TA/SEKJUR/TE/2011/002

ROBOT PENGANGKUT BARANG BERDASARKAN WARNA BERBASIS ATMEGA 32

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh

Gelar Sarjana Teknik Elektro

Disusun oleh :

Nama : **DRAJAT RESTU NURSIGIT**

No.Mahasiswa : **05524004**

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2011

LEMBAR PENGESAHAN PEMBIMBING

ROBOT PENGANGKUT BARANG BERDASARKAN WARNA

BERBASIS ATMEGA 32

TUGAS AKHIR

Disusun oleh :

Nama : **DRAJAT RESTU NURSIGIT**

No.Mahasiswa : **05 524 004**

Yogyakarta, Januari 2011

Pembimbing I	Pembimbing II
Medilla Kusryanto, ST,.MEng.	**Dwi Ana Ratna Wati, ST.M.Eng.**

LEMBAR PENGESAHAN PENGUJI

ROBOT PENGANGKUT BARANG BERDASARKAN WARNA

BERBASIS ATMEGA 32

TUGAS AKHIR

OLEH :

Nama : **DRAJAT RESTU NURSIGIT**

No.Mahasiswa : **05 524 004**

Telah di Pertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana pada Jurusan Teknik Elektro Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Januari 2011

Tim Penguji

Dwi Ana Ratna Wati S.T.,M.Eng
Ketua

Tito Yuwono S.T.,M.Sc
Anggota I

Medilla Kusriyanto S.T.,M.Eng
Anggota II

Mengetahui,
Ketua Jurusan Teknik Elektro
Fakultas Teknologi Industri Universitas Islam Indonesia

(Tito Yuwono S.T., M.Sc.)

HALAMAN PERSEMBAHAN

Segala Puji bagi Allah SWT, atas Rahmat , Ridho dan Karunia-Nya
Atas kekuatan dan cahaya terang padaku
Semua cobaan dan kesabaran yang Aku syukuri dari-Nya
Segala sesuatu dalam karya yang sederhana ini dapat terselesaikan dengan selalu beriman dan bertaqwakepada-Nya

terima kasih,
Ayahku... dan Bundaku ....
Atas segala doa, nasehat , cinta dan kasihnya, pengorbanan, dan perhatiannya Yang telah diberikan takkan terbalas oleh apapun dan sampai kapan pun Atas didikan, pengalaman, kepribadian dari yang kalian ajarkan...
Aku bisa kuat dan berusaha untuk maju sampai sekarang.....
Atas segala doa , perhatiannya serta cinta kasihnya..
Seluruh keluarga besarku
Atas segala harapan, doa dan dukungan yang diberikan dalam setiap langkahku
Sahabat-sahabat dan orang-orang terdekat
Atas segala pengalaman , ketulusan , bantuan dan kebersamaan.

MOTTO

Dan (ingatlah juga), takkala Tuhanmu memaklumkan ; "Sesungguhnya jika kamu bersyukur, pasti kami akan menambah (nikmat) kepadamu, dan jika kamu mengingkari (nikmat-Ku), Maka Sesungguhnya azab-Ku sangat pedih"

(QS. Ibrahim : 7)

" Sungguh bersama kesukaran itu pasti ada kemudahan. Sungguh, oleh karena itu jika kamu telah selesai dari suatu tugas, kerjakan tugas lain dengan sungguh-sungguh. Dan hanya kepada Tuhanmulah kehendaknya kamu memohon dan mengharap "

(QS. Asy-Syarh 5-8)

" Jadilah sabar dan sholat sebagai penolongmu, sesungguhnya Allah beserta orang-orang yang sabar "

(Q.S. Al Baqarah ayat 153).

"Manusia yang berguna adalah manusia yang mampu memberikan manfaat untuk orang banyak

KATA PENGANTAR

Assalamu'alaikum warahmatullahi wabarakatuh

Puji syukur yang sebesar-besar nya penulis haturkan ke hadirat Allah SWT, yang telah melimpah rahmat dan hidayah-Nya, sehingga laporan tugas akhir yang berjudul " **ROBOT PENGANGKUT BARANG BERDASARKAN WARNA BERBASIS ATMEGA 32"** dapat tereselesaikan dengan baik. Laporan tugas akhir ini disusun sebagai syarat untuk memperoleh gelar Sarjana Teknik Elektro Universitas Islam Indonesia.

Penulis menyadari bahwa dalam masa pembuatan laporan tugas akhir ini tidak lepas dari bantuan beberapa pihak. Oleh karena itu penulis mengucapkan banyak terimakasih kepada :

1. Allah SWT yang selalu ada dalam setiap niat dan perkerjaan dalam menyelesaikan laporan tugas akhir ini, dan dengan izin dan kuasa-Nya di berikan kesempatan dan kemudahan dalam menyelesaikan laporan tugas akhir ini.
2. Nabi Muhammad SAW, yang membawa umatnya dari zaman kegelapan sampai zaman terang menderang sampai saat ini kita rasakan.

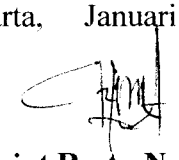
3. Ayahanda Ramelan dan ibunda Sumiati yang senantiasa memberikan dukungan baik moril maupun materil dan do'a. yang selalu memberikan kasih sayang yang tiada tara.
4. Bapak Medilla Kusryanto ST.M.Eng selaku Dosen pembimbing I, yang selalu memberikan arahan dan bimbingannya dalam pembuatan alat dan laporan tugas akhir.
5. Ibu Dwi Ana Ratnawati ST, M.Eng, selaku dosen pembingbing II dan dosen Pengampu Akademik, yang telah memberikan saran dan masukan selama dalam pembuatan laporan tugas akhir.
6. Kakaku Nur yang selalu memberikan semangat dan dorongan dalam menyelesaikan laporan tugas akhir.
7. Segenap Dosen Jurusan Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia, yang telah memberikan ilmu selama penulis duduk dibangku kuliah.
8. Rekan-rekan yang selalu mendukung dan membantu, serta teman teman Keluarga *ELCO '05* yang selalu kompak.
9. Temen-teman di Bascamp Techno yang membantu memberikan ide-ide teknologi.
10. Serta semua pihak yang telah membantu, yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa masih banyak kekurangan, yang ada pada laporan tugas akhir ini. Sehingga penulis mengharapkan kritik dan saran yang sifatnya membangun dan mengembangkan.

Akhir kata penulis sampaikan pula harapan semoga Tugas akhir ini dapat memberi manfaat yang cukup berarti khususnya bagi penulis dan bagi pembaca pada umumnya. Semoga Allah SWT senantiasa selalu memberikan rahmat dan hidayah-Nya kepada kita semua. Amiin.

wassalamu'alaikum warahmatullahi wabarakatuh

Yogyakarta, Januari 2011

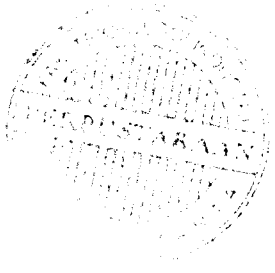


(Drajat Restu Nursigit)

ABTRAKSI

Tumbuhnya teknologi dengan tuntutan efisiensi kerja terutama dalam hal waktu dan efisiensi energi pada distribusi barang. Robot pengangkut barang dengan menggunakan prinsip pengikut garis yang dilengkapi dengan sensor warna telah di buat. Robot berfungsi sebagai tukang angkut barang yang bergerak secara otomatis mengikuti garis arena yang telah ditentukan. Robot yang digunakan untuk membawa dan meletakan objek dari satu tempat ke tempat lain digudang produksi industri. Sistem yang terdiri dari beberapa sensor *photodioda* untuk sensor warna dan sensor garis, dan sensor jarak digunakan untuk mendeteksi keberadaan objek yang akan dipindahkan. Robot akan mendeteksi objek dan mengambil data warna objek sebagai mengambil keputusan kemana objek akan dipindahkan sesuai pemetaan dalam program. Setelah melakukan pengujian beberapa sampel warna dan melakukan pengujian di arena yang telah dibuat robot dapat menyelesaikan dengan baik sesuai dengan algoritma program yang telah dimasukan kedalam mikrokontroler Atmega32.

Kata kunci : robot pengkut barang, Sensor Warna, Sensor jarak



DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemilihan warna terkadang menjadi suatu masalah dalam menentukan ukuran warna jika dilihat secara langsung oleh mata manusia, hal ini terjadi jika mata manusia tersebut terkana buta warna, namun masih dapat melihat beberapa warna tertentu. Prisip dasar warna dikelompokkan menjadi 3 warna dasar yaitu R-G-B *(Red,Green,Blue)* ini adalah ruang warna asli yang digunakan pada sistem grafis komputer. Ketiga warna dasar tersebut memiliki nilai angka dari warna tersebut sehingga akan membantu manusia dalam menentukan warna.

Untuk menentukan warna tersebut dibutuhkan sensor sebagai pendeteksi warna untuk mengubah menjadi nilai atau ukuran-ukuran warna, dari penjelasan tersebut maka akan dibuat sebuah perancangan pengendalian robot pendeteksi warna dan memindahkan barang secara otomatis.

Pengendalian robot sudah sangat beragam dan biasanya tergantung pada tujuan dan biaya yang akan dikeluarkan. Pengendalian dapat dilakukan dengan dua cara yaitu dengan cara manual dengan penekanan tombol kendali dan penerapan sensor sebagai pengendali tergantung situasi yang akan dikendalikan secara otomatis.

Dalam tugas akhir ini akan diangkat permasalahan dalam pemisahan dan pemindahan barang yeng memiliki beda warna, dalam pengendalian robot yang akan

dirancang adalah robot pengangkut barang berdasarkan warna, sehingga dalam penyelesainya robot dapat membantu kerja manusia dalam pemisahan dan pengangkutan barang dengan beda warna.

Robot yang dirancang ini bekerja secara otomatis yang dikendalikan oleh program yang tanamkan pada *chip IC* menggunakan mikrokontroler AVR Atmega32 sebagai pengendali robot. Keunggulan dari rancangan tugas akhir ini adalah pengendalian secara otomatis yang terprogram dan pembacaan warna RGB memiliki 8 bit untuk masing-masing ketiga warna dasar, pembacaan nilai warna mencapai 0 s/d 255 sehingga pembacaan warna akan lebih banyak.

1.2 Rumusan Masalah

Bedasarka uraian latar belakang diatas maka dapat dirumuskan beberapa permasalahan, yaitu sebagi berikut:

1. Merancang dan membuat sebuah sistem robot yang secara otomatis dapat memimindahkan barang yang berbeda warna dan menempatkan pada lokasi yang ditentukan.
2. Merancang sensor pembaca garis atau yang sering disebut dengan *Line Tracking.*
3. Merancang sensor warna untuk mendeteksi warna objek yang akan di pindahkan oleh robot.

1.3 Batasan Masalah

Agar permasalahan yang dibahas sesuai dengan tujuan judul yang di tetapkan dan serta tujuan yang diinginkan, maka perlu ditetapkan pokok-pokok permasalahan yang akan dibahas. Batasan masalah dari penulisan ini dibatasi pada ruang lingkup pembahasan sebagai berikut:

1. Sistem menggunakan mikrokontroler sebagai pengendali robot
2. Menggunakan sensor pembaca garis lintasan untuk menentukan area pemisahan warna barang.
3. Menggunakan modul sensor warna yang dirancang menggunakan sensor *photodioda* dan LED warna.

1.4 Tujuan Tugas Akhir

Tulisan ini bertujuan untuk merancang dan membuat sistem robot otomatis dengan mikrokontroler ATmega32 sebagai pengendali dan menggunakan bantuan sensor warna untuk mendeteksi warna objek dan sensor pembaca garis lintasan.

1.5. Manfaat Penelitian

Manfaat dari penelitian ini adalah untuk memperdalam pengetahuan dan pemahaman tentang prinsip dan cara kerja dalam merancang serta membuat suatu sistem robot otomatis pemindahan barang berdasarkan warna, dan mengembangkan beberapa referensi dalam pengembangan robotika di Jurusan Teknik Elektro.

1.6. Sistematika Penulisan Tugas Akhir.

Sistemmatika dalam penulisan dan pembahasan laporan tugas akhir ini adalah dapat di jelaskan sebagai berikut:

BAB I PENDAHULAN

Bab Pendahuluan berisi tentang Latar Belakang Masalah, Maksud dan Tujuan, Perumusan Masalah, Batasan Masalah, Langkah Penelitian, dan Sistematika Penulisan Laporan.

BAB II TINJAUAN PUSTAKA

Pada bab ini berisi tentang *Literature survey* tentang penelitian sejenis yang telah dilakukan sebelumnya. Analisis, kesimpulan, saran, komentar penelitian sejenis yang telah dilakukan sebelumnya. Penjelasan mengenai kontribusi penelitian yang akan dikerjakan dalam tugas akhir.

BAB III PERANCANGAN SISTEM

Bagian ini menjelaskan metode-metode perancangan yang digunakan, cara mensimulasikan rancangan dan pengujian sistem yang telah dibuat, pembagian fungsi kerja dalam diagram blok serta berisi lebih terperinci tentang apa yang telah disampaikan pada proposal tugas akhir . Penjabaran indikator unjuk kerja sistem : bagaimana validasi atau pengujian sistem akan dilakukan.

BAB IV ANALISA DAN PEMBAHASAN

Bab ini membahas tentang hasil pengujian dan analisis dari sistem yang dibuat dibandingkan dengan dasar teori sistem atau sistem yang lain yang dapat dijadikan sebagai pembanding. Pengujian sistem berdasarkan indikator unjuk kerja yang telah dijelaskan sebelumnya

BAB V PENUTUP

Bagian ini menjelaskan kesimpulan dari tugas akhir yang telah selesai dikerjakan berdasarkan analisis dan pembahasan di bab sebelumnya. Saran untuk pengembangan dan penelitian lebih lanjut

BAB II

TINJAUAN PUSTAKA

2.1. Studi Pustaka

Pada perancangan sebelumnya yaitu dari penelitian dengan judul Konveyor Pemisah Barang Berdasarkan Warna oleh Mustiko Adji, dengan sistem kerja berbentuk konveyor, dalam penelitian berikut yaitu perancangan robot dalam pemisahan barang yang memiliki beda warna objek. Mikrokontroler yang digunakan yaitu ATmega32 yang tidak jauh berbeda dari segi fungsi dan konfigurasi input dan output nya, hanya pada kapasitas memori *flash* pada ATmega32 lebih besar. Untuk keseluruhan dari tujuan dibuat robot pemindah barang yaitu simulasi pemindahan barang dari arena pengambilan ke arena pemisahan warna barang. Dalam perancangan kali ini mengaplikasikan sensor warna dalam sebuah robot yang dapat bergerak mengikuti garis lintasan yang telah ditentukan warnanya.

2.2 Mikrokontroler

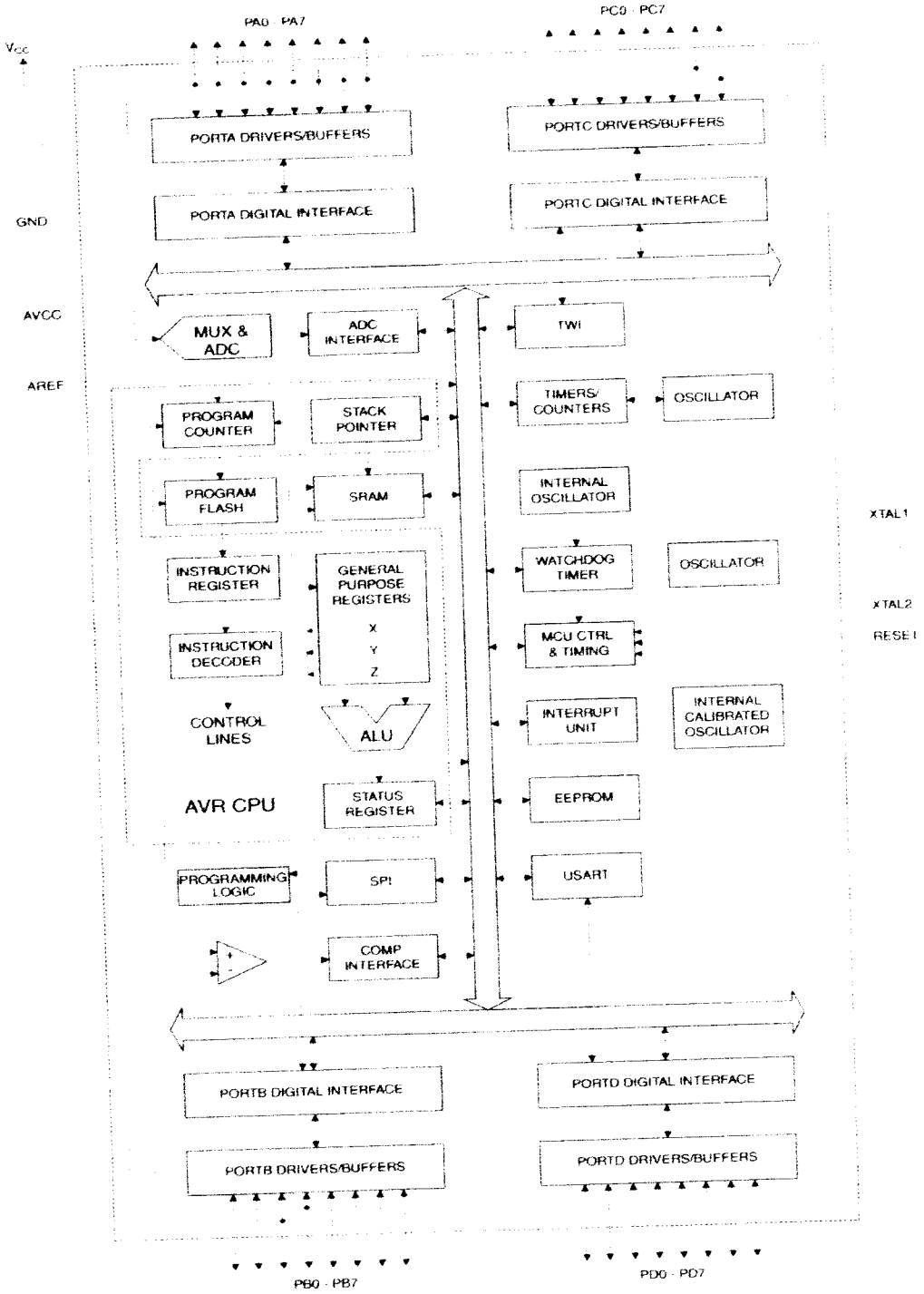
Perkembangan mikrokontroler telah maju dengan pesat dalam berkambang dunia elektronika, khususnya sistem kendali menggunakan mikrokontroler. Mikrokontroler dengan berbahan semikonduktor dalam penemuan silikon menyebabkan pengembangan dalam bidang ini memberikan kotribusi yang sangat berharga bagi perkembangan teknologi dimasa modern sekarang ini.

Pengembang mikrokontroler yang dikembangkan oleh produsen ATMEL yang telah banyak memberikan kontribusi besar dalam mengembangkan dan memasarkan.

Mikrokontroler yang digunakan dalam penelitian ini yaitu menggunakan ATmega32, yang merupakan generasi AVR *(Alf and Vegard's Risc Prosesor)*. Mikrokontroler AVR memeiliki arsitekture *RISC* 8 bit, dimana dalam setiap instruksi dikemas dalam kode 16-bit *(16-bit works)* dan sebagian besar setiap instruksi dieksekusi dalam 1 siklus clock, dengan demikian banyaknya sintak program dan perhitungan yang dimiliki akan semakin cepat dalam penyelesainya. Fitur lain nya adalah memiliki clock internal yang sudah terpasang pada chip ATmega32 dan hal ini akan mempermudah para desainer yang menggunakan kendali mikrokontroler dapat memilih clock internal atau eksternal tergantung kebutuhan.

2.2.1 Arsitekture Mikrokontroler ATmega32

Blok diagram arsitekture ATmega32, hampir sama dengan yang dimiliki mikrokontroler AVR ATmega16, ATmega8535, dapat dilihat pada gambar di bawah ini :



Gambar 2.1. Arsitekture AVR ATmega32

Dari gambar blok diagram tersebut dapat dilihat ATmega32 memiliki :

1. Saluran Input dan output sebanyak 32 buah, yang dibagi menjadi 4 buah PORT yaitu Port A, Port B, Port C, Port D.
2. Memiliki ADC internal 10 bit dalam 8 chanel pada Port A.
3. Memiliki 3 buat *timer / counter* dalam kemampuan pembanding.
4. CPU yang terdiri atas 32 register.
5. *Wacthdog Timer* dengan osilator internal
6. Memori SRAM 2 kbyte
7. Memori Flash 32 kbyte dengan kemampuan *read while write*
8. Memori EEPROM sebesar 1024 byte yang dapat diprogram saat beroprasi dan dapat diprogram tanpa menggunakan catu daya.
9. Unit interupsi internal dan eksternal.
10. PORT antarmuka SPI
11. Antarmuka komparator Analog
12. PORT USART komunikasi serial.

2.2.2 Fitur ATmega32

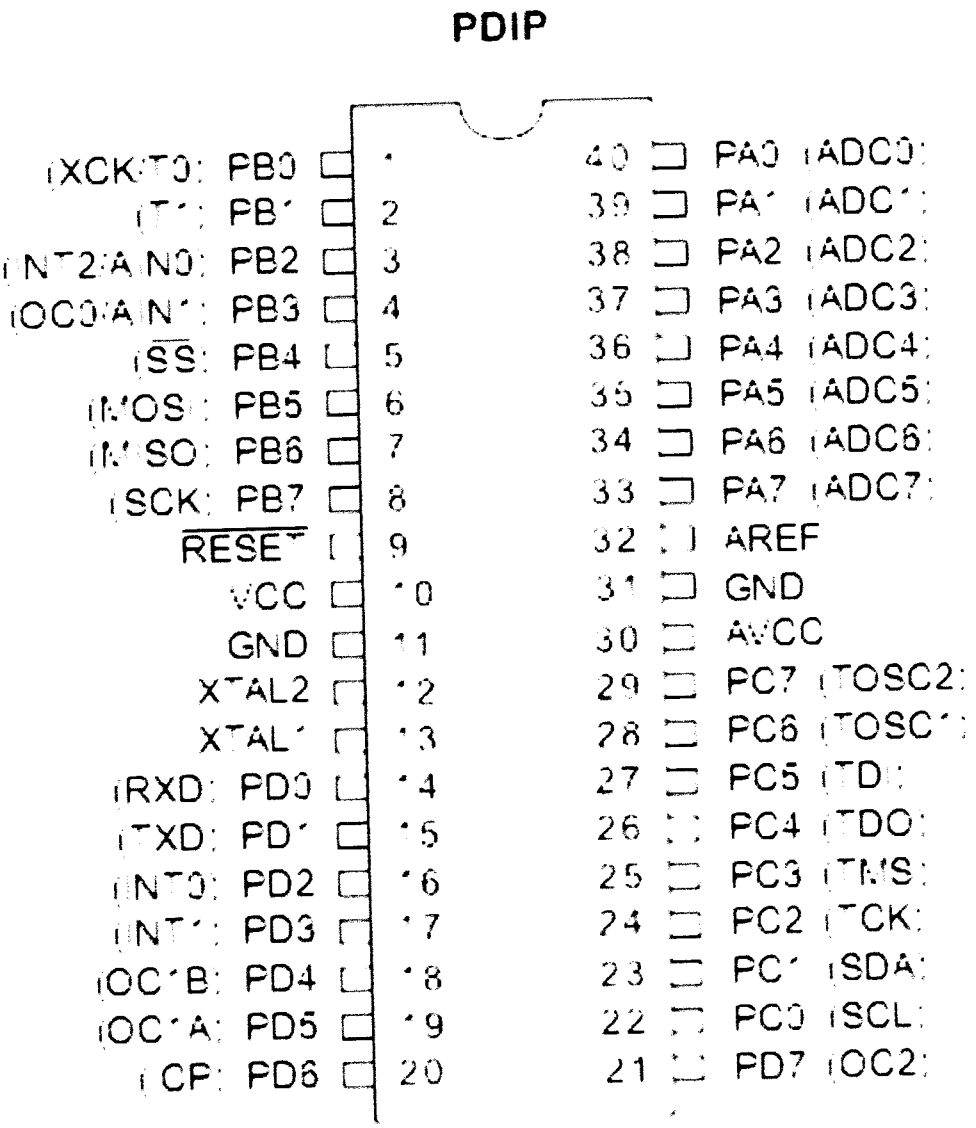
Kapabilitas data dari ATmega32 adalah sebagai berikut:

1. Sistem mikroprosesor 8 bit berbasis *RISC* dengan kecepatan maksimal 16 MHz.
2. Kapasitas Memori flash 32 KB, SRAM 2048 byte, dan EEPROM 512 byte.
3. ADC 10 bit dalam 8 *chanel* pada Port A.
4. PORT USART komunikasi serial memiliki kecepatan 2,5 Mbps
5. Enam pilihan *mode sleep* untuk menghemat daya listrik.

2.2.3 Konfigurasi Pin ATmega32

Konfigurasi pin mikrokontroler dibagi menjadi 4 port yang terdiri dari Port A, Port B, Port C, Port D, masing masing port memiliki 8 jalur input dan output dan set pada setiap pin menurut kebutuhannya.

Konfigurasi pin Atmega32



Gambar 2.2. Pin Mikrokontroler ATmega32

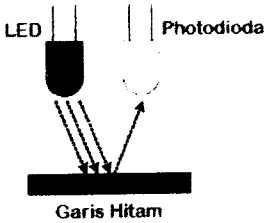
Dari gambar terbut dapat dijelaskan fungsi dari masing-masing pin :

Tabel 2.1 Konfigurasi pin ATMega32

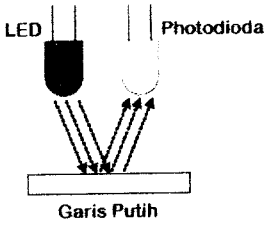
No Pin	Nama	Fungsi
1	PB0 (XCK/TO)	Port B.0 / *Timer-Counter* 0 dan *clock eksternal* untuk USART (XCK)
2	PB1 (T1)	Port B.1 / *Timer-Counter* 1
3	PB2 (INT2/AIN0)	Port B.2 / *Input* (+) Analog komparator (AIN0) dan interupsi eksternal 2 (INT2)
4	PB3 (OC0/AIN1)	Port B.3 / Input (-) Analog komparator (AIN1) dan *output* pembanding *timer/counter* (OC0)
5	PB4 (SS)	Port B.4 / SPI *Slave* Select Input (SS)
6	PB5 (MOSI)	Port B.5 / SPI Bus Master *Out Slave In*
7	PB6 (MISO)	Port B.6 / SPI Bus Master *In Slave Out*
8	PB7 (SCK)	Port B.7 / sinyal clock serial SPI
9	RESET	Me-reset Mikrokontroller
10	VCC	Catu daya (+)
11	GND	Sinyal ground terhadap catu daya
12 - 13	XTAL 2 - XTAL 1	Sinyal input *clock* eksternal (kristal)
14	PD0 (RXD)	penerima data serial
15	PD1 (TXD)	pengirim data serial
16	PD2 (INT0)	Interupsi eksternal 0
17	PD3 (INT1)	Interupsi eksternal 1
18	PD4 (OC1B)	Pembanding *Timer-Counter* 1
19	PD5 (OC1A)	Pembanding *Timer-Counter*
20	PD6 (ICP1)	*Timer-Counter* 1 *Input*
21	PD7 (OC2)	Pembanding *Timer-Counter* 2
22	PC0 (SCL)	Serial bus clock line
23	PC1 (SDA)	Serial bus data *input-output*
24 - 27	PC2 – PC5	Tidak ada pin khusus
28	PC6 (TOSC1)	Timer osilator 1
29	PC7 (TOSC2)	Timer osilator 2
30	AVCC	Tegangan ADC
31	GND	Sinyal ground ADC
32	AREFF	Tegangan referensi ADC
33 - 40	PA0 (ADC0) – PA7 (ADC7)	Port A.0 – Port A.7 dan input untuk ADC (8 channel : ADC0 – ADC7)

2.3 Sensor Pembaca Garis

Sebuah robot bergerak membutuhkan beberapa informasi disekitar robot agar dapat bekerja sesuai dengan keadaan sekitar robot, dalam tugas akhir ini robot berjalan mengikuti jalur lintasan menggunakan sensor garis atau sensor *proximity*. prinsip kerjanya sederhana, hanya memanfaatkan sifat cahaya yang akan dipantulkan jika mengenai benda berwarna terang dan akan diserap jika mengenai benda berwarna gelap. Sebagai sumber cahaya menggunakan *Infrared*, dan untuk menangkap pantulan cahaya *Infrared*, menggunakan *photodioda*. Jika sensor berada diatas garis hitam maka *photodioda* akan menerima sedikit sekali cahaya pantulan *Infrared*. Tetapi jika sensor berada diatas garis putih maka *photodioda* akan menerima banyak cahaya pantulan. Berikut adalah ilustrasinya



Gambar 1. Cahaya pantulan sedikit



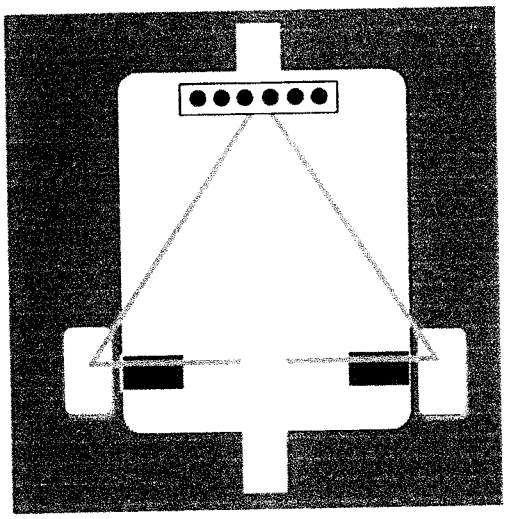
Gambar 2. Cahaya pantulan banyak

Gambar 2.3. Pencahayaan Sensor Garis

Sifat dari *photodioda* adalah jika semakin banyak cahaya yang diterima maka nilai resistansi diodanya semakin kecil. Dengan melakukan sedikit modifikasi, maka besaran resistansi tersebut dapat diubah menjadi tegangan. Sehingga jika sensor

berada diatas garis hitam, maka tegangan keluaran sensor akan kecil, demikian pula sebaliknya.

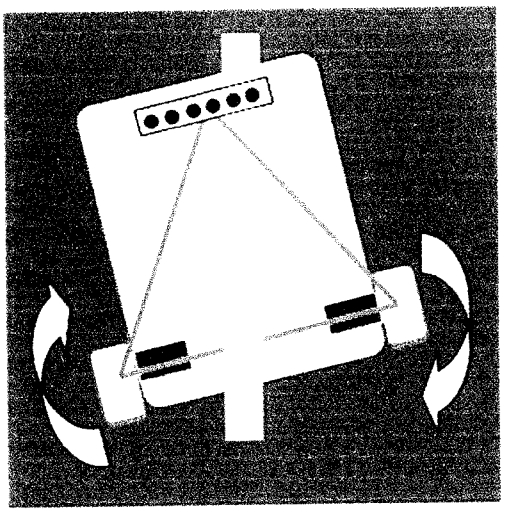
Sensor garis yang digunaka sebanyak 6 pasang sensor yang disusun lurus dan diletakan pada bagian depan roda penggerak, dalam penataan perlu diperhatikan posisi kedua roda belakang. Sesuai ditunjukan pada gambar 2.4. dibuat segitiga sama sisi untuk jarak antara kedua roda belakang dengan tengah sensor yang diletakan didepan. Hal ini dimaksudkan agar respon sensor dan gerakan robot lebih stabil.



Gambar 2.4. Peletakan Sensor Pembaca Garis

Dalam pengendalian robot jika posisi garis tepat berada ditengah atau dapat di kondisikan 001100 maka kecepatan kedua motor diprogram dengan kecepatan maksimal atau *full speed*, tetapi jika pada posisi miring atau sensor tengah tidak mengenai garis makan kecepatan motor kanan dan motor kiri tidak sama agar robot dapat menemukan sensor pada bagian tengah. Jika sensor mengenai garis posisi

sensor paling kanan maka motor kanan akan berputar kearah belakang dan motor kiri berputar kedepan sehingga robot bergerak kekanan garis sampai posisi lurus dengan garis lintasan



Gambar 2.5. Posisi Robot Scaning Garis

Jika posisi semua sensor tidak mengenai garis maka akan berjalan sesuai program yang di set sebelumnya yaitu robot akan belok mencari garis kekiri atau kekanan tergantung dari sensor mana yang terakhir kali menyetuh garis sampai mendapatkan kembali sensor garis yang berada ditengah robot, motode tersebut digunakan agar robot tidak mengalami eror baca garis yang semakin besar dan dapat dikatakan motode pengingat jalur terakhir.

2.4 Sensor Jarak (SRF04 Devantec Ring Finder)

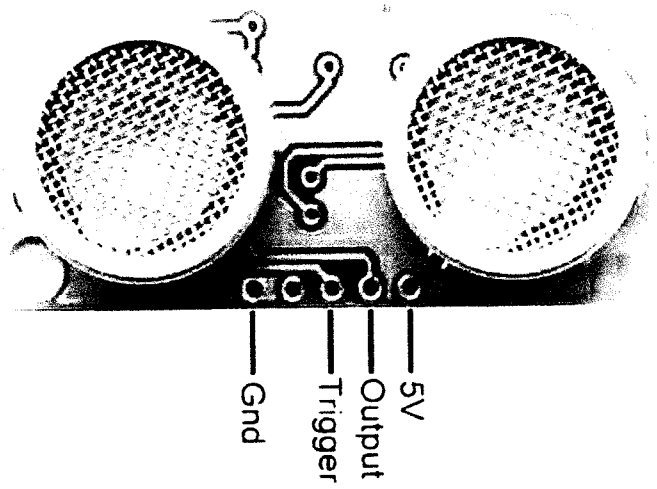
Sensor jarak menggunakan SRF04 dengan memanfaatkan pantulan suara ultrasonik yang dihasilkan oleh sensor ultrasonik, sensor yang tidak terpengaruh oleh cahaya, magnet dan suara. Walaupun sensor ini memanfaatkan suara namun dalam prosesnya sensor ultrasonik menggunakan frekuensi tertentu dan dilengkapi komponen pemfilter suara sehingga suara yang tidak akan terpengaruh oleh suara yang biasa didengar oleh telinga manusia.

Pembagian frekuensi suara dapat dibedakan menjadi 3 bagian yaitu :

1. Suara Infrasonik : suara kurang dari 20 Hz
2. Suara Audio : suara sekitar 20 Hz sampai dengan 20 KHz
3. Suara Ultrasonik : suara yang lebih dari 20 kHz

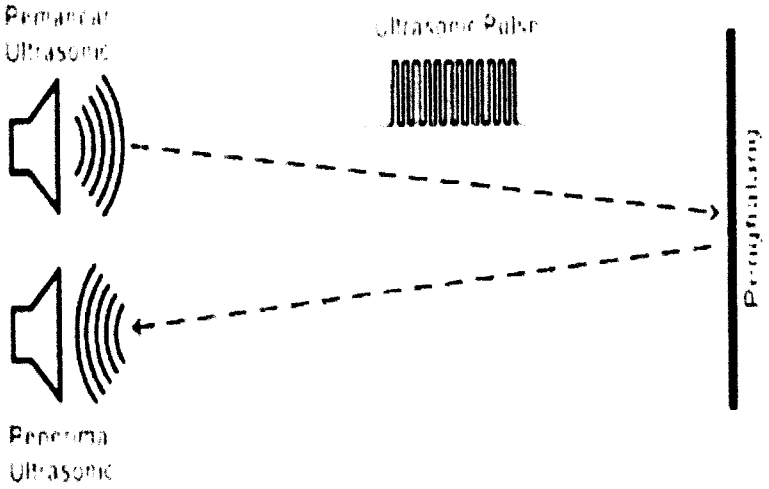
Telinga manusia hanya mampu mendengar suara dalam rentang frekensi antara 20Hz sampai dengan 20kHz. Sedangkan sensor SRF04 menggunakan suara ultrasonik 40kHz sehingga dalam frekuensi ini sensor tidak terganggu oleh suara audio, dan tidak mengganggu indra pendengaran manusia.

Sensor merupakan piranti yang dapat mengubah basaran tertentu yang ada dilingkungan sekitar menjadi besaran lain yang dapat dibaca oleh tranduser. Sensor ultrasonik adalah salah satu alat yang dapat memancarkan gelombang suara ultrasonik dan menerima kembali gelombang tersebut jika terjadi pantulan dari gelombang yang dipancarkan. Dengan pemanfaatan pantulan ini kita dapat menentukan jarak antara sensor dengan media pemantulan gelombang ultrasonik yang dipancarkan.



Gambar 2.6. Sensor jarak SRF04

Sensor SRF04 dapat dikelompokan dalam sensor jarak yang akurat dalam pembacaan jarak yang memiliki keakuratan minimal ± 4 cm dan jarak makasimal 300 cm, sensor ini tidak terpengaruh pada cermin jika dibandingkan dengan sensor PING buatan Parallax. Pada sensor ini memiliki 2 buah I/O yang terpisah, yaitu terminal *Triger* atau pemicu gelombang ultrasonik 40 kHz dan terminal output pantulan penerima yang menghasilkan besaran elektris tertentu. Gelombang suara sebesar 40 kHz akan dipancarkan selama 200 µs. Suara ini akan merambat diudara 344,424 m/detik (atau 1 cm setiap 29,034 µs) mengenai suatu objek pantul atau dinding.



Gambar 2.7. Pulsa pantul pada dinding

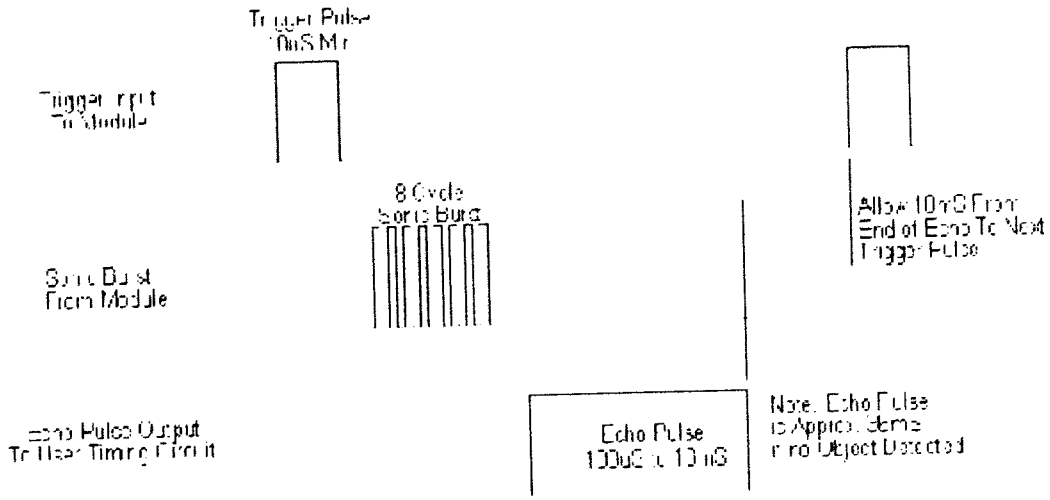
Selama menunggu pulsa pantulan sensor akan menghasilkan sebuah pulsa, pulsa ini akan berhenti (akan berkeadaan *low*) ketika suara pantulan terdeteksi oleh tranduser penerima. Dari hasil pantulan tersebut menghasilkan lebar pulsa yang nantinya mikro dapat mengolah lebar pulsa yang dihasilkan oleh tranduser penerima dan dikonversi dalam bentuk jarak dengan persamaan sebagai beriku :

Jarak = (lebar pulsa x 0,034442 μs) / 2 (dalam ukuran cm)

Jarak = (lebar pulsa / 29,034 μs) / 2 (dalam ukuran cm)

0,034442μs dan 29,034μs dihasilkan dari 1 /29,034μs = 0,034442μs

SRF04 Timing Diagram



Gambar 2.8 Timing diagram SRF04

2.5 Pengenalan Sensor Warna

Ketika memandang suatu benda, cahaya dari benda itu merambat langsung ke mata kita. Warna benda tersebut adalah sinar yang dipantulkannya. Misalnya sebuah benda berwarna merah tampak berwarna merah ketika benda tersebut dikenai sinar putih karena semua spektrum warna kecuali sinar merah diserap oleh benda. Jadi hanya sinar merah yang dipantulkan oleh benda sehingga benda tersebut akan tampak berwarna merah.

Warna sinar dibedakan menjadi dua macam, primer dan sekunder. Warna primer yaitu warna yang didapat tanpa ada pencampuran warna, warna primer adalah merah, hijau, dan biru. Warna sekunder adalah warna yang didapat dari penggabungan dua warna primer, yang termasuk warna sekunder adalah kuning, sian

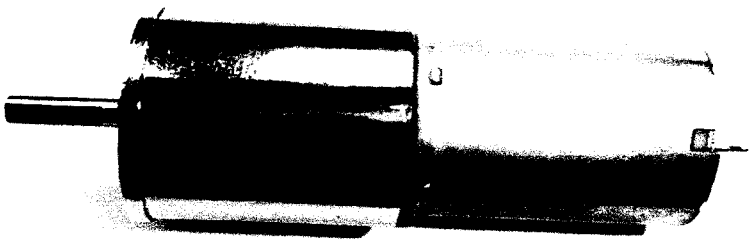
dan magenta. Warna kuning dari pencampuran hijau dan merah, warna sian adalah percampuran warna biru dan hijau, sedangkan warna magenta adalah percampuran warna biru dan merah.

Dari prinsip pemantulan warna oleh benda tidak tampak tersebut maka dapat dipakai sebagai acuan pembuatan sensor warna yang memanfaatkan pemantulan terhadap warna benda yang akan dideteksi. Misalnya benda yang akan dideteksi adalah warna merah, maka digunakan penyinaran dengan warna merah sehingga pemantulan akan sempurna berwarna merah karena benda memantulkan warna merah. Hal ini akan mempermudah sensor *photodioda* dalam pembacaan warna dari intensitas cahaya yang dipantulkan oleh objek.

Gambar 2.9. Bentuk Sensor Warna

2.6 Motor DC

Motor dc merupakan piranti *actuator* atau penggerak yang umum digunakan pada mobile robot, pemilihan motor menggunakan motor dc karena motor dc memiliki kecepatan yang cukup kencang. Pemgaturan kecepatan motor dc dilakukan menggunakan PWM *(puls widt modulation)*



Gambar 2.10. Motor DC Gearbox

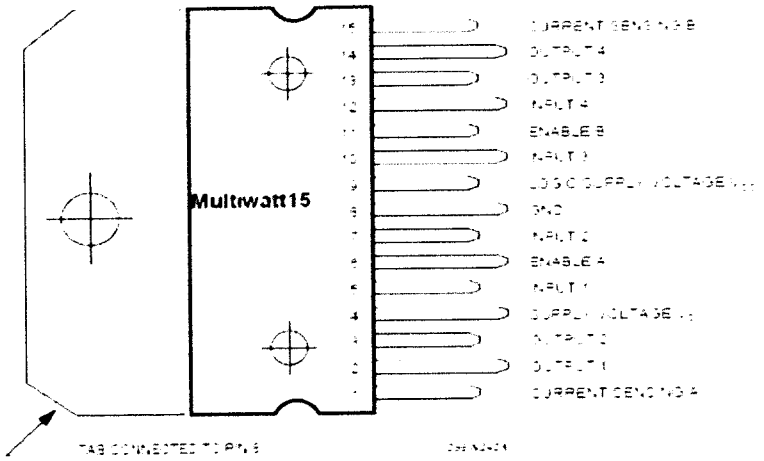
Motor yang digunakan sudah dilengkapi gear turunan untuk menghasilkan torsi atau tenaga pada putaran motor. Manfaat dari gear selain menghasilkan torsi atau tenaga gear tersebut dimanfaatkan untuk pengereman yang cukup bagus jika menggunakan kontrol secara elektronik.

2.7 Penggerak Motor DC IC L298

Sebuah IC L298 berisi empat buah *amplifier*. Setiap dua buah *amplifier* dapat digunakan sebagai sebuah untai *H-bridge* dan dapat diaktifkan dengan sebuah sinyal *enable* yang nantinya terhubung oleh pin PWM sebagai pengaturan kecepatan

putar motor dc. Dalam skripsi ini IC L298 digunakan sebagai penggerak sepasang motor dc pada robot.

IC L298 mampu beroperasi pada tegangan 4,5 V sampai 46 V. Besarnya arus yang dapat dibebankan adalah 2000 mA. Bentuk fisik L298 serta keterangan nama masing-masing pin dapat dilihat pada Gambar .....



Gambar 2.11. Bentuk IC Driver Motor DC

2.8 Motor Servo

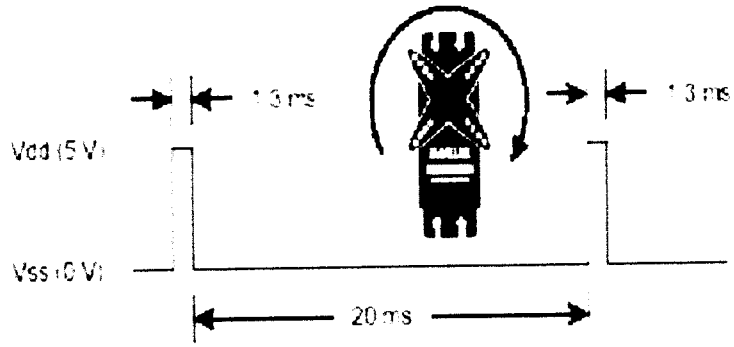
Pada dasarnya motor servo menerima masukan berupa pulsa elektronik. Putaran motor dipengaruhi oleh sinyal pulsa yang diterima. Kecepatan motor tergantung dari lebar pulsa yang diberikan pada motor servo.

Pembagian motor servo menjadi dua jenis yaitu motor servo standart yang memiliki putaran 180 ° dan motor servo continuous yang memiliki putaran 360 °. dalam penggunaan disesuaikan dengan kebutuhan. Motor servo hanya memiliki 3 buah

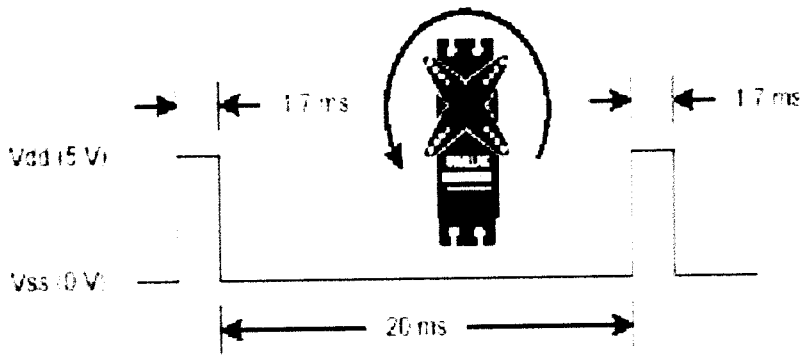
komunikasi yaitu merah (+6 VDC), hitam (- GND) dan putih untuk sinyal pengendali motor servo. Motor servo tidak lagi membutuhkan driver karena dalam box motor servo sudah dilengkapi driver khusus yang dikendalikan oleh sinyal pulsa.

Gambar 2.12. Motor Servo Standard

Motor servo yang digunakan dalam aplikasi robot ini menggunakan jenis standart yaitu memiliki putaran 180 ° yang digunakan pada bagian griper jepit dan griper angkat pada lengan robot. Untuk menggerakan motor servo dengan memberikan nilai lebar pulsa sebesar 1,3 ms maka motor akan bergerak searah jarum jam, dan jika akan menggerakan arah sebaliknya yaitu dengan memberikan lebar pulsa 1,7 ms, hal ini berlaku untuk motor standart dan motor servo continuous.



Gambar 2.13. Nilai pulsa CW



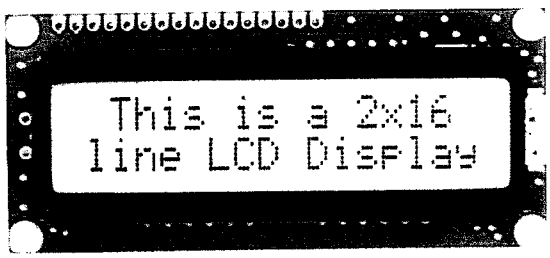
Gambar 2.14. Nilai pulsa CCW

Pemilihan motor servo yang digunakan pada lengan griper untuk mempermudah dalam menentukan sudut putaran dan ke presisian sudut putaran yang mudah kita tentukan dengan mengatur pulsa yang diinputkan pada motor servo, keuntungan lainnya adalah saat mengripper atau mengangkat servo dapat tetap memberikan gerakan pada posisi tetap sesuai yang kita harapkan.

2.9 LCD M162

LCD *(Liquit Cristal Display)* buatan TOPWAY Instrument Inc. Terdiri dari dua bagian, yang pertama merupakan panel lcd sebagai media penampil informasi dalam bentuk huruf/angka dua baris, masing–masing baris bisa menampung 16 huruf/angka.

Bagian kedua merupakan sebuah sistem yang dibentuk dengan mikrokontroler yang ditempelkan dibalik panel lcd, berfungsi untuk mengatur tampilan informasi serta berfungsi mengatur komunikasi M1632 dengan mikrokontroler yang memakai tampilan lcd tersebut. Dengan demikian pemakaian M1632 menjadi sederhana, sistem lain yang memakai M1632 cukup mengirimkan kode-kode ASCII dari informasi yang ditampilkan seperti layaknya memakai sebuah printer.



Gambar 2.15. Bentuk LCD 2x16

BAB III

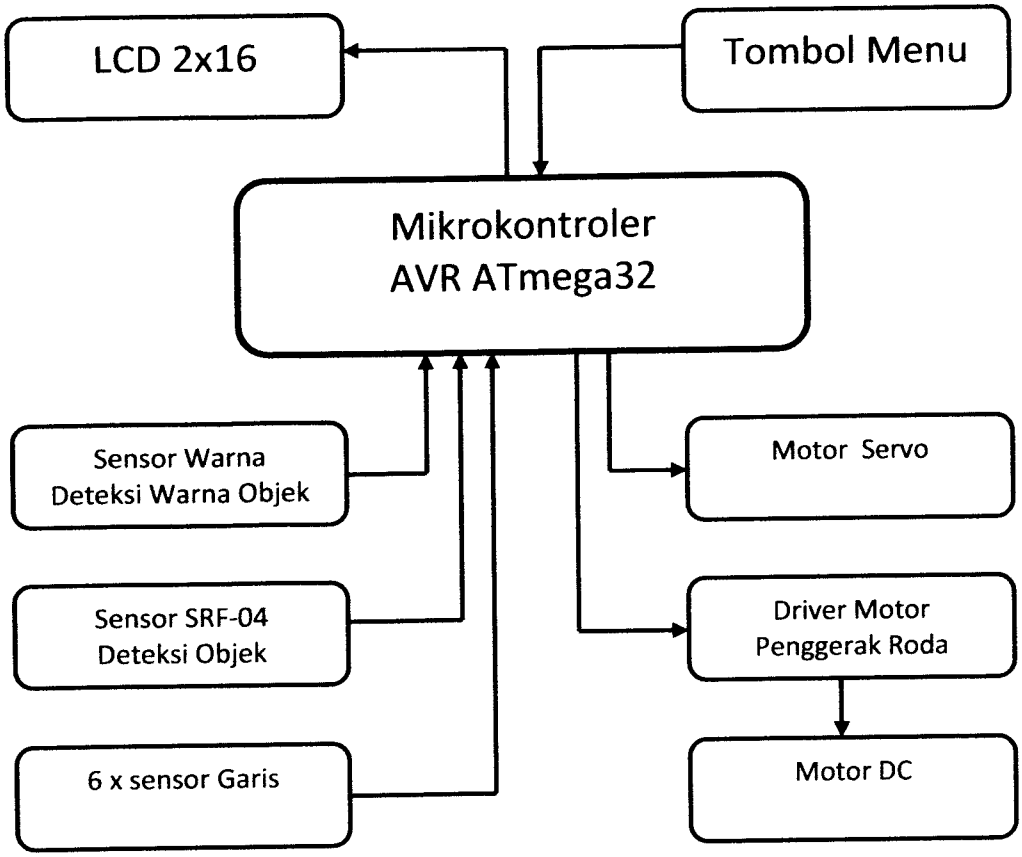
PERANCANGAN SISTEM

Pada bab ini akan dibahas tentang perancangan seluruh sistem robot yang dibuat baik *Hardware* maupun *Software*. Perancangan Hardware meliputi perancangan dan pembuatan mekanik robot. Perancangan *software* meliputi perancangan *flowchart* yang mendukung jalannya robot.

3.1. Blok Diagram

Perancangan secara umum dalam pembuatan sebuah robot pengangkut barang berdasarkan warna, yang terdiri dari sebuah sistem minimum AVR ATmega32 sebagai kendali utama dari berbagai masukan dari 6 pasang sensor garis, 1 buah sensor warna, 1 buah sensor SRF-04, 2 buah motor DC untuk penggerak roda, 2 motor servo untuk penggerakan gripper, serta menambahkan LCD 2x16 karakter sebagai penampil keadaan sistem dalam mengeksekusi program dalam pengolahan keadaan robot dalam program, sehingga kesalahan dan memperkecil nilai eror dalam pembacaan navigasi robot.

Berikut diagram blok yang digunakan dalam robot pengangkut barang berdasarkan warna :



Gambar 3.1. Blok Diagram Sistem

Dari diagram blok dapat kita perhatikan pada setiap komponen memiliki fungsi masing-masing, dan dikontrol pada mikrokontroler sebagai pengendali semua input dan output komponen blok diagram. Fungsi secara umum dari bagian-bagian blok diagram adalah sebagai berikut :

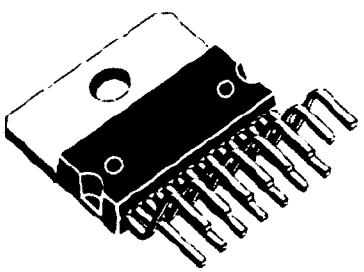
a. Mikrokontroler ATmega32 sebagai otak pengendali dari semua blok diagram yang terhubung, sinyal informasi yang masuk ke mikrokontroler diolah untuk mendapatkan intruksi output yang diinginkan.

b. Sensor jarak SRF-04 berfungsi untuk mengukur jarak antara robot dengan objek yang akan diangkat dan pindahkan. Dengan adanya sensor jarak maka robot dapat mendeteksi keberadaan objek yang akan dijepit.

c. Sensor warna menggunakan *photodioda* yang dikombinasikan dengan LED *super bright* merah, hijau dan biru atau yang umum disebut RGB. Pemilihan sensor menggunakan *photodioda* memiliki respon yang cukup cepat jika dibandingkan dengan LDR *(light Dependen Resistor).* Semakin cepat pembacaan dengan memberikan intruksi berulang -ulang akan menghasilkan pembacaan yang akurat.

d. Sensor garis menggunakan *photodioda* yang dipasangkan dengan *InfraRed* sebagai sumber cahaya untuk dipantulkan pada arena garis, sehingga robot dapat berjalan dengan sebuah informasi pantulan cahaya *InfraRed.*

e. Driver motor roda sebagai pengendali atau penguat untuk mengontrol motor dc sebagai penggerak roda robot. Driver ini menggunakan jenis driver L298D yang memiliki 2 buah output motor dan 4 buah input pengendali arah dan 2 buah pengendali pwm :



Tabel 3.2. Bentuk fisik IC L298

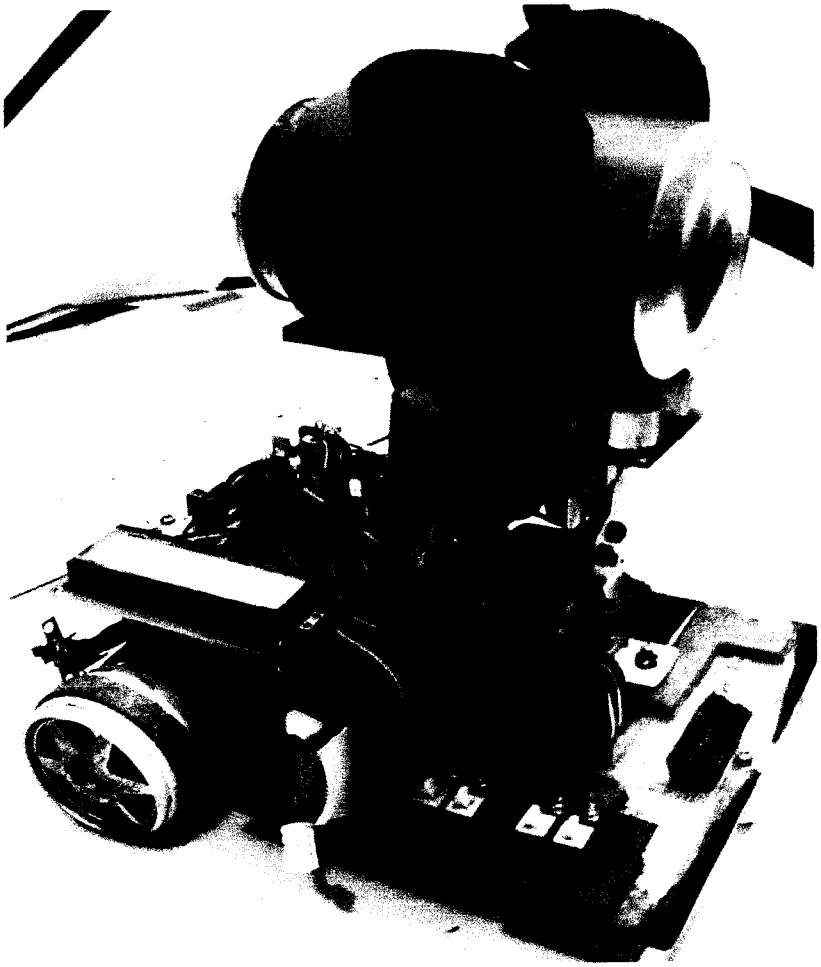
Tabel. 3.1. Pin kontrol IC L298

Pin	Input	Output	Input	Output
Input A	1	Motor maju	0	Motor Mundur
Input A	0		1	
Enable A	PWM Motor A	Nilai PWM mengatur kecepatan putaran		
Enable B	PWM Motor B			
Input B	1	Motor maju	0	Motor mundur
Input B	0		1	

f. Motor servo mengunakan 2 buah yang diantaranya untuk servo griper dan servo angkat lengan robot. Untuk servo tidak membutuhkan driver pengendali, karena didalam box motor sevo sudah dilengkapi komponen dan gear khusus buatan pabrik. Komunikasi yang sangat mudah dan dengan daya yang cukup rendah motor servo sangat cocok digunakan untuk lengan robot dengan memanfaatkan sudut putaran.

3.2. Perancangan Hardware Robot

Robot pengangkut barang berdsarkan warna didesain sesederhana mungkin agar robot tidak terlihat rumit dalam melakukan kerja. Perancangan robot ini didukung oleh rangkaian-rangkaian elektris yang membantu kinerja mikrokontroler untuk pengendali utama untuk semua sistem. Elcktronik yang dirancang menyesuaikan dengan mekanik robot yang telah dibuat sehingga tidak menghabiskan tempat dalam pemasangannya, hal ini dimaksudkan agar robot terlihat sederhana dan rapi dalam perancangannya.



Gambar 3.3. Mekanik Robot

3.2.1. Perancangan Mekanik Robot

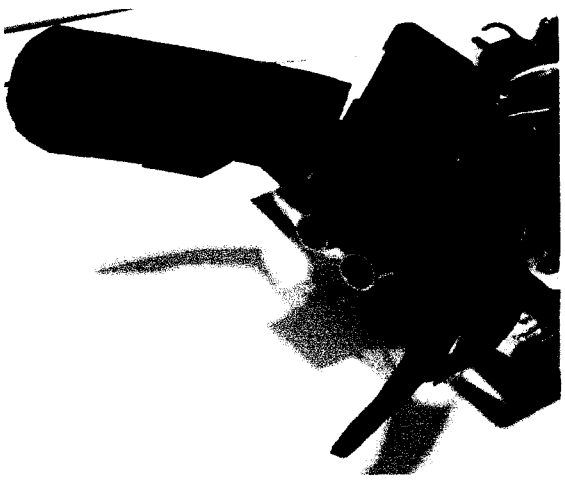
Perancangan mekanik meliputi beberapa bagian yaitu perancangan dimensi robot, perancangan mekanik roda, perancangan lengan griper.

3.2.1.1. Mekanik Roda Robot

Robot memiliki 2 buah roda yang masing-masing roda menggunakan motor dc gearbox, pada bagian depan menggunakan roda ruble atau roda bebas, agar gerakan lebih bebas kesegala arah dan tidak membebani motor penggerak. Roda menggunakan lingkaran roda RC dengan mengganti karet lapisan luar, hal ini agar robot tidak terjadi selip saat melakukan pengereman.

3.2.1.2. Perancangan Lengan Griper

Lengan griper menggunakan 2 buah motor servo standard dengan putaran 180 derajad, 1 buah untuk bagian griper jepit dan 1 buah lagi untuk mengangkat lengan griper. Saat barang yang akan dipindahkan dijepit bagian lengan griper, selanjutnya agar pemindahan barang tidak mengenai lantai maka barang harus diangkat menggunakan servo angkat pada bagian lengan griper.



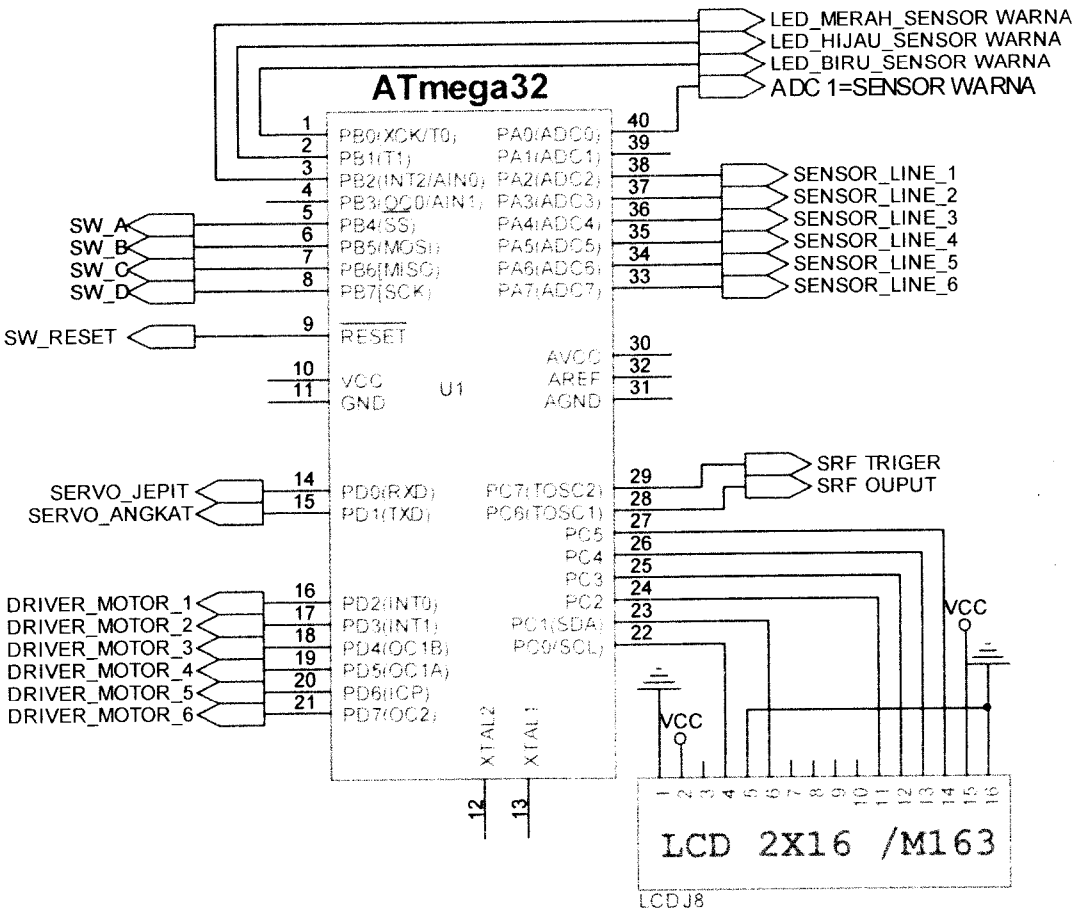
Gambar 3.4. Lengan Griper

3.2.2. Perancangan Elektronik

Perangcangan elektronik meliputi pembuatan sistem minimum AVR ATmega32, modul sensor warna, Sensor jarak SRF04, dan power supply untuk semua catu daya elektronik yang digunakan.

3.2.2.1. Sistem Minimum ATmega32

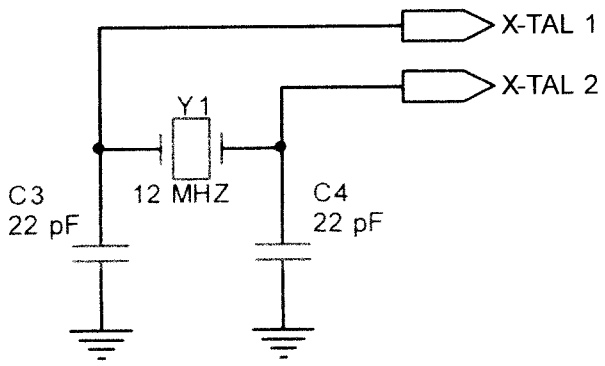
Pengendalian pada robot yaitu pada bagian sistem minimum ATmega32, yang terdiri dari rangkaian osilator, power On/Of, reset, dan beberapa port I/O.



Gambar 3.5. Skematik Sistem minimum AVR

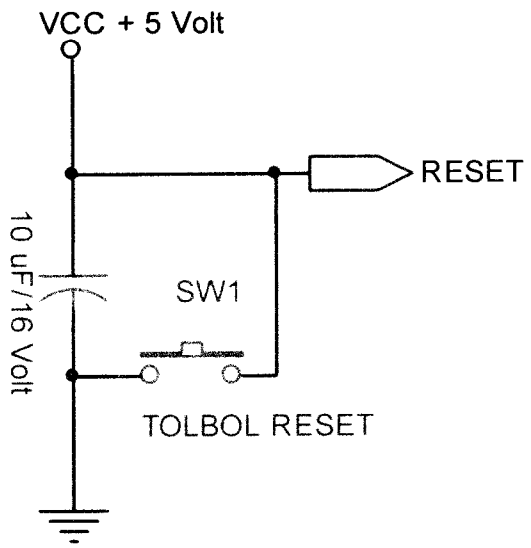
Keterangan gambar :

Sensor garis dihubungakan pada portA.2 s/d portA.7 yang masing-masing sensor dibaca menggunakan ADC internal mikrokontroler, sensor warna dihubungakan pada port A.0 dibaca menggunakan ADC chanel1. 3 buah led untuk sensor warna dihubungkan pada portB.0 – portB.2, 4 buah tombol menu untuk navigasi setingan program dihubungkan pada portB.4 – portB.7 dengan input active low (0). Motor servo dihubungkan pada portD.0 dan portD.1 untuk kedua servo mendapatkan masing-masing pin mikrokontroler untuk mengendalikan arah putaran motor servo. 6 buah pin untuk mengontrol driver motor DC dengan output PWM dan kontrol arah putaran. LCD dihubungkan pada Port C.0 =RS, Port C.2 = Enable, Port C.5 = D4 lcd, port C.5 =D5 lcd, port C.6= D6 lcd, port C.7 = D7 lcd. Sistem minimum membutuhkan osilator external sebagai pembangkit clock external yang berfungsi pembangkit dalam mengeksekusi intruksi program yang dituliskan dimemori program. Osilator eksternal menggunakan 11.059200 Hz dengan kapasitor masing 22 pF



Gambar 3.6. Clock Extal dan Kapasitor

Rangkaian *Reset* berfungsi untuk menjaga agar pada pin RST pada mikrokontroler selalu berlogika rendah saat mikro mengeksekusi program. Mikro direset pada transisi dari tegangan *low* ke tegangan *high.* Pada saat pin *reset* bernilai 1 (satu) saat pengisian kapasitor dan bernilai 0 (nol) saat kapasitor penuh. Pada saat sumber diaktifkan kapasitor terhubung singkat sehingga arus mengalir dari VCC ke pin reset sehingga pin reset bernilai 1 (satu) atau *high*, kemudian kapasitor mengisi ulang sampai nilai tegangan sama dengan VCC. Dengan demikian pin *reset* akan berlogika 0 (nol).



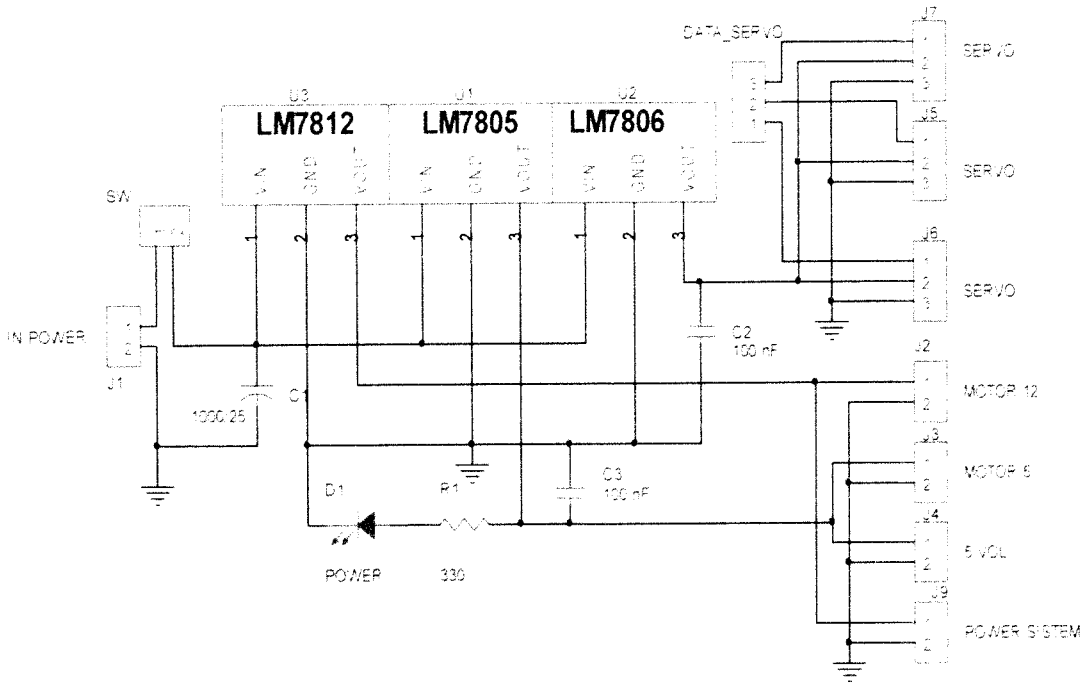
Gambar 3.7. Tombol Reset

Pada tugas akhir ini, mikrokontroler digunakan sebagai pengolah data utama yang memiliki fungsi pengolah data input dan output. Pin-pin mikrokontroler

memiliki fungsi khusus yang mendukung kinerja robot dalam menyelesaikan tugasnya.

3.2.2.2. Perancangan Power Supply

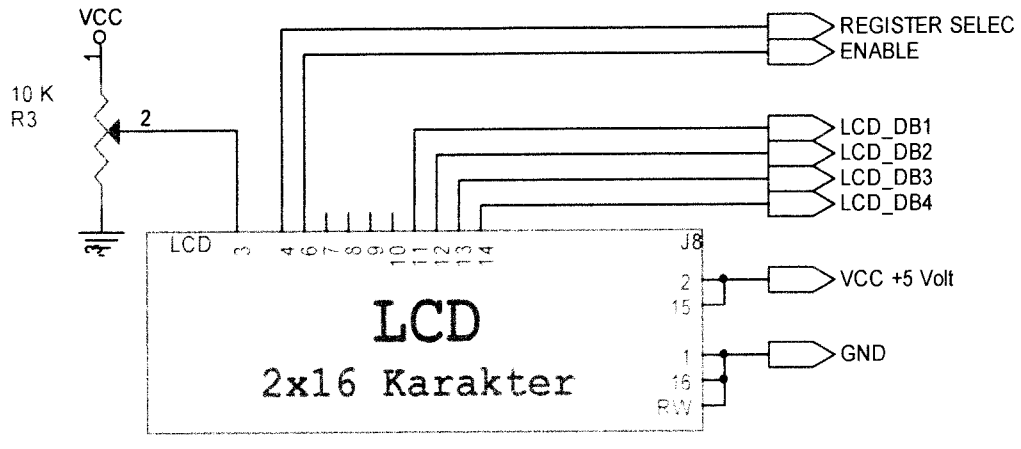
Robot yang dirancang mendapatkan power dari baterai, baterai sumber catu daya utama untuk mensupply semua bagian elektronik. Baterai yang digunakan adalah jenis lippo dengan besar tegangan 11.1 volt 1000 mA, Tegngan sebesar 11.1 volt harus dibagi ke bagian-bagian rangkaian lainnya. Power supply yang dirancang menggunakan IC regulator LM 7805 untuk menghasilkan tegangan 5 Volt untuk menyupply sensor dan sistem minimum, LM 7806 yang menghasilkan 6 Volt digunakan untuk menyupply tegangan motor servo.



Gambar 3.8. Rangkaian Power Supply

3.2.2.3. Rangkaian Penampil LCD

LCD adalah piranti elektronik untuk menampilkan hurup, angka dan simbol, lcd menggunakan 2x16 karakter atau memiliki 2 baris dan 16 kolom untuk tiap baris nya.



Gambar 3.9. LCD 2 x 16 Karakter

Lcd dikomunikasikan 4 buah jalur data dan 2 buah jalur kontrol pemilihan register dan enable. Pin control dan pin data dihubungkan di potC.

3.2.2.4. Rangkaian Sensor Garis

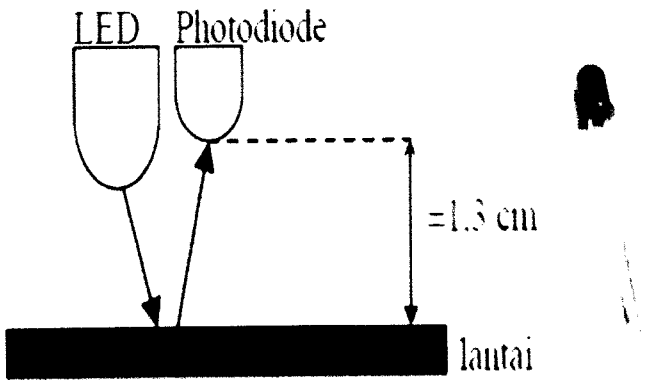
Sensor untuk penjejak garis digunakan 6 buah *photodioda* yang dipasangkan dengan *InfraRed.* Pada dasarnya *photodioda* mengalamai perubahan hambatan dikedua kutubnya bila terjadi perubahan intensitas cahaya yang

mengenainya. Perbandingan *photodioda*, *LDR* , dan *phototransistor* ditunjukkan pada tabel :

Tabel. 3.2. Perbandingan Sensor

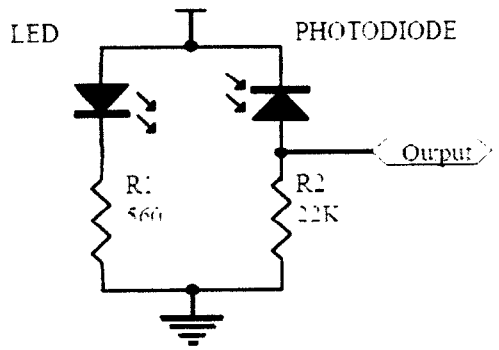
NO	Materi Perbandingan	LDR	Phototransistor	Prhotodioda
1	Perubahan Hambatan	Sedang	**Besar**	**Besar**
2	Perubahan Kenaikan	**Linier**	Tidak Linier	**Linier**
3	Penyerapan Cahaya	Tidak Fokus	**Fokus**	**Fokus**
4	Peka terhadap Ir	Tidak	**ya**	**ya**

Photodioda berfungsi sebagai pemandu jalannya robot dengan cara membaca perubahan intensitas cahaya yang mengenai lantai arena robot. Untuk dapat membedakan garis dan arena dasar maka dibutuhkan cahaya yang cukup yang bersumber dari *InfraRed* yang nantinya akan dipancarkan pada lantai arena dan hasil pantulan akan memberikan cahaya sensor *photodioda. Photodioda* dipasang sejajar dan berpasangan dengan *InfraRed* dan ke duanya tegak lurus terhadap lantai arena. Untuk lebih jelasnya dapat dilihat pada Gambar :



Gambar 3.10. Posisi sensor terhadap lantai

Pada prinsipnya perubahan hambatan pada *photodioda* akan dikombinasikan dengan rangkaian pembagi tegangan dan akan memberikan nilai tegangan yang bervariasi. Pada rangkaian pembagian tegangan tersebut, *photodioda* akan disusun seri dengan resistor referensi. Nilai hambatan tertinggi dari *photodioda* harus lebih besar dari nilai resistor referensi, dan nilai hambatan terendah dari *photodioda* harus lebih kecil dari resistor referensi agar dapat dicapai perubahan tegangan yang signifikan dan bagus. Perubahan intensitas cahaya pada arena juga sangat terpengaruh oleh cahaya lingkungan atau cahaya sekitar. Agar terhindar dari cahaya interferensi dari luar yang dapat mengganggu maka rangkaian sensor perlu diberi pelindung untuk mengurangi gangguan tersebut.

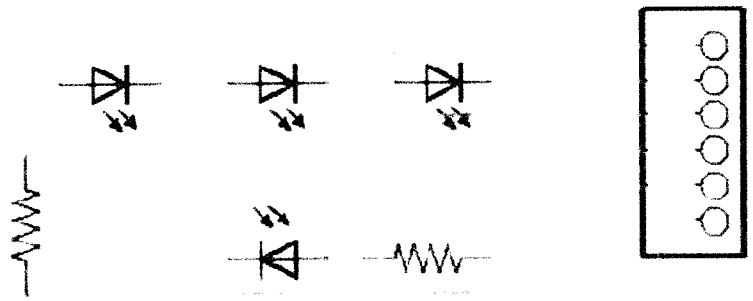


Gambar 3.11. Rangkaian sensor garis

Dari perubahan nilai tegangan pada rangkaian pembagi tegangan pada sensor maka diperoleh perubahan yang cukup untuk dibaca ADC internal pada mikrokontroler dengan resolusi ADC 10-bit sehingga nilai tegangan output sensor yang berada pada tegangan antara 0 volt hingga 5 volt akan dibaca dan dikonversikan bilangan desimal ADC antara 0 hingga 1023.

3.2.2.5. Rangkaian Sensor Warna

Pada dasarnya sensor warna ini hampir sama prinsip kerjanya dengan sensor penjejak garis. Untuk rangkaian penerima digunakan *photodiode* untuk menangkap intensitas cahaya yang berbeda-beda dan proses pembagian tegangannya sama dengan proses sensor penjejak garis. Sensor warna terdiri dari 1 buah *photodiode* dan 3 buah LED *superbright* yang terdiri dari warna primer RGB (*Red, Green, Blue*). Ketiga LED warna tersebut akan menyala bergantian dan diatur oleh pin mikrokontroler dengan proses sistem bergantian(*scanning*). Keluaran dari rangkaian pembagi tegangan sensor akan dibaca oleh pin ADC internal mikrokontroler. Setiap warna LED yang dipancarkan akan memantulkan intensitas cahaya yang berbeda-beda pada setiap warna objek. Pada skripsi ini objek hanya akan diberi tiga warna objek yaitu warna merah, hijau, dan biru.

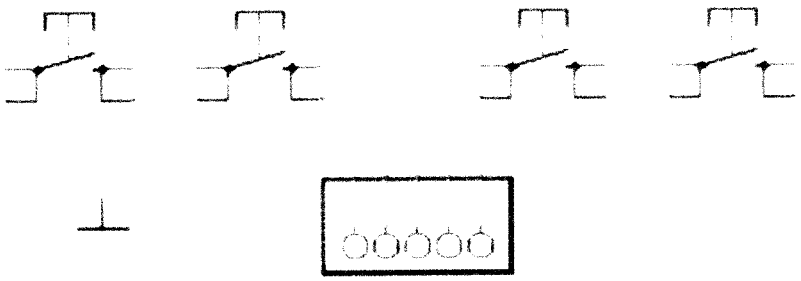


Gambar 3.12. Rangkaian sensor warna

3.2.2.6. Tombol Menu

Tombol yang digunakan dalam robot ini yaitu tombol *push button.* Ada 4 buah tombol yang terpasang dan terhubung pada PORTB. Tombol ini digunakan sebagai pemilih menu pada layar penampil LCD 2x16. Selain digunakan sebagai tombol mulai (*start*) pada robot, tombol ini juga digunakan untuk mengatur nilai konstanta-konstanta kecepatan motor nantinya.

Empat tombol ini terhubung dengan mikrokontroler dengan mode aktif low (0). Rangkaian 4 buah tombol dapat dilihat pada Gambar :



Gambar 3.13. Rangkaian Tombol Menu

3.2.2.7. Baterai

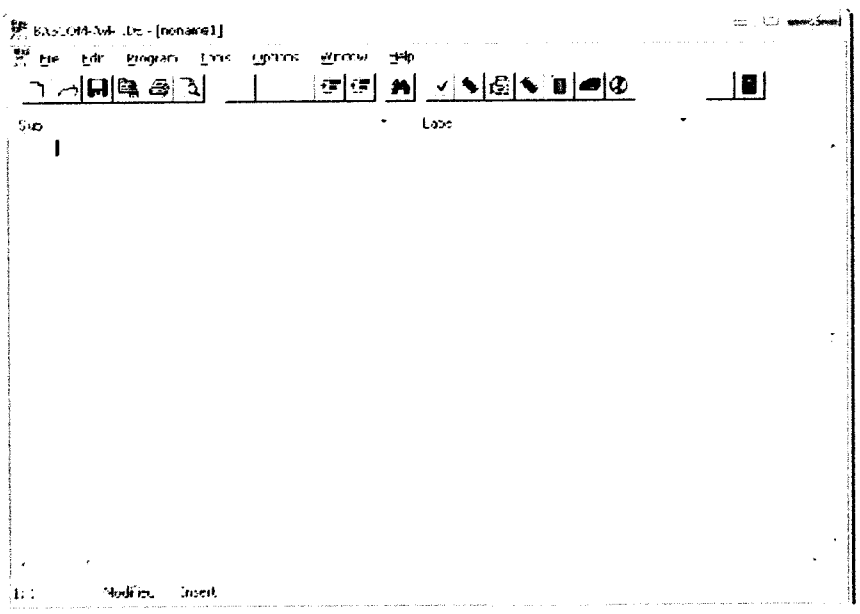
Baterai merupakan bagian yang sangat penting karena digunakan sebagai sumber energi bagi robot ini. Baterai yang digunakan adalah baterai jenis Litium Polymer (Li-Po) yang dapat diisi ulang (*charging*). Baterai jenis ini memiliki kemampuan penyimpanan energi yang cukup lama dengan arus yang cukup besar

3.2.2.8. Perancangan Perangkat Lunak (*Software)*

Perancangan perangkat lunak berisi tentang rancangan program mikrokontroler dan diagram alir (*flowchart*) tiap proses kerja robot untuk setiap bagiannya meliputi berbagai proses, antara lain:

- Pembacaan sensor garis dan kalibrasi
- Pengenalan objek dan proses pengangkutan
- Proses kontrol PWM motor
- Proses pemilihan jalan robot (*mapping*).

Pemrograman mikrokontroler dibuat menggunakan program *Basic Compiller* (*BASCOM*) AVR versi 1.11.9.5. Hasil pemrograman dari *BASCOM* AVR di-*compile* menjadi file berekstensi *.hex, yang akan di-*download* ke mikrokontroler ATMega32 dengan software PonyProg2000 atau sejenisnya.

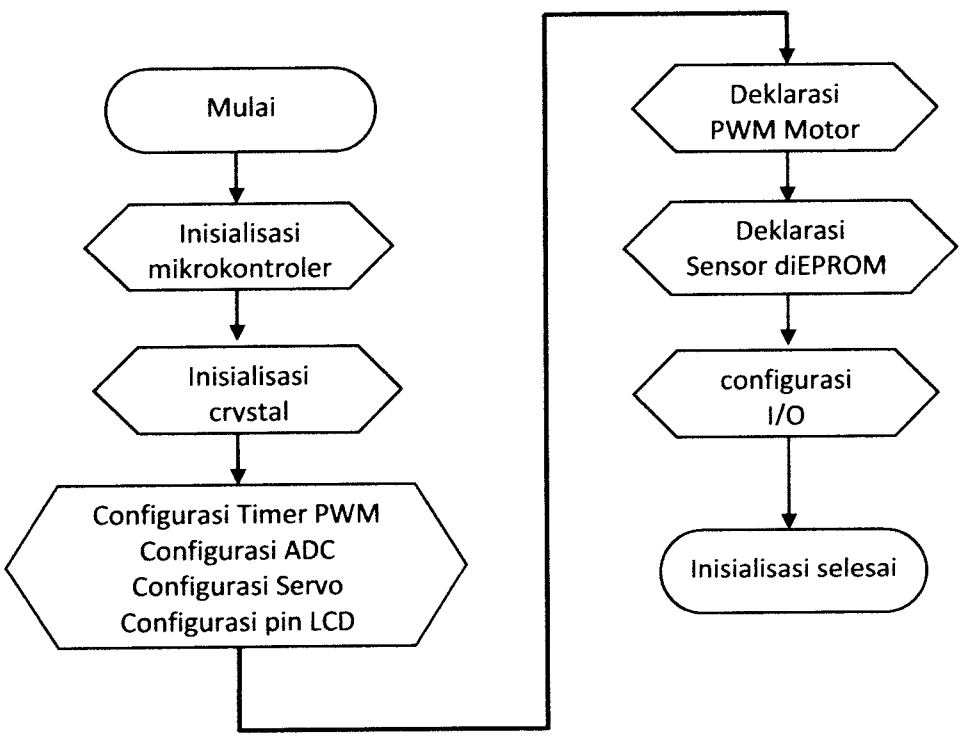


Gambar 3.15. Lembar Kerja pembuatan program

Dalam memulai penulisan program ada beberapa hal yang harus definisikan atau dideklarasikan. Yang perlu didefinisikan antara lain:

- Jenis mikrokontroler yang digunakan
- Detak *clock crystal* pada mikrokontroler
- Konfigurasi fasilitas mikrokontroler seperti *timer*, adc, servo, dan konfigurasi LCD.
- Deklarasi *variable* yang digunakan.
- Pengaturan PORT sebagai masukan, keluaran, atau fungsi khusus.
- Penginisialisasi nama dan konstanta.

Rancangan program awal robot pengakut barang yang terdiri dari deklarasi *variable* dan konfigurasi mikrokontroler.



Gambar 3.16. Diagram alir proses inisialisasi mikrokontroler

Listing program kofigurasi jenis mikrokontroler dan *clock crystal* yang dipakai adalah sebagai berikut:

```
$regfile = "m32def.dat"
$crystal = 11,059200
```

Konfigurasi fasilitas khusus pada mikrokontroler dan sintak pembantu dari *BASCOM* AVR dapat dituliskan pada program berikut ini:

```
Config Timer1= Pwm , Pwm = 8 , Compare A Pwm = Clear Down
Config Compare B Pwm = Clear Down , Prescale = 64
Config Servos=2, Servo1=Portd.0,Servo2=Portd.1,Reload = 10
Config Adc= Single,Prescaler=Auto
Config Lcdpin=Pin,Db4=Portc.3,Db5=Portc.2,Db6=Portc.4
Config Lcdpin=Pin,Db7=Portc.5,E=Portc.1,Rs=Portc.0
Config Lcd = 16 * 2
Deflcdchar 0 , 4 , 14 , 31 , 31 , 14 , 4 , 32 , 32
Cls
Cursor Off
```

Digunakan Timer1 sebagai pembangkit sinyal PWM dengan resolusi 8 bit, pembagian *prescale* = 64 untuk mendapatkan frequensi sebesar ± 100 Hz. Konfigurasi untuk pin motor servo dan jumlah servo yaitu 2 buah. ADC yang digunakan adalah *mode single* dengan *prescaler auto* untuk mendapatkan waktu konversi ADC tercepat. Konfigurasi LCD untuk setiap pin dan jenis LCD 2 x 16.

Untuk deklarasi *variable* digunkan perintah Dim diikuti tipe datanya. Deklarasi untuk beberapa *variable* proses kalibrasi sensor dituliskan pada program berikut ini:

```
Dim X As Byte , Kecepatan As Byte
Dim Ep_kp As Eram Byte , Ep_kd As Eram Byte , Ep_ki As Eram
Byte
Dim Kp As Byte , Kd As Byte , Ki As Byte
Dim Eror_lalu As Integer , Eror As Integer ,
```

```
Dim Motkan As Integer , Motkir As Integer
Dim P As Integer,D As Integer,I As Integer, Z_i As Integer
Dim Temp_kan As Word , Temp_kir As Word
Dim Sp_kan As Integer , Sp_kir As Integer
Dim Delta_eror As Integer , Pid As Integer

Dim Dat_warna As Word ,Data_srf As Word
Dim Warna_merah As Word,Warna_biru As Word,Warna_hijau As Word
Dim Perjalan As Byte
Dim Set_simpang As Byte , Ep_simpang As Eram Byte
Dim Lines As Byte,X_max_e As Eram Byte,
Dim X_max As Byte,Titip As Byte
Dim Ep_ln1 As Eram Word , Ep_ln2 As Eram Word ,
Dim Ep_ln3 As Eram Word ,
Dim Ep_ln4 As Eram Word , Ep_ln5 As Eram Word ,
Dim Ep_ln6 As Eram Word ,
Dim Ep_ln7 As Eram Word , Ep_ln8 As Eram Word
Dim Ref_ln1 As Word , Ref_ln2 As Word , Ref_ln3 As Word ,
Dim Ref_ln4 As Word ,
Dim Ref_ln5 As Word , Ref_ln6 As Word , Ref_ln7 As Word ,
Dim Ref_ln8 As Word
Dim Dat_ln1 As Word , Dat_ln2 As Word ,
Dim Dat_ln3 As Word , Dat_ln4 As Word , Dat_ln5 As Word ,
Dim Dat_ln6 As Word ,
Dim Dat_ln7 As Word , Dat_ln8 As Word
Dim Hi_ln1 As Word , Hi_ln2 As Word , Hi_ln3 As Word ,
Dim Hi_ln4 As Word ,
Dim Hi_ln5 As Word , Hi_ln6 As Word , Hi_ln7 As Word ,
Dim Hi_ln8 As Word
Dim Lo_ln1 As Word , Lo_ln2 As Word , Lo_ln3 As Word ,
Dim Lo_ln4 As Word ,
Dim Lo_ln5 As Word , Lo_ln6 As Word , Lo_ln7 As Word ,
Dim Lo_ln8 As Word ,
Dim Waktu_muter As Word
```

Konfigurasi pin mikrokontroler sebagai masukan atau keluaran diatur pada register DDRX dengan nilai 0 sebagai masukan dan nilai 1 sebagai keluaran, programnya sebagai berikut:

```
Ddrc.6 = 0
Ddrc.7 = 1
Ddrb = &B11100000
Portb = &HFF
Ddra.1 = 1
```

```
Porta.1 = 1
Portd = &HFF
Portc = 255
```

3.2.2.9. Perancangan dan kalibrasi sensor

Perubahan intensitas cahaya yang mengenai *photodiode* akan ditandai dengan perubahan nilai tegangan pada keluaran sensor hasil pembagi tegangan. Nilai tegangan ini akan dibaca oleh ADC 10-bit internal mikrokontroler dan dikonversikan menjadi nilai bilangan desimal pada sebuah *variable* antara 0 hingga 1023, nilai konversi dapat dihitung dari persamaan :

$$\text{Nilai ADC} = \frac{Vin}{VReff} x1024$$

Mikrokontroler pada robot ini konfigurasi nilai referensi ADC diambil dari pin AVCC yaitu 5 volt sehingga nilai VReff adalah 5 volt. Untuk membaca dan mengambil data ADC digunakan perintah Getadc(channel). Sub program dalam pembacaan ADC adalah sebagai berikut:

```
Baca_adc:
Start Adc
Dat_ln1 = Getadc(2)
Dat_ln2 = Getadc(2)
Dat_ln3 = Getadc(3)
Dat_ln4 = Getadc(4)
Dat_ln5 = Getadc(5)
Dat_ln6 = Getadc(6)
Dat_ln7 = Getadc(7)
```

```
Dat_ln8 = Getadc(7)
Stop Adc
        Return
```

Dari pengambilan data sensor tersebut kemudian nilai ADC dibandingkan dengan nilai referensi untuk mendapat logika sensor apakah itu hitam atau putih dan data untuk setiap sensor diubah menjadi nilai per-bit. Proses konversi ini digunakan untuk mempermudah pembacaan logika sensor dan proses ini untuk menggantikan fungsi *comparator*. Jumlah sensor garis pada robot ini adalah enam buah *photodioda* dan ADC yang terpakai adalah *channel* 0 hingga 5 dan hasil konversi akan mengubah nilai tiap bit *variable* Lines. Program untuk konversinya adalah sebagai berikut:

```
Conversi:
Gosub Baca_adc
If Dat_ln1 > Ref_ln1 Then Ln0 = 1
If Dat_ln1 < Ref_ln1 Then Ln0 = 0
If Dat_ln2 > Ref_ln2 Then Ln1 = 1
If Dat_ln2 < Ref_ln2 Then Ln1 = 0
If Dat_ln3 > Ref_ln3 Then Ln2 = 1
If Dat_ln3 < Ref_ln3 Then Ln2 = 0
If Dat_ln4 > Ref_ln4 Then Ln3 = 1
If Dat_ln4 < Ref_ln4 Then Ln3 = 0
If Dat_ln5 > Ref_ln5 Then Ln4 = 1
If Dat_ln5 < Ref_ln5 Then Ln4 = 0
If Dat_ln6 > Ref_ln6 Then Ln5 = 1
If Dat_ln6 < Ref_ln6 Then Ln5 = 0
If Dat_ln7 > Ref_ln7 Then Ln6 = 1
If Dat_ln7 < Ref_ln7 Then Ln6 = 0
If Dat_ln8 > Ref_ln8 Then Ln7 = 1
If Dat_ln8 < Ref_ln8 Then Ln7 = 0
Return
```

Nilai referensi untuk tiap sensor didapat dari proses perekaman data pada saat kalibrasi. Pada saat proses kalibrasi mikrokontroler akan menyimpan nilai tertinggi ADC (lantai putih) dan nilai terendah ADC (lantai hitam). Nilai tengah

diantara nilai tertinggi dan terendah akan digunakan sebagai nilai referensi yang akan digunakan sebagai pembanding diproses konfersi. Persamaan untuk mencari nilai referensi ditunjukan pada persamaaan :

$$\text{Nilai referensi ADC} = \frac{ADC\ Tertinggi + ADC\ terendah}{2}$$

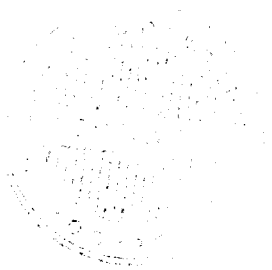
Proses pengambilan data saat proses kalibrasi bisa dilakukan secara otomatis atau manual. Dari pengambilan data tersebut kemudian didapat nilai referensi dari persamaan diatas nilai referensi tersebut kemudian akan disimpan dalam EEPROM sehingga nilai tersebut tidak akan hilang walaupun mikrokontroler dimatikan suplai energinya. Nilai EEPROM akan selalu dipindah ke RAM mikrokontroler saat robot awal dihidupkan. Proses perekaman atau kalibrasi dapat dijelaskan lebih lengkap dengan melihat program berikut ini:

```
Reff_step:
Cls
Lcd "<oke>"
Locate 2 , 1
Lcd " cari putih "
Waitms 300
Do

Gosub Baca_adc
Loop Until Sw_a = 0
Hi_ln1 = Dat_ln1
Hi_ln2 = Dat_ln2
Hi_ln3 = Dat_ln3
Hi_ln4 = Dat_ln4
Hi_ln5 = Dat_ln5
```

```
Hi_ln6 = Dat_ln6
Hi_ln7 = Dat_ln7
Hi_ln8 = Dat_ln8


Cls
Locate 1 , 12
Lcd "<oke>"
Locate 2 , 1
Lcd " cari hitam "
Waitms 300
Do

Gosub Baca_adc
Loop Until Sw_d = 0
Lo_ln1 = Dat_ln1
Lo_ln2 = Dat_ln2
Lo_ln3 = Dat_ln3
Lo_ln4 = Dat_ln4
Lo_ln5 = Dat_ln5
Lo_ln6 = Dat_ln6
Lo_ln7 = Dat_ln7
Lo_ln8 = Dat_ln8
```

Setelah melakukan perekaman data tertinggi dan terendah maka dari kedua data tersebut akan dicari nilai referensi yang nantinya akan disimpan pada *variable* EEPROM. *Listing* programnya sebagai berikut:

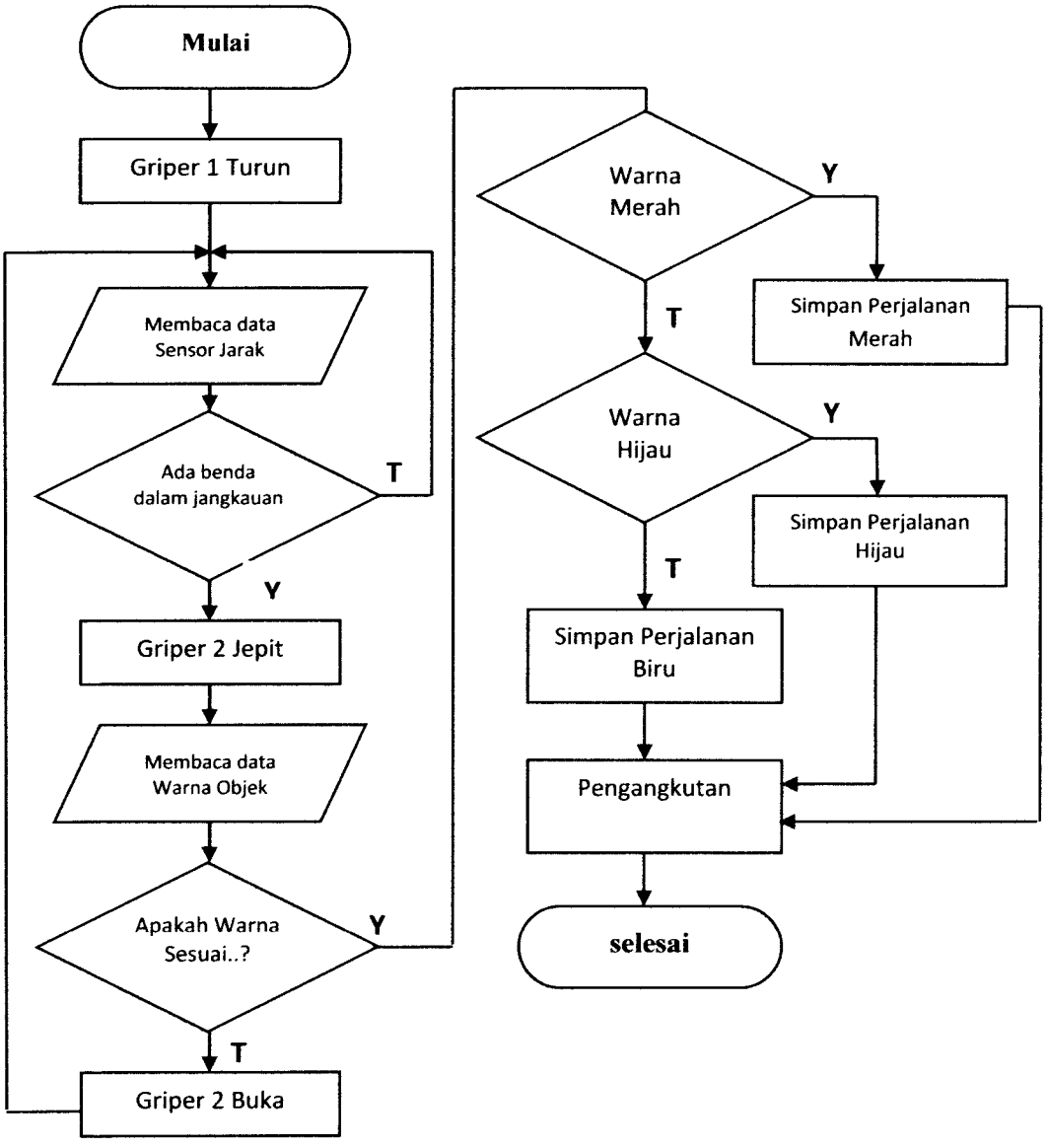
```
Ref_ln1 = Hi_ln1 + Lo_ln1
Ref_ln2 = Hi_ln2 + Lo_ln2
Ref_ln3 = Hi_ln3 + Lo_ln3
Ref_ln4 = Hi_ln4 + Lo_ln4
Ref_ln5 = Hi_ln5 + Lo_ln5
Ref_ln6 = Hi_ln6 + Lo_ln6
Ref_ln7 = Hi_ln7 + Lo_ln7
Ref_ln8 = Hi_ln8 + Lo_ln8


Ref_ln1 = Ref_ln1 / 2
Ref_ln2 = Ref_ln2 / 2
Ref_ln3 = Ref_ln3 / 2
Ref_ln4 = Ref_ln4 / 2
```



```
Ref_ln5 = Ref_ln5 / 2
Ref_ln6 = Ref_ln6 / 2
Ref_ln7 = Ref_ln7 / 2
Ref_ln8 = Ref_ln8 / 2

Ep_ln1 = Ref_ln1
Ep_ln2 = Ref_ln2
Ep_ln3 = Ref_ln3
Ep_ln4 = Ref_ln4
Ep_ln5 = Ref_ln5
Ep_ln6 = Ref_ln6
Ep_ln7 = Ref_ln7
Ep_ln8 = Ref_ln8
```

3.2.2.10. Pengenalan Objek dan proses pengangkutan

Dalam proses pengenalan objek, robot diusahakan dapat mengenali ada tidaknya keberadaan objek. Robot juga akan mencoba mengenali warna objek untuk proses pemetaan peletakan jalur target. Objek pada awal pengambilan sudah diatur letaknya dan jarak antara awal robot mulai (*home*) berjarak kurang lebih 25 cm dari titik peletakan objek. Robot akan berhenti di awal pengangkutan dan titik target jika sensor garis membaca sebuah simpang pertigaan atau semua sensor mengenai garis hitam. Untuk proses pengenalan objek dapat dijelaskan pada diagram alir



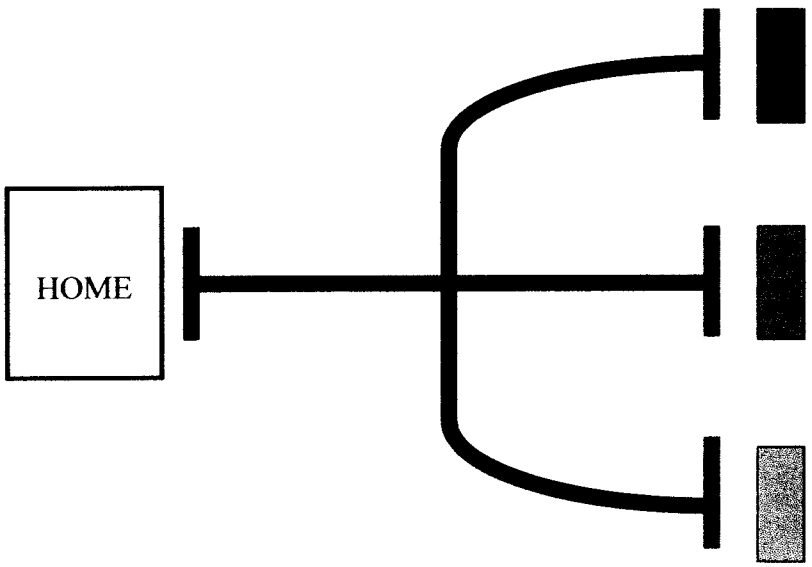
Gambar 3.17. Pengenalan Objek dan proses pengangkutan

```
Scan_objek:
Gosub Grip_turun
Gosub Jepit_buka

Baca_jarak:
   Init_srf = 1
   Waitus 10
   Init_srf = 0
   Data_srf = 0
   Do
      Waitus 1
      Incr Data_srf
      If Echo_srf = 1 Then Exit Do
      Loop Until Data_srf = 1000
     Data_srf = 0

Do
Waitus 1
Incr Data_srf
If Echo_srf = 0 Then Exit Do
If Data_srf > 3000 Then Exit Do
Loop
Data_srf = Data_srf / 10
Return

Baca_warna_jarak:
     Gosub Baca_jarak
     Start Adc
     Dat_warna = Getadc(0)
     Stop Adc
     Dat_warna = Dat_warna
     Return

Baca_warna:
  Led_merah = 0
  Led_biru = 1
  Led_hijau = 0
  Waitms 20
  Gosub Baca_warna_jarak
  Warna_biru = Dat_warna
  Waitms 10

  Led_merah = 1
  Led_biru = 0
  Led_hijau = 0
  Waitms 20
  Gosub Baca_warna_jarak
```

```
Warna_merah = Dat_warna
Waitms 10

Led_merah = 0
Led_biru = 0
Led_hijau = 1
Waitms 20
Gosub Baca_warna_jarak
Warna_hijau = Dat_warna * 10
Waitms 30
Return
```

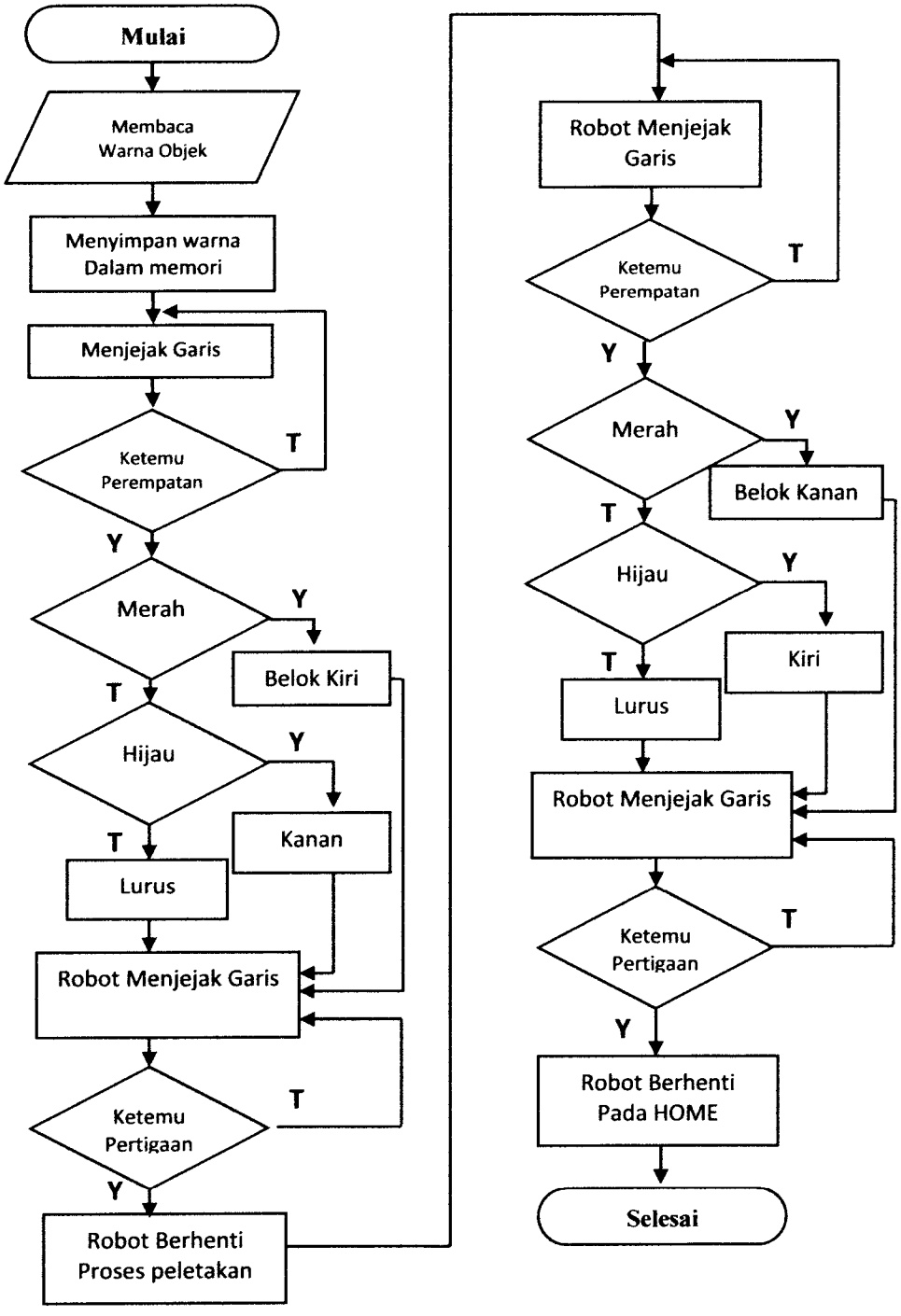
3.2.2.11. Proses Pemilihan Jalan Robot

Sebuah *mobile* robot harus memiliki panduan atau alur perjalanan dalam melaksanakan tugasnya. Dalam hal ini digunakan sistem pemetaan(*mapping*). Robot ini memiliki algoritma pemetaan dalam menentukan arah perjalanan. Kasus yang diberikan pada robot ini yaitu dapat mengangkut barang (objek) dari satu sumber (*home*) dan meletakannya pada sebuah target dengan memilih salah satu jalur perjalanan target yang bergantung dari warna pembacaan objek kemudian dapat kembali ke posisi awal (*home*). Bentuk dari contoh arena pada kasus ini ditunjukan pada gambar dibawah ini :



Gambar 3.18. Arena atau Lapangan Robot

Robot pengangkut barang ini memiliki panduan dalam hal pemilihan jalur yaitu dengan cara sensor garis mendeteksi adanya sebuah persimpangan jalan antara perempatan. Proses dari pemilihan jalur ini dapat dijelaskan dalam diagram alir sebagai berikut:



Gambar 3.19. Diagram alir proses pemetaan arena

```
Gosub Baca_warna
If Warna_merah > 500 And Warna_biru < 500 And Warna_hijau <
500 Then
Perjalan = Merah
Elseif Warna_merah < 500 And Warna_biru > 500 And Warna_hijau
< 200 Then
Perjalan = Biru
Elseif Warna_merah > 300 And Warna_biru > 500 And Warna_hijau
> 150 And Warna_hijau < 500 Then
Perjalan = Hijau

Else
Gosub Jepit_buka
Gosub Mundur_dikit
Goto Scan_objek
End If
Gosub Grip_naik
Gosub Mundur_dikit
Gosub Balik_kiri

Do
Call Ngeline
      Gosub Conversi
      If Ln2 = 0 And Ln3 = 0 And Ln4 = 0 And Ln5 = 0 Then
      Exit Do
      Loop
            If Perjalan = Merah Then
            Gosub Belok_kiri
            Elseif Perjalan = Biru Then
            For X = 1 To 30
            Call Ngeline
            Next X
            Else
            Gosub Belok_kanan
            End If

Do
Call Ngeline
Gosub Conversi
If Ln2 = 0 And Ln3 = 0 And Ln4 = 0 And Ln5 = 0 Then
Exit Do
Loop

Gosub Ngerem
Gosub Grip_turun
Gosub Jepit_buka
Gosub Maju_dikit
```

```
Gosub Mundur_dikit
Gosub Grip_naik
Gosub Jepit_tutup

If Perjalan = Merah Then
Gosub Balik_kanan
Elseif Perjalan = Biru Then
Gosub Balik_kanan
Else
Gosub Balik_kiri
End If

Do
Call Ngeline
Gosub Conversi
If Ln2 = 0 And Ln3 = 0 And Ln4 = 0 And Ln5 = 0 Then
Exit Do
Loop

If Perjalan = Merah Then
Gosub Belok_kanan
Elseif Perjalan = Biru Then
For X = 1 To 40
Call Ngeline
Next X
Else
Gosub Belok_kiri
End If
Loop
```

Dari proses awal pengangkutan objek, robot telah menyimpan data warna objek yang sedang diangkut dan disimpan dalam variable perjalanan. Setelah menemukan perempatan, robot akan memilih apakah akan tetap berjalan lurus, belok kiri, atau belok kanan. Demikian juga saat robot dalam perjalanan pulang menuju *home* awal.

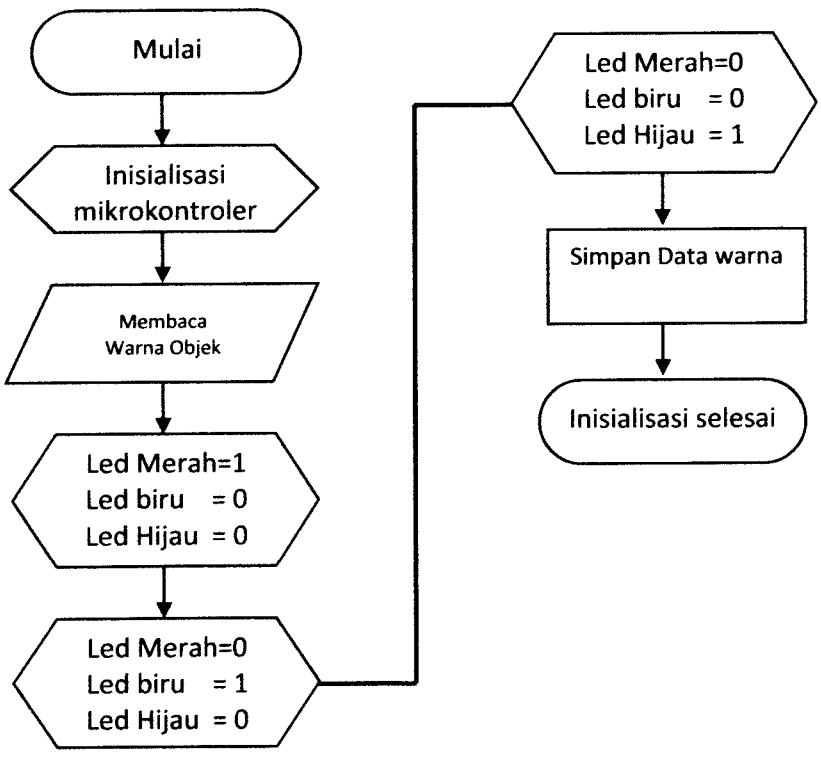
Proses penyimpanan data warna hanya sekali saat robot akan mengambil objek yang akan dipindahkan. Setelah mendapatkan data warna yang telah sesuaikan

dalam pengambilan data robot sudah dapat menentukan perjalan dalam penempatan objek yang akan dipindahkan.

3.2.2.12. Sub flowchart dan program masing-masing sensor

Sub *flowchart* dan program untuk masing-masing sensor yang di terapkan dalam aplikasi robot pemindah barang berdasarkan warna, bagian ini digunakan untuk mempermudah menganilisa bagian program yang digunakan setelah nantinya akan digabungkan dengan program keseluruhan.

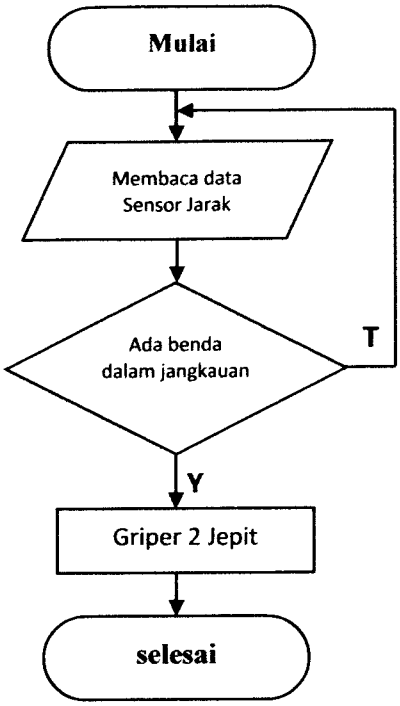
3.2.2.13.1. Sensor Warna



Gambar 3.20. Diagram alir proses pembacaan warna objek

```
Do
Gosub Baca_warna
Locate 2 , 1
Lcd  "R:"  ;  Warna_merah  ;  "  B:"  ;  Warna_biru  ;  "  G:"  ;
Warna_hijau ; "      "
Loop

Baca_warna:
      Led_merah = 1
      Led_biru = 0
      Led_hijau = 1
      Waitms 40
      Gosub Baca_warna_jarak
      Warna_biru = Dat_warna
      Waitms 10
      Led_merah = 1
      Led_biru = 1
      Led_hijau = 0
      Waitms 40
      Gosub Baca_warna_jarak
      Warna_hijau = Dat_warna
      Waitms 10
      Led_merah = 0
      Led_biru = 1
      Led_hijau = 1
      Waitms 40
      Gosub Baca_warna_jarak
      Warna_merah = Dat_warna
      Waitms 10
      Led_merah = 1
      Led_biru = 1
      Led_hijau = 1
      Waitms 40
      Return

 Baca_warna_jarak:
      Start Adc
      Dat_warna = Getadc(0)
      Stop Adc
      Dat_warna = Dat_warna
      Return
```

3.2.2.13.2. Sensor Jarak

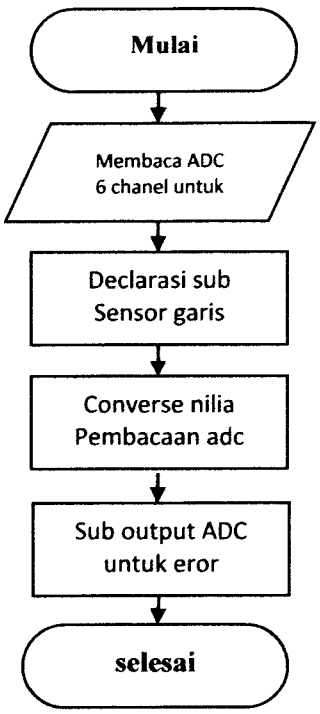


Gambar 3.21. Diagram alir proses pembacaan Sensor jarak

```
Baca_jarak:
   Init_srf = 1
   Waitus 10
   Init_srf = 0
   Data_srf = 0
   Do
      Waitus 1
      Incr Data_srf
      If Echo_srf = 1 Then Exit Do
      Loop Until Data_srf = 1000
           Data_srf = 0

           Do
           Waitus 1
           Incr Data_srf
           If Echo_srf = 0 Then Exit Do
```

```
If Data_srf > 3000 Then Exit Do
Loop
Data_srf = Data_srf / 10
Return
```

3.2.2.13.3. Pembacaan ADC Sensor Garis



Gambar 3.22. Diagram alir proses pembacaan Sensor Garis

```
Baca_adc:
  Start Adc
   Dat_ln1 = Getadc(2)
   Dat_ln2 = Getadc(2)
   Dat_ln3 = Getadc(3)
   Dat_ln4 = Getadc(4)
   Dat_ln5 = Getadc(5)
   Dat_ln6 = Getadc(6)
   Dat_ln7 = Getadc(7)
   Dat_ln8 = Getadc(7)
   Stop Adc
   Return
```

```
Declare Sub Ngeline
  Ln0 Alias Lines.0
  Ln1 Alias Lines.1
  Ln2 Alias Lines.2
  Ln3 Alias Lines.3
  Ln4 Alias Lines.4
  Ln5 Alias Lines.5
  Ln6 Alias Lines.6
  Ln7 Alias Lines.7

Conversi:
Gosub Baca_adc
If Dat_ln1 > Ref_ln1 Then Ln0 = 1
If Dat_ln1 < Ref_ln1 Then Ln0 = 0
If Dat_ln2 > Ref_ln2 Then Ln1 = 1
If Dat_ln2 < Ref_ln2 Then Ln1 = 0
If Dat_ln3 > Ref_ln3 Then Ln2 = 1
If Dat_ln3 < Ref_ln3 Then Ln2 = 0
If Dat_ln4 > Ref_ln4 Then Ln3 = 1
If Dat_ln4 < Ref_ln4 Then Ln3 = 0
If Dat_ln5 > Ref_ln5 Then Ln4 = 1
If Dat_ln5 < Ref_ln5 Then Ln4 = 0
If Dat_ln6 > Ref_ln6 Then Ln5 = 1
If Dat_ln6 < Ref_ln6 Then Ln5 = 0
If Dat_ln7 > Ref_ln7 Then Ln6 = 1
If Dat_ln7 < Ref_ln7 Then Ln6 = 0
If Dat_ln8 > Ref_ln8 Then Ln7 = 1
If Dat_ln8 < Ref_ln8 Then Ln7 = 0
Return

Sub Ngeline
Gosub Conversi
Select Case Lines
Case &B11111100 : Eror = 8
Case &B11111000 : Eror = 6
Case &B11111011 : Eror = 4
Case &B11110011 : Eror = 2
Case &B11110111 : Eror = 1
Case &B11100111 : Eror = 0
Case &B11101111 : Eror = -1
Case &B11001111 : Eror = -2
Case &B11011111 : Eror = -4
Case &B00011111 : Eror = -6
Case &B00111111 : Eror = -8
End Select
End Sub
```

BAB IV

ANALISA DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil pengujian dari sistem yang dibuat, pada tugas akhir ini pengujian yang dilakukan adalah pengujian sensor garis, pengujian modul sensor warna untuk mendeteksi warna, pengujian ultrasonik untuk jarak dan pengujian robot di arena yang telah dibuat dan ditentukan.

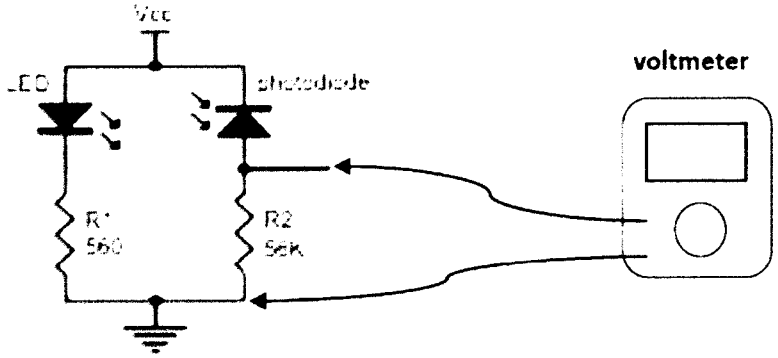
4.1. Pengujian dan Analisis Sensor Garis

Sensor untuk mendeteksi garis lantai digunakan larik sensor *photodiode* berpasangan dengan LED *superbright* warna biru. Untuk mendeteksi pergeseran atau posisi robot terhadap garis digunakan 6 buah sensor secara berjejer dengan jarak antar sensor 12 mm dan jarak sensor dengan titik tengah kedua roda adalah 130 mm. Jarak ujung sensor tegak dengan lantai adalah 10 mm. Dalam robot ini digunakan sensor *photodiode* karena karakteristiknya yang paling memenuhi untuk sensor garis dan digunakan LED *superbright* sebagai sumber cahaya karena LED ini cukup terang untuk memberi pantulan pada lantai berwarna putih dan cahayanya cukup teredam oleh lantai berwarna hitam (garis). Foto dari sensor garis ini dapat ditunjukan pada Gambar :

Gambar 4.1. Sensor Garis

Sensor ini memberikan nilai keluaran berupa tegangan antara 0 hingga 5 volt yang berubah mengikuti intensitas pantulan cahaya yang mengenainya. Dari rangkaian pembagi tegangan dapat diukur nilai keluaran tegangan untuk kondisi lantai hitam dan lantai putih yang diukur dengan voltmeter yang ditunjukan pada Gambar 4.2 dan hasil pembacaan tersebut dilihat pada LCD untuk menampilkan nilai ADC beserta penentuan nilai referensi ADC yang dicatat pada Tabel 4.1 dan dipresentasikan oleh grafik pada Gambar 4.3. Persamaan untuk nilai tegangan output dari pembagian tegangan sensor ini dapat dituliskan pada persamaan:

$$V_{out} = \frac{R2}{R2+Rs} \cdot Vcc = \frac{56K\Omega}{56K\Omega+Rs} \cdot 5\ volt$$



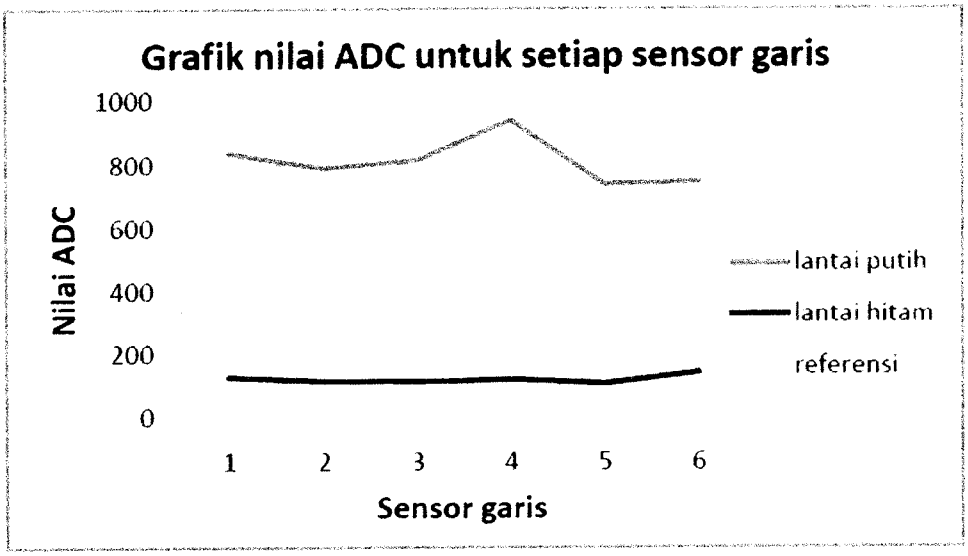
Gambar 4.2. Pengukuran keluaran sensor garis

Untuk mengetahui nilai referensi ADC atau nilai tengah dari nilai perbedaan nilai ADC digunakan persamaan yang dituliskan pada persamaa

$$\text{Nilai referensi ADC} = \frac{ADC\ tertinggi + ADC\ terendah}{2}$$

Tabel 4.1 Pengukuran tegangan keluaran ADC sensor

NO Sensor	**Lantai Putih**		**Lantai Hitam**		**Referensi ADC**
	Tegangan (V)	Nilai ADC	Tegangan (V)	Nilai ADC	
1	4,09	810	0.62	120	465
2	3,85	790	0,65	166	478
3	3,98	816	0,55	114	465
4	4,58	940	0,58	120	530
5	3,60	738	0,51	106	422
6	3,63	744	0,69	142	443

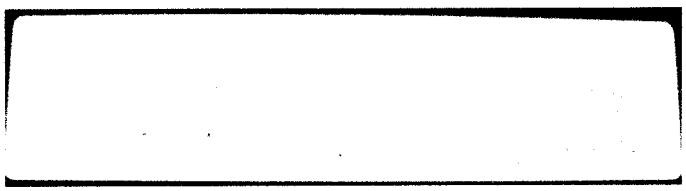


Gambar 4.3. Grafik Nilai ADC setiap sensor

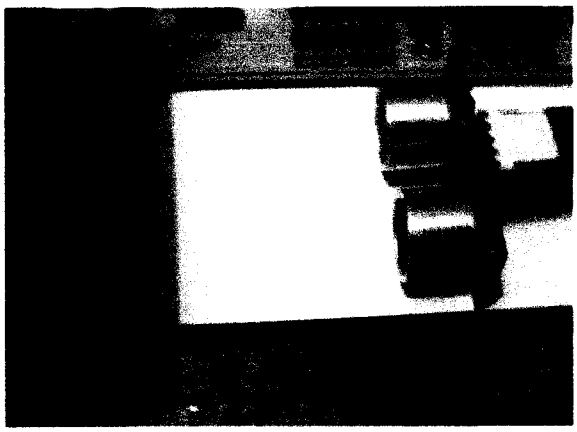
Dari Tabel 4.1 dan Gambar 4.3 terlihat adanya perubahan tegangan dan nilai ADC yang signifikan antara warna hitam dan putih. Selisih nilai ADC antara lantai warna hitam dan warna putih sangat besar sehingga pembacaan sensor telah bekerja dengan baik. Nilai ADC sensor garis sedikit berbeda bila dilakukan pembacaan pada beberapa tempat yang berbeda karena adanya sedikit pengaruh cahaya luar terutama cahaya matahari namun perbedaan ini tidak mengganggu selama proses kalibrasi ikut menyesuaikan. Proses pembacaan ADC dipilh mode *clock* tertinggi yaitu 200 kHz atau waktu konversi ADC dalam mikrokontroler antara 13-260 us dan proses setiap pembacaan *channel* ADC memerlukan waktu sekitar 0,5 ms.

4.2. Pengujian Sensor Ultrasonik SRF04

Pengujian sensor jarak dilakukan untuk mendapatkan nilai ke akuratan dalam pembacaan yang sesuai antara robot dan objek, serta untuk mengetahui jangkauan minimal dan maksimal sensor ultrasonik dengan objek, Pengujian dilakukan dalam posisi berhadapan langsung dengan objek, hal ini dilakukan untuk mendapatkan nilai pantul yang sepurna. Untuk membantu dalam menganalisa dan menentukan jarak yang diukur, maka dalam pengujian ini menggunakan tampilan modul LCD.



Gambar 4.4. Hasil Pengujian Tampilan LCD



Gambar 4.5. Pengujian SRF04 Jarak sebenarnya

Pengujian sensor SRF04 secara langsung dengan memberikan halangan langsung pada sensor SRF04 dan diukur menggunakan penggaris untuk melihat hasil pengukuran seberanya dan nilai *counter* yang ditampilkan pada LCD. LCD menampilkan code 1 untuk hasil pengukurannya dan disesuaikan dengan hasil pengukuran jarak menggunakan penggaris menunjukkan nilai yang sama, dan pada lcd baris ke 2 yaitu code 2 menampilkan hasil *counter* sensor terhadap pengukuran jarak. Untuk menghasilkan nilai pengukuran sebenarnya dapat menggunakan konversi :

$$\text{Pengukuran dalam CM} = \frac{\textit{Nilai Counter} \times 0.034442}{2}$$

Keterangan :

Nilai counter = Lebar pulsa saat diberikan trigger

0.034442 = cepat rambat diudara saat mendapatkan pantulan mengenai objek atau dinding, 0.03442 didapat dari 344.424 m/detik (atau 1 cm setiap 29.034 us) saat memantul kembali.

Dari hasil data sheet SRF04 memiliki ketetapan tersendiri, untuk menghasilkan nilai ukur dalam cm memiliki rumus :

$$\text{Pengukuran dalam CM} = \frac{\textit{Nilai Counter}}{58}$$

Tabel 4.2 Data Pengukuran SRF04

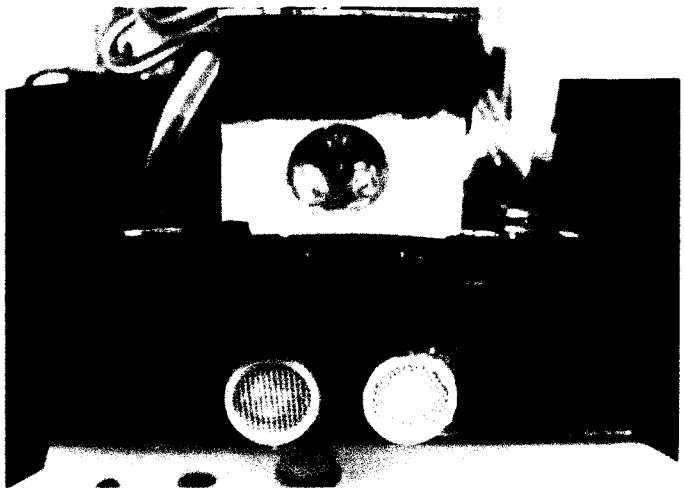
Data	Counter terukur (us)	Jarak pada LCD	Jarak sebenarnya (cm)	Jarak Dengan Rumus (cm)	Nilai error (Jarak sebenarnya – jarak dengan rumus)
1	298	5	5	5,1	-0,1
2	360	6	6	6,1	-0,1
3	585	10	10	10	0
4	705	12	12	12	0
5	878	15	15	15	0
6	996	17	17	17	0
7	1167	20	20	20	0
8	1281	22	22	22	0
9	1419	24	24	24	0
10	2323	40	40	40	0
11	3500	60	60	60	0

Dari tabel data pengukuran sensor jarak SRF-04 dapat disimpulkan, sensor jarak SRF04 dikategorikan 80 % akurat dalam pembacaan. Dari 11 kali pengujian jarak yang dihasilkan terdapat 2 data yang terjadi error.

4.3. Pengujian Sensor Warna

Sensor untuk mendeteksi warna objek digunakan sensor *photodioda* berpasangan dengan tiga LED *superbright* warna merah, hijau, dan biru yang menyala bergantian. Karakteristik pemilihan jenis sensor dan rangkaian hampir sama dengan sensor garis pada pembahasan sebelumnya. Sensor ini dipasang pada lengan dan akan bekerja hanya pada saat proses pembacaan warna objek saja. Pembacaan

nilai ADC dari keluaran sensor akan bergantian sesuai dengan warna yang sedang dipancarkan. Foto penempatan sensor warna akan ditunjukan pada Gambar:



Gambar 4.6. Sensor warna dan peletakan sensor

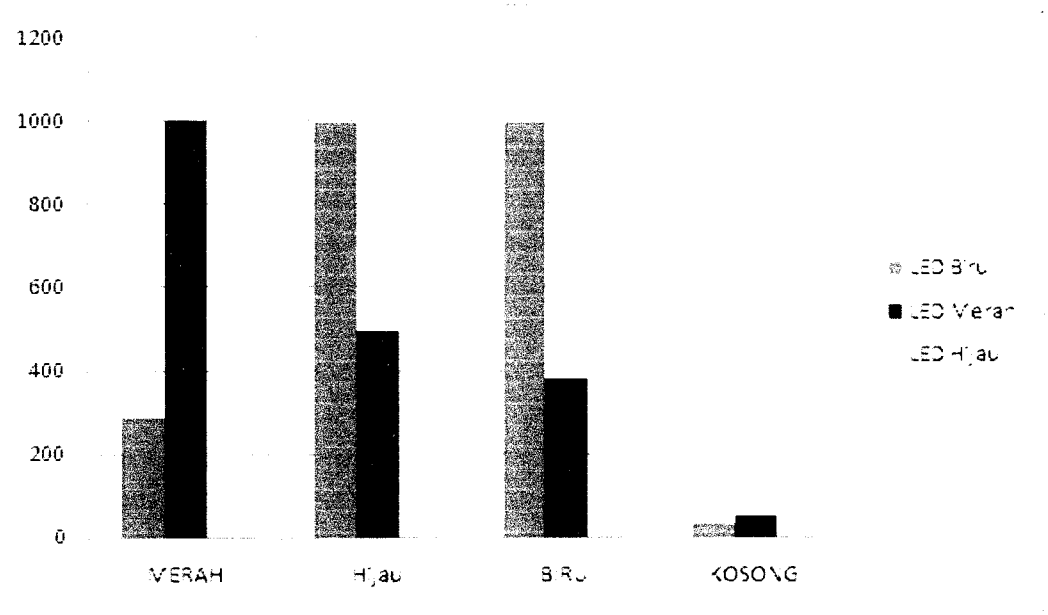
Dari rangkaian pembagi tegangan sensor ini memberikan nilai keluaran berupa tegangan antara 0 hingga 5 volt yang berubah mengikuti intensitas pantulan cahaya yang mengenainya dan akan dibaca oleh pin ADC lalu dikonversikan menjadi bilangan desimal pada setiap *variable* warnanya dengan nilai antar 0 hingga 1023. Nilai tegangan keluaran sensor saat proses *scanning* tidak dapat dibaca oleh multimeter karena perubahan tegangan untuk masing-masing warna sulit dibedakan karena terlalu cepat proses *scanning* atau pergantian nyala LED-nya. Untuk pengujian nilai RGB(*Red, Green, Blue*) hasil pantulan objek yang diuji akan dilihat langsung pada *variable* hasil konversi ADC. Hasil pembacaan tersebut dilihat pada LCD untuk

menampilkan nilai ADC untuk setiap intensitas warna yang dipantulkan dan dicatat pada Tabel data dan dipresentasikan dengan grafik.

Tabel 4.3 Data Pengukuran Sensor Warna objek Lingkaran

No Objek	Warna Objek	Pengujian Ke	ADC LED Merah	ADC LED Biru	ADC LED Hijau
1	Merah	1	995	271	250
		2	995	290	270
		3	995	290	250
		4	995	291	250
		5	995	290	250
		6	995	290	251
		7	995	291	260
		8	995	290	250
Nilai rata-rata tiap LED			**995**	**287**	**253**
2	Biru	1	384	991	140
		2	382	991	130
		3	383	991	150
		4	382	991	149
		5	382	991	150
		6	381	991	160
		7	380	991	160
		8	380	991	150
Nilai rata-rata tiap LED			**381**	**991**	**148**
3	Hijau	1	491	993	310
		2	491	994	320
		3	492	993	310
		4	493	993	310
		5	491	993	320
		6	492	993	300
		7	493	993	310
		8	493	993	300
Nilai rata-rata tiap LED			**492**	**993**	**310**

Dari table hasil pembacaan sensor warna terhadap warna objek yang akan di gunakan pada pengangkutan mendapatkan nilai rata-rata untuk setiap warna objek. Pengambilan data warna dilakukan sebanyak 8 (delapan) kali pengambilan, dan hasil pembacaan di catat untuk mendapatkan nilai rata-rata warna objek yang nantinya akan dimasukan kedalam *listing* program.



Gambar 4.7. Grafik Pengujian Sensor Warna Objek Lingkaran

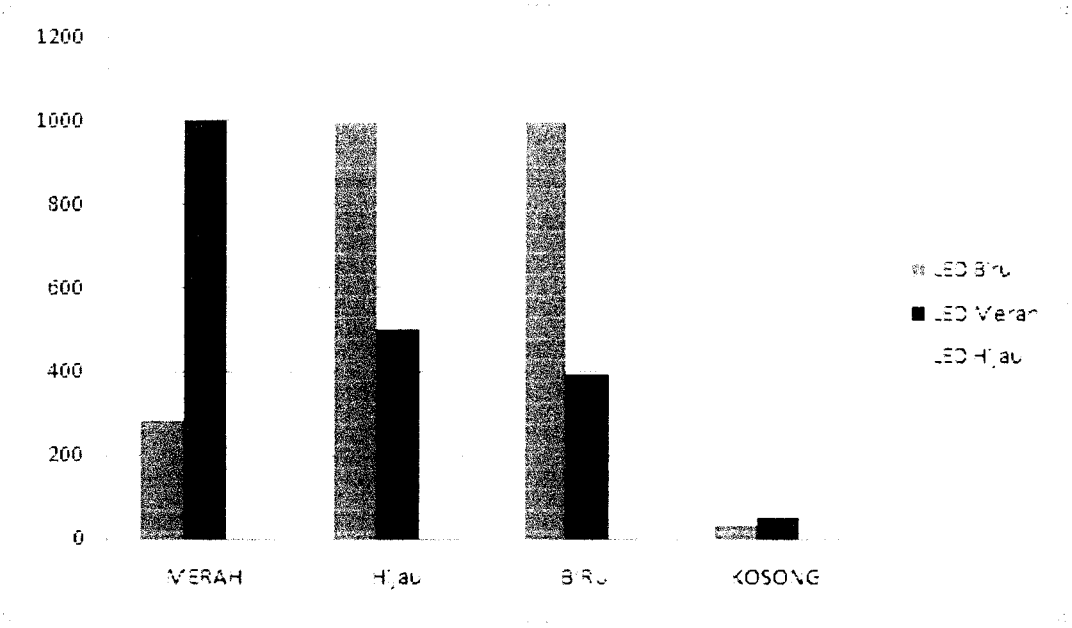
Pengujian dengan menggunakan objek lingkaran dapat dianalisa dengan grafik pembacaan ADC terhadap LED yang dipantulkan pada objek. Pada grafik merah, led biru, merah dan hijau memantulkan lebih banyak cahaya pada sensor *photodioda* dengan memiliki nilai ADC lebih besar dari 800. Grafik hijau terjadi

purubahan pada pantulan led biru, hal ini dikarenakan cahaya yang mengenai objek berwarna hijau lebih sedikit memantulkan cahaya dengan sinar berwarna biru. Grafik objek warna biru terlihat berbeda yang ditampilkan pada grafik hanya pada led hijau lebih banyak memantulkan cahaya yang menimpa pada objek biru, hal ini dikarena cahaya biru dan cahaya merah lebih banyak di serap oleh warna biru dan warna hijau lebih banyak di pantulkan pada sensor *photodioda.*

Pengujian berikutnya adalah pengambilan data dengan tidak menggunakan objek pantul atau kosong. Grafik menunjukan nilai grafik lebih kecil, hal ini maksudnya adalah sinar yang dipancarkan led – led warna tidak terjadi pemantulan cahaya pada sensor *photodiode*.

Tabel 4.4 Data Pengukuran Sensor Warna objek Datar

NO Objek	Warna Objek	ADC LED Merah	ADC LED Biru	ADC LED Hijau
1	Merah	996	280	250
2	Biru	390	991	160
3	Hijau	500	993	330
4	Kosong	30	50	80



Gambar 4.8. Grafik Pengujian Sensor Warna Objek Datar

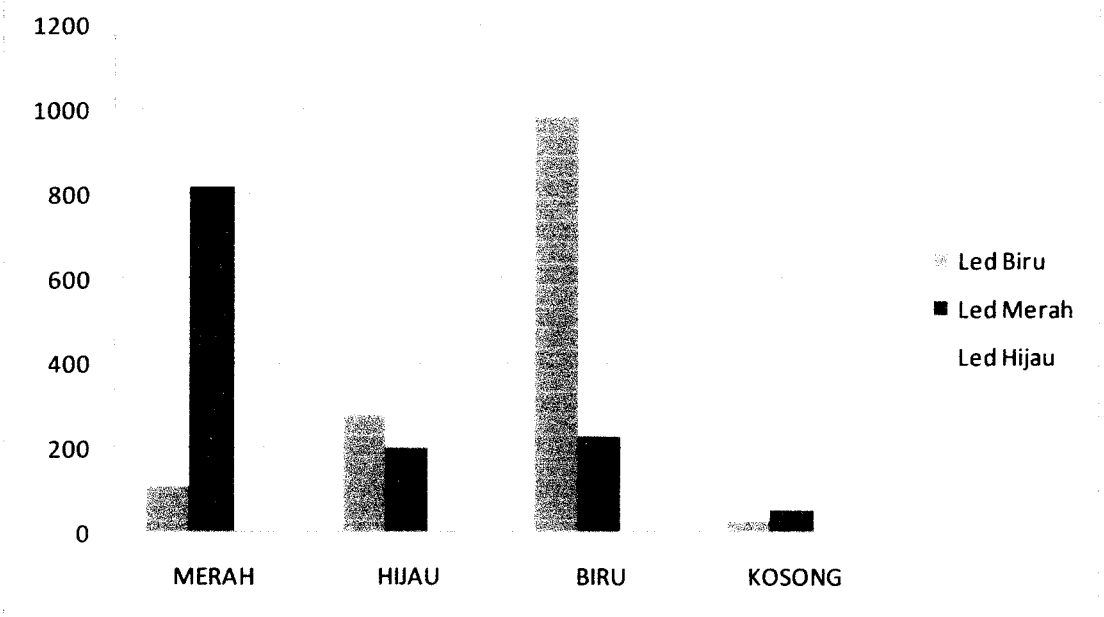
Pengujian dengan objek lengkung dan objek datar tidak jauh berbeda hasil yang ditampilkan pada grafik, hal ini dikrena jarak antara sensor dan objek tidak bebeda, sehingga pantulan cahaya akan sama yang dihasilkannya.

Selain menggunakan objek datar dan objek lengkung, pengujian dilakukan menggunakan objek yang tidak datar dan tidak lengkung dan hasil grafik sangat jauh berbeda dengan memiliki nilai data yang tidak stabil tergantung posisi saat pengambilan data pantulan cahaya. Pengujian dilakukan menggunakan kertas yang remas –remas sampai memiliki permukaan yang tidak beraturan. Hasil pengambilan data cahaya pada sensor terjadi tidak stabil dan tidak selalu sama dikarenakan objek yang selalu berubah dan pantulan cahaya led warna yang tidak sempurna memantul

pada objek target. Dengan demikian dari hasil data yang diperoleh, untuk objek yang tidak beraturan tidak direkomendasikan untuk sensor yang telah dirancang.

Tabel 4.5 Data Pengukuran Sensor Warna objek Tidak Datar

NO Objek	Warna Objek	ADC LED Biru	ADC LED Merah	ADC LED Hijau
1	Merah	110	820	70
2	Hijau	280	200	80
3	Biru	988	130	50
4	Kosong	30	50	80



Gambar 4.9. Grafik Pengujian Sensor Warna Objek Tidak Datar

Dari hasil pengujian terhadap beberapa warna objek dan bentuk dari objek terdapat beberapa perbedaan yang dapat dilihat pada hasil tabel pengukuran dan

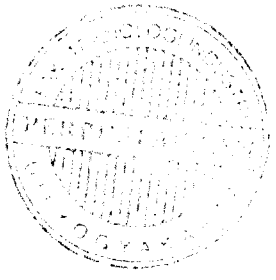
dipersentasikan kedalam grafik. Dengan hasil analisa yang didapatkan maka dalam penetapan bentuk objek yang akan digunakan adalah dengan bentuk objek lingkaran dengan alasan, nilai yang diukur saat melakukan pengujian yang diterapkan pada robot dapat dipertahankan, dengan menggunakan lengan griper yang berbentuk melengkung sehingga dapat menutupi cahaya luar yang masuk pada sensor, sehingga nilai ADC dari pengambilan data tetap sama.

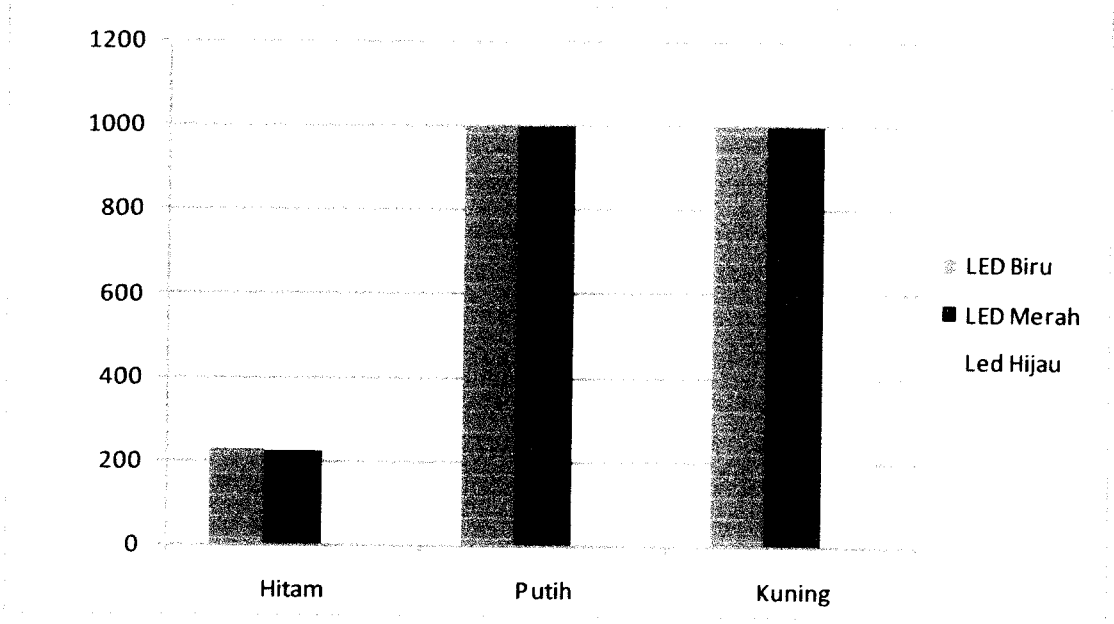
Sensor yang dirancang akan bekerja dan akan mengangkut jika data warna sesuai dengan nilai data yang dimasukan kedalam sintak program yang dicompile.

Robot tidak akan mengangkat dengan data warna duluar yang diset secara program, maka dilakukan beberapa sampel warna yang dengan nilai data yaitu dari nilai warna Hitam, Putih dan warna kuning cerah.

Tabel 4.6 Data Pengukuran Sensor dengan warna lain

NO Objek	Warna Objek	ADC LED Biru	ADC LED Merah	ADC LED Hijau
1	Hitam	227	225	40
2	Putih	996	996	64
3	Kuning	996	996	66





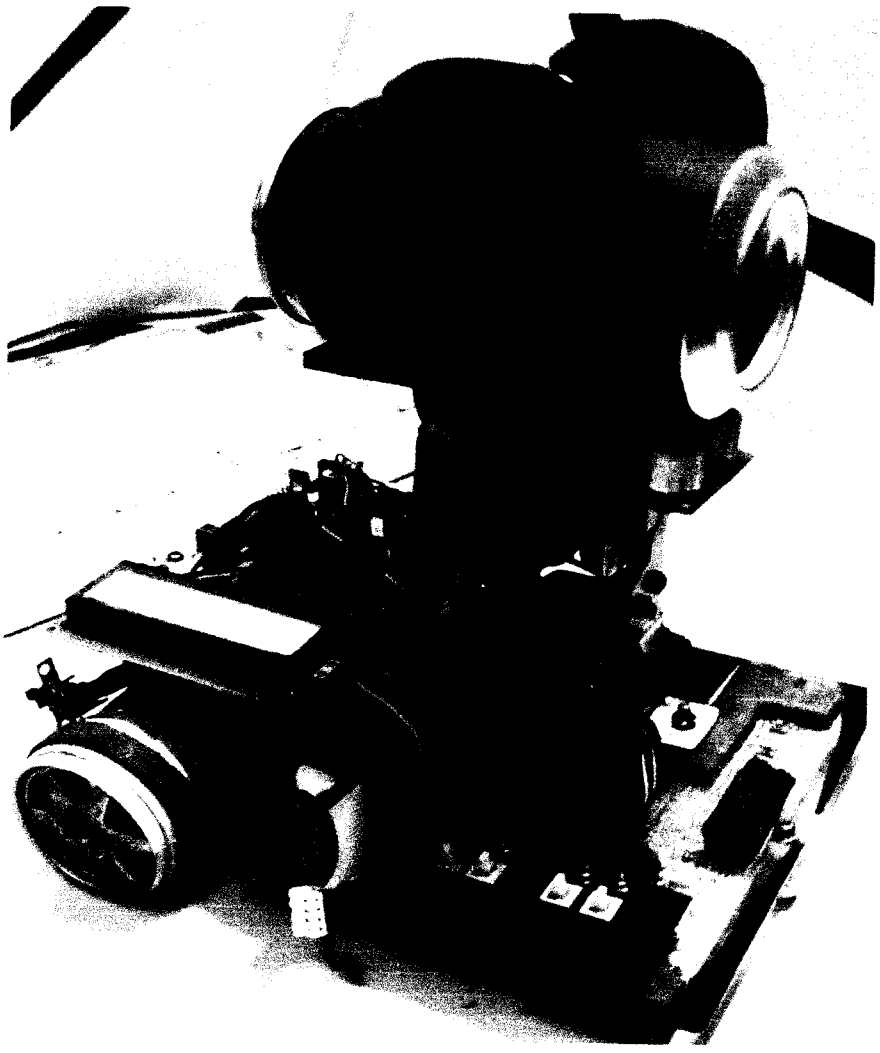
Gambar 4.10. Grafik Pengujian Sensor dengan warna lain

4.4 Pengujian dan hasil dari keseluruhan robot pengangkut barang

Dari proses perancangan dan pembuatan kerangka beserta komponen rangkain elektronika robot ini, kemudian robot diuji untuk melakukan proses pengangkutan barang berupa silinder ringan berdiameter 6 cm. Badan robot hasil dari perancangan robot ini dapt dilihat pada Gambar

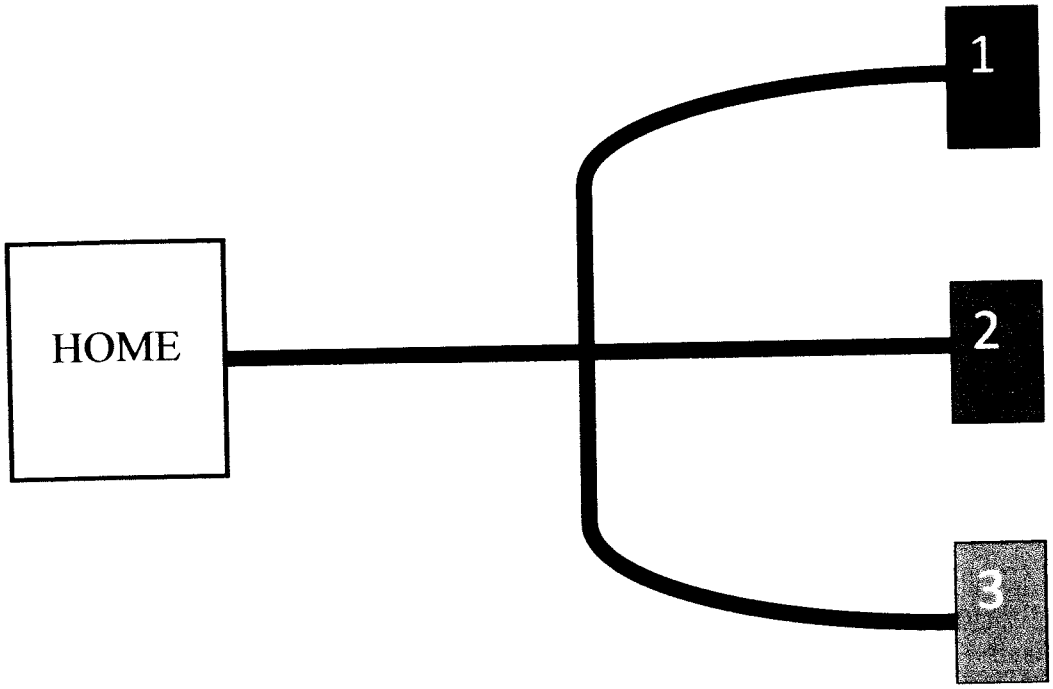
Gambar 4.11. Robot Pengangkut barang Tampak samping

Proses awal mekanik robot adalah menguji pergerakan lengan yang terdiri dari 2 buah servo penggerak dan didapatkan hasil yang cukup baik dalam proses mengambil dan meletakan objek.



Gambar 4.12. Robot saat mengangkat Objek

Percobaan selanjutnya menguji algoritma *mapping* pada arena sesungguhnya yang terdiri dari *home* atau posisi awal start dan tiga posisi target yang diurutkan sesuai warna objek yaitu Merah, Hijau, dan Biru yang dipisahkan jalurnya oleh sebuah perempatan. Arena dapat ditunjukan pada Gambar berikut:



Gambar 4.13. Arena Robot pengangkut barang

Tugas yang diberikan adalah robot dapat mengantarkan objek secara otomatis dengan cara mengambil objek dari posisi *home* kemudian sesuai warna objek robot akan meletakan objek pada target yang ditentukan. Objek yang akan dipindahkan adalah 3 buah objek warna merah yang akan diletakkan pada lokasi terget 1, 3 buah objek warna hijau yang akan diletakkan pada lokasi terget 3, dan tiga buah objek

warna biru yang akan diletakkan pada lokasi terget 2. Peletakan awal objek diletakkan pada posisi *home* secara bergantian dengan urutan warna secara acak dan robot berhasil semua objek dipindahkan dengan warna yang telah ditentukan diselesaikan dengan baik.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil ujicoba penelitian pembuatan prototipe *Robot Pengakut Barang Berdasarkan Warna* ini, maka dapat diambil kesimpulan sebagai berikut:

1. Robot menggunakan sensor garis untuk untuk mengikuti trak atau jalur lintasan arena sebagai navigasi robot. Dengan menggunakan sensor ini robot dapat berjalan dengan cepat yang dikarenakan respon sensor garis yang cukup cepat sehingga sanga mendukung untuk robot yang berjalan cepat.
2. Sensor warna yang dirancang menggunakan sensor *photodioda* yang dipasangkan dengan 3 buah *LED* warna sebagai sumber pantulan cahaya objek yang akan dipantulkan pada objek tertentu. Dengan menggunakan sensor ini aplikasi robot tidak harus menggunakan sensor mahal namun dengan sensor yang murah cara membuatnya robot tidak kalah canggih dengan menggunakan sensor pabrikan.
3. Setelah melakukan pengujian keseluruhan robot dapat menyelesaikan pemindahan barang cukup cepat dengan menggunakan motor DC gearbox, dan dapat lebih mudah untuk mengontrol faktor selip tehadap licinnya lantai.

5.2 Saran

Untuk mengembangkan sistem dimasa yang akan datang maka dapat disarankan beberapa hal sebagai berikut :

1. Perlu ditambahkan Sensor kamera untuk pencitraan bentuk dan dimensi barang. Sehingga robot dapat menentukan ukuran barang yang akan di pindahkan.
2. Dalam penggunaan sensor warna dirancang menambahkan lensa fokus dan menambahkan filter cahaya agar tidak terpengruh cahaya luar.
3. Robot dikembangkan dalam aplikasi lain seperti robot tranfortasi atau robot pengantar makanan.
4. Menambahkan modul suara agar dapat berkomunikasi langsung dengan manusia.

DAFTAR PUSTAKA

Atmel Corporation. *ATMega32 Datasheet*. 25 Agustus 2008.

http://labdasar.ee.itb.ac.id/lab/EL3006/0708/sem2/ATMega32.pdf

Nalwan Andi Paulus. 2004. *Panduan Praktis Penggunaan* dan *Antarmuka Modul LCD M1632*. Jakarta: Elek Media Komputindo.

Datasheet ultra-sonic ranger Update - May 2003.

www.robotstorehk.com/srf04tech.pdf

Konsep Dasar Modul sensor warna. 2010

http://elektronikayuk.wordpress.com/2010/09/02/merakit-dan-memprogram-sensor-warna/

Konsep Dasar pwm motor. 2010

http://projeckavr.wordpress.com/

Listing Program

```
'#=============================================================
'=            ROBOT PENGNGANGKUT BARANG BERDASARKAN WARNA
'=               MENGGUNAKAN MIKROKONTROLER ATMEGA32
'=                        0 5 5 2 4 0 0 4
'#=============================================================

$regfile = "m32def.dat"
$crystal = 11059200

'#=============================================================
Config Timer1 = Pwm , Pwm = 8 ,Compare B Pwm = Clear Down , Prescale = 64
Config Timer1 = Pwm , Compare A Pwm = Clear Down ,
Config Servos = 2 , Servo1 = Portd.0 , Servo2 = Portd.1 , Reload = 10
Config Adc = Single , Prescaler = Auto
Config Lcdpin = Pin , Db4 = Portc.3 , Db5 = Portc.2 , Db6 = Portc.4
Config Lcdpin = Pin , Db7 = Portc.5 , E = Portc.1 , Rs = Portc.0        'LCD
Config Lcd = 16 * 2
Deflcdchar 0 , 4 , 14 , 31 , 31 , 14 , 4 , 32 , 32
Cls
Cursor Off

'#=============================================================
Dim X As Byte , Kecepatan As Byte
Dim Ep_kp As Eram Byte , Ep_kd As Eram Byte , Ep_ki As Eram Byte
Dim Kp As Byte , Kd As Byte , Ki As Byte
Dim Eror_lalu As Integer , Eror As Integer ,
Dim Motkan As Integer ,
Dim Motkir As Integer
Dim P As Integer , D As Integer , I As Integer , Z_i As Integer
Dim Temp_kan As Word , Temp_kir As Word
Dim Sp_kan As Integer , Sp_kir As Integer
Dim Delta_eror As Integer , Pid As Integer

Dim Dat_warna As Word , Data_srf As Word
Dim Warna_merah As Word , Warna_biru As Word , Warna_hijau As Word
Dim Perjalan As Byte
Dim Set_simpang As Byte , Ep_simpang As Eram Byte
Dim Lines As Byte , X_max_e As Eram Byte , X_max As Byte , Titip As Byte
Dim Ep_ln1 As Eram Word , Ep_ln2 As Eram Word , Ep_ln3 As Eram Word ,
Dim Ep_ln4 As Eram Word , Ep_ln5 As Eram Word , Ep_ln6 As Eram Word ,
Dim Ep_ln7 As Eram Word , Ep_ln8 As Eram Word
Dim Ref_ln1 As Word , Ref_ln2 As Word , Ref_ln3 As Word , Ref_ln4 As Word ,
Dim Ref_ln5 As Word , Ref_ln6 As Word , Ref_ln7 As Word , Ref_ln8 As Word
Dim Dat_ln1 As Word , Dat_ln2 As Word ,
Dim Dat_ln3 As Word , Dat_ln4 As Word , Dat_ln5 As Word , Dat_ln6 As Word ,
Dim Dat_ln7 As Word , Dat_ln8 As Word
Dim Hi_ln1 As Word , Hi_ln2 As Word , Hi_ln3 As Word , Hi_ln4 As Word ,
Dim Hi_ln5 As Word , Hi_ln6 As Word , Hi_ln7 As Word , Hi_ln8 As Word
Dim Lo_ln1 As Word , Lo_ln2 As Word , Lo_ln3 As Word , Lo_ln4 As Word ,
Dim Lo_ln5 As Word , Lo_ln6 As Word , Lo_ln7 As Word , Lo_ln8 As Word ,
Dim Waktu_muter As Word
```

```
'#=============================================================================
  Declare Sub Ngeline
'#=============================================================================
  Ln0 Alias Lines.0
  Ln1 Alias Lines.1
  Ln2 Alias Lines.2
  Ln3 Alias Lines.3
  Ln4 Alias Lines.4
  Ln5 Alias Lines.5
  Ln6 Alias Lines.6
  Ln7 Alias Lines.7

  Sw_a Alias Pinb.3
  Sw_b Alias Pinb.2
  Sw_c Alias Pinb.1
  Sw_d Alias Pinb.0

  Buzer Alias Portb.4
  Led_merah Alias Portb.5
  Led_hijau Alias Portb.6
  Led_biru Alias Portb.7

  H_t Alias Set_simpang.0
  New_pd Alias Set_simpang.5
  On_int Alias Set_simpang.4
  On_adc Alias Set_simpang.3

  Init_srf Alias Portc.7                                    'Triger SRF
  Echo_srf Alias Pinc.6                                     'Output SRF

  D_l Alias Portd.3
  D_r Alias Portd.7
  D_la Alias Portd.2
  D_ra Alias Portd.6

  Const Hidup = 1
  Const Mati = 0
  Const Maju = 0
  Const Mundur = 1
  Const Naik = 66
  Const Turun = 20
  Const Buka = 50
  Const Tutup = 78
  Const Merah = 1
  Const Biru = 2
  Const Hijau = 3


'#=============================================================
  Ddrc.6 = 0
  Ddrc.7 = 1
   Ddrb = &B11110000
  Portb = &B11111111
  Ddra.1 = 1
  Porta.1 = 1
  Portd = &HFF
```

```
  Portc = 255

'#=================================================================
  Led_merah = 0
  Led_biru = 0
  Led_hijau = 0

  Eror_lalu = 0
  Eror = 0
  P = 0
  I = 0
  D = 0
  Z_i = 0
  Kp = Ep_kp
  Kd = Ep_kd
  Ki = Ep_ki

  X_max = X_max_e
  Set_simpang = Ep_simpang

  Ref_ln1 = Ep_ln1
  Ref_ln2 = Ep_ln2
  Ref_ln3 = Ep_ln3
  Ref_ln4 = Ep_ln4
  Ref_ln5 = Ep_ln5
  Ref_ln6 = Ep_ln6
  Ref_ln7 = Ep_ln7
  Ref_ln8 = Ep_ln8

'#=================================================================
  Enable Interrupts
  Servo(1) = Tutup
  Servo(2) = Naik

  Gosub Majulah
  Pwm1a = 0
  Pwm1b = 0
  Lcd "Robot Pengangkut"
  Locate 2 , 1
  Lcd "Barang Berdsarkan Warna "
  Waitms 300

  Led_merah = 0
  Led_biru = 0
  Led_hijau = 0

'#=================================================================
  Awal:
  Buzer = 1
  Waitms 1
  Buzer = 0
  Cls
  Waitms 200
  Lcd "Maen     Setting"
  Lowerline
  Lcd "  Coba  Test"
  Do
```

```
 If Sw_b = 0 Then Goto Set_kec
 If Sw_a = 0 Then Goto Main_cerdas
 If Sw_c = 0 Then Goto Coba_lengan
 If Sw_d = 0 Then Goto Setting
 Loop

 Set_kec:
 Cls
 Lcd "OK  back   +   -"
 Locate 2 , 1
 Lcd "  kec = "
 Waitms 200

Kecepatan = 5
 Do
   If Kecepatan = 0 Then Kecepatan = 1
   If Kecepatan > 10 Then Kecepatan = 10

  Locate 2 , 9
   Lcd Kecepatan ; "  "
   Waitms 100
   If Sw_a = 0 Then Goto Siap_go
   If Sw_b = 0 Then Goto Awal
   If Sw_c = 0 Then Incr Kecepatan
   If Sw_d = 0 Then Decr Kecepatan
Loop

 Main_cerdas:
 Cls
 Lcd "OK  back   +   -"
 Locate 2 , 1
 Lcd "  kec = "
 Waitms 200
 Kecepatan = 5
 Do
    If Kecepatan = 0 Then Kecepatan = 1
    If Kecepatan > 10 Then Kecepatan = 10
   Locate 2 , 9
Lcd Kecepatan ; "  "
Waitms 100
    If Sw_a = 0 Then Exit Do
    If Sw_b = 0 Then Goto Awal
    If Sw_c = 0 Then Incr Kecepatan
    If Sw_d = 0 Then Decr Kecepatan
 Loop
 Wait 1
 Cls
 Lcd "Run Way...(^_^)"

 Do
  Do
  Call Ngeline
  Gosub Conversi
  If Ln2 = 0 And Ln3 = 0 And Ln4 = 0 And Ln5 = 0 Then Exit Do
  Loop
  Gosub Ngerem
```

```
 Scan_objek:
      Gosub Grip_turun
      Gosub Jepit_buka

      Do
      Gosub Baca_warna_jarak
      Loop Until Data_srf < 10
      Gosub Maju_dikit
      Gosub Jepit_tutup
      Gosub Baca_warna

'==========Scan Warna Untuk Memilih Perjalanan ======================

If Warna_merah > 500 And Warna_biru < 500 And Warna_hijau < 500 Then
Perjalan = Merah

Elseif Warna_merah < 500 And Warna_biru > 500 And Warna_hijau < 200 Then
Perjalan = Biru

Elseif Warna_merah > 300 And Warna_biru > 500 And Warna_hijau > 150 And
Warna_hijau < 500 Then
Perjalan = Hijau

'====================================================================
   Else
     Gosub Jepit_buka
     Gosub Mundur_dikit
     Goto Scan_objek
   End If
     Gosub Grip_naik
     Gosub Mundur_dikit
     Gosub Balik_kiri                                    ' Buat Belok Kiri

     Do
     Call Ngeline
     Gosub Conversi
     If Ln2 = 0 And Ln3 = 0 And Ln4 = 0 And Ln5 = 0 Then Exit Do
     Loop

   If Perjalan = Merah Then
   Gosub Belok_kiri

   Elseif Perjalan = Biru Then
   For X = 1 To 30
     Call Ngeline
     Next X
     Else

     Gosub Belok_kanan
     End If

     Do
     Call Ngeline
     Gosub Conversi
     If Ln2 = 0 And Ln3 = 0 And Ln4 = 0 And Ln5 = 0 Then Exit Do
     Loop
```

```
      Gosub Grip_turun
      Gosub Jepit_buka
      Gosub Maju_dikit
      Led_merah = 1
      Led_biru = 1
      Led_hijau = 1
      Gosub Mundur_dikit
      Gosub Grip_naik
      Gosub Jepit_tutup

      If Perjalan = Merah Then
         Gosub Balik_kanan
      Elseif Perjalan = Biru Then
         Gosub Balik_kanan
      Else
         Gosub Balik_kiri
      End If

   Do
   Call Ngeline
   Gosub Conversi
   If Ln2 = 0 And Ln3 = 0 And Ln4 = 0 And Ln5 = 0 Then Exit Do
   Loop
   If Perjalan = Merah Then
      Gosub Belok_kanan
      Elseif Perjalan = Biru Then
      For X = 1 To 40
      Call Ngeline
      Next X
      Else
      Gosub Belok_kiri
      End If
      Loop

'#=============================================================================

  Coba_lengan:
  Cls
  Lcd "ADC       Exit"
  Locate 2 , 1
  Lcd "Turun     Naik"
  Waitms 200
  Do
  If Sw_a = 0 Then
  Waitms 100
     Do
     Gosub Jepit_buka
     Gosub Grip_turun
     Gosub Baca_warna
  Locate 1 , 1
  Lcd "SRF=" ; Data_srf ; "   "
  Locate 2 , 1
  Lcd "R" ; Warna_merah ; " B" ; Warna_biru ; " G" ; Warna_hijau ; "      "
  Loop Until Sw_d = 0
  Goto Coba_lengan
  End If
```

```
  If Sw_b = 0 Then Gosub Coba_grip
  If Sw_c = 0 Then Gosub Coba_japit
  If Sw_d = 0 Then Goto Awal
  Loop

'#=============================================================================
Coba_grip:
   Cls
   Lcd "Naik       Awal"
   Locate 2 , 1
   Lcd "Turun    Lengan"
   Waitms 200
   Do
   If Sw_a = 0 Then
   Gosub Grip_naik
   End If

   If Sw_b = 0 Then
   Gosub Grip_turun
   End If

   If Sw_c = 0 Then Gosub Coba_lengan
   If Sw_d = 0 Then Goto Awal
   Loop

Coba_japit:
   Cls
   Lcd "Buka       Awal"
   Locate 2 , 1
   Lcd "Tutup    Lengan"
   Waitms 200
   Do
   If Sw_a = 0 Then
   Gosub Jepit_buka
   End If

   If Sw_b = 0 Then
   Gosub Jepit_tutup
   End If

   If Sw_c = 0 Then Gosub Coba_lengan
   If Sw_d = 0 Then Goto Awal
   Loop

'#=============================================================================
Baca_warna:
  Led_merah = 0
  Led_biru = 1
  Led_hijau = 0
  Waitms 20
  Gosub Baca_warna_jarak
  Warna_biru = Dat_warna
  Waitms 10

  Led_merah = 1
  Led_biru = 0
  Led_hijau = 0
```

```
  Waitms 20
  Gosub Baca_warna_jarak
  Warna_merah = Dat_warna
  Waitms 10

  Led_merah = 0
  Led_biru = 0
  Led_hijau = 1
  Waitms 20
  Gosub Baca_warna_jarak
  Warna_hijau = Dat_warna * 10
  Waitms 30
  Return

'#=============================================================
Jepit_buka:
   Servo(1) = Buka
   Waitms 500
   Return

Jepit_pegang:
   Servo(1) = Tutup
   Waitms 500
   Return

Jepit_tutup:
   Servo(1) = Tutup
   Waitms 500
   Return

Grip_turun:
   Servo(2) = Turun

   Waitms 500
   Return

Grip_naik:
   Servo(2) = Naik
   Wait 1
   Return

'#=============================================================
Siap_go:
   Cls
   Lcd "   Jalan Biasa"
   Locate 2 , 1
   Lcd "   Robot Siap !    "
   Waitms 300
   Do
   If Sw_a = 0 Or Sw_b = 0 Or Sw_c = 0 Or Sw_d = 0 Then
   Waitms 300
   Cls
   Lcd "Run Way...(^_^)"
   Do
   Call Ngeline
   Loop
   End If
```

```
   Loop

'#=================================================================
Sub Ngeline
Gosub Conversi
Select Case Lines
Case &B11111100 : Eror = 8
Case &B11111000 : Eror = 6
Case &B11111011 : Eror = 4
Case &B11110011 : Eror = 2
Case &B11110111 : Eror = 1
Case &B11100111 : Eror = 0
Case &B11101111 : Eror = -1
Case &B11001111 : Eror = -2
Case &B11011111 : Eror = -4
Case &B00011111 : Eror = -6
Case &B00111111 : Eror = -8
End Select

#=================================================================
- Temp_Kanan/Temp_kiri adalah Range kecepatan yang disimpan di eeprom
  yg di atur sebesar 255
- 255 adalah pengaturan kecepatan maksimal dari motor

- Temp_kanan/10 atau Temp_kiri /10 adalah pembagian kecepatan yang
  dapat di atur nilainya
- sebesar 255/10 =25,5 . jadi di dalam program ada menu yg dapat mengatur
  kecepatnnya yang rangenya 1 s/d 10, untuk tiap range mewakili 25,5.

- jika kita men set nilai kecepatan 2 maka kecepatan maksimal
  motor 25,5 x 2 = 51
#=================================================================
   Sp_kan = Temp_kan / 10
   Sp_kir = Temp_kir / 10

' Increments/Decrements kecepatan eeprom
   Temp_kan = 255 * Kecepatan                                    '
   Temp_kir = 255 * Kecepatan

   If Eror = 0 Then Z_i = 0
   P = Kp * Eror
   I = Ki * Eror
   Z_i = I + Z_i

   Delta_eror = Eror - Eror_lalu
   D = Kd * Delta_eror

   Pid = P + Z_i
   Pid = D + Pid
   Eror_lalu = Eror

   Motkan = Sp_kan - Pid
   Motkir = Sp_kir + Pid
   Motkan = Sp_kan + Pid
   Motkir = Sp_kir - Pid
```

```
'Motkan adalah motor kanan, Sp_kanan adalah set point arah putaran dari hasil
perhitungan
'eror baca sensor

   If Motkan >= 0 And Motkir >= 0 Then Gosub Majulah
   If Motkan < 0 And Motkir >= 0 Then Gosub Kananlah        ' batasan PWM
   If Motkan >= 0 And Motkir < 0 Then Gosub Kirilah

'Hubungan antara nilai eror sensor, dan arah putaran motor
'jika pembacaan sensor mengarahkan motor lebih kecil dari 0 eror sensor,
'maka motor akan mundur

   If Motkan < 0 Then Motkan = 0 - Motkan
   If Motkir < 0 Then Motkir = 0 - Motkir

'jika pembacaan sensor mengarahkan motor lebih besar dari 0 eror sensor,
'maka motor akan + atau arah maju

   If Motkan > 255 Then Motkan = 255
   If Motkir > 255 Then Motkir = 255

   Pwm1a = Motkan                                           ' isi PWMnya
   Pwm1b = Motkir

   Waitms X_max
   End Sub

Majulah:
   D_r = 0
   D_l = 0
   D_ra = 1
   D_la = 1
   Return

Kirilah:
   D_r = 0
   D_l = 1
   D_ra = 1
   D_la = 0
   Return

Kananlah:
   D_r = 1
   D_l = 0
   D_ra = 0
   D_la = 1
   Return

Mundurlah:
   D_r = 1
   D_l = 1
   D_ra = 0
   D_la = 0
   Return
```

```
'#=============================================================
Belok_kiri:
  Gosub Kekiri
  Waitms 100
  Do
  Gosub Conversi                                          ' Ngeline ADC nya
  Loop Until Ln3 = 0 Or Ln4 = 0
  Gosub Kekanan
  Waitms 50
    Gosub Diam_dulu
    Return

Belok_kanan:
   Gosub Kekanan
   Waitms 100
   Do
   Gosub Conversi
   Loop Until Ln3 = 0 Or Ln4 = 0
   Gosub Kekiri
   Waitms 50
   Gosub Diam_dulu
   Return

Puter_kiri:
   Gosub Pukiri
   Waitms 100
   Do
   Gosub Conversi
   Loop Until Ln3 = 0 Or Ln4 = 0
   Gosub Kekanan
   Waitms 50
   Gosub Diam_dulu
   Return

Puter_kanan:
   Gosub Pukanan
   Waitms 100
   Do
     Gosub Conversi
     Loop Until Ln3 = 0 Or Ln4 = 0
     Gosub Kekiri
     Waitms 50
     Gosub Diam_dulu
     Return

Balik_kiri:
   Gosub Kekiri
   Waitms 200                                             'Waktu belok Kiri
300 oke
   Do
     Gosub Conversi
     Loop Until Ln3 = 0 Or Ln4 = 0
     Gosub Kekanan
     Waitms 90                                            'Waktu kembali
menuju Ngeline
```

```
      Gosub Diam_dulu
      Return

Balik_kanan:
   Gosub Kekanan
   Waitms 200                                          'Waktu belok
Kanan 300 oke
   Do
     Gosub Conversi
     Loop Until Ln3 = 0 Or Ln4 = 0
     Gosub Kekiri
     Waitms 90                                         'Waktu kembali
menuju Ngeline
     Gosub Diam_dulu
     Return

'#=============================================================
Kekiri:
  D_r = 0
  D_l = 1
  D_ra = 1
  D_la = 0
  Pwm1a = 150
  Pwm1b = 150
  Waitms 10
  Return

Pukiri:
  D_r = 0
  D_l = 1
  D_ra = 1
  D_la = 0
  Pwm1a = 150                                          'kanan
  Pwm1b = 50
  Waitms 10
  Return

Kekanan:
  D_r = 1
  D_l = 0
  D_ra = 0
  D_la = 1
  Pwm1a = 150
  Pwm1b = 150
  Waitms 10
  Return

Pukanan:
  D_r = 1
  D_l = 0
  D_ra = 0
  D_la = 1
  Pwm1a = 50
  Pwm1b = 150
  Waitms 10
  Return
```

```
Diam_dulu:
  D_r = 1
  D_l = 1
  D_ra = 0
  D_la = 0
  Pwm1a = 0
  Pwm1b = 0
  Return
Ngerem:
  D_r = 1
  D_l = 1
  D_ra = 0
  D_la = 0
  Pwm1a = 100
  Pwm1b = 100
  Waitms 50
  Gosub Diam_dulu
  Return
Ngerem_bgt:
  D_r = 1
  D_l = 1
  D_ra = 0
  D_la = 0
  Pwm1a = 200
  Pwm1b = 200
  Waitms 200
  Gosub Diam_dulu
  Return
Maju_dikit:
  D_r = 0
  D_l = 0
  D_ra = 1
  D_la = 1
  Pwm1a = 120
  Pwm1b = 120
  Waitms 200
  Gosub Diam_dulu
  Return
Mundur_dikit:
  D_r = 1
  D_l = 1
  D_ra = 0
  D_la = 0
  Pwm1a = 120
  Pwm1b = 120
  Waitms 200
  Gosub Diam_dulu
  Return

Baca_adc:
  Start Adc
   Dat_ln1 = Getadc(2)
   Dat_ln2 = Getadc(2)
   Dat_ln3 = Getadc(3)
   Dat_ln4 = Getadc(4)
   Dat_ln5 = Getadc(5)
   Dat_ln6 = Getadc(6)
```

```
   Dat_ln7 = Getadc(7)
   Dat_ln8 = Getadc(7)
   Stop Adc
   Return

'#=============================================================
Baca_warna_jarak:
 Gosub Baca_jarak
 Start Adc
 Dat_warna = Getadc(0)
 Stop Adc                                                    ' / 10
 Dat_warna = Dat_warna
 Return

'#=============================================================
Baca_jarak:
   Init_srf = 1
   Waitus 10
   Init_srf = 0
   Data_srf = 0
   Do
      Waitus 1
      Incr Data_srf
      If Echo_srf = 1 Then Exit Do
      Loop Until Data_srf = 1000

 Data_srf = 0
 Do
 Waitus 1
 Incr Data_srf
 If Echo_srf = 0 Then Exit Do
 If Data_srf > 3000 Then Exit Do
 Loop

 'Waitms 10
 Data_srf = Data_srf / 10
 Return

 '#=============================================================
 Conversi:
 Gosub Baca_adc
 If Dat_ln1 > Ref_ln1 Then Ln0 = 1
 If Dat_ln1 < Ref_ln1 Then Ln0 = 0
 If Dat_ln2 > Ref_ln2 Then Ln1 = 1
 If Dat_ln2 < Ref_ln2 Then Ln1 = 0
 If Dat_ln3 > Ref_ln3 Then Ln2 = 1
 If Dat_ln3 < Ref_ln3 Then Ln2 = 0
 If Dat_ln4 > Ref_ln4 Then Ln3 = 1
 If Dat_ln4 < Ref_ln4 Then Ln3 = 0
 If Dat_ln5 > Ref_ln5 Then Ln4 = 1
 If Dat_ln5 < Ref_ln5 Then Ln4 = 0
 If Dat_ln6 > Ref_ln6 Then Ln5 = 1
 If Dat_ln6 < Ref_ln6 Then Ln5 = 0
 If Dat_ln7 > Ref_ln7 Then Ln6 = 1
 If Dat_ln7 < Ref_ln7 Then Ln6 = 0
 If Dat_ln8 > Ref_ln8 Then Ln7 = 1
 If Dat_ln8 < Ref_ln8 Then Ln7 = 0
```

```
If H_t = 1 Then Lines = Not Lines
 Return


'#=============================================================================

 Setting:
 Cls
 Lcd " PID  T  S ARENA"
 Lcd "  Pengaturan ?"
 Waitms 300
 Do
 If Sw_a = 0 Then
   Waitms 300
   Goto Atur_kpd
 End If
 If Sw_b = 0 Then
   Waitms 300
    Goto Atur_sampling
 End If
 If Sw_c = 0 Then
   Goto Atur_rot
 End If
 If Sw_d = 0 Then
   Goto Setting_line
 End If
 Loop

  Atur_kpd:
  Cls
  Lcd "<OKE> KPD   +  -"
  Atur_kp:
  Locate 2 , 4
  Lcd "Kp ="
  Do
     Locate 2 , 9
 Lcd Kp ; "  "
 If Sw_a = 0 Then
 Ep_kp = Kp
 Ep_kd = Kd
 Ep_ki = Ki
 Goto Set_slese
  End If
   If Sw_b = 0 Then
   Waitms 200
   Goto Atur_kd
   End If
    If Sw_c = 0 Then
 Waitms 40
    Incr Kp
    End If
    If Sw_d = 0 Then
    Waitms 40
  Decr Kp
     End If
   Loop
   Atur_kd:
```

```
Locate 2 , 4
Lcd "Kd ="
 Do
   Locate 2 , 9
  Lcd Kd ; "  "
  If Sw_a = 0 Then
     Ep_kp = Kp
     Ep_kd = Kd
     Ep_ki = Ki
      Goto Set_slese
      End If
   If Sw_b = 0 Then
      Waitms 200
       Goto Atur_ki
    End If
    If Sw_c = 0 Then
       Waitms 40
       Incr Kd
End If
    If Sw_d = 0 Then
       Waitms 40
       Decr Kd
    End If
 Loop
 Atur_ki:
 Locate 2 , 4
 Lcd "Ki ="
 Do
    Locate 2 , 9
    Lcd Ki ; "  "
   If Sw_a = 0 Then
      Ep_kp = Kp
      Ep_kd = Kd
       Ep_ki = Ki
       Goto Set_slese
    End If
   If Sw_b = 0 Then
       Waitms 200
      Goto Atur_kp
    End If
If Sw_c = 0 Then
       Waitms 40
       Incr Ki
End If
   If Sw_d = 0 Then
      Waitms 40
       Decr Ki
    End If
 Loop


 Credit:
 Cls
Lcd "Menu      Credit"
 Waitms 300
 Do
 If Sw_a = 0 Then Goto Set_slese
```

```
  If Sw_d = 0 Then

'#================ Tampilan Menu Credit Pembuat Program =========
Cls
Locate 1 , 18
Lcd "ROBOT PENGNGUT BARANG + WARNA ATmega32"
      For X = 1 To 23
      Shiftlcd Left
      Waitms 100
      Next
Cls
Locate 1 , 1
Lcd "Dibuat Oleh SIGIT -TE-Uii"
      For X = 1 To 23
      Shiftlcd Left
      Waitms 100
      Next
      Wait 1
      Cls

'#=============================================================

Goto Set_slese
    End If
    Loop

Atur_sampling:
Cls
Lcd "<OKE>        +  -"
Locate 2 , 1
  Lcd " Time Sampling  "
  Do
    Locate 1 , 7
    Lcd X_max ; "ms" ; "   "
    Locate 1 , 13
    Lcd "+"
         If Sw_a = 0 Then
         X_max_e = X_max
         Goto Set_slese
    End If
         If Sw_c = 0 Then
         Waitms 40
         Incr X_max                                    ' + Time sampling
    End If
         If Sw_d = 0 Then
         Waitms 40
         Decr X_max                                    ' - Time sampling
       End If
  Loop
' incr/decr merupakan untuk menambahkan dan mengurangkan nilai variabel data
' contoh di atas X_mas yng berisikan Time sampling yg bisa +/-

'=============================================================
Atur_rot:
  Waitms 200
  Cls
  Do
```

```
   Gosub Baca_adc
  'Dat_ln2 = Dat_ln2 / 10
  'Dat_ln3 = Dat_ln3 / 10
  'Dat_ln4 = Dat_ln4 / 10
  'Dat_ln5 = Dat_ln5 / 10
  'Dat_ln6 = Dat_ln6 / 10
  'Dat_ln7 = Dat_ln7 / 10
   Locate 1 , 1
   Lcd Dat_ln2 ; "  " ; Dat_ln3 ; "  " ; Dat_ln4 ; "  "
   Locate 2 , 1
    Lcd Dat_ln5 ; "  " ; Dat_ln6 ; "  " ; Dat_ln7 ; "  "
 Loop Until Sw_a = 0
 Goto Set_slese


'#-----------------------------------------------------------------

 Setting_line:
  Cls
  Lcd "<Liat>    <Reff>"
  Waitms 300
  Do
 If Sw_a = 0 Then Goto Lihat_sensor
 If Sw_d = 0 Then Exit Do
  Loop
  Cls
  Lcd "Auto      Manual"
  Waitms 300
 Do
 If Sw_a = 0 Then Exit Do
 Loop

  Cls
 Lcd "    Start?"
  Locate 2 , 1
  Lcd "  Scanning..."
 Waitms 300
 Do
 If Sw_b = 0 Or Sw_c = 0 Then Exit Do
 Loop
 Locate 1 , 1
  Lcd "               "
 Wait 1
 Locate 1 , 1
  Lcd "   Proses..."
 Gosub Kirilah
 Pwm1a = 150
 Pwm1b = 200

 Gosub Baca_adc
 Hi_ln1 = Dat_ln1
 Hi_ln2 = Dat_ln2
 Hi_ln3 = Dat_ln3
 Hi_ln4 = Dat_ln4
 Hi_ln5 = Dat_ln5
 Hi_ln6 = Dat_ln6
 Hi_ln7 = Dat_ln7
```

```
 Hi_ln8 = Dat_ln8
 Lo_ln1 = Dat_ln1
 Lo_ln2 = Dat_ln2
 Lo_ln3 = Dat_ln3
 Lo_ln4 = Dat_ln4
 Lo_ln5 = Dat_ln5
 Lo_ln6 = Dat_ln6
 Lo_ln7 = Dat_ln7
 Lo_ln8 = Dat_ln8
 Waktu_muter = 0
Do
Gosub Baca_adc
If Dat_ln1 > Hi_ln1 Then Hi_ln1 = Dat_ln1
If Dat_ln1 < Lo_ln1 Then Lo_ln1 = Dat_ln1
If Dat_ln2 > Hi_ln2 Then Hi_ln2 = Dat_ln2
If Dat_ln2 < Lo_ln2 Then Lo_ln2 = Dat_ln2
If Dat_ln3 > Hi_ln3 Then Hi_ln3 = Dat_ln3
If Dat_ln3 < Lo_ln3 Then Lo_ln3 = Dat_ln3
If Dat_ln4 > Hi_ln4 Then Hi_ln4 = Dat_ln4
If Dat_ln4 < Lo_ln4 Then Lo_ln4 = Dat_ln4
If Dat_ln5 > Hi_ln5 Then Hi_ln5 = Dat_ln5
If Dat_ln5 < Lo_ln5 Then Lo_ln5 = Dat_ln5
If Dat_ln6 > Hi_ln6 Then Hi_ln6 = Dat_ln6
If Dat_ln6 < Lo_ln6 Then Lo_ln6 = Dat_ln6
If Dat_ln7 > Hi_ln7 Then Hi_ln7 = Dat_ln7
If Dat_ln7 < Lo_ln7 Then Lo_ln7 = Dat_ln7
If Dat_ln8 > Hi_ln8 Then Hi_ln8 = Dat_ln8
If Dat_ln8 < Lo_ln8 Then Lo_ln8 = Dat_ln8

Waitms 1
Incr Waktu_muter
Loop Until Waktu_muter = 1000
Ref_ln1 = Hi_ln1 + Lo_ln1
Ref_ln2 = Hi_ln2 + Lo_ln2
Ref_ln3 = Hi_ln3 + Lo_ln3
Ref_ln4 = Hi_ln4 + Lo_ln4
Ref_ln5 = Hi_ln5 + Lo_ln5
Ref_ln6 = Hi_ln6 + Lo_ln6
Ref_ln7 = Hi_ln7 + Lo_ln7
Ref_ln8 = Hi_ln8 + Lo_ln8

Ref_ln1 = Ref_ln1 / 2
Ref_ln2 = Ref_ln2 / 2
Ref_ln3 = Ref_ln3 / 2
Ref_ln4 = Ref_ln4 / 2
Ref_ln5 = Ref_ln5 / 2
Ref_ln6 = Ref_ln6 / 2
Ref_ln7 = Ref_ln7 / 2
Ref_ln8 = Ref_ln8 / 2

Ep_ln1 = Ref_ln1
Ep_ln2 = Ref_ln2
Ep_ln3 = Ref_ln3
Ep_ln4 = Ref_ln4
Ep_ln5 = Ref_ln5
Ep_ln6 = Ref_ln6
Ep_ln7 = Ref_ln7
```

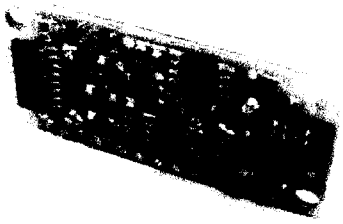
```
Ep_ln8 = Ref_ln8
Locate 1 , 1
Lcd "   Cari..   "

Gosub Conversi
If Ln3 = 0 Then
Do
 Gosub Conversi
 Loop Until Ln3 = 1
 Else
 Do
   Gosub Conversi
   Loop Until Ln3 = 0
  End If
 Gosub Diam_dulu
 Cls
 Lcd "  Scanning"
 Locate 2 , 1
 Lcd "  Selesai!"
 Waitms 300
 Goto Lihat_sensor


'# Proses Perekaman kalibrasi EEPROM ============================
Reff_step:
Cls
Lcd "<OKE>"
  Locate 2 , 1
  Lcd " Cari Putih "
 Waitms 300

 Do
    Gosub Baca_adc
  Loop Until Sw_a = 0

Hi_ln1 = Dat_ln1
Hi_ln2 = Dat_ln2
Hi_ln3 = Dat_ln3
Hi_ln4 = Dat_ln4
Hi_ln5 = Dat_ln5
Hi_ln6 = Dat_ln6
Hi_ln7 = Dat_ln7
Hi_ln8 = Dat_ln8

Cls
Locate 1 , 12
Lcd "<OKE>"
 Locate 2 , 1
  Lcd " Cari Hitam "
  Waitms 300
  Do
       Gosub Baca_adc
       Loop Until Sw_d = 0
       Lo_ln1 = Dat_ln1
       Lo_ln2 = Dat_ln2
       Lo_ln3 = Dat_ln3
       Lo_ln4 = Dat_ln4
```

```
      Lo_ln5 = Dat_ln5
      Lo_ln6 = Dat_ln6
      Lo_ln7 = Dat_ln7
      Lo_ln8 = Dat_ln8

 Ref_ln1 = Hi_ln1 + Lo_ln1
 Ref_ln2 = Hi_ln2 + Lo_ln2
 Ref_ln3 = Hi_ln3 + Lo_ln3
 Ref_ln4 = Hi_ln4 + Lo_ln4
 Ref_ln5 = Hi_ln5 + Lo_ln5
 Ref_ln6 = Hi_ln6 + Lo_ln6
 Ref_ln7 = Hi_ln7 + Lo_ln7
 Ref_ln8 = Hi_ln8 + Lo_ln8

 Ref_ln1 = Ref_ln1 / 2
 Ref_ln2 = Ref_ln2 / 2
 Ref_ln3 = Ref_ln3 / 2
 Ref_ln4 = Ref_ln4 / 2
 Ref_ln5 = Ref_ln5 / 2
 Ref_ln6 = Ref_ln6 / 2
 Ref_ln7 = Ref_ln7 / 2
 Ref_ln8 = Ref_ln8 / 2

 Ep_ln1 = Ref_ln1
 Ep_ln2 = Ref_ln2
 Ep_ln3 = Ref_ln3
 Ep_ln4 = Ref_ln4
 Ep_ln5 = Ref_ln5
 Ep_ln6 = Ref_ln6
 Ep_ln7 = Ref_ln7
 Ep_ln8 = Ref_ln8

 Lihat_sensor:
 Cls
 Lcd "Putih.?   Hitam.?"
 Locate 2 , 1
 Lcd "  Warna Garis ?"
 Waitms 300
 Do
 If Sw_a = 0 Then
H_t = 0
    Ep_simpang = Set_simpang
    Goto Cek_garis
 End If
 If Sw_d = 0 Then
H_t = 1
    Ep_simpang = Set_simpang
    Goto Cek_garis
 End If
 Loop
 Cek_garis:

Cls
  Waitms 300
  Locate 1 , 5
  Lcd "|"
  Locate 1 , 12
```

```
  Lcd "|"
  Locate 2 , 1
  Lcd "Percobaan Garis"
  Do
  Gosub Conversi
  'Locate 1 , 5
  'If Ln7 = 0 Then Lcd Chr(0)
  'If Ln7 = 1 Then Lcd " "
Locate 1 , 6
If Ln6 = 0 Then Lcd Chr(0)
If Ln6 = 1 Then Lcd " "
Locate 1 , 7
If Ln5 = 0 Then Lcd Chr(0)
If Ln5 = 1 Then Lcd " "
Locate 1 , 8
If Ln4 = 0 Then Lcd Chr(0)
If Ln4 = 1 Then Lcd " "
Locate 1 , 9
If Ln3 = 0 Then Lcd Chr(0)
If Ln3 = 1 Then Lcd " "
Locate 1 , 10
If Ln2 = 0 Then Lcd Chr(0)
If Ln2 = 1 Then Lcd " "
Locate 1 , 11
If Ln1 = 0 Then Lcd Chr(0)
If Ln1 = 1 Then Lcd " "
'Locate 1 , 12
'If Ln0 = 0 Then Lcd Chr(0)
'If Ln0 = 1 Then Lcd " "

Loop Until Sw_a = 0 Or Sw_b = 0 Or Sw_c = 0 Or Sw_d = 0
Goto Credit
Set_slese:
Cls
Locate 2 , 1
Lcd "SETTING SELESAI..."
Waitms 400
Cls
Goto Awal

'# Program Selessai
==========================================================
```

SRF04 - Ultra-Sonic Ranger

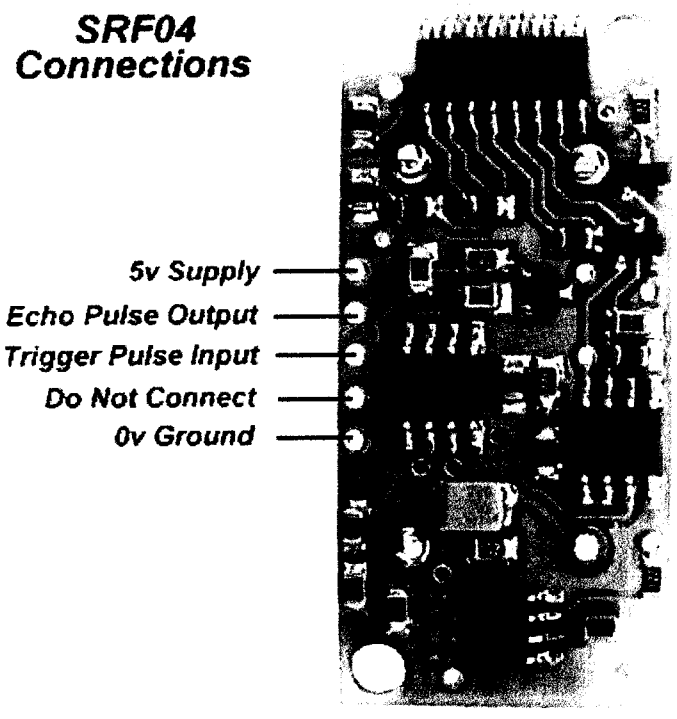
Technical Specification



This project started after I looked at the Polaroid Ultrasonic Ranging module. It has a number of disadvantages for use in small robots etc.

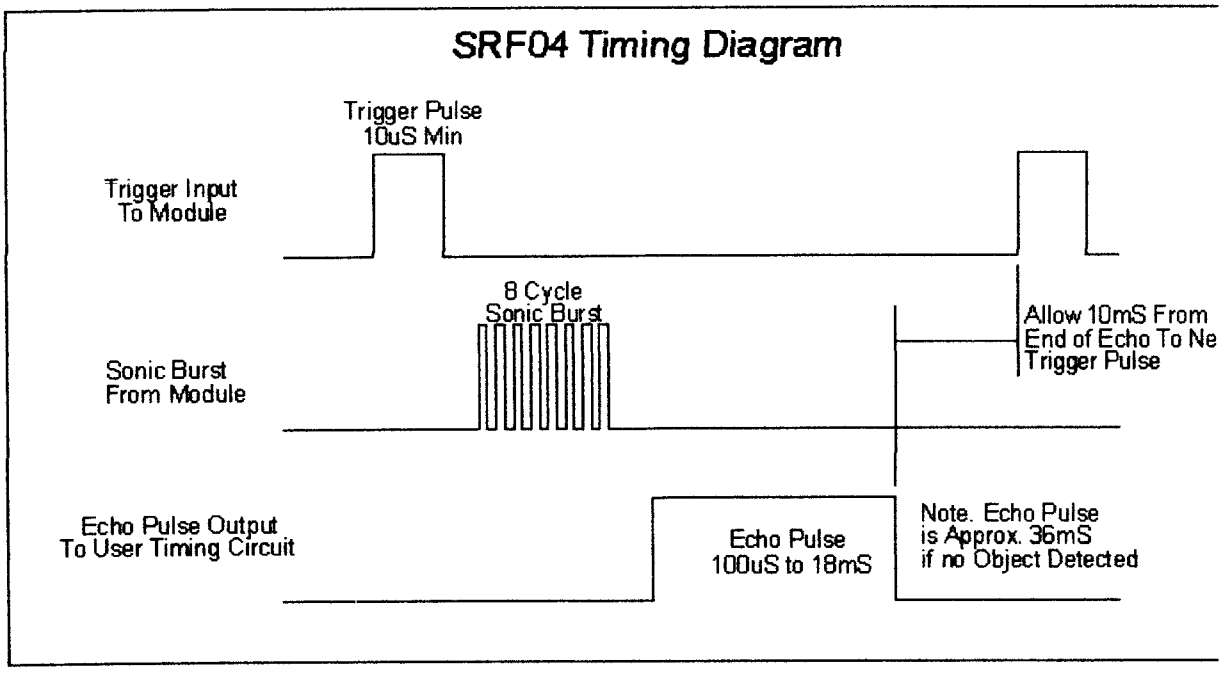
1. The maximum range of 10.7 metre is far more than is normally required, and as a result
2. The current consumption, at 2.5 Amps during the sonic burst is truly horrendous.
3. The 150mA quiescent current is also far too high.
4. The minimum range of 26cm is useless. 1-2cm is more like it.
5. The module is quite large to fit into small systems, and
6. It's EXPENSIVE.

The SRF04 was designed to be just as easy to use as the Polaroid sonar, requiring a short trigger pulse and providing an echo pulse. Your controller only has to time the length of this pulse to find the range. The connections to the SRF04 are shown below:

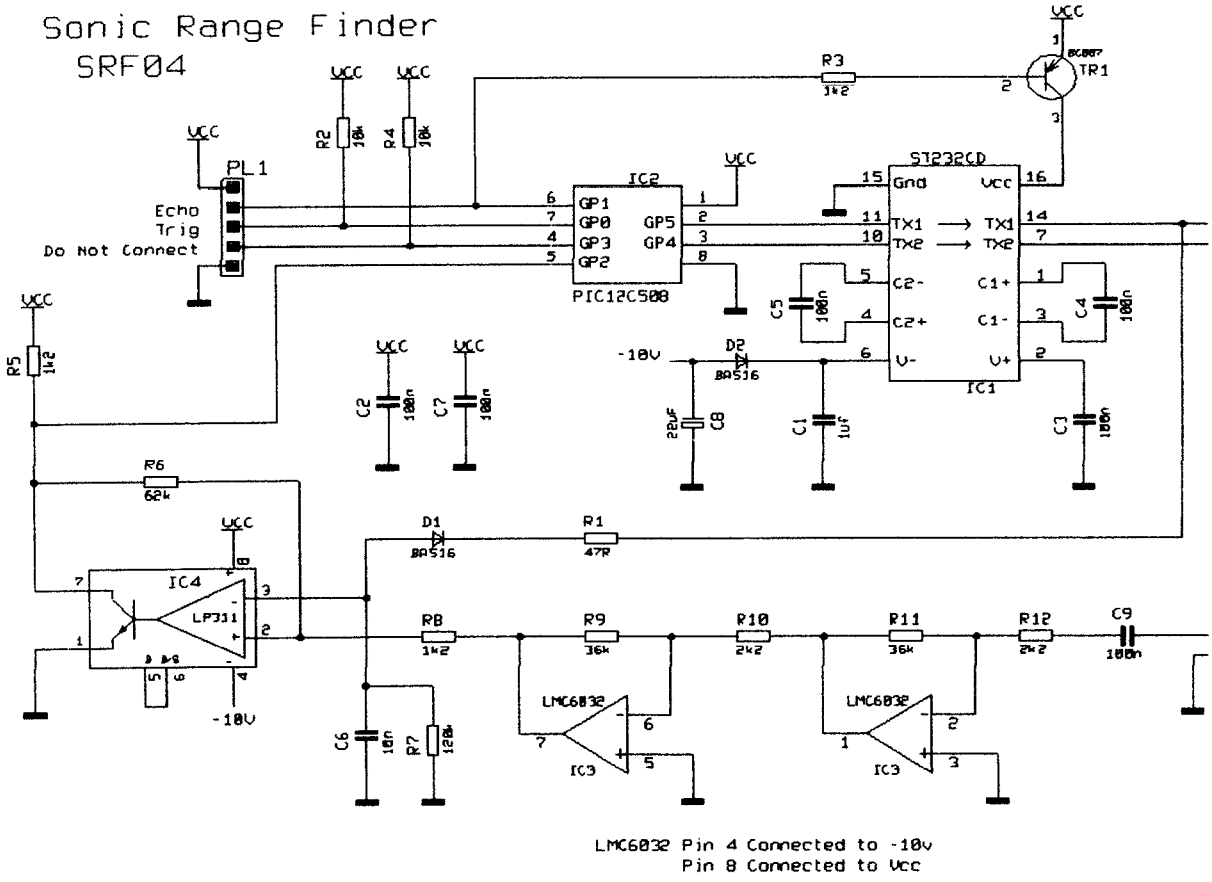


The SRF04 Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging. The SRF04 will send out an 8 cycle burst of ultrasound at 40khz and raise its echo line high. It then listens for an echo, and as soon as it detects one it lowers the echo line again. The echo line is therefore a pulse whose width is proportional to the distance to the object. By timing the pulse it is possible to calculate the range in inches/centimeters or anything else.

If nothing is detected then the SRF04 will lower its echo line anyway after about 36mS.



Here is the schematic, You can download a better quality pdf (161k) version srf1.pdf



The circuit is designed to be low cost. It uses a PIC12C508 to perform the control functions and

standard 40khz piezo transducers. The drive to the transmitting transducer could be simplest driven directly from the PIC. The 5v drive can give a useful range for large objects, but can be problematic detecting smaller objects. The transducer can handle 20v of drive, so I decided to get up close to this level. A MAX232 IC, usually used for RS232 communication makes and ideal driver, providing about 16v of drive.

The receiver is a classic two stage op-amp circuit. The input capacitor C8 blocks some residual DC which always seems to be present. Each gain stage is set to 24 for a total gain of 576-ish. This is close the 25 maximum gain available using the LM1458. The gain bandwidth product for the LM1458 is 1Mhz. The maximum gain at 40khz is 1000000/40000 = 25. The output of the amplifier is fed into an LM311 comparator. A small amount of positive feedback provides some hysterisis to give a clean stable output.

The problem of getting operation down to 1-2cm is that the receiver will pick up direct coupling from the transmitter, which is right next to it. To make matters worse the piezo transducer is a mechanical object that keeps resonating some time after the drive has been removed. Up to 1mS depending on when you decide it has stopped. It is much harder to tell the difference between this direct coupled ringing and a returning echo, which is why many designs, including the Polaroid module, simply blank out this period. Looking at the returning echo on an oscilloscope shows that it is much larger in magnitude at close quarters than the cross-coupled signal. I therefore adjust the detection threshold during this time so that only the echo is detectable. The 100n capacitor C10 is charged to about –6v during the burst. This discharges quite quickly through the 10k resistor R6 to restore sensitivity for more distant echo's.

A convenient negative voltage for the op-amp and comparator is generated by the MAX232. Unfortunately, this also generates quite a bit of high frequency noise. I therefore shut it down whilst listening for the echo. The 10uF capacitor C9 holds the negative rail just long enough to do this.

In operation, the processor waits for an active low trigger pulse to come in. It then generates just eight cycles of 40khz. The echo line is then raised to signal the host processor to start timing. The raising of the echo line also shuts of the MAX232. After a while – no more than 10-12mS normally, the returning echo will be detected and the PIC will lower the echo line. The width of this pulse represents the flight time of the sonic burst. If no echo is detected then it will automatically time out after about 30mS (Its two times the WDT period of the PIC). Because the MAX232 is shut down during echo detection, you must wait at least 10mS between measurement cycles for the +/- 10v to recharge.

Performance of this design is, I think, quite good. It will reliably measure down to 3cm and will continue detecting down to 1cm or less but after 2-3cm the pulse width doesn't get any smaller.

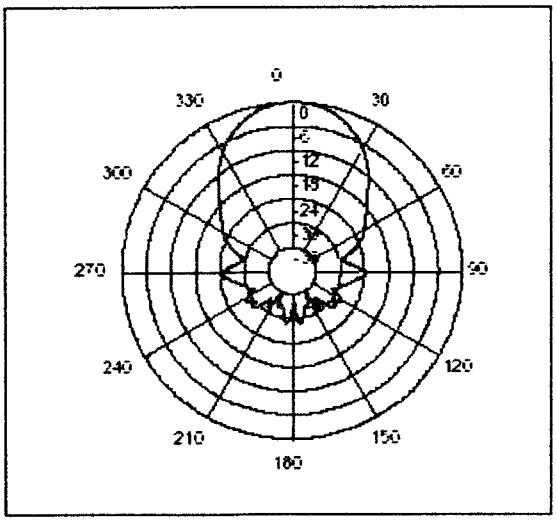
Maximum range is a little over 3m. As and example of the sensitivity of this design, it will detect a 1inch thick plastic broom handle at 2.4m.
Average current consumption is reasonable at less than 50mA and typically about 30mA.

Download the source code and a ready assembled hex file.

Calculating the Distance
The SRF04 provides an echo pulse proportional to distance. If the width of the pulse is measured in uS, then dividing by 58 will give you the distance in cm, or dividing by 148 will give the distance in inches. uS/58=cm or uS/148=inches.

Changing beam pattern and beam width
You can't! This is a question which crops up regularly, however there is no easy way to reduce or change the beam width that I'm aware of. The beam pattern of the SRF04 is conical with the width of

the beam being a function of the surface area of the transducers and is fixed. The beam pattern of the transducers used on the SRF04, taken from the manufacturers data sheet, is shown below.



There is more information in the sonar faq.

Update - May 2003
Since the original design of the SRF04 was published, there have been incremental improvements to improve performance and manufacturing reliability. The op-amp is now an LMC6032 and the comparator is an LP311. The 10uF capacitor is now 22uF and a few resistor values have been tweaked. These changes have happened over a period of time.

All SRF04's manufactured after May 2003 have new software implementing an optional timing control input using the "do not connect" pin. This connection is the PIC's Vpp line used to program the chip after assembly. After programming its just an unused input with a pull-up resistor. When left unconnected the SRF04 behaves exactly as it always has and is described above. When the "do not connect" pin is connected to ground (0v), the timing is changed slightly to allow the SRF04 to work with the slower controllers such as the Picaxe. The SRF04's "do not connect" pin now acts as a timing control. **This pin is pulled high by default and when left unconnected, the timing remains exactly as before.** With the timing pin pulled low (grounded) a 300uS delay is added between the end of the trigger pulse and transmitting the sonic burst. Since the echo output is not raised until the burst is completed, there is no change to the range timing, but the 300uS delay gives the Picaxe time to sort out which pin to look at and start doing so. The new code has shipped in all SRF04's since the end of April 2003. The new code is also useful when connecting the SRF04 to the slower Stamps such as the BS2. Although the SRF04 works with the BS2, the echo line needs to be connected to the lower numbered input pins. This is because the Stamps take progressively longer to look at the higher numbered pins and can miss the rising edge of the echo signal. In this case you can connect the "do not connect" pin to ground and give it an extra 300uS to get there.

:ures

ı-performance, Low-power A'/R® 8-bit Microcontroller
anced RISC Architecture
131 Powerful Instructions – Most Single-clock Cycle Execution
32 x 8 General Purpose Working Registers
Fully Static Operation
Up to 16 MIPS Throughput at 16 MHz
On-chip 2-cycle Multiplier
ıvolatile Program and Data Memories
32K Bytes of In-System Self-Programmable Flash
Endurance: 10,000 Write/Erase Cycles
Optional Boot Code Section with Independent Lock Bits
In-System Programming by On-chip Boot Program
True Read-While-Write Operation
1024 Bytes EEPROM
Endurance: 100,000 Write/Erase Cycles
2K Byte Internal SRAM
Programming Lock for Software Security
.G (IEEE std. 1149.1 Compliant) Interface
Boundary-scan Capabilities According to the JTAG Standard
Extensive On-chip Debug Support
Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
ipheral Features
Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
Real Time Counter with Separate Oscillator
Four PWM Channels
8-channel, 10-bit ADC
8 Single-ended Channels
7 Differential Channels in TQFP Package Only
2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
Byte-oriented Two-wire Serial Interface
Programmable Serial USART
Master/Slave SPI Serial Interface
Programmable Watchdog Timer with Separate On-chip Oscillator
On-chip Analog Comparator
ecial Microcontroller Features
Power-on Reset and Programmable Brown-out Detection
Internal Calibrated RC Oscillator
External and Internal Interrupt Sources
Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
and Packages
32 Programmable I/O Lines
40-pin PDIP, 44-lead TQFP, and 44-pad MLF
erating Voltages
2.7 - 5.5V for ATmega32L
4.5 - 5.5V for ATmega32
eed Grades
0 - 8 MHz for ATmega32L
0 - 16 MHz for ATmega32
wer Consumption at 1 MHz, 3V, 25°C for ATmega32L
Active: 1.1 mA
Idle Mode: 0.35 mA
Power-down Mode: < 1 µA

8-bit AVR® Microcontroller with 32K Bytes In-System Programmable Flash

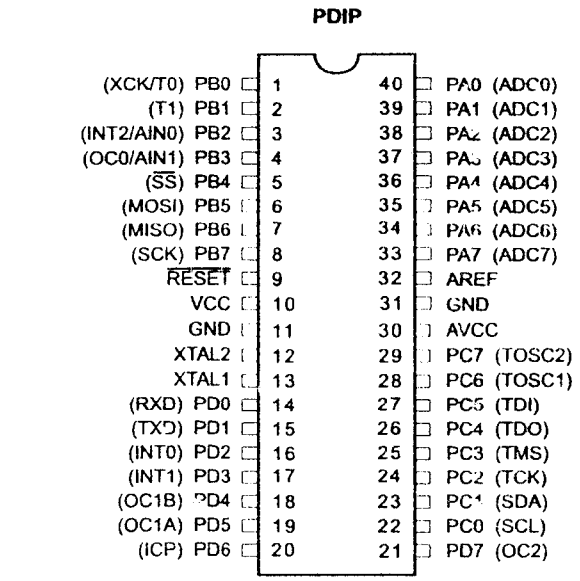
ATmega32
ATmega32L

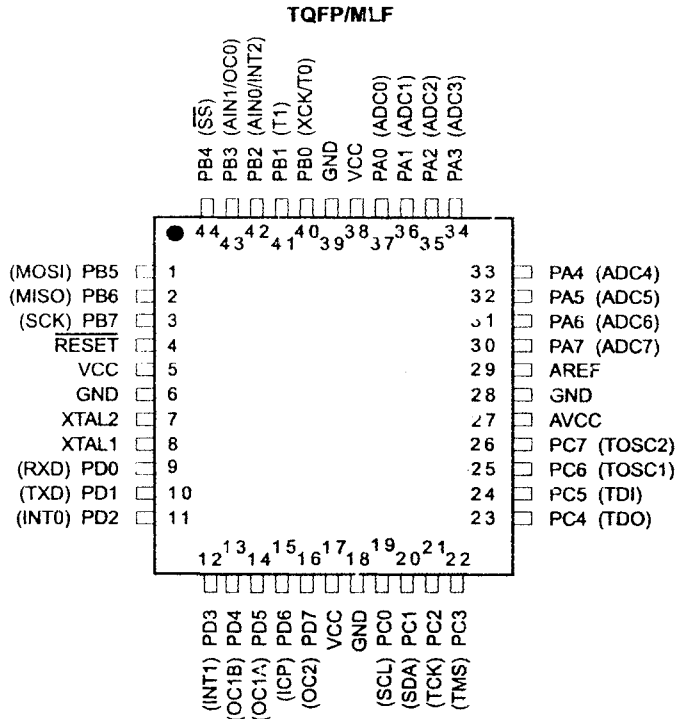
Preliminary

2503F–AVR–12/03

Configurations

Figure 1. Pinouts ATmega32





claimer

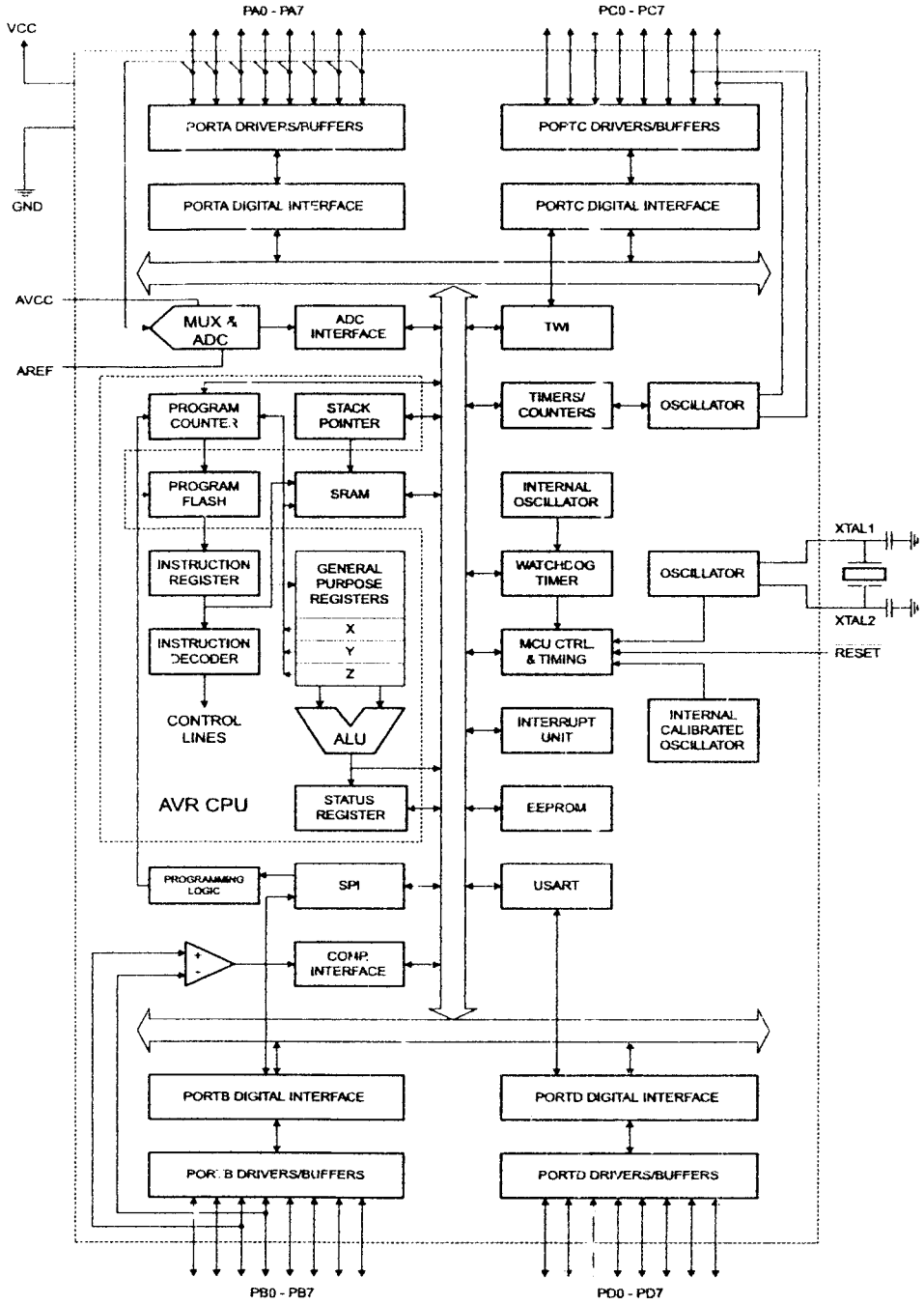
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

rview

The ATmega32 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega32 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

k Diagram

Figure 2. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega32 provides the following features: 32K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 1024 bytes EEPROM, 2K byte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega32 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega32 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Descriptions

Digital supply voltage

Ground.

A (PA7..PA0)

Port A serves as the analog inputs to the A/D Converter

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

(PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega32 as listed on page 55.

(PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

The TD0 pin is tri-stated unless TAP states that shift out data are entered.

Port C also serves the functions of the JTAG interface and other special features of the ATmega32 as listed on page 58.

(PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega32 as listed on page 60.

$\overline{RESET}$

Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 35. Shorter pulses are not guaranteed to generate a reset.

XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting Oscillator amplifier.

AVCC

AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

AREF

AREF is the analog reference pin for the A/D Converter.

ister Summary

ress	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$5F)	SREG	I	T	H	S	V	N	Z	C	8
$5E)	SPH	–	–	–	–	SP11	SP10	SP9	SP8	10
$5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	10
$5C)	OCR0	Timer/Counter0 Output Compare Register								80
$5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	45, 65
$5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	66
$59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	80, 110, 128
$58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	81, 111, 128
$57)	SPMCR	SPMIE	RWWSB		RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	240
$56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	175
($55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	30, 64
($54)	MCUCSR	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	38, 65, 226
($53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	78
($52)	TCNT0	Timer/Counter0 (8 Bits)								80
($51)[1]	OSCCAL	Oscillator Calibration Register								28
	OCDR	On-Chip Debug Register								222
($50)	SFIOR	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	54,83,129,196,216
($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	105
($4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	108
($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								109
($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								109
($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								109
($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								109
($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								109
($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								109
($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								110
($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								110
($45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	123
($44)	TCNT2	Timer/Counter2 (8 Bits)								125
($43)	OCR2	Timer/Counter2 Output Compare Register								125
($42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	126
($41)	WDTCR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	40
($40)[2]	UBRRH	URSEL	–	–	–	UBRR[11:8]				162
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	160
($3F)	EEARH	–	–	–	–	–	–	EEAR9	EEAR8	17
($3E)	EEARL	EEPROM Address Register Low Byte								17
($3D)	EEDR	EEPROM Data Register								17
($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	17
($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	62
($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	62
($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	62
($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	62
($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	62
($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	63
($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	63
($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	63
($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	63
($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	63
($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	63
($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	63
($2F)	SPDR	SPI Data Register								136
($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	136
($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	134
($2C)	UDR	USART I/O Data Register								157
($2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	158
($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	159
($29)	UBRRL	USART Baud Rate Register Low Byte								162
($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	197
($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	212
($26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	214
($25)	ADCH	ADC Data Register High Byte								215
($24)	ADCL	ADC Data Register Low Byte								215
($23)	TWDR	Two-wire Serial Interface Data Register								177
($22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	177

ress	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
($21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	176
($20)	TWBR	Two-wire Serial Interface Bit Rate Register								175

1. When the OCDEN Fuse is unprogrammed, the OSCCAL Register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCDR Register.
2. Refer to the USART description for details on how to access UBRRH and UCSRC.
3. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
4. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

uction Set Summary

nics	Operands	Description	Operation	Flags	#Clocks
TIC AND LOGIC INSTRUCTIONS					
	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V	1
	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,H	1
	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
	Rd	Set Register	Rd ← $FF	None	1
	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
U	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
CH INSTRUCTIONS					
	k	Relative Jump	PC ← PC + k + 1	None	2
		Indirect Jump to (Z)	PC ← Z	None	2
	k	Direct Jump	PC ← k	None	3
	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
		Indirect Call to (Z)	PC ← Z	None	3
	k	Direct Subroutine Call	PC ← k	None	4
		Subroutine Return	PC ← Stack	None	4
		Interrupt Return	PC ← Stack	I	4
	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1 / 2
	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2

onics	Operands	Description	Operation	Flags	#Clocks
	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
RANSFER INSTRUCTIONS					
	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
	Rd, K	Load Immediate	Rd ← K	None	1
	Rd, X	Load Indirect	Rd ← (X)	None	2
	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
	Rd, Y	Load Indirect	Rd ← (Y)	None	2
	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
	Rd, Z	Load Indirect	Rd ← (Z)	None	2
	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
	X, Rr	Store Indirect	(X) ← Rr	None	2
	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
	Y, Rr	Store Indirect	(Y) ← Rr	None	2
	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
	Z, Rr	Store Indirect	(Z) ← Rr	None	2
	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
		Load Program Memory	R0 ← (Z)	None	3
	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
		Store Program Memory	(Z) ← R1:R0	None	-
	Rd, P	In Port	Rd ← P	None	1
	P, Rr	Out Port	P ← Rr	None	1
	Rr	Push Register on Stack	Stack ← Rr	None	2
	Rd	Pop Register from Stack	Rd ← Stack	None	2
ND BIT-TEST INSTRUCTIONS					
	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
	s	Flag Set	SREG(s) ← 1	SREG(s)	1
	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
		Set Carry	C ← 1	C	1
		Clear Carry	C ← 0	C	1
		Set Negative Flag	N ← 1	N	1
		Clear Negative Flag	N ← 0	N	1
		Set Zero Flag	Z ← 1	Z	1
		Clear Zero Flag	Z ← 0	Z	1
		Global Interrupt Enable	I ← 1	I	1
		Global Interrupt Disable	I ← 0	I	1
		Set Signed Test Flag	S ← 1	S	1
		Clear Signed Test Flag	S ← 0	S	1
		Set Twos Complement Overflow.	V ← 1	V	1
		Clear Twos Complement Overflow	V ← 0	V	1
		Set T in SREG	T ← 1	T	1
		Clear T in SREG	T ← 0	T	1
		Set Half Carry Flag in SREG	H ← 1	H	1