang baik agar robot dapat berjalan sesuai target. *Path planning* adalah metode penentuan lintasan yang harus dilalui oleh robot untuk menuju titik tujuan yang diinginkan. Metode perencanaan gerak yang bisa digunakan adalah metode Euclidean. Metode Euclidean adalah metode yang digunakan untuk mengukur jarak antara 2 titik yang berbeda dalam sistem koordinat kartesian (Rezky, 2018).

Dalam pergerakan robot dengan metode *path planning* diperlukan sebuah kontroler supaya robot dapat berjalan sesuai dengan target. Kontroler berfungsi untuk membandingkan nilai sebenarnya dari keluaran *plant* dengan nilai yang diinginkan, menentukan *error*, dan

menghasilkan suatu sinyal kontrol yang akan memperkecil *error* mendekati atau sama dengan nol. Terdapat beberapa macam tipe kontroler yang biasa digunakan, yaitu kontroler proporsional (P), proporsional integral (PI), serta proporsional integral differensial (PID). Kontroler PI digunakan karena memiliki karakteristik respon yang cepat serta dapat menghilangkan *offset* (Erni, 2017). Dengan digunakannya kontroler PI diharapkan respon sistem dapat sesuai dengan target.

1.1 Rumusan Masalah

Menurut permasalahan yang telah dijabarkan pada latar belakang, dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana merancang *path planning* mobile robot dengan metode Euclidean.
2. Bagaimana merancang kontroler untuk sistem pergerakan mobile robot.
3. Bagaimana merancang kontroler untuk sistem orientasi mobile robot.
4. Bagaimana respon sistem dengan *setpoint* yang berbeda.
5. Bagaimana respon sistem saat diberi gangguan.

1.2 Batasan Masalah

Dikarenakan objek penelitian yang luas perlu adanya pembatasan masalah supaya pembahasan lebih fokus pada rumusan masalah yang telah dijabarkan. Batasan masalah dalam perancangan penelitian ini adalah sebagai berikut:

1. Robot yang akan diteliti adalah *Messenger Robot* 1
2. Motor yang digunakan adalah motor dc dengan tipe PG45775
3. Mikrokontroler yang digunakan adalah STM32F407VT6
4. Kontroler yang digunakan adalah Proportional Integral (PI)
5. Metode penentuan parameter kontrol yang digunakan adalah metode Ziegler-Nichols
6. Pembahasan lebih ditekankan pada perancangan gerak robot dan algoritma kontroler
7. Modul *gyroscope* yang digunakan sudah memiliki fitur filter internal dan pengolahan sudut Euler
8. Penelitian tidak ditekankan pada pemodelan sistem

1.3 Tujuan

Tujuan penelitian ini adalah merancang sebuah *Messenger Robot* 1 yang dapat bergerak otomatis sesuai target posisi yang telah ditentukan dengan cepat dan akurat.

1.4 Manfaat Penelitian

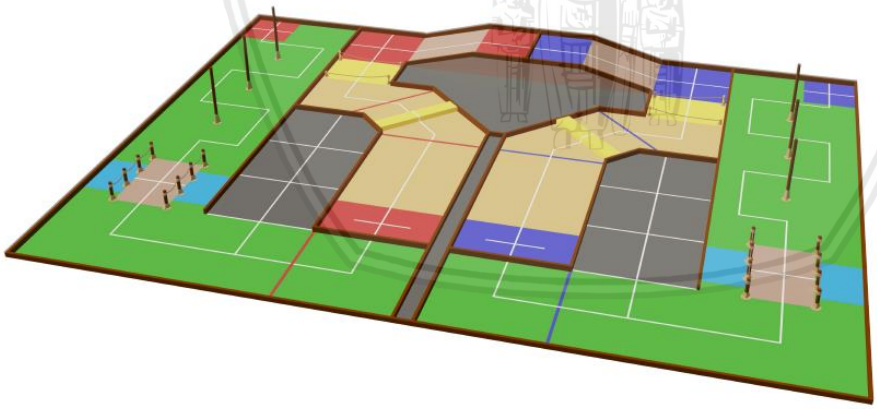
Penelitian ini memiliki beberapa manfaat yaitu bagi penulis sebagai sarana belajar dan penerapan ilmu yang didapat selama di perkuliahan. Sedangkan bagi tim robotika Teknik Elektro Universitas Brawijaya khususnya divisi KRAI adalah memberikan pengalaman merancang dan menerapkan sebuah sistem pergerakan otomatis mobile robot serta sebagai referensi ketika peraturan baru mempunyai kemiripan dengan penelitian ini. Selain itu penelitian ini juga dapat dijadikan sumber referensi serta informasi bagi calon peneliti pada bidang terkait.

BAB II
TINJAUAN PUSTAKA

2.1 Kontes Robot ABU Indonesia (KRAI)

Kontes Robot ABU Indonesia (KRAI) adalah salah satu divisi yang dilombakan rutin tiap tahun pada Kontes Robot Indonesia (KRI) yang diikuti mahasiswa berbagai perguruan tinggi se-Indonesia yang terdaftar di dikti. Peraturan perlombaan KRAI selalu mengacu pada *Asia-pasific Broadcasting Union Robot Contest* (ABU Robocon). ABU Robocon adalah perlombaan yang diadakan oleh persatuan penyiaran se Asia-Pasifik yang bertujuan untuk mengembangkan minat dan bakat para mahasiswa se Asia-Pasifik di bidang robotika. Pada ABU Robocon ini, tema dan peraturan yang ditetapkan setiap tahunnya selalu berbeda sesuai dengan keinginan negara yang menjadi tuan rumah perlombaan. Tema yang diusung biasanya seputar budaya maupun tradisi yang ada pada negara penyelenggara (Dikti, 2019).

Pada peraturan tahun ini tiap tim diwajibkan membuat 2 buah robot, yaitu *Messenger Robot 1* (MR1) dan *Messenger Robot 2* (MR2). MR1 merupakan robot beroda sedangkan MR2 merupakan robot berkaki 4 yang mempunyai tugas berjalan melewati lintasan yang telah ditentukan seperti dalam Gambar 2.1 (ABU Robocon Rule, 2019).

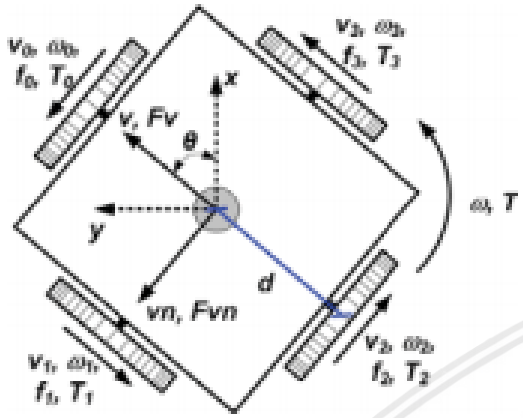


Gambar 2.1 Lapangan Perlombaan KRAI 2019
(Sumber: ABU Robocon 2019 Isometric View**)**

2.2 Mobile Robot

Mobile robot adalah robot yang dapat bergerak kesegala arah tanpa perlu melakukan perubahan orientasi. Pergerakan robot ini sangatlah mempermudah pergerakan yang memerlukan manuver yang tinggi dan perencanaan gerak yang kompleks. Pada sistem pergerakan konvensional, pergerakan tidak mampu dikontrol pada setiap tingkat sudut

kebebasan dalam bergerak secara independen sehingga hanya mampu bergerak pada arah yang terbatas. Mobile robot memiliki 3 *Degree of Freedom* (DOF) terhadap bidang gerak robot menggunakan roda *omni-directional* seperti dalam Gambar 2.2.



Gambar 2.2 Mobile Robot dengan Basis 4 Roda *Omni-directional*
(Sumber: Oliveira, 2009)

dengan:

x, y, θ	= Posisi robot (x, y) dan sudut θ merupakan arah hadap robot
d	= Jarak antara roda dengan titik tengah robot (m)
v_1, v_2, v_3, v_4	= Kecepatan linier tiap roda (m/s)
ω_1, ω_2, ω_3, ω_4	= Kecepatan angular tiap roda (rad/s)
f_0, f_1, f_2, f_3	= Arah gaya tiap roda (N)
T_0, T_1, T_2, T_3	= Arah torsi tiap roda (N·m)
v, vn	= Keceptan linier robot (m/s)
ω	= Kecepatan angular robot (rad/s)
F_v, F_{vn}	= Arah gaya robot terhadap v dan vn (N)
T	= Torsi robot terhadap ω (N·m)

Kinematika mobile robot dapat dihitung berdasarkan struktur mekanis dengan menggunakan pengembangan metode kinematika balik (*inverse kinematics*) dan transformasi antara koordinat bidang dan koordinat robot. Namun untuk mengetahui model kinematika mobile robot pada bidang datar, pertama harus mengidentifikasi posisi robot sebagai (x, y, θ) serta kecepatan linier tiap sumbu dapat dituliskan:

$$v_x(t) = dx(t)/dt\,;\; v_y(t) = dy(t)/dt;\; \omega(t) = d\theta(t)/dt$$

Sedangkan untuk mengubah kecepatan linier pada kondisi statis menjadi kecepatan linear terhadap sumbu robot dapat menggunakan Persamaan (2-1).

$$\begin{bmatrix} v(t) \\ vn(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) & 0 \\ -\sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} v_x(t) \\ v_y(t) \\ \omega(t) \end{bmatrix} \quad (2\text{-}1)$$

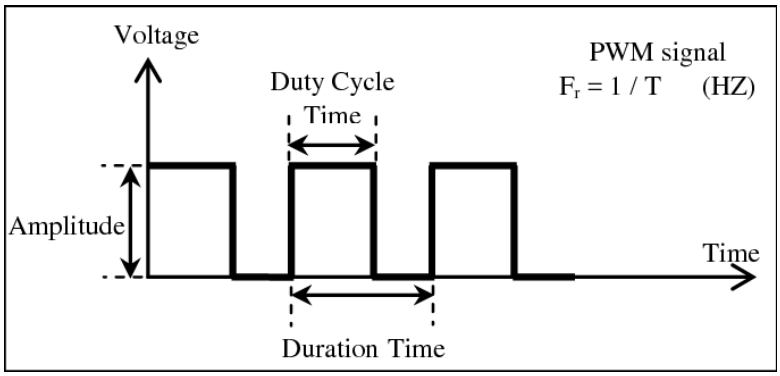
Oleh karena itulah implementasi kinematika balik (*inverse kinematics*) pergerakan robot 4 roda *omni-directional* dapat dituliskan dengan Persamaan (2-2).

$$\begin{bmatrix} v_0(t) \\ v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & d \\ -1 & 0 & d \\ 0 & -1 & d \\ 1 & 0 & d \end{bmatrix} . \begin{bmatrix} v(t) \\ vn(t) \\ \omega(t) \end{bmatrix} \quad (2\text{-}2)$$

(Sumber: Oliveira, 2009)

2.3 Pulse Width Modulation (PWM)

Pulse Width Modulation (PWM) secara umum adalah sebuah metode manipulasi lebar sinyal yang dinyatakan dengan pulsa dalam suatu periode, untuk mendapatkan tegangan rata-rata yang berbeda. Sinyal PWM pada umumnya memiliki amplitudo dan frekuensi dasar yang tetap, namun memiliki lebar pulsa yang bervariasi. Lebar pulsa PWM berbanding lurus dengan amplitudo sinyal asli yang belum termodulasi. Artinya, sinyal PWM memiliki frekuensi gelombang yang tetap namun *duty cycle* bervariasi (antara 0% hingga 100%). Dengan mengatur *duty cycle* akan diperoleh keluaran yang diinginkan. *Duty cycle* adalah besarnya sinyal kontrol yang diberikan pada *plant* (Sulistiono, 2010). Sinyal PWM secara umum ditunjukkan dalam Gambar 2.3.



Gambar 2.3 *Duty Cycle* Pada Sinyal PWM
(Sumber: Mokhtar, 2013)

2.4 Kontroler

Kontroler adalah sebuah subsistem dinamis yang dimasukkan ke dalam suatu sistem untuk memanipulasi suatu persamaan matematis sebuah *plant*. Secara umum kontroler juga dikenal dengan istilah kompensator, pengendali ataupun filter (Ogata, 2010).

Salah satu fungsi kontroler adalah mengurangi *error*, *error* adalah selisih antara nilai *setpoint* dengan respon *plant*. *Setpoint* adalah nilai referensi atau nilai yang diinginkan, sedangkan respon *plant* adalah nilai aktual yang terukur pada keluaran *plant*. Semakin kecil nilai *error* maka kinerja sistem kontrol dinilai semakin baik.

Prinsip kerja kontroler adalah membandingkan nilai keluaran *plant* dengan nilai *setpoint*, menentukan nilai *error* dan akhirnya menghasilkan sinyal kontrol untuk meminimalkan *error* (Ogata, 2010).

2.4.1 Kontroler Proporsional (P)

Kontroler proporsional adalah sebuah kontroler yang mempunyai karakteristik berupa mempercepat respon sistem. Kontroler proporsional memiliki keluaran yang sebanding dengan nilai *error*. Perhitungan keluaran kontroler ini merupakan hasil perkalian antara konstanta *gain* proporsional dengan nilai *error* seperti dalam Persamaan (2-3).

$$u(t) = Kp \times e(t) \qquad (2\text{-}3)$$

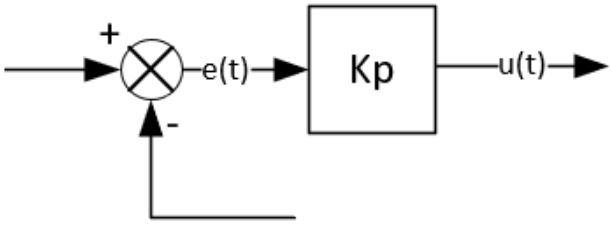
dengan:

Kp = Konstanta *gain* proporsional

e(t) = Nilai *error*

u(t) = Nilai keluaran kontroler

Diagram blok kontroler proporsional ditunjukkan dalam Gambar 2.4.



Gambar 2.4 Diagram Blok Kontroler Proporsional (P)

2.4.2 Kontroler Integral (I)

Kontroler integral adalah sebuah kontroler yang mempunyai karakteristik menghilangkan nilai *error steady state*. Keluaran kontroler ini merupakan hasil penjumlahan perubahan nilai *error* seperti dalam Persamaan (2-4).

$$u(t) = Ki \int e(t)\, dt \qquad (2\text{-}4)$$

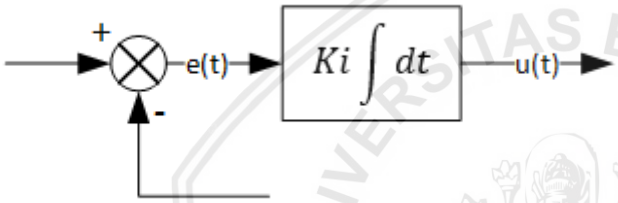
dengan:

Ki = Konstanta *gain* integral

e(t) = Nilai *error*

u(t) = Nilai keluaran kontroler

Diagram blok kontroler integral ditunjukkan dalam Gambar 2.5.



Gambar 2.5 Diagram Blok Kontroler Integral (I)

2.4.3 Kontroler Proporsional Integral (PI)

Kontroler proporsional integral (PI) adalah kontroler yang tersusun atas gabungan kontroler proporsional dan kontroler integral. Penjumlahan tiap kontroler menghasilkan keluaran kontroler PI seperti dalam Persamaan (2-5).

$$u(t) = Kp\, e(t) + Ki \int e(t)\, dt \qquad (2\text{-}5)$$

dengan:

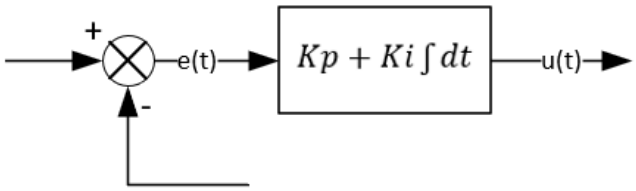
u(t) = Nilai keluaran kontroler

Kp = Konstanta *gain* proporsional

Ki = Konstanta *gain* integral

e(t) = Nilai *error*

Diagram blok kontroler PI ditunjukkan dalam Gambar 2.6.

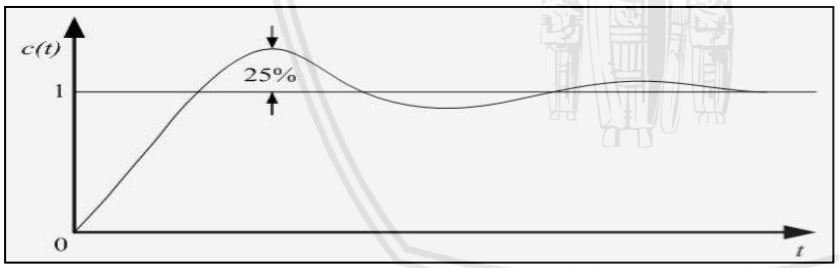


Gambar 2.6 Diagram Blok Kontroler PI

2.5 Aturan Penalaan Ziegler-Nichols

Metode yang digunakan dalam penelitian ini adalah metode Ziegler-Nichols (ZN)**.** Ziegler-Nichols mengemukakan aturan-aturan untuk menentukan nilai gain proporsional K_p, waktu integral T_i, dan waktu derivatif T_d berdasarkan karakteristik respon transien *plant*. Penentuan parameter kontrol PID atau penalaan kontrol PID tersebut dapat dilakukan dengan bereksperimen dengan *plant* (Ogata, 2010).

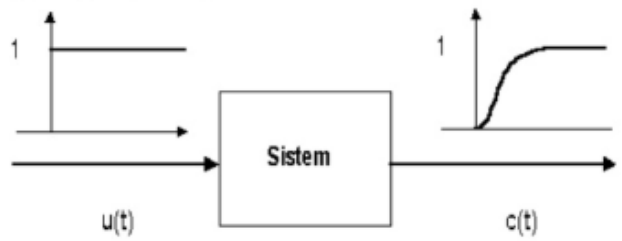
Terdapat dua metode yang disebut dengan aturan penalaan Ziegler-Nichols, metode pertama menggunakan prinsip *open-loop* sedangkan metode kedua menggunakan prinsip *closed-loop*. Pada kedua metode tersebut memiliki tujuan yang sama yaitu untuk mencapai respon 25% *maximum overshoot* pada masukan berjenis *unit step* seperti ditunjukkan dalam Gambar 2.7.



Gambar 2.7 Kurva Respon *Unit Step* Dengan 25% Maksimum *Overshoot*
(Sumber: Ogata, 2010)

2.5.1 Ziegler-Nichols Metode Pertama (*Open loop*)

Metode Ziegler-Nichols (ZN) metode pertama sering disebut metode kurva reaksi, respon *plant* diperoleh secara eksperimental dengan masukan berupa *unit step*, seperti yang ditunjukkan dalam Gambar 2.8.



Gambar 2.8 Respon Sistem Terhadap Masukan *Unit-Step*
(Sumber: Ogata, 2010)

Jika dalam *plant* tersebut terdapat integrator atau *dominan complex-conjugate poles*, maka respon *unit step* berbentuk kurva S, seperti ditunjukkan dalam Gambar 2.9 Jika respon tidak memberikan bentuk kurva S, maka metode ini tidak berlaku. (Ogata, 2010).

Gambar 2.9 Kurva S Respon
(Sumber: Ogata, 2010)

Kurva 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 kurva S, dan menentukan perpotongan antara garis tangen dengan sumbu waktu t dan sumbu c(t) = K.

Fungsi alih *C(s)/U(s)* dapat dilakukan pendekatan dengan sistem orde satu seperti pada Persamaan (2-6).

Dengan: $$\frac{C_{(s)}}{U_{(s)}} = \frac{Ke^{-Ls}}{T_s+1} \qquad (2\text{-}6)$$

Ziegler-Nichols menyarankan untuk menentukan nilai K_p, T_i dan T_d berdasarkan pada formula yang ditunjukkan dalam Tabel 2-1.

Tabel 2-1 Aturan Penalaan Ziegler-Nichols Metode Pertama

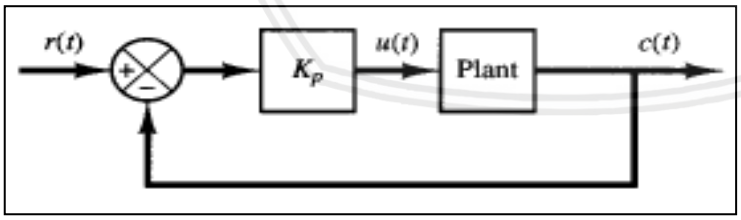
(Sumber: Ogata, 2010)

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

2.5.2 Ziegler-Nichols Metode Kedua (*Closed loop*)

Metode Ziegler-Nichols (ZN) metode kedua didasarkan pada respon sistem *closed loop* dengan masukan berupa *unit step*. *Plant* disusun seri dengan kontroler. Semula parameter integrator diset tak berhingga (Ti = ∞) dan parameter differensial diset nol (Td = 0). Kemudian dengan menggunakan tindakan kontrol proporsional, nilai Kp dinaikkan secara bertahap mulai nol sampai mencapai nilai yang mengakibatkan respon sistem berosilasi secara berkesinambungan. Jika keluaran tidak memiliki osilasi berkesinambungan untuk nilai Kp manapun yang telah diset, maka metode ini tidak berlaku.

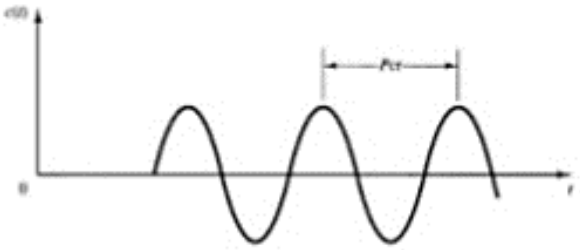
Nilai *gain* kritis dan periode dapat ditentikan dari keluaran sistem yang berosilasi secara berkesinambungan. Diagram blok sistem *closed loop* dengan kontrol proporsional ditunjukkan dalam Gambar 2.10 dan respon sistem yang berosilasi secara kesinambungan dengan periode P_{cr} ditunjukkan dalam Gambar 2.11.



Gambar 2.10 Sistem Closed Loop dengan Menggunakan Kp

(Sumber: Ogata, 2010)

Nilai *gain* proporsional pada sistem mencapai kondisi *sustain oscillation* disebut *critical gain* (K_{cr}). Periode *sustained oscillation* disebut dengan *corresponding period* (P_{cr}).



Gambar 2.11 Proses Penentuan Parameter P_{cr}
(Sumber: Ogata, 2010)

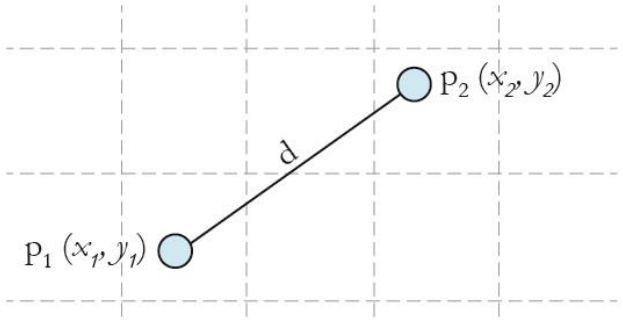
Setelah parameter K_{cr} dan P_{cr} didapatkan, nilai Kp, Ti, dan Td dapat dihitung dengan menggunakan rumus parameter PID untuk ZN metode kedua. Ziegler-Nichols menyarankan penyetelan nilai parameter Kp, Ti, Td dan berdasarkan rumus yang diperlihatkan dalam Tabel 2-2.

Tabel 2-2 Aturan Penalaan Ziegler-Nichols Metode Kedua
(Sumber: Ogata, 2010)

Tipe Kontroler	Kp	Ti	Td
P	$0{,}5 \times K_{cr}$	∞	0
PI	$0{,}45 \times K_{cr}$	$\frac{1}{1{,}2} \times P_{cr}$	0
PID	$0{,}6 \times K_{cr}$	$0{,}5 \times P_{cr}$	$0{,}125 \times P_{cr}$

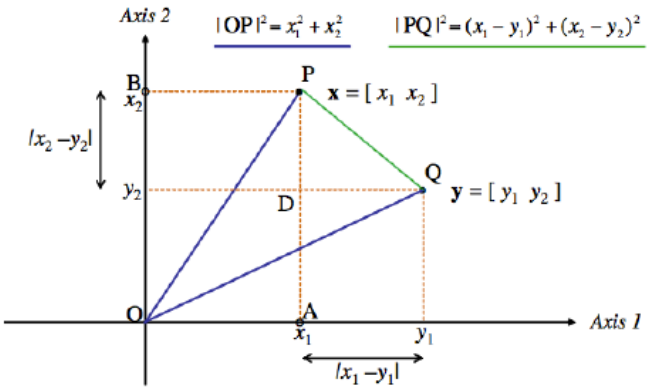
2.6 Fungsi Jarak Euclidean

Fungsi jarak Euclidean adalah salah satu metode perhitungan jarak antar titik dalam sebuah area yang disebut dengan Euclidean *Space*. Fungsi jarak Euclidean sangat erat kaitannya dengan teorema Pythagoras di mana hasil perhitungan kuadrat sisi miring didapatkan dari penjumlahan kuadrat sisi-sisi lainnya. *Phytagoras* pada luasan dua dimensi seperti dalam Gambar 2.12.



Gambar 2.12 Fungsi Jarak Euclidean

Penggunaan teorema Phytagoras dalam metode Euclidean dapat dilihat dalam Gambar 2.13.



Gambar 2.13 Teorema Phytagoras Dalam Area Dua Dimensi

dengan:

$d_{x,y}$ = Jarak antara vektor x dan vektor y

x_1 = Koordinat x pada titik pertama

x_2 = Koordinat x pada titik kedua

y_1 = Koordinat y pada titik pertama

y_2 = Koordinat y pada titik kedua

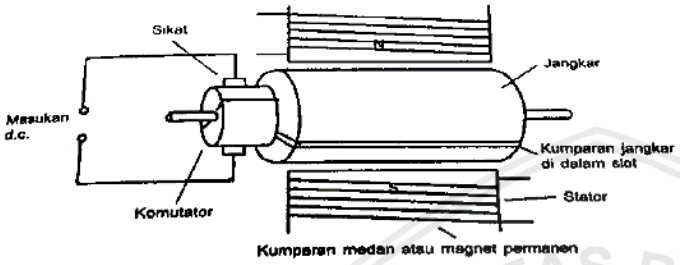
Dalam Gambar 2.13, dapat diketahui bahwa panjang kuadrat vektor $x = [x_1 \;\; x_2]$ merupakan penjumlahan kuadrat koordinat seperti pada segitiga OPA atau OPB dalam Gambar 2.13. $|OP|^2$ adalah Panjang kuadrat koordinat x yang merupakan jarak antara titik O dan titik P. Kemudian, jarak kuadrat antara vektor $x = [x_1 \;\; x_2]$ dan $y = [y_1 \;\; y_2]$ adalah penjumlahan selisih kuadrat koordinat tersebut, hal ini dapat dilihat pada segitiga PQD dalam Gambar 2.13. $|PQ|^2$ adalah kuadrat jarak antara titik P dan titik Q. Untuk mendapatkan jarak antara vektor x dan vektor y yang dapat ditulis dengan notasi $d_{x,y}$, maka dapat dirumuskan seperti dalam Persamaan (2-8).

$$d_{x,y} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (2\text{-}8)$$

2.7 Motor DC

Motor listrik sangat sering digunakan sebagai elemen kontrol akhir dalam sistem kontrol posisi dan kecepatan. Prinsip kerja dasar sebuah motor listrik adalah gaya yang bekerja pada konduktor yang berada di dalam suatu medan magnet ketika ada arus yang melewati konduktor tersebut (W. Bolton, 2004).

Motor DC memerlukan suplai tegangan yang searah pada kumparan medan untuk diubah menjadi energi mekanik. Bagian utama motor DC adalah stator (bagian yang tidak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar). Catu tegangan DC dari baterai menuju ke lilitan melalui sikat yang menyentuh komutator, dua segmen yang terhubung dengan dua ujung lilitan. Kumparan dalam satu lilitan disebut angker dinamo. Angker dinamo adalah sebutan untuk komponen yang berputar di antara medan magnet. Elemen-elemen dasar motor DC dijelaskan dalam Gambar 2.14.



Gambar 2.14 Elemen Dasar Motor DC
(Sumber: W. Bolton 2004)

Motor DC memiliki prinsip kerja jika sikat arang terhubungkan dengan satu sumber arus serah di luar dengan tegangan V, maka satu arus I masuk ke terminal kumparan rotor dan menghasilkan fluks. Dengan adanya fluks stator dan arus rotor akan menghasilkan satu gaya yang bekerja pada kumparan yang dikenal dengan gaya Lorentz.

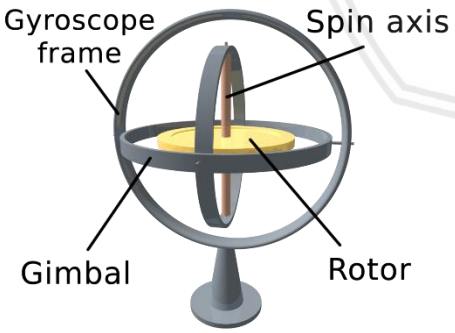
2.8 Rotary Encoder Omron E6C2

Rotary encoder adalah perangkat elektromekanis yang berfungsi mengenali perubahan posisi suatu perangkat yang berbasis rotasi mekanis. Keluaran *rotary encoder* ini berbentuk pulsa sinyal High dan Low. Pada *rotary encoder* E6C2 produksi Omron, dalam sekali putaran poros dapat menghasilkan 2000 pulsa. Posisi sudut poros perangkat berbasis rotasi dapat diolah oleh *rotary encoder* menjadi informasi berupa kode digital untuk kemudian diteruskan oleh rangkaian kontrol. R*otary encoder* Omron E6C2 menggunakan tegangan $5V \pm 5\%$ untuk pulsa keluaran dengan logika *High* dan $0V$ untuk pulsa keluaran dengan logika *Low*. Bentuk fisik *rotary encoder* Omron E6C2 dapat dilihat dalam Gambar 2.15.

Gambar 2.15 *Rotary Encoder*
(Sumber: www.omron.com)

2.9 Modul *Gyroscope* WT61C

Gyroscope adalah sebuah perangkat yang digunakan untuk mengenali perubahan orientasi dengan prisip ketetapan momentum sudut. Secara mekanis *gyroscope* berbentuk menyerupai sebuah roda atau cakram yang berputar di mana poros bebas digunakan sebagai acuan dalam mengenali setiap orientasi. Meskipun orientasinya tidak tetap, perubahannya dalam menanggapi torsi eksternal jauh lebih sedikit dan berlangsung dalam arah yang berbeda jika dibandingkan dengan tanpa momentum sudut yang berkaitan dengan tingginya tingkat putaran dan momen inersia. Orientasi perangkat tetap sama, terlepas dari gerak bidang pemasangan, karena pemasangan perangkat pada sebuah gimbal akan meminimalkan torsi eksternal. Bentuk fisik *gyroscope* secara umum dapat dilihat dalam Gambar 2.16.



Gambar 2.16 *Gyroscope*
(Sumber: www.wikipedia.org)

Modul *gyroscope* WT61C adalah salah satu jenis sensor yang memanfaatkan prinsip *gyroscope* elektrik. Di dalam modul ini terdapat mikrokontroler yang dapat mengolah *raw* data sensor *gyroscope* dengan menyaring sampai mengubahnya menjadi keluaran dalam sudut *euler*. Sudut *euler* adalah sudut yang digunakan untuk merepresentasikan orientasi

suatu benda terhadap sistem koordinat tetap. Sudut *euler* terdiri dari 3 dimensi sudut yaitu Roll (ϕ), Pitch (θ), dan Yaw (ψ). Roll adalah sudut rotasi yang mengacu pada sumbu X. Pitch adalah sudut rotasi yang mengacu pada sumbu Y. Yaw adalah sudut rotasi yang mengacu pada sumbu Z. Metode penyaringan data yang digunakan pada modul ini adalah teknologi filter digital tingkat lanjut yang dapat secara efektif mengurangi derau pengukuran dan meningkatkan akurasi pengukuran. Bentuk fisik modul *gyroscope* WT61C dapat dilihat dalam Gambar 2.17.

Gambar 2.17 Modul *Gyroscope* WT61C
(Sumber: www.wit-motion.com)

Spesifikasi modul *gyroscope* WT61C adalah sebagai berikut:

- Voltage : 3.3V - 5V
- Current : <10mA
- Volume : 51.3mm X 36mm X 15mm
- Measuring dimension : 3D Acceleration, 3D Angular Velocity, 3D Attitude angle
- Range : Acceleration ±16g, Angular velocity ±2000°/s, Angle XY axis ±180° Z axis ±90°
- Resolution : Acceleration 0.0005g, Angular velocity 0.61°/s
- Stability : Acceleration 0.01g, Angular velocity 0.05°/s
- Attitude stabilization : 0.01°
- Data output : time, acceleration, angle
- Data output frequency : 100Hz (baud rate 115200) / 20Hz (9600 baud)
- Data interface : Serial port (TTL)
- Baud rate : 9600, 115200 (default)

2.10 Driver Motor H-Bridge

Driver motor berfungsi untuk mengubah sinyal PWM dari mikrokontroler menjadi tegangan keluaran. Dalam aplikasinya, driver motor biasanya terdiri dari rangkaian transistor yang tersusun sedemikian rupa sehingga dapat mengendalikan arah putar dan kecepatan motor berdasarkan arah *loop* dan tegangan kutub motor. Salah satu d*river* motor yang umum digunakan adalah *driver* motor VNH3SP30. Bentuk fisik *driver* motor VNH3SP30 dapat dilihat dalam Gambar 2.18.

Gambar 2.18 Driver Motor VNH3SP30
(Sumber: www.tokopedia.com)

2.11 Mikrokontroler *STM32F407VGT6*

STM32F407VGT6 menggunakan *chip* ARM®Cortex®-M4 32-bit RISC yang beroperasi pada frekuensi hingga 168 MHz. *Chip* ARM®Cortex-M4 dilengkapi dengan Floating Point Unit (FPU) tunggal yang mendukung instruksi pemrosesan data. Modul ini juga mengimplementasikan set lengkap instruksi *Digital Signal Processing* (DSP) dan *Memory Protection Unit* (MPU) yang digunakan untuk meningkatkan keamanan aplikasi. Bentuk fisik modul *mikrokontroler* STM32F407VGT6 dapat dilihat dalam Gambar 2.19.

Gambar 2.19 *Mikrokontroler* STM32F407VGT6
(Sumber: www.alibaba.com)

STM32F407VGT6 menggabungkan memori tertanam berkecepatan tinggi (memori Flash hingga 1 Mbyte, dan 192 Kbytes SRAM), 4 Kbytes SRAM cadangan, pin *General Purpose Input Output* (GPIO) yang disempurnakan, peripheral yang terhubung ke dua *Advanced Peripheral Bus* (APB), tiga *Advanced High-performance Bus* (AHB) dan matriks bus multi-AHB 32-bit.

Spesifikasi modul mikrokontroler STM32F407VGT6 adalah sebagai berikut:

- Core : Cortex-M4F
- Memory : 1MB Flash
- RAM : 192KB SRAM
- Debug Mode : Serial Wire Debug (SWD) & JTAG Interface
- Package : LQFP100
- I/O Pins : 82
- Timers (16-bit) : 12
- Advanced Control Timers : 2
- General Purpose Timers : 10
- Basic Timers : 2
- PWM Channels : 6
- ADC (12-bit) : 3 (16 Channel)
- I2C (TWI) : 3
- USART : 4
- SPI : 3 Full Duplex
- DMA : 2 (8 Channel)
- USB : 1 (2.0 Full Speed)
- CAN : 2 (2.0 Active)
- Supply Voltage : 1.8 – 3.6 V

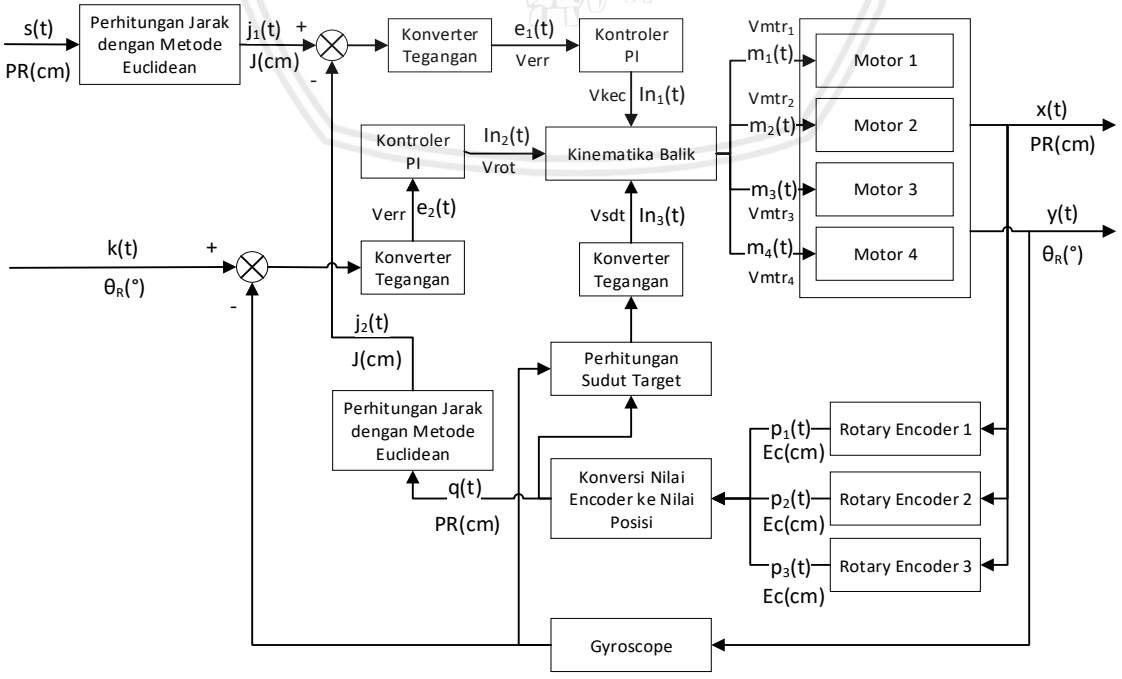
BAB III
METODE PENELITIAN

Untuk menyelesaikan rumusan masalah dan mencapai tujuan penelitian yang telah dijabarkan pada bab pendahuluan, maka dibutuhkan langkah-langkah penyelesaian masalah. Metode yang penulis gunakan adalah sebagai berikut:

1. Perancangan blok diagram
2. Perancangan desain mekanikal alat
3. Spesifikasi desain
4. Perancangan sistem odometri
5. Karakterisasi setiap subsistem
6. Perancangan perangkat keras
7. Perancangan perangkat lunak
8. Pengujian alat
9. Pengambilan kesimpulan dan saran

3.1 Perancangan Blok Diagram

Perancangan blok diagram digunakan sebagai tahap awal dalam pembentukan sebuah sistem beserta rangkaian alat yang akan digunakan. Blok diagram bertujuan menjelaskan garis besar sistem dan diharapkan alat dapat bekerja sesuai dengan blok diagram sistem. Blok diagram sistem dapat dilihat dalam Gambar 3.1.



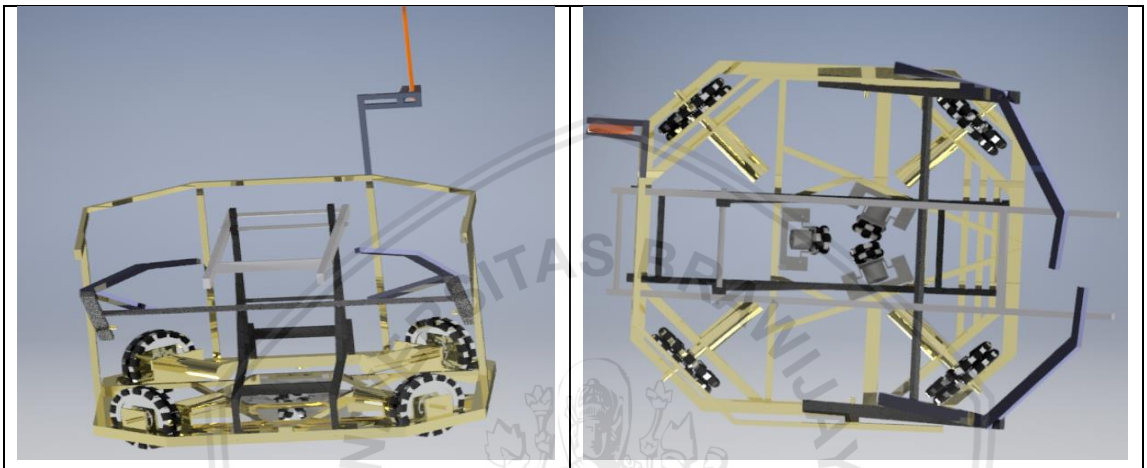
Gambar 3.1 Blok Diagram Sistem Kontrol Pergerakan Robot

Keterangan:

1. s(t): *Setpoint* / Nilai *input* sistem yang diinginkan pada *output* sistem (cm)
2. k(t): *Setpoint* / Nilai *input* sistem yang diinginkan pada *output* sistem (°)
3. j_1(t): Merupakan nilai hasil konversi posisi target ke nilai target jarak (cm)
4. j_2(t): Merupakan nilai hasil konversi posisi aktual ke nilai actual jarak (cm)
5. e_1(t): Merupakan selisih antara *setpoint* posisi dengan nilai aktual posisi (V)
6. e_2(t): Merupakan selisih antara *setpoint* orientasi dengan nilai aktual orientasi (V)
7. In_1(t): Merupakan *output* kontroler PI yang dijadikan *input* kecepatan pada Kinematika Balik (V).
8. In_2(t): Merupakan *output* kontroler PI yang dijadikan *input* rotasi pada Kinematika Balik (V).
9. In_3(t): Merupakan hasil perhitungan sudut target yang dijadikan *input* sudut gerak pada Kinematika Balik (V).
10. In_3(t): Merupakan hasil perhitungan sudut target yang dijadikan *input* sudut gerak pada Kinematika Balik (V).
11. m_1(t): Merupakan *output* Kinematika Balik berupa tegangan jangkar motor DC 1 (V)
12. m_2(t): Merupakan *output* Kinematika Balik berupa tegangan jangkar motor DC 2 (V)
13. m_3(t): Merupakan *output* Kinematika Balik berupa tegangan jangkar motor DC 3 (V)
14. m_4(t): Merupakan *output* Kinematika Balik berupa tegangan jangkar motor DC 4 (V)
15. x(t): Merupakan *output* aktual dari sistem yaitu posisi robot (cm)
16. y(t): Merupakan *output* aktual dari sistem yaitu orientasi robot (cm)
17. p_1(t): Merupakan nilai jarak hasil pembacaan sensor *rotary encoder* 1 (cm)
18. p_2(t): Merupakan nilai jarak hasil pembacaan sensor *rotary encoder* 2 (cm)
19. p_3(t): Merupakan nilai jarak hasil pembacaan sensor *rotary encoder* 3 (cm)
20. q(t): Merupakan nilai hasil konversi nilai encoder ke nilai posisi robot (cm)

3.2 Perancangan Desain Mekanikal Alat

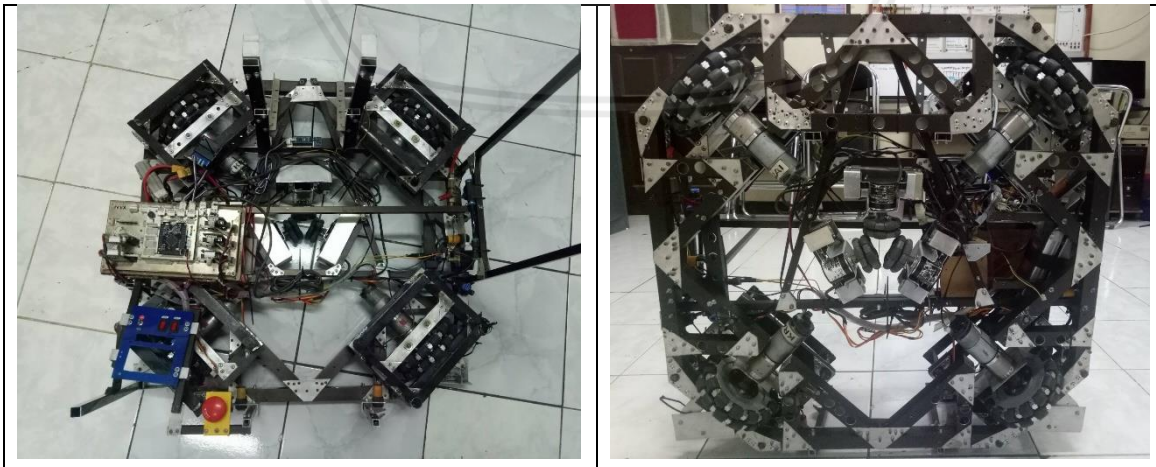
Perancangan desain mekanikal alat dalam penelitian ini menggunakan *software* Autodesk Inventor *for Students*. Bagian yang di desain adalah rangka secara keseluruhan, sistem pergerakan, serta sensor yang dipakai. Penggerak utama robot dalam penelitian ini menggunakan 4 buah motor yang masing-masing motor dipasang roda *omni-directional*. Sedangkan untuk sistem odometri menggunakan 3 buah *rotary encoder* yang dipasang pada titik pusat robot seperti tampak dalam Gambar 3.2.



(a) (b)

Gambar 3.2 Desain 3D Robot (a) Tampak Depan (b) Tampak Atas

Setelah desain 3D robot selesai, tahap selanjutnya adalah merealisasikan ke dalam bentuk fisik. Hasil dari realisasi desain 3D robot dalam Gambar 3.2 dapat dilihat dalam Gambar 3.3.



(a) (b)

Gambar 3.3 Bentuk Robot Setelah Dirakit (a) Tampak Atas (b) Tampak Bawah

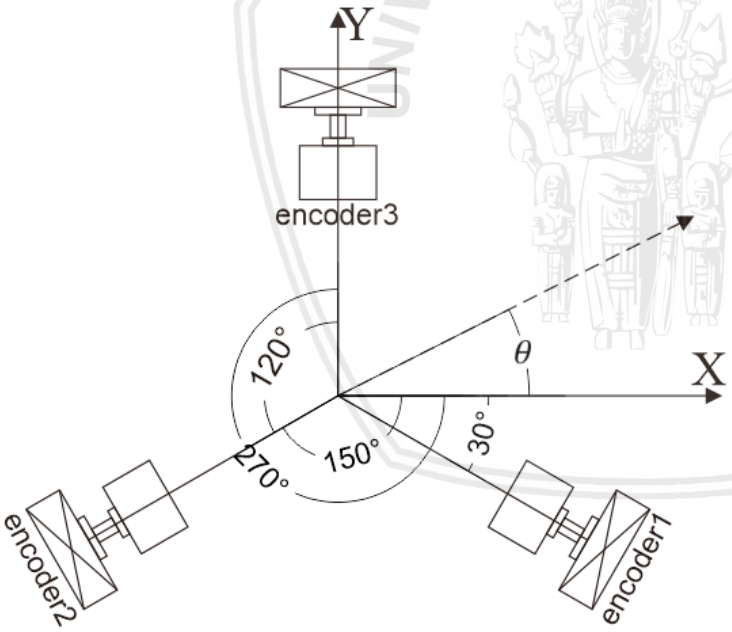
3.3 Spesifikasi Desain

Respon sistem yang diinginkan pada penelitian ini memiliki spesifikasi desain sebagai berikut:

1. *Settling time* kurang dari 5 detik karena dibutuhkan respon sistem yang cepat.
2. *Error steady state* kurang dari 3% karena sistem yang baik memiliki *error steady state* kurang dari 5%.
3. *Overshoot* kurang dari 10% karena sistem yang baik memiliki *overshoot* maksimum 10%.

3.4 Perancangan Sistem Odometri

Dalam penelitian ini *rotary encoder* digunakan sebagai sensor pembaca perubahan posisi robot. *Rotary encoder* yang digunakan sejumlah 3 buah dan memiliki spesifikasi catu daya $5VDC$ dengan resolusi 2000 pulsa per rotasi. Ketiga *rotary encoder* ini disambungkan ke board STM32F407VGT6. Desain letak pemasangan *rotary encoder* dapat dilihat dalam Gambar 3.4.



Gambar 3.4 Desain Pemasangan *Rotary Encoder* Sebagai Umpan Balik Posisi

Dengan desain pemasangan *rotary encoder* seperti dalam Gambar 3.4, didapat persamaan sistem odometri seperti dalam Persamaan (3-1) dan Persamaan (3-2)

$$x(t) = \sum_{t=0}^{\infty} sin(30 + \theta(t)) * encoder1(t) + sin(150 + \theta(t)) * encoder2(t) + sin(270 + \theta(t)) * encoder3(t) \quad (3\text{-}1)$$

$$y(t) = \sum_{t=0}^{\infty} cos\big(30 + \theta(t)\big) * encoder1(t) + cos\big(150 + \theta(t)\big) * encoder2(t) + cos(270 + \theta(t)) * encoder3(t) \quad (3\text{-}2)$$

Sedangkan untuk perhitungan jarak antar titik dan sudut target gerak dalam penelitian ini menerapkan metode Euclidean seperti yang telah dijelaskan dalam Persamaan (2-7), dapat dilihat dalam Persamaan (3-3) dan Persamaan (3-4).

$$Jarak = \sqrt{(Xsekarang - Xtarget)^2 + (Ysekarang - Ytarget)^2} \quad (3\text{-}3)$$

$$Sudut\ Target\ Euclidean = \tan^{-1}\left(\frac{Xsekarang - Xtarget}{Ysekarang - Ytarget}\right) \quad (3\text{-}4)$$

Namun, sudut target hasil perhitungan Euclidean merupakan perhitungan sudut dari posisi robot sekarang menuju koordinat target tanpa mempertimbangkan *error* orientasi robot. Untuk mengatasi hal tersebut, sudut target harus disesuaikan dengan nilai *error* orientasi robot yang dibaca oleh sensor *gyroscope* seperti dalam Persamaan (3-5).

$$Sudut\ Target\ Real = Sudut\ Target\ Euclidean - errorOrientasi \quad (3\text{-}5)$$

3.5 Karakterisasi Setiap Subsistem

3.5.1 Karakterisasi Motor DC PG45

Karakterisasi dilakukan untuk mengetahui karakter atau nilai *gain* yang sesuai sebagai parameter kontrol. Hal tersebut didapatkan dengan mengamati tegangan terhadap kecepatan motor. Karakterisasi ini terdiri atas 4 buah motor DC PG45 yang digunakan sebagai penggerak utama.

Untuk melakukan karakterisasi ini digunakan alat-alat sebagai berikut:

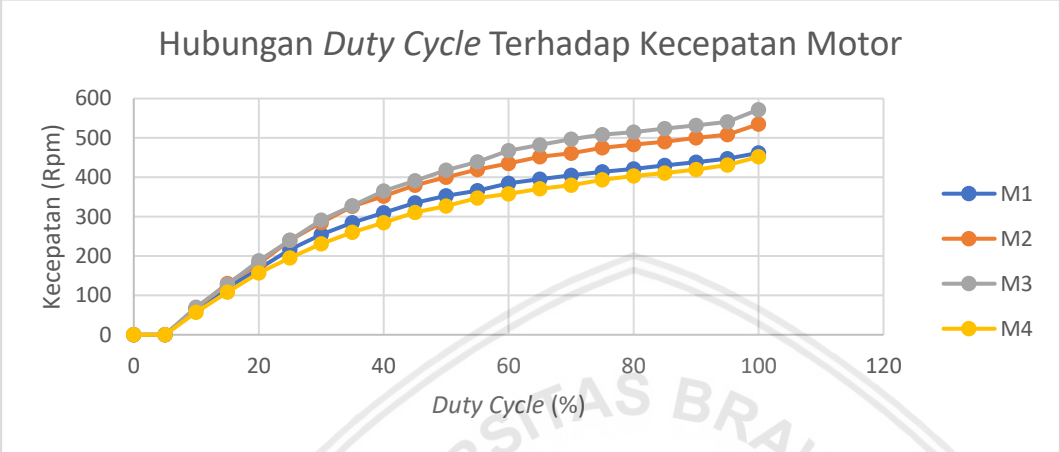
1. Perangkat komputer
2. Motor DC PG45 4 buah
3. *Driver* motor VNH2SP30
4. Baterai Li-Po 24v
5. *Rotary encoder*
6. *Mikrokontroler* STM32F407
7. *Software* STM Studio

Prosedur pengujiannya adalah sebagai berikut:

1. Menghubungkan pin IN1, IN2, dan PWM pada modul *driver* motor ke *mikrokontroler* STM32F407
2. Menghubungkan catu daya 24v pada *driver* motor
3. Menghubungkan OUTA dan OUTB pada modul *driver* motor dengan motor DC
4. Menghubungkan *mikrokontroler* STM32F407 ke perangkat komputer dan buka *software* STM Studio

5. Mengubah nilai *duty cycle* PWM pada *software* STM Studio
6. Mencatat perubahan kecepatan yang tampil pada *software* STM Studio

Hasil karakterisasi 4 buah motor DC PG45 yang terdiri atas motor 1 (M1), motor 2 (M2), motor 3 (M3), dan motor 4 (M4) dapat dilihat dalam Gambar 3.5.



Gambar 3.5 Hubungan Duty Cycle Terhadap Kecepatan Motor

Hasil karakterisasi 4 motor DC PG45 dalam Gambar 3.5 menunjukkan bahwa dari ke 4 motor tersebut memiliki respon yang berbeda-beda. Motor 1 dengan motor 4 menunjukkan respon sistem yang hampir sama. Motor 2 dengan motor 3 menunjukkan respon sistem yang memiliki perbedaan tidak terlalu jauh. Sedangkan motor 4 dengan motor 3 memiliki selisih kecepatan maksimal sebesar 120 Rpm.

3.5.2 Karakterisasi *Driver* Motor DC

Karakterisasi *driver* motor dc dilakukan untuk mengetahui kinerja dan respon rangkaian *driver* motor dc VNH3SP30 dengan membandingkan tegangan keluaran dengan nilai *duty cycle* 0-100% sinyal PWM keluaran *mikrokontroler*.

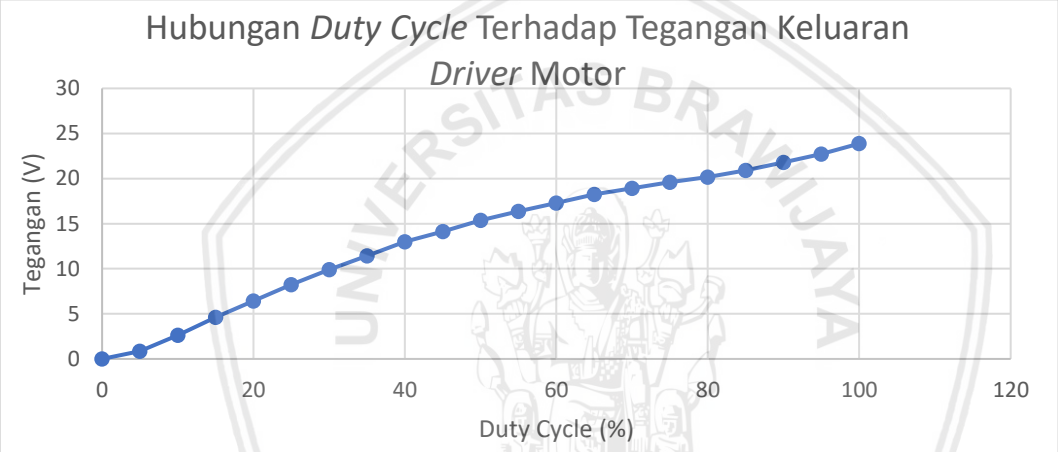
Untuk melakukan karakterisasi ini digunakan alat-alat sebagai berikut:

1. Perangkat komputer
2. Baterai Li-Po 24v
3. *Driver* motor VNH3SP30
4. *Multimeter*
5. STM32F407VGT6
6. Modul ST-Link V2
7. Kabel penghubung

Prosedur pengujiannya adalah sebagai berikut:

1. Menghubungkan tegangan keluaran baterai dengan masukan tegangan referensi *driver* motor dc VNH3SP30.
2. Menghubungkan masukan PWM serta DIR *driver* motor VNH3SP30 dengan pin keluaran PWM dan DIR pada STM32F407VGT6
3. Menghubungkan keluaran tegangan *driver* motor dc VNH3SP30 dengan *multimeter*
4. Menghubungkan *mikrokontroler* ke perangkat komputer menggunakan ST-Link V2 dan atur *duty cycle* sinyal PWM pada STM32F407VGT6 dengan rentang nilai 0-100%
5. Mencatat hasil pembacaan *multimeter* setiap kelipatan kenaikan *duty cycle* 5%

Hasil karakterisasi *driver* motor VNH3SP30 dapat dilihat dalam Gambar 3.6.



Gambar 3.6 Hubungan *Duty Cycle* Terhadap Tegangan Keluaran *Driver* Motor

Hasil karakterisasi *driver* motor dalam Gambar 3.6 menunjukkan bahwa *driver* motor VNH3SP30 memiliki kualitas yang bagus karena hubungan antara *duty cycle* dengan tegangan keluarannya relatif linier.

3.5.3 Karakterisasi *Gyroscope* WT61C

Karakterisasi *gyroscope* WT61C ini dilakukan untuk mengetahui tingkat akurasi sensor *gyroscope* yang digunakan dalam membaca perubahan orientasi robot. Pembacaan perubahan orientasi robot dapat dilakukan dengan membaca nilai Yaw. Oleh karena itu karakterisasi *gyroscope* dilakukan dengan mengamati perubahan nilai Yaw hasil pembacaan sensor *gyroscope* WT61C.

Untuk melakukan karakterisasi ini digunakan alat-alat sebagai berikut:

1. Perangkat komputer
2. Modul gyroscope WT61C

3. Modul USB to TTL
4. Aplikasi miniIMU v5.0
5. Busur derajat 360°

Prosedur pengujiannya adalah sebagai berikut:

1. Memasang modul *gyroscope* WT61C pada titik pusat busur derajat pertama
2. Memasang busur derajat kedua pada bidang datar sebagai acuan
3. Memposisikan sensor *gyroscope* WT61C pada sudut acuan 0°
4. Membuka aplikasi miniIMU v5.0 dan kalibrasi sensor *gyroscope* WT61C
5. Memutar busur derajat pertama pada sudut yang diinginkan dengan acuan nilai sudut pada busur derajat kedua untuk mengetahui tingkat akurasi perubahan orientasi.

Hasil karakterisasi modul *gyroscope* WT61C dapat dilihat dalam Tabel 3-1.

Tabel 3-1 Perbandingan Sudut Yaw Pembacaan Terhadap Sudut Yaw Aktual

Aktual (°)	Yaw (°)	*Error* (°)	Aktual (°)	Yaw (°)	*Error* (°)
0	0	0	0	0	0
5	4,8	0,2	-5	-4,98	-0,02
10	9,73	0,27	-10	-9,84	-0,16
15	14,97	0,03	-15	-14,52	-0,48
20	19,62	0,38	-20	-19,53	-0,47
25	24,48	0,52	-25	-25,06	0,06
30	29,37	0,63	-30	-30,26	0,26
35	35,19	-0,19	-35	-35,29	0,29
40	39,56	0,44	-40	-40,35	0,35
45	44,96	0,04	-45	-45,29	0,29
50	49,86	0,14	-50	-49,49	-0,51
55	55,17	-0,17	-55	-54,59	-0,41
60	59,36	0,64	-60	-59,66	-0,34
65	64,93	0,07	-65	-64,46	-0,54
70	69,75	0,25	-70	-69,78	-0,22
75	74,73	0,27	-75	-74,7	-0,3
80	80,09	-0,09	-80	-79,94	-0,06
85	84,46	0,54	-85	-84,48	-0,52
90	89,2	0,8	-90	-89,26	-0,74
Error Rata-rata (°)		0,251	*Error* Rata-rata (°)		-0,185

Hasil karakterisasi dalam Tabel 3-1 menunjukkan bahwa sensor gyroscope WT61C memiliki performa yang baik. Rata-rata *error* pembacaan sudut Yaw adalah 0,251° pada sudut positif dan -0,185 pada sudut negatif.

3.5.4 Karakterisasi Odometri

Karakterisasi odometri dilakukan untuk mengetahui tingkat akurasi perhitungan odometri yang digunakan dalam membaca perubahan posisi robot.

Untuk melakukan karakterisasi ini digunakan alat-alat sebagai berikut:

1. Perangkat komputer
2. *Rotary external* 3 buah
3. STM32F407VGT6
4. ST-Link v2
5. *Software* STM Studio
6. Modul *gyroscope* WT61C
7. Penggaris

Prosedur pengujiannya adalah sebagai berikut:

1. Menyambungkan ST-Link v2 dengan *mikrokontroler* dan komputer.
2. Membuka *software* STM Studio, *import* variabel koordinat X dan Y
3. Mengaktifkan mode akuisisi data pada software STM Studio dan gerakkan robot sesuai target
4. Mencatat perubahan posisi robot pada koordinat X dan Y

Hasil karakterisasi odometri dapat dilihat dalam Tabel 3-2.

Tabel 3-2 Perubahan Nilai Koordinat Pembacaan Terhadap Koordinat Aktual

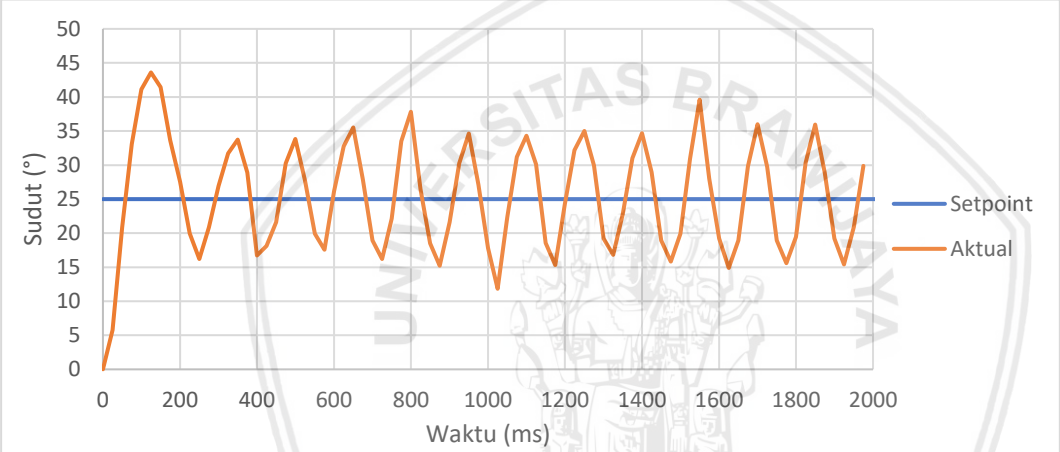
X Aktual (cm)	X Baca (cm)	X *Error* (cm)	Y Aktual (cm)	Y Baca (cm)	Y *Error* (cm)
250	248	2	250	251	1
500	498	2	500	497	3
750	747	3	750	752	2
1000	1003	3	1000	1008	8
1250	1248	2	1250	1255	5
1500	1505	5	1500	1497	3
1750	1747	3	1750	1755	5
2000	1994	6	2000	2005	5
2250	2255	5	2250	2253	3
2500	2503	3	2500	2495	5

Hasil karakterisasi odometri dalam Tabel 3-2 menunjukkan bahwa nilai pembacaan odometri memiliki tingkat akurasi yang baik. Rata-rata *error* pembacaan posisi adalah 3,7 cm. Sedangkan *error* maksimal adalah 6 cm pada sumbu X dan 8 cm pada sumbu Y.

3.6 Penentuan Parameter Kontroler PI Orientasi dengan Ziegler Nichols 2

Sesuai dengan aturan Ziegler-Nichols (ZN) metode kedua, untuk mendapatkan nilai parameter kontrol K_p dan K_i diperlukan proses sebagai berikut:

1. Mengubah nilai K_p dan K_i awal sama dengan 0
2. Memasukkan nilai *setpoint* (Sudut Yaw 25°)
3. Menaikkan nilai K_p hingga sistem berosilasi secara berkesinambungan seperti dalam Gambar 3.7.



Gambar 3.7 Respon Sistem Berosilasi Secara Berkesinambungan

4. Mengamati respon sistem dan mencari K_{cr} dan P_{cr} sebagai berikut:

   K_{cr} = 7

   P_{cr} = 0,15 s

5. Menghitung nilai K_p dan K_i sebagai berikut:

   $K_p = 0{,}45 \times K_{cr}$

   $K_p = 0{,}45 \times 7$

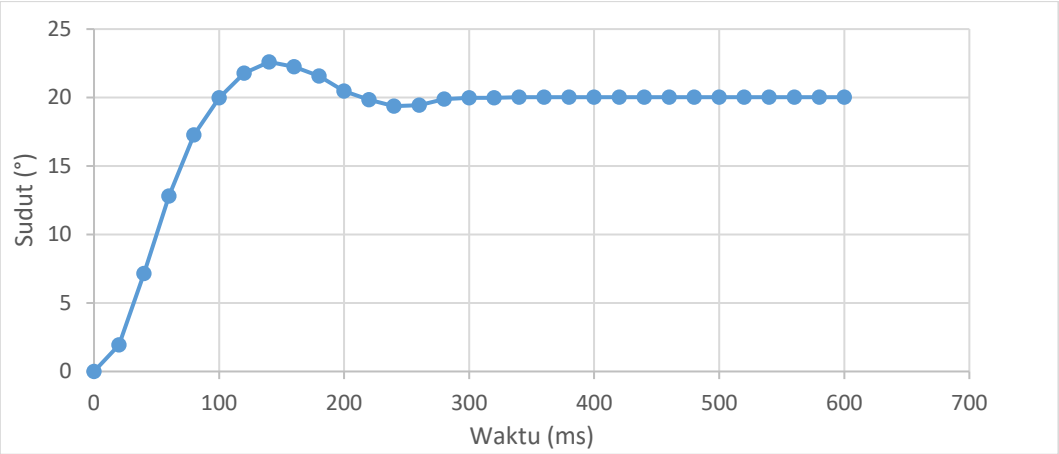
   $K_p = 3{,}15$

   $K_i = \frac{1}{1{,}2} \times P_{cr}$

   $K_i = \frac{1}{1{,}2} \times 0{,}15$

   $K_i = 0{,}125$

6. Mengubah nilai parameter K_p dan K_i sesuai dengan hasil perhitungan dan mengamati respon sistem seperti dalam Gambar 3.8.

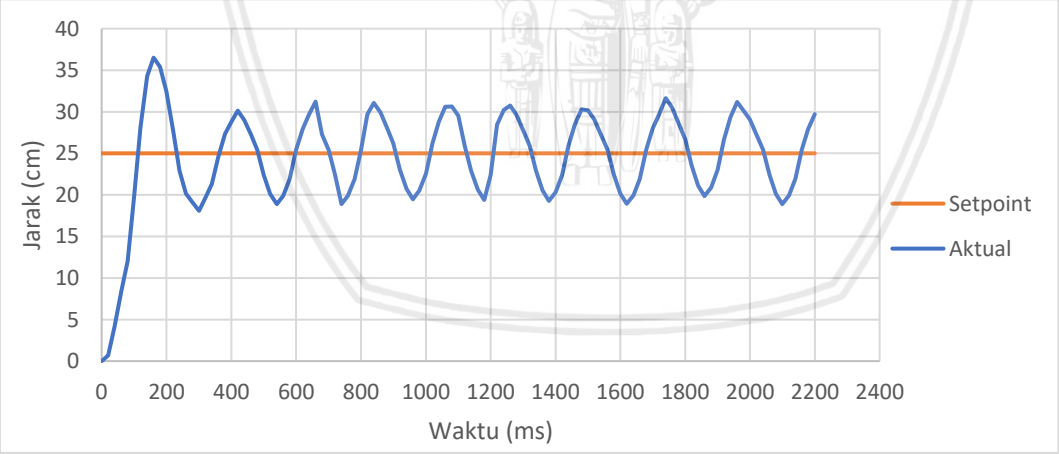


Gambar 3.8 Respon Sistem Hasil Perhitungan ZN Metode Kedua

3.7 Penentuan Parameter Kontroler PI Posisi dengan Ziegler Nichols 2

Sesuai dengan aturan Ziegler-Nichols (ZN) metode kedua, untuk mendapatkan nilai parameter kontrol K_p dan K_i diperlukan proses sebagai berikut:

1. Mengubah nilai K_p dan K_i awal sama dengan 0
2. Memasukkan nilai *setpoint* (Koordinat Y 25cm)
3. Menaikkan nilai K_p hingga sistem berosilasi secara berkesinambungan seperti dalam Gambar 3.9.



Gambar 3.9 Respon Sistem Berosilasi Secara Berkesinambungan

4. Mengamati respon sistem dan mencari K_{cr} dan P_{cr} sebagai berikut:

   K_{cr} = 15

   P_{cr} = 0,2 s

5. Mengitung nilai K_p dan K_i sebagai berikut:

   $K_p = 0{,}45 \times K_{cr}$

   $K_p = 0{,}45 \times 15$

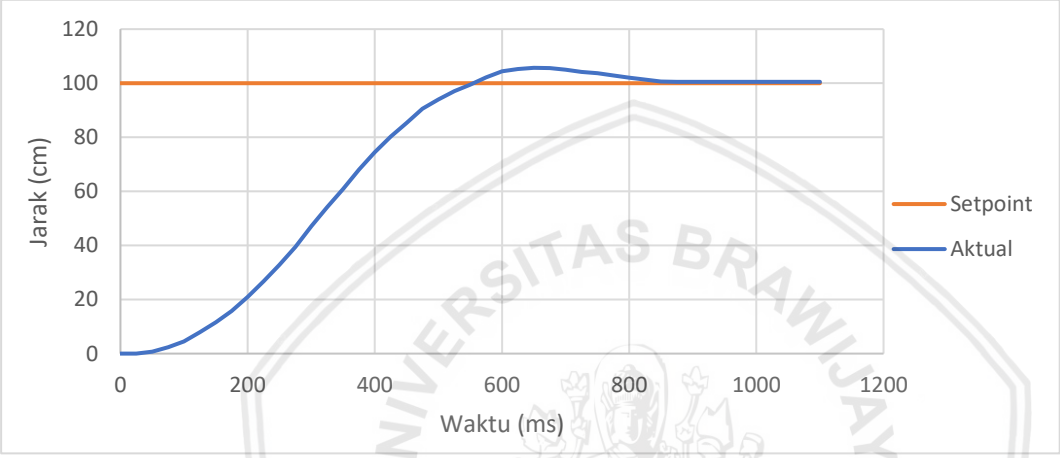
$$K_p = 6{,}75$$

$$K_i = \frac{1}{1{,}2} \times P_{cr}$$

$$K_i = \frac{1}{1{,}2} \times 0{,}2$$

$$K_i = 0{,}166$$

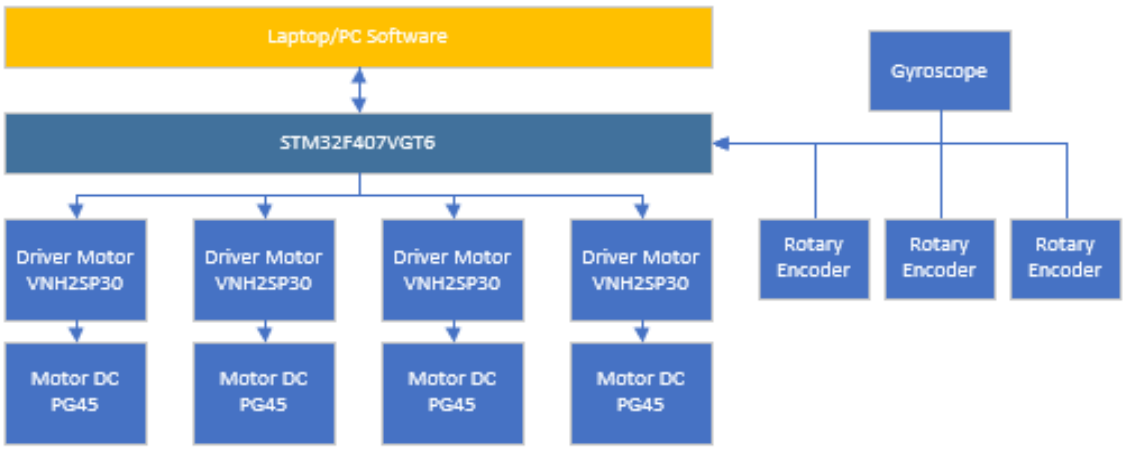
6. Mengubah nilai parameter K_p dan K_i sesuai dengan hasil perhitungan dan mengamati respon sistem seperti dalam Gambar 3.10.



Gambar 3.10 Respon Sistem Hasil Perhitungan ZN Metode Kedua

3.8 Perancangan dan Pembuatan Perangkat Keras

Perancangan dan pembuatan perangkat keras adalah tahap dasar sebelum terbentuknya suatu sistem dengan algoritma pemrogramannya. Hal ini bertujuan supaya seluruh perangkat keras yang digunakan dapat saling terintegrasi dan berjalan sesuai dengan yang direncanakan. Skema perancangan dan pembuatan perangkat keras dapat dilihat dalam Gambar 3.11.



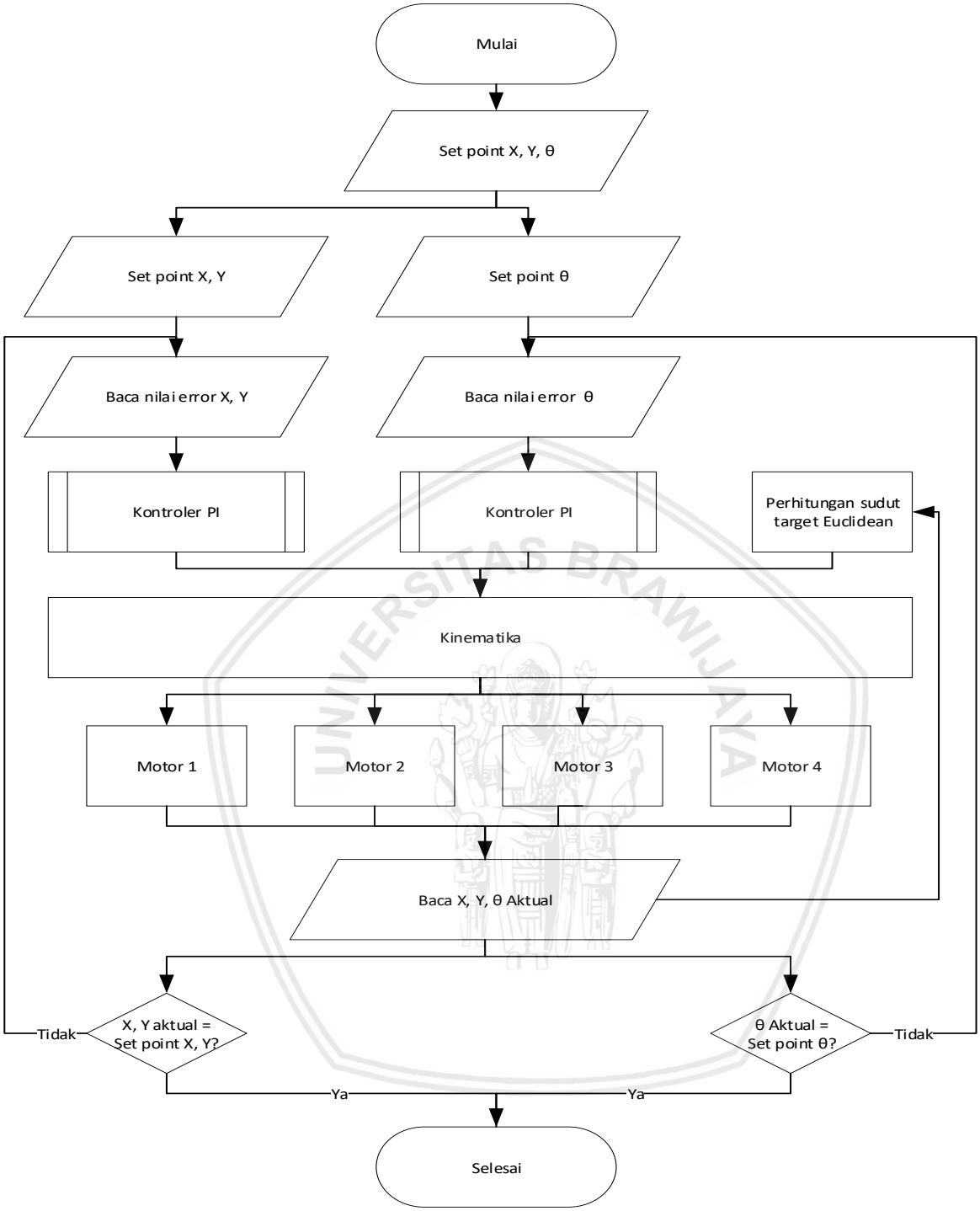
Gambar 3.11 Skema Perangkat Keras Robot MR1

Modul elektronik yang digunakan adalah sebagai berikut:

1. Laptop/PC *Software* digunakan sebagai media pembuatan program serta akuisisi data.
2. Mikrokontroler STM32F407VGT6 sebagai perangkat kontroler
3. *Driver* motor VNH2SP30 digunakan sebagai pengontrol arah putar dan tegangan keluaran ke motor
4. Motor DC PG45 sebagai penggerak utama robot
5. Sensor *gyroscope* sebagai sensor perubahan orientasi robot
6. *Rotary encoder* sebagai sensor posisi robot

3.9 Pembuatan Perangkat Lunak

Dari blok diagram sistem dalam Gambar 3.1, dapat dijelaskan secara rinci dalam *flowchart* mengenai proses yang berjalan pada robot. Hal yang dijelaskan meliputi penentuan *setpoint* koordinat target dan orientasi hingga eksekusi akhir pada motor penggerak robot. Langkah pertama yang dilakukan adalah menentukan *setpoint*, kemudian sistem harus menghitung nilai *error* yang didapat dari perhitungan selisih antara nilai *setpoint* dengan respon *plant*. Setelah nilai *error* diketahui, kontroler akan berkerja mengurangi nilai *error* tersebut. Hasil perhitungan kontroler selanjutnya menjadi input kinematika robot. Selain itu, perhitungan sudut target Euclidean dilakukan untuk menentukan sudut arah gerak robot. Kemudian kinematika robot akan menghitung nilai keluaran tiap motor untuk menggerakkan robot menuju target yang diinginkan. Selama robot melakukan pergerakan, program harus terus membaca perubahan posisi dan orientasi dengan menggunakan sensor yang terpasang. Hasil pembacaan perubahan posisi dan orientasi robot kemudian dibandingkan dengan nilai *setpoint* yang telah ditentukan. Ketika nilai perubahan posisi dan orientasi belum mencapai target, maka program akan kembali menjalankan fungsi baca nilai *error* dan aksi kontroler. *Flowchart* keseluruhan program yang diimplementasikan kedalam mikrokontroler STM32F407VGT6 dapat dilihat dalam Gambar 3.12.



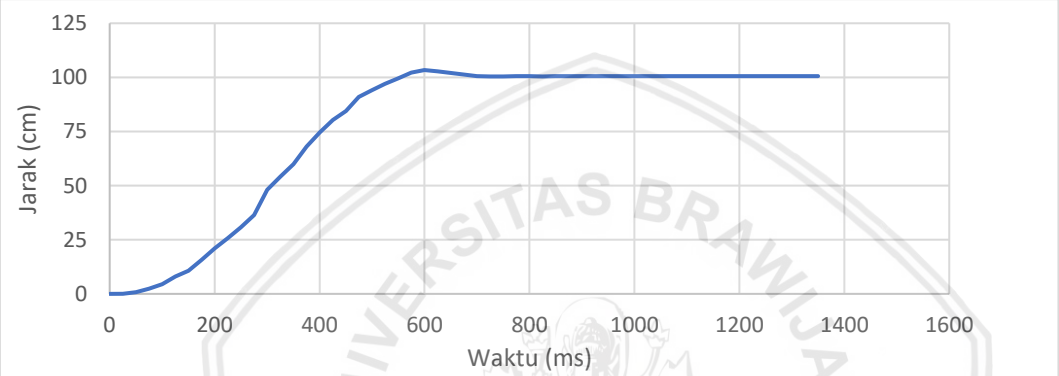
Gambar 3.12 *Flowchart* Sistem Secara Keseluruhan

BAB IV
HASIL DAN PEMBAHASAN

Untuk mengetahui tingkat keberhasilan sebuah sistem dan kesesuaian hasil implementasi dengan perancangan maka harus dilakukan pengujian alat secara keseluruhan. Berikut adalah beberapa pengujian yang dilakukan pada penelitian ini:

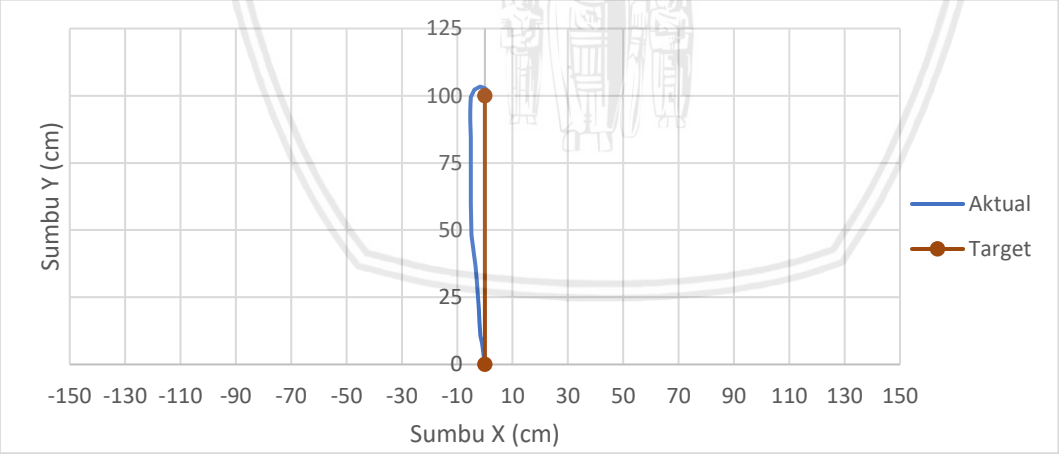
4.1 Pengujian dengan *Setpoint* Koordinat Y 100 cm

Hasil pengujian dengan *setpoint* koordinat Y100cm dapat dilihat dalam Gambar 4.1.



Gambar 4.1 Respon Sistem dengan *Setpoint* Y100cm

Lintasan robot dapat dilihat dalam Gambar 4.2.



Gambar 4.2 Lintasan Robot dengan *Setpoint* Y100cm

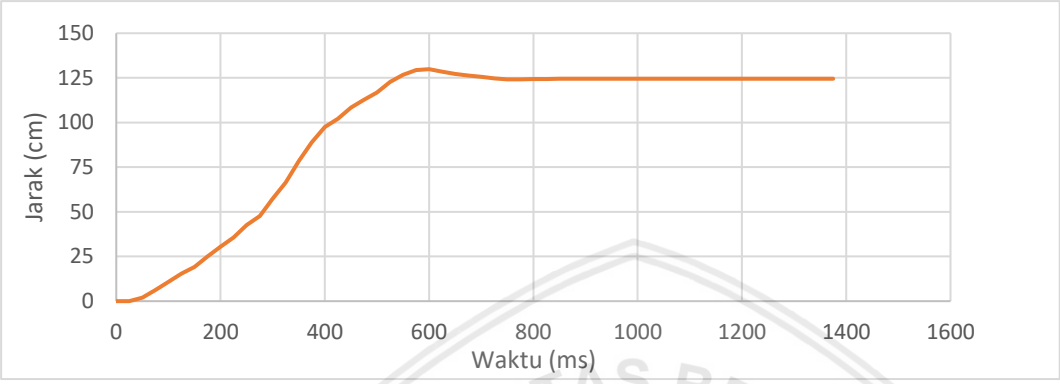
Berdasarkan respon sistem dalam Gambar 4.1 diperoleh nilai *settling time* sebesar 0,85 s, *overshoot* sebesar 3,4 %, dan *error steady state* 0,59 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|100{,}59 - 100|}{100} \times 100\ \%$$
$$= 0{,}59\ \%$$

$$overshoot\ (\%) = \frac{103{,}40 - 100}{100} \times 100\ \%$$
$$= 3{,}4\ \%$$

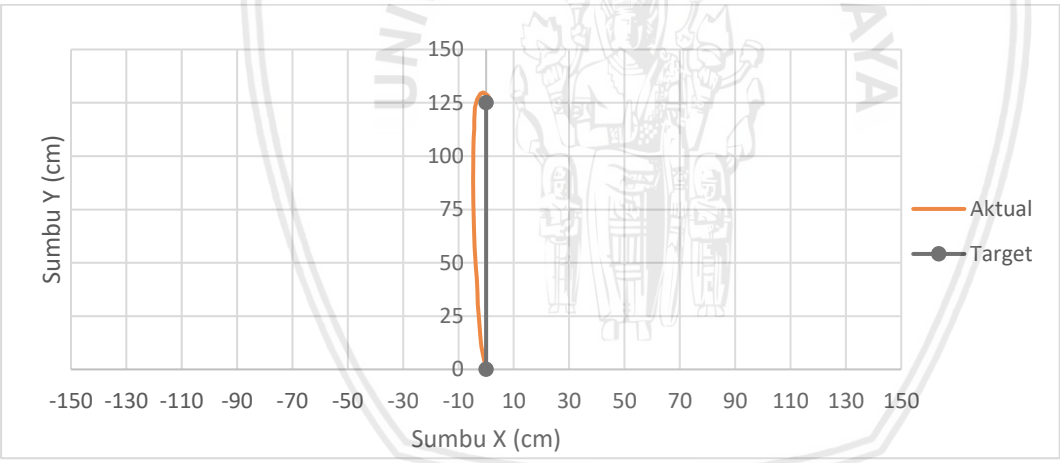
4.2 Pengujian dengan *Setpoint* Koordinat Y 125 cm

Hasil pengujian dengan *setpoint* koordinat Y125cm dapat dilihat dalam Gambar 4.3.



Gambar 4.3 Respon Sistem dengan *Setpoint* Y125cm

Lintasan robot dapat dilihat dalam Gambar 4.4.



Gambar 4.4 Lintasan Robot dengan *Setpoint* Y125cm

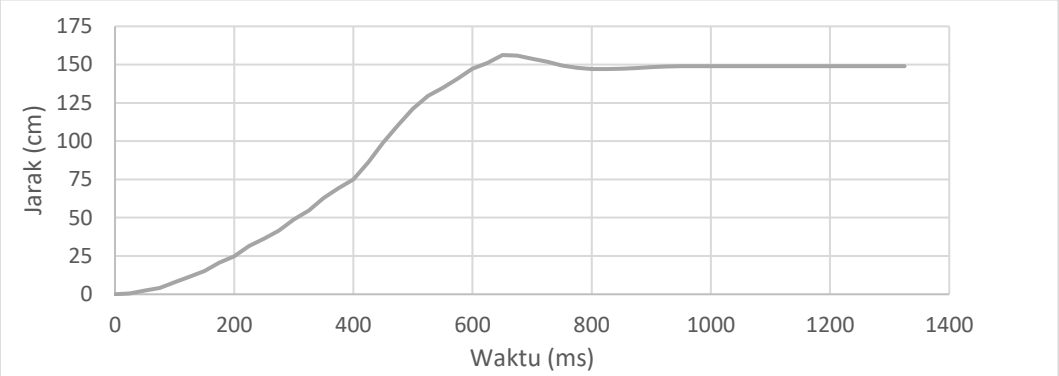
Berdasarkan respon sistem dalam Gambar 4.3 diperoleh nilai *settling time* sebesar 0,85 s, *overshoot* sebesar 3,9 %, dan *error steady state* 0,44 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|124{,}45 - 125|}{125} \times 100\ \%$$
$$= 0{,}44\ \%$$

$$overshoot\ (\%) = \frac{129{,}87 - 125}{125} \times 100\ \%$$
$$= 3{,}9\ \%$$

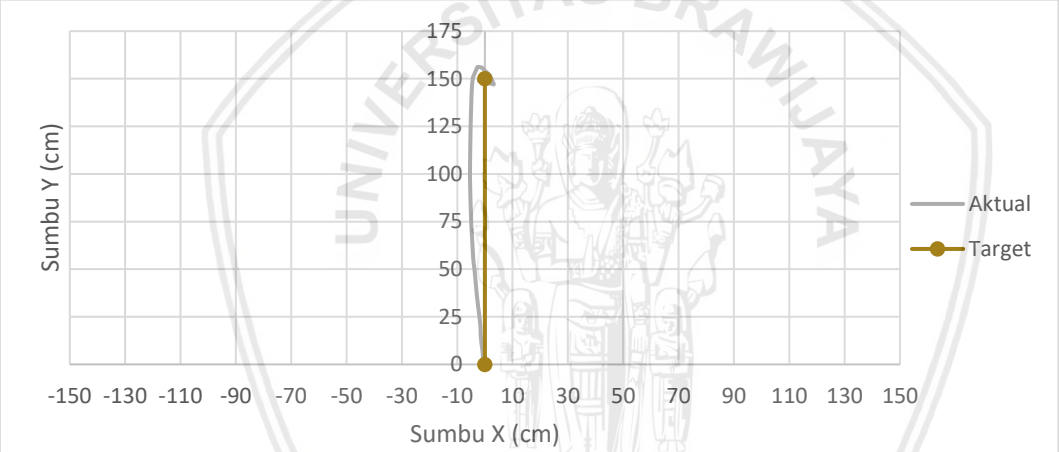
4.3 Pengujian dengan *Setpoint* Koordinat Y 150 cm

Hasil pengujian dengan *setpoint* koordinat Y150cm dapat dilihat dalam Gambar 4.5.



Gambar 4.5 Respon Sistem dengan *Setpoint* Y150cm

Lintasan robot dapat dilihat dalam Gambar 4.6.



Gambar 4.6 Lintasan Robot dengan *Setpoint* Y150cm

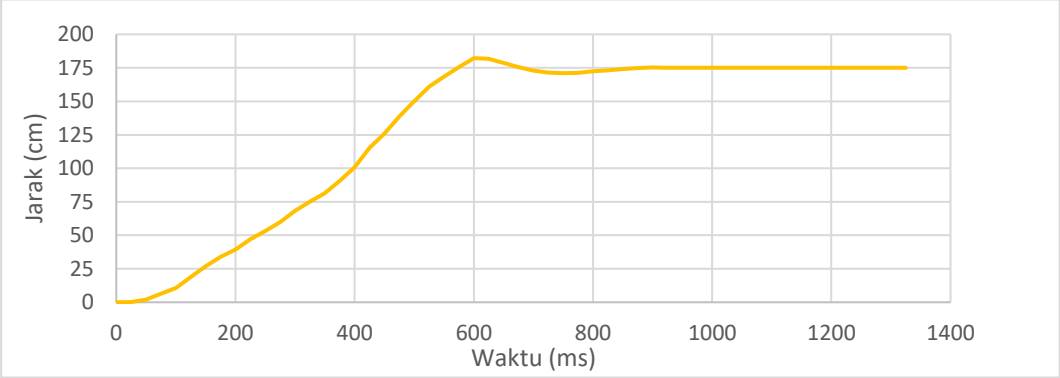
Berdasarkan respon sistem dalam Gambar 4.5 diperoleh nilai *settling time* sebesar 1,05 s, *overshoot* sebesar 4,18 %, dan *error steady state* 0,7 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|148{,}95 - 150|}{150} \times 100\ \%$$
$$= 0{,}7\ \%$$

$$overshoot\ (\%) = \frac{156{,}27 - 150}{150} \times 100\ \%$$
$$= 4{,}18\ \%$$

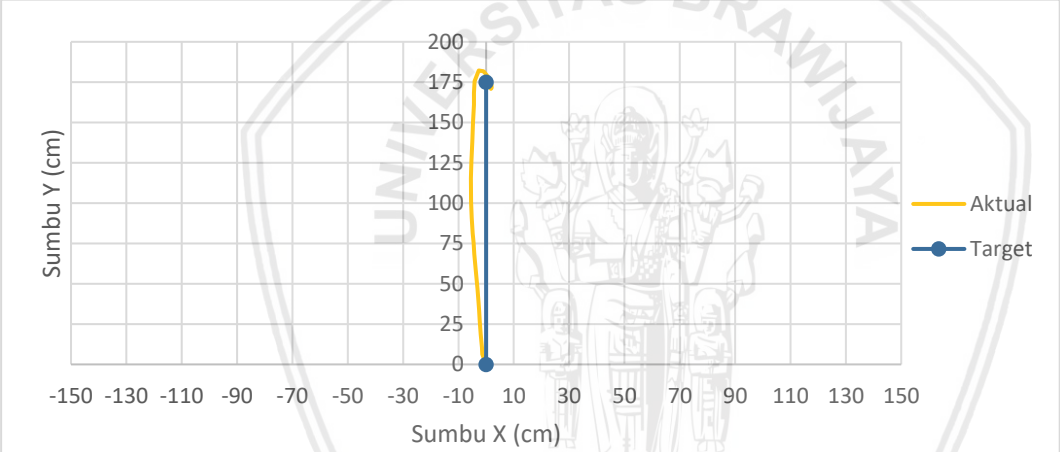
4.4 Pengujian dengan *Setpoint* Koordinat Y 175 cm

Hasil pengujian dengan *setpoint* koordinat Y175cm dapat dilihat dalam Gambar 4.7.



Gambar 4.7 Respon Sistem dengan *Setpoint* Y175cm

Lintasan robot dapat dilihat dalam Gambar 4.8.



Gambar 4.8 Lintasan Robot dengan *Setpoint* Y175cm

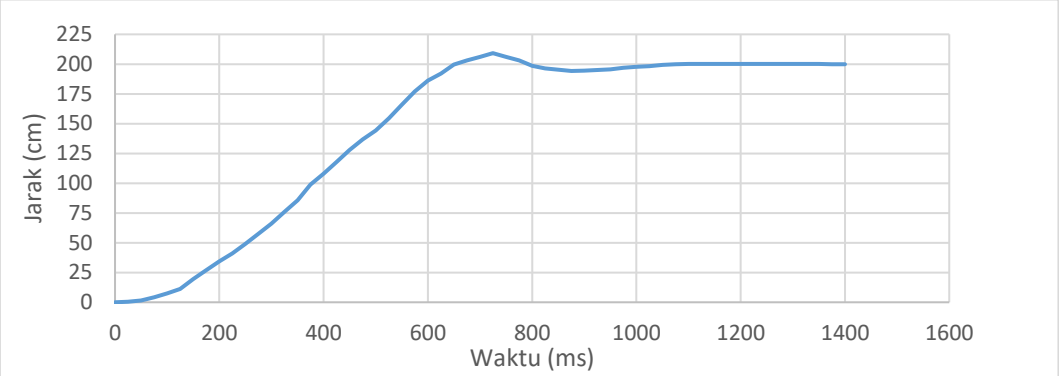
Berdasarkan respon sistem dalam Gambar 4.7 diperoleh nilai *settling time* sebesar 1,05 s, *overshoot* sebesar 4,14 %, dan *error steady state* 0,01 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|175{,}02 - 175|}{175} \times 100\ \%$$
$$= 0{,}01\ \%$$

$$overshoot\ (\%) = \frac{182{,}25 - 175}{175} \times 100\ \%$$
$$= 4{,}14\ \%$$

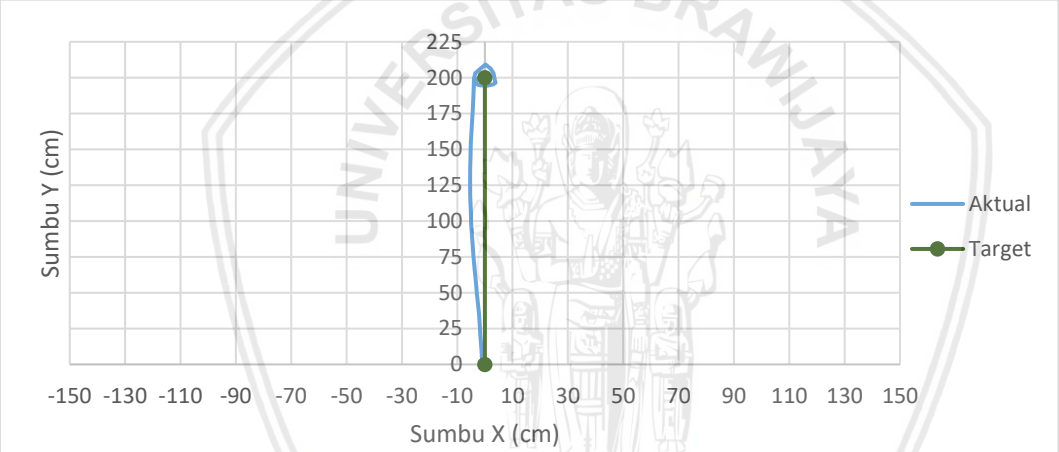
4.5 Pengujian dengan *Setpoint* Koordinat Y 200 cm

Hasil pengujian dengan *setpoint* koordinat Y200cm dapat dilihat dalam Gambar 4.9.



Gambar 4.9 Respon Sistem dengan *Setpoint* Y200cm

Lintasan robot dapat dilihat dalam Gambar 4.10.



Gambar 4.10 Lintasan Robot dengan *Setpoint* Y200cm

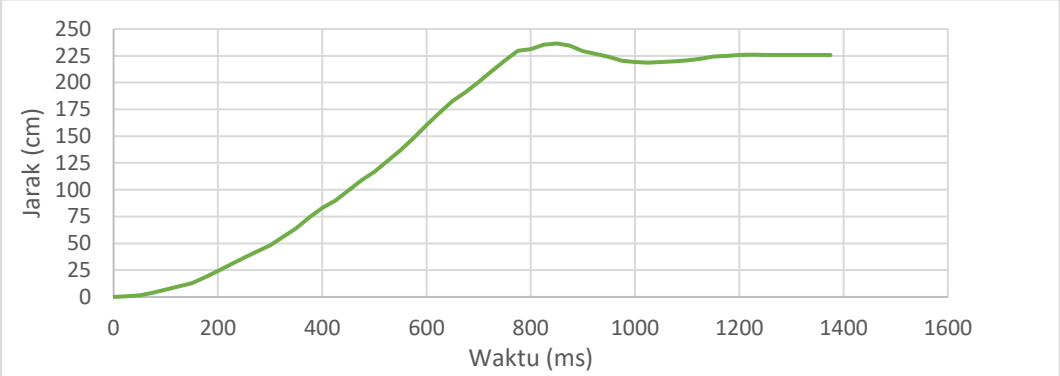
Berdasarkan respon sistem dalam Gambar 4.9 diperoleh nilai *settling time* sebesar 1,15 s, *overshoot* sebesar 4,61 %, dan *error steady state* 0,01 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|200{,}02 - 200|}{200} \times 100\ \%$$
$$= 0{,}01\ \%$$

$$overshoot\ (\%) = \frac{209{,}21 - 200}{200} \times 100\ \%$$
$$= 4{,}61\ \%$$

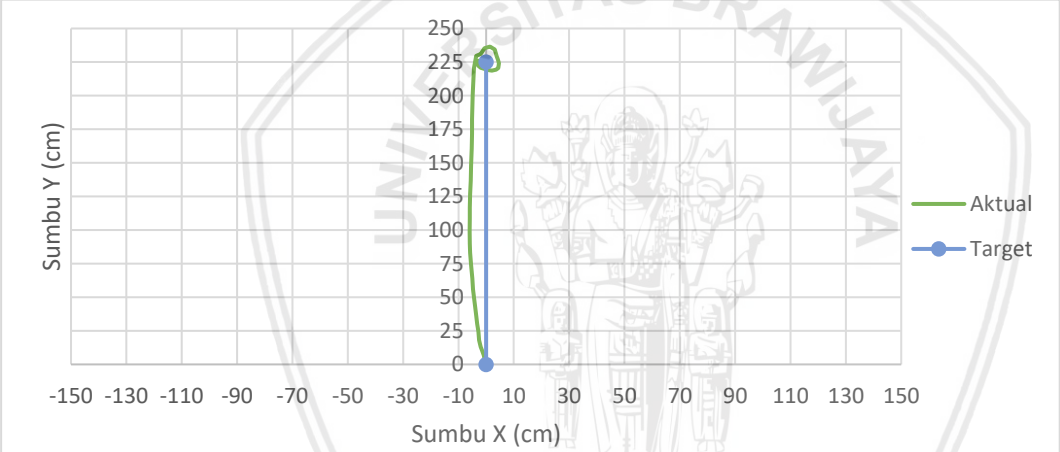
4.6 Pengujian dengan *Setpoint* Koordinat Y 225 cm

Hasil pengujian dengan *setpoint* koordinat Y225cm dapat dilihat dalam Gambar 4.11.



Gambar 4.11 Respon Sistem dengan *Setpoint* Y225cm

Lintasan robot dapat dilihat dalam Gambar 4.12.



Gambar 4.12 Lintasan Robot dengan *Setpoint* Y225cm

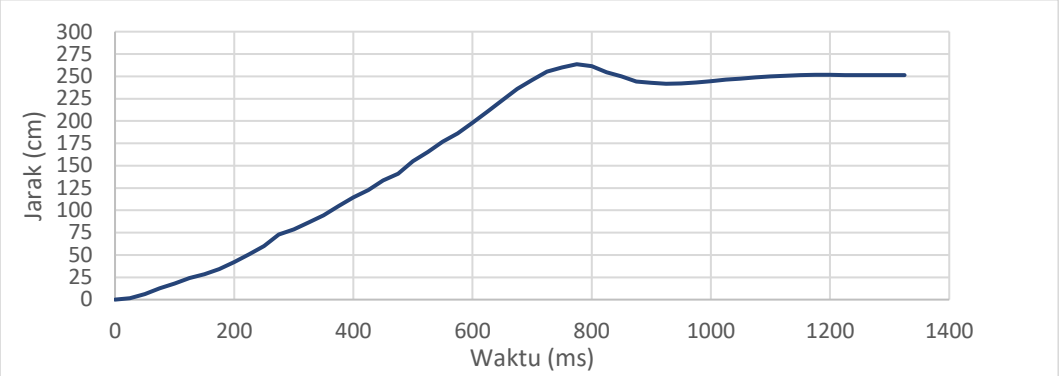
Berdasarkan respon sistem dalam Gambar 4.11 diperoleh nilai *settling time* sebesar 1,3 s, *overshoot* sebesar 5,12 %, dan *error steady state* 0,37 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|225{,}83 - 225|}{225} \times 100\ \%$$
$$= 0{,}37\ \%$$

$$overshoot\ (\%) = \frac{236{,}52 - 225}{225} \times 100\ \%$$
$$= 5{,}12\ \%$$

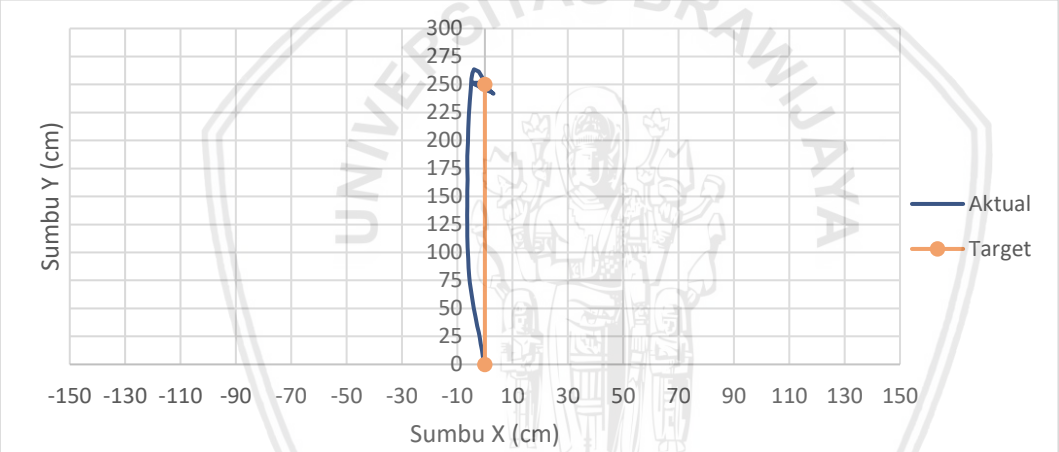
4.7 Pengujian dengan *Setpoint* Koordinat Y 250 cm

Hasil pengujian dengan *setpoint* koordinat Y250cm dapat dilihat dalam Gambar 4.13.



Gambar 4.13 Respon Sistem dengan *Setpoint* Y250cm

Lintasan robot dapat dilihat dalam Gambar 4.14.



Gambar 4.14 Lintasan Robot dengan *Setpoint* Y250cm

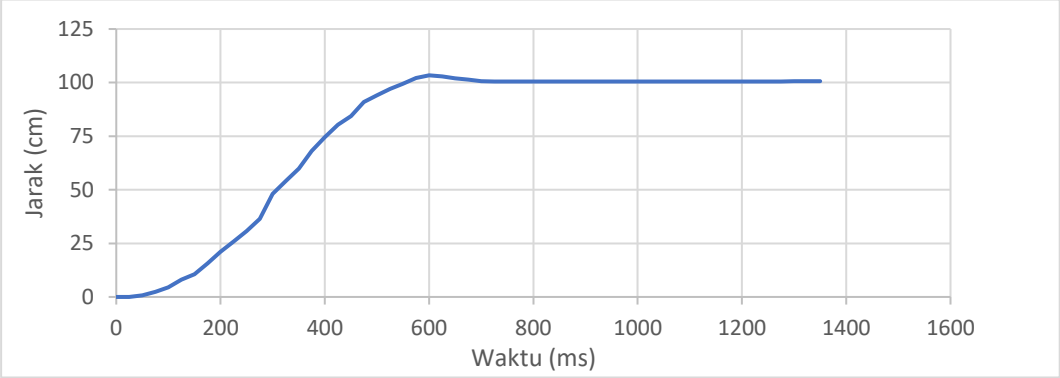
Berdasarkan respon sistem dalam Gambar 4.13 diperoleh nilai *settling time* sebesar 1,275 s, *overshoot* sebesar 5,42 %, dan *error steady state* 0,55 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|251{,}39 - 250|}{250} \times 100\ \%$$
$$= 0{,}55\ \%$$

$$overshoot\ (\%) = \frac{263{,}56 - 250}{250} \times 100\ \%$$
$$= 5{,}42\ \%$$

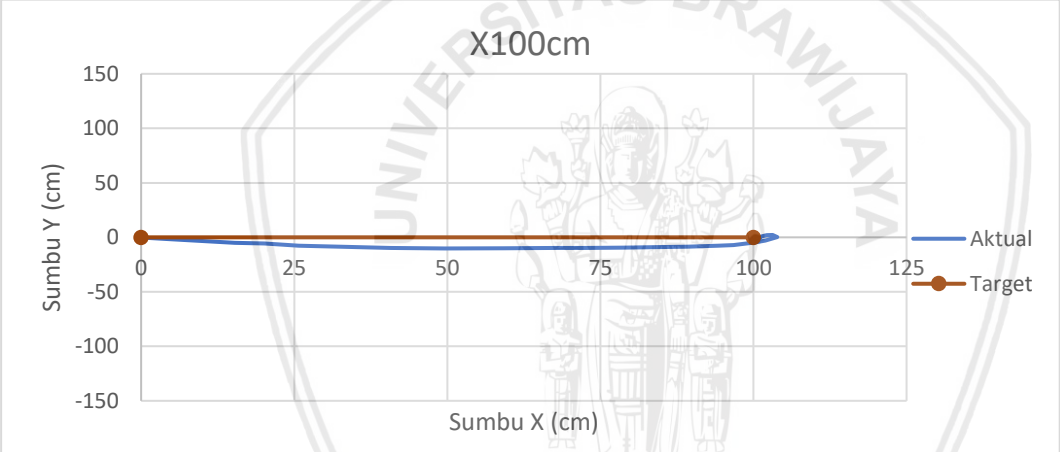
4.8 Pengujian dengan *Setpoint* Koordinat X 100 cm

Hasil pengujian dengan *setpoint* koordinat X100cm dapat dilihat dalam Gambar 4.15.



Gambar 4.15 Respon Sistem dengan *Setpoint* X100cm

Lintasan robot dapat dilihat dalam Gambar 4.16.



Gambar 4.16 Lintasan Robot dengan *Setpoint* X100cm

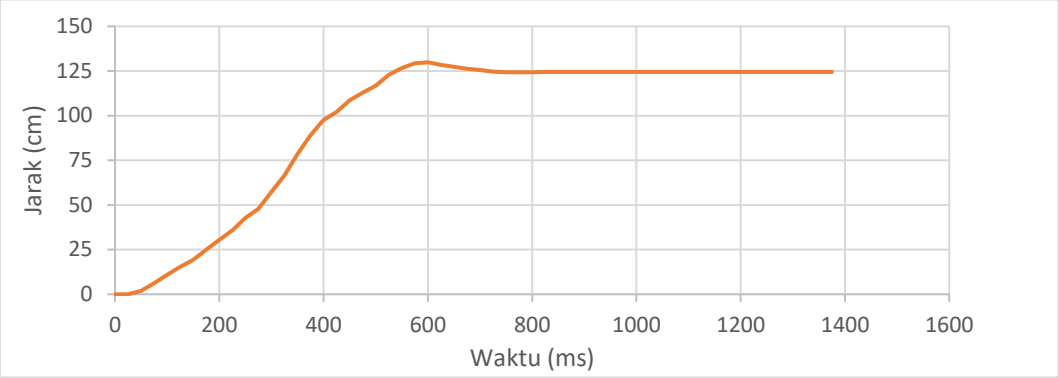
Berdasarkan respon sistem dalam Gambar 4.15 diperoleh nilai *settling time* sebesar 0,85 s, *overshoot* sebesar 3,38 %, dan *error steady state* 1,09 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|101{,}09 - 100|}{100} \times 100\ \%$$
$$= 1{,}09\ \%$$

$$overshoot\ (\%) = \frac{103{,}88 - 100}{100} \times 100\ \%$$
$$= 3{,}38\ \%$$

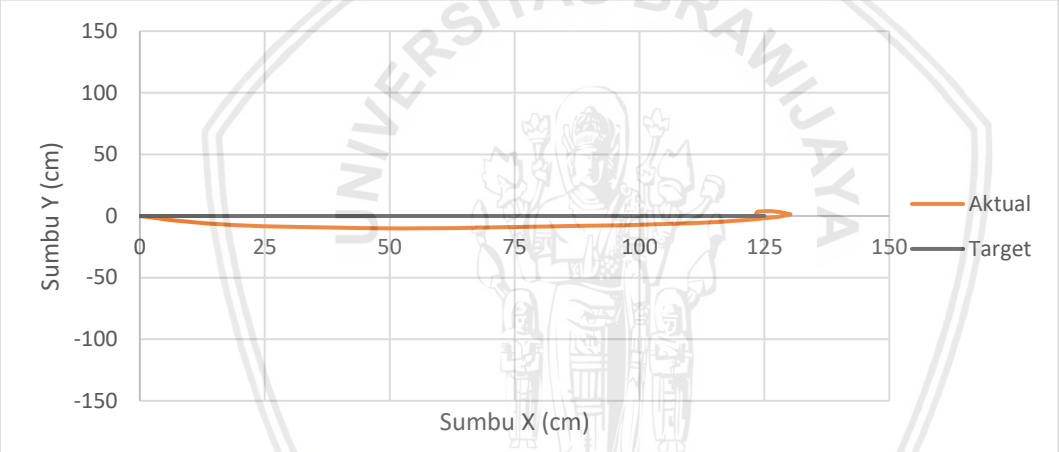
4.9 Pengujian dengan *Setpoint* Koordinat X 125 cm

Hasil pengujian dengan *setpoint* koordinat X125cm dapat dilihat dalam Gambar 4.17.



Gambar 4.17 Respon Sistem dengan *Setpoint* X125cm

Lintasan robot dapat dilihat dalam Gambar 4.18.



Gambar 4.18 Lintasan Robot dengan *Setpoint* X125cm

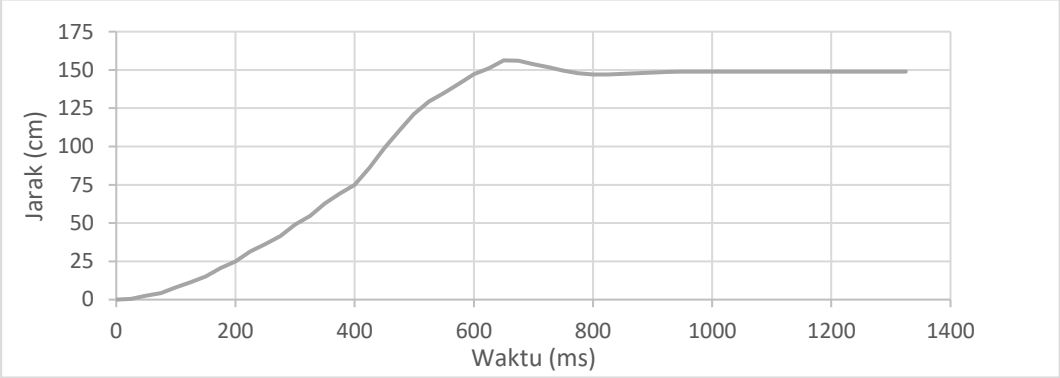
Berdasarkan respon sistem dalam Gambar 4.17 diperoleh nilai *settling time* sebesar 0,85 s, *overshoot* sebesar 4,19 %, dan *error steady state* 0,88 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|123{,}90 - 125|}{125} \times 100\ \%$$
$$= 0{,}88\ \%$$

$$overshoot\ (\%) = \frac{130{,}24 - 125}{125} \times 100\ \%$$
$$= 4{,}19\ \%$$

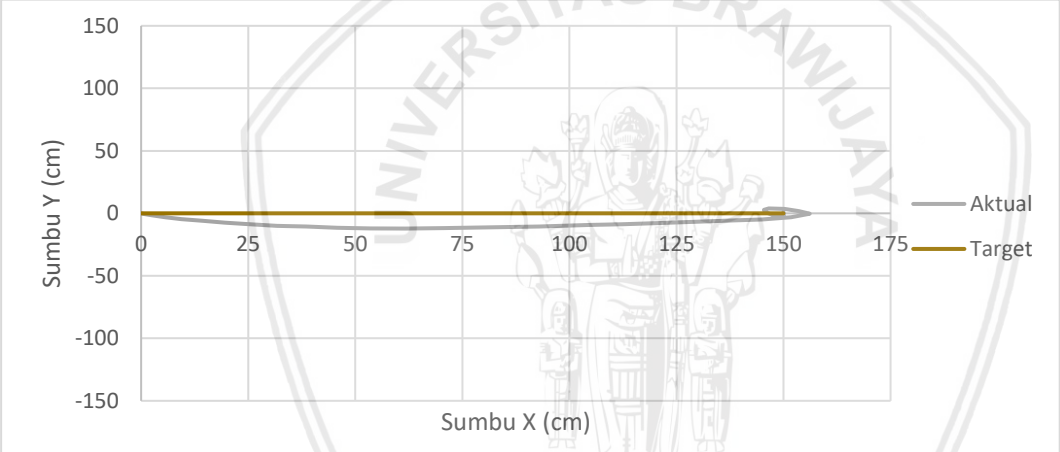
4.10 Pengujian dengan *Setpoint* Koordinat X 150 cm

Hasil pengujian dengan *setpoint* koordinat X150cm dapat dilihat dalam Gambar 4.19.



Gambar 4.19 Respon Sistem dengan *Setpoint* X150cm

Lintasan robot dapat dilihat dalam Gambar 4.20.



Gambar 4.20 Lintasan Robot dengan *Setpoint* X150cm

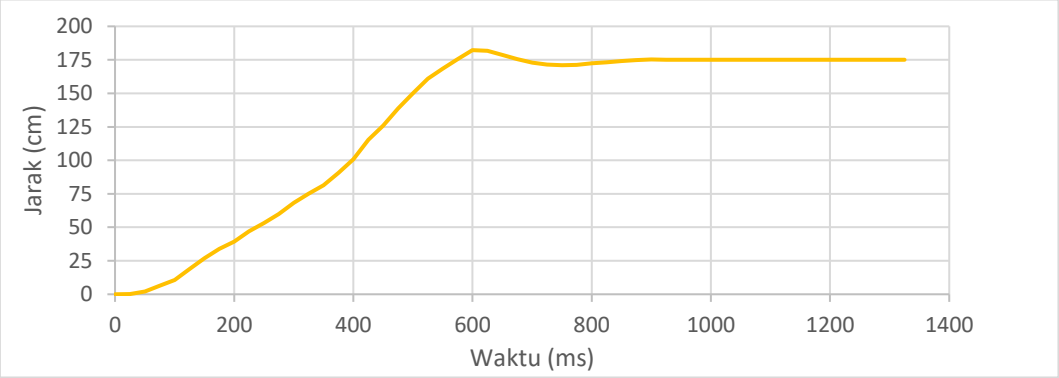
Berdasarkan respon sistem dalam Gambar 4.19 diperoleh nilai *settling time* sebesar 1,1 s, *overshoot* sebesar 4,08 %, dan *error steady state* 0,01 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|150{,}02 - 150|}{150} \times 100\ \%$$
$$= 0{,}01\ \%$$

$$overshoot\ (\%) = \frac{156{,}12 - 150}{150} \times 100\ \%$$
$$= 4{,}08\ \%$$

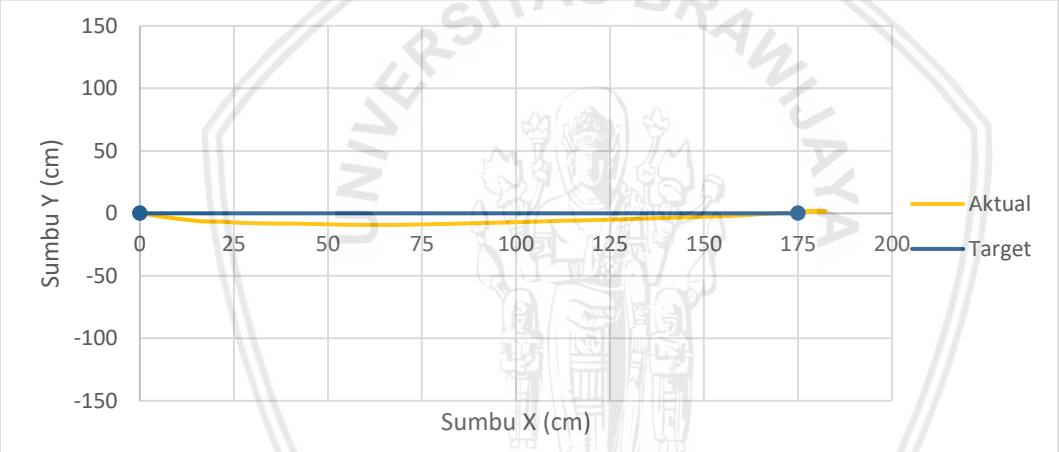
4.11 Pengujian dengan *Setpoint* Koordinat X 175 cm

Hasil pengujian dengan *setpoint* koordinat X175cm dapat dilihat dalam Gambar 4.21.



Gambar 4.21 Respon Sistem dengan *Setpoint* X175cm

Lintasan robot dapat dilihat dalam Gambar 4.22.



Gambar 4.22 Lintasan Robot dengan *Setpoint* X175cm

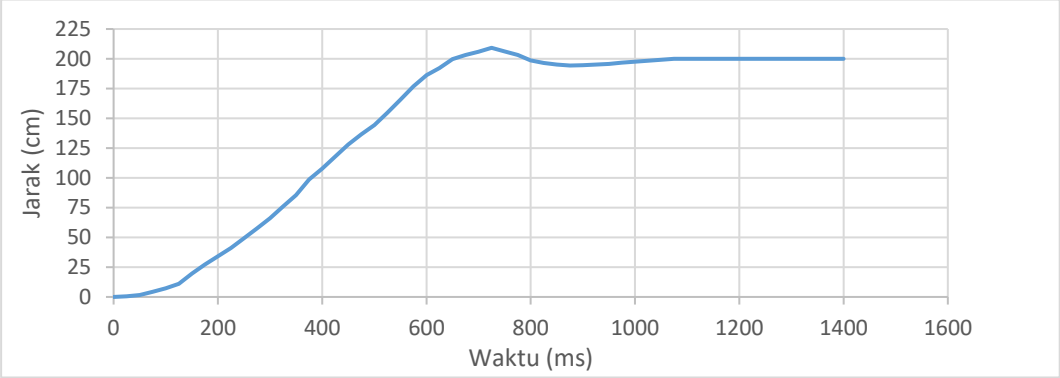
Berdasarkan respon sistem dalam Gambar 4.21 diperoleh nilai *settling time* sebesar 1,05 s, *overshoot* sebesar 4,18 %, dan *error steady state* 0,59 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|173{,}97 - 175|}{175} \times 100\ \%$$
$$= 0{,}59\ \%$$

$$overshoot\ (\%) = \frac{182{,}31 - 175}{175} \times 100\ \%$$
$$= 4{,}18\ \%$$

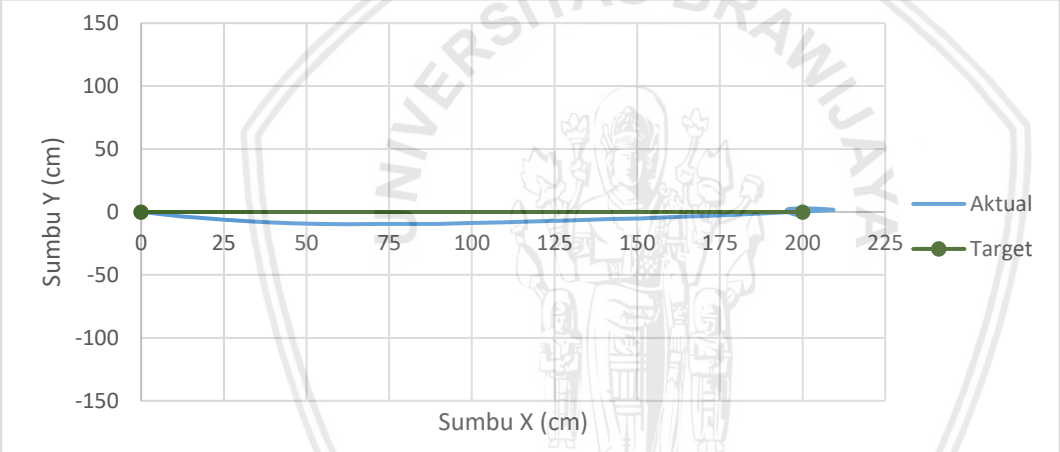
4.12 Pengujian dengan *Setpoint* Koordinat X 200 cm

Hasil pengujian dengan *setpoint* koordinat X200cm dapat dilihat dalam Gambar 4.23.



Gambar 4.23 Respon Sistem dengan *Setpoint* X200cm

Lintasan robot dapat dilihat dalam Gambar 4.24.



Gambar 4.24 Lintasan Robot dengan *Setpoint* X200cm

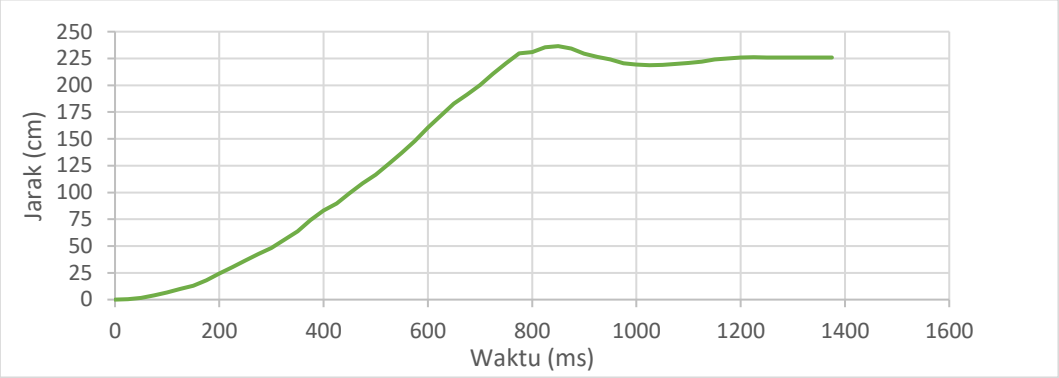
Berdasarkan respon sistem dalam Gambar 4.23 diperoleh nilai *settling time* sebesar 1,15 s, *overshoot* sebesar 4,69 %, dan *error steady state* 0,23 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|199{,}54 - 200|}{200} \times 100\ \%$$
$$= 0{,}23\ \%$$

$$overshoot\ (\%) = \frac{209{,}37 - 200}{200} \times 100\ \%$$
$$= 4{,}69\ \%$$

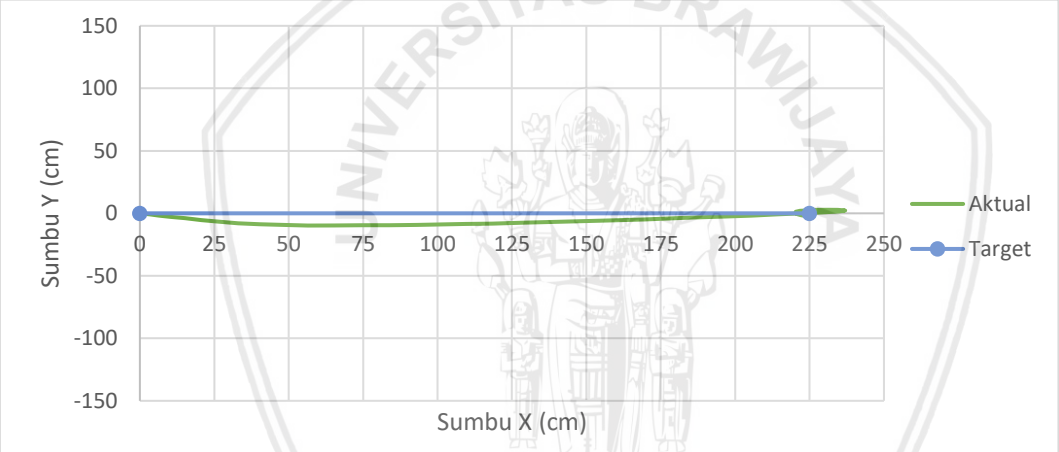
4.13 Pengujian dengan *Setpoint* Koordinat X 225 cm

Hasil pengujian dengan *setpoint* koordinat X225cm dapat dilihat dalam Gambar 4.25.



Gambar 4.25 Respon Sistem dengan *Setpoint* X225cm

Lintasan robot dapat dilihat dalam Gambar 4.26.



Gambar 4.26 Lintasan Robot dengan *Setpoint* X225cm

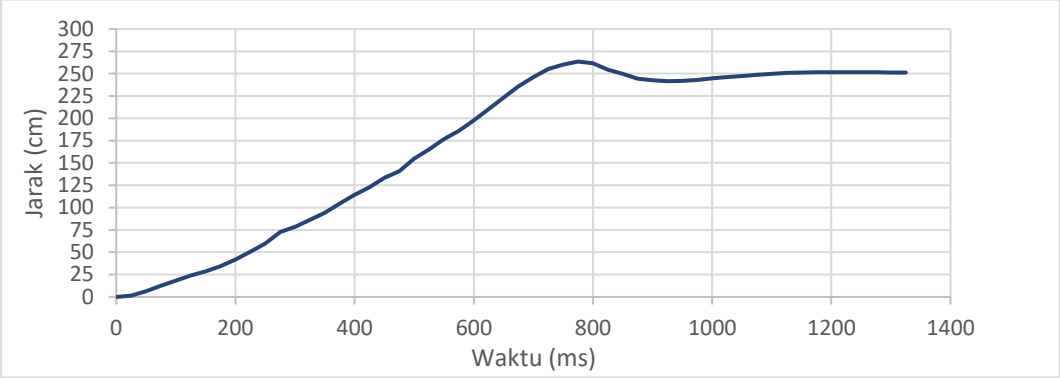
Berdasarkan respon sistem dalam Gambar 4.25 diperoleh nilai *settling time* sebesar 1,3 s, *overshoot* sebesar 5,31 %, dan *error steady state* 0,08 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|225{,}18 - 225|}{225} \times 100\ \%$$
$$= 0{,}08\ \%$$

$$overshoot\ (\%) = \frac{236{,}96 - 225}{225} \times 100\ \%$$
$$= 5{,}31\ \%$$

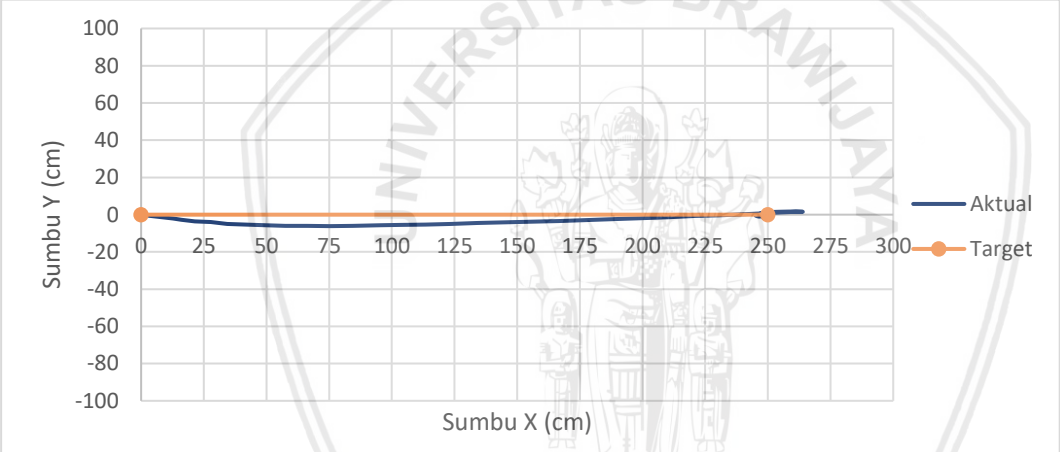
4.14 Pengujian dengan *Setpoint* Koordinat X 250 cm

Hasil pengujian dengan *setpoint* koordinat X250cm dapat dilihat dalam Gambar 4.27.



Gambar 4.27 Respon Sistem dengan *Setpoint* X250cm

Lintasan robot dapat dilihat dalam Gambar 4.28.



Gambar 4.28 Lintasan Robot dengan *Setpoint* X250cm

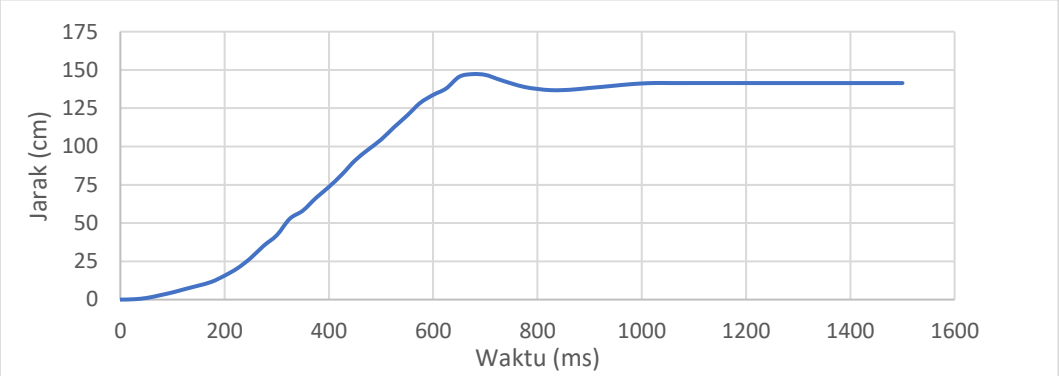
Berdasarkan respon sistem dalam Gambar 4.27 diperoleh nilai *settling time* sebesar 1,3 s, *overshoot* sebesar 5,57 %, dan *error steady state* 0,24 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|249{,}40 - 250|}{250} \times 100\ \%$$
$$= 0{,}24\ \%$$

$$overshoot\ (\%) = \frac{263{,}92 - 250}{250} \times 100\ \%$$
$$= 5{,}57\ \%$$

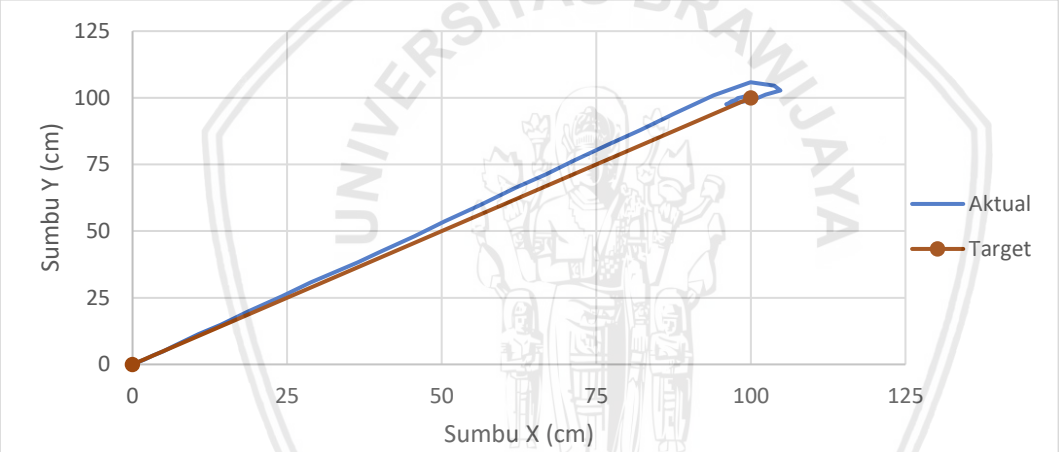
4.15 Pengujian dengan *Setpoint* Koordinat X,Y 100,100 cm

Hasil pengujian dengan *setpoint* koordinat XY100cm dapat dilihat dalam Gambar 4.29.



Gambar 4.29 Respon Sistem dengan *Setpoint* XY100cm

Lintasan robot dapat dilihat dalam Gambar 4.30.



Gambar 4.30 Lintasan Robot dengan *Setpoint* XY100cm

Berdasarkan respon sistem dalam Gambar 4.29 diperoleh nilai *settling time* sebesar 1,05 s, *overshoot* sebesar 4,14 %, dan *error steady state* 0,01 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|141{,}44 - 141{,}42|}{141{,}42} \times 100\ \%$$

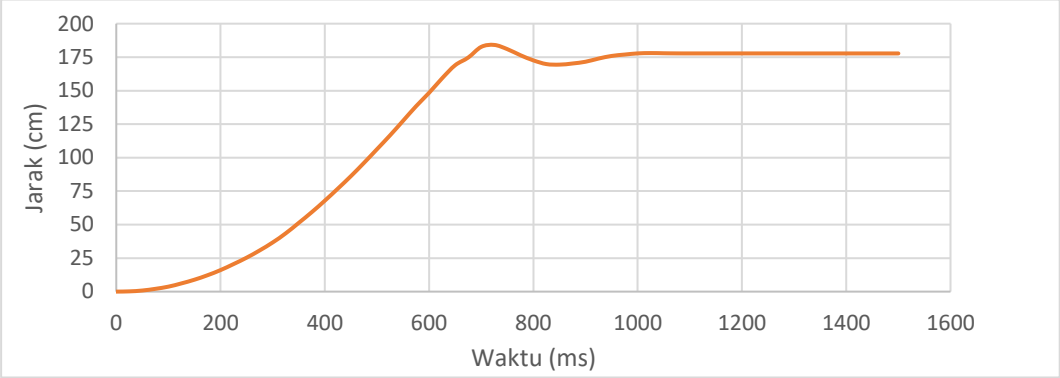
$$= 0{,}01\ \%$$

$$overshoot\ (\%) = \frac{147{,}27 - 141{,}42}{141{,}42} \times 100\ \%$$

$$= 4{,}14\ \%$$

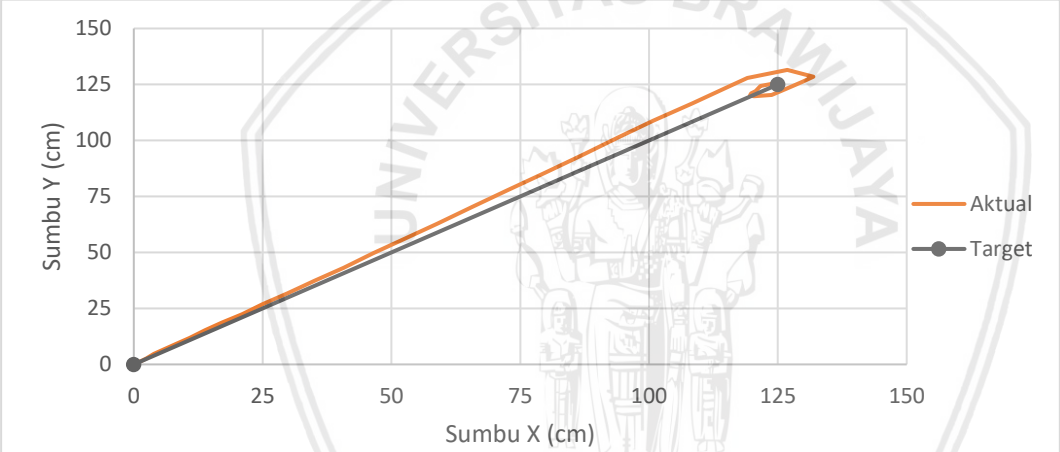
4.16 Pengujian dengan *Setpoint* Koordinat X,Y 125,125 cm

Hasil pengujian dengan *setpoint* koordinat XY125cm dapat dilihat dalam Gambar 4.31.



Gambar 4.31 Respon Sistem dengan *Setpoint* XY125cm

Lintasan robot dapat dilihat dalam Gambar 4.32.



Gambar 4.32 Lintasan Robot dengan *Setpoint* XY125cm

Berdasarkan respon sistem dalam Gambar 4.31 diperoleh nilai *settling time* sebesar 1,1 s, *overshoot* sebesar 4,16 %, dan *error steady state* 0,57 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|177{,}79 - 176{,}78|}{176{,}78} \times 100\ \%$$

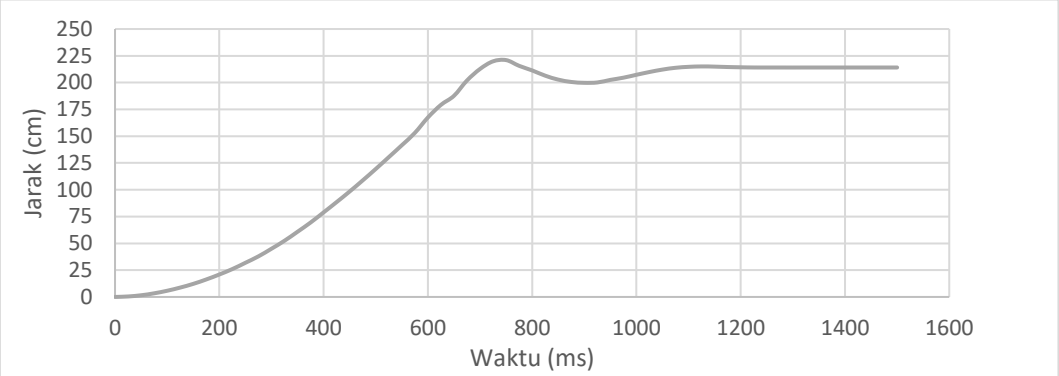
$$= 0{,}57\ \%$$

$$overshoot\ (\%) = \frac{184{,}14 - 176{,}78}{176{,}78} \times 100\ \%$$

$$= 4{,}16\ \%$$

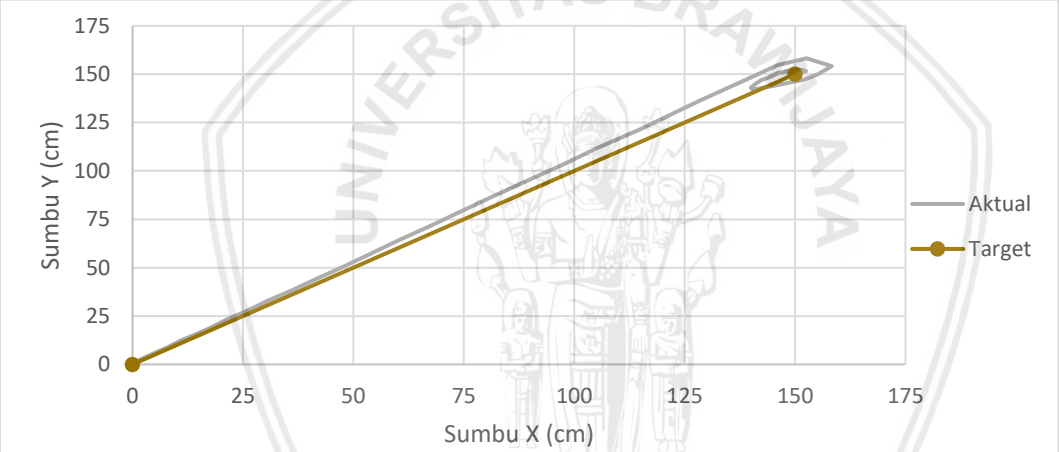
4.17 Pengujian dengan *Setpoint* Koordinat X,Y 150,150 cm

Hasil pengujian dengan *setpoint* koordinat XY150cm dapat dilihat dalam Gambar 4.33.



Gambar 4.33 Respon Sistem dengan *Setpoint* XY150cm

Lintasan robot dapat dilihat dalam Gambar 4.34.



Gambar 4.34 Lintasan Robot dengan *Setpoint* XY150cm

Berdasarkan respon sistem dalam Gambar 4.33 diperoleh nilai *settling time* sebesar 1,175 s, *overshoot* sebesar 4,19 %, dan *error steady state* 0,92 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|214{,}08 - 212{,}13|}{212{,}13} \times 100\ \%$$

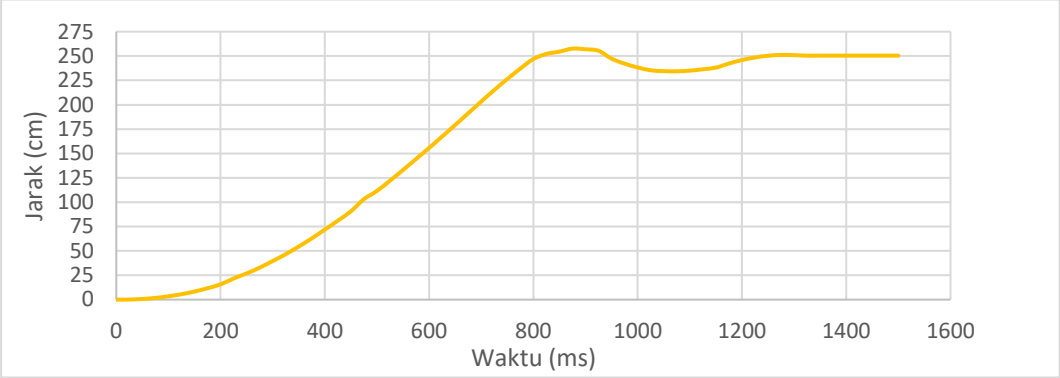
$$= 0{,}92\ \%$$

$$overshoot\ (\%) = \frac{221{,}02 - 212{,}13}{212{,}13} \times 100\ \%$$

$$= 4{,}19\ \%$$

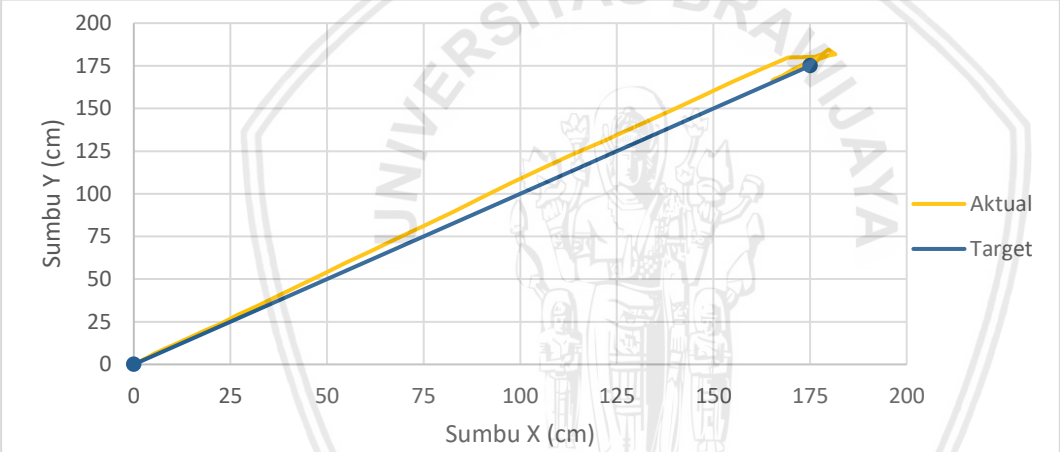
4.18 Pengujian dengan *Setpoint* Koordinat X,Y 175,175 cm

Hasil pengujian dengan *setpoint* koordinat XY175cm dapat dilihat dalam Gambar 4.35.



Gambar 4.35 Respon Sistem dengan *Setpoint* XY175cm

Lintasan robot dapat dilihat dalam Gambar 4.36.



Gambar 4.36 Lintasan Robot dengan *Setpoint* XY175cm

Berdasarkan respon sistem dalam Gambar 4.35 diperoleh nilai *settling time* sebesar 1,25 s, *overshoot* sebesar 4,1 %, dan *error steady state* 1,18 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|250{,}40 - 247{,}49|}{247{,}49} \times 100\ \%$$

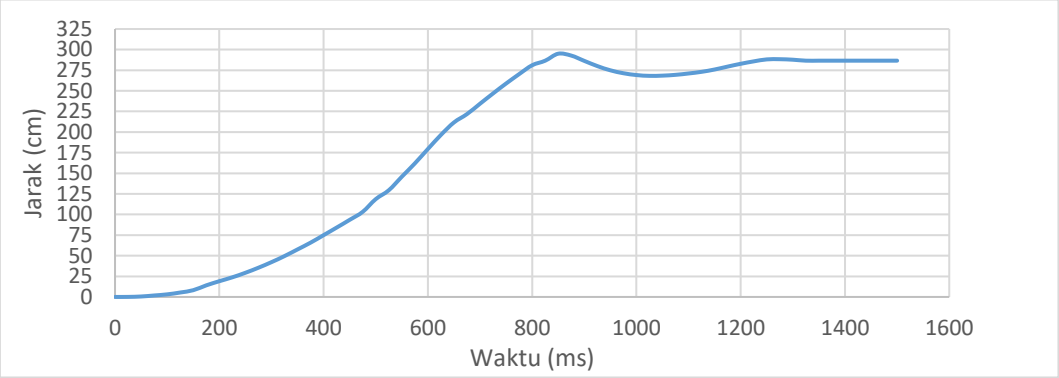
$$= 1{,}18\ \%$$

$$overshoot\ (\%) = \frac{269{,}93 - 247{,}49}{247{,}49} \times 100\ \%$$

$$= 4{,}1\ \%$$

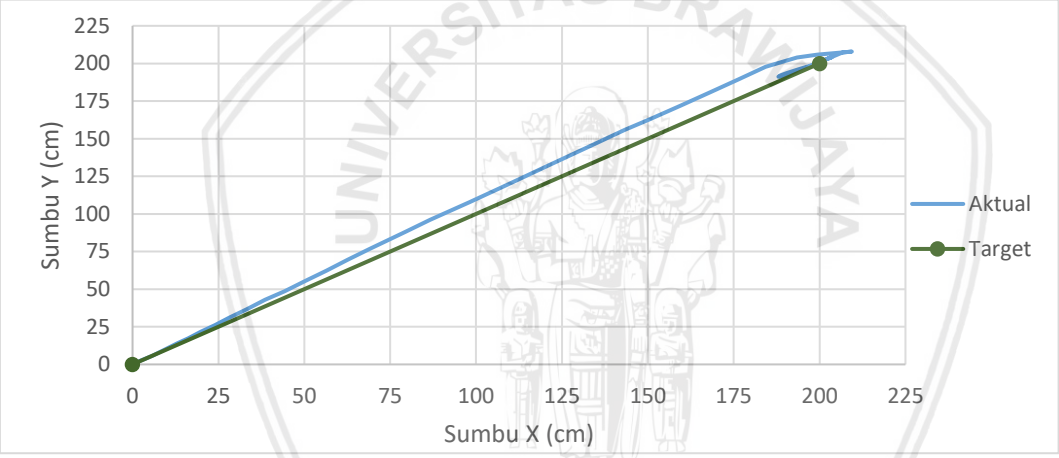
4.19 Pengujian dengan *Setpoint* Koordinat X,Y 200,200 cm

Hasil pengujian dengan *setpoint* koordinat XY200cm dapat dilihat dalam Gambar 4.37.



Gambar 4.37 Respon Sistem dengan *Setpoint* XY200cm

Lintasan robot dapat dilihat dalam Gambar 4.38.



Gambar 4.38 Lintasan Robot dengan *Setpoint* XY200cm

Berdasarkan respon sistem dalam Gambar 4.37 diperoleh nilai *settling time* sebesar 1,325 s, *overshoot* sebesar 4,31 %, dan *error steady state* 1,33 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|286{,}61 - 282{,}84|}{282{,}84} \times 100\ \%$$

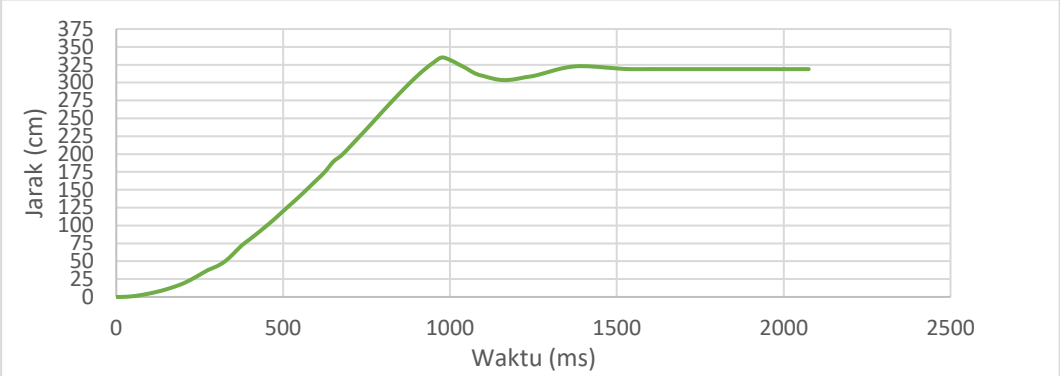
$$= 1{,}33\ \%$$

$$overshoot\ (\%) = \frac{295{,}05 - 282{,}84}{282{,}84} \times 100\ \%$$

$$= 4{,}31\ \%$$

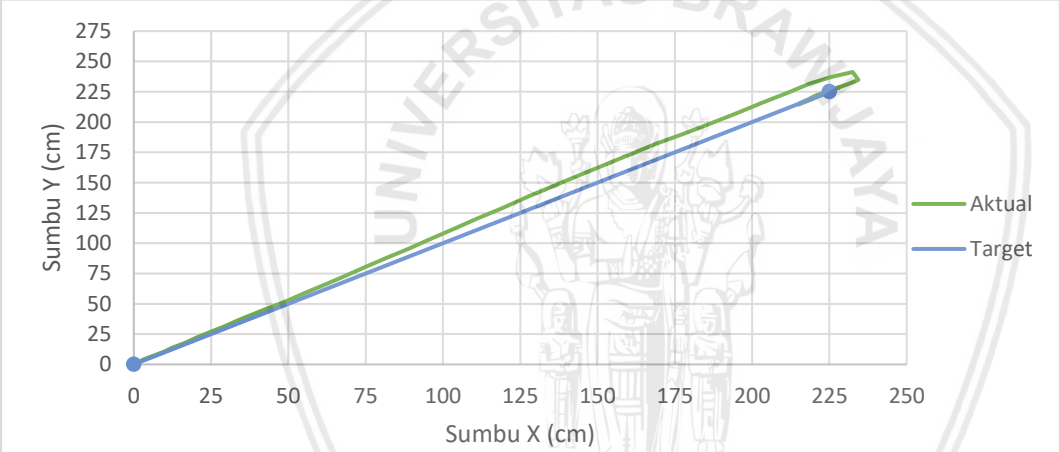
4.20 Pengujian dengan *Setpoint* Koordinat X,Y 225,225 cm

Hasil pengujian dengan *setpoint* koordinat XY225cm dapat dilihat dalam Gambar 4.39.



Gambar 4.39 Respon Sistem dengan *Setpoint* XY225cm

Lintasan robot dapat dilihat dalam Gambar 4.40.



Gambar 4.40 Lintasan Robot dengan *Setpoint* XY225cm

Berdasarkan respon sistem dalam Gambar 4.39 diperoleh nilai *settling time* sebesar 1,55 s, *overshoot* sebesar 5,31 %, dan *error steady state* 0,21 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|318{,}87 - 318{,}20|}{318{,}20} \times 100\ \%$$

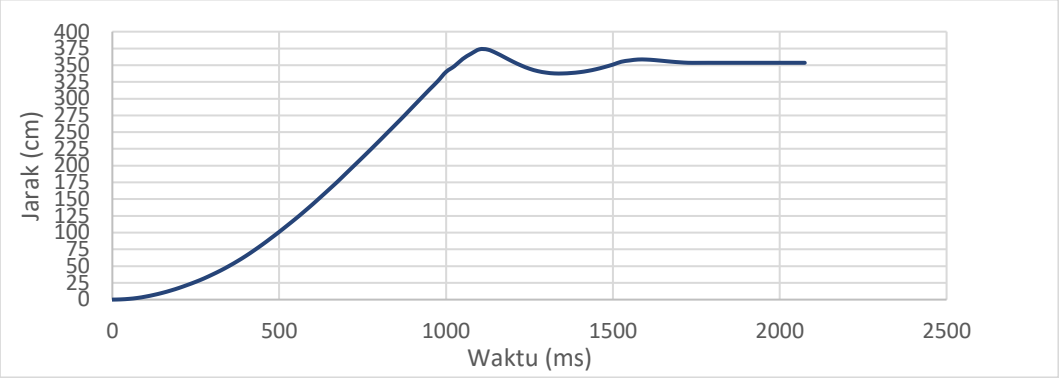
$$= 0{,}21\ \%$$

$$overshoot\ (\%) = \frac{335{,}10 - 318{,}20}{318{,}20} \times 100\ \%$$

$$= 5{,}31\ \%$$

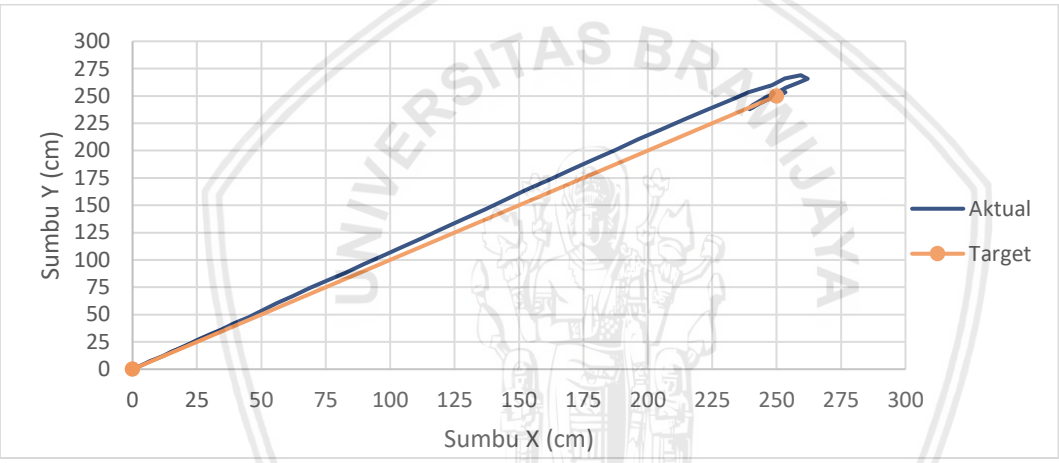
4.21 Pengujian dengan *Setpoint* Koordinat X,Y 250,250 cm

Hasil pengujian dengan *setpoint* koordinat XY250cm dapat dilihat dalam Gambar 4.41.



Gambar 4.41 Respon Sistem dengan *Setpoint* XY250cm

Lintasan robot dapat dilihat dalam Gambar 4.42.



Gambar 4.42 Lintasan Robot dengan *Setpoint* XY250cm

Berdasarkan respon sistem dalam Gambar 4.41 diperoleh nilai *settling time* sebesar 1,65 s, *overshoot* sebesar 5,68 %, dan *error steady state* 0,001 % dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|353{,}56 - 353{,}55|}{353{,}55} \times 100\ \%$$

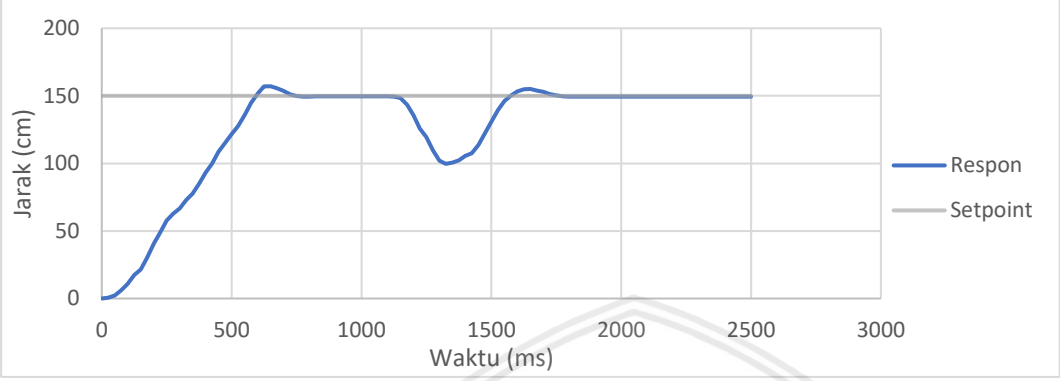
$$= 0{,}001\ \%$$

$$overshoot\ (\%) = \frac{373{,}65 - 353{,}55}{353{,}55} \times 100\ \%$$

$$= 5{,}68\ \%$$

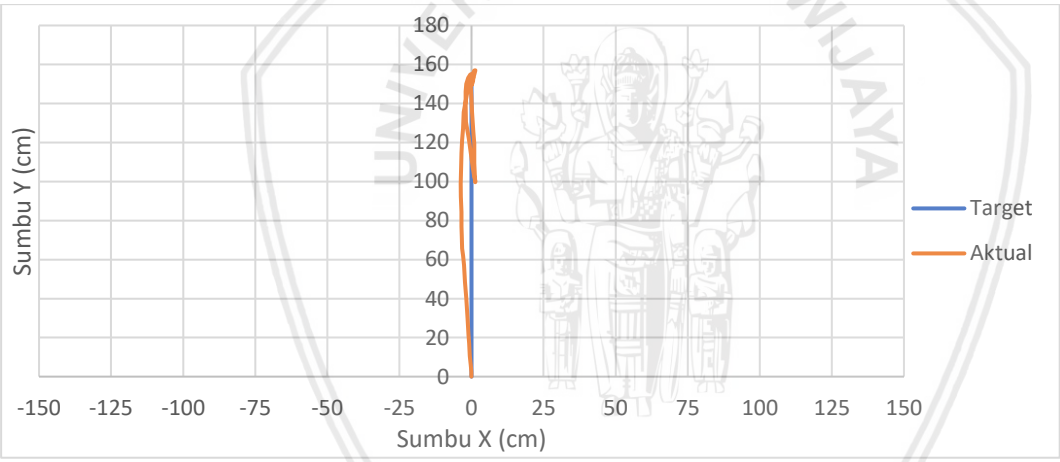
4.22 Pengujian Pada *Setpoint* Y 150 cm dengan Gangguan

Pada pengujian ini, robot diberikan gangguan berupa pemindahan posisi secara paksa setelah robot mencapai target. Hasil pengujian pada *setpoint* koordinat Y150cm dengan gangguan dapat dilihat dalam Gambar 4.43.



Gambar 4.43 Respon Sistem dengan Gangguan

Lintasan robot dapat dilihat dalam Gambar 4.44.



Gambar 4.44 Lintasan Robot dengan Gangguan

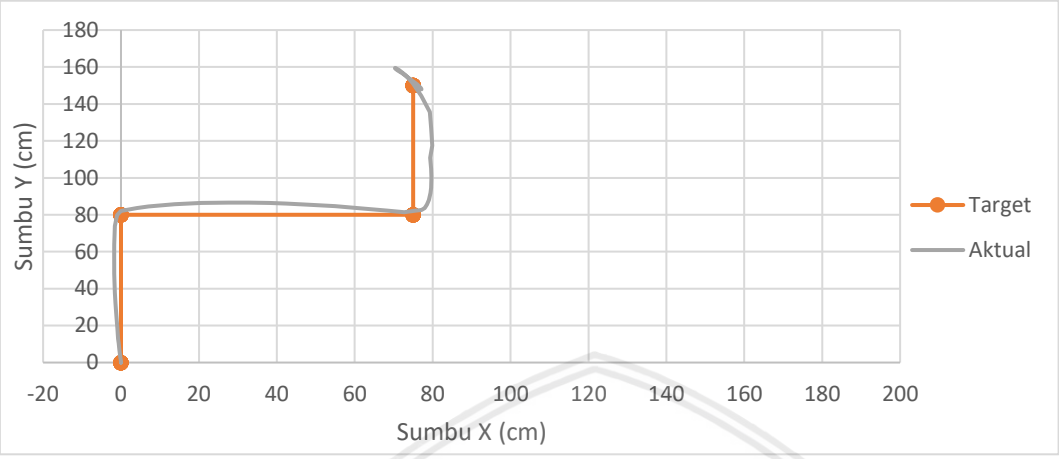
Berdasarkan respon sistem dalam Gambar 4.43 diperoleh nilai *settling time* sebesar 0,75 s, *overshoot* sebesar 4,72 %, *error steady state* 0,33 %, dan *recovery time* 0,5 s dengan perhitungan sebagai berikut:

$$e_{ss}\ (\%) = \frac{|149{,}51 - 150|}{150} \times 100\ \%$$
$$= 0{,}33\ \%$$

$$overshoot\ (\%) = \frac{157{,}08 - 150}{150} \times 100\ \%$$
$$= 4{,}72\ \%$$

4.23 Pengujian dengan *Multi Setpoint* Koordinat

Lintasan robot pada pengujian dengan *multi setpoint* koordinat Y80cm, X75Y80cm, X75Y150cm dapat dilihat dalam Gambar 4.45.

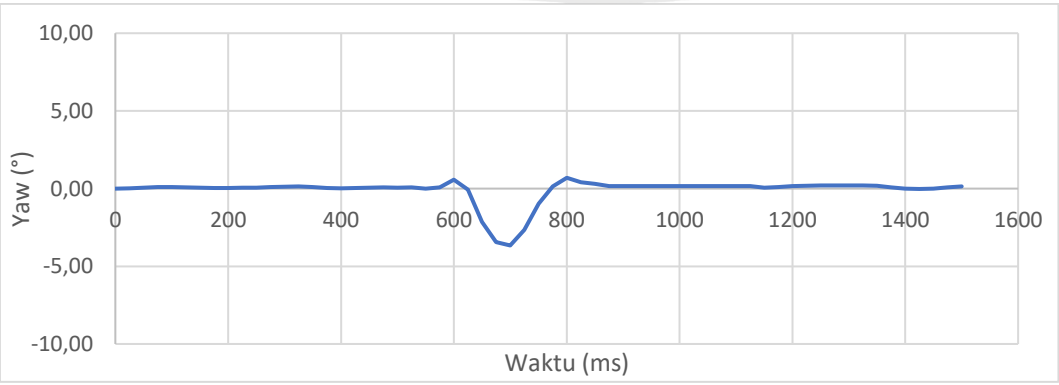


Gambar 4.45 Lintasan Robot dengan *Multi Setpoint*

Dalam Gambar 4.45 diketahui bahwa robot dapat berjalan dengan beberapa *setpoint* yang berbeda secara berkelanjutan. Robot berjalan menuju target koordinat pertama lalu robot langsung melanjutkan perjalanan menuju target selanjutnya sampai seluruh koordinat target yang telah ditentukan tercapai.

4.24 Pengujian Kontroler Orientasi dengan Gangguan

Pada pengujian ini dilakukan untuk mengetahui respon kontroler orientasi robot saat ada gangguan. Gangguan yang dimaksud dalam pengujian ini adalah saat robot berjalan menuju target koordinat terjadi *slip* terhadap lantai yang mengakibatkan orientasi robot berubah dan kontroler orientasi akan menyesuaikan perubahan tersebut supaya orientasi robot dapat kembali sesuai dengan *setpoint* seperti dalam Gambar 4.46.



Gambar 4.46 Respon Kontroler Orientasi dengan Gangguan

Berdasarkan respon sistem dalam Gambar 4.46 diperoleh nilai *overshoot* sebesar 0,7°, *error steady state* 0,14°, dan *recovery time* 0,3s.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisa yang telah dilakukan pada penelitian ini didapat beberapa kesimpulan sebagai berikut:

1. Hasil penerapan metode Euclidean pada sistem dengan menggunakan data odometri dan *gyroscope* dapat menghasilkan *path planning* dengan jarak terpendek serta tercepat.
2. Berdasarkan data respon sistem kontrol posisi robot dengan menggunakan metode kedua teori Ziegler-Nichols, diperoleh parameter kontroler PI dengan nilai $Kp = 6{,}75$ dan $Ki = 0{,}166$.
3. Berdasarkan data respon sistem kontrol orientasi robot dengan menggunakan metode kedua teori Ziegler-Nichols, diperoleh parameter kontroler PI dengan nilai $Kp = 3{,}15$ dan $Ki = 0{,}125$.
4. Berdasarkan data respon sistem pergerakan *mobile robot*, sistem dapat bergerak dari titik awal menuju *set-point* beberapa koordinat kartesian yang berbeda-beda memiliki rata-rata nilai *error steady state* sebesar 0,45%, *settling time* dibawah 2 s, dan rata-rata nilai *overshoot* sebesar 4,48%.
5. Hasil pengujian menunjukkan bahwa sistem dapat melakukan *recovery* lintasan saat terjadi gangguan dengan *recovery time* selama 0,5 s pada *setpoint* Y150cm.

5.2 Saran

Berikut ini adalah beberapa hal yang mungkin dapat dijadikan pertimbangan untuk penelitian selanjutnya:

1. Agar data perubahan posisi lebih presisi perlu dilakukan perancangan posisi peletakan *rotary encoder* yang lebih baik.
2. Untuk meningkatkan akurasi pergerakan dapat ditambahkan kontrol kecepatan tiap motor supaya keluaran putaran tiap motor dapat dijaga sesuai dengan nilai hasil perhitungan *inverse-kinematics*.

DAFTAR PUSTAKA

Ogata, K. (2010). *Modern Control Engineering*. New Jersey: Prentice-Hall, Inc.

Rusli, M. (2015). *Dasar Teknik Kontrol*. Malang: UB Press.

Yudaningtyas, E. (2017). *Belajar Sistem Kontrol: Soal dan Pembahasan*. Malang: UB Press.

Modern, M. P. (n.d.). *Praktikum Sistem Kontrol Modern*. Malang: Laboratorium Sistem Kontrol Teknik Elektro Universitas Brawijaya.

Bolton. W. (2004). *Sistem Instrumentasi Dan Sistem Kontrol*. Jakarta: Penerbit Erlangga.

Astrom, K. J., & Wittenmark, B. (2008). *Adaptive Control (Second ed.)*. Mineola: Dover Publication Inc.

Dikti. (2019). *Panduan Kontes Robot ABU Indonesia 2019*. Jakarta: Dikti.

ABU Robocon. (2019). *ABU Asia-Pasific Robot Contest 2019 Mongolia.* Mongolia: Host Organizing Comittee.

Oliveira, Helder P.; dkk. (2009). *Modelling and Assessing of Omni-directional Robot with Three and Four Wheels*. Contemporary Robotics – Challengers and Solutions, No. 12, pp. 207-229.

Ardilla, F.; dkk. (2011). *Path Tracking Pada Mobile Robot Dengan Umpan Balik Odometry*. The 13th Industrial Electronics Seminar.

Rizaldi, R.; dkk. (2018). *Implementasi Metode Euclidean Distance Untuk Rekomendasi Ukuran Pakaian Pada Aplikasi Ruang Ganti Virtual*. Jurnal Teknologi Informasi dan Ilmu Komputer, No.2, pp. 129-138.

LAMPIRAN

LAMPIRAN 1

LISTING PROGRAM

```
/**********************************
*********************
Project: Robo.MR1 KRAI-2019
Version: MR1-V3.1
Date: 11/06/2019
Author: Danu Pranandaru
Company: Teknik Elektro UB 2015
Comments: SEMANGAATTT!!! KRAI
UB JAYAA!!!
Chip type: STM32F407VGT6 (F4 Mini)
Program type: Main Program
**********************************
*********************/

#include "bismillah.h"

void vTaskRUN1(void *pvParameters) {
for (;;) {
/* General Purpose Counter */
count1++;
count2++;

/* Bluetooth Status (Check Data Stick) */
if(!btstate && !run.mode){
if(test_mode==1)
Inverse_Kinematics(kec,sdt,ww);
else if(!test_mode){
Inverse_Kinematics(0,0,0);
}}
else{
/* Stick Data Capture */
stick_data();

/* Convert Analog Data to Velocity &
Degree Input */
anatospeed();

/* Preparations & Input Data */
Preparations();

/* Main Odometrical Movement
Calculation */
odometrical_movement();

/* Heading Control */
PID_Yaw();
PID_Odometry();

/* Move to the Next Coordinate Target */
odometry_track();

/* Autonomous Movement */
if(run.mode){
if(hold==2)
        Inverse_Kinematics(aksel_v,sudut
_left,odo_yaw.MV+sudut_w);
else
        Inverse_Kinematics(run.spd_tgt,r
un.sdt_tgt,odo_yaw.MV+sudut_w);
}

/* Stick Controled Movement */
else{
if(!mode_imu)
        Inverse_Kinematics(aksel_v,sudut
_left,sudut_w);
else
        Inverse_Kinematics(aksel_v,sudut
_left,odo_yaw.MV+sudut_w);
}}}}

void vTaskRUN2(void *pvParameters) {
for (;;) {
/**********************************
************************/
/* Motor Set & Control */
V1.set = V1.SP;
V2.set = V2.SP;
V3.set = V3.SP;
V4.set = V4.SP;
```

```
eksekusi_motor(mtr1, V1.arah, V1.set);
eksekusi_motor(mtr2, V2.arah, V2.set);
eksekusi_motor(mtr3, V3.arah, V3.set);
eksekusi_motor(mtr4, V4.arah, V4.set);

/**********************************
************************/

TIM4->CCR1 = (( V1.set *
(TIM_4.period - 1)) / 100); //motor1
TIM4->CCR2 = (( V2.set *
(TIM_4.period - 1)) / 100); //motor2
TIM4->CCR3 = (( V3.set *
(TIM_4.period - 1)) / 100); //motor3
TIM4->CCR4 = (( V4.set *
(TIM_4.period - 1)) / 100); //motor4
}}

void vTaskRUN3( void * pvParameters
){
TickType_t xLastWakeTime;
const TickType_t xPeriod =
pdMS_TO_TICKS( 20 );
xLastWakeTime =
xTaskGetTickCount();

for(;;){
vTaskDelayUntil( &xLastWakeTime,
xPeriod );

if(tone[0]<=40){
tone[0]++;
TOGGLE_BUZZER
TOGGLE_RED
}
else{
tone_notif[2]=1;
}}}

void vTaskRUN4( void * pvParameters
){
TickType_t xLastWakeTime;
const TickType_t xPeriod =
pdMS_TO_TICKS( 5 );
xLastWakeTime =
xTaskGetTickCount();

for(;;){
vTaskDelayUntil( &xLastWakeTime,
xPeriod );
}}

void vTaskRUN5( void * pvParameters
){
TickType_t xLastWakeTime;
const TickType_t xPeriod =
pdMS_TO_TICKS( 75 );
xLastWakeTime =
xTaskGetTickCount();

for(;;){
vTaskDelayUntil( &xLastWakeTime,
xPeriod );

if(tone_notif[0] && tone_notif[1]){
BUZZER_OFF
}
else if(!tone_notif[0] && tone_notif[1]){
tone[1]++;
if(tone[1]<=8)
TOGGLE_BUZZER
else if(tone[1]<=12){
tone[1]=0;
BUZZER_OFF
}}
else if(tone_notif[0] && !tone_notif[1]){
tone[1]++;
if(tone[1]<=4)
TOGGLE_BUZZER
else if(tone[1]<=8){
tone[1]=0;
BUZZER_OFF
}}
else if(!tone_notif[0] &&
!tone_notif[1]){
TOGGLE_BUZZER
}}}

int main(void){
SystemInit();
inisTIM2();
inisTIM3();
inisTIM4();
inisTIM12();
External_Interrupt();
init_IO(RCC_AHB1Periph_GPIOD,
P_0|P_1|P_2|P_3|P_4|P_7,
GPIO_Mode_IN, GPIO_OType_PP,
GPIO_PuPd_UP ); //Encoder-
```

```
init_IO(RCC_AHB1Periph_GPIOB,
P_3|P_4, GPIO_Mode_IN,
GPIO_OType_PP, GPIO_PuPd_UP );
//Encoder-
init_IO(RCC_AHB1Periph_GPIOA,
P_12, GPIO_Mode_IN,
GPIO_OType_PP, GPIO_PuPd_UP );
//Encoder-

init_IO(RCC_AHB1Periph_GPIOE,
P_14|P_15, GPIO_Mode_OUT,
GPIO_OType_PP,
GPIO_PuPd_NOPULL ); //DIR Motor
init_IO(RCC_AHB1Periph_GPIOB,
P_10|P_11|P_12|P_13|P_14|P_15,
GPIO_Mode_OUT, GPIO_OType_PP,
GPIO_PuPd_NOPULL ); //DIR Motor
init_IO(RCC_AHB1Periph_GPIOD,
P_8|P_9|P_10|P_11, GPIO_Mode_OUT,
GPIO_OType_PP,
GPIO_PuPd_NOPULL ); //DIR Motor
init_IO(RCC_AHB1Periph_GPIOB,
P_2|P_5|P_6|P_7, GPIO_Mode_OUT,
GPIO_OType_PP,
GPIO_PuPd_NOPULL ); //Buzzer, LED
Indikator (G,B,Y)
init_IO(RCC_AHB1Periph_GPIOC,
P_12, GPIO_Mode_OUT,
GPIO_OType_PP,
GPIO_PuPd_NOPULL); //LED Indikator
(Red)
init_IO(RCC_AHB1Periph_GPIOE, P_0,
GPIO_Mode_OUT, GPIO_OType_PP,
GPIO_PuPd_NOPULL); //LED Indikator
Board (Red)

init_IO(RCC_AHB1Periph_GPIOB,
P_0|P_1|P_8|P_9, GPIO_Mode_IN,
GPIO_OType_PP, GPIO_PuPd_UP );
//Button
init_IO(RCC_AHB1Periph_GPIOE,
P_1|P_6|P_7|P_8|P_9|P_10|P_12,
GPIO_Mode_IN, GPIO_OType_PP,
GPIO_PuPd_UP ); //Mode; PX

init_USART(RCC_APB1Periph_USART
2, 38400, RCC_AHB1Periph_GPIOD,
P_6, RX);       //Stick

init_USART(RCC_APB1Periph_USART
3, 115200, RCC_AHB1Periph_GPIOC,
P_11, RX);      //IMU

xTaskCreate( vTaskRUN1, ( signed char
* ) "RUN1",
configMINIMAL_STACK_SIZE,
NULL, 1, NULL);
xTaskCreate( vTaskRUN2, ( signed char
* ) "RUN2",
configMINIMAL_STACK_SIZE,
NULL, 1, NULL);
xTaskCreate( vTaskRUN3, ( signed char
* ) "RUN3",
configMINIMAL_STACK_SIZE,
NULL, 1, NULL);
xTaskCreate( vTaskRUN4, ( signed char
* ) "RUN4",
configMINIMAL_STACK_SIZE,
NULL, 1, NULL);
xTaskCreate( vTaskRUN5, ( signed char
* ) "RUN5",
configMINIMAL_STACK_SIZE,
NULL, 1, NULL);

vTaskStartScheduler();
while (1)
{}}

void
TIM8_BRK_TIM12_IRQHandler(){
if (TIM_GetITStatus(TIM12,
TIM_IT_CC1)){
TIM_ClearITPendingBit(TIM12,
TIM_IT_CC1);
capture = TIM_GetCapture1(TIM12);
TIM_SetCompare1(TIM12, capture +
CCR1_Val);

}
else if (TIM_GetITStatus(TIM12,
TIM_IT_CC2)){
TIM_ClearITPendingBit(TIM12,
TIM_IT_CC2);
capture = TIM_GetCapture2(TIM12);
TIM_SetCompare2(TIM12, capture +
CCR2_Val);

btwait++;
if(btwait>50){
```

```
btstate=btcheck;
btcheck=0;
btwait=0;
}}
else if (TIM_GetITStatus(TIM12,
TIM_IT_CC3)){
TIM_ClearITPendingBit(TIM12,
TIM_IT_CC3);
capture = TIM_GetCapture3(TIM12);
TIM_SetCompare3(TIM12, capture +
CCR3_Val);
}
else
{
TIM_ClearITPendingBit(TIM12,
TIM_IT_CC4);
capture = TIM_GetCapture4(TIM12);
TIM_SetCompare4(TIM12, capture +
CCR4_Val);
}}

void EXTI9_5_IRQHandler(){
if (EXTI_GetITStatus(EXTI_Line5)){
//Max stable read 500RPM
speed[5]++;

rt_ext1.ensud++;
if(rt_ext1.ensud>rt_ext1.ppr)
rt_ext1.ensud=1;
else if(rt_ext1.ensud<0)
        rt_ext1.ensud=rt_ext1.ppr-1;

rt_ext1.sudut=(rt_ext1.ensud*360)/rt_ext
1.ppr;
if(!rt_ext1.CaptureNumber)
{
rt_ext1.EXTI_ReadValue1 = TIM2-
>CNT;
rt_ext1.CaptureNumber = 1;
}
else if(rt_ext1.CaptureNumber)
{
rt_ext1.EXTI_ReadValue2 = TIM2-
>CNT;
// Hitung Data Periode
if (rt_ext1.EXTI_ReadValue2 >
rt_ext1.EXTI_ReadValue1)rt_ext1.Captu
re = (rt_ext1.EXTI_ReadValue2 -
rt_ext1.EXTI_ReadValue1);
else if (rt_ext1.EXTI_ReadValue2 <
rt_ext1.EXTI_ReadValue1)rt_ext1.Captu
re = ((TIM2 -> ARR -
rt_ext1.EXTI_ReadValue1) +
rt_ext1.EXTI_ReadValue2);
else rt_ext1.Capture = 0;
// Hitung Frekuensi
rt_ext1.Freq = TIM_2.period /
rt_ext1.Capture;
rt_ext1.Rpm =
(rt_ext1.Freq*60)/rt_ext1.ppr;
rt_ext1.CaptureNumber = 0;
}

if(rt_ext1.Rpm<maxrpm){
if(!GPIO_ReadInputDataBit(GPIOD,
GPIO_Pin_7)){
rt_ext1.last_dir = inc;

rt_ext1.encoder++;
theta1=((yaw-
run.setyaw)+sdt_ext1)*pi/180.0;

coX.real += sinf(theta1);
coY.real += cosf(theta1);
}
else{
rt_ext1.last_dir = dec;

rt_ext1.encoder--;
theta1=((yaw-
run.setyaw)+sdt_ext1)*pi/180.0;

coX.real -= sinf(theta1);
coY.real -= cosf(theta1);
}}
else if(rt_ext1.last_dir == inc){
rt_ext1.encoder++;
theta1=((yaw-
run.setyaw)+sdt_ext1)*pi/180.0;

coX.real += sinf(theta1);
coY.real += cosf(theta1);
}
else if(rt_ext1.last_dir == dec){
rt_ext1.encoder--;
theta1=((yaw-
run.setyaw)+sdt_ext1)*pi/180.0;

coX.real -= sinf(theta1);
```

```
coY.real -= cosf(theta1);
}

EXTI_ClearITPendingBit(EXTI_Line5);
}

if (EXTI_GetITStatus(EXTI_Line6)){
//Max stable read 500RPM
speed[6]++;

rt_ext2.ensud++;
if(rt_ext2.ensud>rt_ext2.ppr)
rt_ext2.ensud=1;
else if(rt_ext2.ensud<0)
        rt_ext2.ensud=rt_ext2.ppr-1;

rt_ext2.sudut=(rt_ext2.ensud*360)/rt_ext
2.ppr;
if(!rt_ext2.CaptureNumber)
{
rt_ext2.EXTI_ReadValue1 = TIM2-
>CNT;
rt_ext2.CaptureNumber = 1;
}
else if(rt_ext2.CaptureNumber)
{
rt_ext2.EXTI_ReadValue2 = TIM2-
>CNT;
// Hitung Data Periode
if (rt_ext2.EXTI_ReadValue2 >
rt_ext2.EXTI_ReadValue1)rt_ext2.Captu
re = (rt_ext2.EXTI_ReadValue2 -
rt_ext2.EXTI_ReadValue1);
else if (rt_ext2.EXTI_ReadValue2 <
rt_ext2.EXTI_ReadValue1)rt_ext2.Captu
re = ((TIM2 -> ARR -
rt_ext2.EXTI_ReadValue1) +
rt_ext2.EXTI_ReadValue2);
else rt_ext2.Capture = 0;
// Hitung Frekuensi
rt_ext2.Freq = TIM_2.period /
rt_ext2.Capture;
rt_ext2.Rpm =
(rt_ext2.Freq*60)/rt_ext2.ppr;
rt_ext2.CaptureNumber = 0;
}

if(rt_ext2.Rpm<maxrpm){
if(!GPIO_ReadInputDataBit(GPIOB,
GPIO_Pin_3)){
rt_ext2.last_dir = inc;

rt_ext2.encoder++;
theta2=((yaw-
run.setyaw)+sdt_ext2)*pi/180.0;

coX.real += sinf(theta2);
coY.real += cosf(theta2);
}
else{
rt_ext2.last_dir = dec;

rt_ext2.encoder--;
theta2=((yaw-
run.setyaw)+sdt_ext2)*pi/180.0;

coX.real -= sinf(theta2);
coY.real -= cosf(theta2);
}}
else if(rt_ext2.last_dir == inc){
rt_ext2.encoder++;
theta2=((yaw-
run.setyaw)+sdt_ext2)*pi/180.0;

coX.real += sinf(theta2);
coY.real += cosf(theta2);
}
else if(rt_ext2.last_dir == dec){
rt_ext2.encoder--;
theta2=((yaw-
run.setyaw)+sdt_ext2)*pi/180.0;

coX.real -= sinf(theta2);
coY.real -= cosf(theta2);
}
EXTI_ClearITPendingBit(EXTI_Line6);
}

if (EXTI_GetITStatus(EXTI_Line7)) {
//Max stable read 500RPM
speed[7]++;

rt_ext3.ensud++;
if(rt_ext3.ensud>rt_ext3.ppr)
rt_ext3.ensud=1;
else if(rt_ext3.ensud<0)
        rt_ext3.ensud=rt_ext3.ppr-1;

rt_ext3.sudut=(rt_ext3.ensud*360)/rt_ext
3.ppr;
```

```
if(!rt_ext3.CaptureNumber)
{
rt_ext3.EXTI_ReadValue1 = TIM2-
>CNT;
rt_ext3.CaptureNumber = 1;
}
else if(rt_ext3.CaptureNumber)
{
rt_ext3.EXTI_ReadValue2 = TIM2-
>CNT;
// Hitung Data Periode
if (rt_ext3.EXTI_ReadValue2 >
rt_ext3.EXTI_ReadValue1)rt_ext3.Captu
re = (rt_ext3.EXTI_ReadValue2 -
rt_ext3.EXTI_ReadValue1);
else if (rt_ext3.EXTI_ReadValue2 <
rt_ext3.EXTI_ReadValue1)rt_ext3.Captu
re = ((TIM2 -> ARR -
rt_ext3.EXTI_ReadValue1) +
rt_ext3.EXTI_ReadValue2);
else rt_ext3.Capture = 0;
// Hitung Frekuensi
rt_ext3.Freq = TIM_2.period /
rt_ext3.Capture;
rt_ext3.Rpm =
(rt_ext3.Freq*60)/rt_ext3.ppr;
rt_ext3.CaptureNumber = 0;
}

if(rt_ext3.Rpm<maxrpm){
if(!GPIO_ReadInputDataBit(GPIOB,
GPIO_Pin_4)){
rt_ext3.last_dir = inc;

rt_ext3.encoder++;
theta3=((yaw-
run.setyaw)+sdt_ext3)*pi/180.0;

coX.real += sinf(theta3);
coY.real += cosf(theta3);
}
else{
rt_ext3.last_dir = dec;

rt_ext3.encoder--;
theta3=((yaw-
run.setyaw)+sdt_ext3)*pi/180.0;

coX.real -= sinf(theta3);
coY.real -= cosf(theta3);
}}
else if(rt_ext3.last_dir == inc){
rt_ext3.encoder++;
theta3=((yaw-
run.setyaw)+sdt_ext3)*pi/180.0;

coX.real += sinf(theta3);
coY.real += cosf(theta3);
}
else if(rt_ext3.last_dir == dec){
rt_ext3.encoder--;
theta3=((yaw-
run.setyaw)+sdt_ext3)*pi/180.0;

coX.real -= sinf(theta3);
coY.real -= cosf(theta3);
}
EXTI_ClearITPendingBit(EXTI_Line7);
}}

//Usart Interrupt section
void USART3_IRQHandler(void)
{
rxt=1;
if( USART_GetITStatus(USART3,
USART_IT_RXNE) ){
tone_notif[0]=1;

switch(bitimu){
case 0: rximu[0] = USART3->DR;
if(rximu[0] == 10)
bitimu = 1;
else bitimu = 0; break;
case 1: rximu[1] = USART3->DR;
if(rximu[1] == 21)
bitimu = 2;
else bitimu = 0; break;
case 2: rximu[bitimu] = USART3->DR;
bitimu++; break;
case 3: rximu[bitimu] = USART3->DR;
bitimu++; break;
case 4: rximu[bitimu] = USART3->DR;
bitimu++; break;
case 5: rximu[bitimu] = USART3->DR;
break;
}

if(bitimu==5){
yaw = rximu[2] + rximu[3] + (float)
rximu[4]/100;
```

```
if(yaw>180) yaw = yaw - 360;
bitimu = 0;
}

if(!yaw) GREEN_OFF
else      GREEN_ON

rxt=2;
}
else{
rxt=3;
tone_notif[0]=0;
}}

void USART2_IRQHandler(void)
{
if( USART_GetITStatus(USART2,
USART_IT_RXNE) ){
tone_notif[1]=1;
TOGGLE_REDOB

char bit=USART2->DR; // the character
from the USART2 data register is saved
in bitt
char static data_terakhir;

switch(bit_ke)
{
case 0 : rxdata[bit_ke]=USART2->DR;
bit_ke++; break;
case 1 : rxdata[bit_ke]=USART2->DR;
bit_ke++; break;
case 2 : rxdata[bit_ke]=USART2->DR;
bit_ke++; break;
case 3 : rxdata[bit_ke]=USART2->DR;
bit_ke++; break;
case 4 : rxdata[bit_ke]=USART2->DR;
bit_ke++; break;
case 5 : rxdata[bit_ke]=USART2->DR;
bit_ke++; break;
case 6 : rxdata[bit_ke]=USART2->DR;
bit_ke++; break;
case 7 : rxdata[bit_ke]=USART2->DR;
bit_ke++; break;
case 8 : rxdata[bit_ke]=USART2->DR;
bit_ke=100; break;
}
if(bit==0b10101010&&data_terakhir==0
b01010101){
bit_ke=0;
REDOB_ON
}
else REDOB_OFF
data_terakhir=bit;

if((rxdata[2]==255 && rxdata[3]==255
&& rxdata[4]==255 && rxdata[5]==255)
|| (rxdata[3]==128 && rxdata[5]==128))
btcheck=0;
else btcheck=1;
}
else{
tone_notif[1]=0;
}}

#include "bismillah.h"

//                1 2 3 4 5 6 7 8 9
Rotary rt_ext1={0,0,0,0,0,0,0,0,2000};
Rotary rt_ext2={0,0,0,0,0,0,0,0,2000};
Rotary rt_ext3={0,0,0,0,0,0,0,0,2000};

//Odom
Odom
run={0,0,0,0,0,0,0,0,0,0,200,0,{0,0,0},{0,
0,0}};

//P, I, D, MV,error,
last1_error,last2_error, temp, set, Kp, Ki,
Kd, SP , MAX, MIN, arah
TIMER TIM_3  = {
1000,              999,
83};
TIMER TIM_4  = {
1000,              999,
83};

PID odo_spd
={0,0,0,0,0,0,0,0,0,6.75,0.166,0,0,40,0,0}
;
PID odo_yaw
={0,0,0,0,0,0,0,0,0,3.1499,0.1,0,0,50,-
50,0};


//
        Frequency (Hz)
Period          Prescaler
```

```
TIMER TIM_1  = {
2000,              999,
83};
TIMER TIM_2  = {
        1,                 999999,
          83};
TIMER TIM_12 = {
1282,            50000,
        0};

//Timer Section
void inisTIM2()
{
RCC_APB1PeriphClockCmd(RCC_APB
1Periph_TIM2, ENABLE);

TIM_TimeBaseStructure.TIM_Period =
TIM_2.period;
TIM_TimeBaseStructure.TIM_Prescaler
= TIM_2.prescaler;
TIM_TimeBaseStructure.TIM_ClockDivi
sion = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_CounterM
ode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM2,
&TIM_TimeBaseStructure);

TIM_Cmd(TIM2, ENABLE);
}

void inisTIM12(){

RCC_APB1PeriphClockCmd(RCC_APB
1Periph_TIM12, ENABLE);

/* Time base configuration */
TIM_TimeBaseStructure.TIM_Period =
TIM_12.period;
TIM_TimeBaseStructure.TIM_Prescaler
= TIM_12.prescaler;
TIM_TimeBaseStructure.TIM_ClockDivi
sion = 0;
TIM_TimeBaseStructure.TIM_CounterM
ode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM12,
&TIM_TimeBaseStructure);

/* Output Compare Timing Mode
configuration: Channel1 */
TIM_OCInitStructure.TIM_OCMode =
TIM_OCMode_Timing;
TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse =
CCR1_Val;
TIM_OCInitStructure.TIM_OCPolarity =
TIM_OCPolarity_High;
TIM_OC1Init(TIM12,
&TIM_OCInitStructure);

TIM_OC1PreloadConfig(TIM12,
TIM_OCPreload_Disable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse =
CCR2_Val;
TIM_OC2Init(TIM12,
&TIM_OCInitStructure);

TIM_OC2PreloadConfig(TIM12,
TIM_OCPreload_Disable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse =
CCR3_Val;
TIM_OC3Init(TIM12,
&TIM_OCInitStructure);

TIM_OC3PreloadConfig(TIM12,
TIM_OCPreload_Disable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse =
CCR4_Val;
TIM_OC4Init(TIM12,
&TIM_OCInitStructure);

TIM_OC4PreloadConfig(TIM12,
TIM_OCPreload_Disable);

/* TIM Interrupts enable */
TIM_ITConfig(TIM12, TIM_IT_CC1 |
TIM_IT_CC2 | TIM_IT_CC3,
ENABLE);
TIM_ITConfig(TIM12, TIM_IT_CC4,
DISABLE);
```

```
/* TIM12 enable counter */
TIM_Cmd(TIM12, ENABLE);

NVIC_InitStructure.NVIC_IRQChannel
= TIM8_BRK_TIM12_IRQn;
NVIC_InitStructure.NVIC_IRQChannelP
reemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelS
ubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannel
Cmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
}

//PWM & Movement Section
void inisTIM4(){
GPIO_InitTypeDef GPIO_InitStructure;
TIM_TimeBaseInitTypeDef
TIM_TimeBaseStructure;
TIM_OCInitTypeDef
TIM_OCInitStructure;
TIM_BDTRInitTypeDef
TIM_BDTRInitStructure;

RCC_APB1PeriphClockCmd(RCC_APB
1Periph_TIM4, ENABLE);
RCC_AHB1PeriphClockCmd(RCC_AH
B1Periph_GPIOD, ENABLE);

GPIO_InitStructure.GPIO_Pin =
GPIO_Pin_12|GPIO_Pin_13|GPIO_Pin_
14|GPIO_Pin_15;
GPIO_InitStructure.GPIO_Mode =
GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Speed =
GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_OType =
GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd =
GPIO_PuPd_NOPULL;
GPIO_Init(GPIOD,
&GPIO_InitStructure);

GPIO_PinAFConfig(GPIOD,
GPIO_PinSource12, GPIO_AF_TIM4);
GPIO_PinAFConfig(GPIOD,
GPIO_PinSource13, GPIO_AF_TIM4);
GPIO_PinAFConfig(GPIOD,
GPIO_PinSource14, GPIO_AF_TIM4);
GPIO_PinAFConfig(GPIOD,
GPIO_PinSource15, GPIO_AF_TIM4);

TIM_TimeBaseStructure.TIM_Prescaler
= TIM_4.prescaler;
TIM_TimeBaseStructure.TIM_CounterM
ode = TIM_CounterMode_Up;
TIM_TimeBaseStructure.TIM_Period =
TIM_4.period;
TIM_TimeBaseStructure.TIM_ClockDivi
sion = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_Repetition
Counter = 0;
TIM_TimeBaseInit(TIM4,
&TIM_TimeBaseStructure);

TIM_OCInitStructure.TIM_OCMode =
TIM_OCMode_PWM2;
TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_OutputNStat
e = TIM_OutputNState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;
TIM_OCInitStructure.TIM_OCPolarity =
TIM_OCPolarity_Low;
TIM_OCInitStructure.TIM_OCNPolarity
= TIM_OCNPolarity_Low;
TIM_OCInitStructure.TIM_OCIdleState
= TIM_OCIdleState_Set;
TIM_OCInitStructure.TIM_OCNIdleStat
e = TIM_OCIdleState_Reset;
TIM_OC1Init(TIM4,
&TIM_OCInitStructure);

TIM_BDTRInitStructure.TIM_OSSRStat
e = TIM_OSSRState_Enable;
TIM_BDTRInitStructure.TIM_OSSIState
= TIM_OSSIState_Enable;
TIM_BDTRInitStructure.TIM_LOCKLe
vel = TIM_LOCKLevel_1;
TIM_BDTRInitStructure.TIM_DeadTim
e = 11;
TIM_BDTRInitStructure.TIM_Break =
TIM_Break_Enable;
TIM_BDTRInitStructure.TIM_BreakPola
rity = TIM_BreakPolarity_High;
TIM_BDTRInitStructure.TIM_Automati
cOutput =
TIM_AutomaticOutput_Enable;
```

```
TIM_BDTRConfig(TIM4,
&TIM_BDTRInitStructure);

TIM_OC1Init(TIM4,
&TIM_OCInitStructure);
TIM_OC1PreloadConfig(TIM4,
TIM_OCPreload_Enable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;

TIM_OC2Init(TIM4,
&TIM_OCInitStructure);
TIM_OC2PreloadConfig(TIM4,
TIM_OCPreload_Enable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;

TIM_OC3Init(TIM4,
&TIM_OCInitStructure);
TIM_OC3PreloadConfig(TIM4,
TIM_OCPreload_Enable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;

TIM_OC4Init(TIM4,
&TIM_OCInitStructure);
TIM_OC4PreloadConfig(TIM4,
TIM_OCPreload_Enable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;

TIM_ARRPreloadConfig(TIM4,
ENABLE);

TIM_Cmd(TIM4, ENABLE);

TIM_CtrlPWMOutputs(TIM4,
ENABLE);
}

void inisTIM3(){
GPIO_InitTypeDef GPIO_InitStructure;
TIM_TimeBaseInitTypeDef
TIM_TimeBaseStructure;
TIM_OCInitTypeDef
TIM_OCInitStructure;
TIM_BDTRInitTypeDef
TIM_BDTRInitStructure;

RCC_APB1PeriphClockCmd(RCC_APB
1Periph_TIM3, ENABLE);
RCC_AHB1PeriphClockCmd(RCC_AH
B1Periph_GPIOC, ENABLE);

GPIO_InitStructure.GPIO_Pin =
GPIO_Pin_6|GPIO_Pin_7;
GPIO_InitStructure.GPIO_Mode =
GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Speed =
GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_OType =
GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd =
GPIO_PuPd_NOPULL;
GPIO_Init(GPIOC,
&GPIO_InitStructure);

GPIO_PinAFConfig(GPIOC,
GPIO_PinSource6, GPIO_AF_TIM3);
GPIO_PinAFConfig(GPIOC,
GPIO_PinSource7, GPIO_AF_TIM3);

TIM_TimeBaseStructure.TIM_Prescaler
= TIM_3.prescaler;
TIM_TimeBaseStructure.TIM_CounterM
ode = TIM_CounterMode_Up;
TIM_TimeBaseStructure.TIM_Period =
TIM_3.period;
TIM_TimeBaseStructure.TIM_ClockDivi
sion = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_Repetition
Counter = 0;
TIM_TimeBaseInit(TIM3,
&TIM_TimeBaseStructure);

TIM_OCInitStructure.TIM_OCMode =
TIM_OCMode_PWM2;
TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_OutputNStat
e = TIM_OutputNState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;
```

```
TIM_OCInitStructure.TIM_OCPolarity =
TIM_OCPolarity_Low;
TIM_OCInitStructure.TIM_OCNPolarity
= TIM_OCNPolarity_Low;
TIM_OCInitStructure.TIM_OCIdleState
= TIM_OCIdleState_Set;
TIM_OCInitStructure.TIM_OCNIdleStat
e = TIM_OCIdleState_Reset;
TIM_OC1Init(TIM3,
&TIM_OCInitStructure);

TIM_BDTRInitStructure.TIM_OSSRStat
e = TIM_OSSRState_Enable;
TIM_BDTRInitStructure.TIM_OSSIState
= TIM_OSSIState_Enable;
TIM_BDTRInitStructure.TIM_LOCKLe
vel = TIM_LOCKLevel_1;
TIM_BDTRInitStructure.TIM_DeadTim
e = 11;
TIM_BDTRInitStructure.TIM_Break =
TIM_Break_Enable;
TIM_BDTRInitStructure.TIM_BreakPola
rity = TIM_BreakPolarity_High;
TIM_BDTRInitStructure.TIM_Automati
cOutput =
TIM_AutomaticOutput_Enable;
TIM_BDTRConfig(TIM3,
&TIM_BDTRInitStructure);

TIM_OC1Init(TIM3,
&TIM_OCInitStructure);
TIM_OC1PreloadConfig(TIM3,
TIM_OCPreload_Enable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;

TIM_OC2Init(TIM3,
&TIM_OCInitStructure);
TIM_OC2PreloadConfig(TIM3,
TIM_OCPreload_Enable);

TIM_OCInitStructure.TIM_OutputState
= TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0;

TIM_ARRPreloadConfig(TIM3,
ENABLE);
TIM_Cmd(TIM3, ENABLE);

TIM_CtrlPWMOutputs(TIM3,
ENABLE);
}

void eksekusi_motor(uint16_t mtr,
uint8_t dir, uint16_t pwm){

switch(mtr){
case mtr1:{     switch(dir){
case maju       :{
        Pin(GPIOE,ON,P_14);
Pin(GPIOE,OFF,P_15);
V1.set=pwm;
} break;
case mundur    :{
        Pin(GPIOE,OFF,P_14);
Pin(GPIOE,ON,P_15);
V1.set=pwm;
} break;
case stop       :{
        Pin(GPIOE,ON,P_14);
Pin(GPIOE,ON,P_15);
V1.set=pwm;} break;}
} break;
case mtr2:{     switch(dir){
case maju       :{
        Pin(GPIOB,ON,P_10);
Pin(GPIOB,OFF,P_11);
V2.set=pwm;
} break;
case mundur    :{
        Pin(GPIOB,OFF,P_10);
Pin(GPIOB,ON,P_11);
V2.set=pwm;
} break;
case stop       :{
        Pin(GPIOB,ON,P_10);
Pin(GPIOB,ON,P_11);
V2.set=pwm;} break;}
} break;
case mtr3:{     switch(dir){
case maju       :{
        Pin(GPIOB,ON,P_12);
Pin(GPIOB,OFF,P_13);
V3.set=pwm;
} break;
case mundur    :{
        Pin(GPIOB,OFF,P_12);
```

```
Pin(GPIOB,ON,P_13);
V3.set=pwm;
} break;
case stop       :{
        Pin(GPIOB,ON,P_12);
Pin(GPIOB,ON,P_13);
V3.set=pwm;} break;}
} break;
case mtr4:{     switch(dir){
case maju       :{
        Pin(GPIOB,ON,P_14);
Pin(GPIOB,OFF,P_15);
V4.set=pwm;} break;
case mundur     :{
        Pin(GPIOB,OFF,P_14);
Pin(GPIOB,ON,P_15);
V4.set=pwm;} break;
case stop       :{
        Pin(GPIOB,ON,P_14);
Pin(GPIOB,ON,P_15);
V4.set=pwm;} break;}
} break;
case mtr5:{     switch(dir){
case maju       :{
        Pin(GPIOD,ON,P_8);
Pin(GPIOD,OFF,P_9);
S1.set=pwm;} break;
case mundur     :{
        Pin(GPIOD,OFF,P_8);
Pin(GPIOD,ON,P_9);
S1.set=pwm;} break;
case stop       :{
        Pin(GPIOD,ON,P_8);
Pin(GPIOD,ON,P_9);
S1.set=pwm;} break;}
} break;
case mtr6:{     switch(dir){
case maju       :{
        Pin(GPIOD,ON,P_10);
Pin(GPIOD,OFF,P_11);
V6.set=pwm;} break;
case mundur     :{
        Pin(GPIOD,OFF,P_10);
Pin(GPIOD,ON,P_11);
V6.set=pwm;} break;
case stop       :{
        Pin(GPIOD,ON,P_10);
Pin(GPIOD,ON,P_11);
V6.set=pwm;} break;}
} break;}}
```

```
void Inverse_Kinematics(int speed, float
sudut, int W){
//Set Point RPM
sudut = (sudut/180)*pi;

V4.SP = (int)(speed*(1)*sin(sudut -
(pi/4))+W);
V1.SP = (int)(speed*(1)*sin(sudut -
(pi/4)*3)+W);
V2.SP = (int)(speed*(1)*sin(sudut -
(pi/4)*5)+W);
V3.SP = (int)(speed*(1)*sin(sudut -
(pi/4)*7)+W);

if (V1.SP<0)   {        V1.SP = -V1.SP;
        V1.arah = mundur;     }
else if(V1.SP>0){       V1.SP = V1.SP;
                V1.arah = maju;         }
else {  V1.SP = 0;
        V1.arah = stop;                 }

if (V2.SP<0)   {        V2.SP = -V2.SP;
        V2.arah = mundur;     }
else if(V2.SP>0){       V2.SP = V2.SP;
                V2.arah = maju;         }
else{   V2.SP = 0;
        V2.arah = stop;                 }

if (V3.SP<0)   {        V3.SP = -V3.SP;
        V3.arah = mundur;     }
else if(V3.SP>0){       V3.SP = V3.SP;
                V3.arah = maju;         }
else {  V3.SP = 0;
        V3.arah = stop;                 }

if (V4.SP<0)   {        V4.SP = -V4.SP;
        V4.arah = mundur;     }
else if(V4.SP>0){       V4.SP = V4.SP;
                V4.arah         = maju;}
else{   V4.SP = 0;
        V4.arah         = stop;         }

if(W==0){
if((V1.SP>=V2.SP)&&(V1.SP>=V3.SP)
&&(V1.SP>=V4.SP)){
V2.SP=(speed/V1.SP)*V2.SP;
V3.SP=(speed/V1.SP)*V3.SP;
V4.SP=(speed/V1.SP)*V4.SP;
V1.SP=(speed/V1.SP)*V1.SP;
```

```
}
else
if((V2.SP>=V1.SP)&&(V2.SP>=V3.SP)
&&(V2.SP>=V4.SP)){
V1.SP=(speed/V2.SP)*V1.SP;
V3.SP=(speed/V2.SP)*V3.SP;
V4.SP=(speed/V2.SP)*V4.SP;
V2.SP=(speed/V2.SP)*V2.SP;
}
else
if((V3.SP>=V1.SP)&&(V3.SP>=V2.SP)
&&(V3.SP>=V4.SP)){
V1.SP=(speed/V3.SP)*V1.SP;
V2.SP=(speed/V3.SP)*V2.SP;
V4.SP=(speed/V3.SP)*V4.SP;
V3.SP=(speed/V3.SP)*V3.SP;
}
else
if((V4.SP>=V1.SP)&&(V4.SP>=V2.SP)
&&(V4.SP>=V3.SP)){
V1.SP=(speed/V4.SP)*V1.SP;
V2.SP=(speed/V4.SP)*V2.SP;
V3.SP=(speed/V4.SP)*V3.SP;
V4.SP=(speed/V4.SP)*V4.SP;
}}
if(V1.SP<3)V1.SP=0;
if(V2.SP<3)V2.SP=0;
if(V3.SP<3)V3.SP=0;
if(V4.SP<3)V4.SP=0;
}

void odometrical_movement(){
//Convert real data to mm
coXmm = coX.real / 17.032;
coYmm = coY.real / 17.032;

coXcm = coX.real / 170.32;
coYcm = coY.real / 170.32;

coXbaca = coXcm - run.setx;
coYbaca = coYcm - run.sety;

//Jarak cm
run.dist = sqrt(pow((coXcm - run.setx),
2) + pow((coYcm - run.sety), 2));
if(run.dist<10 || tanda){ odo_spd.Kp =
6.75; tanda=1;}

//Target Sudut
run.sdt_tgt = atan2f((coXcm - run.setx),
(coYcm - run.sety));
run.sdt_tgt *= 57.295779;
run.sdt_tgt -= (yaw - run.setyaw);
if (run.sdt_tgt < 0) run.sdt_tgt = 360 +
run.sdt_tgt;
}

void coin(int _n, int _cox, int _coy){
run.tmpx[_n] = _cox;
run.tmpy[_n] = _coy;
}

void Preparations(){
if(!X){ //Reset & Emergency Button
//Reset Encoder Value
rt_ext1.encoder = 0;
rt_ext2.encoder = 0;
rt_ext3.encoder = 0;

coX.real = 0;
coY.real = 0;

//Reset Coordinate Target & Odometrical
Mode
run.setx = 0;
run.sety = 0;
track = 0;
point = 0;
run.dist = 0;
run.mode = 0;
run.sdt_tgt = 0;
run.spd_tgt = 0;
press = 0;
push = 0;
hold = 0;
aksel_v = 0;
mode_imu=0;
cnt_shagai = 0;
shagai = 0;
rt_shagai.encoder = 0;
rt_shagai.ensud = 0;
S1.SP = 0;
S1.status = 0;
pneumod = 0;
sudut_left = 0;

runyaw = 0;
run.setyaw = 0;
tone[0] = 30;
```

```
meter = 0;
}
else  if(!L2 && !run.mode){
runyaw = yaw;
run.setyaw = runyaw;
mode_imu=1;
tone[0] = 34;

// Skripsi
coin(0, 0,  0);
coin(1, 0, 80);
coin(2,75, 80);
coin(3,75,150);


else if(!R3 && !press && mode_imu &&
!run.mode){
run.mode=1;
press=1;
tone[0] = 36;
tanda = 0;
point=3;
}

if(!R2 && !up && !push){
point++;
push = 1;
}
else if(!R2 && !right && push){
push = 0;
}
else if(!R2 && !down && !push){
point--;
push = 1;
}

if(!start)                    mode_stick=0;
else if(!slect)   mode_stick=1;
}

void odometry_track(){

if(run.dist<1 && run.mode && press &&
!hold){

if(track<point){
hold=0;
track++;
run.setx=run.tmpx[track];
run.sety=run.tmpy[track];
}
else if(track>point){
hold=0;
track--;
run.setx=run.tmpx[track];
run.sety=run.tmpy[track];
}

}}

void PID_Odometry(){
//Speed PID
odo_spd.error = run.dist;
odo_spd.P = odo_spd.Kp *
odo_spd.error;
odo_spd.I = (odo_spd.Ki *
(odo_spd.error + odo_spd.last1_error))/2;
odo_spd.D = (odo_spd.Kd *
(odo_spd.error - 2*odo_spd.last1_error +
odo_spd.last2_error));
odo_spd.MV = (odo_spd.P + odo_spd.I +
odo_spd.D);

if(odo_spd.MV > odo_spd.MAX)
        odo_spd.MV = odo_spd.MAX;
if(odo_spd.MV < odo_spd.MIN)
        odo_spd.MV = odo_spd.MIN;

run.spd_tgt = odo_spd.MV;


odo_spd.last2_error =
odo_spd.last1_error;
odo_spd.last1_error = odo_spd.error;
}
void PID_Yaw(){
//Yaw PID
odo_yaw.error = yaw - runyaw;
if (odo_yaw.error > 180) odo_yaw.error
= odo_yaw.error - 360;
else if (odo_yaw.error < -180)
odo_yaw.error = odo_yaw.error + 360;

odo_yaw.P = odo_yaw.Kp *
odo_yaw.error;
odo_yaw.I = (odo_yaw.Ki *
(odo_yaw.error +
odo_yaw.last1_error))/2;
```

```
odo_yaw.D = (odo_yaw.Kd *
(odo_yaw.error - 2*odo_yaw.last1_error
+ odo_yaw.last2_error));
odo_yaw.MV = (odo_yaw.P + odo_yaw.I
+ odo_yaw.D);


if(odo_yaw.MV>odo_yaw.MAX)

        odo_yaw.MV=odo_yaw.MAX;
else if(odo_yaw.MV<odo_yaw.MIN)
        odo_yaw.MV=odo_yaw.MIN;

odo_yaw.last2_error =
odo_yaw.last1_error;
odo_yaw.last1_error = odo_yaw.error;
}

void stick_data(){
slect=(rxdata[0])&0x01;
L3=(rxdata[0])&0x02;
R3=(rxdata[0])&0x04;
start=(rxdata[0])&0x08;
up=(rxdata[0])&0x10;
right=(rxdata[0])&0x20;
down=(rxdata[0])&0x40;
left=(rxdata[0])&0x80;

L2=(rxdata[1])&0x01;
R2=(rxdata[1])&0x02;
L1=(rxdata[1])&0x04;
R1=(rxdata[1])&0x08;
A=(rxdata[1])&0x10;
O=(rxdata[1])&0x20;
X=(rxdata[1])&0x40;
Q=(rxdata[1])&0x80;
anarix=(rxdata[2])-128;
anariy=-(rxdata[3]-127);
analex=(rxdata[4])-128;
analey=-(rxdata[5]-127);
}

void anatospeed()
{
/* Left Stick */
//Default Depan 0 Belakang 180 Kanan
90 Kiri 270
sudut_left = atan2f(analex, analey);
sudut_left *= 57.295779;       //Rad to
Deg
if (sudut_left < 0) sudut_left = 360 +
sudut_left;

//Depan 180 Belakang 0 Kanan 270 Kiri
90
if(!mode_stick){
if(sudut_left<=180)
sudut_left += 180;
else
sudut_left -= 180;
}

if((abs(analex)==127 ||
abs(analex)==128) || (abs(analey)==127 ||
abs(analey)==128)){
sudut_left = sudut_left;

if(A){
if(aksel_v<min_std) aksel_v=min_std;
else
if(aksel_v>=aksel_std)aksel_v=aksel_std;
else{
if(count1>cnt_std){
aksel_v+=plus_std;
count1=0;
}}}
else{
if(aksel_v<min_spd) aksel_v=min_spd;
else
if(aksel_v>=aksel_spd)aksel_v=aksel_sp
d;
else{
if(count1>cnt_spd){
aksel_v+=plus_spd;
count1=0;
}}}
tmp_sudut_left = sudut_left;
}
else{
if(aksel_v>max_dec){
sudut_left = tmp_sudut_left;
if(count1>cnt_dec){
aksel_v-=minus_dec;
count1=0;
}}
else{
aksel_v=0;
}}
```

```
if(!L1 && R1 && L2 && R2 && A)
                        sudut_w=w_rot;
else if(!L1 && R1 && L2 && R2 &&
!A)     sudut_w=w_rot+25;
else if(!R1 && L1 && L2 && R2 &&
A)              sudut_w=-w_rot;
else if(!R1 && L1 && L2 && R2 &&
!A)     sudut_w=-w_rot+25;
else
                        sudut_w=0;

}

//Interrupt Section
void External_Interrupt(){
init_IO(RCC_AHB1Periph_GPIOA,
P_5|P_6|P_7|P_11, GPIO_Mode_IN,
GPIO_OType_PP, GPIO_PuPd_UP );

EXTI_InitStruct.EXTI_Line =
EXTI_Line5|EXTI_Line6|EXTI_Line7|E
XTI_Line11;
EXTI_InitStruct.EXTI_LineCmd =
ENABLE;
EXTI_InitStruct.EXTI_Mode =
EXTI_Mode_Interrupt;
EXTI_InitStruct.EXTI_Trigger =
EXTI_Trigger_Falling;
EXTI_Init(&EXTI_InitStruct);

SYSCFG_EXTILineConfig(EXTI_PortS
ourceGPIOA, EXTI_PinSource5);
NVIC_InitStruct.NVIC_IRQChannel =
EXTI9_5_IRQn;
NVIC_InitStruct.NVIC_IRQChannelPree
mptionPriority = 0x00;
NVIC_InitStruct.NVIC_IRQChannelSub
Priority = 0x01;
NVIC_InitStruct.NVIC_IRQChannelCm
d = ENABLE;
NVIC_Init(&NVIC_InitStruct);

SYSCFG_EXTILineConfig(EXTI_PortS
ourceGPIOA, EXTI_PinSource6);
NVIC_InitStruct.NVIC_IRQChannel =
EXTI9_5_IRQn;
NVIC_InitStruct.NVIC_IRQChannelPree
mptionPriority = 0x00;
NVIC_InitStruct.NVIC_IRQChannelSub
Priority = 0x02;
NVIC_InitStruct.NVIC_IRQChannelCm
d = ENABLE;
NVIC_Init(&NVIC_InitStruct);

SYSCFG_EXTILineConfig(EXTI_PortS
ourceGPIOA, EXTI_PinSource7);
NVIC_InitStruct.NVIC_IRQChannel =
EXTI9_5_IRQn;
NVIC_InitStruct.NVIC_IRQChannelPree
mptionPriority = 0x00;
NVIC_InitStruct.NVIC_IRQChannelSub
Priority = 0x03;
NVIC_InitStruct.NVIC_IRQChannelCm
d = ENABLE;
NVIC_Init(&NVIC_InitStruct);

SYSCFG_EXTILineConfig(EXTI_PortS
ourceGPIOA, EXTI_PinSource11);
NVIC_InitStruct.NVIC_IRQChannel =
EXTI15_10_IRQn;
NVIC_InitStruct.NVIC_IRQChannelPree
mptionPriority = 0x00;
NVIC_InitStruct.NVIC_IRQChannelSub
Priority = 0x04;
NVIC_InitStruct.NVIC_IRQChannelCm
d = ENABLE;
NVIC_Init(&NVIC_InitStruct);
}

#ifndef __bismillah_H
#define __bismillah_H

#include "stm32f4xx.h"
#include "stm32f4xx_tim.h"
#include "stm32f4xx_exti.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_i2c.h"
#include "stm32f4xx_syscfg.h"
#include "stm32f4xx_usart.h"
#include "misc.h"
#include "TEUB_USART.h"
#include "TEUB_GPIO.h"

//RealtimeOS
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"

#include <math.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>

//CCR Val
#define CCR1_Val 5000
#define CCR2_Val 50000
#define CCR3_Val 1000
#define CCR4_Val 65535

//Movement
#define mtr1            0x1
#define mtr2            0x2
#define mtr3            0x3
#define mtr4            0x4
#define mtr5            0x5
#define mtr6            0x6
#define maju            0x7
#define mundur          0x8
#define stop            0x9

#define inc             91
#define dec             90

#define kri     1
#define kna     2
#define blk     3

GPIO_InitTypeDef
        GPIO_InitStruct;
GPIO_InitTypeDef
        GPIO_InitStructure;
EXTI_InitTypeDef
        EXTI_InitStruct;
NVIC_InitTypeDef
        NVIC_InitStruct;
NVIC_InitTypeDef
        NVIC_InitStructure;
TIM_ICInitTypeDef
        TIM_ICInitStructure;
TIM_TimeBaseInitTypeDef
        TIM_TimeBaseStructure;
TIM_OCInitTypeDef
        TIM_OCInitStructure;

//Calculation var
uint32_t NilaiPrescaler;
#define pi                      3.14159265
#define aksel_spd       100
#define aksel_std       60
#define min_std         30
#define min_spd         50
#define cnt_std         10
#define cnt_spd         15
#define plus_std        1
#define plus_spd        3
#define max_dec         50
#define cnt_dec         3
#define minus_dec       7
#define w_rot           25

#define sdt_ext1        270
#define sdt_ext2        30
#define sdt_ext3        150

#define maxrpm          300

//Convert distance to pulse
#define mm      *17.032
#define cm      *170.32
#define m       *17032

#define baca
        GPIO_ReadInputDataBit
#define RED_ON
        Pin(GPIOC,ON,P_12);
#define RED_OFF
        Pin(GPIOC,OFF,P_12);
#define TOGGLE_RED
        GPIO_ToggleBits(GPIOC,GPIO_
Pin_12);
#define GREEN_ON
        Pin(GPIOB,ON,P_5);
#define GREEN_OFF
        Pin(GPIOB,OFF,P_5);
#define TOGGLE_GREEN
        GPIO_ToggleBits(GPIOB,GPIO_
Pin_5);
#define BLUE_ON
        Pin(GPIOB,ON,P_6);
#define BLUE_OFF
        Pin(GPIOB,OFF,P_6);
#define TOGGLE_BLUE
        GPIO_ToggleBits(GPIOB,GPIO_
Pin_6);
#define YELLOW_ON
        Pin(GPIOB,ON,P_7);
```

```
#define YELLOW_OFF
        Pin(GPIOB,OFF,P_7);
#define TOGGLE_YELLOW
        GPIO_ToggleBits(GPIOB,GPIO_
Pin_7);
#define REDOB_ON
        Pin(GPIOE,ON,P_0);
#define REDOB_OFF
        Pin(GPIOE,OFF,P_0);
#define TOGGLE_REDOB
        GPIO_ToggleBits(GPIOE,GPIO_
Pin_0);
#define BUZZER_ON
        Pin(GPIOB,ON,P_2);
#define BUZZER_OFF
        Pin(GPIOB,OFF,P_2);
#define TOGGLE_BUZZER
        GPIO_ToggleBits(GPIOB,GPIO_
Pin_2);

int
point,rximu[20],capture,bit_ke,sudut,prt[
9],btstate,btcheck,btwait,count1,count2,c
ount3,cnt_shagai,mode_odo,tone[2],tone
_notif[3];
unsigned char
rxdata[10],slect,L1,L2,L3,R1,R2,R3,start,
up,right,down,left,
X,O,A,Q,od[2];
long int speed[9];
float
er[4],sudut_left,tmp_sudut_left,sudut_rig
ht,sudut_smt,yaw,vX,vY,theta1,theta2,th
eta3,tmp[4],runyaw,t1,t2,t3,t4;
int
sudut_w,aksel_v,track,bitimu,anarix,anari
y,analex,analey;
int
Vx,Vy,press,push,pneumod,hold,mode_st
ick,kec,sdt,ww,test_mode,mode_imu,sha
gai,mode_lapangan,button[5],
err,mode_shagai,ls_shagai,ls[10],px_depa
n1,px_depan2,px_kiri,px_kanan,tf_y,tand
a,rxt,meter,start_data;
float coXmm, coYmm, coXcm, coYcm,
coXbaca, coYbaca;
float a[3],w[3],angle[3],T,sd1,sd2,sd3;

//Fungsi Global
void odometry_track();
void inisTIM2();        void inisTIM3();
        void inisTIM12();
void inisTIM4();
void External_Interrupt();
void eksekusi_motor(uint16_t mtr,
uint8_t dir, uint16_t pwm);
void Inverse_Kinematics(int speed, float
sudut, int W);
void odometrical_movement();        void
Preparations();
void stick_data(); void anatospeed();
void PID_Yaw();
void PID_Odometry();        void
coin(int _n, int _cox, int _coy);
void TIM3_IRQHandler(); void
EXTI9_5_IRQHandler();

//Struct section
typedef struct{
float ext1;
float ext2;
float ext3;
float real;
}Rot;
Rot coX; Rot coY;

typedef struct{
float sdt_tgt;
float sdt_now;
float sdt_prg;
int spd_tgt;
int setx;
int sety;
int prgx;
int prgy;
float dist;
float distprog;
int mode;
int spd;
int w;
int tmpx[25];
int tmpy[25];
float setyaw;
}Odom;
Odom run;

typedef struct{
float P;
float I;
float D;
```

```
float MV;
float error;
float last1_error;
float last2_error;
int temp; //variabel penyimpan speed
sementara
float set;
float Kp;
float Ki; //konstanta d tidak dipakai
float Kd; //konstanta i tidak dipakai
float SP; //setpoint kecepatan motor
int MAX;
int MIN;
int w;
unsigned char arah;
int status;
}PID;
PID odo_spd;  PID odo_yaw;

typedef struct{
uint32_t EXTI_ReadValue1;  //1 --> Baca
data pertama
uint32_t EXTI_ReadValue2;  //2 --> Baca
data kedua
uint8_t CaptureNumber;          //3 -
-> Penanda
uint32_t Capture;               //4 -
-> Data hasil kalkulasi
float Freq;
        //5 --> Frekuensi putaran
int Rpm;
        //6 --> Kecepatan putaran
int encoder;                    //7 -
-> Data encoder
int sudut;
        //8 --> Data sudut
int ppr;                                //9 -
-> Data pulse per rotation encoder
int last_dir;
int dirr;
int ensud;
}Rotary;
Rotary rt_shagai;
Rotary rt_ext1;
Rotary rt_ext2;
Rotary rt_ext3;

typedef struct{
uint16_t        frequency;      //1 -->
Frequency Hz (1s)
uint32_t        period;         //2 -->
MAKS COUNT REGISTER CNT
uint16_t        prescaler;      //3 -->
Prescaler Value
}
TIMER;
TIMER TIM_1;
TIMER TIM_2;
TIMER TIM_3;
TIMER TIM_4;
TIMER TIM_12;


#endif //__bismillah_H_
```

LAMPIRAN 2
DATASHEET

life.augmented

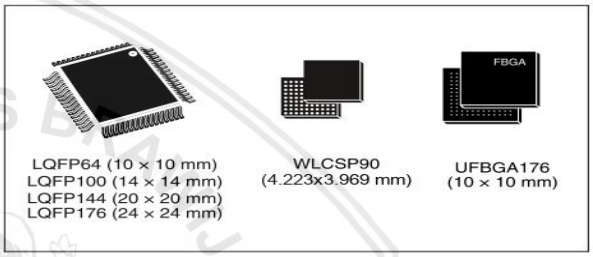
STM32F405xx
STM32F407xx

ARM Cortex-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera

Datasheet - production data

Features

- Core: ARM® 32-bit Cortex®-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator™) allowing 0-wait state execution from Flash memory, frequency up to 168 MHz, memory protection unit, 210 DMIPS/ 1.25 DMIPS/MHz (Dhrystone 2.1), and DSP instructions
- Memories
- Up to 1 Mbyte of Flash memory
- Up to 192+4 Kbytes of SRAM including 64-Kbyte of CCM (core coupled memory) data RAM
- Flexible static memory controller supporting Compact Flash, SRAM, PSRAM, NOR and NAND memories
- LCD parallel interface, 8080/6800 modes
- Clock, reset and supply management
  - 1.8 V to 3.6 V application supply and I/Os
  - POR, PDR, PVD and BOR
  - 4-to-26 MHz crystal oscillator
  - Internal 16 MHz factory-trimmed RC (1% accuracy)
  - 32 kHz oscillator for RTC with calibration
  - Internal 32 kHz RC with calibration
- Low-power operation
  - Sleep, Stop and Standby modes
  - V_{BAT} supply for RTC, 20×32 bit backup registers + optional 4 KB backup SRAM
- 3×12-bit, 2.4 MSPS A/D converters: up to 24 channels and 7.2 MSPS in triple interleaved mode
- 2×12-bit D/A converters
- General-purpose DMA: 16-stream DMA controller with FIFOs and burst support



- Up to 17 timers: up to twelve 16-bit and two 32-bit timers up to 168 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
- Debug mode
  - Serial wire debug (SWD) & JTAG interfaces
  - Cortex-M4 Embedded Trace Macrocell™
- Up to 140 I/O ports with interrupt capability
  - Up to 136 fast I/Os up to 84 MHz
  - Up to 138 5 V-tolerant I/Os
- Up to 15 communication interfaces
  - Up to 3 × I^2C interfaces (SMBus/PMBus)
  - Up to 4 USARTs/2 UARTs (10.5 Mbit/s, ISO 7816 interface, LIN, IrDA, modem control)
  - Up to 3 SPIs (42 Mbits/s), 2 with muxed full-duplex I^2S to achieve audio class accuracy via internal audio PLL or external clock
  - 2 × CAN interfaces (2.0B Active)
  - SDIO interface
- Advanced connectivity
  - USB 2.0 full-speed device/host/OTG controller with on-chip PHY
  - USB 2.0 high-speed/full-speed device/host/OTG controller with dedicated DMA, on-chip full-speed PHY and ULPI
  - 10/100 Ethernet MAC with dedicated DMA: supports IEEE 1588v2 hardware, MII/RMII

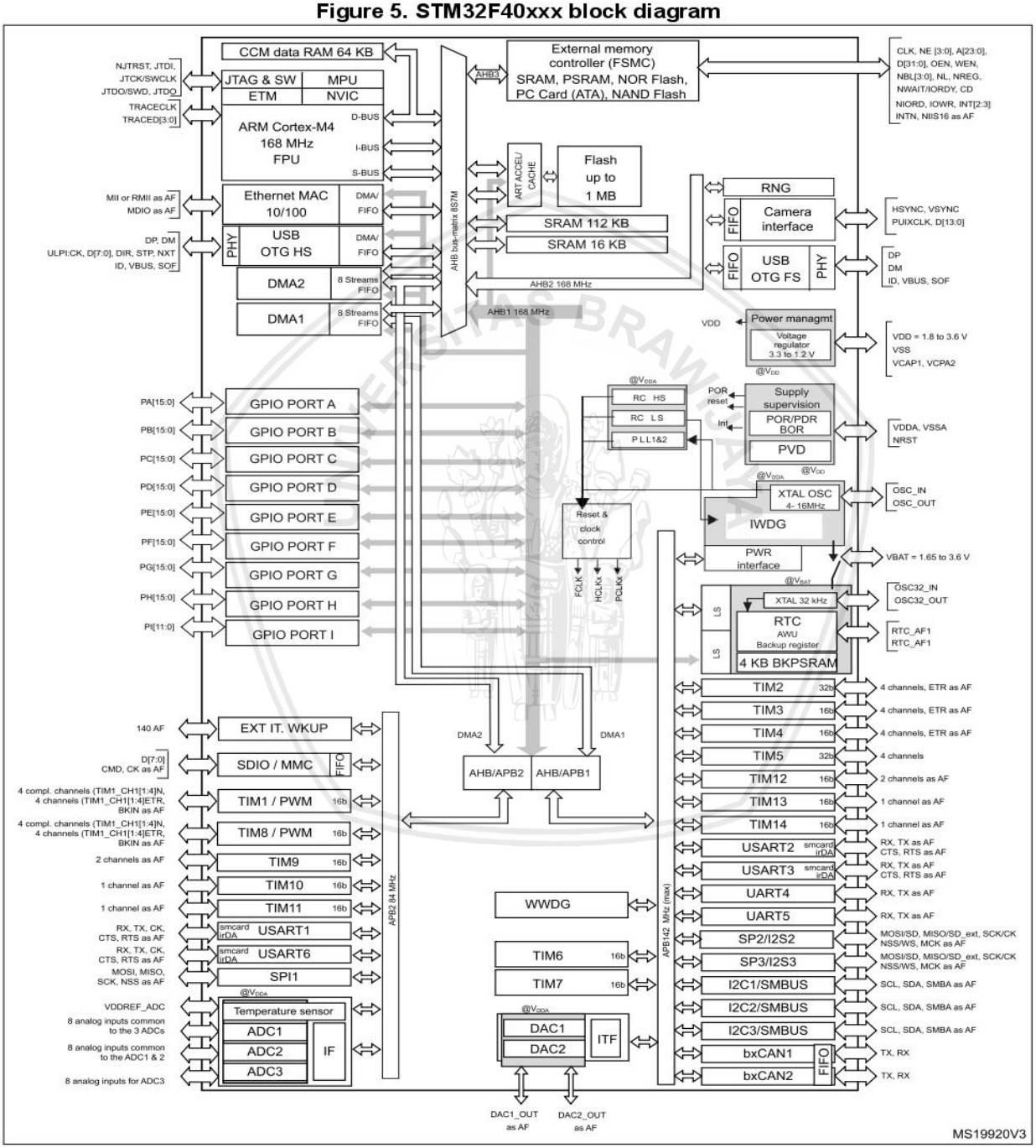
- 8- to 14-bit parallel camera interface up to 54 Mbytes/s
- True random number generator
- CRC calculation unit
- 96-bit unique ID
- RTC: subsecond accuracy, hardware calendar

Table 1. Device summary

Reference	Part number
STM32F405xx	STM32F405RG, STM32F405VG, STM32F405ZG, STM32F405OG, STM32F405OE
STM32F407xx	STM32F407VG, STM32F407IG, STM32F407ZG, STM32F407VE, STM32F407ZE, STM32F407IE

2.2 Device overview

Figure 5. STM32F40xxx block diagram



1. The camera interface and ethernet are available only on STM32F407xx devices.

www.brontoseno.com

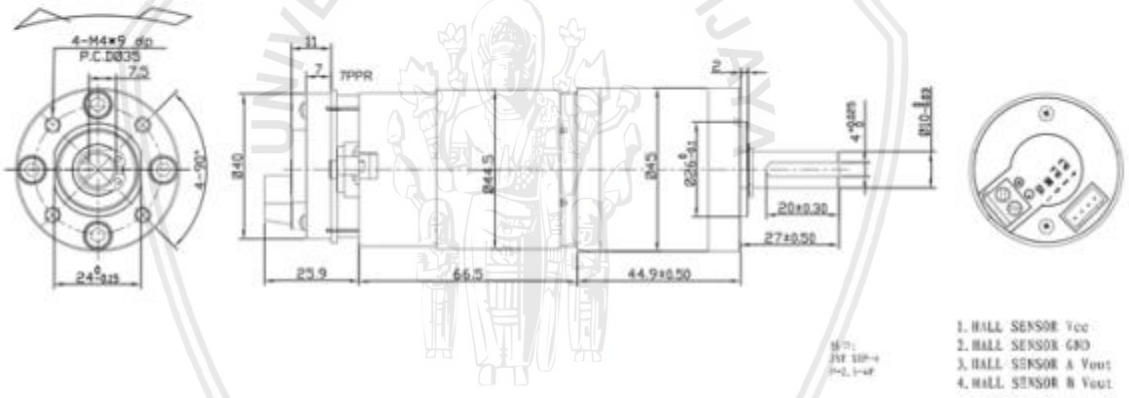
PG45 Series 775 Motor

Dc carbon brushed

Typical appli cations:

Robot , AGV , industrial actuator ,Pan/ Tilt cameras ,Grill,Oven,Cleaning machine, Garbage dispo sers,Packing bank note machine,Coffee machine,Medical mach ineManotat ,Amusement, equipment,infusion pumps,Office equipment,Ho usehold appliances ,Automatic actuator.

Drawing NO.	PG45A19.2K	Model No.	PG45	Voltage	24 DC
Drawing					:1.0



(Customer APP.):

Details	Geared motor specification			
	No Load Current (A)	≤1.500		
	No Load Speed (r.p.m)	468±10%	NO.	
	Rated Load Torque (kgf.cm)	15	1	
	Rated Current (A)	≤6.500	2	
	Rated Load Speed (r.p.m)	398±10%	3	
	Stall Current (A)		4	
	Stall Torque (kgf.cm)		5	
	Rotation Direction	CCW	6	

Draw		Rev		App.	

www.brontoseno.com

Data & specification Motor

Gearbox data	Data	Motor data	Data	Output after gearbox	Data
Number of stage	reduction	Motor name	Rs775	Motor name	PG45RS775
Reduction ratio	19.2	Rated torque	780 gfcm	Torque	15kgfcm
gearbok length	44.9	Voltage	24Vdc	No load speed	463Rpm
Max run in torque	60kgf.cm	No load current	1.5A	Rated load speed	398 (10+- %)
max gear breaking torque	120kg.fcm	Rated curent	6.5A	Stall torque	40kgfcm
Gearing efficincy	81%	Output power	70W	Rottation direction	CCW/CW

Test report

Test by : Andra risciawan

Motor name : PG45 19.2k 24V 7ppr encoder

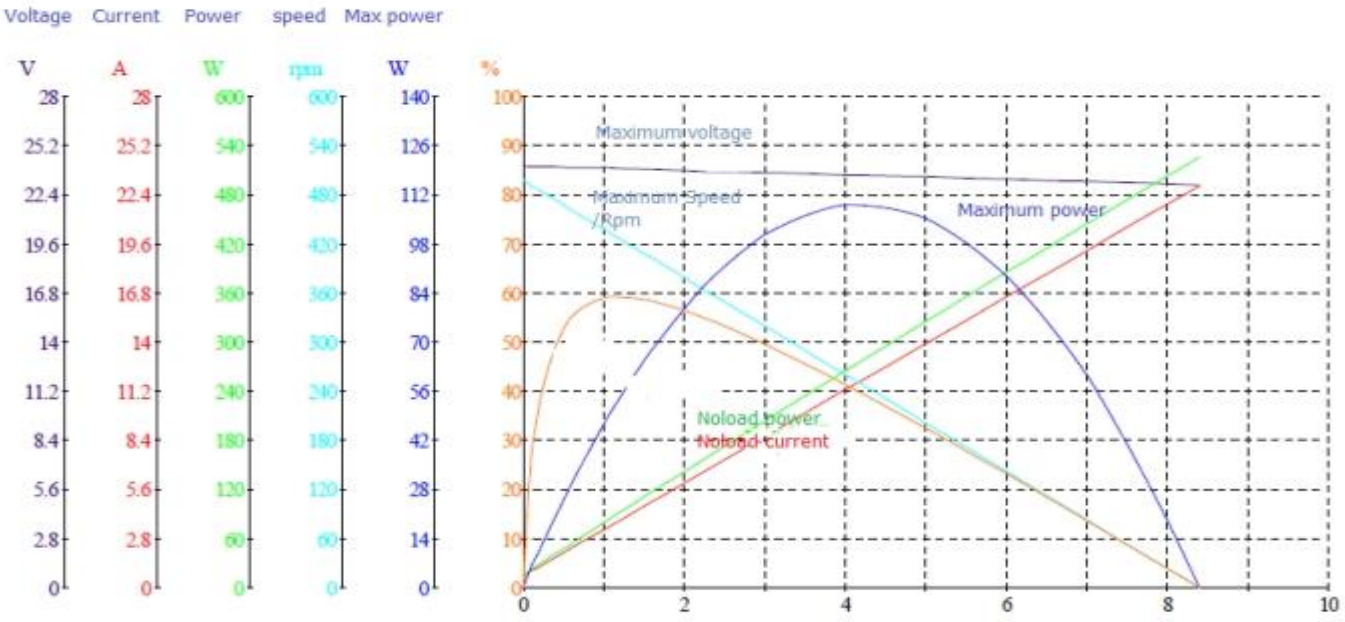
Voltage : 24vdc

Power rated : 100W

Test Date : 5 - 6 - 2 017

Motor type : brushed motor

Reduction : 1:19.2k



DATA	VOLT	Current	POWER	TORQUE	SPEED	MAX POWER	EFFICIENCY
	V	A	W	N.m	rpm	W	%
No_Load	24.02	0.610	14.65	0.000	497.2	0.000	0.0
Eff_max	23.86	3.907	93.21	1.244	423.6	55.17	59.2
Pout_max	23.47	12.10	283.9	4.335	240.6	109.2	38.5
Torque_max	22.96	22.87	525.1	8.400	0.0	0.000	0.0
End	22.96	22.87	525.1	8.400	0.0	0.000	0.0

No	VOLTAGE	I	POWER	Torque	Speed	Max power	EFFICIENCY
	V	A	W	N.m	rpm	W	%
1	24.02	0.610	14.65	0.000	497.2	0.000	0.0
2	24.01	0.769	18.46	0.060	493.7	3.101	16.8
3	24.01	0.806	19.35	0.074	492.8	3.818	19.7
4	24.00	0.917	22.02	0.116	490.4	5.955	27.0
5	24.00	1.002	24.05	0.148	488.5	7.569	31.5
6	24.00	1.013	24.30	0.152	488.2	7.770	32.0

7	23.99	1.148	27.54	0.203	485.2	10.31	37.4
8	23.98	1.439	34.51	0.313	478.7	15.69	45.5
9	23.96	1.763	42.24	0.435	471.5	21.47	50.8
10	23.94	2.123	50.84	0.571	463.4	27.70	54.5
11	23.93	2.457	58.79	0.697	456.0	33.27	56.6
12	23.91	2.746	65.67	0.806	449.5	37.93	57.8
13	23.88	3.427	81.84	1.063	434.3	48.34	59.1
14	23.86	3.907	93.21	1.244	423.6	55.17	59.2
15	23.84	4.405	105.0	1.432	412.5	61.84	58.9
16	23.82	4.802	114.4	1.582	403.6	66.85	58.4
17	23.79	5.390	128.2	1.804	390.4	73.75	57.5
18	23.76	5.968	141.8	2.022	377.5	79.92	56.4
19	23.73	6.551	155.5	2.242	364.5	85.56	55.0
20	23.71	7.044	167.0	2.428	353.5	89.86	53.8
21	23.68	7.643	181.0	2.654	340.1	94.51	52.2
22	23.64	8.534	201.7	2.990	320.2	100.3	49.7
23	23.60	9.425	222.4	3.326	300.3	104.6	47.0
24	23.55	10.32	243.0	3.663	280.4	107.5	44.3
25	23.51	11.21	263.5	3.999	260.5	109.1	41.4
26	23.47	12.10	283.9	4.335	240.6	109.2	38.5
27	23.43	12.99	304.3	4.671	220.7	108.0	35.5
28	23.39	13.88	324.6	5.007	200.8	105.3	32.4
29	23.34	14.77	344.8	5.343	180.9	101.2	29.4
30	23.30	15.66	364.9	5.680	161.0	95.76	26.2
31	23.26	16.55	385.0	6.016	141.1	88.90	23.1
32	23.22	17.44	404.9	6.352	121.2	80.63	19.9
33	23.17	18.33	424.8	6.688	101.3	70.96	16.7
34	23.13	19.22	444.7	7.024	81.4	59.90	13.5

35	23.09	20.11	464.4	7.360	61.5	47.43	10.2
36	23.05	21.01	484.1	7.697	41.6	33.56	6.9
37	23.00	21.90	503.7	8.033	21.7	18.29	3.6
38	22.96	22.79	523.2	8.369	1.8	1.618	0.3
39	22.96	22.87	525.1	8.400	0.0	0.000	0.0

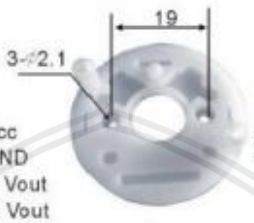
www.brontoseno.com

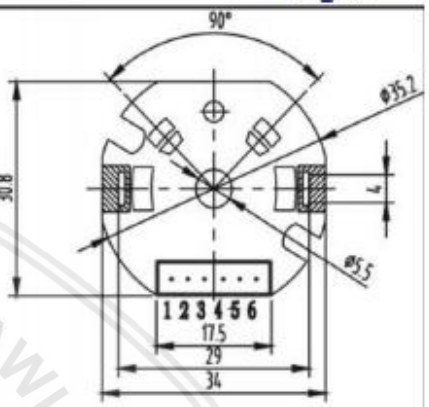
ENCODER MAGNETIC SERIES

JST XHP-6
P=2.5-6P

Two Channel Encoder
Connections :

1. MOTOR-
2. MOTOR+
3. HALL SENSOR Vcc
4. HALL SENSOR GND
5. HALL SENSOR A Vout
6. HALL SENSOR B Vout





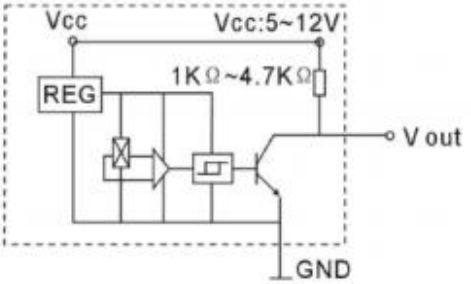
* The encoder mainly install on the Motor of RS-545、555、775
* Pulse encoder 7ppr

ELECTRICAL CHARACTERISTICS

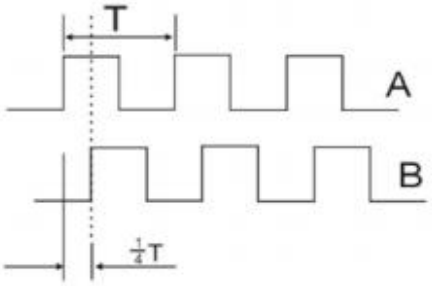
At V_{cc}=4.5V to 24V with 20mA load with Ta=-40℃ to 125℃ [-40°F to 257°F] unless otherwise noted.

CHARACTERISTICS	SYMBOL	TEST CONDITIONS	MIN	REF	MAX	UNITS
Supply Voltage	Vcc	---	4.5	--	24	V
Supply Current	Ice	---	--	14	20	mA
Output Current	Ic	Vce=12V; Gauss<-170	--	--	20	mA
Output Leakage Current	Icex	Output open;25℃[77°F]	--	--	10	μA
Output Rise Time	Tr	RL=820Ω; CL=20pF;25℃	--	0.5	1.5	μS
Output Fall Time	Tf	RL=820Ω; CL=20pF;25℃	--	0.2	1.5	μS

Output Circuit



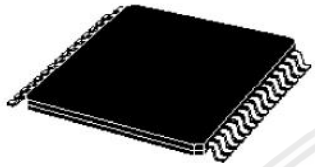
Output Wave



VNH2SP30-E

Automotive fully integrated H-bridge motor driver

Datasheet - production data



MultiPowerSO-30™

Features

Type	$R_{DS(on)}$	I_{out}	V_{ccmax}
VNH2SP30-E	19 mΩ max (per leg)	30 A	41 V

- AEC-Q100 qualified
- 5 V logic level compatible inputs
- Undervoltage and overvoltage shutdown
- Overvoltage clamp
- Thermal shutdown
- Cross-conduction protection
- Linear current limiter
- Very low standby power consumption
- PWM operation up to 20 kHz
- Protection against loss of ground and loss of V_{CC}
- Current sense output proportional to motor current
- Package: ECOPACK®

Description

The VNH2SP30-E is a full bridge motor driver intended for a wide range of automotive applications. The device incorporates a dual monolithic high side driver and two low side switches. The high side driver switch is designed using STMicroelectronics well known and proven proprietary VIPower™ M0 technology which permits efficient integration on the same die of a true power MOSFET with intelligent signal/protection circuitry.

The low side switches are vertical MOSFETs manufactured using STMicroelectronics proprietary EHD (STripFET™) process. The three die are assembled in a MultiPowerSO-30 package on electrically isolated leadframes. This package, specifically designed for the harsh automotive environments, offers improved thermal performance thanks to exposed die pads. Moreover, its fully symmetrical mechanical design allows superior manufacturability at board level. The input signals IN_A and IN_B can directly interface with the microcontroller to select the motor direction and brake condition. The $DIAG_A/EN_A$ or $DIAG_B/EN_B$, when connected to an external pull-up resistor, enable one leg of the bridge. They also provide a feedback digital diagnostic signal. The normal operating condition is explained in the truth table. The motor current can be monitored with the CS pin by delivering a current proportional to its value. The speed of the motor can be controlled in all possible conditions by the PWM up to 20 kHz. In all cases, a low level state on the PWM pin will turn off both the LS_A and LS_B switches. When PWM rises to a high level, LS_A or LS_B turn on again depending on the input pin state.

Table 1. Device summary

Package	Order code
	Tape and reel
MultiPowerSO-30	VNH2SP30TR-E

1 Block diagram and pin description

Figure 1. Block diagram

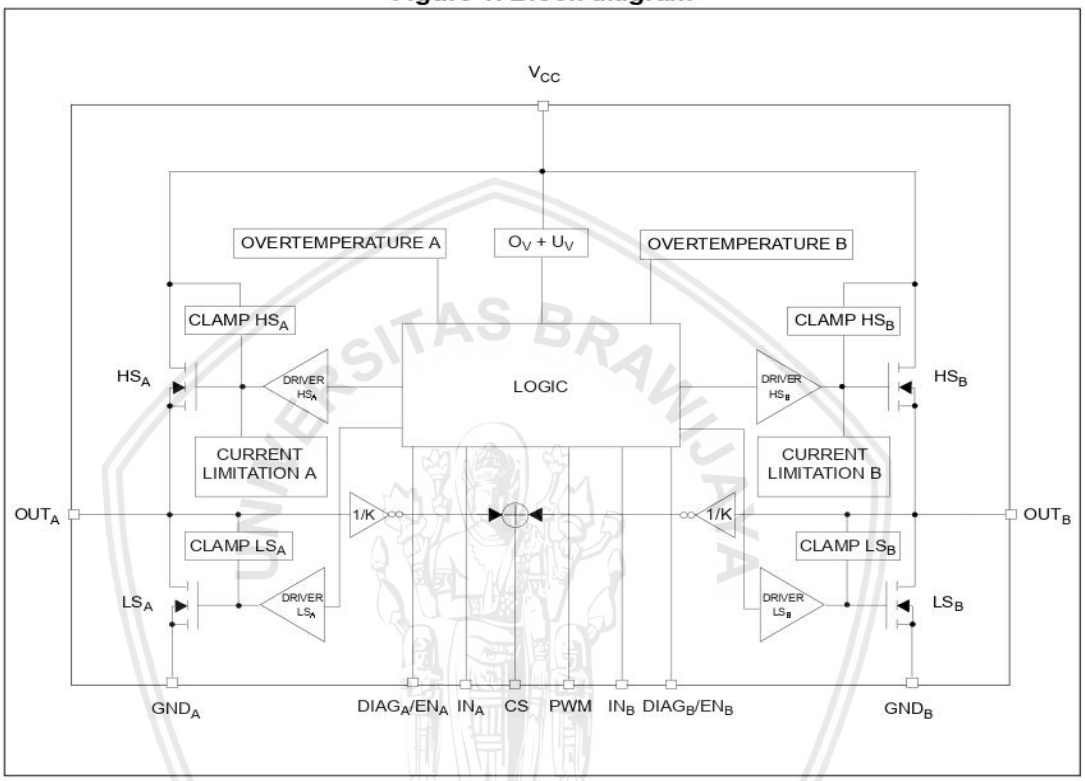


Table 2. Block description

Name	Description
Logic control	Allows the turn-on and the turn-off of the high side and the low side switches according to the truth table
Overvoltage + undervoltage	Shuts down the device outside the range [5.5V..16V] for the battery voltage
High side and low side clamp voltage	Protects the high side and the low side switches from the high voltage on the battery line in all configurations for the motor
High side and low side driver	Drives the gate of the concerned switch to allow a proper $R_{DS(on)}$ for the leg of the bridge
Linear current limiter	Limits the motor current by reducing the high side switch gate-source voltage when short-circuit to ground occurs
Overtemperature protection	In case of short-circuit with the increase of the junction's temperature, shuts down the concerned high side to prevent its degradation and to protect the die
Fault detection	Signals an abnormal behavior of the switches in the half-bridge A or B by pulling low the concerned $EN_x/DIAG_x$ pin

Figure 2. Configuration diagram (top view)

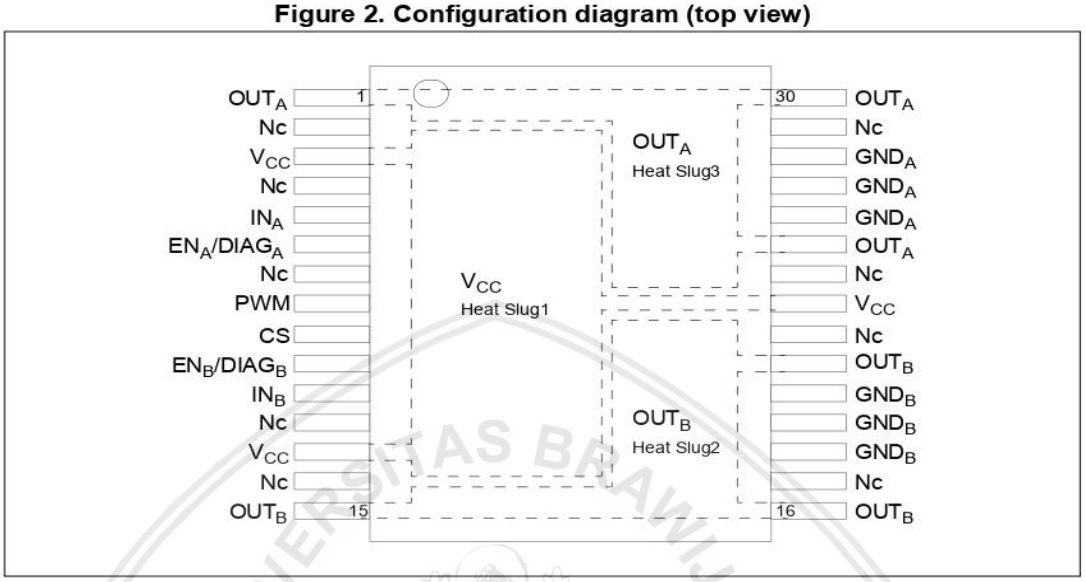


Table 3. Pin definitions and functions

Pin no.	Symbol	Function
1, 25, 30	OUT_A, Heat Slug3	Source of high side switch A / Drain of low side switch A
2, 4, 7, 12, 14, 17, 22, 24, 29	NC	Not connected
3, 13, 23	V_{CC}, Heat Slug1	Drain of high side switches and power supply voltage
6	$EN_A/DIAG_A$	Status of high side and low side switches A; open drain output
5	IN_A	Clockwise input
8	PWM	PWM input
9	CS	Output of current sense
11	IN_B	Counter clockwise input
10	$EN_B/DIAG_B$	Status of high side and low side switches B; open drain output
15, 16, 21	OUT_B, Heat Slug2	Source of high side switch B / Drain of low side switch B
26, 27, 28	GND_A	Source of low side switch A(1)
18, 19, 20	GND_B	Source of low side switch B(1)

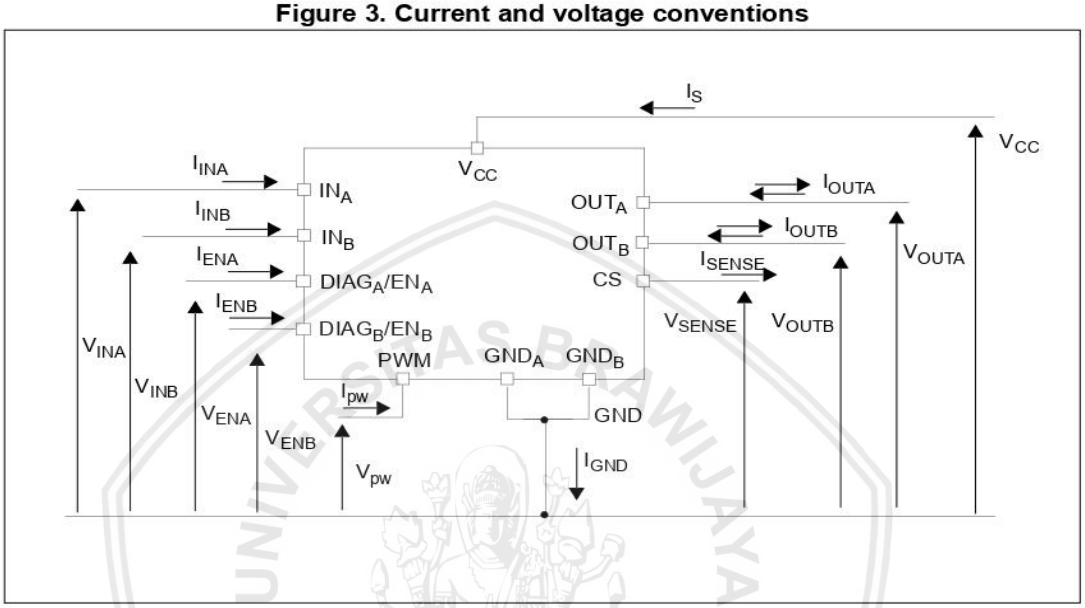
1. GND_A and GND_B must be externally connected together.

Table 4. Pin functions description

Name	Description
V_{CC}	Battery connection
GND_A, GND_B	Power grounds; must always be externally connected together
OUT_A, OUT_B	Power connections to the motor
IN_A, IN_B	Voltage controlled input pins with hysteresis, CMOS compatible. These two pins control the state of the bridge in normal operation according to the truth table (brake to V_{CC}, brake to GND, clockwise and counterclockwise).
PWM	Voltage controlled input pin with hysteresis, CMOS compatible. Gates of low side FETs are modulated by the PWM signal during their ON phase allowing speed control of the motor.
$EN_A/DIAG_A$, $EN_B/DIAG_B$	Open drain bidirectional logic pins. These pins must be connected to an external pull up resistor. When externally pulled low, they disable half-bridge A or B. In case of fault detection (thermal shutdown of a high side FET or excessive ON state voltage drop across a low side FET), these pins are pulled low by the device (see truth table in fault condition).
CS	Analog current sense output. This output sources a current proportional to the motor current. The information can be read back as an analog voltage across an external resistor.

2 Electrical specifications

Figure 3. Current and voltage conventions



2.1 Absolute maximum ratings

Table 5. Absolute maximum ratings

Symbol	Parameter	Value	Unit
V_{CC}	Supply voltage	+41	V
I_{max}	Maximum output current (continuous)	30	A
I_R	Reverse output current (continuous)	-30	
I_{IN}	Input current (IN_A and IN_B pins)	±10	mA
I_{EN}	Enable input current ($DIAG_A/EN_A$ and $DIAG_B/EN_B$ pins)	±10	
I_{pw}	PWM input current	±10	
V_{CS}	Current sense maximum voltage	-3/+15	V
V_{ESD}	Electrostatic discharge (R = 1.5kΩ, C = 100pF) – CS pin – logic pins – output pins: OUT_A, OUT_B, V_{CC}	 2 4 5	 kV kV kV
T_j	Junction operating temperature	Internally limited	°C
T_c	Case operating temperature	-40 to 150	
T_{STG}	Storage temperature	-55 to 150	

2.2 Electrical characteristics

V_{CC} = 9V up to 16 V; -40°C < T_J < 150°C, unless otherwise specified.

Table 6. Power section

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
V_{CC}	Operating supply voltage		5.5		16	V
I_S	Supply current	Off state with all Fault Cleared & ENx=0 IN_A = IN_B = PWM = 0; T_j = 25°C; V_{CC} = 13V IN_A = IN_B = PWM = 0 Off state: IN_A = IN_B = PWM = 0		12 2	30 60	µA µA mA
		On state: IN_A or IN_B = 5V, no PWM			10	mA
R_{ONHS}	Static high side resistance	I_{OUT} = 15A; T_j = 25°C			14	mΩ
		I_{OUT} = 15A; T_j = -40 to 150°C			28	
R_{ONLS}	Static low side resistance	I_{OUT} = 15A; T_j = 25°C			5	
		I_{OUT} = 15A; T_j = -40 to 150°C			10	
V_f	High side free-wheeling diode forward voltage	I_f = 15A		0.8	1.1	V
$I_{L(off)}$	High side off state output current (per channel)	T_j = 25°C; V_{OUTX} = EN_X = 0V; V_{CC} = 13V			3	µA
		T_j = 125°C; V_{OUTX} = EN_X = 0V; V_{CC} = 13V			5	
I_{RM}	Dynamic cross-conduction current	I_{OUT} = 15A (see *Figure 7*)		0.7		A

Table 7. Logic inputs (IN_A, IN_B, EN_A, EN_B)

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
V_{IL}	Input low level voltage	Normal operation ($DIAG_X$/EN_X pin acts as an input pin)			1.25	V
V_{IH}	Input high level voltage		3.25			
V_{IHYST}	Input hysteresis voltage		0.5			
V_{ICL}	Input clamp voltage	I_{IN} = 1mA	5.5	6.3	7.5	
		I_{IN} = -1mA	-1.0	-0.7	-0.3	
I_{INL}	Input low current	V_{IN} = 1.25V	1			µA
I_{INH}	Input high current	V_{IN} = 3.25V			10	
V_{DIAG}	Enable output low level voltage	Fault operation ($DIAG_X$/EN_X pin acts as an output pin); I_{EN} = 1mA			0.4	V

Incremental 40-mm-dia. Rotary Encoder
E6B2-C

CSM_E6B2-C_DS_E_6_1

General-purpose Encoder with External Diameter of 40 mm

- Incremental model
- External diameter of 40 mm.
- Resolution of up to 2,000 ppr.

⚠ Be sure to read *Safety Precautions* on page 4.

For the most recent information on models that have been certified for safety standards, refer to your OMRON website.

Ordering Information

Encoders [Refer to *Dimensions* on page 4.]

Power supply voltage	Output configuration	Resolution (pulses/rotation)	Model
5 to 24 VDC	NPN open-collector output	10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600 720, 800, 1,000, 1,024 1,200, 1,500, 1,800, 2,000	**E6B2-CWZ6C (resolution) 0.5M** Example: E6B2-CWZ6C 10P/R 0.5M
12 to 24 VDC	PNP open-collector output	100, 200, 360, 500, 600 1,000 2,000	**E6B2-CWZ5B (resolution) 0.5M** Example: E6B2-CWZ5B 100P/R 0.5M
5 to 12 VDC	Voltage output	10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600 1,000 1,200, 1,500, 1,800, 2,000	**E6B2-CWZ3E (resolution) 0.5M** Example: E6B2-CWZ3E 10P/R 0.5M
5 VDC	Line-driver output	10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600 1,000, 1,024 1,200, 1,500, 1,800, 2,000	**E6B2-CWZ1X (resolution) 0.5M** Example: E6B2-CWZ1X 10P/R 0.5M

Accessories (Order Separately) [Refer to *Dimensions* on *Rotary Encoder Accessories*.]

Name	Model	Remarks
Couplings	**E69-C06B**	Provided with the product.
	E69-C68B	Different end diameter
	E69-C610B	Different end diameter
	E69-C06M	Metal construction
Flanges	**E69-FBA**	---
	E69-FBA02	E69-2 Servo Mounting Bracket provided.
Servo Mounting Bracket	**E69-2**	---

Note: 1. Refer to *Rotary Encoders Accessories* on your OMRON website for details.
2. Refer to *Precautions For Correct Use of Rotary Encoders* on your OMRON website when using the Rotary Encoders together with a Coupling.

Ratings and Specifications

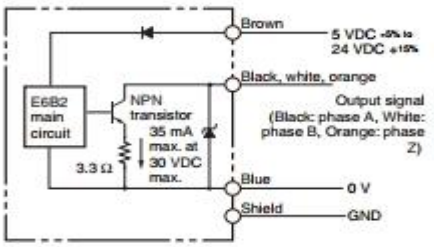
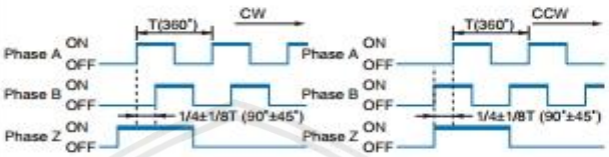
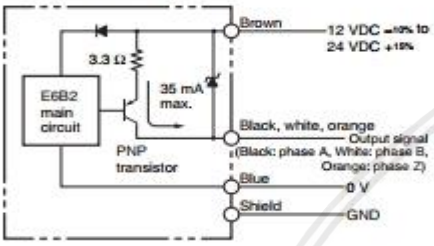
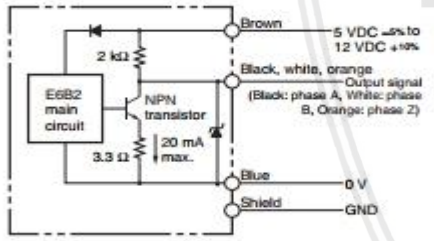
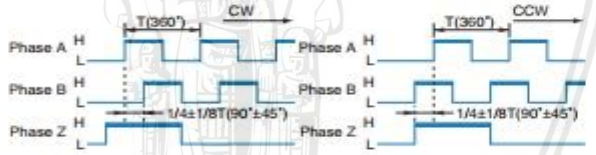
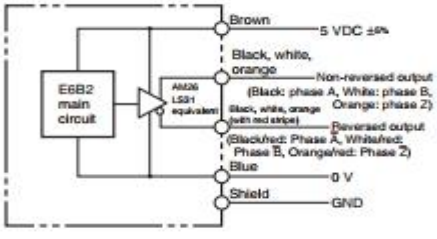
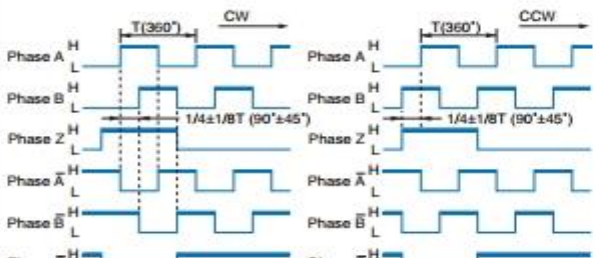
Item / Model		E6B2-CWZ6C	E6B2-CWZ5B	E6B2-CWZ3E	E6B2-CWZ1X
Power supply voltage		5 VDC –5% to 24 VDC +15%, ripple (p-p): 5% max.	12 VDC –10% to 24 VDC +15%, ripple (p-p): 5% max.	5 VDC –5% to 12 VDC +10%, ripple (p-p): 5% max.	5 VDC ±5%, ripple (p-p): 5% max.
Current consumption *1		80 mA max.	100 mA max.		160 mA max.
Resolution (pulses/rotation)		10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600, 720, 800, 1,000, 1,024, 1,200, 1,500, 1,800, 2,000	100, 200, 360, 500, 600, 1,000, 2,000	10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600, 1,000, 1,200, 1,500, 1,800, 2,000	10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600, 1,000, 1,024, 1,200, 1,500, 1,800, 2,000
Output phases		Phases A, B, and Z			Phases A, $\overline{A}$, B, $\overline{B}$, Z, and $\overline{Z}$
Phase difference between outputs		90°±45° between A and B (1/4 T ± 1/8 T)			
Output configuration		NPN open-collector output	PNP open-collector output	Voltage output (NPN output)	Line driver output *2
Output capacity		Applied voltage: 30 VDC max. Sink current: 35 mA max. Residual voltage: 0.4 V max. (at sink current of 35 mA)	Applied voltage: 30 VDC max. Source current: 35 mA max. Residual voltage: 0.4 V max. (at source current of 35 mA)	Output resistance: 2 kΩ Sink current: 20 mA max. Residual voltage: 0.4 V max. (at sink current of 20 mA)	AM26LS31 equivalent Output current High level: Io = –20 mA Low level: Is = 20 mA Output voltage: Vo = 2.5 V min. Vs = 0.5 V max.
Maximum response frequency *3		100 kHz	50 kHz	100 kHz	
Rise and fall times of output		1 µs max. (Control output voltage: 5 V, Load resistance: 1 kΩ, Cable length: 2 m max.)	1 µs max. (Cable length: 2 m max., Sink current: 10 mA)		0.1 µs max. (Cable length: 2 m max., Io = –20 mA, Is = 20 mA)
Starting torque		0.98 mN·m max.			
Moment of inertia		1×10^{-6} kg·m^2 max.; 3 × 10^{-7} kg·m^2 max. at 600 P/R max.			
Shaft loading	Radial	30 N			
	Thrust	20 N			
Maximum permissible speed		6,000 r/min			
Protection circuits		Power supply reverse polarity protection, Load short-circuit protection			---
Ambient temperature range		Operating: –10 to 70°C (with no icing), Storage: –25 to 85°C (with no icing)			
Ambient humidity range		Operating/Storage: 35% to 85% (with no condensation)			
Insulation resistance		20 MΩ min. (at 500 VDC) between current-carrying parts and case			
Dielectric strength		500 VAC, 50/60 Hz for 1 min between current-carrying parts and case			
Vibration resistance		Destruction: 10 to 500 Hz, 150 m/s^2 or 2-mm double amplitude for 11 min 3 times each in X, Y, and Z directions			
Shock resistance		Destruction: 1,000m/s^2 3 times each in X, Y, and Z directions			
Degree of protection		IEC 60529 IP50			
Connection method		Pre-wired Models (Standard cable length: 500 mm)			
Materials		Case: ABS, Main unit: Aluminum, Shaft: SUS420J2			
Weight (packed state)		Approx. 100 g			
Accessories		Coupling, Hexagonal wrench, Instruction manual			

*1. An inrush current of approximately 9 A will flow for approximately 0.3 ms when the power is turned ON.
*2. The line driver output is a data transmission circuit compatible with RS-422A and long-distance transmission is possible with a twisted-pair cable. The quality is equivalent to AM26LS31.
*3. The maximum electrical response speed is determined by the resolution and maximum response frequency as follows:

$$\text{Maximum electrical response speed (rpm)} = \frac{\text{Maximum response frequency}}{\text{Resolution}} \times 60$$

This means that the E6B2-C Rotary Encoder will not operate electrically if its speed exceeds the maximum electrical response speed.

I/O Circuit Diagrams

Model/Output Circuits	Output mode	Connection
E6B2-CWZ6C 	**E6B2-CWZ6C NPN Open-collector Output Model** **E6B2-CWZ5B PNP Open-collector Output Model** Direction of rotation: CW (as viewed from end of shaft)  Note: Phase A is 1/4 T ± 1/8 T faster than phase B. Direction of rotation: CCW (as viewed from end of shaft) Note: Phase A is 1/4 T ± 1/8 T slower than phase B. The ONs in the above timing chart mean that the output transistor is ON and the OFFs mean that the output transistor is OFF.	
E6B2-CWZ5B 		
E6B2-CWZ3E 	**E6B2-CWZ3E Voltage Output Model** Direction of rotation: CW (as viewed from end of shaft)  Note: Phase A is 1/4 T ± 1/8 T faster than phase B. Direction of rotation: CCW (as viewed from end of shaft) Note: Phase A is 1/4 T ± 1/8 T slower than phase B. "H" and "L" in the diagrams are the output voltage levels of phases A, B, and Z.	
E6B2-CWZ1X 	**E6B2-CWZ1X Line Driver Output Model** Direction of rotation: CW (as viewed from end of shaft)  Note: Phase A is 1/4 T ± 1/8 T faster than phase B. Direction of rotation: CCW (as viewed from end of shaft) Note: Phase A is 1/4 T ± 1/8 T slower than phase B. "H" and "L" in the diagrams are the output voltage levels of phases A, B, and Z.	

Color	Terminal
Brown	Power supply (+Vcc)
Black	Output phase A
White	Output phase B
Orange	Output phase Z
Blue	0 V (common)

Color	Terminal
Brown	Power supply (+Vcc)
Black	Output phase A
Black/red stripes	Output phase $\overline{A}$
White	Output phase B
White/red stripes	Output phase $\overline{B}$
Orange	Output phase Z
Orange/red stripes	Output phase $\overline{Z}$
Blue	0 V (common)

Note: Receiver: AM26LS32 equivalent

Note: 1. The shielded cable outer core (shield) is not connected to the inner area or to the case.
2. The phase A, phase B, and phase Z circuits are all identical.
3. Normally, connect GND to 0 V or to an external ground.

Safety Precautions

Be sure to read the precautions for all models in the website at: http://www.ia.omron.com/.

⚠ WARNING

This product is not designed or rated for ensuring safety of persons either directly or indirectly. Do not use it for such purposes.

Precautions for Safe Use

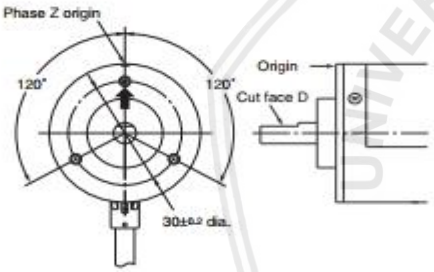
Incorrect wiring may damage internal circuits.

Precautions for Correct Use

Do not use the Encoder under ambient conditions that exceed the ratings.

● Mounting

- Origin Indication
  It is easy to adjust the position of phase Z with the origin indication function. The following illustration shows the relationship between phase Z and the origin. Set cut face D to the phase Z origin as shown in the illustration.



- Do not extend the length of the cable to more than 2 m. If the cable must be more than 2 m, use a Model with a Line-driver Output (max. length: 100 m).

● Wiring

Spurious pulses may be generated when power is turned ON and OFF. Wait at least 0.1 s after turning ON the power to the Encoder before using the connected device, and stop using the connected device at least 0.1 s before turning OFF the power to the Encoder. Also, turn ON the power to the load only after turning ON the power to the Encoder.

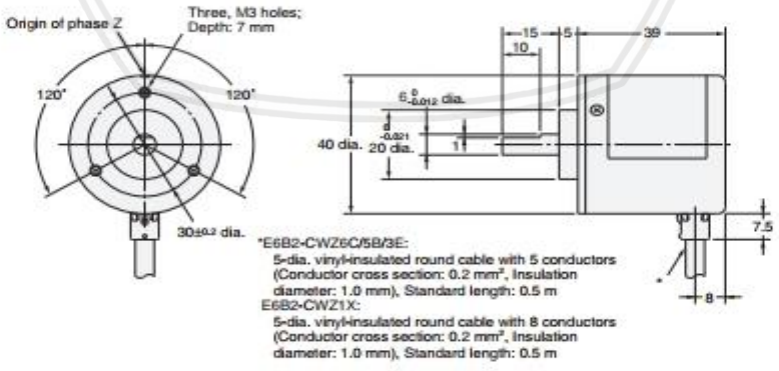
(Unit: mm)

Dimensions

Tolerance class IT16 applies to dimensions in this datasheet unless otherwise specified.

Encoder

E6B2-C



*E6B2-CWZ6C/5B/3E:
5-dia. vinyl-insulated round cable with 5 conductors (Conductor cross section: 0.2 mm², Insulation diameter: 1.0 mm), Standard length: 0.5 m
E6B2-CWZ1X:
5-dia. vinyl-insulated round cable with 8 conductors (Conductor cross section: 0.2 mm², Insulation diameter: 1.0 mm), Standard length: 0.5 m

Accessories (Order Separately)

Couplings
E69-C06B
E69-C68B
E69-C610B
E69-C06M

Flanges
E69-FBA
E69-FBA02

Servo Mounting Bracket
E69-2

Refer to *Rotary Encoders Accessories* on your OMRON website for details.

Terms and Conditions Agreement

Read and understand this catalog.
Please read and understand this catalog before purchasing the products. Please consult your OMRON representative if you have any questions or comments.

Warranties.
(a) Exclusive Warranty. Omron's exclusive warranty is that the Products will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Omron (or such other period expressed in writing by Omron). Omron disclaims all other warranties, express or implied.
(b) Limitations. OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE
PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.
Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right. (c) Buyer Remedy. Omron's sole obligation hereunder shall be, at Omron's election, to (i) replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Product, (ii) repair the non-complying Product, or (iii) repay or credit Buyer an amount equal to the purchase price of the non-complying Product; provided that in no event shall Omron be responsible for warranty, repair, indemnity or any other claims or expenses regarding the Products unless Omron's analysis confirms that the Products were properly handled, stored, installed and maintained and not subject to contamination, abuse, misuse or inappropriate modification. Return of any Products by Buyer must be approved in writing by Omron before shipment. Omron Companies shall not be liable for the suitability or unsuitability or the results from the use of Products in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.
See http://www.omron.com/global/ or contact your Omron representative for published information.

Limitation on Liability; Etc.
OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.
Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

Suitability of Use.
Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.
NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY OR IN LARGE QUANTITIES WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products.
Omron Companies shall not be responsible for the user's programming of a programmable Product, or any consequence thereof.

Performance Data.
Data presented in Omron Company websites, catalogs and other materials is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Omron's test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

Change in Specifications.
Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions.
Information presented by Omron Companies has been checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors or omissions.

2017.7

In the interest of product improvement, specifications are subject to change without notice.

OMRON Corporation
Industrial Automation Company
http://www.ia.omron.com/

维特智能 wit motion

htttp://www.wit-motion.com

WT61C Digital Attitude Sensor

SPECIFICATION

Model ： WT61C

Description ： Six axis Digital attitude angle sensor with case

Production Standard

Enterprise quality system standard: ISO9001:2016

Tilt switch production standard：GB/T191SJ 20873-2016

Criterion of detection：GB/T191SJ 20873-2016

Revision date：2017.10.18

Version	Update content	Author	Date
V1.0	Release	Snow	20171018

维特智能 wit motion

http://www.wit-motion.com

1 Description

- The purpose of this six axis gyro acceleration module uses high precision MPU6050 module, measuring data by the processor reads the MPU6050 and then output through the serial port, and elaborate PCB layout and process to ensure that the MPU6050 received the minimum interference with the highest measurement accuracy.
- An internal voltage stabilizing circuit module, voltage 3.3v~5v, pin compatible with the 3.3V/5V embedded system, convenient connection.
- The advanced digital filtering technology of this product can effectively reduce the measurement noise and improve the measurement accuracy.
- Integrates gesture solver, with dynamic Kalman filter algorithm, can get the accurate attitude in dynamic environment, attitude measurement precision is up to 0.05 degrees with high stability, performance is even better than some professional inclinometer.

Note: This module does not contain magnetometer, no filtering for the yaw angle, so yaw angle is calculated by integration, it will drift, the yaw angle is accurate only in a short time. The X, Y axis angle is accurate because it can be filtered by gravity field, it will not drift.

Built-in JY61 Gyrocope

2 Product Parameters

1) Voltage: 3.3V-5V.
2) Current: <10mA.
3) Volume: 51.3mm X 36mm X 15mm.
4) Measuring dimensions: Acceleration: 3D Angular Velocity: 3D Attitude angle: 3D

5) Range: Acceleration: ± 16g, angular velocity: ± 2000 ° / s, angle : X Y axis ±180° Z axis ±90° .
6) Resolution: Acceleration: 0.0005g, Angular velocity: 0.61 ° / s.
7) Stability: Acceleration: 0.01g, angular velocity 0.05° / s.
8) Attitude stabilization measurement: 0.01 °.
9) Data output: time, acceleration, angle.
10) Data output frequency 100Hz (baud rate 115200) / 20Hz (9600 baud).
11) Data interface: Serial port(TTL)
12) Baud rate: 9600, 115200(default)

3 Product Display

Pin	Function
VCC	Power supply，3.3V/5V input
RX	Serial data input，TTL/232 level
TX	Serial data output，TTL/232 level
GND	GND

4 Axial Direction

As shown in figure above, The axis of the module is in the upper picture, upward for x-axis, to the left for y-axis, Perpendicular to the paper, outward to the z-axis, The direction of rotation is defined by the Law of the right hand. That is to say the direction of four-fingers bending is the

direction of rotation around the axis. The thumb of the right hand points to axis.

5 Hardware Connection

5.1 Serial (TTL) Connection

When connected to the PC software, you need a USB -TTL module. Recommend the following two USB -TTL module:

1. Serial module TTL: Firstly connect the module with the USB - TTL and then connect them to the computer. The ways of connecting module with USB -TTL are:

VCC TX RX GND of the module connected to +3.3/5V RX TX GND of the serial module respectively. It is noteworthy that TX and RX need to be crossed--- RX connected to TX, TX connected to RX.

2. Serial debugging artifacts: set switch 1 to ON, set switch 2 to (silk) 2, switch S1 dial to the (near the figure near 232-485 silk screen), VCC TX RX GND of the module connected to +3.3/5V RX TX GND of the serial module respectively.

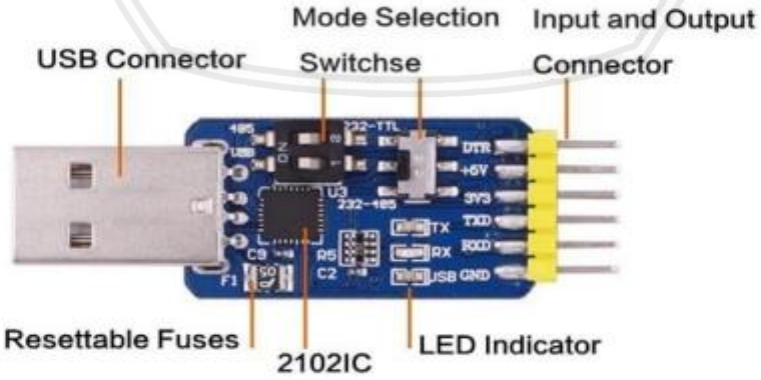
CONNECTION

Connect usb-ttl

Connect the serial port tool

5.2 232 Level Connection

When connected to the PC software, you need a USB -232 module. Recommend the following two USB -TTL module:
https://www.amazon.com/USB-TTL-232-converter-serial/dp/B01CNW061U/ref=sr_1_4?s=industrial&ie=UTF8&qid=1509332890&sr=1-4&keywords=witmotion



Serial debugging artifacts: set switch 1 to ON, set switch 2 to (silk) 2, switch S1 dial to the upper(near the figure near 232-TTL silk screen), VCC TX RX GND of the module connected to +3.3/5V 232R 232T GND of the serial module respectively.