IMPLEMENTASI *INVERS KINEMATICS* PADA SISTEM PERGERAKAN *MOBILE* ROBOT RODA MEKANUM

SKRIPSI

Diajukan Untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Teknik

DISUSUN OLEH :

VERI HENDRAYAWAN

NIM. 105060301111004-63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

IMPLEMENTASI *INVERS KINEMATICS* PADA SISTEM PERGERAKAN *MOBILE* ROBOT RODA MEKANUM

SKRIPSI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

VERI HENDRAYAWAN
NIM. 105060301111004-63

Telah diperiksa dan disetujui oleh :

Pembimbing I	**Pembimbing II**
Ir. Nanang Sulistiyanto, MT.	**Eka Maulana, ST, MT, M.Eng.**
NIP. 19700113 199403 1002	**NIK. 8411300611 0280**

LEMBAR PENGESAHAN

IMPLEMENTASI *INVERS KINEMATICS* PADA SISTEM PERGERAKAN *MOBILE* ROBOT RODA MEKANUM

Disusun oleh:
VERI HENDRAYAWAN
NIM. 105060301111004-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 11 Agustus 2014

DOSEN PENGUJI

Ir. Ponco Siwindarto, M.Eng.Sc
NIP. 19590304 198903 1 001

Moch. Rif'an, ST., MT.
NIP. 19710301 200012 1 001

Dr.-Ing. Onny Setyawati, ST.,MT.,M.Sc
NIP. 19740417 200003 2 007

Mengetahui,
Ketua Jurusan Teknik Elektro

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

PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kepada Allah SWT, karena atas segala petunjuk dan nikmat-Nya lah skripsi ini dapat diselesaikan.

Skripsi berjudul "Implementasi *Invers Kinematics* pada Sistem Pergerakan *Mobile* Robot Roda Mekanum" ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa dalam penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

- Allah SWT atas rahmat dan hidayah yang telah diberikan,
- Rosulullah Muhammad SAW, semoga shalawat serta salam selalu tercurah kepada beliau,
- Ayah dan Ibu atas segala nasehat, kasih sayang, perhatian dan kesabarannya didalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Hadi Suyono, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Moch. Rif'an, ST., MT. selaku Ketua Prodi Strata Satu Jurusan Teknik Elektro Universitas,
- Bapak Ir. Nanang Sulistyanto, MT selaku pembimbing I, terima kasih atas segala bimbingan, pengarahan, ide, saran, motivasi, dan masukan yang diberikan,
- Bapak Eka Maulana, ST, MT, M.Eng selaku pembimbing II, terima kasih atas segala bimbingan, pengarahan, ide, saran, motivasi, dan masukan yang diberikan,
- Bapak Ir. Mahfudz Shidiq, MT. selaku dosen Penasehat Akademik,
- Ibu Ir. Nurussa'adah, MT. selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya,
- Seluruh dosen pengajar Teknik Elektro Universitas Brawijaya,

- Staff Recording Jurusan Teknik Elektro,
- Teman – teman Magnet ankatan 2010,
- Teman – teman Magnetronika 2010,
- Rekan seperjuangan dalam skripsi Basori, Abu, Tanshu, Anas, Erwan, Vicky, Rita, Erny, Nunik, Agwin, Ari terima kasih atas segala bantuan yang telah diberikan,
- Rekan-rekan Tim Robot Kontes Robot Abu Indonesia, Irham, Jaka, Ainun, Mirza, Fikrul, Agus terima kasih atas segala bantuan dan semangat yang telah diberikan,
- Rekan-rekan Asisten Laboratorium Elektroika.
- Seluruh Keluarga Besar Tim Robot UB Jurusan Teknik Elektro atas segala bantuan alat, bahan, dan masukan – masukan yang telah diberikan,
- Sekret KRAI, sekret KRPAI, sekret KRSI, dan Laboratorium Elektronika yang selama ini telah menyediakan tempat bagi penulis dalam mengerjakan skripsi ini,
- Seluruh teman-teman serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan satu-persatu, terimakasih banyak atas bantuan dan dukungannya.

Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi serta bagi masyarakat.

Malang, Juli 2014

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Veri Hendrayawan, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, Juli 2014, *Implementasi Invers Kinematics pada Sistem Pergerakan Mobile Robot Roda Mekanum*, Dosen Pembimbing : Ir. Nanang Sulistiyanto, MT. dan Eka Maulana, ST., MT., M.Eng.

Teknologi pergerakan *mobile* robot sekarang semakin berkembang, salah satu diantaranya adalah penggunaan roda mekanum untuk efisiensi pergerakan *mobile* robot. Dalam perkembangannya teknologi pergerakan konvensional seperti *swerve drive* memiliki banyak kekurangan diantaranya tidak bisa bergerak ke segala arah dan kurangnya efisiensi pergerakan *mobile* robot. Penggunaan *mobile* robot salah satunya pada Kontes Robot Abu Indonesia yang mempunyai lintasan robot yang bervariasi seperti zig-zag dan parabola sehingga diperlukan metode pergerakan baru yang menghasilkan pergerakan *mobile* robot dengan kecepatan dan efisiensi pergerakan yang tinggi. Untuk mengatur pergerakan *mobile* robot roda mekanum digunakan persamaan *invers kinematics* yang mengubah kecepatan robot dalam V_x, V_y, ω menjadi kecepatan masing-masing roda yaitu ω_1, ω_2, ω_3, ω_4 dan arah putaran roda. Penggunaan motor *stepper* memiliki ketelitian cukup baik dengan rata-rata kesalahan 2,09% dan tidak memerlukan umpan balik untuk kontrol kecepatan dalam implementasi penggerak *mobile* robot roda mekanum. Komunikasi data antara mikrokontroler *master* ATMega 16 dan mikrokontroler *slave* ATMega 8 menggunakan komunikasi SPI dengan tingkat keberhasilan 100 % pada frekuensi 4000.000 KHz. *Mobile* robot roda mekanum dapat bergerak ke sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0 dengan kesalahan rata-rata kecepatan roda 3,6 rpm dan kesalahan rata – rata sudut gerak robot sebesar 2,93°.

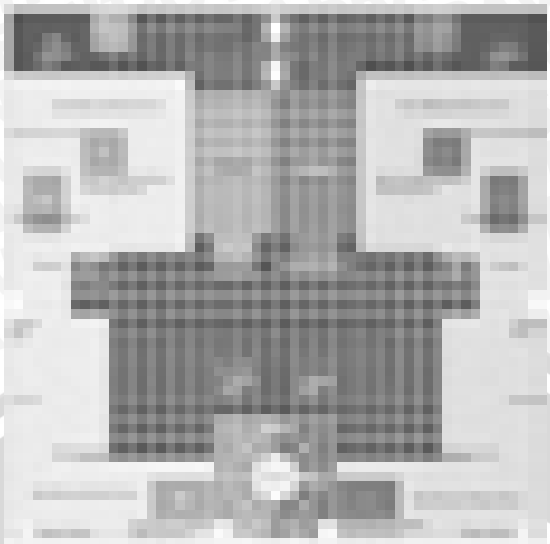
Kata kunci : *invers kinematics* , *mobile* robot roda mekanum, motor *stepper*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi robotika sangat berperan dalam peningkatan efisiensi kehidupan manusia. Peran serta teknologi robotika bisa dilihat dalam kehidupan sehari-hari. Selama ini teknologi robotika semakin berkembang, mulai *mobile* robot sampai *humanoid* robot. Perkembangan teknologi ini tidak luput dari peran institusi pendidikan ataupun para pengembang. Kontes robot internasional *Asia Pasific Broadcasting Union Robot Contest* (ABU Robocon) diadakan setiap tahun dan pertama diadakan pada tahun 2002. Setiap tahun aturan lomba selalu berganti dan menuntut solusi pemecahan permasalan yang paling baik. Salah satu permasalahan tersebut adalah pergerakan robot dengan efisiensi yang tinggi dan waktu yang sesingkat mungkin. Peraturan lomba ABU Robocon tahun 2009 dan 2012 misalnya, robot harus melewati lintasan yang berbentuk zig zag yang tentunya dibutuhkan pergerakan yang berbeda dengan lintasan biasa. Peraturan lomba ABU Robocon 2013 dan 2014 robot dituntut untuk melintasi lintasan yang berbentuk parabolik (Dikti, 2013). Berdasarkan permasalahan tersebut kita dituntut untuk membuat teknologi pergerakan *mobile* robot yang lebih cepat dan efisien, salah satu diantaranya adalah teknologi pergerakan *mobile* robot.

Selama ini kebanyakan teknologi pergerakan *mobile* robot menggunakan sistem pergerakan *differential drive. Differential drive* merupakan salah satu pergerakan robot dengan memanfaatkan kecepatan pada roda kiri dan kanannya. Permasalah utama dari sistem pergerakan *differential drive* adalah terbatasnya pergerakan robot karena hanya memanfaatkan kecepatan pada roda kiri dan kanannya sehinga robot hanya memiliki dua derajat kebebasan, yakni bergerak maju dan belok tetapi robot tidak mampu bergerak ke samping atau biasa disebut robot *non-holonomic* (Boreinstein, 1996).



Gambar 1.1 Lapangan Abu Robocon 2012.
Sumber: Robocon (2012)

Holonomic mobile robot terdiri atas *omnidirectional wheel* dan *mecanum wheel.* Roda mekanum pertama kali ditemukan oleh Bength llon di Swedia. Roda mekanum terdiri dari beberapa roler membentuk sudut 45°. Konfigurasi pemasangan roda mekanum dengan arah putaran dan kecepatan masing-masing roda akan menghasilkan pergerakan robot yang lebih variatif. Perbandingan sistem pergerakan roda mekanum dengan sistem pergerakan menggunakan roda *omni directional* adalah roda mekanum mempunyai keunggulan mekanik robot yang simpel dan stabil pada beban tinggi (Byun, 2004). Penelitian *mobile* robot dengan sistem pergerakan *omni directional* telah dilakukan (Artha, 2013) dengan kesalahan sudut gerak rata-rata adalah 4,83°.

Kinematika *mobile* robot sangat penting untuk mendefinisikan arah dan kecepatan *mobile* robot. Arah dan kecepatan masing-masing roda dihasilkan total vektor gaya dengan arah pergerakan *mobile* robot yang diinginkan tanpa mengubah arah hadap *mobile* robot maupun sudut kemiringan roda (West, 1997). Sistem pergerakan *mobile* robot roda mekanum dapat bergerak dengan sudut 0° sampai 360°.

Perhitungan kinematika mekanum dapat diproses menggunakan perangkat keras pengontrol, salah satunya adalah mikrokontroler. Mikrokontroler merupakan perangkat keras yang bertujuan untuk mengolah data *input* menjadi data *output* sesuai dengan proses perintah yang ada di dalamnya. Komunikasi SPI merupakan

komunikasi data serial antar *peripheral,* salah satu diantaranya komunikasi antar *master* dan *slave.*

Oleh karena itu dalam skripsi ini dirancang dan dibuat *prototype mobile* robot roda mekanum yang dapat bergerak ke segala arah menggunakan mikrokontroler yang diprogram menggunakan bahasa C.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, dapat disusun rumusan masalah sebagai berikut :

1) Bagaimana merancang, mengimplementasikan, dan menganilisa kinematika *mobile* robot roda mekanum.
2) Bagaimana merancang, mengimplementasikan, dan menganilisa komunikasi SPI antara mikrokontroler *master* dengan mikrokontroler *slave.*
3) Bagaimana merancang, mengimplementasikan, dan menganalisa sistem agar didapatkan sudut pergerakan robot sesuai dengan perhitungan kinematiknya.

1.3 Batasan Masalah

Dengan mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan penelitian akan diberikan batasan sebagai berikut :

1) Karakteristik motor *stepper* yang digunakan adalah 1,8°/pulsa.
2) Mode yang digunakan diantaranya mode pergerakan dengan sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0.
3) Mode tanpa pergerakan rotasi robot.
4) Lapangan menggunakan bahan *vinil.*

1.4 Tujuan

Tujuan skripsi ini adalah merancang dan membangun sistem pergerakan *mobile* robot roda mekanum yang bergerak ke segala arah *(holonomic)* dengan mengimplementasikan *invers kinematics* dan menggunakan mikrokontroler sebagai pemroses utamanya.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini sebagai berikut :

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, ruang lingkup, tujuan, dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan *mobile* robot.

BAB III Metodologi

Berisi tentang metode-metode yang dipakai dalam melakukan perancangan, pengujian, dan analisis data.

BAB IV Perancangan

Perancangan dan perealisasian *mobile* robot yang meliputi spesifikasi, perencanaan diagram blok, prinsip kerja dan realisasi *mobile* robot.

BAB V Pengujian dan Analisis

Memuat aspek pengujian meliputi penjelasan tentang cara pengujian dan hasil pengujian. Aspek analisis meliputi penilaian atau komentar terhadap hasil-hasil pengujian. Pengujian dan analisis terhadap *mobile* robot yang telah direalisasikan berdasarkan masing-masing blok dan sistem secara keseluruhan.

BAB VI Kesimpulan dan Saran

Memuat intisari hasil pengujian dan menjawab rumusan masalah serta memberikan rekomendasi untuk perbaikan kualitas penelitian dimasa yang akan datang.

BAB II
TINJAUAN PUSTAKA

Bab ini bertujuan untuk memudahkan dalam memahami cara kerja rangkaian maupun dasar dasar perencanaan dari *mobile* robot ini, maka perlu adanya penjelasan dan uraian teori penunjang yang digunakan dalam penulisan ini. Teori penunjang yang akan dijelaskan dalam bab ini adalah:

- Roda Mekanum
- Kinematika Mekanum
- Komunikasi UART
- Komunikasi Data SPI
- Mikrokontroler *Master* dan *Slave*
- LCD (*Liquid Crystal Display*)
- *Driver* Motor *Stepper*
- Motor *Stepper*

2.1 Roda Mekanum

Roda mekanum berawal pada tahun 1973 oleh Bengt Ilon. Roda mekanum berbasis desain roda yang dikelilingi oleh *roller* dengan membentuk sudut 45°. Arah dan kecepatan masing-masing roda menghasilkan *resultan* gaya yang menerjemahkan pergerakan mobilitas tanpa mengubah arah hadap robot ataupun perubahan sudut roda (West,1997).

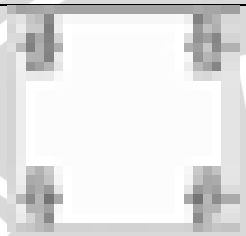
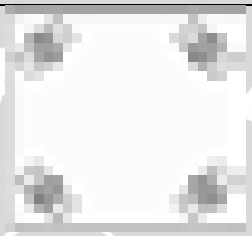
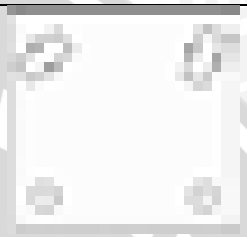
Gambar 2.1 Vektor Roda Mekanum.
Sumber: Song dan Byun (2004 :193)

Mobile robot yang menggunakan empat roda mekanum menghasilkan pergerakan *holonomic* tanpa membutuhkan cara pengendalian konvensional. Kekurangan dalam penggunaan roda mekanum adalah masalah slip pada roda

karena bagian *roller* yang bersentuhan pada lantai hanya satu buah dalam satu waktu (Habib, 2007).

Perkembangan sistem pergerakan robot diantaranya adalah pergerakan menggunakan sistem *swerve drive, omni directional drive*, dan *mecanum drive*. Perbedaan antara ketiga tipe sistem pergerakan tersebut ditunjukkan dalam Tabel 2.1.

Tabel 2.1 Perbedaan Sistem Pergerakan *Mobile* Robot

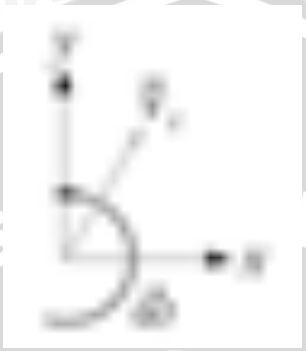
	Mecanum	**Omnidirectional**	**Swerve Drive**
Deskripsi	 Roda dengan sudut *roller* 45°	 Roda dengan sudut *roller* 90°	 Pergerakan dengan pengendalian roda sendiri
Keunggulan	- Desain simpel - Mampu menerima beban tinggi - Pengontrolen simpel - Mampu bergerak diagonal	- Bobot rendah - Desain simpel - Pengointrolan simpel - Mampu bergerak diagonal	- Konsep pengendalian simpel - Desain roda simpel - Kontak dengan lantai secara terus-menerus - Mampu menerima beban tinggi - Toleransi terhadap kondisi lantai
Kekurangan	- Konsep sistem pergerakan kompleks - Konsep Roda kompleks - Kontak dengan lantai yang tidak terus-menerus - Sensitive terhadap kondisi lantai	- Konsep sistem pergerakan yang lebih kompleks - Kontak dengan lantai yang tidak terus-menerus - Sensitif terhadap kondisi lantai	- Desain mekanik yang kompleks - Kompleks dalam program dan kontrol - Gaya gesek dengan lantai yang besar

Sumber: Song dan Byun (2004:204)

Berdasarkan Tabel 2.1 roda mekanum mempunyai banyak kelebihan dibandingkan sistem pergerakan yang lain. Perbedaan antara sistem pergerakan menggunakan roda mekanum dengan roda *omni directional* adalah roda mekanum lebih stabil pada beban tinggi (Byun, 2007). .

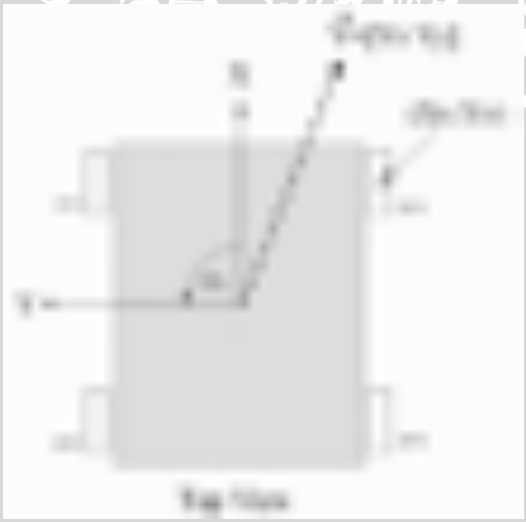
2.2 Kinematika Mekanum

Persamaan kinematik salah satunya dapat mendiskripsikan pergerakan suatu robot sehingga pengontrolan pergerakan robot dapat mudah dilakukan. Persamaan kinematik mendefinisikan beberapa bagian dari pergerakan robot, diantaranya V_t dan ω (Mackenzie, 2006).



Gambar 2.2 Grafik Kinematik Robot.
Sumber : Mackenzie (2006 :21)

Persamaan kinematik dapat mengolah data dari empat roda mekanum menjadi kecepatan sudut masing-masing roda sehingga menghasilkan kecepatan dan pergerakan rotasi robot.

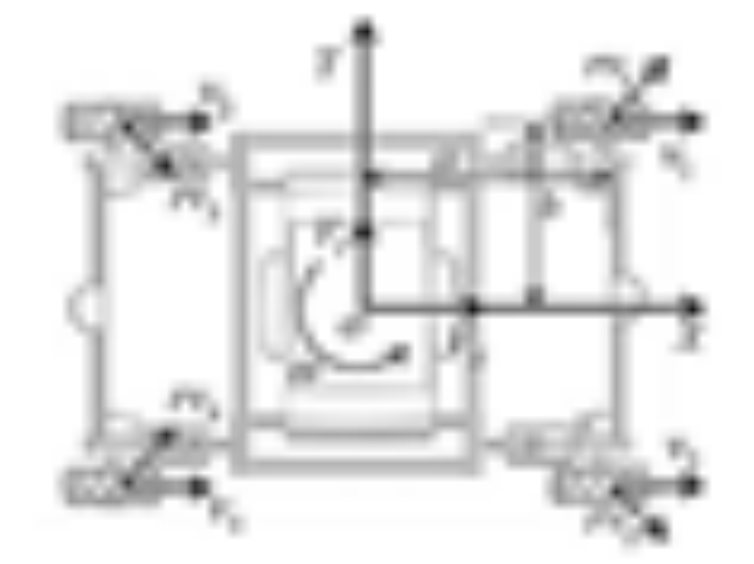


Gambar 2.3 Skema Kinematik Robot.
Sumber: Efendi(2005:1831)

Berdasarkan gambar 2.3 koordinat sistem pergerakan robot dari empat roda dengan jarak antar pusat roda yang sama. Koprdinat x menunjukkan pegerakan ke depan, sedangkan koordinat y menunjukkkan pergerakan ke samping.

Kecepatan robot dalam [*V*] dapat di jabarkan dengan matrik 3x1 yaitu [*Vx Vy* ω] yang merepresentasikan kecepatan translasi dan rotasi robot dalam suatu

waktu. Vektor pergerakan *mobile* robot roda mekanum ditunjukkan dalam Gambar 2.5 :



Gambar 2.5 Skema Pergerakan Roda Mekanum.
Sumber: Choi (2012:2)

Berdasarkan Gambar 2.5 dapat dijelaskan bahwa a adalah jarak antara roda depan dan titik pusat robot. b adalah jarak roda samping dengan titik pusat robot. v_i merepresentasikan kecepatan roda nomor i, rv_i merepresentasikan kecepatan dari *roller*, *Vx, Vy,* ω merepresentasikan kecepatan robot dan kecepatan sudut dalam sumbu z. titik tengah robot. Jarak vertikal, horisontal, arah dan sudut dari *roller* dan roda didefinisikan sebagai berikut :

$i = 1, 2, 3, 4$

$a_i = \{a, a, -a, -a\}$

$b_i = \{b, -b, b, -b\}$

$\alpha_i = \{\pi/4, -\pi/4, -\pi/4, \pi/4\}$

Penomoran roda i (i=1,2,3,4) merepresentasikan dalam koordinat sistem x dan y dapat ditunjukkan dalam persamaan (2-1) sampai (2-3) :

$$v_i + rv_i \cos(\alpha_i) = V_x - b_i\omega \quad (2\text{-}1)$$

$$rv_i \sin(\alpha_i) = V_y + a_i\omega \quad (2\text{-}2)$$

$$v_i = V_x - b\omega - \frac{Vy + ai\omega}{\tan(\alpha i)} \quad (2\text{-}3)$$

berdasarkan persamaan (3), kecepatan keempat roda dapat didefinisikan dalam *tan* (α_i) = (1, -1, -1, 1). Sehingga dapat diperoleh :

$$v_1 = V_x - V_y - a\omega - b\omega \quad (2\text{-}4)$$

$$v_2 = V_x + V_y + a\omega + b\omega \quad (2\text{-}5)$$

$$v_3 = V_x + V_y - a\omega - b\omega \quad (2\text{-}6)$$

$$v_4 = V_x - V_y + a\omega + b\omega \quad (2\text{-}7)$$

Karena, $v = \omega \cdot R$, dengan R adalah jari-jari roda. Maka persamaan (2-4) sampai (2-7) menjadi persamaan (2-8).

$$\begin{bmatrix}\omega 1\\ \omega 2\\ \omega 3\\ \omega 4\end{bmatrix} = \frac{1}{R}\begin{bmatrix}1 & -1 & -a & -b\\ 1 & 1 & a & b\\ 1 & 1 & -a & -b\\ 1 & -1 & a & b\end{bmatrix}\begin{bmatrix}Vx\\ Vy\\ \omega\end{bmatrix} \quad (2\text{-}8)$$

Persamaan (2-8) merupakan persamaan yang digunakan dalam sistem pergerakan *mobile* robot roda mekanum.

2.3 Mikrokontroler

Mikrokontroler yang digunakan dalam tugas akhir ini menggunakan dua IC mikrokontroler, yaitu mikrokontroler *master* dan mikrokontroler *slave*. ATMega 16 digunakan sebagai *master* dan empat ATMega 8 sebagai *slave.*

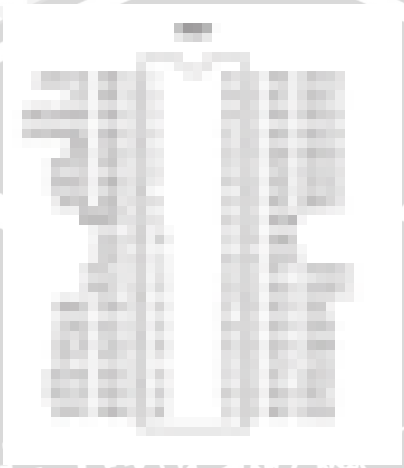
2.3.1 Mikrokontroler Master

ATMega 16 merupakan salah satu mikrokontroler yang banyak digunakan saat ini yaitu mikrokontroler AVR. AVR adalaah mikrokontroler RISC (*reduce instruction set compute*) 8 bit berdasarkan arsitektur havard, yang dibuat oleh atmel pada tahun 1996. AVR memiliki keungulan dibandingkan mikrokontroler lain. Keunggulan mikrokontroler AVR yaitu AVR memiliki kecepatan eksekusi program yang lebih cepat karena sebagian besar instruksi dieksekusi dalam 1 siklus *clock,* lebih cepat dibandingkan dengan mikrokontroler MCS51 yang memiliki arsitektur CISC (*complex instruction set compute),* mikrokontroler MCS51 membutuhkan 12 siklus *clock* untuk mengeksekusi 1 instruksi.

Mikrokontroler AVR juga memiliki fitur yang lengkap (ADC internal, EEPROM Internal, *Timer/Counter*, *watchdog Timer*, PWM, Port I/O, komunikasi serial, Komparator, I2C,dll), sehingga dengan fasilitas yang lengkap ini,

Progammer dan desainer dapat menggunakannya untuk berbagai aplikasi sistem elektronika seperti robot, otomasi industri, peralatan telekomunikasi dan berbagai keperluan lain. Secara umum mikrokontroler AVR dapat dikelompokkan menjadi 3 kelompok, yaitu keluarga AT90xx, ATMega dan ATtiny. Salah satu IC mikrokontroler yang sering digunakan adalah ATMEGA 16.

2.3.1.1 Konfigurasi Pin AVR ATMEGA 16



Gambar 2.5. Konfigurasi *pin* ATMEGA16.
Sumber: Atmel (2007)

Konfigurasi *pin* ATMEGA 16 dengan kemasan 40 *pin* DIP (*Dual Inline Package)* dapat dilihat pada gambar 6.1. dari gambar diatas dapat dijelaskan fungsi dari masing-masing *pin* ATMEGA 16 sebagai berikut:

1. VCC merupakan *pin* yang berfungsi sebagai masukan catu daya.
2. GND merupakan *pin Ground.*
3. *Port* A(PA0…PA7) merupakan *pin input/output* dua arah dan *pin* masukan ADC.
4. *Port* B (PB0…PB7) merupakan *pin input/output* dua arah dan *pin* Fungsi khusus PINB ATMega 16 ditunjukkan dalam Tabel 2.2.

Tabel 2.2 Fungsi Khusus *Port B*

Pin	**Fungsi Khusus**
PB7	SCK (SPI *Bus Serial Clock)*
PB6	MISO (SPI *Bus Master Input/Slave Output)*
PB5	MOSI (SPI *Bus Master Output/Slave Input)*

PB4 $\overline{SS}$ (SPI *Slave Select Input)*

PB3 AIN1 (*Analaog Comparator Negative Input)*

OC0 (*Timer/Counter0 Output Compare Match Output)*

PB2 AIN0 (*Analog Comparator Positive Input)*

INT2 (*External Interrupt 2 Input)*

PB1 T1 (*Timer/Counter1 External Counter Input)*

PB0 T0 T1 (*Timer/Counter0 External Counter Input)*

XCX (*USART External Clock Input/Output)*

---

5. *Reset* merupakan *pin* yang digunakan untuk mereset mikrokontroler
6. XTAL1 dan XTAL3 merupakan *pin* masukan *clock* eksternal.
7. AVCC merupakan *pin* masukan tegangan ADC.
8. AREF merupakan *pin* masukan tegangan referensi ADC.

2.3.2 Mikrokontroler Slave

AVR ATMega 8 adalah mikrokontroler CMOS 8-*bit* berarsitektur AVR RISC yang memiliki 8K *byte in-System Programmable Flash*. Mikrokontroler dengan konsumsi daya rendah ini mampu mengeksekusi instruksi dengan kecepatan maksimum 16MIPS pada frekuensi 16MHz. Jika dibandingkan dengan ATmega8L perbedaannya hanya terletak pada besarnya tegangan yang diperlukan untuk bekerja. Untuk ATMega 8 tipe L, mikrokontroler ini dapat bekerja dengan tegangan antara 2,7 - 5,5 V sedangkan untuk ATMega 8 hanya dapat bekerja pada tegangan antara 4,5 – 5,5 V.

2.3.2.1 Konfigurasi Pin ATMega 8

Gambar 2.6. Konfigurasi Pin Atmega8.
Sumber: Atmel (2006)

ATmega8 memiliki 28 Pin, yang masing-masing pin nya memiliki fungsi yang berbeda-beda baik sebagai port maupun fungsi yang lainnya. Berikut akan dijelaskan fungsi dari masing-masing kaki ATmega8.

- VCC Merupakan *supply* tegangan digital.
- GND Merupakan ground untuk semua komponen yang membutuhkan grounding.
- Port B (PB7...PB0)

  Didalam Port B terdapat XTAL1, XTAL2, TOSC1, TOSC2. Jumlah Port B adalah 8 buah pin, mulai dari pin B.0 sampai dengan B.7. Tiap pin dapat digunakan sebagai *input* maupun *output*. Port B merupakan sebuah 8-bit *bi-directional* I/O dengan internal pull-up resistor. Sebagai *input*, pin-pin yang terdapat pada port B yang secara eksternal diturunkan, maka akan mengeluarkan arus jika *pull-up* resistor diaktifkan. Khusus PB6 dapat digunakan sebagai *input* Kristal (*inverting oscillator amplifier*) dan *input* ke rangkaian *clock* internal, bergantung pada pengaturan *Fuse bit* yang digunakan untuk memilih sumber *clock*. Sedangkan untuk PB7 dapat digunakan sebagai *output* Kristal (*output oscillator amplifier*) bergantung pada pengaturan *Fuse bit* yang digunakan untuk memilih sumber *clock*. Jika sumber *clock* yang dipilih dari *oscillator internal*, PB7 dan PB6 dapat digunakan sebagai I/O atau jika menggunakan *Asyncronous Timer/Counter2* maka PB6 dan PB7 (TOSC2 dan TOSC1) digunakan untuk saluran *input timer*.

- Port C (PC5…PC0)

  Port C merupakan sebuah 7-*bit bi-directional* I/O port yang di dalam masingmasing pin terdapat *pull-up* resistor. Jumlah pin nya hanya 7 buah mulai dari *pin* C.0 sampai dengan *pin* C.6. Sebagai keluaran/*output port* C memiliki karakteristik yang sama dalam hal menyerap arus (*sink*) ataupun mengeluarkan arus *(source*).

- RESET/PC6

  Jika RSTDISBL *Fuse* diprogram, maka PC6 akan berfungsi sebagai *pin* I/O. *Pin* ini memiliki karakteristik yang berbeda dengan *pin-pin* yang terdapat pada *port* C lainnya. Namun jika RSTDISBL *Fuse* tidak diprogram, maka pin ini akan berfungsi sebagai input reset. Dan jika *level* tegangan yang masuk ke pin ini rendah dan pulsa yang ada lebih pendek dari pulsa minimum, maka akan menghasilkan suatu kondisi reset meskipun *clock*-nya tidak bekerja.

- Port D (PD7…PD0)

  Port D merupakan 8-*bit bi-directional* I/O dengan internal *pull-up* resistor. Fungsi dari port ini sama dengan port-port yang lain. Hanya saja pada port ini tidak terdapat kegunaan-kegunaan yang lain. Pada *port* ini hanya berfungsi sebagai masukan dan keluaran saja atau biasa disebut dengan I/O.

- AVcc

  Pin ini berfungsi sebagai *supply* tegangan untuk ADC. Untuk *pin* ini harus dihubungkan secara terpisah dengan VCC karena *pin* ini digunakan untuk analog saja. Bahkan jika ADC pada AVR tidak digunakan tetap saja disarankan untuk menghubungkannya secara terpisah dengan VCC. Jika ADC digunakan, maka AVcc harus dihubungkan ke VCC melalui *low pass filter*.

- AREF

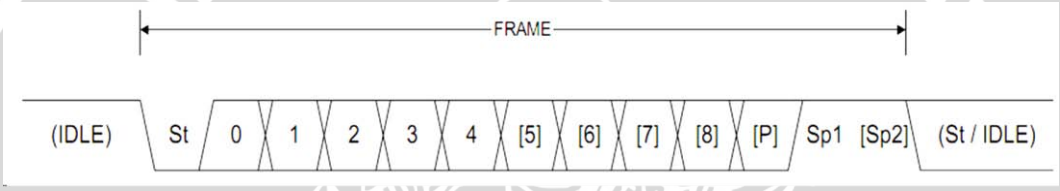
  Merupakan pin referensi jika menggunakan ADC.

2.4 Komunikasi UART

Pada prinsipnya, komunikasi serial ialah komunikasi dengan pengiriman data yang dilakukan per bit, sehingga lebih lambat dibandingkan komunikasi pararel seperti pada port printer yang mampu mengirim 8 bit sekaligus dalam sekali waktu. Devais pada komunikasi serial port dibagi menjadi 2 (dua) kelompok yaitu *Data Communication Equipment* (DCE) dan *Data Terminal Equipment*

(DTE). Pengiriman data secara serial dapat berupa sinkron, yaitu pengiriman clock dilakukan bersamaan dengan data, atau berupa asinkron, yaitu pengiriman clock tidak bersamaan dengan data namun secara dua tahap, saat data dikirim dan data diterima. Untuk istilah yang sering digunakan untuk mengirim dan menerima data secara asinkron biasa disebut *Universal Asynchronous Receiver Transmitter* (UART). Komunikasi data serial menggunakan UART sangat umum dan mudah penggunaannya, misalnya pada port serial PC. Pada UART jalur pengiriman dan penerimaan data serial dipisahkan.

Setiap pengiriman data pada UART menggunakan bit tanda *start* bit dan *stop* bit. Jalur data yang digunakan hanya satu untuk setiap pengiriman data. Data-data serial dikirim melewati jalur data satu persatu setiap satuan waktu. Format pengiriman data serial secara asinkron ditunjukkan dalam Gambar 2.17.



Gambar 2.7 Format frame data serial.
Sumber: Atmel (2007)

Dengan :

St = *Bit start* selalu berlogika rendah

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

P = *Bit* paritas (ganjil atau genap)

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

IDLE = Tidak ada data yang ditransfer pada RX dan TX, IDLE selalu berlogika tinggi

2.5 Komunikasi Serial Peripheral Interface (SPI)

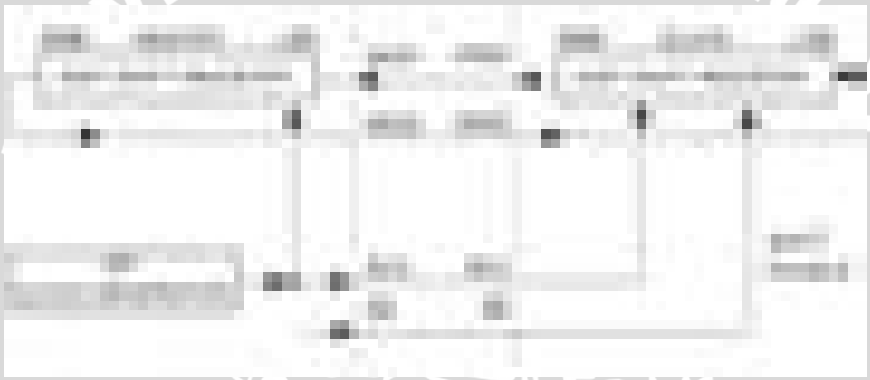
Serial Peripheral Interface (SPI) merupakan salah satu mode komunikasi serial *synchronous* kecepatan tinggi yang dimiliki oleh ATMega. Komunikasi SPI membutuhkan 3 jalur (*wire*) yaitu MOSI, MISO dan SCK. Melalui komunikasi SPI ini data dapat saling dikirimkan baik antar mikrokontroler maupun antar mikrokontroler dengan peripheral lain di luar mikrokontroler.

- MOSI : *Master Output Slave Input*

Artinya jika dikonfigurasi sebagai *master* maka pin MOSI ini sebagai *output* tetapi jika dikonfigurasi sebagai *slave* maka pin MOSI ini sebagai input.

- MISO : *Master Input Slave Output*
  Berkebalikan dengan MOSI diatas, jika dikonfigurasi sebagai *master* maka pin MISO ini sebagai input tetapi dikonfigurasi sebagai *slave* maka pin MISO ini sebagai output.
- CLK : *Clock*
  Jika dikonfigurasi sebagai *master* maka pin CLK berlaku sebagai output (pembangkit *clock*) tetapi jika dikonfigurasi sebagai *slave* maka pin CLK berlaku sebagai input (menerima sumber *clock* dari *master*).

Gambar 2.18 menunjukkan interkoneksi SPI antara *master* dan *slave*.

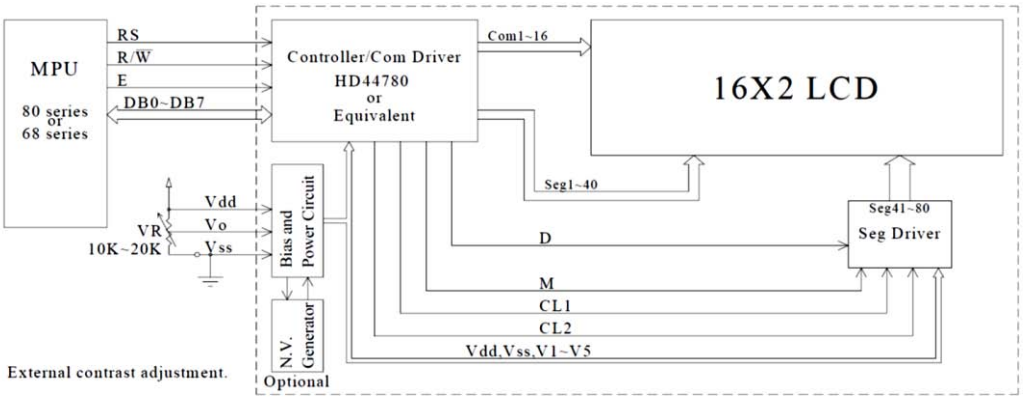


Gambar 2.8 Interkoneksi SPI *Master-Slave*.
Sumber: Atmel (2006)

Pengaturan konfigurasi *master* atau *slave* ditentukan oleh pin SS. Jika pin SS diberi tegangan *high* (‘1’) maka terkonfigurasi sebagai *master* dan jika pin SS diberi tegangan *low* (‘0’) maka terkonfigurasi sebagai *slave*. Untuk mengatur mode kerja komunikasi SPI ini dilakukan dengan menggunakan *register* SPCR, SPSR dan SPDR.

2.6 LCD (Liquid Crystal Display)

HD44780 adalah modul LCD *(liquid crystal display)* matrix dengan konfigurasi 16 karakter dan 2 baris dengan setiap karakternya. Dalam 4 bit antarmuka hanya membutuhkan 4 pin data komunikasi data parallel, pin11 sanpai pin 14 yang dikoneksikan dengan pengendali. Blok diagram ditunjukkan dalam Gambar 2.9.



Gambar 2.9 LCD (Liquid Crystal Display) HD44780.
Sumber: Weinstar (2009)

Liquid Crystal Display atau biasa disebut LCD merupakan alat tampilan yang biasa digunakan untuk menampilkan karakter ASCII sederhana, dan gambar-gambar pada alat-alat digital seperti jam tangan, kalkulator dan lain lain. Deskripsi sederhana cara kerja dari sebuah LCD matrix adalah sebuah *Twisted Nematic* (TN) *Liquid Crystal Display*, yang terdiri dari 2 material yang terpolarisasi, 2 buah kaca, sebuah bentuk elemen elektroda untuk menentukan pixel, dan Integrated Circuit (IC) untuk mengalamatkan baris dan kolom.

Perlu diketahui fungsi dari setiap kaki yang ada pada komponen tersebut.

1) Kaki 1 (GND) : Kaki ini berhubungan dengan tegangan 0 volt (Ground).

2) Kaki 2 (VCC) : Kaki ini berhubungan dengan tegangan +5 Volt yang merupakan tegangan untuk sumber daya.

3) Kaki 3 (VEE/VLCD) : Tegangan pengatur kontras LCD, kaki ini terhubung pada *output* variabel resistor. Kontras mencapai nilai maksimum pada saat kondisi kaki ini pada tegangan 0 volt.

4) Kaki 4 (RS) : Register Select, kaki pemilih register yang akan diakses. Untuk akses ke Register Data, logika dari kaki ini adalah 1 dan untuk akses ke Register Perintah, logika dari kaki ini adalah 0.

5) Kaki 5 (R/W) : Logika 1 pada kaki ini menunjukan bahwa modul LCD sedang pada mode pembacaan dan logika 0 menunjukan bahwa modul LCD sedang pada

mode penulisan. Untuk aplikasi yang tidak memerlukan pembacaan data pada modul LCD, kaki ini dapat dihubungkan langsung ke Ground.

6) Kaki 6 (E) : Enable Clock LCD, kaki mengaktifkan clock LCD. Logika 1 pada kaki ini diberikan pada saat penulisan atau pembacaan data.

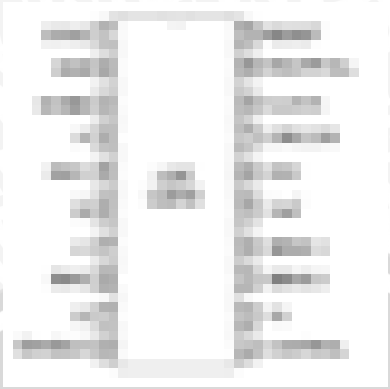
7) Kaki 7 – 14 (D0 – D7) : Data bus, kedelapan kaki LCD ini adalah tempat aliran data sebanyak 4 bit ataupun 8 bit mengalir saat proses penulisan maupun pembacaan data.

8) Kaki 15 (Anoda) : Berfungsi untuk tegangan positif dari *backlight* LCD sekitar 4,5 volt (hanya terdapat untuk LCD yang memiliki back light).

9) Kaki 16 (Katoda) : Tegangan negatif back light LCD sebesar 0 volt (hanya terdapat pada LCD yang memiliki *backlight*).

2.7 *Driver* Motor Stepper

Driver motor *stepper* yang digunakan adalah menggunakan IC L297 yang dapat mengontrol motor *stepper bipolar* dua fasa ataupun *unipolar* empat fasa. Dalam IC L297 ini dapat mengontrol kecepatan motor, arah putaran motor, *half mode* ataupun *full mode.* Untuk mengatur kecepatan motor digunakan sumber *clock* yang diubah frekuensinya sesuai dengan kecepatan yang diinginkan. Sedangkan untuk arah dan mode pergerakan motor *stepper* digunakan *pin* mikrokontroler yang mengaktifkan atau tidak mode tersebut. Gambar konfigurasi pin IC L297 ditunjukkan dalam gambar 2.10



Gambar 2.10 Konfigurasi Pin IC L297.
Sumber: ST (2001)

Penjelasan konfigurasi pin IC L297 ditunjukkan dalam Tabel 2.3.

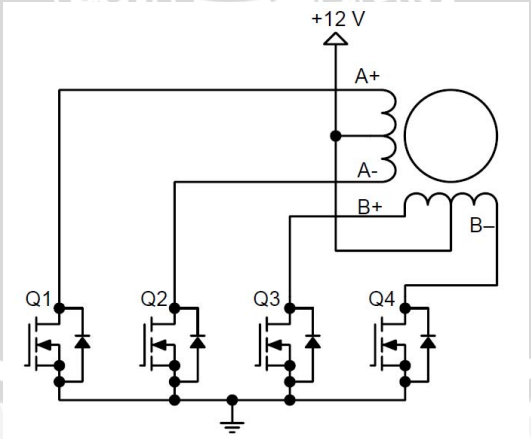
Tabel 2.3 Penjelasan Konfigurasi Pin IC L297

Nama Pin	Fungsi
SYNC	*Output* dari oscillator internal di dalam L297. JIka digunakan lebih dari satu L297 maka pin SYNC akan dihubungkan dengan pin SYNC pada IC L297 yang lain
GND	*Ground*
HOME	Pin ini akan mengeluarkan logika *high* jika motor *stepper* sudah kembali ke posisi semula
A	Sinyal *drive* fasa A
INH1	Kontrol *inhibit* bersifat aktif *low* untuk fasa motor A dan B
B	Sinyal *drive* fasa B
C	Sinyal *drive* fasa C
INH2	Kontrol *inhibit* bersifat aktif *low* untuk fasa motor C dan D
D	Sinyal *drive* fasa D
ENABLE	Jika pin ini mendapatkan logika *low* maka INH1, INH2, A, B, C, D juga akan *low*
CONTROL	*Input* kontrol yang menentukan aksi pemotong
Vs	Tegangan *Supply* sebesar 5 V
SENS2	*Input* untuk me-*load* tegangan saat ini dari fasa motor C dan D
SENS1	*Input* untuk me-*load* tegangan saat ini dari fasa motor A dan B
Vref	Tegangan referensi untuk rangkaian pemotong

OSC	Rangkaian RC dengan menghubungkan hambatan ke Vcc dan kapasitor ke *ground* terhubung pada pin ini untuk menentukan nilai dari pemotong
CW/CCW	Input kontrol arah pergerakan *motor stepper*. Jika *high* maka searah jarum jam dan jika *low* akan berlawanan jarum jam
CLOCK	*Clock* untuk setiap pergerakan *motor stepper*. Step terjadi ketika pulsa berubah dari *low* menjadi *high*
HALF/FULL	Jika logika *high* maka akan dipilih mode *half step*, dan bila logika *low* akan dipilih mode *full step*

2.8 Motor Stepper

Motor *stepper* adalah kategori motor *brushless,* yaitu motor elektrik yang bisa membagi rotasi penuh menjadi ke dalam sejumlah besar langkah. Motor ini bekerja dengan prinsip *elektromagnetisme*. *Rotor* berasal dari magnet besi lunak yang dikelilingi oleh stator elektromagnetik. Prinsip kerja motor ini dengan mengubah pulsa elektronis menjadi gerakan mekanis diskrit. Motor *stepper* bergerak berdasarkan urutan pulsa yang diberikan kepada motor. Karena itu, untuk menggerakkan motor *steper* diperlukan pengendali motor *stepper* yang membangkitkan pulsa periodic (Silabs,2001). Skema penegndalian motor *stepper* ditunjukkan dalam Gambar 2.11.



Gambar 2.11 Skema Pengendalian Motor Steper Secara Umum.
Sumber: Silabs (2001)

Penggunaan *motor stepper* memiliki beberapa keunggulan dibandingkan dengan penggunaan motor DC biasa. Keunggulannya antara lain :

1. Sudut rotasi motor proporsional dengan pulsa masukan sehingga lebih mudah diatur
2. Motor dapat langsung memberikan *torsi* penuh pada saat mulai bergerak
3. Posisi dan pergerakan repetisinya dapat ditentukan secara presisi
4. Memiliki respon yang sangat baik terhadap mulai, stop dan berbalik (perputaran)
5. Sangat *realibel* karena tidak adanya sikat yang bersentuhan dengan *rotor* seperti pada motor DC
6. Dapat menghasilkan perputaran yang lambat sehingga beban dapat dikopel langsung ke porosnya
7. Frekuensi perputaran dapat ditentukan secara bebas dan mudah pada range yang luas.

Berdasarkan metode perancangan rangkain pengendalinya, *motor stepper* dapat dibagi menjadi jenis *unipolar* dan *bipolar*. Rangkaian pengendali *motor stepper unipolar* lebih mudah dirancang karena hanya memerlukan satu switch / transistor setiap lilitannya. Untuk menjalankan dan menghentikan motor ini cukup dengan menerapkan *pulsa* digital yang hanya terdiri atas tegangan positif dan nol (*ground*) pada salah satu terminal lilitan (*wound*) motor sementara terminal lainnya dicatu dengan tegangan positif konstan (V_M) pada bagian tengah (*center tap*) dari lilitan

BAB III
METODE PENELITIAN

Kajian dalam skripsi ini merupakan penelitian yang bersifat *aplikatif*, yaitu pembuatan *mobile* robot penggerak mekanum dengan perhitungan kinematik yang bertujuan menghasilkan robot dengan mobilitas tinggi. Dalam merealisasikan *mobile* robot yang akan dibuat terdapat beberapa hal yang dilakukan, di antaranya:

1. Studi literatur
2. Spesifikasi *mobile* robot
3. Perancangan sistem
4. Pengujian *mobile* robot
5. Pengambilan kesimpulan

3.1 Studi Literatur

Perencanaan sistem dilakukan sebagai langkah awal sebelum terbentuknya suatu sistem beserta rangkaian elektronik pendukungnya, hal ini dimaksudkan agar sistem pengindentifikasi dapat berjalan sesuai dengan yang telah direncanakan. Studi literatur untuk perencanaan *mobile* robot meliputi:

- Komunikasi data *master slave* menggunakan mode SPI
- Persamaan kinematika roda mekanum
- Komunikasi data serial UART
- Motor *stepper*

3.2 Penentuan Spesifikasi *Mobile* Robot

Spesifikasi *mobile* robot secara umum ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi *mobile* robot yang direncanakan adalah sebagai berikut:

1. Roda mekanum dengan diameter 100mm
2. ATMega 16 sebagai *master* dan ATMega 8 sebagai *slave*
3. LCD sebagai tampilan mode pergerakan dan kecepatan roda
4. Push button sebagai *input* mode
5. *Driver* motor menggunakan L297
6. Motor *stepper* 1,8°

3.2.1 Perancangan Sistem

Perancangan sistem dibagi menjadi tiga bagian diantaranya perancangan mekanik, perancangan perangkat keras, dan perancangan perangkat lunak.

3.2.1.1 Perancangan Mekanik

Perancangan mekanik robot pertama dilakukan desain robot. Dalam desain robot diperlukan perhitungan jarak antar sumbu roda, *couple* roda mekanum, serta penempatan titik beban pada robot. Perancangan desain robot dilakukan menggunakan aplikasi 3ds Max 2012.

3.2.1.2 Perancangan dan Pembuatan Perangkat Keras (*Hardware*)

Perancangan perangkat keras pertama dilakukan dengan menyusun blok diagram perangkat keras kemudian melakukan perancangan dan analisis perhitungan elektrik rangkaian. *Layout* skematik rangkaian menggunakan aplikasi Eagle 5.11.0. Perancangan perangkat keras robot diantaranya:

1) *Push button* digunakan untuk *utilitas* dan *input* mode pergerakan dalam V_x, V_y dan ω.
2) Mikrokontroler yang digunakan adalah ATMega 16 sebagai *master* dan ATMega 8 sebagai *slave*. Atmega 8 bertugas untuk mengubah kecepatan sudut roda menjadi jumlah pulsa yang memiliki variabel frekuensi *clock* untuk motor, sedangkan ATMega 16 sebagai *master* untuk perhitungan kinematik mekanum**.**
3) LCD 2x16 yang digunakan untuk tampilan pilihan mode pergerakan yang akan digunakan dan hasil konnversi ke dalam V_x, V_y dan ω.
4) *Driver motor* menggunakan IC L297 sebagai kontrol logika motor dan IC STK6713BMK4 sebagai *driver* fasa motor.
5) Motor yang digunakan adalah motor *stepper* tipe *hybrid* dengan sudut langkah 1,8° per langkah.

3.2.1.3 Perancangan Perangkat Lunak

Perancangan perangkat lunak berupa *flowchart* (diagram alir) bahasa pemograman yang dipakai untuk ATMega16. Bahasa pemograman yang digunakan adalah bahasa C yang nantinya dibuat dan *dicompile* menggunakan *software codevision* AVR. Program terbagi menjadi dua bagian, yaitu *master* dan *slave*.

Mikrokontroler *master* bertugas untuk menerjemahkan mode pergerakan dari *input push button*. Mode pergerakan diterjemahkan dalam sumbu pergerakan robot diantaranya V_x, V_y, ω. Mikrokontroler *master* kemudian mengolah data *input* dalam persamaan kinematika robot yang dihasilkan kecepatan sudut masing-masing roda. Hasil perhitungan kinematik ditampilkan dalam LCD 2x16. Kecepatan sudut masing-masing roda dikirim menuju mikrokontroler *slave* melalui komunikasi SPI.

Mikrokontroler *slave* bertugas untuk menerima data kecepatan sudut roda dari *master* dengan komunikasi SPI. Kecepatan sudut masing-masing roda dikonversi menjadi frekuensi pulsa. Frekuensi pulsa digunakan untuk logika kecepatan motor *stepper*.

3.3 Pengujian *Mobile* Robot

Pengujian *mobile* robot dilakukan untuk memastikan agar sistem berjalan sesuai dengan perancangan. Pengujian *mobile* robot meliputi pengujian perangkat keras dan perangkat lunak.

1. Pengujian Motor *Stepper*

Pengujian motor *stepper* dibagi menjadi dua pengujian diantaranya pengujian sudut putaran motor dan pengujian kecepatan motor. Pengujian sudut putaran motor dilakukan untuk mengetahui perbandingan atara jumlah pulsa dengan sudut putaran roda. Pengujian sudut putaran motor *stepper* dilakukan dengan menggunakan penggaris busur untuk mengetahui sudut putaran motor. Pengambilan data sudut putaran motor dilakukan sebanyak tiga kali dalam setiap pemberian jumlah pulsa.

Pengujian kecepatan motor *stepper* dilakukan untuk mengetahui pengaruh besar frekuensi pulsa dengan kecepatan motor. Pengujian dilakukan dengan memberikan pulsa ke *driver* motor *stepper* dengan *pulse generator* dan membandingkan dengan kecepatan motor yang diukur lewat *tachometer*. Pengujian dilakukan dengan memberikan frekuensi pulsa yang berbeda dan mengukur kecepatan motor *stepper* sebanyak tiga kali.

2. Pengujian Komunikasi Data SPI

Pengujian komunikasi SPI dilakukan untuk membandingkan dan menjamin atara data yang dikirim dengan data yang diterima sesuai. Dalam Pengujian ini

dilakukan untuk pengiriman data dari *slave* menuju *master* ataupun sebaliknya. Pengujian dilakukan dengan mengirim 8 bit data dan ditampilkan pada komputer melalui UART.

3. Pengujian Perhitungan Kinematik Mekanum

Dalam pengujian perhitungan kinematik ini kita menyesuaikan perhitungan yang didapatkan dari penurunan secara teoritis yang diimplementasikan dalam mikrokontroler. Hasil yang diperoleh dilakukan pencocokan kembali dengan perhitungan awal yang ditampilkan melalui LCD 2x16. Pengujian kinematik juga akan dicocokan dengan kecepatan roda sebenarnya dalam satuan rpm.

4. ` Pengujian Sistem Secara Keseluruhan

Pengujian ini dilakukan dengan cara menggabungkan semua bagian *mobile* robot yang dibuat dan melihat kinerja *mobile* robot baik perangkat keras maupun perangkat lunak. Pengujian ini bertujuan untuk mengetahui kinerja *mobile* robot yang dibuat dan memberikan analisa terhadap kinerja *mobile* robot. *Mobile* robot melaksanakan perintah mode sesuai dengan sudut yang ditetapkan diantaranya sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0 . Pengujian keseluruhan ini ditujukan untuk mengetahui ada atau tidaknya sudut simpangan dari robot saat melaksanakan perintah mode pergerakan tertentu.

Pengambilan data dilakukan dengan mengukur sudut simpangan dengan busur penggaris. Pengambilan data dilakukan sebanyak tiga kali dari masing-masing sudut pergerakan.

3.4 Pengambilan Kesimpulan

Kesimpulan diambil berdasarkan data yang diperoleh dari pengujian sistem secara keseluruhan. Jika hasil yang didapatkan telah sesuai dengan yang direncanakan sebelumnya, maka *mobile* robot tersebut telah berhasil memenuhi harapan dan tentunya memerlukan pengembangan lebih lanjut untuk penyempurnaan.

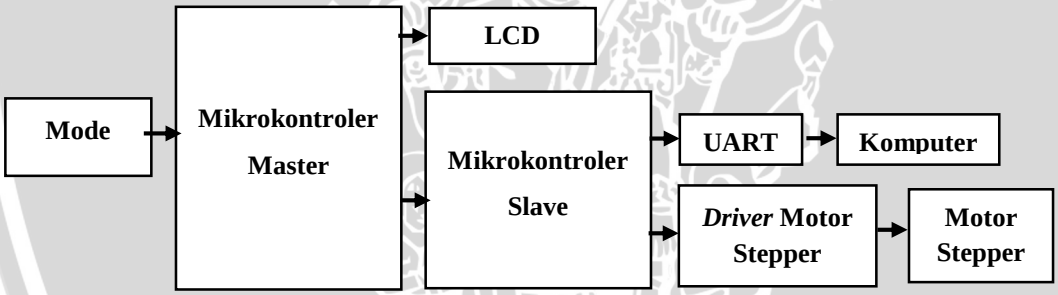
BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Perancangan *mobile* robot roda mekanum ini dilakukan secara bertahap sehingga akan memudahkan dalam analisis pada setiap blok maupun secara keseluruhan. Perancangan ini terdiri dari :

- Perancangan Mekanik *Mobile* Robot.
- Perancangan Perangkat Keras
- Perancangan Komunikasi Mikrokontroler ATMega 16 dan ATMega 8
- Perancangan Sistem Kinematika *Mobile Robot* Roda Mekanum
- Perancangan Perangkat Lunak.

4.1 Perancangan Sistem Robot

Perancangan sistem keseluruhan robot ditunjukkan dalam Gambar 3.1.



Gambar 3.1 Diagram Blok Perancangan Sistem Secara Keseluruhan.

Awalnya mikrokontroler *master* akan mengolah *input* dari tombol mode yang berupa *Vx, Vy dan* ω. Mikrokontroler *master* kemudian akan menghitung persamaan kinematik robot sehingga menghasilkan $\omega_1, \omega_2, \omega_3, \omega_4$ dan arah putaran roda mekanum. Mikrokontroler *master* mengirim data kecepatan dan arah kepada mikrokontroler *slave* menggunakan komunikasi SPI. Mikrokontroler *slave* menerjemahkan data kecepatan sudut dan arah menjadi frekuensi pulsa dan arah motor yang diberikan kepada *driver* motor *stepper*.

4.2 Perancangan Mekanik Robot

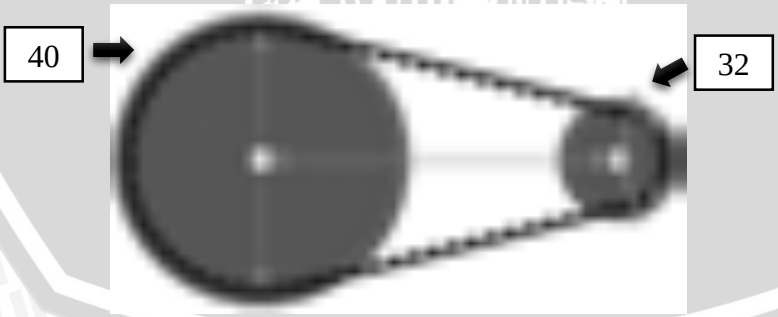
Sistem mekanik yang baik berpengaruh besar pada pergerakan robot, oleh karena itu perancangan mekanik dalam hal ini rangka robot haruslah dibuat sepresisi mungkin. Desain mekanik *mobile* robot ditunjukkan dalam Gambar 4.1

(a) (b)

Gambar 4.1 (a) Rancangan Robot Tampak Atas, (b) Rancangan Robot Tampak Perspektif.

Perancangan dan pembuatan *mobile* robot menggunakan *chasis* dengan bahan alumunium dengan ukuran 20mm x 20mm. *Chasis* robot berbentuk segi empat dengan ukuran luar 500mm x 450mm dan panjang antar sumbu roda 340mm. Hal yang perlu diperhatikan dalam pembuatan mekanik *mobile robot* ini adalah jarak sumbu rodanya dan penempatan titik beban pada robot.

Perancangan mekanik digunakan *timming belt* untuk menghubungkan antara as motor dengan as roda dengan perbandingan jumlah gigi *pulley* motor dan *pulley* pada roda 8:10. Hal tersebut perpengaruh pada penambahan *torsi* robot. Perancangan *couple pulley* ditunjukkan dalam Gambar 4.2



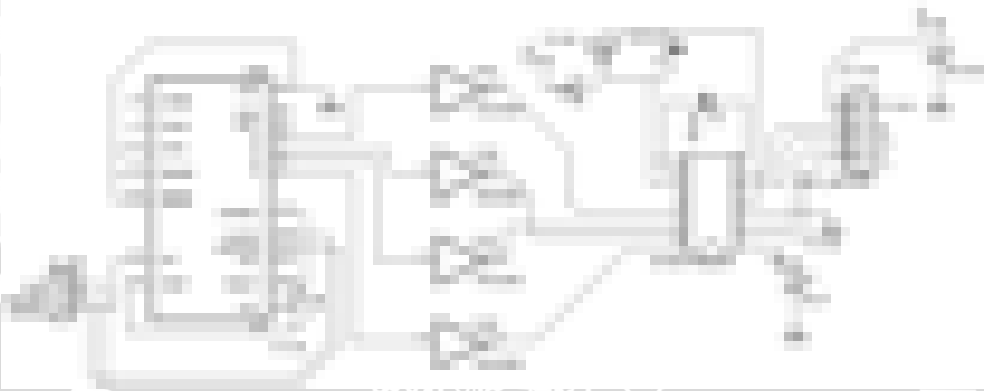
Gambar 4.2 Perancangan *Couple Pulley* Roda.

4.3 Perancangan Perangkat Keras

4.3.1.1 Perancangan Rangkaian *Driver Motor Stepper*

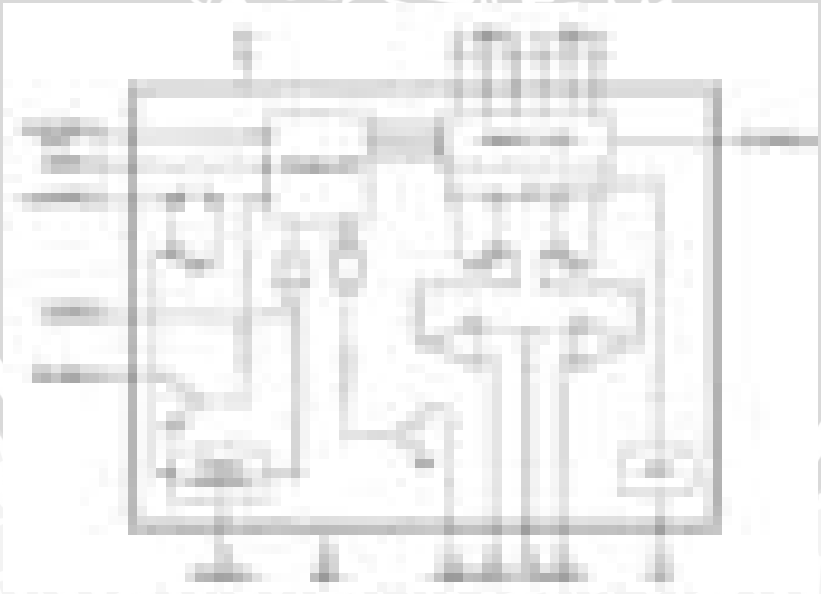
Pada perancangan *driver* motor *stepper* menggunakan IC L297 sebagai kontrol logika motor dan IC STK6713BMK4 sebagai *driver* fasa motor *stepper*. IC

L297 menerima *input* berupa pulsa yang diatur frekuensinya oleh mikrokontroler dan arah putaran motor yang diwakili oleh pin CW/CCW. *Output* dari L297 adalah 4 buah pin logika fasa A, B, C, D. *Output* IC L297 dihubungkan ke IC STK6713BMK4 sebagai *driver* fasa untuk mendapatkan tegangan catu 12 V dan 0 V untuk logika 0 dan 1. Skema rangkaian *driver* motor *stepper* ditunjukkan dalam Gambar 4.3.



Gambar 4.3 Rangkaian Keseluruhan *Driver* Motor.

Rangkaian IC L297 membutuhkan *input* pulsa dari mikrokontroler. Pulsa diatur frekuensinya yang berbanding lurus dengan kecepatan motor *stepper*. *Output* IC L297 berupa logika fasa A, B, C, D. Diagram blok IC L297 ditunjukkan dalam Gambar 4.4.



Gambar 4.4 Diagram Blok IC L297.
Sumber: ST(2001)

Berdasarkan Gambar 4.4 dapat dijelaskan bahwa *input* IC L297 yang digunakan dalam perancangan ini adalah pin *direction*, *clock*, dan *half/full* . *Output*

yang digunakan pada perancangan adalah A, B, C, D. Pin *clock* tersambung ke *D flip flop* dan *translator* sebagai sumber *clock.* Prinsip kontrol arah putaran pada IC L297 yaitu ketika *input D flip flop* (pin *DIR*) adalah logika *high*, maka *output D flip flop* adalah 1 yang berarti motor berputar ke kanan. Apabila pin DIR berlogika 0 maka *output D flip flop* adalah 0 dan motor berputar ke kiri.

Pada blok *translator, input* logika adalah pin *half/full, clock, reset,* dan *DIR*. Dalam perancangan pin *reset* disambung ke Vcc karena *reset* aktiv logika *low*. Tugas translator untuk melakukan kontrol logika pin *output* A, B, C, D dengan prinsip kerja sama dengan rangkaian *register* geser yang menggeser logika dengan kontrol arah pin *DIR* dan kontrol mode step pin *half/full.*

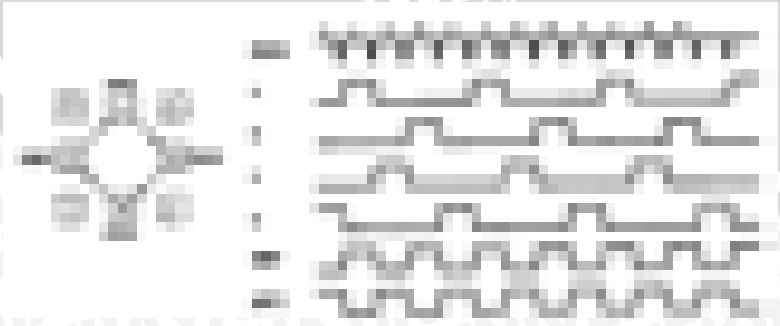
IC L297 mempunyai spesifikasi *input* :

- V_{IL} = 0,6 V
- V_{IH} = 2 V
- I_{IL} = 100 uA
- I_{IH} = 10 uA

IC L297 mempunyai spesifikasi *output* :

- V_{OL} = 0,4 V
- V_{OH} = 3,9 V
- I_{OL} = 10 mA
- I_{OH} = 5 mA

Output dari L297 berupa empat pin logika diantaranya A, B, C, D yang aktiv pada logika *high*. *Timing* diagram *output* IC L297 ditunjukkan dalam Gambar 4.5.



Gambar 4.5 *Timming* Diagram *Output* IC L297
Sumber: ST(2001)

Output fasa L297 sebagai *input* dari IC STK6713BMK4 sebagai *driver* fasa motor *stepper*. *Input* IC STK6713BMK4 aktif pada logika *low* sehingga diperlukan pembalik logika pada IC 74LS04.

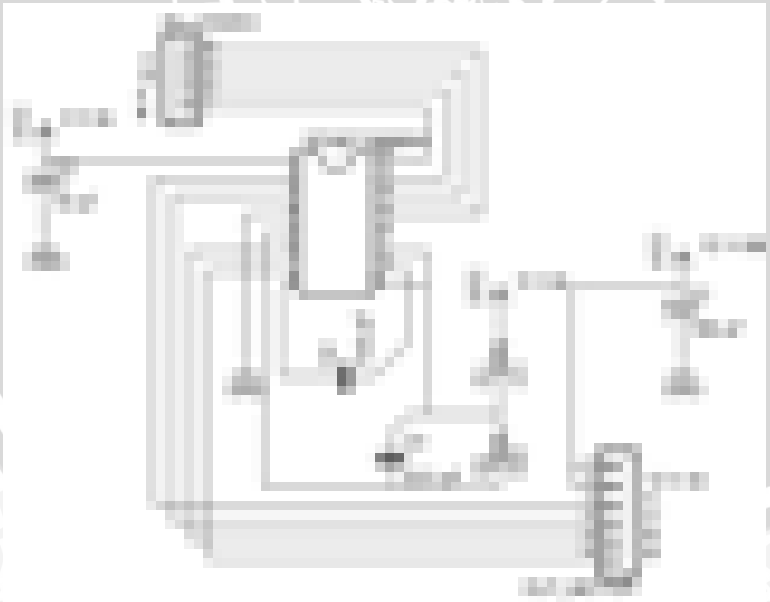
IC 74LS04 memiliki karakteristik *input* sebagai berikut :

- V_{IL} = 0.8 V
- V_{IH} = 2 V
- I_{IH} = 20 uA
- I_{IL} = -0,4mA

IC 74LS04 memiliki karakteristik *output* sebagai berikut :

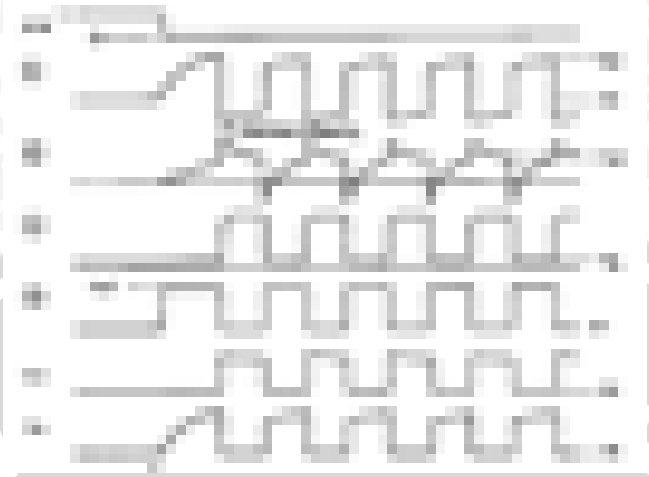
- V_{OL} = 0.5 V
- V_{OH} = 2.7 V
- I_{OL} = 8 mA
- I_{OH} = -0,4 mA.

Pada IC STK6713BMK4 mempunyai karakteristik *input* V_{IL} = 0.8 V dan V_{IH} = 2 V. IC ini sebagai *driver* fasa motor *stepper*. IC ini dilengkapi dengan *current chopper* untuk memotong arus saat motor pada keadaan *lock*. Skema rangkaian IC STK6713BMK4 ditunjukkan dalam Gambar 4.6.



Gambar 4.6 Skema Rangkaian IC STK6713BMK4.
Sumber: Sanyo (2006)

IC STK6713BMK4 mempunyai karakteristik *output* I_{OH} maksimal 3.9 A dan tegangan catu motor 52 V dan tegangan catu rangkaian 5 V. *Timming* diagram *output* IC STK6713BMK4 ditunjukkan dalam Gambar 4.7.



Gambar 4.7 Timming Diagram IC STK6713BMK4.
Sumber: Sanyo (2006)

IC STK6713BMK4 mempunyai sistem rangkaian yang terdiri dari *driver* fasa, *comparator*, dan sensor arus pada fasa. Rangkaian pengganti IC STK6713BMK4 ditunjukkan dalam Gambar 4.8.

Gambar 4.8 Rangkaian pengganti IC STK6713BMK4.
Sumber: Sanyo (2006)

Fasilitas *current chopper* dapat dijelaskan dalam Gambar 4.8. Proses pemotongan arus pada fasa dimulai dari arus yang melewati R5 akan mengisi kapasitor C1, tegangan dibandingkan pada *comparator* kemudian masuk dalam gerbang NOR antara *output comparator* dan *input* logika fasa dari L297. Sebagai pemotong digunakan Vref yang berasal dari pembagi tegangan R_1 dan R_2 pada gambar 4.6 . Vref tersebut menentukan berapa arus yang akan dipotong saat keadaan lilitan fasa motor menyerap banyak arus saat keadaan motor mengunci.

Persamaan perhitungan *current chopper* IC STK6713BMK4 :

$$Vref = \frac{R2}{R1+R2} xVcc \quad (4\text{-}1)$$

$$I_{OH} = K \times \frac{R2}{R1+R2} \times \frac{Vcc}{R7} \quad (4\text{-}2)$$

Dengan:

K = 1,3

R_7 = 0,33 ±3%

Maka dapat dihitung *current chopper* untuk perancangan rangkaian

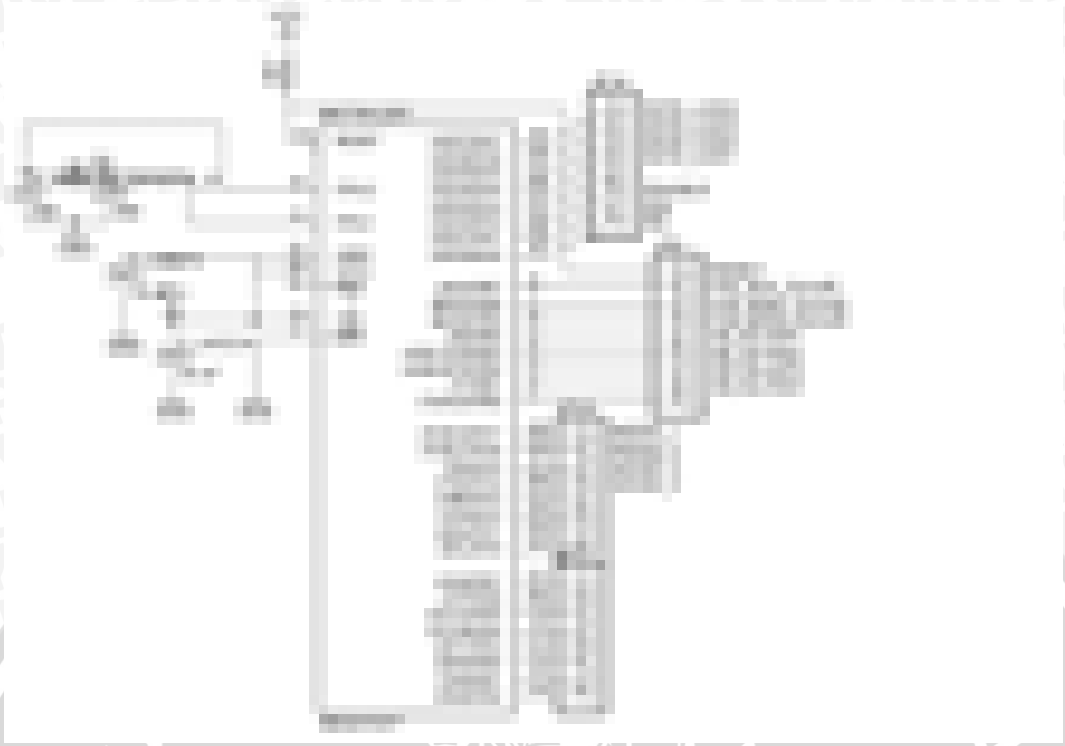
$$I_{OH} = K \times \frac{R2}{R1+R2} \times \frac{Vcc}{R7}$$

$$I_{OH} = 1,3 \times \frac{330}{330+2\,k} \times \frac{5}{0.33}$$

$$I_{OH} = 2.,8 \text{ A}$$

4.3.1.2 Sistem Mikrokontroler Master dan Slave

Pada Perancangan Mikrokontroler *mobile* robot roda mekanum ini menggunakan lima buah mikrokontroler. Sebuah mikrokontroler *master* dan empat buah mikrokontroler *slave*. Penggunaan empat buah mikrokontroler ini bertujuan agar terdapat pembagian tugas yang merata untuk masing – masing mikrokontroler sehingga proses pengolahan datanya lebih cepat. Mikrokontroler *master* menggunakan ATMega16 dan mikrokontroler *slave* menggunakan ATMega8. Mikrokontroler master bertugas untuk melakukan perhitungan kecepatan tiap motor sesuai dengan sudut gerak robot yang diinginkan sedangkan mikrokontroler *slave* bertugas untuk mengolah data kecepatan motor masing masing roda. Skema perancangan mikrokontroler ATMega 16 ditunjukkan dalam Gambar 4.9



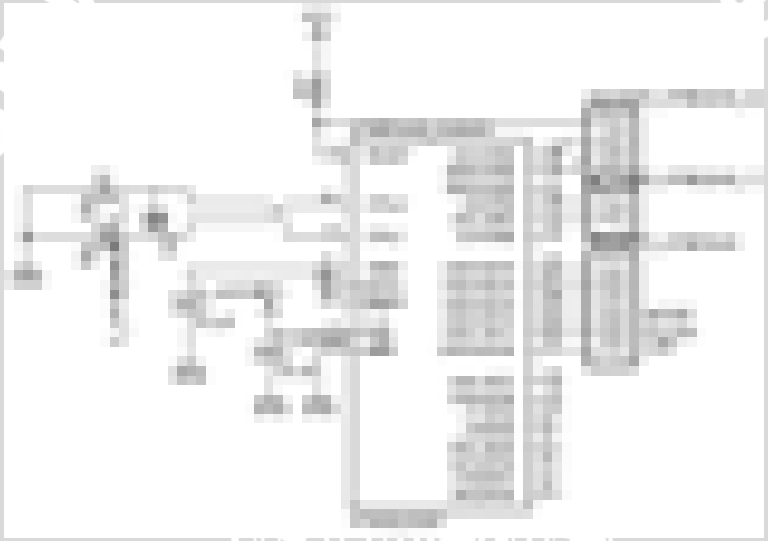
Gambar 4.9 Sistem Minimum Mikrokontroler ATMega16.

Mikrokontroler ATMega16 mempunyai 32 jalur I/O yang dapat diprogram menjadi masukan dan keluaran. Ke-32 jalur tersebut dikelompokkan ke dalam empat *port*, yaitu *port* A, B, C, dan D. Pada perancangan ini pin yang digunakan adalah:

PIN A.0 = digunakan sebagai antarmuka dengan LCD sebagai pin *register select*
PIN A.1 = digunakan sebagai antarmuka dengan LCD sebagai pin R/W
PIN A.2 = digunakan sebagai antarmuka dengan LCD sebagai pin *enable*
PIN A.4 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus.
PIN A.5 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus
PIN A.6 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus
PIN A.7 = digunakan sebagai antarmuka dengan LCD sebagai pin data bus
PIN B.1 = difungsikan sebagai SS (komunikasi SPI dengan *slave* 1)
PIN B.2 = difungsikan sebagai SS1 (untuk komunikasi SPI dengan *slave* 2)
PIN B.3 = difungsikan sebagai SS2 (untuk komunikasi SPI dengan *slave* 3)
PIN B.4 = pin SS difungsikan sebagai SS3 (untuk komunikasi SPI dengan *slave* 4)
PIN B.5 = pin MOSI (untuk komunikasi SPI dengan *slave*)

PIN B.6 = pin MISO (untuk komunikasi SPI dengan *slave*)
PIN B.7 = pin SCK (untuk komunikasi SPI dengan *slave*)
PIN D.0 = Rx untuk transmisi UART
PIN D.1 = Tx untuk transmisi UART
XTAL1 = digunakan sebagai masukan dari rangkaian osilator kristal
XTAL2 = digunakan sebagai masukan dari rangkaian osilator kristal

Mikrokontroler *slave* menggunakan mikrokontroler ATMega 8. Mikrokontroler ini dipilih karena hanya difungsikan sebagai menerima data kecepatan adan arah motor dari mikrokontroler master dan memberikan pulsa, logika arah, dan logika mode motor. Gambar 4.10 menunjukkan rangkaian minimum dari ATMega8.



Gambar 4.10 Sistem Minimum Mikrokontroler ATMega8.

Mikrokontroler ATMega 8 mempunyai 23 jalur I/O yang dapat diprogram menjadi masukan atau keluaran. Ke-23 jalur I/O ini dikelompokkan menjadi 3 kelompok, yaitu *port* B, C, dan D. Pada perancangan ini pin – pin yang digunakan adalah sebagai berikut:

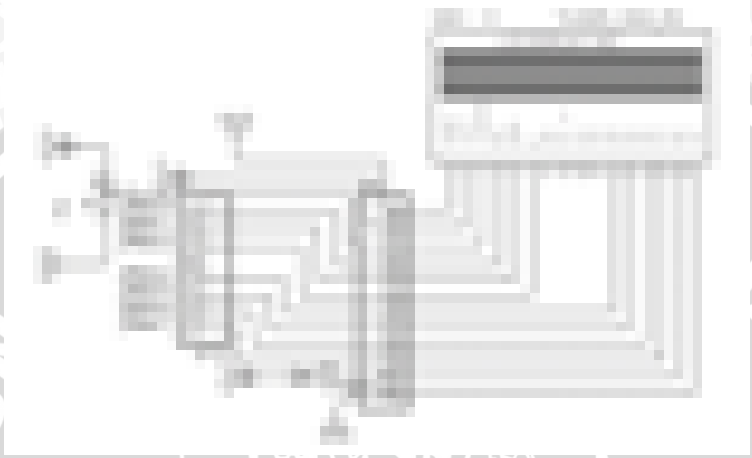
PIN PB.2 = pin SS (untuk komunikasi SPI dengan M*aster*)
PIN PB.3 = pin MOSI (untuk komunikasi SPI dengan M*aster*)
PIN PB.4 = pin MISO (untuk komunikasi SPI dengan M*aster*)
PIN PB.5 = pin SCK (untuk komunikasi SPI dengan M*aster*)
PIN PC.0 = arah putaran motor
PIN PC.1 = pulsa motor
PIN PC.2 = mode motor *HALF/FULL STEP*

PIN PD.0 = Rx untuk transmisi UART

PIN PD.1 = Tx untuk transmisi UART

4.3.1.3 Perancangan *Display* LCD

LCD digunakan untuk *display* data kinematik robot diantaranya input perhitungan kinematik *Vx, Vy, ω* dan *output* kecepatan sudut masing- masing roda. Skematik perancangan LCD 2x16 ditunjukkan dalam Gambar 4.11.



Gambar 4.11 Skematik Perancangan LCD.

Berdasarkan Gambar 4.11 data LCD terhubung dengan PINA.4 sampai PIN.A 7. Sedangkan untuk PINA.0 sampai PINA.2 digunakan untuk RS, R/W, dan Enable. Pin Vee pada LCD dihubungkan dengan *output* variabel resistor yang bernilai 10KΩ dengan dua kaki lainnya dihubungkan dengan Vcc dan GND. Tujuan pemasangan variabel resistor 10 KΩ dilakukan untuk mengubah kontras dengan tegangan yang masuk dalam pin Vee.

4.3.1.4 Perancangan Tombol Mode

Tombol mode digunakan untuk memberi *input* perintah pergerakan pada mikrokontroler dan *utilitas* untuk menu pada LCD. Pada perancangan tombol mode digunakan empat buah *task switch* yang dihubungkan dengan *resistor pull down*. Skematik perancangan rangkaian tombol mode ditunjukkan dalam Gambar 4.12.

Gambar 4.12 Skematik Perancangan Tombol Mode.

Berdasarkan Gambar4.12 Ketika *task switch* tidak ditekan, *output* berlogika *low*, sedangkan ketika *task switch* ditekan *output* berlogika *high.*

4.3.1.5 Perancangan Rangkaian Aktivasi

Rangkaian aktivasi digunakan untuk antar muka mikrokontroler dengan *downloader*. Pengguanaan rangkaian aktivasi ini sebagai antar muka untuk menulis program dari komputer ke *downloader* dan dihubungkan ke mikrokontroler. Skema perancangan rangkaian aktivasi mikrokontroler ditunjukkan dalam Gambar 4.13.

Gambar 4.13 Skema Perancangan Rangkaian Aktivasi Mikrokontroler.

Berdasarkan Gambar 4.13 pin konektor *amphenol* pin 2 terhubung dengan pin MISO mikrokontroler, pin 4 terhubung dengan pin SCK mikrokontroler, pin 6 terhubung dengan pin RESET mikrokontroler, pin 7 adalah ground, pin 9 adalah Vcc, pin 10 terhubung dengan pin MOSI mikrokontroler.

4.4 Perancangan Kinematika *Mobile* Robot Roda Mekanum

Pada *mobile* robot roda mekanum mengguanakan persamaan kinematik untuk kontrol pergerakannya. Persamaan kinematic ditujukan untuk mencari

kecepatan masing masing roda agar didapatkan pergerakan robot yang diinginkan dalam (Vx, Vy, ω). Persamaan keempat roda dapat dijabarkan dalam persamaan (4-3) sampai (4-6) :

$$v_1 = V_x - V_y - a\omega - b\omega \quad (4\text{-}3)$$

$$v_2 = V_x + V_y + a\omega + b\omega \quad (4\text{-}4)$$

$$v_3 = V_x + V_y - a\omega - b\omega \quad (4\text{-}5)$$

$$v_4 = V_x - V_y + a\omega + b\omega \quad (4\text{-}6)$$

Nilai V_1, V_2 ,V_3, V_4 didapatkan dari persamaan kinematik robot (4-3) sampai (4-6). Jika persamaan pergerakan robot yang diketahui $Vx = 0{,}7$, $Vy = 0$, $\omega = 0$ Maka perhitungan kinematik robot adalah sebagai berikut :

$$\omega_1 = \frac{1}{R}(V_x - V_y - a\omega - b\omega)$$
$$= \frac{1}{0{,}05}(0{,}7\text{-}0\text{-}0\text{-}0)$$
$$= 14$$

$$\omega_2 = \frac{1}{R}(V_x + V_y + a\omega + b\omega)$$
$$= \frac{1}{0{,}05}(0{,}7+0+0+0)$$
$$= 14$$

$$\omega_3 = \frac{1}{R}(V_x + V_y - a\omega - b\omega)$$
$$= \frac{1}{0{,}05}(0{,}7+0\text{-}0\text{-}0)$$
$$= 14$$

$$\omega_4 = \frac{1}{R}(V_x - V_y + a\omega + b\omega)$$
$$= \frac{1}{0{,}05}(0{,}7\text{-}0+0+0)$$
$$= 14$$

Pengolahan rumus kinematik mekanum dilakukan dalam mikrokontroler master. *Input* mode terdiri dari pergerkan lurus, pergerakan samping, dan pergerakan diagonal.

Tabel 4.1 Kecepatan Tiap Roda

θ	Kecepatan Robot			ω1	ω2	ω3	ω4
	Vx	Vy	ω				
0°	0,7	0	0	14	14	14	14
45°	0,4	0,4	0	0	16	16	0
90°	0	0,6	0	-12	12	12	-12
135°	-0,4	0,4	0	16	0	0	16
180°	-0,7	0	0	-14	-14	-14	-14
225°	-0,4	-0,4	0	0	-16	-16	0
270°	0	-0,6	0	12	-12	-12	12
315°	0,4	-0,4	0	-16	0	0	-16

4.5 Perancangan Perangkat Lunak

Dalam skripsi ini perancangan perangkat lunak terdiri atas dua bagian, yaitu perancangan perangkat lunak untuk mikrokontroler *master* dan mikrokontroler *slave*. Perancangan perangkat lunak *master* untuk mengolah masukan dari tombol mode dan untuk proses pengolahan rumus kinematika *mobile robot* roda mekanum. Sedangkan perancangan perangkat lunak *slave* untuk mengolah data kecepatan dan arah menjadi pulsa dan logika arah putaran motor.

4.5.1 Perancangan Susunan Perangkat Lunak

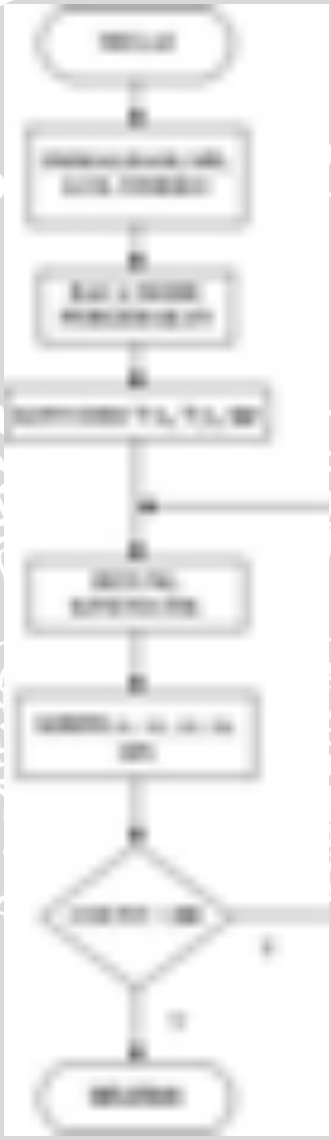
Tahapan proses yang terdapat pada system ini meliputi, *input* mode pergerakan robot melalui PIN I/O. kemudian mode pergerakan diterjemahkan oleh mikrokontroler master menjadi (V_x, V_y, ω). Kemudian mikrokontroler *master* memasukkan (V_x, V_y, ω) kedalam perhitungan kinematik sehingga didapatkan nilai ω_1, ω_2 ,ω_3, ω_4 . konstanta yang tetap pada rumus kinematik dan dideklarasikan dalam program adalah a dan b yang bernilai 170 mm. setelah didapatkan ω_1, ω_2 ,ω_3, ω_4 mikrokontroler *master* mengirim data tersebut ke mikroontroler *slave* dengan komunikasi SPI. Data kecepatan dan arah yang diterima mikrokontroler *slave* akan diterjemahkan oleh mikrokontroler *slave* sehingga didapatkan bentuk data pulsa dan arah putaran motor.

Semua intruksi program disusun secara terstruktur dalam beberapa *subrutin* yang secara khusus menangani fungsi tertentu. *Software* mikrokontroler dibuat

menggunakan *compiler Code Vision AVR* buatan HP infotech. Menggunakan bahasa pemrogaman yaitu bahasa C dan *downloader* menggunakan AVR USB.

4.5.2 Perancangan Program Mikrokontroler *Master*

Program utama mikrokontroler *master* dirancang untuk melakukan perhitungan kinematika robot dan komuikasi SPI dengan tiga mikrokontroler *slave*. Diagram alir program utama *master* ditunjukkan dalam Gambar 4.14.



Gambar 4.14 Flowchart Mikrokontroler *Master*.

Berdasarkan diagram alir yang ditunjukkan dalam Gambar 4.14, pertama mikrokontroler *master* menerima *input* mode pergerakan dalam delapan mode sudut diantarayanya 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0. Mode tersebut dikonversi menjadi V_x, V_y, ω. Masing-masing mode pergerakan sudah di definisikan V_x, V_y, ω

dalam *header* program. Mikrokontroler *master* melakukan perhitungan kinematik sesuai dengan *input* V_x, V_y, ω. Hasil perhitungan kinematik yang berupa kecepatan sudut kemudian dikirimkan ke masing masing *slave* berupa nilai kecepatan sudut dan arah. Dalam program utama terdapat *timer* untuk menghitung kapan waktu robot untuk berhenti. Ketika COUNT>200 maka akan selesai, apabila COUNT<200 program kembali *looping* dalam perhitungan kinematik.

4.5.2.1 Perancangan Program Kinematika Robot

Pada *mobile* robot roda mekanum mengguanakn persamaan kinematik untuk kontrol pergerakannya. Persamaan kinematik ditujukan untuk mencari kecepatan masing masing roda agar didapatkan pergerakan robot yang diinginkan dalam (Vx, Vy, ω). Persamaan keempat roda dapat dijabarkan dalam persamaan (4-3) sampai (4-6) :

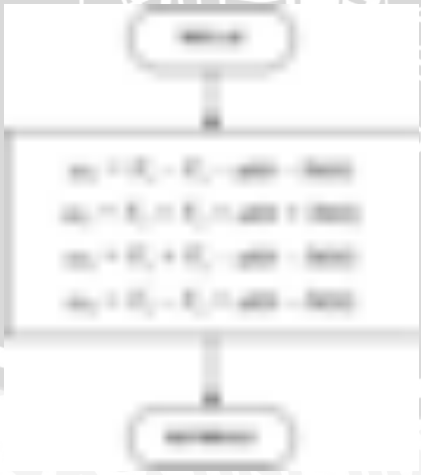
$$\omega_1 = \frac{1}{R}\,(V_x - V_y - a\omega - b\omega) \qquad (4\text{-}3)$$

$$\omega_2 = \frac{1}{R}\,(V_x + V_y + a\omega + b\omega) \qquad (4\text{-}4)$$

$$\omega_3 = \frac{1}{R}\,(V_x + V_y - a\omega - b\omega) \qquad (4\text{-}5)$$

$$\omega_4 = \frac{1}{R}\,(V_x - V_y + a\omega + b\omega) \qquad (4\text{-}6)$$

Persamaan (4-3) sampai (4-6) sebagai perhitungan kinematik robot. Dragram alir program kinematik robot ditunjukkan dalam Gambar 4.15.



Gambar 4.15 Diagram Alir Program *Invers* Kinematik Robot.

Berdasarkan Gambar 4.15 program melakukan perhitungan kinematik robot sehingga didapatkan kecepatan sudut masing-masing roda. ω_1 adalah kecepatan sudut roda 1, ω_2 adalah kecepatan sudut roda 2, ω_3 adalah kecepatan sudut roda 3, ω_4 adalah kecepatan sudut roda 4. Hasil perhitungan kinematik dalam satuan rad/s.

4.5.2.2 Perancangan Program Komunikasi SPI

Diagram alir pengiriman data berfungsi untuk mengatur arah gerak putaran motor dan mengirim data kecepatan tiap roda ke masing–masing *slave* dengan komunikasi SPI. Untuk mengatur arah gerak motor diatur melalui pin pada mikrokontroler *master*. Saat kecepatan bernilai positif maka motor maju, sedangkan saat kecepatan bernilai negatif maka motor berputar sebaliknya. Tiap SS *slave* dihubungkan dengan PINB.1-4 pada mikrokontroler *master* untuk menentukan *slave* mana yang menerima data. Gambar 4.16 menunjukkan diagram alir pengiriman data.

Gambar 4.16 Diagram Alir Program Komunikasi SPI.

Berdasarkan Gambar 4.16 pengiriman data SPI dibagi menjadi dua untuk masing-masing *slave*, yaitu data kecepatan sudut dan arah putaran motor. Terdapat empat mikrokontroler *slave* yang diwakili oleh pin SS, yaitu SL1, SL2, SL3, SL4.

4.5.3 Perancangan Program Mikrokontroler Slave

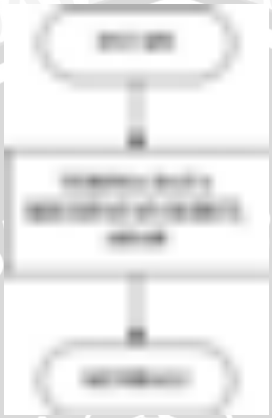
Program utama mikrokontroler *slave* dirancang untuk melakukan proses komunikasi dengan mikrokontroler *master* dengan SPI, diantaranya menerima data kecepatan sudut dan arah dan mengkonversi menjasdi frekuensi pulsa. Diagram Alir program mikrokontroler slave ditunjukkan dalam gambar 4.17.

Gambar 4.17 Diagram Alir Program Utama *Slave*

Berdasarkan Gambar 4.17 mikrokontroler *slave* menerima data kecepatan sudut dari *master* melalui komunikasi SPI. Data kecepatan sudut diubah menjadi bentuk kecepatan (rpm) dan dikirim melalui komunikasi UART. Data kecepatan (rpm) diubah menjadi frekuensi pulsa motor. Data frekuensi pulsa motor diubah menjadi nilai OCR dan dibandingkan dengan nilai TCNT. Apabila data sesuai maka nilai tersebut adalah nilai frekuensi pulsa yang dikeluarkan mikrokontroler.

4.5.3.1 Perancangan Program SPI *Slave*

Mikrokontroler *Slave* menerima data dari mikrokontroler *master* dengan fungsi *interrupt* SPI. Kemdudian data dikonversi dan diolah pada program utama *slave*. Diagram alir SPI ditunjukkan dalam Gambar 4.18.

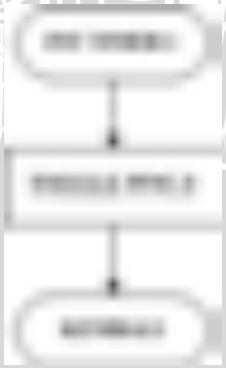


Gambar 4.18 Diagram Alir Komunikasi SPI *Slave*

Mikrokontroler *slave* menerima data SPI kecepatan sudut dan arah, ketika nilainya 0 berarti nilai kecepatan sudut positif dan ketika nilainya 1 berarti nilai kecepatan sudut negatif.

4.5.3.2 Perancangan Program Konversi Frekuensi ke *Timer*

Mikrokontroler *slave* juga bertugas untuk mengubah frekuensi dalam bentuk timer yang menentukan kondisi *toggle* PINC.0. Diagram alir program konversi frekuensi ke *timer* ditunjukkan dalam Gambar 4.19.



Gambar 4.19 Diagram Alir Konversi Frekuensi ke *Timer*

Konversi frekuensi ke *timer* pada intinya adalah membandingkan nilai TCNT dengan nilai OCR. Nilai dari *timer* 1 adalah 16 bit atau 65536, sedangkan nilai OCR masuk disesuaikan dengan nilai frekuensi yang ditentukan. Persamaan (4-7) menunjukkan fungsi frekuensi terhadap nilai OCR.

$$OCR = \frac{1}{f.2.T} \quad (4\text{-}7)$$

Nilai *T* adalah 0,000016 karena frekuensi *timer* yang ditetapkan adalah 625000. *f* adalah nilai frekuensi yang akan diubah menjadi nilai OCR. Saat nilai OCR sama dengan nilai TCNT maka nilai tersebut adalah frekuensi yang digunakan untuk menentukan kondisi *toggle* PINC.0

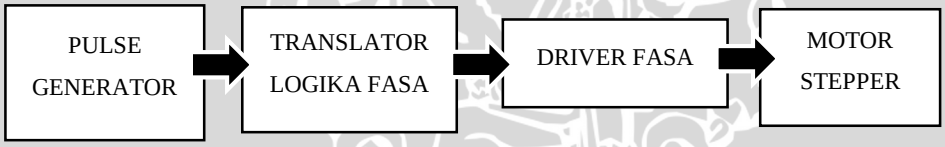
BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui apakah sistem telah bekerja sesuai perancangan yang telah dilakukan. Pengujian dilakukan per blok sistem kemudian secara keseluruhan. Adapun pengujian yang dilakukan sebagai berikut:

1) Pengujian Motor *Stepper*
2) Pengujiaan Komunikasi SPI antara Mikrokontroler *Master* dan *Slave*
3) Pengujian Kinematika Robot
4) Pengujian Keseluruhan Sistem

5.1 Pengujian Motor *Stepper*

Pada pengujian motor *stepper* dibagi menjadi pengujian kecepatan motor *stepper* dan pengujian sudut putaran motor *stepper*. Diagram blok pengujian ditunjukkan dalam Gambar 5.1.



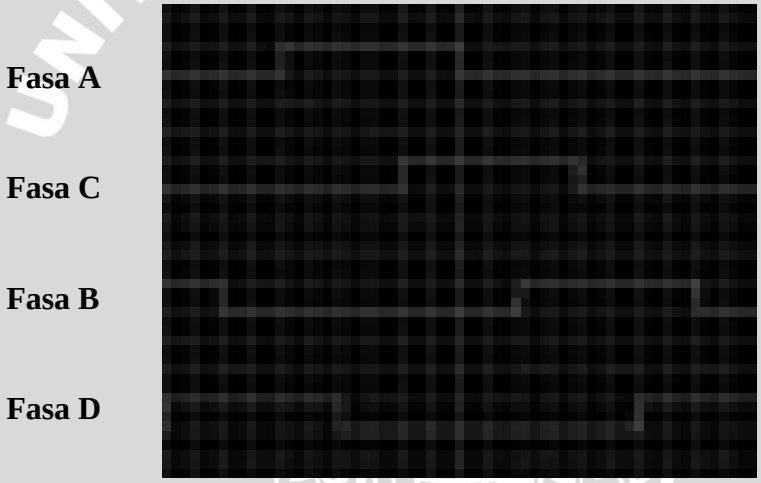
Gambar 5.1 Diagram Blok Pengujian Kecepatan Motor *Stepper.*

Berdasarkan Gambar 5.1 pengujian sudut putaran motor *stepper* dan pengujian kecepatan motor *stepper* dilakukan dengan memberikan pulsa dari generator pulsa. Frekuensi pulsa diproses pada translator logika untuk kontrol logika fasa motor *stepper*, translator logika fasa pada tugas akhir ini menggunakan IC L297. Logika fasa motor masuk pada rangkaian *driver* fasa untuk mendapatkan catu tegangan motor *stepper*, *driver* fasa motor *stepper* menggunakan IC STK6713BMK4. Output *driver* masing-masing fasa dihubungkan dengan masing-masing fasa motor *stepper*.

Sinyal *output generator* pulsa ditunjukkan dalam Gambar 5.2

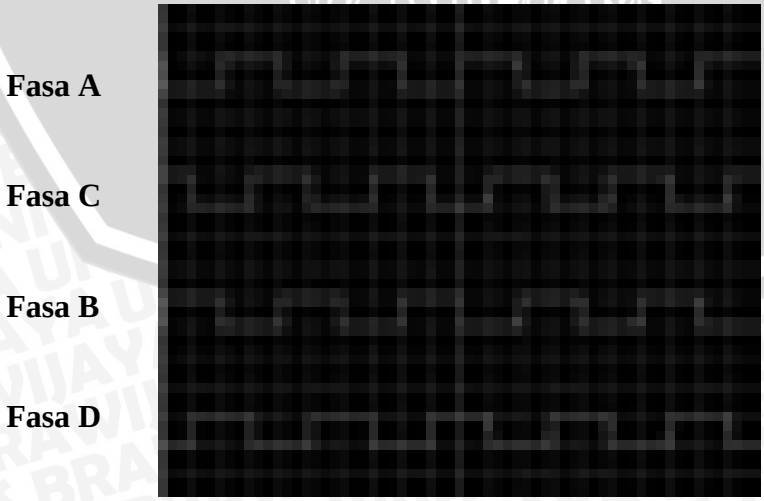
Gambar 5.2 Sinyal *Output Generator* Pulsa dengan Frekuensi 1282 KHz.

IC L297 berfungsi sebagai kontrol logika motor *stepper*. IC L297 bekerja pada dua mode, yaitu mode *full step* dan *half step*. *Output* IC L297 adalah fasa A, B, C, D. Sinyal *output* L297 yang bekerja pada mode *half step* ditunjukkan dalam Gambar 5.3



Gambar 5.3 Sinyal *Output* L297 Mode *Half Step*.

Sinyal *output* L297 pada mode *full step* ditunjukkan dalam Gambar 5.4



Gambar 5.4 Sinyal *Output* L297 Mode *Full step*.

Berdasarkan Gambar 5.2 dan 5.3 dapat dianalisa bahwa pada mode *full step* mencapai siklus penuh dan kembali ke fasa awal lebih cepat dari pada mode *half step*. Sesuai dengan perhitungan teori untuk mode *half step* yang membutuhkan jumlah pemicuan lebih banyak daripada *full step* (dua kali *full step*). Tabel logika fasa motor *stepper* pada mode *half step* ditunjukkan dalam Tabel 5.1

Tabel 5.1 Logika Fasa Mode *Half Step*

Clock	A	C	B	D
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	0	0
6	1	1	0	0
7	1	0	0	0
8	1	0	0	1

Tabel logika fasa motor *stepper* pada mode *full step* ditunjukkan dalam Tabel 5.1

Tabel 5.2 Logika Fasa Mode *Full Step*

Clock	A	C	B	D
1	0	0	1	1
2	0	1	1	0
3	1	1	0	0
4	1	0	0	1

IC 74LS04 sebagai pembalik logika karena IC STK6713BMK4 aktif logika *low*. IC STK6713BMK4 sebagai *driver* fasa, sinyal diberikan untuk mengaktifkan motor *stepper*.

5.1.1 Pengujian Sudut Putaran Motor *Stepper*

. Pengujian sudut putaran motor *stepper* dilakukan dengan memberikan jumlah pulsa yang dibandingkan dengan sudut putaran motor *stepper*. Pengujian ini menggunakan mode *full step*. Hasil pengujian ditunjukkan dalam Tabel 5.3.

Tabel 5.3 Hasil Pengujian Sudut Putaran Motor *Stepper*

Jumlah Pulsa	Perhitungan Sudut Putaran	Pengukuran Sudut Putaran	Error
50	90°	88°	2,2%
		90°	0%
100	180°	177°	1,67%
		178°	1,1%
150	270°	268°	0,74%
		268°	0,74%
200	360°	358°	0,55%
		360°	0%
Rata-rata			0,875%

Berdasarkan Tabel 5.3 Sudut putaran motor *stepper* memiliki kesalahan terbesar 3°. Kesalahan tersebut dikarenakan hasil pengukuran yang menggunakan metode manual dengan melihat langsung sudut pada penggaris busur dipengaruhi oleh sudut pengamatan. Pengujian sudut putaran dilakukan dengan menggunakan mode *full step* pada *driver* L297. Pengujian sudut putaran motor *stepper* dilakukan dilakukan menggunakan penggaris busur dengan ketelitian 1°. Pengujian sudut putaran motor *stepper* ditunjukkan dalam Gambar 5.5

Gambar 5.5 Pengujian Sudut Putaran *Motor Stepper*.

Hasil pengujian putaran *motor stepper* masih dalam batas toleransi dengan kesalahan terbesar sebesar 3 rpm. Dalam perhitungan teori perhitungan putaran sudut motor stepper dapat dijelaskan dalam persamaan (5-1).

Jumlah pulsa = $\frac{Sudut\ putaran}{1{,}8°}$ (5-1)

Dimana :

1,8 °= sudut per langkah motor *stepper*

5.1.2 Pengujian Kecepatan Motor *Stepper*

Pengujian ini dilakukan dengan tujuan untuk mengetahui perbandingan frekuensi pulsa dan mode perputaran motor *stepper* yang berpengaruh terhadap kecepatan motor *stepper*. Dalam pengujian kecepatan motor dilakukan dalam dua mode, yaitu mode *full step* dan *half step*. Dalam pengujian kecepatan motor *stepper* dilakukan dengan membandingkan frekuesi pulsa dan penggunaan mode perputaran motor *stepper*. Frekuensi dihasilkan oleh generator pulsa dengan *dutty cycle* 50% Hasil pengujian kecepatan motor *stepper* ditunjukkan dalam Tabel 5.4.

Tabel 5.4 Kecepatan Motor *Stepper*

Frekuensi (Hz)	**Teori Half Step (rpm)**	**Teori Full Step (rpm)**	**Pengujian**		**Error**	
			***Half Step* (rpm)**	***Full Step* (rpm)**	***Half Step***	***Full Step***
50	7,5	15	8	15	6,67 %	0 %
100	15	30	18	28	20 %	6,67 %
200	30	60	33	60	10 %	0%
300	45	90	45	87	0 %	3,33 %
400	60	120	60	118	0 %	1,67 %
500	75	150	75	150	0 %	0 %
600	90	180	90	178	0 %	1,11 %
700	105	210	105	210	0 %	0 %
800	120	240	120	240	0 %	0 %
900	135	270	135	270	0 %	0 %
1000	150	300	150	295	0 %	1,67 %
1100	165	330	165	330	0 %	0 %
1200	180	360	185	360	2,7 %	0 %
1300	195	390	198	393	1,5 %	0,76 %
1400	210	420	205	418	2,38 %	0,47 %
1500	225	450	223	450	0,8 %	0 %
1600	240	480	240	483	0 %	0,62 %
1700	255	510	255	515	0 %	0,9 %
1800	270	540	270	535	0 %	0,9 %
1900	285	570	285	570	0 %	0 %
2000	300	600	300	600	0 %	0 %
Rata-rata					2,09%	0,86%

Pengujian dilakukan dengan mengukur kecepatan motor (rpm) dengan menggunakan *tachometer*. Berdasarkan Tabel 5.4 Diketahui bahwan kecepatan

motor *stepper* pada mode *full step* dua kali kecepatan motor *stepper* pada mode *half step*. Kesalahan terbesar adalah 5 rpm, hal ini dikarenakan ketelitian dari *tachometer* adalah 5 rpm. Dalam perhitungan teori diketahui kecepatan untuk mode *full step* dan *half step* ditunjukkan dalam persamaan (5-2) dan (5-3)

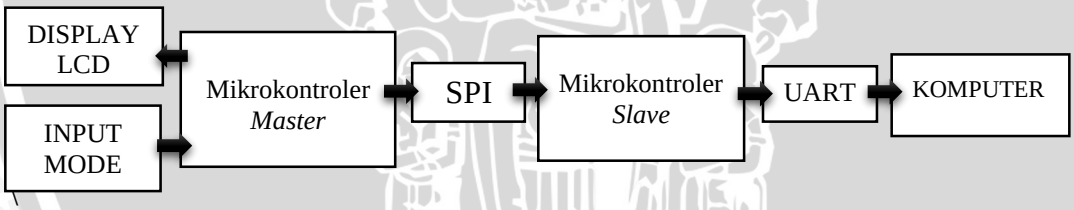
$$\text{Kecepatan } \textit{full step} \text{ (rpm)} = \frac{1{,}8}{360}\ x\ 60\ x\ f \qquad (5\text{-}2)$$

$$\text{Kecepatan } \textit{half step} \text{ (rpm)} = \left(\frac{1{,}8}{360}\ x\ 60\ x\ f\right) : 2 \qquad (5\text{-}3)$$

Kesalahan kecepatan motor *stepper* dilakukan dengan membandingkan kecepatan hasil perhitungan teori dengan kecepatan terukur. Kesalahan kecepatan motor *stepper* terbesar adalah 5 rpm.

5.2 Pengujian Komunikasi SPI Antara Mikrokontroler *Master* dan *slave*

Pengujian ini dilakukan untuk mengetahui dan memeriksa perangkat lunak yang disusun dalam perangkat mikrokontroler apakah sudah dapat menangani komunikasi SPI antara mikrokontroler *master* (ATmega16) dengan mikrokontroler *slave* (ATmega8). Prosedur Pengujian dilakukan dengan menghubungkan LCD, mikrokontroler ATMega16, mikrokontroler ATMega8, dan komputer sesuai dengan Gambar 5.6.



Gambar 5.6 Diagram Alir Pengujian SPI MK *Master-Slave.*

Pada pengujian komunikasi SPI ini mikrokontroler *master* akan mengirimkan data sesuai dengan arah tombol mode ditekan. Sudut ini menjadi masukan mikrokontroler *master* untuk menentukan kecepatan sudut tiap roda. Data kecepatan ini kemudian dikirim ke masing-masing *slave* melalui komunikasi SPI secara bersamaan pada masing-masing *slave*. Mikrokontroler s*lave* mengirim data ke komputer melalui komunikasi UART. Pengiriman data pada komunikasi UART pada setiap *sleve* tidak dapat dilakukan bersama sehingga pengambilan data dilakukan bergantian. Hasil pengujian komunikasi SPI dari tiga kali percobaan ditunjukkan dalam Tabel 5.5.

Tabel 5.5 Hasil Pengujian komunikasi SPI *Master – Slave*

Nilai pada LCD				Nilai pada komputer			
ω_1	**ω_2**	**ω_3**	**ω_4**	***Slave* 1**	***Slave* 2**	***Slave* 3**	***Slave* 4**
14	14	14	14	14	14	14	14
-12	12	12	-12	-12	12	12	-12
0	16	16	0	0	16	16	0
16	0	0	16	16	0	0	16
-14	-14	-14	-14	-14	-14	-14	-14
0	-16	-16	0	0	-16	-16	0
12	-12	-12	12	12	-12	-12	12
-16	0	0	-16	-16	0	0	-16

Dari hasil pengujian komunikasi SPI (*Serial Peripheral Interface*) terlihat bahwa mikrokontroler *master* dapat mengirimkan data konstanta ke mikrokontroler *slave*. Data yang ditampilkan pada mikrokontroler *slave* di komputer sama dengan data yang ditampilkan oleh mikrokontroler *master* pada LCD. Data kecepatan sudut yang akan dikirim dalam komunikasi SPI ditunjukkan dalam Tabel 5.6.

Tabel 5.6 Data Kinematik pada LCD

θ	**Data Kecepatan Sudut LCD**	**Data Komputer**
0°		14 14 14 14
45°		0 16 16 0
90°		-12 12 12 -12
135°		16 0 0 16
180°		-14 -14 -14 -14

225°  0 -16 -16 0

270°  12 -12 -12 12

315°  -16 0 0 -16

Hasil pengujian komunikasi SPI ditampilkan ke komputer melalui komunikasi UART. Data yang telah dikirim dan ditampilkan dalam komputer ditunjukkan dalam Gambar 5.7

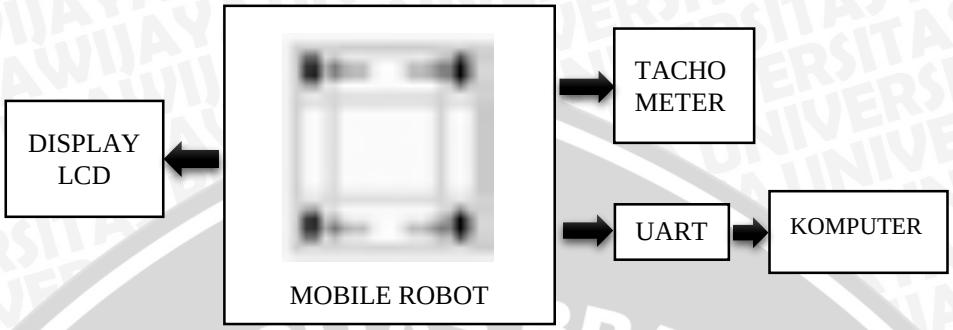
Gambar 5.7 Data Komunikasi SPI dalam Komputer.

Data komunikasi SPI antara data sebelum dan sesudah dikirim sesuai dan memiliki *error* 0%. Frekuensi yang digunakan dalam komunikasi SPI antara mikrokontroler *master* dan *slave* adalah 4000.000 kHz. Pengiriman data dilakukan dua kali, diantaranya data kecepatan sudut dan data arah kecepatan sudut. *Baudrate* yang digunakan dalam komunikasi UART antara mikrokontroler dan komputer menggenakan nilai *baudrate* 9600.

5.3 Pengujian Kinematika Mekanum

Pengujian ini bertujuan untuk menganalisa hasil perhitungan kinematik robot dan mengetahui simpangan kecepatan pada masing masing roda. Prosedur pengujian dilakukan dengan memberi perintah pada robot untuk bergerak ke sudut tertentu. Kecepatan tiap roda hasil perhitungan kinematika robot ditampilkan pada LCD. Kecepatan sudut tiap roda kemudian dikirim ke mikrokontroler slave untuk

diolah menjadi pulsa dan arah. Diagram blok pengujian kecepatan putaran roda ditunjukkan dalam Gambar 5.8.



Gambar 5.8 Diagram Blok Pengujian Kecepatan Roda.

Hasil pengujian kinematika mekanum pada sudut – sudut tertentu yang diperoleh melalui beberapa kali percobaan ditunjukkan dalam tabel 5.7.

Tabel 5.7 Hasil Pengujian Kecepatan Roda

θ	Mode Pergerakan			Tampilan LCD			
	Vx	Vy	ω	ω_1	ω_2	ω_3	ω_4
0°	0,7	0	0	14	14	14	14
	0,7	0	0	14	14	14	14
45°	0,4	0,4	0	0	16	16	0
	0,4	0,4	0	0	16	16	0
90°	0	0,6	0	-12	12	12	-12
	0	0,6	0	-12	12	12	-12
135°	0,4	-0,4	0	16	0	0	16
	0,4	-0,4	0	16	0	0	16
180°	-0,7	0	0	-14	-14	-14	-14
	-0,7	0	0	-14	-14	-14	-14
225°	-0,4	-0,4	0	0	-16	-16	0
	-0,4	-0,4	0	0	-16	-16	0
270°	0	-0,6	0	12	-12	-12	12
	0	-0,6	0	12	-12	-12	12
315°	-0,4	0,4	0	-16	0	0	-16
	-0,4	0,4	0	-16	0	0	-16

Berdasarkan Tabel 5.47 pengujian dilakukan dengan memberikan nilai *Vx*, *Vy*, ω dalam sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0 pada persamaan kinematik dalam mikrokontroler master yang menghasilkan $\omega_1, \omega_2, \omega_3, \omega_4$ yang ditampilkan pada komputer dengan komunikasi UART. Berdasarkan Tabel 5.7 dapat diperoleh hasil bahwa perhitungan kecepatan sudut masing-masing roda sesuai dengan perhitungan kinematik robot. Nilai kecepatan sudut dari persamaan kinematik pada mikrokontroler sesuai dengan perhitungan teorinya.

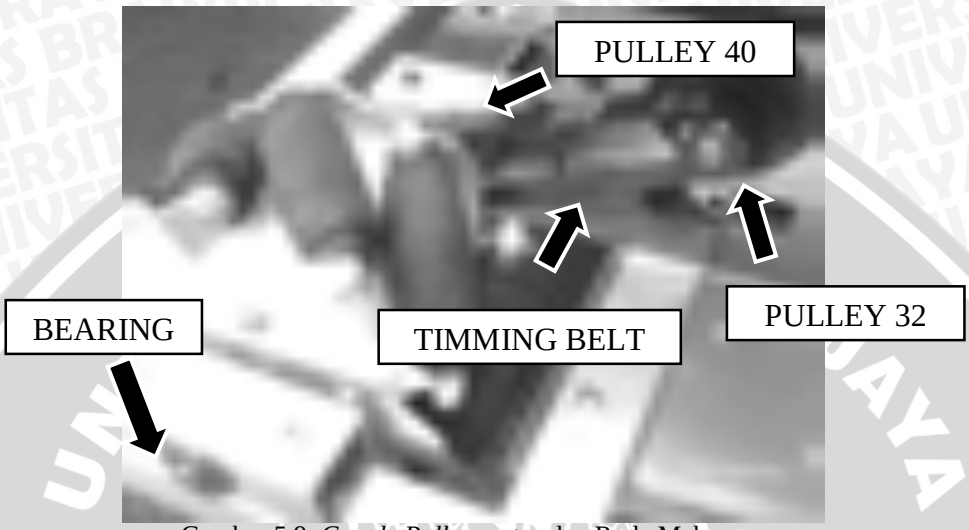
Perbandingan kecepatan roda robot dengan perhitungan kinematik dapat dijelaskan dalam Tabel 5.8.

Tabel 5.8 Pengujian Kecepatan Roda Mekanum

θ	**Teori/ Uji**	**Kecepatan Roda (rpm)**			
		R_1	**R_2**	**R_3**	**R_4**
0°	T	134,6	134,6	134,6	134,6
	U	139	132	140	137
45°	T	0	153,8	153,8	0
	U	0	150	160	0
90°	T	-115,3	115,3	115,3	-115,3
	U	-118	118	118	-120
135°	T	153,8	0	0	153,8
	U	158	0	0	155
180°	T	-134,6	-134,6	-134,6	-134,6
	U	-136	-133	-138	-138
225°	T	0	-153,8	-153,8	0
	U	0	-158	-160	0
270°	T	115,3	-115,3	-115,3	115,3
	U	120	-123	-120	118
315°	T	-115,3	0	0	-115,3
	U	-118	0	0	-120
Error (rpm)		**3,38**	**3,6**	**4,25**	**3,18**

Berdasarkan Tabel 5.8 dapat dijelaskan bahwa perbandingan antara kecepatan motor dalam perhitungan dengan yang sebenarnya adalah kesalahan motor 1 sebesar 3,38 rpm, motor 2 sebesar 3,6 rpm, motor 3 sebesar 4,25 rpm, motor 4 sebesar 3,18 rpm. Kesalahan kecepatan motor terbesar adalah motor 3 dengan kesalahan 4,25 rpm. Rata – rata kesalahan kecepatan motor adalah 3,6 rpm.

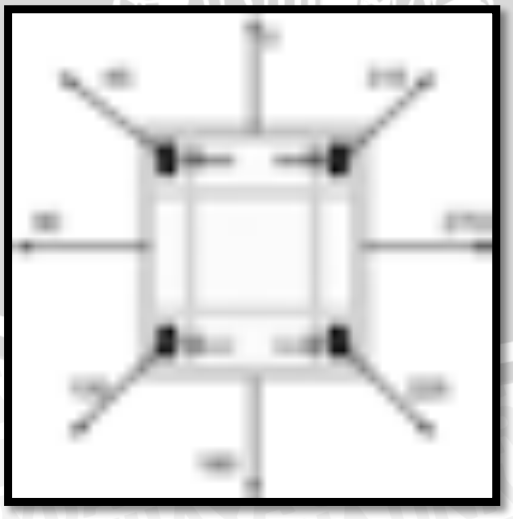
Kecepatan masing-masing roda memiliki perbedaan karena faktor kepresisian mekanik saat *couple pulley* antara motor dengan roda. Rapat dan renggang *timming belt* dan gaya gesek *bearing* berpengaruh juga terhadap kecepatan final dari masing-masing roda. *Couple pulley* motor dan roda mekanum ditunjukkan dalam Gambar 5.9.



Gambar 5.9 *Couple Pulley* motor dan Roda Mekanum.

5.4 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem ini bertujuan untuk mengetahui performansi *mobile* robot roda mekanum. Prosedur pengujian dilakukan memberikan mode pergerakan dengan variasi pergerakan robot, diantaranya mode maju, samping, dan pergerakan diagonal robot. Sudut pergerakan *mobile* robot ditunjukkan dalam Gambar 5.5



Gambar 5.5 Arah Gerak Robot

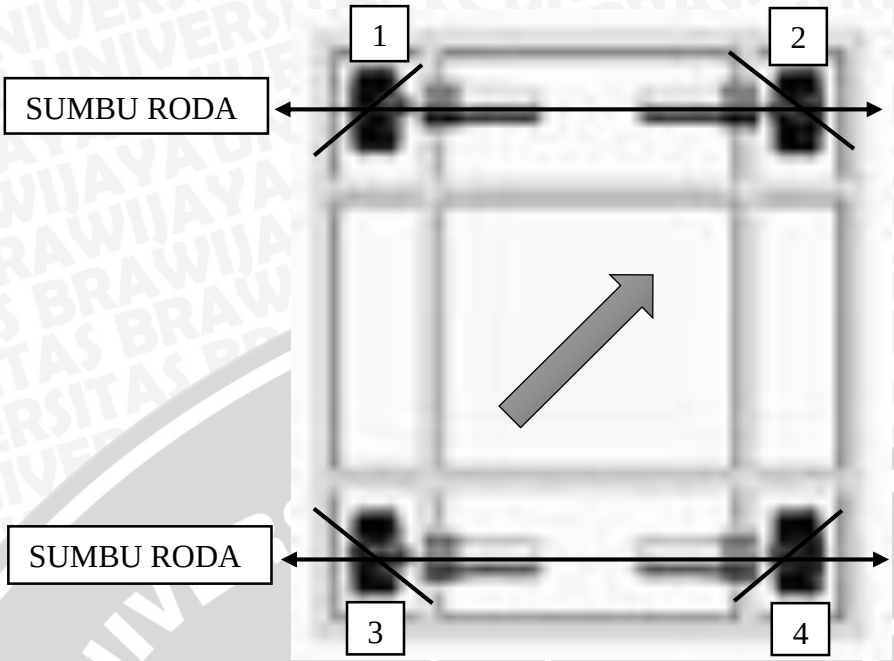
Hasil pengujian untuk respon robot saat dikendalikan untuk bergerak ke arah tertentu setelah beberapa kali percobaan dapat dilihat dalam tabel 5.9.

Tabel 5.9 Hasil Pengujian Arah Gerak Robot

Sudut (θ)	Sudut Kesalahan
0°	2°
	1°
45°	4°
	3,5°
90°	3°
	2°
135°	4°
	3,5°
180°	2°
	3°
225°	2°
	4°
270°	2,5°
	3,5°
315°	3°
	4°
Rata-Rata	**2,93°**

Berdasarkan tabel diatas dapat diperoleh hasil rata – rata kesalahan sudut gerak robot sebesar 2,93°. Pengujian keseluruhan sistem dilakukan pada lapangan berbahan *vinil* dengan lingkaran yang berdiameter 2 m. kesalahan terbesar terjadi pada pergerakan diagonal dengan kesalahan terbesar 4°.

Kesalahan sudut gerak robot dikarenakan keempat roda tidak memiliki kecepatan yang sama persis. Kecepatan robot terbesar pada sudut 45° sebesar 4° dikarenakan pada pergerakan diagonal juga terdapat lembam pada *roller* roda. *Roller* roda mekanum membentuk sudut 45° terhadap sumbu rodanya, sehingga ketika robot bergerak diagonal kelembaman *roller* mempengaruhi posisi gerak robot. Posisi *roller* pada masing masing roda ditunjukkan dalam gambar 5.10



Gambar 5.10 Posisi *Roller* Roda Mekanum.

Ketika robot bergerak dengan sumbu diagonal, roda yang bergerak adalah roda 2 dan 3. Sedangkan roda 1 dan 4 keadaan diam. *Roller* roda 1 dan 4 bisa bergerak dalam sudut 45° yang tegak lurus dengan sumbu roda. Kelembaman ini yang mengakibatkan kesalahan sudut gerak paling besar pada mode pergerakan diagonal.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil perancangan dan pengujian yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut :

1. Pergerakan robot dalam sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0 menggunakan persamaan kinematika mekanum dengan kesalahan roda 1 sebesar 3,38 rpm, roda 2 sebesar 3,6 rpm, roda 3 sebesar 4,25 rpm, dan roda 4 sebesar 3,18 rpm, sehingga diperoleh kesalahan rata-rata roda sebesar 3,6 rpm.
2. Kecepatan motor *stepper* dalam mode *full* step dua kali lebih besar dibandingkan mode *half* step dan kesalahan rata-rata kecepatan motor *stepper* terbesar adalah 2,09%.
3. Komunikasi data antara mikrokontroler *master* ATMega 16 dan mikrokontroler *slave* ATMega 8 menggunakan komunikasi SPI dengan tingkat keberhasilan 100 % pada frekuensi 4000.000 KHz.
4. *Mobile robot* roda mekanum ukuran 100 mm dengan jarak antar sumbu roda 340 mm dapat bergerak ke sudut 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, dan 315^0 dengan kesalahan rata – rata sudut gerak robot sebesar 2,93°.

6.2 Saran

Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah.

1. Perlu dicari alat pengendali robot yang dapat menunjukkan sudut gerak robot yang lebih presisi.
2. Diperlukan studi lebih lanjut agar robot dapat melakukan metode pemetaan posisi.

DAFTAR PUSTAKA

Artha, Raditya. 2013. *Implementasi Robot The Three Omni Directional Menggunakan Kontroler Proportional Integral Derivative (PID) pada Robot Kontes Robot Abu Indonesia (KRAI).* Skripsi tidak dipublikasikan. Malang: Universitas Brawijaya.

Atmel. 2006. *ATMEGA8/ATMEGA8L, 8-bit AVR with 8 kbytes in System Programable Flash.* www.atmel.com/literatur. Diakses tanggal: 8 Mei 2014.

Atmel. 2007. *ATMEGA16/ATMEGA16L, 8-bit AVR with 8 kbytes in System Programable Flash.* www.atmel.com/literatur. Diakses tanggal: 8 Mei 2014.

Berns, Karsten. 2009. *Autonomus Land Vehicle*. Netherland:Krips.

Boreinstein, Everett. 1996. *Navigation Mobile Robot*. Wellesley:Masschuttes.

Braunl, Thomas. 2006. *Embedded Robotics Mobile Robot Design and Applications with Embedded Systems*. Germany:Springer.

Dikti 2012. *Panduan Kontes Robot Indonesia 2012*. Jakarta: DIKTI.

Dikti 2013. *Panduan Kontes Robot Indonesia 2013*. Jakarta: DIKTI.

Dikti 2014. *Panduan Kontes Robot Indonesia 2014*. Jakarta: DIKTI.

Doroftei, I., Stirbu, B.Design. 2010. *Modeling and Control of an Omni-directional Mobile Robot*. Solid StatePhenomena. Vol. 166-167, pp 173-178.

Efendi, Jefri. 2006. *Designing Omni Directional Mobile Robot with Mecanum Wheel*. American Journal of Apllied Science. Vol.3, pp.1831-1835.

Habib, K.. 2007. *Bioinspitarion and Robotics: Walking and Climbing Robot*. Vienna:I-Tech.

Mackenzie, Ian. 2008. *Omnidirectional Drive System Kinematics and Control.* 2008 First Robotics Converence.

Park, J., Kim,S. 2010. *Driving Control of Mobile Robot with Mecanum Wheel using Fuzzy Inference System*.Word Academy of Science Engineering Technology, Vol.6, pp.2519-2523.

Sanyo. 1996. *STK6713BMK4 Unipolar Fixed Current Chopper 4 Phase Stepping Motor Driver*. Sanyo Electric, CO. LTD.

Silabs. 2008. Stepper Motor Reference Design. www.silabs.com. Diakses tanggal: 8 Juni 2014.

Song, J.B., Byun, K.S. 2004. *Design and Control of a Four-Wheeled Omnidirectional Mobile Robot withSteerable Omnidirectional Wheels*. Journal of Robotic Systems, Vol. 21, pp. 193-208.

ST. 2001. *L297 Stepper Motor Controller*. www.st.com. Diakses tanggal: 18 Juni 2014.

Tlale, Nkgatho. 2008. *Kinematics and Dinamics Modelling of a Mecanum Wheeled Mobile Platform.*IEEE Mechatronics and Machine Vision in Practice, Vol.15, pp.657-662.

Tzafestas, Spyros. 2013. *Introduction to Mobile Robot Control.* Elsevier Insight.

Weinstar. 2009. *Weinstar Display 2x16 LCD*. www.weinstar.com.tw. Diakses tanggal: 20 Juli 2014.

West, M., Asada, H.. 1997. *Design of Ball Wheel Mechanisms for Omnidirectional Vehicle with Full Mobility and Invariant Kinematics*. Journal of Mechanical Design, Vol.119, pp.153-161.

LAMPIRAN I

Dokumentasi *mobile* robot

Gambar 1. Roda Mekanum

Gambar 2. Motor *Stepper*

Gambar 3. Elektrik Robot

Gambar 4. Robot Tampak Atas

Gambar 5. Robot Tampak Depan

Gambar 5. Robot Tampak Perspektif

LAMPIRAN II

Listing Program

LISTING PROGRAM MIKROKONTROLER MASTER

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.7 beta 5 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 10/11/2009
Author  : jusuf
Company : jusufcorp
Comments:


Chip type           : ATmega16
Program type        : Application
AVR Core Clock frequency: 16.000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 512
*****************************************************/
#include <mega16.h>          // mk
#include <alcd.h>            // lcd
#include <stdio.h>           // i/o
#include <spi.h>             // spi
#include <delay.h>           // delay
#include <math.h>            // aritmatik
#include <stdlib.h>          // library
#define SL1 PORTB.1          // pin master ss slave 1
#define SL2 PORTB.2          // pin master ss slave 2
```

```
#define SL3 PORTB.3          // pin master ss slave 3
#define SL4 PORTB.4          // pin master ss slave 4

#define sw_1 PINC.4          // mode 1
#define sw_2 PINC.5          // mode 2
#define sw_3 PINC.6          // mode 3
#define sw_4 PINC.7          // mode 4

#define maju 0               // arah maju
#define mundur 1             // arah mundur
#define diam 2               // arah diam

#define phi 3.14159          // masuk perhitungan
#define radius 0.05          // radius roda
#define d_LR 0.34            // konstanta a
#define e_FB 0.34            // konstanta b

char lcd[16];                // lcd 16 karakter
int a,i, count;              //  2 byte -32.768/32.768 count pengitung robot mati

float SP[5],W_roda[5];       // 3 byte 1.2*10-38/3.410+38, kinmtk, array
float Vx,Vy,W;               // robot kordinat
char arah1,arah2,arah3,arah4,;       // arah W

char penanda1,penanda2,nomor;        // utilitas menu LCD

void mecanum_kinematik(void);                // main kinematik
void arah_pergerakan(float V_X, float V_Y, float WW);    // main arah
void kirim(void);                            // main spi

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
```

```
{
// Place your code here

   switch(penanda1)                          // utilitas tombol mode ++/---
   {
         case 0:                             // penanda1 == 1
               if (PINC.5==1)penanda1=1;
                     else penanda1=0; break;
            case 1:                          // penanda1 == 2, nomor --
               if (PINC.5==1){penanda1=2;nomor--;}
                     else penanda1 = 0;
            case 2:                          // penanda1 == 2
               if (PINC.5==1){penanda1=2;}
                     else penanda1 = 0;
      }

   switch(penanda2)
   {
         case 0:                             // penanda2 == 1
               if (PINC.6==1)penanda2=1;
                     else penanda2=0; break;
            case 1:                          // penanda2 == 2, nomor ++
               if (PINC.6==1){penanda2=2;nomor++;}
                     else penanda2 = 0;
            case 2:                          // penanda2 == 2
               if (PINC.6==1){penanda2=2;}
                     else penanda2 = 0;
      }

}

// Timer2 overflow interrupt service routine
```

```
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
// Place your code here
   mecanum_kinematik();          // sub kinemtk
   kirim();                      // sub spi
   count++;                      // count robot
   TCNT2=100;                    // reg tcnt *cacah* tim 2 == 100
}

void main(void)
{
PORTB=0x1E;                      // pull up port.b 1234
DDRB=0xBE;                       // out port.b 123457

DDRD=0x40;                       // out port.d 6

TCCR0=0x03;                      // tim 0  mode 0 normal tp 0xff 250.000 kHz
TCNT0=0x00;                      // tim 0 data cacahan
OCR0=0x00;                       // pembanding  TCNT

// 15625 kHz
ASSR=0x00;                       // async stat reg
TCCR2=0x07;                      // clock value 15625 kHz
TCNT2=0x00;                      // tim count reg
OCR2=0x00;                       // out com reg

TIMSK=0x01;                      // tim intrp mask reg, timer overflow int enbl

UCSRA=0x00;                      // usart contrl stat reg
UCSRB=0x08;                      // data 8
UCSRC=0x86;                      // async parity 0, 1 stop
UBRRH=0x00;                      // baud rate
```

```
UBRRL=0x67;                    // baud rate 9600

// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 4000.000 kHz
// SPI Clock Phase: Cycle Start
// SPI Clock Polarity: Low
// SPI Data Order: LSB First
SPCR=0x70;          // aktiv spi, lsb dulu, master
SPSR=0x00;          // spi stat reg

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTA Bit 0
// RD - PORTA Bit 1
// EN - PORTA Bit 2
// D4 - PORTA Bit 4
// D5 - PORTA Bit 5
// D6 - PORTA Bit 6
// D7 - PORTA Bit 7
// Characters/line: 20
lcd_init(16);

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

delay_ms(500);

nomor=1;            // variable nomor utilitas mode =1
menu:               // menu lcd
count=0;            // nilai variable count awal diset 0
```

```
arah_pergerakan(0, 0, 0);              .// pergerakan robot input 0

while(1){

  TIMSK=0x01;                          // int overflow enable, tim 0

  lcd_clear();                         // clear lcd
  lcd_gotoxy(0,0);                     // menuju kolom, baris lcd 0,0
  lcd_putsf(" Vx -- Vy -- W");         // tampilkan Vx -- Vy -- W

      // counter nomor tombol mode ++/-- dihasilkan nomor
  if(nomor==1)arah_pergerakan(0.7, 0, 0);
  else if(nomor==2)arah_pergerakan(0, 0.6, 0);
  else if(nomor==3)arah_pergerakan(0.4, 0.4, 0);
  else if(nomor==4)arah_pergerakan(0.4, -0.4, 0);
  else if(nomor==5)arah_pergerakan(-0.7, 0, 0);
  else if(nomor==6)arah_pergerakan(-0.4, -0.4, 0);
  else if(nomor==7)arah_pergerakan(0, -0.6, 0);
  else if(nomor==8)arah_pergerakan(-0.4, 0.4, 0);

  while(PINC.4==0){                    // pinc.4=0, jika pinc.5/6=1 menuju pilih
    if((PINC.5==1)||(PINC.6==1)) goto pilih;
  };
  goto keluar;

  pilih :
                                       // fungsi mode saat count nomor =9
  if(nomor==9)nomor=1;
  else if(nomor==0)nomor=8;

  delay_ms(10);
}
```

```
keluar:
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("    MULAI    ");
delay_ms(1000);
count=0;
TIMSK=0x41;       // overflow tim 2, 0
while (1)
  {
  if(count>300){                    // berhientikan robot 3 sekon
   TIMSK=0x01;                      // tim 0 overflow
   arah_pergerakan(0, 0, 0);        // pergerakan robot = 0
   lcd_gotoxy(0,0);                 // kolom, baris lcd 0,0
   lcd_putsf("   BERHENTI    ");    // lcd saat robot berhenti
   if(PINC.7==1) goto menu;         // ketika pinc.7=1, menuju menu lcd
   count=510;                       // variable count diset 510
   mecanum_kinematik();             // sub fungsi perhitungan kinematik
   kirim();                         // sub fungsi kirim spi
  }
  else {
   TIMSK=0x41;                      // overflow tim 2, 0

   if(PINC.7==1) goto menu;         // ketika pinc.7=1, menuju menu lcd

   lcd_gotoxy(1,0);                 // tampilan lcd Vx
   ftoa(Vx,1,lcd);
   lcd_puts(lcd);

   lcd_gotoxy(7,0);                 // tampilan lcd Vy
   ftoa(Vy,1,lcd);
   lcd_puts(lcd);
```

```
        lcd_gotoxy(13,0);                   // tampilan lcd w
        ftoa(W,1,lcd);
        lcd_puts(lcd);

        lcd_gotoxy(0,1);                    // tampilan kec sudut1
        ftoa(SP[1],1,lcd);
        lcd_puts(lcd);

        lcd_gotoxy(4,1);                    // tampilan kec sudut2
        ftoa(SP[2],1,lcd);
        lcd_puts(lcd);

        lcd_gotoxy(8,1);                    // tampilan kec sudut3
        ftoa(SP[3],1,lcd);
        lcd_puts(lcd);

        lcd_gotoxy(12,1);                   // tampilan kec sudut4
        ftoa(SP[4],1,lcd);
        lcd_puts(lcd);

        delay_ms(50);

        lcd_clear();
      }
    }
}

void mecanum_kinematik(void)
{
```

```
    SP[1] = (1/radius)*(Vx - Vy - W*((d_LR + e_FB)/2));      // pers kinematik
    SP[2] = (1/radius)*(Vx + Vy + W*((d_LR + e_FB)/2));
    SP[3] = (1/radius)*(Vx + Vy - W*((d_LR + e_FB)/2));
    SP[4] = (1/radius)*(Vx - Vy + W*((d_LR + e_FB)/2));

// arah W, sp1,2,3,4<0 mundur
        if(SP[1]<0) {arah1=mundur; SP[1]=SP[1];}
            else arah1=maju;
        if(SP[2]<0) {arah2=mundur; SP[2]=SP[2];}
            else arah2=maju;
        if(SP[3]<0) {arah3=mundur; SP[3]=SP[3];}
            else arah3=maju;
        if(SP[4]<0) {arah4=mundur; SP[4]=SP[4];}
            else arah4=maju;

}

void arah_pergerakan(float V_X, float V_Y, float WW)
{

        Vx=V_X;                         //sumbu pergerakan robot Vx
        Vy=V_Y;                         //sumbu pergerakan robot Vy
        W=WW;                           //sumbu pergerakan robot W

    lcd_gotoxy(1,1);                    // tampilan lcd Vx
    ftoa(Vx,1,lcd);
    lcd_puts(lcd);

    lcd_gotoxy(7,1);                    // tampilan lcd Vy
    ftoa(Vy,1,lcd);
    lcd_puts(lcd);
```

```
    lcd_gotoxy(13,1);                    // tampilan lcd W
    ftoa(W,1,lcd);
    lcd_puts(lcd);

    }

void kirim()                             // kirim spi
{
        // arah kec sudut
    SL1=0;                          // sl1=0, kirim kecepatan, sl1 di 1 lagi
    spi(arah1);
    SL1=1;
    SL2=0;                          // sl2=0, kirim kecepatan, sl2 di 1 lagi
    spi(arah2);
    SL2=1;
    SL3=0;                          // sl3=0, kirim kecepatan, sl3 di 1 lagi
    spi(arah3);
    SL3=1;
    SL4=0;                          // sl4=0, kirim kecepatan, sl4 di 1 lagi
    spi(arah4);
    SL4=1;

        // kec sudut nilai
    SL1=0;                          // sl1=0, kirim kecepatan, sl1 di 1 lagi
    spi(SP[1]);
    SL1=1;
    SL2=0;                          // sl2=0, kirim kecepatan, sl2 di 1 lagi
    spi(SP[2]);
    SL2=1;
    SL3=0;                          // sl3=0, kirim kecepatan, sl3 di 1 lagi
    spi(SP[3]);
    SL3=1;
```

```
   SL4=0;                    // sl4=0, kirim kecepatan, sl4 di 1 lagi
   spi(SP[4]);
   SL4=1;
}
```

LISTING PROGRAM MIKROKONTROLER SLAVE

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.7 beta 5 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 10/11/2009
Author  : jusuf
Company : jusufcorp
Comments:


Chip type           : ATmega8
Program type        : Application
Clock frequency     : 16.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega8.h>
#include <stdio.h>
```

```
int a,W,V;                          // variable a= set spi, W data spi
float f_OCR;                        // frekuensi OCR
int data, arah;                     // variable data dan arah
float rpm_roda,rpm_motor,f_stepper;         // kec roda, kec motor, frek stepper

// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
   if(PORTC.0==0)PORTC.0=1;                 // toggle pinc.0 -> frekuensi
   else if(PORTC.0==1)PORTC.0=0;
}

// SPI interrupt service routine
interrupt [SPI_STC] void spi_isr(void)
{
   data=SPDR;                       // data dan arah spi, a set 0, dikembalikan 1 lagi
   if(a==0){
      arah=SPDR;
      a=1;
   }
   else if(a==1){                   // jika a nilai -, maka w= 256- data spdr
      if(arah==1)W=256-SPDR;
      else W=SPDR;
      a=0;
   }
}

void main(void)
{
PORTB=0x00;                         // portb set awal 0
DDRB=0x10;                          // portb.4 input
PORTC=0x00;                         // portc set 0
```

```
DDRC=0x03;                    // portc output

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
TCCR0=0x03;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 62.500 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x0C;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x0F;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
```

```
TIMSK=0x11;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

// SPI initialization
// SPI Type: Slave
// SPI Clock Phase: Cycle Start
// SPI Clock Polarity: Low
// SPI Data Order: LSB First
SPCR=0xE0;
SPSR=0x00;

// Clear the SPI interrupt flag
#asm
   in   r30,spsr
   in   r30,spdr
#endasm

// TWI initialization
// TWI disabled
TWCR=0x00;
```

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

while (1){

      rpm_roda=(W*60)/6.24;                  // rumus kec roda
      rpm_motor=(rpm_roda*10)/8;             // rumus kec motor

// tampilan lcd jika arah mundur
      if(arah==1) {
         printf("MUNDUR - Kec Sudut = %d - rpm_roda = %d - rpm_motor = %d
- f_step = %d\n\n", -W,(int)rpm_roda,(int)rpm_motor,(int)(rpm_motor/(0.15)));
      }
// tampilan lcd arah maju
      else {
          printf("MAJU - Kec Sudut = %d - rpm_roda = %d - rpm_motor = %d -
f_step = %d\n\n", W,(int)rpm_roda,(int)rpm_motor,(int)(rpm_motor/(0.15)));
      }

//  output pin arah menuju pin dir l297
      if(arah==0)PORTC.1=0;
      else if(arah==1)PORTC.1=1;

// rumus OCR, int 16 bit konversi 8 bit -> nilai geser 8 bit
      OCR1AH=(((int)(1/((rpm_motor/(0.15))*0.000032)))&0xFF00)>>8;
      OCR1AL=((int)(1/((rpm_motor/(0.15))*0.000032)))&0xFF;

   }
}
    };
}
}
```

LAMPIRAN III

Datasheet