tingkat dasar

teknik rancang bangun Robot

- petunjuk praktis membangun robot
- dasar-dasar robotika dengan bahan-bahan murah
- pemrograman yang mudah dipahami

Andi Nalwan

Teknik Rancang Bangun Robot

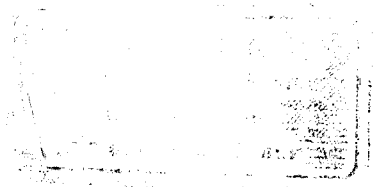
Tingkat Dasar

- Petunjuk Praktis Membangun Robot
- Dasar-dasar Robotika dengan Bahan-bahan Murah
- Pemrograman yang Mudah Dipahami

Andi Nalwan

Penerbit ANDI Yogyakarta

Teknik Rancang Bangun Robot
Oleh: Andi Nalwan



Editor : Nikodemus WK
Setting : Sri Sulistiyani
Desain Cover : Dan-dut
Korektor : Erang

Penerbit: C.V ANDI OFFSET (Penerbit ANDI)
Jl. Beo 38-40, Telp. (0274) 561881 (Hunting), Fax. (0274) 588282 Yogyakarta 55281

Percetakan: ANDI OFFSET
Jl. Beo 38-40, Telp. (0274) 561881 (Hunting), Fax. (0274) 588282 Yogyakarta 55281

Perpustakaan Nasional: Katalog dalam Terbitan (KDT)

Nalwan, Andi
Teknik Rancang Bangun Robot / Andi Nalwan;
– Ed. 1. – Yogyakarta: ANDI,
21 20 19 18 17 16 15 14 13 12
viii + 232 hlm.; 16 x 23 Cm.
10 9 8 7 6 5 4 3 2 1
ISBN: 978 – 979 – 29 – 3097 – 9
I. Judul
1. Robotics

DDC'21 : 629.892

KATA PENGANTAR

Dunia robot dewasa ini semakin berkembang. Banyak fungsi yang biasa dilakukan oleh manusia saat ini mulai dikerjakan oleh robot, seperti pada fungsi pemindahan barang, pengendalian kendaraan, atau pemisahan barang pada roda berjalan. Kondisi tersebut menuntut kita untuk memperdalam ilmu robotika dan meningkatkan pendidikan ilmu ini sejak dini.

Sayang sekali, kit-kit robot untuk edukasi saat ini memiliki harga yang cukup tinggi sehingga hanya terjangkau oleh kalangan tertentu atau hanya dimiliki oleh lembaga pendidikan. Melalui buku ini, penulis berusaha menyajikan dasar-dasar robotika yang menggunakan bahan-bahan murah serta petunjuk pembuatan robot tersebut. Buku ini akan membahas robot mulai dari sisi elektronika yang diawali dengan penjelasan komponen-komponen yang digunakan dan diagram skematiknya hingga ke sisi mekanik yang dibangun dengan bahan-bahan yang mudah diperoleh, serta pemrograman, baik pemrograman sintaksis maupun pemrograman visual yang lebih mudah dipahami.

Dengan adanya buku ini, penulis berharap para pembaca mulai dari pelajar SD, SMP, SMU, SMK hingga perguruan tinggi dan para penghobi tidak hanya dapat menggunakan robot yang ada di lembaga pendidikan, namun bisa membuatnya sendiri dan memiliki robot-robot tersebut di rumah. Dengan kemampuan membangun robot-robot sederhana, penulis berharap pembaca dapat memiliki dasar untuk menerapkan ilmu robotika di lapangan kerja, tidak hanya sebagai operator bagi robot-robot impor, melainkan menjadi pembuat robot-robot tersebut.

Sebagai catatan, jika berminat untuk mendalami dunia robotika secara lebih mendetail melalui *workshop*, pembaca dapat menghubungi

DELTA ELECTRONIC
Kompleks Ruko Manyar Megah Indah Plaza D-22
Ngagel Jaya Selatan
Surabaya, Jawa Timur
Telepon: +62-31-5020210

Email : paulus@delta-electronic.com
Situs Web : www.delta-electronic.com
www.robotindonesia.com

Akhir kata, penulis memohon kritik dan saran yang bermanfaat demi meningkatkan dunia pendidikan robotika di Indonesia.

DAFTAR ISI

BAB I
DASAR ROBOTIKA

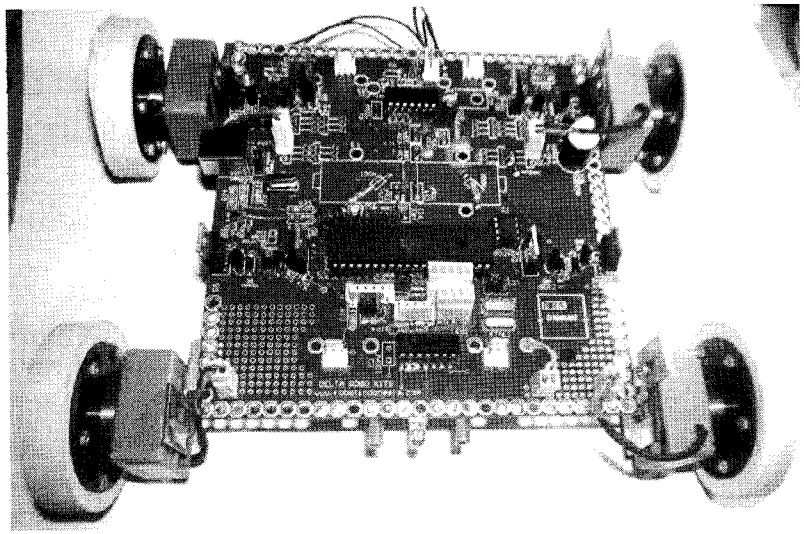
1.1 Pendahuluan

Apabila kita mendengar kata *robot*, perhatian kita pasti tertuju pada sebuah mesin berbentuk mirip manusia lengkap dengan lengan dan kaki seperti yang biasa digambarkan dalam film-film fiksi. Namun, dalam aplikasi sebenarnya robot tidaklah terpaku pada bentuk tersebut.

Robot yang diaplikasikan pada kehidupan sebenarnya didesain dengan bentuk yang sesuai dengan kebutuhannya. Contohnya, robot untuk memindahkan barang pada ban berjalan (*conveyor*) didesain sebagai lengan dan sensor-sensornya, dan tidak dilengkapi dengan roda atau kaki karena memang tidak membutuhkan mobilitas, namun robot pembawa barang yang bergerak di medan yang rata akan dilengkapi dengan roda, dan robot yang bergerak pada medan yang tidak beraturan akan dilengkapi dengan kaki. Singkatnya, agar memperoleh robot dengan desain yang ekonomis, robot-robot tersebut hanya didesain sesuai dengan kebutuhan. Dalam hal ini, robot berbentuk manusia (android) akan lebih sering digunakan untuk keperluan demo atau sebagai penerima tamu (pariwisata).

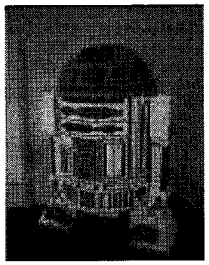
Untuk merancang robot yang tepat guna, terlebih dahulu kita harus mengenali jenis-jenis dan pengelompokan robot yang ada saat ini. Berdasarkan bentuknya, robot dibedakan menjadi beberapa kategori sebagai berikut:

1. *turtle*
   Diciptakan tahun 1970-an. Nama *turtle* diambil dari bentuknya yang mirip rumah kura-kura.
2. *vehicle*
   Robot jenis ini berbentuk seperti kendaraan beroda dan bergerak seperti mobil. Perbedaannya dengan mobil adalah kemampuan terprogram (*programmable*) dan sensor-sensornya.



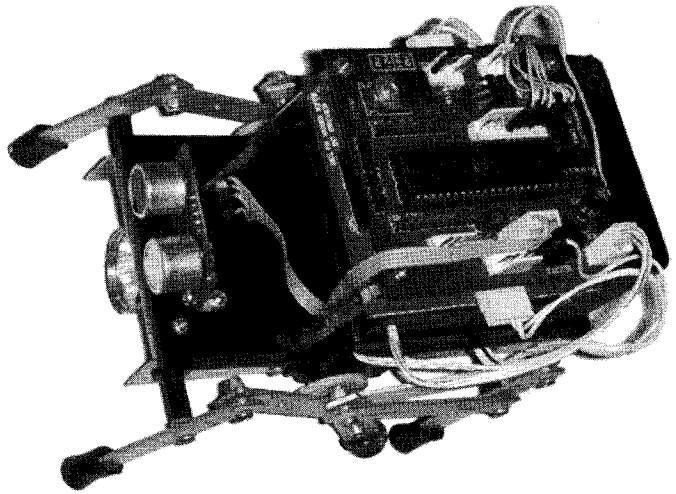
Gambar 1.1 Robot *vehicle*

3. *Rover*
   Bentuk robot ini cenderung pendek dan juga dilengkapi roda seperti jenis *vehicle*, contohnya R2-D2 dalam film *Star Wars*. Robot jenis ini juga dilengkapi dengan beberapa fungsi seperti kemampuan untuk mendeteksi api atau mendeteksi objek.



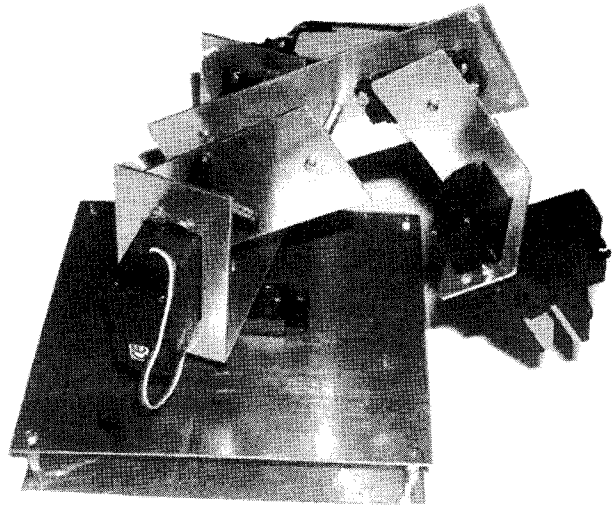
Gambar 1.2 Robot *rover*

4. *Walker*
   Robot jenis ini tidak dilengkapi dengan roda seperti jenis *vehicle* dan *rover*, melainkan bergerak dengan menggunakan kaki. Biasanya robot ini berbentuk mirip serangga dan dilengkapi dengan enam kaki.



Gambar 1.3 Robot *walker*

5. *Appendage*
   Robot ini berupa lengan yang biasanya digunakan untuk mengambil dan memindahkan barang. Lengan ini dapat terpasang pada robot yang bergerak atau pada sebuah tempat yang statis.



Gambar 1.4 Robot *appendage*

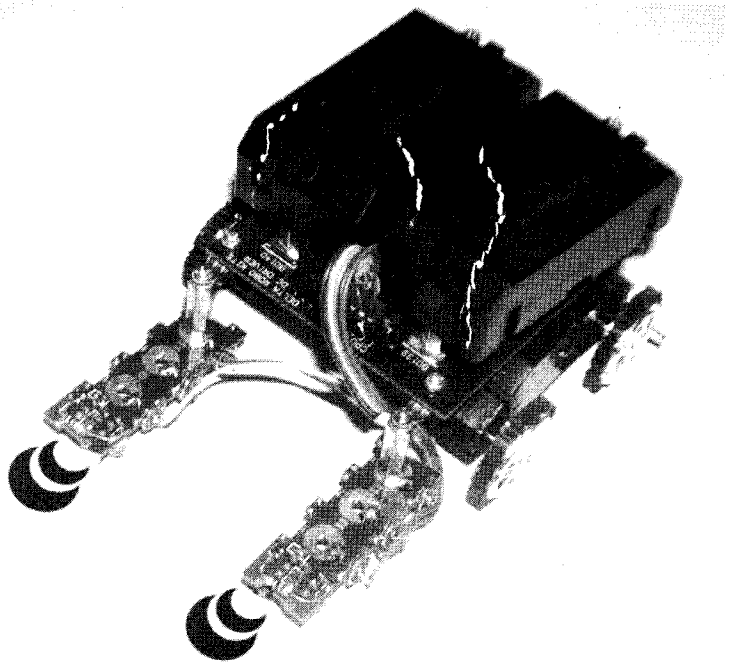
6. Android
   Robot ini didesain menyerupai manusia dan mempunyai kemampuan untuk berkomunikasi dengan manusia.

Berdasarkan proses kendalinya, robot dibedakan menjadi robot otomatis dan robot *teleoperated*.

1. Otomatis

Robot otomatis bergerak berdasarkan perintah-perintah yang telah diprogramkan sebelumnya atau berdasarkan masukan dari sensor-sensornya.

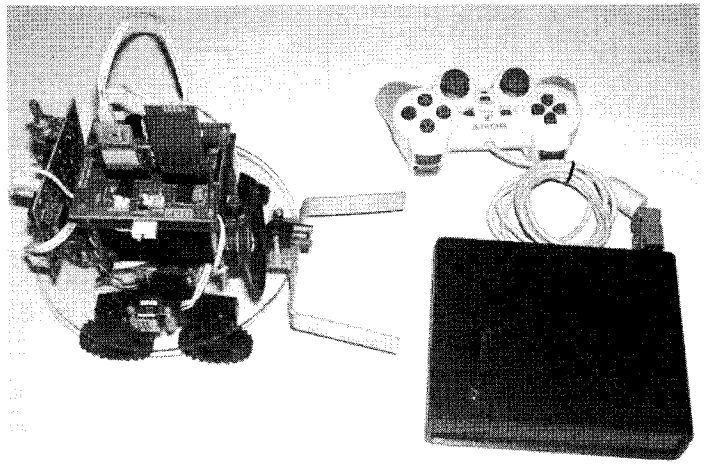
Contohnya seperti yang tampak pada Gambar 1.5, sebuah robot penjejak garis yang dilengkapi dengan "sungut" untuk mendeteksi posisi garis. Robot ini akan bergerak secara otomatis berdasarkan perubahan kondisi sensor sehingga robot akan mengikuti garis.



Gambar 1.5 Robot otomatis

2. *Teleoperated*

Robot jenis ini bergerak berdasarkan perintah-perintah yang dikirimkan oleh operator secara manual. Perintah-perintah tersebut dapat dikirimkan melalui *keypad*, *joystick*, atau *remote control*.



Gambar 1.6 Robot yang dikendalikan dengan *joystick*

1.2 Bagian-Bagian Robot

Secara garis besar robot terdiri atas bagian mekanik dan elektronika, di mana bagian mekanik merupakan bagian-bagian penggerak robot seperti motor dan kerangka robot, sedangkan bagian elektronika terdiri atas rangkaian pengendali motor, sensor, atau rangkaian untuk menerima input perintah dari operator seperti *keypad*, *remote*, dan lain-lain. Selain kedua bagian tersebut, ada juga bagian perangkat lunak untuk robot-robot yang memiliki kecerdasan lebih atau kemampuan pemrograman.

1.2.1 Bagian Mekanik

Bagian ini merupakan penggerak atau kerangka robot. Kerangka robot akan didesain sesuai dengan fungsi robot. Kerangka berbentuk lengan adalah untuk robot yang bersifat statis, dan kerangka robot berkaki untuk robot yang bergerak di medan yang tidak rata. Sedangkan bagian penggerak yang bagaikan otot robot biasa menggunakan motor. Beberapa robot bahkan

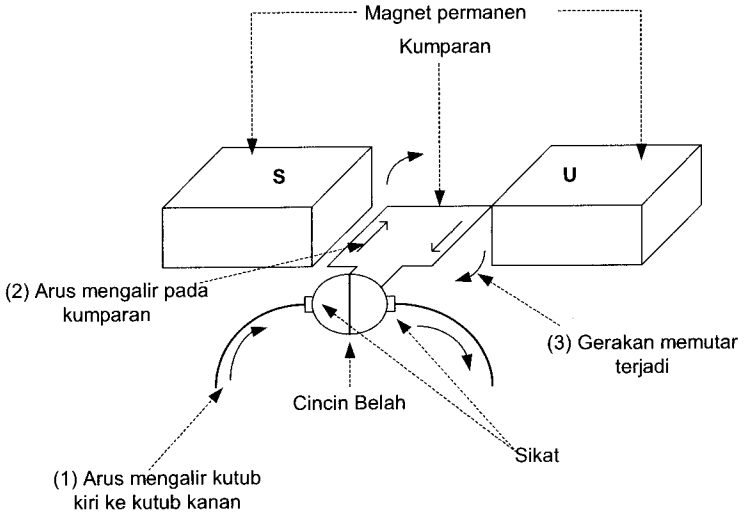
menggunakan kawat otot (*muscle wire*), yaitu kawat yang merapat dan merenggang saat dialiri arus. Kawat ini lebih menyerupai gerakan otot dibandingkan motor, namun harganya yang cukup mahal membuat kawat ini jarang digunakan pada robot.

1.2.1.1 Motor DC

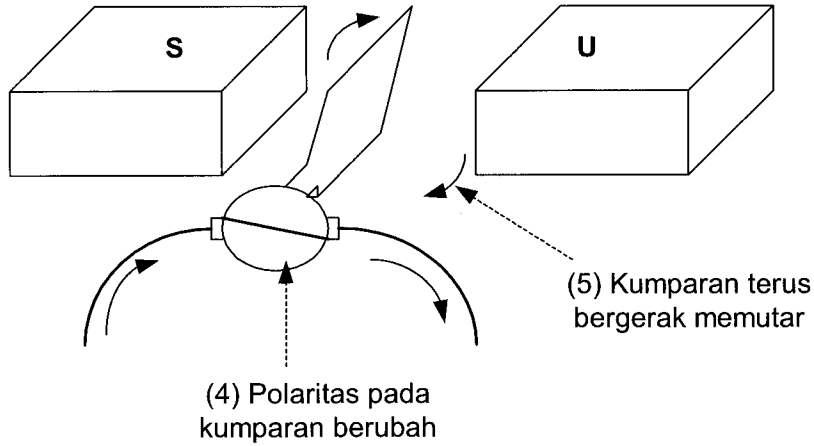
Pada sebuah robot, motor DC merupakan bagian penggerak utama. Hampir semua robot pasti menggunakan motor DC, kecuali beberapa robot yang menggunakan pneumatik, kawat otot, atau motor AC.

Motor DC terdiri atas sebuah magnet permanen dengan dua kutub dan kumparan, cincin belah yang berfungsi sebagai komutator (pemutus arus).

1. Arus mengalir dari sisi kiri cincin belah ke sisi kanan. Arus ini akan dilanjutkan ke kumparan yang terkait pada cincin belah.
2. Arus yang mengalir dalam kumparan menimbulkan medan magnet dan membentuk kutub-kutub magnet pada kumparan.
3. Kutub magnet yang sama dengan kutub magnet permanen akan saling menolak dan kumparan akan bergerak memutar hingga kumparan berada pada posisi di mana kedua kutubnya berbeda dengan kutub magnet permanen.

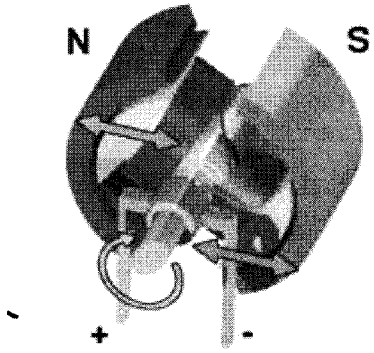


Gambar 1.7 Fase 1 dari motor DC



Gambar 1.8 Fase 2 dari motor DC

4. Perputaran kumparan yang terkait pada cincin belah akan mengakibatkan perubahan polaritas pada kumparan karena sikat-sikat (*brush*) yang dialiri listrik terhubung pada sisi cincin belah yang berbeda.
5. Perubahan polaritas kumparan juga mengakibatkan perubahan kutub pada kumparan sehingga kumparan kembali bergerak memutar.
6. Proses tersebut terjadi berulang-ulang sehingga kumparan akan berputar secara kontinu selama aliran arus terjadi pada kedua kutub sikat.

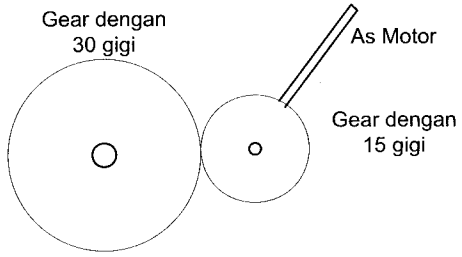


Gambar 1.9 Putaran motor DC

Arah putaran motor DC dapat diubah dengan mengganti polaritas aliran arus yang terhubung dengan sikat-sikatnya, sedangkan kecepatan putar motor tergantung pada seberapa besar arus yang mengalir.

1.2.1.2 Motor *Gearbox*

Sering kali gerakan-gerakan pada robot membutuhkan torsi yang cukup besar dan hal ini tidak dapat dilakukan langsung oleh motor DC saja. Untuk memperbesar torsi dibutuhkan rangkaian gir yang akan mereduksi kecepatan motor dan sekaligus meningkatkan torsi. Proses pengurangan kecepatan dan peningkatan torsi ini berbanding terbalik dan dihitung berdasarkan perbandingan gigi.



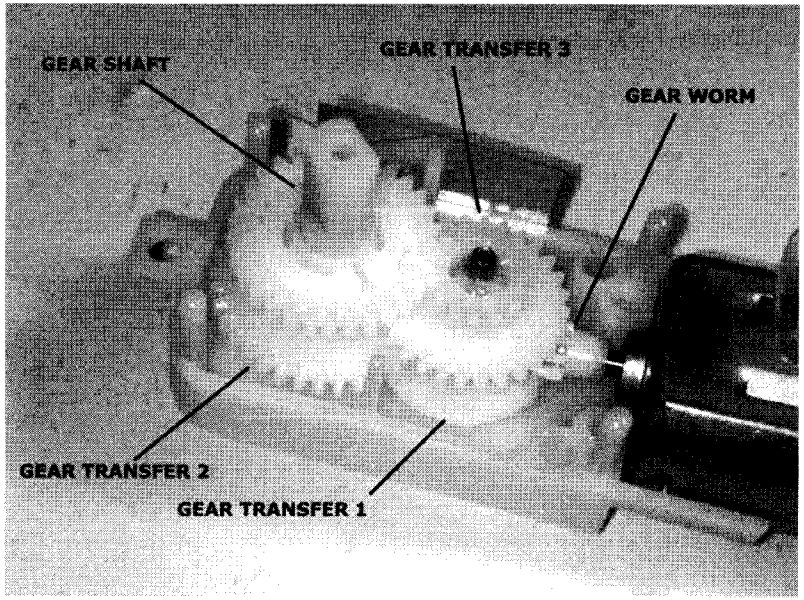
Gambar 1.10 Perhitungan rasio gigi

Perhatikan *Arduino DC motor 004-0065* (Gambar 1.11) yang terdiri atas 1 *worm gear*, 3 buah *transfer gear*, dan 1 *gear shaft*.

1. *Worm gear* adalah gigi yang berbentuk ulir yang berfungsi mengubah arah putaran dari horizontal menjadi vertikal.
2. *Transfer gear* adalah gigi yang berfungsi untuk konversi antara gigi dengan jumlah banyak ke jumlah kecil ataupun sebaliknya.
3. *Gear shaft* adalah gigi yang terhubung langsung dengan as atau sumbu motor.

Perbandingan *gear transfer* dapat dihitung dengan menghitung jumlah gigi yang ada baik sesudah maupun sebelum transfer. Sembilan gigi dari *gear transfer 1* terhubung dengan bagian *gear transfer 2* yang memiliki 35 gigi sehingga diperoleh perbandingan 35:9 dan kecepatan akan menurun sebesar

35/9 serta torsi meningkat sebesar 35/9. Kemudian, bagian 15 gigi dari *gear transfer 2* terhubung dengan bagian *gear transfer 3* yang memiliki 27 gigi sehingga diperoleh perbandingan 27:15 yang akan menurunkan kecepatan sebesar 27/15 dan meningkatkan torsi sebesar 27/15 juga.



Gambar 1.11 *Arduino DC motor 004-0065*

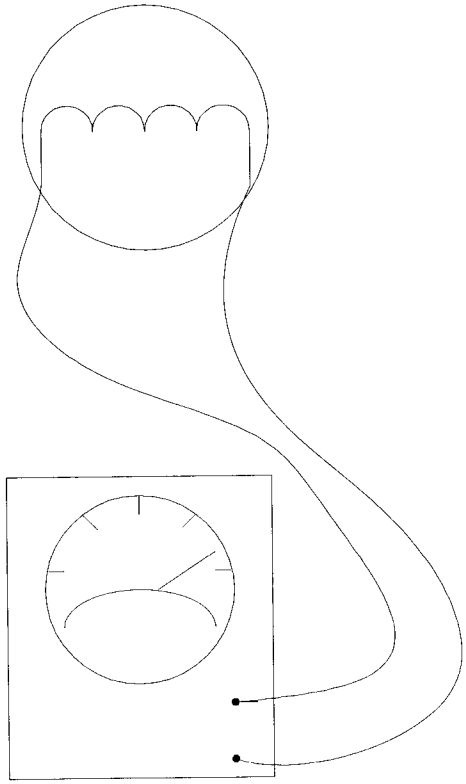
Bagian 14 gigi dari *gear transfer 3* terhubung dengan *gear shaft* dengan 30 gigi sehingga diperoleh perbandingan 30:14. Cara untuk menghitung rasio total dari gir adalah sebagai berikut. Perhitungan rasio dilakukan pada gigi-gigi yang saling bersinggungan.

$$35/9 \times 27/15 \times 30/14 = 15$$

Jadi, rasio dari motor *gearbox* ini adalah 1:15 di mana torsi akan naik 15 kali lipat dan kecepatan turun 15 kali lipat pula.

Motor DC selalu memiliki dua kaki (sering kali terhubung juga dengan kabel) dan terhubung dengan kumparan sehingga cara untuk menguji apakah motor ini masih dapat bekerja dengan baik adalah sebagai berikut.

1. Gunakan multimeter dan atur pada posisi ohm.
2. Hubungkan kedua kabel multimeter dengan kedua kaki/kabel motor.
3. Pastikan jarum pada multimeter menyimpang ke kanan untuk multimeter analog atau *display* multimeter menampilkan nilai hambatan untuk multimeter digital.

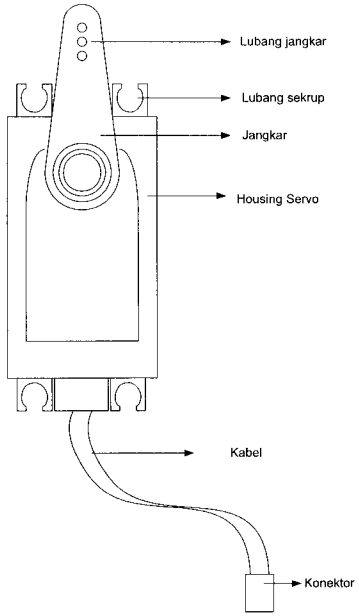


Gambar 1.12 Hubungan multimeter dengan motor DC

4. Beberapa jenis motor DC memiliki lebih dari dua kabel. Sering kali terlihat motor DC memiliki 5 kabel di mana ketiga tambahan kabel tersebut biasa digunakan untuk enkoder, yaitu bagian dari motor yang digunakan untuk mengetahui posisi motor (akan dijelaskan pada subbab selanjutnya). Untuk mengetahui kabel kumparan, pilihlah dua kabel terbesar.

1.2.1.3 Motor Servo

Berbeda dengan motor DC, motor servo tidak bergerak kontinu, melainkan menuju sudut tertentu saja dan berhenti di sudut tersebut. Motor ini digunakan untuk aplikasi gerakan-gerakan sudut dari robot, contohnya gerakan lengan, *gripper* menjepit benda, atau gerakan kaki melangkah. Gambar 1.13 menunjukkan motor servo yang sering digunakan pada aplikasi robotik.

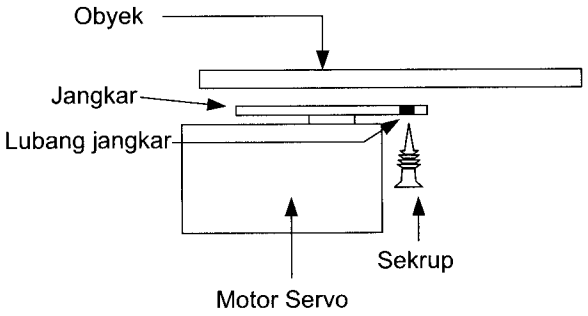


Gambar 1.13 Motor servo

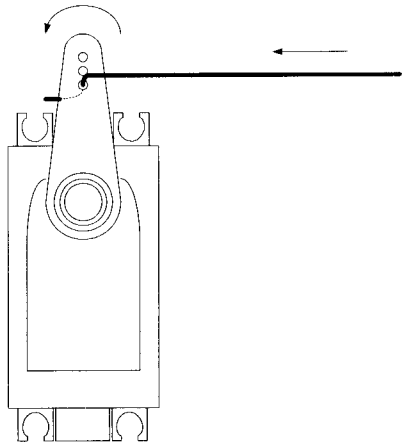
Motor servo ini terdiri atas beberapa bagian, yaitu

1. Jangkar, untuk menghubungkan motor servo dengan objek-objek yang akan digerakkan.
2. Lubang jangkar, bagian ini berfungsi untuk menempatkan sekrup yang mengaitkan jangkar ke objek-objek yang akan digerakkan. Pada Gambar 1.14 tampak lubang jangkar dihubungkan ke objek dengan sekrup untuk gerakan memutar.
3. Lubang sekrup, berfungsi untuk mengaitkan motor servo dengan tubuh robot.

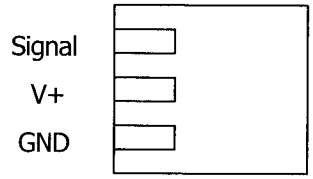
4. *Housing* servo, di dalam bagian ini terdapat motor DC, *gearbox*, dan rangkaian pengatur sudut servo.
5. Kabel, kabel yang menghubungkan rangkaian servo dengan pengendali servo.
6. Konektor, konektor 3 pin yang terdiri atas input tegangan positif (+), input tegangan negatif (GND), dan input pulsa (sinyal) dengan konfigurasi seperti pada Gambar 1.16.



Gambar 1.14 Pemasangan objek pada jangkar untuk gerakan memutar



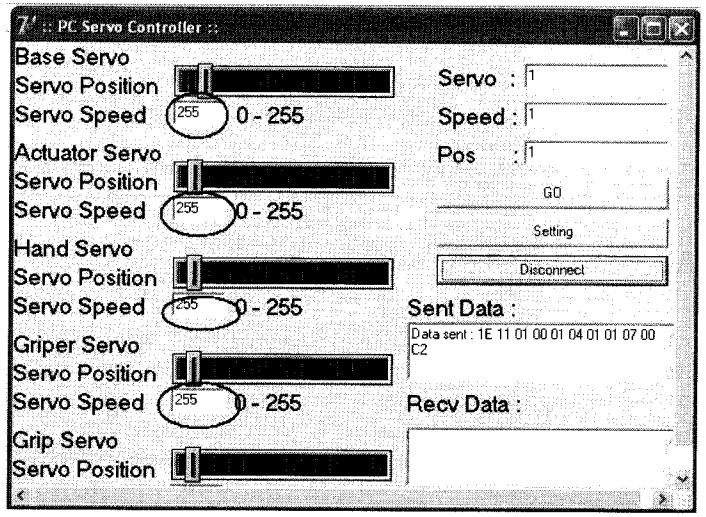
Gambar 1.15 Pemasangan objek untuk gerakan mengait



Gambar 1.16 Konfigurasi konektor servo

Tidak seperti motor DC, pengendalian motor servo harus dilakukan dengan terlebih dahulu melalui rangkaian pengatur sudut di mana rangkaian ini membutuhkan masukan berupa pulsa.

Pulsa-pulsa ini biasanya dibangkitkan oleh komponen cerdas seperti mikrokontroler atau modul elektronik *delta servo controller*. Modul *delta servo controller* berfungsi membangkitkan pulsa-pulsa untuk mengendalikan motor servo dari PC. Dengan bantuan perangkat lunak PC Servo Controller, enam motor servo dapat dikendalikan dengan menggeser-geser panel yang ada pada perangkat lunak tersebut.



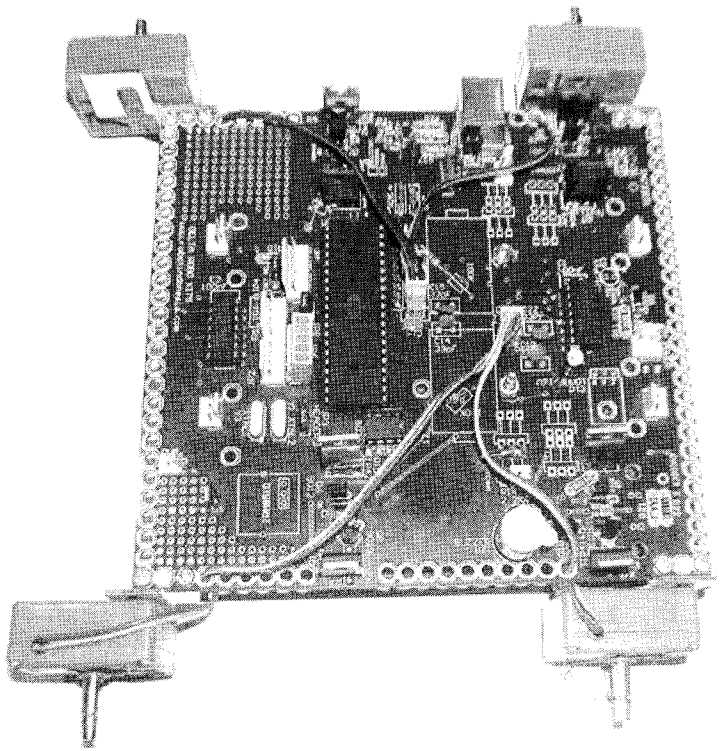
Gambar 1.17 Perangkat lunak PC Servo Controller

1.2.1.4 Kerangka Robot

Seperti yang sudah dibahas sebelumnya, desain kerangka robot sangat tergantung pada jenis robot. Pada kesempatan kali ini akan dibahas kerangka robot untuk jenis *vehicle*, *walker*, dan *appendage*.

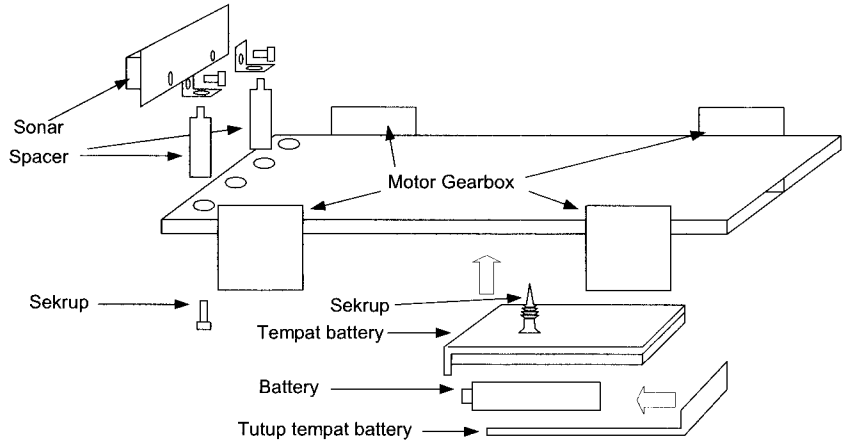
1. **Kerangka robot *vehicle***
   Seperti yang telah dijelaskan sebelumnya bahwa robot *vehicle* berbentuk mirip mobil dengan roda-rodanya. Konstruksi ini merupakan konstruksi robot dengan bagian mekanik yang paling sederhana dan perancangan yang mudah.

   Agar memperoleh desain yang ekonomis dan konstruksi yang tidak terlalu banyak beban, desain kerangka robot *vehicle* sering kali juga menjadi bagian elektronika robot.



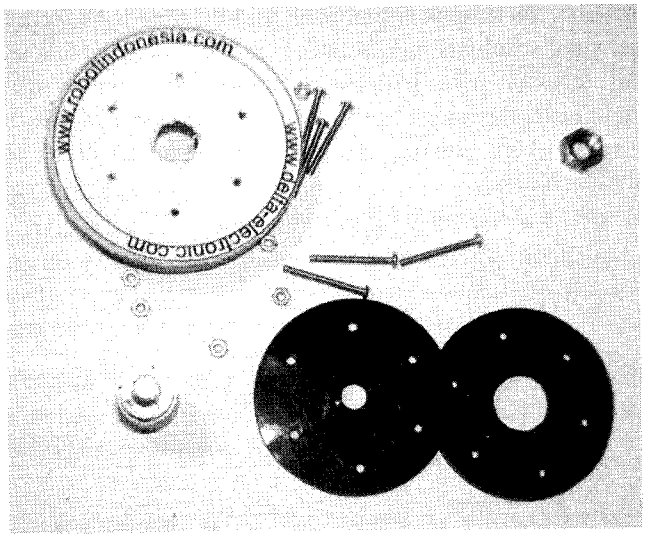
Gambar 1.18 Kerangka robot *vehicle delta robo kits*

Pada Gambar 1.18 tampak empat buah motor *gearbox* DC terpasang pada rangkaian yang sekaligus membentuk kerangka robot *vehicle*. Pada papan rangkaian tersebut juga terdapat lubang-lubang untuk menempatkan *spacer* atau besi penyangga untuk mengaitkan aksesoris-aksesoris seperti sensor dan lain-lain. Lubang tersebut juga digunakan untuk mengaitkan sekrup tempat baterai.



Gambar 1.19 Kerangka robot *vehicle delta robo kits* dengan sensor sonar dan tempat baterainya

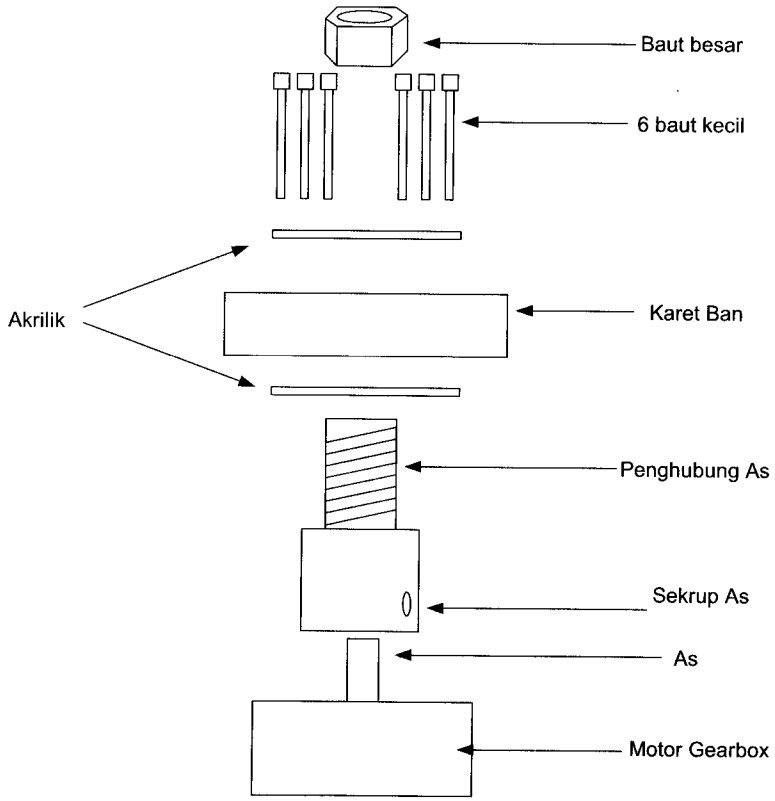
Satu hal penting pada bagian mekanik robot *vehicle* adalah roda, bagian yang mengubah efek putaran motor menjadi gaya gerak robot, baik maju, mundur, maupun berbelok.



Gambar 1.20 *Delta robo wheel V2*

Agar diperoleh efek kelembaman yang baik antara roda dan permukaan lantai, kita membutuhkan ban dengan lapisan karet seperti pada Gambar 1.20. Ban karet tersebut dijepit oleh dua akrilik bundar dan enam buah mur baut kecil dan sebuah mur baut besar. Mur baut besar berfungsi untuk mengunci besi penghubung as dengan ban dan akrilik, sedangkan mur baut kecil berfungsi untuk mengunci akrilik terhadap ban. Akrilik bundar berfungsi untuk menjaga agar besi penghubung as selalu berada pada posisi di tengah ban karet.

Pada besi penghubung as terdapat ulir yang dapat dihubungkan pada baut besar dan menjepit kedua akrilik pada ban. Besi penghubung as ini memiliki lubang sebesar 5 mm dan lubang sekrup (sekrup as) yang dapat menjepit besi ini terhadap as. Dengan adanya sekrup as, roda ini akan cukup fleksibel sehingga dapat dihubungkan ke berbagai macam ukuran as (asalkan di bawah 5 mm).

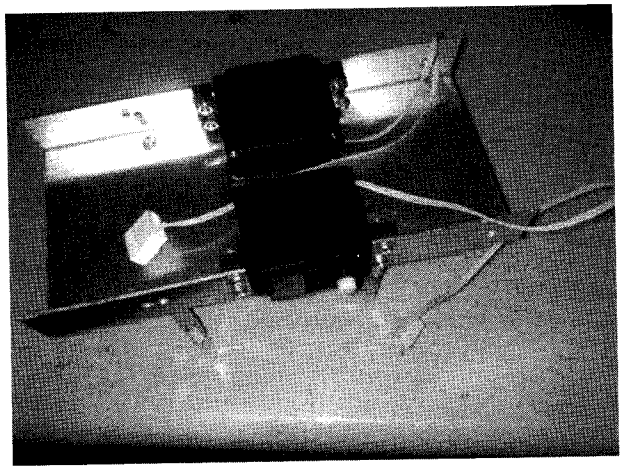


Gambar 1.21 Bagian-bagian *delta robo wheel V2*

2. Kerangka robot *walker*

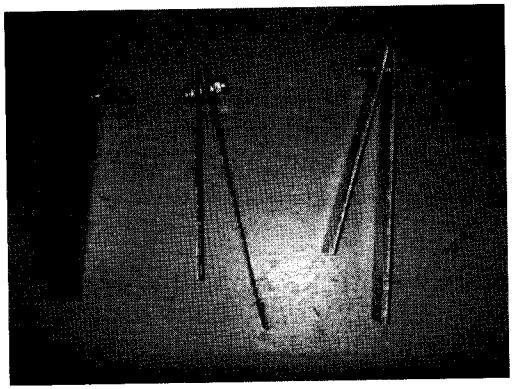
Robot *walker* adalah robot yang bergerak dengan kaki. Konstruksi yang paling mudah adalah robot berkaki enam yang didesain dengan menggunakan motor DC di mana gerakan memutar akan dikonversi menjadi gerakan melangkah pada robot. Berikut adalah langkah-langkah untuk merancang robot *walker* dengan enam kaki.

a. Pasang motor servo HS-311/HS-322/HS-422 yang sudah diubah menjadi mode kontinu.

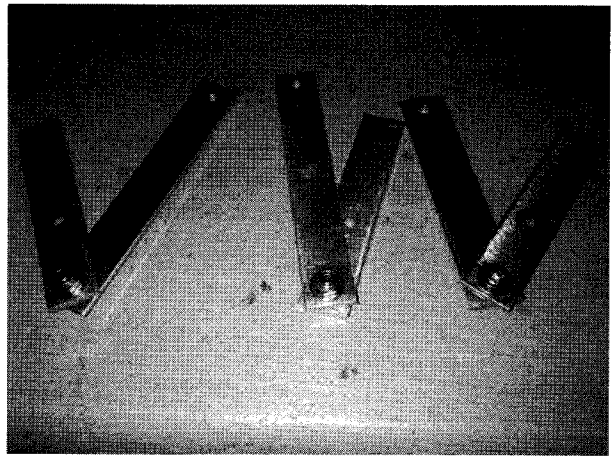


Gambar 1.22 Memasang motor servo

b. Pasang ruas-ruas kaki heksapoda dengan susunan berikut, di mana selalu terdapat ring di antara mur-plat1-plat2-baut.

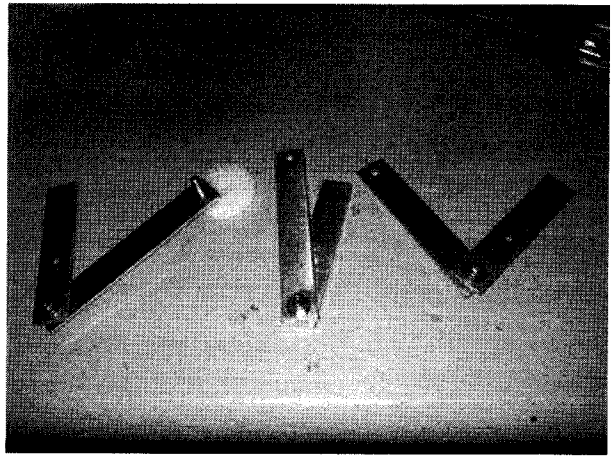


Gambar 1.23 Ruas-ruas kaki robot



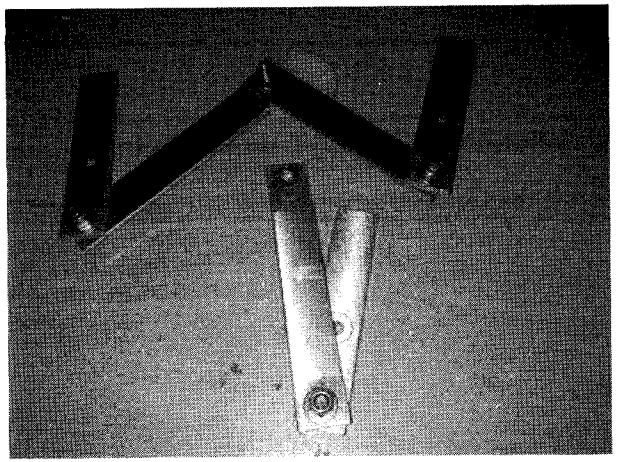
Gambar 1.24 Ruas-ruas kaki robot

c. Pasang pengait motor servo.



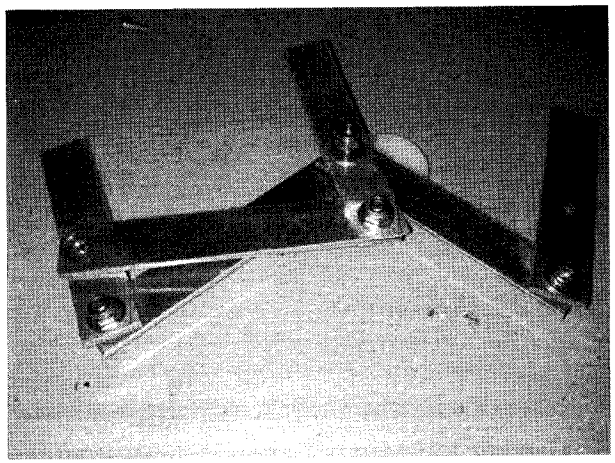
Gambar 1.25 Memasang pengait motor servo

d. Hubungkan kaki beruas panjang dengan kaki beruas pendek.



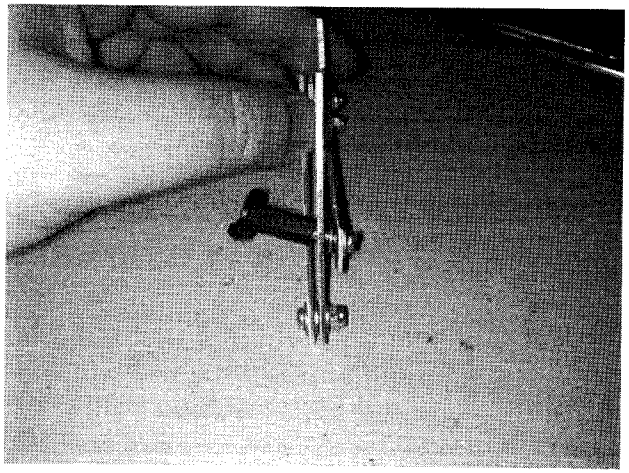
Gambar 1.26 Kaki beruas panjang dan kaki beruas pendek terpasang

e. Pasang kaki bagian tengah.



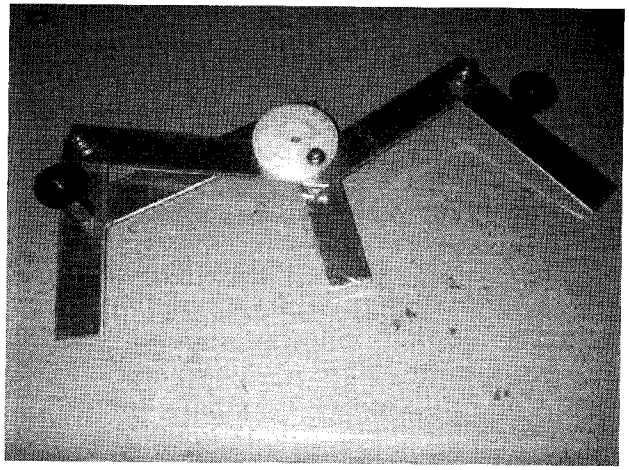
Gambar 1.27 Memasang kaki bagian tengah

f. Pasang *spacer* plastik pada kaki.



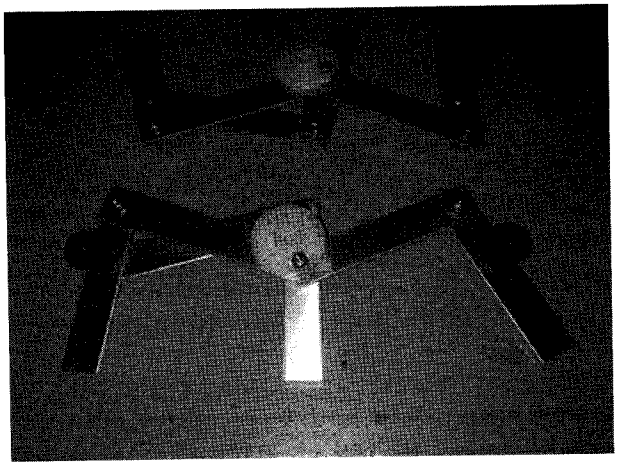
Gambar 1.28 Memasang *spacer* plastik

g. Pasang *spacer* plastik yang kedua.



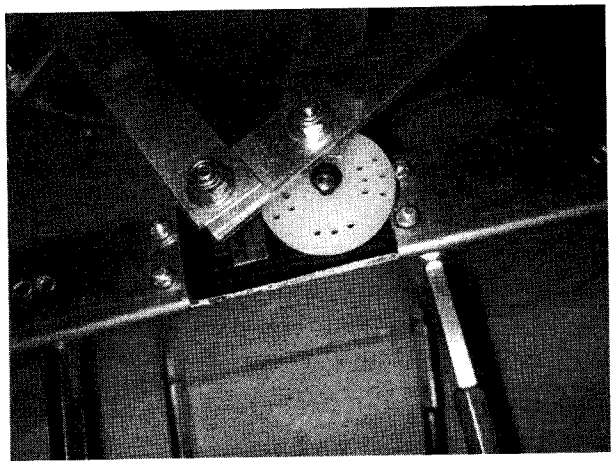
Gambar 1.29 *Spacer* plastik yang kedua

h. Lakukan hal yang sama pada pasangan kaki yang lain sehingga mendapatkan bentuk seperti gambar berikut.



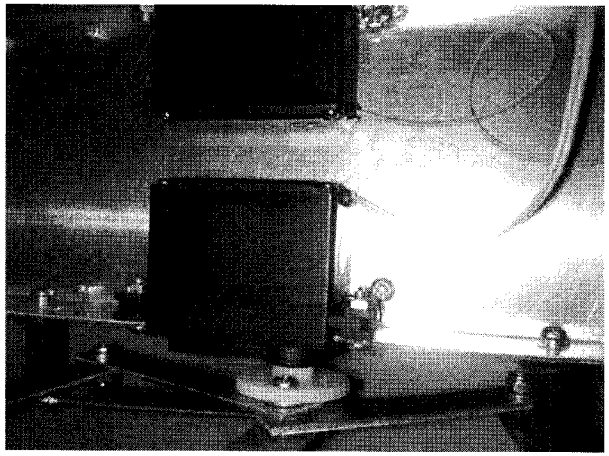
Gambar 1.30 Bentuk kaki, *spacer*, dan motor servo yang sudah terpasang

i. Pasang pengait servo pada pasangan kaki bagian bawah dari gambar dengan sekrup ke motor servo.

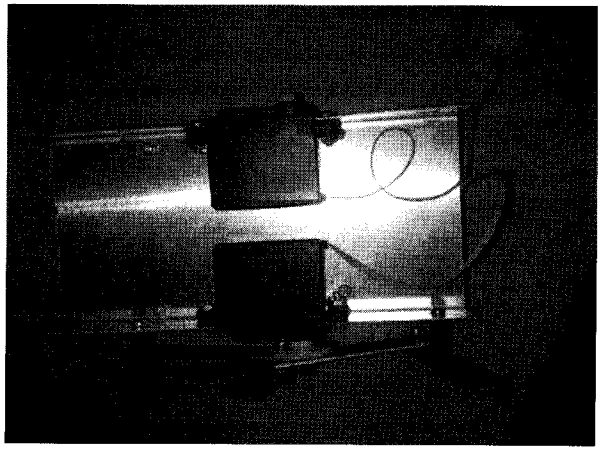


Gambar 1.31 Memasang pengait servo

j. Pasang *spacer* plastiknya pada badan heksapoda.

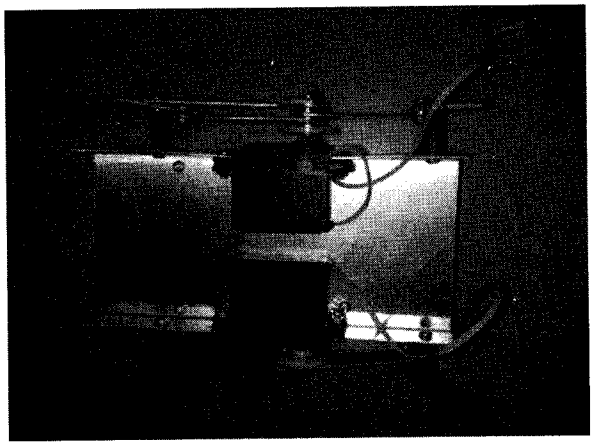


Gambar 1.32 Memasang *spacer* plastik pada badan heksapoda



Gambar 1.33 Memasang *spacer* plastik pada badan heksapoda

k. Pasang pasangan kaki yang ada pada bagian atas gambar sehingga menjadi bentuk seperti berikut.

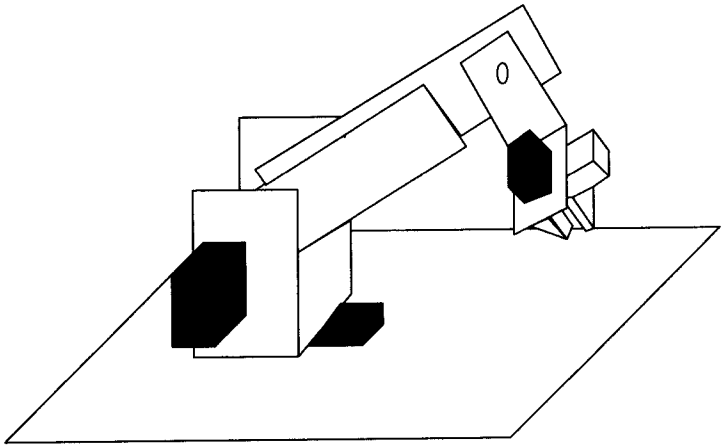


Gambar 1.34 Memasang pasangan kaki

3. Kerangka robot *appendage*

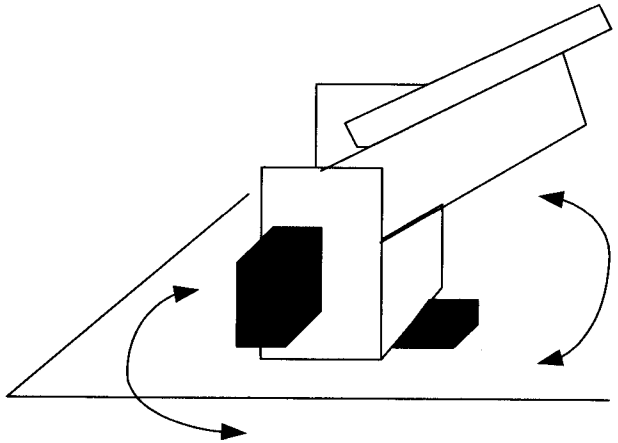
Robot *appendage* didesain menyerupai bentuk lengan dan berfungsi untuk memindahkan barang. Aplikasi proses pemindahan barang ini biasanya dilakukan untuk proses sortir pada sistem ban berjalan (*conveyor*).

Pada bagian ini akan dibahas lengan robot dengan 5 sumbu gerakan, yaitu putaran basis (*base*), gerakan aktuator naik-turun, gerakan tangan naik-turun, gerakan *gripper* memutar, dan gerakan *gripper* menjepit.

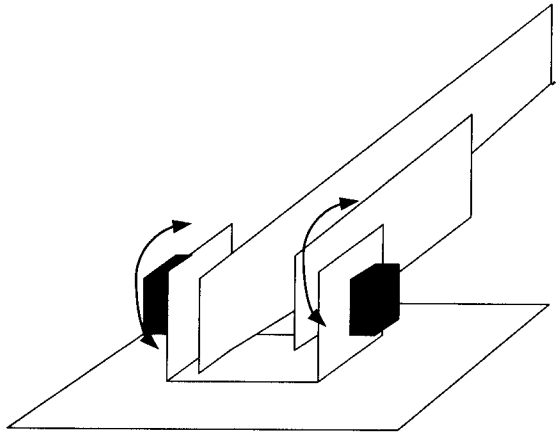


Gambar 1.35 Mekanik lengan robot dengan 5 sumbu gerakan

Motor-motor yang digunakan pada aplikasi ini adalah motor servo di mana bagian aktuator dan bagian basis (*base*) menggunakan motor servo yang cukup besar torsinya, yaitu GWS S04BB sebesar 13 kg, sedangkan ketiga motor servo yang lain sebesar 3 kg.



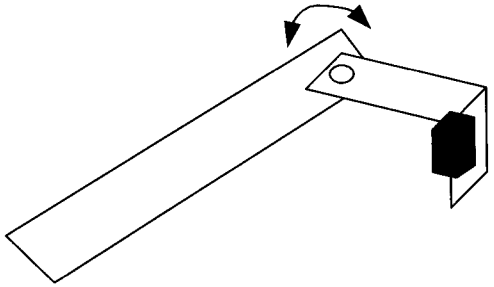
Gambar 1.36 Gerakan basis memutar



Gambar 1.37 Gerakan aktuator naik turun

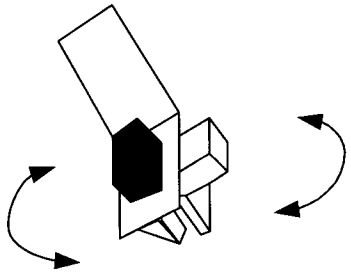
Dengan posisi yang berhadapan gerakan kedua motor servo ini harus dilakukan dengan arah yang berlawanan dengan resultan sudut yang sama.

Bila motor servo di sisi kiri bergerak ke arah CW, motor servo di sisi kanan harus bergerak ke arah CCW.



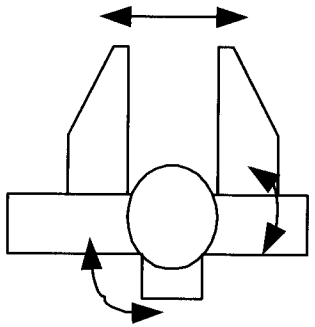
Gambar 1.38 Gerakan tangan naik turun

Gerakan tangan tidak membutuhkan torsi sebesar aktuator sehingga kita dapat menggunakan 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 mendapatkannya.



Gambar 1.39 Gerakan *gripper* memutar

Untuk menyesuaikan posisi dengan objek yang dituju terdapat sebuah motor servo HS-422 yang berfungsi untuk memutar posisi *gripper* hingga posisinya sesuai dengan kondisi objek. Setelah objek diperoleh, lengan robot ini 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 1.40 Gerakan *gripper* merapat dan merenggang

1.2.2 Bagian Elektronika

Seperti yang telah dibahas sebelumnya bahwa motor DC, motor *gearbox*, maupun motor servo membutuhkan listrik untuk mengatur gerakannya. Oleh karena itu, bagian elektronika merupakan salah satu bagian utama dari sebuah robot. Fungsi-fungsi rangkaian elektronika adalah sebagai berikut.

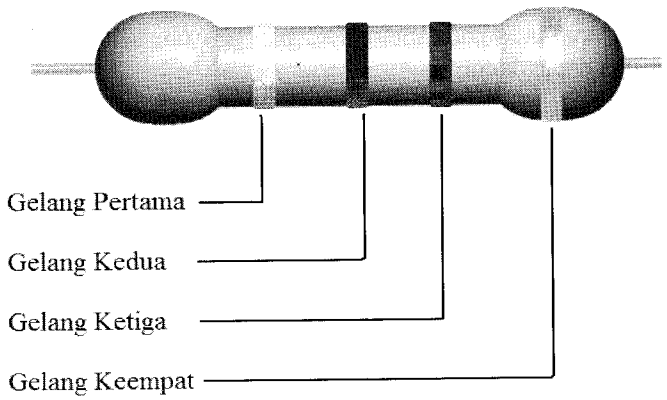
1. Mengatur arah arus pada motor DC atau *gearbox* sehingga arah putaran motor juga dapat diatur.
2. Mengatur besar arus yang mengalir pada motor DC atau *gearbox* sehingga kecepatan gerak motor dapat diatur.
3. Mengatur besar pulsa yang masuk ke motor servo sehingga posisi sudut servo dapat ditentukan.
4. Menerima input dari sensor sehingga robot dapat mengetahui kondisi sensor dan memberikan respons (hal ini digunakan pada robot-robot otomatis).
5. Menerima input dari operator, biasanya berupa *keypad*, *remote control*, atau *joystick*.
6. Mengeluarkan output seperti *display* LCD atau suara sebagai output informasi dari robot.
7. Menyimpan perintah-perintah ke dalam memori (bagian ingatan dari robot).

Untuk merancang rangkaian elektronika, terlebih dahulu kita harus mengenali komponen-komponen elektronika yang digunakan.

1. Resistor

Resistor merupakan komponen elektronika pasif yang mempunyai fungsi dasar untuk menahan arus listrik atau membagi tegangan. Ada berbagai jenis resistor, namun kita hanya akan membahas jenis resistor yang digunakan pada robot-robot dalam buku ini, yaitu resistor karbon dan resistor pasang permukaan (*surface mount device*/SMD).

a. Resistor karbon
Resistor karbon terdiri atas sebuah unsur resistif berbentuk tabung dengan kawat atau tutup logam pada kedua ujungnya. Badan resistor dilindungi dengan cat atau plastik. Resistor komposisi karbon lawas mempunyai badan yang tidak terisolasi, kawat penghubung dililitkan di sekitar ujung unsur resistif dan kemudian disolder. Resistor yang sudah jadi dicat dengan kode warna harganya.



Gambar 1.41 Gelang-gelang resistor karbon

TABEL 1.1 Nilai warna gelang resistor karbon

Warna	Gelang pertama	Gelang Kedua	Gelang Ketiga (pengali)	Gelang keempat (toleransi)	Gelang Kelima (koefisien suhu)
Hitam	0	0	$\times 10^0$		
Cokelat	1	1	$\times 10^1$	± 1% (F)	100 ppm
Merah	2	2	$\times 10^2$	± 2% (G)	50 ppm
Oranye	3	3	$\times 10^3$		15 ppm
Kuning	4	4	$\times 10^4$		25 ppm
Hijau	5	5	$\times 10^5$	± 0.5% (D)	
Biru	6	6	$\times 10^6$	± 0.25% (C)	
Ungu	7	7	$\times 10^7$	± 0.1% (B)	
Abu-abu	8	8	$\times 10^8$	± 0.05% (A)	
Putih	9	9	$\times 10^9$		
Emas			$\times 10^{-1}$	± 5% (J)	
Perak			$\times 10^{-2}$	± 10% (K)	
Kosong				± 20% (M)	

Parameter-parameter yang perlu diperhatikan pada resistor adalah sebagai berikut:

1) Nilai hambatan (ohm)
2) Daya (watt), untuk aplikasi robot yang biasanya menggunakan arus yang sangat lemah, parameter ini tidak terlalu diperhatikan. Namun, apabila arus yang dilewatkan pada resistor ini cukup besar (mendekati 1 ampere), nilai daya harus diperhatikan.
3) Toleransi (%), hal ini perlu diperhatikan untuk aplikasi-aplikasi yang membutuhkan ketelitian tinggi seperti pada pengukuran yang melibatkan nilai resistansi sebagai variabel dari hasil pengukuran tersebut.
4) Koefisien suhu (gelang kelima), parameter ini hanya digunakan untuk aplikasi di tempat-tempat yang memiliki suhu yang sangat ekstrem. Gelang kelima ini jarang terlihat pada resistor.

Untuk mengukur nilai hambatan pada resistor karbon dapat dilakukan dengan membaca gelang-gelang warna seperti pada TABEL 1.1. Dua gelang pertama merupakan informasi dua digit harga resistansi, gelang ketiga merupakan pengali (jumlah nol yang ditambahkan setelah dua digit resistansi), dan gelang keempat merupakan toleransi harga resistansi. Kadang-kadang gelang kelima menunjukkan koefisien suhu, tetapi ini harus dibedakan dengan sistem lima warna sejati yang menggunakan tiga digit resistansi.

Sebagai contoh, hijau-biru-kuning-merah adalah 56 x 10^4 Ω = 560 kΩ ± 2%. Deskripsi yang lebih mudah adalah gelang pertama (hijau) mempunyai harga 5 dan gelang kedua (biru) mempunyai harga 6, dan keduanya dihitung sebagai 56. Gelang ketiga (kuning) mempunyai harga 10^4, yang menambahkan empat nol di belakang 56, sedangkan gelang keempat (merah) merupakan kode untuk toleransi ± 2%, memberikan nilai 560.000 Ω pada keakuratan ± 2%.

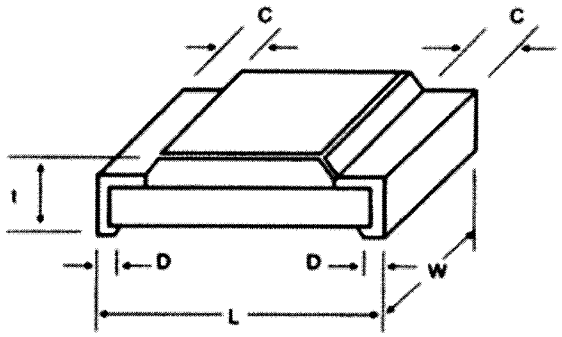
Metode paling mudah untuk mengukur hambatan adalah dengan menggunakan multimeter. Untuk multimeter digital dapat dengan mudah kita lakukan dengan memindah sakelar ke posisi hambatan dan melihat nilai yang tertampil pada layar *display* pada saat kedua ujung kabel multimeter kita hubungkan dengan resistor.

b. Resistor pasang permukaan (*surface mount device*/SMD)
Resistor pasang permukaan (lebih dikenal dengan istilah SMD) dewasa ini dikembangkan untuk membuat desain yang lebih ringkas dan kompak. Dimensinya yang sangat kecil dan hanya memerlukan satu sisi PCB untuk penyolderan membuat desainnya sangat kecil dan ekonomis karena biaya cetak PCB menjadi lebih murah. Di samping itu, komponen SMD tidak memerlukan pengeboran pada PCB sehingga proses pembuatan PCB akan jauh lebih mudah.

Untuk resistor SMD terdapat parameter yang juga perlu diperhatikan, yaitu dimensi, seperti yang tampak pada Gambar 1.42. Oleh karena itu, untuk memilih nilai resistor SMD, biasanya digunakan nilai hambatan dan dimensi. Satuan dimensi yang digunakan untuk ini adalah inci. Contoh, SMD 0805 berarti resistor tersebut memiliki dimensi 0,08 inci x 0,05 inci.

Untuk menentukan nilai ohm pada resistor SMD terdapat 3 digit angka di mana digit pertama dan kedua adalah nilai, dan digit terakhir

adalah jumlah pengali. Contoh, pada resistor yang tertulis 153, nilai yang diperoleh adalah 15 x 10^3 atau 15 kohm.



INCH	MM	DIMENSI				
		L	W	C	D	t
0402	1005	1.0±0.05	0.5±0.05	0.2±0.1	0.25+0.05 0.25-0.1	0.35±0.05
0603	1608	1.6±0.1	0.85±0.1	0.3±0.2	0.2+0.2 0.2-0.1	0.45±0.05
0805	2012	2.05±0.1	1.3±0.1	0.4±0.2	0.3+0.2 0.3-0.1	0.45+0.1 0.45-0.05
1206	3216	3.1±0.1	1.6±0.1	0.45±0.25	0.4+0.2 0.4-0.1	0.55+0.1 0.55-0.05
1210	3225	3.2±0.2	2.6±0.2	0.5±0.2	0.5±0.3	0.6±0.1
2010	5025	5.0±0.2	2.5±0.2	0.6±0.2	0.6±0.2	0.6±0.1
2512	6332	6.4±0.2	3.2±0.2	0.7±0.2	0.7±0.2	0.6±0.1

Gambar 1.42 Dimensi

Seperti yang dijelaskan sebelumnya bahwa fungsi dasar resistor di sini adalah sebagai pembatas arus atau pembagi tegangan. Sesuai dengan hukum Kirchoff:

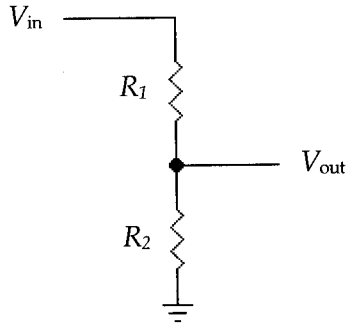
$$V = I \times R$$

V = sumber tegangan

I = arus yang mengalir

R = nilai resistansi

Maka, aliran arus pada resistor akan mengakibatkan terjadinya tegangan sehingga dengan menggunakan dua buah resistor, fungsi resistor juga dapat digunakan sebagai penurun tegangan, yaitu dengan membagi tegangan antara VR1 dan VR2 (Gambar 1.43).



Gambar 1.43 Membagi tegangan antara VR1 dan VR2

Apabila nilai tegangan Vin = 5 volt, R1 = 10 k dan R2 = 20 k, dengan formula

$$V = I \times R \text{ total}$$
$$= I \times (R1 + R2)$$
$$12 = I \times (10\text{ k} + 20\text{ k})$$

akan diperoleh

$$I = \frac{12}{30\text{ k}}$$

I = 0,0004A atau 400uA

Tegangan Vout adalah tegangan yang terjadi pada resistor R2. Tegangan ini dapat dihitung dengan formula

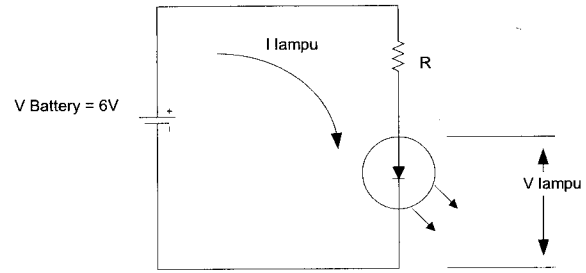
$$V_{R2} = I \times R2$$
$$= 0{,}0004 \times 20.000$$
$$= 8 \text{ volt}$$

Formula tersebut dapat disederhanakan menjadi

$$VR2 = \frac{Vin}{(R1 + R2)} \times R2$$

Fungsi ini dibutuhkan apabila dalam desain rangkaian diperlukan tegangan yang lebih kecil dari sumber tegangan yang ada.

Untuk fungsi resistor sebagai pembatas arus, lihat Gambar 1.44 di mana dengan adanya sumber tegangan 6 volt, kita ingin menyalakan lampu berspesifikasi tegangan 3 volt dan arus maksimum 100mA.



Gambar 1.44 Fungsi pembatas arus

Agar tegangan baterai dapat menyalakan lampu 3 volt, arus yang dikeluarkan melalui baterai terlebih dahulu dihambat oleh sebuah resistor (R). Agar diperoleh tegangan 3 volt pada lampu, tegangan pada resistor adalah 6 – 3 = 3 volt. Agar arus yang mengalir 100mA,

$$I = \frac{VR}{R}$$

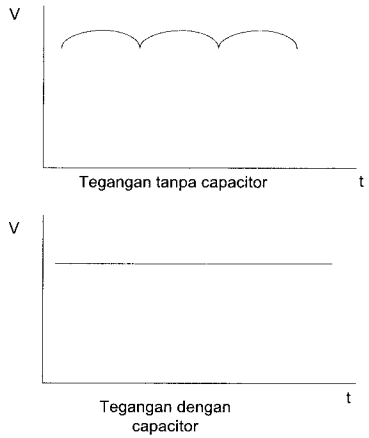
$$100\text{mA} = \frac{3}{R}$$

R = 3/100mA = 30 ohm

2. Kapasitor

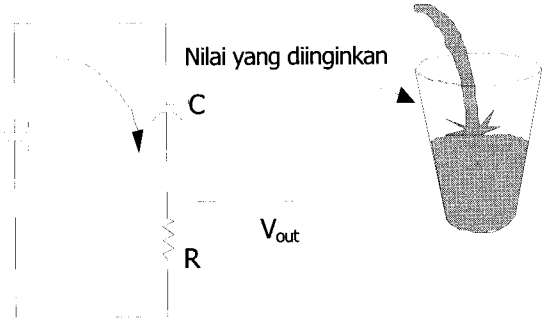
Fungsi dasar kapasitor adalah menyimpan muatan listrik, dan satuan kapasitansi yang digunakan adalah farad. Dengan fungsinya sebagai penyimpan muatan, maka pada aplikasi dalam rangkaian elektronika fungsi tersebut dapat dikembangkan menjadi filter, waktu tunda (*delay*), atau pembangkit getaran (osilator).

Dalam fungsinya sebagai filter, terutama filter frekuensi rendah (*low pass filter*), kapasitor diletakkan secara paralel terhadap bagian-bagian yang akan difilter. Di sana arus listrik yang tidak stabil akan tersimpan di dalam kapasitor sehingga diperoleh tegangan yang lebih rata. Semakin besar nilai kapasitor, semakin rata tegangan yang diperoleh. Hal ini biasa digunakan pada adaptor (*power supply*) DC di mana tegangan yang rata sangat dibutuhkan.



Gambar 1.45 Tegangan dengan dan tanpa kapasitor

Dengan adanya sifat pengisian muatan pada kapasitor, komponen ini juga dapat berfungsi sebagai pembangkit waktu tunda di mana tegangan yang dibangkitkan akan tertunda terlebih dahulu dengan adanya waktu untuk mengisi muatan. Perhatikan Gambar 1.46.



Gambar 1.46 Proses waktu tunda

Proses waktu tunda pada rangkaian Gambar 1.46 dapat dianalogikan dengan proses pengisian air pada gelas ukur di mana dibutuhkan waktu tertentu agar diperoleh air dengan volume yang ditentukan. Demikian pula pada kapasitor, pada kondisi awal kapasitor ibarat gelas kosong yang akan diisi terlebih dahulu oleh muatan listrik yang dihasilkan karena adanya aliran arus ke dalamnya sehingga pada waktu tertentu nilai tegangan kapasitor (Vc) sama dengan tegangan sumber (V). Dengan adanya nilai Vc = V, tegangan pada VR atau Vout akan berubah menjadi 0 volt. Dari sini akan diperoleh waktu tunda di mana Vout akan sama dengan tegangan sumber untuk beberapa saat sebelum turun menjadi 0 volt. Perhitungan waktu tunda dapat diperoleh dengan formula sederhana berikut.

$T = 0{,}7 \times R \times C$

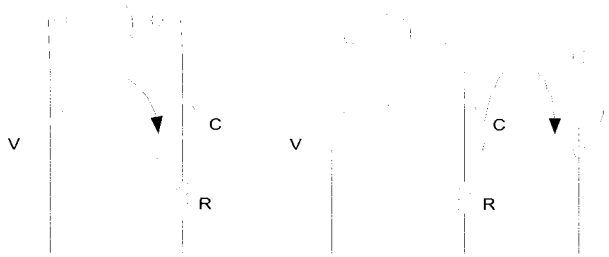
Dengan R = 10 k dan C = 10 uF, maka

$0{,}7 \times 10.000 \times 10 \times 10^{-6}$

= 0,07 detik atau 70 milisekon.

Aplikasi ini biasanya digunakan untuk rangkaian reset dari IC.

Pada fungsi kapasitor sebagai pembangkit getaran listrik (osilator), efek pengisian dan pembuangan muatan pada kapasitorlah yang digunakan. Dengan adanya proses pengisian dan pembuangan yang berulang-ulang, tegangan pada titik tersebut akan naik turun seiring dengan proses pengisian dan pembuangan. Kondisi itu mengakibatkan terjadinya osilasi (getaran).



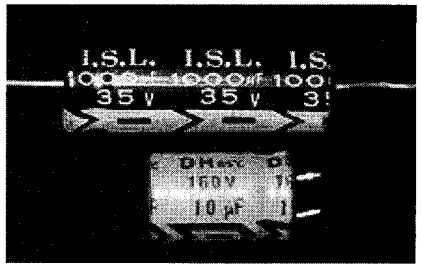
Gambar 1.47 Proses osilasi

Proses ini dapat dilakukan dengan menggunakan dua buah sakelar di mana saat sakelar pertama menutup, arus akan mengalir dan mengisi muatan kapasitor. Kemudian, saat sakelar pertama membuka dan sakelar kedua menutup, arus akan mengalir keluar dari kapasitor dan membuang muatan. Efek ini yang mengakibatkan naik-turunnya tegangan sehingga timbul osilasi.

Selain fungsi-fungsi yang telah disebutkan, kapasitor juga dapat difungsikan sebagai kopling sinyal, *starting capacitor* pada motor, dan lain-lain.

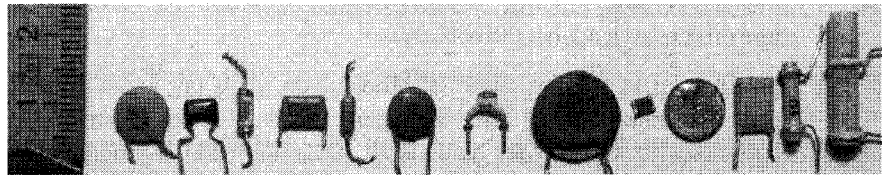
Berikut adalah jenis-jenis kapasitor.

a. Kapasitor elektrolit
   Kapasitor elektrolit didesain dengan bahan elektrolit dan memiliki polaritas positif dan negatif. Polaritas negatif biasanya digambarkan pada tubuh kapasitor dan polaritas positif biasanya berada pada kaki yang terpanjang. Kapasitor ini biasanya memiliki nilai yang cukup besar, yaitu mulai dari 1 uF hingga puluhan ribu uF (mikrofarad) dan biasanya digunakan untuk filter pada *power supply*.



Gambar 1.48 Kapasitor elektrolit

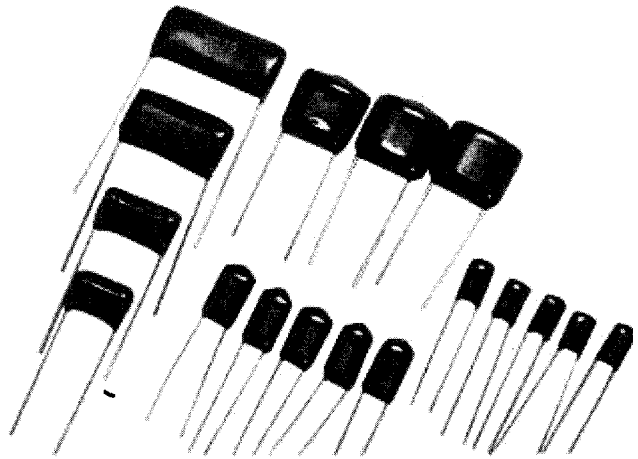
b. Kapasitor keramik
   Kapasitor keramik didesain dengan menggunakan bahan keramik dan berkisar pada nilai 1 pF hingga 680 nF.



Gambar 1.49 Kapasitor keramik

   Cara menghitung nilai kapasitor jenis ini sama dengan cara menghitung nilai resistor SMD, yaitu dua digit pertama adalah nilai dan digit ketiga adalah faktor pengali. Contoh, 224 adalah 22 x 10^4 pF atau 220 nF. Namun, sering kali diperoleh kapasitor dengan satu atau dua digit saja. Pada kapasitor jenis ini, nilai tersebut adalah nilai kapasitansi kapasitor dalam pF.

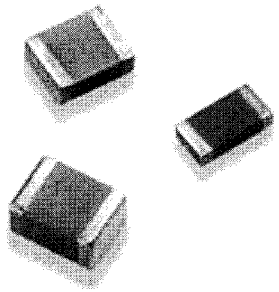
c. Kapasitor milar
   Kapasitor ini biasanya berkisar antara 1 nF hingga 1 uF. Dibandingkan dengan kapasitor keramik, jenis ini lebih tahan terhadap suhu panas.



Gambar 1.50 Kapasitor milar

d. Kapasitor SMD
Selain nilai faradnya, pada kapasitor SMD atau kapasitor pasang permukaan, seperti pada resistor SMD, parameter dimensi juga menjadi parameter utama. Untuk kapasitor berukuran 0,08 x 0,05 inci, tipe dimensinya adalah 0805.

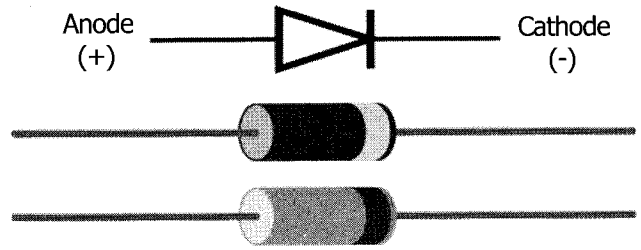
Selain nilai kapasitansi, satu parameter lain yang tak kalah penting pada kapasitor adalah nilai tegangan. Pastikan kapasitor memiliki parameter tegangan yang jauh lebih besar daripada tegangan yang terjadi pada rangkaian. Jenis keramik dan milar biasanya memiliki tegangan puluhan volt sehingga tidak memerlukan perhatian khusus bila Anda bekerja pada rangkaian dengan sumber tegangan baterai 6 atau 12 volt. Namun, untuk jenis elektrolit di mana ada banyak yang memiliki tegangan 16 atau 25 volt, hal ini perlu mendapat perhatian khusus dan usahakan untuk menggunakan tegangan yang jauh di atas sumber daya rangkaian.



Gambar 1.51 Kapasitor

3. Diode

Berbeda dengan resistor dan kapasitor, diode termasuk jenis komponen aktif dengan fungsi utamanya sebagai penyearah. Arus yang bergerak melalui diode hanya dapat mengalir searah dari bagian positif ke bagian negatifnya, sedangkan arah sebaliknya akan terhambat. Hal ini dapat dianalogikan dengan pintu putar pada supermarket yang hanya dapat berputar searah dan tidak dapat berputar ke arah sebaliknya. Seperti pintu putar, diode juga akan memberikan sedikit hambatan pada arus yang mengalir dari positif ke negatif (selanjutnya akan disebut arah maju, sedangkan arah sebaliknya adalah arah mundur) sehingga terjadi penurunan tegangan.

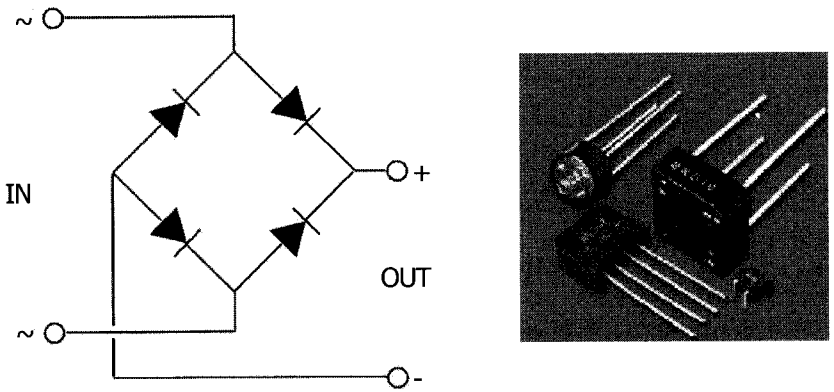


Gambar 1.52 Diode

Untuk diode jenis germanium penurunan tegangannya adalah 0,2 volt, sedangkan untuk diode jenis silikon (paling banyak dijumpai di pasaran) adalah 0,7 volt. Oleh karena itu, fungsi diode dapat dikembangkan menjadi penurun tegangan sebesar 0,7 volt dan penyearah tegangan (AC ke DC).

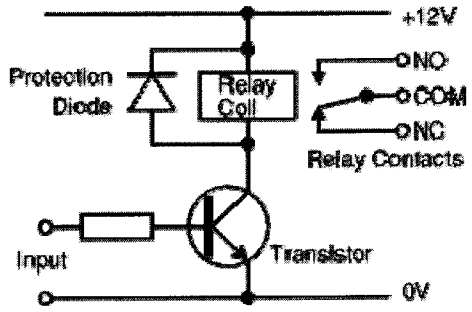
a. Penurun tegangan sebesar 0,7 volt, apabila penurunan tegangan tidak terlalu besar dan terjadi pada arus yang cukup besar di mana metode resistor pembagi tegangan kurang efektif, dengan memberikan diode arah maju, penurunan tegangan sebesar 0,7 volt tanpa harus memperhatikan impedansi rangkaian beban dapat dilakukan.

b. Penyearah tegangan (AC ke DC), hal ini biasa digunakan pada rangkaian *power supply* yang menyearahkan tegangan sinus dari trafo menjadi bentuk DC tak teregulasi melalui empat buah diode yang biasa disebut *bridge* atau penyearah gelombang penuh (*full wave rectifier*). Untuk membuat desain yang ringkas dan lebih ekonomis, sering kali penyearah ini juga terdiri atas dua buah diode, yaitu penyearah setengah gelombang (*half wave rectifier*).

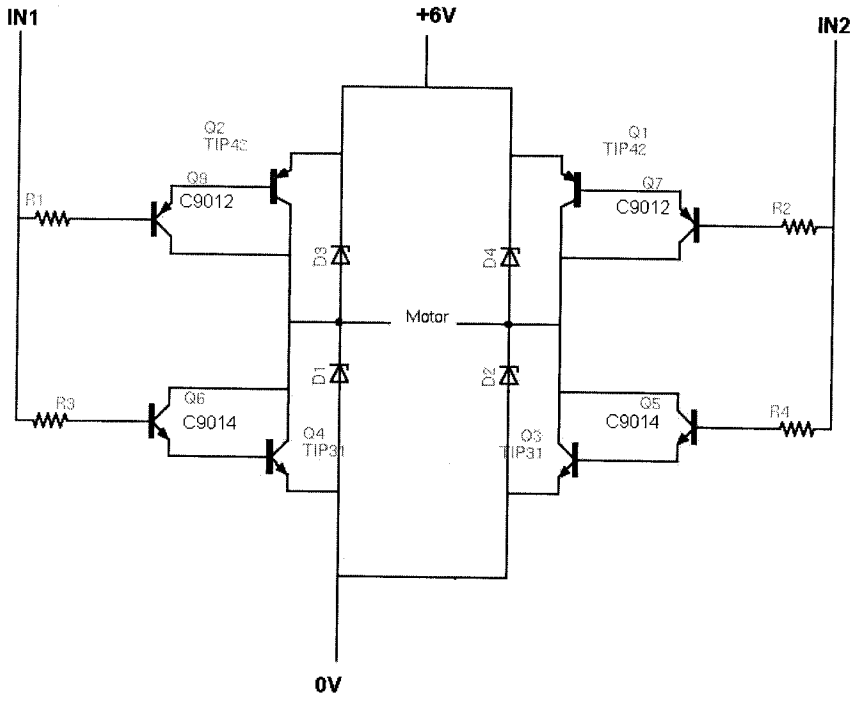


Gambar 1.53 Penyearah tegangan

c. Sebagai pengaman komponen-komponen induktif seperti motor dan relai. Motor dan relai memiliki kumparan, yaitu komponen yang bersifat induktif di mana medan magnet yang ditimbulkan oleh aliran arus pada kumparan tersebut sering kali memberikan tegangan balik yang cukup besar. Penggunaan diode yang dipasang secara membalik dan paralel pada kumparan akan mengakibatkan arus balik mengalir melalui diode tersebut dan efek medan magnet akan segera hilang. Hal ini dibutuhkan untuk mengamankan komponen-komponen lain dari kerusakan.



Gambar 1.54 Pengaman relai

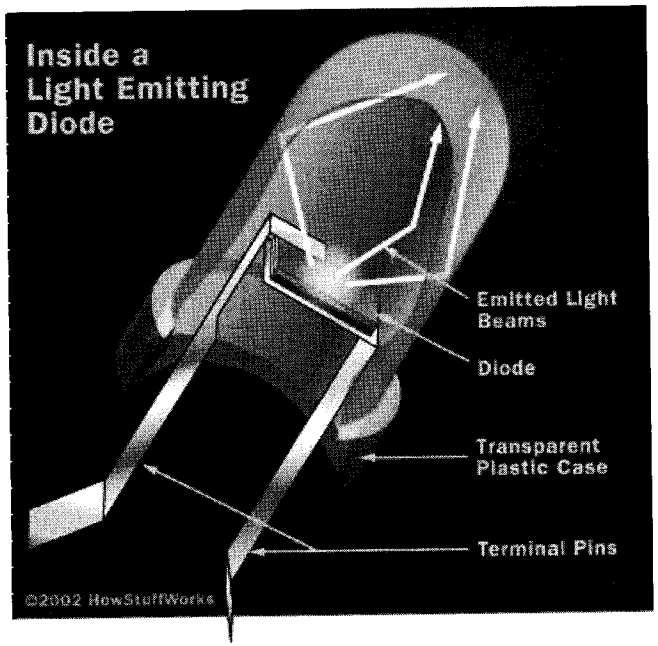


Gambar 1.55 Pengaman motor

Pada motor di mana arah arus yang mengalir dapat dilakukan secara bolak-balik dibutuhkan empat buah diode yang akan mengamankan transistor-transistornya dari tegangan induksi balik.

4. ***Light Emitting Diode*** **(LED)**

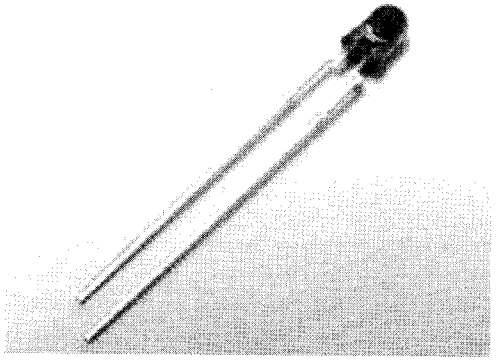
LED juga termasuk dalam jenis diode, hanya saja LED akan mengeluarkan cahaya apabila arus mengalir maju terjadi. Namun, ada hal yang perlu diperhatikan pada LED, yaitu tegangan maju. LED memiliki tegangan maju yang terbatas sesuai dengan spesifikasi yang disebutkan pada lembar data atau *data sheet*-nya, biasanya berkisar antara 2,5 hingga 3 volt, walaupun kadang-kadang ada juga yang lebih. Oleh karena itu, dalam mendesain rangkaian untuk mengaktifkan LED, perlu diperhatikan agar tegangan yang terjadi tidak melebihi tegangan maju yang diizinkan.



Gambar 1.56 LED

LED biasanya berfungsi sebagai indikator tegangan pada suatu titik. Contohnya pada input *power supply* di mana nyala LED mengindikasikan masuknya tegangan, atau pada jalur data di mana kedipan LED mengindikasikan lewatnya aliran data, dan lain-lain.

Bila pada LED biasa cahaya yang dikeluarkan dapat dilihat oleh mata, pada LED inframerah cahaya tersebut tidak tampak oleh mata. Cahaya ini dapat dilihat dengan menggunakan kamera di mana spektrum cahaya yang ditangkap melebihi jangkauan spektrum mata kita. LED inframerah biasanya memiliki warna bening, namun beberapa jenis, terutama jenis *high power infrared*, memiliki warna kebiru-biruan seperti pada Gambar 1.57.



Gambar 1.57 LED jenis *high power infrared*

Sifat cahaya inframerah adalah sebagai berikut.

a. Tidak tampak oleh mata.

b. Dapat dipancarkan pada jarak yang cukup jauh.

c. Memiliki spektrum yang terbatas sehingga mudah untuk membedakan antara cahaya biasa dan inframerah.

Fungsi LED inframerah adalah untuk memancarkan cahaya inframerah agar dapat diterima oleh komponen-komponen yang dapat merespons cahaya pada spektrum tersebut, yaitu fototransistor atau fotodiode yang akan dibahas pada subbab berikutnya. Selain itu, akhir-akhir ini LED inframerah juga sering kali digunakan untuk menerangi objek yang dipantau kamera pada tempat-tempat yang gelap.

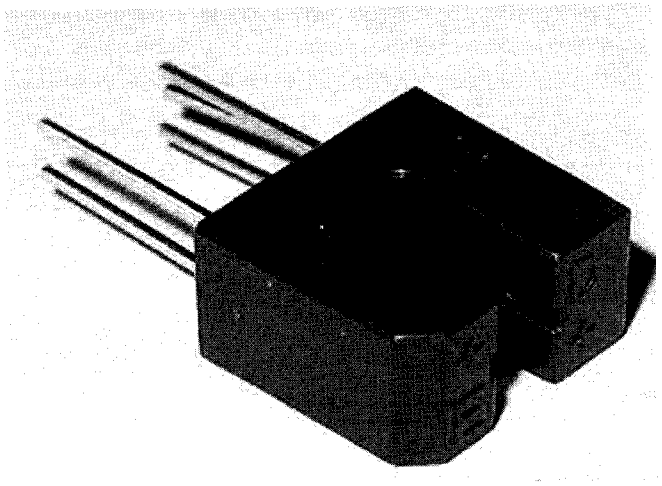
Dari fungsi dan sifat-sifatnya tersebut, fungsi LED inframerah dapat dikembangkan lagi menjadi seperti berikut:

a. *Remote control* inframerah yang akan mengirimkan data-data dalam bentuk sinyal-sinyal inframerah,

b. Sensor garis di mana cahaya inframerah akan memantul pada objek terang dan terserap pada objek gelap (detail mengenai hal ini akan dibahas pada bagian sensor),

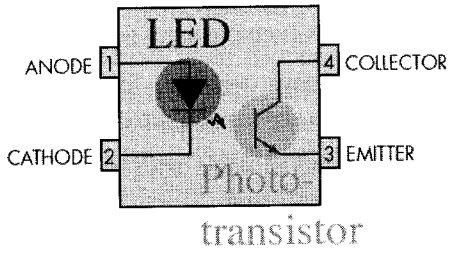
c. Sensor jarak, untuk jarak dekat (beberapa puluh sentimeter) kuat lemah cahaya inframerah yang diterima dapat mewakili informasi jarak dari sensor,

d. Enkoder, dengan menempatkan LED inframerah dan fotodiode serta piringan berlubang di antaranya, LED dapat digunakan untuk menghitung putaran roda. Untuk aplikasi ini sering kali LED dan dan fotodiode sudah didesain dalam bentuk yang lebih ringkas, yaitu *opto interrupter* seperti pada Gambar 1.58.



Gambar 1.58 *Opto interrupter*

e. Sebagai isolasi tegangan di mana LED dan fotodiode didesain dalam satu keping IC, biasa disebut *optocoupler*.



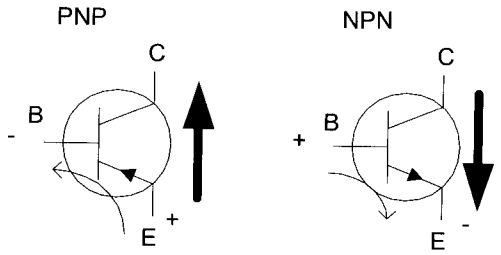
Gambar 1.59 LED dan fototransistor

f. Dan masih banyak lagi fungsi yang lain tergantung pada kebutuhan dan kreativitas pengguna.

5. Transistor

Transistor adalah komponen semikonduktor yang dapat digunakan untuk penguat, rangkaian pemutus dan penyambung, stabilisasi tegangan, modulasi sinyal, dan sebagainya. Transistor dapat berfungsi seperti keran listrik yang memungkinkan pengaliran listrik yang sangat akurat dari sirkuit sumber listriknya.

Pada umumnya, transistor memiliki tiga terminal, yaitu emitor, kolektor, dan basis di mana basis berfungsi sebagai pengatur keran. Berdasarkan polaritasnya, terdapat dua jenis transistor, yaitu NPN dan PNP. Transistor NPN (negatif-positif-negatif) akan mengalirkan arus dari kolektor ke emitor dan transistor PNP (positif-negatif-positif) akan mengalirkan arus dari emitor ke kolektor.



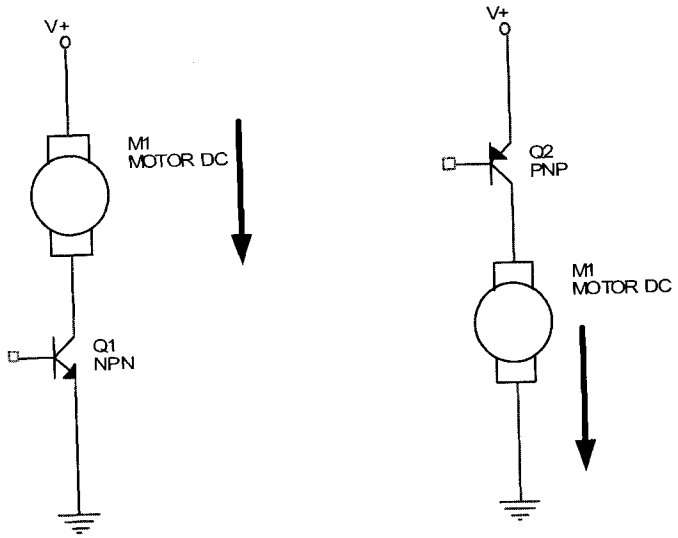
Gambar 1.60 Transistor PNP dan NPN

Agar terjadi aliran arus pada kolektor-emitor atau sebaliknya, basis transistor harus mendapat sedikit aliran arus sebagai pembuka keran. Besarnya aliran arus menentukan besarnya keran tersebut dibuka. Semakin besar arus basis, semakin besar pula aliran arus pada kolektor-emitor atau sebaliknya, hingga pada kondisi terbesar yang disebut *saturasi* di mana kolektor dan emitor seakan terhubung singkat. Sebaliknya, kondisi arus basis yang semakin kecil hingga aliran arus terhenti akan disebut kondisi *cut off*.

Transistor NPN dan PNP ini berbeda dalam cara kerjanya. Transistor NPN (negatif pada kolektor, positif pada basis, dan negatif pada emitor) mendapat arus positif dari basis ke emitor, maka kolektor akan terhubung singkat dengan emitor yang negatif sehingga keduanya menjadi negatif, sedangkan transistor PNP (positif pada kolektor, negatif pada basis, dan positif pada emitor) mendapat arus negatif dari basis ke emitor, maka

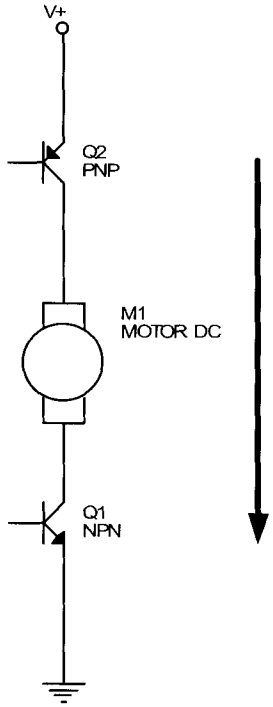
kolektor akan terhubung singkat dengan emitor yang positif sehingga keduanya menjadi positif (Gambar 1.60).

Dalam aplikasi robotik, transistor lebih sering difungsikan sebagai sakelar di mana transistor NPN akan menghubungkan titik kolektor ke referensi negatif (*ground*) sehingga arus akan mengalir dari motor ke referensi negatif, sedangkan transistor PNP akan menghubungkan titik kolektor ke referensi positif (V+) sehingga arus akan mengalir masuk dari referensi positif ke motor.



Gambar 1.61 Transistor sebagai sakelar

Dengan kombinasi kedua transistor tersebut dapat diperoleh rangkaian yang disebut *push pull* (mendorong dan menarik) di mana arus akan didorong masuk dari referensi positif ke motor oleh transistor PNP dan ditarik keluar dari motor ke *ground* oleh transistor NPN sehingga diperoleh aliran arus yang cukup kuat pada motor.



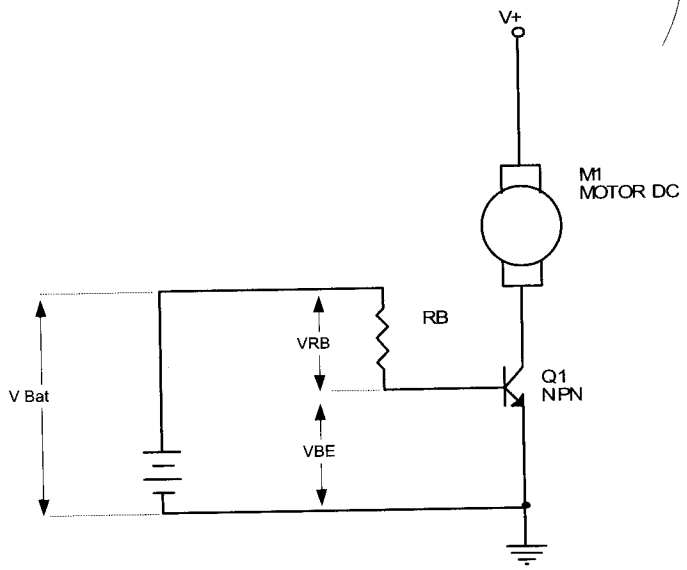
Gambar 1.62 Rangkaian *push pull*

Kombinasi dua buah rangkaian *push pull* (2 NPN dan 2 PNP) akan membentuk rangkaian di mana arah aliran arus motor dapat diatur. Rangkaian ini disebut *H-bridge* dan akan kita bahas pada subbab selanjutnya.

Seperti yang telah disebutkan sebelumnya, besar arus yang mengalir antara kolektor dan emitor tergantung pada besar arus yang mengalir pada basis. Arus kolektor emitor mempunyai nilai yang jauh lebih besar daripada arus basis dan berbanding lurus. Besar nilai arus tersebut tergantung pada nilai penguatan (HFE) dari masing-masing transistor yang dapat dilihat pada lembar data transistor tersebut.

$$I_C = I_B \times H_{FE}$$

Agar diperoleh arus yang sesuai pada basis, dibutuhkan R_B atau resistor yang terhubung seri dengan basis.



Gambar 1.63 R_B

Untuk menghitung nilai R_B dapat dilakukan dengan metode pembagi tegangan seperti yang dijelaskan sebelumnya. V_{BE} transistor biasanya 0,7 volt. Oleh karena itu, apabila tegangan baterai (VBat) = 6 volt, arus maksimal motor 1A dan H_{FE} transistor 100,

$$V_{RB} = V_{Bat} - 0,7$$

$$= 5,3 \text{ volt}$$

$$I_B = \frac{IC}{HFE}$$

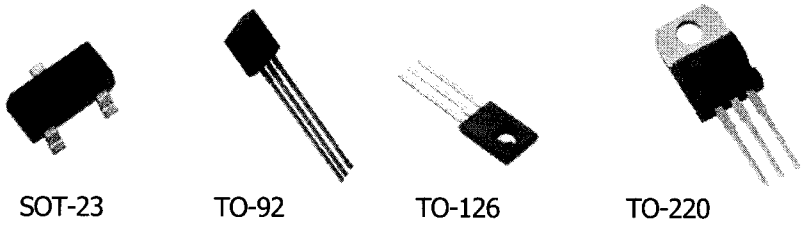
$$= \frac{1}{100} = 0,01A \text{ atau } 10mA$$

$$R_B = \frac{VRB}{IB}$$

$$= \frac{5,3}{0,01} = 530$$

Nilai resistor di pasaran yang mendekati nominal tersebut adalah 470 ohm dan 560 ohm, namun untuk memastikan arus yang lewat pada kolektor betul-betul 1A, sebaiknya gunakan nilai yang sedikit lebih kecil. Perlu dipastikan agar nilai resistor tersebut tidak terlalu kecil karena arus yang berlebih akan mengakibatkan kerusakan pada transistor.

Berdasarkan bentuknya, transistor dapat dibedakan menjadi beberapa macam. Berikut adalah bentuk-bentuk transistor yang digunakan pada aplikasi-aplikasi robot.



Gambar 1.64 Bentuk-bentuk transistor

Berikut adalah teknik untuk mengetahui kaki-kaki transistor dan sekaligus menguji apakah transistor mengalami kerusakan atau tidak.

a. Dengan multimeter digital
   1) Pasang multimeter digital pada skala hambatan 10 k atau 20 k.
   2) Pasang kabel merah di basis transistor.
   3) Pasang kabel hitam di kolektor transistor, dan multimeter akan menunjukkan nilai impedansi.
   4) Pasang kabel hitam di emitor transistor, dan multimeter akan menunjukkan nilai impedansi lagi.
   5) Pasang kabel hitam di basis dan coba pasang kabel merah di kolektor dan emitor secara bergantian. Multimeter tidak boleh menunjukkan impedansi apa pun.
   6) Pemeriksaan transistor PNP juga bisa menggunakan langkah-langkah di atas, namun gantikan kabel merah dengan kabel hitam, dan sebaliknya.

b. Dengan multimeter analog

1) Pasang multimeter digital pada skala hambatan 10 k atau 20 k.
2) Pasang kabel hitam di basis transistor.
3) Pasang kabel merah di kolektor transistor dan jarum multimeter akan bergerak ke kanan.
4) Pasang kabel hitam di emitor transistor dan jarum multimeter akan bergerak ke kanan lagi.
5) Pasang kabel hitam di basis dan coba pasang kabel merah di kolektor dan emitor secara bergantian. Multimeter tidak boleh menunjukkan impedansi apa pun.
6) Pemeriksaan transistor PNP juga bisa menggunakan langkah-langkah di atas, namun gantikan kabel merah dengan kabel hitam, dan sebaliknya.

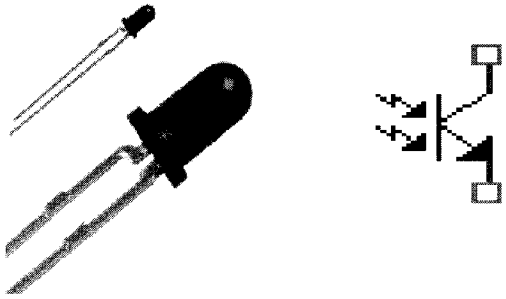
Jadi intinya, teknik pada pengujian transistor NPN dengan multimeter analog akan sama dengan pengujian transistor PNP dengan multimeter digital, dan demikian pula dengan transistor PNP di multimeter analog dan transistor NPN di multimeter digital.

6. Fototransistor

Seperti halnya pada transistor, fototransistor juga berfungsi sebagai keran arus yang akan mengalirkan arus melalui kolektor dan emitor. Perbedaannya hanya terletak pada proses pembukaan kerannya. Bila pada transistor biasa dilakukan dengan mengalirkan arus ke basis, pada fototransistor dilakukan dengan mengarahkan sinar inframerah kepadanya.

Besar-kecilnya arus yang mengalir pada kolektor dan emitor ditentukan oleh kuat lemah cahaya inframerah yang diterima. Semakin besar cahaya inframerah yang diterima, semakin besar pula arus kolektor emitornya.

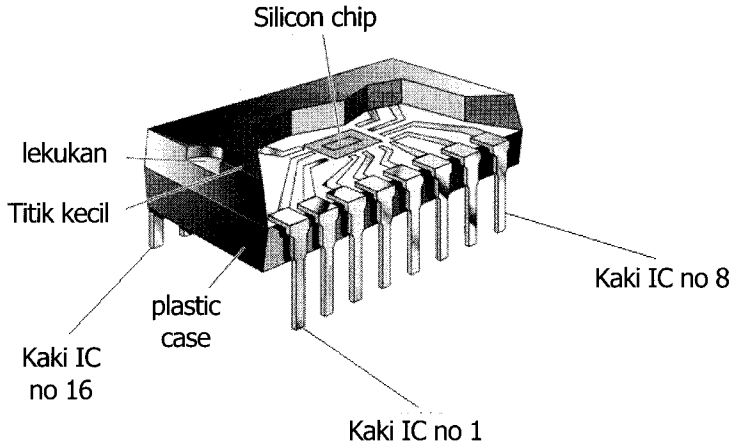
Berdasarkan sifatnya tersebut, fototransistor juga dapat digunakan sebagai pengukur jarak untuk skala beberapa sentimeter. Untuk skala yang lebih jauh, keluaran fototransistor harus dilewatkan ke IC penguat terlebih dahulu.



Gambar 1.65 Fototransistor

7. *Integrated circuit* (IC)

Integrated circuit (IC) atau sirkuit terpadu merupakan rangkaian yang terintegrasi dan terdiri atas puluhan dan bahkan ratusan transistor yang terintegrasi dalam satu keping kemasan.



Gambar 1.66 IC DIP

Gambar 1.66 adalah bentuk IC yang sering kita jumpai, yaitu *dual inline package* (DIP) di mana masing-masing kaki memiliki jarak 0,1 inci atau 2,54 mm. IC bentuk ini akan selalu memiliki lekukan untuk penanda arah pemasangan IC.

Titik kecil menunjukkan nomor urut pertama dari kaki IC dan selanjutnya penomoran kaki akan melingkar mengelilingi bentuk IC, berputar melawan arah jarum jam, sehingga nomor kaki terbesar akan berada di seberang kaki nomor satu.

Bagian *silicon chip* atau keping silikon merupakan bagian inti dari IC di mana kumpulan transistor terdapat di dalamnya. Kawat-kawat tipis akan menghubungkan kepingan tersebut dengan kaki-kaki IC.

Selain bentuk DIP, IC juga memiliki bentuk pasang permukaan atau SMD yang ukurannya jauh lebih kecil dan ringkas.

Gambar 1.67 IC SMD

IC SMD didesain untuk langsung disolder ke PCB. Oleh karena itu, biasanya IC jenis ini memiliki daya tahan panas yang lebih daripada IC DIP. IC SMD memiliki berbagai macam tipe berdasarkan bentuk-bentuknya. Berikut ini adalah IC SMD yang sering dijumpai di pasaran.

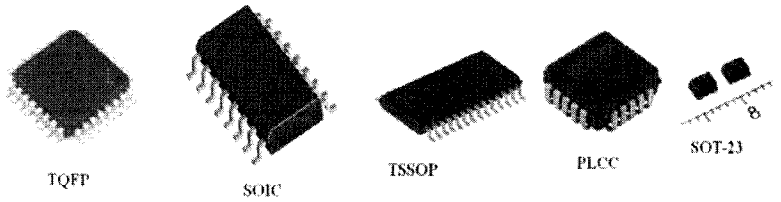
a. *Small outline integrated circuit* atau SOIC, berukuran sekitar 30-50% dari bentuk DIP dan memiliki urutan kaki yang sama dengan bentuk DIP. Tipe ini biasa disebut SO-xx di mana xx adalah jumlah kaki IC.

b. *Thin quad flat pack* atau TQFP, IC bentuk ini memiliki kaki yang mengelilingi seluruh tubuhnya dan biasanya berbentuk bujur sangkar.

c. *Thin shrink small outline package* atau TSSOP, memiliki bentuk yang lebih kecil daripada SOIC.

d. *Plastic lead chip carrier* atau PLCC, berbentuk bujur sangkar dengan kaki yang mengelilingi tubuhnya. IC PLCC lebih sering digunakan untuk dipasang di soket daripada disolder langsung ke permukaan.

e. *Small outline transistor* atau SOT-23



Gambar 1.68 Beberapa IC SMD

Kebanyakan IC memiliki fungsi-fungsi yang spesifik, misalnya

a. IC *operational amplifier* untuk penguatan dan biasanya digunakan pada rangkaian sensor pada robot,

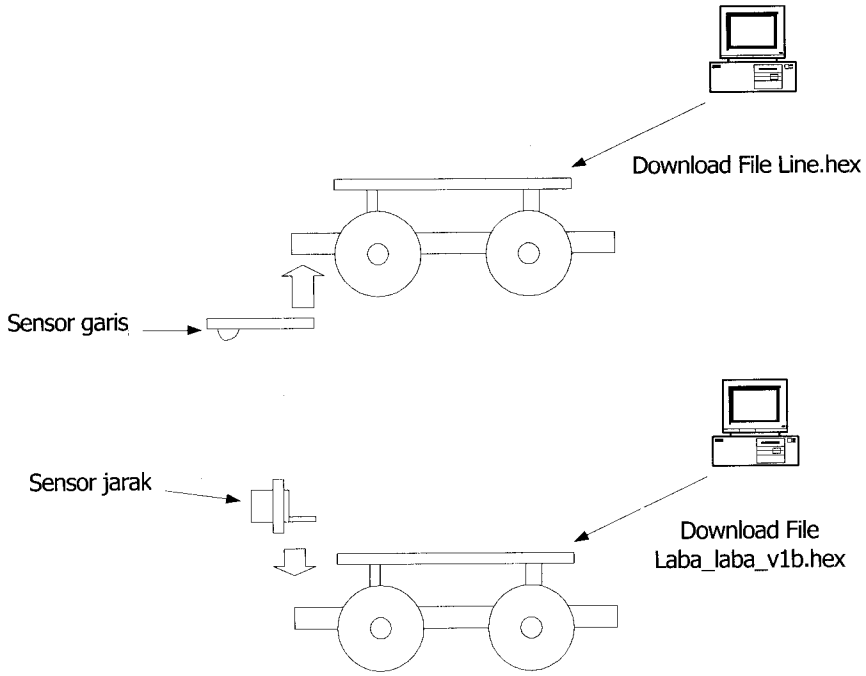
b. IC *power*, untuk *power supply* dan bentuknya biasanya menyerupai transistor TO-220, dan

c. IC gerbang logika untuk fungsi-fungsi logika.

Selain IC-IC yang memiliki fungsi spesifik tersebut, ada juga IC yang memiliki fungsi yang lebih universal dan bisa diprogram sehingga kita dapat membentuk berbagai macam rangkaian hanya dengan mengubah isi program di dalamnya. IC jenis ini termasuk golongan IC cerdas, yaitu mikrokontroler.

Mikrokontroler terdiri atas ribuan transistor dan terdapat berbagai macam fungsi logika, memori, dan bahkan beberapa mikrokontroler tertentu memiliki *operational amplifier* dan fungsi-fungsi khusus seperti ADC. Pada kehidupan kita sehari-hari IC mikrokontroler banyak digunakan pada berbagai macam perangkat elektronik seperti ponsel, televisi, kalkulator, dan sebagainya.

Dengan sebuah mikrokontroler, rangkaian elektronika dapat diprogram untuk berbagai macam fungsi, contohnya fungsi penjejak garis (fail linef.hex) dengan mudah dapat diganti menjadi fungsi penghindar halangan dengan penambahan sensor jarak dan pengisian program penghindar halangan (fail laba_laba_v1b.hex).



Gambar 1.69 Memprogram rangkaian elektronika

Ada beberapa macam bahasa pemrograman untuk mikrokontroler, seperti assembly, C, Basic. Namun, fail yang diunduh adalah fail heksa dengan ekstensi HEX. Bahasa pemrograman berfungsi untuk mempermudah manusia dalam memprogram mikrokontroler dengan bahasa-bahasa yang lebih manusiawi, sedangkan fail heksa yang biasa disebut *program object* dibuat agar mikrokontroler dapat membaca perintah-perintah yang kita programkan. Konversi dari fail yang dibuat dalam bahasa pemrograman ke bentuk *program object* dilakukan dengan proses kompilasi atau *compile*.

Pengisian program dari komputer ke dalam mikrokontroler dilakukan dengan alat yang disebut *programmer* atau *downloader*. Beberapa jenis robot telah dilengkapi oleh unit *programmer* atau *downloader*, namun beberapa jenis lain yang tidak memilikinya menggunakan *downloader* terpisah, yaitu *delta USB in system programming* (DU ISP).

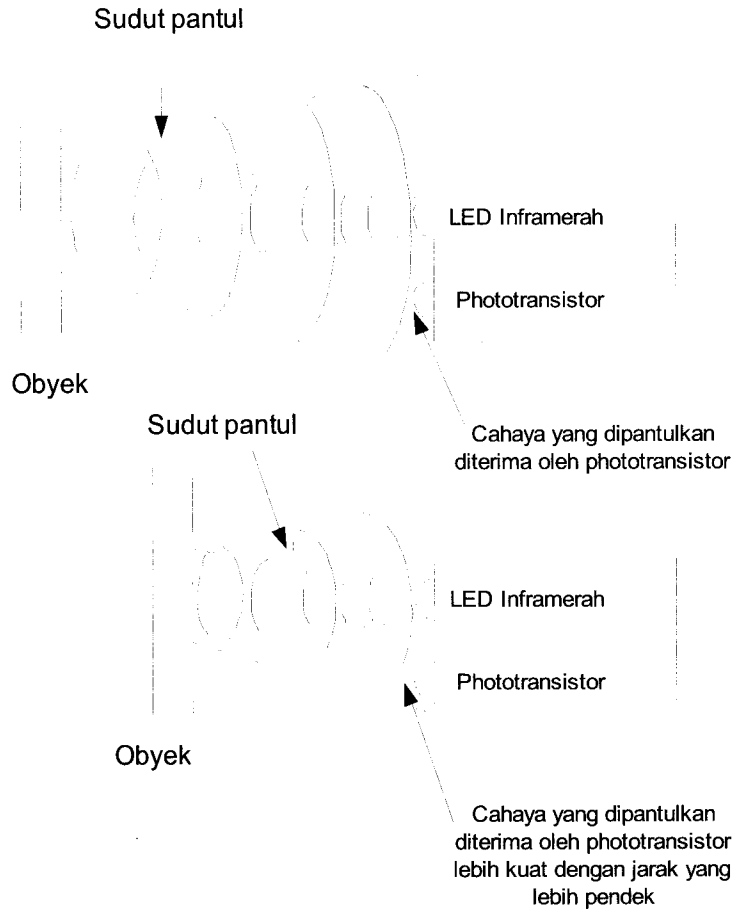
Gambar 1.70 DU ISP

Dalam aplikasi robot kali ini, mikrokontroler yang kita gunakan adalah mikrokontroler keluaran Atmel yang pemrogramannya cukup mudah. Mikrokontroler ini menggunakan teknik *in system programming* (ISP) sehingga tidak membutuhkan perangkat *programmer* yang eksklusif. Kita cukup menggunakan rangkaian yang mengonversi USB ke koneksi serial dengan protokol yang sudah disediakan oleh perangkat lunak *downloader* Atmel yang sifatnya gratis dan dapat diunduh di www.atmel.com. Proses unduh program sudah dapat dilakukan dengan mudah.

Mikrokontroler yang kita gunakan adalah AT89S51 di mana telah tersedia tidak hanya perangkat lunak kompiler C dan *assembler* saja, namun juga kompiler untuk pemrograman visual yang lebih mudah dan manusiawi untuk dipelajari. Masalah pemrograman ini akan dijelaskan pada subbab lebih lanjut, yaitu Bagian Pemrograman Robot.

8. Sensor jarak

Berdasarkan teknik yang digunakan, ada dua jenis sensor jarak pada aplikasi robotik, yaitu sensor ultrasonik dan sensor inframerah. Sensor jarak dengan teknik inframerah bekerja dengan memancarkan cahaya inframerah ke objek dan mengukur kuat lemah pantulannya.



Gambar 1.71 Sensor jarak dengan teknik inframerah

Saat cahaya inframerah mengenai objek, cahaya akan dipantulkan dengan sudut yang sama dengan sudut datang sehingga pantulan akan semakin menyebar dan intensitasnya melemah. Semakin dekat objek dengan sensor, semakin kuat pantulan yang diterima dan semakin besar pula arus yang mengalir pada kolektor emitor fototransistor (lihat bagian fototransistor).

Selama digunakan pada medan yang tidak dipengaruhi cahaya-cahaya inframerah (sinar matahari biasanya mengandung cahaya inframerah),

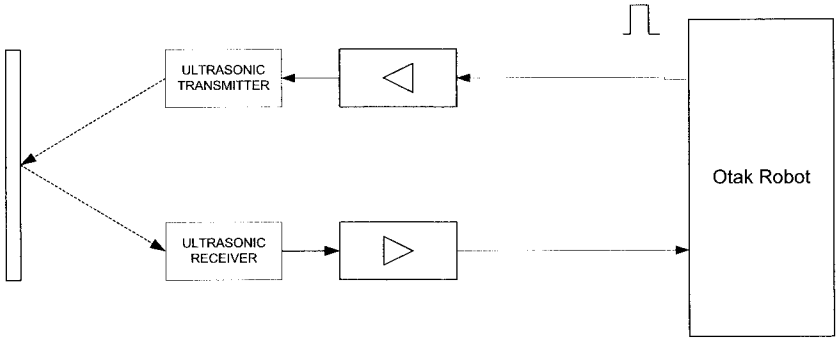
pengukuran jarak menggunakan teknik inframerah ini akan memperoleh hasil yang sangat presisi hingga dalam ordo milimeter. Gunakan fototransistor yang memiliki lapisan pelindung cahaya seperti TSOP 050TB2 untuk mereduksi gangguan cahaya-cahaya liar. Pengukuran jarak dengan menggunakan inframerah ini biasanya digunakan untuk hal sebagai berikut:

a. Pengukuran jarak dekat di bawah 10 sentimeter. Untuk jarak hingga puluhan sentimeter dibutuhkan penguat tambahan berupa IC *instrument amplifier* seperti INA125, yaitu IC *operational amplifier* khusus untuk instrumentasi sehingga memiliki penguatan yang sangat besar.

b. Pengukuran pada medan dengan objek yang lunak seperti kain di mana ultrasonik tidak dapat memantulkan sinyalnya.

Berbeda dengan inframerah, teknik ultrasonik melakukan pengukuran jarak dengan menghitung kecepatan rambat gelombang. Semakin jauh jarak yang diukur, semakin lama kecepatan rambat gelombang tersebut Dibandingkan dengan teknik inframerah, teknik ultrasonik memiliki jangkauan pengukuran yang lebih jauh, biasanya mencapai 3-4 meter. Untuk jarak yang lebih jauh dapat dilakukan dengan memperkuat sinyal ultrasonik dengan bantuan rangkaian *H-bridge* yang digunakan pada pengendali motor.

Proses pengukuran dilakukan dengan menembakkan sinyal ultrasonik dan menghitung kapan sinyal tersebut diterima kembali oleh sensor. Proses penghitungan ini dilakukan oleh bagian otak robot yang biasanya menggunakan IC mikrokontroler. Tampak pada Gambar 1.72 bagian otak robot yang mengirimkan sinyal ultrasonik yang dikuatkan ke *ultrasonic transmitter* dan terpancar menuju objek. Pantulan sinyal dari objek akan diterima oleh *ultrasonic receiver* dan melalui bagian penguat diterima kembali oleh otak robot.



Gambar 1.72 Sistem pengukuran jarak dengan ultrasonik

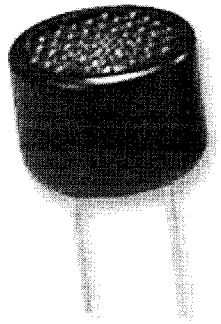
Mikrokontroler akan berhenti melakukan penghitungan saat sinyal ultrasonik diterima kembali. Penghitungan waktu dari saat sinyal ultrasonik pertama kali dipancarkan hingga diterima telah diperoleh. Jarak yang ditempuh oleh rambatan gelombang ultrasonik mulai dari dipancarkan hingga memantul pada objek dan kembali diterima akan diperoleh dengan hasil kali antara kecepatan rambatan suara dan waktu yang diperoleh.

$$S = V \times t$$

$$= 34399{,}22 \times t$$

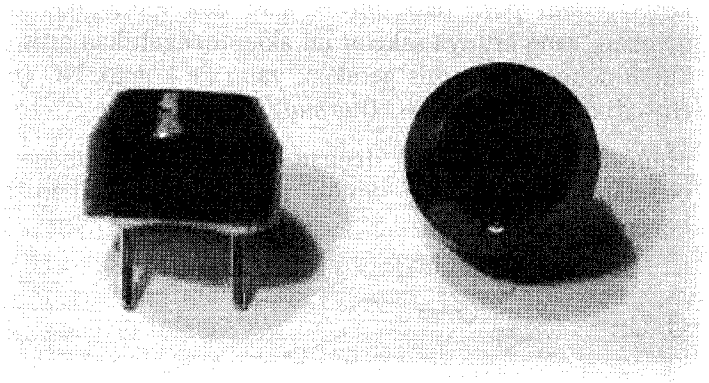
Proses penghitungan jarak ini menggunakan penghitungan waktu dan selain cukup rumit, sering kali juga cukup menyibukkan otak robot. Untuk mempermudah proses tersebut digunakan modul elektronik khusus untuk pengukuran jarak, misalnya *D-sonar*, seperti yang akan dijelaskan pada subbab Modul Elektronik. Dengan bantuan modul ini otak robot hanya perlu memerintahkan pengukuran dan mengambil info jarak yang diterima tanpa perlu melakukan penghitungan lagi.

Gambar 1.73 adalah sensor ultrasonik yang digunakan untuk pengukuran jarak di mana sepasang sensor terdiri atas sensor ultrasonik khusus untuk pemancar (SQ40T) dan untuk penerima (SQ40R).



Gambar 1.73 Sensor ultrasonik

Untuk aplikasi-aplikasi khusus di mana sensor digunakan di medan-medan ekstrem seperti di dalam air, kita dapat menggunakan jenis yang tahan air (*waterproof*) seperti EC4018.



Gambar 1.74 EC4018

9. Sakelar

Sakelar merupakan komponen pemutus dan penghubung arus pada dua buah kepingan logam (lebih dari dua keping untuk beberapa jenis tertentu). Fungsi sakelar pada aplikasi robotika adalah sebagai media input dari operator agar dapat diterima oleh otak robot. Selain itu, sakelar juga dapat berfungsi sebagai sensor sederhana di mana robot akan berhenti atau melakukan sesuatu saat sakelar terbentur. Contohnya pada sakelar yang

dipasang di bagian depan robot di mana robot akan berhenti dan mundur saat sakelar tersebut membentur objek. Sakelar ini biasa disebut sakelar pembatas atau *limit switch.*

Berdasarkan proses kontaknya, sakelar dapat dibedakan menjadi beberapa jenis, yaitu

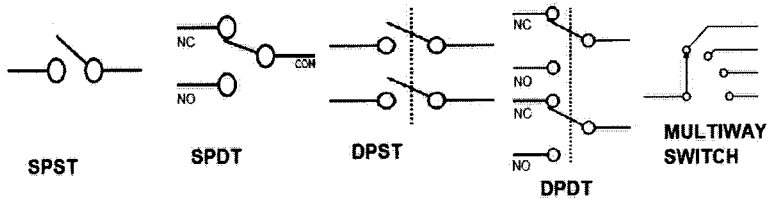
a. SPST, *single pole single throw* atau satu kutub dan satu aksi gerakan, yang artinya sakelar ini akan mengalirkan arus dari satu kutub dengan satu gerakan saja.

b. DPST, *dual pole single throw atau* dua kutub dan satu aksi gerakan, yang artinya sakelar ini akan mengalirkan arus dari dua kutub dengan satu gerakan saja

c. SPDT, *single pole dual throw* atau satu kutub dan dua aksi gerakan, yang artinya sakelar ini akan mengalirkan arus dari satu kutub dengan dua jenis gerakan, yaitu ke kontak NC (*normally closed*) dan ke kontak NO (*normally open*).

d. DPDT, *dual pole dual throw* atau dua kutub dan dua aksi gerakan, yang artinya sakelar ini akan mengalirkan arus dari dua kutub dengan dua jenis gerakan, yaitu ke kontak NC (*normally closed*) dan ke kontak NO (*normally open*).

e. *Multiway switch*, sakelar dengan banyak jalan di mana terdapat beberapa aksi gerakan untuk memindahkan aliran arus ke beberapa kontak.



Gambar 1.75 Jenis sakelar berdasarkan proses kontaknya

Berdasarkan fungsinya, sakelar dapat dibedakan menjadi jenis-jenis berikut:

a. *DIP switch*, biasanya berupa 4, 8, atau 10 sakelar *single throw* dan biasa digunakan untuk pengaturan program pada rangkaian.

b. *Push button* atau sakelar tekan, ada dua macam, yaitu yang kembali saat dilepas dan yang harus ditekan lagi untuk melepas. Sakelar pada jenis kedua dapat dimodifikasi menjadi jenis pertama dengan melepaskan pengaitnya. Untuk sakelar yang kembali saat dilepas biasanya digunakan untuk sensor atau tombol start, sedangkan sakelar *push button* yang harus ditekan lagi untuk kembali biasa digunakan untuk *power.*

c. *Rocker switch*, sama seperti sakelar *push button*, model ini juga memiliki jenis yang kembali saat dilepas dan tidak. Namun, kebanyakan tidak kembali saat dilepas sehingga harus ditekan di sisi sebelahnya. Sakelar ini biasanya dijumpai pada sakelar *power* tegangan 220 V dan kerap kali memiliki tambahan lampu sebagai indikator.

d. *Sliding switch*, merupakan sakelar geser, biasa digunakan untuk sakelar *power* pada tegangan rendah atau untuk pemilih mode.

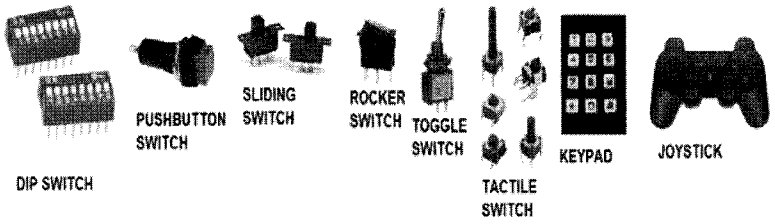
e. *Toggle switch*, berbentuk tangkai dan biasa digunakan pada panel-panel yang membutuhkan kemudahan untuk menekan.

f. *Tactile switch*, termasuk dalam sakelar *push button*, namun bentuknya *PCB mounted* (untuk disolder di PCB). Biasa digunakan untuk sakelar reset, pengatur mode, atau dirangkai menjadi *keypad.*

g. *keypad*, rangkaian *tactile switch* yang dikombinasi dalam bentuk matriks sehingga dapat memberikan lebih banyak jenis input ke robot.

h. *Joystick*, pada dasarnya *joystick* adalah sakelar berbentuk tangkai dengan dua buah potensio (resistor variabel) di dalamnya yang dapat diputar-putar ke berbagai arah dan menciptakan berbagai macam variasi tegangan sebagai input sehingga otak robot dapat mengetahui sedang di posisi mana tangkai tersebut berada. Namun, dewasa ini *joystick* sudah dikembangkan menjadi lebih lengkap dan tersedia juga *push button* sehingga diperoleh variasi

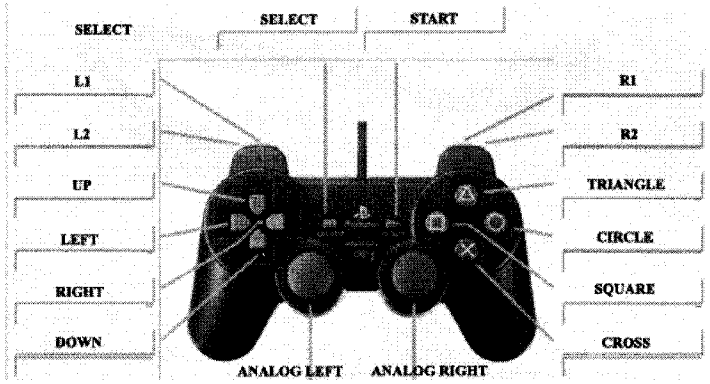
yang jauh lebih banyak. Penggunaan *joystick* pada robot sebagai input dari operator jauh lebih fleksibel daripada jenis sakelar yang lain.



Gambar 1.76 Jenis sakelar berdasarkan fungsinya

Joystick Playstation memiliki tombol-tombol sebagai berikut:

1) *push button* L1;
2) *push button* L2;
3) *push button* R1;
4) *push button* R2;
5) *push button* Up;
6) *push button* Down;
7) *push button* Left;
8) *push button* Right;
9) *push button* Triangle;
10) *push button* Square;
11) *push button* Circle;
12) *push button* Cross;
13) *push button* Start;
14) *push button* Select;
15) *analog switch* Left;
16) *analog switch* Right.

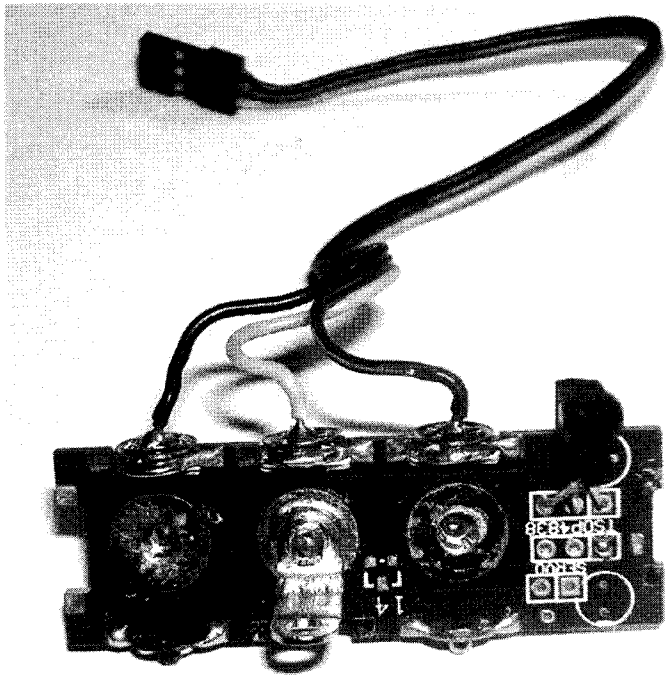


Gambar 1.77 Tombol-tombol *joystick*

Lebih detail mengenai bagaimana mengakses *joystick* ini dapat Anda pelajari pada buku *Membuat Sendiri Robot Humanoid* terbitan Elex Media Komputindo. Di sana akan dibahas protokol-protokol data, *pin out joystick*, dan algoritme program untuk mengakses *joystick*. Untuk pemula proses ini dapat disederhanakan dengan modul elektronik *D-joy controller* yang telah menangani protokol *joystick* dan mengubahnya menjadi serial UART level TTL sehingga dapat dihubungkan langsung ke UART otak robot. Detail mengenai modul ini akan dijelaskan di bagian Modul Elektronik.

10. *Infrared receiver module*

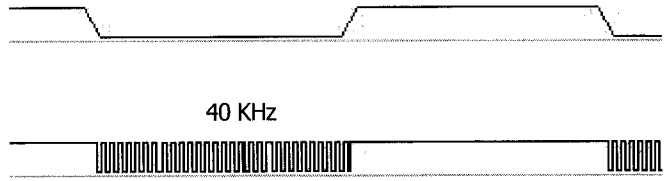
Apabila sakelar, *keypad*, dan *joystick* mengirim perintah operator melalui kabel, *infrared receiver module* akan menerima perintah dari operator melalui *remote control* inframerah dan diteruskan ke otak robot.



Gambar 1.78 *Infrared receiver module* dari *D-logo robotic*

Pada Gambar 1.78 tampak *infrared receiver module* dari *D-logo robotic* yang dilengkapi dengan kabel konektor dan kerangka untuk dipasang ke tubuh robot.

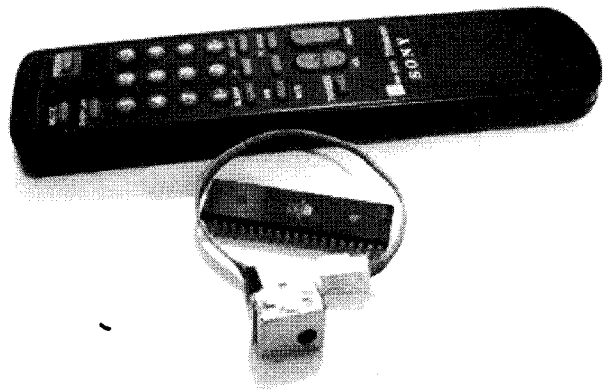
Sinar inframerah yang dipancarkan pada *infrared receiver module* sangat berbeda dengan sinar inframerah untuk keperluan sensor jarak. Sinar inframerah berupa perintah dari operator yang dipancarkan kali ini membawa data yang keakuratannya sangat dibutuhkan. Oleh karena itu, sinar ini dikirimkan dalam bentuk modulasi frekuensi 40 kHz. Data perubahan tegangan dikirim dalam bentuk frekuensi 40 kHz untuk tegangan 0 V dan 0 Hz untuk tegangan 5 V.



Gambar 1.79 Frekuensi 40 kHz

Frekuensi tersebut akan diterima oleh *infrared receiver module* yang memiliki rangkaian filter dan mengubahnya kembali menjadi tegangan 0 V untuk frekuensi 40 kHz dan 5 V untuk frekuensi 0 Hz. Dengan adanya modulasi ini, apabila pada pengiriman terdapat gangguan, filter dari *infrared receiver module* akan mengabaikannya.

Pengaksesan data perintah *infrared receiver module* hanya dapat dilakukan dengan IC mikrokontroler. Pemula yang belum menguasai mikrokontroler dapat menggunakan paket *delta IR control kit* yang terdiri atas *infrared receiver module*, *remote control*, dan IC mikrokontroler AT89S51 yang telah terisi program penerima data inframerah dan konversi menjadi perintah untuk menggerakkan robot maju, mundur, berputar ke kiri, berputar ke kanan, dan berhenti, di mana perintah ini dapat dihubungkan langsung ke rangkaian pengendali motor. Sementara itu, bagi yang ingin mempelajari program mikrokontroler *remote control*, lihatlah pada fail IR.ASM yang ada di dalam CD.



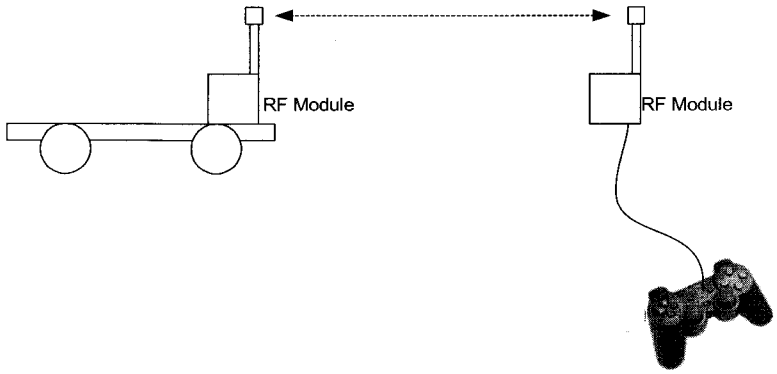
Gambar 1.80 Program mikrokontroler *remote control*

11. *RF module*

RF module merupakan modul komunikasi data menggunakan radio frekuensi. Dibandingkan dengan inframerah, media ini jauh lebih fleksibel karena jaraknya cukup jauh dan dapat menembus halangan. Bila pada inframerah data yang dikirim dimodulasi dengan frekuensi 40 kHz, pada media radio frekuensi data dikirimkan dengan frekuensi yang jauh lebih tinggi. Frekuensi 433 MHz dan 2,4 GHz adalah frekuensi yang biasa digunakan untuk aplikasi *remote control* dan merupakan golongan *ultra high frequency* (UHF).

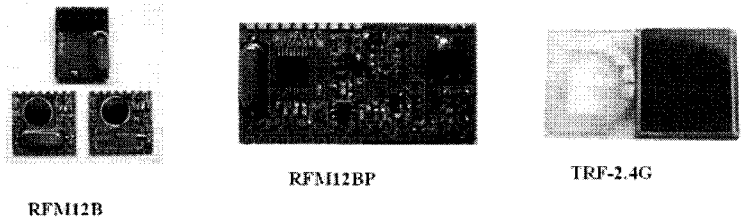
Dibandingkan inframerah, proses pengiriman data menggunakan radio frekuensi lebih kompleks karena, walaupun sedikit, pergeseran frekuensi akan membuat proses transmisi tidak maksimal. Agar diperoleh frekuensi yang tepat tanpa terjadi pergeseran biasanya digunakan rangkaian *phase lock loop* (PLL) atau pengunci fase.

Untuk mempermudah aplikasi radio frekuensi, *RF module* menjadi pilihan terbaik karena modul ini sudah memiliki rangkaian PLL. Dengan *RF module* proses pengiriman dan penerimaan data dapat dilakukan secara digital sehingga modul ini dapat dihubungkan langsung ke mikrokontroler. Dengan bantuan IC mikrokontroler, perintah dari *keypad*, sakelar, atau *joystick* akan dikirimkan ke *RF module* untuk diterima oleh *RF module* pada otak robot. *RF module* bersifat bolak-balik. Oleh karena itu, proses transmisi juga dapat berlaku sebaliknya, yaitu dari robot menuju ke operator seperti yang terlihat pada Gambar 1.81.



Gambar 1.81 Proses transmisi

Untuk frekuensi 433 MHz dapat digunakan modul RFM12B dan untuk frekuensi 2,4 GHz dapat digunakan TRF-2.4G. Pada frekuensi 433 MHz juga terdapat *RF module* khusus untuk daya yang lebih besar, yaitu RFM12BP.



Gambar 1.82 Contoh modul

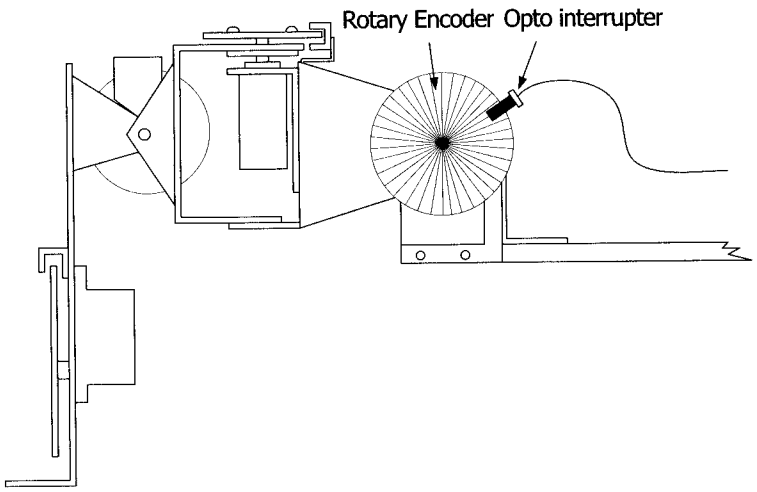
Jika belum menguasai mikrokontroler, para pemula dapat menggunakan modul elektronik *D-joy controller* yang dapat menghubungkan *joystick* dengan *RF module*.

12. *Rotary encoder*

Rotary encoder berupa piringan dengan lubang-lubang sebagai celah di sekelilingnya. Pada aplikasinya, cahaya inframerah akan ditembakkan pada piringan tersebut dan fototransistor akan mendeteksi celah-celah tersebut dengan membedakan kondisi terhalang dan tidak terhalang cahaya.

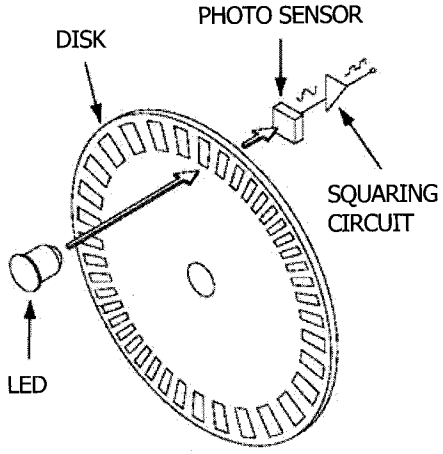
Sensor ini berfungsi untuk mengetahui sudut putar sebuah motor ataupun menghitung kecepatan motor. Motor servo yang pengaturan sudutnya ditentukan oleh lebar pulsa PWM sekalipun sering kali membutuhkan sensor ini. Hal ini karena pengaturan lebar pulsa pada motor servo tidak selalu menunjukkan sudut tertentu. Sumber tegangan motor atau baterai yang menurun pun bisa menjadi penyebab perubahan sudut, walaupun pulsa yang dibangkitkan masih memiliki lebar yang sama.

Beban motor yang membutuhkan torsi lebih besar pun juga dapat menjadi faktor penyebab perubahan sudut tersebut. Dengan adanya *rotary encoder*, otak robot dapat mengetahui apakah gerakan motor telah mencapai sudut yang diharapkan sehingga bila diperlukan, lebar pulsa akan diatur kembali agar sudut tersebut diperoleh.

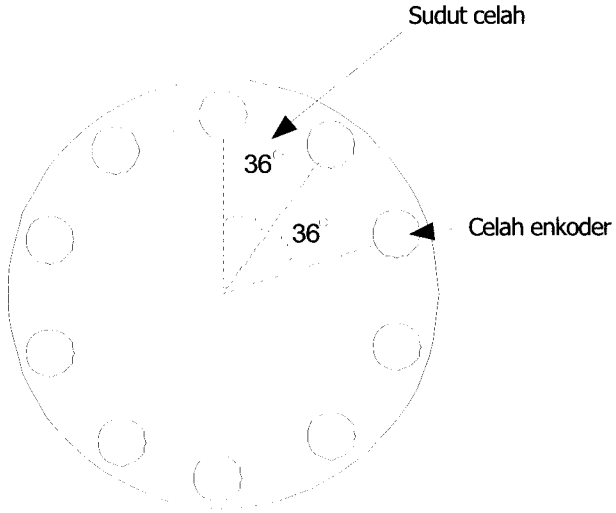


Gambar 1.83 Kaki *delta robo spider* dan *rotary encoder*

Sebagai contoh, pada kaki robot laba-laba, saat otak robot menghitung bahwa gerakan kaki masih belum mencapai sudut yang ditentukan di mana hal ini bisa disebabkan oleh turunnya tegangan baterai atau perubahan medan yang mengakibatkan beban motor berubah, otak robot akan mengatur kembali lebar pulsa agar memperoleh sudut yang tepat.



Gambar 1.84 *Rotary encoder*



Gambar 1.85 Diagram skematik *delta enkoder kit*

LED inframerah dan fotosensor atau fototransistor sering kali dikemas dalam satu paket, yaitu *opto interrupter* seperti yang telah dijelaskan pada subbab sebelumnya. Gambar 1.85 menunjukkan diagram skematik dari *delta encoder kit* yang telah dilengkapi dengan piringan enkoder.

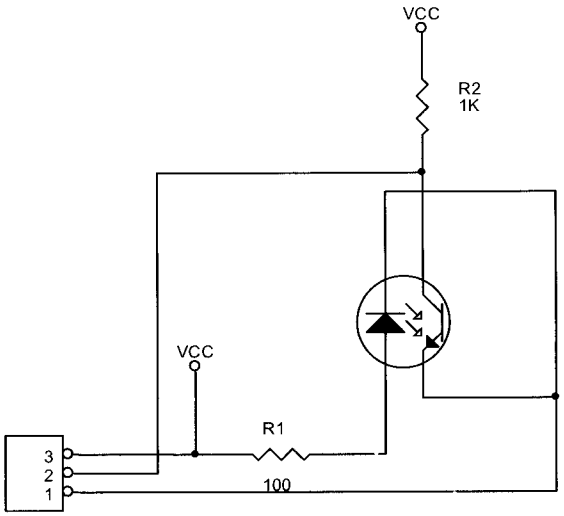
Gambar LED inframerah dan fototransistor tersebut merupakan *photo interrupter* yang memiliki celah di mana piringan enkoder masuk dan berputar di antaranya. Pada saat lubang piringan tidak melalui celah tersebut, cahaya terhalang dan fototransistor berada dalam kondisi terbuka (*cut off*, baca bagian Transistor). Tegangan pada keluaran rangkaian ini (kaki 1 dari konektor 3 pin) adalah sebesar VCC, namun pada saat lubang piringan berada pada celah, cahaya tidak lagi terhalang dan fototransistor akan berada dalam kondisi saturasi. Arus mengalir dari VCC ke emitor melalui R 1 k yang mengakibatkan keluaran dari rangkaian ini terhubung ke *ground* atau tegangan 0 volt. Putaran dari piringan enkoder akan membuat keluaran rangkaian ini berubah-ubah dari logika 0 ke 1 dan kembali lagi ke-0 dan seterusnya.

Mikrokontroler sebagai otak robot akan menghitung perubahan-perubahan kondisi ini menjadi perhitungan yang akan menentukan posisi gerakan motor. Sebagai contoh, pada motor yang memiliki 10 celah enkoder akan diperoleh perhitungan bahwa setiap celah mewakili sudut sebagai berikut.

$$\text{Sudut celah} = \frac{360}{\text{jumlah celah}}$$

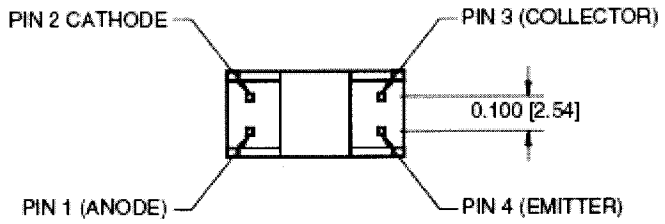
$$= 36^{o}$$

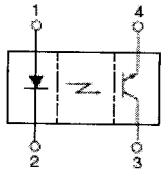
Apabila terdeteksi adanya 3 pulsa, akan diperoleh informasi bahwa motor telah bergerak 36° x 3 = 108°.



Gambar 1.86 Diagram skematik *delta encoder kit*

Opto interrupter yang digunakan adalah tipe MOC703CY dengan konfigurasi kaki seperti pada Gambar 1.86.



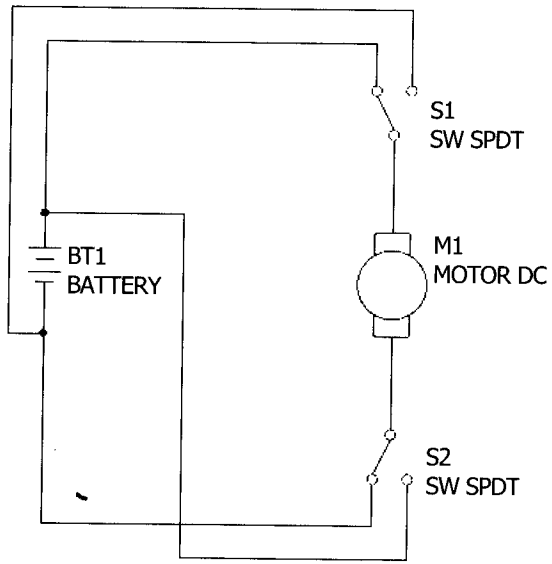


Gambar 1.87 *Opto interrupter* tampak bawah

13. Modul elektronik

Modul elektronik bisa digunakan untuk mempermudah para pemula dalam merancang sistem-sistem elektronika. Modul elektronik adalah rangkaian elektronika yang sudah didesain khusus untuk fungsi-fungsi tertentu, contohnya rangkaian pengendali motor, otak robot, sensor jarak ultrasonik, antarmuka *joystick*, sensor garis, dan lain-lain. Dengan bantuan modul elektronik, aktivitas penyolderan komponen bisa sangat berkurang, bahkan dihilangkan, sehingga perhatian dapat difokuskan untuk mempelajari sistem, algoritme, dan skema modul.

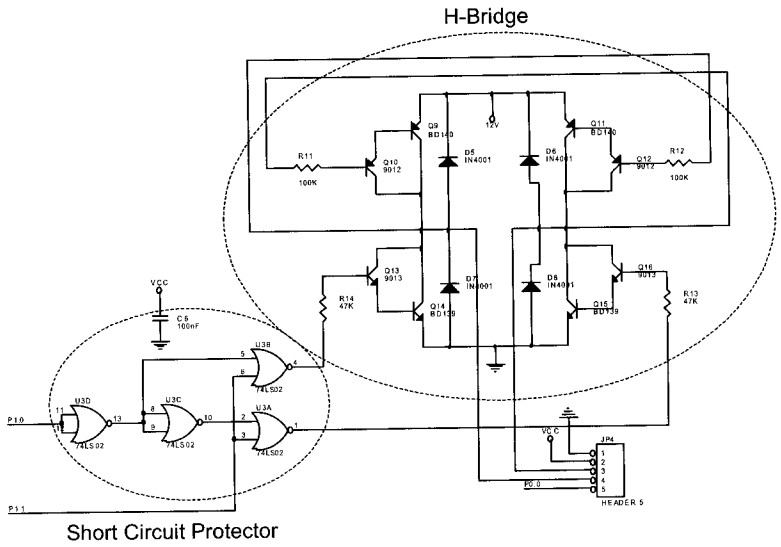
a. *Delta DC driver*



Gambar 1.88 Rangkaian pengatur gerakan motor

Dasar untuk rangkaian pengatur gerakan motor adalah dua buah sakelar DPDT yang akan mengalirkan arus dengan polaritas berbeda. Saat S1 dan S2 berada di posisi kiri, arus akan mengalir dari kutub atas ke kutub bawah motor, sedangkan saat S1 dan S2 berada pada posisi kanan, arus akan mengalir dari kutub bawah ke kutub atas motor sehingga motor bergerak ke arah sebaliknya.

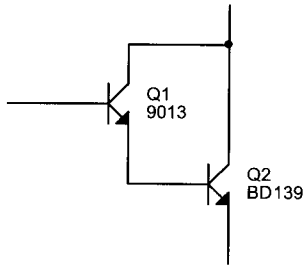
Untuk mewujudkan kejadian ini secara elektronik, kita gunakan sakelar-sakelar elektronik berupa transistor yang didesain dengan model rangkaian *H-bridge*. Seperti yang telah dijelaskan sebelumnya, rangkaian *H-bridge* dapat dibentuk oleh dua pasang rangkaian *push pull* sehingga terbentuk seperti pada Gambar 1.89.



Gambar 1.89 Rangkaian *H-bridge*

Pada gambar tersebut posisi kutub-kutub motor berada pada kaki 3 dan 4 dari JP4. Pada saat transistor Q15 dan Q9 ON, arus akan mengalir dari kaki 4 ke kaki 3 dan sebaliknya, saat Q11 dan Q14 ON, arus akan mengalir dari kaki 3 ke kaki 4. Transistor Q10, Q12, Q13, dan Q16 berfungsi untuk memberikan penguatan tambahan yang membentuk konfigurasi Darlington seperti pada Gambar 1.90, di mana diperoleh formula penguatan sebagai berikut.

$$HFEtotal = HFE_{Q1} \times HFE_{Q2}$$



Gambar 1.90 Konfigurasi Darlington

Diode-diode pada Gambar 1.89 berfungsi sebagai diode proteksi tegangan balik dari motor, sedangkan rangkaian gerbang logika dengan IC 7402 berfungsi sebagai rangkaian penjaga agar tidak terjadi hubung singkat. Rangkaian *H-bridge* memiliki pantangan, yaitu tidak boleh terjadi kondisi ON pada transistor yang posisinya berhadap-hadapan. Kondisi ON boleh terjadi pada transistor yang posisinya saling silang.

Rangkaian IC 74LS02 akan selalu menjaga agar kedua keluarannya berada pada kondisi logika yang berbeda. Bila logika 0 atau tegangan 0 volt ada di U3B, logika 1 atau tegangan 5 volt pasti berada pada U3A dan demikian pula sebaliknya, sehingga tidak mungkin kedua transistor yang saling berhadapan akan ON pada waktu yang bersamaan.

Modul elektronik *delta DC driver* didesain dengan menggunakan dua buah rangkaian *H-bridge* sehingga dapat digunakan untuk mengendalikan dua buah motor DC dengan kapasitas arus sebesar 3A.

Pada Gambar 1.91 tampak *delta DC driver* memiliki koneksi ke bagian otak robot, yaitu *delta robo CPU* melalui konektor *delta robo CPU port* dengan konfigurasi sebagai berkut.

12 V : Input sumber tegangan motor (3-24 volt)

Right encoder : Enkoder motor kanan (hanya untuk motor dengan *rotary encoder*)

Left Encoder : Enkoder motor kiri (hanya untuk motor dengan *rotary encoder*)

Right enable : 0 V = motor kanan aktif, 5 V = motor kanan nonaktif

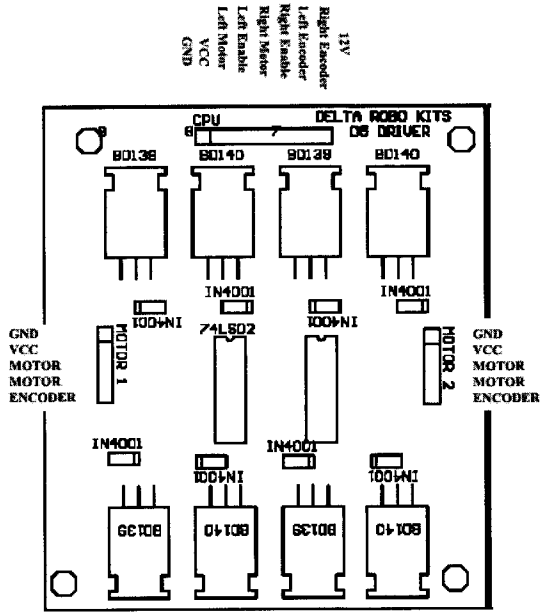
Right motor : 0 V = motor kanan maju, 5 V = motor kanan mundur

Left enable : 0 V = motor kiri aktif, 5 V = motor kiri nonaktif

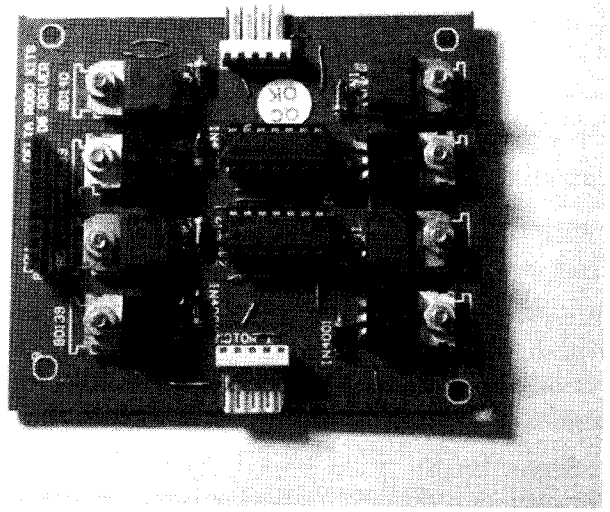
Left motor : 0 V = motor kiri maju, 5 V = motor kiri mundur

VCC : Input sumber tegangan 5 volt

GND : Input sumber tegangan 0 volt



Gambar 1.91 *Delta DC driver*

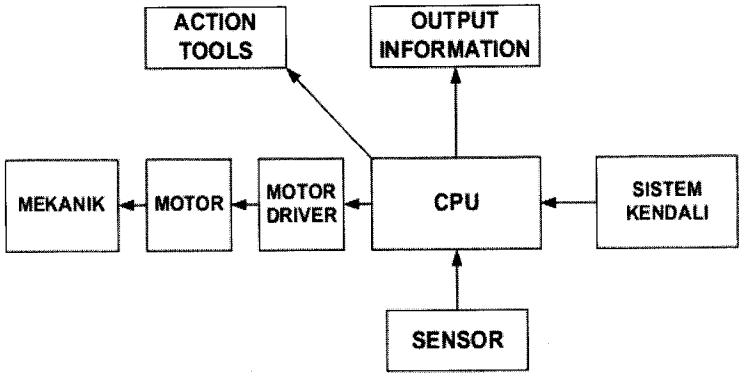


Gambar 1.92 *Delta DC driver*

Bagian *left encoder* dan *right encoder* merupakan perpanjangan dari kaki 5 konektor 5 pin yang terhubung ke motor di mana kaki ini akan terhubung pada bagian keluaran enkoder motor dan langsung diteruskan ke *delta robo CPU.*

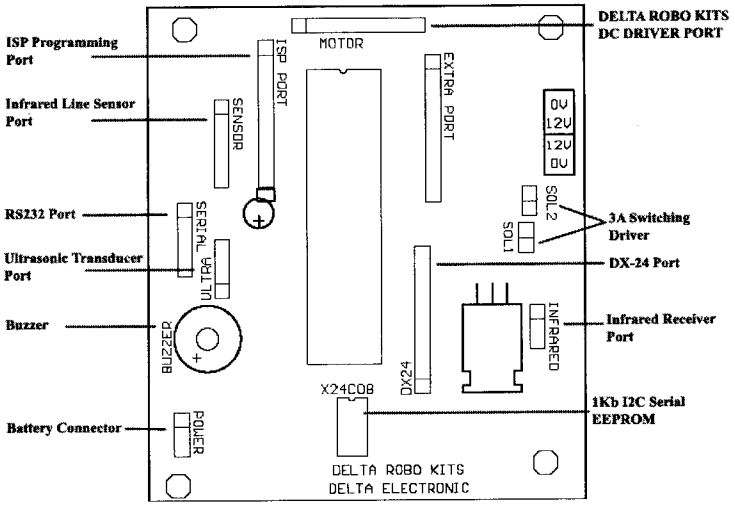
b. *Delta robo CPU*

Delta robo CPU merupakan modul otak dari robot di mana seluruh aktivitas robot akan terpusat dan dilakukan pada modul ini. Input akan diterima dari sensor dan diteruskan sebagai aksi ke *motor driver* atau *action tools* dan *output information*. *Action tools* dapat berupa tangan untuk menjepit atau meriam inframerah untuk permainan robot perang, sedangkan *output information* dapat berupa layar LCD atau LED.



Gambar 1.93 Diagram *delta robo CPU*

Sistem kendali, yaitu *keypad*, *joystick*, atau *remote control*, akan mengirim perintah ke otak robot dan memberikan respons ke *motor driver*, *output information*, atau *action tools*.



Gambar 1.94 *Delta robo kits*

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-labirin agar rute-rute yang telah dilalui tercatat pada memorinya.

Modul *delta robo CPU* merupakan modul yang sesuai untuk aplikasi ini karena modul ini telah memiliki porta *DC driver* untuk mengendalikan motor DC, *infrared line sensor*, dan *ultrasonic transducer* untuk *input sensor*. Porta *infrared receiver* dan porta DX-24 untuk *input remote* dan 1Kb I2C serial EEPROM sebagai memori.

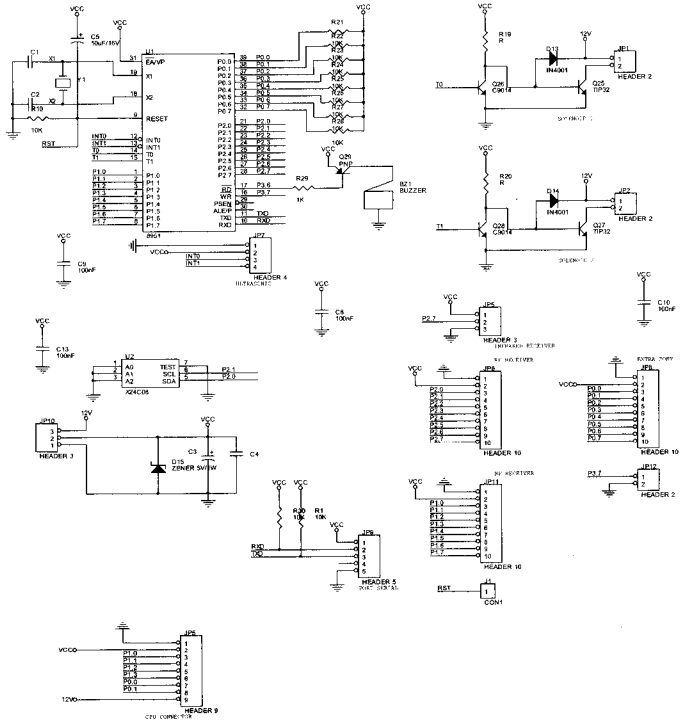
Agar otak robot ini dapat bekerja sesuai kebutuhan, pasanglah IC mikrokontroler yang telah terisi program. Contoh, untuk aplikasi penjejak garis, pasang IC mikrokontroler yang telah terisi fail Linetracer.hex, sedangkan untuk aplikasi *remote control* inframerah, pasang IC mikrokontroler yang telah terisi fail IR.hex.

Selain itu, modul ini juga memiliki porta ISP untuk keperluan mengunduh program ke dalam mikrokontroler *delta robo CPU* di mana pengguna yang sudah mahir dapat memodifikasi program tersebut.

TABEL 1.2 Daftar komponen

Jumlah	Nama Barang	Parameter	Kode
1	*Buzzer*	5 volt	014-0005
2	C1,C2	33 pF SMD0805	026-0030
1	C3	100 uF/16 V	026-0264
5	C4,C8,C9,C10,C13	100 nF	026-0339
1	C5	10 uF SMD 0805	026-0338
2	D13,D14	IN4001	035-0001
1	D15	ZENER 5 V/1 W	035-0009
3	JP1,JP2,JP12	Konektor putih 2 pin	029-0001
2	JP5,JP10	Konektor putih 3 pin	029-0003
1	JP5	*Header* 9 pin	031-0001
3	JP6,JP8,JP11	Konektor putih 10 pin	029-0017
1	JP7	Konektor putih 4 pin	029-0005

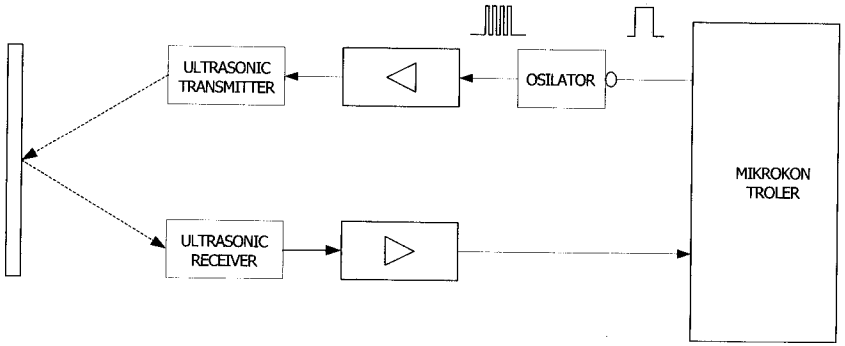
Jumlah	Nama Barang	Parameter	Kode
1	JP9	Konektor putih 5 pin	029-0007
2	Q25,Q27	TIP42	027-0126
2	Q26,Q28	C9014 SOT-23	027-0106
1	Q29	C9012 SOT-23	027-0103
11	R1,R10,R21,R22,R23,R24,	10K SMD 0805	026-0174
	R25,R26,R27,R28,R30		
3	R20,R19, R29	1 k karbon	026-0027
1	U1	AT89S51-24PC DIP	014-0457
1	U2	X24C08 DIP	015-0464
1	Y2	11,0592 MHz *low profile*	029-0032



Gambar 1.95 Skema *delta robo CPU*

c. *D-sonar*

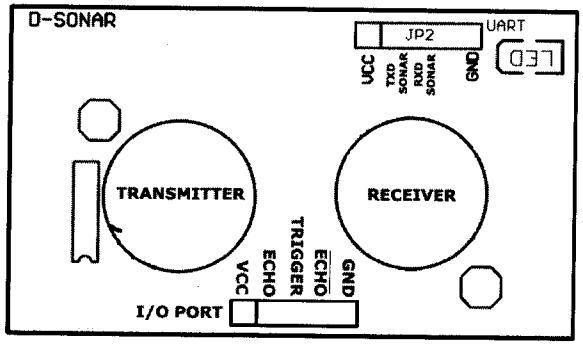
Modul elektronik *D-sonar* berfungsi sebagai sensor pengukur jarak menggunakan metode ultrasonik. Dalam hal ini, seperti yang telah dijelaskan sebelumnya, proses pengukuran dilakukan dengan menghitung waktu yang dimulai saat pertama kali sinyal ultrasonik dipancarkan hingga saat sinyal diterima kembali.



Gambar 1.96 Blok diagram modul *D-sonar*

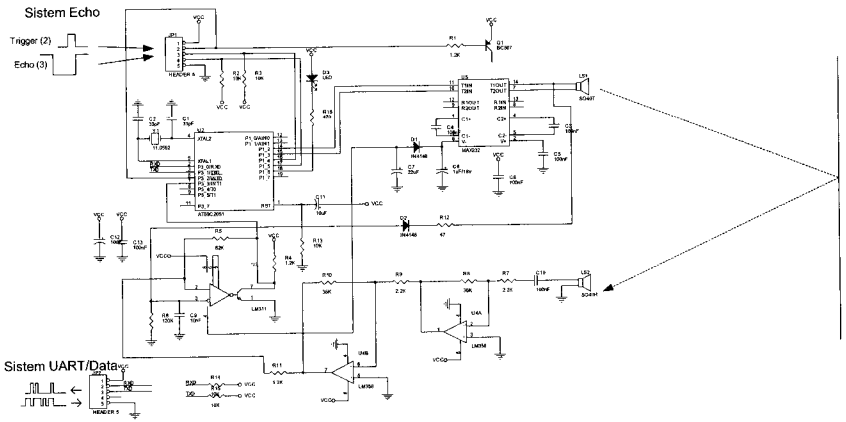
Modul *D-sonar* memiliki rangkaian penguat *ultrasonic transmitter* dan rangkaian penguat *ultrasonic receiver*. Selain itu, di dalam modul *D-sonar* terdapat IC mikrokontroler yang menghitung interval dari sinyal ultrasonik yang dipantulkan ke objek sehingga mikrokontroler yang menjadi otak robot tidak perlu lagi menghitungnya.

Otak robot cukup memerintahkan *D-sonar* untuk menghitung jarak dan *D-sonar* akan mengirimkan hasilnya setelah jarak diperoleh.



Gambar 1.97 Tata letak komponen *D-sonar*

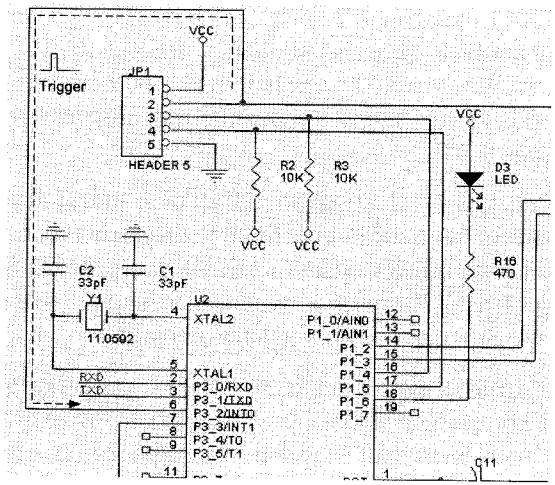
Modul *D-sonar* memiliki dua macam mode dalam proses pengukuran jarak, yaitu mode *echo* di mana informasi jarak akan dikirimkan kembali dalam bentuk lebar pulsa di bagian *echo* dari porta I/O, atau mode UART di mana informasi jarak dikirimkan dalam bentuk paket data.



Gambar 1.98 Sistem *echo*

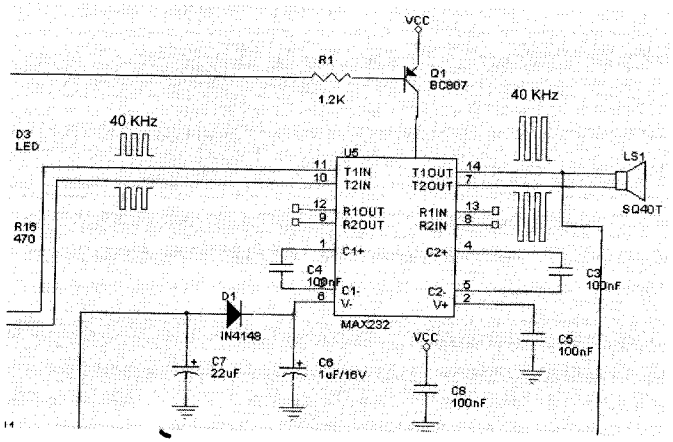
Pada mode ini output *D-sonar* harus dihubungkan pada bagian UART mikrokontroler. Modul ini juga memiliki LED yang akan berkedip saat ditemukan objek yang memantulkan sinyal ultrasonik.

Berikut adalah cara kerja *D-sonar* dengan menggunakan sistem *echo*, sinyal *trigger* positif dimasukkan melalui konektor JP1 kaki nomor 3 dan diterima oleh P3.2/INT0 dari mikrokontroler AT89C2051. Perhatikan garis putus pada Gambar 1.99.



Gambar 1.99 *Trigger*

Dalam hal ini, mikrokontroler AT89C2051 berfungsi sebagai bagian utama dari sistem dan akan memulai proses penghitungan saat *trigger* positif masuk di P3.2/INT0. AT89C2051 akan memancarkan frekuensi 40 kHz yang akan dikuatkan oleh IC MAX232.

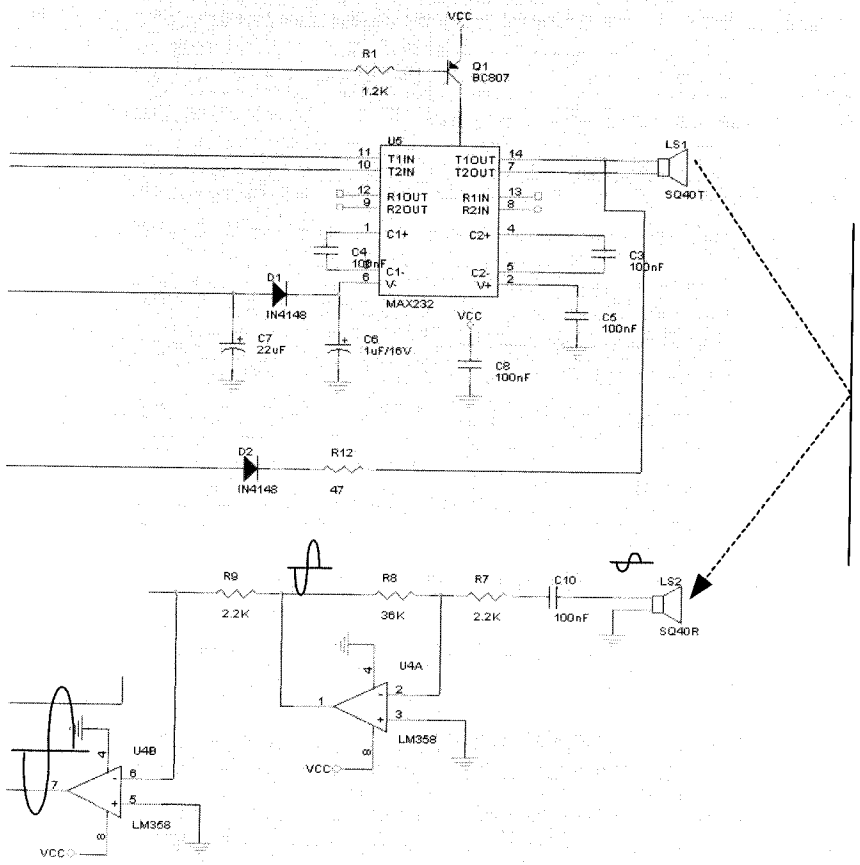


Gambar 1.100 Frekuensi 40 kHz

AT89C2051 membangkitkan dua buah sinyal frekuensi 40 kHz yang berlawanan fase dengan amplitudo sebesar 5 V. Frekuensi ini dikuatkan oleh MAX232 menjadi 10 V. Perbedaan fase dari kedua sinyal 40 kHz akan saling menguatkan dan diumpankan ke *ultrasonic transmitter* SQ40T.

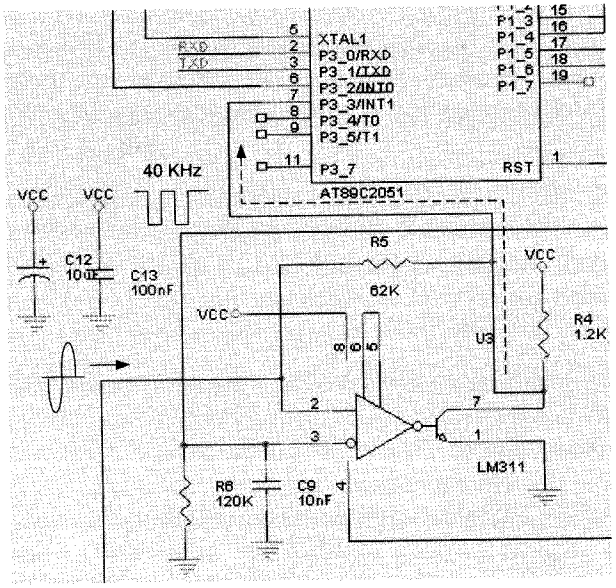
Sinyal ultrasonik akan dipancarkan oleh transmiter SQ40T dan diterima lagi oleh penerima SQ40R. Keluaran dari SQ40R akan dikuatkan dua tingkat oleh IC LM358 yang terdiri atas 2 op amp (Gambar 1.101). Tampak amplitudo dari sinyal yang diterima akan semakin kuat setiap melalui tingkatan penguatan op amp LM358.

Hasil penguatan diteruskan ke IC LM311 yang dalam hal ini diatur sebagai komparator analog yang mengubah sinyal ultrasonik dalam bentuk sinus mencapai sinyal digital yang berupa gelombang otak (Gambar 1.102). Gelombang otak dengan frekuensi 40 kHz hasil dari komparator akan diteruskan ke mikrokontroler AT89C2051 di kaki P3.3/INT1 yang akan mengetahui bahwa pantulan gelombang ultrasonik telah diterima.

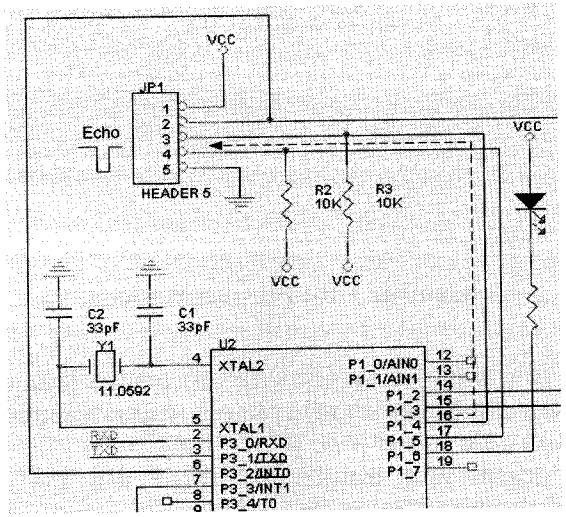


Gambar 1.101 Penguatan op amp

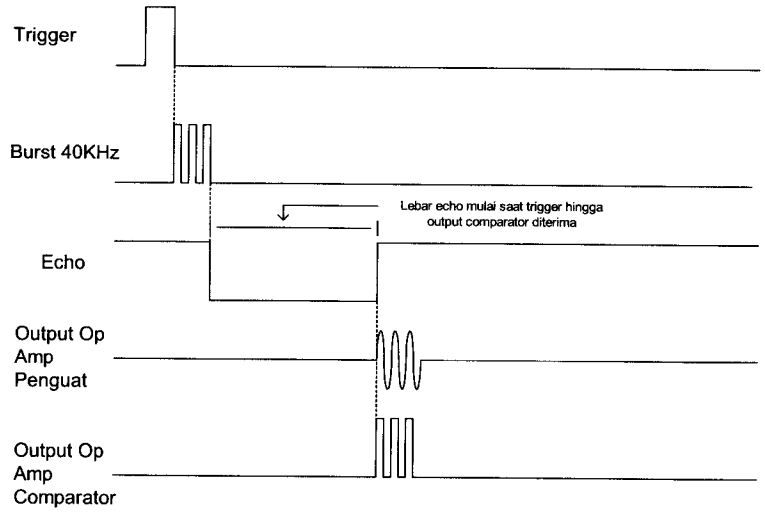
Pada saat itu mikrokontroler akan menghitung lama waktu pantulan dan sekaligus mengubah keluaran *echo* menjadi kondisi logika tinggi sehingga pada bagian *echo* (P1.4 AT89C2051) akan terlihat bentuk pulsa negatif yang diawali saat *trigger* dan diakhiri saat sinyal ultrasonik diterima.



Gambar 1.102 Komparator



Gambar 1.103 *Echo*

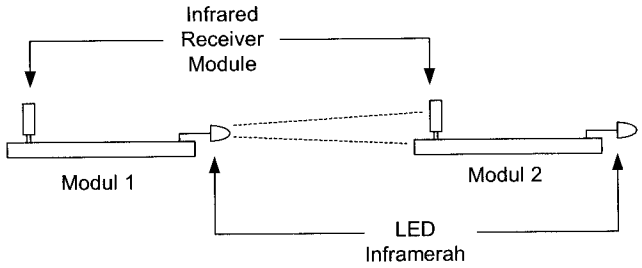


Gambar 1.104 *Timing diagram*

Pada Gambar 1.104 tampak *echo* mulai dibangkitkan ke arah negatif saat *burst* 40 kHz selesai dan diakhiri saat pertama kali output komparator op amp diterima.

d. *Delta infrared fire & target*

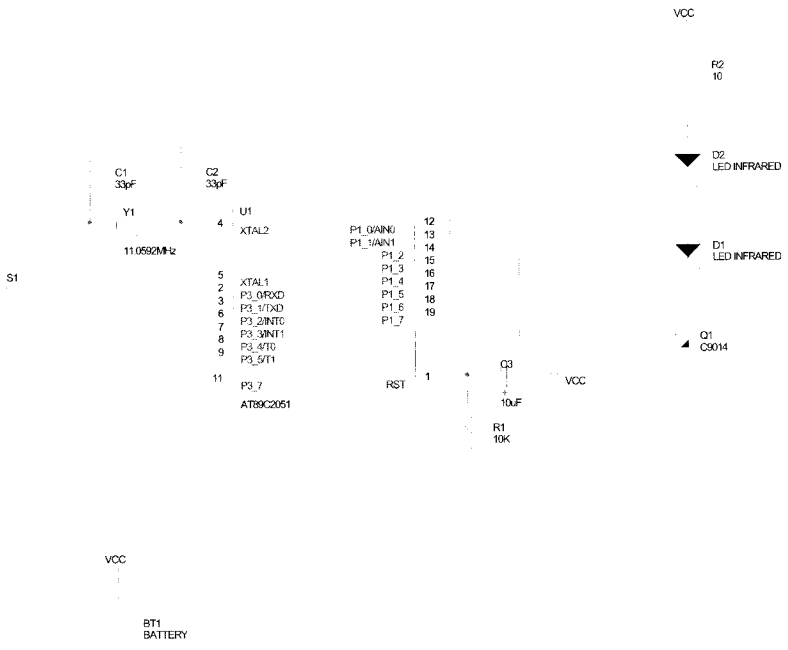
Modul ini berfungsi untuk permainan *robo war* atau perang robot di mana robot dapat menembakkan cahaya inframerah dan akan terdeteksi oleh modul *delta infrared fire & target* yang menjadi sasaran. Modul yang menjadi sasaran akan mengaktifkan *buzzer* sesaat serta memberikan tegangan 5 volt pada keluarannya. Keluaran ini dapat dihubungkan dengan mikrokontroler untuk mendeteksi tembakan atau langsung ke *delta DC driver* bagian *left* dan *right enable* sehingga dapat mematikan rangkaian pengendali tersebut tanpa bantuan mikrokontroler.



Gambar 1.105 *Delta infrared fire & target*

Modul ini terdiri atas dua bagian, yaitu bagian kanon yang membangkitkan sinar inframerah untuk menembak dan bagian sensor yang akan menerima sinar tersebut dan memberikan respons berupa suara *buzzer* dan perubahan tegangan yang dapat dideteksi oleh otak robot.

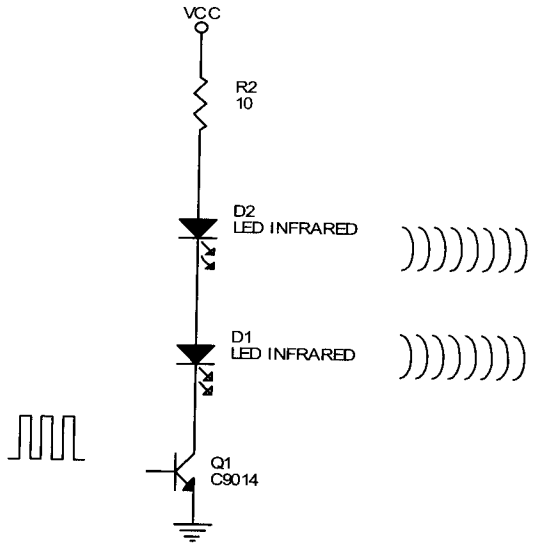
1) Bagian kanon



Gambar 1.106 Skema bagian kanon

Pada bagian kanon, mikrokontroler AT89C2051 bertindak sebagai osilator pembangkit frekuensi pada diode inframerah. Osilasi akan bangkit pada kaki P1.3 saat tombol S1 ditekan. Pada saat tombol S1 tidak ditekan, tegangan pada kaki P3.0/RXD di AT89C2051 adalah berkisar tegangan VCC (sekitar 5-6 volt), namun saat tombol tersebut ditekan, tegangan akan berubah menjadi 0 V. IC AT89C2051 akan terpicu dan osilasi sebesar kurang lebih 40 kHz akan dibangkitkan pada kaki P1.3.

Untuk mengaktifkan LED inframerah dibutuhkan arus yang cukup besar, di mana hal ini tidak dapat diatasi oleh keluaran P1.3 secara langsung. Oleh karena itu ditambahkan sebuah transistor penguat arus sehingga keluaran maksimum AT89C2051 yang hanya 20mA tersebut dapat diperkuat hingga 100mA.

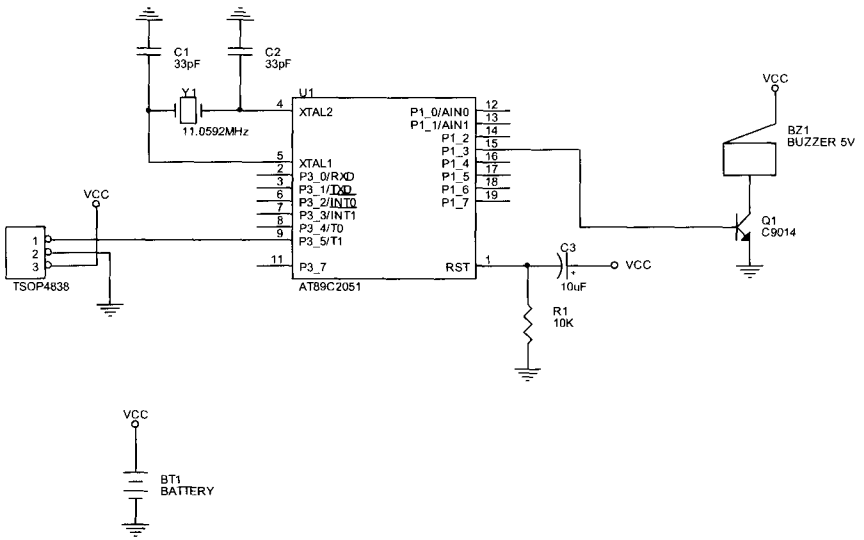


Gambar 1.107 Bagian penguat akhir inframerah

Transistor C9014 adalah sebuah transistor dengan penguatan minimal 50 x (HFE). Oleh karena itu, untuk membangkitkan arus kolektor sebesar 100mA, hanya dibutuhkan arus sebesar 2mA pada basis, sedangkan keluaran P1.3 yang mencapai 20mA sudah lebih dari cukup untuk dihubungkan pada transistor tersebut.

Arus yang mengalir dari kolektor ke emitor akan mengakibatkan diode inframerah aktif dan memancarkan sinar-sinarnya. Hal ini mengakibatkan sinyal-sinyal data yang telah termodulasi akan terpancar dalam bentuk sinyal-sinyal inframerah melalui diode ini.

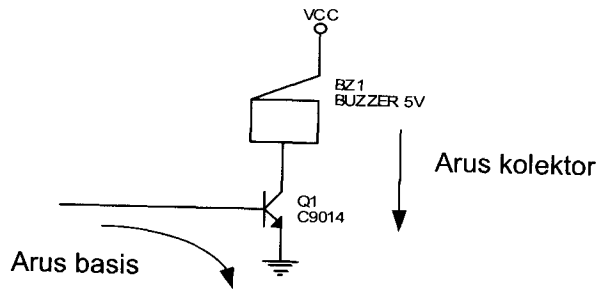
2) Bagian sensor



Gambar 1.108 Skema bagian penerima

Pada bagian ini mikrokontroler AT89C2051 berfungsi untuk mendekode sinyal inframerah yang telah terfilter oleh TSOP4838 di mana frekuensi 40 kHz yang diterima akan difilter menjadi data sebelum diumpankan ke P3.5 dari AT89C2051. Data tersebut didekodekan oleh AT89C2051 dan dicek apakah data inframerah yang diterima memang berasal dari bagian kanon.

Bila data yang diterima sesuai dengan kanon, *buzzer* akan aktif. Hal ini dilakukan dengan mengubah kondisi logika P1.3 pada bagian sensor menjadi logika 1 atau naik ke tegangan 5 V.

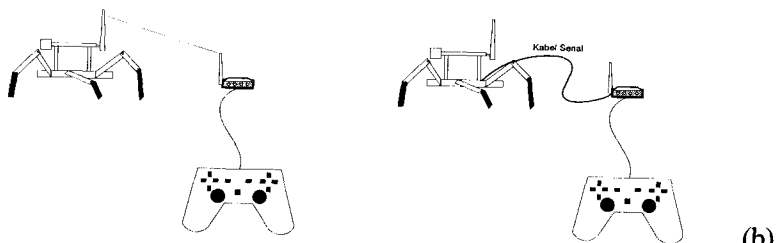


Gambar 1.109 Rangkaian *buzzer*

Arus yang mengalir dari basis menuju emitor akan mengakibatkan transistor ini ON dan arus dari kolektor akan mengalir melalui *buzzer* sehingga *buzzer* berbunyi.

e. *D-joy controller*

D-joy controller merupakan modul elektronik yang menghubungkan otak robot dengan *joystick*, baik melalui kabel (UART) maupun melalui media nirkabel (*wireless*) dengan menggunakan *RF module*.

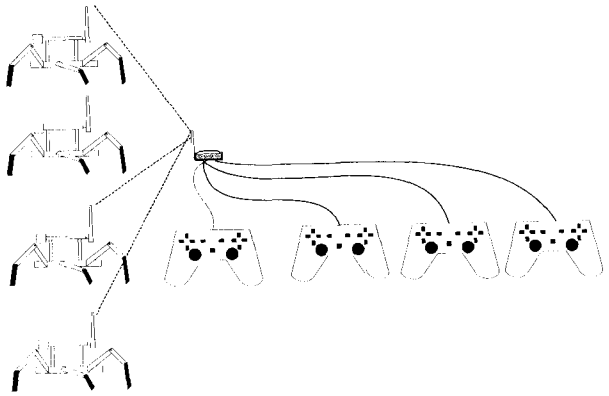


Gambar 1.110 (a) *D-joy controller* mengirim data secara nirkabel dan (b) secara serial

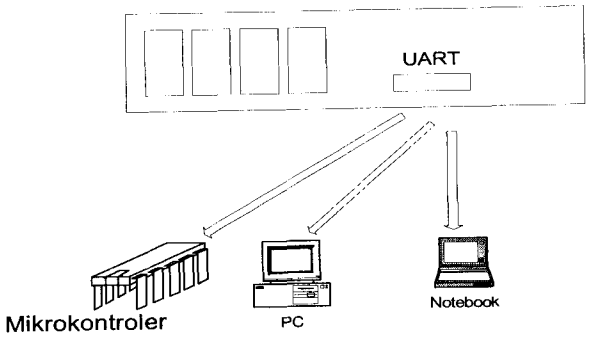
D-joy controller dapat terhubung dengan empat buah *joystick* Playstation sekaligus. Hal ini dibutuhkan untuk aplikasi kendali robot lebih dari satu dan dilakukan secara nirkabel. Proses komunikasi dilakukan secara bergantian, namun dengan kecepatan yang sangat tinggi sehingga terlihat bersamaan. *D-joy controller* akan selalu mengambil data dari keempat *joystick* secara bergantian dan

mengirimkannya ke empat buah *RF module* yang terhubung dengan empat buah otak robot. Dengan teknik ini penggunaan *RF module* akan dapat direduksi sehingga sistem akan menjadi lebih ekonomis. Untuk mengendalikan empat buah robot cukup dibutuhkan 5 buah *RF module*, yaitu 1 di *D-joy controller* dan 4 buah di masing-masing robot.

Selain faktor ekonomis, penggunaan teknik ini juga menghindarkan terjadinya benturan frekuensi. Dalam hal ini, apabila masing-masing *joystick* memiliki *RF module* dan mengirimkan data bersamaan dan mengingat lebar pita frekuensi yang digunakan adalah sama, benturan data akan terjadi, seperti sebuah musyawarah tanpa moderator.

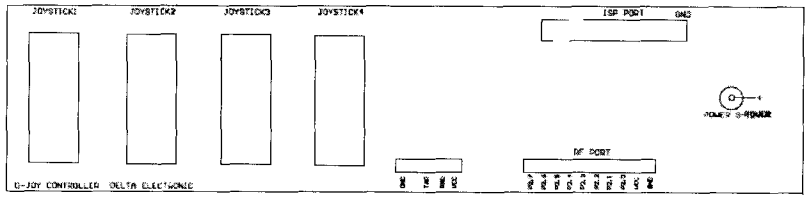


Gambar 1.111 *D-joy controller* terhubung dengan empat buah *joystick*



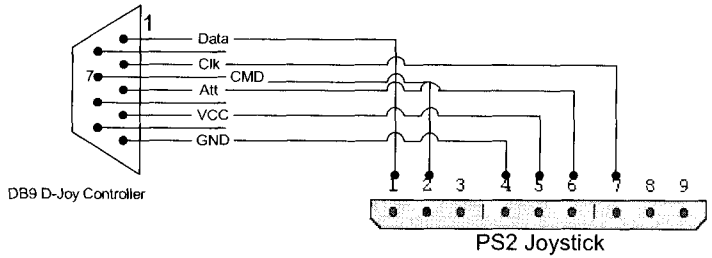
Gambar 1.112 Porta UART

Seperti yang terlihat pada Gambar 1.113, modul *D-joy controller* memiliki empat buah porta DB9 untuk antarmuka dengan *joystick*. Porta UART digunakan untuk menghubungkan *D-joy controller* dengan UART dari otak robot (aplikasi dengan kabel). Porta RF digunakan untuk menghubungkan *D-joy controller* dengan *RF module*. Power 9-12 V merupakan input *power supply* dari *D-joy controller*.



Gambar 1.113 Empat buah porta DB9

Keluaran standar dari *joystick* Playstation tidak menggunakan DB9. Oleh karena itu, perlu dibuat sebuah konverter seperti yang tampak pada Gambar 1.114.



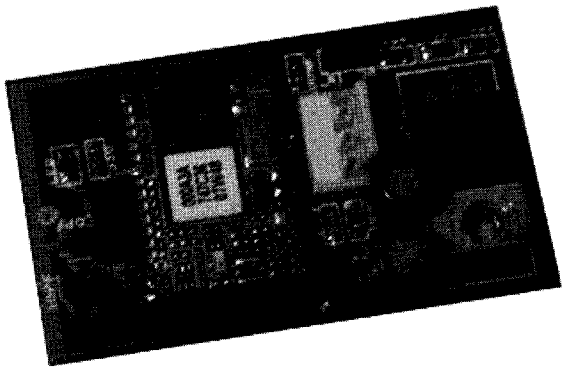
Gambar 1.114 Konverter

Apabila ingin mempelajari pemrograman *D-joy controller* secara lebih mendetail, Anda bisa membaca buku *Merancang Sendiri Robot Humanoid* (Elex Media Komputindo) atau mengunduh manual pengguna *D-joy controller* di www.robotindonesia.com.

f. *Delta Bluetooth module*

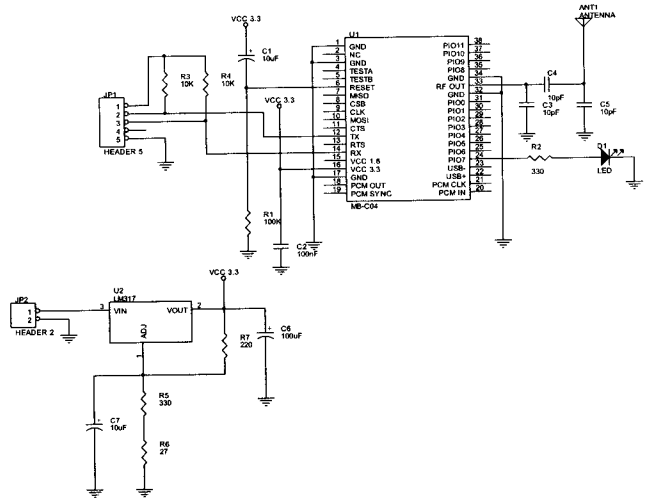
Bluetooth adalah salah satu teknik komunikasi data RF yang cukup praktis. Dengan menggunakan tambahan *Bluetooth to serial*, modul DBM-01 ini sudah dapat terhubung ke PC secara nirkabel.

Modul ini dapat dipasang pada porta UART dari otak robot sehingga robot dapat menerima perintah-perintah yang dikirimkan oleh PC secara nirkabel.



Gambar 1.115 *Delta Bluetooth module*

Dengan menggunakan modul ini, protokol-protokol Bluetooth yang rumit tidak perlu diperhatikan lagi karena proses ini telah diatur oleh DBM-01 yang berfungsi sebagai *Bluetooth to UART*. UART robot akan terdeteksi sebagai COM di PC atau *notebook* melalui media nirkabel.



Gambar 1.116 Skema DBM-01

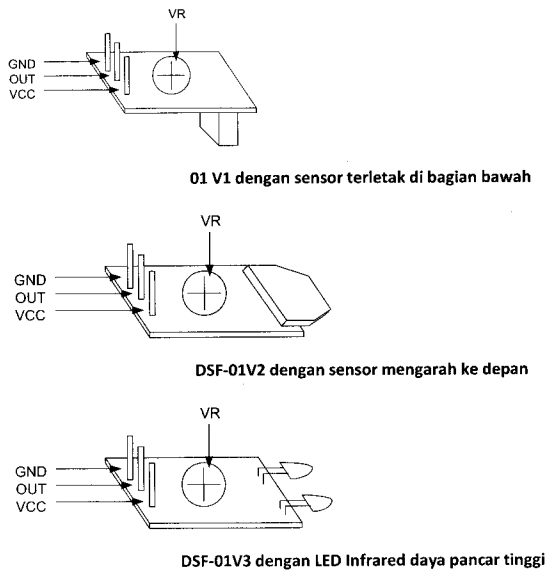
TABEL 1.3 Daftar komponen

Jumlah	*Part*	Parameter	Kode
2	C7,C1	10 uF SMD 0805	026-0338
1	C2	100 nF SMD 0805	026-0339
3	C3,C4,C5	10 pF SMD 0805	
1	C6	100 uF	026-0024
1	D1	LED SMD	021-0045
1	JP1	Konektor putih 5p	029-0007
1	JP2	Konektor putih 2p	029-0001
1	R1	100 k SMD0805	026-0172
2	R5,R2	330 SMD0805	026-0352
2	R3,R4	10 k SMD0805	026-0174
1	R6	27 SMD0805	
1	R7	220 SMD0805	026-0176
1	U1	MB-C04	011-0078
1	U2	LM317 TO-220	015-0465

g. *Delta single line follower*

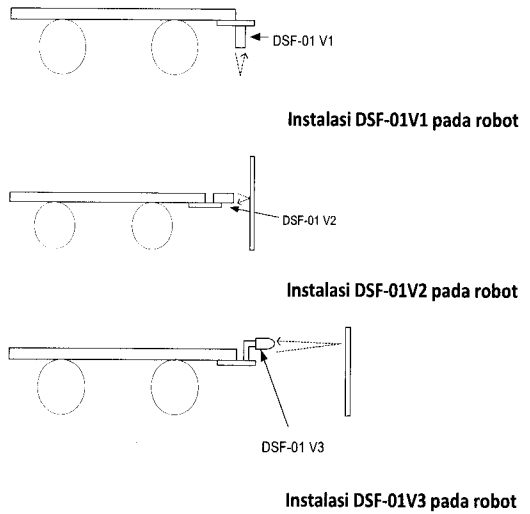
Delta single line follower adalah sebuah modul elektronik yang berfungsi untuk mendeteksi perbedaan antara objek berwarna gelap dan objek berwarna cerah. Untuk objek cerah DSF-01 akan mengeluarkan logika 0 dan LED indikator aktif, sedangkan untuk objek berwarna gelap DSF-01 akan mengeluarkan logika 1 dan LED indikator nonaktif.

Berdasarkan posisinya, ada tiga jenis modul DSF-01, seperti yang tampak pada Gambar 1.117.



Gambar 1.117 Tiga jenis modul DSF-01

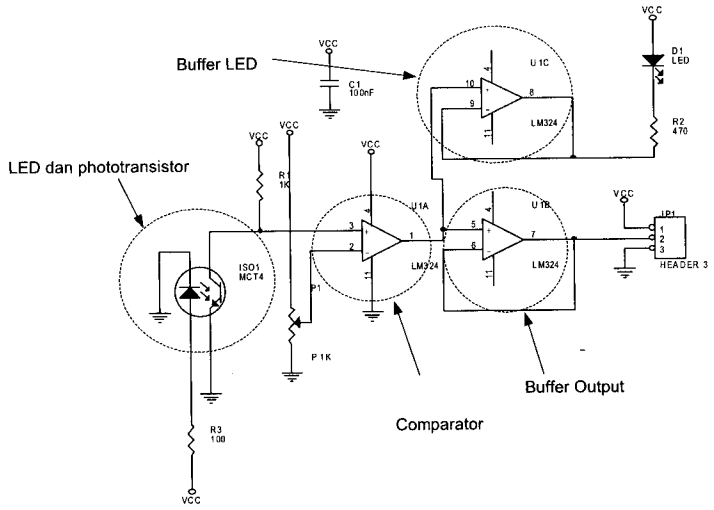
Variabel resistor (VR) yang ada pada modul ini berfungsi untuk mengatur sensitivitas DSF-01. Tampak pada Gambar 1.118, aplikasi dari masing-masing tipe. DSF-01 V1 digunakan untuk mendeteksi garis di bawah robot, sedangkan DSF-01 V2 digunakan untuk mendeteksi garis di depan atau samping robot. DSF-01 V3 digunakan untuk mendeteksi objek yang agak jauh (sekitar 10 cm).



Gambar 1.118 Aplikasi untuk masing-masing tipe

Perubahan terang gelap akan diterima oleh fototransistor, warna terang akan memantulkan cahaya inframerah yang lebih kuat sehingga fototransistor lebih jenuh (*saturated*) dan tegangan keluaran akan semakin rendah. Keluaran dari fototransistor akan dibandingkan dengan tegangan yang ada pada VR. Apabila tegangan VR lebih besar, keluaran dari komparator adalah 0 V, sedangkan bila tegangan VR lebih kecil, keluaran dari komparator adalah 5 V.

Bagian *buffer out* akan menguatkan keluaran komparator ke output mikrokontroler dan *buffer* LED akan menguatkan keluaran komparator dan mengaktifkan LED saat cahaya inframerah yang diterima besar.



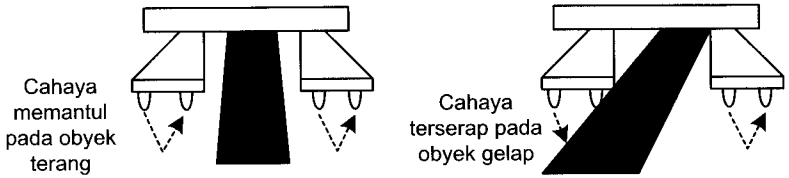
Gambar 1.119 Skema *delta single line follower*

TABEL 1.4 Daftar komponen

Jumlah	*Part*	Parameter	Kode
1	C1	100 nF SMD0805	026-0323
1	D1	LED SMD	021-0045
1	ISO1	LED inframerah TSAL4400	011-0069
		Fototransistor TOPS050TB2	011-0019
		Dapat diganti dengan *opto interrupter*	
1	JP1	LM317	015-0465
1	P1	VR 1 k	026-0226
1	R1	1 k SMD0805	026-0167
1	R2	470 SMD8005	026-0178
1	R3	100 SMD0805	026-0173
1	U1	LM324 SOIC-14	015-0447

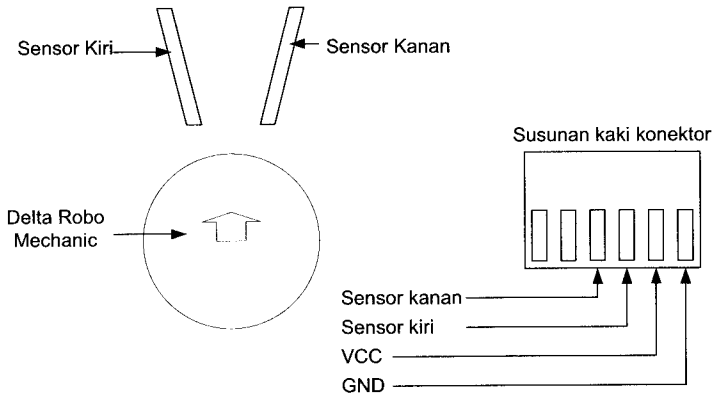
h. *Delta infrared line sensing*
Modul ini berfungsi pada aplikasi penjejak garis (*line tracker/line follower*) sebagai "sungut" robot yang sifatnya fleksibel. Untuk

mengatur cepat-lambatnya respons sungut, terdapat sembilan lubang *spacer* di mana sungut dapat dimajumundurkan dengan mengubah posisi *spacer* pada lubang-lubang tersebut. Semakin jauh sungut dimajukan, semakin cepat respons robot terhadap perubahan garis.



Gambar 1.120 Mendeteksi garis

Selain itu, modul ini juga dapat mengatur lebar garis yang dilewati dengan memutar posisi kedua modul sesuai lebar garis yang dideteksi. Keluaran dari sensor ini sudah berupa kondisi logika dan kompatibel TTL dengan *on board pull up resistor* 10 k. Deteksi sensor diperoleh dengan adanya logika 1 pada *pin out*. Saat cahaya memantul di bidang putih dan mengenai sensor kanan, kaki sensor kanan pada konektor akan berlogika 1 dan sebaliknya, bila cahaya inframerah tidak memantul karena melalui bidang hitam.



Gambar 1.121 Sungut robot

1.3 Bagian Pemrograman Robot

Pada aplikasi robot cerdas, yang membutuhkan bagian otak robot, pemrograman menjadi bagian yang utama. Seperti pada manusia, bagian otak robot ini harus menerima pelajaran-pelajaran terlebih dahulu sebelum dapat digunakan. Pada sebuah robot proses ini disebut proses pemrograman. Secara garis besar, ada dua jenis teknik pemrograman pada robot, yaitu pemrograman sintaksis (*syntax programming*) dan pemrograman visual (*visual programming*).

1. **Pemrograman sintaksis**
   Teknik pemrograman sintaksis dilakukan dengan menggunakan instruksi-instruksi yang disusun membentuk suatu fail yang disebut kode sumber. Agar dapat diprogram ke robot, kode sumber ini terlebih dahulu harus melalui proses kompilasi yang mengubahnya menjadi program objek dalam bentuk HEX.

   Dalam hal ini, bahasa pemrograman yang digunakan adalah bahasa C. Penulisan program mikrokontroler dalam bahasa C dapat ditulis dengan cara apa pun selama aturan penulisan program dalam bahasa C diikuti. Berikut adalah beberapa hal yang perlu diperhatikan tentang bahasa C.

   a. Aturan penulisan bahasa C
      Penulisan program dalam C sama seperti penulisan program lainnya, terdapat program utama dan fungsi. Program utama dapat menjalankan perintah-perintah yang dibuat dalam fungsi. Fungsi ini memiliki nilai balik dan argumen. Contoh penulisan fungsi:

```
<tipe data> <nama_fungsi><tipe data argument>
{
    Ini perintah fungsi;
}
```

      Fungsi ini dapat dipanggil di mana pun di dalam program utama. Fungsi ini dapat mengurangi panjang program karena jika terdapat perintah yang sama dan dikerjakan secara berulang-ulang, perintah-perintah ini dapat dimasukkan dalam satu fungsi. Contoh penggunaan fungsi dalam program utama:

```
unsigned char fungsi_kali(unsigned char x,unsigned char y)
{
    unsigned char i;

    i=x*y;
    return(i);
}

void main(void)
{
    unsigned char hasil;

    hasil=fungsi_kali(5,6);
}
```


Jika program ini dijalankan, variabel hasil akan bernilai 30. Program utama terletak di dalam "main". Setiap kali program dijalankan, maka perintah yang pertama kali dilakukan adalah menjalankan perintah yang terdapat dalam fungsi "fungsi_kali". Jika tidak ada nilai balik dari fungsi, tipe data dan argumen harus diisi dengan "void", yang artinya tidak berlaku. Hal ini harus dilakukan, jika tidak, akan terjadi eror. Contoh penggunaan *void* dalam fungsi dapat dilihat pada program utama "void main(void)". Sedangkan *unsigned char* adalah tipe data nilai balik dan juga argumen, dalam contoh *list* program di atas.

   b. Tipe data
      Tipe data yang digunakan dalam pemograman mikrokontroler berbahasa C tercantum dalam TABEL 1.5.

TABEL 1.5 Tipe data dan batas nilainya

	Tipe Data	Panjang	Batas Nilai
teger	(unsigned) char	8 bits	0 to 255
	signed char		-128 to 127
	unsigned short (int)	16 bits	0 to 65535
	(signed) short (int)		- 32768 to 32767
	unsigned int	16 bits	0 to 65535
	(signed) int		- 32768 to 32767
	unsigned long (int)	32 bits	0 to 4294967295
	(signed) long (int)		- 2147483648 to 2147483647
Real	float	32 bits	9 digit panjang
	double	64 bits	17 digit panjang
	long double	64 bits	17 digit panjang

Ada perbedaan dalam penulisan angka-angka seperti desimal dan heksadesimal, dan juga oktal. Untuk desimal penulisannya tidak berbeda. Untuk heksadesimal penulisannya harus diawali dengan "0x" atau "0X", misalnya penulisan 23 heksadesimal adalah "0x23". Sementara itu, penulisan oktal diawali dengan "0", misalnya 45 oktal adalah 045.

Bilangan desimal, heksadesimal, dan oktal termasuk dalam tipe integer, sedangkan bilangan pecahan harus menggunakan tipe data real. Untuk bilangan pecahan ini penulisannya tidak berbeda. Bilangannya juga dapat ditulis dengan format ekponensial, contohnya 127.78,125.55e2.

c. Variabel

Format untuk pendeklarasian variabel dalam C adalah "<tipe data> <nama variable>". Contoh pendeklarasian variabel a dengan tipe *unsigned char*: "unsigned char a". Tipe data variabel dapat dilihat pada TABEL 1.5. Variabel ini juga dapat ditentukan nilai awalnya. Contoh, deklarasi variabel dengan nilai awal yang ditentukan "unsigned char a=3", maka variabel "a" mempunyai nilai awal 3. Pendeklarasian 2 nama variabel dengan tipe data yang sama dapat dilakukan dengan cara "<tipe data> <variabel1>,<variabel2>. Berikut adalah contoh pendeklarasian variabel.

```
void main ( void )
{
        unsigned char a;
        int i;
        unsigned int k=500,j=250;
        float n=1.25;
}
```

d. Operator

Berikut adalah penjelasan mengenai operator-operator yang digunakan dalam C.

TABEL 1.6 Daftar operator C

	Tipe Data	Panjang	Batas Nilai
Integer	(unsigned) char	8 bits	0 to 255
	signed char		-128 to 127
	unsigned short (int)	16 bits	0 to 65535
	(signed) short (int)		- 32768 to 32767
	unsigned int	16 bits	0 to 65535
	(signed) int		- 32768 to 32767
	unsigned long (int)	32 bits	0 to 4294967295
	(signed) long (int)		- 2147483648 to 2147483647
Real	float	32 bits	9 digit panjang
	double	64 bits	17 digit panjang
	long double	64 bits	17 digit panjang

Penjumlahan, pengurangan 1	++ – – –
Operator aritmetika biner	+ – * / %
Operator geser kiri, kanan	<< >>
Operator untuk operasi bit	& | ^ ~
Operator relasi	> < >= <= == !=
Operator logika	&& || !
Operator pemberian nilai	= += -= *= /= %= <<= >>= |= &= ^=

Operator penjumlahan 1 pada dasarnya sama dengan penjumlahan dan pengurangan pada umumnya, yang membedakannya adalah penjumlahan dan pengurangannya hanya dengan angka 1. Untuk lebih jelasnya, lihat contoh berikut.

```
void main ( void )
{
    char a;
    a++;
    a--;
}
```


Setiap kali program menjalankan perintah "a++", maka nilai "a" akan bertambah satu dan demikian juga sebaliknya.

Operator aritmetika biner digunakan untuk fungsi aritmetika pada umumnya, yaitu pengurangan, penjumlahan, dan pembagian. Untuk lebih jelasnya, lihat contoh berikut.

```
void main ( void )
{
    char a;
    char b;
    a=a+b;
    b=b-a;
    b=a*b;
    a=a/b;
    b=a%b;
}
```

Pada *list* program di atas "a=a+b" akan menjumlahkan nilai variabel a dengan b, kemudian hasilnya akan disimpan di variabel a, pada baris berikutnya untuk pengurangan, lalu untuk perkalian, setelah itu untuk pembagian, sedangkan "b=a%b" adalah untuk menghitung sisa pembagian dari "a/b".

Operator geser kiri atau geser kanan berfungsi untuk menggeser bit pada sebuah variabel ke kiri atau ke kanan. Untuk lebih jelasnya, lihat contoh berikut.

```
void main ( void )
```

```
{
    char a;
    a=0x05;
    a<<=2;
}
```

Pada saat awal nilai a=0x05, dan jika dijadikan biner 8 bit, menjadi "0000 0101". Kemudian, dengan perintah berikutnya variabel a digeser ke kiri sebanyak 2, dan a menjadi "0001 0100", lalu nilai a menjadi 0 x14 heksadesimal.

Operator bit berguna untuk operasi antarbit, operator "|" berguna untuk "OR", operator "&" untuk "AND", "^" untuk "XOR", dan "~" untuk "not". Operator relasi berfungsi untuk relasi antara 2 variabel, contoh operator ">" untuk lebih besar, "<" untuk lebih kecil, dan "!=" untuk tidak sama. Operator logika adalah untuk operasi logika sebuah variabel, kebalikan dari operator bit. Operator "&&" adalah untuk logika "AND", operator "||" untuk logika "OR", dan operator "!" untuk "NOT".

Operator pemberian nilai tidak berbeda dengan operator lainnya, yang membedakan hanyalah hasilnya langsung diberikan ke variabel tersebut, misalnya "a+=5", artinya variabel a akan ditambahkan dengan 5. Demikian juga untuk operator lainnya.

e. Perintah-perintah dalam C
   Proses untuk program percabangan dalam C dinyatakan dalam pernyataan "if-else", "else-if", dan "switch-case".

   1) Pernyataan "if-else", pernyataan ini akan menjalankan perintah pada bagian berikutnya jika kondisinya benar, dan akan menjalankan perintah yang terdapat pada bagian *else* jika pernyataannya salah.

   2) Pernyataan "else-if", pada dasarnya pernyataan ini sama dengan "if-else", yang membedakan hanyalah pernyataan kedua ini membutuhkan lebih banyak percabangan "if-else".

   3) Pernyataan "switch-case" akan menyebabkan alur program untuk bercabang ke salah satu proses dari banyak proses tergantung pada hasil pernyataan yang diberikan karena hasil dari pernyataan ini ditentukan dengan hasil konstanta. Kita tidak dapat menggunakan operator relasional untuk pernyataan ini.

Untuk lebih jelasnya mengenai perintah-perintah di atas, lihat contoh penggunaan "if-else" berikut.

```
void hitung(void);
unsigned int detik = 0 ;
unsigned int menit = 0 ;
void hitung(void)
{
    if(detik >= 59 ){
    detik = 0 ;
    menit ++ ;
}
else
{
    detik ++ ;
}
```

Program di atas akan menghitung detik. Pada fungsi hitung di atas "jika detik>=59", detik=0, dan menit ditambah 1, sedangkan jika detik masih lebih kecil dari 59 (*else*), detiknya saja yang ditambah 1.

```
void hitung(void);
unsigned int detik = 0 ;
unsigned int menit = 0 ;
void hitung(void)
{
    if(detik >= 59 )
    {
        detik = 0 ;
```

```
        menit ++ ;
    }
    else if(detik>=59)&&(menit>=59)
    {
        detik = 0 ;
        menit = 0;
    }
}
```

Terdapat kesamaan dengan program sebelumnya, yang membedakan hanyalah program ini digunakan untuk pernyataan "else-if". Perintah-perintah pada blok "else-if" akan dikerjakan ketika pernyataan pertama salah, tetapi sebelum perintah pada blok "else-if" dikerjakan, pernyataan pada "else-if" dicek terlebih dahulu.

```
void select(void)
{
    switch(sw)
    {
        case 0 : ans = a + b ;
        break ;
        case 1 : ans = a - b ;
        break ;
        case 2 : ans = a*b ;
        break ;
        case 3 : ans = a / b ;
        break ;
        default : error();
        break ;
    }
}
```


Pada *list* program di atas nilai ans bergantung pada nilai sw, misalnya jika nilai sw=1, nilai ans adalah hasil pengurangan a-b. Nilai sw ini hanya dari 0-3 saja, jika nilai sw 4 atau selain dari 0-3, fungsi eror akan dijalankan.

Perintah untuk kontrol pada proses pengulangan di C adalah "while", "for", dan "do-while".

1) "while", proses pada blok *while* ini akan dikerjakan terus-menerus selama pernyataan *while* terpenuhi.
2) "for", proses pada blok *for* akan dikerjakan selama nilai suatu variabel belum terpenuhi, dalam hal ini variabel pengondisi, contohnya "for(<nilai awal>;<nilai akhir><variable>). Untuk lebih jelasnya, lihat contoh penggunaan *for* berikut.

```
void sum(void) ;
unsigned int total = 0 ;
void sum(void)
{
    unsigned int i = 1 ;

    while(i <= 100)
    {
        total += i ;
        i ++ ;
    }
}
```

Pada contoh program di atas perintah di dalam blok *while* akan terus dijalankan sampai nilai i lebih besar dari 100. Contoh program di atas akan menghitung total nilai i dari 1-100. Contoh program berikut juga akan menghitung total nilai i dari 1 sampai 100, tetapi dengan pernyataan for.

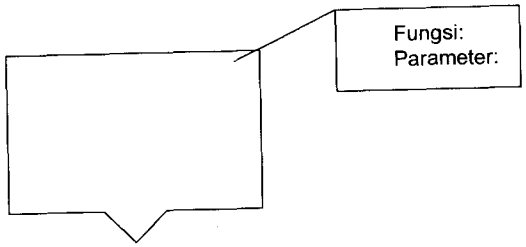
```
void sum(void) ;
```

```
unsigned int total = 0 ;
void sum(void)
{
    unsigned int i ;
    for(i = 1 ; i <= 100 ; i++)
    {
        total += i ;
    }
}
```

Fungsi contoh program di atas sama dengan contoh program dengan *while*, tetapi pernyataannya diganti dengan *for*. Nilai i akan terus dijumlah sampai i>100.

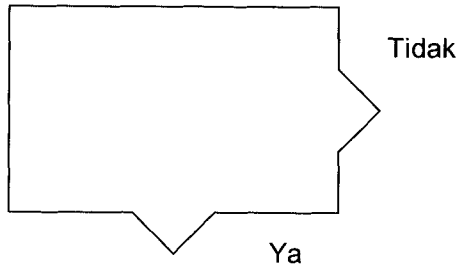
2. Pemrograman visual

Teknik pemrograman ini jauh lebih mudah dibandingkan dengan pemrograman sintaksis, terutama untuk para pemula. Dengan pemrograman visual pemahaman akan dapat dilakukan dengan mudah. Contohnya, untuk proses menggerakkan motor, terlebih dahulu parameter kecepatan diatur untuk menentukan seberapa cepat motor bergerak sebelum proses dijalankan.



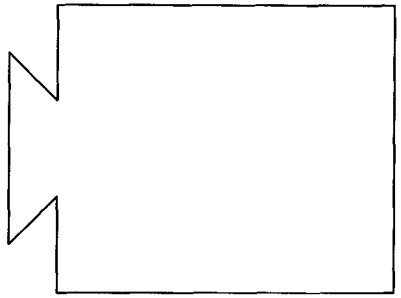
Gambar 1.122 Mengatur parameter kecepatan

Bentuk *decision* merupakan bagian di mana program akan melakukan percabangan berdasarkan kondisi yang diperoleh. Pada bagian ini akan ditentukan ke mana program akan berjalan bila kondisi ya atau tidak terpenuhi.



Gambar 1.123 Kondisi ya dan tidak

Tampak pada gambar terdapat kondisi ya dan tidak dalam sebuah *decision*. Kondisi *tidak* mengarah ke samping dan kondisi *ya* mengarah ke bawah. Bentuk *jump* merupakan bagian di mana program akan melompat ke suatu label yang dituju.

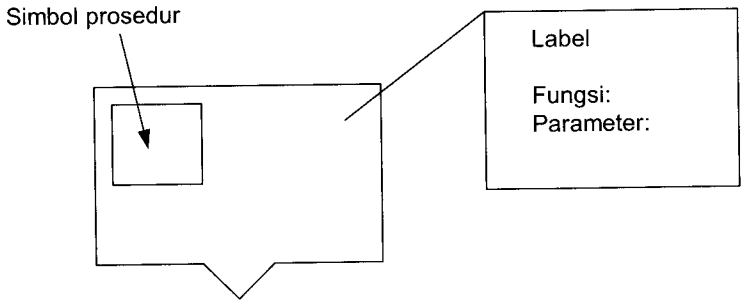


Gambar 1.124 Bentuk *jump*

Bentuk ini dibutuhkan untuk proses lompatan ke suatu label tertentu dalam program yang sedang dirancang.

Pemrograman visual tersebut dapat dilakukan dengan menggunakan Delta Robo Studio. Dengan perangkat lunak tersebut bentuk-bentuk visual berupa *command*, *decision*, dan *jump* akan dirakit menjadi bentuk hex sehingga dapat diunduh ke otak robot. Pada Delta Robo Studio semua bentuk visual ini akan dilengkapi dengan simbol dan deskripsi yang jelas

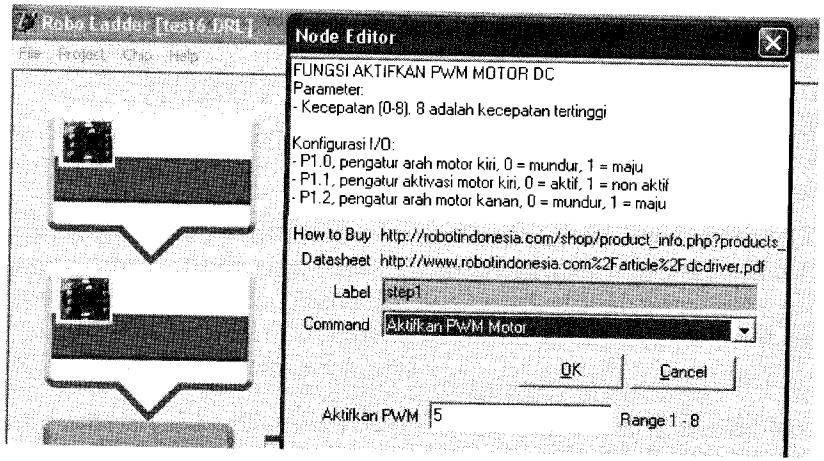
sehingga pemrogram dapat melihat fungsi dan penjelasan setiap blok visual yang akan digunakan.



Gambar 1.125 Simbol prosedur

Seperti pada Gambar 1.125 di mana pada sebuah *command* atau *decision* dapat diberikan sebuah simbol prosedur yang dikerjakan oleh blok visual tersebut. Contohnya, untuk prosedur menggerakkan motor, simbol prosedur dapat berupa gambar motor yang digunakan oleh prosedur tersebut. Dengan mendekatkan *mouse*, pengguna dapat melihat label, fungsi, dan parameter-parameter dari blok visual ini. Label berfungsi sebagai tanda dari lokasi blok visual di mana *decision* atau *jump* dapat diarahkan ke label tersebut.

Pemrogram dapat melihat deskripsi yang lebih detail pada setiap *command* atau *decision* dengan melakukan klik kanan pada *mouse* dan memilih Edit sehingga tampak seperti Gambar 1.126.



Gambar 1.126 Robo Ladder

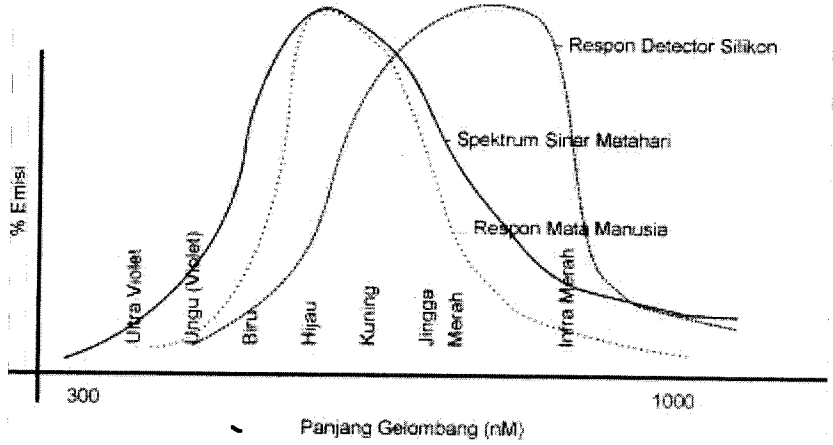
Pemrogram dapat melihat penjelasan mengenai *command* ini sekaligus mengubah nilai-nilai parameternya.

Setelah program yang dirancang selesai, proses kompilasi dapat dilakukan dengan terlebih dahulu memilih mikrokontroler yang digunakan pada robot. Untuk Delta Robo Studio, mikrokontroler yang dapat didukung adalah AT89S51 dan AT89S52. Kemudian, proses kompilasi dilakukan dengan menekan tombol F9 atau *assembly* sehingga fail drl yang dibentuk oleh Delta Robo Studio diubah menjadi bentuk hex yang siap untuk diunduh ke mikrokontroler robot.

BAB II
ROBOT PENJEJAK GARIS SEDERHANA

Robot penjejak garis adalah robot yang bergerak secara otomatis mengikuti lika-liku garis di permukaan lantai. Untuk mengenali pola garis, digunakan sensor cahaya yang akan mendeteksi terang gelap permukaan yang ada di bawahnya. Untuk itu, harus diperoleh perbedaan warna yang mencolok antara warna garis dan warna permukaan lantai.

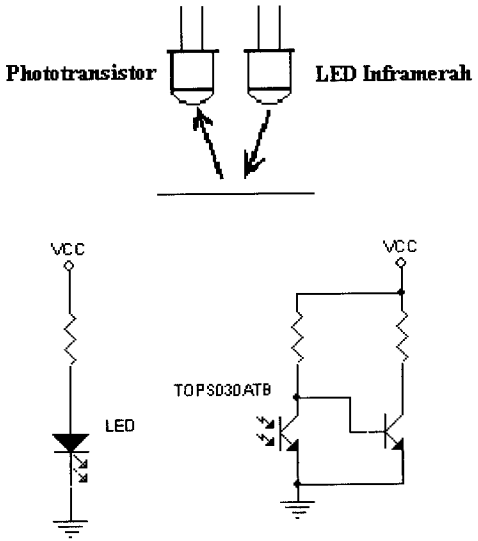
Sensor cahaya yang paling sesuai untuk aplikasi ini adalah sensor inframerah berupa fototransistor. Inframerah memiliki lebar frekuensi yang tertentu (Gambar 2.1) sehingga robot dapat membedakan antara cahaya lampu atau cahaya-cahaya lain yang tidak mengandung inframerah dan cahaya inframerah dari sensor robot. Namun demikian, cahaya matahari juga masih mengandung sinar-sinar inframerah yang mungkin dapat mengganggu. Oleh karena itu, fototransistor yang digunakan sebaiknya memiliki lapisan pelindung cahaya berwarna hitam seperti yang ada pada TOPS030TB untuk fototransistor ukuran 3 mm atau TOPS050TB untuk fototransistor ukuran 5 mm.



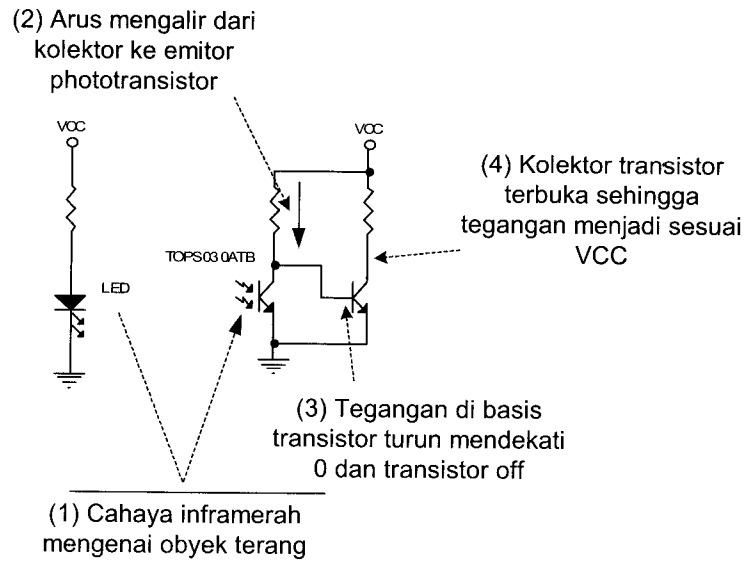
Gambar 2.1 Emisi versus panjang gelombang inframerah

Proses pengenalan terang gelap dilakukan dengan menembakkan sinar inframerah ke objek. Apabila objek yang dituju memiliki warna yang terang, sinar akan dipantulkan dan mengenai fototransistor. Namun, bila objek yang dituju memiliki warna gelap, sinar yang dipantulkan akan lebih sedikit atau bahkan tidak dipantulkan sama sekali karena terserap warna gelap tersebut. Cahaya inframerah yang terpantul akan membias pada fototransistor sehingga fototransistor akan mengalirkan arus dari kolektor ke emitor. Sering kali aliran arus ini masih kurang besar untuk memberikan perubahan kondisi tegangan yang dapat dikenali oleh otak robot. Oleh karena itu, perlu ditambahkan sebuah transistor yang menguatkan perubahan arus tersebut (Gambar 2.2).

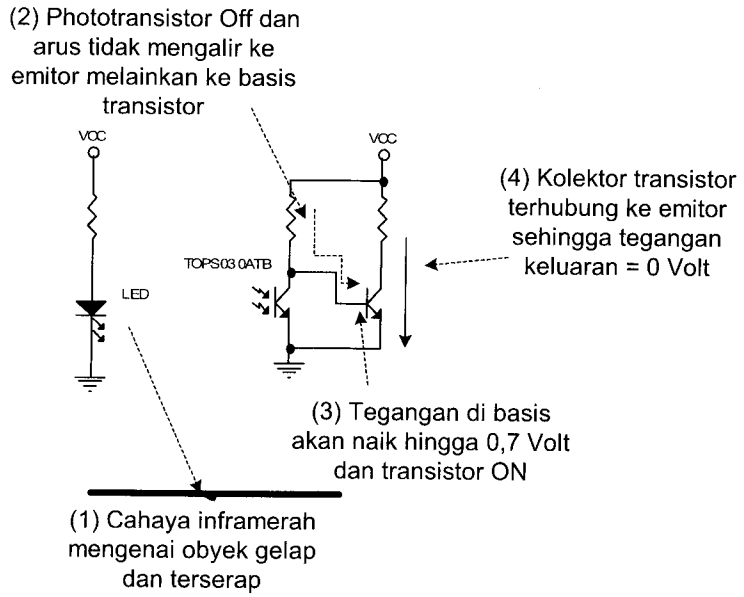
Gambar 2.3 menunjukkan kondisi di mana sensor memperoleh warna terang dan fototransistor mengalirkan arus dari kolektor ke emitor. Karena fototransistor ON (saturasi), kaki kolektornya akan terhubung ke *ground* sehingga tegangan turun mendekati 0 volt. Hal ini mengakibatkan transistor OFF (*cut off*) dan kolektor transistor berada pada kondisi terbuka. Arus akan mengalir dari VCC melalui resistor ke otak robot sehingga input otak robot akan memperoleh tegangan mendekati VCC.



Gambar 2.2 Sensor penjejak garis



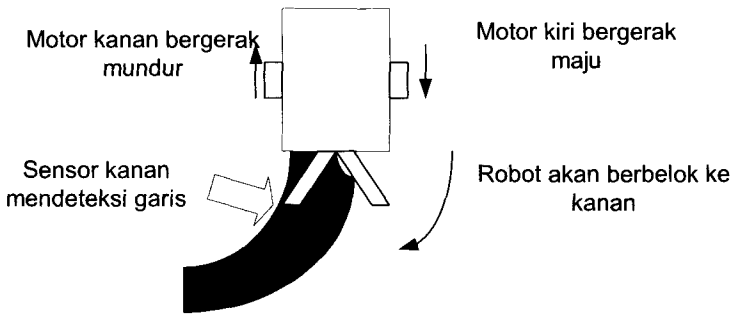
Gambar 2.3 Sensor mengenai objek terang



Gambar 2.4 Sensor mengenai objek gelap

Pada saat cahaya mengenai objek gelap, cahaya inframerah akan terserap dan fototransistor tidak memperoleh bias sehingga berada pada kondisi *cut off*. Arus tidak lagi mengalir ke emitor, melainkan mengalir ke basis transistor dan membuat transistor tersebut saturasi (ON). Kolektor dan emitor akan terhubung sehingga keluaran sensor yang terhubung ke otak robot adalah tegangan 0 volt.

Dengan rangkaian sensor ini akan diperoleh tegangan 0 volt sebagai indikasi adanya warna gelap dan tegangan 5 volt sebagai indikasi adanya warna terang. Dasar proses penjejak garis secara sederhana adalah berbelok ke kiri saat robot terlalu kanan dari garis dan berbelok ke kanan saat robot terlalu kiri dari garis. Robot akan bergerak ke kanan saat sensor kanan mengenai garis dan bergerak ke kiri saat sensor kiri mengenai garis seperti pada Gambar 2.5.

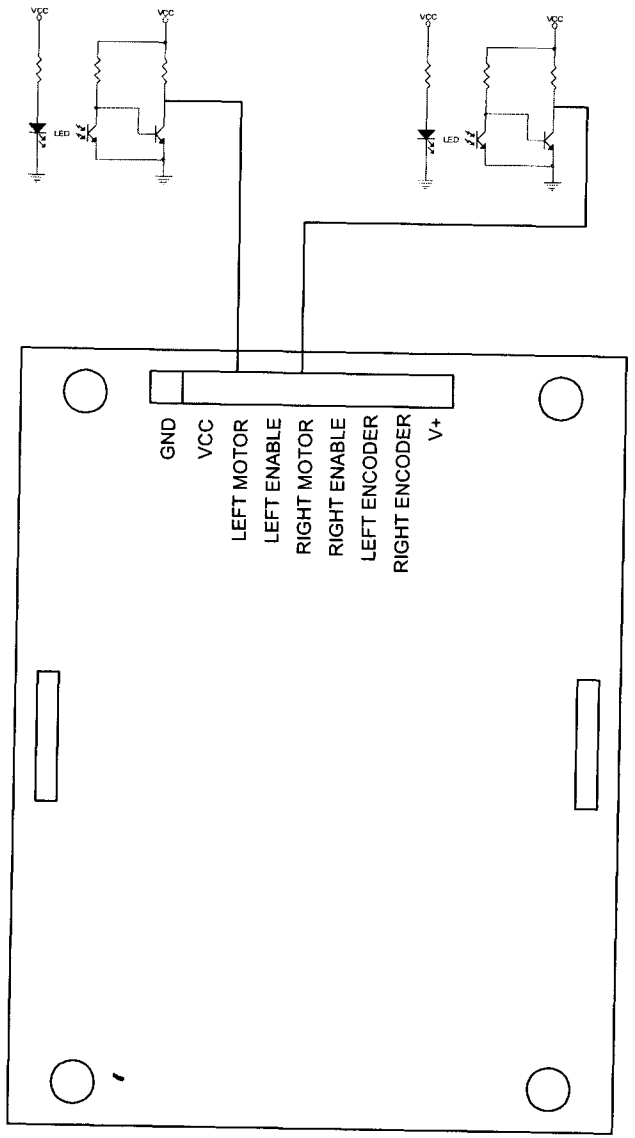


Gambar 2.5 Robot bergerak mengikuti garis

Penggunaan sensor berbentuk sungut akan mempermudah robot penjejak garis dalam menjejak garis dengan berbagai macam ukuran. Sensor yang terletak di ujung sungut yang fleksibel akan mengenali berbagai macam ukuran garis dengan mengatur posisi sungut.

Dalam hal ini, *delta DC driver* yang berfungsi sebagai *dual H-bridge driver* akan menggerakkan motor ke arah belakang saat tegangan di kaki *left* dan *right motor* 0 volt, dan menggerakkan motor ke arah depan saat tegangan di kaki *left* dan *right motor* 5 volt. Sedangkan kaki *left* dan *right enable* dihubungkan ke *ground* (0 volt) untuk mengaktifkan motor. Pada Gambar 2.6 kedua kaki *left* dan *right motor* dari *delta DC driver* dihubungkan ke sensor kiri dan sensor kanan robot. Pada saat sensor kiri mendeteksi garis dan sensor kanan tidak mendeteksi garis, tegangan sensor kiri adalah 5 volt dan sensor

kanan 0 volt. Motor kiri akan bergerak mundur dan motor kanan bergerak maju sehingga robot berputar ke kiri.



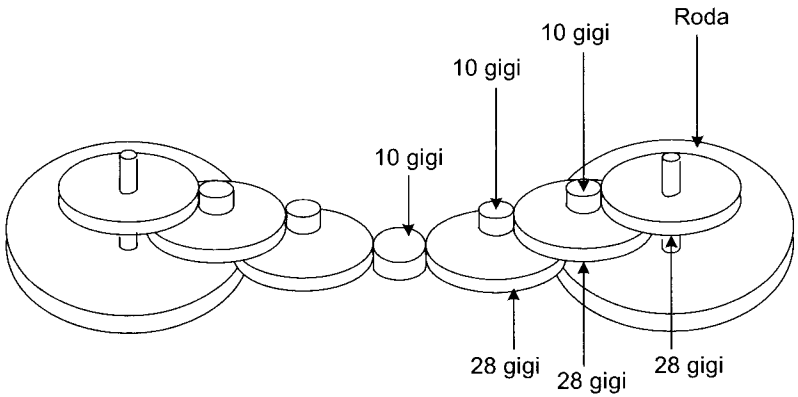
Gambar 2.6 Skema robot penjejak garis

Sebaliknya, saat sensor kanan mendeteksi garis dan sensor kiri tidak, tegangan sensor kanan adalah 5 volt dan sensor kiri 0 volt. Motor kanan akan bergerak mundur dan motor kiri bergerak maju sehingga robot berputar ke kiri.

Untuk mekanik penggerak, di sini kita menggunakan motor Tamiya yang dilengkapi dengan sistem gigi dan roda. Gigi 10 yang terhubung pada as motor terhubung dengan gigi transfer 28 ke 10 sehingga membentuk perbandingan 10:28. Gigi transfer ini juga terhubung lagi dengan gigi transfer 28 ke 10 sehingga membentuk perbandingan 10:28, dan terakhir terhubung dengan gigi 28 yang terhubung juga dengan as roda sehingga diperoleh perbandingan berikut.

$$\frac{10}{28} \times \frac{10}{28} \times \frac{10}{28} = \frac{1000}{21952} = \frac{1}{21{,}952} \text{ atau dapat diambil pendekatan } \frac{1}{22}.$$

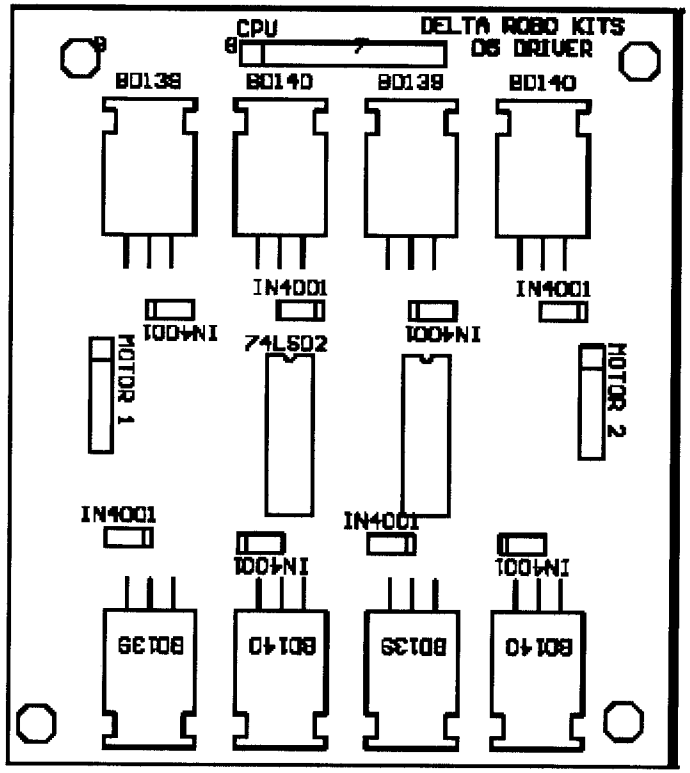
Maka, akan diperoleh torsi 22 kali lebih kuat dengan kecepatan 22 kali lebih lambat.



Gambar 2.7 Mekanik penggerak

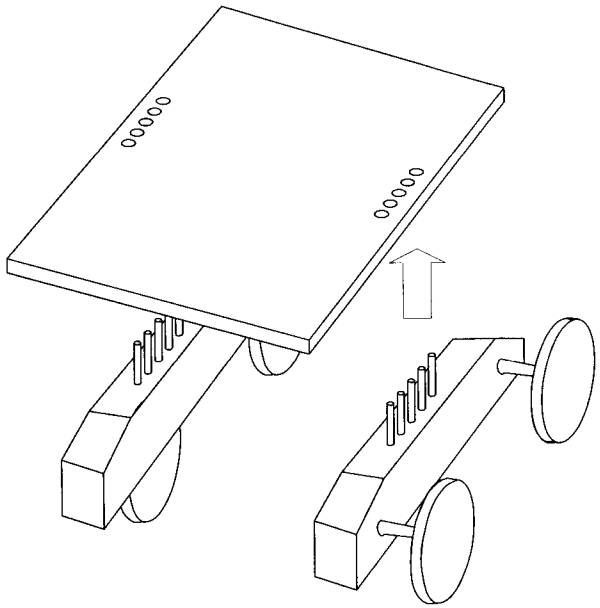
Berikut adalah langkah-langkah penginstalannya.

1. Siapkan *delta DC driver* dan solder komponen-komponen sesuai Gambar 2.8.

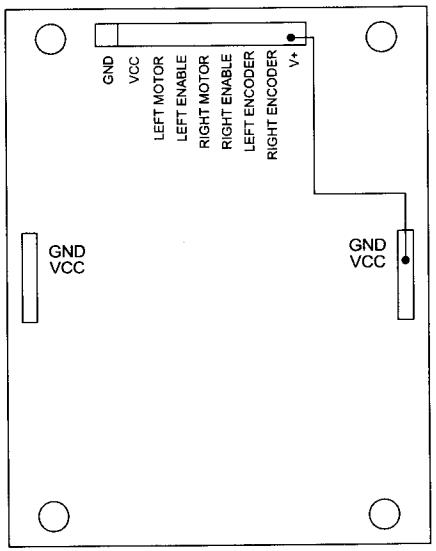


Gambar 2.8 *Delta DC driver*

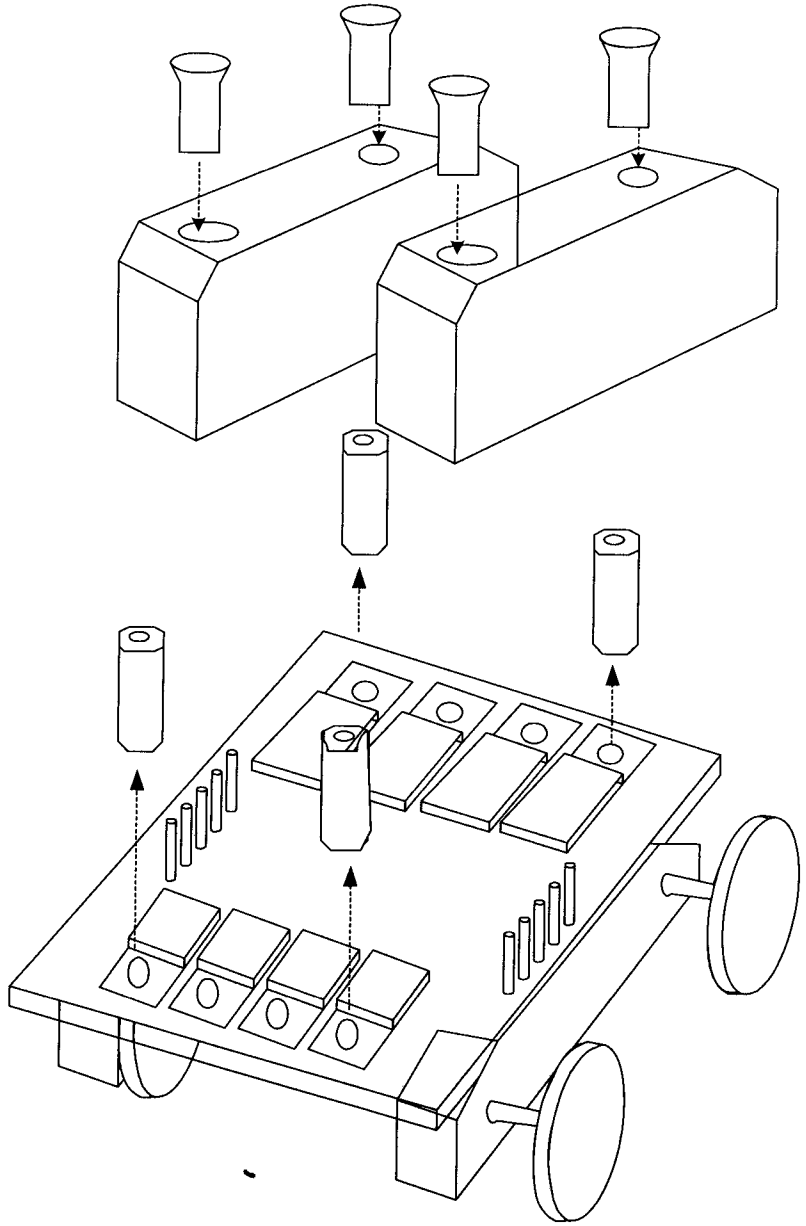
2. Pasang *delta robo wheel* ke *delta DC driver* dan pastikan kelima konektor masuk di lubang-lubang yang ada pada PCB.
3. Pastikan kelima konektor menembus PCB cukup banyak dan lakukan penyolderan pada kelima konektor tersebut.
4. Hubungkan V+ ke VCC dari konektor 5 pin seperti pada Gambar 2.10.
5. Pasang *spacer* di empat posisi *delta DC driver* seperti pada Gambar 2.11.
6. Pasang tempat baterai pada *spacer* seperti pada Gambar 2.11.



Gambar 2.9 Pemasangan *delta DC driver* di mekanik

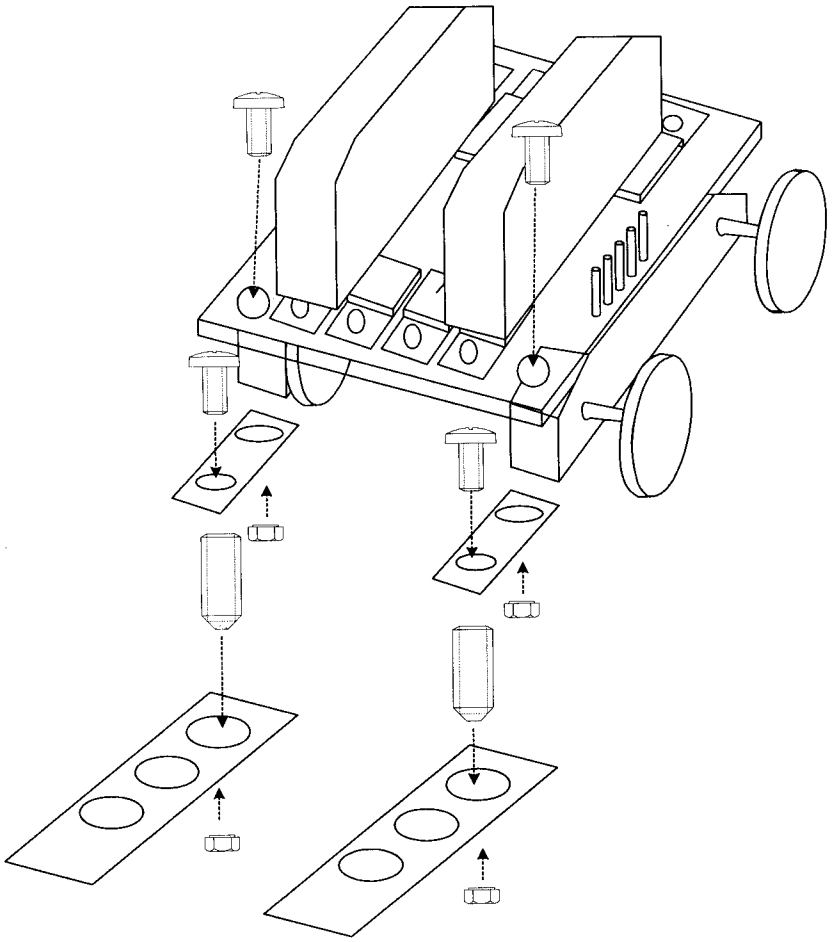


Gambar 2.10 *Jumper* V+ ke motor



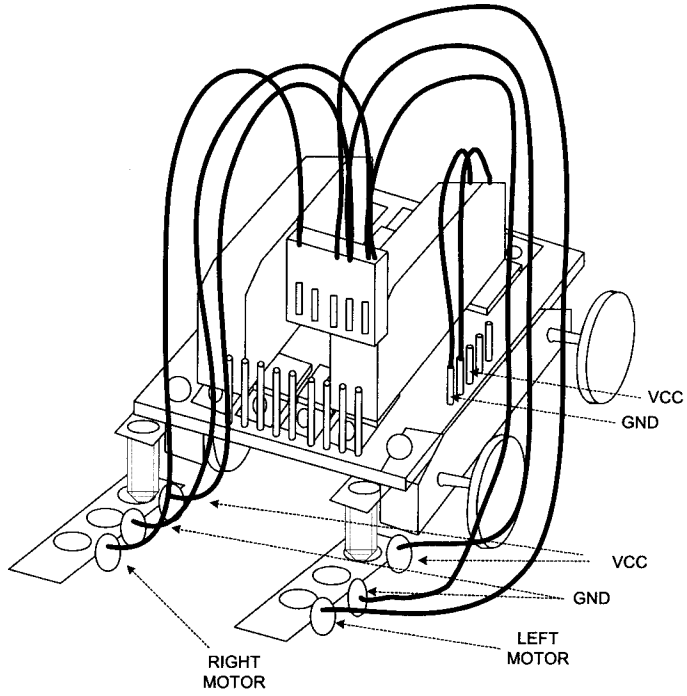
Gambar 2.11 Pemasangan mekanik

7. Pasang sungut (*part A line follower*) seperti pada Gambar 2.12.



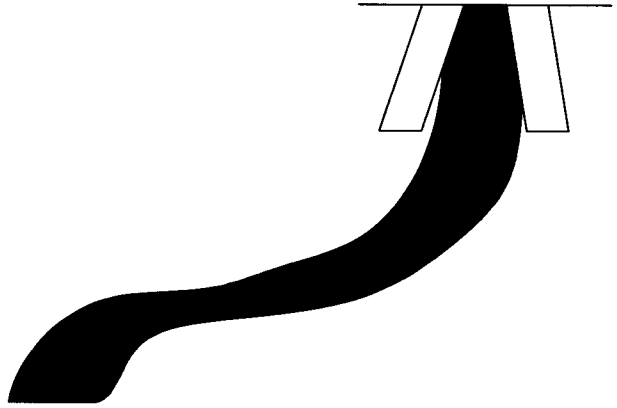
Gambar 2.12 Pemasangan sensor

8. Pasang kabel dari baterai seperti pada Gambar 2.13.
9. Pasang kabel dari *part A line follower* ke *delta DC driver*. Bagian sisi konektor 5 pin dihubungkan ke *delta DC driver* dan bagian sisi konektor kancing dihubungkan ke *part A line follower* seperti pada Gambar 2.13.



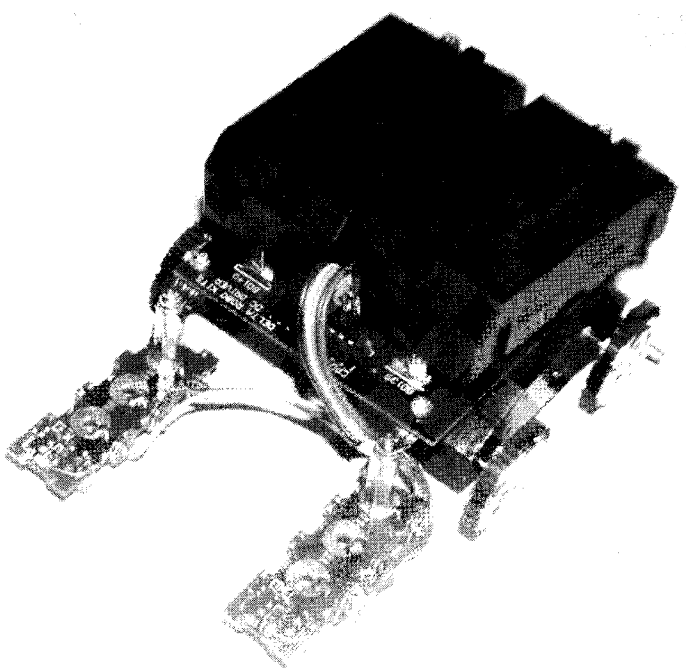
Gambar 2.13 Pengabelan sensor

10. Atur kedua sungut robot agar berada sedikit di luar garis.



Gambar 2.14 Proses sensor mendeteksi garis

11. Aktifkan sakelar baterai dan robot akan bergerak mengikuti lika-liku garis.

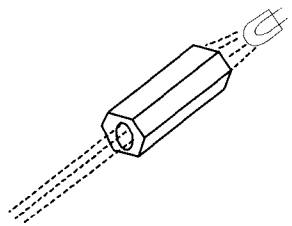


Gambar 2.15 Robot penjejak garis

BAB III
ROBOT TARGET UNTUK SASARAN TEMBAK

Aplikasi robot kali ini berbeda dari biasanya, yaitu robot target untuk sasaran tembak. Dengan menambahkan dua modul *delta infrared fire & target* pada robot yang kita rancang pada Bab II, aplikasi robot sasaran tembak ini dapat dibuat. Bila sebelumnya permainan tembak-menembak sasaran dilakukan di layar televisi dengan Playstation, pada aplikasi robot kali ini permainan tersebut dapat dilakukan secara lebih realistis di luar layar televisi.

Proses menembak robot ini dilakukan dengan memancarkan cahaya inframerah ke sebuah sensor yang ada pada robot. Cahaya inframerah yang terpancar bersifat divergen atau menyebar. Oleh karena itu, agar diperoleh tembakan inframerah yang terfokus, cahaya ini dilewatkan ke sebuah reflektor berupa material logam berbentuk tabung dengan lubang di kedua sisinya. Material ini dapat diperoleh dari *spacer* dengan lubang di kedua sisi.

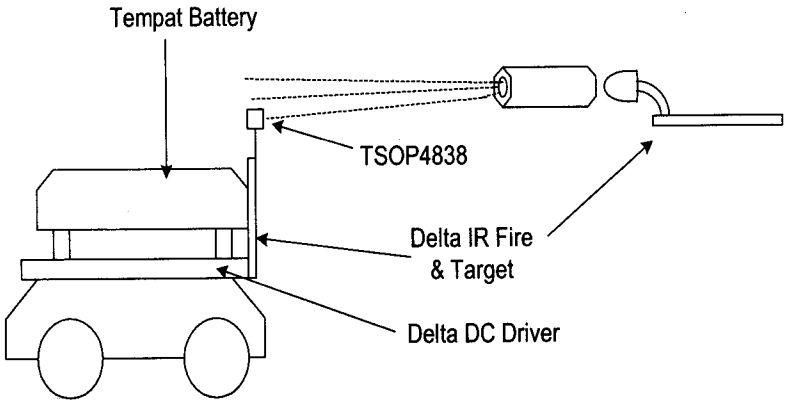


Gambar 3.1 Reflektor

Cahaya yang telah terfokus tersebut akan ditembakkan hingga mengenai TSOP4838 yang berfungsi sebagai *infrared receiver* pada *delta infrared fire & target* yang terpasang pada robot, sedangkan *delta infrared fire & target* akan memerintahkan robot yang menjadi target untuk berhenti. Seperti yang telah dijelaskan sebelumnya pada Bab I, *delta DC driver* yang berfungsi sebagai pengendali motor robot ini akan berhenti apabila kaki *left enable* dan *right enable* diberi tegangan 5 volt, dan *delta*

infrared fire & target akan memberikan tegangan 5 volt pada bagian tersebut sehingga robot terhenti.

Robot hanya akan dapat dijalankan kembali dengan cara merestart sistem, yaitu dengan mematikan tombol *power* pada kotak baterai dan mengaktifkannya kembali.



Gambar 3.2 Cara kerja robot target

TABEL 3.1 Daftar bahan baku

Jumlah	Nama Barang	Kode
1	*Delta DC driver*	003-002
1	*Delta robo wheel*	004-0001
2	*Battery pack*	
2	*Delta infrared fire & target*	
1	*Black housing* 6 pin	022-0043
3	*Black housing* 1 pin	022-0031
2	*Spacer* dengan dua lubang	014-0022

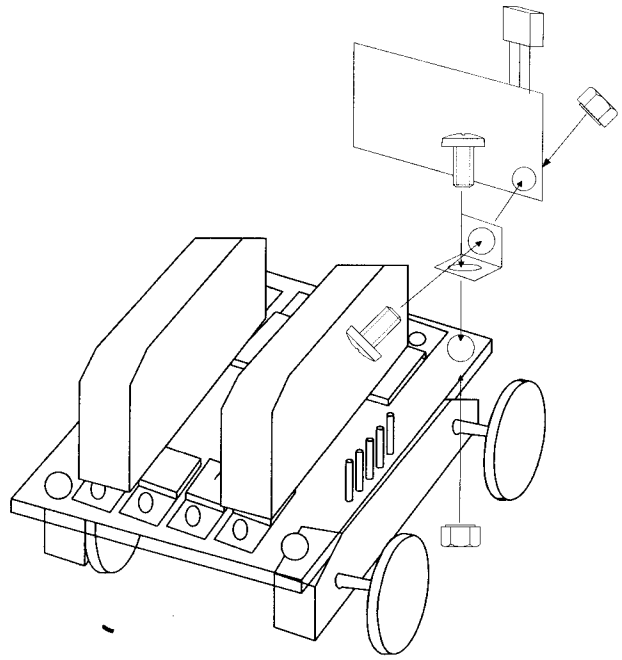
Untuk membangun aplikasi ini, kita membutuhkan bahan-bahan berikut.

1. 1 unit modul *delta DC driver* yang kita rancang pada aplikasi robot di Bab II;
2. 1 unit *delta robo wheel set* berupa motor, *gearbox* lengkap dengan rodanya;

3. 2 unit *battery pack* untuk tempat baterai;
4. 2 unit *delta infrared fire & target.*

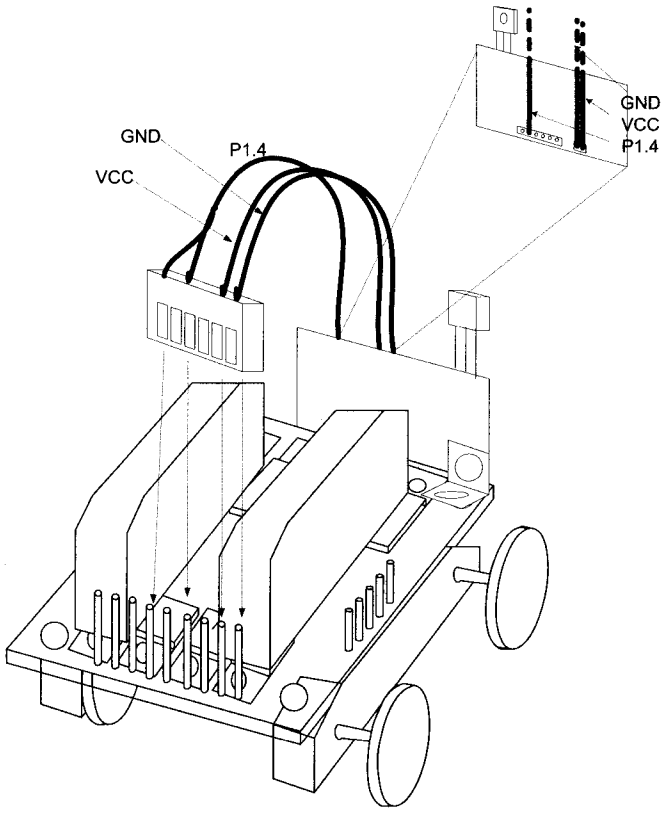
Penjelasan lebih mendetail mengenai bagaimana cara kerja *delta infrared fire & target* menembakkan cahayanya telah dijelaskan pada Bab I. Oleh karena itu, berikut ini akan dijelaskan tentang langkah-langkah penginstalan unit.

1. Siapkan *delta DC driver* dan *delta robo wheel set* seperti pada langkah 1 hingga langkah 8 pada robot penjejak garis (Bab II), kecuali *part A line follower* yang tidak harus dipasang dalam aplikasi ini.
2. Siapkan modul *delta infrared fire & target* 2 unit dan pasang di belakang robot seperti pada Gambar 3.3.

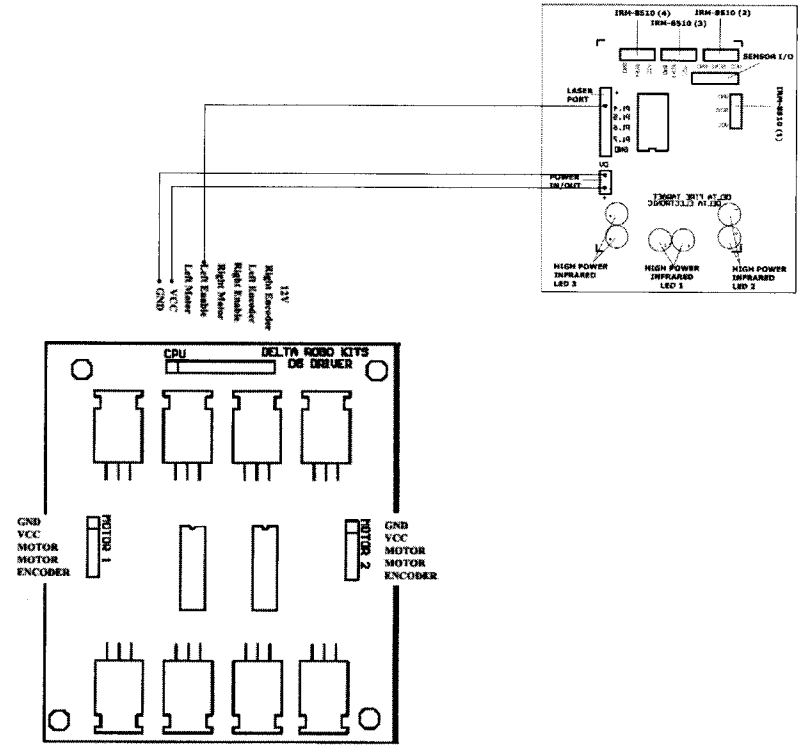


Gambar 3.3 Pemasangan *delta IR fire & target* pada robot

3. Hubungkan +6V dan GND dari modul ke VCC dan GND pada porta *delta DC driver* (Gambar 3.4).
4. Hubungkan keluaran *delta IR fire & target* di P1.4 ke *left enable* dan *right enable delta DC driver.*
5. Diagram pengabelan antara *delta DC driver* dan *delta infrared fire & target* tampak seperti Gambar 3.5.
6. Keluaran dari *delta infrared fire & target* ini akan menghasilkan tegangan 5 volt saat sinar inframerah mengenai bagian sensornya.
7. Seperti pada pembahasan sebelumnya, saat *left enable* dan *right enable* dari *delta DC driver* aktif, motor akan berhenti.

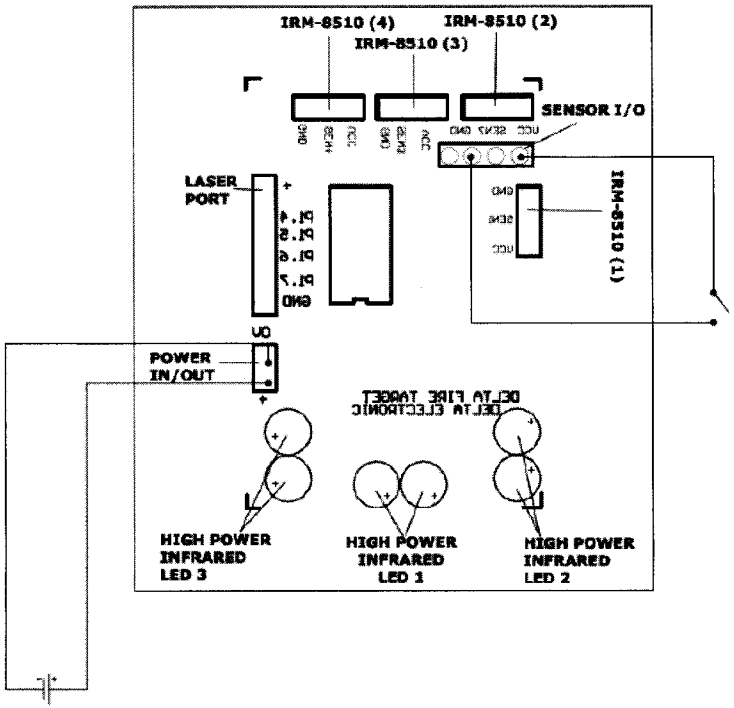


Gambar 3.4 Koneksi *delta IR fire & target* pada robot



Gambar 3.5 Penginstalan modul

8. Robot untuk target telah siap, yang diperlukan berikutnya adalah bagian penembak. Siapkan 1 unit lagi *delta infrared fire & target.*
9. Masukkan LED-LED inframerah ke dalam reflektor berupa *spacer.*
10. Pasang sakelar *push button* di antara P3.0 dan GND (Gambar 3.6).
11. Pasang baterai 6 V di input *power delta infrared fire & target.*
12. Aktifkan robot target dengan menekan sakelar pada baterai.
13. Arahkan reflektor ke arah sensor yang ada pada robot yang sedang berjalan.
14. Robot target akan berhenti saat tembakan mengenai sensor.



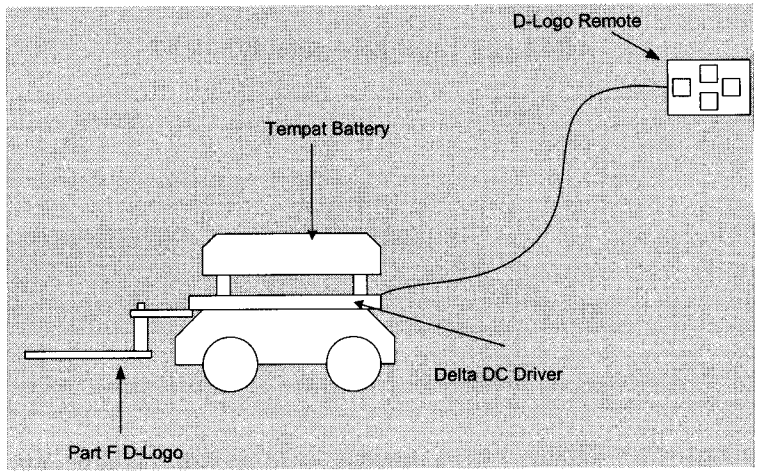
Gambar 3.6 Bagian penembak

Berikut adalah beberapa tip untuk pembuatan robot target ini.

1. Untuk mempermudah penembakan, gunakan pistol mainan yang dilengkapi dengan sakelar pada bagian pelatuk, dan hubungkan ke bagian sakelar *delta infrared fire & target.*
2. Untuk memperpanjang jarak tembak, perpanjang *spacer* dengan dua buah *spacer* dan rekatkan dengan lem pada bagian luarnya.

BAB IV
ROBOT *SOCCER* SEDERHANA

Robot *soccer* sederhana ini adalah aplikasi dasar dari robot *soccer* yang bergerak berdasarkan perintah dari operator. Oleh karena itu, robot ini termasuk jenis *teleoperated* dan membutuhkan *remote* sebagai pengendali.



Gambar 4.1 Robot *soccer* sederhana

Proses kendali tersebut dilakukan dengan menggunakan *D-logo remote* yang terdiri atas empat buah tombol untuk operasi maju, mundur, putar kiri, dan putar kanan, sedangkan *part F D-logo* digunakan untuk menggiring bola yang ada di depan robot.

Agar empat buah sakelar pada *D-logo remote* dapat mengendalikan *delta DC driver*, kita membutuhkan rangkaian yang dapat mengondisikan hasil penekanan sakelar-sakelar tersebut menjadi kondisi-kondisi input yang dibutuhkan oleh *delta DC driver*. Dengan mengacu pada pembahasan bagian *delta DC driver* di Bab I, kita memperoleh gerakan-gerakan motor yang akan terjadi sesuai TABEL 4.1.

TABEL 4.1 Tabel kebenaran *delta DC driver*

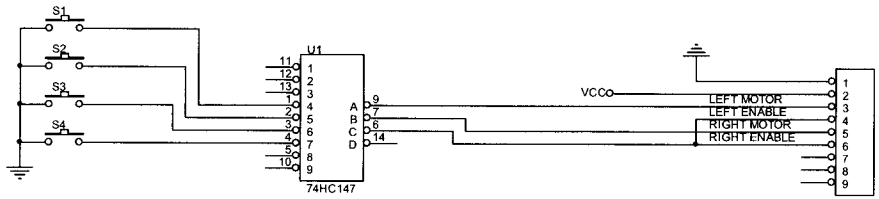
Left Enable	*Right Enable*	*Left Motor*	*Right Motor*	Gerakan Robot
0	0	0	0	Maju
0	0	0	1	Putar kanan
0	0	1	0	Putar kiri
0	0	1	1	Mundur

Kondisi ini dapat diperoleh dengan bantuan *IC encoder 74147* yang memiliki tabel kebenaran seperti pada TABEL 4.2.

TABEL 4.2 Tabel kebenaran IC 74147

Input									Output			
1	2	3	4	5	6	7	8	9	D	C	B	A
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	X	0	1	1	1
X	X	X	X	X	X	0	X	X	1	0	0	0
X	X	X	X	X	0	X	X	X	1	0	0	1
X	X	X	X	0	X	X	X	X	1	0	1	0
X	X	X	0	X	X	X	X	X	1	0	1	1
X	X	0	X	X	X	X	X	X	1	1	0	0
X	0	X	X	X	X	X	X	X	1	1	0	1
0	X	X	X	X	X	X	X	X	1	1	1	0

Pada TABEL 4.1 tampak *left enable* dan *right enable* selalu berada pada kondisi yang sama. Hal ini karena pada aplikasi ini kedua motor selalu aktif bersamaan, perbedaan hanya terletak pada arahnya. Oleh karena itu, kedua input ini dapat dihubungkan bersama pada rangkaian dalam *D-logo remote*.



Gambar 4.2 Skema *D-logo remote*

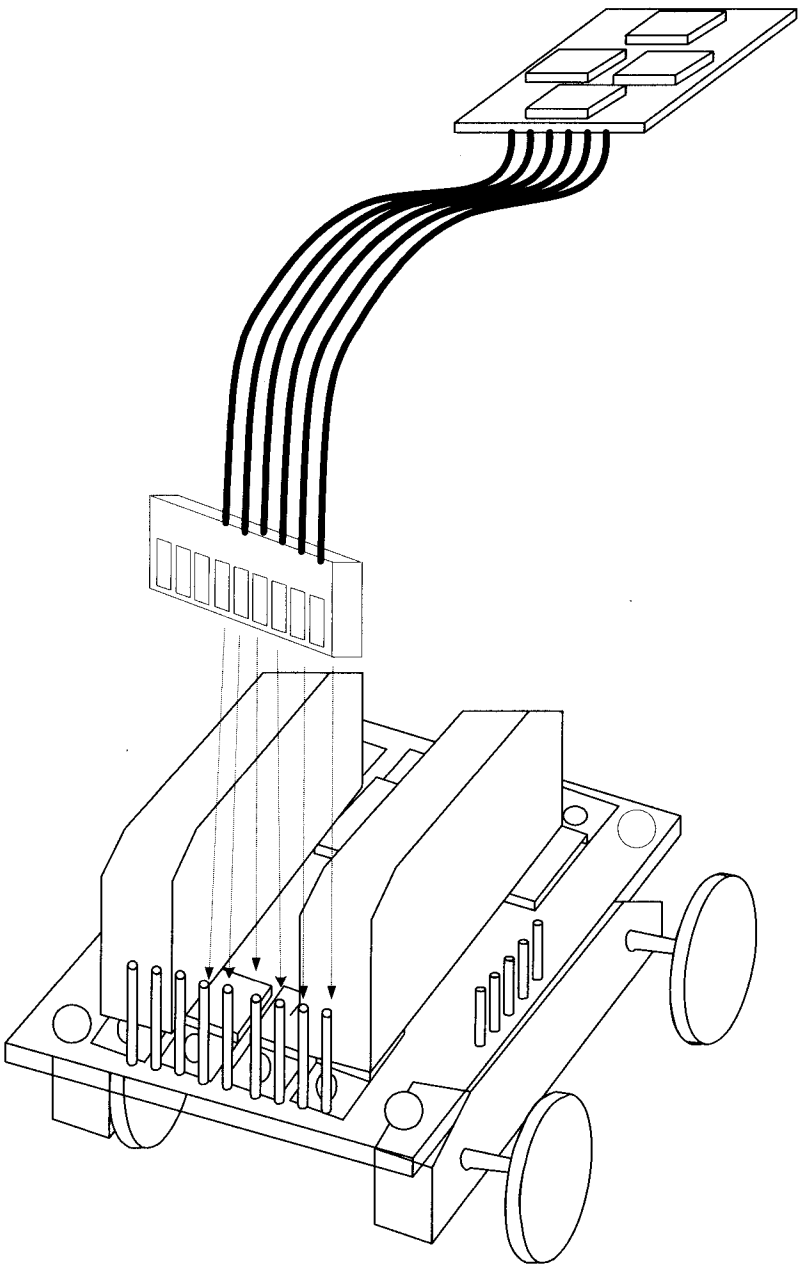
Tampak pada Gambar 4.2 keluaran kaki IC 74147 bagian C terhubung pada *left enable* dan *right enable* dari *delta DC driver*. Dengan rangkaian ini penekanan tombol pada S1, S2, S3, atau S4 akan mengakibatkan kondisi keluaran seperti yang tampak pada bagian berarsir di TABEL 4.2.

TABEL 4.3 Daftar bahan baku

Jumlah	Nama Barang	Kode
1	*Delta DC driver*	003-002
1	*Delta robo wheel*	004-0001
2	*Battery pack*	
4	*Tactile switch*	030-0057
1	IC 74HC147	
1	Konektor 9 pin	029-0017

Berikut adalah langkah-langkah penginstalannya.

1. Siapkan *delta DC driver* dan *delta robo wheel set* seperti pada langkah 1 hingga langkah 8 pada robot penjejak garis (Bab II), kecuali *part A line follower* yang tidak harus dipasang dalam aplikasi ini.
2. Rangkai komponen-komponen seperti pada rangkaian dalam Gambar 4.2.
3. Hubungkan konektor 9 pin dari rangkaian tersebut ke *delta DC driver* seperti pada Gambar 4.3.



Gambar 4.3 Hubungan *D-logo remote* dengan robot

4. Aktifkan sakelar *power* pada *battery pack.*
5. Tekan S1 dan robot akan bergerak maju, S2 untuk memutar robot ke ke kiri, S3 untuk memutar robot ke kanan, dan S4 untuk bergerak mundur.
6. Robot akan berhenti saat sakelar tidak ditekan.

Latihan

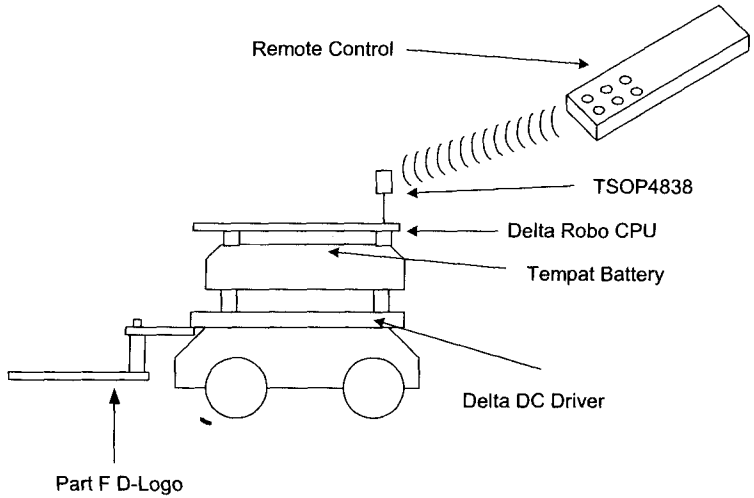
1. Buat tabel hubungan antara tombol-tombol yang ditekan dengan gerakan robot!
2. Jelaskan hubungan antara penekanan tombol, gerakan motor, dan gerakan robot!

BAB V

ROBOT *SOCCER* DENGAN *REMOTE CONTROL* INFRAMERAH

Bila pada aplikasi sebelumnya robot *soccer* dikendalikan dengan menggunakan kabel, pada aplikasi ini robot tidak lagi menggunakan kabel. Penggunaan kabel pada aplikasi robot *soccer* bisa mengganggu jalannya permainan karena ada kemungkinan robot akan tersangkut kabel robot lain. Penggunaan *remote control* inframerah akan mengatasi masalah ini karena perintah-perintah dari *remote* akan dikirimkan ke robot secara nirkabel.

Sinar inframerah memiliki sudut yang sempit sehingga perintah dapat diarahkan ke robot tanpa mengganggu robot lain. Untuk mempersempit sudut sehingga pancaran sinar tidak memengaruhi robot lain, kita dapat menambahkan *spacer* dengan dua lubang seperti pada aplikasi robot sasaran tembak di Bab III.



Gambar 5.1 Robot *soccer* dengan *remote control* inframerah

Pemrosesan sinyal-sinyal inframerah tidak dapat dilakukan langsung oleh *delta DC driver*. Untuk itu, kita akan membutuhkan komponen cerdas mikrokontroler yang terdapat dalam *delta robo CPU*. Seperti yang telah dijelaskan pada bagian *delta robo CPU* di Bab I, modul ini merupakan otak robot yang akan menerima respons dari sensor-sensornya dan memerintahkan *delta DC driver* untuk bergerak sesuai dengan respons yang diterima dari sensor. Dalam hal ini, sensor yang digunakan adalah *infrared receiver* modul TSOP4838 yang mengubah sinyal-sinyal inframerah menjadi data.

Agar *delta robo CPU* dapat berfungsi, terlebih dahulu mikrokontroler yang ada diisi dengan program aplikasi yang diinginkan, yang dalam hal ini adalah program *remote control* inframerah seperti yang ada pada *listing* berikut. Penjelasan lebih mendetail mengenai proses unduh program telah disampaikan di subbab 1.3.

Listing

```
/Disain by Januar Susanto
#include <avr/io.h>
#define RMT_IN      PORTC.7  //pin micro yg terhubung ke
TSSOP

unsigned char count;

void delay(unsigned int time) //prosedur delay
{while(time--);}


void ambil_remote()  //prosedur mengambil data remote sony
{
unsigned char i;

      infra_cmd=0;
      infra_address=0;
      TCCR1B=0x02;  //turn  on  timer  1;  1.000.000  Khz
@8Mhz crystal
      TCNT1=0;
      while(PIN_INFRA==0);
      if(TCNT1>2400)
      {
```

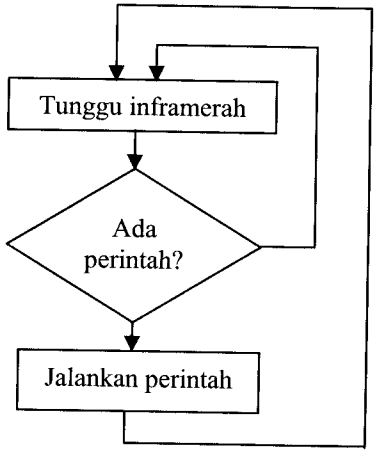


```
            while(PIN_INFRA==1);
            for(i=0;i<7;i++)   //receive infra command (7bit)
            {
                  TCNT1=0;
                  while(PIN_INFRA==0);
                  if(TCNT1>1200) infra_cmd|=0x80;
                  while(PIN_INFRA==1);
                  infra_cmd>>=1;
            }
            for(i=0;i<5;i++)  //receive infra address (5bit)
            {
                  TCNT1=0;
                  while(PIN_INFRA==0);
                  if(TCNT1>1200) infra_address|=0x80;
                  while(PIN_INFRA==1);
                  infra_address>>=1;
            }
      }
      infra_cmd&=0x7F;          //fix 7bit command to 8bit
reading
      infra_address>>=2;      //fix 5bit address to 8bit reading
      infra_address&=0x1F;
      TCCR1B=0x00;                  //turn off timer
}

/*  Definisi koneksi dari driver ke CPU
LeftMotor           PORTB.0
LeftEnable          PORTB.1
RightMotor          PORTB.2
RightEnable   PORTB.3
*/

void main(void)
{
      unsigned char  temp,count;
      while(1)
      {
            temp=ambil_remote();
            if(temp>=0x80)
```

```
                {
                        PORTD.7=0;
                        count=200;
                        switch (temp)
                        {
                        case                0x90                :
PORTB.0=1;PORTB.1=1;PORTB.2=1;PORTB.3=0;
                                break;//belok kiri  > prog +
                case                        0x91                :
PORTB.0=1;PORTB.1=0;PORTB.2=1;PORTB.3=1;
                break;//belok kanan > prog -
                case                        0x92                :
PORTB.0=1;PORTB.1=0;PORTB.2=1;PORTB.3=0;
                                break;//maju   > vol -
                        case                0x93                :
PORTB.0=0;PORTB.1=0;PORTB.2=0;PORTB.3=0;
                                break;//mundur > vol +
                        case                0xA5                :
PORTB.0=1;PORTB.1=1;PORTB.2=1;PORTB.3=1;
                                break;//stop  > TV/Vid
                        case                0xFC                :
PORTB.0=1;PORTB.1=0;PORTB.2=0;PORTB.3=0;
                                break;//putar kiri > select
                        case                0x97                :
PORTB.0=0;PORTB.1=0;PORTB.2=1;PORTB.3=0;
                                break;//putar kanan > A/B
                        }
                }
                if(count == 0)PORTD.7 =1;
                else count--;
        }

}
```

Algoritme program kontrol robot inframerah sebenarnya cukup sederhana, robot akan menunggu perintah yang diterima dari inframerah. Jika perintah yang diterima benar, robot akan mengerjakan perintah tersebut dan kembali menunggu perintah berikutnya dari inframerah.

Prosedur "ambil remote" pada *listing* program di atas berguna untuk mengambil data dari inframerah. Data ini tergantung pada lamanya periode 0 atau 1 dari inframerah tersebut. Data yang dikirim sebesar 8 bit. *Timer* pada *listing* program di atas berguna untuk mengukur lamanya periode 0 atau 1 inframerah. *Timer* dikonfigurasikan dalam mode 16 bit *timer* dan interupsinya diaktifkan. Format data yang dikirimkan adalah *least significant bit* (LSB). Berikut adalah algoritme *listing* program di atas.



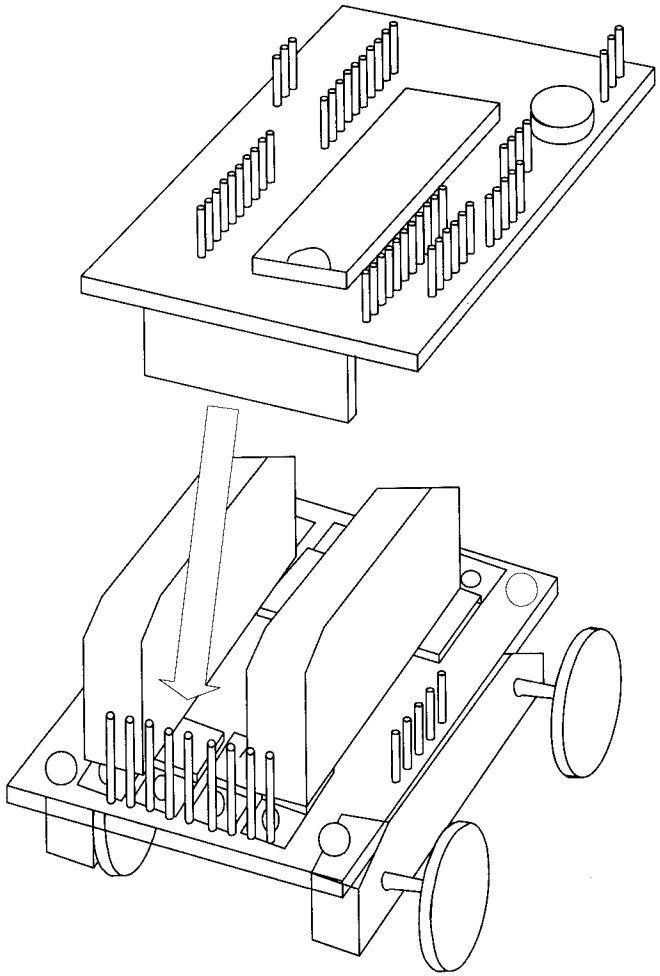
Gambar 5.2 Diagram alir

TABEL 5.1 Daftar bahan baku

Jumlah	Nama Barang	Kode
1	*Delta DC driver*	003-002
1	*Delta robo wheel*	004-0001
2	*Battery pack*	
4	*Delta robo CPU*	003-0001
1	*Part C infrared receiver*	001-0057
1	*Remote control* televisi	
1	*Part F D-logo*	001-0060

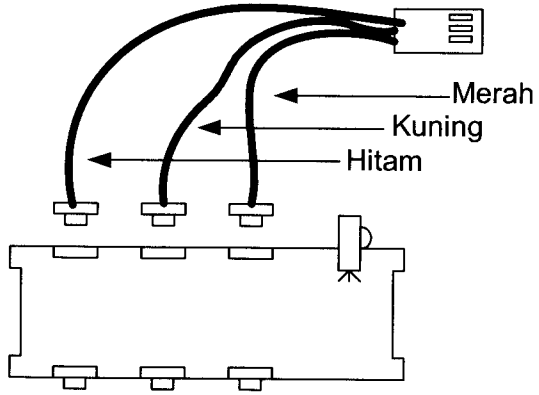
Berikut adalah langkah-langkah penginstalannya.

1. Siapkan *delta DC driver* dan *delta robo wheel set* seperti pada langkah 1 hingga langkah 8 pada robot penjejak garis (Bab II), kecuali *part A line follower* yang tidak harus dipasang dalam aplikasi ini.
2. Siapkan *delta robo CPU* dan hubungkan ke konektor 9 pin pada *delta DC driver*.

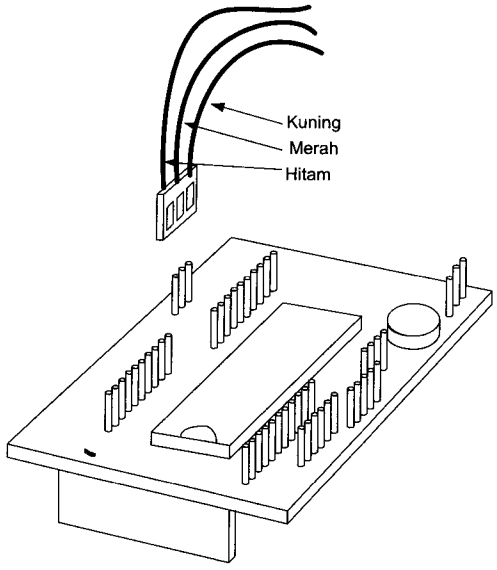


Gambar 5.3 Penginstalan *delta robo CPU* pada robot

3. Pasang *part C infrared receiver*, yaitu potongan PCB yang memiliki sensor TSOP4838.
4. Hubungkan kabel-kabel *part C infrared receiver*.

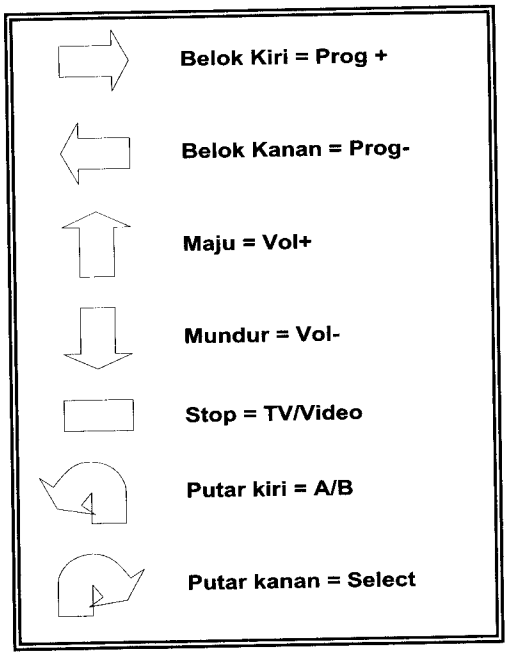


Gambar 5.4 *Part F D-logo*



Gambar 5.5 Koneksi sensor dengan *delta robo CPU*

5. Hubungkan konektor 3 pin dari *part C infrared receiver* ke konektor 3 pin *infrared receiver port* dari *delta robo CPU*.
6. Aktifkan sakelar *power* pada *battery pack*.
7. Unduh program remotekontrol.c ke *delta robo CPU* dengan cara yang telah dijelaskan di subbab 1.3.
8. Ambil *remote control* televisi dan pasang *spacer* dengan dua lubang di depan LED inframerahnya.
9. Arahkan LED inframerah ke sensor *part C infrared receiver* pada robot dan gerakkan robot dengan menggunakan tombol-tombol berikut.



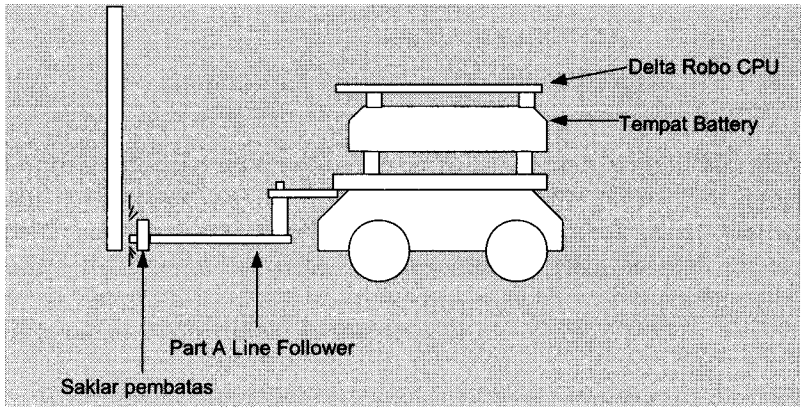
Gambar 5.6 Tombol-tombol gerak

Latihan

1. Coba gunakan fungsi-fungsi tombol lain dari *remote control* untuk menggerakkan robot!
2. Pelajari lembar data TSOP4838 dan jelaskan proses deteksi sinyal inframerah pada TSOP4838!
3. Ukur jarak terjauh yang dapat dicapai *remote control* televisi ke TSOP4838!

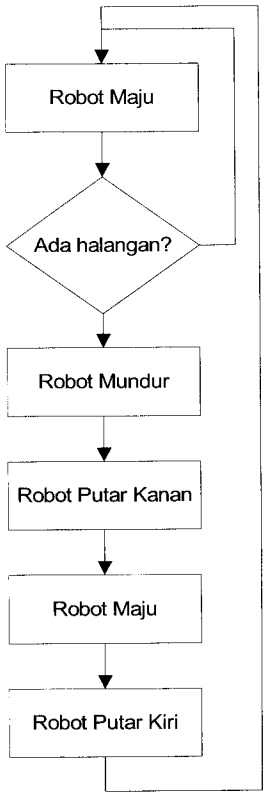
BAB VI
ROBOT PENGHINDAR HALANGAN

Aplikasi robot penghindar halangan termasuk jenis robot otomatis seperti penjejak garis di mana robot bergerak bukan berdasarkan perintah operator, melainkan dari kondisi-kondisi sensor yang ada pada saat itu. Apabila pada penjejak garis robot bergerak mendeteksi garis, robot pendeteksi halangan ini akan bergerak menghindari halangan yang ada di depannya.



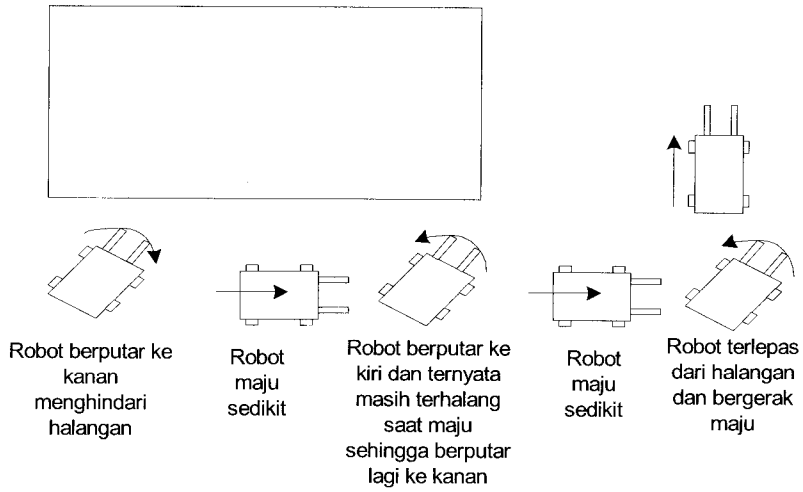
Gambar 6.1 Robot penghindar halangan

Dengan sebuah *tactile switch* sebagai sakelar pembatas dan disolder pada ujung depan *part A line follower*, sensor halangan yang sederhana dapat terpasang pada robot ini. Sensor ini akan aktif saat bagian depan robot membentur halangan. Setelah robot mengetahui adanya halangan, robot harus mengambil tindakan terhadap halangan tersebut. Gambar 6.2 adalah algoritme yang diambil saat robot mendeteksi halangan.



Gambar 6.2 Diagram alir robot penghindar halangan

Robot akan bergerak mundur, berputar ke kanan, bergerak maju, dan berputar lagi ke kiri untuk melewati halangan tersebut. Apabila halangan masih ada, robot akan berputar lagi, maju, putar kiri, dan maju lagi seperti pada Gambar 6.3.



Gambar 6.3 Proses menghindari halangan

Kondisi-kondisi input dari sakelar pembatas tersebut akan diinformasikan ke mikrokontroler yang menjadi otak robot melalui porta-porta inputnya. Seperti pada aplikasi sebelumnya, agar otak robot dapat merespons kondisi input-input tersebut, mikrokontroler yang ada di dalamnya terlebih dahulu diisi dengan *listing* program berikut. Langkah-langkah pemrograman dapat dilihat di subbab 1.3.

Listing

```
/Disain by Januar Susanto
#include<avr\io.h>

/*Definisi hubungan driver motor dan sensor ke mcs51 */
#define MOTOR_KIRI      PORTB.0
#define KIRI_NYALA      PORTB.1
#define MOTOR_KANAN     PORTB.2
#define KANAN_NYALA     PORTB.3
#define SENS_KIRI       PORTB.4
#define SENS_KANAN      PORTB.5

unsigned char i;
```

```
void mundur(void) //prosedur robot mundur
{
      KIRI_NYALA=0;KANAN_NYALA=0;
      MOTOR_KIRI=0;MOTOR_KANAN=0;
}

void maju(void) //prosedur robot maju
{
      KIRI_NYALA=0;KANAN_NYALA=0;
      MOTOR_KIRI=1;MOTOR_KANAN=1;

}

void kiri(void) //prosedur robot belok kiri
{
      KIRI_NYALA=0;KANAN_NYALA=0;
      MOTOR_KIRI=0;MOTOR_KANAN=1;
}

void kanan(void) //prosedur robot belok kanan
{
      KIRI_NYALA=0;KANAN_NYALA=0;
      MOTOR_KIRI=1;MOTOR_KANAN=0;
}

void berhenti(void) //prosedur robot berhenti
{
      KIRI_NYALA=1;KANAN_NYALA=1;
}

/* main program          */
void main(void)
{
      berhenti();
      SENS_KIRI=1;
      SENS_KANAN=1;
      maju();
      i=0;
      while(1)
```



```
      {
            if(i==1)
            {

      if((SENS_KIRI==0)||(SENS_KANAN==0))
                  {
                        mundur();
                        delay(1000);
                        kanan();
                        delay(1000);
                        i=1;
                  }
                  kiri();
                  delay(500);
                  i=0;
            }
            if(i==0)
            {

      if((SENS_KIRI==0)||(SENS_KANAN==0))
                  {
                        mundur();
                        delay(1000);
                        kanan();
                        delay(1000);
                        i=1;
                  }
            }
            maju();
}
```

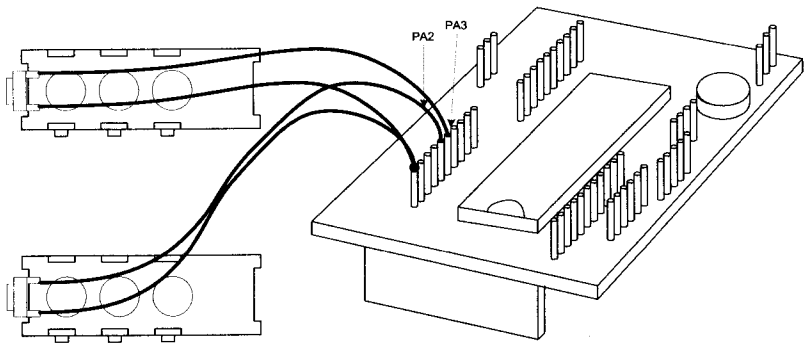
Pada *listing* program di atas terlebih dahulu kita mendefinisikan sambungan dari CPU ke *driver* dan juga ke sensor kiri dan kanan. Untuk prosedur pergerakan robot kita membuatnya sendiri-sendiri untuk mempermudah pengembangan program selanjutnya, dan untuk algoritme program, sudah dijelaskan di atas.

Variabel "i" pada *listing* program di atas berguna sebagai penanda apakah robot sudah pernah terkena halangan atau belum. Penjelasannya seperti

berikut, jika robot terkena halangan, robot berputar ke kanan, terus maju, dan variabel "i" diberi nilai 1 untuk menandakan robot sudah pernah terkena halangan. Jika pada saat maju, setelah terkena halangan, sensor tidak terhalang, robot berputar ke kiri, kemudian maju lagi, variabel "i" diberi nilai 0 untuk menandakan robot tidak terkena halangan. Jika sebaliknya, robot berputar ke kanan dan variabel "i" tetap diberi nilai 1, artinya masih ada penghalang di depan robot dan robot berputar ke kanan, terus maju sampai tidak terkena halangan.

Berikut adalah langkah-langkah penginstalannya.

1. Siapkan *delta DC driver* dan *delta robo wheel set* seperti pada langkah 1 hingga langkah 8 pada robot penjejak garis (Bab II).
2. Siapkan *delta robo CPU* dan hubungkan ke konektor 9 pin pada *delta DC driver*.
3. Pasang *tactile switch* di bagian depan *part A line follower* (Gambar 6.4).
4. Hubungkan kabel dari *tactile switch* ke GND, PA1, dan PA3 (Gambar 6.4).
5. Aktifkan sakelar *power* di baterai dan unduh program obstacle1.c seperti yang dijelaskan pada subbab 1.3.



Gambar 6.4 Penginstalan sensor pada *delta robo CPU*

6. Jalankan robot dan berikan halangan di depan robot. Robot akan bergerak menghindari halangan tersebut setiap sensor menyentuh halangan.

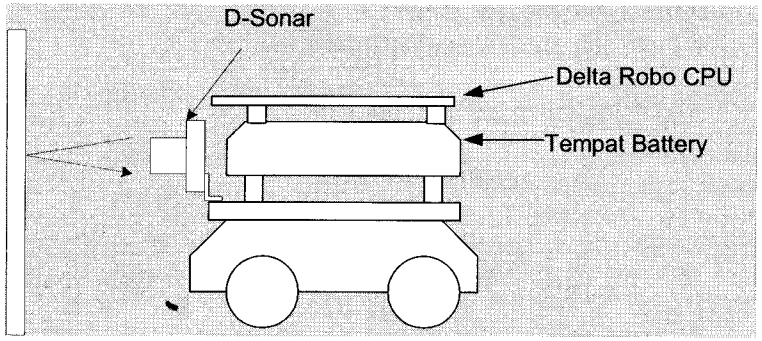
Latihan

1. Coba buat program dengan algoritme robot berputar ke kiri saat sensor sakelar pembatas mengenai objek!
2. Coba buat program di mana robot berbelok ke kanan dan lurus setiap menemui halangan!

BAB VII
ROBOT PENGHINDAR HALANGAN DENGAN ULTRASONIK

Bila pada aplikasi sebelumnya robot akan menghindari halangan saat sensor membentur halangan tersebut, pada aplikasi robot penghindar halangan dengan ultrasonik, robot akan menghindari halangan saat akan membentur. Oleh karena itu, robot tidak perlu melakukan kontak fisik dengan halangan yang dapat menimbulkan kerusakan pada bagian mekanik.

Pendeteksian halangan sebelum robot membentur dilakukan dengan sebuah sensor yang dapat mengetahui keberadaan halangan dari jarak tertentu. Sensor ultrasonik yang menembakkan sinyal ultrasonik dan menghitung lama pantulan akan membuat robot mengetahui jarak dengan halangan yang ada di depannya. Untuk mempermudah proses antarmuka dan pemrograman robot, pengguna dapat menggunakan modul *D-sonar*, yang akan menangani penghitungan jarak. Otak robot hanya perlu mengirimkan perintah untuk meminta informasi dan memutuskan tindakan apa yang harus dilakukan saat informasi jarak diterima.



Gambar 7.1 Robot penghindar halangan dengan ultrasonik

Seperti yang telah dijelaskan di Bab I bahwa modul *D-sonar* dapat bekerja dengan menggunakan mode *trigger-echo*, demikian pula pada aplikasi ini. *Delta robo CPU* akan mengirimkan *trigger* ke modul *D-sonar* dan mengukur lebar pulsa *echo* yang dihasilkan.

Untuk algoritme robot dalam menghindari halangan dapat digunakan algoritme yang sama dengan aplikasi di Bab VI. Perbedaannya hanya terletak pada bagaimana robot mendeteksi halangan. Bila pada Bab VI robot mendeteksi halangan setelah sensor membentur, pada aplikasi ini robot akan selalu menanyakan informasi jarak pada modul *D-sonar* sambil bergerak.

Listing

```
/Disain by Januar Susanto
#include <avr/io.h>

/* definisi sambungan dari CPU ke DSONAR */
#define USI  PORTD.2  //Trigger
#define USO  PORTD.3  //Echo


unsigned char i;
bit x;

at 0x90 sbit  md1; // kiri
at 0x91 sbit  me1; // enable kiri
at 0x92 sbit  md2; // kanan
at 0x93 sbit  me2; // enable kanan

void delay(unsigned int tunda)
{
      while(--tunda);
}

void cek_jarak() //prosedur mengecek jarak dengan ultrasonik
{
      TCCR1B=0x02; //turn on timer 1; 1.000.000 Khz
@8Mhz
      TCCR0=0x02; //turn on timer 0; 1.000.000 Khz @8Mhz
      TCNT1=0;
```



```
      while(TCNT1<1200){USI=1;}
      TCCR0=0x00;
      USI=0;
      TCNT1=0;
      USO=1;
      while(!USO);
      TCCR0=0x02;
      while(USO);
      TCCR0=0x00;

}

void perjalanan(bit a, bit b, bit c, bit d) //pergerakan robot
{
      me1=a;
      me2=b;
      md1=c;
      md2=d;
}


void main()
{
      USO=0;
      while(1)
      {
            cek_jarak();
            if(TCNT0<2000)
            {
                  PORTD.7=1;
                  perjalanan(0,0,1,1);// maju
            }
            if(TCNT0>2000)
            {
                  PORTD.7=0;
                  perjalanan(0,0,0,0);      // mundur
                  for(i=0;i<100;i++)delay(6000);
                  PORTD.7=1;
                  perjalanan(0,0,x,~x);// belok kanan / kiri
```

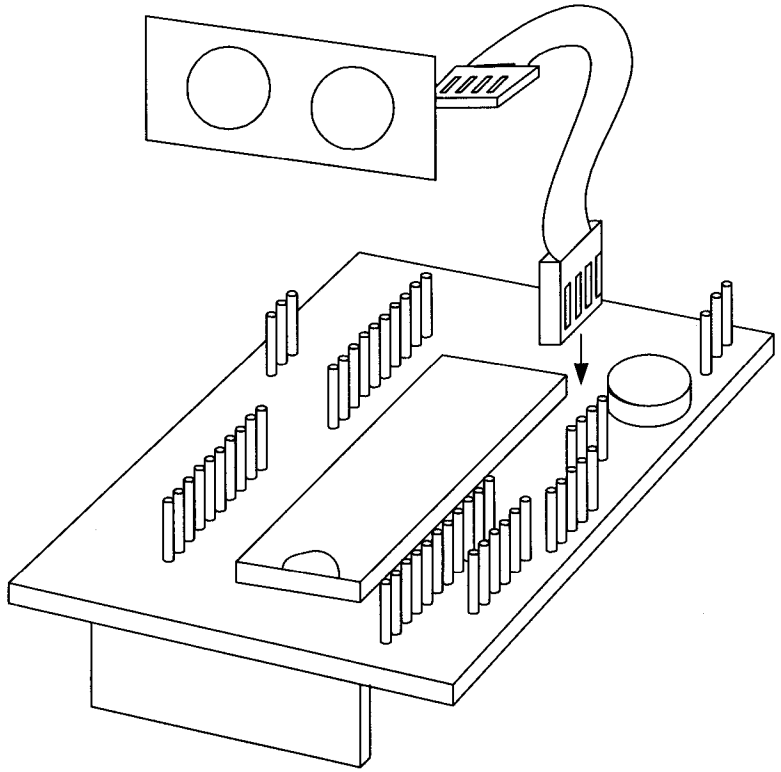
```
                for(i=0;i<100;i++)delay(3000);
            }
            x=~x;
            delay(1500);

        }
    }
```

Pengecekan jarak menggunakan *D-sonar* dilakukan dalam prosedur "cek jarak". Hal yang pertama dilakukan adalah sebagai berikut.

1. Aktifkan *timer*.
2. Beri *trigger* ke *D-sonar* untuk melakukan pengukuran jarak dengan cara memberi logika 1 ke pin *trigger D-sonar*.
3. Tunggu sampai *D-sonar* selesai melakukan pengukuran dengan membaca perubahan logika pada pin *echo* pada *D-sonar*. Pin *echo* ini ada dua, yaitu aktif 1, aktif 0.
4. Bandingkan dengan nilai *timer*. Jika *timer* < *D-sonar*, robot bergerak maju. Jika sebaliknya, robot bergerak mundur.

Berikut adalah langkah-langkah penginstalannya.

1. Siapkan *delta DC driver* dan *delta robo wheel set* seperti pada langkah 1 hingga langkah 8 pada robot penjejak garis (Bab II), kecuali *part A line follower*.
2. Siapkan *delta robo CPU* dan hubungkan ke konektor 9 pin pada *delta DC driver*.
3. Pasang modul *D-sonar* di bagian depan *delta robo CPU*.
4. Hubungkan konektor porta I/O *D-sonar* ke *ultrasonic port* dari *delta robo CPU*.
5. Aktifkan sakelar *power* di baterai dan unduh program obstacle1.c seperti yang dijelaskan pada subbab 1.3.



Gambar 7.2 Penginstalan *D-sonar* pada robot

6. Jalankan robot dan berikan halangan di depan robot. Robot bergerak menghindari halangan tersebut setiap sensor akan menyentuh halangan.

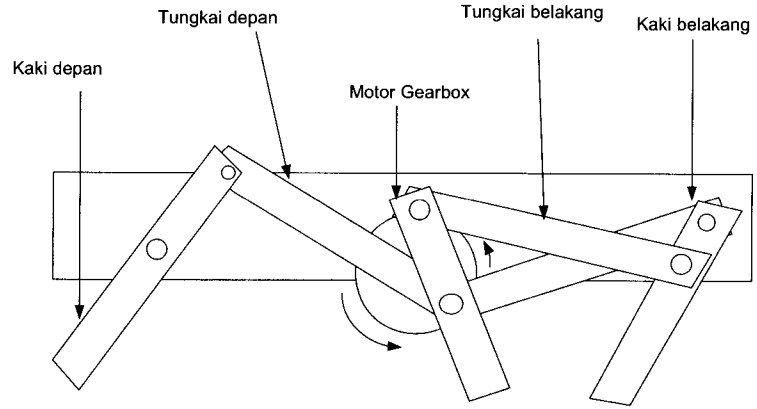
Latihan

1. Coba buat program dengan algoritme robot berputar ke kiri saat sensor berada pada jarak 10 cm dari halangan!
2. Coba buat program di mana robot akan mundur sejauh 5 cm sebelum berputar ke kanan!

BAB VIII

ROBOT BERKAKI ENAM YANG DIKENDALIKAN *REMOTE CONTROL* INFRAMERAH

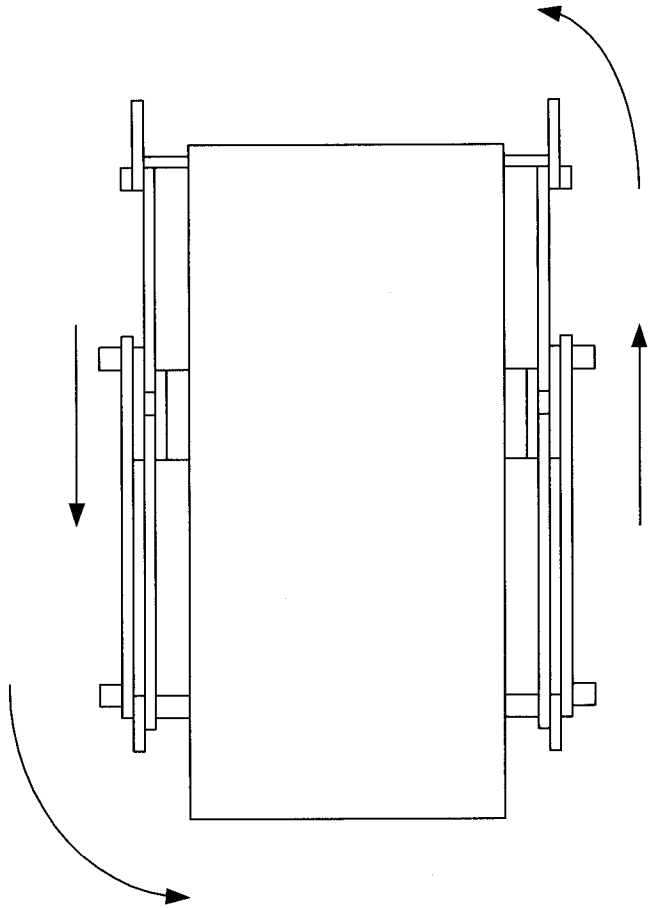
Pada aplikasi kali ini kita menggunakan robot jenis *walker*, yaitu robot yang bergerak bukan dengan roda, melainkan dengan kaki. Robot beroda memang lebih sesuai untuk medan-medan halus dan rata. Untuk medan yang tidak rata robot berkaki lebih sesuai dan lebih leluasa bergerak dibandingkan robot beroda. Gambar 8.1 merupakan robot berkaki enam sederhana yang bergerak dengan mengubah putaran motor menjadi langkah kaki, baik maju untuk putaran motor ke arah depan maupun mundur untuk putaran motor ke arah belakang.



Gambar 8.1 *Delta hexapod mechanic* 005-0033

Motor *gearbox* yang bergerak memutar akan mengakibatkan kaki bagian tengah robot melangkah maju atau mundur. Setengah putaran maju dari motor yang akan menggerakan kaki tengah ke depan akan mengakibatkan kaki depan robot terdorong maju melalui tungkai depan dan sekaligus menarik tungkai belakang agar kaki belakang menjejak atau mendorong ke depan. Setengah putaran berikutnya akan membuat gerakan menjejak atau mendorong ke belakang pada kaki tengah. Gerakan ini juga

membuat kaki belakang tertarik untuk melangkah maju dan demikian pula dengan kaki depan. Proses yang berulang-ulang akan mengakibatkan robot bergerak melangkah maju. Sedangkan gerakan mundur dapat dilakukan hanya dengan memutar motor ke arah sebaliknya. Selain bergerak maju mundur, robot juga dapat berputar ke kiri dan ke kanan dengan melakukan gerakan-gerakan seperti pada Gambar 8.2. Gerakan melangkah maju pada bagian kanan robot dan melangkah mundur pada bagian kiri robot akan mengakibatkan robot berputar ke kiri dan sebaliknya, akan mengakibatkan robot berputar ke kanan.

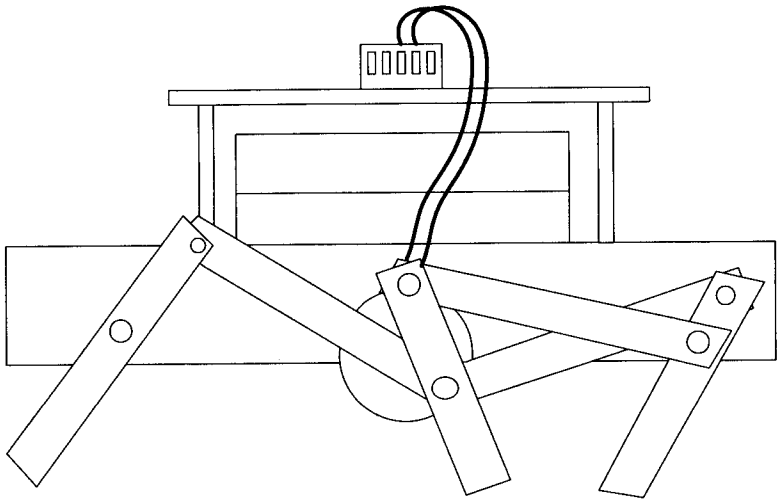


Gambar 8.2 Proses berputar *delta hexapod mechanic*

Robot pada aplikasi ini dikendalikan dengan *remote control* inframerah sehingga teknik pemrograman maupun *listing* program sama persis dengan yang ada pada Bab V tentang robot *soccer* dengan *remote control* inframerah.

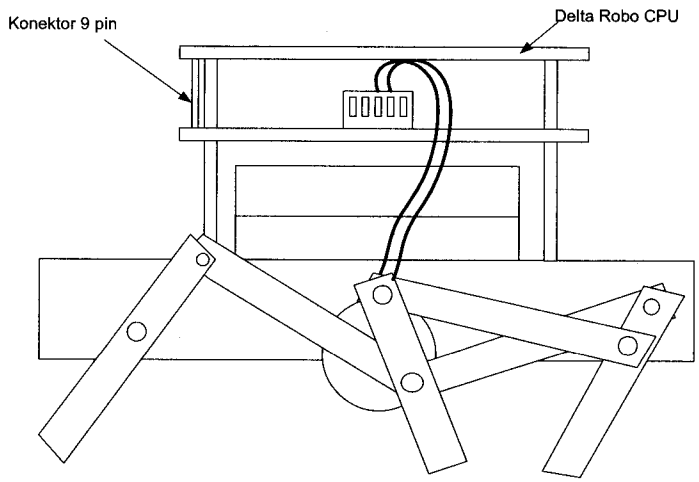
Berikut adalah langkah-langkah penginstalannya.

1. Siapkan mekanik heksapoda, ikuti petunjuk penginstalan seperti yang telah dijelaskan di Bab I di bagian kerangka robot *walker*.
2. Siapkan *delta DC driver* dan hubungkan konektor 5 pin ke bagian konektor motor *delta DC driver*.



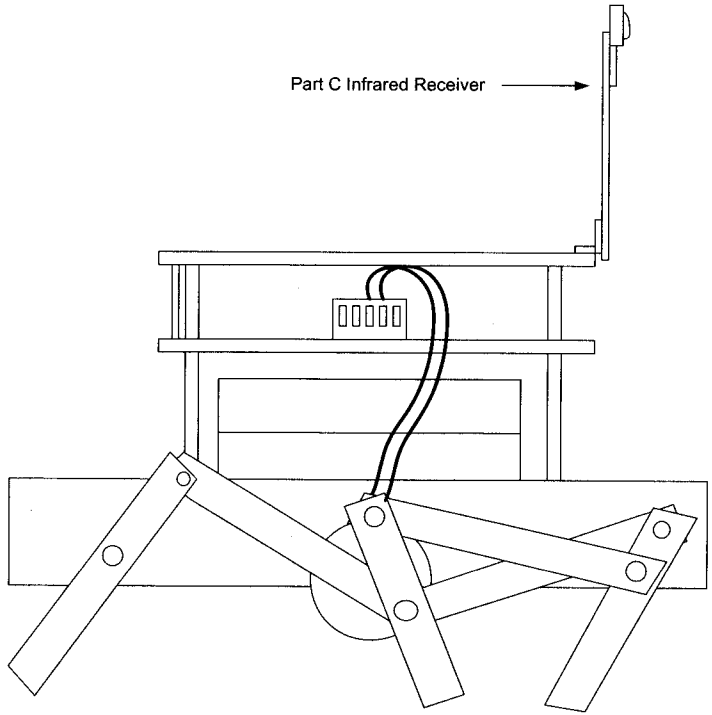
Gambar 8.3 Hubungkan *delta DC driver* dan motor

3. Siapkan *delta robo CPU* dan hubungkan ke konektor 9 pin pada *delta DC driver*.



Gambar 8.4 Penginstalan *delta robo CPU* pada *delta hexapod*

4. Pasang konektor baterai dan *part C infrared receiver* pada *delta robo CPU.*
5. Hubungkan konektor *part C infrared receiver* ke *infrared receiver port* dari *delta robo CPU.* Lebih detail mengenai konfigurasinya dapat dilihat di Gambar 5.3 dan 5.4.
6. Unduh program dengan cara yang telah dijelaskan di subbab 1.3.
7. Gunakan *remote control* untuk mengendalikan robot.

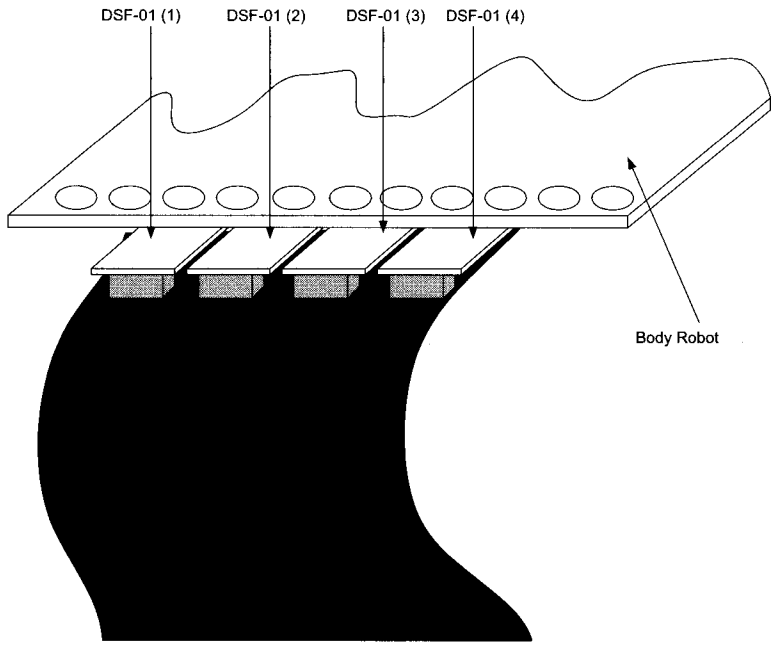


Gambar 8.5 Penginstalan *part C receiver* pada robot

BAB IX
ROBOT CERDAS PENJEJAK GARIS

Robot penjejak garis pada aplikasi ini berbeda dengan aplikasi-aplikasi sebelumnya. Pada aplikasi sebelumnya, robot penjejak garis tidak menggunakan komponen cerdas mikrokontroler dan hanya menggunakan gerbang logika, sedangkan pada aplikasi ini proses penjejak garis dilakukan oleh mikrokontroler sehingga fungsi-fungsi yang tidak dimiliki robot penjejak garis sebelumnya dapat ditambahkan.

Fungsi-fungsi tersebut di antaranya adalah *proportional integral derivative* (PID) yang mengatur agar gerakan robot lebih halus dan fungsi untuk mengikuti garis yang terputus-putus. Untuk fungsi PID, aplikasi penjejak garis ini menggunakan lebih dari dua sensor. Di sini digunakan empat buah DSF-01 V1, yaitu modul *single line follower* yang telah dibahas di Bab I. Empat buah modul ini diletakkan berjajar di bagian depan seperti yang tampak pada Gambar 9.1.

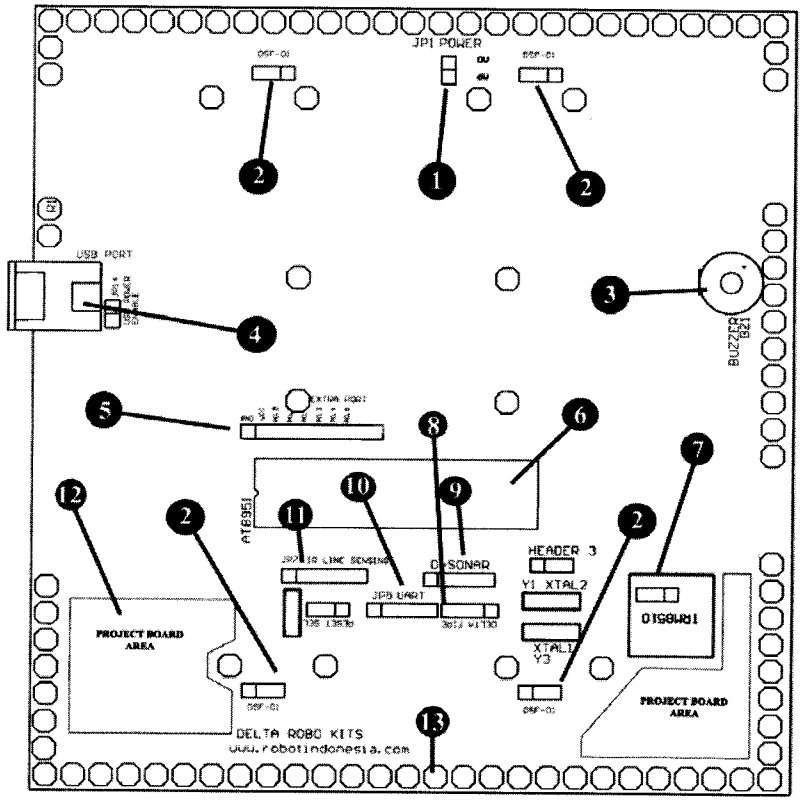


Gambar 9.1 Penginstalan DSF-01 pada robot

Dengan empat buah sensor ini robot dapat mengetahui seberapa besar simpangannya terhadap garis sehingga robot dapat memperbaiki posisi sebelum simpangan terlalu besar dan memperoleh gerakan yang efisien dan lebih halus. Sedangkan untuk kondisi jalur yang terputus-putus robot akan mengingat gerakan yang terakhir sehingga selama jalur yang terputus tersebut tidak berbelok, hal itu tidak akan mengganggu proses jalan robot penjejak garis. Deteksi terhadap garis putus-putus ini akan dibutuhkan pada kondisi sesungguhnya. Contohnya, meskipun garis penunjuk di lantai terputus, robot pembawa barang di gudang akan bergerak sesuai dengan garis tersebut selama tidak terjadi belokan.

Agar penginstalan sensor-sensor garis ini dapat dilakukan dengan mudah, kita akan menggunakan *delta robo kits* yang memiliki lubang-lubang sebesar 3,5 mm, yaitu lubang yang seukuran dengan *spacer* sehingga penempatan keempat sensor tersebut juga dapat dilakukan dengan mudah seperti pada Gambar 9.1.

Berbeda dengan kit robot sebelumnya, *delta robo kits* memiliki *onboard USB programmer* sehingga kita dapat menghubungkan robot langsung ke porta USB PC untuk keperluan pemrograman.

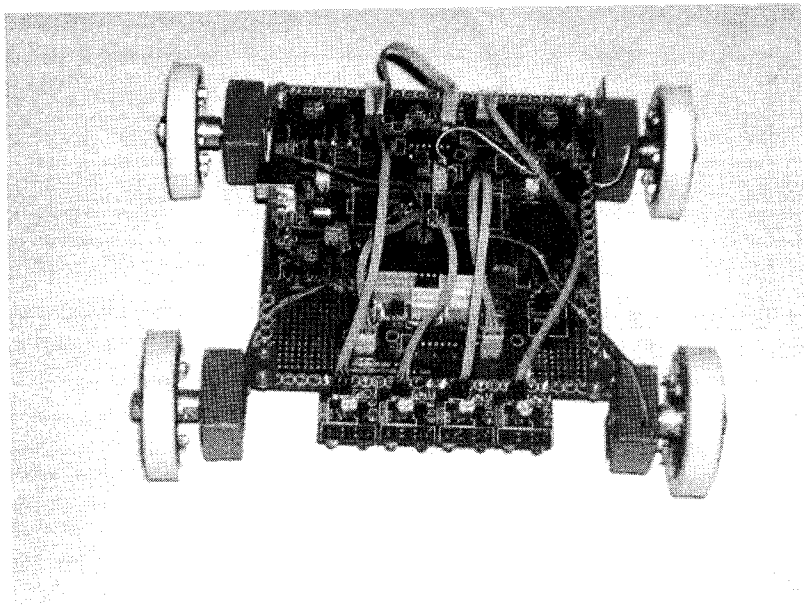


Gambar 9.2 Modul *delta robo kits*

Keterangan gambar:

1. *Power*, sumber daya robot dengan tegangan 5 V DC,
2. Porta DSF-01, untuk koneksi dengan modul *delta single line follower-01*,
3. *Buzzer*,
4. Porta USB, untuk mengunduh program ke otak robot (sistem mikrokontroler),

5. Porta ekstra, I/O ekstra untuk keperluan serbaguna,
6. Mikrokontroler AT89S51 (ATMega8515 opsional),
7. Modul *infrared receiver*, untuk menerima data dari *remote control*,
8. Porta *delta IR fire & target*, untuk kanon dan sensor pada aplikasi *robo warrior*,
9. Porta *D-sonar*, untuk aplikasi pengukuran jarak dengan ultrasonik,
10. Porta UART, untuk komunikasi dengan RS232 PC,
11. Porta *delta IR line sensing*, untuk koneksi dengan sungut pada *line follower*,
12. *Project board area*, bagian untuk menyolder rangkaian tambahan bila diperlukan, dan
13. Lubang *spacer*, bagian untuk memasang *spacer*, sungut, *soccer clamp* (aplikasi *robo soccer*), *bracket*, atau modul-modul lain.



Gambar 9.3 *Delta robo kits* dan empat DSF-01

Listing

```
#include<avr/io.h>
#include<util/delay.h>


#define LEFT_MOTOR              1
#define LEFT_ENABLE             2
#define RIGHT_MOTOR             4
#define RIGHT_ENABLE            8
#define SEN1            16
#define SEN2            32
#define SEN3            64
#define SEN4            128
#define MOTOR_DDR               DDRB
#define MOTOR_PORT              PORTB
#define MOTOR_PIN       PINB
#define LEFT_MOTOR_OUT
        MOTOR_DDR|=LEFT_MOTOR
#define LEFT_MOTOR_IN
        MOTOR_DDR&=~LEFT_MOTOR
#define LEFT_MOTOR_HIGH
        MOTOR_PORT|=LEFT_MOTOR
#define LEFT_MOTOR_LOW
        MOTOR_PORT&=~LEFT_MOTOR
#define LEFT_MOTOR_PIN
        (MOTOR_PIN&LEFT_MOTOR)>>>6
#define RIGHT_MOTOR_OUT
        MOTOR_DDR|=RIGHT_MOTOR
#define RIGHT_MOTOR_IN
        MOTOR_DDR&=~RIGHT_MOTOR
#define RIGHT_MOTOR_HIGH
        MOTOR_PORT|=RIGHT_MOTOR
#define RIGHT_MOTOR_LOW
        MOTOR_PORT&=~RIGHT_MOTOR
#define RIGHT_MOTOR_PIN
        (MOTOR_PIN&RIGHT_MOTOR)>>>6
#define LEFT_ENABLE_OUT
        MOTOR_DDR|=LEFT_ENABLE
```

```
#define LEFT_ENABLE_IN
        MOTOR_DDR&=~LEFT_ENABLE
#define LEFT_ENABLE_HIGH
        MOTOR_PORT|=LEFT_ENABLE
#define LEFT_ENABLE_LOW
        MOTOR_PORT&=~LEFT_ENABLE
#define LEFT_ENABLE_PIN
        (MOTOR_PIN&LEFT_ENABLE)>>>6
#define RIGHT_ENABLE_OUT
        MOTOR_DDR|=RIGHT_ENABLE
#define RIGHT_ENABLE_IN
        MOTOR_DDR&=~RIGHT_ENABLE
#define RIGHT_ENABLE_HIGH
        MOTOR_PORT|=RIGHT_ENABLE
#define RIGHT_ENABLE_LOW
        MOTOR_PORT&=~RIGHT_ENABLE
#define RIGHT_ENABLE_PIN
        (MOTOR_PIN&RIGHT_ENABLE)>>>6
#define SEN1_OUT        MOTOR_DDR|=SEN1
#define SEN1_IN         MOTOR_DDR&=~SEN1
#define SEN1_HIGH       MOTOR_PORT|=SEN1
#define SEN1_LOW        MOTOR_PORT&=~SEN1
#define SEN1_PIN        (MOTOR_PIN&SEN1)>>>6
#define SEN2_OUT        MOTOR_DDR|=SEN2
#define SEN2_IN         MOTOR_DDR&=~SEN2
#define SEN2_HIGH       MOTOR_PORT|=SEN2
#define SEN2_LOW        MOTOR_PORT&=~SEN2
#define SEN2_PIN        (MOTOR_PIN&SEN2)>>>6
#define SEN3_OUT        MOTOR_DDR|=SEN3
#define SEN3_IN         MOTOR_DDR&=~SEN3
#define SEN3_HIGH       MOTOR_PORT|=SEN3
#define SEN3_LOW        MOTOR_PORT&=~SEN3
#define SEN3_PIN        (MOTOR_PIN&SEN3)>>>6
#define SEN4_OUT        MOTOR_DDR|=SEN4
#define SEN4_IN         MOTOR_DDR&=~SEN4
#define SEN4_HIGH       MOTOR_PORT|=SEN4
#define SEN4_LOW        MOTOR_PORT&=~SEN4
#define SEN4_PIN        (MOTOR_PIN&SEN4)>>>6
```



```
unsigned char i;

void Mundur(void)
{
        LEFT_ENABLE_LOW;RIGHT_ENABLE_LOW;
        LEFT_MOTOR_LOW;RIGHT_MOTOR_LOW;
}

void Maju(void)
{
        LEFT_ENABLE_LOW;RIGHT_ENABLE_LOW;
        LEFT_MOTOR_HIGH;RIGHT_MOTOR_HIGH;
}

void Kiri(void)
{
        LEFT_ENABLE_LOW;RIGHT_ENABLE_LOW;
        LEFT_MOTOR_LOW;RIGHT_MOTOR_HIGH;
}

void Kanan(void)
{
        LEFT_ENABLE_LOW;RIGHT_ENABLE_LOW;
        LEFT_MOTOR_HIGH;RIGHT_MOTOR_LOW;
}

void Berhenti(void)
{
        LEFT_ENABLE_HIGH;RIGHT_ENABLE_LOW;
}

void pid_Init(int16_t p_factor, int16_t i_factor, int16_t
d_factor, struct PID_DATA *pid){
 pid->sumError = 0;
 pid->lastProcessValue = 0;
 pid->P_Factor = p_factor;
 pid->I_Factor = i_factor;
 pid->D_Factor = d_factor;
 pid->maxError = MAX_INT / (pid->P_Factor + 1);
```

```
 pid->maxSumError = MAX_I_TERM / (pid->I_Factor + 1);
}
int16_t pid_Controller(int16_t setPoint, int16_t processValue,
struct PID_DATA *pid_st)
{
 int16_t error, p_term, d_term;
 int32_t i_term, ret, temp;

 error = setPoint - processValue;
 if (error > pid_st->maxError){
  p_term = MAX_INT;
 }
 else if (error < -pid_st->maxError){
  p_term = -MAX_INT;
 }
 else{
  p_term = pid_st->P_Factor * error;
 }

 temp = pid_st->sumError + error;
 if(temp > pid_st->maxSumError){
  i_term = MAX_I_TERM;
  pid_st->sumError = pid_st->maxSumError;
 }
 else if(temp < -pid_st->maxSumError){
  i_term = -MAX_I_TERM;
  pid_st->sumError = -pid_st->maxSumError;
 }
 else{
  pid_st->sumError = temp;
  i_term = pid_st->I_Factor * pid_st->sumError;
 }

 d_term = pid_st->D_Factor * (pid_st->lastProcessValue -
processValue);

 pid_st->lastProcessValue = processValue;

 ret = (p_term + i_term + d_term) / SCALING_FACTOR;
```



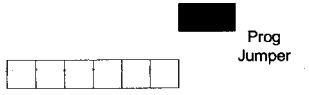
```
 if(ret > MAX_INT){
  ret = MAX_INT;
 }
 else if(ret < -MAX_INT){
  ret = -MAX_INT;
 }

 return((int16_t)ret);
}
void pid_Reset_Integrator(pidData_t *pid_st)
{
 pid_st->sumError = 0;
}


void main(void)
{
        Berhenti();
        pid_Init(1,1,1);
        while(1)
        {


        if((SEN1==1)&&(SEN2==0))&&((SENS3==1)&&(SE
NS4==1))
                {


        X=pid_Controller(1,SENS1,SENS2,SENS3,SENS4);
                        if(x>0)
                        {

                                Berhenti();
                                Kiri();
                        }
                }

        if((SEN1==0)&&(SEN2==1))&&((SENS3==1)&&(SE
NS4==1))
```

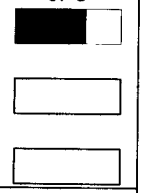
```
        {
    X=pid_Controller(1,SENS1,SENS2,SENS3,SENS4);
            {
                //Mundur();
                Berhenti();
                Kanan();
            }
        }
        if((SEN1==1)&&(SEN2==1)){Maju();}
    }
}
```

Berikut adalah langkah-langkah penginstalannya.

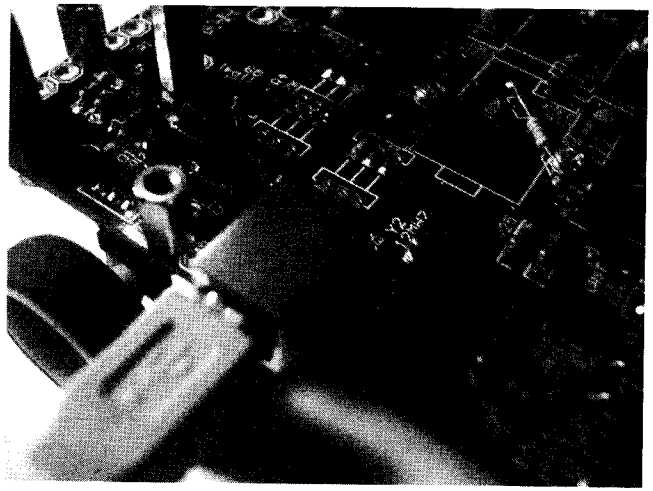
1. Siapkan empat buah DSF-01 seperti pada Gambar 9.1.
2. Hubungkan keempat konektor 3 pin dari DSF-01 ke empat porta DSF-01 seperti pada Gambar 9.3.
3. Buka perangkat lunak AVR Studio.
4. Gunakan mikrokontroler ATMega8515 dan atur *jumper* seperti tabel berikut.

TABEL 9.1 Pengaturan *jumper*

No Ref	Nama *Jumper*	Keterangan	Tata Letak
JP4	*Reset Select*	Memilih AT89S51 atau ATMega8515	
JP8	*Prog Jumper*	Dipasang pada saat pengisian program	

No Ref	Nama *Jumper*	Keterangan	Tata Letak
JP3	*XTAL Select*	Berfungsi untuk memilih kristal 11.0592 atau 8 MHz	
JP14	*USB Power Enable*	Untuk mode pemrograman dengan *power* dari USB PC (motor tidak boleh dalam keadaan aktif)	

5. Aktifkan *power supply* dengan menghubungkan konektor baterai.
6. Hubungkan porta USB PC ke porta USB *delta robo kits* dengan menggunakan kabel USB.

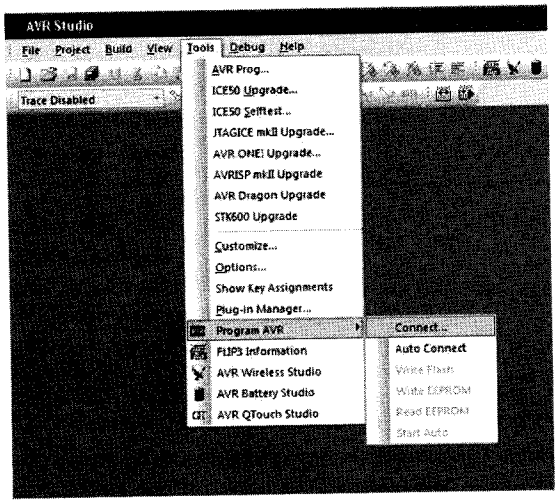


Gambar 9.4 Koneksi USB pada *delta robo kits*

7. Jika PC atau *notebook* tersebut baru pertama kali digunakan untuk melakukan pemrograman pada robot, terlebih dahulu sistem akan menanyakan *driver* pada saat *power supply* robot diaktifkan. Pilih

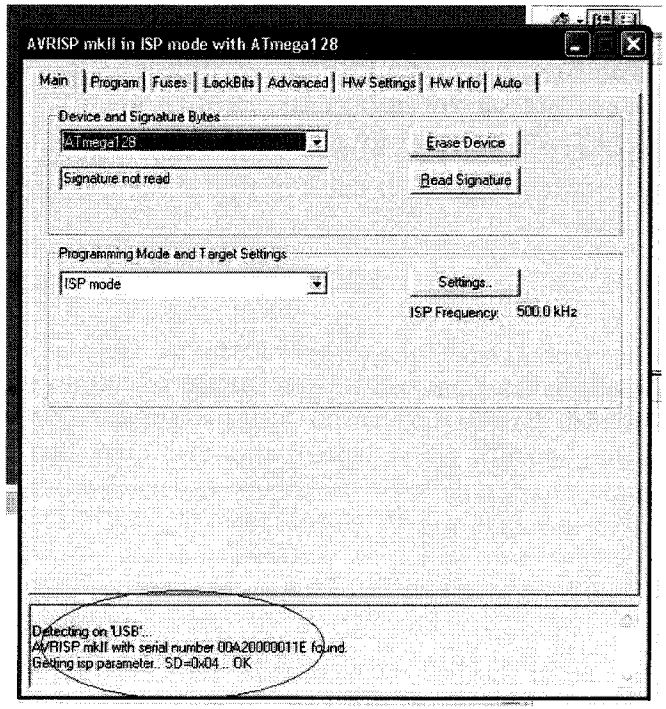
Search dan sistem akan mengarah pada *driver* yang sudah terbentuk saat penginstalan AVR Studio.

8. Buka program AVR Studio 4 dan pilih Auto Connect pada bagian Tools > Program AVR.



Gambar 9.5 Auto Connect pada AVR Studio

9. AVR Studio akan mendeteksi porta USB yang terhubung pada DU ISP secara otomatis dan sekaligus mendeteksi target yang terhubung.



Gambar 9.6 AVR Studio mendeteksi porta USB

10. Pilihlah mikrokontroler yang akan digunakan pada *field device and signature bytes*. Untuk *delta robo kit V2.0* mikrokontroler yang digunakan adalah AT89S51.
11. Pilih tab Program dan Load File Hex pada bagian Program.
12. Klik Program untuk mengunduh fail hex tersebut ke *delta robo kits*.
13. Fail hex yang diunduh adalah hasil kompilasi dari *listing* di atas. Untuk proses kompilasi, kita telah membahasnya di subbab 1.3.
14. Letakkan robot di jalur yang memiliki ketebalan garis lebih besar daripada area sensor (Gambar 9.1).
15. Robot akan berjalan mengikuti garis, walaupun ada garis yang terputus.

Latihan

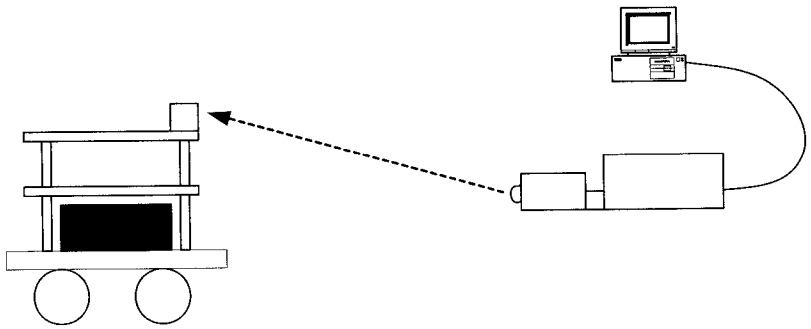
1. Coba tambahkan dua modul DSF-01 lagi pada robot dan modifikasi program agar gerakannya lebih halus!
2. Coba jalankan robot pada warna garis yang berbeda dan putar variabel resistor pada DSF-01 untuk mengatur sensitivitasnya!

BAB X

ROBOT DENGAN GERAKAN YANG TERPROGRAM OLEH PARAMETER

Pada aplikasi sebelumnya robot-robot bergerak berdasarkan perintah dari operator atau perubahan kondisi sensor. Namun, pada aplikasi ini robot bergerak berdasarkan ingatan yang ditanamkan terlebih dahulu ke salah satu bagian otak yang berfungsi sebagai media ingatan, yaitu memori. Proses ini berbeda dengan pemrograman robot. Bila proses pemrograman robot dianalogikan sebagai menanamkan atau mengajarkan sesuatu pada seseorang, proses unduh parameter ini dapat dianalogikan sebagai kegiatan memberikan perintah-perintah pada seseorang untuk diingat dan dilaksanakan.

Proses pemrograman robot dianalogikan dengan proses mengajari seseorang untuk mengemudi. Setelah bisa mengemudi, orang tersebut tidak perlu lagi diajari mengemudi. Sama halnya dengan robot, pada aplikasi ini kita cukup memprogramnya (subbab 1.3) dengan proses-proses gerakan atau aksi-aksi lainnya sekali saja. Selanjutnya, kita tinggal memberikan daftar tugas pada robot mengenai hal-hal yang harus dilakukan. Ini adalah analogi dari proses unduh parameter pada robot. Pada memori robot, program yang diunduh menempati lokasi yang disebut PEROM yang bersifat semipermanen atau tidak dapat diubah, kecuali dihapus dan diprogram ulang, sedangkan parameter ini akan menempati lokasi EEPROM yang dapat diubah-ubah dengan mudah dan tetap ada, walaupun *power* dimatikan. Parameter-parameter itu akan dikerjakan setelah robot menerima perintah untuk mulai bergerak, yaitu saat tombol start ditekan.



Gambar 10.1 Robot terprogram dengan parameter

Parameter-parameter tersebut dapat berupa gerakan maju, mundur, kiri, kanan, atau pengaturan kecepatan. Pada Gambar 10.1 tampak parameter-parameter yang telah diatur pada perangkat lunak di PC dikirimkan ke robot melalui media inframerah. Seperti pada aplikasi sebelumnya, pada aplikasi ini robot yang digunakan adalah *delta low cost programmable robot* yang terdiri atas

1. CPU,
2. *motor driver*,
3. motor dan mekanik roda,
4. enkoder,
5. *part A line follower*,
6. *limit switch*,
7. *infrared USB programmer*, dan
8. kabel RS232.

Parameter-parameter tersebut akan dikirimkan melalui *infrared USB programmer* ke robot dan tersimpan di dalam ingatan robot yang berupa EEPROM.

Listing berikut adalah program yang membuat robot dapat bergerak, mengatur kecepatan, dan menentukan seberapa jauh gerakan dilakukan. Selain mengatur gerakan, *listing* ini juga mengatur pengambilan data-data parameter yang dikirim oleh PC dan menyimpannya di dalam EEPROM.

Setelah semua parameter tersimpan di dalam EEPROM, robot akan bergerak berdasarkan parameter-parameter tersebut saat tombol start ditekan.

Listing

```
#include<avr/io.h>
#include"moving.h"
#include"delay.h"
#include"i2c.h"

#define HEADER      0x1E
#define ID     0x0E
#define NOM 0x01
#define START       PORTD.0
#define PROGPORTD.1

extern unsigned char rmt_data,rmt_nom;
unsigned char ser_buff[50],ser_index,ser_flag,stx,ptx;
unsigned char i,command_index;
unsigned int interval,duration;
unsigned char count;

void program(void);
void start(void);

void eep_write(unsigned int eep_addr,unsigned char dat)
{
      unsigned char page;

      page=0;
      page=eep_addr/256;
      page<<=1;
      page|=0xA0;
      I2C_start();
      I2C_addr(page);
      while(!I2C_check_ack());
      I2C_write_byte(eep_addr);
      while(!I2C_check_ack());
```

```
        I2C_write_byte(dat);
        while(!I2C_check_ack());
        I2C_stop();
        delay_5ms(2);
}

unsigned char eep_read(unsigned int eep_addr)
{
        unsigned char page;

        page=0;
        page=eep_addr/256;
        page<<=1;
        page|=0xA0;
        I2C_start();
        I2C_write_byte(page);
        while(!I2C_check_ack());
        I2C_write_byte(eep_addr);
        while(!I2C_check_ack());
        I2C_start();
        I2C_write_byte(page|=0xA1);
        while(!I2C_check_ack());
        return (I2C_read_byte());
        I2C_ack(1);
        I2C_stop();
}

void putchar(unsigned char c)
{
        while(!(UCSRA&(1<<UDRE)));
        UDR=data;
}

unsigned char getchar()
{
        unsigned char data, status;

        while(!(UCSRA&(1<<RXC))); // Wait for incomming
data
```



```
        status=UCSRA;
        data=UDR;
        return(data);
}

ISR(USART_RXC_vect)
{
        unsigned char status,data;
        status=UCSRA;
        ser_buff[ser_index]=UDR;
        ser_index++;
        if(ser_index>50)ser_index=0;
        if((ser_buff[ser_index-
3]==HEADER)&&(ser_buff[ser_index-
        2]==ID)&&(ser_buff[ser_index-1]==NOM))
        {
                stx=1;
                ser_index=3;
        }
        if((stx==1)&&(ser_index==6))
        {
        ptx=ser_buff[ser_index-1]+ser_index+1;
}
if((stx==1)&&(ser_index==ptx))
{
        ser_flag=1;
}
}

void start(void)
{
        for(i=0;i<command_index;i++)
        {
                count=0;
                duration=eep_read(i*5+3);
                duration<<=8;
                duration|=eep_read(i*5+4);
                ser_buff[0]=eep_read(i*5);
                ser_buff[1]=eep_read(i*5+1);
```

```
                interval=0;
                count=1;
                while(duration<interval)
                {

if((ser_buff[0]>0x80)&&(ser_buff[1]>0x80))maju();
        if((ser_buff[0]<0x80)&&(ser_buff[1]<0x80))mundur();

if((ser_buff[0]>0x80)&&(ser_buff[1]<0x80))belok_kanan();
if((ser_buff[0]<0x80)&&(ser_buff[1]>0x80))belok_kiri();
        }
        }
        count=0;
}

void program(void)
{
        while(1)
        {
                if(START==0)
                {
                        delay_5ms(1);
                        while(START==0);
                        start();
                }
                if(ser_flag==1)
                {

eep_write((command_index*5),ser_buff[ser_index-
        4]);

eep_write((command_index*5)+1,ser_buff[ser_index-
3]);

eep_write((command_index*5)+2,ser_buff[ser_index-
2]);
                eep_write((command_index*5)+3,interval>>8);
                eep_write((command_index*5)+4,interval);
                command_index++;
```



```
                ser_flag=0;
                stx=0;
                ptx=0;
                ser_index=0;
                interval=0;
                count=1;
                }
        }
}

void main(void)
{
        // 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=0x98;
        UCSRC=0x86;
        UBRRH=0x00;
        UBRRL=0x5F;
        command_index=0;
        while(1)
        {
                if(START==0)
                {
                        delay_5ms(1);
                        while(START==0);
                        count=0;
                        start();
                }
                if(PROG==0)
                {
                        delay_5ms(1);
                        while(PROG==0);
                        program();
                }
```

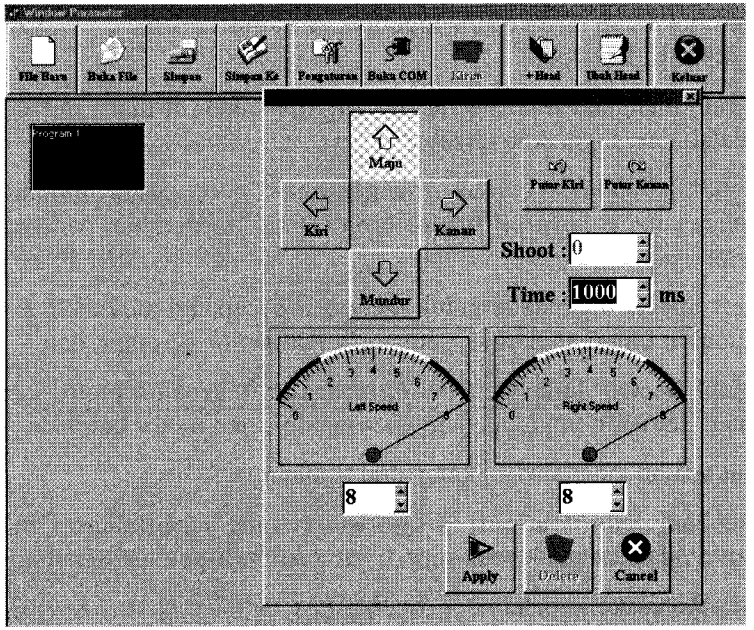
```
        }
    }
```

Program utama pada *listing* di atas hanyalah untuk mengecek tombol start atau program. Jika tombol start ditekan, program akan masuk ke prosedur start. Pada prosedur start, CPU akan membaca isi memori EEPROM yang telah diprogram terlebih dahulu, kemudian menjalankan perintah-perintah yang telah dibaca dari EEPROM tersebut.

Jika tombol program ditekan, CPU akan masuk ke prosedur program. Prosedur program akan menyimpan perintah-perintah yang dikirim dari komputer ke memori EEPROM untuk dijalankan pada saat tombol start ditekan. Pada saat CPU menjalankan perintah-perintah pada prosedur start, tombol program tidak akan berfungsi sampai CPU selesai menjalankan semua perintah-perintah yang terbaca dari EEPROM.

Prosedur yang berfungsi untuk membaca EEPROM adalah "eep_write(alamat memori,data yang akan ditulis)". Jadi, untuk menulis data ke EEPROM diperlukan dua variabel, yaitu alamat tempat data yang akan disimpan di memori EEPROM dan data yang akan diisikan. Sedangkan prosedur untuk membaca EEPROM adalah "eep_read(alamat memori yang dibaca)", untuk prosedur membaca EEPROM hanya diperlukan satu variabel, yaitu alamat memori yang akan dibaca.

Data-data yang akan diterima program CPU berupa kode dan nomor robot. Kode berisi perintah untuk maju, mundur, belok kanan, dan belok kiri, sedangkan nomor robotnya berupa nomor dan indeks robot. Hal ini dilakukan untuk keperluan pengembangan jika ingin memprogram beberapa robot secara bersamaan. Dengan adanya nomor dan indeks robot, robot bisa memilah-milah kode yang akan disimpan ke memori EEPROM-nya. Jika perintah tersebut bukan untuk robot, perintah-perintah itu akan diabaikan. Penentuan lamanya setiap perintah dilakukan dihitung oleh variabel interval. Penghitungan dilakukan dengan cara menghitung lamanya waktu dari satu perintah yang diterima ke perintah yang diterima selanjutnya.



Gambar 10.2 Window Parameter

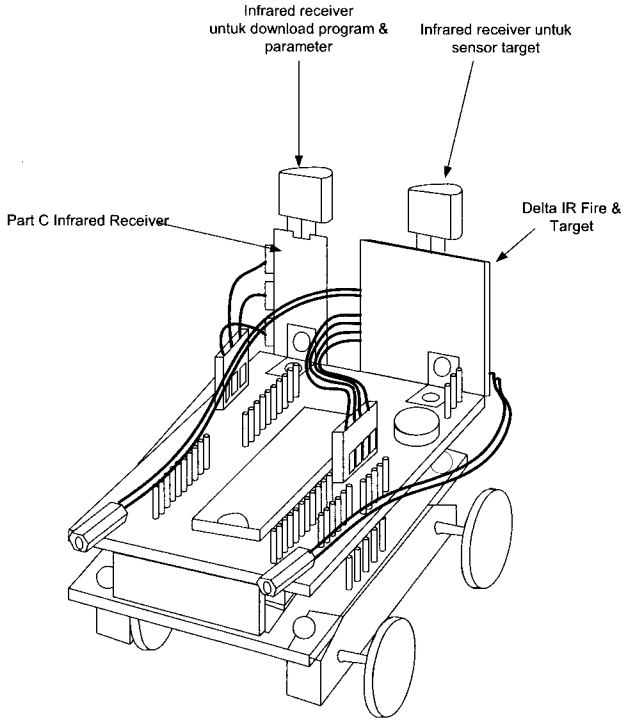
Gambar 10.2 adalah parameter-parameter yang dapat diatur oleh Window Parameter.

1. Tombol Maju, memajukan robot dengan menggerakkan kedua motor ke depan,
2. Tombol Mundur, memundurkan robot dengan menggerakkan kedua motor ke belakang,
3. Tombol Kiri, menggerakkan robot untuk berbelok ke kiri dengan mengatur agar motor kiri pada kecepatan nol dan motor kanan pada kecepatan penuh,
4. Tombol Kanan, menggerakkan robot untuk berbelok ke kanan dengan mengatur agar motor kanan pada kecepatan nol dan motor kiri pada kecepatan penuh,
5. Tombol Putar Kiri, menggerakkan motor kiri ke belakang dan motor kanan ke depan,

6. Tombol Putar Kanan, menggerakkan motor kanan ke belakang dan motor kiri ke depan,
7. Time, mengatur *delay* gerakan yang dilakukan di mana durasi *delay* adalah durasi gerakan tersebut,
8. Shoot, digunakan untuk mengaktifkan porta yang dihubungkan dengan *delta infrared fire & target* untuk menembak pada aplikasi *robo war*,
9. Left Speed, digunakan untuk mengatur kecepatan motor kiri di mana semakin tinggi angkanya, semakin cepat motor berputar, dan
10. Right Speed, digunakan untuk mengatur kecepatan motor kanan di mana semakin tinggi angkanya, semakin cepat motor berputar.

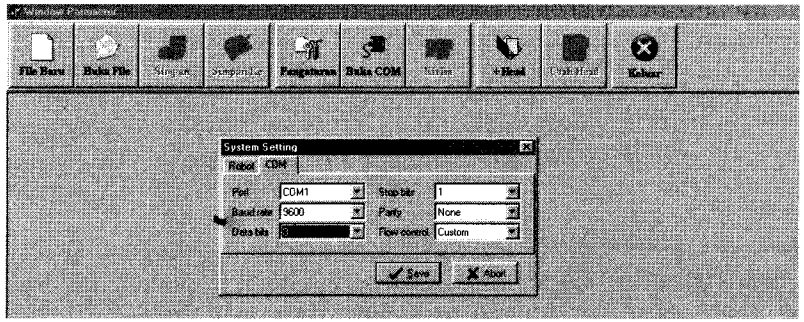
Berikut adalah langkah-langkah penginstalannya.

1. Siapkan *delta low cost programmer* yang terdiri atas CPU, *DC driver*, dan mekanik roda seperti pada aplikasi-aplikasi sebelumnya.
2. Aktifkan sakelar baterai.
3. Unduh program pada *listing* di atas seperti yang dijelaskan pada subbab 1.3.
4. Pasang *part C infrared receiver* dan hubungkan kabel-kabel *part C infrared receiver* seperti pada Gambar 5.3.
5. Hubungkan konektor 3 pin dari *part C infrared receiver* ke konektor 3 pin *infrared receiver port* dari *delta robo CPU.*
6. Pasang modul *delta IR fire & target* dan hubungkan konektor 4 pin ke konektor 4 pin di *delta robo CPU.*
7. Pasang LED inframerah ke *spacer* dua lubang dan letakkan di bagian depan robot.



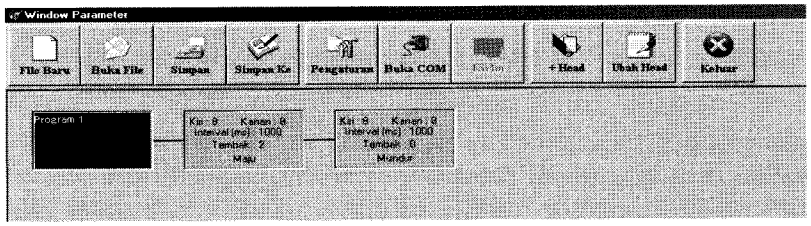
Gambar 10.3 Penginstalan sensor dan *delta IR fire & target*

8. Buka perangkat lunak Window Parameter yang ada pada CD.
9. Atur COM agar sesuai dengan *USB infrared transceiver.*
10. Atur Baud rate pada 600 bps, Parity None, dan Stop bits 1.



Gambar 10.4 Com Setting Window Parameter

11. Buatlah serangkaian perintah pada Window Parameter.



Gambar 10.5 Desain parameter pada Window Parameter

12. Klik *header* sehingga berwarna biru tua dan klik tombol Simpan ke agar Window Parameter mengirimkan parameter-parameternya ke robot.
13. Setelah parameter-parameter tersebut tersimpan di robot, tekan tombol start.
14. Robot akan bergerak sesuai parameter-parameter yang tersimpan, yaitu maju 1.000 mS dan menembak 2 kali, kemudian mundur 1.000 mS.

Latihan

1. Coba atur robot agar dapat bergerak 1 meter ke depan dan perhitungkan berapa *delay* yang dibutuhkan untuk gerakan tersebut!
2. Coba atur robot agar berputar 180 derajat dan perhitungkan berapa *delay* yang dibutuhkan!

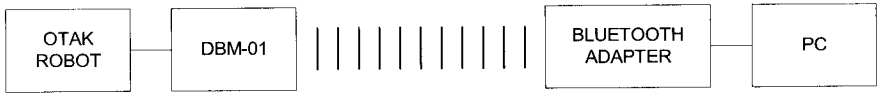
BAB XI

ROBOT YANG DIKENDALIKAN DENGAN BLUETOOTH

Pada aplikasi sebelumnya telah dibahas robot *teleoperated* yang dikendalikan melalui *remote*, baik *remote* berkabel maupun inframerah. Pada aplikasi ini robot dikendalikan melalui media nirkabel yang lebih efektif, yaitu Bluetooth. Untuk media inframerah dibutuhkan transmisi data yang segaris dan tidak terhalang, sedangkan untuk media Bluetooth komunikasi dapat dilakukan tanpa harus mengarahkan pemancar dan tidak terbatas oleh halangan.

Bluetooth adalah suatu teknologi nirkabel dengan lebar pita frekuensi 2,4 GHz dan biasa digunakan oleh perangkat-perangkat elektronik untuk saling berhubungan dalam jarak dekat. Komunikasi ini cukup aman karena Bluetooth memiliki enkripsi data, autentikasi pengguna, *fast frequency hopping*, dan *output power control*.

Proses komunikasi dilakukan dengan menggunakan protokol-protokol tertentu dan dapat dilakukan dalam jarak 10 meter. Untuk mempermudah proses komunikasi antara perangkat-perangkat yang menggunakan Bluetooth, seperti PC atau *notebook*, dengan robot secara nirkabel, kita menggunakan modul DBM-01 (*delta Bluetooth module*) yang berfungsi untuk mengubah protokol-protokol data Bluetooth menjadi data-data serial UART sehingga dapat diterima oleh porta UART otak robot.



Gambar 11.1 Komunikasi Bluetooth

Pada aplikasi ini robot yang digunakan adalah *delta robo kits* yang telah memiliki *onboard USB programmer* sehingga pemrograman dapat dilakukan secara langsung dengan menggunakan AVR Studio seperti

pada Bab IX, sedangkan mikrokontroler yang digunakan adalah AT89S51.

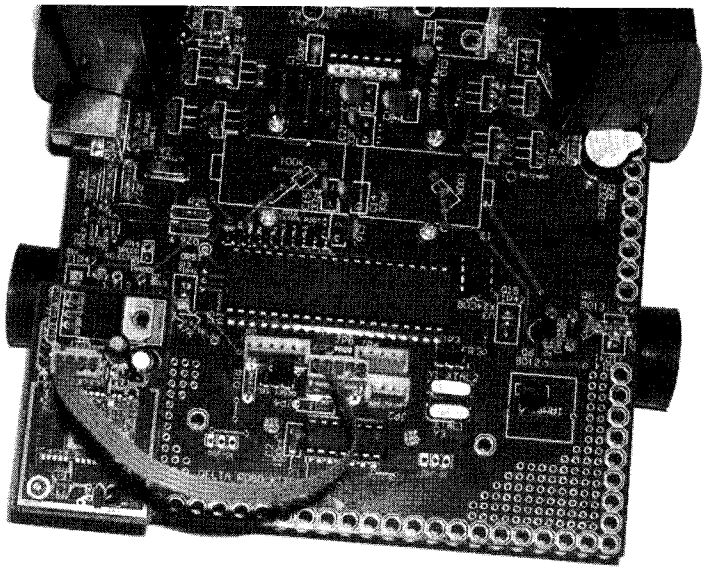
Pada aplikasi kali ini pemrograman robot akan kita lakukan dengan perangkat lunak pemrograman visual, yaitu Delta Robo Studio di mana pendesainan program tidak dilakukan dengan menuliskan instruksi-instruksi, namun dengan menggunakan gambar-gambar dan pengaturan parameter seperti yang telah dijelaskan pada subbab 1.3.

TABEL 11.1 Daftar bahan baku

Jumlah	Nama Barang	Kode
1	*Delta robo kits*	001-0034
1	*Delta Bluetooth modul interface*	070-0144
2	USB Bluetooth	

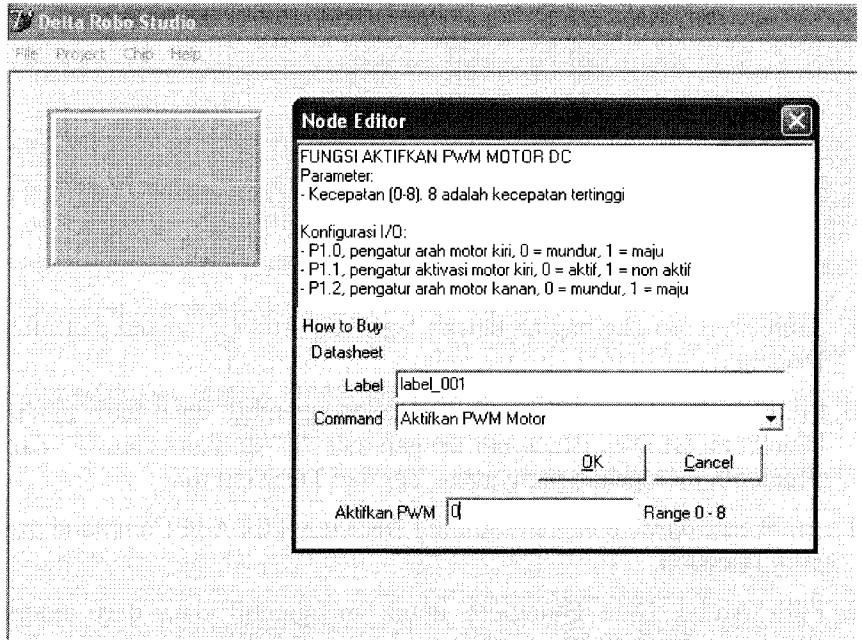
Berikut adalah langkah-langkah penginstalannya.

1. Pasang modul Bluetooth DBM-01 di *delta robo kits* seperti pada gambar berikut.



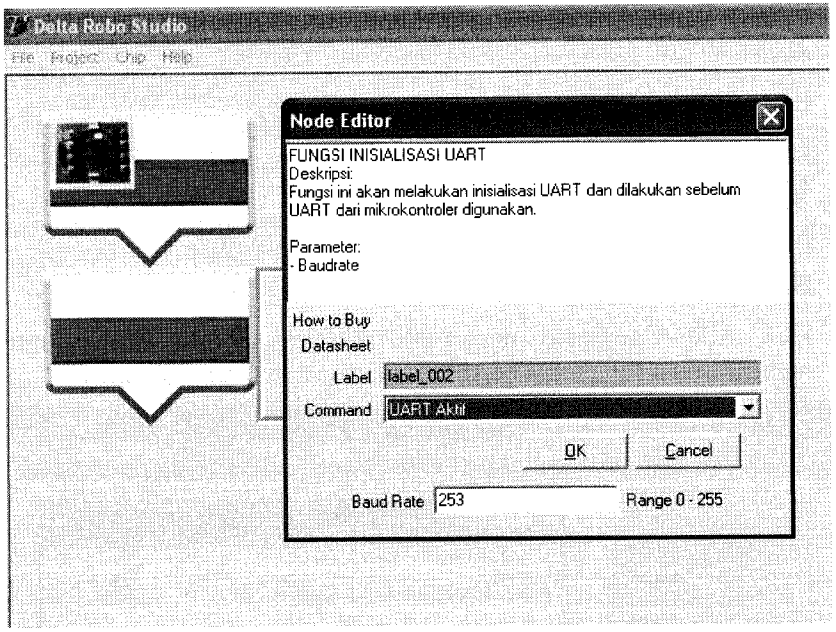
Gambar 11.2 Penginstalan *delta Bluetooth* pada *delta robo kits*

2. Pasang *spacer* penyangga agar robot tidak berlari saat diberi sumber daya.
3. Pasang *battery connector* ke *JP3 delta robo kits*.
4. Hubungkan porta USB PC/*notebook* dengan porta USB *delta robo kits* melalui kabel data.
5. Kopi folder Delta Robo Studio dari CD ke *hard disk*.
6. Buka perangkat lunak Delta Robo Studio.
7. Klik Add Command dan pilih Aktifkan PWM Motor untuk mengaktifkan fungsi PWM, dengan kecepatan 0 terlebih dahulu.



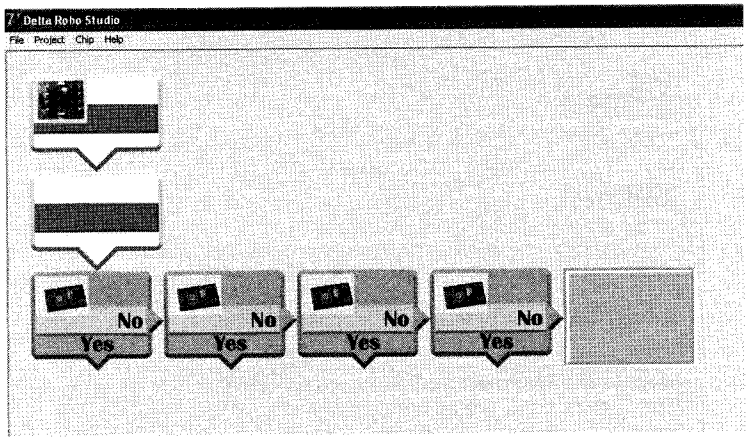
Gambar 11.3 Node Editor Aktifkan PWM pada Delta Robo Studio

8. Arahkan *mouse* ke bawah, klik Add Command dan pilih UART aktif untuk mengaktifkan porta UART *delta robo kits*. Nilai yang digunakan adalah 253 untuk *baud rate* 9600 (untuk informasi yang lebih mendetail perhitungannya, baca artikel *serial port* di bagian Teori Dasar di www.delta-electronic.com/article).



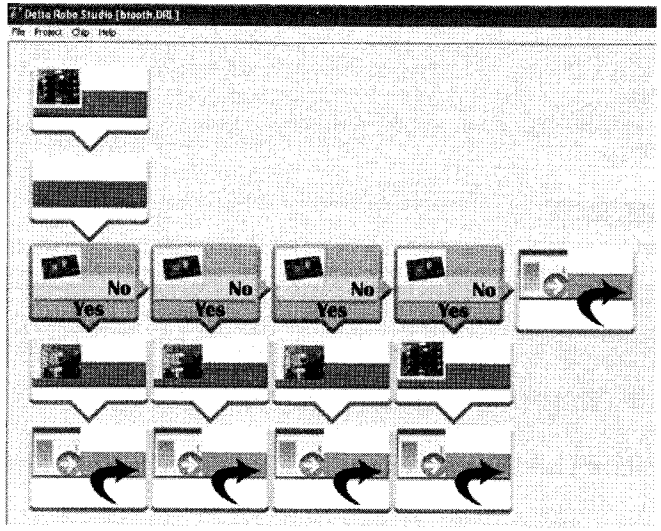
Gambar 11.4 Node Editor, aktifkan UART

9. Arahkan *mouse* ke bagian bawah lagi, klik Add Command dan pilih Decision.
10. Pilih Ambil Data Bluetooth untuk mengambil data dari modul Bluetooth. Isi Data Bluetooth dengan data "F" di mana data "F" dari Bluetooth merupakan pernyataan Yes dari Decision ini.
11. Arahkan *mouse* ke bagian samping Decision, klik Add Command dan pilih Decision.
12. Pilih Ambil Data Bluetooth untuk mengambil data dari modul Bluetooth. Isi Data Bluetooth dengan data "L" di mana data "L" dari Bluetooth merupakan pernyataan Yes dari Decision ini.
13. Lakukan hal yang sama di bagian samping Decision untuk data "I" dan "S" di mana "I" untuk belok kanan dan "S" untuk berhenti sehingga tampak seperti pada Gambar 11.5.



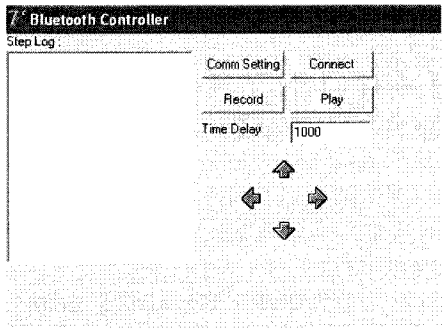
Gambar 11.5 Diagram alir yang dibentuk Delta Robo Studio

14. Klik Add Jump pada bagian samping Decision paling kanan dan pilih Jump ke Decision paling kiri (label_003).
15. Tambahkan perintah robot gerak maju untuk pernyataan Yes pada Decision "F", putar kiri pada Decision "L", putar kanan pada Decision "I", dan Stop pada Decision "S".
16. Kemudian, tambahkan *jump* ke label_003 di bawah masing-masing alur sehingga tampak seperti pada Gambar 11.6.



Gambar 11.6 Hasil akhir diagram alir

17. Simpan program dengan nama Btooth.drl.
18. Lakukan *assembly* hingga terbentuk fail Btooth.hex.
19. Unduh program Btooth.hex ke *delta robo kits* dengan menggunakan AVR Studio.
20. Lepaskan *spacer* penyangga dan kabel USB *delta robo kits*.
21. Pastikan komputer Anda sudah memiliki fungsi Bluetooth.
22. Buka perangkat lunak Bluetooth Controller.
23. Pilih Baud Rate 9600 dan Com sesuai dengan Bluetooth yang terdeteksi.
24. Klik Connect.
25. Tekan Maju dan robot akan bergerak maju hingga tombol dilepas, demikian pula halnya dengan Kiri dan Kanan.



Gambar 11.7 Tombol pengendali DBM-01

26. Dengan fungsi Record, gerakan-gerakan tombol dapat direkam dan diputar ulang dengan fungsi Play.

Latihan

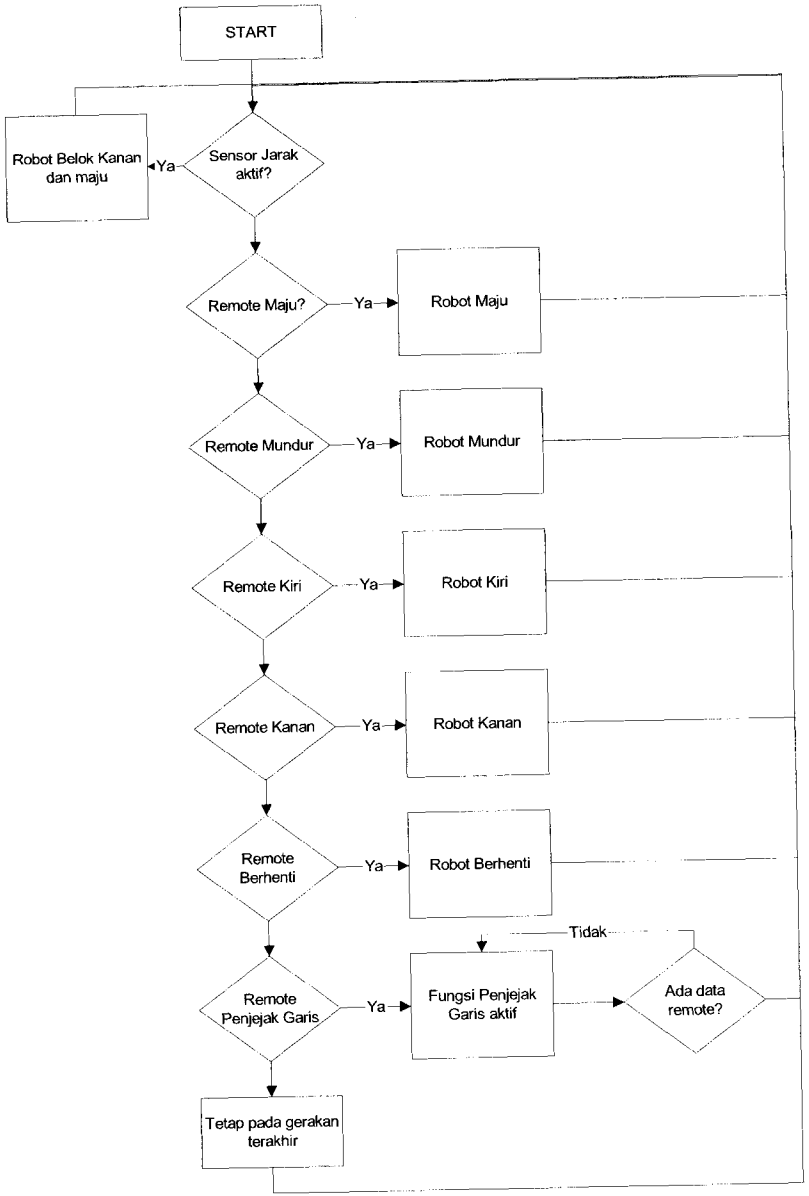
1. Jelaskan secara ringkas proses yang terjadi dari pembuatan program visual hingga pengunduhan program ke robot!
2. Jelaskan proses kendali robot dari perintah di PC hingga hal yang terjadi pada robot!

BAB XII

ROBOT CERDAS YANG DIKENDALIKAN DENGAN *REMOTE CONTROL*

Aplikasi ini merupakan gabungan robot yang bergerak berdasarkan kondisi sensor maupun perintah dari operator. Pada kondisi normal robot akan bergerak berdasarkan perintah dari *remote control*, namun pada saat terdapat halangan robot akan selalu berbelok ke kanan untuk menghindari halangan tersebut. Fungsi ini digunakan untuk mencegah robot dari benturan, walaupun operator melakukan kesalahan. Robot akan mengambil alih fungsi dan menghindar berdasarkan kondisi yang dideteksi sensor.

Pada penerapannya, aplikasi ini dapat digunakan untuk mengendalikan kendaraan dari jarak jauh dan mencegah operator melakukan kesalahan sehingga tidak terjadi tabrakan. Selain fungsi menghindari halangan, ada juga fungsi untuk mengikuti garis. Fungsi ini dilakukan saat robot menemui lintasan garis, operator dapat mengaktifkan fungsi penjejak garis sehingga robot akan bergerak mengikuti lintasan tersebut sampai operator kembali mengambil alih fungsi.



Gambar 12.1 Diagram alir robot cerdas dengan *remote*

Pada diagram alir di Gambar 12.1 saat robot tidak mendeteksi adanya halangan, robot akan bergerak sesuai kondisi perintah terakhir hingga perintah dari *remote control* diterima. Saat perintah dari *remote control* diterima, robot akan menganalisis perintah dan bergerak sesuai dengan perintah tersebut. Apabila tombol penjejak garis ditekan (tombol *power* di *remote* inframerah), robot akan bergerak mengikuti garis hingga operator menekan lagi tombol *remote* untuk mengambil alih fungsi. Hal ini dilakukan apabila ada sesuatu hal yang menyebabkan robot bergerak tidak sesuai dengan garis dan operator dapat membetulkan gerakannya.

Listing

```
/* Disain by Januar Susanto
#include<at89x51.h> //header file I/O mcs51

/* definisi sambungan dari CPU ke DSONAR */
#define USI  P3_2  //Trigger
#define USO  P3_3  //Echo

/*Definisi hubungan driver motor dan sensor ke mcs51 */
#define MOTOR_KIRI      P1_0
#define KIRI_NYALA      P1_1
#define MOTOR_KANAN     P1_2
#define KANAN_NYALA     P1_3
#define SENS1           P1_4
#define SENS2           P1_5

#define RMT_IN      P2_7    //pin micro yg terhubung ke
TSSOP

unsigned char i;
unsigned char count;

void delay(unsigned int time) //prosedur delay
{while(time--);}

void mundur(void) //prosedur robot mundur
{
      KIRI_NYALA=0;KANAN_NYALA=0;
      MOTOR_KIRI=0;MOTOR_KANAN=0;
```

```
}

void maju(void) //prosedur robot maju
{
        KIRI_NYALA=0;KANAN_NYALA=0;
        MOTOR_KIRI=1;MOTOR_KANAN=1;

}

void kiri(void) //prosedur robot belok kiri
{
        KIRI_NYALA=0;KANAN_NYALA=0;
        MOTOR_KIRI=0;MOTOR_KANAN=1;
}

void kanan(void) //prosedur robot belok kanan
{
        KIRI_NYALA=0;KANAN_NYALA=0;
        MOTOR_KIRI=1;MOTOR_KANAN=0;
}

void berhenti(void) //prosedur robot berhenati
{
        KIRI_NYALA=1;KANAN_NYALA=1;
}

void delay(unsigned int tunda)
{
        while(--tunda);
}

void cek_jarak() //prosedur mengecek jarak dengan ultrasonik
{
        TL1=0;
        TR1=1;
        while(TL1<12){USI=1;}
        TR1=0;
        USI=0;
        TH0=0;
```



```
        TL0=0;
        USO=1;
        while(!USO);
        TR0=1;
        while(USO);
        TR0=0;
}

void jejak_garis(void)
{
        While(1)
        {
                if(RMT_PIN==0)
                {
                        baca_remote();
                        if(rmt_nom==1)
                        {
                                if(rmt_data==0x29)break;
                        }
        if((SEN1==1)&&(SEN2==0))
                        {

        //while((SEN1!=1)&&(SEN2!=1))Mundur();
                                while(SEN2==0)
                                {
                                        //Mundur();
                                        Berhenti();
                                        Kiri();
                for(i=0;i<255;i++){if((SEN1==0)&&(SEN2==1
))break;}
                for(i=0;i<255;i++){if((SEN1==0)&&(SEN2==1
))break;}
                for(i=0;i<255;i++){if((SEN1==0)&&(SEN2==1
))break;}
                                }
                        }
                        if((SEN1==0)&&(SEN2==1))
                        {
```

```
                //while((SEN1!=1)&&(SEN2!=1))Mundur();
                        while(SEN1==0)
                        {
                                //Mundur();
                                Berhenti();
                                Kanan();

        for(i=0;i<255;i++){if((SEN1==1)&&(SEN2==0))break;
}

for(i=0;i<255;i++){if((SEN1==1)&&(SEN2==0))break;}

for(i=0;i<255;i++){if((SEN1==1)&&(SEN2==0))break;}
                        }
                }
                if((SEN1==1)&&(SEN2==1)){Maju();}
        }
}

/* main program          */
void main(void)
{
        TMOD=0x21 ;
        TCON=0x00 ;
        USO=0;
        berhenti();
        while(1)
        {
                cek_jarak();
                if(TH0<2)
                {
                        kanan(); // belok kanan
                                for(i=0;i<100;i++)delay(3000);
                }
                if(RMT_PIN==0)
                {
                        baca_remote();
                        if(rmt_nom==1)
                        {
```



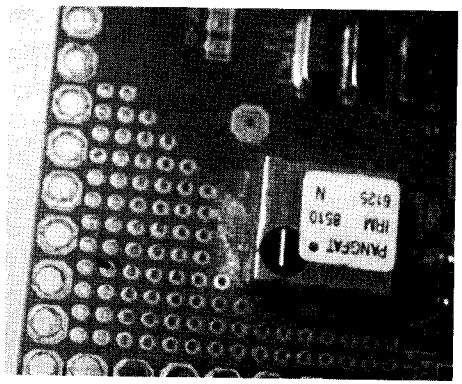
```
                                if(rmt_data==0x10)maju();
                                if(rmt_data==0x11)mundur();

                if(rmt_data==0x12)belok_kanan();
                                if(rmt_data==0x13)belok_kiri();
                                if(rmt_data==0x25)berhenti();

                if(rmt_data==0x28)jejak_garis();
                        }
                }

        }
}
```

TABEL 12.1 Daftar bahan baku

Jumlah	Nama Barang	Kode
1	*Delta robo kits*	001-0034
1	*Delta infrared line sensing*	006-0018
1	*D-sonar*	070-0006
1	IRM8510	011-0007

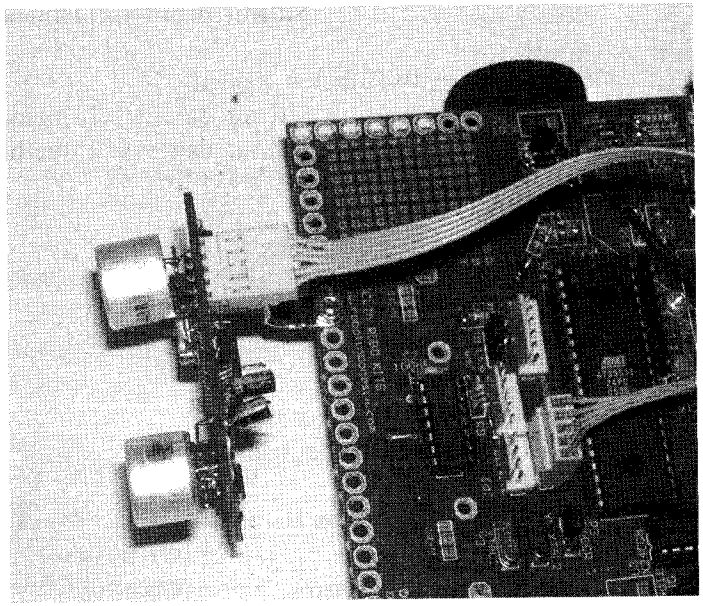
Berikut adalah langkah-langkah penginstalannya.

1. Pasang IRM8510 pada porta IRM8510 di *delta robo kits* seperti pada Gambar 12.2.



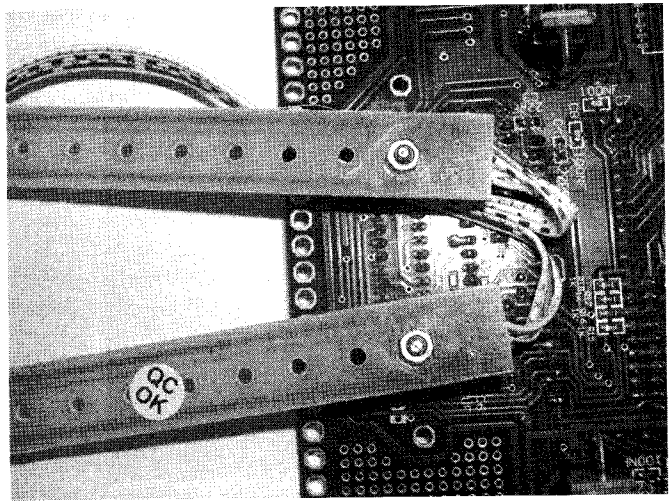
Gambar 12.2 Penginstalan sensor IRM8510

2. Pasang modul sensor jarak *D-sonar* seperti pada Gambar 12.3.



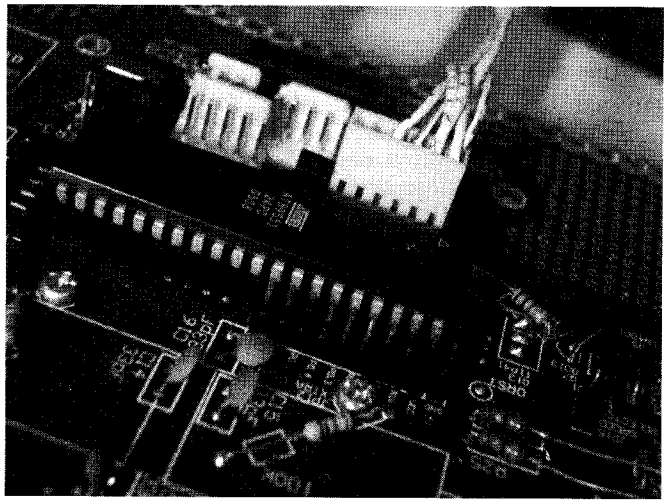
Gambar 12.3 Penginstalan *D-sonar* pada *delta robo kits*

3. Pasang sungut *delta infrared line sensing* seperti pada Gambar 12.4.



Gambar 12.4 Penginstalan *delta IR line sensing*

4. Pasang konektor 6 pin dari *delta infrared line sensing* ke *JP2 infrared line sensing port.*



Gambar 12.5 Koneksi *delta IR line sensing*

5. Pasang *battery connector* ke *JP3 delta robo kits.*
6. Hubungkan porta USB PC/*notebook* dengan porta USB *delta robo kits* melalui kabel data.
7. Unduh fail hex dari *listing* program di atas ke *delta robo kits* dengan menggunakan AVR Studio.
8. Jalankan robot dengan menggunakan perintah-perintah dari *remote control.*
9. Arahkan robot ke halangan dan robot harus berbelok ke kanan lalu lurus hingga menerima perintah dari *remote* atau terdapat halangan lagi.
10. Arahkan robot ke garis dan tekan tombol *power* (aktivasi penjejak garis).
11. Robot akan bergerak mengikuti garis hingga ada lagi perintah dari *remote control* yang mengambil alih fungsi.

Latihan

1. Sebutkan contoh-contoh penerapan aplikasi ini di kehidupan sehari-hari!
2. Cobalah merancang program dengan tambahan fungsi pendeteksi halangan saat fungsi penjejak garis aktif!

LAMPIRAN I
BAHAN-BAHAN PENUNJANG

Nama Bahan	Penjelasan
Hexapod low cost version	Mekanik robot kaki enam
DU-ISP V3.0	*Downloader* USB untuk MCS-51 / AVR
D-sonar	Pengukur jarak ultrasonik
Part C infrared receiver	Potongan *D-logo robotic* untuk penerima inframerah
RFM12B	*RF module* 433 MHz
RFM12BP	*RF module* 433 MHz 500 mW
TRF-2.4 GHz	*Transceiver* 2,4 GHz
Delta encoder	Sensor penghitung putaran motor
MOC703CY	*Opto interrupt*
Delta DC driver	*Dual H-bridge DC motor driver*
Delta robo CPU	Otak robot
Delta infrared fire & target	Modul meriam dan sensor sasaran inframerah
TSOP4838	Modul *infrared receiver* 38 kHz
TSAL4400	LED inframerah daya besar
D-joy controller	Pengendali empat buah *joystick* dengan antarmuka serial
DBM-01	Modul *Bluetooth to serial*
Delta single line follower	Sensor penjejak garis yang dapat dikalibrasi
Delta infrared line sensing	Sensor penjejak garis berupa "sungut"
Delta low cost line follower	Robot penjejak garis sederhana tanpa mikrokontroler
Part A line follower	Potongan *D-logo robotic* untuk sensor penjejak garis

Nama Bahan	Penjelasan
Robo war fire & target kit	Robot untuk aplikasi sasaran tembak
Delta low cost programmable robot	Robot cerdas yang dilengkapi sensor-sensor dan mikrokontroler yang dapat diprogram
D-logo remote	Bagian *remote control* dari *D-logo robotic*
Delta robo kits V2.1	Robot berbasis mikrokontroler yang dapat diprogram melalui USB
DU ISP infra	*Infrared USB programmer*
Part B DC motor	Bagian *D-logo robotic* untuk motor
Part D D-logo robotic	Bagian D dari *D-logo robotic*
Part E D-logo robotic	Bagian E dari *D-logo robotic*
Part F D-logo robotic	Bagian F dari *D-logo robotic*
Part G D-logo robotic	Bagian G dari *D-logo robotic*
Delta robo arm	Robot lengan dengan 5 sumbu gerakan
DSR-08	Modul pengendali 8 motor servo lewat serial
Delta hexapod crawler	Robot dengan mekanik berkaki enam
Delta hexapod avoider	Robot dengan mekanik berkaki enam dan ultrasonik
Delta robo war set	Paket robot untuk aplikasi perang robot
Delta micro tank	Robot tank mikro yang dikendalikan oleh RF 433 MHz
Delta giant hexapod crawler	Robot berkaki enam versi besar
Delta giant spider	Robot berkaki enam dengan 12 / 18 motor servo

LAMPIRAN II
PROSEDUR PENDUKUNG *LISTING* PROGRAM

Infrared
Infra.c

```
/*        InfraRed in and out library                                    */
 /*Protocol RC5 Sony remote control
                                                                        */
 /*Created by : januar
                                                                        */
 /*Email: januar@delta-electronic.com
                                                                        */
/*- infra data 5bit address, 7bit data                                  */
 /*- infra command n infra address define extrnal this file
                                                                        */



                    #include<avr/io.h>
                  #include<avr/interrupt.h>
                   #include<util/delay.h>
                    #include"infra.h"

                    void infra_in(void)
                                {
                      unsigned char i;

                      infra_cmd=0;
                      infra_address=0;
    TCCR1B=0x02; //turn on timer 1; 1.000.000 Khz @8Mhz crystal
                      TCNT1=0;
                    while(PIN_INFRA==0);
                  if(TCNT1>2400)
                                {
                    while(PIN_INFRA==1);
               for(i=0;i<7;i++)   //receive infra command (7bit)
                                {
```

```
                    TCNT1=0;
                  while(PIN_INFRA==0);
               if(TCNT1>1200) infra_cmd|=0x80;
                  while(PIN_INFRA==1);
                    infra_cmd>>=1;
                    }
          for(i=0;i<5;i++)  //receive infra address (5bit)
                    {
                    TCNT1=0;
                  while(PIN_INFRA==0);
               if(TCNT1>1200) infra_address|=0x80;
                  while(PIN_INFRA==1);
                    infra_address>>=1;
                    }
                   }
 infra_cmd&=0x7F;       //fix 7bit command to 8bit reading
  infra_address>>=2;       //fix 5bit address to 8bit reading
                infra_address&=0x1F;
     TCCR1B=0x00;                    //turn off timer
                  }

void infra_out(unsigned char cmd,unsigned char address)
                  {
                  unsigned char i;

                   cmd&=0x7F;
                  address&=0x1F;
                     cli();
                  TCCR1A=0x00;
                  TCCR1B=0x00;
                  TIMSK=0x00;
 TCCR1B=0x02; //turn on timer 1; 1.000.000 Khz @8Mhz
  TCCR0=0x02;  //turn on timer 0; 1.000.000 Khz @8Mhz
                    TCNT1=0;
                 while(TCNT1<2500)
                    {
                     TCNT0=0;
                  INFRA_LED_HIGH;
                   while(TCNT0<10);
```



```
                     TCNT0=0;
                  INFRA_LED_LOW;
                   while(TCNT0<10);
                    }
 TCCR0=0x00; //turn off timer 0
                    TCNT1=0;
                 while(TCNT1<700);
                  for(i=0;i<7;i++)
                    {
                   switch(cmd&0x01)
                        {
     case 0: TCCR0=0x02; //turn on timer 0;
1.000.000 Khz @8Mhz
                          TCNT1=0;
                        while(TCNT1<700)
                          {
                           TCNT0=0;
                        INFRA_LED_HIGH;
                         while(TCNT0<10);
                           TCNT0=0;
                        INFRA_LED_LOW;
                         while(TCNT0<10);
                          }
                    TCCR0=0x00; //turn off timer 0
                          break;
     case 1: TCCR0=0x02; //turn on timer 0;
1.000.000 Khz @8Mhz
                          TCNT1=0;
                        while(TCNT1<1300)
                          {
                           TCNT0=0;
                        INFRA_LED_HIGH;
                         while(TCNT0<10);
                           TCNT0=0;
                        INFRA_LED_LOW;
                         while(TCNT0<10);
                          }
                    TCCR0=0x00; //turn off timer 0
                          break;
```

```
                }
            TCNT1=0;
          while(TCNT1<700);
              cmd>>=1;
            }
        for(i=0;i<5;i++)
            {
         switch(address&0x01)
                {
    case 0: TCCR0=0x02; //turn on timer 0;
1.000.000 Khz @8Mhz
                    TCNT1=0;
                  while(TCNT1<700)
                      {
                        TCNT0=0;
                    INFRA_LED_HIGH;
                     while(TCNT0<10);
                        TCNT0=0;
                    INFRA_LED_LOW;
                     while(TCNT0<10);
                      }
                TCCR0=0x00; //turn off timer 0
                    break;
    case 1: TCCR0=0x02; //turn on timer 0;
1.000.000 Khz @8Mhz
                    TCNT1=0;
                  while(TCNT1<1300)
                      {
                        TCNT0=0;
                    INFRA_LED_HIGH;
                     while(TCNT0<10);
                        TCNT0=0;
                    INFRA_LED_LOW;
                     while(TCNT0<10);
                      }
                TCCR0=0x00; //turn off timer 0
                    break;
              }
            TCNT1=0;
```



```
            while(TCNT1<700);
              address>>=1;
              }
          _delay_ms(30);
            }
```

Infra.h

```
#ifndef _infra_h_INCLUDED_
#define _infra_h_INCLUDED_

#define INFRARED_LED 16
#define INFRARED_LED_OUT   DDRD|=INFRARED_LED
#define INFRARED_LED_IN    DDRD&=~INFRARED_LED
#define INFRARED_LED_HIGH  PORTD|=INFRARED_LED
#define INFRARED_LED_LOW   PORTD&=~INFRARED_LED
#define INFRA 4
#define INFRA_OUT      DDRD|=INFRA
#define INFRA_IN       DDRD&=~INFRA
#define INFRA_HIGH     PORTD|=INFRA
#define INFRA_LOW      PORTD&=~INFRA
#define PIN_INFRA      (PIND&INFRA)>>2

unsigned char infra_cmd,infra_address;

void infra_in(void);
void infra_out(unsigned char cmd,unsigned char address);


#endif
```

I2C

```
#include"i2c.h"
#include"delay.h"
#Include"avrio.h"

void I2C_init(void)
        {
        SDA=1;
        SCL=1;
```

```
}

void I2C_start(void)
{
 SDA=1;
 SCL=1;
delay(50);
 SCL=1;
 SDA=0;
delay(50);
 SCL=1;
}

void I2C_stop(void)
{
 SDA=0;
 SCL=1;
delay(50);
 SCL=1;
 SDA=1;
delay(50);
 SCL=1;
}

void I2C_addr(unsigned char addr)
{
unsigned char i;

 SDA=0;
 SCL=0;
delay(50);
for(i=0;i<8;i++)
{
 if(addr&(0x80>>i)) SDA=1;
else SDA=0;
 SCL=1;
delay(50);
 SCL=0;
delay(50);
```



```
}
SDA=1;
}

unsigned char I2C_read_byte()
{
unsigned char a;
unsigned char b;
unsigned char i;

a=0;
SDA=1;
SCL=0;
delay(50);
for(i=0;i<8;i++)
{
 SCL=1;
delay(50);
 b=SDA;
 a=a<<1;
 a=a|b;
 SCL=0;
delay(50);
}
SDA=1;
return a;
}

void I2C_write_byte(unsigned char d)
{
unsigned char i;

SDA=0;
SCL=0;
delay(100);
for(i=0;i<8;i++)
{
if(d&(0x80>>i)) SDA=1;
else SDA=0;
```

```
    SCL=1;
   delay(50);
    SCL=0;
   delay(50);
    }
  SDA=1;
 }

unsigned char I2C_check_ack()
        {
  unsigned char chk;
   unsigned char i;

    SCL=0;
    SCL=1;
 for(i=10;i>0;i--)
        {
    SDA=0;
   delay(50);
   chk=SDA;
 if(chk==0) break;
   delay(50);
    }
  SCL=0;
 delay(50);
  SDA=1;
if(chk==0) return 1;
  else return 0;
  }

void I2C_ack(unsigned char ans)
        {
  SCL=0;
  SDA=1;
 SDA=ans;
 delay(50);
  SCL=1;
 SDA=ans;
 delay(50);
```



```
 SCL=0;
 SDA=1;
 }
```

I2C.h

```
#ifndef _i2c_h_INCLUDED_
#define _i2c_h_INCLUDED_

#define SCL PORTC.1
#define SDA PORTC.0

 void I2C_init();
 void I2C_start();
 void I2C_stop();
void I2C_addr(unsigned char addr);
 unsigned char I2C_read_byte();
void I2C_write_byte(unsigned char d);
 unsigned char I2C_check_ack();
 void I2C_ack(unsigned char ans);


#endif
```

Moving.C

```
#include"moving.h"

unsigned char rmt_data,rmt_nom;

 void belok_kanan(void)
        {
  MTR_KIRI_NYALA=0;
 MTR_KANAN_NYALA=0;
  ARAH_MTR_KIRI=1;
  ARAH_MTR_KANAN=0;
 }

 void belok_kiri(void)
 {
```

```
MTR_KIRI_NYALA=0;
MTR_KANAN_NYALA=0;
ARAH_MTR_KIRI=0;
ARAH_MTR_KANAN=1;
}

void mundur(void)
{
MTR_KIRI_NYALA=0;
MTR_KANAN_NYALA=0;
ARAH_MTR_KIRI=0;
ARAH_MTR_KANAN=0;
}

void maju(void)
{
MTR_KIRI_NYALA=0;
MTR_KANAN_NYALA=0;
ARAH_MTR_KIRI=1;
ARAH_MTR_KANAN=1;
}

void berhenti(void)
{
MTR_KIRI_NYALA=1;
MTR_KANAN_NYALA=1;
}

void baca_remote(void)
{
unsigned char i;

rmt_data=0;
rmt_nom=0;
TCCR1B=0x02; //turn on timer 1; 1.000.000 Khz @8Mhz crystal
TCNT1=0;
while(PIN_INFRA==0);
if(TCNT1>2400)
{
```



```
while(PIN_INFRA==1);
for(i=0;i<7;i++)  //receive infra command (7bit)
{
TCNT1=0;
while(PIN_INFRA==0);
if(TCNT1>1200) rmt_data|=0x80;
while(PIN_INFRA==1);
rmt_data>>=1;
}
for(i=0;i<5;i++)  //receive infra address (5bit)
{
TCNT1=0;
while(PIN_INFRA==0);
if(TCNT1>1200) rmt_nom|=0x80;
while(PIN_INFRA==1);
rmt_nom>>=1;
}
}
rmt_data&=0x7F;      //fix 7bit command to 8bit reading
rmt_nom>>=2; //fix 5bit address to 8bit reading
rmt_nom&=0x1F;
TCCR1B=0x00;                  //turn off timer
}

void infra_out(unsigned char cmd,unsigned char address)
{

/*cmd&=0x7F;
address&=0x1F;
cli();
TCCR1A=0x00;
TCCR1B=0x00;
TIMSK=0x00;

TCCR1B=0x03;//turn on timer 1; 230.400Khz @14.7456Mhz
TCCR0=0x03;//turn on timer 0; 230.400Khz @14.7456Mhz
TCNT0=0;
TCNT1=0;
while(TCNT1<576)
```

```
                {
                TCNT0=0;
              LED_FIRE_HIGH;
               while(TCNT0<2);
                TCNT0=0;
              LED_FIRE_LOW;
               while(TCNT0<2);
                }


TCCR0=0x00; //turn off timer 0
              TCNT1=0;
          while(TCNT1<175);
            for(i=0;i<7;i++)
                {
              switch(cmd&0x01)
                    {
    case 0: TCCR0=0x02; //turn on timer 0;
1.000.000 Khz @8Mhz
                        TCNT1=0;
                     while(TCNT1<700)
                          {
                          TCNT0=0;
                       LED_FIRE_HIGH;
                       while(TCNT0<10);
                          TCNT0=0;
                       LED_FIRE_LOW;
                       while(TCNT0<10);
                          }
                TCCR0=0x00; //turn off timer 0
                          break;
    case 1: TCCR0=0x02; //turn on timer 0;
1.000.000 Khz @8Mhz
                        TCNT1=0;
                     while(TCNT1<1300)
                          {
                          TCNT0=0;
                       LED_FIRE_HIGH;
                       while(TCNT0<10);
```



```
                          TCNT0=0;
                       LED_FIRE_LOW;
                       while(TCNT0<10);
                          }
                TCCR0=0x00; //turn off timer 0
                          break;
                    }
              TCNT1=0;
          while(TCNT1<700);
              cmd>>=1;
                }
            for(i=0;i<5;i++)
                {
             switch(address&0x01)
                    {
    case 0: TCCR0=0x02; //turn on timer 0;
1.000.000 Khz @8Mhz
                        TCNT1=0;
                     while(TCNT1<700)
                          {
                          TCNT0=0;
                       LED_FIRE_HIGH;
                       while(TCNT0<10);
                          TCNT0=0;
                       LED_FIRE_LOW;
                       while(TCNT0<10);
                          }
                TCCR0=0x00; //turn off timer 0
                          break;
    case 1: TCCR0=0x02; //turn on timer 0;
1.000.000 Khz @8Mhz
                        TCNT1=0;
                     while(TCNT1<1300)
                          {
                          TCNT0=0;
                       LED_FIRE_HIGH;
                       while(TCNT0<10);
                          TCNT0=0;
                       LED_FIRE_LOW;
```

```
                                        while(TCNT0<10);
                                            }
                                    TCCR0=0x00; //turn off timer 0
                                        break;
                                    }
                                TCNT1=0;
                            while(TCNT1<700);
                                address>>=1;
                                }*/

/*TCCR1B=0x02; //turn on timer 1; 1.000.000 Khz @8Mhz
  TCCR0=0x02; //turn on timer 0; 1.000.000 Khz @8Mhz
                                TCNT1=0;
                            while(TCNT1<2500)
                                    {
                                    TCNT0=0;
                                LED_FIRE_HIGH;
                                while(TCNT0<10);
                                    TCNT0=0;
                                LED_FIRE_LOW;
                                while(TCNT0<10);
                                    }
                        TCCR0=0x00; //turn off timer 0
                                TCNT1=0;
                            while(TCNT1<700);
                                for(i=0;i<7;i++)
                                    {
                                    switch(cmd&0x01)
                                        {
                                case 0: TCCR0=0x02; //turn on timer 0;
                        1.000.000 Khz @8Mhz
                                                TCNT1=0;
                                            while(TCNT1<700)
                                                {
                                                TCNT0=0;
                                            LED_FIRE_HIGH;
                                            while(TCNT0<10);
                                                TCNT0=0;
                                            LED_FIRE_LOW;
```



```
                                            while(TCNT0<10);
                                                }
                                        TCCR0=0x00; //turn off timer 0
                                            break;
                                case 1: TCCR0=0x02; //turn on timer 0;
                        1.000.000 Khz @8Mhz
                                                TCNT1=0;
                                            while(TCNT1<1300)
                                                {
                                                TCNT0=0;
                                            LED_FIRE_HIGH;
                                            while(TCNT0<10);
                                                TCNT0=0;
                                            LED_FIRE_LOW;
                                            while(TCNT0<10);
                                                }
                                        TCCR0=0x00; //turn off timer 0
                                            break;
                                        }
                                    TCNT1=0;
                                while(TCNT1<700);
                                    cmd>>=1;
                                    }
                                for(i=0;i<5;i++)
                                    {
                                switch(address&0x01)
                                        {
                                case 0: TCCR0=0x02; //turn on timer 0;
                        1.000.000 Khz @8Mhz
                                                TCNT1=0;
                                            while(TCNT1<700)
                                                {
                                                TCNT0=0;
                                            LED_FIRE_HIGH;
                                            while(TCNT0<10);
                                                TCNT0=0;
                                            LED_FIRE_LOW;
                                            while(TCNT0<10);
                                                }
```

```
                TCCR0=0x00; //turn off timer 0
                        break;
        case 1: TCCR0=0x02; //turn on timer 0;
    1.000.000 Khz @8Mhz
                        TCNT1=0;
                    while(TCNT1<1300)
                            {
                            TCNT0=0;
                        LED_FIRE_HIGH;
                        while(TCNT0<10);
                            TCNT0=0;
                        LED_FIRE_LOW;
                        while(TCNT0<10);
                            }
                TCCR0=0x00; //turn off timer 0
                        break;
                    }
                TCNT1=0;
            while(TCNT1<700);
                address>>=1;
                }*/
            //_delay_ms(30);
                }
```

Moving.h

```
#ifndef _moving_h_INCLUDED_
#define _moving_h_INCLUDED_


  /* Koneksi MCU ke Motor
     ------------------------
      DIRR   PORTB.3
 DIRL   PORTB.2
      ENR    PORTB.0
      ENL    PORTB.1
    ------------------------*/

#define ARAH_MTR_KANAN              PORTB.2
#define MTR_KANAN_NYALA             PORTB.3
```



```
#define ARAH_MTR_KIRI               PORTB.0
  #define MTR_KIRI_NYALA          PORTB.1
#define RMT_PIN                     PORTC.7
  #define BZR                     PORTD.3


            void maju(void);
          void mundur(void);
        void belok_kiri(void);
      void belok_kanan(void);
        void berhenti(void);
      void baca_remote(void);
void infra_out(unsigned char cmd,unsigned char address);

            #endif
```

TENTANG PENULIS

Paulus Andi Nalwan lahir di Kediri, 5 April 1972. Lulus sarjana tahun 1997 dari Jurusan Teknik Elektro Sekolah Tinggi Teknik Surabaya (STTS). Saat ini bekerja sebagai desainer perangkat elektronik dan robotik berdasarkan pemesanan. Selain itu, menulis artikel-artikel di rubrik "Elektronika Populer" *Surabaya Post* dan juga di *Tabloid Komputek*. Menjadi rekoris MURI nomor 2.492 untuk "Robot Perang Teknologi Laser".

Karya-karyanya, antara lain, *Panduan Praktis Antarmuka dan Pemrograman Mikrokontroler AT89C51*, *Panduan Praktis Antarmuka dan Pemrograman LCD M1632*, dan *Membuat Robot Humanoid*.