

Free CD

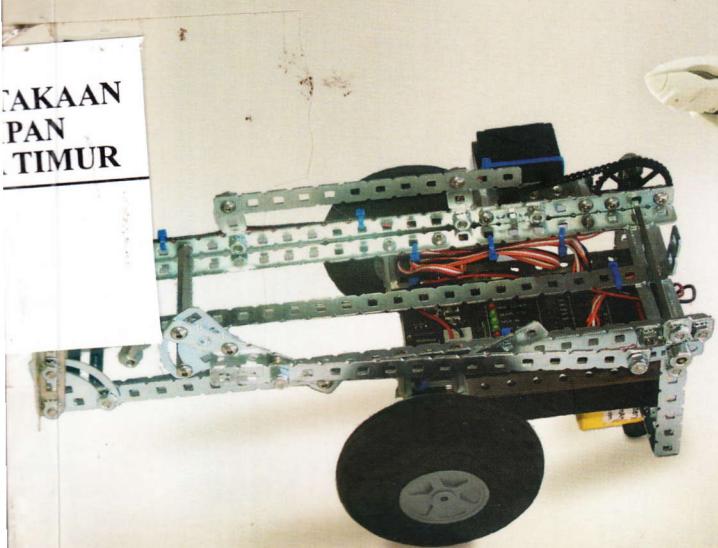
- Aplikasi-aplikasi Pendukung
- Datasheet
- Assembler
- Basic Stamp Editor
- Bascom AUR
- Code Vision C AUR

M e m b u a t S e n d i r i

Robot

Humanoid

*Ciptakan sendiri robot mirip manusia
berbiaya murah*



**Widodo Budiharto
Paulus Andi Nalwan**



Membuat Sendiri Robot Humanoid

Sanksi Pelanggaran Pasal 22:

Undang-Undang Nomor 19 Tahun 2002

Tentang Hak Cipta

1. Barangsiapa dengan sengaja melanggar dan tanpa hak melakukan perbuatan sebagaimana dimaksud dalam Pasal 2 Ayat (1) atau Pasal 49 Ayat (1) dan Ayat (2) dipidana dengan pidana penjara masing-masing paling singkat 1 (satu) bulan dan/atau denda paling sedikit Rp 1.000.000,00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp 5.000.000.000,00 (lima miliar rupiah).
2. Barangsiapa dengan sengaja menyiarakan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran hak cipta atau hak terkait sebagai dimaksud pada Ayat (1) dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).

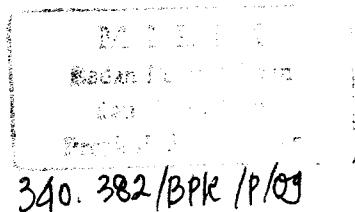
Membuat Sendiri Robot Humanoid

Widodo Budiharto
&
Paulus Andi Nalwan

Penerbit PT Elex Media Komputindo
 KOMPAS GRAMEDIA

Membuat Sendiri Robot Humanoid

Widodo Budiharto & Paulus Andi Nalwan



©2009, PT Elex Media Komputindo, Jakarta

Hak cipta dilindungi undang-undang

Diterbitkan pertama kali oleh

Penerbit PT Elex Media Komputindo

Kelompok Gramedia, Anggota IKAPI, Jakarta 2009

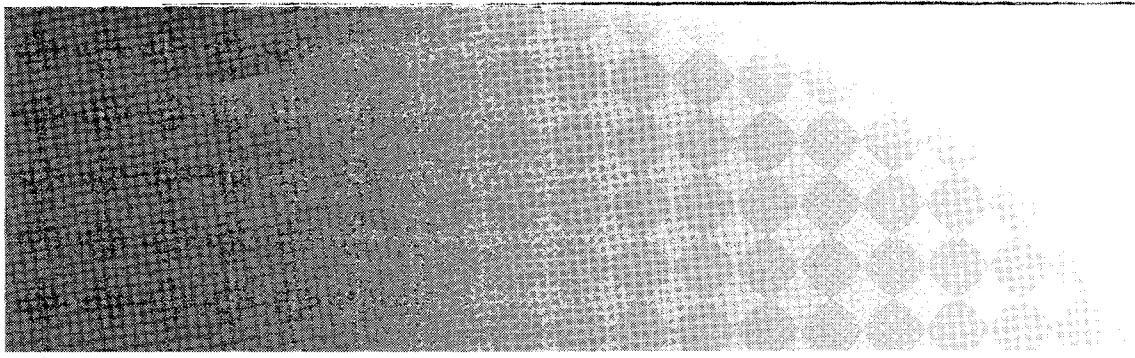
121091564

978-979-27-5625-8

[eEp]

Dilarang keras menerjemahkan, memfotokopi, atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari penerbit.

Dicetak oleh Percetakan PT Gramedia, Jakarta
Isi di luar tanggung jawab percetakan



Kata Pengantar

Membuat Robot humanoid dengan teknologi terkini merupakan hal yang menarik bagi pelajar dan penghobi elektronika. Untuk itulah, penulis dengan senang hati meluncurkan buku panduan membuat robot humanoid. Buku ini sangat tepat dibaca oleh pelajar, mahasiswa, serta penghobi elektronika serta para perekayasa. Isi buku ini membahas secara detail bagaimana membuat robot humanoid yang dikontrol oleh mikrokontroler handal. Buku ini merupakan buku robot terlengkap yang pernah ada di Indonesia.

Untuk memajukan teknologi robotika di Indonesia, penulis juga mempopulerkan teknologi sensor dan robot terkini pada buku ini. Kolaborasi materi 2 praktisi dan penghobi robotika pada buku ini tentunya sangat menarik untuk dibaca dan dicoba.

Tidak ada cara lain untuk mahir robotika selain mencoba langsung. Hanya membaca saja tidak ada artinya. Untuk mempermudah Anda mencoba berbagai proyek robot di buku ini, Anda dapat memiliki kit yang dibahas melalui situs e-commerce penulis di www.delta-electronic.com atau www.toko-robot.com dan www.toko-elektronika.com. Buku ini juga dilengkapi dengan berbagai contoh aplikasi siap terap yang harus Anda coba dan penuh dengan materi dan informasi penting, serta program penting dapat Anda peroleh pada CD pelengkap.

Penulis juga tidak lupa mengucapkan terima kasih sebesar-besarnya kepada Bpk. Dr.Ir.Djoko Purwanto, M.Eng yang telah membimbing dan mengizinkan penulis menggunakan sarana riset di Lab ELKA ITS.

Jika Anda ingin mengikuti training secara inhouse atau private, atau ingin mengadakan seminar atau workshop dapat menghubungi penulis di:

CV Pusat e-Technology

Perumahan Mutiara Gading Timur I , Jalan Cendana 5, Blok D3/2 RT 6/24
Bekasi Timur – Jawa barat 17510

HP:08569887384, Yahoo Messg: **prof_widodo**

Email:Widodo@widodo.com

www.Toko-elektronika.com

www.toko-robot.com

www.Widodo.com

Sebagai penutup, penulis juga mengharapkan saran dan kritik yang membangun guna penyempurnaan buku ini. Anda juga dapat berkomunikasi dengan penulis melalui email di Widodo@widodo.com atau Paulus@delta-electronic.com.

Jakarta, 1 Agustus 2009

Widodo Budiharto & Paulus Andi Nalwan

Daftar Isi

KATA PENGANTAR	V
DAFTAR ISI.....	VII
BAB 1 MENGENAL ROBOT HUMANOID.....	1
Pendahuluan.....	1
Mengenal Robot.....	1
Sejarah Robot.....	1
Robot Humanoid.....	3
Arsitektur Robot Humanoid	5
Latihan	6
BAB 2 PEMROGRAMAN ROBOT DASAR.....	9
Pendahuluan.....	9
Mikrokontroler AVR ATmega 8535	9
Program C Compiler untuk AVR	11
Instalasi CodeVision AVR C Compiler	12
Mencoba Membuat Program	13
Penerapan pada Program	17
Penerapan Output dan Fungsi Delay	17
Penerapan Input Output.....	18
Membuat Robot Avoider	19
Latihan	22

BAB 3 ROBOT HUMANOID BRAT	25
Pendahuluan	25
Mengenal Robot BRAT	25
Perangkat Keras.....	25
Mencoba Mikrokontroler Basic Atom Pro	26
Merakit Robot BRAT	30
Latihan.....	48
BAB 4 LENGAN ROBOT 5 SUMBU GERAKAN	53
Pendahuluan	53
Aplikasi Penggerak Lengan Robot Menggunakan PC.....	59
Aplikasi Penggerak Lengan Robot Menggunakan Mikrokontroler	67
Program Penggerak Lengan Robot Otomatis	68
Latihan.....	76
BAB 5 SENSOR-SENSOR ROBOT.....	77
Pendahuluan	77
Rotary Encoder.....	77
Sensor Jarak Ultrasonik.....	81
Modul Suara D-Voice 04	87
Proses D-Voice secara Manual	89
Proses D-Voice secara UART/Serial dengan Mikrokontroler	91
Pengalamatan Memori D-Voice 04	93
Sensor Penjejak Garis	98
Penjejak Garis dengan Sungut.....	98
Penjejak Garis Tunggal	99
Sensor Warna TCS230.....	102
Proses Pengukuran Nilai Frekuensi	107
Latihan.....	110
BAB 6 BAGIAN PENGENDALI ROBOT	111
Pendahuluan	111
Bagian Input	112
Remote dengan Media Inframerah.....	113
Aplikasi Penerima Data dari Remote Televisi	115
Remote dengan Media Gelombang Radio.....	122
Aplikasi Penerima Data dari D-Joy Controller	129
Bagian Otak.....	134
Download Program melalui Parallel Port.....	135

Download Program melalui USB Port.....	136
Bagian Motor	141
Motor DC.....	142
Motor Stepper.....	144
Motor Servo.....	148
Latihan	155
BAB 7 ROBOT LABA-LABA DENGAN 6 KAKI.....	157
Pendahuluan.....	157
Robot Laba-Laba	157
Pengaturan Gerak Ruas-Ruas Kaki	162
Gerakan-Gerakan Laba-Laba	164
Gerakan Langkah Maju.....	164
Gerakan Langkah Mundur	165
Gerakan Langkah Lurus	165
Gerakan Badan Kiri Maju.....	166
Gerakan Badan Kanan Maju	167
Gerakan Robot Berdiri	168
Hubungan Gerakan dengan Sudut Motor Servo	170
Gerakan Robot Maju dan Berputar.....	173
Latihan	193
BAB 8 COMPUTER VISION UNTUK ROBOT	195
Pendahuluan.....	195
Mengenal Computer Vision pada Robot	195
Program OpenCV	195
Pemrograman Dasar OpenCV.....	196
Pengenalan Wajah	200
Haar Cascade Classifier.....	200
Konsep Pendekripsi Wajah	200
Tracking Wajah	204
Latihan	208
BAB 9 ROBOT PELAYAN HUMANOID	211
Pendahuluan.....	211
Rancangan Robot Pelayan	211
Arsitektur Robot Pelayan	211
Perangkat Lunak Robot Pelayan	214

Face Recognition	214
Konsep Dasar	214
Latihan.....	222
DAFTAR ACCUAN.....	225
LAMPIRAN	227
TENTANG PENULIS.....	229

Bab 1

Mengenal Robot Humanoid

Pendahuluan

Robot humanoid gencar dikembangkan saat ini, karena penerapan robot ke depan akan difokuskan menjadi robot yang memiliki fitur mirip manusia. Bahkan, diharapkan memiliki kemampuan berinteraksi dan berprilaku seperti manusia yang dikenal dengan *Human Robot Interaction*.

Mengenal Robot

Sejarah Robot

Ada banyak hal yang menarik dan tidak membosankan jika Anda bermain dengan elektronika, di antaranya adalah membuat robot. Jika Anda pernah atau memiliki hobi merakit mobil Tamiya, Anda sudah memiliki bekal dasar untuk membuat robot. Karena prinsip yang digunakan pada mobil Tamiya juga banyak digunakan pada robot, yaitu dasar mekanik mesin, roda, dan sumber catu daya.

Kata robot sendiri diperkenalkan ke publik oleh Karel Capek pada saat mementaskan RUR (Rossum's Universal Robots). Namun, awal munculnya Robot dapat diketahui dari bangsa Yunani kuno yang membuat patung yang dapat dipindah-pindahkan. Sekitar 270 BC, Ctesibus, seorang insinyur Yunani membuat organ dan jam air dengan komponen yang dapat dipindahkan. Zaman Nabi Muhammad SAW pun, telah dibuat mesin perang yang menggunakan roda dan dapat melontarkan bom. Bahkan, Al-Jazari (1136-1206) seorang ilmuwan Islam pada dinasti Artuqid adalah yang dianggap pertama kali menciptakan robot humanoid, hebat kan? Nah, Anda sendiri bagaimana? Di zaman sudah maju begini, apa yang telah Anda buat berkenaan dengan robot?

Pada tahun 1770, Pierre Jacquet Droz, seorang pembuat jam berkebangsaan Swiss, membuat 3 boneka mekanis. Uniknya, boneka tersebut dapat melakukan fungsi spesifik, yaitu dapat menulis, yang lainnya dapat memainkan musik dan organ, dan yang ketiga dapat menggambar. Pada tahun 1898, Nikola Tesla membuat sebuah boat yang dikontrol melalui radio remote control dan didemokan di Madison Square Garden. Namun usaha untuk membuat autonomus boat tersebut gagal karena masalah dana.

Pada tahun 1967, Jepang yang pada saat itu merupakan negara yang baru bangkit, mengimpor robot Versatron dari AMF. Awal kejayaan robot dimulai pada tahun 1970, ketika Profesor Victor Scheinman dari Universitas Stanford mendesain lengan standar. Saat ini, konfigurasi kinematiknya dikenal sebagai standar lengan robot. Terakhir, pada tahun 2000 Honda memamerkan robot yang dibangun bertahun-tahun lamanya bernama ASIMO, serta disusul oleh Sony dengan robot anjing AIBO.

Robot banyak dibuat oleh Institusi Riset, Universitas, Departemen Pertahanan, serta institusi besar lainnya seperti NASA dan Sony. Saat ini hampir semua industri manufaktur menggunakan robot karena biaya per jam untuk mengoperasikan robot jauh lebih murah dibandingkan menggunakan manusia. Robot pada awalnya digunakan untuk melakukan fungsi spesifik, misalnya pengecoran, penyolderan, dll. Namun, saat ini sudah banyak robot yang melakukan banyak fungsi.

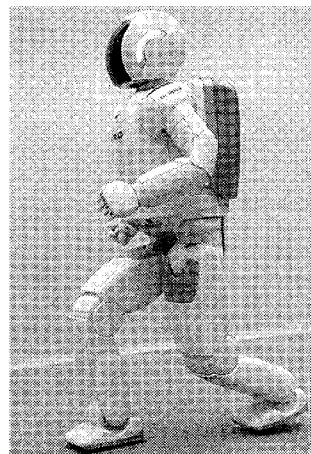
Beberapa penerapan robot saat ini antara lain:

- Merakit dan mengelas kerangka mobil di industri manufaktur.
- Pencari dan pemadam sumber api, intelijen.

- Pelayan toko, robot boneka, robot medis, robot perang (war robot).
- Nanotechnology. Pada tahun 1990, peneliti IBM menggunakan STM (Scanning Tunneling Microscope) untuk memindahkan 35 atom xenon pada kristal nicker untuk membuat huruf “IBM”.
- Multifunction Automated Crawling System (MACS) yang berguna untuk menginspeksi suatu ruang di aircraft yang susah dijangkau oleh manusia.
- Robot penjelajah/explorer, misalnya mencari informasi di bulan atau planet Mars dengan nama Sojourner. NASA mempunyai program telerobotic yang dioperasikan di Office of Space Access and Technology (OSAT).
- Fungsi-fungsi di bidang kedokteran, sosial, dan lain-lain.

Robot Humanoid

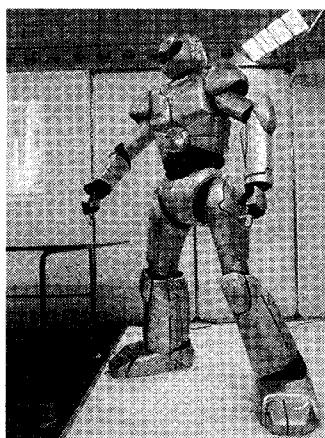
Robot humanoid adalah robot dengan tampilan keseluruhannya mirip dengan tubuh manusia yang membuat mampu berinteraksi secara sosial. Robot humanoid memiliki tingkat kesulitan yang lebih tinggi dibanding robot lainnya. Robot humanoid yang cukup menarik yaitu ASIMO dari Honda seperti Gambar 1.1



Gambar 1.1 Robot Asimo dari HONDA

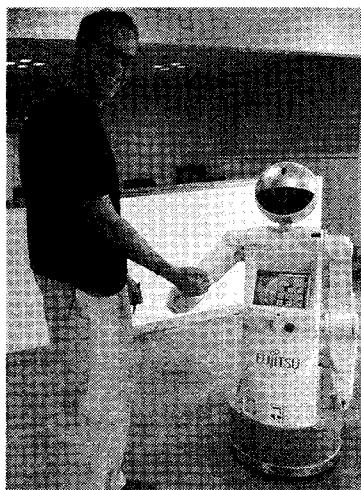
Bab 1: Mengenal Robot Humanoid

Robot humanoid saat ini banyak dibuat untuk robot pelayan atau peniru manusia, misalnya robot pemain ping pong TOPIO di bawah.



Gambar 1.2 Robot pemain Ping Pong TOPIO

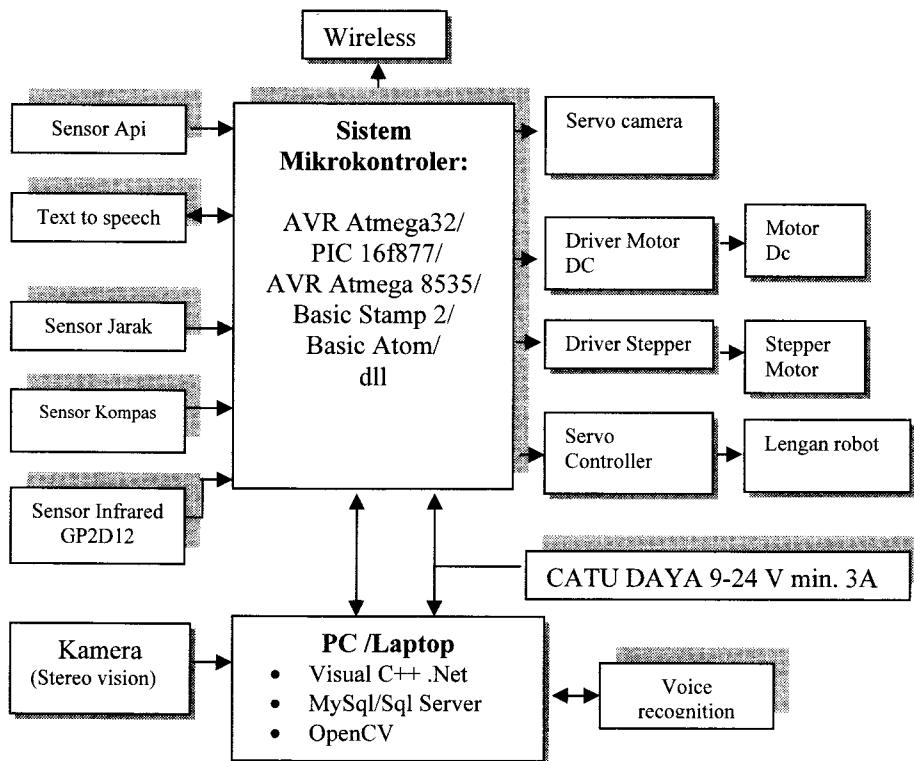
Jika Anda tertarik untuk membuat robot humanoid, saya sarankan Anda mencoba robot humanoid yang berfungsi sebagai robot pelayan (*servant robot*), seperti robot ENON ini yang sangat cerdas. Percayalah, Anda dapat membuatnya.



Gambar 1.3 Robot Asisten pribadi ENON

Arsitektur Robot Humanoid

Gambar di bawah merupakan blok diagram robot humanoid standar yang dapat Anda buat.



Gambar 1.4 Rancangan Robot humanoid standar (Hak Cipta Penulis)

Pada gambar di atas, intinya Anda dapat menggunakan berbagai pengontrol/mikrokontroler untuk membuat robot humanoid tersebut secerdas mungkin. Anda dapat menggunakan mikrokontroler standar yang umum beredar seperti AVR, Basic Stamp, dan Basic Atom dengan kemampuan luar biasa. Semua input yang diterima oleh sensor akan diolah oleh mikrokontroler. Lalu melalui program yang telah kita buat, mikrokontroler akan melakukan aksi ke aktuator seperti lengan robot atau roda dan kaki robot. Teknologi wireless yang digunakan di atas adalah agar robot dapat mentransmisikan data atau menerima perintah secara jarak jauh. Sedangkan PC/Laptop diguna-

kan untuk program utama serta melakukan proses komputasi data dan images kecepatan tinggi yang tidak mampu dilakukan oleh mikrokontroler standar. Untuk memberikan catu daya pada robot dapat digunakan baterai, aki, atau solar cell.

Ok, sampai kapan Anda hanya terpana dengan gambar di atas? Sudah saatnya Anda sebagai generasi penerus bangsa terutama yang menekuni bidang elektronika memikirkan dan berupaya agar mampu mengejar ketertinggalan kita dari negara lainnya. Jika Anda hanya hobi tanpa memiliki bekal pengetahuan dasar di bidang elektronika, tidak masalah. Yang penting Anda mau belajar dan mencoba sekarang juga. Demikianlah pengantar robot humanoid yang dapat dijadikan sumber pijakan utama bagi Anda yang ingin mengembangkan robot humnoid yang Anda impikan.

Latihan

Yang membuat Anda mahir robotika tidak lain adalah pengalaman dalam berekspeten dan ketekunan. Untuk itu, Anda diharapkan menjawab dan mengerjakan semua latihan ini.

1. Sebagai seorang pemula/pelajar, Anda membutuhkan informasi pelengkap terkini tentang dunia elektronika dan robot melalui Internet. Akseslah situs google.com atau yahoo.com untuk mencari informasi komponen elektronika untuk dapat menjelaskan cara kerja dari:
 - a. Resistor
 - b. Kapasitor elektrolit dan kapasitor keramik
 - c. Transistor NPN dan transistor PNP
 - d. MOSFET dan FET
 - e. Relay 5 V
 - f. Dioda penyearah
 - g. IC regulator variabel
 - h. Kamera webcam
 - i. Motor DC dan servo
2. Rangkailah catu daya berbasis solar cell amorphous 5.2V 21mA

3. Carilah info sebanyak mungkin yang berhubungan dengan aktuator robot humanoid, seperti servo, motor DC, stepper, pneumatic, dan lainnya.
4. Pelajarilah pemrograman Visual C++ .Net serta OpenCV.

Bab 2

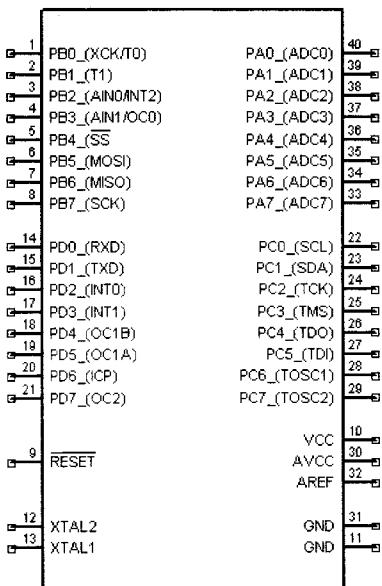
Pemrograman Robot Dasar

Pendahuluan

Pemrograman mikrokontroler merupakan dasar dari pengontrolan kerja robot. Orientasi dari penerapan mikrokontroler adalah untuk mengendalikan suatu sistem berdasarkan informasi input yang diterima, lalu diproses oleh mikrokontroler, dan dilakukan aksi pada bagian output sesuai program yang telah ditentukan sebelumnya.

Mikrokontroler AVR ATmega 8535

Mikrokontroler merupakan pengontrol utama robot yang akan dibuat. Mikrokontroler yang terkenal dan mudah didapatkan saat ini antara lain AVR ATmega 8535. Gambar berikut adalah susunan kaki standar 40 pin DIP mikrokontroler AVR ATmega 8535.



Gambar 2.1 Mikrokontroler ATmega8535/16

Pada gambar di atas terdapat 4 buah port, yaitu PA, PB, PC, dan PD yang semuanya dapat diprogram sebagai input atau output. Jika dilihat lebih detail lagi, pada bagian pemroses mikrokontroler ini terdapat unit CPU utama untuk memastikan eksekusi program. CPU juga dapat mengakses memori, melakukan kalkulasi, pengontrolan, dan penanganan interupsi. Ini dilakukan menggunakan arsitektur harvard (bus untuk memori, program, dan data terpisah) sehingga dihasilkan performa yang tinggi. Hal ini dikarenakan instruksi pada memori program dieksekusi dengan single level pipelining. Dengan demikian, pada saat sebuah instruksi dieksekusi, instruksi berikutnya dapat diakses dari memori program. Konsep ini memungkinkan instruksi dieksekusi pada setiap siklus clock.

Pin I/O pada mikrokontroler AVR dapat dikonfigurasi sebagai input atau output dengan cara mengubah isi I/O dari Data Direction Register. Misalnya, jika ingin port B dikonfigurasikan sebagai output, maka Data Direction Register port B(DDRB) harus di-set sebagai 0xFFH (sama dengan 255). Jika diinginkan sebagai input, DDRB harus di-set ke 0x00H (sama dengan 0).

Contoh

```
DDRB=255; //Port B dikonfigurasi sebagai output  
DDRD=0x00; // Port D dikonfigurasi sebagai input
```

V_{OH} (*output high voltage*) adalah tegangan pada pin I/O mikrokontroler ketika ia mengeluarkan logia “1” yang besarnya sekitar 4.2V dan arus sebesar $20mA(I_{OH})$. Setiap pin I/O mikrokontroler AVR memiliki *internal pull up resistor*. Misalnya, Port B dikonfigurasi sebagai input dan internal pull-up-nya diaktifkan, maka DDRB=00H dan PORTB=00H

Contoh

```
DDRB=0; // Port B dikonfigurasi sebagai input  
PORTB=0; //internal pull-up aktif
```

Untuk mendeteksi input pada salah satu port digunakan fungsi PINx, sedangkan untuk mendeteksi per pin dapat digunakan fungsi Pinx.bit

Contoh:

```
PORTB=PINC; //Semua data di Port C dikirim ke Port B  
PORTB.0=PINC.0 ; //Data di Port C.0 dikirim ke Port B.0
```

Program C Compiler untuk AVR

Ada banyak program yang dapat digunakan sebagai editor dan compiler program mikrokontroler. Yang paling dikenal adalah Bascom AVR, sedangkan program lainnya yang lebih powerfull adalah CodeVision AVR yang berbasis bahasa C. Oleh karena itu, pada praktikum digunakan program tersebut.

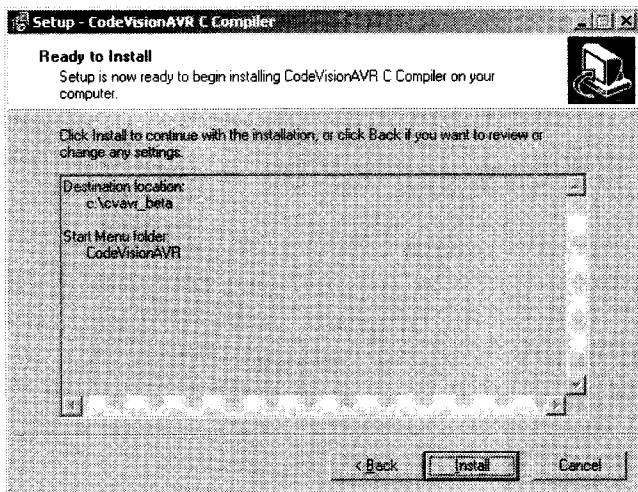
Instalasi CodeVision AVR C Compiler

1. Installah dengan mengeksekusi file setup.exe yang terdapat di CD pendamping buku sehingga muncul tampilan berikut:



Gambar 2.2 Tampilan awal instalasi

2. Klik Next sehingga tampil gambar di bawah ini. Pilih direktori instalasi Anda, lalu klik Next hingga mencapai Finish.

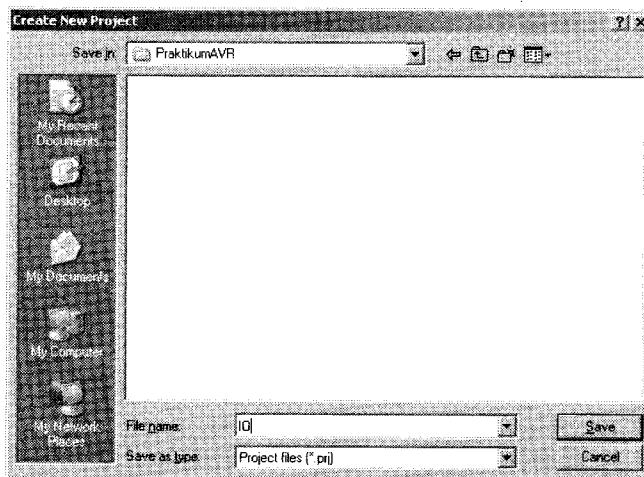


Gambar 2.3 Persiapan instalasi

Mencoba Membuat Program

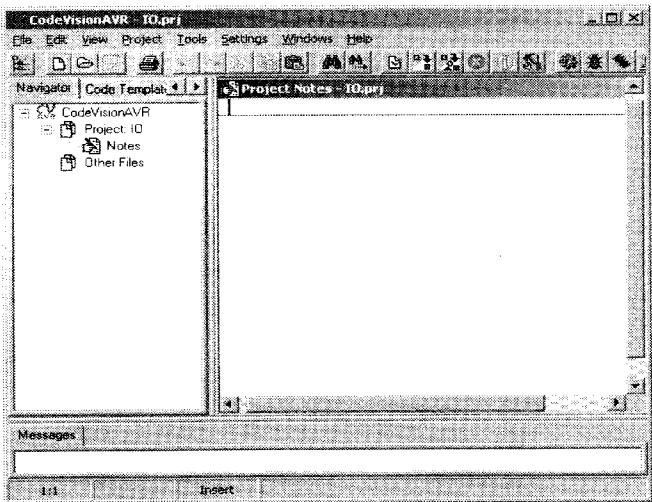
Setelah proses instalasi selesai, jalankan program tersebut hingga tampil program CodeVision AVR C Compiler. Untuk membuat proyek baru, langkah-langkahnya adalah:

1. Pilih menu **File | New**, lalu pilih **New Project**. Abaikan penawaran menggunakan Wizard. Beri nama project Anda pada folder yang Anda inginkan, misalnya bernama IO seperti gambar di bawah ini:



Gambar 2.4 Memberi nama proyek

2. Akan tampil form seperti gambar di bawah ini. Yang harus Anda lakukan adalah membuat file kode yang berekstension .c. Buatlah dengan cara mengklik menu **File | New**, lalu pilih **source**. Beri nama file tersebut, misalnya IO1 (otomatis menjadi IO1.c).



Gambar 2.5 Proyek yang sudah jadi

3. Simpan file tersebut, lalu buka lagi melalui menu File | Open. Pilih tipe file berekstension .c. Konfigurasikan juga proyek Anda, antara lain dengan menambahkan file source .c ini ke proyek Anda. Ketik program seperti gambar di bawah ini:

A screenshot of the CodeVisionAVR software interface showing the source code for 'IO.c'. The window title is 'CodeVisionAVR - IO.prj - [C:\cvavr_beta\examples\PraktikumAVR\IO\IO.c]'. The code is as follows:

```
1 // Mencoba Membuat Program CodeVision AVR
2 // Percobaan Output dan delay pada Port B
3 // LED terhubung di Port B
4 // Hak Cipta e-Technology Center 2008
5
6 #include <mega16.h> //menyertakan library file mega16
7 #include <delay.h> //menggunakan delay
8 //jadi librarynya harus digunakan
9
10 void main(void) { //Program Utama
11
12     DDREB=255; //Port B dikonfigurasi sebagai output
13     //yaitu PB0 -PB7
14     PORTB=255; //Semua Port B mengeluarkan
15     //logika 1 sehingga semua LED mati
16 }
```

Gambar 2.6 File source berekstension .c

Berikut ini adalah program yang harus Anda coba untuk mulai mempelajari mikrokontroler AVR. LED terhubung di Port B. Program akan membuat beberapa LED berkedip.

IO1.c:

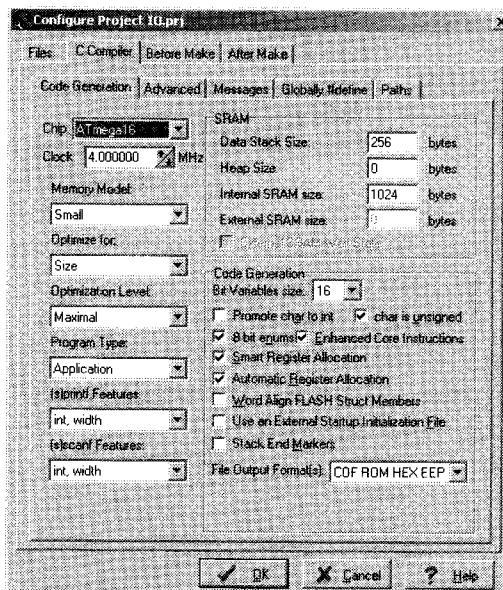
```
// Mencoba Membuat Program CodeVision AVR
// Percobaan Output dan delay pada Port B
// LED terhubung di Port B
// Hak Cipta e-Technology Center 2010

#include <mega16.h> //menyertakan library file mega16
#include <delay.h> //menggunakan delay jadi librarynya harus digunakan

void main(void) {    //Program Utama bertipe data void
    DDRB=255;    //Port B dikonfigurasi sebagai output, yaitu PB0 -PB7
    PORTB=255; //Semua Port B mengeluarkan logika 1, jadi semua LED
    mati

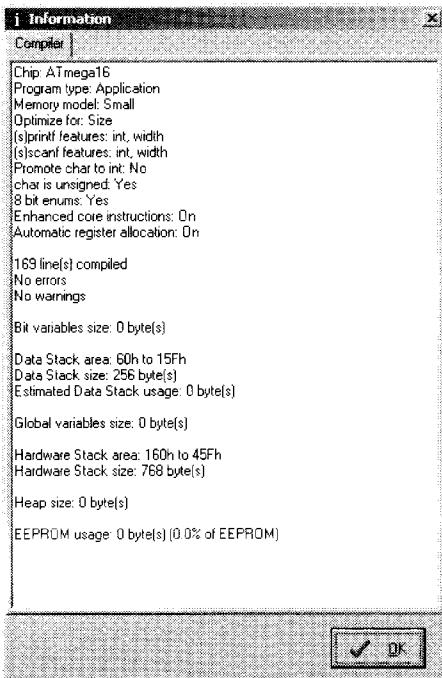
    // Program akan berulang terus
    while (1) {
        PORTB.0=0; //LED PB0 menyala
        PORTB.2=0; //LED PB2 menyala
        delay_ms(500); //Delay 0.5 detik
        PORTB.0=1; //LED PB0 mati
        PORTB.2=1; //LED PB2 mati
        delay_ms(1000); //Delay 1 detik
    }; //akhir looping
} //Akhir program utama
```

4. Pastikan Anda mengkonfigurasi proyek Anda dengan memastikan tipe mikrokontroler dan clock yang digunakan:



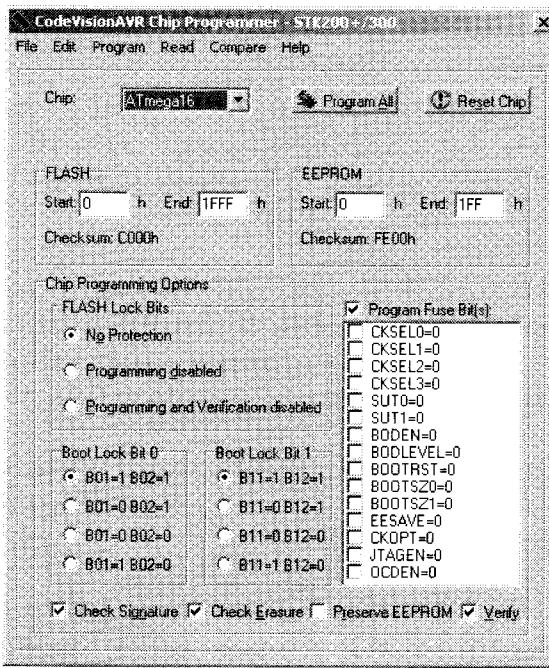
Gambar 2.7 Konfigurasi proyek

5. Setelah kode selesai dibuat, klik menu **Project | Compile** untuk mengkompilasi proyek Anda. Jika tidak ada kesalahan ketik, akan tampil informasi sukses seperti gambar di bawah ini;



Gambar 2.8 Hasil kompilasi yang sukses

6. Terakhir, Anda dapat mengisikan program tersebut ke mikrokontroler melalui menu **Tools | Chip Programmer**, lalu pilih **Program All**. Jika telah selesai, lampu LED Anda akan berkelap kelip dengan durasi 0.5 detik dan 1 detik. Pastikan programmer yang Anda gunakan adalah STK 200/300 (AVR ISP Programmer) berbasiskan port parallel. Jika Anda ingin mengisi melalui port serial/USB, Anda dapat menggunakan DT HiQ Universal Programmer.



Gambar 2.9 Mengisi program ke mikrokontroler

Penerapan pada Program

Penerapan Output dan Fungsi Delay

Langkah-langkahnya

1. Siapkan SmartAVR Robotics ver. 3 atau yang sesuai dan hubungkan dengan kabel AVR ISP Programmer ke PC.
2. Buat program di bawah ini:

Outportb.c:

```
// Percobaan Output dan delay pada Port B
// LED terhubung di Port B
#include <mega16.h> //menyertakan library file mega16
#include <delay.h> //menggunakan delay, jadi librarynya harus digunakan
void main(void) { //Program Utama
    DDRB=255; //Port B dikonfigurasi sebagai output, yaitu PB0 -PB7
    PORTB=255; //Semua Port B mengeluarkan logika 1 ,jadi LED mati
    // Program akan berulang terus
```

Bab 2: Pemrograman Robot Dasar

```
while (1) {
    PORTB.0=0; //LED PB0 menyala
    PORTB.1=1; // LED PB1 mati
    PORTB.2=0; //LED PB2 menyala
    delay_ms(500); //Delay 0.5 detik
    PORTB.0=1; //LED PB0 mati
    PORTB.1=0; // LED PB1 menyala
    PORTB.2=1; //LED PB2 mati
    delay_ms(1000); //Delay 1 detik
}; //akhir looping
} //Akhir program utama
```

3. Kompilasi dan jalankan. Pada alat akan tampil lampu LED yang berkedip secara bergantian sesuai dengan delay yang telah ditentukan. Beri komentar mengapa diperoleh hasil tersebut.

Penerapan Input Output

Langkah-langkahnya:

1. Siapkan SmartAVR Robotics ver. 3, dan hubungkan dengan kabel AVR ISP Programmer ke PC.
2. Buat program di bawah ini:

InputPortCkePortB.c:

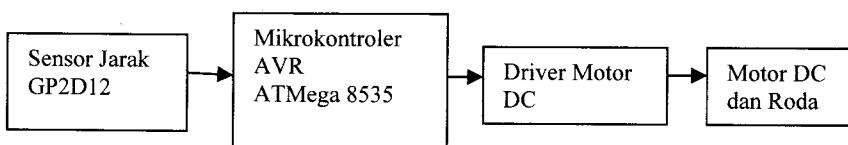
```
// Percobaan Input di Port C dan Output pada Port B
// Saklar di Port C, LED terhubung di Port B
#include <mega16.h> //Menyertakan library file ATmega16
#include <delay.h> //menggunakan delay, jadi librarynya harus digunakan
void main(void) { //Program Utama
    DDRB=255; //Port B dikonfigurasi sebagai output, yaitu PB0 -PB7
    PORTB=255; //Semua Port B mengeluarkan logika 1, semua LED mati
    PORTC =0; //Matikan internal pull up karena sudah ada
               //eksternal pull up pada rangkaian input
    #asm ("nop") //persiapan
               // Program akan berulang terus
while (1) {
    PORTB=PINC ; //Semua push button yang ditekan di Port C, datanya
dikirim ke Port B
    };
               //akhir looping
}
//Akhir program utama
```

3. Amati apa yang terjadi jika satu atau beberapa push button di Port C ditekan. Dapatkah Anda menjelaskan fungsi PINx?

Membuat Robot Avoider

Untuk dapat membuat robot humanoid yang canggih, maka Anda yang belum pernah membuat robot berbasiskan mikrokontroler, penulis wajibkan Anda mencoba robot ini terlebih dahulu. Jangan cuma dipikirkan/dibaca saja karena mungkin Anda akan bingung dan tidak terbayang tingkat kesulitannya. Penulis menyarankan Anda membaca buku sebelumnya, yaitu *Membuat Sendiri Robot Cerdas Edisi Revisi* terbitan Elex Media Komputindo 2009. Langkah-langkah perakitan robot avoider ini adalah sebagai berikut:

1. Siapkan perangkat yang dibutuhkan:
 - Kit mikrokontroler SmartAVR Robotics berbasis Atmega16 atau sejenisnya.
 - SPC DC Motor
 - Roda trolley
 - Sensor jarak Sharp GP2D12
 - Paket motor DC dan roda
 - Body robot
 - IDC 2x5 sebanyak 2 buah
 - Led Tester
 - Komponen serta tools pendukung
2. Hubungkan kit mikrokontroler tersebut dengan kabel AVR ISP Programmer ke PC. Hubungkan kabel serial dari mikrokontroler ke port serial PC.



Gambar 2.10 Blok diagram robot avoider

Bab 2: Pemrograman Robot Dasar

3. Hubungkan sensor Sharp GP2D12 di port A.0, keluaran PortB ke SPC DC Motor (S1-S4), dan keluaran SPC DC Motor ke Paket Motor DC GT2 atau GD20 Mini Metal gear motor.



Gambar 2.11 Paket Motor DC GT2 dan Roda (kiri) dan GD20 Mini Metal Gear Motor 100:1 lengkap dengan roda besar (kanan)

4. Rakitlah robot tersebut sesuai keinginan Anda.
5. Buat program di bawah ini, lalu masukkan ke mikrokontroler dengan bantuan CodeVision C AVR Compiler:

RobotAvoider.c:

```
// Percobaan baca data sensor jarak Sharp GP2D12
// Untuk Robot Avoider (Penghindar halangan)
// SPC DC Motor di Port B.0,B.1, B.2 dan B.3
// Sensor Jarak Sharp GP2D12 di Port A.0
// Hak Cipta Mr. Widodo Budiharto 2010

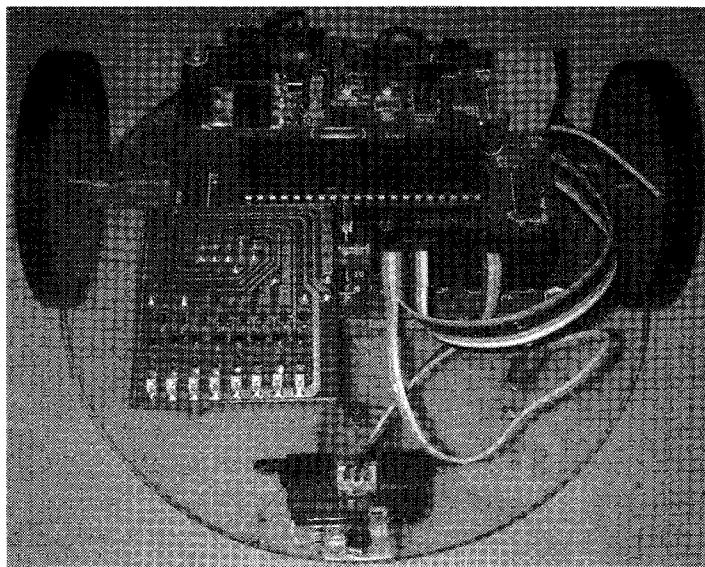
#include <mega8535.h>
#include <stdio.h>
#include <delay.h>
#define ADC_VREF_TYPE 0x60
flash unsigned char string1[]{"data  adc: %d; "};
// Baca 8 bit terpenting
unsigned char read_adc(unsigned char adc_input) {
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Mulai konversi ADC
ADCSRA|=0x40;
// Tunggu konversi ADC selesai
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
void main(void) {
// Inisialisasi Port B dan D sebagai output
PORTB=0xFF;
DDRB=0xFF;
PORTD=0xFF;
DDRD=0xFF;
```

```

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;
// ADC initialization
// ADC Clock frequency: 31,250 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: Free Running
// Menggunakan 8 bit data
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0xA7;
SFIOR&=0x1F;
while (1)
{
    PORTD=read_adc(0); // Baca data sensor di PA.0
    printf(string1,PORTD); // Tampilkan di Hyperterminal PC
    if (PORTD<100)
    {
        PORTB=10; // Maju
        delay_ms(100);
    }
    else
    {
        PORTB=0; // Mundur
        delay_ms(100);
        PORTB=1; // Belok kanan
        delay_ms(50);
    }
};
}

```

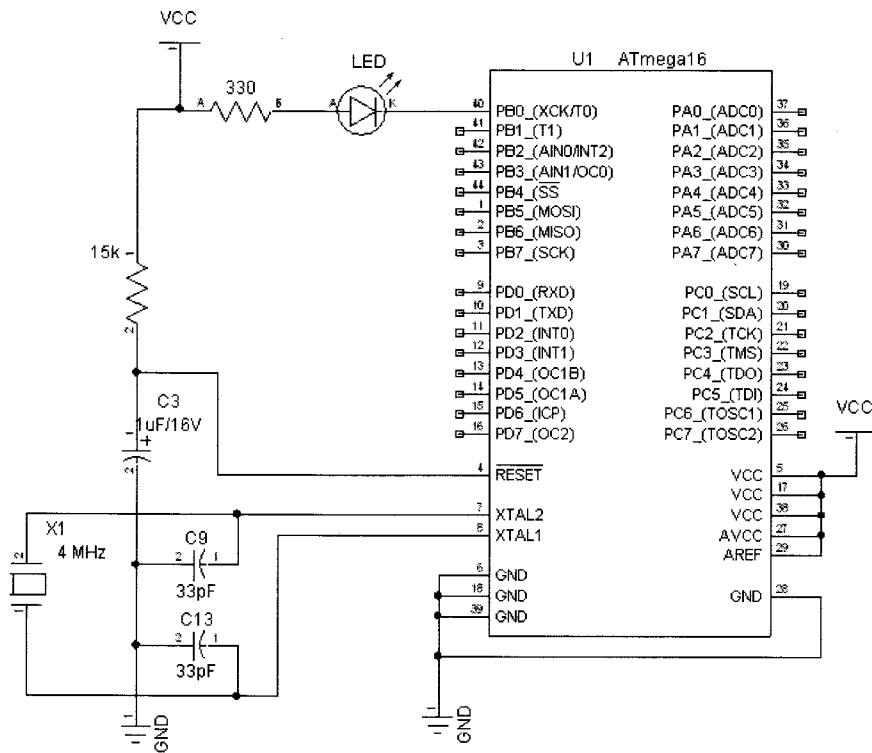
Sensor jarak mengeluarkan tegangan analog yang sesuai dengan jarak yang terukur. Keluaran tegangan analog sensor infrared yang tidak linear, oleh mikrokontroler diubah menjadi data digital untuk kemudian nilainya dibandingkan dengan program dan ditentukan apakah nanti harus maju, belok, atau mundur. Mikrokontroler memberikan sinyal tertentu ke driver motor DC agar menggerakkan motor DC sesuai dengan yang telah kita tentukan di program. Hasil dari robot tersebut dapat dilihat seperti gambar di bawah ini:



Gambar 2.12 Robot Avoider yang sudah jadi

Latihan

1. Gambarkan arsitektur I/O pada mikrokontroler AVR ATmega 8535.
2. Berikan contoh kode C untuk membuat port mikrokontroler AVR ATmega 8535 berlaku sebagai I/O, dengan kondisi berikut:
 - Port B sebagai input yang terhubung ke 8 buah sakelar, internal pull up mati, eksternal pull-up aktif.
 - Port D sebagai output yang terhubung ke relay yang diaktifkan melalui sebuah transistor.
3. Jelaskan cara kerja lampu LED yang dikontrol oleh mikrokontroler di bawah ini:



Gambar 2.13 Rangkaian sistem minimum mikrokontroler ATmega16

4. Buatlah program robot humanoid yang menggunakan 2 mata berupa kamera (stereo vision). Kedua mata tersebut harus dapat mengikuti arah muka kita. Informasi lebih lanjut dapat dibaca di Bab 8 dan 9.
5. Buatlah program yang mampu mendeteksi kedipan mata kita menggunakan OpenCV, untuk melakukan aksi tertentu.

Bab 3

Robot Humanoid BRAT

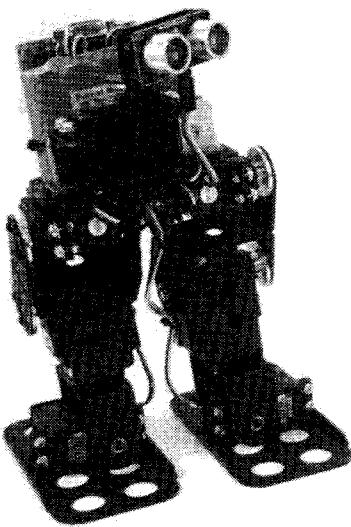
Pendahuluan

Robot humanoid berkaki yang paling banyak diminati untuk dipelajari adalah BRAT. Robot ini dilengkapi dengan sensor accelerometer sehingga dapat memberikan keseimbangan pada body robot tersebut. Memiliki pengetahuan pemrograman dan mekanika robot humanoid berkaki mutlak bagi Anda.

Mengenal Robot BRAT

Perangkat Keras

Robot BRAT merupakan robot humanoid berkaki yang digerakkan menggunakan 6 servo dan dikendalikan oleh mikrokontroler berbasis Basic Atom. BRAT singkatan dari *Bipedal Robotic Articulating Transport*. Karena menggunakan 6 servo, ia memiliki 3 degree of freedom (DOF). Mikrokontroler Basic Atom memiliki kemampuan luar biasa dibandingkan mikrokontroler lainnya. Selain itu, pemrogramannya juga sangat mudah.



Gambar 3.1 Robot Berkaki BRAT

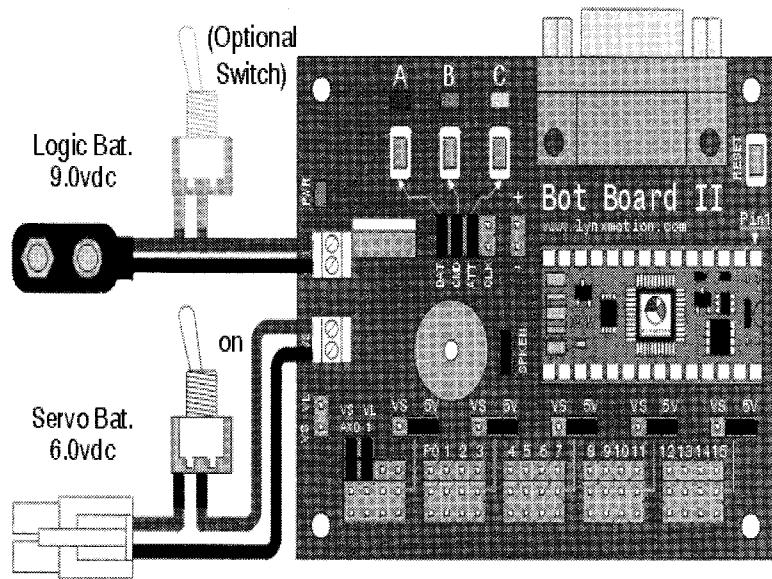
Pada gambar di atas, robot dapat berjalan dengan teliti menggunakan 6 buah servo Hitec. Ia menggunakan sensor jarak PING yang mampu mendeteksi jarak penghalang di depannya hingga 3 meter.

Mencoba Mikrokontroler Basic Atom Pro

Modul Basic Atom Pro pada kit Bot Board II ini jauh lebih handal dibandingkan Basic Stamp karena memiliki kelebihan antara lain ada fitur ADC dan kecepatan eksekusi yang sangat tinggi. Cobalah program sederhana berikut untuk mencoba menggunakan kit mikrokontroler ini.

Langkah-langkahnya:

1. Pasanglah modul ini ke PC dan berikan sumber catu daya.



Gambar 3.2 Kit Bot Board II

2. Install program Basic Atom Pro IDE.
3. Ketik dan kompilasi program di bawah ini. Pelajari juga setidaknya 20 fungsi dasar yang ada pada compiler Basic Atom Pro tersebut.

Aprotut1.bas:

```

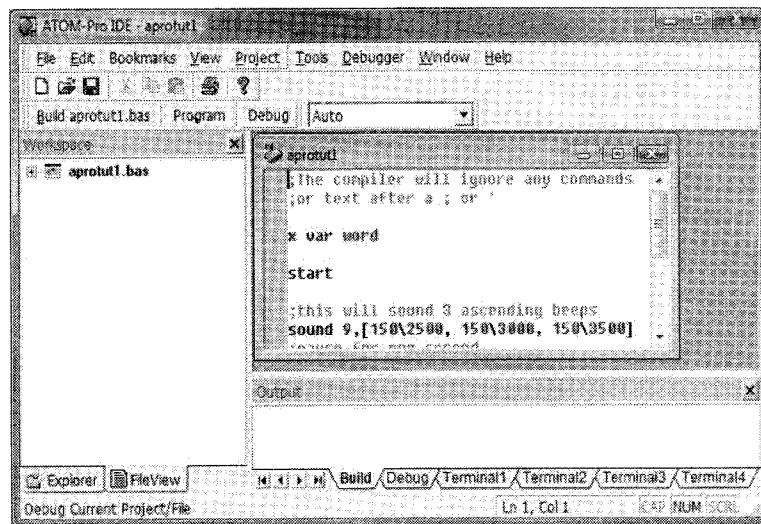
x var word
start

;Menghasilkan bunyi beep.
sound 9,[150\2500, 150\3000, 150\3500]
;pause for one second
pause 1000
;add one to the count
x = x + 1
;Kirim x kembali ke komputer
serout s_out,i9600,[DEC x, 13]

;repeat
goto start

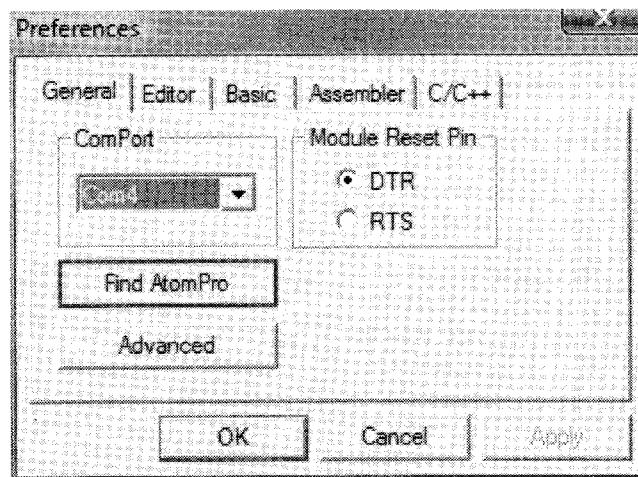
```

Gambar berikut adalah tampilan editor dari Basic Atom Pro IDE



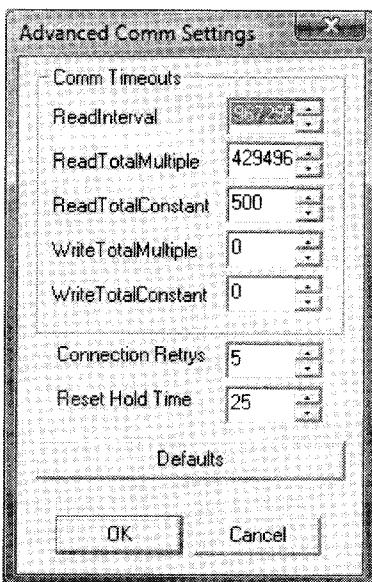
Gambar 3.3 Tampilan ATOM Pro IDE

Anda juga perlu mengatur Preferences dari aplikasi ini yang mengacu pada comPort yang digunakan dan mendeteksi Atom Pro.



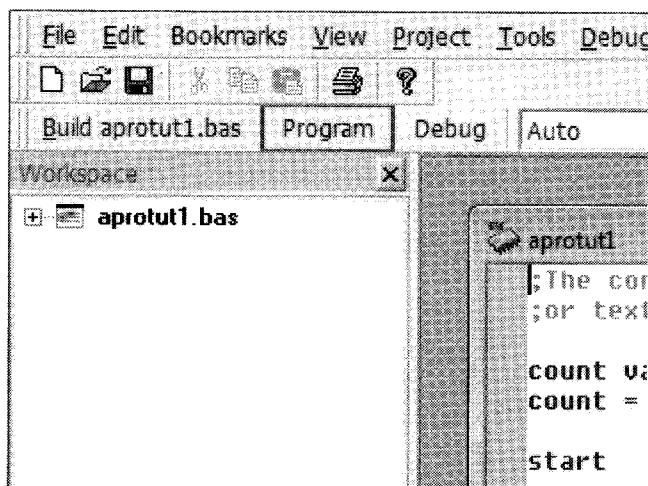
Gambar 3.4 Setting Preferences

E
Untuk hasil yang optimal, atur juga Advanced Comm Setting seperti berikut:



Gambar 3.5 Konfigurasi Communication

4. Lakukan pengisian program dengan mengklik menu Program.



Gambar 3.6 Mengisi program

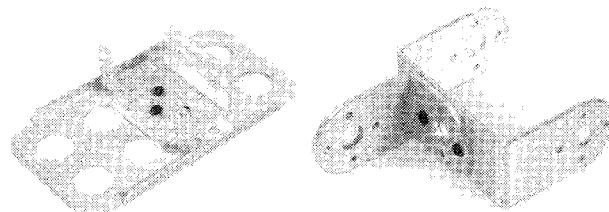
5. Jalankan program. Hasilnya harus seperti berikut:

The screenshot shows a terminal window titled "Output". At the top, there are dropdown menus for "9600", "COM4", "No Parity", "NoFlow Ct", "Echo", and a "Disconnect" button. Below the menu bar, the text "9", "10", "11", "12", "13", and "14" are displayed one by one. At the bottom of the window, there is a toolbar with icons for "Build", "Debug", and four tabs labeled "Terminal1", "Terminal2", "Terminal3", and "Terminal4".

Gambar 3.6 Hasil output

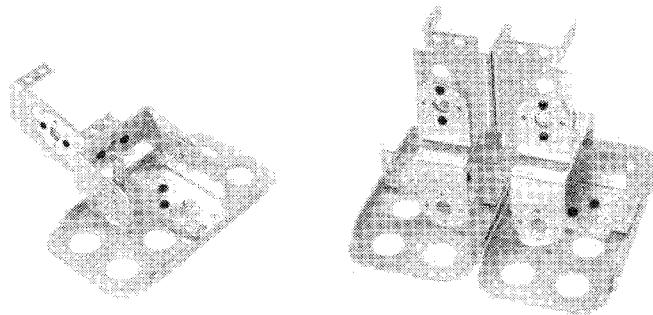
Merakit Robot BRAT

Masalah body robot kadangkala menjadi hambatan bagi pemula. Tetapi, saya ingatkan agar Anda jangan putus asa hanya karena masalah kecil tersebut. Anda dapat merakit body robot BRAT ini menggunakan acrylic atau seng. Anda juga dapat memiliki paket lengkap robot ini dengan harga yang tidak terlalu mahal dibandingkan pengetahuan dan kesenangan yang Anda peroleh dalam merakit dan memprogramnya. Pada intinya, saya ingin menunjukkan bahwa untuk membuat robot berkaki seperti BRAT ini, Anda perlu mengetahui model pemasangan servo dan kalibrasi servo yang memadai agar kaki robot ini dapat bergerak mirip dengan kaki manusia. Oleh karena itu, kita namakan robot seperti ini termasuk robot humanoid. Manual lengkap pemasangan dan konfigurasi robot ini dapat Anda lihat pada situs www.lynxmotion.com. Beberapa komponen yang perlu Anda perhatikan bentuknya untuk jika ingin merakit body robot ini adalah:

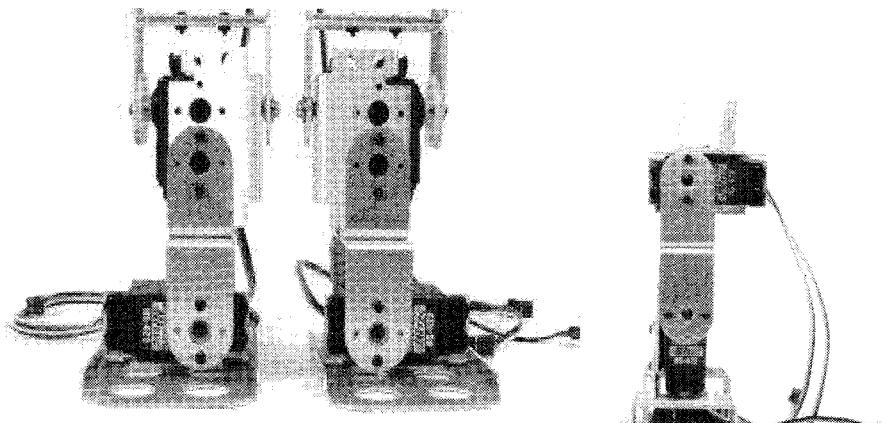


Gambar 3.8 Bagian bawah kaki

Buatlah sambungan telapak kaki dengan betis sehingga bentuknya seperti berikut:



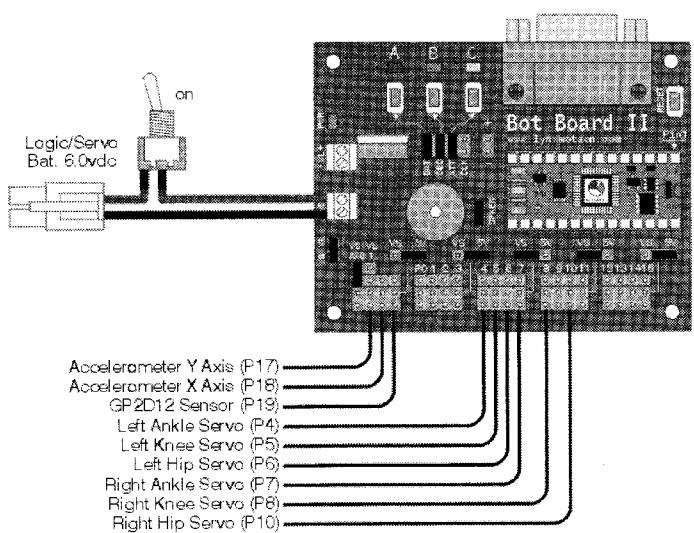
Gambar 3.9 Bagian sambungan telapak kaki sudah jadi



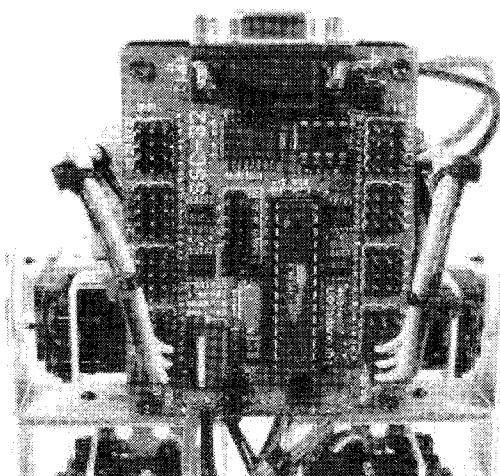
Gambar 3.10 Hasil 6 buah servo Hitec 422 sudah terpasang

Pastikan pemasangan servo dan sensor pada board sesuai, seperti gambar berikut:

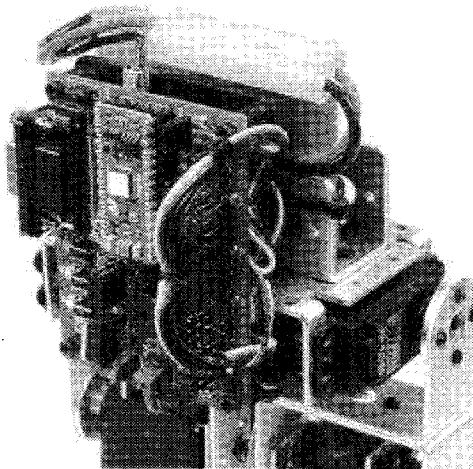
Bab 3: Robot Humanoid BRAT



Gambar 3.11 Susunan sambungan ke servo dan accelerometer

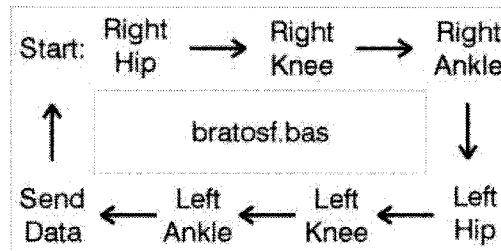


Gambar 3.12 Servo controller terpasang di body



Gambar 3.13 Tampilan bot board II di body robot

Untuk mengkonfigurasi servo agar bekerja optimal, Anda nanti harus menjalankan program **bratosf.bas** terlebih dahulu yang berfungsi sebagai servo offset finder. Prinsipnya sebagai berikut:



Gambar 3.14 Prinsip konfigurasi servo

Bratosf.bas:

```
;System variables ;-----+;
righthip con p10 ;| Servo Offset Finder
rightknee con p8 ;| Written by James Frye
rightankle con p7 ;|
lefthip con p6 ;| Use this program to find
leftknee con p5 ;| the servo offsets for your
leftankle con p4 ;| Lynxmotion BRAT using an Atom Pro|,
;| and a Bot Board!
righthip_start con 0 ;|
rightknee_start con 0 ;-----+;
rightankle_start con 0 ;| How to use
lefthip_start con 0 ;|
leftknee_start con 0 ;| Press A to decrease the servo
leftankle_start con 0 ;| offset by 5us
;
```

Bab 3: Robot Humanoid BRAT

```
rhip var sword      ;| Press C to increase the servo      ;|
rknee var sword    ;| offset by 5us                 ;|
rankle   var sword  ;|                                         ;|
lhip    var sword    ;| Press B to change which servo is |;
lknee   var sword    ;| being manipulated, and to send  |;
lankle  var sword    ;| data back to the terminal     |;
                           ;|                                         ;|
rhip = ((righthip_start / 24) + 1500)
rknee = ((rightknee_start / 24) + 1500)
rankle = ((rightankle_start / 24) + 1500)
lhip = ((lefthip_start / 24) + 1500)
lknee = ((leftknee_start / 24) + 1500)
lankle = ((leftankle_start / 24) + 1500)
currentServo var byte
currentServo = 1
buttonA var bit
buttonB var bit
buttonC var bit

prevA var bit
...
input p12
...

sound 9,[50\3800, 50\4200, 40\4100]

main
gosub pulsemove[rankle,rknee,rhip,lankle,lknee,lhip]

prevA = buttonA
prevB = buttonB
prevC = buttonC

buttonA = in12
buttonB = in13
buttonC = in14

if (buttonA = 0) AND (prevA = 1) then
sound 9,[50\3800]

gosub decreaseOffset

goto main
endif

if (buttonB = 0) AND (prevB = 1) then
sound 9,[50\3600 + (currentServo * 100)]]

currentServo = currentServo + 1
if(currentServo = 7) then
sound 9,[75\3200, 50\3300]
goto output_data

endif
goto main
endif

if (buttonC = 0) AND (prevC = 1) then
```

```

sound 9,[50\4400]

gosub increaseOffset

goto main
endif

goto main

increaseOffset

if(currentServo = 1) AND (rhip < 1600) then
    rhip = rhip + 5

elseif(currentServo = 2) AND (rknee < 1600)
    rknee = rknee + 5

...
elseif(currentServo = 6) AND (lankle < 1600)
    lankle = lankle + 5

else
    sound 9,[150\3500]
endif
return

decreaseOffset

if(currentServo = 1) AND (rhip > 1400) then
    rhip = rhip - 5

...
elseif(currentServo = 6) AND (lankle > 1400)
    lankle = lankle - 5

else
    sound 9,[150\3500]
endif
return

pulsesmove[rankle,rknee,rhip,lankle,lknee,lhip]
pulsout leftankle, (lankle * 2)
pulsout leftknee, (lknee * 2)
pulsout lefthip, (lhip * 2)

pulsout rightankle, (rankle * 2)
pulsout rightknee, (rknee * 2)
pulsout righthip, (rhip * 2)
pause 20

return

output_data

serout s_out,i38400,["righthip_start con ", sdec ((rhip - 1500) * 24),13]
serout s_out,i38400,["rightknee_start con ", sdec ((rknee - 1500) * 24),13]

```

Bab 3: Robot Humanoid BRAT

```
serout s_out,i38400,[ "rightankle_start con ", sdec ((rankle - 1500) * 24),13]

serout s_out,i38400,[ "lefthip_start con ", sdec ((lhip - 1500) * 24),13]
serout s_out,i38400,[ "leftknee_start con ", sdec ((lknee - 1500) * 24),13]
serout s_out,i38400,[ "leftankle_start con ", sdec ((lankle - 1500) * 24),13]

waitloop

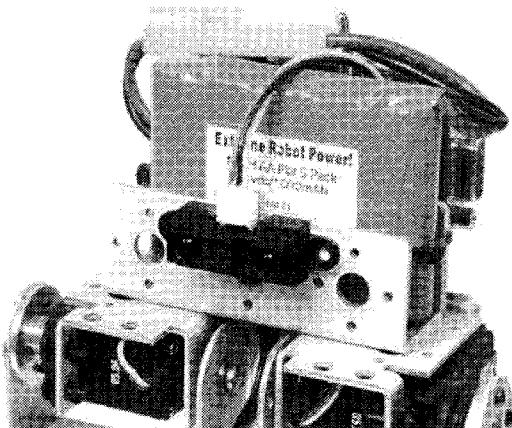
gosub pulsemove[rankle,rknee,rhip,lankle,lknee,lhip]

buttonB = in13

if (buttonB = 0) AND (prevB = 1) then
    currentServo = 1
    sound 9,[200\3800]
    goto main
endif

prevB = buttonB

goto waitloop
```



Gambar 3.15 Pemasangan sensor jarak Sharp GP2D12

Sebagai demo robot penghindar halangan menggunakan sensor jarak infrared Sharp GP2D12, jalankan program Brata2.bas berikut:

Brata2.bas:

```
;Program Sharp GP2D12 pada BRAT
righthip    con p10
rightknee   con p8
rightankle  con p7
lefthip     con p6
leftknee   con p5
leftankle  con p4

true con 1
false con 0
```

```

;calibrate steps per degree.
stepsperdegree fcon 166.6

righthip_start con 0
rightknee_start con 0
rightankle_start con 0

lefthip_start con 0
leftknee_start con 0
leftankle_start con 0

;Interrupt init
ENABLEHSERVO

;Init positions
;Note, movement subroutine arguments are
Rightankle,Rightknee,Righthip,Leftankle,Leftknee,Lefthip,speed
gosub movement [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
;low 1
;low 2
pause 1000

command var byte
xx var byte
ir var word
bat var word
detect var bit
behaviour var byte
filter var byte
temp var word(10)

sound 9,[50\4400]
pause 50
sound 9,[50\3960]
pause 50
sound 9,[50\3400]

behaviour = 0
command = 8
gosub move
pause 2500
detect = 0

main

if (behaviour = 16) then
    behaviour = 0
endif
behaviour = (behaviour +1) ;let's use this to trigger behaviour at times.
gosub readir
adin 16, bat

if (bat < 270) then;battery is getting low
    sound 9,[50\5000,25\4400,50\5000]
endif

if (ir > 250) then
    detect = true
endif
sound 9,[100\(ir * 10)]      ;beep in between steps, higher pitch for closer
obstacles.

```

Bab 3: Robot Humanoid BRAT

```
if detect then
    low 14           ;turn the yellow LED on
    command=2        ;back up one step
    gosub move
    command=11 ;turn left two steps
    for xx=1 to 2
    gosub move
    next
    detect = false
    input 14          ;turn the yellow LED off
else
    low 12           ;turn the red LED on
    command=1
    gosub move
    input 12          ;turn the red LED off
endif

if (behaviour = 5) then
    low 13           ;turn the green LED on
    command = 9
    gosub move
    pause 2000
    sound 9,[150\4400]      ;little whistle while you rest
    pause 50
    sound 9,[50\4400]
    pause 50
    sound 9,[150\3960]
    pause 50
    sound 9,[50\3960]
    pause 50
    sound 9,[250\3400]
    pause 2000
    input 13          ;turn the green LED off
endif

goto main

move:
    if(command = 1) then
Walk Forward
        gosub movement [ 7.0,-20.0,-20.0, -7.0, 20.0, 20.0, 500.0]
        gosub movement [ -7.0,-20.0,-20.0, 7.0, 20.0, 20.0, 500.0]
        gosub movement [ -7.0, 20.0, 20.0, 7.0,-20.0,-20.0, 500.0]
        gosub movement [ 7.0, 20.0, 20.0, -7.0,-20.0,-20.0, 500.0]
        elseif(command = 2)
            ; Headbutt
            ...
            gosub movement [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 500.0]
            gosub movement [ 0.0,-50.0,-90.0, 0.0,-50.0,-90.0, 500.0]
            gosub movement [ 0.0, 32.0,-58.0, 0.0, 32.0,-58.0, 400.0]
            pause 200
            ;gosub movement [ 0.0,-11.0, 7.0, 0.0,-11.0, 7.0, 500.0]
            gosub movement [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 500.0]
            sound 9,[50\4400]
            pause 50
            sound 9,[50\3960]
            pause 50
            sound 9,[50\3960]
        elseif(command = 11)
Turn
        gosub movement [ 0.0,-35.0,-40.0, 0.0, 35.0, 37.0,500.0]
        gosub movement [ 0.0, 35.0, 37.0, 0.0,-35.0,-40.0,500.0]
        gosub movement [-14.0, 35.0, 37.0, 20.0,-35.0,-40.0,500.0]
        gosub movement [-14.0, 35.0, 37.0, 20.0, 35.0, 37.0,500.0]
```

```

gosub movement [ 20.0, 35.0, 37.0,-14.0, 35.0, 37.0,500.0]
gosub movement [ 20.0,-35.0,-40.0,-14.0, 35.0, 37.0,500.0]
...
;

elseif(command = 9)
Rest Position
    gosub movement [ 0.0, 45.0, 45.0, 0.0, 45.0, 45.0, 500.0]
endif
return

;Should never need to edit anything below this line. Add user subroutines
above this and below main.
lefthippos var float
leftkneepos var float
leftanklepos var float
righthippos var float
...
rhspeed var float
rkspeed var float
raspeed var float
speed var float
longestmove var float
;movement
[lefthippos, leftkneepos, leftanklepos, righthippos, rightkneepos, rightanklepos,
speed]
movement
[rightanklepos, rightkneepos, righthippos, leftanklepos, leftkneepos, lefthippos,
speed]
if(speed<>0.0)then
    gosub getlongest[lefthippos-last_lefthippos, |
        leftkneepos-last_leftkneepos, |
        leftanklepos-last_leftanklepos, |
        righthippos-last_righthippos, |
        rightkneepos-last_rightkneepos, |
        rightanklepos-last_rightanklepos], longestmove
    speed = ((longestmove*stepsperdegree)/(speed/20.0))
    gosub getspeed[lefthippos, last_lefthippos, longestmove, speed], lhspeed
    gosub getspeed[leftkneepos, last_leftkneepos, longestmove, speed], lkspeed
    gosub
    getspeed[leftanklepos, last_leftanklepos, longestmove, speed], laspeed
    gosub getspeed[righthippos, last_righthippos, longestmove, speed], rhspeed
    gosub
    getspeed[rightkneepos, last_rightkneepos, longestmove, speed], rkspeed
    gosub
    getspeed[rightanklepos, last_rightanklepos, longestmove, speed], raspeed
else
    lhspeed=0.0;
    lkspeed=0.0;
    laspeed=0.0;
    rhspeed=0.0;
    rkspeed=0.0;
    raspeed=0.0;
endif
hservo [lefthip\TOINT (-lefthippos*stepsperdegree) + lefthip_start\TOINT
lhspeed, |
        righthip\TOINT (righthippos*stepsperdegree) + righthip_start\TOINT
rhspeed, |
        leftknee\TOINT (-leftkneepos*stepsperdegree) + leftknee_start\TOINT
lkspeed, |

```

Bab 3: Robot Humanoid BRAT

```
    rightknee\TOINT (rightkneepos*stepsperdegree) +
rightknee_start\TOINT rkspeed, |
    leftankle\TOINT (-leftanklepos*stepsperdegree) +
leftankle_start\TOINT laspeed, |
    rightankle\TOINT (rightanklepos*stepsperdegree) +
rightankle_start\TOINT raspeed]
;hservowait [lefthip, righthip, leftknee, rightknee, leftankle, rightankle]

idle var byte
finished var byte
junk var word
sensorloop
    finished = true
    gethservo lefthip,junk,idle
    if(NOT idle)then
        finished=false
    endif
    gethservo righthip,junk,idle
    if(NOT idle)then
        finished=false
    endif
    gethservo leftknee,junk,idle
    if(NOT idle)then
        finished=false
    endif
    gethservo rightknee,junk,idle
    ...
;adin 0,ir
;if (ir > 210) then
; detect = true
;endif

    if(NOT finished)then sensorloop

    last_lefthippos = lefthippos
    last_leftkneepos = leftkneepos
    last_leftanklepos = leftanklepos
    last_righthippos = righthippos
    last_rightkneepos = rightkneepos
    last_rightanklepos = rightanklepos
    return

one var float
two var float
three var float
four var float
five var float
six var float
getlongest[one,two,three,four,five,six]
    if(one<0.0)then
        one=-1.0*one
    endif
    if(two<0.0)then
        two=-1.0*two
    endif
    if(three<0.0)then
        three=-1.0*three
    endif
    if(four<0.0)then
        four=-1.0*four
    endif
    ...

```

```

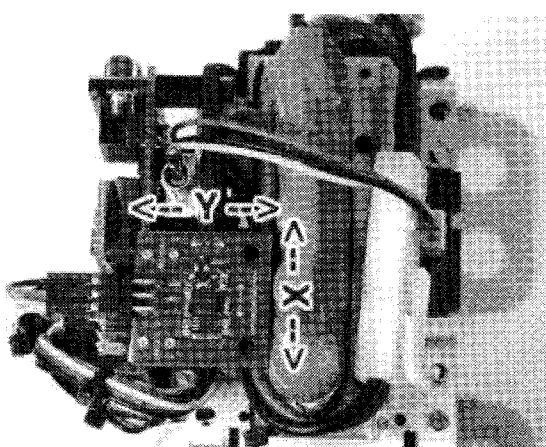
for filter = 0 to 9
    adin 19, temp(filter)
next
ir = 0
for filter = 0 to 9
    ir = ir + temp(filter)
next
ir = ir / 10
;sound 9,[10\4000]
;serout s_out,i38400,[dec ir ,13]
return

```

Jalankan program di atas dan lihat hasilnya. Demo videonya dapat dilihat di CD pelengkap buku ini. Untuk mencoba penerapan sensor accelerometer pada robot, pasangkan sensor tersebut di robot BRAT dengan konfigurasi seperti tabel di bawah:

Tabel 1. Koneksi Bot Board dengan accelerometer

Bot Board I/O	Koneksi
P17 (AX1)	Accelerometer Sensor - Y Axis (Used for Front-to-back tilt)
P18 (AX2)	Accelerometer Sensor - X Axis (Used for Side-to-side tilt)



Gambar 3.16 Pemasangan accelerometer 2 axis

Bab 3: Robot Humanoid BRAT

Buatlah program seperti Brata4.bas di bawah ini. Anda pasti terpana dan terkejut dengan gerakannya yang seperti manusia:

```
Brata4.bas:
;Program Accelerometer pada BRAT
;Robot bisa bangun dan berdiri
righthip    con p10
rightknee   con p8
rightankle  con p7
lefthip     con p6
leftknee   con p5
leftankle  con p4

true con 1
false con 0

;calibrate steps per degree.
stepsperdegree fcon 166.6
..


;Interrupt init
ENABLEHSERVO

;Init positions
;Note, movement subroutine arguments are
Rightankle,Rightknee,Righthip,Leftankle,Leftknee,Lefthip,speed
gosub movement [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
;low 1
;low 2
pause 1000

command var byte
xx var word
ir var word
lastir var word
AMx var word
AMy var word
prevAMx var word
prevAMy var word
bat var word
detect var bit
behaviour var byte
filter var byte
temp var word(10)

main

if (behaviour = 16) then
    behaviour = 0
endif

adin 16, bat
gosub readAMx
gosub readAMy
gosub readir

;Accelerometer readings
;If x is high, then the robot it on its left side
;If x is low, then the robot it on its right side
```

na

```
;If y is high, then the robot it on its front
;If y is low, then the robot it on its back

if (AMx > 550) OR (AMx < 370) OR (AMy > 570) OR (AMy < 380) then
    pause 600

    gosub readAMx
    if (AMx > 580) then ;rollover!
        command = 14
        gosub move

        elseif (AMx < 370) ;rollover!
            command = 13
            gosub move
        endif

    pause 200
    gosub readAMY
    if (AMy > 600) then      ;you're flat on your face! get up!
        command = 7
        gosub move

        elseif (AMy < 390)      ;then you're staring at the sky! get up!
            command = 8
            gosub move
        endif

    endif

    if (bat < 270) then;battery is getting low
        sound 9,[50\5000,25\4400,50\5000]
    endif

    if (ir > 250) then
        detect = true
    endif
    sound 9,[100\(ir * 10)]      ;beep in between steps, higher pitch for closer
    obstacles.

if detect then
    low 14                  ;turn the yellow LED on
    command=2                ;back up one step
    gosub move
    command=11 ;turn left two steps
    for xx=1 to 2
    gosub move
    next
    detect = false
    input 14                 ;turn the yellow LED off
else
    low 12                  ;turn the red LED on
    command=1
    gosub move
    input 12                 ;turn the red LED off
    behaviour = (behaviour +1);let's use this to trigger behaviour at times.
endif

if (behaviour = 5) then
    gosub restsequence
endif
```

Bab 3: Robot Humanoid BRAT

```
goto main

move:
    if(command = 1) then
        Walk Forward
            gosub movement [ 7.0,-20.0,-20.0, -7.0, 20.0, 20.0, 500.0]
            gosub movement [ -7.0,-20.0,-20.0, 7.0, 20.0, 20.0, 500.0]
            gosub movement [ -7.0, 20.0, 20.0, 7.0,-20.0,-20.0, 500.0]
            gosub movement [ 7.0, 20.0, 20.0, -7.0,-20.0,-20.0, 500.0]
        elseif(command = 2)
        ;;
    Walk Backwards
        gosub movement [ -7.0,-20.0,-20.0, 7.0, 20.0, 20.0, 500.0]
        gosub movement [ 7.0,-20.0,-20.0, -7.0, 20.0, 20.0, 500.0]
        gosub movement [ 7.0, 20.0, 20.0, -7.0,-20.0,-20.0, 500.0]
        gosub movement [ -7.0, 20.0, 20.0, 7.0,-20.0,-20.0, 500.0]
    elseif(command = 3)
        ;;
        gosub movement [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 500.0]
        sound 9,[50\4400]
        pause 50
        sound 9,[50\4400]
        pause 50
        sound 9,[50\4400]
    elseif(command = 6)
    ;;
Headbutt
    gosub movement [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 500.0]
    gosub movement [ 0.0,-50.0,-90.0, 0.0,-50.0,-90.0, 500.0]
    gosub movement [ 0.0, 32.0,-58.0, 0.0, 32.0,-58.0, 400.0]
    pause 200
    ;gosub movement [ 0.0,-11.0, 7.0, 0.0,-11.0, 7.0, 500.0]
    gosub movement [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 500.0]
    sound 9,[50\4400]
    pause 50
    sound 9,[50\3960]
    pause 50
    sound 9,[50\3960]
    elseif(command = 11)
    ;;
Turn Left
    gosub movement [ 0.0,-35.0,-40.0, 0.0, 35.0, 37.0,500.0]
    gosub movement [ 0.0, 35.0, 37.0, 0.0,-35.0,-40.0,500.0]
    gosub movement [-14.0, 35.0, 37.0, 20.0,-35.0,-40.0,500.0]
    gosub movement [-14.0, 35.0, 37.0, 20.0, 35.0, 37.0,500.0]
    gosub movement [ 20.0, 35.0, 37.0,-14.0, 35.0, 37.0,500.0]
    gosub movement [ 20.0,-35.0,-40.0,-14.0, 35.0, 37.0,500.0]
    ;;

elseif(command = 15)
Surprise!
    gosub movement [ 0.0,-45.0,-45.0, 0.0,-45.0,-45.0, 350.0]
    gosub movement [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 500.0]
    ;;
elseif(command = 9)
Rest Position
    gosub movement [ 0.0, 45.0, 45.0, 0.0, 45.0, 45.0, 500.0]
    endif
return

;Should never need to edit anything below this line. Add user subroutines
above this and below main.
lefthippos var float
```

```

leftkneepos var float
leftanklepos var float
righthippos var float
...
laspeed var float
rhspeed var float
rkspeed var float
raspeed var float
speed var float
longestmove var float
;movement
[lefthippos, leftkneepos, leftanklepos, righthippos, rightkneepos, rightanklepos,
speed]
movement
[rightanklepos, rightkneepos, righthippos, leftanklepos, leftkneepos, lefthippos,
speed]
if(speed<>0.0)then
    gosub getlongest[lefthippos-last_lefthippos, |
        leftkneepos-last_leftkneepos, |
        leftanklepos-last_leftanklepos, |
        righthippos-last_righthippos, |
        rightkneepos-last_rightkneepos, |
        rightanklepos-last_rightanklepos], longestmove
    speed = ((longestmove*stepsperdegree)/(speed/20.0))
    gosub getspeed[lefthippos,last_lefthippos,longestmove,speed], lhspeed
    gosub getspeed[leftkneepos,last_leftkneepos,longestmove,speed], lkspeed
    gosub
    getspeed[leftanklepos,last_leftanklepos,longestmove,speed], laspeed
    gosub getspeed[righthippos,last_righthippos,longestmove,speed], rhspeed
    gosub
    getspeed[rightkneepos,last_rightkneepos,longestmove,speed], rkspeed
    gosub
    getspeed[rightanklepos,last_rightanklepos,longestmove,speed], raspeed
else
    lhspeed=0.0;
    lkspeed=0.0;
    laspeed=0.0;
    rhspeed=0.0;
    rkspeed=0.0;
    raspeed=0.0;
endif
hservo [lefthip\TOINT (-lefthippos*stepsperdegree) + lefthip_start\TOINT
lhspeed, |
    righthip\TOINT (righthippos*stepsperdegree) + righthip_start\TOINT
rhspeed, |
    leftknee\TOINT (-leftkneepos*stepsperdegree) + leftknee_start\TOINT
lkspeed, |
    rightknee\TOINT (rightkneepos*stepsperdegree) +
rightknee_start\TOINT rkspeed, |
    leftankle\TOINT (-leftanklepos*stepsperdegree) +
leftankle_start\TOINT laspeed, |
    rightankle\TOINT (rightanklepos*stepsperdegree) +
rightankle_start\TOINT raspeed]
;hservowait [lefthip,righthip,leftknee,rightknee,leftankle,rightankle]

idle var byte
finished var byte
junk var word
sensorloop
    finished = true
    gethservo lefthip,junk,idle
    if(NOT idle)then
        finished=false

```

Bab 3: Robot Humanoid BRAT

```
        endif
        gethservo righthip,junk,idle
        if(NOT idle)then
            finished=false
        endif
        gethservo leftknee,junk,idle
        if(NOT idle)then
            finished=false
        endif
        gethservo rightknee,junk,idle
        if(NOT idle)then
            finished=false
        endif
        gethservo leftankle,junk,idle
        if(NOT idle)then
            finished=false
        endif
        ;add sensor handling code here

;adin 0,ir
;if (ir > 210) then
;  detect = true
;endif

        if(NOT finished)then sensorloop

        last_lefthippos = lefthippos
        last_leftkneepos = leftkneepos
        last_leftanklepos = leftanklepos
        last_righthippos = righthippos
        last_rightkneepos = rightkneepos
        last_rightanklepos = rightanklepos
        return

one var float
two var float
three var float
four var float
five var float
six var float
getlongest[one,two,three,four,five,six]
        if(one<0.0)then
            one=-1.0*one
        endif
        if(two<0.0)then
            two=-1.0*two
        endif
        ...
        if(one<three)then
            one=three
        endif
        if(one<four)then
            one=four
        endif
        if(one<five)then
            one=five
        endif
        if(one<six)then
            one=six
```

```

        endif
        return one

newpos var float
oldpos var float
longest var float
maxval var float
getspeed[newpos,oldpos,longest,maxval]
if(newpos>oldpos)then
    return ((newpos-oldpos)/longest)*maxval
endif
return ((oldpos-newpos)/longest)*maxval
...
for xx = 1 to 500

adin 19,ir
pause 10

if (((lastir - ir) > -500) AND ((lastir - ir) < -200)) then
    command = 15
    gosub move

    sound 9,[70\4400]
    pause 35
    sound 9,[45\3800]
    pause 50
    sound 9,[50\3900]
    pause 35
    sound 9,[50\3800]
    pause 35
    sound 9,[70\4300]
    pause 35
    endif

    lastir = ir
next

sound 9,[150\4400]           ;little whistle while you rest
pause 50
sound 9,[50\4400]
pause 50
sound 9,[150\3960]
pause 50
sound 9,[50\3960]
pause 50
sound 9,[250\3400]
input 13           ,turn the green LED off

for xx = 1 to 500
adin 19,ir
pause 10

if (((lastir - ir) > -500) AND ((lastir - ir) < -200)) then;Surprise!
    command = 15
    gosub move

    sound 9,[70\4400]
    pause 35
    sound 9,[45\3800]
    pause 50
    sound 9,[50\3900]

```

Bab 3: Robot Humanoid BRAT

```
pause 35
sound 9,[50\3800]
pause 35
sound 9,[70\4300]
pause 35
endif

lastir = ir
next

return

readir

for filter = 0 to 9
    adin 19, temp(filter)
next
ir = 0
for filter = 0 to 9
ir = ir + temp(filter)
next
ir = ir / 10
;sound 9,[10\4000]
;serout s_out,i38400,[dec ir ,13]
return

readAMx

for filter = 0 to 9
    adin 18, temp(filter)
next
AMx = 0
for filter = 0 to 9
AMx = AMx + temp(filter)
next
AMx = AMx / 10
;sound 9,[10\4000]
;serout s_out,i38400,[dec ir ,13]
return

readAMy
...
```

Lihatlah gerakan yang ditunjukkan robot tersebut. Tampilan videonya dapat dilihat pada CD pelengkap buku.

Latihan

1. Cobalah jalankan program demo Bot Board II dengan module Basic Atom Pro sebagai berikut:

```

nn var byte

ledA var bit
ledB var bit
ledC var bit

butA var bit
butB var bit
butC var bit

prev_butA      var bit
prev_butB      var bit
prev_butC      var bit
temp      var word

BUTTON_DOWN    con 0
BUTTON_UP      con 1

ledA = 0
ledB = 0
ledC = 0
butA = 0
butB = 0
butC = 0

nn = 0

'Wiggle LEDs
main:

adin 16,temp
serout S_OUT,i57600,[dec6 temp\6," ",13]

if (butA = BUTTON_DOWN) AND (prev_butA = BUTTON_UP) then
    nn = nn + 1
    sound P9,[5\2500]
    low P9
endif
if (butB = BUTTON_DOWN) AND (prev_butB = BUTTON_UP) then
    nn = nn - 1
    sound P9,[10\1000]
    low P9
endif
if (butC = BUTTON_DOWN) AND (prev_butC = BUTTON_UP) then
    nn = 0
    sound P9,[10\4000]
    low P9
endif

ledA = nn.BIT0
ledB = nn.BIT1
ledC = nn.BIT2

if ( nn.BIT0 = 1 ) then
    high p0
    high p8
    high p4
else
    low p0
    low p8
    low p4

```

Bab 3: Robot Humanoid BRAT

```
        endif
        if ( nn.BIT1 = 1 ) then
            high p1
            high p5
        else
            low p1
            low p5
        endif
        if ( nn.BIT2 = 1 ) then
            high p2
            high p10
            high p6
        else
            low p2
            low p10
            low p6
        endif
        if ( nn.BIT3 = 1 ) then
            high p3
            high p7
            high p11
        else
            low p3
            low p7
            low p11
        endif

    pause 50

    prev_butA = butA
    prev_butB = butB
    prev_butC = butC
    gosub button_led_control

    goto main

    'Subroutine to read the buttons and control the LEDs. Button states are
    put in
    'variables butA, butB, and butC. LED states are read from variables
    ledA, ledB,
    'and ledC.
    button_led_control:
    'Make P4-P6 inputs to read buttons. This turns the LEDs off briefly
    input p12
    input p13
    input p14
    butA = IN12
    butB = IN13
    butC = IN14

    'Output LOW to each LED that should be on
    if ledA = 1 then
        low p12
    endif
    if ledB = 1 then
        low p13
    endif
    if ledC = 1 then
        low p14
    endif
    return
```

2. Kembangkanlah robot BRAT ini menjadi robot untuk Kontes Robot Seni Indonesia (KRSI). Kontes ini diadakan setiap tahun. Robot ini harus mampu mendengarkan musik yang dimainkan dan mengikutinya dengan gerakan selembut mungkin, misalnya melakukan tarian Jaipong. Diharapkan Anda menggunakan sensor suara dari Hitec karena point penilaian yang diberikan lebih besar jika hanya menggunakan kabel. Anda perlu menambahkan beberapa servo pada tubuh dan tangan agar gerakan tarian robot tersebut lebih alami.

Bab 4

Lengan Robot 5 Sumbu Gerakan

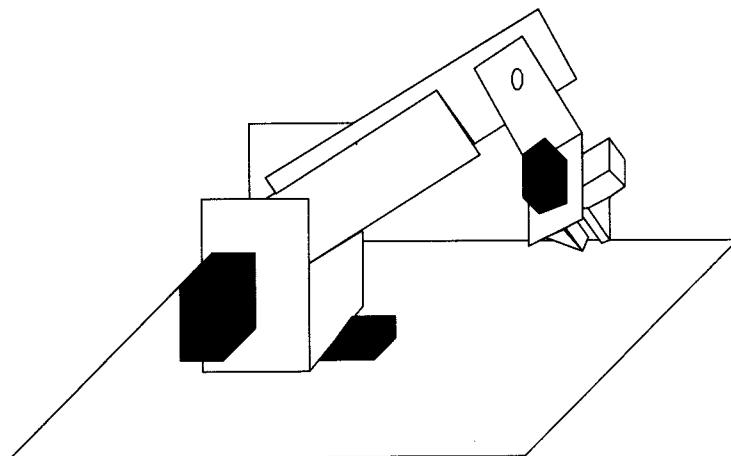
Pendahuluan

Lengan adalah salah satu bagian yang seringkali digunakan pada robot, terutama untuk aplikasi robot humanoid. Penguasaan lengan robot dengan teknologi gripper canggih dan pengontrolan servo untuk mengetahui posisinya sangat penting dalam membuat robot humanoid.

Lengan Robot

Lengan robot digunakan untuk mengambil objek-objek di sekitar robot. Pada kesempatan ini akan dibahas mengenai lengan robot dengan 5 sumbu gerakan, yaitu putaran base, gerakan aktuator naik-turun, gerakan tangan naik-turun, gerakan gripper memutar, dan gerakan gripper menjepit

Bab 4: Lengan Robot 5 Sumbu Gerakan

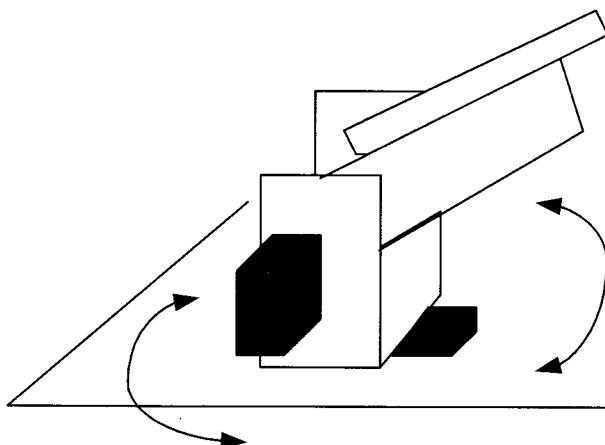


Gambar 4.1 Lengan Robot dengan 5 Sumbu Gerakan

Aplikasi ini dapat dibangun dengan menggunakan:

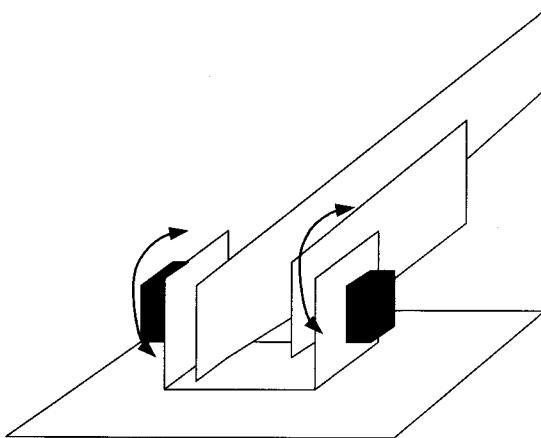
- Robotic Arm Mechanic
- Servo Controller DSR-08
- 3 pcs Motor Servo GWS S04BB
- 3 pcs Motor Servo Hitec HS-422
- 1 pcs Gripper
- 1 pcs Modul USB to serial DU-232 (untuk PC non-serial)
- 1 pcs Modul DST-51 (untuk aplikasi stand alone)

Sebuah motor servo GWS S04BB yang memiliki torsi 13 kg.cm berfungsi untuk mengatur gerakan base dari lengan robot secara memutar.



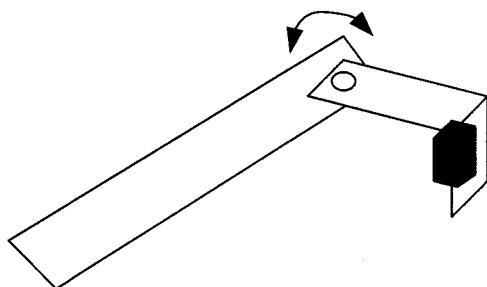
Gambar 4.2 Gerakan Base Memutar

Gerakan aktuator untuk naik dan turun merupakan gerakan yang paling berat karena selain mengangkat objek juga mendapat beban dari mekanik lengan itu sendiri. Untuk itu, digunakan dua buah motor servo GWS S04BB yang diletakkan di sisi kiri dan kanan aktuator.



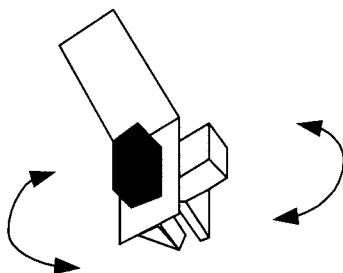
Gambar 4.3 Gerakan Aktuator Naik-Turun

Dengan posisinya yang berhadapan, gerakan kedua motor servo ini harus dilakukan dengan arah yang berlawanan dengan resultan sudut yang sama. Bila motor servo di sisi kiri bergerak ke arah CW, motor servo di sisi kanan harus bergerak ke arah CCW.



Gambar 4.4 Gerakan Tangan Naik-Turun

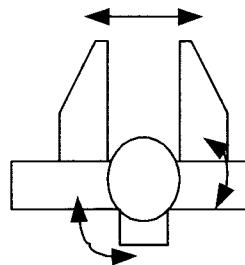
Gerakan tangan tidak membutuhkan torsi sebesar aktuator sehingga dapat digunakan HS-422 yang memiliki torsi 4.1 kg.cm. Beban dari motor servo ini adalah tangan, gripper, dua buah motor servo HS-422, dan objek yang diambil apabila lengan telah memperoleh objek yang dituju.



Gambar 4.5 Gerakan Memutar Gripper

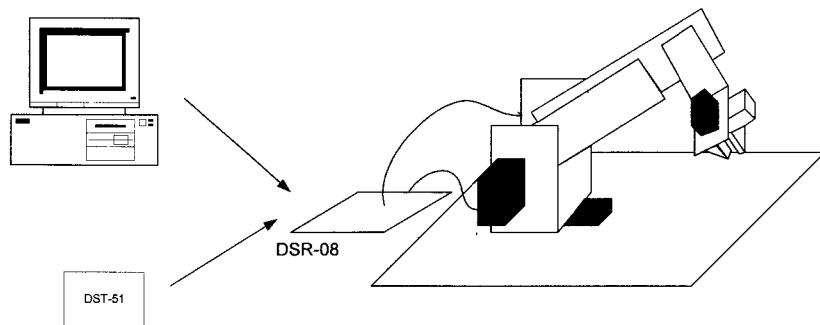
Untuk menyesuaikan posisi dengan objek yang dituju terdapat sebuah motor servo HS-422 yang berfungsi memutar posisi gripper hingga posisinya sesuai dengan kondisi objek,

Setelah objek diperoleh, lengan robot akan melakukan gerakan mencengkeram dengan merapatkan gripper. Hal ini dilakukan dengan memutar sebuah motor servo HS-422 yang terhubung pada bagian ini sehingga gripper bergerak merapat atau merenggang.



Gambar 4.6 Gerakan Gripper Merapat dan Meranggang

Untuk menggerakkan keenam motor servo tersebut dilakukan dengan bantuan Modul DSR-08 yang memiliki kemampuan untuk mengendalikan 8 buah motor servo secara independent. Dengan adanya modul ini, proses gerakan motor servo dapat dilakukan dengan mudah melalui perintah-perintah yang dikirimkan secara serial melalui port UART.

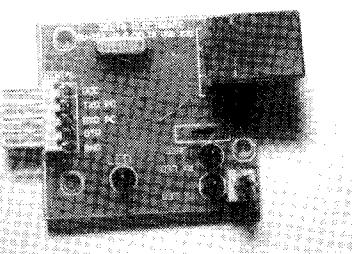


Gambar 4.7 Kendali Lengan Robot

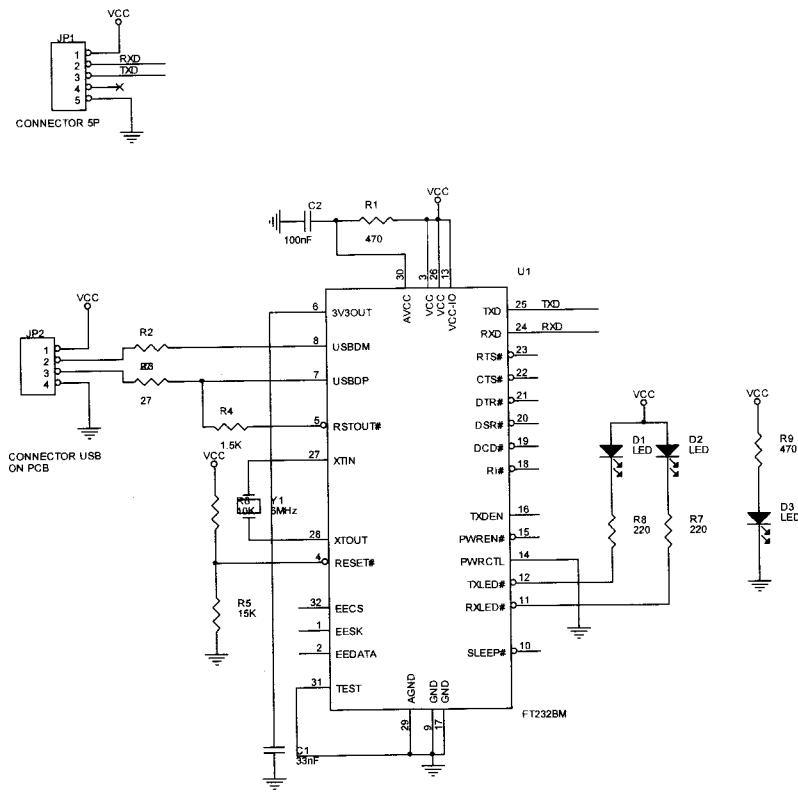
Pada Gambar 4.7 terlihat bahwa proses kendali lengan robot ini dapat dilakukan melalui PC maupun stand alone melalui mikrokontroler DST-51. Modul DST-51 memiliki Port UART sehingga dapat langsung dihubungkan ke UART dari DSR-08.

Untuk aplikasi dengan PC dibutuhkan kabel serial CD-232 yang telah dilengkapi konverter RS232 ke TTL. Untuk PC yang tidak memiliki port serial dapat digunakan Modul DU-232 sebagai konverter dari USB ke serial.

Bab 4: Lengan Robot 5 Sumbu Gerakan



Gambar 4.8 Modul USB to 232 Converter DU-232

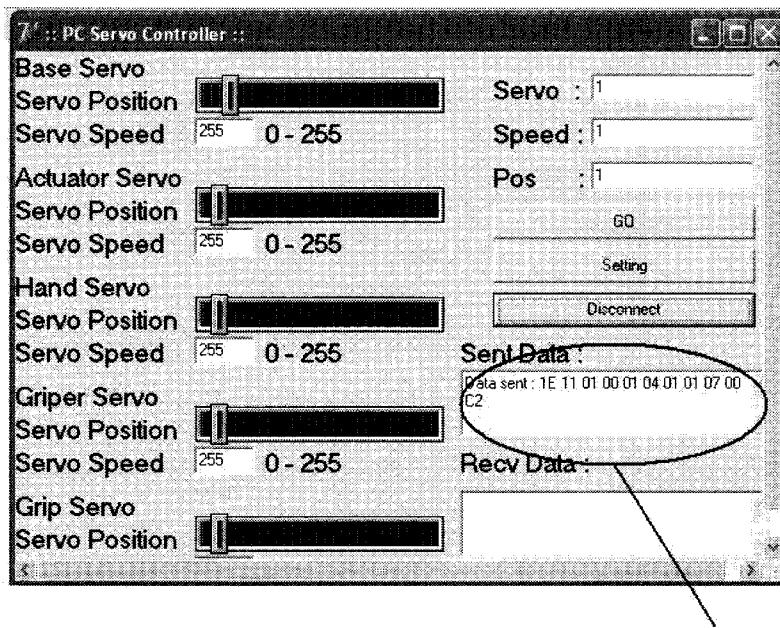


Gambar 4.9 Diagram Skematik Modul USB to 232 Converter DU-232

Aplikasi Penggerak Lengan Robot Menggunakan PC

Seperti yang dijelaskan sebelumnya, lengan robot ini dapat digerakkan menggunakan PC melalui modul DSR-08 yang berfungsi sebagai servo controller. Melalui Port UART modul DSR-08, Anda dapat mengendalikan lengan ini secara visual melalui PC atau notebook.

Dengan Program PC Servo Controller, gerakan setiap sumbu dari lengan robot ini dapat dilakukan hanya dengan menggeser-geser posisi mouse. Selain mempermudah gerakan-gerakan sumbu lengan robot, Anda juga dapat memonitor protokol data yang keluar

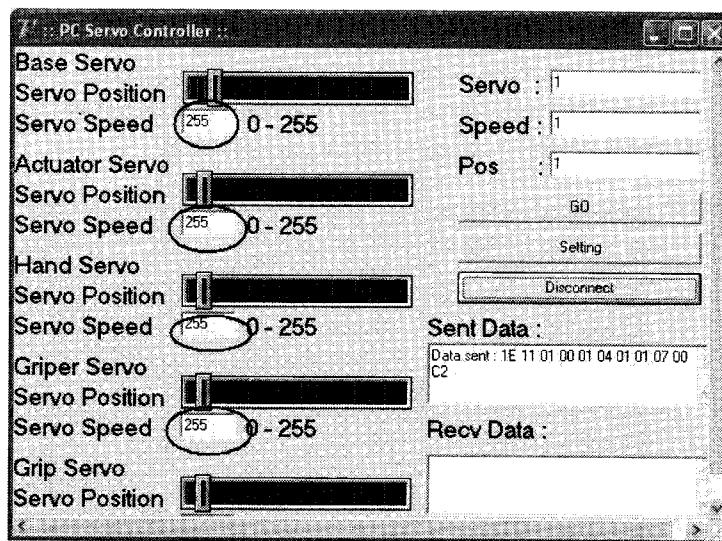


Gambar 4.10 Protokol Data yang dihasilkan PC Servo Controller

Protokol yang ditampilkan tersebut juga dikirimkan ke UART DSR-08 dan menggerakkan sumbu-sumbu lengan robot sehingga data protokol versus kondisi sumbu lengan robot dapat disimpan sebagai catatan yang akan berguna ketika Anda akan membuat aplikasi penggerak lengan otomatis di kemudian hari.

Bab 4: Lengan Robot 5 Sumbu Gerakan

Pengaturan sudut servo dilakukan dengan menggeser-geser posisi sakelar geser dari masing-masing servo yang ada pada tampilan PC Servo Controller. Sedangkan kecepatan diatur dengan memberikan nilai pada field di bawah sakelar geser dengan range 128–255. Pada bagian kanan, Anda juga dapat mengisikan nilai sudut servo secara manual.



Gambar 4.11 Pengaturan Kecepatan Servo

Berikut ini adalah listing program PC Servo Controller yang dibuat dengan menggunakan Delphi.

```
pcservo.pas:  
//Create by: Januar Susanto  
//DELTA ELECTRONIC 2010  
unit Unit1;  
interface  
  
uses  
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
  Dialogs, StdCtrls, CPort, ComCtrls, VrSpinner, VrControls, VrGauge,  
  VrSlider;  
  
type  
  TForm1 = class(TForm)  
    ComPort1: TComPort;  
    ButtonSetting: TButton;  
    ButtonConnect: TButton;  
    VrSliderBase: TVrSlider;  
    Label1: TLabel;  
    Label2: TLabel;
```

```

Label3: TLabel;
BaseSpeedEdit: TEdit;
Label4: TLabel;
VrSliderActuator: TVrSlider;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
ActuatorSpeedEdit: TEdit;
Label8: TLabel;
SentMemo: TMemo;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
VrSliderHand: TVrSlider;
SpeedHandEdit: TEdit;
Label13: TLabel;
Label14: TLabel;
Label15: TLabel;
Label16: TLabel;
VrSliderGriper: TVrSlider;
GriperSpeedEdit: TEdit;
Label17: TLabel;
Label18: TLabel;
Label19: TLabel;
Label20: TLabel;
VrSliderGrip: TVrSlider;
GripSpeedEdit: TEdit;
ServoNum: TEdit;
Label21: TLabel;
SpeedEdit: TEdit;
Label22: TLabel;
Edit1: TEdit;
Label23: TLabel;
Button1: TButton;
Label24: TLabel;
Label25: TLabel;
RecvMemo: TMemo;
procedure ButtonSettingClick(Sender: TObject);
procedure ButtonConnectClick(Sender: TObject);
procedure VrSliderBaseMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure BaseSpeedEditChange(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure ActuatorSpeedEditChange(Sender: TObject);
procedure VrSliderActuatorMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
procedure SpeedHandEditChange(Sender: TObject);
procedure GriperSpeedEditChange(Sender: TObject);
procedure GripSpeedEditChange(Sender: TObject);
procedure VrSliderHandMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure VrSliderGriperMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure VrSliderGripMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Button1Click(Sender: TObject);
procedure ComPort1RxChar(Sender: TObject; Count: Integer);
private
  { Private declarations }
public
  { Public declarations }
end;

```

Bab 4: Lengan Robot 5 Sumbu Gerakan

```
var
  Form1: TForm1;
  posisi_servo : string;
  header,DI,DN,SI,SN,L,NU,P: byte;
  KS_B,KS_A,KS_H,KS_GR,KS_G:byte;
  Chk,Cksum: byte;
  byte_count:word;
implementation

{$R *.dfm}

procedure TForm1.ButtonSettingClick(Sender: TObject);
begin
  ComPort1.ShowSetupDialog;
end;

procedure TForm1.ButtonConnectClick(Sender: TObject);
begin
  if ComPort1.Connected then
  begin
    ComPort1.Close;
    ButtonConnect.Caption:='Connect';
  end
  else
  begin
    Try ComPort1.Open; Except
    begin
      exit;
    end;
    end;
    ButtonConnect.Caption:='Disconnect'
  end;
  if ComPort1.Connected then
  begin
    ComPort1.AbortAllAsync;
    ComPort1.ClearBuffer(true,true);
  end;
end;

procedure TForm1.VrSliderBaseMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  PS:byte;
begin
  Chk:=0;
  header:=$1E;
  Chk:=Chk+header;
  DI:=$11;
  Chk:=Chk+DI;
  DN:=$01;
  Chk:=Chk+DN;
  SI:=$00;
  Chk:=Chk+SI;
  SN:=$01;
  Chk:=Chk+SN;
  L:=$04;
  Chk:=Chk+L;
  P:=$01;
  NU:=$01;
  PS:=VrSliderBase.Position;
  Chk:=Chk+P;
  Chk:=Chk+NU;
```

```

Chk:=Chk+PS;
Chk:=Chk+KS_B;
Cksum:=256-Chk;
ComPort1.Write(header,1);
ComPort1.Write(DI,1);
ComPort1.Write(DN,1);
ComPort1.Write(SI,1);
ComPort1.Write(SN,1);
ComPort1.Write(L,1);
ComPort1.Write(P,1);
ComPort1.Write(NU,1);
ComPort1.Write(PS,1);
ComPort1.Write(KS_B,1);
ComPort1.Write(cksum,1);
SentMemo.Lines.Add('Data sent : '+IntToHex(header,2)+' '+IntToHex(DI,2) +
'+IntToHex(DN,2)+' '+IntToHex(SI,2)+' ' +
+IntToHex(SN,2)+' '+IntToHex(L,2)+' '+IntToHex(P,2)+' '+IntToHex(NU,2) +
'+IntToHex(PS,2)+' '+
IntToHex(KS_B,2)+' '+IntToHex(cksum,2))
end;

procedure TForm1.BaseSpeedEditChange(Sender: TObject);
begin
  KS_B:=255-StrToInt(BaseSpeedEdit.Text);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  SentMemo.Clear;
  RecvMemo.Clear;
  KS_B:=255-StrToInt(BaseSpeedEdit.Text);
  KS_A:=255-StrToInt(ActuatorSpeedEdit.Text);
  KS_H:=255-StrToInt(SpeedHandEdit.Text);
  KS_GR:=255-StrToInt(GriperSpeedEdit.Text);
  KS_G:=255-StrToInt(GripSpeedEdit.Text);
end;

procedure TForm1.ActuatorSpeedEditChange(Sender: TObject);
begin
  KS_A:=255-StrToInt(ActuatorSpeedEdit.Text);
end;

procedure TForm1.VrSliderActuatorMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  PS:byte;
  str:string;
  count:word;
  i:word;
begin
  Chk:=0;
  header:=$1E;
  Chk:=Chk+header;
  DI:=$11;
  Chk:=Chk+DI;
  DN:=$01;
  Chk:=Chk+DN;
  SI:=$00;
  Chk:=Chk+SI;
  SN:=$01;
  Chk:=Chk+SN;
  L:=$04;
  Chk:=Chk+L;

```

Bab 4: Lengan Robot 5 Sumbu Gerakan

```
P:=$01;
NU:=$02;
PS:=VrSliderActuator.Position;
Chk:=Chk+P;
Chk:=Chk+NU;
Chk:=Chk+PS;
Chk:=Chk+KS_A;
Cksum:=256-Chk;
ComPort1.Write(header,1);
ComPort1.Write(DI,1);
ComPort1.Write(DN,1);
ComPort1.Write(SI,1);
ComPort1.Write(SN,1);
ComPort1.Write(L,1);
ComPort1.Write(P,1);
ComPort1.Write(NU,1);
ComPort1.Write(PS,1);
ComPort1.Write(KS_A,1);
ComPort1.Write(cksum,1);
SentMemo.Lines.Add('Data sent : '+IntToHex(header,2)+' '+IntToHex(DI,2)+'
```

+IntToHex(DN,2)+' '+IntToHex(SI,2)+'

+IntToHex(SN,2)+' '+IntToHex(L,2)+' '+IntToHex(P,2)+' '+IntToHex(NU,2)+'

+IntToHex(PS,2)+' '+

IntToHex(KS_B,2)+' '+IntToHex(cksum,2))

```
ComPort1.ReadStr(str,count);
end;

procedure TForm1.SpeedHandEditChange(Sender: TObject);
begin
  KS_H:=255-StrToInt(SpeedHandEdit.Text);
end;

procedure TForm1.GriperSpeedEditChange(Sender: TObject);
begin
  KS_GR:=255-StrToInt(GriperSpeedEdit.Text);
end;

procedure TForm1.GripSpeedEditChange(Sender: TObject);
begin
  KS_G:=255-StrToInt(GripSpeedEdit.Text);
end;

procedure TForm1.VrSliderHandMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  PS:byte;
begin
  Chk:=0;
  header:=$1E;
  Chk:=Chk+header;
  DI:=$11;
  Chk:=Chk+DI;
  DN:=$01;
  Chk:=Chk+DN;
  SI:=$00;
  Chk:=Chk+SI;
  SN:=$01;
  Chk:=Chk+SN;
  L:=$04;
  Chk:=Chk+L;
  P:=$01;
  NU:=$04;
  PS:=VrSliderHand.Position;
```

```

Chk:=Chk+P;
Chk:=Chk+NU;
Chk:=Chk+PS;
Chk:=Chk+KS_H;
Cksum:=256-Chk;
ComPort1.Write(header,1);
ComPort1.Write(DI,1);
ComPort1.Write(DN,1);
ComPort1.Write(SI,1);
ComPort1.Write(SN,1);
ComPort1.Write(L,1);
ComPort1.Write(P,1);
ComPort1.Write(NU,1);
ComPort1.Write(PS,1);
ComPort1.Write(KS_H,1);
ComPort1.Write(cksum,1);
SentMemo.Lines.Add('Data sent : '+IntToHex(header,2)+' '+IntToHex(DI,2) +
'+IntToHex(DN,2)+' '+IntToHex(SI,2) +
'+IntToHex(SN,2)+' '+IntToHex(L,2) +
'+IntToHex(P,2) +
'+IntToHex(NU,2) +
'+IntToHex(PS,2) +
'+IntToHex(KS_H,2) +
'+IntToHex(cksum,2))
end;

procedure TForm1.VrSliderGriperMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  PS:byte;
begin
  Chk:=0;
  header:=$1E;
  Chk:=Chk+header;
  DI:=$11;
  Chk:=Chk+DI;
  DN:=$01;
  Chk:=Chk+DN;
  SI:=$00;
  Chk:=Chk+SI;
  SN:=$01;
  Chk:=Chk+SN;
  L:=$04;
  Chk:=Chk+L;
  P:=$01;
  NU:=$05;
  PS:=VrSliderGriper.Position;
  Chk:=Chk+P;
  Chk:=Chk+NU;
  Chk:=Chk+PS;
  Chk:=Chk+KS_GR;
  Cksum:=256-Chk;
  ComPort1.Write(header,1);
  ComPort1.Write(DI,1);
  ComPort1.Write(DN,1);
  ComPort1.Write(SI,1);
  ComPort1.Write(SN,1);
  ComPort1.Write(L,1);
  ComPort1.Write(P,1);
  ComPort1.Write(NU,1);
  ComPort1.Write(PS,1);
  ComPort1.Write(KS_GR,1);
  ComPort1.Write(cksum,1);
  SentMemo.Lines.Add('Data sent : '+IntToHex(header,2) +
'+IntToHex(DI,2) +
'+IntToHex(DN,2) +
'+IntToHex(SI,2) +
'

```

Bab 4: Lengan Robot 5 Sumbu Gerakan

```
+IntToHex(SN,2)+' '+IntToHex(L,2)+' '+IntToHex(P,2)+' '+IntToHex(NU,2)+'
```

```
'+IntToHex(PS,2)+'
```

```
IntToHex(KS_GR,2)+'
```

```
+IntToHex(cksum,2))
```

```
end;
```

```
procedure TForm1.VrSliderGripMouseUp(Sender: TObject; Button: TMouseButton;
```

```
Shift: TShiftState; X, Y: Integer);
```

```
var
```

```
PS:byte;
```

```
begin
```

```
Chk:=0;
```

```
header:=$1E;
```

```
Chk:=Chk+header;
```

```
DI:=$11;
```

```
Chk:=Chk+DI;
```

```
DN:=$01;
```

```
Chk:=Chk+DN;
```

```
SI:=$00;
```

```
Chk:=Chk+SI;
```

```
SN:=$01;
```

```
Chk:=Chk+SN;
```

```
L:=$04;
```

```
Chk:=Chk+L;
```

```
P:=$01;
```

```
NU:=$06;
```

```
PS:=VrSliderGrip.Position;
```

```
Chk:=Chk+P;
```

```
Chk:=Chk+NU;
```

```
Chk:=Chk+PS;
```

```
Chk:=Chk+KS_G;
```

```
Cksum:=256-Chk;
```

```
ComPort1.Write(header,1);
```

```
ComPort1.Write(DI,1);
```

```
ComPort1.Write(DN,1);
```

```
ComPort1.Write(SI,1);
```

```
ComPort1.Write(SN,1);
```

```
ComPort1.Write(L,1);
```

```
ComPort1.Write(P,1);
```

```
ComPort1.Write(NU,1);
```

```
ComPort1.Write(PS,1);
```

```
ComPort1.Write(KS_G,1);
```

```
ComPort1.Write(cksum,1);
```

```
SentMemo.Lines.Add('Data sent : '+IntToHex(header,2)+'
```

```
'+IntToHex(DI,2)+'
```

```
'+IntToHex(DN,2)+'
```

```
'+IntToHex(SI,2)+'
```

```
'+IntToHex(SN,2)+'
```

```
'+IntToHex(L,2)+'
```

```
'+IntToHex(P,2)+'
```

```
'+IntToHex(NU,2)+'
```

```
'+IntToHex(PS,2)+'
```

```
IntToHex(KS_G,2)+'
```

```
+IntToHex(cksum,2))
```

```
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
```

```
PS:byte;
```

```
KS:byte;
```

```
begin
```

```
Chk:=0;
```

```
header:=$1E;
```

```
Chk:=Chk+header;
```

```
DI:=$11;
```

```
Chk:=Chk+DI;
```

```
DN:=$01;
```

```
Chk:=Chk+DN;
```

```
SI:=$00;
```

```
Chk:=Chk+SI;
```

```

SN:=$01;
Chk:=Chk+SN;
L:=$04;
Chk:=Chk+L;
P:=$01;
NU:=StrToInt(ServoNum.Text);
PS:=StrToInt(Edit1.Text);
KS:=255-StrToInt(SpeedEdit.Text);
Chk:=Chk+P;
Chk:=Chk+NU;
Chk:=Chk+PS;
Chk:=Chk+KS;
Cksum:=256-Chk;
ComPort1.Write(header,1);
ComPort1.Write(DI,1);
ComPort1.Write(DN,1);
ComPort1.Write(SI,1);
ComPort1.Write(SN,1);
ComPort1.Write(L,1);
ComPort1.Write(P,1);
ComPort1.Write(NU,1);
ComPort1.Write(PS,1);
ComPort1.Write(KS_G,1);
ComPort1.Write(cksum,1);
SentMemo.Lines.Add('Data sent : '+IntToHex(header,2)+' '+IntToHex(DI,2) +
'+IntToHex(DN,2)+' '+IntToHex(SI,2) +' '
+IntToHex(SN,2)+' '+IntToHex(L,2) +' '+IntToHex(P,2) +' '+IntToHex(NU,2) +
'+IntToHex(PS,2) +' '+
IntToHex(KS_G,2) +' '+IntToHex(cksum,2))
end;

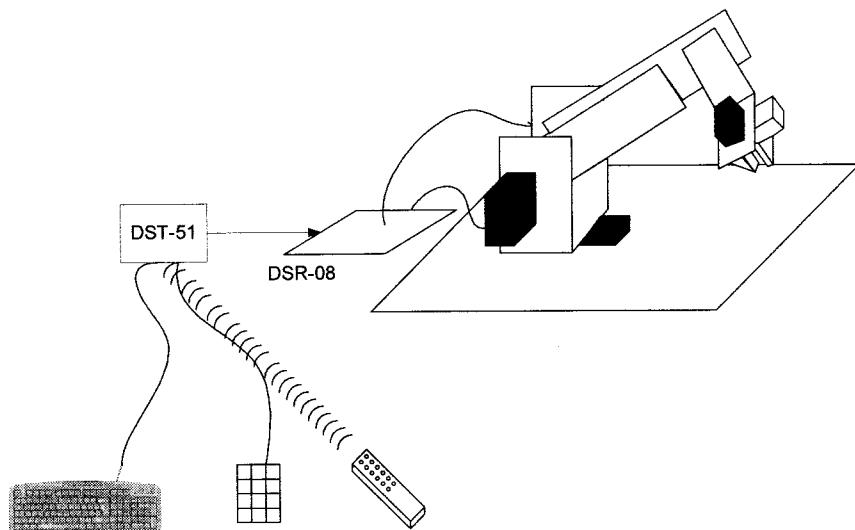
procedure TForm1.ComPort1RxChar(Sender: TObject; Count: Integer);
var
str,hexa:string;
i:word;
begin
  ComPort1.ReadStr(str,count);
  for i:=1 to length(str) do
  begin
    hexa:=hexa+' '+IntToHex(byte(str[i]),2);
  end;
  RecvMemo.Lines.Add('Data recv: '+hexa);
end;
end.

```

Aplikasi Penggerak Lengan Robot Menggunakan Mikrokontroler

Pada aplikasi tertentu, terutama pada robot-robot yang bergerak, penggunaan PC atau notebook biasanya dihindari untuk alasan efisiensi. Pada aplikasi seperti ini, penggunaan mikrokontroler adalah pilihan yang sesuai.

Untuk operasi secara manual, Anda dapat menambahkan tombol, keypad, keyboard, atau remote control pada mikrokontroler untuk menggerakkan sumbu-sumbu lengannya.



Gambar 4.12 Proses Kendali Melalui Mikrokontroler secara Manual

Pada Gambar 4.12 digunakan Modul DST-51 sebagai sistem mikrokontrolernya karena modul ini telah memiliki antarmuka untuk PC keyboard dan keypad sebagai input manual. Dengan menambahkan Modul IR-8510, maka DST-51 juga dapat menerima input dari remote control televisi.

Program Penggerak Lengan Robot Otomatis

Selain aplikasi manual, lengan robot sebetulnya lebih sering digunakan untuk aplikasi otomatis di mana gerakan-gerakan sumbunya telah terprogram sebelumnya dalam memori. Program berikut diawali dengan gerakan base memutar ke kiri. Satu detik kemudian, aktuator turun dan tangan turun, lalu gripper berotasi dan mencengkeram. Satu detik kemudian, tangan naik dan base memutar ke depan kembali.

Perlu dicatat bahwa protokol data pada tabel-tabel program ini tidaklah selalu demikian. Hal ini sangat bergantung pada posisi saat Anda memasang jangkar dari motor servo. Oleh karena itu, sebaiknya gunakan PC Servo Controller untuk menentukan nilai-nilai di setiap tabel servo terlebih dahulu sebelum menggunakan program ini. Contoh, pada TabelBase2 terdapat nilai 01, 01, 10h,80h. Data 01 yang pertama adalah nomor urut DSR-08 sendiri yang selalu 01 pada aplikasi ini (karena hanya ada satu modul DSR-08).

Data 01 yang kedua adalah nomor motor servo yang dikendalikan, yaitu motor servo penggerak Base. Data 10h adalah step sudut yang diperoleh dari byte 8 protokol data yang tampil pada PC Servo Controller. Data 80h atau 128 desimal adalah delay pengatur kecepatan yang diperoleh dari byte 9 protokol data yang tampil pada PC Servo Controller.

armrobo.asm:

```
; PROGRAM LENGAN ROBOT
; Create by: Paulus Andi Nalwan
; DELTA ELECTRONIC 2010

$MOD51
        DSEG
DeviceID      EQU    00h
PerintahKontrolMotor   EQU    01
STX          EQU    1Eh
NomorRobot    EQU    01      ;Nomor urut Robot

        Org     30H

;VARIABEL-VARIABEL
DeviceAddress:  Ds      1      ;Nomor urut robot
TargetType:     Ds      1      ;Jenis Slave
TargetNum:      Ds      1      ;Nomor urut Slave
SourceType:     Ds      1      ;Variabel Master
SourceNum:      Ds      1      ;Variabel nomor urut Master
CheckSumData:   Ds      1      ;Checksum paket data serial
PanjangDataSerial: Ds      1      ;Panjang data pada
;paket serial
Buffer:        Ds      25     ;Buffer data serial
Counter_5mS:    Ds      1

        CSEG
        Org     00
Ljmp     Start

        Org     03h
Reti

        Org     0Bh
Reti
```

Bab 4: Lengan Robot 5 Sumbu Gerakan

```
Org      13h
Reti

Org      23h
Reti

Start:
Lcall   Init_Serial           ;Inisial Serial Port 57600 bps
Mov     PCON, #0h              ;
Mov     TH1, #0FAh             ;

Loop:
Mov     DPTR, #TabelBase1      ;Gerakan Base memutar ke kiri
Lcall   SetPosisiServo        ;
Lcall   Delay_1detik          ;
Mov     DPTR, #TabelActuator1  ;Gerakan Actuator turun
Lcall   SetDualServo          ;
Lcall   Delay_1detik          ;
Mov     DPTR, #TabelHand1       ;Gerakan Hand/Tangan turun
Lcall   SetPosisiServo        ;
Lcall   Delay_1detik          ;
Mov     DPTR, #TabelGripper1   ;Gerakan rotasi gripper
Lcall   SetPosisiServo        ;
Mov     DPTR, #TabelGrip1       ;Gerakan gripper membuka
Lcall   SetPosisiServo        ;
Lcall   Delay_1detik          ;
Lcall   Delay_1detik          ;

Mov     DPTR, #TabelGrip2       ;Gerakan gripper menutup
Lcall   SetPosisiServo        ;
Lcall   Delay_1detik          ;

Mov     DPTR, #TabelHand2       ;Gerakan tangan naik
Lcall   SetPosisiServo        ;
Lcall   Delay_1detik          ;

Mov     DPTR, #TabelBase2       ;Gerakan base memutar ke depan
Lcall   SetPosisiServo        ;
Lcall   Delay_1detik          ;
Ljmp   $                        

;=====
;TABEL
;- Bagian ini berfungsi untuk menentukan sudut-sudut servo dan kecepatan
;- Untuk kecepatan, min = 30 dan max = 128 / 80h
;- Formasi TabelActuator: [Nomor Servo Controller][Sudut servo][kecepatan]
;- Formasi Tabel2 lainnya: [Nomor Servo Controller][Nomor Servo][Sudut
servo][kecepatan]

TabelActuator1:
DB      01,9,80h               ;range sudut actuator 05 - 09
                                ;05 angkat dan 09 turun

TabelBase1:
DB      01,01,15,80h           ;range sudut Base 05 - 15
                                ;15 Counter Clock Wise, 10
;Centre

TabelBase2:
DB      01,01,10,80h
```

```

TabelHand1:           ;range tangan 05 - 15
    DB      01,04,05,80h ;05 turun, 10 centre

TabelHand2:
    DB      01,04,8,80h

TabelGripper1:
    DB      01,05,08,80h ;Range gripper rotation 05
;- 15

TabelGrip1:
    DB      01,06,05,80h ;Range Gripping 05 - 15

TabelGrip2:
    DB      01,06,10,80h

KirimStopMotor:
    Mov     TargetType,#11h
    Mov     TargetNum,#01
    Mov     DeviceAddress,#NomorRobot
    Mov     Buffer,#02
    Mov     Buffer+1,#07
    Mov     Buffer+2,R6
    Ljmp   KirimPaket

SetDualServo:
    Push   DPH
    Push   DPL
    Lcall  SetServo
    Mov    @R0,#02
    Inc    R0

    Mov    A,#00          ;Set Posisi servo
    Movc  A,@A+DPTR
    Mov    @R0,A
    Inc    R0
    Inc    DPTR
    ;

    Mov    A,#00          ;Set delay / kekuatan servo
    Movc  A,@A+DPTR
    Mov    @R0,A
    Inc    R0
    Inc    DPTR
    Lcall  KirimPaket
    Pop    DPL
    Pop    DPH

    Lcall  SetServo
    Mov    @R0,#03
    Inc    R0

    Mov    A,#00          ;Set Posisi servo
    Movc  A,@A+DPTR
    Mov    B,A
    Mov    A,#14
    Clr    C
    Subb  A,B
    Mov    @R0,A
    Inc    R0
    Inc    DPTR
    ;

```

Bab 4: Lengan Robot 5 Sumbu Gerakan

```
Mov    A, #00                      ;Set delay / kekuatan servo
Movc   A, @A+DPTR
Mov    @R0, A
Inc    R0
Inc    DPTR
Lcall  KirimPaket
Ret

SetPosisiServo:
Lcall  SetServo

Mov    A, #00                      ;Set Nomor Servo
Movc   A, @A+DPTR
Mov    @R0, A
Inc    R0
Inc    DPTR

Mov    A, #00                      ;Set Posisi servo
Movc   A, @A+DPTR
Mov    @R0, A
Inc    R0
Inc    DPTR

Mov    A, #00                      ;Set delay / kekuatan servo
Movc   A, @A+DPTR
Mov    @R0, A
Inc    R0
Inc    DPTR

KirimPaket:
Push   DPH
Push   DPL
Lcall  KirimPaketData
Lcall  AmbilDataSerial

Pop    DPL
Pop    DPH
Ret

SetServo:
Mov    TargetType, #11h           ;Target Servo Controller
(DSR-30 / DSR-08)

Mov    A, #00                      ;Set nomor DSR-08
Movc   A, @A+DPTR
Mov    TargetNum, A
Inc    DPTR
Mov    DeviceAddress, #NomorRobot ;Set nomor robot = 01
Mov    R0, #Buffer                ;Set panjang data
Mov    @R0, #04
Inc    R0
Mov    @R0, #PerintahKontrolMotor ;Kirim perintah kontrol motor
Inc    R0
Ret

=====
; Ambil data serial (ACK atau info dari DSR-08)

AmbilDataSerial:
Lcall  Serial_In
Cjne  A, #1Eh, AmbilDataSerial
```

```

Lcall  Serial_In
Lcall  Serial_In
Lcall  Serial_In
Lcall  Serial_In
Lcall  Serial_In
Mov    B,A
Mov    R0,#Buffer
Mov    @R0,A
Inc    R0
LoopDataIn:
Lcall  Serial_In
Mov    @R0,A
Inc    R0
Djnz  B,LoopDataIn
Lcall  Serial_In
Ret

KirimPaketData:
Lcall  AwalPaketMaster

Mov    R0,#Buffer
Mov    A,@R0
Mov    R7,A
Lcall  SerialOutCS

Mov    R0,#Buffer+1
LoopBufferOut:
Mov    A,@R0
Inc    R0
Lcall  SerialOutCS
Djnz  R7,LoopBufferOut
Mov    A,#00
Clr    C
Subb  A,ChecksumData
Lcall  Serial_Out
Ret

AwalPaketMaster:
Mov    A,#STX
Lcall  Serial_Out
Mov    CheckSumData,A

Mov    A,TargetType
Lcall  SerialOutCS

Mov    A,TargetNum
Lcall  SerialOutCS

Mov    A,#DeviceID
Lcall  SerialOutCS

Mov    A,DeviceAddress
Lcall  SerialOutCS
Ret

AwalPaket:
Mov    A,#STX           ;Start
Lcall  Serial_Out
Mov    CheckSumData,A

Mov    A,SourceType

```

Bab 4: Lengan Robot 5 Sumbu Gerakan

```
Lcall SerialOutCS
Mov A,SourceNum
Lcall SerialOutCS

Mov A,#DeviceID
Lcall SerialOutCS

Mov A,DeviceAddress
Lcall SerialOutCS

Ret

SaveBuffer:
Mov @R0,A ; R0
Inc R0
Add A,CheckSumData ;Tambah check sum
Mov CheckSumData,A ;
Ret

SerialOutCS:
Lcall Serial_Out
Add A,ChecksumData
Mov CheckSumData,A
Ret

Init_Serial:
MOV SCON,#52H ; Mode 1 Ren
; MOV TMOD,#20H ; T0 Mode 2, T1 Mode 2
Mov A,TMOD
Anl A,#0FH
Orl A,#20H
Mov TMOD,A
MOV TH1,#0FFH ; 57600
Setb TR1
MOV PCON,#80H ;
RET

Serial_Out:
Clr TI
Mov SBUF,A
TungguKirim:
Jnb TI,TungguKirim
Clr TI
Ret

Serial_In:
Clr RI
TungguTerima:
Jnb RI,TungguTerima
Mov A,SBUF
Clr RI
Ret

KirimPesan_Serial:
Mov A,#00H
Movc A,@A+DPTR
Cjne A,#0FH,KirimTerus
Ret

KirimTerus:
Lcall Serial_Out
Inc DPTR
Ajmp KirimPesan_Serial
```

```

Delay_1detik:
    Mov     Counter_5mS,#0200

Tunggu_1detik:
    Lcall   Delay_5mS
    Djnz   Counter_5mS,Tunggu_1detik
    Ret

Delay_5mS:
    Push    TMOD
    Mov     TMOD,#21H           ;Timer Mode 16 bit counter
    Mov     TH0,#0EDH
    Mov     TL0,#0FFH
    Setb   TR0

Tunggu_5mS:
    Jbc    TF0,Sudah_5mS
    Ljmp   Tunggu_5mS

Sudah_5mS:
    Clr    TR0
    Pop    TMOD
    Ret

CariPerintah1B:
LoopCariPerintah:
    Mov     A,#00H
    Movc   A,@A+DPTR
    Jz    PerintahTidakKetemu
    Clr    C
    Subb   A,B
    Jz    PerintahKetemu
    Inc    DPTR
    Inc    DPTR
    Inc    DPTR
    Ljmp   LoopCariPerintah

PerintahTidakKetemu:
    Setb   C
    Ret

PerintahKetemu:
    Inc    DPTR
    Mov     A,#00H
    Movc   A,@A+DPTR
    Inc    DPTR
    Push   ACC
    Mov     A,#00H
    Movc   A,@A+DPTR
    Mov     DPH,A
    Pop    DPL
    Mov     A,#00H
    Clr    C
    Ret
END

```

Latihan

1. Sebutkan tiga contoh aplikasi lengan robot otomatis pada kebutuhan industri.
2. Tuliskan program yang menampilkan protokol data yang dikirimkan ke lengan robot ke layar LCD M1632 pada Modul DST-51.
3. Tuliskan program pengendali lengan robot dengan menggunakan Remote Sony, di mana Tombol Prog+ dan Prog- adalah pengatur kiri-kanan/turun-naik. Select adalah tombol pemilih servo.

Bab 5

Sensor-Sensor Robot

Pendahuluan

Sensor pada robot layaknya indera pada manusia atau binatang. Bagian ini berfungsi untuk menyampaikan kondisi yang ada di sekitar robot ke bagian otak sehingga robot dapat melakukan tindakan sesuai keadaan tersebut. Sensor biasa terdapat pada robot-robot jenis otomatis yang mengambil keputusan sendiri akan tindakan yang harus dilakukan. Namun demikian, tidak tertutup kemungkinan sensor juga ada pada robot teleoperate.

Rotary Encoder

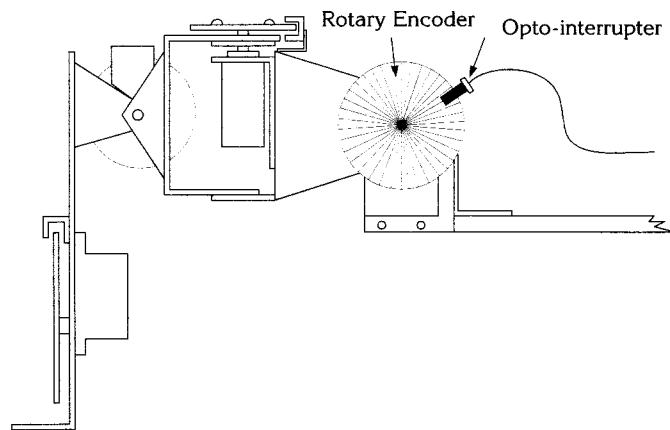
Robot teleoperate biasanya dikendalikan oleh seorang operator secara manual. Untuk mempermudah kerja operator, sensor tetap dapat difungsikan untuk memberitahu operator mengenai keadaan yang ada atau memberikan batasan agar operator tidak melakukan tindakan yang membahayakan robot.

Bab 5: Sensor-Sensor Robot

Sebagai contoh, sensor ultrasonik yang diletakkan di bagian pinggir tubuh robot akan memberitahukan operator tentang jarak robot terhadap halangan. Sensor ini memberitahu operator agar menghentikan atau membelokkan arah robot sehingga tidak membentur halangan. Sensor seperti ini seringkali ditemukan pada kendaraan-kendaraan sebagai peringatan pengemudi akan halangan yang ada di belakang saat mundur kendaraan.

Selain menunggu perintah dari operator untuk mengamankan robot dari tindakan yang membahayakan, otak robot juga bisa mengambil inisiatif dengan memerintahkan motor untuk berhenti bila jarak sudah mencapai titik berbahaya dan operator mengabaikan peringatan.

Rotary encoder berfungsi untuk mengetahui sudut putar sebuah motor atau menghitung kecepatan motor. Motor servo yang pengaturan sudutnya ditentukan oleh lebar pulsa PWM sekalipun seringkali membutuhkan sensor ini. Hal ini karena pengaturan lebar pulsa pada motor servo tidak selalu menunjukkan suatu sudut tertentu. Sumber tegangan motor atau baterai yang turun juga bisa menjadi penyebab perubahan sudut, walaupun pulsa yang dibangkitkan masih memiliki lebar yang sama. Beban motor yang membutuhkan torsi lebih besar juga dapat menjadi faktor penyebab perubahan sudut tersebut. Dengan adanya rotary encoder, otak robot dapat mengetahui apakah gerakan motor telah mencapai sudut yang diharapkan atau belum. Dengan demikian, bila diperlukan lebar pulsa akan diatur kembali agar sudut tersebut diperoleh.

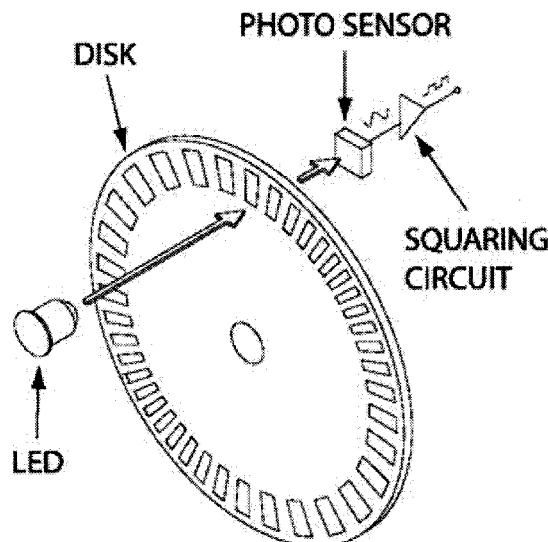


Gambar 5.1 kaki Delta Robo Spider dan Rotary encoder

Sebagai contoh, pada kaki robot laba-laba, saat otak robot menghitung bahwa gerakan kaki masih belum mencapai sudut yang ditentukan—akibat turunnya tegangan baterai atau perubahan medan yang membuat beban motor berubah—maka otak robot akan mengatur kembali lebar pulsa agar diperoleh sudut yang tepat.

Untuk menghitung sudut putar terdapat dua macam teknik encoder, yaitu:

1. Mechanical Encoder, yaitu encoder yang menggunakan efek mekanis di mana terdapat sebuah cincin pada motor yang menimbulkan kontak saat terjadi putaran.
2. Optical Encoder, yaitu encoder yang menggunakan teknik inframerah di mana putaran motor akan memotong pancaran cahaya inframerah ke sensor. Teknik ini lebih sering digunakan karena tidak terjadi efek mekanis yang dapat menimbulkan kesalahan atau rugi-rugi mekanis.

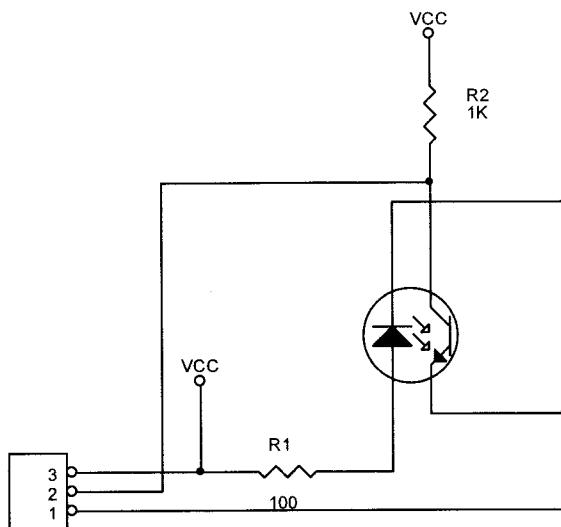


Gambar 5.2 Rotary Encoder

LED inframerah dan photosensor atau phototransistor seringkali dikemas dalam satu paket sebagai opto-interrupter. Gambar 5.3 menunjukkan diagram skematik dari Delta Encoder Kit yang telah dilengkapi dengan piringan encoder.

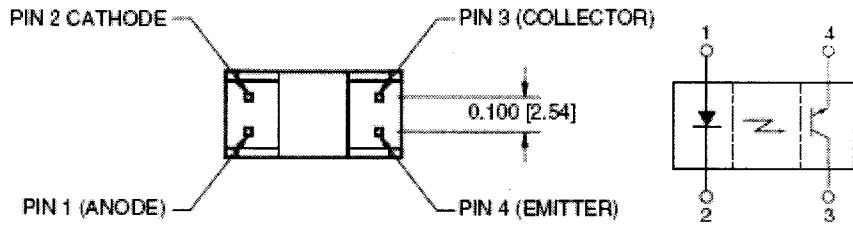
Bab 5: Sensor-Sensor Robot

Gambar LED inframerah dan phototransistor tersebut adalah photo interrupter yang memiliki celah, di mana piringan encoder masuk dan berputar di antaranya. Pada saat lubang piringan tidak melalui celah tersebut, cahaya terhalang dan phototransistor berada dalam kondisi OFF. Tegangan pada keluaran rangkaian ini (kaki 1 dari konektor 3 pin) adalah sebesar VCC. Pada saat lubang piringan berada pada celah, cahaya tidak lagi terhalang dan phototransistor akan berada dalam kondisi ON. Arus mengalir dari VCC ke emitor melalui R 1K yang mengakibatkan keluaran dari rangkaian ini terhubung ke Ground atau tegangan 0 volt. Putaran dari piringan encoder akan membuat keluaran rangkaian ini berubah-ubah dari logika 0 ke 1, kembali lagi ke 0, dan seterusnya.

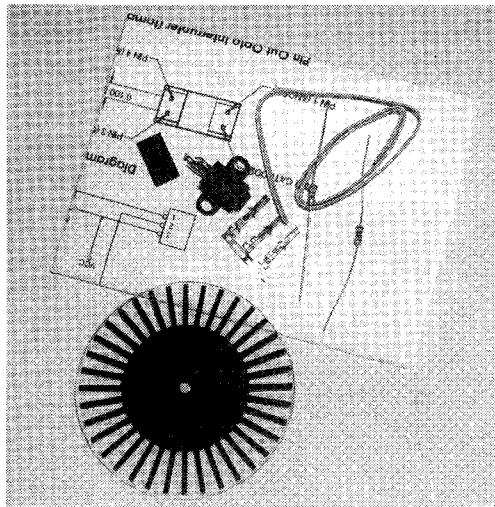


Gambar 5.3 Diagram Skematik Delta Encoder Kit

Opto interrupter yang digunakan adalah tipe MOC703CY dengan konfigurasi kaki seperti pada Gambar 5.4



Gambar 5.4 Opto interrupter tampak bawah



Gambar 5.5 Delta Encoder Kit

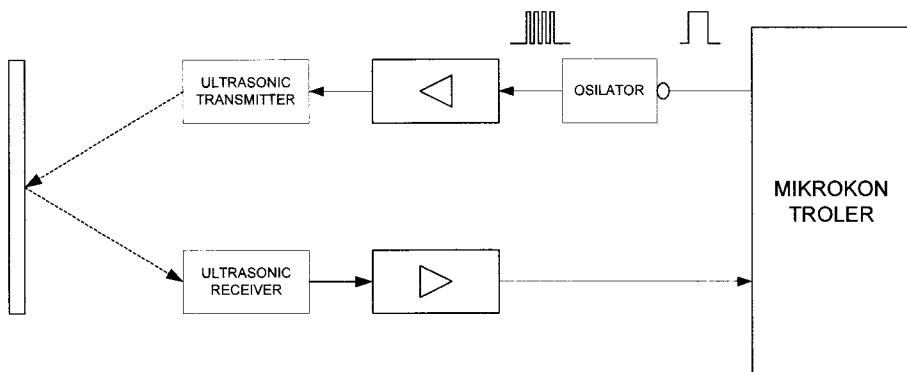
Sensor Jarak Ultrasonik

Pengukuran jarak adalah salah satu elemen penting pada sebuah robot yang sedang bergerak. Hal ini dibutuhkan oleh robot untuk mengetahui posisi robot terhadap objek-objek tertentu. Dalam jarak antara 3 cm hingga 3 meter, ultrasonik adalah media yang sesuai. Contoh penerapannya untuk lengan robot adalah untuk menentukan kapan lengan akan menjulur saat mendeteksi objek di depannya

Bab 5: Sensor-Sensor Robot

Proses pengukuran dilakukan dengan menembakkan sinyal ultrasonik dan menghitung kapan sinyal tersebut diterima kembali oleh sensor. Sensor ultrasonik yang sering digunakan di pasaran adalah sensor yang memiliki respon frekuensi 40 kHz. Oleh karena itu, untuk memancarkan sinyal dengan respon maksimum, dibutuhkan gelombang dengan frekuensi 40 kHz yang dibangkitkan dengan osilator.

Pada Gambar 5.6, tampak bahwa osilator dibangkitkan oleh trigger dari mikrokontroler dan dikuatkan oleh bagian penguat sebelum dipancarkan oleh pemancar ultrasonik. Sinyal ultrasonik akan terpancar setelah getaran ke-8 dari osilator dan dipantulkan oleh objek. Mikrokontroler akan mulai melakukan perhitungan setelah getaran ke-8 dari osilator dilakukan.



Gambar 5.6 Sistem pengukuran jarak dengan ultrasonic

Penerima ultrasonik akan menerima pantulan dari objek dan mengubahnya menjadi getaran-getaran listrik. Namun, getaran tersebut masih terlalu lemah sehingga perlu dikuatkan oleh sebuah penguat sebelum masuk ke mikrokontroler.

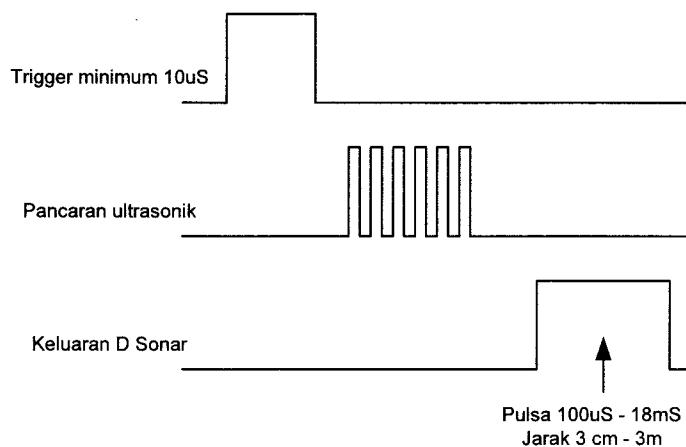
Mikrokontroler akan berhenti melakukan perhitungan saat sinyal ultrasonik diterima kembali. Perhitungan waktu dari saat sinyal ultrasonik pertama kali dipancarkan hingga diterima telah diperoleh. Jarak yang ditempuh oleh rambatan gelombang ultrasonik mulai dari dipancarkan hingga memantul pada objek dan kembali diterima akan diperoleh dengan mengalikan kecepatan rambatan suara dengan waktu yang diperoleh tadi sehingga diperoleh persamaan berikut.

$$s = v \times t$$

$$= 34399,22 \times t$$

Sedangkan jarak antara sensor dengan objek adalah $\frac{1}{2}$ kali jarak rambatan gelombang ultrasonik.

Untuk mempermudah kinerja otak dari robot dalam mengukur jarak, maka proses perhitungan waktu dan konversi ke dalam jarak tadi dilakukan oleh mikrokontroler tersendiri yang dikemas dalam suatu modul yang disebut D-Sonar.



Gambar 5.7 Timing Diagram D-Sonar

Dengan Modul D-Sonar, mikrokontroler yang menjadi otak robot hanya perlu mengirimkan sebuah trigger dan mikrokontroler pada D-Sonar yang akan membangkitkan sinyal ultrasonik serta melakukan perhitungan seperti yang telah dijelaskan sebelumnya.

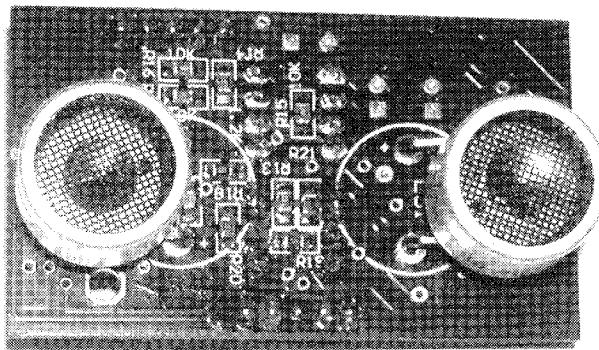
Hasil perhitungan akan dikonversi menjadi sebuah pulsa, di mana lebar pulsa akan menentukan jarak yang diperoleh. Otak robot hanya perlu menghitung lebar pulsa tersebut untuk memperoleh nilai jarak antara sensor dengan objek.

Selain dalam bentuk pulsa, D-Sonar juga dapat melakukan proses perhitungan dengan perintah yang dikirim dengan UART/Serial. Hasil dari

Bab 5: Sensor-Sensor Robot

perhitungan akan dikirim ke bagian port serial D-Sonar dalam bentuk hasil pengukuran jarak.

Perintah-perintah untuk pengukuran jarak dapat dikirim dari mikrokontroler atau PC dengan protokol tertentu yang dijelaskan pada Tabel 1. Mikrokontroler atau PC dapat meminta informasi jarak dalam bentuk Hex maupun ASCII dengan menentukan byte 06 sebagai 01 atau 02.



Gambar 5.8 Modul D-Sonar

Tabel 1 Protokol Permintaan data jarak dari PC/Mikrokontroler

Byte	Nilai	Deskripsi
00	1E	Awal Paket
01	02	ID D-Sonar
02	00 - FF	Nomor urut D-Sonar. Pada kondisi standarnya selalu 01
03	00	ID Pengirim adalah Master/PC
04	00 - FF	Nomor urut Master/PC
05	01	Panjang data
06	01 / 02	01 = Perintah meminta data dalam bentuk periode 02 = Perintah meminta data dalam bentuk jarak
07		Checksum di mana total semua data + checksum = 00

Tabel di atas menunjukkan protokol data yang merupakan informasi jarak dari Modul D-Sonar yang dikirimkan melalui port serialnya ke mikrokontroler atau PC.

Informasi jarak yang dikirimkan akan berupa periode atau jarak tergantung permintaan dari PC atau mikrokontroler. Bila sebelumnya ID tujuan adalah D-Sonar, maka pada paket ini ID tujuan adalah mikrokontroler Master/PC, yaitu 00. Demikian pula pada nomor urut tujuan.

Informasi jarak akan dikirimkan pada byte 7, 8, dan 9. Byte 7 mewakili byte 2 dari data jarak, byte 8 adalah byte 1 data jarak, dan byte 9 adalah byte 0 data jarak dalam bentuk BCD. Informasi periode dikirimkan pada byte 7 dan 8 dalam bentuk Hex

Tabel 2 Protokol data informasi jarak dari D-Sonar ke PC/Mikrokontroler

Byte	Nilai	Deskripsi
00	1E	Awal Paket
01	00	ID Penerima adalah Master/PC
02	00 - FF	Nomor urut Master/PC
03	02	ID D-Sonar
04	00 - FF	Nomor urut D-Sonar. Pada kondisi standardnya selalu 01
05	04/03	Panjang data, 04 untuk jarak dan 03 untuk periode
06	01 / 02	01 = Data jarak dikirim dalam bentuk periode 02 = Data jarak dikirim dalam bentuk jarak
07	dd	Byte 2 dari nilai jarak (BCD) atau byte 1 dari nilai periode (Hex)
08	dd	Byte 1 dari nilai jarak (BCD) atau byte 0 dari nilai periode (Hex)
09	dd	Byte 0 dari nilai jarak dalam BCD
10		Checksum di mana total semua data+ checksum = 00

Dsonar.asm

Program meminta jarak pada D-Sonar melalui serial
 IDDsonar EQU 02

```

  .DATA
BufferDSonar Ds      4
Digit3DSonar Ds      1          ;Variabel penyimpan data ASCII DSonar
Digit2DSonar Ds      1          ;
Digit1DSonar Ds      1          ;

  .CODE
Org    00
Lcall  Init_Serial
Mov    DPTR, #PesananMintaJarak      ;Kirim pesan
                                         ;minta jarak
Lcall  KirimPesanan_Serial          ;
Lcall  AmbilDataDSonar             ;Ambil data D-Sonar

```

Bab 5: Sensor-Sensor Robot

```

Mov      Digit3DSonar,BufferDSonar+1 ;Data-data yang diperoleh dicopy
Mov      Digit2DSonar,BufferDSonar+2 ;ke variabel penyimpan data ASCII
Mov      Digit1DSonar,BufferDSonar+3 ;DSonar
Ljmp    $

AmbilDataDSonar:
BukanAwalPaket:
    Lcall   Serial_In           ;Tunggu awal paket
    Cjne   A,#1Eh,BukanAwalPaket ;;

BukanID:
    Lcall   Serial_In           ;Apakah = 00 / ID Master
    Jnz    BukanID              ;;

    Lcall   Serial_In           ;Ambil nomor urut master

BukanDSonar:
    Lcall   Serial_In           ;Apakah ID D Sonar?
    Cjne   A,#IDDSonar,BukanDSonar ;;
    Lcall   Serial_In           ;Ambil nomor urut D Sonar
    Lcall   Serial_In           ;Ambil panjang data
    Mov    B,A
    Mov    R0,#BufferDSonar

LoopAmbilData:
    Lcall   Serial_In           ;Ambil isi data
    Mov    @R0,A
    Inc    R0
    Djnz   B,LoopAmbilData     ;;
    Lcall   Serial_In
    Ret

PesananJarak:
    DB      1Eh,02,01,00,01,02,0DBh

Init_Serial:
    MOV    SCON,#52H      ; Mode 1 Ren
    MOV    TMOD,#20H      ; T0 Mode 2, T1 Mode 2
    MOV    TH1,#0FDH      ; 9600 Baudrate
    Setb  TR1
    MOV    PCON,#0H        ;
    RET

Serial_Out:
    Clr    TI
    Mov    SBUF,A

TungguKirim:
    Jnb    TI,TungguKirim
    Ret

Serial_In:
    Clr    RI

TungguTerima:
    Jnb    RI,TungguTerima
    Mov    A,SBUF
    Ret

KirimanPesan_Serial:
    Mov    A,#00H

```

```
Movc    A,@A+DPTR
Cjne  A,#0FH,KirimTerus
Ret

KirimTerus:
Acall  Serial_Out
Inc    DPTR
Ajmp  KirimPesan_Serial
```

Modul Suara D-Voice 04

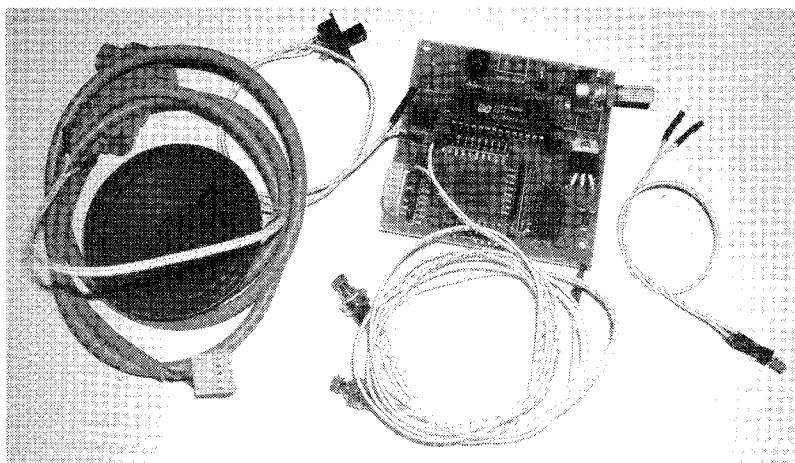
Pada aplikasi-aplikasi tertentu, robot juga dibutuhkan untuk merekam dan memutar ulang suara. Contohnya pada robot yang bertindak sebagai pengintai atau mata-mata. Selain aplikasi tersebut, modul ini juga dapat digunakan sebagai efek-efek suara robot yang direkam terlebih dahulu dan diputar pada kondisi-kondisi tertentu.

Modul D-Voice 04 adalah sebuah subsistem yang mampu merekam dan memutar suara hingga 8 menit. Ia memiliki dua cara dalam melakukan proses merekam dan memutar suara, yaitu secara manual dan dengan protokol data melalui UART port

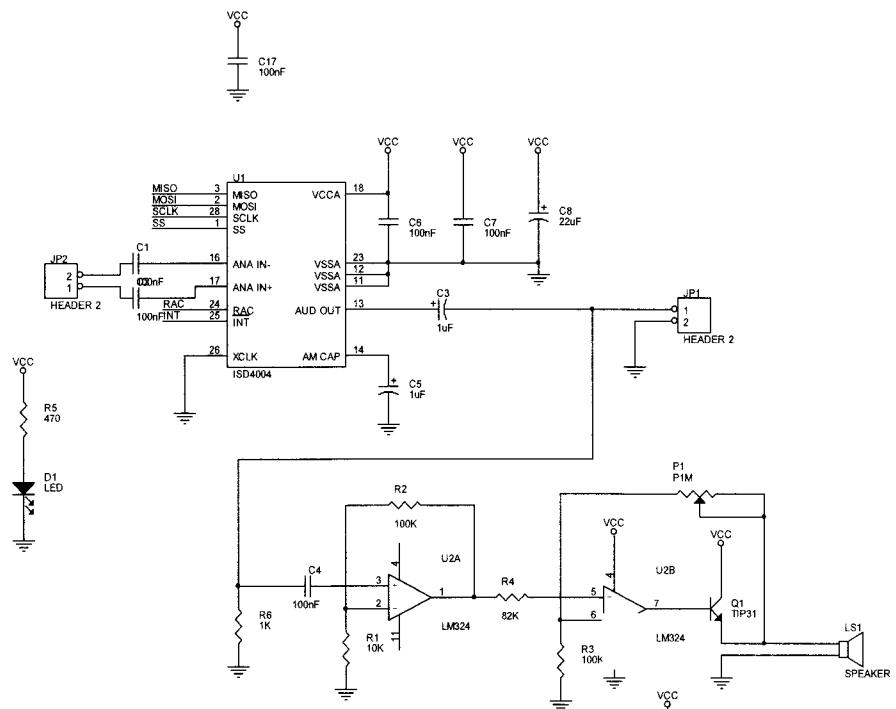
Modul D-voice 04 ini menggunakan IC ISD4004 yang menggunakan antarmuka SPI (Gambar 5.8). Untuk mempermudah pengguna dalam mengakses ISD4004, maka mikrokontroler AT89S51 pada modul ini berfungsi melakukan konversi perintah-perintah dari UART menjadi SPI.

IC Op Amp LM324 berfungsi untuk menguatkan dan mengatur volume suara yang keluar dari ISD4004. Transistor TIP31 dengan arus maksimum 2A akan memberikan penguatan hingga maksimum 10 watt pada keluaran D-Voice sehingga dapat menghasilkan suara yang cukup keras.

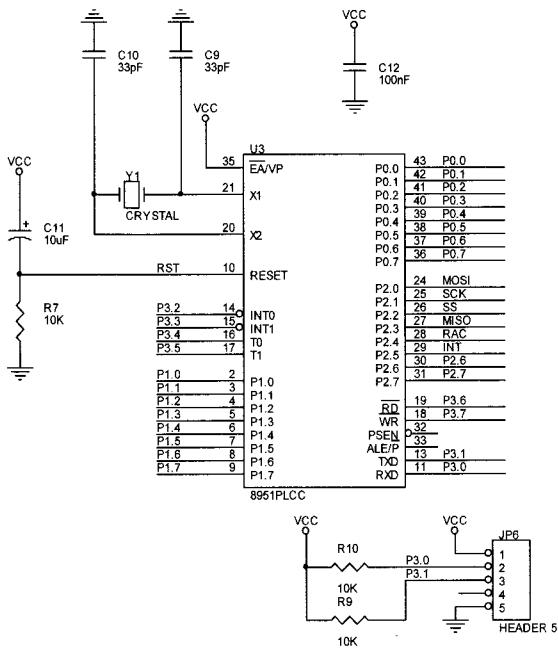
Bab 5: Sensor-Sensor Robot



Gambar 5.9 Modul D-Voice 04



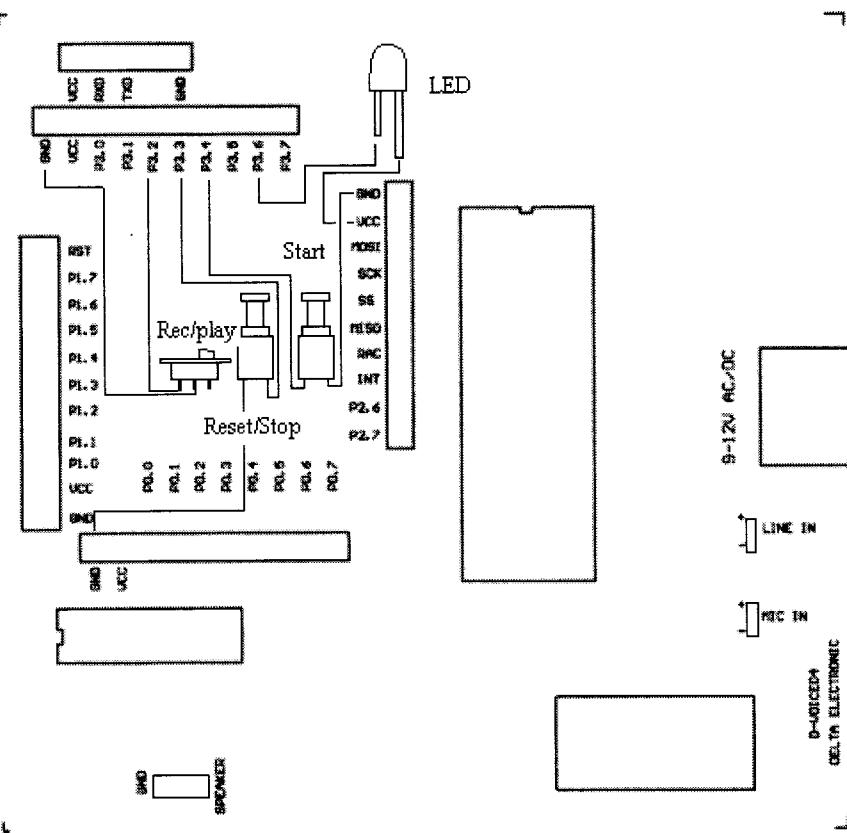
Gambar 5.10a Diagram Skematik D-Voice 04 bagian Suara



Gambar 5.10b Diagram Skematik D-Voice 04 Bagian Kontrol

Proses D-Voice secara Manual

Untuk merekam dan memutar suara secara manual dapat dilakukan dengan menggunakan tombol-tombol push button yang tersedia dalam Paket D-Voice 04 sebagai berikut.



Gambar 5.11 Instalasi sakelar dan LED pada D-Voice 04 untuk akses manual

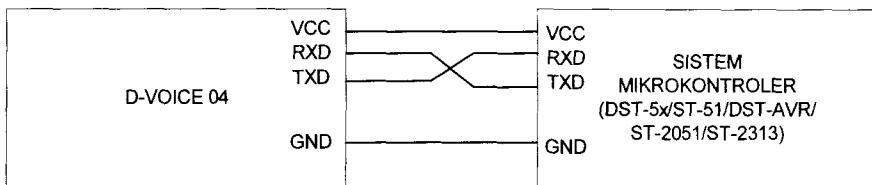
1. Lakukan instalasi seperti pada gambar.
 2. Hubungkan sumber suara (max 20 mV p-p) ke Line In D-Voice 04.
 3. Hubungkan port serial/UART ke PC bila sumber suara menggunakan sound card. Hal ini diperlukan untuk mereduksi noise dengan menghubungkan ground D-Voice dengan ground PC.
 4. Aktifkan power supply dengan memberikan sumber daya 9V-12V AC atau DC ke konektor power.
 5. Pastikan posisi sakelar Rec/Play pada posisi record, yaitu P3.2 terhubung dengan Ground.

6. Tekan tombol Reset/Stop untuk me-reset posisi rekaman ada pada posisi awal.
7. Tekan dan lepaskan tombol start sehingga LED aktif bersamaan dengan mengaktifkan sumber suara.
8. Tekan dan lepaskan lagi tombol start sehingga LED padam untuk menghentikan proses perekaman.
9. Proses dapat diulang dengan menekan dan melepas lagi tombol start. LED akan aktif lagi menandakan proses perekaman berlangsung. LED ini akan kembali nonaktif saat proses perekaman berhenti pada penekanan tombol start berikutnya. Rekaman suara akan tersimpan pada posisi selanjutnya kecuali bila tombol Reset/Stop ditekan.
10. Pindah posisi sakelar ke mode Play dan Restart sistem dengan mencabut dan memasang kembali sumber daya.
11. Sampai pada tahap ini, D-Voice sudah siap untuk diakses proses pemutarannya dengan mikrokontroler/PC.
12. Tekan tombol Reset/Stop untuk mengatur posisi D-Voice pada posisi awal rekaman.
13. Tekan tombol Start untuk mendengarkan isi rekaman.
14. Saat rekaman suara berakhir, tekan lagi tombol Start hingga terdengar rekaman kedua.

Proses D-Voice secara UART/Serial dengan Mikrokontroler

Untuk aplikasi-aplikasi tertentu, seperti suara pada robot, bel otomatis, voice announcer pada sistem antrian, D-Voice 04 dapat dihubungkan pada sistem mikrokontroler. Untuk proses ini dapat dilakukan dengan menghubungkan konektor UART/Serial dari D-Voice 04 dengan UART/Serial dari sistem mikrokontroler. Susunan pin out dari konektor UART/Serial ini kompatibel dengan UART/Serial dari sistem mikrokontroler Delta seperti DST-51/DST-52/DST-AVR/ST-2051 dan ST2313, yaitu seperti pada Gambar 5.10.

Bab 5: Sensor-Sensor Robot



Gambar 5.12 Instalasi D-Voice 04 dengan Sistem Mikrokontroler

Posisi RXD (pin 2) dan TXD (pin 3) dihubungkan secara cross (Gambar 5.12) sehingga serial output (TXD) sistem mikrokontroler terhubung dengan serial input (RXD) D-Voice 04 dan sebaliknya.

Perintah-perintah akan dikirim dari sistem mikrokontroler melalui port ini dengan protokol seperti pada tabel di bawah. D-Voice 04 akan membalas ke mikrokontroler berupa ACK (Acknowledge) yang mengindikasikan bahwa perintah telah diterima.

Tabel 3 Protokol Data D-Voice 04

Pengiriman data dari PC/Master ke Dvoice			
Byte	Deskripsi	Nilai	Keterangan
00	Header	1E	Awal paket data
01	Destination ID	13	ID Subsystem
02	Destination Number	01	No urut Sub System
03	Source ID	00	ID Pengirim 00 = Master (PC/Mikrokontroler)
04	Source Number	01	No urut Master
05	Length		Panjang paket data
06	Perintah		Baca jenis-jenis perintah
07-08	Isi Data		
	Check Sum		Total dari keseluruhan data = 0
Jenis-jenis perintah			
		Nilai	Keterangan
		01	Perintah Power UP
		02	Perintah Putar mulai dari alamat tertentu
		03	Perintah Putar pada alamat saat ini
		04	Perintah Rekam dengan alamat tertentu
		05	Perintah Rekam pada alamat saat ini
		07	Perintah Stop putar
		08	Perintah Stop Rekam
		09	Perintah Power Down

Contoh: Untuk mengaktifkan suara pada alamat 00 dilakukan dengan protokol 1E 13 01 00 01 03 02 00 C8 yang artinya Master (dalam hal ini sistem mikrokontroler dengan nomor urut 1) mengirimkan perintah ke D-Voice 04 dengan nomor urut 1 (pada versi ini selalu 01) untuk berada pada mode putar (Play) dan menuju posisi alamat 00 00 atau rekaman pertama.

Khusus untuk perintah putar pada alamat tertentu atau rekam pada alamat tertentu akan terdapat byte ke-7 dan byte ke-8. Byte ke-7 mewakili byte tinggi dari alamat rekaman suara D-Voice dan byte ke-8 adalah byte rendah dari alamat rekaman suara D-Voice 04.

Tabel 4 Pengiriman data dari D-Voice 04

Pengiriman data dari DVoice ke PC/Master		
	Nilai	Keterangan
Header	1E	Awal paket data
Destination ID	00	ID Pengirim 00 = Master (PC/Microcontroller)
Destination Number	01	No urut Master
Source ID	13	ID Subsystem
Source Number	01	No urut Subsystem
Length		Panjang paket data
Perintah	06	Kode ACK
Isi Data	4F/45	OK/Error
Check Sum		Total dari keseluruhan data = 0

D-Voice 04 akan membalas ACK berupa protokol data sesuai dengan Tabel 4. Untuk perintah yang dikirimkan di atas, maka balasannya adalah 1E 00 01 13 01 03 06 02 4F 73 bila data diterima dengan benar.

Pengalamatan Memori D-Voice 04

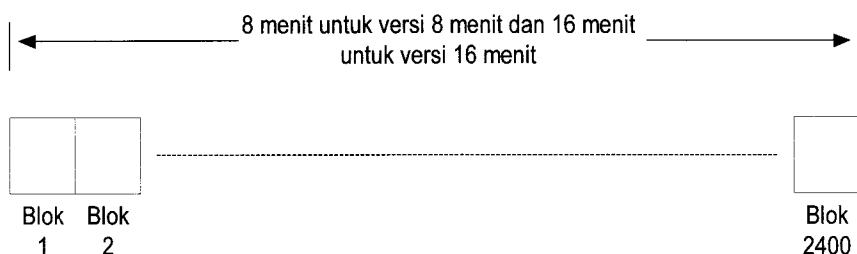
Modul D-Voice 04 terdiri atas versi 8 menit dan versi 16 menit yang dibedakan dari chipset-nya yang menggunakan ISD4004-8M atau ISD4004-16M. Versi 8 menit memiliki sampling rate 8 kHz, sedangkan versi 16 menit memiliki sampling rate yang lebih rendah, yaitu 4 kHz.

Bab 5: Sensor-Sensor Robot

Modul ini memiliki 2400 blok memori sehingga untuk versi 16 menit setiap blok memori akan menyimpan suara dengan durasi:

$$\text{durasi blok} = \frac{16 \times 60}{2400}$$

Yaitu = 0,4 detik



Gambar 5.13 Pengalamatan memori D-Voice 04

Dengan 2400 blok, maka dibutuhkan pengalamatan sebanyak 12 bit yang terdiri atas 2 byte, byte tinggi dan byte rendah. Untuk memutar suara pada lokasi 1 menit pertama dapat dilakukan dengan cara sebagai berikut.

1 menit = 60 detik. Pada versi 16 menit, 1 blok adalah 0,4 detik sehingga 1 menit akan menempati pada blok berikut

$$\begin{aligned}\text{Jumlah blok} &= \frac{60}{0,4} \\ &= 150 \text{ atau } 96h\end{aligned}$$

Dengan demikian, perintah untuk memutar suara pada alamat 1 menit pertama adalah perintah di mana byte tinggi dari protokol adalah 00 dan byte rendah adalah 96h

Dvttest:

'Program tester D-Voice 04 untuk merekam selama 2 menit dan memutar mulai 1 menit pertama
' By Paulus Andi Nalwan
'www.robotindonesia.com
.DATA

```

        Org      30
TargetType   Ds      1
TargetNum    Ds      1
ChecksumData Ds      1
Buffer       Ds      25
CounterDetik Ds      1
Counter_5mS  Ds      1

STX     EQU     1Eh

.CODE
Org      00
Ljmp    Start

Start:
Lcall   Init_Serial

;=====
; PROSES REKAM SELAMA 2 MENIT
Mov      TargetType,#13h           ;Target = Dvoice
Mov      TargetNum,#01
Mov      Buffer,#03                ;Panjang data = 3 byte
Mov      Buffer+1,#04              ;Perintah Rekam dengan alamat tertentu
Mov      Buffer+2,#00              ;Alamat 0
Mov      Buffer+3,#00h
Lcall   KirimPaketData          ;Kirim paket perintah ke D-Voice
Lcall   AmbilDataSerial          ;Ambil data ACK

Mov      CounterDetik,#120
LoopRekam:
Lcall   Delay_1detik
Djnz   CounterDetik,LoopRekam

;=====
; PERINTAH STOP      RECORD
Mov      TargetType,#13h           ;Target = Dvoice
Mov      TargetNum,#01
Mov      Buffer,#03                ;Panjang data = 3 byte
Mov      Buffer+1,#07              ;Perintah stop
Mov      Buffer+2,#00              ;Alamat 0
Mov      Buffer+3,#00h
Lcall   KirimPaketData          ;Kirim paket perintah ke D-Voice
Lcall   AmbilDataSerial          ;Ambil data ACK

;=====
; PERINTAH PUTAR MULAI DARI 1 MENIT PERTAMA HINGGA BERAKHIR

Mov      TargetType,#13h           ;Target = Dvoice
Mov      TargetNum,#01
Mov      Buffer,#03                ;Panjang data = 3 byte
Mov      Buffer+1,#02              ;Perintah putar
Mov      Buffer+2,#00              ;Alamat 1 menit pertama
Mov      Buffer+3,#96h
Lcall   KirimPaketData          ;Kirim paket perintah ke D-Voice
Lcall   AmbilDataSerial          ;Ambil data ACK
Ljmp   $

```

Bab 5: Sensor-Sensor Robot

```
AmbilDataSerial:  
    Lcall Serial_In  
    Cjne A,#1Eh,AmbilDataSerial  
    Lcall Serial_In  
    Lcall Serial_In  
    Lcall Serial_In  
    Lcall Serial_In  
    Lcall Serial_In  
    Mov B,A  
    Mov R0,#Buffer  
    Mov @R0,A  
    Inc R0  
LoopDataIn:  
    Lcall Serial_In  
    Mov @R0,A  
    Inc R0  
    Djnz B,LoopDataIn  
    Lcall Serial_In  
    Ret  
  
KirimPaketData:  
    Lcall AwalPaketMaster  
  
    Mov R0,#Buffer  
    Mov A,@R0  
    Mov R7,A  
    Lcall SerialOutCS  
  
    Mov R0,#Buffer+1  
LoopBufferOut:  
    Mov A,@R0  
    Inc R0  
    Lcall SerialOutCS  
    Djnz R7,LoopBufferOut  
    Mov A,#00  
    Clr C  
    Subb A,ChecksumData  
    Lcall Serial_Out  
    Ret  
  
AwalPaketMaster:  
    Mov A,#STX  
    Lcall Serial_Out  
    Mov CheckSumData,A  
  
    Mov A,TargetType  
    Lcall SerialOutCS  
  
    Mov A,TargetNum  
    Lcall SerialOutCS  
  
    Mov A,#00  
    Lcall SerialOutCS  
  
    Mov A,#01  
    Lcall SerialOutCS  
    Ret  
  
Init_Serial:  
    MOV SCON, #52H ; Mode 1 Ren  
    MOV TMOD, #20H ; T0 Mode 2, T1 Mode 2
```

```

        MOV     TH1,#0FDH      ; 9600 Baudrate
        Setb   TR1
        MOV     PCON,#00H      ;
        RET

Serial_Out:
        Clr     TI
        Mov     SBUF,A
TungguKirim:
        Jnb     TI,TungguKirim
        Clr     TI
        Ret

Serial_In:
        Clr     RI
LoopSerial_In:
        Jnb     RI,LoopSerial_in
        Mov     A,SBUF
        Cir     RI
        Ret

SerialOutCS:
        Lcall   Serial_Out
        Add     A,ChecksumData
        Mov     CheckSumData,A
        Ret

Delay_1detik:
        Mov     Counter_5mS,#0200

Tunggu_1detik:
        Acall   Delay_5ms
        Djnz   Counter_5mS,Tunggu_1detik
        Ret

Delay_5mS
        Push   TMOD
        Push   TCON
        Push   TH0
        Push   TL0
        Mov    TMOD,#21H          ;Timer Mode 16 bit counter
        Mov    TH0,#0EDH
        Mov    TL0,#0FFH
        Setb   TR0

Tunggu_5mS:
        Jbc   TF0,Sudah_5mS
        Ajmp  Tunggu_5mS

Sudah_5mS:
        Pop   TL0
        Pop   TH0
        Pop   TCON
        Pop   TMOD
        Ret

```

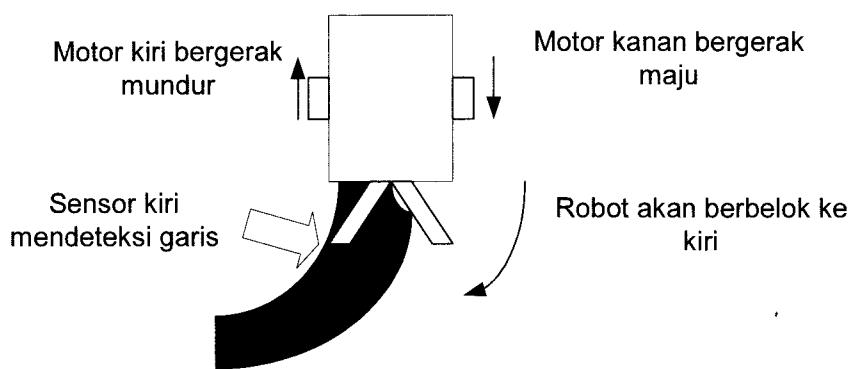
Sensor Penjejak Garis

Aplikasi robot line follower membutuhkan sensor yang memantau kondisi garis yang diikutinya. Untuk mengetahui keberadaan garis ini dilakukan dengan menembakkan sinar inframerah dan mendeteksi pantulannya. Garis hitam akan menyerap sinar inframerah dan sensor tidak mendeteksi pantulan tersebut sedangkan warna putih akan memantulkan sinar inframerah dan sensor akan mendeteksi adanya pantulan.

Ada dua macam sensor penjejak garis yang ada, yaitu sensor penjejak garis tunggal atau single line follower dan sensor garis inframerah (Delta Infrared Line Sensing) yang berupa sungut.

Penjejak Garis dengan Sungut

Penjejak garis dengan model sungut adalah berupa sepasang sensor yang dapat diatur posisinya baik secara maju-mundur maupun kiri-kanan



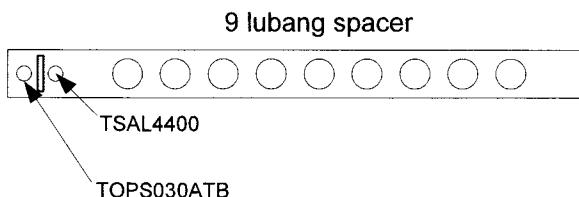
Gambar 5.14 Delta Infrared Line Sensing pada robot yang berjalan

Pada saat sungut sebelah kanan tidak mendeteksi garis hitam lagi, maka motor kanan akan bergerak maju dan motor kiri bergerak mundur sehingga robot akan berbelok ke kiri untuk memperbaiki posisinya.

Pengaturan posisi maju-mundur digunakan untuk mengatur waktu respon sensor. Untuk robot yang bergerak dengan kecepatan rendah, maka

respon sensor harus dibuat lebih lambat dan sebaliknya pada robot yang bergerak dengan kecepatan tinggi, maka respon sensor harus dibuat lebih cepat.

Agar sensor dapat lebih cepat merespon perubahan garis, sungut dapat diatur lebih maju sehingga robot akan mengetahui perubahan tersebut lebih dini. Pengaturan ini dapat dilakukan dengan mengatur posisi lubang spacer dari Delta IR Line Sensing yang digunakan untuk menahan dirinya terhadap tubuh robot (Gambar 5.14). Sedangkan untuk mengatur lebar garis yang dideteksi, Delta IR Line Sensing juga dapat digeser ke kanan dan ke kiri sesuai ketebalan garis.

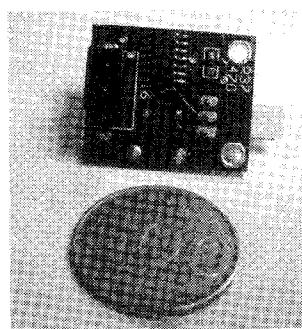


Gambar 5.15 Delta IR Line Sensing

Penjejak Garis Tunggal

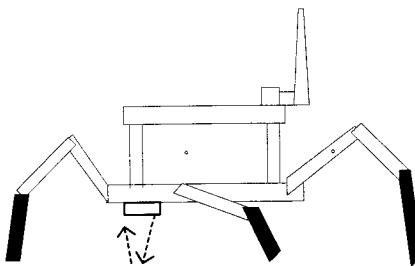
Penjejak garis tunggal atau single line follower merupakan sensor penjejak yang terpisah, bukan berupa pasangan sehingga sensor ini dapat diletakkan di sudut-sudut tertentu pada robot. Aplikasi sensor ini tidak hanya untuk mendeteksi garis, tetapi juga untuk mendeteksi keberadaan objek-objek di sekitar robot dalam jarak dekat. DSF-01 atau Delta Single Line Follower mempunyai berbagai variasi untuk fungsi-fungsi tertentu sebagai berikut.

Bab 5: Sensor-Sensor Robot



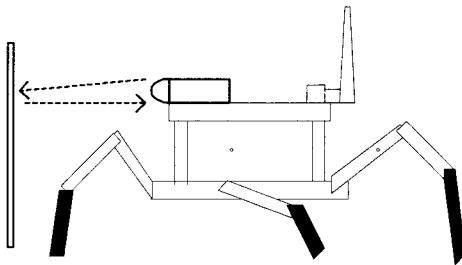
Gambar 5.16 DSF-01 Versi 1

DSF-01 V1 dengan jarak deteksi 0,5–1 cm dengan bentuk fisik sensor yang mengarah ke bawah berfungsi untuk mendeteksi garis pada bagian bawah robot.



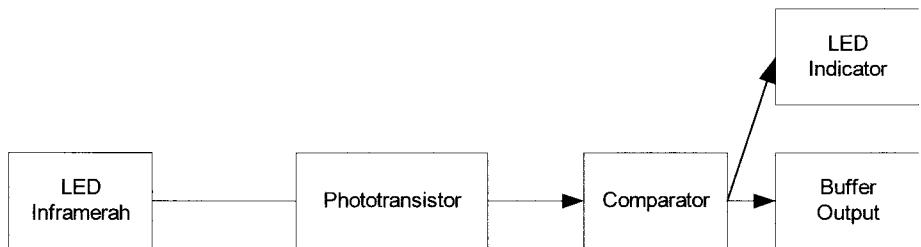
Gambar 5.17 Posisi DSF-01 Versi 1 pada robot

Untuk DSF-01 Versi 2 dengan jarak 0,5–3 cm, sensor ini dapat mendeteksi garis dalam jarak yang lebih jauh dan dapat diletakkan pada robot yang lebih tinggi posisinya dari permukaan lantai. Sedangkan DSF-01 Versi 3 yang memiliki jarak deteksi 0,5–9 cm lebih cocok digunakan untuk mendeteksi objek secara horizontal. Sensor ini dapat diletakkan pada sudut-sudut robot untuk mendeteksi keberadaan objek di sekitarnya. Dibandingkan jenis ultrasonik, sensor ini memiliki harga yang jauh lebih ekonomis.



Gambar 5.18 Posisi DSF-01 Versi 3 pada robot

DSF-01 Versi 3 menggunakan high power LED Inframerah TSAL4400 dan phototransistor TOPS-030TB2 sebagai penerimanya sehingga pancaran inframerah lebih kuat dibandingkan versi 1 dan 2. Pancaran yang kuat ini akan menghasilkan jarak yang lebih jauh pula.



Gambar 5.19 Diagram Blok Modul DSF-01

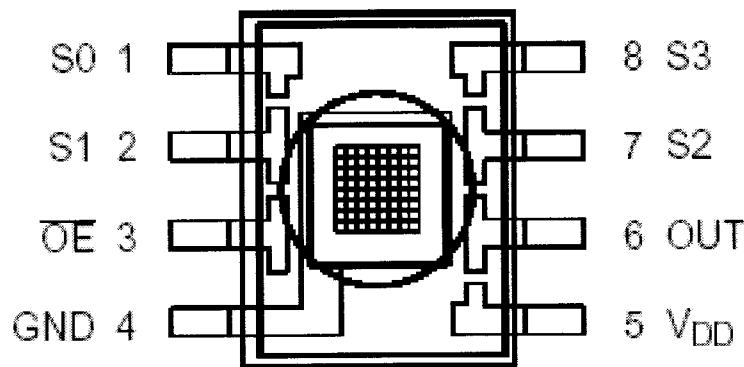
Modul DSF-01 baik versi 1 hingga versi 3 terdiri atas bagian pemancar dan penerima. Sisi pemancar berisi sebuah LED Inframerah yang selalu aktif mengirimkan cahaya ke objek. Sedangkan pada bagian penerima berupa phototransistor yang mengubah cahaya yang diterima menjadi sinyal listrik atau perubahan tegangan. Namun, perubahan tegangan tersebut hanya mengayun dalam rentang yang kecil sehingga tidak dapat dikenali oleh I/O logic dari mikrokontroler. Dengan komparator, ayunan tersebut akan diubah menjadi perubahan tegangan logik yang besar. Saat ayunan berada di atas tegangan komparasi, keluaran akan berlogika 1 dan saat ayunan berada di bawah tegangan komparasi, keluaran akan berlogika 0.

Pengaturan tegangan komparasi dilakukan dengan mengatur variabel resistor yang ada pada DSF-01. Objek diletakkan pada posisi di mana keberadaannya harus mulai terdeteksi. Variabel resistor diputar perlahan-lahan hingga LED merah aktif yang merupakan indikasi bahwa keberadaan objek telah terdeteksi.

Keluaran dari DSF-01 masih diperkuat lagi dengan sebuah buffer sehingga dapat diperoleh keluaran dengan arus yang cukup untuk didekksi mikrokontroler maupun fungsi-fungsi lainnya, misalnya dihubungkan ke transistor pengendali relay.

Sensor Warna TCS230

Salah satu identifikasi objek yang sering dibutuhkan pada aplikasi robotik adalah pengenalan warna. Dengan teknologi infrared, proses pengenalan warna dapat dilakukan dengan memperhitungkan kuat lemahnya penerusan cahaya.

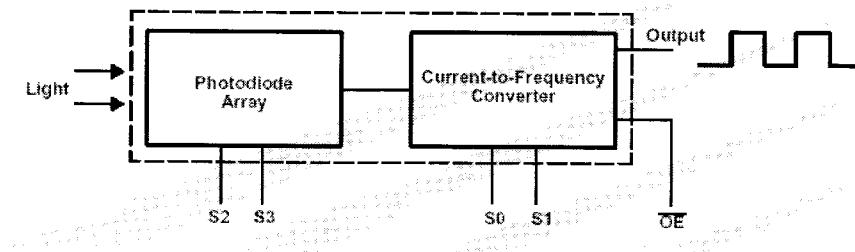


Gambar 5.20 Sensor warna TCS-230

TCS230 adalah sensor warna keluaran Texas Advance Optoelectronic Solutions yang berisi 8×8 matrix photodiode yang masing-masing berukuran $120 \mu\text{m}$. Photodiode tersebut terdiri atas 16 photodiode dengan filter merah, 16 dengan filter hijau, 16 dengan filter biru, dan 16 tanpa filter. Masing-masing kelompok photodiode ini diaktifkan melalui internal

decoder dengan memberikan kondisi logika tertentu pada input S2 dan S3.

Kuat lemah cahaya inframerah yang diterima oleh photodiode akan diubah menjadi kuat lemah arus. Melalui bagian pengubah arus ke frekuensi (*current to frequency converter*), kuat lemah arus yang diterima oleh photodioda diubah menjadi besaran frekuensi.



Gambar 5.21 Blok Diagram TCS-230

Pengenalan warna dilakukan dengan mendeteksi kuat lemah cahaya R, G, B, dan bening (*clear*) yang diterima oleh photodioda TCS230. Hal ini dilakukan dengan memilih kelompok-kelompok photodioda dan mengukur besaran-besaran frekuensi yang muncul di bagian keluaran TCS230 saat masing-masing kelompok tersebut diaktifkan.

Besaran frekuensi yang keluar dari TCS230 juga dapat diatur skala maksimumnya menjadi 2% atau 12 kHz, 20% atau 120 kHz, dan 100% atau 600 kHz melalui kondisi kaki S0 dan S1.

S0	S1	OUTPUT FREQUENCY SCALING (f_0)		S2	S3	PHOTODIODE TYPE
L	L	Power down		L	L	Red
L	H	2%		L	H	Blue
H	L	20%		H	L	Clear (no filter)
H	H	100%		H	H	Green

Gambar 5.22 Tabel Kebenaran TCS-230

Struktur warna dari setiap objek selalu terbentuk dari komponen warna dasar merah, hijau, dan biru. Jadi, pengenalan suatu warna dapat dilakukan dengan menghitung kuat lemah dari warna-warna dasar objek tersebut.

Bab 5: Sensor-Sensor Robot

Untuk mempermudah aplikasi, dapat digunakan Modul DCLI-230, Delta Color Interface yang sensornya sudah tersolder ke PCB dan dilengkapi dengan konektor untuk antarmuka dengan mikrokontroler.

Pada modul ini juga sudah disediakan 8 buah LED putih yang terhubung ke keluaran mikrokontroler sehingga LED bisa dinyala-matikan lewat program untuk penghematan daya. LED hanya diaktifkan pada saat proses pengenalan warna dan berfungsi selain untuk penerangan juga menetralkan pengaruh-pengaruh akibat pantulan cahaya pada warna-warna sekitarnya.

Tabel 5 Deskripsi I/O DCLI-230

Pin	Sinyal	Deskripsi
1	GND	Ground Power Supply. Semua tegangan mengacu ke Ground.
2	VDD	Tegangan Supply (2.7 – 5 volt)
3	S1	Input pemilih skala frekuensi output (penjelasan pada Tabel 6)
4	S0	Input pemilih skala frekuensi output (penjelasan pada Tabel 6)
5	S2	Input pemilih filter yang diaktifkan (penjelasan pada Tabel 7)
6	OE	Enable untuk f_0 (Aktif LOW)
7	S3	Input pemilih filter yang diaktifkan (penjelasan pada tabel 7)
8	OUT	Keluaran frekuensi (f_0)
9	LED	Kontrol LED (Aktif LOW)
10	NC	No Connect, tidak terhubung

Keluaran frekuensi dapat diatur skalanya melalui S0 dan S1, seperti pada Tabel 6. Untuk skala frekuensi rendah digunakan pada mikrokontroler yang mempunyai kecepatan rendah.

Tabel 6 Pengaturan Skala Frekuensi

S0	S1	Skala frekuensi Output
0	0	Power Down
0	1	2%
1	0	20%
1	1	100%

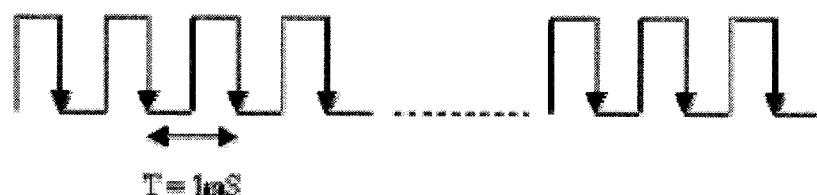
Tabel 7 Pengaturan Filter Warna

S2	S3	Photodiode yang aktif
0	0	Pemfilter merah
0	1	Pemfilter biru
1	0	Tanpa filter
1	1	Pemfilter hijau

DCLI-230 juga dapat mengatur aktivasi filter warna melalui S2 dan S3 dengan konfigurasi sesuai tabel. Dengan demikian, DCLI-230 dapat menentukan warna merah, hijau, atau biru yang difilter.

Keluaran dari DCLI-230 adalah berupa frekuensi sehingga mikrokontroler harus menghitung terlebih dahulu nilai frekuensi yang dikeluarkan oleh modul ini. Ada dua metoda yang digunakan untuk proses ini, yaitu metoda perhitungan nilai frekuensi dan metoda perhitungan nilai perioda.

Metoda perhitungan nilai perioda dilakukan dengan menghitung lebar perioda dari frekuensi yang dikeluarkan oleh DCLI-230. Hal ini digunakan untuk aplikasi deteksi warna yang membutuhkan respon yang tinggi di mana proses pengenalan warna langsung dilakukan setelah satu siklus frekuensi terdeteksi. Contoh pada gambar 5.23, proses pengenalan warna sudah dapat dinyatakan selesai setelah satu perioda berakhir, yaitu 1 ms. Dengan persamaan $f = 1/T$, maka nilai frekuensi telah diperoleh.



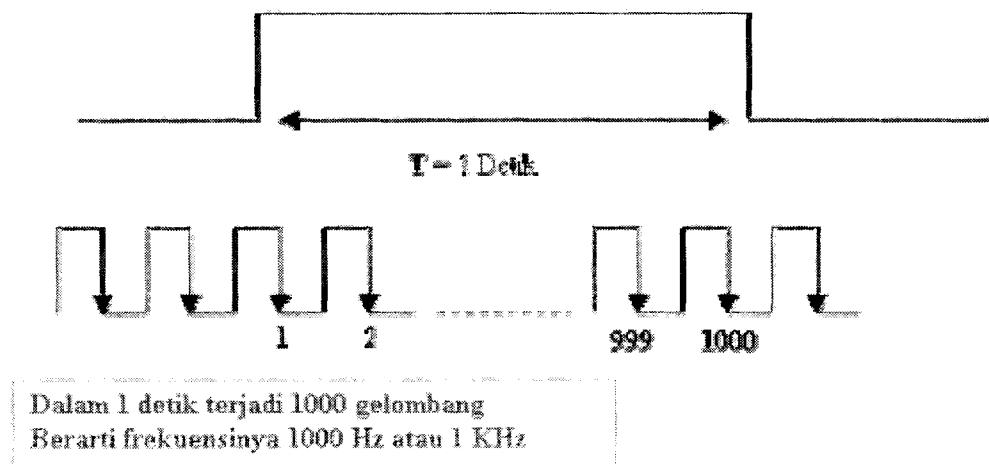
1 gelombang penuh periodenya 1ms,
Berarti frekuensinya $1/1\text{ms} = 1000 \text{ Hz}$ atau 1 KHz

Gambar 5.23 Metoda Penghitungan Perioda

Metoda perhitungan nilai frekuensi dilakukan dengan menghitung jumlah pulsa yang masuk ke mikrokontroler dalam setiap detik. Hal ini dilakukan untuk aplikasi yang membutuhkan ketelitian yang tinggi. Adanya noise kadang-kadang akan mengakibatkan perubahan nilai perioda sehingga

mempengaruhi perhitungan pengenalan warna yang membuat akurasi menjadi kurang baik. Dengan perhitungan frekuensi, maka mikrokontroler akan menghitung berapa jumlah pulsa yang masuk selama satu detik. Dengan proses ini, sedikit perubahan nilai periode tidak akan memberikan banyak pengaruh pada pengukuran frekuensi sehingga dapat diperoleh hasil yang lebih konsisten.

Kelemahan metoda ini adalah kecepatan proses pengukuran yang membutuhkan waktu paling cepat satu detik untuk mengenali warna.



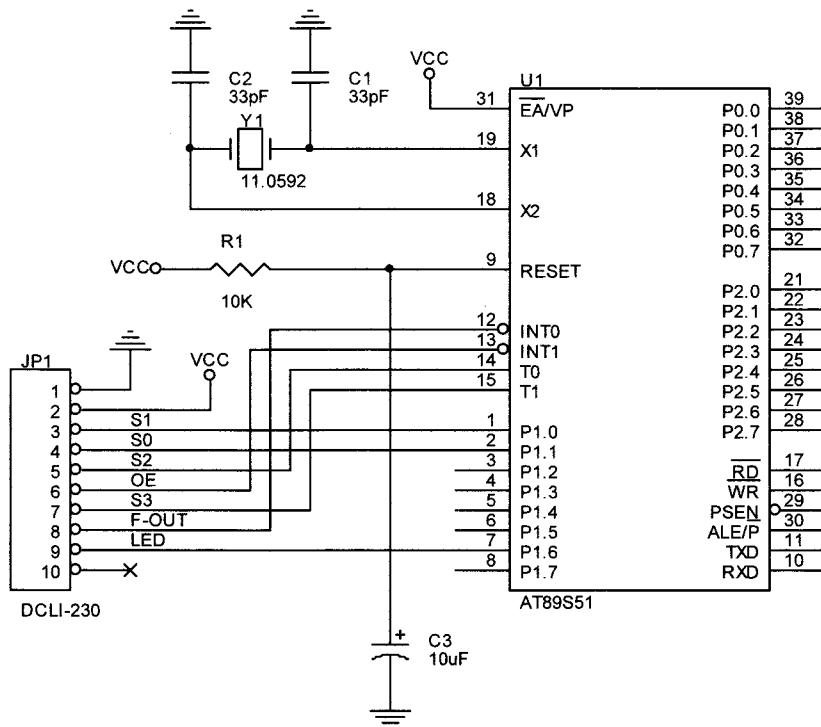
Gambar 5.24 Metoda Penghitungan Frekuensi

Hasil pengukuran dalam nilai frekuensi tadi, baik dari metoda perhitungan frekuensi maupun perhitungan periода, akan dibandingkan dengan nilai frekuensi skala penuh (full scale) dan filter cahaya mana yang diaktifkan. Semakin besar intensitas warna, semakin besar pula frekuensi yang dihasilkan.

Apabila skala penuh terjadi pada saat filter merah (R) diaktifkan, maka dapat dinyatakan bahwa warna merah dengan intensitas tertinggi sedang dideteksi. Apabila frekuensi yang dihasilkan mendekati nol pada saat filter merah (R), hijau (G), biru (B), dan cerah (clear) diaktifkan, berarti warna yang terdeteksi adalah hitam.

Proses Pengukuran Nilai Frekuensi

Berikut ini adalah aplikasi yang digunakan untuk menghitung nilai frekuensi yang diterima dari DCLI-230 dengan menggunakan metoda perhitungan frekuensi.



Gambar 5.25 Diagram Skematik AT89S51 dengan DCLI-230

Untuk menghitung banyaknya jumlah pulsa yang masuk dilakukan dengan menggunakan interupsi eksternal 0. Program akan melakukan delay selama satu detik dengan menggunakan Timer 0. Selama proses delay, interupsi eksternal akan diaktifkan sehingga mikrokontroler akan menghitung jumlah interupsi yang terjadi sebagai jumlah pulsa yang ada dalam perioda satu detik tersebut.

Nilai byte rendah akan tersimpan di Register A dan nilai byte tinggi akan tersimpan di Register B dalam bentuk hexa. Contohnya, untuk frekuensi 8

Bab 5: Sensor-Sensor Robot

kHz atau 8000 Hz akan diperoleh nilai 1F40h, di mana Register B bernilai 1Fh dan Register A bernilai 40h.

Untuk pengaturan nilai skala maksimum frekuensi pada 10 kHz, maka S0 dan S1 diaktifkan melalui P1.1 dan P1.0. Sedangkan agar diperoleh cahaya yang cukup terang, LED putih diaktifkan dengan memberikan logika 0 pada P1.6.

DCLI230.asm:

```
.Data
    CountR      Org  50h
    CountG      DS   2
    CountB      DS   2
    Count_1Detik DS   1
    S0          Bit   P1.1
    S1          Bit   P1.0
    S2          Bit   T0
    S3          Bit   T1
    OE          Bit   INT1
    LED         Bit   P1.6

.Code
    Org 000h
    Ljmp Start
    Org 003h
    Ljmp EX0isr
    Org 00Bh
    Reti
    Org 013h
    Reti
    Org 01Bh
    Reti
    Org 023h
    Reti

Start:
    Setb S1           ;Set frekuensi 10KHz
    Clr  S0           ;
    Clr  LED          ;Aktifkan LED
    Setb EA           ;aktifkan interupsi
    Lcall Ukur_Merah
    Lcall Ukur_Hijau
    Lcall Ukur_Biru
    Lcall DoSomething_WithData
    Sjmp Start

DoSomething_WithData:
;Proses pengolahan data tergantung jenis aplikasi
;letakkan kode programnya disini
;.....
    Ret

Ukur_Merah:
    Clr  S2
    Clr  S3
    Clr  OE
```

```

Lcall  Ukur
Setb   OE
Mov    R0, #CountR
Mov    @R0,A
Mov    A,B
Inc    R0
Mov    @R0,A
Ret

Ukur_Hijau:
Setb   S2
Setb   S3
Clr    OE
Lcall  Ukur
Setb   OE
Mov    R0, #CountG
Mov    @R0,A
Mov    A,B
Inc    R0
Mov    @R0,A
Ret

Ukur_Biru:
Clr    S2
Setb   S3
Clr    OE
Lcall  Ukur
Setb   OE
Mov    R0, #CountB
Mov    @R0,A
Mov    A,B
Inc    R0
Mov    @R0,A
Ret

Ukur:
Mov    TMOD, #01h
Mov    Count_1Detik, #20
Clr    A
Mov    B, #0
Jnb   INT0, $
Setb   EX0
Tunggu_1Detik:
Mov    TH0, #-50000/256
Mov    TL0, #-50000
Clr    TF0
Setb   TR0
Jnb   TF0, $
Clr    TR0
Djnz  Count_1Detik, Tunggu_1Detik
Clr    EX0
Ret

EX0isr:
Clr    EX0
Inc    A
Cjne  A, #0, Skip
Inc    B
Skip:
Setb   EX0
Reti

```

Latihan

1. Jelaskan bagaimana proses terjadinya pulsa-pulsa dari Rotary Encoder
2. Tuliskan protokol untuk meminta informasi jarak pada D-Sonar beserta protokol balasannya
3. Tuliskan protokol untuk memperdengarkan suara di menit kedua dari Modul D-Voice 04
4. Berapa arus maju maksimum dari LED Infrared TSAL4400?
5. Buat aplikasi penjejak garis dengan menggunakan Modul DST-51 dan Modul DSF-01
6. Buat program perhitungan frekuensi pada keluaran DCLI-230 dengan menggunakan metoda perhitungan perioda.
7. Telusuri lebih dalam sensor accelerometer, voice recognition, magnet, PIR, dan kompas.

Bab 6

Bagian Pengendali Robot

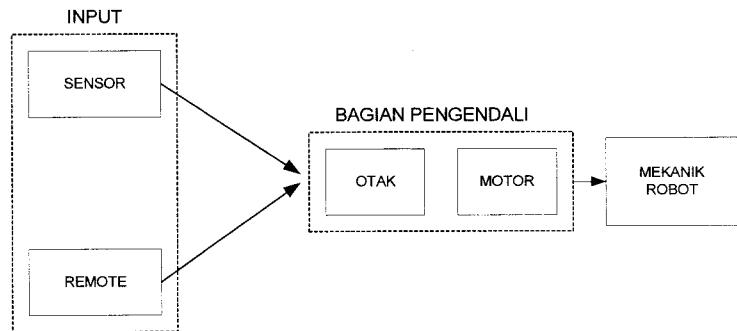
Pendahuluan

Bagian pengendali robot adalah bagian yang mengatur gerakan-gerakan robot berdasarkan input yang diperoleh secara manual melalui perintah operator, misalnya dengan remote, atau input yang diperoleh secara otomatis yang biasanya berasal dari sensor.

Input-input yang diperoleh tersebut akan digunakan oleh robot untuk melakukan reaksi lebih lanjut terutama pada bagian mekaniknya. Untuk itulah bagian pengendali ini dibutuhkan di antara input dan mekanik. Bagian ini terdiri atas otak sebagai pemroses dan pemberi keputusan serta motor sebagai pelaksana keputusan.

Selain motor, pada saat ini juga sudah dikembangkan teknologi *Muscle Wire* atau Kawat Otot sebagai pelaksana gerakan robot. Dengan mengalirkan arus tertentu pada kawat tersebut, kawat akan merapat dan merenggang sehingga menyerupai gerakan otot pada mahluk hidup. Namun, pada kesempatan kali ini akan kita bahas motor sebagai pelaksana keputusan.

Bagian input berupa remote dan sensor yang sebetulnya tidak mutlak harus ada pada sebuah robot. Ada juga beberapa robot yang memperoleh input dari data-data yang diprogramkan oleh sebuah console. Biasanya proses ini dilakukan pada robot-robot industrial.

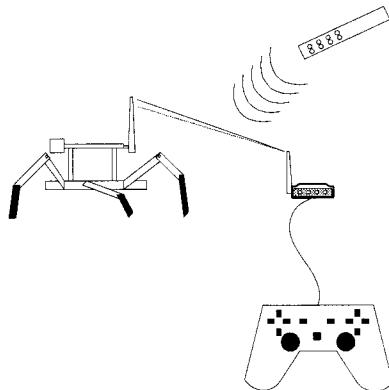


Gambar 6.1 Blok Diagram Bagian Pengendali Robot

Bagian Input

Seperti yang terlihat pada Gambar 6.1, bagian ini terdiri atas bagian sensor dan joystick/keypad sebagai pemberi masukan pada otak. Bagian sensor telah dijelaskan pada Bab 5. Oleh karena itu, pembahasan akan langsung masuk ke bagian remote. Pengertian remote kali ini bukanlah hanya merupakan remote kontrol seperti pada televisi atau AC, melainkan lebih luas daripada itu.

Remote adalah sebuah perangkat pengendali jarak jauh di mana perintah-perintahnya dikirimkan melalui media infra merah atau gelombang radio.



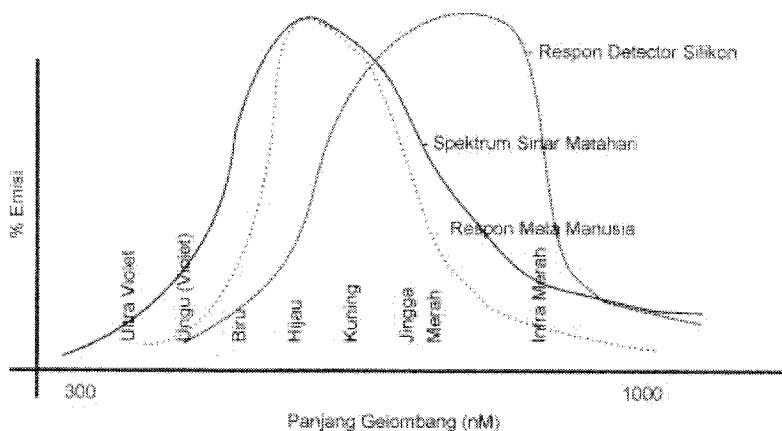
Gambar 6.2 Robot yang dikendalikan secara manual melalui joystick / remote control

Remote dengan Media Inframerah

Media inframerah merupakan media komunikasi data remote yang paling sederhana dan ekonomis. Oleh karena itu, media ini seringkali digunakan untuk remote peralatan-peralatan elektronik seperti televisi, AC, VCD, dan lain-lain.

Prinsip kerja remote ini adalah mengubah data menjadi pancaran cahaya inframerah dan mengubah kembali sinyal-sinyal tersebut menjadi data. Untuk media pengubah data menjadi pancaran cahaya inframerah tersebut kita dapat menggunakan remote inframerah yang sudah tersedia seperti remote AC maupun televisi. Sedangkan untuk mengubah kembali sinyal-sinyal tersebut menjadi data dapat digunakan modul bandpass filter seperti IR-8510 ataupun TSOP4838.

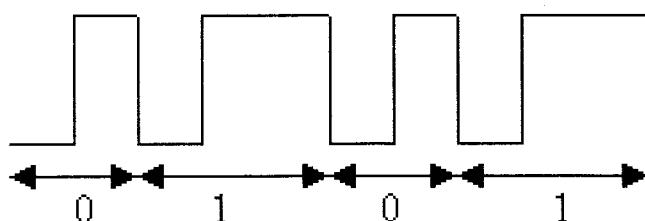
Untuk mempelajari media inframerah, terlebih dahulu kita perlu mengetahui apakah inframerah tersebut. Inframerah adalah sebuah cahaya dengan panjang gelombang yang titik puncaknya berada di luar respon mata manusia seperti tampak pada gambar 6.3



Gambar 6.3 Spektrum Inframerah

Sinar inframerah dalam aplikasi ini berfungsi sebagai media pembawa perintah dari sistem kendali ke robot. Agar data perintah yang dikirim dapat diterima dengan akurat, terlebih dahulu dilakukan modulasi *Pulse Code Modulation* atau PCM. Kondisi logika rendah dalam durasi pendek dan logika tinggi dalam durasi pendek akan mewakili data biner 0.

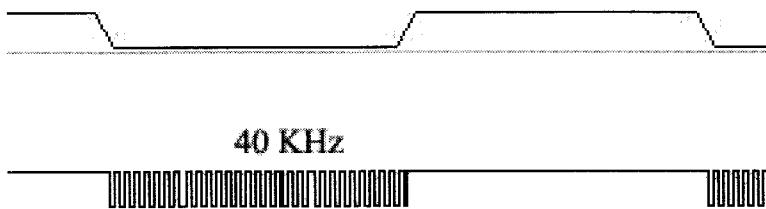
Kondisi logika rendah dalam durasi pendek dan logika tinggi dalam durasi panjang akan mewakili data biner 1.



Gambar 6.4 Timing Diagram PCM

Agar cahaya inframerah dapat mengirimkan data perintah dalam jarak yang jauh dan akurat, cahaya tersebut harus digetarkan dengan frekuensi tertentu. Standard frekuensi yang biasa digunakan oleh remote control berada pada area 38–42 kHz.

Data-data perintah yang sudah dimodulasikan dalam bentuk PCM ini akan dimodulasi lagi dengan frekuensi carrier sebesar 38–42 kHz sebelum diumpankan ke dioda pemancar.



Gambar 6.5 Timing Diagram Modulasi 40 KHz

Apabila sistem kendali yang digunakan adalah remote control televisi, Anda tidak perlu lagi mengatur PCM maupun modulasi 40 kHz ini. Sinyal yang dipancarkan dari remote control sudah berupa kode PCM dengan modulasi 40 kHz. Anda hanya perlu membuat demodulator 40 kHz dan program untuk mengurai PCM pada mikrokontroler yang ada pada robot. Bila sistem kendali Anda berupa PC atau joystick yang tidak memiliki pemancar infra merah, Anda harus menambahkan mikrokon-

troler yang berfungsi untuk mengatur PCM dan modulasi 40 kHz. Bagian ini akan dijelaskan lebih detail pada penjelasan tentang Proses Kendali Robot. Beberapa PC memang memiliki media inframerah, namun media ini menggunakan standard IrDA dengan kecepatan 1 Mbps dan ditujukan untuk komunikasi jarak dekat dengan kecepatan tinggi saja.

Data-data hasil decoding dengan menggunakan sistem PCM (Gambar 6.4) akan membentuk byte-byte tertentu yang merupakan kode dari setiap tombol-tombol dari remote tersebut.

Tabel 1 Tabel Fungsi Tombol dan Data yang Diterima (Remote Sony)

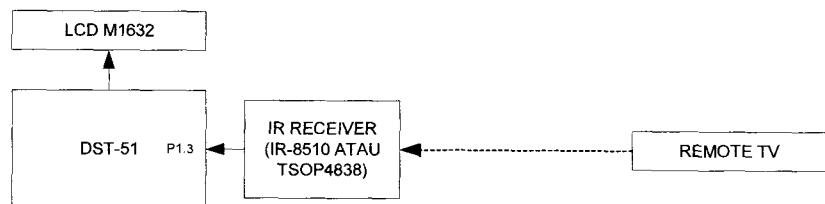
Tombol	Hexa	Tombol	Hexa
1	#080	Vol-	#093
2	#081	Power(toggle)	#095
3	#082	PIC Mode	#096
4	#083	A/B	#097
5	#084	TV/Video	#0A5
6	#085	Sleep	#0B6
7	#086	+	#0F4
8	#087	-	#0F5
9	#088	Select	#0FC
0	#089		
Prog+	#090		
Prog-	#091		
Vol+	#092		

Media komunikasi dengan inframerah ini mempunyai kelemahan. Bagian pemancar dan penerima harus bebas hambatan dan berada pada garis lurus. Robot akan kehilangan komando apabila berada di tempat yang terhalang atau tidak segaris dengan pemancar.

Aplikasi Penerima Data dari Remote Televisi

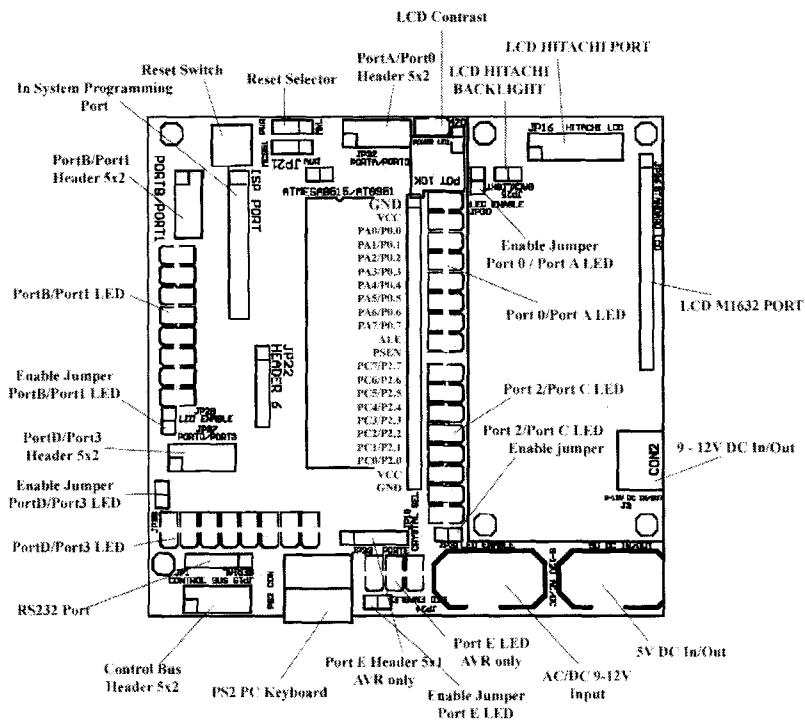
Pada program ini Anda dapat mempelajari cara mengambil data yang diterima dari modul penerima inframerah. Untuk mempermudah pengamatan data-data tersebut, maka data yang diterima akan ditampilkan pada layar LCD.

Bab 6: Bagian Pengendali Robot



Gambar 6.6 Diagram Blok Aplikasi Penampil Data Remote TV pada LCD

Modul DST-51. Sistem minimum AT89S51 akan digunakan sebagai sistem minimum mikrokontroler, mengingat pada modul ini sudah tersedia port LCD sebagai media penampil data.



Gambar 6.7 Deskripsi Modul DST-51 V3

Sinyal dari IR-8510 atau TSOP4838 akan diterima oleh Port P1.3 dan diterjemahkan menggunakan teknik PCM menjadi sebuah byte. Proses tersebut dilakukan dengan menggunakan Timer 0 dari AT89S51 yang menghitung lebar pulsa yang masuk pada P1.3

Rsony.asm:

```
$MOD51
.CODE

ROM EQU 000H

    Org ROM ;Reset Vector
    Ajmp Start ;
    Org ROM+3H ;External Interrupt 0 Vector
    Reti ;
    Org ROM+0BH ;Timer 0 Interrupt Vector
    Reti ;
    Org ROM+13H ;External Interrupt 1 Vector
    Reti ;
    Org ROM+1BH ;Timer 1 Interrupt Vector
    Reti ;
    Org ROM+23H ;Serial Interrupt Vector
    Reti ;

PanjangData EQU 26

Start:
    Lcall Init_LCD
    Acall InitTimer ;Inisial timer untuk trap data

Loop:
    Acall AmbilRemote
    Cjne A,#7FH,$+3 ;Kode di bawah 80H, abaikan
    Jc Loop ;
    Acall TunjukTabelTombol ;Tunjuk tabel sesuai kode yang diterima
    Movx A,@DPTR ;Bila tabel kosong, abaikan
    Jz Loop ;
    Lcall Posisi_Awal ;Tampilkan isi tabel bila tidak kosong
    Lcall KirimPesan_LCD ;
    Lcall Posisi_Awal ;
    Ajmp Loop ;

TunjukTabelTombol:
    Mov DPTR,#TabelTombol
    Clr C
    Subb A,#80H
    Mov B,#7
    Mul AB

    Clr C
    Add A,DPL
    Jnc TidakIncDPH
    Inc DPH

TidakIncDPH:
    Mov DPL,A
    Mov A,B
    Add A,DPH
```

Bab 6: Bagian Pengendali Robot

```
Mov      DPH,A
Ret

AmbilRemote:
    Acall  CekStartBit           ;Tunggu Start Bit
    Mov    R7,#8                ;Panjang data bit = 8

AmbilData:
    Push   A                   ;Ambil 1 bit
    Acall  Bit                 ;
    Pop    A                   ;
    Rrc   A                   ;Geser ke akumulator
    Djnz  R7,AmbilData        ;Lakukan 8x
    Clr   TR0
    Ret

CekStartBit:
    Jb    P1.3,$
    Setb TR0
    Jnb   P1.3,$
    Mov   A,TH0
    Cjne A,#08H,$+3
    Jnc   CekStartBitSelesai
    Clr   TR0
    Mov   TH0,#00
    Mov   TL0,#00
    Ajmp  CekStartBit

CekStartBitSelesai:
    Clr   TR0
    Mov   TH0,#00
    Mov   TL0,#00
    Ret

Bit:
    Jb    P1.3,$
    Setb TR0
    Jnb   P1.3,$
    Clr   TR0
    Mov   A,TH0
    Cjne A,#03,$+3
    Jnc   Bit1
    Clr   C
    Mov   TH0,#0
    Mov   TL0,#0
    Ret

Bit1:
    Setb C
    Mov   TH0,#0
    Mov   TL0,#0
    Ret

InitTimer:
    Mov   TH0,#00               ;
    Mov   TL0,#00               ;
    Push  A                   ;
    Mov   A,TMOD
    Anl   A,#0F0H
    Orl   A,#01H
    Mov   TMOD,A
    Pop   A
```

```

        Ret

RS      Bit      P2.1
.CODE
Org     *
;-----
;Geser Display ke kanan

GeserDisplay_Kanan:
    Mov     A,#1FH
    Acall   Kirim_Perintah
    Acall   Delay_LCD
    Ret

;-----
;Geser Display ke kiri

GeserDisplay_Kiri:
    Mov     A,#18H
    Acall   Kirim_Perintah
    Acall   Delay_LCD
    Ret

;-----
;Kembali ke posisi awal

Posisi_Awal:
    Mov     A,#02H
    Acall   Kirim_Perintah
    Acall   Delay_LCD
    Ret

;-----
;Geser Cursor ke kiri

GeserCursor_Kiri:
    Mov     A,#10H
    Acall   Kirim_Perintah
    Acall   Delay_LCD
    Ret

;-----
;Geser Cursor ke kanan

GeserCursor_Kanan:
    Mov     A,#14H
    Acall   Kirim_Perintah
    Acall   Delay_LCD
    Ret

Kirimpesan_LCD:
LoopKirimpesan_LCD:
    Mov     A,#00H
    Movc   A,@A+Dptr
    Cjne   A,#0FH,Kirim_LCD
    Ret
;Ambil data dari memori yg ditunjuk
;
;Kirim ke LCD selama belum ditemukan
;0FH

Kirim_LCD:
    Acall   Kirim_Karakter
    Inc    Dptr
    Ajmp   LoopKirimpesan_LCD
;Tunjuk ke memori selanjutnya

```

Bab 6: Bagian Pengendali Robot

```
***** SUBROUTINE INISIAL LCD *****
Init_LCD:
    Setb    RS

    Mov     A,#30H          ;Kirim 30H
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#30H          ;Kirim 30H
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#30H          ;Kirim 30H
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#20H          ;Send Init
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#2FH           ;8x5 2lines
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#08H          ;
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#01H          ;
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#07H          ;
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#0EH           ;Display ON
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD

    Mov     A,#06H           ;Mode Increment Address
    Acall   Kirim_Perintah ;
    Acall   Delay_LCD
    Ret

Kirim_Perintah:
    Clr     P2.1
    Acall   Kirim_DataLCD      ;4 bit sebanyak 2 x
    Swap    A
    Acall   Kirim_DataLCD      ;
    Acall   Delay_LCD
    Ret

Kirim_DataLCD
    Mov     P0,A              ;Kirim ke Port 0
    Acall   PulseE_Clock      ;Beri 1 pulsa E Clock
    ret

PulseE_Clock:
    Push   A
    Mov    A,#0AOH
    Orl    A,P2
    Mov    P2,A
```

```

Anl    A, #0FH
Mov    P2,A
Pop    A
Nop
Ret

Kirim_Karakter:
Setb   P2.1
Acall  Kirim_DataLCD      ; ke LCD 2x
Swap   A
Acall  Kirim_DataLCD      ;
Acall  Delay_LCD          ; Delay LCD
Ret

Baris2_LCD:
Mov    A, #0C0H
Acall  Kirim_Perintah
Ret

Baris1_LCD:
Mov    A, #02H
Acall  Kirim_Perintah
Acall  Delay_LCD2
Ret

Delay_LCD:
Push   B
Mov    B, #06H
Delay_LCD_Loop:
Push   B
Acall  Delay_LCD2
Pop    B
Djnz   B,Delay_LCD_Loop
Pop    B
Ret

Delay_LCD2:
Mov    B, #0FFH
Djnz   B, *
Ret

;=====
; TABEL TOMBOL-TOMBOL REMOTE CONTROL TIPE RM-827S
; - Data-data yang masih berisi 00 adalah kode cadangan untuk remote control
; - Sony tipe yang lain
=====

TabelTombol:
DB    '1    ',0FH
DB    '2    ',0FH
DB    '3    ',0FH
DB    '4    ',0FH
DB    '5    ',0FH
DB    '6    ',0FH
DB    '7    ',0FH
DB    '8    ',0FH
DB    '9    ',0FH
DB    '0    ',0FH
DB    0,0,0,0,0,0,0,0 ;8a
DB    0,0,0,0,0,0,0,0 ;8b
DB    '1-   ',0FH
DB    '2-   ',0FH

```

Bab 6: Bagian Pengendali Robot

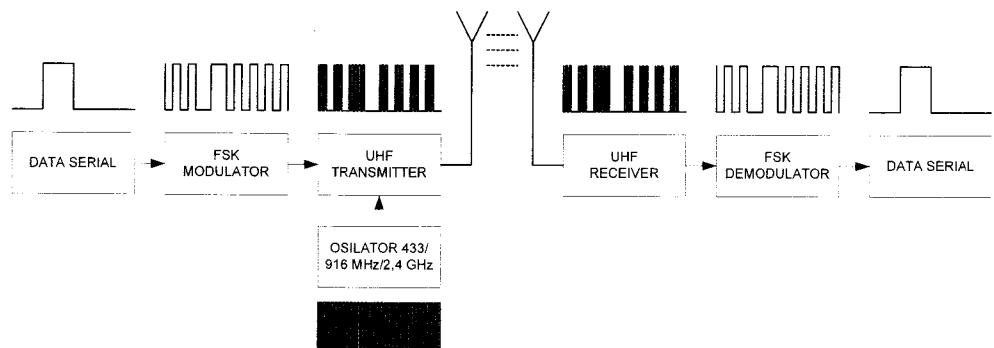
```
DB      0,0,0,0,0,0,0 ;8e
DB      0,0,0,0,0,0,0 ;8f
DB      'Prog+',0FH
DB      'Prog-',0FH
DB      'Vol+',0FH
DB      'Vol-',0FH
DB      0,0,0,0,0,0,0
DB      'Power',0FH
DB      'Pic Md',0FH
DB      'A/B',0FH
DB      0,0,0,0,0,0,0 ;98
DB      0,0,0,0,0,0,0 ;99
DB      0,0,0,0,0,0,0 ;9a
...
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      '+',0FH
DB      '-',0FH
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      0,0,0,0,0,0,0
DB      'Select',0FH
END
```

Remote dengan Media Gelombang Radio

Gelombang radio merupakan media yang paling leluasa, mengingat media ini mampu menembus halangan dan tidak harus berada pada garis lurus. Namun, dibanding inframerah, media ini lebih rentan terhadap noise apabila disain pemancar dan penerimanya kurang baik. Hal ini disebabkan oleh tingginya frekuensi yang digunakan sehingga sedikit perubahan pada jalur PCB saja akan membawa pengaruh yang cukup besar.

Bila pada media inframerah digunakan PCM sebagai modulasi data, pada gelombang radio digunakan metode ASK (*Amplitude Shift Keying*) atau FSK (*Frequency Shift Keying*). PSK (*Phase Shift Keying*) kadang-kadang juga digunakan untuk aplikasi-aplikasi tertentu. Yang paling umum digunakan adalah FSK, di mana getaran dengan frekuensi lebih tinggi biasanya mewakili logika 0 dan frekuensi lebih rendah biasanya mewakili logika 1.

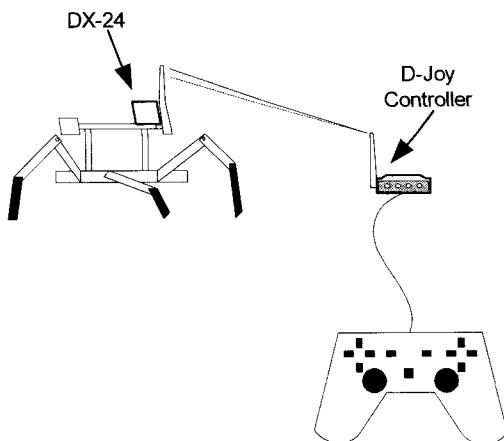
Sinyal FSK tersebut sebelum diumpulkan ke antena pemancar terlebih dahulu digetarkan dengan frekuensi tertentu oleh osilator. Yang umum digunakan pada aplikasi robotik adalah 433 MHz, 916 MHz, dan 2,4 GHz.



Gambar 6.8 Komunikasi Gelombang Radio dengan Modulasi FSK

Frekuensi dari osilator adalah frekuensi pembawa (*carrier*) dari pemancahar UHF. Dengan menggunakan frekuensi yang tinggi, maka ukuran panjang antena dapat diperpendek.

Untuk mempermudah proses pengendalian melalui gelombang radio ini dapat digunakan modul D-Joy Controller, yaitu sebuah modul Joystick Controller yang mengubah gerakan-gerakan dan penekanan tombol joystick menjadi pancaran data gelombang radio. Pada sisi penerima dapat digunakan Modul DX-24 yang mengubah sinyal-sinyal gelombang radio yang diterima menjadi data.



Gambar 6.9 Komunikasi D-Joy Controller dengan Modul DX-24 pada robot

Dengan menggunakan D-Joy Controller, proses modulasi-demodulasi FSK maupun modulasi-demodulasi 2.4GHz telah diatur oleh modul DX-24 yang ada pada robot maupun yang ada pada D-Joy Controller. Selain itu, proses pengalaman dan pengaturan frekuensi dapat diatur dengan mengubah nilai-nilai register pada DX-24.

Modul D-Joy Controller ini juga mengkonversi data dari Joystick Playstation menjadi data-data yang akan dikirim ke DX-24. Data dari Joystick dikirim melalui kaki data (1) dari konektor joystick.

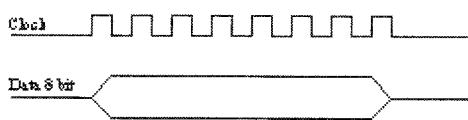


Gambar 6.10 Konektor Joystik

Kaki nomor 2 berfungsi untuk mengirim data dari mikrokontroler ke joystick. Baik data maupun command, proses pengiriman selalu diiringi dengan sinyal clock pada kaki nomor 7 (Gambar 6.10). Setelah Command diterima oleh joystick, sinyal ACK akan dikirimkan oleh joystick ke mikrokontroler sebagai indikasi bahwa perintah dari mikrokontroler telah diterima.

Tabel 2 Konfigurasi Pin Joystick

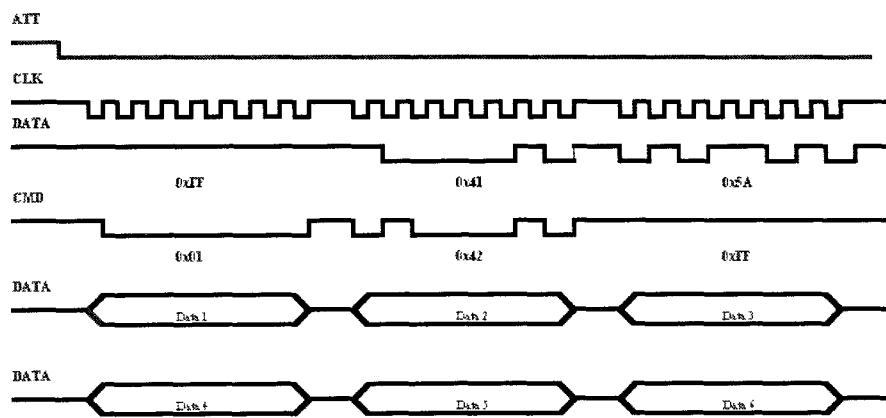
Pin	Nama	Keterangan
1	DATA	Data
2	CMD	Command
3	N/C	Tidak dihubungkan
4	GND	Ground
5	VCC	Vcc
6	ATT	Attention / Pemilih
7	CLK	Clock
8	N/C	Tidak di hubungkan
9	ACK	Acknowledge



Gambar 6.11 Pengiriman 1 byte data secara serial

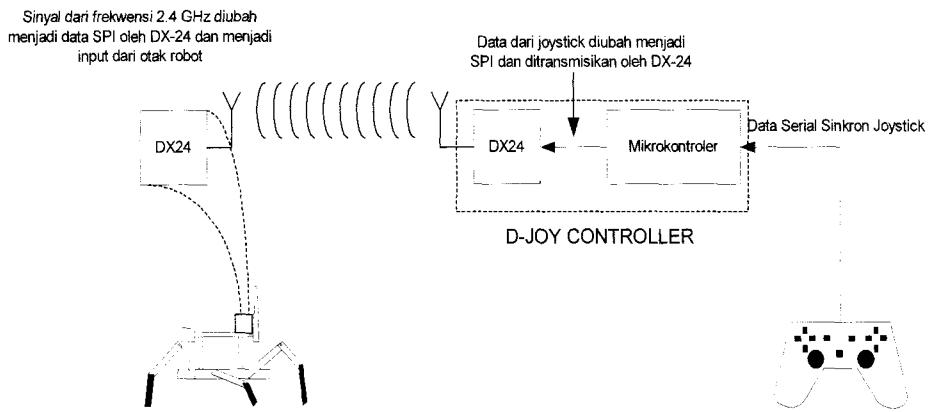
Jika mikrokontroller ingin mengambil data dari joystick, mikrokontroler harus membuat pin ATT berlogika low dan mengirim data sebagai start command (0x01). Setelah start command diterima oleh joystick, maka joystick akan mengirim data yang menunjukkan ID dari joystick (0x41 untuk joystick digital dan 0x73 untuk joystick analog). Pada saat joystick mengirim ID, mikrokontroler juga mengirim data (0x42) untuk meminta data dari joystick. Setelah data (0x42) diterima oleh joystick, maka joystick membalas dengan mengirimkan data (0x52) sebagai pemberitahuan bahwa data akan dikirim. Setelah itu, joystick mengirimkan data 6×8 bit yang berisi informasi tombol mana saja yang ditekan (semua tombol adalah aktif low).

Bab 6: Bagian Pengendali Robot



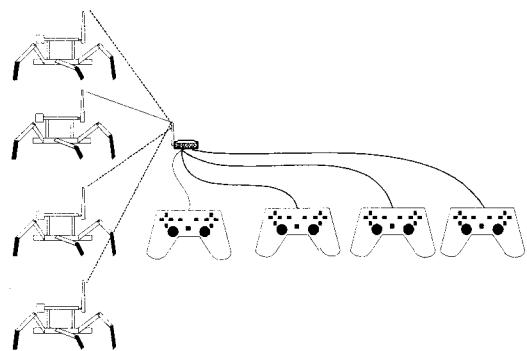
Gambar 6.12 Timing Diagram Pengambilan data

Dengan adanya D-Joy Controller, proses pengambilan data tersebut telah diatur oleh modul ini sehingga anda hanya perlu mempelajari data yang diterima pada DX-24 penerima (di bagian robot saja).



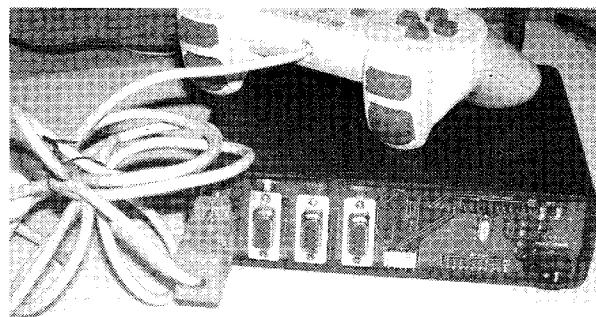
Gambar 6.13 Proses Komunikasi Data dari Joystick ke Robot melalui Gelombang Radio

Modul D-Joy Controller ini memiliki 4 port joystick sehingga dapat digunakan untuk mengendalikan 4 buah robot dengan 4 joystick hanya dengan menggunakan 5 buah Modul RF DX-24. Empat buah modul diletakkan pada 4 robot dan sebuah di D-Joy Controller (Gambar 6.14).

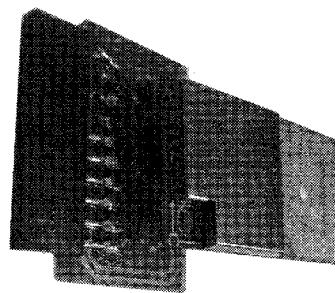


Gambar 6.14 Kendali 4 robot dengan 4 joystick oleh sebuah D-Joy Controller

Modul D-Joy Controller akan melakukan scanning data dari keempat joystick yang terhubung dan mengirimkannya secara bergantian ke alamat masing-masing DX-24 yang ada pada robot.



Gambar 6.15 Modul D-Joy Controller dengan sebuah joystick terhubung



Gambar 6.16 Modul DX-24 Transceiver 2.4GHz

Bab 6: Bagian Pengendali Robot

Tabel 3 Data RF Joystick

Data	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Data 1	Analog kanan arah kanan kiri (00 sampai FF)							
Data 2	Analog kanan arah atas bawah (00 sampai FF)							
Data 3	L2	R2	L1	R1			X	
Data 4	select	x	x	Start	up	right	down	left

Tabel 3 menunjukkan data-data yang diterima oleh bagian DX-24 pada robot. Dengan adanya D-Joy Controller, Anda dapat fokus mengolah data yang ada di tabel tersebut tanpa harus mengatur sinyal-sinyal dari joystick. Selain pengiriman data melalui gelombang radio, modul ini juga dapat mengirimkan data melalui port serial dengan protokol seperti pada Tabel 4. Pengiriman data melalui port serial ini dilakukan dengan menekan tombol select pada joystick.

Tabel 4 Protokol Data Serial D-Joy Controller

Pengiriman data dari Joystick ke PC/Master			
Byte	Data	Nilai	Keterangan
Byte 0	Header	1E	
Byte 1	Destination ID	00	
Byte 2	Destination Number	01	
Byte 3	Source ID	12	ID D-Joy Controller
Byte 4	Source Number	01	Nomor urut Joystick/Robot
Byte 5	Length	07	01 = Keluaran data dari Joystick 02 = Keluaran data dari robot yang dikirim lewat RF
Byte 6	Perintah	01/02	
	Isi Data		
	Checksum		

Keluaran data dari Joystick

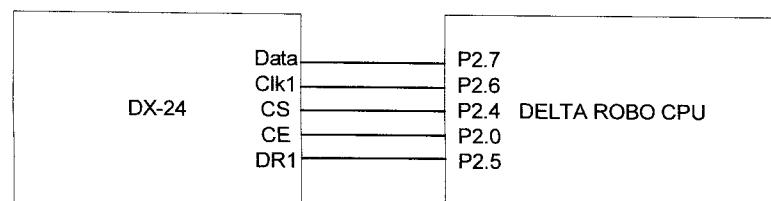
Byte 6	Jenis Perintah	01	
Byte 7	Data 1 Joystick		
Byte 8	Data 2 Joystick		
Byte 9	Data 3 Joystick		
Byte 10	Data 4 Joystick		
Byte 11	Data 5 Joystick		
Byte 12	Data 6 Joystick		-
Byte 13	Checksum		

Keluaran data dari robot (hanya pada Delta Robo Soccer)

Byte 6	Jenis Perintah	02	
Byte 7	Kondisi logika Extra Port dari bit 3 hingga bit 7		
Byte 8	Selalu 00	00	
Byte 9	Selalu 00	00	
Byte 10	Selalu 00	00	
Byte 11	Selalu 00	00	
Byte 12	Checksum		

Aplikasi Penerima Data dari D-Joy Controller

Pada aplikasi berikut ini, Anda dapat mempelajari bagaimana data joystick yang dikirimkan oleh D-Joy Controller diterima oleh DX-24 yang ada pada robot dan diolah oleh mikrokontroler sebagai otak dari robot.



Gambar 6.17 Hubungan DX-24 dengan Delta Robo CPU

Delta Robo CPU yang menggunakan mikrokontroler AT89S51 dalam hal ini berfungsi sebagai otak robot dan terhubung dengan DX-24 sesuai

Bab 6: Bagian Pengendali Robot

konfigurasi yang dapat dilihat pada Gambar 6.17. Data-data akan tersimpan pada Buffer TRW24 dan Anda dapat menggunakan data-data tersebut untuk aplikasi robotik sesuai kebutuhan.

D-joy.asm:

```
$MOD51
;=====
;DEKLARASI I/O SESUAI DELTA ROBO CPU PORT
DATA      BIT    P2.7
CLK1     BIT    P2.6
CS        BIT    P2.4
CE        BIT    P2.0
CLK2     BIT    P2.2
DATA2    BIT    P2.3
DR2      BIT    P2.1
DR1      BIT    P2.5

.DSEG
Org      30h
PanjangData EQU    4
BufferTRW24  Ds     PanjangData

;=====
; MAIN PROGRAM
; - Data dari dari joystick akan disimpan pada BufferTRW24, BuffertRW24+1
;   hingga BufferTRW24+3
;

.CSEG
Org      00

Lcall    InitRecTRW24
Clr      CS
Setb    CE

Loop:
Jnb      DR1,$
Mov      R7,#PanjangData
Mov      R0,#BufferTRW24
Lcall    LoopTerimaData
Ljmp    Loop

;
;=====

;KUMPULAN SUBROUTINE
;*****;Subroutine
;untuk mengirim data TRW24 dengan pengalaman sebanyak 2 byte
; - R6 = alamat byte 2
; - R5 = alamat byte 1
; - R7 = panjang data

SendTRW24:
Mov      A,R6                      ;Kirim byte 2 alamat tujuan
```

```

        Acall  Kirim_Data           ;
        Mov    A,R5                ; Kirim byte 1 alamat tujuan
        Acall  Kirim_Data           ;
        Mov    R0,#BufferTRW24      ;
LoopSendTRW24:
        MOV    A,@R0
        Inc    R0
        ACALL KIRIM_DATA
        Djnz   R7,LoopSendTRW24

        CLR    CE
        Mov    B,#80h
        Djnz   B,$
        SETB   CE
        DEC    R1
        Ret

KIRIM:
        PUSH   B
        MOV    B,#8
KIRIM8X:
        CLR    CLK1
        RLC    A
        MOV    DATA,C
        SETB   CLK1
        CLR    CLK1
        CLR    DATA
        DJNZ   B,KIRIM8X
        POP    B
        RET

KIRIM_PRINTAH:
        CLR    CE
        SETB   CS
        SJMP   KIRIM_KAN

KIRIM_DATA:
        SETB   CE
        CLR    CS
KIRIM_KAN:
        ACALL KIRIM
        RET

;=====
;Subroutine untuk menerima data TRW24
;- R7 = panjang data
;- R0 = alamat awal buffer data

TERIMA:
        Mov    R0,#BufferTRW24
        Mov    R7,#PanjangData
        CLR    CS
        SETB   CE
        SETB   DR1
        JNB    DR1,$
LoopTerimaData:
        Acall  TerimaData
        Mov    @R0,A
        Inc    R0
        Djnz   R7,LoopTerimaData
        RET

```

Bab 6: Bagian Pengendali Robot

```
TerimaData:
    PUSH    B
    MOV     B, #8
    MOV     A, #0H
TERIMA8X:
    CLR     CLK1
    SETB   DATA
    SETB   CLK1
    RL    A
    Nop
    Nop
    MOV     C, DATA
    MOV     A.0, C
    DJNZ   B, TERIMA8X
    CLR     CLK1
    POP    B
    Ret

=====
; ; SUBROUTINE UNTUK INISIALISASI DX-24 / TRW-24

InitSENDTRW24:
    Acall  Init_TRW24

    Acall  Ambil_Tabel
    Clr   A.0
    Acall  Kirim_Printah

    Ret

InitRecTRW24:
    Acall  Init_TRW24
    Acall  Ambil_Tabel
    Acall  Kirim_Printah
    Ret

INIT_TRW24:
    CLR    CE
    CLR    DATA
    CLR    CS
    CLR    CLK1

    Mov   A, #0h           ; Send Reserved Byte
    Acall Kirim_Printah
    Mov   A, #0             ;
    Acall Kirim_Printah
    Mov   A, #0H            ;
    Acall Kirim_Printah
    Mov   A, #1*8           ; Channel 2 panjang 1 byte
    Acall Kirim_Printah
    Mov   A, #8*PanjangData ; Channel 1 panjang 6 byte (48 bit)
    Acall Kirim_Printah

    Mov   A, #CH2AddressByte4
    Acall Kirim_Printah
    Mov   A, #CH2AddressByte3
    Acall Kirim_Printah
    Mov   A, #CH2AddressByte2
    Acall Kirim_Printah
    Mov   A, #CH2AddressByte1
    Acall Kirim_Printah
```

```

Mov      A,#CH2AddressByte0
Acall   Kirim_Printah

Mov      A,#CH1AddressByte4
Acall   Kirim_Printah
Mov      A,#CH1AddressByte3
Acall   Kirim_Printah
Mov      A,#CH1AddressByte2
Acall   Kirim_Printah
Mov      A,#CH1AddressByte1
Acall   Kirim_Printah
Mov      A,#CH1AddressByte0
Acall   Kirim_Printah

MOV      B,#2
MOV      DPTR,#TABEL_INIT
INT_TRW24KIRIM:
Acall   Ambil_Tabel
ACALL   KIRIM_PRINTAH
INC     DPTR
DJNZ   B, INT_TRW24KIRIM
RET

AMBIL_TABEL:
MOV      A,#0
MOVC   A,@A+DPTR
RET

TABEL_INIT:
DB      01000011B ;ADDR Length = 16
;16 bit CRC & ON Chip CRC Generator Enable

DB      01101111B ;Single Channel
;ShockBurst
;1 MBps
;16 MHz Crystal (011)
;Power 0dB

DB      00000001B ;Frekuensi 2.4GHz + 0 x 10 MHz

NomorRobot EQU    01

CH2AddressByte4    EQU    00
CH2AddressByte3    EQU    00
CH2AddressByte2    EQU    00
CH2AddressByte1    EQU    00
CH2AddressByte0    EQU    00

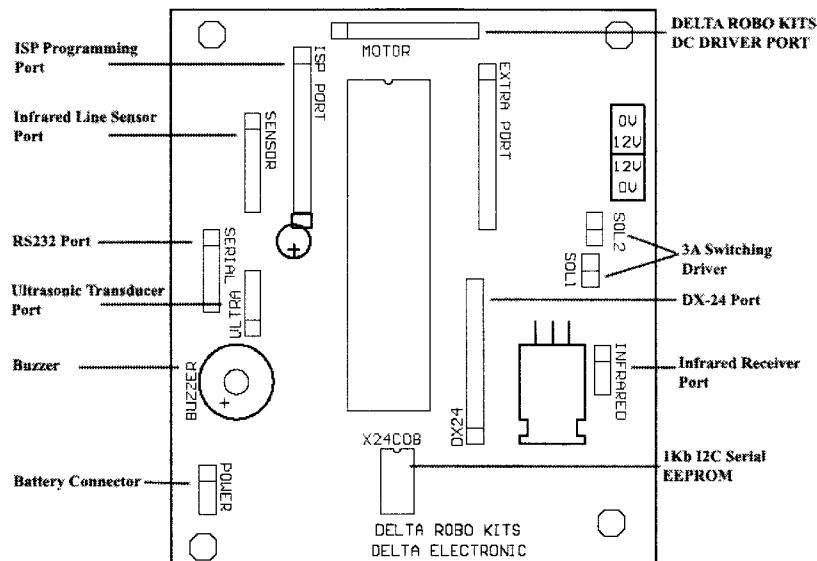
CH1AddressByte4    EQU    00
CH1AddressByte3    EQU    00
CH1AddressByte2    EQU    00
CH1AddressByte1    EQU    00
CH1AddressByte0    EQU    NomorRobot ;Sesuai dengan nomor joystick yang
digunakan
END

```

Bagian Otak

Seperti yang telah dijelaskan pada bagian sebelumnya, bagian ini berfungsi sebagai pemberi keputusan dan perintah agar bagian motor aktif menggerakkan mekanik-mekanik robot. Selain itu, otak juga harus memiliki unsur memori yang berfungsi sebagai media ingatan robot. Contohnya, robot yang menyimpan gerakan-gerakan yang dilakukan pada memori saat melewati labirin agar rute yang telah dilalui tercatat pada memorinya.

Modul Delta Robo CPU merupakan modul yang sesuai untuk aplikasi ini karena modul ini telah memiliki Port DC Driver untuk mengendalikan Motor DC, Infrared Line Sensor dan Ultrasonic Transducer untuk Input Sensor, Infrared Receiver Port dan DX-24 Port untuk Input Remote, dan 1 kb I2C Serial EEPROM sebagai memori. Selain itu, modul ini juga memiliki Port ISP untuk keperluan men-download program ke dalam mikrokontroler Delta Robo CPU



Gambar 6.18 Delta Robo CPU sebagai Otak Robot

Agar otak robot ini dapat berfungsi, terlebih dahulu harus dilakukan pemrograman pada mikrokontroler yang ada di dalamnya. Pada Delta

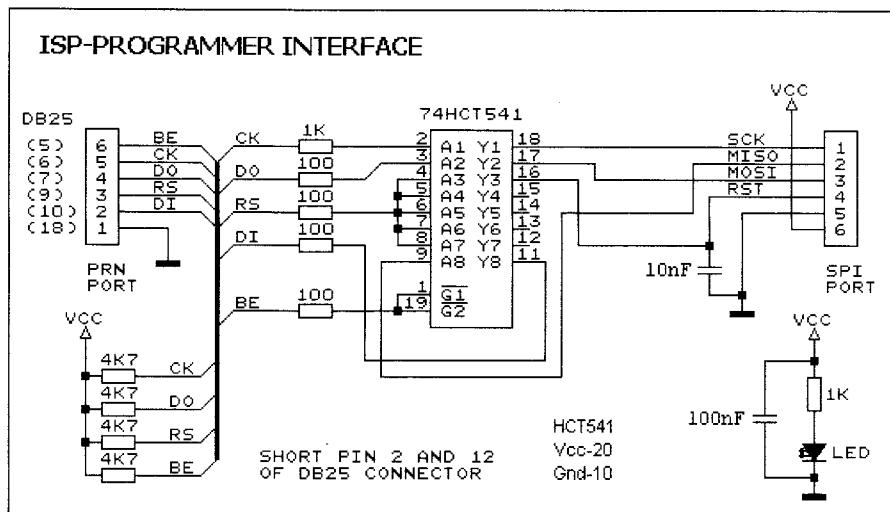
Robo CPU tersedia port ISP (In System Programming) yang berfungsi untuk keperluan tersebut.

Perangkat programmer dapat dihubungkan pada port ini dan proses download program secara ISP dapat dilakukan untuk mengisi Flash EPROM dari mikrokontroler AT89S51 yang ada dalam Delta Robo CPU.

Terdapat dua jenis proses penulisan program (download program) dari PC ke Delta Robo CPU atau sistem mikrokontroler AT89S51 seperti DST-51, yaitu dengan melalui Parallel Port PC atau Port USB.

Download Program melalui Parallel Port

Untuk membangun programmer melalui parallel port PC dapat dengan mudah dilakukan dengan merangkai beberapa komponen yang ada pada skema pada Gambar 6.19.



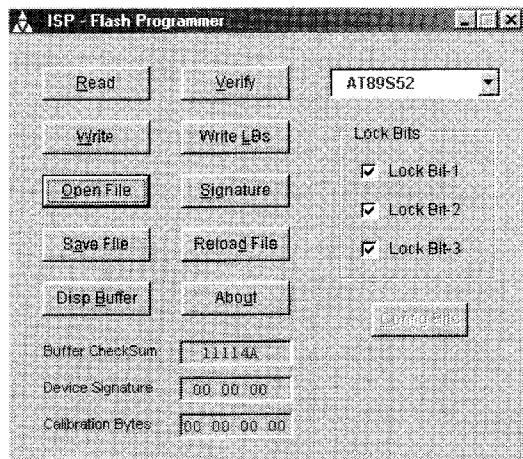
Gambar 6.19 Skema ISP Programmer buatan Asim Khan

- Jalankan Software ISP PGM3.0 buatan Asim Khan.
- Jalankan Patch ISP-XP.bat untuk Windows XP.
- Aktifkan power supply.



Bab 6: Bagian Pengendali Robot

- Tekan tombol Signature. Signature bit dari mikrokontroler akan tampil pada layar yang mengindikasikan bahwa PC telah terhubung dengan baik pada sistem mikrokontroler.
- Tekan Open File dan pilih file hex yang akan di-download. File ini dapat berupa file yang ada pada contoh-contoh program atau hasil assembly dari program rancangan pengguna melalui RIDE atau Delta Studio.
- Tekan Write. Akan tampil progress bar yang menunjukkan bahwa proses download sedang berjalan.
- Untuk me-reset sistem, pengguna dapat melakukannya melalui PC dengan menekan tombol signature. Apabila signature bit tampil, berarti proses reset berhasil.

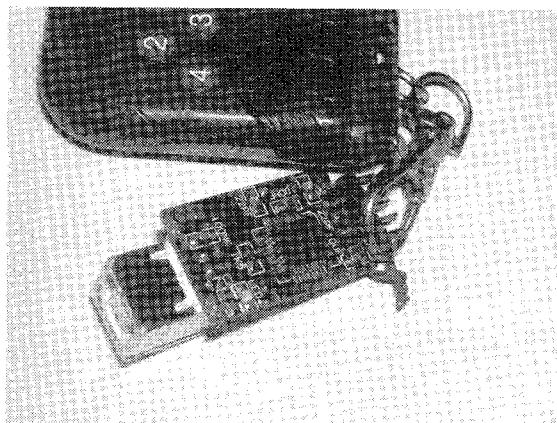


Gambar 6.20 Asim Khan ISP Programmer

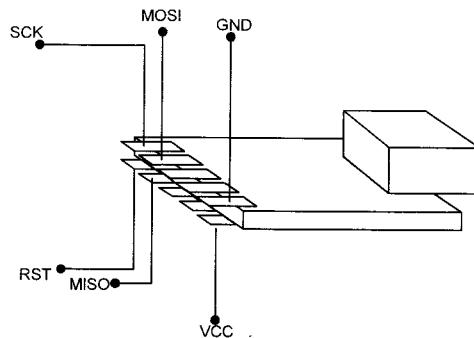
Download Program melalui USB Port

Dewasa ini parallel port sudah tidak lagi digunakan pada notebook dan bahkan pada PC sekalipun. Teknologi USB telah mendominasi port-port ini di berbagai jenis PC maupun notebook. Untuk itu dibutuhkan programmer yang dapat melakukan proses download melalui USB.

DU-ISP V2.0 adalah programmer USB yang dapat digunakan untuk men-download program ke berbagai macam mikrokontroler Atmel yang menggunakan teknik ISP.



Gambar 6.21 Modul DU-ISP V2.0



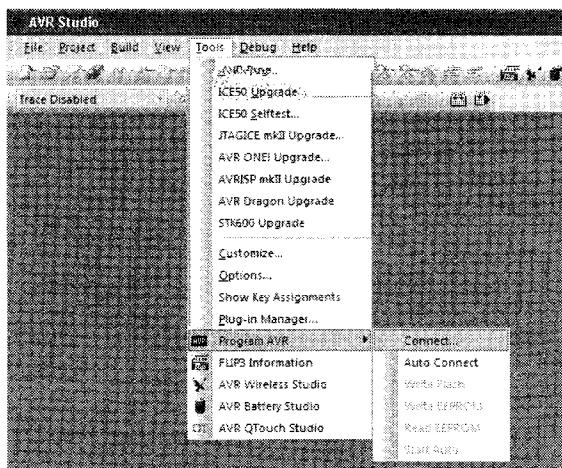
Gambar 6.22 Gambar Tata Letak Port DU-ISP V2.0

Berikut adalah langkah-langkah untuk men-download program dengan menggunakan DU-ISP V2.0

- Instalasikan program AVRStudio 4.16 dan USB Driver untuk DU-ISP V2.0 akan terinstal dengan sendirinya pada PC/notebook Anda.

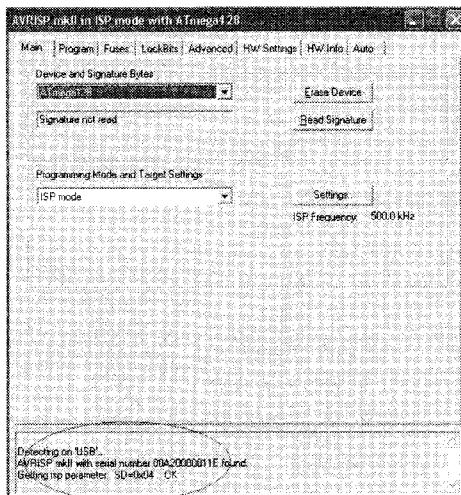
Bab 6: Bagian Pengendali Robot

- Buka program AVR Studio 4.16 dan pilih Auto Connect pada bagian Tools – Program AVR.



Gambar 6.23 AVRStudio V4.16

- AVR Studio akan mendeteksi port USB yang terhubung pada DU-ISP secara otomatis dan sekaligus mendeteksi mikrokontroler target yang terhubung.

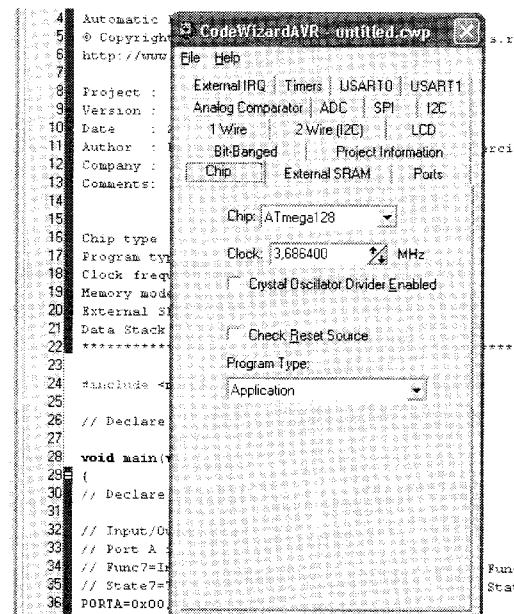


Gambar 6.24 AVRStudio mendeteksi mikrokontroler pada DU-ISP V2.0

- Pilih Tab Program dan load file hex pada bagian program.
 - Klik Program dan DU-ISP akan menuliskan program ke dalam mikrokontroler target.

Selain menggunakan AVRStudio, proses download program juga dapat dilakukan dengan menggunakan Software Code Vision. Software ini biasanya digunakan untuk source program dalam Bahasa C. Selain proses download, di sini juga dilakukan proses compile dari C ke bentuk hex terlebih dahulu. Berikut langkah-langkah yang dilakukan untuk penggunaan Code Vision.

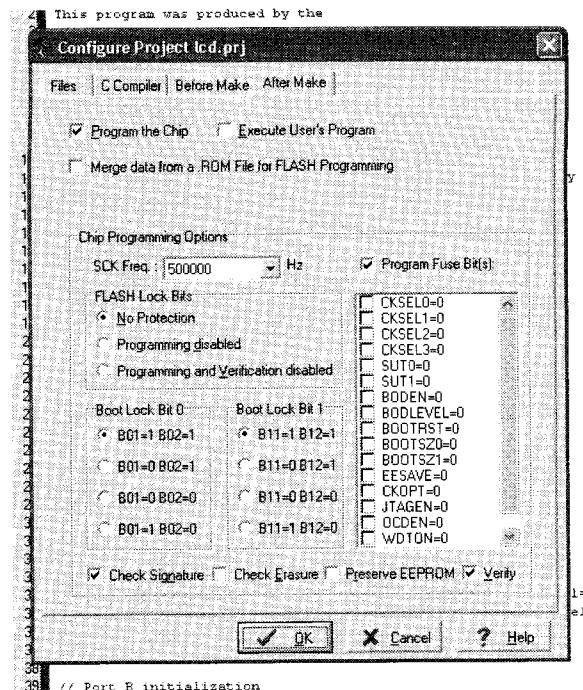
- Panggil Program Code Vision dan panggil salah satu project yang ada pada folder CVAVR/Examples sebagai contoh project.
 - Pastikan project yang diambil menggunakan mikrokontroler yang sama dengan target.
 - Apabila berbeda, pilih Tools kemudian Code Wizard Project dan sesuaikan jenis mikrokontroler yang digunakan.



Gambar 6.25 Wizzard Code Vision

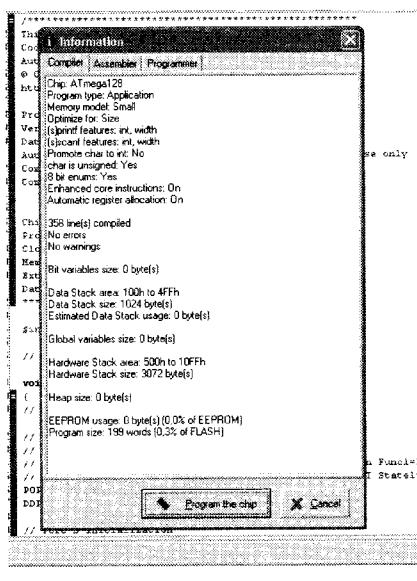
Bab 6: Bagian Pengendali Robot

- Pilih File, kemudian Generate save and exit. Simpan nama file baru sesuai ekstensi yang sudah ditentukan oleh Wizard.
 - Pilih Project, kemudian Configure dan After make. Aktifkan Program the Chip agar proses programming langsung dijalankan saat proses compile selesai.



Gambar 6.26 Konfigurasi Project

- Klik OK dan klik Project – Make sehingga proses compile dijalankan dan selanjutnya Code Vision akan menanyakan apakah proses programming dilakukan.



Gambar 6.27 Proses Download Program pada Code Vision

- Klik Program the Chip dan proses programming akan segera dilakukan.

Bagian Motor

Bagian ini merupakan bagian pelaksana dari perintah-perintah yang diberikan oleh otak robot. Berdasarkan fungsinya, terdapat beberapa macam motor yang biasa digunakan pada robot, yaitu motor DC untuk aplikasi yang membutuhkan kecepatan tinggi, motor stepper untuk aplikasi dengan akurasi tinggi, dan motor servo untuk gerakan-gerakan berupa gerakan sudut.

Dalam mengendalikan motor-motor tersebut, otak robot tidak dapat langsung mengakses motor, kecuali motor servo yang sudah memiliki antarmuka. Namun demikian, dengan menggunakan antarmuka servo controller, maka proses pengendali motor servo akan lebih mudah dilakukan.

Motor DC

Motor ini adalah motor yang paling sederhana untuk pengaktifannya. Hanya dengan memberikan tegangan DC, motor ini akan berputar secara kontinyu ke arah tertentu. Membalik arah putaran motor dapat dilakukan dengan mengubah polaritas arus yang mengalir pada motor.

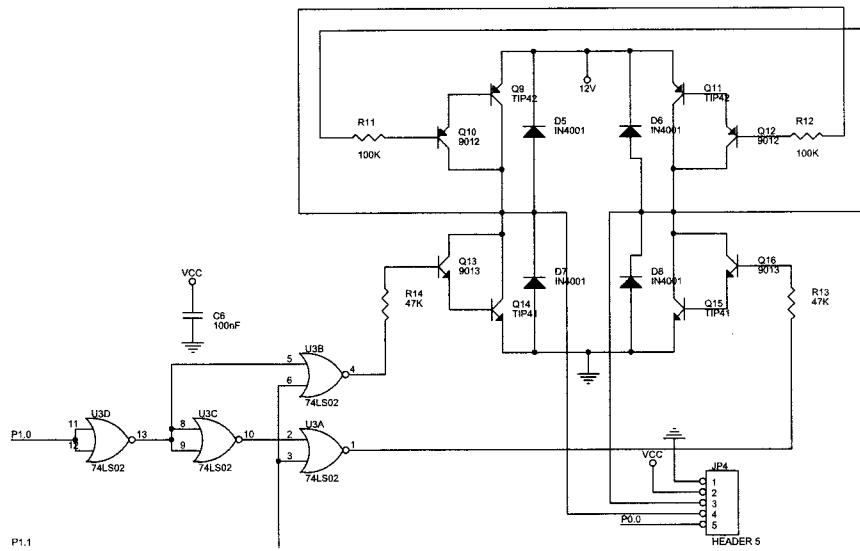
Motor DC biasanya mempunyai kecepatan putar yang cukup tinggi dan sangat cocok digunakan untuk roda robot yang membutuhkan kecepatan gerak yang tinggi. Juga dapat digunakan pada baling-baling atau kipas pada aplikasi robot pemadam api.

Untuk mengendalikan motor ini dibutuhkan sebuah rangkaian yang disebut rangkaian Half Bridge. Rangkaian ini akan membuat arus mengalir pada motor melalui dua kutubnya secara bergantian sesuai arah yang diinginkan. Gambar 6.28 menunjukkan rangkaian Half Bridge yang dilengkapi dengan proteksi hubung singkat. Rangkaian ini dapat dijumpai pada Modul Delta DC Driver yang memiliki dua rangkaian Half Bridge sehingga dapat mengendalikan dua buah motor DC.

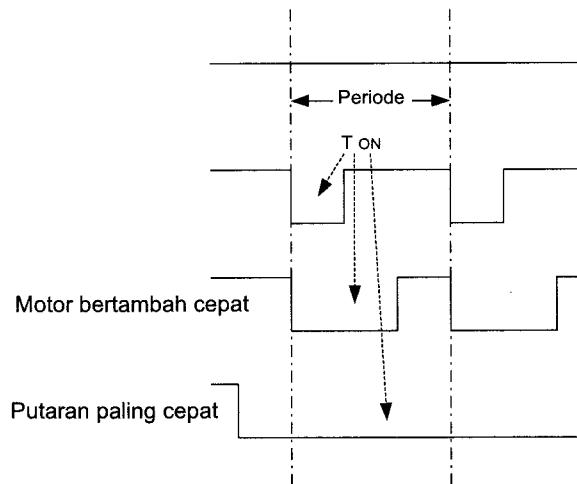
Apabila transistor PNP yang kiri aktif, transistor NPN kanan juga aktif sedangkan PNP kanan dan NPN kiri non-aktif. Arus akan mengalir dari sumber daya positif ke kaki 3 JP4 yang terhubung ke kutub motor dan diteruskan hingga kaki 4 JP4 ke sumber daya negatif atau ground. Sebaliknya pada saat transistor PNP kanan yang aktif, maka transistor NPN kiri juga aktif sedangkan PNP kiri dan NPN kanan non-aktif. Arus akan mengalir dari sumber daya ke kaki 4 JP4 dan terus menuju ke sumber daya negatif melalui kaki 3. Dioda berfungsi untuk mencegah adanya tegangan reverse akibat induksi motor.

Pada rangkaian Half Bridge, hal yang tidak boleh terjadi adalah keempat bagian transistor, yaitu NPN kiri, NPN kanan, PNP kiri, dan PNP kanan aktif bersamaan. Hal ini akan menghubungkan sumber daya positif dan negatif. Untuk mencegah kondisi ini, rangkaian gerbang logika yang dibentuk oleh IC 74LS02 diatur sehingga NPN kiri dan NPN kanan aktif bergantian. Hal ini ditentukan oleh kondisi logika pada P1.0 sebagai penentu arah gerakan motor. Sedangkan P1.1 berfungsi untuk mengatur apakah motor dalam keadaan aktif atau tidak. Bila kondisi logika P1.0 adalah logika 0, maka keluaran U3B akan berlogika 0 pula selama P1.1 aktif (berlogika 0). Hal ini akan mengakibatkan transistor NPN kiri non-aktif. Sedangkan keluaran U3C akan berlogika 1 yang mengakibatkan keluaran U3A juga berlogika 1 dan transistor NPN

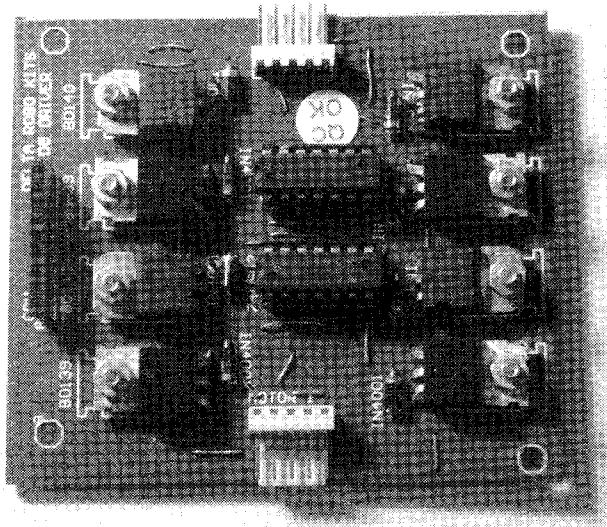
kanan aktif. P1.1 juga berfungsi untuk mengatur kecepatan gerak motor. Dengan membangkitkan PWM pada P1.1, maka kecepatan gerak motor akan dapat diatur melalui bagian ini.



Gambar 6.28 Skema Half Bridge pada Delta DC Driver



Gambar 6.29 Bentuk Sinyal PWM



Gambar 6.30 Modul Delta DC Driver dengan Dual Half Bridge

Motor Stepper

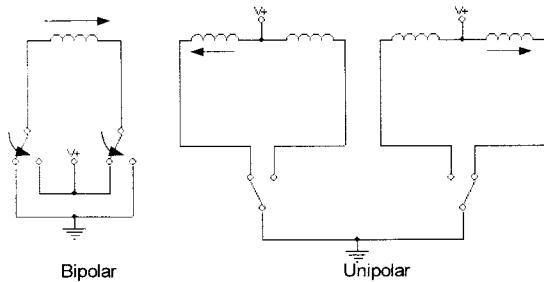
Pada dasarnya ada 2 jenis motor stepper, yaitu bipolar dan unipolar. Sebuah motor stepper berputar 1 step apabila terjadi perubahan arus pada koilnya yang mengubah pole magnetik di sekitar pole stator. Perbedaan utama antara Bipolar dan Unipolar adalah sebagai berikut :

Bipolar

- Arus pada koil dapat berbolak balik untuk mengubah arah putar motor.
- Lilitan motor hanya satu dan dialiri arus dengan arah bolak-balik.

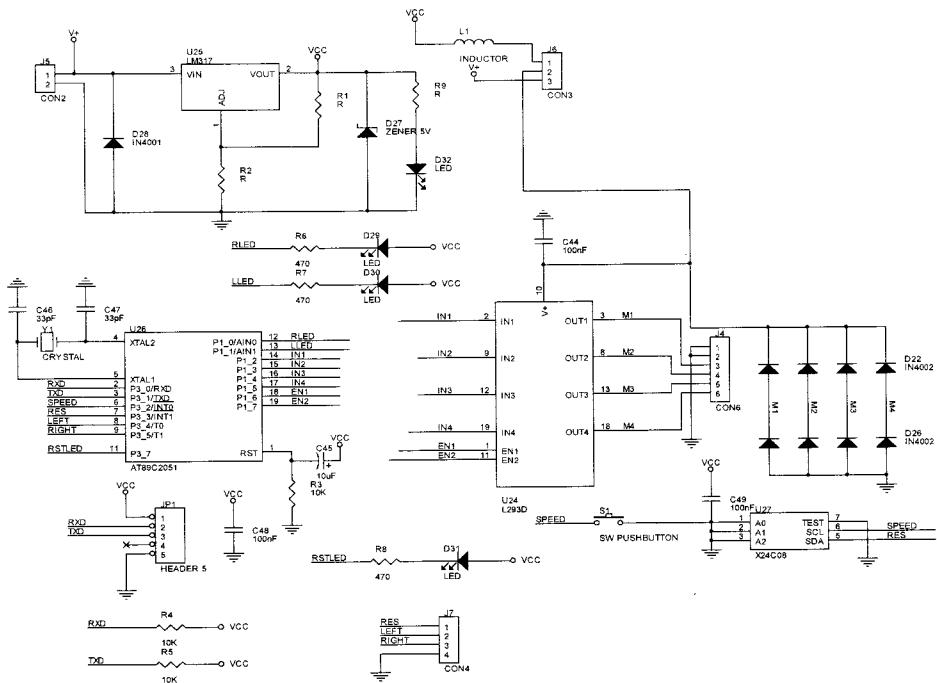
Unipolar

- Arus mengalir satu arah dan perubahan arah putar motor tergantung lilitan (koil) yang dialiri arus.
- Lilitan terpisah dalam 2 bagian dan masing-masing bagian hanya dilewati arus dalam satu arah saja.



Gambar 6.31 Perbedaan Motor Stepper Bipolar dan Unipolar

Dengan menggunakan IC L293D, proses kendali motor stepper bipolar maupun unipolar dapat dilakukan. Gambar 6.32 menampilkan skema dari Modul SST-06 Subsystem Stepper Interface, yaitu sebuah sistem pengendali motor stepper yang telah dilengkapi mikrokontroler yang mengatur gerakan motor berdasarkan perintah serial dari UART ataupun manual.



Gambar 6.32 Diagram Skematik Pengendali Stepper SST-06 V2.0

Bab 6: Bagian Pengendali Robot

Modul SST-06 V2.0 ini selain dapat mengendalikan motor stepper melalui perintah-perintah serial juga dapat mengendalikan motor stepper secara manual dengan menghubungkan Left atau Right pada J7 ke Ground. Dengan menghubungkan Res pada J7 ke Ground, maka motor akan bergerak ke Home Position (posisi awal).

Sakelar push button S1 berfungsi untuk mengubah kecepatan motor secara manual. Perubahan kecepatan ini dapat dilihat dari perubahan kecepatan kedip LED saat motor bekerja. Memori AT24C08 akan menyimpan nilai kecepatan tersebut sehingga pada saat power diaktifkan kembali, kecepatan gerak yang digunakan adalah kondisi terakhir terjadi sebelum power dimatikan.

Pengaturan gerakan melalui UART dilakukan dengan mengirimkan perintah-perintah yang akan dilaksanakan oleh Modul SST-06 sesuai Tabel 5 berikut.

Tabel 5 Protokol SST-06

Pengiriman data dari PC/Master ke SST-06			
Byte		Nilai	Keterangan
00	Header	1E	Awal paket data
01	Destination ID	06	ID Subsystem
02	Destination Number	01-FF	No urut Sub System
03	Source ID	00	ID Pengirim 00 = Master (PC/Microcontroller)
04	Source Number	01-FF	No urut Master
05	Length		Panjang paket data
06	Perintah	01	Start Motor kontinyu
		02	Stop Motor
		03	Start Motor ke posisi tertentu
		04	Set Kecepatan Motor
		05	Set Home Position
07+n	Checksum	00-FF	Total mulai 1E hingga checksum = 00

Perintah Start Motor Kontinyu			
Byte		Nilai	Keterangan
07	Set Direction	00/01	00 CCW/01 CW

Perintah Stop Motor			
Byte		Nilai	Keterangan
	-		-

Perintah Start Motor ke Posisi Tertentu			
Byte		Nilai	Keterangan
07	Set Direction	00/01	00 CCW/01 CW
08	Jumlah putaran	00-FF	
09	Jumlah step	00-200	Untuk sudut 1.8 derajat/step

Perintah Set Kecepatan Motor			
Byte		Nilai	Keterangan
07	Kecepatan	00-100	
Perintah Set Home Position			
Byte		Nilai	Keterangan
	-		-

Pengiriman data dari SST-06 ke PC/Master			
Byte		Nilai	Keterangan
00	Header	1E	Awal paket data
01	Destination ID	00	ID Pengirim 00 = Master (PC/Microcontroller)
02	Destination Number	01-FF	No urut Master
03	Source ID	14	ID Sub System
04	Source Number	01-FF	No urut Sub System
05	Length		Panjang paket data
06	Perintah	0B	Info kecepatan motor
07	Isi Data		
07+n	Check Sum		Total dari keseluruhan data = 0

Info kecepatan motor			
Byte		Nilai	Keterangan
07	Kecepatan Motor 1	00 - FF	step per second

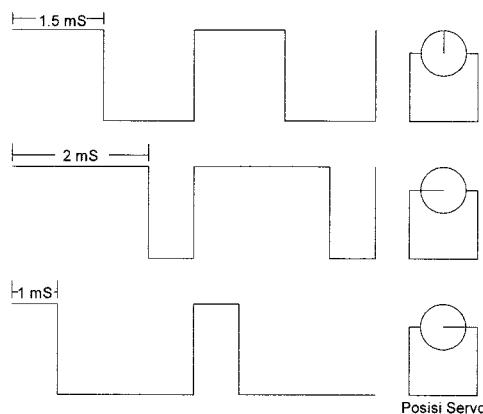
08	Kecepatan Motor 2	00 - FF	step per second
----	-------------------	---------	-----------------

ACK			
Byte		Nilai	Keterangan
07	Status ACK	45 / 4F	45 = E atau error da 4F = O atau OK

Motor Servo

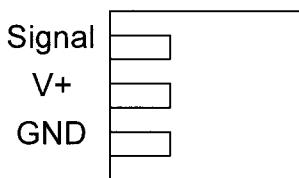
Berbeda dengan motor DC dan motor stepper, motor servo adalah sebuah motor dengan sistem umpan balik tertutup di mana posisi dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri atas sebuah motor DC, serangkaian gear, potensiometer, dan rangkaian kontrol.

Potensiometer berfungsi untuk menentukan batas sudut dari putaran servo. Sedangkan sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor. Seperti tampak pada gambar, dengan pulsa 1.5 ms pada periode selebar 2 ms, maka sudut dari sumbu motor akan berada pada posisi tengah. Semakin lebar pulsa OFF, akan semakin besar gerakan sumbu ke arah jarum jam. Semakin kecil pulsa OFF, akan semakin besar gerakan sumbu ke arah yang berlawanan dengan jarum jam.



Gambar 6.33 Teknik PWM untuk mengatur sudut motor servo

Motor servo biasanya hanya bergerak mencapai sudut tertentu saja dan tidak kontinyu seperti motor DC maupun motor stepper. Walau demikian, untuk beberapa keperluan tertentu, motor servo dapat dimodifikasi agar bergerak kontinyu.



Gambar 6.34 Pin Out Kabel Motor Servo

Pada robot, motor ini sering digunakan untuk bagian kaki, lengan, atau bagian-bagian lain yang mempunyai gerakan terbatas dan membutuhkan torsi cukup besar.

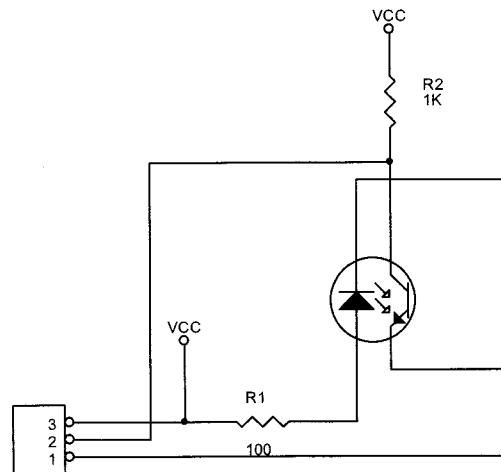
Seperti telah dijelaskan di atas, pengaturan gerakan motor servo dilakukan dengan memberikan pulsa PWM yang dapat dibangkitkan oleh mikrokontroler. Untuk aplikasi robot seringkali dibutuhkan lebih dari sebuah motor servo sehingga program pada mikrokontroler akan cukup sibuk mengatur kestabilan nilai PWM. Untuk mempermudah aplikasi ini, biasanya digunakan modul servo controller DSR-08. Modul ini dapat mengendalikan 8 buah motor servo secara terpisah, baik sudut maupun kecepatannya.

Proses kendali motor servo dilakukan dengan mengirimkan perintah-perintah protokol melalui port UART seperti yang dijelaskan pada Tabel 5. Dengan begitu, mikrokontroler pada otak robot hanya perlu mengirimkan sebuah perintah untuk mengatur sudut motor servo saja. Proses PWM selanjutnya akan ditangani DSR-08 dan otak robot dapat melakukan kegiatan yang lain.

Selain proses kendali melalui UART, DSR-08 juga dapat dihubungkan pada 8 buah rotary encoder sehingga dapat memantau berapa lubang encoder yang dilewati setiap kali perintah perubahan sudut dilakukan. Sebagai contoh, pada saat otak robot memerintahkan DSR-08 untuk menggerakkan motor servo 1 dari suatu sudut ke sudut yang lain dan ternyata melewati 5 buah lubang encoder, maka DSR-08 akan menyimpan nilai 5 pada memori penyimpanan nilai encoder bagi motor servo 1. Apabila suatu saat otak robot menanyakan, DSR-08 dapat

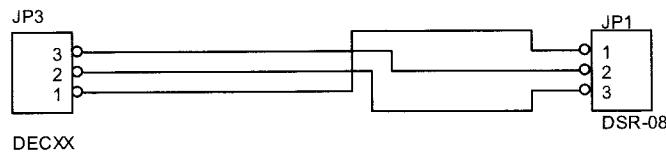
Bab 6: Bagian Pengendali Robot

memberikan informasi tersebut. Untuk menghitung jumlah lubang encoder ini dapat digunakan rangkaian DEC-01 seperti pada Gambar 6.35.



Gambar 6.35 Diagram Skematik DEC-01

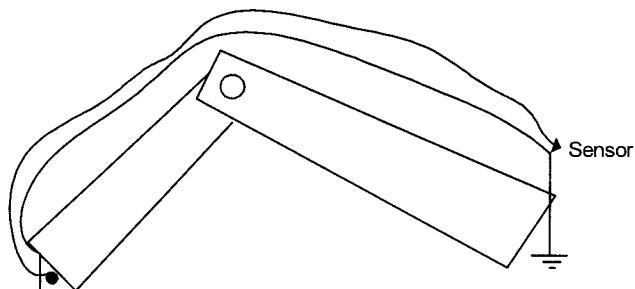
Hubungan antara DEC-01 dan DSR-08 dapat dilakukan melalui konektor 3 pin pada masing-masing modul dengan pengkabelan seperti pada Gambar 6.36.



Gambar 6.36 Pengkabelan DEC-01 dan DSR-08

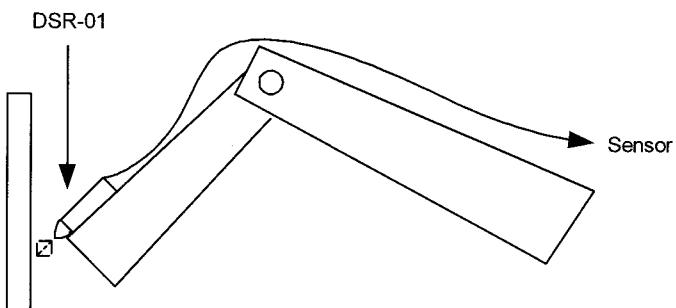
Untuk mencegah kerusakan akibat kegagalan mekanis atau kesalahan pengaturan gerak servo, misalnya apabila pengguna mengatur servo agar bergerak 20 step ke CCW dan pada saat itu terdapat penghalang yang mengakibatkan mekanik terganjal pada saat step ke-10, maka

sensor ini akan aktif dan memerintahkan DSR-08 untuk bergerak ke arah yang telah diprogram sebelumnya hingga sensor tidak mendeteksi memaksa membebaskan mekanik dari ganjalan tersebut. Sensor ini aktif dengan adanya logika 0 di bagian inputnya. Pengguna dapat menggunakan limit switch yang menghubungkan input ini dengan ground saat mekanik membentur sesuatu.



Gambar 6.37 Hubungan Sensor Limit Switch dan DSR-08

Pada aplikasi-aplikasi tertentu, benturan mekanis tidak diizinkan. Oleh karena itu, sensor infrared DSF-01 dapat digunakan sebagai masukan sensor bagi DSR-08.

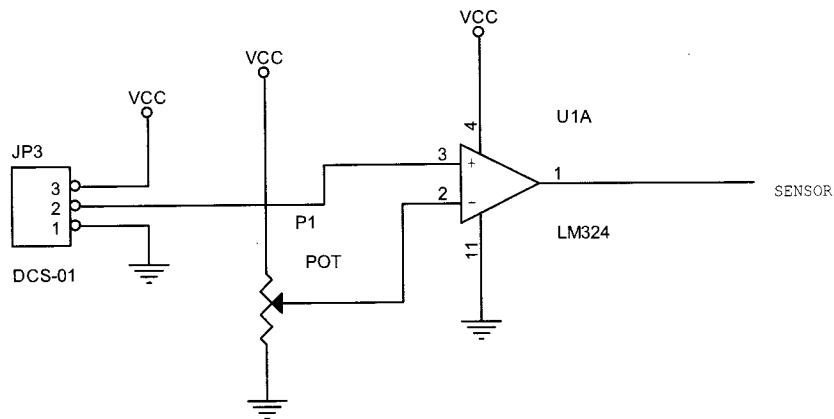


Gambar 6.38 Pemasangan DSF-01 pada DSR-08

Untuk aplikasi di mana pemasangan sensor di bagian mekanik tidak dimungkinkan, pengguna dapat menggunakan sensor arus DCS-01 yang diletakkan pada bagian power dari servo sehingga pada saat mekanik tertahan sesuatu dan penggunaan arus melebihi kapasitas,

Bab 6: Bagian Pengendali Robot

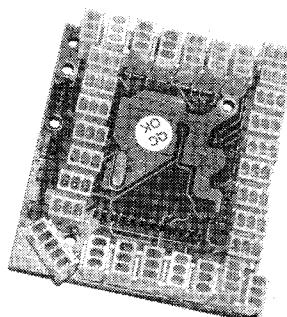
maka DSR-08 akan memerintahkan servo secara otomatis bergerak ke arah sebaliknya sehingga sensor tidak lagi aktif.



Gambar 6.39 Skema DCS-01 dan DSR-08

Potensio dalam hal ini berfungsi untuk mengatur besar nilai arus yang akan mengaktifkan sensor.

Selain pengaturan gerak ke sudut tertentu, servo controller juga dapat diperintahkan untuk bergerak secara relatif dari sudut tertentu. Hal ini akan meringankan beban otak robot karena tidak perlu mencatat atau mengingat posisi sebelumnya. Otak robot hanya memerintahkan servo controller ini untuk bergerak ke arah CW atau CCW sebanyak sekian step saja.



Gambar 6.40 Modul DSR-08

Tabel 7 Protokol DSR-08

Pengiriman data dari PC/Master ke Servo Controller			
Byte		Nilai	Keterangan
00	Header	1E	Awal paket data
01	Destination ID	11	ID Subsystem
02	Destination Number	01-FF	No urut Sub System
03	Source ID	00	ID Pengirim 00 = Master (PC/Microcontroller)
04	Source Number	01-FF	No urut Master
05	Length		Panjang paket data
06	Perintah	01	Perintah gerakkan servo
		02	Perintah minta posisi servo
		03	Perintah minta nilai encoder
		04	Perintah mengatur arah saat sensor mendeteksi kegagalan mekanis
		05	Perintah menggerakkan servo secara relatif
07+n	Checksum	00-FF	Total mulai 1E hingga checksum = 00

Perintah gerakkan servo (01)			
Byte		Nilai	Keterangan
07	Nomor Motor	01-FF	Nomor urut motor
08	Step Sudut Servo	01-25	
09	Kecepatan Servo	80-FF	Kecepatan terendah delay FF atau 255 Kecepatan tertinggi delay 80h atau 128

Perintah Minta Posisi Servo (02)			
Byte		Nilai	Keterangan
07	Nomor motor	01-FF	Nomor urut motor
08	Step Sudut Servo	01-25	Informasi step sudut servo saat ini

Perintah Minta Nilai Encoder (03)			
Byte		Nilai	Keterangan
07	Nomor Motor	01-FF	Nomor urut motor
08	Nilai Encoder	00-xx	Xx = jumlah lubang pada encoder

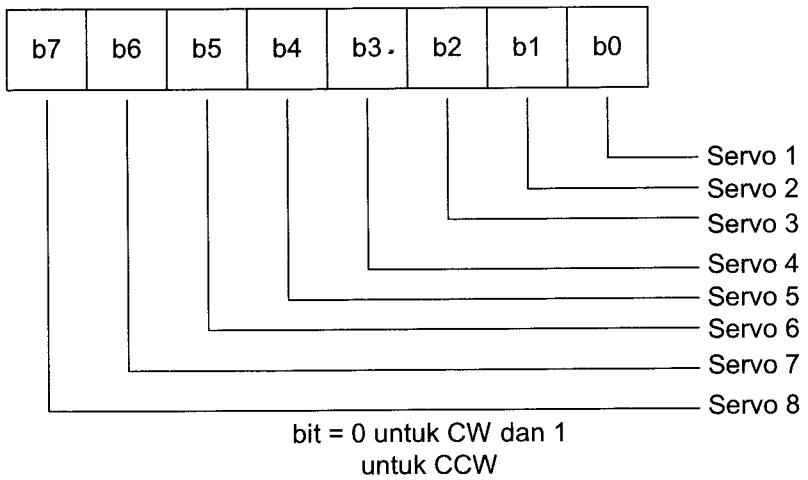
Bab 6: Bagian Pengendali Robot

Perintah Atur Arah Saat Sensor Mendeteksi Kegagalan Mekanis (04)			
Byte		Nilai	Keterangan
07	Arah Motor-motor	00-FF	Setiap bit mewakili posisi setiap servo (lihat Gambar 6.41)

Perintah Mengatur Gerakan Relatif (05)			
Byte		Nilai	Keterangan
07	Nomor Motor	01-FF	Nomor urut motor
08	Arah gerakan servo	00/01	00 = CW / 01 = CCW
09	Step Relatif	01-25	Step relatif terhadap step sudut saat itu

Perintah Stop Servo (06)			
Byte		Nilai	Keterangan
07	Nomor Motor	01-FF	Nomor urut motor

Pengiriman data dari Servo Controller ke PC/Master			
Byte		Nilai	Keterangan
00	Header	1E	Awal paket data
01	Destination ID	00	ID Master (PC/Microcontroller) = 00
02	Destination Number	01-FF	No urut Sub System
03	Source ID	11	ID SubSystem
04	Source Number	01-FF	No urut Master
05	Length		Panjang paket data
06	Perintah	01	Informasi posisi servo
		02	Informasi nilai encoder
07+n	Checksum	00-FF	Total mulai 1E hingga checksum = 00



Gambar 6.41 Register Pengatur Arah Saat Sensor Mendeteksi Kegagalan Mekanis

Latihan

1. Tuliskan program untuk mengaktifkan LED pada P1.0 saat tombol 1 pada remote Sony ditekan.
2. Pasang DX-24 pada Port 2 DST-51 dan tuliskan program untuk menampilkan nilai-nilai data yang dikirimkan joystick ke layar LCD M1632.
3. Gambarkan diagram skematik pada gambar 6.28 dengan menggunakan IC L293D sebagai pengantinya.
4. Tuliskan protokol untuk menggerakkan SST-06 ke kiri dengan kecepatan 10 step per detik.
5. Tuliskan protokol untuk menggerakkan motor servo kedua ke step 20 dengan delay 128 ms untuk kecepatannya.

Bab 7

Robot Laba-laba dengan 6 kaki

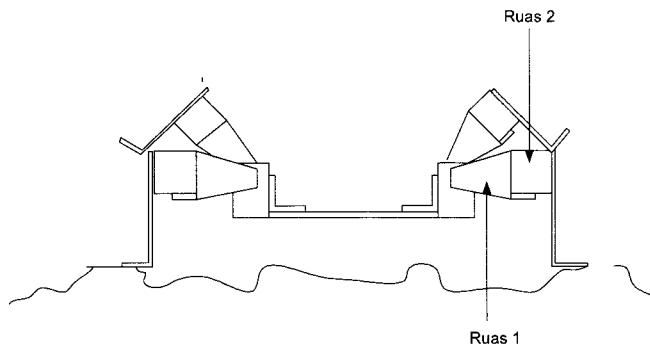
Pendahuluan

Mekanik laba-laba adalah bentuk mekanik robot yang paling praktis untuk dikembangkan pada aplikasi robotik di segala medan. Pada bab ini akan dijelaskan cara membuat robot laba-laba ini.

Robot Laba-Laba

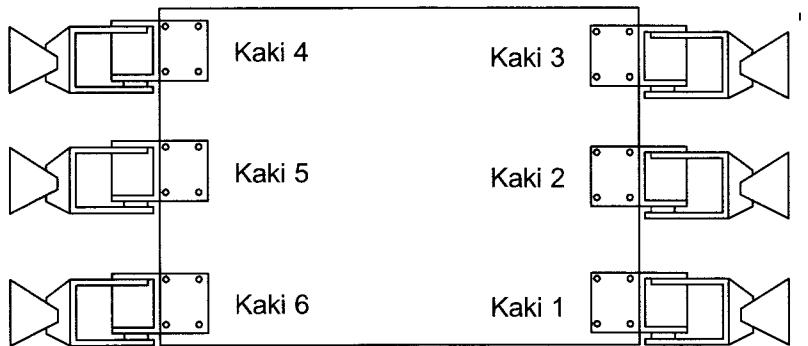
Dibandingkan dengan robot beroda maupun bipedal, mekanik ini mempunyai tingkat stabilitas yang lebih tinggi baik di medan datar maupun di medan kasar atau tidak beraturan. Enam kaki pada mekanik laba-laba tersebut berfungsi menjaga posisi robot agar stabil di medan-medan tak beraturan. Setiap kaki memiliki tiga ruas sehingga dapat bergerak melampaui halangan-halangan kecil.

Bab 7: Robot Laba-Laba dengan 6 Kaki



Gambar 7.1 Robot Laba-laba dengan kaki 2 dan 5 (kaki-kaki tengah) terangkat

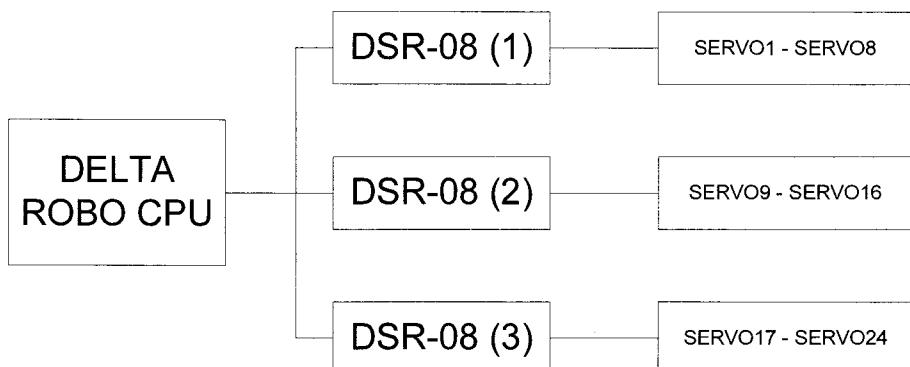
Gerakan setiap ruas dari kaki-kaki tersebut merupakan gerakan sudut sehingga motor servo adalah alternatif terbaik dalam hal ini. Seperti telah dijelaskan pada bagian pengendali servo, posisi sudut dari motor ini diatur oleh teknik PWM.



Gambar 7.2 ke enam kaki laba-laba tampak atas

Dengan menggunakan enam kaki dan dua ruas, maka dibutuhkan setidaknya 12 motor servo yang harus diatur PWM-nya. Pengaturan PWM merupakan fungsi pengaturan timing yang sangat kritis karena sedikit perubahan akan mengakibatkan pergeseran posisi gerakan sudut ruas. Untuk itu, agar kinerja otak robot tidak terlalu berat dengan adanya pengaturan PWM ini, maka digunakan beberapa servo controller yang berfungsi sebagai otak tambahan dari robot.

Fungsi otak tambahan ini sebetulnya seperti syaraf motorik dari manusia yang menerima perintah dari otak dan kadang-kadang juga bereaksi langsung tanpa menunggu respon dari otak pada kasus-kasus khusus. Pada pengendali servo DSR-08, hal ini terjadi bila salah satu input sensor aktif. Modul ini akan segera merespon dengan menggerakkan servo ke arah berlawanan hingga sensor tidak aktif lagi tanpa menunggu perintah dari otak. Hal ini berfungsi untuk mencegah kerusakan mekanik robot apabila mekanik tersangkut sesuatu dan sensor aktif, maka servo controller akan segera membebaskan mekanik tersebut dengan menggerakkan ke arah sebaliknya hingga sensor tidak mendeteksi adanya keadaan tersangkut tersebut.



Gambar 7.3 Blok Diagram Sistem

Delta Robo CPU selaku otak dari robot laba-laba hanya mengirimkan perintah-perintah untuk menggerakkan motor-motor servo pada posisi yang telah ditentukan oleh otak tersebut. DSR-08 akan menerima dan membangkitkan PWM pada motor servo yang digerakkan sesuai informasi sudut yang diberikan.

Setelah menerima informasi dari Delta Robo CPU, DSR-08 juga mengirimkan konfirmasi berupa ACK ke Delta Robo CPU yang merupakan indikasi bahwa perintah telah diterima dengan baik dan akan segera dilaksanakan. Dengan proses ini, kinerja otak robot (Delta Robo CPU) tidak terlalu berat dan dapat digunakan untuk aplikasi-aplikasi yang lain.

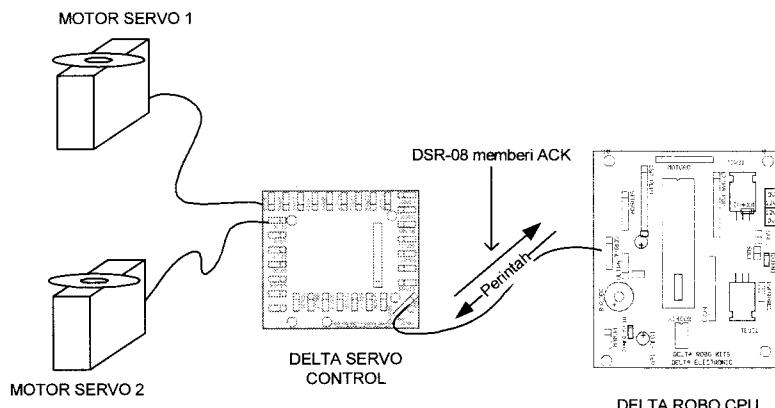
Bab 7: Robot Laba-Laba dengan 6 Kaki

Dengan Modul DSR-08, maka otak robot dapat mengirimkan perintah menggerakkan motor servo, meminta informasi lebar pulsa PWM, meminta informasi sudut servo (dengan bantuan rotary encoder), menghentikan motor servo, dan lain-lain. Pada umumnya, yang paling sering digunakan adalah perintah menggerakkan motor servo. Otak robot akan mengirimkan perintah tersebut dalam sebuah paket yang berisi variabel-variabel sebagai berikut:

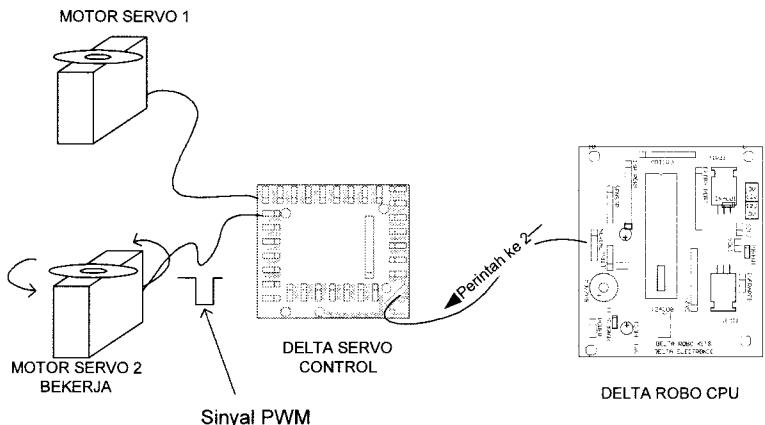
- Nomor urut DSR-08 yang menentukan modul DSR-08 mana yang diakses.
- Nomor servo yang diakses.
- Posisi servo atau lebar pulsa PWM yang menentukan sudut motor servo.
- Frekuensi PWM yang menentukan kekuatan gerak motor servo

Lebih detail mengenai variabel-variabel ini dapat dilihat di bagian protokol pada penjelasan tentang bagian pengendali servo.

Setelah perintah diterima, DSR-08 akan memberikan informasi pada otak bahwa perintah telah diterima dengan baik dan akan dikerjakan. Otak robot mengetahui bahwa DSR-08 telah menerima perintah dengan baik dan sedang mengerjakan perintah tersebut sambil menunggu perintah selanjutnya.



Gambar 7.4 Modul DSR-08 memberi informasi siap pada Delta Robo CPU



Gambar 7.5 Perintah kedua dikirimkan saat DSR-08 sedang mengerjakan perintah pertama

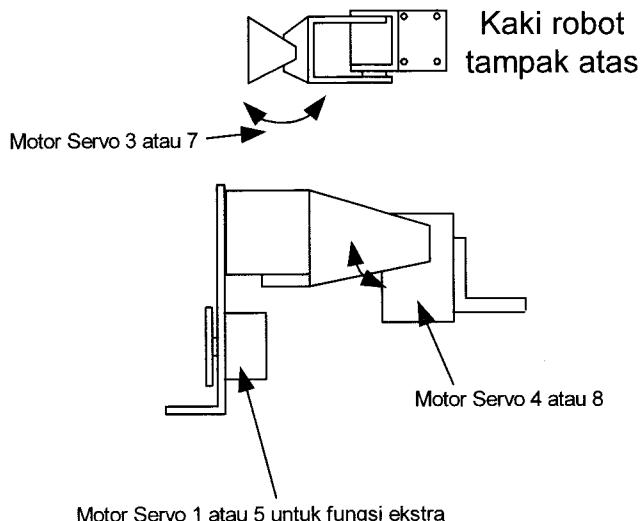
Walaupun DSR-08 sedang mengerjakan perintah pertama, dalam gambar di atas adalah perintah menggerakkan motor servo, tetapi DSR-08 sudah dapat menerima perintah berikutnya. Sebagai contoh, jika diberi perintah menggerakkan motor servo 1, DSR-08 akan langsung mengerjakannya sehingga motor servo 1 dapat menyusul bergerak tanpa menghentikan gerakan motor servo 2 yang masih belum selesai.

Aplikasi Robot Laba-laba ini dibangun dengan menggunakan perlengkapan berikut:

- 1 Modul Delta Robo CPU sebagai otak.
- 3 Modul DSR-08 sebagai servo controller.
- 6 Motor Servo Hitec HS-805BB.
- 6 Motor Servo Hitec HS-645MG.
- 6 Motor Servo Hitec HS-422HD (untuk fungsi ekstra).
- 6 Spider Leg Mechanic.
- Battery pack 6V/18.000 mAH.

Pengaturan Gerak Ruas-Ruas Kaki

Untuk mengendalikan 18 motor servo pada mekanik laba-laba ini digunakan 3 buah DSR-08. Masing-masing DSR-08 akan mengendalikan 8 motor servo.



Gambar 7.6 Posisi-posisi Motor Servo Pada Mekanik Kaki Laba-laba

Ke-18 motor servo tersebut tersebar pada keenam kaki laba-laba. Masing-masing kaki memiliki 2 buah motor servo penggerak ruas dan 1 motor servo ekstra. Gambar di atas menunjukkan nomor-nomor motor servo pada setiap kaki laba-laba. Untuk kaki ganjil (1, 3, dan 5) digunakan motor servo 1 hingga 4 dari setiap DSR-08. Sedangkan untuk kaki genap (2, 4, dan 6) digunakan motor servo 5 hingga 8 dari setiap DSR-08.

Motor Servo 4 atau motor servo 8 berfungsi untuk mengangkat dan menurunkan kaki laba-laba ruas pertama. Ruas ini adalah ruas yang menopang tubuh laba-laba saat berdiri sehingga motor servo yang digunakan adalah motor servo yang memiliki torsi paling besar, yaitu HS-805BB dengan torsi 24,7 kg-cm.

Untuk ruas kedua yang bergerak secara horizontal, untuk mengatur maju mundurnya kaki dikendalikan oleh motor servo 3 atau motor servo 7. Walaupun tidak sekuat ruas pertama, namun ruas ini juga membu-

tuhkan motor servo yang cukup kuat tetapi dengan ukuran fisik yang kecil. HS-645MG adalah sebuah motor servo dengan dimensi yang kecil namun memiliki torsi 9,6 kg dan metal gear.

Motor servo 1 atau motor servo 4 berfungsi sebagai motor servo ekstra yang dapat digunakan untuk fungsi-fungsi tambahan tertentu, misalnya menarik suction pada aplikasi gerakan memanjang.

Tabel 1 Nomor Urut Motor Servo dan Fungsinya

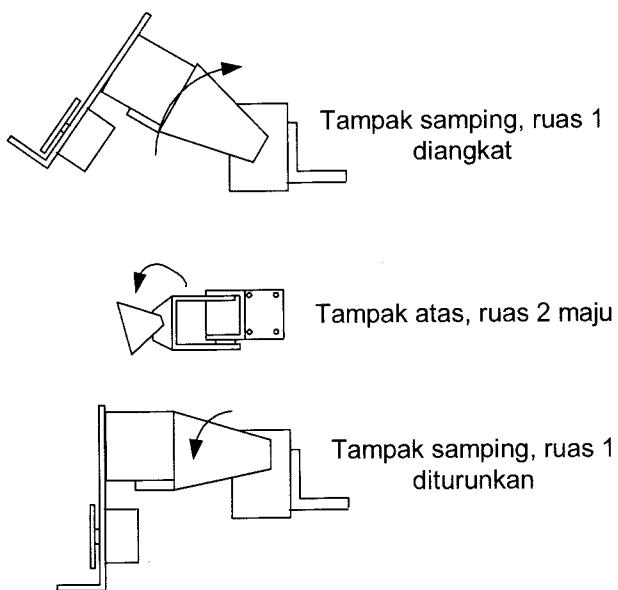
Nomor urut DSR-08	Nomor Kaki Laba-laba	Nomor Motor Servo	Keterangan
1	1	1	Fungsi Ekstra
1	1	3	Ruas 2
1	1	4	Ruas 1
1	2	5	Fungsi Ekstra
1	2	7	Ruas 2
1	2	8	Ruas 1
2	3	9	Fungsi Ekstra
2	3	11	Ruas 2
2	3	12	Ruas 1
2	4	13	Fungsi Ekstra
2	4	15	Ruas 2
2	4	16	Ruas 1
3	5	17	Fungsi Ekstra
3	5	19	Ruas 2
3	5	20	Ruas 1
3	6	21	Fungsi Ekstra
3	6	23	Ruas 2
3	6	24	Ruas 1

Gerakan-Gerakan Laba-Laba

Secara garis besar, gerakan laba-laba ini dibedakan menjadi gerakan kaki dan gerakan tubuh. Gerakan-gerakan kaki di sini terdiri atas gerak langkah maju, mundur, dan lurus. Sedangkan gerakan tubuh adalah gerakan untuk berdiri-turun, gerakan badan ke kiri maju-mundur dan gerakan badan ke kanan maju-mundur.

Gerakan Langkah Maju

Gerakan langkah maju adalah gerakan mengangkat kaki laba-laba dan menggerakkannya ke arah depan hingga meletakkan kaki kembali ke lantai untuk menopang tubuhnya.



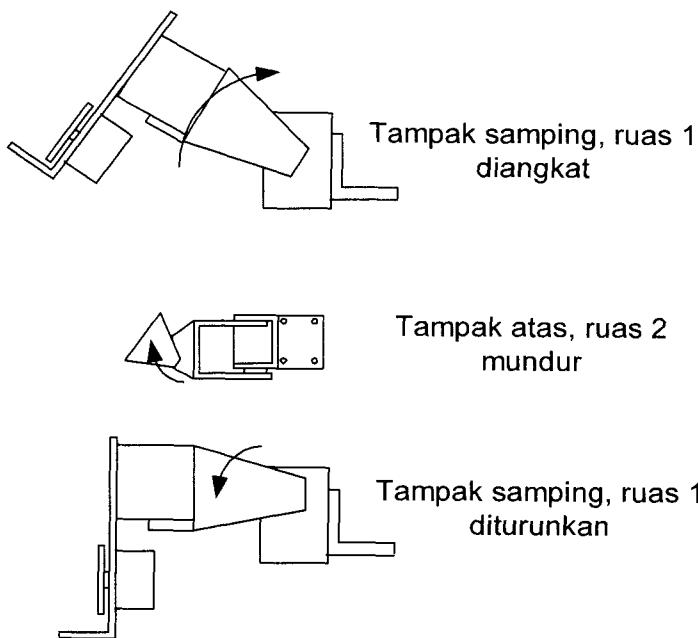
Gambar 7.7 Gerakan langkah maju

Pada fungsi ini, motor servo HS-805BB berlaku sebagai penggerak ruas satu dan motor servo HS-645MG sebagai penggerak ruas dua. Ruas 1 terangkat dengan ketinggian tertentu, lalu dilanjutkan dengan ruas 2 yang memutar maju kaki robot. Halangan-halangan kecil yang lebih

rendah dari ketinggian terangkatnya ruas 1 dapat terlewati pada tahap ini.

Gerakan Langkah Mundur

Gerakan langkah mundur hampir sama dengan gerakan langkah maju. Perbedaannya hanya pada pengaturan ruas dua saja yang berputar ke arah mundur.

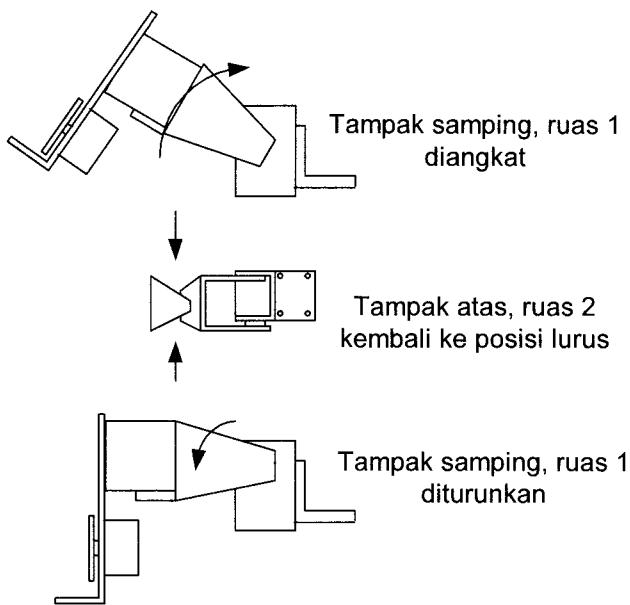


Gambar 7.8 Gerakan Langkah Mundur

Gerakan Langkah Lurus

Gerakan ini berfungsi untuk mengembalikan posisi kaki ke arah lurus, dari apapun kondisi sebelumnya. Baik dalam kondisi maju maupun mundur, gerakan ini akan membawa kembali kaki ke posisi lurus dengan cara melangkah.

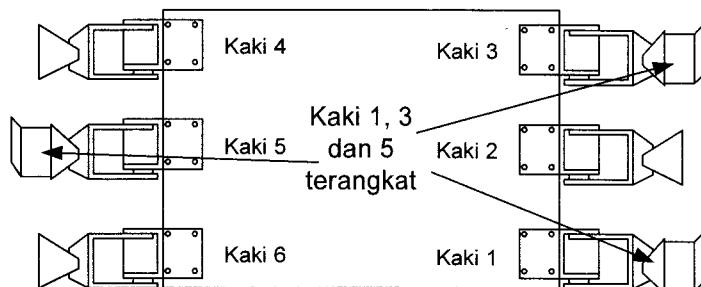
Bab 7: Robot Laba-Laba dengan 6 Kaki



Gambar 7.9 Gerakan Langkah Lurus

Gerakan Badan Kiri Maju

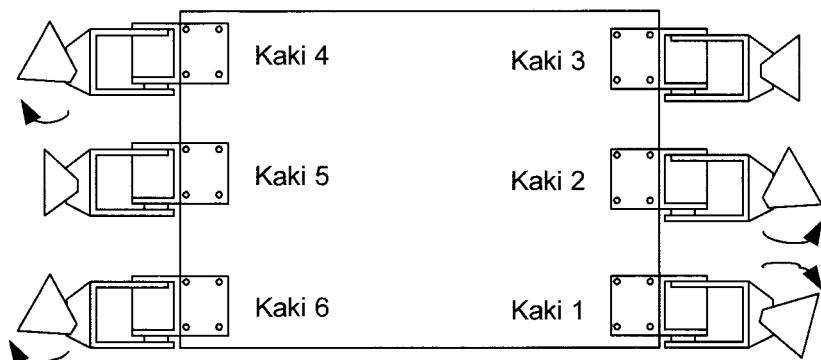
Gerakan ini berfungsi untuk menggerakkan tubuh robot bagian kanan maju. Proses gerakan dilakukan dengan mengangkat 3 buah kaki robot. Dua kaki kiri dan satu kaki kanan (tampak dari sisi robot)



Gambar 7.10 Gerakan mengangkat kaki 1, 3 dan 5

Pada saat ini, tumpuan tubuh robot berada pada kaki 2, 4, dan 6. Agar robot bergerak maju pada bagian kiri (kiri dari sudut pandang robot),

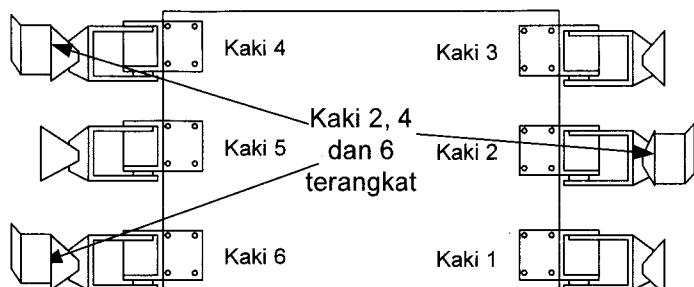
maka kaki-kaki ini diatur untuk mendorong ke belakang. Gerakan tersebut langsung dilanjutkan dengan gerakan maju kaki 1 disertai turunnya kaki 1, 3, dan 5. Dengan bantuan gravitasi, gerakan maju ini akan semakin lancar dan tubuh bagian kiri robot akan bergerak maju.



Gambar 7.11 Kaki 2, 4 dan 6 mendorong ke belakang dilanjutkan kaki 1 maju serta kaki 1, 3 dan 5 turun

Gerakan Badan Kanan Maju

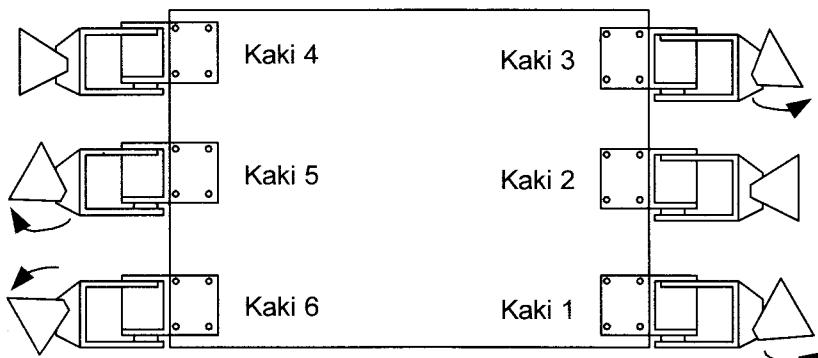
Gerakan ini berfungsi untuk menggerakkan tubuh robot bagian kiri maju dan biasanya merupakan salah satu bagian dari gerakan maju robot. Proses gerakan dilakukan dengan mengangkat 3 buah kaki robot. Dua kaki kanan dan satu kaki kiri (dari sisi robot)



Gambar 7.12 Gerakan mengangkat kaki 2, 4 dan 6

Bab 7: Robot Laba-Laba dengan 6 Kaki

Sama halnya dengan gerakan maju kiri, pada saat ini tumpuan tubuh robot berada pada kaki 1, 3, dan 5. Agar robot bergerak maju pada bagian kanan (kanan dari sudut pandang robot), maka kaki-kaki ini diatur untuk mendorong ke belakang. Gerakan tersebut langsung dilanjutkan dengan gerakan maju kaki 6 disertai turunnya kaki 2, 4, dan 6. Dengan bantuan gravitasi, maka gerakan maju ini akan semakin lancar dan tubuh bagian kanan robot akan bergerak maju.

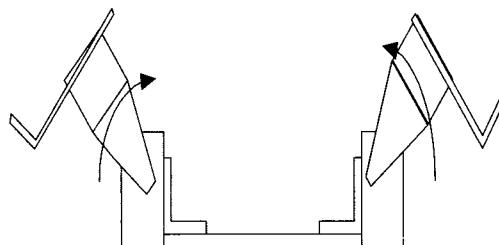


Gambar 7.13 Kaki 1, 3 dan 5 mendorong ke belakang dilanjutkan kaki 6 maju serta kaki 2, 4 dan 6 turun

Gerakan Robot Berdiri

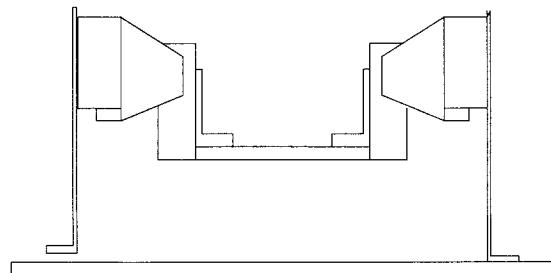
Gerakan ini adalah gerakan laba-laba untuk ke arah posisi siap. Gerakan ini difungsikan untuk mengatur posisi semua ruas kaki laba-laba ke arah di mana keenam kaki mengarah lurus ke samping dan menyentuh lantai. Ini adalah posisi di mana apapun kondisi sebelumnya, keenam kaki ini akan bergerak ke arah posisi siap untuk berdiri.

Gerakan ini diawali dengan gerakan meluruskan ruas-ruas dua dari setiap kaki sehingga pada saat mengangkat kaki, ruas-ruas tersebut tidak saling berbenturan. Kemudian dilanjutkan dengan gerakan mengangkat keenam kaki sehingga ruas 2 dan ruas 3 dari masing-masing kaki akan dapat bergerak dengan bebas.



Gambar 7.14 Gerakan mengangkat keenam kaki

Untuk meletakkan kaki-kaki laba-laba tersebut ke lantai tidak dilakukan dengan perintah menuju sudut, tetapi dengan menghentikan PWM pada motor-motor servo pengendali ruas satu. Akibatnya, ruas ini kehilangan tenaga dan kaki akan turun ke lantai. Teknik ini digunakan mengingat arah gerakan yang searah dengan gravitasi bumi akan bergerak terlalu cepat apabila dilakukan dengan perintah menuju sudut.



Gambar 7.15 Posisi berdiri

Tabel 2 Gerakan-gerakan servo dan nilai-nilai step

Nomor Motor Servo	Keterangan	Data Sudut Servo (Step)	Arah CW
1	Fungsi Ekstra		
3	Ruas 2	10 - 16	Kaki maju
4	Ruas 1	15 - 24	Kaki terangkat
5	Fungsi Ekstra		
7	Ruas 2	10 - 16	Kaki maju

Bab 7: Robot Laba-Laba dengan 6 Kaki

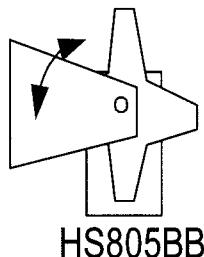
8	Ruas 1	15 - 24	Kaki terangkat
9	Fungsi Ekstra		
11	Ruas 2	10 - 16	Kaki maju
12	Ruas 1	15 - 24	Kaki terangkat
13	Fungsi Ekstra		
15	Ruas 2	10 - 16	Kaki mundur
16	Ruas 1	15 - 24	Kaki terangkat
17	Fungsi Ekstra		
19	Ruas 2	10 - 16	Kaki mundur
20	Ruas 1	15 - 24	Kaki terangkat
21	Fungsi Ekstra		
23	Ruas 2	10 - 16	Kaki mundur
24	Ruas 1	15 - 24	Kaki terangkat

Hubungan Gerakan dengan Sudut Motor Servo

Bagian ini akan membahas hubungan antara gerakan-gerakan mekanik pada robot laba-laba terhadap pengaturan sudut motor servo. Terdapat dua gerakan dasar, yaitu gerakan ruas satu yang berfungsi mengangkat dan menurunkan tubuh laba-laba dan gerakan ruas dua yang berfungsi mengarahkan kaki laba-laba ke depan.

Gerakan Ruas Satu

Ruas satu digerakkan oleh motor servo yang paling kuat torsinya, yaitu HS-805BB dengan torsi 24,7 kg. Penggunaan motor servo dengan torsi terbesar pada bagian ruas ini ditujukan untuk mengangkat tubuh laba-laba berdiri.



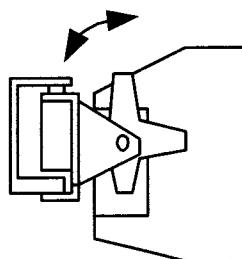
Gambar 7.16 Gerakan Ruas 1

Pada gerakan motor servo dikenal istilah searah jarum jam atau Clock Wise (CW) dan berlawanan arah jarum jam atau Counter Clock Wise (CCW). Pada ruas satu tampak bila servo bergerak searah jarum jam, maka ruas satu atau kaki laba-laba akan terangkat. Ini berarti laba-laba sendiri akan bergerak ke posisi tidur. Sebaliknya, bila motor servo bergerak berlawanan dengan arah jarum jam (CCW), maka ruas satu atau kaki laba-laba akan turun dan laba-laba sendiri akan bergerak berdiri.

Motor servo Hitech mempunyai spesifikasi durasi pulsa antara 0,9 ms untuk Full CCW hingga 2,1 ms untuk Full CW, di mana titik tengahnya adalah 1,5mS. Sedangkan DSR-08 mempunyai resolusi sebesar 87 μ s per step. Oleh karena itu, nilai untuk Full CCW dan Full CW dapat diperoleh dengan persamaan berikut:

$$\begin{aligned}
 \text{Nilai Full CCW} &= 0,9/0,087 \\
 &= 10,3 \text{ atau dibulatkan } 10 \\
 \text{Nilai Full CW} &= 2,1/0,087 \\
 &= 24,13 \text{ atau dibulatkan } 24
 \end{aligned}$$

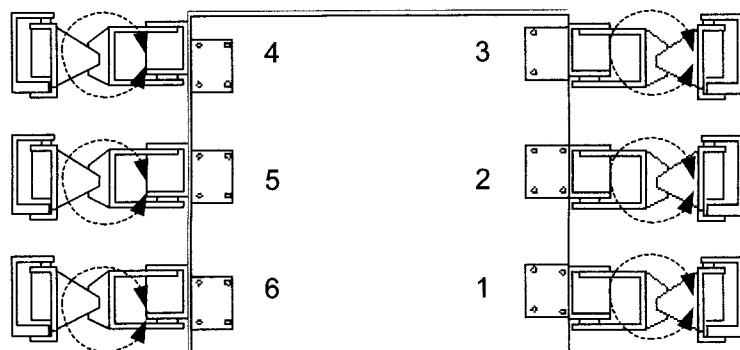
Agar tidak terjadi kerusakan mekanik, maka posisi servo pada ruas satu dari kaki laba-laba harus diatur sehingga motor servo berada pada posisi Full CW pada saat kondisi terangkat penuh, tepat sebelum menyentuh tubuh laba-laba. Hal ini digunakan untuk menghindarkan kerusakan akibat pengisian nilai sudut pada DSR-08 yang terlalu ekstrim dan kaki tertahan tubuh laba-laba. Motor servo akan memaksakan diri untuk menuju posisi yang tertahan tersebut dan akhirnya akan menimbulkan kerusakan mekanik pada bagian gear atau kerusakan elektronik pada bagian driver.



Gambar 7.17 Gerakan Ruas 2

Gerakan Ruas Dua

Meskipun tidak sekuat torsi motor servo pada ruas satu, namun pada ruas ini tetap dibutuhkan torsi yang cukup kuat. Di sini digunakan HS-645 MG yang memiliki torsi 9,6 kg-cm. Hal ini karena fungsi motor tersebut adalah menggerakkan tubuh laba-laba yang cukup berat ke arah maju maupun mundur.



Gambar 7.18 Arah Gerakan Ruas 2

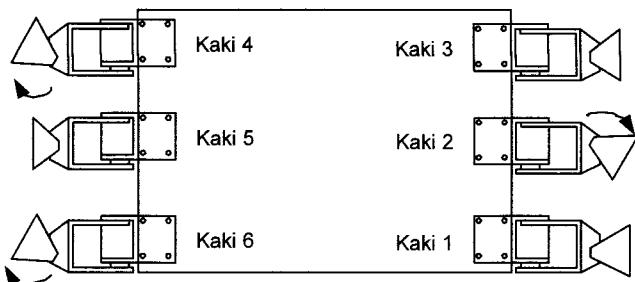
Posisi servo pada kaki 1, 2, dan 3 berlawanan dengan kaki 4, 5, dan 6 sehingga bila pada kaki 1, 2, dan 3 gerakan CW adalah gerakan ke arah maju, maka pada kaki 4, 5, dan 6 gerakan CW adalah gerakan ke arah mundur. Pada ruas ini, motor servo diatur agar arah Full CCW adalah arah mundur terjauh yang diperbolehkan pada kaki 1, 2, dan 3 atau arah maju terjauh yang diperbolehkan pada kaki 4, 5, dan 6.

Gerakan Robot Maju dan Berputar

Gerakan ini merupakan serangkaian dari gerakan maju badan robot bagian kiri dan kanan secara bergantian sehingga membentuk gerakan maju yang dibantu dengan gravitasi bumi sehingga robot dapat bergerak lebih cepat. Proses ini hanya dapat dilakukan pada saat robot bergerak di bidang yang mendatar di mana tidak dibutuhkan suction yang menahan tubuh robot terhadap permukaan bidang.

Untuk gerakan berputar, proses yang dilakukan hampir sama dengan gerakan robot maju, yaitu diawali dengan mengangkat tiga buah kaki robot. Namun, dalam hal ini perbedaannya adalah letak arah putar ruas kedua dari kaki-kaki yang tidak terangkat. Untuk gerakan maju, arah putar ruas diatur agar semua kaki mendorong ke belakang. Namun pada gerakan berputar, arah diatur agar bagian kiri dan kanan mendorong ke arah yang berlawanan sehingga robot akan bergerak memutar.

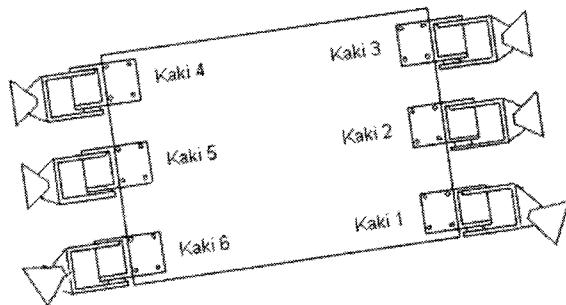
Agar robot bergerak memutar ke kiri, kaki 2 akan mendorong maju dan kaki 4 dan 6 mendorong mundur. Sebaliknya agar robot bergerak memutar ke kanan, kaki 2 akan mendorong mundur dan kaki 4 dan 6 mendorong maju.



Gambar 7.19 Robot berputar ke kiri (dari sudut pandang robot)

Pada Gambar 7.19 tampak bahwa kaki 2 mendorong maju, kaki 4 dan 6 mendorong mundur sehingga tubuh robot berputar ke kiri diiringi turunnya kaki 1, 3, dan 5.

Bab 7: Robot Laba-Laba dengan 6 Kaki

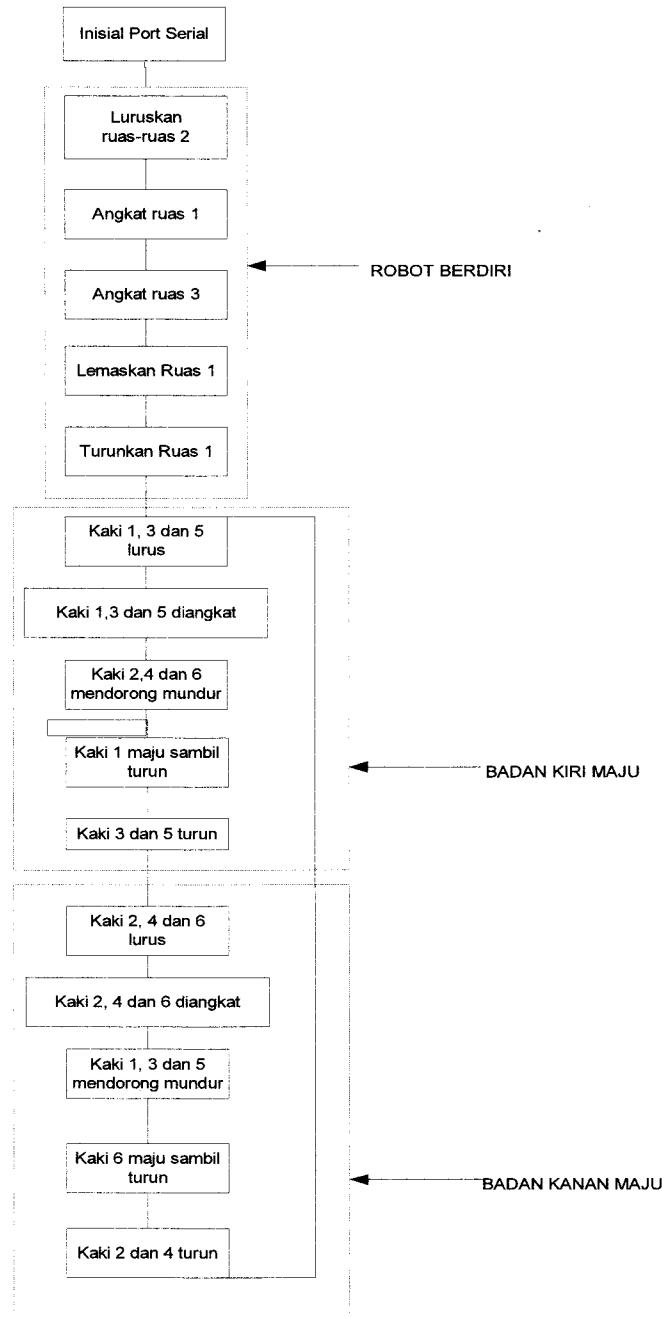


Gambar 7.20 Robot bersiap untuk berputar lagi

Proses gerakan dilanjutkan dengan kaki 1 dan 6 melangkah maju serta kaki 2, 3, 4, dan 5 kembali ke posisi lurus sehingga robot dapat siap untuk berputar lagi bila masih diperintahkan.

Berbeda dengan gerakan memutar tadi, kali ini gerakan meluruskan kaki 2, 3, 4, dan 5 serta langkah maju kaki 1 dan 6 berupa sebuah langkah dengan mengangkat terlebih dahulu kaki tersebut sehingga tubuh robot tidak berubah posisi untuk saat ini. Setelah proses ini dilakukan, maka robot telah siap untuk kembali melakukan gerakan putar apabila masih diperintahkan demikian.

Gambar 7.21 adalah diagram alir dari robot yang bergerak maju. Proses diawali dengan inisialisasi serial port, yaitu port yang digunakan untuk komunikasi antara otak robot dengan modul-modul servo controller. Kemudian, robot mulai melakukan gerakan-gerakan untuk berdiri dengan diawali mengangkat keenam kaki, meluruskan dan menurunkan kembali ke posisi siap. Gerakan badan maju, baik kiri maupun kanan mulai dilakukan setelah robot bergerak berdiri.



Gambar 7.21 Diagram Alir Robot Bergerak Maju

Bab 7: Robot Laba-Laba dengan 6 Kaki

Spider.asm:

```
$MOD51
; PROGRAM DELTA ROBO SPIDER BERJALAN
$MOD51
        DSEG
DeviceID      EQU    00h
PerintahKontrolMotor EQU    01
STX          EQU    1Eh
NomorRobot    EQU    03      ; Nomor urut Robot
Org 30H

;VARIABEL-VARIABEL
DeviceAddress:   Ds     1      ; Nomor urut robot
TargetType:      Ds     1      ; Jenis Slave
TargetNum:       Ds     1      ; Nomor urut Slave
SourceType:      Ds     1      ; Variabel Master
SourceNum:       Ds     1      ; Variabel nomor urut Master
CheckSumData:   Ds     1      ; Checksum paket data serial
PanjangDataSerial: Ds     1      ; Panjang data pada paket serial
Buffer:          Ds    25      ; Buffer data serial
BufferPerintah:  Ds     5      ; Buffer perintah dari Wireless
Joystick
Counter_5mS:     Ds     1

        CSEG
Org 00
Ljmp Start

        Org 03h
Reti

        Org 0Bh
Reti

        Org 13h
Reti

        Org 23h
Reti

Start:
        Mov P0,#00           ; Level indicator penuh
        Lcall Init_Serial    ; Inisial Serial Port 115200 bps
        Mov PCON,#80h
        Mov TH1,#0FFh
;
;
;

        Lcall Wakeup         ; Robot berdiri

        Lcall Delay_500MS
        Lcall Kaki1LangkahMaju ; Majukan kaki2 depan
        Lcall Delay_500MS
        Lcall Kaki6LangkahMaju ; 

Loop:
        Lcall MajuModel
        Ljmp Loop

;=====
; GERAKAN-GERAKAN ROBOT

MajuModel:
```

```

Lcall  BadanKiriAngkat
Lcall  Delay_100mS
Lcall  BadanKananAngkat
Lcall  Delay_100mS
Ret

MundurModel1:
Lcall  BadanKiriMundur
Lcall  Delay_100mS
Lcall  BadanKananMundur
Lcall  Delay_100mS
Ret

WakeUp:
Lcall  Standby           ;Gerakkan kaki ke posisi siap
Lcall  Delay_500mS        ;
Lcall  Turun6Kaki         ;Berdiri
Ret

Standby:
Lcall  GeserLurus6Kaki   ;Luruskan 6 kaki
Lcall  Delay_1detik        ;
Lcall  Angkat6Kaki         ;Angkat 6 kaki
Lcall  Delay_500mS        ;
Lcall  Ruas1Lemas          ;Lemaskan ruas-ruas 1
Ret

Turun6Kaki:
Mov   DPTR,#TabelTurunKaki
Lcall SetPosisiServo      ;turun
Lcall SetPosisiServo
Lcall SetPosisiServo
Lcall SetPosisiServo
Lcall SetPosisiServo
Lcall SetPosisiServo
Ret

;=====
;Gerakan langkah-langkah kaki

Kaki1LangkahMundur:
Mov   DPTR,#Kaki1Mundur
Lcall SetPosisiServo      ;angkat kaki
Lcall Delay_50mS
Lcall SetPosisiServo      ;maju kaki
Lcall Delay_50ms
Lcall SetPosisiServo      ;turun ruas 3
Lcall Delay_50mS
Lcall SetPosisiServo      ;turun kaki
Lcall Delay_100mS
Ret

Kaki2LangkahMundur:
Mov   DPTR,#Kaki2Mundur
Lcall SetPosisiServo      ;angkat kaki
Lcall Delay_50mS
Lcall SetPosisiServo      ;maju kaki
Lcall Delay_50ms
Lcall SetPosisiServo      ;turun ruas 3
Lcall Delay_50mS
Lcall SetPosisiServo      ;turun kaki
Lcall Delay_100mS

```

Bab 7: Robot Laba-Laba dengan 6 Kaki

Ret

Kaki3LangkahMundur:

```
Mov      DPTR, #Kaki3Mundur
Lcall   SetPosisiServo      ;angkat kaki
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;maju kaki
Lcall   Delay_50ms
Lcall   SetPosisiServo      ;turun ruas 3
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;turun kaki
Lcall   Delay_100mS
Ret
```

Kaki4LangkahMundur:

```
Mov      DPTR, #Kaki4Mundur
Lcall   SetPosisiServo      ;angkat kaki
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;maju kaki
Lcall   Delay_50ms
Lcall   SetPosisiServo      ;turun ruas 3
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;turun kaki
Lcall   Delay_100mS
Ret
```

Kaki5LangkahMundur:

```
Mov      DPTR, #Kaki5Mundur
Lcall   SetPosisiServo      ;angkat kaki
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;maju kaki
Lcall   Delay_50ms
Lcall   SetPosisiServo      ;turun ruas 3
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;turun kaki
Lcall   Delay_100mS
Ret
```

Kaki6LangkahMundur:

```
Mov      DPTR, #Kaki6Mundur
Lcall   SetPosisiServo      ;angkat kaki
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;maju kaki
Lcall   Delay_50ms
Lcall   SetPosisiServo      ;turun ruas 3
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;turun kaki
Lcall   Delay_100mS
Ret
```

Kaki1LangkahMaju:

```
Mov      DPTR, #Kaki1Maju
Lcall   SetPosisiServo      ;angkat kaki
Lcall   Delay_50mS
Lcall   SetPosisiServo      ;maju kaki
Lcall   Delay_50ms
Lcall   SetPosisiServo      ;turun ruas 3
Lcall   Delay_50mS
```

```

Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki2LangkahMaju:
Mov    D PTR, #Kaki2Maju
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki3LangkahMaju:
Mov    D PTR, #Kaki3Maju
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki4LangkahMaju:
Mov    D PTR, #Kaki4Maju
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki5LangkahMaju:
Mov    D PTR, #Kaki5Maju
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki6LangkahMaju:
Mov    D PTR, #Kaki6Maju
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3

```

Bab 7: Robot Laba-Laba dengan 6 Kaki

```
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki6LangkahLurus:
Mov    DPTR, #Kaki6Lurus
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki5LangkahLurus:
Mov    DPTR, #Kaki5Lurus
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki4LangkahLurus:
Mov    DPTR, #Kaki4Lurus
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki3LangkahLurus:
Mov    DPTR, #Kaki3Lurus
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

Kaki2LangkahLurus:
Mov    DPTR, #Kaki2Lurus
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
```

```

Lcall  Delay_50mS
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

```

```

KakiILangkahLurus:
Mov    DPTR,#KakiILangkahLurus
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;maju kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;turun kaki
Lcall  Delay_100mS
Ret

```

```

KakiITengahLurus:
Mov    DPTR,#TKakiITengahLurus
Lcall  SetPosisiServo      ;angkat kaki
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;maju kaki
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50ms
Lcall  SetPosisiServo      ;turun ruas 3
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_50mS
Lcall  SetPosisiServo      ;turun kaki
Lcall  SetPosisiServo      ;angkat kaki
Lcall  Delay_100mS
Ret

```

```

;=====
; SUBROUTINE RUAS 3
;Ruas3Lemas:
Push  DPH
Push  DPL
Mov   R7,#01
Mov   R6,#02
Lcall KirimStopMotor
Mov   R7,#01
Mov   R6,#06
Lcall KirimStopMotor
Mov   R7,#02
Mov   R6,#02
Lcall KirimStopMotor
Mov   R7,#02
Mov   R6,#06
Lcall KirimStopMotor
Mov   R7,#03
Mov   R6,#02
Lcall KirimStopMotor
Mov   R7,#03
Mov   R6,#06
Lcall KirimStopMotor
Pop   DPL
Pop   DPH
Ret

```

Bab 7: Robot Laba-Laba dengan 6 Kaki

```
AngkatRuas3:  
    Mov      DPTR, #TabelAngkatRuas3  
    Lcall   SetPosisiServo          ;angkat ruas 3  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Ret  
  
GeserLurus6Kaki:  
    Mov      DPTR, #TabelLurus6Kaki  
    Lcall   SetPosisiServo          ;lurus  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Ret  
  
Ruas2Lemas:  
    Push    DPH  
    Push    DPL  
    Mov     R7, #01  
    Mov     R6, #03  
    Lcall   KirimStopMotor  
    Mov     R7, #01  
    Mov     R6, #07  
    Lcall   KirimStopMotor  
    Mov     R7, #02  
    Mov     R6, #03  
    Lcall   KirimStopMotor  
    Mov     R7, #02  
    Mov     R6, #07  
    Lcall   KirimStopMotor  
    Mov     R7, #03  
    Mov     R6, #03  
    Lcall   KirimStopMotor  
    Mov     R7, #03  
    Mov     R6, #07  
    Lcall   KirimStopMotor  
    Pop    DPL  
    Pop    DPH  
    Ret  
  
=====  
; SUBROUTINE RUAS 1  
  
Angkat6Kaki:  
    Mov      DPTR, #TabelAngkat6Kaki  
    Lcall   SetPosisiServo          ;angkat  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Ret
```

```

Ruas1Lemas:
    Push    DPH
    Push    DPL
    Mov     R7, #01
    Mov     R6, #04
    Lcall   KirimStopMotor
    Mov     R7, #01
    Mov     R6, #08
    Lcall   KirimStopMotor
    Mov     R7, #02
    Mov     R6, #04
    Lcall   KirimStopMotor
    Mov     R7, #02
    Mov     R6, #08
    Lcall   KirimStopMotor
    Mov     R7, #03
    Mov     R6, #04
    Lcall   KirimStopMotor
    Mov     R7, #03
    Mov     R6, #08
    Lcall   KirimStopMotor
    Pop    DPL
    Pop    DPH
    Ret

```

```

KirimStopMotor:
    Mov     TargetType, #11h
    Mov     TargetNum, R7
    Mov     DeviceAddress, #NomorRobot
    Mov     Buffer, #02
    Mov     Buffer+1, #07
    Mov     Buffer+2, R6
    Ljmp   KirimPaket

```

```

SetPosisiServo:
    Mov     TargetType, #11h           ;Target Servo Controller
    (DSR-30 / DSR-08)

    Mov     A, #00                   ;Set nomor DSR-08
    Movc   A, @A+DPTR
    Mov     TargetNum, A
    Inc    DPTR
    Mov     DeviceAddress, #NomorRobot ;Set nomor robot = 01
    Mov     R0, #Buffer
    Mov     @R0, #04
    Inc    R0
    Mov     @R0, #PerintahKontrolMotor ;Kirim perintah kontrol motor
    Inc    R0

    Mov     A, #00                   ;Set Nomor Servo
    Movc   A, @A+DPTR
    Mov     @R0, A
    Inc    R0
    Inc    DPTR
    ;

    Mov     A, #00                   ;Set Posisi servo
    Movc   A, @A+DPTR
    Mov     @R0, A
    Inc    R0
    Inc    DPTR
    ;

```

Bab 7: Robot Laba-Laba dengan 6 Kaki

```
Mov      A, #00                      ;Set delay / kekuatan servo
Movc    A, @A+DPTR
Mov     @R0, A
Inc     R0
Inc     DPTR
;

KirimPaket:
Push   DPH
Push   DPL
Lcall  KirimPaketData
Lcall  AmbilDataSerial

Pop    DPL
Pop    DPH
Ret

;=====
; Ambil data serial (ACK atau info dari DSR-08)

AmbilDataSerial:
Lcall  Serial_In
Cjne  A, #1Eh, AmbilDataSerial
Lcall  Serial_In
Lcall  Serial_In
Lcall  Serial_In
Lcall  Serial_In
Lcall  Serial_In
Mov    B, A
Mov    R0, #Buffer
Mov    @R0, A
Inc    R0
LoopDataIn:
Lcall  Serial_In
Mov    @R0, A
Inc    R0
Djnz  B, LoopDataIn
Lcall  Serial_In
Ret

KirimPaketData:
Lcall  AwalPaketMaster

Mov    R0, #Buffer
Mov    A, @R0
Mov    R7, A
Lcall  SerialOutCS

Mov    R0, #Buffer+1
LoopBufferOut:
Mov    A, @R0
Inc    R0
Lcall  SerialOutCS
Djnz  R7, LoopBufferOut
Mov    A, #00
Clr    C
Subb  A, ChecksumData
Lcall  Serial_Out
Ret
```

```

AwalPaketMaster:
    Mov     A,#STX
    Lcall   Serial_Out
    Mov     CheckSumData,A

    Mov     A,TargetType
    Lcall   SerialOutCS

    Mov     A,TargetNum
    Lcall   SerialOutCS

    Mov     A,#DeviceID
    Lcall   SerialOutCS

    Mov     A,DeviceAddress
    Lcall   SerialOutCS
    Ret

AwalPaket:
    Mov     A,#STX           ;Start
    Lcall   Serial_Out
    Mov     CheckSumData,A

    Mov     A,SourceType
    Lcall   SerialOutCS
    Mov     A,SourceNum
    Lcall   SerialOutCS

    Mov     A,#DeviceID
    Lcall   SerialOutCS

    Mov     A,DeviceAddress
    Lcall   SerialOutCS
    Ret

SaveBuffer:
    Mov     @R0,A           ;
    Inc     R0
    Add     A,CheckSumData ;Tambah check sum
    Mov     CheckSumData,A ;;
    Ret

SerialOutCS:
    Lcall   Serial_Out
    Add     A,ChecksumData
    Mov     CheckSumData,A
    Ret

BadanKiriMundur:
    Mov     DPTR,#TbBdKiriMundur
    LJmp   BadanMaju

BadanKananMundur:
    Mov     DPTR,#TbBdKananMundur
    LJmp   BadanMaju

BadanKananAngkat:
    Mov     DPTR,#TabelBdnKananMaju
    LJmp   BadanMaju

```

Bab 7: Robot Laba-Laba dengan 6 Kaki

```
BadanKiriAngkat:  
    Mov      DPTR, #TabelBdnKiriMaju  
  
BadanMaju:  
    Lcall   SetPosisiServo  
    Lcall   Delay_100mS  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   Delay_100mS  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Ret  
  
PutarKiri:  
    Mov      DPTR, #TabelPutarKiri  
    Lcall   SetPosisiServo  
    Lcall   Delay_100mS  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   Delay_100mS  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   Delay_100mS  
    Lcall   Delay_100mS  
    Lcall   Kaki6LangkahMaju  
    Lcall   Kaki1LangkahMaju  
    Lcall   KakiTengahLurus  
    Lcall   Kaki3LangkahLurus  
    Lcall   Kaki4LangkahLurus  
    Ret  
  
PutarKanan:  
    Mov      DPTR, #TabelPutarKanan  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   Delay_100mS  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   Delay_100mS  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo  
    Lcall   SetPosisiServo
```

```

Lcall  Delay_100mS
Lcall  Delay_100mS
Lcall  Kaki6LangkahMaju
Lcall  Kaki1LangkahMaju
Lcall  KakiTengahLurus
Lcall  Kaki3LangkahLurus
Lcall  Kaki4LangkahLurus
Ret

```

```

=====
; TABEL GERAKAN SERVO
=====
```

TabelBdnKiriMaju:

DB	01,03,13,80h	;Ruas 2 lurus
DB	02,03,13,80h	;Ruas 2 lurus
DB	03,03,13,80h	;Ruas 2 lurus
DB	01,04,18,80h	;Kaki 1 angkat
DB	02,04,17,80h	;Kaki 3 angkat
DB	03,04,18,80h	;Kaki 5 angkat
DB	01,07,10,80h	;Kaki 2 mundur
DB	02,07,15,80h	;Kaki 4 mundur
DB	03,07,16,80h	;Kaki 6 mundur
DB	01,03,16,80h	;Kaki 1 maju
DB	01,04,15,80h	;Kaki 1 turun
DB	02,04,15,80h	;Kaki 3 turun
DB	03,04,15,80h	;Kaki 5 turun

TabelBdnKananMaju:

DB	01,07,13,80h	;Ruas 2 lurus
DB	02,07,13,80h	;Ruas 2 lurus
DB	03,07,13,80h	;Ruas 2 lurus
DB	01,08,18,80h	;Kaki 2 angkat
DB	02,08,17,80h	;Kaki 4 angkat
DB	03,08,18,80h	;Kaki 6 angkat
DB	01,03,10,80h	;Kaki 1 mundur
DB	02,03,10,80h	;Kaki 3 mundur
DB	03,03,16,80h	;Kaki 5 mundur
DB	03,07,11,80h	;Kaki 6 maju
DB	01,08,15,80h	
DB	02,08,15,80h	
DB	03,08,15,80h	

TbBdKiriMundur:

DB	01,03,13,80h	;Ruas 2 lurus
DB	02,03,13,80h	;Ruas 2 lurus
DB	03,03,13,80h	;Ruas 2 lurus
DB	01,04,18,80h	;Kaki 1 angkat
DB	02,04,17,80h	;Kaki 3 angkat
DB	03,04,18,80h	;Kaki 5 angkat
DB	01,07,16,80h	;Kaki 2 maju
DB	02,07,10,80h	;Kaki 4 maju
DB	03,07,10,80h	;Kaki 6 maju
DB	01,03,10,80h	;Kaki 1 mundur

Bab 7: Robot Laba-Laba dengan 6 Kaki

DB	01,04,15,80h	;Kaki 1 turun
DB	02,04,15,80h	;Kaki 3 turun
DB	03,04,15,80h	;Kaki 5 turun

TbBdKananMundur:

DB	01,07,13,80h	;Ruas 2 lurus
DB	02,07,13,80h	;Ruas 2 lurus
DB	03,07,13,80h	;Ruas 2 lurus

DB	01,08,18,80h	;Kaki 2 angkat
DB	02,08,17,80h	;Kaki 4 angkat
DB	03,08,18,80h	;Kaki 6 angkat

DB	01,03,16,80h	;Kaki 1 maju
DB	02,03,16,80h	;Kaki 3 maju
DB	03,03,10,80h	;Kaki 5 maju
DB	03,07,16,80h	;Kaki 6 mundur

DB	01,08,15,80h
DB	02,08,15,80h
DB	03,08,15,80h

TabelPutarKiri:

DB	01,03,13,80h	;Ruas 2 lurus
DB	02,03,13,80h	;Ruas 2 lurus
DB	03,03,13,80h	;Ruas 2 lurus

DB	01,04,18,80h	;Kaki 1 angkat
DB	02,04,17,80h	;Kaki 3 angkat
DB	03,04,18,80h	;Kaki 5 angkat

DB	01,07,16,80h	;Kaki 2 maju
DB	02,07,15,80h	;Kaki 4 mundur
DB	03,07,16,80h	;Kaki 6 mundur

DB	01,04,15,80h	;Kaki 1 turun
DB	02,04,15,80h	;Kaki 3 turun
DB	03,04,15,80h	;Kaki 5 turun

TabelPutarKanan:

DB	01,07,13,80h	;Ruas 2 lurus
DB	02,07,13,80h	;Ruas 2 lurus
DB	03,07,13,80h	;Ruas 2 lurus

DB	01,08,18,80h	;Kaki 1 angkat
DB	02,08,17,80h	;Kaki 3 angkat
DB	03,08,18,80h	;Kaki 5 angkat

DB	01,03,10,80h	;Kaki 2 mundur
DB	02,03,10,80h	;Kaki 4 maju
DB	03,03,10,80h	;Kaki 6 maju

DB	01,08,15,80h	;Kaki 1 turun
DB	02,08,15,80h	;Kaki 3 turun
DB	03,08,15,80h	;Kaki 5 turun

TabelSerongKiri:

DB	02,01,11,80h	;Sucker kaki 3 lepas
DB	02,05,11,80h	;Sucker kaki 4 lepas
DB	01,03,11,80h	;Ruas 2 kaki 1 mundur
DB	03,07,13,80h	;Ruas 2 kaki 6 lurus
DB	02,03,11,80h	;Ruas 2 kaki 3 mundur
DB	02,07,13,80h	;Ruas 2 kaki 4 lurus

TabelSerongKanan:

DB	02,01,11,80h	;Sucker kaki 3 lepas
DB	02,05,11,80h	;Sucker kaki 4 lepas
DB	01,03,13,80h	;Ruas 2 kaki 1 lurus
DB	03,07,15,80h	;Ruas 2 kaki 6 mundur
DB	02,03,13,80h	;Ruas 2 kaki 3 lurus
DB	02,07,15,80h	;Ruas 2 kaki 4 mundur

Kaki1Lurus:

DB	01,04,20,80h	;Ruas 1 diangkat
DB	01,03,13,80h	;Ruas 2 lurus
DB	01,02,15,80h	;Ruas 3 siap
DB	01,04,15,80h	;Ruas 1 turun

Kaki2Lurus:

DB	01,08,20,80h	;Ruas 1 diangkat
DB	01,07,13,80h	;Ruas 2 lurus
DB	01,06,14,80h	;Ruas 3 siap
DB	01,08,15,80h	;Ruas 1 turun

Kaki3Lurus:

DB	02,04,20,80h	;Ruas 1 diangkat
DB	02,03,13,80h	;Ruas 2 lurus
DB	02,02,13,80h	;Ruas 3 siap
DB	02,04,15,80h	;Ruas 1 turun

Kaki4Lurus:

DB	02,08,20,80h	;Ruas 1 diangkat
DB	02,07,13,80h	;Ruas 2 lurus
DB	02,06,14,80h	;Ruas 3 siap
DB	02,08,15,80h	;Ruas 1 turun

Kaki5Lurus:

DB	03,04,20,80h	;Ruas 1 diangkat
DB	03,03,13,80h	;Ruas 2 lurus
DB	03,02,13,80h	;Ruas 3 siap
DB	03,04,15,80h	;Ruas 1 turun

Kaki6Lurus:

DB	03,08,20,80h	;Ruas 1 diangkat
DB	03,07,13,80h	;Ruas 2 lurus
DB	03,06,14,80h	;Ruas 3 siap
DB	03,08,15,80h	;Ruas 1 turun

Kaki1Maju:

DB	01,04,20,80h	;Ruas 1 diangkat
DB	01,03,15,80h	;Ruas 2 maju
DB	01,02,15,80h	;Ruas 3 siap
DB	01,04,15,80h	;Ruas 1 turun

Bab 7: Robot Laba-Laba dengan 6 Kaki

Kaki2Maju:

DB	01,08,20,80h	;Ruas 1 diangkat
DB	01,07,15,80h	;Ruas 2 maju
DB	01,06,14,80h	;Ruas 3 siap
DB	01,08,15,80h	;Ruas 1 turun

Kaki3Maju:

DB	02,04,20,80h	;Ruas 1 diangkat
DB	02,03,15,80h	;Ruas 2 maju
DB	02,02,13,80h	;Ruas 3 siap
DB	02,04,15,80h	;Ruas 1 turun

Kaki4Maju:

DB	02,08,20,80h	;Ruas 1 diangkat
DB	02,07,11,80h	;Ruas 2 maju
DB	02,06,14,80h	;Ruas 3 siap
DB	02,08,15,80h	;Ruas 1 turun

Kaki5Maju:

DB	03,04,20,80h	;Ruas 1 diangkat
DB	03,03,11,80h	;Ruas 2 maju
DB	03,02,14,80h	;Ruas 3 siap
DB	03,04,15,80h	;Ruas 1 turun

Kaki6Maju:

DB	03,08,20,80h	;Ruas 1 diangkat
DB	03,07,12,80h	;Ruas 2 maju
DB	03,06,14,80h	;Ruas 3 siap
DB	03,08,15,80h	;Ruas 1 turun

Kaki1Mundur:

DB	01,04,20,80h	;Ruas 1 diangkat
DB	01,03,11,80h	;Ruas 2 mundur
DB	01,02,15,80h	;Ruas 3 siap
DB	01,04,15,80h	;Ruas 1 turun

Kaki2Mundur:

DB	01,08,20,80h	;Ruas 1 diangkat
DB	01,07,11,80h	;Ruas 2 mundur
DB	01,06,14,80h	;Ruas 3 siap
DB	01,08,15,80h	;Ruas 1 turun

Kaki3Mundur:

DB	02,04,20,80h	;Ruas 1 diangkat
DB	02,03,11,80h	;Ruas 2 mundur
DB	02,02,13,80h	;Ruas 3 siap
DB	02,04,15,80h	;Ruas 1 turun

Kaki4Mundur:

DB	02,08,20,80h	;Ruas 1 diangkat
DB	02,07,15,80h	;Ruas 2 mundur
DB	02,06,14,80h	;Ruas 3 siap
DB	02,08,15,80h	;Ruas 1 turun

Kaki5Mundur:

DB	03,04,20,80h	;Ruas 1 diangkat
DB	03,03,15,80h	;Ruas 2 mundur
DB	03,02,14,80h	;Ruas 3 siap
DB	03,04,15,80h	;Ruas 1 turun

```

Kaki6Mundur:
    DB      03,08,20,80h ;Ruas 1 diangkat
    DB      03,07,15,80h ;Ruas 2 mundur
    DB      03,06,12,80h ;Ruas 3 siap
    DB      03,08,15,80h ;Ruas 1 turun

;=====
; Gerakan-gerakan kaki tengah

TKakiTengahLurus:
    DB      01,08,20,80h ;Ruas 1 diangkat
    DB      03,04,20,80h ;Ruas 1 diangkat
    DB      01,07,13,80h ;Ruas 2 lurus
    DB      03,03,13,80h ;Ruas 2 lurus
    DB      01,06,14,80h ;Ruas 3 siap
    DB      03,02,13,80h ;Ruas 3 siap
    DB      01,08,15,80h ;Ruas 1 turun
    DB      03,04,15,80h ;Ruas 1 turun

Kaki2Turun:
    DB      01,08,15,80h ;Ruas 1 kaki 2 turun
    DB      01,05,23,80h ;Sedot sucker

Kaki5Turun:
    DB      03,04,15,80h ;Ruas 1 kaki 5 turun
    DB      03,01,17,80h ;Sedot sucker

;=====

TabelLurus6Kaki:
    DB      01,03,13,80h
    DB      01,07,13,80h
    DB      02,03,13,80h
    DB      02,07,11,80h
    DB      03,03,13,80h
    DB      03,07,14,80h

TabelAngkat6Kaki:
    DB      01,04,20,80h ;angkat ruas 1
    DB      01,08,20,80h ;
    DB      02,04,20,80h ;
    DB      02,08,20,80h ;
    DB      03,04,20,80h ;
    DB      03,08,20,80h ;

TabelAngkatRuas3:
    DB      01,02,15,80h ;Siapkan ruas 3
    DB      01,06,14,80h ;
    DB      02,02,13,80h ;
    DB      02,06,14,80h ;
    DB      03,02,13,80h ;
    DB      03,06,15,80h ;

TabelTurunKaki:
    DB      01,04,15,80h
    DB      01,08,14,80h
    DB      02,04,15,80h
    DB      02,08,14,80h
    DB      03,04,15,80h
    DB      03,08,15,80h

```

Bab 7: Robot Laba-Laba dengan 6 Kaki

```
Init_Serial:  
    MOV     SCON, #52H      ; Mode 1 Ren  
;    MOV     TMOD, #20H      ; T0 Mode 2, T1 Mode 2  
    Mov     A, TMOD  
    Anl     A, #0FH  
    Orl     A, #20H  
    Mov     TMOD, A  
    MOV     TH1, #0FFH      ; 57600  
    Setb   TR1  
    MOV     PCON, #80H      ;  
    RET  
  
Serial_Out:  
    Clr     TI  
    Mov     SBUF, A  
TungguKirim:  
    Mov     0A6H, #1EH  
    Mov     0A6H, #0E1H  
    Jnb     TI, TungguKirim  
    Clr     TI  
    Ret  
  
Serial_In:  
    Clr     RI  
TungguTerima:  
    Mov     0A6H, #1EH  
    Mov     0A6H, #0E1H  
    Jnb     RI, TungguTerima  
    Mov     A, SBUF  
    Clr     RI  
    Ret  
  
KirimPesan_Serial:  
    Mov     A, #00H  
    Movc   A, @A+DPTR  
    Cjne   A, #0FH, KirimTerus  
    Ret  
  
KirimTerus:  
    Lcall  Serial_Out  
    Inc    DPTR  
    Ajmp  KirimPesan_Serial  
  
=====  
; SUBROUTINE DELAY  
  
Delay_1detik:  
    Mov     Counter_5mS, #0200  
  
Tunggu_1detik:  
    Lcall  Delay_5mS  
    Mov     0A6h, #1Eh  
    Mov     0A6h, #0E1h  
    Djnz   Counter_5mS, Tunggu_1detik  
    Ret  
  
Delay_500ms:  
    Mov     Counter_5mS, #0100  
  
Tunggu_500mS:  
    Lcall  Delay_5mS  
    Mov     0A6h, #1Eh  
    Mov     0A6h, #0E1h
```

```

Djnz    Counter_5mS, Tunggu_500mS
Ret

Delay_100mS:
Mov     Counter_5mS, #020

Tunggu_100mS:
Lcall   Delay_5mS
Mov     0A6h, #1Eh
Mov     0A6h, #0E1h
Djnz    Counter_5mS, Tunggu_100mS
Ret

Delay_5mS:
Push    TMOD
Mov     TMOD, #21H           ;Timer Mode 16 bit counter
Mov     TH0, #0EDH
Mov     TL0, #0FFH
Setb    TR0

Tunggu_5mS:
Jbc    TF0, Sudah_5mS
Ljmp   Tunggu_5mS

Sudah_5mS:
Clr    TR0
Pop    TMOD
Ret

Delay_50mS:
Push    07H
Mov     R7, #10
loopdelay50MS:
Lcall   Delay_5mS
Mov     0A6h, #1Eh
Mov     0A6h, #0E1h
Djnz    R7, Loopdelay50MS
Pop    07H
Ret

=====
END

```

Latihan

1. Gambarkan mekanik yang dibutuhkan agar servo dengan fungsi ekstra dapat menarik suction untuk menempel di dinding keramik.
2. Tuliskan algoritma gerakan robot laba-laba pada bidang miring dengan tambahan fungsi suction.
3. Tuliskan algoritma gerakan robot laba-laba yang bergerak mundur.

Bab 7: Robot Laba-Laba dengan 6 Kaki

4. Tuliskan listing program untuk robot laba-laba yang dikendalikan melalui D-Joy Controller Wireless.
5. Kembangkan pengontrolan wireless robot menggunakan TLP/RLP 433 MHz atau Bluetooth.

Bab 8

Computer Vision untuk Robot

Pendahuluan

Jika ingin membangun robot humanoid yang handal, maka penguasaan computer vision sangat penting. Arah dari penguasaan teknologi ini adalah membuat robot mirip manusia yang dapat diterapkan sebagai robot pelayan (*servant robot*). Computer vision berguna mengolah input images, umumnya melalui kamera pada robot.

Mengenal Computer Vision pada Robot

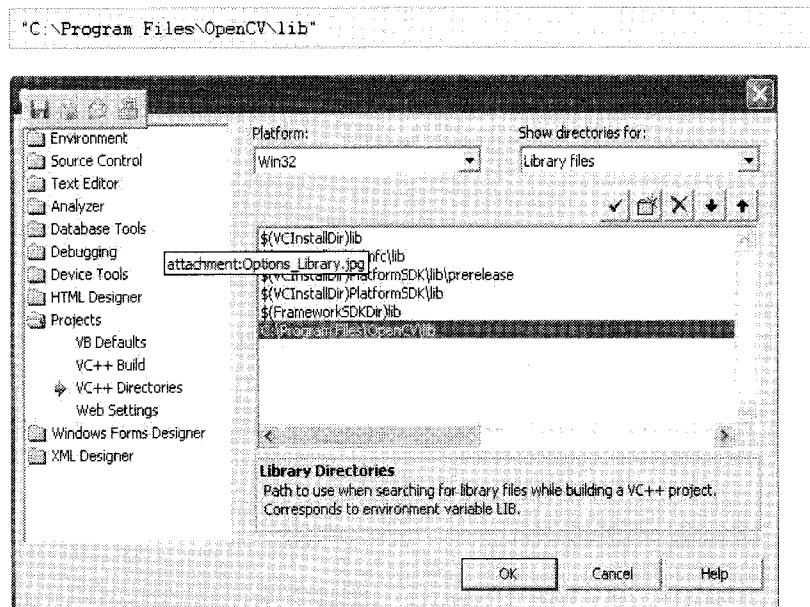
Program OpenCV

OpenCV adalah program open source berbasiskan C++ yang saat ini banyak digunakan sebagai program computer vision. Salah satu penerapannya adalah pada robotika. Dengan OpenCV, Anda dapat membuat interaksi antara manusia dan robot (Human-Robot Interaction). Misalnya, wajah manusia dideteksi oleh camera/webcam, lalu diproses oleh komputer, kemudian diproses oleh robot untuk melakukan aksi tertentu, seperti mengikuti/mengenal wajah orang tersebut.

Kesemuanya itu membutuhkan OpenCV sebagai program utama antara webcam dan pengolahnya, yaitu komputer. Silahkan kunjungi situs opencv.org untuk men-download atau mengetahui berita terbaru tentang software ini.

Pemrograman Dasar OpenCV

Anda membutuhkan editor dan kompiler Visual Studio .Net 2005 untuk mengedit dan mengkompilasi program OpenCV. Anda terlebih dahulu harus mengkonfigurasi Visual C++ .Net tersebut di mana file library dan source-nya harus disertakan (lihat tutorialnya di internet). Beberapa file library juga harus ditambahkan pada input linker di Visual C++. Anda juga dapat mengikuti paket training 3 hari fullday/workshop sehari Computer Vision pada Robot yang diadakan oleh CV Pusat e-Technology milik penulis, baik secara private maupun inhouse.



Gambar 8.1 Konfigurasi File Library

Sebagai contoh, buatlah program Win32 console application untuk menampilkan sebuah gambar di Windows.

Demo.cpp:

```
// Demo Program menampilkan gambar imut
// By Mr. Widodo 2010
// www.toko-robot.com      www.toko-elektronika.com

#include "stdafx.h"
#include "conio.h"
#include <cv.h>
#include <highgui.h>

int main(int argc, char** argv)
{
    IplImage* img = cvLoadImage(argv[1]);
    cvNamedWindow ("Contoh_DISPLAY_GAMBAR", CV_WINDOW_AUTOSIZE);
    cvShowImage("Contoh_DISPLAY_GAMBAR", img);
    cvWaitKey(0);
    cvReleaseImage(&img);
    cvDestroyWindow("Contoh_DISPLAY_GAMBAR");
}
```

Program di atas akan meload file .jpg yang kita berikan menggunakan cvLoadImage(), lalu ditampilkan menggunakan cvNamedWindow(). Setelah dikompilasi, eksekusi file .exe yang tercipta atau jalankan dengan perintah:

Demo fira.JPG

Akan tampil gambar berikut :



Gambar 8.2 Hasil program menampilkan image cute

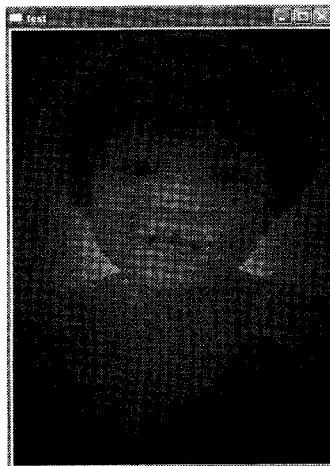
Bab 8: Computer Vision untuk Robot

Berikut contoh program memodifikasi/memroses gambar:

Gambar.cpp:

```
#include "stdafx.h"
#include <cv.h>
#include <highgui.h>
#include <math.h>
int main( int argc, char** argv ) {
    CvPoint center;
    double scale=-3;
    IplImage* image = argc==2 ? cvLoadImage(argv[1]) : 0;
    if(!image) return -1;
    center = cvPoint(image->width/2,image->height/2);
    for(int i=0;i<image->height;i++)
        for(int j=0;j<image->width;j++) {
            double dx=(double)(j-center.x)/center.x;
            double dy=(double)(i-center.y)/center.y;
            double weight=exp((dx*dx+dy*dy)*scale);
            uchar* ptr =&CV_IMAGE_ELEM(image,uchar,i,j*3);
            ptr[0] = cvRound(ptr[0]*weight);
            ptr[1] = cvRound(ptr[1]*weight);
            ptr[2] = cvRound(ptr[2]*weight); }
    cvSaveImage("copy.png", image );
    cvNamedWindow( "test", 1 );
    cvShowImage( "test", image );
    cvWaitKey();
    return 0;
}
```

Jalankan program dengan perintah *Gambar fira.JPG*. Hasilnya adalah seperti gambar berikut :



Gambar 8.2 Hasil program

Jika Anda ingin mendeteksi webcam dan mengambil image dari webcam tersebut, gunakan demo kode berikut ini, dimana fungsi yang umum digunakan adalah :

```
pCapture = cvCaptureFromCAM( CV_CAP_ANY );
```

WebcamCapture.cpp:

```
// Demo koneksi ke webcam dan simpan frame
// www.toko-elektronika.com 2010
#include "stdafx.h"
#include "stdio.h"
#include "string.h"
#include "cv.h"
#include "highgui.h"

int main(int argc, char ** argv)
{
    CvCapture * pCapture      = 0;
    IplImage *  pVideoFrame   = 0;
    int          i;
    char         filename[50];

    // Inisialisasi video capture
    pCapture = cvCaptureFromCAM( CV_CAP_ANY );
    if( !pCapture )
    {
        fprintf(stderr, "Gagal inisialisasi webcam\n");
        return -1;
    }

    // Ambil 3 frame video
    for(i=0; i<3; i++)
    {
        pVideoFrame = cvQueryFrame( pCapture );
        if( !pVideoFrame )
        {
            fprintf(stderr, "failed to get a video frame\n");
        }

        // tulis ke file
        sprintf(filename, "VideoFrame%d.jpg", i+1);
        if( !cvSaveImage(filename, pVideoFrame) )
        {
            fprintf(stderr, "failed to write image file %s\n", filename);
        }
    }

    // Terminasi video capture
    cvReleaseCapture( &pCapture );

    return 0;
}
```

Pengenalan Wajah

Haar Cascade Classifier

OpenCV adalah open source library computer vision yang memudahkan pemrograman deteksi wajah, face tracking, face recognition, kalman filtering, dan berbagai metode artificial intelligent.

OpenCV menggunakan sebuah tipe face detector yang disebut Haar Cascade Classifier. Gambar menunjukkan face detector berhasil bekerja pada sebuah gambar. Jika ada sebuah image (bisa dari file/live video), face detector menguji setiap lokasi image dan mengklasifikasinya sebagai "wajah" atau "bukan wajah". Klasifikasi dimisalkan sebuah skala fix untuk wajah, misal 50x50 pixel. Jika wajah pada image lebih besar atau lebih kecil dari pixel tersebut, classifier terus menerus jalan beberapa kali untuk mencari wajah pada gambar tersebut.

Classifier menggunakan data yang disimpan pada file XML untuk memutuskan bagaimana mengklasifikasi setiap lokasi image. OpenCV menggunakan 4 data XML untuk deteksi wajah depan dan 1 untuk wajah profile. Termasuk juga 3 file XML bukan wajah: 1 untuk deteksi full body, 1 untuk upper body, dan 1 untuk lower body. Anda harus memberitahukan classifier di mana menemukan file data yang akan dipakai. Salah satunya bernama haarcascade_frontalface_default.xml. Pada OpenCV, terletak pada folder:

Program_Files/OpenCV/data/haarcasades/haarcascade_frontalface_default.xml.

Konsep Pendekstian Wajah

OpenCV face detector menggunakan metode Paul Viola dan Michael Jones. Silahkan baca detail paper mereka di CD Program. Metode mengkombinasikannya adalah sebagai berikut:

- Fitur rectangular sederhana yang disebut fitur Haar.
- Integral image untuk deteksi fitur yang cepat.

- Metode machine learning AdaBoost.
- Sebuah pengklasifikasi cascade untuk mengkombinasikan banyak fitur secara efisien.

Fitur yang digunakan Viola dan Jones menggunakan bentuk gelombang Haar. Bentuk gelombang Haar adalah sebuah gelombang kotak. Pada 2 dimensi, gelombang kotak adalah pasangan persegi yang bersebelahan, 1 terang dan 1 gelap. Haar ditentukan oleh pengurangan pixel rata-rata daerah gelap dari pixel rata-rata daerah terang. Jika perbedaan di atas threshold (di-set selama learning), fitur tersebut dikatakan ada.

Implementasi Deteksi Wajah:

1. Variable CvHaarClassifierCascade * pCascade menyimpan data dari file XML. Untuk me-load data XML ke pCascade, Anda dapat menggunakan fungsi cvLoad(). cvLoad adalah fungsi umum untuk me-load data dari file yang membutuhkan 3 parameter input. Jika Anda membuat kode pada C, set parameter sisanya menjadi 0. Jika menggunakan C++, hilangkan parameter yang tidak digunakan.
2. Sebelum mendeteksi wajah pada image, Anda membutuhkan objek CvMemStorage. Detector akan mendaftar wajah yang terdetekti ke buffer. Yang harus Anda kerjakan adalah membuatnya:

```
pStorage=CvCreateMemStorage(0);
```

dan me-release-nya ketika telah selesai.

```
cvReleaseMemStorage(&pStorage);
```

3. Anda akan sering me-load data dari file. Tentu ada kemungkinan salah path. Sebaiknya berikan pengecekan untuk memastikan file di-load dengan benar.

```
if(!pInpImg || !pStorage || !pCascade)
{
printf ("Inisialisasi gagal \n");
}
exit (-1);
}
```

4. Untuk menjalankan detektor, panggil objek cvHaarDetect. Fungsi ini membutuhkan 7 parameter; 3 pertama adalah pointer image, XML data, dan memory buffer, sisanya di-set pada default C++.

```
pFaceRectSeq =cvHaarDetectObjects
(pInpImg, pCascade, pStorage,
1.1, //tingkatkan skala pencarian dengan 10% tiap passing
3, //drop group yang kurang dari 3 deteksi
CV_HAAR_DO_CANNY_PRUNNING //skip region yang tidak berisi wajah
cvSize(0,)); //gunakan XML default untuk skala pencarian terkecil.
```

5. Untuk membuat display Window gunakan cvNamedWindow seperti berikut:

```
cvNamedWindow ("Haar Window", CV_WINDOW_AUTOSIZE);
Untuk memasukkan image ke display, panggil fungsi cvShowImage() dengan
nama yang telah dibuat pada window dan nama image yang ingin
ditampilkan.
```

Berikut ini kode lengkapnya:

```
DetectFace.cpp:
// Hak Cipta cognitics.com
#include "stdafx.h"
#include <stdio.h>
#include "cv.h"
#include "highgui.h"

// ****
#define OPENCV_ROOT  "C:/Program Files/OpenCV"
// ****

void displayDetections(IplImage * pInpImg, CvSeq * pFaceRectSeq);

int main(int argc, char** argv)
{
    // variables
    IplImage * pInpImg = 0;
    CvHaarClassifierCascade * pCascade = 0; // face detector
    CvMemStorage * pStorage = 0;           // memory for detector to use
    CvSeq * pFaceRectSeq;                // memory-access interface

    // pengecekan
    if(argc < 2)
    {
        printf("Missing name of image file!\n"
               "Usage: %s <imagefilename>\n", argv[0]);
        exit(-1);
    }

    // initializations
    pInpImg = (argc > 1) ? cvLoadImage(argv[1], CV_LOAD_IMAGE_COLOR) : 0;
    pStorage = cvCreateMemStorage(0);
```

```

pCascade = (CvHaarClassifierCascade *)cvLoad
((OPENCV_ROOT"/data/haarcascades/haarcascade_frontalface_default.xml"),
 0, 0, 0);

// validate that everything initialized properly
if( !pInpImg || !pStorage || !pCascade )
{
    printf("Initialization failed: %s\n",
        (!pInpImg)? "can't load image file" :
        (!pCascade)? "can't load haar-cascade -- "
                    "make sure path is correct" :
        "unable to allocate memory for data storage", argv[1]);
    exit(-1);
}

// detect faces in image
pFaceRectSeq = cvHaarDetectObjects
    (pInpImg, pCascade, pStorage,
     1.1,                               // increase search scale by 10% each
pass
     3,                                // merge groups of three detections
     CV_HAAR_DO_CANNY_PRUNING,          // skip regions unlikely to contain
a face
     cvSize(40,40));                   // smallest size face to detect =
40x40

// tampilan wajah yang terdeteksi
displayDetections(pInpImg, pFaceRectSeq);

// clean up and release resources
cvReleaseImage(&pInpImg);
if(pCascade) cvReleaseHaarClassifierCascade(&pCascade);
if(pStorage) cvReleaseMemStorage(&pStorage);

return 0;
}

void displayDetections(IplImage * pInpImg, CvSeq * pFaceRectSeq)
{
    const char * DISPLAY_WINDOW = "Haar Window";
    int i;

    // create a window to display detected faces
    cvNamedWindow(DISPLAY_WINDOW, CV_WINDOW_AUTOSIZE);

    // draw a rectangular outline around each detection
    for(i=0;i<(pFaceRectSeq->total:0); i++ )
    {
        CvRect* r = (CvRect*)cvGetSeqElem(pFaceRectSeq, i);
        CvPoint pt1 = { r->x, r->y };
        CvPoint pt2 = { r->x + r->width, r->y + r->height };
        cvRectangle(pInpImg, pt1, pt2, CV_RGB(0,255,0), 3, 4, 0);
    }

    // tampilkan
    cvShowImage(DISPLAY_WINDOW, pInpImg);
    cvWaitKey(0);
    cvDestroyWindow(DISPLAY_WINDOW);
}

```



Gambar 8.3 Wajah cantik yang terdeteksi

Tracking Wajah

OpenCV face tracker menggunakan algoritma CamShift. Camfsit terdiri dari beberapa langkah :

1. Membuat histogram warna untuk merepresentasikan wajah.
Camshift merepresentasikan wajah yang di-track sebagai histogram nilai warna. Ketinggian setiap bar berwarna mengindikasikan berapa banyak pixel pada daerah image yang memiliki "hue". Hue adalah satu dari 3 nilai yang menjelaskan warna pixel pada HSV (Hue, Saturation, Value).
2. Menghitung probabilitas wajah untuk setiap pixel pada frame video yang diterima.
3. Menggeser lokasi dari persegi wajah pada setiap frame video.

4. Camshift menggeser estimasinya dari lokasi wajah, membuat tepat terpusat pada area dengan konsentrasi tinggi dari pixel terang pada image face probability. Ia akan mencari lokasi baru tersebut dengan memulai pada lokasi sebelumnya dan menghitung pusat gravitasi dari nilai face-probability dalam sebuah kotak. Lalu, menggeser kotak hingga kotaknya melewati pusat gravitasi. Dilakukan beberapa kali ke pusat kotak. Fungsi cvCamShift() mengimplementasikan langkah untuk menggeser ke lokasi baru. Proses shifting kotak untuk menghubungkan dengan pusat gravitasi berdasarkan algoritma "Mean Shift" oleh Dorin Comaniciu.
5. Menghitung ukuran dan sudut.

Metode continuosly adaptive digunakan, bukan hanya "Mean Shift" karena ia juga menyesuaikan ukuran dan sudut dari kotak rectangle setiap kali *shifting* dengan cara skala dan orientasi yang terbaik pada pixel face-probability di dalam lokasi kotak.

Berikut ini contoh kodennya:

```
FaceTrack.cpp
// Demo Tracking wajah
// Hak Cipta Cognotics.com
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include "capture.h"
#include "capture.c"
#include "facedet.h"
#include "facedet.c"
#include "camshift_wrapper.h"
#include "camshift_wrapper.c"

const char * DISPLAY_WINDOW = "DisplayWindow";
#define OPENCV_ROOT "C:/Program Files/OpenCV"

//// Global variables
IplImage * pVideoFrameCopy = 0;

//// Definisi fungsi
int initAll();
void exitProgram(int code);
void captureVideoFrame();

void main( int argc, char** argv )
{
    CvRect * pFaceRect = 0;
    if( !initAll() ) exitProgram(-1);
```

Bab 8: Computer Vision untuk Robot

```
// Capture and display video frames until a face
// is detected
while( 1 )
{
    // mencari wajah pada next frame
    captureVideoFrame();
    pFaceRect = detectFace(pVideoFrameCopy);

    // Tampilkan image
    cvShowImage( DISPLAY_WINDOW, pVideoFrameCopy );
    if( (char)27==cvWaitKey(1) ) exitProgram(0);

    // exit loop when a face is detected
    if(pFaceRect) break;
}

// initialize tracking
startTracking(pVideoFrameCopy, pFaceRect);

// Track the detected face using CamShift
while( 1 )
{
    CvBox2D faceBox;
    CvPoint facePoint;

    // ambil next frame video
    captureVideoFrame();

    // track wajah pada video frame
    faceBox = track(pVideoFrameCopy);

    // Bentuk elips
    cvEllipseBox(pVideoFrameCopy, faceBox,
    CV_RGB(50,255,0), 2, CV_AA, 0 );
    cvShowImage( DISPLAY_WINDOW, pVideoFrameCopy );
    if( (char)27==cvWaitKey(1) ) break;
}

exitProgram(0);
}

int initAll()
{
    if( !initCapture() ) return 0;
    if( !initFaceDet(OPENCV_ROOT
                      "/data/haarcascades/haarcascade_frontalface_default.xml"))
        return 0;

    // Buat jendela display
    cvNamedWindow( DISPLAY_WINDOW, 1 );

    // Initialize tracker
    captureVideoFrame();
    if( !createTracker(pVideoFrameCopy) ) return 0;

    // Set parameter Camshift
    setVmin(60);
    setSmin(50);

    return 1;
}
```

```

void exitProgram(int code)
{
    // Release resources
    cvDestroyWindow( DISPLAY_WINDOW );
    cvReleaseImage( &pVideoFrameCopy );

    // Release resources
    closeCapture();
    closeFaceDet();
    releaseTracker();

    exit(code);
}

void captureVideoFrame()
{
    printf ("Data : %d\n");
    // Capture the next frame
    IplImage * pVideoFrame = nextVideoFrame();
    if( !pVideoFrame ) exitProgram(-1);

    // Copy it to the display image, inverting it if needed
    if( !pVideoFrameCopy )
        pVideoFrameCopy = cvCreateImage( cvGetSize(pVideoFrame), 8, 3
    );
    cvCopy( pVideoFrame, pVideoFrameCopy, 0 );
    pVideoFrameCopy->origin = pVideoFrame->origin;

    if( 1 == pVideoFrameCopy->origin ) // 1 means the image is inverted
    {
        cvFlip( pVideoFrameCopy, 0, 0 );
        pVideoFrameCopy->origin = 0;
    }
}

```



Gambar 8.6 Hasil deteksi dan tracking wajah Mr. Widodo

Latihan

1. Untuk mahir computer vision pada robot, tidak ada cara lain selain Anda mengerjakan tugas yang penulis berikan. Buatlah program Robot Line Follower yang mengikuti wajah orang di depannya.
2. Buatlah program edge detection menggunakan OpenCV.
3. Buatlah program Threshold dan ROI menggunakan OpenCV.
4. Jelaskan cara kerja Haar Cascade Classifier.
5. Buatlah program dan file library XML untuk deteksi objek buatan Anda, misalnya deteksi gelas dan gunting. Anda harus membuat file positif berupa file gambar yang ingin dideteksi minimal 12 gambar dengan berbagai pose dan file negatif berisi gambar yang bukan ingin dideteksi. Lalu, lakukan pembuatan file vector dengan contoh perintah:

```
createsamples.exe -info positiveSample/info.txt -vec data/vector.vec -  
num 18 -w 20 -h 20
```

Kemudian melakukan training dengan perintah :

```
haartraining.exe -data data/cascade -vec data/vector.vec -bg  
negativeSample/infofile.txt -npos 12 -nneg 2 -mem 1000 -mode ALL -  
w 20 -h 20 -nonsym
```

Lalu buat file XML dengan perintah:

```
haarconv.exe data output.xml 20 20
```



Gambar 8.7 Hasil deteksi dan tracking obyek buatan XML

Bab 9

Robot Pelayan Humanoid

Pendahuluan

Robot pelayan adalah robot yang mampu melayani manusia, dalam hal ini adalah robot pelayan pada rumah makan/cafe. Berbekal materi sebelumnya dan kerja keras Anda, robot yang dipaparkan di sini harus Anda buat hingga sukses jika ingin benar benar mewujudkan robot humanoid yang mirip manusia.

Rancangan Robot Pelayan

Arsitektur Robot Pelayan

Perkembangan pesat teknologi robot telah menuntut hadirnya robot cerdas yang mampu melengkapi dan membantu pekerjaan manusia. Salah satu harapan dari perkembangan teknologi robot adalah hadirnya robot pelayan di sekitar kita. Robot tipe ini tentunya harus mampu berinteraksi sosial dengan manusia, antara lain dapat mengenal wajah, suara, dan berbicara. Di Indonesia sendiri dapat dikatakan belum ada robot pelayan yang benar-benar telah diterapkan pada dunia komersial.

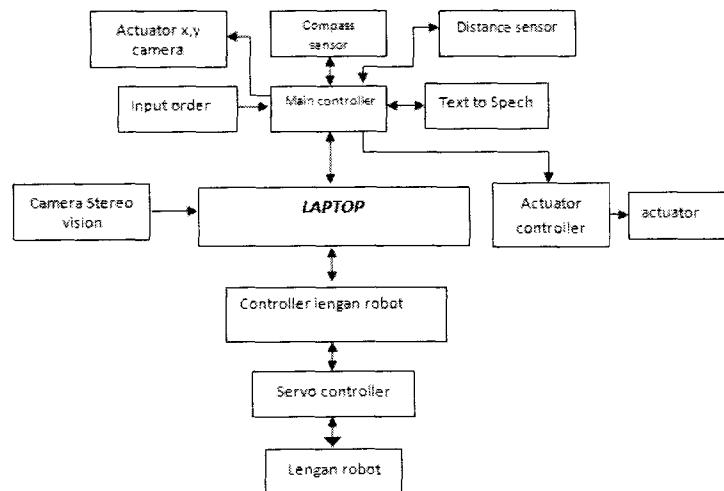
Bab 9: Robot Pelayan Humanoid

Oleh karena itu, penulis tertarik untuk memotivasi pembaca agar membuat robot ini.

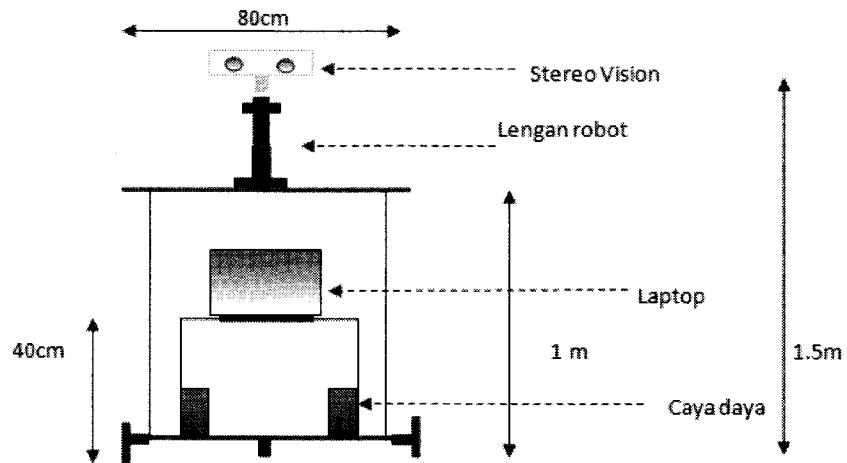
Robot yang akan Anda buat berbasiskan 2 buah mikrokontroler Basic Stamp dan AVR Atmega8535 untuk mengendalikan webcam serta aktuator. Perangkat yang digunakan secara detail adalah:

- 2 buah mikrokontroler Basic Stamp dan 1 buah mikrokontroler AVR
- Modul Text to speech dari Parallax
- SG6 Arm Robot 5 DOF Crustcrawler
- 2 buah servo continuous Parallax
- 2 buah webcam standar
- Sensor jarak PING
- Motor dan roda robot serta komponen pendukung

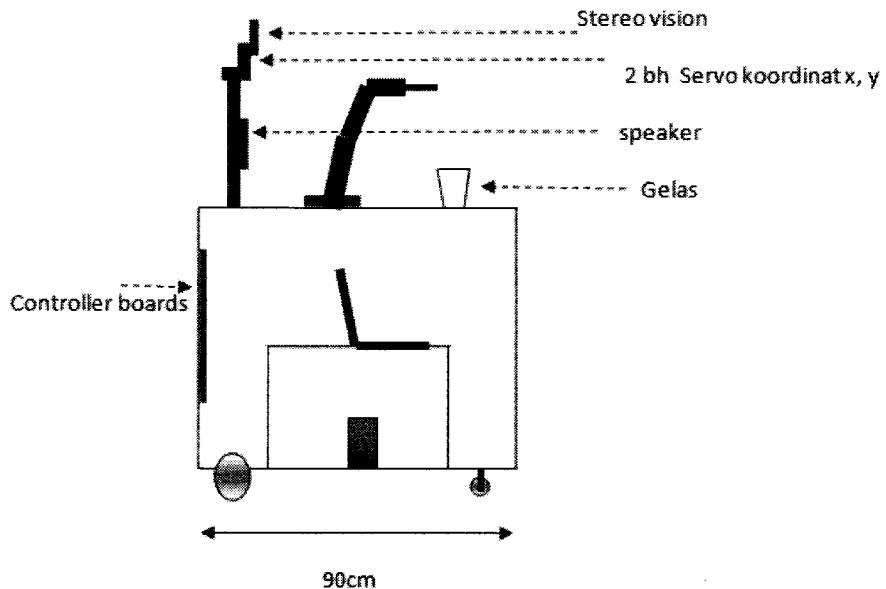
Berikut ini adalah rancangannya. Intinya, robot mendekksi wajah seseorang menggunakan webcam, robot menerima input order dari switch manual atau speech recognition. Setelah itu, robot dapat memberikan order orang tersebut menggunakan arm robot dengan berjalan menggunakan kaki atau roda. Berikut arsitektur robot penulis yang bernama Srikandi.



Gambar 9.1 Arsitektur robot pelayan Srikandi (servant robot, hak cipta penulis)



Gambar 9.2 Tampak depan robot pelayan Srikandi



Gambar 9.3 Tampak samping robot pelayan Srikandi

Anda dapat memesan demo robot ini dalam bentuk sudah jadi (robot people follower) tanpa lengan robot seharga 9 juta rupiah. Jika lengkap dengan lengan robot, harganya sekitar 30 juta di luar laptop/PC.

Perangkat Lunak Robot Pelayan

Untuk mengendalikan Basic Stamp digunakan bahasa PBASIC. Untuk mengendalikan webcam dan image processing di PC, digunakan OpenCV. Sedangkan untuk mengendalikan AVR digunakan CodeVision C AVR. Untuk antarmuka antara PC dengan pengguna (User Interface) digunakan Visual C# .Net. Agar robot dapat berbicara, digunakan kit Text to Speech dari Parallax. Contoh robot berbicara ini sudah dibahas penulis pada buku sebelumnya, yaitu 10 Proyek Robot Spektakuler terbitan Elex Media Komputindo.

Face Recognition

Konsep Dasar

Untuk membuat robot pelayan dari arsitektur di atas, pertama Anda harus membuat program face recognition yang akan mengenali wajah pemesan dan object recognition untuk mengenali gelas atau bukan. Object recognition dapat dibuat dengan membuat file library XML. Face recognition di sini menggunakan Eigenspace dan PCA, yaitu suatu teknik yang umum dikenal jika pemula ingin belajar pengenalan dan identifikasi wajah. Penulis mengharapkan Anda membaca paper asli dari penemu teknik ini di CD Program. Penulis juga mengharapkan Anda mempelajari Machine Learning. Program dimulai dengan training, di mana program akan melakukan proses pembelajaran. Selanjutnya, dilakukan pengetesan image, yang kemudian diteruskan dengan proses pengenalan wajah. Buatlah proyek baru dan beri nama FaceEigen sebagai berikut:

FaceEigen.cpp:

```
// Program demo Face recognition berbasis Eigenspace
// Hak Cipta Cognotics.com

#include "stdafx.h"
#include <stdio.h>
#include <string.h>
#include "cv.h"
#include "cvaux.h"
#include "highgui.h"

//// Global variables
IplImage ** faceImgArr      = 0; // array of face images
CvMat * personNumTruthMat = 0; // array of person numbers
int nTrainFaces            = 0; // the number of training images
int nEigens                 = 0; // the number of eigenvalues
IplImage * pAvgTrainImg    = 0; // the average image
IplImage ** eigenVectArr    = 0; // eigenvectors
CvMat * eigenValMat        = 0; // eigenvalues
CvMat * projectedTrainFaceMat = 0; // projected training faces

//// Function prototypes
void learn();
void recognize();
void doPCA();
void storeTrainingData();
int loadTrainingData(CvMat ** pTrainPersonNumMat);
int findNearestNeighbor(float * projectedTestFace);
int loadFaceImgArray(char * filename);
void printUsage();

void main( int argc, char** argv )
{
    if( !strcmp(argv[1], "train") )
    {
        learn();
    }
    if( !strcmp(argv[1], "test") )
    {
        recognize();
    }
}

void learn()
{
    int i, offset;

    // load training data
    nTrainFaces = loadFaceImgArray("train.txt");
    if( nTrainFaces < 2 )
    {
        fprintf(stderr,
                "Need 2 or more training faces\n"
                "Input file contains only %d\n", nTrainFaces);
        return;
    }
}
```

Bab 9: Robot Pelayan Humanoid

```
// do PCA on the training faces
doPCA();

// project the training images onto the PCA subspace
projectedTrainFaceMat = cvCreateMat( nTrainFaces, nEigens, CV_32FC1 );
offset = projectedTrainFaceMat->step / sizeof(float);
for(i=0; i<nTrainFaces; i++)
{
    //int offset = i * nEigens;
    cvEigenDecompose(
        faceImgArr[i],
        nEigens,
        eigenVectArr,
        0, 0,
        pAvgTrainImg,
        //projectedTrainFaceMat->data.fl + i*nEigens),
        projectedTrainFaceMat->data.fl + i*offset);
}

// store the recognition data as an xml file
storeTrainingData();
}

void recognize()
{
    int i, nTestFaces = 0;           // the number of test images
    CvMat * trainPersonNumMat = 0;   // the person numbers during training
    float * projectedTestFace = 0;

    // load test images and ground truth for person number
    nTestFaces = loadFaceImgArray("test.txt");
    //printf("%d test faces loaded\n", nTestFaces);

    // load the saved training data
    if( !loadTrainingData( &trainPersonNumMat ) ) return;

    // project the test images onto the PCA subspace
    projectedTestFace = (float *)cvAlloc( nEigens*sizeof(float) );
    for(i=0; i<nTestFaces; i++)
    {
        int iNearest, nearest, truth;

        // project the test image onto the PCA subspace
        cvEigenDecompose(
            faceImgArr[i],
            nEigens,
            eigenVectArr,
            0, 0,
            pAvgTrainImg,
            projectedTestFace);

        iNearest = findNearestNeighbor(projectedTestFace);
        truth    = personNumTruthMat->data.i[i];
        nearest  = trainPersonNumMat->data.i[iNearest];

        printf("Wajah ini mirip dengan = %d, Truth = %d\n", nearest, truth);
    }
}
```

```

int loadTrainingData(CvMat ** pTrainPersonNumMat)
{
    CvFileStorage * fileStorage;
    int i;

    // create a file-storage interface
    fileStorage = cvOpenFileStorage( "facedata.xml", 0, CV_STORAGE_READ );
    if( !fileStorage )
    {
        fprintf(stderr, "Can't open facedata.xml\n");
        return 0;
    }

    nEigens = cvReadIntByName(fileStorage, 0, "nEigens", 0);
    nTrainFaces = cvReadIntByName(fileStorage, 0, "nTrainFaces", 0);
    *pTrainPersonNumMat = (CvMat *)cvReadByName(fileStorage, 0,
"trainPersonNumMat", 0);
    eigenValMat = (CvMat *)cvReadByName(fileStorage, 0, "eigenValMat",
0);
    projectedTrainFaceMat = (CvMat *)cvReadByName(fileStorage, 0,
"projectedTrainFaceMat", 0);
    pAvgTrainImg = (IplImage *)cvReadByName(fileStorage, 0, "avgTrainImg",
0);
    eigenVectArr = (IplImage **)cvAlloc(nTrainFaces*sizeof(IplImage *));
    for(i=0; i<nEigens; i++)
    {
        char varname[200];
        sprintf( varname, "eigenVect_%d", i );
        eigenVectArr[i] = (IplImage *)cvReadByName(fileStorage, 0, varname, 0);
    }

    // release the file-storage interface
    cvReleaseFileStorage( &fileStorage );

    return 1;
}

void storeTrainingData()
{
    CvFileStorage * fileStorage;
    int i;

    // create a file-storage interface
    fileStorage = cvOpenFileStorage( "facedata.xml", 0, CV_STORAGE_WRITE
);

    // store all the data
    cvWriteInt( fileStorage, "nEigens", nEigens );
    cvWriteInt( fileStorage, "nTrainFaces", nTrainFaces );
    cvWrite(fileStorage, "trainPersonNumMat", personNumTruthMat,
cvAttrList(0,0));
    cvWrite(fileStorage, "eigenValMat", eigenValMat, cvAttrList(0,0));
    cvWrite(fileStorage, "projectedTrainFaceMat", projectedTrainFaceMat,
cvAttrList(0,0));
    cvWrite(fileStorage, "avgTrainImg", pAvgTrainImg, cvAttrList(0,0));
    for(i=0; i<nEigens; i++)
    {
        char varname[200];
        sprintf( varname, "eigenVect_%d", i );
        cvWrite(fileStorage, varname, eigenVectArr[i],
cvAttrList(0,0));
    }
}

```

Bab 9: Robot Pelayan Humanoid

```
// release the file-storage interface
cvReleaseFileStorage( &fileStorage );
}

int findNearestNeighbor(float * projectedTestFace)
{
    //double leastDistSq = 1e12;
    double leastDistSq = DBL_MAX;
    int i, iTrain, iNearest = 0;

    for(iTrain=0; iTrain<nTrainFaces; iTrain++)
    {
        double distSq=0;

        for(i=0; i<nEigens; i++)
        {
            float d_i =
                projectedTestFace[i] -
                projectedTrainFaceMat->data.fl[iTrain*nEigens +
i];

            distSq += d_i*d_i; // Euclidean
        }

        if(distSq < leastDistSq)
        {
            leastDistSq = distSq;
            iNearest = iTrain;
        }
    }

    return iNearest;
}

void doPCA()
{
    int i;
    CvTermCriteria calcLimit;
    CvSize faceImgSize;

    // set the number of eigenvalues to use
    nEigens = nTrainFaces-1;

    // allocate the eigenvector images
    faceImgSize.width = faceImgArr[0]->width;
    faceImgSize.height = faceImgArr[0]->height;
    eigenVectArr = (IplImage**)cvAlloc(sizeof(IplImage*) * nEigens);
    for(i=0; i<nEigens; i++)
        eigenVectArr[i] = cvCreateImage(faceImgSize, IPL_DEPTH_32F, 1);

    // allocate the eigenvalue array
    eigenValMat = cvCreateMat( 1, nEigens, CV_32FC1 );

    // allocate the averaged image
    pAvgTrainImg = cvCreateImage(faceImgSize, IPL_DEPTH_32F, 1);

    // set the PCA termination criterion
    calcLimit = cvTermCriteria( CV_TERMCRIT_ITER, nEigens, 1);

    // compute average image, eigenvalues, and eigenvectors
```

```

cvCalcEigenObjects(
    nTrainFaces,
    (void*)faceImgArr,
    (void*)eigenVectArr,
    CV_EIGOBJ_NO_CALLBACK,
    0,
    0,
    &calcLimit,
    pAvgTrainImg,
    eigenValMat->data.fl);

cvNormalize(eigenValMat, eigenValMat, 1, 0, CV_L1, 0);
}

int loadFaceImgArray(char * filename)
{
    FILE * imgListFile = 0;
    char imgFilename[512];
    int iFace, nFaces=0;

    // open the input file
    if( !(imgListFile = fopen(filename, "r")) )
    {
        fprintf(stderr, "Can't open file %s\n", filename);
        return 0;
    }

    // count the number of faces
    while( fgets(imgFilename, 512, imgListFile) ) ++nFaces;
    rewind(imgListFile);

    // allocate the face-image array and person number matrix
    faceImgArr      = (IplImage **)cvAlloc( nFaces*sizeof(IplImage *) );
    personNumTruthMat = cvCreateMat( 1, nFaces, CV_32SC1 );

    // store the face images in an array
    for(iFace=0; iFace<nFaces; iFace++)
    {
        // read person number and name of image file
        fscanf(imgListFile,
               "%d %s", personNumTruthMat->data.i+iFace, imgFilename);

        // load the face image
        faceImgArr[iFace] = cvLoadImage(imgFilename, CV_LOAD_IMAGE_GRAYSCALE);

        if( !faceImgArr[iFace] )
        {
            fprintf(stderr, "Can't load image from %s\n", imgFilename);
            return 0;
        }
    }

    fclose(imgListFile);
    return nFaces;
}

void printUsage()
{
    printf("Usage: eigenface <command>\n",
           "  Valid commands are\n"

```

Bab 9: Robot Pelayan Humanoid

```
    "      train\n"
    "      test\n");
}
```

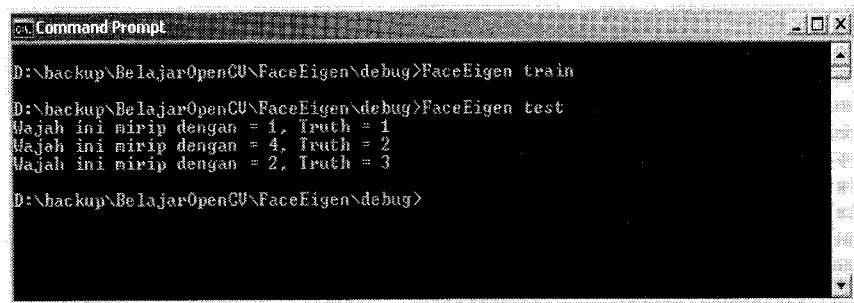
Siapkan basis data file gambar berekstension .pgm dengan aneka posisi wajah. Berikan data percobaan untuk training.txt, misalnya:

```
1 s1/1.pgm
2 s2/1.pgm
3 s3/1.pgm
4 s1/2.pgm
5 s2/2.pgm
6 s3/2.pgm
```

Berikan data percobaan untuk test.txt:

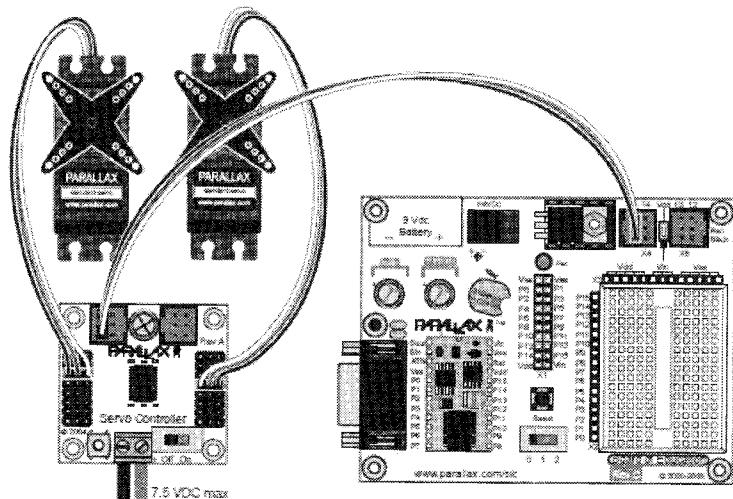
```
1 s1/1.pgm
2 s1/2.pgm
3 s2/1.pgm
```

Jika program ini dijalankan dengan melakukan training terlebih dahulu, lalu test, maka akan terlihat bahwa gambar 1, 2 , dan 3 sesuai dengan basis data 1, 4, dan 2.



Gambar 9.4 Hasil face recognition

Jika program diatas telah berhasil dikembangkan, langkah berikutnya adalah membuat program object recognition. Setelah itu, membuat lengan robot yang dapat mengambil dan menerima gelas. Untuk mengendalikan servo pada lengan robot, paling mudah menggunakan Parallax Servo Controller yang dapat memperoleh posisi servo.



Gambar 9.5 Hubungan Parallax Servo Controller

Berikut adalah demo mencari versi kit dari Parallax Servo Controller, di mana output mikrokontroler berupa data serial yang dihubungkan ke pin 15.

Ver.bsp:

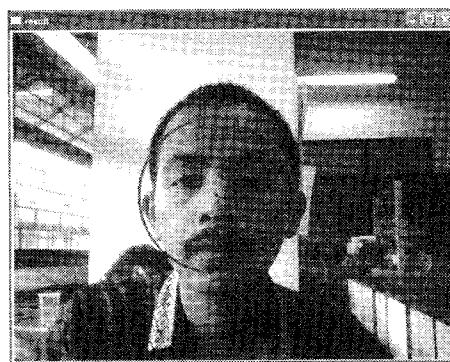
```
' {$STAMP BS2p}
' {$PBASIC 2.5}
Sdat PIN 15
baud CON 1021
buff VAR Byte(3)

findPSC:
DEBUG "Mencari PSC", CR
SEROUT Sdat, baud + $8000, ["!SCVER?",CR]
SERIN Sdat, baud, 500, findPSC, [STR buff\3]
DEBUG "PSC ver:", buff(0), buff(1), buff(2), CR
STOP
```

Jika lengan robot Anda telah sukses bergerak sesuai program demo, langkah terakhir adalah mengembangkan dan mengintegrasikan program pengenal wajah dan menyimpan informasi wajah tersebut beserta namanya. Setelah robot mengenal dan mengingat pesanan dari pelanggan tersebut, gelas yang berisi minuman pemesan harus diberikan oleh robot dengan berjalan menggunakan roda. Robot juga harus berkomunikasi secara efektif menggunakan modul Text to Speech yang telah dipasang. Selamat Mencoba ☺.

Latihan

1. Jelaskan cara kerja Face recognition serta beberapa teknik yang umum digunakan.
2. Kembangkanlah demo kode pada bab ini untuk mewujudkan robot pelayan humanoid menggunakan stereo vision berbasis webcam yang mampu mendeteksi tingkat kedalaman dari suatu image.
3. Kembangkan juga robot yang mampu mengikuti wajah seseorang (people follower) seperti gambar di bawah:



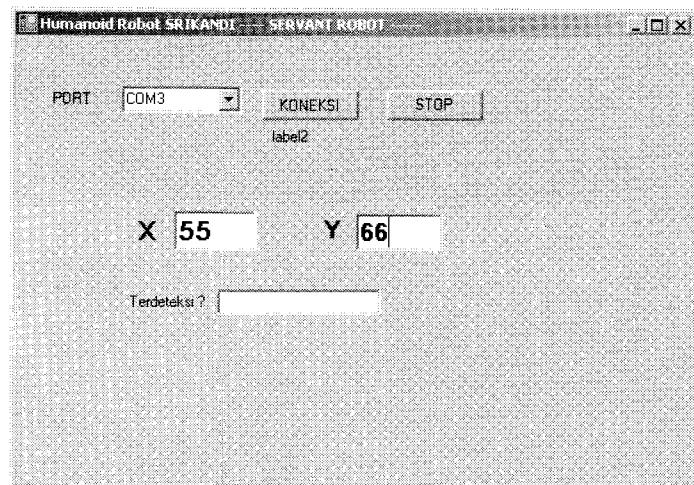
Gambar 9.6 Hasil deteksi dan tracking wajah robot people follower

Untuk mengoptimalkan pengenalan wajah, gunakan teknik penyimpanan 3 sampel wajah per orang.

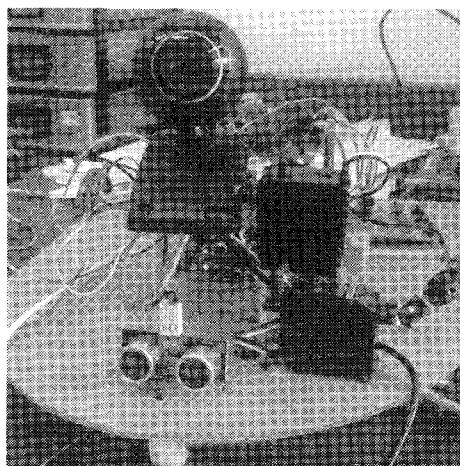


Gambar 9.7 Basis data wajah yang disimpan per orang dengan Eigenspace

Berikut ini contoh user interface berbasis C# untuk mendeteksi posisi wajah yang terdeteksi



Gambar 9.8 User interface pada PC

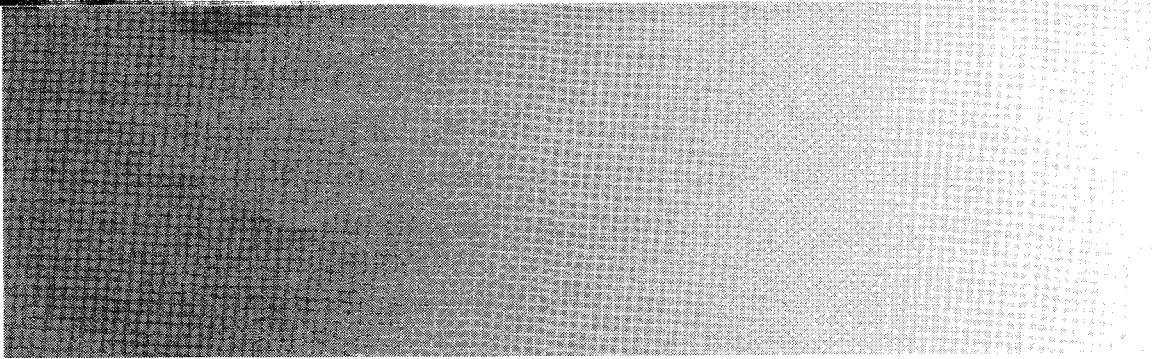


Gambar 9.9 Prototipe Robot people follower dengan webcam

Daftar Acuan

1. Budiharto, Widodo, *10 Proyek Robot Spektakuler*, Elex Media Komputindo, 2008.
2. Budiharto, Widodo, *Robotika, Teori dan Implementasi*, Andi Offset Yogyakarta, 2009.
3. Budiharto, Widodo, *Membuat Robot Cerdas*, Elex Media Komputindo, 2009.
4. Budiharto, Widodo, *12 Proyek Mikrokontroler untuk Pemula*, Elex Media Komputindo, 2006.
5. Team IE, *Panduan Praktis Mikrokontroler Keluarga AVR*, Innovative Electronic Surabaya, 2006.
6. Budiharto, Widodo, *Perancangan Sistem dan Aplikasi Mikrokontroler*, Elex Media Komputindo, 2005
7. Braunl, Thomas, *Embedded Robotics*, Springer-Verlag berlin Heidelberg, 2003.
8. Budiharto, Widodo, *Visual Basic .Net 2005*, Andi Offset Yogyakarta, 2006.
9. Budiharto, Widodo, *Panduan Praktikum Mikrokontroler AVR ATmega16*, Elex Media Komputindo, 2008.
10. Manual kit Delta Electronic

11. Manual Crustcrawler dan Lynxmotion
12. www.toko-elektronika.com
13. www.cognotics.com
14. www.atmel.com
15. www.sourceforge.net/projects/OpenCV
16. www.parallax.com
17. www.mcselec.com
18. www.ti.com
19. www.wikipedia.org
20. www.maxim-ic.com
21. www.beyondlogic.com
22. www.hpinfotech.ro
23. www.diptrace.com
24. www.cmucam.org
25. www.crustcrawler.com



Lampiran

Daftar Kit Penunjang Praktek

Kit	Penjelasan
ST-8535 USB Version	Kit mikro AVR Atmega8535 dengan programmer berbasis USB
DEC-xx	Encoder Kit
D-Sonar	Ultrasonic Interface, kompatibel SRF-04 + UART Protocol
D-Voice 04	Voice Interface dengan durasi 8 / 16 menit
Delta IR Line Sensing	Penjejak garis berupa sungut yang dapat diatur tebal tipis maupun responnya
TSAL4400	LED Inframerah daya tinggi
TOPS030TB2	Phototransistor dengan lapisan pelindung
DSF-01	Penjejak garis tunggal dengan jarak deteksi yang variatif
Hexapod Mechanic	Mekanik robot berkaki 6 yang digerakkan oleh dua motor servo continuous
TCS-230	Sensor warna dengan keluaran frekwensi
DCLI-230	Modul Sensor Warna dengan LED putih dan kotak hitam pelindung cahaya
DST-51	Sistem Minimum AT89S51 dengan antarmuka LCD, PC Keyboard dan 32 LED Display
ST-51	Sistem Minimum AT89S51 yang paling ekonomis
IR-8510	Modul Infrared Transmitter & Receiver
TSOP4838	Modul Infrared Receiver
LCD M1632	LCD Matrix 16 x 2

DX-24	Modul antarmuka TRW-24 dengan sistem mikrokontroler
D-Joy Controller	Modul antarmuka 4 Joystick Playstation dengan keluaran UART
D-Joy Controller (Wireless Version)	Modul antarmuka 4 Joystick Playstation dengan keluaran RF ke DX-24
DU-ISP V2.0	Downloader ISP untuk berbagai macam mikrokontroler Atmel (Kompatibel Vista)
Delta DC Driver	Modul Pengendali 2 buah Motor DC secara Half Bridge + pelindung kondisi inhibit
Delta Robo CPU	Modul Otak Robot yang dilengkapi konektor-konektor untuk sensor dan kompatibel Delta DC Driver
SST-06	Sub System Stepper Interface yang dapat mengendalikan motor stepper bipolar maupun unipolar secara manual atau berdasar perintah dari UART
DSR-08	Smart Servo Controller yang dilengkapi metode perhitungan nilai encoder dan deteksi kondisi kegagalan mekanik
L293D	IC Driver motor dengan kapasitas arus maksimum 2A
HS-422	Motor Servo Hitec dengan torsi 4.1kg /cm
HS-645MG	Motor Servo Hitec dengan torsi 9.6kg /cm metal gear
HS-805BB	Motor Servo Hitec dengan torsi 24.7kg /cm
GWS S04BB	Motor Servo GWS dengan torsi 13 kg/cm
DCS-01	Sensor arus dengan kapasitas maksimum 20A
Spider Leg Mechanic	Kaki Robot laba-laba dengan ruas-ruasnya
DU-232	Antarmuka USB to RS232
Gripper	Mekanik penjepit pada lengan robot
CB-232	Kabel serial dilengkapi RS232 to TTL Converter
Delta Robo Arm	Mekanik lengan robot dengan 5 sumbu gerakan
KIT ROBOT PELENGKAP UTAMA	
ROBOT BRAT	
SG6 Arm Robot 5 axis	
Lego Mindstrom NXT	
Robot People Follower berbasis Vision	
Silahkan order di www.toko-elektronika.com	

Tentang Penulis

Ir. Paulus Andi Nalwan adalah praktisi dan konsultan robotika dan embedded system ternama di Indonesia. Ia sangat berpengalaman dalam pengembangan aplikasi elektronika berbasis mikrokontroler dan memiliki toko elektronika online di www.delta-electronic.com. Anda dapat menghubungi penulis melalui paulus@delta-electronic.com.

Widodo Budiharto adalah praktisi dan akademisi di bidang TI di BINUS University dan konsultan senior untuk robotika dan embedded system. Ia memiliki situs elektronika online www.toko-elektronika.com. Penulis dapat dihubungi di widodo@widodo.com.

