UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM BILATERAL TELEOPERATION MULTI DOF DENGAN SERIAL SERVO

SKRIPSI

DIKO HARNELDO
0806365665

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
DESEMBER 2010

UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM BILATERAL TELEOPERATION MULTI DOF DENGAN SERIAL SERVO

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

DIKO HARNELDO
0806365665

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
DESEMBER 2010

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Diko Harneldo**
NPM	**: 0806365665**
Tanda Tangan	**:**
Tanggal	**: 16 Desember 2010**

PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Diko Harneldo
NPM	: 0806365665
Program Studi	: Teknik Elektro
Judul Skripsi	: Rancang Bangun Sistem Bilateral Teleoperation Multi DOF Dengan Serial Servo

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

DEWAN PENGUJI

Pembimbing	: Dr. Abdul Muis S.T., M.Eng	( )
Penguji	: Ir. Wahidin Wahab M.Sc, Ph.D	( )
Penguji	: Dr. Ir. Feri Yusivar M.Eng	( )

Ditetapkan di : Depok

Hari / Tanggal : 4 Januari 2011

UCAPAN TERIMAKASIH

Puji syukur saya panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penyusunan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Departemen Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, baik dari masa perkuliahan sampai pada penyusunan skripsi ini sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Untuk karena itu, saya mengucapkan terima kasih kepada :

(1) Dr. Abdul Muis, ST, M.Eng, selaku dosen pembimbing yang telah menyediakan waktu, tenaga dan pikiran didalam mengarahkan saya dalam penyusunan skripsi ini;

(2) orangtua saya yang telah memberikan bantuan dukungan material maupun moril; dan

(3) sahabat dan orang-orang terdekat yang telah banyak membantu penulis dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 16 Desember 2010

Diko Harneldo

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama	: Diko Harneldo
NPM	: 0806365665
Program Studi	: Teknik Elektro
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis karya	: Skripsi

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

Rancang Bangun Sistem Bilateral Teleoperation Multi DOF

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelolanya dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 16 Desember 2010
Yang menyatakan

Diko Harneldo

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Nama : Diko Harneldo
Program studi : Teknik Elektro
Judul : Rancang Bangun Sistem Bilateral Teleoperation Multi DOF Dengan Serial Servo

Sistem bilateral teleoperation menggunakan dua jenis informasi yaitu teleoperation dan telepresence. Dua informasi ini bergerak dalam dua arah membentuk sistem *closed loop*.
Pengendalian bilateral teleoperation banyak digunakan pada lingkungan-lingkungan dengan tingkat resiko tinggi. Pengendalian ini harus dilakukan secara *real time*, dan diperlukan sebuah proses *feedback* yang digunakan untuk mengetahui apakah yang dikerjakan sesuai dengan yang diperintahkan. Kemudian diperlukan pula suatu sistem dimana operator juga dapat merasakan sensasi sentuhan manipulator *slave* saat menyentuh objek sehingga operator seolah dapat berinteraksi langsung dengan objek.
Dalam skripsi ini dibahas mengenai perancangan sistem bilateral teleoperation multi DOF menggunakan motor serial servo dan prosesor Vortex86DX.
Evaluasi kinerja sistem dilakukan dengan memperhatikan respon *slave* terhadap perubahan posisi sudut yang diberikan pada *master*, serta respon sistem ketika terdapat objek yang menahan pergerakan manipulator *slave*.
Pada kondisi normal, selisih sudut terbesar antara master dan slave adalah 7,624° dan selisih sudut rata-rata adalah 0,103°. Pada kondisi tertahan, perbedaan sudut terbesar antara master dan slave mencapai 11,143°.

Kata kunci:
Bilateral teleoperation, RB-100, RB-110, Dynamixel AX-12+, Kondo KRS 2552 HV

ABSTRACT

Nama : Diko Harneldo
Program studi : Teknik Elektro
Judul : Design of Bilateral Teleoperation System Multi DOF with Serial Servo

Bilateral teleoperation system use two kind of information which is teleoperation and telepresence. These information are exchanged forming a closed loop system.
Bilateral teleoperation mainly used at high risk environment. It has to have real time capability and has a reliable feedback mechanism. Also, there is a need for a system to provide the sense of touch to operator.
This final project explains about bilateral teleoperation design for multi DOF system using serial servo and Vortex86DX processor.
The system evaluated by monitoring slave responses against master at normal condition and disturbed condition.
At normal condition, maximum angular position difference between master and slave is 7.624° with 0.103° average. At disturbed condition, maximum angular position difference between master and slave is 11.143°.

Keywords:
Bilateral teleoperation, RB-100, RB-110, Dynamixel AX-12+, Kondo KRS 2552 HV

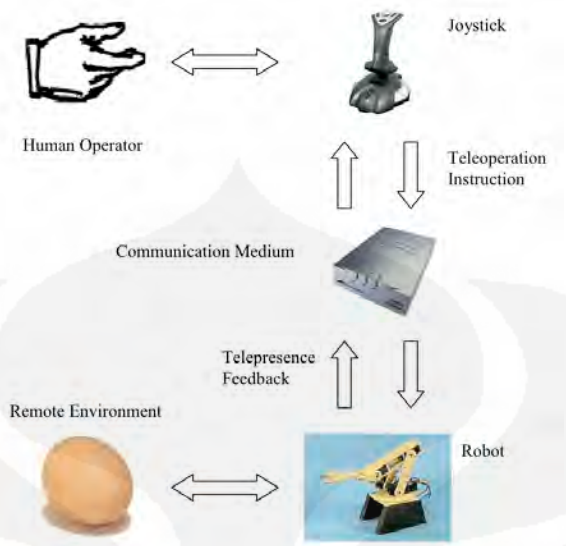
BAB 1
PENDAHULUAN

1.1 Latar Belakang

Telerobotics merupakan salah satu bidang dari robotika yang membahas tentang pengendalian robot dari jarak jauh. Dari sudut pandang teknis, *telerobotics* mencakup dua hal yaitu *teleoperation* dan *telepresence*. *Teleoperation* dapat diartikan sebagai kemampuan operator untuk memanipulasi objek dari jauh. Sedangkan *telepresence* dapat dideskripsikan sebagai umpan balik dari teleoperation dimana keadaan dari lingkungan pengendalian robot diumpan balikkan kepada operator dalam berbagai bentuk seperti visual, audio, atau sensasi sentuhan (*force feedback*). *Telepresence* dalam bentuk sensasi sentuhan dikenal dengan nama *haptic*.

Contoh dari *haptic* misalnya, tangan manusia dapat memegang sebuah telur tanpa memecahkan telur tersebut. Hal ini dapat terjadi karena tangan manusia dapat memberikan tekanan yang cukup kepada telur agar telur tersebut tidak pecah atau terlepas dari genggaman tangan. Tekanan yang diberikan oleh tangan manusia didasarkan oleh kemampuan untuk merasakan sentuhan.

Sebuah sistem yang melibatkan *teleoperation* dan *haptic* biasanya disebut *bilateral teleoperation* karena sistem ini menggunakan dua jenis informasi yang bergerak dalam dua arah membentuk sebuah sistem *closed loop*. Pada skripsi ini kata *teleoperation* dimaksudkan untuk mendeskripsikan *bilateral teleoperation*. Sisi operator pada sistem *teleoperation* disebut dengan master, sedangkan sisi robot dan lingkungannya disebut dengan slave. Salah satu contoh konfigurasi bilateral teleoperation yang umum dapat dilihat pada Gambar 1.1



Gambar 1.1. Contoh Bilateral Teleoperation

Teleoperation banyak digunakan pada keadaan-keadaan dimana manusia tidak mungkin melakukan tugasnya. Sebagai contoh; pada pembuangan sampah nuklir dan operasi penjinakan bom. Perancangan *teleoperation* yang dilakukan pada skripsi ini terkait dengan proyek pembuatan robot *search and rescue* untuk kondisi pasca gempa.

Teleoperation pertama kali dikembangkan pada pertengahan tahun 1940 oleh Raymond Goertz. Saat ini teleoperation digunakan dalam berbagai bidang antara lain untuk robot luar angkasa seperti Mars Rover dari NASA, untuk melakukan *telesurgery* atau pembedahan jarak jauh, dan untuk menangani bahan-bahan radio aktif.

1.2 Perumusan Masalah

Perancangan sistem *teleoperation* di lingkungan Universitas Indonesia telah beberapa kali dilakukan dengan menggunakan konfigurasi sistem yang berbeda-beda dan terbatas pada 1 sendi atau 1 *degree of freedom* (DOF). Metode umum yang digunakan adalah menggunakan kombinasi mikrokontroler, *analog to digital converter* (ADC), PC, dan motor servo. Konfigurasi ini belum dapat mengakomodasi kebutuhan sistem *teleoperation* yang layak, misalnya sistem *teleoperation* multi DOF dengan *force feedback*.

Permasalahan yang dibahas pada skripsi ini adalah bagaimana merancang sebuah sistem *teleoperation* yang dapat memenuhi kebutuhan multi DOF, dapat dikembangkan lebih lanjut, serta memiliki kemampuan komputasi yang tinggi.

1.3 Tujuan Penulisan

Tujuan penulisan skripsi ini adalah merancang sebuah sistem *multi DOF bilateral teleoperation* yang lebih mudah untuk dikembangkan dan lebih dapat mengakomodasi kebutuhan komputasi jika sistem *teleoperation* ini ingin dikembangkan menjadi sistem yang lebih kompleks di masa yang akan datang dengan harga yang relatif lebih murah dibandingkan dengan harga sistem teleoperation lain yang beredar dipasaran.

1.4 Pembatasan Masalah

Pembahasan dalam skripsi ini meliputi penggunaan board RB-100 dan RB-110 sebagai *development board*, pengendalian serial servo Dynamixel AX-12+ dan Kondo KRS 2552 HV, integrasi *development board* dan serial servo menjadi sebuah sistem *bilateral teleoperation*. Skripsi ini tidak membahas penambahan pengendali eksternal.

1.5 Metode Penelitian

Metode penelitian yang digunakan dalam penyusunan skripsi ini meliputi:

1. Studi Literatur mengenai sistem yang sudah pernah dikembangkan di Universitas Indonesia
2. Pedekatan diskusi dengan pembimbing skripsi
3. Perakitan perangkat keras dan perancangan perangkat lunak
4. Pengujian sistem terintegrasi

1.6 Sistematika Penulisan

Skripsi ini akan ditulis dalam 5 bagian yaitu:

Bab Pertama, Pendahuluan, memuat latar belakang, perumusan masalah, tujuan, pembatasan masalah, metodologi penelitian, sistematika penelitian, dan sistematika penulisan. **Bab Kedua**, Landasan Teori, menggambarkan landasan pengetahuan mengenai komunikasi serial, serial servo, mikrokontroler, dan hal-hal lain yang digunakan pada skripsi ini. **Bab Ketiga,** Perancangan,

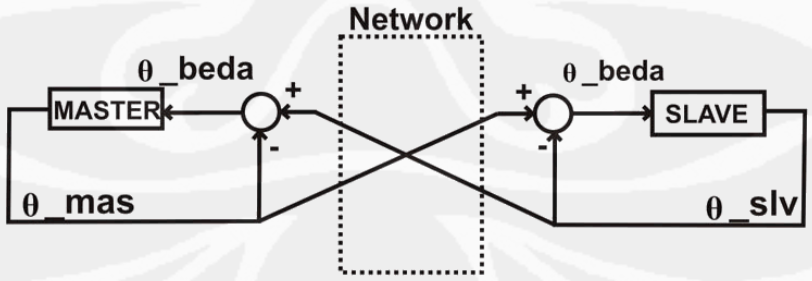
menggambarkan langkah pengerjaan skripsi ini mulai dari perancangan sistem hingga pengaturan konfigurasi yang digunakan. **Bab Keempat**, Analisa, berisi analisa dan pembahasan mengenai evaluasi kerja dari sistem. **Bab Kelima**, Penutup, bab ini merangkum pembahasan skripsi ini.

BAB 2
LANDASAN TEORI

2.1 Prinsip Bilateral Teleoperation

Bilateral teleoperation adalah konfigurasi dua manipulator yang saling mempengaruhi satu sama lain dan terkoneksi melalui sebuah jaringan. Ketika operator menggerakkan manipulator *master*, manipulator *slave* akan bergerak mengikuti gerakan *master*. Operator juga dapat merasakan sensasi sentuhan manipulator *slave* saat menyentuh objek. Idealnya, sebuah sistem bilateral memungkinkan operator dapat merasakan seolah berinteraksi langsung dengan objek.

Pada pengendalian bilateral di skripsi ini akan dikendalikan posisi (sudut) *slave* atas perintah *master*. Serta membuat sisi *master* dapat merasakan apa yang dialami oleh sisi *slave*. Untuk itu kedua manipulator harus dibuat saling mempengaruhi satu sama lain. Dimana sudut *master* memberikan acuan pada manipulator *slave* dan sudut *slave* memberikan acuan pada manipulator *master*. Sudut yang akan masuk ke masing-masing manipulator adalah *error* atau beda dari sudut *slave* dan sudut *master*. Kedua manipulator terkoneksi melalui sebuah jaringan untuk saling mengirimkan referensi sudut. Komunikasi antar kedua manipulator dilakukan dengan menggunakan komunikasi data serial. Hal tersebut dapat diilustrasikan dengan gambar blok diagram berikut.



Gambar 2.1. Blok Diagram Umum Bilateral Teleoperation

2.2 Motor Serial Servo

Motor serial servo merupakan motor servo standar yang ditambahkan unit mikrokontroler agar dapat melakukan fungsi-fungsi tambahan dan dapat

dikendalikan melalui komunikasi serial. Jika pada motor servo standar pengendalian dilakukan menggunakan *pulse width modulation* (*PWM*), maka pada motor serial servo pengendalian dilakukan melalui perintah-perintah dalam bentuk komunikasi serial. Selain dapat dikendalikan melalui komunikasi serial, serial servo juga dapat memberikan umpan balik berupa posisi, temperatur, beban, tegangan, dan lain-lain tergantung pada jenis serial servo yang digunakan. Penggunaan serial servo dapat mengurangi beban kerja prosesor atau kontroler utama karena tugas pembacaan posisi dan lain-lain dibebankan kepada mikrokontroler yang terletak di dalam serial servo. Pada skripsi ini digunakan 2 jenis serial servo yaitu Dynamixel AX-12+ dan Kondo KRS 2552 HV.

2.2.1 Dynamixel AX-12+

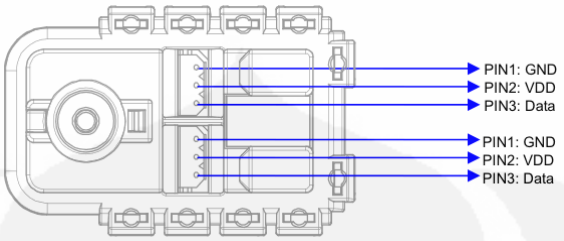
Dynamixel AX-12+ merupakan serial servo produksi Robotis yang mengkhususkan diri pada bidang robot *humanoid*.

Berikut adalah spesifikasi utama dari serial servo ini:

Tabel 2.1. Spesifikasi Dynamixel AX-12+

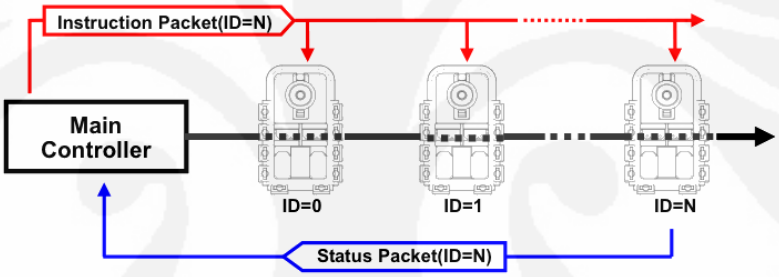
Parameter	Nilai
Tegangan Operasi	7 – 10 Volt
Holding Torque	12 – 16.5 kgf.cm
Resolusi	0.35°/step (1024 step)
Sudut Operasi Efektif	300°
Sudut Operasi Maksimum	360°
Arus Maksimum	900 mA
Protokol	Half Duplex, 8 bit data, 1 stop bit, no parity
Baudrate	7343bps – 1Mbps
Level Sinyal	TTL
Maksimum ID	254
Feedback	Posisi, Temperatur, Load, Tegangan

Berikut adalah konfigurasi pin dari Dynamixel AX-12+



Gambar 2.2. Pin Dynamixel AX-12+

Salah satu kelebihan dari serial servo adalah adanya koneksi *multi-drop* yang dapat menghubungkan beberapa serial servo ke satu serial komunikasi sehingga sangat membantu mengurangi penggunaan port input/output pada mikrokontroler secara signifikan.



Gambar 2.3. Koneksi Multi-Drop

Serial servo ini dikendalikan oleh paket-paket instruksi sesuai dengan protokol yang digunakan oleh Dynamixel AX-12+. Bentuk umum dari paket instruksinya adalah sebagai berikut:

OXFF	0XFF	ID	LENGTH	INSTRUCTION	PARAMETER1	PARAMETER N	CHECK SUM

Gambar 2.4. Instruksi Dynamixel AX-12+

Seluruh data protokol Dynamixel di atas dalam bentuk hexadesimal. Dua buah **0xFF** merupakan header dari paket komunikasi, mengindikasikan dimulainya pengiriman paket. **ID** merupakan nomor identifikasi unik yang terdapat di setiap motor yang terkoneksi dalam rantai *multi-drop*. Terdapat 254 ID yang dapat digunakan dari 0x00 sampai 0xFD. ID 0xFE digunakan untuk *broadcast* paket dimana isi dari paket akan diterima dan dilaksanakan oleh setiap serial servo yang

terkoneksi. **LENGTH** menunjukkan panjang dari paket dimana nilai LENGTH adalah jumlah dari parameter + 2. **INSTRUCTION** merupakan instruksi yang akan dieksekusi oleh Dynamixel. Jenis-jenis instruksi akan dijelaskan pada bagian selanjutnya. **PARAMETER** adalah informasi tambahan yang diperlukan oleh eksekusi INSTRUCTION. **CHECKSUM** merupakan metode penghitungan untuk pemeriksaan *error*. Nilai *checksum* dihitung dengan cara CheckSum = ~ (ID + Length + Instruction + Parameter1 + … + Parameter N). Tanda ~ merepesentasikan operasi logika negasi.

Instruksi yang tersedia pada Dynamixel AX-12+ dapat dilihat pada tabel di bawah ini

Tabel 2.2. Tabel Instruksi Dynamixel AX-12+

Instruction	Function	Value	Num of Parmeter
PING	No action. Used for obtaining status packet	0x01	0
READ DATA	Reading value in the control tabe	0x02	2
WRITE DATA	Writing value to the control table	0x03	2 ~
REG WRITE	Similar to WRITE DATA, but stays in standby mode until ACTION instruction is given	0x04	2 ~
ACTION	Triggers the action registered by REG WRITE instruction	0x05	0
RESET	Reset to factory default	0x06	0
SYNC WRITE	Used for controlling many actuators at the same time	0x07	4~

Setelah menerima paket instruksi, Dynamixel AX-12+ akan mengirimkan umpan balik berupa tanggapan dari paket instruksi yang diterima. Umpan balik disebut dengan paket status. Jeda waktu antara penerimaan paket instruksi sampai pengiriman paket status dapat diatur dengan cara mengubah nilai pada *Control Table* dengan *address* 0x05. Jeda waktu secara *default* adalah 5µs. Struktur dari paket status dapat dijelaskan oleh gambar berikut

Gambar 2.5. Paket Status Dynamixel AX-12+

Paket status memiliki struktur yang hampir sama seperti paket instruksi. Paket status memberikan informasi *error* jika terjadi kesalahan dalam pengiriman instruksi, kesalahan fungsi instruksi, atau kesalahan fisik dari servo. Informasi kesalahan akan muncul dalam bentuk byte data yang tertulis dalam **ERROR**. Berikut adalah tabel penjelasan kemungkinan pesan kesalahan yang muncul.

Tabel 2.3. Tabel Error pada Paket Status

Bit	Nama	Keterangan
Bit 7	Selalu 0	-
Bit 6	Instruction Error	Bernilai 1 jika servo menerima instruksi yang tidak terdefinisi
Bit 5	Overload Error	Bernilai 1 jika nilai beban melebihi torsi maksimum
Bit 4	CheckSum Error	Bernilai 1 jika checksum pada paket instruksi salah
Bit 3	Range Error	Bernilai 1 jika parameter paket instruksi melebihi range yang terdefinisi
Bit 2	Overheating Error	Bernilai 1 jika suhu servo melebihi temperatur operasi yang ditetapkan
Bit 1	Angle Limit Error	Bernilai 1 jika servo digerakkan melebihi batas yang ditetapkan
Bit 0	Input Voltage Error	Bernilai 1 jika tegangan input melebihi batas tegangan operasi yang ditetapkan

Serial servo Dynamixel dioperasikan dengan cara menuliskan dan membaca register-register yang terletak pada EEPROM dan RAM serial servo tersebut. Berikut adalah tabel register-register pada Dynamixel AX-12+.

Tabel 2.4. Control Table Dynamixel AX-12+

Control Table	Address	Item	Access	Initial Value
EEPROM Area	0(0X00)	Model Number(L)	RD	12(0x0C)
	1(0X01)	Model Number(H)	RD	0(0x00)
	2(0X02)	Version of Firmware	RD	?
	3(0X03)	ID	RD,WR	1(0x01)
	4(0X04)	Baud Rate	RD,WR	1(0x01)
	5(0X05)	Return Delay Time	RD,WR	250(0xFA)
	6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
	7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
	8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
	9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
	10(0x0A)	(Reserved)	-	0(0x00)
	11(0X0B)	the Highest Limit Temperature	RD,WR	85(0x55)
	12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
	13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
	14(0X0E)	Max Torque(L)	RD,WR	255(0XFF)
	15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
	16(0X10)	Status Return Level	RD,WR	2(0x02)
	17(0X11)	Alarm LED	RD,WR	4(0x04)
	18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
	19(0X13)	(Reserved)	RD,WR	0(0x00)
	20(0X14)	Down Calibration(L)	RD	?
	21(0X15)	Down Calibration(H)	RD	?
	22(0X16)	Up Calibration(L)	RD	?
	23(0X17)	Up Calibration(H)	RD	?
RAM Area	24(0X18)	Torque Enable	RD,WR	0(0x00)
	25(0X19)	LED	RD,WR	0(0x00)
	26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
	27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
	28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
	29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
	30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
	31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
	32(0X20)	Moving Speed(L)	RD,WR	0
	33(0X21)	Moving Speed(H)	RD,WR	0
	34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
	35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
	36(0X24)	Present Position(L)	RD	?
	37(0X25)	Present Position(H)	RD	?
	38(0X26)	Present Speed(L)	RD	?
	39(0X27)	Present Speed(H)	RD	?
	40(0X28)	Present Load(L)	RD	?
	41(0X29)	Present Load(H)	RD	?
	42(0X2A)	Present Voltage	RD	?
	43(0X2B)	Present Temperature	RD	?
	44(0X2C)	Registered Instruction	RD,WR	0(0x00)
	45(0X2D)	(Reserved)	-	0(0x00)
	46[0x2E)	Moving	RD	0(0x00)
	47[0x2F)	Lock	RD,WR	0(0x00)
	48[0x30)	Punch(L)	RD,WR	32(0x20)
	49[0x31)	Punch(H)	RD,WR	0(0x00)

Jenis instruksi yang paling sering digunakan pada skripsi ini adalah *Write Position* dan *Read Position*. Contoh penggunaan paket instruksi dan tanggapan paket status adalah sebagai berikut:

Menggerakkan servo dengan ID 0 ke posisi 180°

Goal Position Address: 0x1E

Data: 0x200

Paket Instruksi: FF FF 00 05 03 1E 00 02 D7

Paket Status: FF FF 00 02 00 FD

2.2.2 Kondo KRS 2552 HV

Kondo KRS 2552 HV merupakan salah satu serial servo produksi Kondo, perusahaan yang mengkhususkan diri pada pengembangan robot *humanoid*.

Gambar 2.6. Kondo KRS 2552 HV

Serial servo ini beroperasi pada tegangan 9 – 12V dengan sudut putaran efektif 270° dan sudut putaran maksimum 360°. Menyediakan koneksi *multi-drop* dengan total servo terkoneksi sampai 31 servo. Kelebihan serial servo ini dibandingkan dengan Dynamixel AX-12+ terletak pada penggunaan *metal gear* sehingga lebih tahan lama dan tidak cepat rusak.

Serial servo ini menggunakan protokol khusus dari Kondo yang disebut dengan ICS 3.0. Protokol ini mendukung 3 jenis pilihan *baudrate* yaitu 115200, 625000, dan 1,25Mbps namun untuk jenis motor ini hanya bisa menggunakan 115200. Pengaturan komunikasi yang digunakan oleh serial servo ini adalah 1 *start* bit, 8 bit data, 1 stop bit, *no flow control*, dan *even parity*.

Protokol yang digunakan oleh serial servo ini relatif lebih mudah dan lebih singkat bila dibandingkan dengan protokol pada Dynamixel AX-12+ akan tetapi hal ini juga memberikan dampak negatif yaitu tidak menjamin integritas data, tidak memiliki pemeriksaan *error*, dan tidak memberikan pesan *error* bila terjadi kesalahan.

Bentuk umum dari paket instruksi serial servo ini adalah sebagai berikut:

TX	1 (CMD)	2 (SC)	3 - N(Data)
	Command + ID	SubCommand	Data

Gambar 2.7. Paket Instruksi Kondo KRS 2552 HV

Hanya terdapat 4 jenis **Command** yang digunakan pada protokol ICS yaitu:

Tabel 2.5. Command pada Protokol ICS

100xxxxx	Pengaturan posisi
101xxxxx	Pembacaan parameter (jenis parameter tergantung sub command)
110xxxxx	Penulisan parameter (jenis parameter tergantung sub command)
111xxxxx	Pengaturan ID

xxxxx pada tabel diatas merupakan bentuk biner ID dari servo yang akan diproses. Misalnya untuk servo dengan ID = 12 maka nilai xxxxx akan menjadi 01100. ID maksimum yang dapat digunakan adalah 11111(biner) atau 1F(hexa) atau 31(desimal).

SubCommand merupakan bagian protokol yang menentukan fungsi yang akan dijalankan oleh servo. Pada ICS 3.0 terdapat 4 jenis SubCommand.

Tabel 2.6. Jenis SubCommand

Parameter	Nilai	Keterangan
EEPROM	0x00	Akses ke EEPROM
STRC	0x01	Manipulasi streching pulsa
SPD	0x02	Kontrol kecepatan motor
CUR	0x03	Kontrol arus pada motor

Data merupakan parameter yang akan dibaca atau ditulis ke serial servo.

Setelah menerima paket instruksi, serial servo akan memberikan tanggapan yang disebut paket status. Bentuk paket status dari protokol ICS adalah sebagai berikut:

RX	1	2	3	4	5	6
	Loopback Command Sent			R_CMD	SC	Data

Gambar 2.8. Paket Status Kondo KRS 2552 HV

Loopback Command Sent berisi nilai yang sama dengan nilai yg dikirimkan pada saat TX. *Loopback command sent* memastikan bahwa data yang dikirim telah sampai ke servo dengan benar.

R_CMD dan **SC** adalah *command* dan *sub command* yang sama dengan *command* dan *sub command* pada saat pengiriman.

Data merupakan nilai terakhir yang dimiliki oleh servo sebelum nilai tersebut diubah oleh data yang baru.

Khusus untuk perintah pengaturan posisi, tidak membutuhkan *sub command*. Sudut operasi dapat diatur pada jangkauan 0 – 16383 desimal atau 0 – 3FFF hexa.

TX	1 (CMD)	2	3
	100xxxxx + ID	POS_H	POS_L

Gambar 2.9. Pengaturan Posisi

POS_H dan **POS_L** adalah tempat untuk menuliskan sudut operasi dengan aturan sebagai berikut:

MSB	POS_H / POS_L						LSB
7	6	5	4	3	2	1	0
0	x	x	x	x	x	x	x

Gambar 2.10. Format POS_H dan POS_L

Pada kenyataannya tidak seluruh jangkauan sudut operasi dapat digunakan. Secara *default* sudut operasi dibatasi pada jangkauan 3500 – 11500 desimal. Berdasarkan nilai ini kita dapat menghitung nilai tengah atau sudut 90° yaitu 7500 desimal. Nilai 7500 desimal jika dikonversi menjadi bilangan biner akan menjadi 1110101001100. Bilangan ini kemudian didistribusikan ke dalam POS_H dan POS_L dengan syarat MSB (*most significant bit*) masing-masing variabel harus bernilai 0. Dengan ketentuan tersebut maka nilai POS_L adalah 7 bit dari kanan ditambah bit 0 pada MSB menjadi 01001100 biner atau 4C hexa. Sedangkan nilai POS_H adalah sisa bit ditambah bit 0 pada MSB menjadi 00111010 biner atau 3A hexa.

Penggunaan protokol ICS akan lebih mudah dipahami melalui contoh. Berikut adalah contoh untuk menggerakkan serial servo ID 1 ke posisi 90° atau 7500 desimal.

Command : 100xxxxx (pengaturan posisi)
ID : xxx00001 (ID 1)
Data : 0011 1010 0100 1100 (3A4C)

TX	1 (CMD)	2	3
	81	3A	4C

Gambar 2.11. Contoh Pengaturan Posisi

Perintah di atas kemudian akan ditanggapi oleh serial servo dengan mengirimkan paket status dalam bentuk berikut

RX	1	2	3	4	5	6
	81	3A	4C	81	0D	AC

Gambar 2.12. Contoh Paket Status

3 byte pertama dari paket status merupakan paket yang persis sama dengan paket yg dikirim. 1 byte berikutnya merupakan *command* yang sama dengan *command* yang dikirim. 2 byte terakhir merupakan posisi servo sebelum digerakkan ke posisi 3A4C. Dalam contoh ini posisi terakhir servo sebelum digerakkan adalah 0DAC.

Jika nilai 0000 diberikan pada POS_H dan POS_L, maka servo akan memasuki *free run mode* dimana servo tidak akan memberikan torsi dan mengunci posisi terakhir.

2.3 USB to TTL Converter

Serial servo yang digunakan pada skripsi ini menggunakan komunikasi serial pada level tegangan TTL (0 – 5V). USB to TTL *converter* digunakan ntuk melakukan pengaturan awal pada motor seperti pengaturan ID dan *baudrate*. USB to TTL *converter* yang digunakan pada skripsi ini berbasis chip FTDI 232BM yang mempunyai *baudrate* maksimum sebesar 3Mbps untuk komunikasi serial TTL.

Gambar 2.13. USB to TTL Converter

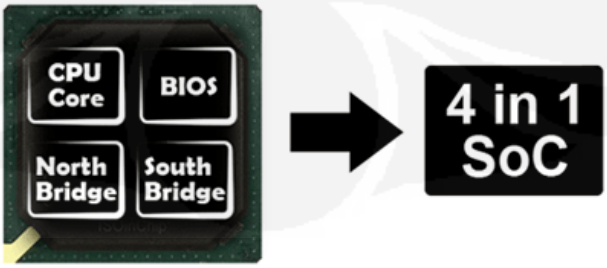
Pemilihan USB to TTL *converter* berbasis chip ini dimaksudkan agar dapat langsung berkomunikasi dengan serial servo Dynamixel AX-12+ yang menggunakan baudrate 1Mbps dimana ini merupakan baudrate yang tidak umum digunakan dalam komunikasi serial.

2.4 Development Board

Development board yang digunakan pada skripsi ini merupakan dua buah *board* dari DMP Electronics yaitu RB-100 dan RB-110. Kedua *board* dapat beroperasi seperti sebuah komputer biasa. Kompatibel dengan berbagai sistem operasi seperti Windows dan Linux.

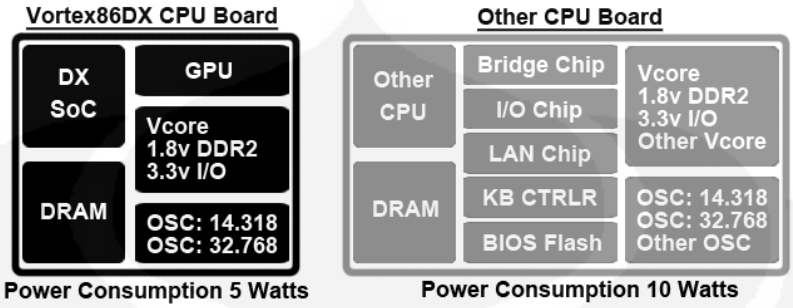
Gambar 2.14. Development Board RB-100

Kedua board ini menggunakan Vortex86DX, sebuah *Single on Chip* (SoC) yang berbasiskan prosesor 32 bit dari keluarga intel x86. SoC yang digunakan beroperasi pada kecepatan 1GHz dan memiliki DRAM (*Dynamic Random Access Memory*) sebesar 256MB. Penggunaan SoC diyakini dapat mengurangi kompleksitas rangkaian dan konsumsi energi. Vortex86DX mengintegrasikan 4 buah chip yang umumnya terpisah yaitu CPU, *NorthBridge*, *SouthBridge*, dan BIOS ke dalam sebuah chip BGA.



Gambar 2.15. Vortex86DX Single on Chip

Penggunaan Vortex86DX pada *development board* diyakini dapat mengurangi konsumsi daya karena penggunaannya dapat mengurangi penggunaan IC lain secara signifikan.



Gambar 2.16. Perbandingan CPU Board

RB-100 dan RB-110 memiliki banyak kelebihan dan memang dikhususkan untuk aplikasi robotika. Fitur-fitur andalan dari kedua board dapat dilihat pada tabel dibawah ini:

Tabel 2.7. Spesifikasi RB-100

Model	RB-100
Processor	DMP Vortex86DX
BIOS	AMI BIOS
Memory	256MB DDR2
ADC	Analog Device AD-7918 10bit
I/O Interface	Micro SD slot and USB 2.0 port
Connectors	PWM x 24
	RS-232 x 1
	USB x 1
	RS-485 x1
	Half Duplex TTL Serial x 1
	SPI & I2C x1
	ADC x 8
	Full Duplex TTL Serial x 1
	LAN x 1
	Mic In x 1
	Line Out x 1

	JTAG x 1
	Mini PCI x 1
Power Input	6 – 24V
Compatible OS	DOS, Windows 98, Windows XP
	Windows Embedded CE 6.0
	Windows XP Embedded, Linux

Terdapat sedikit perbedaan antara RB-100 dan RB-110. Pada RB-110 hanya terdapat 16 port PWM. Akan tetapi kekurangan ini ditutupi dengan bertambahnya 2 buah port komunikasi. RB-110 menggunakan *chip* FTDI 2232 untuk menghasilkan port komunikasi serial TTL dengan *baudrate* mencapai 12Mbps. Selain itu, RB-110 juga menggunakan sistem *dual power supply* untuk memisahkan supply ke board dan supply ke motor pada port PWM. Hal ini sangat baik untuk menghindari kerusakan yang dapat disebabkan oleh arus balik dari motor, selain itu juga dapat menjaga stabilitas asupan power untuk *board* itu sendiri.

Kedua board menyediakan *library* C++ yang dapat digunakan untuk membantu pengembangan software. Dengan adanya library ini, pengembang tidak lagi perlu bersentuhan langsung dengan register-register *low level* pada Vortex86DX.

2.5 Sistem Operasi

Sistem operasi yang digunakan pada skripsi ini adalah linux. Penulis menggunakan salah satu distro linux yang disebut ttylinux. Ttylinux adalah salah satu jenis linux yang cocok digunakan untuk *embedded system* karena sistemnya tidak terlalu kompleks dan tidak membutuhkan kapasitas RAM dan *hard disk* yang besar. Selain itu ttylinux juga tersedia untuk berbagai arsitektur prosesor seperti i486, i686, x86_64, ARM, dan PowerPC. Terdapat 2 versi ttylinux yaitu versi 9 dan versi 12 dimana versi 9 digunakan untuk arsitektur i486 sedangkan versi 12 digunakan untuk arsitektur i686. Untuk dapat menggunakan ttylinux versi 9, dibutuhkan *hardware* minimal prosesor 486SX, 24MB RAM, dan 12MB partisi *hard disk.*

BAB 3
PERANCANGAN

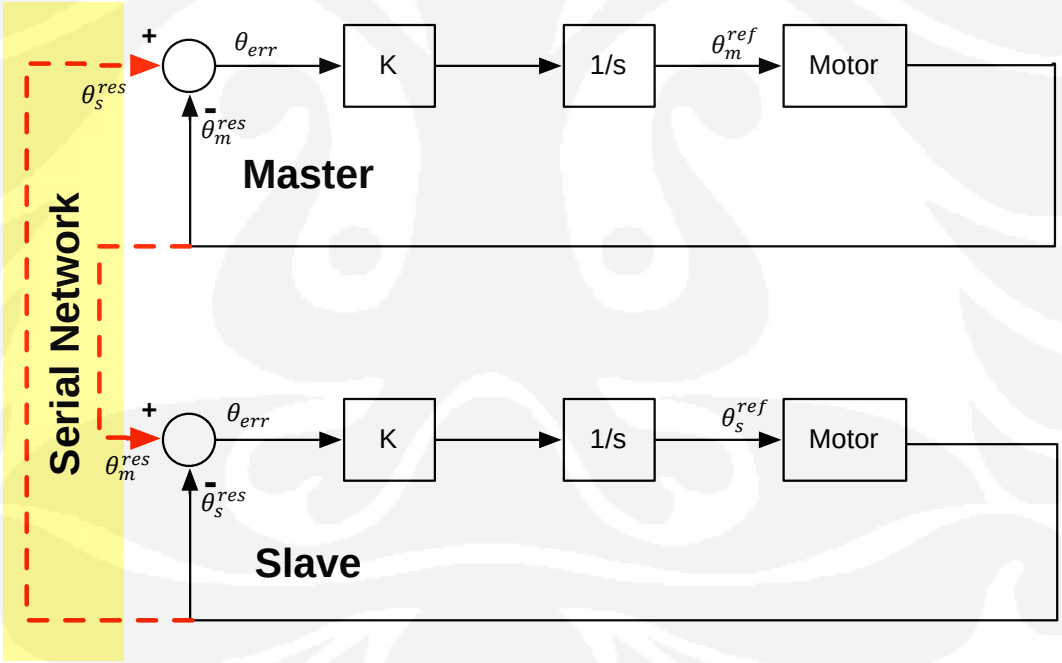
3.1 Tujuan Perancangan

Tujuan perancangan pada skripsi ini adalah mendapatkan sistem *bilateral teleoperation* yang mampu mengakomodasi kebutuhan akan multi DOF, komunikasi serial kecepatan tinggi, komunikasi melalui TCP/IP, *multi platform*, dan mudah untuk dikembangkan.

3.2 Langkah Perancangan

3.2.1 Diagram Blok Sistem

Berdasarkan studi literatur, definisi sistem *bilateral teleoperation*, dan penentuan tujuan perancangan, dihasilkan sebuah rancangan umum blok diagram dari sistem.



Gambar 3.1. Diagram Blok Sistem

Keadaan ideal sistem bilateral mengharapkan $\theta_m = \theta_s$ dimana θ_m merupakan sudut pada motor master dan θ_s merupakan sudut pada motor slave.

$$\theta_m = \theta_s \quad (1)$$

Pada blok diagram master,

$$\theta_{err} = \theta_s - \theta_m \quad (2)$$

$$\theta_m^{ref} = \int K.\theta_{err}\,.dt \quad (3)$$

$$\theta_m^{ref} = \int K(\theta_s^{res} - \theta_m^{res})dt \quad (4)$$

$$\theta_m^{ref} = \theta_m^{res} + K(\theta_s^{res} - \theta_m^{res})dt \quad (5)$$

Sedangkan pada blok diagram slave,

$$\theta_{err} = \theta_m - \theta_s \quad (6)$$

$$\theta_s^{ref} = \int K.\theta_{err}\,.dt \quad (7)$$

$$\theta_s^{ref} = \int K(\theta_m^{res} - \theta_s^{res})dt \quad (8)$$

$$\theta_s^{ref} = \theta_s^{res} + K(\theta_m^{res} - \theta_s^{res})dt \quad (9)$$

Dimana θ_{err} adalah selisih dari sudut master dan sudut slave, θ_m^{ref} adalah sudut yang dituju oleh motor master, θ_m^{res} adalah sudut output dari motor master, θ_s^{ref} adalah sudut yang dituju motor slave, θ_s^{res} adalah sudut output dari motor slave, dan dt adalah *sampling time*.

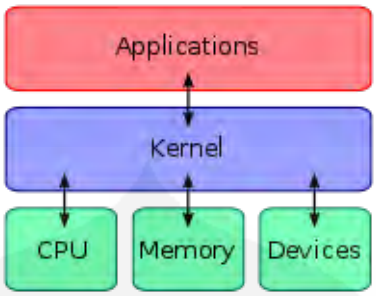
Pada diagram blok diatas terlihat bahwa sudut putaran motor dipengaruhi oleh 2 buah komponen utama. Sudut putaran motor pada sisi master dipengaruhi oleh posisi slave, dan posisi master sebelumnya. Hal yang sama juga terjadi pada putaran motor slave dimana sudut putaran motornya dipengaruhi oleh posisi master, dan posisi slave sebelumnya.

3.2.2 Pengenalan Perangkat Keras

Proses pengenalan perangkat keras dilakukan dengan cara studi literatur dan membaca datasheet serta manual kedua buah *development board* dan motor servo yang digunakan.

3.2.3 Installasi Sistem Operasi

Pemilihan sistem operasi yang tepat untuk aplikasi *bilateral teleoperation* adalah hal yang sangat penting. Sistem operasi yang digunakan harus memiliki inti yang mampu menghubungkan software aplikasi yang dirancang dengan hardware yang tersedia pada board. Inti dari sistem operasi ini biasanya disebut dengan *Kernel*.



Gambar 3.2. Kernel sebagai Penghubung

Kernel ini lah yang nantinya akan mengakses langsung register-register yang tersedia pada prosesor yang digunakan untuk menjalankan fungsi-fungsi yang disediakan oleh kernel. Fungsi-fungsi yang disediakan oleh kernel kemudian akan diakses oleh *library* yang disediakan oleh *compiler* atau para pengembang software lain. *Library* tersebut kemudian digunakan untuk mengembangkan aplikasi pada skripsi ini.

Selain itu, sistem operasi yang digunakan harus dapat dimanipulasi dengan bebas. Hal ini dimaksudkan agar ketika timbul kebutuhan untuk memodifikasi sistem pada masa yang akan datang, modifikasi dapat dilakukan dengan bebas tanpa harus melanggar lisensi-lisensi tertentu. Misalnya, terdapat kemungkinan untuk menambahkan kemampuan pemrosesan *real-time*.

Berdasarkan kriteria di atas maka Linux dipilih sebagai sistem operasi pada *development board*. Varian linux yang digunakan adalah ttylinux yang memang dikhususkan untuk *embedded system*.

Kelebihan yang dimiliki oleh ttylinux ini antara lain sistem file yang sangat kecil dan hanya membutuhkan sedikit RAM pada kondisi normal. Kekurangannya adalah ttylinux ini tidak memiliki GUI (*Graphic User Interface*) sehingga semua kegiatan operasi berlangsung di *command line* (*terminal*). Selain itu perintah-perintah yang disediakan oleh ttylinux tidak selengkap distro-distro linux yang lain.

Instalasi ttylinux dilakukan dengan bantuan software virtual box. Berikut adalah langkah-langkah untuk instalasi ttylinux:

1. Buat sebuah virtual mesin pada virtual box
2. Set memori ke 256MB agar sama seperti memori pada RB-100/RB-110
3. Hilangkan tanda centang pada pilihan *Boot Hard Disk*

4. Masuk ke *command line* (pada windows) atau *terminal* (pada linux atau Mac)
5. Masuk ke foler virtual box di C:\Program Files\oracle\virtualbox
6. Jalankan perintah vboxmanage internalcommands createrawvmdk –filename C:\usb.vmdk –rawdisk \\.\physicaldrive1 –register
7. Kembali ke virtual box dan pilih virtual mesin yang dibuat pada langkah 1
8. Sambungkan SD card ke komputer
9. Pilih *storage*, kemudian mount file ISO ttylinux sebagai CDROM IDE Primary slave dan usb.vmdk yang dibuat pada langkah 6 sebagai IDE Primary Master
10. Klik start dan virtual mesin akan menjalankan ttylinux
11. Login sebagai root sesuai dengan *username* dan *password* yang diberikan
12. Masuk ke direktori /sbin
13. Jalankan perintah ./ttylinux-installer –mbr /dev/hda /dev/hdc dan tunggu sampai proses instalasi selesai
14. Ttylinux sudah ter-install di SD card

3.2.4 Perubahan Kernel Ttylinux

Sebelum dapat menggunakan seluruh fasilitas pada development board, diperlukan sedikit modifikasi pada sistem operasi yang telah di-install. Setelah melakukan instalasi di atas akan ditemukan beberapa masalah karena *kernel* yang digunakan pada ttylinux tidak diatur sesuai dengan hardware dan pengaturan pada board RB-100/Rb-110. Beberapa masalah yang timbul antara lain tidak adanya *driver* LAN, *driver* Hi-Speed Serial TTL, dan *driver audio*. Untuk mengatasi masalah ini harus dilakukan penggantian *kernel* ttylinux. Proses penggantian *kernel* dilakukan dalam 3 tahap yaitu konfigurasi, *build*, dan instalasi. Pada tahap konfigurasi, digunakan file konfigurasi milik X-Linux. Berikut adalah langkah-langkah untuk mengganti kernel linux, semua langkah selain *download* dilakukan di terminal.

1. *Download kernel source* dari www.kernel.org
2. Ekstrak *kernel source*
3. *Download* X-Linux *source* dari ftp://download@ftp.dmp.com.tw/os-xlinux/xlinux-5.7-src-dx.zip

4. Ekstrak X-Linux *source* dan *copy* file .config ke direktori ekstrak *kernel source* pada langkah 2
5. Masuk ke direktori *kernel source* dan jalankan perintah make menuconfig
6. *Load* file .config hasil ekstraksi pada langkah 4
7. Masuk ke bagian *device driver* dan aktifkan *driver* FTDI
8. Simpan hasil konfigurasi dan keluar dari menuconfig
9. *Compile kernel source* sesuai dengan arsitektur prosesor RB-100/RB-110, dalam hal ini arsitektur yang digunakan adalah i486. Hal ini dikenal dengan nama *cross compile*
10. Jalankan perintah make untuk melakukan *cross compile* ke arsitektur i486.
    make ARCH=i486 CROSS_COMPILE=/usr/local/bin/i486-linux
11. Setelah proses *compile* selesai, akan dihasilkan dua buah file yaitu bzimage dan system map. *Copy* file bzimage ke dalam folder /boot di SD card
12. Modifikasi file lilo.conf yang terletak pada direktori /etc di SD card
13. Tambahkan baris berikut setelah baris terakhir file lilo.conf
    image = /boot/bzimage
    label = ttylinux-i486
    root = <disamakan dengan root pada image sebelumnya>
    append = “8250.nr_uarts = 4”
14. Ubah nilai *default* pada line 24 menjadi ttylinux-i486
15. Simpan file lilo.conf
16. Masuk ke direktori /sbin pada SD card dan eksekusi ./lilo

Setelah melakukan *reboot* maka sistem akan otomatis menggunakan kernel terakhir yang baru saja di-install.

3.2.5 Pengaturan Development Board dan Sistem Operasi

Untuk memanfaatkan seluruh fasilitas ttylinux maka diperlukan beberapa pengaturan pada beberapa file.

Port komunikasi selain RS-232 dan RS-485 secara *default* tidak diaktifkan oleh RB-100/RB-110. Pengaturan aktifasi port ini dapat dilakukan melalui BIOS pada menu *Chipset*. Setelah diaktifkan melalui BIOS, *port* ini harus dikenali oleh ttylinux. Untuk menambahkan *port* komunikasi ke ttylinux harus dilakukan

modifikasi pada direktori /dev dan mengeksekusi perintah mknod –m 660 /dev/ttyS2 c 4 66 untuk mengaktifkan COM3 dan mknod –m 660 /dev/ttyS3 c 4 67 untuk mengaktifkan COM4.

Untuk mengatur konfigurasi *firewall* dapat dilakukan melalui file /etc/firewall.conf dengan menambahkan jenis *service* dan *port* yang digunakan untuk *service* tersebut. Terdapat *bugs* pada file konfigurasi ini dimana pengaturan yang telah dilakukan pada file firewall.conf sama sekali tidak bekerja. Akibatnya board tidak dapat menerima ataupun melakukan koneksi sama sekali sehingga *transfer* file ataupun koneksi SSH ke board tidak dapat dilakukan. Hal ini dapat diatasi dengan cara menghapus *script firewall* pada direktori /etc/rc.d/init.d menggunakan perintah rm.

Ttylinux menyediakan fitur SSH (*Secure Shell*) yang dapat digunakan untuk koneksi *remote login*. SSH merupakan sebuah protokol jaringan yang memungkinkan terjadinya pertukaran data menggunakan *secure channel* antara dua buah peralatan dalam jaringan. Semua fungsi ttylinux pada *development board* dapat diakses melalui komputer lain tanpa perlu bersentuhan langsung dengan *board*, memasang *keyboard*, atau monitor pada *board* dengan menggunakan SSH. Fitur ini secara *default* tidak diaktifkan karena akan menghambat kinerja prosesor pada saat pertama kali digunakan. Untuk mengaktifkan fitur ini, pengguna harus mengubah file ssh yang terletak pada direktori /etc/sysconfig. Setelah melakukan perubahan ini, fitur SSH hanya akan aktif sementara sampai terjadi *logout*. Untuk melakukan perubahan permanen, harus dilakukan perubahan file rc.sysinit pada direktori /etc/rc.d dengan menambahkan tanda # pada setiap baris mulai dari baris 573 sampai baris 587.

3.2.6 Membuat Cross Compiler Tool-Chain

Cross Compiler Tool Chain adalah sekumpulan peralatan untuk melakukan *software development*. Penggunaan *cross compiler* memungkinkan pemrograman pada mesin yang memakai prosesor jenis A dan hasil pemrograman-nya dapat digunakan pada mesin yang memakai prosesor jenis B. Pada skripsi ini pemrograman dilakukan pada komputer dengan arsitektur prosesor i686 sedangkan program hasilnya akan dijalankan pada RB-100/RB-110 yang menggunakan arsitektur prosesor i486. Proses pemrograman dilakukan pada

komputer lain karena ttylinux pada RB-100/RB-110 tidak menyediakan IDE (*Integrated Development Environment*) yang memadai. Selain itu ttylinux hanya menyediakan *tiny C compiler* (TCC) yang tidak selengkap *GNU C compiler* (GCC).

Berikut adalah langkah-langkah untuk membuat *cross compiler tool chain*, semua langkah kecuali *download* dilakukan melalui terminal.

1. *Download ttylinux source code* dari minimalinux.org/ttylinux/source.html
2. Ekstrak hasil *download*
3. Masuk ke direktori /home/…/ttylinux-src-xxx/cross-tools-2.9-2.6.xx/_scripts/ dan edit file generic-linux-gnu.sh
4. Pada file generic-linux-gnu.sh, masuk ke line 632 dan sisipkan line berikut `sed -i "s/getline/uc_&/" scripts/unifdef.c`
5. Masuk ke direktori /hom/…/ttylinux-src-xxx/cross-tools-2.9-2.6.xx
6. Jalankan perintah `make setup && make dload`, ttylinux akan men-*download source* yang dibutuhkan untuk membangun *cross compiler*.
7. Jalankan perintah `make i486` untuk menghasilan *cross compiler* dengan target mesin i486.

3.2.7 Library dan Framework

Skripsi ini menggunakan ulang (*re-use*) beberapa *library* dan *framework* dari *developer* lain. Penggunaan ulang *library* ini akan mempermudah pengembangan aplikasi karena tidak perlu menulis ulang fungsi-fungsi yang telah dikembangkan oleh *developer* lain.

Aplikasi pada skripsi ini dipersiapkan agar dapat dengan mudah di-*compile* pada sistem operasi dan arsitektur prosesor yang berbeda tanpa perlu menulis ulang atau melakukan modifikasi sesuai dengan sistem operasi dan arsitektur yang lain. Hal ini dimaksudkan agar skripsi ini dapat dengan mudah dilanjutkan tanpa terikat oleh satu sistem operasi, satu *compiler*, dan satu *development board*. Untuk mengakomodasi kebutuhan *multi-platform* digunakan *library* dan *framework* ACE (*Adaptive Communication Environment*) milik Douglas Schmidt. ACE menyediakan *Operating System Adapter Layer* agar *source code* yang dikembangkan dapat dijalankan pada banyak sistem operasi

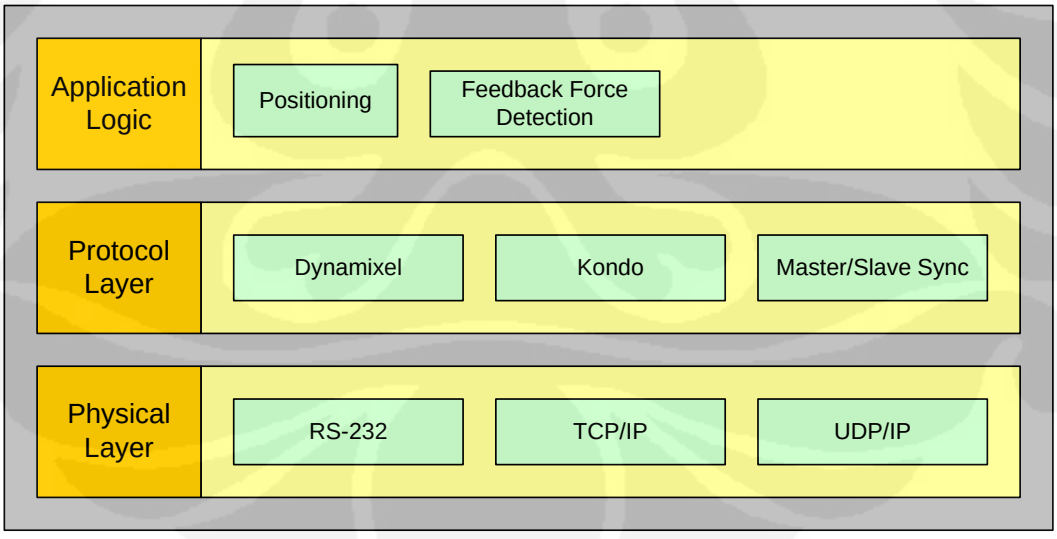
seperti Windows, Linux, Mac OS X, UNIX, BSD, bahkan pada sistem operasi *embedded* seperti VxWorks dan Windows CE.

Selain menyediakan fasilitas *multi-platform*, ACE juga menyediakan *library* yang memungkinkan pembuatan aplikasi *multi-threading* dan mempermudah komunikasi antar proses (Inter Process Communication). Keuntungan lain dari penggunaan ACE adalah mempermudah *network programming*. Aplikasi yang dibuat pada skripsi ini dipersiapkan agar dapat berkomunikasi melalui TCP/IP sehingga pada masa yang akan datang dapat dikembangkan menjadi aplikasi *web based embedded* atau *wireless embedded*.

Aplikasi yang dibuat juga menggunakan *library* milik Jimmy Merari yang mengadaptasi *library* ACE untuk memfasilitasi adanya *multi-protokol*. Pada skripsi ini terdapat 3 jenis protokol di luar protokol standar seperti TCP/IP. 3 jenis protokol ini adalah protokol dynamixel, protokol kondo, dan protokol *main controller*. *Library* ini digunakan untuk memfasilitasi perbedaan ketiga protokol tersebut.

3.2.8 Arsitektur Software

Aplikasi yang dibuat dalam skripsi ini dibagi menjadi 3 *layer* yaitu *application layer*, *protocol layer*, dan *physical layer*.



Gambar 3.3. Arsitektur Software

Aplikasi ini menggunakan sistem *plugins* agar dapat dikembangkan dengan lebih mudah. Jika pada masa yang akan datang ingin ditambahkan jenis serial servo lain atau terjadi penambahan fungsi maka tidak perlu memodifikasi

program utama tetapi cukup menambahkan *plugins*. Berikut penjelasan modul-modul *plugins* pada aplikasi ini:

LibCore merupakan inti dari aplikasi ini, bertugas untuk penjadwalan *pooling*, konfigurasi dan manajemen *hardware* yang terkoneksi, pengaturan mekanisme *plugins*, dan enkapsulasi *physical layer*.

LibKondo merupakan *plugins* komunikasi untuk serial servo Kondo KRS-2552-HV

LibDynamixel merupakan *plugins* komunikasi untuk serial servo Dynamixel AX-12+

LibHostMaster merupakan *plugins* komunikasi bilateral pada sisi *master*

LibHostSlave merupakan *plugins* komunikasi bilateral pada sisi *slave*

LibEntities merupakan modul yang berisi hal-hal yang terkait dengan *aplication layer* misalnya *trigger* untuk set posisi, *read* posisi, dan pembacaan *force feedback.*

3.2.9 Protokol Komunikasi Master – Slave

Format protokol untuk komunikasi master dan slave adalah sebagai berikut

Tabel 3.1. Format Protokol Komunikasi Master – Slave

Offset	Field	Keterangan
0	STX	Start transmission (0x02)
1	ADDRESS	Valid address 1 – 127
2 : :	Protocol Data Unit	Function code dan data
(n-1)	ETX	End Transmission (0x03)
n	LRC	Error checking. Metode XOR seluruh data dari offset 2 sampai offset (n-1)

Protocol Data Unit (PDU) berisi *function code* dan data yang dapat bernilai berapapun termasuk 0x02 dan 0x03 yang difungsikan sebagai STX dan ETX. Untuk membedakan antara data dan STX/ETX maka dilakukan prosedur *byte stuffing*. Prosedur *byte stuffing* memeriksa seluruh data pada PDU, jika

ditemukan data bernilai 0x02, 0x03, dan 0x10 maka prosedur akan menambahkan karakter 0x10 sebelum data tersebut sebagai penanda bahwa data 0x02 dan 0x03 bukan STX/ETX. Nilai LRC dihitung sebelum penerapan prosedur *byte stuffing*. Contoh *byte stuffing* dapat dilihat pada tabel berikut

Tabel 3.2. Data Sebelum Byte Stuffing

STX	PDU	ETX	LRC
0x02	0x01 0x45 0x71 **0x10** 0x56 **0x02** 0x28 **0x03**	0x03	0x59

Tabel 3.3. Data Setelah Byte Stuffing

STX	PDU	ETX	LRC
0x02	0x01 0x45 0x71 **0x10 0x10** 0x56 **0x10 0x02** 0x28 **0x10 0x03**	0x03	0x59

Format *function code* pada protokol ini dapat dilihat pada tabel dibawah

Tabel 3.4. Format Function Code

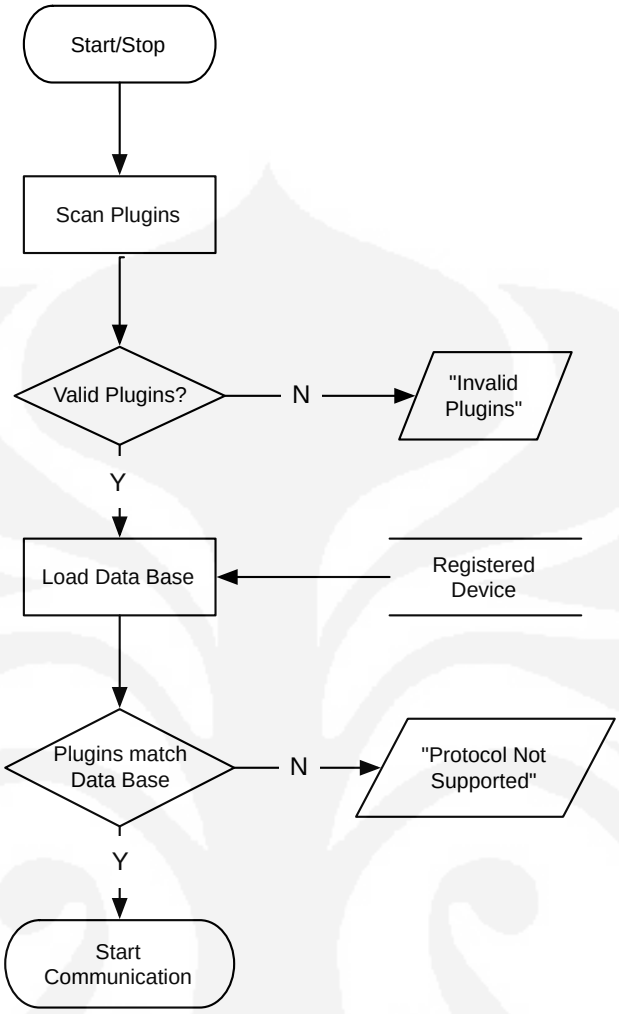
Offset	Field	Keterangan
0	Command Code	0x01
1	Entity Count	Banyaknya parameter yang di-*sync*
2	Hashed number device 0	Hasil operasi CRC32 dari nama device yang terdaftar pada database robot.db.sqlite
3		
4		
5		
6	Position Motor 0	Nilai posisi untuk motor 0. Nilai ternormalisasi dengan skala 0 – 1. Dinyatakan dalam format *floating point* IEEE 754
7		
8		
9		
10	Speed Motor 0	Nilai kecepatan untuk motor 0. Nilai ternormalisasi dengan skala 0 – 1. Dinyatakan dalam format *floating point* IEEE 754
11		
12		
13		
14	Torque Motor 0	Nilai torsi untuk motor 0. Nilai

15 16 17		ternormalisasi dengan skala 0 – 1. Dinyatakan dalam format floating point IEEE 754
: : :		Pengulangan untuk motor berikutnya dengan format yang sama

Pengiriman data juga merupakan sebuah *query* untuk meminta data posisi dari board lain. Setelah mengirimkan data, board akan mendapatkan respon berupa data dari board pasangan dengan format yang sama.

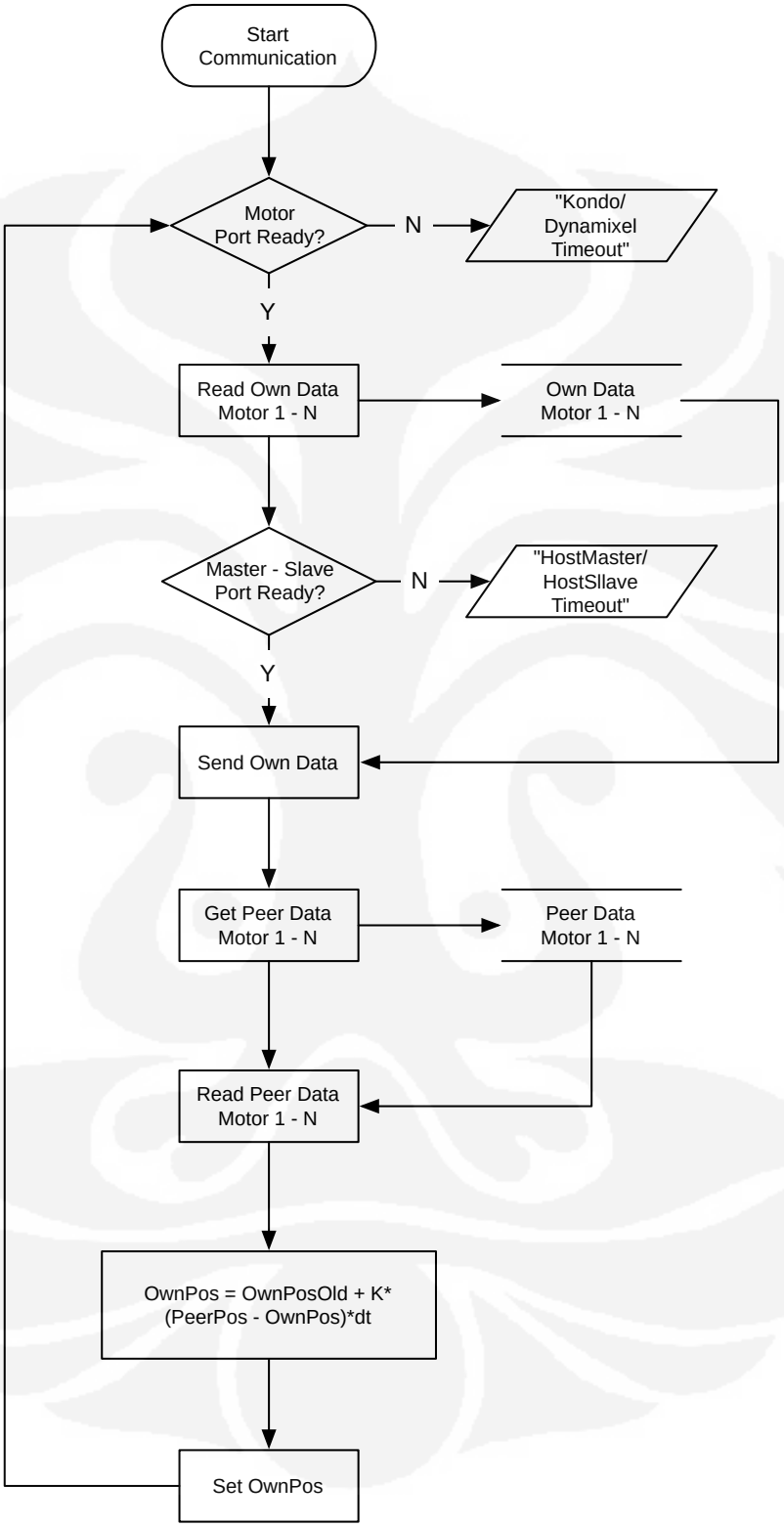
3.2.10 Algoritma

Secara umum algoritma utama yang dilakukan pada 2 buah *main controller* adalah saling mempertukarkan data posisi masing-masing dan mengolah data posisi tersebut agar posisi servo yang dikendalikan oleh *main controller* seolah-olah bergerak secara bersamaan. Dengan data posisi dapat menghasilkan sensasi sentuhan walaupun kurang akurat.



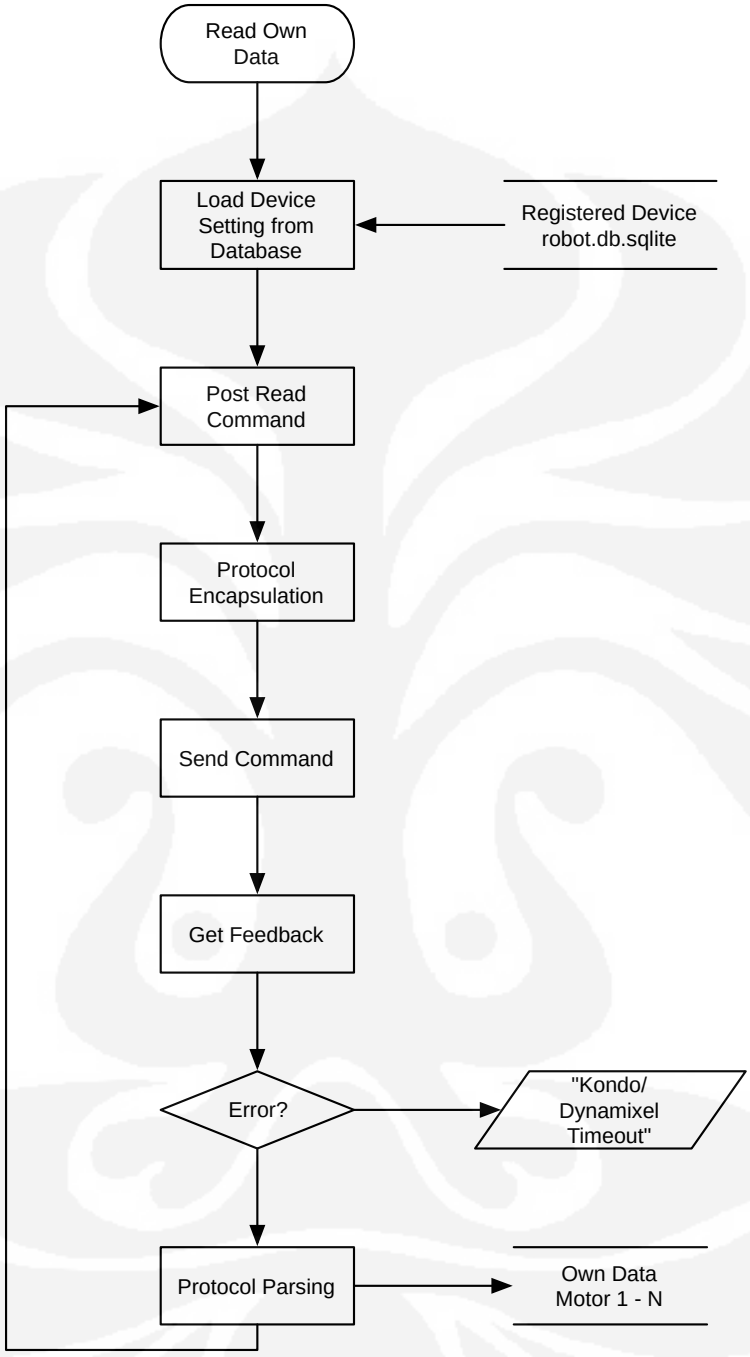
Gambar 3.4. Flowchart Algoritma pada Main Controller

Algoritma komunikasi antara master – slave dijelaskan oleh *flowchat* di bawah ini



Gambar 3.4. Flowchat Komunikasi Master – Slave

Algoritma komunikasi antara board dan motor dijelaskan oleh flowchat di bawah ini



Gambar 3.5. Flowchart Komunikasi Board – Motor

BAB 4
PENGUJIAN DAN ANALISA

4.1 Tujuan Pengujian

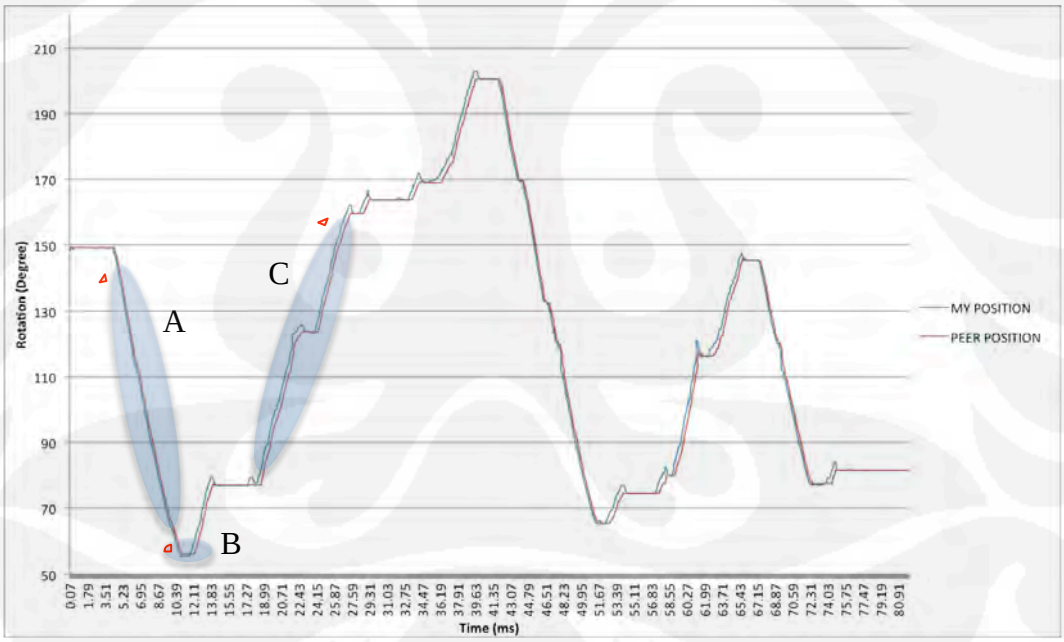
Pengujian sistem dilakukan untuk mengetahui besarnya perbedaan sudut antara *master* dan *slave* dalam suatu waktu tertentu.

4.2 Metode Pengujian

Pengujian dilakukan dengan cara meng-*export* data posisi master dan slave pada *sampling* waktu rata-rata 20ms. Data yang diproses berjumlah maksimum 3700 data dan minimum 2400 data. Sudut putaran dinormalisasi dengan skala 0° - 300° menjadi 0 – 1.

4.3 Data Hasil Pengujian dan Analisa

Berikut adalah grafik data hasil pengujian pertama



Gambar 4.1. Grafik Hasil Pengujian Pertama

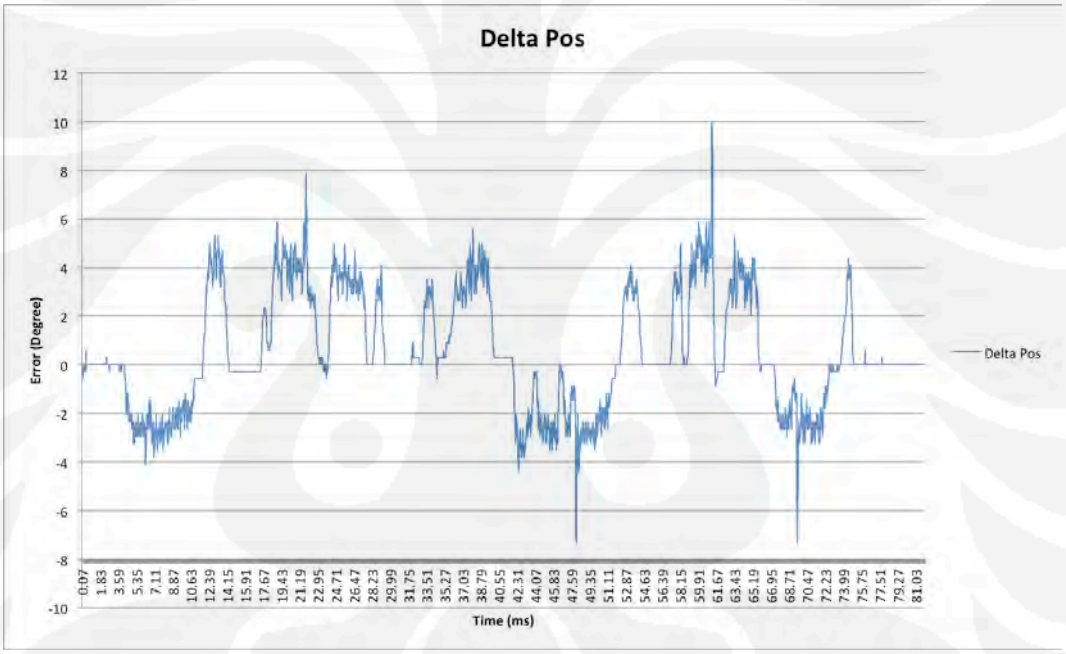
Sumbu Y grafik merupakan posisi sudut putaran servo dalam derajat sedangkan sumbu X merupakan sumbu waktu.

Dari grafik diatas terlihat adanya perbedaan antara posisi sudut *master* (grafik *My Position*) dan *slave* (grafik *Peer Position*). Pada posisi A terlihat *slave*

berusaha menyusul posisi *master* pada putaran *clockwise*. Pada posisi B terlihat *slave* berhasil menyamakan posisi dengan *master*. Pada posisi C terlihat *slave* berusaha menyusul posisi *master* pada putaran *counter clockwise*. Pada posisi D terlihat *slave* kembali berhasil menyamakan posisi dengan *master*.

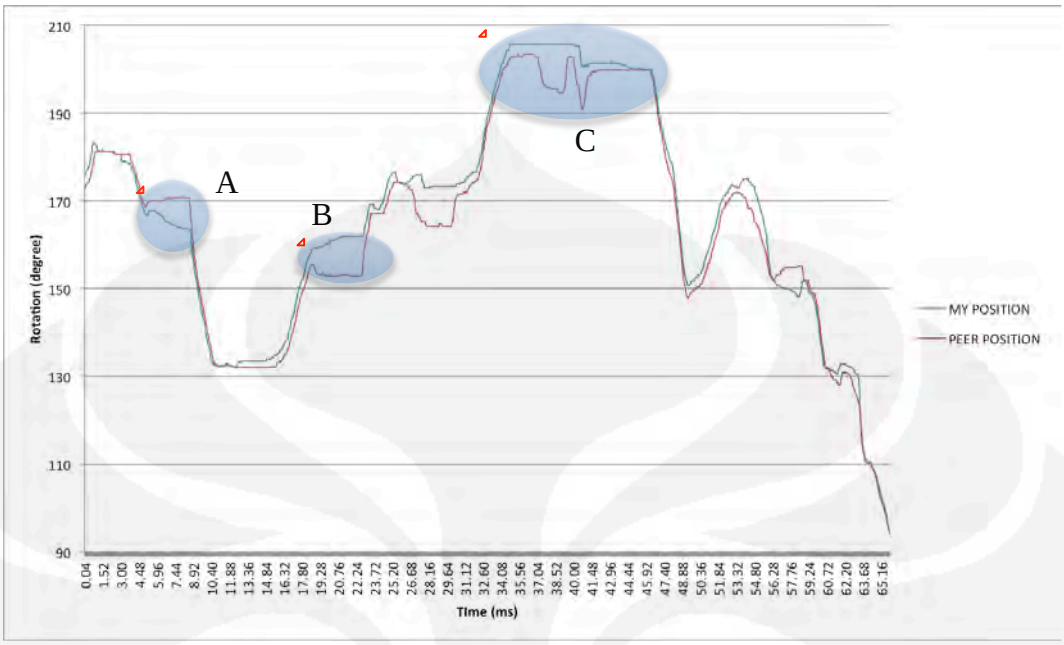
Pada pengujian pertama, operator menggerakkan servo pada sisi *master* untuk mengetahui respon *slave* terhadap *master*.

Hasil pengolahan data menunjukkan bahwa perbedaan posisi sudut rata-rata antara *master* dan *slave* adalah sebesar 0.833° dengan perbedaan sudut terbesar yaitu 9.745° dengan kondisi *master* mendahului *slave* dan 6.572° dengan kondisi *slave* mendahului *master*. Berikut adalah grafik perbedaan sudut antara *master* dan *slave*



Gambar 4.2. Error Pengujian Pertama

Berikut adalah grafik data hasil pengujian kedua



Gambar 4.3. Grafik Hasil Pengujian Kedua

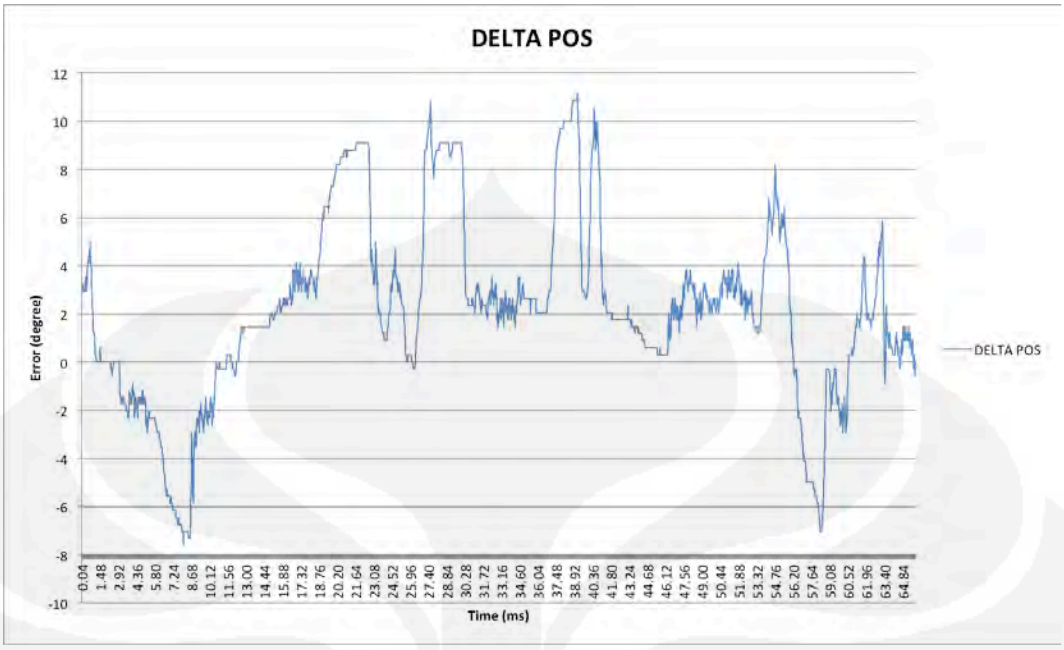
Pengujian kedua dimaksudkan untuk menguji respon *master* apabila terjadi gangguan lingkungan ekstrim pada *slave*. Pada pengujian ini disimulasikan servo *slave* tersangkut dan tidak dapat bergerak ke arah yang dituju servo *master*.

Pada posisi A disimulasikan servo *slave* tertahan oleh lingkungan. Terlihat servo *master* masih bisa bergerak sampai posisi tertentu hingga akhirnya tertahan. Begitu halangan pada servo *slave* dilepas, servo *slave* segera menyesuaikan diri dengan posisi servo *master*. Hal yang sama juga terjadi pada posisi B tetapi dalam arah putaran yang berkebalikan.

Pada posisi C disimulasikan servo *master* tertahan sedangkan servo *slave* mendapat gaya dari lingkungan untuk tetap bergerak. Terlihat bahwa servo *slave* masih bisa bergerak sampai posisi tertentu hingga akhirnya tertahan dan berusaha menyesuaikan diri dengan servo *master*.

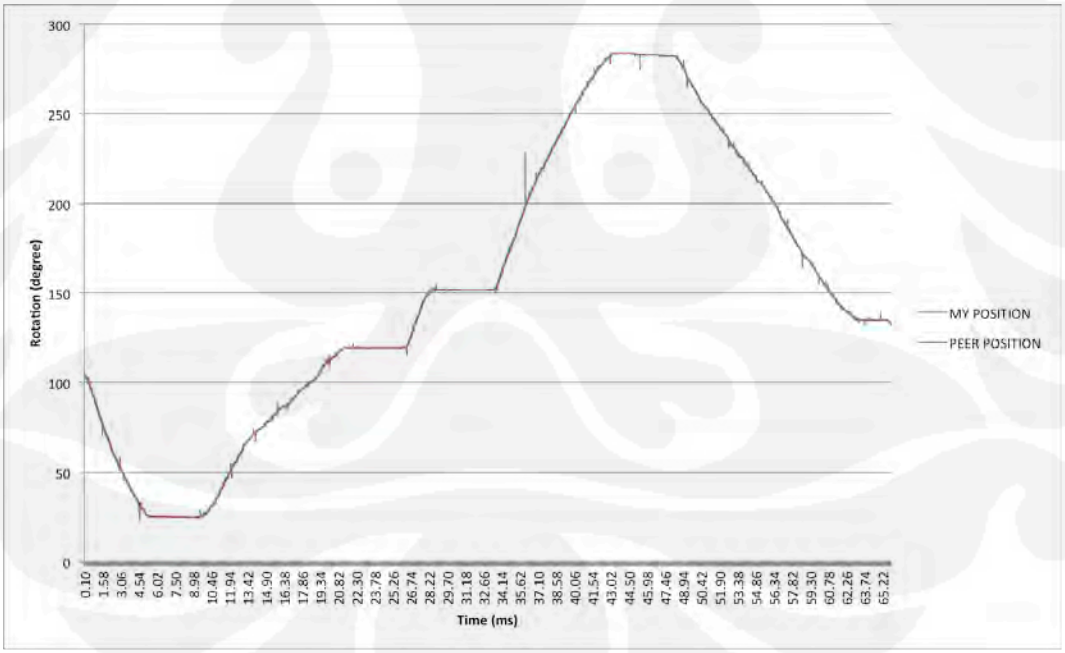
Selisih maksimum dari perbedaan posisi *master* dan *slave* merupakan jarak maksimum yang dapat ditempuh servo ketika servo pasangannya tertahan. Melalui pengolahan data diketahui bahwa jarak maksimum yang dapat ditempuh servo ketika servo pasangannya tertahan adalah 11.143°.

Selisih posisi sudut antara master dan slave terlihat pada grafik berikut



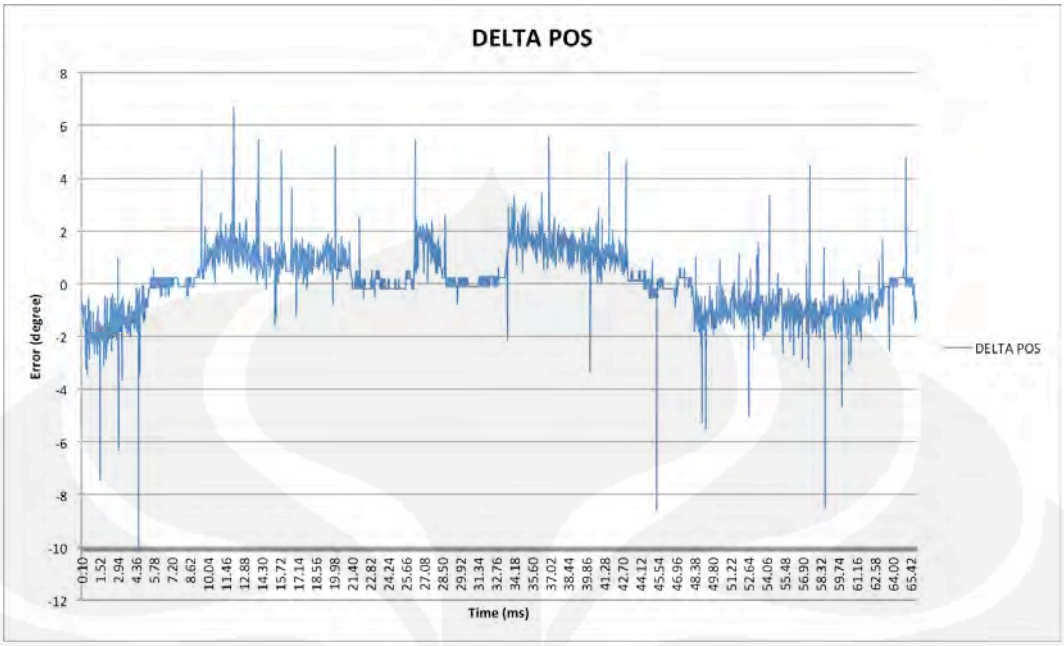
Gambar 4.4. Error Pengujian Kedua

Pengujian ketiga dilakukan dengan menggerakkan servo pada master secara perlahan. Berikut adalah grafik hasil pengujian ketiga



Gambar 4.5. Grafik Hasil Pengujian Ketiga

Dari grafik di atas terlihat bahwa slave merespon lebih baik pada putaran pelan. Selisih putaran sudut antara master dan slave dapat dilihat dari grafik dibawah ini



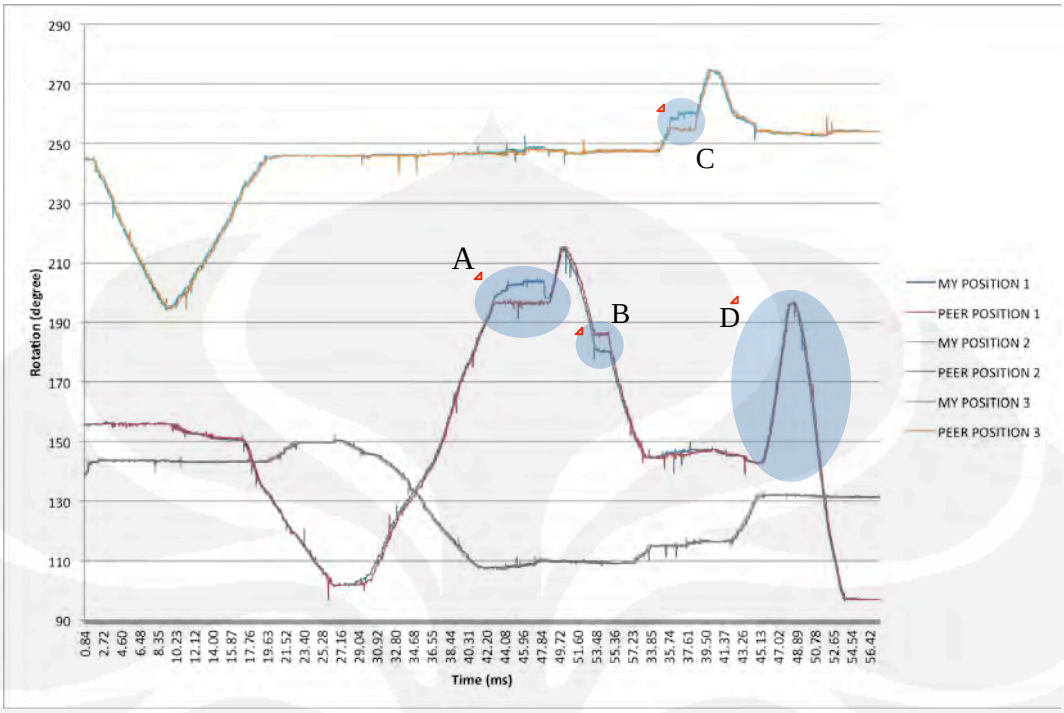
Gambar 4.6. Error Pengujian Ketiga

Hasil pengolahan data menunjukkan bahwa perbedaan posisi sudut rata-rata antara *master* dan *slave* adalah sebesar 0.675° dengan perbedaan sudut terbesar yaitu 6.675° dengan kondisi *master* mendahului *slave* dan 10.2° dengan kondisi *slave* mendahului *master*.

Pengujian keempat dilakukan dengan menggunakan 3 motor sekaligus (3 DOF) pada sisi *master* dan *slave*. Pengujian dilakukan seperti gambar di bawah ini

Gambar 4.7. Platform Pengujian Multi DOF

Hasil pengujian multi DOF dapat dilihat pada grafik dibawah ini



Gambar 4.8. Grafik Hasil Pengujian Keempat

Dari grafik di atas terlihat bahwa 3 motor dapat berjalan secara bersamaan baik pada sisi *master* maupun pada sisi *slave*. Slave dapat merespon gerakan master dengan baik, terlihat dari grafik *master* dan *slave* yang berhimpitan.

Pada titik A disimulasikan *slave* motor 1 tertahan, tampak *master* motor 1 tidak dapat melanjutkan putarannya. Dari pengolahan data diketahui bahwa selisih maksimal antara *master* dan *slave* pada titik A adalah sebesar 11.58°. Pada titik B disimulasikan *master* motor 1 tertahan, tampak *slave* motor 1 juga tertahan dan tidak dapat melanjutkan putaran. Selisih maksimal antara *master* dan *slave* pada titik B adalah sebesar 9.26°. Pada titik C disimulasikan *slave* motor 3 tertahan, *master* motor 3 tidak dapat meneruskan putaran. Selisih maksimal antara *master* dan *slave* pada titik C adalah sebesar 6.3°. Pada titik D disimulasikan *master* motor 1 digerakkan lebih cepat oleh operator, tampak dari kemiringan grafik yang lebih curam. *Slave* motor 1 dapat merespon dengan baik terbukti dengan kurva yang berhimpitan dan rata-rata selisih sudut antara *master* dan *slave* sebesar

BAB 5
KESIMPULAN

Dari keseluruhan pembahasan dalam skripsi ini dapat disimpulkan beberapa hal, yaitu :

1. Sistem bilateral teleoperation berhasil direalisasikan dengan menggunakan serial servo Kondo KRS 2552 HV dan Dynamixel AX-12+ serta development board RB-100 dan RB-110.
2. Dengan menggunakan aplikasi yang dirancang pada skripsi ini, servo slave dapat mengikuti servo master, begitu pula sebaliknya.
3. Adanya torsi pada servo dapat dimanfaatkan untuk menimbulkan efek haptic
4. Pada performa terbaiknya, perbedaan posisi sudut rata-rata antara master dan slave adalah 0.103°
5. Perbedaan posisi sudut terbesar antara master dan slave dalam kondisi normal adalah 7.624°
6. Pada kondisi slave tertahan, perbedaan sudut yang terjadi antara master dan slave bisa mencapai 11.143°.

DAFTAR REFERENSI

Ching, Ho. (2006). *Internet Based Bilateral Teleoperation*.Georgia: Georgia Institute of Technology.

Hokayem, Peter F. & Spong, Mark W. (2006). *Bilateral Teleoperation: an Historical Survey*. *Journal of Automatica 42 (2006) 2035 – 2057*.

Kroah, G. & Hartman. (2007). *Linux Kernel in Nutshell*. California: O' Reilly.

Huston, Stephen D. (2004). *The ACE Programmers Guide: Practical Design Pattern for Network and Systems*. Washington: Addison-Wesley.

Schmidt, Douglas C. (2002). *C++ Network Programming*. Boston: Addison-Wesley

Yaghmour, K., Masters J., Ben-Yossef, G., & Gerum, P. (2008). *Building Embedded Linux System* (2nd ed.). California: O' Reilly.