UNIVERSITAS INDONESIA

PENGEMBANGAN SISTEM KONTROL PERGERAKAN ROBOT ARTIKULASI 5 DERAJAT KEBEBASAN BERBASIS *WEB*

SKRIPSI

HENDRA PRIMA SYAHPUTRA
0405020332

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK MESIN
DEPOK
DESEMBER 2009

UNIVERSITAS INDONESIA

PENGEMBANGAN SISTEM KONTROL PERGERAKAN ROBOT ARTIKULASI 5 DERAJAT KEBEBASAN BERBASIS *WEB*

SKRIPSI

Diajukan sebagai salah satu syarat
untuk memperoleh gelar sarjana teknik

HENDRA PRIMA SYAHPUTRA
0405020332

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK MESIN
DEPOK
DESEMBER 2009

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama	**: Hendra Prima Syahputra**
NPM	**: 0405020332**
Tanda Tangan	**: ……**
Tanggal	**:**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Hendra Prima Syahputra
NPM	:	0405020332
Program Studi	:	Teknik Mesin
Judul Skripsi	:	PENGEMBANGAN SISTEM KONTROL PERGERAKAN ROBOT ARTIKULASI 5 DERAJAT KEBEBASAN BERBASIS *WEB*

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

DEWAN PENGUJI

Pembimbing	: Dr. Ir. Gandjar Kiswanto, M.Eng	( )
Penguji	: Ir. Hendri DS Budiono, M.Eng	( )
Penguji	: Ir. Henky S Nugroho, M.T	( )
Penguji	: Dr. Ario S Baskoro, S.T, M.T, M.Eng	( )

Ditetapkan di :

Tanggal :

ABSTRAK

Nama : Hendra Prima Syahputra

Program Studi : Teknik Mesin

Judul : Pengembangan Sistem Kontrol Robot Artikulasi 5 Derajat Kebebasan Berbasis *Web*

Penelitian ini merancang sebuah sistem yang mampu mengontrol sebuah robot artikulasi dengan lima derajat kebebasan dari jarak jauh melalui media internet yang berbasiskan aplikasi *web*. Dalam penelitian ini digunakan sebuah komputer yang bertindak sebagai *server* yang dilengkapi dengan dua buah *web camera* untuk memantau kondisi dan pergerakan robot dan juga sebuah mikrokontroler pengontrol robot sebagai pemroses dan pengontrol masukan untuk menggerakkan robot. Melalui sebuah *web browser* pada komputer yang bertindak sebagai client, sistem pada komputer *server* diakses oleh pengguna dan menampilkan sebuah antarmuka yang dirancang sebagai panel kontrol robot. Melalui antarmuka ini pengguna dapat memberi masukan berupa perintah untuk menggerakkan robot yang dapat diberikan dalam dua pilihan mode basis kontrol, yaitu *cursor-based/inverse kinematics* dan *manual/forward kinematics*. Sistem mampu merespon perintah yang diberikan kemudian memroses dan mengeksekusinya dalam bentuk pergerakan robot sesuai dengan mode dan perintah dari masukan yang diberikan.

Kata kunci:

Kontrol robot, aplikasi *web*

ABSTRACT

Name : Hendra Prima Syahputra

Study Program : Mechanical Engineering

Title : Development of Web Based 5-DOF Articulated Robot Motion Control System

This research is aimed to design and develop a system capable of remotely controlling a five-degree-of-freedom articulated robot through internet platform on a web-based application. The research was built with single computer act as a server coupled with a pair of web camera to monitor the status and movement of the robot and also coupled with a robot-controller micro controler as a processor and controller of inputs to move the robot. Through the web browser on user's computer acting as *client*, the system is accessed by the user and displays an interface designed to be a robot's control panel. Through this interface, the user can input command to move the robot which can be given in two different control modes, cursor-based/inverse kinematics and manual/forward kinematics. System responds the command then processes and executes it in form of robot movement based on control mode and command of the given input.

Keywords:

Robot control, web application

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Mesin pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, penyusunan skripsi ini sangatlah sulit bagi saya. Oleh karena itu, saya mengucapkan terima kasih kepada:

1) Orang tua saya yang telah memberikan dukungan moril dan materiil;
2) Dr. Ir. Gandjar Kiswanto, M.Eng selaku dosen pembimbing yang telah meluangkan waktunya untuk menyemangati saya dan menginspirasi saya dalam pengerjaan skripsi ini;
3) Dr. Ir.Harinaldi, M.Eng selaku kepala Departemen Teknik Mesin;
4) Dr. Ir.Abdul Muis, M.Eng yang telah memberikan semangat dan pengetahuan elektronika kepada saya;
5) Sahabat saya Auralius Manurung, S.T, Gunawan, S.T, M.Syafiuddin, S.T, Erwin Nugraha Bayuaji, dan Ahmad Zakiyudin yang telah memberikan pengetahuan software dan elektronika kepada saya;
6) Seluruh teman dalam Tim Robot Universitas Indonesia (TRUI) atas segala pengalaman dan pengetahuan yang saya dapat di TRUI;
7) Tim WRMA (*Web* Based Manipulation), Anom Tejo, Adithya Bayu dan Teguh Santoso atas kerja samanya dalam penyelesaian skripsi ini;
8) Seluruh teman seperjuangan saya mesin 2005.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Depok, Desember 2009

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR ISTILAH

ISO : International Organization for Standardization.
DOF : Degree of Freedom.
DSP : Digital Signal Processing.
DC : Direct Current.
NC : Normally Close.
NO : Normally Open.
IC : Integrated Circuit.
CPU : Central Processing Unit
I/O : Input Output
RISC : Reduced Instruction Set Computing
PWM : Pulse Width Modulation
LED : Light Emitting Diode
PCB : Printed Circuit Board
EMF : Electromagnetic Field
LCD : Liquid Crystal Display
USB : Universal Serial Bus
UART : Universal Asynchronous Receiver-Transmitter
CPR : Count per Revolution

BAB 1
PENDAHULUAN

1.1 Latar Belakang

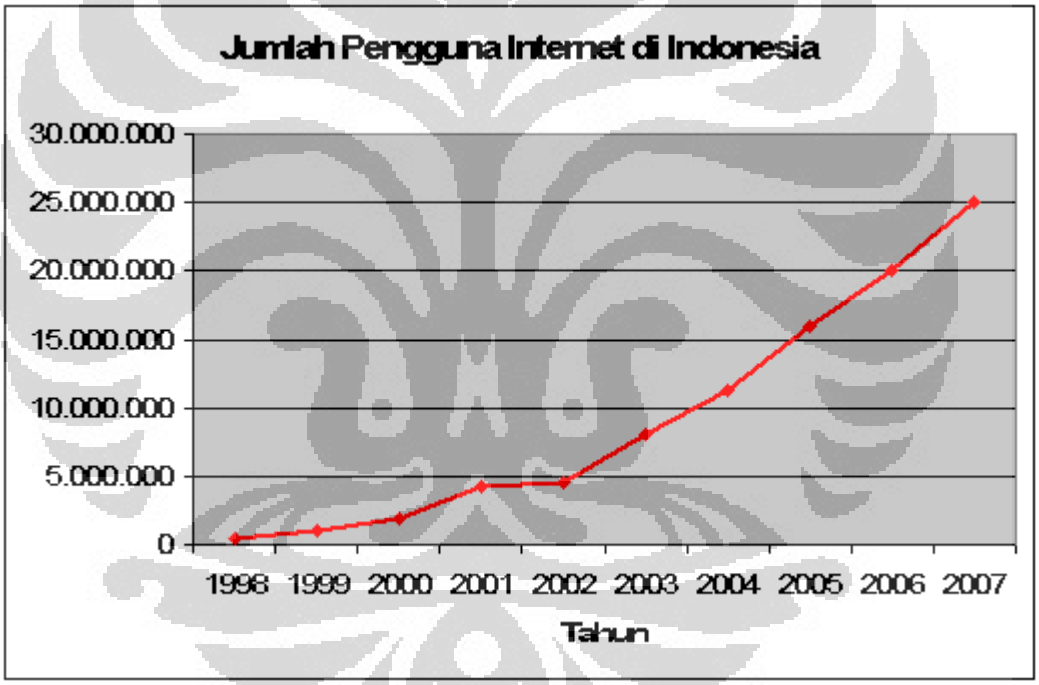
Penggunaan robot didalam dunia perindustrian semakin luas, hal ini terjadi seiring dengan meningkatnya kebutuhan masyarakat akan barang – barang yang berkualitas. Karena kelebihan robot yang mampu bekerja lebih banyak dan lebih konsisten dibandingkan manusia maka mulai bermunculan robot – robot dengan bentuk dan kegunaan yang bervariasi. Perkembangan ini didukung dengan kemajuan teknologi didalam bidang mekanikal, elektronika dan teknologi informasi. Teknologi tersebut kemudian diaplikasi pada berbagai macam robot untuk meningkatkan kecerdasan, daya tahan, dan akurasi dari robot – robot tersebut.

Salah satu dasar pengembangan robot adalah memberikan pertolongan kepada manusia untuk melakukan pekerjaan yang sulit dan membahayakan bagi manusia. Beberapa industri yang menuntut tingkat kepresisian yang tinggi tidak menggunakan lagi manusia sebagai pekerjanya namun robot sebagai pekerja didalam pabrik. Hal ini menuntut pengembangan robot yang mampu memenuhi permintaan manusia. Salah satu kebutuhan tersebut adalah pengendalian robot dari jarak jauh. Pengendalian robot dari jarak jauh sangat dibutuhkan untuk mempermudah manusia memonitor pekerjaan robot dan mengkoreksi pekerjaan robot pada pabrik – pabrik yang tidak lagi menggunakan manusia sebagai pekerjanya.

Salah satu metode pengendalian robot dari jarak jauh yaitu dengan menggunakan internet. Pengendalian menggunakan internet didasari karena perkembangan teknologi informasi sekarang yang banyak sekali menggunakan internet. Kebutuhan pengendalian melalui internet semakin diperlukan karena adanya kebutuhan untuk mengendalikan dan memonitor sistem yang dikendalikan dari tempat yang berbeda dari sistem yang dikendalikan. Keterbatasan ruang dan waktu untuk bekerja dan berpindah tempat juga menjadi alasan yang kuat untuk mengembangkan

pengendalian berbasis internet. Dunia perdagangan sampai dunia pendidikan pun sudah banyak dikembangkan dengan menerapkan teknologi internet ini. Hal ini disebabkan karena kemudahan dan kecepatan akses internet yang semakin meningkat dari tahun ke tahun dan perkembangan jaringan internet yang semakin menyebar di seluruh dunia. Sehingga sudah saatnya pengendalian robot pun dapat dilakukan melalui internet.

Penggunaan internet sebagai sarana pertukaran informasi dan data di Indonesia semakin meningkat. Sesuai dengan data statistik dari APJII (Asosiasi Penyelenggara Jasa Internet Indonesia) pengguna internet di Indonesia meningkat dari tahun ke tahun seperti terlihat pada gambar berikut :



Gambar 1.1 Jumlah Pengguna Internet di Indonesia Tahun 1998 – 2007 [1]

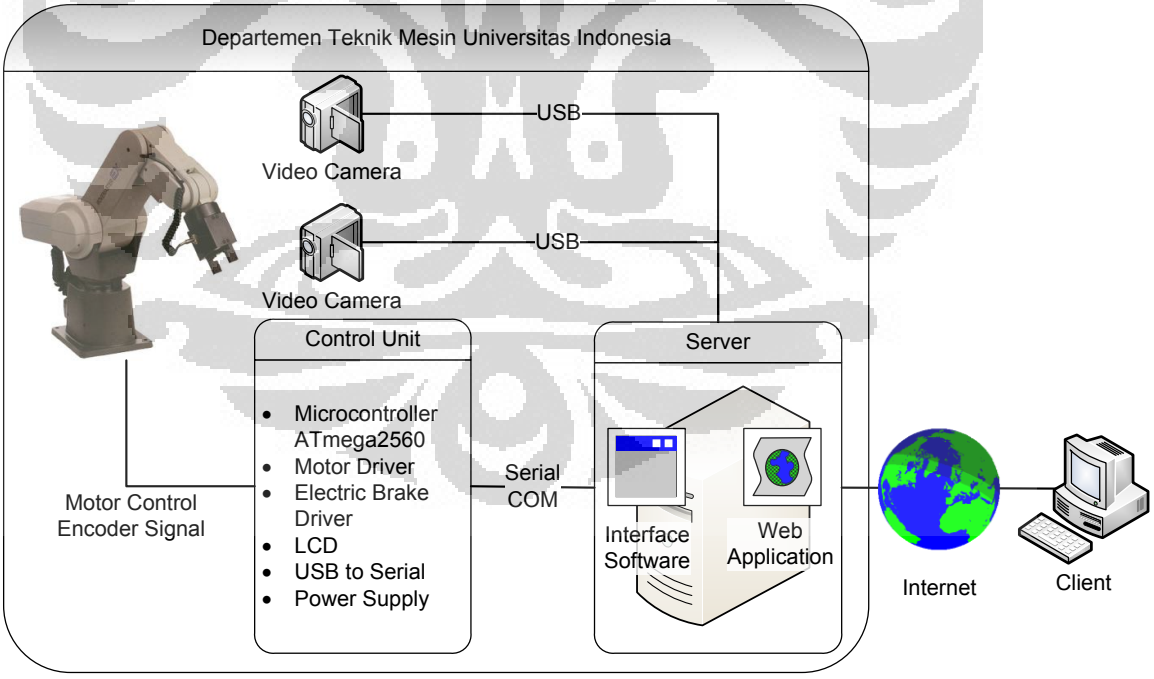
Peningkatan penggunaan internet di Indonesia inilah yang menjadi alasan penelitian kendali berbasis internet perlu dikembangkan dan diaplikasikan dalam dunia perindustrian di Indonesia.

1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk mengembangan prototipe sistem pengendalian robot industri dari jarak jauh berbasis internet dalam rangka pengembangan *intellegent manufacturing system* di Laboratorium Teknologi Manufaktur dan Otomasi Departemen Teknik Mesin Universitas Indonesia.

1.3 Perumusan Masalah

Sebuah sistem dikembangkan untuk mengendalikan lengan robot dengan 5 (lima) derajat kebebasan dari jarak jauh melalui internet. Untuk dapat memenuhi kriteria pengendalian berbasis jaringan internet, sistem harus memiliki fasilitas yang dapat mewakili panel kontrol (control unit) yang terdapat pada mesin sebenarnya dan juga harus dilengkapi dengan fasilitas pemantauan proses pergerakan robot secara visual agar pengguna dapat dengan mudah mempergunakan sistem tersebut. Sistem pengendalian berbasis jaringan internet ini mempunyai *architecture system* seperti berikut :



Gambar 1.2 Rancangan Sistem Keseluruhan

1.4 Pembatasan Masalah

Skripsi ini membahas mengenai komunikasi data antara jaringan internet, komputer, dan *microcontroller* untuk menggerakkan robot dari jarak jauh. Data yang dikirim adalah data pergerakan robot artikulasi dengan 5 derajat kebebasan yaitu robot Mitsubishi Movemaster RV-M1. *Microcontroller* yang digunakan adalah *microcontroller* AVR dari ATMEL *Corporation*.

1.5 Metodologi Penelitian

Metodologi penelitian ini adalah sebagai berikut:

1. Studi literatur terhadap artikel pengendalian berbasis robot
2. Studi mekanisme robot Mitsubishi Movemaster RV-M1.
3. Pemodelan sistem arsitektur pengendalian robot berbasis *web*.
4. Pengembangan sistem pengendalian robot.
5. Eksperimen rancangan sistem arsitektur pengendalian robot.
6. Pengujian & analisa sistem kendali robot.

1.6 Sistematika Penulisan

BAB 1. PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang masalah, perumusan masalah, pembatasan masalah, tujuan penelitian, metodologi penelitian dan sistematika penulisan.

BAB 2. MEKANIKA DAN KONTROL ROBOT INDUSTRI

Bab ini menjelaskan tentang pengenalan berbagai tipe robot industri dan teori – teori yang dipakai dalam pengendalian robot industri.

BAB 3. TINJAUAN SISTEM MEKANIKAL ROBOT

Bab ini menjelaskan tentang tijauan sistem mekanikal robot Movemaster RV-M1 dan spesifikasinya sebagai dasar perancangan sistem elekronik dan perangkat lunak.

BAB 4. PENGEMBANGAN PERANGKAT KERAS SISTEM PENGENDALI

Bab ini menjelaskan tentang perancangan dan pembuatan dari sistem elektronika yang dipakai dalam pengendalian robot berbasis *web* ini.

BAB 5. PENGEMBANGAN SISTEM PERANGKAT LUNAK SISTEM PENGENDALI

Bab ini menjelaskan tentang perancangan dan pembuatan dari sistem perangkat lunak dan algoritma yang dipakai dalam pengendalian robot berbasis *web* ini.

BAB 6. PENGEMBANGAN SISTEM KOMUNIKASI DENGAN *WEB* SERVER

Bab ini menjelaskan tentang pengembangan komunikasi dengan *web* server yang dikembangkan untuk sistem kendali robot berbasis *web* ini.

BAB 7. PENGUJIAN & ANALISA SISTEM

Bab ini menjelaskan tentang analisa sistem secara umum untuk mengetahui kelebihan dan kekurangan dari sistem ini.

BAB 8. KESIMPULAN & SARAN PENELITIAN LEBIH LANJUT

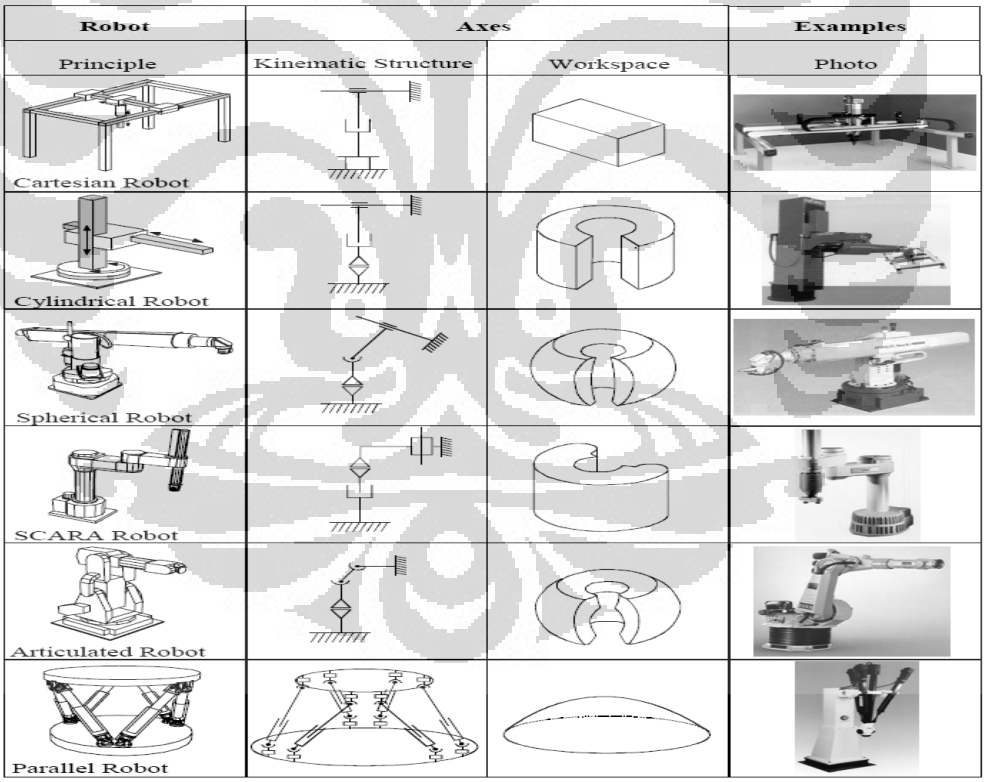
Bab ini menjelaskan kesimpulan penelitian dan saran untuk penelitian selanjutnya.

BAB 2
MEKANIKA DAN KONTROL ROBOT INDUSTRI

Bagian ini menjelaskan hal – hal yang berhubungan dengan mekanika dan kontrol robot industri pada umumnya.

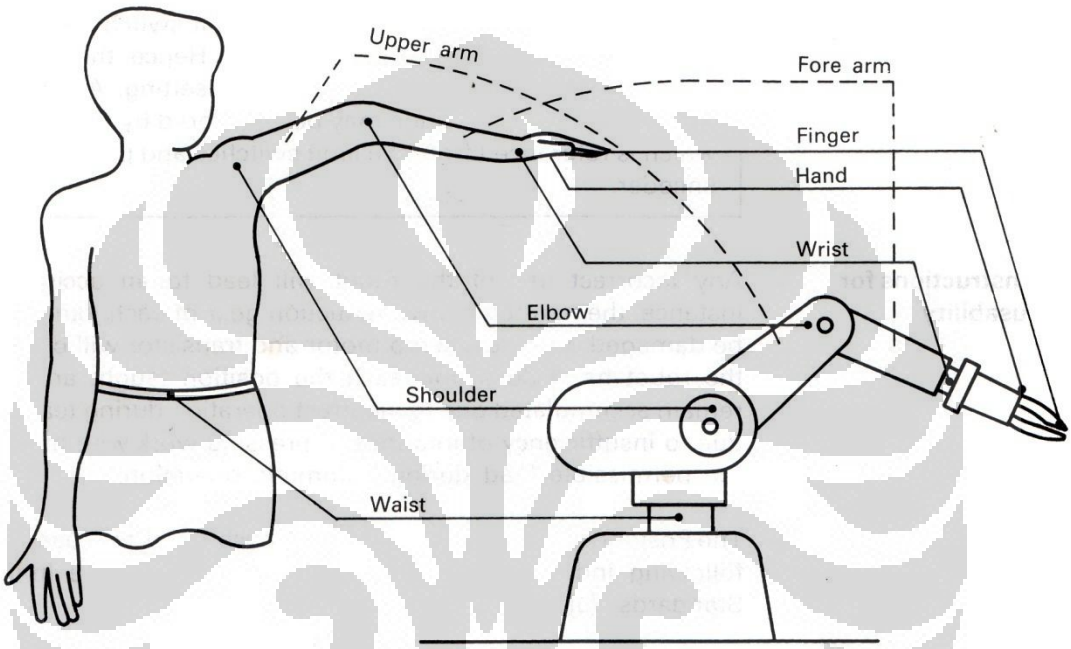
2.1 Konstruksi Robot

Definisi robot industri menurut standar ISO 8373 (*Manipulating Industrial Robot*) adalah : *"An automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation application".*



Gambar 2.1 Klasifikasi Konstruksi Mekanik Robot Industri pada ISO 8373 [2]

Sesuai dengan gambar pada ISO 8373, maka robot yang dipakai dalam penelitian ini merupakan robot dengan tipe artikulasi dengan 5 derajat kebebasan, namun dalam ukuran yang lebih kecil sehingga disebut *industrial micro-robot*. Pengembangan robot dengan 5 derajat kebebasan didasari oleh pergerakan yang dapat dihasilkan oleh manusia.



Gambar 2.2 Dasar Perancangan Robot Artikulasi [3]

Dengan kemampuan yang menyerupai lengan manusia ini maka robot – robot tersebut dapat digunakan di pabrik sebagai pengganti pekerja manusia.

Robot artikulasi dapat dilengkapi dengan berbagai peralatan pada bagian *end-effector*. Mulai dari *Spot Welding, Laser, Gripper, atau Machining Tool*, hal ini yang menyebabkan penggunaanya sangat luas pada dunia industri. Beberapa robot bahkan sudah mempunyai kecerdasan buatan yang diterapkan didalam *controller* robot tersebut. Dengan adanya kecerdasan buatan, robot mampu menganalisa dan merencanakan pergerakan dirinya tanpa adanya bantuan dari manusia.

2.2 Derajat Kebebasan

Derajat kebebasan merupakan paket pergerakan yang mampu dilakukan oleh suatu sistem mekanik. Derajat kebebasan atau sering disebut dengan *DOF (Degree of Freedom)* dihitung berdasarkan kemampuan perubahan orientasi sistem mekanik tersebut. Dalam perancangan robot DOF ditentukan berdasarkan kegunaan robot tersebut. Sebuah robot dapat dikatakan *redundant* apabila mempunyai DOF yang melebihi DOF minimal yang diperlukan untuk melakukan tugasnya. Untuk robot yang *redundant* perlu dirancang suatu algoritma pergerakan yang bersifat *task priority*.

2.3 Pengendalian Menggunakan Metode Forward Kinematics

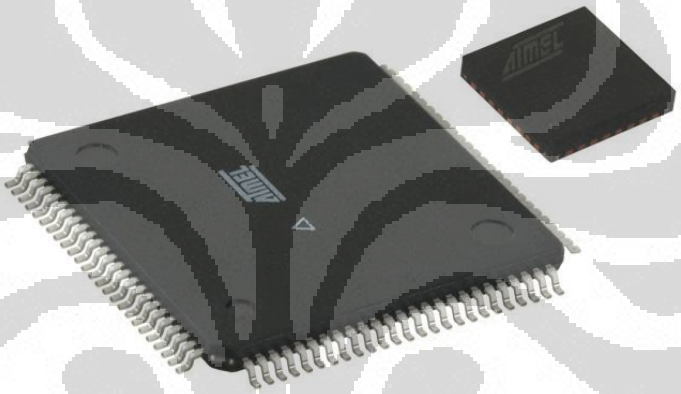
Forward kinematics merupakan suatu perhitungan pergerakan dan orientasi robot berdasarkan parameter pergerakan dari masing – masing *axis*-nya. Aplikasi *forward kinematics* sering dilakukan dipabrik dengan cara melatih robot untuk melaksanakan tugasnya. Parameter pergerakan robot ditentukan oleh *operator* yang menugaskan robot melakukan tugas tertentu. Pada penelitian ini pengendalian robot menggunakan metode forward kinematics dilakukan oleh *client* dengan memasukkan parameter pergerakan robot pada *web*.

2.4 Pengendalian Menggunakan Metode Inverse Kinematics

Inverse kinematics merupakan suatu perhitungan pergerakan dan orientasi robot berdasarkan posisi yang ingin dicapai oleh robot. Posisi tersebut yang akan menentukan parameter pergerakan dari masing – masing *axis*-nya. Untuk menerapkan *inverse kinematics* diperlukan adanya algoritma khusus & model matematika untuk menentukan parameter pergerakan yang diinginkan. Aplikasi *inverse kinematics* sering digunakan pada robot yang memerlukan kecerdasan dalam melaksanakan tugasnya. Didalam penelitian ini pergerakan *inverse kinematic* dilakukan dengan metode *cursor based,* sehingga robot harus harus mampu melakukan pergerakan hanya berdasarkan penunjukan koordinat yang dilakukan oleh *client* melalui layar GUI (*Graphic User Interface*) yang terlihat pada layar monitor *client*.

2.5 Embedded System

Embedded system merupakan suatu sistem komputer yang didedikasikan untuk menjalankan satu atau berbagai macam tugas dan biasanya melakukan proses data secara *real-time.* Sistem ini sering kali memiliki pengendali seperti *microcontroller* atau *digital signal processor* (DSP). Penggunaan sistem ini pada kendali robot berbasis *web* sangat bermanfaat karena kemudahan pemakaian dan harganya yang cukup terjangkau. Instruksi perintah diprogram pada suatu memori yang kapasitasnya tidak terlalu besar, program dapat dimasukkan dengan menggunakan komputer dan programmer yang disediakan oleh produsennya.



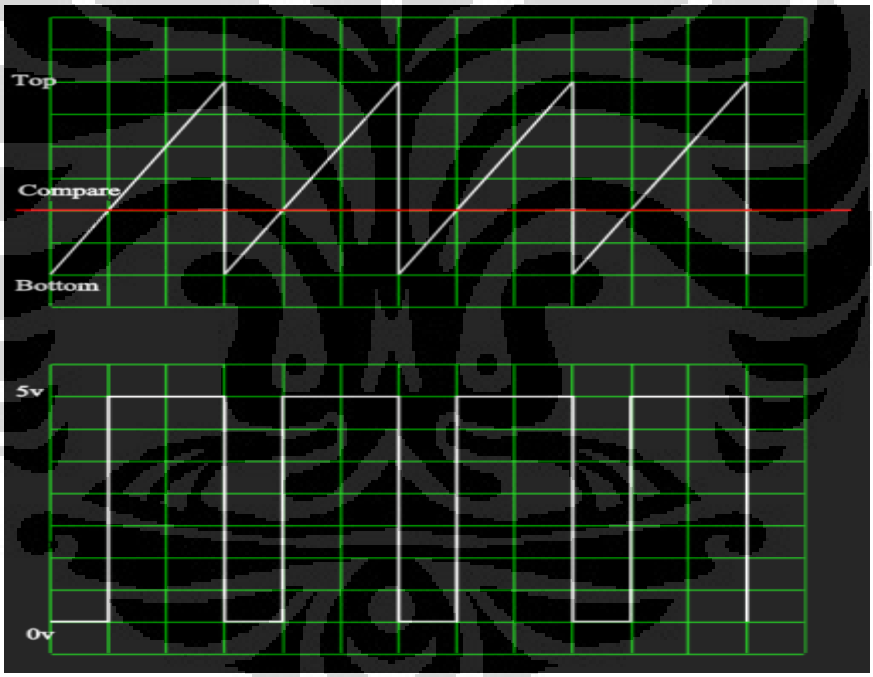
Gambar 2.3 Paket Microcontroller [4]

2.6 Sensor pada Robot Industri

Peran *sensor* dalam pengendalian robot industri sangat besar karena *sensor* merupakan sebuah perangkat yang dapat mendeteksi perubahan pada robot baik secara fisik, kimia ataupun biologi. Perubahan tersebut merupakan *input* bagi *sensor* yang kemudian diubah menjadi sinyal *output*. Sinyal yang didapat dari sensor ini kemudian dapat diolah pada *controller* robot sebagai data untuk melakukan pengendalian. Robot industri pada umumnya menggunakan *sensor* elektromekanikal seperti *encoder,* dan *limit switch.* Layaknya manusia *sensor* merupakan pancaindra bagi robot, sehingga keberadaan sensor sangat penting.

2.7 Actuator Robot Industri

Aktuator merupakan perangkat untuk menggerakkan sebuah sistem mekanik. Pada robot industri terdapat berbagai macam actuator seperti *hydrolic actuator*, *pneumatic actuator*, dan *electric actuator*. Robot Movemaster RV-M1 menggunakan *actuator* berupa motor DC (Direct Current) yang merupakan *electric actuator*. Pengendalian motor DC yang paling umum digunakan adalah menggunakan metode PWM (*Pulse Width Modulation*). Metode PWM merupakan pengendalian sinyal secara digital dengan mengatur lebar pulsa yang dikeluarkan. Lebar pulsa yang dikeluarkan ini dapat mengatur kecepatan motor DC apabila dikombinasikan dengan *driver* motor. Pada *microcontroller* sudah terdapat fitur khusus untuk menghasilkan sinyal PWM yang mempunyai output seperti gambar berikut :



Gambar 2.4 Sinyal PWM [5]

2.8 Bahasa Pemrogramam Sistem Kontrol

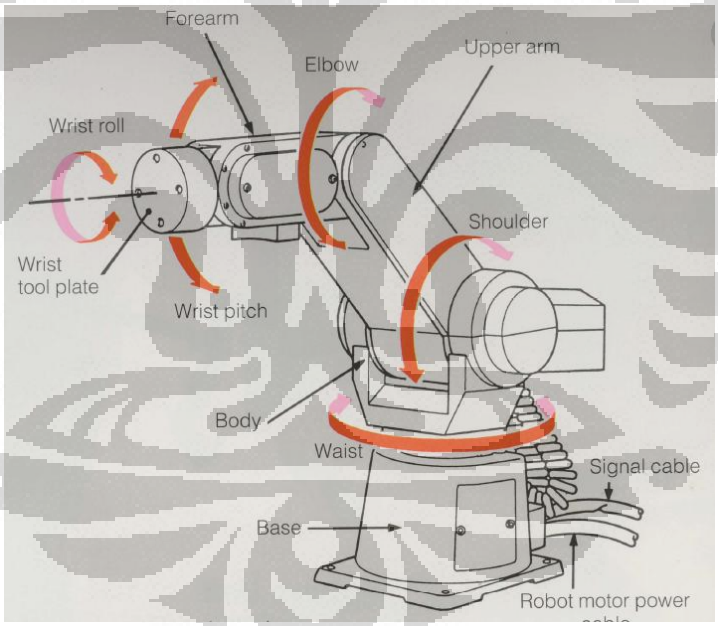
Dalam pengembangan perangkat lunak sebuah sistem kontrol digunakan bahasa pemrograman yang berfungsi untuk mengatur informasi antara *input*, proses dan *output* dari sistem yang dikendalikan. Salah satu bahasa pemrograman yang banyak dipakai adalah bahasa C. Bahasa C banyak digunakan untuk pemrograman sistem seperti aplikasi *operating system* dan *embedded system* karena kemampuannya untuk mengakses berbagai *address* perangkat keras serta efisiensi program yang meringankan kerja sistem. Bahasa C dipakai pada pengembangan robot berbasis *web* ini terutama pada pemrograman *microcontroller* dan *software* komunikasi dari komputer ke *microcontroller* karena kelebihan – kelebihan tersebut.

BAB 3
TINJAUAN SISTEM MEKANIKAL ROBOT

Tinjauan sistem mekanikal robot yang akan digunakan ini bertujuan untuk menentukan pemilihan perangkat elektronik yang tepat.

3.1 Desain Mekanikal Robot Movemaster RV-M1

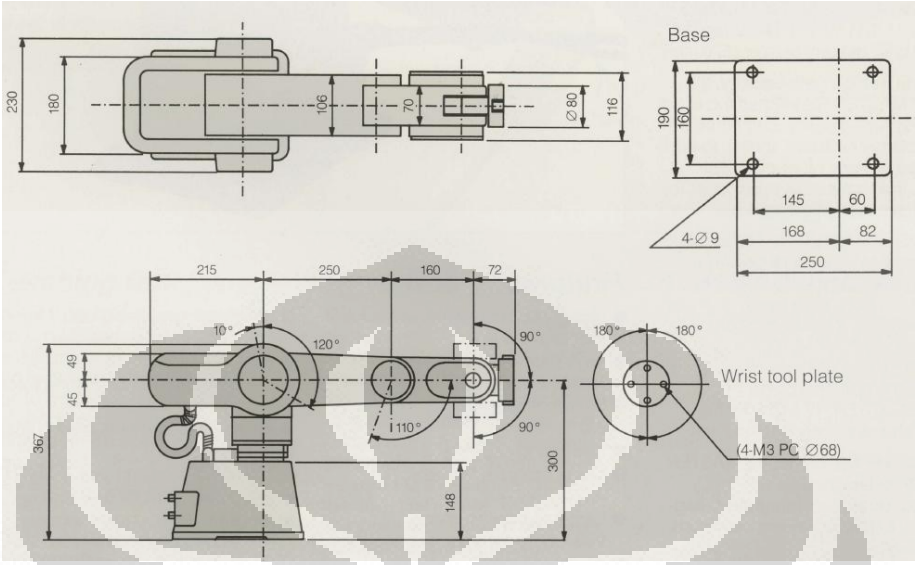
Robot yang digunakan dalam pengembangan sistem kontrol robot berbasis *web* ini adalah robot artikulasi dengan 5 derajat kebebasan yaitu Mitsubishi Movemaster RV-M1. Robot ini mempunyai desain mekanikal seperti yang tertera pada gambar 3.1.



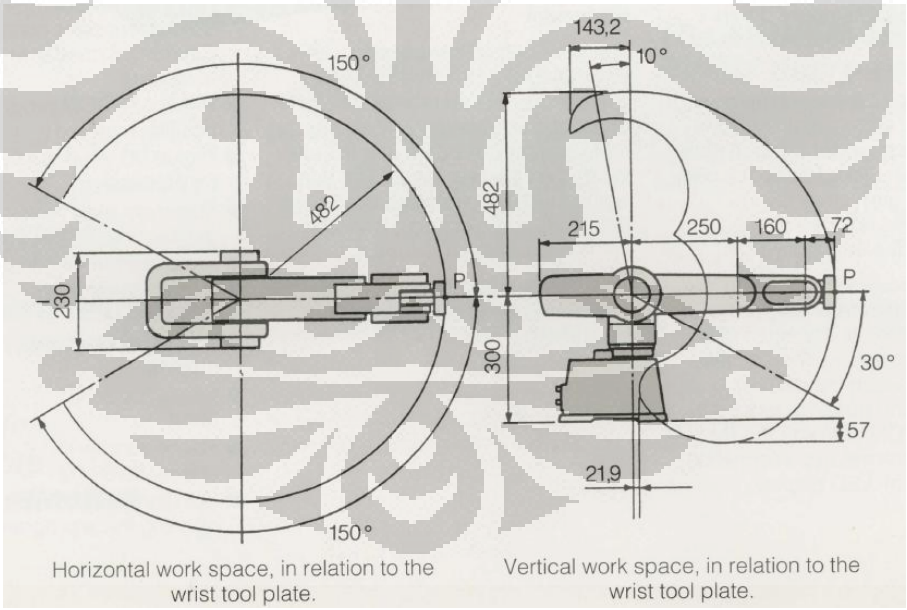
Gambar 3.1 Desain Mekanikal Mitsubishi Movemaster RV-M1 [3]

Desain mekanikal robot seperti ini merupakan jenis robot *manipulator* dengan tipe Artikulasi. Setiap robot manipulator mempunyai keterbatasan dalam melakukan pergerakan bergantung pada desain mekanikalnya sendiri. Keterbatasan pergerakan ini dapat didata dengan membuat diagram pergerakan robot atau biasa disebut dengan

working space diagram. Dimensi dan *working space* robot Mitsubishi Movemaster RV-M1 ini dapat dilihat pada gambar 3.2 dan 3.3.



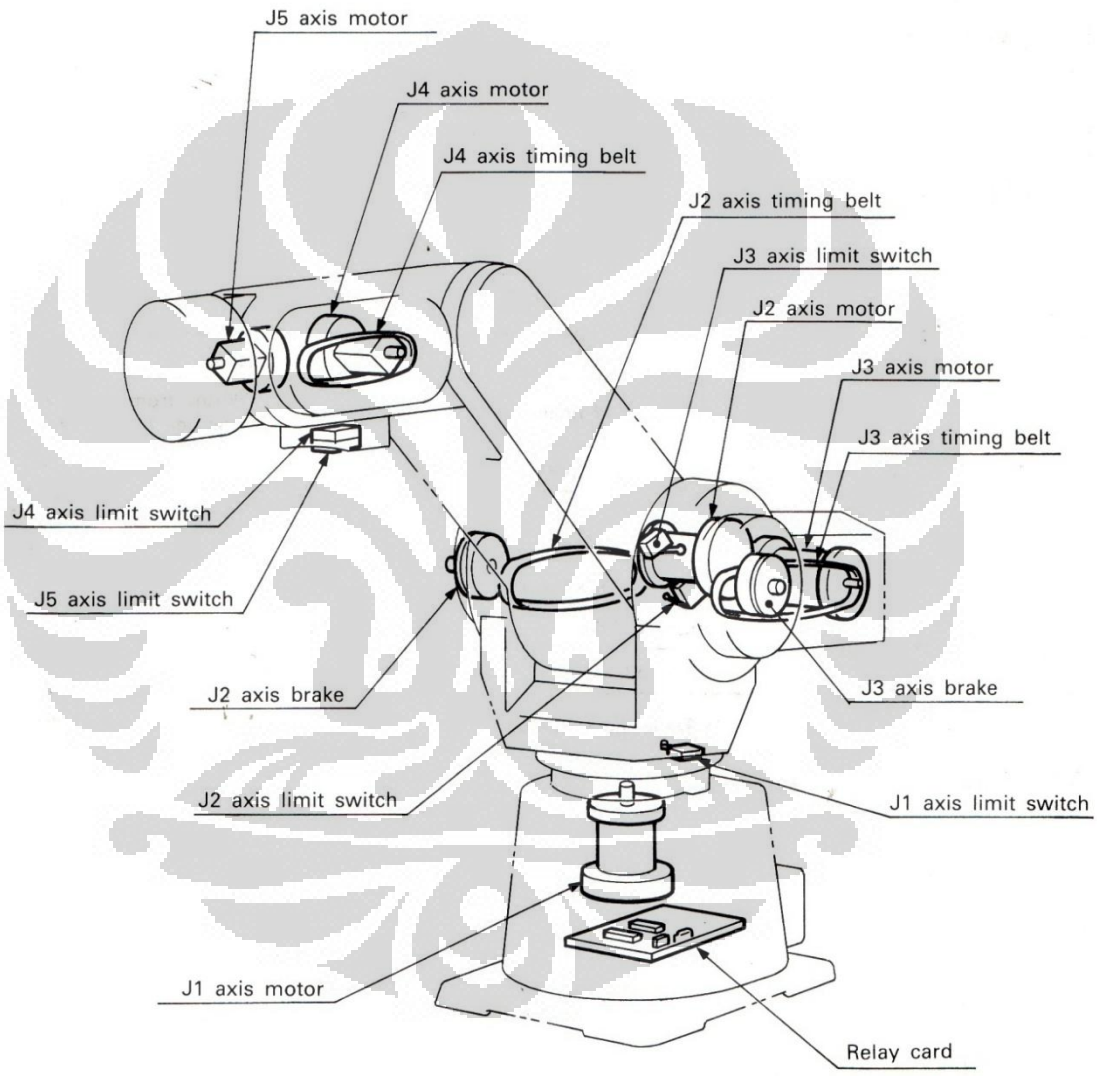
Gambar 3.2 Dimensi Mitsubishi Movemaster RV-M1 [3]



Gambar 3.3 Working Space Mitsubishi Movemaster RV-M1 [3]

3.2 Sistem Penggerak Robot

Robot movemaster RV-M1 memiliki sistem penggerak yang cukup sederhana pada masing – masing *axis*-nya. Sistem penggerak ini cukup akurat untuk menggerakkan robot ke posisi yang diinginkan. Posisi alat perangkat sistem penggerak robot dapat dilihat pada gambar berikut :



Gambar 3.4 Sistem Penggerak Mitsubishi Movemaster RV-M1 [3]

3.2.1 Motor DC (*Direct Current*)

Robot Movemaster RV-M1 mempunyai penggerak bertenaga motor DC untuk menggerakkan masing – masing axisnya, berikutlah salah satu gambar dari motor DC tersebut :

Gambar 3.5 Motor DC pada Mitsubishi Movemaster RV-M1

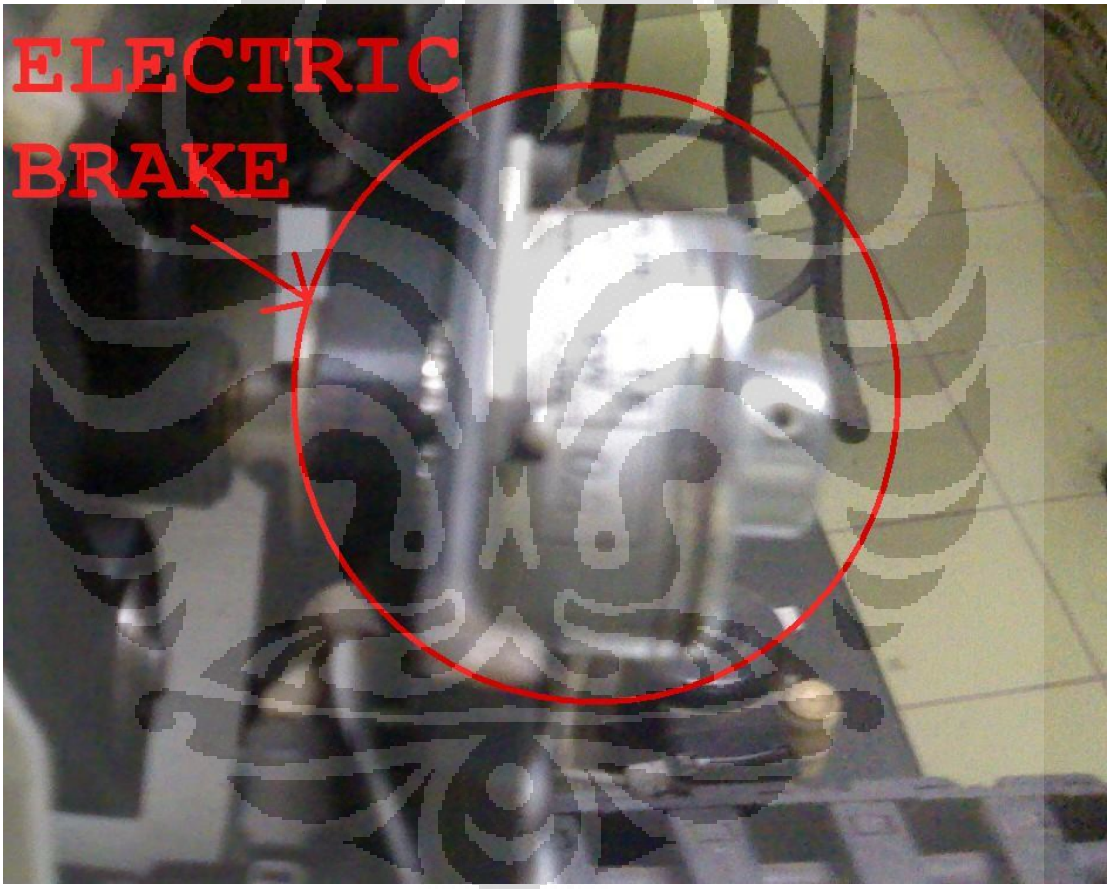
DC motor yang digunakan adalah Sanyo Denki DC Servo Motor dilengkapi dengan *encoder*. Spesifikasi yang tertera pada *name plate* motor DC tersebut adalah :

1. 2.3 Ampere.
2. 24 Volt.
3. 28 Watt.

Motor DC yang digunakan pada setiap axis mempunyai spesifikasi yang serupa namun pada setiap axis motor tersebut di hubungkan dengan gearbox dengan konfigurasi yang berbeda bergantung pada *load* yang diterimanya.

3.2.2 Rem Elektronik (Electric Brake)

Axis 2 dan *Axis* 3 dilengkapi dengan *electric brake* karena kedua *axis* tersebut merupakan *axis* yang memiliki *load* terbesar. *Electric brake* merupakan perangkat elektromekanikal yang bekerja dengan memanfaatkan prinsip elektromagnet untuk menghubungkan atau memisahkan poros yang berputar dengan *chassis* robot yang diam. *Electric brake* yang digunakan pada robot Movemaster RV-M1 ini merupakan tipe NC (Normally Closed) dan membutuhkan tegangan 12V untuk merubahnya menjadi *open state*. Berikut adalah gambar dari *electric brake* tersebut :

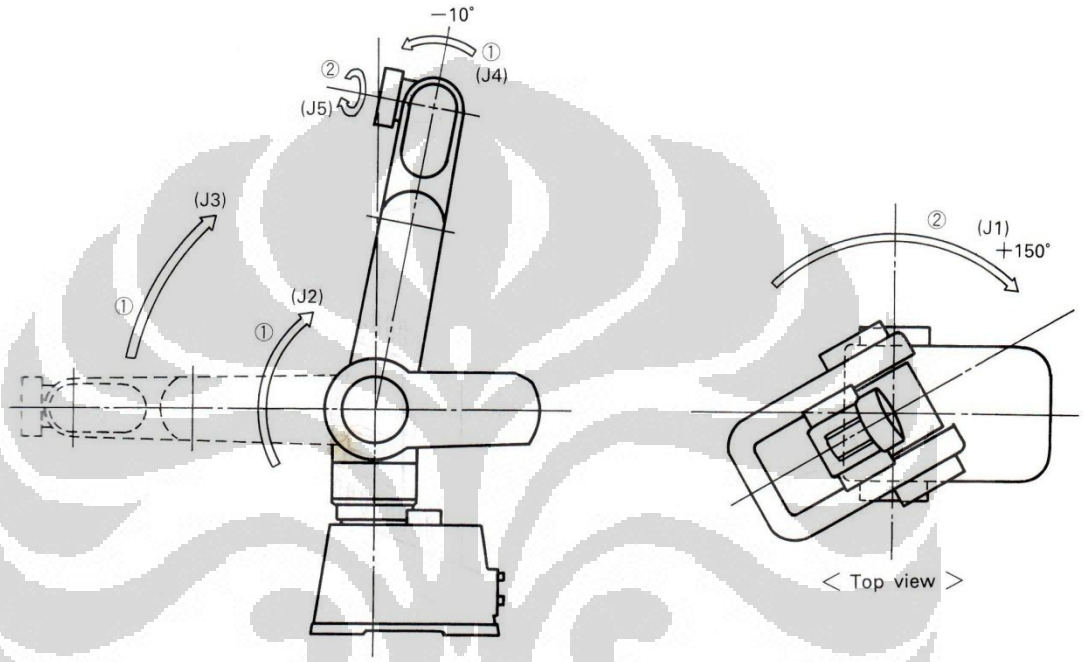


Gambar 3.6 *Electric Brake* pada Mitsubishi Movemaster RV-M1

Fungsi *electric brake* pada kedua *axis* tersebut sangat vital karena motor DC pada kedua axis tersebut tidak memiliki torsi diam yang cukup untuk menahan kedua axis tersebut sehingga dibantu oleh *electric brake*.

3.3 Posisi Default Robot

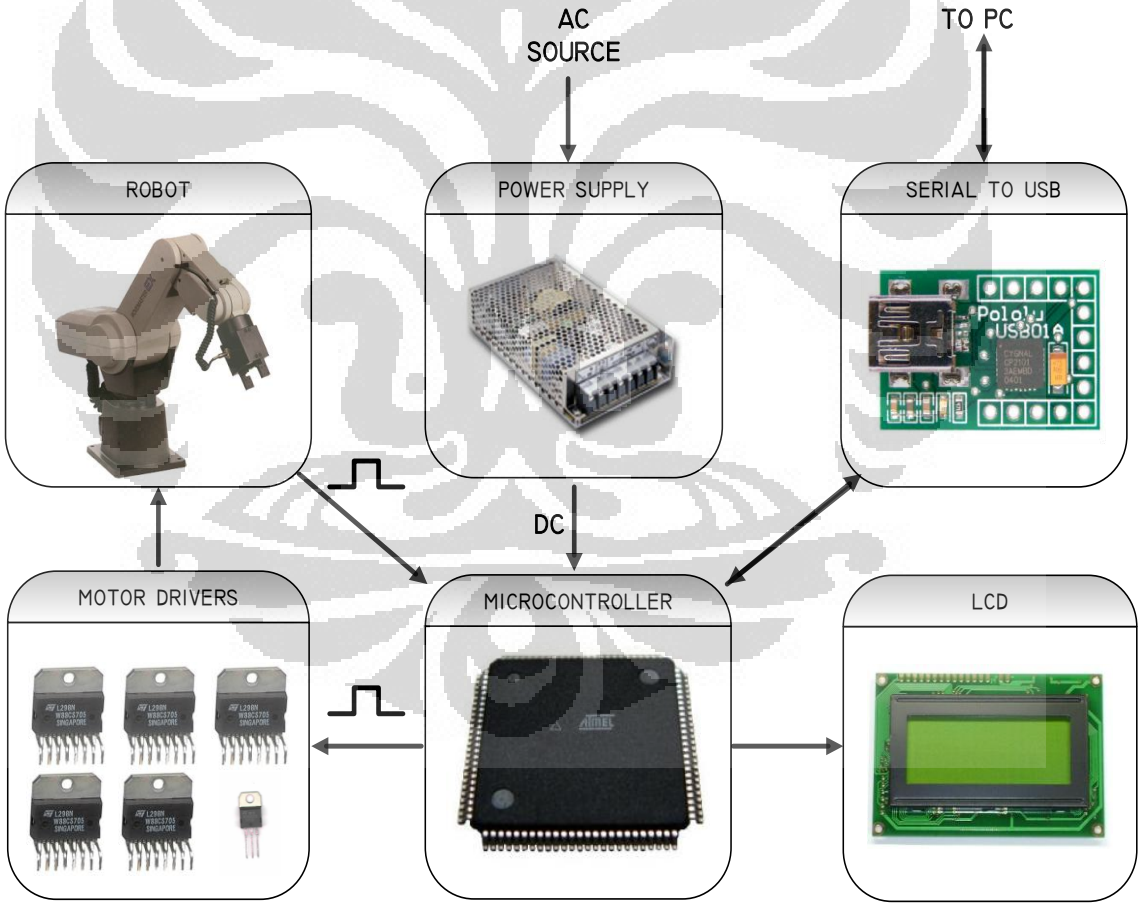
Posisi default robot merupakan posisi yang sangat penting karena seringkali kita membutuhkan acuan dalam menentukan pergerakan robot. Sesuai dengan buku petunjuk robot Movemaster RV-M1, posisi *default* robot dapat dilihat pada gambar berikut :



Gambar 3.7 Posisi Default Robot [3]

BAB 4
PENGEMBANGAN PERANGKAT KERAS SISTEM PENGENDALI

Berdasarkan tinjauan sistem mekanikal maka dilakukan pengembangan perangkat keras untuk pengendalian robot berbasis *web* ini. Perangkat keras yang digunakan dirancang untuk memenuhi kebutuhan pengendalian sistem mekanik. Berikut adalah diagram rancangan sistem pengendali :



Gambar 4.1 Rancangan Sistem Pengendali

4.1 Sensor

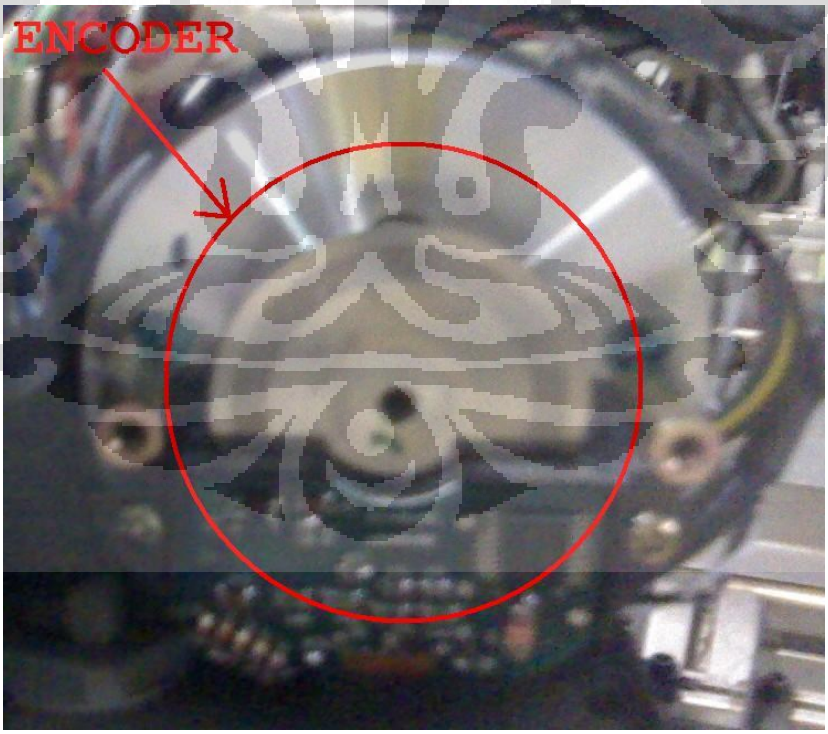
Robot Movemaster RV-M1 mempunyai beberapa sensor elektromekanikal. Sensor-sensor ini yang berfungsi untuk mendeteksi pergerakan robot dan mengirimkan data ke *microcontroller*.

4.1.1 Encoder

Encoder merupakan perangkat *electromechanical* yang berfungsi sebagai alat *feedback* berupa pulsa *digital* yang sangat bermanfaat dalam pengendalian motor DC. Setiap motor penggerak pada robot Movemaster RV-M1 memiliki *encoder* dengan spesifikasi sebagai berikut :

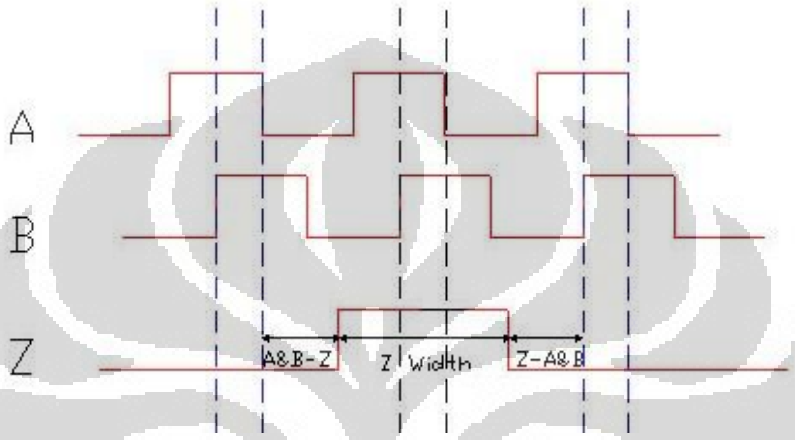
1. *Incremental Encoder*.
2. *Digital Voltage* 5V.
3. 200 CPR.

Berikut adalah gambar *encoder* yang terpasang pada motor DC :



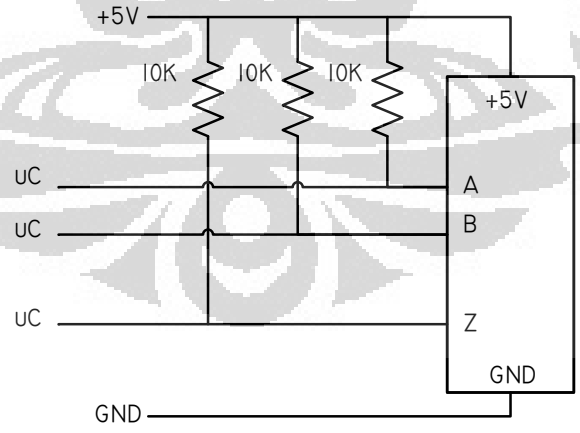
Gambar 4.2 *Encoder* pada Mitsubishi Movemaster RV-M1

Setiap encoder mengeluarkan 3 fasa yaitu A, B dan Z, setiap fasa ini mengeluarkan *timing* pulsa digital yang berbeda – beda, pada penelitian ini digunakan satu fasa saja untuk kendali motor DC berdasarkan sudut robot yaitu fasa A. Untuk fasa B dan Z dapat digunakan untuk pengembangan lebih lanjut. Sebuah encoder mempunyai *digital output* sebagai berikut :



Gambar 4.3 Sinyal *Encoder* [6]

Untuk *interface* pada encoder dengan microcontroller dapat digunakan contoh rangkaian elektronika sebagai berikut:



Gambar 4.4 *Interface* Encoder ke Microcontroller

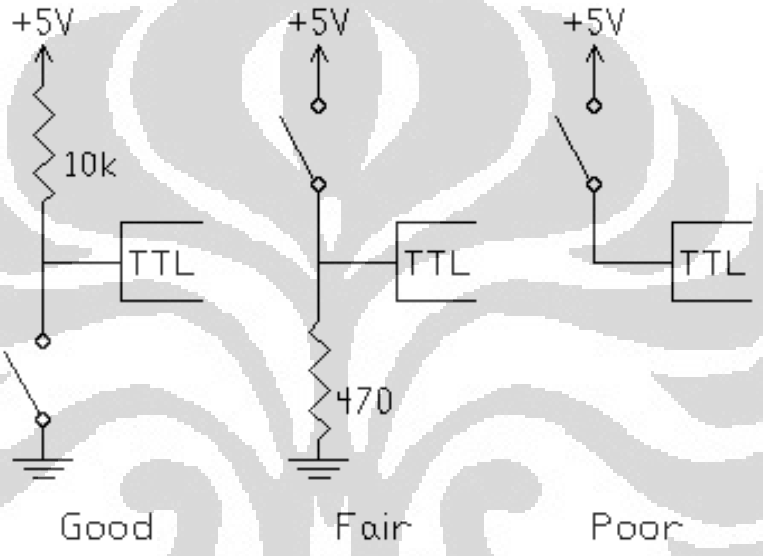
4.1.2 Limit Switch

Robot Movemaster RV-M1 memiliki 5 *limit switch* sebagai *feedback* posisi robot. *Limit switch* pada penelitian ini digunakan sebagai *feedback* pada program pencari posisi *default* robot sesuai dengan *datasheet* robot Movemaster RV-M1. Setiap *axis* memiliki 1 limit switch sebagai *sensor* posisi *default*. *Limit switch* yang digunakan adalah tipe *tactile* dengan kondisi NC (Normally Closed). Berikut adalah gambar dari *limit switch* tersebut :

Gambar 4.5 Limit Switch yang Digunakan Robot Movemaster RV-M1

Gambar 4.6 Limit Switch pada Mitsubishi Movemaster RV-M1

Limit switch merupakan perangkat *electromekanikal* yang bekerja dengan memanfaatkan sentuhan objek lain. Sentuhan dari objek lain dalam penelitian ini adalah lengan robot dapat merubah kondisi *limit switch* dari posisi NC ke *open state*. Pada posisi open arus listrik yang disuplai ke *microcontroller* terputus, kondisi ini dapat dimanfaatkan sebagai kondisi pada *microcontroller*. Untuk *interface limit switch* ke *microcontroller* dapat dilakukan dengan cara memberikan *pullup resistor* seperti gambar berikut :



Gambar 4.7 Interface Limit Switch ke Microcontroller [7]

Tujuan pemasangan *pullup resistor* adalah menghindari kondisi mengambang pada input *microcontroller.* Microcontroller biasanya mempunyai default *input* bernilai 1 (*high*) sehingga apabila ada sedikit perubahan dari luar (ketika sensor dipasang) maka microcontroller bisa saja berubah menjadi kondisi 0 (*low*), namun sulit untuk kembali ke kondisi *high* sehingga perlu dipasang *pullup resistor* untuk mengembalikan kondisi *input* tersebut. *Pullup resistor* bekerja dengan meningkatkan arus pada rangkaian input, pada saat *limit switch* maupun *encoder* tidak memberikan keluaran maka kondisi *input* adalah 1 sedangkan begitu mendapat *trigger* kondisi *input* menjadi 0.

4.2 Kendali Digital

Sistem kendali yang dikembangkan untuk mengendalikan robot Movemaster RV-M1 adalah kendali *digital*. Sistem kendali *digital* digunakan karena kemudahan untuk menerapkan logika kendali dan mempermudah pengembangan sistem lebih lanjut.

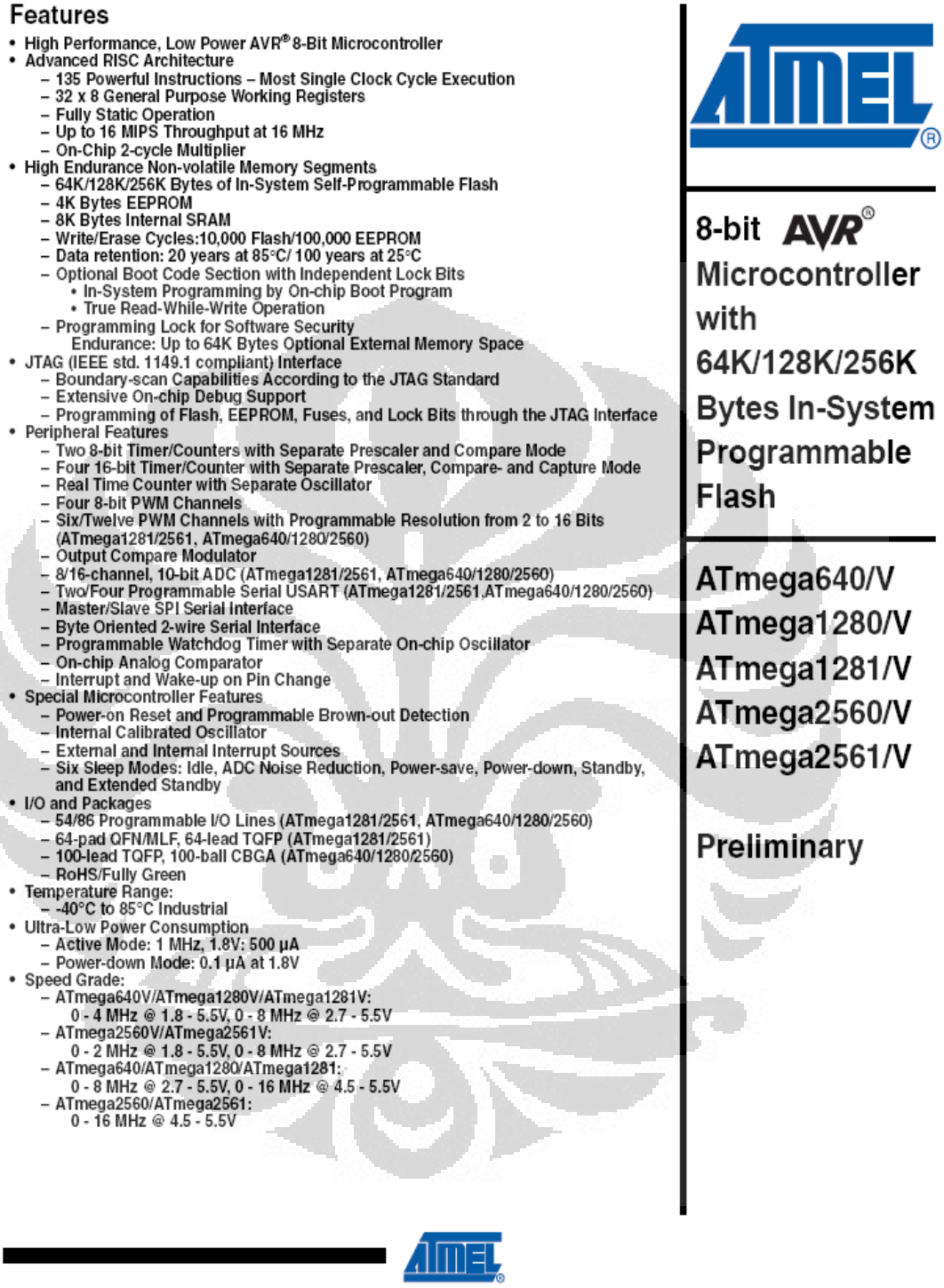
4.2.1 Microcontroller

Microcontroller merupakan perangkat elektronika yang menyerupai sebuah komputer namun berukuran lebih kecil dan biasanya berbentuk IC. Layaknya sebuah komputer, *microcontroller* merupakan sebuah CPU yang memiliki perangkat pelengkap seperti *crystal oscillator, timer, serial communication, digital I/O dan analog I/O*. *Microcontroller* menggunakan *memory* dengan kapasitas kecil yang dapat diprogram ulang dengan komputer melalui bahasa pemrograman yang bervariasi. *Microcontroller* dapat digunakan untuk mengendalikan berbagai perangkat elektronika dengan logika pemrograman, sehingga dalam dalam penelitian kontrol robot berbasis *web* ini digunakan *microcontroller* sebagai pengontrol robot. Tipe *microcontroller* yang dipilih adalah ATmega2560 dari Atmel. Pemilihan microcontroller jenis ini didasari dari kebutuhan *controller* dari robot movemaster RV-M1 yaitu :

1. Mempunyai minimal 5 PWM channel.
2. Mempunyai minimal 5 Interrupt channel.
3. Mampu melakukan komunikasi serial.
4. Kapasitas memory yang besar.
5. Kecepatan eksekusi yang tinggi.

Tabel 4.1 Spesifikasi ATmega2560 [8]

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega640	64KB	4KB	8KB	86	12	4	16
ATmega1280	128KB	4KB	8KB	86	12	4	16
ATmega1281	128KB	4KB	8KB	54	6	2	8
ATmega2560	256KB	4KB	8KB	86	12	4	16
ATmega2561	256KB	4KB	8KB	54	6	2	8



Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256K Bytes of In-System Self-Programmable Flash
  - 4K Bytes EEPROM
  - 8K Bytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    Endurance: Up to 64K Bytes Optional External Memory Space
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561,ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1 MHz, 1.8V: 500 µA
  - Power-down Mode: 0.1 µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega2560V/ATmega2561V:
    0 - 2 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
  - ATmega2560/ATmega2561:
    0 - 16 MHz @ 4.5 - 5.5V

8-bit AVR® Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

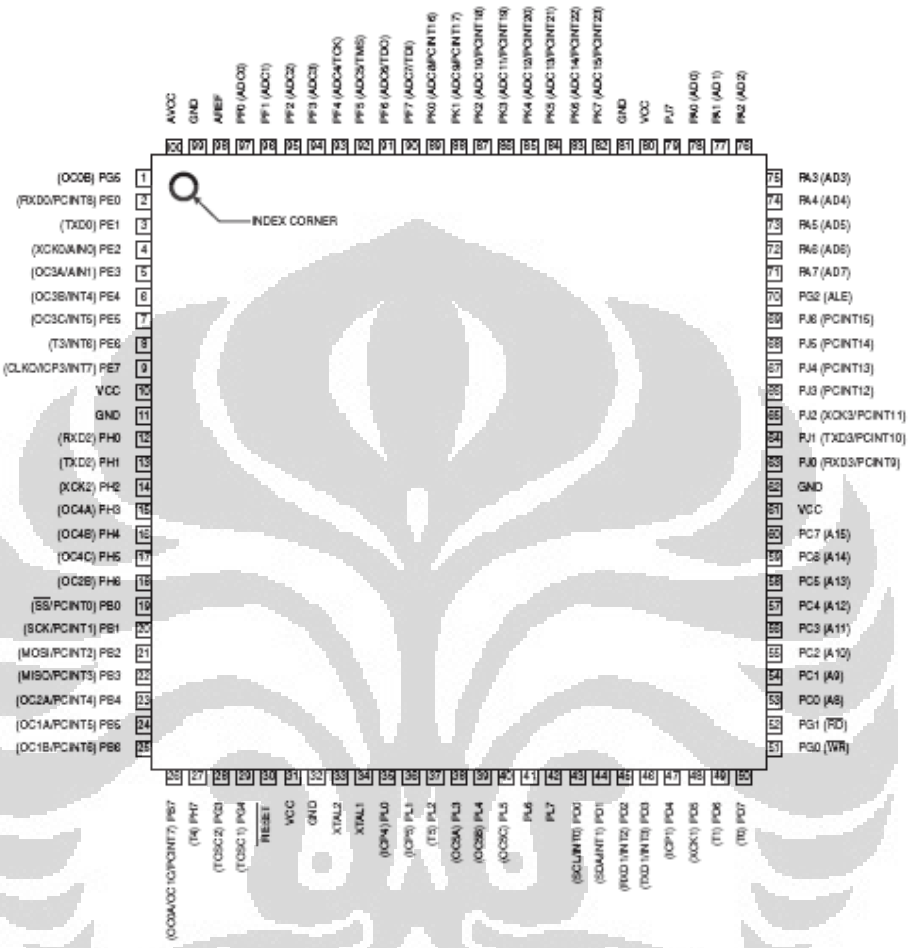
Preliminary

Gambar 4.8 Spesifikasi Lengkap ATmega2560 [8]

Kecepatan proses data bergantung pada penggunaan *oscillator* pada *microcontroller*. *Oscillator* merupakan sebuah perangkat yang menghasilkan frekuensi tertentu untuk menjadi *clock source* bagi microcontroller. *Clock source* pada *microcontroller* menjadi patokan waktu bagi *microcontroller* untuk melakukan sebuah instruksi. *Microcontroller* ATMEL pada umumnya mempunyai karakter satu clock satu instruksi dan memiliki *oscillator internal* yang terdapat didalam *microcontroller*. *Oscillator internal* yang terdapat pada *microcontroller* dirasa kurang cukup memadai untuk aplikasi kontrol robot berbasis *web* ini karena frekuensi hanya 1Mhz. Dengan frekuensi 1Mhz microcontroller hanya mampu mencapai kecepatan 1MIPS (*Million Instruction per Second*). Microcontroller ATmega2560 pada penelitian ini menggunakan *clock source* 16Mhz sehingga mampu mencapai kecepatan 16MIPS. Kecepatan ini dianggap cukup untuk dipakai dalam aplikasi pengendalian robot berbasis *web* ini. Berikut gambar adalah *oscillator* yang dipakai pada *microcontroller* ATmega 2560 :

Gambar 4.9 *Oscillator* 16Mhz yang Dipakai pada ATmega2560 [9]

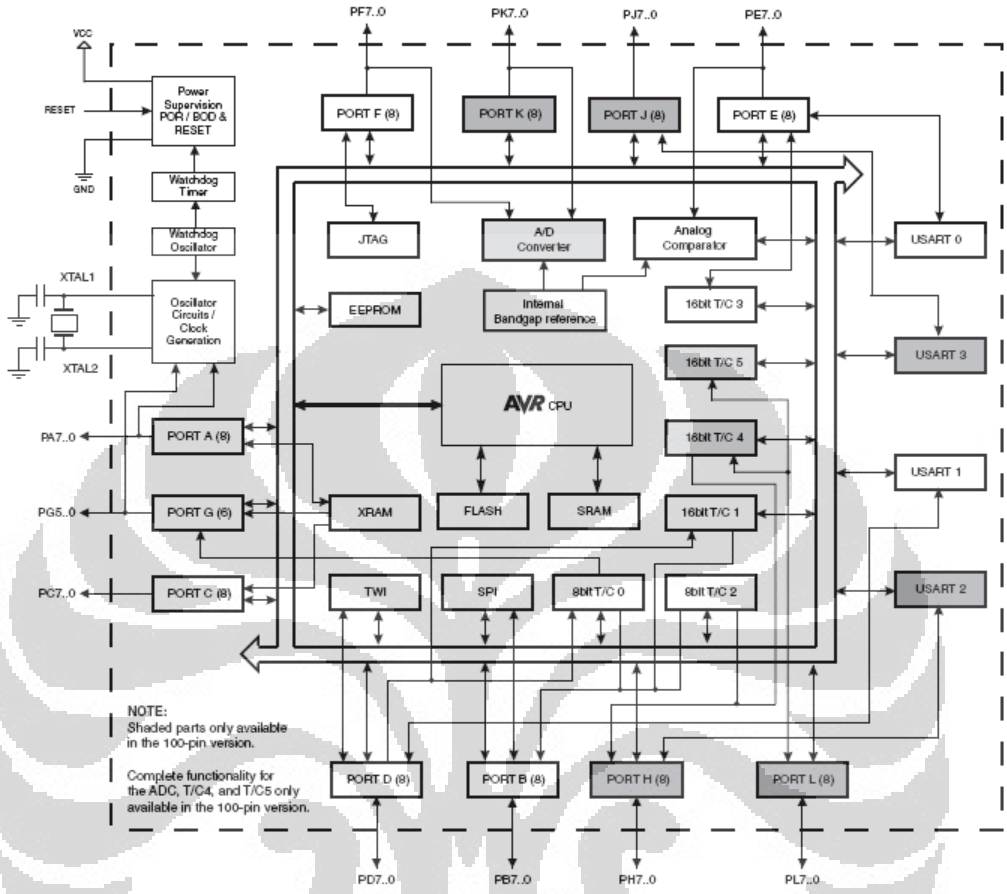
Microcontroller ATmega2560 mempunyai diagram *pinout* seperti gambar dibawah ini :



Gambar 4.10 *Pinout* ATmega2560 [8]

Seperti terlihat dalam gambar diatas fitur yang dimiliki ATmega2560 sudah cukup memadai untuk mengontrol robot movemaster RV-M1. Fitur yang ada tidak semua dipakai karena disengaja untuk pengembangan robot lebih lanjut. Setiap *microcontroller* mempunyai sistem arsitektur yang unik, ATmega2560 merupakan microcontroller 8-Bit RISC (*Reduced Instruction Set Computing*), sehingga aplikasinya cukup sederhana karena ada fungsi – fungsi pemrograman yang sudah disederhanakan.

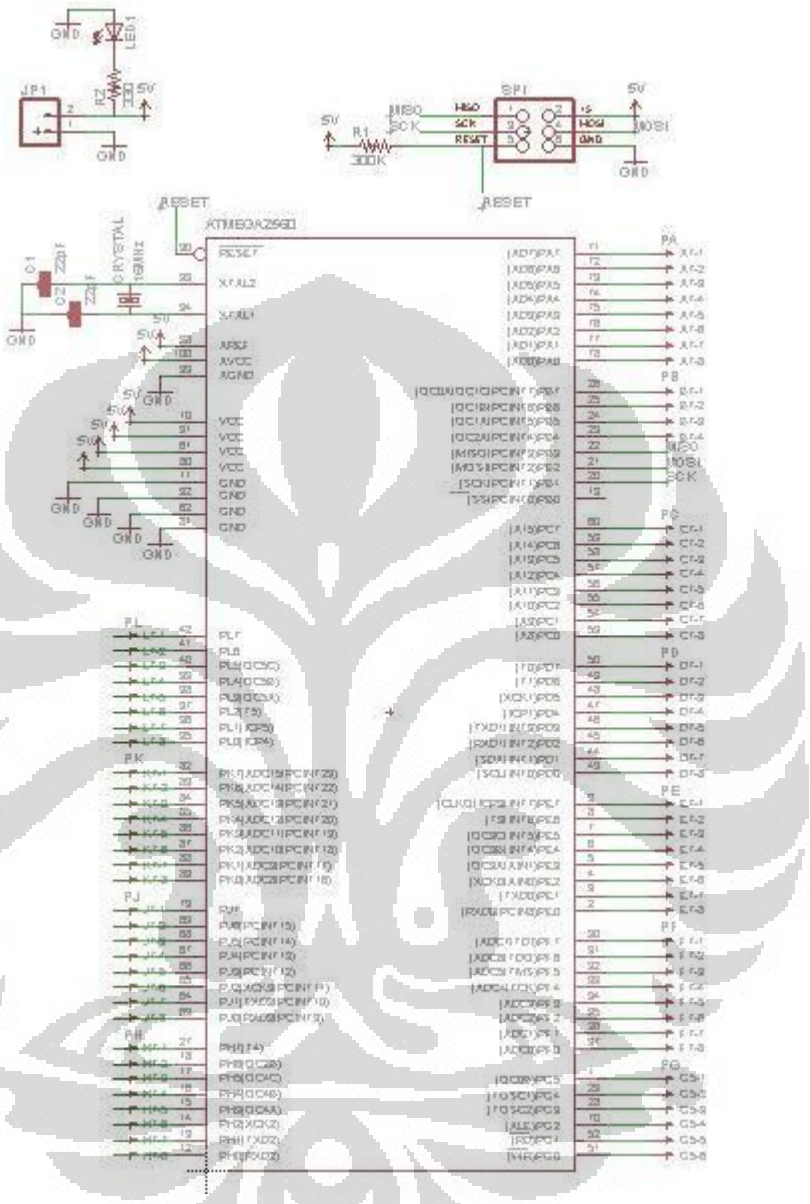
Berikut adalah *block diagram* dari *microcontroller* ATmega2560 :



Gambar 4.11 Block diagram ATmega2560 [8]

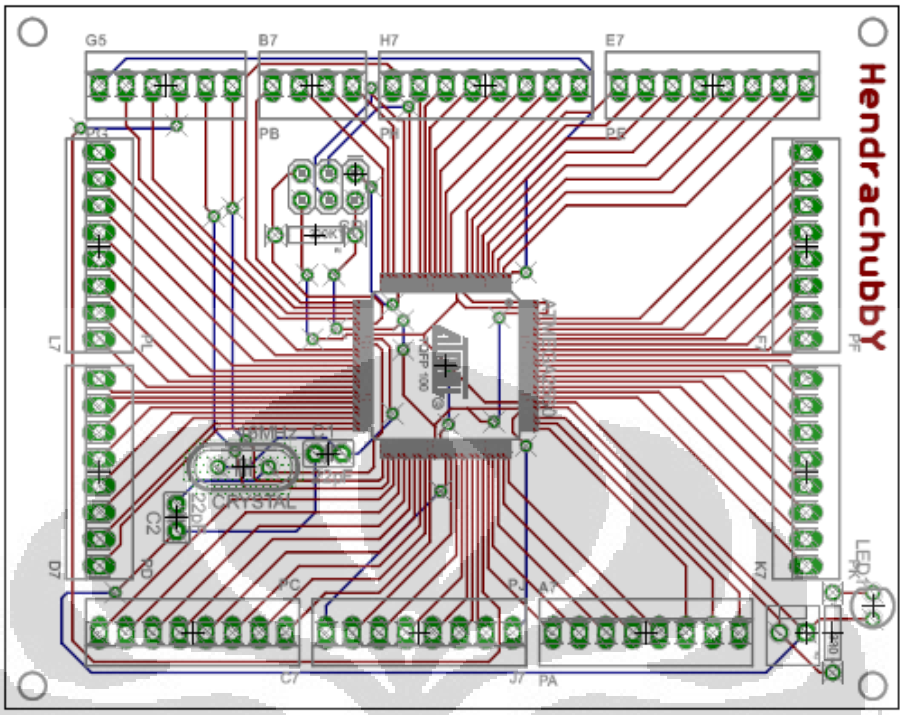
ATmega2560 diproduksi dalam bentuk IC, sehingga untuk *interfacing* dengan perangkat elektronika lain diperlukan *expansion board*. Sebenarnya *expansion board* untuk ATmega2560 sudah dirancang oleh ATMEL namun memiliki harga yang sangat mahal sehingga dibutuhkan perancangan *expansion board* sendiri dari ATmega2560. Untuk itu dirancang *expansion board* yang sederhana menggunakan *oscillator* 16Mhz. Perancangan *expansion board* ini harus berdasarkan pada *datasheet* ATmega2560 yang terlampir dan menggunakan *software* desain elektronika Eagle 5.6.0 dari CADsoft karena mudah digunakan dan *open source*.

Berikut adalah schematic dari *expansion board* ATmega2560 :

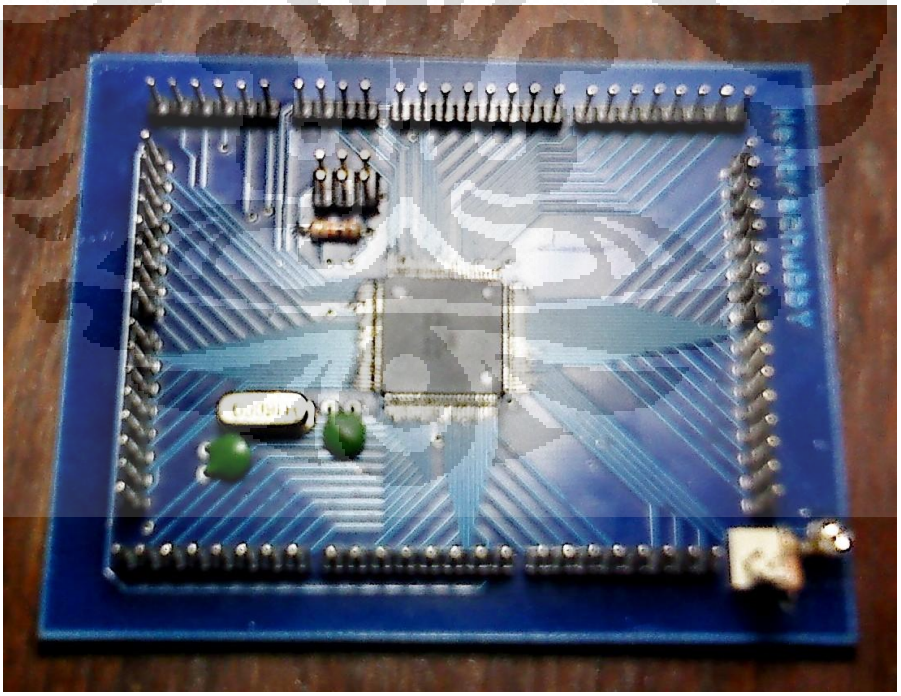


Gambar 4.12 Schematic Expansion Board ATmega2560

Seperti telihat pada gambar, setiap pin di expansi, hanya *pin – pin support* saja yang ditambahkan *hardware* seperti *oscillator* dan *port programming* 6 pin standar Atmel karena untuk memprogram ATmega2560 digunakan *programmer* AVR ISP MKII.



Gambar 4.13 PCB Layout Expansion Board ATmega2560



Gambar 4.14 Expansion Board ATmega2560

Sesuai dengan fitur yang terdapat pada ATmega2560 maka dibuat pin assignment yang sesuai dengan kebutuhan dan program yang terdapat didalamnya. Pin assignment ini dapat dilihat pada table berikut :

Tabel 4.2 Pin Assignment ATmega2560

Microcontroller I/O	Input atau Output	GPIO General Purpose I/O	Fungsi pada Axis	Cara Koneksi
PORTA.0	Output	Driver Input 1 Axis 1	Kontrol Arah Motor	Langsung
PORTA.1	Output	Driver Input 2 Axis 1	Kontrol Arah Motor	Langsung
PORTA.2	Output	Driver Input 1 Axis 2	Kontrol Arah Motor	Langsung
PORTA.3	Output	Driver Input 2 Axis 2	Kontrol Arah Motor	Langsung
PORTA.4	Output	Driver Input 1 Axis 3	Kontrol Arah Motor	Langsung
PORTA.5	Output	Driver Input 2 Axis 3	Kontrol Arah Motor	Langsung
PORTA.6	Output	Driver Input 1 Axis 4	Kontrol Arah Motor	Langsung
PORTA.7	Output	Driver Input 2 Axis 4	Kontrol Arah Motor	Langsung
PORTB.4	Input	Limit Switch Axis 5	Digital Sensing	10K Pullup Resistor
PORTB.5	Output	PWM Channel Axis 4	Kontrol Kecepatan	Langsung
PORTB.6	Output	PWM Channel Axis 5	Kontrol Kecepatan	Langsung
PORTB.7	Output	PWM Gripper	Kontrol Kecepatan	Langsung
PORTC.0	Output	LCD RS	LCD Kontrol	Langsung
PORTC.1	Output	LCD R/W	LCD Kontrol	Langsung
PORTC.2	Output	LCD E	LCD Kontrol	Langsung
PORTC.4	Output	LCD Data I/O	LCD Kontrol	Langsung
PORTC.5	Output	LCD Data I/O	LCD Kontrol	Langsung
PORTC.6	Output	LCD Data I/O	LCD Kontrol	Langsung
PORTC.7	Output	LCD Data I/O	LCD Kontrol	Langsung
PORTD.0	Output	Driver Input 1 Gripper	Kontrol Arah Motor	Langsung
PORTD.1	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTD.2	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTD.3	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTD.4	Output	Driver Input 2 Gripper	Kontrol Arah Motor	Langsung
PORTE.4	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTE.5	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTF.0	Output	Brake Signal	Kontrol Brake	Langsung
PORTF.1	Output	Brake Signal	Kontrol Brake	Langsung
PORTF.2	Output	Driver Input 1 Axis 5	Kontrol Arah Motor	Langsung
PORTF.3	Output	Driver Input 2 Axis 5	Kontrol Arah Motor	Langsung
PORTF.4	Input	Limit Switch Axis 1	Feedback	10K Pullup Resistor
PORTF.5	Input	Limit Switch Axis 2	Feedback	10K Pullup Resistor
PORTF.6	Input	Limit Switch Axis 3	Feedback	10K Pullup Resistor
PORTF.7	Input	Limit Switch Axis 4	Feedback	10K Pullup Resistor
PORTH.0	Input	Serial Communication	PC Interface	Langsung
PORTH.1	Input	Serial Communication	PC Interface	Langsung
PORTL.3	Output	PWM Channel Axis 1	Kontrol Kecepatan	Langsung
PORTL.4	Output	PWM Channel Axis 2	Kontrol Kecepatan	Langsung
PORTL.5	Output	PWM Channel Axis 3	Kontrol Kecepatan	Langsung

4.2.2 Motor Driver

Dalam pengendalian motor DC melalui *microcontroller* diperlukan perangkat elektronika yaitu *motor driver*. *Motor driver* diperlukan karena perbedaan suplai daya yang dikeluarkan *microcontroller* dengan yang dibutuhkan oleh motor DC. *Microcontroller* menggunakan *power* 0-5 V sedangkan motor DC menggunakan 24VDC. Selain itu motor yang dikontrol juga harus mempunyai kemampuan berbalik arah dan pengaturan kecepatan, algoritma kontrol arah dan kecepatan ini diberikan oleh *microcontroller* dan dijalankan oleh *motor driver*. *Motor driver* juga berguna sebagai pengaman dari arus balik akibat adanya medan elektromagnetis dari motor, arus balik ini sangat berbahaya bagi *microcontroller*.

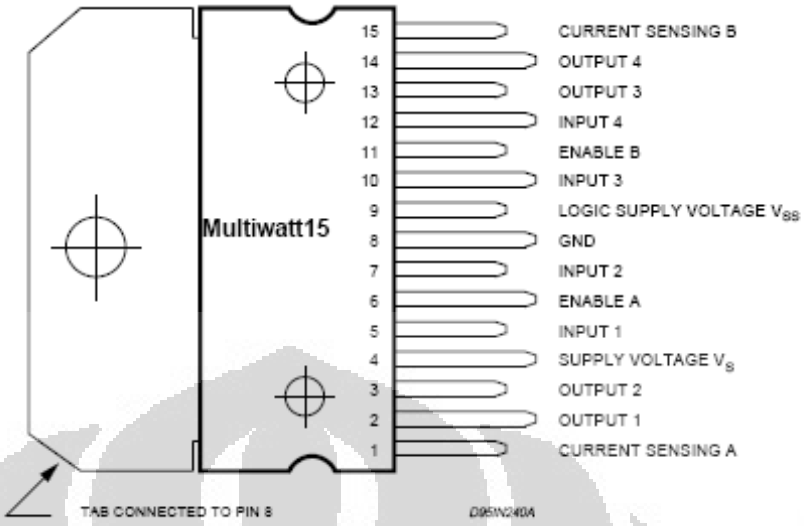
Motor driver untuk menggerakkan motor pada robot movemaster RV-M1 harus memiliki kemampuan sebagai berikut :

1. Power Supply 24 V.
2. PWM Support.
3. Arus > 3A.

Ini disebabkan karena spesifikasi motor DC pada robot movemaster RV-M1 memerlukan *motor driver* dengan kemampuan tersebut. Maka dirancang *motor driver* dengan basis IC L298D. Berikut adalah spesifikasi IC driver motor tersebut berdasarkan *datasheet*-nya :

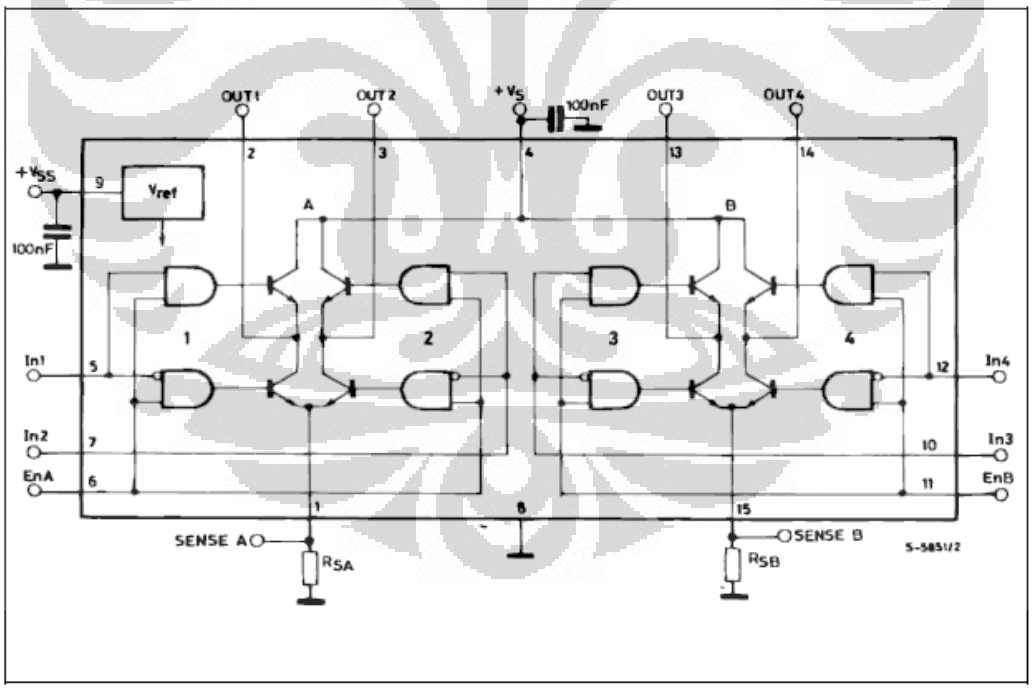
Tabel 4.3 Spesifikasi L298D [10]

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C



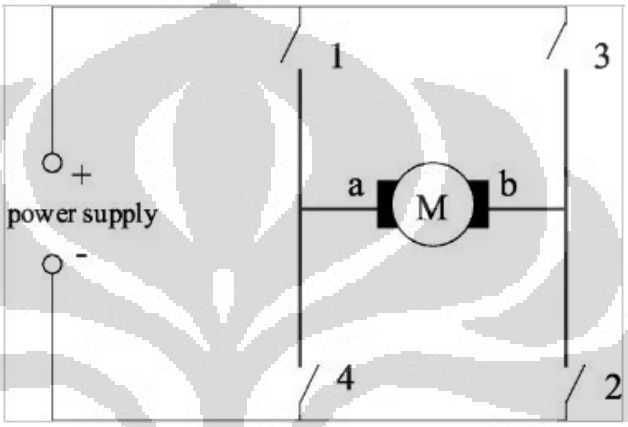
Gambar 4.15 IC L298D [10]

IC *motor driver* L298D mempunyai *block diagram* sebagai berikut :



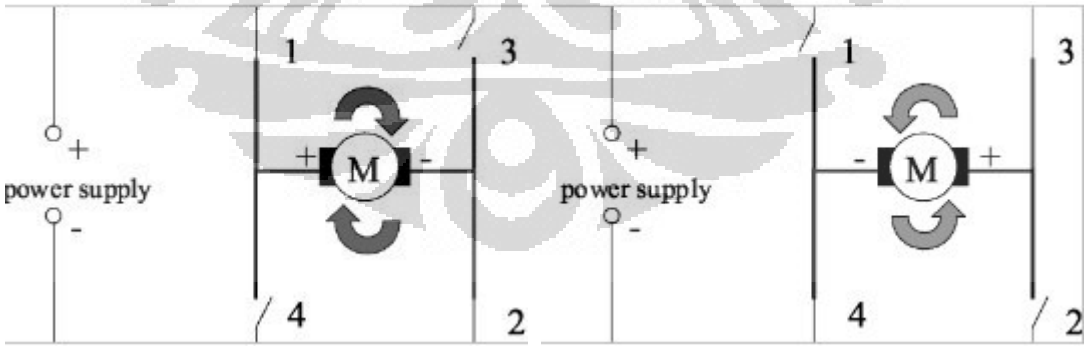
Gambar 4.16 Block Diagram L298D [10]

Dari spesifikasi diatas kemudian dirancang *board control* untuk *interfacing* ke *microcontroller*. IC L298 sebenarnya terdiri dari 2 *H-bridge* yang terintegrasi pada satu chip. *H-Bridge* atau yang diterjemahkan secara kasar sebagai "Jembatan H", adalah sebuah rangkaian dimana motor menjadi titik tengahnya dengan dua jalur yang bisa dibuka tutup untuk melewatkan arus pada motor tersebut, persis seperti huruf "H" (dengan motor berada pada garis horizontal).



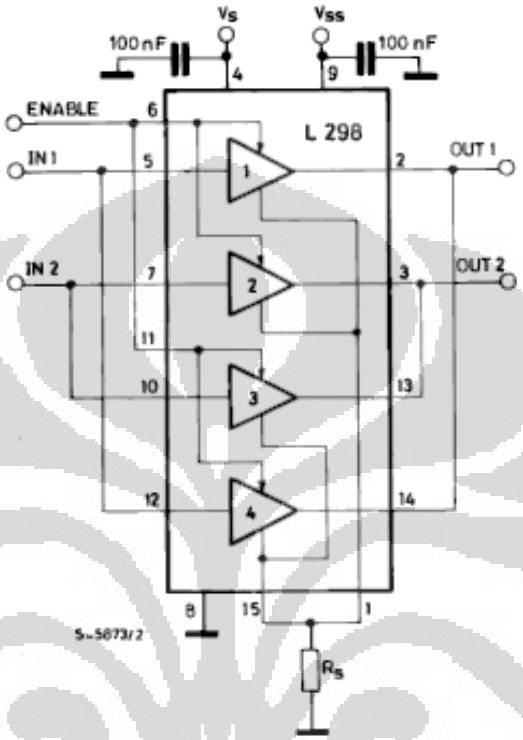
Gambar 4.17 Rangkaian H-Bridge [11]

Rangkaian ini memungkinkan motor dapat berputar secara CW (*Clock Wise*) dan CCW (*Counter Clock Wise*) sesuai dengan logika yang diberikan oleh *microcontroller*.



Gambar 4.18 Pengaturan Arah Putaran Motor pada *H-Bridge* [11]

Kedua H-Bridge yang terdapat pada IC L298 dapat dimanfaatkan dengan memparalel kedua H-Bridge tersebut sehingga dapat menahan arus sampai dengan 4A. Berikut adalah skematik dari rangkaian tersebut :



Gambar 4.19 Skematik L298D Paralel [10]

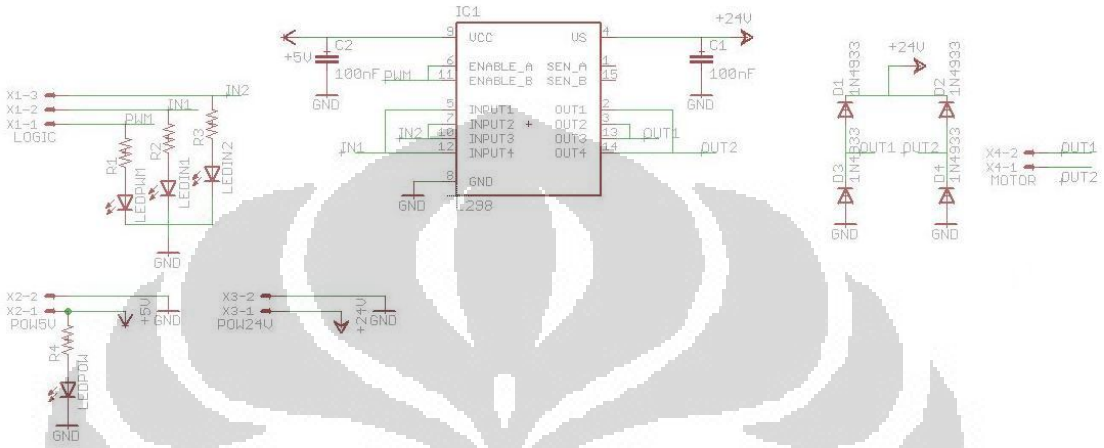
Untuk mengaktifasi pergerakan motor, IC L298D memerlukan *input* dengan pola yang dapat dilihat pada *truth table* berikut :

Tabel 4.4 *Truth Table* L298D [10]

Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

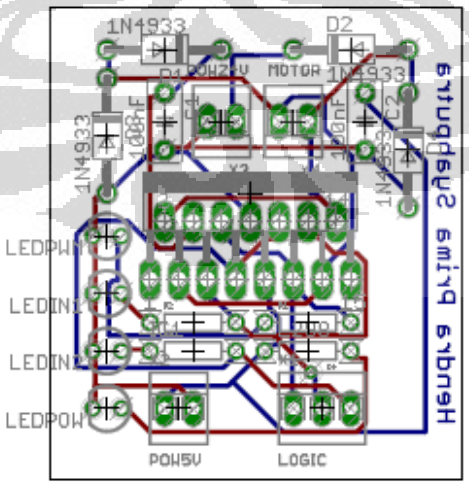
L = Low H = High X = Don't care

Untuk mempermudah *programmer* dalam mengakses *motor driver* ini ditambahkan berberapa LED dengan warna yang berbeda. Hal ini dapat membantu *programmer* memastikan dan memeriksa adanya *input* PWM maupun sinyal arah yang sesuai dengan *truth table* sehingga skematik dasarnya dikembangkan menjadi :



Gambar 4.20 Skematik L298D Paralel dengan LED

Berikut adalah PCB *layout double layer* dari *motor driver board* tersebut setelah dilengkapi dengan LED dan back EMF protection. *Back EMF Protection* berfungsi untuk menahan arus balik yang ditimbulkan oleh motor akibat adanya reaksi elektromagnetis pada motor DC :



Gambar 4.21 *PCB Layout Motor Driver*

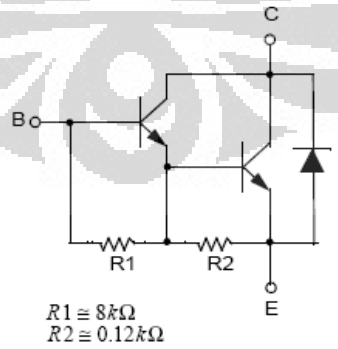
4.2.3 Driver Electric Brake

Pada robot movemaster RV-M1 terdapat 2 *electric brake* yaitu pada *axis* 2 dan 3. Electric brake ini bersifat NC (Normally Closed) sehingga perlu adanya driver untuk mengaktifkan brake tersebut sehingga menjadi *open state*. Berdasarkan spesifikasi, *electric brake* tersebut membutuhkan tegangan 12V sehingga dibuat perangkat *switching* dengan menggunakan IC TIP122. TIP122 merupakan transistor *darlington pair* yang dapat digunakan untuk aplikasi *switching* linear dengan spesifikasi sebagai berikut :

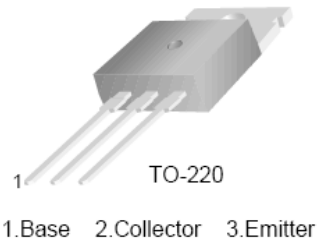
Tabel 4.5 Spesifikasi TIP122 [12]

Symbol	Parameter		Value	Units
V_{CBO}	Collector-Base Voltage	: TIP120	60	V
		: TIP121	80	V
		: TIP122	100	V
V_{CEO}	Collector-Emitter Voltage	: TIP120	60	V
		: TIP121	80	V
		: TIP122	100	V
V_{EBO}	Emitter-Base Voltage		5	V
I_C	Collector Current (DC)		5	A
I_{CP}	Collector Current (Pulse)		8	A
I_B	Base Current (DC)		120	mA
P_C	Collector Dissipation (T_a=25°C)		2	W
P_C	Collector Dissipation (T_C=25°C)		65	W
T_J	Junction Temperature		150	°C
T_{STG}	Storage Temperature		- 65 ~ 150	°C

Block diagram TIP 122 :

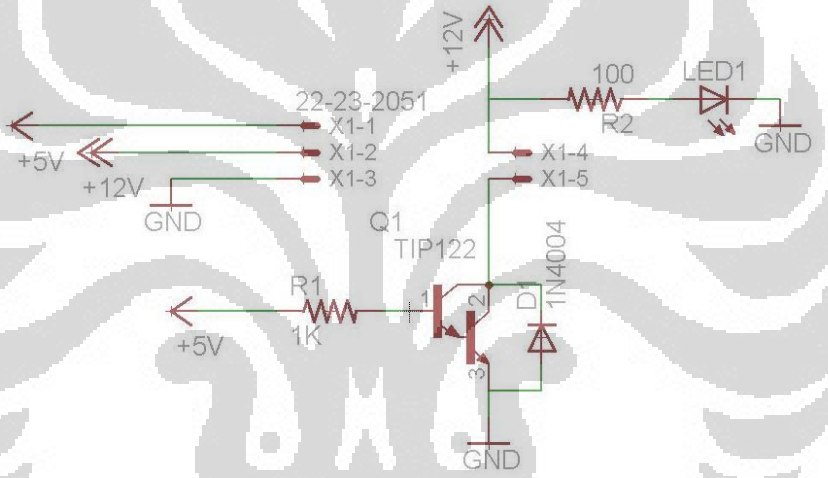


Gambar 4.22 *Block Diagram* TIP122 [12]



Gambar 4.23 TIP122 [12]

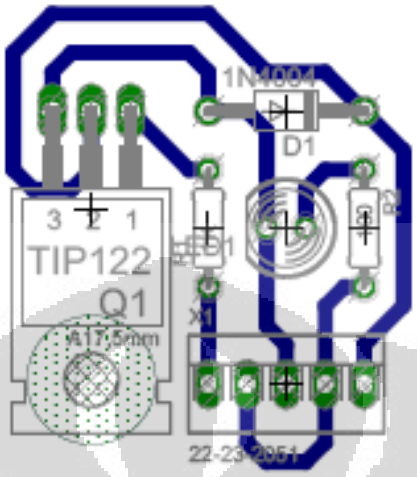
Berikut adalah skematik rangkaian *switching* menggunakan transistor TIP122 tersebut :



Gambar 4.24 Skematik Driver *Electric Brake*

Seperti terlihat pada gambar skematik, rangkaian *switching* ini diaktifasi oleh *microcontroller* yang memberikan tegangan 5V pada basis transistor. Apabila ada tegangan tersebut maka transistor akan teraktifasi sehingga tegangan sebesar 12V mengalir dari collector ke emittornya, tegangan inilah yang akan mengaktifkan *electric brake* pada robot movemaster RV-M1. *Electric brake* dapat menimbulkan tegangan balik karena *electric brake* bekerja seperti *solenoid* yang terdiri dari gulungan induksi elektromagnetis sehingga harus dilengkapi dengan dioda pengaman untuk mencegah arus balik merusak transistor TIP122.

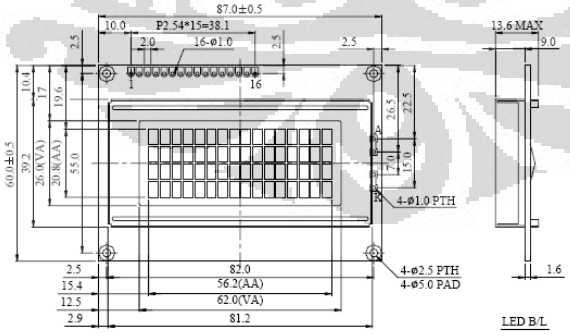
Berikut adalah layout PCB dari rangkaian *switching* ini :



Gambar 4.25 *PCB Layout Driver Electric Brake*

4.2.4 Modul LCD (Liquid Crystal Display)

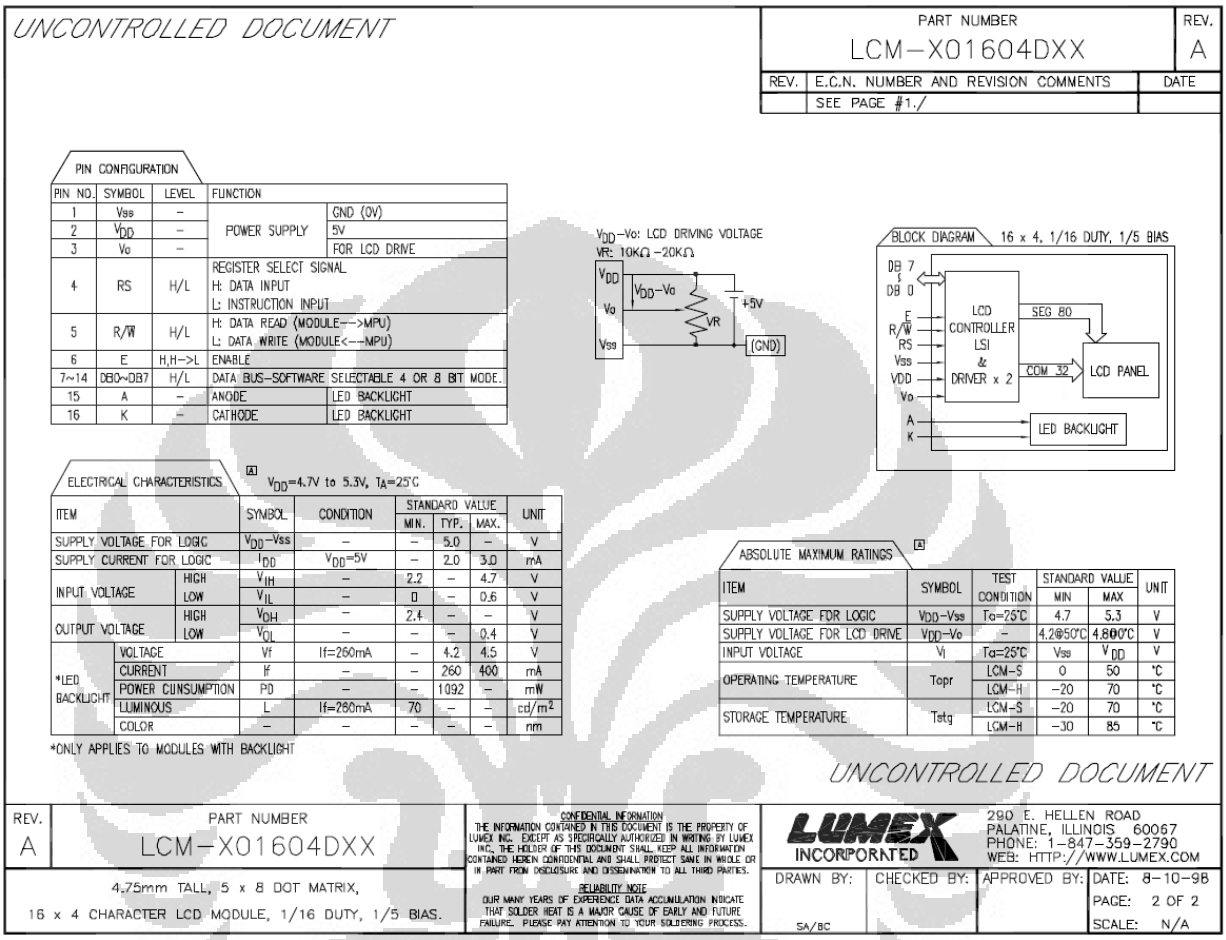
LCD digunakan dalam rangkaian kontrol sebagai perangkat display sekaligus memberikan informasi kepada user mengenai berbagai macam proses yang terjadi dalam microcontroller. Modul LCD ini sangat berguna untuk membantu programmer dalam debugging program pada microcontroller. Modul LCD yang digunakan adalah LCD karakter 16 x 4.



PIN NO.	SYMBOL
1	Vss
2	Vdd
3	Vo
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	A/Vee
16	K

Gambar 4.26 *Pinout* LCD 16 x 4 [13]

LCD yang dipakai adalah tipe LCM-S01604DSR dari Lumex Inc, berikut adalah spesifikasi dan gambar dari LCD tersebut :



UNCONTROLLED DOCUMENT

PART NUMBER	REV.
LCM−X01604DXX	A

REV.	E.C.N. NUMBER AND REVISION COMMENTS	DATE
	SEE PAGE #1./	

PIN CONFIGURATION

PIN NO.	SYMBOL	LEVEL	FUNCTION	
1	Vss	−	POWER SUPPLY	GND (0V)
2	V_{DD}	−		5V
3	Vo	−		FOR LCD DRIVE
4	RS	H/L	REGISTER SELECT SIGNAL H: DATA INPUT L: INSTRUCTION INPUT	
5	R/W	H/L	H: DATA READ (MODULE−−>MPU) L: DATA WRITE (MODULE<−−MPU)	
6	E	H,H−>L	ENABLE	
7~14	DB0~DB7	H/L	DATA BUS−SOFTWARE SELECTABLE 4 OR 8 BIT MODE.	
15	A	−	ANODE	LED BACKLIGHT
16	K	−	CATHODE	LED BACKLIGHT

V_{DD}−Vo: LCD DRIVING VOLTAGE
VR: 10KΩ −20KΩ

BLOCK DIAGRAM 16 x 4, 1/16 DUTY, 1/5 BIAS

ELECTRICAL CHARACTERISTICS [A] V_{DD}=4.7V to 5.3V, T_A=25°C

ITEM			SYMBOL	CONDITION	STANDARD VALUE MIN.	TYP.	MAX.	UNIT
SUPPLY VOLTAGE FOR LOGIC			V_{DD}−Vss	−	−	5.0	−	V
SUPPLY CURRENT FOR LOGIC			I_{DD}	V_{DD}=5V	−	2.0	3.0	mA
INPUT VOLTAGE		HIGH	V_{IH}	−	2.2	−	4.7	V
		LOW	V_{IL}	−	0	−	0.6	V
OUTPUT VOLTAGE		HIGH	V_{OH}	−	2.4	−	−	V
		LOW	V_{OL}	−	−	−	0.4	V
*LED BACKLIGHT	VOLTAGE		Vf	If=260mA	−	4.2	4.5	V
	CURRENT		If	−	−	260	400	mA
	POWER CONSUMPTION		PD	−	−	1092	−	mW
	LUMINOUS		L	If=260mA	70	−	−	cd/m²
	COLOR		−	−	−	−	−	nm

*ONLY APPLIES TO MODULES WITH BACKLIGHT

ABSOLUTE MAXIMUM RATINGS [A]

ITEM	SYMBOL	TEST CONDITION	STANDARD VALUE MIN	MAX	UNIT
SUPPLY VOLTAGE FOR LOGIC	V_{DD}−Vss	Ta=25°C	4.7	5.3	V
SUPPLY VOLTAGE FOR LCD DRIVE	V_{DD}−Vo	−	4.2@50°C	4.8@0°C	V
INPUT VOLTAGE	Vi	Ta=25°C	Vss	V_{DD}	V
OPERATING TEMPERATURE	Topr	LCM−S	0	50	°C
		LCM−H	−20	70	°C
STORAGE TEMPERATURE	Tstg	LCM−S	−20	70	°C
		LCM−H	−30	85	°C

UNCONTROLLED DOCUMENT

REV.	PART NUMBER
A	LCM−X01604DXX

4.75mm TALL, 5 x 8 DOT MATRIX,
16 x 4 CHARACTER LCD MODULE, 1/16 DUTY, 1/5 BIAS.

CONFIDENTIAL INFORMATION
THE INFORMATION CONTAINED IN THIS DOCUMENT IS THE PROPERTY OF LUMEX INC. EXCEPT AS SPECIFICALLY AUTHORIZED IN WRITING BY LUMEX INC., THE HOLDER OF THIS DOCUMENT SHALL KEEP ALL INFORMATION CONTAINED HEREIN CONFIDENTIAL AND SHALL PROTECT SAME IN WHOLE OR IN PART FROM DISCLOSURE AND DISSEMINATION TO ALL THIRD PARTIES.

RELIABILITY NOTE
OUR MANY YEARS OF EXPERIENCE DATA ACCUMULATION INDICATE THAT SOLDER HEAT IS A MAJOR CAUSE OF EARLY AND FUTURE FAILURE. PLEASE PAY ATTENTION TO YOUR SOLDERING PROCESS.

LUMEX INCORPORATED
2290 E. HELLEN ROAD
PALATINE, ILLINOIS 60067
PHONE: 1−847−359−2790
WEB: HTTP://WWW.LUMEX.COM

DRAWN BY:	CHECKED BY:	APPROVED BY:	DATE: 8−10−98
SA/BC			PAGE: 2 OF 2
			SCALE: N/A

Gambar 4.27 Spesifikasi LCD 16 x 4 [13]

Gambar 4.28 LCD 16 x 4 LCM-S01604DSR [13]

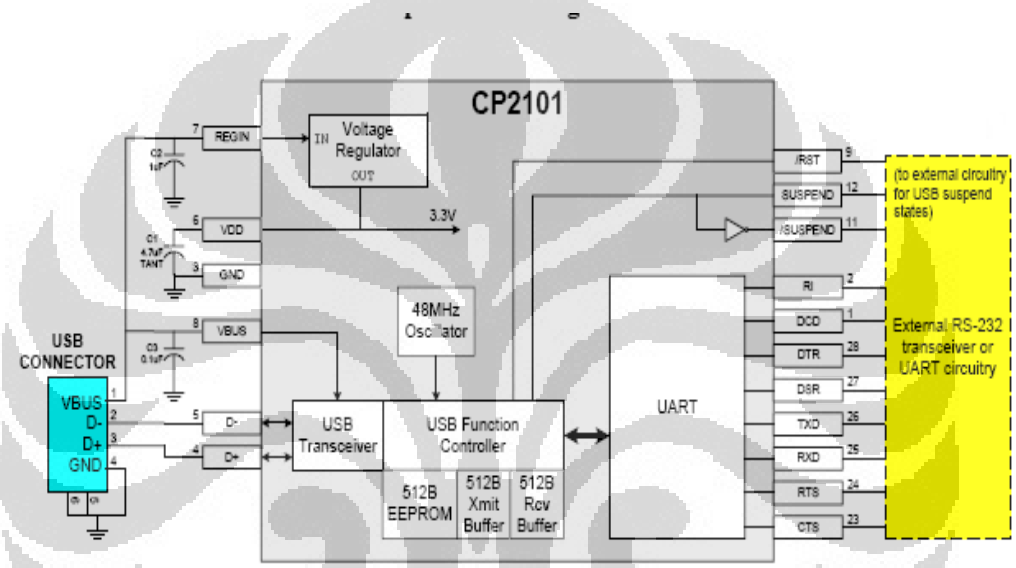
4.2.5 Adaptor USB – *Serial*

Untuk melakukan *transfer* data dari *microcontroller* ke komputer *server* diperlukan komunikasi antara kedua perangkat tersebut. Komunikasi *serial* merupakan komunikasi yang dimana pengiriman data dilakukan per bit. Data yang dikirimkan merupakan data yang berupa ASCII yang tercantum pada tabel berikut :

Tabel 4.6 Tabel ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

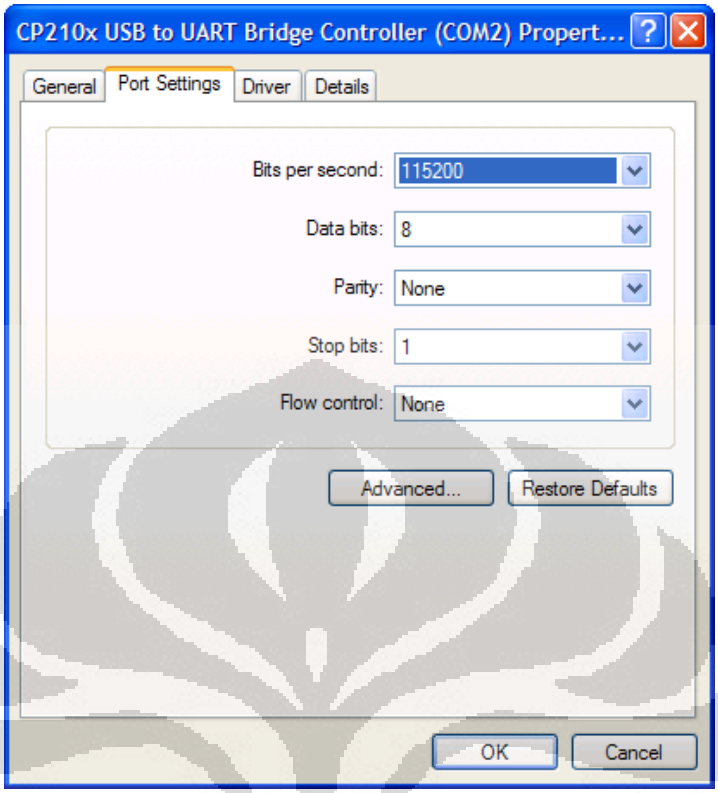
Adaptor serial ke USB merupakan perangkat yang menghubungkan *microcontroller* ke komputer *server* dari PORT RX dan TX pada *microcontroller* ke PORT USB pada komputer *server*. *Adaptor* ini melakukan konversi level tegangan dari komputer ke *microcontroller* dan sebaliknya. Hal ini dilakukan karena komputer menggunakan voltase logika 12V sedangkan *microcontroller* menggunakan voltase logika 5V. Adaptor USB – Serial ini menggunakan IC CP2101 *single chip USB to UART controller* yang mempunyai *block diagram* sebagai berikut :



Gambar 4.29 Block Diagram IC CP2101 [14]

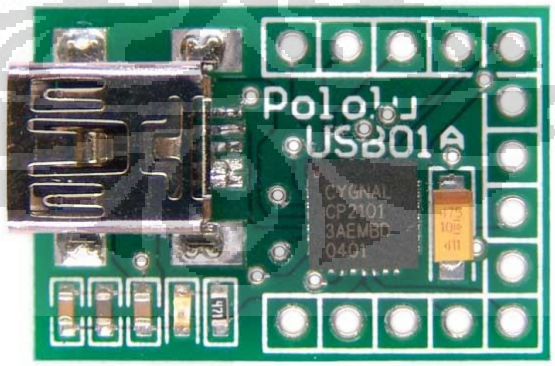
Seperti terlihat pada *block diagram*, IC tersebut melakukan konversi level tegangan dari PC ke microcontroller dan sebaliknya. Untuk konfigurasi komunikasi serial antara komputer server dengan microcontroller digunakan konfigurasi sebagai berikut :

1. Baud Rate : 115200 Bps.
2. Data Bits : 8 Bit.
3. Parity : None.
4. Stop Bits : 1.
5. Flow Control : None.



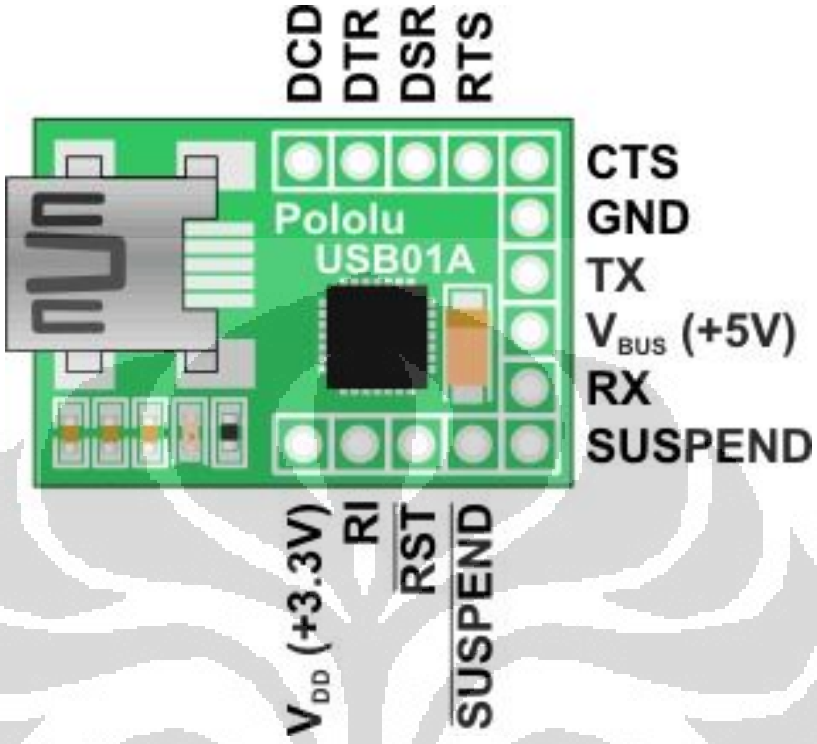
Gambar 4.30 *Setting* Komunikasi Serial Pada PC

Board dari IC CP2101 ini dibuat oleh *Pololu* (www.pololu.com), berikut adalah *board* dari IC CP2101 tersebut :



Gambar 4.31 Pololu USB to Serial Board [14]

Berikut adalah *Pinout* dari Pololu USB to Serial Board :



Gambar 4.32 Pololu USB to Serial Board Pinout [14]

Board ini mendapatkan daya dari port USB pada komputer yang memiliki tegangan 5V dan bisa langsung terkoneksi ke *microcontroller* dengan menggunakan PIN RX dan TX dari *board* ini.

4.2.6 Power Supply

Power supply untuk memberikan daya pada rangkaian elekronik adalah *switching power supply* dengan tegangan 5V, 12V dan 24V. *Power supply* tipe *switching* memungkinkan pemberian daya yang stabil pada range tegangan yang berbeda – beda. *Power supply* pada penelitian ini merupakan *power supply* produksi dari meanwell tipe S-100F untuk tegangan 12V dan 24V. *Power supply* ini memiliki proteksi *overload* yang sudah terdapat pada rangkaiannya. Sedangkan untuk tegangan 5V digunakan *power supply* bekas yang biasa dipakai pada alat *scanner*.

Berikut adalah spesifikasi power supply 24V dan 12V tersebut :

Tabel 4.7 Tabel Spesifikasi *Power supply* [15]

MODEL		S-100F-5	S-100F-7.5	S-100F-12	S-100F-15	S-100F-24	S-100F-48
OUTPUT	DC VOLTAGE	5V	7.5V	12V	15V	24V	48V
	RATED CURRENT	20A	13.5A	8.5A	6.7A	4.5A	2.2A
	CURRENT RANGE	0 ~ 20A	0 ~ 13.5A	0 ~ 8.5A	0 ~ 6.7A	0 ~ 4.5A	0 ~ 2.2A
	RATED POWER	100W	101W	102W	100.5W	108W	105.6W
	RIPPLE & NOISE (max.) Note.2	100mVp-p	125mVp-p	125mVp-p	125mVp-p	150mVp-p	150mVp-p
	VOLTAGE ADJ. RANGE	4.5 ~ 5.5V	6.75 ~ 8.25V	10.8 ~ 13.2V	13.5 ~ 16.5V	21.6 ~ 26.4V	43.2 ~ 52.8V
	VOLTAGE TOLERANCE Note.3	±2.0%	±1.0%	±1.0%	±1.0%	±1.0%	±1.0%
	LINE REGULATION	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
	LOAD REGULATION	±1.0%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
	SETUP, RISE TIME	1000ms, 30ms at full load					
	HOLD UP TIME (Typ.)	20ms at full load					
INPUT	VOLTAGE RANGE	88 ~ 132VAC/176 ~ 264VAC selected by jumper or switch 248 ~ 370VDC					
	FREQUENCY RANGE	47 ~ 63Hz					
	EFFICIENCY (Typ.)	76%	78%	80%	81%	83%	84%
	AC CURRENT (Typ.)	3.15A/115VAC 1.5A/230VAC					
	INRUSH CURRENT (Typ.)	COLD START 20A/115VAC 40A/230VAC					
	LEAKAGE CURRENT	<1mA / 240VAC					

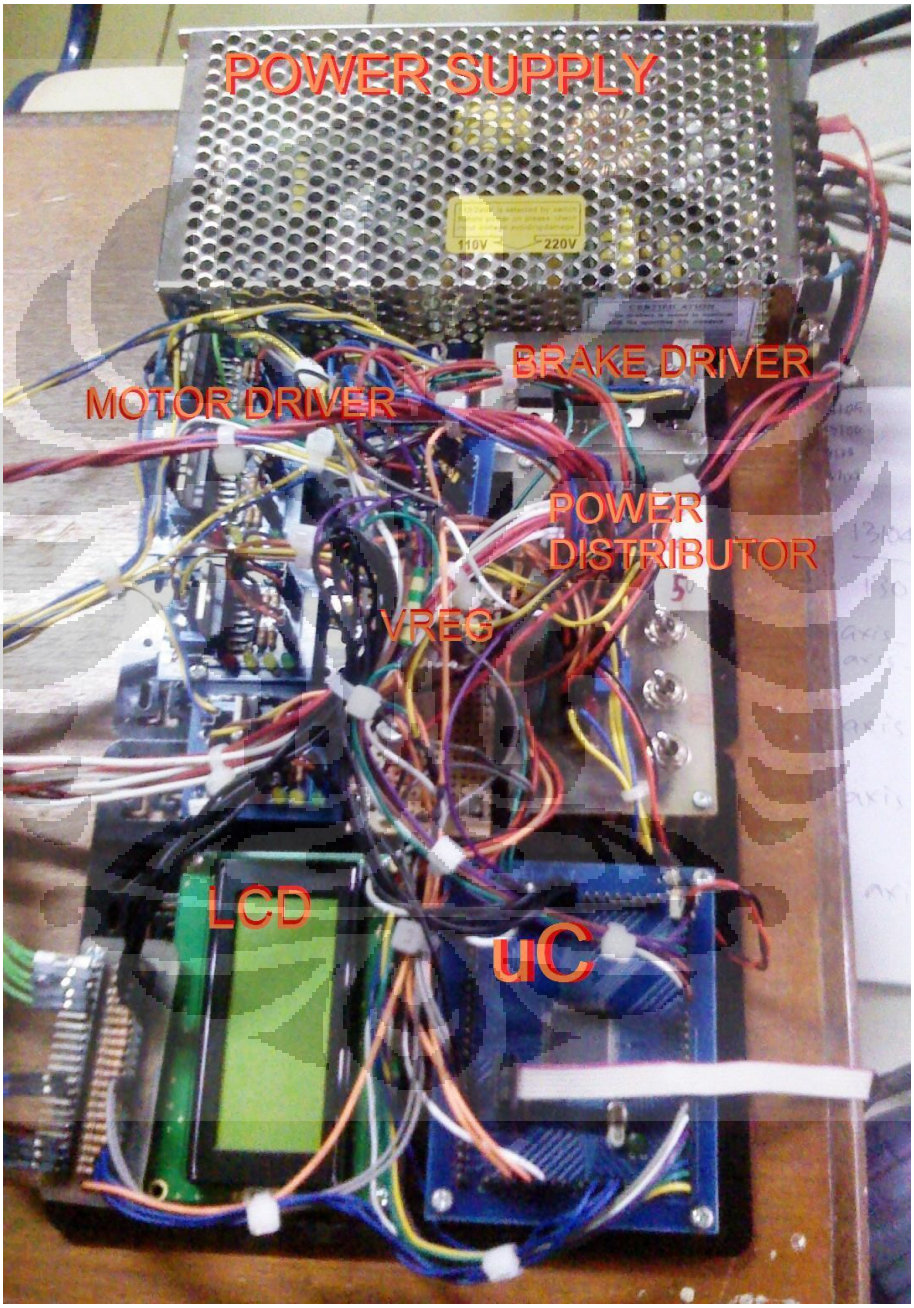
Gambar 4.33 Power Supply 12V dan 24V [15]

Untuk *power supply* digital dengan tegangan 5V digunakan *power supply* dengan arus 3A.

Gambar 4.34 Power Supply 5V

4.2.7 Control Unit

Seluruh perangkat elektronik diatas kemudian di rangkai sehingga menjadi sebuah perangkat keras *control unit* dari robot movemaster RV-M1. Berikut adalah gambar *control unit* tersebut :



Gambar 4.35 Perangkat Keras *Control Unit*

BAB 5
PENGEMBANGAN SISTEM PERANGKAT LUNAK

SISTEM PENGENDALI

Perangkat lunak pada sebuah sistem pengendali berfungsi untuk mengatur informasi antara *input*, proses dan *output* dari sistem yang dikendalikan. Microcontroller diprogram dengan algoritma yang disesuaikan dengan *software interface* ke PC. Program pada microcontroller harus mampung menerima dan mengirim data ke komputer dan harus mampu mengontrol perangkat keras lainnya yang terhubung ke robot. Program yang dibuat menggunakan *compiler* Codevision AVR ini mempunyai beberapa fungsi yang dibuat untuk mengatur microcontroller agar dapat menjalankan fungsi yang dibutuhkan. Berikut adalah algoritma yang dirancang untuk perangkat lunak sistem pengendali :

```
Algorithm :

    START
        1) Setting microcontroller.
        2) Menyalakan lampu indikator (robot siap menerima perintah).
        3) Menunggu perintah dari komputer server
        4) IF perintah dari server = nilai perintah default, lakukan
           pergerakan posisi default.
        5) IF perintah dari server = nilai perintah manual, lakukan
           pergerakan manual.
        6) IF perintah dari server = nilai perintah web, lakukan pergerakan
           berbasis web.
        7) IF perintah dari server = nilai berhenti, berhenti menerima
           perintah
    END
```

Untuk menjalankan sebuah perintah yang diberikan, *microcontroller* harus mengenali nilai perintah yang dikirim oleh komputer *server*. Nilai perintah ini merupakan sebuah nilai yang ditentukan untuk menjadi protokol komunikasi antara *microcontroller* dengan komputer *server*. Nilai perintah tersebut berupa bit yang dikirimkan secara serial oleh komputer *server*.

5.1 Fungsi Utama

Fungsi utama berfungsi untuk menerima perintah dari PC dan pengaturan awal microcontroller.

Terdapat 4 pilihan perintah yaitu :

1. Default position : value yang diterima dari PC = 255
2. Manual mode : value yang diterima dari PC = 254
3. *Web* mode : value yang diterima dari PC = 253
4. Reset mode : value yang diterima dari PC = 252

LED indikator akan menyala apabila microcontroller siap menerima perintah.

```
Algorithm :

    START
        8)  Setting microcontroller.
        9)  Setting nilai - nilai perintah.
        10) Menampilkan tulisan pada LCD Robot Initialize
        11) Tunggu 3 detik.
        12) Menampilkan tulisan pada LCD Ready to Execute
        13) Tunggu 3 detik.
        14) Mengosongkan LCD.
        15) Menyalakan lampu indikator (robot siap menerima perintah).
        16) Mulai loop menunggu perintah.
        17) IF nilai perintah = 255.
            a. menuju fungsi default_position.
            b. kirim verifikasi posisi default.
            c. matikan lampu indicator.
        18) IF nilai perintah = 254.
            a. masuk mode manual.
            b. matikan lampu indicator.
        19) IF nilai perintah = 253
            a. menuju mode web.
            b. matikan lampu indicator.
        20) IF nilai perintah = 252.
            a. keluar dari loop
            b. matikan lampu indicator.
    END
```

5.2 Pengaturan I/O PORT Microcontroller

Fungsi untuk mengatur fungsi setiap PORT, baik sebagai *input* atau *output*.

```
Algorithm :
    START
       1)  Setting PORT F
       2)  Setting PORT A
       3)  Setting PORT B
       4)  Setting PORT D
       5)  Setting PORT E
       6)  Setting PORT H
       7)  Setting PORT L
       8)
    END
```

5.3 Pengaturan LCD

LCD diatur untuk bekerja pada PORT C.

```
Algorithm :
    START
       1) Mulai penulisan in-line assembly language.
       2) Setting LCD pada PORT C.
       3) Tutup penulisan in-line assembly languange.
    END
```

5.4 Pengaturan Clock Speed

Oscillator yang dipakai adalah 16Mhz dengan faktor pembagi 1. Kecepatan microcontroller dapat diatur dengan menaikkan atau menurunkan faktor pembagi.

```
Algorithm :
    START
       1) Setting oscillator.
       2) Setting factor pembagi.
    END
```

5.5 Pengaturan Komunikasi Serial

Fungsi pengaturan parameter komunikasi serial, PORT komunikasi serial yang dipakai adalah serial 2.

```
Algorithm :
    START
       1) Setting tipe data serial communication, 8 Data, 1 Stop, No Parity
       2) Aktifasi transmitter receiver
       3) Setting mode serial asynchronous
       4) Setting baud rate 115200 BPS.
    END
```

5.6 Pengaturan Timer

Timer yang digunakan adalah *timer* 1 dan 5 dengan perhitungan frekuensi PWM :

fPWM = fOSC / (N*(1+1023))

fPWM = 16Mhz / (64*1024)

fPWM = 244.14 Hz

tPWM = 1 / 244.14

tPWM = 0.0041 s

tPWM = 4 ms

```
Algorithm :
    START
       1) Pengaturan sumber clock, sumber clock dari system clock.
       2) Pengaturan Frekuensi Timer menjadi 2000.000 kHz.
       3) Pengaturan mode PWM menjadi fast PWM.
       4) Menyalakan semua timer
       5) Menyalakan noise canceler.
    END
```

5.7 Pengaturan Interrupt

Semua *interrupt* pada microcontroller di-*set* dengan mode *raising edge*.

```
Algorithm:
    START
       1) Menyalakan INT0 dengan mode rising edge.
       2) Menyalakan INT1 dengan mode rising edge.
       3) Menyalakan INT2 dengan mode rising edge.
       4) Menyalakan INT3 dengan mode rising edge.
       5) Menyalakan INT4 dengan mode rising edge.
       6) Menyalakan INT5 dengan mode rising edge.
       7) Menyalakan INT6 dengan mode rising edge.
       8) Menyalakan INT7 dengan mode rising edge.
       9)
    END
```

5.8 Fungsi Pengontrolan Motor

Fungsi ini untuk mengatur frekuensi PWM dan memberikan sinyal *pada motor driver* sesuai dengan *truth table motor driver*.

```
Input : Parameter pergerakan axis.
Output: Sinyal truth table dan PWM

Algorithm :
    START
       1) Menunggu data yang masuk.
       2) Menentukan pemilihan truth table motor driver masing-masing axis
          berdasarkan arah_axis yang diperintahkan.
       3) Mengirim sinyal ke motor driver.
       4) Mengaktifkan PWM sesuai dengan perintah.
    END
```

5.9 Fungsi Pengiriman Text Secara Serial

Fungsi mengirim data *text* melalui serial port 2.

```
Input : Text dari user
Output: Data bit menuju PC
Algorithm :
    START
       1) Periksa registry serial apakah kosong.
       2) Apabila kosong, data = registry.
       3) Kirim semua text.
    END
```

5.10 Fungsi Pengiriman Nilai Secara Serial

Fungsi mengirim nilai melalui serial port 2.

```
Input : Nilai dari user
Output: Data bit menuju PC

Algorithm :
    START
       1) Periksa registry serial apakah kosong.
       2) Apabila kosong, data = registry.
       3) Kirim semua data.
    END
```

5.11 Fungsi Menampilkan Data Pada LCD

Fungsi-fungsi untuk menampilkan pembacaan pulsa *encoder* pada LCD.

```
Input : Data perhitungan encoder.
Output: Data ASCII pada LCD.

Algorithm :
START
   1) Menuju posisi LCD 0,0
   2) Konversi data int menjadi ASCII
   3) Tampilkan data di LCD
END
```

5.12 Menghentikan Pergerakan Robot

Fungsi-fungsi untuk memberhentikan masing – masing *axis*.

```
Algorithm :
START
   1) Mengatur signal ke motor driver menjadi stop sesuai truth table.
   2) Mematikan timer.
   3) Reset perhitungan encoder.
END
```

5.13 Menggerakkan Semua Axis Secara Bersamaan

Fungsi menggerakkan robot, berlaku untuk setiap *axis*. Fungsi ini akan menggerakkan setiap motor sampai pulsa yang dihitung *encoder* sesuai dengan target.

```
Input : Data pergerakan robot setiap axis.
Output: Sinyal truth table dan PWM setiap axis.

Algorithm :
START
   1) Menerima data pergerakan setiap axis.
   2) Melakukan konversi perhitungan encoder menjadi sudut real.

   Perhitungan pulsa encoder Pengukuran dilakukan dengan menampilkan pulsa
   encoder pada LCD, kemudian mengambil rata-rata yang kemudian dibagi
```

```
   dengan datapada datasheet Movemaster RV-M1 untuk mendapatkan nilai pulsa
   per 1 derajat pergerakan encoder pulses.

      axis_1      : 16932
                  : 16939
                  : 16940
                  : 16934
      avg         : 16936
      robot spec  : 300'
      ~ 57 = 1' actual

      axis_2      : 13105
                  : 13100
                  : 13103
                  : 13104
      avg         : 13103
      robot spec  : 130'
      ~ 101 = 1' actual

      axis_3      : 7368
                  : 7396
                  : 7351
                  : 7362
      avg         : 7370
      robot spec  : 110'
      ~ 67 = 1' actual

      axis_4      : 8862
                  : 8861
                  : 8875
                  : 8874
      avg         : 8868
      robot spec  : 180'
      ~ 50 = 1' actual

      axis_5      : 5228
                  : 5150
                  : 5330
                  : 5301
      avg         : 5252
      robot spec  : 180
      ~ 30 = 1' actual

   3) Menggerakkan semua axis secara bersamaan.
   4) Mulai loop pengecekan encoder.
   5) IF perhitungan encoder tercapai > motor berhenti.
   6) IF posisi semua axis tercapai > keluar dari loop.
END
```

5.14 Mode Posisi Default

Fungsi untuk memerintahkan robot untuk posisi *default*-nya.

```
Algorithm :
    START
       1) Buka brake axis 2.
       2) Gerakkan axis 2 menuju posisi default.
       3) Stop axis 2 apabila switch axis 2 menyala.
       4) Buka brake axis 3.
       5) Gerakkan axis 3 menuju posisi default.
```

```
        6)  Stop axis 3 apabila switch axis 3 menyala.
        7)  Gerakkan axis 1 menuju posisi default.
        8)  Stop axis 1 apabila switch axis 1 menyala.
        9)  Gerakkan axis 4 menuju posisi default.
        10) Stop axis 4 apabila switch axis 4 menyala.
        11) Gerakkan axis 5 menuju posisi default.
        12) Stop axis 5 apabila switch axis 5 menyala.
    END
```

5.15 Mode Manual

Fungsi ini merupakan fungsi untuk menampung data yang dikirim apabila robot ingin digerakkan secara manual. Setelah data pergerakan sudah lengkap robot langsung mengeksekusi pergerakan sesuai dengan data yang dikirim dari *server*.

```
Algorithm :
    START
        1)  Kirim text untuk meminta user memasukkan data.
        2)  Tunggu data dikirim.
        3)  Set data yang dikirim sebagai parameter pergerakan robot.
        4)  Point 2) dan 3) dilakukan terus sampai data lengkap
        5)  Eksekusi pergerakan robot.
    END
```

5.16 Mode Web

Fungsi ini merupakan fungsi untuk menampung data yang dikirim dari komputer *server* apabila robot ingin digerakkan secara *web based*. Setelah data pergerakan sudah lengkap robot langsung mengeksekusi pergerakan sesuai dengan data yang dikirim dari komputer *server*. Fungsi ini berbeda dengan fungsi manual karena tidak mengirimkan informasi kepada client dan bersifat otomatis.

```
Algorithm :
STATUS
    1)  Tunggu data dikirim.
    2)  Set data yang dikirim sebagai parameter pergerakan robot.
    3)  Point 1) dan 2) dilakukan terus sampai data lengkap.
    4)  IF semua nilai pergerakan adalah 1 maka robot melakukan perintah
        posisi default.
    5)  Eksekusi pergerakan robot.
END
```

BAB 6
PENGEMBANGAN SISTEM KOMUNIKASI DENGAN *WEB* SERVER

Sistem komunikasi dengan *web server* dibuat untuk mengambil data dari dihasilkan *web server* yang merupakan data perintah dari *client.* Sistem komunikasi dibuat dengan menggunakan *interface software* yang di-*install* pada komputer *server.* Program ini harus mampu mengambil data yang dihasilkan oleh *web server* dan mengirim data tersebut ke *microcontroller* untuk menggerakkan robot.

6.1 Program Interface pada Komputer Server

Program ini dibuat menggunakan aplikasi WIN16 dengan *compiler* Turbo C. Berikut adalah algoritma *interface software* yang dirancang untuk komunikasi komputer *server* dengan *microcontroller*.

```
Algorithm :
      START
          1) Menunggu perintah dari user
          2) IF Perintah dari user = perintah manual, lakukan aktifkan mode
             manual
          3) IF Perintah dari user = perintah web, lakukan aktifkan mode
             web.
          4) Membuka PORT komunikasi komputer server.
          5) Mengirim data ke microcontroller secara serial.
          6) Menutup PORT komunikasi.
      END
```

6.1.1 Fungsi Utama

Fungsi utama pada *interface software* ini ini berfungsi untuk memberitahukan *user* perintah yang ingin dilakukannya.

```
Algorithm :
      START
          7) Melakukan looping seluruh fungsi.
          8) Mengosongkan layar tampilan.
          9) Menampilkan text pilihan perintah.
          10) Switch, menunggu perintah oleh user
               i.   case 1 :
                 a. Membuka PORT komunikasi.
```

```
            b. Mengaktifasi mode manual
         ii.   case 2 :
            a. Membuka PORT komunikasi.
            b. Mengaktifasi mode web.
       iii.   case 3 : Keluar program.
    11) IF perintah salah program akan memberitahukan user kalau input
        salah.
END
```

6.1.2 Fungsi Membuka PORT Komunikasi

Fungsi membuka PORT komunikasi pada komputer . PORT komunikasi yang dipakai adalah COM1.

```
Algorithm :
    START
        1) Matikan interrupt COM1.
        2) Pengaturan baud rate, 115200BPS.
        3) Pengaturan tipe serial, 8 Bits, No Parity, 1 Stop Bit.
        4) Menyalakan DTR, RTS, and OUT2.
    END
```

6.1.3 Fungsi Mengambil Data Komunikasi Serial

Fungsi ini berfungsi untuk menerima data dari microcontroller dan menampilkan data tersebut pada layar monitor komputer *server*. Fungsi ini digunakan untuk konfirmasi microcontroller yang telah atau sedang menjalankan perintah.

```
Algorithm :
    START
        1) Reset counter dan mendefinikan array buffer.
        2) Melakukan loop pengecekan input.
        3) IF ada input tampilkan data.
        4) If keyboard ditekan keluar loop.
    END
```

6.1.4 Fungsi Pemilihan Pergerakan *Manual*

Robot movemaster RV-M1 juga bisa digerakkan secara *manual* melalui fungsi ini. Misalkan untuk melakukan gerakan *default* robot melalui komputer *server*.

```
Algorithm :
    START
        1) Mengosongkan tampilan layar.
        2) Menampilkan tulisan informasi pilihan.
        3) Menunggu input dari user untuk pilihan perintah.
        4) Switch (perintah user)
            i.   case 1 : Mengirim perintah posisi default.
           ii.   case 2 : Menuju perintah mode manual.
        5) IF perintah salah program akan memberitahu user kalau input
           salah.
    END
```

6.1.5 Fungsi Pemilihan Pergerakan *Web*

Untuk mengaktifkan pengendalian berbasis *web* diperlukan adanya perintah dari interface software ini karena pengecekan data akan berlangsung terus menerus setelah mode *web* diaktifkan. Fungsi dapat menghentikan dan memulai pengencekan data tersebut.

```
Algorithm :
       START
          1) Mengosongkan tampilan layar.
          2) Menampilkan tulisan pemilihan mode web
          3) Menunggu perintah user.
          4) Switch (perintah)
              i.    case 1 : Mengaktifkan mode pengecekan file
             ii.    case 2 : Mengirim data reset ke microcontroller
          5) IF perintah salah program akan memberitahu user kalau input
             salah
       END
```

6.1.6 Perintah Posisi Default

Fungsi mengirim data untuk perintah posisi default. *Microcontroller* akan melakukan perintah posisi default apabila mendapat bit 255. Setelah posisi *default* dicapai *microcontroller* akan mengirim informasi konfirmasi.

```
Algorithm :
       START
          1) Mengosongkan tampilan layar.
          2) Mengirim bit 255 ke microcontroller.
          3) Menunggu data verifikasi dari microcontroller.
       END
```

6.1.7 Perintah Pergerakan *Manual*

Fungsi mengirim data pergerakan robot secara *manual*. Setiap *axis* memerlukan 3 data yaitu : kecepatan, arah, dan sudut.

```
Algorithm
       START
          1) Mengosongkan tampilan layar.
          2) Menampilkan tulisan untuk meminta data dari user
          3) Menerima data dari user.
          4) Mengirim data ke microcontroller.
       END
```

6.1.8 Fungsi Pengecekan Data Pergerakan *Web*

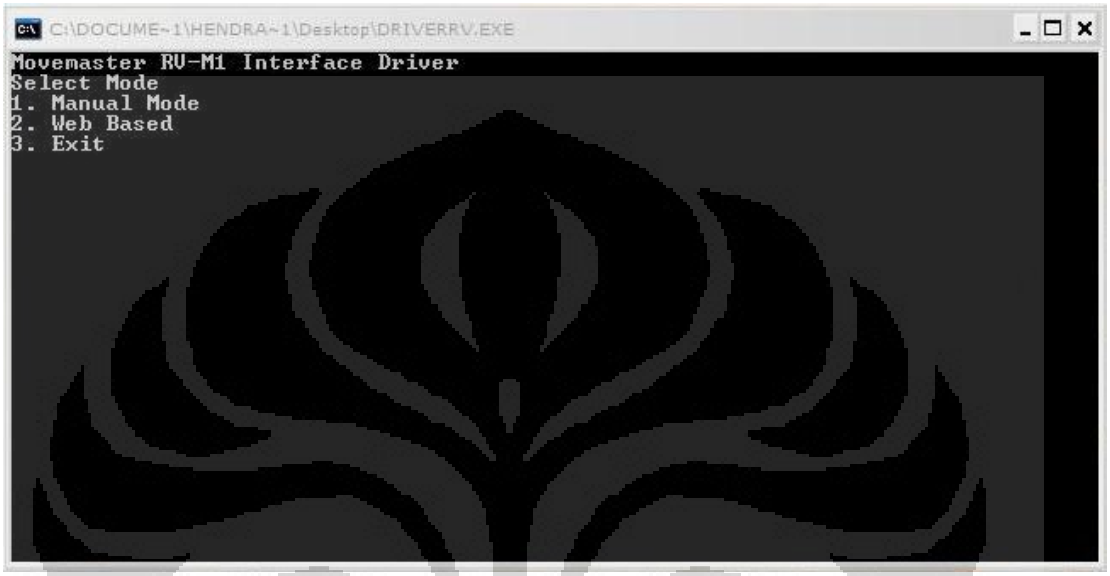
Fungsi ini akan selalu mengecek keberadaan file kdt.txt yang dihasilkan oleh *web server*.

```
Algorithm :
       START
          1) Memulai loop pengecekan file.
          2) Mengosongkan tampilan layar.
          3) Membuka text file pada path D:/adtj/www/htdocs/i-RoMan/kdt.txt
          4) IF keyboard ditekan
             a. Data yang akan dikirim bernilai 0
             b. Mengirim data ke microcontroller
             c. Tutup file text
             d. Hapus file text.
             e. Keluar fungsi
          5) IF file ditemukan
             a. Tampilkan informasi file ditemukan.
             b. Membaca isi file.
             c. Menyimpan isi file dalam array pergerakan, mengirim data
                pergerakan.
             d. Mengirim data aktifasi mode web = 255.
             e. Menampilkan data yang dikirim
             f. Menutup file text
             g. Menghapus file text.
          6) IF file tidak ditemukan tampilkan informasi file tidak
             ditemukan.
          7) Tunggu 1 Detik
          8) Kembali ke awal fungsi
       END
```

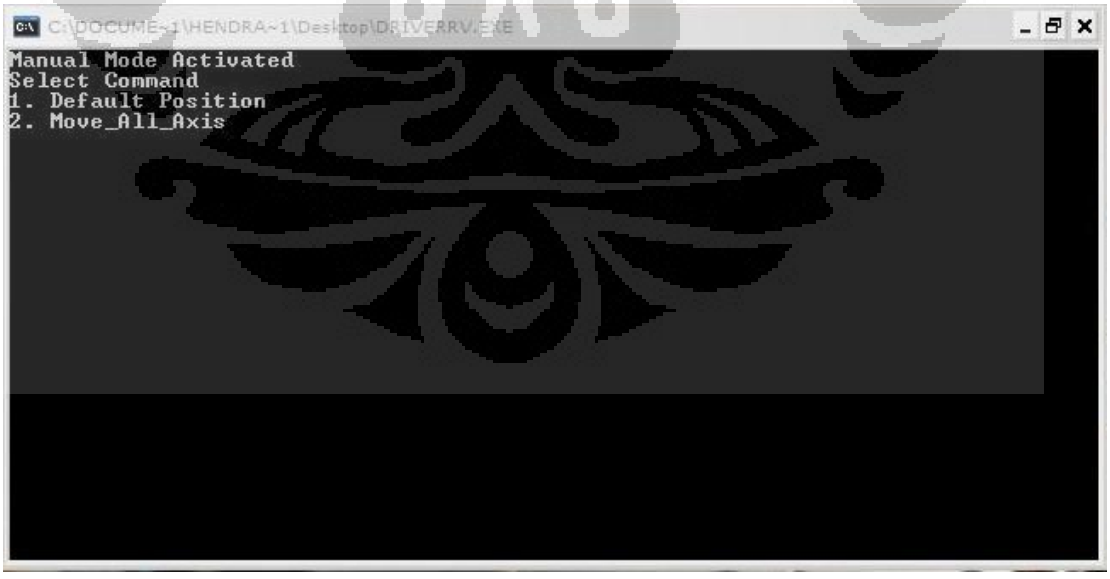
Software ini harus dijalankan sebelum mengirim perintah dari *web*, software ini yang akan mengolah data yang disimpan *web* dalam text file. Program dapat dijalankan dengan mengeksekusi file bernama DRIVERRV.exe yang merupakan *executable file* program ini. Begitu dijalankan maka program akan meminta beberapa *input* dari user untuk memilih satu dari beberapa metode pergerakan yang ada. Untuk metode *emergency* tidak diaktifasi melalui software ini, namun software ini bersifat sebagai penerima data emergency yang dihasilkan oleh *web* server yang kemudian disimpan dalam file text yang sama. Isi data *emergency* sebenarnya adalah data pergerakan namun setiap parameter bernilai 0 sehingga robot akan berhenti begitu mendapat data seperti itu. Software ini juga dilengkapi *user interface* yang sederhana yang berfungsi untuk membantu user mengetahui kondisi program. Selain itu user

juga dapat menghentikan program dengan memasukkan *input* melalui *keyboard*. Program ini akan menutup secara otomatis disaat mendapat perintah *exit*.

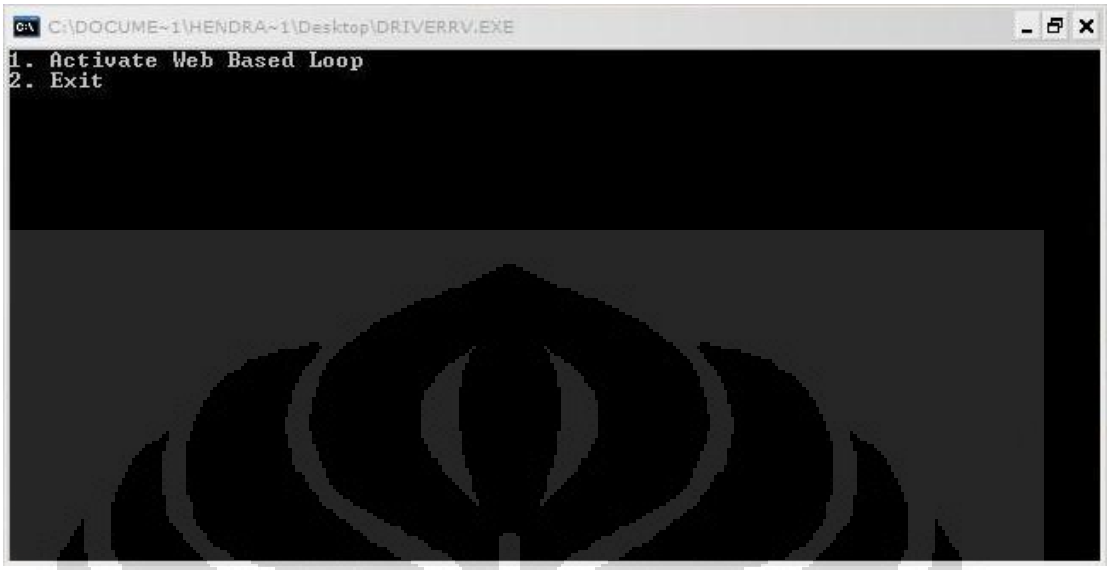
Berikut adalah tampilan *software* pada saat dieksekusi :



Gambar 6.1 Tampilan Awal



Gambar 6.2 Tampilan Mode Manual

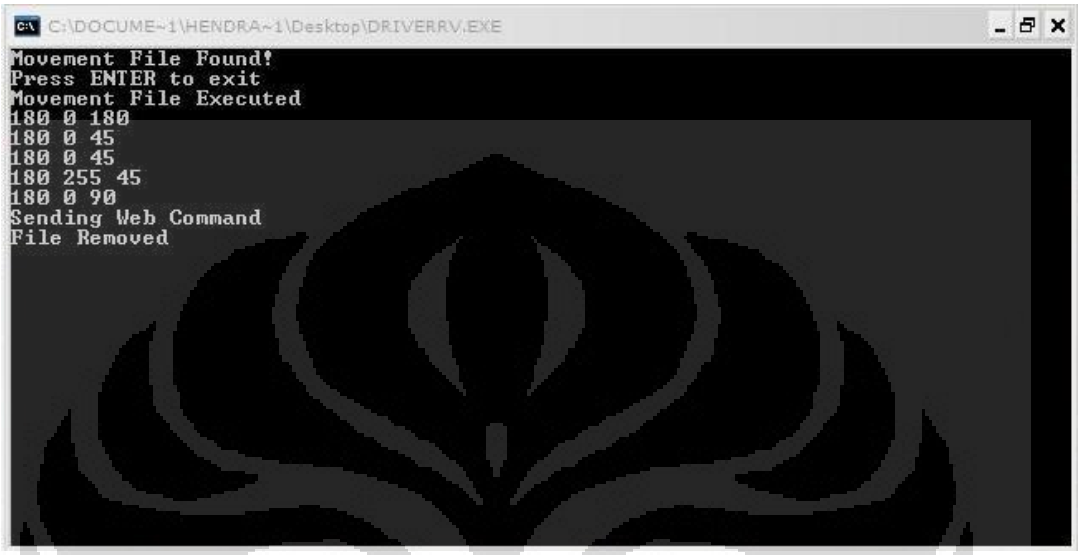


Gambar 6.3 Tampilan Mode *Web*

Apabila *text file* tidak ditemukan maka *software* akan memberitahukan informasi tersebut, *software* akan terus memeriksa keberadaan *text file* dalam interval 1 detik.

Gambar 6.4 Tampilan Mode *Web* Text File Tidak Ditemukan

Apabila *text file* ditemukan, maka software akan langsung mengambil data dan mengirimkannya secara *serial* ke *microcontroller*, menampilkan data yang dibaca dan menghapus *text file* tersebut.

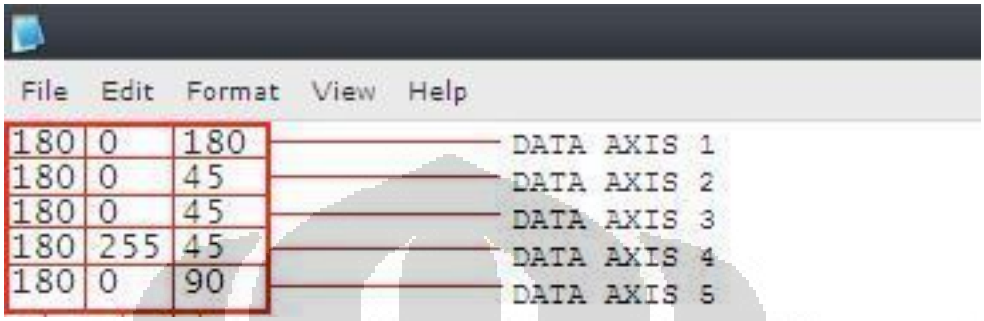


Gambar 6.5 Tampilan Mode *Web* Text File Ditemukan

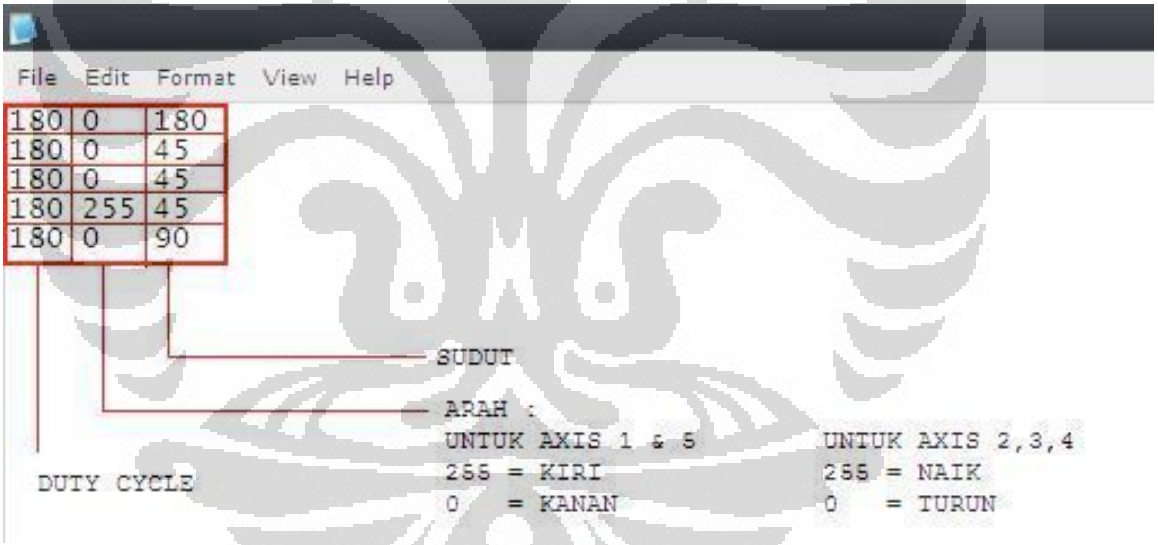
Data yang terdapat dalam *file text* merupakan range angka dari 0 sampai 255, angka 0 sampai 255 dapat dikirim untuk kemudian diolah oleh *microcontroller*. Sehingga didalam *file* kdt.txt terdapat *format* data yang dihasilkan oleh yaitu sebagai berikut :

Gambar 6.6 Format Data Didalam File kdt.txt

Kesepakatan data ini dilakukan untuk mempermudah pembacaan data pada *text file*. Setiap data mempunyai fungsinya masing-masing yang akan dijelaskan oleh gambar berikut :



Gambar 6.7 Paket Data Didalam File kdt.txt

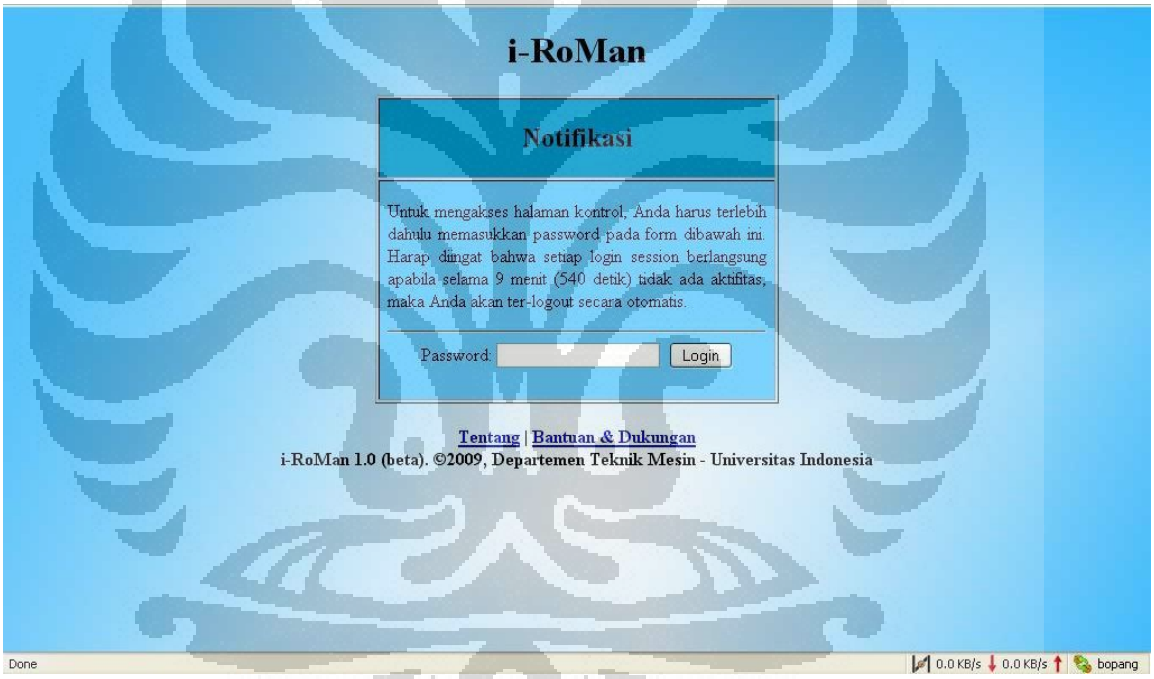


Gambar 6.8 Fungsi Masing-Masing Data Didalam File kdt.txt

Setiap *axis* membutuhkan 3 data tersebut, sehingga apabila dijumlahkan maka terdapat 15 data pada *file* kdt.txt tersebut. Semua nilai ini akan diubah menjadi 0 apabila terdapat kondisi *emergency* sehingga dengan seketika menghentikan gerakan dari robot. Kondisi *emergency* sendiri akan muncul apabila *client* menekan tombol *emergency* pada *web* begitu *client* menganggap kondisi robot perlu dihentikan.

6.2 Web Server

Pada bagian ini tidak dibahas sepenuhnya mengenai *web server* dan *user interface* pada *web*. Hal ini dikarenakan pengembangan *web server* dijelaskan pada buku tersendiri dari peneliti lain yang tergabung dalam group pengembangan kendali robot berbasis *web* ini. Sehingga untuk pembahasan lebih lanjut mengenai *web server* pada pengembangan sistem kontrol robot berbasis *web* ini harus mengacu pada buku tersebut. Untuk memberikan gambaran mengenai *web server* tersebut maka akan disertakan sertakan beberapa gambar *user interface* yang ada pada *web server* tersebut.



Gambar 6.9 *User Login* pada *Web*

Gambar diatas merupakan halaman notifikasi login client, halaman ini diberi *password* untuk menjaga keamanan dan client akan otomatis ter-*logout* apabila selama 9 menit tidak ada aktifitas.

Gambar 6.10 Halaman Kontrol Utama Pada *Web*

Gambar diatas adalah bentuk *user interface* yang ditampilkan dilayar *browser*. Terdapat 2 metode pergerakan robot yaitu *cursor based inverse kinematics* dan *forward kinematic*. Terdapat 2 kamera *webcam* yang memonitor pergerakan robot secara *real-time* untuk membantu *user* yang berada jauh dari robot. User dapat mengaktifasi mode *emergency* apabila terdapat kondisi yang membahayakan atau dapat merusak robot. Dari layar *web* inilah kemudian diproses keluaran berupa file .txt pada komputer *server*, file tersebut yang akan diproses oleh *software interface* pada komputer server dan kemudian diteruskan ke kontrol unit untuk menggerakkan robot. Maka lengkaplah hubungan dari *client* sampai ke robot didalam penelitian kontrol robot berbasis *web* ini.

BAB 7
PENGUJIAN SISTEM

7.1 *Pengujian Pergerakan Axis 1*

Pengujian pergerakan *axis* 1 dapat dilihat pada gambar berikut :

Gambar 7.1 Pengujian *Axis* 1

7.2 *Pengujian Pergerakan Axis 2*

Pengujian pergerakan *axis* 2 dapat dilihat pada gambar berikut :

Gambar 7.2 Pengujian *Axis* 2

7.3 *Pengujian Pergerakan Axis 3*

Pengujian pergerakan *axis* 3 dapat dilihat pada gambar berikut :

Gambar 7.3 Pengujian *Axis* 3

7.4 *Pengujian Pergerakan Axis 4*

Pengujian pergerakan *axis* 4 dapat dilihat pada gambar berikut :

Gambar 7.4 Pengujian *Axis* 4

7.5 *Pengujian Pergerakan Axis 5*

Pengujian pergerakan *axis* 5 dapat dilihat pada gambar berikut :

Gambar 7.5 Pengujian *Axis* 5

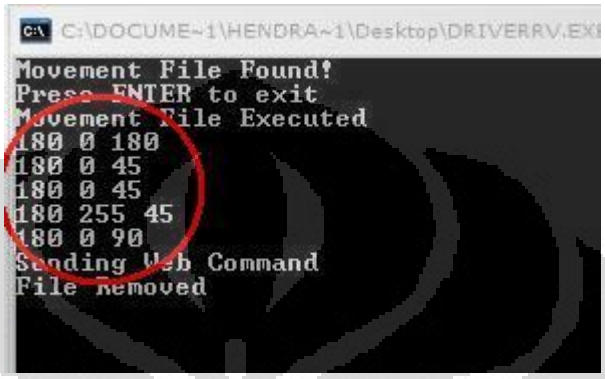
7.6 *Pengujian Pergerakan Simultan*

Pengujian pergerakan simultan *axis* 1 sampai *axis* 5 dapat dilihat pada gambar berikut :

Gambar 7.6 Pengujian Gerak Simultan

7.7 *Akurasi Data*

Data yang dikirim antara *web server* dan *microcontroller* dapat dimonitor dari layar tampilan software. Berikut adalah contoh data yang dibaca software dan dikirim ke microcontroller.



Gambar 7.7 Proses Verifikasi Data

Setelah uji 1000 kali proses pembacaan dan pengiriman data, tidak terjadi kesalahan maupun *error* pada *software*. Arah pergerakan robot dan perhitungan *encoder* sudah tepat. Namun akurasi pergerakan robot masih rendah dikarenakan tidak adanya algoritma kendali adaptif.

Untuk pengujian akurasi data dari microcontroller ke robot dilakukan dengan menampilkan data perhitungan encoder pada LCD setelah robot bergerak. Berikut adalah data yang berhasil diambil dari robot :

1. Axis 1 : memiliki kalibrasi sudut 1° = 57 pulsa encoder. Dengan pergerakan robot -180°, data encoder yang didapat sebesar 10374 pulsa. Sehingga didapat error pergerakan sebesar ± 2°.
2. Axis 2 : memiliki kalibrasi sudut 1° = 101 pulsa encoder. Dengan pergerakan robot -45°, data encoder yang didapat sebesar 5050 pulsa. Sehingga didapat error pergerakan sebesar ± 5°.

3. Axis 3 : memiliki kalibrasi sudut 1° = 67 pulsa encoder. Dengan pergerakan robot -45°, data encoder yang didapat sebesar 3216 pulsa. Sehingga didapat error pergerakan sebesar ± 3°.
4. Axis 4 : memiliki kalibrasi sudut 1° = 50 pulsa encoder. Dengan pergerakan robot 45°, data encoder yang didapat sebesar 2350 pulsa. Sehingga didapat error pergerakan sebesar ± 2°.
5. Axis 5 : memiliki kalibrasi sudut 1° = 30 pulsa encoder. Dengan pergerakan robot 90°, data encoder yang didapat sebesar 2730 pulsa. Sehingga didapat error pergerakan sebesar ± 1°.

Adanya error ini disebabkan karena robot dipengaruhi oleh gaya-gaya yang didapat oleh robot sepeti inersia pada sistem mekanikal robot. Error ini dapat dikurangi dengan menambahkan algoritma kendali kecepatan motor yang adaptif.

BAB 8
KESIMPULAN & SARAN PENELITIAN LEBIH LANJUT

8.1 *Kesimpulan*

1. Pengendalian robot berbasis *web* sangat bergantung pada akurasi data yang diterima dan dikirim antara masing – masing sistem.
2. Pengendalian robot secara digital memungkinkan berbagai macam algoritma kendali untuk dikembangkan dalam pengendalian robot.
3. Sistem kendali robot berbasis *web* dapat diterapkan pada aplikasi industri yang membutuhkan pemantauan pergerakan robot.
4. Sistem *emergency* dalam pengendalian robot berbasis *web* harus dikembangkan secara optimal untuk menjamin keamanan sistem elektronik dan mekanikal.

8.2 *Saran Penelitian Lebih Lanjut*

Untuk pengembangan robot lebih lanjut disarankan untuk melakukan beberapa perubahan untuk meningkatkan kinerja sistem, yaitu :

1. Motor driver L298 dapat ditingkatkan kemampuannya dengan memberikan heatsink atau dengan menggantinya dengan driver motor yang memiliki daya tahan lebih.
2. Meningkatkan kecepatan proses data dengan memakai microcontroller dengan kecepatan lebih tinggi.
3. Control unit dapat dibuat menjadi *single board* saja untuk mengurangi pemakaian kabel sehingga dapat mengurangi resiko kesalahan teknis akibat putusnya kabel.
4. Pemakaian pulsa *encoder* fasa B & Z akan meningkatkan akurasi robot.

DAFTAR REFERENSI

[1] APJII(December 2007). Perkembangan Jumlah Pelanggan & Pemakai Internet (kumalatif). 19 February 2009. http://www.apjii.or.id

[2] ISO 8373(1994). Manipulating Industrial Robot. 20 July 2009. http://www.dira.dk/Portals/0/Robotter/robotdef.pdf

[3] Mitshubishi. (1998). Movemaster RV-M1 Instruction Manual. Japan

[4] DigiKey. (2003). ATmega2560 Microcontroller Package. April 27, 2009. www.digikey.com

[5] Society of Robots. (November 30, 2008). Pulse Width Modulation Tutorial. June 2009. http://www.societyofrobots.com/member_tutorials/node/229

[6] Quantum Devices. (March 27, 2009). What is meant by Rotary Incremental Encoder Index Pulse "gating"?. July 10, 2009. http://quantumdevices.wordpress.com/2009/03/27/what-is-meant-by-rotray-incremental-encoder-gating

[7] Wankhede, Mahesh. (2007). Switch On I/O Ports. August 20, 2009. http://www.freewebs.com/maheshwankhede/ports.html

[8] ATMEL. ATMega2560 Microcontroller PDF. April 3, 2009. http://www.atmel.com/dyn/products/product_card.asp?family_id=607&family_name=AVR+8%2DBit+RISC+&part_id=2014

[9] Ucables. (2006). 16mhz Smd Crystal Oscillator. December 21, 2009. http://ucables.com/ref/16MHZ-SMD-CRYSTAL-OSC-R46887

[10] ST. L298D Datasheet. August 23, 2009. http://www.datasheetcatalog.com/datasheets_pdf/T/I/P/1/TIP122.shtml

[11] Sains, Insan (2008). H-Bridge Driver : Kontrol Arah Motor. January 22, 2009. http://insansainsprojects.wordpress.com/2008/06/05/h-bridge-driver-kontrol-arah-motor

[12] ST. TIP122 Datasheet. June 8, 2009. http://www.datasheetcatalog.com/datasheets_pdf/T/I/P/1/TIP122.shtml

[13] Lumax Incorporated. LCM-S01604DSR Datasheet, May 19, 2009. http://www.lumex.com/products.aspx?id=11

[14] Pololu. USB to Serial Adaptor. March 27, 2009.
http://www.pololu.com/catalog/product/391

[15] Meanwell. S-100F PSU Datasheet. February 20, 2009.
http://www.meanwell.com/search/s-100f/default.htm

[16] Dietel & Dietel. (2003). How to Program in C. 3rd edition. New Jersey: Prentice Hall.

[17] Codevision. Codevision AVR C compilier. January 10, 2009.
http://www.hpinfotech.ro/html/cvavr.htm

LAMPIRAN 1

```
/*****************************************
Program Control Robot Manipulator 5 DOF
Author              : Hendra Prima Syahputra
Chip type           : ATmega2560
Program type        : Application
Clock frequency     : 16.000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 2048
*****************************************/

#include <mega2560.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <lcd.h>

#define MAX 0xFF
#define kanan 1
#define kiri 0
#define stop 2
#define naik 1
#define turun 0
#define buka 1
#define tutup 0

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

int encoder_axis_1, encoder_axis_2, encoder_axis_3, encoder_axis_4,
encoder_axis_5;

char lcd_encoder_axis_1[5],
     lcd_encoder_axis_2[5],
     lcd_encoder_axis_3[5],
     lcd_encoder_axis_4[5],
     lcd_encoder_axis_5[5];

int perintah;

int duty_cycle_axis_1 = 0;
int arah_axis_1 = 0;
int angle_axis_1 = 0;
int duty_cycle_axis_2 = 0;
int arah_axis_2 = 0;
int angle_axis_2 = 0;
```

```
int duty_cycle_axis_3 = 0;
int arah_axis_3 = 0;
int angle_axis_3 = 0;
int duty_cycle_axis_4 = 0;
int arah_axis_4 = 0;
int angle_axis_4 = 0;
int duty_cycle_axis_5 = 0;
int arah_axis_5 = 0;
int angle_axis_5 = 0;

/*
Kirim Text
*/
void kirim_char_serial_2(char flash *fmtstr,...)
{
    while ((*fmtstr != '\0'))
    {
      while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
      UDR2 = *fmtstr;
      fmtstr++;
    }
}

/*
Tq to Erwin > Kirim Nilai
*/
void kirim_nilai_serial_2(char *str)
{
    while(*str != '\0')
    {
        while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
        UDR2=*str;
        str = str + 1;
    }
}

/*
    %d - decimal
    %u - unsigned decimal
    %o - octal
    %x - hex
    %c - character
    %s - strings
*/

void lcd_setting()
{
    /*
    LCD on port C
    */

    #asm
    .equ __lcd_port=0x08 ;PORTC
    #endasm
}
```

```
void oscillator_setting()
{
    /*
    Crystal Division Factor : 1
    */

    #pragma optsize-
    CLKPR=0x80;
    CLKPR=0x00;
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif
}

void usart_2_setting()
{
    /*
    USART2 initialization
    Communication Parameters: 8 Data, 1 Stop, No Parity
    USART2 Receiver: On
    USART2 Transmitter: On
    USART2 Mode: Asynchronous
    USART2 Baud Rate: 115200
    */

    UCSR2A=0x00;
    UCSR2B=0x18;
    UCSR2C=0x06;
    UBRR2H=0x00;
    UBRR2L=0x08;
}

void ports_setting()
{
     /*
    board configuration

        F0F7K0K7
    E0              A0
    E7              A7
    H0              J7
    H7     uC       J0
    B4              C7
    B7              C0
    G0
    G5
        L0L7D0D7
    */
    PORTF =0b00000000;
    DDRF  =0b00001111;
    //        ||||||||
    //        |||||||\___0: brake_axis_2
    //        ||||||\____1: brake_axis_3
    //        |||||\_____2: input_1_axis_5
    //        ||||\______3: input_2_axis_5
    //        |||\_______4: limit_switch_1
    //        ||\________5: limit_switch_2
    //        |\_________6: limit_switch_3
    //        \__________7: limit_switch_4
```

```
//DDRK     = 0xFF;
//PORTK    = 0x00;

PORTA= 0b00000000;
DDRA = 0b11111111;
//        ||||||||
//        |||||||\___0: input_1_axis_1
//        ||||||\____1: input_2_axis_1
//        |||||\_____2: input_1_axis_2
//        ||||\______3: input_2_axis_2
//        |||\_______4: input_1_axis_3
//        ||\________5: input_2_axis_3
//        |\_________6: input_1_axis_4
//        \__________7: input_2_axis_4

PORTB= 0b00000000;
DDRB = 0b11100000;
//        ||||||||
//        |||||||\___0:
//        ||||||\____1:
//        |||||\_____2:
//        ||||\______3:
//        |||\_______4: limit_switch_5
//        ||\________5: PWM_axis_4
//        |\_________6: PWM_axis_5
//        \__________7: PWM_gripper

//PORTC
//DDRC = 0b11111111;
//        ||||||||
//        |||||||\___0: LCD_D0
//        ||||||\____1: LCD_D1
//        |||||\_____2: LCD_D2
//        ||||\______3:
//        |||\_______4: LCD_D4
//        ||\________5: LCD_D5
//        |\_________6: LCD_D6
//        \__________7: LCD_D7

PORTD= 0b00000000;
DDRD = 0b00110001;
//        ||||||||
//        |||||||\___0: input_1_gripper
//        ||||||\____1: encoder_axis_1
//        |||||\_____2: encoder_axis_2
//        ||||\______3: encoder_axis_3
//        |||\_______4: input_2_gripper
//        ||\________5: LED
//        |\_________6:
//        \__________7:
```

```
    PORTE= 0b11111111;
    DDRE  =0b00000000;
    //        ||||||||
    //        |||||||\___0:
    //        ||||||\____1:
    //        |||||\_____2:
    //        ||||\______3:
    //        |||\_______4: encoder_axis_4
    //        ||\________5: encoder_axis_5
    //        |\_________6:
    //        \__________7:

    //PORTG= 0b11111111;
    //DDRG = 0b00000000;
    //        ||||||||
    //        |||||||\___0:
    //        ||||||\____1:
    //        |||||\_____2:
    //        ||||\______3:
    //        |||\_______4:
    //        ||\________5:
    //        |\_________6:
    //        \__________7:

    //PORTH= 0b00000000;
    DDRH = 0b00000000;
    //        ||||||||
    //        |||||||\___0: RX_usart_2
    //        ||||||\____1: TX_usart_2
    //        |||||\_____2:
    //        ||||\______3:
    //        |||\_______4:
    //        ||\________5:
    //        |\_________6:
    //        \__________7:

    //PORTJ
    //DDRJ = 0b11111111;
    //        ||||||||
    //        |||||||\___0:
    //        ||||||\____1:
    //        |||||\_____2:
    //        ||||\______3:
    //        |||\_______4:
    //        ||\________5:
    //        |\_________6:
    //        \__________7:

    //PORTL= 0b11111111;
    DDRL = 0b11111111;
    //        ||||||||
    //        |||||||\___0:
    //        ||||||\____1:
    //        |||||\_____2:
    //        ||||\______3: PWM_axis_1
    //        |||\_______4: PWM_axis_2
    //        ||\________5: PWM_axis_3
    //        |\_________6:
    //        \__________7:
}
```

```
void timers_setting()
{
    /*
    Timer/Counter 1 initialization
    Clock source: System Clock
    Clock value: 2000.000 kHz
    Mode: Fast PWM top=00FFh
    OC1A output: Non-Inv.
    OC1B output: Non-Inv.
    OC1C output: Non-Inv.
    Noise Canceler: On
    Input Capture on Falling Edge
    Timer 1 Overflow Interrupt: Off
    Input Capture Interrupt: Off
    Compare A Match Interrupt: Off
    Compare B Match Interrupt: Off
    Compare C Match Interrupt: Off
    */
    TCCR1A=0xA9;
    TCCR1B=0x8A;
    TCNT1H=0x00;
    TCNT1L=0x00;
    ICR1H=0x00;
    ICR1L=0x00;
    OCR1AH=0x00;
    OCR1AL=0x00;
    OCR1BH=0x00;
    OCR1BL=0x00;
    OCR1CH=0x00;
    OCR1CL=0x00;
    /*
    Timer/Counter 5 initialization
    Clock source: System Clock
    Clock value: 2000.000 kHz
    Mode: Fast PWM top=00FFh
    OC5A output: Non-Inv.
    OC5B output: Non-Inv.
    OC5C output: Non-Inv.
    Noise Canceler: On
    Input Capture on Falling Edge
    Timer 5 Overflow Interrupt: Off
    Input Capture Interrupt: Off
    Compare A Match Interrupt: Off
    Compare B Match Interrupt: Off
    Compare C Match Interrupt: Off
    */
    TCCR5A=0xA9;
    TCCR5B=0x8A;
    TCNT5H=0x00;
    TCNT5L=0x00;
    ICR5H=0x00;
    ICR5L=0x00;
    OCR5AH=0x00;
    OCR5AL=0x00;
    OCR5BH=0x00;
    OCR5BL=0x00;
    OCR5CH=0x00;
    OCR5CL=0x00;
}
```

```
void interrupt_setting()
{
    /*
    External Interrupt(s) initialization
    INT0: On
    INT0 Mode: Rising Edge
    INT1: On
    INT1 Mode: Rising Edge
    INT2: On
    INT2 Mode: Rising Edge
    INT3: On
    INT3 Mode: Rising Edge
    INT4: On
    INT4 Mode: Rising Edge
    INT5: On
    INT5 Mode: Rising Edge
    INT6: On
    INT6 Mode: Rising Edge
    INT7: On
    INT7 Mode: Rising Edge
    */
    EICRA=0xFF;
    EICRB=0xFF;
    EIMSK=0xFF;
    EIFR=0xFF;
}

void brake_axis_2_off()
{
    PORTF.0 = 1;
}

void brake_axis_3_off()
{
    PORTF.1 = 1;
}

void brake_axis_2_on()
{
    PORTF.0 = 0;
}

void brake_axis_3_on()
{
    PORTF.1 = 0;
}


void motor_control
(
    float duty_cycle_axis_1, int arah_axis_1,
    float duty_cycle_axis_2, int arah_axis_2,
    float duty_cycle_axis_3, int arah_axis_3,
    float duty_cycle_axis_4, int arah_axis_4,
    float duty_cycle_axis_5, int arah_axis_5
)
{
    int input_1_axis_1, input_2_axis_1;
    int input_1_axis_2, input_2_axis_2;
```

```
int input_1_axis_3, input_2_axis_3;
int input_1_axis_4, input_2_axis_4;
int input_1_axis_5, input_2_axis_5;

//axis_1
if (arah_axis_1 == kanan)
{
    input_1_axis_1 = 1;
    input_2_axis_1 = 0;
}

if (arah_axis_1 == kiri)
{
    input_1_axis_1 = 0;
    input_2_axis_1 = 1;
}
if (arah_axis_1 == stop)
{
    input_1_axis_1 = 0;
    input_2_axis_1 = 0;
}

//axis_2
if (arah_axis_2 == naik)
{
    input_1_axis_2 = 1;
    input_2_axis_2 = 0;
}

if (arah_axis_2 == turun)
{
    input_1_axis_2 = 0;
    input_2_axis_2 = 1;
}

if (arah_axis_2 == stop)
{
    input_1_axis_2 = 0;
    input_2_axis_2 = 0;
}

//axis_3
if (arah_axis_3 == naik)
{
    input_1_axis_3 = 1;
    input_2_axis_3 = 0;
}

if (arah_axis_3 == turun)
{
    input_1_axis_3 = 0;
    input_2_axis_3 = 1;
}
if (arah_axis_3 == stop)
{
    input_1_axis_3 = 0;
    input_2_axis_3 = 0;
}
```

```
    //axis_4
    if (arah_axis_4 == naik)
    {
        input_1_axis_4 = 1;
        input_2_axis_4 = 0;
    }

    if (arah_axis_4 == turun)
    {
        input_1_axis_4 = 0;
        input_2_axis_4 = 1;
    }

    if (arah_axis_4 == stop)
    {
        input_1_axis_4 = 0;
        input_2_axis_4 = 0;
    }

    //axis_5
    if (arah_axis_5 == kanan)
    {
        input_1_axis_5 = 1;
        input_2_axis_5 = 0;
    }

    if (arah_axis_5 == kiri)
    {
        input_1_axis_5 = 0;
        input_2_axis_5 = 1;
    }
    if (arah_axis_5 == stop)
    {
        input_1_axis_5 = 0;
        input_2_axis_5 = 0;
    }

    PORTA.0 = input_1_axis_1;
    PORTA.1 = input_2_axis_1;
    PORTA.2 = input_1_axis_2;
    PORTA.3 = input_2_axis_2;
    PORTA.4 = input_1_axis_3;
    PORTA.5 = input_2_axis_3;
    PORTA.6 = input_1_axis_4;
    PORTA.7 = input_2_axis_4;
    PORTF.2 = input_1_axis_5;
    PORTF.3 = input_2_axis_5;

    OCR5AH = MAX*duty_cycle_axis_1;
    OCR5AL = MAX*duty_cycle_axis_1;
    OCR5BH = MAX*duty_cycle_axis_2;
    OCR5BL = MAX*duty_cycle_axis_2;
    OCR5CH = MAX*duty_cycle_axis_3;
    OCR5CL = MAX*duty_cycle_axis_3;
    OCR1AH = MAX*duty_cycle_axis_4;
    OCR1AL = MAX*duty_cycle_axis_4;
    OCR1BH = MAX*duty_cycle_axis_5;
    OCR1BL = MAX*duty_cycle_axis_5;
}
```

```
interrupt [INT1] void ext_int1_isr(void)
{
    encoder_axis_1++;
}

interrupt [INT2] void ext_int2_isr(void)
{
    encoder_axis_2++;
}

interrupt [INT3] void ext_int3_isr(void)
{
    encoder_axis_3++;
}



interrupt [INT4] void ext_int4_isr(void)
{
    encoder_axis_4++;
}

interrupt [INT5] void ext_int5_isr(void)
{
    encoder_axis_5++;
}

void display_encoder_axis_1()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_1,lcd_encoder_axis_1);
    lcd_puts(lcd_encoder_axis_1);
}

void display_encoder_axis_2()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_2,lcd_encoder_axis_2);
    lcd_puts(lcd_encoder_axis_2);
}

void display_encoder_axis_3()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_3,lcd_encoder_axis_3);
    lcd_puts(lcd_encoder_axis_3);
}

void display_encoder_axis_4()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_4,lcd_encoder_axis_4);
    lcd_puts(lcd_encoder_axis_4);
}
void display_encoder_axis_5()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_5,lcd_encoder_axis_5);
    lcd_puts(lcd_encoder_axis_5);
}
```

```
void stop_axis_1()
{
    PORTA.0 = 0;
    PORTA.1 = 0;
    OCR5AH = 0x00;
    OCR5AL = 0x00;
    encoder_axis_1 = 0;
}

void stop_axis_2()
{
    PORTA.2 = 0;
    PORTA.3 = 0;
    OCR5BH = 0x00;
    OCR5BL = 0x00;
    encoder_axis_2 = 0;
    brake_axis_2_on();
}

void stop_axis_3()
{
    PORTA.4 = 0;
    PORTA.5 = 0;
    OCR5CH = 0x00;
    OCR5CL = 0x00;
    encoder_axis_3 = 0;
    brake_axis_3_on();
}

void stop_axis_4()
{
    PORTA.6 = 0;
    PORTA.7 = 0;
    OCR1AH = 0x00;
    OCR1AL = 0x00;
    encoder_axis_4 = 0;
}

void stop_axis_5()
{
    PORTF.2 = 0;
    PORTF.3 = 0;
    OCR1BH = 0x00;
    OCR1BL = 0x00;
    encoder_axis_5 = 0;
}

void move_axis_1_no_stop(int duty_cycle_axis_1, int arah_axis_1)
{
    int input_1_axis_1, input_2_axis_1;

    if (arah_axis_1 == kanan)
    {
        input_1_axis_1 = 1;
        input_2_axis_1 = 0;
    }
```

```
    if (arah_axis_1 == kiri)
    {
        input_1_axis_1 = 0;
        input_2_axis_1 = 1;
    }
    if (arah_axis_1 == stop)
    {
        input_1_axis_1 = 0;
        input_2_axis_1 = 0;
    }

    PORTA.0 = input_1_axis_1;
    PORTA.1 = input_2_axis_1;

    OCR5AH = duty_cycle_axis_1;
    OCR5AL = duty_cycle_axis_1;
}

void move_axis_2_no_stop(int duty_cycle_axis_2, int arah_axis_2)
{
    int input_1_axis_2, input_2_axis_2;

    if (arah_axis_2 == naik)
    {
        input_1_axis_2 = 1;
        input_2_axis_2 = 0;
    }

    if (arah_axis_2 == turun)
    {
        input_1_axis_2 = 0;
        input_2_axis_2 = 1;
    }
    if (arah_axis_2 == stop)
    {
        input_1_axis_2 = 0;
        input_2_axis_2 = 0;
    }

    PORTA.2 = input_1_axis_2;
    PORTA.3 = input_2_axis_2;

    OCR5BH = duty_cycle_axis_2;
    OCR5BL = duty_cycle_axis_2;

    brake_axis_2_off();
}

void move_axis_3_no_stop(int duty_cycle_axis_3, int arah_axis_3)
{
    int input_1_axis_3, input_2_axis_3;

    if (arah_axis_3 == naik)
    {
        input_1_axis_3 = 1;
        input_2_axis_3 = 0;
    }
```

```
    if (arah_axis_3 == turun)
    {
        input_1_axis_3 = 0;
        input_2_axis_3 = 1;
    }
    if (arah_axis_3 == stop)
    {
        input_1_axis_3 = 0;
        input_2_axis_3 = 0;
    }

    PORTA.4 = input_1_axis_3;
    PORTA.5 = input_2_axis_3;

    OCR5CH = duty_cycle_axis_3;
    OCR5CL = duty_cycle_axis_3;

    brake_axis_3_off();
}

void move_axis_4_no_stop(int duty_cycle_axis_4, int arah_axis_4)
{
    int input_1_axis_4, input_2_axis_4;

     if (arah_axis_4 == naik)
    {
        input_1_axis_4 = 1;
        input_2_axis_4 = 0;
    }

    if (arah_axis_4 == turun)
    {
        input_1_axis_4 = 0;
        input_2_axis_4 = 1;
    }
    if (arah_axis_4 == stop)
    {
        input_1_axis_4 = 0;
        input_2_axis_4 = 0;
    }

    PORTA.6 = input_1_axis_4;
    PORTA.7 = input_2_axis_4;

    OCR1AH = duty_cycle_axis_4;
    OCR1AL = duty_cycle_axis_4;
}

void move_axis_5_no_stop(int duty_cycle_axis_5, int arah_axis_5)
{
    int input_1_axis_5, input_2_axis_5;

    if (arah_axis_5 == kanan)
    {
        input_1_axis_5 = 1;
        input_2_axis_5 = 0;
    }
```

```
    if (arah_axis_5 == kiri)
    {
        input_1_axis_5 = 0;
        input_2_axis_5 = 1;
    }
    if (arah_axis_5 == stop)
    {
        input_1_axis_5 = 0;
        input_2_axis_5 = 0;
    }

    PORTF.2 = input_1_axis_5;
    PORTF.3 = input_2_axis_5;

    OCR1BH = duty_cycle_axis_5;
    OCR1BL = duty_cycle_axis_5;
}

void move_all_axis
(
    int duty_cycle_axis_1, int arah_axis_1, int angle_axis_1,
    int duty_cycle_axis_2, int arah_axis_2, int angle_axis_2,
    int duty_cycle_axis_3, int arah_axis_3, int angle_axis_3,
    int duty_cycle_axis_4, int arah_axis_4, int angle_axis_4,
    int duty_cycle_axis_5, int arah_axis_5, int angle_axis_5
)
{
    int encoder_axis_1_pulse, encoder_axis_2_pulse, encoder_axis_3_pulse,
    encoder_axis_4_pulse, encoder_axis_5_pulse;
    #asm ("sei");
    encoder_axis_1_pulse = angle_axis_1 * 56;
    encoder_axis_2_pulse = angle_axis_2 * 100;
    encoder_axis_3_pulse = angle_axis_3 * 67;
    encoder_axis_4_pulse = angle_axis_4 * 50;
    encoder_axis_5_pulse = angle_axis_5 * 30;

    move_axis_1_no_stop(duty_cycle_axis_1,arah_axis_1);
    move_axis_2_no_stop(duty_cycle_axis_2,arah_axis_2);
    move_axis_3_no_stop(duty_cycle_axis_3,arah_axis_3);
    move_axis_4_no_stop(duty_cycle_axis_4,arah_axis_4);
    move_axis_5_no_stop(duty_cycle_axis_5,arah_axis_5);

    while(1)
    {
        if (encoder_axis_1 >= encoder_axis_1_pulse)
        {
            stop_axis_1();
        }

        if (encoder_axis_2 >= encoder_axis_2_pulse)
        {
            stop_axis_2();
        }

        if (encoder_axis_3 >= encoder_axis_3_pulse)
        {
            stop_axis_3();
        }
```

```
        if (encoder_axis_4 >= encoder_axis_4_pulse)
        {
            stop_axis_4();
        }

        if (encoder_axis_5 >= encoder_axis_5_pulse)
        {
            stop_axis_5();
        }

if ( PORTA.0 == 0 && PORTA.1 == 0 && PORTA.2 == 0 && PORTA.3 == 0 && PORTA.4
== 0 && PORTA.5 == 0 && PORTA.6 == 0 && PORTA.7 == 0 && PORTF.2 == 0 &&
PORTF.3 == 0)
        {
            break;
        }
    }
    #asm ("cli");
}

void default_position()
{
    while( PINF.5 != 1 )
    {
        brake_axis_2_off();
        motor_control(0,kanan,0.8,naik,0,stop,0,stop,0,stop);
    }
    brake_axis_2_on();
    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);

    while( PINF.4 != 1 )
    {
        motor_control(0.5,kanan,0,stop,0,stop,0,stop,0,stop);
    }

    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);

    while( PINF.6 != 1 )
    {
        brake_axis_3_off();
        motor_control(0,kanan,0,stop,0.7,naik,0,stop,0,stop);
    }
    brake_axis_3_on();
    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);

    while( PINF.7 != 1 )
    {
        motor_control(0,stop,0,stop,0,stop,0.7,turun,0,stop);
    }
    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);

    while( PINB.4 != 1 )
    {
        motor_control(0,stop,0,stop,0,stop,0,stop,0.5,kanan);
    }
    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);
}
```

```
void gripper_control(float gripper_speed, int gripper_status)
{
    int input_1_gripper, input_2_gripper;

    //gripper
    if (gripper_status == buka)
    {
        input_1_gripper = 1;
        input_2_gripper = 0;
    }

    if (gripper_status == tutup)
    {
        input_1_gripper = 0;
        input_2_gripper = 1;
    }
    if (gripper_status == stop)
    {
        input_1_gripper = 0;
        input_2_gripper = 0;
    }

   PORTD.0 = input_1_gripper;
   PORTD.4 = input_2_gripper;

   OCR1CH = MAX*gripper_speed;
   OCR1CL = MAX*gripper_speed;
}

void object_grabbing()
{
    default_position();

move_all_axis(0.6,kiri,190,0.4,turun,70,0.4,turun,40,0.5,naik,30,0.5,kiri,90
);
    delay_ms(1000);
    move_all_axis(0,stop,0,0,stop,0,0.7,turun,25,0.7,naik,15,0,stop,0);
    delay_ms(1000);
    gripper_control(0.6,tutup);
    delay_ms(650);
    move_all_axis(0.5,kiri,60,0,stop,0,0.8,naik,25,1,turun,5,1,kiri,90);
    delay_ms(1000);
    move_all_axis(0,stop,0,0,stop,0,0,stop,0,0,stop,0,0.5,kanan,180);
    delay_ms(5000);
    move_all_axis(0,stop,0,0,stop,0,0,stop,0,0,stop,0,0.5,kiri,180);
    delay_ms(1000);
    move_all_axis(0,stop,0,0,stop,0,0.7,turun,25,0.7,naik,15,0,stop,0);
    delay_ms(1000);
    gripper_control(0.6,buka);
    delay_ms(650);
    default_position();
}
```

```
void move_all_axis_serial()
{
    kirim_char_serial_2("Ready To Recieve Value, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            duty_cycle_axis_1 = UDR2;
            break;
        }
    }
    kirim_char_serial_2("duty_cycle_axis_1 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            arah_axis_1 = UDR2;

            switch(arah_axis_1)
            {
                case 255 :
                arah_axis_1 = kanan;
                break;

                case 0 :
                arah_axis_1 = kiri;
                break;
            }
            break;
        }
    }
    kirim_char_serial_2("arah_axis_1 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            angle_axis_1 = UDR2;
            break;
        }
    }
    kirim_char_serial_2("angle_axis_1 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            duty_cycle_axis_2 = UDR2;
            break;
        }
    }
    kirim_char_serial_2("duty_cycle_axis_2 Recieved, ENTER to continue\n");
```

```
   while(1)
   {
    if(UCSR2A & 0x80)
    {
        arah_axis_2 = UDR2;

        switch(arah_axis_2)
        {
            case 255 :
            arah_axis_2 = naik;
            break;

            case 0 :
            arah_axis_2 = turun;
            break;
        }
        break;
    }
}
kirim_char_serial_2("arah_axis_2 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_2 = UDR2;
        break;
    }
}
kirim_char_serial_2("angle_axis_2 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_3 = UDR2;
        break;
    }
}
kirim_char_serial_2("duty_cycle_axis_3 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_3 = UDR2;

        switch(arah_axis_3)
        {
            case 255 :
            arah_axis_3 = naik;
            break;

            case 0 :
            arah_axis_3 = turun;
            break;
        }
        break;
    }
}
```

```
kirim_char_serial_2("arah_axis_3 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_3 = UDR2;
        break;
    }
}
kirim_char_serial_2("angle_axis_3 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_4 = UDR2;
        break;
    }
}
kirim_char_serial_2("duty_cycle_axis_4 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_4 = UDR2;

        switch(arah_axis_4)
        {
            case 255 :
            arah_axis_4 = naik;
            break;

            case 0 :
            arah_axis_4 = turun;
            break;
        }
        break;
    }
}
kirim_char_serial_2("arah_axis_4 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_4 = UDR2;
        break;
    }
}
kirim_char_serial_2("angle_axis_4 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_5 = UDR2;
        break;
    }
```

```
    }
    kirim_char_serial_2("duty_cycle_axis_5 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            arah_axis_5 = UDR2;

            switch(arah_axis_5)
            {
                case 255 :
                arah_axis_5 = kanan;
                break;

                case 0 :
                arah_axis_5 = kiri;
                break;
            }
            break;
        }
    }
    kirim_char_serial_2("arah_axis_5 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            angle_axis_5 = UDR2;
            break;
        }
    }
    kirim_char_serial_2("angle_axis_2 Recieved, ENTER to continue\n");

    move_all_axis
    (
        duty_cycle_axis_1, arah_axis_1, angle_axis_1,
        duty_cycle_axis_2, arah_axis_2, angle_axis_2,
        duty_cycle_axis_3, arah_axis_3, angle_axis_3,
        duty_cycle_axis_4, arah_axis_4, angle_axis_4,
        duty_cycle_axis_5, arah_axis_5, angle_axis_5
    );
}

void web_based()
{
            int default_position_activate = 0;

            while(1)
            {
                if(UCSR2A & 0x80)
                {
                    duty_cycle_axis_1 = UDR2;
                    break;
                }
            }
```

```
while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_1 = UDR2;

        switch(arah_axis_1)
        {
            case 255 :
            arah_axis_1 = kanan;
            break;

            case 0 :
            arah_axis_1 = kiri;
            break;
        }
    break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_1 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_2 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_2 = UDR2;

        switch(arah_axis_2)
        {
            case 255 :
            arah_axis_2 = naik;
            break;

            case 0 :
            arah_axis_2 = turun;
            break;
        }
        break;
    }
}
```

```
 while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_2 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_3 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_3 = UDR2;

        switch(arah_axis_3)
        {
            case 255 :
            arah_axis_3 = naik;
            break;

            case 0 :
            arah_axis_3 = turun;
            break;
        }
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_3 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_4 = UDR2;
        break;
    }
}
```

```
while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_4 = UDR2;

        switch(arah_axis_4)
        {
            case 255 :
            arah_axis_4 = naik;
            break;

            case 0 :
            arah_axis_4 = turun;
            break;
        }
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_4 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_5 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_5 = UDR2;

        switch(arah_axis_5)
        {
            case 255 :
            arah_axis_5 = kanan;
            break;

            case 0 :
            arah_axis_5 = kiri;
            break;
        }
        break;
    }
}

while(1)
{
```

```
                if(UCSR2A & 0x80)
                {
                    angle_axis_5 = UDR2;
                    break;
                }
            }

            if
            (
                duty_cycle_axis_1 == 1 && arah_axis_1 == 1 &&
                angle_axis_1 == 1 &&
                duty_cycle_axis_2 == 1 && arah_axis_2 == 1 &&
                angle_axis_2 == 1 &&
                duty_cycle_axis_3 == 1 && arah_axis_3 == 1 &&
                angle_axis_3 == 1 &&
                duty_cycle_axis_4 == 1 && arah_axis_4 == 1 &&
                angle_axis_4 == 1 &&
                duty_cycle_axis_5 == 1 && arah_axis_5 == 1 &&
                angle_axis_5 == 1
            )
            {
                default_position();
                default_position_activate = 1;
            }

            if( default_position_activate == 0)
            {
                move_all_axis
                (
                    duty_cycle_axis_1, arah_axis_1, angle_axis_1,
                    duty_cycle_axis_2, arah_axis_2, angle_axis_2,
                    duty_cycle_axis_3, arah_axis_3, angle_axis_3,
                    duty_cycle_axis_4, arah_axis_4, angle_axis_4,
                    duty_cycle_axis_5, arah_axis_5, angle_axis_5
                );
            }
}

void led_indicator_off()
{
    PORTD.5 = 0;
}

void led_indicator_on()
{
    PORTD.5 = 1;
}

void main(void)
{
    oscillator_setting();
    ports_setting();
    timers_setting();
    interrupt_setting();
    usart_2_setting();

    while(1)
    {
        int executed_value = 0;
        int perintah;
```

```
            lcd_init(16);

            lcd_setting();
            lcd_gotoxy(0,0);
            lcd_putsf("Robot Initialize");

            delay_ms(3000);

            lcd_clear();

            lcd_gotoxy(0,0);
            lcd_putsf("Ready to Execute");
            delay_ms(3000);

            lcd_clear();

            led_indicator_on();

            while(executed_value == 0)
            {
                if(UCSR2A & 0x80)
                {
                    perintah = UDR2;

                    switch(perintah)
                    {
                        case 255 :
                        default_position();
                        kirim_char_serial_2("Default Position Executed,
                        press ENTER to continue\n");
                        executed_value = 1;
                        led_indicator_off();
                        break;

                        case 254 :
                        move_all_axis_serial();
                        executed_value = 1;
                        led_indicator_off();
                        break;

                        case 253 :
                        lcd_gotoxy(0,0);
                        lcd_putsf("Web Based Mode");
                        web_based();
                        executed_value = 1;
                        led_indicator_off();
                        break;

                        case 252 :
                        executed_value = 1;
                        led_indicator_off();
                        break;
                    }
                }
            }
        }
}
```

LAMPIRAN 2

```
/*************************************************************
Program Interface PC Dengan Microcontroller
Aplikasi Kontrol Robot Berbasis Web
Author : Hendra Prima Syahputra
Departemen Teknik Mesin Fakultas Teknik Universitas Indonesia
*************************************************************/

/*header - header*/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>

/*
Serial Port Address
COM1 0x3F8
COM2 0x2F8
COM3 0x3E8
cOM4 0x2E8
*/
#define PORT1 0x3F8

void get_serial_data()
{
        int c;
        int counter;
        char buffer[100];

        while(1)
        {
                c = inportb(PORT1 + 5);
                if(c & 1)
                {
                        buffer[counter] = inport(PORT1);
                        printf("%c", buffer[counter]);
                }
                if(kbhit())
                {
                        break;
                }
        }
}

void open_com_port()
{
        */
        outportb(PORT1 + 1 , 0);    /* Turn off interrupts - Port1 */


       /*PORT 1 - Communication Settings*/

        outportb(PORT1 + 3 , 0x80);  /* SET DLAB ON */
        outportb(PORT1 + 0 , 0x01);  /* Set Baud rate
                                        Divisor Latch Low Byte*/
                                     /* Default 0x03 =  38,400 BPS */
                                     /*         0x01 = 115,200 BPS */
```

```
                                        /*          0x02 =  57,600 BPS */
                                        /*          0x06 =  19,200 BPS */
                                        /*          0x0C =   9,600 BPS */
                                        /*          0x18 =   4,800 BPS */
                                        /*          0x30 =   2,400 BPS */
        outportb(PORT1 + 1 , 0x00);     /* Set Baud rate
                                        Divisor Latch High Byte */
        outportb(PORT1 + 3 , 0x03);     /* 8 Bits, No Parity, 1 Stop Bit */
        outportb(PORT1 + 2 , 0xC7);     /* FIFO Control Register */
        outportb(PORT1 + 4 , 0x0B);     /* Turn on DTR, RTS, and OUT2 */
}

void default_position_command()
{
        */
        int value_default = 255;

        clrscr();
        outportb(PORT1, value_default);

        get_serial_data();
}


void move_all_axis_command()
{
        clrscr();
        printf("Please Insert Parameter\n");
        printf("Value Range 0-255\n");

        int duty_cycle_axis_1, arah_axis_1, angle_axis_1;
        int duty_cycle_axis_2, arah_axis_2, angle_axis_2;
        int duty_cycle_axis_3, arah_axis_3, angle_axis_3;
        int duty_cycle_axis_4, arah_axis_4, angle_axis_4;
        int duty_cycle_axis_5, arah_axis_5, angle_axis_5;

        int value_move_all_axis = 254;

        outportb(PORT1, value_move_all_axis);

        get_serial_data();

        //duty_cycle_axis_1
        printf("duty_cycle_axis_1 :  ");
        scanf("%d", &duty_cycle_axis_1);
        outportb(PORT1, duty_cycle_axis_1);

        get_serial_data();

        //arah_axis_1
        printf("arah_axis_1 :  ");
        scanf("%d", &arah_axis_1);
        outportb(PORT1, arah_axis_1);

        get_serial_data();

        //angle_axis_1
        printf("angle_axis_1 :  ");
        scanf("%d", &angle_axis_1);
        outportb(PORT1, angle_axis_1);
```

```
get_serial_data();

//duty_cycle_axis_2
printf("duty_cycle_axis_2 :  ");
scanf("%d", &duty_cycle_axis_2);
outportb(PORT1, duty_cycle_axis_2);

get_serial_data();

//arah_axis_2
printf("arah_axis_2 :  ");
scanf("%d", &arah_axis_2);
outportb(PORT1, arah_axis_2);

get_serial_data();

//angle_axis_2
printf("angle_axis_2 :  ");
scanf("%d", &angle_axis_2);
outportb(PORT1, angle_axis_2);

get_serial_data();

//duty_cycle_axis_3
printf("duty_cycle_axis_3 :  ");
scanf("%d", &duty_cycle_axis_3);
outportb(PORT1, duty_cycle_axis_3);

get_serial_data();

//arah_axis_3
printf("arah_axis_3 :  ");
scanf("%d", &arah_axis_3);
outportb(PORT1, arah_axis_3);

get_serial_data();

//angle_axis_3
printf("angle_axis_3 :  ");
scanf("%d", &angle_axis_3);
outportb(PORT1, angle_axis_3);

get_serial_data();

//duty_cycle_axis_4
printf("duty_cycle_axis_4 :  ");
scanf("%d", &duty_cycle_axis_4);
outportb(PORT1, duty_cycle_axis_4);

get_serial_data();

//arah_axis_4
printf("arah_axis_4 :  ");
scanf("%d", &arah_axis_4);
outportb(PORT1, arah_axis_4);

get_serial_data();

//angle_axis_4
```

```
        printf("angle_axis_4 :  ");
        scanf("%d", &angle_axis_4);
        outportb(PORT1, angle_axis_4);

        get_serial_data();

        //duty_cycle_axis_5
        printf("duty_cycle_axis_5 :  ");
        scanf("%d", &duty_cycle_axis_5);
        outportb(PORT1, duty_cycle_axis_5);

        get_serial_data();

        //arah_axis_5
        printf("arah_axis_5 :  ");
        scanf("%d", &arah_axis_5);
        outportb(PORT1, arah_axis_5);

        get_serial_data();

        //angle_axis_5
        printf("angle_axis_5 :  ");
        scanf("%d", &angle_axis_5);
        outportb(PORT1, angle_axis_5);

        get_serial_data();
}

void manual_mode()
{
        int select_command;

        clrscr();
        printf("Manual Mode Activated\n");
        printf("Select Command\n");
        printf("1. Default Position\n");
        printf("2. Move_All_Axis\n");
        scanf("%d", &select_command);

        switch(select_command)
        {
                case 1 :
                default_position_command();
                break;

                case 2 :
                move_all_axis_command();
                break;

                default:
                printf("Invalid Mode Selection\n");
        }
}

void web_based_loop()
{
        while(1)
        {
                clrscr();
```

```
                FILE *file_pointer;
                int counter = 0;
                int movement_data[15] = {0};
                int value_web_based;

            /*Membuka file*/
            file_pointer = fopen("D:/adtj/www/htdocs/i-RoMan/kdt.txt","rb");

            /*Menghentikan loop dengan kode keyboard*/
            if(kbhit())
                {
/*
Pada saat keluar, data yang   dikirim bernilai 0  untuk memastikan robot
berhenti, nilai ini sama dengan data emergency yang digenerate oleh web.
*/
                        movement_data[0] = 0;
                        movement_data[1] = 0;
                        movement_data[2] = 0;
                        movement_data[3] = 0;
                        movement_data[4] = 0;
                        movement_data[5] = 0;
                        movement_data[6] = 0;
                        movement_data[7] = 0;
                        movement_data[8] = 0;
                        movement_data[9] = 0;
                        movement_data[10] = 0;
                        movement_data[11] = 0;
                        movement_data[12] = 0;
                        movement_data[13] = 0;
                        movement_data[14] = 0;

                        /*Mengirim data ke microcontroller*/
                        delay(100);
                        outportb(PORT1, movement_data[0]);
                        delay(100);
                        outportb(PORT1, movement_data[1]);
                        delay(100);
                        outportb(PORT1, movement_data[2]);
                        delay(100);
                        outportb(PORT1, movement_data[3]);
                        delay(100);
                        outportb(PORT1, movement_data[4]);
                        delay(100);
                        outportb(PORT1, movement_data[5]);
                        delay(100);
                        outportb(PORT1, movement_data[6]);
                        delay(100);
                        outportb(PORT1, movement_data[7]);
                        delay(100);
                        outportb(PORT1, movement_data[8]);
                        delay(100);
                        outportb(PORT1, movement_data[9]);
                        delay(100);
                        outportb(PORT1, movement_data[10]);
                        delay(100);
                        outportb(PORT1, movement_data[11]);
                        delay(100);
                        outportb(PORT1, movement_data[12]);
                        delay(100);
                        outportb(PORT1, movement_data[13]);
```

```
                        delay(100);
                        outportb(PORT1, movement_data[14]);

                        /*Menutup file dan menghapus file*/
                        fclose(file_pointer);
                        remove("D:/adtj/www/htdocs/i-RoMan/kdt.txt");
                        break;
                }

                /*File ditemukan*/
                else if(file_pointer != NULL)
                {
                        printf("Movement File Found!\n");
                        printf("Press ENTER to exit\n");

                    /*Membaca isi file untuk dimasukkan
                    sebagai data pergerakan*/
                    while(!feof(file_pointer))
{

fscanf(file_pointer,"%d",&movement_data[counter]);
                    counter++;
                    }

                        /*Mengirim data ke microcontroller*/
                        delay(100);
                        outportb(PORT1, movement_data[0]);
                        delay(100);
                        outportb(PORT1, movement_data[1]);
                        delay(100);
                        outportb(PORT1, movement_data[2]);
                        delay(100);
                        outportb(PORT1, movement_data[3]);
                        delay(100);
                        outportb(PORT1, movement_data[4]);
                        delay(100);
                        outportb(PORT1, movement_data[5]);
                        delay(100);
                        outportb(PORT1, movement_data[6]);
                        delay(100);
                        outportb(PORT1, movement_data[7]);
                        delay(100);
                        outportb(PORT1, movement_data[8]);
                        delay(100);
                        outportb(PORT1, movement_data[9]);
                        delay(100);
                        outportb(PORT1, movement_data[10]);
                        delay(100);
                        outportb(PORT1, movement_data[11]);
                        delay(100);
                        outportb(PORT1, movement_data[12]);
                        delay(100);
                        outportb(PORT1, movement_data[13]);
                        delay(100);
                        outportb(PORT1, movement_data[14]);

                        printf("Movement File Executed\n");
```

```
/*Menampilkan data yang dikirim*/

printf("%d %d %d\n", movement_data[0], movement_data[1],
movement_data[2]);
printf("%d %d %d\n", movement_data[3],    movement_data[4],
movement_data[5]);
printf("%d %d %d\n", movement_data[6], movement_data[7],
movement_data[8]);
printf("%d %d %d\n", movement_data[9], movement_data[10],
movement_data[11]);
printf("%d %d %d\n", movement_data[12], movement_data[13],
movement_data[14]);

/*Mengembalikan kondisi microcontroller pada kondisi web based*/

                        delay(3000);
                        printf("Sending Web Command\n");

                        delay(3000);
                        value_web_based = 253;

                        delay(4000);
                        outportb(PORT1, value_web_based);

/*Menutup dan membuang file*/

                        fclose(file_pointer);
                        remove("D:/adtj/www/htdocs/i-RoMan/kdt.txt");

                        printf("File Removed\n");
                }

                /*File tidak ditemukan*/
                else if(file_pointer == NULL)
                {
                        printf("Movement File NOT Found!\n");
                        printf("Press ENTER to exit\n");

                        fclose(file_pointer);
                }

                else
                {
                        fclose(file_pointer);
                }
                delay(1000);
        }
}

void web_mode()
{
        int value_web_based;
        int web_based_command;
        clrscr();
        printf("1. Activate Web Based Loop\n");
        printf("2. Exit\n");
        scanf("%d", &web_based_command);

        switch(web_based_command)
```

```
        {
                case 1 :
                value_web_based = 253;
                outportb(PORT1, value_web_based);
                web_based_loop();
                break;

                case 2 :
                value_web_based = 252;
                outportb(PORT1, value_web_based);
                break;

                default:
                printf("Invalid Mode Selection\n");
        }
}
void exit_program()
{
        /*Fungsi untuk keluar dari program*/
        exit(0);
}

void main()
{
        while(1)
        {
                int select_mode;
                clrscr();

                printf("Movemaster RV-M1 Interface Driver\n");
                printf("Select Mode\n");
                printf("1. Manual Mode\n");
                printf("2. Web Based\n");
                printf("3. Exit\n");
                scanf("%d", &select_mode);

                switch(select_mode)
                {
                        case 1 :
                        open_com_port();
                        manual_mode();
                        break;

                        case 2 :
                        open_com_port();
                        web_mode();
                        break;

                        case 3 :
                        exit_program();
                        break;

                        default:
                        printf("Invalid Mode Selection\n");
                }
                getch();
        }
}
```

UNIVERSITAS INDONESIA

PENGEMBANGAN SISTEM KONTROL PERGERAKAN ROBOT ARTIKULASI 5 DERAJAT KEBEBASAN BERBASIS *WEB*

SKRIPSI

HENDRA PRIMA SYAHPUTRA
0405020332

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK MESIN
DEPOK
DESEMBER 2009

UNIVERSITAS INDONESIA

PENGEMBANGAN SISTEM KONTROL PERGERAKAN ROBOT ARTIKULASI 5 DERAJAT KEBEBASAN BERBASIS *WEB*

SKRIPSI

Diajukan sebagai salah satu syarat
untuk memperoleh gelar sarjana teknik

HENDRA PRIMA SYAHPUTRA
0405020332

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK MESIN
DEPOK
DESEMBER 2009

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama	**: Hendra Prima Syahputra**
NPM	**: 0405020332**
Tanda Tangan	**: ……**
Tanggal	**:**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Hendra Prima Syahputra
NPM	:	0405020332
Program Studi	:	Teknik Mesin
Judul Skripsi	:	PENGEMBANGAN SISTEM KONTROL PERGERAKAN ROBOT ARTIKULASI 5 DERAJAT KEBEBASAN BERBASIS *WEB*

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

DEWAN PENGUJI

Pembimbing	: Dr. Ir. Gandjar Kiswanto, M.Eng	( )
Penguji	: Ir. Hendri DS Budiono, M.Eng	( )
Penguji	: Ir. Henky S Nugroho, M.T	( )
Penguji	: Dr. Ario S Baskoro, S.T, M.T, M.Eng	( )

Ditetapkan di :

Tanggal :

ABSTRAK

Nama : Hendra Prima Syahputra

Program Studi : Teknik Mesin

Judul : Pengembangan Sistem Kontrol Robot Artikulasi 5 Derajat Kebebasan Berbasis *Web*

Penelitian ini merancang sebuah sistem yang mampu mengontrol sebuah robot artikulasi dengan lima derajat kebebasan dari jarak jauh melalui media internet yang berbasiskan aplikasi *web*. Dalam penelitian ini digunakan sebuah komputer yang bertindak sebagai *server* yang dilengkapi dengan dua buah *web camera* untuk memantau kondisi dan pergerakan robot dan juga sebuah mikrokontroler pengontrol robot sebagai pemroses dan pengontrol masukan untuk menggerakkan robot. Melalui sebuah *web browser* pada komputer yang bertindak sebagai client, sistem pada komputer *server* diakses oleh pengguna dan menampilkan sebuah antarmuka yang dirancang sebagai panel kontrol robot. Melalui antarmuka ini pengguna dapat memberi masukan berupa perintah untuk menggerakkan robot yang dapat diberikan dalam dua pilihan mode basis kontrol, yaitu *cursor-based/inverse kinematics* dan *manual/forward kinematics*. Sistem mampu merespon perintah yang diberikan kemudian memroses dan mengeksekusinya dalam bentuk pergerakan robot sesuai dengan mode dan perintah dari masukan yang diberikan.

Kata kunci:

Kontrol robot, aplikasi *web*

ABSTRACT

Name : Hendra Prima Syahputra

Study Program : Mechanical Engineering

Title : Development of Web Based 5-DOF Articulated Robot Motion Control System

This research is aimed to design and develop a system capable of remotely controlling a five-degree-of-freedom articulated robot through internet platform on a web-based application. The research was built with single computer act as a server coupled with a pair of web camera to monitor the status and movement of the robot and also coupled with a robot-controller micro controler as a processor and controller of inputs to move the robot. Through the web browser on user's computer acting as *client*, the system is accessed by the user and displays an interface designed to be a robot's control panel. Through this interface, the user can input command to move the robot which can be given in two different control modes, cursor-based/inverse kinematics and manual/forward kinematics. System responds the command then processes and executes it in form of robot movement based on control mode and command of the given input.

Keywords:

Robot control, web application

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Mesin pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, penyusunan skripsi ini sangatlah sulit bagi saya. Oleh karena itu, saya mengucapkan terima kasih kepada:

1) Orang tua saya yang telah memberikan dukungan moril dan materiil;
2) Dr. Ir. Gandjar Kiswanto, M.Eng selaku dosen pembimbing yang telah meluangkan waktunya untuk menyemangati saya dan menginspirasi saya dalam pengerjaan skripsi ini;
3) Dr. Ir.Harinaldi, M.Eng selaku kepala Departemen Teknik Mesin;
4) Dr. Ir.Abdul Muis, M.Eng yang telah memberikan semangat dan pengetahuan elektronika kepada saya;
5) Sahabat saya Auralius Manurung, S.T, Gunawan, S.T, M.Syafiuddin, S.T, Erwin Nugraha Bayuaji, dan Ahmad Zakiyudin yang telah memberikan pengetahuan software dan elektronika kepada saya;
6) Seluruh teman dalam Tim Robot Universitas Indonesia (TRUI) atas segala pengalaman dan pengetahuan yang saya dapat di TRUI;
7) Tim WRMA (*Web* Based Manipulation), Anom Tejo, Adithya Bayu dan Teguh Santoso atas kerja samanya dalam penyelesaian skripsi ini;
8) Seluruh teman seperjuangan saya mesin 2005.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Depok, Desember 2009

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR ISTILAH

ISO : International Organization for Standardization.
DOF : Degree of Freedom.
DSP : Digital Signal Processing.
DC : Direct Current.
NC : Normally Close.
NO : Normally Open.
IC : Integrated Circuit.
CPU : Central Processing Unit
I/O : Input Output
RISC : Reduced Instruction Set Computing
PWM : Pulse Width Modulation
LED : Light Emitting Diode
PCB : Printed Circuit Board
EMF : Electromagnetic Field
LCD : Liquid Crystal Display
USB : Universal Serial Bus
UART : Universal Asynchronous Receiver-Transmitter
CPR : Count per Revolution

BAB 1
PENDAHULUAN

1.1 Latar Belakang

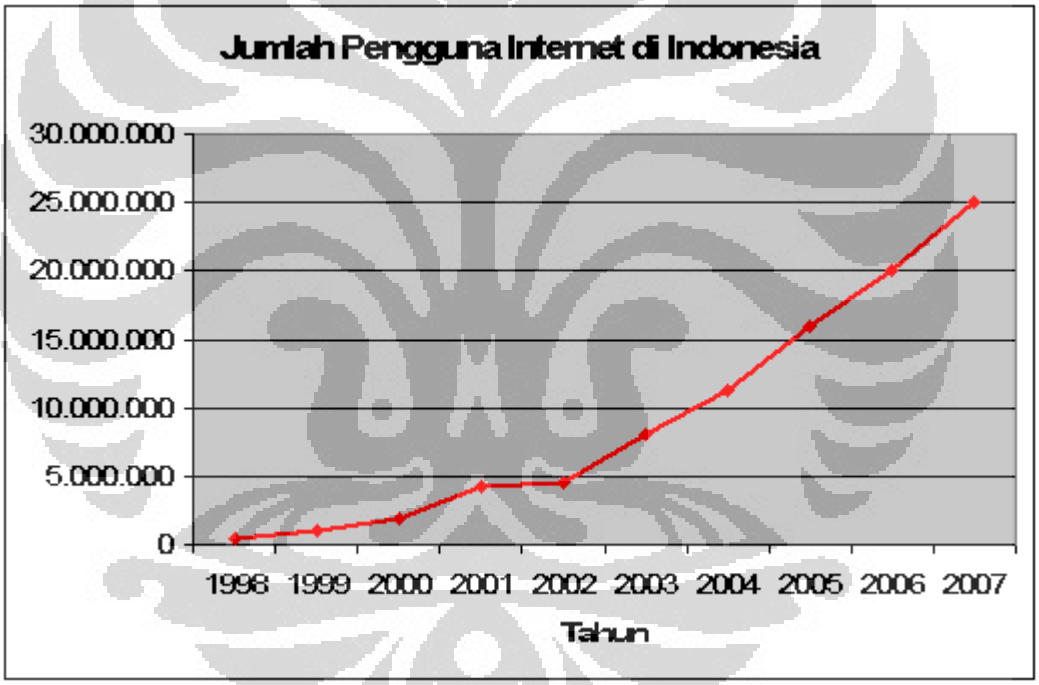
Penggunaan robot didalam dunia perindustrian semakin luas, hal ini terjadi seiring dengan meningkatnya kebutuhan masyarakat akan barang – barang yang berkualitas. Karena kelebihan robot yang mampu bekerja lebih banyak dan lebih konsisten dibandingkan manusia maka mulai bermunculan robot – robot dengan bentuk dan kegunaan yang bervariasi. Perkembangan ini didukung dengan kemajuan teknologi didalam bidang mekanikal, elektronika dan teknologi informasi. Teknologi tersebut kemudian diaplikasi pada berbagai macam robot untuk meningkatkan kecerdasan, daya tahan, dan akurasi dari robot – robot tersebut.

Salah satu dasar pengembangan robot adalah memberikan pertolongan kepada manusia untuk melakukan pekerjaan yang sulit dan membahayakan bagi manusia. Beberapa industri yang menuntut tingkat kepresisian yang tinggi tidak menggunakan lagi manusia sebagai pekerjanya namun robot sebagai pekerja didalam pabrik. Hal ini menuntut pengembangan robot yang mampu memenuhi permintaan manusia. Salah satu kebutuhan tersebut adalah pengendalian robot dari jarak jauh. Pengendalian robot dari jarak jauh sangat dibutuhkan untuk mempermudah manusia memonitor pekerjaan robot dan mengkoreksi pekerjaan robot pada pabrik – pabrik yang tidak lagi menggunakan manusia sebagai pekerjanya.

Salah satu metode pengendalian robot dari jarak jauh yaitu dengan menggunakan internet. Pengendalian menggunakan internet didasari karena perkembangan teknologi informasi sekarang yang banyak sekali menggunakan internet. Kebutuhan pengendalian melalui internet semakin diperlukan karena adanya kebutuhan untuk mengendalikan dan memonitor sistem yang dikendalikan dari tempat yang berbeda dari sistem yang dikendalikan. Keterbatasan ruang dan waktu untuk bekerja dan berpindah tempat juga menjadi alasan yang kuat untuk mengembangkan

pengendalian berbasis internet. Dunia perdagangan sampai dunia pendidikan pun sudah banyak dikembangkan dengan menerapkan teknologi internet ini. Hal ini disebabkan karena kemudahan dan kecepatan akses internet yang semakin meningkat dari tahun ke tahun dan perkembangan jaringan internet yang semakin menyebar di seluruh dunia. Sehingga sudah saatnya pengendalian robot pun dapat dilakukan melalui internet.

Penggunaan internet sebagai sarana pertukaran informasi dan data di Indonesia semakin meningkat. Sesuai dengan data statistik dari APJII (Asosiasi Penyelenggara Jasa Internet Indonesia) pengguna internet di Indonesia meningkat dari tahun ke tahun seperti terlihat pada gambar berikut :



Gambar 1.1 Jumlah Pengguna Internet di Indonesia Tahun 1998 – 2007 [1]

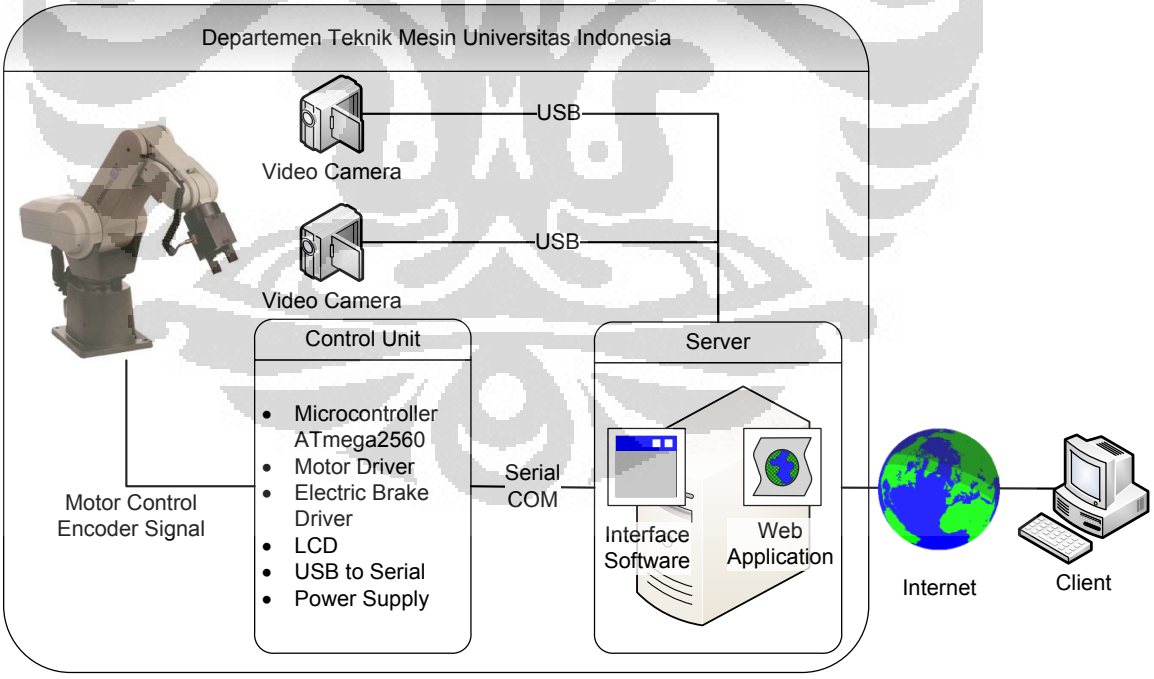
Peningkatan penggunaan internet di Indonesia inilah yang menjadi alasan penelitian kendali berbasis internet perlu dikembangkan dan diaplikasikan dalam dunia perindustrian di Indonesia.

1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk mengembangan prototipe sistem pengendalian robot industri dari jarak jauh berbasis internet dalam rangka pengembangan *intellegent manufacturing system* di Laboratorium Teknologi Manufaktur dan Otomasi Departemen Teknik Mesin Universitas Indonesia.

1.3 Perumusan Masalah

Sebuah sistem dikembangkan untuk mengendalikan lengan robot dengan 5 (lima) derajat kebebasan dari jarak jauh melalui internet. Untuk dapat memenuhi kriteria pengendalian berbasis jaringan internet, sistem harus memiliki fasilitas yang dapat mewakili panel kontrol (control unit) yang terdapat pada mesin sebenarnya dan juga harus dilengkapi dengan fasilitas pemantauan proses pergerakan robot secara visual agar pengguna dapat dengan mudah mempergunakan sistem tersebut. Sistem pengendalian berbasis jaringan internet ini mempunyai *architecture system* seperti berikut :



Gambar 1.2 Rancangan Sistem Keseluruhan

1.4 Pembatasan Masalah

Skripsi ini membahas mengenai komunikasi data antara jaringan internet, komputer, dan *microcontroller* untuk menggerakkan robot dari jarak jauh. Data yang dikirim adalah data pergerakan robot artikulasi dengan 5 derajat kebebasan yaitu robot Mitsubishi Movemaster RV-M1. *Microcontroller* yang digunakan adalah *microcontroller* AVR dari ATMEL *Corporation*.

1.5 Metodologi Penelitian

Metodologi penelitian ini adalah sebagai berikut:

1. Studi literatur terhadap artikel pengendalian berbasis robot
2. Studi mekanisme robot Mitsubishi Movemaster RV-M1.
3. Pemodelan sistem arsitektur pengendalian robot berbasis *web*.
4. Pengembangan sistem pengendalian robot.
5. Eksperimen rancangan sistem arsitektur pengendalian robot.
6. Pengujian & analisa sistem kendali robot.

1.6 Sistematika Penulisan

BAB 1. PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang masalah, perumusan masalah, pembatasan masalah, tujuan penelitian, metodologi penelitian dan sistematika penulisan.

BAB 2. MEKANIKA DAN KONTROL ROBOT INDUSTRI

Bab ini menjelaskan tentang pengenalan berbagai tipe robot industri dan teori – teori yang dipakai dalam pengendalian robot industri.

BAB 3. TINJAUAN SISTEM MEKANIKAL ROBOT

Bab ini menjelaskan tentang tijauan sistem mekanikal robot Movemaster RV-M1 dan spesifikasinya sebagai dasar perancangan sistem elekronik dan perangkat lunak.

BAB 4. PENGEMBANGAN PERANGKAT KERAS SISTEM PENGENDALI

Bab ini menjelaskan tentang perancangan dan pembuatan dari sistem elektronika yang dipakai dalam pengendalian robot berbasis *web* ini.

BAB 5. PENGEMBANGAN SISTEM PERANGKAT LUNAK SISTEM PENGENDALI

Bab ini menjelaskan tentang perancangan dan pembuatan dari sistem perangkat lunak dan algoritma yang dipakai dalam pengendalian robot berbasis *web* ini.

BAB 6. PENGEMBANGAN SISTEM KOMUNIKASI DENGAN *WEB* SERVER

Bab ini menjelaskan tentang pengembangan komunikasi dengan *web* server yang dikembangkan untuk sistem kendali robot berbasis *web* ini.

BAB 7. PENGUJIAN & ANALISA SISTEM

Bab ini menjelaskan tentang analisa sistem secara umum untuk mengetahui kelebihan dan kekurangan dari sistem ini.

BAB 8. KESIMPULAN & SARAN PENELITIAN LEBIH LANJUT

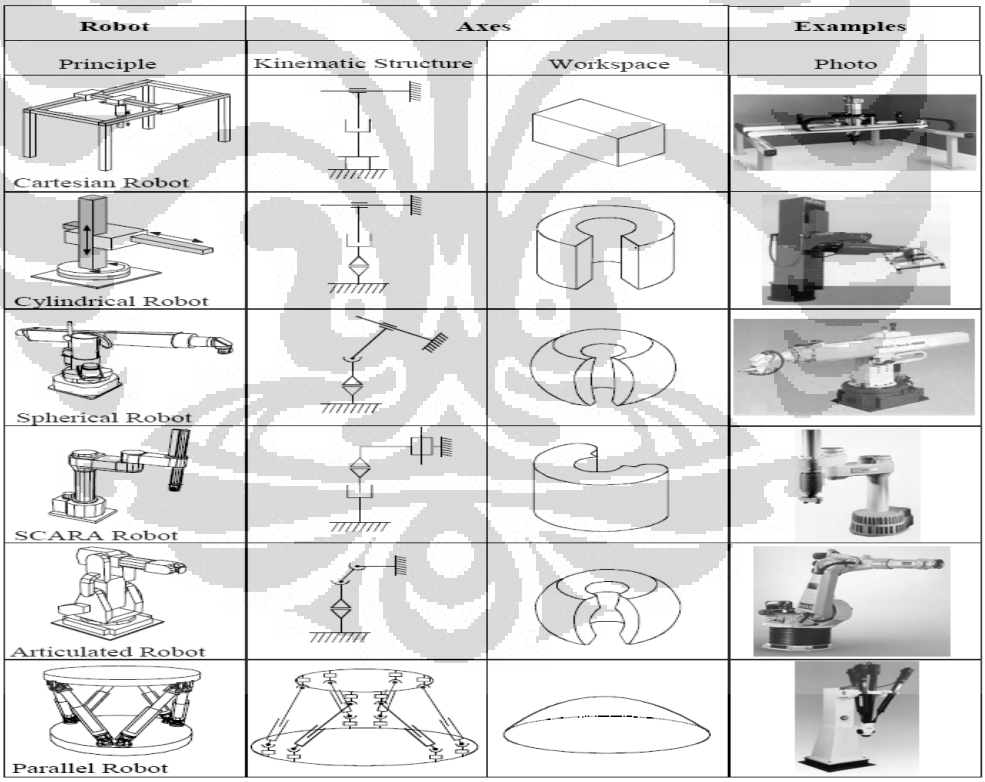
Bab ini menjelaskan kesimpulan penelitian dan saran untuk penelitian selanjutnya.

BAB 2
MEKANIKA DAN KONTROL ROBOT INDUSTRI

Bagian ini menjelaskan hal – hal yang berhubungan dengan mekanika dan kontrol robot industri pada umumnya.

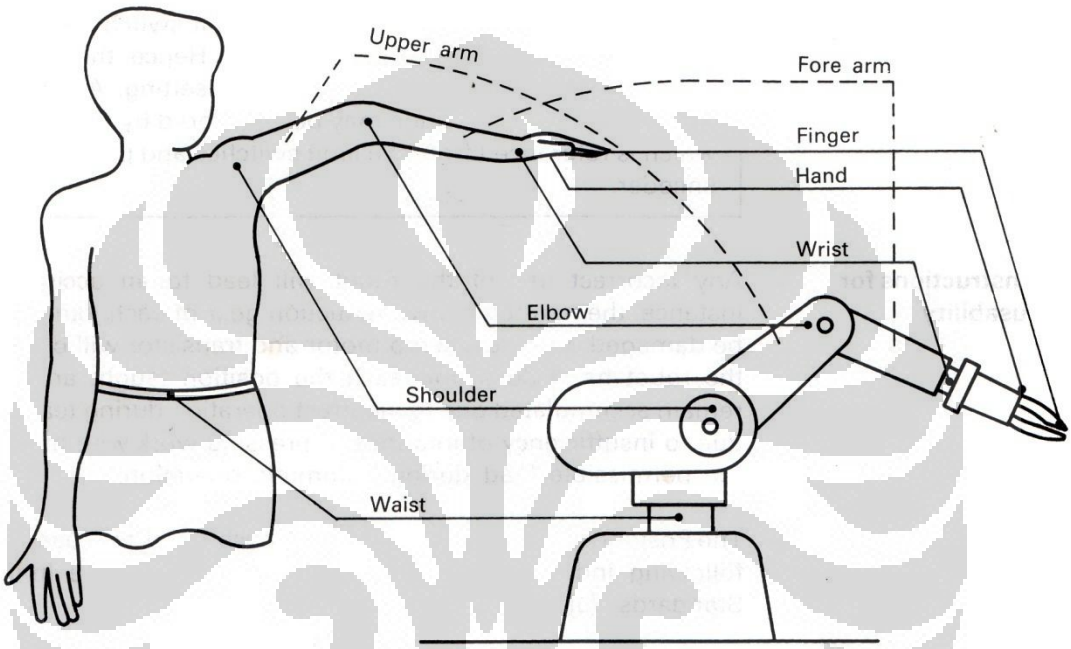
2.1 Konstruksi Robot

Definisi robot industri menurut standar ISO 8373 (*Manipulating Industrial Robot*) adalah : *"An automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation application".*



Gambar 2.1 Klasifikasi Konstruksi Mekanik Robot Industri pada ISO 8373 [2]

Sesuai dengan gambar pada ISO 8373, maka robot yang dipakai dalam penelitian ini merupakan robot dengan tipe artikulasi dengan 5 derajat kebebasan, namun dalam ukuran yang lebih kecil sehingga disebut *industrial micro-robot*. Pengembangan robot dengan 5 derajat kebebasan didasari oleh pergerakan yang dapat dihasilkan oleh manusia.



Gambar 2.2 Dasar Perancangan Robot Artikulasi [3]

Dengan kemampuan yang menyerupai lengan manusia ini maka robot – robot tersebut dapat digunakan di pabrik sebagai pengganti pekerja manusia.

Robot artikulasi dapat dilengkapi dengan berbagai peralatan pada bagian *end-effector*. Mulai dari *Spot Welding, Laser, Gripper, atau Machining Tool*, hal ini yang menyebabkan penggunaanya sangat luas pada dunia industri. Beberapa robot bahkan sudah mempunyai kecerdasan buatan yang diterapkan didalam *controller* robot tersebut. Dengan adanya kecerdasan buatan, robot mampu menganalisa dan merencanakan pergerakan dirinya tanpa adanya bantuan dari manusia.

2.2 Derajat Kebebasan

Derajat kebebasan merupakan paket pergerakan yang mampu dilakukan oleh suatu sistem mekanik. Derajat kebebasan atau sering disebut dengan *DOF (Degree of Freedom)* dihitung berdasarkan kemampuan perubahan orientasi sistem mekanik tersebut. Dalam perancangan robot DOF ditentukan berdasarkan kegunaan robot tersebut. Sebuah robot dapat dikatakan *redundant* apabila mempunyai DOF yang melebihi DOF minimal yang diperlukan untuk melakukan tugasnya. Untuk robot yang *redundant* perlu dirancang suatu algoritma pergerakan yang bersifat *task priority*.

2.3 Pengendalian Menggunakan Metode Forward Kinematics

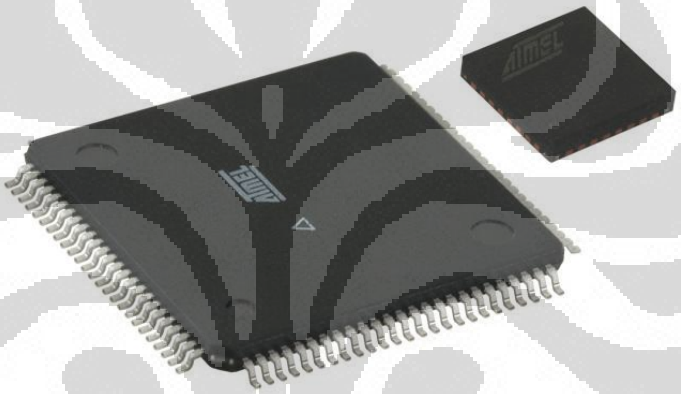
Forward kinematics merupakan suatu perhitungan pergerakan dan orientasi robot berdasarkan parameter pergerakan dari masing – masing *axis*-nya. Aplikasi *forward kinematics* sering dilakukan dipabrik dengan cara melatih robot untuk melaksanakan tugasnya. Parameter pergerakan robot ditentukan oleh *operator* yang menugaskan robot melakukan tugas tertentu. Pada penelitian ini pengendalian robot menggunakan metode forward kinematics dilakukan oleh *client* dengan memasukkan parameter pergerakan robot pada *web*.

2.4 Pengendalian Menggunakan Metode Inverse Kinematics

Inverse kinematics merupakan suatu perhitungan pergerakan dan orientasi robot berdasarkan posisi yang ingin dicapai oleh robot. Posisi tersebut yang akan menentukan parameter pergerakan dari masing – masing *axis*-nya. Untuk menerapkan *inverse kinematics* diperlukan adanya algoritma khusus & model matematika untuk menentukan parameter pergerakan yang diinginkan. Aplikasi *inverse kinematics* sering digunakan pada robot yang memerlukan kecerdasan dalam melaksanakan tugasnya. Didalam penelitian ini pergerakan *inverse kinematic* dilakukan dengan metode *cursor based,* sehingga robot harus harus mampu melakukan pergerakan hanya berdasarkan penunjukan koordinat yang dilakukan oleh *client* melalui layar GUI (*Graphic User Interface*) yang terlihat pada layar monitor *client*.

2.5 Embedded System

Embedded system merupakan suatu sistem komputer yang didedikasikan untuk menjalankan satu atau berbagai macam tugas dan biasanya melakukan proses data secara *real-time.* Sistem ini sering kali memiliki pengendali seperti *microcontroller* atau *digital signal processor* (DSP). Penggunaan sistem ini pada kendali robot berbasis *web* sangat bermanfaat karena kemudahan pemakaian dan harganya yang cukup terjangkau. Instruksi perintah diprogram pada suatu memori yang kapasitasnya tidak terlalu besar, program dapat dimasukkan dengan menggunakan komputer dan programmer yang disediakan oleh produsennya.



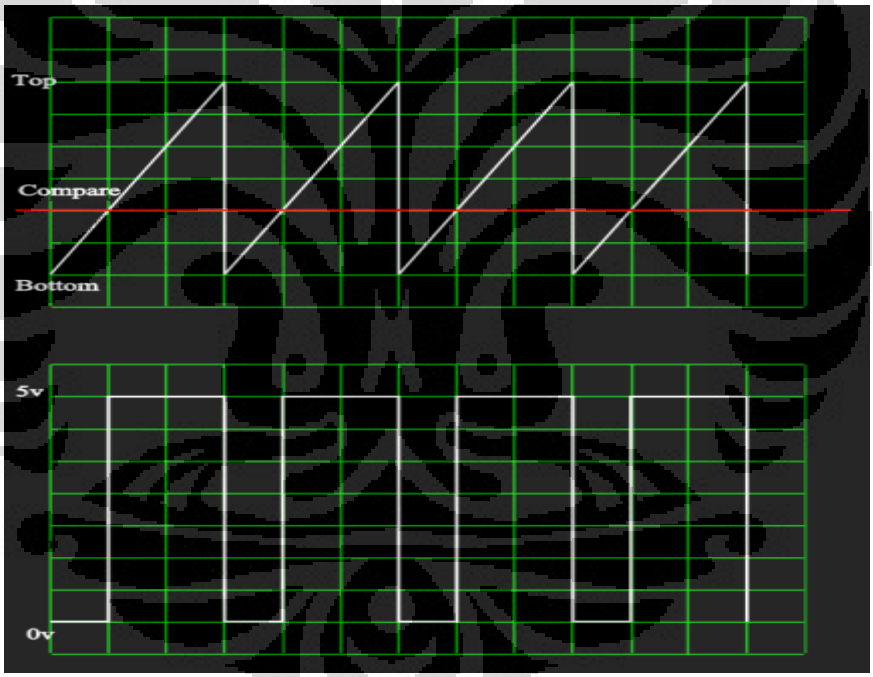
Gambar 2.3 Paket Microcontroller [4]

2.6 Sensor pada Robot Industri

Peran *sensor* dalam pengendalian robot industri sangat besar karena *sensor* merupakan sebuah perangkat yang dapat mendeteksi perubahan pada robot baik secara fisik, kimia ataupun biologi. Perubahan tersebut merupakan *input* bagi *sensor* yang kemudian diubah menjadi sinyal *output*. Sinyal yang didapat dari sensor ini kemudian dapat diolah pada *controller* robot sebagai data untuk melakukan pengendalian. Robot industri pada umumnya menggunakan *sensor* elektromekanikal seperti *encoder,* dan *limit switch.* Layaknya manusia *sensor* merupakan pancaindra bagi robot, sehingga keberadaan sensor sangat penting.

2.7 Actuator Robot Industri

Aktuator merupakan perangkat untuk menggerakkan sebuah sistem mekanik. Pada robot industri terdapat berbagai macam actuator seperti *hydrolic actuator*, *pneumatic actuator*, dan *electric actuator*. Robot Movemaster RV-M1 menggunakan *actuator* berupa motor DC (Direct Current) yang merupakan *electric actuator*. Pengendalian motor DC yang paling umum digunakan adalah menggunakan metode PWM (*Pulse Width Modulation*). Metode PWM merupakan pengendalian sinyal secara digital dengan mengatur lebar pulsa yang dikeluarkan. Lebar pulsa yang dikeluarkan ini dapat mengatur kecepatan motor DC apabila dikombinasikan dengan *driver* motor. Pada *microcontroller* sudah terdapat fitur khusus untuk menghasilkan sinyal PWM yang mempunyai output seperti gambar berikut :



Gambar 2.4 Sinyal PWM [5]

2.8 Bahasa Pemrogramam Sistem Kontrol

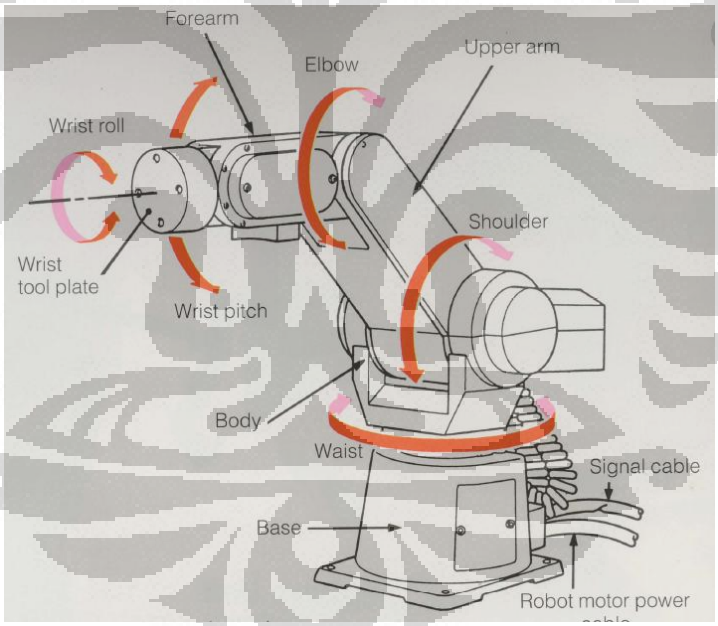
Dalam pengembangan perangkat lunak sebuah sistem kontrol digunakan bahasa pemrograman yang berfungsi untuk mengatur informasi antara *input*, proses dan *output* dari sistem yang dikendalikan. Salah satu bahasa pemrograman yang banyak dipakai adalah bahasa C. Bahasa C banyak digunakan untuk pemrograman sistem seperti aplikasi *operating system* dan *embedded system* karena kemampuannya untuk mengakses berbagai *address* perangkat keras serta efisiensi program yang meringankan kerja sistem. Bahasa C dipakai pada pengembangan robot berbasis *web* ini terutama pada pemrograman *microcontroller* dan *software* komunikasi dari komputer ke *microcontroller* karena kelebihan – kelebihan tersebut.

BAB 3
TINJAUAN SISTEM MEKANIKAL ROBOT

Tinjauan sistem mekanikal robot yang akan digunakan ini bertujuan untuk menentukan pemilihan perangkat elektronik yang tepat.

3.1 Desain Mekanikal Robot Movemaster RV-M1

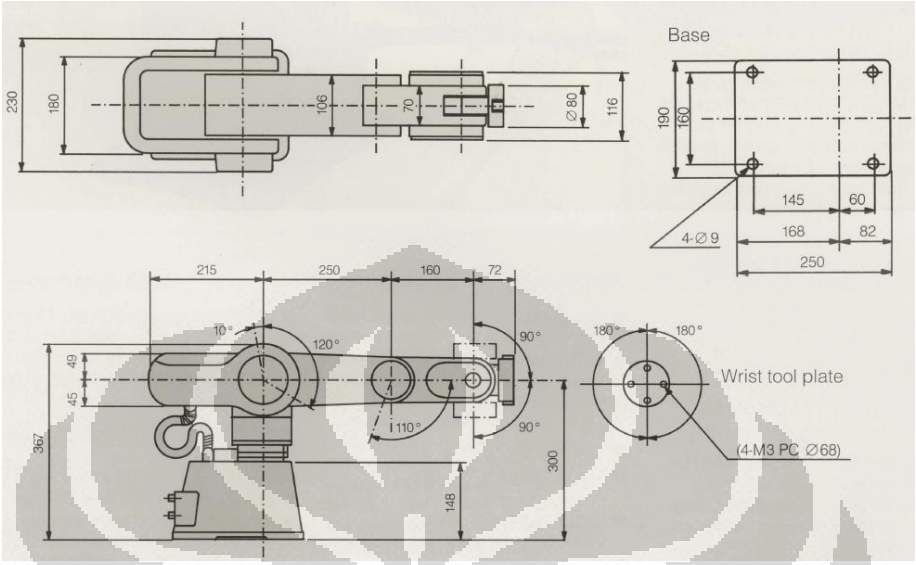
Robot yang digunakan dalam pengembangan sistem kontrol robot berbasis *web* ini adalah robot artikulasi dengan 5 derajat kebebasan yaitu Mitsubishi Movemaster RV-M1. Robot ini mempunyai desain mekanikal seperti yang tertera pada gambar 3.1.



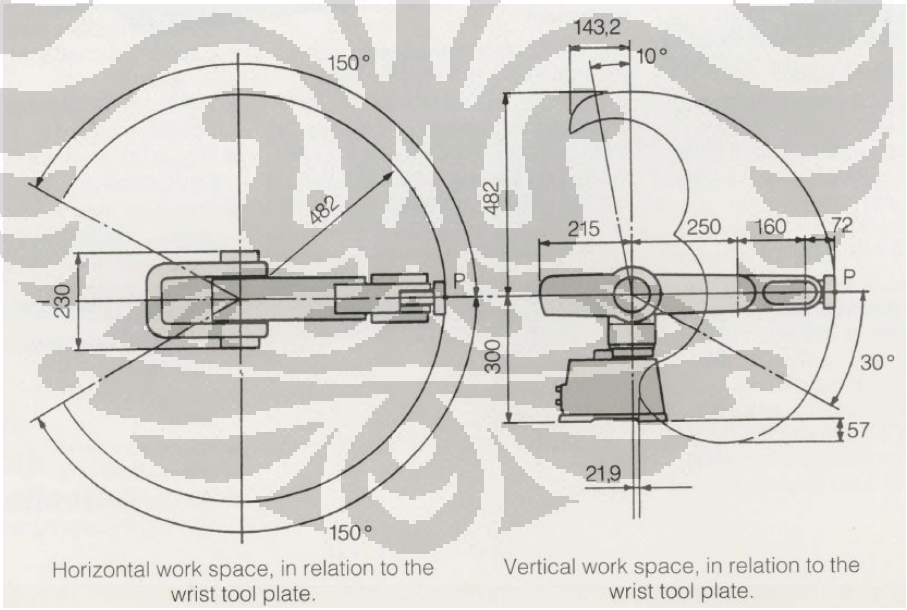
Gambar 3.1 Desain Mekanikal Mitsubishi Movemaster RV-M1 [3]

Desain mekanikal robot seperti ini merupakan jenis robot *manipulator* dengan tipe Artikulasi. Setiap robot manipulator mempunyai keterbatasan dalam melakukan pergerakan bergantung pada desain mekanikalnya sendiri. Keterbatasan pergerakan ini dapat didata dengan membuat diagram pergerakan robot atau biasa disebut dengan

working space diagram. Dimensi dan *working space* robot Mitsubishi Movemaster RV-M1 ini dapat dilihat pada gambar 3.2 dan 3.3.



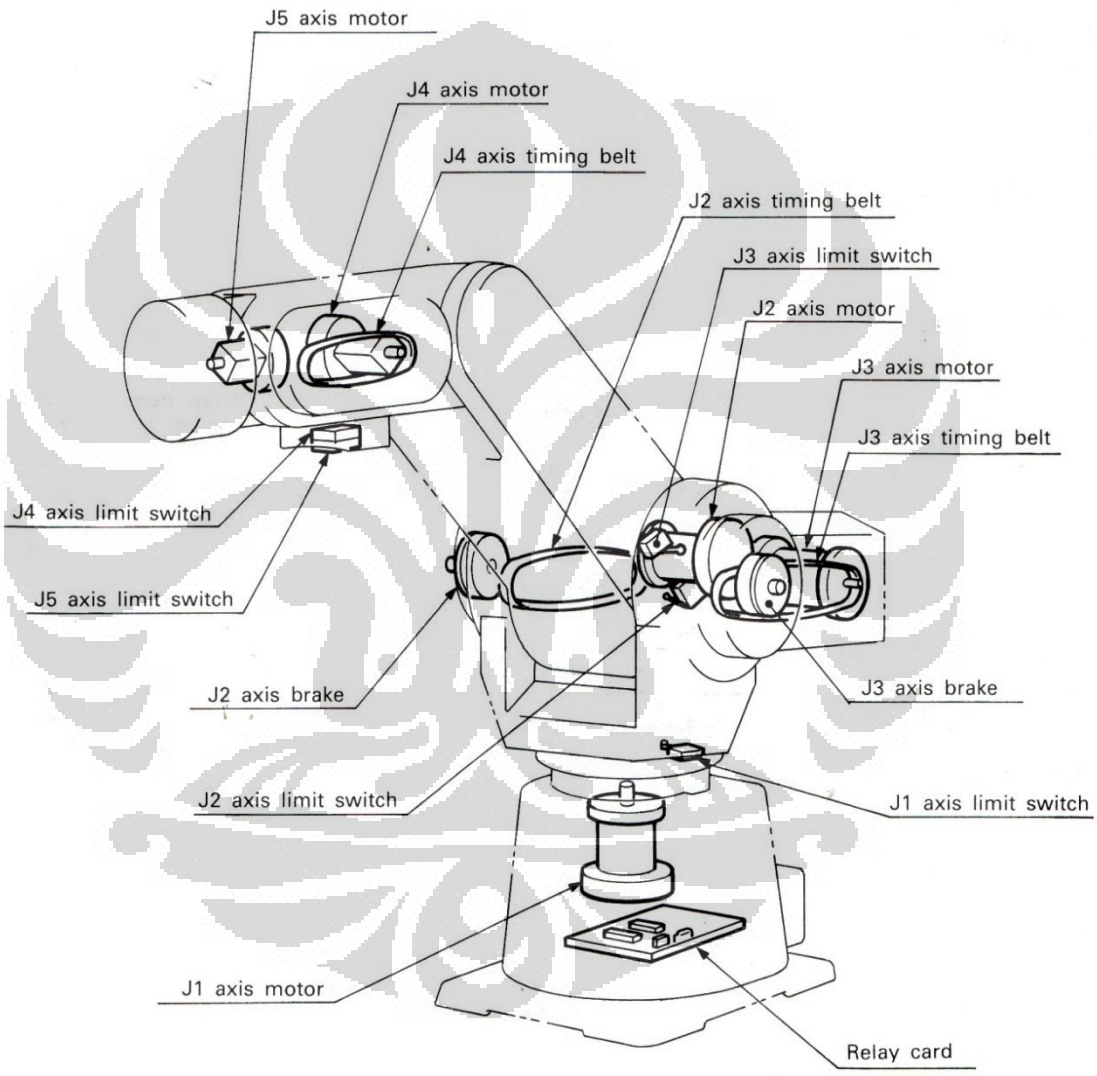
Gambar 3.2 Dimensi Mitsubishi Movemaster RV-M1 [3]



Gambar 3.3 Working Space Mitsubishi Movemaster RV-M1 [3]

3.2 Sistem Penggerak Robot

Robot movemaster RV-M1 memiliki sistem penggerak yang cukup sederhana pada masing – masing *axis*-nya. Sistem penggerak ini cukup akurat untuk menggerakkan robot ke posisi yang diinginkan. Posisi alat perangkat sistem penggerak robot dapat dilihat pada gambar berikut :



Gambar 3.4 Sistem Penggerak Mitsubishi Movemaster RV-M1 [3]

3.2.1 Motor DC (*Direct Current*)

Robot Movemaster RV-M1 mempunyai penggerak bertenaga motor DC untuk menggerakkan masing – masing axisnya, berikutlah salah satu gambar dari motor DC tersebut :

Gambar 3.5 Motor DC pada Mitsubishi Movemaster RV-M1

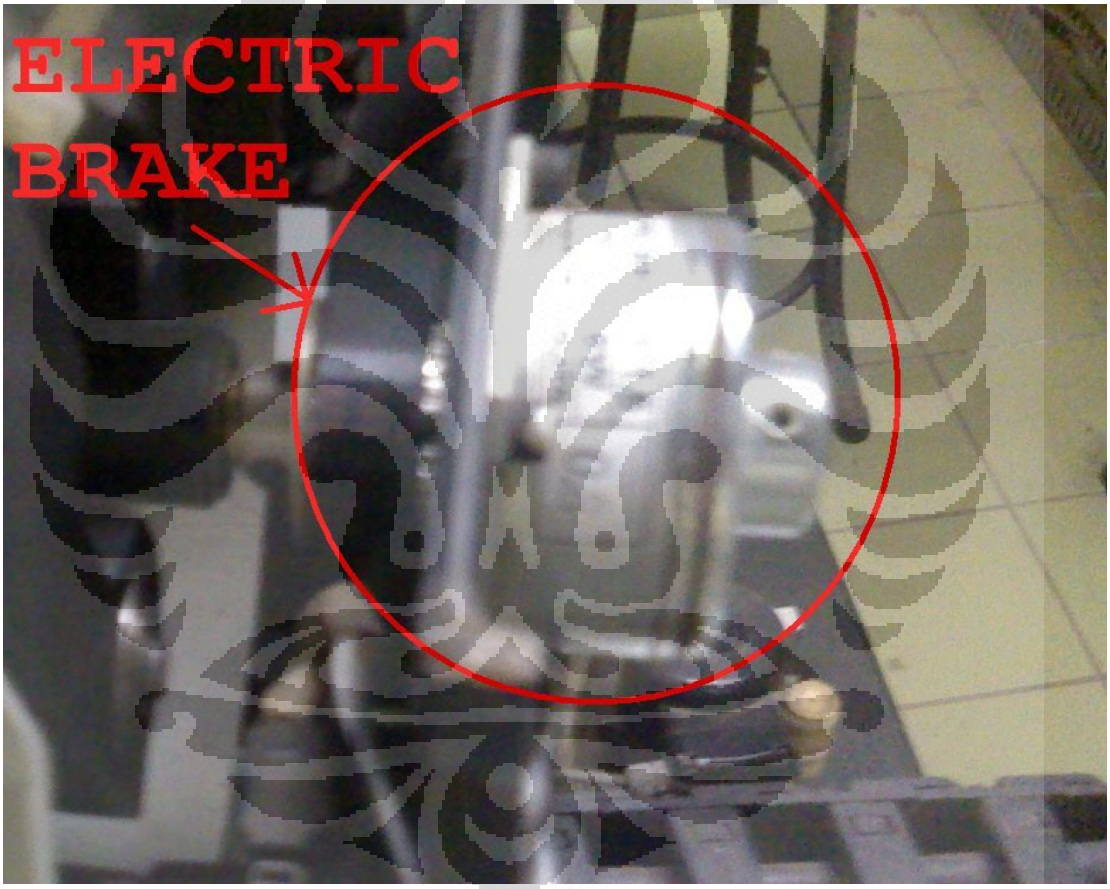
DC motor yang digunakan adalah Sanyo Denki DC Servo Motor dilengkapi dengan *encoder*. Spesifikasi yang tertera pada *name plate* motor DC tersebut adalah :

1. 2.3 Ampere.
2. 24 Volt.
3. 28 Watt.

Motor DC yang digunakan pada setiap axis mempunyai spesifikasi yang serupa namun pada setiap axis motor tersebut di hubungkan dengan gearbox dengan konfigurasi yang berbeda bergantung pada *load* yang diterimanya.

3.2.2 Rem Elektronik (Electric Brake)

Axis 2 dan *Axis* 3 dilengkapi dengan *electric brake* karena kedua *axis* tersebut merupakan *axis* yang memiliki *load* terbesar. *Electric brake* merupakan perangkat elektromekanikal yang bekerja dengan memanfaatkan prinsip elektromagnet untuk menghubungkan atau memisahkan poros yang berputar dengan *chassis* robot yang diam. *Electric brake* yang digunakan pada robot Movemaster RV-M1 ini merupakan tipe NC (Normally Closed) dan membutuhkan tegangan 12V untuk merubahnya menjadi *open state*. Berikut adalah gambar dari *electric brake* tersebut :

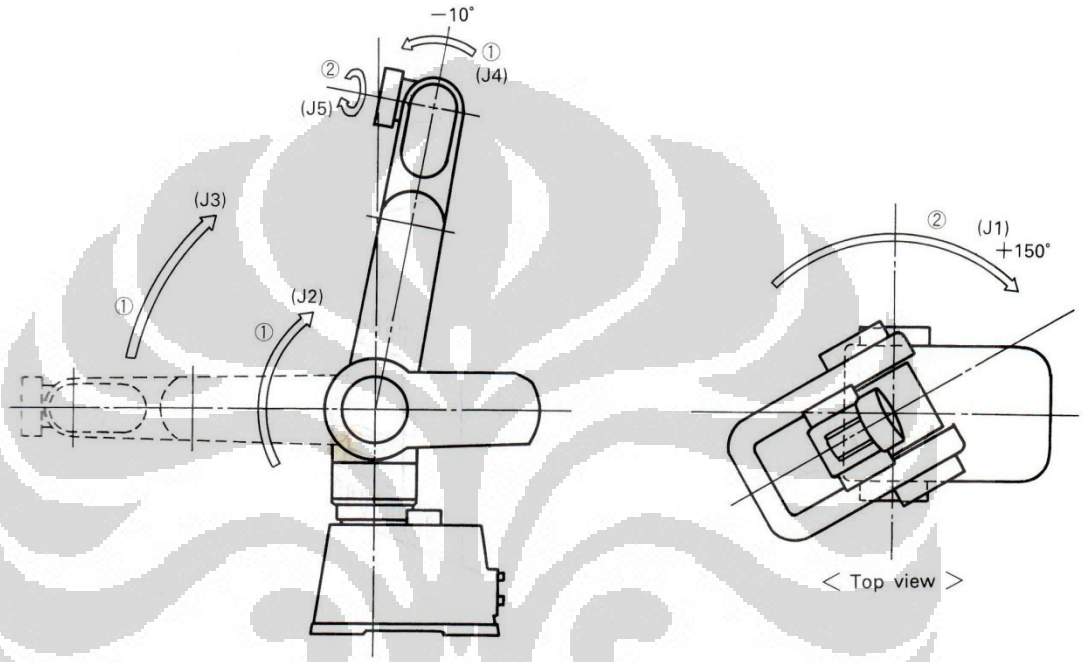


Gambar 3.6 *Electric Brake* pada Mitsubishi Movemaster RV-M1

Fungsi *electric brake* pada kedua *axis* tersebut sangat vital karena motor DC pada kedua axis tersebut tidak memiliki torsi diam yang cukup untuk menahan kedua axis tersebut sehingga dibantu oleh *electric brake*.

3.3 Posisi Default Robot

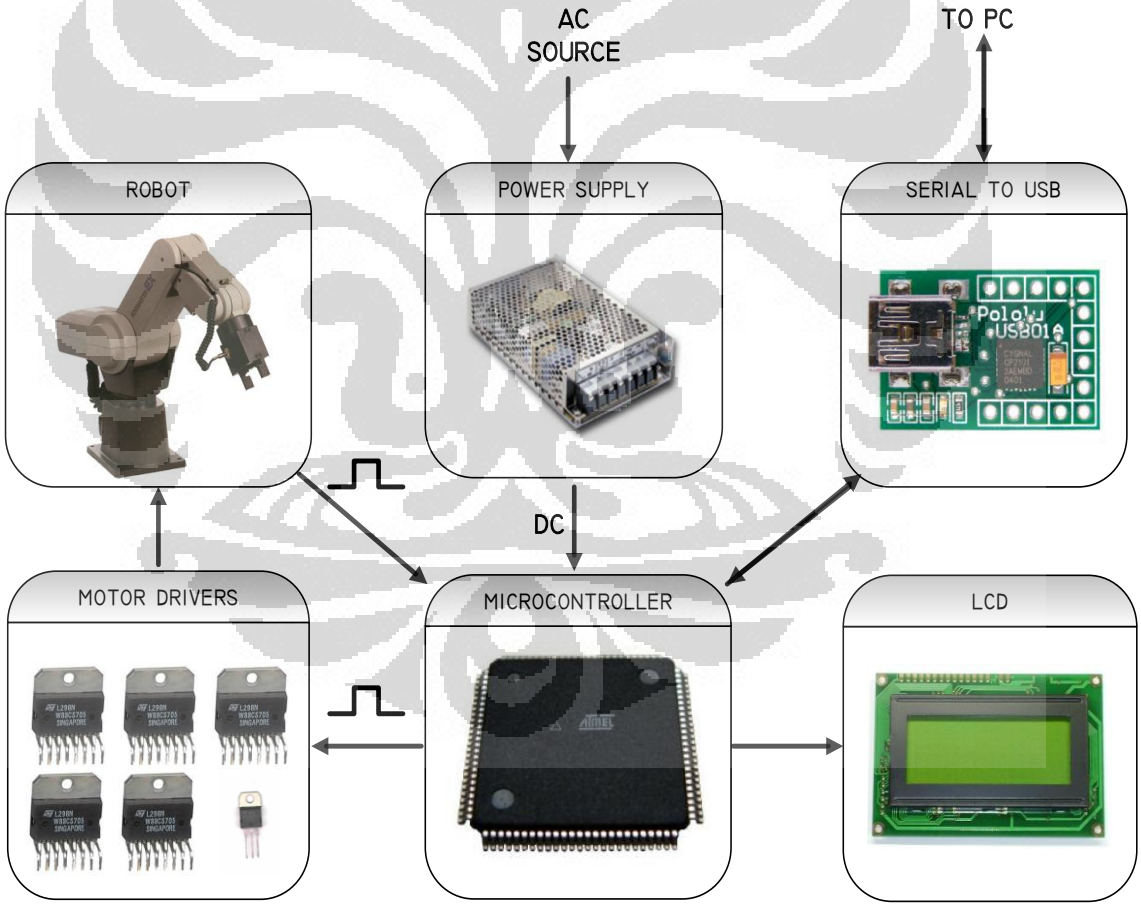
Posisi default robot merupakan posisi yang sangat penting karena seringkali kita membutuhkan acuan dalam menentukan pergerakan robot. Sesuai dengan buku petunjuk robot Movemaster RV-M1, posisi *default* robot dapat dilihat pada gambar berikut :



Gambar 3.7 Posisi Default Robot [3]

BAB 4
PENGEMBANGAN PERANGKAT KERAS SISTEM PENGENDALI

Berdasarkan tinjauan sistem mekanikal maka dilakukan pengembangan perangkat keras untuk pengendalian robot berbasis *web* ini. Perangkat keras yang digunakan dirancang untuk memenuhi kebutuhan pengendalian sistem mekanik. Berikut adalah diagram rancangan sistem pengendali :



Gambar 4.1 Rancangan Sistem Pengendali

4.1 Sensor

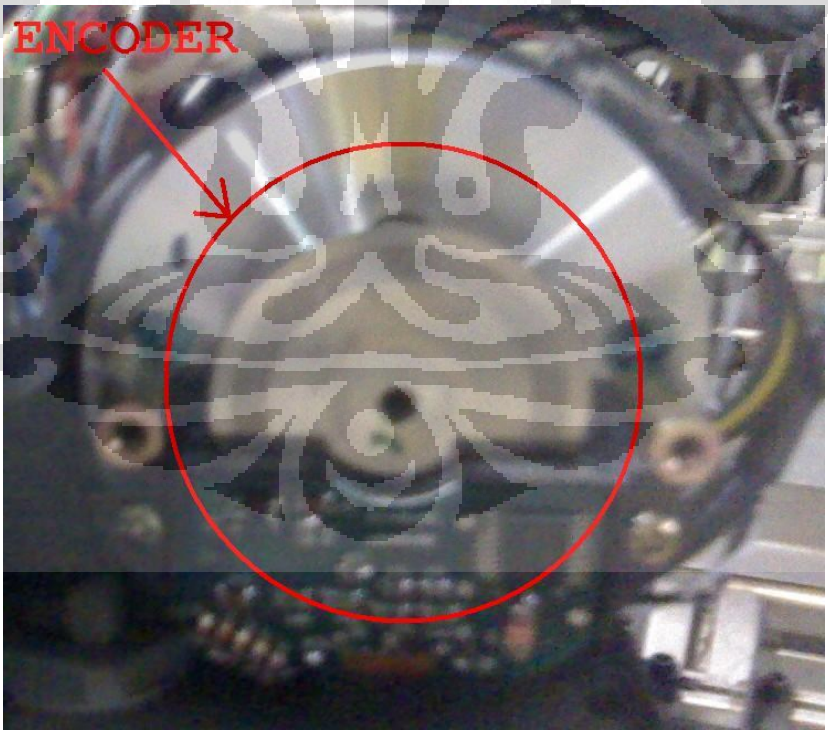
Robot Movemaster RV-M1 mempunyai beberapa sensor elektromekanikal. Sensor-sensor ini yang berfungsi untuk mendeteksi pergerakan robot dan mengirimkan data ke *microcontroller*.

4.1.1 Encoder

Encoder merupakan perangkat *electromechanical* yang berfungsi sebagai alat *feedback* berupa pulsa *digital* yang sangat bermanfaat dalam pengendalian motor DC. Setiap motor penggerak pada robot Movemaster RV-M1 memiliki *encoder* dengan spesifikasi sebagai berikut :

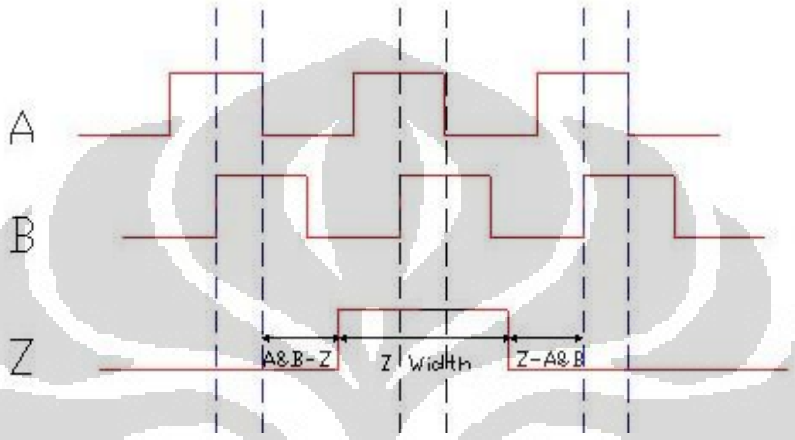
1. *Incremental Encoder*.
2. *Digital Voltage* 5V.
3. 200 CPR.

Berikut adalah gambar *encoder* yang terpasang pada motor DC :



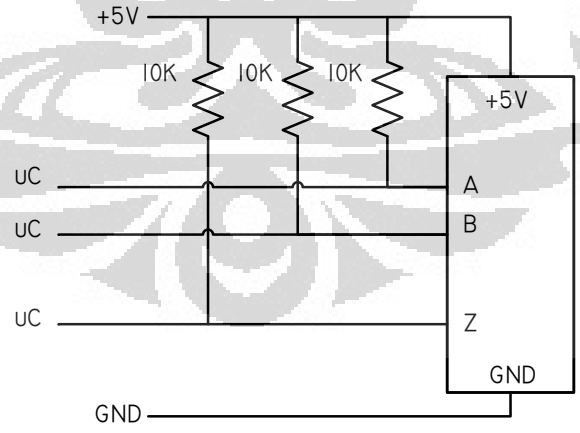
Gambar 4.2 *Encoder* pada Mitsubishi Movemaster RV-M1

Setiap encoder mengeluarkan 3 fasa yaitu A, B dan Z, setiap fasa ini mengeluarkan *timing* pulsa digital yang berbeda – beda, pada penelitian ini digunakan satu fasa saja untuk kendali motor DC berdasarkan sudut robot yaitu fasa A. Untuk fasa B dan Z dapat digunakan untuk pengembangan lebih lanjut. Sebuah encoder mempunyai *digital output* sebagai berikut :



Gambar 4.3 Sinyal *Encoder* [6]

Untuk *interface* pada encoder dengan microcontroller dapat digunakan contoh rangkaian elektronika sebagai berikut:



Gambar 4.4 *Interface* Encoder ke Microcontroller

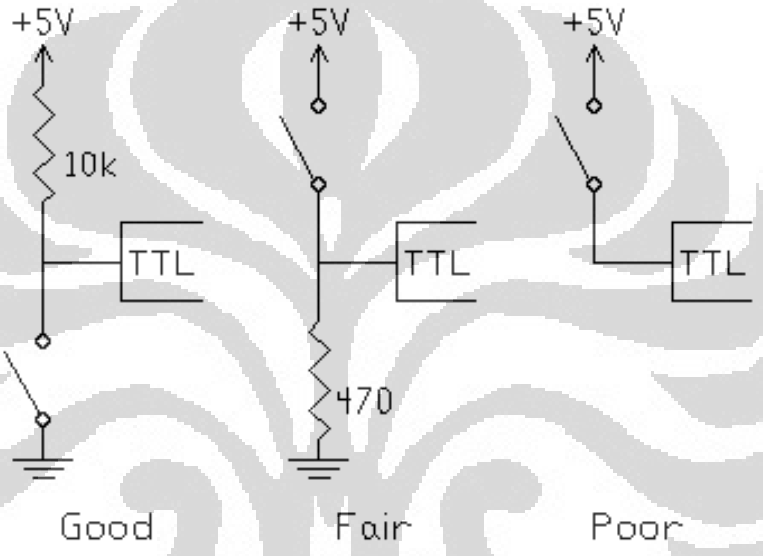
4.1.2 Limit Switch

Robot Movemaster RV-M1 memiliki 5 *limit switch* sebagai *feedback* posisi robot. *Limit switch* pada penelitian ini digunakan sebagai *feedback* pada program pencari posisi *default* robot sesuai dengan *datasheet* robot Movemaster RV-M1. Setiap *axis* memiliki 1 limit switch sebagai *sensor* posisi *default*. *Limit switch* yang digunakan adalah tipe *tactile* dengan kondisi NC (Normally Closed). Berikut adalah gambar dari *limit switch* tersebut :

Gambar 4.5 Limit Switch yang Digunakan Robot Movemaster RV-M1

Gambar 4.6 Limit Switch pada Mitsubishi Movemaster RV-M1

Limit switch merupakan perangkat *electromekanikal* yang bekerja dengan memanfaatkan sentuhan objek lain. Sentuhan dari objek lain dalam penelitian ini adalah lengan robot dapat merubah kondisi *limit switch* dari posisi NC ke *open state*. Pada posisi open arus listrik yang disuplai ke *microcontroller* terputus, kondisi ini dapat dimanfaatkan sebagai kondisi pada *microcontroller*. Untuk *interface limit switch* ke *microcontroller* dapat dilakukan dengan cara memberikan *pullup resistor* seperti gambar berikut :



Gambar 4.7 Interface Limit Switch ke Microcontroller [7]

Tujuan pemasangan *pullup resistor* adalah menghindari kondisi mengambang pada input *microcontroller.* Microcontroller biasanya mempunyai default *input* bernilai 1 (*high*) sehingga apabila ada sedikit perubahan dari luar (ketika sensor dipasang) maka microcontroller bisa saja berubah menjadi kondisi 0 (*low*), namun sulit untuk kembali ke kondisi *high* sehingga perlu dipasang *pullup resistor* untuk mengembalikan kondisi *input* tersebut. *Pullup resistor* bekerja dengan meningkatkan arus pada rangkaian input, pada saat *limit switch* maupun *encoder* tidak memberikan keluaran maka kondisi *input* adalah 1 sedangkan begitu mendapat *trigger* kondisi *input* menjadi 0.

4.2 Kendali Digital

Sistem kendali yang dikembangkan untuk mengendalikan robot Movemaster RV-M1 adalah kendali *digital*. Sistem kendali *digital* digunakan karena kemudahan untuk menerapkan logika kendali dan mempermudah pengembangan sistem lebih lanjut.

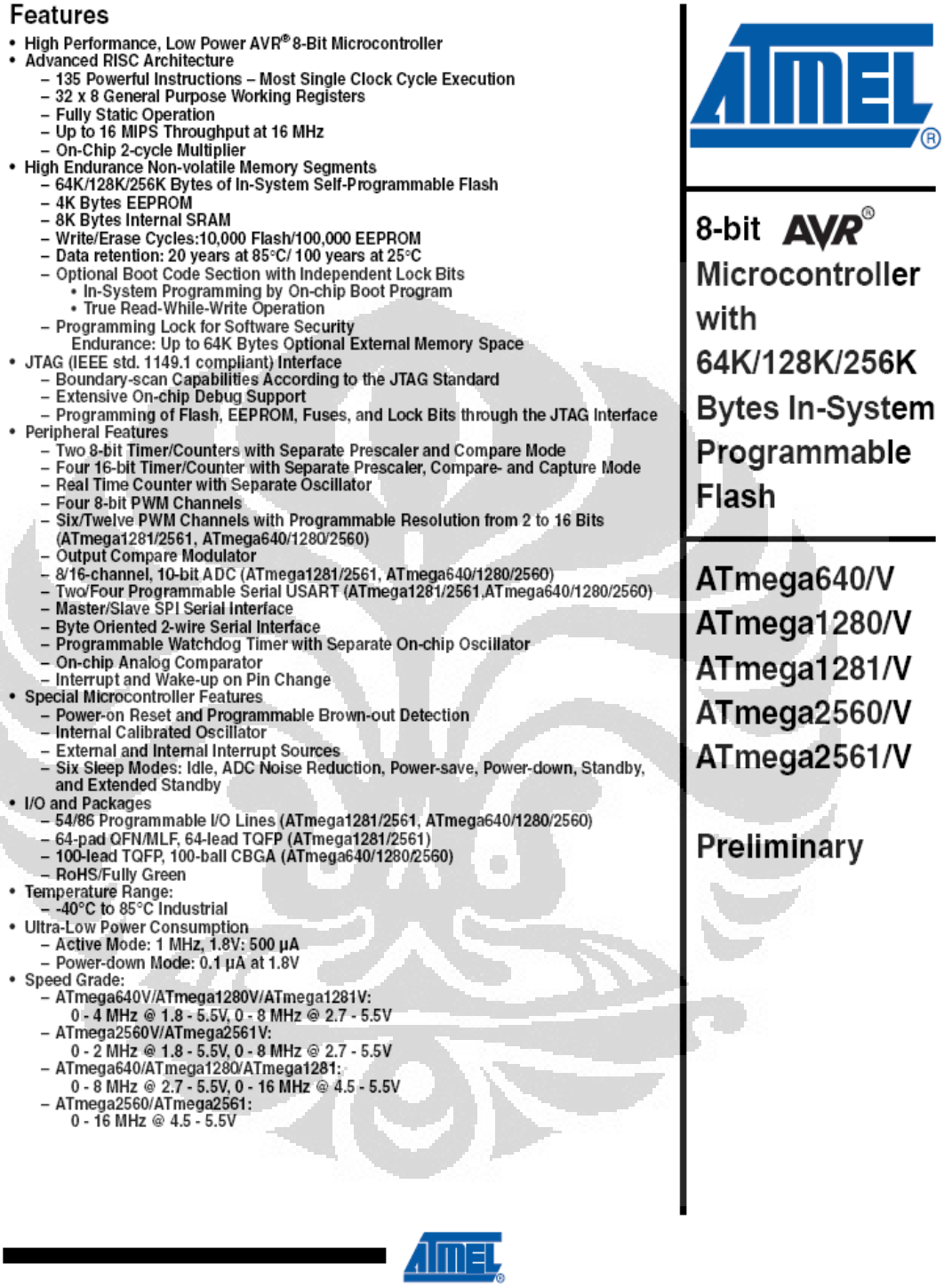
4.2.1 Microcontroller

Microcontroller merupakan perangkat elektronika yang menyerupai sebuah komputer namun berukuran lebih kecil dan biasanya berbentuk IC. Layaknya sebuah komputer, *microcontroller* merupakan sebuah CPU yang memiliki perangkat pelengkap seperti *crystal oscillator, timer, serial communication, digital I/O dan analog I/O*. *Microcontroller* menggunakan *memory* dengan kapasitas kecil yang dapat diprogram ulang dengan komputer melalui bahasa pemrograman yang bervariasi. *Microcontroller* dapat digunakan untuk mengendalikan berbagai perangkat elektronika dengan logika pemrograman, sehingga dalam dalam penelitian kontrol robot berbasis *web* ini digunakan *microcontroller* sebagai pengontrol robot. Tipe *microcontroller* yang dipilih adalah ATmega2560 dari Atmel. Pemilihan microcontroller jenis ini didasari dari kebutuhan *controller* dari robot movemaster RV-M1 yaitu :

1. Mempunyai minimal 5 PWM channel.
2. Mempunyai minimal 5 Interrupt channel.
3. Mampu melakukan komunikasi serial.
4. Kapasitas memory yang besar.
5. Kecepatan eksekusi yang tinggi.

Tabel 4.1 Spesifikasi ATmega2560 [8]

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega640	64KB	4KB	8KB	86	12	4	16
ATmega1280	128KB	4KB	8KB	86	12	4	16
ATmega1281	128KB	4KB	8KB	54	6	2	8
ATmega2560	256KB	4KB	8KB	86	12	4	16
ATmega2561	256KB	4KB	8KB	54	6	2	8



Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256K Bytes of In-System Self-Programmable Flash
  - 4K Bytes EEPROM
  - 8K Bytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    Endurance: Up to 64K Bytes Optional External Memory Space
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561,ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1 MHz, 1.8V: 500 µA
  - Power-down Mode: 0.1 µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega2560V/ATmega2561V:
    0 - 2 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
  - ATmega2560/ATmega2561:
    0 - 16 MHz @ 4.5 - 5.5V

8-bit AVR® Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

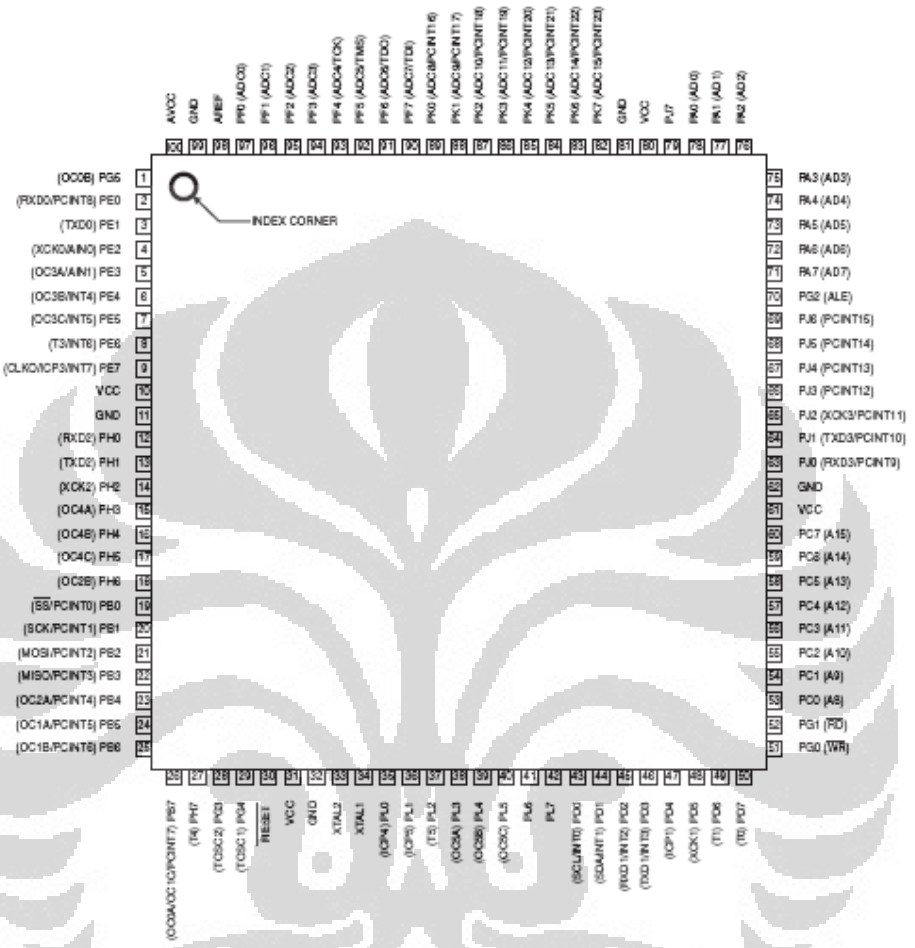
Preliminary

Gambar 4.8 Spesifikasi Lengkap ATmega2560 [8]

Kecepatan proses data bergantung pada penggunaan *oscillator* pada *microcontroller*. *Oscillator* merupakan sebuah perangkat yang menghasilkan frekuensi tertentu untuk menjadi *clock source* bagi microcontroller. *Clock source* pada *microcontroller* menjadi patokan waktu bagi *microcontroller* untuk melakukan sebuah instruksi. *Microcontroller* ATMEL pada umumnya mempunyai karakter satu clock satu instruksi dan memiliki *oscillator internal* yang terdapat didalam *microcontroller*. *Oscillator internal* yang terdapat pada *microcontroller* dirasa kurang cukup memadai untuk aplikasi kontrol robot berbasis *web* ini karena frekuensi hanya 1Mhz. Dengan frekuensi 1Mhz microcontroller hanya mampu mencapai kecepatan 1MIPS (*Million Instruction per Second*). Microcontroller ATmega2560 pada penelitian ini menggunakan *clock source* 16Mhz sehingga mampu mencapai kecepatan 16MIPS. Kecepatan ini dianggap cukup untuk dipakai dalam aplikasi pengendalian robot berbasis *web* ini. Berikut gambar adalah *oscillator* yang dipakai pada *microcontroller* ATmega 2560 :

Gambar 4.9 *Oscillator* 16Mhz yang Dipakai pada ATmega2560 [9]

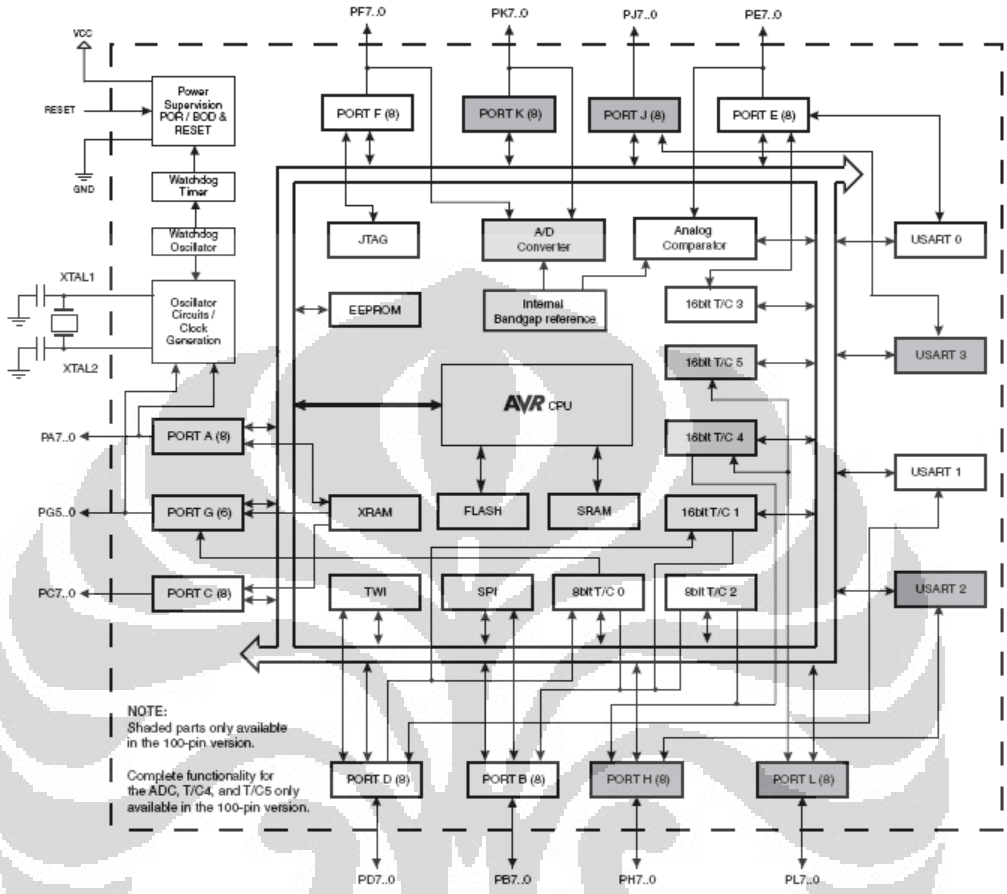
Microcontroller ATmega2560 mempunyai diagram *pinout* seperti gambar dibawah ini :



Gambar 4.10 *Pinout* ATmega2560 [8]

Seperti terlihat dalam gambar diatas fitur yang dimiliki ATmega2560 sudah cukup memadai untuk mengontrol robot movemaster RV-M1. Fitur yang ada tidak semua dipakai karena disengaja untuk pengembangan robot lebih lanjut. Setiap *microcontroller* mempunyai sistem arsitektur yang unik, ATmega2560 merupakan microcontroller 8-Bit RISC (*Reduced Instruction Set Computing*), sehingga aplikasinya cukup sederhana karena ada fungsi – fungsi pemrograman yang sudah disederhanakan.

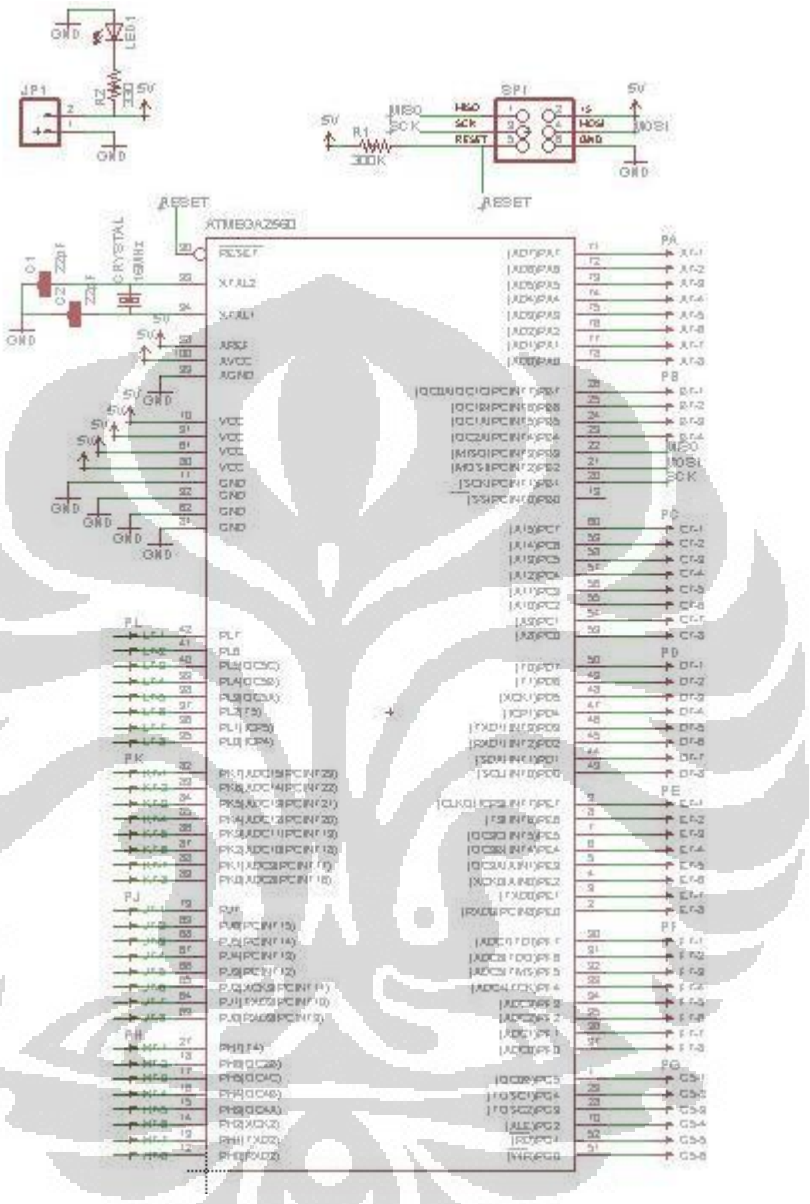
Berikut adalah *block diagram* dari *microcontroller* ATmega2560 :



Gambar 4.11 Block diagram ATmega2560 [8]

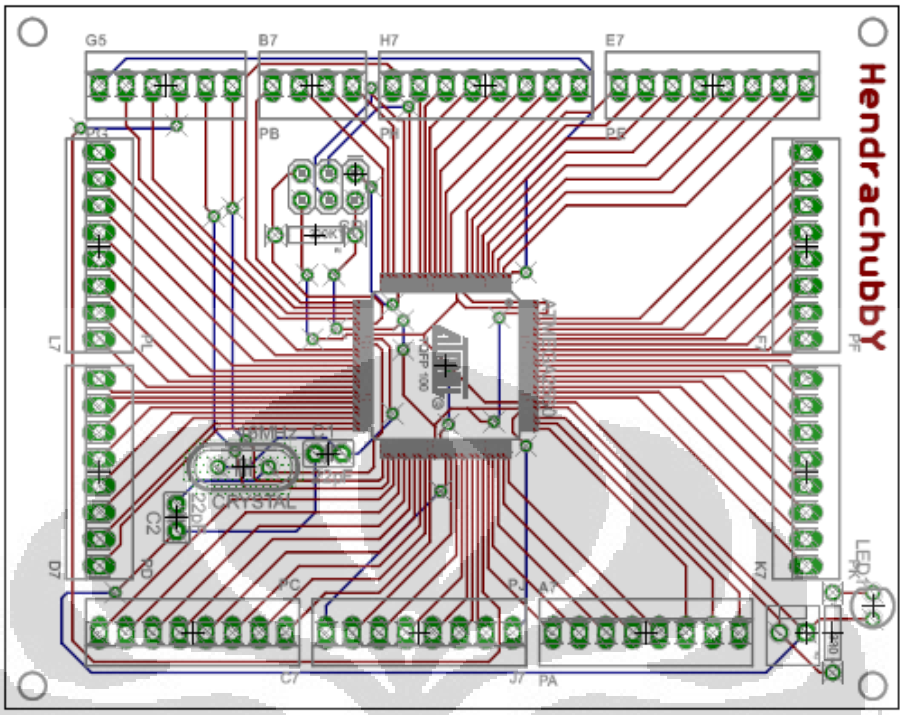
ATmega2560 diproduksi dalam bentuk IC, sehingga untuk *interfacing* dengan perangkat elektronika lain diperlukan *expansion board*. Sebenarnya *expansion board* untuk ATmega2560 sudah dirancang oleh ATMEL namun memiliki harga yang sangat mahal sehingga dibutuhkan perancangan *expansion board* sendiri dari ATmega2560. Untuk itu dirancang *expansion board* yang sederhana menggunakan *oscillator* 16Mhz. Perancangan *expansion board* ini harus berdasarkan pada *datasheet* ATmega2560 yang terlampir dan menggunakan *software* desain elektronika Eagle 5.6.0 dari CADsoft karena mudah digunakan dan *open source*.

Berikut adalah schematic dari *expansion board* ATmega2560 :

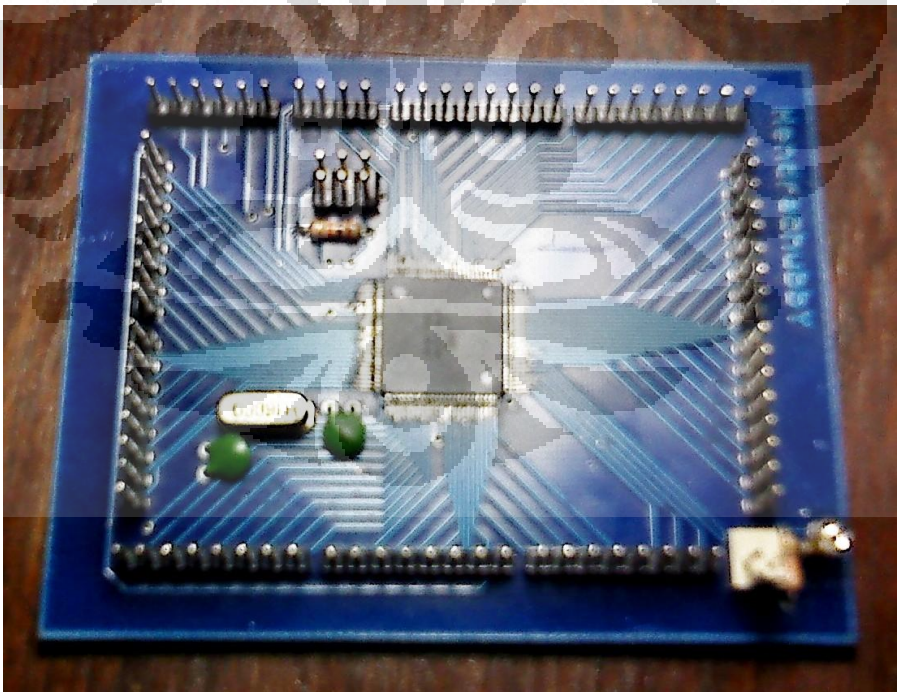


Gambar 4.12 Schematic Expansion Board ATmega2560

Seperti telihat pada gambar, setiap pin di expansi, hanya *pin – pin support* saja yang ditambahkan *hardware* seperti *oscillator* dan *port programming* 6 pin standar Atmel karena untuk memprogram ATmega2560 digunakan *programmer* AVR ISP MKII.



Gambar 4.13 PCB Layout Expansion Board ATmega2560



Gambar 4.14 Expansion Board ATmega2560

Sesuai dengan fitur yang terdapat pada ATmega2560 maka dibuat pin assignment yang sesuai dengan kebutuhan dan program yang terdapat didalamnya. Pin assignment ini dapat dilihat pada table berikut :

Tabel 4.2 Pin Assignment ATmega2560

Microcontroller I/O	Input atau Output	GPIO General Purpose I/O	Fungsi pada Axis	Cara Koneksi
PORTA.0	Output	Driver Input 1 Axis 1	Kontrol Arah Motor	Langsung
PORTA.1	Output	Driver Input 2 Axis 1	Kontrol Arah Motor	Langsung
PORTA.2	Output	Driver Input 1 Axis 2	Kontrol Arah Motor	Langsung
PORTA.3	Output	Driver Input 2 Axis 2	Kontrol Arah Motor	Langsung
PORTA.4	Output	Driver Input 1 Axis 3	Kontrol Arah Motor	Langsung
PORTA.5	Output	Driver Input 2 Axis 3	Kontrol Arah Motor	Langsung
PORTA.6	Output	Driver Input 1 Axis 4	Kontrol Arah Motor	Langsung
PORTA.7	Output	Driver Input 2 Axis 4	Kontrol Arah Motor	Langsung
PORTB.4	Input	Limit Switch Axis 5	Digital Sensing	10K Pullup Resistor
PORTB.5	Output	PWM Channel Axis 4	Kontrol Kecepatan	Langsung
PORTB.6	Output	PWM Channel Axis 5	Kontrol Kecepatan	Langsung
PORTB.7	Output	PWM Gripper	Kontrol Kecepatan	Langsung
PORTC.0	Output	LCD RS	LCD Kontrol	Langsung
PORTC.1	Output	LCD R/W	LCD Kontrol	Langsung
PORTC.2	Output	LCD E	LCD Kontrol	Langsung
PORTC.4	Output	LCD Data I/O	LCD Kontrol	Langsung
PORTC.5	Output	LCD Data I/O	LCD Kontrol	Langsung
PORTC.6	Output	LCD Data I/O	LCD Kontrol	Langsung
PORTC.7	Output	LCD Data I/O	LCD Kontrol	Langsung
PORTD.0	Output	Driver Input 1 Gripper	Kontrol Arah Motor	Langsung
PORTD.1	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTD.2	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTD.3	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTD.4	Output	Driver Input 2 Gripper	Kontrol Arah Motor	Langsung
PORTE.4	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTE.5	Input	Encoder Input	Feedback	10K Pullup Resistor
PORTF.0	Output	Brake Signal	Kontrol Brake	Langsung
PORTF.1	Output	Brake Signal	Kontrol Brake	Langsung
PORTF.2	Output	Driver Input 1 Axis 5	Kontrol Arah Motor	Langsung
PORTF.3	Output	Driver Input 2 Axis 5	Kontrol Arah Motor	Langsung
PORTF.4	Input	Limit Switch Axis 1	Feedback	10K Pullup Resistor
PORTF.5	Input	Limit Switch Axis 2	Feedback	10K Pullup Resistor
PORTF.6	Input	Limit Switch Axis 3	Feedback	10K Pullup Resistor
PORTF.7	Input	Limit Switch Axis 4	Feedback	10K Pullup Resistor
PORTH.0	Input	Serial Communication	PC Interface	Langsung
PORTH.1	Input	Serial Communication	PC Interface	Langsung
PORTL.3	Output	PWM Channel Axis 1	Kontrol Kecepatan	Langsung
PORTL.4	Output	PWM Channel Axis 2	Kontrol Kecepatan	Langsung
PORTL.5	Output	PWM Channel Axis 3	Kontrol Kecepatan	Langsung

4.2.2 Motor Driver

Dalam pengendalian motor DC melalui *microcontroller* diperlukan perangkat elektronika yaitu *motor driver*. *Motor driver* diperlukan karena perbedaan suplai daya yang dikeluarkan *microcontroller* dengan yang dibutuhkan oleh motor DC. *Microcontroller* menggunakan *power* 0-5 V sedangkan motor DC menggunakan 24VDC. Selain itu motor yang dikontrol juga harus mempunyai kemampuan berbalik arah dan pengaturan kecepatan, algoritma kontrol arah dan kecepatan ini diberikan oleh *microcontroller* dan dijalankan oleh *motor driver*. *Motor driver* juga berguna sebagai pengaman dari arus balik akibat adanya medan elektromagnetis dari motor, arus balik ini sangat berbahaya bagi *microcontroller*.

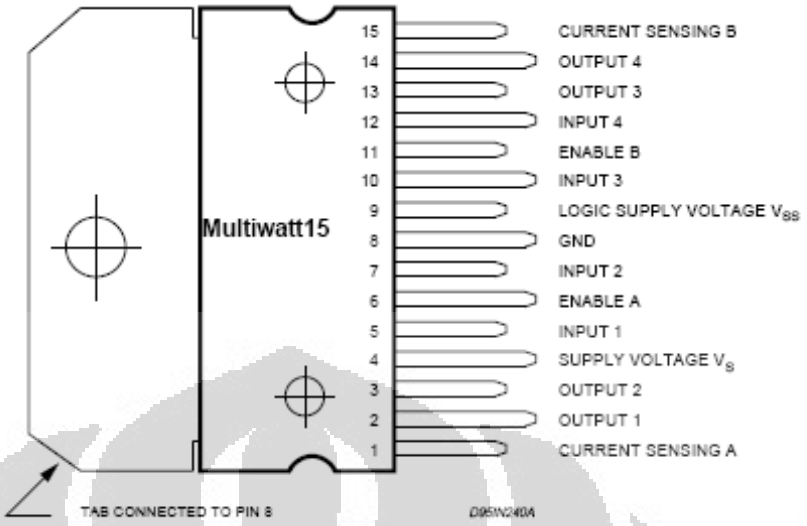
Motor driver untuk menggerakkan motor pada robot movemaster RV-M1 harus memiliki kemampuan sebagai berikut :

1. Power Supply 24 V.
2. PWM Support.
3. Arus > 3A.

Ini disebabkan karena spesifikasi motor DC pada robot movemaster RV-M1 memerlukan *motor driver* dengan kemampuan tersebut. Maka dirancang *motor driver* dengan basis IC L298D. Berikut adalah spesifikasi IC driver motor tersebut berdasarkan *datasheet*-nya :

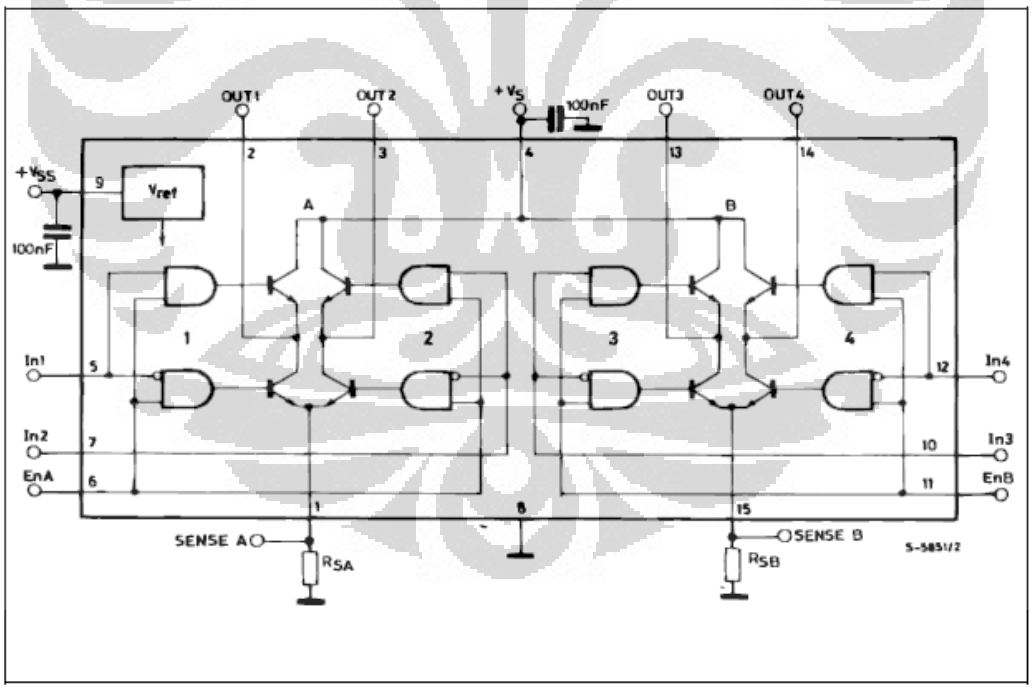
Tabel 4.3 Spesifikasi L298D [10]

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C



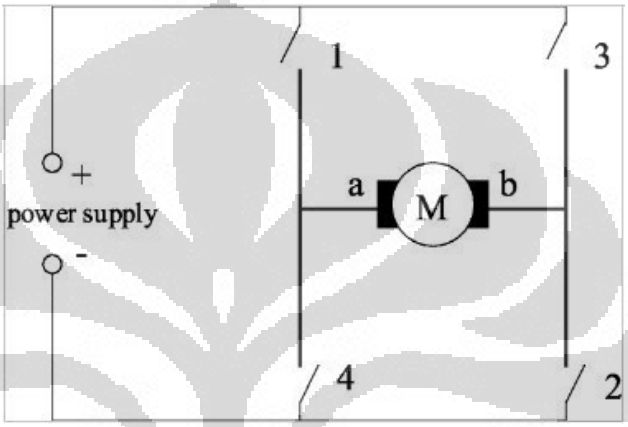
Gambar 4.15 IC L298D [10]

IC *motor driver* L298D mempunyai *block diagram* sebagai berikut :



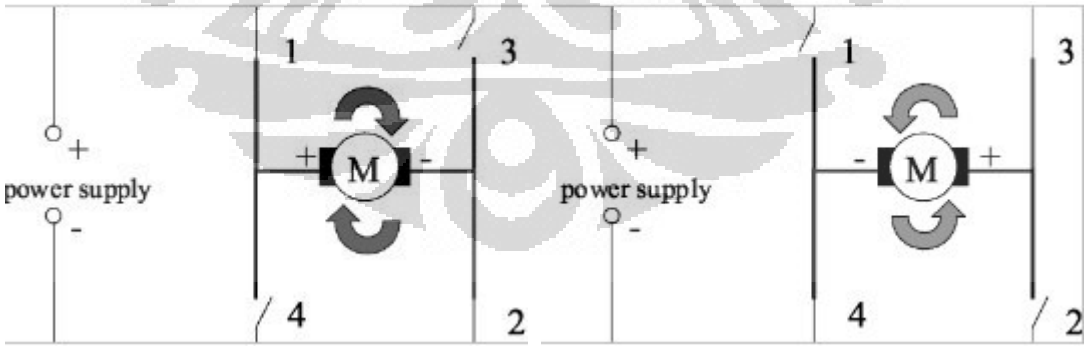
Gambar 4.16 Block Diagram L298D [10]

Dari spesifikasi diatas kemudian dirancang *board control* untuk *interfacing* ke *microcontroller*. IC L298 sebenarnya terdiri dari 2 *H-bridge* yang terintegrasi pada satu chip. *H-Bridge* atau yang diterjemahkan secara kasar sebagai "Jembatan H", adalah sebuah rangkaian dimana motor menjadi titik tengahnya dengan dua jalur yang bisa dibuka tutup untuk melewatkan arus pada motor tersebut, persis seperti huruf "H" (dengan motor berada pada garis horizontal).



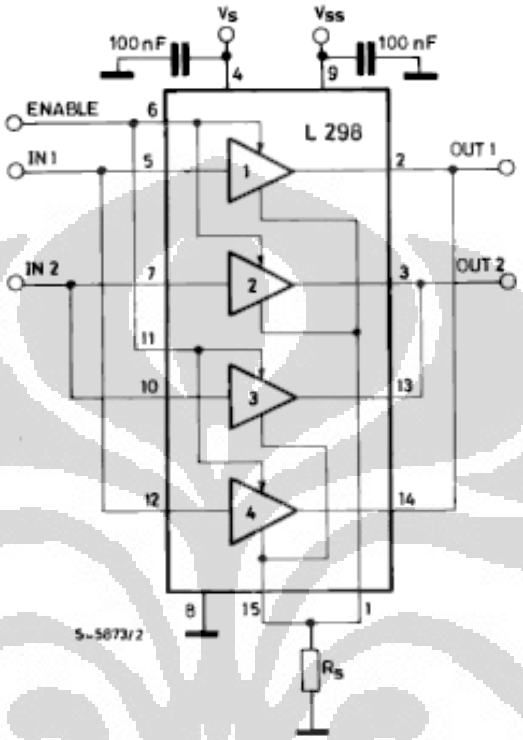
Gambar 4.17 Rangkaian H-Bridge [11]

Rangkaian ini memungkinkan motor dapat berputar secara CW (*Clock Wise*) dan CCW (*Counter Clock Wise*) sesuai dengan logika yang diberikan oleh *microcontroller*.



Gambar 4.18 Pengaturan Arah Putaran Motor pada *H-Bridge* [11]

Kedua H-Bridge yang terdapat pada IC L298 dapat dimanfaatkan dengan memparalel kedua H-Bridge tersebut sehingga dapat menahan arus sampai dengan 4A. Berikut adalah skematik dari rangkaian tersebut :



Gambar 4.19 Skematik L298D Paralel [10]

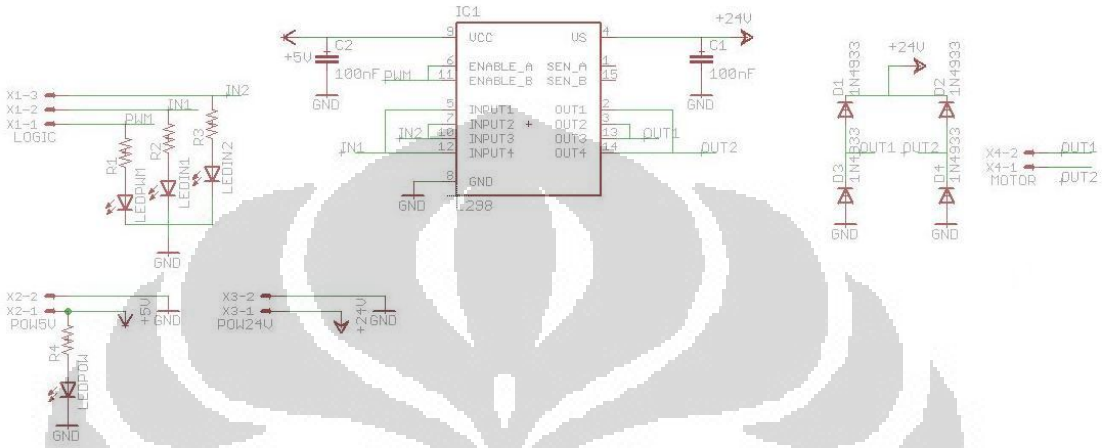
Untuk mengaktifasi pergerakan motor, IC L298D memerlukan *input* dengan pola yang dapat dilihat pada *truth table* berikut :

Tabel 4.4 *Truth Table* L298D [10]

Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

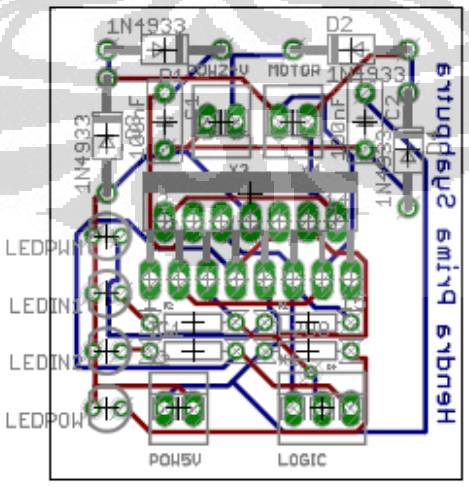
L = Low H = High X = Don't care

Untuk mempermudah *programmer* dalam mengakses *motor driver* ini ditambahkan berberapa LED dengan warna yang berbeda. Hal ini dapat membantu *programmer* memastikan dan memeriksa adanya *input* PWM maupun sinyal arah yang sesuai dengan *truth table* sehingga skematik dasarnya dikembangkan menjadi :



Gambar 4.20 Skematik L298D Paralel dengan LED

Berikut adalah PCB *layout double layer* dari *motor driver board* tersebut setelah dilengkapi dengan LED dan back EMF protection. *Back EMF Protection* berfungsi untuk menahan arus balik yang ditimbulkan oleh motor akibat adanya reaksi elektromagnetis pada motor DC :



Gambar 4.21 *PCB Layout Motor Driver*

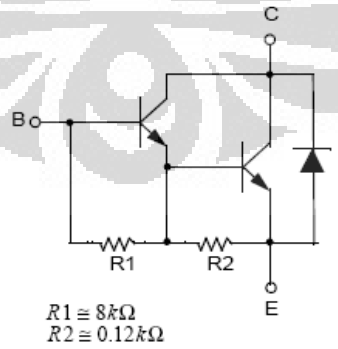
4.2.3 Driver Electric Brake

Pada robot movemaster RV-M1 terdapat 2 *electric brake* yaitu pada *axis* 2 dan 3. Electric brake ini bersifat NC (Normally Closed) sehingga perlu adanya driver untuk mengaktifkan brake tersebut sehingga menjadi *open state*. Berdasarkan spesifikasi, *electric brake* tersebut membutuhkan tegangan 12V sehingga dibuat perangkat *switching* dengan menggunakan IC TIP122. TIP122 merupakan transistor *darlington pair* yang dapat digunakan untuk aplikasi *switching* linear dengan spesifikasi sebagai berikut :

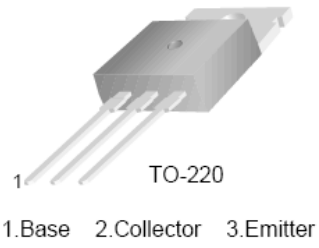
Tabel 4.5 Spesifikasi TIP122 [12]

Symbol	Parameter		Value	Units
V_{CBO}	Collector-Base Voltage	: TIP120	60	V
		: TIP121	80	V
		: TIP122	100	V
V_{CEO}	Collector-Emitter Voltage	: TIP120	60	V
		: TIP121	80	V
		: TIP122	100	V
V_{EBO}	Emitter-Base Voltage		5	V
I_C	Collector Current (DC)		5	A
I_{CP}	Collector Current (Pulse)		8	A
I_B	Base Current (DC)		120	mA
P_C	Collector Dissipation (T_a=25°C)		2	W
P_C	Collector Dissipation (T_C=25°C)		65	W
T_J	Junction Temperature		150	°C
T_{STG}	Storage Temperature		- 65 ~ 150	°C

Block diagram TIP 122 :

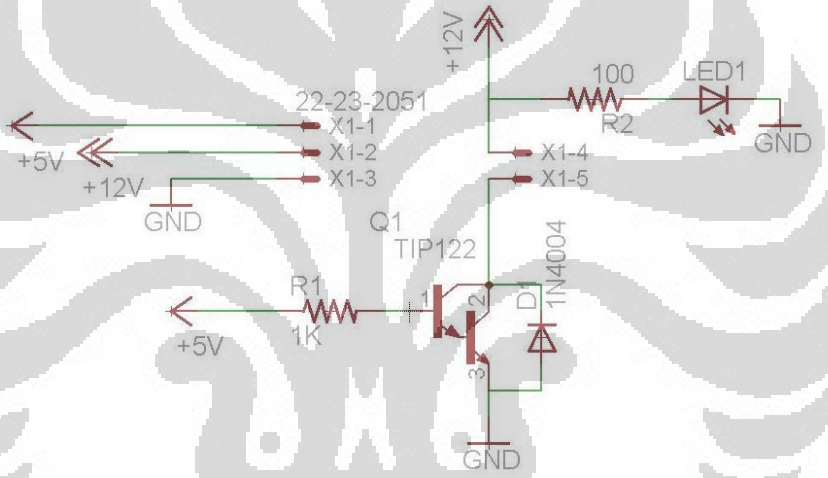


Gambar 4.22 *Block Diagram* TIP122 [12]



Gambar 4.23 TIP122 [12]

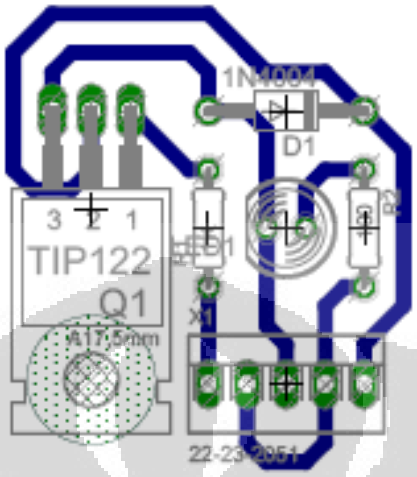
Berikut adalah skematik rangkaian *switching* menggunakan transistor TIP122 tersebut :



Gambar 4.24 Skematik Driver *Electric Brake*

Seperti terlihat pada gambar skematik, rangkaian *switching* ini diaktifasi oleh *microcontroller* yang memberikan tegangan 5V pada basis transistor. Apabila ada tegangan tersebut maka transistor akan teraktifasi sehingga tegangan sebesar 12V mengalir dari collector ke emittornya, tegangan inilah yang akan mengaktifkan *electric brake* pada robot movemaster RV-M1. *Electric brake* dapat menimbulkan tegangan balik karena *electric brake* bekerja seperti *solenoid* yang terdiri dari gulungan induksi elektromagnetis sehingga harus dilengkapi dengan dioda pengaman untuk mencegah arus balik merusak transistor TIP122.

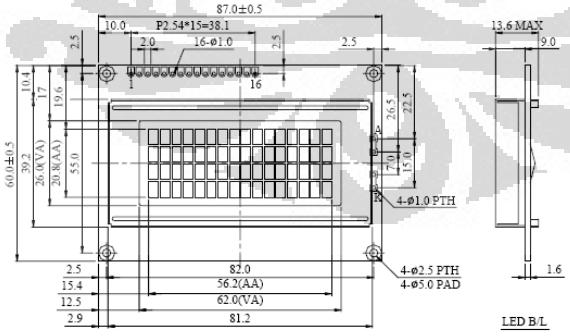
Berikut adalah layout PCB dari rangkaian *switching* ini :



Gambar 4.25 *PCB Layout Driver Electric Brake*

4.2.4 Modul LCD (Liquid Crystal Display)

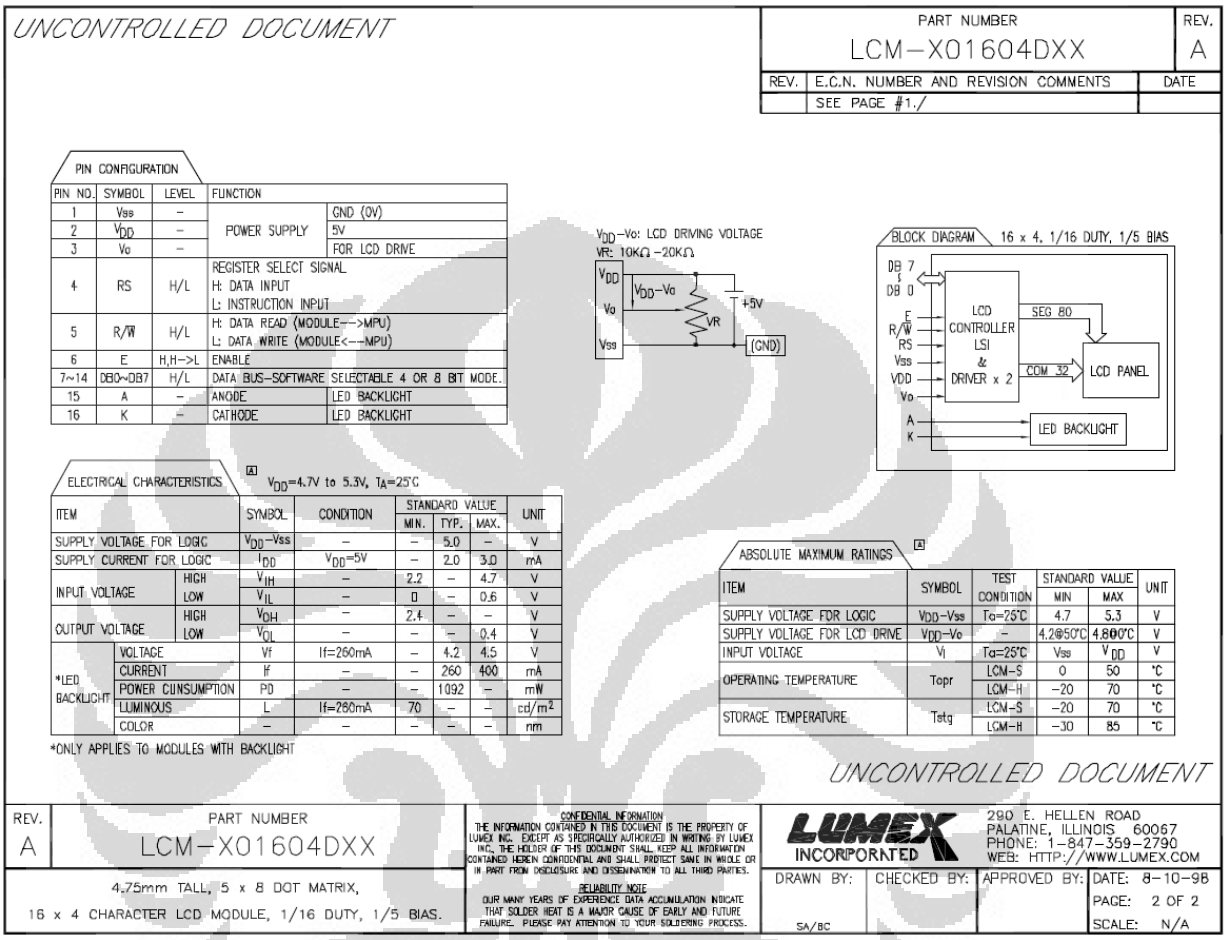
LCD digunakan dalam rangkaian kontrol sebagai perangkat display sekaligus memberikan informasi kepada user mengenai berbagai macam proses yang terjadi dalam microcontroller. Modul LCD ini sangat berguna untuk membantu programmer dalam debugging program pada microcontroller. Modul LCD yang digunakan adalah LCD karakter 16 x 4.



PIN NO.	SYMBOL
1	Vss
2	Vdd
3	Vo
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	A/Vee
16	K

Gambar 4.26 *Pinout* LCD 16 x 4 [13]

LCD yang dipakai adalah tipe LCM-S01604DSR dari Lumex Inc, berikut adalah spesifikasi dan gambar dari LCD tersebut :



UNCONTROLLED DOCUMENT

PART NUMBER	REV.
LCM−X01604DXX	A

REV.	E.C.N. NUMBER AND REVISION COMMENTS	DATE
	SEE PAGE #1./	

PIN CONFIGURATION

PIN NO.	SYMBOL	LEVEL	FUNCTION	
1	Vss	−	POWER SUPPLY	GND (0V)
2	V_{DD}	−		5V
3	Vo	−		FOR LCD DRIVE
4	RS	H/L	REGISTER SELECT SIGNAL H: DATA INPUT L: INSTRUCTION INPUT	
5	R/W	H/L	H: DATA READ (MODULE−−>MPU) L: DATA WRITE (MODULE<−−MPU)	
6	E	H,H−>L	ENABLE	
7~14	DB0~DB7	H/L	DATA BUS−SOFTWARE SELECTABLE 4 OR 8 BIT MODE.	
15	A	−	ANODE	LED BACKLIGHT
16	K	−	CATHODE	LED BACKLIGHT

V_{DD}−Vo: LCD DRIVING VOLTAGE
VR: 10KΩ −20KΩ

BLOCK DIAGRAM 16 x 4, 1/16 DUTY, 1/5 BIAS

ELECTRICAL CHARACTERISTICS [A] V_{DD}=4.7V to 5.3V, T_A=25°C

ITEM			SYMBOL	CONDITION	STANDARD VALUE MIN.	TYP.	MAX.	UNIT
SUPPLY VOLTAGE FOR LOGIC			V_{DD}−Vss	−	−	5.0	−	V
SUPPLY CURRENT FOR LOGIC			I_{DD}	V_{DD}=5V	−	2.0	3.0	mA
INPUT VOLTAGE		HIGH	V_{IH}	−	2.2	−	4.7	V
		LOW	V_{IL}	−	0	−	0.6	V
OUTPUT VOLTAGE		HIGH	V_{OH}	−	2.4	−	−	V
		LOW	V_{OL}	−	−	−	0.4	V
*LED BACKLIGHT	VOLTAGE		Vf	If=260mA	−	4.2	4.5	V
	CURRENT		If	−	−	260	400	mA
	POWER CONSUMPTION		PD	−	−	1092	−	mW
	LUMINOUS		L	If=260mA	70	−	−	cd/m²
	COLOR		−	−	−	−	−	nm

*ONLY APPLIES TO MODULES WITH BACKLIGHT

ABSOLUTE MAXIMUM RATINGS [A]

ITEM	SYMBOL	TEST CONDITION	STANDARD VALUE MIN	MAX	UNIT
SUPPLY VOLTAGE FOR LOGIC	V_{DD}−Vss	Ta=25°C	4.7	5.3	V
SUPPLY VOLTAGE FOR LCD DRIVE	V_{DD}−Vo	−	4.2@50°C	4.8@0°C	V
INPUT VOLTAGE	Vi	Ta=25°C	Vss	V_{DD}	V
OPERATING TEMPERATURE	Topr	LCM−S	0	50	°C
		LCM−H	−20	70	°C
STORAGE TEMPERATURE	Tstg	LCM−S	−20	70	°C
		LCM−H	−30	85	°C

UNCONTROLLED DOCUMENT

REV.	PART NUMBER
A	LCM−X01604DXX

4.75mm TALL, 5 x 8 DOT MATRIX,
16 x 4 CHARACTER LCD MODULE, 1/16 DUTY, 1/5 BIAS.

CONFIDENTIAL INFORMATION
THE INFORMATION CONTAINED IN THIS DOCUMENT IS THE PROPERTY OF LUMEX INC. EXCEPT AS SPECIFICALLY AUTHORIZED IN WRITING BY LUMEX INC., THE HOLDER OF THIS DOCUMENT SHALL KEEP ALL INFORMATION CONTAINED HEREIN CONFIDENTIAL AND SHALL PROTECT SAME IN WHOLE OR IN PART FROM DISCLOSURE AND DISSEMINATION TO ALL THIRD PARTIES.

RELIABILITY NOTE
OUR MANY YEARS OF EXPERIENCE DATA ACCUMULATION INDICATE THAT SOLDER HEAT IS A MAJOR CAUSE OF EARLY AND FUTURE FAILURE. PLEASE PAY ATTENTION TO YOUR SOLDERING PROCESS.

LUMEX INCORPORATED
2290 E. HELLEN ROAD
PALATINE, ILLINOIS 60067
PHONE: 1−847−359−2790
WEB: HTTP://WWW.LUMEX.COM

DRAWN BY:	CHECKED BY:	APPROVED BY:	DATE: 8−10−98
SA/BC			PAGE: 2 OF 2
			SCALE: N/A

Gambar 4.27 Spesifikasi LCD 16 x 4 [13]

Gambar 4.28 LCD 16 x 4 LCM-S01604DSR [13]

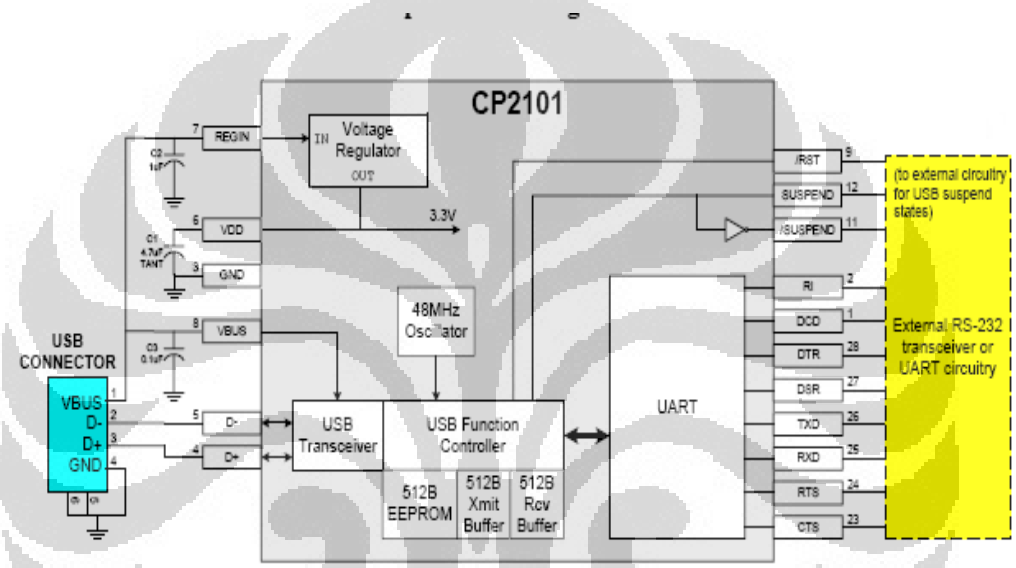
4.2.5 Adaptor USB – *Serial*

Untuk melakukan *transfer* data dari *microcontroller* ke komputer *server* diperlukan komunikasi antara kedua perangkat tersebut. Komunikasi *serial* merupakan komunikasi yang dimana pengiriman data dilakukan per bit. Data yang dikirimkan merupakan data yang berupa ASCII yang tercantum pada tabel berikut :

Tabel 4.6 Tabel ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

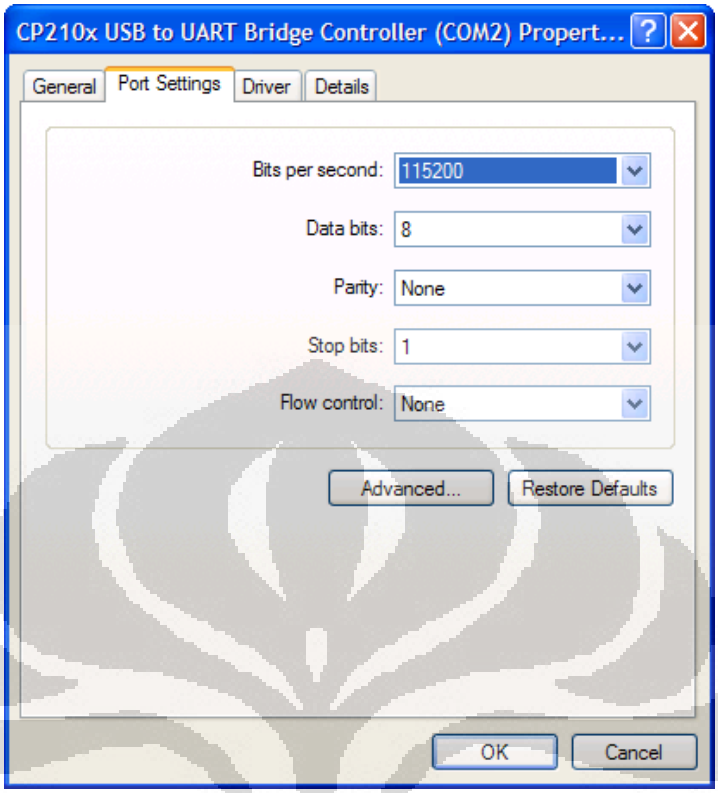
Adaptor serial ke USB merupakan perangkat yang menghubungkan *microcontroller* ke komputer *server* dari PORT RX dan TX pada *microcontroller* ke PORT USB pada komputer *server*. *Adaptor* ini melakukan konversi level tegangan dari komputer ke *microcontroller* dan sebaliknya. Hal ini dilakukan karena komputer menggunakan voltase logika 12V sedangkan *microcontroller* menggunakan voltase logika 5V. Adaptor USB – Serial ini menggunakan IC CP2101 *single chip USB to UART controller* yang mempunyai *block diagram* sebagai berikut :



Gambar 4.29 Block Diagram IC CP2101 [14]

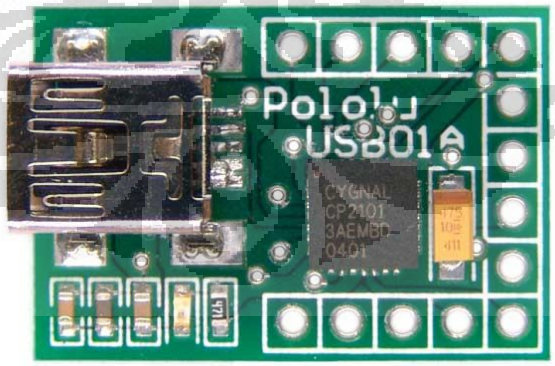
Seperti terlihat pada *block diagram*, IC tersebut melakukan konversi level tegangan dari PC ke microcontroller dan sebaliknya. Untuk konfigurasi komunikasi serial antara komputer server dengan microcontroller digunakan konfigurasi sebagai berikut :

1. Baud Rate : 115200 Bps.
2. Data Bits : 8 Bit.
3. Parity : None.
4. Stop Bits : 1.
5. Flow Control : None.



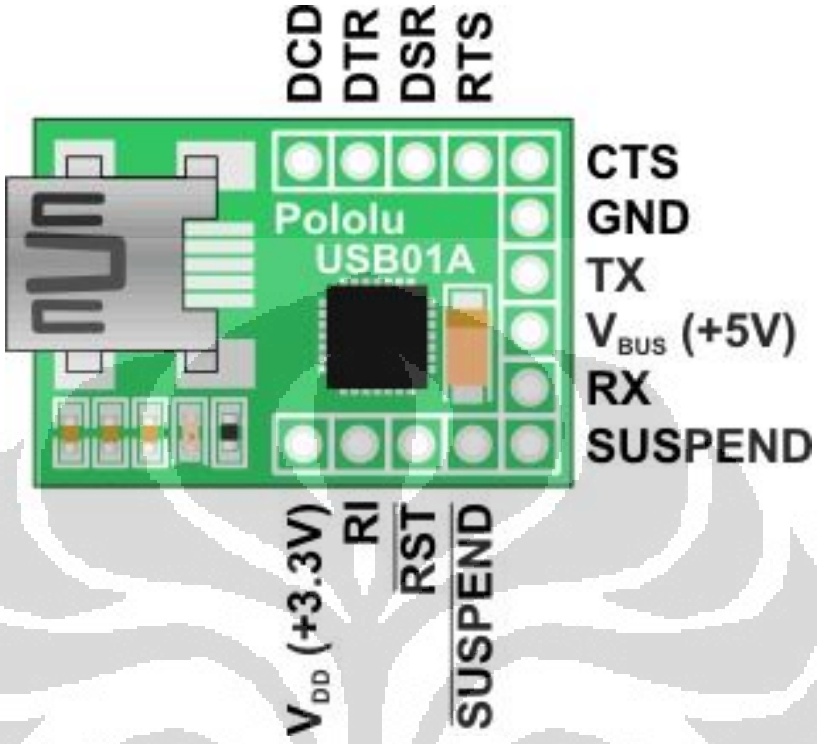
Gambar 4.30 *Setting* Komunikasi Serial Pada PC

Board dari IC CP2101 ini dibuat oleh *Pololu* (www.pololu.com), berikut adalah *board* dari IC CP2101 tersebut :



Gambar 4.31 Pololu USB to Serial Board [14]

Berikut adalah *Pinout* dari Pololu USB to Serial Board :



Gambar 4.32 Pololu USB to Serial Board Pinout [14]

Board ini mendapatkan daya dari port USB pada komputer yang memiliki tegangan 5V dan bisa langsung terkoneksi ke *microcontroller* dengan menggunakan PIN RX dan TX dari *board* ini.

4.2.6 Power Supply

Power supply untuk memberikan daya pada rangkaian elekronik adalah *switching power supply* dengan tegangan 5V, 12V dan 24V. *Power supply* tipe *switching* memungkinkan pemberian daya yang stabil pada range tegangan yang berbeda – beda. *Power supply* pada penelitian ini merupakan *power supply* produksi dari meanwell tipe S-100F untuk tegangan 12V dan 24V. *Power supply* ini memiliki proteksi *overload* yang sudah terdapat pada rangkaiannya. Sedangkan untuk tegangan 5V digunakan *power supply* bekas yang biasa dipakai pada alat *scanner*.

Berikut adalah spesifikasi power supply 24V dan 12V tersebut :

Tabel 4.7 Tabel Spesifikasi *Power supply* [15]

MODEL		S-100F-5	S-100F-7.5	S-100F-12	S-100F-15	S-100F-24	S-100F-48
OUTPUT	DC VOLTAGE	5V	7.5V	12V	15V	24V	48V
	RATED CURRENT	20A	13.5A	8.5A	6.7A	4.5A	2.2A
	CURRENT RANGE	0 ~ 20A	0 ~ 13.5A	0 ~ 8.5A	0 ~ 6.7A	0 ~ 4.5A	0 ~ 2.2A
	RATED POWER	100W	101W	102W	100.5W	108W	105.6W
	RIPPLE & NOISE (max.) Note.2	100mVp-p	125mVp-p	125mVp-p	125mVp-p	150mVp-p	150mVp-p
	VOLTAGE ADJ. RANGE	4.5 ~ 5.5V	6.75 ~ 8.25V	10.8 ~ 13.2V	13.5 ~ 16.5V	21.6 ~ 26.4V	43.2 ~ 52.8V
	VOLTAGE TOLERANCE Note.3	±2.0%	±1.0%	±1.0%	±1.0%	±1.0%	±1.0%
	LINE REGULATION	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
	LOAD REGULATION	±1.0%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
	SETUP, RISE TIME	1000ms, 30ms at full load					
	HOLD UP TIME (Typ.)	20ms at full load					
INPUT	VOLTAGE RANGE	88 ~ 132VAC/176 ~ 264VAC selected by jumper or switch 248 ~ 370VDC					
	FREQUENCY RANGE	47 ~ 63Hz					
	EFFICIENCY (Typ.)	76%	78%	80%	81%	83%	84%
	AC CURRENT (Typ.)	3.15A/115VAC 1.5A/230VAC					
	INRUSH CURRENT (Typ.)	COLD START 20A/115VAC 40A/230VAC					
	LEAKAGE CURRENT	<1mA / 240VAC					

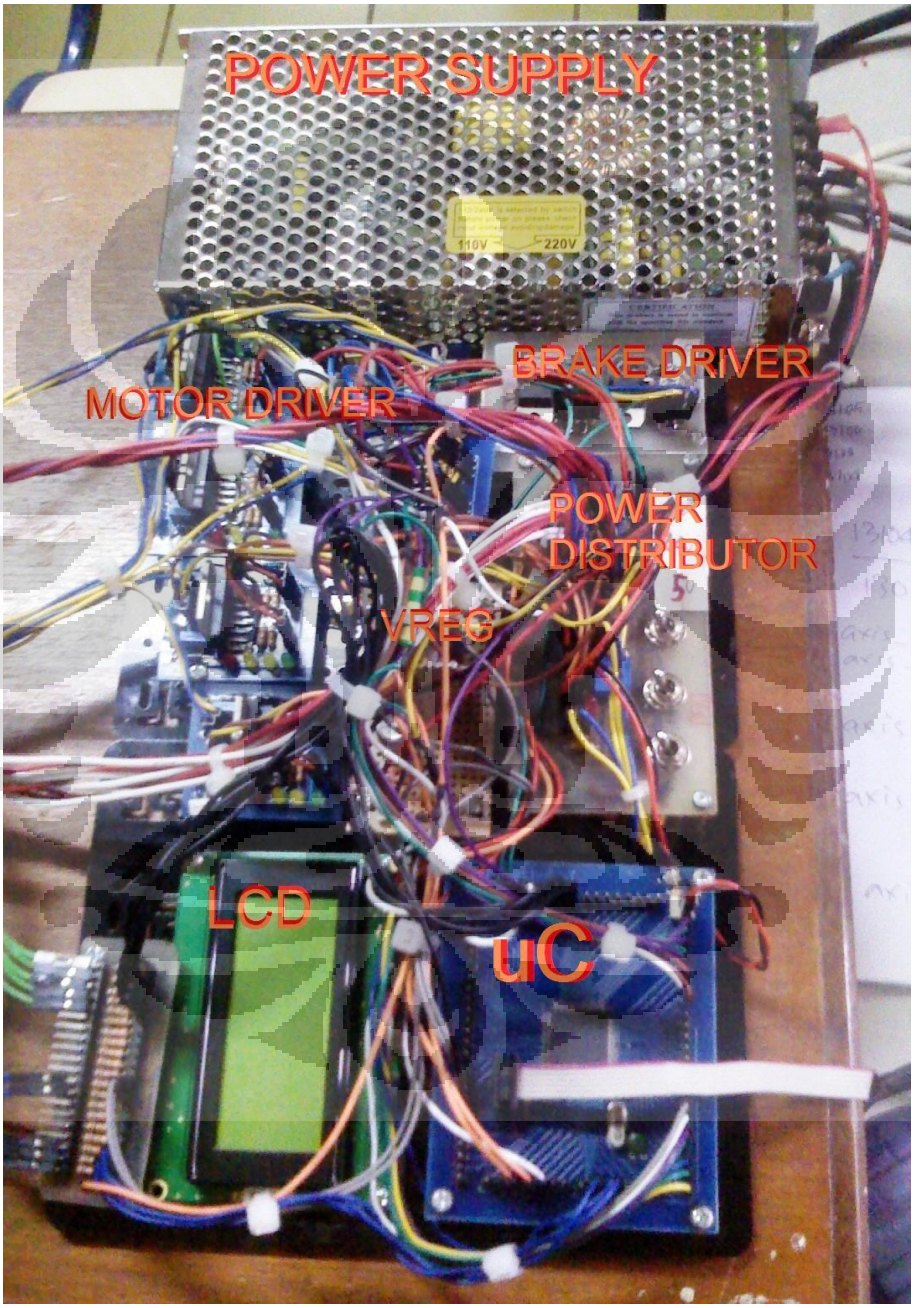
Gambar 4.33 Power Supply 12V dan 24V [15]

Untuk *power supply* digital dengan tegangan 5V digunakan *power supply* dengan arus 3A.

Gambar 4.34 Power Supply 5V

4.2.7 Control Unit

Seluruh perangkat elektronik diatas kemudian di rangkai sehingga menjadi sebuah perangkat keras *control unit* dari robot movemaster RV-M1. Berikut adalah gambar *control unit* tersebut :



Gambar 4.35 Perangkat Keras *Control Unit*

BAB 5
PENGEMBANGAN SISTEM PERANGKAT LUNAK

SISTEM PENGENDALI

Perangkat lunak pada sebuah sistem pengendali berfungsi untuk mengatur informasi antara *input*, proses dan *output* dari sistem yang dikendalikan. Microcontroller diprogram dengan algoritma yang disesuaikan dengan *software interface* ke PC. Program pada microcontroller harus mampung menerima dan mengirim data ke komputer dan harus mampu mengontrol perangkat keras lainnya yang terhubung ke robot. Program yang dibuat menggunakan *compiler* Codevision AVR ini mempunyai beberapa fungsi yang dibuat untuk mengatur microcontroller agar dapat menjalankan fungsi yang dibutuhkan. Berikut adalah algoritma yang dirancang untuk perangkat lunak sistem pengendali :

```
Algorithm :

    START
        1) Setting microcontroller.
        2) Menyalakan lampu indikator (robot siap menerima perintah).
        3) Menunggu perintah dari komputer server
        4) IF perintah dari server = nilai perintah default, lakukan
           pergerakan posisi default.
        5) IF perintah dari server = nilai perintah manual, lakukan
           pergerakan manual.
        6) IF perintah dari server = nilai perintah web, lakukan pergerakan
           berbasis web.
        7) IF perintah dari server = nilai berhenti, berhenti menerima
           perintah
    END
```

Untuk menjalankan sebuah perintah yang diberikan, *microcontroller* harus mengenali nilai perintah yang dikirim oleh komputer *server*. Nilai perintah ini merupakan sebuah nilai yang ditentukan untuk menjadi protokol komunikasi antara *microcontroller* dengan komputer *server*. Nilai perintah tersebut berupa bit yang dikirimkan secara serial oleh komputer *server*.

5.1 Fungsi Utama

Fungsi utama berfungsi untuk menerima perintah dari PC dan pengaturan awal microcontroller.

Terdapat 4 pilihan perintah yaitu :

1. Default position : value yang diterima dari PC = 255
2. Manual mode : value yang diterima dari PC = 254
3. *Web* mode : value yang diterima dari PC = 253
4. Reset mode : value yang diterima dari PC = 252

LED indikator akan menyala apabila microcontroller siap menerima perintah.

```
Algorithm :

    START
        8)  Setting microcontroller.
        9)  Setting nilai - nilai perintah.
        10) Menampilkan tulisan pada LCD Robot Initialize
        11) Tunggu 3 detik.
        12) Menampilkan tulisan pada LCD Ready to Execute
        13) Tunggu 3 detik.
        14) Mengosongkan LCD.
        15) Menyalakan lampu indikator (robot siap menerima perintah).
        16) Mulai loop menunggu perintah.
        17) IF nilai perintah = 255.
            a. menuju fungsi default_position.
            b. kirim verifikasi posisi default.
            c. matikan lampu indicator.
        18) IF nilai perintah = 254.
            a. masuk mode manual.
            b. matikan lampu indicator.
        19) IF nilai perintah = 253
            a. menuju mode web.
            b. matikan lampu indicator.
        20) IF nilai perintah = 252.
            a. keluar dari loop
            b. matikan lampu indicator.
    END
```

5.2 Pengaturan I/O PORT Microcontroller

Fungsi untuk mengatur fungsi setiap PORT, baik sebagai *input* atau *output*.

```
Algorithm :
    START
       1)  Setting PORT F
       2)  Setting PORT A
       3)  Setting PORT B
       4)  Setting PORT D
       5)  Setting PORT E
       6)  Setting PORT H
       7)  Setting PORT L
       8)
    END
```

5.3 Pengaturan LCD

LCD diatur untuk bekerja pada PORT C.

```
Algorithm :
    START
       1) Mulai penulisan in-line assembly language.
       2) Setting LCD pada PORT C.
       3) Tutup penulisan in-line assembly languange.
    END
```

5.4 Pengaturan Clock Speed

Oscillator yang dipakai adalah 16Mhz dengan faktor pembagi 1. Kecepatan microcontroller dapat diatur dengan menaikkan atau menurunkan faktor pembagi.

```
Algorithm :
    START
       1) Setting oscillator.
       2) Setting factor pembagi.
    END
```

5.5 Pengaturan Komunikasi Serial

Fungsi pengaturan parameter komunikasi serial, PORT komunikasi serial yang dipakai adalah serial 2.

```
Algorithm :
    START
       1) Setting tipe data serial communication, 8 Data, 1 Stop, No Parity
       2) Aktifasi transmitter receiver
       3) Setting mode serial asynchronous
       4) Setting baud rate 115200 BPS.
    END
```

5.6 Pengaturan Timer

Timer yang digunakan adalah *timer* 1 dan 5 dengan perhitungan frekuensi PWM :

fPWM = fOSC / (N*(1+1023))

fPWM = 16Mhz / (64*1024)

fPWM = 244.14 Hz

tPWM = 1 / 244.14

tPWM = 0.0041 s

tPWM = 4 ms

```
Algorithm :
    START
       1) Pengaturan sumber clock, sumber clock dari system clock.
       2) Pengaturan Frekuensi Timer menjadi 2000.000 kHz.
       3) Pengaturan mode PWM menjadi fast PWM.
       4) Menyalakan semua timer
       5) Menyalakan noise canceler.
    END
```

5.7 Pengaturan Interrupt

Semua *interrupt* pada microcontroller di-*set* dengan mode *raising edge*.

```
Algorithm:
    START
       1) Menyalakan INT0 dengan mode rising edge.
       2) Menyalakan INT1 dengan mode rising edge.
       3) Menyalakan INT2 dengan mode rising edge.
       4) Menyalakan INT3 dengan mode rising edge.
       5) Menyalakan INT4 dengan mode rising edge.
       6) Menyalakan INT5 dengan mode rising edge.
       7) Menyalakan INT6 dengan mode rising edge.
       8) Menyalakan INT7 dengan mode rising edge.
       9)
    END
```

5.8 Fungsi Pengontrolan Motor

Fungsi ini untuk mengatur frekuensi PWM dan memberikan sinyal *pada motor driver* sesuai dengan *truth table motor driver*.

```
Input : Parameter pergerakan axis.
Output: Sinyal truth table dan PWM

Algorithm :
    START
       1) Menunggu data yang masuk.
       2) Menentukan pemilihan truth table motor driver masing-masing axis
          berdasarkan arah_axis yang diperintahkan.
       3) Mengirim sinyal ke motor driver.
       4) Mengaktifkan PWM sesuai dengan perintah.
    END
```

5.9 Fungsi Pengiriman Text Secara Serial

Fungsi mengirim data *text* melalui serial port 2.

```
Input : Text dari user
Output: Data bit menuju PC
Algorithm :
    START
       1) Periksa registry serial apakah kosong.
       2) Apabila kosong, data = registry.
       3) Kirim semua text.
    END
```

5.10 Fungsi Pengiriman Nilai Secara Serial

Fungsi mengirim nilai melalui serial port 2.

```
Input : Nilai dari user
Output: Data bit menuju PC

Algorithm :
    START
       1) Periksa registry serial apakah kosong.
       2) Apabila kosong, data = registry.
       3) Kirim semua data.
    END
```

5.11 Fungsi Menampilkan Data Pada LCD

Fungsi-fungsi untuk menampilkan pembacaan pulsa *encoder* pada LCD.

```
Input : Data perhitungan encoder.
Output: Data ASCII pada LCD.

Algorithm :
START
   1) Menuju posisi LCD 0,0
   2) Konversi data int menjadi ASCII
   3) Tampilkan data di LCD
END
```

5.12 Menghentikan Pergerakan Robot

Fungsi-fungsi untuk memberhentikan masing – masing *axis*.

```
Algorithm :
START
   1) Mengatur signal ke motor driver menjadi stop sesuai truth table.
   2) Mematikan timer.
   3) Reset perhitungan encoder.
END
```

5.13 Menggerakkan Semua Axis Secara Bersamaan

Fungsi menggerakkan robot, berlaku untuk setiap *axis*. Fungsi ini akan menggerakkan setiap motor sampai pulsa yang dihitung *encoder* sesuai dengan target.

```
Input : Data pergerakan robot setiap axis.
Output: Sinyal truth table dan PWM setiap axis.

Algorithm :
START
   1) Menerima data pergerakan setiap axis.
   2) Melakukan konversi perhitungan encoder menjadi sudut real.

   Perhitungan pulsa encoder Pengukuran dilakukan dengan menampilkan pulsa
   encoder pada LCD, kemudian mengambil rata-rata yang kemudian dibagi
```

```
   dengan datapada datasheet Movemaster RV-M1 untuk mendapatkan nilai pulsa
   per 1 derajat pergerakan encoder pulses.

      axis_1      : 16932
                  : 16939
                  : 16940
                  : 16934
      avg         : 16936
      robot spec  : 300'
      ~ 57 = 1' actual

      axis_2      : 13105
                  : 13100
                  : 13103
                  : 13104
      avg         : 13103
      robot spec  : 130'
      ~ 101 = 1' actual

      axis_3      : 7368
                  : 7396
                  : 7351
                  : 7362
      avg         : 7370
      robot spec  : 110'
      ~ 67 = 1' actual

      axis_4      : 8862
                  : 8861
                  : 8875
                  : 8874
      avg         : 8868
      robot spec  : 180'
      ~ 50 = 1' actual

      axis_5      : 5228
                  : 5150
                  : 5330
                  : 5301
      avg         : 5252
      robot spec  : 180
      ~ 30 = 1' actual

   3) Menggerakkan semua axis secara bersamaan.
   4) Mulai loop pengecekan encoder.
   5) IF perhitungan encoder tercapai > motor berhenti.
   6) IF posisi semua axis tercapai > keluar dari loop.
END
```

5.14 Mode Posisi Default

Fungsi untuk memerintahkan robot untuk posisi *default*-nya.

```
Algorithm :
    START
       1) Buka brake axis 2.
       2) Gerakkan axis 2 menuju posisi default.
       3) Stop axis 2 apabila switch axis 2 menyala.
       4) Buka brake axis 3.
       5) Gerakkan axis 3 menuju posisi default.
```

```
        6)  Stop axis 3 apabila switch axis 3 menyala.
        7)  Gerakkan axis 1 menuju posisi default.
        8)  Stop axis 1 apabila switch axis 1 menyala.
        9)  Gerakkan axis 4 menuju posisi default.
        10) Stop axis 4 apabila switch axis 4 menyala.
        11) Gerakkan axis 5 menuju posisi default.
        12) Stop axis 5 apabila switch axis 5 menyala.
    END
```

5.15 Mode Manual

Fungsi ini merupakan fungsi untuk menampung data yang dikirim apabila robot ingin digerakkan secara manual. Setelah data pergerakan sudah lengkap robot langsung mengeksekusi pergerakan sesuai dengan data yang dikirim dari *server*.

```
Algorithm :
    START
        1)  Kirim text untuk meminta user memasukkan data.
        2)  Tunggu data dikirim.
        3)  Set data yang dikirim sebagai parameter pergerakan robot.
        4)  Point 2) dan 3) dilakukan terus sampai data lengkap
        5)  Eksekusi pergerakan robot.
    END
```

5.16 Mode Web

Fungsi ini merupakan fungsi untuk menampung data yang dikirim dari komputer *server* apabila robot ingin digerakkan secara *web based*. Setelah data pergerakan sudah lengkap robot langsung mengeksekusi pergerakan sesuai dengan data yang dikirim dari komputer *server*. Fungsi ini berbeda dengan fungsi manual karena tidak mengirimkan informasi kepada client dan bersifat otomatis.

```
Algorithm :
STATUS
    1)  Tunggu data dikirim.
    2)  Set data yang dikirim sebagai parameter pergerakan robot.
    3)  Point 1) dan 2) dilakukan terus sampai data lengkap.
    4)  IF semua nilai pergerakan adalah 1 maka robot melakukan perintah
        posisi default.
    5)  Eksekusi pergerakan robot.
END
```

BAB 6
PENGEMBANGAN SISTEM KOMUNIKASI DENGAN *WEB* SERVER

Sistem komunikasi dengan *web server* dibuat untuk mengambil data dari dihasilkan *web server* yang merupakan data perintah dari *client.* Sistem komunikasi dibuat dengan menggunakan *interface software* yang di-*install* pada komputer *server.* Program ini harus mampu mengambil data yang dihasilkan oleh *web server* dan mengirim data tersebut ke *microcontroller* untuk menggerakkan robot.

6.1 Program Interface pada Komputer Server

Program ini dibuat menggunakan aplikasi WIN16 dengan *compiler* Turbo C. Berikut adalah algoritma *interface software* yang dirancang untuk komunikasi komputer *server* dengan *microcontroller*.

```
Algorithm :
      START
          1) Menunggu perintah dari user
          2) IF Perintah dari user = perintah manual, lakukan aktifkan mode
             manual
          3) IF Perintah dari user = perintah web, lakukan aktifkan mode
             web.
          4) Membuka PORT komunikasi komputer server.
          5) Mengirim data ke microcontroller secara serial.
          6) Menutup PORT komunikasi.
      END
```

6.1.1 Fungsi Utama

Fungsi utama pada *interface software* ini ini berfungsi untuk memberitahukan *user* perintah yang ingin dilakukannya.

```
Algorithm :
      START
          7) Melakukan looping seluruh fungsi.
          8) Mengosongkan layar tampilan.
          9) Menampilkan text pilihan perintah.
          10) Switch, menunggu perintah oleh user
               i.   case 1 :
                 a. Membuka PORT komunikasi.
```

```
            b. Mengaktifasi mode manual
         ii.   case 2 :
            a. Membuka PORT komunikasi.
            b. Mengaktifasi mode web.
       iii.   case 3 : Keluar program.
    11) IF perintah salah program akan memberitahukan user kalau input
        salah.
END
```

6.1.2 Fungsi Membuka PORT Komunikasi

Fungsi membuka PORT komunikasi pada komputer . PORT komunikasi yang dipakai adalah COM1.

```
Algorithm :
    START
        1) Matikan interrupt COM1.
        2) Pengaturan baud rate, 115200BPS.
        3) Pengaturan tipe serial, 8 Bits, No Parity, 1 Stop Bit.
        4) Menyalakan DTR, RTS, and OUT2.
    END
```

6.1.3 Fungsi Mengambil Data Komunikasi Serial

Fungsi ini berfungsi untuk menerima data dari microcontroller dan menampilkan data tersebut pada layar monitor komputer *server*. Fungsi ini digunakan untuk konfirmasi microcontroller yang telah atau sedang menjalankan perintah.

```
Algorithm :
    START
        1) Reset counter dan mendefinikan array buffer.
        2) Melakukan loop pengecekan input.
        3) IF ada input tampilkan data.
        4) If keyboard ditekan keluar loop.
    END
```

6.1.4 Fungsi Pemilihan Pergerakan *Manual*

Robot movemaster RV-M1 juga bisa digerakkan secara *manual* melalui fungsi ini. Misalkan untuk melakukan gerakan *default* robot melalui komputer *server*.

```
Algorithm :
    START
        1) Mengosongkan tampilan layar.
        2) Menampilkan tulisan informasi pilihan.
        3) Menunggu input dari user untuk pilihan perintah.
        4) Switch (perintah user)
            i.   case 1 : Mengirim perintah posisi default.
           ii.   case 2 : Menuju perintah mode manual.
        5) IF perintah salah program akan memberitahu user kalau input
           salah.
    END
```

6.1.5 Fungsi Pemilihan Pergerakan *Web*

Untuk mengaktifkan pengendalian berbasis *web* diperlukan adanya perintah dari interface software ini karena pengecekan data akan berlangsung terus menerus setelah mode *web* diaktifkan. Fungsi dapat menghentikan dan memulai pengencekan data tersebut.

```
Algorithm :
       START
          1) Mengosongkan tampilan layar.
          2) Menampilkan tulisan pemilihan mode web
          3) Menunggu perintah user.
          4) Switch (perintah)
              i.    case 1 : Mengaktifkan mode pengecekan file
             ii.    case 2 : Mengirim data reset ke microcontroller
          5) IF perintah salah program akan memberitahu user kalau input
             salah
       END
```

6.1.6 Perintah Posisi Default

Fungsi mengirim data untuk perintah posisi default. *Microcontroller* akan melakukan perintah posisi default apabila mendapat bit 255. Setelah posisi *default* dicapai *microcontroller* akan mengirim informasi konfirmasi.

```
Algorithm :
       START
          1) Mengosongkan tampilan layar.
          2) Mengirim bit 255 ke microcontroller.
          3) Menunggu data verifikasi dari microcontroller.
       END
```

6.1.7 Perintah Pergerakan *Manual*

Fungsi mengirim data pergerakan robot secara *manual*. Setiap *axis* memerlukan 3 data yaitu : kecepatan, arah, dan sudut.

```
Algorithm
       START
          1) Mengosongkan tampilan layar.
          2) Menampilkan tulisan untuk meminta data dari user
          3) Menerima data dari user.
          4) Mengirim data ke microcontroller.
       END
```

6.1.8 Fungsi Pengecekan Data Pergerakan *Web*

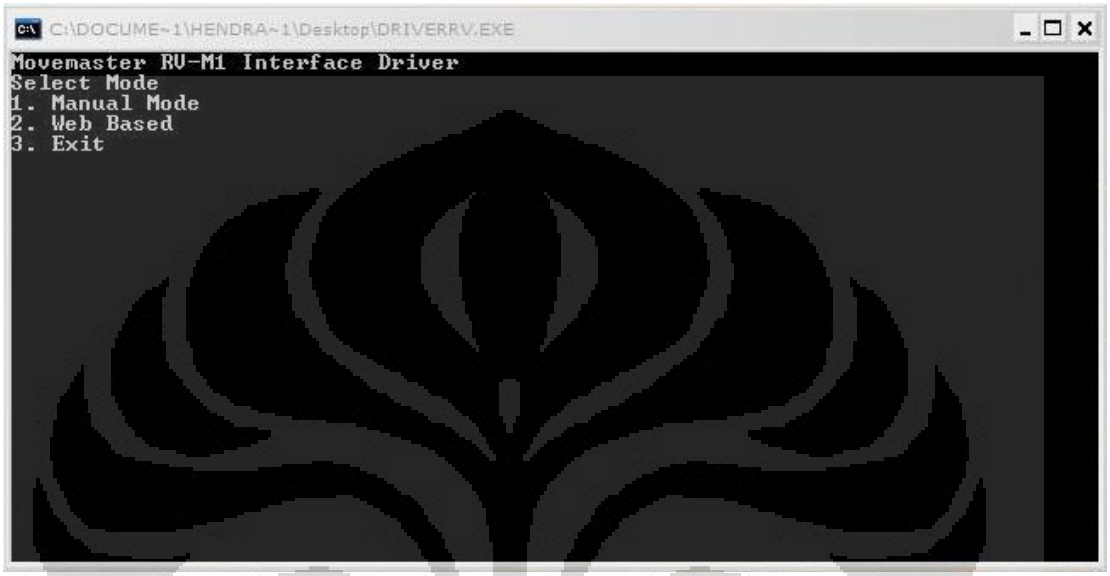
Fungsi ini akan selalu mengecek keberadaan file kdt.txt yang dihasilkan oleh *web server*.

```
Algorithm :
       START
          1) Memulai loop pengecekan file.
          2) Mengosongkan tampilan layar.
          3) Membuka text file pada path D:/adtj/www/htdocs/i-RoMan/kdt.txt
          4) IF keyboard ditekan
             a. Data yang akan dikirim bernilai 0
             b. Mengirim data ke microcontroller
             c. Tutup file text
             d. Hapus file text.
             e. Keluar fungsi
          5) IF file ditemukan
             a. Tampilkan informasi file ditemukan.
             b. Membaca isi file.
             c. Menyimpan isi file dalam array pergerakan, mengirim data
                pergerakan.
             d. Mengirim data aktifasi mode web = 255.
             e. Menampilkan data yang dikirim
             f. Menutup file text
             g. Menghapus file text.
          6) IF file tidak ditemukan tampilkan informasi file tidak
             ditemukan.
          7) Tunggu 1 Detik
          8) Kembali ke awal fungsi
       END
```

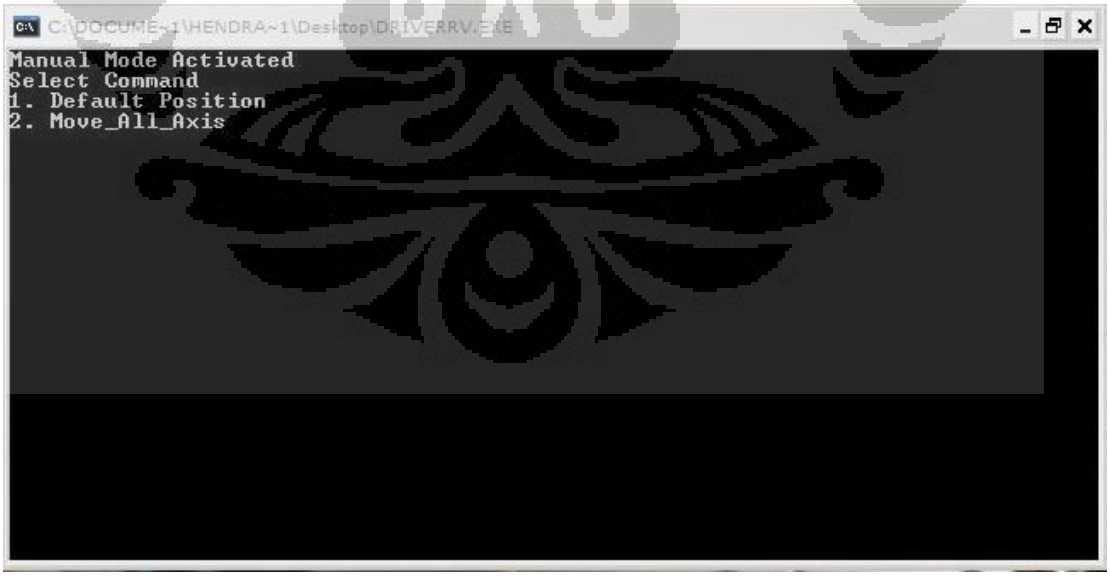
Software ini harus dijalankan sebelum mengirim perintah dari *web*, software ini yang akan mengolah data yang disimpan *web* dalam text file. Program dapat dijalankan dengan mengeksekusi file bernama DRIVERRV.exe yang merupakan *executable file* program ini. Begitu dijalankan maka program akan meminta beberapa *input* dari user untuk memilih satu dari beberapa metode pergerakan yang ada. Untuk metode *emergency* tidak diaktifasi melalui software ini, namun software ini bersifat sebagai penerima data emergency yang dihasilkan oleh *web* server yang kemudian disimpan dalam file text yang sama. Isi data *emergency* sebenarnya adalah data pergerakan namun setiap parameter bernilai 0 sehingga robot akan berhenti begitu mendapat data seperti itu. Software ini juga dilengkapi *user interface* yang sederhana yang berfungsi untuk membantu user mengetahui kondisi program. Selain itu user

juga dapat menghentikan program dengan memasukkan *input* melalui *keyboard*. Program ini akan menutup secara otomatis disaat mendapat perintah *exit*.

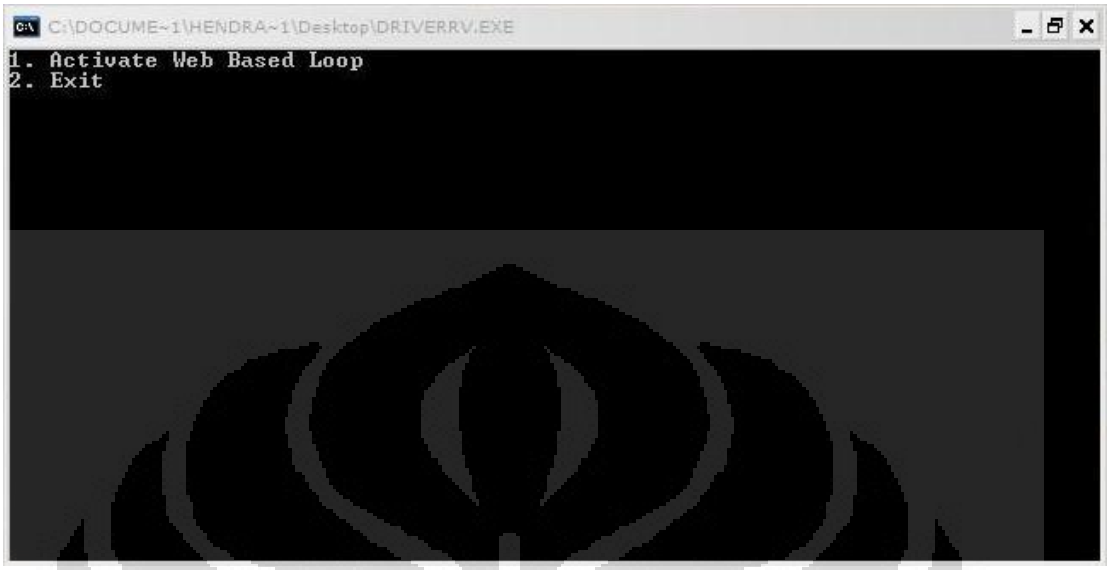
Berikut adalah tampilan *software* pada saat dieksekusi :



Gambar 6.1 Tampilan Awal



Gambar 6.2 Tampilan Mode Manual

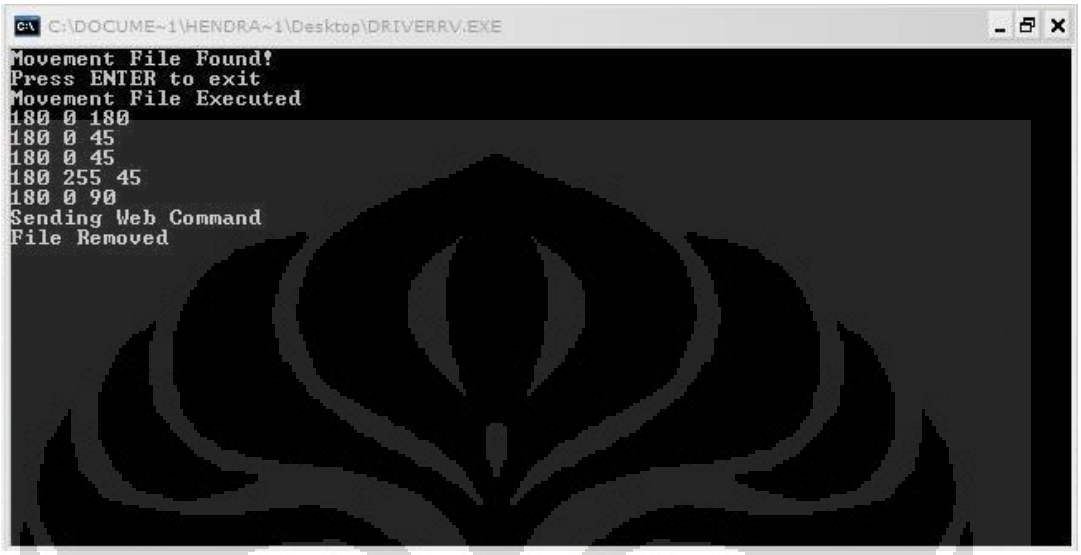


Gambar 6.3 Tampilan Mode *Web*

Apabila *text file* tidak ditemukan maka *software* akan memberitahukan informasi tersebut, *software* akan terus memeriksa keberadaan *text file* dalam interval 1 detik.

Gambar 6.4 Tampilan Mode *Web* Text File Tidak Ditemukan

Apabila *text file* ditemukan, maka software akan langsung mengambil data dan mengirimkannya secara *serial* ke *microcontroller*, menampilkan data yang dibaca dan menghapus *text file* tersebut.

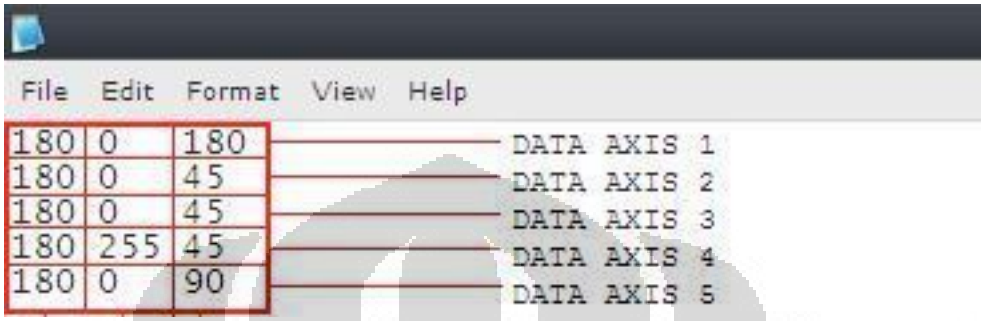


Gambar 6.5 Tampilan Mode *Web* Text File Ditemukan

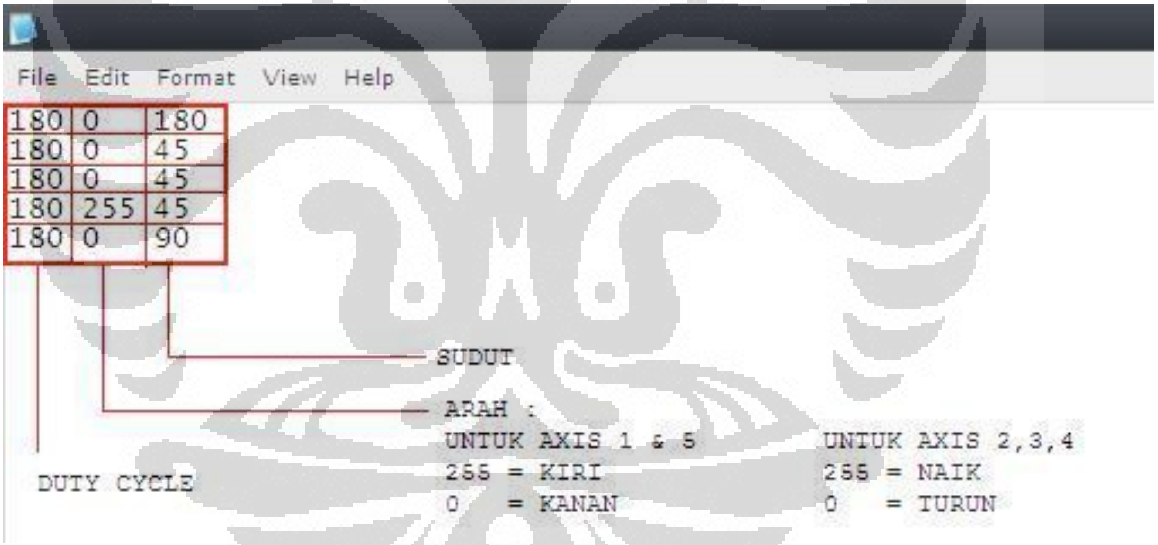
Data yang terdapat dalam *file text* merupakan range angka dari 0 sampai 255, angka 0 sampai 255 dapat dikirim untuk kemudian diolah oleh *microcontroller*. Sehingga didalam *file* kdt.txt terdapat *format* data yang dihasilkan oleh yaitu sebagai berikut :

Gambar 6.6 Format Data Didalam File kdt.txt

Kesepakatan data ini dilakukan untuk mempermudah pembacaan data pada *text file*. Setiap data mempunyai fungsinya masing-masing yang akan dijelaskan oleh gambar berikut :



Gambar 6.7 Paket Data Didalam File kdt.txt

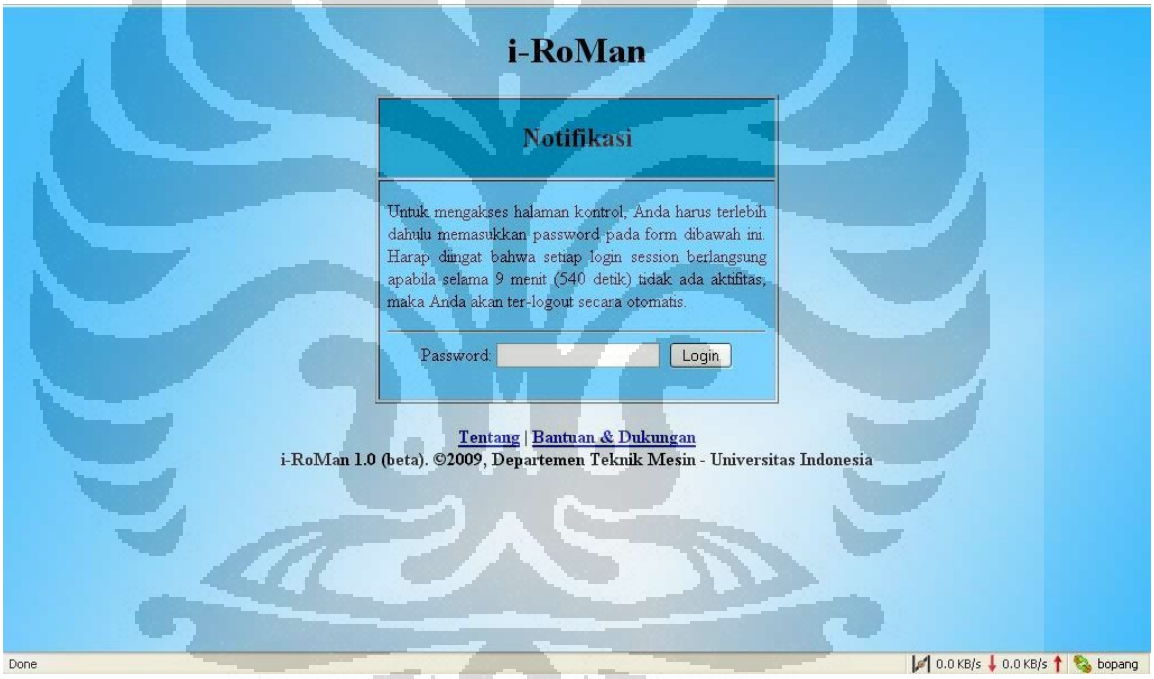


Gambar 6.8 Fungsi Masing-Masing Data Didalam File kdt.txt

Setiap *axis* membutuhkan 3 data tersebut, sehingga apabila dijumlahkan maka terdapat 15 data pada *file* kdt.txt tersebut. Semua nilai ini akan diubah menjadi 0 apabila terdapat kondisi *emergency* sehingga dengan seketika menghentikan gerakan dari robot. Kondisi *emergency* sendiri akan muncul apabila *client* menekan tombol *emergency* pada *web* begitu *client* menganggap kondisi robot perlu dihentikan.

6.2 Web Server

Pada bagian ini tidak dibahas sepenuhnya mengenai *web server* dan *user interface* pada *web*. Hal ini dikarenakan pengembangan *web server* dijelaskan pada buku tersendiri dari peneliti lain yang tergabung dalam group pengembangan kendali robot berbasis *web* ini. Sehingga untuk pembahasan lebih lanjut mengenai *web server* pada pengembangan sistem kontrol robot berbasis *web* ini harus mengacu pada buku tersebut. Untuk memberikan gambaran mengenai *web server* tersebut maka akan disertakan sertakan beberapa gambar *user interface* yang ada pada *web server* tersebut.



Gambar 6.9 *User Login* pada *Web*

Gambar diatas merupakan halaman notifikasi login client, halaman ini diberi *password* untuk menjaga keamanan dan client akan otomatis ter-*logout* apabila selama 9 menit tidak ada aktifitas.

Gambar 6.10 Halaman Kontrol Utama Pada *Web*

Gambar diatas adalah bentuk *user interface* yang ditampilkan dilayar *browser*. Terdapat 2 metode pergerakan robot yaitu *cursor based inverse kinematics* dan *forward kinematic*. Terdapat 2 kamera *webcam* yang memonitor pergerakan robot secara *real-time* untuk membantu *user* yang berada jauh dari robot. User dapat mengaktifasi mode *emergency* apabila terdapat kondisi yang membahayakan atau dapat merusak robot. Dari layar *web* inilah kemudian diproses keluaran berupa file .txt pada komputer *server*, file tersebut yang akan diproses oleh *software interface* pada komputer server dan kemudian diteruskan ke kontrol unit untuk menggerakkan robot. Maka lengkaplah hubungan dari *client* sampai ke robot didalam penelitian kontrol robot berbasis *web* ini.

BAB 7
PENGUJIAN SISTEM

7.1 *Pengujian Pergerakan Axis 1*

Pengujian pergerakan *axis* 1 dapat dilihat pada gambar berikut :

Gambar 7.1 Pengujian *Axis* 1

7.2 *Pengujian Pergerakan Axis 2*

Pengujian pergerakan *axis* 2 dapat dilihat pada gambar berikut :

Gambar 7.2 Pengujian *Axis* 2

7.3 *Pengujian Pergerakan Axis 3*

Pengujian pergerakan *axis* 3 dapat dilihat pada gambar berikut :

Gambar 7.3 Pengujian *Axis* 3

7.4 *Pengujian Pergerakan Axis 4*

Pengujian pergerakan *axis* 4 dapat dilihat pada gambar berikut :

Gambar 7.4 Pengujian *Axis* 4

7.5 *Pengujian Pergerakan Axis 5*

Pengujian pergerakan *axis* 5 dapat dilihat pada gambar berikut :

Gambar 7.5 Pengujian *Axis* 5

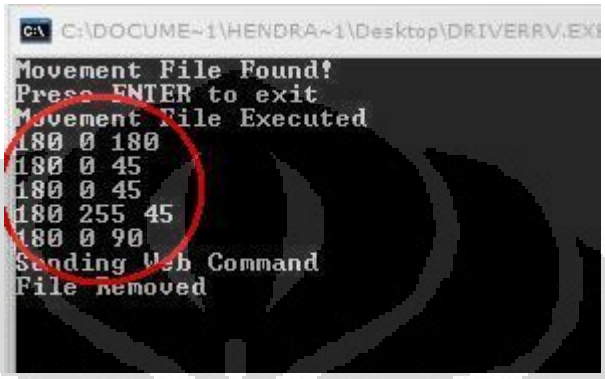
7.6 *Pengujian Pergerakan Simultan*

Pengujian pergerakan simultan *axis* 1 sampai *axis* 5 dapat dilihat pada gambar berikut :

Gambar 7.6 Pengujian Gerak Simultan

7.7 *Akurasi Data*

Data yang dikirim antara *web server* dan *microcontroller* dapat dimonitor dari layar tampilan software. Berikut adalah contoh data yang dibaca software dan dikirim ke microcontroller.



Gambar 7.7 Proses Verifikasi Data

Setelah uji 1000 kali proses pembacaan dan pengiriman data, tidak terjadi kesalahan maupun *error* pada *software*. Arah pergerakan robot dan perhitungan *encoder* sudah tepat. Namun akurasi pergerakan robot masih rendah dikarenakan tidak adanya algoritma kendali adaptif.

Untuk pengujian akurasi data dari microcontroller ke robot dilakukan dengan menampilkan data perhitungan encoder pada LCD setelah robot bergerak. Berikut adalah data yang berhasil diambil dari robot :

1. Axis 1 : memiliki kalibrasi sudut 1° = 57 pulsa encoder. Dengan pergerakan robot -180°, data encoder yang didapat sebesar 10374 pulsa. Sehingga didapat error pergerakan sebesar ± 2°.
2. Axis 2 : memiliki kalibrasi sudut 1° = 101 pulsa encoder. Dengan pergerakan robot -45°, data encoder yang didapat sebesar 5050 pulsa. Sehingga didapat error pergerakan sebesar ± 5°.

3. Axis 3 : memiliki kalibrasi sudut 1° = 67 pulsa encoder. Dengan pergerakan robot -45°, data encoder yang didapat sebesar 3216 pulsa. Sehingga didapat error pergerakan sebesar ± 3°.
4. Axis 4 : memiliki kalibrasi sudut 1° = 50 pulsa encoder. Dengan pergerakan robot 45°, data encoder yang didapat sebesar 2350 pulsa. Sehingga didapat error pergerakan sebesar ± 2°.
5. Axis 5 : memiliki kalibrasi sudut 1° = 30 pulsa encoder. Dengan pergerakan robot 90°, data encoder yang didapat sebesar 2730 pulsa. Sehingga didapat error pergerakan sebesar ± 1°.

Adanya error ini disebabkan karena robot dipengaruhi oleh gaya-gaya yang didapat oleh robot sepeti inersia pada sistem mekanikal robot. Error ini dapat dikurangi dengan menambahkan algoritma kendali kecepatan motor yang adaptif.

BAB 8
KESIMPULAN & SARAN PENELITIAN LEBIH LANJUT

8.1 *Kesimpulan*

1. Pengendalian robot berbasis *web* sangat bergantung pada akurasi data yang diterima dan dikirim antara masing – masing sistem.
2. Pengendalian robot secara digital memungkinkan berbagai macam algoritma kendali untuk dikembangkan dalam pengendalian robot.
3. Sistem kendali robot berbasis *web* dapat diterapkan pada aplikasi industri yang membutuhkan pemantauan pergerakan robot.
4. Sistem *emergency* dalam pengendalian robot berbasis *web* harus dikembangkan secara optimal untuk menjamin keamanan sistem elektronik dan mekanikal.

8.2 *Saran Penelitian Lebih Lanjut*

Untuk pengembangan robot lebih lanjut disarankan untuk melakukan beberapa perubahan untuk meningkatkan kinerja sistem, yaitu :

1. Motor driver L298 dapat ditingkatkan kemampuannya dengan memberikan heatsink atau dengan menggantinya dengan driver motor yang memiliki daya tahan lebih.
2. Meningkatkan kecepatan proses data dengan memakai microcontroller dengan kecepatan lebih tinggi.
3. Control unit dapat dibuat menjadi *single board* saja untuk mengurangi pemakaian kabel sehingga dapat mengurangi resiko kesalahan teknis akibat putusnya kabel.
4. Pemakaian pulsa *encoder* fasa B & Z akan meningkatkan akurasi robot.

DAFTAR REFERENSI

[1] APJII(December 2007). Perkembangan Jumlah Pelanggan & Pemakai Internet (kumalatif). 19 February 2009. http://www.apjii.or.id

[2] ISO 8373(1994). Manipulating Industrial Robot. 20 July 2009. http://www.dira.dk/Portals/0/Robotter/robotdef.pdf

[3] Mitshubishi. (1998). Movemaster RV-M1 Instruction Manual. Japan

[4] DigiKey. (2003). ATmega2560 Microcontroller Package. April 27, 2009. www.digikey.com

[5] Society of Robots. (November 30, 2008). Pulse Width Modulation Tutorial. June 2009. http://www.societyofrobots.com/member_tutorials/node/229

[6] Quantum Devices. (March 27, 2009). What is meant by Rotary Incremental Encoder Index Pulse "gating"?. July 10, 2009. http://quantumdevices.wordpress.com/2009/03/27/what-is-meant-by-rotray-incremental-encoder-gating

[7] Wankhede, Mahesh. (2007). Switch On I/O Ports. August 20, 2009. http://www.freewebs.com/maheshwankhede/ports.html

[8] ATMEL. ATMega2560 Microcontroller PDF. April 3, 2009. http://www.atmel.com/dyn/products/product_card.asp?family_id=607&family_name=AVR+8%2DBit+RISC+&part_id=2014

[9] Ucables. (2006). 16mhz Smd Crystal Oscillator. December 21, 2009. http://ucables.com/ref/16MHZ-SMD-CRYSTAL-OSC-R46887

[10] ST. L298D Datasheet. August 23, 2009. http://www.datasheetcatalog.com/datasheets_pdf/T/I/P/1/TIP122.shtml

[11] Sains, Insan (2008). H-Bridge Driver : Kontrol Arah Motor. January 22, 2009. http://insansainsprojects.wordpress.com/2008/06/05/h-bridge-driver-kontrol-arah-motor

[12] ST. TIP122 Datasheet. June 8, 2009. http://www.datasheetcatalog.com/datasheets_pdf/T/I/P/1/TIP122.shtml

[13] Lumax Incorporated. LCM-S01604DSR Datasheet, May 19, 2009. http://www.lumex.com/products.aspx?id=11

[14] Pololu. USB to Serial Adaptor. March 27, 2009.
http://www.pololu.com/catalog/product/391

[15] Meanwell. S-100F PSU Datasheet. February 20, 2009.
http://www.meanwell.com/search/s-100f/default.htm

[16] Dietel & Dietel. (2003). How to Program in C. 3rd edition. New Jersey: Prentice Hall.

[17] Codevision. Codevision AVR C compilier. January 10, 2009.
http://www.hpinfotech.ro/html/cvavr.htm

LAMPIRAN 1

```
/*****************************************
Program Control Robot Manipulator 5 DOF
Author              : Hendra Prima Syahputra
Chip type           : ATmega2560
Program type        : Application
Clock frequency     : 16.000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 2048
*****************************************/

#include <mega2560.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <lcd.h>

#define MAX 0xFF
#define kanan 1
#define kiri 0
#define stop 2
#define naik 1
#define turun 0
#define buka 1
#define tutup 0

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

int encoder_axis_1, encoder_axis_2, encoder_axis_3, encoder_axis_4,
encoder_axis_5;

char lcd_encoder_axis_1[5],
     lcd_encoder_axis_2[5],
     lcd_encoder_axis_3[5],
     lcd_encoder_axis_4[5],
     lcd_encoder_axis_5[5];

int perintah;

int duty_cycle_axis_1 = 0;
int arah_axis_1 = 0;
int angle_axis_1 = 0;
int duty_cycle_axis_2 = 0;
int arah_axis_2 = 0;
int angle_axis_2 = 0;
```

```
int duty_cycle_axis_3 = 0;
int arah_axis_3 = 0;
int angle_axis_3 = 0;
int duty_cycle_axis_4 = 0;
int arah_axis_4 = 0;
int angle_axis_4 = 0;
int duty_cycle_axis_5 = 0;
int arah_axis_5 = 0;
int angle_axis_5 = 0;

/*
Kirim Text
*/
void kirim_char_serial_2(char flash *fmtstr,...)
{
    while ((*fmtstr != '\0'))
    {
      while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
      UDR2 = *fmtstr;
      fmtstr++;
    }
}

/*
Tq to Erwin > Kirim Nilai
*/
void kirim_nilai_serial_2(char *str)
{
    while(*str != '\0')
    {
        while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
        UDR2=*str;
        str = str + 1;
    }
}

/*
    %d - decimal
    %u - unsigned decimal
    %o - octal
    %x - hex
    %c - character
    %s - strings
*/

void lcd_setting()
{
    /*
    LCD on port C
    */

    #asm
    .equ __lcd_port=0x08 ;PORTC
    #endasm
}
```

```
void oscillator_setting()
{
    /*
    Crystal Division Factor : 1
    */

    #pragma optsize-
    CLKPR=0x80;
    CLKPR=0x00;
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif
}

void usart_2_setting()
{
    /*
    USART2 initialization
    Communication Parameters: 8 Data, 1 Stop, No Parity
    USART2 Receiver: On
    USART2 Transmitter: On
    USART2 Mode: Asynchronous
    USART2 Baud Rate: 115200
    */

    UCSR2A=0x00;
    UCSR2B=0x18;
    UCSR2C=0x06;
    UBRR2H=0x00;
    UBRR2L=0x08;
}

void ports_setting()
{
     /*
    board configuration

        F0F7K0K7
    E0              A0
    E7              A7
    H0              J7
    H7     uC       J0
    B4              C7
    B7              C0
    G0
    G5
        L0L7D0D7
    */
    PORTF =0b00000000;
    DDRF  =0b00001111;
    //        ||||||||
    //        |||||||\___0: brake_axis_2
    //        ||||||\____1: brake_axis_3
    //        |||||\_____2: input_1_axis_5
    //        ||||\______3: input_2_axis_5
    //        |||\_______4: limit_switch_1
    //        ||\________5: limit_switch_2
    //        |\_________6: limit_switch_3
    //        \__________7: limit_switch_4
```

```
//DDRK     = 0xFF;
//PORTK    = 0x00;

PORTA= 0b00000000;
DDRA = 0b11111111;
//        ||||||||
//        |||||||\___0: input_1_axis_1
//        ||||||\____1: input_2_axis_1
//        |||||\_____2: input_1_axis_2
//        ||||\______3: input_2_axis_2
//        |||\_______4: input_1_axis_3
//        ||\________5: input_2_axis_3
//        |\_________6: input_1_axis_4
//        \__________7: input_2_axis_4

PORTB= 0b00000000;
DDRB = 0b11100000;
//        ||||||||
//        |||||||\___0:
//        ||||||\____1:
//        |||||\_____2:
//        ||||\______3:
//        |||\_______4: limit_switch_5
//        ||\________5: PWM_axis_4
//        |\_________6: PWM_axis_5
//        \__________7: PWM_gripper

//PORTC
//DDRC = 0b11111111;
//        ||||||||
//        |||||||\___0: LCD_D0
//        ||||||\____1: LCD_D1
//        |||||\_____2: LCD_D2
//        ||||\______3:
//        |||\_______4: LCD_D4
//        ||\________5: LCD_D5
//        |\_________6: LCD_D6
//        \__________7: LCD_D7

PORTD= 0b00000000;
DDRD = 0b00110001;
//        ||||||||
//        |||||||\___0: input_1_gripper
//        ||||||\____1: encoder_axis_1
//        |||||\_____2: encoder_axis_2
//        ||||\______3: encoder_axis_3
//        |||\_______4: input_2_gripper
//        ||\________5: LED
//        |\_________6:
//        \__________7:
```

```
    PORTE= 0b11111111;
    DDRE  =0b00000000;
    //        ||||||||
    //        |||||||\___0:
    //        ||||||\____1:
    //        |||||\_____2:
    //        ||||\______3:
    //        |||\_______4: encoder_axis_4
    //        ||\________5: encoder_axis_5
    //        |\_________6:
    //        \__________7:

    //PORTG= 0b11111111;
    //DDRG = 0b00000000;
    //        ||||||||
    //        |||||||\___0:
    //        ||||||\____1:
    //        |||||\_____2:
    //        ||||\______3:
    //        |||\_______4:
    //        ||\________5:
    //        |\_________6:
    //        \__________7:

    //PORTH= 0b00000000;
    DDRH = 0b00000000;
    //        ||||||||
    //        |||||||\___0: RX_usart_2
    //        ||||||\____1: TX_usart_2
    //        |||||\_____2:
    //        ||||\______3:
    //        |||\_______4:
    //        ||\________5:
    //        |\_________6:
    //        \__________7:

    //PORTJ
    //DDRJ = 0b11111111;
    //        ||||||||
    //        |||||||\___0:
    //        ||||||\____1:
    //        |||||\_____2:
    //        ||||\______3:
    //        |||\_______4:
    //        ||\________5:
    //        |\_________6:
    //        \__________7:

    //PORTL= 0b11111111;
    DDRL = 0b11111111;
    //        ||||||||
    //        |||||||\___0:
    //        ||||||\____1:
    //        |||||\_____2:
    //        ||||\______3: PWM_axis_1
    //        |||\_______4: PWM_axis_2
    //        ||\________5: PWM_axis_3
    //        |\_________6:
    //        \__________7:
}
```

```
void timers_setting()
{
    /*
    Timer/Counter 1 initialization
    Clock source: System Clock
    Clock value: 2000.000 kHz
    Mode: Fast PWM top=00FFh
    OC1A output: Non-Inv.
    OC1B output: Non-Inv.
    OC1C output: Non-Inv.
    Noise Canceler: On
    Input Capture on Falling Edge
    Timer 1 Overflow Interrupt: Off
    Input Capture Interrupt: Off
    Compare A Match Interrupt: Off
    Compare B Match Interrupt: Off
    Compare C Match Interrupt: Off
    */
    TCCR1A=0xA9;
    TCCR1B=0x8A;
    TCNT1H=0x00;
    TCNT1L=0x00;
    ICR1H=0x00;
    ICR1L=0x00;
    OCR1AH=0x00;
    OCR1AL=0x00;
    OCR1BH=0x00;
    OCR1BL=0x00;
    OCR1CH=0x00;
    OCR1CL=0x00;
    /*
    Timer/Counter 5 initialization
    Clock source: System Clock
    Clock value: 2000.000 kHz
    Mode: Fast PWM top=00FFh
    OC5A output: Non-Inv.
    OC5B output: Non-Inv.
    OC5C output: Non-Inv.
    Noise Canceler: On
    Input Capture on Falling Edge
    Timer 5 Overflow Interrupt: Off
    Input Capture Interrupt: Off
    Compare A Match Interrupt: Off
    Compare B Match Interrupt: Off
    Compare C Match Interrupt: Off
    */
    TCCR5A=0xA9;
    TCCR5B=0x8A;
    TCNT5H=0x00;
    TCNT5L=0x00;
    ICR5H=0x00;
    ICR5L=0x00;
    OCR5AH=0x00;
    OCR5AL=0x00;
    OCR5BH=0x00;
    OCR5BL=0x00;
    OCR5CH=0x00;
    OCR5CL=0x00;
}
```

```
void interrupt_setting()
{
    /*
    External Interrupt(s) initialization
    INT0: On
    INT0 Mode: Rising Edge
    INT1: On
    INT1 Mode: Rising Edge
    INT2: On
    INT2 Mode: Rising Edge
    INT3: On
    INT3 Mode: Rising Edge
    INT4: On
    INT4 Mode: Rising Edge
    INT5: On
    INT5 Mode: Rising Edge
    INT6: On
    INT6 Mode: Rising Edge
    INT7: On
    INT7 Mode: Rising Edge
    */
    EICRA=0xFF;
    EICRB=0xFF;
    EIMSK=0xFF;
    EIFR=0xFF;
}

void brake_axis_2_off()
{
    PORTF.0 = 1;
}

void brake_axis_3_off()
{
    PORTF.1 = 1;
}

void brake_axis_2_on()
{
    PORTF.0 = 0;
}

void brake_axis_3_on()
{
    PORTF.1 = 0;
}


void motor_control
(
    float duty_cycle_axis_1, int arah_axis_1,
    float duty_cycle_axis_2, int arah_axis_2,
    float duty_cycle_axis_3, int arah_axis_3,
    float duty_cycle_axis_4, int arah_axis_4,
    float duty_cycle_axis_5, int arah_axis_5
)
{
    int input_1_axis_1, input_2_axis_1;
    int input_1_axis_2, input_2_axis_2;
```

```
int input_1_axis_3, input_2_axis_3;
int input_1_axis_4, input_2_axis_4;
int input_1_axis_5, input_2_axis_5;

//axis_1
if (arah_axis_1 == kanan)
{
    input_1_axis_1 = 1;
    input_2_axis_1 = 0;
}

if (arah_axis_1 == kiri)
{
    input_1_axis_1 = 0;
    input_2_axis_1 = 1;
}
if (arah_axis_1 == stop)
{
    input_1_axis_1 = 0;
    input_2_axis_1 = 0;
}

//axis_2
if (arah_axis_2 == naik)
{
    input_1_axis_2 = 1;
    input_2_axis_2 = 0;
}

if (arah_axis_2 == turun)
{
    input_1_axis_2 = 0;
    input_2_axis_2 = 1;
}

if (arah_axis_2 == stop)
{
    input_1_axis_2 = 0;
    input_2_axis_2 = 0;
}

//axis_3
if (arah_axis_3 == naik)
{
    input_1_axis_3 = 1;
    input_2_axis_3 = 0;
}

if (arah_axis_3 == turun)
{
    input_1_axis_3 = 0;
    input_2_axis_3 = 1;
}
if (arah_axis_3 == stop)
{
    input_1_axis_3 = 0;
    input_2_axis_3 = 0;
}
```

```
    //axis_4
    if (arah_axis_4 == naik)
    {
        input_1_axis_4 = 1;
        input_2_axis_4 = 0;
    }

    if (arah_axis_4 == turun)
    {
        input_1_axis_4 = 0;
        input_2_axis_4 = 1;
    }

    if (arah_axis_4 == stop)
    {
        input_1_axis_4 = 0;
        input_2_axis_4 = 0;
    }

    //axis_5
    if (arah_axis_5 == kanan)
    {
        input_1_axis_5 = 1;
        input_2_axis_5 = 0;
    }

    if (arah_axis_5 == kiri)
    {
        input_1_axis_5 = 0;
        input_2_axis_5 = 1;
    }
    if (arah_axis_5 == stop)
    {
        input_1_axis_5 = 0;
        input_2_axis_5 = 0;
    }

    PORTA.0 = input_1_axis_1;
    PORTA.1 = input_2_axis_1;
    PORTA.2 = input_1_axis_2;
    PORTA.3 = input_2_axis_2;
    PORTA.4 = input_1_axis_3;
    PORTA.5 = input_2_axis_3;
    PORTA.6 = input_1_axis_4;
    PORTA.7 = input_2_axis_4;
    PORTF.2 = input_1_axis_5;
    PORTF.3 = input_2_axis_5;

    OCR5AH = MAX*duty_cycle_axis_1;
    OCR5AL = MAX*duty_cycle_axis_1;
    OCR5BH = MAX*duty_cycle_axis_2;
    OCR5BL = MAX*duty_cycle_axis_2;
    OCR5CH = MAX*duty_cycle_axis_3;
    OCR5CL = MAX*duty_cycle_axis_3;
    OCR1AH = MAX*duty_cycle_axis_4;
    OCR1AL = MAX*duty_cycle_axis_4;
    OCR1BH = MAX*duty_cycle_axis_5;
    OCR1BL = MAX*duty_cycle_axis_5;
}
```

```
interrupt [INT1] void ext_int1_isr(void)
{
    encoder_axis_1++;
}

interrupt [INT2] void ext_int2_isr(void)
{
    encoder_axis_2++;
}

interrupt [INT3] void ext_int3_isr(void)
{
    encoder_axis_3++;
}



interrupt [INT4] void ext_int4_isr(void)
{
    encoder_axis_4++;
}

interrupt [INT5] void ext_int5_isr(void)
{
    encoder_axis_5++;
}

void display_encoder_axis_1()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_1,lcd_encoder_axis_1);
    lcd_puts(lcd_encoder_axis_1);
}

void display_encoder_axis_2()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_2,lcd_encoder_axis_2);
    lcd_puts(lcd_encoder_axis_2);
}

void display_encoder_axis_3()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_3,lcd_encoder_axis_3);
    lcd_puts(lcd_encoder_axis_3);
}

void display_encoder_axis_4()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_4,lcd_encoder_axis_4);
    lcd_puts(lcd_encoder_axis_4);
}
void display_encoder_axis_5()
{
    lcd_gotoxy(0,0);
    itoa(encoder_axis_5,lcd_encoder_axis_5);
    lcd_puts(lcd_encoder_axis_5);
}
```

```
void stop_axis_1()
{
    PORTA.0 = 0;
    PORTA.1 = 0;
    OCR5AH = 0x00;
    OCR5AL = 0x00;
    encoder_axis_1 = 0;
}

void stop_axis_2()
{
    PORTA.2 = 0;
    PORTA.3 = 0;
    OCR5BH = 0x00;
    OCR5BL = 0x00;
    encoder_axis_2 = 0;
    brake_axis_2_on();
}

void stop_axis_3()
{
    PORTA.4 = 0;
    PORTA.5 = 0;
    OCR5CH = 0x00;
    OCR5CL = 0x00;
    encoder_axis_3 = 0;
    brake_axis_3_on();
}

void stop_axis_4()
{
    PORTA.6 = 0;
    PORTA.7 = 0;
    OCR1AH = 0x00;
    OCR1AL = 0x00;
    encoder_axis_4 = 0;
}

void stop_axis_5()
{
    PORTF.2 = 0;
    PORTF.3 = 0;
    OCR1BH = 0x00;
    OCR1BL = 0x00;
    encoder_axis_5 = 0;
}

void move_axis_1_no_stop(int duty_cycle_axis_1, int arah_axis_1)
{
    int input_1_axis_1, input_2_axis_1;

    if (arah_axis_1 == kanan)
    {
        input_1_axis_1 = 1;
        input_2_axis_1 = 0;
    }
```

```
    if (arah_axis_1 == kiri)
    {
        input_1_axis_1 = 0;
        input_2_axis_1 = 1;
    }
    if (arah_axis_1 == stop)
    {
        input_1_axis_1 = 0;
        input_2_axis_1 = 0;
    }

    PORTA.0 = input_1_axis_1;
    PORTA.1 = input_2_axis_1;

    OCR5AH = duty_cycle_axis_1;
    OCR5AL = duty_cycle_axis_1;
}

void move_axis_2_no_stop(int duty_cycle_axis_2, int arah_axis_2)
{
    int input_1_axis_2, input_2_axis_2;

    if (arah_axis_2 == naik)
    {
        input_1_axis_2 = 1;
        input_2_axis_2 = 0;
    }

    if (arah_axis_2 == turun)
    {
        input_1_axis_2 = 0;
        input_2_axis_2 = 1;
    }
    if (arah_axis_2 == stop)
    {
        input_1_axis_2 = 0;
        input_2_axis_2 = 0;
    }

    PORTA.2 = input_1_axis_2;
    PORTA.3 = input_2_axis_2;

    OCR5BH = duty_cycle_axis_2;
    OCR5BL = duty_cycle_axis_2;

    brake_axis_2_off();
}

void move_axis_3_no_stop(int duty_cycle_axis_3, int arah_axis_3)
{
    int input_1_axis_3, input_2_axis_3;

    if (arah_axis_3 == naik)
    {
        input_1_axis_3 = 1;
        input_2_axis_3 = 0;
    }
```

```
    if (arah_axis_3 == turun)
    {
        input_1_axis_3 = 0;
        input_2_axis_3 = 1;
    }
    if (arah_axis_3 == stop)
    {
        input_1_axis_3 = 0;
        input_2_axis_3 = 0;
    }

    PORTA.4 = input_1_axis_3;
    PORTA.5 = input_2_axis_3;

    OCR5CH = duty_cycle_axis_3;
    OCR5CL = duty_cycle_axis_3;

    brake_axis_3_off();
}

void move_axis_4_no_stop(int duty_cycle_axis_4, int arah_axis_4)
{
    int input_1_axis_4, input_2_axis_4;

     if (arah_axis_4 == naik)
    {
        input_1_axis_4 = 1;
        input_2_axis_4 = 0;
    }

    if (arah_axis_4 == turun)
    {
        input_1_axis_4 = 0;
        input_2_axis_4 = 1;
    }
    if (arah_axis_4 == stop)
    {
        input_1_axis_4 = 0;
        input_2_axis_4 = 0;
    }

    PORTA.6 = input_1_axis_4;
    PORTA.7 = input_2_axis_4;

    OCR1AH = duty_cycle_axis_4;
    OCR1AL = duty_cycle_axis_4;
}

void move_axis_5_no_stop(int duty_cycle_axis_5, int arah_axis_5)
{
    int input_1_axis_5, input_2_axis_5;

    if (arah_axis_5 == kanan)
    {
        input_1_axis_5 = 1;
        input_2_axis_5 = 0;
    }
```

```
    if (arah_axis_5 == kiri)
    {
        input_1_axis_5 = 0;
        input_2_axis_5 = 1;
    }
    if (arah_axis_5 == stop)
    {
        input_1_axis_5 = 0;
        input_2_axis_5 = 0;
    }

    PORTF.2 = input_1_axis_5;
    PORTF.3 = input_2_axis_5;

    OCR1BH = duty_cycle_axis_5;
    OCR1BL = duty_cycle_axis_5;
}

void move_all_axis
(
    int duty_cycle_axis_1, int arah_axis_1, int angle_axis_1,
    int duty_cycle_axis_2, int arah_axis_2, int angle_axis_2,
    int duty_cycle_axis_3, int arah_axis_3, int angle_axis_3,
    int duty_cycle_axis_4, int arah_axis_4, int angle_axis_4,
    int duty_cycle_axis_5, int arah_axis_5, int angle_axis_5
)
{
    int encoder_axis_1_pulse, encoder_axis_2_pulse, encoder_axis_3_pulse,
    encoder_axis_4_pulse, encoder_axis_5_pulse;
    #asm ("sei");
    encoder_axis_1_pulse = angle_axis_1 * 56;
    encoder_axis_2_pulse = angle_axis_2 * 100;
    encoder_axis_3_pulse = angle_axis_3 * 67;
    encoder_axis_4_pulse = angle_axis_4 * 50;
    encoder_axis_5_pulse = angle_axis_5 * 30;

    move_axis_1_no_stop(duty_cycle_axis_1,arah_axis_1);
    move_axis_2_no_stop(duty_cycle_axis_2,arah_axis_2);
    move_axis_3_no_stop(duty_cycle_axis_3,arah_axis_3);
    move_axis_4_no_stop(duty_cycle_axis_4,arah_axis_4);
    move_axis_5_no_stop(duty_cycle_axis_5,arah_axis_5);

    while(1)
    {
        if (encoder_axis_1 >= encoder_axis_1_pulse)
        {
            stop_axis_1();
        }

        if (encoder_axis_2 >= encoder_axis_2_pulse)
        {
            stop_axis_2();
        }

        if (encoder_axis_3 >= encoder_axis_3_pulse)
        {
            stop_axis_3();
        }
```

```
        if (encoder_axis_4 >= encoder_axis_4_pulse)
        {
            stop_axis_4();
        }

        if (encoder_axis_5 >= encoder_axis_5_pulse)
        {
            stop_axis_5();
        }

if ( PORTA.0 == 0 && PORTA.1 == 0 && PORTA.2 == 0 && PORTA.3 == 0 && PORTA.4
== 0 && PORTA.5 == 0 && PORTA.6 == 0 && PORTA.7 == 0 && PORTF.2 == 0 &&
PORTF.3 == 0)
        {
            break;
        }
    }
    #asm ("cli");
}

void default_position()
{
    while( PINF.5 != 1 )
    {
        brake_axis_2_off();
        motor_control(0,kanan,0.8,naik,0,stop,0,stop,0,stop);
    }
    brake_axis_2_on();
    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);

    while( PINF.4 != 1 )
    {
        motor_control(0.5,kanan,0,stop,0,stop,0,stop,0,stop);
    }

    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);

    while( PINF.6 != 1 )
    {
        brake_axis_3_off();
        motor_control(0,kanan,0,stop,0.7,naik,0,stop,0,stop);
    }
    brake_axis_3_on();
    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);

    while( PINF.7 != 1 )
    {
        motor_control(0,stop,0,stop,0,stop,0.7,turun,0,stop);
    }
    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);

    while( PINB.4 != 1 )
    {
        motor_control(0,stop,0,stop,0,stop,0,stop,0.5,kanan);
    }
    motor_control(0,stop,0,stop,0,stop,0,stop,0,stop);
}
```

```
void gripper_control(float gripper_speed, int gripper_status)
{
    int input_1_gripper, input_2_gripper;

    //gripper
    if (gripper_status == buka)
    {
        input_1_gripper = 1;
        input_2_gripper = 0;
    }

    if (gripper_status == tutup)
    {
        input_1_gripper = 0;
        input_2_gripper = 1;
    }
    if (gripper_status == stop)
    {
        input_1_gripper = 0;
        input_2_gripper = 0;
    }

   PORTD.0 = input_1_gripper;
   PORTD.4 = input_2_gripper;

   OCR1CH = MAX*gripper_speed;
   OCR1CL = MAX*gripper_speed;
}

void object_grabbing()
{
    default_position();

move_all_axis(0.6,kiri,190,0.4,turun,70,0.4,turun,40,0.5,naik,30,0.5,kiri,90
);
    delay_ms(1000);
    move_all_axis(0,stop,0,0,stop,0,0.7,turun,25,0.7,naik,15,0,stop,0);
    delay_ms(1000);
    gripper_control(0.6,tutup);
    delay_ms(650);
    move_all_axis(0.5,kiri,60,0,stop,0,0.8,naik,25,1,turun,5,1,kiri,90);
    delay_ms(1000);
    move_all_axis(0,stop,0,0,stop,0,0,stop,0,0,stop,0,0.5,kanan,180);
    delay_ms(5000);
    move_all_axis(0,stop,0,0,stop,0,0,stop,0,0,stop,0,0.5,kiri,180);
    delay_ms(1000);
    move_all_axis(0,stop,0,0,stop,0,0.7,turun,25,0.7,naik,15,0,stop,0);
    delay_ms(1000);
    gripper_control(0.6,buka);
    delay_ms(650);
    default_position();
}
```

```
void move_all_axis_serial()
{
    kirim_char_serial_2("Ready To Recieve Value, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            duty_cycle_axis_1 = UDR2;
            break;
        }
    }
    kirim_char_serial_2("duty_cycle_axis_1 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            arah_axis_1 = UDR2;

            switch(arah_axis_1)
            {
                case 255 :
                arah_axis_1 = kanan;
                break;

                case 0 :
                arah_axis_1 = kiri;
                break;
            }
            break;
        }
    }
    kirim_char_serial_2("arah_axis_1 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            angle_axis_1 = UDR2;
            break;
        }
    }
    kirim_char_serial_2("angle_axis_1 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            duty_cycle_axis_2 = UDR2;
            break;
        }
    }
    kirim_char_serial_2("duty_cycle_axis_2 Recieved, ENTER to continue\n");
```

```
   while(1)
   {
    if(UCSR2A & 0x80)
    {
        arah_axis_2 = UDR2;

        switch(arah_axis_2)
        {
            case 255 :
            arah_axis_2 = naik;
            break;

            case 0 :
            arah_axis_2 = turun;
            break;
        }
        break;
    }
}
kirim_char_serial_2("arah_axis_2 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_2 = UDR2;
        break;
    }
}
kirim_char_serial_2("angle_axis_2 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_3 = UDR2;
        break;
    }
}
kirim_char_serial_2("duty_cycle_axis_3 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_3 = UDR2;

        switch(arah_axis_3)
        {
            case 255 :
            arah_axis_3 = naik;
            break;

            case 0 :
            arah_axis_3 = turun;
            break;
        }
        break;
    }
}
```

```
kirim_char_serial_2("arah_axis_3 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_3 = UDR2;
        break;
    }
}
kirim_char_serial_2("angle_axis_3 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_4 = UDR2;
        break;
    }
}
kirim_char_serial_2("duty_cycle_axis_4 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_4 = UDR2;

        switch(arah_axis_4)
        {
            case 255 :
            arah_axis_4 = naik;
            break;

            case 0 :
            arah_axis_4 = turun;
            break;
        }
        break;
    }
}
kirim_char_serial_2("arah_axis_4 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_4 = UDR2;
        break;
    }
}
kirim_char_serial_2("angle_axis_4 Recieved, ENTER to continue\n");

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_5 = UDR2;
        break;
    }
```

```
    }
    kirim_char_serial_2("duty_cycle_axis_5 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            arah_axis_5 = UDR2;

            switch(arah_axis_5)
            {
                case 255 :
                arah_axis_5 = kanan;
                break;

                case 0 :
                arah_axis_5 = kiri;
                break;
            }
            break;
        }
    }
    kirim_char_serial_2("arah_axis_5 Recieved, ENTER to continue\n");

    while(1)
    {
        if(UCSR2A & 0x80)
        {
            angle_axis_5 = UDR2;
            break;
        }
    }
    kirim_char_serial_2("angle_axis_2 Recieved, ENTER to continue\n");

    move_all_axis
    (
        duty_cycle_axis_1, arah_axis_1, angle_axis_1,
        duty_cycle_axis_2, arah_axis_2, angle_axis_2,
        duty_cycle_axis_3, arah_axis_3, angle_axis_3,
        duty_cycle_axis_4, arah_axis_4, angle_axis_4,
        duty_cycle_axis_5, arah_axis_5, angle_axis_5
    );
}

void web_based()
{
            int default_position_activate = 0;

            while(1)
            {
                if(UCSR2A & 0x80)
                {
                    duty_cycle_axis_1 = UDR2;
                    break;
                }
            }
```

```
while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_1 = UDR2;

        switch(arah_axis_1)
        {
            case 255 :
            arah_axis_1 = kanan;
            break;

            case 0 :
            arah_axis_1 = kiri;
            break;
        }
    break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_1 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_2 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_2 = UDR2;

        switch(arah_axis_2)
        {
            case 255 :
            arah_axis_2 = naik;
            break;

            case 0 :
            arah_axis_2 = turun;
            break;
        }
        break;
    }
}
```

```
 while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_2 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_3 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_3 = UDR2;

        switch(arah_axis_3)
        {
            case 255 :
            arah_axis_3 = naik;
            break;

            case 0 :
            arah_axis_3 = turun;
            break;
        }
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_3 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_4 = UDR2;
        break;
    }
}
```

```
while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_4 = UDR2;

        switch(arah_axis_4)
        {
            case 255 :
            arah_axis_4 = naik;
            break;

            case 0 :
            arah_axis_4 = turun;
            break;
        }
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        angle_axis_4 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        duty_cycle_axis_5 = UDR2;
        break;
    }
}

while(1)
{
    if(UCSR2A & 0x80)
    {
        arah_axis_5 = UDR2;

        switch(arah_axis_5)
        {
            case 255 :
            arah_axis_5 = kanan;
            break;

            case 0 :
            arah_axis_5 = kiri;
            break;
        }
        break;
    }
}

while(1)
{
```

```
                if(UCSR2A & 0x80)
                {
                    angle_axis_5 = UDR2;
                    break;
                }
            }

            if
            (
                duty_cycle_axis_1 == 1 && arah_axis_1 == 1 &&
                angle_axis_1 == 1 &&
                duty_cycle_axis_2 == 1 && arah_axis_2 == 1 &&
                angle_axis_2 == 1 &&
                duty_cycle_axis_3 == 1 && arah_axis_3 == 1 &&
                angle_axis_3 == 1 &&
                duty_cycle_axis_4 == 1 && arah_axis_4 == 1 &&
                angle_axis_4 == 1 &&
                duty_cycle_axis_5 == 1 && arah_axis_5 == 1 &&
                angle_axis_5 == 1
            )
            {
                default_position();
                default_position_activate = 1;
            }

            if( default_position_activate == 0)
            {
                move_all_axis
                (
                    duty_cycle_axis_1, arah_axis_1, angle_axis_1,
                    duty_cycle_axis_2, arah_axis_2, angle_axis_2,
                    duty_cycle_axis_3, arah_axis_3, angle_axis_3,
                    duty_cycle_axis_4, arah_axis_4, angle_axis_4,
                    duty_cycle_axis_5, arah_axis_5, angle_axis_5
                );
            }
}

void led_indicator_off()
{
    PORTD.5 = 0;
}

void led_indicator_on()
{
    PORTD.5 = 1;
}

void main(void)
{
    oscillator_setting();
    ports_setting();
    timers_setting();
    interrupt_setting();
    usart_2_setting();

    while(1)
    {
        int executed_value = 0;
        int perintah;
```

```
            lcd_init(16);

            lcd_setting();
            lcd_gotoxy(0,0);
            lcd_putsf("Robot Initialize");

            delay_ms(3000);

            lcd_clear();

            lcd_gotoxy(0,0);
            lcd_putsf("Ready to Execute");
            delay_ms(3000);

            lcd_clear();

            led_indicator_on();

            while(executed_value == 0)
            {
                if(UCSR2A & 0x80)
                {
                    perintah = UDR2;

                    switch(perintah)
                    {
                        case 255 :
                        default_position();
                        kirim_char_serial_2("Default Position Executed,
                        press ENTER to continue\n");
                        executed_value = 1;
                        led_indicator_off();
                        break;

                        case 254 :
                        move_all_axis_serial();
                        executed_value = 1;
                        led_indicator_off();
                        break;

                        case 253 :
                        lcd_gotoxy(0,0);
                        lcd_putsf("Web Based Mode");
                        web_based();
                        executed_value = 1;
                        led_indicator_off();
                        break;

                        case 252 :
                        executed_value = 1;
                        led_indicator_off();
                        break;
                    }
                }
            }
        }
}
```

LAMPIRAN 2

```
/*************************************************************
Program Interface PC Dengan Microcontroller
Aplikasi Kontrol Robot Berbasis Web
Author : Hendra Prima Syahputra
Departemen Teknik Mesin Fakultas Teknik Universitas Indonesia
*************************************************************/

/*header - header*/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>

/*
Serial Port Address
COM1 0x3F8
COM2 0x2F8
COM3 0x3E8
cOM4 0x2E8
*/
#define PORT1 0x3F8

void get_serial_data()
{
        int c;
        int counter;
        char buffer[100];

        while(1)
        {
                c = inportb(PORT1 + 5);
                if(c & 1)
                {
                        buffer[counter] = inport(PORT1);
                        printf("%c", buffer[counter]);
                }
                if(kbhit())
                {
                        break;
                }
        }
}

void open_com_port()
{
        */
        outportb(PORT1 + 1 , 0);    /* Turn off interrupts - Port1 */


       /*PORT 1 - Communication Settings*/

        outportb(PORT1 + 3 , 0x80);  /* SET DLAB ON */
        outportb(PORT1 + 0 , 0x01);  /* Set Baud rate
                                        Divisor Latch Low Byte*/
                                     /* Default 0x03 =  38,400 BPS */
                                     /*         0x01 = 115,200 BPS */
```

```
                                        /*          0x02 =  57,600 BPS */
                                        /*          0x06 =  19,200 BPS */
                                        /*          0x0C =   9,600 BPS */
                                        /*          0x18 =   4,800 BPS */
                                        /*          0x30 =   2,400 BPS */
        outportb(PORT1 + 1 , 0x00);     /* Set Baud rate
                                        Divisor Latch High Byte */
        outportb(PORT1 + 3 , 0x03);     /* 8 Bits, No Parity, 1 Stop Bit */
        outportb(PORT1 + 2 , 0xC7);     /* FIFO Control Register */
        outportb(PORT1 + 4 , 0x0B);     /* Turn on DTR, RTS, and OUT2 */
}

void default_position_command()
{
        */
        int value_default = 255;

        clrscr();
        outportb(PORT1, value_default);

        get_serial_data();
}


void move_all_axis_command()
{
        clrscr();
        printf("Please Insert Parameter\n");
        printf("Value Range 0-255\n");

        int duty_cycle_axis_1, arah_axis_1, angle_axis_1;
        int duty_cycle_axis_2, arah_axis_2, angle_axis_2;
        int duty_cycle_axis_3, arah_axis_3, angle_axis_3;
        int duty_cycle_axis_4, arah_axis_4, angle_axis_4;
        int duty_cycle_axis_5, arah_axis_5, angle_axis_5;

        int value_move_all_axis = 254;

        outportb(PORT1, value_move_all_axis);

        get_serial_data();

        //duty_cycle_axis_1
        printf("duty_cycle_axis_1 :  ");
        scanf("%d", &duty_cycle_axis_1);
        outportb(PORT1, duty_cycle_axis_1);

        get_serial_data();

        //arah_axis_1
        printf("arah_axis_1 :  ");
        scanf("%d", &arah_axis_1);
        outportb(PORT1, arah_axis_1);

        get_serial_data();

        //angle_axis_1
        printf("angle_axis_1 :  ");
        scanf("%d", &angle_axis_1);
        outportb(PORT1, angle_axis_1);
```

```
get_serial_data();

//duty_cycle_axis_2
printf("duty_cycle_axis_2 :  ");
scanf("%d", &duty_cycle_axis_2);
outportb(PORT1, duty_cycle_axis_2);

get_serial_data();

//arah_axis_2
printf("arah_axis_2 :  ");
scanf("%d", &arah_axis_2);
outportb(PORT1, arah_axis_2);

get_serial_data();

//angle_axis_2
printf("angle_axis_2 :  ");
scanf("%d", &angle_axis_2);
outportb(PORT1, angle_axis_2);

get_serial_data();

//duty_cycle_axis_3
printf("duty_cycle_axis_3 :  ");
scanf("%d", &duty_cycle_axis_3);
outportb(PORT1, duty_cycle_axis_3);

get_serial_data();

//arah_axis_3
printf("arah_axis_3 :  ");
scanf("%d", &arah_axis_3);
outportb(PORT1, arah_axis_3);

get_serial_data();

//angle_axis_3
printf("angle_axis_3 :  ");
scanf("%d", &angle_axis_3);
outportb(PORT1, angle_axis_3);

get_serial_data();

//duty_cycle_axis_4
printf("duty_cycle_axis_4 :  ");
scanf("%d", &duty_cycle_axis_4);
outportb(PORT1, duty_cycle_axis_4);

get_serial_data();

//arah_axis_4
printf("arah_axis_4 :  ");
scanf("%d", &arah_axis_4);
outportb(PORT1, arah_axis_4);

get_serial_data();

//angle_axis_4
```

```
        printf("angle_axis_4 :  ");
        scanf("%d", &angle_axis_4);
        outportb(PORT1, angle_axis_4);

        get_serial_data();

        //duty_cycle_axis_5
        printf("duty_cycle_axis_5 :  ");
        scanf("%d", &duty_cycle_axis_5);
        outportb(PORT1, duty_cycle_axis_5);

        get_serial_data();

        //arah_axis_5
        printf("arah_axis_5 :  ");
        scanf("%d", &arah_axis_5);
        outportb(PORT1, arah_axis_5);

        get_serial_data();

        //angle_axis_5
        printf("angle_axis_5 :  ");
        scanf("%d", &angle_axis_5);
        outportb(PORT1, angle_axis_5);

        get_serial_data();
}

void manual_mode()
{
        int select_command;

        clrscr();
        printf("Manual Mode Activated\n");
        printf("Select Command\n");
        printf("1. Default Position\n");
        printf("2. Move_All_Axis\n");
        scanf("%d", &select_command);

        switch(select_command)
        {
                case 1 :
                default_position_command();
                break;

                case 2 :
                move_all_axis_command();
                break;

                default:
                printf("Invalid Mode Selection\n");
        }
}

void web_based_loop()
{
        while(1)
        {
                clrscr();
```

```
                FILE *file_pointer;
                int counter = 0;
                int movement_data[15] = {0};
                int value_web_based;

            /*Membuka file*/
            file_pointer = fopen("D:/adtj/www/htdocs/i-RoMan/kdt.txt","rb");

            /*Menghentikan loop dengan kode keyboard*/
            if(kbhit())
                {
/*
Pada saat keluar, data yang   dikirim bernilai 0  untuk memastikan robot
berhenti, nilai ini sama dengan data emergency yang digenerate oleh web.
*/
                        movement_data[0] = 0;
                        movement_data[1] = 0;
                        movement_data[2] = 0;
                        movement_data[3] = 0;
                        movement_data[4] = 0;
                        movement_data[5] = 0;
                        movement_data[6] = 0;
                        movement_data[7] = 0;
                        movement_data[8] = 0;
                        movement_data[9] = 0;
                        movement_data[10] = 0;
                        movement_data[11] = 0;
                        movement_data[12] = 0;
                        movement_data[13] = 0;
                        movement_data[14] = 0;

                        /*Mengirim data ke microcontroller*/
                        delay(100);
                        outportb(PORT1, movement_data[0]);
                        delay(100);
                        outportb(PORT1, movement_data[1]);
                        delay(100);
                        outportb(PORT1, movement_data[2]);
                        delay(100);
                        outportb(PORT1, movement_data[3]);
                        delay(100);
                        outportb(PORT1, movement_data[4]);
                        delay(100);
                        outportb(PORT1, movement_data[5]);
                        delay(100);
                        outportb(PORT1, movement_data[6]);
                        delay(100);
                        outportb(PORT1, movement_data[7]);
                        delay(100);
                        outportb(PORT1, movement_data[8]);
                        delay(100);
                        outportb(PORT1, movement_data[9]);
                        delay(100);
                        outportb(PORT1, movement_data[10]);
                        delay(100);
                        outportb(PORT1, movement_data[11]);
                        delay(100);
                        outportb(PORT1, movement_data[12]);
                        delay(100);
                        outportb(PORT1, movement_data[13]);
```

```
                        delay(100);
                        outportb(PORT1, movement_data[14]);

                        /*Menutup file dan menghapus file*/
                        fclose(file_pointer);
                        remove("D:/adtj/www/htdocs/i-RoMan/kdt.txt");
                        break;
                }

                /*File ditemukan*/
                else if(file_pointer != NULL)
                {
                        printf("Movement File Found!\n");
                        printf("Press ENTER to exit\n");

                    /*Membaca isi file untuk dimasukkan
                    sebagai data pergerakan*/
                    while(!feof(file_pointer))
{

fscanf(file_pointer,"%d",&movement_data[counter]);
                    counter++;
                    }

                        /*Mengirim data ke microcontroller*/
                        delay(100);
                        outportb(PORT1, movement_data[0]);
                        delay(100);
                        outportb(PORT1, movement_data[1]);
                        delay(100);
                        outportb(PORT1, movement_data[2]);
                        delay(100);
                        outportb(PORT1, movement_data[3]);
                        delay(100);
                        outportb(PORT1, movement_data[4]);
                        delay(100);
                        outportb(PORT1, movement_data[5]);
                        delay(100);
                        outportb(PORT1, movement_data[6]);
                        delay(100);
                        outportb(PORT1, movement_data[7]);
                        delay(100);
                        outportb(PORT1, movement_data[8]);
                        delay(100);
                        outportb(PORT1, movement_data[9]);
                        delay(100);
                        outportb(PORT1, movement_data[10]);
                        delay(100);
                        outportb(PORT1, movement_data[11]);
                        delay(100);
                        outportb(PORT1, movement_data[12]);
                        delay(100);
                        outportb(PORT1, movement_data[13]);
                        delay(100);
                        outportb(PORT1, movement_data[14]);

                        printf("Movement File Executed\n");
```

```
/*Menampilkan data yang dikirim*/

printf("%d %d %d\n", movement_data[0], movement_data[1],
movement_data[2]);
printf("%d %d %d\n", movement_data[3],    movement_data[4],
movement_data[5]);
printf("%d %d %d\n", movement_data[6], movement_data[7],
movement_data[8]);
printf("%d %d %d\n", movement_data[9], movement_data[10],
movement_data[11]);
printf("%d %d %d\n", movement_data[12], movement_data[13],
movement_data[14]);

/*Mengembalikan kondisi microcontroller pada kondisi web based*/

                        delay(3000);
                        printf("Sending Web Command\n");

                        delay(3000);
                        value_web_based = 253;

                        delay(4000);
                        outportb(PORT1, value_web_based);

/*Menutup dan membuang file*/

                        fclose(file_pointer);
                        remove("D:/adtj/www/htdocs/i-RoMan/kdt.txt");

                        printf("File Removed\n");
                }

                /*File tidak ditemukan*/
                else if(file_pointer == NULL)
                {
                        printf("Movement File NOT Found!\n");
                        printf("Press ENTER to exit\n");

                        fclose(file_pointer);
                }

                else
                {
                        fclose(file_pointer);
                }
                delay(1000);
        }
}

void web_mode()
{
        int value_web_based;
        int web_based_command;
        clrscr();
        printf("1. Activate Web Based Loop\n");
        printf("2. Exit\n");
        scanf("%d", &web_based_command);

        switch(web_based_command)
```

```
        {
                case 1 :
                value_web_based = 253;
                outportb(PORT1, value_web_based);
                web_based_loop();
                break;

                case 2 :
                value_web_based = 252;
                outportb(PORT1, value_web_based);
                break;

                default:
                printf("Invalid Mode Selection\n");
        }
}
void exit_program()
{
        /*Fungsi untuk keluar dari program*/
        exit(0);
}

void main()
{
        while(1)
        {
                int select_mode;
                clrscr();

                printf("Movemaster RV-M1 Interface Driver\n");
                printf("Select Mode\n");
                printf("1. Manual Mode\n");
                printf("2. Web Based\n");
                printf("3. Exit\n");
                scanf("%d", &select_mode);

                switch(select_mode)
                {
                        case 1 :
                        open_com_port();
                        manual_mode();
                        break;

                        case 2 :
                        open_com_port();
                        web_mode();
                        break;

                        case 3 :
                        exit_program();
                        break;

                        default:
                        printf("Invalid Mode Selection\n");
                }
                getch();
        }
}
```