BAB II
LANDASAN TEORI

Dalam bab ini akan dijelaskan mengenai teori-teori tentang *mobile robot* serta penunjang untuk merancang sebuah prototipe *mobile robot* penjinak bom, baik teori perangkat keras maupun perangkat lunak yang akan digunakan.

2.1 ROBOT PENJINAK BOM

Salah satu langkah untuk mencegah terjadinya ledakan bom adalah menjinakkan bom tersebut sebelum meledak. Namun menjinakkan bom merupakan salah satu pekerjaan yang memiliki risiko tinggi, karena bom tersebut dapat meledak kapan saja. Untuk mengurangi risiko jatuhnya korban jiwa dalam upaya menjinakkan bom, diperlukan sebuah *security robot* yang dapat menggantikan tugas manusia.

Penanganan bom dan/atau bahan peledak juga dapat dilakukan dengan menggunakan robot. Belakangan, dikembangkan robot untuk menekan risiko tersebut. Sebenarnya penggunaan robot semacam itu oleh pasukan penjinak bahan peledak atau Tim Gegana Polri sudah dilakukan sejak lama. Sayangnya, robot-robot yang digunakan masih produk *impor*, antara lain berasal dari Israel dan Inggris.

Pemanfaatan *security* robot semacam itu yang paling menghebohkan akhir-akhir ini tentu saja terjadi ketika penggerebekan teroris di Dusun Beji, Kedu, Temanggung, Jawa Tengah. Robot penjinak bom tersebut berjalan perlahan melintasi halaman dan menyelinap ke dalam rumah target. Robot ini mampu mengambil gambar, video bahkan memindahkan benda. Dengan pergerakannya membopong kamera, robot ini memuluskan langkah polisi dalam membekuk orang yang bersembunyi dalam rumah di tengah ladang jagung yang berhawa dingin tersebut, yang diduga sebagai mastermind dari serangkaian tindakan terror yang terjadi di Indonesia selama satu dekade terakhir. [14]

2.1.1 Kriteria Mobile Robot Penjinak Bom

Robot penjinak bom diharuskan memiliki kehandalan seperti, gerakan robot harus presisi, mempunyai simpangan getaran *body* yang kecil ketika robot mengambil atau membawa sebuah objek, dapat digunakan untuk segala medan, memungkinkan dikontrol jarak jauh, dapat bergerak cepat dan tepat dalam menjinakan bom. [14]

2.2 PERANGKAT KERAS *(HARDWARE)*

2.2.1 Mikrokontroler ATmega8535

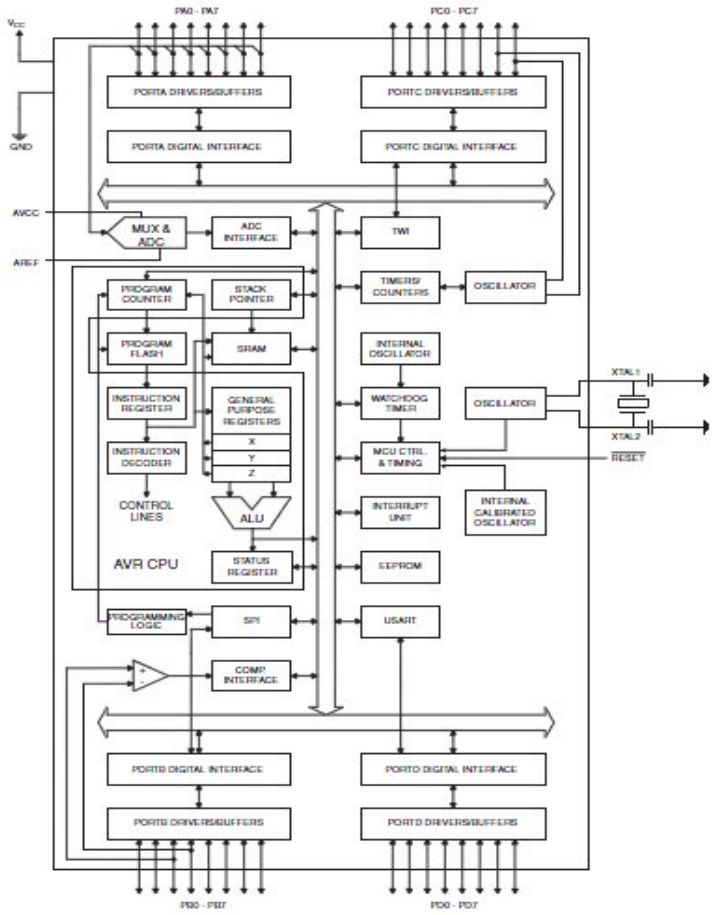
Mikrokontroler AVR *(Alf and Vegard's Risc prosesor)* dari Atmel ini menggunakan arsitektur RISC *(Reduced Instruction Set Computer)* yang artinya prosesor tersebut memiliki set instruksi program yang lebih sedikit dibandingkan dengan MCS-51 yang menerapkan arsitektur CISC *(Complex Instruktion Set Computer)*. [12]

Mikrokontroler ini berfungsi sebagai pusat pengendali robot dan sebagai pengambil keputusan dari data-data yang dikirimkan oleh sensor-sensor. Mikrokontroler yang digunakan adalah ATmega8535.

Mikrokontroler ATmega8535 memiliki fitur-fitur utama, seperti berikut:

1. Saluran I/O sebanyak 32 buah yaitu Port A, Port B, Port C dan Port D.
2. ADC 10 bit sebanyak 8 saluran.
3. Tiga unit *Timer/Counter* dengan kemampuan pembandingan
4. CPU yang terdiri atas 32 buah register.
5. *Watchdog Timer* dengan osilator internal.
6. SRAM sebesar 512 *byte*.
7. Memori *flash* sebesar 8 *Kbyte* dengan kemampuan *Read While Write.*
8. Unit interupsi internal dan eksternal.
9. *Port* antarmuka SPI.
10. EEPROM sebesar 512 *byte* yang dapat diprogram saat operasi.
11. Antarmuka komparator analog.
12. *Port* USART untuk komunikasi serial.

2.2.1.1 Diagram Blok ATmega8535



Gambar 2.1 *Diagram Blok ATmega8535* [12]

2.2.1.2 Deskripsi Pin ATmega8535

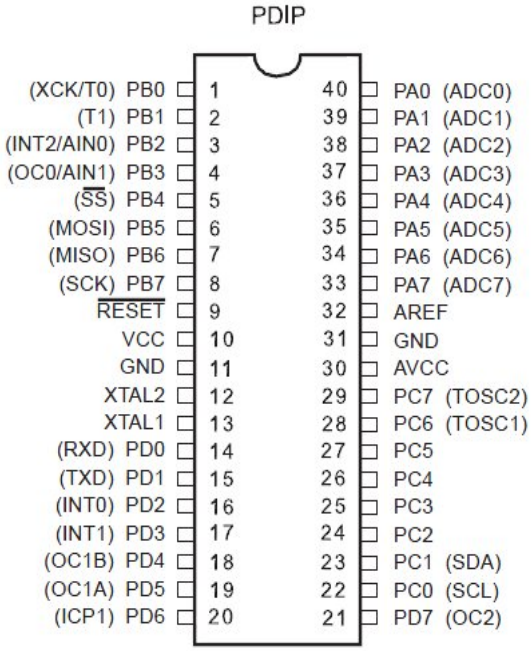
ATmega8535 memiliki 40 pin dengan konfigurasi pin seperti pada tabel 2.1 [12]

Tabel 2.1 *Deskripsi Pin ATmega8535*

Nama Pin	Fungsi
Vcc	Catu daya
GND	Ground
PortA (PA7…PA0)	Port I/O 8 bit dua arah dengan resistor pull-up internal Juga berfungsi sebagai masukan analog ke ADC (ADC0 s.d. ADC7)

PortB (PB7…PB0)	Port I/O 8 bit dua arah dengan resistor pull-up internal Fungsi khusus masing-masinh pin:	
	Port Pin	Fungsi lain
	PB0	T0 (Timer/Counter0 External Counter Input)
	PB1	T1 (Timer/Counter1 External Counter Input)
	PB2	AIN0 (Analog Comparator Positive Input)
	PB3	AIN1 (Analog Comparator Negative Input)
	PB4	SS (SPI Slave Select Input)
	PB5	MOSI (SPI Bus Master Output/Slave Input)
	PB6	MISO (SPI Bus Master Input/Slave output)
	PB7	SCK (SPI Bus Serial Clock)
PortB (PC7…PC0)	Port I/O 8 bit dua arah dengan resistor pull-up internal Dua pin yaitu PC6 dan PC7 berfungsi sebagai oascilator luar untuk Timer/Counter2	
PortB (PD7…PD0)	Port I/O 8 bit dua arah dengan resistor pull-up internal Fungsi khusus masing-masinh pin:	
	Port Pin	Fungsi lain:
	PD0	RXD (UART Input Line)
	PD1	TXD (UART Output Line)
	PD2	INT0 (External Interrupt 0 Input)
	PD3	INT1 (External Interrupt 1 Input)
	PD4	OC1B (Timer/Counter1 Output CompareB Match Output)
	PD5	OC1A (Timer/Counter1 Output CompareA Match Output)
	PD6	ICP (Timer/Counter1 Input Capture Pin)
	PD7	OC2 (Timer/Counter2 Output Compare Match Output)
RESET	Masukan reset, sebuah reset akan terjadi jika pin ini diberi logika rendah (low) melebihi periode minimum yang diperlukan	
XTAL1	Masukan ke inverting oscillator amplifier dan masukan ke rangkaian clock internal	
XTAL2	Keluaran dari inverting oscillator amplifier	
AVCC	Catu daya untuk port A dan ADC	
AREF	Referensi masukan analog untuk ADC	
AGND	Ground Analog	

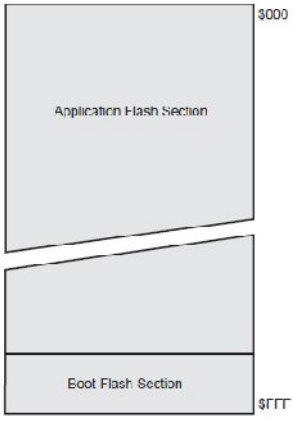
2.2.1.3 Konfigurasi Pin ATmega8535



Gambar 2.2 *Konfigurasi Pin ATmega8535* [12]

2.2.1.4 Peta Memori

Arsitektur AVR terdiri atas dua memori utama, yaitu data memori dan program memori. Sebagai tambahan fitur dari ATmega8535, terdapat EEPROM 512 byte sebagai memori data dan dapat diprogram saat operasi. [12]



Gambar 2.3 *Peta Memori Program*

2.2.1.5 Register

Tabel 2.2 *Register ATmega8535*

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C
0x3E (0x5E)	SPH	–	–	–	–	–	SP10	SP9	SP8
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
0x3C (0x5C)	OCR0	Timer/Counter0 Output Compare Register							
0x3B (0x5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE
0x3A (0x5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
0x37 (0x57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
0x35 (0x55)	MCUCR	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00
0x34 (0x54)	MCUCSR	–	ISC2	–	–	WDRF	BORF	EXTRF	PORF
0x33 (0x53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
0x32 (0x52)	TCNT0	Timer/Counter0 (8 Bits)							
0x31 (0x51)	OSCCAL	Oscillator Calibration Register							
0x30 (0x50)	SFIOR	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte							
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte							
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte							
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte							
0x29 (0x49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte							
0x28 (0x48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte							
0x27 (0x47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte							
0x26 (0x46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte							
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
0x24 (0x44)	TCNT2	Timer/Counter2 (8 Bits)							
0x23 (0x43)	OCR2	Timer/Counter2 Output Compare Register							
0x22 (0x42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB
0x21 (0x41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0
0x20[1] (0x40)[1]	UBRRH	URSEL	–	–	–	UBRR[11:8]			
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8
0x1E (0x3E)	EEARL	EEPROM Address Register Low Byte							
0x1D (0x3D)	EEDR	EEPROM Data Register							
0x1C (0x3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE
0x1B (0x3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
0x1A (0x3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
0x19 (0x39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0x15 (0x35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x14 (0x34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
0x13 (0x33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x0F (0x2F)	SPDR	SPI Data Register							
0x0E (0x2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
0x0C (0x2C)	UDR	USART I/O Data Register							
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
0x09 (0x29)	UBRRL	USART Baud Rate Register Low Byte							
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
0x05 (0x25)	ADCH	ADC Data Register High Byte							
0x04 (0x24)	ADCL	ADC Data Register Low Byte							
0x03 (0x23)	TWDR	Two-wire Serial Interface Data Register							
0x02 (0x22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
0x01 (0x21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0

Tabel 2.2 di atas menunjukan register-register yang ada pada mikrokontroler ATmega8535. [12]

2.2.1.6 Timer/Counter

Mikrokontroler ATmega8535 memiliki tiga buah timer, yaitu *Timer/Counter0* (8bit), *Timer/Counter1* (16 bit), *Timer/Counter2* (8 bit). [12]

2.2.1.6.1 Timer/Counter0

Timer/Counter0 adalah *Timer/Counter* 8 bit yang multifungsi. Fitur-fitur dari *Timer/Counter0* pada ATmega8535 adalah sebagai berikut :

1. *Counter* kanal.
2. *Timer* di-nol-kan saat proses pembandingan tercapai (*compare match*).
3. Sebagai pembangkit gelombang PWM.
4. Sebagai pembangkit frekuensi.
5. *Clock pre-scaler* 10 bit.
6. Sumber interupsi dari *compare match* (OCF0) dan *overflow* (TOV0).

Pengaturan *Timer/Counter0* diatur oleh TCCR0 (*Timer/Counter Control Register0*) yang dapat dilihat pada gambar 2.4.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FOC0	WGM00	COM00	COM01	WGM01	CS02	CS01	CS00

Gambar 2.4 *Register TCCR0*

Penjelasan untuk tiap bit-bitnya adalah sebagai berikut:

a. Bit 7 – FOC0 : *Force Output Compare.*

b. Bit 6,3 – WGM01; WGM00 : *Wafeform Generation Unit.*

Bit ini mengontrol kenaikan dari counter, sumber dari nilai maksimum *counter* dan tipe dari jenis *timer/counter* yang dihasilkan, yaitu mode normal, *clear timer*, *mode compare match*, dan dua tipe dari PWM (*Pulse Width Modulation*).

c. Bit 5,4 – COM01; COM00 : *Compare Match Output Mode.*

Bit ini mengontrol pin OC0 (*Pin Output Compare*). Apabila kedua bit ini nol atau *clear*, maka pin OC0 berfungsi sebagai pin biasa. Bila salah satu bit set,

maka fungsi pin ini tergantung pada pengaturan bit pada WGM00 dan WGM01.

d. Bit 2,1,0 – CS02; CS01; CS00 : *Clock Select.*

Ketiga bit tersebut untuk memilih sumber *clock* yang akan digunakan oleh *Timer/Counter0.* [12]

2.2.1.6.2 Timer/Counter1

Timer/Counter1 adalah *Timer/Counter* 16 bit yang memungkinkan program pewaktuan lebih akurat. Fitur-fitur dari *Timer/Counter1* pada ATmega8535 adalah sebagai berikut :

1. Desain 16 bit, sehingga memungkinkan untuk menghasilkan PWM 16 bit.
2. Dua buah unit pembanding.
3. Dua buah register pembanding.
4. Satu buah *input capture unit.*
5. *Timer* di-nol-kan saat proses pembanding tercapai (*match compare*).
6. Dapat menghasilkan gelombang PWM.
7. Periode PWM yang dapat diubah-ubah.
8. Sebagai pembangkit frekuensi.
9. Empat buah sumber interupsi (TOV1, OCF1A, OCF1B dan ICF1).

Pengaturan Timer/Counter1 diatur oleh TCCR1A (Timer/Counter1 Control RegisterA) dan TCCR1B (Timer/Counter1 Control RegisterB).

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

Gambar 2.5 *Register TCCR1A*

Penjelasan untuk tiap bit-bitnya adalah sebagai berikut:

a. Bit 7,6 – COM1A1; COM1A0: *Compare Output Mode* untuk kanal A.

b. Bit 5,4 – COM1B1; COM1B0: *Compare Output Mode* untuk kanal B.

COM1A1, COM1A0, COM1B1 dan COM1B0 mengatur pin pembanding keluaran (*Output Compare pins*), yaitu pin OC1A dan OC1B. Jika salah satu

atau keduanya dari COM1B1 dan COM1A0 diset 1, maka OC1A akan terhubung (aktif) sebagai pin keluaran. Begitu juga jika salah satu atau keduanya dari COM1B1 dan COM1B0 diset 1, maka OC1B akan terhubung (aktif) sebagai pin keluaran. Sebelum mengaktifkan OC1A dan OC1B, pin-pin yang bersangkutan tersebut harus diatur sebagai keluaran atau DDR (*Data Direction Register*) dari pin tersebut diset sebagai keluaran. Pengaturan COM1A1, COM1A0, COM1B1 dan COM1B0 harus menyesuaikan pengaturan bit WGM13, WGM12, WGM11, dan WGM10 terlebih dahulu.

c. Bit 3,2 – FOC1A; FOC1B : *Force Output Compare* untuk kanal A dan B. Bit ini harus diset 0 ketika TCRR1A dioperasikan sebagai fungsi PWM.

d. Bit 1,0 – WGM11; WGM10 : *Waveform Generation Mode.*

Bit-bit ini satu kesatuan dengan WGM13, WGM12 yang terdapat pada register TCCR1B. Bit-bit tersebut mengatur urutan perhitungan dari counter, menentukan nilai TOP (nilai maksimal dari pengaturan counter), dan menentukan pilihan tipe pengoperasian *Timer/Counter1*.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ICN1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

Gambar 2.6 *Register TCCR1B*

Dari register tersebut, bit-bit yang berperan dalam pengaturan PWM adalah bit 4, bit 3, bit 2, bit 1, bit 0.

a. Bit 2, 1, 0 – CS12; CS11; CS00 : *Clock SelectI.*

Ketiga bit tersebut untuk memilih sumber clock yang akan digunakan oleh Timer/Counter1.

b. Bit 4, 3 – WGM13; WGM12 : *Waveform Generation Mode.*

Bit-bit satu kesatuan dengan bit WGM11 dan bit WGM10 yang terdapat pada register TCCR1A. Seperti pada bit WGM11 dan bit WGM10, bit WGM13, dan bit WGM12 berfungsi mengatur urutan perhitungan dari counter, menentukan nilai TOP (nilai maksimal dari pengaturan counter), dan menentukan pilihan tipe pengoprasian Timer/Counter1. [12]

2.2.1.6.3 Timer/Counter2

Timer/Counter2 adalah *Timer/Counter* 8 bit yang multifungsi. Fitur-fitur dari *Timer/Counter2* pada ATmega8535 adalah sebagai berikut :

1. Sebagai *Counter* 1 kanal.
2. Timer di-nol-kan saat proses pembandingan tercapai (*match compare*).
3. Dapat menghasilkan gelombang PWM.
4. Sebagai pembangkit frekuensi.
5. *Clock* prescaler 10 bit.
6. Sumber interupsi dari *compare match* (OCF0) dan *overflow* (TOV0).

Pengaturan *Timer/Counter2* diatur oleh TCCR2 (*Timer/Counter2 Control Register*).

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

Gambar 2.7 *Register TCCR2*

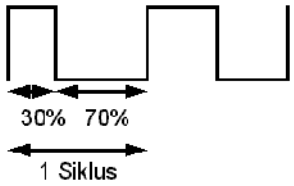
Penjelasan untuk tiap bit-bitnya adalah sebagai berikut:

a. Bit 7 – FOC2 : Force Output Compare
b. Bit 6,3 – WGM21; WGM20 : Wafeform Generation Unit
   Bit ini mengontrol kenaikan dari counter, sumber nilai maksimum counter dan tipe dari jenis timer/counter yang dihasilkan, yaitu mode normal, clear timer, mode compare match, dan dua tipe dari PWM (*Pulse Width Modulation*).
c. Bit 5,4 – COM21; COM20 : Compare Match Output Mode
   Bit ini mengontrol pin OC2 (*Output Compare*). Apabila kedua bit ini nol atau clear, maka pin OC2 berfungsi sebagai pin biasa. Bila salah satu bit diset, maka fungsi pin ini tergantung pada pengaturan bit pada WGM20 dan WGM21.
d. Bit 2,1,0 – CS22; CS21; CS20 : Clock Select
   Ketiga bit tersebut untuk memilih sumber clock yang akan digunakan oleh Timer/Counter2. [12]

2.2.1.7 PWM *(Pulse Width Modulation)*

Pulse Width Modulation (PWM) digunakan untuk mengatur kecepatan motor DC sesuai yang diinginkan oleh penggunanya. Dalam PWM gelombang kotak, frekuensi tinggi dibangkitkan sebagai *output* digital. Untuk contoh, sebuah *port bit* secara kontinyu melakukan kegiatan saklar *on* dan *off* pada frekuensi yang relatif tinggi. Selanjutnya, bila sinyal diumpankan pada *Low Pass Filter* (LPF), tegangan pada *output filter* akan sama dengan *Root Mean Squere* (RMS) dari sinyal gelombang kotak. Selanjutnya tegangan RMS dapat divariasi dengan mengubah *duty cycle* dari sinyal.

Duty cycle menyatakan fraksi waktu sinyal pada keadaan logika *high* dalam satu siklus. Satu siklus diawali oleh transisi *low* to *high* dari sinyal dan berakhir pada transisi berikutnya. Selama satu siklus, jika waktu sinyal pada keadaan *high* sama dengan *low* maka dikatakan sinyal mempunyai *duty cycle* 50 %. *Duty cycle* 20 % menyatakan sinyal berada pada logika 1 selama 1/5 dari waktu total. [12]



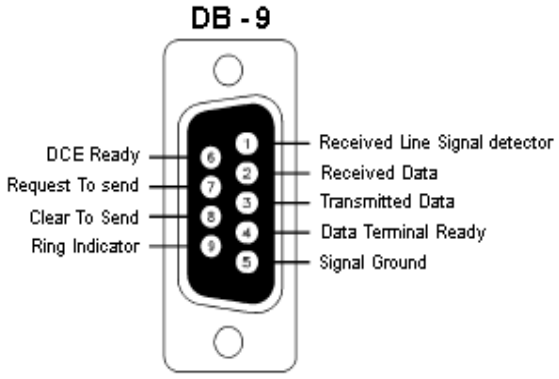
Gambar 2.8 *Duty Cycle 30 %*

2.2.2 Port Serial/RS-232

Port serial lebih sulit ditangani dari pada *port* paralel karena peralatan yang dihubungkan ke *port* serial harus berkomunikasi dengan menggunakan transmisi serial, sedangkan data di komputer diolah secara paralel. Sehingga, data dari / dan ke *port* serial harus dikonversikan ke (dan dari) bentuk paralel untuk bisa digunakan secara *hardware* hal ini bisa digunakan oleh UART *(Universal Asynchronus Receiver Transmitter).* [9]

Adapun keunggulan menggunakan *port* serial dari pada *port* paralel sebagai transfer data yaitu :

1. Kabel *port* serial bisa lebih panjang dibandingkan kabel *port* paralel. Hal ini karena *port* serial mengirimkan logika 1 sebagai –3 Volt hingga –25 Volt dan logika 0 sebagai +3 Volt hingga +25 Volt, sedangkan *port* paralel menggunakan TTL, yakni hanya 0 Volt untuk logika 0 dan +5 Volt untuk logika 1. ini berarti *port* serial memiliki rentang kerja 50 Volt sehingga kehilangan daya karena panjang kabel bukan merupakan masalah serius jika dibandingkan dengan *port* paralel.
2. Transmisi serial memerlukan lebih sedikit kabel dibandingkan dengan transmisi paralel.
3. *Port* serial memungkinkan untuk berkomunikasi dengan menggunakan Infra Red.



Gambar 2.9 *Konfigurasi Port Serial DB-9*

EIA *(electronic Industry association)* mengeluarkan spesifikasi listrik untuk standar RS-232 yaitu:

1. *Space* (logika 0) antara +3 sampai +15 Volt.
2. *Mark* (logika 1) antara –3 sampai –15 Volt.
3. Daerah antara +3 Volt dan –3 Volt tidak ditetapkan.
4. Tegangan rangkaian terbuka tidak boleh melebihi 25 Volt (terhadap *Ground*).
5. Arus pada rangkaian tertutup *(Short Circuit)* atau hubung singkat tidak boleh melebihi 500mA.

Tabel 2.3 *Konfigurasi pin dan nama sinyal konektor port serial DB9*

Nama Pin	Nama Sinyal	Direction	Keterangan
1	DCD	In	Data Carrier Detect/Received Line Signal Detect
2	RxD	In	Received Data
3	TxD	Out	Transmiter Data
4	DTR	Out	Data Terminal Ready
5	GND	-	Ground
6	DSR	In	Data Set Ready
7	RTS	Out	Request to Send
8	CTS	In	Clear to Send
9	RI	In	Ring Indicator

Keterangan mengenai saluran RS-232 pada konektor DB9 adalah sebagai berikut :

1. *Received Line Signal Detect*, dengan saluran ini DCE memberitahukan ke DTE bahwa terminal masukan ada data masukan.
2. *Received Data*, digunakan DTE menerima data dari DCE.
3. *Transmite Data*, digunakan DTE mengirimkan data ke DCE.
4. *Data Terminal Ready*, pada saluran ini DTE memberitahukan kesiapan sinyalnya.
5. *Signal Ground*, saluran Ground.
6. *Ring Indicator,* pada saluran ini DCE memberitahukan ke DTE bahwa stasiun menghendaki hubungan dengannya.
7. *Clear to Send,* dengan saluran ini DCE memberitahukan ke DTE boleh mengirimkan data.
8. *Request to Send*, dengan saluran ini DCE diminta mengirimkan data oleh DTE.
9. *DCE Ready,* sinyal aktif pada saluran ini menunjukan bahwa DCE sudah siap. [9]

2.2.3 Radio Frekuensi

Radio Frekuensi disingkatnya RF. Bagian dari spektrum elektromagnetis dengan frekuensi antara 10 kilohertz dan 3000 gigahertz. Perbandingan frekuensi dapat dilihat pada tabel 2.4. [4]

Tabel 2.4 *Range Frekuensi*

Nama band	Singkatan	Band ITU	Frekuensi	Panjang gelombang
			< 3 Hz	> 100.000 km
Extremly low frequency	ELF	1	3–30 Hz	100.000-10.000 km
Super low frequency	SLF	2	30-300 Hz	10.000-1000 km
Ultra low frequency	ULF	3	300-3000 Hz	1000-100 km
Very low frequency	VLF	4	3-30 KHz	100-10 km
Low frequency	LF	5	30-300 KHz	10-1 km
Medium frequency	MF	6	300-3000 KHz	1000 – 100 m
High frequency	HF	7	3-30 MHz	100-10 m
Very High frequency	VHF	8	30-300 MHz	10-1 m
Ultra high frequency	UHF	9	300-3000 MHz	1000-100 mm
Super high frequency	SHF	10	3-30 GHz	100-10 mm
Extremly high frequency	EHF	11	30-300 GHz	10-1 mm
			Di atas 300 GHz	< 1 m

2.2.4 Modul Penggerak

Roda yang dihubungkan menggunakan *gear* plastik pada robot digerakkan menggunakan dua buah motor *gear* DC yang dipasang pada roda sebelah kiri dan kanan. Pemilihan motor *gear* DC didasarkan pada putaran dan torsi yang lebih besar dibandingkan dengan motor *stepper* atau motor *servo*, juga didasarkan atas ketersediaan di pasaran selain harga murah juga banyak variasinya. [7]



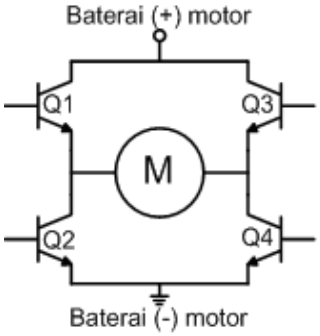
Gambar 2.10 *Motor DC*

2.2.4.1 Driver Motor

Motor *gear* DC tidak dapat dikendalikan secara langsung oleh mikrokontroler, karena kebutuhan arus yang besar sedangkan keluaran arus dari mikrokontroler sangat kecil. *Driver* Motor merupakan alternatif yang dapat digunakan untuk menggerakkan motor DC. Ada dua cara yang dapat digunakan untuk mendapatkan *driver* Motor: [7]

2.2.4.1.1 Driver Motor Transistor

Motor DC biasanya dikontrol menggunakan konfigurasi transistor yang dikenal dengan istilah *H-Bridge*. Konfigurasi ini biasanya menggunakan 4 buah transistor NPN atau dua transistor NPN dan dua transistor PNP.



Gambar 2.11 *Konfigurasi H-Bridge*

Tabel 2.5 *Tabel kebenaran konfigurasi H-Bridge*

	Maju	**Mundur**	**Stop**
Q1	1	0	0
Q2	0	1	0
Q3	0	1	0
Q4	1	0	0

Gambar 2.11 menunjukkan konfigurasi transistor NPN yang digunakan sebagai pengontrol motor DC. Arus yang mengalir ke motor DC polaritasnya dapat diatur dengan memberikan logika ke transistor Q1 sampai Q4. Pengaturannya seperti tabel kebenaran pada tabel 2.5 Transistor Q1 dan Q2 atau

Q3 dan Q4 tidak diperbolehkan kondisi keduanya dalam keadaan *high* karena akan menyebabkan *short circuit* terhadap baterai. [7]

2.2.4.1.2 IC Driver Motor

L298 dan VNH3SP30 adalah contoh IC yang dapat digunakan sebagai *driver* motor DC. IC ini menggunakan prinsip kerja *H-Bridge*. Tiap *H-Bridge* dikontrol menggunakan level tegangan TTL yang berasal dari *output* mikrokontroler. L298 dapat mengendalikan 2 buah motor DC sedangkan VNH3SP30 hanya dapat mengendalikan 1 buah motor saja. Akan tetapi kelebihan dari IC VNH3SP30 dapat mensuplay arus kontinyu sampai dengan 30 A dan tegangan maksimum sampai dengan 16 volt.

Pengaturan kecepatan kedua motor dilakukan dengan cara pengontrolan lama pulsa aktif (metode PWM - *Pulse Width Modulation*) yang dikirimkan ke rangkaian *driver* motor oleh modul kendali motor. *Duty cycle* PWM yang dikirimkan menentukan kecepatan putar motor DC. Gambar 2.12 menunjukkan bentuk IC VNH3SP30 dan L298 [7]



Gambar 2.12 *IC VNH3SP30 dan L298*

2.2.4.2 Motor Servo

Motor servo pada dasarnya adalah motor DC dengan kualifikasi khusus yang sesuai dengan aplikasi "*sevosing*" didalam teknik control. Dalam kamus *Oxfrod* istilah "servo" diartikan sebagai " *a mechanism that control a large mechanism* ".tidak ada sepisi baku yang disepakati untuk menyatakan bahwa suatu motor dc adalah motor servo. Namun secara umum dapat difinisikaan bahwa motor harus memilki kemampuan yang baik dalam mengatasi perubahan yang cepat dalam posisi dan kecepatan. Motor servo juga dikehendaki handal dalam beroperasi dalam lingkup torsi yang berubah - berubah. Berapa tipe motor

yang dijual dengan paket rangkaian drivernya telah memiliki rangkaia control kecepatan yang menyatu didalamnya. Putaran motor tidak lagi berdasarkan tegangan supplay ke motor, namun berdasarkan tegangan input khusus yang berfungsi sebagai referensi kecepatan output. Motor servo merupakan motor yang diatur dan dikontrol menggunakan pulsa. Motor standard ini memiliki tiga posisi yaitu posisi 0^0, posisi 90^0, dan posisi 180^0. [2]



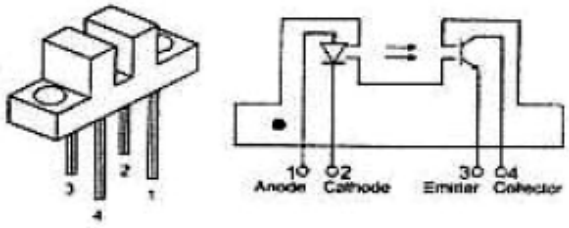
Gambar 2.13 *Motor Servo*

2.2.5 Sensor

Sensor adalah suatu alat untuk mengukur atau mendeteksi kejadian alam seperti sesuatu yang bergerak, panas, sinar dan mengubahnya menjadi representasi digital atau analog. Adapun sensor yang digunakan pada mobile robot diantaranya: [5]

2.2.5.1 Sensor Optocouler

Optocoupler merupakan gabungan dari Led (IR Led) dan photo transistor, yang dapat mengubah besaran cahaya menjadi besaran listrik. Ada banyak macam bentuk dari *optocoupler* yang ada salah satunya adalah *optocoupler* yang berbentuk seperti huruf U.

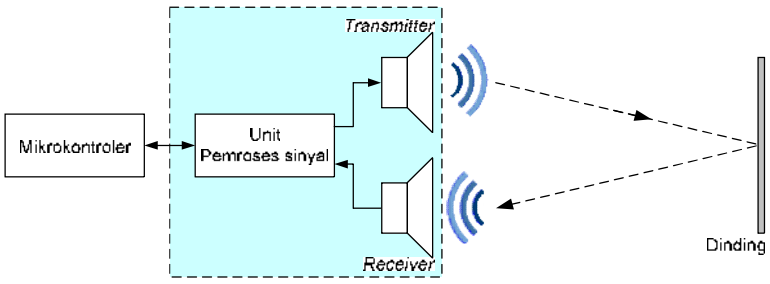


Gambar 2.14 *Sensor Optocoupler*

Pada saat cahaya dari led inframerah yang menuju ke phototransistor terhalang, cahaya dari led inframerah yang menuju ke phototransistor diterima maka phototransistor akan melewatkan arus yang melaluinya. Pada keadaan phototransistor menerima cahaya maka *optocoupler* akan menghasilkan beda potensial sebesar beberapa mV. [15]

2.2.5.2 Sensor Ultrasonik

Telinga manusia hanya bisa mendengar frekuensi bunyi antara 20 Hz sampai 20 kHz. Gelombang diatas itu tidak dapat didengar oleh telinga manusia yang dinamakan dengan gelombang *ultrasonik*. Gelombang jenis ini bisa dimanfaatkan sebagai sensor pengukur jarak tanpa terjadinya kontak fisik dengan objek yang diukur jaraknya (dinding). Sensor *ultrasonik* bekerja pada frekuensi 40Khz, mempunyai bagian pengirim gelombang (*transmitter*) dan bagian penerima gelombang (*receiver*). Jarak dapat diketahui dengan menghitung waktu yang dibutuhkan oleh gelombang *ultrasonik* sejak awal pengiriman sampai kembali ke penerima gelombang. [16]



Gambar 2.15 *Prinsip Kerja Ultrasonik*

2.2.6 Kamera

Salah satu kemampuan *mobile robot* penjinak bom diantaranya adalah dapat dikendalikan secara jarak jauh, ketika *mobile robot* berada jauh dengan *user* atau pengendali dan tidak terlihat oleh mata *user*, *mobile robot* masih harus dapat dikendalikan. Maka dibutuhkan suatu alat yang dapat dijadikan sebagai alat *monitoring* untuk memantau keadaan disekitar robot. Alat yang dapat dimanfaatkan untuk melihat keadaan sekitar robot diantaranya adalah kamera. Spesifikasi kamera yang dapat digunakan pada *mobile robot* diantaranya adalah kamera yang memiliki bobot ringan, kualitas gambar yang bagus, dan mudah untuk diletakan pada *mobile robot*. [17]

Kamera yang digunakan pada *mobile robot* menggunakan kamera *mini* CCTV *wireless*, dengan spesifikasi sebagai berikut:

- TV System : PAL/CCTR/NTSC
- Resolusi : 380 line
- Scan frequency : PAL 50 Hz, NTSC 60 Hz
- Minimum illumination : 3 lux
- 30 meter line of sight (tanpa halangan)
- 10-15 meter bila ada halangan



Gambar 2.16 *Kamera Mini Wireless*

2.2.7 Catu Daya

Catu daya memegang peranan yang sangat penting dalam hal perancangan sebuah robot. Tanpa bagian ini robot tidak akan berfungsi. Begitu juga bila pemilihan catu daya tidak tepat, maka robot tidak akan bekerja dengan baik.

Penentuan sistem catu daya yang akan digunakan ditentukan oleh banyak faktor, diantaranya:

1. Tegangan

   Setiap modul sensor atau aktuator tidak memiliki tegangan yang sama. Hal ini akan berpengaruh terhadap desain catu daya. Tegangan tertinggi dari salah satu modul sensor atau aktuator akan menentukan nilai tegangan catu daya.

2. Arus

   Arus memiliki satuan Ah (*Ampere-hour*). Semakin besar Ah, semakin lama daya tahan baterai bila digunakan pada beban yang sama.

3. Teknologi Baterai

   Baterai isi ulang ada yang dapat diisi hanya apabila benar-benar kosong, dan ada pula yang dapat di isi ulang kapan saja tanpa harus menunggu baterai tersebut benar-benar kosong.

Secara umum, ada beberapa jenis dan bentuk baterai yang dapat digunakan untuk sistem catu daya pada sebuah robot, diantaranya baterai *Nickel Metal Hydride* (Ni-MH). Baterai ini mempunyai teknologi terbaik untuk *rechargeable* baterai, yakni dapat diisi ulang lebih dari 400 kali serta memiliki tahanan dalam yang rendah dengan tegangan kerja sebesar 1.2 volt, sehingga dapat memberikan arus yang relatif besar. Jika digunakan untuk beban yang berat, baterai ini dapat menjadi panas.

Kapasitas simpan baterai Ni-MH ini sangat bervariasi, yakni sampai 2700 mAh. Berikut ini adalah contoh sebuah baterai Ni-MH 2700 mAh. [3]



Gambar 2.17 *Baterai Ni-MH 2700mAh*

Untuk pengisian baterai ini dapat dilakukan kapan saja, namun untuk hasil yang lebih baik, setelah beberapa kali pengisian, baterai ini harus dikosongkan terlebih dahulu sebelum diisi. Selain jenis baterai Ni-MH, masih banyak lagi jenis baterai yang tersedia di pasaran dengan spesifikasi yang beragam dan dapat digunakan untuk catu daya pada sebuah robot. Diantaranya baterai Ni-CAD, Alkaline, Lithium, Lead Acid dan sebagainya, yang masing-masing mempunyai kelebihan dan kekurangannya.

2.3 PERANGKAT LUNAK *(SOFTWARE)*

Perangkat lunak *(software)* merupakan faktor penting dalam tahap perancangan robot. Perangkat lunak ini berupa algoritma gerak dan tugas robot dalam bentuk *listing* program yang ditanamkan kedalam mikrokontroler serta *software* yang menunjang dalam perancangan robot. Program dapat bermacam-macam bentuk versi dan bahasa pemrogramannya, sesuai dengan spesifikasi dari mikrokontroler yang digunakan.

Mikrokontroler ATmega8535 dapat menggunakan bahasa pemrograman *basic*, *Software* yang digunakan adalah Bascom AVR. Selain *software* untuk bahasa pemrograman mikrokontroler dibutuhkan *software interface* antara *mobile robot* dengan kendali PC yaitu *software Visual Basic* dan algoritma untuk mengendalikan gerak robot agar lebih stabil menggunakan algoritma Kendali PI.

2.3.1 BASCOM AVR

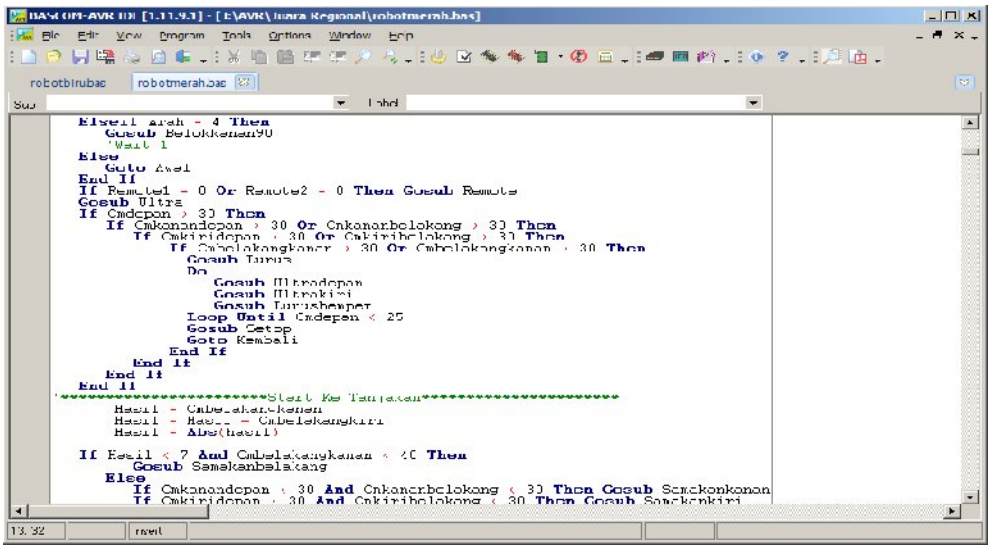
Instruksi yang dapat digunakan pada *editor* Bascom-AVR relatif cukup banyak dan bergantung dari tipe dan jenis AVR yang digunakan. Berikut ini beberapa instruksi-instruksi dasar yang dapat digunakan pada mikrokontroler ATmega8535.

Tabel 2.6 *Beberapa Instruksi Dasar BascomAVR*

Instruksi	Keterangan
DO…LOOP	Perulangan
GOSUB	Memanggil prosedur
IF…THEN	Percabangan
FOR..NEXT	Perulangan

WAIT	Delay atau waktu tunda detik
WAITMS	Delay atau waktu tunda milidetik
WAITUS	Delay atau waktu tunda mikrodetik
SELECT…CASE	Pencabangan
PULSOUT	Membangkitkan pulsa
PULSIN	Menerima/membaca pulsa yang diterima
GOTO	Menuju/loncat ke alamat memori tertentu

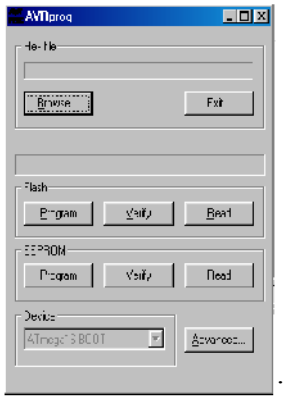
Bascom-AVR adalah *software* yang khusus dibuat untuk keluarga AVR, *software* ini berjalan pada sistem operasi *Windows*. *Software* ini dapat berjalan pada komputer dengan sistem minimum, tanpa harus membutuhkan spesifikasi komputer yang canggih. Gambar 2.18 menunjukkan tampilan jendela program Bascom-AVR yang berjalan pada sistem operasi *Windows*. [12]



Gambar 2.18 *Tampilan Editor Bascom-AVR*

2.3.2 AVRprog

AVRprog merupakan *software* yang dapat digunakan untuk memprogram mikrokontroler ATmega8535 dengan menggunakan AVR910. Berikut ini adalah tampilan jendela program AVRprog [12]



Gambar 2.19 *Tampilan Programmer AVRprog*

2.3.3 Software Visual Basic V6.0

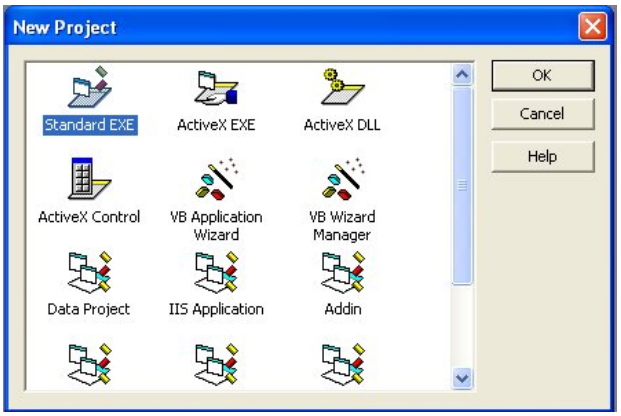
Perancangan *software* dititik beratkan pada pembangunan sebuah program *interface* yang *user friendly* dan yang terpenting adalah *software* harus mampu berkomunikasi dengan *hardware* sehingga dapat menyampaikan informasi yang sesuai. Pada sistem ini *software* yang digunakan adalah *Visual Basic 6.0.*

Bahasa pemrograman adalah bahasa yang dimengerti oleh *object* untuk melakukan tugas-tugas tertentu, salah satu contoh bahasa *Visual Basic*. Bahasa pemrograman *Visual Basic* yang dikembangkan oleh *Microsoft* sejak tahun 1991 merupakan pengembangan dari pendahulunya yaitu bahasa pemrograman BASIC (*Baginners All-purpose Symbolic Instruction Code*) yang dikembangkan pada era 1950-an. *Visual Basic* merupakan salah satu *Development Tool* yaitu alat bantu untuk membuat berbagai macam program *object*, khususnya yang menggunakan sistem operasi *Windows*, juga salah satu bahasa pemrograman *object* yang mendukung *object* (*Object Oriented Programming* = OOP). Dalam pemrograman berbasis obyek (OOP) kita perlu mengenal istilah *object*, *property, method* dan *event.* Berikut adalah keterangan mengenai hal tersebut diatas:

1. *Object* adalah komponen didalam sebuah program
2. *Property* adalah karakteristik yang dimiliki oleh *object*.
3. *Method* adalah aksi yang dapat dilakukan oleh *object*.
4. *Event* adalah kejadian yang dapat dialami oleh *object*.

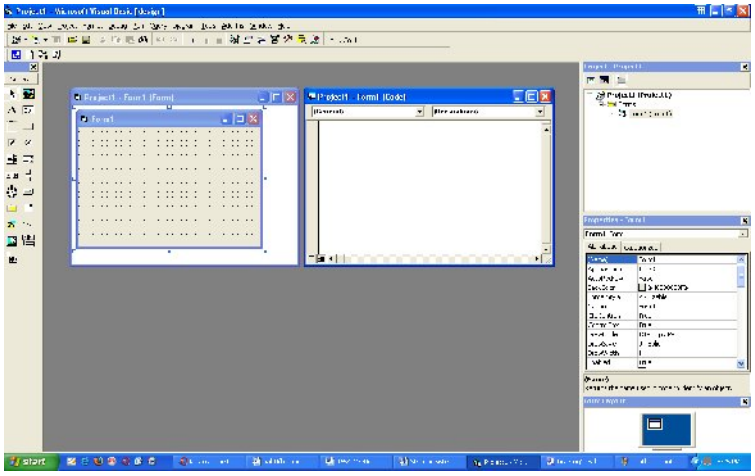
Seperti program berbasis *Windows* lainnya, *Visual Basic* terdiri dari banyak jendela (*windows*) ketika kita akan melalui *Visual Basic* sekumpulan *windows* yang saling berkaitan inilah yang disebut dengan *Integrated Development Environment* (IDE). Program yang berbasis *windows* bersifat *Event-Driven*, artinya program bekerja berdasarkan *event* yang terjadi pada *object* di dalam program tersebut, misalnya jika seorang *user* mengklik sebuah tombol maka program akan memberikan "reaksi" terhadap *event* klik tersebut. Program akan memberikan reaksi sesuia dengan kode-kode program yang dibuat untuk suatu *event* pada *object* tertentu. Pada waktu memulai *Visual Basic* beberapa *windows* kecil berada di dalam sebuah *windows* besar (*windows* induk), bentuk inilah yang dikenal dengan format *Multiple Document Interface* (MDI).

Pada gambar 2.21 memperlihatkan tampilan awal saat akan membuat sebuah *New Project* pada *Visual Basic 6.0.* [6]



Gambar 2.20 *Tampilan Awal Visual Basic*

Pada gambar 2.20 memperlihatkan contoh tampilan *Integrated Development Environment* (IDE) pada sebuah *project Visual Basic* dengan sebuah *form* dan sebuah *Command Button*.



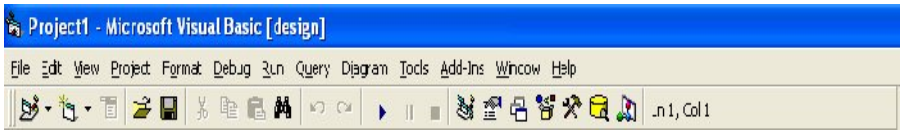
Gambar 2.21 *Tampilan Integrated Development Environment (IDE)*

2.3.3.1 Menu pilihan pada Visual Basic

2.3.3.1.1 Menu Bar/Toolbar

Menu Bar Visual Basic berisi semua perintah *Visual Basic* yang dapat dipilih untuk melakukan tugas tertentu, isi dari menu ini sebagaian hampir sama dengan program-program *windows* pada umumnya.

Toolbar adalah tombol-tombol (*shortcut*) yang mewakili suatu perintah tertentu pada *Toolbar*. Ini dapat dilihat pada gambar 2.22. [6]



Gambar 2.22 *Menu Bar/Toolbar*

2.3.3.1.2 Toolbox

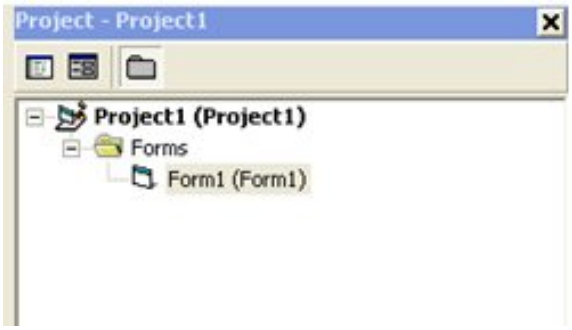
Toolbox adalah sebuah "kotak piranti" yang mengandung semua objek atau "*control*" yang dibutuhkan untuk membentuk suatu program aplikasi. Kontrol adalah suatu objek yang akan menjadi penghubung antara program aplikasi dan *user*nya yang kesemuanya harus diletakkan di dalam jendela *form*. *Toolbox* dapat disembunyikan untuk memberikan ruangan bagi element pada *Intergrated Development Environment* (IDE) lainnya. Sehingga lebih mempermudah desain maupun penulisan program. Ini dapat dilihat pada gambar 2.23. [6]



Gambar 2.23 *Toolbox*

2.3.3.1.3 Project Window

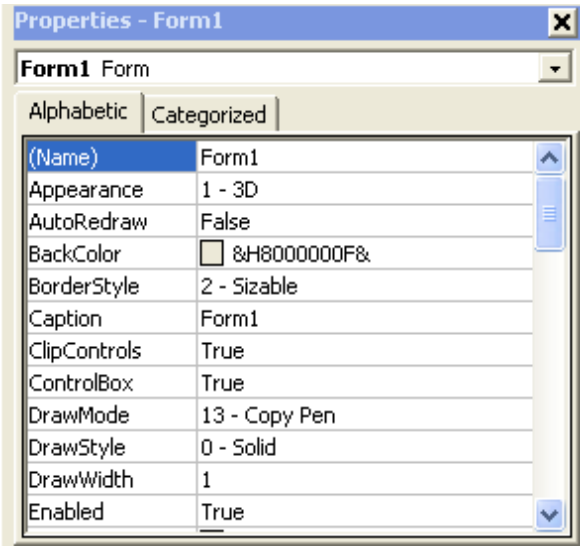
Window ini menampilkan seluruh *form*, *class*, *class module* dan komponen lainnya yang ada pada sebuah *project*. Ini dapat dilihat pada gambar 2.24. [6]



Gambar 2.24 *Project Window*

2.3.3.1.4 Property Window

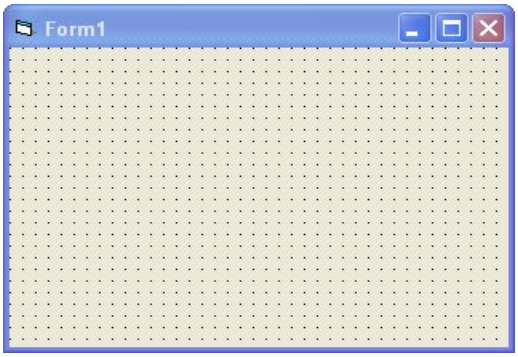
Window ini berisi seluruh *property* dari masing-masing *object* pada sebuah *project* yang meliputi *property form* dan kontrol-kontrol yang ada pada *form* tersebut. Beberapa *property* dapat diisikan pada tahap desain dan adapula *property* yang harus diisikan dengan menuliskan kode selama program dijalankan (*runtime*). Ini dapat dilihat pada gambar 2.25. [6]



Gambar 2.25 *Property Window*

2.3.3.1.5 Form

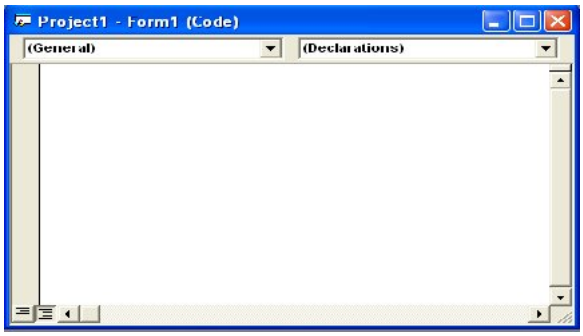
Form adalah sebuah atau beberapa *window* untuk pembuatan program aplikasi. *Form* ini dapat memuat berbagai macam *control* (tombol-tombol maupun teks) yang diperlukan dalam desain program yang sesuai dengan kebutuhan program. Ini dapat dilihat pada gambar 2.26. [6]



Gambar 2.26 *Form*

2.3.3.1.6 Code Window

Pada *window* inilah semua kode/perintah tentang program dituliskan dengan memperhatikan *event* apa saja yang diperlukan. Ini dapat dilihat pada gambar 2.28. [6]



Gambar 2.27 *Code Window*

2.3.3.1.7 Pengaksesan Port Serial dengan Visual Basic 6.0

Pada *port* serial komputer dengan *Visual Basic 6.0* dapat diakses dengan menggunakan MSComm. *Library* untuk pengaksesan *port* serial melalui kontrol MSComm yaitu Mscomm32.ocx. Kontrol MSComm pada jendela *toolbox* didapat dari komponen kontrol Microsoft Comm Control 6.0. kontrol MSComm menyediakan fasilitas komunikasi program aplikasi dengan *port* serial untuk mengirim dan menerima data melalui *port* serial. Setiap MSComm hanya menangani satu *port* serial sehingga jika ingin menggunakan lebih dari satu *port* serial, maka harus menggunakan MSComm sebanyak *port* serial yang dipakai.

Jumlah properti pada MSComm sangat banyak sehingga tidak akan dibahas secara keseluruhan. Namun hanya membahas beberapa properti yang cukup sesuai dengan kebutuhan saja properti-properti yang sering dipakai adalah sebagai berikut :

- *CommPort* : digunakan untuk menentukan nomor *port* serial yang akan dipakai.
- *Setting* : digunakan untuk menset nilai *baud rate*, *pariti*, jumlah bit data, dan jumlah *bit stop*.
- *PortOpen* : digunakan untuk membuka ataupun menutup *port* serial yang dihubungkan dengan MSComm ini.
- *Input* : digunakan untuk mengambil data string yang ada pada *buffer* penerima.
- *Output* : digunakan untuk menulis data *string* pada *buffer* kirim.

Commport 1 :

```
Private Sub Form_Load ()
MSComm1.CommPort = 1
MSComm1.Settings = "9600,N,8,1"
MSComm1.PortOpen = True
MSComm1.Output =  “A”
MSComm1.PortOpen = False
End Sub
```

Kode-kode program pada prosedur di atas akan melakukan aksi sebagai berikut:

→ *Port* serial yang digunakan adalah Comm 1

→ *Setting* MSComm dengan *baud rate* 9600. tanpa bit paritas, jumlah data 8 bit dan jumlah *stop bit* adalah1

→ Membuka *port* serial Comm 1

→ Mengirim satu karakter (“A”)

→ Menutup kembali com serial yang dipakai

Even pada MSComm hanya mempunyai satu even saja yaitu even OnComm saja. Even OnComm dibangkitkan jika nilai properti dari CommEvent berubah yang mengindikasikan telah terjadi even pada *port* serial baik even komunikasi maupun even *error* berikut ini merupakan properti CommEvent, nilai-nilai properti ini tidak tersedia pada saat *design time*, tetapi hanya dapat dibaca pada saat *run time.*

- comEventFrame : *Hardware* mendeteksi adanya kesalahan framing.
- comEventRxParity : *Hardware* mendeteksi adanya kesalahan parity.
- comEventRxOver : *Buffer* penerima mengalami over flow, tidak ada ruang kosong lagi pada *buffer* penerima.
- comEventTxFull : *Buffer* pengiriman telah penuh.
- comEventOverrun : *Port* mengalami *overrun*

- comEventBreak : Sinyal *Break* dikirim
- comEventDCB : Mendapatkan kembali *Device Control Clock* (DCB) dari *port* serial.

Berikut adalah nilai–nilai properti dari even komunikasi pada CommEvent yaitu:

- commEvSend : Jumlah karakter pada *Buffer* kirim lebih sedikit dari pada nilai properti *Sthreshold*. Even ini akan dibangkitkan jika nilai properti *Sthreshold* tidak diisi dengan "0".
- comEvReceive : Telah diterima karakter sebanyak nilai properti *Rthreshold*. Even ini akan dibangkitkan terus menerus sampai data diambil dari *Buffer* penerima menggunakan perintah *Input*. Even ini akan dibangkitkan jika nilai pada properti *Rthreshold* tidak diisi "0".
- commEvCTS : Terjadi perubahan pada saluran *Clear to Send.*
- commEvDSR : Terjadi perubahan pada saluran *Data Set Ready*.
- commEvRing : Terjadi perubahan pada saluran *Carrier Detect.*
- comEvRing : Terdeteksi adanya sinyal *Ring.*
- comEvEOF : Karakter *End of File* diterima.

Berikut adalah contoh program pada penggunaan Even OnComm untuk berkomunikasi. Namun yang hanya dibaca hanya even comEvReceive saja, yang lainya diabaikan.

```
Private Static Sub MsComm1_OnCom ()
Dim Buffer as variant
Select Case MSComm1.CommEvent
        Case comEvReceive
                If  MSComm1,InbufferCount = 1 then
                        Buffer = CStr(MSComm1.Input)
                        Text1.Text  = asc(Midbuffer, 1,1))
                End If
```

End If

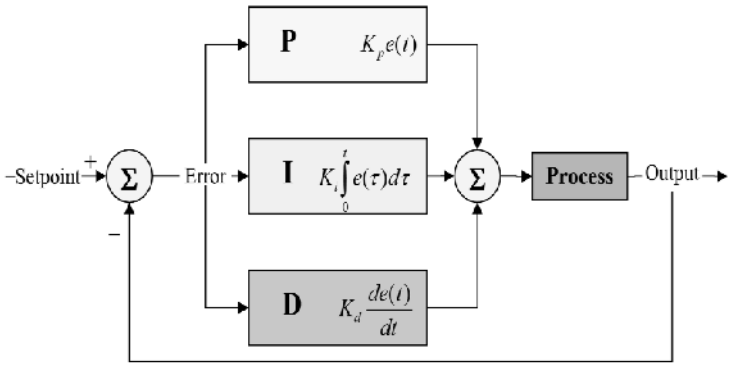
End Select

End Sub

Kode-kode program pada prosedur di atas akan melakukan aksi dengan mendeteksi even comEvReceive, kemudian menentukan apakah sudah diterima 1 karakter pada *buffer* penerima. [6]

2.4 PID *(Proporsional Integral Derivative)*

PID *(Proporsional Integral Derivative)* adalah suatu sistem kendali umpan balik yang banyak digunakan dalam sistem kendali industri. Sebuah pengendali PID digunakan untuk menghitung error (e) sebagai nilai perbedaan antara variabel proses (PV) yang diukur dan *setpoint* (SP) yang diinginkan. pengendali berupaya untuk meminimalkan kesalahan dengan menyesuaikan *proses input control* (CO). Parameter PID yang digunakan dalam perhitungan harus diset sesuai dengan sifat sistem. Gambar 2.28 merupakan diagram blok sistem kendali PID. [13]



Gambar 2.28 *Diagram Blok Sistem PID*

Perhitungan (algoritma) PID melibatkan tiga parameter yang terpisah yaitu *Proporsional*, *Integral* dan *Derivative*. Nilai *proporsional* menentukan reaksi terhadap kesalahan saat ini, nilai *integral* menentukan reaksi berdasarkan jumlah kesalahan baru-baru ini, dan *derivative* menentukan nilai reaksi berdasarkan tingkat di mana kesalahan telah berubah. Jumlah dari ketiga tindakan ini digunakan untuk mengatur proses melalui elemen kontrol seperti posisi kontrol

katup atau kecepatan motor yang disebut sebagai variabel manipulasi (MV). Secara umum perhitungan PID dinyatakan dalam persamaan 2.1 :

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) + K_d \frac{d}{dt} e(t) \quad \ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

Beberapa aplikasi mungkin hanya memerlukan satu atau dua parameter saja untuk mendapatkan sistem kendali yang tepat, baik itu PI, PD, P dan I saja. Hal ini dicapai dengan menetapkan output gain kontrol yang tidak diinginkan ke nol. Kontrol PI sangat umum, karena tindakan *derivatif* sangat peka terhadap gangguan *(noise)*.

2.4.1 Aksi Kontrol PI

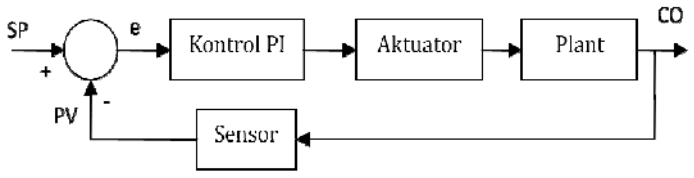
Kontrol PI *(Proporsional dan Integral)* adalah bagian dari kontrol PID dengan tanpa menggunakan aksi *derivative*. Pengendali PI merupakan gabungan kontrol *proporsional* dan *integral* dalam waktu kontinyu. Sinyal keluaran pengendali PI dirumuskan sebagai berikut : [11]

$$( ) = \quad ( ) + - \int \quad ( ) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.2)$$

Dalam persamaan diskrit, persamaan 2.2 tersebut dapat di rumuskan sebagai berikut:

$$( ) = \quad ( ) + \quad . \quad \Sigma \quad (\quad ( ) \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.3)$$

Diagram blok untuk perumusan di atas diperlihatkan pada gambar 2.29:



Gambar 2.29 *Diagram Blok Sistem kontrol PI*

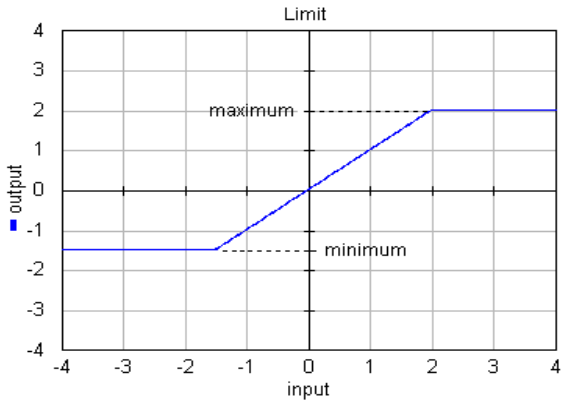
Nilai sumasi error persamaan 2.3 dinyatakan dengan persamaan 2.4 di bawah ini:

$$( ) = \quad (\quad - 1) + \quad (\quad) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.4)$$

2.4.1.1 Windup dan Anti-Windup

Nilai sumasi atau penjumlahan yang selalu positif dan membesar akan membuat nilai CO menjadi sangat besar melampaui batas saturasi dari aktuator, gejala ini disebut dengan *integral windup*. Untuk mengatasinya dapat dilakukan dengan aksi *anti-windup*.

Anti-windup pada dasarnya adalah pembatasan sebuah model saturasi penggerak yang dipasang pada sisi umpan maju kontroler. Batas saturasi model penggerak biasanya di *setting* pada batas saturasi dari penggerak aslinya. Namun untuk kepentingan keamanan operasi pengontrolan, batas tersebut dapat juga di*setting* lebih kecil dari nilai satursi penggerak aslinya.



Gambar 2.30 *Kurva Model Saturasi*

Hubungan logika *input – output* model saturasi ini secara logika dapat di tuliskan:

If (Input ≤ Min) Output = Min

Else If (Input ≥ Max) Output = Max

Else Output = Input

Pengertian algoritma di atas menjelaskan jika *input* lebih kecil atau sama dengan batas minimumnya maka *output* sama dengan batas minimumnya, atau jika *input* lebih besat atau sama dengan batas maksimumnya maka hasil *output* sama dengan batas maksimumnya. Dan jika tidak memenuhi kondisi sebelumnya *output* sama dengan *input*. [10]