# PENENTUAN SUDUT LENGAN ROBOT HUMANOID BERDASARKAN KOORDINAT YANG DIKIRIM DARI PC MENGGUNAKAN *USER INTERFACE* YANG DIBUAT DARI Qt

**SKRIPSI**

Diajukan Untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik


Disusun Oleh:

**ADIYATMA GHAZIAN PRATAMA**

**NIM. 105060300111065 - 63**

**KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN**

**UNIVERSITAS BRAWIJAYA**

**FAKULTAS TEKNIK**

**JURUSAN TEKNIK ELEKTRO**

**MALANG**

**2014**

# LEMBAR PERSETUJUAN

**PENENTUAN SUDUT LENGAN ROBOT HUMANOID BERDASARKAN KOORDINAT YANG DIKIRIM DARI PC MENGGUNAKAN *USER INTERFACE* YANG DIBUAT DARI Qt**

## SKRIPSI

**JURUSAN TEKNIK ELEKTRO**

Diajukan untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik


Disusun Oleh:
**ADIYATMA GHAZIAN PRATAMA**
**NIM. 105060300111065 - 63**


**Telah diperiksa dan disetujui oleh :**

| Pembimbing 1 | Pembimbing 2 |
|---|---|
| **Ir. Nurussa'adah, MT.** | **Mochammad Rif'an, ST., MT.** |
| **NIP. 19680706 199203 2 001** | **NIP. 19710301 200012 1 001** |

# LEMBAR PENGESAHAN

**PENENTUAN SUDUT LENGAN ROBOT HUMANOID BERDASARKAN KOORDINAT YANG DIKIRIM DARI PC MENGGUNAKAN *USER INTERFACE* YANG DIBUAT DARI Qt**

**SKRIPSI**
**JURUSAN TEKNIK ELEKTRO**

Disusun Oleh:

**ADIYATMA GHAZIAN PRATAMA**

**NIM. 105060300111065 - 63**

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 3 Desember 2014

**MAJELIS PENGUJI**

| | |
|---|---|
| **Ir. Ponco Siwindarto, M.Eng.Sc** | **Adharul Muttaqin, ST., MT.** |
| **NIP. 19590304 198903 1 001** | **NIP. 19760121 200501 1 001** |

**Ir. M. Julius, St., MS.**
**NIP. 19540720 198203 1 002**

Mengetahui,
Ketua Jurusan Teknik Elektro

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

# PENGANTAR

*Bismillahirrohmanirrohim.*

Alhamdulillah, puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penentuan Sudut Lengan Robot Humanoid Berdasarkan Koordinat Yang Dikirim Dari PC Menggunakan User Interface Yang Dibuat Dari Qt". Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.

Penulis menyadari bahwa penyusunan skripsi ini tidak lepas dari bantuan, bimbingan serta dorongan dari berbagai pihak. Pada kesempatan ini penulis menyampaikan rasa terima kasih kepada:

- Allah SWT yang selalu tahu kapan waktu yang terbaik untuk hambaNya dan Rasulullah Muhammad SAW, semoga shalawat serta salam selalu tercurah kepada beliau.
- Kedua orang tua tercinta, Bapak Edi Sudiarto dan Ibu Agus Purwani Ningtyas yang senantiasa mendoakan, memberikan nasihat, dorongan, perhatian, dan kesabaran selama ini. Serta adik saya Fahrizal Ghazian Putra yang senantiasa memberikan dukungan.
- Bapak M. Aziz Muslim, ST., MT, Ph.D sebagai Ketua Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Hadi Suyono, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
- Ibu Ir. Nurussa'adah, MT. selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya sekaligus sebagai dosen pembimbing yang selalu memberi semangat Elektronika 2010 untuk cepat menyelesaikan skripsinya dan banyak memberi masukan, nasehat, pengarahan, motivasi dan saran hingga selesainya skripsi ini.
- Bapak Mochammad Rif'an, ST. MT. sebagai Kaprodi Teknik Elektro Universitas Brawijaya sekaligus sebagai dosen pembimbing yang telah

banyak memberi masukan, saran, nasehat, dan motivasi hingga selesainya skripsi ini.

- Seluruh dosen Teknik Elektro Universitas Brawijaya teristimewa Bapak Ir. Hery Purnomo, selaku dosen pembimbing akademik dan seluruh dosen konsentrasi Elektronika atas segala bimbingan yang diberikan selama saya berproses di Universitas Brawijaya.
- Staf Pengajaran, staf rekording, dan staf Ruang Baca Jurusan Teknik Elektro terutama Pak Mulyadi, Mbak Eka, Mbak Kokom, Mbak Heni, Mas Jun, Mas Nugroho dan Mbak Frida yang telah membantu segala urusan saya selama ini.
- Sahabat yang seperti saudara saya di Himpunan Mahasiswa Hoki, Dheo, Iqbal, Arez, Maman, Afnan, Laksana, Imam, Azwar, Nizar, Radek, Feri, Riza, Rize, Fajar, Aziz, Abu Tempe, Kefin, Comel, Samto atas kerelaannya berbagi semangat, cerita, cinta, dan waktu yang menyenangkan selama menjadi mahasiswa.
- Teman-teman Asisten Laboratorium Elektronika, Yoga, Vicky, Erwan, Zainma, Nurdin, Bayu, Ainun, Rico, Jaka, Bustanul, Firman, Adis, Dilla dan asisten-asisten lainnya, terimakasih atas kerjasama, pengalaman, pengorbanan, semangat dan bantuan selama ini.
- Teman-teman Divisi Mikrokontroller Hakiki, Firman, Rita, Roqib, Ilham, Bima, Graha, Tito, Wildan atas kerjasama, ilmu, bantuan dan semangat selama ini.
- Sahabat yang memberikan sumbangsih masing-masing dalam penyelesaian skripsi ini, Basori, Tanshuda, Veri, Azri, Dina, Anas, Irham, Gusonela, Mas Fikri.
- Teman-teman Tim robot, aslab Laboratorium Sistem Kontrol, aslab Laboratorium Sistem Digital dan RisTIEyang markasnya pernah bahkan ada yang seringkali saya tempati mengerjakan skripsi.
- Rekan-rekan seperjuangan dalam pengerjaan skripsi Faisol, Yuda, Batman, Dayat dkk atas segala bantuan dan dukungan selama ini.
- Teman-teman Elektronika 2010 atas kerjasama, bantuan, semangat selama ini.

- Keluarga besar angkatan 2010 “MAGNET’10” yang memberikan doa, semangat, serta dukungan.
- Seluruh teman-teman serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan satu-persatu, terimakasih banyak atas bantuan dan dukungannya

Dalam penyusunan skripsi ini, penulis menyadari bahwa skripsi ini belumlah sempurna, karena keterbatasaan ilmu dan kendala-kendala lain yang terjadi selama pengerjaan skripsi. Oleh karena itu, penulis mengharap kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang. Penulis berharap, semoga tulisan ini dapat bermanfaat bagi kita semua.

Malang, November 2014

Penulis

# ABSTRAK

**Adiyatma Ghazian Pratama**, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, November 2014, Penentuan Sudut Lengan Robot Humanoid Berdasarkan Koordinat Yang Dikirim Dari PC Menggunakan *User Interface* Yang Dibuat Dari Qt, Dosen Pembimbing: Ir. Nurussa'adah, MT., Mochammad Rif'an, ST., MT.

Pada saat pencarian gerakan tari, biasanya tim KRSI Teknik Elektro Universitas Brawijaya melihat contoh gerakan, memperkirakan sudut di setiap sendi, kemudian diproses oleh Mikrokontroller untuk menggerakkan motor DC *servo* di setiap sendinya. Kekurangan metode ini adalah proses *trial* yang tidak sebentar untuk mencari gerakan tari sesuai yang diinginkan.

Pada skripsi ini akan dirancang aplikasi untuk menggerakkan motor DC servo pada lengan robot sesuai koordinat yang telah ditentukan dengan memasukkan nilai koordinatnya dari PC. Metode yang digunakan unutuk mendapatkan perubahan sudut tiap sendi pada lengan robot menggunakan Persamaan Invers Kinematik. Pada referensi yang didapat mengenai Invers Kinematik posisi basis berada di tanah sedangkan lengan robot berada di atas. Oleh sebab itu ada beberapa variabel yang didapat pada referensi tersebut mengalami modifikasi pada perancangan ini. Masukan koordinat adalah dari PC yang menggunakan *User Interface* yang dibuat dari Qt.

Hasil yang didapatkan pada pengujian keseluruhan sistem adalah kesalahan yang terjadi adalah karena nilai karakterisasi masing-masing servo yang masih memiliki error sehingga berpengaruh terhadap keseluruhan gerakan menuju koordinat tujuan.

**Kata Kunci**: Lengan Kanan Robot, Invers Kinematik, *Qt*.

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Kontes Robot Seni tari Indonesia (KRSI) merupakan suatu ajang kompetisi perancangan dan pembuatan robot yang disertai dengan unsur-unsur seni dan budaya bangsa yang telah terkenal di bumi pertiwi. Pada saat pencarian gerakan tari, biasanya tim KRSI Teknik Elektro Universitas Brawijaya melihat contoh gerakan, memperkirakan sudut di setiap sendi, kemudian diproses oleh Mikrokontroller untuk menggerakkan motor DC *servo* di setiap sendinya. Kekurangan metode ini adalah proses *trial* yang tidak sebentar untuk mencari gerakan tari sesuai yang diinginkan.

Permasalahan yang tidak efisien tersebut pernah diangkat dengan memasang *exoskeleton* yang di setiap sendinya dilengkapi sensor posisi sudut kemudian data diproses untuk menggerakkan sendi pada robot *humanoid* hingga gerakan robot sesuai yang diinginkan, kemudian data sudut pada setiap persendian dapat ditampilkan ke *Personal Computer* untuk dimasukkan ke dalam program pergerakan otomatis robot dimana hasil akhinya adalah Sendi *Exoskeleton* berhasil menggerakkan sendi robot humanoid dengan kesalahan rata-rata 0,59°, kesalahan terbesar 1° dan kesalahan terkecil 0°. Sementara sudut yang ditampilkan ke PC memiliki kesalahan rata-rata 1,13°, kesalahan terbesar 4° dan kesalahan terkecil 0°(Tanshuda, 2014:57).

Pada skripsi ini akan dirancang aplikasi untuk menggerakkan motor DC servo pada lengan robot sesuai koordinat yang telah ditentukan dengan memasukkan nilai koordinatnya dari PC. Metode yang digunakan unutuk mendapatkan perubahan sudut tiap sendi pada lengan robot menggunakan Persamaan Invers Kinematik. Pada referensi yang didapat mengenai Invers Kinematik posisi basis berada di tanah sedangkan lengan robot berada di atas. Oleh sebab itu ada beberapa variabel yang didapat pada referensi tersebut

mengalami modifikasi pada perancangan ini. Masukan koordinat adalah dari PC yang menggunakan *User Interface* yang dibuat dari Qt.

## 1.2 Rumusan Masalah

Berdasarkan masalah yang telah dijelaskan pada latar belakang, dapat dibuat rumusan sebagai berikut:

a. Bagaimana merancang aplikasi untuk menggerakkan motor servo pada lengan robot sesuai koordinat yang telah ditentukan dari antarmuka pada PC yang dibuat dengan *software* Qt.

b. Bagaimana perhitungan invers kinematik untuk mendapatkan persamaan gerakan servo.

## 1.3 Batasan Masalah

Dalam menyusun penelitian, batasan masalah yang digunakan dalam penelitian :

- Tampilan aplikasi pada PC menggunakan software Qt.
- Mikrokontroller yang digunakan adalah ATMega32.
- Servo yang digunakan sejumlah 5 atau 5 DOF (*Degree of Freedom*).
- Antarmuka dengan PC menggunakan komunikasi serial.
- Robot yang digunakan berbentuk humanoid bagian lengan kanan.
- Servo yang digunakan Hitech 5685 dan Corona DS238HV.
- Parameter yang diperhitungkan hanya sekedar aspek praktis, tanpa membahas mekanika secara mendalam.
- PC digunakan untuk mengirimkan data koordinat ke mikrokontroller.

## 1.4 Tujuan

Berdasarkan rumusan masalah di atas, tujuan dari penelitian ini adalah sebagai berikut:

a. Membuat *prototype* lengan kanan robot yang bergerak sesuai koordinat yang diinginkan dari PC dengan *software* Qt.

b. Mengetahui kesalahan yang terjadi pada *prototype* yang dibuat.

## 1.5 Sistematika Pembahasan

Sistematika penulisan yang digunakan dalam penyusunan laporan penelitian ini adalah sebagai berikut.

**BAB I Pendahuluan**

Berisi latar belakang, rumusan masalah, batasan masalah, tujuan, dan sistematika pembahasan.

**BAB II Tinjauan Pustaka**

Berisi tinjauan pustaka atau dasar teori yang digunakan untuk dasar penelitian yang dilakukan dan untuk mendukung pembuatan alat

**BAB III Metode Penelitian**

Berisi metode penelitian yang akan dilakukan, meliputi metode yang digunakan, obyek penelitian dan data yang diperlukan serta langkah penelitian.

**BAB IV Perancangan**

Berisi pembahasan, analisis terhadap masalah yang diajukan dalam skripsi dengan memperhatikan hasil pengujian dan data yang diperoleh.

**BAB V Pengujian dan Analisis**

Berisi kesimpulan dari tujuan skripsi yang akan dibuat serta saran dari penulis.

**BAB VI Kesimpulan dan Saran**

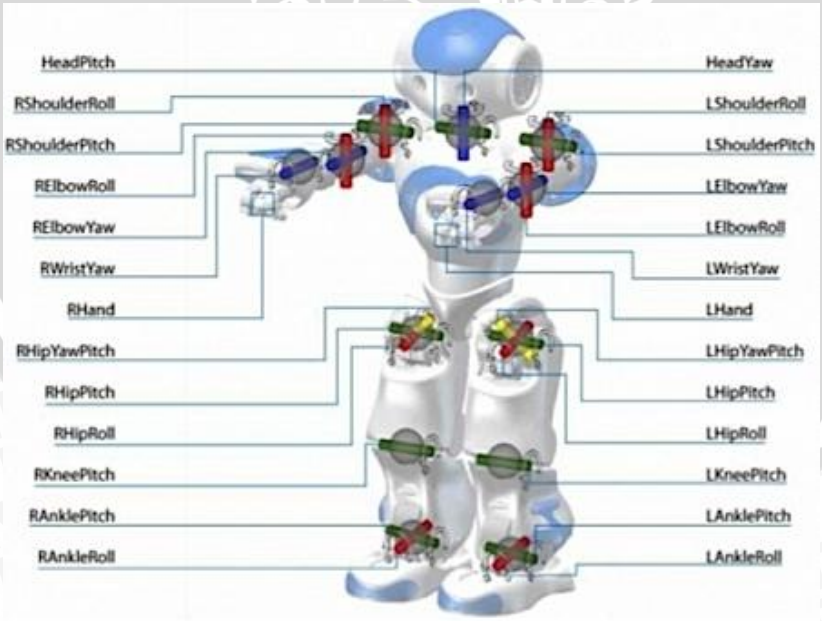
Memuat intisari hasil pengujian dan menjawab rumusan masalah serta memberikan rekomendasi untuk perbaikan kulaitas penelitian di masa yang akan datang.

# BAB II
# TINJAUAN PUSTAKA

## 2.1 Robot Humanoid

Robot *humanoid* adalah robot yang bentuknya berdasarkan struktur tubuh manusia, mampu melakukan interaksi dengan peralatan maupun lingkungan yang dibuat untuk manusia. Secara umum robot *humanoid* memiliki tubuh dengan kepala, sepasang lengan dan sepasang kaki, meskipun ada pula beberapa bentuk robot *humanoid* yang hanya berupa sebagian dari tubuh manusia, misalnya dari pinggang ke atas. Beberapa robot *humanoid* juga memiliki wajah, lengkap dengan mata dan mulut.

Robot *humanoid* digunakan sebagai sarana penelitian dalam berbagai bidang ilmu. Untuk membuat robot *humanoid*, peneliti perlu mempelajari struktur dan perilaku tubuh manusia (biomekanik). Simulasi pergerakan tubuh manusia mengarahkan pada pemahaman yang lebih baik mengenai hal tersebut.




Gambar 2.1 Derajat Kebebasan Aldebaran Nao

Sumber: Shahbazi (2012)

Secara keseluruhan, sebuah robot terdiri dari sistem elektrik, mekanik, serta program. Ditinjau dari segi kontrol, sebuah robot *humanoid* biasanya merupakan sistem *loop* tertutup, yang terdiri dari sensor, pengontrol, dan aktuator. Sensor berfungsi seperti sistem pengindraan, kontroller sebagai otak, serta aktuator sebagai sistem otot dan persendian. Robot humanoid yang digunakan memiliki beberapa sendi yang disebut sebagai *Degree of Freedom* (DOF) yang berupa aktuator, seperti motor, *pneumatic, hydraulic,* dan lain-lain. Gambar 2.1 menunjukkan contoh permodelan persendian atau *Degree of Freedom* pada robot Aldebaran Nao.

## 2.2 Invers Kinematik

Perancangan kinematika lengan robot harus didasarkan pada pendeskripsian *frame* lengan robot. Perancangan kinematika lengan robot sebagai dasar dari perancangan perangan lunak.

Bagian-bagian pada lengan robot diantaranya batang-batang logam yang berfungsi sebagai link. Pergerakan lengan diatur oleh Mikrokontroller dan PC sebagai *interface* sekaligus masukan data.

Sistem koordinat lengan robot mengacu pada standar yang digunakan Mark W. Spong, 2004, setiap *link* diberi nomor dimulai dari bagian *basement* yang tidak bergerak. *Basement* disebut link 0. Batang yang bergerak dan berhubungan dengan *link* 1 disebut *link* 2 dan seterusnya. Sehingga secara keseluruhan lengan robot terbentuk dari *link* 0, *link* 1, *link* 2, dan *link* 3. Antara *link* yang satu dengan *link* tetangganya dihubungkan dengan *joint*. *Joint i+1* adalah *joint* yang menghubungkan *link i* dengan *link i+1*. Sehingga *joint 1* menghubungkan *link 0* dengan *link 1*. Sedangkan *link 1* dan *link 2* dihubungkan dengan *joint 2* dan seterusnya. *Frame* merupakan sistem koordinat yang menggambarkan posisi sebuah *link* relatif terhadap *link* lainnya. Sistem koordinat ini melekat pada *link*. Penomoran *frame* sesuai dengan penomoran *link* yang dilekatinya. *Frame* ditempatkan pada *link* sehingga sumbu Z dari *frame {i}* yaitu $Z_i$, sejajar dengan sumbu putar (putar) dari *joint i.*

Saat didapat sudut joint $\theta_1$, $\theta_2$ maka dapat dijabarkan koordinat *end-affector* x dan y. Untuk memerintah robot untuk berpindah ke lokasi tujuan maka dibutuhkan invers.

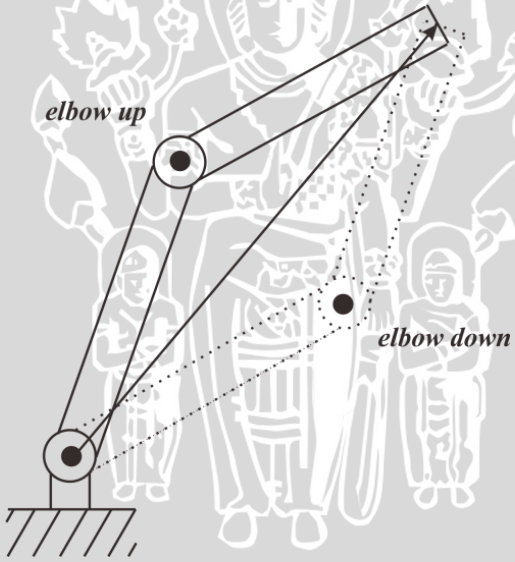
Jika diberikan *(x,y)* koordinat dalam daerah jangkauan kemungkinan terdapat 2 solusi seperti dalam Gambar 2.2 yang disebut konfigurasi *elbow up* dan *elbow down* atau mungkin hanya terdapat satu solusi jika lengan diperpanjang hingga mencapai jangkauan kooordinat yang dituju.

Sudut $\theta_2$ dapat dicari dengan menggunakan Hukum Kosinus yang terlihat dalam Gambar 2.3.

$$\cos\theta_2 = \frac{x^2+y^2-\alpha_1^2-\alpha_2^2}{2\alpha_1\alpha_2} := D. \quad (2\text{-}1)$$

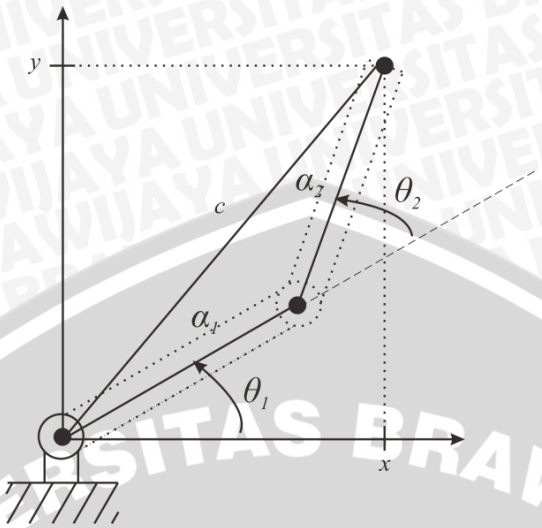
Sehingga $\theta_2$ dapat dijabarkan sebagai

$$\theta_2 = \cos^{-1} D. \quad (2\text{-}2)$$




Gambar 2.2 Konfigurasi *elbow up* dan *elbow down*

Sumber: Mark W. Spong, 2004




Gambar 2.3 Penyelesaian untuk sudut-sudut *joint* pada lengan planar dua *link*

Sumber: Mark W. Spong, 2004

Cara terbaik untuk mendapatkan $\theta_2$ dengan memperhatikan bahwa cos($\theta_2$) pada Persamaan 2-2 sehingga sin($\theta_2$) adalah

$$\sin\theta_2 = \pm\sqrt{1-D^2} \tag{2-3}$$

Sehingga $\theta_3$ dapat ditulis

$$\theta_2 = \tan^{-1}\frac{\pm\sqrt{1-D^2}}{D}. \tag{2-4}$$

Keuntungan dari pendekatan ini adalah tanda positif dan negatif pada Persamaan (4.5) yang mengakibatkan solusi untuk konfigurasi *elbow up* dan *elbow down* dapat terselesaikan. Sehingga $\theta_1$ dapat ditulis

$$\theta_1 = \tan^{-1}\frac{y}{x} - \tan^{-1}\left(\frac{\alpha_2\sin\theta_2}{\alpha_1+\alpha_2\cos\theta_2}\right) \tag{2-6}$$

Oleh karena $\theta_1$ berhubungan dengan $\theta_2$ secara fisik maka untuk setiap nilai $\theta_1$ yang berbeda adalah dari nilai $\theta_2$ yang dipilih.

## 2.3 Motor DC Servo

Motor servo adalah motor yang mampu bekerja dua arah (searah putaran jarum jam dan berlawanan arah putaran jarum jam) dimana arah dan sudut pergerakan rotornya dapat dikendalikan dengan memberikan pengaturan *duty*

*cycle* sinyal PWM pada bagian pin kontrolnya. Motor servo ditunjukkan dalam Gambar 2.4.


Gambar 2.4 Motor Servo

Sumber: *redwingrc.com*

Motor servo merupakan sebuah motor DC yang memiliki rangkaian kontrol elektronik dan internal *gear* untuk mengendalikan pergerakan dan sudut angularnya.

Motor servo adalah motor servo yang berputar lambat, dimana biasanya ditunjukkan oleh *rate* putarannya yang lambat, namun demikian memiliki torsi yang kuat karena internal *gear*-nya. Lebih dalam dapat digambarkan bahwa sebuah motor servo memiliki:

1) 3 jalur kabel: *power*, *ground*, dan *control.*
2) Sinyal kontrol untuk mengendalikan posisi
3) Operasional dari servo motor dikendalikan oleh sebuah pulsa sebesar ± 20 ms, dimana lebar pulsa antara 0,5 ms dan 2 ms menyatakan akhir dari *range* sudut maksimum.
4) Konstruksi didalamnya meliputi internal *gear*, potensiometer, dan *feedback control.*

Jenis-jenis dan kegunaan motor servo antara lain:

1) Motor servo standar $180^o$

   Motor servo jenis ini hanya mampu bergerak dua arah (searah putaran jarum jam dan berlawanan arah putaran jarum jam) dengan defleksi

masing-masing sudut mencapai $90^o$ sehingga total defleksi sudut dari kanan-tengah-kiri adalah $180^o$.

2) Motor servo *continous*

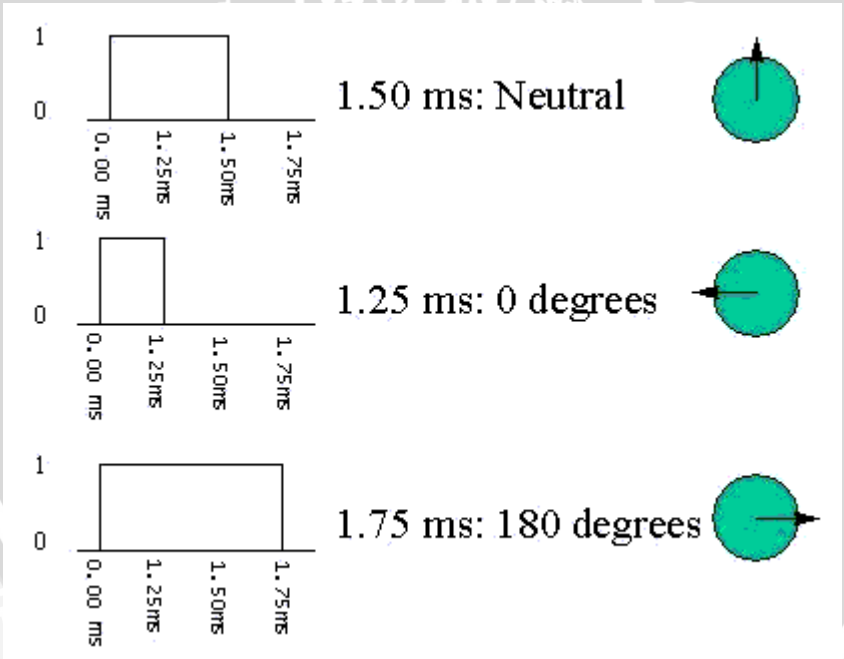
Motor servo jenis ini mampu bergerak dua arah (searah putaran jarum jam dan berlawanan arah putaran jarum jam) tanpa batasan defleksi sudut putar (dapat berputar secara kontinyu).

Kebanyakan motor servo digunakan sebagai:

1) *Manipulator*
2) *Moving camera*
3) *Robot arms*

Mode pemberian pulsa pada motor servo

Sudut dari motor servo tergantung dari besarnya pulsa yang akan diterima oleh motor servo.



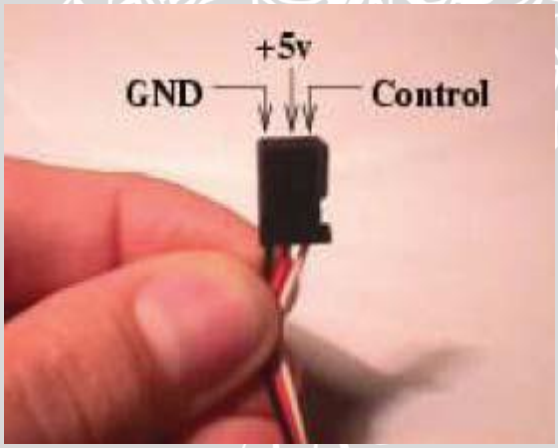

Gambar 2.5 Contoh posisi dan waktu pemberian pulsa

Sumber: Sigit Riyanto, 2007

Contoh dimana bila diberikan pulsa dengan besar 1.5 ms mencapai gerakan $90^o$, maka bila diberikan data kurang dari 1.5 ms maka posisi mendekati $0^o$ dan bila diberikan data lebih dari 1.5 ms maka posisi mendekati $180^o$. Contoh dan waktu pemberian pulsa ditunjukkan dalam Gambar 2.5.

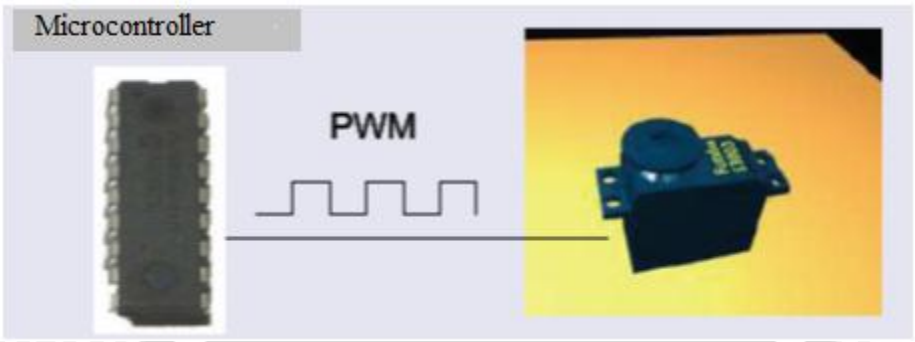
1) Motor servo akan bekerja secara baik jika pada bagian pin kontrolnya diberikan sinyal PWM (Pulse Width Modulation) dengan frekuensi 50 Hz.
2) Pada saat sinyal dengan frekuensi 50 Hz tersebut dicapai pada kondisi $T_{on}$ *duty cycle* 1.5 ms, maka rotor dari motror akan berhenti tepat di tengah-tengah (sudut $0^o$/netral).
3) Pada saat $T_{on}$ *duty cycle* dari sinyal yang diberikan kurang dari 1.5 ms, maka rotor akan berputar ke arah kiri dengan membentuk sudut yang besarnya linier terhadap besarnya $T_{on}$ *duty cycle,* dan akan bertahan di posisi tersebut.
4) Dan sebaliknya, jika $T_{on}$ *duty cycle* dari sinyal yang diberikan lebih dari 1.5 ms, maka rotor akan berputar ke arah kanan dengan membentuk sudut yang linier pula terhadap besarnya $T_{on}$ *duty cycle,* dan bertahan di posisi tersebut.

Kebanyakan motor servo mempunyai bentuk yang relatif sama, perbedaannya hanya berdasarkan dimensi. Setiap motor servo memiliki karakteristik yang berbeda-beda.




Gambar 2.6 Pin-pin dan pengkabelan motor servo

Sumber: Sigit Riyanto, 2007

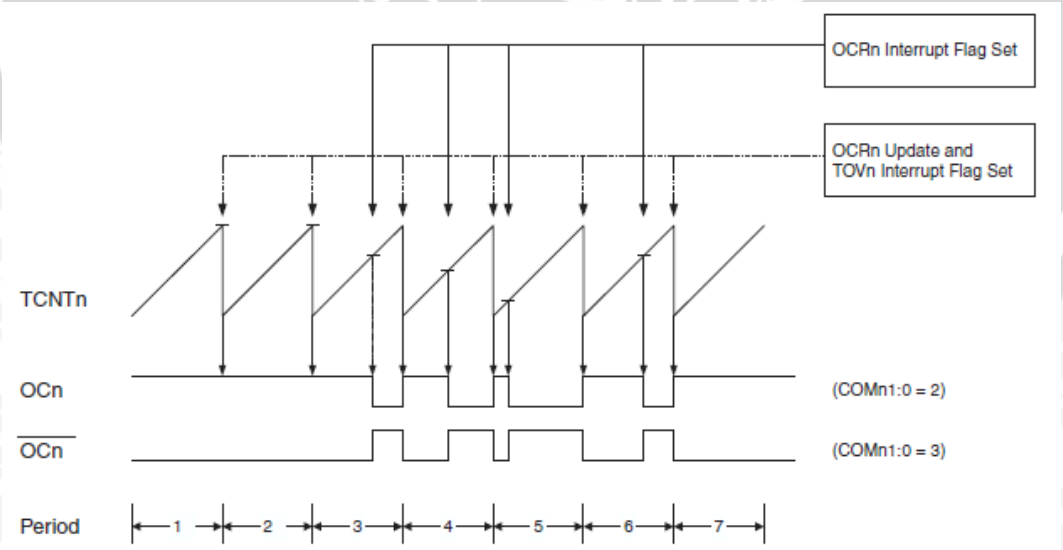



Gambar 2.7 Ilustrasi pengendalian motor servo

Sumber: Sigit Riyanto, 2007

## 2.4 *Pulse Width Modulation* (PWM)

Sinyal PWM dapat dibangkitkan dengan menggunakan *peripheral Timer/Counter* yang terdapat pada Mikrokontroller. Pada Mikrokontroller ATMega32 terdapat dua buah *Timer/Counter* 8-bit dan satu buah *Timer/Counter* 16-bit. *Timer/Counter* 1 merupakan *Timer/Counter* 16 bit. Salah satu mode yang bisa digunakan untuk membangkitkan sinyal PWM adalah mode *Fast PWM*, seperti yang terdapat dalam Gambar 2.8.




Gambar 2.8 Mode *Fast PWM*

Sumber: Atmel (2010:108)

Mode *Fast Pulse Width Modulation* atau *Fast* PWM menyediakan pilihan pembangkitan gelombang PWM frekuensi tinggi. *Fast* PWM berbeda dengan opsi PWM lainnya karena operasi *single-slope*-nya. *Counter* mencacah dari BOTTOM hingga TOP, kemudian dimulai lagi dari BOTTOM. Dalam mode *non-inverting Compare Output, Output Compare* (OC1x) dinolkan pada *compare match* antara TCNT1 dan OCR1x, dan masuk kondisi set pada compare match dan dinolkan pada BOTTOM. Dalam mode *inverting Compare Output*, keluaran berada pada kondisi *set* pada *compare match* dan dinolkan pada BOTTOM. Berdasarkan operasi *single-slope*, frekuensi yang beroperasi pada mode *Fast* PWM bisa dua kali lebih tinggi dari mode *Phase Correct* PWM yang menggunakan operasi *dual-slope*. Setiap pengiriman data pada UART menggunakan bit tanda *start* bit dan *slop* bit.

## 2.5. Mikrokontroller ATMega32

Salah satu mikrokontroller yang banyak digunakan saat ini yaitu mikrokontroller AVR. AVR adalah mikrokontroller RISC (*reduce instruction set compute*) 8 bit berdasarkan arsitektur Harvard, yang dibuat oleh atmel pada tahun 1996. AVR memiliki keunggulan dibandingkan mikrokontroller lain. Keunggulan mikrokontroller AVR yaitu AVR memiliki kecepatan eksekusi program yang lebih cepat karena sebagian besar instruksi dieksekusi dalam siklus 1 clock, lebih cepat dibandingkan dengan mikrokontroller MCS51 yang memiliki arsitektur CISC (*complex instruction set compute*), mikrokontroller MCS51 membutuhkan 12 siklus clock untuk mengeksekusi 1 instruksi.

Mikrokontroller AVR juga memiliki fitur yang lengkap (ADC internal, EEPROM Internal, *Timer/Counter, watchdog Timer, PWM, Port I/O, komunikasi serial, Komparator, I2C, dll*), sehingga dengan fasilitas yang lengkap ini, *Programmer* dan desainer dapat menggunakannya untuk berbagai aplikasi sistem elektronika seperti robot, otomasi industri, peralatan telekomunikasi, dan berbagai keperluan lain. Secara umum mikrokontroller AVR dapat dikelompokkan menjadi 3 kelompok, yaitu keluarga AT90xx, ATMega dan Attiny. Salah satu IC mikrokontroller yang sering digunakan adalah ATMega32.

ATMega32 mempunyai *throughput* mendekati 1 MIPS per MHz membuat disainer sistem untuk mengoptimasi konsumsi daya *versus* kecepatan proses (data sheet ATMEGA 32, 2007:3). Beberapa keistimewaan dari AVR ATMega32 antara lain:

1. *Microcontroller* AVR 8 bit yang memiliki kemampuan tinggi, dengan daya rendah.
2. Arsitektur RISC dengan *throughput* mencapai 16 MIPS pada frekuensi 16 MHz.
3. Memiliki kapasitas *flash* memori 16 Kbyte, EEPROM 512 *Byte* dan SRAM 1 *Kbyte*.
4. Saluran I/O sebanyak 32 buah, yaiut port A, port B, port C dan port D.
5. CPU yang terdiri dari 32 buah register
6. Unit interupsi internal dan eksternal.
7. *Port* USART untuk komunikasi serial
8. Fitur *pheripheral*
   a. Tiga buah *Timer/Counter* dengan kemampuan pembandingan.
      a) 2 (dua) buah *Timer/Counter* 8 bit dengan *prescaler* terpisah dan *Mode Compare*
      b) 1 (satu) buah *Timer/Counter 16* bit dengan *prescaller* terpisah, *Mode Compare*, dan *Mode Capture*
   b. *Real Time Counter* dengan *Oscilator* tersendiri
   c. 4 *channel* PWM
   d. 8 *channel*, 10-*bit* ADC
      a) 8 *Single-ended Channel*
      b) 7 *Differential Channel* hanya pada kemasan TQFP
      c) 2 *Differential Channel* dengan *programmable Gain 1x, 10x,* atau 200x
   e. *Byte-Oriented Two-wire Serial Interface*
   f. *Progammable* serial USART
   g. Antarmuka SPI
   h. *Watchdog Timer* dengan *oscillator internal*

i. *On-chip Analog Comparator*

(data sheet ATMEGA 32, 2007:1)

ATMega32 memerlukan sistem minimum agar dapat digunakan sebagai mana mestinya. Ada 3 bagian penting dalam sistem minimum ATmega32 ini, yaitu:

1. Catu Daya

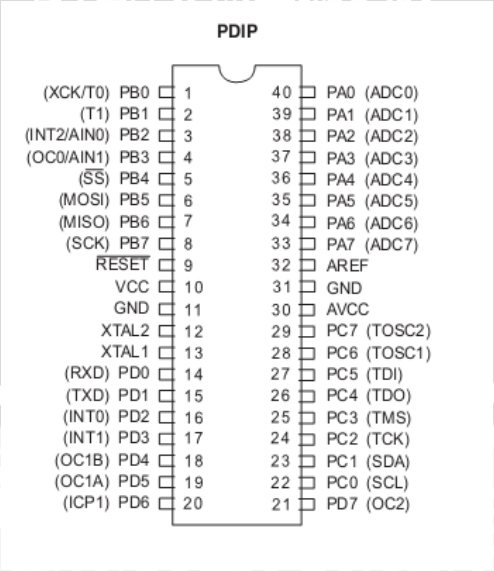
   Catu daya adalah sumber *power*/tegangan yang akan menghidupkan sistem minimum ini. Catu daya yang dibutuhkan oleh sistem minimum ini sebesar 4,5 - 5V.

2. *Clock/Crystal*

   *Clock/Crystal* merupakan hal yang sangat penting juga dalam rangkaian sistem minimum karena bagian ini yang berfungsi memberikan *clock* untuk berjalannya transfer data.

3. *Reset*

   Rangkaian *reset* berfungsi sebagai *interrupt* untuk *set* ke program awal. Ketika pin ini diaktifkan maka program akan berjalan lagi dimulai dari awal.




Gambar 2.9 Konfigurasi pin ATMega32

Sumber: Data sheet ATMega32

4. *Port* A(PA0…PA7) merupakan pin *input/output* dua arah dan pin masukan ADC.
5. *Port* B (PB0…PB7) merupakan pin *input/output* dua arah dan pin fungsi khusus, seperti dapat dilihat pada Tabel 2.1.

Tabel 2.1 Fungsi khusus Port

| *Pin* | Fungsi Khusus |
|---|---|
| **PB7** | SCK (SPI *Bus Serial Clock)* |
| **PB6** | MISO (SPI *Bus Master Input/Slave Output)* |
| **PB5** | MOSI (SPI *Bus Master Output/Slave Input)* |
| **PB4** | $\overline{SS}$ (SPI *Slave Select Input)* |
| **PB3** | AIN1 (*Analaog Comparator Negative Input)*<br>OC0 (*Timer/Counter0 Output Compare Match Output)* |
| **PB2** | AIN0 (*Analog Comparator Positive Input)*<br>INT2 (*External Interrupt 2 Input)* |
| **PB1** | T1 (*Timer/Counter1 External Counter Input)* |
| **PB0** | T0 T1 (*Timer/Counter0External Counter Input)*<br>XCX (*USART External Clock Input/Output)* |

6. AVCC merupakan pin masukan tegangan ADC.
7. AREF merupakan pin masukan tegangan referensi ADC.

(data sheet ATMega32, 2007:5)

## 2.6 Qt

Qt Framework sudah sejak lama digunakan untuk mengembangkan aplikasi lintas platform. Qt sendiri dibuat pada tahun 1996 oleh perusahaan dari

swedia yang bernama Trolltech. Karena sifarnya yang lintas platform sehingaa dapat membuat aplikasi yang berjalan diatas platform Windows, Linux, dan Mac. Dengan Qt kode yang sama dapat dijalankan pada target platform yang berbeda.

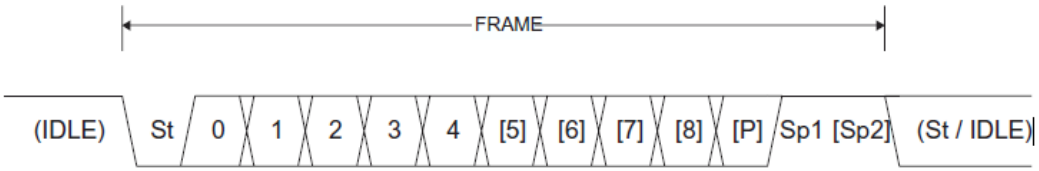
Gambar 2.10 Logo Qt *Framework*

Qt Framework sudah didesain sedemikian rupa sehingga mudah digunakan oleh developer tanpa harus mengorbankan fleksibilitas dan efisiensi. Qt mendukung pengembangan dengan dua bahasa utama yaitu Object Oriented C++ dan Java. Pada perancangan ini antarmuka dalam PC dibuat dari aplikasi Qt ini .

## 2.7 Komunikasi Serial

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

Setiap pengiriman data pada UART menggunakan bit tanda *start* bit dan *Stop* bit. Jalur data yang digunakan hanya satu untuk setiap pengiriman data. Data-data serial dikirim melalui jalur data satu persatu setiap satuan waktu. Format pengiriman data serial secara asinkron ditunjukkan dalam Gambar 2.11.




Gambar 2.11 Format *frame* data serial UART

Sumber: Atmel (2010:141)

dengan:

St = *Bit start* selalu berlogika rendah

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

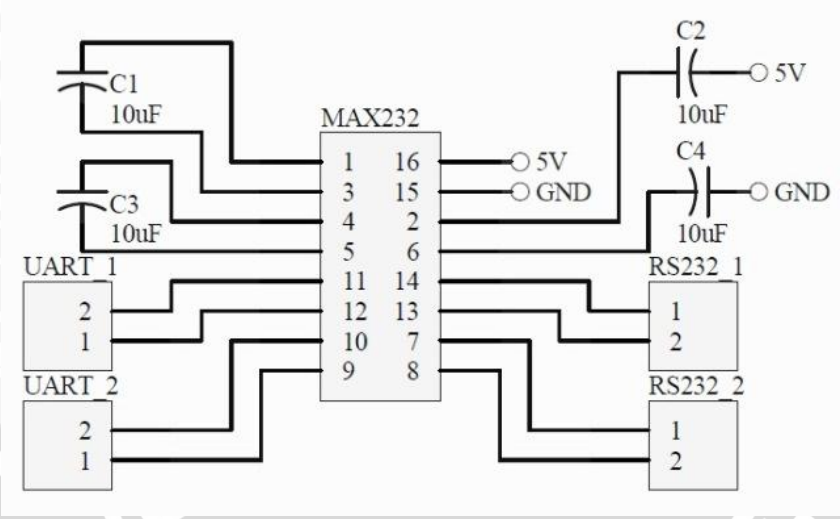
P = *Bit* paritas

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

IDLE = Tidak ada data yang ditransfer pada Rx dan Tx, IDLE selalu berlogika tinggi.

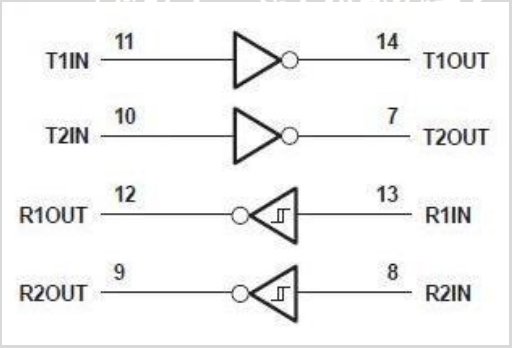
## 2.8 MAX 232

MAX232 adalah chip yang digunakan untuk menghubungkan antara isyarat RS232 dan UART. MAX232 mempunyai 16 pin dan bekerja pada tegangan 5V. Antarmuka RS232 kebanyakannya terdapat pada komputer dan beberapa peralatan lain. Isyarat UART terdapat pula pada *microcontroller* atau IC (*Integrated Circuit*). Komunikasi RS232 dan komunikasi UART hampir sama dimana meliputi isyarat menghantar dan menerima (*transmit and receive*). Apa yang membedakan kedua komunikasi ini hanya level tegangannya.




Gambar 2.12 Rangkaian *driver* RS-232

Komunikasi RS232 menggunakan tegangan -15V hingga +15V saat isyarat UART dicatu tegangan 0V hingga 5V. Level tegangan logika tinggi untuk RS232 adalah -15V hingga -3V, dan logika rendahnya adalah 3V hingga 15V. Level tegangan logika tinggi untuk UART adalah 2V hingga 5V, dan logika rendahnya adalah 0V hingga 0.8V.




Gambar 2.13 Susunan pin dari IC MAX232

MAX232 mempunyai dua set UART/RS232 converter. Pin 11 dan 10 adalah input UART atau UART Transmit, pin 14 dan 7 adalah output RS232 atau RS232 Receive. Pin 12 dan 9 adalah output UART atau UART Receive, pin 13 dan 8 adalah input RS232 atau RS232 Transmit.

# BAB III
# METODE PENELITIAN

## 3.1 Metode Penelitian

Penyusunan proposal ini didasarkan pada masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat bekerja sesuai dengan yang direncanakan dengan mengacu pada rumusan masalah. Langkah-langkah yang perlu dilakukan untuk merealisasikan alat yang dirancang adalah penentuan spesifikasi alat, studi literatur, perancangan dan pembuatan alat, dan pengambilan kesimpulan.

## 3.2 Spesifikasi Alat

Spesifikasi alat secara global diterapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan, yaitu:

- Servo yang digunakan servo HiTech dengan tipe 5685 dan Corona DS238HV
- Mikrokontroller yang dipakai adalah ATMega32 buatan Atmel yang berfungsi sebagai pemroses data dari PC (*Personal Computer*)
- Mikrokontroller menggunakan suplai tegangan 5V DC
- Robot yang digunakan berbentuk humanoid bagian lengan kanan
- Servo menggunakan suplai baterai 2 cell 7,4 V
- Alat bekerja menggunakan komunikasi serial

## 3.3 Studi Literatur

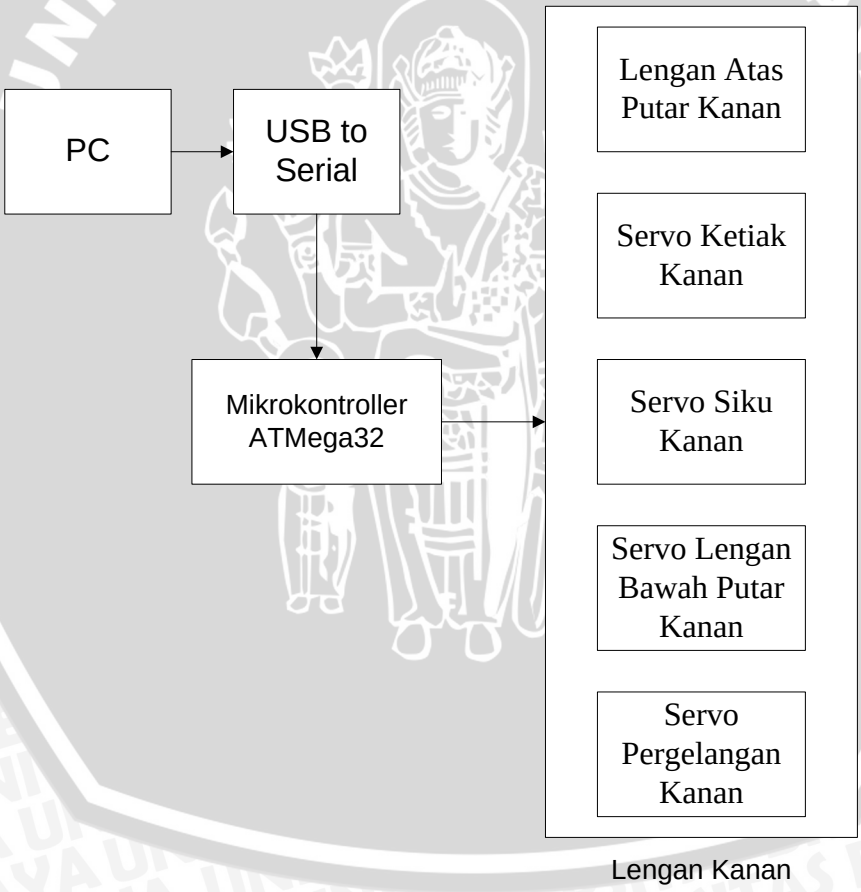
Studi literatur dilakukan untuk mempelajari teori penunjang sistem yang dibutuhkan dalam perencanaan dan pembuatan alat. Teori yang diperlukan antara lain berkaitan dengan rangkaian mikrokontroller ATMega32, membuat aplikasi dengan Qt, komunikasi serial, dan cara kerja motor DC servo.

### 3.4 Perancangan dan Perealisasian Alat

Perancangan dan perealisasian alat dalam penelitian ini dibagi menjadi dua bagian, yaitu *hardware* dan *software.*

#### 3.4.1 Perancangan *Hardware*

Dalam perancangan sistem ini pada bagian *hardware* terdapat beberapa blok diagram yang meliputi komputer sebagai pengirim data atau input ke mikrokontroller, USB to serial, mikrokontroller pengolah data, servo sebagai aktuator, LCD sebagai penampil sudut servo.




Gambar 3.1 Diagram blok sistem Keseluruhan

Penjelasan masing-masing blok diagram:

a. PC (*Personal Computer*)

   Berfungsi sebagai pengontrol yang mengirimkan data koordinat ke mikrokontroller dan menyimpan aplikasi yang akan dibuat.

b. USB to TTL

   Merupakan rangkaian yang digunakan untuk komunikasi serial atau sebagai jembatan data antara PC dengan mikrokontroller. Rangkaian ini digunakan karena adanya perbedaan level tegangan yang terjadi pada PC dengan Mikrokontroller, komunikasi yang terjadi merupakan UART (*Universal Asynchronous Receiver/Transmitter*). Komunikasi ini hanya satu arah yaitu dari PC untuk menggerakkan servo tanpa ada umpan balik ke PC.

c. ATMega32

   Merupakan IC Mikrokontroller yang berfungsi sebagai pemroses data koordinat dari PC (*Personal Computer*) agar dapat menggerakkan servo sesuai keinginan.

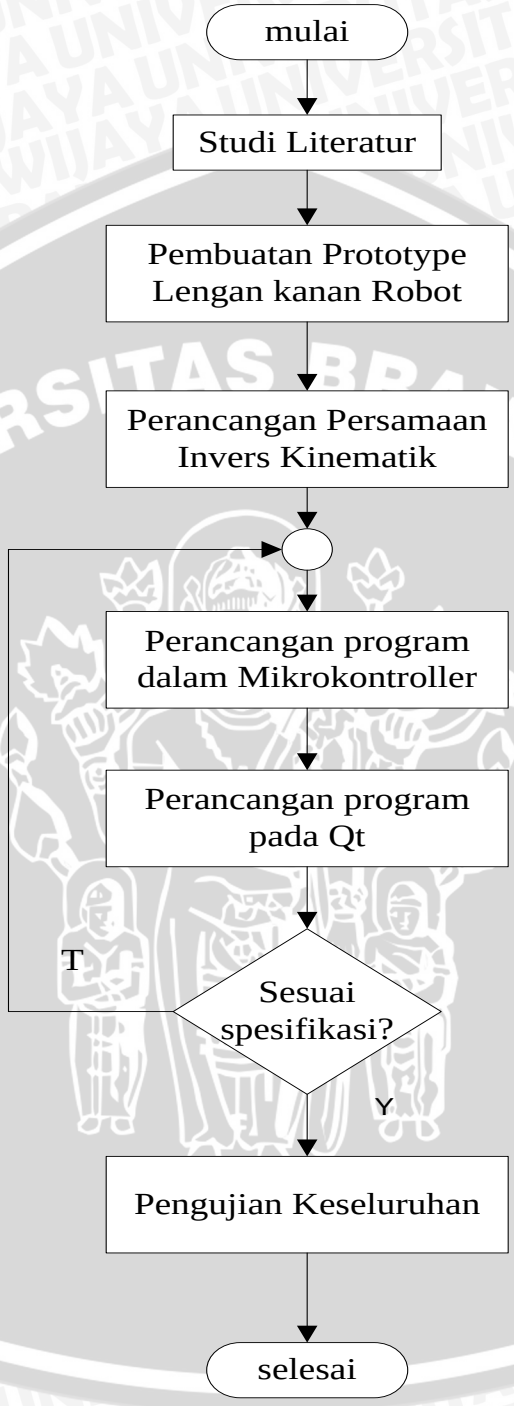
d. Servo Lengan Kanan

   Merupakan bagian dari robot yang berfungsi sebagai aktuator dan pada alat ini merupakan bagian yang akan dikendalikan sesuai dengan keinginan. Jenis servo yang digunakan adalah motor servo standar $180^0$ sebanyak 5 buah antara lain:

   i. Servo Lengan Atas Putar Kanan
   ii. Servo Ketiak Kanan
   iii. Servo Siku Kanan
   iv. Servo Lengan Bawah Putar Kanan
   v. Servo Pergelangan Kanan.

### 3.4.2 Perancangan dan Pembuatan Perangkat Lunak (*Software*)

Perancangan perangkat lunak berupa *flowchart* (diagram alir) bahasa pemrograman yang dipakai untuk ATMega32 adalah bahasa C dan pada Qt menggunakan bahasa C.




Gambar 3.2 Diagram alir pembuatan *prototype* Lengan Kanan Robot

### 3.4.3 Pengujian Alat

Analisis kinerja alat apakah sesuai dengan yang direncanakan maka dilakukan pengujian sistem. Pengujian dilakaukan pada masing-masing blok pada

perancangan *hardware* serta pengujian keseluruhan untuk mengetahui *software* dapat berjalan atau tidak.

#### 3.4.3.1 Pengujian Perangkat Keras (*Hardware*)

Pada bagian ini pengujian dilakukan pada masing-masing blok. Pengujian ini bertujuan untuk mengetahui apakah masing-masing blok dapat bekerja sesuai dengan fungsinya seperti yang telah direncanakan. Pengujian tersebut meliputi:

1. Pengujian Serial Terminal

   Pengujian yang dilakukan untuk melihat kinerja komunikasi PC ke Mikrokontroller.
2. Pengujian sinyal PWM

   Pengujian ini dilakukan untuk menguji keluaran lebar sinyal *high* PWM dari masukan data pada mikrokontroller.
3. Pengujian Motor DC Servo

   Pengujian ini bertujuan untuk mengetahui keluaran sudut motor DC servo dari sinyal PWM yang diberikan

#### 3.4.3.2 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem ini dengan menyambungkan semua *hardware* yang dibuat berdasarkan blok diagram dan memasukkan program berupa *software* yang bekerja untuk mengendalikan *hardware* yang telah dibuat. Sistem bekerja dengan baik jika dapat berjalan sesuai *flowchart* yang telah direncanakan.

## 3.5 Pengambilan Kesimpulan

Kesimpulan didapat berdasarkan hasil perealisasian sistem Penentuan Sudut Lengan Robot Humanoid Berdasarkan Koordinat Yang Dikirim Dari PC Menggunakan User Interface Yang Dibuat Dari Qt. Beberapa hal hasil pengujian disampaikan dalam kesimpulan disertai realita yang disusun secara berurutan.

# BAB IV
# PERANCANGAN DAN PEMBUATAN ALAT

Perancangan dan pembuatan alat dilakukan secara bertahap dalam bentuk blok diagram sehingga memudahkan dalam analisisnya. Perancangan alat terdiri atas perancangan perangkat keras dan perancangan perangkat lunak.

## 4.1 Perancangan Diagram Blok Sistem

Perancangan alat diawali dengan pembuatan diagram blok sistem secara keseluruhan. Diagram blok sistem secara keseluruhan ditunjukkan dalam Gambar 4.1.


Gambar 4.1 Diagram Blok Sistem keseluruhan

Fungsi masing-masing bagian dalam diagram blok sistem tersebut adalah sebagai berikut:

1) PC berfungsi sebagai antarmuka untuk mengirimkan data koordinat ke Mikrokontroller.
2) USB *to* Serial berfungsi untuk komunikasi serial atau sebagai jembatan data antara PC dengan mikrokontroller. Rangkaian ini digunakan karena adanya perbedaan level tegangan yang terjadi pada PC dengan Mikrokontroller, komunikasi yang terjadi merupakan UART (*Universal Asynchronous Receiver/Transmitter*). Komunikasi ini hanya satu arah yaitu dari PC untuk menggerakkan servo tanpa ada umpan balik ke PC.

3) ATMega32 berfungsi sebagai pemroses data koordinat dari PC (*Personal Computer*) agar dapat menggerakkan servo sesuai gerakan tari yang diinginkan dan koordinat yang sesuai.
4) Servo Lengan Kanan Merupakan bagian dari robot yang berfungsi sebagai aktuator dan pada alat ini merupakan bagian yang akan digerakkan menuju koordinat yang dituju. Jenis servo yang digunakan adalah motor servo standar $180^0$ sebanyak 5 buah yang daerah kerjanya disesuaikan dengan gerakan tarinya antara lain:
    a. Servo Lengan Atas Putar Kanan; memiliki *range* sudut $0^0$ sampai $90^0$.
    b. Servo Ketiak Kanan; memiliki *range* sudut $0^0$ sampai $90^0$.
    c. Servo Siku Kanan; memiliki *range* sudut $0^0$ sampai $90^0$.
    d. Servo Lengan Bawah Putar Kanan; memiliki *range* sudut $0^0$ sampai $90^0$.
    e. Servo Pergelangan Kanan; memiliki *range* sudut $0^0$ sampai $90^0$.

Saat semua *hardware* seperti pada Diagram Blok dalam Gambar 4.1 sudah tersambung seluruhnya maka langkah pertama yang harus dilakukan adalah memberi nilai koordinat pada antarmuka di PC. Nilai tersebut dikirim dari PC ke Mikrokontroller untuk selanjutnya di Mikrokontroller tersebut melakukan proses agar dapat menggerakkan servo. Proses tersebut antara lain adalah mengkonversi nilai dari tipe data *char* ke *float,* menghitung nilai sudut dari persamaan Invers Kinematik, dan konversi nilai sudut ke nilai PWM. Nilai PWM tersebut yang akan menggerakkan servo agar lengan robot mencapai koordinat tujuan. Dalam hal ini yang mencapai bagian lengan robot yang mencapai koordinat tujaun adalah ujung lengan robot.

## 4.2 Perancangan Perangkat Keras

Perancangan dan pembuatan perangkat keras terdiri atas perancangan mekanik dan perancangan elektronik. Perancangan mekanik terdiri atas perancangan robot humanoid lengan kanan. Sedangkan perancangan elektronik terdiri atas perancangan rangkaian *Mainboard Microcontroller*

### 4.2.1 Perancangan Robot Humanoid Lengan Kanan

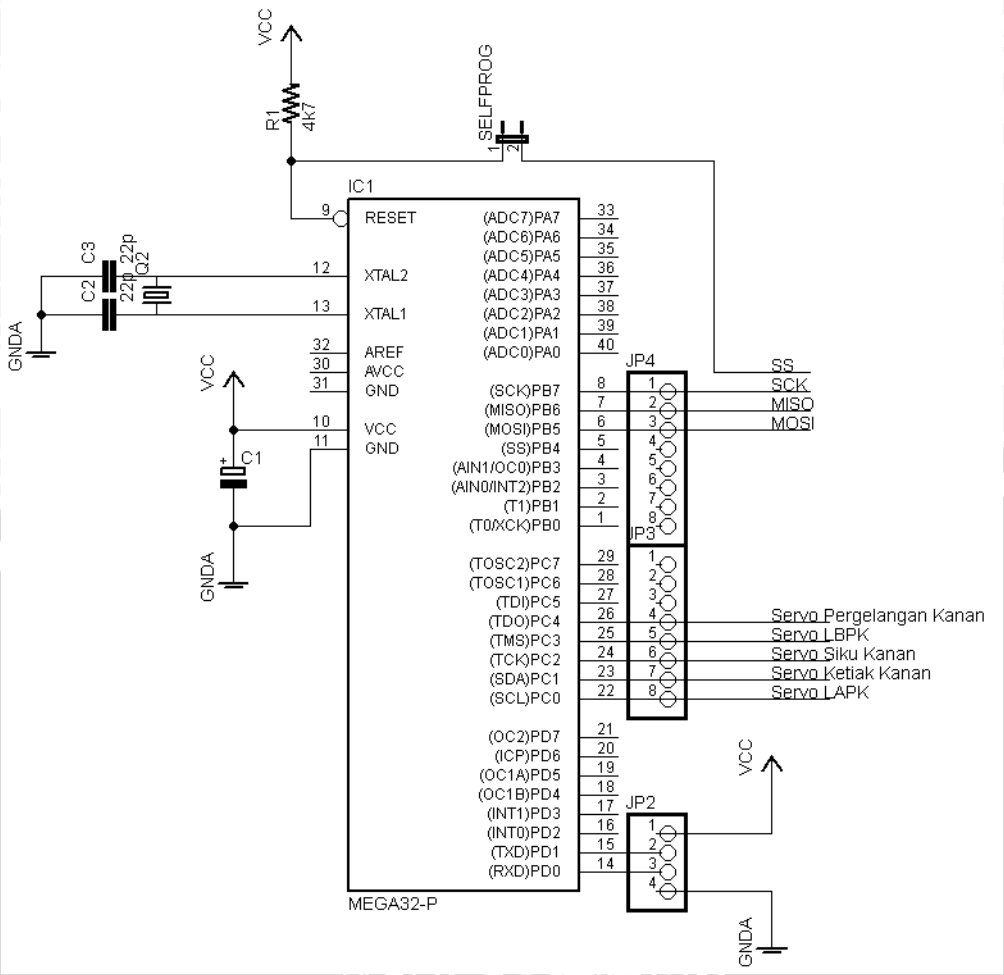
Bentuk mekanik alat disini utamanya adalah lengan robot yang terdiri dari lima DOF (*degree of freedom*). Dimensi lengan robot tidak disesuaikan dengan aturan dari KRSI (Kontes Robot Seni Indonesia). Susunannya terlihat dalam Gambar 4.3.

### 4.2.2 Perancangan *Mainboard Microcontroller*

Rangkaian ini merupakan basis dari sistem elektronik yang digunakan. Rangkaian ini mengakomodasi *minimum system* ATMega32 dengan berbagai perangkat luar. Fitur yang tersedia untuk rangkaian ini adalah soket ATMega32 lima pin *general* I/O, satu paket antarmuka UART, serta konektor catu daya.

Rangkaian ini yaitu ditunjukkan dalam Gambar 4.2, menghubungkan pin ATMega32 dengan keluaran pada *mainboard.* Sumber *clock* menggunakan *External* XTAL 11,0592 MHz. Kapasitor kembar yang menghubungkan dua kaki XTAL bernilai 22 pF. Resistor *Pull-up* yang menghubungkan pin RESET dan $V_{CC}$ adalah 4,7 kΩ.

Pada antarmuka *servo* terdapat konektor catu daya, yaitu tegangan langsung baterai *Lithium Polymer 2 cell* (7,3 – 8,6 V), sesuai spesifikasi yang digunakan. Lima pin *general* I/O yaitu PORTC.0 hingga PORTC.4 digunakan sebagai keluaran sinyal PWM dari Mikrokontroller yang digunakan sebagai masukan pin data *servo.*



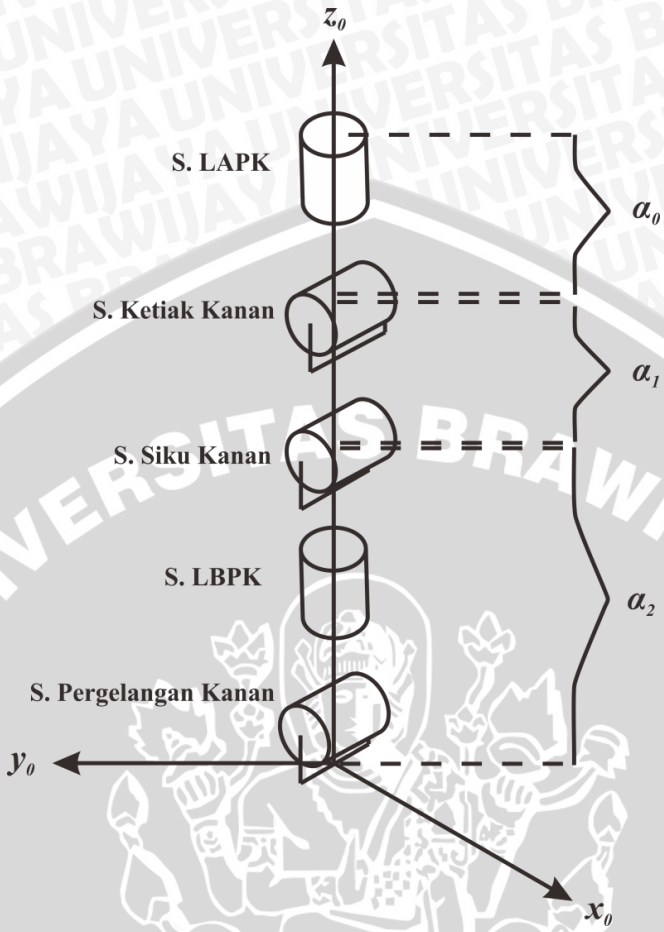

Gambar 4.2 Rangkaian *Mainboard Microcontroller*

## 4.3 Perancangan Invers Kinematik

Perancangan dan pembuatan perangkat lunak terlebih dahulu harus mendapatkan nilai sudut masing-masing servo. Langkahnya antara lain terdiri atas pendeskiripsian frame lengan robot, perancangan kinematika lengan robot, dan perhitungan sudut.

### 4.3.1 Pendeskripsian *frame* lengan robot

Langkah awal yaitu menentukan panjang link pada lengan robot, lalu dapat dirancang kinematika lengan robot sesuai pada Persamaan (2-1) hingga (2-7).

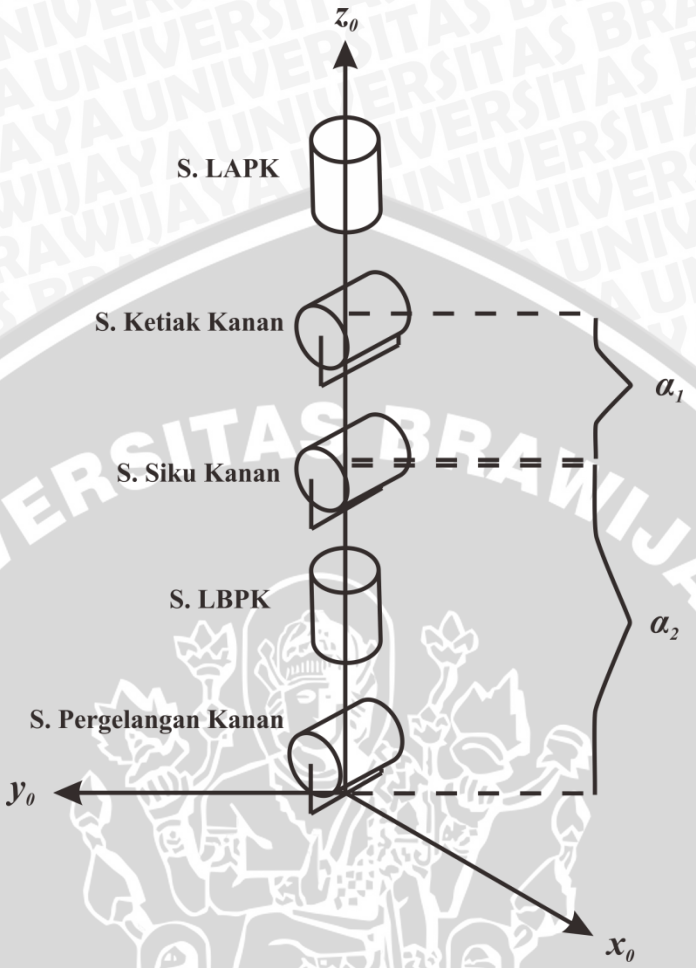



Gambar 4.3 Pendeskripsian *frame* lengan robot

Tabel 4.1 Panjang masing-masing *link* pada lengan robot.

| Link | Panjang (cm) |
|---|---|
| 0 | 4.5 |
| 1 | 6 |
| 2 | 7 |

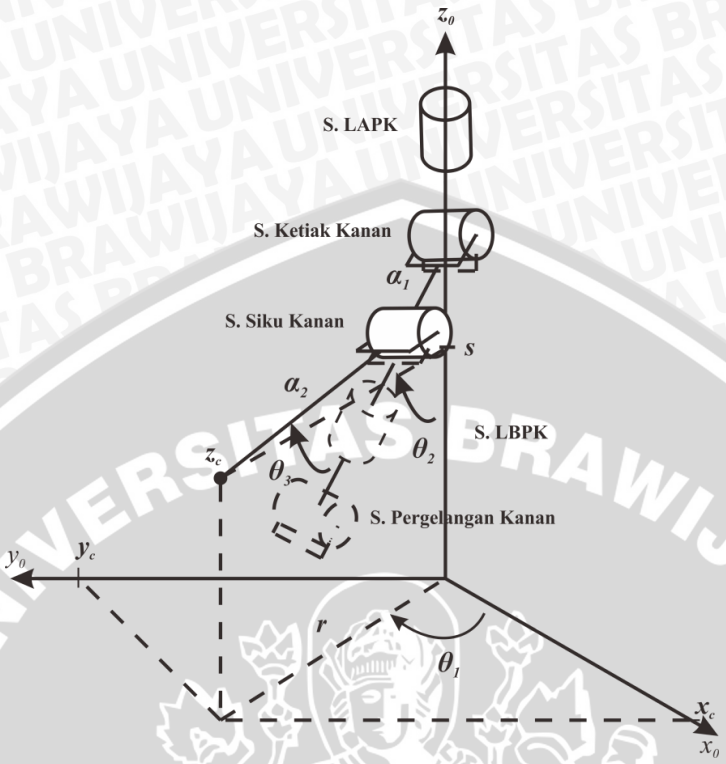
Pada perancangan kinematika lengan robot posisi awalnya adalah seperti ditunjukkan dalam Gambar 4.4 dimana posisi ujung lengan robot dianggap terletak pada koordinat (0,0,0). Koordinat tujuan terletak pada Kuadran I dimana sumbu $x_0$, $y_0$, dan $z_0$ adalah positif. Pada perhitungan invers kinematik dalam perancangan ini nilai *link 0* atau $\alpha_0$ diabaikan karena tidak masuk di dalam Persamaan.




Gambar 4.4 Posisi awal dan koordinat tujuan yang terletak pada Kuadran I

### 4.3.2 Perancangan Kinematika Lengan Robot

Posisi akhir yang dituju menyebabkan terjadinya perubahan nilai sudut servo pada beberapa servo seperti terlihat dalam Gambar 4.5 dimana terjadi perubahan nilai sudut pada Servo Ketiak Kanan, Servo Lengan Atas Putar Kanan (LAPK), dan Servo Siku Kanan untuk mencapai posisi titik yang diinginkan. Setelah sampai pada posisi titik yang diinginkan maka servo tertentu akan melakukan gerakan tari yaitu Servo Lengan Bawah Putar Kanan dan Servo Pergelangan Kanan.




Gambar 4.5 Posisi akhir lengan robot berdasarkan koordinat yang dituju

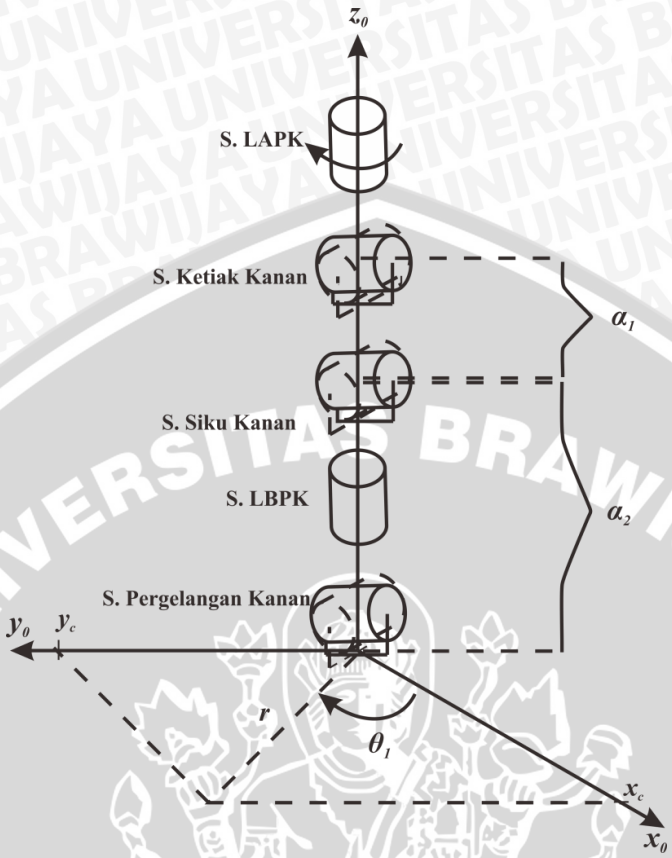
### 4.3.3 Perhitungan Sudut

Koordinat tujuan ($x_c$, $y_c$, $z_c$) ini untuk mencapainya terdapat 3 servo yang harus melakukan perubahan sudut. Pertama yang bergerak adalah Servo Lengan Atas Putar Kanan yang berputar sejauh $\theta_1$ seperti ditunjukkan dalam Gambar 4.6 dan bergerak sesuai dalam Persamaan (4-1).

$$\theta_1 = \tan^{-1}\frac{y_c}{x_c} \tag{4-1}$$

Kedua yang bergerak adalah Servo Ketiak Kanan yang bergerak sejauh $\theta_2$. Nilai $\theta_2$ didapat dengan terlebih dahulu mendapatkan nilai $\theta_3$ seperti ditunjukkan dalam Gambar 4.7 menggunakan Aturan Kosinus. Penjelasan Gambar 4.7 prinsipnya sama dengan Gambar 2.2, namun yang berbeda adalah posisi basisnya sehingga menyebabkan beberapa nilai mengalami penyesuaian maka Persamaan (4-2) untuk mendapatkan nilai sudut $\theta_3$ berasal dari Persamaan (2-1) sehingga

$$\cos\theta_3 = \frac{r^2+s^2-\alpha_1^2-\alpha_2^2}{2\alpha_1\alpha_2} := D. \tag{4-2}$$




Gambar 4.6 Gerakan pertama menuju koordinat tujuan

dimana $s = \alpha_1 + \alpha_2 - z_c$ dan $r = \sqrt{x_c^2 + y_c^2}$. Nilai s dalam Persamaan (4-2) berbeda dengan dalam Persamaan (2-1) karena posisi basis yang berbeda. Persamaan (4-3) berasal dari Persamaan (2-2) sehingga $\theta_3$ dapat ditulis

$$\theta_3 = \cos^{-1} D. \tag{4-3}$$

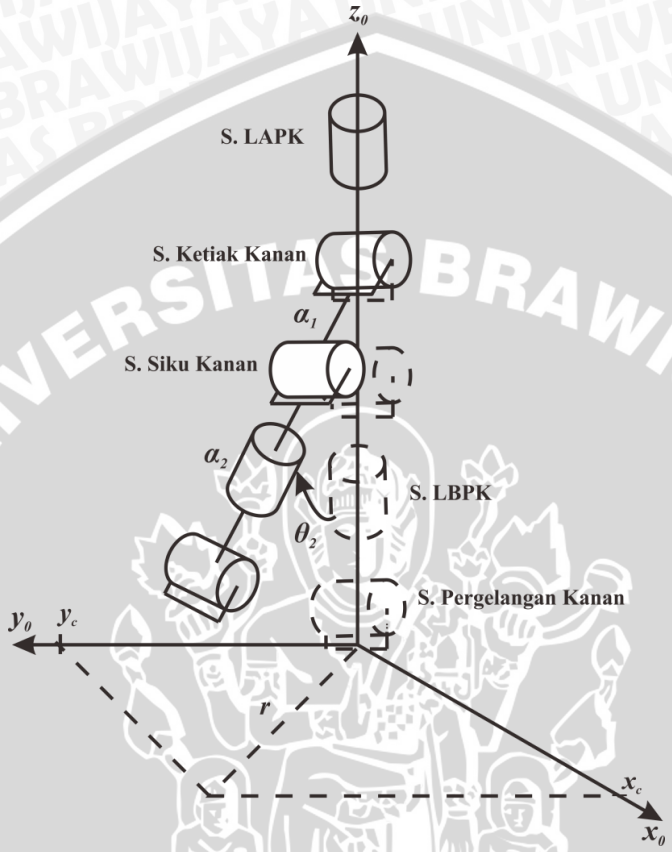
Cara terbaik mendapatkan $\theta_3$ dengan memperhatikan nilai *cos* $\theta_3$ dari Persamaan (4.3) maka *sin* $\theta_3$ didapatkan

$$\sin \theta_3 = \pm \sqrt{1 - D^2} \tag{4-4}$$

Persamaan (4-4) berasal dari Persamaan (2-3) sehingga nilai $\theta_3$ dapat ditulis sebagai

$$\theta_3 = \tan^{-1} \frac{\pm\sqrt{1-D^2}}{D}. \tag{4-5}$$

Persamaan (4-5) berasal dari Persamaan (2-4)



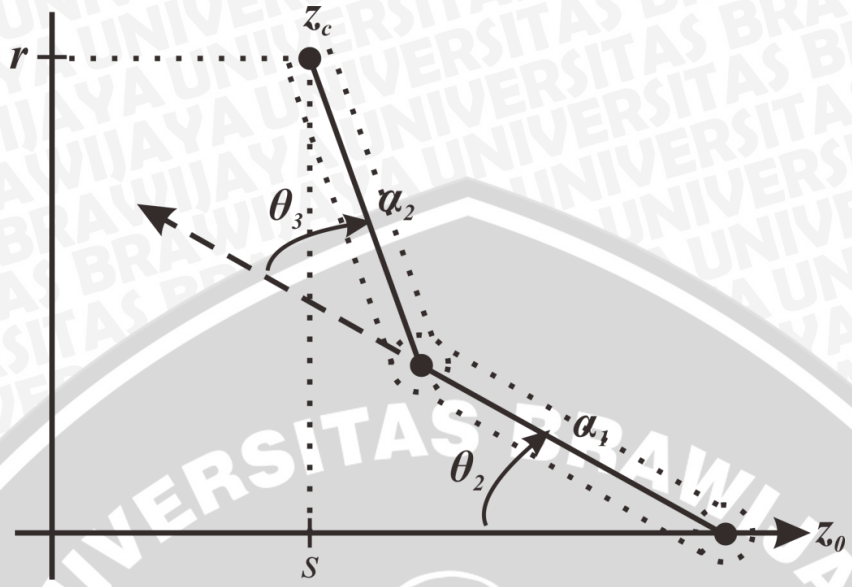

Gambar 4.7 Gerakan kedua menuju koordinat tujuan

Keuntungan dari pendekatan ini adalah tanda positif dan negatif pada Persamaan (4.4) yang mengakibatkan posisi Servo Siku Kanan ada di atas atau di bawah dapat terselesaikan.

$$\theta_2 = \tan^{-1}\frac{r}{s} - \tan^{-1}\left(\frac{\alpha_2 \sin\theta_3}{\alpha_1 + \alpha_2 \cos\theta_3}\right) \quad (4\text{-}6)$$

Persamaan (4-6) berasal dari Persamaan (2-5).

Oleh karena $\theta_2$ berhubungan dengan $\theta_3$ secara fisik maka untuk setiap nilai $\theta_2$ yang berbeda adalah dari nilai $\theta_3$ yang dipilih.

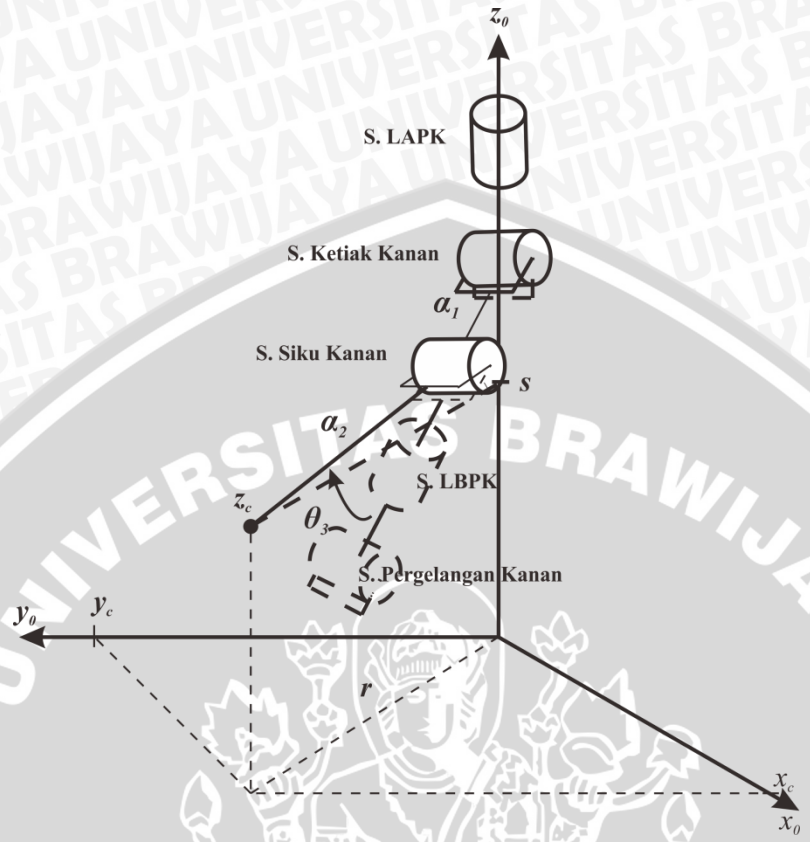



Gambar 4.8 Proyeksi Servo Ketiak Kanan pada bidang $z_0$-$r$

Servo yang ketiga yang bergerak adalah Servo Siku Kanan sejauh $\theta_3$. Saat servo ini selesai bergerak maka itulah posisi koordinat tujuan ($x_c$, $y_c$, $z_c$) yang dituju.

Setelah lengan robot mencapai koordinat tujuan maka selanjutnya adalah melakukan gerakan tari yang dimana yang bergerak adalah Servo Lengan Bawah Putar Kanan dan Servo Pergelangan Kanan.

Gerakan tari yang pertama adalah dari Servo Lengan Bawah Putar Kanan yang bergerak dari posisi 0° ke 45° kembali ke 0° lalu ke -45° lalu kembali ke posisi 0°. Setelah itu gerak selanjutnya adalah dari Servo Pergelangan Kanan yang bergerak 30° dari posisi awal.




Gambar 4.9 Gerakan ketiga menuju koordinat tujuan

## 4.4 Perancangan Perangkat Lunak

Perancangan perangkat lunak untuk memproses sistem secara keseluruhan ini dapat diartikan bahwa bagaimana suatu subsistem dapat terintegrasi satu sama lain sehingga menjadi satu sistem yang kompleks. Bagaimana perangkat masukan berupa PC yang mengirim data koordinat dapat diterima lalu diproses oleh mikrokontroller untuk menggerakkan servo guna menuju koordinat yang dituju lalu dapat menggerakkan gerakan tari. Untuk melakukan serangkaian proses tersebut maka dalam penelitian ini dirancang dua perangkat lunak pemroses sistem yaitu pada antarmuka yang dibuat dari software Qt dan Mikrokontroller.

### 4.4.1 Perancangan Perangkat Lunak pada Mikrokontroller

Data yang diterima oleh Mikrokontroller harus memiliki format data:

$$xxyyzzu \qquad (4\text{-}7)$$

dimana:

x = nilai koordinat tujuan x yang dikirim dari PC (berformat 2 digit)

y = nilai koordinat tujuan y yang dikirim dari PC (berformat 2 digit)

z = nilai koordinat tujuan z yang dikirim dari PC (berformat 2 digit)

u = sebagai penanda bahwa data tersebut telah diterima di Mikrokontroller

Pada tabel ASCII, karakter ‘u’ memiliki nilai desimal 117. Maka penulisan program di Mikrokontroller adalah.

```
if (data==117)
  {
   xc=konversi(rx_buffer[0],rx_buffer[1]);
   yc=konversi(rx_buffer[2],rx_buffer[3]);
   zc=konversi(rx_buffer[4],rx_buffer[5]);

   hitung_servo();
   rx_wr_index=0;                                    (4-8)
   state=1;
   }
   else
   {
    state=0;
   }
}
```

Mikrokontroller akan memproses nilai yang diterima bila dapat membaca karakter ‘u’. Proses yang pertama kali dilakukan adalah konversi nilai yang dibaca sebagai nilai koordinat *x*, nilai koordinat *y* dan nilai koordinat z. Nilai yang sudah dipilah tersebut (bertipe data *char*) lalu diproses dalam fungsi konversi sehingga bertipe data *float*. Nilai dipilah untuk nilai puluhan dan nilai satuan. Nilai puluhan yang pada persamaan (4-9) dilambangkan dengan variabel ‘a’ dikalikan dengan

10 dan nilai satuan dilambangkan dengan variabel ‘b’. Maksud dari ‘48’ dalam persamaan (4-9) adalah nilai desimal dari karakter 0 pada tabel ASCII.

```
float konversi(char a, char b)
{
 float x = (a-48) * 10;
 float y = b-48;
 float z;
 return z = x+y;
}
```
(4-9)

Setelah nilai sudah dikonversi, didapat masing-masing nilai koordinat tujuan (x,y,z) dimana nilai tersebut adalah sebagai masukan fungsi untuk menghitung nilai invers kinematiknya. Penulisan program invers kinematik terlihat pada persamaan (4-10).

```
hitung_servo ()
{
 alfa1=6;
 alfa2=7;

 tetha1= atan (yc/xc)*180/pi;

 s= alfa1+alfa2-zc;
 r= sqrt(xc*xc+yc*yc);
 //D= ((r*r+s*s-alfa1*alfa1-alfa2*alfa2)/(2*alfa1*alfa2))
 D1=(r*r+s*s-alfa1*alfa1-alfa2*alfa2);
 D2=(2*alfa1*alfa2);
 D= (D1/D2);
 tetha3= acos (D)*180/pi;

 sintetha3= sqrt (1-D*D);
```

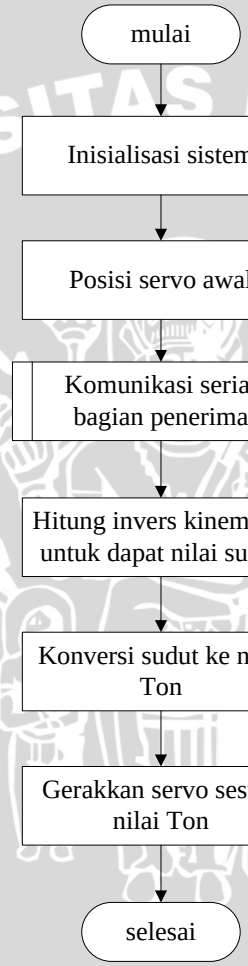
```
tetha2a= atan (r/s)*180/pi;
ab= alfa2*sintetha3;
bb= alfa1+alfa2*D;
tetha2b= atan (ab/bb)*180/pi;
tetha2= tetha2a - tetha2b;

      //servo1=-150,67*tetha1+23413
      servo1a=tetha1*150,67;
      servo1b=-servo1a;
      servo1=servo1b+23413;

        //servo2=95,433*tetha2+10073
        servo2a=tetha2*95,433;
        servo2=servo2a+10073;

          //servo3=-99,687*tetha3+23012
          servo3a=tetha3*99,687;
          servo3b=-servo3a;
}
```

(4-10)

Persamaan konversi nilai sudut ke nilai $T_{ON}$ terletak di dalam fungsi pada persamaan (4-10). Terdapat tiga variabel masing-masing bernama servo1 untuk nilai $T_{ON}$ pada servo Lengan Atas Putar Kanan, servo2 untuk nilai $T_{ON}$ pada servo Ketiak Kanan, dan servo3 untuk nilai $T_{ON}$ pada Servo Siku Kanan. Nilai servo1 berasal dari persamaan (5-4), nilai servo2 berasal dari persamaan (5-5) dan nilai servo3 berasal dari persamaan (5-6). Pada persamaan (4-8) bila data yang diterima tersebut benar (data==117) maka akan melakukan kondisi ‘state=1’ sedangkan bila tidak akan melakukan kondisi ‘state=0’. Kondisi ‘state=0’ adalah kondisi awal masing-masing servo dan ‘state=1’ adalah kondisi masing-masing servo saat sudah didapat nilai Invers Kinematik dimana nilainya adalah servo1, servo2, dan servo3. Variabel servo1 adalah nilai yang diberikan pada servoA[0] pada kondisi

'state=1'. Variabel servo2 adalah nilai yang diberikan pada servoA[1] pada kondisi 'state=1'. Variabel servo3 adalah nilai yang diberikan pada servoA[2] pada kondisi 'state=1'.

Diagram alir perangkat lunak sistem pada Mikrokontroller ditunjukkan dalam Gambar 4.10



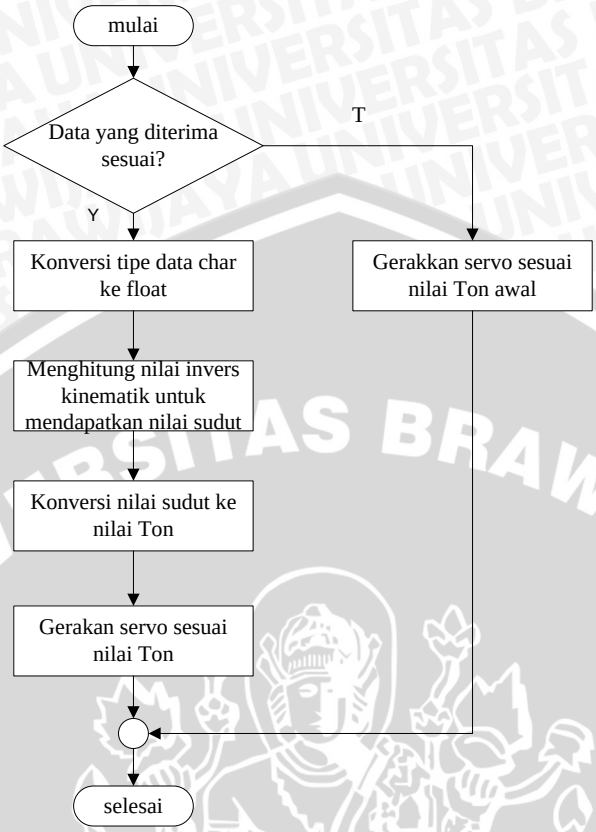

Gambar 4.10 Diagram alir pada Mikrokontroller

Proses ke-1 yaitu mikrokontroller melakukan inisialisasi sistem secara keseluruhan yaitu inisialisasi *interrupt, timer,* serta UART. *Timer* dan *Interrupt* digunakan untuk membangkitkan PWM agar dapat menggerakkan Motor DC *Servo*. UART digunakan sebagai komunikasi serial yang digunakan. Proses ke-2 adalah posisi awal lengan kanan robot dimana masing-masing servo diberikan nilai $T_{ON}$ awal. Selanjutnya pada proses ke-3, mikrokontroller menerima data

koordinat tujuan dari PC dengan terlebih dahulu melewati *USB to TTL Converter*. Proses ke-4 yaitu proses menghitung data koordinat yang sudah diterima sehingga keluarannya dihasilkan data sudut yaitu $\theta_1$, $\theta_2$, dan $\theta_3$. Proses ke-5 yaitu proses mengubah data sudut menjadi data digital untuk membangkitkan sinyal PWM melalui sebuah konversi. Pada fungsi ini, nilai digital tersebut akan dikonversi menjadi nilai $T_{ON}$ (lebar periode logika tinggi) sinyal PWM untuk menggerakkan Motor DC *Servo*. Pembangkitan sinyal PWM dilakukan oleh *Timer/Counter* yang sudah diinisialisasi pada awal program. Proses ke-6 yaitu menggerakkan servo sesuai sudut yang sudah didapat.

Proses konversi yang terjadi dalam perangkat lunak Mikrokontroller antara lain konversi tipe data karakter (dari PC) ke float (yang diolah Mikrokontroller), konversi koordinat ke sudut, serta sudut ke nilai $T_{ON}$ sinyal PWM.

Pada proses komunikasi serial dari PC data yang dikirim memiliki tipe data char sedangkan yang diproses di Mikrokontroller berupa tipe data float, Oleh sebab itu perlu dikonversi terlebih dahulu seperti ditunjukkan dalam Gambar 4.11. Saat data koordinat yang diterima oleh Mikrokontroller (yang masih berbentuk tipe data char) tidak sesuai dengan format data yang dikehendaki maka akan dihasilkan nilai $T_{ON}$ pada posisi awal, sedangkan bila nilainya sesuai maka akan melakukan konversi tipe data. Setelah itu nilai tersebut diolah pada Persamaan Invers Kinematik sehingga dihasilkan nilai sudut. Nilai sudut ini kemudian diolah agar didapat nilai $T_{ON}$.




Gambar 4.11 Diagram alir Komunikasi serial bagian penerima hingga menggerakkan *servo*

Servo yang digunakan tidak memiliki *datasheet* dari pabrikannya, jadi untuk mendapatkan parameter yang dibutuhkan untuk proses konversi, diperlukan karakterisasi. Karakterisasi diasumsikan bahwa proses konversi yang dijalankan adalah linier, maka program konversi bisa menggunakan hukum persamaan garis linier, yaitu:

$$y = m.x + c \tag{4-11}$$

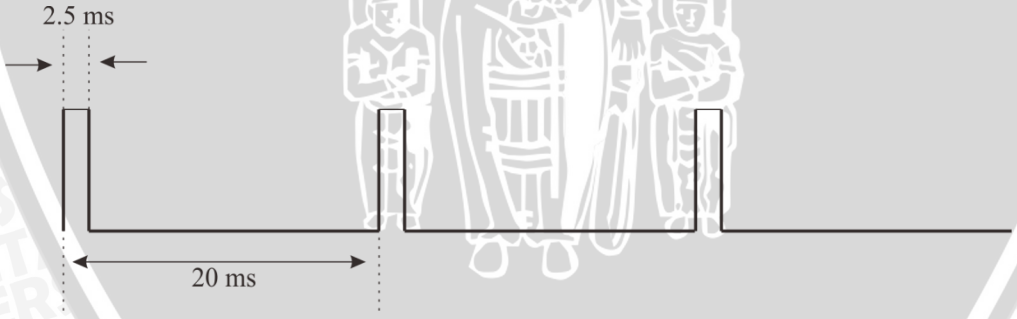
Variabel $y$ adalah keluaran proses konversi, $x$ adalah masukan proses konversi, sementara $m$ adalah gradien dan $c$ adalah konstanta yang nilainya dapat dicari dengan mengetahui dua buah persamaan garis di dua titik yaitu dengan mencari batas atas dan batas bawah yang diperlukan untuk setiap masukan dan keluaran, dapat dihasilkan dua persamaan dengan $m$ dan $c$ yang sama serta batas keluaran dan masukan yang berbeda.

$$y_0 = m.x_0 + c \tag{4-12}$$

$$y_1 = m.x_1 + c \quad (4\text{-}13)$$

Konversi sudut ke sinyal PWM dibagi menjadi dua tahap. Pertama, data sudut dikonversi menjadi nilai TOP yang menjadi parameter akhir siklus positif sinyal PWM. Kemudian nilai TOP tersebut dimasukkan ke dalam algoritma pembangkitan sinyal PWM yang dapat dilakukan dengan *Timer/Counter*.

Konversi sudut ke nilai TOP menggunakan persamaan linier, seperti pada Persamaan (4.7) untuk menemukan keluaran nilai TOP yang sesuai. Paramater yang digunakan adalah batas atas dan batas bawah nilai sudut sebagai masukan serta nilai TOP sebagai keluaran. Sudut yang digunakan adalah sudut efektif Motor DC *Servo*. Rentang nilai TOP didapatkan dari karakterisasi sudut motor DC *Servo* terhadap nilai TOP yang mempengaruhinya. Nilai TOP tertentu akan menghasilkan sinyal PWM dengan lebar $T_{ON}$ tertentu. Sementara itu Motor DC *Servo* memiliki batas lebar $T_{ON}$ sinyal PWM tertentu yang bisa digunakan untuk menggerakkan dan menahan posisi stall. Persamaan didapatkan dengan mencari nilai gradien dan konstanta melalui metode substitusi dua persamaan linier dua variabel.

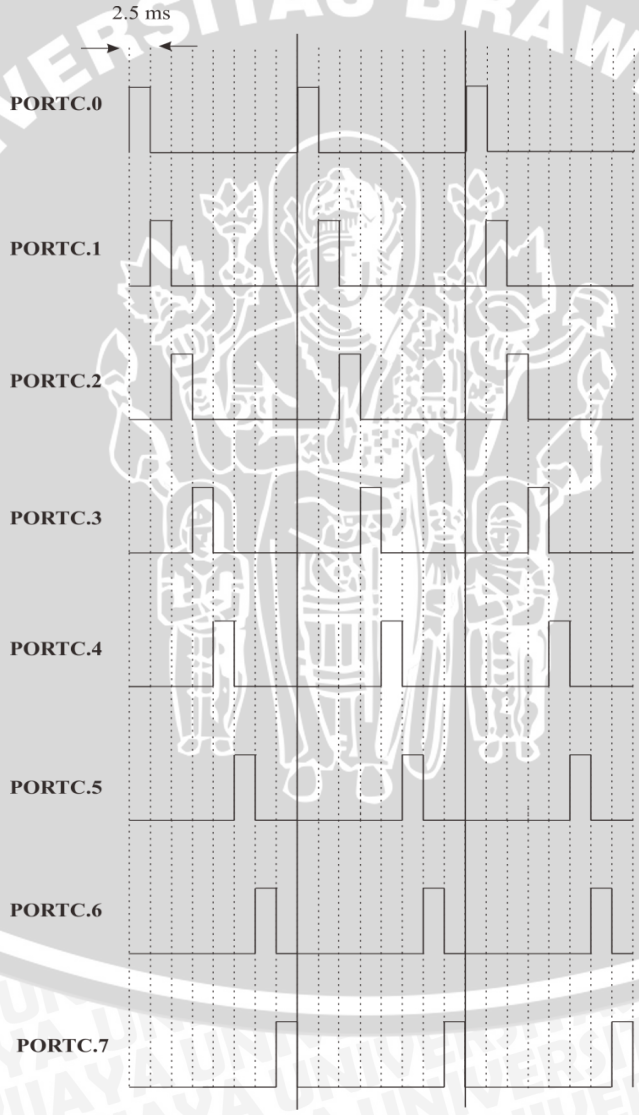



Gambar 4.12 Sinyal kontrol motor DC *Servo*

Selanjutnya adalah membangkitkan sinyal PWM. Gambar 4.12 menunjukkan sinyal PWM yang dibutuhkan untuk dapat mengontrol Motor DC *Servo*. Agar dapat mengontrol Motor DC *Servo*, diperlukan sinyal PWM dengan periode 20 ms. Sementara itu, siklus positifnya hanya membutuhkan lebar $T_{ON}$ maksimal sekitar 2,5 ms pada awal satu siklus periodik, selebihnya adalah siklus negatif. Sinyal ini harus berulang terus secara periodik, jika tidak maka *servo*

kehilangan kemampuan untuk mempertahankan posisi. Amplitudo pulsa yang dibutuhkan untuk menggerakkan servo antara 3-5 Vpp.

Motor DC *Servo* yang akan dikontrol tidak harus dikeluarkan melalui pin *Timer/Counter,* tetapi bisa juga melalui pin I/O. Dengan menggunakan pin I/O, servo yang bisa dikontrol jumlahnya lebih banyak. Untuk mengeluarkan sinyal PWM pada pin biasa, bisa dilakukan dengan menggilir keluaran *timer* secara bergantian ke setiap pin I/O, seperti ditunjukkan dalam Gambar 4.13.




Gambar 4.13 Sinyal kontrol Motor DC *Servo* jamak pada pin I/O

Pin I/O yang ingin digunakan, misalnya PORTC dapat diatur untuk mengeluarkan sinyal PWM dari satu buah keluaran *Timer/Counter*. Sinyal PWM pada satu *timer* diinterupsi setiap 2,5 ms, ditunjukkan oleh garis putus-putus dalam Gambar 4.13. Setelah interupsi selesai dieksekusi, pembangkitan sinyal PWM dimulai lagi, dan sinyal dikeluarkan ke pin selanjutnya. Proses berulang terus hingga maksimal delapan kali. Lebar $T_{ON}$ satu sinyal PWM pada satu pin yang berbeda dapat diatur melalui *Output Compare Register (*OCR*)* yang diisi oleh nilai TOP dari hasil konversi sudut. Dengan mengganakan satu buah *Timer/Counter1* (16 bit), dapat dihasilkan sinyal berdasarkan nilai TOP dari 0 hingga 65535. Dua buah OCR, yaitu OCR1A dan OCR1B juga dapat digunakan untuk menggerakkan hingga 16 buah *servo* untuk masing-masingnya delapan *servo.* Perhitungan nilai pada register OCRA dan OCRB diperoleh dari datasheet ATMega32 sebagai berikut:

$$f_{OCPWM} = \frac{f_{clk\_I/O}}{N.(1+TOP)} \quad (4\text{-}14)$$

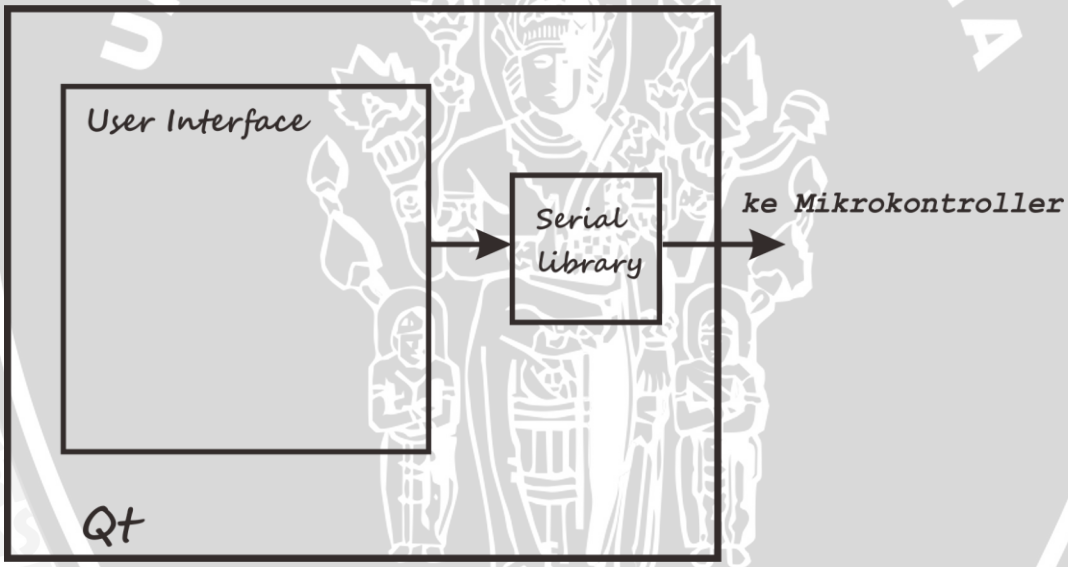
$$TOP = \frac{f_{clk\_I/O}}{N.f_{OCPWM}} - 1 \quad (4\text{-}15)$$

$f_{OC_{nx}PWM}$ adalah frekuensi PWM pada mode *output compare.* Dengan mengetahui frekuensinya, maka periode output compare adalah satu dibagi frekuensinya. $f_{clk\_I/O}$ adalah frekuensi clock yang digunakan, dalam perancangan ini menggunakan *clock* 11,0592 MHz. Nilai *N* adalah *prescaler* atau pembagi. TOP adalah nilai pada *Output Compare Register.*

### 4.4.2 Perancangan Antarmuka pada PC

Nilai yang diinginkan dapat diterima oleh Mikrokontroller memiliki format data seperti ditunjukkan dalam persamaan (4-7) yang harus bisa dikirim dari PC. Oleh sebab itu pada perancangan desain antarmuka pada Qt terdapat dua *library* yang dibutuhkan yaitu *user interface* dan serial *library* seperti ditunjukkan dalam Gambar 4.14.

*Library user interface* terletak pada program utama yang berisikan desain tampilan GUI. Pada *serial library* terdapat setting port serial, proses pengambilan nilai dari *Line edit* masing-masing nilai koordinat, proses menyeleksi nilai yang diinput, dan proses mengirim data serial ke Mikrokontroller.

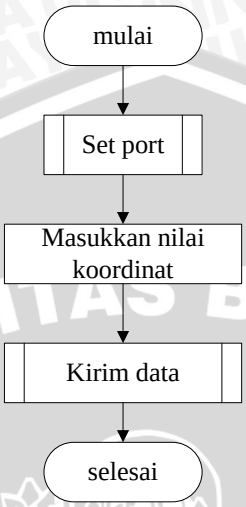



Gambar 4.14

Gambar 4.14 Desain *User Interface* dari Qt

Desain tampilan GUI yang dibutuhkan mancakup setting nilai COM yang sesuai antara letak *hardware USB to Serial* dengan setting di GUI, tombol ‘Connect’ dan ‘Disconnect’ sebagai “saklar” komunikasi serial pada PC, *Line edit* masing-masing koordinat sebagai letak input nilai yang diinginkan, dan tombol ‘Send’ untuk mengirim data koordinat yang sudah ditulis.

Diagram alir perangkat lunak antarmuka pada PC yang dibuat dengan software Qt ditunjukkan dalam Gambar 4.15.



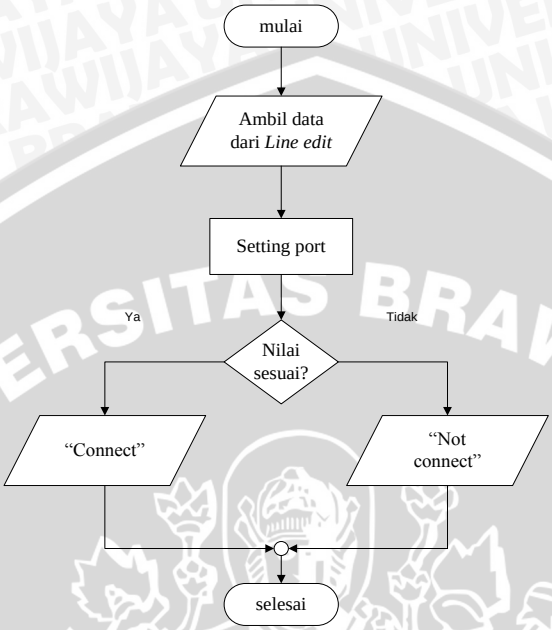

Gambar 4.15 Diagram alir antarmuka pada PC

Proses ke-1 adalah men-*setting port* yang tersambung ke USB to TTL agar sesuai dengan antarmuka dengan cara mengetikkan nama COMn (n=1,2,3...n) yang sesuai pada kolom port. Proses ke-2 adalah mengetikkan nilai koordinat yang dituju. Proses ke-3 adalah mengambil data yang sudah diketik tersebut lalu dikirim ke Mikrokontroller.

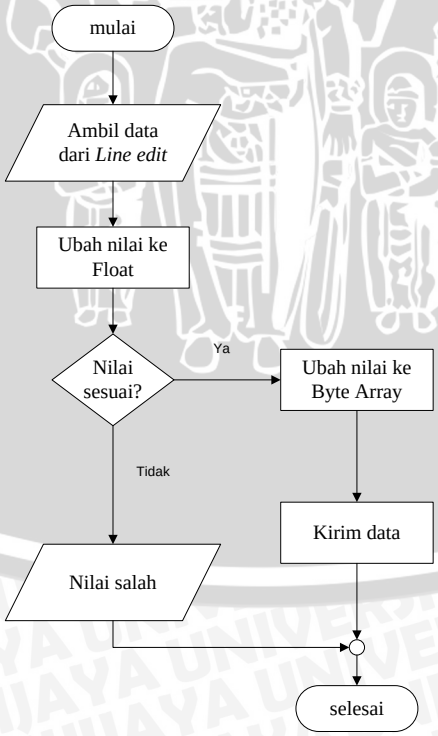
Proses *set port* memiliki algoritma yang ditunjukkan dalam Gambar 4.16. Proses ke-1 adalah mengambil data dari *line edit* letak port yang diketik. Proses ke-2 dicek apakah nama *port* yang diketik sudah sesuai atau tidak dengan letak USB to TTL, bila sesuai akan muncul tulisan ‘Connect’ pada antarmuka dan bila tidak sesuai akan muncul tulisan ‘Not Connect’.

Proses kirim data koordinat tujuan memiliki algoritma yang ditunjukkan dalam Gambar 4.17. Proses pertama adalah mengambil data yang sudah diketikkan di *Line edit* masing-masing koordinat x,y,z. Data yang diketik ini berbentuk String lalu diubah ke tipe data float untuk dicek apakah nilainya sesuai atau tidak, bila sesuai maka akan dikonversi ke *ByteArray* lalu data ini dikirim ke Mikrokontroller dan bila data ini tidak sesuai maka akan muncul tampilan pesan

salah, data tersebut tidak diubah ke *ByteArray* dan data tersebut tidak dikirim ke Mikrokontroller.




Gambar 4.16 Diagram alir *Set port*

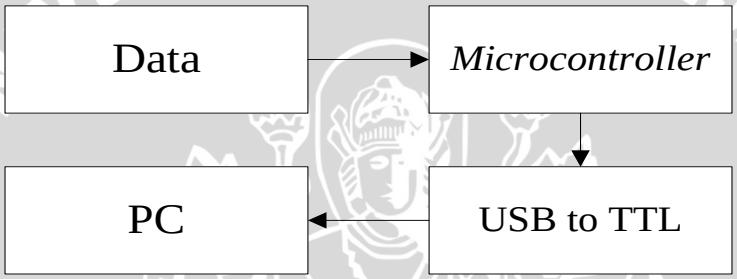



Gambar 4.17 Diagram alir Kirim data

# BAB V
# PENGUJIAN DAN ANALISIS

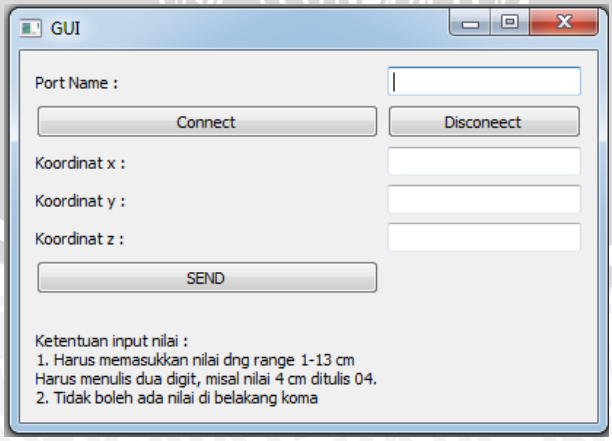
## 5.1 Pengujian *Serial Terminal*

Pengujian *serial terminal* dapat dilakukan dengan mengetikkan data tertentu ke dalam sebuah program, kemudian diatur agar menuliskan perintah ke komputer. Sementara dalam Gambar 5.1 menunjukkan pelaksanaan pengujian *Serial Terminal.*



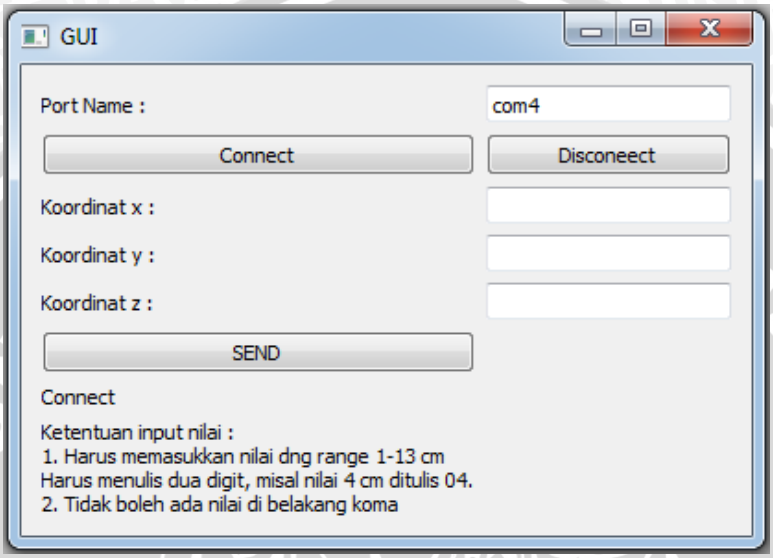

Gambar 5.1 Skema pengujian *USB to TTL*

Pada program *Interface* yang telah dibuat dengan software Qt seperti terlihat dalam Gambar 5.2 akan mengirim data dari PC kepada PC yang lainnya. PC lainnya ini berfungsi untuk melihat data yang dikirim kepada Mikorokontroller.



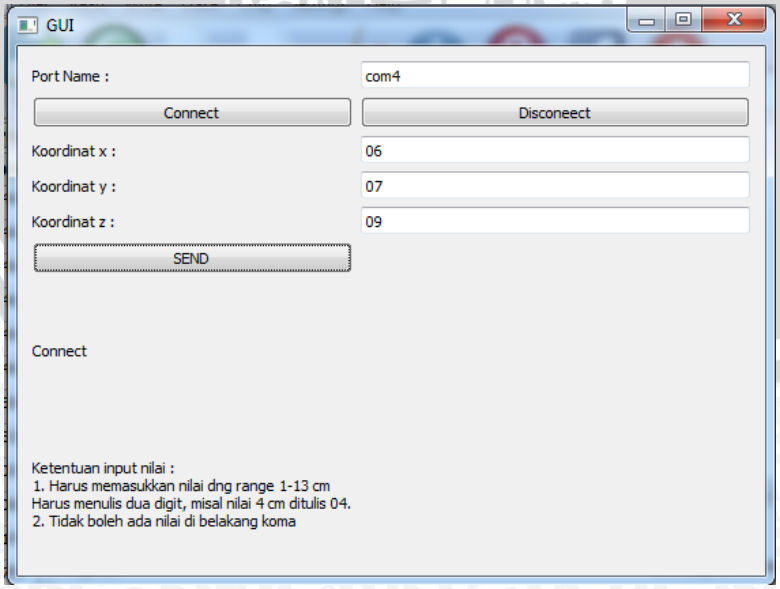

Gambar 5.2 Tampilan interface yang dibuat dari Qt

Pertama kali yang harus dilakukan adalah menyesuaikan nama port pada PC dengan *Port Name* pada *interface* seperti yang ditunjukkan dalam Gambar 5.3 lalu klik tombol CONNECT untuk menghubungkan dengan *Hardware* yang digunakan dalam hal ini adalah USB to TTL.



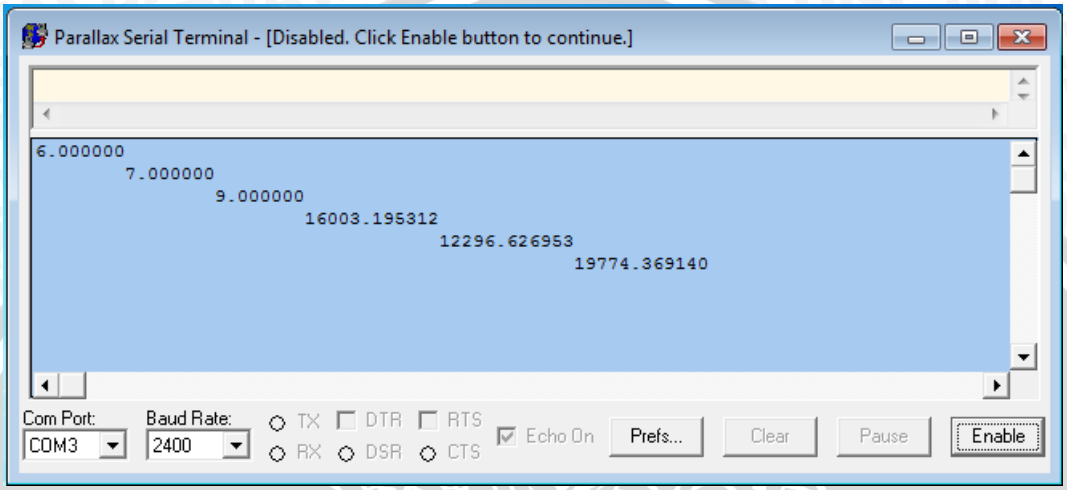

Gambar 5.3 Tampilan *interface* yang sudah tersambung dengan *Hardware*

Selanjutnya adalah menuliskan nilai koordinat tujuan dan jika sudah di klik tombol SEND seperti yang ditunjukkan dalam Gambar 5.4.




Gambar 5.4 Tampilan *interface* yang sudah diberi masukan dan dikirim

Gambar 5.5 menampilkan data yang diterima di PC lainnya dengan menggunakan aplikasi Parallax Serial Terminal. Paket data yang diterima ditunjukkan dalam gambar dimana juga perlu men-*setting* nilai baudrate dan *COM Port.*




Gambar 5.5 Tampilan data yang diterima di PC lainnya

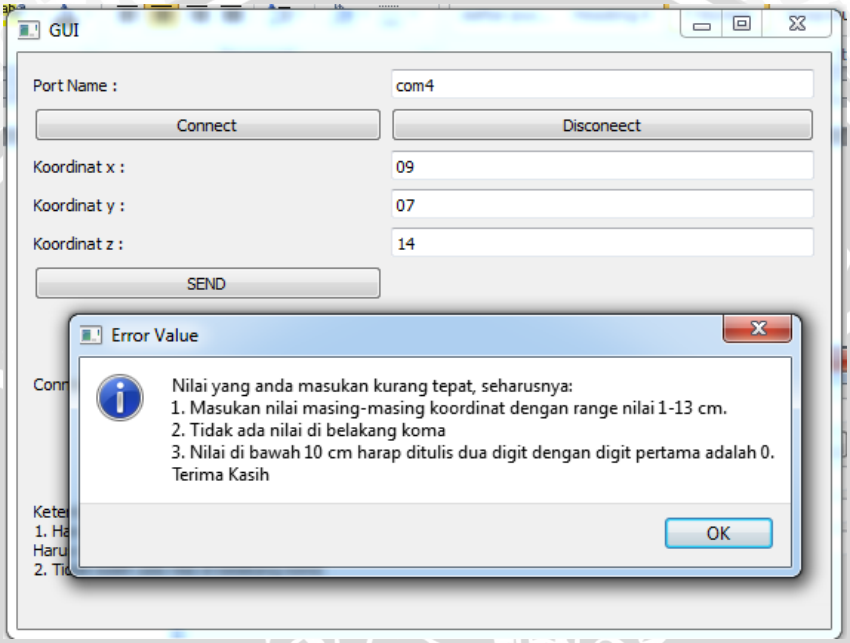
Pada perancangan ini *setting serial* yang digunakan antara Qt dengan Mikrokontroller haruslah sama antara lain *baudrate* sebesar 2400 dan frame datanya adalah 8 data bit, 1 stop bit, dan tidak ada bit paritas.

Bila nilai koordinat yang dikirim dari PC tidak sesuai yaitu bukan diantara rentang nilai 1-13 cm, ada nilai di belakang koma, dan/atau nilai di bawah 10 cm tidak ditulis 2 digit maka akan keluar tampilan seperti dalam Gambar 5.6.

Alasan nilai di bawah 10 cm tidak dapat diproses bila tidak ditulis 2 digit adalah karena program penerima di Mikrokontroller disetting demikian sebagai proses konversi dari tipe data char menjadi float seperti ditunjukkan dalam persamaan (4-9)

Pada pengujian kali ini dikirim nilai untuk koordinat tujuan (x,y,z) adalah (6, 7, 9) cm. Lalu dilihat pada Parallax Serial Terminal pada PC penerima apakah nilai tersebut dapat diterima oleh mikrokontroller atau tidak, dan hasilnya sukses diterima oleh Mikrokontroller. Nilai tersebut dapat dilihat di 3 baris pertama

dalam Gambar 5.5. Tiga baris di bawahnya adalah nilai TOP servo yang terkait dengan invers kinematik bila nilai koordinat tujuannya adalah (6,7,9) cm. Nilai tersebut didapatkan dari persamaan Invers Kinematik seperti dalam persamaan (4-1) hingga persamaan (4-6) yang sudah ditulis pada program seperti dalam persamaan (4-10)

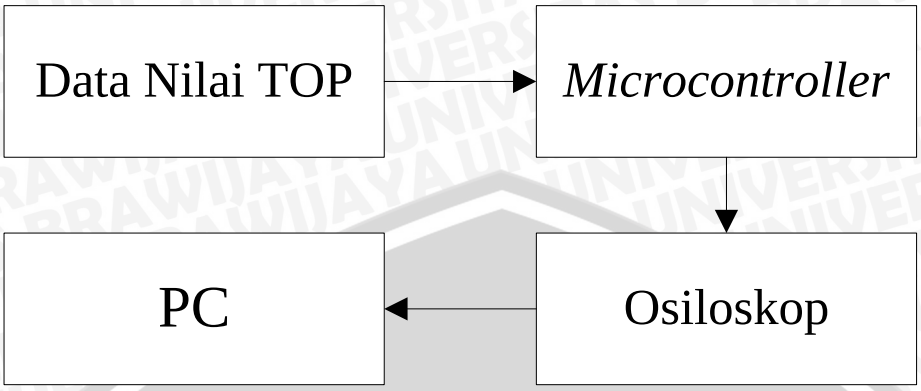



Gambar 5.6 Tampilan *interface* saat nilai yang dikirim tidak memenuhi persyaratan.

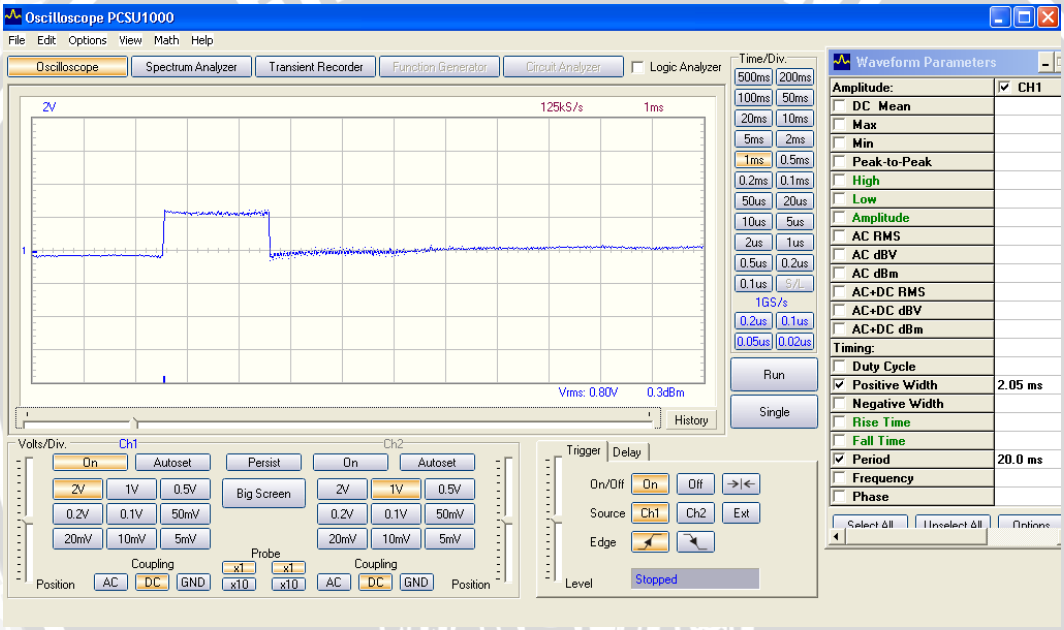
## 5.2 Pengujian Sinyal PWM

Pengujian ini memiliki tujuan untuk mengetahui pengaruh perubahan nilai TOP pada pulsa PWM yang menggerakkan motor DC servo. Skema pengujian PWM motor *servo* seperti dalam Gambar 5.3.

Proses pembangkitan sinyal PWM menggunakan algoritma kontrol *servo* jamak pada pin I/O. Nilai TOP dalam desimal akan dimasukkan ke dalam program dan dikeluarkan ke setiap pin *servo* pada *mainboard* untuk dilihat grafiknya pada osilodkop secara bergantian. Pin yang digunakan adalah PORTC. Sinyal PWM jamak ini dibutuhkan untuk mempercapat dan mempermudah proses pengujian sinyal PWM. Osiloskop yang digunakan merupakan Osiloskop Velleman PCLAB PCSU1000. Hasil pengujian terlihat dalam Gambar 5.4.




Gambar 5.7 Skema pengujian sinyal PWM




Gambar 5.8 Tampilan pengukuran sinyal PWM pada PCLAB

Osiloskop memiliki fitur untuk menampilkan *positive width* ($T_{ON}$) dan juga periode sinyal. *Probe* ditancapkan pada *channel* 1, dengan Volt/div sebesar 2 Volt/div dan time/div sebesar 1 ms/div. Sinyal PWM yang diuji berdasarkan nilai TOP dari 6634,52 hingga 27.647. Nilai-nilai tersebut dipilih karena mewakili karakteristik $T_{ON}$ yang biasa digunakan pada motor DC *servo*. Nilai $T_{ON}$ teori didapat dari persamaan (4-10). Dengan mengganti nilai frekuensi PWM pada *output compare* menjadi nilai $T_{ON}$ diperoleh persamaan:

$$\frac{1}{T_{ON}} = \frac{f_{clk\_I/O}}{N.(1+TOP)} \qquad (5\text{-}1)$$

sehingga dengan nilai $N$ = 1, $f_{clk\_I/O}$ = 11,0592 MHz, dengan nilai $T_{ON}$ yang kita inginkan sebagai nilai yang diketahui dan nilai TOP nilai yang dicari maka

$$TOP = (f_{clk\_I/O} - T_{ON}) - 1 \quad (5\text{-}2)$$

$$TOP = (11.059.200 - T_{ON}) - 1 \quad (5\text{-}3)$$

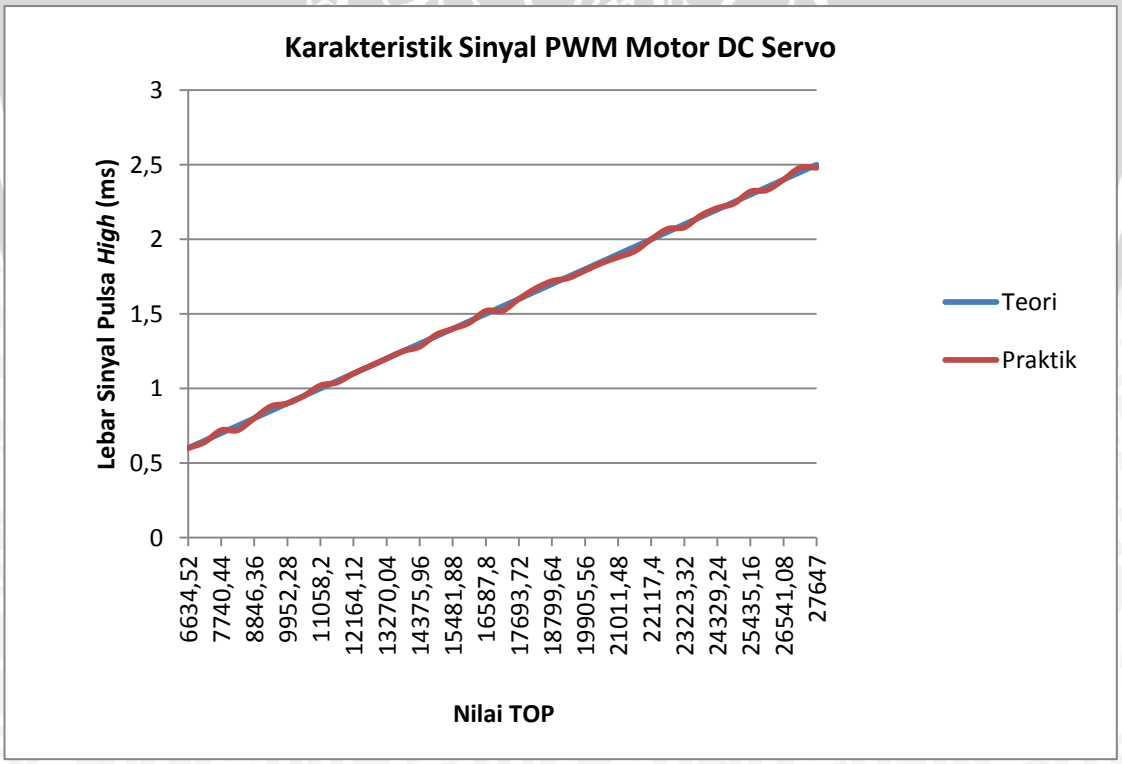
Nilai $T_{ON}$ yang diketahui tersebut sebagai nilai $T_{ON}$ teori. Nilai $T_{ON}$ secara praktik didapat dari pengujian di PCLAB.

Tabel 5.1 Data Hasil Pengujian Sinyal PWM

| TOP | $T_{ON}$ teori | $T_{ON}$ praktek | % error |
|---|---|---|---|
| 6634,52 | 0,60 | 0,60 | 0,00 |
| 7187,48 | 0,65 | 0,64 | 1,50 |
| 7740,44 | 0,70 | 0,72 | 2,80 |
| 8293,4 | 0,75 | 0,72 | 4,00 |
| 8846,36 | 0,80 | 0,8 | 0,00 |
| 9399,32 | 0,85 | 0,88 | 3,40 |
| 9952,28 | 0,90 | 0,9 | 0,00 |
| 10505,24 | 0,95 | 0,95 | 0,00 |
| 11.058,20 | 1,00 | 1,02 | 1,90 |
| 11.611,16 | 1,05 | 1,04 | 0,90 |
| 12.164,12 | 1,10 | 1,10 | 0,00 |
| 12.717,08 | 1,15 | 1,15 | 0,00 |
| 13.270,04 | 1,20 | 1,20 | 0,00 |
| 13.823 | 1,25 | 1,25 | 0,00 |
| 14.375,96 | 1,30 | 1,28 | 1,50 |
| 14.928,92 | 1,35 | 1,36 | 0,70 |
| 15.481,88 | 1,40 | 1,40 | 0,00 |
| 16.034,84 | 1,45 | 1,44 | 0,60 |
| 16.587,80 | 1,50 | 1,52 | 1,30 |
| 17.140,76 | 1,55 | 1,52 | 1,90 |
| 17.693,72 | 1,60 | 1,60 | 0,00 |
| 18.246,68 | 1,65 | 1,67 | 1,10 |
| 18.799,64 | 1,70 | 1,72 | 1,10 |
| 19.352,60 | 1,75 | 1,74 | 0,50 |
| 19.905,56 | 1,80 | 1,79 | 0,50 |
| 20.458,52 | 1,85 | 1,84 | 0,50 |
| 21.011,48 | 1,90 | 1,88 | 1,00 |
| 21.564,44 | 1,95 | 1,92 | 1,50 |
| 22.117,40 | 2,00 | 2,00 | 0,00 |

| | | | |
|---|---|---|---|
| 22.670,36 | 2,05 | 2,07 | 0,90 |
| 23.223.32 | 2,10 | 2,08 | 0,90 |
| 23.776,28 | 2,15 | 2,16 | 0,40 |
| 24.329,24 | 2,20 | 2,21 | 0,40 |
| 24.882,20 | 2,25 | 2,24 | 0,40 |
| 25.436,16 | 2,30 | 2,32 | 0,80 |
| 25.988,12 | 2,35 | 2,33 | 0,80 |
| 26.541,08 | 2,40 | 2,40 | 0,00 |
| 27.094,04 | 2,45 | 2,48 | 1,20 |
| 27.647 | 2,50 | 2,48 | 0,80 |
| | | TOTAL | 0,85 |

Data hasil pengujian ditampilkan dalam Tabel 5.1. Dari tabel tersebut dapat disimpulkan bahwa sinyal PWM yang dibangkitkan *Microcontroller* tergolong akurat. Periode sinyal *high* yang dimasukkan ke program dalam nilai TOP dapat dikeluarkan lagi ke setiap pin dengan *error* yang sangat kecil, yaitu dengan *error* tertinggi 4% dan *error* terendah 0%.

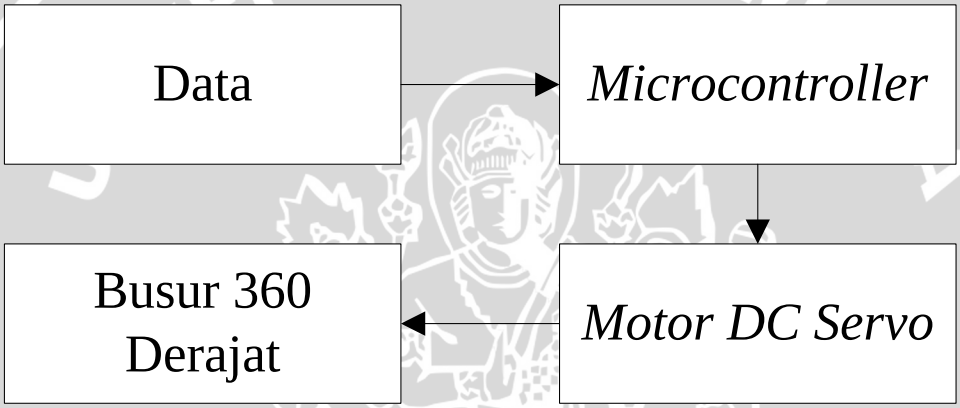



Gambar 5.9 Grafik pengujian sinyal PWM

Data dari Tabel 5.1 dapat diubah menjadi grafik dalam Gambar 5.9 dimana keluaran sinyal PWM berupa garis linear dengan data praktik dan teori yang berimpit.

### 5.3 Pengujian Motor DC *Servo*

Pengujian dilakukan untuk mengetahui keluaran sudut motor DC *servo* dari sinyal PWM yang diberikan. Sinyal PWM dikeluarkan oleh mikrokontroller seperti pada pengujian sebelumnya, tetapi pada pengujian ini, sinyal tidak dikeluarkan ke osiloskop melainkan langsung ke motor DC *servo.*




Gambar 5.10 Skema Pengujian Motor DC *Servo*

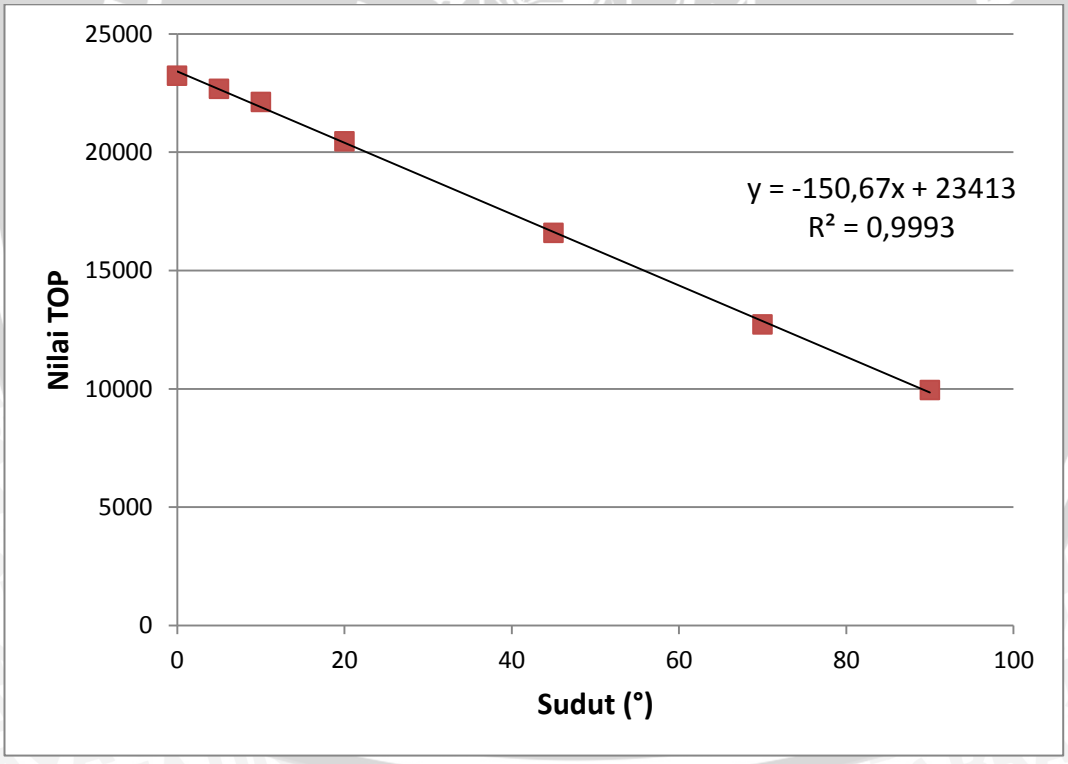
Untuk pengujian kali ini motor DC *servo* dipasangi busur lingkaran penuh dan di bagian *horn* dipasang plat tipis sebagai jarum penunjuknya. Motor DC servo yang diuji adalah Hitec HS-5685 dan Corona DS238HV. Algoritma yang digunakan sama persis dengan pengujian sinyal PWM, hanya saja keluaran sistem yang diamati adalah motor DC *servo.* Untuk menghitung teori sudut motor DC *servo* dapat dilakukan dengan menggunakan persamaan linear.

Servo yang diuji ada tiga buah yaitu sebuah servo Hitec untuk posisi servo Lengan Atas Putar Kanan dan dua buah servo Corona untuk posisi servo Ketiak Kanan dan posisi servo Siku Kanan. Hanya ketiga servo tersebut yang dilihat hubungan antara nilai TOP yang diberikan dengan posisi sudut yang dihasilkan karena ketiga servo itulah yang berhubungan langsung dengan persamaan invers

kinematik untuk mencapai koordinat yang diinginkan. Amplitudo pulsa yang dibutuhkan untuk menggerakkan servo antara 3-5 Vpp.

Tabel 5.2 Data pengujian posisi nilai TOP yang diberikan pada Servo Hitec terhadap sudut yang dihasilkan pada posisi Servo Lengan Atas Putar Kanan

| TOP | $T_{ON}$ (ms) | Sudut (°) |
|---|---|---|
| 9952,28 | 0,9 | 90 |
| 12717,08 | 1,15 | 70 |
| 16587,8 | 1,5 | 45 |
| 20458,52 | 1,85 | 20 |
| 22117,4 | 2,0 | 10 |
| 22670,36 | 2,05 | 5 |
| 23223,32 | 2,1 | 0 |




Gambar 5.11Grafik nilai TOP terhadap posisi sudut Servo Lengan Atas Putar Kanan

Pada Tabel 5.2 terlihat bahwa nilai *y* adalah TOP dan *x* adalah nilai sudut sehingga Persamaan Linier untuk konversi sudut Servo Lengan Atas Putar Kanan dengan terlebih dahulu melakukan regresi adalah

$$y = -150{,}67.x + 23413 \quad (5\text{-}4)$$

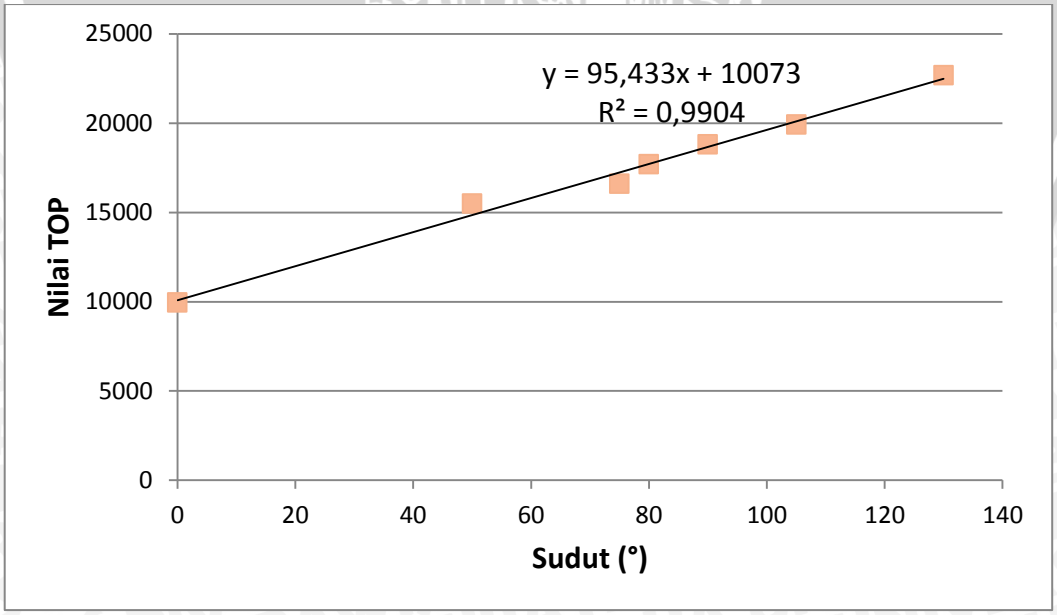
dimana

y = TOP

x = $\theta_1$ (°)

Tabel 5.3 Data pengujian posisi nilai TOP yang diberikan pada Servo Corona terhadap sudut yang dihasilkan pada posisi Servo Ketiak Kanan

| TOP | $T_{ON}$ (ms) | Sudut (°) |
|---|---|---|
| 6634,52 | 0,60 | Tidak bergerak di posisi manapun |
| 7740,44 | 0,70 | Tidak bergerak di posisi manapun |
| 8846,36 | 0,80 | Tidak bergerak di posisi manapun |
| 9952,28 | 0,90 | 0 |
| 15481,88 | 1,40 | 50 |
| 16587,8 | 1,50 | 75 |
| 17693,72 | 1,60 | 80 |
| 18799,64 | 1,70 | 90 |
| 19905,56 | 1,80 | 105 |
| 22670,36 | 2,05 | 130 |




Gambar 5.12 Grafik nilai TOP terhadap posisi sudut Servo Ketiak Kanan

Mendapatkan persamaan konversi dari sudut ke nilai TOP pada posisi Servo Ketiak Kanan dengan dilakukan regresi terlebih dahulu sehingga Persamaan Linier untuk konversi sudut Servo Ketiak Kanan adalah

$$y = 95{,}433.x + 10073 \quad (5\text{-}5)$$

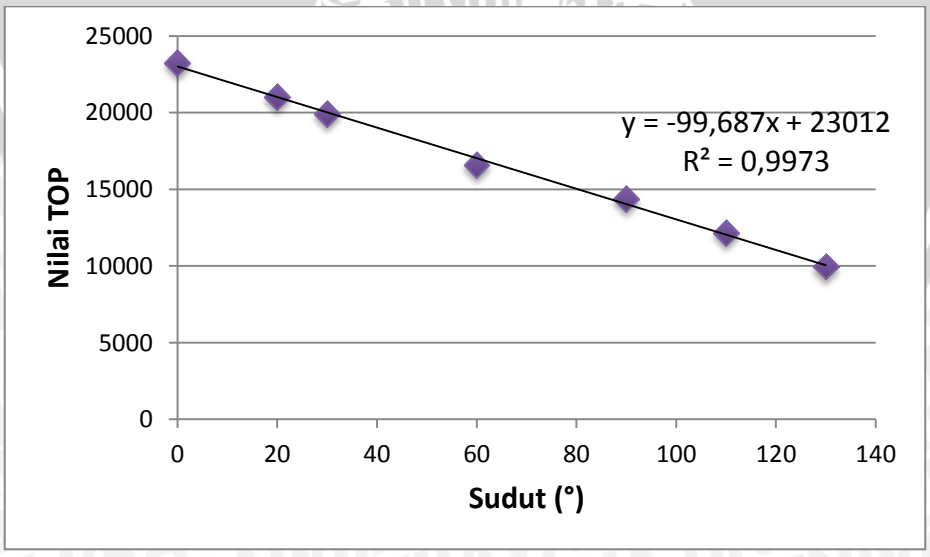
dimana

y = TOP

x = $\theta_2$ (°)

Tabel 5.4 Data pengujian posisi nilai TOP yang diberikan pada Servo Corona terhadap sudut yang dihasilkan pada posisi Servo Siku Kanan

| TOP | $T_{ON}$ (ms) | Sudut (°) |
|---|---|---|
| 9952,28 | 0,9 | 130 |
| 12164,12 | 1,10 | 110 |
| 14375,96 | 1,30 | 90 |
| 16587,8 | 1,5 | 60 |
| 19905,56 | 1,80 | 30 |
| 21011,48 | 1,90 | 20 |
| 23223,32 | 2,10 | 0 |




Gambar 5.13Grafik nilai TOP terhadap posisi sudut Servo Siku Kanan

Mendapatkan persamaan konversi dari sudut ke nilai TOP pada posisi Servo Ketiak Kanan dilakukan regresi sehingga Persamaan Linier untuk konversi sudut Servo Siku Kanan adalah

$$y = -99{,}687.x + 23012 \quad (5\text{-}6)$$

dimana

y = TOP

x = $\theta_3$ (°).

## 5.4 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem ini dilakukan dengan menyambungkan semua *hardware* yang dibuat berdasarkan blok diagram dan memasukkan program berupa *software* yang bekerja untuk menggerakkan *hardware* yang telah dibuat. Sistem bekerja dengan baik jika dapat berjalan sesuai *flowchart* yang telah direncanakan.

Pada pengujian ini, sistem yang akan diuji dirangkai mengikuti diagram blok dalam Gambar 4.1. Algoritma yang digunakan disusun berdasarkan diagram alir dalam Gambar 4.10 pada Mikrokontroller dan diagram alir dalam Gambar 4.15 pada antarmuka Qt.

Sesuai dengan perancangan awal, percobaan ini memiliki masukan berupa koordinat tujuan yang diterima oleh mikrokontroller dari PC, kemudian dibaca dan dihitung sudutnya pada program di mikrokontroler untuk menggerakkan servo yang sesuai agar dapat bergerak menuju koordinat yang dituju dan bergerak sesuai gerakan tari yang direncanakan. Pengujian ini dititikberatkan pada akurasi sudut servo pada motor DC *servo* dengan perhitungan persamaan yang ada.

Data koordinat yang dikirim terlebih dahulu diukur apakah nilai $T_{ON}$-nya dapat menggerakkan servo atau tidak bila sudah sesuai maka akan dicoba langsung ke servonya.

Pengujian kali ini dicoba dengan nilai yang tertera pada pengujian serial dimana nilai yang dikirimkan adalah sama dengan pada pengujian komunikasi serial seperti terlihat dalam Gambar 5.5 yaitu (6,7,9) cm. Ketiga nilai TOP tersebut dapat diuji langsung karena nilainya masuk dalam rentang nilai TOP yang sesuai dengan masing-masing servo.

Bila data yang tidak sesuai dikirim langsung tanpa dicek nilainya akan menyebabkan servo tidak bergerak karena nilai sudut yang didapat dari Persamaan Invers Kinematik tidak berada di jangkauan *end-affector*. Walaupun nilai koordinat yang bisa dijangkau pada 1-13 cm namun ada beberapa titik yang tidak bisa dijangkau dalam *range* ini.

Nilai error koordinat tujuan dengan di kenyataan sangat terkait dengan nilai $R^2$ masing-masing servo. Semakin nilai $R^2$ mendekati 1 maka akan semakin kecil nilai error-nya.

# BAB VI
# PENUTUP

## 6.1 Kesimpulan

Berdasarkan pengujian dan analisis yang telah dilakukan dapat ditarik kesimpulan sebagai berikut.

1. Sinyal PWM mampu dibangkitkan Mikorokontroller dengan rentang antara 0,6-2,5 ms dengan error yang paling kecil sebesar 0,00% dan paling besar 4%.
2. Komunikasi serial antara PC dengan Mikrokontroller dapat dilakukan.
3. Hasil pengujian ketiga servo yang berhubungan dengan persamaan invers kinematik memiliki hubungan nilai $T_{ON}$ dengan sudut yang berbeda. Servo Lengan Atas Putar Kanan pada posisi 0° membutuhkan $T_{ON}$ sebesar 2,1 ms dan pada posisi 90° membutuhkan $T_{ON}$ sebesar 0,9 ms. Servo Ketiak Kanan pada posisi 0° membutuhkan $T_{ON}$ sebesar 0,9 ms dan pada posisi 90° membutuhkan $T_{ON}$ sebesar 1,7 ms. Servo Siku Kanan pada posisi 0° membutuhkan $T_{ON}$ sebesar 2,1 ms dan pada posisi 90° membutuhkan $T_{ON}$ sebesar 1,3 ms. Dari ketiga rentang $T_{ON}$ pada ketiga servo yang berbeda persamaan linier konversi dari perubahan nilai sudut terhadap nilai TOP-nya juga berbeda.
4. Perancangan aplikasi untuk menggerakkan motor DC Servo pada lengan robot kanan sesuai koordinat yang telah ditentukan dari PC dapat dilakukan sampai servo dapat bergerak sampai menuju koordinat yang telah ditentukan. Kesalahan yang terjadi adalah karena nilai karakterisasi masing-masing servo yang masih memiliki error sehingga berpengaruh terhadap keseleruhan gerakan menuju koordinat tujuan.

## 6.2 Saran

Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah.

1. Penelitian ini menggunakan persamaan invers kinematik untuk Servo Lengan Kanan sehingga penelitian selanjutnya dapat dikembangkan dengan bagian Lengan Kiri dan atau dengan kuadran yang lebih luas.
2. Bisa membuat *prototype* yang lebih baik dari segi mekanik dan kepresisian.

# DAFTAR PUSTAKA


Alfauzi, Tanshuda. 2014. *Perancangan Exoskeleton Motion Capture System Sebagai Panduan Gerakan Tari Pada Robot Humanoid KRSI.* Skripsi tidak diterbitkan. Malang: Fakultas Teknik Universitas Brawijaya.

Atmel. 2007. *8-bit AVR with 16K Bytes In-System Programmable Flash ATMega32, ATMega32L.* San Jose: Atmel.

GWS. 2013. *http://gwsus.com*. Grand Wing Sytem USA Inc. Diakses tanggal 22 Desember 2013.

HITEC. 2013. *http://hitecrcd.com/products/servos/premium-digital-servos*. Hitec RCD USA, Inc. Diakses tanggal 10 Maret 2014.

Shahbazi, Hamed dkk. 2012. *Modeling Of Mesencephalic Locomotor Region For Nao Humanoid Robot*. Iran: Emerald Group Publishing Limited.

Sigit, Riyanto. 2007. *Robotika, Sensor, dan Aktuator.* Yogyakarta: Graha ilmu.

Spong, Mark W. 2004. *Robot Dynamics and Control.* John Wiley & Sons. New York.

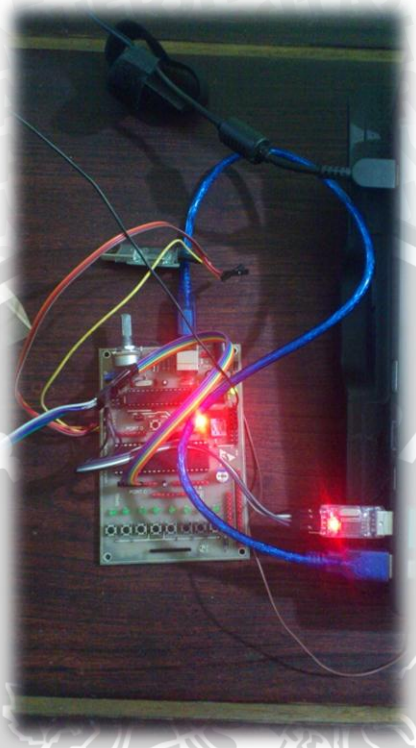
Winoto, Ardi. 2008. *Mikrokontroller AVR Atmega9/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR.* Bandung: Informatika.

# LAMPIRAN I

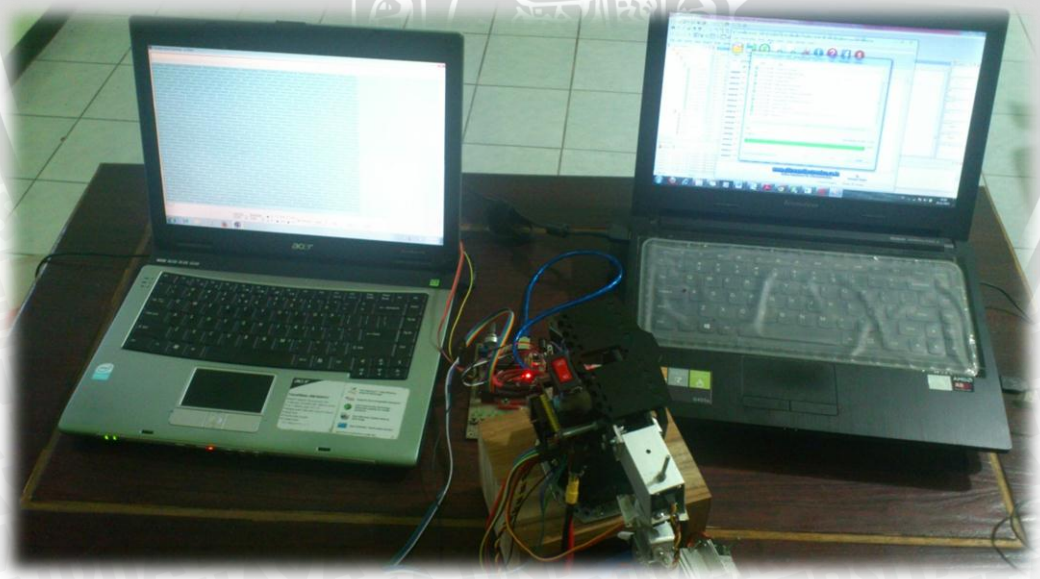
**Dokumentasi Alat**

Gambar Hardware Keseluruhan

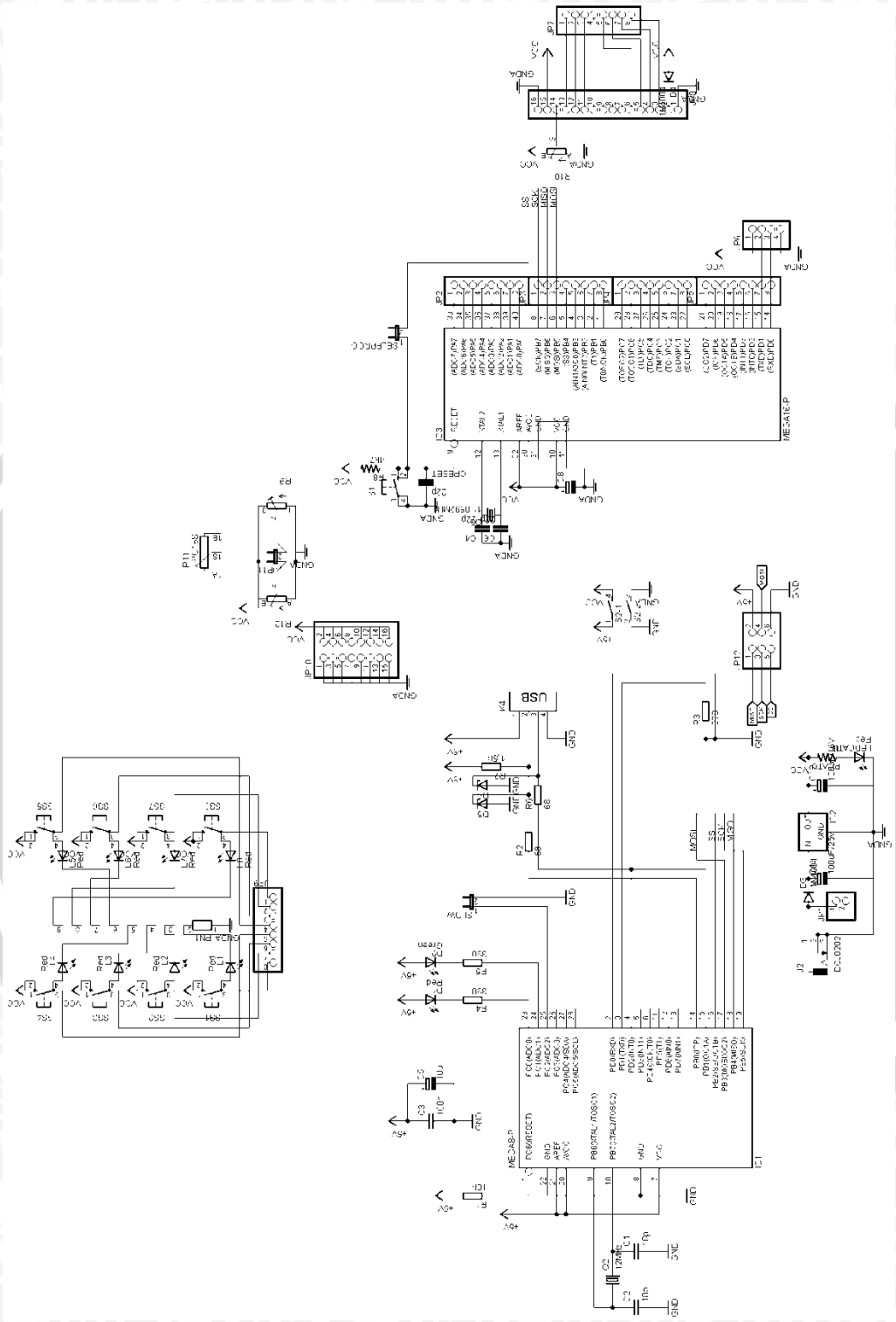
Gambar Lengan robot Humanoid tampak belakang



Gambar USB to TTL dan Mikrokontroller ATMega32



Gambar Pengujian Keseluruhan Sistem



Rangkaian *Mainboard Microcontroller*

# LAMPIRAN II

## Listing Program pada Mikrokontroller

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Advanced
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Penentuan Sudut Lengan Robot Humanoid Berdasarkan
Koordinat Yang Dikirim Dari PC Menggunakan User Interface Yang
Dibuat Dari Qt
Version : 1.01
Date    : 03/10/2014
Author  : Adiyatma
Company : TEUB
Comments: Program Utama


Chip type               : ATmega32
Program type            : Application
AVR Core Clock frequency: 11,059200 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 512
*****************************************************/
//=======================HEADER==============================//
#include <mega32.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
```

```
//============================================================//

//============Variabel yang digunakan=======================//

float xc, yc, zc;
float tetha1;
float tetha2a, tetha2b, tetha2;
float tetha3, sintetha3;
float r,s;
float D, D1, D2, D3;
float ab, bb;
float alfa1, alfa2;
float servo1a, servo1b, servo1;
float servo2a, servo2b, servo2;
float servo3a, servo3b, servo3;
float pi= 3.141592654;

unsigned char x=0;
float servoA[5];

int state;
int statehand;

//============================================================//
//=====================================USART=================
==================//
#ifndef RXB8
#define RXB8 1
#endif
```

```
#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4     //awal
//#define RXEN 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7   //awal
//#define RXCIE 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
```

```
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 96
char rx_buffer[RX_BUFFER_SIZE];//,data;

#if RX_BUFFER_SIZE <= 256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

float konversi(char a, char b)
{
 float x = (a-48) * 10;
 float y = b-48;
 float z;
 return z = x+y;
}
hitung_servo ()
{
 alfa1=6;
 alfa2=7;

 tetha1= atan (yc/xc)*180/pi;

 s= alfa1+alfa2-zc;
```

```
r= sqrt(xc*xc+yc*yc);
//D= ((r*r+s*s-alfa1*alfa1-alfa2*alfa2)/(2*alfa1*alfa2))
D1=(r*r+s*s-alfa1*alfa1-alfa2*alfa2);
D2=(2*alfa1*alfa2);
D= (D1/D2);
tetha3= acos (D)*180/pi;

sintetha3= sqrt (1-D*D);
tetha2a= atan (r/s)*180/pi;
ab= alfa2*sintetha3;
bb= alfa1+alfa2*D;
tetha2b= atan (ab/bb)*180/pi;
tetha2= tetha2a - tetha2b;

        //servo1=-150,67*tetha1+23413
        servo1a=tetha1*150,67;
        servo1b=-servo1a;
        servo1=servo1b+23413;

           //servo2=95,433*tetha2+10073
           servo2a=tetha2*95,433;
           servo2=servo2a+10073;

               //servo3=-99,687*tetha3+23012
               servo3a=tetha3*99,687;
               servo3b=-servo3a;
               servo3=servo3b+23012;
}
// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
```

```
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
 rx_buffer[rx_wr_index++]=data;

#if RX_BUFFER_SIZE == 256
   // special case for receiver buffer size=256
   if (++rx_counter == 0)
      {
#else
   if (rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
#endif
      rx_buffer_overflow=1;
      }
   }
    if (data==117)
   {
    xc=konversi(rx_buffer[0],rx_buffer[1]);
    yc=konversi(rx_buffer[2],rx_buffer[3]);
    zc=konversi(rx_buffer[4],rx_buffer[5]);
    hitung_servo();
    rx_wr_index=0;
    state=1;
    }
```

```
    else
    {
     state=0;
    }
}


#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index++];
#if RX_BUFFER_SIZE != 256
if (rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#endif
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif


// Timer1 input capture interrupt service routine
interrupt [TIM1_CAPT] void timer1_capt_isr(void)
```

```
{
// Place your code here
if(++x>5) x=0;
PORTC=0x00;

OCR1A=servoA[x];

switch(x-1)
    {
case 0 : PORTC=0x01;
         break;
case 1 : PORTC=0x02;
         break;
case 2 : PORTC=0x04;
         break;
case 3 : PORTC=0x08;
         break;
case 4 : PORTC=0x10;
         break;
    }

}

// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{

  switch(x-1)
    {
    case 0 : PORTC.0=0;
```

```
            break;
    case 1 : PORTC.1=0;
            break;
    case 2 : PORTC.2=0;
            break;
    case 3 : PORTC.3=0;
            break;
    case 4 : PORTC.4=0;
            break;
    }

}

void main(void)
{
PORTC=0x00;
DDRC=0xFF;

PORTD=0x00;
DDRD=0b01000000;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 11,0592 MHz
// Mode: Fast PWM top=ICR1
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: On
```

```
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x02;
TCCR1B=0x19;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x8F;
ICR1L=0xFF;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x30;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 2400
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x06;
UBRRH=0x01;
UBRRL=0x1F;

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

```
state=0;

while (1)
      {

       //keluaran
       switch(state)
       {
              case 0:
//posisi awal
              servoA[0]=23223,32;
              servoA[1]=9952,28;
              servoA[2]=23223.32;
              servoA[3]=23223.32;
              servoA[4]=13823.3;

              break;

              case 1:
//gerakan invers
              servoA[0]=servo1;
              servoA[1]=servo2;
              servoA[2]=servo3;

              break;

              default:
              printf("error");
              break;
```

```
        }
    }
}
```

# LAMPIRAN III

**Listing Program pada Qt**

**Header Serial**

```
#ifndef SERIALDATA_H
#define SERIALDATA_H

#include <QObject>
#include <QSerialPort>
#include <QtSerialPort/QtSerialPort>
#include <QMessageBox>
#include <QByteArray>

class serialdata : public QObject
{
    Q_OBJECT
public:
    explicit serialdata(QObject *parent = 0);

signals:
    void isConnect(QString);
    void valueError();
public slots:
    void setPortOpen();
    void setPortName(QString namePort);
    void setArraySudut1(QString sudutA);
    void setArraySudut2(QString sudutB);
    void setArraySudut3(QString sudutC);
    void sendAllData();
    void sendAwalData();
    void disconnectPort();
    bool checkNilai1(QString a);
    bool checkNilai2(QString b);
    bool checkNilai3(QString c);
    void checkNilaiFeedback();

private:
     QSerialPort *control_Port_1;
     QString portName,sudutA1,sudutB2,sudutC3;
     const char *sudut1;
     const char *sudut2;
     const char *sudut3;
     QByteArray suduta, sudutb, sudutc;
     char bufferData[96];

};

#endif // SERIALDATA_H
```

**Header Pesan Salah**

```
#ifndef ERRORMESSAGE_H
#define ERRORMESSAGE_H

#include <QDialog>
#include <QMessageBox>

class ErrorMessage : public QDialog
{
    Q_OBJECT
public:
    explicit ErrorMessage(QWidget *parent = 0);

signals:

public slots:
    void dialogErrorValue();

};

#endif // ERRORMESSAGE_H
```

**Program Utama**

```
#include <QApplication>
#include <QGridLayout>
#include <QPushButton>
#include <QLineEdit>
#include <QLabel>
#include "errormessage.h"
#include "serialdata.h"
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    ErrorMessage *error = new ErrorMessage();
    QWidget *widget = new QWidget();
    QGridLayout *layout = new QGridLayout();
    QPushButton *buttonConnect = new QPushButton("Connect");
    QLineEdit *linePort = new QLineEdit();
    QLineEdit *lineSudutA = new QLineEdit();
    QLineEdit *lineSudutB = new QLineEdit();
    QLineEdit *lineSudutC = new QLineEdit();
    QPushButton *buttonSendData = new QPushButton("SEND");
    QPushButton *buttonDisconnect = new QPushButton("Disconeect");
    QLabel *labelPortName = new QLabel("Port Name : ");
    QLabel *labelConnect = new QLabel();
    QLabel *labelKetSudutA = new QLabel("Koordinat x : ");
    QLabel *labelKetSudutB = new QLabel("Koordinat y : ");
    QLabel *labelKetSudutC = new QLabel("Koordinat z : ");
    QLabel *labelKeterangan = new QLabel ("Ketentuan input nilai
:\n1. Harus memasukkan nilai dng range 1-13 cm \nHarus menulis dua
digit, misal nilai 4 cm ditulis 04.\n2. Tidak boleh ada nilai di
belakang koma");
    serialdata *myData = new serialdata();
    //labelConnect->setText("Not Connect");
    layout->addWidget(labelPortName,0,0,1,1);
    layout->addWidget(linePort,0,1,1,1);
    layout->addWidget(buttonConnect,1,0,1,1);
    layout->addWidget(labelConnect,6,0,1,1);
    layout->addWidget(labelKetSudutA,2,0,1,1);
    layout->addWidget(lineSudutA,2,1,1,1);
    layout->addWidget(labelKetSudutB,3,0,1,1);
    layout->addWidget(lineSudutB,3,1,1,1);
    layout->addWidget(labelKetSudutC,4,0,1,1);
    layout->addWidget(lineSudutC,4,1,1,1);
    layout->addWidget(buttonSendData,5,0,1,1);
    layout->addWidget(buttonDisconnect,1,1,1,1);
    layout->addWidget(labelKeterangan,7,0,1,1);

    widget->setLayout(layout);
    widget->show();


QObject::connect(myData,SIGNAL(isConnect(QString)),labelConnect,SL
OT(setText(QString)));

QObject::connect(buttonConnect,SIGNAL(clicked()),myData,SLOT(setPo
rtOpen()));
```

```
QObject::connect(linePort,SIGNAL(textEdited(QString)),myData,SLOT(
setPortName(QString)));

QObject::connect(buttonDisconnect,SIGNAL(clicked()),myData,SLOT(di
sconnectPort()));

QObject::connect(lineSudutA,SIGNAL(textEdited(QString)),myData,SLO
T(setArraySudut1(QString)));

QObject::connect(lineSudutB,SIGNAL(textEdited(QString)),myData,SLO
T(setArraySudut2(QString)));

QObject::connect(lineSudutC,SIGNAL(textEdited(QString)),myData,SLO
T(setArraySudut3(QString)));

QObject::connect(buttonSendData,SIGNAL(clicked()),myData,SLOT(send
AllData()));

QObject::connect(myData,SIGNAL(valueError()),error,SLOT(dialogErro
rValue()));
    return a.exec();
}
```

**Program Serial**

```
#include "serialdata.h"
#include <QDebug>

serialdata::serialdata(QObject *parent) :
    QObject(parent)
{
    control_Port_1 = new QSerialPort(this);
}
void serialdata::setPortName(QString namePort)
{
    portName = namePort;

}
void serialdata::setPortOpen()
{
    control_Port_1->setPortName(portName);
    control_Port_1->setBaudRate(QSerialPort::Baud2400);
    control_Port_1->setDataBits(QSerialPort::Data8);
    control_Port_1->setFlowControl(QSerialPort::NoFlowControl);
    control_Port_1->setParity(QSerialPort::NoParity);
    control_Port_1->setStopBits(QSerialPort::OneStop);
    control_Port_1->open(QIODevice::ReadWrite);
    if(control_Port_1->isOpen())
    {
        emit isConnect("Connect");
    }
    else{
        emit isConnect("Not Connect");
    }
}
void serialdata::setArraySudut1(QString sudutA)
{
    suduta = sudutA.toLatin1();
    sudutA1 = sudutA;
}
void serialdata::setArraySudut2(QString sudutB)
{

    sudutb = sudutB.toLatin1();
    sudutB2 = sudutB;
}
void serialdata::setArraySudut3(QString sudutC)
{
    sudutc = sudutC.toLatin1();
    sudutC3 = sudutC;
}
void serialdata::sendAllData()
{
    if(this->checkNilai1(sudutA1)==true && this-
>checkNilai2(sudutB2)==true && this->checkNilai3(sudutC3)==true){
    control_Port_1->write(suduta);
    control_Port_1->write(sudutb);
    control_Port_1->write(sudutc);
    control_Port_1->write("u",1);
```

```
        //control_Port_1->write(EnterAscii);
        qDebug()<<"Supposed
Connected"<<sudutA1<<"\t"<<sudutB2<<"\t"<<sudutC3;}
        else{
            emit valueError();
        }
}
void serialdata::disconnectPort()
{
    control_Port_1->close();
    emit isConnect("Disconnect");
}
bool serialdata::checkNilai1(QString a)
{
    float x = a.toFloat();
    if(x >= 1.00 && x<= 13.00)
    {
        return true;
    }
    else
    {
        return false;
    }
}
bool serialdata::checkNilai2(QString b)
{
    float x = b.toFloat();

    if(x >= 1.00 && x <= 13.00)
    {
        return true;
    }
    else
    {
        return false;
    }
}
bool serialdata::checkNilai3(QString c)
{
    float x = c.toFloat();

    if(x >= 1.00 && x<= 13.00)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

**Program pesan error**

```
#include "errormessage.h"

ErrorMessage::ErrorMessage(QWidget *parent) :
    QDialog(parent)
{
}
void ErrorMessage::dialogErrorValue()
{
    QMessageBox::information(this,"Error Value", "Nilai yang anda
masukan kurang tepat, seharusnya: \n1. Masukan nilai masing-masing
koordinat dengan range nilai 1-13 cm. \n2. Tidak ada nilai di
belakang koma \n3. Nilai di bawah 10 cm harap ditulis dua digit
dengan digit pertama adalah 0. \nTerima Kasih");

}
```

# LAMPIRAN IV

**Datasheet**

# LAMPIRAN V

**Invers Kinematik**