# SISTEM PEMETAAN PADA MODEL ROBOT KONTES ROBOT CERDAS INDONESIA DIVISI EXPERT SINGLE

**SKRIPSI**

**JURUSAN TEKNIK ELEKTRO**

Diajukan untuk memenuhi persyaratan

memperoleh gelar Sarjana Teknik


Disusun oleh:

**EKO SETIAWAN**

**NIM. 0510630037-63**

**DEPARTEMEN PENDIDIKAN NASIONAL**

**UNIVERSITAS BRAWIJAYA**

**FAKULTAS TEKNIK**

**MALANG**

**2009**

**LEMBAR PERSETUJUAN**

# SISTEM PEMETAAN PADA MODEL ROBOT KONTES ROBOT CERDAS INDONESIA DIVISI EXPERT SINGLE

**SKRIPSI**
**JURUSAN TEKNIK ELEKTRO**

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik


Disusun oleh:
**EKO SETIAWAN**
**NIM. 0510630037-63**

**Telah diperiksa dan disetujui oleh :**

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| **Ir. Nanang Sulistyanto** | **Panca Mudjirahardjo, ST. MT.** |
| **NIP. 19700113 199403 1 002** | **NIP. 19700329 200012 1 001** |

**LEMBAR PENGESAHAN**

# SISTEM PEMETAAN PADA MODEL ROBOT KONTES ROBOT CERDAS INDONESIA DIVISI EXPERT SINGLE

**SKRIPSI**
**JURUSAN TEKNIK ELEKTRO**

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
**EKO SETIAWAN**
**NIM. 0510630037-63**

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 23 Desember 2009

**Dosen Penguji:**

| | |
|---|---|
| **Adharul Muttaqin, ST., MT** | **Ir. M. Julius St., MS** |
| **NIP. 19760121 200501 1 001** | **NIP. 19540720 198203 1 002** |

**M. Rif'an, ST., MT**

**NIP. 19710301 200012 1 001**

Mengetahui

Ketua Jurusan Teknik Elektro

**Rudy Yuwono, ST.,MSc.**

**NIP. 19710615 199802 1 003**

# ABSTRAK

**Eko Setiawan,** Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, November 2009, *Sistem Pemetaan pada Model Robot Kontes Robot Cerdas Indonesia Divisi Expert Single*, Dosen Pembimbing: Ir. Nanang Sulistiyanto dan Panca Mudjirahardjo, ST, MT.

Robot yang diikutsertakan dalam Kontes Robot Cerdas Indonesia harus mampu melakukan tugasnya dalam lapangan perlombaan tanpa adanya kendali dari luar lapangan. Pada Divisi Expert Single, lapangan pertandingan yang digunakan berubah-ubah pada tiap-tiap pertandingan sesuai dengan undian. Perubahan lapangan ini mempengaruhi pergerakan robot yang sudah direncanakan sebelum pertandingan. Informasi mengenai susunan lapangan pertandingan diperlukan agar pergerakan robot sesuai dengan lapangan.

Berdasarkan kondisi tersebut diciptakan Sistem Pemetaan pada Model Robot Kontes Robot Cerdas Indonesia Divisi Expert Single yang mampu membuat basis data pemetaan tentang kondisi lapangan pertandingan yang telah dilewati robot. Sistem akan memetakan kondisi lapangan pada posisi robot secara terus menerus. Posisi robot diperoleh melalui perhitungan putaran roda yang diperoleh dari sensor rotari. Hasil perhitungan disesuaikan dengan lapangan menggunakan sistem koreksi. Informasi basis data pemetaan yang terbentuk disimpan dalam *SD Card*. Proses perhitungan posisi, pembuatan dan penyimpanan basis data pemetaan dilakukan dengan menggunakan mikrokontroler Renesas R8C/13.

Hasil pengujian sensor rotari menunjukkan pengolah sinyal sensor rotari dapat merespon 3000 pulsa per detik yang setara dengan kecepatan 3,50 m/s pada model robot yang digunakan. Sensor rotari mempunyai kesalahan pembacaan maksimal sebesar 0,29%. Waktu yang diperlukan dalam melakukan satu siklus proses sebesar 0,05841 detik. Pengujian sistem pada dua kombinasi lapangan menunjukkan bahwa sistem dapat menghasilkan basis data pemetaan sesuai dengan kondisi lapangan.

**Kata kunci:** sistem pemetaan, robot, mikrokontroler renesas, *sd card*

# PENGANTAR

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul "Sistem Pemetaan pada Model Robot Kontes Robot Cerdas Indonesia Divisi Expert Single". Skripsi ini disusun sebagai persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Dalam menyelesaikan skripsi ini, banyak bantuan, bimbingan, dan dorongan yang diterima oleh penulis. Untuk itu, penulis mengucapkan terima kasih kepada:

- Orang tua dan adik-adik penulis yang selalu memberikan dukungan,
- Bapak Rudy Yuwono, ST., MSc sebagai Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Aziz Muslim, ST., MT., Ph.D sebagai Sekertaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Ir. M. Julius St, MS sebagai Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Ir. Nanang Sulistiyanto sebagai Dosen Pembimbing I atas segala bimbingan, pengarahan, gagasan, ide, saran serta motivasi yang telah diberikan,
- Bapak Panca Mudjirahardjo, ST, MT sebagai Dosen Pembimbing II atas segala bimbingan, pengarahan, saran, kritik, dan masukan yang telah diberikan,
- Staff Recording Jurusan Teknik Elektro,
- Teman - teman Streamline angkatan 2005,
- Teman - teman tim Robot Teknik Elektro Universitas Brawijaya,
- Teman - teman Elkamania,
- Rekan pengerjaan skripsi, Subhan, Arif, Agung, Sauki, Mas Asril
- Teman kos KRD19, kos Kertosentono 22

Penulis menyadari bahwa tugas akhir ini masih belum sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang. Penulis berharap, semoga tugas akhir ini bermanfaat bagi kita semua.

Malang, Desember 2009

Penulis

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Kontes Robot Cerdas Indonesia (KRCI) merupakan salah satu pertandingan robot tingkat nasional yang diadakan oleh Direktorat Jenderal Pendidikan Tinggi (Dirjen Dikti) setiap tahun. Pertandingan ini dibagi menjadi beberapa divisi yakni Divisi Senior Beroda, Senior Berkaki, Expert Single, dan Expert Battle. Masing-masing divisi mempunyai aturan, tugas, dan arena yang berbeda. Salah satu tugas yang terdapat pada tiap-tiap divisi adalah proses kembalinya robot ke posisi pertama kali diletakkan (*home*) setelah melakukan semua tugasnya.

Pertandingan robot ini dilaksanakan dalam suatu lapangan pertandingan. Lapangan yang digunakan merupakan simulasi rumah yang terdiri dari beberapa ruang. Pada beberapa divisi, lapangan pertandingan yang digunakan terdiri dari beberapa macam susunan lapangan. Penentuan susunan yang akan digunakan dalam pertandingan ditentukan berdasarkan undian yang dilakukan pada awal pertandingan. Divisi dengan jumlah kombinasi susunan lapangan terbanyak adalah Divisi Expert Single dengan 24 kombinasi lapangan bawah dan 12 kombinasi lapangan atas. Perbedaan antara satu kombinasi dengan kombinasi lainnya adalah posisi ruang dalam lapangan, posisi pintu masuk masing-masing ruang, posisi *home*, dan posisi objek yang meliputi lilin, boneka bayi, *furniture*, dan lain-lain.

Robot-robot dalam pertandingan KRCI merupakan robot yang bergerak tanpa adanya kendali dari luar lapangan. Strategi robot yang meliputi rute pergerakan disusun terlebih dahulu sebelum pertandingan dimulai. Ketidakpastian susunan lapangan pertandingan menuntut diciptakannya robot yang mampu beradaptasi dan melaksanakan tugasnya sesuai dengan lapangan pertandingan. Robot memerlukan suatu informasi tentang susunan lapangan untuk menentukan rute pergerakan dalam melaksanakan tugasnya, satunya adalah proses kembali ke *home*.

Permasalahan ini mendorong diciptakannya suatu sistem pemetaan yang mampu menyimpan informasi tentang kondisi arena yang telah dijelajahi robot. Informasi tersebut dapat digunakan untuk menentukan rute pergerakan robot. Dalam pembuatan sistem pemetaan, posisi robot dalam lapangan perlu diketahui sehingga informasi mengenai kondisi sekitar dapat disimpan dengan benar. Penentuan posisi robot dapat

dilakukan dengan menggunakan berbagai cara. Salah satu cara untuk dapat mengetahui posisi robot adalah dengan melakukan perhitungan berdasarkan pergerakan roda robot. Sensor yang mampu mendeteksi pergerakan roda robot adalah sensor rotari.

Sistem pemetaan yang telah dibangun harus disimpan dan dapat diakses sewaktu-waktu. Dalam penyimpanan data pemetaan diperlukan unit penyimpan yang mampu diakses oleh robot dan oleh komputer untuk diperiksa kebenarannya. *Secure Digital Card* (*SD Card*) merupakan salah satu unit penyimpan yang sering digunakan pada aplikasi peralatan elektronik. *SD Card* menyediakan *Serial Peripheral Interface* (SPI) sebagai protokol komunikasi sehingga memungkinkan mikrokontroler untuk mengaksesnya. Kemudahan penentuan kapasitas yang digunakan, keterjangkauan harga dan dimensi yang kecil memungkinkan *SD Card* untuk digunakan sebagai unit penyimpan data pada robot. Data pemetaan yang telah dibuat dapat disimpan dalam *SD Card*, sehingga apabila diperlukan oleh robot data tersebut dapat diakses kembali.

## 1.2 Rumusan Masalah

Rumusan masalah skripsi ini ditekankan pada:

1). Bagaimana membuat rangkaian sensor rotari.
2). Bagaimana membuat sistem penentu posisi dengan menggunakan mikrokontroler.
3). Bagaimana membuat sistem antarmuka mikrokontroler dengan *SD Card*.
4). Bagaimana membuat basis data sistem pemetaan pada *SD Card*.

## 1.3 Ruang Lingkup

Pada skripsi ini dibatasi oleh hal-hal sebagai berikut:

1). Model robot yang digunakan adalah model robot mobil sistem diferensial.
2). Informasi arena yang disimpan dalam sistem pemetaan adalah keberadaan dinding.
3). Pendeteksian keberadaan dinding dimodelkan dengan menggunakan tombol.
4). Sudut orientasi arah model robot merupakan kelipatan 90º.
5). Lapangan yang dipetakan memiliki permukaan datar.
6). Lapangan yang dipetakan merupakan lantai bawah lapangan KRCI Divisi Expert Single.
7). Selip antara roda dengan lantai selama pergerakan robot diabaikan.
8). Sistem yang digunakan pada *SD Card* adalah FAT16.

## 1.4 Tujuan

Tujuan skripsi ini adalah merancang dan membuat sistem yang mampu menghasilkan dan menyimpan informasi kondisi arena yang telah dijelajahi oleh model robot.

## 1.5 Sistematika Penulisan

Sistematika penulisan dalam laporan ini adalah:

**BAB I Pendahuluan**

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

**BAB II Tinjauan Pustaka**

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan alat.

**BAB III Metodologi Penelitian**

Berisi tentang metode yang digunakan dalam penyusunan penelitian.

**BAB IV Perancangan dan Pembuatan**

Membahas mengenai perancangan dan pembuatan alat yang meliputi spesifikasi, perancangan sistem, prinsip kerja dan realisasi alat.

**BAB V Pengujian dan Analisis**

Memuat hasil pengujian dan analisis terhadap alat yang telah direalisasikan.

**BAB VI Kesimpulan dan Saran**

Memuat kesimpulan dan saran-saran yang diperoleh dari penelitian.

# BAB II
# TINJAUAN PUSTAKA

Beberapa teori pendukung yang perlu dibahas dalam pembuatan sistem ini meliputi literatur mengenai Kontes Robot Cerdas Indonesia Divisi Expert Single, robot mobil sistem diferensial, sistem koordinat lokal dan global, mikrokontroler Renesas R8C/13, sensor rotari, *Secure Digital Card* (*SD Card*), sistem *File Allocation Table* 16 (FAT16), modul *Liquid Crystal Display* (LCD).

## 2.1 Kontes Robot Cerdas Indonesia Divisi Expert Single

Kontes Robot Cerdas Indonesia merupakan pertandingan robot tingkat nasional yang diadakan oleh Direktorat Jendral Pendidikan Tinggi (Dirjen DIKTI). Pertandingan ini klasifikasikan ke dalam beberapa divisi yakni Divisi Senior Beroda, Senior Berkaki, Expert Single, dan Expert Batle. Divisi Expert Single merupakan pertandingan robot dengan tugas sebagai berikut:

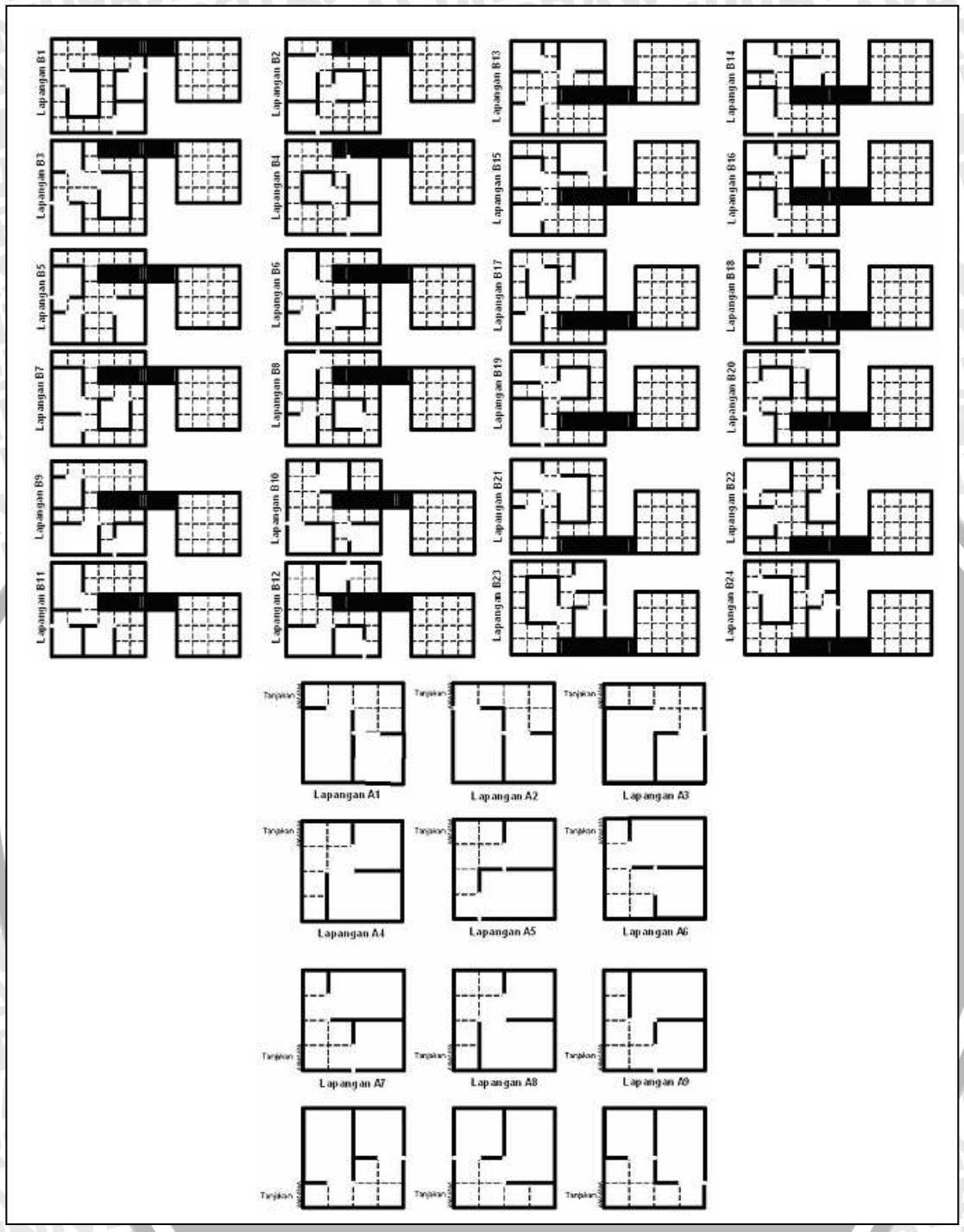
1) Naik tanjakan
2) Turun tanjakan
3) Menemukan bayi
4) Mengangkat bayi
5) Mematikan 1 lilin di lapangan atas
6) Mematikan lilin kesatu di lapangan bawah
7) Mematikan lilin kedua di lapangan bawah
8) Meletakkan bayi di *home*

Robot yang digunakan pada divisi ini memiliki panjang, lebar, dan tinggi maksimum 31 cm x 31 cm x 30 cm. Lapangan yang digunakan terdiri dari lapangan bawah dan atas. Lapangan bawah memiliki dimensi 300 cm x 300 cm x 30 cm. Sedangkan lapangan atas berdimensi 200 cm x 200 cm x 30 cm. Gambar 2.1 menunjukkan lapangan yang digunakan dalam pertandingan KRCI Expert Single. Lapangan atas dan bawah tersusun dari blok-blok persegi dengan ukuran 50 cm x 50 cm.


Gambar 2.1. Arena KRCI Divisi Expert Single

Sumber: DIKTI, 2008.

Lapangan pertandingan mempunyai dua jenis ruang yang digunakan. Ruang pertama mempunyai dimensi 150 cm x 100 cm dan ruang kedua berdimensi 100 cm x 100 cm. Objek lapangan yang meliputi lilin, boneka bayi, *furniture*, dan lain-lain diletakkan dalam salah satu ruang sesuai dengan undian. Posisi ruang dan tangga pada lapangan atas dan bawah dapat berubah-ubah sesuai undian yang dilakukan pada setiap awal pertandingan. Berdasarkan peraturan KRCI Divisi Expert Single, konfigurasi lapangan untuk lapangan atas berjumlah 12 konfigurasi dan lapangan bawah berjumlah 24 konfigurasi. Gambar 2.2 menunjukkan konfigurasi lapangan lomba.




Gambar 2.2 Konfigurasi lapangan bawah dan atas

Sumber: DIKTI, 2008.

## 2.2 Robot Mobil Sistem Diferensial

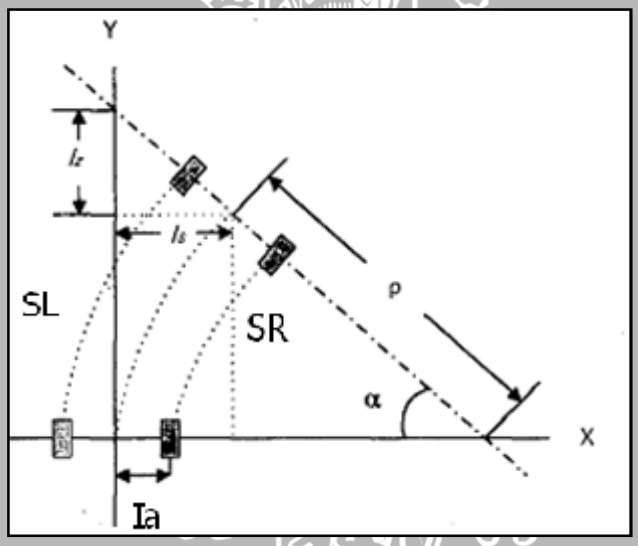
Robot mobil sistem diferensial merupakan robot mobil yang bergerak dengan menggunakan dua buah roda yang dapat digerakkan secara terpisah. Roda diletakkan pada kedua sisi kanan dan kiri robot dalam. Pada model robot ini biasanya dilengkapi dengan roda pasif tambahan yang berfungsi sebagai penyangga robot agar tidak jatuh.

Robot dapat merubah arah pergerakannya dengan mengatur putaran pada masing-masing roda. Oleh karena itu pada robot ini tidak diperlukan pengendali arah gerakan tambahan.

Posisi robot dapat ditentukan dengan memantau putaran tiap-tiap roda. Apabila putaran roda sisi kanan dan sisi kiri sama, maka robot akan bergerak lurus. Apabila besarnya putaran roda sisi kanan dan kiri robot tidak sama, maka robot akan menghasilkan gerak melingkar sesuai dengan ilustrasi dalam Gambar 2.3. Jarak tempuh roda kanan dan kiri diketahui sebesar $S_R$ dan $S_L$. Besarnya sudut α yang terbentuk dapat ditentukan dengan menggunakan persamaan (2-1). Jari-jari ρ dari gerak melingkar dapat ditentukan dengan menggunakan persamaan (2-2).

$$\alpha = \frac{\Delta S_L - \Delta S_R}{2.I_a} \quad (2\text{-}1)$$

$$\rho = \frac{I_a(S_L + S_R)}{S_L - S_R} \quad (2\text{-}2)$$




Gambar 2.3. Pergerakan robot dengan putaran roda berbeda

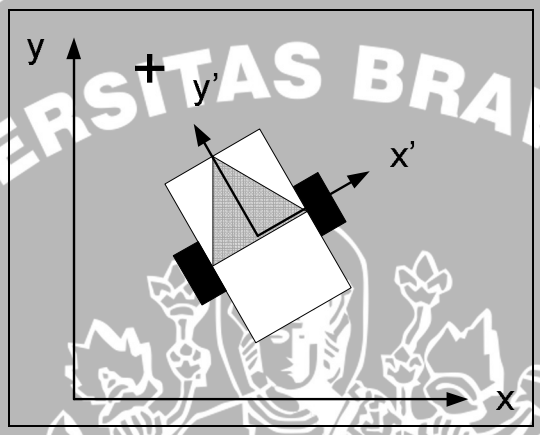
Sumber: Harashima,1999.

## 2.3 Sistem Koordinat Lokal dan Global

Sistem koordinat dapat digunakan untuk menentukan posisi objek. Sistem koordinat memberikan informasi tentang posisi objek terhadap posisi tertentu. Dalam aplikasi robotika, sistem koordinat dapat dibedakan menjadi sistem koordinat lokal dan global. Sistem koordinal lokal memberikan informasi tentang posisi robot terhadap posisi sebelumnya. Sistem koordinat global memberikan informasi posisi robot terhadap posisi awal atau pada titik yang telah ditentukan sebagai titik nol.

Sistem transformasi diperlukan untuk mendapatkan koordinat global pada posisi tersebut berdasar informasi nilai koordinat lokal. Gambar 2.4 menunjukkan koordinat lokal dan global sebuah robot. Sumbu x' dan y' merupakan sumbu dari koordinat lokal dan sumbu x dan y adalah sumbu koordinat global. Sebuah robot dimisalkan mempunyai koordinat global [$r_x$, $r_y$] dan mempunyai arah global φ. Nilai koordinat global [$o_x$, $o_y$] dari suatu posisi yang berada pada koordinat lokal [$o_x$', $o_y$'] terhadap robot dapat ditentukan dengan menggunakan transformasi dalam persamaan (2-3).

$$[o_x, o_y] = \text{Trans}(r_x, r_y) \,.\, \text{Rot}(\varphi) \,.\, [o_x', o_y'] \qquad (2\text{-}3)$$




Gambar 2.4. Sistem koordinat global dan lokal.
Sumber: Braunl, 2006.

Tranformasi sistem koordinat dalam persamaan (2-3) dapat dihitung dengan menggunakan matrik koordinat *homogeneous* yang telah disederhanakan. Koordinat *homogeneous* yang ditemukan oleh Mobius tahun 1827 biasa digunakan pada aplikasi lengan robot dalam menentukan posisi. Dengan menyederhanakan bentuk matrik transformasi tiga sumbu ke dalam dua sumbu, maka persamaan (2-3) dapat dikerjakan dengan menggunakan persamaan matrik (2-4).

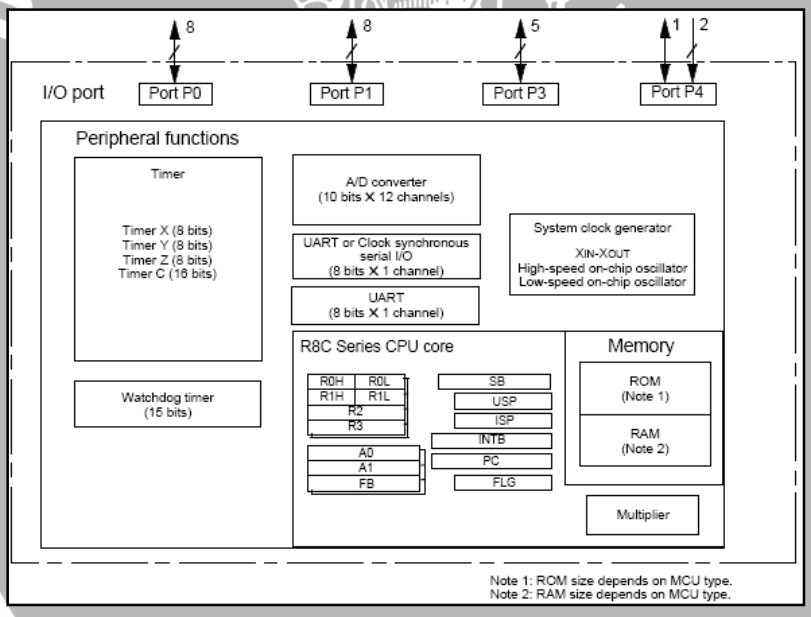
$$\begin{bmatrix} o_x \\ o_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & r_x \\ 0 & 1 & r_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} o_x' \\ o_y' \\ 1 \end{bmatrix} \qquad (2\text{-}4)$$

## 2.4 Mikrokontroler Renesas R8C/13

Mikrokontroler R8C/13 merupakan MCU 16-bit produksi Renesas. Dikatakan 16 bit, karena MCU ini memiliki *core* M16C yang berukuran 16-bit. Dikatakan R8C karena MCU ini memiliki lebar jalur data sebesar 8-bit. Dengan demikian untuk mengakses data berukuran 16-bit, R8C/13 harus mengakses memori sebanyak 2 kali.

Gambar 2.5 menunjukkan diagram blok mikrokontroler R8C/13. Fitur-fitur yang ada pada R8C/13 adalah sebagai berikut:

- ROM program 16 kB
- RAM 1 KB
- *On-chip oscillator*
- *Watchdog timer*
- Port I/O 22 buah
- Antarmuka serial 2 buah
- *Clock stop detect*
- *Power-on Reset* dan *Low Voltage Detect*
- ADC dengan resolusi 10-bit sebanyak 12 kanal
- Timer 8-bit 3 buah dan timer 16-bit 1 buah
- *In circuit programming* dan *debugging*



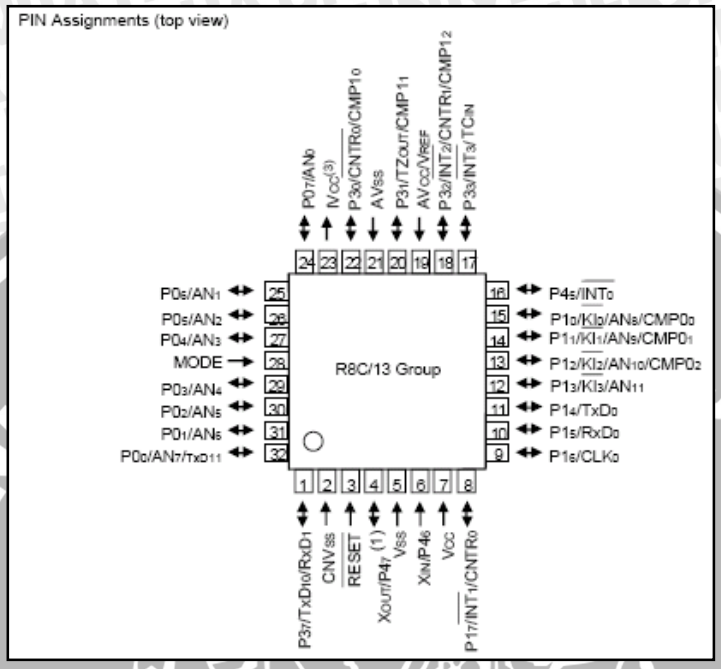

Gambar 2.5. Diagram blok mikrokontroler R8C/13

Sumber: Renesas, 2005.

### 2.4.1 *Programmable Input Output*

*Programable input output* merupakan salah satu fitur yang disediakan oleh mikrokontroler R8C/13. Mikrokontroler ini terdiri dari 22 buah jalur masukan keluaran yang dapat diatur sesuai dengan kebutuhan. Gambar 2.6 menunjukkan susunan pin R8C/13. Jalur masukan keluaran dikelompokkan ke dalam empat jenis port. Port merupakan kumpulan jalur I/O (input/output) yang diatur dengan menggunakan register

yang sama. R8C terdiri dari empat jenis port, yakni Port 0 yang terdiri dari $P_{00}$-$P_{07}$, Port 1 yang terdiri dari $P_{10}$-$P_{17}$, Port 3 yang terdiri dari $P_{30}$-$P_{33}$, $P_{37}$, dan Port 4 yang terdiri dari $P_{45}$. Pada R8C/13 juga terdapat dua jalur tambahan yakni $P_{46}$ dan $P_{47}$ yang dapat digunakan sebagai jalur masukan.




Gambar 2.6. Susunan pin mikrokontroler R8C/13

Sumber: Renesas, 2005.

Masing-masing jalur dapat diatur sebagai masukan atau keluaran dengan mengatur register delapan bit *Pin Direction* (PD). Tiap bit pada register ini merupakan perwakilan dari tiap jalur pada port yang bersangkutan. Pemberian logika 1 pada register menjadikan jalur berfungsi sebagai jalur keluaran. Sedangkan logika 0 akan mengatur jalur sebagai jalur keluaran. Bit yang tidak digunakan dalam satu register apabila ditulis tidak akan memberikan perubahan. Pada register PD0 terdapat proteksi yang harus dimatikan apabila ingin merubah nilai register PD0.

*Programable input output* juga dilengkapi pull-up internal yang dapat diaktifkan. Pengaturan pull-up internal dilakukan dengan mengatur *Pull-up control Register* (PUR). Pull-up pada port 0-3 diatur dengan menggunakan register PUR0 dan pull-up pada port 4 diatur dengan register PUR1. Pull-up internal pada masing-masing jalur diaktifkan dengan memberi logika 1.

Data keluaran pada masing-masing jalur dapat ditentukan dengan mengisi register Port (P). Logika masukan pada masing-masing jalur dapat diketahui dengan

membaca isi register Port. Tiap register P0, P1, P3, dan P4 merupakan perwakilan tiap port dan tiap bit dari masing-masing register Port merupakan perwakilan dari tiap jalur.

### 2.4.2 Timer

Mikrokontroler R8C/13 dilengkapi dengan tiga fitur timer 8 bit yakni timer X, Y, Z dan satu timer 16 bit yakni timer C. Masing-masing timer dilengkapi dengan *prescaler* yang berfungsi sebagai penentu frekuensi dari sumber *clock*. Setiap timer pada R8C/13 dapat bekerja mandiri tanpa terpengaruh timer lainnya. Sumber *clock* pada masing-masing timer berfungsi sebagai pemicu proses penghitungan waktu pada timer.

Masing-masing timer dapat beroperasi dalam beberapa fungsi. Tabel 2.1 menunjukkan perbandingan timer pada mikrokontroler R8C/13. Nilai frekuensi kerja timer dapat diatur dengan cara mengganti nilai yang terdapat dalam register *Timer Count Source Setting* (TCSS). Nilai dari timer akan ditempatkan pada register PRE dan T pada masing-masing timer.

### 2.4.3 Antarmuka Serial

Antarmuka serial R8C/13 terdiri dari dua buah yakni UART0 dan UART1. Masing-masing antarmuka serial mempunyai *clock* tersendiri sehingga dapat bekerja secara terpisah. UART0 dapat diatur dalam dua *mode* yakni sinkron dan asinkron. UART1 hanya dapat bekerja pada mode asinkron.

Mode sinkron pada UART0 digunakan untuk mengirimkan data dengan menggunakan *clock* dari *master*. Gambar 2.7 menunjukkan diagram blok antarmuka serial yang terdapat pada R8C/13. Mode sinkron dapat dimanfaatkan sebagai komunikasi *Serial Peripheral Interface* (SPI). Komunikasi serial mode sinkron dapat digunakan untuk komunikasi SPI dengan cara mengatur register agar sinyal yang akan dikirimkan sama dengan sinyal pada SPI.

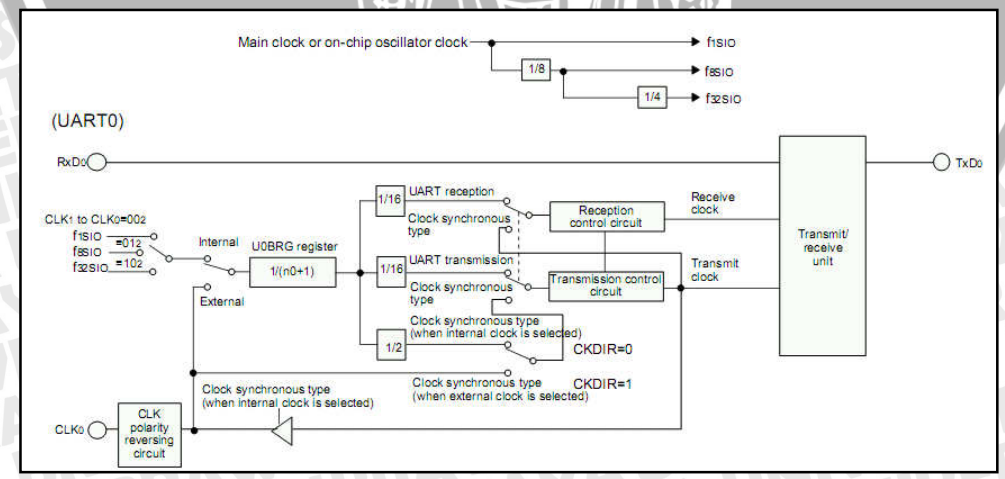
Register U0C0 bit UFORM digunakan untuk menentukan bit yang dikirim pertama. Bit CKPOL digunakan untuk mengatur data yang akan dikirim pada *clock* tepi naik atau tepi turun. Nilai frekuensi yang digunakan *clock* dapat diatur dengan mengganti nilai yang terdapat pada register U0C0 bit CK0 dan CK1. Data akan dikirim melalui pin TX dan akan diterima melalui pin RX. Pin TX dan RX tersebut memiliki fungsi sama dengan pin MOSI dan MISO pada mode SPI.

Tabel 2.1. Perbandingan timer pada R8C/13.

| Item | | Timer X | Timer Y | Timer Z | Timer C |
|---|---|---|---|---|---|
| Configuration | | 8-bit timer with 8-bit prescaler | 8-bit timer with 8-bit prescaler | 8-bit timer with 8-bit prescaler | 16-bit free-run timer |
| Count | | Down | Down | Down | Up |
| Count source | | •f1<br>•f2<br>•f8<br>•f32 | •f1<br>•f8<br>•fRING<br>•Input from CNTR1 pin | •f1<br>•f2<br>•f8<br>•Timer Y underflow | •f1<br>•f8<br>•f32<br>•fRING-fast |
| Function | Timer mode | provided | provided | provided | not provided |
| | Pulse output mode | provided | not provided | not provided | not provided |
| | Event counter mode | provided | provided[1] | not provided | not provided |
| | Pulse width measurement mode | provided | not provided | not provided | not provided |
| | Pulse period measurement mode | provided | not provided | not provided | not provided |
| | Programmable waveform generation mode | not provided | provided | provided | not provided |
| | Programmable one-shot generation mode | not provided | not provided | provided | not provided |
| | Programmable wait one-shot generation mode | not provided | not provided | provided | not provided |
| | Input capture mode | not provided | not provided | not provided | provided |
| | Output compare mode | not provided | not provided | not provided | provided |
| Input pin | | CNTR0 | CNTR1 | $\overline{INT0}$ | TCIN |
| Output pin | | CNTR0<br>$\overline{CNTR0}$ | CNTR1 | TZOUT | CMP00 to CMP02<br>CMP10 to CMP12 |
| Related interrupt | | Timer X int<br>$\overline{INT1}$ int | Timer Y int<br>$\overline{INT2}$ int | Timer Z int<br>$\overline{INT0}$ int | Timer C int<br>$\overline{INT3}$ int<br>compare 0 int<br>compare 1 int |
| Timer stop | | provided | provided | provided | provided |

Note: Select the input from the CNTR1 pin as a count source of timer mode.

Sumber: Renesas, 2005.




Gambar 2.7. Diagram blok antarmuka serial R8C/13

Sumber: Renesas, 2005.

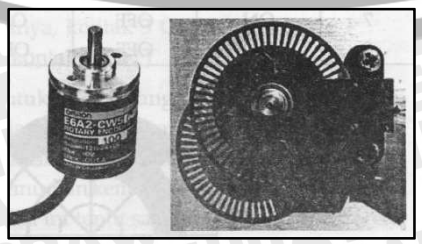
## *2.5* Sensor *Rotary Encoder*

*Rotary encoder* atau yang dikenal dengan *shaft encoder* adalah perangkat elektro-mekanikal yang digunakan untuk mengkonversi sudut dari perputaran poros atau roda ke dalam kode digital. Komponen ini biasa digunakan dalam bidang robotika, perangkat komputer dan perangkat elektronik lainnya. *Rotary encoder* dibedakan menjadi dua jenis yakni *rotary encoder* absolut dan relatif.

*Rotary encoder* absolut merupakan jenis sensor yang mampu menghasikan kode digital yang unik untuk masing-masing sudut poros. Pada prinsipnya sensor ini tersusun atas plat baja dan kontak. Plat baja dipotong dan disusun dengan pola tertentu kemudian ditempelkan pada suatu poros. Plat baja dan kontak ini secara prinsip kerja menyerupai saklar dalam kondisi ON apabila keduanya saling bersentuhan dan OFF apabila terpisah. Plat baja dan kontak diatur sedemikian rupa sehingga menghasilkan kondisi yang berbeda untuk tiap-tiap sudut poros. Bentuk fisik dari *rotary encoder* absolut ditunjukkan dalam Gambar 2.8.


Gambar 2.8. Rotary encoder absolut
Sumber: Sigit, 2007.

*Rotary encoder* relatif tidak dapat mengukur posisi sudut poros melainkan hanya mengukur perubahan sudut poros terhadap posisi sudut sebelumnya. Rotari ini digunakan ketika metode rotari absolut tidak dapat digunakan. Secara prinsip sistem ini terdiri dari piringan yang dipasang pada poros dan sensor optik. Sistem ini menggunakan metode saklar optik, misal photodiode, untuk menghasilkan pulsa listrik yang digunakan sebagai masukan bagi rangkaian kontrol elektronika. Gambar 2.9 di bawah ini menunjukkan bentuk fisik *rotary encoder* relatif.



Gambar 2.9. Rotary encoder relatif
Sumber: Sigit, 2007.

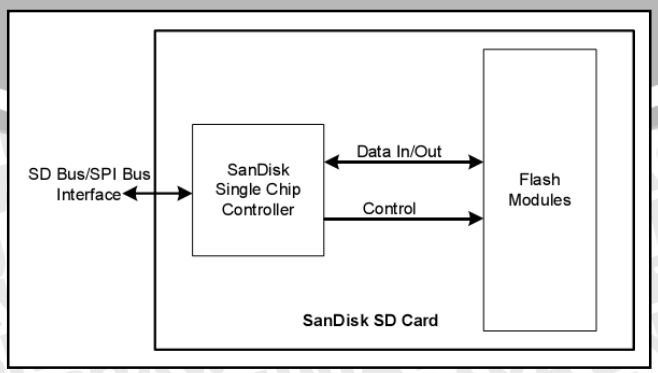
Salah satu kekurangan sistem relatif adalah tidak dapat menentukan arah putaran poros. Agar dapat mengetahui arah putaran poros, pada sistem harus ditambahkan dua buah sensor optik yang dipasang pada sudut berbeda. Pembacaan arah putaran dan jumlah putaran dilakukan dengan membaca dan membandingkan sinyal keluaran dari kedua buah sensor. Tipe *rotary encoder* ini dikenal dengan *quadrature encoder*.

## 2.6 *Secure Digital Card* (*SD Card*)

*Secure Digital Card* merupakan kartu memori yang didesain untuk memberikan kemudahan dan kapasitas yang relatif besar pada perangkat-perangkat elektronik. *SD Card* memiliki dimensi 32 mm x 24 mm x 2,25 mm dengan berat 2 gram. Komunikasi *SD card* menggunakan sembilan pin yang terdiri atas pin *clock*, *command*, empat pin data dan tiga pin catu daya yang dirancang untuk beroperasi pada tegangan rendah. Tegangan kerja dari SD Card berkisar antara 2 - 3,6 V. Bentuk fisik SD Card ditunjukkan dalam Gambar 2.10.


Gambar 2.10. Bentuk fisik *SD Card*
Sumber: Transcend, 2009.

*SD card* saat ini telah berisi lebih dari 1024 MB memori yang dirancang untuk aplikasi penyimpan data. *SD card* sudah berisi kontroler yang mengatur protokol antarmuka, algoritma pengamanan, algoritma penyimpanan data, algoritma pencarian kembali (*Error Correction Code*/ECC), manajemen daya, dan kontrol *clock*. Diagram blok umum *SD card* dapat dilihat dalam Gambar 2.11.




Gambar 2.11. Diagram blok umum *SD Card*
Sumber: SanDisk, 2003

### 2.6.1 Mode *Serial Peripheral Interface*

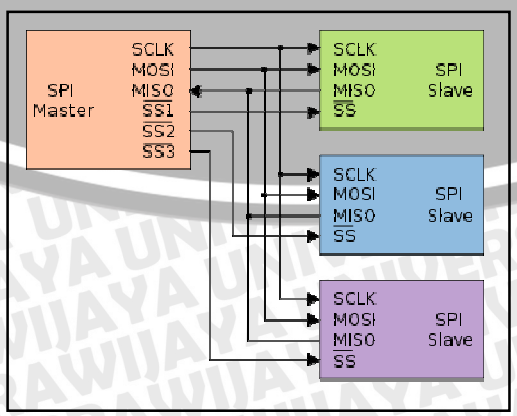
*SD Card* dapat diakses dengan menggunakan dua mode komunikasi data yakni protokol komunikasi *SD Card* dan *Serial Peripheral Interface* (SPI). Konfigurasi pin yang digunakan kedua buah komunikasi ini berbeda. Tabel 2.2 menunjukkan fungsi dari masing-masing pin untuk tiap komunikasi.

Tabel 2.2. Konfigurasi pin *SD Card*

| Pin No. | SD Mode | | | SPI Mode | | |
|---|---|---|---|---|---|---|
| | Name | Type | Description | Name | Type | Description |
| 1 | CD/DAT | I/O/PP[3] | Card Detect/Data Line [Bit3] | CS | I | Chip Select (neg true) |
| 2 | CMD | PP | Command/Response | DI | I | Data In |
| 3 | $V_{SS1}$ | S | Supply voltage ground | VSS | S | Supply voltage ground |
| 4 | $V_{DD}$ | S | Supply voltage | VDD | S | Supply voltage |
| 5 | CLK | I | Clock | SCLK | I | Clock |
| 6 | $V_{SS2}$ | S | Supply voltage ground | VSS2 | S | Supply voltage ground |
| 7 | DAT0 | I/O/PP | Data Line [Bit0] | DO | O/PP | Data Out |
| 8 | DAT1 | I/O/PP | Data Line [Bit1] | RSV | | |
| 9 | DAT2 | I/O/PP | Data Line [Bit2] | RSV | | |

Sumber: Transcend, 2009

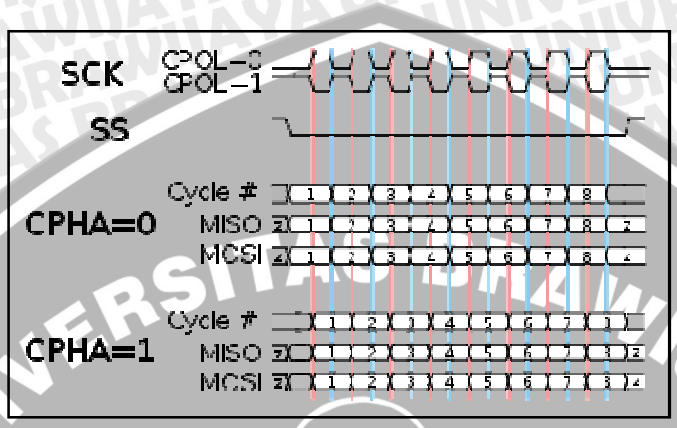
Mode SPI adalah protokol komunikasi sekunder yang ada dalam *SD Card*. SPI merupakan komunikasi serial *synchronous full duplex* yang dipatenkan oleh Motorola. Perangkat yang berkomunikasi menggunakan SPI dibedakan menjadi dua yakni *master* dan *slave*. SPI menggunakan empat buah jalur yakni SCLK, MOSI(Data In), MISO(Data Out), dan SS(CS). SCLK merupakan jalur *clock* yang berasal dari *master*. *Clock* pada SCLK digunakan sebagai acuan dalam pengiriman data. MOSI merupakan jalur data dari *master* menuju ke *slave*. MISO adalah jalur data dari *slave* menuju ke *master*. SS berfungsi sebagai jalur aktivasi *slave*. *Slave* akan aktif apabila SS berlogika rendah (*low*). Susunan jalur pada SPI ditunjukkan dalam Gambar 2.12.




Gambar 2.12. Susunan jalur SPI

Sumber: www.wikipedia.org

Komunikasi dimulai pada saat jalur SS berlogika *low*. Perpindahan data dimulai pada saat *clock* pada jalur SCLK dibangkitkan. Perpindahan data yang terjadi adalah perpindahan data dari *master* menuju ke *slave* pada jalur MOSI dan dari *slave* menuju ke *master* pada jalur MISO. Perpindahan data akan terus berjalan secara serial hingga jalur SS berlogika *high*. Diagram pewaktuan SPI ditunjukkan dalam Gambar 2.13.




Gambar 2.13. *Timing diagram* SPI

Sumber: www.wikipedia.org

SD Card berfungsi sebagai *slave* ketika mode SPI diaktifkan. Mode SPI diaktifkan dengan memberikan logika *low* pada pin CS. Panjang paket perintah (*command frame*) dari *master* ke *SD Card* sebesar enam byte. Semua komunikasi antara *master* dan *SD card* diatur oleh *master* yang umumnya berupa mikrokontroler atau mikroprosesor. Byte CRC bersifat opsional dalam mode SPI, tapi harus diisi untuk membentuk paket perintah enam byte. Tabel 2.3 menunjukkan tabel *command frame*. Setiap perintah di dalam komunikasi SPI diekpresikan dalam bentuk CMD<x>, dengan <x> adalah nomor indeks perintah yang bernilai 0 sampai 63. Kode biner setiap perintah merupakan nilai biner dari nomer indeks perintah. Sebagai contoh, kode biner untuk CMD0 adalah ‘000000’ dan kode biner untuk CMD39 adalah ‘100111’. Ketika satu paket perintah dikirimkan ke *SD Card*, maka akan ada respon (R1, R2, atau R3) yang dikirim balik ke *master*. Jenis respon yang dikirimkan oleh *SD Card* tergantung dari perintah yang diterima *SD Card*.

Tabel 2.3. *Command frame* SD card

| Byte 1 | | | | Bytes 2—5 | | Byte 6 | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 0 | 31 | 0 | 7 | 0 |
| 0 | 1 | Command | | Command Argument | | CRC | 1 |

Sumber : SanDisk, 2003

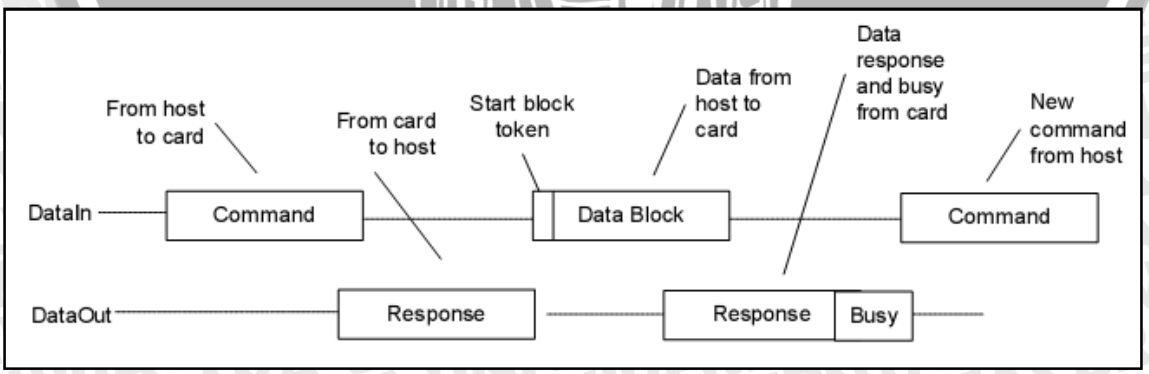
### 2.6.2 Inisialisasi Awal

*SD Card* berada dalam kondisi *idle* saat pertama kali diberi catu. SD Card memerlukan waktu guna mempersiapkan kondisi *idle*. Pada kondisi ini, *master* harus memberikan *clock* sedikitnya 74 *clock* dan jalur data berlogika *high*. *Master* dapat memeriksa SD Card apakah masih dalam kondisi mempersiapkan atau sudah selesai dengan mengirimkan perintah CMD0. *SD Card* akan memberikan balasan respon R1 yang berisi informasi mengenai kondisi *SD Card*. Apabila respon R1 bernilai 0x01 maka proses persiapan sudah selesai dan *SD Card* berada dalam kondisi *idle*.

Pada kondisi *idle*, SD Card hanya dapat mengenali perintah CMD0, CMD1, dan CMD58. Agar dapat mengakses data, SD Card harus diinisialisasi terlebih dahulu dengan perintah CMD1. Untuk mengetahui akhir inisialisasi, *master* harus melakukan cek terhadap respon R1. Inisialisasi sudah selesai jika respon R1 bernilai 0x00 (bit *In-Idle-State* bernilai 0). Setelah inisialisasi, operasi baca/tulis dalam mode SPI dapat dilakukan.

### 2.6.3 Operasi Baca Tulis

Memori pada SD Card dapat diakses blok per blok (*single block mode*) atau beberapa blok sekaligus (*multiple block mode*). Satu blok memori setara dengan 512 byte. Operasi tulis blok per blok dilakukan dengan mengirimkan perintah CMD24. *SD Card* akan menunggu data blok yang dikirimkan oleh *master*. Setelah semua data diterima, *SD Card* akan memberikan data respon ke *master*. Diagram alir operasi tulis per blok ditunjukkan dalam Gambar 2.14.

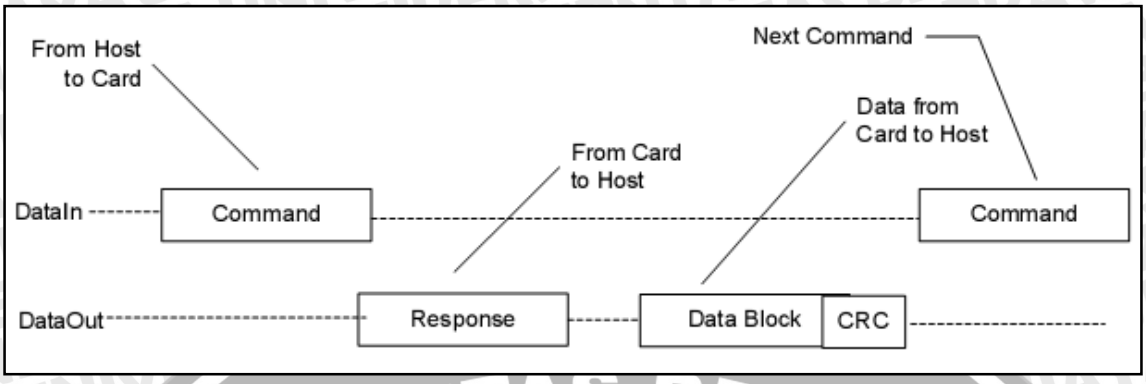



Gambar 2.14. Operasi tulis per blok

Sumber: SanDisk, 2003

Operasi baca per blok dilakukan dengan mengirimkan perintah CMD17. Setelah menerima perintah, *SD Card* akan mengirimkan respon yang diikuti dengan data blok. *Master* dapat mulai membaca data yang terkirim dengan memeriksa *start block token*

yang dikirimkan pertama kali sebelum data blok dikirim. Diagram alir operasi baca per blok ditunjukkan dalam Gambar 2.15.



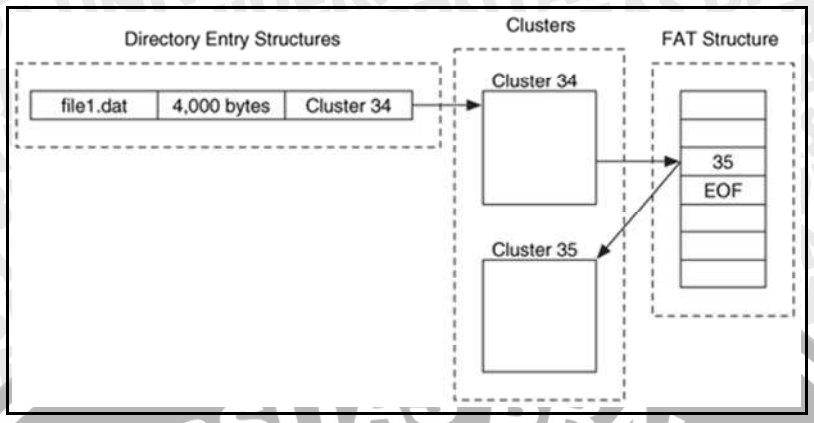

Gambar 2.15. Operasi baca per blok
Sumber: SanDisk, 2003

## 2.7 Sistem *File Allocation Table* 16 (FAT 16)

FAT (*File Allocation Table*) merupakan salah satu sistem *file* paling sederhana yang umum dipakai dalam banyak sistem operasi. Dalam sistem *file* FAT setiap *file* atau *directory* dialokasikan dalam sebuah struktur data yang disebut *directory entry*. *Directory entry* terdiri atas 32 byte yang berisi nama dan metafile dari suatu *file* atau *directory* seperti ditunjukkan dalam Tabel 2.4. Setiap *file* disimpan dalam satuan data yang disebut *cluster*. Satu c*luster* terdiri dari beberapa sektor dan satu sektor terdiri dari beberapa byte. Jika sebuah *file* dialokasikan lebih dari satu *cluster* maka lokasi *cluster* lainnya dapat ditemukan dengan membaca informasi dalam struktur yang disebut FAT. Struktur FAT digunakan untuk mengetahui lokasi *cluster* selanjutnya dari sebuah *file* dan status *cluster* apakah sudah terpakai atau belum. Saat ini dikenal ada tiga macam jenis FAT, yaitu FAT12, FAT16, dan FAT32. Perbedaan utama di antara ketiganya adalah ukuran *entry* dalam struktur FAT. Pada FAT16, kondisi tiap cluster diwakili oleh 16 bit data FAT. Gambar 2.16 menunjukkan hubungan antara *directory entry*, *cluster*, dan FAT.

Susunan memori sebuah media penyimpanan terdiri atas *Master Boot Record* (MBR) dan partisinya. Sebuah partisi terdiri atas *reserved* area, FAT area, dan data area. Dalam FAT16, *root directory* yang berisi *directory entry* terletak di awal data area. Gambar 2.17 menunjukkan struktur partisi FAT16. MBR merupakan 512 byte pertama yang digunakan sebagai kode boot pada saat pertama kali mengakses memori. Pada MBR terdapat *partition entry* yang berfungsi sebagai tabel informasi dari suatu partisi. Tabel 2.5 menunjukkan isi dari MBR dan Tabel 2.6 menunjukkan isi dari *partition*

*entry*. Salah satu informasi penting yang terdapat dalam *partition entry* adalah informasi *sector* pertama dari suatu partisi.



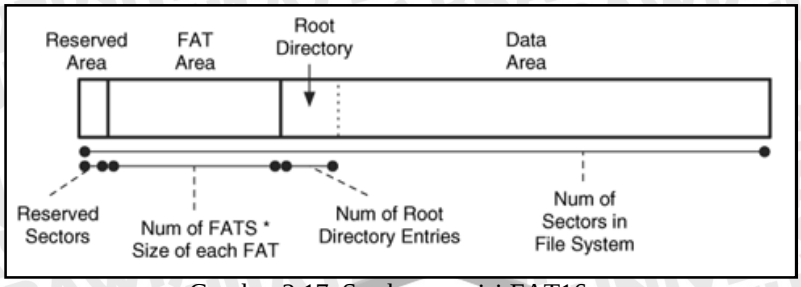

Gambar 2.16. Hubungan antara *directory entry*, *cluster*, dan FAT.

Sumber : Carrier, 2005.

Tabel 2.4. Isi directory entry.

| Byte | Keterangan |
|---|---|
| 0-0 | Karakter pertama nama *file* dalam ASCII dan status alokasi (0xE5 atau 0x00 jika belum terpakai) |
| 1-10 | Karakter 2 sampai 11 nama *file* dalam ASCII |
| 11 | Atribut *file* (*read only*, *write only*, *archive*, dsb) |
| 12 | *Reserved* |
| 13 | Waktu *file* dibuat (kelipatan 10 detik) |
| 14-15 | Waktu *file* dibuat (jam, menit, detik) |
| 16-17 | Hari *file* dibuat |
| 18-19 | Hari *file* diakses |
| 20-21 | Alamat *cluster* pertama (0 untuk FAT12 dan FAT16), 2 byte atas |
| 22-23 | Waktu *file* ditulis (jam, menit, detik) |
| 24-25 | Hari *file* ditulis |
| 26-27 | Alamat *cluster* pertama, 2 byte |
| 28-31 | Ukuran *file* (dalam byte) |

Sumber: Carrier, 2005




Gambar 2.17. Struktur partisi FAT16.

Sumber: Carrier, 2005.

Tabel 2.5. *Master* Boot Record.

| Byte | Keterangan |
|---|---|
| 0-445 | *Boot code* |
| 446-461 | *Partition entry* pertama |
| 462-477 | *Partition entry* kedua |
| 478-493 | *Partition entry* ketiga |
| 494-509 | *Partition entry* keempat |
| 510-511 | *Signature value* (0xAA55) |

**Sumber:** Carrier, 2005.

Tabel 2.6. Partition Entry.

| Byte | Keterangan |
|---|---|
| 0 | *Bootable flag* |
| 1-3 | Alamat awal (CHS) |
| 4 | Jenis partisi |
| 5-7 | Alamat akhir (CHS) |
| 8-11 | Alamat awal (LBA) |
| 12-15 | Jumlah *sector* dalam partisi |

Sumber: Carrier, 2005.

Pada awal suatu partisi terdapat *boot sector* yang berfungsi sebagai tabel informasi dari suatu partisi. Tabel 2.7 menunjukkan isi dari *boot sector*. Posisi sektor dari FAT area dan data area dapat diketahui dengan membaca isi dari *boot sector*. Alamat sektor yang tersimpan dalam *boot sector* merupakan alamat relatif pada partisi tersebut. Alamat yang sebenarnya dapat ditentukan dengan menjumlah alamat relatif dan alamat awal dari suatu partisi yang bersangkutan.

Tabel 2.7. Boot sector.

| Byte | Keterangan |
|---|---|
| 0-2 | *Jump kode* dan NOP |

| Byte | Keterangan |
|---|---|
| 3-5 | Nama OEM |
| 11-12 | Jumlah byte tiap *sector* |
| 13 | Jumlah *sector* tiap *cluster* |
| 14-15 | Ukuran *reserved sector* |
| 16 | Jumlah FAT |
| 17-18 | *Root directory entry* maksimal |
| 19-20 | Jumlah *sector* dalam partisi (16 bit) |
| 21 | Deskripsi media |
| 22-23 | Jumlah *sector* tiap FAT |
| 24-25 | Jumlah *sector* tiap *track* |
| 26-27 | Jumlah *head* |
| 28-31 | Jumlah *hidden sector* dalam satu partisi |
| 32-35 | Jumlah *sector* dalam satu partisi (32 bit) |
| 36 | BIOS INT13h |
| 37 | Tidak digunakan |
| 38 | *Extended boot signature* |
| 39-42 | Nomer serial volume |
| 43-53 | Nama volume |
| 54-61 | Nama file sistem |
| 62-509 | Tidak digunakan |
| 510-511 | *Signature value* (0xAA55) |

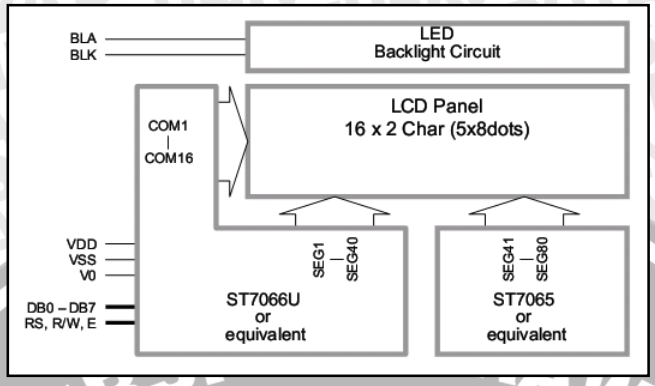
Sumber: Carrier, 2005.

## 2.8 Modul *Liquid Crystal Display* (LCD)

LCD digunakan sebagai penampil untuk memudahkan pengguna dalam memantau alat. Fitur-fitur modul LCD ini adalah sebagai berikut.

- Dapat menampilkan 16 karakter×2 baris
- Setiap huruf disusun 5×8 titik
- *Character Code* ROM
- *Character Generator* RAM (CGRAM) dan *Display Data* RAM (DDRAM)
- Dua mode antarmuka, yaitu 4 bit dan 8 bit.

Mode antarmuka 4 bit biasa digunakan untuk menghemat penggunaan pin mikrokontroler. Gambar 2.18 menunjukkan diagram blok LCD. Sedangkan fungsi pin LCD ditunjukkan dalam Tabel 2.8.




Gambar 2.18. Diagram blok LCD

Sumber : Shenzhen, 2005

Tabel 2.8. Fungsi pin LCD

| No. pin | Nama pin | I/O | Keterangan |
|---|---|---|---|
| 1 | VSS | Catu daya | Ground (0 volt) |
| 2 | VDD | Catu daya | Positif catu daya (5volt) |
| 3 | V0 | Catu daya | Tegangan referensi kontras LCD |
| 4 | RS | Input | *Register Select*<br>RS=HIGH: kirim data *display*<br>RS=LOW: kirim data instruksi |
| 5 | R/W | Input | *Read/Write Control Bus*<br>R/W=HIGH: mode baca<br>R/W=LOW: mode tulis |
| 6 | E | Input | *Data enable* |
| 7 | DB0 | I/O | *Bi-directional Tri-state Data Bus*<br>Mode 8 bit: gunakan DB0 – DB7<br>Mode 4 bit: gunakan DB4 – DB7 |
| : | : | | |
| 14 | DB7 | | |
| 15 | BLA | Catu daya | Positif catu daya *backlight* |
| 16 | BLK | Catu daya | Negatif catu daya *backlight* |

Sumber: Shenzhen, 2005.

# BAB III
# METODOLOGI PENELITIAN

Penyusunan skripsi ini didasarkan pada masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat bekerja sesuai yang direncanakan pada rumusan masalah. Langkah-langkah yang dilakukan dalam penyusunan skripsi ini meliputi perancangan, pembuatan alat, pengujian alat, dan pengambilan kesimpulan.

## 3.1 Perancangan dan Pembuatan Alat

Perancangan dilakukan dengan melakukan studi literatur terlebih dahulu mengenai materi-materi penunjang. Studi literatur yang dipelajari meliputi pengetahuan tentang Kontes Robot Cerdas Indonesia Divisi Single Expert, robot mobil diferensial, sistem koordinat lokal dan global, mikrokontroler R8C/13, sensor *rotary encoder*, *secure digital card* (*SD Card*), sistem *file allocation table 16* (FAT16), dan modul *liquid crystal display* (LCD).

Pengetahuan yang diperoleh dari studi literatur digunakan untuk membuat blok diagram alat dan menentukan prinsip kerja alat. Penentuan perangkat keras yang digunakan pada alat dilakukan secara bertahap sesuai dengan blok diagram. Perangkat lunak alat dirancang dengan membuat diagram alir program.

Pembuatan alat dilakukan dengan membuat PCB (*printed circuit board*), pemasangan komponen, dan pengemasan alat. Pembuatan perangkat lunak dilakukan dengan merealisasikan diagram alir ke dalam bahasa C menggunakan *software High-performance Embeded Workshop* dan mengisi perangkat lunak tersebut ke dalam alat.

## 3.2 Pengujian Alat

Untuk mengetahui apakah alat yang telah dibuat sesuai dengan yang direncanakan maka dilakukan pengujian rangkaian. Pengujian yang dilakukan terbagi dua jenis, yaitu :

a. Pengujian tiap bagian

Perangkat keras yang telah dibuat tahap demi tahap akan diuji satu persatu sesuai dengan prinsip kerja alat. Pengujian ini meliputi pengujian modul LCD, pengujian sensor rotari, pengujian sensor rotari terhadap jarak, pengujian perhitungan posisi, pengujian *SD Card*, dan pengujian waktu satu siklus proses.

b. Pengujian secara keseluruhan

Pengujian secara keseluruhan dilakukan dengan menghubungkan perangkat keras sesuai dengan blok diagram keseluruhan dan menjalankan perangkat lunak alat keseluruhan.

### 3.3 Pengambilan Kesimpulan dan Saran

Tahap terakhir adalah pengambilan kesimpulan dan saran. Kesimpulan diperoleh dari hasil pengujian alat dan kesesuaiannya dengan teori yang telah dipelajari. Saran diberikan untuk memperbaiki kesalahan, dan kemungkinan pengembangan alat agar lebih baik untuk penelitian selanjutnya.

# BAB IV
# PERANCANGAN DAN PEMBUATAN

Perancangan dan pembuatan alat ini terdiri atas dua bagian, perancangan perangkat keras dan perangkat lunak. Perancangan dan pembuatan alat dilakukan secara bertahap untuk memudahkan analisis sistem. Beberapa aspek yang perlu dijelaskan dalam bab ini meliputi penentuan spesifikasi alat, perencanaan masing-masing blok rangkaian serta perencanaan sistem secara keseluruhan.

## 4.1 Spesifikasi Alat

Spesifikasi alat yang akan dirancang adalah sebagai berikut.

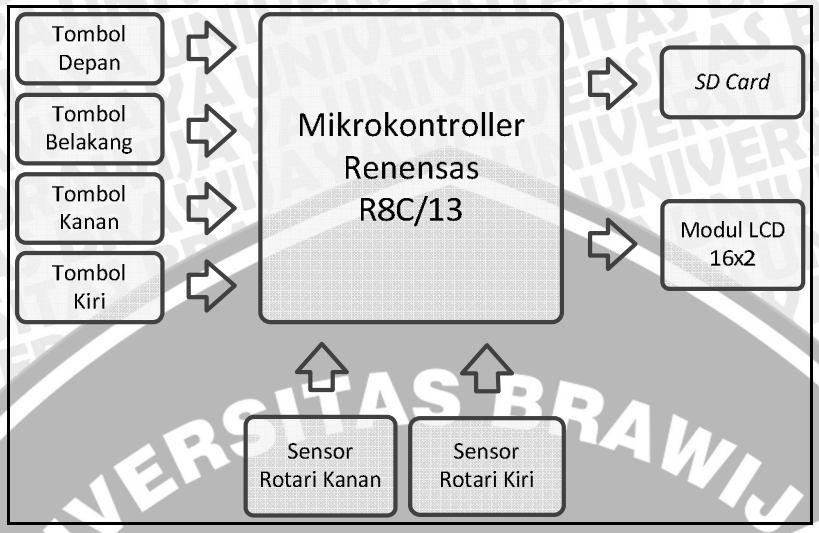
1). Jari-jari roda model robot sebesar 4,75 cm.
2). Jarak antara kedua roda model robot sebesar 26,5 cm.
3). Sensor rotari yang digunakan untuk menentukan posisi dari model robot merupakan sensor rancangan sendiri dengan memanfaatkan *optoswitch*.
4). Jumlah pulsa yang dihasilkan oleh sensor rotari dalam satu putaran sebanyak 256 pulsa.
5). Pendeteksi keberadaan dinding sekitar dimodelkan dengan menggunakan tombol.
6). Mikrokontroler yang digunakan sebagai unit pengolah data adalah Renesas R8C/13.
7). Data pemetaan yang terbentuk merupakan informasi kondisi tiap-tiap blok lapangan.
8). Data pemetaan yang terbentuk disimpan dalam *SD Card* berkapasitas 1 GB.

## 4.2 Perancangan Sistem

Perancangan alat secara keseluruhan ditunjukkan dengan diagram blok seperti dalam Gambar 4.1. Komponen penyusun yang digunakan adalah sensor rotari, tombol masukan, mikrokontroler Renesas R8C/13, *SD Card*, dan modul LCD.

Sensor rotari berfungsi sebagai pencatat pergerakan roda kanan dan kiri model robot. Setiap roda akan dipasang dua buah sensor rotari yang berdampingan guna mendeteksi arah putaran masing-masing roda. Penentuan arah putaran tiap roda ditentukan dengan memeriksa sensor rotari manakah yang mendeteksi lubang terlebih dahulu. Sensor rotari akan menghitung banyak putaran roda dengan cara memeriksa jumlah lubang yang terdeteksi pada piringan berlubang. Data jumlah putaran roda dan arah putaran roda yang dihasilkan oleh sensor rotari akan diolah untuk menentukan

posisi model robot dengan menggunakan persamaan pada robot mobil sistem diferensial.




Gambar 4.1. Diagram blok sistem.

Keempat tombol masukan merupakan permodelan sensor yang berfungsi untuk mendeteksi keberadaan dinding pada keempat sisi robot. Tombol depan digunakan untuk memberikan informasi keberadaan dinding di depan model robot. Tombol belakang, kanan dan kiri digunakan untuk memberikan informasi keberadaan dinding pada sisi belakang, kanan dan kiri model robot. Keempat tombol ditekan secara manual apabila pada sisi yang bersangkutan diasumsikan terdapat dinding.

Pengolahan data akan dilakukan oleh mikrokontroler secara periodik. Mikrokontroler akan mengolah data jumlah dan arah putaran tiap-tiap roda yang akan diolah menjadi posisi robot serta data keberadaan dinding sekitar model robot yang diperoleh dari tombol masukan. Pengolahan data yang dilakukan meliputi perhitungan arah orientasi model robot, perhitungan koordinat lokal dan global model robot serta penentuan posisi robot pada blok arena.

Hasil perhitungan akan disimpan ke dalam *SD Card* dalam *file* dengan format teks. Data yang tersimpan dalam *SD Card* adalah data posisi masing-masing blok pada lapangan lomba, keberadaan keempat dinding pada tiap-tiap blok, dan posisi terakhir dari model robot. Modul LCD dalam sistem digunakan untuk memberikan informasi posisi robot pada saat itu sehingga dapat memudahkan pemantauan sistem.

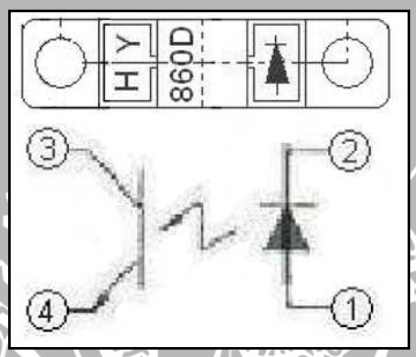
## 4.3 Perancangan Perangkat Keras

Perancangan dan pembuatan perangkat keras pada sistem pemetaan robot meliputi rangkaian sensor rotari, pembagian jalur masukan keluaran mikrokontroler,

rangkaian reset mikrokontroler, rangkaian tombol masukan, rangkaian *SD Card*, dan rangkaian modul LCD.

### 4.3.1 Rangkaian Sensor Rotari

Rangkaian sensor rotari menggunakan komponen *optoswitch* untuk mendeteksi banyak dan arah putaran roda. Rangkaian sensor rotari mendeteksi lubang dari piringan berlubang yang terpasang pada tiap roda. Komponen *optoswitch* yang digunakan adalah tipe HY860D produksi Hongkong Hingyip Electronic Co.Ltd. Komponen ini merupakan kombinasi LED inframerah sebagai pemancar cahaya dan phototransistor NPN sebagai penerima cahaya. Gambar 4.2 menunjukkan konfigurasi pin dari *optoswitch*.




Gambar 4.2. Konfigurasi pin *optoswitch*.
Sumber: Hingyip, 2009.

Gambar rangkaian sensor rotari ditunjukkan dalam Gambar 4.3. Tegangan sumber yang digunakan pada rangkaian sensor rotari sebesar 3,3 V. Berdasarkan datasheet *optoswitch* HY860D, arus bias maju ($I_F$) yang diperlukan untuk menyalakan LED adalah sebesar 20 mA dengan tegangan jatuh ($V_F$) maksimal sebesar 1,6 V, sehingga nilai minimum resistansi $R_1$ dapat ditentukan dengan menggunakan persamaan berikut.

$$R_{1(\min)} = \frac{V_{CC} - V_F}{I_F} \quad (4\text{-}1)$$

$$R_{1(\min)} = \frac{3{,}3\ V - 1{,}6\ V}{20\ mA}$$

$$R_{1(\min)} = 85\ \Omega$$

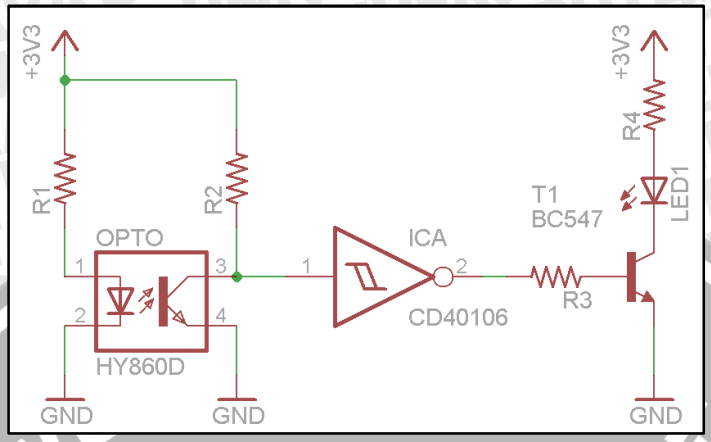
Nilai $R_1$ yang digunakan pada rangkaian sebesar 100Ω, maka besar $V_F$ dapat diketahui dengan menggunakan persamaan berikut.

$$V_F = V_{CC} - R_1 . I_F \quad (4\text{-}2)$$

$$V_F = 3{,}3\ V - 100.20.10^{-3}\ V$$

$V_F = 1{,}3\ V$

Tegangan jatuh ($V_F$) yang dihasilkan sebesar 1,3 V. Nilai tersebut berada di bawah nilai maksimal, sehingga dapat diaplikasikan pada rangkaian.




Gambar 4.3. Rangkaian sensor rotari.

Bagian penerima *optoswitch* bekerja pada dua kondisi yakni kondisi saat cahaya LED terhalang (kondisi gelap) dan tidak terhalang (kondisi terang). Berdasarkan datasheet *optoswitch* HY860D, pada saat $I_F$ bernilai 20 mA dan $I_C$ bernilai 0,2 mA maka tegangan saturasi collector-emitter ($VCE_{SAT}$) maksimal bernilai 0,4 V. Nilai resistansi minimum $R_2$ dapat ditentukan dengan menggunakan persamaan sebagai berikut.

$$R_{2(min)} = \frac{V_{CC} - VCE_{SAT(maks)}}{I_C} \quad (4\text{-}3)$$

$$R_{2(min)} = \frac{3{,}3\ V - 0{,}4\ V}{0{,}2\ mA}$$

$$R_{2(min)} = 14{,}5\ k\Omega$$

Nilai $R_2$ yang digunakan dalam rangkaian sebesar 15kΩ, sehingga nilai $VCE_{SAT}$ rangkaian dapat dihitung dengan menggunakan persamaan berikut.

$$VCE_{SAT} = V_{CC} - R_2 . I_C \quad (4\text{-}4)$$

$$VCE_{SAT} = 3{,}3\ V - 15.10^3 . 0{,}2 . 10^{-3}\ V$$

$$VCE_{SAT} = 0{,}3\ V$$

Berdasarkan perhitungan, $VCE_{SAT}$ yang dihasilkan sebesar 0,3 V. Nilai tersebut berada di bawah nilai maksimum, sehingga $R_2$ dapat diaplikasikan dalam rangkaian.

Kaki emitter phototransistor dihubungkan dengan *schmitt-trigger* guna mengurangi pengaruh *noise* pada saat pembacaan piringan berlubang. CD40106 dipilih sebagai komponen *schmitt-trigger* karena memiliki tegangan kerja 3-15 V. Komponen ini memiliki tegangan keluaran logika rendah ($V_{OL}$) sebesar 0,05 V dan tegangan keluaran logika tinggi ($V_{OH}$) sebesar $V_{DD}$ – 0,05 V. Nilai tegangan keluaran ini masih

memenuhi tegangan masukan logika rendah dan tinggi dari mikrokontroler R8C/13. Berdasarkan datasheet R8C/13, tegangan masukan logika rendah ($V_{IL}$) maksimal sebesar $0{,}2V_{CC}$ dan tegangan masukan logika tinggi ($V_{IH}$) minimal sebesar $0{,}8V_{CC}$. *Schmitt-trigger* CD40106 mempunyai tegangan histerisis sebesar $0{,}4V_{DD}$.

Pada bagian keluaran sensor selain dihubungkan dengan pin masukan mikrokontroler juga dihubungkan dengan LED yang dikuatkan terlebih dahulu menggunakan transistor BC547. Pemasangan LED ini berfungsi sebagai indikator logika keluaran dari sensor. Transistor BC547 dirancang dalam keadaan saturasi pada saat logika keluaran schmitt-trigger tinggi. Saat saturasi, BC547 mempunyai tegangan kolektor-emiter ($V_{CE}$) maksimal sebesar 0,25 V apabila diberi arus basis ($I_B$) sebesar 0,5 mA. Tegangan basis-emiter ($V_{BE}$) saat saturasi sebesar 0,7 V. Dengan menggunakan data tersebut, besarnya nilai $R_3$ dapat ditentukan dengan menggunakan persamaan berikut.

$$R_3 = \frac{V_{CC} - V_{BE}}{I_B} \quad (4\text{-}5)$$

$$R_3 = \frac{3{,}3\ V - 0{,}7\ V}{0{,}5\ mA}$$

$$R_3 = 5{,}2\ k\Omega$$

Nilai $R_3$ mendekati yang terdapat di pasaran sebesar 5,6 kΩ. Nilai $I_B$ yang mengalir pada transistor dapat diketahui dengan menggunakan persamaan berikut.

$$I_B = \frac{V_{CC} - V_{BE}}{R_3} \quad (4\text{-}6)$$

$$I_B = \frac{3{,}3\ V - 0{,}7\ V}{5{,}6.10^3 \Omega}$$

$$I_B = 0{,}46\ mA$$

Nilai $I_B$ yang mengalir apabila dibulatkan menjadi 0,5 mA. Dengan demikian $R_3$ dapat diaplikasikan dalam rangkaian.

Berdasarkan datasheet LED produksi Everlight, besarnya tegangan bias maju ($V_F$) adalah 2,4 V saat arus bias maju ($I_F$) bernilai 10 mA. Tegangan kolektor-emitor ($V_{CE(sat)}$) transistor pada saat keadaan saturasi sebesar 0,09 V. Dengan menggunakan persamaan (4-7), dapat ditentukan nilai $R_4$ minimal yang terpasang pada rangkaian.

$$R_{4(\min)} = \frac{V_{CC} - V_F - V_{CE(sat)}}{I_F} \quad (4\text{-}7)$$

$$R_{4(\min)} = \frac{3{,}3\ V - 2\ V - 0{,}09\ V}{10\ mA}$$

$$R_{4(\min)} = 121\ \Omega$$

Nilai $R_4$ yang digunakan dalam rangkaian sebesar 150Ω. Besar $V_F$ pada rangkaian dapat diketahui dengan menggunakan persamaan berikut.

$$V_F = V_{CC} - R_4 I_F - V_{CE(sat)} \quad (4\text{-}8)$$

$$V_F = 3{,}3 - 150.0{,}01 - 0{,}09\ V$$

$$V_F = 1{,}71\ V$$

Nilai $V_F$ yang bekerja pada rangkaian masih berada diantara nilai minimal sebesar 1,5 V dan nilai nominal sebesar 2 V, sehingga $R_4$ dapat diaplikasikan dalam rangkaian.

### 4.3.2 Masukan Keluaran Mikrokontroler Renesas R8C/13

Mikrokontroler Renesas R8C/13 merupakan pengolah data utama dari sistem pemetaan ini. Mikrokontroler ini berfungsi untuk menghitung jumlah putaran roda, mengolah data pemetaan, melakukan komunikasi SPI dengan *SD Card* dan mengatur tampilan pada modul LCD. Berdasarkan datasheet mikrokontroler Renesas R8C/13, tegangan catu daya yang dapat digunakan berkisar 2,7 – 5,5 V. Dalam perancangan ini, mikrokontroler akan menggunakan tegangan sumber sebesar 3,3 V. Pin masukan keluaran mikrokontroler pada perancangan ini akan difungsikan sesuai dengan Tabel 4.1.

Tabel 4.1. Fungsi pin mikrokontroler.

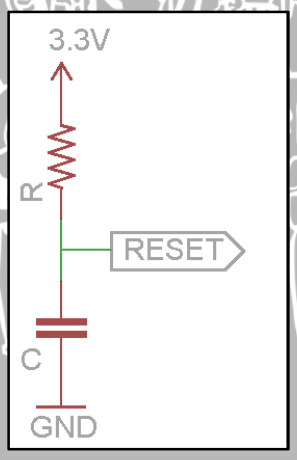
| No. | Pin | Fungsi |
|---|---|---|
| 1 | $P1_7$ ($CNTR_0$) | Jalur masukan pulsa sensor rotari kanan |
| 2 | $P3_7$ | Jalur masukan arah putaran sensor rotari kanan |
| 3 | $P3_2$ ($CNTR_1$) | Jalur masukan pulsa sensor rotari kiri |
| 4 | $P3_3$ | Jalur masukan arah putaran sensor rotari kiri |
| 5 | $P4_5$ | Jalur masukan dari tombol depan |
| 6 | $P1_0$ | Jalur masukan dari tombol samping kanan |
| 7 | $P1_1$ | Jalur masukan dari tombol samping kiri |
| 8 | $P1_2$ | Jalur masukan dari tombol belakang |
| 9 | $P1_3$ | Jalur SS untuk *SD Card* |
| 10 | $P1_4$ ($TxD_0$) | Jalur MOSI untuk *SD Card* |
| 11 | $P1_5$ ($RxD_0$) | Jalur MISO untuk *SD Card* |
| 12 | $P1_6$ ($CLK_0$) | Jalur SCK untuk *SD Card* |
| 13 | $P3_0$ | Jalur masukan saklar *SD Card* |
| 14 | $P0_1$ | Jalur perintah RS untuk modul LCD |
| 15 | $P0_2$ | Jalur perintah R/W untuk modul LCD |
| 16 | $P0_3$ | Jalur perintah *Enable* untuk modul LCD |
| 17 | $P0_{4-7}$ | Jalur data pada modul LCD |

Fasilitas Timer X pada mikrokontroler difungsikan sebagai penghitung jumlah pulsa sensor rotari kanan. Dengan memanfaatkan mode *event counter* yang terdapat pada Timer X, keluaran sensor rotari dihubungkan dengan pin $INT_1/CNTR_0$. Timer Y difungsikan sebagai penghitung jumlah pulsa sensor rotari kiri. Timer Y diatur dalam mode timer dan sumber pencacah memanfaatkan masukan dari pin $INT_2/CNTR_1$ yang dihubungkan dengan keluaran sensor rotari.

Fasilitas Serial UART0 pada mikrokontroler difungsikan sebagai unit komunikasi SPI *SD Card* dengan memanfaatkan mode *synchronous*. Jalur Tx difungsikan sebagai MOSI, sedangkan jalur Rx sebagai MISO. Sumber *clock* dari komunikasi ini memanfaatkan fungsi sumber *clock* CLK dari UART0.

### 4.3.3 Rangkaian Reset Mikrokontroler Renesas R8C/13

Rangkaian reset berfungsi untuk mengembalikan keadaan awal program saat mikrokontroler diberi catu daya. Proses reset pada dilakukan dengan memberikan logika rendah pada pin RESET selama 500 µs atau $1/f_{RING}$x20 (Renesas 2005:14) pada saat catu daya stabil. Gambar 4.4 menunjukkan rangkaian reset mikrokontroler.




Gambar 4.4. Rangkaian reset mikrokontroler.

Besarnya nilai R dan C pada rangkaian dapat ditentukan dengan menggunakan persamaan berikut.

$$V_{CC} = V_R + V_C \quad (4\text{-}9)$$

$$V_{CC} = R.i(t) + \frac{1}{C}\int i(t)dt$$

Dengan menggunakan transformasi Laplace diperoleh persamaan :

$$\frac{V_{CC}}{s} = I_{(s)}R + \frac{1}{Cs}I_{(s)} \quad (4\text{-}10)$$

$$I_{(s)} = \left(\frac{1}{s+\frac{1}{RC}}\right)\frac{V_{CC}}{R}$$

Dengan transformasi balik didapatkan persamaan sebagai berikut:

$$i(t) = \frac{V_{CC}}{R}.e^{-\frac{t}{RC}} \quad (4\text{-}11)$$

Dari persamaan arus dapat diperoleh nilai tegangan reset sebagai berikut:

$$V_{RST} = V_{CC} - V_R \quad (4\text{-}12)$$

$$V_{RST} = V_{CC} - \left(\frac{V_{CC}}{R}.e^{-\frac{t}{RC}}\right)R$$

$$V_{RST} = V_{CC}\left(1 - e^{-\frac{t}{RC}}\right)$$

Besarnya nilai t dapat ditentukan dengan menggunakan persamaan berikut.

$$\frac{V_{RST}}{V_{CC}} = 1 - e^{-\frac{t}{RC}} \quad (4\text{-}13)$$

$$1 - \left(\frac{V_{RST}}{V_{CC}}\right) = e^{-\frac{t}{RC}}$$

$$\ln\left(\frac{1}{1-\frac{V_{RST}}{V_{CC}}}\right) = \ln\left(e^{\frac{t}{RC}}\right)$$

$$t = RC.\ln\left(\frac{1}{1-\frac{V_{RST}}{V_{CC}}}\right)$$

Berdasarkan datasheet mikrokontroler R8C/13 diketahui level tegangan rendah pada pin reset maksimal sebesar $0{,}2V_{CC}$ dan waktu reset selama 2500 µs. Misal ditentukan nilai kapasitor sebesar 10 µF, maka besarnya nilai resistor dapat dihitung dengan menggunakan persamaan berikut.

$$R = \frac{t}{C.\ln\left(\frac{1}{1-\frac{V_{RST}}{V_{CC}}}\right)} \quad (4\text{-}14)$$

$$R = \frac{2500.10^{-6}}{10.10^{-6}.\ln\left(\frac{1}{0{,}8}\right)}$$

$$R = 1120\,\Omega$$

Nilai R yang terdapat di pasaran sebesar 1,2 kΩ. Apabila menggunakan R 1,2 kΩ dapat diketahui waktu reset dengan menggunakan persamaan berikut.

$$t = RC.\ln\left(\frac{1}{1-\frac{V_{RST}}{V_{CC}}}\right) \quad (4\text{-}15)$$

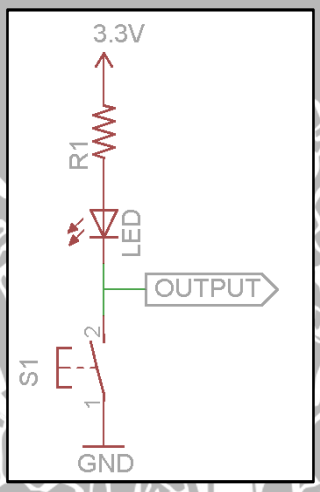
$$t = 1{,}2.10^{3}.10^{-5}.\ln\left(\frac{1}{1-\frac{0{,}2\,V_{CC}}{V_{CC}}}\right)$$

$t = 2677{,}2\ \mu s$

Berdasarkan perhitungan dapat diketahui waktu reset yang terbentuk selama 7266,2 µs. Nilai tersebut lebih besar dari batas minimal waktu reset sebesar 2500 µs, sehingga R 1,2 kΩ dapat diaplikasikan dalam rangkaian.

### 4.3.4 Rangkaian Tombol Masukan

Tombol masukan pada perancangan ini digunakan sebagai permodelan dari sensor pendeteksi keberadaan dinding. Informasi dinding sekitar dikirimkan ke mikrokontroler dengan cara menekan tombol masukan secara manual. Tombol masukan akan ditekan apabila diasumsikan terdapat dinding. Rangkaian tombol ditunjukkan dalam Gambar 4.5.




Gambar 4.5. Rangkaian tombol masukan.

LED berfungsi sebagai indikator kondisi tombol masukan. Berdasarkan datasheet LED produksi Everlight, tegangan maju ($V_F$) yang bekerja pada LED sebesar 2 V saat diberi arus maju ($I_F$) sebesar 10 mA. Pada saat tombol tertekan maka saklar akan tertutup. Kondisi ini akan menyebabkan arus mengalir melalui LED dan membuat LED aktif. Besarnya nilai resisistor $R_1$ minimal dapat ditentukan dengan menggunakan persamaan berikut.

$$R_1 = \frac{V_{CC} - V_F}{I_F} \quad (4\text{-}16)$$

$$R_1 = \frac{3{,}3\ V - 2\ V}{10\ mA}$$

$$R_1 = 130\ \Omega$$

Nilai $R_1$ yang terdapat dipasaran sebesar 150 Ω. Apabila R 150 Ω digunakan dalam rangkaian maka besar tegangan maju yang bekerja dapat diketahui dengan menggunakan persamaan berikut.

$$V_F = V_{CC} - R_1 . I_F \quad (4\text{-}17)$$

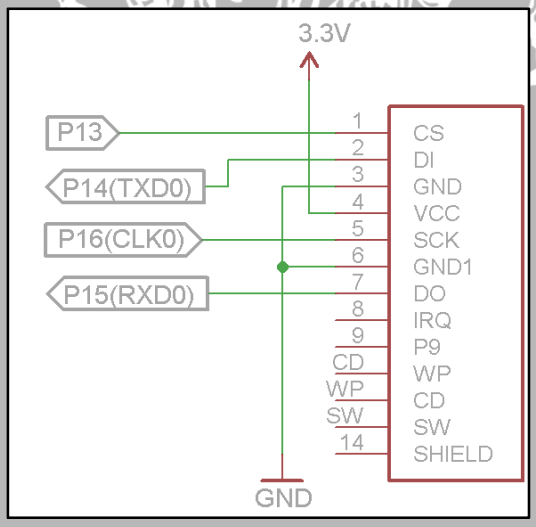
$$V_F = 3{,}3\ V - 150.10^{-2}\ V$$

$$V_F = 1{,}8\ V$$

Nilai tegangan maju yang bekerja pada LED sebesar 1,8 V. Nilai tersebut masih berada di atas nilai $V_F$ minimum, sehingga dapat dipasang pada rangkaian.

### 4.3.5 Rangkaian Secure Digital Card

Secure Digital Card pada perancangan ini digunakan sebagai media penyimpan data keluaran dari sistem pemetaan. Berdasarkan datasheet *SD Card* Sandisk, tegangan catu daya berkisar antara 2,7 – 3,6 V. Dalam perancangan, rangkaian *SD Card* menggunakan tegangan catu daya sebesar 3,3 V. *SD Card* akan berkomunikasi dengan menggunakan mode SPI. Pada rangkaian mikrokontroler akan memanfaatkan fasilitas serial mode *synchronous* untuk berkomunikasi dengan SPI dari *SD Card*. Pin CS pada *SD Card* akan dihubungkan dengan $P1_3$. Pin DI akan dihubungkan dengan $P1_4(TxD_0)$ dari mikrokontroler. Pin DO dihubungkan dengan $P1_5(RxD_0)$. Sedangkan sumber *clock* SCLK pada mode ini menggunakan keluaran dari $P1_6(CLK_0)$. Gambar rangkaian dari *SD Card* ditunjukkan dalam Gambar 4.6.

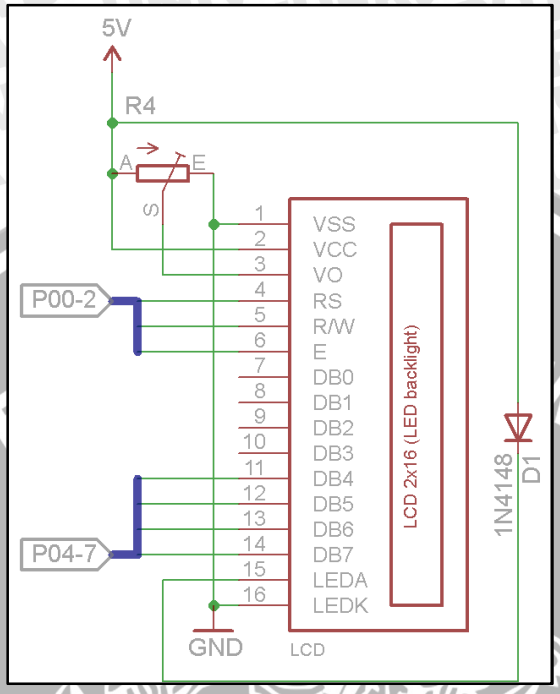



Gambar 4.6. Rangkaian *SD Card.*

### 4.3.6 Rangkaian Modul Liquid Crystal Display

Pada perancangan ini modul LCD digunakan untuk menampilkan posisi model robot pada arena sehingga dapat diperiksa kesesuaian pergerakan model robot dengan pengolahan data. Modul LCD diantarmukakan dengan mikrokontroler menggunakan mode empat jalur data. Pin RS dari LCD terhubung ke $P0_0$ mikrokontroller, pin R/W

dari LCD terhubung ke $P0_1$ dan pin *Enable* dari LCD terhubung ke $P0_2$ mikrokontroller. Sedangkan pin data dari LCD (PB4 sampai PB7) dihubungkan dengan $P0_4$ sampai $P0_7$. Gambar rangkaian dari modul LCD dapat dilihat dalam Gambar 4.7.




Gambar 4.7 Rangkaian modul LCD.

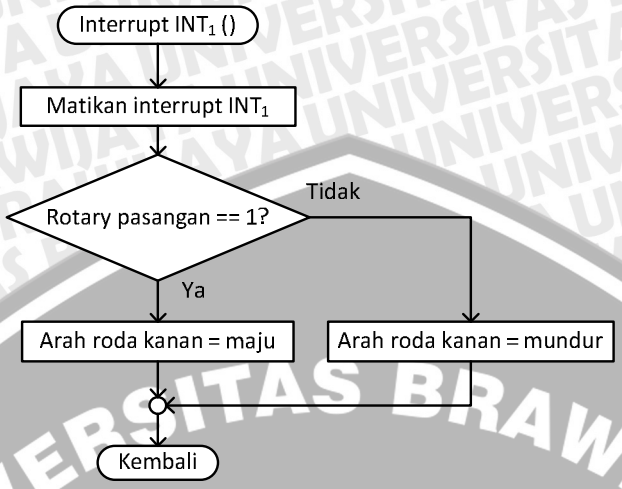
## 4.4 Perancangan Perangkat Lunak

Pada perancangan perangkat lunak akan dibagi menjadi enam sub pokok bahasan yaitu pembahasan mengenai sub rutin interrupt $INT_1$ dan $INT_2$, perhitungan dalam menentukan posisi model robot, sub rutin koreksi, format basis data, sub rutin penyimpanan basis data dan perangkat lunak utama.

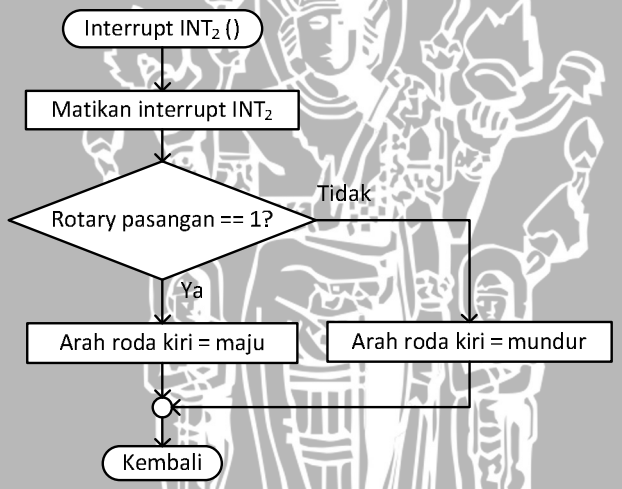
### 4.4.1 Sub Rutin Interrupt $INT_1$ dan $INT_2$

Pada sub rutin ini akan dilakukan proses penentuan arah putaran roda. Pada kondisi ini pin $CNTR_0$ ($P1_7$) akan difungsikan sebagai $INT_1$ dan pin $CNTR_1$ ($P3_2$) akan difungsikan sebagai $INT_2$. Pada saat interupsi terjadi maka akan dilakukan pengecekan kondisi logika dari sensor rotari pasangannya guna menentukan arah putaran masing-masing roda. Arah putaran roda dianggap maju apabila kondisi logika sensor pasanganya berlogika 1. Interupsi ekternal ini akan diaktifkan satu kali dalam satu siklus proses program utama. Pengaktifan interupsi dilakukan pada akhir program utama selesai dikerjakan dan penon-aktifan dilakukan saat sub rutin interupsi tersebut

dikerjakan. Diagram alir program sub rutin interrupt $INT_1$ dan $INT_2$ ditunjukkan dalam Gambar 4.8 dan 4.9.




Gambar 4.8. Diagram alir program sub rutin interrupt $INT_1$



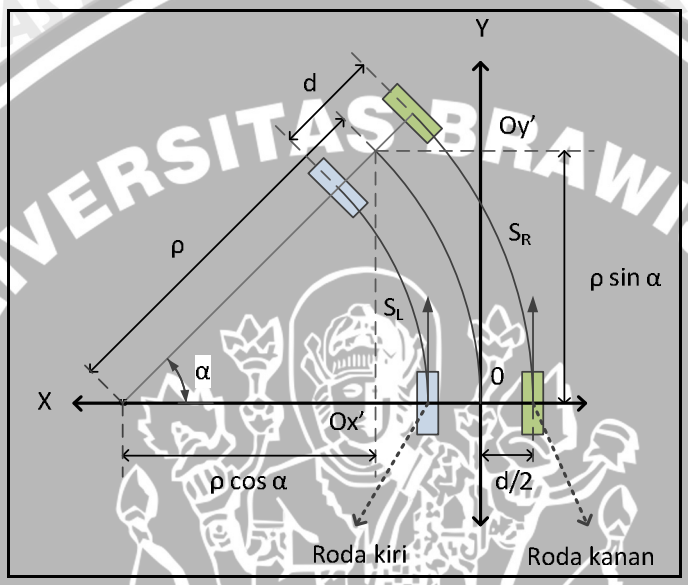

Gambar 4.9. Diagram alir program sub rutin interrupt $INT_2$.

### 4.4.2 Perhitungan untuk Menentukan Posisi Robot

Perhitungan untuk menentukan posisi model robot didasarkan pada perbedaan jarak tempuh dan arah gerak tiap roda. Data yang diperoleh dari sensor rotari adalah data pulsa (n) tiap waktu pengambilan dan data arah putaran. Data pulsa akan bernilai positif jika arah putaran roda maju dan bernilai negatif apabila sebaliknya. Apabila diketahui jari-jari roda (2r) dan jumlah pulsa per putaran (ppr), akan dapat dihitung besarnya jarak yang telah ditempuh (S) dengan menggunakan Persamaan (4-18)

$$S = \frac{n}{ppr} . 2\pi . r \qquad (4\text{-}18)$$

Setelah diperoleh nilai jarak tempuh roda kanan ($S_R$) dan kiri ($S_L$) dapat dilakukan perhitungan selanjutnya guna menentukan sudut dan perubahan posisi yang terbentuk akibat perbedaan jarak tempuh tiap roda. Gambar 4.10 menunjukkan ilustrasi gerak robot mobil sistem diferensial. Gerak robot mobil sistem diferensial dapat dianggap sebagai gerak melingkar dengan satu titik pusat yang sama. Diketahui besarnya jarak antara kedua roda sebesar d. Nilai sudut (α) yang terbentuk akibat pergerakan dapat ditentukan dengan menggunakan persamaan dasar lingkaran .




Gambar 4.10. Ilustrasi gerak robot.

$$S_R = \frac{\alpha}{2\pi} x 2\pi(\rho + d/2) \quad (4\text{-}19)$$

$$S_L = \frac{\alpha}{2\pi} x 2\pi(\rho - d/2) \quad (4\text{-}20)$$

Dengan mensubstitusikan Persamaan (4-19) dan (4-20) akan dapat diperoleh nilai jari-jari yang terbentuk (ρ) seperti pada persamaan berikut.

$$\rho = \frac{d(S_R + S_L)}{2(S_R - S_L)} \quad (4\text{-}21)$$

Besarnya sudut yang terbentuk (α) dapat ditentukan dengan menggunakan Persamaan (4-19) dan (4-21).

$$\alpha = \frac{(S_R - S_L)}{d} \quad (4\text{-}22)$$

Perubahan posisi model robot terhadap posisi sebelumnya dapat ditentukan dengan menggunakan persamaan trigonometri. Apabila diasumsikan posisi sebelumnya sebagai pusat koordinat lokal dengan arah sumbu y positif merupakan arah depan model robot maka nilai koordinat lokal ($o_x$', $o_y$') dapat ditentukan dengan menggunakan Persamaan (4-23) dan (4-24).

$$o'_x = -(\rho - \rho.\cos\alpha) \quad (4\text{-}23)$$

$$o'_y = \rho.\sin\alpha \quad (4\text{-}24)$$

Langkah selanjutnya adalah menentukan koordinat global yang terbentuk terhadap titik awal pergerakan model robot. Apabila diketahui koordinat global posisi sebelumnya atau titik nol koordinat lokal yakni ($r_x$, $r_y$, φ') maka dengan menggunakan matrik transformasi *homogeneous* dapat ditentukan koordinat global posisi terakhir terhadap titik awal pergerakan.

$$\begin{bmatrix} o_x \\ o_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & r_x \\ 0 & 1 & r_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} cos\varphi' & -sin\varphi' & 0 \\ sin\varphi' & cos\varphi' & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} o_x{}' \\ o_y{}' \\ 1 \end{bmatrix} \quad (4\text{-}25)$$

Persamaan matrik transformasi *homogeneous* (4-25) dapat disederhanakan menjadi Persamaan (4-26) dan (4-27) agar dapat diolah ke dalam program mikrokontroler. Sedangkan sudut global posisi terakhir merupakan penjumlahan sudut lokal dengan sudut sebelumnya.

$$o_x = o'_x.\cos\ \varphi' - o'_y.\sin\ \varphi' + r_x \quad (4\text{-}26)$$

$$o_y = o'_x.\sin\ \varphi' + o'_y.\cos\ \varphi' + r_y \quad (4\text{-}27)$$

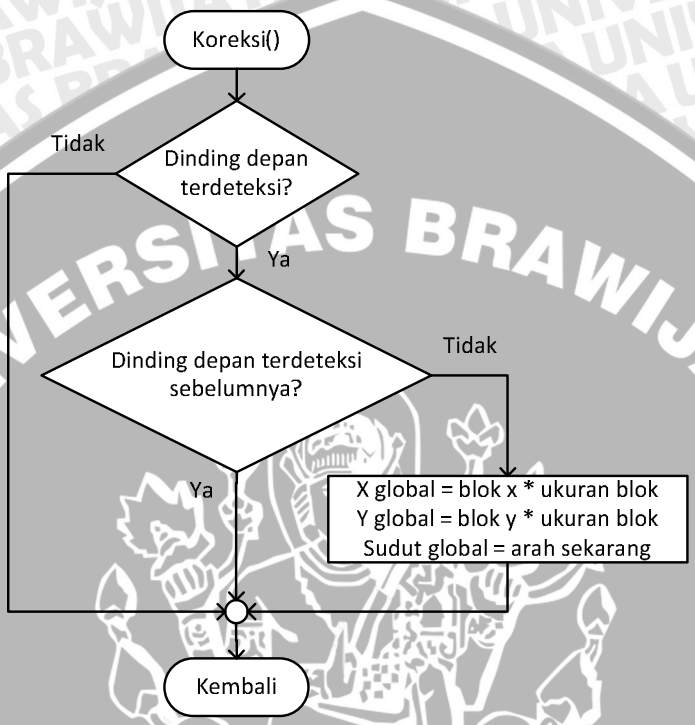
$$\varphi = \varphi' + \alpha \quad (4\text{-}28)$$

Lapangan yang terdiri dari blok-blok dapat diasumsikan sebagai sistem koordinat dengan satuan terkecil dalam ukuran satu blok. Sumbu y positif searah dengan arah depan posisi awal robot. Apabila diasumsikan blok awal pergerakan model robot sebagai blok dengan koordinat (0,0) maka penentuan blok posisi model robot pada arena dapat diketahui dengan cara membagi nilai koordinat global $o_x$ dan $o_y$ dengan ukuran blok yang sesungguhnya. Arah model robot yang digunakan dalam pemetaan merupakan kelipatan 90°. Arah yang digunakan dapat diasumsikan sebagai empat arah mata angin (utara, timur, selatan, dan barat) apabila dilihat dari atas model robot. Penentuan arah model robot dilakukan dengan mencari arah terdekat dengan nilai sudut global yang terbentuk.

### 4.4.3 Sub Rutin Koreksi

Sub-rutin koreksi diperlukan untuk mengurangi kesalahan koordinat global yang diperoleh dari perhitungan data rotari. Sub-rutin ini akan memeriksa keberadaan dinding di depan robot. Berdasarkan aturan KRCI 2009, posisi dinding selalu berada pada kelipatan blok arena. Informasi tersebut digunakan sebagai dasar dari sub-rutin koreksi.

Sub-rutin ini akan mengembalikan nilai koordinat x dan y global yang tersimpan sesuai dengan posisi tengah blok robot pada saat keberadaan dinding depan terdeteksi. Sedangkan nilai sudut global yang tersimpan akan dikembalikan pada nilai tengah dari masing-masing arah mata angin robot. Diagram alir sub rutin koreksi ditunjukkan dalam Gambar 4.11.




Gambar 4.11. Diagram alir sub rutin koreksi.

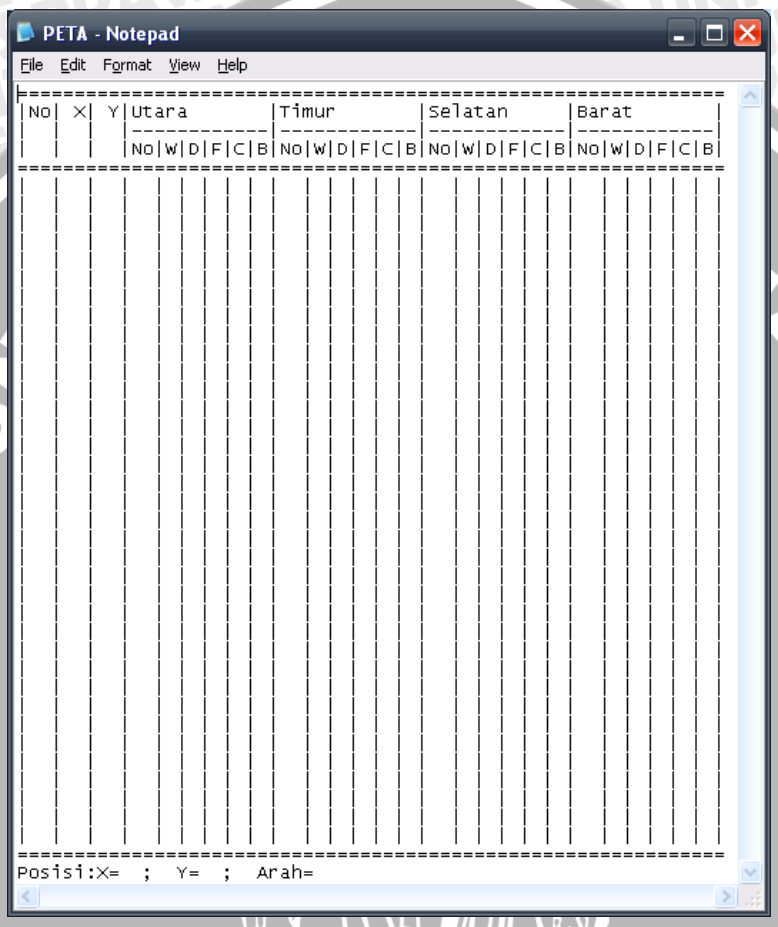
### 4.4.4 Format Basis Data

Data pemetaan yang terbentuk akan disimpan dalam basis data dengan format teks. Basis data dirancang untuk dapat menyimpan informasi posisi dari masing-masing blok, blok tetangga dari tiap-tiap blok, posisi dinding dari tiap-tiap blok, dan posisi terakhir dari model robot.

Basis data dirancang dengan bentuk tabel dalam format teks yang ditunjukkan dalam Gambar 4.12. Basis data terdiri dari tujuh kolom utama yakni kolom No, X, Y, Utara, Timur, Selatan, dan Barat. Kolom No berfungsi sebagai tempat meletakkan nomer dari tiap-tiap blok. Setiap blok dari lapangan akan mempunyai nomer unik. Penomeran tiap blok dilakukan berdasarkan urutan blok yang tersimpan dalam basis data dari hasil perjalanan model robot. Kolom X dan Y merupakan posisi x dan y blok terhadap posisi awal model robot. Sumbu positif y diasumsikan sejajar dengan arah

depan posisi awal model robot. Posisi x=0 dan y=0 merupakan posisi awal model robot (*home*).

Kolom Utara, Timur, Selatan, dan Barat merupakan kondisi lingkungan model robot pada arah-arah tersebut. Arah utara merupakan arah depan robot pada posisi pertama kali model robot diletakkan. Penentuan arah arah timur, selatan, dan barat dilakukan berdasarkan arah mata angin apabila dilihat dari bagian atas model robot.




Gambar 4.12. Format basis data pemetaan.

Pada tiap-tiap kolom Utara, Timur, Selatan, dan Barat terdapat enam sub-kolom yakni sub-kolom No, W, D, F, C, dan B. Sub-kolom No merupakan nomer blok tetangga dari blok yang bersangkutan pada arah sesuai dengan kolom utama. Sub-kolom W digunakan untuk meletakkan data keberadaan dinding dari blok yang bersangkutan pada arah sesuai dengan kolom utama. Apabila terdapat dinding maka kolom ini akan terisi dengan karakter ‘W’ dan apabila tidak ada dinding terisi dengan karakter ‘-’. Sedangkan sub-kolom D, F, C, dan B pada sistem ini tidak digunakan. Sub-kolom D, F, C, dan B digunakan untuk meletakkan informasi tentang keberadaan pintu, *furniture*, lilin dan bayi.

Pada basis data, posisi terakhir dari model robot dicatat pada bagian bawah tabel. Posisi x dan y merupakan posisi blok akhir model robot terhadap posisi awalnya. Sedangkan data arah merupakan arah orientasi akhir model robot terhadap arah awal apabila dilihat dari bagian atas. Basis data dirancang dengan jumlah masukan blok sebanyak 36 blok. Hal tersebut dikarenakan jumlah maksimal blok yang terdapat pada lapangan bawah sebayak 36 blok.

### 4.4.5 Sub Rutin Penyimpanan Basis Data

Sub rutin ini digunakan pada proses penyimpanan basis data terbaru ke dalam sistem FAT16. Jika belum diketahui alamat *cluster* tempat *file* basis data pemetaan berada, terlebih dahulu akan dilakukan pemeriksaan terhadap isi dari *root directory*. Tujuan dari pemeriksaan *root directory* adalah mencari nomer *cluster* dari data dengan nama *file* "PETA.TXT". Pencarian ini dikerjakan dengan mencocokkan setiap data dengan nilai ASCII dari "PETA". Apabila ditemukan nama *file* yang sesuai maka nomer *cluster* dari *file* tersebut akan disimpan.

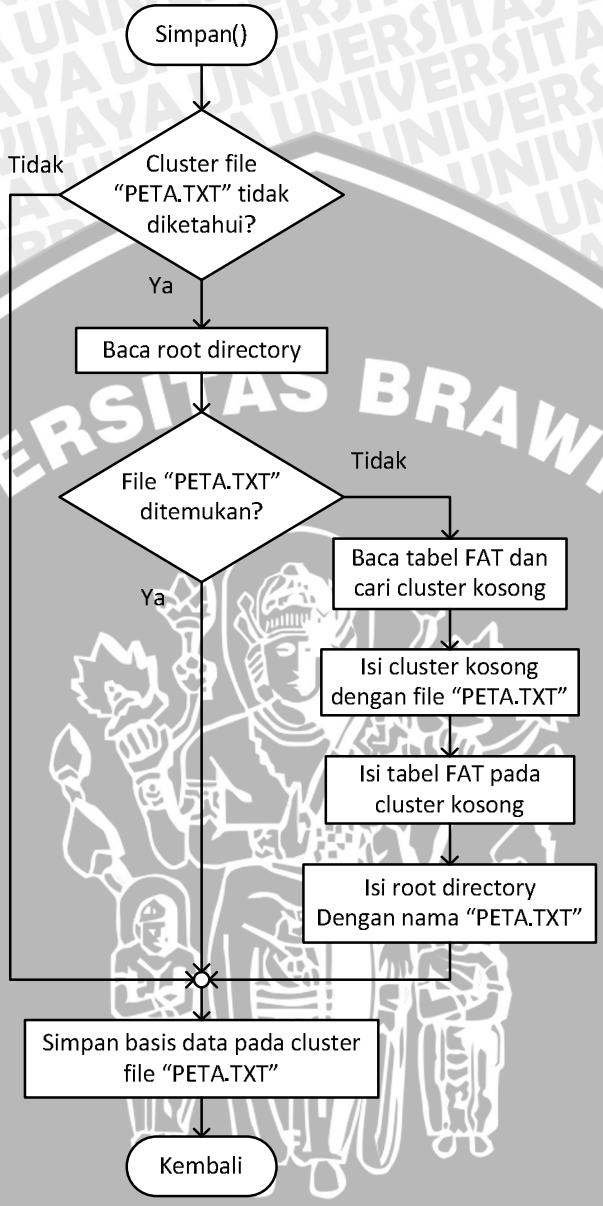
Apabila tidak ditemukan *file* yang bersangkutan, maka akan dilakukan pembuatan *file* basis data pada *cluster* yang kosong. Penentuan nomer *cluster* kosong dapat diketahui dengan membaca tabel FAT dan mencari nomer *cluster* yang tidak digunakan. Proses pembuatan *file* baru meliputi pengisian *cluster* kosong dengan data ASCII dari basis data, pembaruan tabel FAT dan pengisian *root directory* dengan nama *file* dan alamat *cluster*.

Proses selanjutnya adalah pembaruan basis data pada *cluster* yang telah diketahui. Proses pembaruan dilakukan dengan cara membaca data tiap-tiap *sector* dari *cluster* kemudian dilakukan perubahan terhadap data terbaru *file* basis data. Setelah perubahan dilakukan, data dari *sector* tersebut akan ditulis kempali pada posisi aslinya. Proses tersebut dilakukan berulang-ulang hingga semua *sector* pada *cluster* tempat *file* basis data berada diperbaharui. Gambar 4.13 menunjukkan diagram alir dari sub rutin penyimpanan basis data.

### 4.4.6 Perangkat Lunak Utama

Pada program utama dilakukan beberapa proses yakni, pembacaan data sensor, pengolahan data sensor, pembuatan dan penulisan data pemetaan ke dalam *SD Card*. Pembacaan data sensor meliputi pembacaan arah putaran roda, jumlah putaran roda, dan

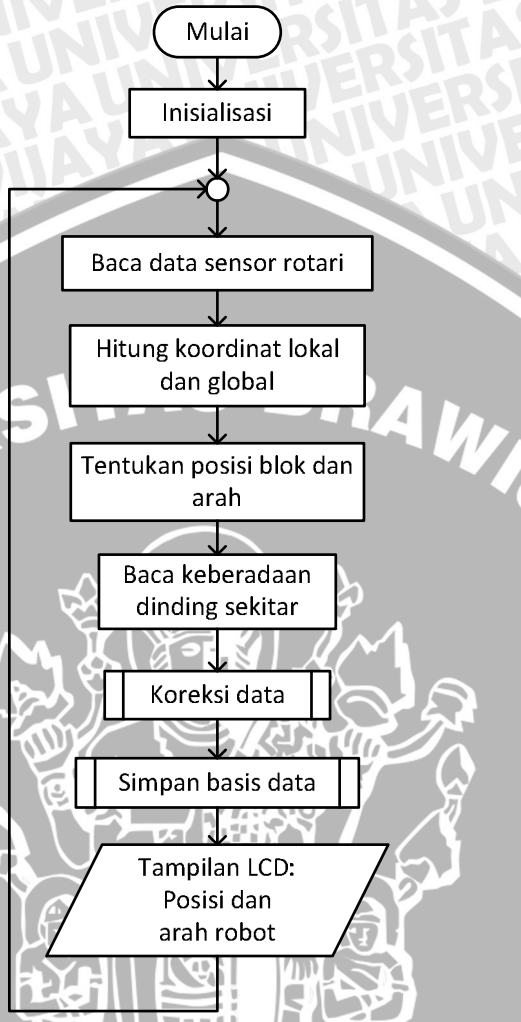
keberadaan dinding sekitar model robot. Jumlah putaran roda diperoleh dengan membaca data pencacah yang tersimpan pada register PREX, TX, PREY, dan TY.




Gambar 4.13. Diagram alir sub rutin penyimpanan basis data.

Proses pengolahan yang dilakukan meliputi perhitungan koordinat lokal model robot terhadap posisi sebelumnya, perhitungan sudut orientasi model robot dan perhitungan koordinat global model robot terhadap titik awal. Setelah koordinat global model robot diperoleh maka program akan memutuskan pada posisi blok mana model robot berada. Program akan membaca data keberadaan dinding sekitar model robot. Program akan menyesuaikan informasi keberadaan dinding sekitar model robot dengan sudut orientasi model robot pada saat itu. Data hasil pengolahan ini akan disimpan ke

dalam basis data pada *SD Card*. Diagram alir dari program utama ditunjukkan dalam Gambar 4.14.




Gambar 4.14. Diagram alir program utama.

# BAB V
# PENGUJIAN DAN ANALISIS

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

1) Pengujian modul LCD 16x2
2) Pengujian sensor rotari
   a) Pengujian pengolah sinyal
   b) Pengujian jumlah pulsa satu putaran
3) Pengujian sensor rotari terhadap jarak
4) Pengujian perhitungan posisi
5) Pengujian *SD Card*
   a) Pengujian penulisan *SD Card*
   b) Pengujian pembacaan *SD Card*
6) Pengujian waktu satu siklus
7) Pengujian keseluruhan

## 5.1 Pengujian Modul LCD 16x2

Pegujian modul LCD bertujuan untuk mengetahui keberhasilan LCD menampilkan tulisan sesuai dengan perangkat lunak yang terdapat dalam mikrokontroler. Pengujian dilakukan dengan menghubungkan LCD dan mikrokontroler yang sudah berisi perangkat lunak untuk menampilkan tulisan “Program:” pada baris atas dan “Uji LCD….” pada baris bawah. Daftar program yang digunakan dalam pengujian ini terdapat dalam Lampiran II.

Pada pengujian ini, LCD dapat menampilkan tulisan ”Program:” pada baris pertama dan teks “Uji LCD….” pada baris kedua. Tampilan hasil pengujian modul LCD dapat dilihat dalam Gambar 5.1. Berdasar hasil pengujian diketahui bahwa LCD dapat menampilkan tulisan dengan benar dan perangkat lunak yang digunakan untuk menampilkan tulisan pada LCD dapat bekerja dengan baik. Dengan demikian LCD dapat digunakan pada sistem keseluruhan.

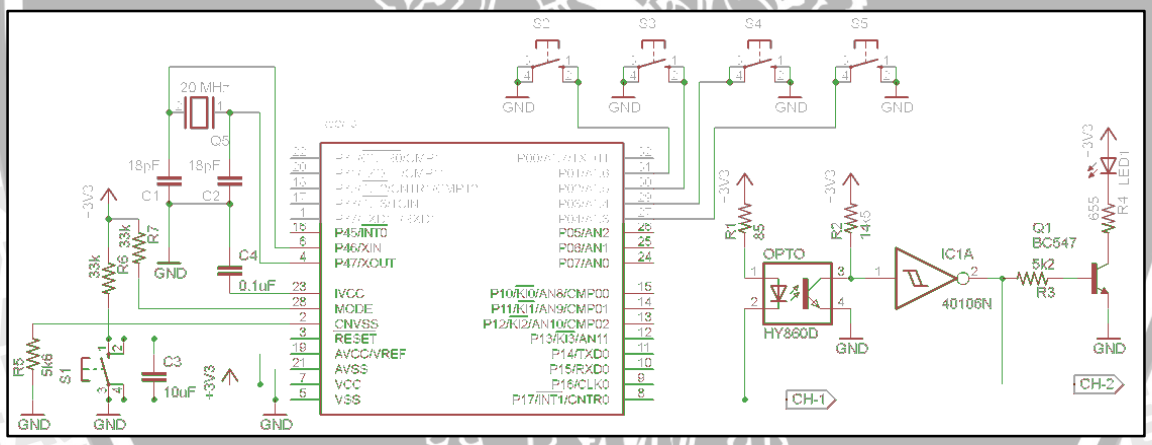

Gambar 5.1 Tampilan hasil pengujian modul LCD

## 5.2 Pengujian Sensor Rotari

Pengujian sensor rotari dibedakan menjadi dua bagian yakni pengujian pengolah sinyal dan pengujian jumlah pulsa dalam satu putaran.

### 5.2.1 Pengujian Pengolah Sinyal

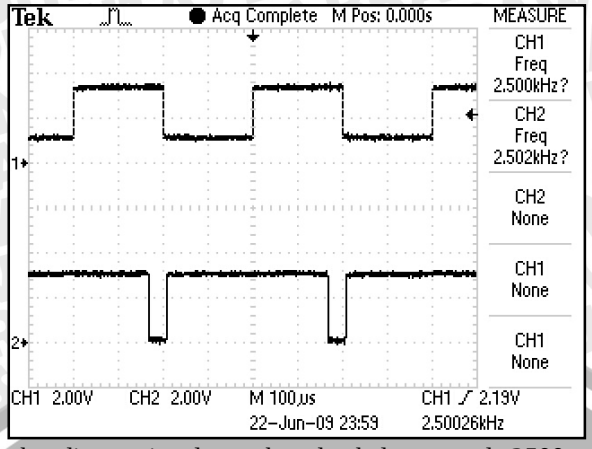
Pengujian ini dilakukan untuk mengetahui apakah pengolah sinyal sensor rotari dapat meneruskan sinyal yang dari *optoswitch* dengan benar. Pengujian dilakukan dengan memberikan pulsa pada LED *optoswitch* sehingga dapat berkedip. Sinyal keluaran pengolah sinyal sensor rotari kemudian diamati dengan menggunakan osiloskop. Pada pengujian, *optoswitch* dihubungkan dengan mikrokontroler seperti dalam Gambar 5.2.



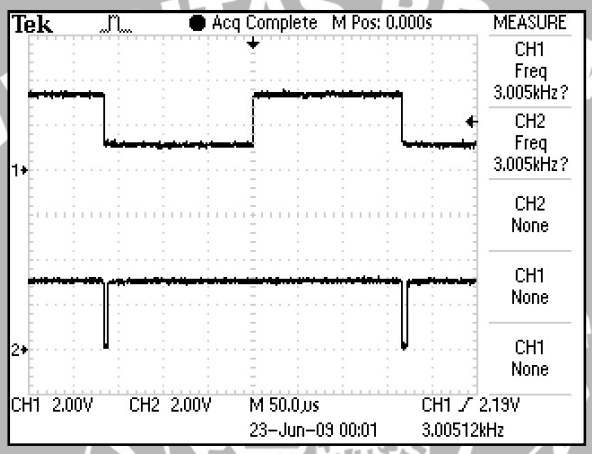

Gambar 5.2. Rangkaian pengujian pengolah sinyal sensor rotari.

Mikrokontroler digunakan sebagai pembangkit pulsa dengan memanfaatkan fasilitas timer yang terdapat di dalamnya. Daftar kode program yang digunakan pada pengujian ini terdapat pada Lampiran II. Pada pengujian ini, *channel* 1 osiloskop dihubungkan dengan keluaran timer mikrokontroler yang dihubungkan dengan LED *optoswitch* dan *channel* 2 osiloskop dihubungkan dengan keluaran sensor rotari. Pengujian dilakukan dengan membangkitkan pulsa sebanyak 2500 pulsa/s, 3000 pulsa/s dan 3500 pulsa/s.

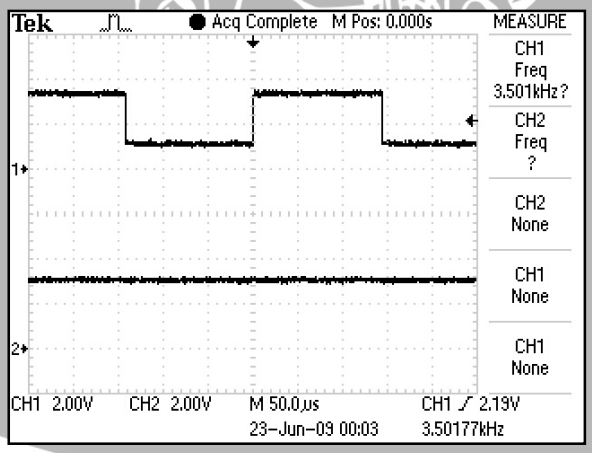
Hasil pengujian kecepatan respon sensor rotari untuk tiap-tiap frekuensi uji ditunjukkan dalam Gambar 5.3.




a) Perbandingan sinyal masukan dan keluaran pada 2500 pulsa/s




b) Perbandingan sinyal masukan dan keluaran pada 3000 pulsa/s




c) Perbandingan sinyal masukan dan keluaran pada 3500 pulsa/s

Gambar 5.3. Tampilan sinyal pada osiloskop

Hasil pengujian menunjukkan pengolah sinyal sensor rotari mampu meneruskan sinyal dari *optoswitch* dengan benar pada 3000 pulsa/s. Berdasarkan perancangan, diketahui jari-jari roda model robot sebesar 4,75 cm dan jumlah pulsa per putaran sensor sebesar 256 pulsa. Kecepatan robot yang dapat menggunakan pengolah sinyal sensor rotari ini dapat diketahui dengan menggunakan persamaan berikut.

$$v = \frac{2\pi \, x \, R \, x \, f}{ppr} \tag{5-1}$$

$$v = \frac{2\pi \, x \, 4{,}75 \, x \, 10^{-2} \, x \, 3 \, x \, 10^{3}}{256} m/s = 3{,}4975 \, m/s$$

Berdasarkan pemantauan yang dilakukan terhadap video robot-robot lomba nasional, diperoleh informasi bahwa kecepatan robot lomba kurang dari 2 m/s. Hal ini menunjukkan bahwa pengkondisi sinyal sensor rotari dapat diterapkan pada robot-robot lomba.

### 5.2.2 Pengujian Jumlah Pulsa Satu Putaran

Pengujian ini dilakukan untuk mengetahui apakah sensor rotari mampu menghasilkan jumlah pulsa dalam satu putaran sesuai dengan perancangan. Pengujian ini juga bertujuan untuk mengetahui kesesuaian perangkat lunak mikrokontroler yang berfungsi sebagai penghitung pulsa sensor rotari. Prosedur pengujian dilakukan dengan menghubungkan keluaran sensor rotari, mikrokontroler dan modul LCD sesuai dengan Gambar 5.4.


Gambar 5.4. Diagram blok pengujian jumlah pulsa sensor rotari.

Mikrokontroler digunakan sebagai penghitung pulsa sensor rotari. Proses perhitungan pulsa dilakukan dengan memanfaatkan fasilitas timer mikrokontroler yang difungsikan sebagai penghitung pulsa. Jumlah pulsa hasil pembacaan timer ditampilkan dalam LCD. Pengujian dilakukan dengan memutar roda model robot yang dilengkapi sensor rotari sebanyak satu putaran menggunakan tangan. Daftar kode program penghitung pulsa sensor rotari yang digunakan pada pengujian ini terdapat pada Lampiran II.

Hasil pengujian yang diperoleh setelah melakukan sepuluh kali pengambilan data ditunjukkan dalam Tabel 5.1.

Tabel 5.1. Hasil pengujian rotari tiap putaran

| Pengujian ke- | Jumlah pulsa sensor rotari | Kesalahan (%) |
|---|---|---|
| 1 | 255 | 0,39 |
| 2 | 256 | 0,00 |
| 3 | 256 | 0,00 |

| Pengujian ke- | Jumlah pulsa sensor rotari | Kesalahan (%) |
|---|---|---|
| 4 | 255 | 0,39 |
| 5 | 256 | 0,00 |
| 6 | 256 | 0,00 |
| 7 | 256 | 0,00 |
| 8 | 255 | 0,39 |
| 9 | 255 | 0,39 |
| 10 | 256 | 0,00 |

Berdasarkan Tabel 5.1, dapat diperoleh hasil bahwa kesalahan maksimal yang dihasilkan oleh sensor rotari sebesar 0,39 % dari 256 pulsa yang telah dirancang untuk satu putaran. Kesalahan ini disebabkan oleh adanya roda gigi pada sensor rotari yang kurang rapat sehingga terjadi selip antara roda dengan piringan berlubang dari sensor rotari. Kesalahan yang terjadi pada rotari ini dapat ditoleransi mengingat kecilnya persentase kesalahan dan sensor rotari dapat digunakan dalam sistem.

## 5.3 Pengujian Sensor Rotari terhadap Jarak

Pengujian ini bertujuan untuk mengetahui kesesuaian data pulsa sensor rotari terhadap jarak tempuh robot. Pada pengujian, sensor rotari terpasang pada model robot yang akan digerakkan lurus sejauh tiga meter. Jarak ini dipilih karena merupakan jarak terjauh dari lapangan.

Dua buah sensor rotari yang terpasang pada masing-masing roda kanan dan kiri dihubungkan dengan mikrokontroler yang berfungsi sebagai pencacah pulsa. Daftar kode program penghitung pulsa sensor rotari yang digunakan pada pengujian ini terdapat pada Lampiran II. Mikrokontroler akan menampilkan jumlah pulsa masing-masing sensor rotari kanan dan kiri pada LCD. Gambar 5.5 menunjukkan tampilan LCD. Pada pengujian ini, model robot didorong lurus sejauh tiga meter kemudian diamati pulsa yang terbaca.


Gambar 5.5. Tampilan LCD.

Hasil pengujian yang diperoleh setelah dilakukan sepuluh kali pengambilan data ditunjukkan dalam Tabel 5.2.

Tabel 5.2. Data pengujian sensor rotari terhadap jarak

| Pengujian ke- | Jumlah pulsa rotari | | Perhitungan jarak | | Kesalahan rotari | | Persentase kesalahan rotari | |
|---|---|---|---|---|---|---|---|---|
| | Kanan | Kiri | Kanan (cm) | Kiri (cm) | Kanan (cm) | Kiri (cm) | Kanan (%) | Kiri (%) |
| 1 | 2573 | 2572 | 299,81 | 299,70 | 0,19 | 0,30 | 0,06 | 0,10 |
| 2 | 2569 | 2577 | 299,35 | 300,28 | 0,65 | 0,28 | 0,22 | 0,09 |
| 3 | 2569 | 2575 | 299,35 | 300,05 | 0,65 | 0,05 | 0,22 | 0,02 |
| 4 | 2571 | 2575 | 299,58 | 300,05 | 0,42 | 0,05 | 0,14 | 0,02 |
| 5 | 2568 | 2571 | 299,23 | 299,58 | 0,77 | 0,42 | 0,26 | 0,14 |
| 6 | 2567 | 2572 | 299,12 | 299,70 | 0,88 | 0,30 | 0,29 | 0,10 |
| 7 | 2567 | 2571 | 299,12 | 299,58 | 0,88 | 0,42 | 0,29 | 0,14 |
| 8 | 2569 | 2571 | 299,35 | 299,58 | 0,65 | 0,42 | 0,22 | 0,14 |
| 9 | 2568 | 2572 | 299,23 | 299,70 | 0,77 | 0,30 | 0,26 | 0,10 |
| 10 | 2573 | 2573 | 299,81 | 299,81 | 0,19 | 0,19 | 0,06 | 0,06 |

Data pulsa rotari kanan dan kiri yang diperoleh kemudian diolah menjadi data jarak dengan menggunakan persamaan sebagai berikut.

$$jarak = \frac{jumlah\ pulsa}{pulsa\ per\ rotasi}\ x\ 2\pi\ x\ (jari - jari\ roda) \quad (5\text{-}2)$$

Berdasarkan perancangan, jari-jari roda model robot yang digunakan sebesar 4,75 cm dan jumlah pulsa per rotasi sebanyak 256 pulsa. Hasil perhitungan dari Tabel 5.2 menunjukkan nilai kesalahan terbesar pada rotari kanan sebesar 0,88 cm atau setara dengan 0,29 % dan rotari kiri sebesar 0,42 cm atau setara dengan 0,14 %. Kesalahan yang terjadi disebabkan oleh terdapatnya ruang antar roda gigi pada sistem mekanik sensor rotari. Ruang antar roda gigi ini dapat menimbulkan selip antara roda dengan piringan berlubang pada sensor. Kesalahan yang dihasilkan sensor rotari ini relatif kecil sehingga dapat ditoleransi dan sensor rotari dapat digunakan dalam sistem.

## 5.4 Pengujian *SD Card*

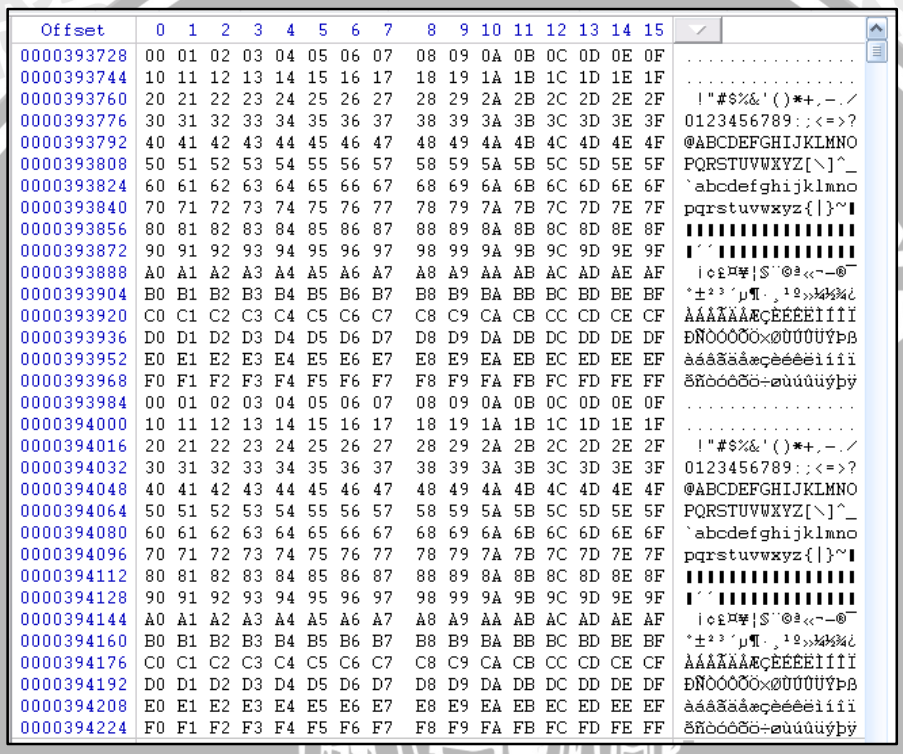
Pengujian *SD Card* terdiri dari dua pengujian yakni pengujian penulisan *SD Card* dan pembacaan *SD Card*.

### 5.5.1 Pengujian Penulisan *SD Card*

Pengujian ini bertujuan untuk mengetahui apakah *SD Card* dapat bekerja dengan baik pada operasi tulis. Pengujian ini juga bertujuan untuk mengetahui apakah perangkat lunak mikrokontroler yang digunakan untuk menulis *SD Card* dapat bekerja dengan baik. Pengujian dilakukan dengan menuliskan data 0,1,2,..,255,0,1,2,..,255 ke

dalam sektor 769 *SD Card*. Prosedur yang dilakukan pada pengujian ini adalah menghubungkan *SD Card* dengan mikrokontroler yang sudah terisi perangkat lunak untuk menulis *SD Card*. Data pada sektor *SD Card* yang bersangkutan akan diperiksa dengan menggunakan dengan menggunakan *software* WinHex . Daftar kode program pengujian *SD Card* ini terdapat pada Lampiran II.

Hasil pengujian diperoleh dengan melihat data yang tersimpan dalam *SD Card* dengan menggunakan *software* WinHex. Gambar 5.6 menunjukkan data tersimpan dalam *SD Card* pada alamat 393728 atau sektor 769.



| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000393728 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 0000393744 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| 0000393760 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |
| 0000393776 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F |
| 0000393792 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F |
| 0000393808 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F |
| 0000393824 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F |
| 0000393840 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F |
| 0000393856 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F |
| 0000393872 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F |
| 0000393888 | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF |
| 0000393904 | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | BA | BB | BC | BD | BE | BF |
| 0000393920 | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF |
| 0000393936 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF |
| 0000393952 | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF |
| 0000393968 | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | FA | FB | FC | FD | FE | FF |
| 0000393984 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 0000394000 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| 0000394016 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |
| 0000394032 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F |
| 0000394048 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F |
| 0000394064 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F |
| 0000394080 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F |
| 0000394096 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F |
| 0000394112 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F |
| 0000394128 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F |
| 0000394144 | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF |
| 0000394160 | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | BA | BB | BC | BD | BE | BF |
| 0000394176 | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF |
| 0000394192 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF |
| 0000394208 | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF |
| 0000394224 | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | FA | FB | FC | FD | FE | FF |

Gambar 5.6. Hasil pengujian penulisan *SD Card* .

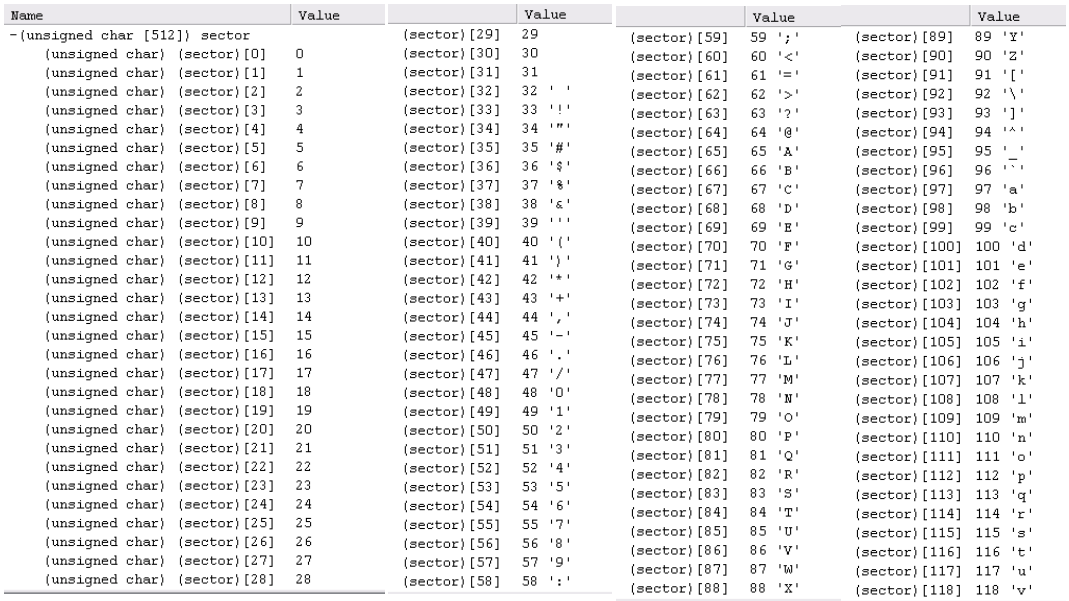
Berdasarkan Gambar 5.6 dapat terlihat bahwa data yang tersimpan dalam *SD Card* sesuai dengan data yang diisikan oleh perangkat lunak mikrokontroler. Data yang tersimpan adalah nilai 0,1,2,..,254,255,0,1,2,..,254,255 yang ditulis pada sektor 769 secara berurutan. Hal ini menunjukkan bahwa *SD Card* dapat bekerja dengan baik pada operasi tulis dan perangkat lunak yang digunakan untuk menulis *SD Card* dapat bekerja dengan baik. Berdasar pengujian ini maka perangkat lunak yang berfungsi untuk menulis *SD Card* dapat digunakan pada sistem.

### 5.5.2 Pengujian Pembacaan *SD Card*

Pengujian ini bertujuan untuk mengetahui apakah *SD Card* dapat bekerja dengan baik pada operasi baca. Pengujian ini juga bertujuan untuk mengetahui apakah

perangkat lunak yang digunakan untuk membaca *SD Card* dapat bekerja dengan baik. Pengujian dilakukan dengan melakukan proses pembacaan pada sektor 769 *SD Card*. Pengujian ini merupakan lanjutan dari pengujian penulisan *SD Card*. Data yang telah dituliskan dalam *SD Card* akan dibaca dengan menggunakan mikrokontroler. Daftar kode program pengujian *SD Card* ini terdapat pada Lampiran II. Data hasil pembacaan akan disimpan dalam mikrokontroler. Data tersebut kemudian diperiksa nilai dengan memanfaatkan fasilitas FoUSB/UART *debugger* pada *High-performance Embedded Workshop*.

Gambar 5.7 sampai 5.10 menunjukkan nilai hasil pembacaan data *SD Card* yang tersimpan dalam mikrokontroler.



| Name | Value |
|---|---|
| -(unsigned char [512]) sector | |
| (unsigned char) (sector)[0] | 0 |
| (unsigned char) (sector)[1] | 1 |
| (unsigned char) (sector)[2] | 2 |
| (unsigned char) (sector)[3] | 3 |
| (unsigned char) (sector)[4] | 4 |
| (unsigned char) (sector)[5] | 5 |
| (unsigned char) (sector)[6] | 6 |
| (unsigned char) (sector)[7] | 7 |
| (unsigned char) (sector)[8] | 8 |
| (unsigned char) (sector)[9] | 9 |
| (unsigned char) (sector)[10] | 10 |
| (unsigned char) (sector)[11] | 11 |
| (unsigned char) (sector)[12] | 12 |
| (unsigned char) (sector)[13] | 13 |
| (unsigned char) (sector)[14] | 14 |
| (unsigned char) (sector)[15] | 15 |
| (unsigned char) (sector)[16] | 16 |
| (unsigned char) (sector)[17] | 17 |
| (unsigned char) (sector)[18] | 18 |
| (unsigned char) (sector)[19] | 19 |
| (unsigned char) (sector)[20] | 20 |
| (unsigned char) (sector)[21] | 21 |
| (unsigned char) (sector)[22] | 22 |
| (unsigned char) (sector)[23] | 23 |
| (unsigned char) (sector)[24] | 24 |
| (unsigned char) (sector)[25] | 25 |
| (unsigned char) (sector)[26] | 26 |
| (unsigned char) (sector)[27] | 27 |
| (unsigned char) (sector)[28] | 28 |

| | Value |
|---|---|
| (sector)[29] | 29 |
| (sector)[30] | 30 |
| (sector)[31] | 31 |
| (sector)[32] | 32 ' ' |
| (sector)[33] | 33 '!' |
| (sector)[34] | 34 '"' |
| (sector)[35] | 35 '#' |
| (sector)[36] | 36 '$' |
| (sector)[37] | 37 '%' |
| (sector)[38] | 38 '&' |
| (sector)[39] | 39 ''' |
| (sector)[40] | 40 '(' |
| (sector)[41] | 41 ')' |
| (sector)[42] | 42 '*' |
| (sector)[43] | 43 '+' |
| (sector)[44] | 44 ',' |
| (sector)[45] | 45 '-' |
| (sector)[46] | 46 '.' |
| (sector)[47] | 47 '/' |
| (sector)[48] | 48 '0' |
| (sector)[49] | 49 '1' |
| (sector)[50] | 50 '2' |
| (sector)[51] | 51 '3' |
| (sector)[52] | 52 '4' |
| (sector)[53] | 53 '5' |
| (sector)[54] | 54 '6' |
| (sector)[55] | 55 '7' |
| (sector)[56] | 56 '8' |
| (sector)[57] | 57 '9' |
| (sector)[58] | 58 ':' |

| | Value | | Value |
|---|---|---|---|
| (sector)[59] | 59 ';' | (sector)[89] | 89 'Y' |
| (sector)[60] | 60 '<' | (sector)[90] | 90 'Z' |
| (sector)[61] | 61 '=' | (sector)[91] | 91 '[' |
| (sector)[62] | 62 '>' | (sector)[92] | 92 '\' |
| (sector)[63] | 63 '?' | (sector)[93] | 93 ']' |
| (sector)[64] | 64 '@' | (sector)[94] | 94 '^' |
| (sector)[65] | 65 'A' | (sector)[95] | 95 '_' |
| (sector)[66] | 66 'B' | (sector)[96] | 96 '`' |
| (sector)[67] | 67 'C' | (sector)[97] | 97 'a' |
| (sector)[68] | 68 'D' | (sector)[98] | 98 'b' |
| (sector)[69] | 69 'E' | (sector)[99] | 99 'c' |
| (sector)[70] | 70 'F' | (sector)[100] | 100 'd' |
| (sector)[71] | 71 'G' | (sector)[101] | 101 'e' |
| (sector)[72] | 72 'H' | (sector)[102] | 102 'f' |
| (sector)[73] | 73 'I' | (sector)[103] | 103 'g' |
| (sector)[74] | 74 'J' | (sector)[104] | 104 'h' |
| (sector)[75] | 75 'K' | (sector)[105] | 105 'i' |
| (sector)[76] | 76 'L' | (sector)[106] | 106 'j' |
| (sector)[77] | 77 'M' | (sector)[107] | 107 'k' |
| (sector)[78] | 78 'N' | (sector)[108] | 108 'l' |
| (sector)[79] | 79 'O' | (sector)[109] | 109 'm' |
| (sector)[80] | 80 'P' | (sector)[110] | 110 'n' |
| (sector)[81] | 81 'Q' | (sector)[111] | 111 'o' |
| (sector)[82] | 82 'R' | (sector)[112] | 112 'p' |
| (sector)[83] | 83 'S' | (sector)[113] | 113 'q' |
| (sector)[84] | 84 'T' | (sector)[114] | 114 'r' |
| (sector)[85] | 85 'U' | (sector)[115] | 115 's' |
| (sector)[86] | 86 'V' | (sector)[116] | 116 't' |
| (sector)[87] | 87 'W' | (sector)[117] | 117 'u' |
| (sector)[88] | 88 'X' | (sector)[118] | 118 'v' |

Gambar 5.7. Hasil pengujian pembacaan *SD Card*.

| | Value | | Value | | Value | | Value | | Value |
|---|---|---|---|---|---|---|---|---|---|
| (sector)[119] | 119 'w' | (sector)[149] | 149 | (sector)[179] | 179 | (sector)[209] | 209 | (sector)[239] | 239 |
| (sector)[120] | 120 'x' | (sector)[150] | 150 | (sector)[180] | 180 | (sector)[210] | 210 | (sector)[240] | 240 |
| (sector)[121] | 121 'y' | (sector)[151] | 151 | (sector)[181] | 181 | (sector)[211] | 211 | (sector)[241] | 241 |
| (sector)[122] | 122 'z' | (sector)[152] | 152 | (sector)[182] | 182 | (sector)[212] | 212 | (sector)[242] | 242 |
| (sector)[123] | 123 '{' | (sector)[153] | 153 | (sector)[183] | 183 | (sector)[213] | 213 | (sector)[243] | 243 |
| (sector)[124] | 124 '\|' | (sector)[154] | 154 | (sector)[184] | 184 | (sector)[214] | 214 | (sector)[244] | 244 |
| (sector)[125] | 125 '}' | (sector)[155] | 155 | (sector)[185] | 185 | (sector)[215] | 215 | (sector)[245] | 245 |
| (sector)[126] | 126 '~' | (sector)[156] | 156 | (sector)[186] | 186 | (sector)[216] | 216 | (sector)[246] | 246 |
| (sector)[127] | 127 | (sector)[157] | 157 | (sector)[187] | 187 | (sector)[217] | 217 | (sector)[247] | 247 |
| (sector)[128] | 128 | (sector)[158] | 158 | (sector)[188] | 188 | (sector)[218] | 218 | (sector)[248] | 248 |
| (sector)[129] | 129 | (sector)[159] | 159 | (sector)[189] | 189 | (sector)[219] | 219 | (sector)[249] | 249 |
| (sector)[130] | 130 | (sector)[160] | 160 | (sector)[190] | 190 | (sector)[220] | 220 | (sector)[250] | 250 |
| (sector)[131] | 131 | (sector)[161] | 161 | (sector)[191] | 191 | (sector)[221] | 221 | (sector)[251] | 251 |
| (sector)[132] | 132 | (sector)[162] | 162 | (sector)[192] | 192 | (sector)[222] | 222 | (sector)[252] | 252 |
| (sector)[133] | 133 | (sector)[163] | 163 | (sector)[193] | 193 | (sector)[223] | 223 | (sector)[253] | 253 |
| (sector)[134] | 134 | (sector)[164] | 164 | (sector)[194] | 194 | (sector)[224] | 224 | (sector)[254] | 254 |
| (sector)[135] | 135 | (sector)[165] | 165 | (sector)[195] | 195 | (sector)[225] | 225 | (sector)[255] | 255 |
| (sector)[136] | 136 | (sector)[166] | 166 | (sector)[196] | 196 | (sector)[226] | 226 | (sector)[256] | 0 |
| (sector)[137] | 137 | (sector)[167] | 167 | (sector)[197] | 197 | (sector)[227] | 227 | (sector)[257] | 1 |
| (sector)[138] | 138 | (sector)[168] | 168 | (sector)[198] | 198 | (sector)[228] | 228 | (sector)[258] | 2 |
| (sector)[139] | 139 | (sector)[169] | 169 | (sector)[199] | 199 | (sector)[229] | 229 | (sector)[259] | 3 |
| (sector)[140] | 140 | (sector)[170] | 170 | (sector)[200] | 200 | (sector)[230] | 230 | (sector)[260] | 4 |
| (sector)[141] | 141 | (sector)[171] | 171 | (sector)[201] | 201 | (sector)[231] | 231 | (sector)[261] | 5 |
| (sector)[142] | 142 | (sector)[172] | 172 | (sector)[202] | 202 | (sector)[232] | 232 | (sector)[262] | 6 |
| (sector)[143] | 143 | (sector)[173] | 173 | (sector)[203] | 203 | (sector)[233] | 233 | (sector)[263] | 7 |
| (sector)[144] | 144 | (sector)[174] | 174 | (sector)[204] | 204 | (sector)[234] | 234 | (sector)[264] | 8 |
| (sector)[145] | 145 | (sector)[175] | 175 | (sector)[205] | 205 | (sector)[235] | 235 | (sector)[265] | 9 |
| (sector)[146] | 146 | (sector)[176] | 176 | (sector)[206] | 206 | (sector)[236] | 236 | (sector)[266] | 10 |
| (sector)[147] | 147 | (sector)[177] | 177 | (sector)[207] | 207 | (sector)[237] | 237 | (sector)[267] | 11 |
| (sector)[148] | 148 | (sector)[178] | 178 | (sector)[208] | 208 | (sector)[238] | 238 | (sector)[268] | 12 |

Gambar 5.8. Hasil pengujian pembacaan *SD Card* (lanjutan 1).

| | Value | | Value | | Value | | Value | | Value |
|---|---|---|---|---|---|---|---|---|---|
| (sector)[269] | 13 | (sector)[299] | 43 '+' | (sector)[329] | 73 'I' | (sector)[359] | 103 'g' | (sector)[389] | 133 |
| (sector)[270] | 14 | (sector)[300] | 44 ',' | (sector)[330] | 74 'J' | (sector)[360] | 104 'h' | (sector)[390] | 255 |
| (sector)[271] | 15 | (sector)[301] | 45 '-' | (sector)[331] | 75 'K' | (sector)[361] | 105 'i' | (sector)[391] | 255 |
| (sector)[272] | 16 | (sector)[302] | 46 '.' | (sector)[332] | 76 'L' | (sector)[362] | 106 'j' | (sector)[392] | 136 |
| (sector)[273] | 17 | (sector)[303] | 47 '/' | (sector)[333] | 77 'M' | (sector)[363] | 107 'k' | (sector)[393] | 137 |
| (sector)[274] | 18 | (sector)[304] | 48 '0' | (sector)[334] | 78 'N' | (sector)[364] | 108 'l' | (sector)[394] | 138 |
| (sector)[275] | 19 | (sector)[305] | 49 '1' | (sector)[335] | 79 'O' | (sector)[365] | 109 'm' | (sector)[395] | 139 |
| (sector)[276] | 20 | (sector)[306] | 50 '2' | (sector)[336] | 80 'P' | (sector)[366] | 110 'n' | (sector)[396] | 140 |
| (sector)[277] | 21 | (sector)[307] | 51 '3' | (sector)[337] | 81 'Q' | (sector)[367] | 111 'o' | (sector)[397] | 141 |
| (sector)[278] | 22 | (sector)[308] | 52 '4' | (sector)[338] | 82 'R' | (sector)[368] | 112 'p' | (sector)[398] | 142 |
| (sector)[279] | 23 | (sector)[309] | 53 '5' | (sector)[339] | 83 'S' | (sector)[369] | 113 'q' | (sector)[399] | 143 |
| (sector)[280] | 24 | (sector)[310] | 54 '6' | (sector)[340] | 84 'T' | (sector)[370] | 114 'r' | (sector)[400] | 144 |
| (sector)[281] | 25 | (sector)[311] | 55 '7' | (sector)[341] | 85 'U' | (sector)[371] | 115 's' | (sector)[401] | 145 |
| (sector)[282] | 26 | (sector)[312] | 56 '8' | (sector)[342] | 86 'V' | (sector)[372] | 116 't' | (sector)[402] | 146 |
| (sector)[283] | 27 | (sector)[313] | 57 '9' | (sector)[343] | 87 'W' | (sector)[373] | 117 'u' | (sector)[403] | 147 |
| (sector)[284] | 28 | (sector)[314] | 58 ':' | (sector)[344] | 88 'X' | (sector)[374] | 118 'v' | (sector)[404] | 148 |
| (sector)[285] | 29 | (sector)[315] | 59 ';' | (sector)[345] | 89 'Y' | (sector)[375] | 119 'w' | (sector)[405] | 149 |
| (sector)[286] | 30 | (sector)[316] | 60 '<' | (sector)[346] | 90 'Z' | (sector)[376] | 120 'x' | (sector)[406] | 150 |
| (sector)[287] | 31 | (sector)[317] | 61 '=' | (sector)[347] | 91 '[' | (sector)[377] | 121 'y' | (sector)[407] | 151 |
| (sector)[288] | 32 ' ' | (sector)[318] | 62 '>' | (sector)[348] | 92 '\' | (sector)[378] | 122 'z' | (sector)[408] | 152 |
| (sector)[289] | 33 '!' | (sector)[319] | 63 '?' | (sector)[349] | 93 ']' | (sector)[379] | 123 '{' | (sector)[409] | 153 |
| (sector)[290] | 34 '"' | (sector)[320] | 64 '@' | (sector)[350] | 94 '^' | (sector)[380] | 124 '\|' | (sector)[410] | 154 |
| (sector)[291] | 35 '#' | (sector)[321] | 65 'A' | (sector)[351] | 95 '_' | (sector)[381] | 125 '}' | (sector)[411] | 155 |
| (sector)[292] | 36 '$' | (sector)[322] | 66 'B' | (sector)[352] | 96 '`' | (sector)[382] | 126 '~' | (sector)[412] | 156 |
| (sector)[293] | 37 '%' | (sector)[323] | 67 'C' | (sector)[353] | 97 'a' | (sector)[383] | 127 | (sector)[413] | 157 |
| (sector)[294] | 38 '&' | (sector)[324] | 68 'D' | (sector)[354] | 98 'b' | (sector)[384] | 128 | (sector)[414] | 158 |
| (sector)[295] | 39 ''' | (sector)[325] | 69 'E' | (sector)[355] | 99 'c' | (sector)[385] | 129 | (sector)[415] | 159 |
| (sector)[296] | 40 '(' | (sector)[326] | 70 'F' | (sector)[356] | 100 'd' | (sector)[386] | 130 | (sector)[416] | 160 |
| (sector)[297] | 41 ')' | (sector)[327] | 71 'G' | (sector)[357] | 101 'e' | (sector)[387] | 131 | (sector)[417] | 161 |
| (sector)[298] | 42 '*' | (sector)[328] | 72 'H' | (sector)[358] | 102 'f' | (sector)[388] | 132 | (sector)[418] | 162 |

Gambar 5.9. Hasil pengujian pembacaan *SD Card* (lanjutan 2).

| | Value | | Value | | Value | | Value |
|---|---|---|---|---|---|---|---|
| (sector)[419] | 163 | (sector)[449] | 193 | (sector)[479] | 223 | (sector)[509] | 253 |
| (sector)[420] | 164 | (sector)[450] | 194 | (sector)[480] | 224 | (sector)[510] | 254 |
| (sector)[421] | 165 | (sector)[451] | 195 | (sector)[481] | 225 | (sector)[511] | 255 |
| (sector)[422] | 166 | (sector)[452] | 196 | (sector)[482] | 226 | | |
| (sector)[423] | 167 | (sector)[453] | 197 | (sector)[483] | 227 | | |
| (sector)[424] | 168 | (sector)[454] | 198 | (sector)[484] | 228 | | |
| (sector)[425] | 169 | (sector)[455] | 199 | (sector)[485] | 229 | | |
| (sector)[426] | 170 | (sector)[456] | 200 | (sector)[486] | 230 | | |
| (sector)[427] | 171 | (sector)[457] | 201 | (sector)[487] | 231 | | |
| (sector)[428] | 172 | (sector)[458] | 202 | (sector)[488] | 232 | | |
| (sector)[429] | 173 | (sector)[459] | 203 | (sector)[489] | 233 | | |
| (sector)[430] | 174 | (sector)[460] | 204 | (sector)[490] | 234 | | |
| (sector)[431] | 175 | (sector)[461] | 205 | (sector)[491] | 235 | | |
| (sector)[432] | 176 | (sector)[462] | 206 | (sector)[492] | 236 | | |
| (sector)[433] | 177 | (sector)[463] | 207 | (sector)[493] | 237 | | |
| (sector)[434] | 178 | (sector)[464] | 208 | (sector)[494] | 238 | | |
| (sector)[435] | 179 | (sector)[465] | 209 | (sector)[495] | 239 | | |
| (sector)[436] | 180 | (sector)[466] | 210 | (sector)[496] | 240 | | |
| (sector)[437] | 181 | (sector)[467] | 211 | (sector)[497] | 241 | | |
| (sector)[438] | 182 | (sector)[468] | 212 | (sector)[498] | 242 | | |
| (sector)[439] | 183 | (sector)[469] | 213 | (sector)[499] | 243 | | |
| (sector)[440] | 184 | (sector)[470] | 214 | (sector)[500] | 244 | | |
| (sector)[441] | 185 | (sector)[471] | 215 | (sector)[501] | 245 | | |
| (sector)[442] | 186 | (sector)[472] | 216 | (sector)[502] | 246 | | |
| (sector)[443] | 187 | (sector)[473] | 217 | (sector)[503] | 247 | | |
| (sector)[444] | 188 | (sector)[474] | 218 | (sector)[504] | 248 | | |
| (sector)[445] | 189 | (sector)[475] | 219 | (sector)[505] | 249 | | |
| (sector)[446] | 190 | (sector)[476] | 220 | (sector)[506] | 250 | | |
| (sector)[447] | 191 | (sector)[477] | 221 | (sector)[507] | 251 | | |
| (sector)[448] | 192 | (sector)[478] | 222 | (sector)[508] | 252 | | |

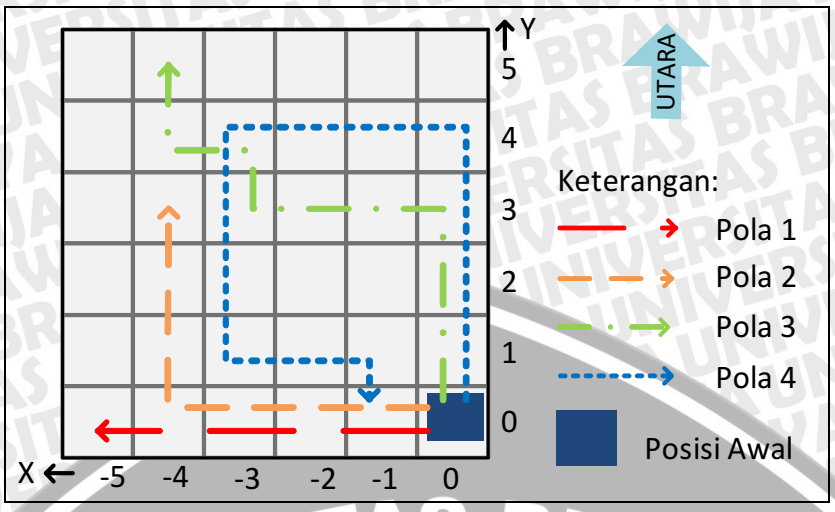
Gambar 5.10. Hasil pengujian pembacaan *SD Card* (lanjutan 3).

Berdasarkan Gambar 5.7 sampai 5.10 dapat terlihat bahwa data yang tersimpan dalam mikrokontroler sesuai dengan data yang dituliskan ke dalam sektor 769 *SD Card* pada pengujian penulisan *SD Card*. Hal tersebut menunjukkan bahwa *SD Card* dapat bekerja dengan baik pada operasi baca dan perangkat lunak yang digunakan untuk membaca *SD Card* dapat bekerja dengan baik. Berdasar pengujian ini maka perangkat lunak yang berfungsi untuk membaca *SD Card* dapat digunakan pada sistem.

## 5.5 Pengujian Perhitungan Posisi

Pengujian perhitungan posisi bertujuan untuk mengetahui kesesuaian data posisi blok yang dihasilkan dengan menggunakan proses perhitungan data rotari dengan posisi blok model robot sebenarnya. Pengujian ini dilakukan pada pola pergerakan model robot tanpa sistem koreksi. Hal ini dilakukan karena pada sistem koreksi data posisi akan dikembalikan sesuai dengan nilai tengah dari blok sehingga tidak dapat diketahui kebenaran data hasil perhitungan sesungguhnya.

Pemilihan pola pergerakan didasarkan pada pola pergerakan terjauh tanpa sistem koreksi yang memungkinkan dari setiap kombinasi lapangan. Pola pergerakan terjauh dari masing-masing lapangan diseleksi dan dipilih pergerakan yang memiliki pola dasar berbeda. Gambar 5.11 menunjukkan pola pergerakan yang digunakan pada pengujian perhitungan posisi.




Gambar 5.11. Pola pergerakan pengujian perhitungan posisi.

Pengujian dilakukan dengan mendorong model robot bergerak sesuai dengan pola pergerakan yang telah ditentukan. Daftar kode program pengujian terdapat dalam Lampiran II. Data posisi blok yang diperoleh dari perhitungan kemudian ditampilkan pada LCD. Gambar 5.12 menunjukkan tampilan LCD data posisi blok model robot. Tampilan LCD akan dicatat dan diperiksa kebenarannya pada saat model robot telah mencapai titik terakhir dari masing-masing pola pergerakan. Hasil pengujian setelah dilakukan beberapa kali ditunjukkan dalam Tabel 5.3.


Gambar 5.12. Tampilan LCD pengujian perhitungan posisi.

Tabel 5.3. Hasil pengujian perhitungan posisi.

| Pengujian ke- | Pola 1 | | | | Pola 2 | | | | Pola 3 | | | | Pola 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X | Y | Arah | Ket. | X | Y | Arah | Ket. | X | Y | Arah | Ket. | X | Y | Arah | Ket. |
| 1 | -5 | 0 | B | V | -4 | 3 | U | V | -4 | 5 | U | V | -1 | 0 | S | V |
| 2 | -5 | 0 | B | V | -4 | 3 | U | V | -4 | 5 | U | V | -1 | 0 | S | V |
| 3 | -5 | 0 | B | V | -4 | 3 | U | V | -4 | 5 | U | V | -1 | 0 | S | V |
| 4 | -5 | 0 | B | V | -4 | 3 | U | V | -4 | 5 | U | V | -1 | 0 | S | V |
| 5 | -5 | 0 | B | V | -4 | 3 | U | V | -4 | 5 | U | V | -1 | 0 | S | V |

Dengan:

U = utara T = timur V = benar

B = barat S = selatan XX = salah

Berdasarkan hasil pengujian diperoleh bahwa data posisi blok hasil perhitungan sesuai dengan posisi blok pada keadaan yang sebenarnya. Dengan demikian perhitungan

posisi model robot sudah benar dan perangkat lunak untuk menghitung posisi dapat diterapkan pada sistem.

## 5.6 Pengujian Waktu Satu Siklus

Pengujian ini bertujuan untuk mengetahui waktu yang diperlukan untuk melakukan satu kali siklus proses, yakni pembacaan data pulsa masing-masing sensor rotari dan tombol, pengolahan data, serta penyimpanan basis data pemetaan ke dalam *SD Card*. Pengujian dilakukan dengan merangkai alat secara keseluruhan.


Gambar 5.13. Tampilan LCD pengujian waktu satu siklus.

Pengujian ini memanfaatkan fasilitas Timer Z yang terdapat dalam mikrokontroler Renesas R8C/13 sebagai pewaktu. Daftar kode program pengujian terdapat dalam Lampiran II. Pada pengujian, data timer yang tersimpan dalam register TZPR dan PREZ akan dibaca. Selain kedua register tersebut, ditambahkan pula variabel UNDERF*LOW* yang berfungsi sebagai penghitung apabila data yang tersimpan pada register TZPR dan PREZ melebihi batas bawah. Data pada register TZPR, PREZ dan variabel UNDERF*LOW* merupakan pencacah mundur dari 255 menjadi nilai terkecil. Sebelum memulai melakukan proses, register TZPR, PREZ, dan variabel UNDERF*LOW* akan diberi nilai 255 kemudian timer Z akan diaktifkan dengan menggunakan frekuensi timer 2,5 MHz. Timer Z akan dimatikan pada akhir proses kemudian nilai register TZPR, PREZ, dan variabel UNDERF*LOW* akan ditampilkan pada LCD seperti dalam Gambar 5.13. Hasil yang diperoleh setelah dilakukan pengujian beberapa kali ditunjukkan dalam Tabel 5.4.

Tabel 5.4. Hasil pengujian waktu yang diperlukan satu siklus proses.

| Pengujian ke- | UNDERF*LOW* | TZ | PREZ | Waktu (s) |
|---|---|---|---|---|
| 1 | 253 | 205 | 53 | 0,057630 |
| 2 | 253 | 198 | 66 | 0,058341 |
| 3 | 253 | 198 | 240 | 0,058272 |
| 4 | 253 | 207 | 163 | 0,057381 |
| 5 | 253 | 214 | 2 | 0,056728 |
| 6 | 253 | 199 | 133 | 0,058212 |
| 7 | 253 | 207 | 64 | 0,057420 |

| Pengujian ke- | UNDERF*LOW* | TZ | PREZ | Waktu (s) |
|---|---|---|---|---|
| 8 | 253 | 205 | 227 | 0,057560 |
| 9 | 253 | 207 | 15 | 0,057440 |
| 10 | 253 | 205 | 227 | 0,057560 |

Nilai waktu diperoleh dengan melakukan perhitungan terhadap data yang diperoleh. Persamaan (5-3) digunakan untuk menentukan nilai waktu dalam satu siklus proses.

$$waktu = \frac{(65536.(255-underflow)+256.(255-tz)+(255-prez))}{f_{timer}} \quad (5\text{-}3)$$

Berdasarkan hasil pengujian diperoleh waktu terlama yang diperlukan dalam mengerjakan satu siklus proses sebesar 0,058 detik. Pada pengujian kecepatan respon sensor rotari diketahui bahwa kecepatan maksimal robot yang dapat menggunakan sistem ini sebesar 3,5 m/s. Jarak terkecil yang dapat dipantau oleh sistem dapat diketahui dengan menggunakan persamaan berikut.

$$jarak = kecepatan\ x\ waktu = 3{,}5\ {}^{m}/_{s}\ x\ 0{,}058\ s = 0{,}203\ m = 20{,}3\ cm \quad (5\text{-}4)$$

Nilai jarak tersebut lebih kecil dari setengah ukuran satu blok yakni 50 cm, sehingga sistem masih dapat memantau kondisi tiap-tiap blok lapangan. Hal ini memungkinkan sistem dapat bekerja dengan baik pada kecepatan maksimal robot.

## 5.7 Pengujian Keseluruhan

Pengujian ini bertujuan untuk menganalisis kerja sistem secara keseluruhan apakah sesuai dengan perancangan. Pengujian dilakukan dengan menggerakkan model robot pada dua kombinasi lapangan bawah. Kombinasi lapangan bawah yang digunakan pada pengujian adalah kombinasi B4 dan B17.

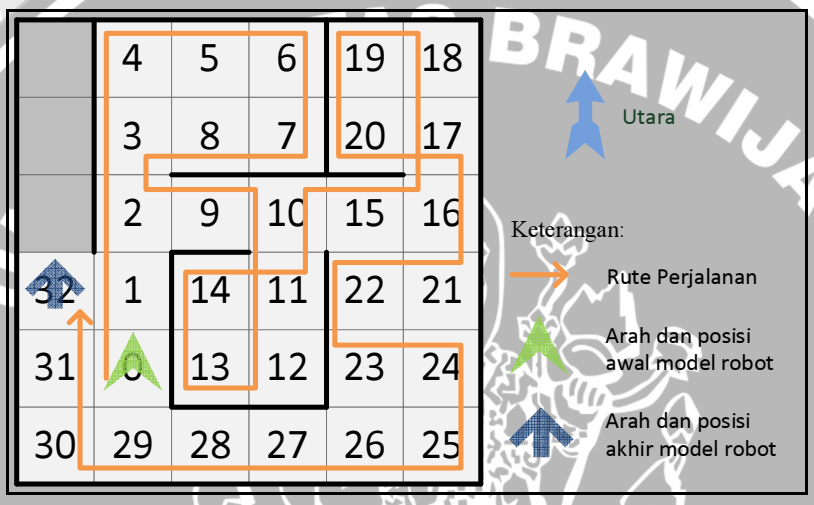
Mula-mula model robot diletakkan pada posisi awal (*home*) dan orientasi sesuai dengan kombinasi lapangan yang digunakan. Kemudian model robot digerakkan sesuai dengan rute pergerakan yang telah direncanakan. Pergerakan pada model robot dilakukan dengan cara didorong secara manual. Pada tiap-tiap blok lapangan diambil posisi dinding pada blok yang bersangkutan melalui penekanan tombol sesuai dengan posisi dinding. Pergerakan robot dilakukan hingga semua blok dijelajahi oleh model robot.

Setelah model robot berhenti pada blok terakhir, *SD Card* yang terpasang pada model robot dilepas untuk diperiksa basis data yang terbentuk. *SD Card* diperiksa

dengan membuka *file* "PETA.TXT" menggunakan media komputer. Basis data yang terbentuk kemudian dicocokkan dengan kondisi lapangan yang sebenarnya.

### 5.7.1 Pengujian Lapangan B4

Pada pengujian, model robot digerakkan sesuai dengan pola pergerakan yang telah direncanakan. Tiap-tiap blok yang sudah dijelajahi oleh model robot diberi nomer. Penomeran masing-masing blok dilakukan secara berurutan sesuai dengan pola pergerakan yang direncanakan. Pola pergerakan dan penomeran masing-masing blok ditunjukkan dalam Gambar 5.14.




Gambar 5.14. Pola pergerakan dan penomeran blok Lapangan B4.

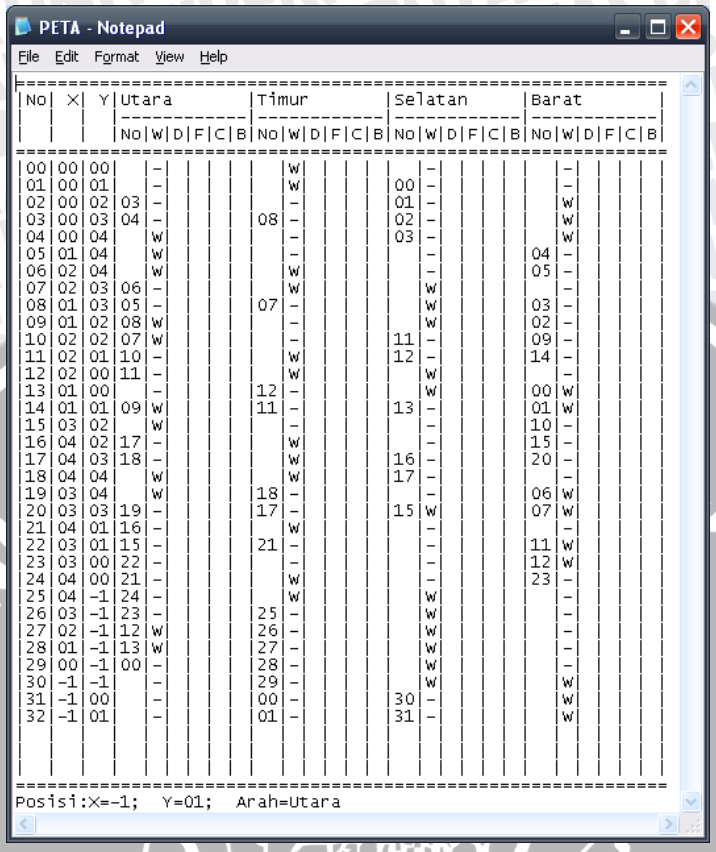
Setelah dilakukan pengujian, basis data yang tersimpan dalam *SD Card* diperiksa dengan menggunakan komputer. Pengujian dilakukan sebanyak lima kali pengujian dengan pola pergerakan yang sama. Pengujian menghasilkan basis data pemetaan yang sama pada tiap-tiap pengujian. Gambar 5.15 menunjukkan tampilan basis data yang dihasilkan oleh sistem pada pengujian ini.

Berdasarkan Gambar 5.15, posisi dinding blok ke-14 adalah pada bagian utara dan barat dari robot. Informasi tersebut sesuai dengan kondisi sebenarnya yang ditunjukkan dalam Gambar 5.14. Pengujian sistem pada lapangan B4 mampu menghasilkan basis data pemetaan yang sesuai dengan kondisi uji.

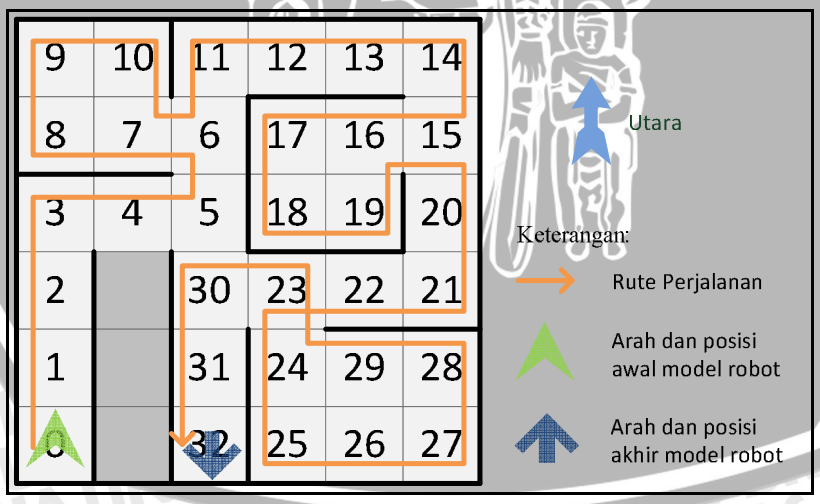
### 5.7.2 Pengujian Lapangan B17

Pada pengujian, model robot digerakkan sesuai dengan pola pergerakan yang telah direncanakan. Tiap-tiap blok yang sudah dijelajahi oleh model robot diberi nomer. Penomeran masing-masing blok dilakukan secara berurutan sesuai dengan pola

pergerakan yang direncanakan. Pola pergerakan dan penomeran masing-masing blok ditunjukkan dalam Gambar 5.16.




Gambar 5.15. Basis data lapangan B4




Gambar 5.16. Pola pergerakan dan penomeran blok Lapangan B17.

Setelah model robot berada pada posisi blok terakhir, *SD Card* sistem diambil dan dibaca dengan komputer. Pembacaan data *SD Card* ini dimaksudkan untuk memeriksa basis data yang tersimpan dalam *SD Card*. Pengujian dilakukan sebanyak lima kali pengujian dengan pola pergerakan yang sama. Hasil pembacaan basis data oleh komputer ditunjukkan dalam Gambar 5.17.

PETA - Notepad

File Edit Format View Help

| No | X | Y | Utara | | | | | Timur | | | | | Selatan | | | | | Barat | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | No | W | D | F | C | B | No | W | D | F | C | B | No | W | D | F | C | B | No | W | D | F | C | B |
| 00 | 00 | 00 | | - | | | | | | - | | | | | | W | | | | | | W | | | | |
| 01 | 00 | 01 | | - | | | | | | W | | | | | 00 | - | | | | | | W | | | | |
| 02 | 00 | 02 | | - | | | | | | W | | | | | 01 | - | | | | | | W | | | | |
| 03 | 00 | 03 | | W | | | | | | - | | | | | 02 | - | | | | | | W | | | | |
| 04 | 01 | 03 | | W | | | | | | - | | | | | | - | | | | | 03 | - | | | | |
| 05 | 02 | 03 | | - | | | | | | W | | | | | | - | | | | | 04 | - | | | | |
| 06 | 02 | 04 | | - | | | | | | W | | | | | 05 | - | | | | | 07 | - | | | | |
| 07 | 01 | 04 | 10 | - | | | | | 06 | - | | | | | 04 | W | | | | | 08 | - | | | | |
| 08 | 00 | 04 | | - | | | | | 07 | - | | | | | 03 | W | | | | | | W | | | | |
| 09 | 00 | 05 | | W | | | | | | - | | | | | 08 | - | | | | | | W | | | | |
| 10 | 01 | 05 | | W | | | | | | W | | | | | 07 | - | | | | | 09 | - | | | | |
| 11 | 02 | 05 | | W | | | | | | - | | | | | 06 | - | | | | | 10 | W | | | | |
| 12 | 03 | 05 | | W | | | | | | - | | | | | | W | | | | | 11 | - | | | | |
| 13 | 04 | 05 | | W | | | | | | - | | | | | | W | | | | | 12 | - | | | | |
| 14 | 05 | 05 | | W | | | | | | W | | | | | | - | | | | | 13 | - | | | | |
| 15 | 05 | 04 | 14 | - | | | | | | W | | | | | | - | | | | | 16 | - | | | | |
| 16 | 04 | 04 | 13 | W | | | | | 15 | - | | | | | 19 | - | | | | | 17 | - | | | | |
| 17 | 03 | 04 | 12 | W | | | | | 16 | - | | | | | | - | | | | | 06 | W | | | | |
| 18 | 03 | 03 | 17 | - | | | | | | - | | | | | | W | | | | | 05 | W | | | | |
| 19 | 04 | 03 | 16 | - | | | | | | W | | | | | | W | | | | | 18 | - | | | | |
| 20 | 05 | 03 | 15 | - | | | | | | W | | | | | | - | | | | | 19 | W | | | | |
| 21 | 05 | 02 | 20 | - | | | | | | W | | | | | | W | | | | | | - | | | | |
| 22 | 04 | 02 | 19 | W | | | | | 21 | - | | | | | | W | | | | | | - | | | | |
| 23 | 03 | 02 | 18 | W | | | | | 22 | - | | | | | 24 | - | | | | | | - | | | | |
| 24 | 03 | 01 | 23 | - | | | | | 29 | - | | | | | 25 | - | | | | | | W | | | | |
| 25 | 03 | 00 | 24 | - | | | | | | - | | | | | | W | | | | | | W | | | | |
| 26 | 04 | 00 | | - | | | | | | - | | | | | | W | | | | | 25 | - | | | | |
| 27 | 05 | 00 | | - | | | | | | W | | | | | | W | | | | | 26 | - | | | | |
| 28 | 05 | 01 | 21 | W | | | | | | W | | | | | 27 | - | | | | | | - | | | | |
| 29 | 04 | 01 | 22 | W | | | | | 28 | - | | | | | 26 | - | | | | | 24 | - | | | | |
| 30 | 02 | 02 | 05 | - | | | | | 23 | - | | | | | | - | | | | | | W | | | | |
| 31 | 02 | 01 | 30 | - | | | | | 24 | W | | | | | | - | | | | | | W | | | | |
| 32 | 02 | 00 | 31 | - | | | | | 25 | W | | | | | | W | | | | | | - | | | | |

Posisi:X=02; Y=00; Arah=Sltan

Gambar 5.17. Basis data lapangan B17.

Berdasarkan Gambar 5.17, koordinat semua blok sesuai dengan pola pergerakan yang direncanakan. Hal tersebut terlihat pada blok ke-30. Sisi utara blok ke-30 adalah blok ke-5, sisi timurnya adalah blok ke-23, sisi selatannya adalah blok ke-31. Pada basis data, sisi selatan blok ke-30 tidak diketahui. Hal tersebut dikarenakan pada saat pengambilan data di blok ke-30, sistem belum mengetahui pergerakan selanjutnya model robot. Pengujian ini menunjukkan bahwa sistem dapat menghasilkan basis data sesuai dengan kondisi uji lapangan B17.

# BAB VI
# KESIMPULAN DAN SARAN

## 6.1 Kesimpulan

Dari hasil perancangan dan pengujian yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut.

1. Sensor rotari dengan menggunakan komponen *optoswitch* dapat bekerja pada 3000 pulsa per detik yang setara dengan kecepatan 3,5 m/s pada robot dengan dengan jari-jari roda 4,75 cm. Sensor rotari mempunyai kesalahan terbesar 0,29% yang disebabkan oleh adanya selip antara roda dengan piringan berlubang.
2. Sistem penentu posisi memanfaatkan mikrokontroler R8C/13 sebagai unit pengolah data sensor rotari. Sistem penentu posisi dapat memperkirakan posisi robot dengan tepat pada jarak terjauh 14 blok. Dimensi blok yang digunakan sebesar 50 cm x 50 cm.
3. Antarmuka mikrokontroler dan *SD Card* dengan memanfaatkan fasilitas UART0 R8C/13 dapat membaca dan menulis memori pada *SD Card* dengan benar. Waktu yang diperlukan untuk mengerjakan satu siklus proses sebesar 0,058 detik.
4. Sistem pemetaan mampu menghasilkan basis data dengan format teks yang tersimpan dalam *SD Card*. Pengujian sistem pemetaan pada dua kombinasi lapangan bawah KRCI Expert Single mampu memetakan lapangan dengan tepat dan menghasilkan basis data pemetaan dengan benar.

## 6.2 Saran

Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah:

1. Perlunya penggunaan mikrokontroler dengan kapasitas memori ROM lebih besar sehingga dapat memungkinkan penambahan perangkat lunak mikrokontroler.
2. Perlu adanya penambahan sensor pendeteksi objek lain seperti *uneven floor*, *furniture*, lilin dan bayi agar sistem lebih teliti dalam membuat basis data.
3. Sistem dapat dikembangkan menjadi sistem penentu pergerakan dengan memanfaatkan basis data yang sudah tersimpan.

# DAFTAR PUSTAKA


Balogh, Richard. 2007. *Practical Kinematics of the Differential Driven Mobile Robot*. Bratislava: Universitas Slovak

Braunl, Thomas. 2006. *Embedded Robotics Second Edition*. Jerman: Springer

*Differential Wheeled Robot*. http://en.wikipedia.org, diakses tanggal 4 Januari 2009

DIKTI. 2008. *Panduan Kontes Robot Cerdas Indonesia 2009*. Jakarta: DIKTI

Everlight. 2004. *Technical Data Sheet 3.0mm Round Type LED Lamps 204HD*. Everlight Electronics Co.,Ltd.

Harashima, Furnio. 1999. *Kinematic Correction of Differential Drive Mobile Robot and Design for Velocity Trajectory with Acceleration Constraints on Motor Controller*. Proceeding of the 1999 IEEE/RSJ, 930-935.

National Semiconductor. 1998. *CD40106BM/CD40106BC Hex Schmitt Trigger*. National Semiconductor Corporation

ON Semiconductor. 2004. *BC546B, BC547A, B, C, BC548B, C*. Semiconductor Componens Industries.

Renesas. 2005. *R8C/13 Hardware Manual Rev 1.10*. Renesas Technology

*Rotary Encoder*. http://en.wikipedia.org, diakses tanggal 2 Februari 2009

SanDisk. 2003. *SanDisk Secure Digital Card Product Manual*. SanDisk Corporation

*Serial Peripheral Interface Bus*. http://en.wikipedia.org, diakses tanggal 2 Februari 2009

Shenzhen. 2005. *LMB162ABC LCD Module User Manual*. Shenzhen Topway Technology Co., Ltd.

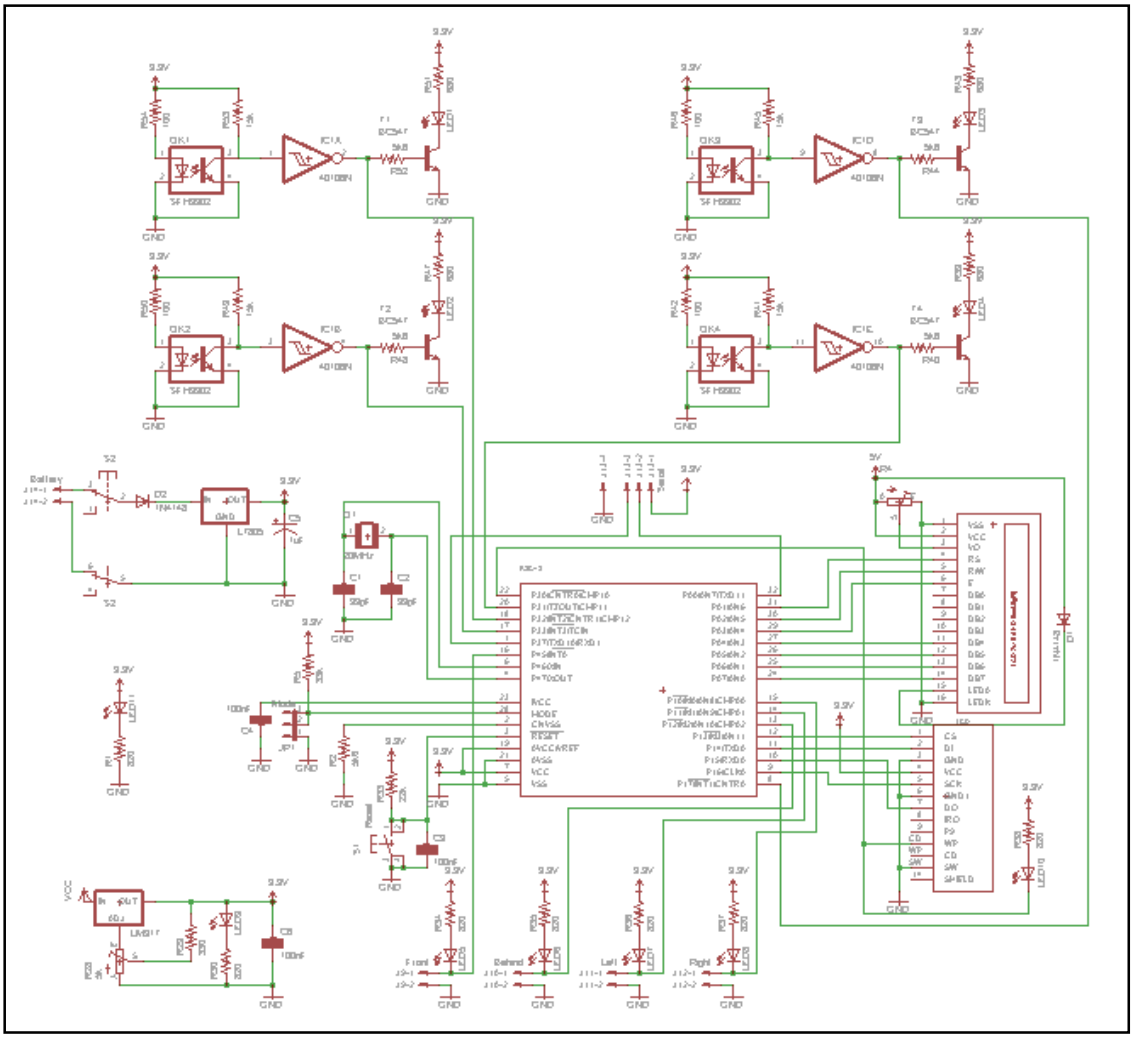
Sigit, Riyanto. 2007. *Robotika, Sensor, dan Aktuator*. Yogyakarta: Graha Ilmu

Transcend. 2009. *TS1G-2GSDG Secure Digital Card*. Transcend Information Inc.

# LAMPIRAN

# LAMPIRAN I

## Rangkaian Alat



Gambar rangkaian alat

# LAMPIRAN II

## Listing Program

# LAMPIRAN II-1

## Listing Program Utama

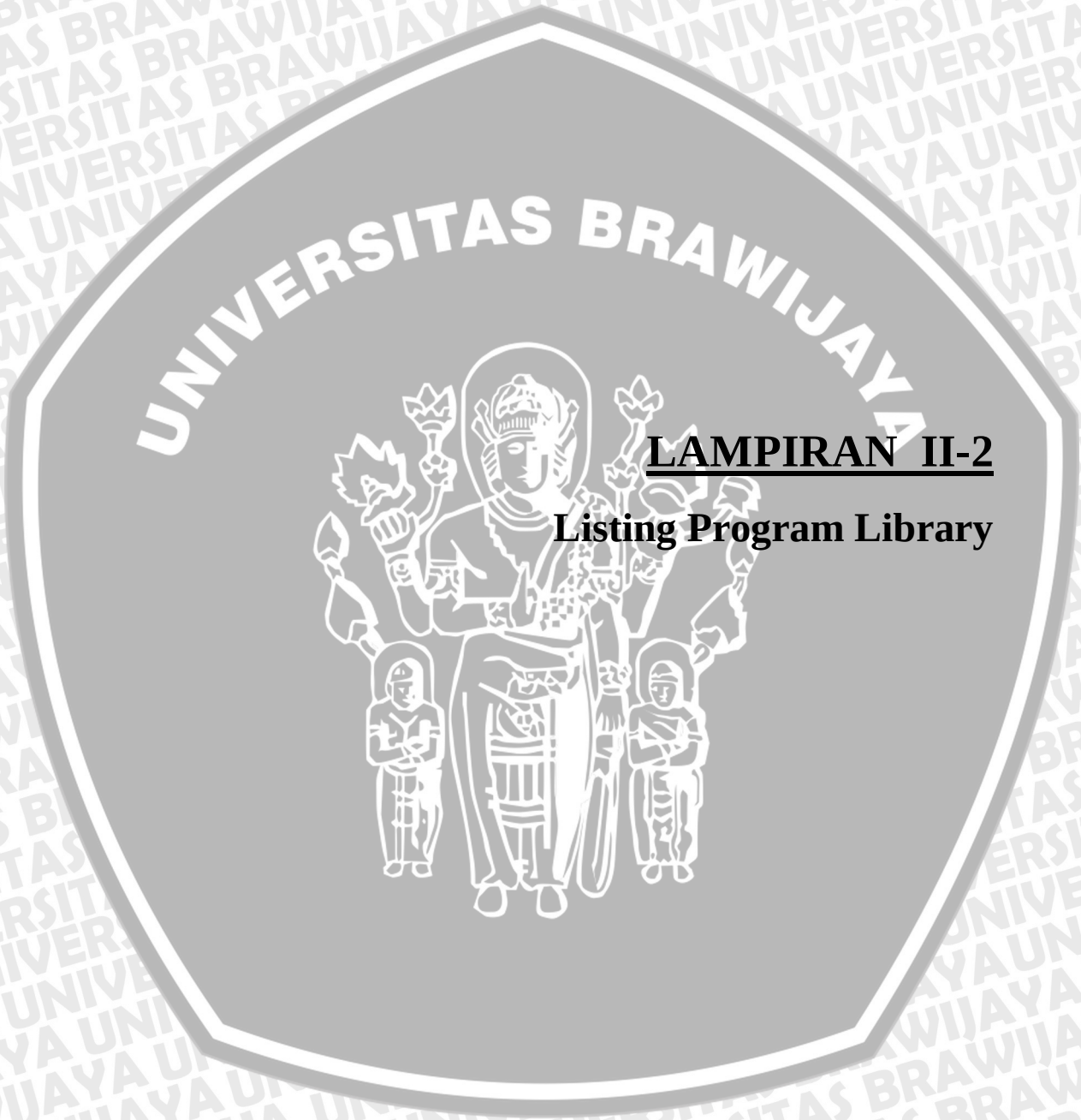
**Program Keseluruhan**

```
#include"sfr_r813.h"
#include"lcd.h"
#include"peta.h"
#include"sdc.h"
#include"fat.h"
#include"float.h"
#include"math.h"

long pulsa_ka=0,pulsa_ki=0;
char arah_ka=0,arah_ki=0;
struct koordinat{
        float x;
        float y;
        float sudut;
};
struct koordinat lokal={0,0,0};
struct koordinat global={0,0,0};
struct posisi{
        int x;
        int y;
};
struct posisi blok={0,0};
struct posisi flag={0,0};
struct kondisi{
        char utara;
        char barat;
        char selatan;
        char timur;
};
struct kondisi dinding={0,0,0,0};
unsigned int sudut=0;

int koreksi=0,depan=0;
unsigned int arah=0;
char sdc_insert=0;
char status=0;

unsigned char sector[512];
unsigned int fbr_sector=0;
unsigned int fat_sector=0;
unsigned int root_sector=0;
unsigned char sector_cluster=0;
unsigned int sector_fat=0;
unsigned long cluster=0;

#pragma INTERRUPT 25 dir_right
void dir_right(void)
{
        //cek arah roda kanan
        int1ic=0;
        if(p3_1==0)arah_ka=1;           //maju
        else arah_ka=0;                 //mundur
}
#pragma INTERRUPT 21 dir_left
void dir_left(void)
{
        //cek arah roda kiri
        int2ic=0;
        if(p3_3==0)arah_ki=1;           //maju
        else arah_ki=0;                 //mundur
}
void init_clock(void)
{
        asm("FCLR I");          // interrupt dimatikan
        prcr = 1;               // proteksi clock control dimatikan
        cm13 = 1;               // XIN-XOUT pin
        cm14 = 0;               // low speed oscillator on
        cm15 = 1;
        cm05 = 0;               // main clock : start oscillation
        cm16 = 0;               // main clock = No division mode
        cm17 = 0;
        cm06 = 0;               // main clock division enabled
```

```
        asm("nop");             // mait for stability
        asm("nop");
        asm("nop");
        asm("nop");
        ocd2 = 0;               // select crystal oscilator
        prcr = 0;               // proteksi clock control diaktifkan
        asm("FSET I");
}
void init_timer(void)
{
        //timer x sebagai counter roda kanan
        txmod1 = 1;
        txmod0 = 0;
        r0edg = 1;
        tx = prex = 255;
        txs = 1;

        //timer y sebagai counter roda kiri
        tymod0 = 0;
        r1edg = 1;
        tyck1 = 1;
        tyck0 = 1;
        typr = prey = 255;
        tys = 1;

        //  timer z
        //f = 1/8 fRING = 2,5 MHz
        //tPROSES = 0,0262 s
        tzmod1 = 0;
        tzmod0 = 0;
        tzck1 = 0;          tzck0 = 1;
        prez = 255;
        tzpr = 255;
        tzs = 0;
}
void init_interrupt(void)
{
        int1ic = 2;
        int2ic = 3;
}
void init_io(void)
{
        /*IO R8C/13:
                p4_5: wall - front
                p1_2: wall - back
                p1_0: wall - right
                p1_1: wall - left

                p1_7(cntr0): right rotary - counter
                p3_1: right rotary - direction

                p3_2(cntr1): left rotary - counter
                p3_3: left rotary - direction

                p1_4(txd0): sdcard - mosi
                p1_5(rxd0): sdcard - miso
                p1_3: sdcard - ss
                p1_6: sdcard - sck
                p3_0: sdcard - detect

                p0_1: lcd - rs
                p0_2: lcd - r/w
                p0_3: lcd - enable
                p0_4-p0_7: lcd - data
        */

        prc2 = 1;
        pd0 = 0xff;
        pd1 = 0b01011000;
        pd3 = 0;
        pd4 = 0;

        pur1 = 0xff;
        pur0 = 0xff;
        prc2 = 0;
}
```

```
void main(void)
{
	//inisialisasi
	init_clock();
	init_timer();
	init_interrupt();
	init_io();
	lcd_init();
	spi_init();

	lcd_clear();
	lcd_gotoxy(0,0);
	lcd_puts("Sistem Pemetaan");
	lcd_gotoxy(1,0);
	lcd_puts("Model Robot KRCI");
	delay_us(50000);

	if(SDC==0)
	{
		while(sdc_init()!=SUCCESS);
		clk1_u0c0 = 0;
		clk0_u0c0 = 0;
		u0brg = 0;		// Freq SCK = 20 MHz/2(0+1) = 10 MHz
		fbr_sector=get_address_fbr();
		status=get_info_fbr();
		cluster=(unsigned long)cek_file();
		if(cluster==0 && status==1)
		{	make_peta();
			lcd_clear();
			lcd_gotoxy(1,0);
			lcd_puts("build file...");
			delay_us(30000);
		}
		else if(cluster!=0 && status==1)
		{	lcd_clear();
			lcd_gotoxy(1,0);
			lcd_puts("file exist...");
			delay_us(30000);
		}
		else if(status==0)
		{
			lcd_clear();
			lcd_gotoxy(1,0);
			lcd_puts("FAT error...");
			delay_us(30000);
		}
	}
	else
	{
		lcd_clear();
		lcd_gotoxy(1,0);
		lcd_puts("SD Card missing");
		delay_us(30000);
	}

	while(1)
	{
		ambil_data();
		hitung_lokal();
		hitung_global();
		flag.x = cek_posisi(&global.x,&blok.x);
		flag.y = cek_posisi(&global.y,&blok.y);
		cek_dinding();
		koreksi_data();
		if(status==1) store_peta();

		lcd_clear();
		lcd_gotoxy(0,0);
		lcd_puts("Blok:X=");
		lcd_number(blok.x);
		lcd_puts("  Y=");
		lcd_number(blok.y);
		lcd_gotoxy(1,0);
		lcd_puts("Arah=");
		switch(arah)
		{	case UTARA :	lcd_puts("utara");break;
```

```
                case BARAT :    lcd_puts("barat");break;
                case SELATAN:   lcd_puts("selatan");break;
                case TIMUR :    lcd_puts("timur");break;
        }
        lcd_gotoxy(1,13);
        lcd_number(sudut);

        int1ic = 2;
        int2ic = 3;
};
}
```




# LAMPIRAN II-2

## Listing Program Library

**lcd.h**

```
#ifndef LCD_H
#define LCD_H

#define RS     p0_1
#define RW    p0_2
#define EN     p0_3
#define D4     p0_4
#define D5     p0_5
#define D6     p0_6
#define D7     p0_7

#define d_RS pd0_1
#define d_RW     pd0_2
#define d_EN pd0_3
#define d_D4 pd0_4
#define d_D5 pd0_5
#define d_D6 pd0_6
#define d_D7 pd0_7

void delay(unsigned long tunggu);
void lcd_clear(void);
void lcd_data(char c,char dat);
void lcd_gotoxy(unsigned char y,unsigned char x);
void lcd_puts(char* dat);
void lcd_number(long x);
void lcd_init(void);

#endif
```

**lcd.c**

```
#include "sfr_r813.h"
#include "lcd.h"

void delay(unsigned long tunggu)
{
    while(tunggu--);
}
void lcd_clear(void)
{
    lcd_data(0,0x01);
    lcd_data(0,0x02);
    delay(1000);
}
void lcd_data(char c,char dat)
{
    RW = 0;   RS = c;
    delay(10);
    EN = 1; delay(10);
    if ((dat & 0x80)==0x80) D7=1; else D7=0;
    if ((dat & 0x40)==0x40) D6=1; else D6=0;
    if ((dat & 0x20)==0x20) D5=1; else D5=0;
    if ((dat & 0x10)==0x10) D4=1; else D4=0;
    EN = 0; delay(10);
    EN = 1; delay(10);
    if ((dat & 0x08)==0x08) D7=1; else D7=0;
    if ((dat & 0x04)==0x04) D6=1; else D6=0;
    if ((dat & 0x02)==0x02) D5=1; else D5=0;
    if ((dat & 0x01)==0x01) D4=1; else D4=0;
    EN = 0; delay(10);
}
void lcd_gotoxy(unsigned char y,unsigned char x)
{
    int a;
    switch(y)
    {
        case 0: a = 0x80; break;
        case 1:a = 0xC0; break;
    }
    a+=x;  lcd_data(0,a);
}
void lcd_puts(char* dat)
{
```

```
    char i = 0;
    while( dat[i] != 0 )
    {
        lcd_data(1, dat[i]); i++;
    }
}
void lcd_number(long x)
{
    int bil1, bil2, bil3, bil4, bil5;
    if(x < 0)
    {
        lcd_data(1, 0b00101101);
        x = -x;
    }

    bil1 =(int) x/100000l;    // nilai ratus ribuan
    x = x - 100000*bil1;
    bil2 =(int) x/10000l;     // nilai puluh ribuan
    x = x - 10000*bil2;
    bil3 =(int) x/1000l;      // nilai ribuan
    x = x-1000*bil3;
    bil4 =(int) x/100l;       // nilai ratusan
    x = x - 100*bil4;
    bil5 =(int) x/10l;        // nilai puluhan
    x = x - 10*bil5;          // nilai satuan

    // tampilkan nilai pada lcd
    if(bil1 != 0) {lcd_data(1, bil1 + 48);}
    if(bil1 != 0 | bil2 != 0) {lcd_data(1, bil2 + 48);}
    if(bil1 != 0 | bil2 != 0 | bil3 != 0) {lcd_data(1, bil3 + 48);}
    if(bil1 != 0 | bil2 != 0 | bil3 != 0 | bil4 != 0) {lcd_data(1, bil4 + 48);}
    if(bil1 != 0 | bil2 != 0 | bil3 != 0 | bil4 != 0 | bil5 != 0) {lcd_data(1, bil5 + 48);}
    lcd_data(1,(int) x + 48);
}
void lcd_init(void)
{
    prc2 = 1;
    d_RS = 1;
    d_RW = 1;
    d_EN = 1;
    d_D4 = 1;
    d_D5 = 1;
    d_D6 = 1;
    d_D7 = 1;
    prc2 = 0;

    delay(5000);
    lcd_data(0,0x33);
    lcd_data(0,0x32);
    lcd_data(0,0x2B);   //function set: 4 bit, 2 line, 5x8 pixel
    lcd_data(0,0x0C);   //display on off: display on, cursor off, blink off
    lcd_data(0,0x01);
    lcd_data(0,0x06);    //entry mode set: increment, disable shift
    delay(5000);
}
```

**sdc.h**

```
#ifndef SDC_H
#define SDC_H

#define     DIN       p1_4
#define     DOUT      p1_5
#define     CLK       p1_6
#define     CS        p1_3
#define     SDC       p3_0

#define     INPUT       0
#define     OUTPUT      1
#define     HIGH        1
#define     LOW         0

#define     d_DIN       pd1_4
#define     d_DOUT      pd1_5
#define     d_CLK       pd1_6
#define     d_CS        pd1_3
```

```
#define         d_SDC         pd3_0

// commands: first bit 0 (start bit), second 1 (transmission bit); CMD-number +
0ffsett 0x40
#define CMD0    0x40   //GO_IDLE_STATE
#define CMD1    0x41   //SEND_OP_CODE
#define CMD17   0x51   //READ_SINGLE_BLOCK
#define CMD24   0x58   //WRITE_SINGLE_BLOCK

// error/success codes
#define SUCCESS          0x00
#define RESP_ERROR       0x01
#define INIT_ERROR       0x02
#define READ_ERROR       0x03
#define WRITE_ERROR      0x04
#define CRC_ERROR        0x05
#define BUSY             0x06

void delay_us(unsigned long d);
void spi_init(void);
void uart_trans(unsigned char data);
unsigned char uart_rec(void);
void enable_rec(void);
void enable_trans(void);
void uart_end(void);
void send_cmd(unsigned char cmd, unsigned long addr, unsigned char crc);
char sdc_init(void);
char get_response(void);
char get_start_token(void);
char get_dataresp(void);
char read_block(unsigned long addr);
char write_block(unsigned long addr);

#endif
```

**sdc.c**

```
#include "sfr_r813.h"
#include "sdc.h"

extern unsigned char response;
extern unsigned char sector[512];
extern unsigned long fbr_sector;

void delay_us(unsigned long d)
{
    unsigned long n;
    n=d*5;
    while(n--){};
}
void spi_init(void)
{
    DIN    = HIGH;
    CLK    = HIGH;
    CS     = HIGH;

    d_DIN     = OUTPUT;
    d_DOUT    = INPUT;
    d_CLK     = OUTPUT;
    d_CS      = OUTPUT;
    d_SDC     = INPUT;

    u0mr   = 0x01;
    u0c0   = 0x8a;     // f32SIO
    u0brg  = 1;        // 20 MHz/32/(2*(1+1)) = 156250 Hz
    u0irs  = 1;        // interrupt saat transmission completed
}
void uart_trans(unsigned char data)
{   while( !ti_u0c1 );
    u0tb = data;
    while( !ir_s0tic );
    ir_s0tic = 0;
}
```

```
unsigned char uart_rec(void)
{   uart_trans(0xff);
    while ( !ir_s0ric );
    ir_s0ric = 0;
    return( (unsigned char) u0rb);
}
void enable_rec(void)
{   while( !ti_u0c1 );
    while( !txept_u0c0 );
    re_u0c1 = 1;
}
void enable_trans(void)
{
    te_u0c1 = 1;
}
void uart_end(void)
{   while( !ti_u0c1 );
    while( !txept_u0c0 );
    re_u0c1 = 0;
    te_u0c1 = 0;
}
void send_cmd(unsigned char cmd, unsigned long addr, unsigned char crc)
{
    unsigned char frame[6];
    unsigned char temp;
    signed char x;

    frame[0] = cmd;

    for(x=3; x>=0; x--)
    {   temp = (char) (addr>>(8*x));
        frame[4-x] = temp;
    }

    frame[5] = crc;

    for(x=0; x<6; x++) uart_trans(frame[x]);
}
```

```
char get_response(void)
{
    unsigned char i=0;
    unsigned char response;

    enable_rec();
    while(i<=8)          // dibatasi selama NCR
    {   response = uart_rec();
        if(response == 0x00) break;
        if(response == 0x01) break;
        i++;
    }
    return response;
}
char get_start_token()
{
    unsigned int i=0;
    unsigned char response;
    enable_rec();
    while(i<=25000)          // dibatasi selama NCR
    {   response = uart_rec();
        if(response == 0xfe) break;
        i++;
    }
    return response;
}
char get_dataresp(void)
{
    char i=0;
    char response,rvalue;

    enable_trans();
    enable_rec();

    while(i<=8)                    // tunggu busy
    {
        response = uart_rec();     // 0x00 = busy
        response &= 0x1f;
```

```
        switch(response)
        { case 0x05: rvalue = SUCCESS; break;
          case 0x0b: return (CRC_ERROR);
          case 0x0d: return (WRITE_ERROR);
        }
        i++;
    }
    do
    {
        response = uart_rec();
        rvalue=BUSY;
    }while(response==0);
    return rvalue;
}
char sdc_init(void)
{
    unsigned char response;
    char rvalue;
    int n;

    delay_us(10000);
    DIN = HIGH;
    CS = HIGH;

    enable_trans();
    for(n = 11;n>0;n--)uart_trans(0xff);        //>74 clock
    CS = LOW;
    send_cmd(CMD0,0,0x95);
    enable_rec();
    response = get_response();
    n=0;
    do
    {
        CS = HIGH;
        uart_trans(0xff);
        CS  = LOW;
        send_cmd(CMD1,0,0xff);
        response = get_response();
        n++;
        if(response == 0x00) rvalue = SUCCESS;
        else rvalue = INIT_ERROR;
    }
    while(response == 0x01 && n<=30);
    CS = HIGH;
    uart_trans(0xff);
    uart_end();
    return rvalue;
}
char read_block(unsigned long addr)          // addr: alamat dlm sector
{
    unsigned int n;
    char rvalue;

    n = addr+fbr_sector;
    addr = (unsigned long)n*512UL;

    DIN = HIGH;
    CS = LOW;

    enable_trans();
    uart_trans(0xff);

    send_cmd(CMD17,addr,0xff);
    enable_rec();

    if(get_response()==0x00)
    {
        if(get_start_token()==0xfe)
        {
            for(n=0; n<512; n++) sector[n] = uart_rec();
            uart_rec();
            uart_rec();
            rvalue = SUCCESS;
        }
        else rvalue = READ_ERROR;
    }
```

```
    else rvalue = RESP_ERROR;

    CS = HIGH;
    uart_trans(0xff);
    uart_end();
    return rvalue;
}
char write_block(unsigned long addr)          // addr dlm sector, bkn byte
{
    unsigned int n=0;
    char rvalue;
    char dresp;

    n = addr+fbr_sector;
    addr = (unsigned long)n*512UL;

    CS = LOW;
    enable_trans();
    send_cmd(CMD24, addr, 0xff);

    enable_rec();
    if(get_response() == 0x00)       // R1 0x00 jika tidak ada error
    {   uart_trans(0xff);       // tunggu NWR
        uart_trans(0xfe);       // start block token

        for(n=0; n<512; n++) uart_trans( sector[n] );    // kirim 512 byte

        uart_trans(0xff);       // 2 byte CRC
        uart_trans(0xff);

        rvalue = get_dataresp();
    }
    else rvalue = RESP_ERROR;
    CS = HIGH;
    uart_rec();
    uart_trans(0xff);  // kasi 8 clock, untuk MMC selesaikan proses internal
    return rvalue;
}
```

**fat.h**

```
#ifndef MMC_FAT_H
#define MMC_FAT_H

unsigned long get_address_fbr(void);
char get_info_fbr(void);
unsigned long cek_file(void);
unsigned long cek_fat(void);
void make_peta(void);
void store_peta(void);

#endif
```

**fat.c**

```
#include "sfr_r813.h"
#include "lcd.h"
#include "sdc.h"
#include "fat.h"
#include "peta.h"

extern unsigned char sector[512];
extern unsigned int fbr_sector;
extern unsigned int fat_sector;
extern unsigned int root_sector;
extern unsigned char sector_cluster;
extern unsigned int sector_fat;
extern unsigned long cluster;
extern long pulsa_ka,pulsa_ki;
extern char arah_ka,arah_ki;
extern struct koordinat{
    float x;
    float y;
    float sudut;
};
extern struct koordinat lokal;
```

```
extern struct koordinat global;
extern struct posisi{
    int x;
    int y;
};
extern struct posisi blok;
extern struct posisi flag;
extern struct kondisi{
    char utara;
    char barat;
    char selatan;
    char timur;
};
extern struct kondisi dinding;
extern unsigned int sudut,arah;
extern int koreksi,depan;

unsigned long get_address_fbr(void)
{
    unsigned long rvalue;

    fbr_sector=0;
    read_block(0);

    rvalue = (unsigned long) sector[454];
    rvalue |= (unsigned long) sector[455]<<8;
    rvalue |= (unsigned long) sector[456]<<16;
    rvalue |= (unsigned long) sector[457]<<32;

    return rvalue;
}
char get_info_fbr(void)
{
    read_block(0);
    sector_cluster = (unsigned char)sector[13];
    fat_sector = (unsigned int)sector[15]<<8;
    fat_sector |= (unsigned int)sector[14];
    sector_fat = (unsigned int)sector[23]<<8;
    sector_fat |= (unsigned int)sector[22];
    root_sector = (unsigned int)(2*sector_fat)+fat_sector;

    //checking FAT16 or FAT32
    if(sector[22]==0) return 0;         //FAT32
    else return 1;                      //FAT16
}
unsigned long cek_file(void)
{
    unsigned int n=0,m=0;
    unsigned long nomer=0;
    while(m<32 && nomer==0)
    {
        read_block(root_sector+m);
        n=0;
        while(n<16 && nomer==0)
        {
            if(sector[n*32]!=0xE5 && sector[n*32]!=0)
            {
if(sector[n*32]=='P' && sector[n*32+1]=='E' && sector[n*32+2]=='T' &&
sector[n*32+3]=='A')
                {
if(sector[n*32+4]==' ' && sector[n*32+5]==' ' && sector[n*32+6]==' ' &&
sector[n*32+7]==' ')
                    {
                        nomer = sector[n*32+27]*256;
                        nomer += sector[n*32+26];
                    }
                }
            }
            n++;
        }
        m++;
    }
    return nomer;
}
```

```
unsigned long cek_fat(void)
{
    unsigned long nomer=0;
    unsigned int m=0,n=0;
    while(m<sector_fat && nomer==0)
    {
        read_block(fat_sector+m);
        n=0;
        while(n<256 && nomer==0)
        {
            if(sector[n*2]==0)
            {
                nomer=m*256+n;
                if(sector[(n+1)*2]!=0 && sector_cluster<=4) nomer=0;
                if(sector[(n+2)*2]!=0 && sector_cluster<=2) nomer=0;
            }
            n++;
        }
        m++;
    }
    return nomer;
}
void make_peta(void)
{
    unsigned char a=0,c=0,d=0;
    unsigned int b=0;
    unsigned int n=0,m=0;
    unsigned long add;

    // check fat tabel for empty cluster
    cluster=(unsigned long)cek_fat();

    // write root directory for new directory entry
    a=0;m=0;
    while((m<32)&&(a==0))
    {
        read_block(root_sector+m);
        n=0;
        while((n<16)&&(a==0))
        {
            if(sector[n*32]==0xE5 || sector[n*32]==0)
            {
                sector[n*32] = 'P';
                sector[n*32+1] = 'E';
                sector[n*32+2] = 'T';
                sector[n*32+3] = 'A';
                for(b=4; b<8; b++)sector[n*32+b] = ' ';
                sector[n*32+8] = 'T';
                sector[n*32+9] = 'X';
                sector[n*32+10] = 'T';
                sector[n*32+27] = cluster/256;
                sector[n*32+26] = cluster-(sector[n*32+27]*256);
                sector[n*32+28] = 0x9f;                    // 3KB
                sector[n*32+29] = 0x0a;
                sector[n*32+30] = sector[n*32+31] = 0;
                a=1;
            }
            n++;
        }
        if(a!=0) write_block(root_sector+m);
        m++;
    }

    // write fat table for file cluster
    a=cluster/256;
    read_block(fat_sector+a);
    if(sector_cluster==2)
    {
        sector[(cluster-a*256)*2] = cluster;
        sector[(cluster+1-a*256)*2] = cluster+1;
        sector[(cluster+2-a*256)*2] = sector[(cluster+2-a*256)*2+1] = 0xff;
    }
    if(sector_cluster==4)
    {
        sector[(cluster-a*256)*2] = cluster;
```

```
    sector[(cluster+1-a*256)*2] = sector[(cluster+1-a*256)*2+1] = 0xff;

}
if(sector_cluster>=8)
{
    sector[(cluster-a*256)*2] = sector[(cluster-a*256)*2+1] = 0xff;
}
write_block(fat_sector+a);                  // edit FAT1
write_block(fat_sector+a+sector_fat);       // edit FAT2

// make data file: PETA.TXT
for(n=0;n<=35;n++)
{
    if(n==0)
    {
        for(m=0;m<512;m++)sector[m]=0x20;
        for(m=0;m<62;m++)sector[m]=0x3d;
        sector[62]=0x0d;sector[63]=0x0a;
        for(d=0;d<3;d++)
        {
            sector[d*64+64]=0x7c;
            sector[d*64+64+3]=0x7c;
            sector[d*64+64+6]=0x7c;
            for(m=0;m<4;m++)
            {
                sector[d*64+64+9+m*13]=0x7c;
                if(d==2)
                {
                    sector[d*64+64+12+m*13]=0x7c;
                    sector[d*64+64+14+m*13]=0x7c;
                    sector[d*64+64+16+m*13]=0x7c;
                    sector[d*64+64+18+m*13]=0x7c;
                    sector[d*64+64+20+m*13]=0x7c;
                }
            }
            sector[d*64+64+61]=0x7c;
            sector[d*64+64+62]=0x0d;sector[d*64+64+63]=0x0a;
        }

        sector[74]='U';sector[75]='t';sector[76]='a';sector[77]='r';sector[78]='a';
        sector[87]='T';sector[88]='i';sector[89]='m';sector[90]='u';sector[91]='r';
        sector[100]='S';sector[101]='e';sector[102]='l';sector[103]='a';
        sector[104]='t';sector[105]='a';sector[106]='n';

sector[113]='B';sector[114]='a';sector[115]='r';sector[116]='a';sector[117]='t';
sector[65]='N';sector[66]='o';sector[69]='X';sector[72]='Y';

        for(d=0;d<4;d++)
        {
            for(m=0;m<12;m++)
            {
                sector[138+m+(d*13)]='-';
            }
        }
        for(m=0;m<4;m++)
        {
            sector[202+(m*13)]='N';
            sector[202+(m*13)+1]='o';
            sector[202+(m*13)+3]='W';
            sector[202+(m*13)+5]='D';
            sector[202+(m*13)+7]='F';
            sector[202+(m*13)+9]='C';
            sector[202+(m*13)+11]='B';
        }
        for(m=0;m<62;m++)sector[256+m]=0x3d;
        sector[256+62]=0x0d;sector[256+63]=0x0a;
    }
    else if(n==3 || n==11 || n==19 || n==27 || n==35)
    {
        for(m=0;m<512;m++)sector[m]=0x20;
    }

    if(n<=2){a=0;b=320;c=0;}
    else if(n>2 && n<=10){a=3;b=0;c=1;}
    else if(n>10 && n<=18){a=11;b=0;c=2;}
    else if(n>18 && n<=26){a=19;b=0;c=3;}
    else if(n>26 && n<=34){a=27;b=0;c=4;}
```

```
        else if(n==35){a=35;b=0;c=5;}

        sector[(n-a)*64+b+0]=0x7c;
        sector[(n-a)*64+b+3]=0x7c;
        sector[(n-a)*64+b+6]=0x7c;
        for(d=0;d<4;d++)
        {
            sector[(n-a)*64+b+9+d*13]=0x7c;
            sector[(n-a)*64+b+12+d*13]=0x7c;
            sector[(n-a)*64+b+14+d*13]=0x7c;
            sector[(n-a)*64+b+16+d*13]=0x7c;
            sector[(n-a)*64+b+18+d*13]=0x7c;
            sector[(n-a)*64+b+20+d*13]=0x7c;
        }
        sector[(n-a)*64+b+61]=0x7c;
        sector[(n-a)*64+b+62]=0x0d;sector[(n-a)*64+b+63]=0x0a;

        if(n==2 || n==10 || n==18 || n==26 || n==34 || n==35)
        {
            if(n==35)
            {
                for(m=0;m<62;m++)sector[64+m]=0x3d;
                sector[64+62]=0x0d;sector[64+63]=0x0a;
                sector[128]='P';sector[129]='o';sector[130]='s';sector[131]='i';
                sector[132]='s';sector[133]='i';sector[134]=':';
                sector[135]='X';sector[136]='=';sector[139]=';';
                sector[142]='Y';sector[143]='=';sector[146]=';';
                sector[149]='A';sector[150]='r';
                sector[151]='a';sector[152]='h';sector[153]='=';
            }
            write_block(root_sector+32+c+(cluster-2)*32);
        }
    }
}
void store_peta(void)
{
    unsigned char a=0,c=0,i=0;
    unsigned int b=0;

    char n=0,m=0,nomer=0;
    int buff_x,buff_y;
    unsigned char temp[4][2];

    // check root entry for cluster of file: PETA.TXT
    cluster=0;
    cluster=(unsigned long)cek_file();

    // read database
    n=0;i=0;nomer=40;
    for(a=0;a<4;a++)
    {
        for(b=0;b<2;b++)temp[a][b]=0x20;
    }
    while(n<=35)
    {
        if(n<=2){a=0;b=320;c=0;}
        else if(n>2 && n<=10){a=3;b=0;c=1;}
        else if(n>10 && n<=18){a=11;b=0;c=2;}
        else if(n>18 && n<=26){a=19;b=0;c=3;}
        else if(n>26 && n<=34){a=27;b=0;c=4;}
        else if(n==35){a=35;b=0;c=5;}

        if(n==0 || n==3 || n==11 || n==19 || n==27 || n==35)
            read_block(root_sector+32+c+(cluster-2)*32);

        // cek nilai x yang sudah tersimpan
        //buff_x = 0; buff_y = 0;
        if(sector[(n-a)*64+b+4]==0x2d)
        {
            buff_x = (int)sector[(n-a)*64+b+5]-0x30;
            buff_x = -(buff_x);
        }
        else if(sector[(n-a)*64+b+4]!=0x20)
        {
            buff_x = (int)(sector[(n-a)*64+b+4]-0x30)*10;
            buff_x += (int)sector[(n-a)*64+b+5]-0x30;
        }
```

```
// cek nilai y yang sudah tersimpan
if(sector[(n-a)*64+b+7]==0x2d)
{
    buff_y = (int)sector[(n-a)*64+b+8]-0x30;
    buff_y = -(buff_y);
}
else if(sector[(n-a)*64+b+7]!=0x20)
{
    buff_y = (int)(sector[(n-a)*64+b+7]-0x30)*10;
    buff_y += (int)sector[(n-a)*64+b+8]-0x30;
}

// periksa nomer blok
if(sector[(n-a)*64+b+4]!=0x20 && sector[(n-a)*64+b+7]!=0x20)
{
    if((buff_x==blok.x) && (buff_y==blok.y + 1))
    {
        temp[0][0] = sector[(n-a)*64+b+1];      // nomer blok UTARA
        temp[0][1] = sector[(n-a)*64+b+2];
    }
    else if((buff_x==blok.x+1)&&(buff_y==blok.y))
    {
        temp[1][0] = sector[(n-a)*64+b+1];      // nomer blok TIMUR
        temp[1][1] = sector[(n-a)*64+b+2];
    }
    else if((buff_x==blok.x)&&(buff_y==blok.y-1))
    {
        temp[2][0] = sector[(n-a)*64+b+1];      //nomer blok SELATAN
        temp[2][1] = sector[(n-a)*64+b+2];
    }
    else if((buff_x==blok.x-1)&&(buff_y==blok.y))
    {
        temp[3][0] = sector[(n-a)*64+b+1];      // nomer blok BARAT
        temp[3][1] = sector[(n-a)*64+b+2];
    }
}
if((sector[(n-a)*64+b+4]==0x20) && (sector[(n-a)*64+b+7]==0x20))
    {
        if(nomer==40)nomer = n;
    }
    else if((buff_x==blok.x) && (buff_y==blok.y)){nomer = n;}

    n++;
}

// store data: no,x,y,data blok
n = nomer;
if(n<=2){a=0;b=320;c=0;}
else if(n>2 && n<=10){a=3;b=0;c=1;}
else if(n>10 && n<=18){a=11;b=0;c=2;}
else if(n>18 && n<=26){a=19;b=0;c=3;}
else if(n>26 && n<=34){a=27;b=0;c=4;}
else if(n==35){a=35;b=0;c=5;}

read_block(root_sector+32+c+(cluster-2)*32);
sector[(n-a)*64+b+1] = (nomer/10);
sector[(n-a)*64+b+2] = nomer-(sector[(n-a)*64+b+1]*10)+0x30;
sector[(n-a)*64+b+1] += 0x30;

if(blok.x<0)
{
    sector[(n-a)*64+b+4]=0x2d;
    sector[(n-a)*64+b+5]=(unsigned char)-blok.x+0x30;
}
else
{
    sector[(n-a)*64+b+4]=(unsigned char)blok.x/10;
sector[(n-a)*64+b+5]=(unsigned char)blok.x-(sector[(n-a)*64+b+4]*10)+0x30;
    sector[(n-a)*64+b+4]+=0x30;
}
if(blok.y<0)
{
    sector[(n-a)*64+b+7]=0x2d;
    sector[(n-a)*64+b+8]=(unsigned char)-blok.y+0x30;
}
```

```
    else
    {
        sector[(n-a)*64+b+7]=(unsigned char)blok.y/10;
sector[(n-a)*64+b+8]=(unsigned char)blok.y-(sector[(n-a)*64+b+7]*10)+0x30;
        sector[(n-a)*64+b+7]+=0x30;
    }

    sector[(n-a)*64+b+13] = dinding.utara;          // dinding UTARA
    sector[(n-a)*64+b+26] = dinding.timur;          // dinding TIMUR
    sector[(n-a)*64+b+39] = dinding.selatan;        // dinding SELATAN
    sector[(n-a)*64+b+52] = dinding.barat;          // dinding BARAT

    sector[(n-a)*64+b+10] = temp[0][0];             // blok tetangga UTARA
    sector[(n-a)*64+b+11] = temp[0][1];
    sector[(n-a)*64+b+23] = temp[1][0];             // blok tetangga TIMUR
    sector[(n-a)*64+b+24] = temp[1][1];
    sector[(n-a)*64+b+36] = temp[2][0];             // blok tetangga SELATAN
    sector[(n-a)*64+b+37] = temp[2][1];
    sector[(n-a)*64+b+49] = temp[3][0];             // blok tetangga BARAT
    sector[(n-a)*64+b+50] = temp[3][1];
    write_block(root_sector+32+c+(cluster-2)*32);

    // store actual position
    read_block(root_sector+37+((cluster-2)*32));
    if(blok.x<0)
    {
        sector[128+9]= 0x2d;
        sector[128+10]=(unsigned char)-blok.x+0x30;
    }
    else
    {
        sector[128+9]=(unsigned char)blok.x/10;
        sector[128+10]=(unsigned char)blok.x-(sector[128+9]*10)+0x30;
        sector[128+9]+=0x30;
    }
    if(blok.y<0)
    {
        sector[128+16]=0x2d;
        sector[128+17]=(unsigned char)-blok.y+0x30;
    }
    else
    {
        sector[128+16]=(unsigned char)blok.y/10;
        sector[128+17]=(unsigned char)blok.y-(sector[128+16]*10)+0x30;
        sector[128+16]+=0x30;
    }

    switch(arah)
    {   case UTARA :    sector[128+26]='U';
                        sector[128+27]='t';
                        sector[128+28]='a';
                        sector[128+29]='r';
                        sector[128+30]='a';break;
        case BARAT :    sector[128+26]='B';
                        sector[128+27]='a';
                        sector[128+28]='r';
                        sector[128+29]='a';
                        sector[128+30]='t';break;
        case SELATAN: sector[128+26]='S';
                        sector[128+27]='l';
                        sector[128+28]='t';
                        sector[128+29]='a';
                        sector[128+30]='n';break;
        case TIMUR :    sector[128+26]='T';
                        sector[128+27]='i';
                        sector[128+28]='m';
                        sector[128+29]='u';
                        sector[128+30]='r';break;
    }
    write_block(root_sector+37+((cluster-2)*32));
}
```

**peta.h**

```
#ifndef     PETA_H
#define     PETA_H

#define JARI2_KA          4.75
#define JARI2_KI          4.75
#define JARAK_RODA        26.5
#define BLOK              50
#define BATAS             20
#define PPR_KA            256
#define PPR_KI            256
#define UTARA             0
#define BARAT             90
#define SELATAN           180
#define TIMUR             270

#define D_DEPAN       p4_5
#define D_BLKG        p1_2
#define D_KANAN       p1_0
#define D_KIRI        p1_1
#define DINDING       0
#define KOSONG        1

void ambil_data(void);
void hitung_lokal(void);
void hitung_global(void);
void cari_posisi(void);
void cek_dinding(void);
int cek_posisi(float *k, int *p);
void koreksi_data(void);

#endif
```

**peta.c**

```
#include   "sfr_r813.h"
#include   "peta.h"
#include   "float.h"
#include   "math.h"

extern long pulsa_ka,pulsa_ki;
extern char arah_ka,arah_ki;
extern struct koordinat{
    float x;
    float y;
    float sudut;
};
extern struct koordinat lokal;
extern struct koordinat global;
extern struct posisi{
    int x;
    int y;
};
extern struct posisi blok;
extern struct posisi flag;
extern struct kondisi{
    char utara;
    char barat;
    char selatan;
    char timur;
};
extern struct kondisi dinding;
extern unsigned int sudut,arah;
extern int koreksi,depan;

void ambil_data(void)
{
    pulsa_ka=(long)((255-tx)*256+(255-prex));
    tx = prex = 255;
    pulsa_ki=(long)((255-typr)*256+(255-prey));
    typr = prey = 255;
    if(arah_ka == 0)pulsa_ka = -pulsa_ka;
    if(arah_ki == 0)pulsa_ki = -pulsa_ki;
}
```

```
void hitung_lokal(void)
{
    float jarak_ka,jarak_ki,jari2_pst;
    double temp;
    jarak_ka = (float)2*3.14*JARI2_KA*pulsa_ka/PPR_KA;  // satuan cm
    jarak_ki = (float)2*3.14*JARI2_KI*pulsa_ki/PPR_KI;      // satuan cm
    lokal.sudut = (float)(jarak_ka - jarak_ki)/JARAK_RODA;  // satuan radian
    while(lokal.sudut>6.2832 || lokal.sudut<0)
    {
        if(lokal.sudut>6.2832)lokal.sudut=(float)lokal.sudut-6.2832;
        else if(lokal.sudut<0)lokal.sudut=(float)lokal.sudut+6.2832;
    }

    if(jarak_ka!=jarak_ki)
    {
        jari2_pst = (float)(JARAK_RODA*(jarak_ka + jarak_ki))/(2*(jarak_ka -
jarak_ki));
        lokal.x = (float)jari2_pst*cos(lokal.sudut)-jari2_pst;
        lokal.y = (float)jari2_pst*sin(lokal.sudut);
    }
    else
    {
        jari2_pst = FLT_MAX;
        lokal.x = 0;
        lokal.y = (float)jarak_ka;
    }
}
void hitung_global(void)
{
    global.x += (float)lokal.x*cos(global.sudut) - lokal.y*sin(global.sudut);
    global.y += (float)lokal.x*sin(global.sudut) + lokal.y*cos(global.sudut);
    global.sudut += lokal.sudut;                        // satuan radian
    while(global.sudut>6.2832 || global.sudut<0)
    {
        if(global.sudut>6.2832)global.sudut=(float)global.sudut-6.2832;
        else if(global.sudut<0)global.sudut=(float)global.sudut+6.2832;
    }
    sudut = (unsigned int)57.3248*global.sudut;
    if(sudut<45 || sudut>135) arah=UTARA;
    else if(sudut>=45 && sudut<135) arah = BARAT;
    else if(sudut>=135 && sudut<225) arah = SELATAN;
    else if(sudut>=225 && sudut<315) arah = TIMUR;
}
int cek_posisi(float *koordinat, int *blok)
{
    int nilai1=0,nilai2=0;
    char f;
    if((*koordinat > BATAS) || (*koordinat < -BATAS))
    {
        if(*koordinat > 0)
        {
            nilai1 = (*koordinat + BATAS)/BLOK;
            nilai2 = ((*koordinat - BATAS)/BLOK)+1;
            if(nilai1 == nilai2)
            {
                *blok = nilai1;
                f = 1;
            }else f = 0;
        }
        else
        {
            nilai1 = (*koordinat - BATAS)/BLOK;
            nilai2 = ((*koordinat + BATAS)/BLOK)-1;
            if(nilai1 == nilai2)
            {
                *blok = nilai1;
                f = 1;
            }else f = 0;
        }
    }
    else
    {
        *blok = 0;
        f = 1;
    }
    return f;
```

```
}
void cek_dinding(void)
{
    unsigned char temp;
    struct kondisi buffer;
    // UTARA: arah robot pertama kali diletakkan
    // Robot direction: UTARA (default)
    arah=UTARA;
    if(D_DEPAN==DINDING) buffer.utara='W'; else buffer.utara='-';
    if(D_KANAN==DINDING) buffer.timur='W'; else buffer.timur='-';
    if(D_BLKG==DINDING) buffer.selatan='W'; else buffer.selatan='-';
    if(D_KIRI==DINDING) buffer.barat='W'; else buffer.barat='-';
    if(sudut>=45 && sudut<135)
    {   // Robot direction: BARAT
        arah = BARAT;
        temp = buffer.utara;
        buffer.utara = buffer.timur;
        buffer.timur = buffer.selatan;
        buffer.selatan = buffer.barat;
        buffer.barat = temp;
    }
    else if(sudut>=135 && sudut<225)
    {   // Robot direction: SELATAN
        arah = SELATAN;
        temp = buffer.utara;
        buffer.utara = buffer.selatan;
        buffer.selatan = temp;
        temp = buffer.timur;
        buffer.timur = buffer.barat;
        buffer.barat = temp;
    }
    else if(sudut>=225 && sudut<315)
    {   // Robot direction: TIMUR
        arah = TIMUR;
        temp = buffer.utara;
        buffer.utara = buffer.barat;
        buffer.barat = buffer.selatan;
        buffer.selatan = buffer.timur;
        buffer.timur = temp;
    }

    if(flag.x==1)
    {
        dinding.timur = buffer.timur;
        dinding.barat = buffer.barat;
    }
    if(flag.y==1)
    {
        dinding.utara = buffer.utara;
        dinding.selatan = buffer.selatan;
    }
    }
    void koreksi_data(void)
    {
    if(D_DEPAN==DINDING) koreksi=1;
    else koreksi=0;
    if(koreksi == 1 && depan != D_DEPAN)
    {
        global.x = blok.x*BLOK;
        global.y = blok.y*BLOK;
        switch(arah)
        {       case UTARA:         global.sudut = 0;
                                    sudut = 0;
                                    break;
                case BARAT:         global.sudut = 1.57;
                                    sudut = 90;
                                    break;
                case SELATAN:       global.sudut = 3.14;
                                    sudut = 180;
                                    break;
                case TIMUR:         global.sudut = 4.71;
                                    sudut = 270;
                                    break;
        }
    }
    depan = D_DEPAN;}
```

# LAMPIRAN II-3

## Listing Program Pengujian

**Program Uji LCD**

```
#include "sfr_r813.h"
#include "lcd.h"

void init_clock(void)
{
        asm("FCLR I");
        prcr = 1;
        cm13 = 1;
        cm14 = 0;
        cm15 = 1;
        cm05 = 0;
        cm16 = 0;
        cm17 = 0;
        cm06 = 0;
        asm("nop");
        asm("nop");
        asm("nop");
        asm("nop");
        ocd2 = 0;
        prcr = 0;
        asm("FSET I");
}
void init_io(void)
{
        prc2 = 1;
        pd0 = 0xff;
        pd1 = 0b01011000;
        pd3 = 0;
        pd4 = 0;
        pur1 = 0xff;
        pur0 = 0xff;
        prc2 = 0;
}
void main(void)
{
        init_clock();
        init_io();
        lcd_init();

        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_puts("Program:");
        lcd_gotoxy(1,0);
        lcd_puts("Uji LCD....");
}
```

**Program Uji Kecepatan Respon Sensor Rotari**

```
#include "sfr_r813.h"

#define  T1      p1_0
#define  T2      p1_1
#define  T3      p1_2
#define  T4      p1_3

void MainClockInit(void)
{
        asm("FCLR I");
        prcr = 1;
        cm13 = 1;
        cm14 = 0;
        cm15 = 1;
        cm05 = 0;
        cm16 = 0;
        cm17 = 0;
        cm06 = 0;
        asm("nop");
        asm("nop");
        asm("nop");
        asm("nop");
        ocd2 = 0;
        prcr = 0;
        asm("FSET I");
```

```
}
void init_timer()
{
	//mk sebagai sumber clock dengan keluaran pada pin CNTR0
	//timer X pulse output mode
	txmod0=1;
	txmod1=0;
	txmod2=0;
	r0edg=0;
	txocnt=0;
	txedg=0;
	txund=0;

	txck0=1;
	txck1=0;
}
void init_io()
{
	prc2 = 1;
	pd0 = 0xff;
	pd1 = 0b10000000;
	pd3 = 0;
	pd4 = 0;

	pur1 = 0xff;
	pur0 = 0xff;
	prc2 = 0;
}
void main()
{
	MainClockInit();
	init_timer();
	init_io();

	while(1)
	{
		// frekuensi = frekuensi_timer/(2*(tx+1)*(prex+1))
		if(T1==0 && T2==1 && T3==1 && T4==1)
		{
			// frekuensi: 2,5 kHz
			txs = 0;
			tx = 1;
			prex = 249;
			txck1 = 0;
			txck0 = 1;
			txs = 1;
		}
		else if(T1==1 && T2==0 && T3==1 && T4==1)
		{
			// frekuensi: 3 kHz
			txs = 0;
			tx = 6;
			prex = 237;
			txck1 = 1;
			txck0 = 1;
			txs = 1;
		}
		else if(T1==1 && T2==1 && T3==0 && T4==1)
		{
			// frekuensi: 3,5 kHz
			txs = 0;
			tx = 2;
			prex = 118;
			txck1 = 0;
			txck0 = 1;
			txs = 1;
		}
		else if(T1==1 && T2==1 && T3==1 && T4==0)
		{
			txs = 0;
		}
	}
}
```

**Program Penghitung Pulsa Sensor Rotari**

```
#include"sfr_r813.h"
#include"lcd.h"

long pulsa_ka=0,pulsa_ki=0;
long data_ka=0,data_ki=0;
int arah_ka=0,arah_ki=0;
int proses=0;

#pragma INTERRUPT 25 dir_right
void dir_right(void)
{
        //check direction of right wheel
        int1ic=0;
        if(p3_1==0)arah_ka=1;           /*Forward*/
        else arah_ka=0;                 /*Backward*/
}
#pragma INTERRUPT 21 dir_left
void dir_left(void)
{
        //check direction of left wheel
        int2ic=0;
        if(p3_3==0)arah_ki=1;           /*Forward*/
        else arah_ki=0;                 /*Backward*/
}
#pragma INTERRUPT 24 procces
void procces(void)
{
        proses--;
}
void MainClockInit(void)
{
        asm("FCLR I");
        prcr = 1;
        cm13 = 1;
        cm14 = 0;
        cm15 = 1;
        cm05 = 0;
        cm16 = 0;
        cm17 = 0;
        cm06 = 0;
        asm("nop");
        asm("nop");
        asm("nop");
        asm("nop");
        ocd2 = 0;
        prcr = 0;
        asm("FSET I");
}
void init_timer()
{
//timer x as counter right wheel
txmod1 = 1;
txmod0 = 0;
r0edg = 1;
tx = prex = 255;
txs = 1;

//timer y as counter left wheel
tymod0 = 0;
r1edg = 1;
tyck1 = 1;
tyck0 = 1;
typr = prey = 255;
tys = 1;

//timer z as timer main proses
//          f = 1/8 fRING = 2,5 MHz
//          tPROSES = 0,0262 s
tzmod1 = 0;
tzmod0 = 0;
tzck1 = 0;
tzck0 = 1;
prez = 255;
tzpr = 255;
```

```
        tzs = 1;
}
void init_interrupt()
{
        tzic = 1;
}
void init_io()
{
        prc2 = 1;
        pd0 = 0xff;
        pd1 = 0b00011010;
        pd3 = 0;
        pd4 = 0;

        pur1 = 0xff;
        pur0 = 0xff;
        prc2 = 0;
}
void main()
{
        MainClockInit();
        init_timer();
        init_interrupt();
        init_io();
        init_lcd();

        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_puts("Program:");
        lcd_gotoxy(1,0);
        lcd_puts("Pulsa Counter");
        proses=50;

        while(1)
        {
                if(proses<=0)
                {
                        tzic = 0;
                        proses = 5;

                        data_ka = (long)((255-tx)*256+(255-prex));
                        tx = prex = 255;
                        data_ki = (long)((255-typr)*256+(255-prey));
                        typr = prey = 255;
                        if(arah_ka == 0)data_ka = -data_ka;
                        if(arah_ki == 0)data_ki = -data_ki;
                        pulsa_ka+=data_ka;
                        pulsa_ki+=data_ki;

                        lcd_clear();
                        lcd_gotoxy(0,0);
                        lcd_puts("Kanan:");
                        lcd_number(pulsa_ka);
                        lcd_gotoxy(0,0);
                        lcd_puts("Kiri:");
                        lcd_number(pulsa_ki);

                        tzic = 1;
                        int1ic = 2;
                        int2ic = 3;
                }
        }
}
```

**Program Uji SD Card**

```
#include "sfr_r813.h"
#include "lcd.h"
#include "sdc.h"

int n=0;
unsigned char response=0;
unsigned char sector[512];
unsigned long fbr_sector=0;
```

```
void MainClockInit(void)
{
	asm("FCLR I");
	prcr = 1;
	cm13 = 1;
	cm14 = 0;
	cm15 = 1;
	cm05 = 0;
	cm16 = 0;
	cm17 = 0;
	cm06 = 0;
	asm("nop");
	asm("nop");
	asm("nop");
	asm("nop");
	ocd2 = 0;
	prcr = 0;
	asm("FSET I");
}
void init_io(void)
{
	prc2 = 1;
	pd0 = 0xff;
	pd1 = 0b01011010;
	pd3 = 0;
	pd4 = 0;

	pur1 = 0xff;
	pur0 = 0xff;
	prc2 = 0;
}
void main(void)
{
	MainClockInit();
	init_io();
	init_lcd();
	init_spi();
	lcd_clear();
	lcd_gotoxy(0,0);
	lcd_puts("Program:");
	lcd_gotoxy(1,0);
	lcd_puts("Uji SD Card");
	delay_us(50000);

	while(init_sdc() != SUCCESS);
	clk1_u0c0 = 0;
	clk0_u0c0 = 0;
	u0brg     = 0;                  // Freq SCK = 20 MHz/2(0+1) = 10 MHz

	//block uji address= 0x60200 --> cluster 2
	lcd_clear();
	lcd_gotoxy(0,0);
	lcd_puts("sd card");
	lcd_gotoxy(1,0);
	lcd_puts("reading...");
	delay(50000);
	read_block(769);              // periksa isi sector 769 sebelum di tulis

	lcd_clear();
	lcd_gotoxy(0,0);
	lcd_puts("sd card");
	lcd_gotoxy(1,0);
	lcd_puts("writing...");
	delay(50000);
	for(n=0;n<256;n++)sector[n]=n;
	for(n=0;n<256;n++)sector[256+n]=n;
	write_block(769);             // tulis sector 769 dengan nilai

	lcd_clear();
	lcd_gotoxy(0,0);
	lcd_puts("sd card");
	lcd_gotoxy(1,0);
	lcd_puts("reading new...");
	delay(50000);
	read_block(769);              // periksa sector 769 apakah sudah terisi }
```

**Program Uji Perhitungan Posisi**

```
#include"sfr_r813.h"
#include"lcd.h"
#include"peta.h"
#include"float.h"
#include"math.h"

long pulsa_ka=0,pulsa_ki=0;
char arah_ka=0,arah_ki=0;
struct koordinat{
        float x;
        float y;
        float sudut;
};
struct koordinat lokal={0,0,0};
struct koordinat global={0,0,0};
struct posisi{
        int x;
        int y;
};
struct posisi blok={0,0};
struct posisi flag={0,0};
struct kondisi{
        char utara;
        char barat;
        char selatan;
        char timur;
};
struct kondisi dinding={0,0,0,0};
unsigned int sudut=0;

int proses=0,koreksi=0,depan=0;
unsigned int arah=0;
char sdc_insert=0;
char status=0;
unsigned char sector[512];
unsigned int fbr_sector=0;
unsigned int fat_sector=0;
unsigned int root_sector=0;
unsigned char sector_cluster=0;
unsigned int sector_fat=0;
unsigned long cluster=0;

#pragma INTERRUPT 25 dir_right
void dir_right(void)
{
        //check direction of right wheel
        int1ic=0;
        if(p3_1==0)arah_ka=1;          /*Forward*/
        else arah_ka=0;                /*Backward*/
}
#pragma INTERRUPT 21 dir_left
void dir_left(void)
{
        //check direction of left wheel
        int2ic=0;
        if(p3_3==0)arah_ki=1;          /*Forward*/
        else arah_ki=0;                /*Backward*/
}
#pragma INTERRUPT 24 procces
void procces(void)
{
        proses--;
}
void init_clock(void)
{
        asm("FCLR I");
        prcr = 1;
        cm13 = 1;
        cm14 = 0;
        cm15 = 1;
        cm05 = 0;
        cm16 = 0;
```

```
	cm17 = 0;
	cm06 = 0;
	asm("nop");
	asm("nop");
	asm("nop");
	asm("nop");
	ocd2 = 0;
	prcr = 0;
	asm("FSET I");
}
void init_timer(void)
{
	//timer x as counter right wheel
	txmod1 = 1;
	txmod0 = 0;
	r0edg = 1;
	tx = prex = 255;
	txs = 1;

	//timer y as counter left wheel
	tymod0 = 0;
	r1edg = 1;
	tyck1 = 1;
	tyck0 = 1;
	typr = prey = 255;
	tys = 1;

	//  timer z as timer main proses
	//		f = 1/8 fRING = 2,5 MHz
	//		tPROSES = 0,0262 s
	tzmod1 = 0;
	tzmod0 = 0;
	tzck1 = 0;
	tzck0 = 1;
	prez = 255;
	tzpr = 255;
	tzs = 0;
}
```

```
void init_interrupt(void)
{
	tzic = 1;
	int1ic = 2;
	int2ic = 3;
}
void init_io(void)
{
	prc2 = 1;
	pd0 = 0xff;
	pd1 = 0b01011000;
	pd3 = 0;
	pd4 = 0;

	pur1 = 0xff;
	pur0 = 0xff;
	prc2 = 0;
}
void main(void)
{
	//inisialisasi
	init_clock();
	init_timer();
	init_interrupt();
	init_io();
	lcd_init();
	spi_init();

	lcd_clear();
	lcd_gotoxy(0,0);
	lcd_puts("Program:");
	lcd_gotoxy(1,0);
	lcd_puts("Uji Posisi");
	delay_us(50000);

	proses = 8;
	tzs=1;
```

```
while(1)
{
        if(proses<=0)
        {
        proses = 8;

        ambil_data();
        hitung_lokal();
        hitung_global();
        flag.x = cek_posisi(&global.x,&blok.x);
        flag.y = cek_posisi(&global.y,&blok.y);
        cek_dinding();
        koreksi_data();

        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_puts("Blok:X=");
        lcd_number(blok.x);
        lcd_puts("  Y=");
        lcd_number(blok.y);
        lcd_gotoxy(1,0);
        lcd_puts("Arah=");
        switch(arah)
        {       case UTARA :    lcd_puts("utara");break;
                case BARAT :    lcd_puts("barat");break;
                case SELATAN:   lcd_puts("selatan");break;
                case TIMUR :    lcd_puts("timur");break;
        }

        int1ic = 2;
        int2ic = 3;

        }
};
}
```

**Program Uji Waktu**

```
#include "sfr_r813.h"
#include "lcd.h"
#include "peta.h"
#include "sdc.h"
#include "fat.h"
#include "float.h"
#include "math.h"

long pulsa_ka=0,pulsa_ki=0;
char arah_ka=0,arah_ki=0;
struct koordinat{
        float x;
        float y;
        float sudut;
};
struct koordinat lokal={0,0,0};
struct koordinat global={0,0,0};
struct posisi{
        int x;
        int y;
};
struct posisi blok={0,0};
struct posisi flag={0,0};
struct kondisi{
        char utara;
        char barat;
        char selatan;
        char timur;
};
struct kondisi dinding={0,0,0,0};
unsigned int sudut=0;

int proses=0,koreksi=0,depan=0;
unsigned int arah=0;
char sdc_insert=0;
char status=0;
```

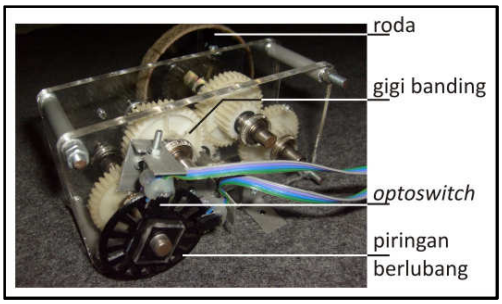
```
unsigned char sector[512];
unsigned int fbr_sector=0;
unsigned int fat_sector=0;
unsigned int root_sector=0;
unsigned char sector_cluster=0;
unsigned int sector_fat=0;
unsigned long cluster=0;

#pragma INTERRUPT 25 dir_right
void dir_right(void)
{
	//check direction of right wheel
	int1ic=0;
	if(p3_1==0)arah_ka=1;		/*Forward*/
	else arah_ka=0;			/*Backward*/
}
#pragma INTERRUPT 21 dir_left
void dir_left(void)
{
	//check direction of left wheel
	int2ic=0;
	if(p3_3==0)arah_ki=1;		/*Forward*/
	else arah_ki=0;			/*Backward*/
}
#pragma INTERRUPT 24 procces
void procces(void)
{
	proses--;
}

void init_clock(void)
{
	asm("FCLR I");
	prcr = 1;
	cm13 = 1;
	cm14 = 0;
	cm15 = 1;
	cm05 = 0;
	cm16 = 0;
	cm17 = 0;
	cm06 = 0;
	asm("nop");
	asm("nop");
	asm("nop");
	asm("nop");
	ocd2 = 0;
	prcr = 0;
	asm("FSET I");
}
void init_timer(void)
{
//timer x as counter right wheel
txmod1 = 1;
txmod0 = 0;
r0edg = 1;
tx = prex = 255;
txs = 1;

//timer y as counter left wheel
tymod0 = 0;
r1edg = 1;
tyck1 = 1;
tyck0 = 1;
typr = prey = 255;
tys = 1;

//  timer z as timer main proses
//		f = 1/8 fRING = 2,5 MHz
//		tPROSES = 0,0262 s
tzmod1 = 0;
tzmod0 = 0;
tzck1 = 0;
tzck0 = 1;
prez = 255;
tzpr = 255;
```

```
        tzs = 0;
}
void init_interrupt(void)
{
        tzic = 1;
        int1ic = 2;
        int2ic = 3;
}
void init_io(void)
{
        prc2 = 1;
        pd0 = 0xff;
        pd1 = 0b01011000;
        pd3 = 0;
        pd4 = 0;

        pur1 = 0xff;
        pur0 = 0xff;
        prc2 = 0;
}

void main(void)
{
        //inisialisasi
        init_clock();
        init_timer();
        init_interrupt();
        init_io();
        lcd_init();
        spi_init();

        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_puts("Program:");
        lcd_gotoxy(1,0);
        lcd_puts("Uji Waktu");
        delay_us(50000);

if(SDC==0)
{
        while(sdc_init()!=SUCCESS);
        clk1_u0c0 = 0;
        clk0_u0c0 = 0;
        u0brg     = 0;       // Freq SCK = 20 MHz/2(0+1) = 10 MHz
        fbr_sector=get_address_fbr();
        status=get_info_fbr();
        cluster=(unsigned long)cek_file();
        if(cluster==0 && status==1)
        {       make_peta();
                lcd_clear();
                lcd_gotoxy(1,0);
                lcd_puts("build file...");
                delay_us(30000);
        }
        else if(cluster!=0 && status==1)
        {       lcd_clear();
                lcd_gotoxy(1,0);
                lcd_puts("file exist...");
                delay_us(30000);
        }
        else if(status==0)
        {
                lcd_clear();
                lcd_gotoxy(1,0);
                lcd_puts("FAT error...");
                delay_us(30000);
        }
}
else
{
        lcd_clear();
        lcd_gotoxy(1,0);
        lcd_puts("SD Card missing");
        delay_us(30000);
}
```

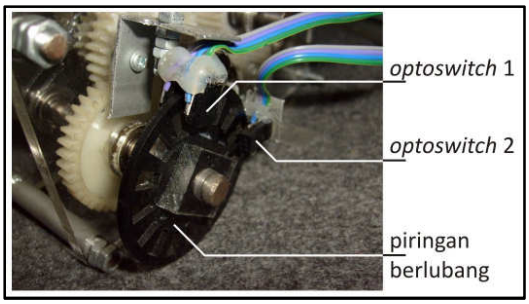
```
// timer Z dijalankan
tzpr=prez=proses=255;
tzs=1;

ambil_data();
hitung_lokal();
hitung_global();
flag.x = cek_posisi(&global.x,&blok.x);
flag.y = cek_posisi(&global.y,&blok.y);
cek_dinding();
koreksi_data();
if(status==1) store_peta();

lcd_clear();
lcd_gotoxy(0,0);
lcd_puts("Blok:X=");
lcd_number(blok.x);
lcd_puts("  Y=");
lcd_number(blok.y);
lcd_gotoxy(1,0);
lcd_puts("Arah=");
switch(arah)
{       case UTARA :    lcd_puts("utara");break;
        case BARAT :    lcd_puts("barat");break;
        case SELATAN:   lcd_puts("selatan");break;
        case TIMUR :    lcd_puts("timur");break;
}
lcd_gotoxy(1,13);
lcd_number(sudut);
int1ic = 2;
int2ic = 3;

// timer Z dimatikan
tzs=0;
lcd_clear();
lcd_gotoxy(0,0);
lcd_puts("UNDERFLOW=");lcd_number(proses);
lcd_gotoxy(1,0);
lcd_puts("TZ=");lcd_number(tzpr);
lcd_puts(" PREZ=");lcd_number(prez);

}
```

# LAMPIRAN III

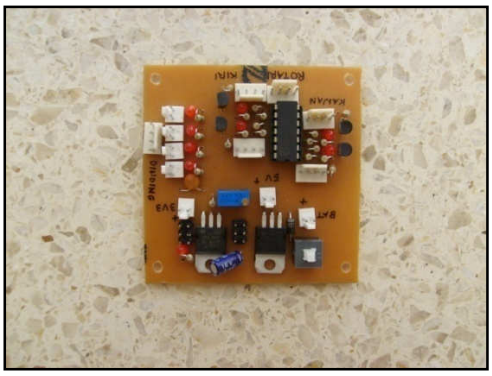
**Foto Alat**



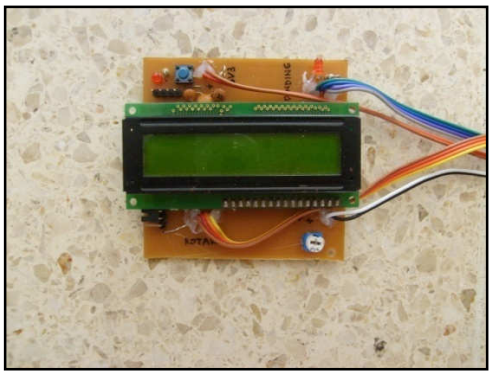

Gambar roda gigi sensor rotari




Gambar posisi *optoswitch* sensor rotari

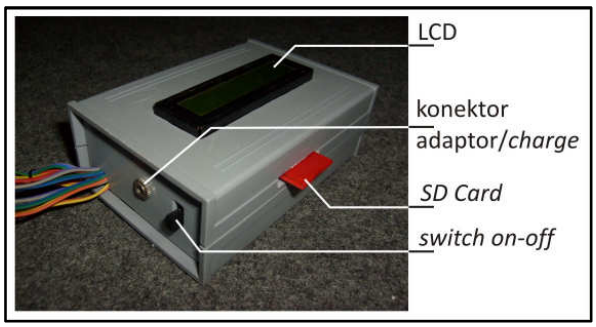


Gambar PCB I alat

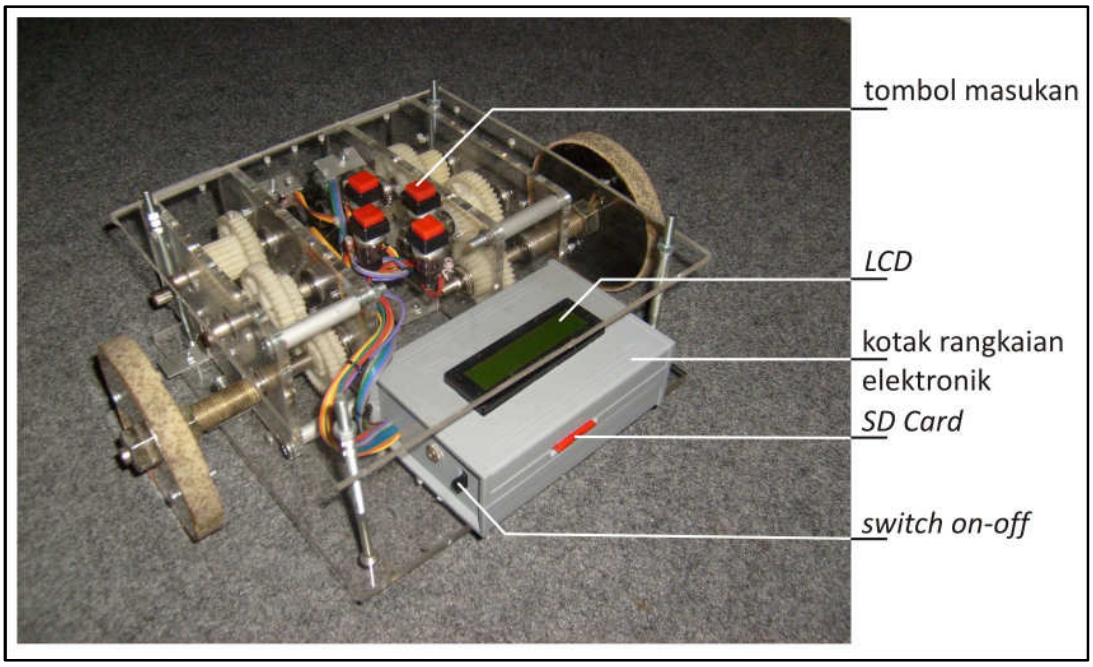


Gambar PCB II alat

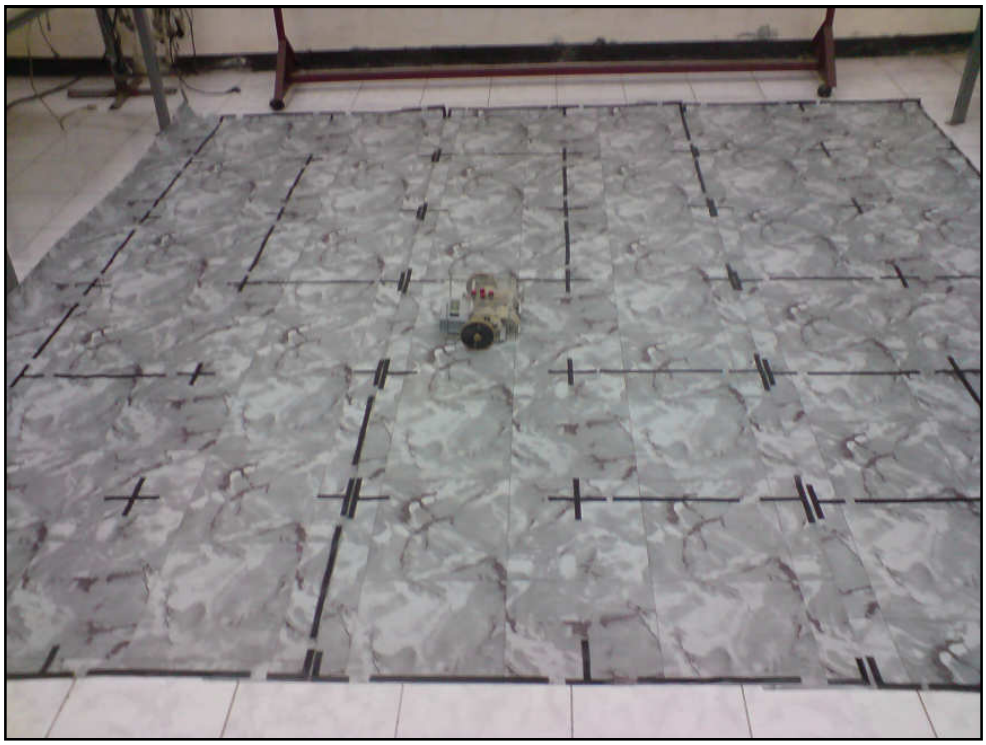
Gambar bagian dalam alat




Gambar bagian elektronik alat




Gambar alat keseluruhan



Gambar alat dan lapangan uji