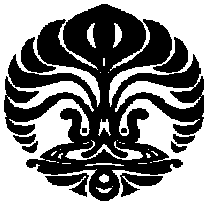


UNIVERSITAS INDONESIA

SISTEM KENDALI GERAK *CONTINUOUS PATH TRACKING* DENGAN MENGGUNAKAN *CUBIC TRAJECTORY PLANNING* PADA ROBOT MANIPULATOR 4 *DOF*

SKRIPSI

HARDIANSYAH RAHMAT NURHAKIM
0606073940

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JULI 2010

UNIVERSITAS INDONESIA

SISTEM KENDALI GERAK *CONTINUOUS PATH TRACKING* DENGAN MENGGUNAKAN *CUBIC TRAJECTORY PLANNING* PADA ROBOT MANIPULATOR 4 *DOF*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

HARDIANSYAH RAHMAT NURHAKIM
0606073940

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JULI 2010

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul :

SISTEM KENDALI GERAK *CONTINUOUS PATH TRACKING* DENGAN MENGGUNAKAN *CUBIC TRAJECTORY PLANNING* PADA ROBOT MANIPULATOR 4 *DOF*

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro, Departemen Teknik Elektro, Fakultas Teknik, Universitas Indonesia, sejauh yang saya ketahui, bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 9 Juli 2010

(Hardiansyah Rahmat N)
06 06 07 394 0

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama	: Hardiansyah Rahmat Nurhakim
NPM	: 0606073940
Program Studi	: Teknik Elektro
Judul Skripsi	:

SISTEM KENDALI GERAK *CONTINUOUS PATH TRACKING* DENGAN MENGGUNAKAN *CUBIC TRAJECTORY PLANNING* PADA ROBOT MANIPULATOR 4 *DOF*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr. Abdul Muis ST., M.Eng ( )

Penguji : Ir. Wahidin Wahab M.Sc, Ph.D ( )

Penguji : Dr. Ir. Feri Yusivar M.Eng ( )

Ditetapkan di : Depok

Tanggal : 9 Juli 2010

KATA PENGANTAR

Tiada kata yang patut penulis ucapkan selain Puji dan Syukur kepada Allah Subhanahu Wa Ta'ala atas rahmat dan pertolongan-Nya, sehingga penulis dapat menyelesaikan skripsi ini yang merupakan salah satu syarat mencapai gelar Sarjana Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Penulis menyadari bahwa, tanpa bantuan dari berbagai pihak, sangatlah sulit untuk menyelesaikan skripsi ini dengan baik dan tepat waktu. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Dr. Abdul Muis, ST, M.Eng, selaku dosen pembimbing yang telah meluangkan waktunya untuk memberikan arahan, bimbingan dan diskusi sehingga skripsi ini dapat diselesaikan dengan baik.
2. Orang tua dan keluarga tercinta yang telah memberikan motivasi dan dukungan, semoga ini menjadi salah satu kebanggaan kalian.
3. Para peneliti sebelum ini yang memberikan sumber bacaan yang banyak bagi penulis.
4. Sahabat-sahabat perjuangan Tim Robot UI, yang telah mewarnai hidup penulis selama disini, semoga prestasi yang diraih semakin baik lagi.
5. Rekan-rekan mahasiswa Departemen Teknik Elektro FTUI yang telah memberikan bantuan dalam penulisan skripsi ini.
6. Dan seluruh civitas akademika Departemen Teknik Elektro yang tidak dapat saya sebutkan satu persatu.

Akhir kata, semoga Allah SWT membalas segala kebaikan semua pihak yang telah membantu Penulis dalam penyusunan skripsi ini. Dan semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan.

Depok, Juli 2010

Penulis

Hardiansyah Rahmat

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Universitas Indonesia, saya bertanda tangan di bawah ini :

Nama	: Hardiansyah Rahmat N
NPM	: 0606073940
Program studi	: Teknik Kendali
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul :

SISTEM KENDALI GERAK *CONTINUOUS PATH TRACKING*
DENGAN MENGGUNAKAN *CUBIC TRAJECTORY PLANNING*
PADA ROBOT MANIPULATOR 4 *DOF*

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia / formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta sebagai pemegang Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 9 Juli 2010
Yang menyatakan

Hardiansyah Rahmat N

ABSTRAK

Nama : Hardiansyah Rahmat N
Program Studi : Teknik Elektro
Judul : SISTEM KENDALI GERAK CONTINUOUS PATH TRACKING MENGGUNAKAN CUBIC TRAJECTORY PLANNING PADA ROBOT MANIPULATOR 4 DOF

Aplikasi robotika di lingkungan industri telah memberikan berbagai keuntungannya. Definisi robot industri atau yang juga dikenal dengan robot manipulator pun muncul. Robot manipulator tersebut biasanya berbentuk tangan yang diciptakan untuk satu atau beberapa fungsi tertentu.

Pada tugas akhir ini dirancang sebuah prototipe robot manipulator berbentuk tangan 4 DOF dengan konfigurasi sendi PRRR. Sendi robot manipulator ini digerakkan dengan motor DC dan motor servo yang dikendalikan secara *closed-loop* menggunakan mikrokontroler. Pergerakan sendi dan *end-effector* robot direncanakan menggunakan *trajectory planning*, dan diperintahkan serta dapat diamati secara real-time oleh software komputer berbasis .net.

Dari hasil pengujian, robot manipulator ini telah dapat dikendalikan untuk pergerakan titik ke titik dengan tingkat akurasi mencapai 2.22 %. Robot pun telah mampu bergerak mengikuti lintasan garis lurus.

Kata kunci : Robot, manipulator, motor servo, motor dc, .net, c#, *kinematics*, *trajectory planning*, *cubic polynomial*, *continuous path tracking*

ABSTRACT

Name : Hardiansyah Rahmat
Study Program : Electrical Engineering
Judul : CONTINUOUS PATH TRACKING MOTION CONTROL USING CUBIC POLYNOMIAL TRAJECTORY PLANNING IN ROBOT MANIPULATOR WITH 4 DOF

Applications of Robotics in industry has given many advantages. The definition of industrial robot or manipulator robot emerge. The manipulator robot is usually arm-shaped which is created for one or several particular functions.

A prototype of arm-shaped 4 DOF manipulator robot with PRRR joints configuration is produced. Joints of this manipulator robot is moved by DC and servo motor which is controlled in closed-loop with microcontroller. The movement of the joints and end-effector robot are designed with a trajectory planning, then commanded and examined in real-time using computer software based on .net.

According to the result, the manipulator robot can be controlled for point to point motion with accuracy up to 2.22%. The robot can also move tracking the straight path continuosly.

Key words : robot, manipulator, servo motor, dc motor, .net, c#, *kinematics*, *trajectory planning*, *cubic polynomial*, *continuous path tracking*

DAFTAR ISI

HALAMAN JUDUL i
PERNYATAAN ORISINALITAS SKRIPSI ii
HALAMAN PENGESAHAN iii
KATA PENGANTAR iv
HALAMAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS v
ABSTRAK vi
ABSTRACT ..... **vii**
DAFTAR ISI ..... **viii**
DAFTAR GAMBAR ..... **xi**
DAFTAR TABEL ..... **xiii**
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan Masalah 1
1.3 Tujuan 2
1.4 Batasan Masalah 2
1.5 Metodologi 2
1.6 Sistematika Penulisan 2
BAB 2 LANDASAN TEORI 6
2.1 Robot Manipulator 6
2.1.1 Klasifikasi Robot Manipulator 7
2.1.2 Konfigurasi Kinematik Robot Manipulator 8
2.2 Analisis Kinematika Robot Manipulator 10
2.2.1 Kinematika Vs Dinamika Robot Manipulator 10
2.2.2 Kontrol Kinematika Robot 11
2.3 Sistem Kendali Lintasan Gerak Robot Manipulator 13
2.3.1 Path Planning Dengan Interpolasi Linier 14
2.3.2 Trajectory Planning 14
2.3.2.1 Joint Space Trajectory Planning 14
2.3.2.2 Cartesian Trajectory Planning 17
2.4 Sistem Kendali Aktuator Pada Sendi Robot 18
2.4.1 Sistem Kendali Open Loop Vs Closed Loop 18
2.4.2 Independent Joint Control 19
2.5 Aktuator Pada Robot Manipulator 19
2.5.1 Motor DC 19
2.5.2 Motor Servo 20
2.6 Sensor Posisi Pada Robot Manipulator 21
2.6.1 Potensiometer 21

2.6.2 Rotary Incremental Encoder 21
2.6.3 Limit Switch 22
2.7 Mikrokontroler Atmega 2560 22
2.7.1 Pin Input Output 23
2.7.2 Timer Penghasil Sinyal Pwm 24
2.8 Pengembangan Software Antarmuka Berbasis .Net 25
2.9 Komunikasi Komputer Dengan Mikrokontroler 26

BAB 3 PERANCANGAN SISTEM 27

3.1 Perancangan Sistem 27
3.2 Perancangan Mekanik 28
3.2.1 Sendi Translasi 29
3.2.2 Lengan Robot 3 DOF 29
3.2.3 Wrist Dan Gripper 31
3.3 Perancangan Sistem Elektrik 31
3.3.1 Distribusi Daya Dan Diagram Sinyal Elektrik 31
3.3.2 Mikrokontroler Atmega2560 32
3.4 Perancangan Perangkat Lunak 34
3.4.1 Program Pada PC 34
3.4.1.1 Program Forward Kinematics 34
3.4.1.2 Program Inverse Kinematics 35
3.4.1.3 Program Joint Space Trajectory Planning 37
3.4.1.4 Program Path Planning Dengan Interpolasi Linier 38
3.4.2 Program Pada Mikrokontroler 39
3.4.2.1 Program Komunikasi Serial 39
3.4.2.2 Program Gerak Sinkron Motor Servo 41
3.4.2.3 Program Kendali Motor DC 43
3.4.3 Simulasi Program 44
3.4.3.1 Point To Point Motion 44
3.4.3.2 Continuous Path Tracking 45

BAB 4 PENGUJIAN DAN ANALISIS SISTEM 51

4.1 Pengujian Perangkat Keras 51
4.1.1 Uji Linearitas Dan Histerisis Potensiometer 51
4.1.2 Uji Karakteristik Motor Servo Gws 2BBMG 53
4.2 Pengujian Sistem Untuk Gerak Point To Point Dengan Joint Space Trajectory Planning 55
4.2.1 Uji Gerak Sendi dengan Linear Trajectory Planning 55
4.2.2 Uji Gerak Sendi Menggunakan Cubic Polynomial 57
4.2.3 Uji Gerak Sendi Berbeban Menggunakan Cubic Polynomial 59
4.3 Analisis Hasil Pengujian Sistem Untuk Gerak Point To Point 61
4.3.1 Linear vs Cubic Trajectory Planning 61
4.3.2 Cubic Trajectory Planning Pada Sendi Berbeban 63
4.3.3 Analisis Keterlambatan Gerak Sendi Dalam Mengikuti Trajectory 65
4.4 Pengujian Sistem Untuk Gerak Continuos Path Tracking dengan Cubic Polynomial Trajectory Planning Pada Joint Space 69
4.4.1 Uji Gerak End-Effector Dengan Perhentian Di Setiap Titik Singgah 70
4.4.2 Uji Gerak End-Effector Tanpa Perhentian Di Setiap Titik Singgah 71

4.5 Analisis Hasil Pengujian Sistem Gerak Continuos Path Tracking 73
BAB 5 KESIMPULAN 74
DAFTAR REFERENSI 76
LAMPIRAN A : Perancangan Mekanik Robot Manipulator
LAMPIRAN B : Perancangan Elektrik Robot Manipulator
LAMPIRAN C : Tutorial Penggunaan Software

DAFTAR GAMBAR

Gambar 2.1 KOMPONEN SISTEM ROBOT MANIPULATOR 6
Gambar 2.2 Simbol Representasi Sendi Putar dan Sendi Prismatik 8
Gambar 2.3 (a) Konfigurasi Kinematic Chain Articulated Manipulator
(b) Workspace Articulated Manipulator 9
Gambar 2.4 (a) Konfigurasi Kinematic-Chain Spherical Manipulator,
(b) workspacespherical Manipulator 9
Gambar 2.5 (a) Konfigurasi Kinematic-Chain SCARA manipulator,
(b) Workspace SCARA Manipulator 9
Gambar 2.6 (a) Konfigurasi Kinematic-Chain cylindrical manipulator,
(b) workspace cylindrical Manipulator 10
Gambar 2.7 (a) Konfigurasi Kinematic-Chain Cartesian manipulator,
(b) workspace Cartesian Manipulator 10
Gambar 2.8 Model kontrol kinematika robot 11
Gambar 2.9 Manipulator dua sendi 12
Gambar 2.10 Manipulator tiga sendi 12
Gambar 2.11 Tiga segmen trajectory LSPB 16
Gambar 2.12 Blok Diagram Sistem Kendali open loop 18
Gambar 2.13 Blok Diagram Sistem Kendali closed loop 18
Gambar 2.14 Blok Diagram Independent Joint Control 19
Gambar 2.15 Variasi lebar pulsa positif terhadap sudut servo 20
Gambar 2.16 Pemasangan encoder pada motor 22
Gambar 2.17 Limit Switch 22
Gambar 2.18 Phase PWM Generator 25
Gambar 3.1 Rancangan sistem 27
Gambar 3.2 (a) Konfigurasi Kinematic-Chain sebuah lengan robot manipulator,
(b)workspace sebuah lengan robot manipulator 28
Gambar 3.3 Konfigurasi model lengan Robot Manipulator 28
Gambar 3.4 Desain Sendi Translasi 29
Gambar 3.5 (a) Dudukan servo (b) desain lengan penghubung
(c) Penempatan potensiometer pada lengan 30
Gambar 3.6 (a) Desain wrist, (b) desain gripper 31
Gambar 3.7 Distribusi Daya Sistem 31
Gambar 3.8 Konfigurasi PIN Atmega2560 pada Sistem 33
Gambar 3.9 Flowchart Program Forward Kinematics 35
Gambar 3.10 Flowchart Program Inverse Kinematics 36
Gambar 3.11 Flowchart Program Cubic Polynomial Trajectory Planning 38
Gambar 3.12 Flowchart Program Path Planning dengan interpolasi linier 39
Gambar 3.13 (a) Contoh Paket Data yang dikirim Komputer ke Mikrokontroler,
(b) Alur Komunikasi data Komputer dan Mikrokontroler 40
Gambar 3.14 Paket berisi data sensor posisi 41
Gambar 3.15 Keluaran tiga sinyal PWM berbeda dengan metode Phase and
Frequency Correct PWM 43
Gambar 3.16 Blok Diagram Pengendali motor DC 44
Gambar 3.17 Perubahan posisi pada linear trajectory planning 45

Gambar 3.18 Trajectory planning dengan Cubic polynomial 46
Gambar 3.19 Koodinat titik-titik singgah hasil Interpolasi Linier garis lurus 47
Gambar 3.20 Posisi End-effector untuk path points pada tabel 3.4 48
Gambar 3.21 Perubahan posisi End-effector pada sumbu X untuk path points pada tabel 3.4 49
Gambar 3.22 Perubahan posisi End-effector pada sumbu Y untuk path points pada tabel 3.4 50

Gambar 4.1 Pembacaan ADC pada potensiometer dengan kecepatan servo lambat 51
Gambar 4.2 Pembacaan ADC pada potensiometer dengan kecepatan servo cepat 52
Gambar 4.3 Resolusi Servo GWS 2BBMG 53
Gambar 4.4 Grafik hasil uji gerak point to point motion menggunakan linear trajectory planning 57
Gambar 4.5 Grafik hasil uji gerak point to point motion menggunakan cubic polynomial trajectory planning 59
Gambar 4.6 Grafik hasil uji gerak point to point motion menggunakan cubic polynomial trajectory planning pada sendi berbeban 61
Gambar 4.7 Pergerakan sendi pada linear trajectory planning 63
Gambar 4.8 Pergerakan sendi pada cubic trajectory planning 64
Gambar 4.9 Keterlambatan yang terjadi pada linear trajectory planning 67
Gambar 4.10 Keterlambatan yang terjadi pada cubic trajectory planning 68
Gambar 4.11 Koordinat titik-titik singgah hasil Path Planning 69
Gambar 4.12 Pergerakan end-effector pada sumbu X 70
Gambar 4.13 Pergerakan end-effector pada sumbu Y 71
Gambar 4.14 Pergerakan end-effector pada koordinat cartesian 71
Gambar 4.15 Pergerakan end-effector pada sumbu X 72
Gambar 4.16 Pergerakan end-effector pada sumbu Y 72
Gambar 4.17 Pergerakan end-effector pada koordinat cartesian 72

DAFTAR TABEL

Tabel 3.1	Konfigurasi lengan manipulator	29
Tabel 3.2	Konfigurasi PIN di Atmega2560 pada Sistem	32
Tabel 3.3	Koodinat titik-titik singgah hasil Interpolasi Linier garis lurus	47
Tabel 3.4	Hasil Perhitungan Inverse Kinematics untuk tiap titik singgah pada lintasan garis lurus	48
Tabel 4.1	Kalibrasi servo pada posisi maksimal dan netralnya	54
Tabel 4.2	Hasil uji gerak point to point motion menggunakan linear trajectory planning	57
Tabel 4.3	Hasil uji gerak point to point motion menggunakan cubic trajectory planning	60
Tabel 4.4	Hasil uji gerak point to point motion menggunakan cubic trajectory planning pada kondisi berbeban	62
Tabel 4.5	Koodinat titik-titik singgah	69
Tabel 4.6	Konfigurasi sendi di tiap titik singgah cubic polynomial trajectory planning pada sendi berbeban	70

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Robotika adalah bidang ilmu pengetahuan yang masih relatif muda dari teknologi modern yang kita ketahui telah melampaui batasan teknologi tradisional. Bidang ilmu ini sebelumnya adalah bagian dari bidang ilmu pengetahuan seperti teknik elektro, teknik mesin, sistem kontrol, otomasi industri maupun ilmu komputer. Namun kini robotika telah berdiri sendiri, bahkan menjadi sebuah disiplin ilmu yang sangat kompleks. Dalam memahami kompleksitas robotika dan aplikasinya, dibutuhkan pengetahuan pendukung dari berbagai disiplin ilmu lainnya yang hingga sekarang terus berkembang pesat seperti teknik kontrol, teknik instrumentasi, teknik manufaktur, teknik material, piranti elektronika, otomotif, teknik telekomunikasi dan terutama sekali teknologi komputer dan informasi khususnya kecerdasan buatan.

Dari awal berkembangnya robotika, sebagian besar aplikasinya tak dapat dipisahkan dengan dunia industri. Istilah industrial robot dan robot manipulator pun muncul sebagai julukan bagi robot di dunia industri tersebut. Teknologi robotika yang beroperasi di lingkungan pabrik industri memberikan berbagai keuntungan diantaranya menurunkan biaya produksi, meningkatkan ketepatan dan produktivitas, meningkatkan fleksibilitasnya dibandingkan dengan mesin khusus, dan kondisi kerja yang lebih manusiawi seperti membosankan, berulang, atau pekerjaan berbahaya dapat dilakukan oleh robot. Sehingga pengenalan pertama dari bidang ilmu robotika harus mencakup dasar-dasar yang kuat untuk mendukung aplikasi tersebut.

Definisi yang popular di dunia mengenai bentuk robot industri atau robot manipulator adalah sebuah robot tangan yang diciptakan untuk satu atau beberapa fungsi tertentu, memiliki bentuk lengan-lengan kaku yang terhubung secara seri dengan sendi yang dapat bergerak berputar (rotasi) atau memanjang dan memendek (translasi). Satu sisi lengan disebut sebagai pangkal dan ditanam pada bidang atau meja yang diam, sedangkan sisi lengan lainnya disebut sebagai *end*

effector yang dapat dimuati oleh peralatan yang sesuai dan mendukung fungsi robot tersebut.

Dalam tulisan ini, istilah robot akan lebih ditekankan pada definisi robot industri seperti yang telah dijelaskan diatas walaupun sebenarnya saat ini robot industri sudah menerapkan teknologi mobile robot. Sehingga pengertian robot pun menjadi lebih spesifik yaitu sebagai robot industri yang merupakan manipulator multifungsi terprogram yang dirancang untuk menggerakkan berbagai objek dan melakukan fungsi tertentu. Manipulator sendiri adalah sebuah sistem mekanis yang membutuhkan metode pemodelan matematika untuk mewakili aspek geometrik, dan manipulasi aspek dinamisnya. Robot terprogram berarti pada robot dibutuhkan otak berupa komputer yang diprogram sehingga memungkinkan robot melakukan fungsinya serta kemampuannya dalam beradaptasi melalui berbagai jenis sensor yang tersedia. Dengan pemodelan dan pemrograman tersebut, kita akan mampu merancang, merencanakan mengendalikan gerakan robot untuk melakukan tugas-tugas spesifik tertentu. Dan pembahasan dua komponen inilah yang akan menjadi topik utama pada makalah ini.

Dalam aplikasinya robot manipulator berugas untuk melakukan fungsi spesifik tertentu. Dibutuhkan sistem kendali gerak lebih lanjut yang dapat membuat robot maniputor tersebut mengerjakan fungsinya dengan tingkat keakuratan dan *repeatability* yang tinggi. Secara umum ada dua sistem kendali gerak yang di terapkan pada robot manipulator. Sistem kendali yang pertama adalah point to point motion control (PTP). Pada sistem kendali gerak PTP robot manipulator melaksanakan tugas utama untuk berpindah dari satu titik ke titik lain. Contoh aplikasi sistem kendali gerak ini adalah pada mesin bor otomatis seperti CNC. Sistem kendali gerak yang satunya lagi dikenal dengan *continuous path control*. Berbeda dengan PTP yang hanya berorientasi pada titik awal dan akhir tobot, sistem kendali gerak *continuous path control* memungkinkan robot untuk mengikuti lintasan tertentu.

Pada tulisan ini akan dirancang sebuah robot manipulator yang dapat diaplikasikan kedua sistem kendali gerak tersebut. Robot manipulator tersebut adalah robot manipulator 4 DOF dengan konfigurasi sendi PRRR. Penggerak yang digunakan berupa motor servo untuk setiap sendi rotasi dan motor DC pada

sendi prismatik. Sama halnya pada industri, pergerakan robot manipulator ini dirancang untuk dapat dikendalikan dan dikonfigurasi ulang oleh komputer dengan perangkat lunak berbasis .Net menggunakan bahasa pemrograman C#.

1.2 Perumusan Masalah

Secara umum, masalah yang menjadi fokus perhatian dari sebuah robot manipulator yang bisa dikaji dan dianalisis adalah pada beberapa hal berikut diantaranya :

1. Perancangan robot manipulator dengan memperhatikan aspek geometris dan aspek dinamis robot.
2. Analisis kinematika robot
3. Pemilihan aktuator yang tepat.
4. Penentuan sistem kendali aktuator yang handal.
5. Sistem kendali multi aktuator robot.
6. Perencanaan jalur lintasan robot.
7. Sistem kendali gerak dan posisi *end-effector* robot.
8. Pengendalian robot dengan software komputer melalui mikrokontroler.

1.3 Tujuan

Tujuan dari tugas akhir ini adalah untuk merancang dan kemudian memproduksi sebuah robot manipulator berupa lengan mekanik dengan beberapa jenis aktuator, yang gerakannya direncanakan dan diperintahkan secara langsung oleh *software* komputer, dan dikendalikan secara *closed-loop* dengan feedback dari sensor posisi menggunakan mikrokontroler.

1.4 Batasan Masalah

Melihat kompleksitas dan keterbatasan penulis akan pengetahuan di disiplin ilmu yang lain, maka pembahasan mengenai pembuatan robot manipulator pada makalah ini pun akan dibatasi pada beberapa subjek diantaranya:

a. Perancangan lengan mekanik manipulator 6 DOF. Dimana 4 sendi saja yang akan menentukan orientasi posisi robot sedangkan 2 sendi lainnya adalah *end effector* robot.

b. Sistem kendali *closed-loop* beberapa buah aktuator menggunakan mikrokontroler.
c. Pemodelan matematika untuk analisis kinematik manipulator dengan menggunakan persamaan trigonometri.
d. Perencanaan posisi robot berdasarkan waktu untuk sistem kendali *point to point* dan *continuous path tracking* menggunakan *joint space trajectory planning*.
e. Kendali *real-time* manipulator oleh *software* komputer berbasis visual studio C# menggunakan jalur serial RS-232.

1.5 Metodologi

Berikut adalah gambaran tentang robot manipulator dan langkah-langkah yang dilakukan dalam penyelesaian tugas akhir ini :

1. Rancang bangun robot manipulator, meliputi :
   - Desain dan produksi lengan 6 DOF
   - Pemilihan aktuator untuk sendi-sendi robot
   - Pemasangan sensor *feedback* untuk beberapa aktuator
   - Kendali *closed loop* aktuator.
   - Kendali multi aktuator dengan mikrokontroler
   - Komunikasi serial RS-232 antara pc dan mikrokontroler
2. Rancang bangung *software* pc berbasis .net
   - Perhitungan *forward* dan *inverse kinematics* robot
   - Perencanaan posisi untuk gerak tiap aktuator
   - Simulasi real-time pergerakan robot
   - Representasi grafik posisi aktuator acuan dan kenyataan
   - Representasi grafik posisi *end-effector* acuan dan kenyataan
3. Pengujian sistem dan pengambilan data pada beberapa acuan waktu dan beban yang berbeda.
4. Analsis data dan penarikan kesimpulan

1.6 Sistematika Penulisan

Pembahasan mengenai rancang bangun robot manipulator pada tulisan ini dijelaskan secara terperinci dengan sistematika penulisan sebagai berikut :

BAB 1 Pendahuluan

Bab 1 berisi gambaran permasalahan secara umum yang diangkat dalam penelitian ini. Bab ini menjelaskan latar belakang masalah, rumusan masalah, tujuan penelitian, dan sistematika penulisan penelitian ini.

BAB 2 Dasar Teori

Bab 2 berisi tentang tinjauan literatur mengenai berbagai pembahasan tentang sistem mekanik, elektrik, dan kendali robot manipulator.

BAB 3 Perancangan Sistem Robot Manipulator

Bab 3 berisi tentang rancang bangun sistem mekanik, elektrik, pemrograman mikrokontroler untuk kendali *closed-loop* dan multi degree beberapa aktuator, penggunaan *software* antarmuka berbasis visual studio C#, komunikasi data serial untuk kendali robot melalui komputer serta sistem kendali gerak robot.

BAB 4 Pengujian dan Analisis Sistem Robot Manipulator

Bab 4 berisi tentang pengujian dan analisis penggunaan sensor posisi, algoritma kendali aktuator pada mikrokontroler, komunikasi serial, serta kendali gerak aktuator dan posisi *end-effector* robot.

BAB 5 Penutup

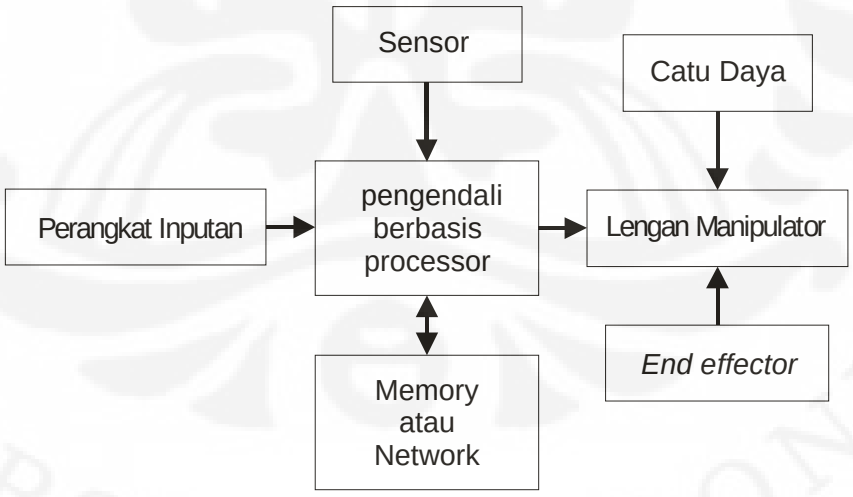
Bab 5 berisi kesimpulan dari rancang bangun sistem robot manipulator

BAB 2
DASAR TEORI

2.1 Robot Manipulator

Robot industri pada awalnya diilustrikan sebagai robot manipulator yaitu robot berbentuk lengan lengkap dari mulai tangan, pergelangan tangan, dan ujung tangan yang dapat dipasang berbagai peralatan berkesesuaian sehingga dapat berfungsi untuk melakukan tugas yang dikehendaki. Sebuah robot manipulator harus dipandang sebagai lebih dari sekedar serangkaian hubungan mekanis. Lengan mekanik (manipulator) hanyalah salah satu komponen dalam sistem robotika yang secara keseluruhan terdiri dari lengan mekaniknya itu sendiri, sumber daya eksternal, fungsi dari ujung lengan robot manipulator misalnya sebagai gripper, sensor internal dan eksternal, antar muka komputer, dan komputer pengendali seperti yang ditunjukkan oleh gambar 2.1. Bahkan perangkat lunak yang diprogram harus dipertimbangkan sebagai bagian integral dari keseluruhan sistem, karena bagaimana robot diprogram dan dikendalikan dapat berdampak besar pada kinerja, keakuratan dan *repeatabilty*

.



Gambar 2.1 Komponen sistem robot manipulator

Robot manipulator memiliki tiga tingkatan / level teknologi, yaitu : Rendah, medium dan tinggi. Masing-masing memiliki karakteristik sendiri yang sangat unik sehingga sangat jelas perbedaan antara ketiga level tersebut. Tingkatan teknologi robot manipulator dapat dibedakan dengan mengetahui

karakteristik masing robot diantaranya jumlah *Degree of Freedom*, kapasitas beban, kecepatan, akurasi, aktuasi dan pengendalinya.

2.1.1 Klasifikasi Robot Manipulator

Robot manipulator dapat diklasifikasikan dalam beberapa kriteria, diantaranya :

1. Catu daya robot

   Catu daya adalah sebuah unit yang menyediakan tenaga pada robot manipulator sehingga dapat bekerja. Catu daya dalam suatu sistem robot manipulator dibagi menjadi dua bagian, yaitu untuk bagian kontroler dan manipulator. Pada bagian kontroler biasanya digunakan catu daya elektrik, sedangkan pada bagian manipulator bisa digunakan catu daya elektrik, pneumatik, atau hidrolik.

2. Lingkup Aplikasi

   Robot sering diklasifikasikan berdasarkan aplikasinya yaitu sebagai robot *assembly* dan *non-assembly*. Robot *assembly* digunakan dalam proses perakitan, dibuat cenderung kecil, dan di catu oleh daya elektrik. Sedangkan robot *non-assembly* utama digunakan dalam proses pengelasan, pengecatan, *material handling*, dan mesin bongkar muat.

3. Metode Kontrol

   Berdasarkan metode kontrolnya robot manipulator dibedakan menjadi robot *servo* dan robot *non-servo*. Robot yang pertama kali dibuat adalah robot *non-servo*. Robot ini pada dasarnya adalah perangkat dengan kontrol *open loop* yang gerak dan berhentinya dibatasi oleh sistem mekanik yang telah ditentukan (secara hardware). Berbeda dengan robot *non-servo,* robot servo menggunakan komputer untuk kontrol *closed loop* dalam menentukan gerakannya. Robot servo dapat diprogram untuk melakukan berbagai fungsi, namun kurang memiliki repeatability yang baik tanpa bantuan sensor pendukung.

   Pengendalian robot servo dibedakan sesuai dengan metode yang digunakan dalam menempatkan *end-effector* yaitu metode *point-to-point* dan metode *continuous path.* Robot *point-to-point* akan bergerak menuju serangkaian titik-titik tujuan, tetapi tidak ada kontrol di jalan penghubung

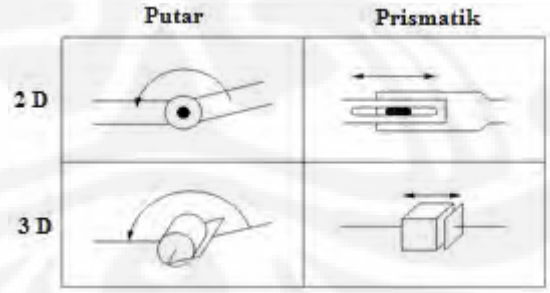
titik-titiknya. Berbeda dengan continuous path dimana seluruh jalur *end-effector* dapat dikontrol.

4. Geometri.

   Sebagian besar manipulator industri pada saat ini memiliki enam atau lebih sedikit derajat-of-kebebasan. Manipulator ini biasanya digolongkan secara kinematis berdasarkan tiga sendi lengan pertama dan dengan pergelangan tangan (*wrist* dan *end-effector*) yang dijelaskan secara terpisah. Hampir semua manipulator masuk ke dalam salah satu dari lima jenis geometrik berikut : articulated (RRR), spherical (RRP), SCARA (RRP), cylindrical (RPP), atau Cartesian (PPP) yang akan dibahas pada subbab berikutnya.

2.1.2 Konfigurasi Kinematik Robot Manipulator

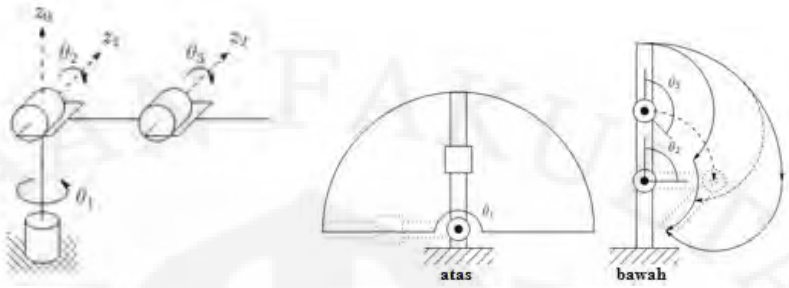
Robot Manipulator terdiri dari lengan-lengan *(links)* yang dihubungkan oleh sendi (*joints*) untuk membentuk rantai kinematik (*kinematic chain*). Sendi biasanya berputar (*revolute*) atau linier (*prismatik*). Sebuah sendi putar layaknya sebuah engsel yang memungkinkan gerak putar antara dua link. Sedangkan sebuah sendi prismatik memungkinkan gerak linier antara dua link. Representasi gambar dari sendi putar dan prismatik dapat dilihat pada gambar 2.2.



Gambar 2.2 Simbol Representasi Sendi Putar dan Sendi Prismatik

Meskipun ada banyak susunan yang dimungkinkan dari penggunaan sendi putar dan sendi prismatik untuk membangun *kinematic chain*, dalam prakteknya hanya beberapa jenis susunan yang digunakan. Berikut adalah beberapa jenis *kinematic chain* atau konfigurasi lengan yang paling khas digunakan pada robot manipulator berdasarkan ruang kerjanya (*workspace*) dengan keunggulan dan keterbatasannya masing-masing, diantarnya:

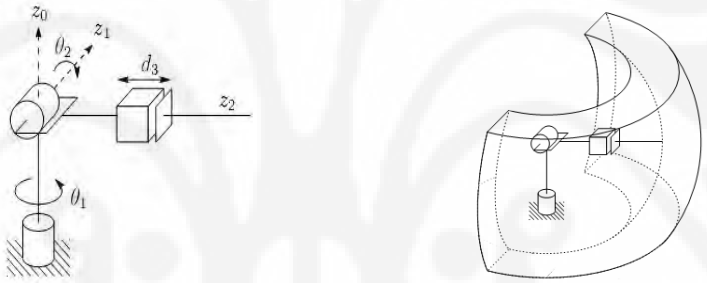
1. *Articulated manipulator* (RRR)



(a) (b)

Gambar 2.3 (a) Konfigurasi *Kinematic Chain Articulated Manipulator* (b) W*orkspace Articulated Manipulator*

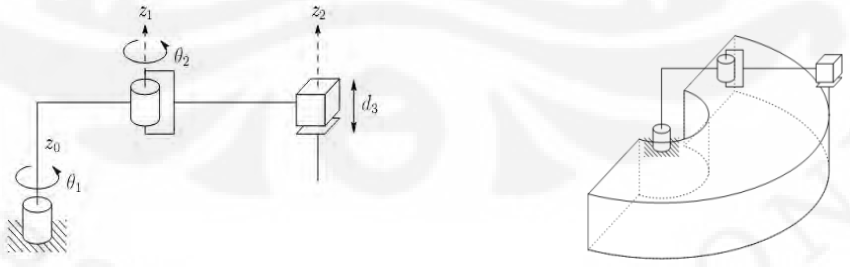
2. *Spherical Manipulator* (RRP)



(a) (b)

Gambar 2.4 (a) Konfigurasi *Kinematic-Chain Spherical Manipulator*, (b) *workspacespherical Manipulator*

3. *SCARA Manipulator* (RRP)



(a) (b)

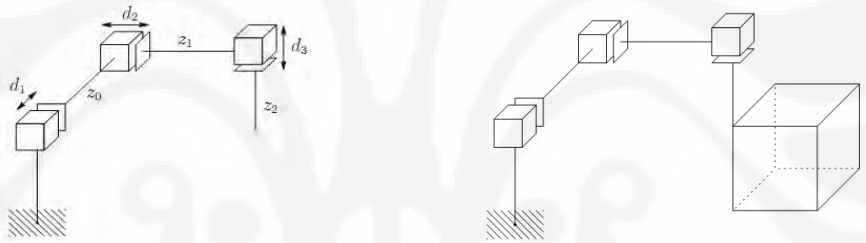
Gambar 2.5 (a) Konfigurasi *Kinematic-Chain SCARA manipulator, (b) workspace SCARA Manipulator*

4. *Cylindrical Manipulator* (RPP)

(a) (b)

Gambar 2.6 (a) Konfigurasi *Kinematic-Chain cylindrical manipulator,* (b) *workspace cylindrical Manipulator*

5. *Cartesian manipulator* (PPP)



(a) (b)

Gambar 2.7 (a) Konfigurasi *Kinematic-Chain Cartesian manipulator,* (b) *workspace Cartesian Manipulator*

2.2 Analisis Kinematika Robot Manipulator

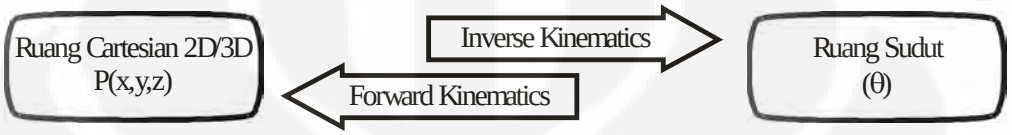
2.2.1 Kinematika vs Dinamika Robot Manipulator

Robot manipulator dapat dianalisis dalam dua kajian yaitu analisis kinematika dan dinamika. Analisis kinematika berkaitan dengan pemodelan gerakan robot tanpa memandang efek inersia/ kelembaman yang terjadi ketika robot melakukan gerakan, sedangkan analisis dinamika berhubungan dengan efek inersia dari struktur robot secara fisik hasil dari gerakan yang ditimbulkan oleh torsi aktuator ketika robot sedang melakukan pergerakan.

Kontrol dinamika tidak bisa bekerja sendirian dalam kontrol robotik tanpa bantuan kontrol kinematika. Sedangkan kontrol kinematika dapat diterapkan langsung tanpa memasukkan unsur kontrol dinamika. Pada dasarnya pengertian umum tentang kontrol robot adalah bagaimana cara mengontrol gerakan robot.

Ketika berbicara masalah kontrol gerak, maka yang lebih utama adalah bagaimana membuat kontrol kinematik yang tepat yang mampu menyelesaikan permasalahan-permasalahan posisi yang diinginkan dari pada mendahulukan kontrol dinamikanya. Sebab pemodelan dinamika biasanya sangat rumit dan jika hasilnya tidak tepat maka fungsi pemodelan kinematikanya menjadi tidak akurat dan bahkan tidak berguna dalam memperbaiki kualitas kontrol keseluruhan.

2.2.2 Kontrol Kinematika Robot



Gambar 2.8 Model kontrol kinematika robot

Kontroler dinyatakan sebagai kontroler kinematika karena mengandung komponen transformasi ruang cartesian ke ruang sudut sendi maupun sebaliknya.

1. *Forward Kinematics*

   Forward Kinematics adalah metode untuk menentukan orientasi dan posisi *end-effector* dari besarnya sudut sendi dan panjang link lengan robot. Persamaan *forward kinematics* didapatkan berdasarkan jumlah *DOF* dan jenis *kinematic chain* robot manipulator.

2. *Inverse Kinematics*

   Inverse kinematics merupakan kebalikan dari *forwards kinematics*. Metode ini diperlukan untuk mengetahui nilai sudut pada sendi-sendi yang diperlukan agar *end-effector* dapat mencapai posisi yang dikehendaki.

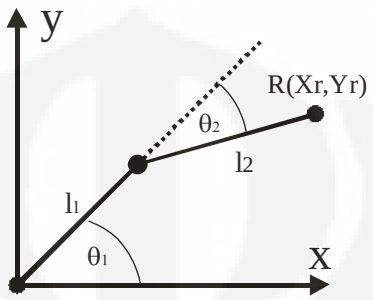
Inverse Kinematics lebih banyak diaplikasikan, namun penghitungannya jauh lebih rumit dikarenakan beberapa hal diantaranya :

1. Melibatkan persamaan non-linier
2. Solusi yang dihasilkan bisa banyak, dan kadang tak hingga
3. Kemungkinan tidak mendapatkan solusi pun ada jika posisi berada di luar *workspace* atau di luar *configuration space.*

Ada beberapa metode untuk memecahkan masalah *Inverse Kinematics* diatas diantaranya adalah dengan menggunakan persamaan trigonometri.

2.2.3 Kinematika Robot Manipulator 3 Sendi Rotasi (Robot 3-R) Menggunakan Persamaan Trigonometri

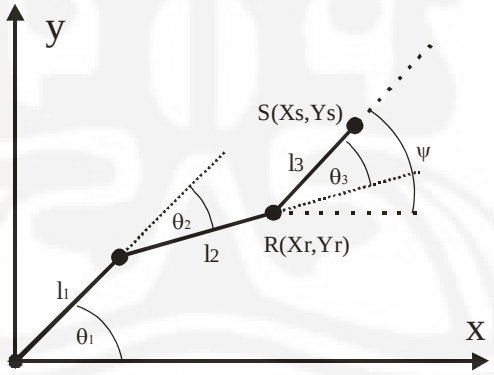
Jika terdapat manipulator dengan dua lengan dan dihubungkan dengan sendi putar seperti pada gambar berikut,



Gambar 2.9 Manipulator dua sendi

maka letak *end-effector* yaitu posisi x_r,y_r dapat dihitung dengan mengetahui nilai θ_1 dan θ_2 serta panjang lengan 1 (l_1) dan lengan 2 (l_2) melalui persamaan berikut:

$$x_r = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y_r = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \quad (2.1)$$



Gambar 2.10 Manipulator tiga sendi

Untuk robot manipulator 3 lengan seperti pada gambar 2.10, letak *end-effector* dapat dihitung dengan memanfaatkan hasil perhitungan pada *forward kinematics* robot tangan planar dua sendi dengan persamaan :

$$x_s = x_r + l_3 \cos\psi$$
$$y_s = y_r + l_3 \sin\psi \quad (2.2)$$

dimana Ψ adalah sudut hadap lengan 3 terhadap sumbu – x dengan hubungan $\Psi = \theta_1 + \theta_2 + \theta_3$.

Pada kinematika invers, jika (X_s, Y_s) dan ψ diketahui maka X_r dan Y_r didapatkan. Setelah itu barulah dapat dicari nilai dan dengan menggunakan persamaan :

$$\theta_2 = \arccos \frac{(x_r^2 + y_r^2 - l_1^2 - l_2^2)}{(2\, l_1 l_2)} \quad (2.3)$$

$$\theta_1 = \arctan \frac{y_r(l_1 + l_2\cos\theta_2) - x_r l_2 \sin\theta_2}{x_r(l_1 + l_2\cos\theta_2) + y_r l_2 \sin\theta_2} \quad (2.4)$$

Dari nilai dan yang didapat, maka dapat ditentukan besarnya .

2.3 Sistem Kendali Lintasan Gerak Robot Manipulator

Forward dan *inverse kinematics* merupakan **s**olusi dari kontrol kinematika robot manipulator, yang hanya memperhatikan aspek geometri dari robot. Solusi ini tidak memperhatikan batasan-batasan lain yang diberlakukan oleh *workspace* di mana robot beroperasi. Secara khusus, *forward* dan *inverse kinematics* tidak memperhitungkan kemungkinan akan tabrakan antara robot manipulator dengan benda di sekitar *workspace.* Oleh karena itu dibutuhkan sebuah sistem perencanaan lintasan gerak robot (*path planning*), sehingga robot akan mampu bergerak pada titik-titik lintasan (path points) yang meliputi titik awal, titik-titik singgah hingga titik tujuan.

Ada berbagai cara dalam melakukan *path planning*, dimana semuanya memiliki tujuan yang sama yaitu memberikan serangkaian titik yang disebut *via points* pada sepanjang lintasan. Cara yang paling sederhana dalam melakukan *path planning* adalah dengan memberikan serangkaian urutan posisi *end-effector*. Pada cara ini *inverse kinematics* dibutuhkan dalam mengubah posisi *end-effector* menjadi konfigurasi tiap sendi.

Agar robot dapat bergerak dengan tepat ke setiap *path points* hasil *path planning* tadi, maka diperlukan perencanaan gerak. Perencanaan gerak ini dilakukan dengan membuat fungsi posisi berdasarkan waktu. Fungsi waktu tersebut biasa dikenal dengan sebutan *trajectory.* Karena *trajectory* merupakan fungsi waktu, maka dapat dihitung juga besar kecepatan dan percepatan robot sepanjang jalur lintasan. Perencanaan *trajectory* atau *trajectory planning* dapat dilakukan baik dengan *joint space* maupun *cartesian space.*

2.3.1 *Path Planning* dengan Interpolasi linier

Untuk mengikuti jalur lintasan yang telah ditentukan, maka yang terlebih dahulu dilakukan adalah dengan menentukan titik-titik singgah (*via points*) pada lintasan atau biasa disebut *path planning*. Ada banyak algoritma path planning, salah satunya adalah dengan menggunakan interpolasi linier.

Persamaan umum untuk interpolasi linier adalah sebagai berikut,

$$P(t) = P_0 + \frac{f(P)}{k}.t \tag{2.5}$$

Dimana :

f(P) = fungsi lintasan

P_o = koordinat awal

k = konstanta (waktu untuk menempuh jarak dari koordinat awal hingga akhir)

t = *time sampling*

Selanjutnya dari persamaan interpolasi linier tersebut diturunkan ke dalam jalur lintasan yang diinginkan. Lintasan tersebut dapat berupa garis lurus, lingkaran.

Untuk lintasan garis lurus dengan koordinat awal $P_1(x_1,y_1,z_1)$ dan koordinat akhir $P_2(x_2,y_2,z_2)$ didapat persamaan berikut,

$$P_x(t) = x_1 + \frac{x_2 - x_1}{k}.t,\ P_y(t) = y_1 + \frac{y_2 - y_1}{k}.t,\ P_z(t) = z_1 + \frac{z_2 - z_1}{k}.t \tag{2.6}$$

Sedangkan untuk lintasan berupa garis lingkaran dengan pusat koordinat di P(x,y,z) dan jari – jari r, maka didapat persamaan berikut,

$$P_x(t) = x_o + R.\cos\left(\frac{2\pi.t}{k}\right),\ P_y(t) = y_o + R.\sin\left(\frac{2\pi.t}{k}\right),\ P_z(t) = z_o \tag{2.7}$$

2.3.2 Trajectory Planning

2.3.2.1 Joint Space Trajectory Planning

Perencanaan *trajectory* dengan secara langsung menentukan konfigurasi atau sudut masing-masing sendi dari titik awal hingga titik akhir pergerakan robot disebut dengan *joint space trajectory planning*. Perencanaan pada *joint space* lebih sederhana dan cepat, dikarenakan tidak diperlukan *path planning* dan perhitungan *inverse kinematics* pada setiap perpindahan *path point.* Dengan

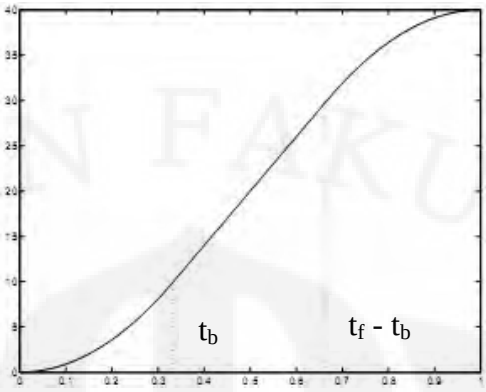
pemanfaatan perencanaan ini, maka titik akhir dapat dicapai setepat mungkin sesuai dengan acuan yang diberikan. Hal ini disebabkan kemampuannya dalam mengatasi batasan (constrain) dari sistem. Contohnya adalah jika pergerakan sendi harus dimulai dan berhenti dengan kecepatan nol. Perencanaan ini banyak diaplikasikan pada robot atau mesin dengan orientasi gerak titik ke titik (point to point motion) seperti mesin bor. Namun dalam aplikasinya tersebut biasanya ditambahkan perhitungan *inverse kinematics* diawal untuk mengkonversi koordinat titik awal dan titik akhir *end effector* menjadi konfigurasi sudut awal dan akhir sendi – sendinya. Kerugian dari *trajectory planning* ini adalah bahwa posisi *end effector* tidak secara langsung terkontrol dan tidak dapat menghindari halang rintang pada *workspace*.

J*oint space trajectory planning* dilakukan secara terpisah untuk masing-masing sendi tanpa keterkaitan antara satu sama lainnya. Perencanaan ini dilakukan dari posisi awal $q(t_o)$ hingga titik akhir $q(t_f)$ sendi. Fokus utama dari *joint space trajectory planning* adalah pada masalah penentuan q(t) sebagai konfigurasi sendi pada waktu t, dimana t berada antara waktu t_o hingga t_f dengan $t_f - t_o$ adalah waktu yang dibutuhkan untuk eksekusi *trajectory*.

Terdapat beberapa metode dalam perhitungan perencanaan ini, diantaranya adalah :

a. Linear Segments with Parabolic Blends (LSPB)

Salah satu metode *trajectory planning* dalam penentuan besarnya posisi sendi adalah Linear Segments with Parabolic Blends (LSPB). Metode ini cocok digunakan pada sistem dengan batasan berupa kecepatan yang diharuskan konstan disebagian panjang jalur lintasan. Pada LSPB, kecepatan sendi dinaikkan dari awal posisi hingga mencapai nilai yang diinginkan dan kemudian diturunkan ketika mendekati posisi tujuan. Untuk mencapai hal ini, *trajectory* yang diinginkan ditentukan dalam tiga segmen. Segmen pertama adalah dari t_o ke t_b dimana trajectory berupa fungsi kuadrat yang akan menaikkan kecepatan secara linear. Segmen kedua adalah pada saat t_b (blend time), dimana *trajectory* akan beralih ke fungsi linear sehingga dihasilkan kecepatan konstan. Segmen terakhir adalah pada waktu t_f - t_b dimana trajectory beralih sekali lagi ke fungsi kuadrat yang menurunkan kecepatan secara linear.



Gambar 2.11 Tiga segmen trajectory LSPB

Berikut adalah solusi lengkap dari *trajectory* LSPB :

$$q(t) = \begin{cases} q_0 + \frac{a}{2}t^2 & 0 \le t \le t_b \\ \frac{q_f + q_0 - Vt_f}{2} + Vt & t_b < t \le t_f - t_b \\ q_f - \frac{at_f^2}{2} + at_f t - \frac{a}{2}t^2 & t_f - t_b < t \le t_f \end{cases} \quad (2.8)$$

dimana :

$$\frac{V}{t_b} = \alpha \quad , \quad t_b = \frac{q_0 - q_f + Vt_f}{V} \quad , \quad \frac{q_f - q_0}{t_f} < V \le \frac{2(q_f - q_0)}{t_f} \quad (2.9)$$

b. Cubic Polynomial Trajectories

Metode kedua untuk trajectory path planning adalah *Cubic Polinomial Trajectory*. Metode ini digunakan jika robot harus bergerak dengan kecepatan awal dan kecepatan akhir yang ditentukan. Ada empat batasan pada metode ini, yaitu :

$$\begin{aligned} q(t_0) &= q_0 & q(t_f) &= q_f \\ \dot{q}(t_0) &= v_0 & \dot{q}(t_f) &= v_f \end{aligned}$$

dimana v_o dan v_f masing-masing adalah kecepatan awal dan akhir sendi.

Persamaan pada *cubic trajectory* merupakan fungsi polinomial dari t. Dengan adanya empat batasan yang harus diselesaikan, maka dibutuhkan funsi polinomial dengan empat koefisien bebas yang nilainya tergantung pada batasan yang harus diselesaikan. Persamaan umum *cubic poynomial* adalah :

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (2.10)$$

Untuk posisi serta kecepatan awal dan akhir join, persamaannya adalah sebagai berikut :

$$\begin{aligned} q_0 &= a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \\ v_0 &= a_1 + 2a_2 t_0 + 3a_3 t_0^2 \\ q_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ v_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{aligned} \tag{2.11}$$

Dengan mendefinisikan q_o, q_f, v_o, dan v_f, maka koefisien bebas pada *cubic polynomial* (a_0, a_1, a_2, a_3) dapat dicari dengan persamaan matriks berikut :

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} \tag{2.12}$$

2.3.2.2 Cartesian Trajectory planning

Trajectory planning yang terbaik dilakukan dengan menentukan secara pasti posisi dan orientasi *end-effectror* pada semua *path points*. Perencanaan *trajectory* seperti ini dikenal dengan *cartesian trajectory planning*. Untuk mencapai *trajectory planning* ini, maka pada setiap waktu sampling sistem harus memperoleh data sudut setiap sendi lalu melakukan konversi ke posisi dan orientasi *end-effector* menggunakan *forward kinematics*. Setelah itu dihitung nilai kesalahan *trajectory* yang terjadi, dan akan ditentukan posisi berikutnya. Posisi yang didapat tersebut akan dikonversi lagi menjadi sudut-sudut sendi dengan *inverse kinematics* untuk kemudian dieksekusi menjadi perpindahan posisi. Dengan *trajectory planning* seperti ini maka secara langsung posisi *end-effector* dapat selalu dipantau dan dikoreksi.

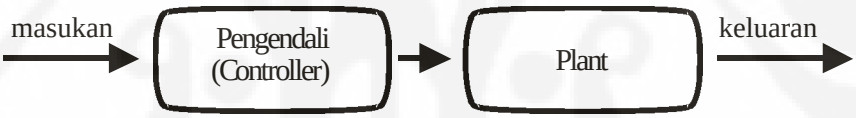
Cartesian trajectory planning baik digunakan pada robot manipulator yang dalam fungsinya mengharuskan *end-effector* untuk mengikuti jalur lintasan tertentu (comtinuos path tracking) seperti mesein pengecatan. Namun karena waktu komputasinya yang cukup lama membuat perencanaan *trajectory* ini jarang dipilih. Pada aplikasi yang sebenarnya digunakan gabungan antara *cartesian* dan *joint space trajectory planning*, dimana posisi end-effector hanya dikendalikan pada setiap titik-titik singgah *end-effector*. Sistem akan mengambil data sudut setiap sendi pada titik singgah, dan mengganti sudut acuan setiap sendi pada titik singgah tersebut dengan data yang didapat. Sudut acuan setiap sendi pada titik

singgah tersebut sebelumnya didapat dari perhitungan *path points* di awal pergerakan robot. Dengan pendekatan ini diharapkan sistem sudah bisa mengikuti lintasan yang diinginkan.

2.4 Sistem Kendali Aktuator pada Sendi Robot

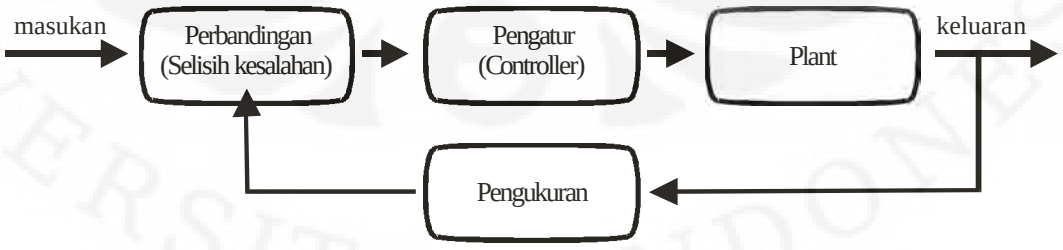
2.4.1 Sistem Kendali *Open Loop* vs *Closed Loop*

Sistem kendali dapat dikatakan sebagai hubungan antara komponen yang membentuk sebuah konfigurasi sistem, yang akan menghasilkan tanggapan sistem yang diharapkan. Pada sistem kendali dikenal sistem lup terbuka (open loop system) dan sistem lup tertutup (closed loop system). Sistem kendali *open loop* umumnya menggunakan pengendali (*controller*) serta aktuator kendali (*control actuator*) yang berguna untuk memperoleh respon sistem yang baik. Sistem kendali ini keluarannya tidak diperhitungkan ulang oleh controller. Suatu keadaan apakah plant benar-benar telah mencapai target seperti yang dikehendaki masukan atau referensi, tidak dapat mempengaruhi kinerja kontroler.



Gambar 2.12 Blok Diagram Sistem Kendali *open loop*

Berbeda dengan sistem kendali *open loop*, pada sistem kendali *closed loop* dimanfaatkan variabel yang sebanding dengan selisih respon yang terjadi terhadap respon yang diinginkan. Sistem seperti ini juga sering dikenal dengan sistem kendali umpan balik (*feedback* system).

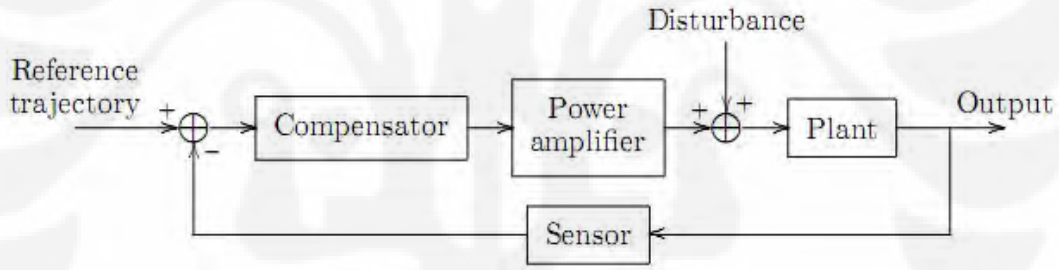


Gambar 2.13 Blok Diagram Sistem Kendali *closed loop*

2.4.2 *Independent Joint Control*

Sistem kendali posisi pada robot manipulator menitikberatkan pada penentuan rangkaian konfigurasi masing-masing sendi yang dibutuhkan untuk membuat *end-effector* melakukan gerakan tertentu. *Independent joint control* adalah metode kendali *closed loop* sederhana dimana masing-masing sendi pada manipulator dikendalikan sebagai sistem Single Input / Single Output (SISO). Pengaruh dari pergerakan sendi lainnya dianggap sebagai gangguan atau *disturbance*. Struktur dasar sistem kendali SISO ditunjukkan pada gambar 2.14.

Pemilihan kompensator P, I atau D yang tepat adalah perhatian utama dalam desain sistem kendali SISO ini. Dengan pemilihan kompensator yang tepat maka konfigurasi sendi akan sesuai dengan input yang diberikan. Tidak hanya itu, sistem juga dapat mengurangi perngaruh disturbance yang terjadi karena berbagai hal baik dari sendi itu sendiri seperti pengaruh gesekan maupun dari sendi lainnya.



Gambar 2.14 Blok Diagram *Independent Joint Control*

2.5 Aktuator pada Robot Manipulator

2.5.1 Motor DC

Motor DC Brush merupakan salah satu jenis aktuator yang paling banyak digunakan dalam industri ataupun sistem robot manipulator. Motor ini menggunakan prinsip elektromagnetika untuk menghasilkan kerja yaitu putaran. Motor DC terdiri dari rotor yang berputar dan bagian magnet sebagai stator (bagian yang diam). Arus yang datang melalui sikat / brush dan medan magnet stator akan menyebabkan rotor berputar. Bagian magnet pada stator biasa menggunakan magnet permanen. Arah arus yang datang menuju rotor berpengaruh pada arah putar motor DC.

Motor DC sering diaplikasikan terutama pada sistem robot yang memerlukan torsi cukup tinggi karena masih bisa menambahkan reduksi gear untuk meningkatkan torsi. Selain itu merubah kecepatan motor DC pun cukup mudah yaitu dengan menggunakan sinyal PWM yang akan dijelaskan pada sub-bab berikutnya. Hanya saja kendali *closed loop* posisi dan kecepatan tidak dapat langsung di diterapkan pada motor DC karena masih membutuhkan sensor posisi sebagai umpan balik nilai posisi dan kecepatan putar motor DC.

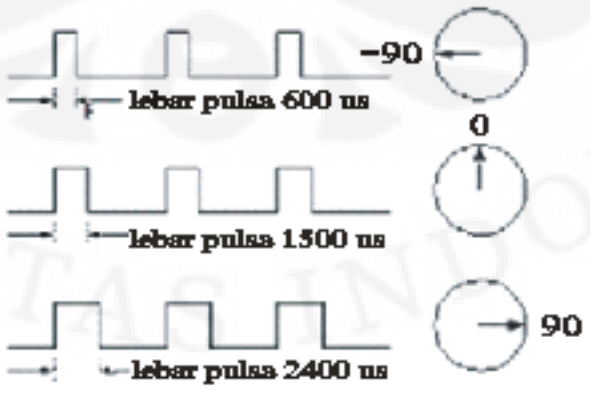
2.5.2 Motor Servo

Motor RC servo atau biasa disingkat servo saja merupakan sebuah perangkat yang merupakan gabungan dari beberapa komponen berikut :

a. Motor DC brush
b. Reduksi gear untuk meningkatkan torsi
c. Sensor posisi dan rangkaian kontrol elektrik.

Kebanyakan motor servo berputar sekitar 90-180 derajat. Namun beberapa diantaranya ada yang dapat berputar 360 derajat atau lebih. Hanya saja motor ini tidak dapat berputar terus menerus sehingga tidak dapat digunakan untuk mengendalikan roda (kecuali dimodifikasi). Kepresisian dalam pemposisiannya, membuat servo cukup ideal sebagai aktuator pada lengan dan kaki robot.

Karena servo adalah alat yang sudah cukup lengkap, kendali *closed loop* untuk kecepatan dan posisi sudut sangat mudah diimplemetasikan. Posisi sudut servo diatur dengan memberikan sinyal pulsa berperiode 20 ms dan lebar pulsa positif berkisar antara 600 hingga 2400 us.



Gambar 2.15 Variasi lebar pulsa positif terhadap sudut servo

Sebenarnya kecepatan servo tidak dapat diatur karena servo sendiri sudah memberikan spesifikasi kecepatan yang dikenal sebagai *turn rate* atau *transmit time. Turn rate* adalah waktu yang diperlukan untuk menempuh perpindahan posisi sebesar 60^o dalam keadaan tidak berbeban. Namun dengan metoda khusus yang akan dibahas pada bab berikutnya, servo dapat dibuat berjalan lebih lambat tanpa mempengaruhi torsi dari servo.

2.6 Sensor Posisi pada Robot manipulator

2.6.1 *Potensiometer*

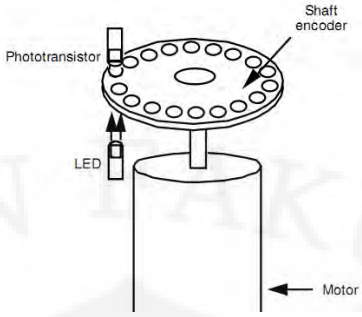
Penggunaan *potensiometer* untuk pengontrolan posisi cukup praktis karena hanya membutuhkan satu tegangan eksitasi dan biasanya tidak membutuhkan pengolah sinyal yang rumit. Namun penggunaan potensiometer sebagai sensor posisi harus melalui pengujian terlebih dahulu terkait dengan karakteristik *linearitas* dan *hysterisis*nya.

Linearitas menyatakan seberapa linier hubungan antara nilai pengukuran sensor dengan nilai yang sebenarnya di lingkungan. Sedangkan hysterisis menyatakan seberapa dekat grafik pembacaan naik dan grafik pembacaan turun sensor pada kondisi yang sama. Kondisi ideal dicapai ketika hubungan antara pembacaan sensor dan nilai sebenarnya di lingkungan membentuk grafik linear dan grafik pembacaan naik serta pembacaan turun tepat berhimpitan.

Pengujian akan menentukan seberapa linier dan seberapa besar error hysterisis potensiometer. Jika hasil uji menunjukkan ketidaklinieritasan dan adanya error hysterisis. Maka diperlukan tindakan tambahan untuk menentukan nilai fisik yang sebenarnya dari nilai pembacaan yang terukur. Setelah itu barulah potensiometer benar-benar bisa digunakan untuk proses pengukuran.

2.6.2 *Rotary Incremental Encoder*

Rotary Incremental Encoder adalah sebuah peralatan menggunakaan sebuah cakram yang berlubang dan berputar. Terdapat dua sisi bagian yaitu sisi bagian atas dan sisi bagian bawah. Sisi bagian atas terdapat emitter yang mengeluarkan sinyal dan bagian bawah sebagai detektor sinyal. Cahaya yang keluar dari emitter akan diterima oleh detektor dengan melewati cakram. Cahaya akan diterima detektor bila cakram berada pada posisi lubang.

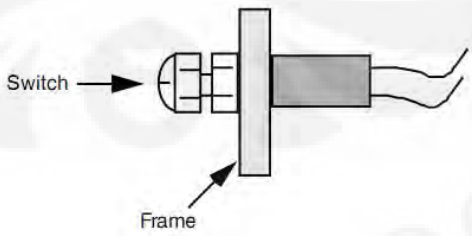


Gambar 2.16 Pemasangan encoder pada motor

Pada *incremental* encoder umumnya terdapat dua pulsa output yang berfungsi untuk menentukan orientasi gerak, posisi dan kecepatan motor. Namun ada juga encoder yang hanya memiliki satu pulsa keluaran. Encoder jenis ini tidak dapat mengetahui apakah motor sedang melaju maju atau mundur. Encoder jenis ini sangat cocok sekali dipasangkan dengan motor yang memiliki torsi cukup besar. Sehinggauntuk kendali yang mengharuskan motor mengetahui posisinya, cukup dengan mendefinisikan bahwa motor sedang maju atau mundur maka pulsa encoder bisa ditambahkan atau dikurang.

2.6.3 *Limit Switch*

Limit switch adalah salah satu jenis kontak sensor yaitu sensor yang bekerja dengan sentuhan. Pada penggunaan gripper kadang diperlukan informasi bahwa benda sudah berada dalam genggaman. Hal ini ditandai dengan beban yang akan menyentuh sensor ketika digenggam, sehingga akan mengaktifkan / menonaktifkan sensor. Sensor akan mengirimkan sinyal elektrik menuju kontroler dan kontroler akan meresponnya dengan mengirimkan perintah kepada robot untuk melakukan kerja tertentu.



Gambar 2.17 Limit Switch

2.7 Mikrokontroler Atmega 2560

Mikrokontroler tersusun dalam satu chip dimana prosesor, memori dan I/O terintegrasi menjadi satu kesatuan kontrol sistem sehingga mikrokontroler dapat

dikatakan sebagai komputer mini yang dapat bekerja secara inovatif sesuai dengan kebutuhan sistem. Dalam aplikasinya mikrokontrolerlah yang langsung berhubungan dan bertugas untuk mengendalikan robot manipulator. Namun untuk mendukung aplikasi robot industri sebagai otomasi yang fleksibel, penggunaan mikrokontroler biasanya dipasangkan dengan sebuah komputer yang dapat saling berkomunikasi dalam mengirimkan data baik untuk memprogram ulang tugas yang dijalankan oleh mikrokontroler maupun memberikan nilai pada parameter-parameter di mikrokontroler sehingga manipulator dapat dikendalikan dari komputer.

Salah satu mikrokontroler yang beredar di pasaran adalah atmega2560 keluaran ATMEL. Atmega2560 merupakan mikrokontroler 8 bit berbasis arsitektur *RISC (Reduced Instruction Set Computing)* dimana set instruksinya dikurangi baik dari segi ukurannya maupun kompleksitas mode pengalamatannya. Satu instruksi biasnya berukuran 16 bit dan sebagian besar dieksekusi dalam 1 siklus clock.

2.7.1 PIN Input Output

Atmega2560 memiliki 86 buah I/O yang masing-masing memiliki fungsi khusus yang bisa di program selain sebagai pin I/O biasa. Fungsi khusus tersebut diantaranya adalah sebagai berikut :

1. ADC (Analog Digital Converter)

   Atmega 2560 memiliki 16 pin ADC yang dapat digunakan untuk mengkonversi nilai tegangan yang berupa analaog menjadi data berupa bit-bit biner atau data digital.

   Nilai tegangan yang dapat dikonversi berkisar antara 0 volt hingga tegangan pada PIN AREF atau AVCC, dengan persamaan sebagai berikut:

$$\text{Nilai digital} = \frac{\text{nilai analog tegangan}}{\text{tegangan AVCC / AREF}} x1023 \qquad (2.13)$$

   Dimana 1023 adalah resolusi ADC pada atmega 2560 yaitu 10 bit.

2. Timer

   Atmega 2560 didukung oleh 6 timer dengan detail 2 timer 8 bit dan 4 timer 16 bit. Setiap timer tersebut memiliki pewaktuan masing-masing yang merupkan hasil prescaler dari clock utama mikrokontroler. Timer-

timer ini bisa diset menjadi mode Compare yaitu mode dimana timer bisa menghasilkan sinyal PWM dengan perioda dan lebar positif pulsa tertentu.

3. USART

   USART (Universal Serial Asynchronous Receiver Transmitter) merupakan fungsi khusus yang tidak kalah penting yang dimiliki atmega2560. Dengan fungsi ini mikrokontroler dapat berkomunikasi dengan perangkat lain seperti komputer, ponsel, bahkan dengan mikrokontroler lain melalu jalur serial.

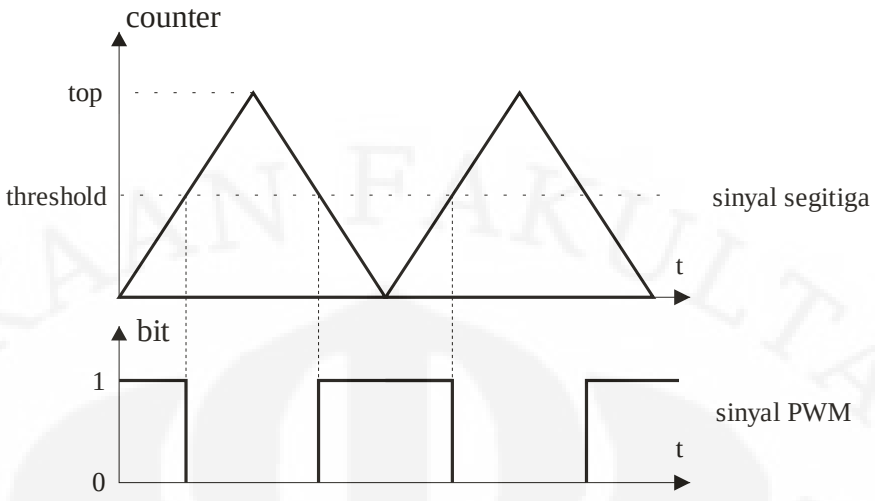
4. Interrupt

   Digunakan jika terdapat masukan yang perubahan bitnya perlu di respon sesaat itu juga. Dengan memilih salah satu mode misalnya *falling edge* pada PIN khusus ini, maka interrupt akan terjadi ketika PIN tersebut mengalami perubahan nilai bit dari 1 ke 0. Ada mode lain yang dapat membuat interrupt terjadi, yaitu mode *rising edge* yang merupakan kebalikan dari mode *falling edge* dan mode toggle yang merupakan gabungan antara *falling edge* dan *rising edge.* Sesaat setelah interrupt terjadi maka interrupt service routine yang merupakan respon pun akan segera dieksekusi.

2.7.2 Timer penghasil Sinyal PWM

Untuk menghasilkan sinyal PWM, timer pada atmega 2560 harus diset sebagai mode compare. Pada mode ini timer dapat diprogram untuk menghasilkan sinyal PWM dengan periode dan lebar pulsa positif tertentu. Ada dua mode pembentukan sinyal PWM yaitu fast PWM dan phase correct PWM.

Pada mode phase correct PWM, sinyal PWM didapatkan dari threshold sinyal segitiga pada nilai counter tertentu. Jika nilai counter berada di bawah nilai threshold, maka dihasilkan pulsa high atau bernilai 1. Dan sebaliknya, jika nilai counter berada di atas nilai threshold maka dihasilkan pulsa low atau bernilai 0. Untuk lebih jelasnya perhatikan gambar 2.18 dan langkah-langkah dalam pembuatan sinyal PWM mode phase correct PWM berikut :



Gambar 2.18 Phase PWM Generator

Langkah-langkah dalam membuat phase correct PWM :

1. Dibuat sinyal segitiga dengan periode tertentu, dimana nilai periode tersebut didapat dengan mengatur prescaler timer dan batas overflow timer (nilai top). Persamaan untuk mendapatkan periode sinyal adalah sebagai berikut :

$$T = \frac{1}{f} = \frac{prescaler * 2 * top}{clock\ mikrokontroler} \quad (2.14)$$

2. Duty Cycle PWM didapatkan dengan melakukan threshold pada nilai counter tertentu dibawah nilai top timer sesuai dengan persamaan berikut :

$$\text{Duty cycle (\%)} = \frac{\text{nilai threshold}}{\text{nilai top}} x100\% \quad (2.15)$$

Pada atmega 2560 hanya terdapat 4 timer yang dapat diatur nilai topnya. Keempat timer ini memiliki masing-masing 3 buah keluaran yang berarti terdapat 12 port pada atmega 2560 yang dapat menghasilkan sinyal PWM berbeda.

2.8 Pengembangan Software Antarmuka Berbasis .Net

Dalam aplikasi industri, robot manipulator adalah teknologi otomasi yang fleksibel dimana tidak seperti pada otomasi tetap yang merupakan sebuah mesin yang khusus hanya dapat mengangani suatu pekerjaan spesifik tertentu. Kefleksibilitasan tersebut dapat dicapai karena sebuah robot manipulator dapat diprogram ulang atau pun dapat dikendalikan secara *remote control* oleh operator melalui komputer.

Dalam melakukan program ulang ataupun kendali *remote control*, dibutuhkan sebuah program antarmuka pada komputer. Pembuatan program antarmuka tersebut dapat menggunakan banyak sekali bahasa pemrograman. Salah satunya adalah C# (baca : See-Sharp) programming language yang merupakan salah satu bahasa pemrograman baru dan menjanjikan banyak kemudahan bagi programer.

C# adalah bahasa pemrograman baru yang diciptakan Microsoft yang digunakan oleh banyak developer .NET untuk mengembangkan aplikasi dengan platform .NET. .NET sendiri merupakan suatu komponen tambahan Windows yang terintegrasi yang dibuat dengan tujuan pengembangan berbagai macam aplikasi. C# bersifat sederhana karena dibuat berdasarkan bahasa C/C++ dan sudah mendukung Object Oriented Programming.

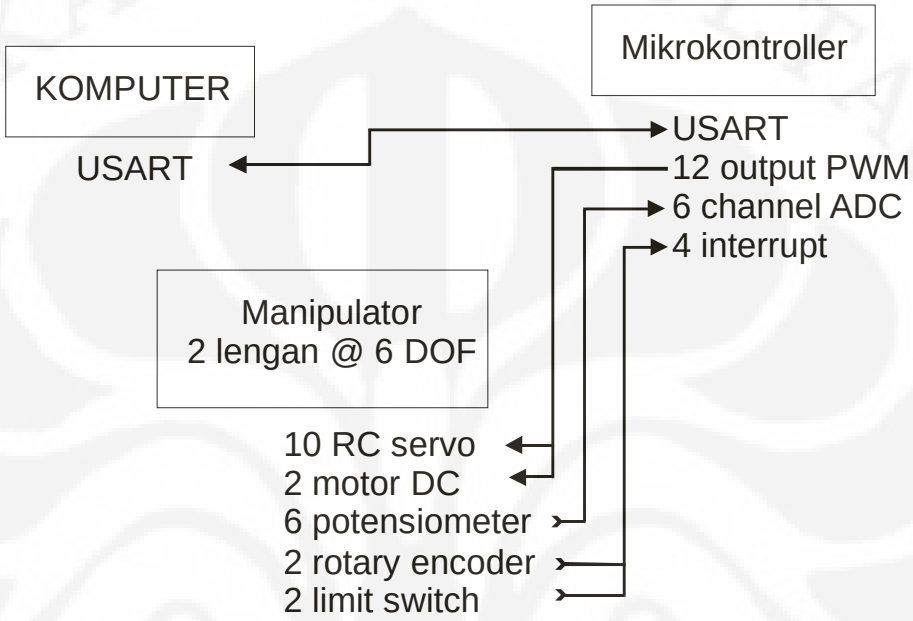
2.9 Komunikasi Komputer dengan Mikrokontroler

Sangat memungkinkan untuk sebuah komputer mengendalikan manipulator di tempat yang berbeda. Hal ini didukung dengan berbagai media dan metode pengiriman data diantara keduanya. Media yang cukup baik dalam menghantarkan informasi adalah kabel. Dan metode pengiriman data yang relatif cepat dengan error yang cukup kecil adalah melalui jalur serial.

Untuk aplikasi yang menggunakan arsitektur server-client seperti pada komunikasi antara komputer dan mikrokontroler, dibutuhkan suatu protokol yang sederhana. Error handling dibuat seminimal mungkin dan tidak diperlukan fitur kemananan sama sekali. Server dalam hal ini komputer meminta client yaitu mikrokontroler menjalankan prosedur yang sudah didefinisikan sebelumnya di dalam tabel fungsi yang ada di client. Client menjalankan prosedur lalu mengembalikan hasilnya ke client.

BAB 3
PERANCANGAN SISTEM

3.1 Perancangan Sistem



Gambar 3.1 Rancangan sistem

Robot manipulator ini terdiri dari 2 buah lengan masing-masing 6 DOF. Sendi pertama merupakan sendi translasi, tiga sendi berikutnya merupakan sendi rotasi, dan dua sendi terakhir merupakan *end-effector* yang terdiri dari wrist dan gripper. Dari keenam sendi tersebut hanya sendi translasi yang menggunakan motor DC, sisanya menggunakan motor RC servo. Setiap servo dilengkapi dengan potensiometer untuk feedback kontrol posisi sendi. Sedangkan pada motor DC digunakan rotary encoder sebagai sensor feedbacknya ditambah dengan limit switch yang digunakan untuk menandai lengan di posisi z = 0.

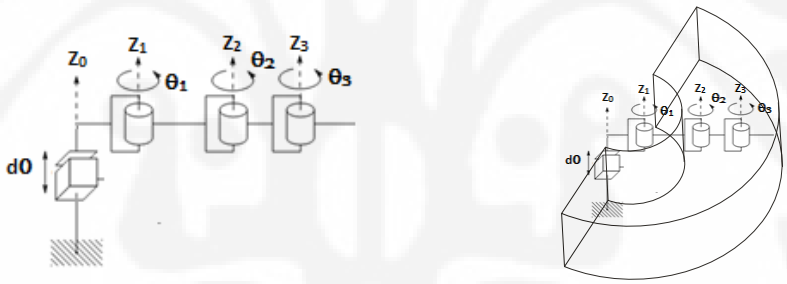
Pada mikrokontroler digunakan 12 output PWM untuk mengendalikan posisi dan kecepatan 10 motor servo dan 2 motor DC pada manipulator. ADC dimanfaatkan untuk membaca posisi 3 buah sendi rotasi pertama di masing-masing lengan. 2 interrupt digunakan untuk menghitung putaran rotary encoder dan mengetahui tertekannya limit switch di setiap lengan.

USART pada mikrokontroler terhubung dengan port serial pada komputer melalui RS-232 serial interface.

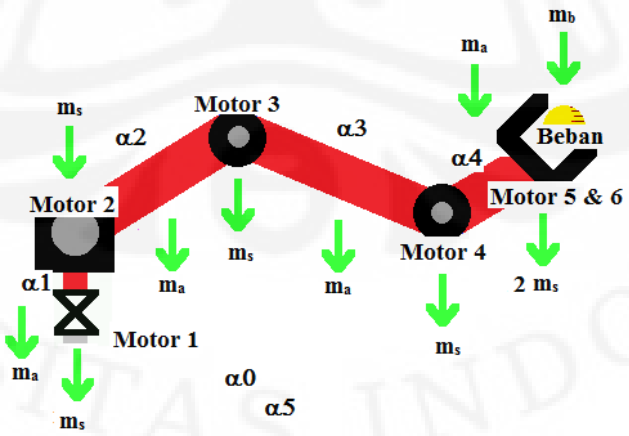
3.2 Perancangan Mekanik

Robot manipulator ini dirancang untuk memiliki kemiripan dengan dua lengan manusia. Workspace dari *kinematic chain* yang paling sesuai dengan rancangan ini adalah *articulated manipulator*. Namun demi struktur yang lebih kokoh untuk tugas mengangkat beban dan perhitungan kinematika yang lebih sederhana serta kemiripannya dengan badan manusia. Maka digunakan konfigurasi Kinematic Chain modifikasi sehingga didapat konfigurasi baru dengan workspace berupa *articulated manipulator* dan struktur yang lebih kokoh.

Konfigurasi *kinematic-chain* yang dipilih untuk dua buah lengan robot manipulator ini adalah serupa dengan *SCARA* yang dibalik. Dimana sendi translasinya menjadi dasar bagi pundak robot manipulatornya dan sendi-sendi rotasinya menjadi bahu, siku dan pergelangan tangan robotnya. Kinematic Chain serta workspace dari rancangan manipulator kedua lengan robot ini adalah sebagai berikut :



Gambar 3.2 (a) Konfigurasi *Kinematic-Chain* sebuah lengan robot manipulator, *(b)workspace* sebuah lengan robot manipulator



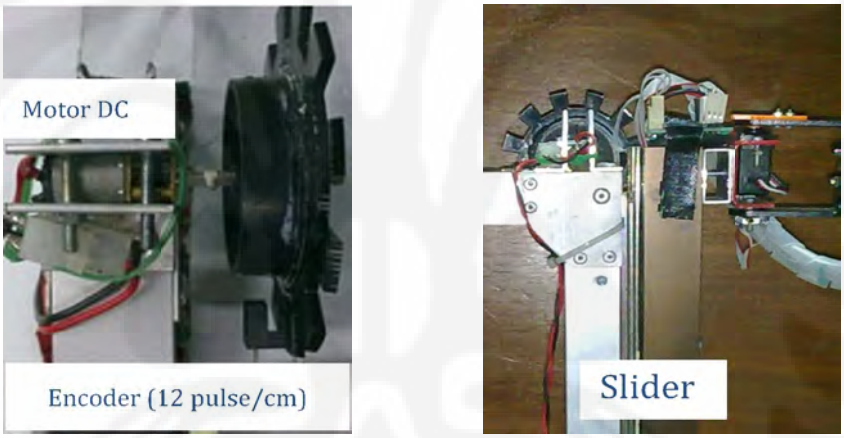
Gambar 3.3 Konfigurasi model lengan Robot Manipulator

Tabel 3.1 Konfigurasi lengan manipulator

Nomor Link	Panjang (cm)
α_2	8.5
α_3	6
α_4	7.5
d_1	24

3.2.1 Sendi Translasi

Pada sendi ini digunakan sebuah motor DC sebagai aktuator dan katrol untuk merubah gerak rotasi motor DC menjadi gerak translasi lengan. Motor DC yang digunakan harus memiliki torsi yang cukup tinggi sehingga tetap dapat menahan melawan momen gaya lengan pada posisi tertingginya.



Gambar 3.4 Desain Sendi Translasi

Pada poros motor DC diletakkan rotary encoder untuk sensor posisi ketinggian lengan. Rotary encoder yang digunakan merupakan incremental encoder, dengan resolusi sebesar 12 pulsa untuk perubahan posisi tinggi lengan sebesar 1 cm. Kemudian tepat di dasar sumbu translasi lengan diletakkan limit switch sebagai penanda bahwa posisi lengan berada pada ketinggian terendah.

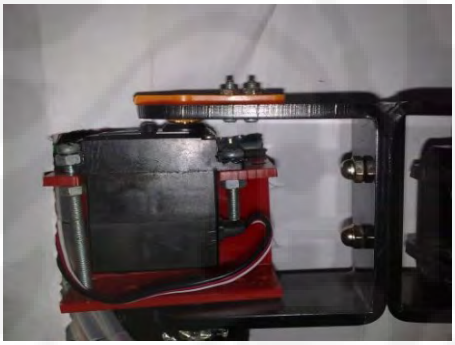
3.2.2 Lengan robot 3 DOF

Lengan robot merupakan lengan planar 3 DOF dengan masing-masing DOF merupakan sendi rotasi yang bergerak horizontal dan dihubungkan secara serial. Sendi pertama merupakan sendi rotasi bahu, sendi kedua merupakan sendi

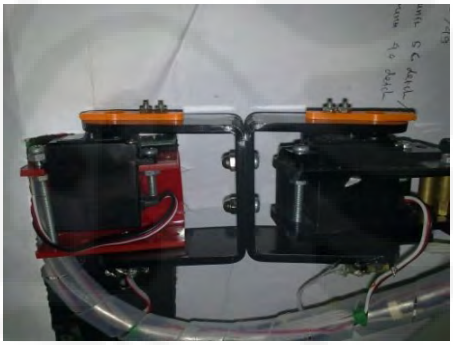
rotasi siku, dan sendi ketiga merupakan sendi rotasi pergelangan horizontal lengan.

Pada sendi ini digunakan motor servo sebagai aktuator. Setiap motor servo dibuatkan dudukan yang digunakan untuk menahan dudukan dan menempelkan servo dengan *link* yang terbuat dari bahan acrylic. Pada servo bagian bawah ditambahkan poros yang sejajar dengan poros servo. Poros tersebut bermanfaat untuk membagi beban yang diterima sehingga poros servo tidak akan patah. Tepat di bawah poros pada dudukan servo diletakkan potensiometer sebagai sensor posisi.

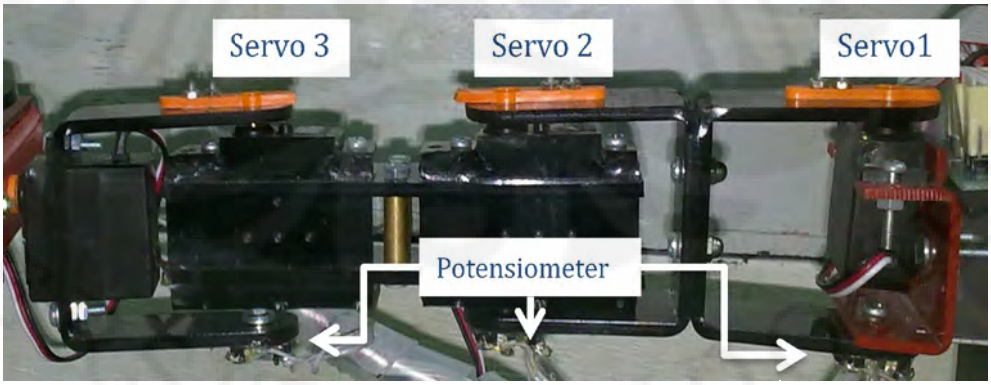
Untuk lengan penghubung sendi (*links*) α_2, α_3, dan α_4, digunakan bahan acrylic yang cukup ringan dan tebal. Lengan dibuat paralel atas dan bawah agar kokoh dan di pasang pada poros servo serta poros bawah dudukan servo sehingga posisi lengan lebih mendatar.



(a)



(b)



(c)

Gambar 3.5 (a) Dudukan servo (b) desain lengan penghubung
(c) penempatan potensiometer pada lengan

3.2.3 *Wrist* dan *Gripper*

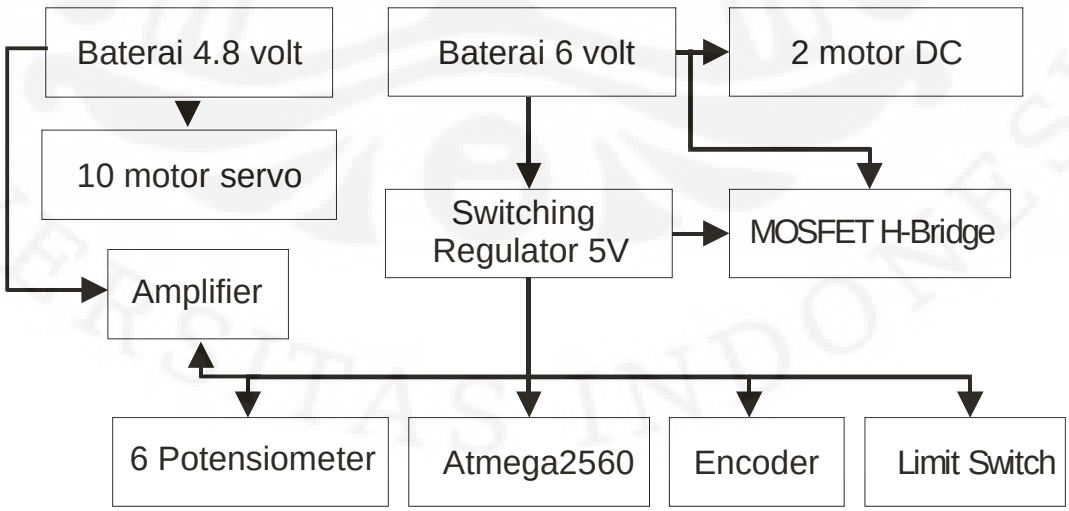
(a) (b)

Gambar 3.6 (a) Desain wrist,
(b) Desain gripper

Wrist atau pergelangan tangan pada manipulator ini merupakan sendi rotasi dengan arah gerak vertikal. Sedangkan *end-effector*nya merupakan gripper yang bergerak translasi membuka dan menutup. Permukaan *gripper* yang akan bersentuhan dengan objek didesain sekasar mungkin sehingga menambah gaya gesek yang diperlukan untuk menggenggam objek.

3.3 Perancangan Sistem Elektrik

3.3.1 Distribusi Daya



Gambar 3.7 Distribusi Daya Sistem

Sistem elektrik robot manipulator ini menggunakan 2 buah batre sebagai sumber dayanya. Masing-masing batre memiliki rating tegangan sebesar 4.8 volt dan 6 volt. Batre 4.8 volt menggunakan jenis NiMH dan dipakai sebagai catu daya servo. Batre 6 volt menggunakan jenis SLA (aki kering), dipakai sebagai catu daya motor DC dan di regulasi menjadi tegangan sebesar 5 volt untuk catu daya mikrokontroler beserta sensor dan sinyal kendali motor.

Penggunaan dua buah batre ini dikarenakan masukan tegangan servo tidak dianjurkan sebagai keluaran dari regulator. Selain boros, rating arus regulasi yang dibutuhkan harus cukup tinggi akibat supply arus yang dibutuhkan servo tidak dapat di prediksi. Jika supply arus dari batre kurang, maka torsi dan kecepatan servo akan berkurang. Selain itu posisi servo pun tidak akan stabil dan terkadang bergetar. Sedangkan di pasaran cukup sulit mencari IC regulator tegangan dengan rating arus yang tinggi.

Fungsi serta perancangan rangkaian elektrik untuk *switching regulator* 5V, *rotary encoder, limit switch,* potensiometer, *H-Bridge* dan amplifier servo seperti ditunjukkan pada gambar 3.9 akan dijelaskan lebih mendetail pada Lampiran B.

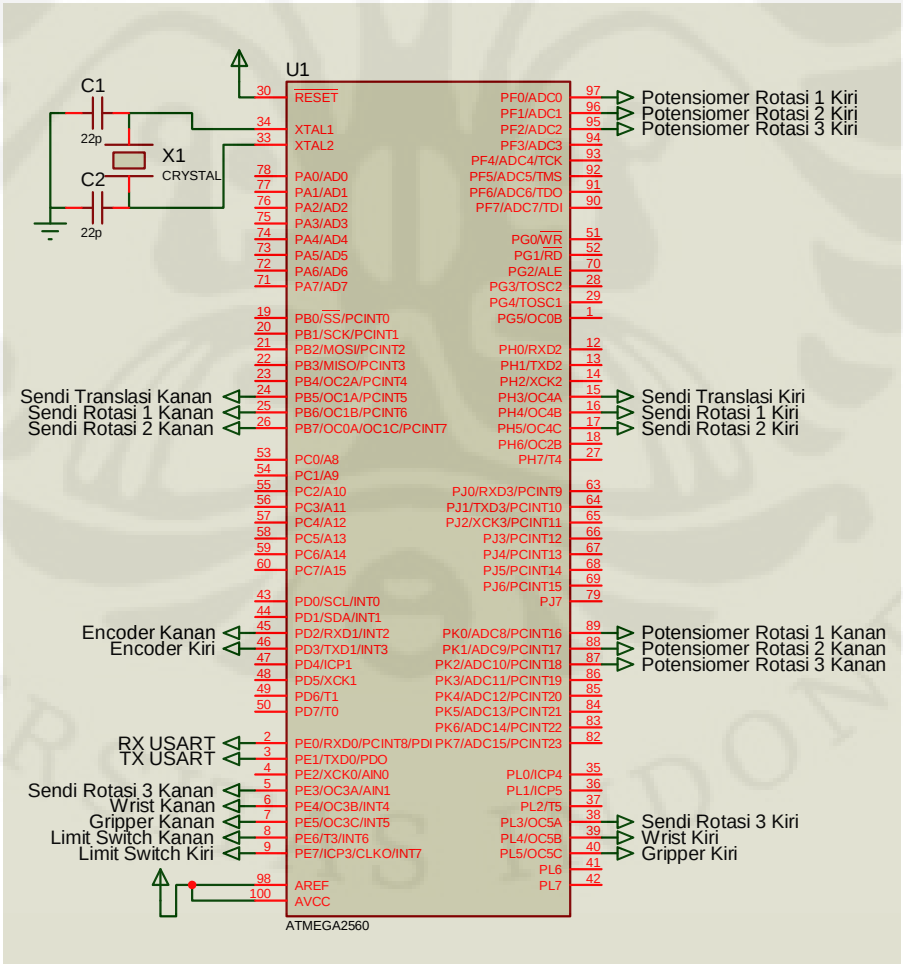
3.3.2 Mikrokontroler atmega2560

Tabel 3.2 Konfigurasi PIN di Atmega2560 pada Sistem

Atmega 2560	
PIN E.3 - sendi rotasi 3 kanan	PIN K.0 Potensiometer sendi rotasi 1 Kanan
PIN E.4 - wrist kanan	PIN K.1 Potensiometer sendi rotasi 2 Kanan
PIN E.5 - gripper kanan	PIN K.2 Potensiometer sendi rotasi 3 Kanan
PIN B.5 - sendi translasi kanan	PIN F.0 Potensiometer sendi rotasi 1 Kiri
PIN B.6 - sendi rotasi 1 kanan	PIN F.1 Potensiometer sendi rotasi 2 Kiri
PIN B.7 - sendi rotasi 2 kanan	PIN F.2 Potensiometer sendi rotasi 3 Kiri
PIN L.3 - sendi rotasi 3 kiri	
PIN L.4 - wirst kiri	PIN D.2 Encoder Kanan
PIN L.5 - gripper kiri	PIN D.3 Encoder Kiri
PIN H.3 - sendi translasi kiri	PIN E.6 Limit switch kanan
PIN H.4 - sendi rotasi 1 kiri	PIN E.7 Limit switch kiri
PIN H.5 - sendi rotasi 2 kiri	PIN A.0 Enable sendi translasi kanan
	PIN A.1 Enable sendi translasi kiri
PIN E.0 RX USART	
PIN E.1 TX USART	

Berikut adalah konfigurasi pin I/O yang digunakan pada sistem robot manipulator ini :

a. PIN E.3, E4, E5, B.5, B.6, B.7, L.3, L.4, L.5, H.3, H.4 dan H.5 merupakan PIN I/O khusus yang masing-masingnya dapat memberikan sinyal PWM dengan duty cycle yang berbeda.
b. PIN F3, F4, F.5, K.3, K.4, K.5 merupakan PIN ADC pada atmega2560. ADC dibutuhkan untuk merubah nilai tegangan keluaran potensiometer menjadi nilai digital.
c. PIN E.5, E.6, D.2, dan D.3 adalah PIN interrupt. Digunakan karena respon akibat perubahan nilai bit pada sensor yang terpasang pada PIN-PIN ini perlu secepat mungkin.
d. PIN E.0 dan PINE.1 dibutuhkan untuk komunikasi serial antara mikrokontroler dengan komputer.



Gambar 3.8 Konfigurasi PIN Atmega2560 pada Sistem

3.4 Perancangan Perangkat Lunak

Perancangan perangkat lunak pada sistem ini dibagi kedalam dua bagian yaitu program pada PC dan program pada mikrokontroler. PC mempunyai peran dalam melakukan komputasi kinematika, *trajectory planning* manipulator. Hal ini didasarkan pada alasan perhitungan yang akan cukup rumit dan lama jika dilakukan oleh mikrokontroler. Komputasi oleh PC tadi akan menghasilkan data berupa acuan sudut tiap servo dan posisi tiap motor DC. Data ini akan dikirim ke mikrokontroler untuk langsung dieksekusi menjadi sinyal kendali yang menggerakkan servo dan motor DC ke sudut dan posisi acuan tadi. Hasil pergerakkan itu akan direkam oleh mikrokontroler dan dikirimkan ke komputer untuk di tampilkan dan dibandingkan.

3.4.1 Program pada PC

Pada PC dibuat sebuah GUI berbasis .Net menggunakan bahasa pemrograman C# untuk melakukan komputasi dan simulasi gerak manipulator menggunakan *forward kinematics*, *inverse kinematics* dan *trajectory planning.* Hasil dari komputasi akan berupa titik-titik posisi setiap sendi yang siap di sajikan dalam bentuk gambar. Selain titik-titik tersebut di hasilkan juga acuan sudut tiap servo dan posisi tiap motor DC untuk dikirimkan ke mikrokontroler.

3.4.1.1 Program *Forward Kinematics*

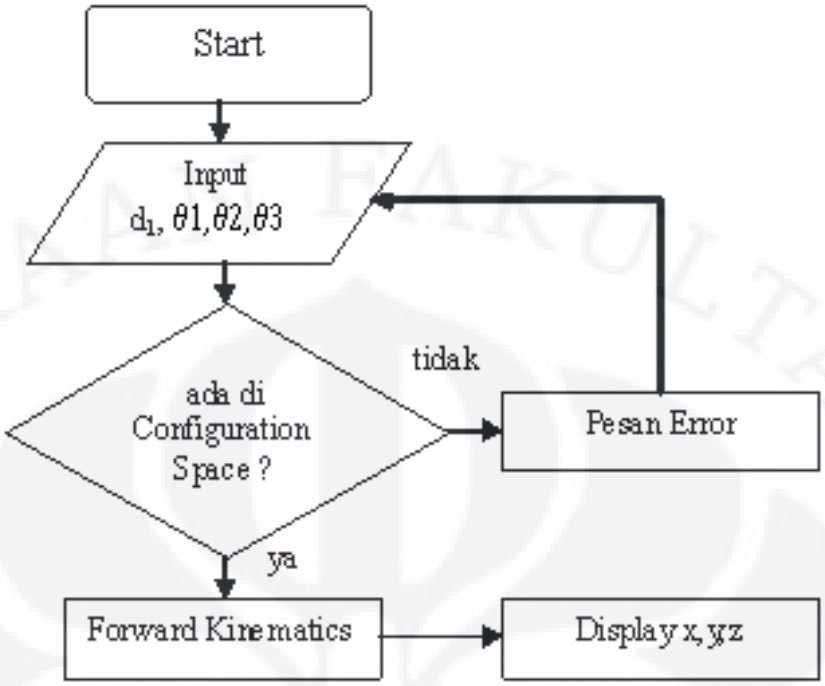
Perhitungan *forward kinematics* pada robot manipulator ini memenuhi persamaan berikut **:**

$$x, y, z = f(d_1, \theta_1, \theta_2, \theta_3) \tag{3.1}$$

dimana :

$$\begin{aligned} z &= d_1 \\ x &= l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ y &= l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \end{aligned} \tag{3.2}$$

dengan d_1 merupakan tinggi lengan dari dasar, dan $\theta_1, \theta_2, \theta_3$ merupakan besar sudut tiap lengan seperti yang ditunjukkan pada gambar 2.10.



Gambar 3.9 Flowchart Program *Forward Kinematics*

Program dibuat dengan masukan berupa nilai d_1, $\theta_1, \theta_2, \theta_3$, dan keluaran berupa posisi *end-effector* robot pada bidang cartesian (x,y,z). Nilai masukan d_1, $\theta_1, \theta_2, \theta_3$, dibatasi sesuai dengan *configuration space* tiap sendi. Jika nilai masukan melebihi atau lebih kecil dari *configuration space* yang bisa diraih sendi, maka program akan menghasilkan pesan error.

Perhitungan *forward kinematics* pada sistem ini berfungsi untuk menganalisis gerakan aktual yang terjadi pada manipulator dan melihat seberapa tepat sistem kendali gerak mencapai titik acuan.

3.4.1.2 Program *Inverse Kinematics*

Perhitungan *inverse kinematics* pada robot manipulator ini dilakukan dengan menggunakan persamaan trigonometri sederhana yang memenuhi persamaan 2.18**.**

$$d_1, \theta_1, \theta_2, \theta_3 = f(x, y, z, \alpha, \psi) \tag{3.3}$$

dimana :

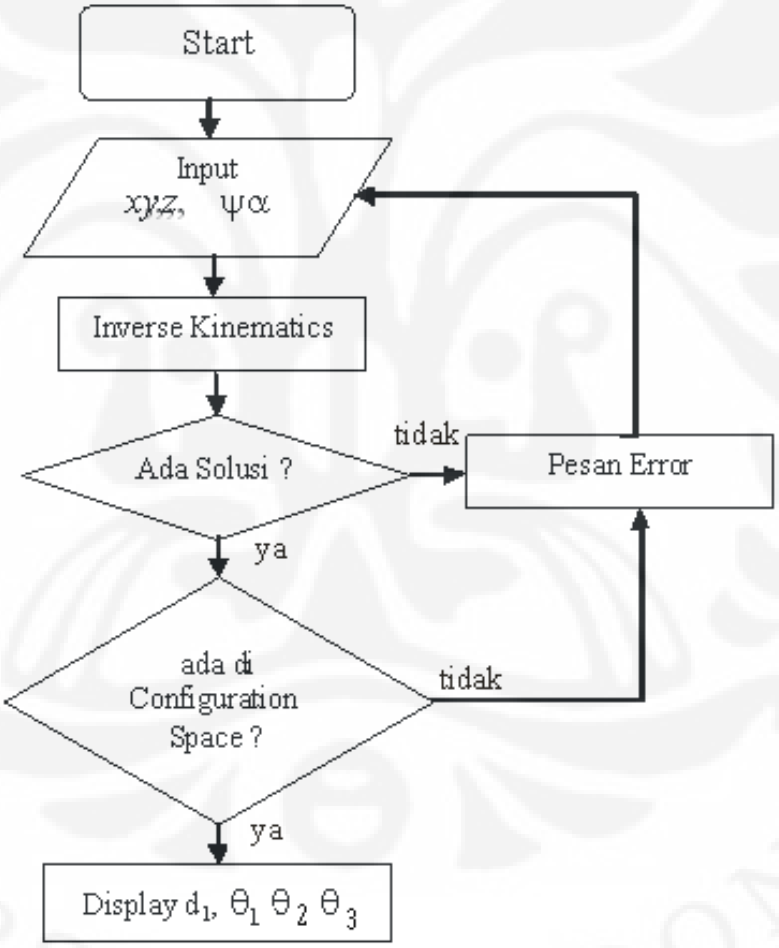
$$d_1 = z$$
$$x_r = x - l_3 \cos\psi$$
$$y_r = y - l_3 \sin\psi \tag{3.4}$$

$$\theta_2 = \alpha * \left| \arccos \frac{(x_r^2 + y_r^2 - l_1^2 - l_2^2)}{(2\, l_1 l_2\ )} \right| \tag{3.5}$$

$$\theta_1 = \arctan \frac{y_r(l_1 + l_2\cos\theta_2) - x_r l_2 \sin\theta_2}{x_r(l_1 + l_2\cos\theta_2) + y_r l_2 \sin\theta_2}$$

$$\theta_3 = \psi - \theta_1 - \theta_2 \tag{3.6}$$

dengan l_1, l_2, dan l_3 didapat dari konfigurasi lengan sistem yang ditunjukkan oleh tabel 3.1. ψ merupakan sudut hadap *end-effector* terhadap sumbu – x seperti yang ditunjukkan pada gambar 2.10 dan α adalah tanda sendi 2 apakah positif atau negatif (+1 atau -1).



Gambar 3.10 Flowchart Program *Inverse Kinematics*

Penambahan parameter ψ dibutuhkan karena solusi pada *inverse kinematics* bisa sangat banyak jika hanya diketahui parameter x,y dan z-nya saja. Perlu parameter lain berupa orientasi arah *end-effector* sehingga solusinya menjadi tinggal 2 saja. Dua solusi tersebut adalah pada sudut θ_2 yang bisa positif atau negatif. Sehingga diperlukan parameter tambahan yaitu α yang membuat solusi dari *inverse kinematics* pada lengan manipulator ini menjadi hanya tinggal satu atau tidak ada solusi sama sekali.

Perhitungan diatas belum memperhatikan *configuration space* servo pada setiap sendi. Sehingga solusi yang dihasilkan belum tentu bisa diraih oleh manipulator. Oleh karena itu diperlukan algoritma tambahan berupa perbandingan nilai yang didapat dengan *configuration space* setiap sendi.

Program dibuat dengan masukan berupa nilai x, y, z, α, ψ dan keluaran berupa nilai $d_1, \theta_1, \theta_2, \theta_3$. Jika solusi *inverse kinematics* tidak ditemukan maka program akan memberikan pesan error.

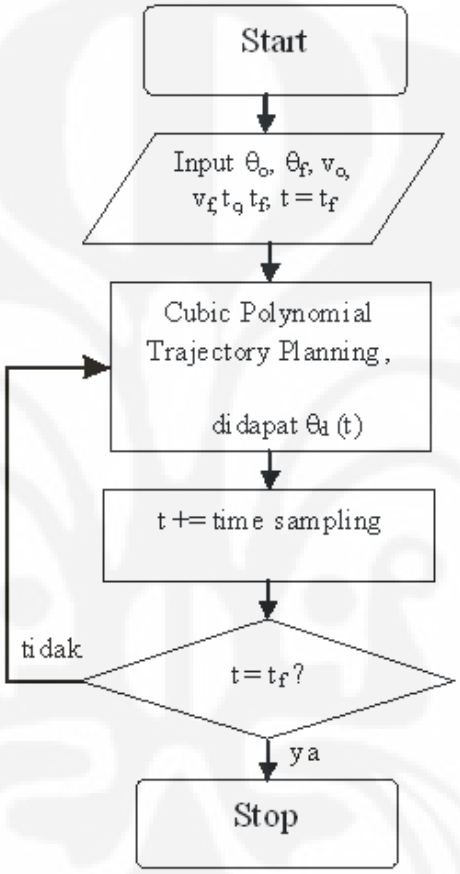
3.4.1.3 Program *Joint Space Trajectory Planning*

Seperti yang telah dijelaskan pada bab II, *joint trajectory planning* diperlukan agar *end-effector* dapat dengan tepat berpindah dari satu titik ke titik lain (*point to point motion*). Pada sistem manipulator ini banyak digunakan motor servo yang telah memiliki kendali posisi di dalamnya. Oleh karena itu pemilihan metode pada *joint space trajectory planning* bisa dipilih dengan bebas.

Pada sistem ini dipilih metode *Cubic polynomial trajectory*. Pemilihan ini didasarkan kepada asumsi bahwa perhitungan matriks untuk mendapatkan nilai koefisien bebas a_o .. a_3 cukup cepat. Selain itu batasan berupa kecepatan awal dan akhir sendi merupakan variabel yang bisa dicari dalam metode ini. Hal ini akan sangat penting ketika *cartesian space trajectory* yang merupakan gabungan *cubic polynomial trajectory* antar *via points* akan diterapkan pada sistem.

Program dibuat dengan terlebih dahulu mendefinisikan konfigurasi masing-masing sendi di titik awal dan di titik akhir (misal untuk sendi i, maka posisi awalnya = q_{io} dan posisi akhirnya = q_{if}). Konfigurasi tiap sendi inilah yang akan menjadi masukan bagi fungsi trajectory masing-masing sendi itu sendiri.

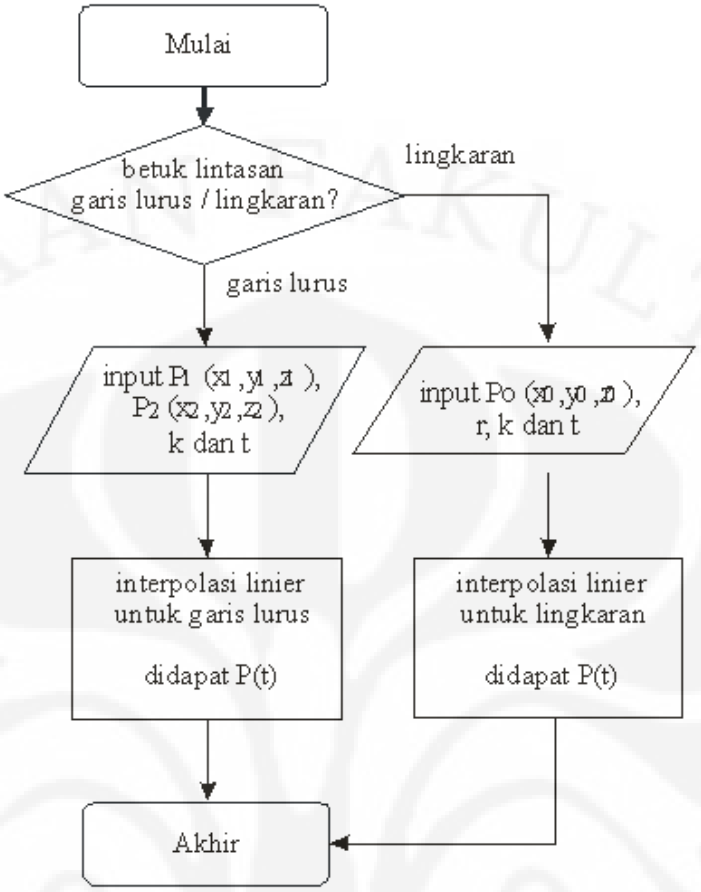
Setelah konfigurasi awal dan akhir diketahui, selanjutnya adalah pendefinisian t_f atau waktu eksekusi, v_o atau kecepatan awal dan v_f atau kecepatan akhir. Lalu dicari fungsi waktu *trajectory* dari konfigurasi masing-masing sendi $q_i(t)$ sesuai dengan persamaan 2.10. Nilai $q_i(t)$ inilah yang akan dieksekusi oleh sendi robot manipulator.



Gambar 3.11 Flowchart Program *Cubic Polynomial Trajectory Planning*

3.4.1.4 Program Path Planning dengan Interpolasi Linier

Lintasan yang dilalui robot pada sistem ini dibatasi hanya berupa garis lurus atau lingkaran. Berikut adalah tahapan program untuk *path planning* dengan menggunakan interpolasi linier sesuai dengan persamaan 2.5.



Gambar 3.12 Flowchart Program *Path Planning* dengan interpolasi linier

3.4.2 Program pada mikrokontroler

Peran penting mikrokontroler adalah mengendalikan tiap servo dan motor DC sesuai dengan acuan sudut dan posisi yang dikirimkan oleh PC. Dalam melaksanakan perannya ini mikrokontroler harus bisa melakukan fungsi-fungsi berikut :

a. menerima data dari komputer dengan cepat dan efisien
b. melakukan *independent joint control* pada setiap sendi
c. Mengirimkan data pergerakan tiap sendi ke komputer.

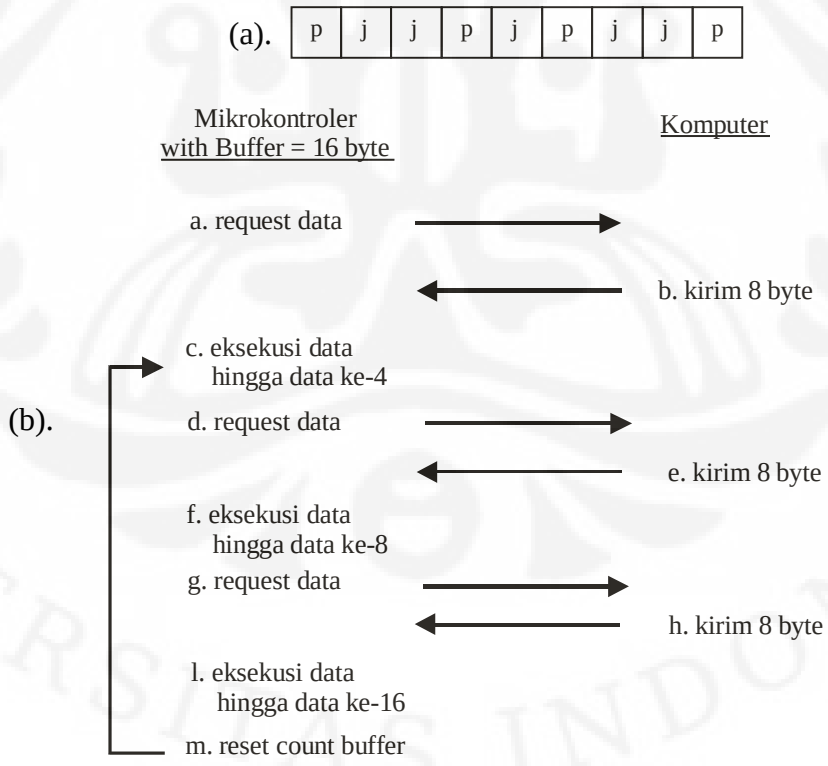
3.4.2.1 Program komunikasi serial

Komunikasi dilakukan dengan pengiriman byte-byte data oleh komputer yang berisikan prosedur beserta parameter-parameter yang sudah didefinisikan sebelumnya di dalam tabel fungsi yang ada di mikrokontroler. Komputer akan mengirimkan paket data dengan panjang yang selalu sama seperti terlihat pada gambar 3.13 (a), jika mikrokontroler melakukan permintaan. Paket data yang

masuk mikrokontroler akan langsung disimpan terlebih dahulu ke dalam sebuah array untuk digunakan saat dibutuhkan. Setelah sejumlah data pada paket digunakan, mikrokontroler akan melakukan permintaan data yang kemudian akan disimpannya lagi ke dalam array yang sama. Komunikasi semacam ini membutuhkan tingkat keberhasilan pengiriman data yang tinggi karena tidak adanya acknowledgment.

Keuntungan dari komunikasi ini adalah pada *traffic* data keluar masuk yang lebih sedikit karena tidak adanya *acknowledgment*. Selain itu data yang dibutuhkan mikrokontroler akan selalu tersedia. Komunikasi sepertu ini sangat cocok digunakan pada sistem robot manipulator ini agar pergerakan aktuator tidak terhenti karena harus meminta dan menunggu data masuk dari komputer.

Misalkan p = kode prosedur , dan j = parameter prosedurnya, maka isi paket dengan panjang = 8 byte data ditunjukkan pada gambar 3.13 (a). Untuk sistematika pengiriman dan permintaan byte data antara mikokontroler dan komputer ditunjukkan pada gambar 3.13 (b).



Gambar 3.13 (a) Contoh Paket Data yang dikirim Komputer ke Mikrokontroler, (b) Alur Komunikasi data Komputer dan Mikrokontroler

Selain untuk menggerakkan aktuator, komunikasi juga dimanfaatkan untuk mengirimkan data posisi potensiometer servo dan jumlah pulsa encoder motor DC. Dikarenakan di setiap *sampling time* semua data sensor posisi tersebut dibutuhkan PC agar bisa kemudian bisa ditampilkan dalam bentuk grafik, maka semua data sensor tersebut dimasukkan kedalam satu paket data dengan diawali header yang akan menunjukkan bahwa paket tersebut adalah paket yang mengirimkan data-data posisi sensor. Misal M = header paket, maka paket data tersebut ditunjukkan pada gambar berikut :

Gambar 3.14 Paket berisi data sensor posisi

3.4.2.2 Program gerak sinkron motor servo

Motor servo sudah memiliki sistem kendali posisi tersendiri didalamnya. Oleh karena itu untuk kendali motor servo melalui mikrokontroler cukup digunakan kendali *open loop*. Mikrokontroler cukup memberikan sinyal PWM yang merepresentasikan posisi sudut tertentu yang kemudian akan membiarkan servo itu sendiri yang mengendalikannya. Alasan penggunaan kendali *open loop* pada sistem ini dikarenakan pembebanan torsi pada servo tidak cukup besar sehingga diasumsikan bahwa servo selalu dapat mencapai posisi sesuai dengan sinyal PWM yang diberikan. Selain itu penggunaan metode *joint space trajectory planning* membuat pengaruh momentum yang dapat membuat osilasi pada posisi akhir servo menjadi berkurang. Yang menjadi permasalahan utama pada kendali servo adalah bagaimana mengendalikan 10 servo sehingga bersamaan.

Servo dikendalikan posisinya dengan memberikan sinyal PWM yang berkesesuaian. Untuk mengendalikan 10 servo secara bersamaan, berarti diperlukan 10 sinyal PWM yang dengan kontinus selalu mengendalikan servo. Hal ini tidak sulit dilakukan pada atmega2560 karena tersedianya 12 output PWM yang independen sehingga dapat memberikan sinyal PWM berbeda-beda untuk setiap servo.

Sinyal PWM yang digunakan adalah hasil *compare mode* pada timer 16-bit dengan mode *phase and frequency correct PWM with top = ICR* dan interrupt

pada *timer overflow*. Mode Phase and frequency correct PWM dipilih karena pada kendali multi servo sinyal PWM untuk semua servo diharapkan bisa berjalan pada fase yang sama. Dengan mode ini titik tengah positif pulsa semua sinyal PWM berada pada garis yang sama sehingga bisa dikatakan bahwa semua sinyal PWM berada pada fase yang sama seperti yang ditunjukkan oleh gambar 3.15.

Berikut adalah langkah –langkah dalam menghasilkan sinyal phase PWM untuk mengendalikan motor servo dan motor DC:

1. Pilih clock timer yang merupakan hasil pembagian clock utama mikrokontroler dengan prescaler tertentu.
2. Cari nilai top (nilai overflow) counter timer dengan persamaan berikut :

$$\text{top}=\frac{\text{clock timer}}{2 \text{ x frekuensi}} \qquad (3.7)$$

dimana besarnya frekuensi terbaik untuk sinyal kendali servo adalah 50 Hz atau periode sebesar 20 ms (periode = 1 / frekuensi).

Lalu masukkan ke register ICR, ICR = top.

3. Untuk memberikan sinyal kendali dengan lebar positif pulsa tertentu maka tinggal diset nilai OCR / nilai thresholdnya dengan memenuhi persamaan berikut:

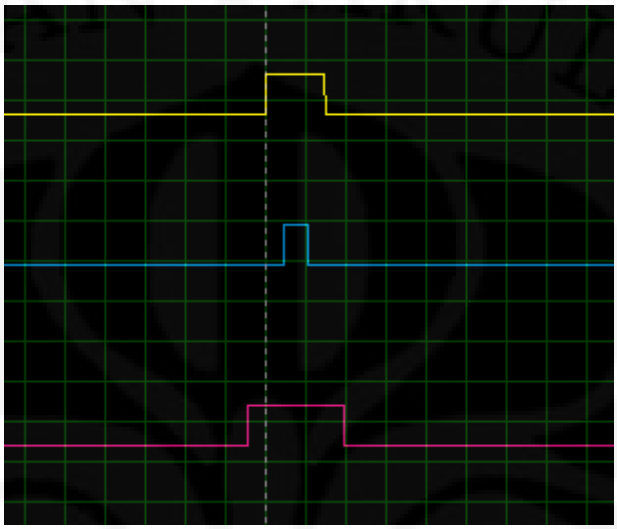
$$\text{OCR}=\frac{\text{lebar positif pulsa (us)}}{20.000 \text{ us}} \text{ x top} \qquad (3.8)$$

Setelah nilai OCR diset maka sinyal PWM yang dihasilkan akan selalu memiliki duty cycle yang sesuai dengan persamaan 2.15 hingga nilai OCR yang baru diterapkan.

4. Pada setiap perubahan satu periode sinyal PWM akan ada interrupt yang biasa disebut *Interrupt Timer Overflow*. Dengan memanfaatkan interrupt ini maka nilai OCR atau posisi servo akan bisa diubah setiap 20 ms yang dengan kata lain sampling time kendali servo adalah sebesar 20 ms. Pengaturan kecepatan dan *Cubic Trajectory planning* pun dengan mudah bisa di implementasikan.

Sistem ini menggunakan servo yang lebar pulsa positifnya berkisar antara 600 – 2400 us untuk menggerakkan servo ke posisi -90 ^{o} sampai 90^{o}. Besar clock utama adalah 12 MHz, dan prescaler yang dipakai adalah 8 sehingga clock timer

adalah 1.5 MHz. Untuk menghasilkan sinyal PWM dengan perioda 20 ms maka nilai top yang harus di set adalah 15.000. Sehingga nilai OCR yang akan menghasilkan lebar pulsa positif antara 600 – 2400 us adalah sebesar 450 - 1800.

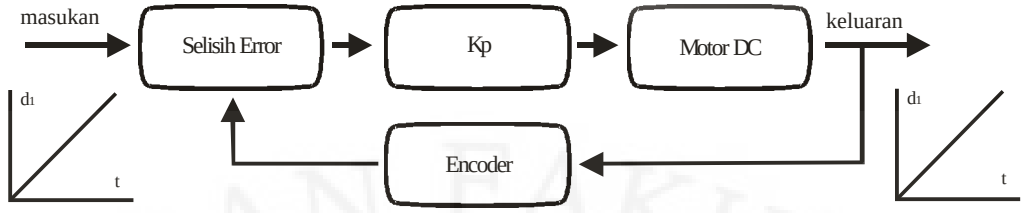


Gambar 3.15 Keluaran tiga sinyal PWM berbeda dengan metode Phase and Frequency Correct PWM

3.4.2.3 Program kendali motor DC

Motor DC yang digunakan diusahakan memiliki torsi sebesar mungkin sehingga pada posisi paling atas pun masih tetap bisa menahan lengan manipulator. Pada sistem ini digunakan motor DC dengan torsi yang cukup besar. Torsi pada motor DC ini bisa mengangkat lengan di posisi paling atas dalam kondisi motor DC tanpa catu daya apapun. Motor DC dengan torsi yang besar relatif lebih mudah di kendalikan karena kecepatannya tidak terlalu tinggi dan yang paling penting adalah pengaruh pembebanan torsi tidak mengganggu performa sistem kendali. Sehingga kompensator untuk kendali motor torsi ini tidak perlu rumit, cukup menggunakan kompensator P (proporsional) saja diharapkan keluaran sistem sudah cukup baik.

Pada sistem ini setiap motor DC dikendalikan secara independen posisinya dengan input berupa fungsi ramp dan kompensator berupa kontroler P (proporsional) seperti yang terlihat pada gambar 3.16.



Gambar 3.16 Blok Diagram Pengendali motor DC

Disetiap waktu pencuplikan / *sampling time* yang diset sebesar 20 ms, mikrokontroler akan melakukan pengecekan posisi motor DC dari perubahan jumlah pulsa pada encoder. Posisi yang terbaca akan diselisihkan dengan posisi yang harus dicapai pada *sampling time* berikutnya. Perbedaan selisih ini akan direspon dengan pemberian sinyal PWM untuk kecepatan motor DC yang sesuai. Dengan sistem kendali seperti ini diharapkan motor DC dapat selalu mengikuti acuan posisi yang merupakan fungsi trajectory pada persamaan Cubic polynomial trajectory planning.

3.4.3 Simulasi Program

3.4.3.1 Point to Point Motion

Point to point motion lebih menitik-beratkan pada bagaimana perubahan posisi dan kecepatan di masing-masing sendi sehingga end effector dapat bergerak dari titik awalnya menuju titik acuan dengan cepat dan tepat. *Joint space trajectory planning* merupakan metode kendali yang sangat tepat dan cepat untuk maksud ini. Berikut disimulasikan gerakan salah satu sendi manipulator baik menggunakan *linear trajectory planning* maupun dengan menggunakan *Cubic Polynomial Trajectory Planning*.

a. Simulasi gerak sendi menggunakan *linear trajectory planning*

Gerak sendi menggunakan *linear trajectory planning* disimulasikan dengan perpindahan sendi dari suatu titik ke titik lain tanpa perubahan kecepatan sama sekali. Berbeda dengan LSPB, pada *linear trajectory planning* sendi akan berpindah secara linier memenuhi persamaan sederhana dibawah :

$$q(t) = a_0 + a_1 t, \quad (3.9)$$

$$v(t) = a_1, \quad (3.10)$$

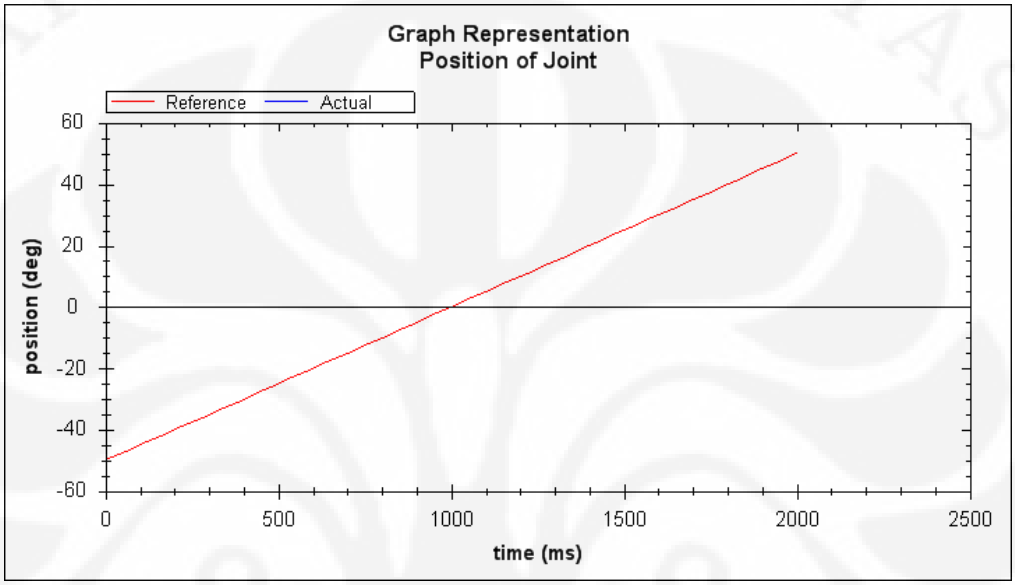
$$\text{dimana } a_o = q_o, \text{ dan } a_1 = (q_f - q_o) / t_f, \quad (3.11)$$

Simulasi dilakukan dengan mendefinisikan q_0, q_f, t_o, dan t_f sebagai berikut :

$q_o = -50^o$ $q_f = +50^o$

$t_o = 0$ s $t_f = 2000$ ms

Berikut adalah grafik perubahan posisi sendi dengan sampling time sebesar 20 ms :



Gambar 3.17 Perubahan posisi pada *linear trajectory planning*

b. Simulasi gerak sendi menggunakan *Cubic polynomial*

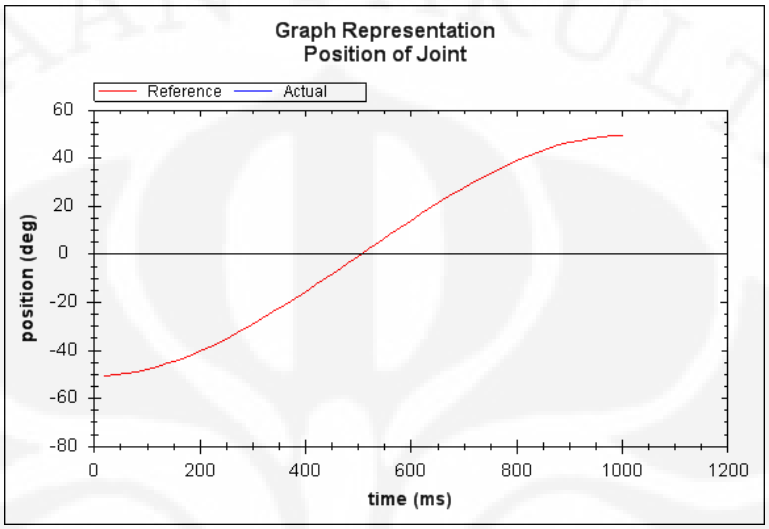
Pada aplikasi nyata penggunaan *linear trajectory planning* jarang digunakan dikarenakan akan adanya ketidak-kontinuan kecepatan pada setiap perpindahan path point. Hal tersebut akan mengakibatkan gerakan aneh pada setiap perpindahan *path point* yang disebut jerk. Oleh karena itu dibutuhkan *trajectory planning* yang bisa mengatasi masalah kecepatan awal dan akhir di setiap *path points*. Seperti yang telah dijelaskan pada bagian teori, *cubic polynomial trajectory planning* dapat digunakan untuk mengatasi masalah tersebut.

Simulasi untuk gerak sendi menggunakan *Cubic polynomial trajectory planning* dilakukan dengan nilai empat batasan yang diperlukan sebagai berikut :

$q_o = -50^o$ $q_f = +50^o$

$V_o = 0$ deg/s $V_f = 0$ deg/s

Dengan mengetahui terlebih dahulu nilai koefisien bebas melalui perkalian matriks pada persamaan 2.12, maka grafik perubahan posisi dan kecepatan sendi dengan sampling time sebesar 20 ms dari dari t_o = 0 s hingga t_f = 2000 ms adalah sebagai berikut :



Gambar 3.18 Trajectory planning dengan *Cubic polynomial*

3.4.3.2 Continuous Path Tracking

Jika *Joint space trajectory planning* baik digunakan pada sistem dengan gerakan titik ke titik, maka *Cartesian trajectory planning* baik digunakan pada sistem yang diharuskan secara kontinu mengikuti jalur tertentu (*continuous path tracking*). Seperti dijelaskan pada subbab 3.4.1.4, dilakukan beberapa tahapan dalam melakukan *continuous path tracking* ini dimulai dari penentuan titik singgah / *via points* dengan *path planning* hingga mendapatkan persamaan gerak sendi hasil *cubic trajectory path planning* antar *via points* yang berurutan.

a. Perhitungan path planning dengan interpolasi linier

Pada simulasi gerak sistem ini, didefinisikan sebuah lintasan garis lurus. Lintasan garis lurus yang ditempuh menghubungkan titik P_1 (180,-40,0) dan P_2(180,40,0) mm . Dengan menggunakan time sampling (t) sebesar 100 ms dan konstanta atau waktu tempuh (k) sebesar 1000 ms. Akan didapat sepuluh titik singgah dengan koordinat yang memenuhi persamaan berikut :

$$P_x(t) = 180 + \frac{180-180}{1000} x100t = 180$$

$$P_y(t) = -40 + \frac{40-(-40)}{1000} x100t = -40 + 8t$$

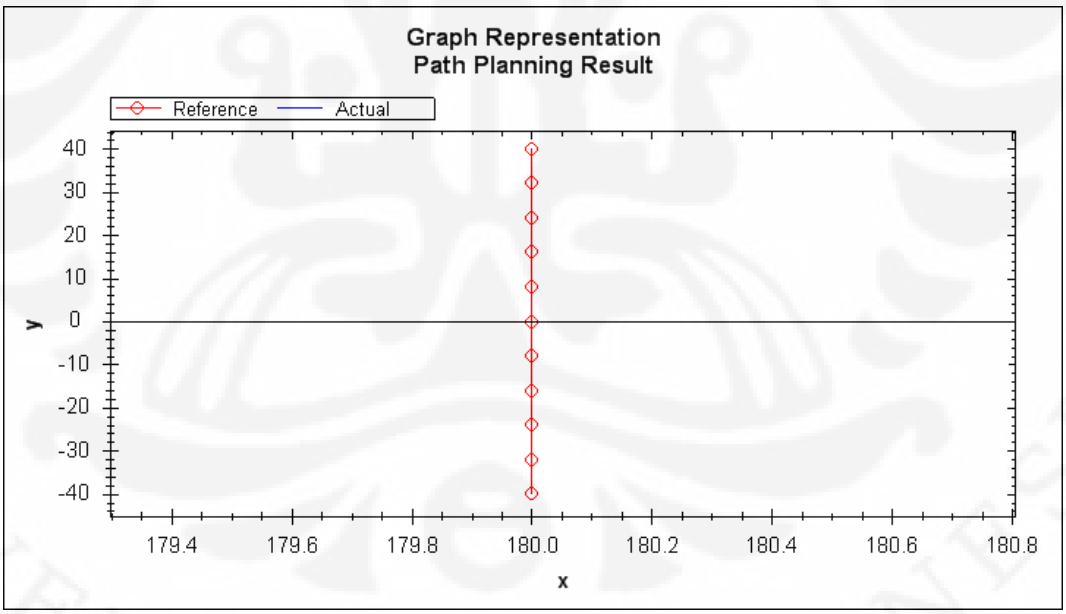
$$P_z(t) = 0 + \frac{0-0}{1000} x100t = 0$$

Sepuluh titik singgah tersebut ditunjukkan pada tabel berikut :

Tabel 3.3 Koodinat titik-titik singgah hasil Interpolasi Linier garis lurus

T	1	2	3	4	5	6	7	8	9	10
P (x)	180	180	180	180	180	180	180	180	180	180
P (y)	-32	-24	-16	-8	0	8	16	24	32	40
P (z)	0	0	0	0	0	0	0	0	0	0

Dan direpresentasikan oleh gambar berikut :



Gambar 3.19 Koodinat titik-titik singgah hasil Interpolasi Linier garis lurus

b. Perhitungan *inverse kinematics*

Program *inverse kinematics* pada subbab 3.4.1.2 digunakan untuk mengkonversi koordinat pada bidang cartesian menjadi konigurasi sudut-sudut sendi manipulator. Dari data yang didapat pada tabel 3.3, dihasilkan

konfigurasi sendi untuk setiap titik singgah seperti ditunjukkan dalam tabel 3.4 berikut :

Tabel 3.4 Hasil Perhitungan *Inverse Kinematics* untuk tiap titik singgah pada lintasan garis lurus

T	1	2	3	4	5	6	7	8	9	10
θ_1	-50	-46	-43	-39	-35	-30	-25	-21	- 16	-11
θ_2	83	86	87	88	88	88	87	85	83	80
θ_3	-33	-39	-44	-49	-54	-58	-61	-65	-67	-69
d_1	0	0	0	0	0	0	0	0	0	0

c. Simulasi gerak *end effector* menggunakan *Cubic polynomial*

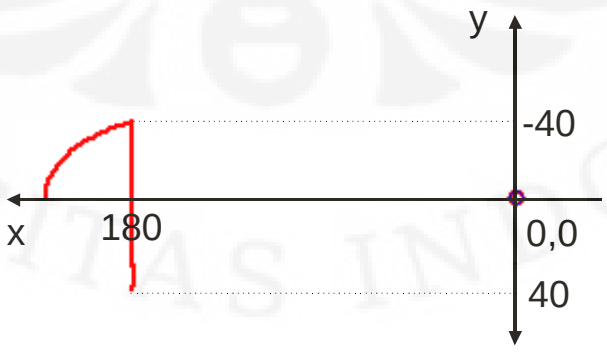
Hasil *inverse kinematics* akan digunakan untuk masukan persamaan gerak menggunakan *cubic polinomial* dengan tambahan batasan lain berupa kecepatan di setiap titik singgah dan waktu tempuh antar titik singgah.

Misalkan diinginkan perhentian disetiap path points, maka digunakan aturan sebagai berikut :

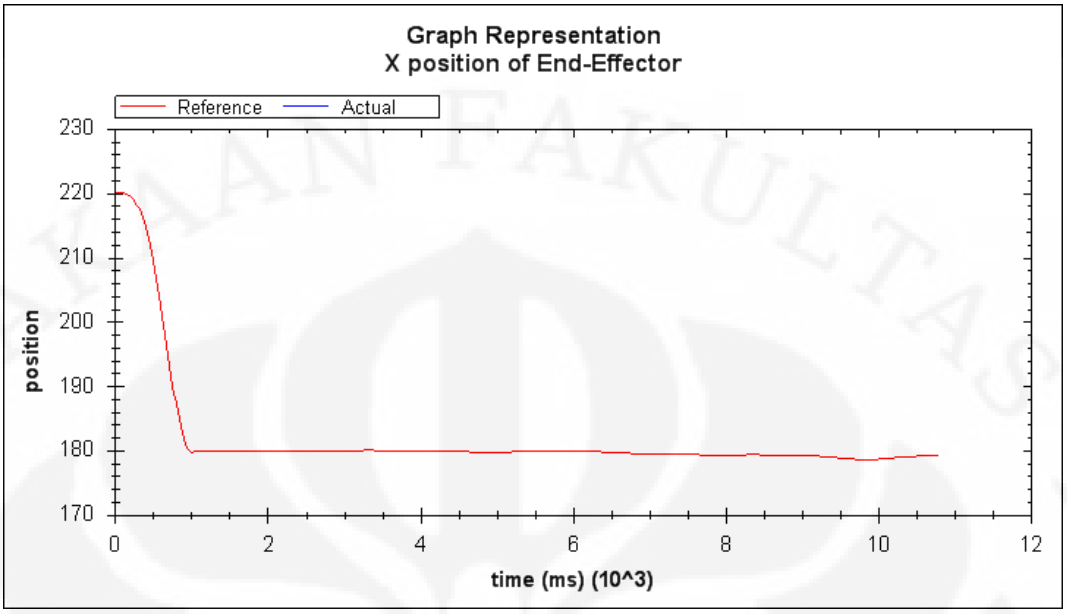
$v_{n0} = 0$, $v_{nf} = 0$, dimana n = 1,2,3 ….. 10.

maka untuk 4 sendi dengan masing-masing 10 path point, maka akan dihasilkan 10 kurva untuk masing-masing sendi tersebut yang akan serupa dengan kurva pada gambar 3.18.

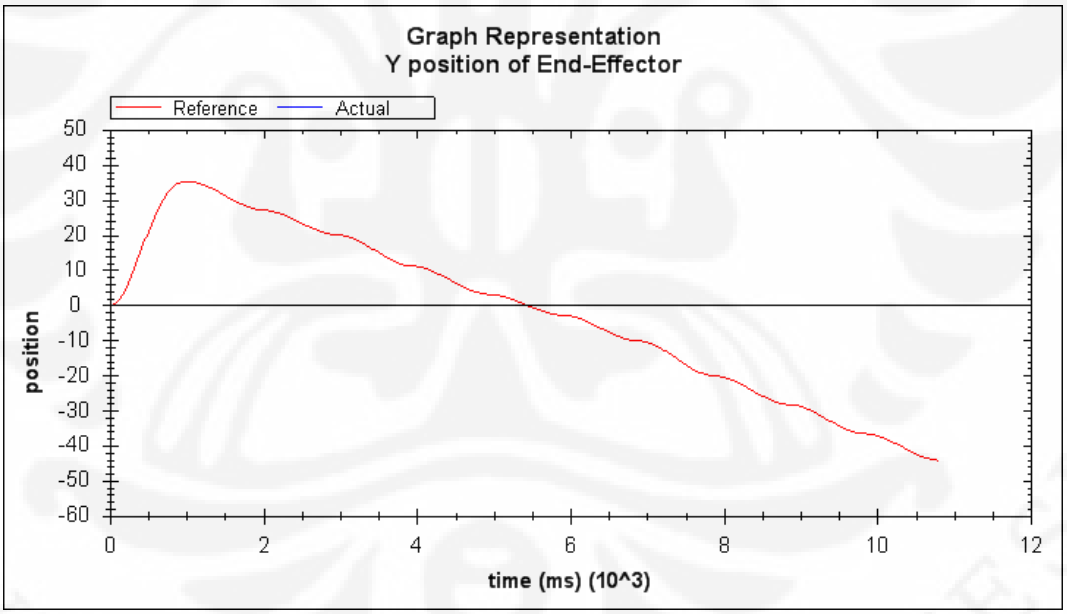
Dari data tersebut kemudian digunakan program *forward kinematics* untuk mengetahui posisi end-effector. Berikut adalah hasil simulasi pergerakan end-effector untuk semua konfigurasi sendi di setiap path points pada tabel 3.4 :



Gambar 3.20 Posisi End-effector untuk p*ath points* pada tabel 3.4



Gambar 3.21 Perubahan posisi End-effector pada sumbu X untuk p*ath points* pada tabel 3.4



Gambar 3.22 Perubahan posisi End-effector pada sumbu Y untuk p*ath points* pada tabel 3.4

Pada aplikasi nyata, perhentian pada setiap titik-singgah diharapkan tidak ada atau dengan kata lain sistem diharapkan dapat bergerak kontinus agar mengurangi *cycle time*. Sehingga diperlukan aturan untuk menentukan nilai kecepatan awal dan akhir pada setiap path points. Aturan yang paling sederhana adalah sebagai berikut :

$v_{1\text{-}0} = 0$, $v_{10\text{-}f} = 0$, $v_{n\text{-}f} = v_{(n+1)\text{-}0}$, dimana n = 1,2,3 ….. 10.

Nilai kecepatan awal dan akhir pada setiap path points tersebut dapat bervariasi dan biasanya diatur dengan memperhatikan beban pada manipulator. Semakin berat beban pada *end-effector* robot manipulator biasanya membutuhkan kecepatan awal yang cukup besar sehingga beban dengan mudah dapat segera bergerak, dan membutuhkan kecepatan akhir yang kecil sehingga pengaruh momentum dapat berkurang.

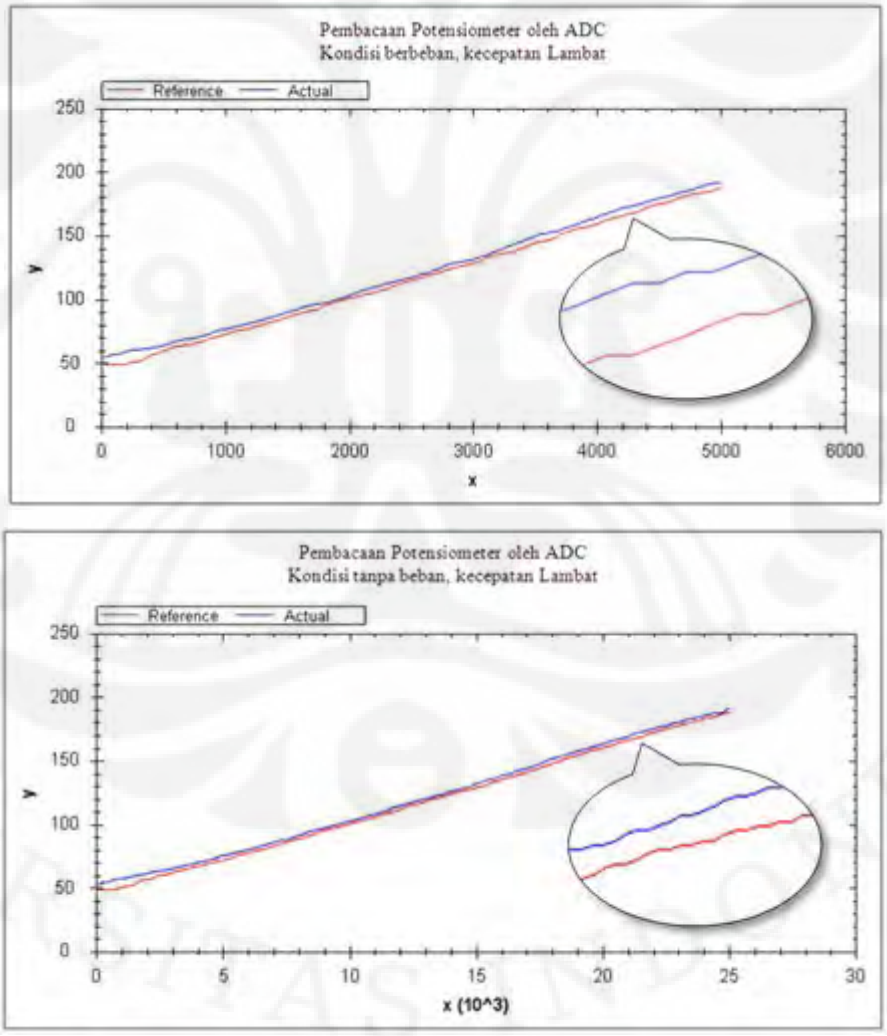
BAB 4

PENGUJIAN DAN ANALISIS SISTEM

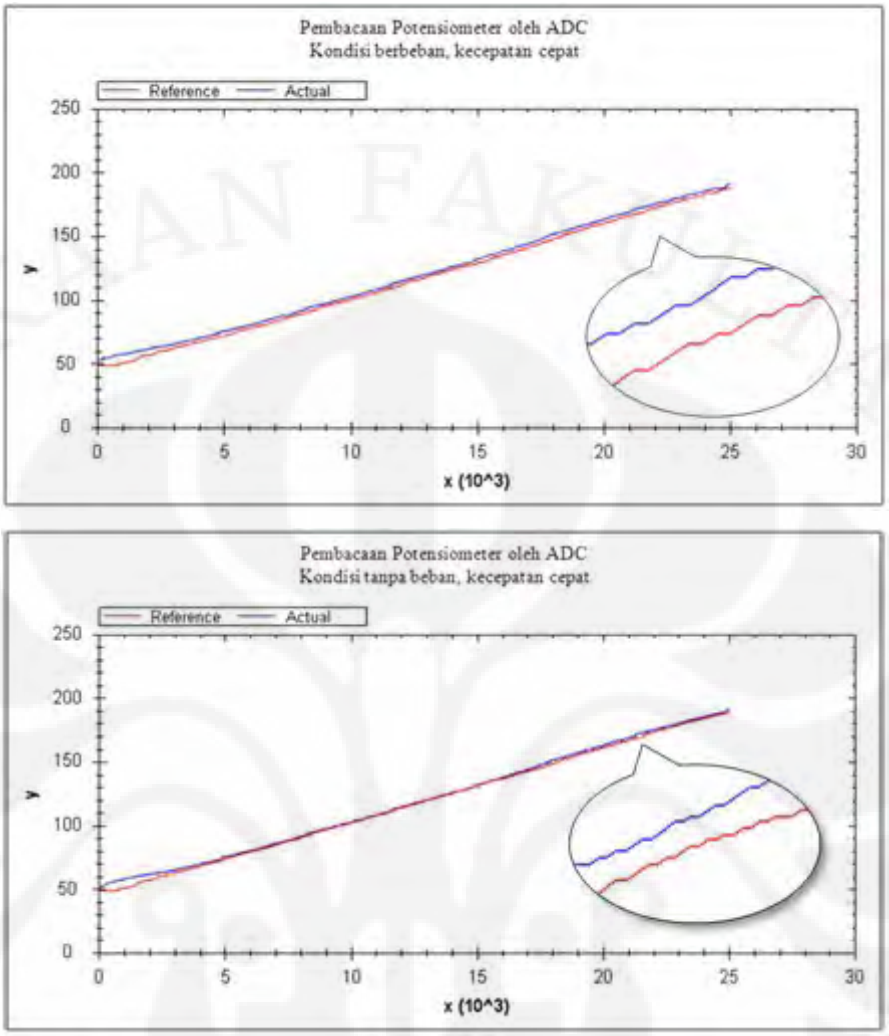
4.1 Pengujian Perangkat keras

4.1.1 Uji linearitas dan histerisis potensiometer

Sensor potensiometer yang digunakan diuji dengan menempelkannya pada servo. Servo kemudian digerakkan di dua kecepatan ko nstan berbeda yaitu kecepatan lambat dan cepat, dari posisi awal hingga akhir lalu kembali ke posisi awal. Dan didapatkan kurva hubungan antara pembacaan naik (kurva berwarna merah) dan pembacaan turun (kurva berwarna biru) nilai ADC dengan waktu pembacaan sensor untuk kecepatan lambat dan cepat sebagai berikut :



Gambar 4.1 Pembacaan ADC pada potensiometer dengan kecepatan servo lambat



Gambar 4.2 Pembacaan ADC pada potensiometer dengan kecepatan servo cepat

Dengan asumsi bahwa waktu pembacaan sebanding dengan posisi servo yang menyatakan bahwa servo benar-benar bergerak konstan, maka diperoleh hasil dimana potensiometer memiliki kelinieran yang cukup tinggi dan error hysterisis yang cukup kecil.

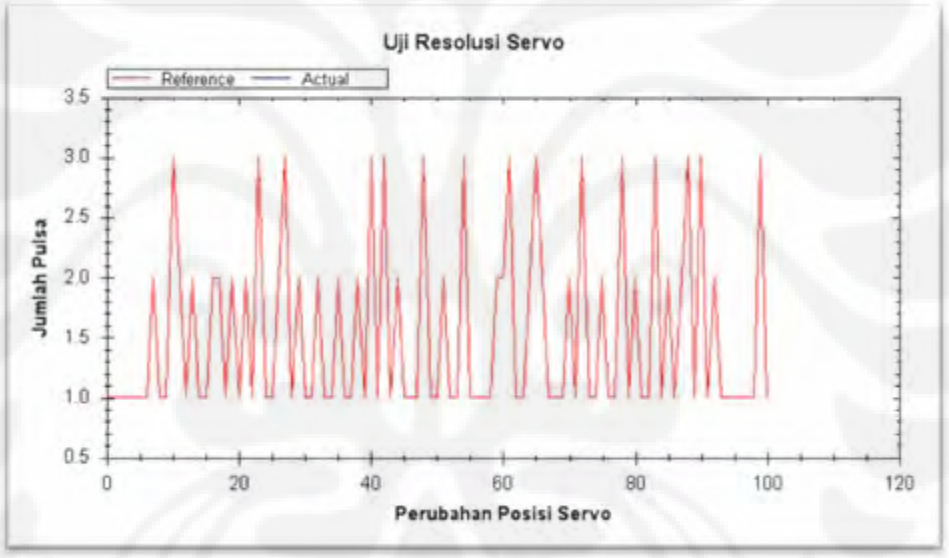
Pada pengambilan data ADC potensiometer dengan kecepatan servo yang lambat, terlihat grafik yang cukup halus dibandingkan dengan grafik pada kecepatan cepat. Hal ini dipengaruhi oleh besarnya resolusi ADC yang digunakan yaitu hanya 8 bit. Dengan resolusi sekecil ini berarti tingkat akurasi pembacaan posisi pada potensiometer hanya sebesar 360^o / 256 bit = 1.41^o / bit saja. Hal ini membuat simulasi pergerakan manipulator dan grafik data posisi *end-effector* manipulator yang ditunjukkan *software* PC kurang begitu merepresentasikan posisi yang sebenarnya.

4.1.2 Uji karakteristik motor servo GWS 2BBMG

Resolusi Servo

Resolusi servo merupakan waktu pulsa positif terkecil yang dapat memberikan perubahan pada posisi motor servo. Walaupun servo dikendalikan dengan sinyal PWM antara 600 us hingga 2400 us belum berarti setiap perubahan 1us mengakibatkan perubahan pada posisi motor servo. Untuk kendali posisi motor servo, resolusi ini sangat penting mengingat dapat presentase error posisi akan semakin besar dengan semakin buruknya resolusi motor servo.

Motor servo diberikan sinyal PWM dengan perioda 20 ms dan lebar pulsa positif yang selalu bertambah 1,3 uS. Berikut adalah data hasil pengujiannya :



Gambar 4.3 Resolusi Servo GWS 2BBMG

Dari data yang didapat bisa diketahui resolusi servo dengan mengukur waktu mulai hingga akhir pengujian dibagi dengan jumlah gerakkan yang dilakukan servo. Tercatat bahwa terjadi sebanyak 400 kali pengiriman sinyal PWM dengan lebar pulsa sebesar 1.33 us. Sehingga dapat diketahui bahwa waktu mulai hingga akhir pengiriman adalah selama 520 us. Selama waktu itu tercatat terjadi 180 kali gerakan servo. Sehingga didapat resolusi servo sebesar :

Resolusi = waktu uji / jumlah gerakan

= 520 us / 180

= 2.89 uS = 0.289^{o}

Kalibrasi Posisi Servo

Kalibrasi servo dilakukan dengan terlebih dahulu mengecek nilai ADC pada kondisi normalnya yaitu posisi 0^o. Setelah itu servo tidak diberikan catu daya dulu dan digerakkan menuju posisi 90^o dan -90^o menggunakan acuan sebuah busur derajat. Pada posisi tersebut dicatat nilai ADC rata-ratanya yang kemudian akan digunakan untuk menentukan berapa lebar pulsa positif PWM yang diperlukan servo untuk menuju posisi-posisi tadi. Berikut adalah tabel hasil pembacaan ADC serta lebar positif pulsa PWM pada posisi-posisi tadi :

Tabel 4.1 Kalibrasi servo pada posisi maksimal dan netralnya

Posisi Servo	Nilai ADC	Lebar Pulsa Positif PWM
-90^o	62	518
0^o	126	1125
90^o	191	1732

Dari hasil kalibrasi tersebut didapatkan persamaan lebar pulsa positif PWM terhadap motor servo sebagai berikut :

Lebar pulsa positif PWM = 518 + (derajat servo + 90) x 1214 / 180.0)

Kalibrasi perlu dilakukan pada semua servo karena pemasangan sendi dan lengan tidak bisa tepat di posisi netral servo.

4.1.3 Uji pengiriman data serial

Transmisi serial yang digunakan menggunakan konfigurasi sebagai berikut:

a. Baud rate = 9600
b. Data bits = 8
c. Stop Bits = 1

Pengujian dilakukan dengan mengendalikan manipulator melalui PC dengan data berupa nomor servo beserta nilai pulsa posistif PWM-nya. Pada sistem ini terdapat 10 motor servo. Setiap motor dapat dikendalikan secara terpisah ataupun bersamaan.

Pengujian pertama dilakukan dengan memberikan instruksi kepada mikrokontroler untuk menjalankan tiap servo secara berurutan ke posisi 90^o. Setelah itu semua servo diinstruksikan untuk kembali ke posisi 0^o masing-masing secara bersamaan. Instruksi ini diulang kembali namun dengan tambahan parameter waktu tempuh servo yang dibuat semakin cepat hingga mencapai turn ratenya. Dan hasilnya adalah semua servo bergerak sebagaimana mestinya.

Pengujian kedua dilakukan dengan merekam data posisi sensor ADC yang diukur potensimeter setiap servo per 20 ms sekali oleh komputer. Instruksi gerakan servo yang diberikan masih sama dengan pengujian pertama. Pada pengujian ini dilakukan perekaman data untuk beberapa posisi servo. Untuk perekaman data posisi hingga tiga buah servo, manipulator masih dapat berjalan sebagaimana yang diperintahkan. Tetapi tidak halnya jika perekaman data dilakukan untuk lebih dari tiga servo. Pada perekaman data posisi untuk lebih dari tiga servo sering terjadi hang, dimana tidak ada lagi servo yang bergerak.

Salah satu alasan yang memungkinkan hal ini terjadi karena kecepatan pengiriman yang kurang terlalu cepat yaitu hanya 9600 bps. Sebenarnya kecepatan pengiriman data bisa dinaikkan hinnga 38400 bps, namun dikarenakan protokol pada komunikasi di sistem ini tidak bisa mentolerir adanya error maka pilihan tersebut tidak diambil.

Solusi yang kemudian digunakan adalah dengan membatasi jumlah data posisi sensor yang bisa dikirimkan ke PC per 20 ms menjadi hanya untuk 3 posisi sensor saja. Karena manipulator seharusnya merngirimkan 8 data sensor (masing-masing 6 sensor potensiometer dan 2 sensor encoder) setiap 20 ms sekali, maka sekarang pengiriman data posisi tiap sensor hanya bisa dilakukan setiap 60 ms. Hal ini mengakibatkan simulasi pergerakan manipulator yang ditunjukkan pada *software* PC menjadi kurang halus karena perubahan posisi dilakukan pada waktu yang lebih lama.

4.2 Pengujian Sistem untuk Gerak Point to Point dengan Joint Space Trajectory Planning

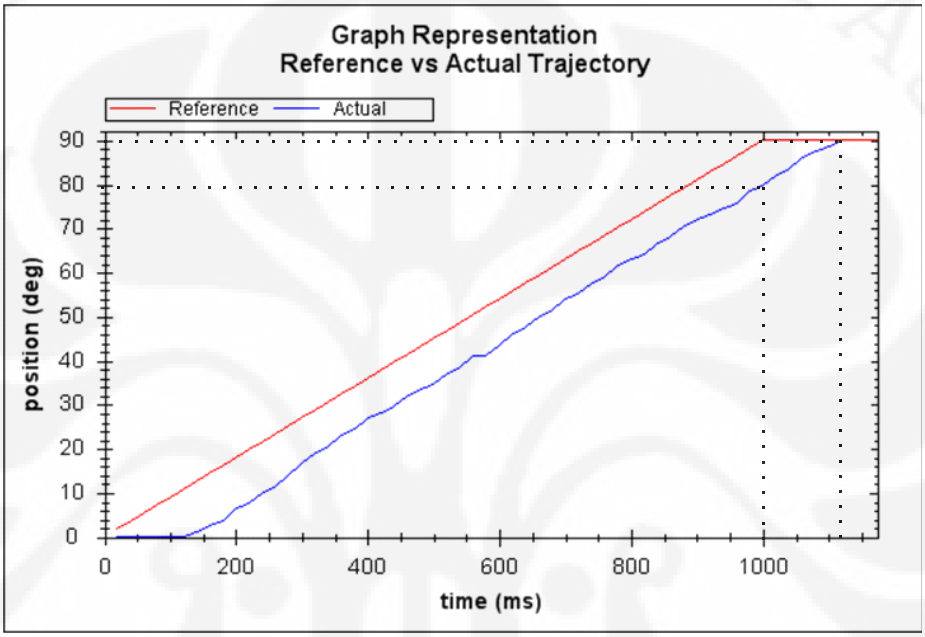
4.2.1 Uji gerak sendi menggunakan *linear trajectory planning*

Pengujian gerak sendi ini dilakukan pada servo yang telah terkalibrasi posisi 0 dan 90 derajatnya dengan menggunakan ADC dan busur. Sinyal kendali

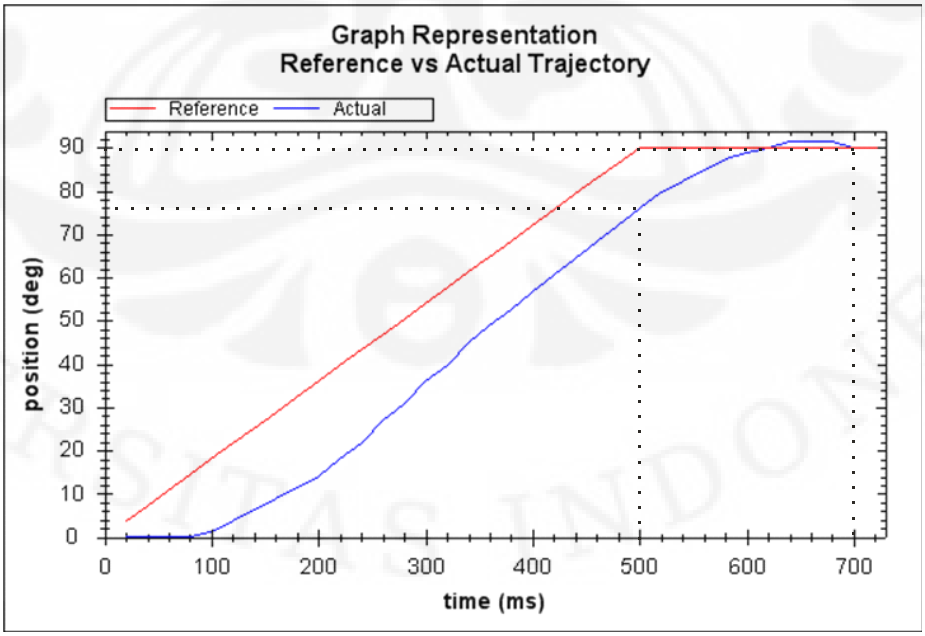
yang dikirimkan merupakan fungsi posisi terhadap waktu yang memenuhi persamaan berikut :

$$q(t) = \frac{(qf - qo)}{t_f} \; t \quad \text{dimana } t = 20 \text{ ms}$$

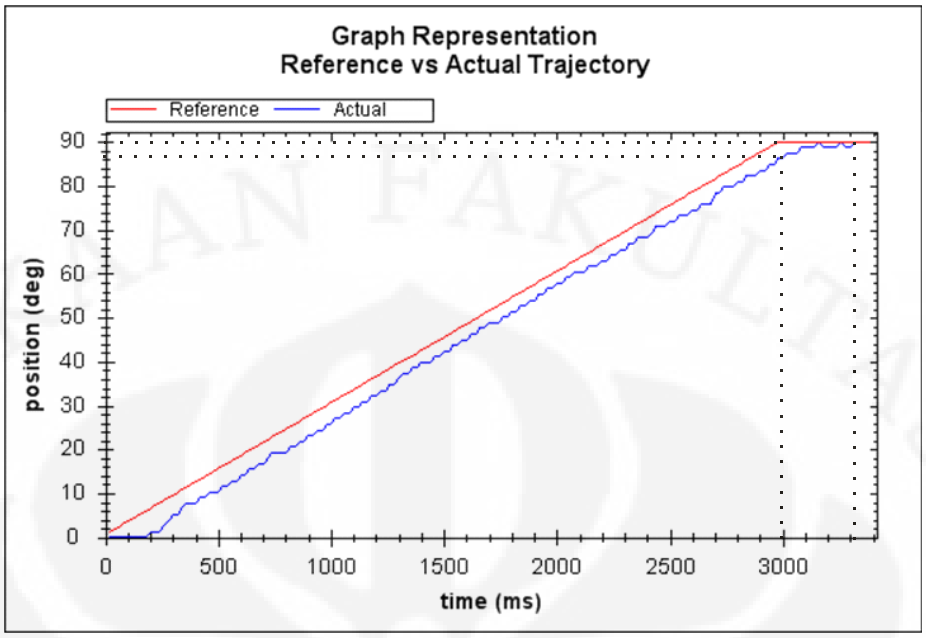
Pengujian dilakukan dengan beberapa parameter t_f. Dan didapatkan masing –masing grafik posisi untuk setiap t_f sebagai berikut :



(a) t_f = 1000 ms



(b) t_f = 500 ms



(c) t_f = 3000 ms

Gambar 4.4 Grafik hasil uji gerak point to point motion menggunakan *linear trajectory planning*

Tabel 4.2 Hasil uji gerak point to point motion menggunakan *linear trajectory planning*

	tf = 1000 ms	tf = 500 ms	tf = 3000 ms
delay (ms)	100	80	100
transient time (ms)	1115	700	3300
$q(t_f)$ (deg)	80	76	87
error posisi	11.11 %	15.56 %	3.33 %

Keterangan :

- delay adalah waktu keterlambatan sendi dalam mengikuti trajectory yang diberikan.
- Transient time adalah waktu hingga kondisi stabil
- $q(t_f)$ adalah posisi sendi pada saat waktu tempuh (t_f)
- error posisi adalah kesalahan posisi sendi terhadap trajectory referensi pada waktu tempuh (t_f)

4.2.2 Uji gerak sendi menggunakan *Cubic polynomial trajectory planning*

Untuk membandingkan hasil pergerakan sendi antara tanpa *trajectory planning* dengan menggunakan cubic *polynomial trajectory planning*. Maka

parameter q_o, q_f, dan t_f dibuat sama. Perbedaannya adalah pada penggunaan batasan berupa $v_o = 0$ dan $v_f = 0$.

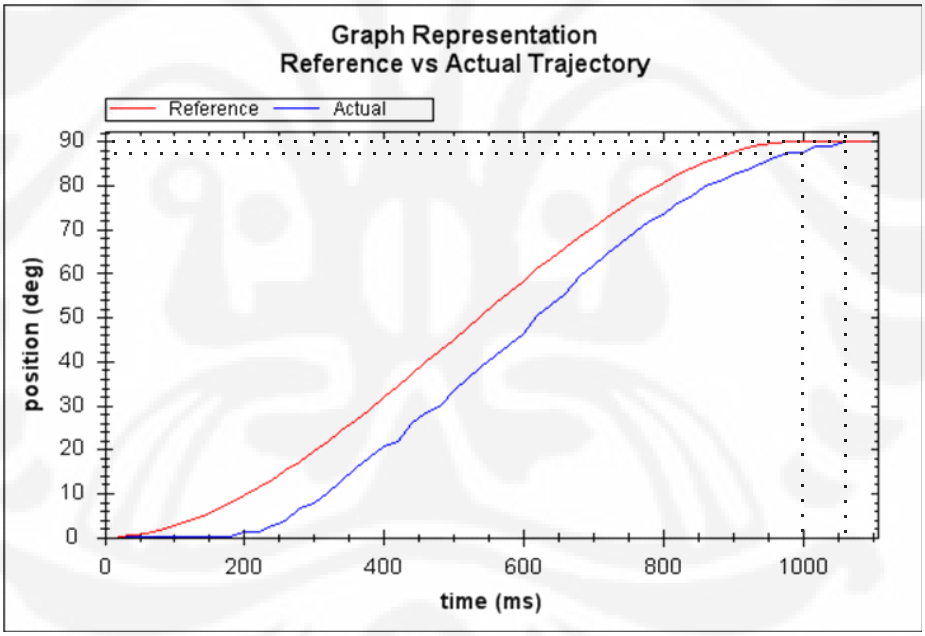
Dengan menggunakan *trajectory planning* maka pergerakan sendi diatur setiap waktu cupliknya memenuhi persamaan berikut :

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

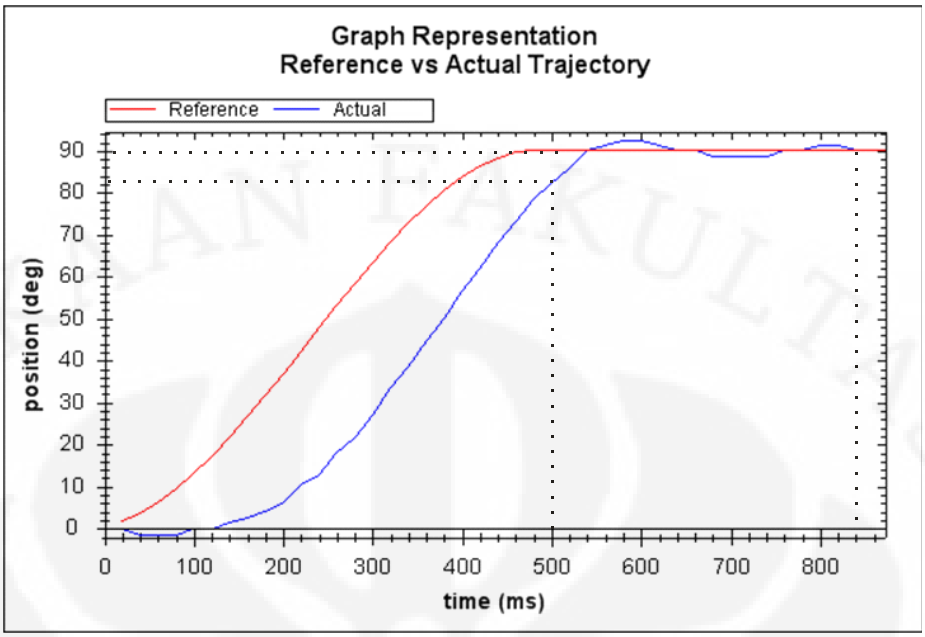
Dimana koefisien bebasnya dapat dicari dengan persamaan matriks berikut :

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix}$$

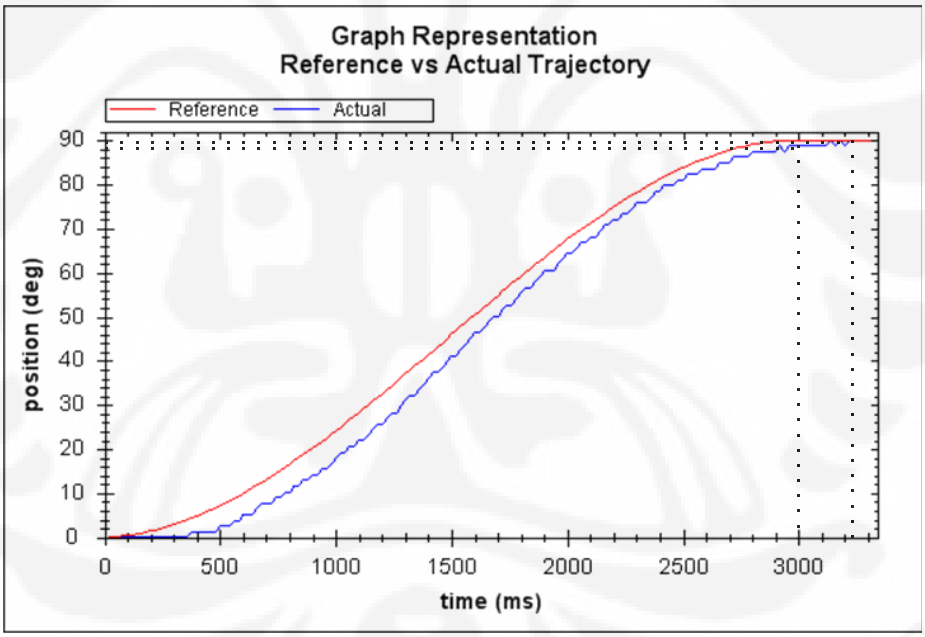
Pengujian dilakukan pada sendi yang sama, dan didapatkan masing – masing grafik posisi acuan dan hasil uji untuk setiap t_f sebagai berikut :



(a) t_f = 1000 ms



(b) t_f = 500 ms



(c) t_f = 3000 ms

Gambar 4.5 Grafik hasil uji gerak point to point motion menggunakan *cubic polynomial trajectory planning*

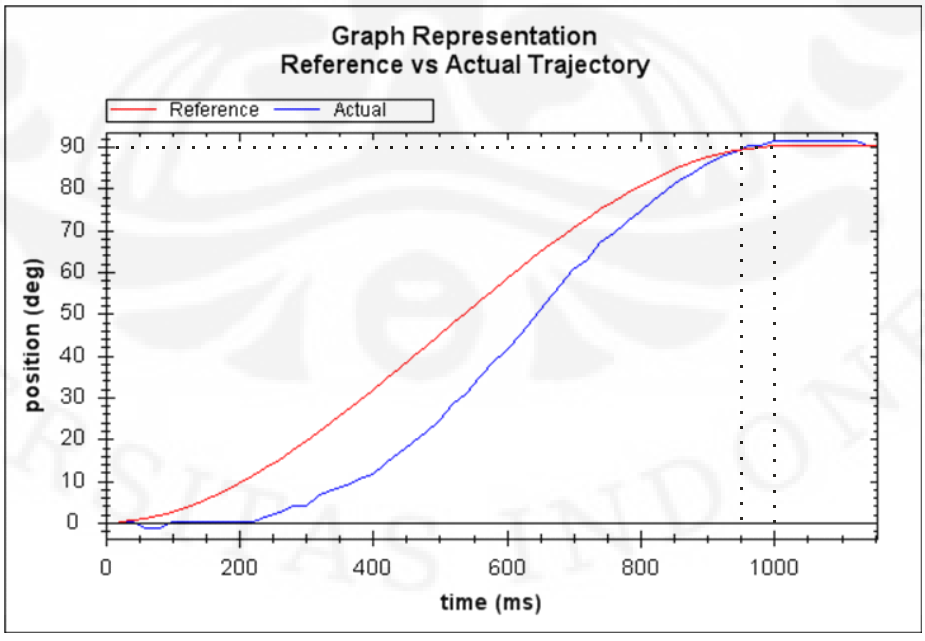
Tabel 4.3 Hasil uji gerak point to point motion menggunakan *cubic trajectory planning*

	tf = 1000 ms	tf = 500 ms	tf = 3000 ms
delay (ms)	90	140	100
transient time (ms)	1060	840	3220
$q(t_f)$ (deg)	88	83	88.5
error posisi	2.22 %	7.78 %	1.67 %

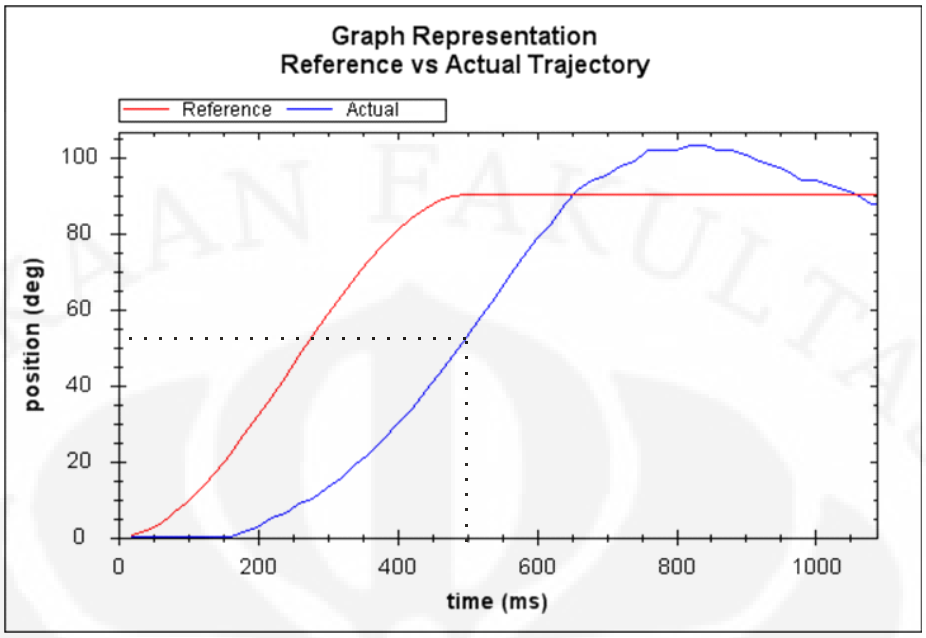
Dari tabel 4.3 diketahui bahwa error posisi di sekitar waktu tempuh dan delay selama pergerakan lebih baik dibandingkan pada tabel 4.2. Walaupun sebenarnya delay pada t_f = 500 ms lebih lama, namun itu lebih disebabkan karena spesifikasi sistem yang kurang baik dalam melakukan gerak dengan percepatan yang cukup tinggi. Sehingga dari pengujian ini dapat diketahui bahwa *cubic trajectory planning* akan lebih baik diterapkan pada pergerakan *continuous path tracking* karena lebih baik dalam mengikuti trajectory yang diberikan.

4.2.3 Uji gerak sendi berbeban menggunakan *Cubic polynomial trajectory planning*

Pengujian dilakukan dengan memasangkan beban pada ujung lengan sendi yang digerakkan dengan *cubic trajectory planning* menggunakan parameter yang sama dengan pengujian sebelumnya.

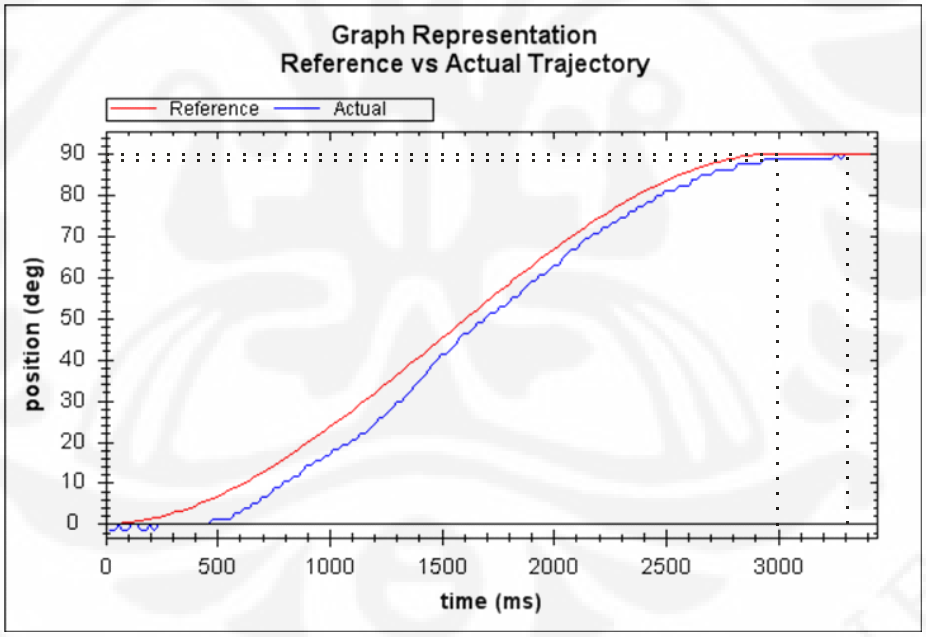


(a) t_f = 1000 ms



(b) t_f = 500 ms

a. t_f = 3000 ms



(c) t_f = 3000 ms

Gambar 4.6 Grafik hasil uji gerak point to point motion menggunakan *cubic polynomial trajectory planning* pada sendi berbeban

Tabel 4.4 Hasil uji gerak point to point motion menggunakan *cubic trajectory planning* pada kondisi berbeban

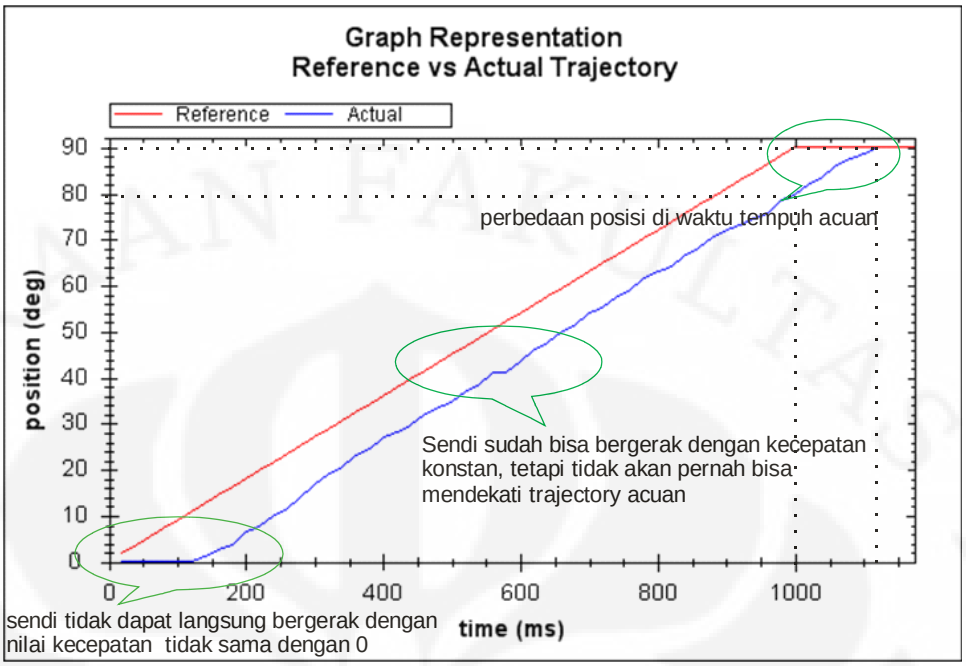
	tf = 1000 ms	tf = 500 ms	tf = 3000 ms
delay (ms)	175	220	180
transient time (ms)	1150	1275	3320
$q(t_f)$ (deg)	88.5	53	88
error posisi	1.67 %	41.1 %	2.22 %

Jika dibandingkan dengan data pada tabel 4.3, dapat kita ketahui bahwa pengaruh penambahan beban cukup signifikan baik pada membesarnya delay dan error posisi. Hal ini disebabkan oleh bertambahnya pengaruh momen gaya, momen inersia dan momentum sistem. Dikarenakan sebuah sistem pasti memiliki batasan tertentu berupa spesifikasi torsi, kecepatan dan respon gerak. Maka dengan pengaruh penambahan beban tersebut dibutuhkan perencanaan *trajectory* yang dapat mengatasinya, misalnya dengan mendefinisikan kecepatan atau percepatan awal gerak sistem. *Cubic trajectory planning* adalah salah satu *trajectory planning* yang mampu mengatasi batasan tersebut karena dapat mendefinisikan kecepatan awal dan akhir sistem. Sehingga *cubic trajectory planning* akan lebih baik diterapkan pada kendali gerak baik *point to point* maupun *continuous path tracking.*

4.3 Analisis Hasil Pengujian Sistem untuk Gerak Point to Point

4.3.1 *Linear vs Cubic Trajectory Planning*

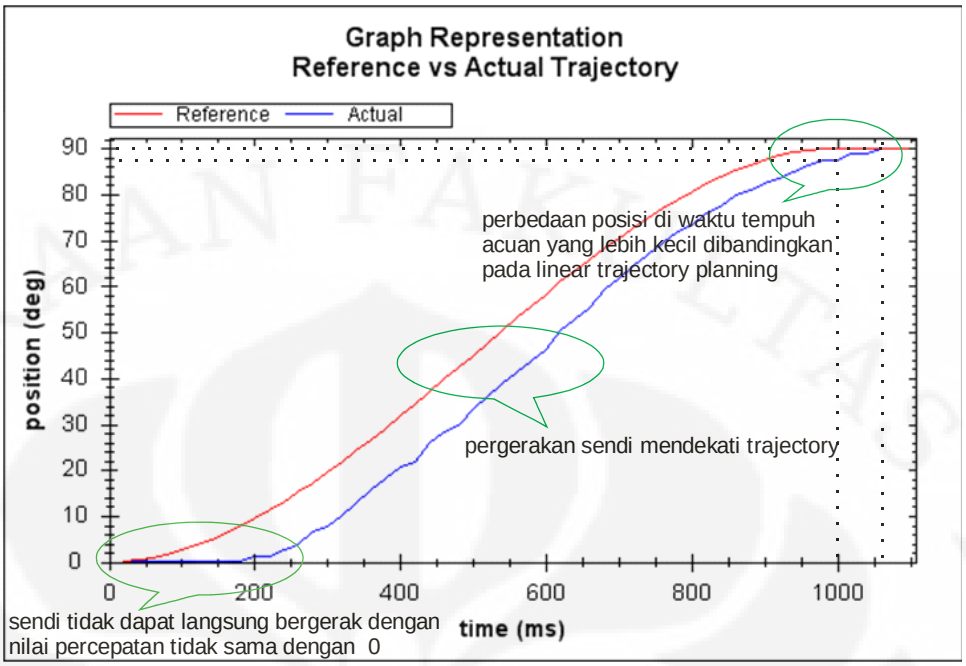
Pada *linear trajectory planning,* kecepatan disepanjang *trajectory* acuan adalah konstan yaitu sama dengan perubahan posisi dibagi waktu tempuh. Ini berarti bahwa kecepatan di awal dan di akhir posisi pun akan sama dengan kecepatan konstan tersebut. Dalam kondisi yang sebenarnya, sendi akan bergerak dari posisi diam dan berhenti ketika mencapai posisi akhir. Hal ini menunjukkan bahwa pada kondisi yang sebenarnya akan terjadi perubahan kecepatan tiba-tiba di posisi awal, dari kecepatan nol menuju kecepatan konstan pada *trajectory* acuan. Perubahan kecepatan tiba-tiba ini dinamakan ketidak-kontinuan kecepatan.



Gambar 4.7 Pergerakan sendi pada *linear trajectory planning*

Ketidak-kontinuan kecepatan ini akan berakibat pada kemampuan sistem mengikuti *trajectory* yang diberikan. Di posisi awal sudah pasti sendi tidak akan bisa mengikuti *trajectory* acuan karena nyatanya perubahan kecepatan sendi tidak bisa langsung menuju kecepatan konstannya. Kemudian dikarenakan kecepatan pada *linear trajectory planning* adalah konstan, maka selanjutnya dipastikan sendi tidak akan bisa mengikuti *trajectory* acuan walaupun sendi sudah bisa bergerak dengan kecepatan konstan yang direferensikan. Representasi dari keadaan ini dijelaskan lebih detail pada gambar 4.7 diatas.

Ketidak-kontinuan kecepatan juga akan berakibat pada kemampuan sistem menuju set point yang diberikan. Dari gambar 4.6 terlihat bahwa posisi sendi selalu tertinggal oleh posisi acuannya. Pada waktu tempuh yang ditetapkan yaitu 1000 ms, seharusnya sendi sudah berada pada posisi 90^o. Namun kenyataannya sendi masih berada pada posisi 80^o. Setelah itu sendi akan terus berusaha menuju posisi acuan dan diharapkan segera diam sesaat setelah mencapainya. Tetapi pada *linear trajectory planning*, hal tersebut tidak akan bisa tercapai dikarenakan tidak mungkin kecepatan sendi langsung berubah menjadi nol di posisi acuan. Sehingga sudah pasti akan terjadi *error* di sekitar posisi acuan yang mengakibatkan waktu tempuh akan menjadi lebih lama lagi.



Gambar 4.8 Pergerakan sendi pada *cubic trajectory planning*

Berbeda pada *cubic trajectory planning*, ketidak-kontinuan kecepatan tidak terjadi karena memang sendi direncanakan untuk bergerak dengan kecepatan awal dan kecepatan akhir sebesar nol. Namun dari gambar 4.8 terlihat bahwa sendi tetap tidak bisa mengikuti *trajectory* acuan di awal pergerakan. Hal ini dikarenakan pada *cubic trajectory planning* ternyata tetap terjadi ketidak-kontinuan di awal dan akhir posisi sendi, namun bukan pada kecepatan melainkan pada percepatannya. Ketidak-kontinuan pada percepatan ini menimbulkan gerakan *impulsive* pada pergerakan awal sendi sehingga mengakibatkan sendi tidak bisa mengikuti *trajectory* acuan. Akan tetapi pada waktu cuplik berikutnya sendi secara perlahan akan mengikuti dan mendekati kurva trajectory yang diberikan. Dan diharapkan di sekitar posisi akhir pergerakan sendi, tidak akan terjadi *error* karena kecepatan akhirnya yang diatur sama dengan nol. Hal ini sebenarnya akan tercapai yaitu terlihat pada perbedaan posisi trajectory acuan dan posisi sebenarnya yang lebih kecil dibandingkan pada *linear trajectory plannig.* Namun dari hasil percobaan diketahui bahwa waktu tempuh untuk mencapai kondisi stabil pada *cubic polynomial* sama bahkan pada t_f = 500 ms lebih lama dibandingkan pada *linear trajectory planning.* Hal tersebut dikarenakan pengaruh momentum manipulator yang mengakibatkan kesulitan dalam pengurangan kecepatan menuju nol.

Dari analisis ini didapatkan kesimpulan bahwa dengan *cubic trajectory planning*, sendi lebih dapat mengikuti *trajectory* acuan walaupun tetap tidak bisa tepat menuju posisi akhir pada waktu acuan yang diberikan dimana hal tersebut harus dicapai pada kendali gerak *Point to Point*. Oleh karena itu dibutuhkan *trajectory planning* dengan polinomial yang lebih tinggi, sehingga dapat memperhatikan batasan percepatan sistem yang menjadi kendala utama pada *cubic trajectory planning. Trajectory planning* yang bisa digunakan adalah *quntic trajectory planning* dengan dua tingkat polinomial lebih tinggi yang memberikan kemampuan untuk memperhatikan batasan pada percepatan awal dan akhir *trajectory*.

4.3.2 *Cubic trajectory planning* pada sendi berbeban

Pada sendi yang ditambahkan beban di ujung lengannya, terlihat perbedaan yang sangat signifikan untuk nilai waktu tempuh (t_f) yang berbeda. Kesamaannya hanya terlihat pada kemampuannya mengikuti *trajectory* di posisi-posisi awal dimana lebih lambat dibandingkan pada keadaan tanpa beban. Ini dimungkinkan karena sendi lebih sulit menggerakkan beban dari posisi diamnya. Sehingga kecepatan untuk menempuh *trajectory* akan berkurang dan akhirnya posisi *trajectory* acuan pun menjadi lebih lambat untuk diikuti.

Perbedaan yang signifikan terjadi pada posisi akhir di setiap waktu tempuh. Pada t_f = 500 ms perbedaan posisi sangat besar, berbeda pada t_f = 1000 ms dimana posisi sendi malah melebihi posisi acuan, dan berbeda juga pada t_f = 3000 ms dimana kurva yang didapat hampir sama dengan kurva pada *cubic trajectory planning* tanpa beban di t_f yang sama. Keadaan ini jelas dipengaruhi oleh beban yang ditambahkan pada sendi, dimana :

a. Pada t_f = 500 ms, beban mengakibatkan sendi bergerak dengan kecepatan yang lebih lambat di awal. Walaupun di waktu berikutnya posisi sendi menjadi lebih dekat dengan posisi acuan, namun tetap saja posisinya di akhir waktu tempuh berbeda jauh jika dibandingkan dengan pada kondisi tanpa beban.
b. Pada t_f = 1000 ms, beban mengakibatkan posisi sendi melebihi trajectory yang diberikan. Hal ini dikarenakan pengaruh momentum yang ditimbulkan beban. Ketika hendak menuju posisi akhir dengan kecepatan

nol, sendi akan mengurangi kecepatannya. Namun karena pengaruh momentum akibat kecepatan sebelumnya yang lebih tinggi, maka penurunan kecepatan pun menjadi lebih lambat. Oleh karena itu sendi malah melebihi posisi acuan sebelum waktu tempuh yang ditetapkan.

c. Pada tf = 3000 ms, pengaruh beban hanya terlihat di awal saja. Selebihnya sendi dapat bergerak dengan kecepatan acuan dengan pengaruh momentum yang kecil di sekitar waktu tempuh. Namun tetap pengaruh beban terlihat pada perbedaan posisi sendi dan posisi acuan di waktu tempuh yang lebih tinggi jika dibandingkan pada kondisi tanpa beban.

Dari hasil analisis ini didapatkan kesimpulan bahwa untuk mencapai posisi yang sesusai dengan posisi acuan di waktu tempuh yang ditetapkan, dapat diatur nilai kecepatan awal dan akhir *trajectory* yang pada percobaan ini diatur pada nilai nol menjadi nilai tertentu. Nilai tersebut bisa didapat dengan melakukan analisis dinamika untuk kecepatan pada posisi awal dan akhir sendi.

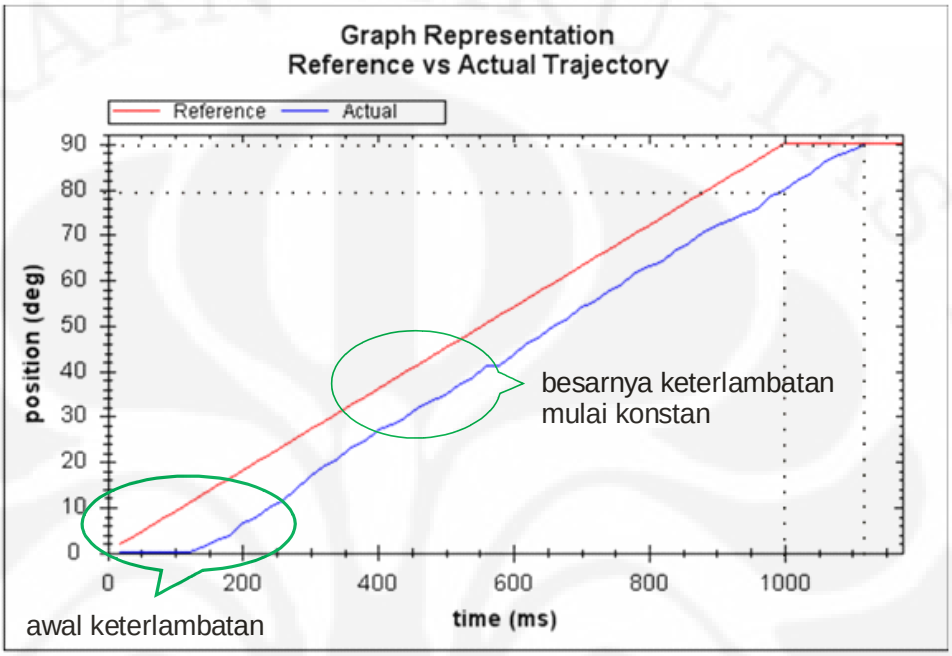
4.3.3 Analisis keterlambatan gerak sendi dalam mengikuti trajectory

Dari gambar 4.4, 4.5 dan 4.6 dapat diketahui bahwa terjadi keterlambatan pada pergerakan sendi mengikuti trajectory yang diinginkan. Hal tersebut diakibatkan komponen dinamis pada aspek fisis sistem yaitu pengaruh percepatan dan torsi maksimum yang bisa diterapkan pada servo sebagai penggerak sendi. Pada lampiran A dilakukan perhitungan pada komponen dinamis sistem, yaitu mengenai percepatan yang mampu diterapkan pada servo akibat pengaruh momen inersia yang diterima sistem yaitu sebesar 0.067 rad/s^2 atau sama dengan 3.86^o/s^2. Dari hasil perhitungan tersebut dapat dilakukan analisis terhadap hasil yang didapat pada setiap pengambilan data sebagai berikut:

a. Pada *linear trajectory planning*

Dari tiga percobaan yang dilakukan pada *linear trajectory planning* diketahui bahwa besarnya keterlambatan hingga waktu tempuh selalu tetap. Perubahan keterlambatan hanya terjadi diawal pergerakan yang diakibatkan oleh besarnya percepatan yang dibutuhkan minimal sebesar 30^o/s^2. Dengan nilai momen inersia yang tetap, maka kebutuhan torsi untuk menggerakkan sistem dengan percepatan tersebut akan meningkat juga. Karena torsi maksimum yang dapat diberikan servo lebih kecil, maka servo tidak akan bisa

melakukan pergerakan dengan percepatan tersebut secara langsung. Hal ini berakibat pada pergerakan servo yang lebih lambat di awal sehingga pasti terjadi keterlambatan dalam mengikuti trajectory.



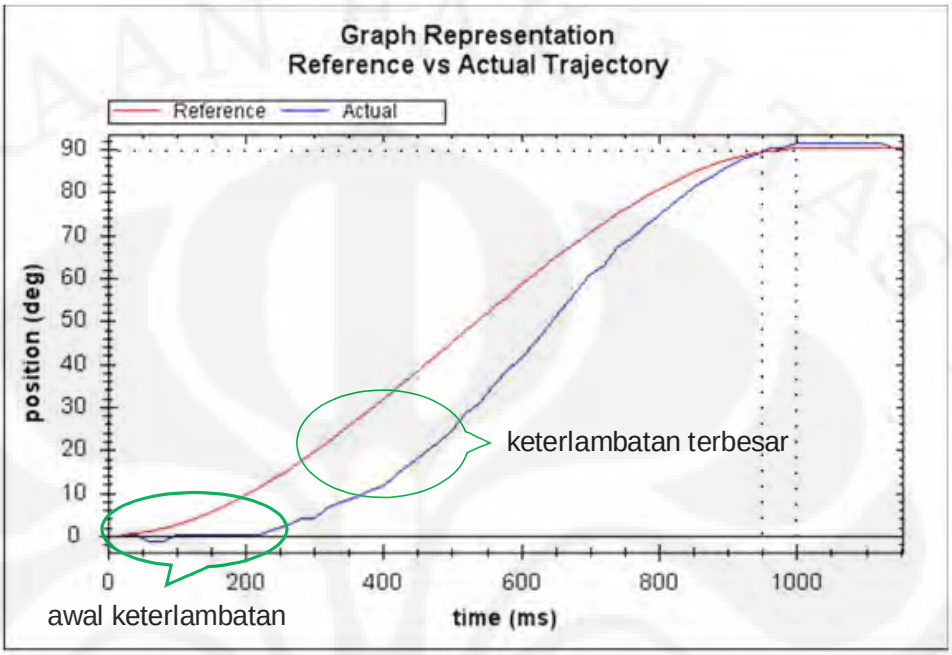
Gambar 4.9 Keterlambatan yang terjadi pada *linear trajectory planning*

Untuk berikutnya setelah sistem mampu bergerak dengan kecepatan yang ditetapkan, maka besarnya percepatan adalah nol. Torsi yang dibutuhkan pun menjadi lebih kecil dan sistem mampu bergerak pada kecepatan konstan. Namun karena tidak ada perubahan kecepatan pada *linear trajectory planning*, maka grafiknya pun akan terus linier dan tidak akan pernah mendekati trajectory yang diberikan. Atau dengan kata lain besarnya keterlambatan pun akan selalu konstan.

b. Pada *cubic trajectory planning*

Berbeda dengan *linear trajectory planning*, besarnya keterlambatan dalam mengikuti trajectory pada *cubic trajectory planning* berubah-ubah. Sistem mengalami perlambatan di awal pergerakan, lalu membesar dan mengecil kembali di akhir pergerakan. Untuk keterlambatan di awal, penyebabnya persis seperti yang terjadi pada *linear trajectory planning*. Namun besarnya

keterlambatan yang terjadi lebih kecil dikarenakan percepatan yang dibutuhkan pun lebih kecil.



Gambar 4.10 Keterlambatan yang terjadi pada *cubic trajectory planning*

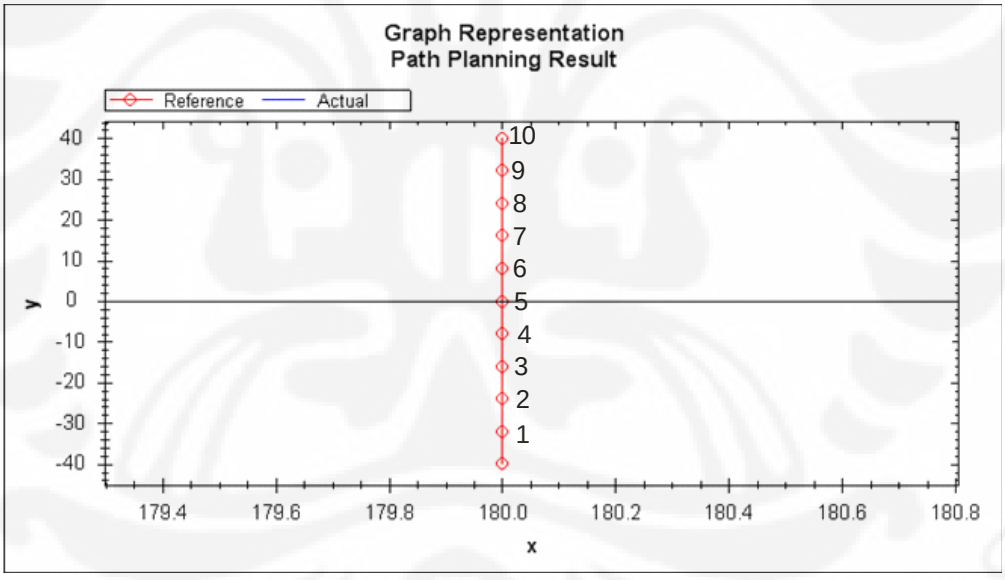
Di pertengahan *trajectory* terjadi keterlambatan yang lebih besar. Dari grafik fungsi trajectory dapat diketahui bahwa besarnya gradien garis yang terbesar terjadi di sekitar pertengahan waktu tempuh. Besarnya gradien merepresentasikan besarnya perubahan posisi, yang berarti dapat merepresentasikan juga besarnya perubahan kecepatan dan percepatan sistem. Dengan nilai gradien yang tinggi, maka perubahan percepatan di sekitar waktu tersebut juga tinggi. Dan bila perubahan percepatan yang dibutuhkan melebihi percepatan yang bisa diterima servo, maka akan terjadi hal yang sama pada awal pergerakan yaitu keterlambatan mengikuti trajectory.

Pada kondisi berbeban seperti yang ditunjukkan pada gambar 4.6, momen inersia yang diterima sistem akan lebih besar. Hal ini berarti besarnya percepatan yang dapat diberikan servo lebih kecil lagi, sedangkan percepatan yang dibutuhkan tetap. Oleh karena itu keterlambatan yang terjadi akan lebih besar dibandingkan pada kondisi tanpa beban.

4.4 Pengujian Sistem untuk Gerak Continuos Path Tracking dengan Cubic Polynomial Trajectory Planning pada Joint Space

Dari pengujian gerak Point to Point dengan menggunakan *cubic trajectory planning* diketahui bahwa pada waktu tempuh acuan, posisi sendi lebih mendekati *trajectory* acuan. Oleh karena itu *cubic trajectory planning* akan lebih baik diterapkan pada gerak *continuous path tracking*, dimana sistem diharapkan dapat mengikuti *trajectory* sedekat mungkin. Dengan itu maka sistem akan semirip mungkin bergerak membentuk lintasan yang diberikan. Berikut akan diuji kehandalan *cubic trajectory planning* dalam mengikuti lintasan acuannya.

Pada pengujian ini digunakan persamaan garis lurus yang menghubungkan titik P_1 (180,-40,0) dan P_2(180,40,0) mm, yang kemudian di interpolasi linier menjadi titik-titik singgah end-effector robot seperti yang ditunjukkan sebelumnya pada tabel 3.3 dan disajikan ulang pada gambar dan tabel berikut :



Gambar 4.11 Koordinat titik-titik singgah hasil Path Planning

Tabel 4.5 Koodinat titik-titik singgah

T	1	2	3	4	5	6	7	8	9	10
P (x)	180	180	180	180	180	180	180	180	180	180
P (y)	-32	-24	-16	-8	0	8	16	24	32	40
P (t)	0	0	0	0	0	0	0	0	0	0

Titik-titik singgah tersebut kemudian di konversi menjadi konfigurasi-konfigurasi sendi menggunakan *inverse kinematics*, hasil konversi tersebut ditunjukkan pada tabel berikut :

Tabel 4.6 Konfigurasi sendi di tiap titik singgah

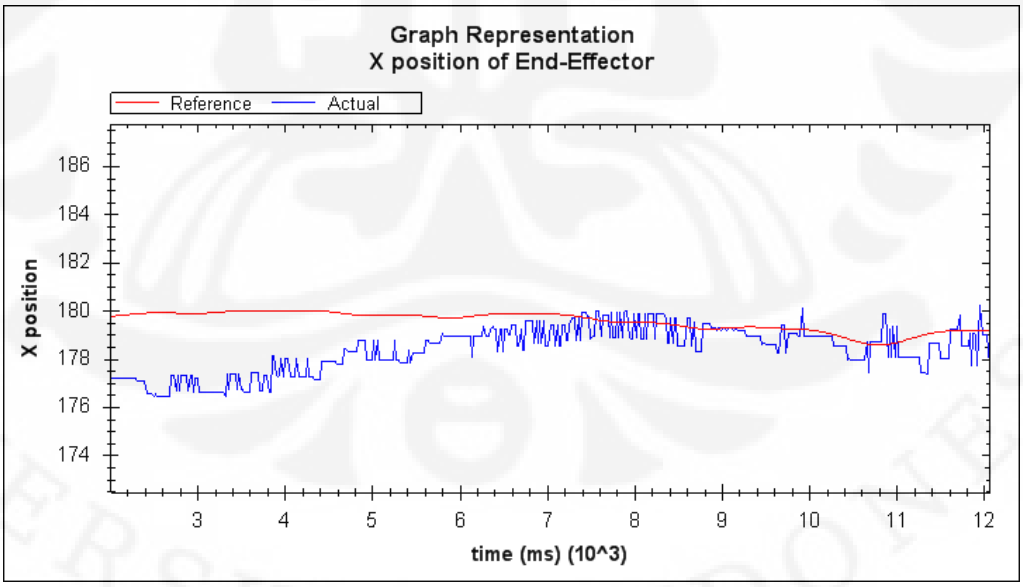
T	1	2	3	4	5	6	7	8	9	10
θ_1	49.81	46.64	43.02	39.08	34.84	30.36	25.69	20.87	15.92	10.86
θ_2	-83.11	-85.64	-87.44	-88.52	-88.88	-88.52	-87.44	-85.64	-83.11	-79.84
θ_3	33.3	39.02	44.42	49.44	54.04	58.16	61.75	64.77	67.19	68.98
d_1	0	0	0	0	0	0	0	0	0	0

4.4.1 Uji gerak *end-effector* dengan perhentian di setiap titik singgah

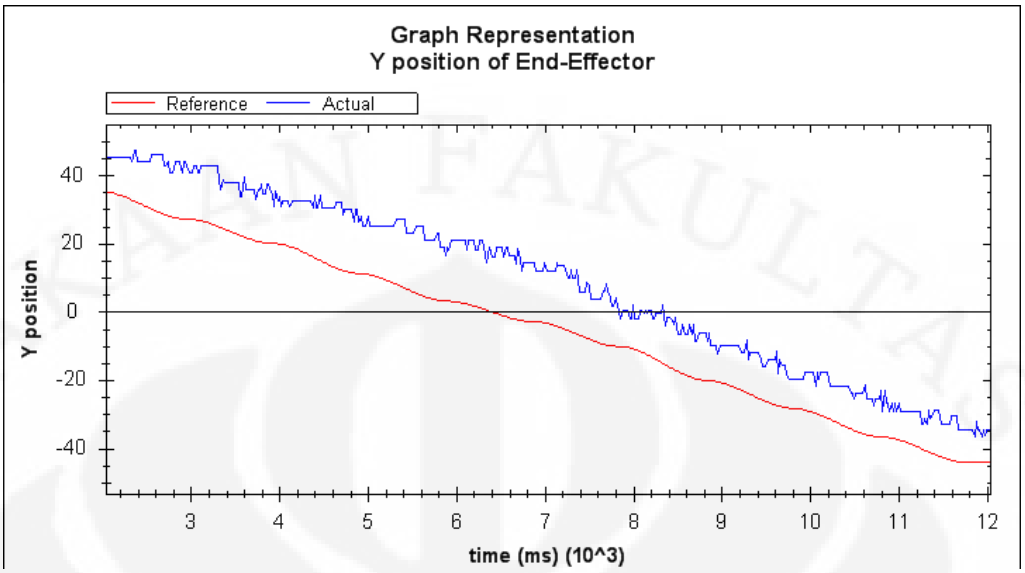
Pengujian path tracking dengan perhentian di setiap titik singgah dilakukan dengan mendefinisikan aturan bahwa kecepatan awal dan akhir setiap titik singgah adalah sama dengan 0 yaitu :

$v_{n-0} = 0$, $v_{n-f} = 0$, dimana n = 1,2,3 ….. 10

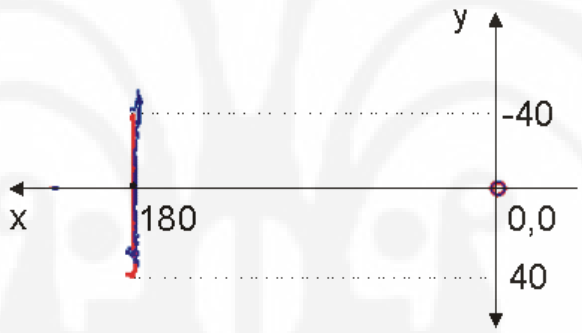
Dengan t_f antara titik singgah sebesar 1000 ms, didapatkan pergerakan *end-effector* yang direpresentasikan oleh gambar berikut :



Gambar 4.12 Pergerakan end-effector pada sumbu X



Gambar 4.13 Pergerakan end-effector pada sumbu Y



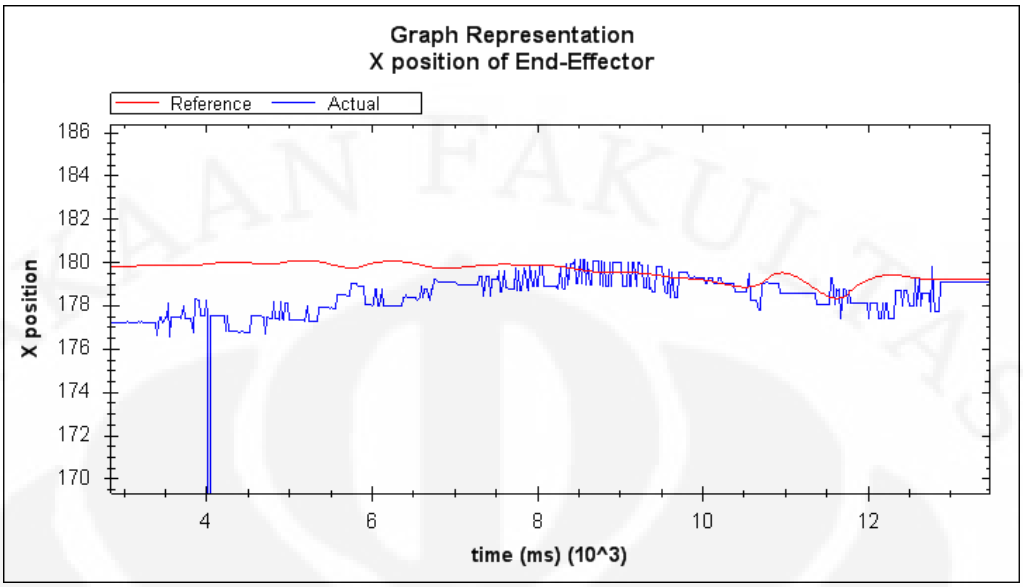
Gambar 4.14 Pergerakan end-effector pada koordinat cartesian

4.4.2 Uji gerak *end-effector* tanpa perhentian di setiap titik singgah

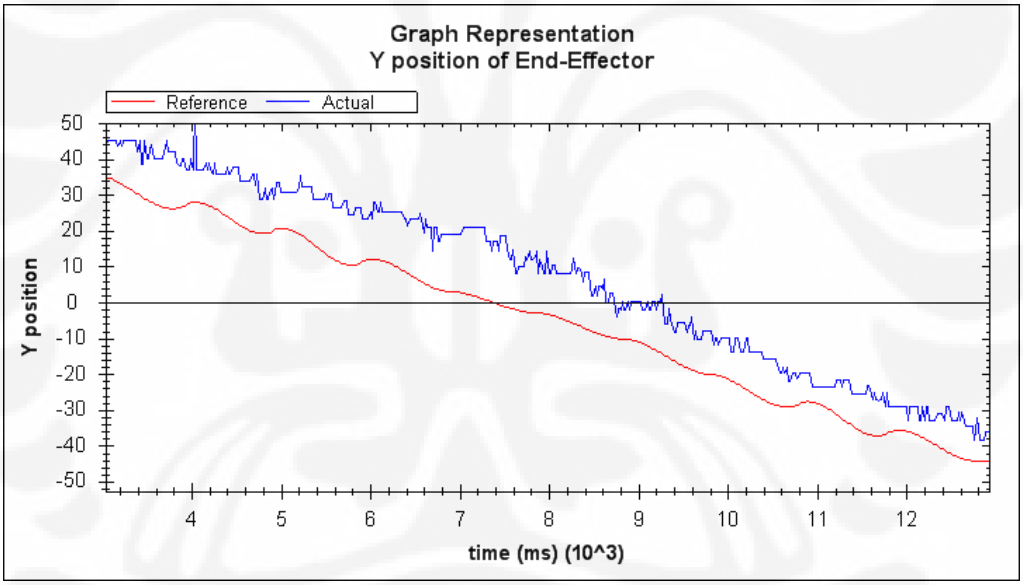
Pengujian path tracking tanpa perhentian di setiap titik singgah dilakukan dengan mendefinisikan aturan bahwa kecepatan awal dan akhir setiap titik singgah adalah sama dengan 0 yaitu :

$v_{1\text{-}0} = 0$, $v_{10\text{-}f} = 0$, $v_{n\text{-}f} = v_{(n+1)\text{-}0}$, dimana n = 1,2,3 ….. 10

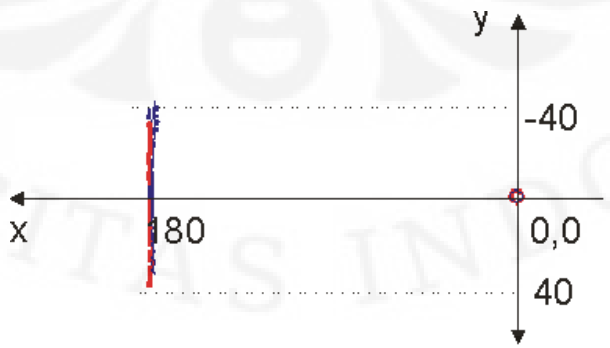
Dengan t_f antara titik singgah sebesar 1000 ms, didapatkan pergerakan *end-effector* yang direpresentasikan oleh gambar berikut :



Gambar 4.15 Pergerakan end-effector pada sumbu X



Gambar 4.16 Pergerakan end-effector pada sumbu Y



Gambar 4.17 Pergerakan end-effector pada koordinat cartesian

4.5 Analisis hasil Pengujian Sistem Gerak Continuos Path Tracking

Pada dua pengujian gerak *continuous path tracking* ini didapatkan hasil yang cukup baik dikarenakan posisi x dan y end-effector hasil pengukuran berbanding lurus dengan posisi x dan y *end-effector* acuan. Adapun perbedaan antara dua acuan yang digunakan adalah pada ada atau tidaknya perhentian di setiap titik singgah. Seperti yang telah dijelaskan sebelumnya, dengan perhentian di setiap titik singgah maka cycle time akan menjadi lebih lama. Oleh karena itu akan jarang sekali dijumpai sistem gerak dengan perhentian tersebut.

Kesulitan yang akan dihadapi untuk gerak path tracking tanpa perhentian adalah pada penentuan kecepatan awal dan akhir di setiap titik singgah. Ada beberapa pendekatan yang bisa dilakukan, salah satunya dengan aturan yang digunakan pada percobaan. Sebenarnya aturan tersebut tidak terlalu baik karena akan memakan waktu lama untuk *trial – error*. Namun memang hanya cara inilah yang bisa digunakan jika *trajectory planning* dilakukan pada joint space. Berbeda dengan pada cartesian space, penentuan nilai kecepatan tersebut bisa didapatkan dengan perhitungan dinamika robot

Dengan melihat gambar pergerakan end-effector pada sumbu- X dan sumbu-Y, terjadi banyaknya *noise* pada kurva posisi *end-effector* yang terjadi sebenarnya. Hal tersebut diakibatkan oleh beberapa faktor, diantaranya :

a. Tingkat akurasi pembacaan sensor oleh ADC yang kecil, yaitu hanya 1.4^{o}/bit saja.
b. Kuantisasi ADC yang cukup lebar, sehingga pada pembacaan posisi potensiometer tertentu kadang didapat dua nilai ADC berbeda secara bergantian terus menerus.
c. Linierisasi yang dilakukan untuk pembacaan potensiometer hanya menggunakan nilai ADC untuk posisi potensiometer di sudut -90^{o}, $+90^{o}$, dan 0^{o}. Linearisasi yang lebih baik sebenarnya bisa dilakukan dengan least square.
d. Resolusi servo yang hanya sebesar 0.289^{o}, sehingga tidak akan terlalu baik untuk pergerakan-pergerakan sendi yang harus lebih kecil dari nilai resolusi servo yang digunakan..

BAB 5
KESIMPULAN

1. Dalam pergerakan robot manipulator, dibutuhkan perencanaan trajectory yang memetakan posisi sendi dalam fungsi waktu. Perencanaan trajectory dapat dilakukan baik pada joint space maupun cartesian space.
2. *Linear trajectory planning* tidak baik digunakan pada perubahan kecepatan pergerakan sendi robot manipulator karena terdapat ketidak-kontinuan kecepatan di path points. Kesalahan posisi di waktu tempuh 3000 ms adalah sebesar 4.4%, dan kesalahan tersebut semakin besar dengan semakin kecilnya waktu tempuh.
3. *Cubic polynomial trajectory planning* juga tidak terlalu baik digunakan pada pergerakan sendi robot manipulator karena terdapat ketidak-kontinuan percepatan di path points. Hal ini menyebabkan kemampuan sendi mengikuti trajectory kurang bagus. Kesalahan posisi di waktu tempuh 3000 ms sebesar 2.2%, dan kesalahan tersebut semakin besar dengan semakin kecilnya waktu tempuh walaupun perbandingannya tidak sebesar pada *linear trajectory planning*.
4. Pada keadaan sendi berbeban, momen gaya manipulator berpengaruh pada semakin lambatnya kemampuan sendi bergerak dari posisi awalnya. Sedangkan momentum manipulator berpengaruh pada kesulitan sendi mengurangi percepatannya.
5. *Cubic polynomial trajectory planning* dapat digunakan untuk gerakan *continuous path tracking*, walau bagaimanapun *trajectory planningnya* tetap saja tidak akan bisa secara tepat mengikuti garis lurus.
6. Penggunaan c*ubic polynomial trajectory planning* pada manipulator untuk mengikuti garis lurus menghasilkan kurva posisi end-effector yang cukup lurus, baik ketika *trajectory planning* pada titik singgah dilakukan dengan perhentian maupun secara kontinu.
7. Terjadinya *noise* pada kurva posisi end-effector diakibatkan oleh resolusi pembacaan sudut sebesar 1.4^{o}, lebarnya kuantisasi tegangan pada penggunaan ADC 8 bit, resolusi servo sebesar 0.289^{o}, dan metode linearisasi untuk konversi pembacaan ADC ke posisi potensiometer yang sangat sederhana.

8. Analisis dinamika robot diperlukan untuk menentukan kecepatan awal dan akhir pergerakan *trajectory* sendi sehingga kemampuan manipulator mengikuti *trajectory* lebih baik.
9. *Trajectory planning* dengan orde yang lebih tinggi yaitu *quintic trajectory planning* dapat mengatasi ketidak-kontinuan percepatan yang terjadi pada *cubic trajectory planning* karena penambahan batasan percepatan awal dan akhir *trajectory*. Sehingga pergerakan sendi dapat lebih baik dalam mengikuti *trajectory* yang diberikan.

DAFTAR REFERENSI

[1] Spong, Mark W., Hutchinson, Seth, & Vidyasagar,M. R*obot modeling and control.* New York : JOHN WILEY & SONS,INC.

[2] Pitowarno, Endra.(2006). R*obotika, desain, kontrol dan kecerdasan buatan.* Yogyakarta : Penerbit Andi

[3] Society of Robot.(2009). *Actuators - servos.* Desember 24, 2009. http://www.societyofrobots.com/actuators_servos.shtml

[4] McComb, Gordon.(2001).T*he robot builder's bonanza* (2nd Ed). New York : McGraw-Hill

[5] Nakamura, M., Goto, S., & Kyura N. (2004). M*echatronic servo system control.* Germany : Springer

[6] Lynxmotion, Inc. (2009). *Arm assembly guides.* September 25,2009 http://www.lynxmotion.com/ViewPage.aspx?ContentCode=assem01&CategoryID=19

[7] Prakoso, Ian Agung. Rancang bangun robot permainan catur berbasis kamera. *Jurnal Tugas Akhir Jurusan Teknik Elektro FTI-ITS*

[8] Afandi, Moh. Imam. Simulasi pengujian trajectory planning pada robot lengan Anthropomorphic. *Pusat penelitian kalibrasi, instrumentasi dan metrologi –lipi.*

A.1 Perhitungan Aspek fisik pada manipulator

A.1.1 Komponen statis pada manipulator

Statika menjelaskan mengenai bagaimana karakteristik sebuah sistem mekanis ketika semuanya benar-benar dalam keadaan diam. Ini adalah perhitungan paling mendasar yang secara matematis juga tidak cukup rumit untuk diselesaikan. Yang perlu dimengerti dan diselesaikan adalah bagaimana membentuk suatu persamaan dari bagian-bagian mekanis yang digunakan.

Hukum III Newton menjelaskan bahwa untuk setiap gaya akan ada gaya berlawanan yang besarnya sama. Jika terdapat beban pada aktuator sebesar 5 kg, maka aktuator pun harus dituntut untuk bisa mengangkat minimal beban seberat 5 kg tersebut. Hal tersebut masih sederhana dan akan lebih rumit lagi jika kita mempertimbangkan hal lainnya seperti gesekan pada sendi, multi aktuator, tingkat efisiensi, dan distribusi beban yang tidak merata.

Berikut beberapa hal yang berhubungan erat dengan kinematika robot :

a. Momen Gaya

Momen gaya merupakan perhitungan yang paling berguna pada kinematika robot. Persamaan dasar momen gaya (τ) adalah sama dengan gaya (F) dikali jarak lengan dimana gaya tersebut sedang dilakukan (l).

$$\tau = F \,.\, l = m \,.g \,.l$$

Nilai dari momen gaya sebenanya adalah besar torsi yang diperlukan. Nilai referensi ini digunakan untuk menentukan rating minimal aktuator yang dibutuhkan sendi untuk menggerakkan lengan saat mengangkat beban.

b. Gesekan

Gaya gesek dihitung dengan mengalikan gaya tegak lurus yang berlaku pada beban dengan koefisien gesek antara beban dengan permukaan gerak.

$$f = N \, x \, \mu$$

Dengan mengetahui besarnya gaya gesek, maka akan didapat referensi nilai gaya yang dibutuhkan untuk melawan pengaruh gesekan

tadi sehingga dapat mulai menggerakkan beban. Namun dalam aplikasinya perhitungan gaya gesek (*f*) merupakan hal yang tidak mudah karena terdapat beberapa situasi yang sulit untuk diperhitungkan seperti tegangan permukaan, kelembaban, dll. Secara sederhana hal pertama yang harus diperhatikan adalah apa yang disebut koefisien gesek (μ).

A.1.2 Perhitungan komponen statis pada manipulator

Berikut adalah data panjang serta bobot motor dan lengan pada setiap *join* dan *link* pada robot manipulator ini :

α_1 = 6 cm, α_2 = 8 cm, α_3 = 6 cm, α_4 = 7.5 cm (α_n = panjang link ke - n)

m_{s1} = 0 gr, m_{s2} = 30 gr, m_{s3} = 30 gr, m_{s4} = 30 gr (m_{jn} = beban join ke - n), massa sendi 1 = 0 karena tertempel pada tubuh robot.

m_{a1} = 100 gr, m_{a2} = 30 gr, m_{a3} = 100 gr, m_{a4} = 50 gr (m_{an} = beban link ke - n)

m_b = masa *wrist + end-effector* = 150 gr

a. Perhitungan momen gaya pada setiap sendi manipulator

Dari data diatas dapat dihitung momen gaya disetiap sendi sebagai berikut :

Sendi 1 (tanpa beban)

$$\tau 1 = \frac{\alpha_1}{2} * m_{a1} + \alpha_1 * m_{s2} + (\alpha_1 + \frac{\alpha_2}{2}) * m_{a2} + (\alpha_1 + \alpha_2) * m_{s3} + (\alpha_1 + \alpha_2 + \frac{\alpha_3}{2}) * m_{a3} + (\alpha_1 + \alpha_2 + \alpha_3) * m_{s4} + (\alpha_1 + \alpha_2 + \alpha_3 + \frac{\alpha_4}{2}) * m_{a4} + (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4) * mb$$

$$\tau 1 = \frac{6}{2} * 100 + 6 * 30 + (6 + \frac{8}{2}) * 30 + (6 + 8) * 30 + (6 + 8 + \frac{6}{2}) * 100 + (6 + 8 + 6) * 30 + (6 + 8 + 6 + \frac{7.5}{2}) * 50 + (6 + 8 + 6 + 7.5) * 150$$

$$\tau 1 = 300 + 180 + 300 + 420 + 1700 + 600 + 1200 + 4125$$

$$\tau 1 = 8825 \text{ gr.cm}$$

Hasil perhitungan diatas menunjukkan bahwa untuk sendi 1 dibutuhkan motor dengan torsi diatas 8.825 kgcm. Servo GWS 2BBMG

memiliki rating torsi sebesar 5.4 kgcm. Sehingga jelas bahwa untuk sendi 1 tidak bisa digunakan motor servo melainkan motor DC.

Sendi 2, 3 dan 4 (tanpa beban)

Pada ketiga sendi ini manipulator hanya akan selalu bergerak rotasi pada sumbu x-y sehingga tidak akan dipengaruhi oleh gravitasi. Perhitungan komponen statisnya tidak diperlukan untuk menentukan besar torsi yang diperlukan motor servo menggerakkan lengan. Namun digunakan pada saat desain mekanik lengan agar sambungan sendi ke lengan kokoh (tidak patah).

Hal ini menjadi fokus perhatian utama yang cukup menyulitkan saat perancangan mekanik karena resiko yang dapat dihadapi adalah patahnya gear dari motor servo atau patahnya sambungan antar sendi dan lengan. Metoda penyambungan sendi dengan lengan yang baik sangat dibutuhkan disini karena selain dapat menghindari resiko tadi, pengaruh dari gesekan pun nantinya tidak akan terlalu besar yang artinya torsi yang dibutuhkan servo menggerakkan lengan menjadi lebih kecil juga.

b. Perhitungan gesekan pada sambungan sendi dan lengan

Pengaruh gaya gesek pada sambungan sendi dan lengan mekanik robot manipulator sebenarnya tidak terlalu signifikan, terutama jika didapatkan desain sambungan antara sendi dan lengan yang efektif. Yang paling perlu diperhatikan adalah pada bagian *end-effector* yaitu gripper pada saat mengangkat beban. Itu pun jika memang harus di desain sehingga beban benar-benar bisa ditekan erat dengan hanya mengandalkan pengaruh dari koefisien gesek antara gripper dengan beban. Namun jika aplikasinya memungkinkan desain gripper yang bisa memegang beban dengan desain tertentu juga, maka gesekan tidak akan berpengaruh lagi dan akhirnya yang perlu diperhatikan hanyalah momen gayanya saja.

(lanjutan)

A.1.3 Komponen Dinamis Manipulator

Dinamika adalah studi tentang struktur mekanik selama periode waktu sedangkan statika adalah studi tentang struktur pada satu waktu. Pada dasarnya statika mempelajari hal-hal yang tidak bergerak, sementara dinamika memperlajari hal-hal yang bergerak. Statika berkaitan erat dengan momen gaya, gaya, tegangan, torsi, tekanan, dll, sedangakan dinamika berkaitan dengan perpindahan, kecepatan, percepatan, momentum, dll.

a. Perpindahan dan Kecepatan

Fokus penting pada studi dinamika salah satunya adalah pada bagaimana merancang sebuah robot untuk berjalan di kecepatan yang ditetapkan. Ada dua faktor yang mempengaruhi kecepatan gerak robot tersebut yaitu kecepatan sudut dan torsi motor yang dibutuhkan. Dua faktor tersebut saling mempengaruhi satu sama lain dan tujuan akhirnya adalah bagaimana sendi tersebut dapat menggerakkan lengan dengan kecepatan sudut yang ditentukan dan torsi yang sesuai.

Setelah kita mendesain robot manipulator tersebut sehingga dapat melakukan gerakan dengan kecepatan (v) yang ditentukan, barulah dengan pasti kita dapat mengendalikannya untuk berpindah dengan jarak tertentu (s) dalam waktu tertentu (t). Dimana dinyatakan pada persamaan berikut : $s = v / t$.

b. Panjang Lengan dan torsi

Dari persamaan momen gaya diketahui bahwa semakin panjang lengan maka akan semakin besar momen gayanya. Hal tersebut juga akan mempengaruhi besar torsi yang dibutuhkan untuk menggerakkan lengan tersebut. Jika momen gaya yang dibutuhkan lengan lebih besar dari pada torsi yang diberikan aktuator, maka lengan robot akan berjalan lebih lambat atau tidak bergerak sama sekali. Sehingga yang harus adalah lakukan adalah membandingkan torsi aktuator, akselerasi robot, dan panjang lengan. Ketiga faktor ini harus seimbang untuk mencapai kecepatan dan posisi yang tepat.

c. Torsi motor dan gaya

Gaya yang cukup besar dibutuhkan untuk menggerakkan beban yang cukup besar atau agar lengan memiliki percepatan yang cukup tinggi. Dengan perhitungan momen gaya dan memperhatikan pengaruh gesekan maka besarnya gaya yang bekerja pada lengan dapat dihitung dan nilai torsi motor minimal pun didapatkan.

d. Percepatan

Ada dua hal yang menjadi pertimbangan mengapa percepatan perlu diperhatikan pada robot manipulator yaitu :

1. Perbedaan gaya pada jalur lintasan lengan robot manipulator

   Berbeda dengan lintasan robot yang bergerak datar tanpa dipengaruhi percepatan gravitasi. Robot manipulator bergerak naik turun dengan kemiringan tertentu ketika akan berpindah ke posisi tertentu. Dibutuhkan percepatan (*g'*) untuk melawan percepatan dari gravitasi bumi (*g*) dimana nilainya dipengaruhi sudut kemiringannya (α) dengan persamaan : g' = g * sin α

2. Mengurangi pengaruh momentum

   Momentum (*p*) adalah besarnya kecepatan (*v*) dikalikan dengan massa beban (*m*). Semakin besar momentum suatu lengan ketika hendak berhenti, maka akan semakin besar juga peluang kesalahan posisi perhentiannya. Hal ini dikarenakan ada sisa gaya yang mendorong lengan untuk bergerak dan kurangnya gaya yang diperlukan untuk mengehentikan motor. Cara yang bisa dilakukan adalah melakukan perlambatan ketika akan menuju posisi perhentian sehingga akan mengurangi besarnya kecepatan dan otomatis juga mengurangi besarnya momentum. Oleh karena itu faktor percepatan menjadi penting.

Merubah nilai percepatan (*a*) berarti merubah nilai gaya (*F*) seperti ditunjukkan pada persamaan berikut : F = m x a. Dan dengan mengetahui besarnya gaya yang dibutuhkan, itu artinya kita bisa menerapkan torsi motor yang sesuai.

(lanjutan)

A.1.4 Perhitungan komponen dinamis Manipulator

Torsi yang didapat pada perhitungan statis adalah torsi minimal yang diperlukan oleh servo untuk mulai menggerakkan lengan. Total torsi yang dibutuhkan aktuator untuk menggerakkan lengan adalah total torsi yang sudah memiliki unsur gaya akibat pengaruh percepatan gerak lengan. Atau dengan kata lain merupakan penjumlahan dari torsi yang dibutuhkan untuk mengatasi momen gaya lengan saat diam dan torsi yang dibutuhkan untuk menggerakkan lengan ketika diberikan percepatan.

Pada sendi pertama, aktuator menggerakkan robot vertikal naik dan turun. Untuk membuat robot naik keatas dengan kecepatan yang sama, aktuator harus memiliki percepatan dengan besar minimal sebesar percepatan gravitasi bumi. Dan untuk memiliki percepatan maka dibutuhkan gaya tambahan pada robot yang besarnya adalah F_1 = m x a, dimana m adalah beban total manipulator yaitu sebesar m = 520 gr. Sehingga dapat dihitung torsi total pada sendi pertama yaitu sebagai berikut :

$\tau_{1\text{-total}} = \tau_1$(hasil perhitungan kinematika) + F_1

$\tau_{1\text{-total}} =$ 8825 gr.cm + 520 gr.cm = 9345 gr.cm

Perhitungan diatas dilalukan pada kondisi tanpa beban tambahan. Oleh karena itu untuk mengantisipasi torsi yang dibutuhkan motor DC ketika ada beban tambahan, maka digunakan reduksi gear sehingga menambah *safety factor* sistem. Penggunaan reduksi gear memang dapat mengurangi kecepatan motor DC, hanya saja seperti yang dijelaskan di awal bab ini bahwa cycle time sistem tidak begitu diperhatikan.

Pada sendi kedua telah kita ketahui bahwa pengaruh kinematika untuk mengetahui besar torsi yang dibutuhkan aktuator tidaklah ada. Hal tersebut bisa dibuktikan dengan melihat karakter lengan yang tetap diam ketika aktuator tidak diberikan catu daya. Torsi dibutuhkan aktuator ketika akan menggerakkan lengan. Torsi tersebut dibutuhkan agar aktuator memiliki percepatan yang bisa membuat lengan memiliki kecepatan untuk bergerak. Torsi ini diberikan pada persamaan

gerak rotasi sebagai berikut : $\tau = I \times \alpha$, dimana I adalah inersia momen putar dan α = percepatan sudut lengan.

Berikut penurunan rumus dan contoh perhitungan untuk sendi kedua :

Sesuai dengan sifat fisik batang kaku, inersia momen putar pada salah satu ujung lain sebagai sumbu putar I adalah 1/3 ml^2.

$$I = \sum \frac{1}{3} m \times r^2$$

Nilai inersia momen putar maksimal dapat diasumsikan dengan menggunakan masa total lengan dan panjang total lengan sebagai berikut :

$$I = \frac{1}{3} * 520 gr * 21.5^2 cm^2 = 80.123 gr.cm^2 = 80,123 kg.cm^2 / rad$$

Dengan mengetahui bahwa torsi yang mampu diberikan oleh servo adalah sebesar τ = 5.4 kg.cm, maka didapat bahwa percepatan sudut minimum yang bisa dicapai sendi 2 sesaat ketika bergerak adalah sebesar $\tau / I = 0.067$ rad/dt^2.

Dengan memperhatikan bahwa sistem hanya memerlukan gerak dengan kecepatan konstan ketika lengan robot telah bergerak, maka percepatan sudut lengan akan sama dengan 0 sehingga torsi yang dibutuhkan sendi 2 saat kecepatan konstan akan sama dengan 0. Hal ini berarti servo yang direncanakan untuk dipakai diawal, dapat digunakan pada sistem.

A.2 Rancang Bangun manipulator

Dengan meninjau perhitungan pada aspek fisik robot sebelumnya, maka perancangan dari manipulator harus benar-benar direncanakan dan diproduksi dengan baik sehingga tidak akan mempengaruhi pemodelan dan kendali robot baik dengan kontrol kinematika maupun kontrol dinamikanya kelak.

Melihat hasil dari perhitungan kinematika ada beberapa aturan desain yang harus diterapkan pada masing-masing sambungan sendi dan lengan setiap bagian manipulator. Berikut adalah detail desain untuk masing-masing sendinya :

(lanjutan)

1. Sendi pada pinggang robot
    a. Penggunaan motor DC dengan reduksi gear untuk meningkatkan torsi akibat beban seluruh sistem pada robot manipulator yang ditanggung oleh sendi ini.
    b. Pemasangan potensiometer linier sebagai sensor posisi.
2. Sendi translasi
    a. Penggunaan motor DC.
    b. Penggunaan katrol untuk merubah motor DC yang bergerak rotasi menjadi sebuah sendi translasi.
    c. Pemasangan encoder pada poros motor DC dibutuhkan karena perputaran poros bisa lebih dari satu putaran.
    d. Pemasangan limit switch di ujung bawah dan atas *workspace* robot diperlukan untuk mereset posisi lengan di ketinggian minimum dan maksimum posisi robot.
    e. Poros sendi translasi dibuat minimal selebar servo pada sendi 2 untuk mengurangi pengaruh dari momentum gerak lengan.
3. Sendi 2, 3 dan 4 / sendi rotasi
    a. Pemasangan servo pada sendi 2 harus sedekat mungkin dengan poros translasi (α_1 dibuat seminimal mungkin).
    b. Penambahan poros di sisi bawah servo pada sendi-sendi ini dibutuhkan untuk membagi beban yang dialami poros atas servo. Poros dibuat dengan menggunakan ball bearing untuk mengurangi gesekan.
    c. Di poros bawah servo pada sendi-sendi ini dipasangkan potensiometer linier sebagai sensor posisi.
4. Lengan 2 (α_2)
    a. Dibuat dalam dua lapisan akrilik mendatar yang dihubungkan oleh akrilik tegak di tengahnya sehingga lebih kokoh.
    b. Lengan lapisan atas dan bawahnya masing-masing ditempelkan pada poros atas dan bawah servo sendi 2 dan 3

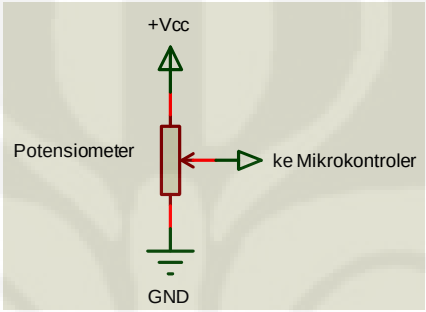
(lanjutan)

5. Lengan 3 (α_3)
   a. Dibuat dalam dua lapisan akrilik mendatar yang dihubungkan oleh akrilik tegak di tengahnya.
   b. Lengan lapisan atas dan bawahnya masing-masing ditempelkan pada tubuh atas dan bawah servo sendi 3 dan 4.
6. Lengan 4 (α_4)
   a. Difungsikan untuk menahan servo sendi ke-5 yang merupakan *wrist* dan servo ke-6 yang merupakan *end-effector* lengan.
   b. Dibuat dalam dua lapisan akrilik mendatar yang dihubungkan oleh akrilik tegak di ujungnya sebagai tempat *wrist* yang merupakan sendi rotasi dengan arah putar vertikal.
   c. Ujung lain dari lengan lapisan atas dan bawahnya ditempelkan pada poros atas dan bawah servo sendi 4
7. *Wrist*
   a. Servo ke-5 yang digunakan sebagai wrist diletakkan pada lengan 4 secara vertikal dan poros atasnya ditempelkan pada gripper sebagai *end-effector*nya.
   b. Poros atas servo ini harus berada sedekat mungkin dengan ujung lengan ke-4.
8. *End-effector*
   a. Berupa *gripper* yang bergerak horizontal membuka dan menutup.
   b. Bentuk *gripper* disesuaikan dengan objek yang akan diangkat sehingga mengurangi pengaruh gesekan.
   c. Koefisien gesek antara *gripper* dengan objek harus sebesar mungkin agar objek tidak terjatuh.
   d. Pemasangan sensor kontak seperti *limit switch* digunakan untuk menandai bahwa *gripper* telah menggenggam objek.

B.1 Rangkaian Elektrik Robot

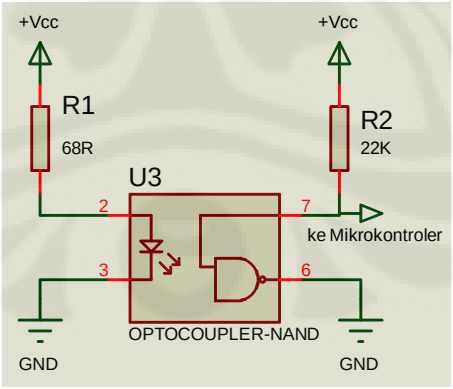
B.1.1 Rangkaian *voltage divider* pada potensiometer

Penggunaan *potensiometer* untuk pengontrolan posisi sangat praktis karena hanya cukup merangkainya sebagai *voltage divider*. Berikut adalah skematik rangkaian *voltage divider* pada potensiometer.



B.1.2 Rangkaian *opto-coupler* pada encoder

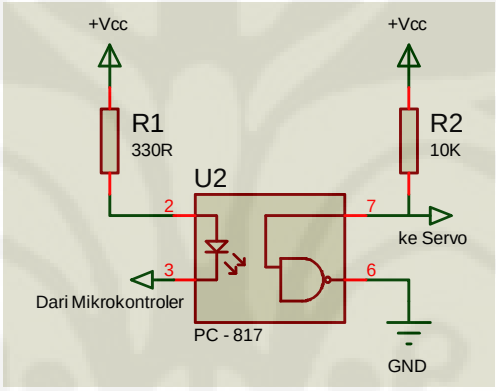
Incremental rotary encoder yang digunakan merupakan produksi sendiri. Rangkaian pendeteksi celah dari cangkram / *shaft* encoder dibuat menggunakan opto-coupler berbentuk U yang banyak tersedia di pasaran. Berikut adalah skematik lengkap dari rangkaian pendeteksi *shaft* encoder tersebut :



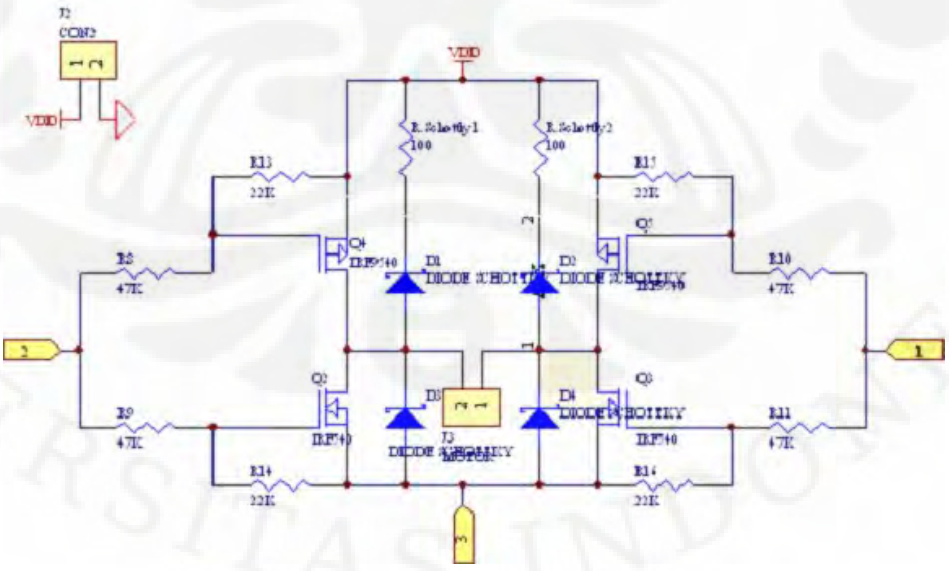
B.1.3 Rangkaian *amplifier* servo

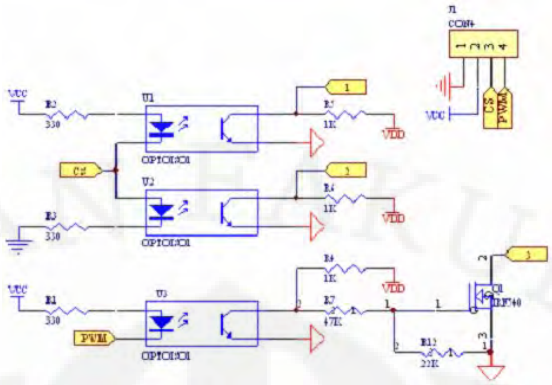
Sebuah servo biasanya memiliki rating arus yang cukup tinggi, sehingga tidak dapat dikendalikan oleh sinyal keluaran langsung dari mikrokontroler. Diperlukan rangkaian *amplifier* arus yang berfungsi untuk meningkatkan arus

keluaran mikrokontroler sehingga dapat digunakan untuk mengendalikan servo. Rangkaian *amplifier* tersebut dibuat menggunakan sebuah transistor NPN. Agar didapatkan fungsi lain yaitu sebagai rangkaian pemisah antara catu daya servo dan mikrokontroler, maka digunakan IC *optocoupler* dengan tipe PC817 yang merupakan sebuah phototransistor dengan arus basis berupa cahaya yang dihasilkan LED infra merah. Berikut adalah skematik lengkap dari rangkaian pendeteksi *shaft* encoder tersebut :



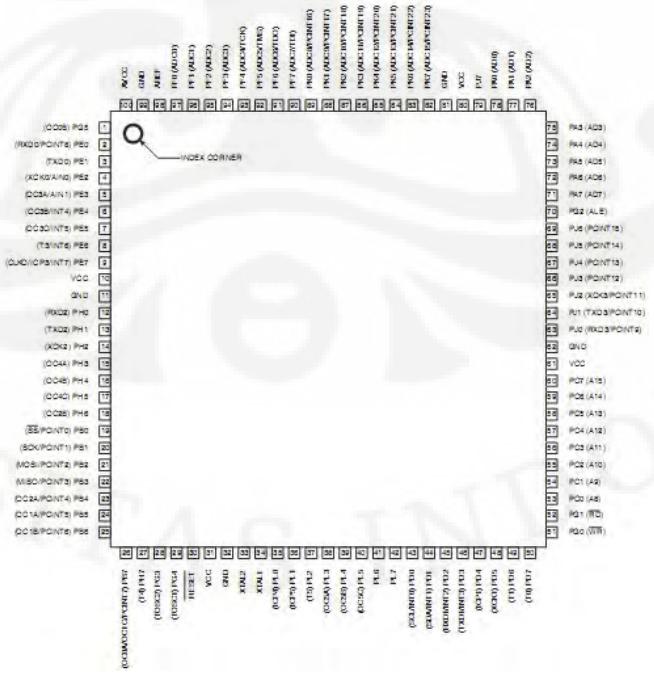
B.1.4 Rangkaian MOSFET H-Bridge





B.2 System Minimum Atmega2560

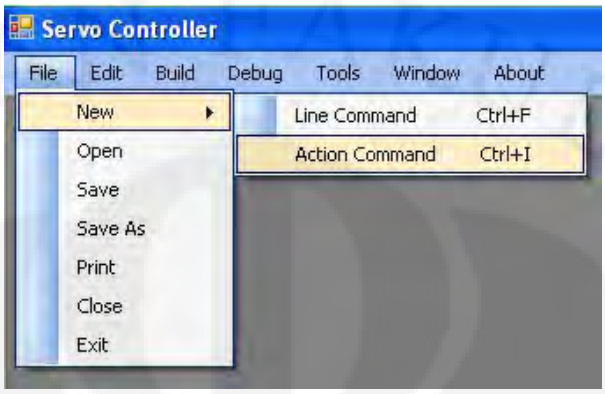
AVR merupakan seri mikrokontroler CMOS 8-bit buatan Atmel. Mikrokontroler AVR ini memiliki arsitektur RISC (*Reduce Instruction Set Computing*) 8 bit, di mana semua instruksi dikemas dalam kode 16-bit (16 bits word) dan sebagian besar instruksi dieksekusi dalam 1 (satu) siklus clock. AVR mempunyai 32 *general-purpose register*, fleksibel *timer/counter* dengan mode *compare*, internal dan eksternal *interrupt*, serial UART, *programmable Watchdog Timer*, *two-wire* (I^2C), ADC, PWM internal dan mode *power saving*. AVR juga mempunyai *In-System Programmable Flash on* dalam sistem menggunakan hubungan serial SPI. ATmega2560 adalah salah satu jenis dari mikrokontroler AVR. ATmega2560 mempunyai *throughput* mendekati 1 MIPS per MHz untuk mengoptimasi komsumsi daya terhadap kecepatan proses.



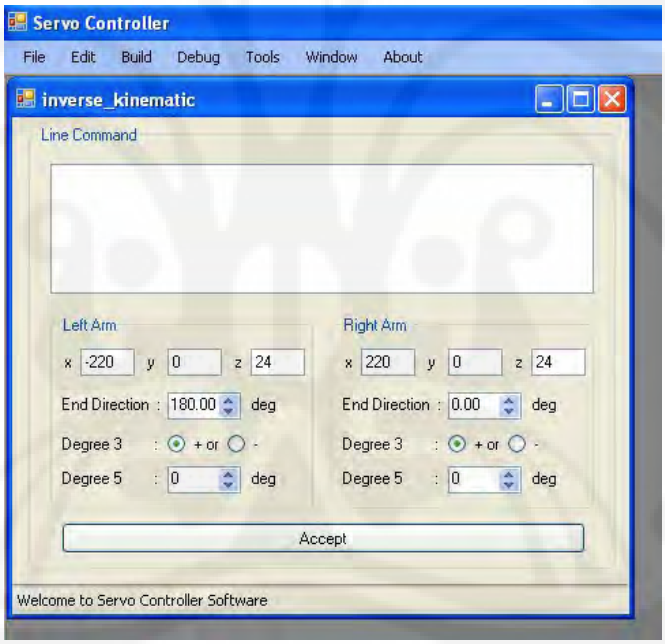
Pada manipulator ini pemilihan Mikrokontroler AVR ATmega2560 didasarkan pada salah satu fiturnya yang memiliki 4 buah timer 16 bit yang masing-masingnya memiliki tiga keluaran dan dapat di set pada mode output compare sehingga dapat menghasilkan total dua belas sinyal PWM yang independen yang dapat diatur perioda dan duty cyclenya.

Selain itu fitur lain yang digunakan adalah USART, dimana dengan fitur ini atmega 2560 memungkinkan untuk berkomunikasi dengan divais lain seperti komputer melalui jalur serial. Fitur lainnya yang tidak kalah penting adalah fitur *interrupt timer overflow* yang akan sangat berguna untuk mengatur kecepatan dari servo motor.

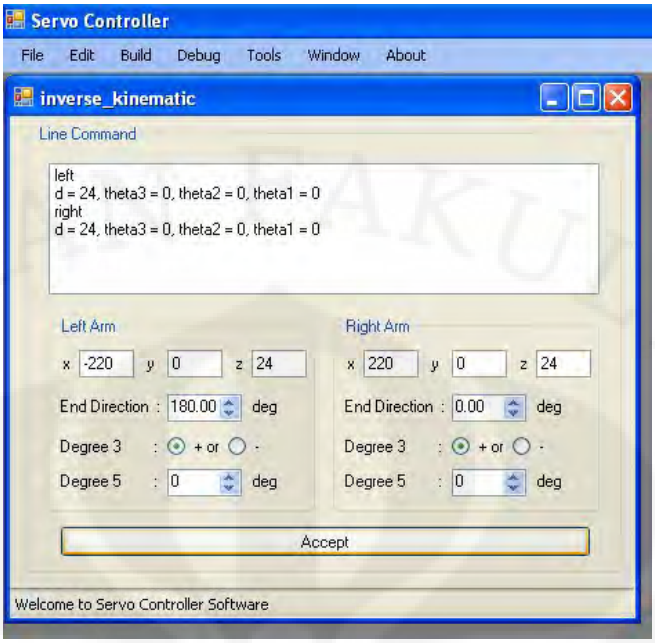
C.1 Program Inverse Kinematics



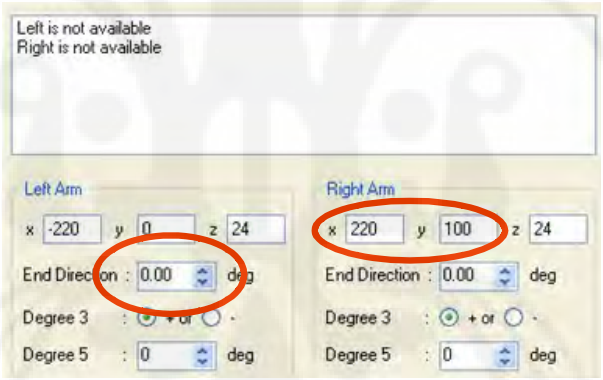
Klik File → New → Action Command, kemudian akan muncul jendela windows baru sebagai berikut :



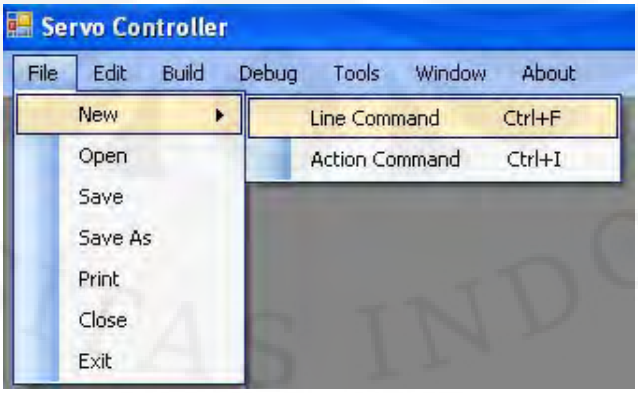
Masukkan koordinat x,y, dan z yang akan dituju oleh *end-efector* masing-masing pada kolom “x”, “y”, dan “”z. Masukkan juga besar sudut orientasi *end-effector* pada kolom “End Direction”, serta tanda sudut sendi rotasi 2 apakah + / - pada kolom “degree 3”. Setelah selesai klik Accept dan didapat hasil sebagai berikut :



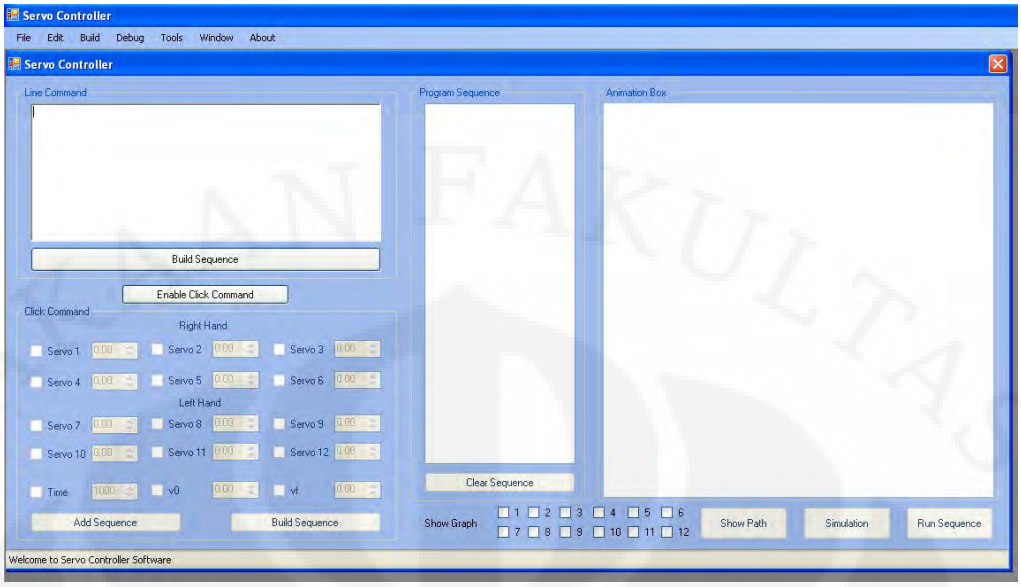
Jika kombinasi nilai x, y, z, sudut orientasi *end-effector* dan tanda sudut sendi rotasi 2 berada di luar workspace robot manipulator, maka akan tampil pesan error seperti berikut :



C.2 Program Forward Kinematics



Klik File → New → Line Command, kemudian akan muncul jendela windows baru sebagai berikut :



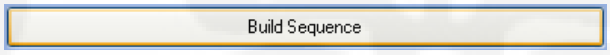
C.2.1 Mode Line Command

Pada mode ini perintah dilakukan dengan menuliskan baris-baris kode sebagai berikut :

s + 500; //mendefinisikan waktu tempuh (t_f) sebesar 500 ms

2 + 10, 3 – 10; //mendefinisikan penambahan 10^o posisi sendi rotasi 2 dan pengurangan 10^o posisi sendi 3 secara sinkron

4 + 90; //mendefinisikan penambahan 90^o hanya untuk posisi sendi rotasi 4 saja.



//klik untuk mengakhiri pengisian urutan konfigurasi pergerakan manipulator

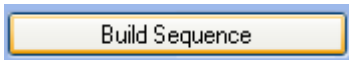
C.2.2 Mode Click Command

// mendefinisikan waktu tempuh (t_f) sebesar 1000 ms

//mendefinisikan perubahan posisi pada sendi rotasi 2 menuju sudut 90^o dan perubahan posisi pada sendi rotasi 3 menuju sudut -90^o secara sinkron

//mendefinisikan kecepatan awal (v_o) sebesar 0^o/s dan kecepatan akhir (v_f) sebesar 20^o/s.

//klik untuk mengisi konfigurasi pergerakan manipulator urutan berikutnya

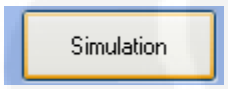


///klik untuk mengakhiri pengisian urutan konfigurasi pergerakan manipulator /klik

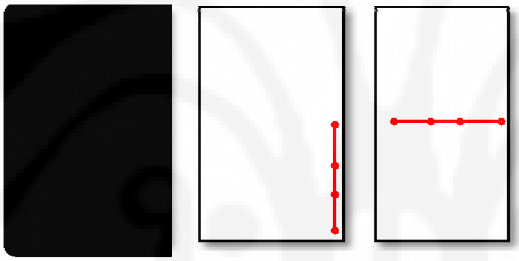
C.2.3 Simulasi Pergerakan

Misalkan urutan pergerakan robot manipulator adalah sebagai berikut :

1. Sendi rotasi 2 = 90^o
2. Sendi rotasi 2 = - 90^o
3. Sendi rotasi 2 = 0^o

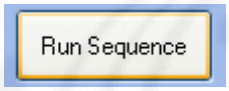


Klik tombol ini maka dengan waktu t_f, v_o, dan v_f yang sebelumnya telah didefinisikan, simulasi akan menunjukkan pergerakan manipulator sesuai dengan *trajectory* yang dihasilkan oleh *cubic trajectory planning.* Untuk lebih jelasnya perhatikan gambar berikut:



(1) (2) (3)

C.2.4 Pengamatan Pergerakan Manipulator secara Real-Time

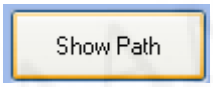


Klik tombol ini maka dengan terlebih dahulu mendefinisikan urutan pergerakan robot manipulator, dan menghubungkan PC dengan rangkaian elektrik serta rancangan mekanik robot manipulator. Pergerakan setiap sendi pada robot dapat diamati secara real-time oleh software.

C.2.5 Grafik trajectory acuan dan posisi aktual masing-masing sendi robot manipulator

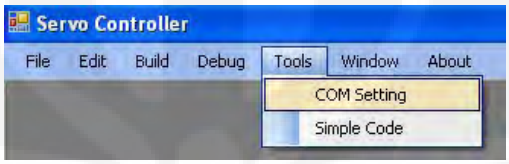
Setelah semua urutan pergerakan robot manipulator selesai dilakukan, maka akan muncul grafik yang memperlihatkan trajectory acuan dan posisi aktual seperti diperlihatkan sebelumnya pada gambar 4.4 – 4.6.

C.2.6 Grafik posisi end-effector pada koordinat cartesian, sumbu-x dan pada sumbu-y

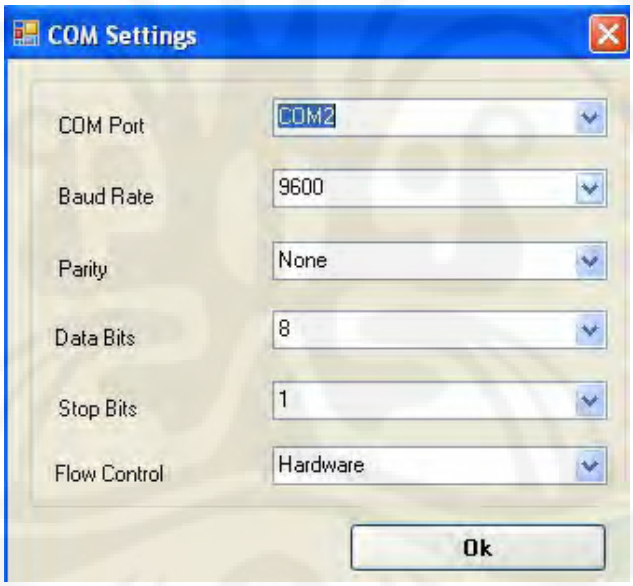
Setelah semua urutan pergerakan robot manipulator selesai dilakukan, klik tombol 

untuk melihat posisi end-effector pada koordian cartesian, pada sumbu-x dan pada sumbu-y seperti yang ditunjukkan masing-masing oleh gambar 4.14, 4.12, dan 4.13.

C.3 Setting Port Terminal Rs232 Untuk Komunikasi Serial



Klik Tools → Com Setting, kemudian akan muncul jendela windows sebagai berikut :



Isi parameter-parameter tersebut, disesuaikan dengan konfigurasi yang diset pada komputer dan mikrokontroler