2. TEORI PENUNJANG

2.1. Bluetooth 1.0

Bluetooth merupakan jenis teknologi *short-range radio frequency* (RF) yang berkerja pada frekuensi 2,4 GHz. *Bluetooth* dikhususkan pada *low-power consumption*, sehingga jangkauannya hanya sebesar 10 m (dengan kecepatan *transfer data* sebesar 1Mbps) meskipun jangkauan *high powered bluetooth motor* dapat mencapai 100m.

Teknologi *bluetooth* memiliki tingkat keamanan yang cukup tinggi terhadap *noise* karena menggunakan *frequency hopping spread spectrum* (FHSS). FHSS merupakan tipe *spread spectrum modulation* yang menggunakan *narrowband carrier* yang frekuensinya selalu berubah dengan pola tertentu (biasa disebut *frequency hopping*), sehingga hanya *transmitter* dan *receiver* yang telah tersinkronisasi saja yang bisa saling berhubungan. Perubahan frekuensi FHSS dapat mencapai 1600 kali tiap detik.

2.2. Modul Embedded Blutooth™ 500

Pada aplikasi ini, Embedded Blutooth™ 500 (EB500) digunakan sebagai perantara komunikasi *bluetooth* antara mikrokontroler dengan komputer (menggunakan *Dongle USB Bluetooth*).

2.2.1. Karakteristik Operasional EB500

Komunikasi *bluetooth* yang digunakan EB500 memiliki karakteristik sebagai berikut:

1. Kekuatan pengiriman sinyal sebesar 4dBm (maksimum).
2. Menggunakan tipe komunikasi serial dengan jangkauan komunikasi pada lapangan terbuka dapat lebih dari 100 meter (328 kaki).
3. EB500 dapat bekerja dengan baik pada temperatur 0^{o} hingga 70^{o}C.
4. *Supply Power* sebesar 5 hingga 12 VDC.

5. Konsumsi arus sebesar 3mA hingga 35mA (bergantung pada kondisi koneksi dan *baudrate).*

2.2.2. Kofigurasi *Pin-Pin* pada EB500

Tabel 2.1. Konfigurasi *Pin*

Pin Name	Pin	Type	Description
VSS	1,2	GND	Ground
RX	3	TTL output	UART data output
TX	4	CMOS/TTL input	UART data input
RX Flow (CTS)	5	CMOS/TTL input, weak pull down	Signaled high to stop module data transmission
TX Flow (CTS)	6	TTL output	Signaled high to stop host data transmission
	7	Reserved	Reserved for future use
Connection Status	8	TTL output	High when there is an active wireless connection
Mode Control	9	CMOS/TTL input, weak pull down	Low for command mode / High for data mode
	10-19	Reserved	Reserved for future use
VIN	20	VCC	Module supply, 5 to 12V DC

Sumber: A7 Engineering. EB500 *User Manual.* Februari 2004. <http://www.parallax.com/dl/docs/prod/comm/eb500.pdf>.

Pin yang dipakai adalah VSS, VIN, RX, TX, dan *Connection Status*. *Pin RX Flow* dan *TX Flow* tidak digunakan karena format *data* serial yang dipakai tidak menggunakan sistem *flow control*. Sedangkan *pin Mode Control* tidak digunakan, karena proses perubahan *mode* pada EB500 dapat dilakukan secara *software*.

2.2.3. Format Pengoperasian EB500

Komunikasi yang digunakan antara EB500 dan mikrokontroler adalah secara serial TTL (Standar Pabrik = 9600 *Baud*, 8 *Data Bits*, 1 *Stop Bits*, *No*

Parity, dan *No Flow Control*) dimana *baudrate* dan konfigurasi *flow control*–nya dapat dimodifikasi sesuai keinginan pemrogram.

EB500 memiliki 2 mode operasi utama yaitu *command mode* dan *data mode*. Setiap kali dilakukan *power up*, EB500 akan selalu masuk dalam *command mode* dan siap untuk menerima *serial command.* Pada *command mode* ini terdapat beberapa perintah yang dapat dikirim untuk menggunakan berbagai macam fitur yang dimiliki oleh EB500. Perintah-perintah itu antara lain sebagai berikut:

1. `get con <CR>` (notasi **<CR>** merupakan $0A_H$ dan $0D_H$): Perintah untuk melihat kondisi *connectable* dari EB500 yang digunakan. Setelah EB500 menerima perintah ini, EB500 akan terlebih dahulu mengirimkan "`ACK`" (perintah benar) atau "`NAK`" (perintah salah) kemudian kondisi dari koneksi (apabila "*on*", berarti EB500 akan menerima semua koneksi yang masuk. Namun apabila "*off*", maka EB500 akan menolak semua koneksi yang masuk).
2. `get dis <CR>`: Perintah untuk melihat kondisi *discoverable* dari EB500 yang digunakan. Setelah EB500 menerima perintah ini, EB500 akan terlebih dahulu mengirimkan "`ACK`" (perintah benar) atau "`NAK`" (perintah salah) kemudian kondisi dari koneksi (apabila "*on*", berarti EB500 dapat ditemukan oleh *bluetooth motor* yang lain. Namun apabila "*off*", berarti EB500 dalam keadaan tidak tampak sehingga tidak dapat ditemukan oleh *bluetooth motor* yang lain).
3. `lst [timeout] <CR>`: Perintah untuk mencari *bluetooth motor* lain yang masuk dalam jangkauan EB500 dan dalam keadaan *discoverable on*. *Timeout* disini hanya bersifat optional, variabel *timeout* dapat diisi lama waktu (detik) yang digunakan dalam melakukan pencarian *bluetooth motor* lain. Setelah EB500 menerima perintah ini, EB500 akan terlebih dahulu mengirimkan "`ACK`" (perintah benar) atau "`NAK`" (perintah salah) kemudian akan memulai pencarian. Apabila dalam pencarian ditemukan adanya *bluetooth device* yang lain, maka EB500 akan mengirimkan *hardware address* dari *bluetooth device* tersebut. Apabila menerima perintah ini, EB500 akan selalu melakukan proses pencarian hingga terjadi "*timeout*". Untuk membatalkan proses pencarian EB500, pengguna dapat mengirimkan notasi `<CR>`.

4. `set con status <CR>`: Perintah untuk merubah kondisi *connectable* dari EB500. Variabel *status* diisi *on* (aktif) atau *off* (non aktif). EB500 hanya mengirimkan "`ACK`" atau "`NAK`" setelah menerima perintah ini.
5. `set dis status <CR>`: Perintah untuk merubah kondisi *discoverable* dari EB500. Variabel *status* diisi *on* (aktif) atau *off* (non aktif). EB500 hanya mengirimkan "`ACK`" atau "`NAK`" setelah menerima perintah ini.

Proses komunikasi pada EB500 secara serial dengan mengirimkan karakter-karakter ASCII. Contoh mencari *bluetooth device* aktif di dekat EB500 yaitu:

Perintah yang dikirim ke EB500 adalah:

```
lst <CR>
```

Perintah yang dikirim dalam notasi *hexadecimal* adalah sebagai berikut:

```
6D 74 75 0A 0D
```

Data yang diterima dalam notasi *hexadecimal* adalah sebagai berikut:

```
41 43 4B 0A 0D  <Delay> 30 30 3B 31 30 3B 36
30 3B 41 45   3B 39 33 3B 34 44 0A 0D <Delay>
30 30 3B 30 44 3B 31 38  3B 30 31 3B 30 34
3B 46 46 0A 0D <Delay>
```

Data yang diterima tersebut adalah:

```
ACK <CR> <Delay>
00:10:60:AE:93:4D <CR> <Delay>
00:0D:18:01:04:FF <CR> <Delay>
```

Setelah menerima "`ACK`", EB500 akan segera mengirimkan *hardware address* dari *bluetooth dongle* yang ada di sekitarnya. `00:10:60:AE:93:4D` merupakan contoh nilai *hardware address* dari *bluetooth dongle* yang ditemukan pertama kali, sedangkan `00:0D:18:01:04:FF` merupakan *hardware address* dari *bluetooth dongle* kedua yang ditemukan. EB500 akan selalu melakukan proses pencarian hingga terjadi *timeout* atau menerima notasi `<CR>` dari mikrokontroler AT89S51. *Delay* yang terjadi pada contoh di atas disebabkan karena EB500 sedang melakukan proses pencarian.

Apabila terjadi kesalahan dalam pengolahan *data*, maka EB500 akan mengirimkan kondisi *error*. Di bawah ini adalah tabel dari kode *error* beserta dengan deskripsi dari penyebab munculnya *error* tersebut.

Tabel 2.2. Kode *Error*

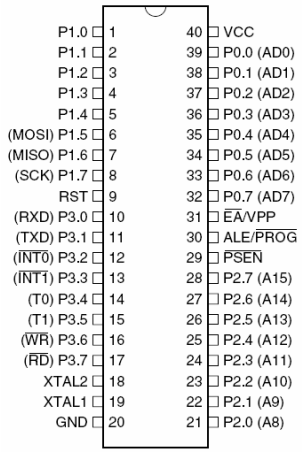
Kode Error	Deskripsi
Err 1	Kode ini muncul apabila *bluetooth motor* yang akan dihubungkan tidak terkonfigurasi dengan baik (misal: *bluetooth motor* yang akan dihubungkan membutuhkan *bluetooth security*)
Err 2	Kode ini muncul apabila EB500 berusaha untuk menghubungkan diri dengan alamat *bluetooth motor* yang salah atau *bluetooth motor* tidak tersedia
Err 3	Kode ini muncul pada saat adanya koneksi aktif dan perintah yang diberikan tidak berlaku jika EB500 dalam kondisi terhubung
Err 4	Kode ini muncul pada saat tidak adanya koneksi aktif dan perintah yang diberikan hanya dapat dilaksanakan jika EB500 dalam kondisi tidak terhubung

Sumber: A7 Engineering. EB500 *User Manual.* Februari 2004. <http://www.parallax.com/dl/docs/prod/comm/eb500.pdf>.

2.3. Mikrokontroler

Mikrokontroler yang dipakai adalah mikrokontroler AT89S51 produksi ATMEL, yang merupakan keluarga dari MCS-51 Intel. Pembahasan mengenai keluarga MCS-51 bersumber pada *Atmel Corporation Microcontroller databook*. Mikrokontroler seri AT89S51 memiliki beberapa karakteristik seperti:

- 4 Kbytes *Flash internet reprogrammable memory*
- 128 x 8-*bit internal data* RAM (*Random Access Memory*)
- Memiliki 2 x 16-*bit timer* / *counter*
- *Programmable serial* UART yang *full duplex*
- Memiliki 4 x 8-*bit I*/*O port*, jadi 32 jalur *I*/*O*
- Memiliki *memory addressing* 64K ROM dan 64K RAM
- 3 *level* pengunci program memori
- Memiliki enam *interrupt* yang dapat diatur penggunaan, dan prioritasnya
- Kecepatan maksimum 40MHz pada Vcc=5 Volt
- Diprogram dengan menggunakan kabel ISP *programming*



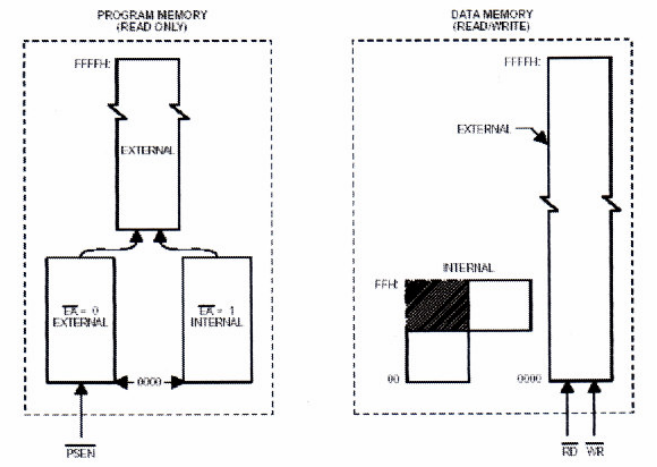
Gambar 2.1. AT89S51

Sumber: Atmel Corporation. *AT89S51 8-Bit Microcontroller With 4K Bytes In System Programmable Flash*.2004.<http://www.atmel.com>

2.3.1. Memory

Pada mikrokontroler ini terdapat 2 macam *memory* yaitu *program memory* dan *data memory*. *Program memory* hanya dapat di-*read*, tetapi tidak dapat di-*write*. Mikrokontroler dapat mengakses *program memory* dan *data memory* sampai 64 kbyte. Pada *internal* mikrokontroler AT89S51 hanya terdapat 4 *kbyte program memory* dan 128 *byte data memory*.

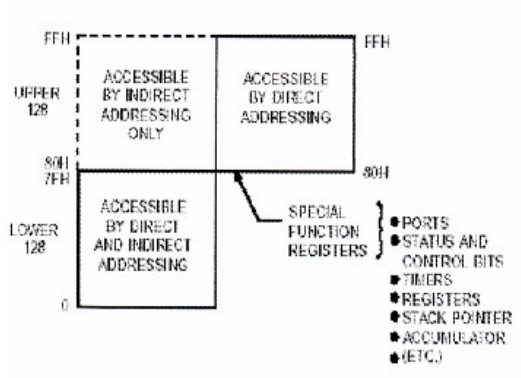
Internal data memory pada mikrokontroler AT89S51 ada 128 *byte*, *data memory* tersebut dibagi menjadi beberapa bagian:



Gambar 2.2. *Memory* Pada Mikrokontroler

Sumber: Atmel Corporation. *AT89S51 8-Bit Microcontroller With 4K Bytes In System Programmable Flash*.2004.<http://www.atmel.com>

Bagian pertama pada *data memory* ada pada alamat 0_H sampai $7F_H$. Pada alamat ini *data memory* dapat diakses baik secara *direct addressing* maupun *indirect addressing*. Pada *address* 80_H sampai FF_H *data memory* terbagi menjadi dua bagian. *Data memory* yang hanya bisa diakses dengan *indirect addressing* dan yang hanya bisa diakses lewat *direct addressing*. *Data memory* yang hanya bisa diakses lewat *direct addressing* merupakan *special function register*. *Data memory* ini menyimpan *register-register* yang diperlukan oleh mikrokontroler, misalnya *port-port*, *status and control bit*, *timer*, *stack pointer*, *accumulator*, dan lain-lain.

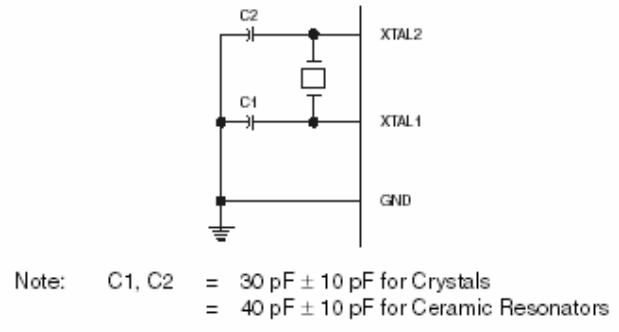


Gambar 2.3. *Internal Data Memory* Pada Mikrokontroler

Sumber: Atmel Corporation. *AT89S51 8-Bit Microcontroller With 4K Bytes In System Programmable Flash*.2004.<http://www.atmel.com>

2.3.2. CPU Timing

Pada mikrokontroler ini terdapat *on-chip oscillator* yang digunakan sebagai sumber *clock* untuk CPU. Untuk menggunakan *oscillator* ini *crystal* dihubungkan antara *pin* XTAL1 dan *pin* XTAL2, selain itu kedua *pin* ini juga dihubungkan dengan kapasitor ke *ground*. CPU *timing* ini ditentukan dari *crystal* yang dipakai.

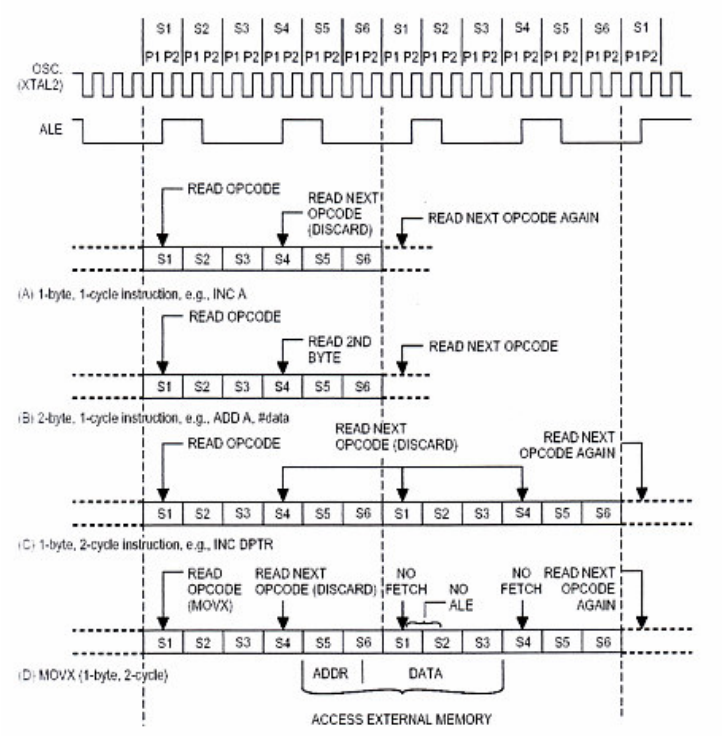


Gambar 2.4. Rangkaian *Crystal* Pada Mikrokontroler

Sumber: Atmel Corporation. *AT89S51 8-Bit Microcontroller With 4K Bytes In System Programmable Flash*.2004.<http://www.atmel.com>

2.3.3. Machine Cycles

Satu *machine cycles* terdiri dari 6 *states* (S1..S6). Satu *states* membutuhkan waktu 2 periode *oscillator*, jadi satu *machine cycles* membutuhkan 12 periode *oscillator*. Pada gambar 2.5. dapat dilihat selang waktu yang dibutuhkan untuk beberapa macam perintah.



Gambar 2.5. *Timing Machine Cycle*

Sumber: Atmel Corporation. *AT89S51 8-Bit Microcontroller With 4K Bytes In System Programmable Flash*.2004.<http://www.atmel.com>

2.3.4. Special Function Register

Pada alamat 80_H sampai FF_H pada *data memory* terdapat *special function register* (SFR). *Register-register* SFR ini mempunyai fungsi khusus dalam mikrokontroler. Berikut ini adalah tabel SFR dan penjelasannya:

PSW: Progrm Status Word (*Bit* Addressable)

CY	AC	F0	RS1	RS0	OV	—	P

CY	PSW.7	*Carry flag*.
AC	PSW.6	*Auxilary carry flag*.
F0	PSW.5	*Flag* 0 digunakan untuk user.
RS1	PSW.4	*Register Bank selector bit 1*.
RS0	PSW.3	*Register Bank selector bit 0.*
OV	PSW.2	*Overflow flag*.
–	PSW.1	
P	PSW.0	*Parity flag*.

IE: *Interrupt Enable* Register (*Bit* Addressable)

EA	—	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	*Enable* semua *interrupt*. Jika EA='0', tidak ada *interrupt* yang jalan. Jika EA='1', jalan tidaknya *interrupt* tergantung pada *enable* tiap *interrupt*.
–	IE.6	
ET2	IE.5	*Enable Timer 2 overflow* atau capture *interrupt*.
ES	IE.4	*Enable serial port interrupt*.
ET1	IE.3	*Enable Timer 1 overflow interrupt*.
EX1	IE.2	*Enable Timer 0 overflow interrupt*.
EX0	IE.0	*Enable External Interrupt 0*.

TCON: *Timer/Counter Control* Register (*Bit* Addressable)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	*Timer 1 overflow flag.*
TR1	TCON.6	*Timer 1 run control bit*.
TF0	TCON.5	*Timer 0 overflow flag.*
TR0	TCON.4	*Timer 0 run control bit.*
IE1	TCON.3	*External Interrupt 1 edge flag.*
IT1	TCON.2	*Interrupt 1 type control bit*
IE0	TCON.1	*External Interrupt 0 edge flag.*
IT0	TCON.0	*Interrupt 0 type control bit.*

Sumber: Atmel Corporation. *AT89S51 8-Bit Microcontroller With 4K Bytes In System Programmable Flash*.2004.<http://www.atmel.com>

TMOD: *Timer/Counter* Mode *Control* Register (Not *Bit* Addressable)

Timer 1				Timer 0			
GATE	C/T	M1	M0	GATE	C/T	M1	M0

GATE	Jika GATE='1' *TIMER/COUNTER*x jalan jika pin INTx berlogika *high* (*hardwarecontrol*). Jika GATE='0', *TIMER/COUNTER*x jalan jika TRx='1'(*software control*).
C/T	selector Timer atau Counter.
M1	Mode selector bit.
M0	Mode selector bit.

M1	M0	Operating Mode	
0	0	0	13-bit Timer
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	Split Timer Mode: (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

SCON: *Serial Port Control* Register (*Bit* Addressable)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	*Serial Port mode specifier*.
SM1	SCON.6	*Serial Port mode specifier*.
SM2	SCON.5	*Enable* komunikasi *multiprocessor*
REN	SCON.4	*Enable/Disable reception*.
TB8	SCON.3	Bit ke 9 yang akan di-*transmit*
RB8	SCON.2	Bit ke 9 yang di-*receive*
TI	SCON.1	*Transmit interrupt flag*.
RI	SCON.0	*Receive interrupt flag*.

SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 OR Fosc./32
1	1	3	9-Bit UART	Variable

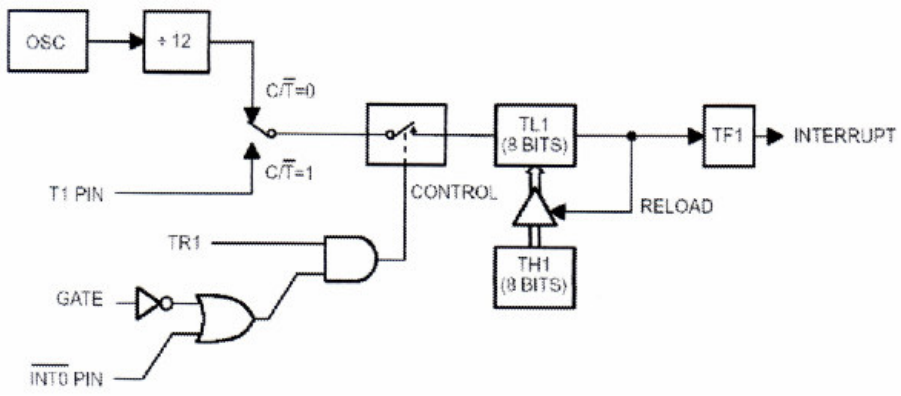
Sumber: Atmel Corporation. *AT89S51 8-Bit Microcontroller With 4K Bytes In System Programmable Flash*.2004.<http://www.atmel.com>

2.3.5. 8-bit Auto-Reload Timer

Timer 0 dan *Timer* 1 pada mikrokontroler dapat dibuat pada *mode auto-reload*. Untuk mengatur *mode* ini, maka pada *register* TMOD harus diisi pada *mode* 2. Pada *mode* 2 ini, *Timer register* TL_x digunakan sebagai *counter* 8 *bit*

counter dengan *automatic reload*. Pada waktu *Timer*$_x$ *overflow*, isi dari TH_x akan di-*load* ke TL_x.

Waktu yang diperlukan untuk naiknya nilai pada *Timer* sesuai dengan 1 *machine cycle* atau seper duabelas kali frekuensi *oscillator*, sedangkan pada *counter* tergantung nilai T_x (*pin Timer*). Gambar 2.6. merupakan gambaran dari rangkaian *Timer* 1 mikrokontroler.



Gambar 2.6. Rangkaian *Timer* 1 Pada Mode *Auto-Reload*

Sumber: Atmel Corporation. *AT89S51 8-Bit Microcontroller With 4K Bytes In System Programmable Flash*.2004.<http://www.atmel.com>

2.3.6. Serial Mode 1

Pada *mode* 1 pengiriman *data serial* terdiri dari 10 *bit*, *start bit* (‘0’), 8 *bit data* dan *stop bit* (‘1’). Waktu *receive*, *stop bit* akan masuk ke RB8 pada SCON. *Baudrate* dapat ditentukan oleh *Timer* 1 *overflow rate* maupun *Timer* 2 *overflow rate*. Transmisi diinisialisasi pada saat SBUF diisi, tetapi transmisi sesungguhnya terjadi pada saat S1P1 *machine cycle* setelah *overflow Timer* 2.

Transmisi mulai pada saat ‘SEND’ aktif, kemudian memberikan *start bit* pada TXD. Satu *bit timer* berikutnya ‘DATA’ diaktifkan, sehingga meng-*enable output bit* dari *transmit shift register* ke TXD. *Shift pulse* pertama terjadi satu *bit time* setelah itu.

Selagi *data bit* di-*shift* ke kanan, ‘0’ akan di-*clock* dari kiri. Pada saat MSB dari data pada posisi *output* dari *shift register*, ‘1’ yang berada pada posisi ke sembilan akan berada di sebelah kiri *bit* MSB, sedangkan sisanya berisi 0. Kondisi ini akan mengaktifkan *flag* dari TX *control unit* untuk melakukan *shift* terakhir, lalu ‘SEND’ akan di-*deactive*-kan dan TI diaktifkan.

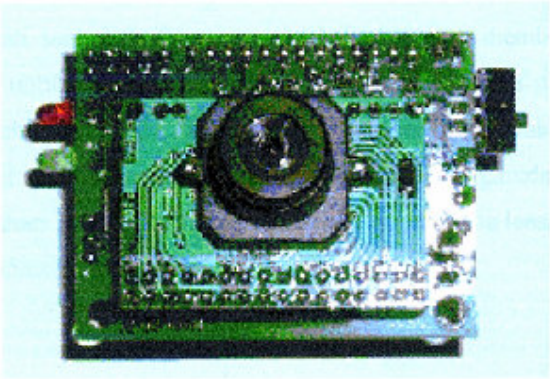
Penerimaan dimulai pada saat terjadi transisi ‘1’ ke ‘0’ terdeteksi pada *pin* RXD. RXD disampel pada 16 kali *baudrate* yang dipakai. Pada waktu transisi terdeteksi, *divide by-16 counter* akan di-*reset*, dan $1FF_H$ akan di-*input*-kan ke *shift register*. Pe-*reset*-an *divide by-16 counter* digunakan untuk penyesuaian dengan *bit times* berikutnya. 16 *state* pada *counter* membagi setiap *bit time* menjadi 16. Pada *bit* ke 7, 8 dan 9 pada *bit time*, *bit detector* akan mengambil sampel dari RXD. Sampel yang mempunyai nilai yang sama 2 dan 3 kali pada *(bit* 7, 8 dan 9) digunakan sebagai nilai data. Pengambilan sampel ini digunakan untuk menghindari terjadinya *noise*. Untuk menghindari *bit* pertama yang salah (jika nilai dari *first bit time* tidak 0), rangkaian *receive* akan melakukan *reset* dan unit akan mencari transisi ‘1’ ke ‘0’ bit yang lain. Jika *start bit valid*, *bit* akan di-*shift* pada *input shift register* dan sisa penerimaan akan dijalankan.

Pada saat *data bit* datang dari kanan, ‘1’ akan di-*shift* ke kiri. Saat *start bit* datang pada posisi paling kiri dari *shift register*, maka *flag* RX *control* blok akan menyala untuk melakukan satu *shift* terakhir, me-*load* SBUF dan RB8 serta men-*set* RI. Hal ini dilakukan bila RI = ’0’ dan SM2 = ’0’ atau *stop bit* = ’1’. Jika dua kondisi itu tidak terpenuhi, *frame* yang didapat akan hilang. Jika dua kondisi terpenuhi, *stop bit* akan masuk ke RB8, 8 *bit data* masuk ke SBUF dan RI=’1’. Pada saat ini baik kondisi terpenuhi atau tidak, rangkaian akan tetap mencari transisi ‘1’ ke ‘0’.

2.4. Kamera CMUcam

Kamera CMUcam adalah sebuah *board* mikrokontroler SX28 yang dihubungkan dengan kamera CMOS Omnivision OV6620. Sistem ini berkomunikasi dengan komunikasi serial pada level TTL atau bisa juga pada level RS232. Kamera CMUcam ini mempunyai bermacam-macam fungsi, yaitu:

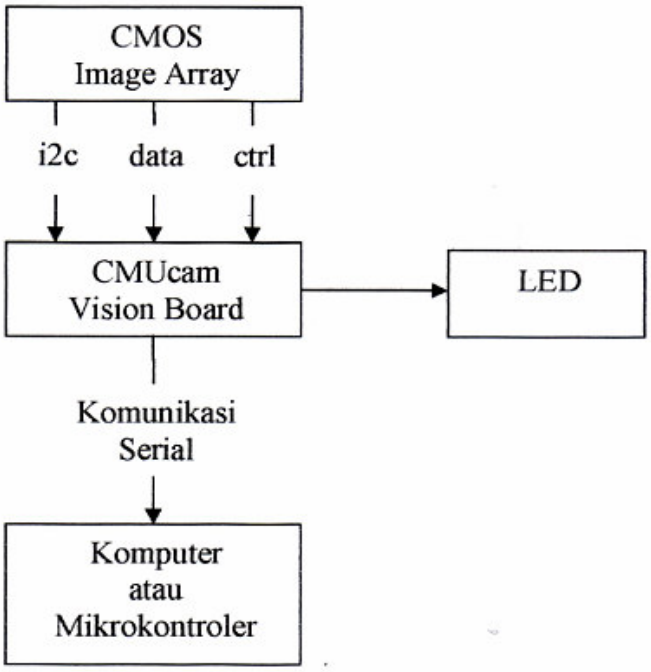
- Melacak bercak warna yang ditentukan pada kecepatan 17 frame per detik.
- Mengatur parameter kamera.
- Menangkap gambar.
- Resolusi 80x143.
- Komunikasi serial dengan *baudrate* 115.200/38.400/19.200/9.600.



Gambar 2.7. Kamera CMUcam

2.4.1. Konfigurasi dan Penggunaan Umum

Konfigurasi yang paling umum dari CMUcam adalah dihubungkan ke prosesor lain melalui komunikasi serial RS232. Prosesor ini bisa berupa komputer atau mikrokontroler. CMUcam menggunakan daya yang kecil sehingga bisa digunakan untuk menambah sistem visual (ke sistem kecil yang tidak mempunyai daya besar). Protokol komunikasinya dirancang mampu untuk mengakomodasi prosesor yang lambat. Bila devais tidak punya komunikasi serial RS232 bisa juga dilakukan komunikasi serial pada level TTL.



Gambar 2.8. Blok Diagram CMUcam

2.4.2. Mengenai Kamera CMOS

Mulai dari saat diaktifkan pertama kali, kamera membutuhkan waktu sekitar 15 detik untuk menyesuaikan cahaya. Perubahan drastis pada lingkungan seperti lampu dihidupkan dan dimatikan bisa menyebabkan waktu penyesuaian yang sama. Saat digunakan di luar ruangan (untuk menghindari radiasi infra merah dari matahari) perlu digunakan filter infra merah pada lensa kamera untuk menghindari *ambient light level*.

Ciri dari kamera yang perlu diperhatikan adalah *array* sensor CMOS mengeluarkan nilai di antara 16 dan 240 setiap pixel. Efek ini akan terlihat saat kamera sedang melacak warna, mencari data *mean color*, atau menangkap gambar.

2.4.3. Komunikasi Serial

Kamera CMUcam bisa berkomunikasi melalui *port* serial dengan level tegangan TTL ataupun RS232 (untuk mengirimkan perintah dan mengambil data dari dan ke CMUcam). Adapun parameter yang dibutuhkan adalah:

- *Baudrate* 115.200 (dapat diubah).
- 8 *bit* data.
- 1 *stop bit*.
- Tanpa *parity*.
- Tanpa *flow control*.

Perintah yang dikirimkan menggunakan karakter ASCII. Apabila berhasil menerima transmisi, maka CMUcam akan mengirimkan *string* "`ACK`". Bila gagal, maka akan mengirimkan *string* "`NCK`". Setelah mengirimkan *string* "`ACK`" atau "`NCK`", maka akan mengirimkan karakter *line feed*. Saat tanda titik dua (":") diterima, berarti CMUcam dalam keadaan kosong dan menunggu perintah. Karakter *white space* mempunyai arti untuk memisahkan argumen. Beberapa perintah yang bisa digunakan antara lain:

- `\r`

  `\r` adalah karakter ASCII dari *enter* yaitu $0D_H$. Perintah ini digunakan untuk men-*set* kamera dalam keadaan kosong. Seperti pada semua perintah (bila berhasil menerima), CMUcam akan mengirimkan *string* "`ACK`" atau

"NCK" (bila gagal menerima). Perintah ini biasa digunakan untuk mengakhiri perintah lain.

- `CR [reg1 value1 [reg2 value2 … reg16 value16] ]\r`

  Perintah ini untuk men-*set register internal* kamera ("*Camera Register*") secara langsung. Semua data yang dikirim melalui perintah ini sebaiknya dalam bentuk karakter desimal (yang dapat dibaca), kecuali kamera telah di-*set* dalam *raw mode*. Perintah ini bisa digunakan untuk mengirim kombinasi 16 nilai *register*. Memanggil perintah ini (tanpa menulis argumen) akan menyebabkan nilai *register* di-*set* ke keadaan awal (*default*).

Tabel 2.3. Register dan Nilai Kamera

Register		Nilai	Efek
5	Contrast	0-255	
6	Brightness	0-255	
18	Color Mode	36	YCrCb Auto White Balance On
		32	YCrCb Auto White Balance Off
		44	RGB Auto White Balance On
		40	RGB Auto White Balance Off (default)
17	Clock Speed	2	17 fps
		3	13 fps
		4	11 fps
		5	9 fps
		6	8 fps
		7	7 fps
		10	5 fps
		12	4 fps
19	Auto Exposure	32	Auto Gain off
		33	Auto Gain on (default)

Sumber: Anthony Rowe. *CMUcam User Manual*. Juli 2004. <http://www-2.cs.cmu.edu/~cmucam/Download/CMUcamManual.pdf>.

- `DF\r`

  Perintah ini untuk mengambil frame ("*Dump Frame*") ke *port* serial. Perintah ini adalah satu-satunya yang secara *default* mengirimkan karakter ASCII yang tidak terbaca. Perintah ini mengirimkan data paket tipe F yang terdiri dari video mentah kolom per kolom dengan *byte* sinkronisasi *frame* dan *byte* sinkronisasi kolom. Karena *data rate* yang dibutuhkan untuk mengirim video mentah jauh melebihi kecepatan maksimal *serial port*, maka hanya satu kolom per *frame* dikirim pada saat itu. Untuk mendapatkan rasio aspek yang

tepat, jumlah kolom tiap *pixel* dikali dua. Kamera mampu mengeluarkan sebuah *frame* dengan kecepatan penuh (17 kolom per detik) pada *baudrate* 115.200. Pada *baudrate* yang lebih rendah (atau pada *baudrate* 115.200 ditambah *delay*), maka *frame rate* harus diturunkan untuk bisa melihat gambar dengan resolusi penuh.

Format data paket tipe F:

1 r g b r g b r g b … 2 r g b r g b … 2 3

1 = *frame* baru

2 = akhir dari kolom

3 = akhir *frame*

Nilai RGB berada antara 16 sampai 240. Nilai ini dalam *format* paket data tipe F akan diubah menjadi bilangan *hexadecimal*. Contoh, jika seluruh daerah yang di-*capture* memiliki nilai R = 30, G = 60, B = 75, maka data yang diterima dari CMUcam (angka dalam bilangan *hexadecimal*) adalah:

01 1E 3C 4B 1E 3C 4B … 02 1E 3C 4B … 02 03

- `GV\r`

  Perintah ini untuk mengetahui versi *firmware* ("*Get Version*") dari kamera. Mengirimkan "`ACK`" diikuti dengan *string* versi dari *firmware*.

- `HM active\r`

  Perintah ini membuat kamera hanya mengambil posisi resolusi setengah horizontal ("*Half-horinzontal Mode*") untuk perintah "`DF`" pada saat mengeluarkan gambar *bitmap*. Nilai "active"=1 berarti mode ini aktif dan setiap kolom ganjil akan diproses. Secara *default* nilai "active"=0 berarti mode ini dimatikan.

- `SW [x y x2 y2]\r`

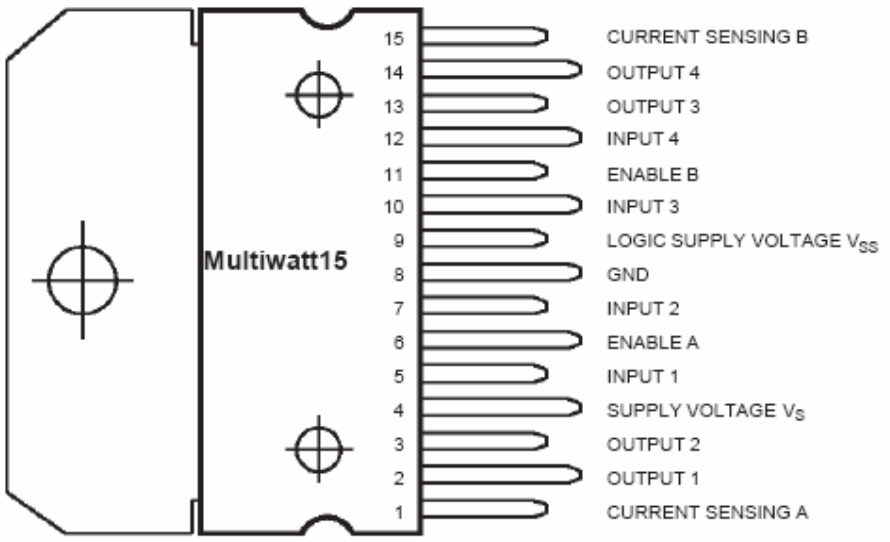
  Perintah ini untuk mengatur ukuran *window* dari kamera ("*Set Window*"). Menggunakan sistem koordinat kartesius pada pojok kiri atas dan pojok kanan bawah pada *window* yang ingin di-*set*. Bila perintah ini dikirim tanpa argumen akan men-*set* ke nilai *default*-nya (`1,1,80,143`).

2.5. Driver Motor

Driver motor yang digunakan adalah IC L298. IC L298 ini berisi dua *full-bridge driver* yang mampu mengendalikan dua buah motor. IC L298 di rancang untuk beban bertegangan tinggi dengan arus besar yang dikendalikan oleh level *logic* TTL yang dapat digunakan sebagai *relay*, *solenoid*, motor DC dan motor *stepper*.

Beberapa karakteristik dari IC L298 adalah:

- Tegangan operasi *supply* sampai dengan 46 volt
- Total arus DC sampai dengan 4 A
- Tegangan saturasi rendah
- Memiliki pengaman temperatur tinggi
- Tegangan *logic* "0" sampai dengan 1,5 volt
- Memiliki 2 *enable input*



Gambar 2.9. Konfigurasi *Pin Driver* Motor

Sumber: SGS-Thomson Microelectronics. *Dual Full-Bridge Driver*. 2004. <http://www.dialelec.com/477.html>

Konfigurasi dari kaki-kaki driver motor L298 diperlihatkan pada gambar 2.10. Fungsi dari tiap-tiap kaki sebagai berikut:

- *Sense* A; *Sense* B (*pin* 1; *pin* 15)

  Pin ini digunakan untuk me-*monitor* arus pada *bridge* A dan *bridge* B.
- *Output* 1; *Output* 2 (*pin* 2; *pin* 3)

Pin ini merupakan *pin output* untuk *bridge* A.

- Vs (*pin* 4)

  Merupakan *pin supply* tegangan untuk *output*.

- *Input* 1; *Input* 2 (*pin* 5; *pin* 7)

  Pin ini digunakan untuk mengontrol *bridge* A.

- *Enable* A; *Enable* B (*pin* 6; *pin* 11)

  Pin ini digunakan untuk mengaktifkan atau menonaktifkan *bridge* A dan *bridge* B.

- GND (pin 8)

  Pin ini merupakan *pin ground* untuk driver.

- Vss (pin 9)

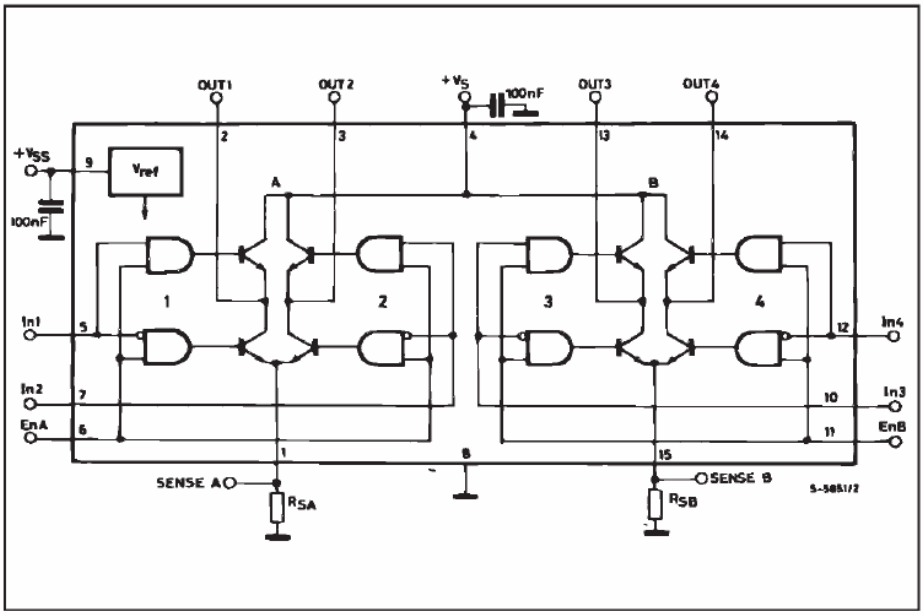
  Pin ini merupakan *pin supply logic* untuk *driver*.

- *Input* 3; *Input* 4 (*pin* 10; *pin* 12)

  Pin ini digunakan untuk mengontrol *bridge* B.

- Output 3; Output 4 (pin 13; pin 14)

  Pin ini merupakan *pin output* untuk *bridge* B.



Gambar 2.10. Blok Diagram IC L298

Sumber: SGS-Thomson Microelectronics. *Dual Full-Bridge Driver*. 2004. <http://www.dialelec.com/477.html>.

2.6. MAX 232

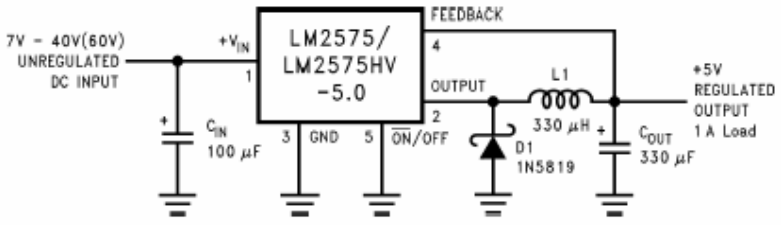
Komunikasi RS232 merupakan komunikasi yang digunakan pada PC (*Personal Computer*) melalui *serial port* dan bersifat *point to point*. Pada *device* umumnya menggunakan level tegangan TTL. Untuk dapat berkomunikasi dengan PC, maka diperlukan konverter TTL – RS232. Secara umum komunikasi ini terdiri dari TXD, RXD, dan *ground*. Komunikasi RS232 bekerja dengan menggunakan *ground* sebagai referensi dari tegangan TXD dan RXD, sehingga untuk komunikasi jarak jauh RS232 tidak dapat digunakan.

Dalam aplikasi ini komunikasi untuk RS232 menggunakan IC MAX 232 yang digunakan untuk menerima keluaran dari mikrokontroler, mengolahnya dan mengirimkan sinyal hasil pengolahan tersebut kepada CMUcam. MAX 232 juga digunakan untuk menerima *data frame* dari CMUcam dan mengirimkannya pada mikrokontroler.

2.7. LM 2575

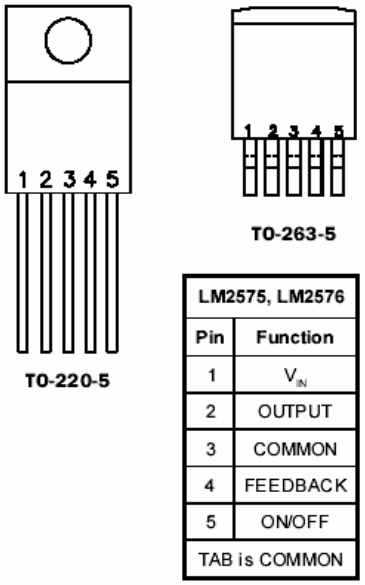
LM2575 adalah sebuah rangkaian *regulator* dengan variasi *output* 3.3V, 5V, 12V, 15V. Pada pengerjaan Tugas Akhir ini menggunakan LM 2575T-5 karena *supply* yang digunakan untuk *regulator* dan menggerakkan *mobile robot* adalah 12V juga diperlukan tegangan *output* 5V untuk *supply* mikrokontroler. Beberapa karakteristik dari IC LM2575T-5 adalah:

- Tegangan operasi *supply* antara 7-40 volt
- Arus *output* maksimum adalah 1 A
- Frekuensi *internal oscillator* 52 KHz



Gambar 2.11. Aplikasi LM 2575T-5

Sumber: National Semiconductor. *LM1575/LM2575/LM2575HV SIMPLE SWITCHER® 1A Step-Down Voltage Regulator*. Agustus 2004. <http://cache.national.com/ds/LM/LM1575.pdf>.



LM2575, LM2576	
Pin	Function
1	V_{IN}
2	OUTPUT
3	COMMON
4	FEEDBACK
5	ON/OFF
TAB is COMMON	

Gambar 2.12. Konfigurasi *Pin* LM 2575T-5

Sumber: Semtech. *LM2575 & LM2576 1A & 3A Miniconverter Switching Regulators*. Januari 2001.
<http://www.datasheetarchive.com/search.php?s=10&q=LM2575>.

2.8. Borland Delphi

Borland Delphi adalah bahasa pemrograman yang bekerja dalam lingkup Microsoft Windows. Bahasa yang digunakan didasarkan dengan bahasa pemrograman Pascal, dan merupakan bahasa pemrograman yang berorientasi obyek atau yang dikenal dengan bahasa *Object Oriented Programing*. Selain berbagai fasilitas pemrograman yang bersifat umum, Borland Delphi juga mendukung pemrograman *database*. Pada sub bab berikut ini akan dibahas beberapa istilah penting dari Borland Delphi yaitu:

- *Project*, merupakan sekumpulan form, unit dan beberapa *file* lainnya yang menjadi program aplikasi. *File* utama *project* disimpan dalam *file* berakhiran ".dpr". Selain itu, Borland Delphi akan membuat sejumlah *file* yang diperlukan. Misalnya *unit file* ".pas", *form file* ".dfm", *Resource file* ".res".
- *Form*, merupakan obyek yang dipakai sebagai tempat bekerja program aplikasi. *Form* berbentuk jendela dan dapat dibayangkan sebagai kertas atau meja kerja yang dapat ditulis atau diletakkan obyek-obyek lain ke dalamnya.

Setiap *form* mengandung *unit* yang akan berisi perintah-perintah yang akan mengendalikan *form*.

- *Unit*, merupakan modul kode program, satu program mungkin mempunyai satu atau lebih *unit*. *Unit* ada dua macam, yang pertama *unit* yang dapat dipisahkan dengan *form*, sebaliknya *unit* yang kedua merupakan *unit* yang tidak dapat dipisahkan oleh *form*. *Unit* yang tidak dapat dipisahkan oleh *form* biasanya merupakan *unit* yang mengatur dan mengendalikan segala sesuatu yang berhubungan dengan *form*.
- *Property*, digunakan untuk menentukan *setting* suatu obyek. Suatu obyek biasanya mempunyai beberapa *property*, yang dapat diatur langsung dari lembar *property* pada *Object Inspector* maupun diatur lewat kode program. *Property setting* akan menentukan cara kerja dari obyek yang bersangkutan saat program aplikasi dijalankan.
- *Event*, merupakan peristiwa atau kejadian yang diterima oleh suatu obyek. *Event* yang diterima obyek akan memicu Delphi untuk menjalankan program yang ada di dalamnya.

2.9. Joystick

Pada komputer terdapat multi I/O card yang mempunyai 15 pin *header* yang ada pada *motherboard* yang dapat terhubung dengan *Joystick*. Card ini terdiri dari *bus interface electronics* dan 4 buah *monostable multivibrator* (semuanya terdapat dalam 558 chip). *Monostable multivibrator* merupakan *timer* yang menghasilkan sinyal pulsa dengan lebar pulsa yang telah ditentukan pada saat mendapat sinyal *trigger*.

Joystick terdiri dari dua buah potensiometer dengan *variable* resistor yang bernilai antara 0 Ω dan 100 KΩ (beberapa *Joystick* bernilai lebih dari 150KΩ). Nilai minimum resistor dari potensiometer terjadi ketika *Joystick* berada pada posisi kiri atas. Komputer membaca dan menulis data *Joystick* (pada umumnya) pada alamat 201_H. *Joystick buttons* menentukan *on/off input*. Jika button ber-*logic* '1', maka *data* dapat terkirim pada komputer. Jika *button* ditekan sehingga ber-*logic* '0', maka tidak ada *data* yang masuk.

Game port 201h byte:

7	6	5	4	3	2	1	0
but4	but3	but2	but1	stk4	stk3	stk2	stk1

7 6 5 4 3 2 1 0

* . . . . Button B2 (pin 14), 0=closed, 1=open (default)
. * . . . . . . Button B1 (pin 10), 0=closed, 1=open (default)
. . * . . . . . Button A2 (pin 7), 0=closed, 1=open (default)
. . . * . . . . Button A1 (pin 2), 0=closed, 1=open (default)
. . . . * . . . Monostable BY (from pin 13), 1=timing, 0=timed-out
. . . . . * . . Monostable BX (from pin 11), 1=timing, 0=timed-out
. . . . . * . Monostable AY (from pin 6), 1=timing, 0=timed-out
. . . . . * Monostable AX (from pin 3), 1=timing, 0=timed-out

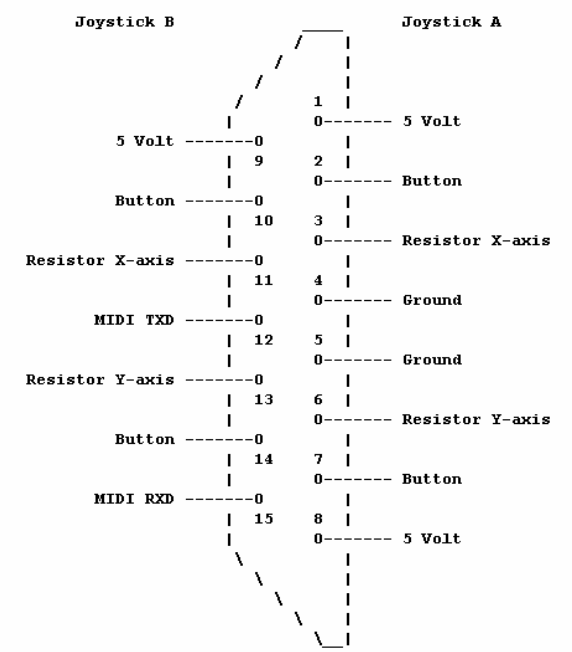
Gambar 2.13. *Address Joystick*

Sumber: Engdahl, Tomi. *PC Analogue Joystick Interface*. Mei 2002. <http://www.epenorama.net/documents/joystick/pc_joystick.html>.

	UP			
	x = 0 kohm y = 0 kohm	x = 50 kohm y = 0 kohm	x = 100 kohm y = 0 kohm	
LEFT	x = 0 kohm y = 50 kohm	x = 50 kohm y = 50 kohm	x = 100 ohm y = 50 ohm	RIGHT
	x = 0 kohm y = 100 kohm	x = 50 kohm y = 100 kohm	x = 100 kohm y = 100 kohm	
	DOWN			

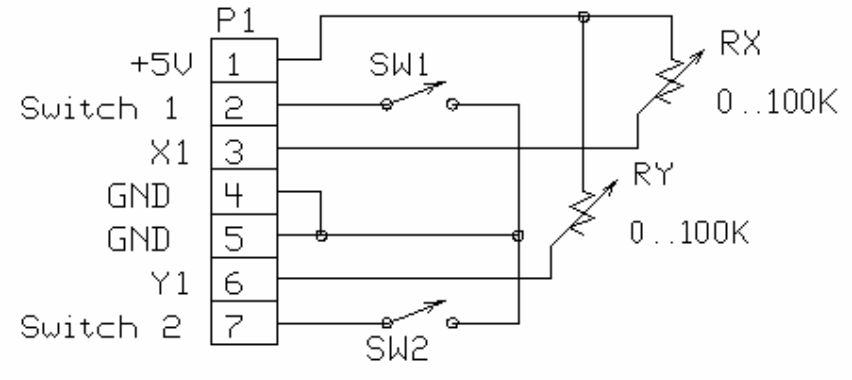
Gambar 2.14. *Position Joystick*

Sumber: Engdahl, Tomi. *PC Joystick Interface Circuits*. Januari 2005. <http://www.epenorama.net/documents/joystick/pc_circuits.html>.



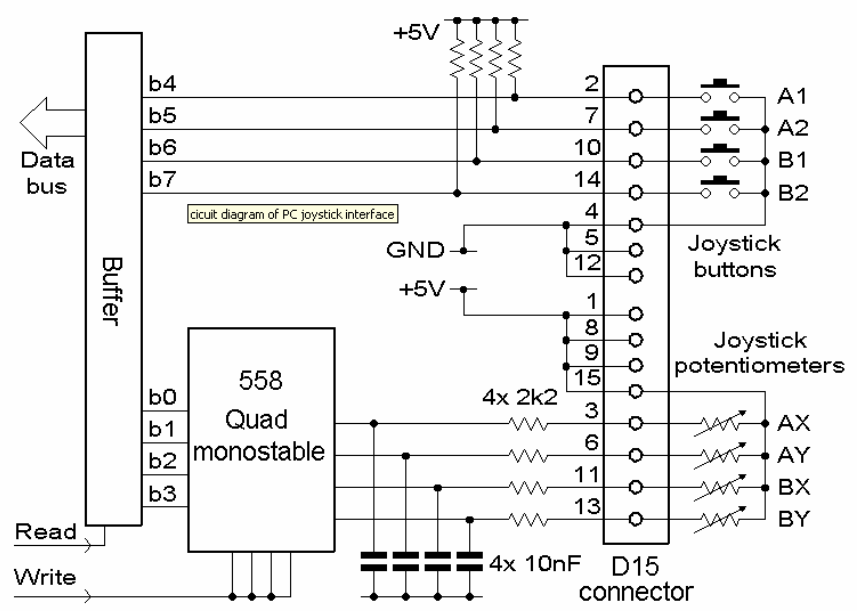
Gambar 2.15. *Pin Joystick*

Sumber: Engdahl, Tomi. *PC Analogue Joystick Interface*. Mei 2002. <http://www.epenorama.net/documents/joystick/pc_joystick.html>.



Gambar 2.16. *Circuit Diagram Joystick*

Sumber: Engdahl, Tomi. *PC Analogue Joystick Interface*. Mei 2002. <http://www.epenorama.net/documents/joystick/pc_joystick.html>.



Gambar 2.17. *Circuit Diagram of PC Joystick Interface*

Sumber: Engdahl, Tomi. *PC Analogue Joystick Interface*. Mei 2002. <http://www.epenorama.net/documents/joystick/pc_joystick.html>.