BAB II

LANDASAN TEORI

2.1 Penelitian *Micromouse Robot* oleh Rusmini Setiawardhana dan M. Iqbal Nugrah (2011)

Robot Micromouse adalah robot cerdas yang dapat bergerak bebas di dalam sebuah labirin (*maze*) tanpa menyentuh objek sekitarnya, robot mengetahui ke arah mana harus bergerak, berapa derajat harus berputar jika menemui jalan buntu pada area labirin. Robot *Micromouse* ini termasuk kedalam jenis Robot *Mobile* yaitu *Autonomous Mobile Robot* dimana pengendalian gerakan dari robot yang berdasarkan program kemudi yang diberikan sehingga seolah-olah robot tersebut bergerak sendiri. Arah pergerakan *Mobile Robot* ini ditentukan ketika ada respon terhadap objek di depan, kanan dan kiri. Robot yang dibuat menggunakan mikrokontroler ATMEGA 8535 sebagai pengendali, sensor inframerah GP2D12 untuk mendeteksi adanya tembok atau tidak adanya tembok dan *driver* motor untuk menggerakkan motor sebagai aktuator. Robot ini menggunakan algoritma *backtracking* untuk mencari jalan terpendek dalam sebuah labirin ke tempat yang dituju. (Setiawardhana, 2011).

2.1.1 Prinsip Kerja *Search Mode*

Mode pencarian *(Search mode)* adalah proses untuk menemukan posisi *finish* dengan cara menelusuri labirin. Pada proses ini, robot akan bergerak secara berurutan dengan cara mengutamakan belok kanan bila menemukan persimpangan. Bila didalam perjalanan robot tidak menemukan belok kanan maka

robot akan lebih memilih jalan lurus. Dan bila robot tidak menemukan juga jalan lurus maka robot akan belok kiri. (Setiawardhana, 2011).

2.1.2 Prinsip Kerja Mode Kembali (*Return Mode*)

Return mode adalah proses robot berjalan kembali dari *finish* menuju *start* dengan jalur terpendeknya. Pada *return mode*, robot diharapkan sudah berjalan dengan kecepatan yang lebih dibandingkan dengan *search mode*. Ketika *return mode* dijalankan, apakah robot bisa mencapai *start* dengan jalur terpendek? Jika proses ini belum berhasil, maka terjadi kesalahan pada proses pemetaan. Dengan demikian, proses akan diulang pada *search mode* dan dievaluasi sampai tidak terjadi kesalahan. Prinsip dasar algoritma *backtracking* ini yaitu robot akan mengutamakan belok kanan apabila menemukan persimpangan dibanding lurus atau belok kiri. Bila tidak ada belok kanan maka robot cenderung untuk jalan lurus. (Setiawardhana, 2011).

2.2 Penelitian *Micromouse Robot* oleh Anita Nur Syafidtri (2010)

Robot Micromouse adalah robot cerdas yang dapat bergerak bebas di dalam sebuah labirin (*maze*) tanpa menyentuh objek sekitarnya, robot mengetahui ke arah mana harus bergerak, berapa derajat harus berputar jika menemui jalan buntu pada area labirin. Robot *Micromouse* ini termasuk ke dalam jenis Robot *Mobile* yaitu *Autonomous Mobile Robot*. *Autonomous*. *Mobile Robot* adalah pengendalian gerakan dari robot yang berdasarkan program kemudi yang diberikan sehingga seolah -olah robot tersebut bergerak sendiri. Robot ini dibangun atau dibuat dengan menggunakan mikrokontroler ATMEGA 128 sebagai pengendali, tiga buah sensor inframerah GP2D12 buatan SHARP untuk mendeteksi adanya tembok atau tidak adanya tembok,

dua buah *driver* motor yaitu SPC *Stepper Motor* dan dua buah *stepper motor* yang sumber tegangannya berasal dari satu buah baterai *Lithium Polymer*. Robot *Micromouse* memiliki dua mode yaitu mode eksplorasi dan mode hafalan. Mode eksplorasi robot akan melalui semua jalan yang ada pada labirin sedangkan mode hafalan robot akan bergerak sesuai dengan data yang tersimpan pada *Electrically Erasable Programmable Read-Only Memory* (EEPROM) mikrokontroler yang diperoleh dari mode eksplorasi. Robot ini menggunakan algoritma *Depth-First Search* untuk mencari jalan keluar dalam suatu labirin. Bahasa pemograman yang digunakan adalah bahasa C. Hasil pengujian, menunjukkan bahwa robot dapat menemui jalan keluarnya sendiri dengan beragam konfigurasi labirin, dengan asumsi titik mulai dan titik akhir tetap. (Syafidtri, 2010).

2.3 Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh Edsger W. Dijkstra yang merupakan salah satu varian bentuk algoritma popular dalam pemecahan persoalan yang terkait dengan masalah optimisasi dan bersifat sederhana. Algoritma ini menyelesaikan masalah mencari sebuah lintasan terpendek (sebuah lintasan yang mempunyai panjang minimum) dari *vertex* a ke *vertex* z dalam *graph* berbobot, bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh *node* negatif, namun jika terjadi demikian, maka penyelesaian yang diberikan adalah *infinity*. (Lubis, 2009).

Misalkan sebuah graf berbobot dengan *n* buah simpul dinyatakan dengan matriks ketetanggan $M = [m_{ij}]$, yang dalam hal ini memenuhi :

$[m_{ij}]$ = bobot sisi (i, j).

$[m_{ij}] = 0$.

$[m_{ij}] = \infty$, jika tidak ada sisi dari simpul *i* ke simpul *j*.

Selain matriks *M*, juga akan digunakan tabel $S = [s_i]$, yang dalam hal ini berlaku :

$[s_i] = 1$, jika dan hanya jika simpul *i* termasuk ke dalam lintasan terpendek.

$[s_i] = 0$, jika dan hanya jika simpul *i* tidak termasuk ke dalam lintasan terpendek dan keluarannya adalah sebuah tabel $D = [d_i]$, yang dalam hal ini menyatakan :

$[d_i]$ = panjang lintasan dari simpul awal sumber *a* ke simpul akhir tujuan *i*.

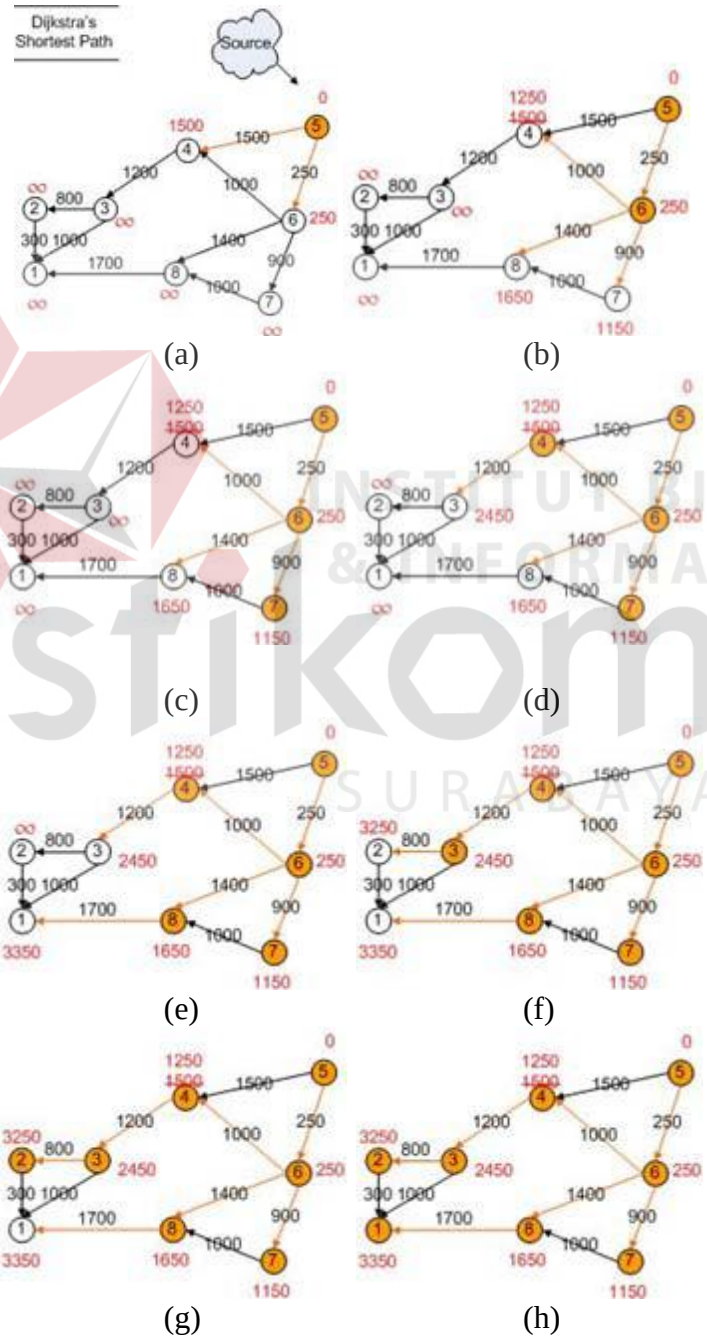
Dengan simpul awal adalah *a*, dan dengan jarak dari simpul *i* diartikan sebagai jarak antara simpul *a* dan *i*, maka algoritma Djikstra akan menginisialisasikan nilai jarak awal dan memperbaikinya tahap demi tahap.

Langkah-langkah algoritma Dijkstra dapat ditunjukkan sebagai berikut :

1. Inisialisasi suatu nilai jarak untuk setiap simpul dengan nilai nol untuk simpul awal dan nilai tak hingga untuk setiap simpul sisanya.
2. Seluruh simpul ditandai sebagai belum dikunjungi dan simpul *a* ditentukan sebagai simpul saat ini.
3. Untuk simpul saat ini, diperhitungkan seluruh tetangga langsung yang belum dikunjungi dan jarak *tentative* dikalkulasikan dari simpul *a*. Jika jarak yang didapatkan lebih kecil daripada jarak yang sudah dicatat sebelumnya, maka jarak yang minimum akan disimpan.
4. Jika telah selesai dengan pengecekan terhadap semua tetangga terdekat dari simpul saat ini, simpul ditandai sebagai sudah dikunjungi.
5. Sebuah simpul yang ditandai sebagai sudah dikunjungi tidak akan pernah diperiksa ulang dan jarak yang tercatat adalah akhir dan minimal.

6. Jika seluruh simpul sudah selesai dikunjungi, maka selesai, dan selain dari itu simpul dipilih dengan jarak minimum dari simpul a untuk ditetapkan sebagai simpul saat ini dan berulang seperti pada langkah c.

Langkah-langkah solusi pada algoritma Dijkstra dapat ditunjukkan seperti pada gambar 2.1.



Gambar 2.1 Langkah-langkah solusi pada algoritma Dijkstra.

Pseudo code algoritma Dijkstra dapat ditunjukkan sebagai berikut:

```
function Dijkstra(Graph, source):
for each vertex v in Graph:
// Initializations
dist[v] := infinity ;
// Unknown distance function from
source to v
previous[v] := undefined ;
// Previous node in optimal path from
source
end for ;
dist[source] := 0 ;
// Distance from source to source
Q := the set of all nodes in
Graph ;
// All nodes in the graph are
unoptimized - thus are in Q
while Q is not empty:
// The main loop
u := vertex in Q with
smallest dist[] ;
if dist[u] = infinity:
break ;
// all remaining vertices are
inaccessible from source
fi ;
remove u from Q ;
for each neighbor v of u:
// where v has not yet been removed
from Q.
alt := dist[u] +
dist_between(u, v) ;
if alt < dist[v]:
// Relax (u,v,a)
dist[v] := alt ;
previous[v] := u ;
fi ;
end for ;
end while ;
return dist[] ;
end Dijkstra.
```

(Handaka, 2010).

Algoritma Dijkstra untuk menentukan jalur terpendek dari suatu *graph*, maka akan menemukan jalur yang terbaik, karena pada waktu penentuan jalur yang dipilih, akan dianalisis bobot dari *node* yang belum terpilih, lalu dipilih *node* dengan bobot yang terkecil. Jika ternyata ada bobot yang lebih kecil melalui *node* tertentu maka bobot akan dapat berubah. Algoritma Dijkstra akan berhenti ketika semua *node* sudah terpilih, dan dengan algoritma Dijkstra ini dapat menemukan

jarak terpendek dari seluruh *node*, tidak hanya untuk *node* dari asal dan tujuan tertentu saja. (Lubis, 2009).

2.4 Pengolahan Citra

2.2.1 Kecerahan Gambar

Pengubahan kecerahan gambar bertujuan untuk membuat citra menjadi lebih terang atau lebih gelap. Kecerahan gambar dapat diperbaiki dengan menambah atau mengurangi setiap pixel pada citra dengan sebuah nilai konstan. Secara matematis operasi ini dapat ditunjukkan seperti pada persamaan 2.1.

$$f(x,y)' = f(x,y) + b \quad (2.1)$$

Jika b positif, kecerahan gambar bertambah, sebaliknya jika b negatif kecerahan gambar berkurang. (Munir, 2004)

Algoritma pengubahan kecerahan gambar dapat ditunjukkan sebagai berikut:

```
void ImageBrightness(citra Image, int N, int M, int b)
/* Mengubah kecerahan citra image yang berukuran N x M dengan
penambahan intensitas sebesar b.
*/
{     int i, j, n;
      for(i=0;i<=N-1;i++)
            for(j=0;j<=M-1;j++)
                  Image[i][j]+=b;
}
```

(Munir, 2004).

2.2.2 Konversi Citra *True* Color ke *Grayscale*

Citra *true color* dapat dikonversi menjadi *grayscale* menggunakan operasi titik. Salah satu tujuannya memudahkan citra untuk diproses kedalam

operasi-operasi pengolahan citra selanjutnya. Rumus yang digunakan untuk konversi ini dapat ditunjukkan seperti pada persamaan 2.2 (Basuki, 2005) :

$$Ko = \frac{R+G+B}{3} \quad (2.2)$$

Mata manusia memiliki 3 jenis sensor kerucut pada retina yang mendeteksi rentang warna yang berbeda pada spektrum cahaya yang tampak. Sensitifitas mata manusia terhadap warna berbeda-beda, seperti mata lebih sensitif pada warna hijau, kemudian warna merah, dan terakhir warna biru. Oleh karena itu, konversi *true color* ke *grayscale* lebih tepat dengan cara memberi bobot yang berbeda pada setiap elemen warna, alih-alih untuk meratakannya secara langsung. Dari modifikasi rumus di atas maka didapatkan seperti yang ditunjukkan pada persamaan 2.3 (Basuki, 2005).

$$Ko = wr.R + wg.G + wb.B \quad (2.3)$$

Bobot-bobot wr, wg, dan wb merupakan bobot untuk elemen warna merah, hijau, dan biru. *National Television System Committee* (NTSC) mendefinisikan bobot untuk konversi citra *true color* ke *grayscale* ini adalah sebagai berikut: wr = 0.299, wg = 0.587, dan wb = 0.144.

2.2.3 *Thresholding*

Contoh operasi titik berdasarkan intensitas adalah operasi pengambangan (*thresholding*). Pada operasi pengambangan, nilai intensitas pixel dipetakan ke salah satu dari dua nilai, α_1 atau α_2, berdasarkan nilai ambang (*threshold*) T dapat ditunjukkan seperti pada persamaan 2.4.

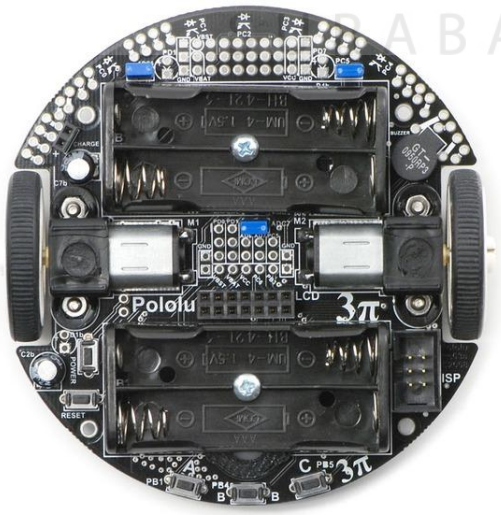
$$f(x,y)' = \begin{cases} \alpha_1, & \mathrm{f(x,y)} < T \\ \alpha_2, & \mathrm{f(x,y)} \geq \mathrm{T} \end{cases} \quad (2.4)$$

Jika $\alpha_1 = 0$ dan $\alpha_2 = 1$, maka operasi pengambangan mentransformasikan citra hitam-putih ke citra biner. Dengan kata lain, nilai intensitas *pixel* semula dipetakan ke dua nilai saja: hitam dan putih. Nilai ambang yang dipakai dapat berlaku untuk keseluruhan *pixel* atau untuk wilayah tertentu saja (berdasarkan penyebaran nilai intensitas pada wilayah tersebut).

(Munir, 2004).

2.5 *Mobile Robot*

Mobile robot yang digunakan yaitu robot komersil Pololu 3π. Pololu 3π adalah sebuah robot berukuran kecil, memiliki performa tinggi, didukung oleh 4 buah baterai AAA sebagai catu daya dan memiliki *power system* yang unik yang dapat menggerakkan motor dengan tegangan 9.25 Volt teregulasi. 3π mampu mencapai kecepatan hingga 100 cm/s (*centimeter per second*) saat bergerak dengan tegangan baterai yang tidak berubah, hal ini mengakibatkan kinerja yang konsisten. 3π robot seperti yang ditunjukkan pada gambar 2.2.

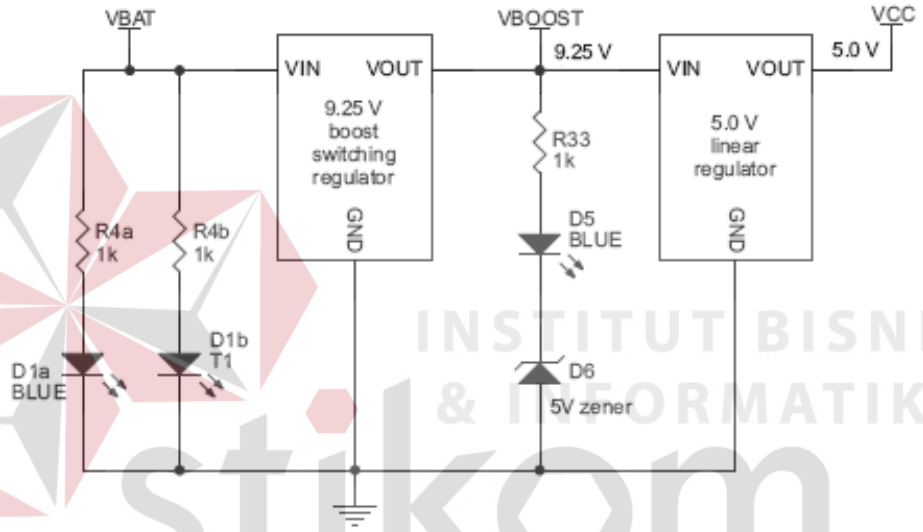


Gambar 2.2 Pololu 3π *robot*.

3π berbasis mikrokontroler ATMega328 yang memiliki fitur memori program sebesar 32 KB, *Random Accerss Memory* (RAM) sebesar 2 KB, dan EEPROM sebesar 1 KB.

2.3.1 *Power Management*

Skematik *power management* pada 3π dapat ditunjukkan seperti pada gambar 2.3.



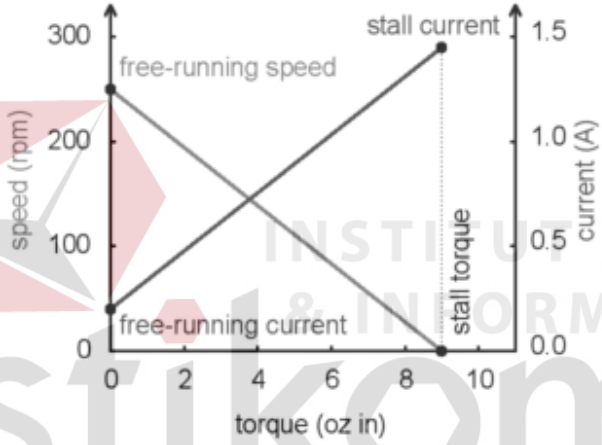
Gambar 2.3 *Power management* pada 3π.

Tegangan 4 buah baterai AAA dapat bervariasi antara 3.5 – 5.5 Volt (V) dan bahkan 6 V jika *alkaline* yang digunakan. Berarti tidak memungkinkan untuk meregulasi tegangan menjadi naik atau turun menjadi 5 V. Tetapi tidak pada 3π, *switching regulator* menaikkan tegangan terlebih dahulu menjadi 9.25 V (Vboost), kemudian *linear regulator* meregulasi Vboost turun menjadi 5 V (VCC). Vboost digunakan untuk menggerakkan motor sedangkan VCC digunakan untuk mikrokontroler dan semua sinyal digital.

2.3.2 Motor dan *Gearboxes*

Motor merupakan sebuah mesin yang mengubah energi listrik menjadi energi gerak. Ada banyak berbagai jenis motor, tetapi yang paling banyak digunakan untuk robot dengan biaya rendah yaitu *brushed DC motor* yang merupakan jenis motor pada 3π.

Tiap motor memiliki kecepatan maksimal jika tidak ada gaya yang diterapkan dan torsi maksimal jika motor benar-benar berhenti, karekteristik motor tersebut dapat ditunjukkan pada grafik seperti pada gambar 2.4.



Gambar 2.4 Karakteristik motor 3π.

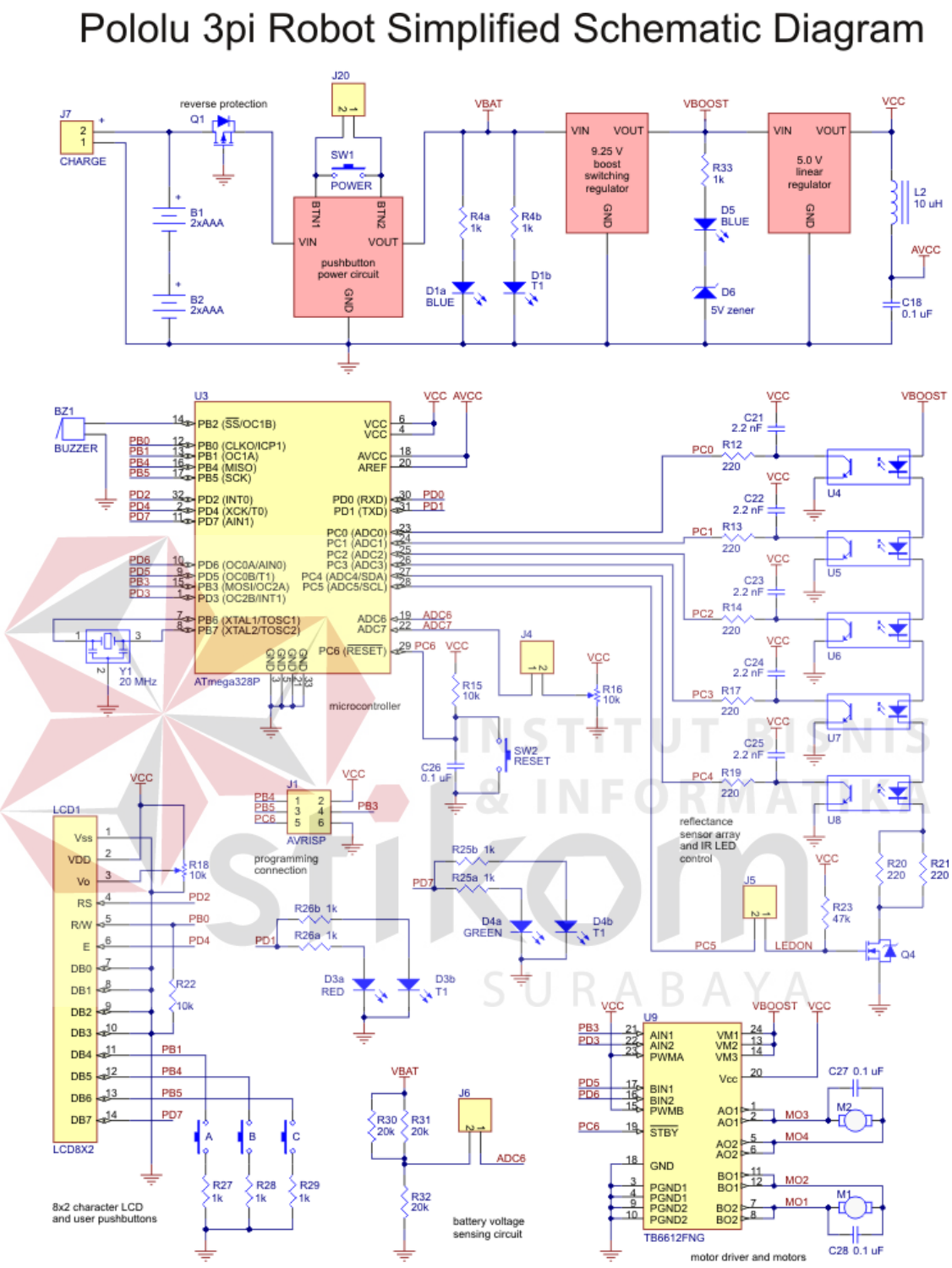
Kecepatan bebas motor biasanya ribuan *Revolutions per Minute* (RPM), terlalu cepat dari kecepatan yang diinginkan. Gearbox merupakan sebuah sistem roda gigi yang mengubah *output* kecepatan tinggi dengan torsi rendah menjadi *output* kecepatan lebih rendah dengan torsi yang tinggi, lebih cocok digunakan untuk mengatur gerak laju robot. Rasio roda gigi yang digunakan pada 3π adalah 30:1, berarti bahwa untuk setiap 30 putaran poros motor, poros output akan berputar sekali, mengurangi kecepatan dengan faktor 30, dan idealnya meningkatkan torsi dengan faktor 30. Parameter yang dihasilkan dari motor 3π dirangkum seperti pada tabel 2.1.

Tabel 2.1 Parameter motor pada 3π.

Parameter	Nilai
Gear Ratio	30:1
Free-running speed	700 rpm
Free-running current	60 mA
Stall torque	6 oz·in
Stall current	540 mA

Dua roda 3π masing-masing memiliki radius 0.67 inci, yang berarti bahwa gaya maksimum yang dapat dihasilkan dengan dua motor saat berkendara ke depan adalah 2 × 6/0.67 = 18 oz. 3π beratnya sekitar 7 ons dengan baterai, sehingga motor cukup kuat untuk mengangkat 3π mendaki lereng vertikal atau mempercepatnya pada 2 kali percepatan gravitasi. Kinerja aktual dibatasi oleh gesekan ban, di lereng yang cukup curam roda akan tergelincir sebelum berhenti. Hal ini terjadi pada kemiringan sekitar 30° - 40°.

Rangkaian elektronika 3π dapat ditunjukkan seperti pada gambar 2.5.



Gambar 2.5 Rangkaian elektronika 3π.

(Pololu).

2.6 Mikrokontroler ATMEGA644P-TQFP44

ATMega644P merupakan mikrokontroler 8-bit *Alf and Vegard's Risc processor* (AVR) dengan arsitektur *Reduced Instruction Set Computing* (RISC). Dengan mengeksekusi instruksi dalam satu *clock* tunggal, ATMega644P mencapai *throughput* mendekati 1 *Milions Instruction Per Second* (MIPS) per Mega Hertz (MHz) memungkinkan perancang sistem untuk mengoptimalkan konsumsi daya dibandingkan kecepatan proses.

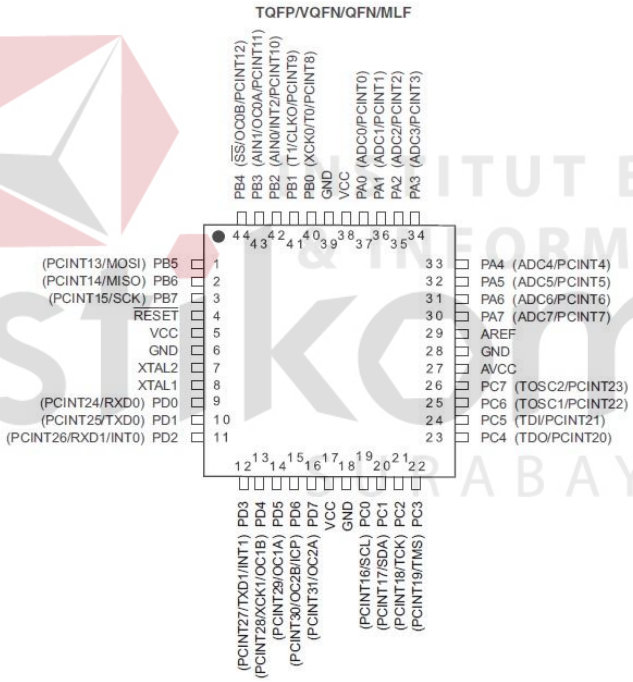
Inti AVR mengkombinasikan *rich instruction set* dengan 32 *general purpose working registers*. Semua 32 register secara langsung terhubung ke *Arithmetic Logic Unit* (ALU), memungkinkan dua register independen diakses untuk diakses dalam satu instruksi yang dieksekusi dalam satu siklus *clock*. Kelebihan dari arsitektur tersebut yaitu kode yang dihasilkan menjadi lebih efisien sementara mencapai *throughtput* hingga sepuluh kali lebih cepat daripada mikrokontroler *Complex Instruction Set Computing* (CISC) konvensional.

Di dalam mikrokontroler ATMega644P sudah terdiri dari:

1. *Central Processor Unit* (CPU) yang terdiri dari 32 buah *register*.
2. 131 *instruksi* yang umumnya hanya membutuhkan 1 siklus *clock*.
3. 32 *saluran* Input/Output (I/O) yang dapat diprogram.
4. *Analog to Digital Converter* (ADC) 10 bit sebanyak 8 *channel*.
5. Dua *buah* 8 bit dan sebuah 16 bit Timer/Counter.
6. *Watchdog timer* dengan osilator *internal*.
7. *Tegangan* operasi 1.8 V – 5.5 V.
8. Internal SRAM sebesar 4 *Kilo Byte* (KB).
9. Memori *Flash* sebesar 64 KB dengan kemampuan *Read While Write*.

10. *EEPROM* sebesar 2 KB yang dapat diprogram saat operasi.
11. *Unit* interupsi *internal* dan *external*.
12. *Master*/*Slave* SPI *Serial Interface*.
13. *Antarmuka* komparator *analog*.
14. 6 *channel Pulse Width Modulation* (PWM).
15. *Hampir* mencapai 20 MIPS pada Kristal 20 MHz.
16. 2 *port* USART *programmable* untuk komunikasi serial.

Susunan kaki standar 44 pin TQFP *microcontroller* AVR ATMega644P dapat ditunjukkan seperti pada gambar 2.6.



Gambar 2.6 Susunan kaki *microcontroller* ATMega644P.

Berikut penjelasan umum susunan kaki ATmega644P:

1. VCC merupakan pin masukan positif catu daya. Pada ATMega644P memerlukan catu daya sebesar 1,8 V – 5,5 V.
2. GND sebagai pin *ground*.

3. Port A (PA0..PA7) merupakan pin I/O dua arah dan pin fungsi khusus yaitu sebagai *input* ADC dan *pin change interrupt*.
4. Port B (PB0..PB7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu *Timer/Counter*, *analog* c*omparator*, SPI, dan *pin change interrupt*.
5. Port C (PC0..PC7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu TWI (*Two Wire Interface*), *analog comparator*, *Timer Osilator*, dan *pin change interrupt.*
6. Port D (PDO..PD7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu *analog comparator*, *external interrupt* dan *serial communication*.
7. RESET merupakan pin yang digunakan untuk me-*reset microcontroller*.
8. XTAL1 dan XTAL2 sebagai I/O *clock* eksternal. Suatu *microcontroller* membutuhkan sumber detak (*clock*) agar dapat mengeksekusi instruksi yang ada di memori. Semakin tinggi nilai kristalnya, maka semakin cepat pemrosesan instruksi *microcontroller* tersebut.
9. AVCC sebagai tegangan *input* untuk ADC.
10. AREF sebagai tegangan referensi eksternal untuk ADC.

(Atmel, 2010)

2.4.1 *Analog to Digital Converter* (ADC)

AVR memiliki fitur ADC sebanyak 8 bit *channel* dengan resolusi 10 bit *register* yang digunakan, untuk *setting* ADC ini sebagai berikut:

A. ADC *Control and Status Register A* (ADCSRA)

Bit-bit konfigurasi pada *register* ADCSRA dapat ditunjukkan seperti pada tabel 2.2.

Tabel 2.2 *Register* ADCSRA.

Bit	7	6	5	4	3	2	1	0
Name	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Keterangan:

- ADEN : 1 = ADC *enable*, 0 = ADC *disable*.
- ADSC : 1 = mulai konversi, 0 = konversi belum terjadi.
- ADATE : 1 = *auto trigger* diaktifkan, *trigger* berasal dari sinyal yang dipilih (set pada *register* SFIOR *bit* ADTS). ADC akan *start* konversi pada *edge* positif sinyal *trigger*.
- ADIF : diset ke 1, jika konversi ADC selesai dan data *register* ter-*update*. Namun, ADC *Conversion Complete Interrupt* dieksekusi jika *bit* ADIE dan *bit*-I dalam *register* SREG di-*set*.
- ADIE : di-*set* jika *bit*-I dalam SREG di-*set*.
- ADPS[2..0] : *Bit* pengatur *clock* ADC, faktor pembagi 0..7 yang dapat ditunjukkan seperti pada tabel 2.3.

Tabel 2.3 Konfigurasi *clock* ADC.

ADPS[2..0]	Besar *clock* ADC
000-001	f_{osc} / 2
010	f_{osc} / 4
011	f_{osc} / 8
100	f_{osc} / 16
101	f_{osc} / 32
110	f_{osc} / 64
111	f_{osc} / 128

B. ADC *Muliplexer* (ADMUX)

Bit-bit konfigurasi pada *register* ADMUX dapat ditunjukkan seperti pada tabel 2.4.

Tabel 2.4 *Register* ADMUX.

Bit	7	6	5	4	3	2	1	0
Name	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Keterangan:

- REFS 0,1 : Pemilihan tegangan referensi ADC

  00 : VRef = AREF.

  01 : VRef = AVCC dengan kapasitor eksternal pada AREF.

  10 : VRef = *reserved*.

  11 : VRef = internal 2,56 *volt* dengan kapasitor pada AREF.

- ADLAR : untuk *setting* format data hasil konversi ADC, *default* = 0.

  ADLAR = 0, hasil konversi bit ke-0 hingga 7 berada pada *register* ADCL dan bit ke-8 hingga 9 berada pada *register* ADCH.

  ADLAR = 1, hasil konversi bit ke-0 hingga 1 berada pada *register* ADCL dan bit ke-2 hingga 9 berada pada *register* ADCH.

- MUX[0..4] : pemilihan *channel* ADC yang digunakan, 0 = *channel1*, 1 = *channel2*, dan seterusnya.

C. *Special Function IO Register* (SFIOR)

SFIOR merupakan *register* 8 bit pengatur sumber picu konversi ADC, susunannya dapat ditunjukkan seperti pada tabel 2.5.

Tabel 2.5 *Register* SFIOR.

Bit	7	6	5	4	3	2	1	0
Name	ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Keterangan:

- ADTS[0..2] : Pemilihan *trigger* (pengatur picu) untuk konversi ADC, bit-bit ini akan berfungsi jika bit ADATE pada *register* ADCSRA bernilai 1. Konfigurasi bit ADTS[0..2] dapat ditunjukkan seperti pada tabel
- ADHSM : 1, ADC *high speed mode enabled*.

Untuk operasi ADC, bit ACME, PUD, PSR2, dan PSR10 tidak diaktifkan.

(Andrianto, 2008).

2.4.2 Komunikasi *Universal Syncronous Asyncronous Receiver / Transmitter* (USART)

Register yang perlu ditentukan nilainya yaitu sebagai berikut:

A. USART *Baud Rate Register* (UBRR)

UBBR merupakan *register* 16 bit, yang berfungsi untuk menentukan kecepatan transmisi data yang digunakan. UBBR dibagi menjadi dua yaitu UBRRH dan UBRRL. UBRR[11..0] merupakan bit penyimpanan konstanta kecepatan komunikasi serial. UBRRH menyimpan 4 bit tertinggi, dan UBRRL menyimpan 8 bit sisanya. Data yang dimasukkan ke UBRRH dan UBRRL dihitung menggunakan rumus yang dapat ditunjukkan seperti pada tabel 2.6.

Tabel 2.6 Rumus perhitungan UBRR.

Mode operasi	Rumus nilai UBRR
Mode asinkron kecepatan normal (U2X=0)	$UBRR = \frac{fosc}{16 \times baudrate} - 1$
Mode asinkron kecepatan ganda (U2X=1)	$UBRR = \frac{fosc}{8 \times baudrate} - 1$
Sinkron	$UBRR = \frac{fosc}{2 \times baudrate} - 1$

U2X adalah bit pada *register* UCSRA, yang berfungsi untuk menggandakan *transfer rate* menjadi dua kalinya. Hanya berlaku untuk mode asinkron, untuk mode sinkron bit ini di-*set* 0.

B. USART *Control and Status Register* A (UCSRA)

Susunan bit-bit pada *register* UCSRA dapat ditunjukkan seperti pada tabel 2.7.

Tabel 2.7 *Register* UCSRA.

Bit	7	6	5	4	3	2	1	0
Name	ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

C. USART *Control and Status Register* B (UCSRB)

UCSRB merupakan *register* 8 bit pengatur aktivasi penerima pengirim USART, komposisinya dapat ditunjukkan seperti pada tabel 2.8.

Tabel 2.8 *Register* UCSRB.

Bit	7	6	5	4	3	2	1	0
Name	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Initial Value	0	0	0	0	0	0	0	0

Keterangan:

- RXCIE : bit pengatur aktivasi interupsi penerimaan data serial.
- TXCIE : bit pengatur aktivasi interupsi pengiriman data serial.
- UDRIE : bit pengatur aktivasi interupsi, yang berhubungan dengan kondisi bit UDRE pada UCSRA.
- RXEN : bit pengatur aktivasi penerimaan serial.
- TXEN : bit pengatur aktivasi pengiriman serial.
- UCSZ2 bersama-sama dengan bit UCSZ1 dan UCSZ0 di *register* UCSZ0 menentukan ukuran karakter serial yang dikirimkan yang dapat ditunjukkan seperti pada tabel 2.9.

Tabel 2.9 Penentuan ukuran karakter komunikasi USART.

UCSZ[2..0]	Ukuran karakter dalam bit
000	5
001	6
010	7
011	8
100-110	Tidak dipergunakan
111	9

D. USART *Control and Status Register* C (UCSRC)

UCSRC merupakan *register* 8 bit yang digunakan untuk mengatur mode dan kecepatan komunikasi serial yang dilakukan, komposisinya dapat ditunjukkan seperti pada tabel 2.10.

Tabel 2.10 *Register* UCSRC.

Bit	7	6	5	4	3	2	1	0
Name	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Initial Value	0	0	0	0	0	0	0	0

Keterangan:

- URSEL : merupakan bit pemilih akses antara UCSRC dan UBRR.
- UMSEL : merupakan bit pemilih mode komunikasi serial antara sinkron dan asinkron.
- UPM[1..0] : merupakan bit pengatur paritas.
- USBS : merupakan bit pemilih ukuran bit *stop*.
- UCSZ1 dan UCSZ0 : merupakan bit pengatur jumlah karakter serial.
- UCPOL : merupakan bit pengatur hubungan antara perubahan data keluaran dan data masukan serial dengan *clock* sinkronisasi. Hanya berlaku untuk mode sinkron, untuk mode asinkron bit ini di-*set* 0.

(Andrianto, 2008).

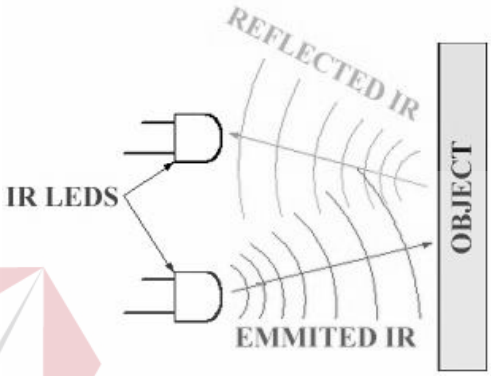
2.7 Sensor Reflektansi *Infra Red* (IR)

Sensor reflektansi IR merupakan sebuah perangkat yang didalamnya berisi sebuah fototransistor (sensitif terhadap cahaya IR) dan emitor IR. Jumlah cahaya yang dipantulkan dari emitor IR ke *phototransistor* menghasilkan pengukuran reflektansi permukaan, misalnya untuk menentukan apakah permukaan berwarna hitam atau putih. Fototransistor memiliki sensitifitas puncak pada panjang gelombang IR, juga sensitif terhadap cahaya tampak. Untuk alasan tersebut, perangkat harus dilindungi dari pencahayaan sekitarnya sebanyak mungkin untuk memperoleh hasil yang dapat diandalkan.

Reflektansi sensor juga dapat digunakan untuk mengukur jarak, dengan syarat reflektansi permukaan tetap konstan. Sebuah sensor reflektansi juga dapat

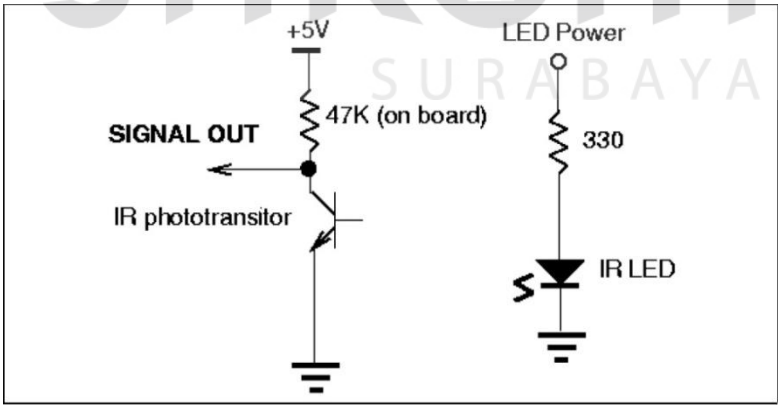
mendeteksi pada pola tergambar di permukaan atau segmen pada roda yang digunakan untuk mengkondekan rotasi pada poros.

Prinsip dasar dari sensor reflektansi IR yaitu mengirim cahaya IR melalui IR-LED, yang kemudian tercermin oleh setiap objek di depan sensor dan kemudian mengambil cahaya tercermin yang dapat ditunjukkan seperti pada gambar 2.7 (Ikalogic, 2011).



Gambar 2.7 Prinsip dasar sensor reflektansi IR.

Rangkaian dasar sensor reflektansi IR dapat ditunjukkan seperti pada gambar 2.8.



Gambar 2.8 Rangkaian dasar sensor reflektansi IR.

Intensitas cahaya yang jatuh pada fototransistor menentukan pengendalian basis dan dengan demikian pada konduktifitas transistor. Jumlah intensitas cahaya yang lebih besar menyebabkan arus besar mengalir melalui

kolektor – emiter. Karena transistor merupakan elemen aktif yang memiliki penguatan arus, fototransistor lebih sensitif daripada fotoresistor sederhana. Elemen yang memancarkan cahaya yaitu IR-*Light Emitting Diode* (LED) menggunakan sebuah resistor untuk membatasi arus yang mengalir melalui perangkat dan biasanya IR – LED selalu dalam keadaan *on*.

(ELEC 201, 1998).