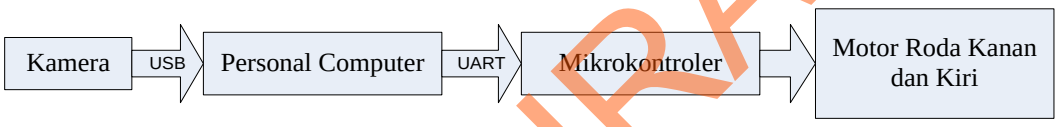
BAB III

METODE PENELITIAN

3.1 Model Penelitian

Pengerjaan Tugas Akhir ini dapat terlihat jelas dari blok diagram yang tampak pada gambar 3.1. Blok diagram tersebut menggambarkan proses dari *capture* gambar hingga perintah ke motor. Terdapat beberapa komponen penting pada blok diagram tersebut antara lain adalah webcam, PC, *microcontroller* dan motor DC.

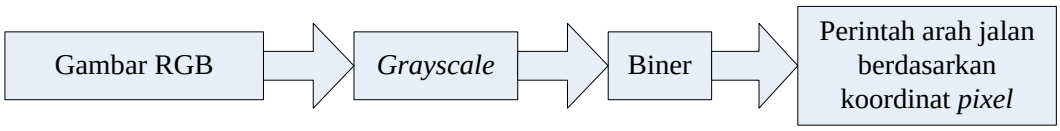


Gambar 3.1 Blok Diagram

Gambar 3.1 merupakan proses mengalirnya data input hingga menjadi output. Data yang diperoleh dari kamera berupa data lintasan yang tampak oleh kamera robot pada saat di lintasan lalu data tersebut dikirim ke *personal computer* melalui USB. Pada *personal computer* hasil kamera tersebut diolah pada *software Microsoft Visual C++ 2008* menggunakan library OpenCV dan hasilnya dikirm melalui serial menuju *microcontroller*. *Microcontroller* mengendalikan motor sesuai dengan perintah dari PC, sehingga motor DC dapat berjalan sesuai dengan jalan yang ditempuh *mobile robot.*

3.2 Proses Pendeteksian Jalan

Proses Pendeteksian Jalan yang dilakukan oleh *processor* (*notebook*) tampak pada blok diagram gambar 3.2.



Gambar 3.2 Blok Diagram Pengolahan gambar

Pada gambar 3.2 gambar RGB diperoleh dari hasil *capture* kamera yang lalu dirubah ke gambar *grayscale* kemudian biner. Pada gambar biner, gambar jalan terlihat kontras antara jalan dan bahu jalan. Sehingga diambil lah koordinat pada bahu jalan untuk menjadi acuan *mobile robot* bergerak ke kiri, ke kanan atau lurus.

3.2.1 Proses Perubahan Warna dari RGB ke *Grayscale*

Proses perubahan warna dari RGB ke *Grayscale* bertujuan untuk mempermudah proses selanjutnya yaitu proses merubah *grayscale* menjadi biner. Sehingga gambar yang diterima oleh *processor* (*notebook*) dirubah langsung ke *grayscale*. Perubahan gambar RGB ke *Grayscale* menggunakan library openCV pada visual C++ menggunakan perintah sebagai berikut.

```
cvCvtColor( src, image2Gray, CV_BGR2GRAY);
```

Pada perintah tersebut sudah terdapat dua *frame*, yang satu berisi gambar asli dari kamera (`src`) sedangkan yang lainnya adalah *frame* yang disediakan untuk hasil perubahan ke *grayscale* (`image2Gray`). Sehingga maksud dari potongan perintah tersebut adalah mengubah gambar `src` ke *grayscale* (`CV_BGR2GRAY`) lalu disimpan pada *frame* bernama `image2Gray`.

3.2.2 Proses Perubahan Gambar *Grayscale* ke Biner

Proses perubahan gambar *grayscale* ke biner bertujuan untuk membedakan warna secara kontras antara bahu jalan dan badan jalan. Sehingga perintah untuk arah kanan dan kiri maupun lurus dapat semakin jelas. Berikut adalah perintah

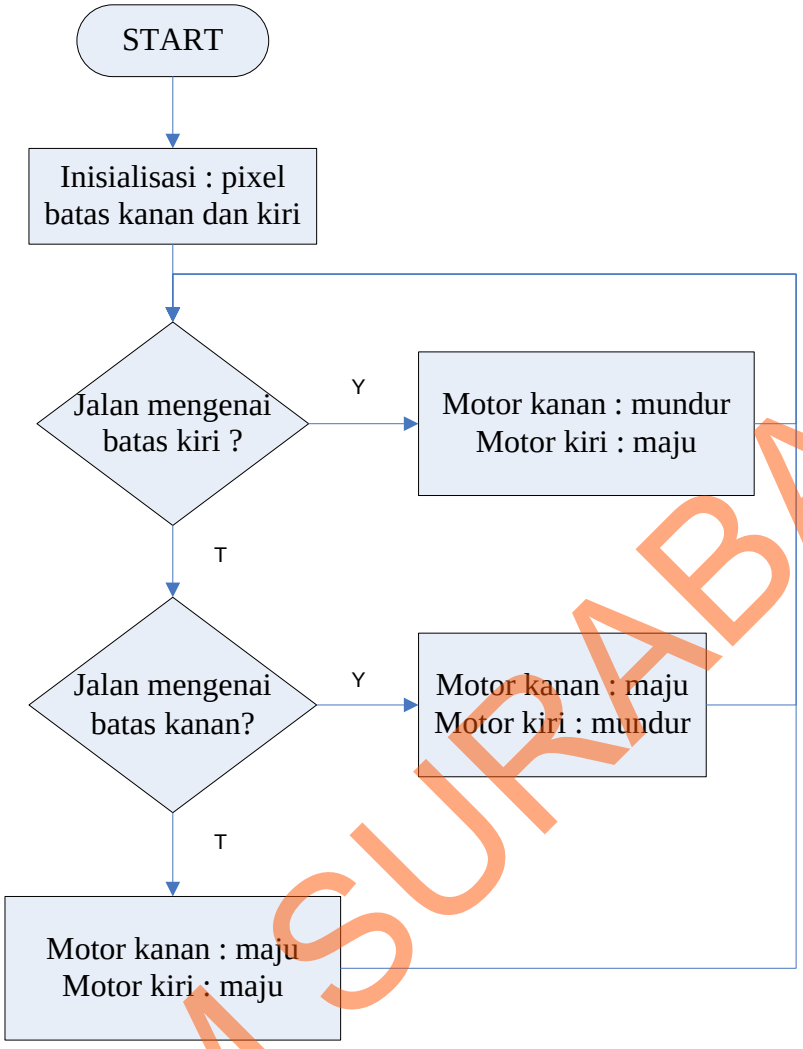
yang memanfaatkan *library* OpenCV untuk mengubah gambar *grayscale* menjadi biner.

```
cvThreshold(image2Gray,image2Gray,50,255,CV_THRESH_BINARY);
```

Pada perintah tersebut gambar *grayscale* dari *frame* `image2Gray` dirubah menjadi biner (`CV_THRESH_BINARY`) dengan *threshold* 50 dan disimpan pada *frame* yang sama yaitu `image2Gray`. *Threshold* bertujuan mengubah *pixel* diatas *threshold* untuk menjadi *pixel* bernilai 255 sedangkan dibawah *threshold* dirubah menjadi 0, dengan demikian didapatkanlah gambar biner.

3.2.3 Proses Pengambilan Koordinat sebagai Acuan

Pada proses ini penulis menetapkan koordinat *pixel* yang digunakan sebagai acuan *mobile robot* untuk bergerak ke kiri maupun ke kanan. Koordinat *pixel* yang dipakai adalah koordinat *pixel* bahu jalan (kanan dan kiri robot) pada posisi robot di tengah jalan, sehingga koordinat tersebut dapat digunakan sebagai acuan. Apabila posisi acuan tersebut terdeteksi mengenai jalan maka robot melakukan aksi, yaitu belok kiri atau kanan tergantung pada acuan kiri atau kanan yang terkena badan jalan. Berikut adalah *flowchart* untuk memperjelas dari proses pendeteksian jalan.



Gambar 3.3 Flowchat Pendeteksian Jalan

Pada gambar 3.3 dijelaskan bahwa terdapat *pixel* yang menjadi batas kanan dan kiri. *Pixel* tersebut menjadi acuan motor robot untuk bergerak ke kanan dan ke kiri. Berikut bagian dari kode program dalam menentukan *pixel* hingga keputusan robot bergerak ke kanan atau kiri.

```
CvScalar s;
            for(i=0;i<120;i++)
            {
                  s=cvGet2D(image2Gray,460,i);
                  if (s.val[0]==255)
                        k++;
            }
            for(i=520;i<640;i++)
            {
                  s=cvGet2D(image2Gray,460,i);
                  if (s.val[0]==255)
                        j++;
            }
```

Pixel dapat diambil dengan memanfaatkan library CvScalar, pada potongan perintah tersebut terlihat bahwa *pixel* koordinat Y yang diambil pada *pixel* 460 sedangkan *pixel* koordinat X yang diambil antara *pixel* ke 0 sampai 120 (acuan kiri) dan *pixel* 520 hingga 640 (acuan kanan). Apabila *pixel* acuan tersebut terkena badan jalan yang bernilai 255 maka diaktifkan flag kanan atau kiri yang digunakan untuk melanjutkan perintah output.

Jarak antara kamera dengan jalan yang terproyeksi oleh *pixel* baris 460 adalah 80 cm. Dengan jarak 80 cm, perbandingan jarak yang sebenarnya dengan *pixel* adalah 1 : 13. Dapat diartikan bahwa 1 cm sama dengan 13 *pixel*, dengan ketentuan jarak kamera dengan *pixel* yang dimaksud adalah 80 cm.

3.3 Komunikasi Menggerakan *Mobile Robot*

Untuk dapat menggerakan *mobile robot* maka perintah dari *Processor* harus dikirim ke *microcontroller* melalui UART, lalu data yang diterima *microcontroller* dibaca dan diproses selanjutnya hingga memperoleh output sesuai yang diinginkan.

3.3.1 Pengiriman data dari *Processor* (*Notebook*)

Pengiriman data dari *processor* berupa pengiriman data karakter. Pengiriman dilakukan melalui UART dari *processor* yang sudah dilengkapi

dengan RS-232 menuju mikrontroler dan data yang dikirim adalah “x” (perintah kiri), “y” (perintah kanan) dan “z” (perintah maju). *Processor* mendeteksi perintah kiri maka processor mengaktifkan flag kiri yang lalu menjadi tanda untuk pengiriman data “x” ke *microcontroller*. Berikut adalah perintah untuk mengirim data.

```
if (k<=3)
{
	cout<<(stderr,"kiri\n");
	if (hanny.Open(1,9600))
	{
		char* start= "x";
		hanny.SendData(start, 1);
		cout<<(stderr,"openedX\n");
	}
	k=0;
}
else if (j<=3)
{
	cout<<(stderr,"kanan\n");
	if (hanny.Open(1,9600))
	{
		char* start= "y";
		hanny.SendData(start, 1);
		cout<<(stderr,"openedY\n");
	}
	j=0;
}
else
{
	cout<<(stderr,"lurus\n");
	k=0;
	j=0;
	if (hanny.Open(1,9600))
	{
		char* start= "z";
		hanny.SendData(start, 1);
		cout<<(stderr,"OpenL\n");
	}
}
```

Pada potongan perintah tersebut ditunjukan bahwa COM serial yang digunakan adalah port 1 dan *baudrate* yang digunakan sebesar 9600. Perintah pengiriman variabel `start` dengan dengan perubahan nilai menjadi x ,y atau z

sesuai dengan perintah, dan banyaknya karakter yang dikirim adalah 1 `hanny.SendData(start, 1)`. Sebelum perintah tersebut terdapat *header* dan *function* yang terlampir digunakan untuk melengkapi perintah mengirim data secara serial.

3.3.2 Penerimaan Data *Microcontroller*

Data dari *processor* diolah oleh *microcontroller* agar dapat mengeluarkan output yang nantinya dapat mengendalikan motor DC. Berikut adalah potongan perintah yang digunakan *microcontroller* untuk menerima data serial.

```
angka=getchar();

         if (angka=='y') // kanan
        {   OCR1A=40;
            PORTD.7=1;  //mki
            OCR1B=40;
            PORTD.6=0;
            putchar('a');
        }
        else if (angka=='x') // kiri
        {
            OCR1A=40;
            PORTD.7=0;
            OCR1B=40;
            PORTD.6=1;
            putchar('b');
        }
        else if (angka=='z') // lurus
        {
            OCR1A=40;
            PORTD.7=1;
            OCR1B=40;
            PORTD.6=1;
            putchar('c');
        }
        else if(angka == 'a')
        {
            OCR1A=OCR1B=PORTD.7=PORTD.6=0;
        }
```

Pada potongan perintah diatas ditunjukan bahwa variabel `angka` digunakan untuk penyimpanan karakter sementara yang dikirim lalu dipilah-pilah

sesuai dengan perintah lalu diproses menjadi output yang digunakan untuk mengendalikan motor DC.

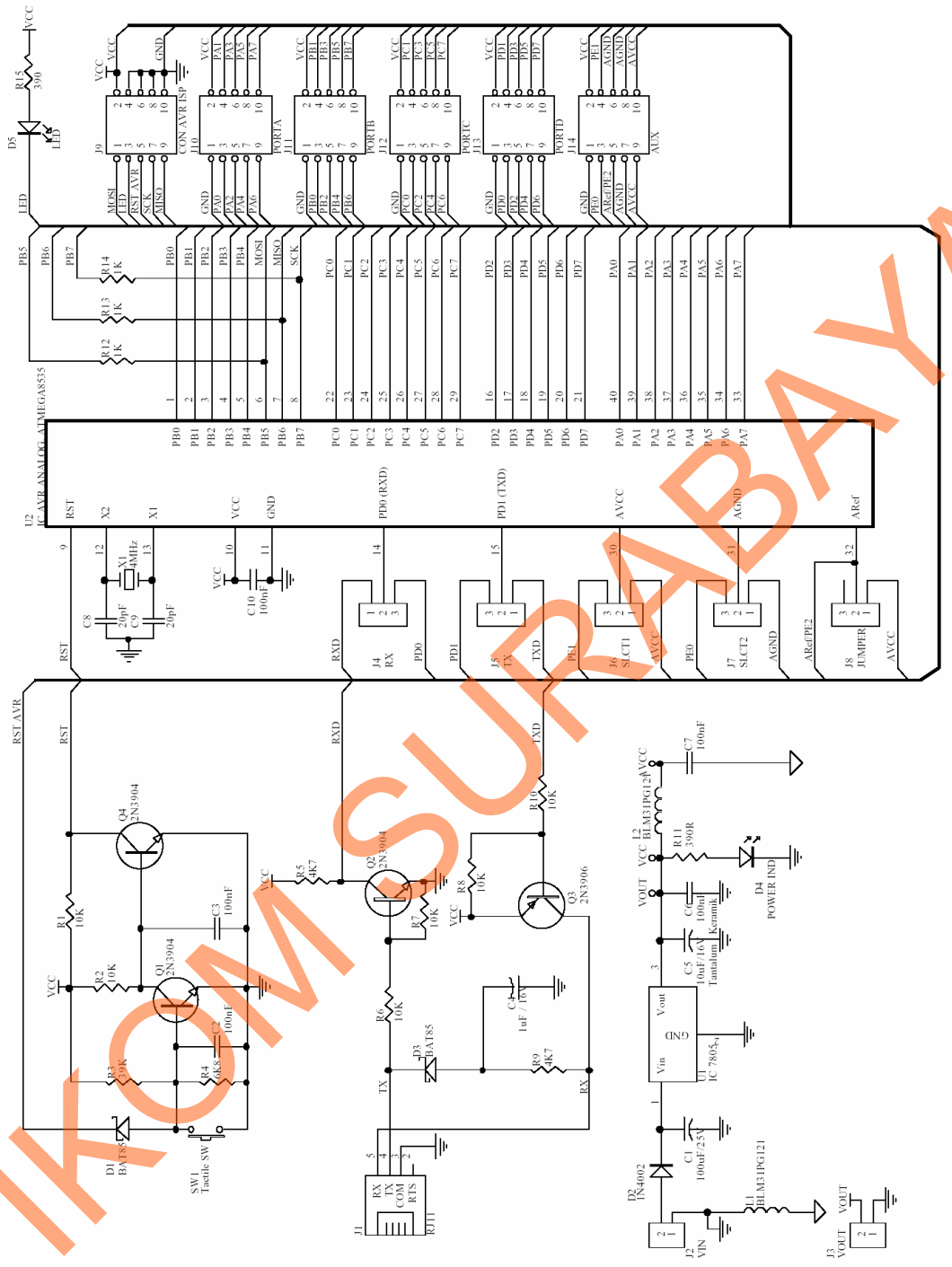
3.4 Perancangan Perangkat Keras

3.4.1 Minimum Sistem ATMEGA8535

Minimum sistem ATMEGA8535 yang digunakan adalah minimum sistem dari innovative electronics. Dengan spesifikasi sebagai berikut

1. *Microcontroller* ATMEGA8535 yang mempunyai 8KB Flash Memory dan 8 channel ADC dengan resolusi 10 bit.
2. Mendukung varian AVR 40 pin, antara lain : ATMega8535, ATMega8515, AT90S8515, AT90S8535, dll. Untuk tipe AVR tanpa internal ADC membutuhkan Conversion Socket.
3. Memiliki jalur I/O hingga 35 pin.
4. Terdapat eksternal Brown Out Detector sebagai rangkaian reset.
5. Konfigurasi *jumper* untuk melakukan pemilihan beberapa model pengambilan tegangan referensi untuk tipe AVR dengan internal ADC.
6. LED Programming indicator.
7. Frekwensi Osilator sebesar 4 MHz.
8. Tersedia jalur komunikasi serial UART RS-232 dengan konektor RJ-11.
9. Tersedia port untuk pemrograman secara ISP.
10. Tegangan input Power Supply 9-12 VDC dan output tegangan 5 VDC.

Konfigurasi minimum sistem digambarkan pada gambar 3.4.

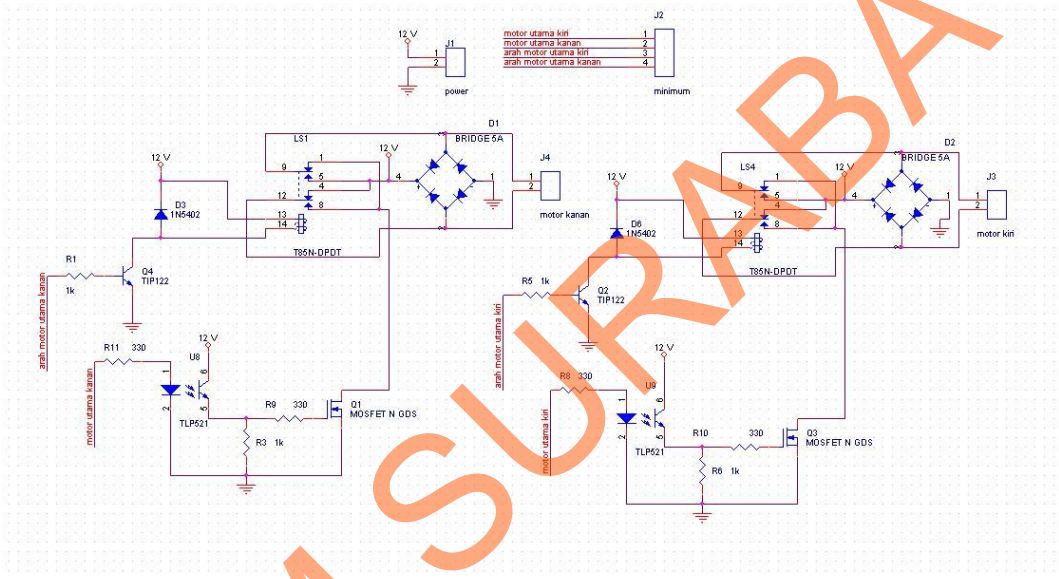


Gambar 3.4 Konfigurasi Minimum Sistem ATMega8535.

(Innovative Electronics, 2004)

3.4.2 Relay Driver

Relay driver digunakan untuk mengendalikan motor DC. Penggunaan relay driver sebagai pengendali motor dikarenakan motor yang dikendalikan adalah motor DC 20 watt dan 12 VDC, sehingga memerlukan *motor driver* yang sesuai dengan spesifikasi motor tersebut. Gambar 3.5 adalah konfigurasi rangkaian *relay driver*.



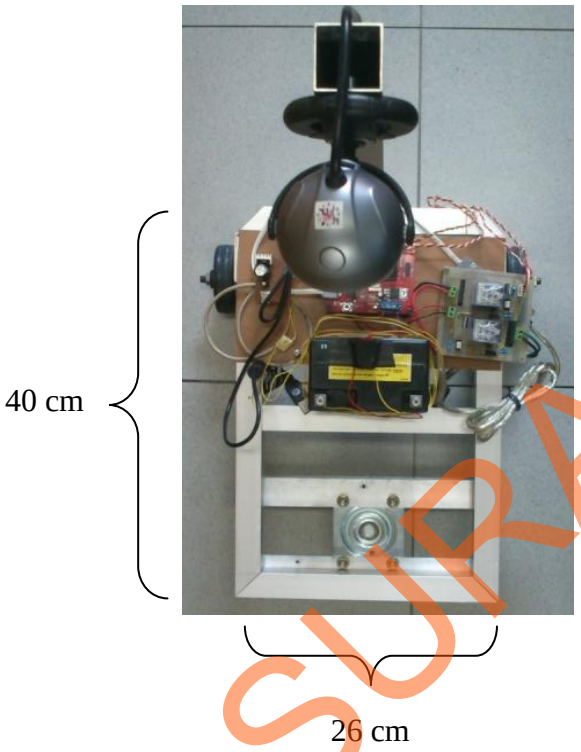
Gambar 3.5 Rangkaian *Relay Driver*

Relay driver tersebut digunakan karena dapat digunakan untuk mengurangi kecepatan motor sesuai dengan nilai PWM yang digunakan *microcontroller* sehingga kecepatan *mobile robot* dapat dikendalikan untuk mengurangi error yang terjadi pada saat *mobile robot* berjalan di lintasannya.

3.4.3 Desain Mekanik Robot

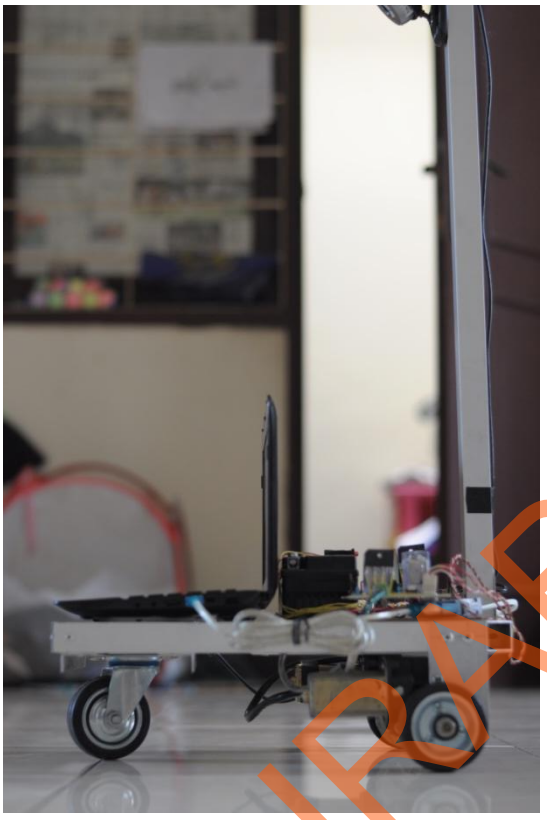
Desain mekanik robot, terdiri dari dua motor, *base* robot dan tiga buah roda. Robot dirancang dengan perangkat elektronik lainnya seperti kamera, *relay driver* dan minimum sistem, sehingga dapat menjadi suatu bentuk *mobile robot*. *Mobile robot* ini dirancang seperti robot *line follower*, namun karena metode dan

perangkat keras yang digunakan berbeda dengan *line follower* maka bentuk disesuaikan dengan kebutuhan. Berikut adalah bentuk keseluruhan dari *mobile robot*.



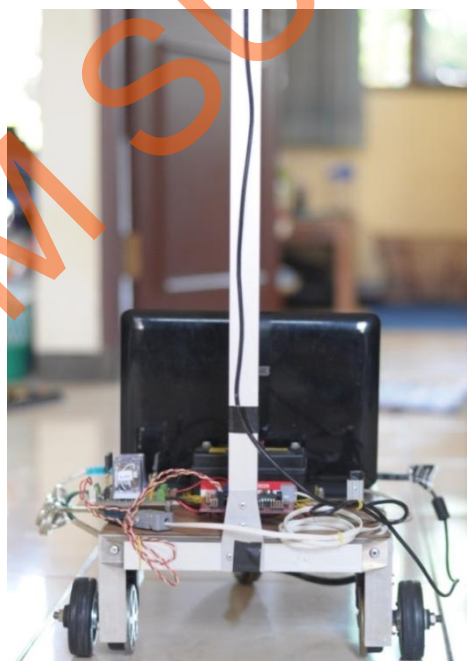
Gambar 3.6 Robot tampak atas

Gambar 3.7 Robot tampak samping kanan



Gambar 3.8 Robot tampak samping kiri

Gambar 3.9 Robot tampak depan



Gambar 3.10 Robot tampak belakang

3.5 Pulse Width Modulation (PWM)

Pengendalian motor DC menggunakan PWM. PWM digunakan untuk menurunkan kecepatan sehingga laju dari *mobile robot* dapat dikendalikan dan error(keluar dari jalur) dapat dikurangi. Laju dari *mobile robot* dapat dikendalikan dengan mengubah-ubah nilai PWM yang dikendalikan melalui *microcontroller*. Sehingga kecepatan dari mobile robot dapat diklasifikasikan menjadi 4 kecepatan, yaitu berhenti, kecepatan pelan, sedang hingga cepat. Pemberian nilai PWM dapat diberikan sebagai berikut :

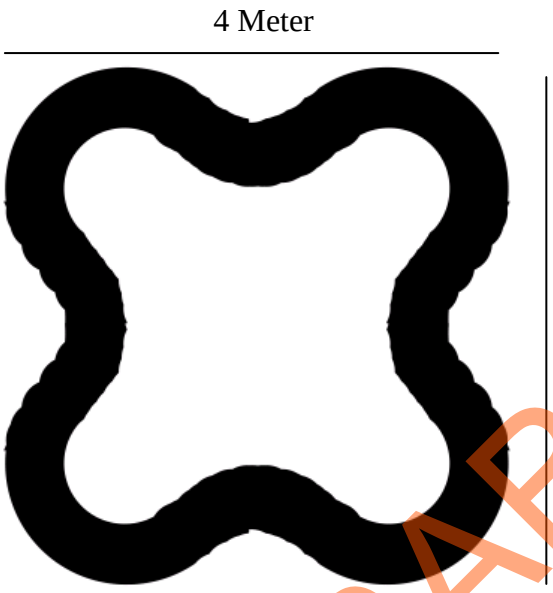
Tabel 3.1 Kecepatan Motor DC

No.	Kecepatan	Nilai PWM
1.	Berhenti	0
2.	Pelan	1-85
3.	Sedang	86-170
4.	Cepat	171-255

Pada penelitian ini, mobile robot menggunakan kecepatan pelan dengan tujuan error yang dihasilkan sangat kecil sehingga mobile robot dapat berjalan dengan baik. Dengan demikian mobile robot pada penelitian ini diberikan nilai PWM antara 1 -85.

3.6 Perancangan Jalan

Jalan yang digunakan untuk menjadi lintasan pada penelitian ini mempunyai karakteristik tikungan ke kiri dan ke kanan sama banyak dan sama besar, sehingga penelitian ini berimbang antara tikungan kanan dan tikungan kiri. Dengan panjang jalan 16 m dan lebar jalan 39 cm yang terbuat dari bahan vinyl yang dicetak dengan *digital printing*. Warna jalan dan bahu jalan hanya terdiri dari warna hitam dan putih. Berikut adalah gambar jalan yang digunakan pada penelitian ini.



Gambar 3.11 Jalan Mobile Robot

*) lebar jalan 39cm.