PERANCANGAN DAN ANALISIS PERBANDINGAN POSISI SENSOR GARIS PADA ROBOT *MANAGEMENT* SAMPAH

SKRIPSI

KONSENTRASI TEKNIK ELEKTRONIKA

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
BAMBANG DWI PRAKOSO
NIM. 0910633036-63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2014

PENGANTAR

Bismillahirrohmanirrohim.

Puji dan syukur kepada Allah SWT atas segala petunjuk serta nikmat-Nya, sehingga penulis dapat menyelesaikan skripsi ini.

Skripsi berjudul "Perancangan Dan Analisis Perbandingan Posisi Sensor Garis Pada Robot *Management* Sampah" ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

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

- M. Aziz Muslim, ST., MT., Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Hadi Suyono, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Dr. Ir. Sholeh Hadi Pramono, M.Sc. dan Eka Maulana, ST.,MT.,MEng. sebagai Dosen Pembimbing 1 dan Dosen Pembimbing 2 atas segala bimbingan, pengarahan, ide, saran, motivasi, dan masukan yang diberikan,
- Ir. Nurussa'adah, MT. selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya
- Bapak Ris Budiarto dan (almh.) Ibu Endah Mayangkarti dan kakak Agung Prakoso tercinta atas segala nasehat, kasih sayang, perhatian dan kesabarannya didalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Seluruh dosen pengajar Teknik Elektro Universitas Brawijaya,
- Staff Recording Jurusan Teknik Elektro,
- Rekan - rekan TEUB angkatan 2009,
- Rekan – rekan Tim Robot *Management* Sampah Imam, Yudi, Deaz, Ebay, terima kasih atas segala bantuan yang telah diberikan,

- Seluruh teman-teman serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan satu-persatu, terimakasih banyak atas bantuan dan dukungannya.

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

Malang, Agustus 2014

Penulis

ABSTRAK

Bambang Dwi Prakoso, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Juli 2014, Perancangan Dan Analisis Perbandingan Posisi Sensor Garis Pada Robot *Management* Sampah, Dr.Ir. Sholeh Hadi Pramono, M.Sc. dan Eka Maulana, ST., MT., M.Eng.

Abstrak - Robot *management* sampah merupakan salah satu wujud pengaplikasian perkembangan dari robot *line follower*. Salah satu bagian terpenting dari robot *management* sampah adalah sensor garis.

Sensor garis digunakan untuk membaca lintasan yang berupa sebuah garis hitam dengan alas putih. Tujuan dari perancangan dan analisis posisi sensor garis pada robot management sampah adalah untuk membandingkan performansi robot tersebut. Sebelum melakukan analisis perbandingan sensor garis hal yang pertama dilakukan adalah melakukan pengujian dengan desain sensor garis yang berbeda-beda, kemudian untuk pengujian analisis diperlukan robot *management* sampah, sensor garis,dan lintasan. Pada skripsi ini digunakan 3 desain perancangan yaitu bentuk lurus, segitiga, dan setengah lingkaran. Dari ketiga analisis desain tersebut dihasilkan hasil perbandingan performansi sensor yang berbeda-beda.

Pengujian dilakukan pada lintasan garis lurus. Pengujian pertama adalah pengujian sensor bentuk lurus (180^0) dapat mencapai *set point* pada pencuplikan data ke- 39, menghasilkan s*ettling time* sebesar 7,8 s dan *overshoot* maksimum sebesar 28,57%. Pengujian kedua adalah sensor bentuk segitiga dapat mencapai *set point* pada pencuplikan data ke-33, menghasilkan *settling time* sebesar 6,6 s dan *overshoot* maksimum sebesar 28,57%. Yang terakhir adalah bentuk sensor setengah lingkaran dapat mencapai set point pada pencuplikan data ke-63, tidak dapat mencapai *set point* dan terjadi osilasi.

Kata Kunci : Sensor Garis, Sensor Photodioda, Robot Management Sampah, *Line Follower*

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Mobile robot merupakan konstruksi robot yang paling populer. *Mobile robot* dapat dibuat sebagai pengikut garis *(line follower)*, aplikasi dari *line follower* biasanya digunakan sebagai sarana transportasi di area industri dan perlombaan robot *line tracer* untuk mahasiswa. Robot *management* sampah yang merupakan salah satu jenis robot beroda yang bergerak berdasarkan jalur garis *(line follower)* yang sudah ditentukan dengan daya penggerak berupa motor DC. Robot *line follower* merupakan *platform* serbaguna yang memungkinkan atas perkembangan yang terjadi dari berbagai aplikasi. Dalam struktur modern, kita dapat dengan mudah memodifikasi dan menambahkan dari segi fungsinya. Dari segi harga, robot *line follower* adalah pilihan tepat untuk pengembangan lebih lanjut. (Carrick M., 2006)

Hasil penelitian menunjukkan bahwa rancang-bangun robot *line follower* meliputi tiga hal yaitu rancang-bangun sistem mekanik, sistem *hardware* elektronika, dan sistem *software*. Sistem mekanik robot *line follower* meliputi rangka robot dan penggerak robot berupa roda robot. *Hardware* elektronika meliputi rangkaian sensor garis, pengkondisi sinyal, mikrokontroler, *driver* motor DC, motor DC, LED indikator, dan catu daya (baterai). Sedangkan rancang bangun *software* meliputi program *assembly* mikrokontroler yang menunjukkan alur kerja robot sesuai dengan *flowchart* yang telah dirancang. (Heri S., 2002 : 1)

Unjuk kerja robot *line follower* menunjukkan proses kerja robot untuk mengikuti garis. Kerja robot *line follower* meliputi aspek *hardware* dan aspek *software*. Secara *hardware* unjuk kerja robot dapat diketahui dari ketepatan sensor dalam mendeteksi garis berwarna hitam dan ketepatan sensor dalam mendeteksi warna putih. Secara *software* unjuk kerja robot pengikut garis dapat diketahui dari *flowchart* yang menunjukkan unjuk kerja sebuah robot *line follower*. (Heri S., 2002 : 2)

Kestabilan robot *management* sampah bisa dipengaruhi oleh variasi desain sensor garis. Selama ini desain sensor garis yang digunakan cenderung berbentuk lurus (180^0), kelebihan desain bentuk lurus adalah perancangan yang tidak terlalu rumit, sedangkan kekurangan desain sensor garis bentuk lurus adalah kurang cepat dalam pencapaian *set point*. Namun belum ada penelitian apabila bentuk sensor garis yang bervariasi. Oleh karena itu penulis ingin membandingkan performansi robot dengan bentuk sensor yang bervariasi, yaitu dengan mendesain sensor garis dengan desain bentuk segitiga bentuk lurus (180°) dan bentuk setengah lingkaran. Diharapkan dengan variasi sensor garis yang baru bisa mendapatkan performansi sensor terbaik yang akan di aplikasikan pada robot *line follower*.

1.2 Rumusan Masalah

Berdasarkan uraian dalam latar belakang, dapat disusun rumusan masalah sebagai berikut :

1) Bagaimana merancang dan membuat sensor garis dengan bentuk segitiga, setengah lingkaran, dan lurus (180^0) ?
2) Bagaimana membandingkan performansi masing-masing desain sensor terhadap pergerakan robot *management* sampah ?

1.3 Ruang Lingkup

Dalam perancangan untuk skripsi ini permasalahan dibatasi oleh hal-hal sebagai berikut :

1) Pembahasan ditekankan pada bentuk sensor garis dan analisis perbandingannya.
2) Sensor yang digunakan adalah *photodiode*.
3) Aktuator yang digunakan adalah motor DC NISCA 12 volt dengan kecepatan 1930 rpm.
4) Pemrograman mikrokontroler (*slave*) menggunakan *software CodeVisionAVR C Compiler* untuk pengolahan data.

5) Mikrokontroler yang digunakan yaitu 3 buah Atmega8 (slave) dan 1 buah mikrokontroler *master* Atmega16.
6) Dimensi robot yaitu panjang 100 cm,lebar 40 cm dan tinggi 45 cm.

1.4 Tujuan

Tujuan dari penelitian ini adalah mendesain sensor garis dengan bentuk segitiga, setengah lingkaran, dan bentuk lurus (180^0) serta melakukan analisis perbandingan performansi sersor hasil desain terhadap pergerakan robot *management* sampah.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini adalah BAB I berisi pendahuuan. Pendahuluan latar belakang penelitian, rumusan masalah, ruang lingkup, tujuan penelitian dan sistematika penulisan. BAB II berisi dasar teori. Dasar teori membahas tentang teori-teori yang mendukung dan dapat digunakan untuk membantu menyelesaikan penelitian. BAB III berisi metode penelitian. Metode penelitian membahas tentang literatur yang digunakan, penentuan spesifikasi alat, mekanisme perancangan alat, mekanisme pengujian dan analisis, mekanisme pengambilan kesimpulan dan saran. BAB IV berisi perancangan alat. Perancangan dan pembuatan alat membahas tentang proses perancangan dan pembuatan alat yang meliputi spesifikasi, perancangan blok diagram, dan realisasi alat. BAB V berisis pengujian dan analisis alat. Pengujian dan analisis alat membahas tentang cara pengujian dan hasil pengujian alat yang telah direalisasikan. BAB IV berisi kesimpulan dan saran. Kesimpulan dan saran berisi tentang kesimpulan berdasarkan hasil yang telah diperoleh dari tujuan, perancangan dan pengujian. Selain itu terdapat juga saran yang dapat dilakukan untuk pengembangan lebih lanjut.

BAB II
DASAR TEORI

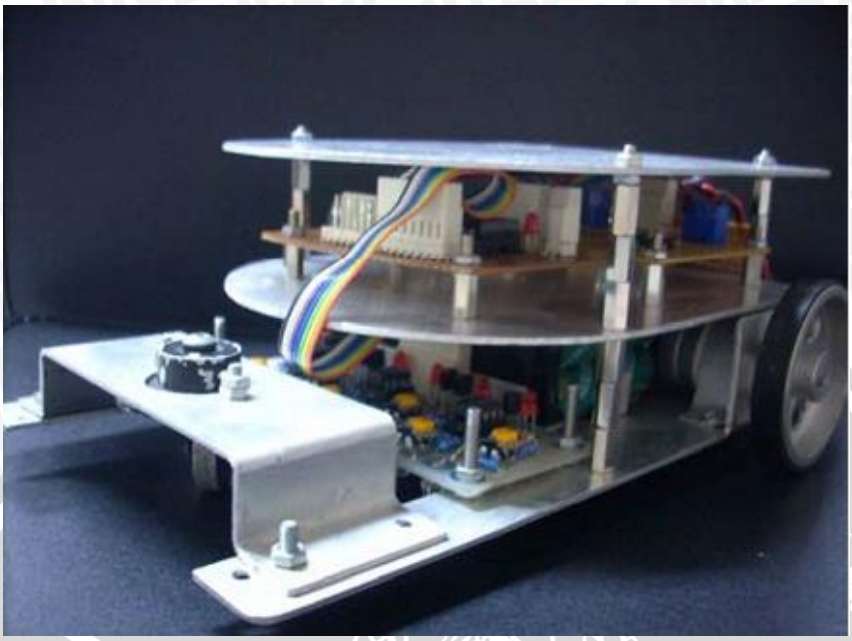
Untuk memudahkan dalam memahami cara kerja rangkaian maupun dasar-dasar perencaan alat ini, maka perlu adanya penjelasan dan uraian teori penunjang yang digunakan dalam penulisan ini. Teori penunjang yang akan dijelaskan dalam bab ini adalah :

- *Line Follower*
- Mikrokontroller ATMega 8
- LED
- Sensor *Photodiode*
- Multiplekser
- LCD

2.1 Line Follower

Robot *line follower* atau robot pengikut garis adalah suatu jenis robot yang pergerakannya dengan mendeteksi garis sehingga robot tersebut dapat melakukan perpindahan posisi dari satu titik ke titik yang lain. Jika posisi robot menyimpang dari garis, maka robot akan melambat. Namun jika robot tepat berada diatas garis, maka robot akan bergerak cepat. Robot juga dapat kembali ke garis pada saat robot terlepas sama sekali dari garis. Hal ini bisa dilakukan karena robot selalu mengingat kondisi terakhir pembacaan sensor. Jika terakhir kondisinya adalah disebelah kiri garis, maka robot akan bergerak ke kanan, demikian pula sebaliknya.

Robot pengikut garis yang digunakan dalam penelitian ini adalah robot beroda yang memiliki dua motor penggerak berkarakteristik sama. Untuk itu diperlukan suatu sistem kendali yang dapat mengolah data yang dimasukkan dalam sistem kontrolernya menjadi data keluaran yang digunakan untuk mengatur gerak robot. Kontrol untuk mengatur kecepatan dari masing-masing motor penggerak sehingga dihasilkan gerakan obot yang mudah diatur menggunakan metode PID. (Bagus I. S., 2013)



Gambar 2.1 Robot *line follower*
Sumber: Heri, S., 2002

2.2 Mikrokontroler

2.2.1 ATmega8

Secara umum, Mikrokontroler adalah suatu IC (*integrated circuit)* yang memiliki kemampuan untuk membuat keputusan berdasarkan sinyal dari luar (input), dan berdasarkan algoritma yang dibentuk menjadi suatu program tertentu. Pada dasarnya mikrokontroler memiliki mikroprosesor, *timer*, *counter*, perangkat I/O dan internal memori.

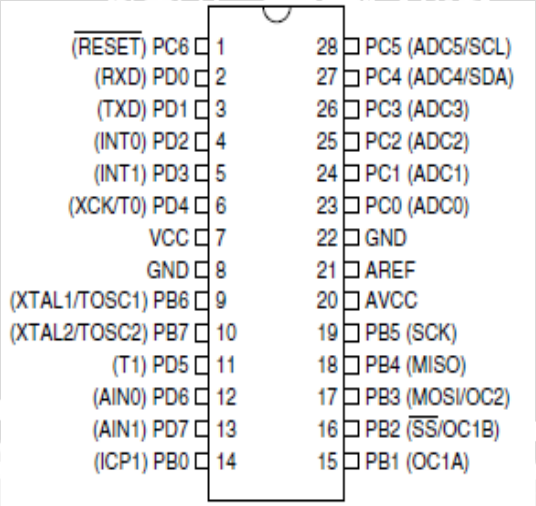
ATmega8 menurut datasheet adalah mikrokontroler CMOS 8-bit berarsitektur AVR RISC yang memiliki 8K Bytes *ISP Flash*. Dengan mengeksekusi instruksi dalam clock yang sangat cepat, ATmega8 memiliki throughput mendekati 1 MIPS per MHZ, yang memungkinkan perancang sistem mengoptimalkan konsumsi daya terhadap kecepatan pemrosesan.

Sebagai suatu sistem kontrol mikrokontroler ATmega8 bila dibandingkan dengan mikroprosesor memiliki kemampuan dan segi ekonomis yang bisa diandalkan karena dalam mikrokontroler sudah terdapat RAM dan ROM sedangkan

mikroprosesor di dalamnya tidak terdapat keduanya. Secara umum konfigurasi yang dimiliki mikrokontroler ATmega8 adalah sebagai berikut :

- Sebuah CPU 8 bit dengan menggunakan teknologi dari Atmel.
- Jalur dua arah (*bidirectional*) yang digunakan sebagai saluran masukan atau keluaran yang dikontrol oleh *register* DDR.
- Dua buah *timer/counter* 8 bit dan sebuah *timer/counter* 16 bit.
- 23 jalur *programable* I/O (input/output).
- *Analog to Digital Converter (ADC)* 10-bit dan *Analog comparator* di dalam chip.
- Osilator internal dan rangkaian pewaktu.
- Sebuah komunikasi serial USART yang dapat diprogram.
- Sebuah *master/slave* serial SPI yang dapat diprogram.
- Sebuah *Two Wire Serial Interface*.

Masing-masing pin mikrokontroler ATmega8 mempunyai fungsi tersendiri. Dengan mengetahui fungsi masing-masing pin mikrokontroler ATmega8, perancangan aplikasi mikrokontroler ATmega8 akan lebih mudah dan maksimal. ATmega8 mempunyai 28 pin, susunan masing-masing pin ditunjukkan dalam Gambar 2.2.



Gambar 2.2.Konfigurasi Pin ATmega8
Sumber: ATMEL, 2011

Fungsi kaki-kaki PIN dalam ATmega8 sesuai dengan *data sheet* ATmega8 antara lain sebagai berikut:

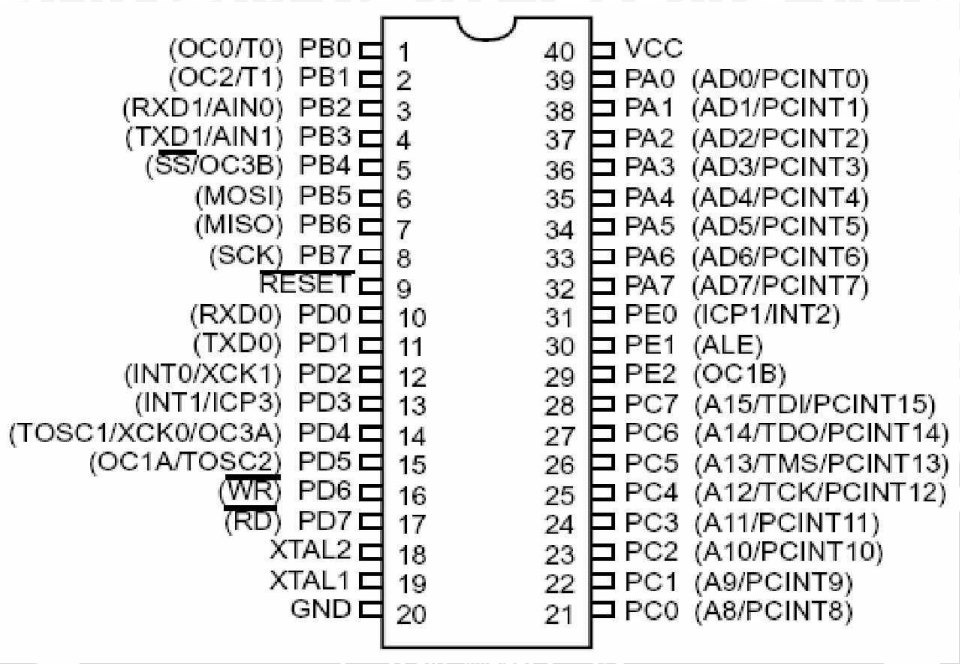
- *Port* B (Pin B0..7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* B antara lain adalah : *Port* B0 {ICP (*Timer/counter input capture pin*)}, *Port* B1 {OC1A (*Timer/counter* 1 *output compare* A *match output*)}, *Port* B2 yang bisa digunakan sebagai SS (SPI *slave select input*) atau OC1B (*Timer/counter* 1 *output compare* B *match output*) , *Port* B3 yang bisa digunakan sebagai MOSI (SPI *bus master output/slave input*) atau OC2 (*timer/counter 2 compare match output*), *Port* B4 {MISO (SPI *bus master input/slave output*)}, *Port* B7 {SCK (SPI *bus serial clock*)}, *Port* B5 {SCK (SPI *bus serial clock*)}, *Port* B6 (XTAL1 & TOSC1 (*Timer Oscilator* pin1)), dan *Port* B7 (XTAL2 & TOSC2 (*Timer Oscilator* pin2)).
- *Port* C (Pin C0..5), merupakan saluran masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* C adalah sebagai ADC (*input* ADC *channel* 0..5), selain itu pada beberapa *Port* memiliki fungsi lain antara lain *Port* C4 (SDA (*Two-Wire serial bus data input/output line*)), dan *Port* C5 (SCL (*Two-Wire serial bus clock line*)).
- *Pin* C6 memiliki fungsi *RESET*, merupakan saluran dua masukan untuk mereset mikrokontroler dengan cara memberi masukan logika rendah.
- *Port* D (Pin D0..7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus dari *Port* D antara lain adalah : *Port* D0 {RXD (USART *input pin*)}, *Port* D1 {TXD (USART *output pin*)}, *Port* D2 {INT0 (*Eksternal interupt 0 input*)}, *Port* D3 {INT1 (*Eksternal interupt 1 input*)}, *Port* D4 (T0 (*timer/counter0 eksternal counter input*) & XCK (USART *eksternal clock input/output*), *Port* D5 (T1 (*timer/counter* eksternel *counter input*)), *Port* D6 (AIN0 (*Analog comparator positive input*))dan *Port* D7 (AIN1 (*Analog comparator negative input*)).
- Pin 7 VCC, merupakan masukan untuk catu daya positif DC sebesar 5 volt.

- Pin 8 GND, merupakan *ground* dari seluruh rangkaian.
- Pin B6 dan Pin B7 (XTAL2 dan XTAL1), merupakan saluran untuk mengatur pewaktuan sistem. Untuk pewaktuan dapat menggunakan pewaktuan internal maupun eksternal.
- Pin 21 AREF, merupakan pin referensi analog untuk masukan ADC.
- Pin 22 GND, merupakan *ground* dari ADC.
- Pin 20 AVCC, merupakan catu untuk perangkat ADC.

2.2.2 ATmega162

ATMega162 merupakan seri mikrokontroler CMOS 8-bit buatan Atmel, berbasis arsitektur RISC (*Reduced Instruction Set Computer*). Hampir semua instruksi dieksekusi dalam satu siklus clock. ATMega162 mempunyai 32 register *general-purpose*, *timer*/*counter* fleksibel dengan mode *compare*, *interrupt* internal dan eksternal, *serial* UART, *programmable Watchdog Timer*, dan *mode power saving*. ATMega162 mempunyai ADC dan PWM internal. ATMega162 juga mempunyai *In-Sistem Programmable Flash on-chip* yang mengijinkan memori program untuk diprogram ulang dalam sistem menggunakan hubungan serial SPI. Atmega162 mempunyai *throughput* mendekati 1 MIPS per MHz membuat disainer sistem dapat mengoptimasi konsumsi daya versus kecepatan proses.

Untuk memaksimalkan performa dan paralelisme, ATMega162 menggunakan arsitektur Harvard (memori dan bus terpisah untuk program dan data). Gambar 2.3 menunjukkan pin-pin pada ATMega162 dengan kemasan 40-pin DIP (*Dual In-Line Package*).



Gambar 2.3.Pin-Pin pada ATMega162 dengan Kemasan 40-Pin DIP (*Dual In-Line Package*)

Sumber: Atmel, 2011: 2

Adapun fungsi dari pin-pin yang terdapat pada Mikrokontroler ATMega162 dijelaskan sebagai berikut :

a) VCC, suplai tegangan digital
b) GND, pin *ground*
c) PORT A (PA0 – PA7). PORT A merupakan port I/O 8-*bit bidirectional* yang dilengkapi dengan resistor *pull-up* internal (dapat dipilih untuk tiap *bit*). Selain sebagai port I/O, PORT A juga mempunyai fungsi lain seperti antarmuka memori eksternal dan pin *change interrupt*.
d) PORT B (PB0 – PB7). PORT B merupakan port I/O 8-*bit bidirectional* dengan resistor *pull-up* internal (dapat dipilih untuk tiap *bit*). Selain itu, PORT B juga mempunyai fungsi lain yaitu *Serial Peripheral Interface* (SPI), *Analog Comparator, input/output Timer/Counter*.
e) PORT C (PC0 – PC7). PORT C merupakan port I/O 8-*bit bidirectional* dengan resistor *pull-up* internal (dapat dipilih untuk tiap *bit*). Selain itu

PORT C juga mempunyai fungsi lain yaitu JTAG, antarmuka memori eksternal, dan pin *change interrupt*.

f) PORT D (PD0 – PD7). PORT D merupakan port I/O 8-*bit bidirectional* dengan resistor *pull-up* internal (dapat dipilih untuk tiap *bit*). Selain itu PORT D juga mempunyai fungsi lain yaitu USART, *interupsi eksternal, strobe memori eksternal, timer/counter.*

g) PORT E (PE0 – PE2) PORT E merupakan port I/O 3-*bit bidirectional* dengan resistor *pull-up* internal (dapat dipilih untuk tiap *bit*). Selain itu PORT E juga mempunyai fungsi lain yaitu *timer/counter, latch enable memori eksternal, interupsi eksternal.*

h) RESET, berfungsi untuk mereset mikrokontroler jika diberikan sinyal *active low* dalam selang waktu tertentu.

i)XTAL1, input ke inverting *oscillator* amplifier dan ke rangkaian detak internal.

j)XTAL2, output dari inverting *oscillator amplifier*.

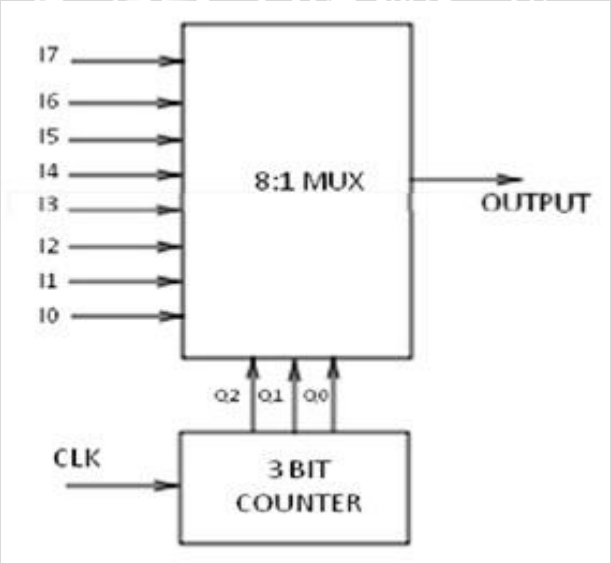
Universal synchronous dan *asynchronous* pemancar dan penerima *serial* adalah suatu alat komunikasi serial sangat fleksibel. Mikrokontroller ATMega162 memiliki dua buah *port* USART untuk komunikasi *serial*, yaitu USART0 dan USART1. Fasilitas komunikasi serial USART mikrokontroler ini memiliki fitur sebagai berikut:

1) *Operasi full duplex* (register penerima dan pengirim *serial* dapat berdiri sendiri)
2) *Operasi Asychronous* atau *synchronous*
3) *Master* atau *slave* mendapat *clock* dengan operasi *synchronous*
4) Pembangkit *baud rate* dengan resolusi tinggi
5) Dukung *frames serial* dengan 5, 6, 7, 8 atau 9 *data bit* dan 1 atau 2 *stop bit*
6) Tahap *odd* atau *even parity* dan *parity check* didukung oleh *hardware*
7) Pendeteksian data *overrun*
8) Pendeteksi *framing error*

9) Pemfilteran gangguan (*noise*) meliputi pendeteksian *bit false start* dan pendeteksian *low pass filter* digital
10) Tiga *interrupt* terdiri atas TX *complete*, TX *data register empty*, dan RX *complete*
11) Mode komunikasi *multi-processor*
12) Mode komunikasi *double speed asynchronous*

2.3 Multiplekser

Multiplekser (MUX) merupakan salah satu rangkaian kombinasional yang paling banyak digunakan dalam desain digital. Multiplekser adalah gerbang pemilih salah satu data dari beberapa masukan untuk satu keluaran. Tergantung pada kode digital yang diterapkan pada pemilihan salah satu masukan dari 2n data masukan yang dipilih dan dikirim ke saluran keluaran tunggal. Multiplekser 8:1 ditunjukkan dalam Gambar 2.4 memiliki 8 masukan 1 keluaran membutuhkan 3 bit counter sinkron untuk pemilihan input.(P. Rajshekar & M. Malathi, 2011)



Gambar 2.4. MUX dengan *Counter Sinkron*
Sumber: P. Rajshekar & M.Malathi, 2011

2.4 LED

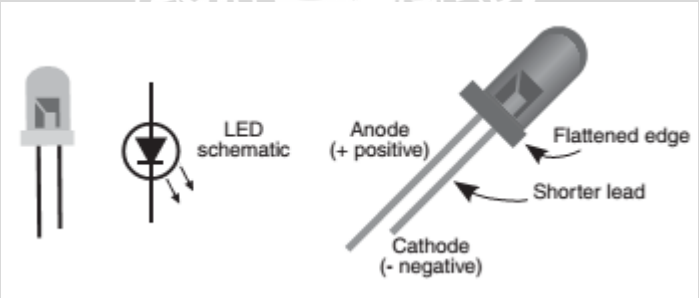
Light Emitting Diode (LED) adalah semikonduktor tipe khusus yang dirancang untuk memancarkan jumlah berlebihan cahaya. Kebanyakan LED direkayasa untuk menghasilkan warna tertentu dari cahaya, dan juga inframerah serta ultraviolet. LED sering digunakan dalam sirkuit tegangan DC daya rendah (<12volt). LED bisa rusak jika arus yang di alirkan melebihi batas maksimalnya. Maka dari itu, LED harus dirangkai dengan resistor untuk membatasi besar arus yang mengalir ke LED.

Banyak macam bentuk dan ukuran LED yang terdapat di pasaran. Yang paling umum adalah silinder dan dibentuk dengan kubah atas. Ukuran yang umum adalah:

- kecil (T1) : berdiameter 3mm
- standart (T1-3/4) : berdiameter 5mm
- Jumbo : berdiameter 10mm

Polaritas LED dapat di amati dari dua kakinya. Kaki konektor LED yang lebih pendek merupakan katoda berpolaritas negatif(-), sedangkan anoda berpolaritas positif (+). Untuk lebih jelas tentang spesifikasi LED dapat dilihat dalam **Lampiran3.**

Outline dan simbol skematis untuk *Light Emiting Diode* (LED) ditunjukkan dalam Gambar 2.5



Gambar 2.5. *Outline* dan Simbol Skematis untuk *Light Emiting Dioda* (LED).
Sumber: Mccomb, G., 2011: 396

2.5 Sensor *Photodiode*

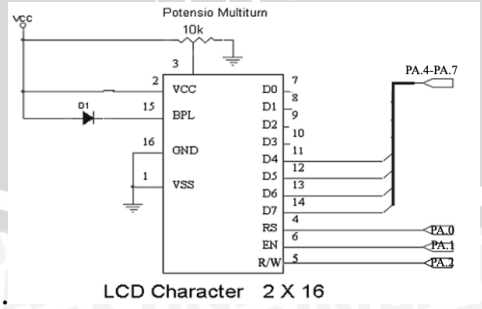
Photodiode adalah dioda yang bekerja saat dibias mundur (*reserve* bias). Prinsip kerja dari *photodiode* merupakan kebalikan dari prinsip kerja LED.

Photodiode akan mengalirkan arus melalui kutup anoda (+) menuju katoda (-) jika *photodiode* menerima cahaya. *Photodiode* dibuat dari semikonduktor dengan bahan yang populer adalah silicon(Si) atau galium arsenida (GaAs), dan yang lain meliputi InSb, InAs, PbSe. Material ini menyerap cahaya dengan karakteristik panjang gelombang mencakup: 2500 Å - 11000 Å untuk silicon, 8000 Å – 20,000 Å untuk GaAs. Gambar 2.6 menunjukkan *photodiode* yang terbuat dari bahan silikon tipe OP910.

Gambar 2.6. *Photodiode* Silikon Tipe OP910
Sumber: Nalwan, 2004

2.6 LCD (*Liquid Crystal Display*)

Liquid Crystal Display (LCD) merupakan komponen elektronika yang digunakan untuk menampilkan karakter baik berupa karakter angka, huruf, atau karakter lainnya, sehingga tampilan tersebut dapat dilihat secara visual. Gambar 2.7 menunjukkan rangkaian interface ke LCD Karakter 2x16 dan Tabel 2.1 menunjukkan Pin-Pin I/O LCD



Gambar 2.7 Rangkaian Interface ke LCD Karakter 2X16
Sumber: Nalwan, 2004

Tabel 2.1. Pin Pin I/O LCD

No	Simbol	Level	Fungsi
1	Vss		GND
2	Vcc		*Power supply* 5 Volt
3	Vee		LCD Drive
4	RS	H/L	H:*data input* L:*ins input*
5	R/W	H/L	H:*Read* L:*Write*
6	E	H	*Enable signal*
7	DB0	H/L	
8	DB1	H/L	
9	DB2	H/L	
10	DB3	H/L	*Data Bus*
11	DB4	H/L	
12	DB5	H/L	
13	DB6	H/L	
14	DB7	H/L	
15	V-BL		*Power supply* 4 – 4,2 Volt
16	V-BL		GND

Sumber: Nalwan, 2004

Pada perancangan sistem ini memakai LCD modul M1632 yang merupakan sebuah modul LCD dot matrik yang membutuhkan daya kecil.LCD modul M1632 dilengkapi panel LCD dengan tingkat kontras yang cukup tinggi serta pengendali LCD CMOS yang telah terpasang dalam modul tersebut. LCD modul M1632 mempunyai spesifikasi sebagai berikut :

- Memiliki 16 karakter dan 2 baris tampilan yang terdiri atas 5 x 7 dot matrik ditambah dengan kursor.
- Memerlukan catu daya DC 5 V.
- Otomatis *reset* saat catu daya dinyalakan.

BAB III
METODE PENELITIAN

3.1 Jenis dan Cara Perolehan Data

Data yang nanti akan diperoleh adalah berupa grafik respon performansi dari tiap sensor yang akan diuji. Kemudian setelah mendapat grafik respon dari pengujian tiap sensor maka akan dianalisis *overshoot* dan *settling time*nya.

3.2 Variabel dan Cara Analisis Data

Variabel yang digunakan dalam menganilis overshoot dan settling time antara lain :

1) *Sampling time*
2) *Time steady state*
3) *Time set point*

3.3 Penentuan Spesifikasi Alat

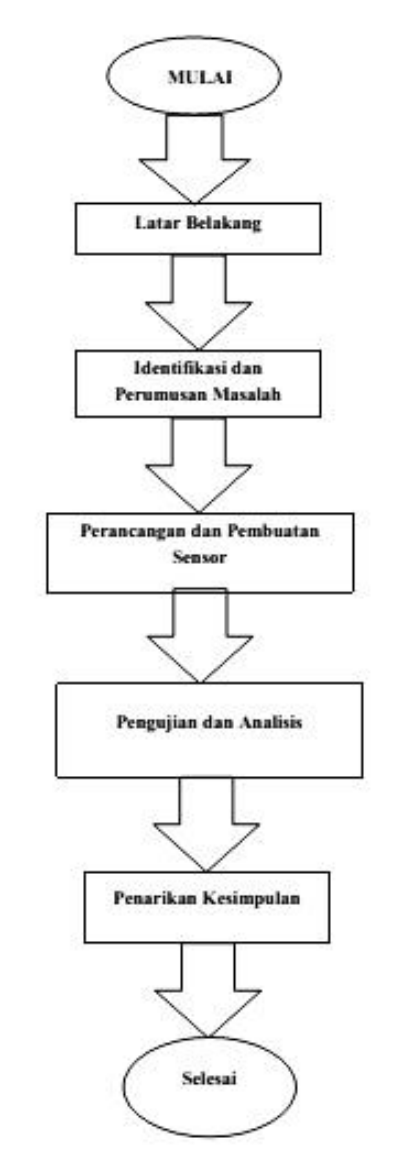
Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan yaitu :

1) Jumlah mikrokontroler adalah 1 *Master* dan 3 *Slave*.
2) Mikrokontroler *Master* yang digunakan adalah AVR ATMega16. Sedangkan 3 mikrokontroler *Slave* adalah AVR ATMega8.
3) LCD digunakan untuk menampilkan data pembacaan sensor.
4) Catu daya motor menggunakan AKI DC 12 V.
5) Alat gerak atau aktuator robot adalah motor DC NISCA N4775 12V 1930 rpm.

3.4 Kerangka Solusi Masalah

Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan penelitian yang terdapat di bab pendahuluan maka diperlukan langkah-langkah untuk menyelesaikan masalah. Kerangka solusi masalah adalah tahapan-tahapan yang perlu

dilakukan untuk menyelesaikan penelitian ini. Kerangka solusi masalah penelitian dibentuk dalam sebuah *flowchart* yang dapat dilihat dalam Gambar 3.1.



Gambar 3.1 Kerangka solusi Masalah Penelitian

3.4.1. Perancangan dan Pembuatan Mekanik

Perancangan dan pembuatan mekanik robot management sampah meliputi tahapan-tahapan sebagai berikut:

1) Penentuan ukuran dimensi robot.

2) Penentuan peletakan posisi motor, roda, dan sensor garis.

3) Pembuatan gambar perspektif 3 dimensi robot.

4) Pembuatan mekanik robot berdasarkan pada perancangan.

3.4.2. Perancangan dan Pembuatan Perangkat Keras

Secara garis besar perancangan perangkat keras dibagi dalam beberapa tahap sebagai berikut :

1) Penentuan spesifikasi alat.

2) Pembuatan blok diagram keseluruh sistem.

3) Penentuan komponen yang akan digunakan.

4) Merakit perangkat keras masing – masing blok.

3.4.3 Perancangan dan Pembuatan Perangkat Lunak

Setelah menentukan spesifikasi alat dan mengetahui karakteristik perangkat keras, maka dibutuhkan perangkat lunak untuk mengendalikan perangkat keras. Perancangan perangkat lunak dilakukan dengan pembuatan diagram alir (*flowchart*). Bahasa pemrograman untuk mikrokontroler adalah bahasa C dan *Software tool* yang digunakan adalah AVR Studio 4.18, WinAVR/AVR-GCC 1.6.

3.5 Pengujian dan Analisis Sistem

Untuk memastikan sistem ini berjalan dengan baik maka perlu dilakukan pengujian sistem, meliputi pengujian perangkat keras maupun perangkat lunak yang dilakukan baik secara blok rangkaian maupun keseluruhan sistem.

3.6 Pengambilan Kesimpulan dan Saran

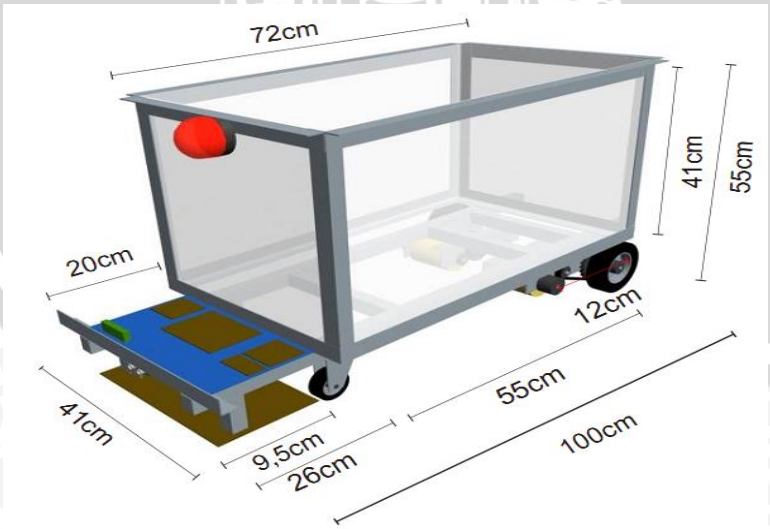
Kesimpulan diambil berdasarkan data yang diperoleh dari pengujian masing – masing sensor garis dan membandingkan respon dari ketiga pengujian sensor garis tersebut pada robot management sampah. Sedangkan saran diberikan untuk pengembangan skripsi ini.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Pada bab ini akan dijelaskan mengenai perancangan dan pembuatan robot *Management* sampah mulai dari perancangan mekanik robot, perancangan perangkat keras, dan perancangan perangkat lunak. Perancangan perangkat keras meliputi perancangan diagram blok sistem, minimum sistem *ATMega8*, perancangan sensor garis dan perancangan desain sensor garis. Perancangan dan pembuatan dilakukan secara bertahap dan sistematis, sehingga akan memudahkan dalam analisis sistem.

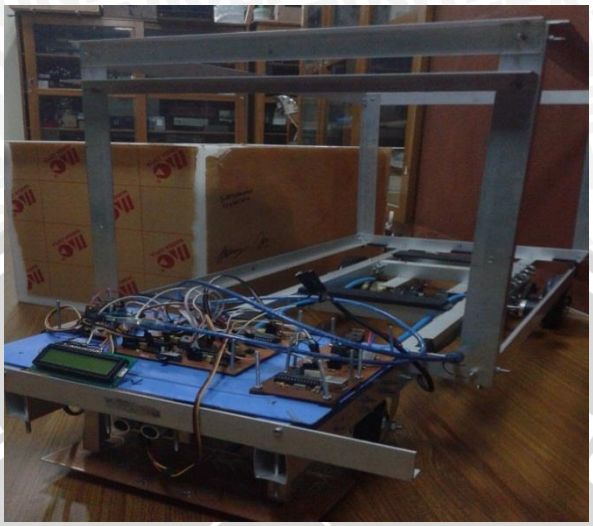
4.1. Perancangan Mekanik Robot

Mekanik robot line follower dirancang menyerupai miniatur truk pengangkut dengan panjang 100 cm, lebar 40 cm dan tinggi 45 cm. Robot memiliki box dengan ukuran 70 cm x 40 cm x 40 cm, dua roda depan merupakan roda bebas dan dua roda belakang yang dikopel motor DC sebagai penggerak dengan sistem differensial (terpisah). Desain mekanik robot yang digambar dengan menggunakan *software 3ds Max 9* tampak perspektif depan samping ditunjukkan dalam Gambar 4.1.



Gambar 4.1. Desain Mekanik Robot Tampak Depan Samping

Sedangkan robot *line follower* hasil perancangan ditunjukkan dalam Gambar 4.2.



Gambar 4.2. Robot *Management* Sampah

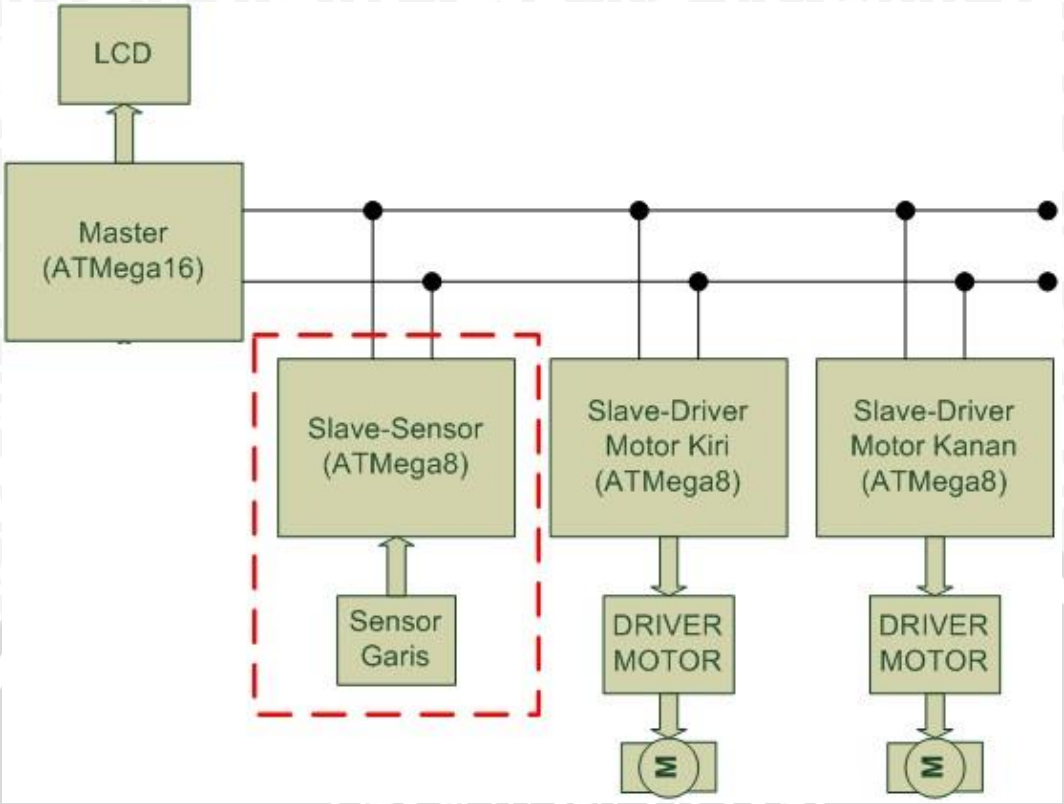
Dalam Gambar 4.2 terdapat kerangka *box* yang berfungsi sebagai tempat *box* sampah, rangkaian elektrik diletakkan dibagian kepala robot, sensor garis diletakkan dibagian depan bawah robot.

4.2 Perancangan Perangkat Keras (*Hardware*)

Dalam perancangan perangkat keras terdapat enam bagian perancangan, yaitu diagram blok secara keseluruhan, diagram blok sensor, perancangan sistem catu daya sebagai penyuplai tegangan, perancangan minimum sistem mikrokontroler ATmega8, perancangan tiga buah desain sensor garis, dan penentuan nilai tegangan keluaran pada setiap sensor garis.

4.2.1 Diagram Blok

Secara garis besar, diagram blok perancangan *hardware* yaitu diagram blok sistem keseluruhan robot dan diagram blok sensor garis. Diagram blok sistem keseluruhan robot ditunjukkan dalam gambar 4.3 dan gambar blok diagram untuk sensor garis ditunjukkan dalam gambar 4.4.

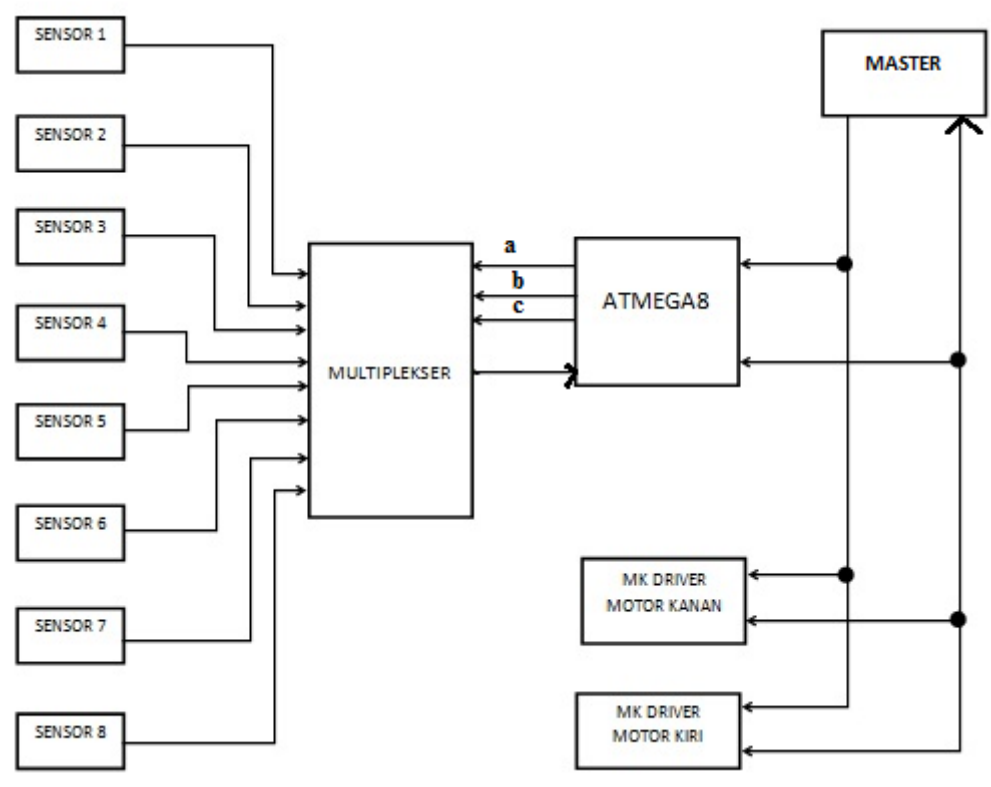


Gambar 4.3. Diagram Blok keseluruhan sistem

Dalam penelitian ini terdapat bagian perancangan dan pembuatan perangkat keras, yaitu:

1) Sensor garis digunakan untuk membaca garis dengan menggunakan sensor *photodiode*.
2) Mikrokontroler Slave-Sensor AVR ATMega8 sebagai pengolah pembacaan sensor.
3) Mikrokontroler Master AVR ATMega16 sebagai pengolah data dan pusat kendali sistem dari robot *management* sampah.
4) LCD digunakan untuk menampilkan pembacaan sensor atau data.
5) 2 buah Mikrokontroler ATMega8 Slave-Driver Motor Kiri dan Kanan sebagai pengendali Driver Motor Kiri dan Kanan.
6) 2 buah *Driver motor* digunakan untuk mengendalikan sinyal dari mikrokontroler untuk menggerakkan motor.
7) 2 buah Motor DC NISCA N4775 12 V 1930 rpm sebagai alat gerak atau aktuator robot.

Sedangkan dalam penelitian ini yang difokuskan hanya pada perancangan sensor garisnya, yaitu diagram blok yang ditunjukkan dalam Gambar 4.5.



Gambar 4.4. Diagram Blok Sensor Garis

Sedangkan cara kerja pada blok sensor garis berdasarkan blok diagram yaitu :

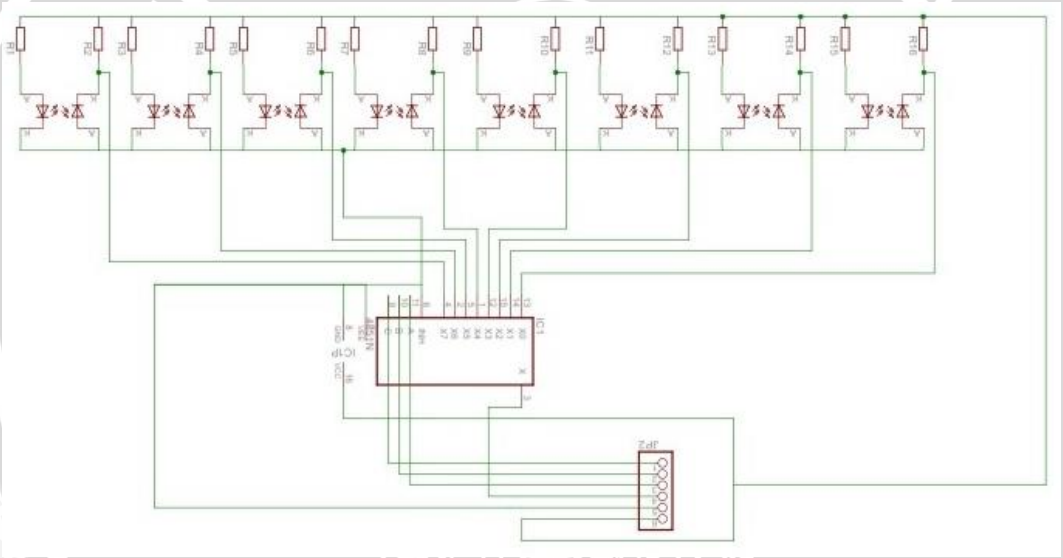
1) Menggunakan 8 buah sensor *photodiode*.
2) Menggunakan Multiplekser (sebagai switching untuk 8 sensor dan 1 pin adc).
3) Mikrokontroler membaca satu persatu sensor untuk pengolahan data.
4) Data diolah di mikrokontroler untuk membedakan warna hitam dan putih.

Tabel kebenaran pembacaan sensor yang diberi masukan logika dari mikrokontroler ditunjukkan dalam Tabel 4.1.

Tabel 4.1. Tabel Kebenaran Pembacaan Sensor

Input Dari Mikrontroler			
a	B	c	Sensor Aktif
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

Diagram blok sensor garis yang ditunjukkan dalam Gambar 4.4 kemudian direalisasikan dalam bentuk rangkaian yang ditunjukkan dalam Gambar 4.5.



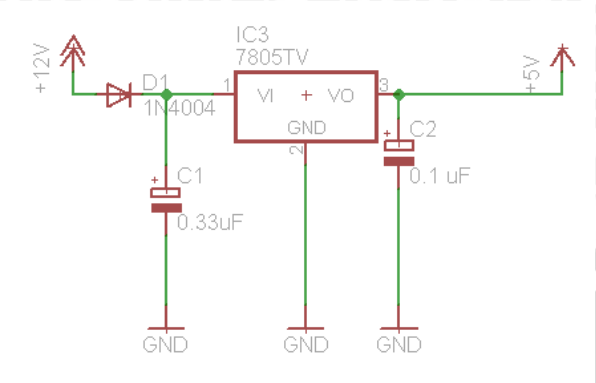
Gambar 4.5. Skema Rangkaian Sensor Garis

4.2.2 Perancangan Catu Daya Sistem

Robot ini menggunakan dua jenis catu daya. Catu daya 12V untuk motor yang bersumber dari AKI 12V. Serta catu daya 5V untuk rangkaian minimum sistem mikrokontroler ATmega16 dan ATmega8 yang bersumber dari baterai *lipo* (*lithium polimer*) 11,1V.

Pada perancangan digunakan catu daya sebesar 5V yang diperoleh dari rangkaian *Fixed Output Regulator* pada datasheet LM7805. IC78xx mempunyai

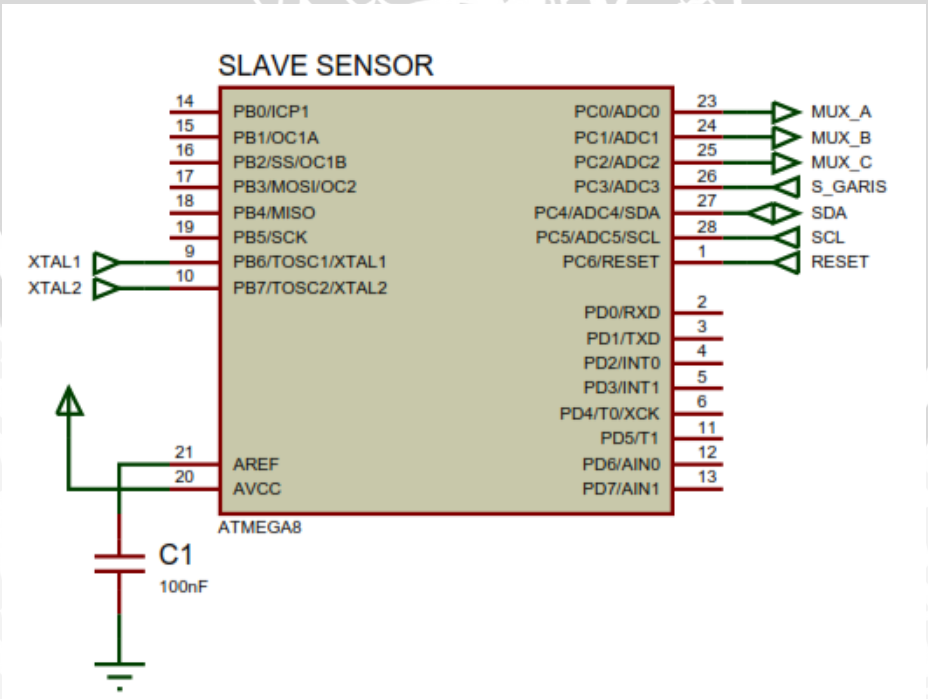
tiga kaki, satu V_{in}, satu untuk V_{out} dan satu untuk GND (Blocher, 2003). Skema rangkaian catu daya ditunjukkan dalam Gambar 4.6.



Gambar 4.6.Rangkaian Catu daya 5V

4.2.3 Perancangan Minimum Sistem Mikrokontroler ATmega8

Pada robot ini juga digunakan 3 mikrokontroler ATmega8 sebagai *slave*. Fungsi dari 3 mikrokontroler tersebut antara lain sebagai pengolah data sensor *photodiode*, pengatur arah dan kecepatan putaran motor kanan,dan pengatur arah dan kecepatan putaran motor kiri. Konfigurasi kaki I/O dari mikrokontroler *slave* (ATmega8) ditunjukkan dalam Gambar 4.7.



Gambar 4.7. Rangkaian Minimum Sistem Mikrokontroler ATmega8

Pin ADC mikrokontroler ATMega8 digunakan untuk membaca sensor *photodiode* melalui multiplekser analog. Pembacaan 8 sensor *photodiode* tersebut menggunakan metode scanning/switching dengan multiplekser analog tersebut. pemilihan sensor mana yang akan dibaca dapat diset melalui pin selector pada multiplekser. Seting inisialisasi ADC pada ATMega8 adalah:

- Resolusi yang digunakan yaitu 8-bit (nilai maksimal 255).
- Menggunakan Vref AVCC.
- Pin AVCC dihubungkan ke VCC yang tegangannya bernilai 5V.

4.2.4 Perancangan Sensor Garis

Untuk merancang desain sensor garis *software* yang digunakan adalah *Eagle 6.4.0*, dan untuk pembuatan sensor garis menggunakan beberapa komponen seperti :

1) 8 buah sensor *photodiode*.
2) 8 buah LED superbright.
3) 8 buah resistor 300 Ω.
4) 8 buah resistor 10 kΩ.
5) Pin header.
6) Multiplekser.

Untuk membatasi arus yang mengalir sebesar 30 mA menuju LED (D1) dengan tegangan +VCC sebesar 5 Volt, digunakan resistansi (R1) sebesar 300 Ω.

$$VCC = V_{D1} + V_{R1}$$
$$5\text{V} = 4\text{V} + V_{R1}$$
$$V_{R1} = 1\text{V} \qquad (4.1)$$

VCC = Tegangan sumber DC (V)
V_{D1} = Tegangan LED (V)
V_{R1} = Tegangan resistor (V)

Sehingga didapatkan resistansi R1 sebesar :

$$V_{R1} = I.R1$$
$$1\text{V} = 0{,}003\text{A} \,.\, R1$$
$$R1 = 300\Omega \qquad (4.2)$$

V_{R1} = Tegangan resistor (V)
I = Arus yang mengalir (A)
$R1$ = Resistansi resistor (Ω)

Tegangan keluaran hasil penerimaan pantulan cahaya oleh *photodiode* (D2), akan dikirim menuju pin ADC ATMega8 melalui multiplekser.

Untuk membatasi arus yang mengalir sebesar 0,625 mA menuju *photodiode* (D2) dengan tegangan +VCC sebesar 5 Volt, digunakan resistansi (R2) sebesar 10 kΩ.

$$VCC = VD2 + VR2$$

$$5\ V = (-1{,}25)V + VR2$$

$$VR2 = 6{,}25\ V \qquad (4.3)$$

VCC = Tegangan Sumber DC (V)

VD2 = Tegangan Pada *Photodiode* (V)

VR2 = Tegangan Pada R2 (V)

Sehingga didapatkan resistansi R2 sebesar :

$$VR2 = I\ .R2$$

$$6{,}25\ V = (6{,}25.10^{-4})A.R2$$

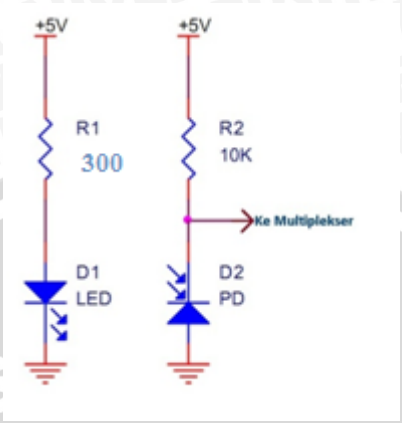
$$R2 = 10\ k\Omega \qquad (4.4)$$

VR2 = Tegangan Pada R2 (V)

I = Arus yang mengalir pada *photodiode* (A)

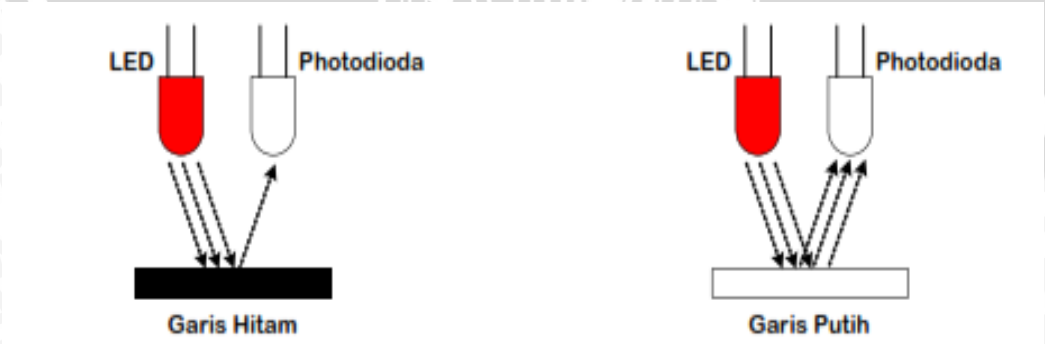
R2 = Resistansi R2 (Ω)

Untuk membuat rangkaian pembagi tegangan. Skema rangkaian sensor garis ditunjukkan dalam gambar 4.8.



Gambar 4.8. Rangkaian Sensor Garis

Prinsip kerja sensor garis hanya memanfaatkan sifat cahaya yang akan dipantulkan jika mengenai benda berwarna terang dan akan diserap jika mengenai benda berwarna gelap. Sebagai sumber cahaya kita gunakan LED (Light Emiting Diode) yang akan memancarkan cahaya merah. Dan untuk menangkap pantulan cahaya LED, kita gunakan *photodiode*. Jika sensor berada diatas garis hitam maka *photodiode* akan menerima sedikit sekali cahaya pantulan. Tetapi jika sensor berada diatas garis putih maka *photodiode* akan menerima banyak cahaya pantulan. Ilustrasi prinsip kerja sensor garis ditunjukkan dalam Gambar 4.9 :



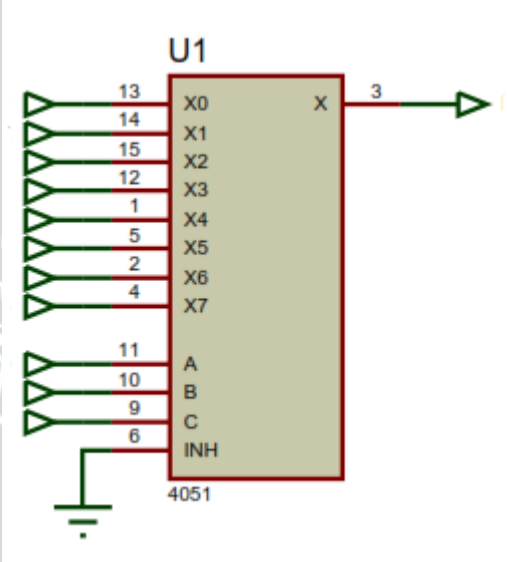
Gambar 4.9. Prinsip Kerja Sensor Garis

Sifat dari *photodiode* 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.

4.2.5 Rangkaian Multiplekser

Rangkaian multiplekser berfungsi sebagai *switching* menuju pin ADC pada mikrokontroler ATMega8. Karena pada rangkaian sensor garis membutuhkan 8 pin ADC untuk memproses sensor *photodiode* sedangkan pada mikrokontroler ATMega8 hanya terdapat 7 pin ADC, sehingga dibutuhkan rangkaian multiplekser untuk memproses 8 sensor *photodiode*. Rangkaian multiplekser ditunjukkan dalam gambar 4.10.



Gambar 4.10. Rangkaian Multiplekser

Berikut ini adalah fungsi dari kaki-kaki rangkaian multiplekser dalam gambar 4.11 :

- Pin X0 : Input dari sensor *photodiode* 1
- Pin X1 : Input dari sensor *photodiode* 2
- Pin X2 : Input dari sensor *photodiode* 3
- Pin X3 : Input dari sensor *photodiode* 4
- Pin X4 : Input dari sensor *photodiode* 5
- Pin X5 : Input dari sensor *photodiode* 6
- Pin X6 : Input dari sensor *photodiode* 7
- Pin X7 : Input dari sensor *photodiode* 8

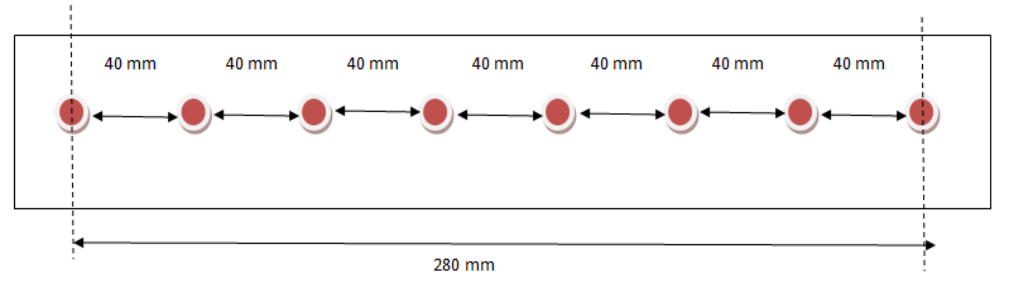
- Pin A : Input dari mikrokontroler PortC0
- Pin B : Input dari mikrokontroler PortC1
- Pin C : Input dari mikrokontroler PortC2
- Pin INH : *Ground* dari ADC
- Pin X : Menuju portC3 (ADC)

4.2.6 Perancangan Desain Sensor Garis

Pada skripsi ini ada 3 desain sensor yang akan dianalisis dan dilakukan pengujian yaitu sensor garis bentuk lurus, sensor garis bentuk setengah lingkaran, dan yang terakhir sensor bentuk setengah lingkaran.

4.2.6.1 Perancangan Desain Sensor Garis Bentuk Lurus

Perancangan sensor garis bentuk lurus merupakan dasar dari variasi posisi sensor garis yang akan dianalisis. Gambar perancangan sensor bentuk garis lurus ditunjukkan dalam Gambar 4.11.



Gambar 4.11. Perancangan Sensor Garis Bentuk Lurus

Sensor ini mempunyai spesifikasi sebagai berikut :

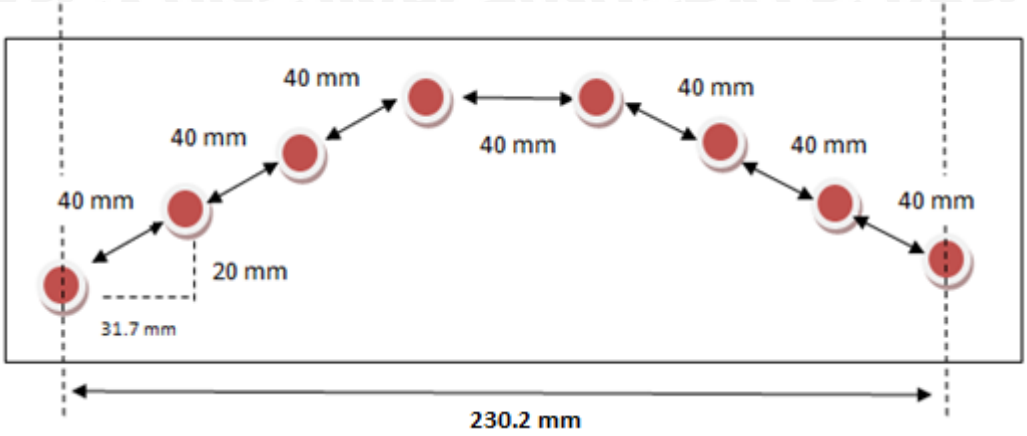
- Panjang sensor 280 mm
- Mempunyai jarak antar sensor garis sebesar 40 mm

Jarak antar sensor *photodiode* diberi jarak sebesar 40 mm karena lintasan garis hitam yang digunakan selebar 60 mm. Antar sensor *photodiode* diberi jarak sebesar 40 mm supaya saat sensor garis mulai membaca lintasan hitam dapat dilakukan secara optimal.

4.2.6.2 Perancangan Desain Sensor Garis Bentuk Segitiga

Perancangan sensor garis bentuk segitiga adalah variasi dari sensor bentuk lurus. Modifikasi dilakukan dengan cara mengubah posisi sensor garis sehingga

membentuk pola segitiga. Gambar perancangan sensor bentuk segitiga ditunjukkan dalam Gambar 4.12

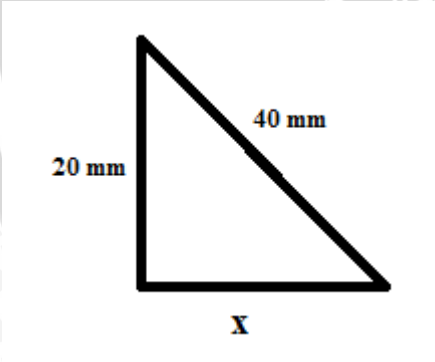


Gambar 4.12. Perancangan Sensor Garis Bentuk Segitiga

Spesifikasi dari sensor bentuk segitiga hampir sama dengan sensor bentuk lurus, yaitu :

- Panjang sensor 230,2 mm.
- Mempunyai jarak antar sensor 40 mm.

Berikut perhitungan spesifikasi dari desain sensor garis bentuk segitiga :



Dari gambar diatas maka x didapatkan sebagai berikut :

$$X^2 = (40)^2\ mm - (20)^2\ mm$$

$$X^2 = 1600\ mm - 400\ mm$$

$$X = \sqrt{1200}\ mm$$

$$X = 31{,}7\ mm \qquad (4.5)$$

Sehingga panjang sensor garis desain segitiga adalah :

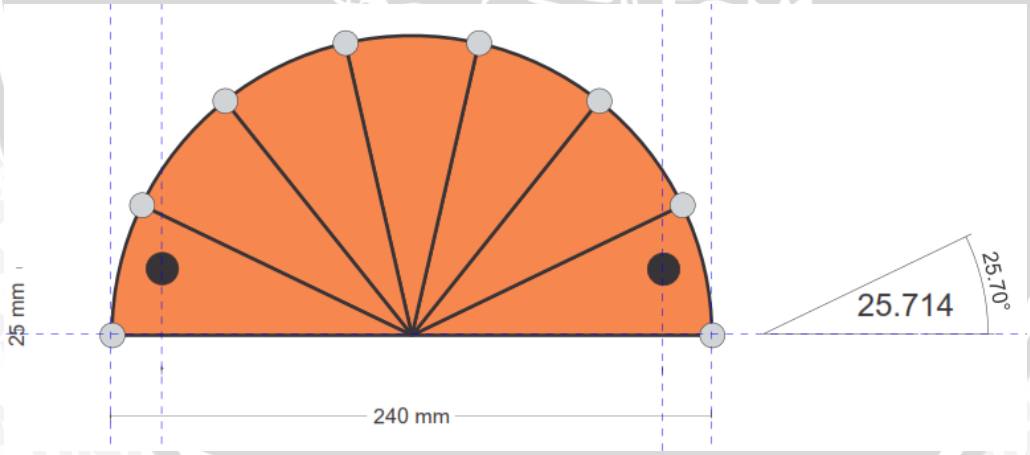
$$Panjang\ Sensor = (31{,}7\ x\ 8)\ mm + 40\ mm$$

$$Panjang\ Sensor = 230{,}2\ mm \quad (4.6)$$

Masih sama dengan desain sensor garis yang sebelumnya, perancangan sensor garis bentuk segitiga ini mempunyai jarak antar sensor *photodiode* sebesar 40 mm, dan masih dengan alas an yang sama yaitu agar pembacaan sensor garis lebih optimal pada lintasan garis hitam yang mempunyai lebar sebesar 60 mm.

4.2.6.3 Perancangan Desain Sensor Garis Bentuk Setengah Lingkaran

Perancangan sensor garis bentuk setengah lingkaran adalah varian bentuk sensor yang memerlukan ketelitian untuk mendapatkan bentuk yang presisi dan membentuk pola setengah lingkaran. Gambar perancangan sensor bentuk setengah lingkaran ditunjukkan dalam Gambar 4.13.



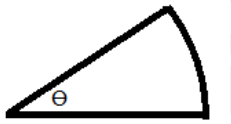
Gambar 4.13. Perancangan Sensor Garis Bentuk Setengah Lingkaran

Sensor bentuk setengah lingkaran mempunyai spesifikasi lebih kompleks dibandingkan dengan 2 varian lainnya, yaitu :

- Sudut antar sensor sebesar 25.70^0.
- Panjang sensor 240 mm.

Berikut perhitungan dari desain sensor bentuk setengah lingkaran :

Karena dalam perancangan sensor bentuk setengah lingkaran ini sama seperti lainnya yaitu menggunakan 8 buah sensor *photodiode* maka sudutnya adalah :



$$\theta = 180° \div 7$$

$$\theta = 25{,}70° \quad (4.6)$$

4.2.7 Penentuan Tegangan Keluaran Setiap Sensor Garis

Penentuan tegangan keluaran sensor garis bertujuan untuk menentukan jika pada saat sensor garis membaca garis hitam maka logika digitalnya adalah *high* dan sebaliknya jika membaca garis putih maka logikanya adalah *low*. Untuk mengetahui tegangan keluaran analog pada tiap sensor dilakukan secara manual dengan menggunakan voltmeter, sedangkan untuk mengetahui keluaran digital pada tiap sensor digunakan persamaan :

$$V_{ADC} = \frac{V_{analog}}{5} \; x \; 255 \quad (4.7)$$

Hasil perhitungan tegangan keluaran tiap sensor garis ditunjukkan dalam tabel 4.2.

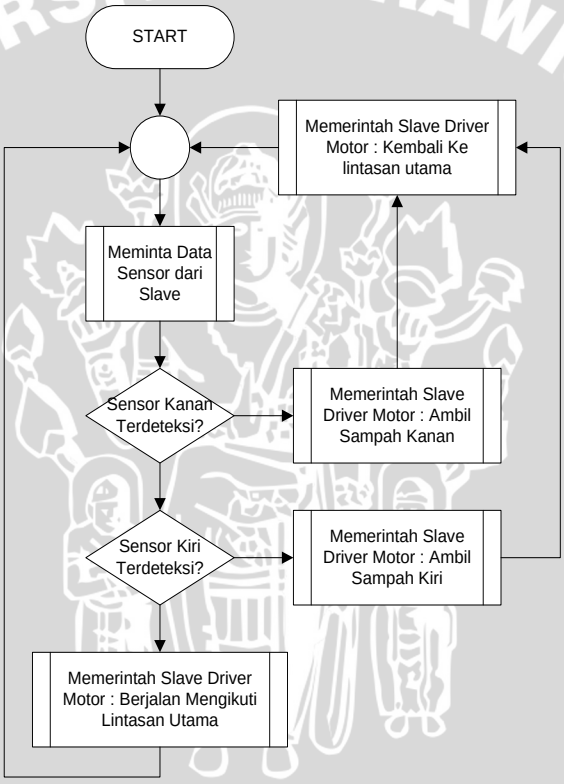
Tabel 4.2. Perhitungan Tegangan keluaran Analog dan Digital

Sensor	Tegangan Analog (V)		Konversi ADC (mV)	
	Hitam	Putih	Hitam	Putih
1	3.48	1.2	177.48	61.2
2	3.2	1.1	137.85	56.1
3	3.13	1.34	159.63	68.34
4	3.2	1.1	137.85	56.1
5	3.2	1.25	137.85	63.75
6	3.18	1.46	162.18	74.46
7	3.3	1.25	168.3	63.75
8	3.2	1.2	137.85	61.2

Setelah didapatkan hasil seperti pada tabel 4.2 maka tiap sensor akan diinisialisasi agar dapat membedakan ketika membaca garis hitam menjadi logika *high* dan ketika membaca garis putih menjadi logika *low*.

4.3 Perancangan Perangkat Lunak (*Software*)

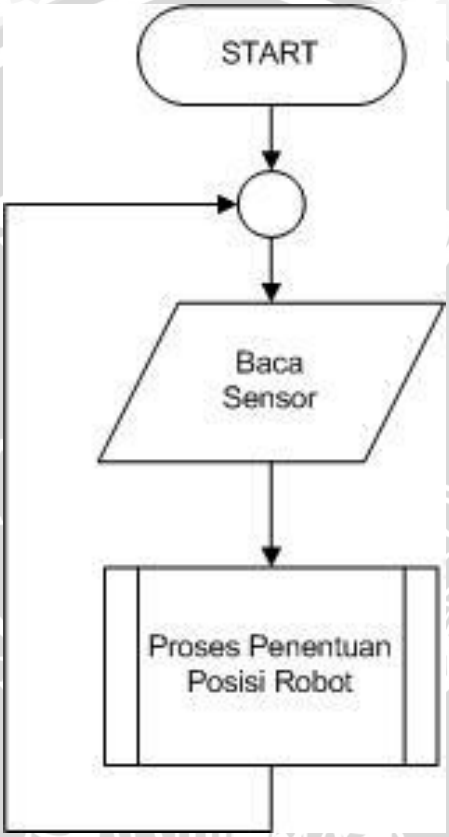
Setelah menentukan spesifikasi alat dan mengetahui karakteristik perangkat keras, maka dibutuhkan perangkat lunak untuk mengendalikan perangkat keras. Perancangan perangkat lunak dilakukan dengan pembuatan diagram alir (*flowchart*). Bahasa pemrograman untuk mikrokontroler adalah bahasa C dan Software tool yang digunakan adalah AVR Studio 4.18, WinAVR/AVR-GCC 1.6. Diagram alir perancangan perangkat lunak mikrokontroler ditunjukkan dalam Gambar 4.14.



Gambar 4.14. *Flowchart* sistem secara kesuluruhan

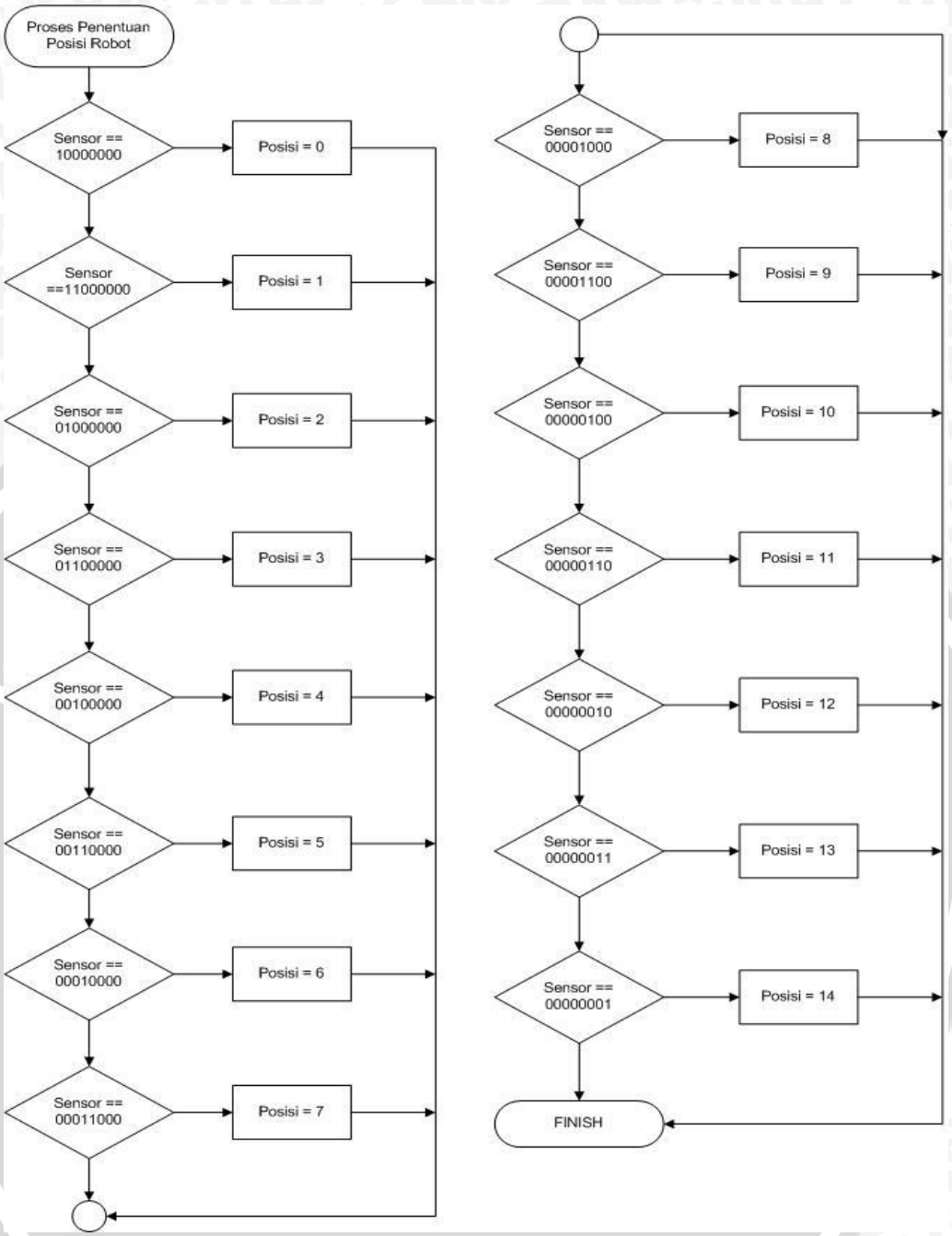
4.3.1 Perancangan Perangkat Lunak Pada Sensor Garis

Diagram alir pada sensor garis dirancang untuk memerintah mikrokontroler agar sensor garis dapat melakukan pembacaan dengan benar sehingga dapat memperoleh data yang diinginkan dengan menggunakan algoritma yang ditunjukkan dalam Lampiran 2. *Flowchart* Pembacaan Sensor Pada Robot *Management* Sampah ditunjukkan dalam Gambar 4.15:



Gambar 4.15. *Flowchart* Pembacaan Sensor Pada Robot *Management* Sampah

Sedangkan untuk *Flowchart* Proses Penentuan Posisi Robot ditunjukkan dalam Gambar 4.16.



Gambar 4.16. *Flowchart* Proses Penentuan Posisi Robot

BAB V
PENGUJIAN DAN PEMBAHASAN

Pengujian dan analisis dilakukan untuk menganalisis apakah sistem telah bekerja sesuai perancangan. Pengujian dilakukan per blok kemudian secara keseluruhan. Pengujian yang perlu dilakukan adalah sebagai berikut:

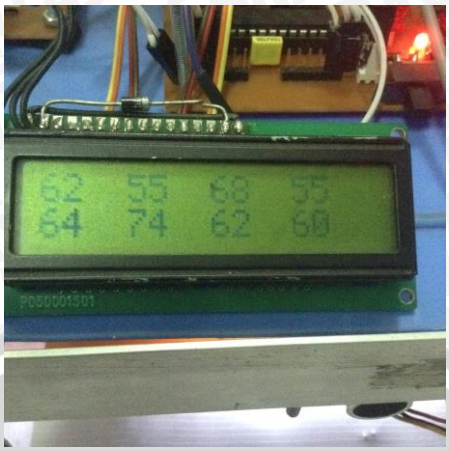
1) Pengujian nilai keluaran ADC
2) Penentuan nilai posisi sensor
3) Pengujian sensor bentuk segitiga.
4) Pengujian sensor bentuk setengah lingkaran.
5) Pengujian sensor bentuk lurus (180^0).

5.1. Pengujian Nilai Keluaran ADC

Pengujian nilai keluaran ADC berfungsi untuk mengetahui hasil pembacaan nilai ADC pada mikrokontroler sesuai dengan hasil perhitungan ADC secara teori. Pada pengujian ini sensor *photodiode* ditempatkan pada garis hitam secara bergantian per-sensor. Pengujian nilai ADC pada garis hitam ditunjukkan dalam Gambar 5.1.

Gambar 5.1. Pengujian Nilai ADC Pada Garis Hitam

Sedangkan pengujian nilai ADC pada garis putih ditunjukkan dalam Gambar 5.2.



Gambar 5.2. Pengujian Nilai ADC Pada Garis Putih

Untuk mengetahui hasil pembacaan ADC, nilai ADC pengujian dibandingkan dengan perhitungan ADC secara teori yang telah dihitung pada BAB IV. Perbandingan hasil nilai ADC secara teori dan secara pengujian ditunjukkan dalam Tabel 5.1.

Tabel 5.1. Perbandingan Hasil nilai ADC Secara Teori Dan Secara Pengujian

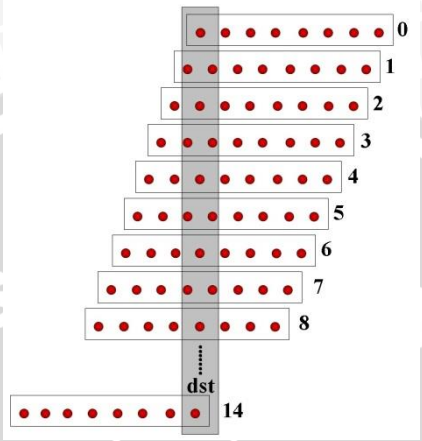
Sensor	Hitam		Putih	
	Teori (mV)	Pengujian (mV)	Teori (mV)	Pengujian (mV)
1	177.48	175	61.2	62
2	137.85	139	56.1	55
3	159.63	156	68.34	68
4	137.85	135	56.1	55
5	137.85	139	63.75	64
6	162.18	162	74.46	74
7	168.3	169	63.75	62
8	137.85	137	61.2	60

Hasil pengujian nilai ADC secara teori dan secara sistem mempunyai hasil yang berbeda, disebabkan beberapa faktor antara lain pengaruh cahaya luar, toleransi alat ukur dan ketelitian sistem. Tetapi dengan hasil yang berbeda tidak terlalu mempengaruhi sistem karena error yang terjadi tidak terlalu berbeda jauh.

5.2. Penentuan Nilai Posisi Sensor

Penentuan nilai posisi sensor ini diperlukan untuk melakukan proses pengujian setiap bentuk variasi sensor garis. Untuk melakukan pengujian diperlukan suatu nilai error yaitu selisih antara *set point* dengan posisi robot yang terukur sensor garis. Nilai *set point* (posisi sensor robot yang diinginkan) yang

akan dipakai harus ditentukan terlebih dahulu sebelum mencari grafik osilasi berkesinambungan. *Set point* pada sistem ini adalah posisi robot yang berada pada tengah garis yang bernilai 7. Nilai posisi sensor robot *management* sampah ditunjukkan dalam Gambar 5.1.

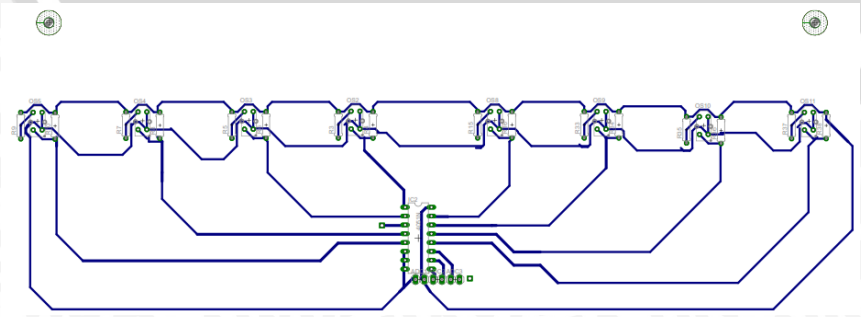


Gambar 5.3. Nilai Posisi Sensor Robot *Management* Sampah

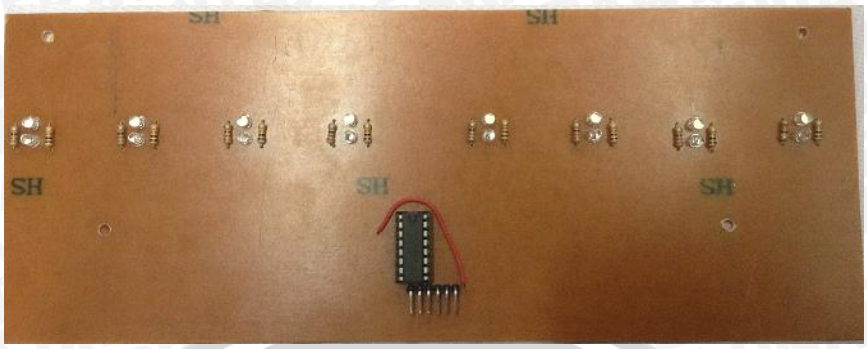
Pengujian ini bertujuan untuk mengetahui performansi jalannya robot apabila robot menggunakan variasi sensor garis yang berbeda. Proses pengujian ini dilakukan 3 kali setiap sensor, yaitu mengetahui performansi robot jika pengujian dimulai dari posisi 0, 7, dan 14 dengan diberi variasi sensor yang berbeda-beda.

5.3. Pengujian Sensor Bentuk Lurus

Sebelum masuk kedalam pengujian berikut ini adalah hasil *layout* dan bentuk fisik sensor garis bentuk lurus yang merupakan dasar dari bentuk sensor garis. Hasil *layout* sensor bentuk lurus dapat dilihat dalam Gambar 5.4 sedangkan bentuk fisiknya dapat dilihat dalam Gambar 5.5.



Gambar 5.4. Desain *Layout* Sensor Garis Bentuk Lurus



Gambar 5.5. Sensor Garis Bentuk Lurus

Pengujian ini bertujuan untuk mengetahui bagaimana performa robot ketika menggunakan sensor bentuk lurus (180°). Berikut hasil pengujian sensor bentuk lurus pada posisi 0, 7 dan 14 dengan *sampling time* 200 ms.

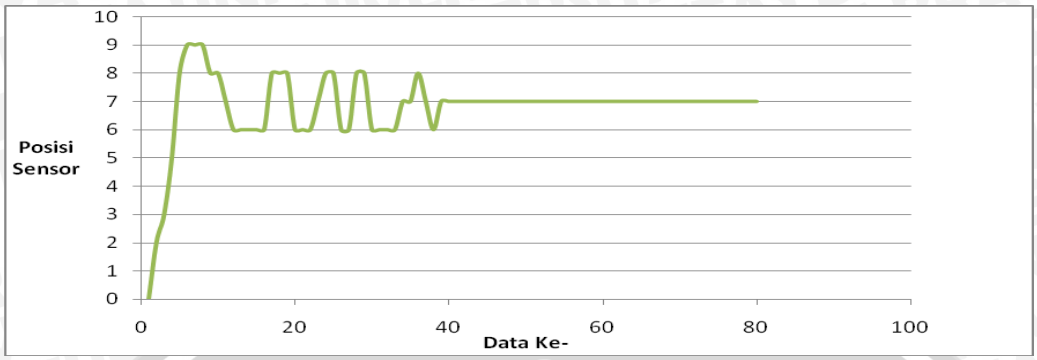
a. Hasil pengujian sensor pada posisi 0

Hasil pengujian sensor bentuk lurus pada posisi 0 dapat dilihat dalam tabel 5.2

Tabel 5.2. Pengujian Sensor Bentuk Lurus Pada Posisi 0

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	0	21	6	41	7	61	7
2	2	22	6	42	7	62	7
3	3	23	7	43	7	63	7
4	5	24	8	44	7	64	7
5	8	25	8	45	7	65	7
6	9	26	6	46	7	66	7
7	9	27	6	47	7	67	7
8	9	28	8	48	7	68	7
9	8	29	8	49	7	69	7
10	8	30	6	50	7	70	7
11	7	31	6	51	7	71	7
12	6	32	6	52	7	72	7
13	6	33	6	53	7	73	7
14	6	34	7	54	7	74	7
15	6	35	7	55	7	75	7
16	6	36	8	56	7	76	7
17	8	37	7	57	7	77	7
18	8	38	6	58	7	78	7
19	8	39	7	59	7	79	7
20	6	40	7	60	7	80	7

Grafik respon pengujian sensor bentuk lurus pada posisi 0 ditunjukkan dalam Gambar 5.6



Gambar 5.6. Respon Pengujian Sensor Bentuk Lurus Pada Posisi 0

Hasil pengujian sensor bentuk lurus terhadap posisi 0 menghasilkan *settling time* dan *overshoot* maksimum, berikut perhitungannya :

- Maksimum *overshoot :*

$$\%MP = \frac{|Tpuncak - Tsteady\ state|}{Tsteadystate}\ x\ 100\ \%$$

$$\%MP = \frac{|9-7|}{7}\ x\ 100\ \%$$

$$\%MP = 28.5\% \qquad (5.1)$$

- *Settling Time* :

  Dari tabel 5.2 dapat diketahui bahwa settling time terjadi pada pencuplikan data ke-39 dengan sampling time sebesar 200 ms, maka :

$$TS = \frac{39\ x\ 200}{1000}$$

$$TS = 7{,}8\ s \qquad (5.2)$$

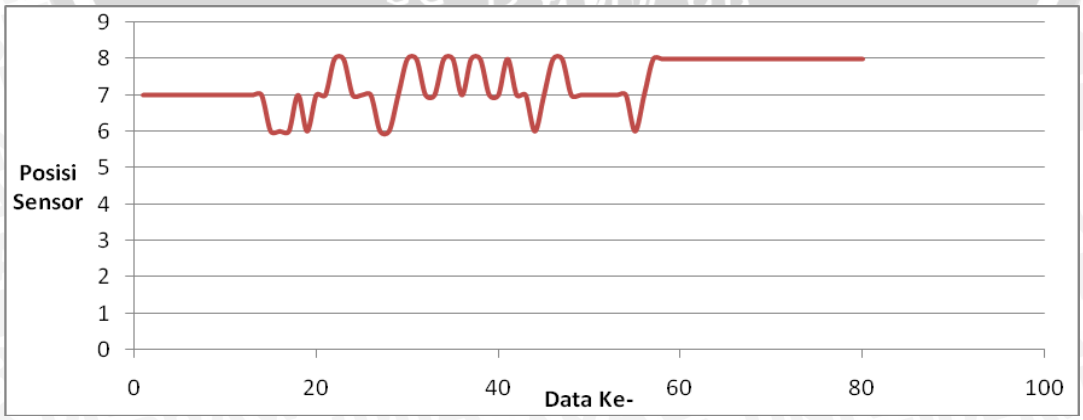
b. Hasil pengujian sensor pada posisi 7

Hasil pengujian sensor bentuk lurus pada posisi 7 dapat dilihat dalam tabel 5.3

Tabel 5.3. Pengujian Sensor Bentuk Segitiga Pada Posisi 7

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	7	21	7	41	8	61	8
2	7	22	8	42	7	62	8
3	7	23	8	43	7	63	8
4	7	24	7	44	6	64	8
5	7	25	7	45	7	65	8
6	7	26	7	46	8	66	8
7	7	27	6	47	8	67	8
8	7	28	6	48	7	68	8
9	7	29	7	49	7	69	8
10	7	30	8	50	7	70	8
11	7	31	8	51	7	71	8
12	7	32	7	52	7	72	8
13	7	33	7	53	7	73	8
14	7	34	8	54	7	74	8
15	6	35	8	55	6	75	8
16	6	36	7	56	7	76	8
17	6	37	8	57	8	77	8
18	7	38	8	58	8	78	8
19	6	39	7	59	8	79	8
20	7	40	7	60	8	80	8

Grafik respon pengujian sensor bentuk lurus pada posisi 7 ditunjukkan dalam Gambar 5.7



Gambar 5.7. Respon Pengujian Sensor Bentuk Lurus Pada Posisi 7

Hasil pengujian sensor bentuk lurus terhadap posisi 7 robot tidak dapat mencapai *set point* tetapi hanya terjadi osilasi.

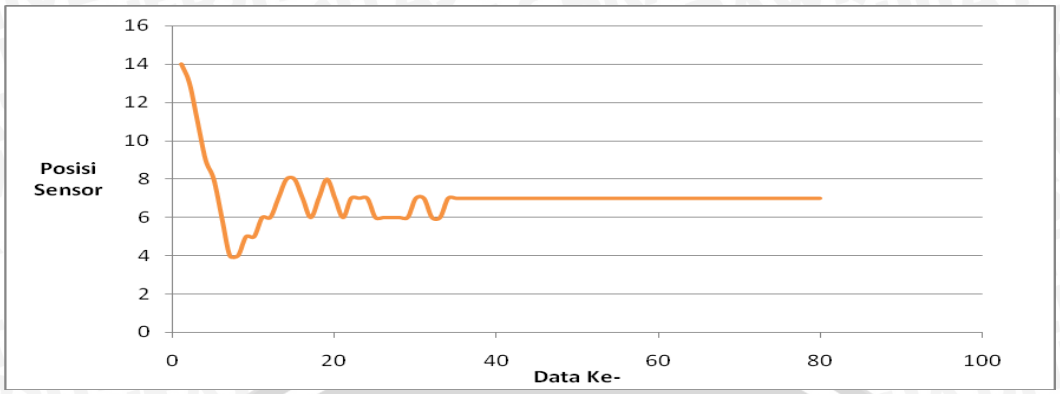
c. Hasil pengujian sensor pada posisi 14

Hasil pengujian sensor bentuk lurus pada posisi 14 dapat dilihat dalam tabel 5.4

Tabel 5.4. Pengujian Sensor Bentuk Lurus Pada Posisi 14

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	14	21	6	41	7	61	7
2	13	22	7	42	7	62	7
3	11	23	7	43	7	63	7
4	9	24	7	44	7	64	7
5	8	25	6	45	7	65	7
6	6	26	6	46	7	66	7
7	4	27	6	47	7	67	7
8	4	28	6	48	7	68	7
9	5	29	6	49	7	69	7
10	5	30	7	50	7	70	7
11	6	31	7	51	7	71	7
12	6	32	6	52	7	72	7
13	7	33	6	53	7	73	7
14	8	34	7	54	7	74	7
15	8	35	7	55	7	75	7
16	7	36	7	56	7	76	7
17	6	37	7	57	7	77	7
18	7	38	7	58	7	78	7
19	8	39	7	59	7	79	7
20	7	40	7	60	7	80	7

Grafik respon pengujian sensor bentuk lurus pada posisi 14 ditunjukkan dalam Gambar 5.8



Gambar 5.8. Respon Pengujian Sensor Bentuk Lurus Pada Posisi 14

Hasil pengujian sensor bentuk segitiga terhadap posisi 14 menghasilkan *settling time* dan *overshoot* maksimum, berikut perhitungannya :

- Maksimum *overshoot*

$$\%MP = \frac{|Tpuncak - Tsteady\ state|}{Tsteadystate}\ x\ 100\ \%$$

$$\%MP = \frac{|-10+7|}{-7}\ x\ 100\ \%$$

$$\%MP = 42.86\% \quad (5.3)$$

- *Settling time*

  Dari tabel 5.4 dapat diketahui bahwa settling time terjadi pada pencuplikan data ke-34 dengan sampling time sebesar 200 ms, maka :

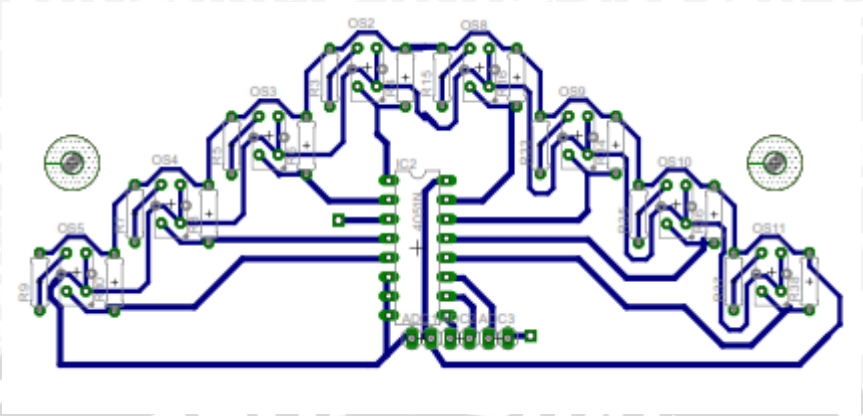
$$TS = \frac{34\ x\ 200}{1000}$$

$$TS = 6{,}8\ s \quad (5.4)$$

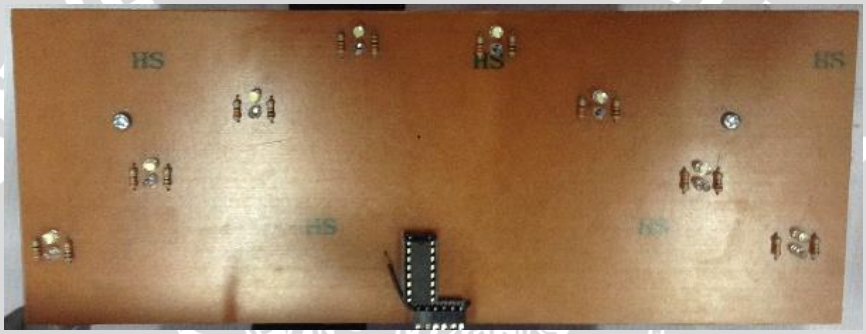
5.4. Pengujian Sensor Bentuk Segitiga

Bentuk varian sensor yang kedua ini adalah modifikasi dari sensor sebelumnya. Modifikasi dilakukan dengan cara memindah posisi sensor garis dan mendesain sehingga membentuk bentuk segitiga. Sebelum masuk kedalam pengujian berikut ini adalah hasil *layout* dan bentuk fisik sensor garis bentuk segitiga yang merupakan modifikasi dari bentuk sensor garis lurus. Hasil *layout*

sensor bentuk lurus ditunjukkan dalam Gambar 5.9 sedangkan bentuk fisiknya ditunjukkan dalam Gambar 5.10.



Gambar 5.9. Desain *Layout* Sensor Garis Bentuk Segitiga



Gambar 5.10. Bentuk fisik Sensor Bentuk Segitiga

Pengujian ini bertujuan untuk mengetahui bagaimana performansi robot ketika menggunakan sensor bentuk segitiga. Berikut hasil pengujian sensor bentuk segitiga pada posisi 0, 7 dan 14.

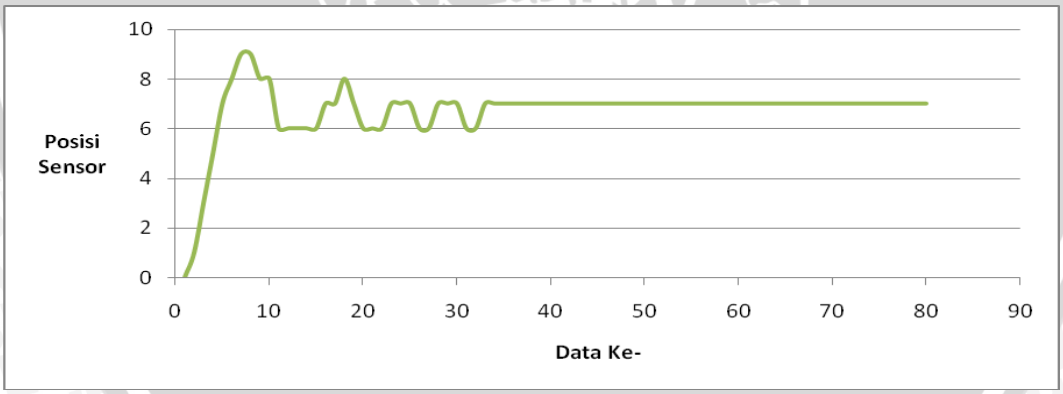
a. Hasil pengujian sensor pada posisi 0

Hasil pengujian sensor bentuk segitiga pada posisi 0 dapat dilihat dalam tabel 5.5

Tabel 5.5. Pengujian Sensor Bentuk Segitiga Pada Posisi 0

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	0	21	6	41	7	61	7
2	1	22	6	42	7	62	7
3	3	23	7	43	7	63	7
4	5	24	7	44	7	64	7
5	7	25	7	45	7	65	7
6	8	26	6	46	7	66	7
7	9	27	6	47	7	67	7
8	9	28	7	48	7	68	7
9	8	29	7	49	7	69	7
10	8	30	7	50	7	70	7
11	6	31	6	51	7	71	7
12	6	32	6	52	7	72	7
13	6	33	7	53	7	73	7
14	6	34	7	54	7	74	7
15	6	35	7	55	7	75	7
16	7	36	7	56	7	76	7
17	7	37	7	57	7	77	7
18	8	38	7	58	7	78	7
19	7	39	7	59	7	79	7
20	6	40	7	60	7	80	7

Grafik respon pengujian sensor bentuk segitiga pada posisi 0 ditunjukkan dalam gambar 5.11



Gambar 5.11. Respon Pengujian Sensor Bentuk SegitigaPada Posisi 0

Hasil pengujian sensor bentuk segitiga terhadap posisi 0 menghasilkan *settling time* dan *overshoot* maksimum, berikut perhitungannya :

- Maksimum *overshoot* :

$$\%MP = \frac{|Tpuncak - Tsteady\ state|}{Tsteadystate}\ x\ 100\ \%$$

$$\%MP = \frac{|9-7|}{7}\ x\ 100\ \%$$

$$\%MP = 28.5\% \quad (5.5)$$

- *Settling time :*

  Dari tabel 5.4 dapat diketahui bahwa settling time terjadi pada pencuplikan data ke-33 dengan sampling time sebesar 200 ms, maka :

$$TS = \frac{33\ x\ 200}{1000}$$

$$TS = 6{,}6\ s \quad (5.6)$$

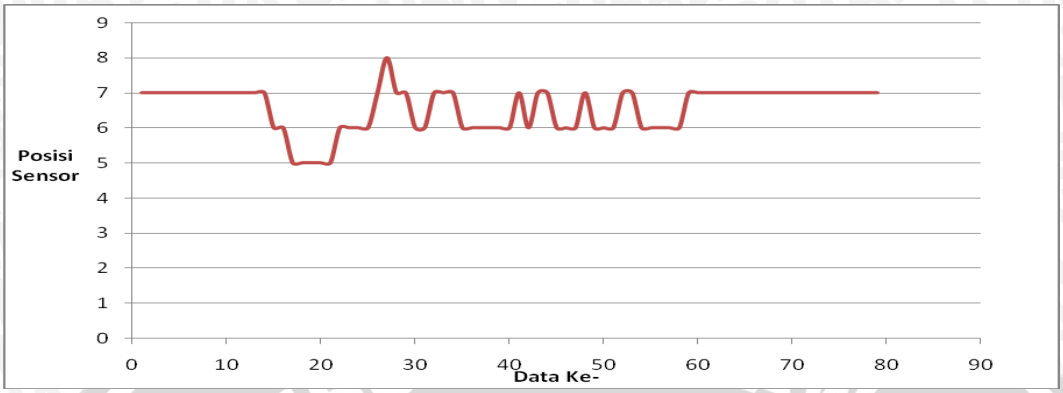
b. Hasil pengujian sensor pada posisi 7

Hasil pengujian sensor bentuk segitiga pada posisi 7 dapat dilihat dalam tabel 5.6

Tabel 5.6. Pengujian Sensor Bentuk segitiga Pada Posisi 7

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	7	21	5	41	7	61	7
2	7	22	6	42	6	62	7
3	7	23	6	43	7	63	7
4	7	24	6	44	7	64	7
5	7	25	6	45	6	65	7
6	7	26	7	46	6	66	7
7	7	27	8	47	6	67	7
8	7	28	7	48	7	68	7
9	7	29	7	49	6	69	7
10	7	30	6	50	6	70	7
11	7	31	6	51	6	71	7
12	7	32	7	52	7	72	7
13	7	33	7	53	7	73	7
14	7	34	7	54	6	74	7
15	6	35	6	55	6	75	7
16	6	36	6	56	6	76	7
17	5	37	6	57	6	77	7
18	5	38	6	58	6	78	7
19	5	39	6	59	7	79	7
20	5	40	6	60	7	80	7

Grafik respon pengujian sensor bentuk segitiga pada posisi 7 ditunjukkan dalam gambar 5.12



Gambar 5.12. Respon Pengujian Sensor Bentuk Segitiga Pada Posisi 7

Hasil pengujian sensor bentuk segitiga terhadap posisi 7 menghasilkan osilasi pada pencuplikan data ke-15 dan kembali menuju set point pada pencuplikan data ke-60.

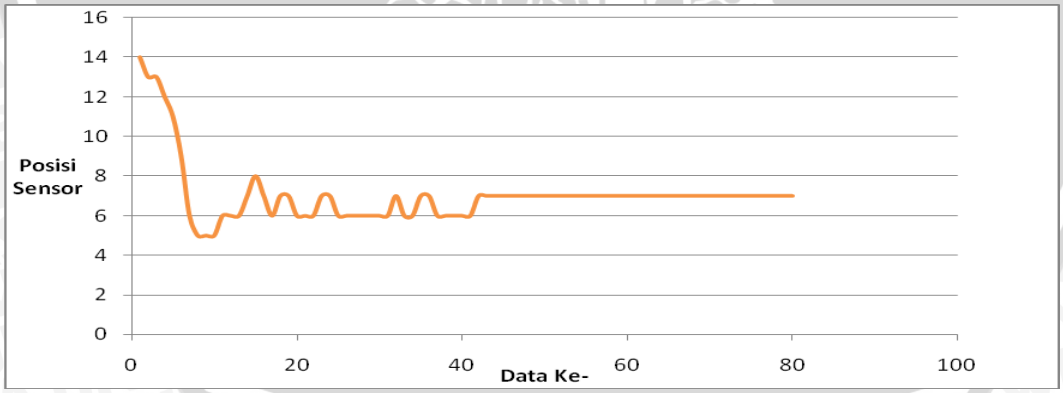
b. Hasil pengujian sensor pada posisi 14

Hasil pengujian sensor bentuk segitiga pada posisi 14 dapat dilihat dalam tabel 5.7.

Tabel 5.7. Pengujian Sensor Bentuk Segitiga Pada posisi 14

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	14	21	6	41	6	61	7
2	13	22	6	42	7	62	7
3	13	23	7	43	7	63	7
4	12	24	7	44	7	64	7
5	11	25	6	45	7	65	7
6	9	26	6	46	7	66	7
7	6	27	6	47	7	67	7
8	5	28	6	48	7	68	7
9	5	29	6	49	7	69	7
10	5	30	6	50	7	70	7
11	6	31	6	51	7	71	7
12	6	32	7	52	7	72	7
13	6	33	6	53	7	73	7
14	7	34	6	54	7	74	7
15	8	35	7	55	7	75	7
16	7	36	7	56	7	76	7
17	6	37	6	57	7	77	7
18	7	38	6	58	7	78	7
19	7	39	6	59	7	79	7
20	6	40	6	60	7	80	7

Grafik respon pengujian sensor bentuk segitiga pada posisi 14 ditunjukkan dalam gambar 5.13.



Gambar 5.13. Grafik Respon Pengujian Sensor Bentuk Segitiga Pada Posisi 14

Hasil pengujian sensor bentuk segitiga terhadap posisi 0 menghasilkan *settling time* dan *overshoot* maksimum, berikut perhitungannya :

- Maksimum *overshoot* :

$$\%MP = \frac{|Tpuncak - Tsteady\ state|}{Tsteadystate}\ x\ 100\ \%$$

$$\%MP = \frac{|9-7|}{7}\ x\ 100\ \%$$

$$\%MP = 28.5\% \quad (5.7)$$

- *Settling time :*

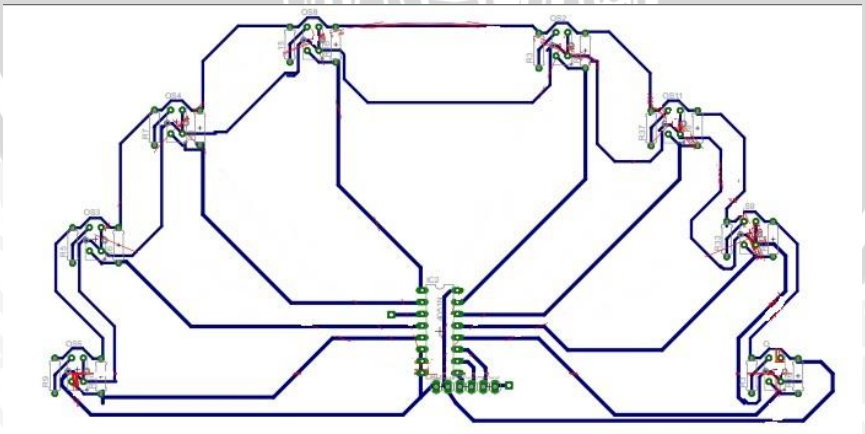
  Dari tabel 5.7 dapat diketahui bahwa settling time terjadi pada pencuplikan data ke-42 dengan sampling time sebesar 200 ms, maka :

$$TS = \frac{42\ x\ 200}{1000}$$

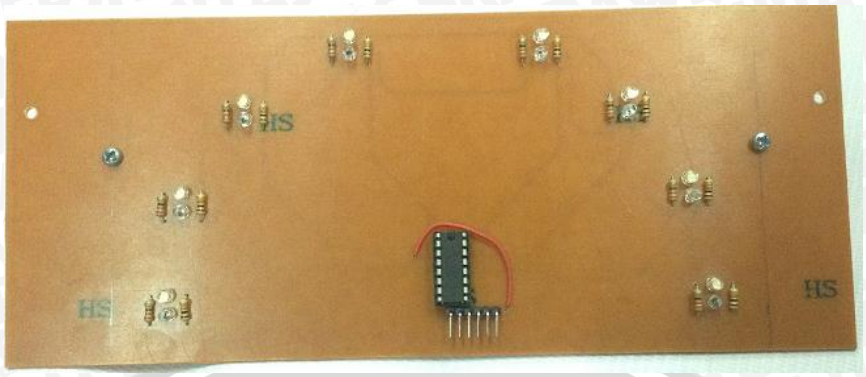
$$TS = 8{,}4\ s \quad (5.8)$$

5.5. Pengujian Sensor Bentuk Setengah Lingkaran

Sensor ini merupakan varian paling akhir pada skripsi ini. Sensor ini memerlukan ketelitian lebih pada saat membuatnya. Bentuk sensor ini merupakan modifikasi dari sensor bentuk lurus yang kemudian diubah posisinya hingga membentuk sensor bentuk setengah lingkaran. Berikut ini adalah hasil *layo*ut dan bentuk fisik sensor garis bentuk setengah lingkaran yang merupakan modifikasi dari bentuk sensor garis lurus. Hasil layout sensor bentuk lurus ditunjukkan dalam Gambar 5.14 sedangkan bentuk fisiknya ditunjukkan dalam Gambar 5.15



Gambar 5.14. Desain *Layout* Sensor Bentuk Setengah Lingkaran.



Gambar 5.15. Bentuk Fisik Sensor Bentuk Setengah Lingkaran

Pengujian ini bertujuan untuk mengetahui bagaimana performa robot ketika menggunakan sensor bentuk setengah lingkaran. Berikut hasil pengujian sensor bentuk setengah lingkaran pada posisi 0, 7 dan 14.

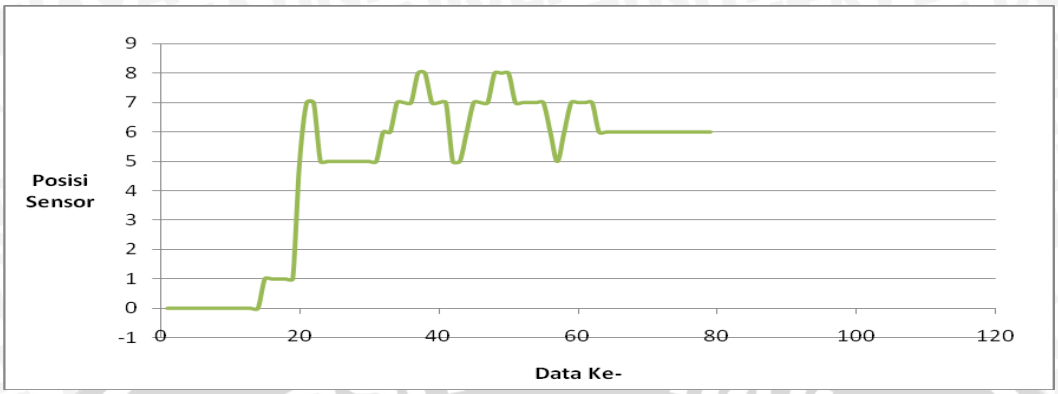
a. Hasil pengujian sensor pada posisi 0

Hasil pengujian sensor bentuk setengah lingkaran pada posisi 0 dapat dilihat dalam tabel 5.8

Tabel 5.8. Pengujian Sensor Bentuk Setengah Lingkaran Pada Posisi 0

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	0	21	7	41	7	61	7
2	0	22	7	42	5	62	7
3	0	23	5	43	5	63	6
4	0	24	5	44	6	64	6
5	0	25	5	45	7	65	6
6	0	26	5	46	7	66	6
7	0	27	5	47	7	67	6
8	0	28	5	48	8	68	6
9	0	29	5	49	8	69	6
10	0	30	5	50	8	70	6
11	0	31	5	51	7	71	6
12	0	32	6	52	7	72	6
13	0	33	6	53	7	73	6
14	0	34	7	54	7	74	6
15	1	35	7	55	7	75	6
16	1	36	7	56	6	76	6
17	1	37	8	57	5	77	6
18	1	38	8	58	6	78	6
19	1	39	7	59	7	79	6
20	5	40	7	60	7	80	6

Grafik respon pengujian sensor bentuk setengah lingkaran pada posisi 0 ditunjukkan dalam Gambar 5.16



Gambar 5.16. Grafik Respon Pengujian Sensor Bentuk Setengah Lingkaran Pada Posisi 0

Hasil pengujian sensor bentuk setengah lingkaran terhadap posisi 0 robot tidak dapat mencapai *set point* tetapi hanya terjadi osilasi.

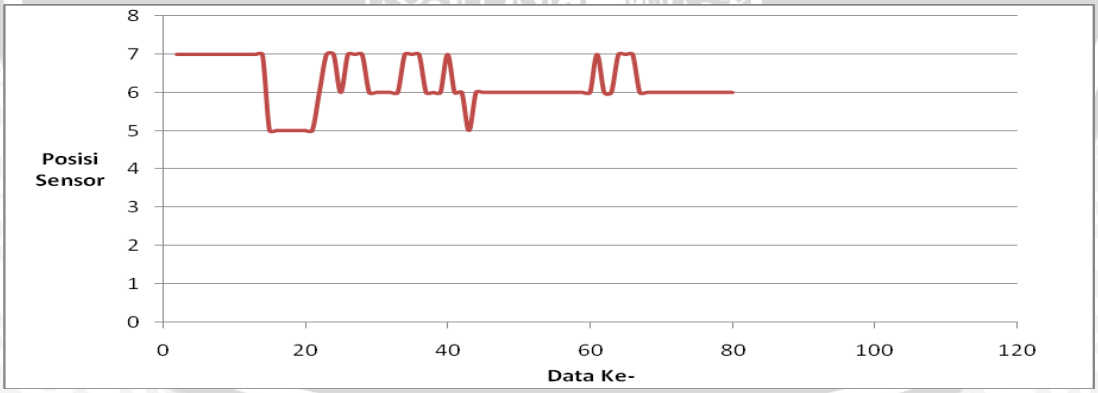
b. Hasil pengujian sensor pada posisi 7

Hasil pengujian sensor bentuk setengah lingkaran pada posisi 7 dapat dilihat dalam tabel 5.9

Tabel 5.9. Respon Pengujian Sensor Bentuk Setengah Lingkaran Pada Posisi 7

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	7	21	6	41	6	61	6
2	7	22	7	42	5	62	6
3	7	23	7	43	6	63	7
4	7	24	6	44	6	64	7
5	7	25	7	45	6	65	7
6	7	26	7	46	6	66	6
7	7	27	7	47	6	67	6
8	7	28	6	48	6	68	6
9	7	29	6	49	6	69	6
10	7	30	6	50	6	70	6
11	7	31	6	51	6	71	6
12	7	32	6	52	6	72	6
13	7	33	7	53	6	73	6
14	5	34	7	54	6	74	6
15	5	35	7	55	6	75	6
16	5	36	6	56	6	76	6
17	5	37	6	57	6	77	6
18	5	38	6	58	6	78	6
19	5	39	7	59	6	79	6
20	5	40	6	60	7	80	6

Grafik respon pengujian sensor bentuk setengah lingkaran pada posisi 7 ditunjukkan dalam Gambar 5.17



Gambar 5.17. Grafik Respon Pengujian Sensor Bentuk Setengah Lingkaran Pada Posisi 7

Hasil pengujian sensor bentuk setengah lingkaran terhadap posisi 7 robot tidak dapat mencapai *setpoint* tetapi hanya terjadi osilasi.

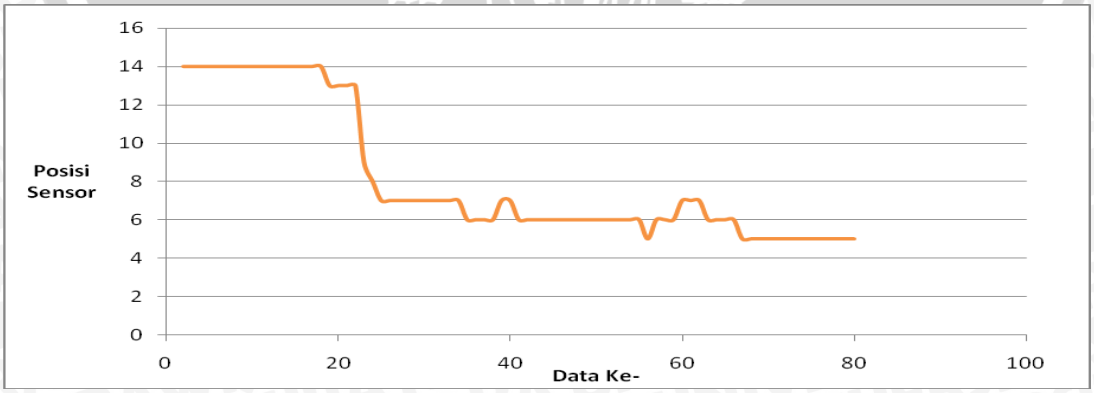
b. Hasil pengujian sensor pada posisi 14

Hasil pengujian sensor bentuk setengah lingkaran pada posisi 14 dapat dilihat dalam tabel 5.10

Tabel 5.10. Respon Pengujian Sensor Bentuk Setengah Lingkaran Pada Posisi 14

Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor	Data Ke-	Posisi Sensor
1	14	21	13	41	6	61	7
2	14	22	9	42	6	62	6
3	14	23	8	43	6	63	6
4	14	24	7	44	6	64	6
5	14	25	7	45	6	65	6
6	14	26	7	46	6	66	5
7	14	27	7	47	6	67	5
8	14	28	7	48	6	68	5
9	14	29	7	49	6	69	5
10	14	30	7	50	6	70	5
11	14	31	7	51	6	71	5
12	14	32	7	52	6	72	5
13	14	33	7	53	6	73	5
14	14	34	6	54	6	74	5
15	14	35	6	55	5	75	5
16	14	36	6	56	6	76	5
17	14	37	6	57	6	77	5
18	13	38	7	58	6	78	5
19	13	39	7	59	7	79	5
20	13	40	6	60	7	80	5

Grafik respon pengujian sensor bentuk setengah lingkaran pada posisi 14 ditunjukkan dalam Gambar 5.18



Gambar 5.18. Grafik Respon Pengujian Sensor Bentuk Setengah Lingkaran Pada Posisi 14

Hasil pengujian sensor bentuk setengah lingkaran terhadap posisi 14 robot tidak dapat mencapai *set point*.

Dari beberapa hasil pengujian dan pencuplikan data sensor diatas, maka diambil hasil dari respon yang terbaik tiap sensor. Sensor bentuk segitiga dapat mencapai *set point* pada pencuplikan data ke-33, sensor bentuk lurus (180^0) dapat mencapai *set point* pada pencuplikan data ke-39, dan sensor bentuk setengah lingkaran tidak dapat mencapai *set point*.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan pengujian dan analisis perbandingan posisi sensor garis pada robot *Management* Sampah dapat diambil kesimpulan, sebagai berikut :

1) Sensor bentuk lurus (180^0) dapat mencapai set point pada pencuplikan data ke- 39, menghasilkan s*ettling time* sebesar 7,8 s dan *overshoot* maksimum sebesar 28,57%.
2) Sensor bentuk segitiga dapat mencapai set point pada pencuplikan data ke-33, menghasilkan *settling time* sebesar 6,6 s dan *overshoot* maksimum sebesar 28,57%.
3) Sensor bentuk setengah lingkaran dapat mencapai kestabilan posisi sensor pada pencuplikan data ke-63, tidak dapat mencapai *set point* dan terjadi osilasi.
4) Dari ketiga desain sensor dapat diambil kesimpulan bahwa sensor bentuk segitiga mempunyai respon yang paling baik dari pada dua desain yang lainnya (lurus dan setengah lingkaran) yaitu pada saat pengujian sensor bentuk segitiga posisi 0 karena menghasilkan *settling time* dan maksimum *overshoot* yang lebih baik.

6.2 Saran

Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah:

1) Penulis menyarankan untuk memperbanyak variasi pada desain sensor garis.
2) Untuk memperoleh grafik respons pergerakan robot yang lebih baik disarankan menggunakan jumlah sensor *photodiode* yang lebih banyak.
3) Untuk mendapatkan respon yang lebih optimal disarankan memperkecil jarak antara sensor *photodiode* pada sensor garis.

DAFTAR PUSTAKA

Atmel. 2011. *ATmega16/ATmega16L Datasheet*. San Jose: Atmel Corporation. http://www.atmel.com/images/doc2466.pdf.

Atmel. 2011. *ATmega8/ATmega8L Datasheet*. San Jose: Atmel Corporation. http://www.atmel.com/images/doc2553.pdf.

Bagus, I.S. 2013. *Perancangan Robot Auto Line Follower yang Menerapkan Metode Osilasi Ziegler-Nichols Untuk Tuning Parameter PID pada Kontes Robot Indonesia*. Skripsi Teknik Elektro Universitas Brawijaya. Malang: Indonesia

Blocher, R. 2003. *Dasar Elektronika*. Yogyakarta: Andi.

Carrick, M. 2006. *GTK+ amd Glade3 GUI Programming Tutorial*. Jakarta: PT Elek Media Komputindo Kelompok Gramedia.

Kuswandi, Son. 2000. Kendali Cerdas *(Intelligent Control)*. EEPIS Press.

Nalwan, P. A. 2004. *Penggunaan dan Antarmuka Modul LCD M1632*. Jakarta: PT Elek Media Komputindo Kelompok Gramedia.

Nuraini, D. & Pangestu, S. 2011. *Tugas Akhir Robot Pengangkut Sampah Otomatis untuk Distribusi Sampah di dalam Gedung (Studi Kasus di Gedung Robotika ITS)*. Skripsi tidak dipublikasikan Surabaya: ITS.

Ogata, K. 1997. *Teknik Kontrol Automatik Jilid 1*. Jakarta: Penerbit Erlangga.

Rajshekar, P. & Malathi, M. 2011. *Power Efficient Multiplexer Using DLDFF Synchronus Counter*. Bangalore: ISVLSI.

McComb, G. 2011. *Robot Builder Bonanza, 4th Edition.* New York City: McGraw-Hill.

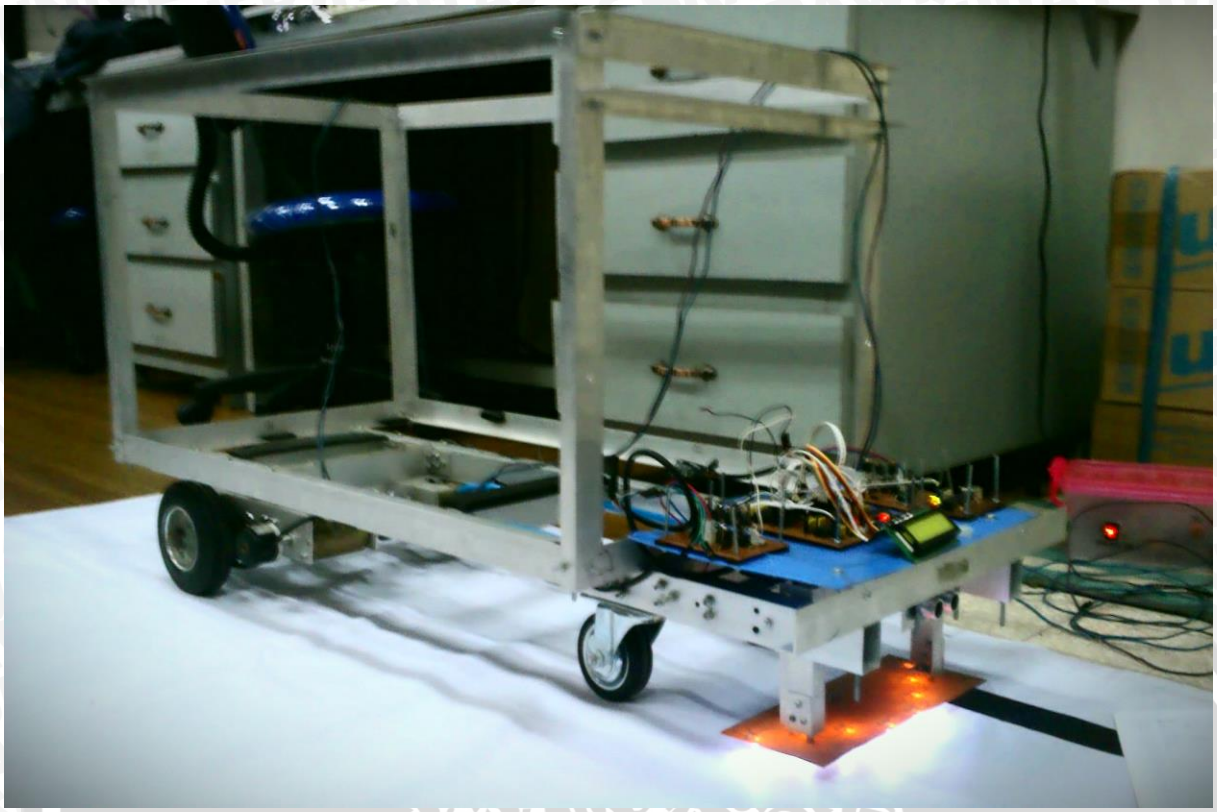
Rusmadi, Dedy. 2005. Aneka Rangkaian Elektronika Alarm dan Bel Listrik, Pioner Jaya: Bandung.

Susanto, Heri. 2002. *Robot Line Follower Berbasis Mikrokontroler AT89S51 Sebagai Media Pembelajaran Rancang Bangun Dan Unjuk Kerja Robot Beroda Otomatis*. Skripsi tidak diterbitkan: PTE. FT UNY

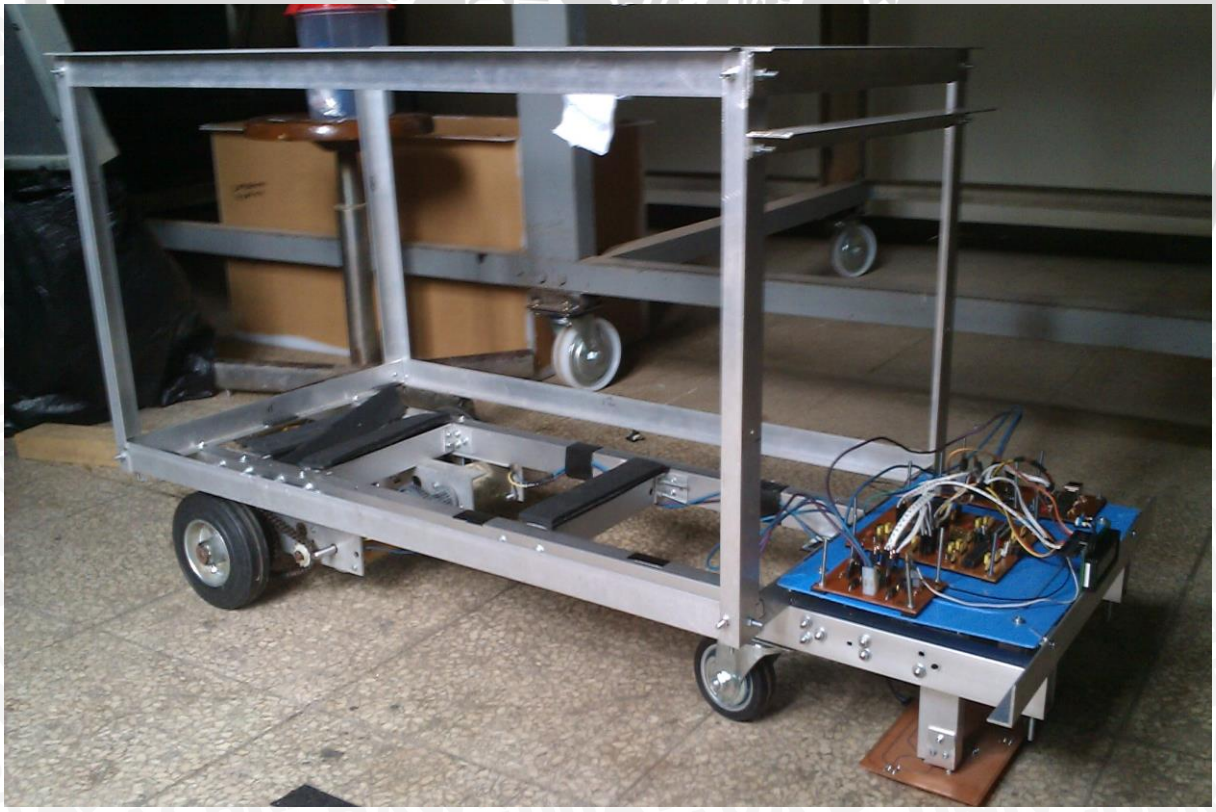
Siegwart, R. & Illah R. N. 2004. *Introduction to Autonomous Mobile Robots*. London: MIT Press.

LAMPIRAN I

Dokumentasi Alat

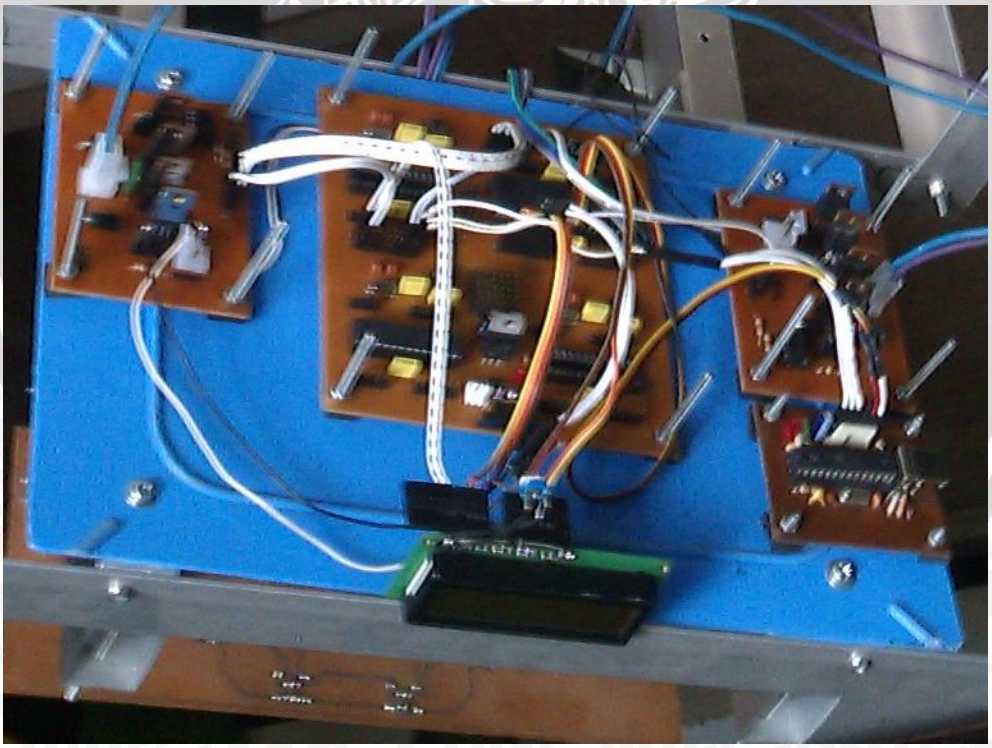


Gambar 1. Robot Saat Pengujian



Gambar 2. Robot Setelah Selesai Pembuatan

Gambar 3. Robot Tampak Perspektif



Gambar 4. Bagian Elektrik Robot

LAMPIRAN II

Listing Program

LISTING PROGRAM MIKROKONTROLER SLAVE SENSOR

```
#define ADC_VREF_TYPE 0x60

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <compat/twi.h>

#include "twi_slave.h"
// #include "SPI.h"

volatile unsigned char TWI_buf[4];     // Transceiver buffer. Set the size in the header file
volatile unsigned char TWI_msgSize  = 0;            // Number of bytes to be transmitted.
volatile unsigned char TWI_state    = 0xF8;  // State byte. Default set to
TWI_NO_STATE.

uint8_t i=0;
volatile unsigned char
data_sensor[8],a[8],tengah[8]={94,76,101,152,143,95,100,153},sensor,PV;

unsigned char read_sensor(unsigned char num);
unsigned char read_adc(unsigned char adc_input);

int main (void)
{
        _delay_ms(500);
        //SPI_SlaveInit();
        TWAR = 0xD1;// 0x08;
        TWCR = 0x45;
        sei();
        ADMUX=ADC_VREF_TYPE & 0xff;
        ADCSRA=0x87;
        DDRC |= _BV(0); DDRC |= _BV(1); DDRC |= _BV(2);


        while (1)
        {

                for(i=0;i<8;i++){
                        data_sensor[i] = read_sensor(i);
                        if (data_sensor[i]>tengah[i]) a[i]=0;
                else a[i]=1;
                }

                sensor=(
(a[7]*128)+(a[6]*64)+(a[5]*32)+(a[4]*16)+(a[3]*8)+(a[2]*4)+(a[1]*2)+(a[0]*1) );
                switch(sensor){
              case 0b11111110:       // ujung kanan
                  PV = 0;
```

```
                break;
            case 0b11111100:
                PV = 1;
                break;
            case 0b11111101:
                PV = 2;
                break;
            case 0b11111001:
                PV = 3;
                break;
                case 0b11111011:
                PV = 4;
                break;
            case 0b11110011:
                PV = 5;
                break;
                case 0b11110111:
                PV = 6;
                break;
            case 0b11100111:        // tengah
                PV = 7;
                break;
                case 0b11101111:
                PV = 8;
                break;
            case 0b11001111:
                PV = 9;
                break;
                case 0b11011111:
                PV = 10;
                break;
            case 0b10011111:
                PV = 11;
                break;
                case 0b10111111:
                PV = 12;
                break;
            case 0b00111111:
                PV = 13;
                break;
            case 0b01111111:        // ujung kiri
                PV = 14;
                break;
        }
            TWI_buf[0] = PV;


    }
}
/*
ISR(SPI_STC_vect){
    SPDR=PV;
```

```
}
*/
ISR(TWI_vect){
        static unsigned char TWI_bufPtr;

 switch (TWSR)
 {
   case TWI_STX_ADR_ACK:          // Own SLA+R has been received; ACK has been
returned
//    case TWI_STX_ADR_ACK_M_ARB_LOST: // Arbitration lost in SLA+R/W as
Master; own SLA+R has been received; ACK has been returned
     TWI_bufPtr   = 0;                     // Set buffer pointer to first data location
   case TWI_STX_DATA_ACK:         // Data byte in TWDR has been transmitted;
ACK has been received
     TWDR = TWI_buf[TWI_bufPtr++];
     TWCR = (1<<TWEN)|                     // TWI Interface enabled
          (1<<TWIE)|(1<<TWINT)|                // Enable TWI Interupt and clear the flag
to send byte
          (1<<TWEA)|(0<<TWSTA)|(0<<TWSTO)|          //
          (0<<TWWC);                       //
     break;
          case TWI_STX_DATA_NACK:       // Data byte in TWDR has been
transmitted; NACK has been received.
                              // I.e. this could be the end of the transmission.
     if (TWI_bufPtr == TWI_msgSize) // Have we transceived all expected data?
     {
//       TWI_statusReg.lastTransOK = TRUE;              // Set status bits to completed
successfully.
     }
     else                     // Master has sent a NACK before all data where sent.
     {
      TWI_state = TWSR;                    // Store TWI State as errormessage.
     }

     TWCR = (1<<TWEN)|                     // Enable TWI-interface and release TWI
pins
          (1<<TWIE)|(1<<TWINT)|                // Keep interrupt enabled and clear the
flag
          (1<<TWEA)|(0<<TWSTA)|(0<<TWSTO)|          // Answer on next address
match
          (0<<TWWC);                       //

     //TWI_busy = 0;   // Transmit is finished, we are not busy anymore
     break;
   case TWI_SRX_GEN_ACK:          // General call address has been received; ACK
has been returned
//    case TWI_SRX_GEN_ACK_M_ARB_LOST: // Arbitration lost in SLA+R/W as
Master; General call address has been received; ACK has been returned
//      TWI_statusReg.genAddressCall = TRUE;
   case TWI_SRX_ADR_ACK:          // Own SLA+W has been received ACK has been
returned
```

```
//    case TWI_SRX_ADR_ACK_M_ARB_LOST: // Arbitration lost in SLA+R/W as
Master; own SLA+W has been received; ACK has been returned
                                    // Dont need to clear
TWI_S_statusRegister.generalAddressCall due to that it is the default state.
//      TWI_statusReg.RxDataInBuf = TRUE;
      TWI_bufPtr   = 0;                         // Set buffer pointer to first data location

                                    // Reset the TWI Interupt to wait for a new event.
      TWCR = (1<<TWEN)|                         // TWI Interface enabled
            (1<<TWIE)|(1<<TWINT)|               // Enable TWI Interupt and clear the flag
to send byte
            (1<<TWEA)|(0<<TWSTA)|(0<<TWSTO)|    // Expect ACK on this
transmission
            (0<<TWWC);
      //TWI_busy = 1;

      break;
    case TWI_SRX_ADR_DATA_ACK:      // Previously addressed with own SLA+W;
data has been received; ACK has been returned
    case TWI_SRX_GEN_DATA_ACK:      // Previously addressed with general call;
data has been received; ACK has been returned
      TWI_buf[0]     = TWDR;//TWI_buf[TWI_bufPtr++]     = TWDR;
//      TWI_statusReg.lastTransOK = TRUE;           // Set flag transmission successfull.
                                    // Reset the TWI Interupt to wait for a new event.
      TWCR = (1<<TWEN)|                         // TWI Interface enabled
            (1<<TWIE)|(1<<TWINT)|               // Enable TWI Interupt and clear the flag
to send byte
            (1<<TWEA)|(0<<TWSTA)|(0<<TWSTO)|    // Send ACK after next
reception
            (0<<TWWC);                          //
      //TWI_busy = 1;
      break;
    case TWI_SRX_STOP_RESTART:      // A STOP condition or repeated START
condition has been received while still addressed as Slave
                                    // Enter not addressed mode and listen to address
match
      TWCR = (1<<TWEN)|                         // Enable TWI-interface and release TWI
pins
            (1<<TWIE)|(1<<TWINT)|               // Enable interrupt and clear the flag
            (1<<TWEA)|(0<<TWSTA)|(0<<TWSTO)|    // Wait for new address match
            (0<<TWWC);                          //

      //TWI_busy = 0;  // We are waiting for a new address match, so we are not busy

      break;
    case TWI_SRX_ADR_DATA_NACK:     // Previously addressed with own SLA+W;
data has been received; NOT ACK has been returned
    case TWI_SRX_GEN_DATA_NACK:     // Previously addressed with general call;
data has been received; NOT ACK has been returned
    case TWI_STX_DATA_ACK_LAST_BYTE: // Last data byte in TWDR has been
transmitted (TWEA = “0”); ACK has been received
```

```
//    case TWI_NO_STATE              // No relevant state information available; TWINT =
“0”
   case TWI_BUS_ERROR:        // Bus error due to an illegal START or STOP
condition
     TWI_state = TWSR;              //Store TWI State as errormessage, operation also
clears noErrors bit
     TWCR =   (1<<TWSTO)|(1<<TWINT);   //Recover from TWI_BUS_ERROR, this
will release the SDA and SCL pins thus enabling other devices to use the bus
     break;
   default:
     TWI_state = TWSR;                          // Store TWI State as errormessage,
operation also clears the Success bit.
     TWCR = (1<<TWEN)|                          // Enable TWI-interface and release TWI
pins
          (1<<TWIE)|(1<<TWINT)|                 // Keep interrupt enabled and clear the
flag
          (1<<TWEA)|(0<<TWSTA)|(0<<TWSTO)|      // Acknowledge on any new
requests.
          (0<<TWWC);                            //

     //TWI_busy = 0; // Unknown status, so we wait for a new address match that might be
something we can handle
          }

}
unsigned char read_adc(unsigned char adc_input)
{
          ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
          // Delay needed for the stabilization of the ADC input voltage
          _delay_us(50);
          // Start the AD conversion
          ADCSRA|=0x40;
          // Wait for the AD conversion to complete
          while ((ADCSRA & 0x10)==0);
          ADCSRA|=0x10;
          return ADCH;
}

void selectora(int a,int b,int c)
{
          if(a)
                    PORTC |= _BV(0);
          else
                    PORTC &= ~_BV(0);
          if(b)
                    PORTC |= _BV(1);
          else
                    PORTC &= ~_BV(1);
          if(c)
                    PORTC |= _BV(2);
          else
                    PORTC &= ~_BV(2);
```

```
}

unsigned char read_sensor(unsigned char num){

        unsigned char temp=0;

        switch(num)
   {
        case 6: selectora(1,1,1);       break;
        case 5: selectora(1,1,0);      break;
      case 7: selectora(1,0,1);       break;
      case 4: selectora(1,0,0);        break;
      case 0: selectora(0,1,1);        break;
      case 3: selectora(0,1,0);       break;
      case 2: selectora(0,0,1);        break;
      case 1: selectora(0,0,0);       break;
        }
        temp = read_adc(3);
        return temp;
}
```