IMPLEMENTASI PID *CONTROLLER* PADA SIMULASI ROBOT LENGAN MENGGUNAKAN *GRAPHICAL PROGRAMMING*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Dwi Wijayanti
NIM: 125150301111025

PROGRAM TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

PENGESAHAN

Implementasi PID Controller Pada Simulasi Robot Lengan
Menggunakan *Graphical Programming*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Dwi Wijayanti
NIM: 125150301111025

Skripsi ini telah diuji dan dinyatakan lulus pada
21 April 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wijaya Kurniawan, S.T, M.T	Mochammad Hannats Hanafi I., S.ST, MT
NIK: 201201 820125 1 001	NIK: 201405 881229 1 001

Mengetahui
Ketua Program Studi Teknik Informatika

Isa Arwani, S.Kom., M.Sc.
NIP: 19830922 201212 1 003

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 5 Januari 2016

Dwi Wijayanti

NIM: 125150301111025

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga laporan skripsi yang berjudul "Implementasi PID Controller Pada Simulasi Robot Lengan Menggunakan *Graphical Programming*" ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Kedua orang tua serta keluarga yang senantiasa menjadi semangat penulis untuk segera menyelesaikan studi S1, serta yang selalu memberikan nasehat, doa dan restunya demi kesuksesan dan kelancaran studi penulis.
2. Bapak Wijaya Kurniawan dan Bapak Hannats Hanafi selaku dosen pembimbing skripsi yang telah membantu memberikan masukan dan ide dalam proses pengerjaan skripsi.
3. Teman-teman Asisten Laboratorium Sistem Komputer dan Robotika yang selalu memberikan dukungan serta semangat.
4. Teman-teman Sistem Komputer angkatan 2011, 2012 dan 2013 yang telah menjadi keluarga kedua bagi penulis selama menempuh pendidikan di Universitas Brawijaya.

Penulis menyadari bahwa skripsi ini masih memiliki banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Penulis berharap skripsi ini dapat bermanfaat dan membantu penelitian-penelitian selanjutnya yang terkait dengan topik skripsi ini.

Malang, 5 Januari 2016

Penulis

Dwi Wijayanti

ABSTRAK

Di era yang modern ini teknologi robotika bukanlah hal yang asing lagi bagi masyarakat, berbagai aspek kehidupan telah memanfaatkan teknologi robotika guna mempermudah dan menunjang kegiatan yang dilakukan. Salah satunya pada dunia industri, banyak berbagai jenis robot yang telah dimanfaatkan pada dunia industri seperti pada proses *packing* yang memanfaatkan robot lengan. Keakuratan gerak dan minimnya *error* yang terjadi pada pergerakan robot lengan dapat dikontrol dengan PID kontroler, namun proses penentuan konstanta PID masih banyak menggunakan sistem *trial and error* dikarenakan perbedaan *plant* yang mengakibatkan perbedaan nilai PID pada setiap *plant* yang digunakan.

Pada penelitian ini akan dibuat sebuah simulasi implementasi PID kontroler pada simulator robot lengan, robot lengan yang digunakan memiliki 2 *joint* dan 2 *link* dengan seluruh *joint* yang digunakan merupakan *joint revolute*. Pada sistem ini *user* akan memasukkan nilai koordinat yang diinginkan, dari koordinat ini akan ditentukan sudut yang harus dibentuk oleh masing-masing *joint* dengan menggunakan *inverse kinematics*. Dari sudut tersebut akan dikontrol dengan PID lalu diubah menjadi PWM *pulse* untuk dapat disimulasikan dengan Motor DC, dari kecepatan motor DC inilah yang akan diintegralkan untuk menjadi sudut lagi yang akan diolah dengan *forward kinematics* agar diketahui nilai koordinat yang berhasil dicapai oleh sistem.

Kata kunci: PID, *Kinematics*, Robot Lengan, Motor DC, PWM.

repository.ub.ac.id

ABSTRACT

In this modern era, Robotics technology was familiar to the public, various aspects of life have been utilizing Robotics technology to facilitate and support the activities being undertaken. One of them in the industrialized world, many different types of robots that have been exploited in the industrialized world as in the process of packing that makes use of the robot arm. Accuracy of motion and the lack of an error that occurred on the movement of the robotic arm can be controlled with a PID controller, but the process of determining the PID constants using the system are still a lot of trial and error due to the difference in plant which resulted in the difference in the value of the PID on each plant are used.

On this research will study about implementation of PID controller in robot arm using graphical programming, a robot arm used to have two joint and 2 links with the rest of the joint is revolute joint. On these systems, user will enter values for the desired coordinates, these coordinates will be determined from the angle to be set up by each joint using inverse kinematics. From the corner will be controlled by PID and then converted into a PWM pulse to be simulated with a DC Motor, a DC motor speed is integrated over to become the corner again which will be processed with forward so that the coordinate values known kinematics was reached by the systems.

Keywords: PID, Kinematics, Arm Robot, Motor DC, PWM

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL x
DAFTAR GAMBAR xi
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 2
1.3 Tujuan 2
1.4 Manfaat 2
1.5 Batasan masalah 3
1.6 Sistematika penulisan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Tinjauan Pustaka 5
2.1.1 Hmi (*Human Machine Interface*) Dan Pengontrolan Labview Mengunakan Sistem Pid Pada *Prototipe Rotary Stabilizer* Tegangan Satu Fasa 5
2.1.2 Synchronized Robots: A PID Control Project With The LEGO Mindstorm NXT 5
2.1.3 A Labview Based Submarine Depth Control Simulator With PID And Fuzzy Controller 6
2.1.4 Penelitian Saat Ini 6
2.2 Dasar Teori 6
2.2.1 Robot Lengan 7
2.2.2 Simulasi 8
2.2.3 Graphical Programming 8
2.2.4 NI LABVIEW 8
2.2.5 PID Control 13

2.2.6 Kinematics 16
BAB 3 METODOLOGI 18
3.1 Metode Penelitian 18
3.1.1 Alur Metode Penelitian 18
3.1.2 Studi Literatur 19
3.1.3 Analisis Kebutuhan 19
3.1.4 Desain Sistem 20
3.1.5 Implementasi 23
3.1.6 Pengujian dan Analisis Sistem 25
3.1.7 Penarikan Kesimpulan dan Saran 25
BAB 4 PERSYARATAN 26
4.1 Deskripsi Umum 26
4.1.1 Perspektif Sistem 26
4.1.2 Karateristik Pengguna 26
4.1.3 Persyaratan *Hardware* 26
4.1.4 Persyaratan *Software* 26
4.1.5 Lingkungan Operasi Sistem 27
4.1.6 Batasan Sistem 27
4.1.7 Asumsi dan Ketergantungan 27
BAB 5 PERANCANGAN DAN IMPLEMENTASI 28
5.1 Perancangan Sistem 28
5.1.1 Perancangan Perangkat Lunak 28
5.1.2 Perancangan *User Interface* 30
5.1.3 Perancangan Forward Kinematics 30
5.1.4 Perancangan Inverse Kinematics 33
5.1.5 Perancangan Batasan Koordinat 35
5.2 Implementasi Sistem 36
5.2.1 Implementasi *Inverse Kinematics* 36
5.2.2 Implementasi PID kontroler 38
5.2.3 Implementasi PWM (*Pulse Width Modulation*) 40
5.2.4 Implementasi simulator Motor DC 42
5.2.5 Implementasi *Forward Kinematics* 43

5.2.6 Implementasi Desain *Front Panel* 44
5.2.7 Implementasi Pembuatan *Installer* 45
5.2.8 Cara penggunaan aplikasi 52
BAB 6 PENGUJIAN DAN ANALISIS 55
6.1 Sistematika pengujian 55
6.2 Pengujian nilai PID 55
6.2 Pengujian Nilai Koordinat 61
BAB 7 PENUTUP 63
7.1 Kesimpulan 63
7.2 Saran 63
DAFTAR PUSTAKA 64

DAFTAR TABEL

Tabel 2.1 Fungsi aksi kontrol pada PID 14
Tabel 5.1 koordinat dan sudut *Inverse Kinematics* 33
Tabel 5.2 Koordinat dan sudut *Forward Kinematics* 35
Tabel 6.1 Pengujian nilai Kp dengan X = 3, Y = 5, L1 = 4 cm dan L2 = 3 cm 57
Tabel 6.2 Pengujian nilai Ki dengan X = 3, Y = 5, L1 = 4 cm dan L2 = 3 cm 58
Tabel 6.3 Pengujian nilai Kd dengan X = 3, Y = 5, L1 = 4 cm dan L2 = 3 cm 60
Tabel 6.4 Pengujian nilai Koordinat 62

DAFTAR GAMBAR

Gambar 2.1 Robot Lengan 7
Gambar 2.2 Representasi Simbolis *Joint* Robot 8
Gambar 2.3 *Front Panel* Labview 9
Gambar 2.4 Blok Diagram Labview 10
Gambar 2.5 *Tool Pallete* 11
Gambar 2.6 *Control Pallete* 12
Gambar 2.7 *Function Pallete* 12
Gambar 2.8. Tipe Data Labview Dengan Indikasi Perbedaan Warna 13
Gambar 2.9 Blok Diagram PID Kontroler 14
Gambar 2.10 Blok Diagram Aksi Kontrol Proporsional 15
Gambar 2.11 Blok Diagram Aksi Kontrol Integral 15
Gambar 2.12 Blok Diagram Aksi Kontrol Derivative 15
Gambar 2.13 Robot Manipulator 2 *Link* 16
Gambar 3.1 Alur Metode Penelitian 18
Gambar 3.2 Diagram Blok Sistem 21
Gambar 3.3 *Use Case Diagram* 22
Gambar 3.4 Sub Vi *Inverse Kinematics* 23
Gambar 3.5 Sub Vi *Forward Kinematics* 24
Gambar 5.1 *Flowchart* Sistem 29
Gambar 5.2 Perancangan *User Interface* 30
Gambar 5.3 Analisis *Forward Kinematics* 31
Gambar 5.4 Analisis *Inverse Kinematics* 34
Gambar 5.5 Grafik Batas Koordinat untuk L1 = 4 cm dan L2 = 3 cm 36
Gambar 5.6 Implementasi *Inverse Kinematics* teta 2 37
Gambar 5.7 Implementasi Inverse Kinematics teta 1 37
Gambar 5.8 Implementasi PID Kontroler 38
Gambar 5.9 Fungsi *Summation* 39
Gambar 5.10 Fungsi *Gain* 39
Gambar 5.11 Fungsi *Integrator* dan *Derivative* 39
Gambar 5.12 *Build Array* & *Simulation Time Waveform* 40
Gambar 5.13 *Implementasi PWM Pulse* 40

Gambar 5.14 *Servo Angle* 41
Gambar 5.15 *Implementasi Plant Motor DC* 42
Gambar 5.16 Implementasi *Forward Kinematics* 43
Gambar 5.17 Fungsi *Add and Multiply* 44
Gambar 5.18 *Implementasi Desain Front Panel* 44
Gambar 5.19 *Tab New* 45
Gambar 5.20 *Empty Project* 46
Gambar 5.21 *Add File* 46
Gambar 5.22 Hasil *Add File* 47
Gambar 5.23 *New Application* 47
Gambar 5.24 *Application Builder Information* 48
Gambar 5.25 *Tab Properties Application (EXE)* 48
Gambar 5.26 *Tab Source File* 49
Gambar 5.27 Hasil *Proses Building* (EXE) 49
Gambar 5.28 *New Installer* 50
Gambar 5.29 *Tab Properties Installer* 50
Gambar 5.30 Hasil *Build Installer* 50
Gambar 5.31 *Folder* Hasil Pembuatan Aplikasi dan *Installer* 51
Gambar 5.32 *Setup File Installer* 51
Gambar 5.33 Proses *Installing* 51
Gambar 5.34 Hasil *Installing* 52
Gambar 5.35 Aplikasi 2 DOF PID 52
Gambar 5.36 Bagian *Front Panel* 52
Gambar 6.1 Respon Sistem Dengan Kp = 0,5 , Ki = 0 dan Kd = 0 53
Gambar 6.2 Respon Sistem Dengan Kp = 50, Ki = 8 dan Kd = 0 58
Gambar 6.3 Respon Sistem Dengan Kp = 50, Ki = 8 dan Kd = 0,8 59
Gambar 6.4 Respon Sistem Dengan X = 4 dan Y = 4 61

BAB 1 PENDAHULUAN

Bagian utama skripsi terdiri dari beberapa komponen atau bab yang tersusun dengan alur yang logis. Pendahuluan merupakan komponen/bab pertama yang harus menjelaskan apa yang akan dikerjakan dalam skripsi dan mengapa ini perlu dikerjakan.

1.1 Latar belakang

Berdasarkan materi yang telah dibahas oleh *engineering.nyu* dari *Robot Institute of America* definisi robot adalah sebuah manipulator yang dapat diprogram ulang dan disinkronkan antar peralatan dan pekerjaannya untuk melakukan kegiatan yang berhubungan dengan gerak guna menyelesaikan suatu. Di era yang modern ini berbagai bidang kehidupan telah mulai untuk memanfaatkan perkembangan teknologi robotika. Salah satunya adalah bidang industri, dalam proses *packing* ataupun proses pembuatan sebuah produk pada industri berskala besar biasanya telah menerapkan teknologi robotika. Terdapat berbagai jenis dan bentuk dari teknologi robotika yang salah satunya adalah robot berbentuk lengan.

Perkembangan teknologi robotika termasuk robot lengan tidak dapat lepas dari riset-riset yang dilakukan berbagai pihak. Riset-riset ini biasanya dilakukan oleh lembaga riset ataupun institut pendidikan melalui mahasiswanya. Sama halnya dengan Universitas Brawijaya yang juga berusaha memberikan kontribusi bagi dunia perkembangan robotika dengan membentuk sebuah komunitas ROBOTIIK dibawah naungan Fakultas Ilmu Komputer Universitas Brawijaya. Berbagai riset telah dilakukan komunitas ROBOTIIK FILKOM UB termasuk riset untuk robot lengan.

Robot lengan merupakan robot manipulator yang memiliki *joint* (sendi) dan bagian dasar dari robot bisa terpaku diarea kerja atupun terpasang pada rel sehingga memungkinkan robot untuk bergerak dari satu lokasi ke lokasi yang lainnya dalam satu area kerja (eprints.undip, 2015). Sebagai manipulator, robot lengan memiliki aktuator berupa motor servo, akurasi gerak dan minimnya *error* pada pergerakan motor servo merupakan indikator utama keberhasilan suatu sistem robot lengan. Pada dasarnya ketepatan dan akurasi data pada robot lengan dapat dikontrol dengan kontroler PID. PID (Proporsional, Integral dan *Derivative*) adalah aksi kontrol yang digunakan untuk menstabilkan gerakan robot terdiri dari aksi kontrol proporsional yang mempunyai keunggulan *rise time* yang cepat, aksi kontrol integral yang mempunyai keunggulan untuk memperkecil *error* ,dan aksi kontrol derivative yang mempunyai keunggulan untuk memperkecil *error* atau meredam *overshot/undershot* (Rahmat, 2012).

Penentuan nilai PID sebagai kontrol juga tergantung dari jenis *plant* yang digunakan, dimana *plant* merupakan objek fisik yang dikontrol. Misal pada *plant* berupa motor DC, perbedaan jenis motor yang digunakan maka nilai PID yang dibutuhkan untuk akurasi gerak juga berbeda. Hal ini lah yang dapat menimbulkan

kesulitan bagi *user*. Salah satu cara yang sering digunakan *user* untuk menentukan nilai PID yang tepat bagi *plant*-nya adalah dengan menggunakan sistem *trial and error* dimana nilai PID pada *coding* diubah-ubah berdasarkan nilai acak yang dipilih hingga menemukan nilai yang tepat lalu dicoba pada *hardware*, dalam sisi ketepatan gerak dan waktu pengerjaan hal ini dinilai sangat kurang efektif. Berdasarkan latar belakang tersebut diperlukan sebuah sistem simulator penghitungan PID kontroler untuk simulasi gerak motor servo pada robot lengan sehingga dapat diperoleh titik akhir gerak sesuai keinginan.

Pada tugas akhir ini akan dirancang sebuah simulator yang terfokus pada implementasi PID controller untuk gerak motor servo pada robot lengan menggunakan *graphical programming* Labview, dipilihnya *graphical programming* Labview ini bertujuan untuk mempermudah proses *coding* dan membuat sebuah interface yang mudah dimengerti oleh user.

1.2 Rumusan masalah

Berdasarkan latar belakang diatas, maka dapat dirumuskan permasalahan sebagai berikut

1. Bagaimana perancangan kontrol PID pada simulator gerak motor servo pada setiap *joint* di robot lengan ?
2. Bagaimana implementasi kontrol PID pada simulator gerak motor servo disetiap *joint* pada robot lengan menggunakan NI Labview ?
3. Apa pengaruh yang diberikan kontrol PID pada simulator gerak motor servo disetiap *joint* pada robot lengan ?
4. Bagaimana kesesuaian sistem simulator dalam mensimulasikan gerak motor servo pada robot lengan berdasarkan input kontrol PID dari *user* ?

1.3 Tujuan

Penelitian ini bertujuan untuk:

1. Merancang kontrol PID pada simulator gerak motor servo pada setiap *joint* di robot lengan.
2. Mengimplementasikan kontrol PID pada simulator gerak motor servo disetiap *joint* pada robot lengan menggunakan NI Labview
3. Mengetahui pengaruh yang diberikan kontrol PID pada simulator gerak motor servo disetiap *joint* pada robot lengan.
4. Mengetahui tingkat keberhasilan sistem simulator dalam mensimulasikan gerak motor servo pada robot lengan berdasarkan *input* kontrol PID dari *user*

1.4 Manfaat

Manfaat yang diharapkan dari penelitian ini adalah:

1. Bagi Penulis

Dapat meningkatkan pengetahuan serta membantu pemerintah dan masyarakat sebagai bentuk pengaplikasian teori yang telah diperoleh selama perkuliahan.

2. Bagi Peneliti Selanjutnya
   Dengan penelitian ini diharapkan menjadi jembatan dalam berkembangnya pengaplikasian teknologi dan informasi terutama pada bidang robotika.
3. Bagi Masyarakat
   Dengan kehadiran sistem ini diharapkan dapat membantu dalam meningkatkan minat dan ketertarikan akan perkembangan teknologi robotika dengan membantu menyediakan aplikasi penghitungan PID pada simulator robot lengan yang mudah digunakan.
4. Bagi Pemerintah
   Diharapkan dapat membantu pemerintah dalam perkembangan ilmu pengetahuan serta pembangunan negara menuju negara yang melek teknologi melalui riset-riset yang dilakukan masyarakatnya sendiri.

1.5 Batasan masalah

Dikarenakan objek penelitian merupakan robot yang memiliki perbedaan baik *design, hardware* maupun *software* antar satu robot dengan robot yang lainnya maka diperlukan adanya batasan penelitian, diantaranya

1. Menggunakan bahasa pemrograman *Graphical Programming* Labview dari National Instrument.
2. Objek Penelitian adalah sebuah simulator motor servo pada robot lengan 2 DOF (*Degree of Freedom*) karena penghitungan PID pada robot lengan sangat berpengaruh dan mudah dilihat ketika terjadi *error*.
3. Algoritma yang digunakan untuk penghitungan gerak motor servo pada robot lengan adalah *PID Controller.*
4. Sudut motor servo didapat dari integral kecepatan *plant* motor DC
5. *Plant* motor DC yang digunakan adalah Motor DC *Separately Excited* 5kW 220V.
6. *Joint* yang digunakan pada robot lengan terdapat 2 *joint* dan 2 *link* (2 DOF) dengan keseluruhan *joint* merupakan *joint revolute*.

1.6 Sistematika penulisan

Penulisan skematik pembahasan dan penyusunan proposal skripsi dapat diuraikan sebagai berikut:

BAB 1 PENDAHULUAN

BAB 1 Pendahuluan dijelaskan mengenai latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan dari Perancangan sistem alat ukur.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi beberapa dasar teori dan referensi penelitian yang pernah ada yang memiliki tujuan dan perancangan sistem.

BAB 3 METODE PENELITIAN

Bab ini membahas beberapa langkah kerja yang akan dilakukan dalam penelitian, diantaranya alur metodologi penelitian, analisis kebutuhan sistem, perancangan, spesifikasi sistem, dan pengujian sistem.

BAB 4 PERSYARATAN

Bab ini menjelaskan secara rinci "Implementasi PID Controller Pada Simulasi Robot Lengan Menggunakan *Graphical Programming*" yang meliputi deskripsi umum dari sistem, kebutuhan antarmuka sistem, kebutuhan perangkat keras dan lunak, *use case* diagram, kebutuhan fungsional sistem, dan batasan desain sistem.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan perancangan sistem penelitian serta implementasi sistem penelitian berupa implementasi perangkat keras dan perangkat lunak, serta batasan implementasi.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini membahas beberapa langkah kerja dalam melaksanakan pengujian.

BAB 7 PENUTUP

Bab ini membahas tentang kesimpulan dan saran yang diperoleh dari rumusan masalah penelitian dengan melakukan analisis dan pengujian.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian. Jika dibutuhkan sesuai dengan karakteristik penelitiannya dan syarat kecukupan khusus keminatan tertentu, bisa juga terdapat kajian pustaka yang menjelaskan secara umum penelitian terdahulu yang berhubungan dengan topik skripsi dan menunjukkan persamaan dan perbedaan skripsi tersebut terhadap penelitian terdahulu yang dituliskan.

2.1 Tinjauan Pustaka

Tinjauan pustaka berisikan survei serta pembahasan literatur yang memiliki keterkaitan dengan bidang yang sedang diteliti. Studi ini merupakan gambaran singkat dari apa yang telah dipelajari, argumentasi dan biasanya diorganisasikan secara kronologis serta sistematis. Beberapa materi yang diperlukan sebagai referensi dalam merancang sistem ini adalah mengenai labview, kontroler PID, PWM, motor DC dan motor servo. Materi materi tersebut dapat diperoleh dari buku, jurnal, maupun artikel diinternet, berikut beberapa rangkuman artikel terkait dengan penelitian yang akan dilakukan.

2.1.1 HMI (*Human Machine Interface*) Dan Pengontrolan Labview Mengunakan Sistem Pid Pada *Prototipe Rotary Stabilizer* Tegangan 1 Fasa

Pada skripsi yang disusun oleh Rahmat Noviali dari Politeknik Negeri Jakarta telah dibahas tentang sebuah rancangan HMI (*human machine interface*) dengan menggunakan labview untuk *prototype rotary stabilizer* tegangan 1 fasa. Penelitian ini bertujuan untuk mempermudah kendali pada pengontrolan *rotary stabilizer* tegangan 1 fasa. Berdasarkan penelitian yang telah dilakukan diperoleh hasil sebagai berikut

a) Kontroller PID memiliki pengaruh yang cukup besar dalam pengaturan otomasi sistem.
b) Penggunaan HMI dinilai sangat memudahkan baik dalam proses pengujian maupun pengamatan.
c) Respon pada pengujian beban , kontroler PID lebih cepat dibandingkan dengan kontroler manual, sebesar 15 % untuk beban 1 resistif 100 Watt dan 12% untuk beban 2 resistif 200 Watt.

2.1.2 Synchronized Robots: A PID Control Project With The LEGO Mindstorm NXT

Synchronized Robots: A PID Control Project with the LEGO Mindstorm NXT merupakan penelitian yang dilakukan oleh Dr. Chad Eric Davis dari *University of*

Oklahoma. Penelitian ini menitik beratkan pada penggunaan PID controller pada robot *LEGO Mindstorm NXT* untuk pembelajaran PID controller bagi mahasiswa di *University of Oklahoma*. Selain itu dalam penelitian *Synchronized Robots: A PID Control Project with the LEGO Mindstorm NXT* ini digunakan aplikasi LABVIEW programming untuk pembuatan GUI (*graphical user interface*) hal ini dinilai sangat efektif dikarenakan dapat memperudah baik dari segi coding ataupun dari sisi penggunaan aplikasi. Dari penelitian tersebut dapat diperoleh hasil sebagai berikut

1) Sangat membantu mahasiswa dalam memahami PID Control.
2) Memberi mahasiswa rasa percaya diri mereka bisa menerapkan kontrol PID di masa depan.
3) Mampu meningkatkan kemampuan mahasiswa untuk bekerja dalam sebuah tim.
4) mampu meningkatkan kompetensi mahasiswa untuk menggunakan LabVIEW.

2.1.3 A Labview Based Submarine Depth Control Simulator With PID And Fuzzy Controller

Penelitian ini dilakukan oleh Mehmet Ekici, Hakan Kahveci, Adem Sefa Akpinar dari *Karadeniz Technical University, Department of Electrical and Electronics Engineering, Trabzon, Turkey*. Pada penelitian ini dirancang sebuah simulator dari kapal selam dengan menggunakan *graphical programming* Labview National Instrument. Dengan simulator ini pengguna dimungkinkan untuk melihat grafik navigasi sistem serta tingkat stabilitas sistem. Penelitian tersebut menggunakan dua metode yakni PID kontroler dan Fuzzy yang digunakan sebagai metode pengontrolan sistem navigasi kapal selam. Berdasarkan hasil yang diperoleh menunjukkan bahwa metode fuzzy dinilai lebih efektif dalam pengontrolan kapal selam.

2.1.4 Penelitian Saat Ini

Penelitian pada skripsi ini merupakan perkembangan dari penelitian – penelitian yang tercantum pada studi literatur dengan penambahan serta perubahan ruang lingkup penelitian mengikuti perkembangan teknologi . Penelitian dilakukan pada simulator robot lengan 2 DOF dengan mengutamakan faktor ketepatan komputasi nilai PID serta kemudahan dalam penggunaan aplikasi. PID kontroler akan digunakan sebagai penentu besar perubahan sudut motor servo sebagai posisi akhir dari gerak sebuah robot lengan 2 DOF berdasarkan nilai PID yang dimasukkan.

2.2 Dasar Teori

Dasar teori berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian.

2.2.1 Robot Lengan

Berdasarkan deskripsi dari *Robot Institute of America* pada tahun 1979 menyebutkan bahwa robot merupakan sebuah manipulator multifungsi yang dapat diproram ulang yang didesain untuk memindahkan benda, bagian, alat atau penyetaraan *device* dengan variabel *programming*-nya untuk mampu menyelesaikan suatu *task.*

Istilah robot sendiri berasal dari *Czech Robota* yang berarti pekerja, istilah ini pertama kali diperkenalkan oleh Karel Capek pada tahun 1921 (Wakamenta, 2011). Terdapat dua jenis robot yang kini dikenal didunia robotika yaitu jenis robot manipulator dan jenis robot mobile, robot manipulator merupakan robot industri yang biasanya berbentuk seperti lengan manusia memiliki *joint* (sendi) yang biasanya berupa motor sebagai penggerak serta *link* yang merupakan penghubung antar *joint*, bagian dasar robot tertanam pada bidang sehingga robot tidak mampu berpindah tempat namun mampu memindahkan. Sedangkan robot *mobile* merupakan robot yang memiliki ciri khas berupa aktuator yang mampu menggerakkan sebagian atau seluruh bagian tubuh robot sehingga mampu berpindah dari satu titik ke titik tertentu.

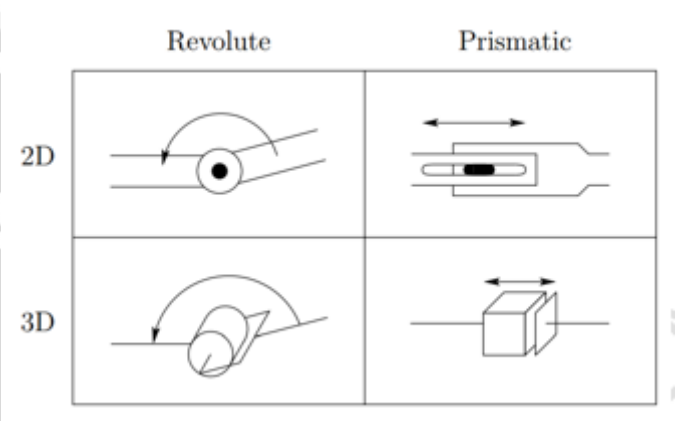
Robot lengan merupakan bagian dari jenis robot manipulator yang hanya bisa memindahkan tanpa bisa berpindah. Akurasi dan ketepatan gerak merupakan indikator utama dalam keberhasilan design sebuah robot lengan. Untuk dapat menghasilkan akurasi gerak maka diperlukan sebuah perhitungan matematis untuk *controlling* dan memperkirakan adanya *error* pada gerakan. Contoh pada robot lengan dapat dilihat pada Gambar 2.1.

Gambar 2.1. Robot Lengan (sumber: eko, 2015)

Pada robot lengan terdapat dua jenis *joint* (sendi) yang dapat digunakan yaitu *joint prismatic* dan *joint revolute*. Perbedaan dari kedua *joint* ini terdapat pada arah geraknya*, joint prismatic* memungkinkan robot untuk melakukan pergerakan memanjang dan memendek sedangkan *joint revolute* memungkinkan robot untuk

melakukan gerakan memutar. Representasi dari *joint* robot dapat dilihat pada Gambar 2.2 Untuk implementasi *joint revolute* biasa digunakan adalah motor elektrik yang terdiri dari 3 jenis yaitu Motor DC, Motor Stepper dan Motor Servo.

Motor DC merupakan motor yang memiliki kecepatan konstan tergantung dari *input voltage* yang diberikan, motor stepper memiliki karakteristik gerakan yang terpatah-patah jenis motor ini dapat ditemua pada *printer* sedangkan motor servo dapat bergerak memutar sesuai besar sudut tertentu. Motor servo sering digunakan untuk *joint revolute* lengan robot yang berdimensi kecil, untuk dimensi yang lebih besar banyak digunakan motor stepper.



Gambar 2.2. Representasi simbolis *joint* robot (sumber: eko, 2015)

2.2.2 Simulasi

Berdasarkan pengertian dari kamus besar bahasa indonesia definisi simulasi adalah metode pelatihan yang meragakan sesuatu dalam bentuk tiruan yang mirip dengan keadaan yang sesungguhnya atau penggambaran suatu sistem atau proses dengan peragaan berupa model statistik (KBBI, 2016).

2.2.3 Graphical Programming

Graphical Programming merupakan salah satu *type* dari bahasa pemrograman selain *text base programming,* perbedaan dari dua jenis type pemrograman ini terdapat pada cara pemrogramannys. *Text base programming* merupakan cara *programming* dengan menggunakan *text* sedangkan *graphical programming* merupakan cara *programming* menggunakan gambar. *Graphical programming* memiliki prinsip *drag and drop,* sehingga cara pemrogramannya dengan menempatkan gambar/symbol dari fungsi program, dan menghubungkannya satu sama lain dengan *wire*.

2.2.4 NI LABVIEW

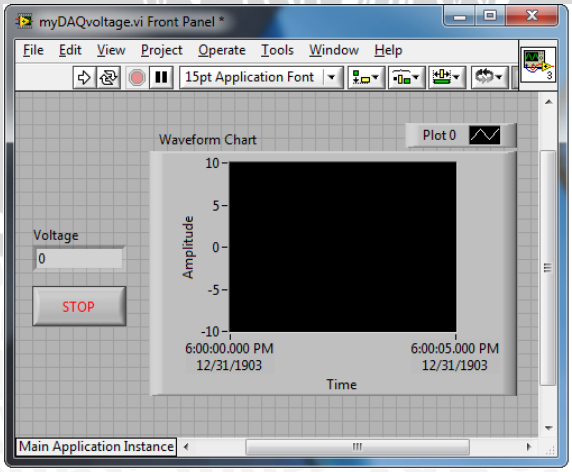
LabVIEW merupakan *software* berbasis *graphical programming* untuk aplikasi pengumpulan, penyimpanan, pengolah, pengelola data dan aotomasi berbasis komputer dengan teknik pemrograman berbasis grafis (G-Software). Program

LabVIEW disebut juga Virtual Instrument (VI), karena instrumen yang digunakan adalah instrumen maya yang mengambil bentuk tampilan seperti instrumen aslinya (instrumen konvensional) seperti *osciloscop, multimeter, XY plotter, gauge indicator, thermometer,* dan sebagainya. LabVIEW juga dirancang untuk dapat melakukan fungsi-fungsi pengendalian proses untuk sistem otomasi industri Virtual Instrument yang dimaksud adalah bahwa instrumen-instrumen ukur yang digunakan merupakan instrumen maya (instrumen yang secara fisik tidak tampak wujud nyatanya sebagai hasil rekayasa perangkat lunak). Software LabVIEW merupakan produk National Instruments telah menyediakan fungsi-fungsi untuk kontrol kontinyu (PID) dan pembuatan panel HMI (*human machine interface*), database, perhitungan matematik dan statistik data hasil pengukuran, pembuatan *technical report* dan publikasi melalui internet (Rahmat, 2012). Dengan semua fasilitas tersebut, implementasi pengukuran dan pengendalian menggunakan komputer menjadi sangat praktis dan sangat dapat dipercaya hasilnya. Program LabVIEW yang disebut VI berisi tiga bagian utama, yaitu

1. *VI Front Panel.*
2. *VI Block Diagram.*
3. *Icon/Connector Pane.*

1. VI front panel

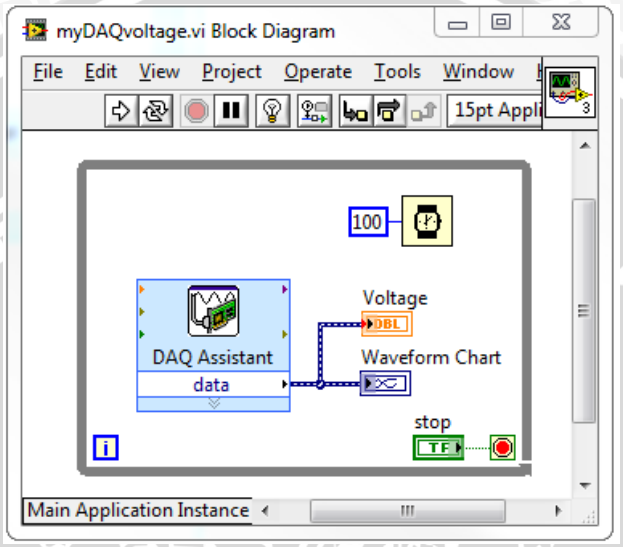
VI *front panel* merupakan bagian dari LABVIEW yang berhubungan langsung dengan *user* atau bisa disebut dengan GUI (*Graphical User Interface*). Pada *front panel* biasanya berisi kontrol (*input* yang dapat diberikan oleh *user*) dan indikator yang digunakan untuk memantau jalannya suatu *system*. Beberapa contoh fungsi kontrol diantaranya *numerical control, knob, pointer slide, dial, switch dan string control*. Sedangkan fungsi-fungsi indikator antara lain *numerical indicator, slide indicator, string indicator, tank, gauge, meter, graph chart* dan *XY graph* (Rahmat, 2012). Kemudahan dalam penggunaan sebuah aplikasi tergantung dari desain dari *front panel*. Salah satu contoh tampilan dari *front panel* dapat dilihat pada Gambar 2.3.



Gambar 2.3. *Front Panel* Labview (sumber: ni.com)

2. Blok Diagram Labview

Blok diagram labview merupakan bagian dimana program disusun (*coding* program), blok diagram ini berisikan instruksi-instruksi berupa gambar (*function pallet*) yang akan disusun dan dihubungkan satu sama lain dengan menggunakan *wire* hingga membentuk algoritma dari sebuah *system.* Setiap *function pallet* disimbolkan dengan *Icon* yang memiliki terminal *input* dan *output* (*Connector pane*). Interkoneksi antar *function pallet* dilakukan dengan menggunakan data *wire*. Contoh dari blok diagram dan *connector pane* dapat dilihat pada Gambar 2.4.

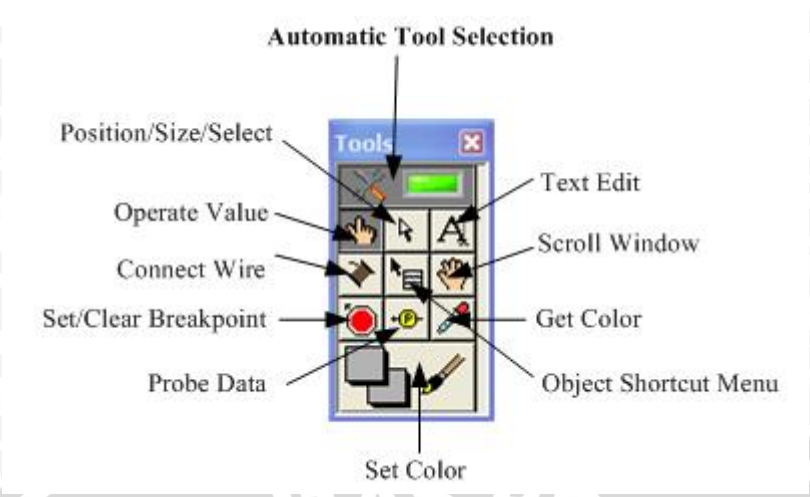


Gambar 2.4. Blok Diagram Labview (sumber: ni.com)

3. Icon/Connector Pane

Untuk dapat membuat sebuah VI (*virtual instrument*) terdapat tiga buah *pallete* yang akan digunakan baik di *front panel* ataupun pada bagian blok diagram. *Pallete* ini berisikan *set tool* dan instruksi dalam bentuk *graphical* yang akan digunakan untuk menyusun sebuah program. Terdapat tiga jenis *pallete* diantaranya *tool pallete, control pallete* dan *function pallete.*

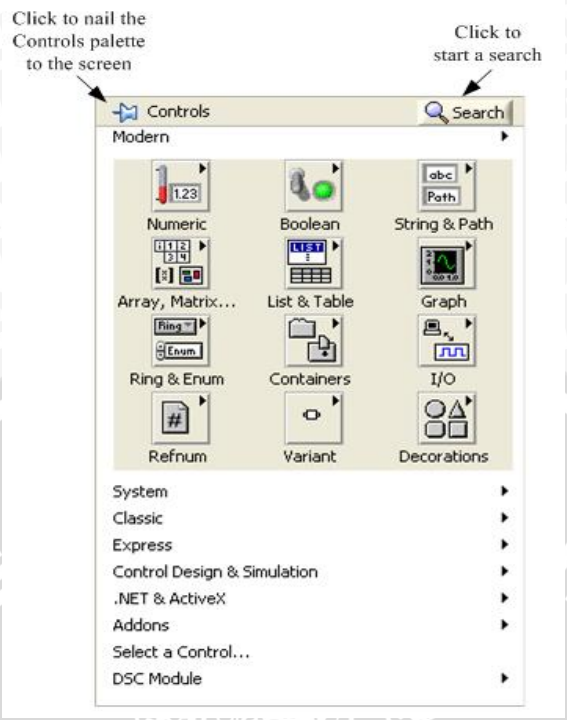
Tool pallete digunakan untuk membuat atau memodifikasi VI baik melalui blok diagram ataupun melalui *front panel*. *Tools palette* ini merupakan jenis operasi special dari kursor *mouse* dengan kata lain, bentuk kursor berbeda untuk jenis pekerjaan berbeda. Untuk menampilkan tools ini dengan cara *Klik View >>Tools Palette* (Rahmat, 2012)**.** gambar dari *tool pallete* dapat dilihat pada Gambar 2.5.



Gambar 2.5. *Tool pallete* (sumber: finn, 2008)

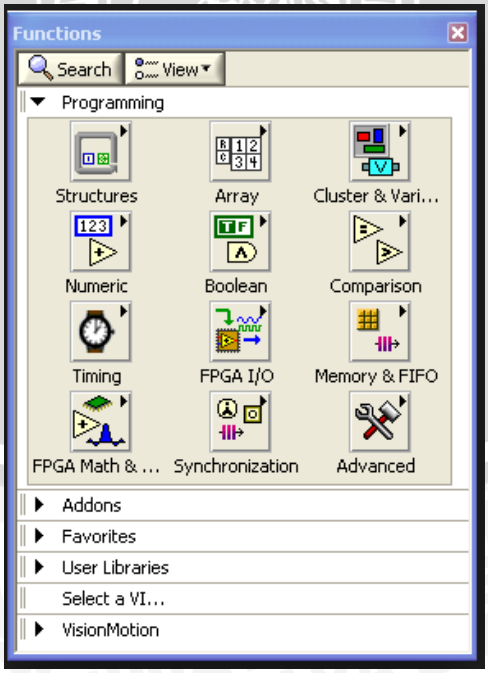
1. *Automatic tool Selection* merupakan indokator kursor *mode*.
2. *Operate Value* untuk merubah nilai dari elemen baik pada *front panel* ataupun pada blok diagram.
3. *Position/size/select*.
4. *Edit text* untuk menambahkan *text*.
5. *Scroll window* untuk melihat halaman yang tersembunyi.
6. *Object Shortcut Menu*, untuk membuka *content menu*, tombol ini seperti klik kanan pada elemen.
7. *Probe Data*, untuk memasukkan *probe* pada blok diagram sehingga nilai data yang mengalir dapat dilihat pada blok diagram.
8. *Get Color*, untuk melihat warna pada elemen yang di select.
9. *Set Color* untuk mengganti warna elemen pada *front panel* ataupun blok diagram.
10. *Set/Clear Breakpoint*, untuk menempatkan atau menghapus breakpoint pada blok diagram. Breakpoint digunakan ketika proses debugging, proses run akan berhenti seketika saat berada pada elemen yang diberi *breakpoint* walaupun pada aturan dataflow labview tidak dimungkinkan program berhenti ditengah proses.
11. *Connect Wire*, digunakan untuk menggambar kabel yang akan menghubungkan dua elemen.

Control pallete, berisikan elemen yang dapat dimasukkan pada *front panel* yang terdiri dari kontrol dan indikator. *Control pallete* inilah yang akan digunakan untuk membangun GUI dari sebuah aplikasi pada labview. Terdapat dua cara untuk menampilkan control pallete yaitu dengan klik *view -> control pallete* atau dengan cara klik kanan dimanapun pada *front panel*. Gambar 2.6 menunjukkan tampilan dari *control pallete*.



Gambar 2.6. Control pallete (sumber: finn, 2008)

Jika *control pallete* digunakan untuk membangun GUI pada *front panel* maka *Function pallete* digunakan untuk *coding* program pada bagian blok diagram. Ada dua cara untuk membuak *function pallete*, pertama dengan klik *view-> function pallete* atau cara kedua dengan klik kanan dibagian manapun ada blok diagram. Tampilan *function pallete* dapat dilihat pada Gambar 2.7.



Gambar 2.7. *Function pallete* (sumber: finn, 2008)

4. Data type pada Labview

Pada *front panel* dan diagram blok memiliki tipe data yang spesifik. Pada labview tipe data memiliki indikator warna yang berbeda sehingga dapat dengan mudah dibedakan serta tidak diperlukan adanya inisialisasi data. Bahkan beberapa tipe data dapat langsung dikonversikan ke tipe data yang lain secara otomatis tanpa adanya proses atau elemen yang bertugas khusus untuk mengkonversikannya misalkan tipe data *integer* dan *floating*. Beberapa tipe data yang terdapat pada labview dapat dilihat pada Gambar 2.8.

Gambar 2.8. Tipe Data Pada Labview Dengan Indikasi Perbedaan Warna Kabel. (sumber: ni.com)

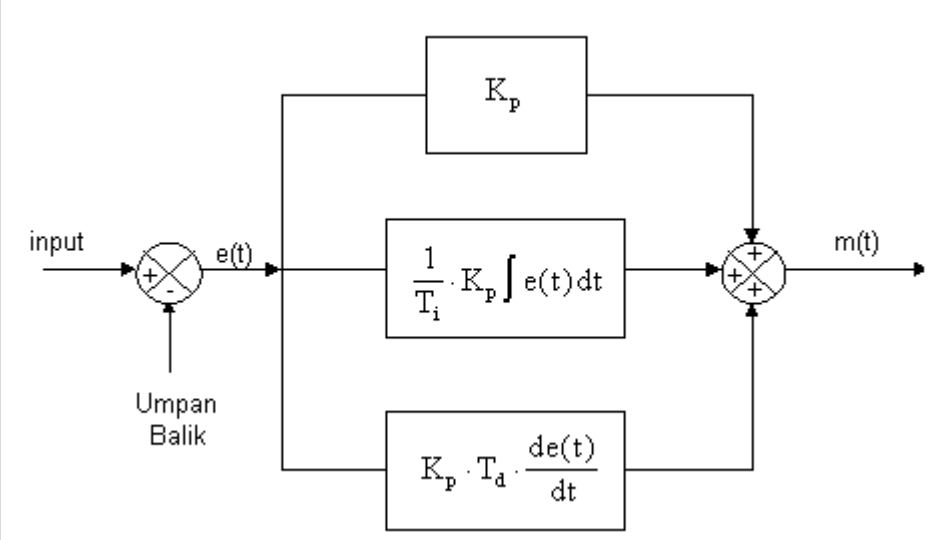
2.2.5 PID Control

PID (*Proportional Integral Derivative*) merupakan suatu metode yang digunakan untuk memaksimalkan proses suatu pengolah sinyal sehingga mampu didapatkan *output* yang maksimal. Pada proses PID terdapat tiga bagian yaitu aksi kontrol *Proportional*, aksi kontrol *Integral* dan aksi kontrol *Derivative*. Masing-masing dari aksi kontrol ini memiliki peran serta keunggulan tersendiri, diantaranya aksi kontrol proporsional memiliki *rise time* (waktu naik) yang cepat artinya sistem akan mampu mencapai kinerja maksimum dengan waktu yang cepat. Aksi kontrol *Integral* memiliki kelebihan untuk dapat memperkecil terjadinya *error* sedangkan aksi kontrol *derivative* memiliki peran untuk memperkecil *error* dan mengurangi *overshoot*. aksi kontrol PID beserta fungsinya dapat dilihat pada Tabel 2.1.

Tabel 2.1. Fungsi aksi kontrol pada PID (sumber: ctms.engin.umich.edu)

Response	Rise Time	Overshoot	Settling Time	S-S Error
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	No Change

Untuk dapat memaksimalkan output jika ketiga aksi kontrol ini digabungkan maka akan membentuk suatu sistem dengan rise time cepat, sedikit *error* dan tidak banyak *overshoot*, inilah yang disebut sebagai PID sebagai kontroller. Parameter pengontrol *Proportional Integral Derivative* (PID) selalu didasari atas tinjauan terhadap karakteristik yang di atur (*plant*). Dengan demikian bagaimanapun rumitnya suatu *plant*, perilaku *plant* tersebut harus di ketahui terlebih dahulu sebelum pencarian parameter PID itu dilakukan (Rahmat, 2012). Blok diagram PID dapat dilihat pada Gambar 2.9.



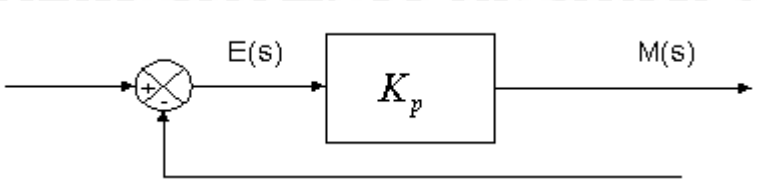
Gambar 2.9. Blok Diagram PID kontroller (sumber: elektroindonesia.com)

2.2.5.1 Kontrol Propotional

Kontroler proportional merupakan kontroler hasil perkalian dari Kp (Konstanta proporsional) dan *error* sistem atau dapat direpresentasikan dalam bentuk matematis sebagai berikut

$$u(t) = Kp.E(t) \quad (2.1)$$

Kp berlaku sebagai *Gain* (penguat) saja tanpa memberikan efek dinamik kepada kinerja kontroler. Pengaruh kontrol Proportional pada sistem yaitu dapat menambah atau mengurangi kestabilan sistem, dapat memperbaiki respon transien, mengurangi *error steady state*. Untuk dapat menghilangkan *error* maka dibutuhkan nilai Kp yang besar jika nilai Kp besar maka sistem tidak akan stabil. Blok diagram aksi kontrol proporsional dapat dilihat pada Gambar 2.10.

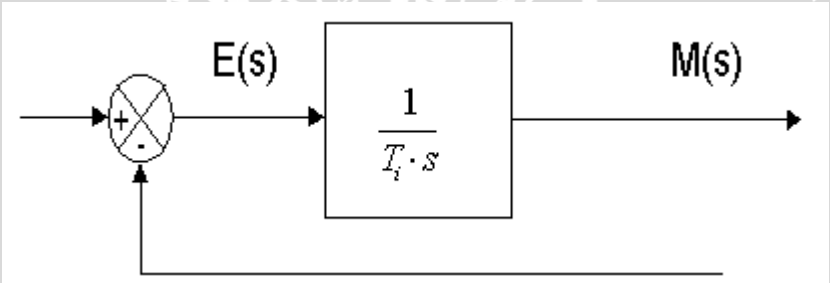


Gambar 2.10. Blok diagram Aksi kontrol Proporsional (sumber: elektroindonesia.com)

2.2.5.2 Kontrol Integral

$$u(t) = Ki \int e(t)\, dt \qquad (2.2)$$

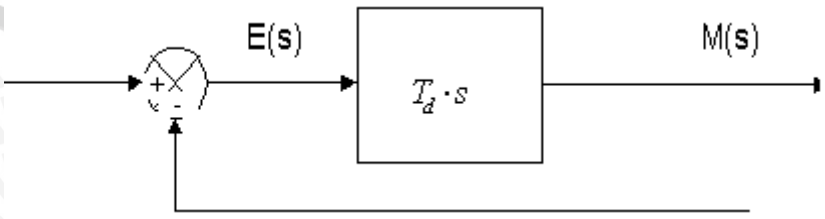
Persamaan diatas merupakan persamaan matematis dari kontrol integral. Dengan Ki adalah konstanta Integral, Jika e(T) mendekati konstan (bukan nol) maka u(t) akan menjadi sangat besar sehingga diharapkan dapat memperbaiki error. Jika e(T) mendekati nol maka efek kontrol I ini semakin kecil. Kontrol I dapat memperbaiki sekaligus menghilangkan respon steady-state, namun pemilihan Ki yang tidak tepat dapat menyebabkan respon transien yang tinggi sehingga dapat menyebabkan ketidakstabilan sistem. Blok diagram aksi kontrol integral dapat dilihat pada Gambar 2.11.



Gambar 2.11. Blok Diagram aksi kontrol integral (sumber: elektroindonesia.com)

2.2.5.3 Kontrol Derivative

Kontrol *derivative* memiliki pengaruh pada sistem berupa efek redaman pada sistem yang berisolasi sehingga nilai Kp dapat diperbesar, selain itu kontrol derivative mampu memperbaiki respon transien dan kontrol *derivative* hanya akan berubah saat ada perubahan *error*, sehingga saat ada *error* statis kontrol *derivative* tidak akan berfungsi.



Gambar 2.12. Blok Diagram aksi kontrol *Derivative* (sumber: elektroindonesia.com)

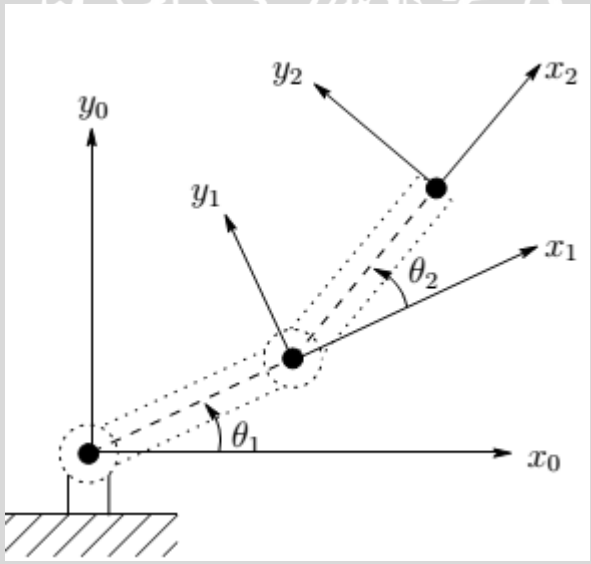
2.2.6 Kinematics

Kinematika merupakan bidang ilmu yang mempelajari gerak tubuh tanpa mempertimbangkan kekuatan ataupun penyebab terjadinya gerakan, sedangkan kinematika pada robot mengacu pada studi analisis gerakan manipulator robot (Sedar, 2006). Dalam kinematika terdapat istilah DOF (*Degree of Freedom*) yang merupakan derajat kebebasan dimana robot dapat bergerak pada suatu sumbu, kebebasan gerak dari robot manipulator ini tergantung dari jumlah *joint* yang dimiliki (eko, 2015).

Pergerakan sebuah robot selalu berkaitan dengan *workspace* atau ruang kerjanya, ruang kerja robot merupakan seluruh lokasi yang mampu dijangkau oleh ujung dari robot manipulator. Untuk dapat menentukan potensi pergerakan robot maka diperlukan kinematika robot, kinematika robot dapat didefinisikan menjadi *inverse kinematics* dan *forward kinematics*.

2.2.6.1 Forward kinematics

Forward kinematics atau kinematika maju merupakan perhitungan dalam menentukan posisi ujung robot manipulator terhadap koordinat berdasarkan perubahan gerak pada masing – masing *joint* yang menyusunnya (eko, 2015). Sistem koordinat merupakan garis imajiner yang memudahkan dalam proses penentuan dan penghitungan posisi. Proses *forward kinematics* pada robot 2 *link* dijelaskan melalui Gambar 2.13 berikut



Gambar 2.13. Robot manipulator 2 *link* (sumber: eko, 2015)

Pada Gambar 2.13 terdapat sebuah robot yang memiliki 2 *joint* dan 2 *link* dimana seluruh *joint* yang digunakan merupakan *joint revolute*, untuk dapat merubah posisi akhir dari ujung lengan robot maka robot harus merubah sudut masing – masing *joint*. Setiap *joint* memerlukan sistem koordinat lokal sebagai acuan perubahan *joint* terhadap *joint* sebelumnya. *Joint* 0 direpresentasikan dengan menggunakan sistem koordinat 0 (x0 dan y0) yang tidak berpindah, *joint* 1 direpresentasikan dengan sistem koordinat 1 (x1 dan y1) dimana sumbu x1 dan

y1 hanya akan berubah posisi ketika *joint* 0 berputar, serta sumbu x2 dan y2 hanya akan berpindah posisi ketika *joint* 1 berubah. Untuk dapat menentukan posisi ujung lengan terhadap koordinat dasar dapat digunakan persamaan trigonometri pada berikut

$$x = x_2 = \alpha_1 \cos\theta_1 + \alpha_2 \cos(\theta_1 + \theta_2)$$
$$y = y_2 = \alpha_1 \sin\theta_1 + \alpha_2 \sin(\theta_1 + \theta_2) \quad (2.3)$$

Dengan α_1 adalah panjang *link* 0 dan α_2 adalah panjang *link* 1.

2.2.6.2 Inverse Kinematics

Inverse kinematics merupakan kebalikan dari *forward kinematics*, jika *forward kinematics* merupakan perhitungan untuk menentukan posisi dengan diketahui perubahan pada masing-masing *joint* maka *inverse kinematics* menentukan perubahan masing-masing *joint* berdasarkan posisi yang diketahui. Pada robot 2 *link* 2 *joint* dapat ditentukan perubahan dari masing-masing *joint* berdasarkan sistem koordinat yang diketahui dapat digunakan rumus sebagai berikut

$$\theta_2 = cos^{-1}\left(\frac{x^2 + y^2 - L1^2 - L2^2}{2.L1.L2}\right) \quad (2.4)$$

$$\theta_1 = tan^{-1}\left(\frac{y(L1+L2.\cos\theta2) - x(L2.\sin\theta2)}{x(L1+L2.\cos\theta2) + y(L2.\sin\theta2)}\right) \quad (2.5)$$

Dengan:

X = titik koordinat x yang dituju

Y = titik koordinat y yang dituju

L1= panjang *link* 0

L2 = panjang *link* 1

Θ_2 = Sudut pada sistem koordinat 1

Θ_1 = Sudut pada sistem koordinat 0

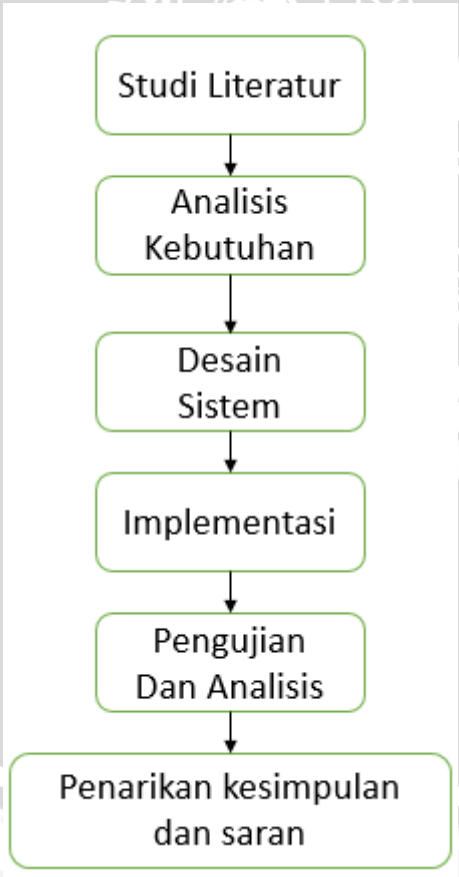
BAB 3 METODOLOGI

Bab metodologi akan membahas tentang struktur penelitian yang akan dilakukan, sehingga dengan adanya metodologi penelitian ini diharapkan proses penelitian dapat berjalan dengan terstruktur. Beberapa bahasan yang akan diulas dalam bab metodologi adalah Metode penelitian yang berisikan alur metode penelitian dan studi literatur yang diperlukan, analisis kebutuhan, desain sistem dan implementasi sistem, pengujian analisis serta penarikan kesimpulan dan saran.

3.1 Metode Penelitian

Dalam perancangan sistem terdapat proses-proses yang saling berkaitan agar menciptakan suatu perancangan yang terstruktur dengan baik. Pada bab ini akan dijelaskan langkah-langkah yang dilakuan dalam analisis dari sistem yang dirancang.

3.1.1 Alur Metode Penelitian



Gambar 3.1 Alur metode penelitian

Gambar 3.1 merupakan gambar yang menjelaskan tentang alur metode penelitian. Langkah pertama yang diperlukan untuk merancang sebuah sistem adalah studi literatur dimana pada tahap ini harus dilaksanakan pengkajian pada

setiap teori yang mendukun. Langkah kedua adalah analisis kebutuhan dimana pada tahap ini akan dianalisis segala kebutuhan sistem sehingga sistem dapat berjalan dengan baik serta lebih terarah dalam proses pengerjaan. Langkah ketiga adalah Pengumpulan data, pada proses ini dilakukan analisis data secara matematis yang nantinya akan dijadikan acuan dalam proses pengujian sistem apabila data matematis yang telah dikumpulkan sesuai dengan hasil data olah sistem maka sistem telah berhasil mensimulasikan PID kontroler pada robot lengan 2 DOF. Langkah keempat adalah desain sistem, pada langkah ini akan didesain seperti apa proses sistem akan berjalan dan tampilan *front panel* yang digunakan, dengan proses ini diharapkan proses pengerjaan sistem dapat lebih terarah. Langkah kelima adalah Implementasi yang merupakan proses pembuatan sistem. Langkah keenam adalah Pengujian dan analisis, langkah ini diperlukan untuk menguji tingkat akurasi dan keberhasilan sistem dalam mensimulasikan gerakan lengan robot dengan PID kontroler. Langkah terakhir adalah penarikan kesimpulan dan saran, kesimpulan dan saran didapatkan dari hasil analisis yang telah dilakukan.

3.1.2 Studi Literatur

Studi literatur yang dilakukan bertujuan untuk mengkaji hal-hal yang berhubungan dengan teori-teori yang mendukung dalam perencanaan dan perealisasian alat. Adapun teori-teori yang dikaji adalah sebagai berikut:

1) *Inverse Kinematics*.
2) *Forward Kinematics*.
3) *Programming* Labview.
4) PID kontroler.
5) PWM *Pulse*.
6) DC Motor.
7) Integral Kecepatan.

3.1.3 Analisis Kebutuhan

Analisis kebutuhan akan digunakan sebagai acuan dasar dalam membangun sebuah sistem, analisis kebutuhan akan mengidentifikasikan apa yang dibutuhkan oleh sistem dan apa yang harus disediakan oleh sistem. Analisis kebutuhan dibagi menjadi 2 macam yaitu analisis kebutuhan *user* dan analisis kebutuhan sistem.

3.1.3.1 Analisis kebutuhan *user* (*user requirement*)

Analisis kebutuhan *user* menjelaskan tentang kebutuhan secara functional saja, apa saja yang dibutuhkan oleh *user dan* apa yang dapat dilakukan *user* terhadap sistem. Dalam sistem ini *user* memerlukan simulator yang mampu mensimulasikan gerakan masing-masing *joint* pada robot kengan untuk dapat membentuk sudut sehingga mampu mencapai koordinat yang di-*input* oleh *user* dengan menggunakan kontrol PID.

3.1.3.2 Analisis kebutuhan sistem

Analisis kebutuhan sistem akan mengidentifikasi perangkat-perangkat yang akan digunakan dalam membangun dan mengimplementasikan sistem. Perangkat yang digunakan harus sesuai dengan peran yang akan dijalankan dalam kerja sistem sehingga dapat mempermudah dalam perancangan. Didalam analisis kebutuhan sistem dijelaskan kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional menggambarkan kebutuhan sistem agar dapat dibangun dan berjalan dengan baik sedangkan kebutuhan non-fungsional menggambarkan batasan kebutuhan sistem. Adapun kebutuhan sistem didefinisikan sebagai berikut

Kebutuhan Fungsional:

Hardware:

Dikarenakan sistem yang dibangun adalah sistem berupa simulator maka *hardware* yang dibutuhkan adalah sebuah PC/Laptop yang digunakan untuk proses *coding* dan *running* dari simulator. Pada proses pembuatan sistem ini digunakan laptop Samsung type RV413 dengan *operating system* windows 8.1 Pro 32-bit, prosesor AMD E-300 APU *with* Radeon(tm) HD *Graphics* 1.30GHz dan RAM 2,00 Gb.

Software:

1. NI Labview 2014
2. *MathScript Module* 2014
3. *Control and Simulation System Module* 2014

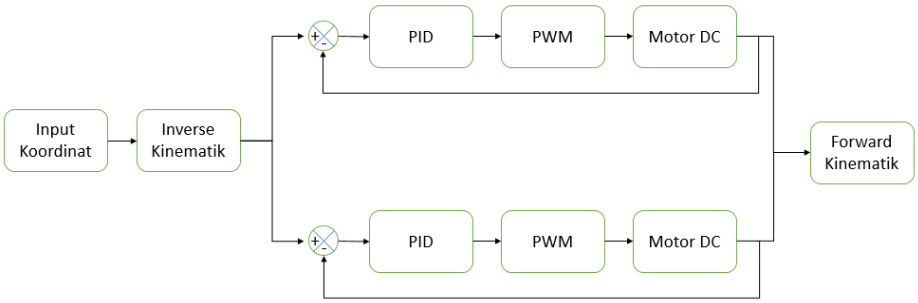
Kebutuhan Non-Fungsional:

1. Spesifikasi komputer yang digunakan dalam proses pembuatan maupun proses *running* sistem, semakin bagus spesifikasi komputer yang digunakan maka semakin cepat proses *running* sistem.

2. Seluruh *software* dan module yang dibutuhkan tersedia serta dapat berjalan dengan baik.

3.1.4 Desain Sistem

Tahap desain sistem ini bertujuan agar perancangan sistem penelitian yang dilakukan menjadi terstruktur. Desain sistem pada penelitian ini seperti Gambar 3.2 berikut:



Gambar 3.2. Diagram Blok Sistem

Dari diagram alur pada Gambar 3.2 dapat diketahui bahwa *user* dapat memberikan *input* berupa koordinat. *Input* koordinat yang dimasukkan oleh *user* akan diolah dengan menggunakan *inverse kinematics* sehingga menghasilkan nilai sudut θ_1 dan θ_2, masing masing besar sudut ini akan diolah dengan PID kontroler dan masuk ke PWM (*Pulse Width Modulation*) yang merupakan input dari Motor DC. Kecepatan putar dari motor DC inilah yang akan diintegralkan sehingga menjadi sudut hasil pengolahan PID kontroler. Terdapat 2 motor DC yang menghasilkan 2 sudut, dari kedua sudut tersebut diolah dengan *forward kinematics* dan dihasilkan koordinat hasil pengolahan sistem.

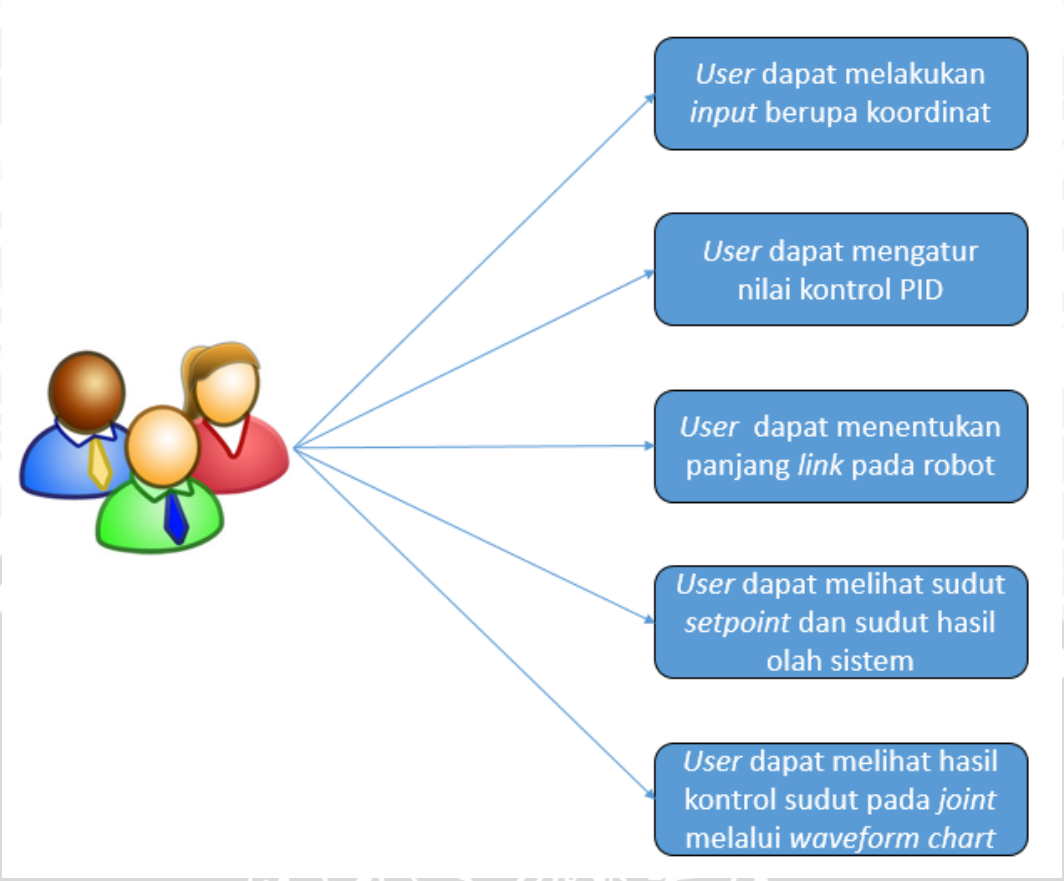
3.1.4.1 Spesifikasi Sistem

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

1. *Software* dalam pembuatan aplikasi menggunakan Labview 2013 dari National Instrument.
2. Input yang digunakan berupa titik koordinat.
3. Nilai koordinat akan diproses dengan menggunakan *inverse kinematics* untuk didapatkan nilai sudut.
4. Hasil pengolahan PID kontroler akan dikonversikan menjadi PWM *pulse.*
5. PWM *pulse* akan disimulasikan menjadi motor DC, kecepatan motor DC akan diintegralkan menjadi sudut akhir.
6. Dari sudut akhir yang diperoleh dari integral kecepatan motor DC akan diproses dengan menggunakan *forward kinematics* sehingga menghasilkan nilai koordinat kembali.

3.1.4.2 Use Case Diagram

Use case diagram menunjukkan kebutuhan *user* yang harus tersedia dalam sistem. Proses perancangan *use case* diagram ini diperlukan untuk mengetahui fitur yang akan dibuat pada sistem sehingga proses desain serta perancangan dapat lebih terarah. Use case diagram pada sistem ini dapat dilihat pada Gambar 3.3



Gambar 3.3. *Use case* diagram

Pada Gambar 3.3 menunjukkan bahwa *user* dapat melakukan *input* berupa sistem koordinat, mengatur kontrol PID agar besar sudut hasil olahan sistem mendekati *setpoint*, *user* dapat menentukan panjang *link* pada robot lengan yang nantinya akan menentukan titik koordinat yang mampu dicapai oleh sistem, selain itu pada sistem ini *user* dapat melihat hasil simulasi dengan menggunakan *chart* dan melihat sudut *setpoint* yang dapat dibandingkan dengan sudut hasil olah sistem.

3.1.4.3 Desain *front panel*

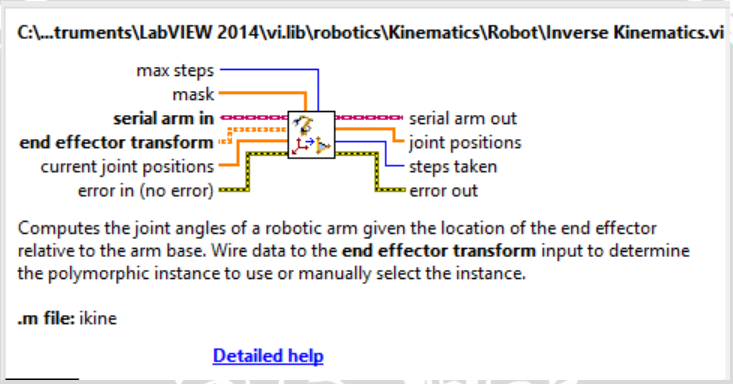
Pada desain *front panel* akan dibahas tata letak GUI (*graphical user interface*) yang nantinya akan berhubungan langsung dengan *user*. Perancangan desain *front panel* ini sangat dibutuhkan untuk dapat menyusun sebuah *user interface* yang diharapkan dapat digunakan dengan mudah oleh *user*. Pada sistem ini direncanakan indikator output dapat dilihat dengan menggnakan grafik dimana satu grafik akan menunjukkan dua indikator yaitu nilai sudut setpoint dan nilai sudut hasil olah sistem. Selain itu akan dibutuhkan kontrol bagi *user* untuk dapat memasukkan nilai koordinat, nilai PID dan nilai panjang lengan yang diinginkan. Agar *user* dapat membandingkan hasil sistem dan setpoint maka dibutuhkan indikator untuk nilai sudut dan nilai koordinat yang dihasilkan sistem.

3.1.5 Implementasi

Pada tahap ini akan dilakukan proses pengimplementasian sistem sesuai dengan perancangan yang telah dibuat sehingga sistem dapat mencapai target yang telah ditentukan sebelumnya. Dikarenakan sistem yang dibuat adalah sistem simulator maka proses implementasi yang dibutuhkan merupakan implementasi program dimana program ini merupakan hasil dari penggabungan beberapa sub program yang digunakan, berikut implementasi sub program yang digunakan:

3.1.5.1 Implementasi Inverse kinematics

Pada proses ini akan dilakukan pengimplementasian fungsi *inverse kinematics* pada sistem. Proses implementasi ini dilakukan dengan membuat fungsi sesuai rumus yang akan dibahas pada Sub Bab 5.1.3, pada dasarnya labview telah menyediakan sub vi untuk fungsi *inverse kinematics* pada robot, namun sub vi ini kurang sesuai dengan sistem karena memiliki *input* dan *output* yang berbeda. Spesifikasi *input* pada sub vi *inverse kinematics* dapat dilihat pada Gambar 3.4.



Gambar 3.4. Sub VI *Inverse Kinematics* (sumber: ni.com)

Pada Gambar 3.5 sub VI *inverse kinematics* memiliki *input* berupa *max steps*, *mask, serial arm in, end effector* dll. Sedangkan pada sistem hanya dibutuhkan logika rumus sederhana untuk mengolah *input* koordinat menjadi sudut, sehingga penggunaan sub vi *inverse kinematics* dinilai terlalu kompleks untuk sistem yang bahkan bisa menimbulkan kegagalan sistem.

3.1.5.2 Implementasi Kontrol PID

Sama seperti implementasi *inverse kinematics*, proses implementasi PID kontroler juga dilakukan dengan rumus sesuai pada Sub Bab 2.2.5. Selain *inverse kinematics* sebenarnya labview juga telah menyediakan sub vi PID kontroler yang dapat digunakan namun penulis lebih memilih membuat PID dengan menggunakan rumus karena input dan output yang digunakan lebih sederhana dan lebih mudah dimengerti.

3.1.5.3 Implementasi PWM pulse

Hasil nilai olahan dari PID akan disimulasikan menjadi PWM *pulse*, penggunaan PWM *pulse* ini diperlukan sebagai input dari motor DC yang nantinya

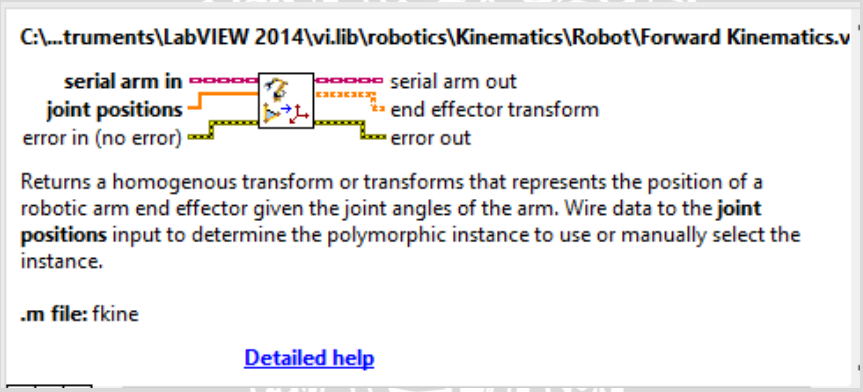
akan mensimulasikan motor servo. Pada implementasi PWM pulse akan digunakan *formula node* dari labview, *formula node* ini memungkinkan *programmer* untuk memprogram rumus dalam bentuk matematis. Digunakannya *formula node* ini agar memudahkan proses *coding*.

3.1.5.4 Implementasi Motor DC simulation

Sistem ini merupakan sistem yang mensimulasikan gerakan robot lengan yang pada dasarnya memiliki aktuator berupa motor servo, dikarenakan labview belum memiliki simulator untuk motor servo maka digunakan motor DC. Motor servo akan menghasilkan perubahan sudut sedangkan motor DC hanya berputar sesuai dengan voltase yang diberikan. Dari kecepatan motor DC ini akan diintegralkan hingga didapatkan besar sudut yang dihasilkan oleh sistem.

3.1.5.5 Implementasi *Forward kinematics*

Untuk proses implementasi *forward kinematics* dilakukan dengan menggunakan rumus yang dibuat dengan menggunakan fungsi *sinus, cosinus, multiply, divide* dan *add* pada labview. Sama dengan *inverse kinematics* labview juga telah menyediakan sub vi *forward kinematics* namun *input* yang dimiliki terlalu rumit untuk diimplementasikan pada sistem ini. Berikut spesifikasi input pada sub vi *forward kinematics*



Gambar 3.5. Sub VI F*orward Kinematics* (sumber: ni.com)

Gambar 3.5 merupakan spesifikasi *input* dan *output* dari sub vi *forward kinematics*. Input dari sub vi *forward kinematics* terdapat *input* berupa *serial arm in, joint potition* dan *error* sedangkan pada sisi *output* terdapat *serial arm out, end effector* dan *error out*. Komposisi *input output* ini dinilai terlalu rumit untuk diimplementasikan pada sistem sehingga perlu dibuat fungsi *forward kinematics* dari rumus.

3.1.5.6 Implementasi tata letak *front panel*

Pada tahap ini akan dilakukan implementasi tampilan awal dari *front panel* yang akan menunjukkan indikator besar sudut pada masing-masing *joint* serta kontrol yang dapat dilakukan. Desain *front panel* ini diharapkan dapat memudahkan *user* dalam penggunaan aplikasi.

3.1.6 Pengujian dan Analisis Sistem

Pada tahap pengujian sistem ini menggunakan parameter yang disesuaikan pada perancangan sistem. Beberapa rancangan skenario pengujian sistem adalah sebagai berikut :

1. Pengujian pada sisi pengaruh nilai PID terhadap besar sudut masing-masing *joint* untuk mendekati besar sudut setpoint hasil *inverse kinematics*.
2. Pengujian perubahan posisi titik koordinat dengan nilai Kp, Ki, dan Kd yang sama.

Sedangkan parameter yang dianalisis pada sistem sebagai berikut

1. Besar sudut pada masing-masing *joint* berdasarkan *setpoint* dari *inverse kinematics* dan dari integral kecepatan motor DC.

2. Nilai titik koordinat yang diinputkan oleh *user* dan nilai koordinat yang berhasil dicapai oleh sistem.

3. Nilai Kp, Ki, Kd yang tepat.

4. Titik koordinat yang mampu dicapai sesuai dengan panjang lengan yang di-*setting*.

3.1.7 Penarikan Kesimpulan dan Saran

Proses penarikan kesimpulan dilakukan setelah melakukan proses implementasi, pengujian dan analisis terhadap sistem. Kesimpulan didapatkan berdasarkan hasil pengujian yang telah dianalisis, kesimpulan ini diharapkan dapat menjadi acuan dalam penelitian selanjutnya yang masih memiliki keterkaitan topik dengan sistem kontrol PID pada simulasi robot lengan.

BAB 4 PERSYARATAN

Bab ini menjelaskan persyaratan minimal yang harus terpenuhi supaya tahap perancangan dan implementasi dapat berjalan dengan baik. Bab persyaratan ini berisikan deskripsi umum system yang menunjukkan pengkondisian system agar sistem dapat berjalan dengan baik.

4.1 Deskripsi Umum

Baba deskripsi umum ini akan membahas tentang gambaran umum sistem yang akan bekerja, bagaimana sistem bekerja, syarat agak sistem dapat bekerja serta *user* yang akan menggunakan sistem. Beberapa bahasan yang akan diulas dalam deskripsi umum ini adalah perspektif sistem, persyaratan hardware, persyaratan software, lingkungan operasi sistem, batasan sistem dan asumsi ketergantungan.

4.1.1 Perspektif Sistem

Sistem ini dikatakan dapat bekerja sesuai tujuan apabila sistem mampu mengolah *input* koordinat yang akan di-*inverse* kan untuk menjadi sudut yang akan menjadi *input* bagi *pulse* PWM, *pulse* PWM akan disimulasikan dengan menggunakan motor DC yang kecepatannya diintegralkan dan menjadi sudut. Selisih dari nilai sudut akhir dan input sudut inilah yang akan dikontrol dengan PID, nilai sudut hasil sistem akan diolah dengan *forward kinematics* hingga menjadi koordinat akhir yang mampu dicapai lengan robot.

4.1.2 Karateristik Pengguna

Sistem ini diperuntukkan bagi pengguna yang mengerti tentang sistem kontrol, karena simulator hanya bertujuan untuk menentukan nilai PID yang tepat terhadap suatu keadaan robot lengan.

4.1.3 Persyaratan *Hardware*

Dikarenakan sistem yang dibuat adalah sistem simulasi maka *hardware* yang dibutuuhkan hanyalah seperangkat komputer dengan spesifikasi yang mampu menjalankan *software* labview dari national instrument, pada proses pembuatan sistem ini digunakan laptop Samsung type RV413 dengan *operating system* windows 8.1 Pro 32-bit, prosesor AMD E-300 APU *with* Radeon(tm) HD *Graphics* 1.30GHz dan RAM 2,00 Gb.

4.1.4 Persyaratan *Software*

Persyaratan *software* merupakan kebutuhan sistem akan *software* yang harus dipenuhi, apabila persyaratan ini tidak dipenuhi maka sistem tidak akan berjalan dengan baik atau tidak mampu berjalan sama sekali. Berikut adalah kebutuhan *software* yang harus terpenuhi untuk dapat menyusun sistem simulator robot lengan dengan kontroler PID:

1. NI Labview 2014.

2. *Modul MathScript* Labview National Instrument 2014.
3. *Modul Control and simulation loop* National Instrument 2014.

4.1.5 Lingkungan Operasi Sistem

Dikarenakan sistem merupakan sebuah simulator maka lingkungan dari sistem yang dibutuhkan adalah lingkungan yang menyediakan dan memungkinkan adanya 1 unit komputer guna jalannya simulator.

4.1.6 Batasan Sistem

Beberapa batasan yang ada pada sistem ini antara lain :

1. Hanya dapat menerima input berupa koordinat.
2. Perolehan sudut akhir berasal dari integral kecepatan motor DC dikarenakan simulator servo pada labview belum ada.
3. Proses *coding graphical programming* dengan menggunakan labview *graphical programming* dari National Instrument.
4. Panjang lengan menentukan titik koordinat yang mampu dicapai.
5. Seluruh sudut input dan output dalam satuan derajat.

4.1.7 Asumsi dan Ketergantungan

Beberapa asusmi dan ketergantungan pada sistem ini antara lain :

1. Kecepatan eksekusi simulator tergantung pada spesifikasi komputer yang digunakan, semakin tinggi spesifikasi komputer yang digunakan maka semakin cepat proses komputasi pada simulator.
2. Kesesuaian nilai PID kontroller tergantung pada plant motor yang digunakan.
3. Titik koordinat yang mampu dicapai ujung lengan terbatas, tergantung dari panjang lengan robot.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini akan membahas mengenai perancangan dari sistem yang akan dibuat dan berdasarkan hasil perancangan tersebut akan dilakukan implementasi sistem perangkat lunak.

5.1 Perancangan Sistem

Pada tahap perancangan sistem memaparkan tahapan perancangan sistem, proses dimana sistem akan dibuat serta *user interface* yang akan dibuat.

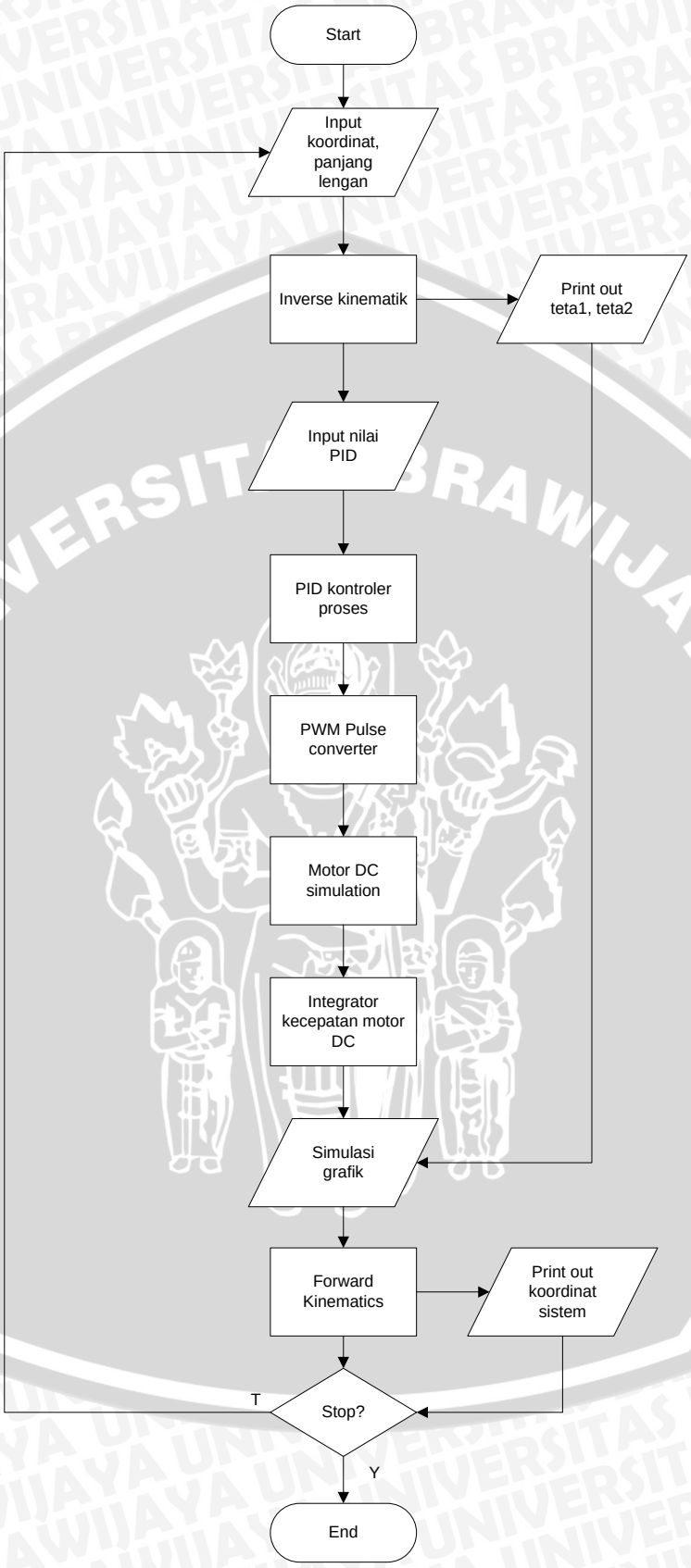
5.1.1 Perancangan Perangkat Lunak

Pada perancangan perangkat lunak sistem dilakukan pembagian proses pengkodean menjadi beberapa sub Vi, yaitu:

1. *Inverse Kinematics*
2. PID kontrol
3. PWM *Pulse*
4. Motor DC
5. *Forward Kinematics*

Sub Vi *inverse kinematics* akan digunakan untuk mengkonversikan titik koordinat yang dimasukkan *user* menjadi perubahan besar sudut untuk masing-masing *joint*. Pada proses PID kontrol akan dilakukan *tuning* secara manual oleh *user* dengan memasukkan nilai PID hingga ditemukan nilai yang tepat. Dikarenakan simulator untuk motor servo belum ada pada labview *graphical programming* maka digunakanlah matematis motor DC untuk mensimulasikan motor servo. Dari kecepatan motor DC akan diintegralkan hingga didapatkan sudut akhir. Untuk dapat mensimulasikan dengan menggunakan motor DC maka diperlukan *pulse* PWM sebagai pemicu daya DC sesuai dengan input yang diberikan oleh *user*. Setelah nilai sudut didapatkan maka akan diolah menggunakan *forward kinematics* hingga ditemukan titik koordinat yang mampu dicapai oleh lengan robot. Cara kerja dari sistem dapat dilihat pada *flowchart* Gambar 5.1.

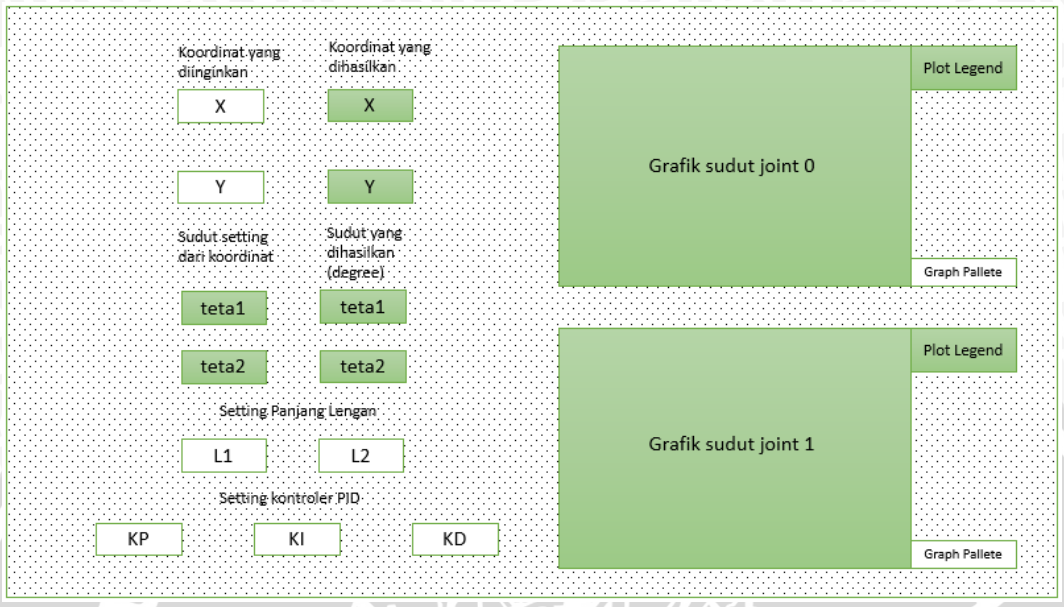
Pada *flowchart* Gambar 5.1 dapat dilihat bahwa *User* akan memasukkan nilai koordinat yang ingin dicapai oleh lengan robot, nilai koordinat tersebut akan diolah dengan menggunakan *inverse kinematics*. Karena lengan yang digunakan adalah 2 DOF maka dari proses *inverse kinematics* akan dihasilkan 2 sudut yang akan ditampilkan dalam bentuk *chart* dan angka. 2 buah sudut inilah yang akan dikontrol dengan menggunakan PID kontroler, lalu dikonversikan menjadi PWM *pulse* dan akan disimulasikan dengan menggunakan motor DC. Kecepatan dari motor DC ini akan diintegralkan hingga menghasilkan sudut hasil kontrol sistem, sudut hasil ini akan menjadi *feedback* bagi PID sehingga dapat mengurangi adanya *error*.



Gambar 5.1 Flowchart sistem

5.1.2 Perancangan *User Interface*

User interface merupakan salah satu fitur yang dimiliki oleh sistem ini, perancangan *user interface* untuk simulator ini dapat dilihat pada Gambar 5.2.



Gambar 5.2. Perancangan *user interface*

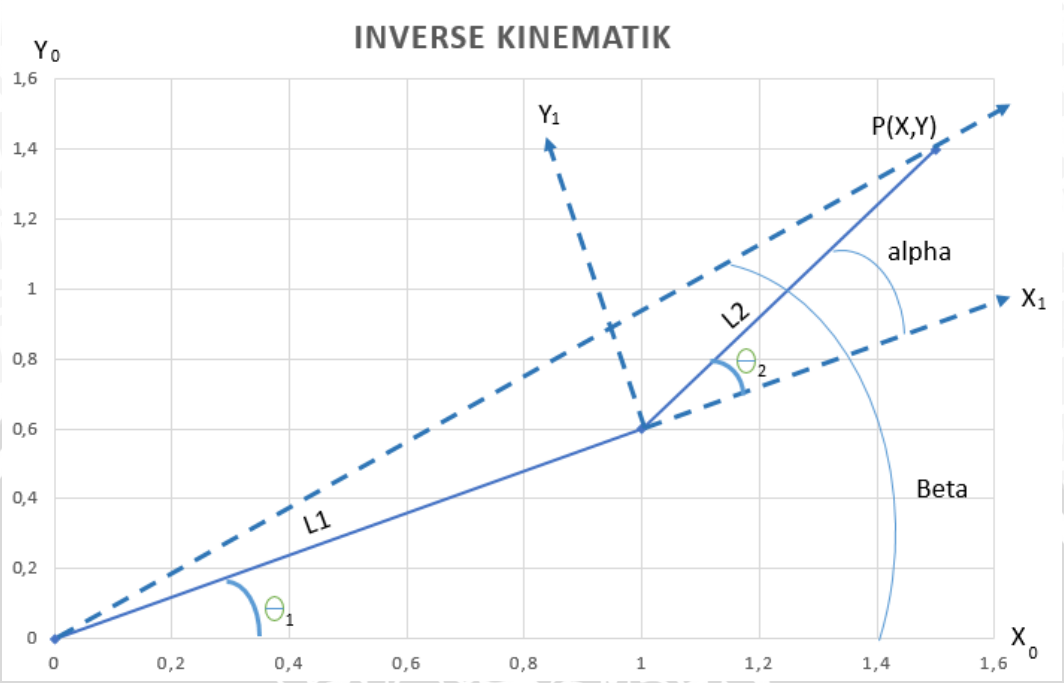
Pada desain *front panel* diatas dibuat sedemikian rupa, dimaksudkan agar *user* dapat mudah dalam menggunakan aplikasi. Warna hijau pada desain *front panel* menunjukkan fungsi sebagai indikator yang berarti hanya sebagai tampilan dengan nilai yang tidak dapat diubah. Warna putih pada desain *front panel* menunjukkan fungsi sebagai kontrol, dimana fungsi ini merupakan fungsi *input* yang dapat diubah-ubah oleh *user*.

Tata letak grafik indikator perubahan sudut pada *joint* 0 dan *joint* 1 diletakkan pada sebelah kanan *front panel* dan kontrol indikator koordinat, sudut, panjang lengan serta *setting* kontrol PID diletakkan disebelah kiri *front panel* hal ini dimaksudkan agar *user* dapat dengan mudah membedakan antara kontrol dan indikator yang digunakan. Posisi kontrol koordinat X,Y dan indikator koordinat X,Y berada bersebelahan dimaksudkan agar *user* dapat dengan mudah membandingkan hasil koordinat pengolahan sistem dengan koordinat yang diinginkan. Posisi indikator sudut setting dari koordinat dengan sudut hasil pengolahan sistem juga diletakkan bersebelahan agar *user* dapat mengetahui dan memebandingkan besar sudut sebelum melalui kontrol PID dan sebelum melalui kontrol PID. *Graph pallete* dan *plot legend* diletakkan pada samping kanan grafik, *default* dari labview *plot legend* berada di atas grafik, hal ini dimaksudkan agar tampilan *front panel* cukup untuk 1 layar komputer sehingga *user* tidak perlu melakukan *scrolling* untuk dapat melihat keseluruhan tampilan *front panel*.

5.1.3 Perancangan Inverse Kinematics

Pada sub bab ini akan dijelaskan tentang proses penghitungan *inverse kinematics* guna mengubah input koordinat dari *user* menjadi besar sudut yang

dibutuhkan masing-masing *joint* agar ujung lengan mampu mencapai titik koordinat yang diinginkan. Untuk memahami analisis *inverse kinematics* dapat dilihat dengan Gambar 5.3 berikut



Gambar 5.3. Analisis *inverse kinematics*

Berdasarkan *forward kinematics* 2 DOF didapatkan rumus

$$X^2 = L1^2\, cos^2\theta_1 + L2^2.cos^2(\theta_1 + \theta_2) + 2.L1.L2\cos(\theta_1 + \theta_2)\cos\theta_1 \quad (5.7)$$

$$Y^2 = L1^2\, sin^2\theta_1 + L2^2\, sin^2(\theta_1 + \theta_2) + 2.L1.L2\sin(\theta_1 + \theta_2)\sin\theta_1 \quad (5.8)$$

Maka

$$X^2 + Y^2 = (L1^2\, cos^2\theta_1 + L2^2.cos^2(\theta_1 + \theta_2) + 2.L1.L2\cos(\theta_1 + \theta_2) + \cos\theta_1)\ L1^2\, sin^2\theta_1 + L2^2\, sin^2(\theta_1 + \theta_2) + 2.L1.L2\sin(\theta_1 + \theta_2)\sin\theta_1 \quad (5.9)$$

$$X^2 + Y^2 = L1^2(\, sin^2\theta_1 + cos^2\theta_1) + L2^2\left(sin^2(\theta_1 + \theta_2) + cos^2(\theta_1 + \theta_2)\right) + 2.L1.L2\,(\sin(\theta_1 + \theta_2)\sin\theta_1 + \cos(\theta_1 + \theta_2)\cos\theta_1) \quad (5.10)$$

$$X^2 + Y^2 = L1^2 + L2^2 + 2.L1.L2\ \cos\theta_2 \quad (5.11)$$

$$X^2 + Y^2 - L1^2 + L2^2 = 2.L1.L2\ \cos\theta_2 \quad (5.12)$$

$$\cos\theta_2 = \frac{X^2 + Y^2 - L1^2 - L2^2}{2.L1.L2} \quad (5.13)$$

$$\theta_2 = cos^{-1}\frac{X^2 + Y^2 - L1^2 - L2^2}{2.L1.L2} \quad (5.14)$$

berdasarkan rumus diatas dapat diketahui bahwa besar sudut θ_2 untuk mencapai koordinat tujuan pada robot lengan 2 DOF dapat diketahui dengan menggunakan rumus

$$\theta_2 = cos^{-1}\frac{X^2 + Y^2 - L1^2 - L2^2}{2.L1.L2} \quad (5.15)$$

sedangkan untuk dapat mengetahui perubahan sudut θ_1 yang diperlukan untuk ujung lengan robot dapat mencapai titik koordinat yang dituju maka menggunakan rumus sebagai berikut

$$\tan \alpha = \frac{\mathrm{L2} \sin \theta 2}{L1 + \mathrm{L2} \cos \theta 2} \tag{5.16}$$

$$\tan \beta = \frac{\mathrm{y}}{\mathrm{x}} \tag{5.17}$$

$$\theta_1 = \beta - \alpha \tag{5.18}$$

$$\tan \theta_1 = \frac{\tan \beta - tan\, \alpha}{1 + tan\beta\, tan\alpha} \tag{5.19}$$

$$\tan \theta_1 = \frac{\frac{\mathrm{y}}{\mathrm{x}} - \frac{\mathrm{L2} \sin \theta 2}{L1 + \mathrm{L2} \cos \theta 2}}{1 + \frac{\mathrm{y}}{\mathrm{x}} \cdot \frac{\mathrm{L2} \sin \theta 2}{L1 + \mathrm{L2} \cos \theta 2}} \tag{5.20}$$

$$\tan \theta_1 = \frac{y\,(L1 + \mathrm{L2} \cos \theta 2) - x(\mathrm{L2} \sin \theta 2)}{x(L1 + \mathrm{L2} \cos \theta 2) + y(\mathrm{L2} \sin \theta 2)} \tag{5.21}$$

$$\theta_1 = tan^{-1} \frac{y\,(L1 + \mathrm{L2} \cos \theta 2) - x(\mathrm{L2} \sin \theta 2)}{x(L1 + \mathrm{L2} \cos \theta 2) + y(\mathrm{L2} \sin \theta 2)} \tag{5.22}$$

berdasarkan kedua rumus tersebut dapat dihitung perubahan besar sudut yang dibutuhkan lengan robot untuk dapat mencapai titik koordinat yang dituju. Berikut contoh perhitungan penentuan besar sudut jika diketahui koordinat yang dituju.

Diketahui : $X = 3, Y = 5, L1 = 4, L2 = 3$

Ditanyakan: $\theta_1\ dan\ \theta_2$

Jawab: $\theta_2 = cos^{-1} \frac{X^2 + Y^2 - L1^2 - L2^2}{2.\mathrm{L1}.\mathrm{L2}}$

$$\theta_2 = cos^{-1} \frac{3^2 + 5^2 - 4^2 - 3^2}{2.4.3} = 41{,}40$$

$$\theta_1 = tan^{-1} \frac{y\,(L1 + L2\ cos\ \theta 2) - x(L2\ sin\ \theta 2)}{x(L1 + L2\ cos\ \theta 2) + y(L2\ sin\ \theta 2)}$$

$$\theta_1 = tan^{-1} \frac{5\,(4 + 3 \cos 41{,}40) - 3(3 \sin 41{,}40)}{3(4 + 3 \cos 41{,}40) + 5(3 \sin 41{,}40)} = 46{,}95$$

Berikut tabel yang menunjukkan perubahan besar sudut masing masing *joint* yang dibutuhkan untuk mencapai titik koordinat tertentu.

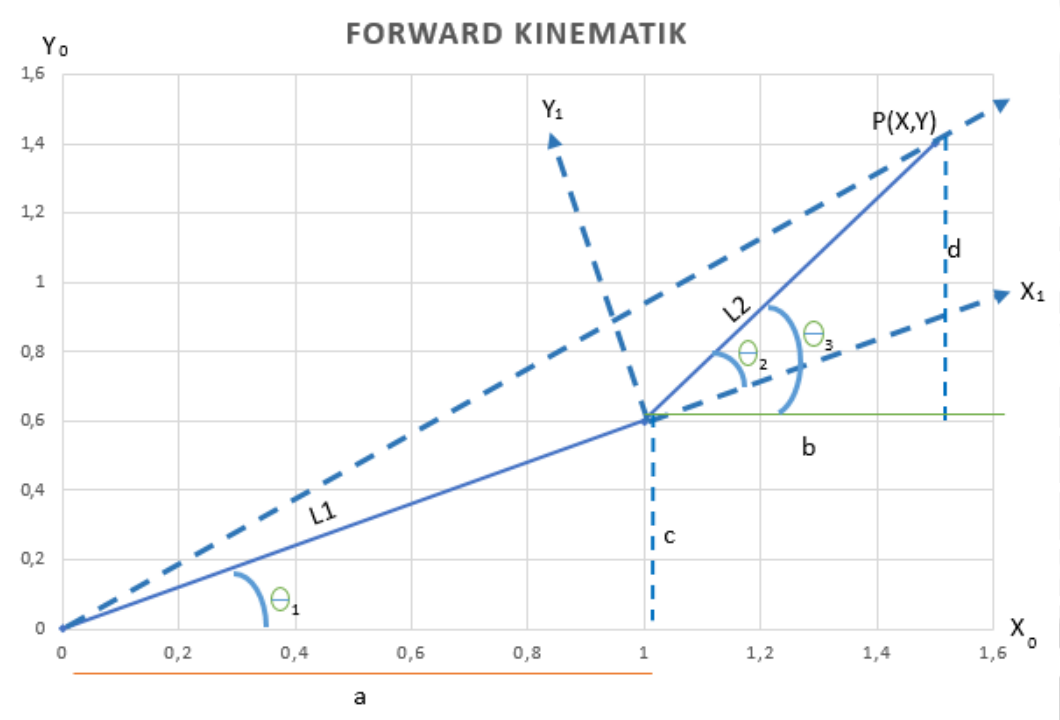
Tabel 5.1. Koordinat Dan Sudut *Inverse Kinematics*

X	Y	Θ_1	Θ_2
0	7	88,8	0
1	6	55,2	60
2	6	49,84	51,31
3	6	49,2	33,55
4	5	30,92	48,2
5	4	17,22	48,2
6	3	12,24	33,5
7	0	0	0

Tabel diatas merupakan tabel nilai koordinat dan perubahan sudut yang diperlukan masing-masing *joint* untuk dapat mencapai titik koordinat yang telah ditentukan. Jumlah sample diambil berdasarkan batas maksimum dari titik koordinat yang mampu dicapai lengan dengan panjang L1 = 4cm dan L2 = 3cm.

5.1.4 Perancangan Forward Kinematics

Sub bab ini akan dijelaskan proses penghitungan *forward kinematics* yang merupakan proses perubahan sudut menjadi koordinat sehingga hasil koordinat akhir yang dihasilkan sistem dapat dibandingkan dengan titik koordinat yang diinputkan oleh *user*. Pada *forward kinematics* diketahui sudut perubahan pada masing masing *joint* dan akan dicari titik koordinat yang berhasil dicapai berdasarkan sudut pada masing-masing sendi tersebut. Berikut perancangan rumus *forward kinematics* untuk robot lengan 2 DOF menurut Eko Setiawan melalui modul *robotics* 2015.



Gambar 5.4. Analisis *forward kinematics*

Posisi titik P(x,y) terhadap sistem koordinat 0 (x_0, y_0)

$$a = L1 \cos \theta_1 \quad (5.1)$$

$$b = L2 \cos \theta_3 = L2 \cos(\theta_1 + \theta_2) \quad (5.2)$$

$$c = L1 \sin \theta_1 \quad (5.3)$$

$$d = L2 \sin \theta_3 = L2 \sin(\theta_1 + \theta_2) \quad (5.4)$$

Maka

$$X = a + b = L1 \cos \theta_1 + L2 \cos(\theta_1 + \theta_2) \quad (5.5)$$

$$Y = c + d = L1 \sin \theta_1 + L2 \sin(\theta_1 + \theta_2) \quad (5.6)$$

Berdasarkan kedua rumus tersebut dapat dihitung titik koordinat yang dituju berdasarkan perubahan besar sudut masing-masing *joint*. Berikut contoh perhitungan penentuan titik koordinat jika diketahui besar sudut pada masing – masing *joint*.

Diketahui : $\theta_1 = 30{,}92 \quad \theta_2 = 48{,}2 \quad L1 = 4\ cm \quad L2 = 3cm$

Ditanyakan: X, Y

Jawab:

$$X = L1 \cos \theta_1 + L2 \cos(\theta_1 + \theta_2)$$

$$= 4 \cos 30{,}92 + 3 \cos(30{,}92 + 48{,}2)$$

$$= 4.0{,}857 + 3.(0{,}188)$$

$$= 3{,}428 + 0{,}56$$

$= 3{,}988$

$Y = L1 \sin\theta_1 + L2 \sin(\theta_1 + \theta_2)$

$= 4 \sin 30{,}92 + 3 \sin(30{,}92 + 48{,}2)$

$= 4.0{,}513 + 3.0{,}982$

$= 2{,}052 + 2{,}946$

$= 4{,}998$

Berikut tabel yang menunjukkan titik koordinat yang mampu dicapai berdasarkan perubahan sudut pada masing-masing *joint*.

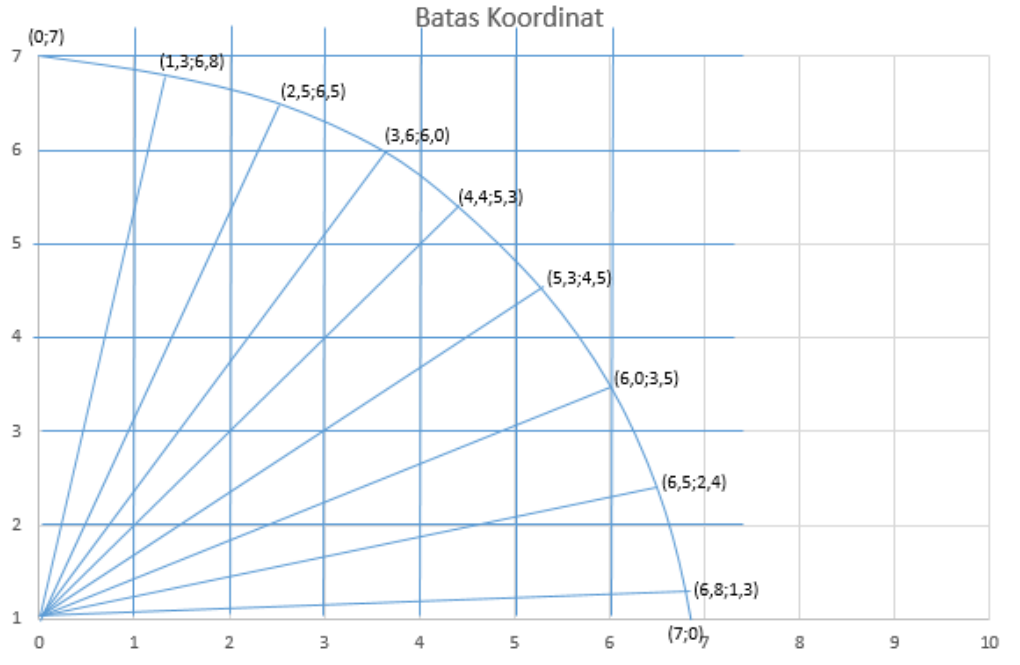
Tabel 5.2. Koordinat dan sudut *forward kinematics*

Θ_1	Θ_2	X	Y
88,8	0	0	7
55,2	60	1	6
49,84	51,31	2	6
49,2	33,55	3	6
30,92	48,2	4	5
17,22	48,2	5	4
12,24	33,5	6	3
0	0	7	0

Tabel diatas merupakan tabel penentuan titik koordinat berdasarkan perubahan besar sudut pada masing-masing *joint*. Jumlah *sample* diambil berdasarkan batas maksimum dari titik koordinat yang mampu dicapai lengan dengan panjang L1 = 4cm dan L2 = 3cm.

5.1.5 Perancangan Batasan Koordinat

Sub bab perancangan batasan koordinat akan menjelaskan tentang batasan titik koordinat yang mampu dicapai ujung robot berdasarkan panjang lengan yang dimiliki. Setiap robot lengan memiliki panjang lengan yang telah ditetukan, hal ini mempengaruhi titik koordinat yang mampu dicapai oleh ujung lengan. Semakin pendek panjang lengan maka semakin terbatas titik koordinat yang mampu dicapai. Pada penelitian ini digunakan panjang lengan L1 = 4 cm dan L2 = 3 cm, batasan koordinat yang mampu dicapai dengan panjang lengan L1 = 4 cm dan L2 = 3 cm dapat dilihat pada gambar grafik koordinat berikut.



Gambar 5.5. Grafik batas koordinat untuk L1 =4 cm dan L2 = 3 cm

Berdasarkan grafik pada Gambar 5.5, dengan panjang lengan L1 = 4 cm dan L2 = 3 cm koordinat yang dapat dicapai adalah koordinat yang masuk pada daerah melingkar. Koordinat (0;7), (1,3;6,8), (2,5;6,5), (3,6;6), (4,4;5,3), (5,3;4,5), (6;3,5), (6,5;2,4), (6,8;1,3), (7;0) merupakan batas koordinat posistif yang mampu dicapai oleh lengan robot dengan panjang L1 = 4 cm dan L2 = 3 cm. Setiap panjang lengan robot sangat mempengaruhi koordinat yang mampu dicapai, sehingga perbedaan panjang lengan robot juga akan mempengaruhi titik koordinat yang mampu dicapai.

5.2 Implementasi Sistem

Implementasi sistem dapat dilakukan apabila proses perancangan dari sistem telah terpenuhi karena implementasi sistem ini seluruhnya mengacu kepada perancangan sistem yang telah ditentukan sebelumnya. Pada bagian implementasi sistem memaparkan secara rinci spesifikasi perangkat lunak yang digunakan oleh sistem.

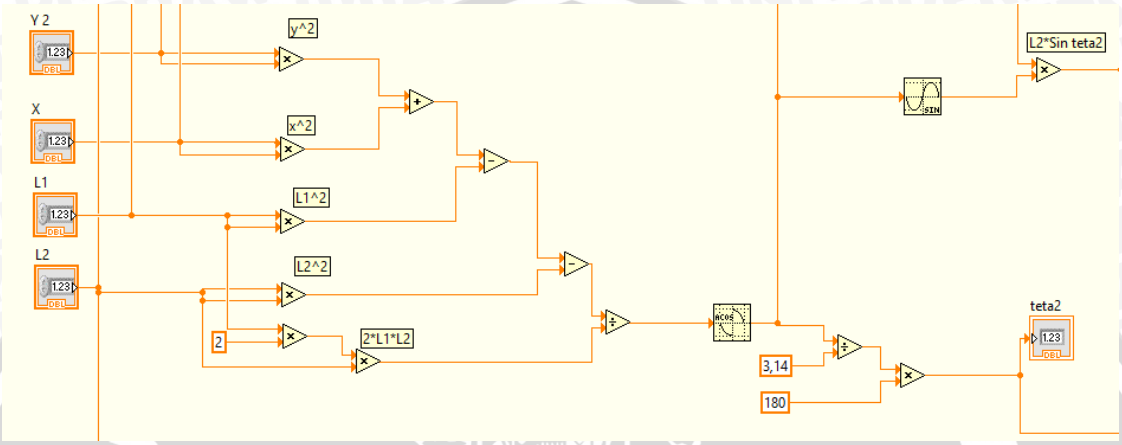
5.2.1 Implementasi *Inverse Kinematics*

Proses implementasi *Inverse Kinematics* dilakukan sesuai rumus 5.12 dan 5.22 yaitu

$$\theta_2 = cos^{-1} \frac{X^2 + Y^2 - L1^2 - L2^2}{2.L1.L2}$$

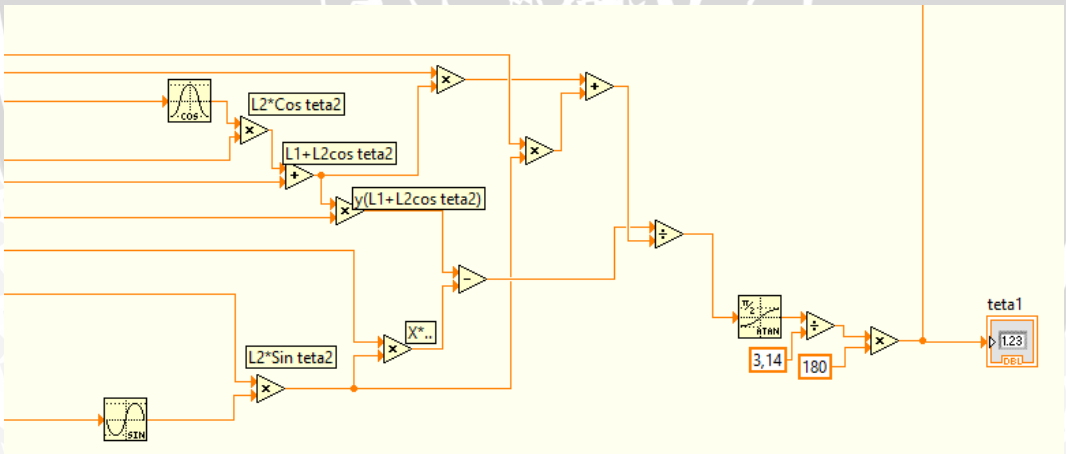
$$\theta_1 = tan^{-1} \frac{y\ (L1 + L2\ cos\ \theta2) - x(L2\ sin\ \theta2)}{x(L1 + L2\ cos\ \theta2) + y(L2\ sin\ \theta2)}$$

Inverse Kinematics ini akan digunakan untuk mengkonversikan sumbu koordinat yang dimasukkan oleh *user* menjadi besar perubahan sudut yang harus dilakukan oleh masing-masing *joint*, pada pengimplementasiannya sesuai dengan perancangan akan dibuat rumus dari awal sehingga tidak menggunakan sub VI *Inverse kinematics* hal ini dikarenakan Sub VI *inverse kinematics* memiliki *input* dan *output* yang terlalu kompleks untuk sistem. Implementasi *inverse Kinematics* dapat dilihat pada Gambar 5.6 dan Gambar 5.7.



Gambar 5.6. Implementasi *Inverse kinematics* teta 2

Pada Gambar 5.6 merupakan implementasi rumus *inverse kinematics* untuk menentukan nilai sudut θ_2, pembuatan rumus *inverse kinematics* ini dilakukan dengan menggunakan fungsi *add* dan *multiply* yang telah tersedia pada labview sehingga penyusunan kode disesuaikan dengan rumus yang berlaku.



Gambar 5.7. Implementasi *inverse kinematics* teta 1

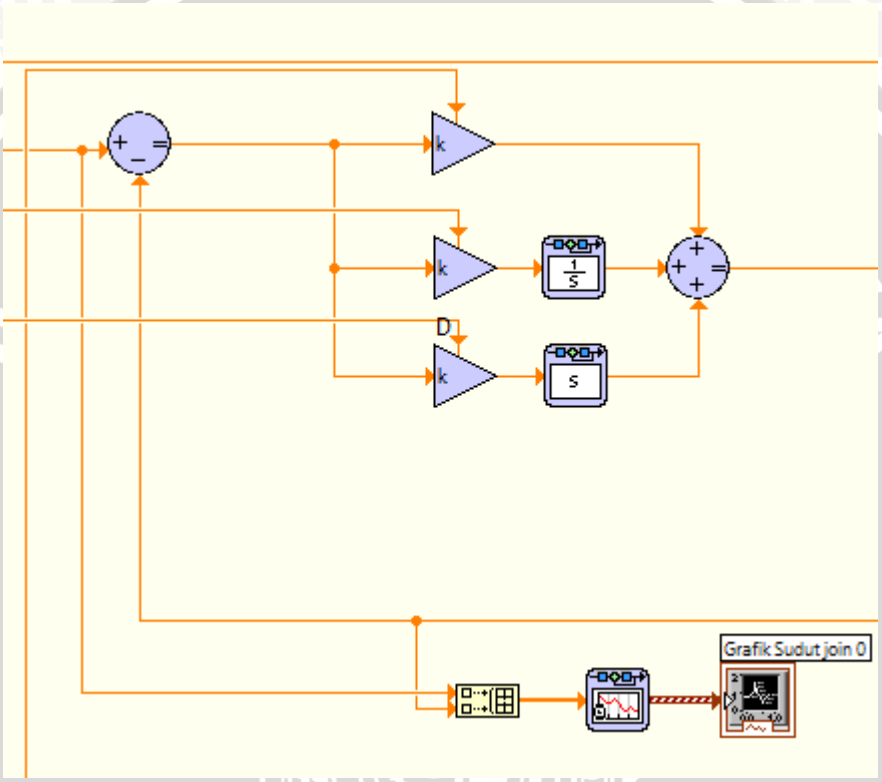
Pada Gambar 5.7 merupakan implementasi rumus *inverse kinematics* untuk menentukan sudut θ_1. Dapat dilihat bahwa terdapat fungsi sinus dan cosinus pada program di Gambar 5.6 dan 5.7, perlu diketahui bahwa input fungsi *sinus* dan *cosinus* dari labview adalah radian sehingga ketika akan ditampilkan nilai sudut yang dibentuk diperlukan adanya konversi dari radian ke derajat dengan menggunakan rumus

$$Derajat = \frac{Radian}{\pi} \; x \; 180 \qquad (5.23)$$

Dengan derajat adalah nilai sudut yang diperoleh dalam derajat, radian adalah nilai sudut yang akan diolah dalam radian dan π = 3.14.

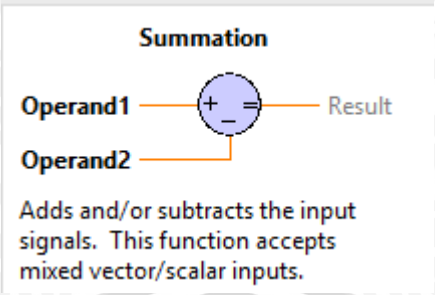
5.2.2 Implementasi PID kontroler

Sudut yang telah diperoleh dari pengolahan *inverse kinematics* akan dilakukan kontrol dengan menggunakan PID kontroler. Sama halnya dengan *inverse kinematics* proses pembuatan kontroler PID dilakukan berdasarkan rumus Implementasi PID kontroler dapat dilihat pada Gambar 5.8.



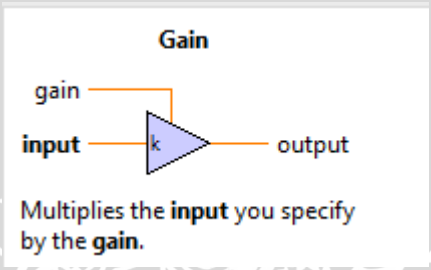
Gambar 5.8. Implementasi PID kontroler.

Gambar 5.8 menunjukkan proses implementasi PID controler dengan menggunakan *library integral* dan *derivative* milik labview. Gambar 5.8 merupakan satu kontrol PID untuk satu motor DC, jika sistem menggunakan dua motor DC maka membutuhkan dua kontroler PID namun dengan satu *input* nilai Kp, Ki, dan Kd. Pada Gambar 5.8 dapat dilihat terdapat fungsi *summation* yang digunakan untuk melakukan pengurangan input PID berupa sudut dengan *feedback* sudut hasil yang peroleh oleh sistem, hasil pengurangan inilah yang disebut sebagai *error* yang nantinya akan dikontrol dengan menggunakan PID kontroler terutama kontroler Ki yang berfungsi untuk mengilangkan *error steady state*, gambar dari fungsi *summation* beserta *input*-nya dapat dilihat pada Gambar 5.9



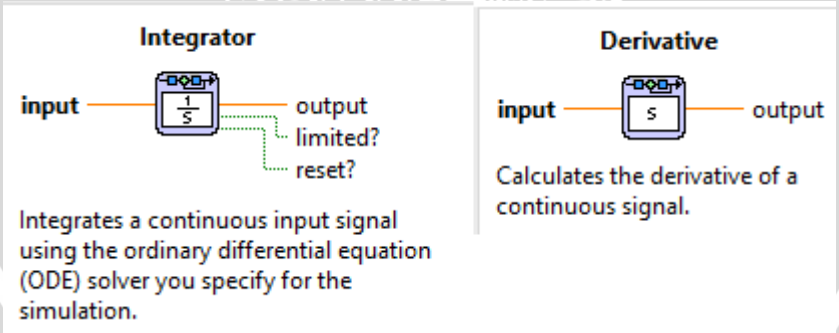
Gambar 5.9. Fungsi *summation* (sumber: ni.com)

Selain fungsi *summation*, juga digunakan fungsi *Gain* dimana fungsi *gain* ini merupakan fungsi yang memungkinkan adanya input lebih dari satu dengan hanya satu *output*, pada fungsi PID fungsi *gain* ini menerima dua *input* yaitu *input* sudut dan *input* kontroler yang digunakan. Gambar fungsi *gain* berserta *input*-nya dapat dilihat pada Gambar 5.10



Gambar 5.10. Fungsi *Gain* (sumber: ni.com)

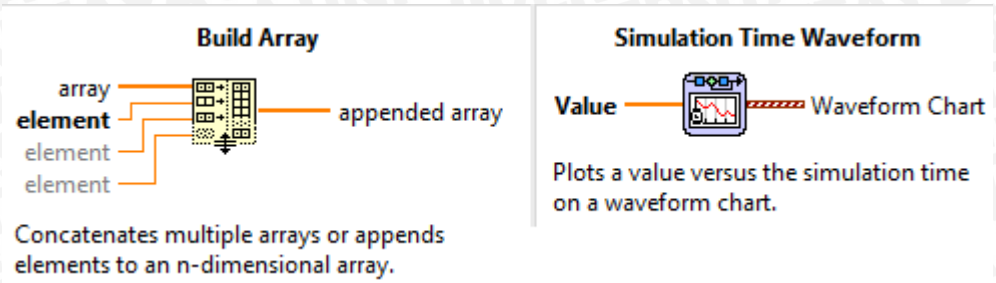
Selanjutnya fungsi yang digunakan adalah fungsi *integrator* dan *derivative*, fungsi ini telah disediakan oleh labview melalui modul *control and simulation loop* sehingga dapat digunakan dengan mudah. Gambar fungsi *integrator* dan *derivative* beserta fungsinya dapat dilihat pada Gambar 5.11.



Gambar 5.11. fungsi *integrator* dan *derivative* (sumber: ni.com)

Hasil *output* yang akan ditampilkan dengan menggunakan grafik adalah sudut *setpoint* yang diperoleh dari proses *inverse kinematics* koordinat yang di-*input* dan sudut yang mampu dibentuk sistem dari kontrol PID, dua nilai sudut ini dapat ditampilkan pada satu grafik dengan menggunakan *array*, ditampilkannya dua nilai sudut ini dalam satu grafik bertujuan agar *user* dapat dengan mudah membandingkan sudut *setpoint* dan respon sistem yang dihasilkan. Untuk proses pembacaan *array* menjadi gambar digunakan fungsi *simulation time waveform*

yang akan membaca *array* setiap waktunya untuk ditampilkan pada grafik. Fungsi *simulation time waveform* ini juga telah disediakan labview melalui *control and simulation* modul. Fungsi *build array* dan *simulation time waveform* beserta spesifikasi *input*-nya dapat dilihat pada Gambar 5.12.

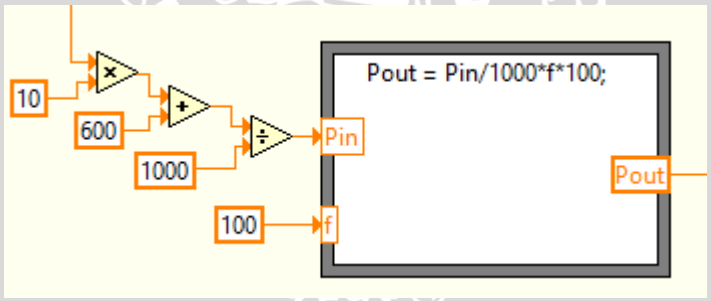


Gambar 5.12. Fungsi *build array* dan *simulation time waveform*

(sumber: ni.com)

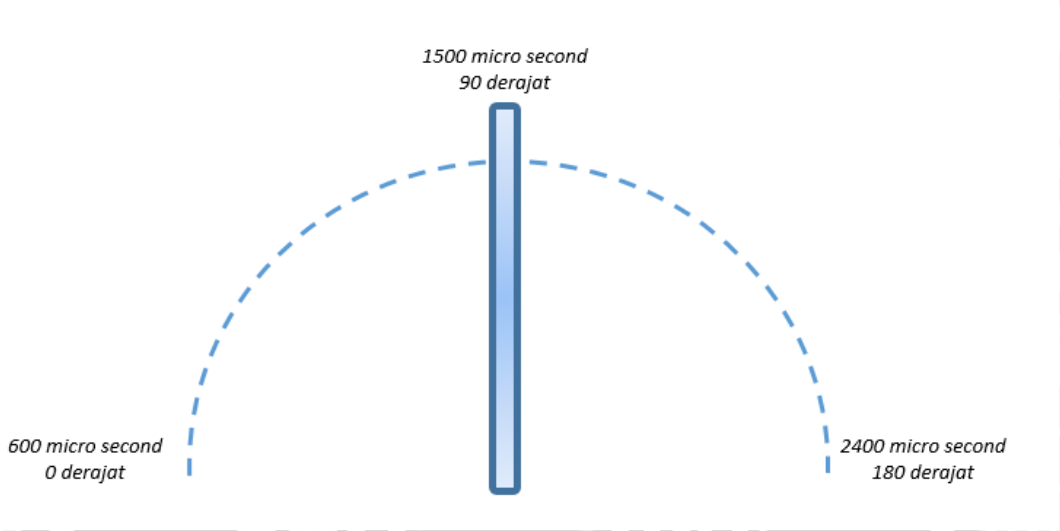
5.2.3 Implementasi PWM (*Pulse Width Modulation*)

Diperlukannya PWM (*Pulse Width Modulation*) dikarenakan aktuator yang digunakan berupa motor DC, Labview belum memiliki simulator maupun *mathScript* sebagai plant motor servo. Kendali motor servo pada labview biasanya langsung dilakukan pada *hardware* sehingga untuk mengatasi hal ini digunakanlah motor DC. Input dari motor DC adalah *voltage* karena input yang dimiliki sistem berupa sudut maka dari sudut inilah yang akan diubah menjadi PWM *pulse* dan menjadi input bagi motor DC. Implementasi PWM *pulse* dapat dilihat pada Gambar 5.13.



Gambar 5.13. Implementasi PWM *pulse.*

Pada Gambar 5.13 memiliki *output* berupa prosentase *duty cycle* yang akan digunakan untuk simulasi motor DC, input didapatkan dari nilai sudut yang dikonversikan menjadi PWM *input*. Berdasarkan pembahasan realtechsuport.com mengenai *servo control* pada *week* 12, PWM input yang digunakan merupakan PWM input pada servo motor HS311 yang mampu berputar 180^{o}, untuk dapat mendapatkan prosesntase *duty cycle* untuk simulasi motor DC maka diperlukan konversi sudut menjadi pulse PWM dalam satuan *milisecond* (ms) dari waktu PWM ini akan diubah untuk dijadikan prosentase *duty cycle*. Penjelasan proses konversi sudut menjadi waktu PWM dalam *milisecond* akan dibahas dengan gambar berikut



Gambar 5.14. servo angle

Berdasarkan datasheet dari servo motor HS311 sudut maksimal dari servo yang dapat dibentuk adalah 180^o dengan *maximum pulse* 2400 *microsecond* dan sudut minimal yang dapat dibentuk servo adalah 0^o dengan *minimum pulse* 600 *microsecond* maka dapat ditentukan

2400 µs – 600 µs = 1800 µs

180^o – 0^o = 180^o

Sehingga dalam 1^o perubahan sudut ditentukan pengali *pulse* sebesar

$$\frac{1800}{180} = 10$$

Maka konversi sudut ke pulse PWM dalam *milisecond* dapat dilakukan dengan rumus

Pulse Width = (val x 10) + minpulse (5.24)

Dengan val adalah besar sudut dan minpulse adalah *minimum pulse* yang digunakan yaitu 600 µs. Karena satuan yang diperlukan untuk *input* adalah *milisecond* maka diperlukan konversi lagi untuk mengubah *microsecond* menjadi *mili second* yaitu dengan menggunakan rumus

milisecond = $\frac{microsecond}{1000}$ (5.25)

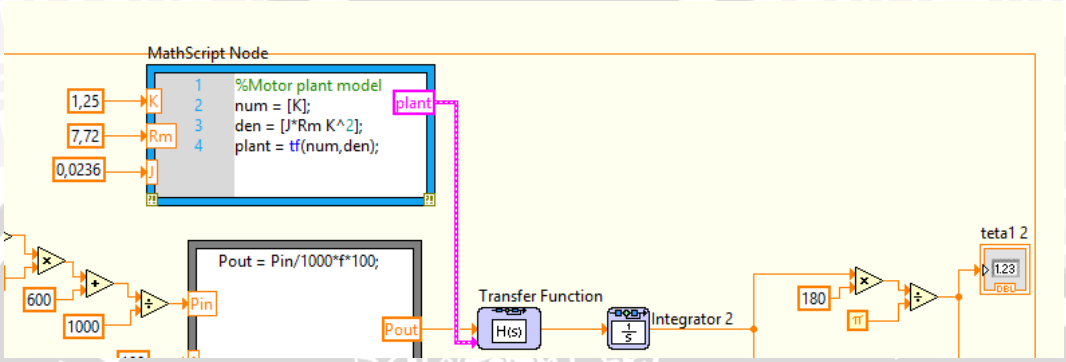
setelah didapatkan *input* berupa lebar pulsa PWM maka diperlukan konversi lagi untuk mendapatkan prosentase *duty cycle* guna *input* simulasi motor DC. Konversi menjadi *duty cycle* dapat dilakukan dengan menggunakan rumus

$$\%\ duty\ cycle = \frac{Pin}{1000}\ x\ f\ x\ 100 \quad (5.26)$$

Dengan Pin merupakan *Pulse input* dan f adalah frekuensi, dalam program ini digunakan frekuensi sebesar 100 Hz.

5.2.4 Implementasi simulator Motor DC

Dikarenakan simulator untuk motor servo masih belum tersedia di labview maka digunakan simulator motor DC dengan memanfaatkan *MathScript module* yang disediakan oleh labview. Cara penggunaan *MathScript* motor DC ini dengan memasukkan plant matematis motor DC pada *mathScript module* lalu di lakukan transfer function. Implementasi simulator motor DC dapat dilihat pada Gambar 5.15, dari kecepatan motor DC inilah yang akan di-*integralkan* untuk didapatkan nilai sudut hasil olah sistem yang telah dikontrol dengan PID.



Gambar 5.15. Implementasi plant motor DC

Gambar 5.15 menunjukkan simulasi motor DC dengan menggunakan *MathScript Node*, jenis motor yang dipakai dalam penelitian ini dipilih berdasarkan penelitian yang telah dilakukan oleh F.I Ahmed et all dengan judul "*P-I and I-P controllers in closed loop for motor DC drives*" yaitu Motor DC *Separately Excited* 0,5kW 220V dengan spesifikasi

Konstanta B-EMF (k) = 1,25 V/rad/s

Resistansi (Rm) = 7,72 Ω

Moment Inersia (j) = 0,0236

Pada Gambar 5.15 motor DC direpresentasikan dengan menggunakan *transfer function*, *transfer function* merupakan bentuk deskripsi sistem secara matematis yang membahas tentang hubungan antara *input* dan *output* sebuah sistem. Berdasarkan materi yang disajikan ni.com bentuk *transfer function* dari motor DC dapat direpresentasikan dengan rumus

$$\frac{\Omega m(s)}{Vm(s)} = \frac{Km}{JeqRms+Km^2} \quad (5.27)$$

Dengan

Km = *Motor back-EMF constant* (V/(rad/s))

Rm = *Motor armature resistance* (Ohms)

Jeq = *Equivalent moment of inertia* (kg*m^2)

Yang diasumsikan jeq = jm yaitu *Motor armature moment of inertia*. Model matematis ini dapat digunakan untuk desain *closed-loop controller*, Model matematis ini direpresentasikan dengan menggunakan *MathScript node* yang

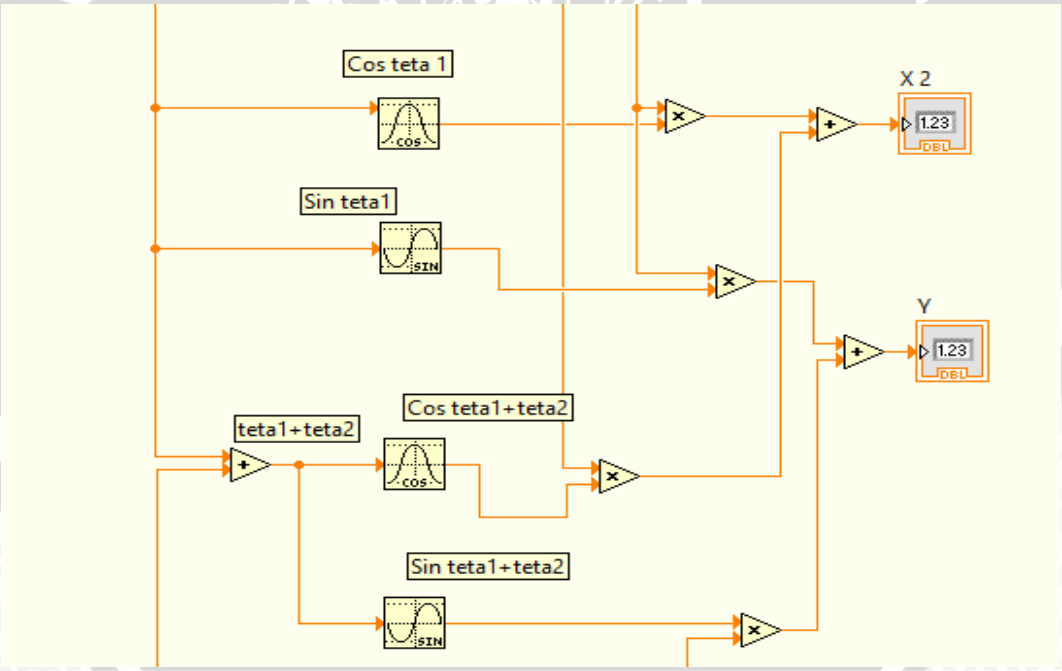
merupakan bagian dari *MathScript RT Module* dengan nilai input K, Rm dan J dari nilai Motor DC *separately excited* yang kemudian digunakan fungsi tf (*transfer function*), lalu dilakukan *transfer function* untuk dapat menghasilkan kecepatan motor DC dengan rumus

$$H(s) = \frac{1}{s+1} \quad (5.28)$$

Setelah didapatkan kecepatan motor DC maka untuk dapat menentukan sudut yang dibentuk digunakan *integrator*, perlu diingat bahwa kecepatan motor DC yang didapatkan memiliki satuan rad/s sehingga sudut hasil *integrator* adalah radian untuk dapat dilihat perubahan sudutnya harus dilakukan konversi dari radian ke derajat dengan cara sudut dikalikan 180 dan dibagi phi atau 3,14.

5.2.5 Implementasi *Forward Kinematics*

Pada sistem ini *forward kinematics* digunakan untuk mengubah sudut pada masing-masing *joint* menjadi titik koordinat yang mampu diraih sistem, hal ini bertujuan agar *user* dapat membandingkan *input* koordinat yang diinginkan dan koordinat yang dihasilkan oleh sistem. Implementasi *forward kinematics* dilakukan dengan penulisan rumus secara langsung, implementasi *forward kinematics* dapat dilihat pada Gambar 5.16.



Gambar 5.16. Implementasi *forward kinematics*

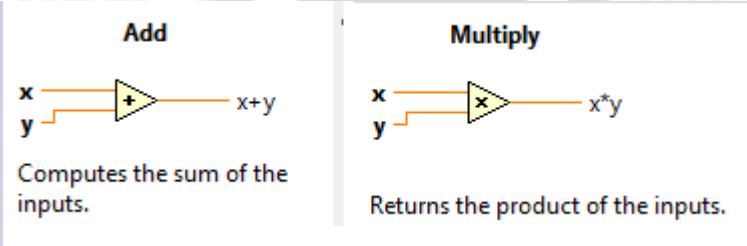
Gambar 5.16 menunjukkan implementasi *forward kinematics* sesuai dengan rumus 5.5 dan 5.6 yaitu

$$X = L1 \cos\theta_1 + L2 \cos(\theta_1 + \theta_2)$$

$$Y = L1 \sin\theta_1 + L2 \sin(\theta_1 + \theta_2)$$

Pada Gambar 5.11 merupakan implementasi dari rumus 5.5 dan 5.6, proses imlementasi menggunakan fungsi *add, sinus, cosinus* dan *multtiply*. Cara kerja

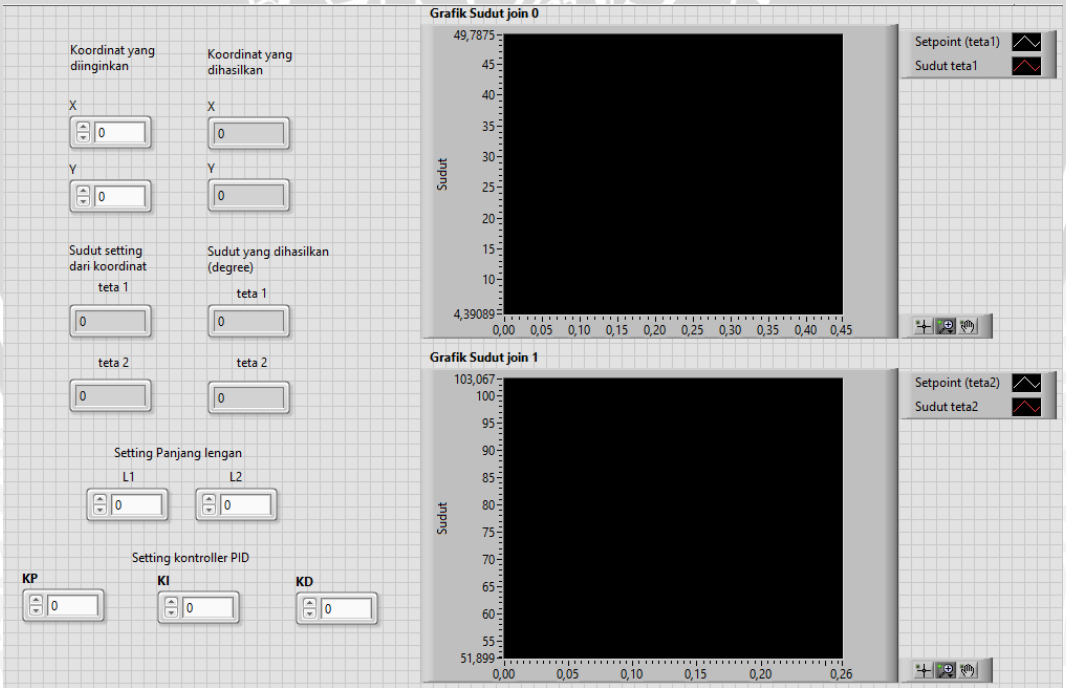
fungsi *add* dan *multiply* ini sama seperti fungsi matematis pada umumnya namun sedikit berbeda pada fungsi *sinus* dan *cosinus*, telah dijelaskan bahwa fungsi *sinus* dan *cosinus* memiliki *input* dan *ouput* berupa radian, namun pada implementasi *forward kinematics* tidak diperlukan adanya konversi dari radian ke derajat dikarenakan yang diperlukan adalah besar sudutnya. Unttuk fungsi *add* dan *multiply* sendiri memiliki cara kerja yang sama seperti matematika pada umumnya, spesifikasi *input* dan *output* fungsi *add* dan *multiply* dapat dilihat pada Gambar 5.17.



Gambar 5.17. fungsi add dan multiply (sumber: ni.com)

5.2.6 Implementasi Desain *Front Panel*

Implementasi desain *front panel* dilakukan sesuai dengan perancangan yang telah dilakukan pada Sub Bab 5.1.2 dengan tujuan untuk mempermudah *user* dalam penggunaan aplikasi. Implementasi desain *front panel* dapat dilihat pada Gambar 5.18.



Gambar 5.18. Implementasi desain *front panel*

Pada Gambar 5.18 merupakan implementasi desain *front panel* yang akan berinteraksi secara langsung dengan *user.* Pada pembuatan *front panel* ini digunakan jenis *silver control* and *indicator*, dipilihnya *type silver* ini dikarenakan

tampilannya yang rapi dengan memiliki *border* disetiap tampilan kontrol dan indikatornya. Selain *type control* dan *indicator silver* digunakan juga *waveform chart* karena grafik tipe ini memiliki tampilan yang lebih baik dann menarik dalam menampilkan nilai berupa perbandingan sudut.

5.2.7 Implementasi Pembuatan *Installer*

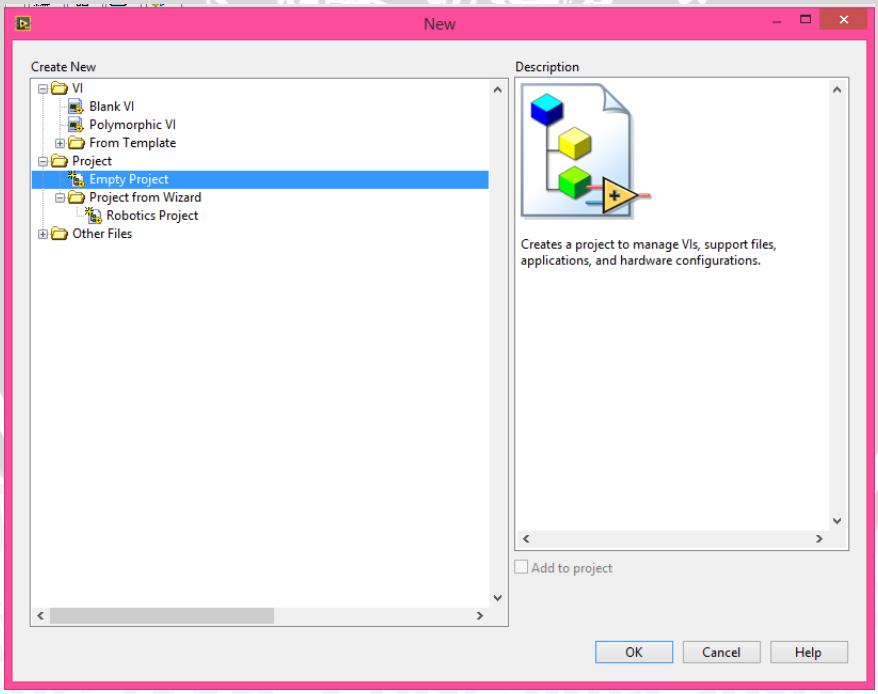
Pada tahap ini akan dilakukan proses pembuatan *installer* dari aplikasi, sehingga *user* tidak perlu memiliki *software* labview untuk dapat menggunakan aplikasi. Pembuatan *installer* ini memiliki beberapa tahap yaitu

1. Pembuatan *file Executable*
2. Proses *Building Installer*
3. Cara *install* dan *unistall program*

5.2.7.1 Pembuatan *file executable*

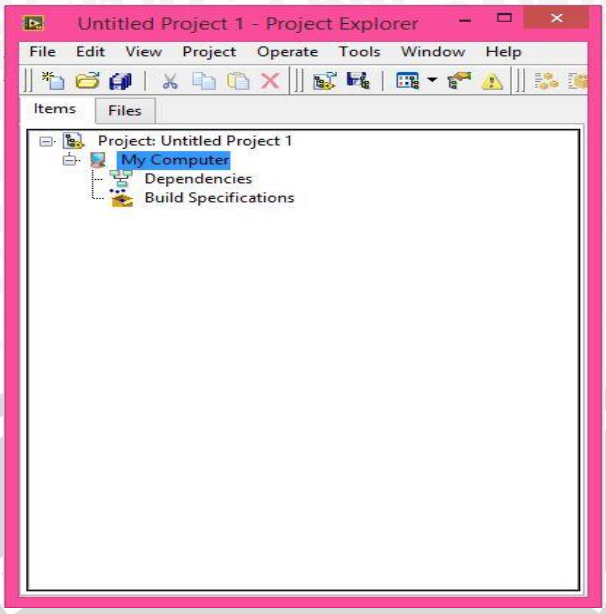
Tahap pertama dalam proses pembuatan *installer* adalah membuat *file executable* dengan format *file* .exe. Pembuatan *file* dengan format .exe ini akan menjadi *file aplication* simulator yaitu *file* yang akan menjalankan aplikasi. Sebelum membuat *file executable* harus dibuat *blank project* pada Labview National Instrument. Berikut Langkah-langkah dalam pembuatan *file executable*

1. Buat *blank project* pada Labview dengan cara klik *file -> new* pada toolbar labview maka akan keluar tab seperti Gambar 5.19.



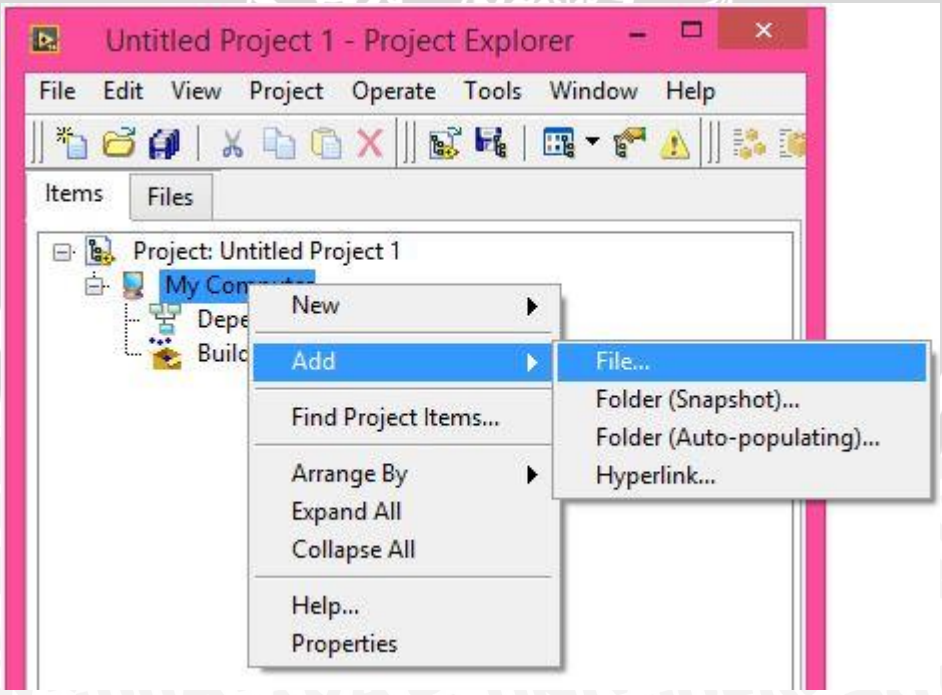
Gambar 5.19. *Tab new*

Pada Gambar 5.19 *expand file* project dan pilih *empty project* lalu klik "*OK*" tampilan *empty project* dapat dilihat pada Gambar 5.20

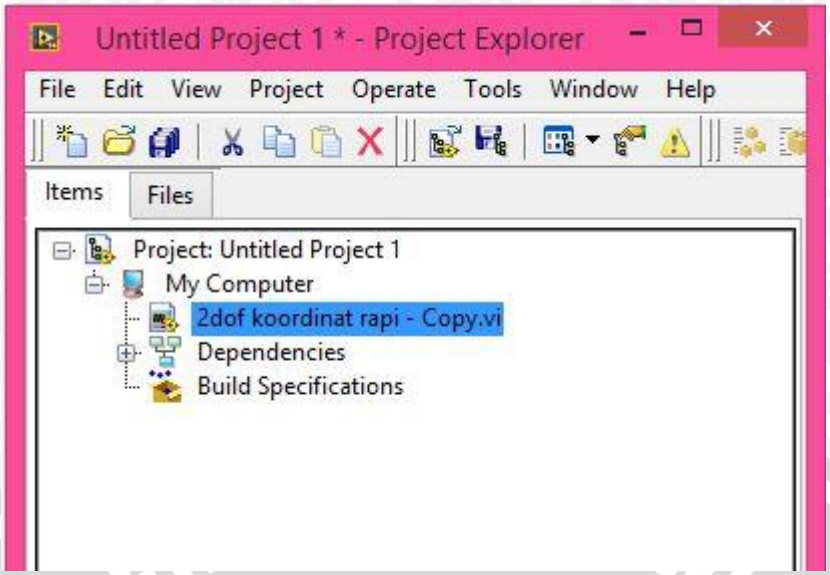


Gambar 5.20. *Empty project*

2. Setelah proses pembuatan *empty project* selesai, masukkan seluruh *file* yang berhubungan dengan aplikasi pada *empty project* yang telah dibuat dengan cara klik kanan pada *My Computer* di *empty project -> add -> File*, lalu pilih file yang terkait dengan simulator yang akan dijadikan *installer*. Langkah kedua ini dapat dilihat pada Gambar 5.21 dan hasil dari langkah 2 dapat dilihat pada Gambar 5.21.

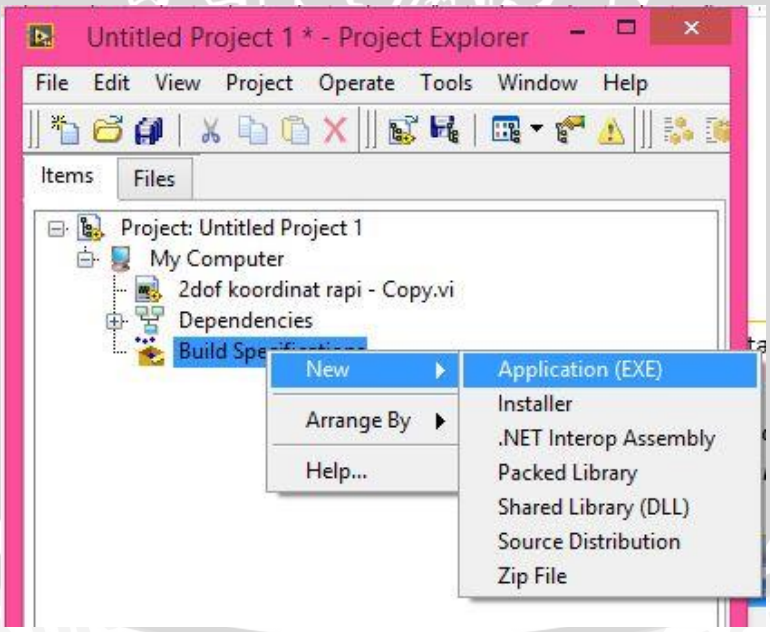


Gambar 5.21. *Add file*

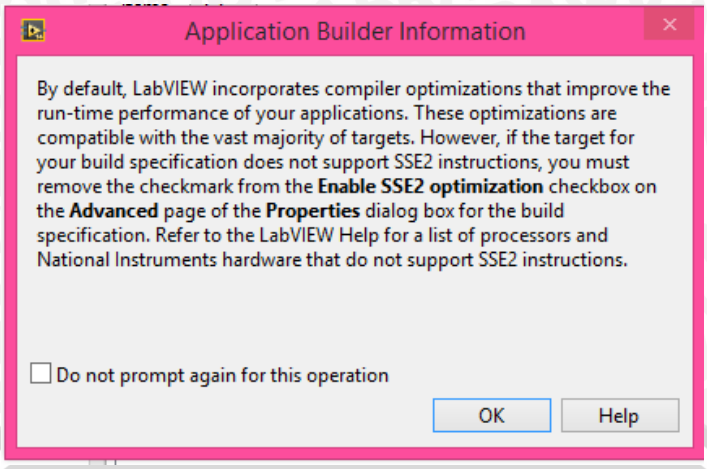


Gambar 5.22. Hasil *Add file*

3. Langkah ketiga adalah proses pembuatan aplikasi .exe dilakukan dengan cara klik kanan pada *build spesification* -> new -> Aplication (EXE), sebelumnya *save* terlebih dahulu *empty project*, langkah *aplication* (EXE) akan bertujuan untuk membangun *aplication* dengan format .exe. Langkah ini dapat dilihat pada Gambar 5.23, setelah klik *Aplication* (EXE) maka keluar *tab* seperti pada Gambar 5.24 lalu klik OK

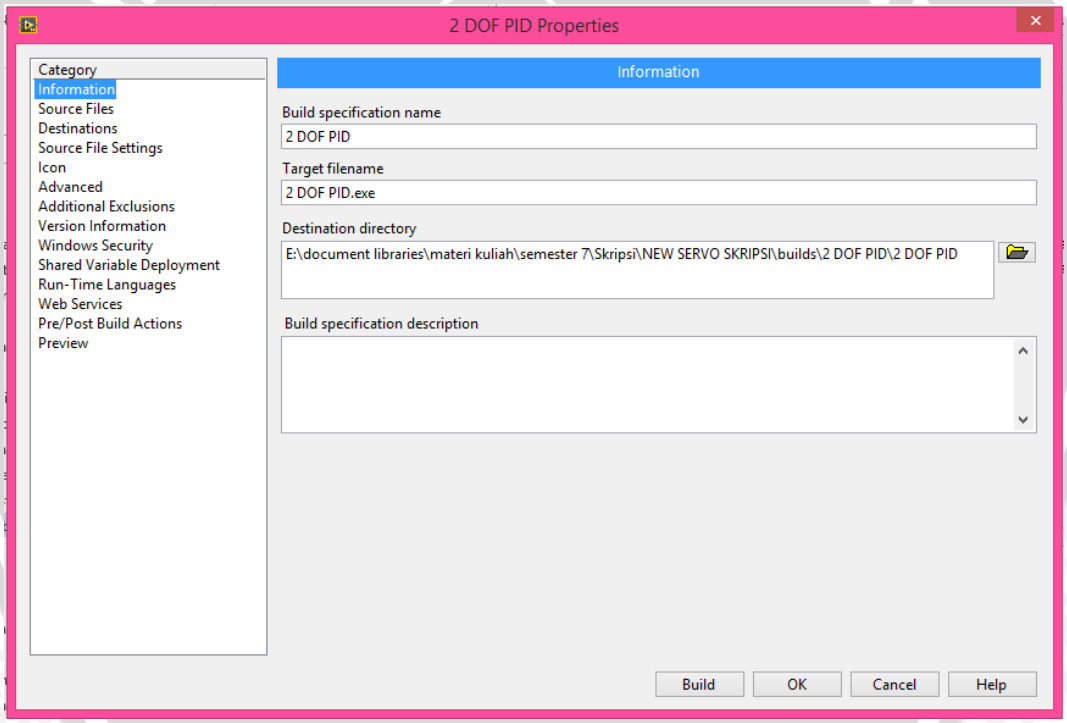


Gambar 5.23. *New Application* (EXE)



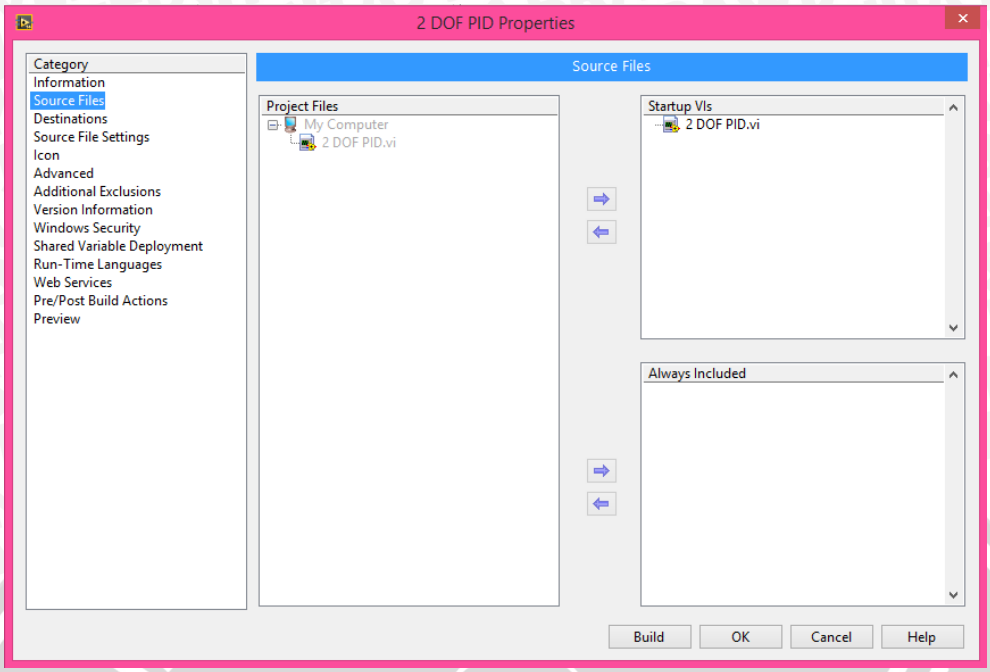
Gambar 5.24. *Application builder information*

Setelah klik OK maka akan keluar tab *properties* untuk *aplication* .exe yang akan dibangun seperti pada Gambar 5.25 Ubah nama aplikasi dan target *file* sesuai yang diinginkan, pada sistem ini penulis menggunakan nama 2 DOF PID

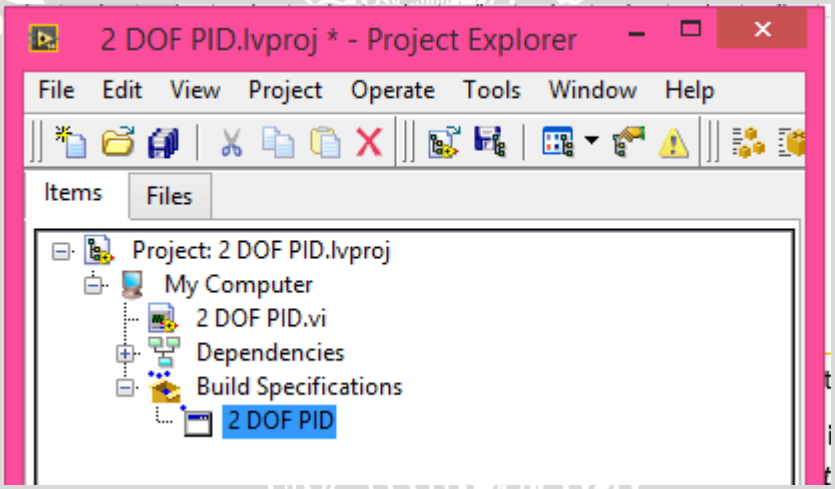


Gambar 5.25. *Tab properties application* (EXE)

Tentukan *startup* Vis pada *tab Source Files* seperti pada Gambar 5.26 dengan cara klik Vi pada *project File* yang diinginkan untuk jadi *startup* VI, karena VI yang diperlukan hanya satu maka *startup* yang digunakan juga *file* 2 DOF PID.vi, setelah klik 2 DOF PID.vi lalu klik panah yang mengarah menuju *windows startup* Vis lalu klik *Build* dan tunggu hingga proses *build* selesai. Gambar 5.27 menunjukkan hasil dari proses *building*.



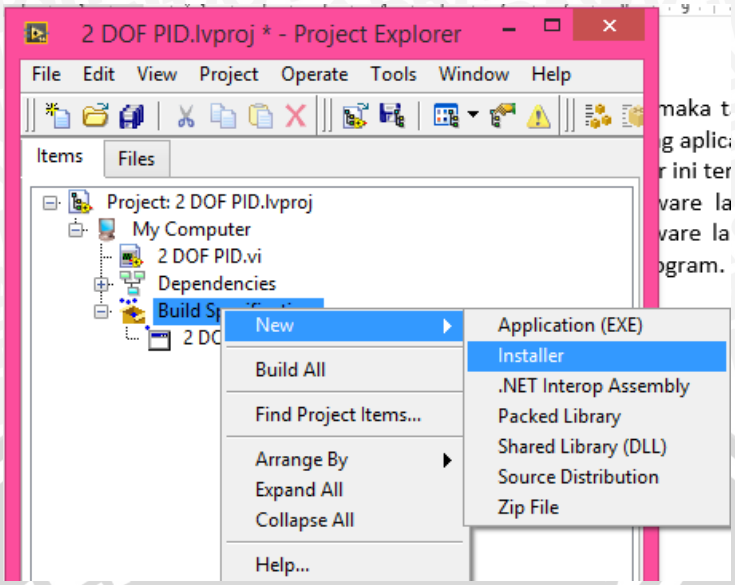
Gambar 5.26 *Tab source file*



Gambar 5.27. Hasil proses *building Application (EXE)*

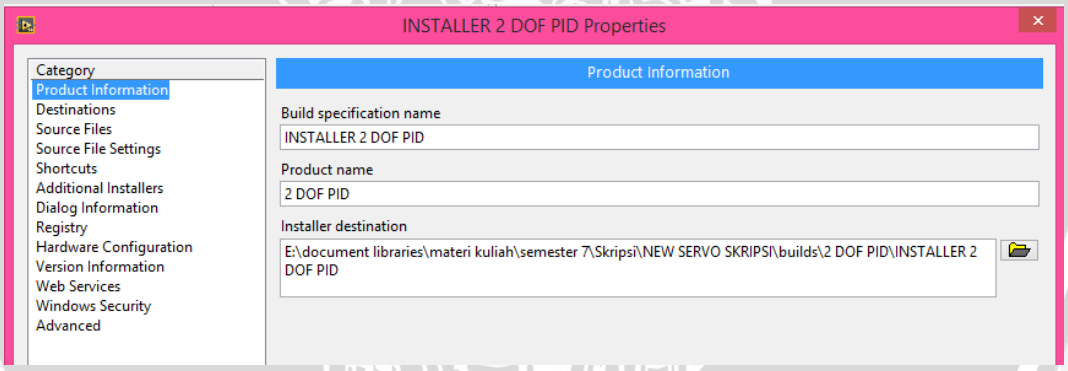
5.2.7.2 Building Installer

Setelah proses pembuatan *file* .exe selesai maka tahap selanjutnya adalah pembuatan *installer* yang digunakan untuk *running aplication* .exe. aplikasi 2 DOF PID.exe tidak akan dapat di *running* tanpa *installer* ini terkecuali pada laptop atau komputer yang digunakan telah ter-*install software* labview, jika laptop atau komputer yang digunakan belum memiliki *software* labview maka *installer* ini wajib di-*install* untuk dapat melakukan *running* program. Langkah awal yang harus dilakukan adalah dengan klik kanan *build spesification -> new -> installer*, langkah ini dapat dilihat pada Gambar 5.28.

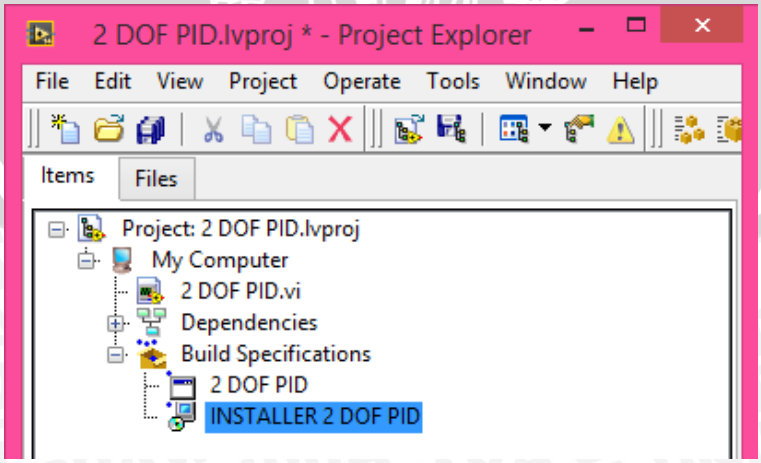


Gambar 5.28. *new installer*

Setelah proses ini maka akan keluar tab properties untuk *installer*, *setting* sesuai dengan kebutuhan sama seperti saat *tab properties* aplikasi .exe lalu klik *Build*. *Tab properties installer* dapat dilihat pada Gambar 5.29 hasil dari proses pembuatan dapat dilihat pada Gambar 5.30.



Gambar 5.29. *Tab properties Installer*

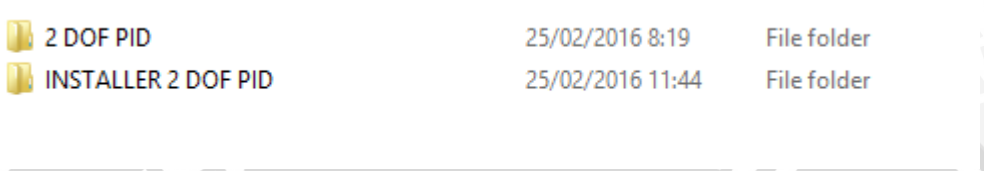


Gambar 5.30. Hasil *build installer*

5.2.7.3 Cara *install* dan *unistall* Program

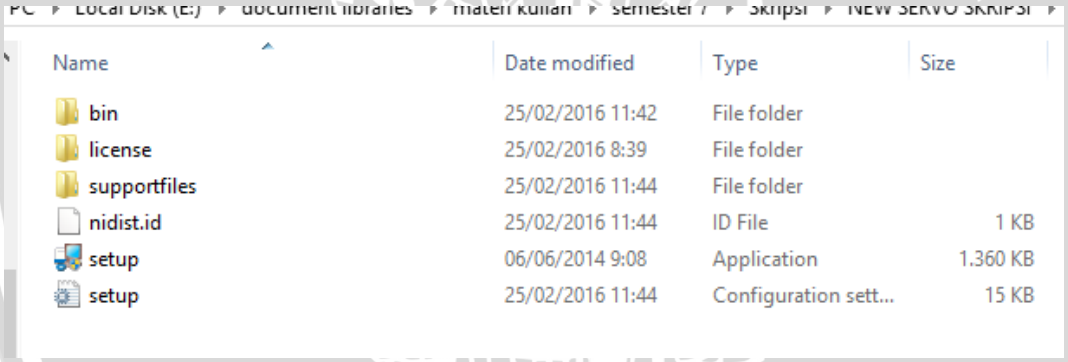
1. Cara *Install* Program *run-time enginee* Labview

Cara *install* program ini sama seperti *install* program pada umumnya, setelah proses pembuatan *installer* maka pada *folder build* akan terdapat 2 buah folder yaitu *folder aplication* dan *folder installer* yang telah dibuat. *Aplication* .exe terdapat pada *folder* 2 DOF PID seddangkan *installer* untuk labview *run-time enginee* terdapat di *folder* INSTALLER 2 DOF PID. Aplication .exe tidak akan dapat di *running* jika *run-time enginee* labview tidak di-*install*, *folder* hasil pembuatan aplikasi dan *installer* dapat dilihat pada Gambar 5.31.

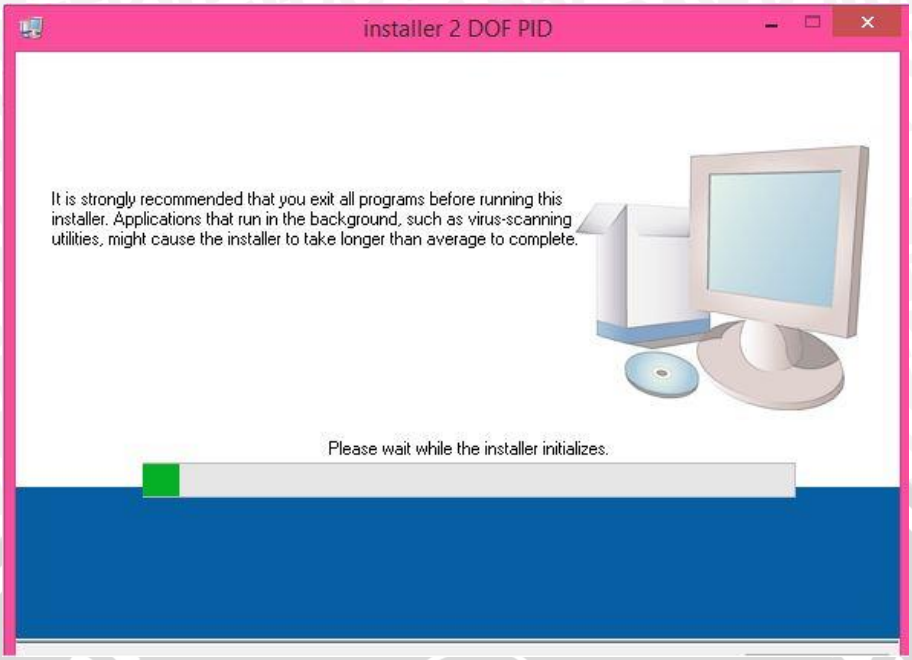


Gambar 5.31. *Folder* hasil pembuatan aplikasi dan *installer*

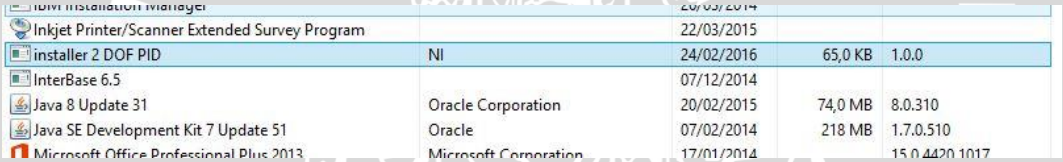
Untuk dapat melakukan instalasi *run autosetup* pada *folder installer* seperti pada Gambar 5.32, Lalu ikuti instruksi instalasi hingga selesai. Proses instalasi dapat dilihat pada Gambar 5.33.



Gambar 5.32. *setup* pada *file installer*



Gambar 5.33. Proses *installing*



Gambar 5.34. Hasil *installing*

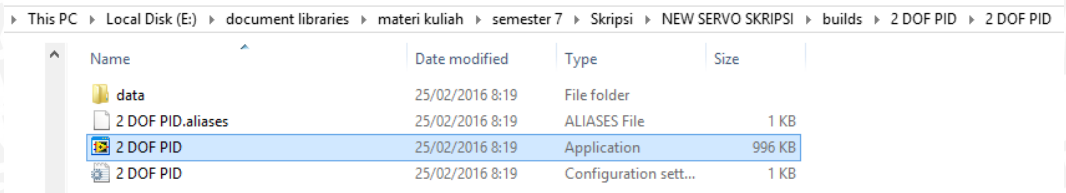
Gambar 5.34 menunjukkan bahwa *run-time enginee* labview yang akan digunakan untuk *running* aplikasi telah ter-*install* pada laptop. Setelah ter-*install run-time enginee* ini maka aplikasi 2 DOF PID dapat di-*running* tanpa harus memiliki *software* labview.

2. Cara *Uninstall* Program

   Cara *uninstall* program aplikasi ini sama dengan program pada umumnya yaitu lewat *front panel*, pilih aplikasi *installer* 2 DOF PID lalu klik *unistall*.

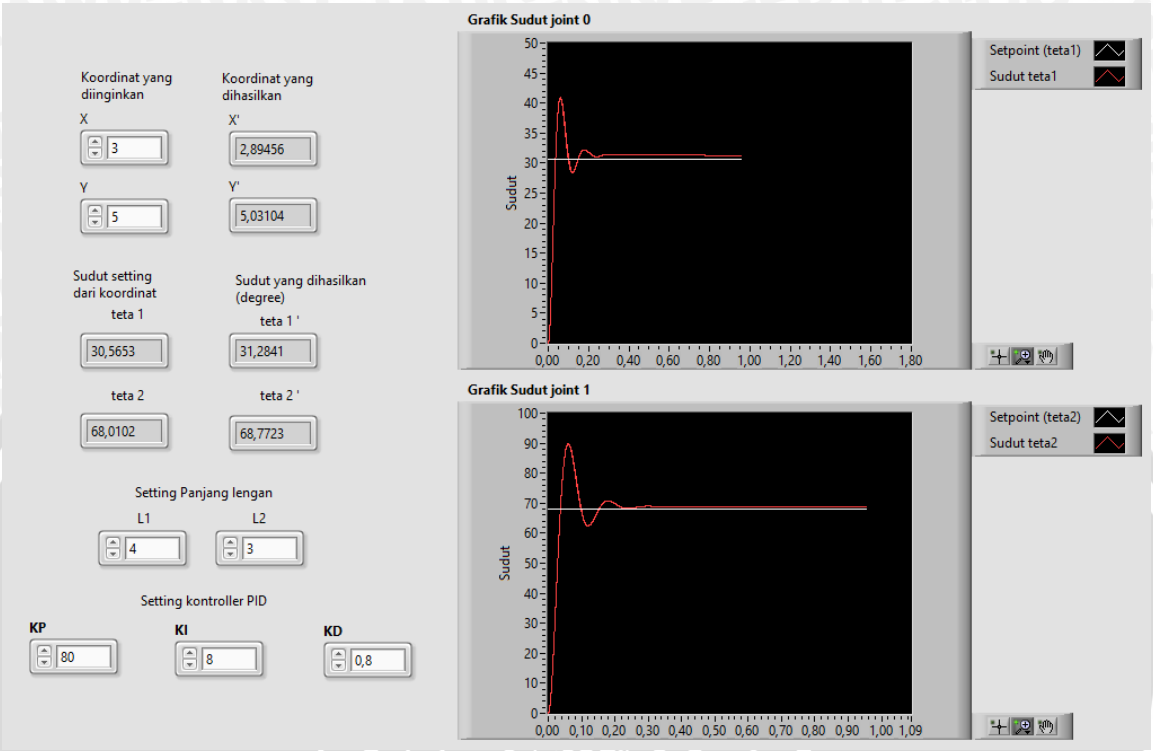
5.2.8 Cara penggunaan aplikasi

Setelah melakukan instalasi *run-time enginee* labview maka aplikasi dapat dijalankan, Cara menjalankan aplikasi dengan *double* klik pada application 2 DOF PID seperti pada Gambar 5.35.



Gambar 5.35. Aplikasi 2 DOF PID

Saat dijalankan maka *default* aplikasi akan *autorun* namun nilai yang ditunjukkan adalah NAN (*Not a Number*), hal ini dikarenakan aplikasi belum memiliki *input* baik dari koordinat, panjang lengan maupun nilai PID sehingga tidak ada nilai yang diolah. Untuk dapat menggunakan aplikasi dengan baik maka perlu diperhatikan fungsi dari masing-masing bagian pada *front panel* aplikasi berikut



Gambar 5.36. Bagian *front panel*

Pada Gambar 5.36 dapat dilihat bahwa *front panel* memiliki beberapa bagian yang digunakan untuk menjalankan aplikasi diantaranya:

A. Bagian A merupakan bagian control dimana *user* akan memasukkan *input* koordinat yang diinginkan.

B. Bagian B merupakan bagian indikator yaitu bagian yang memperlihatkan titik koordinat akhir yang mampu dicapai oleh sistem. Koordinat ini diperoleh dari proses *forward kinematics* sudut yang dihasilkan oleh sistem.

C. Bagian C menunjukkan besar sudut pada *joint* 0 dan *joint* 1 untuk dapat mencapai titik koordinat tujuan. Besar sudut pada bagian C diperoleh dari proses *inverse kinematics* sistem koordinat yang dimasukkan oleh *user*.

D. Bagian D menunjukkan besar sudut pada *joint* 0 dan *joint* 1 yang dihasilkan oleh sistem. Besar sudut bagian D didapat dari integral kecepatan motor DC.

E. Bagian E merupakan bagian control dimana user dapat emasukkan panjang lengan untuk *link* 1 dan *link* 2. Perlu diperhatikan bahwa panjang lengan ini akan menentukan titik koordinat yang mampu dicapai, apabila titik

koordinat yang dimasukkan oleh *user* tidak dapat dicapai oleh lengan maka sistem akan *error* dengan indikasi sistem menghasilkan NAN (*Not a Number*).

F. Bagian F menunjukkan bagian *control* untuk nilai PID, Pada bagian F inilah user akan mengubah nilai PID hingga respon sistem sesuai dengan apa yang diinginkan.

G. Bagian G merupakan grafik yang menunjukkan respon sistem yang mendekati *setpoint* pada *joint* 0. Grafik ini menunjukkan 2 *line* yaitu *line* berwarna putih yang menunjukkan *setpoint* sudut yang berasal dari *inverse kinematics* koordinat yang diinput *user* dan *line* berwarna merah yang merupakan sudut dari hasil olah sistem.

H. Bagian H merupakan grafik yang menunjukkan respon sistem yang mendekati *setpoint* pada *joint* 1. Grafik ini menunjukkan 2 *line* yaitu *line* berwarna putih yang menunjukkan *setpoint* sudut yang berasal dari *inverse kinematics* koordinat yang diinput *user* dan *line* berwarna merah yang merupakan sudut dari hasil olah sistem.

I. Bagian I merupakan *Plot Legend*, yaitu indikator yang menunjukkan *line* yang terdapat dalam grafik *joint* 0.

J. Bagian J merupakan *Graph Pallete* yang dapat digunakan untuk memanipulasi grafik seperti *zoom in* dan *zoom out* pada grafik *joint* 0.

K. Bagian I merupakan *Plot Legend*, yaitu indikator yang menunjukkan *line* yang terdapat dalam grafik *joint* 1.

L. Bagian I merupakan *Graph Pallete*, yaitu indikator yang menunjukkan *line* yang terdapat dalam grafik *joint* 1.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan dilakukan proses pengujian sistem yang bertujuan untuk mengetahui performa dan tingkat akurasi sistem serta memastikan bahwa apakah sistem akan berjalan dengan baik. Proses pengujian ini memiliki dua tahap yaitu pengujian perubahaan nilai koordinat dengan nilai PID yang saa dan pengujian nilai koordinat yang sama dengan nilai PID yang berbeda.

6.1 Sistematika pengujian

Sebelum melakukan pengujian terlebih dahulu diperlukan kejelasan tentang sistematika pengujian yang digunakan. Telah disebutkan bahwa pengujian terdiri dari dua tahap yaitu pengujian nilai koordinat dengan nilai PID tetap dan pengujian nilai PID dengan nilai koordinat yang tetap. Data yang akan diambil ada proses pengujian adalah *settling time* dan *error steady state* dari nilai koordinat yang dihasilkan. *Settling time* (waktu tunak) adalah ukuran waktu dimana respon sistem telah mencapai kurang lebih 5% atau kurang lebih 2% dibanding dengan nilai *steady state*-nya, sedangkan *error steady state* merupakan selisih nilai akhir dengan *setpoint* yang diinginkan. Penghitungan *Settling time* dilakukan dengan menggunakan rumus

$$\textit{Settling time} = \frac{Sudut\ perolehan - Sudut\ setpoint}{Sudut\ setpoint}\ x\ 100\% \quad (6.1)$$

Sedangkan untuk *error steady state* dari titik koordinat diperoleh dengan menggunakan rumus

$$Ess\ X = X' - X \quad (6.2)$$

$$Ess\ Y = Y' - Y \quad (6.3)$$

Dengan :

X = Sumbu X dari titik koordinat *setpoint*.

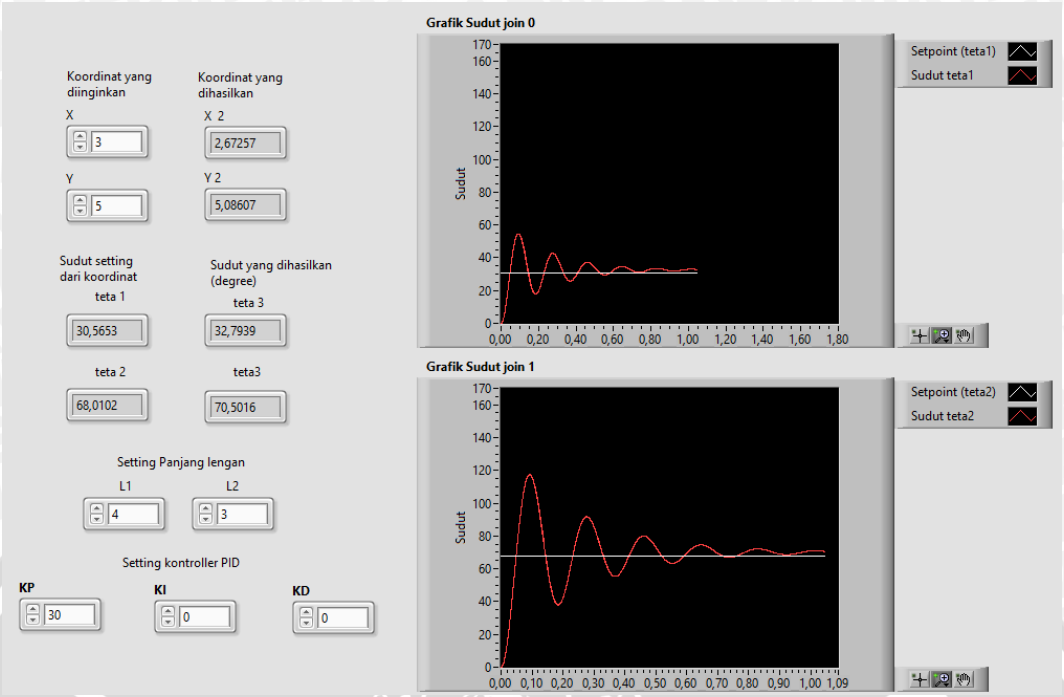
Y = Sumbu Y dari titik koordinat *setpoint*.

X’ = Sumbu X dari hasil olah sistem.

Y’ = Sumbu Y dari hasil olah sistem.

6.2 Pengujian nilai PID

Pengujian nilai PID ini bertujuan untuk melihat respon sistem ketika memiliki *setpoint* koordinat tetap dengan nilai PID yang berbeda.



Gambar 6.1. Respon sistem dengan Kp = 30 Ki = 0 dan Kd = 0

Untuk dapat menentukan nilai PID yang tepat untuk sistem dengan *plant* motor tertentu maka cara yang digunakan adalah dengan menguji satu persatu nilai Kp, Ki, dan Kd yang mungkin digunakan. Ketika melakukan pengujian untuk mendapatkan nilai Kp yang tepat maka nilai Ki dan nilai Kd di-*set* menjadi 0 sehingga dapat benar-benar dilihat pengaruh nilai Kp terhadap sistem. Begitu juga saat melakukan pengujian nilai Ki maka nilai Kp di-*set* dengan nilai yang terbaik berdasarkan *settling time*-nya dan nilai Kd di-*set* 0 sehingga dapat diketahui pengaruh nilai Ki dan Kp terhadap sistem. Untuk pengujian nilai Kd maka nilai Kp dan Ki akan di-*set* ke nilai terbaik berdasarkan *settling time* yang diperoleh.

Pada Gambar 6.1 menunjukkan respon sistem saat pengujian pengaruh nilai Kp dengan nilai X = 3, Y = 5, panjang *link* 1 = 4 cm, panjang *link* 2 = 3 cm dan nilai Kp = 30 Ki = 0 dan Kd = 0. Menunjukkan respon sistem yang kurang stabil dalam mendekati setpoint, sehingga untuk nilai Kp = 0,5 dinilai kurang bagus untuk dijadikan kontrol. Hasil pengujian nilai Kp dengan titik koordinat X = 3, Y = 5, panjang *link* 1 = 4 cm dan panjang *link* 2 = 3 cm selengkapnya dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian nilai Kp dengan X = 3, Y = 5, L1 = 4cm, L2 = 3cm

Kp	Ki	Kd	% St0	% St1	St 0	St	X'	Y'	Ess x	Ess y
1	0	0	200	88,2	1,2	1,24	-2,3	2,12	5,3	2,88
9	0	0	21,9	9,7	1,1	1,3	2,06	5,20	0,94	-0,20
15	0	0	13,11	5,88	1,05	1,08	2,43	5,14	0,57	-0,14
21	0	0	9,5	4,11	1,15	1,35	2,59	5,11	0,41	-0,11
25	0	0	7,86	3,52	1,2	1,3	2,66	5,09	0,34	-0,09
30	0	0	6,5	4,4	1,12	1,1	2,71	5,08	0,29	-0,08
39	0	0	5	4,4	1,59	0,9	2,7	5,06	0,3	-0,06
42	0	0	8,1	4,4	1,3	0,98	2,79	5,06	0,21	-0,06
48	0	0	4,91	4,4	1,28	0,9	2,82	5,05	0,18	-0,05
50	0	0	4,91	4,4	1,2	0,8	2,82	5,05	0,18	-0,05

Keterangan tabel:

Kp = nilai kontroler proporsional

Ki = nilai kontroler integral

Kd = nilai kontroler *derivative*

%St0 = prosentase selisih nilai sudut *joint* 0 untuk *settling time*

%St1 = prosentase selisih nilai sudut *joint* 1 untuk *settling time*

St 0 = *settling time joint* 0

St 1 = *settling time joint* 1

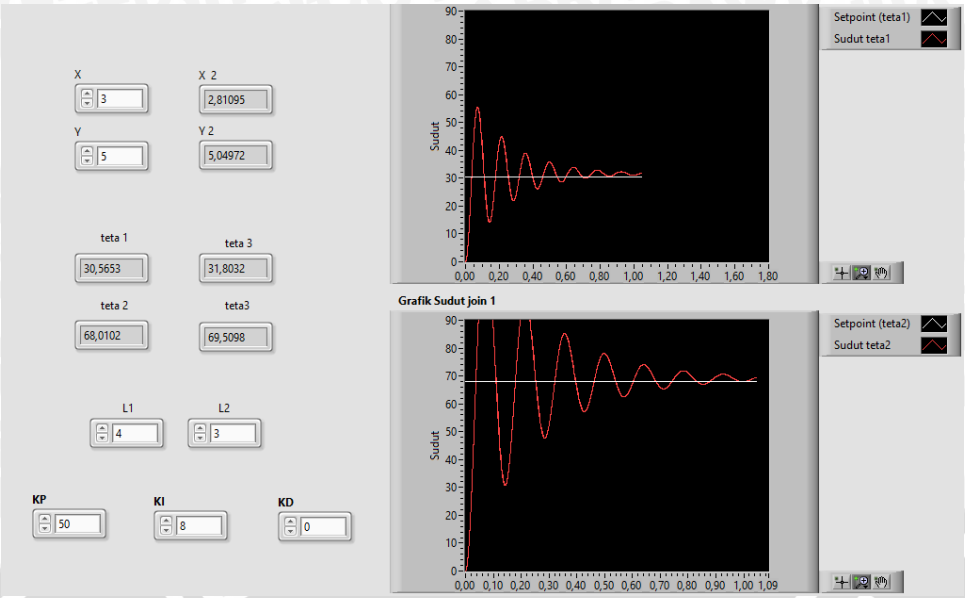
X' = nilai sumbu koordinat x hasil olah sistem

Y' = nilai sumbu koordinat y hasil olah sistem

Ess x = nilai error steady sumbu koordinat x

Ess y = nilai error steady sumbu koordinat y

Tabel 6.1 merupakan tabel hasil pengujian Kp, pada tabel diatas terlihat pengaruh nilai Kp terhadap respon sistem diantaranya nilai *error steady state* yang semakin berkurang. Dapat diambil contoh pada nilai Kp 1 dan nilai Kp 15, sewaktu Kp bernilai 1 *error* pada *joint* 0 mencapai 5,3 sedangkan ketika Kp bernilai 15 *error* pada *joint* 0 hanya 0,57, hal ini membuktikan bahwa semakin besar nilai Kp maka semakin kecil *error* yang dihasilkan. Namun untuk *settling time* Kp tidak memiliki pengaruh yang *significant* hal ini dapat dilihat ketika nilai Kp 1 *joint* 0 mencapai *settling time* pada 1,2 *second* sedangkan ketika nilai Kp 15 *joint* 0 mencapai *settling time* pada 1,05 *second*.



Gambar 6.2. Respon sistem dengan Kp = 50 Ki = 8 dan Kd = 0

Pada Gambar 6.2 menunjukkan respon sistem saat pengujian pengaruh nilai Ki dengan nilai X = 3, Y = 5, panjang *link* 1 = 4 cm, panjang *link* 2 = 3 cm dan nilai Kp = 50 Ki = 8 dan Kd = 0. Menunjukkan respon sistem yang kurang stabil dalam mendekati setpoint. Hasil pengujian nilai Ki dengan titik koordinat X = 3, Y = 5, panjang *link* 1 = 4 cm dan panjang *link* 2 = 3 cm selengkapnya dapat dilihat pada Tabel 6.2.

Tabel 6.2. Tabel hasil pengujian nilai Ki dengan X = 3, Y = 5, L1 = 4cm, L2 = 3cm

Kp	Ki	Kd	% St0	% St1	St 0	St	X'	Y'	Ess x	Ess y
50	0,01	0	4,91	4,4	1,02	0,88	2,80	5,05	0,20	-0,05
50	0,05	0	4,91	4,4	1,16	0,87	2,85	5,04	0,15	-0,04
50	0,1	0	4,91	4,4	1,16	0,86	2,82	5,05	0,18	-0,05
50	0,5	0	4,91	4,4	1,30	0,86	2,83	5,05	0,17	-0,05
50	1	0	4,91	4,4	1,34	0,90	2,83	5,04	0,17	-0,04
50	3	0	4,91	4,4	1,35	0,90	2,84	5,04	0,16	-0,04
50	5	0	4,91	4,4	1,17	0,89	2,85	5,04	0,15	-0,04
50	8	0	4,91	4,4	1,00	0,87	2,87	5,03	0,13	-0,03
50	10	0	4,91	4,4	1,17	0,90	2,87	5,03	0,13	-0,03
50	15	0	4,91	4,4	1,03	1,02	2,87	5,03	0,13	-0,03

Keterangan tabel:

Kp = nilai kontroler proporsional

Ki = nilai kontroler integral

Kd = nilai kontroler *derivative*

%St0 = prosentase selisih nilai sudut *joint* 0 untuk *settling time*

%St1 = prosentase selisih nilai sudut *joint* 1 untuk *settling time*

St 0 = *settling time joint* 0

St 1 = *settling time joint* 1

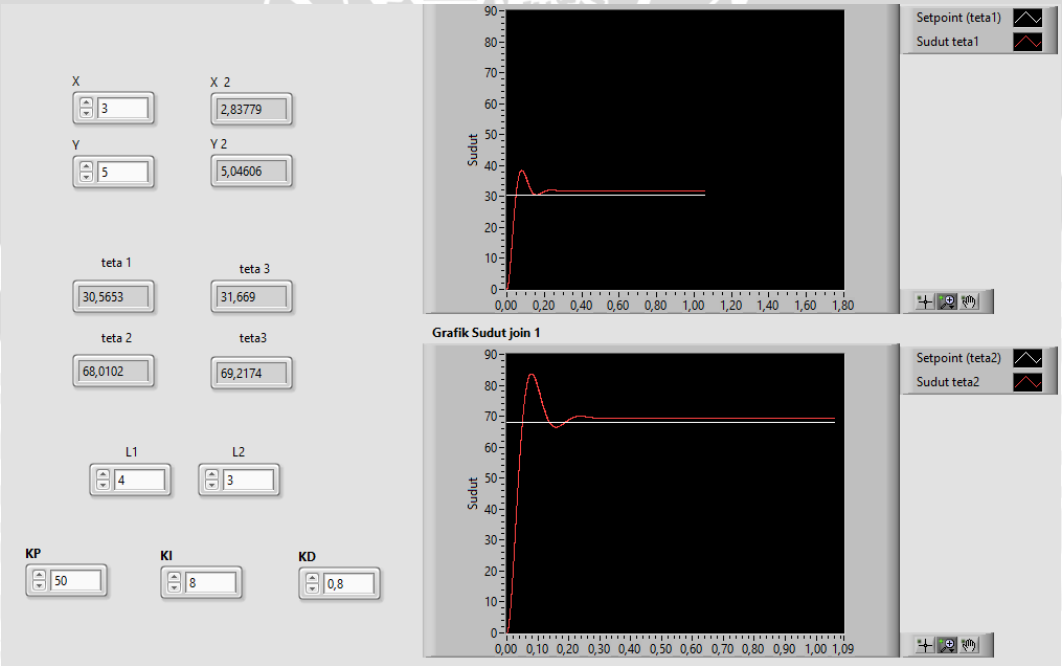
X' = nilai sumbu koordinat x hasil olah sistem

Y' = nilai sumbu koordinat y hasil olah sistem

Ess x = nilai error steady sumbu koordinat x

Ess y = nilai error steady sumbu koordinat y

Tabel 6.2 menunjukkan hasil pegujian pengaruh Ki terhadap sistem, dari hasil yang diperoleh dapat terlihat bahwa Ki memiliki pengaruh untuk menghilangkan *error steady state* semakin besar nilai Ki maka semakin kecil nilai *error steady state* nya, hal ini dibuktikan dengan nilai *error steady state* sudah menunjukkan selisih dibawah 1 sejak awal pengujian. Selain itu Ki juga memiliki pengaruh pada peningkatan *settling time*, seperti yang terlihat ketika nilai Ki 0,01 memiliki nilai *settling time* pada *joint* 0 sebesar 1,02 sedangkan saat nilai Ki 0,05 memiliki nilai *settling time* pada *joint* 0 sebesar 1,16 dan ketika nilai Ki 0,5 memiliki nilai *settling time* pada *joint* 0 sebesar 1,30 begitu juga pada *joint* 1 memiliki keadaan dimana *settling time* meningkat. Maka dapat disimpulkan semakin besar nilai Ki semakin meningkat nilai *settling time*-nya.



Gambar 6.3. Respon sistem dengan Kp = 50 Ki = 8 dan Kd = 0,8

Pada Gambar 6.3 menunjukkan respon sistem saat pengujian pengaruh nilai Kd dengan nilai X = 3, Y = 5, panjang *link* 1 = 4 cm, panjang *link* 2 = 3 cm dan nilai Kp = 50 Ki = 8 dan Kd = 0,8. Menunjukkan respon sistem yang kurang stabil dalam

mendekati setpoint. Hasil pengujian nilai Kd dengan titik koordinat X = 3, Y = 5, panjang *link* 1 = 4 cm dan panjang *link* 2 = 3 cm selengkapnya dapat dilihat pada Tabel 6.3.

Tabel 6.3 tabel hasil pengujian nilai Kd dengan X = 3, Y = 5, L1 = 4cm, L2 = 3cm

Kp	Ki	Kd	% St0	% St1	St 0	St	X'	Y'	Ess x	Ess y
50	8	0,01	4,91	4,4	1,02	0,88	2,88	5,03	0,12	-0,03
50	8	0,05	4,91	4,4	0,88	0,74	2,87	5,04	0,13	-0,04
50	8	0,09	4,91	4,4	0,74	0,6	2,88	5,04	0,12	-0,04
50	8	0,3	4,91	4,4	0,6	0,6	2,84	5,04	0,16	-0,04
50	8	0,5	4,91	4,4	0,44	0,42	2,83	5,04	0,17	-0,04
50	8	0,8	4,91	4,4	0,2	0,2	2,83	5,04	0,17	-0,04
50	8	2	4,91	4,4	0,2	0,2	2,79	5,05	0,21	-0,05
50	8	5	4,91	4,4	0,19	0,2	2,80	5,05	0,2	-0,05
50	8	8	4,91	4,4	0,50	0,5	2,80	5,04	0,2	-0,04
50	8	10	4,91	4,4	0,6	0,7	2,80	5,04	0,2	-0,04

Keterangan tabel:

Kp = nilai kontroler proporsional

Ki = nilai kontroler integral

Kd = nilai kontroler *derivative*

%St0 = prosentase selisih nilai sudut *joint* 0 untuk *settling time*

%St1 = prosentase selisih nilai sudut *joint* 1 untuk *settling time*

St 0 = *settling time joint* 0

St 1 = *settling time joint* 1

X' = nilai sumbu koordinat x hasil olah sistem

Y' = nilai sumbu koordinat y hasil olah sistem

Ess x = nilai error steady sumbu koordinat x

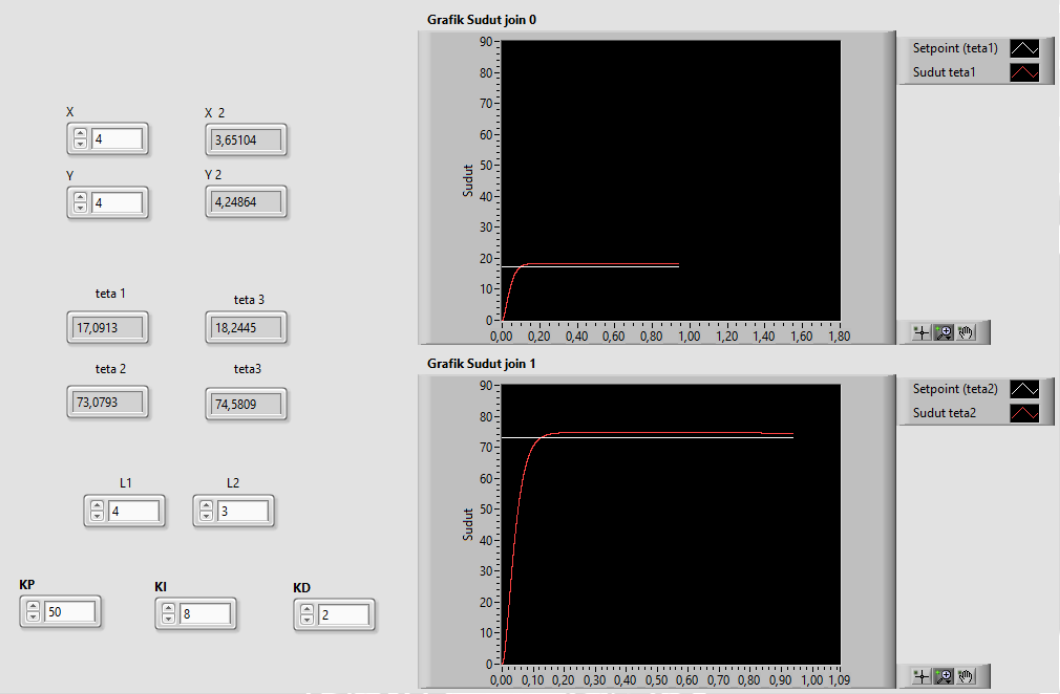
Ess y = nilai error steady sumbu koordinat y

Tabel 6.3 menunjukkan pengaruh nilai Kd terhadap sistem, berdasarkan nilai yang diperoleh dapat diketahui bahwa semakin besar nilai Kd maka semakin kecil *settling time*-nya namun perubahan nilai *error steady state* sangat kecil. Hal ini dapat dilihat pada saat nilai Kd 0,01 memiliki *settling time* pada *joint* 0 sebesar 1,02 dan *joint* 1 sebesar 0,88, sedangkan saat nilai Kd 0,05 memiliki nilai *settling time* pada *joint* 0 sebesar 0,88 dan pada *joint* 1 sebesar 0,74. Berdasarkan data

tersebut maka semakin besar nilai Kd, nilai *settling time* juga akan semakin menurun.

6.2 Pengujian Nilai Koordinat

Pengujian nilai koordinat ini bertujuan untuk melihat respon sistem ketika nilai koordinat berubah-ubah namun nilai PID tetap, serta melihat batasan koordinat yang mampu dijangau oleh panjang lengan tertentu. Berbeda dengan prose pengujian nilai PID yang mengubah nilai PID dengan nilai koordinat yang tetap, untuk pengujian nilai koordinat ini akan dilakukan dengan nilai koordinat berubah dan nilai PID yang tetap.



Gambar 6.4. Respon sistem dengan X = 4 dan Y = 4

Gambar 6.4 menunjukkan respon sistem dengan titik koordinat 4,4 dan nilai PID tetap yang diperoleh dari pengujian PID, menghasilkan respon sistem dengan waktu naik cepat namun memiliki sedikit *overshoot* diawal respon. Hasil dari pengujian nilai koordinat dengan nilai Kp = 50, Ki=8, Kd=2 dan panjang lengan L1 = 4 cm dan L2 = 3 cm dapat dilihat pada Tabel 6.4.

Tabel 6.4. Pengujian Nilai Koordinat

X	Y	X'	Y'	Ess X	Ess Y	%st0	%st1	St0	St1
1	1	1,05	0,9	-0,05	0,1	2,56	0,36	0,02	0,11
4	4	3,76	4,19	0,24	-0,19	2,9	0,1	0,10	0,08
3	5	2,94	5,01	0,06	-0,1	1,31	0,1	0,10	0,08
2	1	1,88	1,16	0,12	-0,16	1,82	0,3	0,9	0,08
5	7	NAN	NAN	NAN	NAN	NAN	NAN	NAN	NAN
6	6	NAN	NAN	NAN	NAN	NAN	NAN	NAN	NAN
8	1	NAN	NAN	NAN	NAN	NAN	NAN	NAN	NAN

Pada Tabel 6.4 menunjukkan bahwa hasil titik koordinat yang diperoleh sistem memiliki selisih yang tidak terlalu jauh dengan titik koordinat yang ditentukan oleh user, hal ini ditunjukkan dengan nilai error steady state pada sumbu x maupun sumbu y selalu bernilai dibawah 1.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan rumusan masalah yang ada, hasil perancangan, implementasi dan pengujian yang dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1) Pada penelitian ini telah dibuat Simulator yang mengimplementasikan PID kontroler untuk robot lengan 2 DOF dengan *input* berupa koordinat dan pengaruh kontrol PID dapat dilihat melalui grafik, proses perancangan dilakukan melalui proses *inverse kinematics*, output dari *inverse kinematics* akan dikontrol dengan PID yang kemudian disimulasikan dengan menggunakan motor DC, kecepatan motor DC inilah yang akan diintegralkan untuk diperoleh sudut hasil olah sistem dan diproses dengan *forward kinematics* untuk didapatkan nilai koordinat kembali.
2) Proses implementasi kontrol PID dilakukan dengan menggunakan fungsi *summation, gain, integrator* dan *derivative* yang telah disediakan labview melalui *control and simulation module*, serta hasilnya ditampilkan dalam grafik dengan memanfaatkan fungsi *array*.
3) Berdasarkan penelitian sistem dapat diketahui bahwa Kontrol Proporsional memiliki peran untuk menurunkan/mempercepat *rise time,* meningkatkan *overshoot,* memperkecil *error,* namun untuk *settling time* hanya berpengaruh kecil*,* kontrol Integral memiliki peran untuk menurunkan/mempercepat *rise time,* menghilangkan *error* namun meningkatkan *overshoot* dan meningkatkan *settling time*, kontrol Derivative memiliki peran untuk menurunkan *overshoot* namun memiliki pengaruh kecil untuk *rise time* dan perubahan *error*. Pada sistem ini diperoleh nilai PID yang sesuai untuk panjang lengan robot L1 = 4 cm dan L2 = 3 cm sebesar P = 50 I = 8 dan D = 0,8.
4) Dari hasil pengujian pada tabel uji koordinat, data yang diperoleh sesuai antara koordinat yang di-*input user* dan koordinat hasil pengolahan sistem. Saat nilai koordinat input user X = 1 dan Y = 1 koordinat hasil olah sistem didapatkan X' = 1,05 dan Y' = 0,9. Sedangkan saat nilai koordinat input user X = 4 dan Y = 4 diperoleh nilai koordinat sistem X' = 3,76 dan Y' = 4,19, saat nilai koordinat X = 3 dan Y = 5 nilai koordinat yang mampu dicapai oleh sistem sebesar X' = 2,94 dan Y' = 5,01.

7.2 Saran

Beberapa saran yang bisa dilakukan untuk mengembangkan sistem ini adalah sebagai berikut:

1. Untuk pengembangan lebih lanjut, diharapkan sistem ini dilengkapi dengan fitur auto tuning, sehingga user bisa mengetahui nilai PID yang tepat tanpa harus mencoba nilai PID satu persatu.
2. Untuk pengembangan lebih lanjut, diharapkan sistem dapat mensimulasikan lengan robot lebih dari 2 DOF.

DAFTAR PUSTAKA

Blok Diagram PID controller. Tersedia di: < http://www.elektroindonesia.com/elektro/tutor12.html>[diakses tanggal 16 Maret 2016].

David. Chart Eric. 2014. Syncronized Robot: A PID control Project With a Lego Mindstorm NXT. Indianapolis: University of Oklahoma.

Definisi Robot. Tersedia di: <http://eprints.undip.ac.id/41644/3/BAB_2.pdf >[diakses tanggal 17 September 2015]

Definisi Simulator. Tersedia di:< http://kbbi.web.id/> [diakses tanggal 17 September 2015].

Ekihci, Mehmet. Kahveci, Hakan. Akpinar, aden sefa. A Labview based submarine Depth Control Simulator with PID and Fuzzy Controller.Trabzone: Karadeniz Technical University.

F.I Ahmed et al. P-I and I-P controllers in closed loop for motor DC Drives.Cairo: Cairo University.

Haugen, finn. 2008. Introduction to Labview 8.5.tersedia di:<http://www.techteach.no/labview/lv85/labview/> [diakses tanggal 1 oktober 2015]

Introduction: PID Controller Design. Tersedia di:< http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID#introduction> [diakses tanggal 2 Oktober 2015]

Introduction to Robotics.New York. Tersedia di: <http://engineering.nyu.edu/mechatronics/smart/pdf/Intro2Robotics.pdf. > [diakses tanggal 17 September 2015]

Micile.Robot Arm Arduino App. Tersedia di:< http://www.instructables.com/id/Robot-Arm-Arduino-App/> [diakses tanggal 2 Oktober 2015]

National Instrument.tersedia di:<http://www.ni.com/example/31436/en/> - [diakses tanggal 30 September 2015]

Noviali, Rahmad.2012. Hmi (Human Machine Interface) Dan Pengontrolan Labview Mengunakan Sistem Pid Pada Prototipe Rotary Stabilizer Tegangan 1 Fasa. Jakarta: Politeknik Negeri Jakarta.

Robot Workx Company. An Introduction to robotic technology. Ohio.tersedia di:< https://www.robots.com/education> [diakses tanggal 21 September 2015].

Servo Control. Tersedia di:<http://www.realtechsupport.org/UB/MRI/MR1_Servos.pdf> [diakses tanggal 2 Oktober 2015]

Setiawan, Eko. 2015. Robotics lecture nodes (indonesia version). Faculty of computer science and engineering. Malang: Brawijaya University

Suef, M. P, Yudha. A, Maria.Sistem Kendali (*Control System*). Tersedia di:< http://share.its.ac.id/pluginfile.php/30119/mod_resource/content/2/MINGGU07_OTOMASI_INDUSTRI.pdf> [diakses tanggal 3 Maret 2016]

Wakamenta, YN.2011. Bab 2. dosen.narotama.ac.id/wp-content/uploads/2012/01/Sejarah-Robot.pdf [diakses tanggal 21 September 2015]