BAB II
TINJAUAN PUSTAKA

2.1 Sejarah Robot

Robot adalah alat menaknika yang dapat melakukan tugas fisik, baik lewat pantauan manusia, maupun bekerja secara komputerisasi yang menghasilkan kecerdasan dan perilaku yang individu. Abu Al -'Iz Ibn Ismail ibn al-Razaz al-Jazari (1136-1206) adalah orang Arab penting dalam ilmu robotik Islam. Al-Jazari dilahirkan di Al-Jazira, Mesopotamia semasa Zaman Kegemilangan Islam. Dia tinggal di Diyar Bakir, Turki. Dia adalah penulis Kitáb fí ma'rifat al-hiyal al-handasiyya *(Buku Pengetahuan Ilmu Mekanik*) pada tahun 1206, di mana dia menerangkan 50 alat mekanik di samping dengan arahan bagaimana membinanya.

Seperti juga ayahnya dia bekerja untuk raja-raja Urtuq atau Artuqid di Diyar Bakir dari tahun 1174 sampai tahun 1200 sebagai ahli teknikal. Pada waktu itu, Al Jazari telah mampu mencipta robot manusia (humanoid) yang boleh diprogram. Al Jazari mengembangkan prinsip hidrolik untuk menggerakkan mesin yang kemudian hari dikenal sebagai mesin robot.

Saat ini hampir tidak ada orang yang tidak mengenal robot, namun pengertian robot tidaklah dipahami secara sama oleh setiap orang. Sebagian membayangkan robot adalah suatu mesin tiruan manusia (*Humanoid*), meski demikian *Humanoid* bukanlah satu-satunya jenis robot, jenis robot yang lain yaitu robot *mobile*,robot *manipulator* (tangan),robot berkaki,robot *flying*, robot *under water*.

Pada kamus *Webster* pengertian robot adalah :

An automatic device that performs function ordinarily ascribed to human beings

(sebuah alat otomatis yang melakukan fungsi berdasarkan kebutuhan manusia)

Dari kamus *Oxford* diperoleh pengertian robot adalah:

A machine capable of carrying out a complex series of actions automatically, especially one programmed by a computer.

(Sebuah mesin yang mampu melakukan serangkaian tugas rumit secara otomatis, terutama yang diprogram oleh komputer).

Pengertian dari *Webster* mengacu pada pemahaman banyak orang bahwa robot melakukan tugas manusia, sedangkan pengertian dari *Oxford* lebih umum, beberapa organisasi di bidang robot membuat definisi tersendiri.

International Organization for Standardization (ISO 8373) mendefinisikan robot sebagai:

An automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications

(Sebuah *manipulator* yang terkendali, multifungsi, dan mampu diprogram untuk bergerak dalam tiga aksis atau lebih, yang tetap berada di tempat atau bergerak untuk digunakan dalam aplikasi otomasi industri)

Dari beberapa definisi di atas, kata kunci yang ada yang dapat menerangkan pengertian robot adalah:

1. Dapat memperoleh informasi dari lingkungan (melalui sensor).
2. Dapat diprogram.
3. Dapat melaksanakan beberapa tugas yang berbeda.
4. Bekerja secara otomatis
5. Cerdas (*intelligent*)
6. Digunakan di industri

2.2 Definisi Sensor

Sensor adalah alat untuk mendeteksi/mengukur sesuatu, yang digunakan untuk mengubah variasi mekanis, magnetis, panas, sinar dan kimia menjadi tegangan dan arus listrik. Dalam lingkungan sistem pengendali dan robotika, sensor memberikan kesamaanyang menyerupai mata, pendengaran, hidung, lidah yang kemudian akan diolah oleh kontroler sebagai otaknya (Petruzella, 2001). Sensor sangat berperan penting dalam dunia robotika yang berfungsi sebagai

input navigasi suatu robot, adapun jenis jenis sensor yaitu: sensor suara,sensor cahaya, sensor tekanan sensor api, sensor suhu, sensor kelambapan, sensor ultrasonic,sensor magnet. Pada dasarnya sensor dangan penting di dunia robotik tergantung kebutuhan dari robot itu sendiri.

2.2.1 Sensor Warna

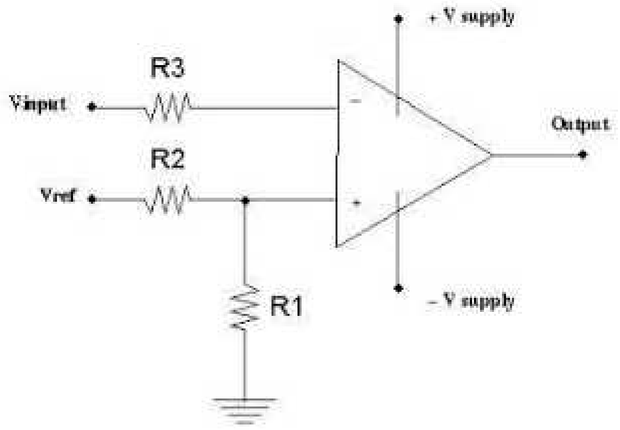
Sensor warna adalah sensor yang digunakan untuk mendeteksi warna tertentu. Sensor ini menggunakan photodiode sebagai penangkap cahaya yang dipancarkan dari *Light Emiting Diode*(LED), photodiode yang berfungsi sebagai pengubah cahaya yang ditangkap kemudian mengubahnya menjadi energi listrik/ *Direct Current*. sensor ini banyak digunakan untuk keperluan industri yang digunakan untuk mendeteksi warna pada suatu kertas, cat mobil ataupun warna dari suatu makanan, apakah warna yang digunakan itu sudah baik atau tidak. Pada robot penari, sensor tersebut digunakan untuk mendeteksi beda warna dari setiap zona, dimana dalam perlombaan itu terdiri dari 3 zona yang masing masing berwarna berbeda.

2.3 Definisi *Comparator Oprational Amplifier*

Pada umumnya *Operational Amplifier* atau penguat operasional digunakan sebagai *Comparator* (Komparator) atau pembanding. Komparator adalah komponen elektronik yang berfungsi membandingkan dua nilai masukan kemudian memberikan hasil keluaran dari satu keadaan dimana keluaran yang lebih besar atau lebih kecil. Komparator bisa dibuat dari konfigurasi *Open-Loop* dan *Close-Loop*.

2.3.1 Komparator *Open Loop*

Komparator *Open Loop* adalah Jika kedua *input* pada Op Amp maka Op Amp akan membandingkan kedua saluran *input* tersebut. Hasil komparasi dua tegangan pada saluran masukan akan menghasilkan tegangan saturasi positif (+Vsat) atau saturasi negatif (-Vsat).



Gambar 2.1 : Konfigurasi Komparator *Open Loop*

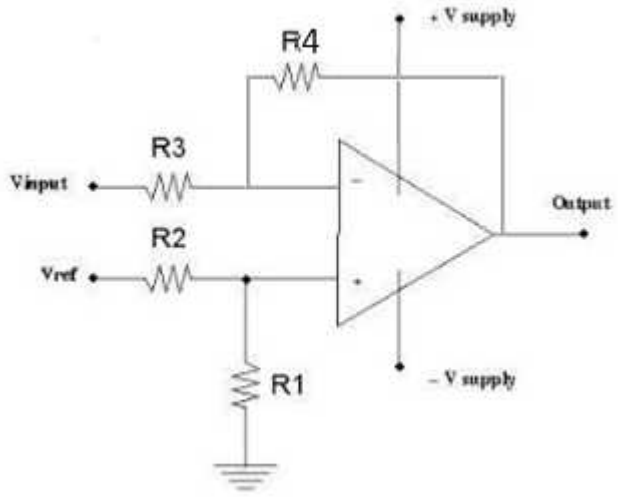
(Sumber : http://ilham-kn.blogspot.com/2013/12/komparator.html. diakses pada tanggal 01 Juni 2014)

Penjelasan dari gambar diatas adalah apabila tegangan pada *input* inverting (V*input*) lebih positif dari pada *input* non inverting (Vref), maka *output*nya akan sama dengan (-Vsupply) dan bila tegangan pada *input* inverting (V*input*) lebih negatif dari pada *input* non inverting (Vref), maka *output*nya akan sama dengan (+Vsupply). Vref di hubungkan ke +V supply, kemudian R1 dan R2 digunakan sebagai pembagi tegangan, sehingg nilai tegangan yang di referensikan pada masukan + op-amp adalah sebesar :

$$Vref = \left(\frac{R1}{R1+R2}\right) \times (+Vsupply) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

2.3.2 Komparator *Close loop*

Jika pada komparator *Open Loop* tegangan keluaran selalu +Vsuppy dan -Vsupply, maka pada komparator *close loop output*nya dapat kita atur sesuai keinginan kita dengan cara mengubah resistansi R3 dan R4.



Gambar 2.2 : Konfigurasi Komparator *Close loop*
(Sumber : http://ilham-kn.blogspot.com/2013/12/komparator.html. diakses pada tanggal 01 Juni 2014)

Pada prinsipnya komparator *close loop* sama dengan komparator *Open Loop*, yaitu membandingkan masukan dan menghasilkan keluaran, akan tetapi pada kompartor *close loop* tegangan keluaran dapat kita atur dengan merubah R3 dan Rp dengan persamaan :

$$Vout = -\frac{R4}{R3} Vinput + \left(\frac{R1}{R2+R1}\right)\left(\frac{R3+R4}{R2}\right) Vref \quad \text{.....(2)}$$

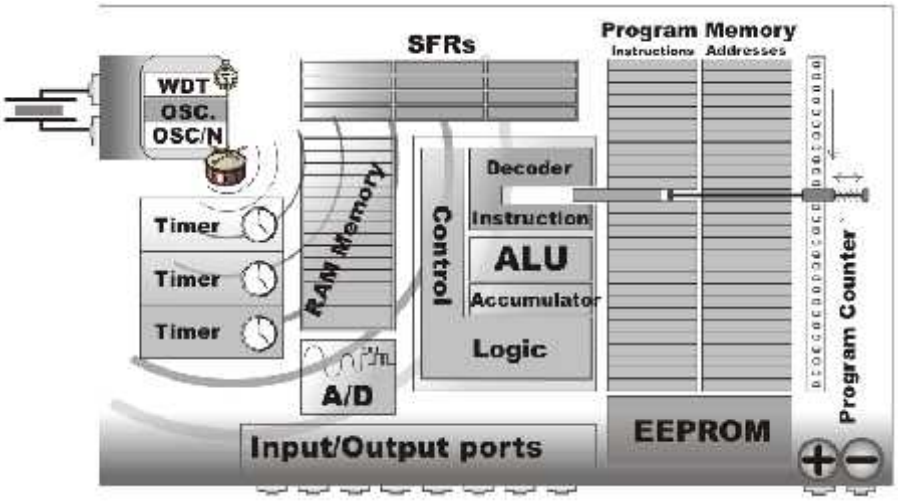
2.4 Definisi Mikrokontroler

Mikrokontroler merupakan suatu IC yang didalamnya berisi *CPU, ROM, RAM,* dan I/O yang dapat diprogram yang dapat disimpan didalam memory sehingga dapat mengendalikan perintah secara otomatis. Mikrokontroler terdiri dari beberapa bagian yaitu :

1. *CPU (Central Processing Unit)*
2. *RAM (Random Access Memory)*
3. *EEPROM/EPROM/PROM*
4. *I/O, Serial & Parallel*

5. *Timer*

6. *Interupt Controller*



Gambar 2.3: Struktur Dari Mikrokontroler

(Sumber : Prasanto. Pengenalan Mikrokontroler. 2008 : 8 . diakses pada tanggal 01 Juni 2014)

2.4.1 *Central Processing Unit* (*CPU*)

CPU merupakan bagian utama dalam suatu mikrokontroler. *CPU* pada mikrokontroler ada yang berukuran 8 bit ada pula yang berukuran 16 bit. *CPU* ini akan membaca program yang tersimpan di dalam *ROM* dan melaksanakannya.

2.4.2 *Read Only Memory* (*ROM*)

ROM merupakan suatu memori (alat untuk mengingat) yang sifatnya hanya dibaca saja. Dengan demikian *ROM* tidak dapat ditulisi. Dalam dunia mikrokontroler *ROM* digunakan untuk menyimpan program bagi mikrokontroler tersebut. Program tersimpan dalm format biner (‘0’ atau ‘1’). Susunan bilangan biner tersebut bila telah terbaca oleh mikrokontroler akan memiliki arti tersendiri.

2.4.3 *Random Acces Memory* (*RAM*)

Berbeda dengan *ROM*, *RAM* adalah jenis memori selain dapat dibaca juga dapat ditulis berulang kali. Tentunya dalam pemakaian mikrokontroler ada semacam data yang bisa berubah pada saat mikrokontroler tersebut bekerja. Perubahan data tersebut tentunya juga akan tersimpan ke dalam memori. Isi pada RAM akan hilang jika catu daya listrik hilang.

2.4.4 *Electrically Erasable Programmable Read Only Memory*

Beberapa mikrokontroler memiliki *EEPROM* yang terintegrasi pada chipnya. *EEPROM* ini dugunakan untuk menyimpan sejumlah kecil parameter yang dapat berubah dari waktu ke waktu. Jenis memori ini bekerja relatif pelan, dan kemampuan untuk dihapus/tulis nya juga terbatas.

2.4.5 *FLASH (EPROM)*

FLASH memberikan pemecahan yang lebih baik dari *EEPROM* ketika dibutuhkan sejumlah besar memori non-volatile untuk program. *FLASH* ini bekerja lebih cepat dan dapat dihapus/tulis lebih sering dibanding *EEPROM*.

2.4.6 *Input/Output*

Input biasa digunakandigunakan untuk

1. *UART* (*Universal Asynchronous Receiver Transmitter*) adalah adapter serial port adapter untuk komunikasi serial asinkron.
2. *USART* (*Universal Synchronous/Asynchronous Receiver Transmitter*) merupakan adapter serial port untuk komunikasi serial sinkron dan asinkron. Komunikasi serial sinkron tidak memerlukan *start* atau *stop* bit dan dapat beroperasi pada click yang lebih tinggi dibanding asinkron.
3. *SPI* (*serial peripheral interface*) merupakan port komunikasi serial sinkron.
4. *SCI* (*serial communications interface*) merupakan enhanced *UART* (asynchronous serial port)
5. *I2C* bus (*Inter-Integrated Circuit bus*) merupakan antarmuka serial 2 kawat yang dikembangkan oleh Philips. Dikembangkan untuk aplikasi 8 bit dan

banyak digunakan pada konsumem elektronik, otomotif dan indistri. *I2C* bus ini berfungsi sebagai antarmuka jaringan multi-master, multi-slave dengan deteksi tabrakan data. Jaringan dapat dipasangkan hingga 128 titik dalam jarak 10 meter. Setiap titik dalam jaringan dapat mengirim dan menerima data. Setiap titik dalam jaringan harus memiliki alamat yang unik.

6. *ADC* (*Analog to Digital Conversion*) Fungsi *ADC* adalah merubah besaran analog (biasanya tegangan) ke bilangan digital. Mikrokontroler dengan fasilitas ini dapat digunakan untuk aplikasi-aplikasi yang memerlukan informasi analog (misalnya voltmeter, pengukur suhu dll). Terdapat beberapa tipe dari ADC yaitu *Succesive Approximation A/D converters, Single Slope A/D converters, Delta-Sigma A/Ds converters, Flash A/D.*

7. *DAC* (*Digital to Analog Converters*) Kebalikan dar *ADC* yaitu merubah besaran digital (biasanya tegangan) ke bilangan analog.

2.4.7 *Interupt Controller*

Interupt merupakan metode yang efisien bagi mikrokontroler untuk memproses periperalnya, mikrokontroler hanya bekerja memproses periperal tersebut hanya pada saat terdapat data diperiperal tersebut. Pada saat terjadi interupt, mikrokontroler menunda operasi yang sedang dilakukan kemudian mengidentifikasi interupsi yang datang dan menjalankan rutin pelayanan interupsi.

Rata-rata mikrokontroler memiliki setidak-tidaknya sebuah interupsi eksternal, interupsi yang dimiliki bisa dipicu oleh "edge" atau "level". Edge triggered interupt bekerja tidak tergantung pada pada waktu terjadinya interupsi, tetapi interupsi bisa terjadi karena glitch. Sedangkan Level triggered interupt harus tetap pada logika high atau low sepanjang waktu tertentu agar dapat terjadi interupsi, interupsi ini tahan terhadap glitch.

2.4.8 *Timer*

Timer /Counter adalah suatu peripheral yang tertanam didalam microcontroller yang berfungsi pewaktu. Dengan peripheral ini pengguna

microcontroller dapat dengan mudah menentukan kapan suatu perintah dijalankan (delay), tentu saja fungsi timer tidak hanya untuk penundaan perintah saja, timer juga dapat berfungsi sebagai oscilator, PWM, ADC, dan lain-lain.

2.5 Servo Kontroler

Servo Kontroler adalah sebuah perangkat yang menerima sinyal perintah dari sistem kontrol pemrosesan atau mikrokontroler , dan mengirimkan arus listrik ke motor servo untuk menghasilkan gerak sebanding dengan sinyal diperintahkan. Biasanya sinyal perintah merupakan kecepatan yang diinginkan dan juga dapat mewakili torsi atau posisi yang diinginkan. Servo kontroler memiliki tujuan utama yaitu untuk mempermudah dalam mengendalikan servo yang jumlahnya lebih dari satu, tujuan lain dari servo kontroler yaitu dapat memperkecil penggunaan port pada chip pemrosesan atau mikrokontroler, karena servo kontroler terhubung dengan cara I2C pada chip pemrosesan (mikrokontroler).

2.5.1 Motor Servo

Motor servo adalah motor yang mampu bekerja dua arah (*CW* dan *CCW*) dimana arah dan sudut pergerakan rotornya dapat dikendalikan dengan memberikan variasi lebar pulsa (*duty cycle*) sinyal *PWM* pada bagian pin kontrolnya. Motor servo memiliki dua tipe yaitu servo standard dan servo *rotation* (*continuous*). Dimana biasanya untuk tipe standar hanya dapat melakukan pergerakan sebesar 180° sedangkan untuk tipe *continuous* dapat melakukan rotasi atau 360°.

Pada dasarnya motor servo tersusun dari motor DC, rangkaian kontrol, *gearbox* dan potensiometer. Tampak seperti gambar motor servo beserta komponen internal motor servo dibawah ini.

Gambar 2.4 : Bentuk Motor Servo

(Sumber : http://Wikipediaindonesia.co.id/Motor-Servo/2012.html. diakses pada tanggal 01 Juni 2014)

Terlihat jelas bahwa motor DC yang digunakan sangat kecil sehingga motor servo memiliki dimensi yang cukup kecil jika dibandingkan dengan motor DC pada umumnya. Rangkaian kontrol pada motor servo digunakan untuk mengontrol motor DC yang ada pada motor servo, dikarenakan untuk mengakses motor servo kita harus memberikan pulsa-pulsa kepada sinyal kontrol tersebut. Gearbox berfungsi untuk meningkatkan torsi dari motor servo, sebernarnya terdapat dua macam bahan penyusun *gearbox* yang digunakan untuk motor servo yaitu metal gear (biasanya untuk torsi yang sangat besar) dan nylon gear (berwarna putih seperti gambar diatas). Dan potensiometer berfungsi untuk menentukan batas sudut dari putaran servo. Sedangkan sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirimkan.

2.6 Definisi Bahasa C++

Bahasa C pertama kali digunakan di komputer Digital Equipment Corporation PDP-11 yang menggunakan sistem opersi UNIX C adalah bahasa yang standar, artinya suatu program yang ditulis dengan bahasa C tertentu akan dapat dikonversi dengan bahasa C yang lain dengan sedikit modifikasi. Standar bahasa C yang asli adalah standar dari UNIX. Patokan dari standar UNIX ini diambil dari buku yang ditulis oleh *Brian Kerningan dan Dennis Ritchie* berjudul “The C Programming Language”, diterbitkan oleh Prentice-Hall tahun 1978. Deskripsi C dari Kerninghan dan Ritchie ini kemudian kemudian dikenal secara umum sebagai “K dan R C”

2.6.1 PENULISAN PROGRAM BAHASA C

Program Bahasa C tidak mengenal aturan penulisan di kolom tertentu, jadi bisa dimulai dari kolom manapun. Namun demikian, untuk mempermudah pembacaan program dan untuk keperluan dokumentasi, sebaiknya penulisan bahasa C diatur sedemikian rupa sehingga mudah dan enak dibaca.
Berikut contoh penulisan Program Bahasa C:

```
#include <mega8535.h> #include <delay.h>
main ()
{
……
……
}
```

Program dalam bahasa C selalu berbentuk fungsi seperti ditunjukkan dalam **main ().** Program yang dijalankan berada di dalam tubuh program yang dimulai dengan tanda kurung buka **{** dan diakhiri dengan tanda kurung tutup **}**. Semua yang tertulis di dalam tubuh program ini disebut dengan blok.

Tanda () digunakan untuk mengapit **argumen** suatu fungsi. Argumen adalah suatu nilai yang akan digunakan dalam fungsi tersebut. Dalam fungsi **main** diatas tidak ada argumen, sehingga tak ada data dalam (). Dalam tubuh fungsi antara tanda **{** dan tanda **}** ada sejumlah pernyataan yang merupakan perintah yang harus dikerjakan oleh prosesor. Setiap pernyataan diakhiri dengan tanda titik koma **;**

Baris pertama **#include <…>** bukanlah pernyataan, sehingga tak diakhiri dengan tanda titik koma (;). Baris tersebut meminta kompiler untuk menyertakan file yang namanya ada di antara tanda **<…>** dalam proses kompilasi. File-file ini (ber-ekstensi .h) berisi deklarasi fungsi ataupun variable. File ini disebut **header**. File ini digunakan semacam perpustakaan bagi pernyataan yang ada di tubuh program.

#include merupakan salah satu jenis pengarah praprosesor *(preprocessor directive).* Pengarah praprosesor ini dipakai untuk membaca file yang di antaranya berisi deklarasi fungsi dan definisi konstanta. Beberapa file judul disediakan dalam C. File-file ini mempunyai ciri yaitu namanya diakhiri dengan ekstensi **.h.** Misalnya pada program #include <stdio.h> menyatakan pada kompiler agar membaca file bernama *stdio.h* saat pelaksanaan kompilasi.Bentuk umum #include:

#include "namafile"

Bentuk pertama (#include <namafile>) mengisyaratkan bahwa pencarian file dilakukan pada direktori khusus, yaitu direktori file *include.* Sedangkan bentuk kedua (#include "namafile") menyatakan bahwa pencarian file dilakukan pertama kali pada direktori aktif tempat program sumber dan seandainya tidak ditemukan pencarian akan dilanjutkan pada direktori lainnya yang sesuai dengan perintah pada sistem operasi.

2.6.2 TIPE DATA

Tipe data merupakan bagian program yang paling penting karena tipe data mempengaruhi setiap instruksi yang akan dilaksanakan oleh computer. Misalnya saja 5 dibagi 2 bisa saja menghasilkan hasil yang berbeda tergantung tipe datanya. Jika 5 dan 2 bertipe integer maka akan menghasilkan nilai 2, namun jika keduanya bertipe float maka akan menghasilkan nilai 2.5000000. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien dan efektif.

Tabel 2.1 Bentuk Tipe data

No	Tipe Data	Ukuran	Range (Jangkauan)
1	Char	1 byte	-128 s/d 127

2	Int	2 byte	-32768 s/d 32767
3	Unsigned int	2 byte	0 s/d 65535
4	Long Int	4 byte	-2147483648 s/d 2147483648
5	Unsigned Long int	4 byte	0 s/d 4294967296
6	Float	4 byte	-3.4E-38 s/d 3.4E+38
7	Double	8 byte	1.7E-308 s/d 1.7E+308
8	Long Double	10 byte	3.4E-4932 s/d 1.1E+4932
9	Char	1 byte	-128 s/d 127
10	Unsigned char	1 byte	0 s/d 255

2.6.3 KONSTANTA

Konstanta merupakan suatu nilai yang tidak dapat diubah selama proses program berlangsung. Konstanta nilainya selalu tetap. Konstanta harus didefinisikan terlebih dahulu di awal program. Konstanta dapat bernilai integer, pecahan, karakter dan string.

Contoh konstanta : 50; 13; 3.14; 4.50005; ‘A’; ‘Bahasa C’.

2.6.4 VARIABEL

Variabel adalah suatu pengenal (identifier) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu variable bisa diubah-ubah sesuai

kebutuhan. Nama dari suatu variable dapat ditentukan sendiri oleh pemrogram dengan aturan sebagai berikut :

1. Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Bahasa C bersifat case-sensitive artinya huruf besar dan kecil dianggap berbeda.
2. Tidak boleh mengandung spasi.
3. Tidak boleh mengandung symbol-simbol khusus, kecuali garis bawah (underscore). Yang termasuk symbol khusus yang tidak diperbolehkan antara lain : $, ?, %, #, !, &, *, (,), -, +, = dsb
4. Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai.

2.6.5 DEKLARASI

Deklarasi diperlukan bila kita akan menggunakan pengenal (identifier) dalam program. Identifier dapat berupa variable, konstanta dan fungsi.

2.6.5.1 DEKLARASI VARIABEL

Bentuk umum pendeklarasian suatu variable adalah : Nama_tipe nama_variabel; Contoh :

int x; // *Deklarasi x bertipe integer*

2.6.5.2 DEKLARASI KONSTANTA

Dalam bahasa C konstanta dideklarasikan menggunakan preprocessor #define.

Contohnya :

#define PHI 3.14 #define nim “0111500382”

2.6.5.3 DEKLARASI FUNGSI

Fungsi merupakan bagian yang terpisah dari program dan dapat diaktifkan atau dipanggil di manapun di dalam program. Fungsi dalam bahasa C ada yang sudah disediakan sebagai fungsi pustaka seperti printf(), scanf(), getch()

dan untuk menggunakannya tidak perlu dideklarasikan.

Fungsi yang perlu dideklarasikan terlebih dahulu adalah fungsi yang dibuat oleh programmer. Bentuk umum deklarasi sebuah fungsi adalah :

Tipe_fungsinama_fungsi(parameter_fungsi) ;

Contohnya :

```
float luas_lingkaran(int jari);
void tampil();
int tambah(int x, int y);
```

2.6.6 OPERATOR

Operator adalah symbol-simbol yang mempunyai fungsi-fungsi tertentu sesuai dengan simbolnya masing-masing. Ada banyak macam operator, antaralain :

2.6.6.1 OPERATOR PENUGASAN

Operator penugasan (*Assignment operator*) dalam bahasa C berupa tanda sama dengan (“=”).

2.6.6.2 OPERATOR ARITMATIKA

Bahasa C menyediakan lima operator aritmatika, yaitu :

1 .* : untuk perkalian

2 / : untuk pembagian

3 % : untuk sisa pembagian (modulus)

4 + : untuk penjumlahan

5 - : untuk pengurangan

2.6.6.3 OPERATOR HUBUNGAN (PERBANDINGAN)

Operator hubungan digunakan untuk membandingkan hubungan antara dua buah operand /sebuah nilai atau variable. Operasi majemuk seperti pada tabel dibawah ini:

Tabel 2.2 Operator Hubungan

Operator	Arti	Contoh	
<	Kurang dari	X<Y	Apakah X kurang dari Y
<=	Kurang dari sama dengan	X<=Y	Apakah X Kurang dari sama dengan Y
>	Lebih dari	X>Y	Apakah X Lebih dari Y
>=	Lebih dari sama dengan	X==Y	Apakah X Lebih dari sama dengan Y
==	Sama dengan	X==Y	Apakah X Sama dengan Y
!=	Tidak sama dengan	X!= Y	Apakah X Tidak sama dengan Y

2.6.6.4 OPERATOR LOGIKA

Jika operator hubungan membandingkan hubungan antara dua buah operand, maka operator logika digunakan untuk membandingkan logika hasil dari operator-operator hubungan.

Operator logika ada tiga macam, yaitu :

1. && : Logika AND (DAN)

2. || : Logika OR (ATAU)
3. ! : Logika NOT (INGKARAN)

Operasi AND akan bernilai benar jika dua ekspresi bernilai benar. Operasi OR akan bernilai benar jika dan hanya jika salah satu ekspresinya bernilai benar. Sedangkan operasi NOT menghasilkan nilai benar jika ekspresinya bernilai salah, dan akan bernilai salah jika ekspresinya bernilai benar.

```
#include <mega8535.h>
#include <delay.h> void
Void Main()
{
        Char in1;
        Char in2;
        DDRB=0xFF;
        PORTB=0xFF
        In1=0xFF
        In2=0x40
        Id((in1==0xf0) && (in2==0x40)
}
```

2.6.6.5 OPERATOR BITWISE (MANIPULASI PER BIT)

Operator bitwise digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori.

Operator bitwise dalam bahasa C di SDCC adalah sebagai berikut :

1. << : Pergeseran bit ke kiri
2. >> : Pergeseran bit ke kanan
3. & : Bitwise AND
4. ^ : Bitwise XOR (exclusive OR)

5. | : Bitwise OR
6. ~ : Bitwise NOT
7. Pertukaran Nibble dan Byte
8. Mengambil Bit yang paling Berbobot

1. OPERASI GESER KIRI (<<)

Operasi geser kiri merupakan operasi yang akan menggeser bit-bit kekiri sehingga bit 0 akan berpindah ke bit 1 kemudian bit 1 akan berpindah ke bit 2 dan seterusnya. Operasi geser kiri membutuhkan dua buah operan disebelah kiri tanda << merupakan nilai yang akan digeser sedangkan disebelah kanannya merupakan jumlah bit penggerseran.

Contohnya :

Datanya = 0x03 << 2 ; // 0x03 digeser kekiri 2 bit hasilnya ditampung
didatanya a << = 1 // Isi variabel A digeser ke kiri 1 bit hasilnya
// kembali disimpan di A

2. OPERASI GESER KANAN(>>)

Operasi geser kiri merupakan operasi yang akan menggeser bit-bit kekanan sehingga bit 7 akan berpindah ke bit 6 kemudian bit 6 akan berpindah ke bit 5 dan seterusnya. Operasi geser kanan membutuhkan dua buah operan disebelah kiri tanda << merupakan nilai yang akan digeser sedangkan disebelah kanannya merupakan jumlah bit penggerseran.

Contohnya :

Datanya = 0x03 >> 2 ; // 0x03 digeser kekiri 2 bit hasilnya ditampung di
datanya a >> = 1 // Isi variabel A digeser ke kiri 1 bit hasilnya
// kembali disimpan di A

3. OPERASI BITWISE AND (&)

Operasi bitwise AND akan melakukan operasi AND pada masing-masing bit, sehingga bit 0 akan dioperasikan dengan bit 0 dan bit 1 dan seterusnya.

Contohnya :

Hasil = 0x05 | 0x31;

	Operasinya	0x01 = 00000001
		0x31 = 00110001
	Hasil	0x01 = 00110001

4. OPERASI BITWISE XOR(^)

Operasi bitwise XOR akan melakukan operasi XOR pada masing-masing bit, sehingga bit 0 akan dioperasikan dengan bit 0 dan bit 1 dan seterusnya.

Contohnya :

Hasil = 0x02 ^ 0xFA;

	Operasinya	0x02 = 00000010
		0xFA = 11111010
	Hasil	0x01 = 11111000

5. OPERASI BITWISE NOT(~)

Operasi bitwise XOR akan melakukan operasi XOR pada masing-masing bit, sehingga bit 0 akan dioperasikan dengan bit 0 dan bit 1 dan seterusnya.

Contohnya :

Hasil = ~ 0x31; 0x31 = 00110001 Hasil ~0x31 =
11001110

6. PERTUKARAN NIBBLE DAN BYTE

Pertukaran nibble dalam bahasa C dikenali SDCC dengan bentuk pernyataan sebagai berikut ini:

```
volatile unsigned char i:
i = (( i << 4) | ( i >> 4));  //pertukaran nibble
```

Dan pernyataan sebagai berikut ini sebagai pertukaran byte:

```
volatile unsigned char j:
j = (( j << 8) | ( j >> 8));  //pertukaran byte
```

7. MENGAMBIL BIT YANG PALING BERBOBOT

Untuk mendapatkan bit yang paling berbobot (MSB) untuk tipe long, short, int, dan char maka dapat dilakukan dengan pertanyaan berikut:

```
Volatile unsigned char gint;
Unsigned char hop;
Hop = (gint >> 7) & 1    // mengambil MSB
```

2.6.6.6 OPERATOR UNARY

Operator Unary merupakan operator yang hanya membutuhkan satu operand saja. Dalam bahasa C terdapat beberapa operator unary, yaitu :

Tabel 2.3 Operasi Unary

Operator	Arti/Maksud	Letak	Contoh	Equivalen
-	Unary minus	Sebelum operator	A+-B*C	A+(-B)*C
++	Peningkatan dengan penambahan nilai 1	Sebelum dan sesudah	A++	A=A+1
--	Penurunan dengan pengurangan nilai 1	Sebelum dan sesudah	A--	A=A-1
Sizeof	Ukuran darioperasi dalam byte	Sebelum	Sizeof(1)	-
!	Unary not	Sebelum	!A	-
~	Bitwise Bot	Sebelum	~A	-
&	Menghasilkan alamat memori operasi	Sebelum	&A	-
*	Menghasilkan nilai dari pointer	Sebelum	*A	-

Contohnya :

```
n = 0
Jum = 2 *
    ++n;
    Jum = 2
    * n++;
```

2.6.6.7 OPERATOR MAJEMUK

Operator majemuk terdiri dari dua operator yang digunakan untuk menyingkat penulisan. Operasi majemuk seperti pada tabel dibawah ini

Tabel 2.4 Operasi Majemuk

Operator	Contoh	Kependekan dari
+=	Counter +=1;	Counter = counter + 1
-=	Counter -=1	Counter = counter - 1
*=	Counter *=1	Counter = counter * 1
/=	Counter /=1	Counter = counter / 1
%=	Counter %=1	Counter = counter % 1
<<=	Counter <<=1	Counter = counter << 1
>>=	Counter >>=1	Counter = counter >> 1
&=	Counter &=1	Counter = counter & 1
|=	Counter |=1	Counter = counter | 1
^=	Counter ^=1	Counter = counter ^ 1
~=	Counter ~=1	Counter = counter ~ 1

2.6.7 KOMENTAR PROGRAM

Komentar program hanya diperlukan untuk memudahkan pembacaan dan pemahaman suatu program (untuk keperluan dokumentasi program). Dengan kata lain, komentar program hanya merupakan keterangan atau penjelasan program. Untuk memberikan

komentar atau penjelasan dalam bahasa C digunakan pembatas /* dan */ atau menggunakan tanda // untuk komentar yang hanya terdiri dari satu baris. Komentar program tidak akan ikut diproses dalam program (akan diabaikan).

Contoh pertama :

// *program ini dibuat oleh ….*

Dibelakang tanda // tak akan diproses dalam kompilasi. Tanda ini hanya untuk satu baris kalimat.

Contoh kedua :

/* *program untuk memutar motor*
***DC atau motor stepper* /**

Bentuk ini berguna kalau pernyataannya berupa kalimat yang panjang sampai beberapa baris.

2.6.8 PENYELEKSIAN KONDISI

Penyeleksian kondisi digunakan untuk mengarahkan perjalanan suatu proses. Penyeleksian kondisi dapat diibaratkan sebagai katup atau kran yang mengatur jalannya air. Bila katup terbuka maka air akan mengalir dan sebaliknya bila katup tertutup air tidak akan mengalir atau akan mengalir melalui tempat lain. Fungsi penyeleksian kondisi penting artinya dalam penyusunan bahasa C, terutama untuk program yang kompleks.

2.6.8.1 STRUKTUR KONDISI "IF…."

Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang ada di dalam blok if akan diproses dan dikerjakan.

Bentuk umum struktur kondisi if adalah :

```
if(kondisi)
        pernyataan;
```

2.6.8.2 STRUKTUR KONDISI "IF.....ELSE…."

Dalam struktur kondisi if.....else minimal terdapat dua pernyataan. Jika kondisi yang diperiksa bernilai benar atau terpenuhi maka pernyataan pertama yang dilaksanakan dan jika kondisi yang diperiksa bernilai salah maka pernyataan yang kedua yang dilaksanakan. Bentuk umumnya adalah sebagai berikut :

```
if(kondisi)
        pernyataan
        -1
else
        pernyataan-2
```

2.6.8.3 STRUKTUR KONDISI "SWITCH...CASE... DEFAULT…"

Struktur kondisi switch....case....default digunakan untuk penyeleksian kondisi dengan kemungkinan yang terjadi cukup banyak. Struktur ini akan melaksanakan salah satu dari beberapa pernyataan 'case' tergantung nilai kondisi yang ada di dalam switch. Selanjutnya proses diteruskan hingga ditemukan pernyataan 'break'. Jika tidak ada nilai pada case yang sesuai dengan nilai kondisi, maka proses akan diteruskan kepada pernyataan yang ada di bawah 'default'.

Bentuk umum dari struktur kondisi ini adalah :

```
switch(kondisi)
{
     case 1 :
```

```
        pernyataan-1;
        break;
        case 2 :
        pernyataan-2;
        break;
        .....
        case n :
        pernyataan-n;
        break;
        default : pernyataan-m
}
```

2.6.9 PERULANGAN

Dalam bahasa C tersedia suatu fasilitas yang digunakan untuk melakukan proses yang berulangulang sebanyak keinginan kita. Misalnya saja, bila kita ingin meng*input* dan mencetak bilangan dari 1 sampai 100 bahkan 1000, tentunya kita akan merasa kesulitan. Namun dengan struktur perulangan proses, kita tidak perlu menuliskan perintah sampai 100 atau 1000 kali, cukup dengan beberapa perintah saja. Struktur perulangan dalam bahasa C mempunyai bentuk yang bermacam-macam.

2.6.9.1 STRUKTUR PERULANGAN " WHILE"

Perulangan WHILE banyak digunakan pada program yang terstruktur. Perulangan ini banyak digunakan bila jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (true) dan akan berhenti bila kondisinya bernilai salah.

Bentuk umum dari struktur kondisi ini adalah:

```
While (ekspresi)
{
        Pernyataan_1
        Pernyataan_2
```

```
}
```

Contoh Program 1 :

```
while (!TF0);
        {
            TF0 = 0;
            TR0 = 0;
        }
```

2.6.9.2 STRUKTUR PERULANGAN "DO.....WHILE…"

Pada dasarnya struktur perulangan do....while sama saja dengan struktur while, hanya saja pada proses perulangan dengan while, seleksi berada di while yang letaknya di atas sementara pada perulangan do....while, seleksi while berada di bawah batas perulangan. Jadi dengan menggunakan struktur do…while sekurang-kurangnya akan terjadi satu kali perulangan.

Bentuk umum dari struktur kondisi ini adalah:

```
Do
     {
          Pernyataan_1
          Pernyataan_2
     }
While (ekspresi)
```

2.6.9.3 STRUKTUR PERULANGAN "FOR"

Struktur perulangan for biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah perulangannya. Dari segi penulisannya, struktur perulangan for tampaknya lebih efisien karena susunannya lebih simpel dan sederhana. Bentuk umum perulangan for adalah sebagai berikut :

```
for(inisialisasi; syarat;
     penambahan) pernyataan;
```

Keterangan:

Inisialisasi : pernyataan untuk menyatakan keadaan awal dari variabel kontrol. ***syarat*** : ekspresi relasi yang menyatakan kondisi untuk keluar dari perulangan. ***penambahan*** : pengatur perubahan nilai variabel kontrol.

Contoh

```
for  (k = 0; k<4 ; k++)
     {
          i=tabel1(
          k); PORTA
          = i;
          tunda50(1
          00);
     }
```

2.6.10 ARAY (LARIK)

Array merupakan kumpulan dari nilai-nilai data yang bertipe sama dalam urutan tertentu yang menggunakan nama yang sama. Letak atau posisi dari elemen array ditunjukkan oleh suatu index. Dilihat dari dimensinya array dapat dibagi menjadi Array dimensi satu, array dimensi dua dan array multi-dimensi.

2.6.10.1 ARRAY DIMENSI SATU

Setiap elemen array dapat diakses melalui indeks. Indeks array secara default dimulai dari 0. Deklarasi Array Bentuk umum :

Deklarasi array dimensi satu:

[Tipe_array][nama_array][elemen1];

2.6.10.2 ARRAY DIMENSI DUA

Array dua dimensi merupakan array yang terdiri dari m buah baris dan n buah kolom. Bentuknya dapat berupa matriks atau tabel.

Deklarasi array dimensi dua :

[Tipe_array][nama_array][elemen1][ele
men2];

2.6.10.3 ARRAY MULTI-DIMENSI

Array multi-dimensi merupakan array yang mempunyai ukuran lebih dari dua. Bentuk pendeklarasian array sama saja dengan array dimensi satu maupun array dimensi dua. Bentuk umumnya yaitu :

[tipe_array][nama_array][elemen1][elemen2]…[elemenN];

2.6.11 FUNGSI

Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilnya. Fungsi merupakan elemen utama dalam bahasa C karena bahasa C sendiri terbentuk dari kumpulan fungsi-fungsi. Dalam setiap program bahasa C, minimal terdapat satu fungsi yaitu fungsi main(). Fungsi banyak diterapkan dalam program-program C yang terstruktur. Keuntungan penggunaan fungsi dalam program yaitu program akan memiliki struktur yang jelas (mempunyai readability yang tinggi) dan juga akan menghindari penulisan bagian program yang sama.

2.6.11.1 PENDEFISIAN FUNGSI

Sebelum digunakan fungsi harus didefinisikan terlebih dahulu. Bentuk definisi fungsi adalah:

```
Tipe_Nilai_Balik nama_fungsi(argumen1, argumen2)
     {
          Pernyataan1;
               Pernyataan1;
               return(ekspresi);
     }
```

Contoh:

int jumlah(int bil1,int bil2) //definisi fungsi jumlah

```
{
        int hasil;
        hasil = bil1 +
                bil2
                return(
                hasil);
}
int jumlah(int bil1,int bil2)
```

1 2 3 4

Keterangan:

1. tipe data nilai balik fungsi
2. merupakan nama fungsi
3. tipe argumen
4. nama argumen

2.7 Definisi Code Vision AVR

CodeVisionAVR pada dasarnya merupakan perangkat lunak pemrograman microcontroller keluarga AVR berbasis bahasa C. Ada tiga komponen penting yang telah diintegrasikan dalam perangkat lunak ini: Compiler C, IDE dan Program generator.

Gambar 2.5 Logo Code Vision AVR

(Sumber : http://pccontrol.wordpress.com/dasar-c-avr-codevision/. diakses pada tanggal 01 Juni 2014)

Berdasarkan spesifikasi yang dikeluarkan oleh perusahaan pengembangnya, Compiler C yang digunakan hampir mengimplementasikan semua komponen standar yang ada pada bahasa C standar ANSI (seperti struktur program, jenis tipe data, jenis operator, dan *library* fungsi standar-berikut penamaannya). Tetapi walaupun demikian, dibandingkan bahasa C untuk aplikasi komputer, compiler C untuk mikrokontroler ini memiliki sedikit perbedaan yang disesuaikan dengan arsitektur AVR tempat program C tersebut ditanamkan (*embedded*). Khusus untuk *library* fungsi, disamping *library* standar (seperti fungsi-fungsi matematik, manipulasi String, pengaksesan memori dan sebagainya), CodeVisionAVR juga menyediakan fungsi-fungsi tambahan yang sangat bermanfaat dalam pemrograman antarmuka AVR dengan perangkat luar yang umum digunakan dalam aplikasi kontrol. Beberapa fungsi *library* yang penting diantaranya adalah fungsi-fungsi untuk pengaksesan LCD, komunikasi I2C, IC RTC (*Real time Clock*), sensor suhu LM75, SPI (*Serial Peripheral Interface*) dan lain sebagainya.

Untuk memudahkan pengembangan program aplikasi, CodeVisionAVR juga dilengkapi IDE yang sangat *user friendly*, selain menu-menu pilihan yang umum dijumpai pada setiap perangkat lunak berbasis Windows, CodeVisionAVR ini telah mengintegrasikan perangkat lunak *downloader* yang dapat digunakan untuk mentransfer kode mesin hasil kompilasi kedalam sistem memori mikrokontroler AVR yang sedang deprogram. CodeVisionAVR adalah suatu kompiler berbasis bahasa C, yang terintegrasi untuk memprogram dan sekaligus compiler aplikasi AVR (*Alf and Vegard's Risc processor*) terhadap mikrokontroler dengan sistem berbasis window. CodeVisionAVR ini dapat mengimplematasikan hampir semua interuksi bahasa C yang sesuai dengan arsitektur AVR, bahkan terdapat beberapa keunggulan tambahan untuk memenuhi keunggulan spesifikasi dari CodeVisionAVR yaitu hasil kompilasi studio *debugger* dari ATMEL.

Integrated Development Environtment (IDE) telah diadaptasikan pada chip AVR yaitu *In-System Programmer software*, memungkinkan programmer untuk mentransfer program ke chip mikrokontroler secara otomatis setelah proses

assembly/kompilasi berhasil. *In-System Programmer software* didesign untuk bekerja dan dapat berjalan dengan perangkat lunak lain seperti AVR Dragon, AVRISP, Atmel STK500, dan lain sebagainya. Disamping *library* standar C, CodeVisionAVR C compiler memiliki librari lain untuk:

1. Modul LCD *Alpanumerik*
2. *Delays*
3. *Protokol Semikonduktor Maxim/Dallas*, Dan lainya.

CodeVisionAVR juga memiliki CodeWizardAVR sebagai generator program otomatis, yang memungkinkan kita untuk menulis, segala bentuk pengaturan Chip dalam waktu singkat, dan semua kode yang dibutuhkan untuk mengimplementasikan fungsi-fungsi seperti:

1. Pengaturan akses *External Memory*

Untuk chip-chip AVR yang memungkinkan koneksi memori eksternal SRAM, dapat juga mengatur ukuran memori dan wait state (tahap tunggu) dari memori ketika memori tersebut diakses.

2. Identifikasi *chip reset source*

Adalah suatu layanan dimana kita dapat membuat kode secara otomatis yang dapat mengidentifikasi kondisi yang menyebabkan *chip di reset.*

3. Inisialisasi *port input/output*

Pengaturan port-port yang kan dijadikan gerbang masukan dan keluaran dapat secara otomatis *digenerate* codenya. Yang kita lakukan hanya memilih *port-port* yang akan digunakan sebagai *input* atau *output*.

4. Inisialisasi *Interupsi external*

Pengaturan *interupsi eksternal* yang nantinya akan digunakan untuk menginterupsi program utama

5. Inisialisasi *timers/counters*

Pengaturan *timers* yang berfungsi untuk mengatur frekuensi yang nantinya digunakan pada interupsi.

6. Inisialisasi *timer watchdog*

Pengaturan *timers* yang berfungsi untuk mengatur frekuensi yang nantinya digunakan pada *interupsi*, sehingga *interupsi* akan dilayani oleh suatu fungsi *wdt_timeout_isr* .

7. Inisialisasi *UART* (*USART*) dan komunuikasi serial

Pengaturan komunikasi serial sebagai penerima atau pengirim data.

8. Inisialisasi komparasi analog

Pengaturan yang berkaitan dengan masukan data yang digunakan dalam aplikasi yang membutuhkan komparasi pada ADC nya.

9. Inisialisasi *ADC*

Pengaturan *ADC* (*Analog-Digital Converter*) yang berfungsi untuk merubah format analog menjadi format digital untuk diolah lebih lanjut.

10. Inisialisasi antarmuka SPI

Pengaturan chip yang berkaitan dengan *Clock rate, Clock Phase,* dan lainnya.

11. Inisialisasi antarmuka *Two Wire BUS*

Pengaturan Chip yang berhubungan dengan pola jalur komunikasi antara register yang terdapat pada chip AVR.

12. Inisialisasi antarmuka CAN

Pengaturan chip yang lebih kompleks, yang dapat mengatur *interupsi*, transmisi data, *timers*, dan lainnya.

13. Inisialisasi sensor temperatur, thermometer, dan lainnya

Pengaturan yang berhubungan dengan sensor temperatur *one wire bus*, memiliki fungsi-fungsi yang ada pada librari CodeVisionAVR.

14. Inisialisasi *one wire bus*

Pengaturan yang berhubungan dengan sensor temperatur yang memiliki fungsi-fungsi yang ada pada librari CodeVisionAVR. Seperti Maxim/Dallas Semiconductor.

15. Inisialisasi modul LCD

Pengaturan port-port yang kan digunakan sebagai penghubung dengan LCD *alphanumerik.*

Untuk lebih jelas tampilan pengaturan yang disediakan oleh AVR dapat dilihat pada gambar dibawah ini :

Gambar 2.6 Tampilan Pegaturan CodeVisionAVR

(Sumber : http://pccontrol.wordpress.com/dasar-c-avr-codevision/. diakses pada tanggal 01 Juni 2014)

2.8 Komponen-Komponen

Dalam pembuatan robot penari *Humanoid* ini terdapat beberapa jenis komponen yang digunakan, kompnen komponen tersebut terdapat pada bagian : komponen sensor suara, komponen pemeroses, dan komponen motor dan driver motor.

2.8.1 Komponen Sensor Warna

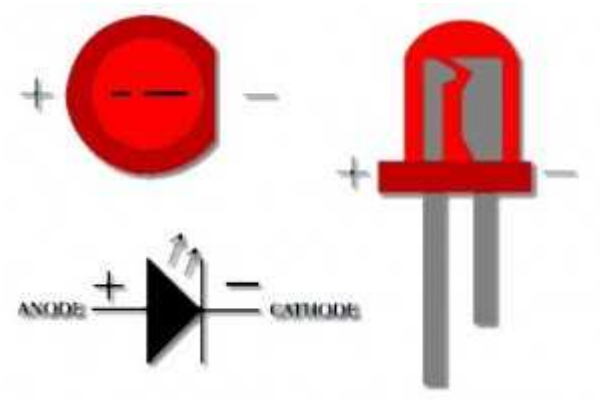
Sensor Warna yang penulis rancang aadalah menggunakan prinsip komperator yang berfungsi membandingkan sinyal *output* dari Photodioda dengan tegangan referensi, sehingga mendapatkan logika 1 atau 0 pada *output* komperator. Sensor warna tersebut tersusun dari bebe rapa komponen, seperti : LED, Photodioda, resistor, dan IC LM358.

2.8.1.1 Light Emiting Dioda(LED)

Lampu LED atau kepanjangannya *Light Emitting Diode* adalah suatu lampu indikator dalam perangkat elektronika yang biasanya memiliki fungsi untuk menunjukkan status dari perangkat elektronika tersebut.

Kualitas cahayanya memang berbeda dibandingkan dengan lampu TL atau lampu lainnya. Tingkat pencahayaan LED dalam ruangan memang tak lebih terang dibandingkan lampu neon, inilah mengapa LED dianggap belum layak dipakai secara luas. Untungnya para ilmuwan di University of Glasgow menemukan cara untuk membuat LED bersinar lebih terang. Solusinya adalah

dengan membuat lubang mikroskopis pada permukaan LED sehingga lampu bisa menyala lebih terang tanpa menggunakan tambahan energi apapun. Pelubangan tersebut menerapkan sistem *nano-imprint litography* yang sampai saat ini proyeknya masih dikembangkan bersama-sama dengan Institute of Photonics. Sementara ini beberapa jenis lampu LED sudah dipasarkan oleh Philips. Anda bisa menemui beberapa model lampu LED bergaya bohlam yang hadir dalam warna putih susu dan juga warna-warni. Daya yang diperlukan lampu jenis ini hanya sekitar 4-10 watt saja dibandingkan lampu neon sejenis yang mencapai 12-20 watt. Jika dihitung secara seksama memang bisa diakui bahwa lampu LED menggunakan daya yang lebih hemat daripada lampu TL.



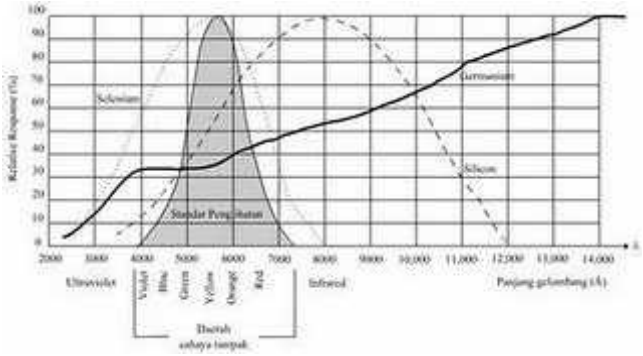
Gambar 2.7 Light Emiting Diode

(Sumber : http://sulistina.weebly.com/pengertian-led.html)

2.8.1.2 Photodioda

Photodioda adalah suatu jenis dioda yang resistansinya berubah-ubah kalau cahaya yang jatuh pada dioda berubahubah intensitasnya.Dalam gelap nilai tahanannya sangat besar hingga praktis tidak ada arus yang mengalir.Semakin kuat cahaya yang jatuh pada dioda maka makin kecil nilai tahanannya, sehingga arus yang mengalir semakin besar. Jika photodioda persambungan p-n bertegangan balik disinari, maka arus akan berubah secara linier dengan kenaikan fluks cahaya yang dikenakan pada persambungan tersebut. Photodioda terbuat dari bahan semikonduktor. Biasanya yang dipakai adalah silicon (Si) atau gallium arsenide (GaAs), dan lain-lain termasuk indium

antimonide (InSb), indium arsenide (InAs), lead selenide (PbSe), dan timah sulfide (PBS). Bahan-bahan ini menyerap cahaya melalui karakteristik jangkauan panjang gelombang, misalnya: 250 nm ke 1100 untuk nm silicon, dan 800 nm ke 2,0 μm untuk GaAs.



Gambar 2. Karakteristik Photodioda

(Sumber : http://ryankudeta.wordpress.com/2012/12/17/pengertian-photodioda/)

2.8.1.3 IC LM 339

LM 339 merupakan rangkaian terintegrasi yang memiliki dua penguat operasional. Terdiri dari 8 masukan, memiliki faktor penguatan yang besar dan frekuensi internal yang berubah-ubah, yang mana di desain secara spesifik untuk beroperasi dari sebuah power supply melalui sebuah range tegangan. IC ini memilliki spesifikasi sebagi berikut :

1. Frekuensi internal yang dapat di ubah untuk penguatanya.
2. Penguatan tegangan yang besar (100dB).
3. Memiliki besar range tegangan antara 3V-32V.
4. Arus bias *input* rendah (20nA).
5. Arus offset *input* rendah (2nA).
6. Tegangan offset *input* rendah (2mV).
7. Tegangan *output* besar, berkisar 0 sampai (Vcc-1,5V).

Gambar 2.9 IC LM339

2.8.1.4 Resistor

Resistor merupakan sarana untuk mengontrol atau tahanan arus dan tegangan yang bekerja dalam rangkaian-rangkaian elektronik. Spesifikasi-spesifikasi untuk suatu resistor umumnya meliputi nilai resistansi (dinyatakan dalam ohm (Ω), kilohm (kΩ) atau megohm (MΩ), nilai ketepatan atau toleransi (dinyatakan sebagai penyimpangan maksimum yang diizinkan dari nilai yang tertera), dan rating daya (yang harus sama dengan atau lebih besar daripada disipasi daya maksimumnya) (Tooley, Michael. 2003:19).

Adapun fungsi dari resistor, yaitu :

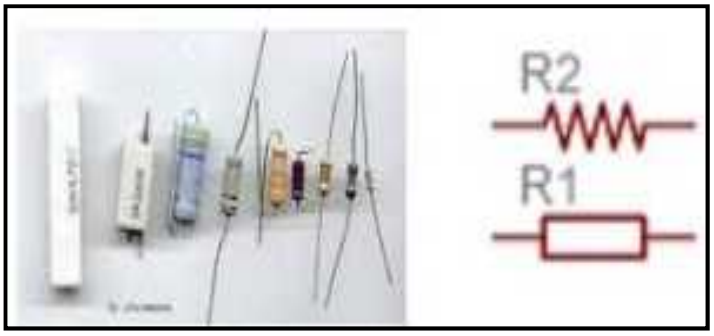
1. Menahan sebagian arus listrik agar sesuai dengan kebutuhan suatu rangkaian elektronika.
2. Menurunkan tegangan sesuai dengan yang dibutuhkan oleh rangkaian elektronika.
3. Membagi tegangan.
4. Bekerja sama dengan transistor dan kondensator dalam suatu rangkaian untuk membangkitkan frekuensi tinggi dan frekuensi rendah.

Dilihat dari fungsinya, resistor dapat dibagi menjadi :

1. Resistor Tetap (*Fixed Resistor*)

Yaitu resistor yang nilainya tidak dapat berubah, jadi selalu tetap (*konstan*). Resistor ini biasanya dibuat dari karbon. Berfungsi sebagai pembagi

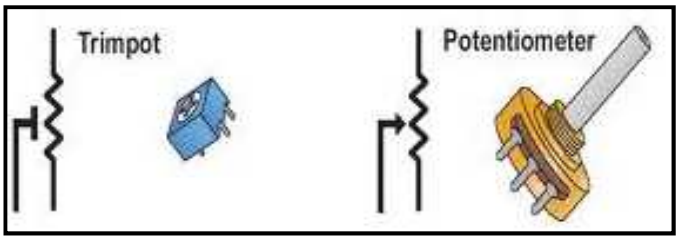
tegangan, mengatur atau membatasi arus pada suatu rangkaian serta memperbesar dan memperkecil tegangan.



Gambar 2.10: Bentuk Fisik dan Lambang Fixed Resistor
(Rangkaian Elektronika.info.2013 http://rangkaianelektronika.info/fungsi-dan-jenis-jenis-resistor/ diakses pada tanggal 31 Mei 2014)

2. Resistor Tidak Tetap (*Variable Resistor*)

Yaitu resistor yang nilainya dapat berubah-ubah dengan jalan menggeser atau memutar toggle pada alat tersebut, sehingga nilai resistor dapat kita tetapkan sesuai dengan kebutuhan. Berfungsi sebagai pengatur *volume* (mengatur besar kecilnya arus), tone control pada *sound system*, pengatur tinggi rendahnya nada (*bass* atau t*reble*) serta berfungsi sebagai pembagi tegangan arus dan tegangan.



Gambar 2.11 Bentuk Fisik dan Lambang Variable Resistor
(Sumber :Rangkaian Elektronika.info.2013
http://rangkaianelektronika.info/fungsi-dan-jenis-jenis-resistor/
diakses pada tanggal 31 Mei 2014)

2.8.2 Komponen Mikrokontroler

Mikrokontroler merupakan tempat pemrosesan *input* sehingga menghasilkan *output* yang dapat bekerja secara otomatis berdasarkan program yang telah dituliskan dan tersimpan pada memori chip mikro. Adapun komponen pendukung pada mikrokontroler yang penulis gunakan yaitu : IC ATmega 8535, dan Kristal.

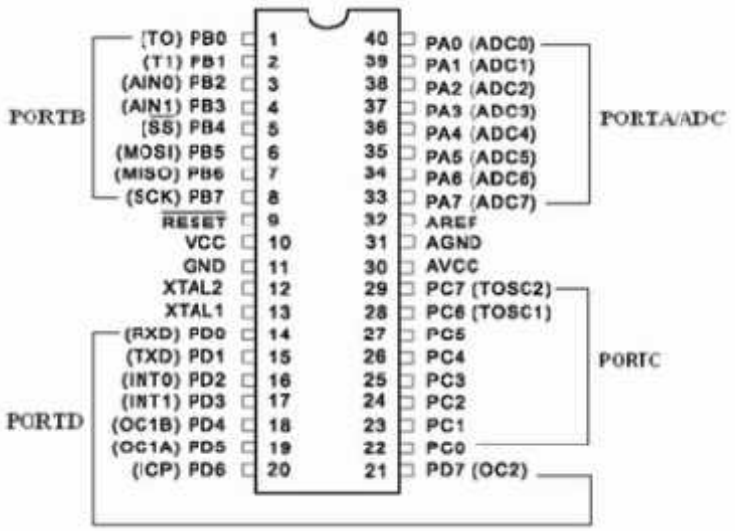
2.8.2.1 IC ATMEGA 8535

ATMega8535 adalah mikrokontroler CMOS 8 *bit* daya rendah berbasis arsitektur RISC. Instruksi dikerjakan pada satu siklus *clock*, ATMega8535 mempunyai *throughput* mendekati 1 MIPS per MHz, hal ini membuat ATMega8535 dapat bekerja dengan kecepatan tinggi walaupun dengan penggunaan daya rendah. Mikrokontroler ATmega8535 memiliki beberapa fitur atau spesifikasi yang menjadikannya sebuah solusi pengendali yang efektif untuk berbagai keperluan. Fitur-fitur tersebut antara lain:

1. Saluran I/O sebanyak 32 buah, yang terdiri atas *Port* A, B, C dan D
2. ADC (*Analog to Digital Converter*)
3. Tiga buah *Timer/Counter* dengan kemampuan perbandingan
4. *CPU* yang terdiri atas 32 *register*
5. *Watchdog Timer* dengan *osilator internal*
6. SRAM sebesar 512 *byte*
7. Memori *Flash* sebesar *8kb* dengan kemampuan *read while write*
8. Unit Interupsi *Internal* dan *External*
9. *Port* antarmuka SPI untuk men-*download* program ke *flash*
10. EEP*ROM* sebesar 512 *byte* yang dapat diprogram saat operasi
11. Antarmuka komparator *analog*
12. *Port* USART untuk komunikasi serial.

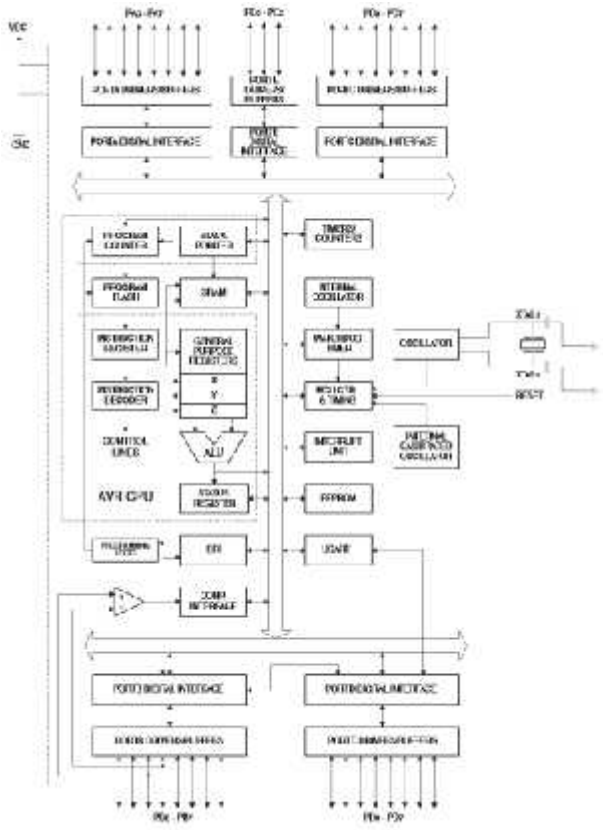
1. Konfigurasi ***Pin*** ATMega8535

Mikrokontroler AVR ATMega memiliki 40 *pin* dengan 32 *pin* diantaranya digunakan sebagai *port paralel*. Satu *port paralel* terdiri dari 8 *pin*, sehingga jumlah *port* pada mikrokontroler adalah 4 *port*, yaitu *port* A, *port* B, *port* C dan *port* D. Sebagai contoh adalah *port* A memiliki *pin* antara *port* A.0 sampai dengan *port* A.7, demikian selanjutnya untuk *port* B, *port* C, *port* D. Diagram *pin* mikrokontroler dapat dilihat pada gambar berikut:



Gambar 2.12 : Diagram Pin ATMega8535

(Sumber : Prasanto. Pengenalan Mikrokontroler. 2008 : 8 . diakses pada tanggal 01 Juni 2014)



Gambar 2.13 : Blok diagram ATMEGA8535

(Sumber : Prasanto. Pengenalan Mikrokontroler. 2008 : 8 . diakses pada tanggal 01 Juni 2014)

Berikut ini adalah tabel penjelasan mengenai pin yang terdapat pada mikrokontroler ATMega8535:

Tabel 2.5 : Tabel Penjelasan pin pada ATMega8535

Vcc	Tegangan suplai (*5 volt*)
GND	*Ground*
RESET	*Input reset level* rendah, pada *pin* ini selama lebih dari panjang pulsa *minimum* akan menghasilkan *reset* walaupun *clock* sedang berjalan. RST pada *pin* 9 merupakan *reset* dari AVR. Jika pada *pin* ini diberi masukan *low* selama minimal 2 *machine cycle* maka

	sistem akan di-*reset*
XTAL 1	*Input* penguat *osilator inverting* dan *input* pada rangkaian operasi *clock internal*
XTAL 2	*Output* dari penguat *osilator inverting*
Avcc	*Pin* tegangan suplai untuk *port* A dan ADC. *Pin* ini harus dihubungkan ke Vcc walaupun ADC tidak digunakan, maka *pin* ini harus dihubungkan ke Vcc melalui *low pass filter*
Aref	*pin* referensi tegangan *analog* untuk ADC
AGND	*pin* untuk *analog ground*. Hubungkan kaki ini ke GND, kecuali jika *board* memiliki *analog ground* yang terpisah

Berikut ini adalah penjelasan dari *pin* mikrokontroler ATMega8535 menurut *port*-nya masing-masing:

1. ***Port*** A

Pin33 sampai dengan *pin* 40 merupakan *pin* dari *port* A. Merupakan 8 *bit directional port* I/O. Setiap *pin*-nya dapat menyediakan *internal pull-up resistor* (dapat diatur per *bit*). *Output buffer port* A dapat memberi arus 20 mA dan dapat mengendalikan *display* LED secara langsung. *Data Direction Register port* A (DDRA) harus di-*setting* terlebih dahulu sebelum *port* A digunakan. *Bit-bit* DDRA diisi 0 jika ingin memfungsikan *pin-pin port* A yang disesuaikan sebagai *input*, atau diisi 1 jika sebagai *output*. Selain itu, *pin-pin* pada *port* A juga memiliki fungsi-fungsi alternatif khusus seperti yang dapat dilihat dalam tabel:

Tabel 2.6 Tabel Penjelasan pin pada port A

Pin	**Keterangan**
PA.7	ADC7 (ADC *Input Channel* 7)
PA.6	ADC6 (ADC *Input Channel* 6)

PA.5	ADC7 (ADC *Input Channel* 5)
PA.5	ADC4 (ADC *Input Channel* 4)
PA.3	ADC3 (ADC *Input Channel* 3)
PA.2	ADC2 (ADC *Input Channel* 2)
PA.1	ADC1 (ADC *Input Channel* 1)
PA.0	ADC0 (ADC *Input Channel* 0)

2. *Port* B

Pin 1 sampai dengan *pin* 8 merupakan *pin* dari *port* B. Merupakan 8 *bit directional port* I/O. Setiap *pin*-nya dapat menyediakan *internal pull-up resistor* (dapat diatur per *bit*). *Output buffer port* B dapat memberi arus 20 mA dan dapat mengendalikan *display* LED secara langsung. *Data Direction Register port* B (DDRB) harus di-*setting* terlebih dahulu sebelum *port* B digunakan. *Bit-bit* DDRB diisi 0 jika ingin memfungsikan *pin-pin port* B yang disesuaikan sebagai *input*, atau diisi 1 jika sebagai *output*. Selain itu, *pin-pin port* B juga memiliki fungsi-fungsi alternatif khusus seperti yang dapat dilihat dalam tabel:

Tabel 2.7 Tabel Penjelasan pin pada port B

Pin	**Keterangan**
PB.7	SCK (SPI *Bus Serial Clock*)
PB.6	VISO (SPI *Bus Master Input/Slave Output*)
PB.5	VOSI (SPI *Bus Master Output/Slave Input*)
PB.4	SS (SPI *Slave Select Input*)
PB.3	AIN1 (*Analog Comparator Negative Input*)OCC (*Timer/Counter0 Output Compare Match Output*)
PB.2	AIN0 (*Analog Comparator Positive Input*)INT2 (*External Interrupt2 Input*)

PB.1	T1 (*Timer/Counter1 External Counter Input*)
PB.0	T0 (*Timer/Counter0 External Counter Input*)XCK (JSART *External Clock Input/Output*)

3. ***Port*** C

Pin 22 sampai dengan *pin* 29 merupakan *pin* dari *port* C. *Port* C sendiri merupakan *port input* atau *output*. Setiap *pin*-nya dapat menyediakan *internal pull-up resistor* (dapat diatur per *bit*). *Output buffer port* C dapat memberi arus 20 mA dan dapat mengendalikan *display* LED secara langsung. *Data Direction Register port* C (DDRC) harus di-*setting* terlebih dahulu sebelum *port* C digunakan. *Bit-bit* DDRC diisi 0 jika ingin memfungsikan *pin-pin port* C yang disesuaikan sebagai *input*, atau diisi 1 jika sebagai *output*. Selain itu, *pin-pin port* D juga memiliki fungsi-fungsi alternatif khusus seperti yang dapat dilihat dalam tabel II.6:

Tabel 2.8 Tabel Penjelasan pin pada port C

Pin	**Keterangan**
PC.7	TOSC2 (*Timer Oscillator Pin* 2)
PC.6	TOSC1 (*Timer Oscillator Pin* 1)
PC.1	SDA (*Two-Wire Serial Bus Data Input/Output Line*)
PC.0	SCL (*Two-Wire Serial Bus Clock Line*)

4. ***Port*** D

Pin 14 sampai dengan *pin* 20 merupakan *pin* dari *port* D. Merupakan 8 *bit directional port* I/O. Setiap *pin*-nya dapat menyediakan *internal pull-up resistor* (dapat diatur per *bit*). *Output buffer port* D dapat memberi arus 20 mA dan dapat mengendalikan *display* LED secara langsung. *Data Direction Register port* D (DDRD) harus di-*setting* terlebih dahulu sebelum *port* D digunakan. *Bit-bit* DDRD diisi 0 jika ingin memfungsikan *pin-pin port* D yang

disesuaikan sebagai *input*, atau diisi 1 jika sebagai *output*. Selain itu, *pin-pin port* D juga memiliki fungsi-fungsi alternatif khusus seperti yang dapat dilihat dalam tabel:

Tabel 2.9 Tabel Penjelasan pin pada port D

Pin	**Keterangan**
PD.0	RDX (*UART input line*)
PD.1	TDX (*UART output line*)
PD.2	INT0 (*external interrupt 0 input*)
PD.3	INT1 (*external interrupt 1 input*)
PD.4	OC1B (*Timer/Counter1 output compareB match output*)
PD.5	OC1A (*Timer/Counter1 output compareA match output*)
PD.6	ICP (*Timer/Counter1 input capture pin*)
PD.7	OC2 (*Timer/Counter2 output compare match output*)

2.8.2.2 Kristal

Kristal lazimnya digunakan untuk rangkaian osilator yang menuntut stabilitas frekuensi yang tinggi dalam jangka waktu yang panjang. Alasan utamanya adalah karena perubahan nilai frekuensi kristal seiring dengan waktu, atau disebut juga dengan istilah faktor penuaan frekuensi (*frequency aging*), jauh lebih kecil dari pada osilator-osilator lain

Gambar 2.14 : Simbol Kristal (Kiri), Bentuk fisik Kristal (Kanan)
(Sumber : http://gazarobotik.wordpress.com/ diakses pada tanggal 28 Mei 2014)

2.8.3 Komponen Servo Kontroler Dan Motor Servo

Komponen yang penulis gunakan adalah servo kontroler merek SPC dan motor servo yang digunakan adalah merek towerpro MG996. Adapun spesifikasi masing masing komponen yaitu:

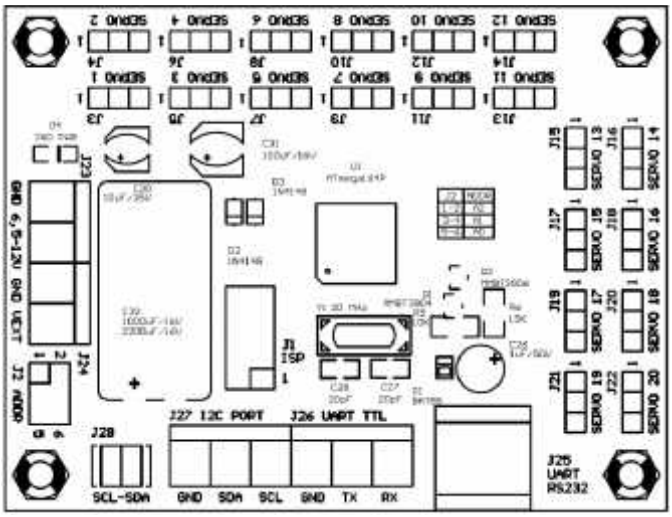
2.8.3.1 SPC Servo Motor *Kontroller*

Servo kontroler merupakan perangkat yang mempermudah dalam pengontrolan servo yang lebih dari satu, selain itu penggunaan servo kontroler dapat memper kecil penggunaan port pada mikrokontroler. Disini penulis menggunakan perangkat servo kontroler dengan merek SPC Servo Kontroller, buatan inovative elekronik, servo kontroler ini dapat berkomikikasi melalui I2C dan Serial, servo kontroler ini dilengkapi dengan 20 chanel *output* untuk mengontrol servo.

Spesifikasi SPC SERVO MOTOR CONTROLLER sebagai berikut:

1. Mampu mengendalikan hingga 20 motor servo secara serentak maupun sekuensial.
2. Mampu mengendalikan motor servo standar maupun kontinu.
3. Catu daya untuk SPC SERVO MOTOR CONTROLLER terpisah dengan catu daya untuk motor servo.
4. Catu daya untuk SPC SERVO MOTOR CONTROLLER dapat diperoleh dari sumber catu daya dengan tegangan 6,5 – 12 Volt.
5. Tiap modul SPC mampu mengendalikan 20 motor servo.
6. Resolusi pulsa kontrol servo sebesar 1 µs.
7. Dilengkapi dengan kemampuan servo *ramping.*
8. Dilengkapi dengan kemampuan membaca pulsa kontrol (posisi) servo.
9. Dilengkapi dengan kemampuan *Enable* dan *Disable* servo.
10. Dilengkapi dengan kemampuan menyimpan dan menjalankan sampai dengan maksimal 32 sekuen gerakan.
11. Dilengkapi dengan kemampuan menyimpan dan kembali ke posisi *home* (*default*).

12. Dilengkapi dengan kemampuan menyimpan sekuen gerakan yang melibatkan beberapa motor servo.
13. Tersedia antarmuka UART RS-232, UART TTL, dan I2C.
14. Jika menggunakan I2C, SPC SERVO MOTOR CONTROLLER dapat di-*cascade* hingga 8 modul.



Gambar 2.15 : Tata Letak SPC Servo Motor Kontroller

(Sumber : Manual Book SPC Servo Motor Kontroller)

2.8.3.2 Towerpro MG996

Servo towerpro mg996 merupakan jenis motor servo DC standar yang dapat beroperasi dari sudut -90 derajad sampai dengan 90 derajat, adapun spesifikasi motor servo yaitu:

1. Jenis servo *standard*
2. Beroperasi dengan catu daya :

   4,8V dengan torsi 8Kg

   5,8V deengan torsi 11Kg

Gambar 2.26 : Bentuk Fisik Servo Towerpro Mg996

(Sumber : http:// www.google.com/towerpromg996. diakses pada tanggal 01 Juni 2014)