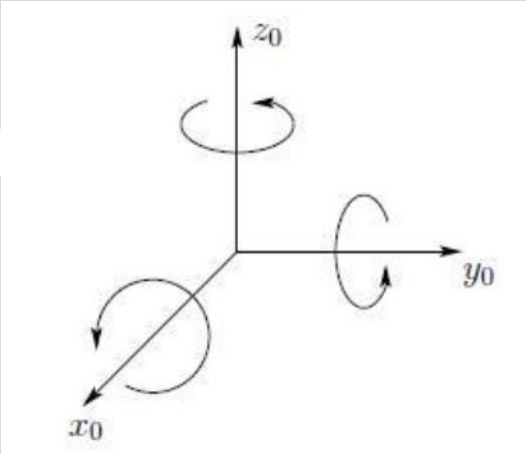
BAB II
Tinjauan Pustaka

2.1 Kinematika Robot Lengan

Kinematika robot lengan terdiri dari pergerakan rotasi dan translasi.

2.1.1 Rotasi

Pada gerakan rotasi yaitu gerakan berputar pada sebuah sumbu yang tetap, gerakkan tersebut dapat berputar pada sumbu x, y maupun z. seperti yang terlihat dalam Gambar 2.1.



Gambar 2.1 Rotasi pada Tiap-tiap Sumbu Koordinat Kartesian
Sumber: Spong, Mark W, 2007

Matrik transformasi rotasi pada tiap-tiap sumbu terlihat pada persamaan dibawah ini (persamaan 2.1, 2.2 dan 2.3), θ untuk sumbu z, α untuk sumbu x dan ϕ untuk sumbu y.

$$R_{z,\theta} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{……..(2.1)}$$

$$R_{x,\alpha} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \qquad \text{……..(2.2)}$$

$$R_{y,\phi} \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \quad \text{……..(2.3)}$$

Jadi untuk merepresentasikan sebuah posisi (p_1) dari sumbu koordinat yang telah berotasi ke dalam posisi (p_0) sumbu koordinat semula dirumuskan pada persamaan 2.6.

$$p_0 \begin{bmatrix} p_{0x} \\ p_{0y} \\ p_{0z} \end{bmatrix} \quad \text{…….(2.4)}$$

$$p_1 \begin{bmatrix} p_{1x} \\ p_{1y} \\ p_{1z} \end{bmatrix} \quad \text{..….(2.5)}$$

$$p_0 \quad R_0^1 p_1 \quad \text{..….(2.6)}$$

2.1.2 Translasi

Sedangkan pada translasi, artinya terdapat pergeseran sumbu koordinat pada jarak tertentu dari sumbu koordinat semula. Jadi untuk merepresentasikan posisi (p_1) dari sumbu koordinat translasi kedalam sumbu koordinat semula dirumuskan pada persamaan 2.8.

$$p_0 \quad p_1 + d_0^1 \quad \text{…..(2.7)}$$

$$\begin{bmatrix} p_{0x} \\ p_{0y} \\ p_{0z} \end{bmatrix} \begin{bmatrix} p_{1x} \\ p_{1y} \\ p_{1z} \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \quad \text{.....(2.8)}$$

Apabila terjadi dua gerakan yaitu rotasi dan translasi secara bersamaan, artinya sumbu koordinat semula diputar (rotasi) dengan sudut tertentu (R_0^1) kemudian digeser (translasi) sejauh jarak tertentu (d_0^1). Dapat kita selesaikan dengan menggabung persamaan rotasi dan translasi (persamaan 2.6 dan 2.7) seperti pada persamaan 2.9.

$$p_0 \quad R_0^1 p_1 + d_0^1 \qquad \text{....(2.9)}$$

Persamaan 2.9 dapat kita reprentasikan kedalam bentuk matrik agar lebih mudah perhitungan dan sederhana penulisannya.

2.1.3 *DENAVIT-HARTENBERG (D-H) Representation*

Representasi bentuk matrik H yang terdiri dari matrik rotasi dan translasi, kita sebut sebagai *homogenous transformation. Homogenous transformation* dapat memudahkan kita untuk mendapat posisi suatu titik yang diperlihatkan dari koordinat *frame* i kedalam bentuk koordinat *frame* j yang telah dilakukan gerakkan translasi dan rotasi.

$$\begin{array}{l} p_0 \quad R_0^1 p_1 + d_0^1 \\ \begin{bmatrix} p_0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ 1 \end{bmatrix} \\ H \quad \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \end{array} \qquad \text{.....(2.10)}$$

Pada umumnya untuk menentukan *frame*/koordinat memiliki aturan-aturan tertentu agar nantinya memudahkan kita dalam menganalisa posisi dari pergerakan robot tersebut, Diantaranya adalah *DENAVIT-HARTENBERG,* atau *D-H convenction,* Pada *D-H convenction* suatu matrik *homogenous transformation* A_i adalah merepresentasikan hasil dari empat transformasi dasar.

$$A_i \quad Rot_{z_i\theta_i} Trans_{z_i d_i} Trans_{x_i a_i} Rot_{x_i\alpha_i} \qquad \text{.....(2.11)}$$

$$A_i \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{.....(2.12)}$$

$$A_i \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \cos\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & \cos\theta_i \cos\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{……(2.13)}$$

$$T_0^n \begin{bmatrix} n_x & s_x & a_x & P_x \\ n_y & s_y & a_y & P_y \\ n_z & s_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{…….(2.14)}$$

Pada *D-H convenction,* terdapat beberapa parameter yaitu a_i, α_i, d_i, dan θ_i. a_i kita sebut sebagai *length,* α_i,disebut sebagai *twist,* d_i disebut sebagai *offset,* dan θ_i disebut sebagai *angle.*

2.1.4 *Forward kinematic*

Apabila kita menghendaki untuk menghitung posisi (dari koordinat dasar) robot lengan dengan memberikan nilai pada D-H parameter, maka hal itu disebut *forward kinematic*. *Forward kinematic* adalah suatu metode perhitungan posisi sebagi fungsi sudut dengan menghitung setiap *link*. Jika kita memiliki n *link* maka kita bisa dirumuskan pada persamaan 2.15.

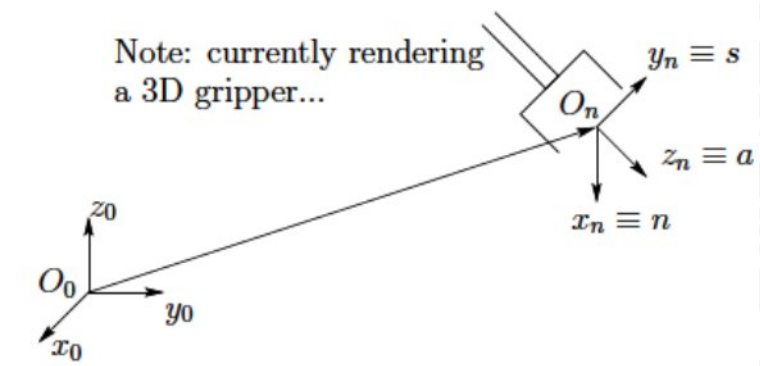
$$T_0^n \quad A_1 \ldots\ldots A_n \quad \text{…….(2.15)}$$

Matrik n $\begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$ adalah vektor yang merepresentasikan arah sumbu o_1x_1 kedalam $o_0x_0y_0z_0$, n adalah arah normal (keatas).

Matrik s $\begin{bmatrix} s_x & s_y & s_z \end{bmatrix}^T$ adalah vektor yang merepresentasikan arah sumbu o_1y_1 kedalam $o_0x_0y_0z_0$, s adalah arah *sliding* (pergeseran).

Matrik a $\begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$ adalah vektor yang merepresentasikan arah sumbu o_1z_1 kedalam $o_0x_0y_0z_0$, a adalah *appoarch* (pendekatan).

Matrik p $\begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$ adalah vektor yang merepresentasikan dari sumbu asal o_0 ke sumbu asal o_1 direpresentasikan kedalam sumbu koordinat $o_0x_0y_0z_0$



Gambar 2.2 Parameter Pada *Forward Kinematic*

Sumber: Spong, Mark W, 2007

2.1.5 *Invers Kinematic*

Kebalikan dari metode *forward kinematic* adalah *invers kinematik*. Pada *forward kinematic* kita memberikan nilai pada setiap sudut motor servo dan mendapatkan posisi *end-effector*, sedangkan pada *invers kinematic* kita memberi masukan berupa posisi dan robot lengan akan mencari sudut pada tiap motor servo agar tepat sesuai dengan posisi yang kita inginkan. Solusi umum dari *invers kinematic* memang tidak ada, hal ini disebabkan solusinya sangat bergantung pada peletakan *frame* (sumbu koordinat). Adapun pendekatan yang dilakukan untuk menentukan sudut motor servo yaitu dengan pendekatan *Geometric Solution* dan *Algebraic Solution.*

Langkah awal untuk menentukan sudut pada tiap-tiap link adalah dengan melakukan *invers* pada matrik *forward kinematic*. Persamaan-persamaan yang dihasilkan dari *invers* matrik *forward kinematic* kemudian dicari solusinya dengan melakukan subtitusi dan eliminasi berdasarkan dua pendekatan di atas.

$$A_1^{-1}T_0^n \quad A_1^{-1}A_1 \ldots A_n \tag{2.16}$$

hingga

$$A_{n-1}^{-1} \ldots A_1^{-1}T_0^n \quad A_{n-1}^{-1} \ldots A_1^{-1}A_1 \ldots\ldots A_n \tag{2.17}$$

2.2 Motor Servo

Motor servo adalah sebuah motor dengan sistem *closed feedback* di mana posisi dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri atas sebuah motor, serangkaian *internal gear*, potensiometer dan rangkaian kontrol. Potensiometer berfungsi untuk menentukan batas sudut putaran servo. Sedangkan sudut sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor.

2.2.1 Bentuk Fisik Motor Servo

Bentuk fisik motor servo hampir menyerupai motor DC yang dilengkapi dengan *gear box* dan terdapat motor DC biasa. Perbedaan antara motor DC *gear box* dengan motor servo adalah pada rangkaian umpan balik dan adanya sensor pendeteksi gerakan (*potensiometer)* pada motor servo sehingga motor servo mampu mempertahankan posisi sudut gerakan yang kita inginkan. Bentuk fisik motor servo selengkapnya ditunjukkan dalam *Gambar 2.3.*

Gambar 2.3 Motor Servo
(a) Bentuk Fisik Motor Servo (b) Dimensi Motor Servo
sumber : http://fahmizaleeits.wordpress.com

2.2.2 Jenis – Jenis Motor Servo

Jenis motor servo ada 2 yaitu jenis motor servo *continous* dan motor servo standat. Kedua motor servo ini tidak jauh berbeda hanya saja pada putarannya. Berikut ini adalah penjelasan kedua jenis motor servo tersebut.

1). Motor Servo *Standar 180°*

Motor servo jenis ini merupakan motor yang hanya mampu bergerak dua arah (*CW* dan *CCW)* dan mempunyai *defleksi* masing-masing sudut mencapai 90° sehingga total *defleksi* sudut dari kanan – tengah – kiri adalah 180°.

2). Motor Servo *Continuous*

Motor servo jenis ini mampu bergerak dua arah *(CW* dan *CCW)* dan tanpa batasan *defleksi* sudut putar (dapat berputar secara kontinyu) sehingga motor ini berputar 360°.

2.2.3 Konfigurasi Pin Motor Servo

Motor servo hanya memiliki 3 kabel yang mana masing-masing fungsinya terdiri dari *Positive, Ground* dan *Control.* Motor servo mampu bergerak searah jarum jam ataupun berlawanan arah jarum jam tanpa membalik PIN konektor pada motor servo, hal ini disebabkan bahwa pada motor servo telah terdapat *driver* untuk membalik polaritas motor DC yang ada pada motor servo. Konfigurasi pin pada motor servo adalah seperti yang di tunjukkan dalam Gambar 2.4 berikut ini.

Gambar 2.4 Konfigurasi Pin Motor Servo
sumber : http://fahmizaleeits.wordpress.com

Keterangan :

1. Positive (*Red Cabel*)

Pada kabel konektor ini diberikan tegangan *positive* sesuai kebutuhan dan spesifikasi pada motor servo yang digunakan.

2. Ground (*Brown Cabel*)

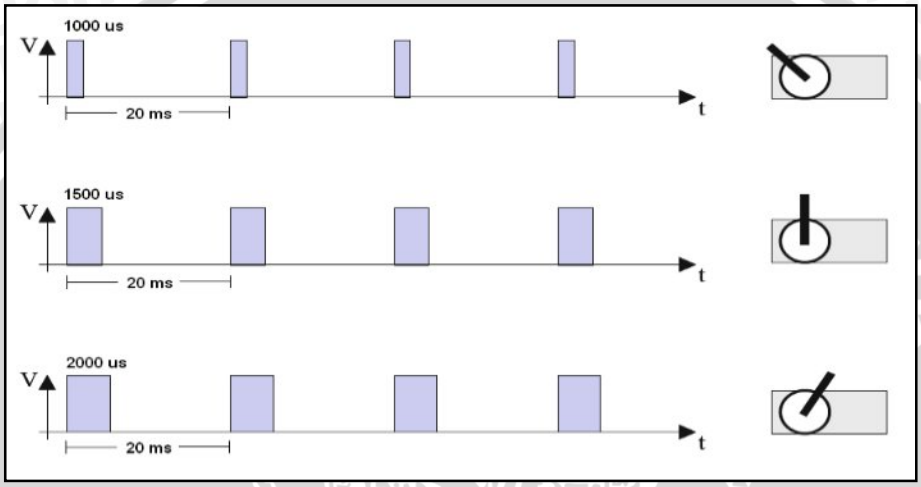
Pada kabel konektor ini dihubungkan ke *Ground* (-) baterai

3. Control (*Orange or Yellow Cabel*)

Pada pin ini dikirimkan sinyal pulsa sesuai spesifikasi yang dibutuhkan pada motor servo untuk menggerakkan motor servo mengikuti arah jarum jam atau berlawanan arah jarum jam.

2.2.4 Mode Pensinyalan Motor Servo

Pergerakan motor servo diatur oleh pulsa yang dikirimkan dari Mikrokontroler menuju motor servo. Secara umum motor servo standar akan bergerak menuju posisi tengah atau 90° jika diberikan pulsa 1,5 ms dan akan bergerak menuju 0° jika diberikan pulsa sebesar 1 ms, begitu pula jika diberikan pulsa sebesar 2 ms maka motor servo akan bergerak menuju sudut 180°. Dalam Gambar 2.5 ditunjukkan pergerakan motor servo seperti yang sudah dijelaskan.



Gambar 2.5 Pengaturan Sudut Motor Servo

Sumber **:**Thomas Braunl, 2008 : 81

Dengan demikian, untuk menggerakkan motor servo searah jarum jam maka dapat memberikan pulsa menuju 2 ms (lebih besar). Sedangkan jika ingin menggerakkan berlawanan arah jarum jam dilakukan pengurangan pulsa yang diberikan (lebih kecil) menuju 1ms. Jadi agar didapat posisi motor servo yang diinginkan, maka lebar pulsa periodik yang harus diberikan dapat dihitung dengan rumus berikut.

$D = D\left(\frac{1000}{180}\right) + 1000\ \mu s$ atau

$T = (0{,}18)D + 1000\ \mu s$

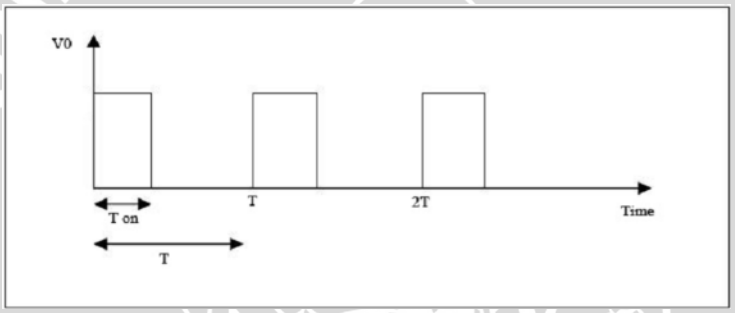
Keterangan :

D = posisi sudut motor servo (derajat)

T = lebar pulsa (μs)

Motor servo akan bekerja secara baik jika pada bagian pin kontrolnya diberikan sinyal *PWM* dengan frekuensi 50 Hz. Dimana pada saat sinyal dengan frekuensi 50 Hz tersebut dicapai pada kondisi *ton duty cycle* 1.5 ms, maka rotor dari motor akan berhenti tepat di tengah (sudut 0°/ netral). Pada saat *ton duty cycle* dari sinyal yang diberikan kurang dari 1.5 ms, maka rotor akan berputar ke arah kiri dengan membentuk sudut yang besarnya *linier* terhadap besarnya *ton duty cycle* dan akan bertahan diposisi tersebut. Begitu juga sebaliknya, jika *ton duty cycle* dari sinyal yang diberikan lebih dari 1.5 ms, maka rotor akan berputar ke arah kanan dengan membentuk sudut yang *linier* pula terhadap besarnya *ton duty cycle*, dan bertahan diposisi tersebut.

Pada sinyal PWM, frekuensi sinyal konstan sedangkan *duty cycle* bervariasi dari 0%-100%. Dengan mengatur *duty cycle* akan diperoleh keluaran yang diinginkan. Sinyal PWM (Pulse Width Modulation) secara umum dapat dilihat dalam Gambar 2.6.



Gambar 2.6 Sinyal PWM Secara Umum

Sumber: www.electronics-scheme.com

$$Dutycycle = \frac{Ton}{T} \; x \; 100\% \ldots (\%) \quad \text{.....(2-17)}$$

Dengan :

T on = Periode logika tinggi

T = Periode keseluruhan

$$Vdc = Dutycycle \; x \; Vcc \ldots (V) \quad \text{.....(2-18)}$$

Sedangkan frekuensi sinyal dapat ditentukan dengan rumus berikut :

$$f0n = \frac{fclk\ I/0}{N.256} \ldots (Hz)$$

2.3 *Visual Basic* 6.0

Microsoft Visual Basic (sering disingkat sebagai VB saja) merupakan sebuah bahasa pemrograman yang menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program perangkat lunak berbasis sistem operasi Microsoft Windows dengan menggunakan model pemrograman (COM), Visual Basic merupakan turunan bahasa pemrograman BASIC dan *menawarkan* pengembangan perangkat lunak berbasis grafik dengan cepat, beberapa bahasa *script* seperti *Visual Basic for Applications (VBA)* dan *Visual Basic Scripting Edition (VBScript),* mirip seperti halnya *Visual Basic*, tetapi cara kerjanya yang berbeda. Para programmer dapat membangun aplikasi dengan menggunakan komponen yang disediakan oleh *Microsoft Visual Basic* program-program yang ditulis dengan *Visual Basic* juga dapat menggunakan *Windows API*, tapi membutuhkan deklarasi fungsi luar tambahan. Dalam pemrograman untuk bisnis, *Visual Basic* memiliki pangsa pasar yang sangat luas. Dalam sebuah survey yang dilakukan pada tahun 2005, 62% pengembang perangkat lunak dilaporkan menggunakan berbagai bentuk *Visual Basic*, diikuti oleh *C++,JavaScript, C#,* dan *Java.*

Pada pembuatan proyek akhir ini digunakan *Microsoft Visual Basic 6.0. Visual Basic 6.0* merupakan sebuah pengembangan dari bahasa *BASIC. BASIC* dirancang tahun 1950an dan ditujukan untuk dapat digunakan oleh para *programmer* pemula. Sedangkan *Visual Basic* itu sendiri memiliki pengertian kata *Visual* dalam nama pemrograman ini mewakili pada metode untuk membuat *Graphical User Interface (GUI).* Dengan hanya mengatur letak dari elemen-elemen sebuah *interface* tanpa menuliskan baris kode yang banyak. Kata *BASIC* sendiri merupakan kependekan dari *Beginners All-Purpose Symbolic Instruction Code*. Di dalam bahasa *Visual Basic* telah dilengkapi dengan beberapa ratus pernyataan, fungsi dan kata kunci, banyak di antaranya berkaitan langsung dengan *GUI* dari *windows*.

2.3.1 Operator dalam Visual Basic

Operator yang biasa digunakan dalam pemrograman *Visual Basic* diantaranya adalah :

a. Operator Matematika

Penggunaan operator matematika lebih ditujukan untuk pembuatan rumus atau *formula*. Rumus atau *formula* adalah pernyataan yang menggabungkan angka, *Variable,* Operator, dan kata kunci untuk membuat suatu nilai baru.

Table 2.1 Operator Matematika dalam *VB*

Simbol	Operasi Matematis	Contoh
^	Pemangkatan	5^2 hasilnya 25
*	Perkalian	2*2 hasilnya 4
/	Pembagian (hasil pecahan)	5 / 2 hasilnya 2,5
\	pembagian (hasil bulat)	5 \ 2 hasilnya 2
Mod	Sisa pembagian	5 Mod 2 hasilnya 1
+	Penjumlahan	5 + 2 hasilnya 7
-	Pengurangan	5 – 2 hasilnya 3
&	Penggabungan *string*	penggabungan *string*

Sumber: Achmad Basuki, 2006

b. Operator Perbandingan

Operator perbandingan digunakan untuk membandingkan dua *variable* atau objek. Untuk lebih jelas lihat dalam Tabel 2.2.

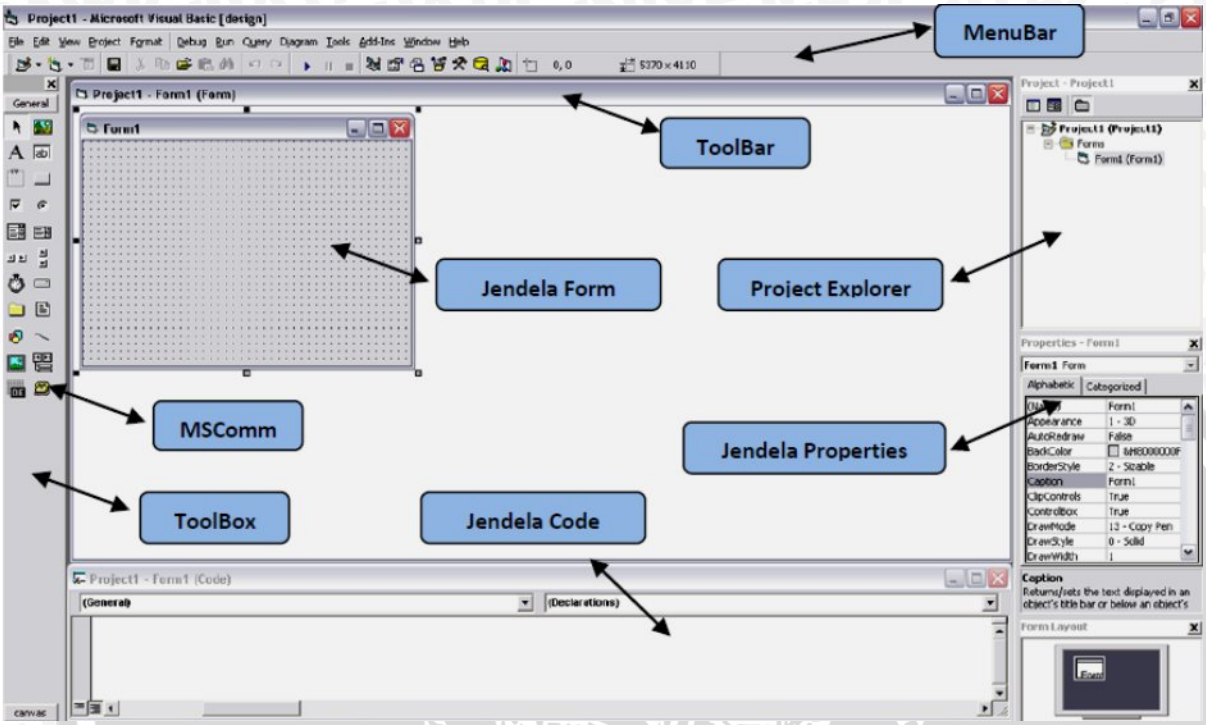
Tabel 2.2 Operator Perbandingan dalam VB

Simbol	Operasi Matematis	Contoh
<	lebih kecil	5< 2 hasilnya FALSE
>	lebih besar	5 > 2 hasilnya TRUE
<=	lebih kecil atau sama dengan	5<=2 hasilnya FALSE
>=	lebih besar atau sama dengan	5 >= 2 hasilnya TRUE
=	sama dengan	5 = 2 hasilnya FALSE
<>	tidak sama dengan	5 <> 2 hasilnya TRUE

Sumber: Achmad Basuki, 2006

2.3.2 Tampilan *Visual Basic* 6.0

Untuk dapat men-*design* dengan *visual basic 6.0,* kita harus memahami beberapa fungsi *icon* dasar dalam *visual basic.* Berikut tampilan dari *visual basic* 6.0 (lihat Gambar 2.7).



Gambar 2.7 Tampilan Visual Basic 6.0

Sumber: Achmad Basuki, 2006

Keterangan :

1. ***Menubar***

   MenuBar pada Visual Basic merupakan *Menu Bar standart* yang ditampilkan pada awal *Project*.

2. ***Toolbar***

   Pada *ToolBar* dapat dimasukkan komponen tambahan atau memakai komponen yang sudah ada.

3. ***Toolbox***

   Bila *Toolbox* tidak muncul klik tombol *Toolbox* ( ) pada bagian *Toolbar* atau klik menu *View* > *Toolbox*.

4. **Jendela *Form***

   Bila Jendela *Form* tidak muncul klik tombol *View Object* ( ) pada bagian *Project Explorer* atau klik menu *View* > *Object.*

5. Jendela *Code*

Bila Jendela *Code* tidak muncul klik tombol *View Code* ( ) di pada bagian *Project Explorer* atau klik *menu View > Code*.

6. *Project Explorer*

Bila *Project Explorer* tidak muncul klik tombol *Project Explorer* ( ) pada bagian *Toolbar* atau klik menu *View > Project Explorer*.

7. Jendela *Properties*

Bila Jendela *Properties* tidak muncul klik tombol *Properties Window* ( ) pada bagian *Toolbar* atau klik menu *View > Properties Window*.

8. MSComm

MSComm adalah suatu peralatan atau komponen dari Visual Basic 6.0, fungsi dari MSComm adalah untuk melakukan komunikasi Serial antarmuka atau *Interface* dari suatu modul Elektrik yang dilengakapi dengan konektor DB-9.

Pembuatan program aplikasi menggunakan Visual Basic dilakukan dengan membuat tampilan aplikasi pada form, kemudian diberi *script* program di dalam komponen-komponen yang diperlukan. Form disusun oleh komponen-komponen yang berada di *[Toolbox]*, dan setiap komponen yang dipakai harus diatur propertinya lewat jendela *[Property]*.

2.4 Mikrokontroler Atmega168

Mikrokontroler Atmega168 merupakan mikrokontroler keluaran AVR yang merupakan mikrokontroler AVR CMOS 8 bit berdaya rendah. Mikrokontroler Atmega168 dapat mengeksekusi instruksi hingga 1 MIPS per MHz dalam satu siklus waktu. Karakteristik utama yang dimiliki oleh mikrokontroler Atmega168 adalah:

1). Memori program dan data yang *nonvolatile*.
2). Sistem *self-programable flash* 16 kByte.
3). EEPROM sebesar 512 Byte, dan 1kByte SRAM internal.
4). 23 saluran I/O dan 32 *general purpose register*.
5). Dua timer/counter 8 bit dengan *prescaller* terpisah, dan mode pembanding (*compare mode*).
6). Satu buah timer/counter 16 bit dengan *prescaller* terpisah, mode pembanding dan perekam (*capture*).
7). Internal dan eksternal *interrupt*.

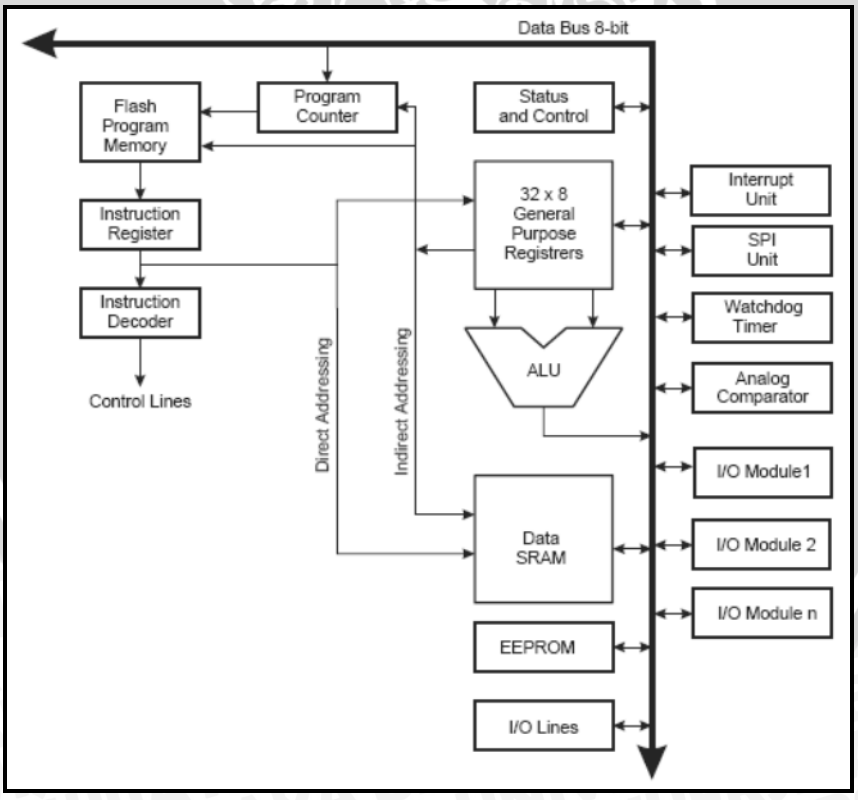
8). Empat buah pin PWM.

9). Serial USART.

2.4.1 Arsitektur AVR

Mikrokontroler AVR menggunakan arsitektur harvard yang memisahkan memori dan bus untuk program dan data sehingga memaksimalkan performa. Instruksi pada memori program dieksekusi secara *pipeline*. Ketika satu instruksi masih dieksekusi, instruksi selanjutnya sudah disiapkan untuk eksekusi tanpa menunggu eksekusi pertama selesai. Konsep ini memungkinkan instruksi untuk dieksekusi setiap satu siklus waktu.

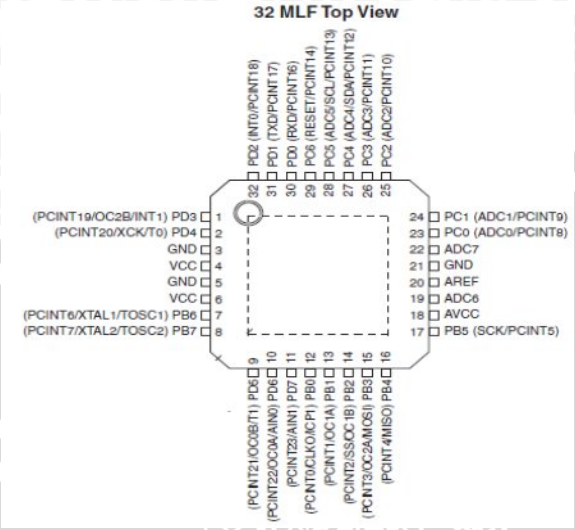
Register file terdiri atas 32 x 8 bit *general purpose working register* dengan waktu akses satu siklus waktu, yang memungkinkan operasi ALU (*Arithmetic Logic Unit*) dijalankan dalam satu siklus waktu. Dua *operand* diambil dari register, operasi ALU dijalankan, dan hasilnya disimpan kembali dalam *register file*. Operasi aritmetika dan logika dapat dijalankan oleh ALU, baik antar register atau antara register dengan konstanta. Hasil dari operasi aritmetika disimpan dalam register status (*Status Register*), menggantikan isi yang sebelumnya. Arsitektur AVR ditunjukkan dalam Gambar 2.8.



Gambar 2.8 Arsitektur AVR

Sumber: Atmel, 2004 : 7

Program *flash memory* dibagi dalam dua bagian, *Boot Program* dan *Application Program*. Kedua bagian memiliki *lock bits* untuk mengunci operasi tulis (*write*) dan baca/tulis (*read/write*). Konfigurasi pin mikrokontroler atmega168 dapat dilihat dalam Gambar 2.9.



Gambar 2.9 Konfigurasi Pin Atmega 168

Sumber: Atmel, 2004 : 2

2.4.2 Struktur dan Operasi Port

Mikrokontroler atmega168 ini mempunyai 4 buah port, yang memiliki 8 buah jalur I/O. Beberapa karakteristik port mikrokontroler atmega168 dijelaskan secara singkat:

1). Unit I/O dapat dialamati perjalur atau per port

2). Setiap jalur I/O memiliki buffer, penahan (latch), kemudi input dan kemudi output.

3). Setiap jalur I/O terdapat register pengatur apakah dijadikan input atau dijadikan output.

4). Port B adalah I/O *bi-directional* 8 bit dengan resistor *pull-up* internal. Sebagai masukan, pin port B yang diberi *pull-low* secara eksternal akan mengalirkan arus bila resistor *pull-up* diaktifkan. Port B juga memiliki fungsi khusus, seperti yang terlihat dalam Tabel 2.3.

Tabel 2.3 Fungsi Khusus Port B Atmega168

Port Pin	Alternate Functions
PB7	XTAL2 (Chip Clock Oscillator pin 2) TOSC2 (Timer Oscillator pin 2)
PB6	XTAL1 (Chip Clock Oscillator pin 1 or External clock input) TOSC1 (Timer Oscillator pin 1)
PB5	SCK (SPI Bus Master clock Input)
PB4	MISO (SPI Bus Master Input/Slave Output)
PB3	MOSI (SPI Bus Master Output/Slave Input) OC2 (Timer/Counter2 Output Compare Match Output)
PB2	$\overline{SS}$ (SPI Bus Master Slave select) OC1B (Timer/Counter1 Output Compare Match B Output)
PB1	OC1A (Timer/Counter1 Output Compare Match A Output)
PB0	ICP1 (Timer/Counter1 Input Capture Pin)

Sumber:Atmel, 2004 : 56

5). Port C adalah I/O *bi-directional* 8 bit dengan resistor *pull-up* internal. Sebagai masukan, pin port C yang diberi *pull-low* secara eksternal akan mengalirkan arus bila resistor *pull-up* diaktifkan. Port C juga memiliki fungsi khusus, seperti yang terlihat dalam Tabel 2.4.

Tabel 2.4 Fungsi Khusus Port C Atmega168

Port Pin	Alternate Function
PC6	$\overline{RESET}$ (Reset pin)
PC5	ADC5 (ADC Input Channel 5) SCL (Two-wire Serial Bus Clock Line)
PC4	ADC4 (ADC Input Channel 4) SDA (Two-wire Serial Bus Data Input/Output Line)
PC3	ADC3 (ADC Input Channel 3)
PC2	ADC2 (ADC Input Channel 2)
PC1	ADC1 (ADC Input Channel 1)
PC0	ADC0 (ADC Input Channel 0)

Sumber:Atmel, 2004 : 59

6). Port D adalah I/O *bi-directional* 8 bit dengan resistor *pull-up* internal. Sebagai masukan, pin port D yang diberi *pull-low* secara eksternal akan mengalirkan arus bila resistor *pull-up* diaktifkan. Port D juga memiliki fungsi khusus, seperti yang terlihat dalam Tabel 2.5.

Tabel 2.5 Fungsi Khusus Port D Atmega168

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input)
PD6	AIN0 (Analog Comparator Positive Input)
PD5	T1 (Timer/Counter 1 External Counter Input)
PD4	XCK (USART External Clock Input/Output) T0 (Timer/Counter 0 External Counter Input)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

Sumber**:**Atmel, 2004 : 61

2.4.3 Timer/Counter

Mikrokontroler atmega168 memiliki 3 buah *timer/counter* yang terdiri atas 2 buah *timer/counter* 8 bit dan 1 buah *timer/counter* 16 bit. Ketiga *timer/counter* ini dapat diatur dalam mode yang berbeda. Selain itu semua *timer/counter* dapat difungsikan sebagai sumber interupsi.*Timer/counter* dapat digunakan dalam 4 mode operasi, yaitu :

1). *Mode* pertama (*mode* 0) adalah *mode* normal, *timer* digunakan sebagai pencacah tunggal yang dapat mencacah dari 0x00 sampai dengan 0xFF. Setelah mencapai nilai 0xFF maka register *counter* akan *reset* atau kembali ke 0x00.

2). *Mode* kedua (mode 1) adalah *Phase Correct PWM* (PCP). *Mode* ini digunakan untuk menghasilkan sinyal PWM dimana nilai *register counter* yang mencacah naik dan turun secara terus menerus akan selalu dibandingkan dengan *register* pembanding OCRn. Hasil perbandingan *register counter* dan OCRn digunakan untuk membangkitkan sinyal PWM yang dikeluarkan pada pin OCn.

3). *Mode* ketiga (mode 2) adalah *clear timer on compare match* (CTC). *Register counter* akan mencacah naik kemudian akan di-*reset* atau kembali menjadi 0x00 pada saat nilai TCNT sama dengan OCRn.

4). *Mode* keempat (mode 3) adalah *fast* PWM. Mode ini hampir sama dengan *mode phase correct* PWM, hanya perbedaannya adalah *register counter* mencacah naik saja dan tidak mencacah turun.