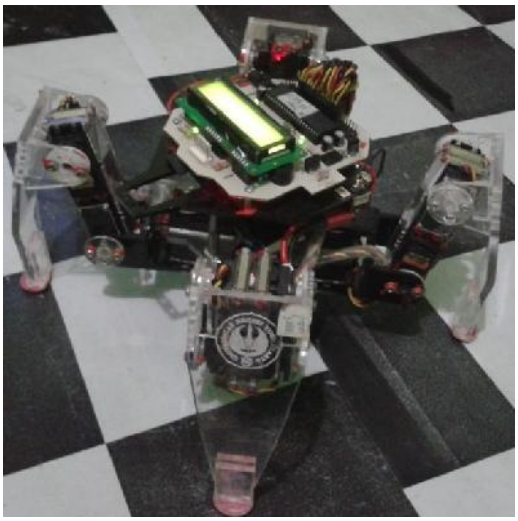
SERVO KONTROLER SEBAGAI PENGGERAK KAKI ROBOT DENGAN KOMUNIKASI SERIAL BERBASIS MIKROKONTROLER ATMEGA16

PROYEK AKHIR

Diajukan kepada Fakultas Teknik Universitas Negeri Yogyakarta untuk

Memenuhi Sebagian Persyaratan Guna Memperoleh

Gelar Ahli Madya Teknik



OLEH:

FEBRI CATUR PRAYOGO

NIM. 09507131017

PROGRAM STUDI TEKNIK ELEKTRONIKA

JURUSAN PENDIDIKAN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI YOGYAKARTA

2013

PERSETUJUAN

Proyek akhir yang berjudul "SERVO KONTROLER SEBAGAI PENGGERAK KAKI ROBOT DENGAN KOMUNIKASI SERIAL BERBASIS MIKROKONTROLER ATMEGA16 "ini telah disetujui oleh pembimbing untuk diujikan.

Yogyakarta, Mei 2013

Menyetujui,

Dosen Pembimbing Proyek Akhir

Dr. Eko Marpanaji M.T.
NIP. 19670608 199303 1 001

PENGESAHAN

Proyek akhir yang berjudul "SERVO KONTROLER SEBAGAI PENGGERAK KAKI ROBOT DENGAN KOMUNIKASI SERIAL BERBASIS MIKROKONTROLER ATMEGA16 " ini telah dipertahankan di depan Dewan Penguji pada tanggal..... dan dinyatakan lulus.

Dewan Penguji

Nama	Jabatan	Tanda Tangan	Tanggal
Dr. Eko Marpanaji	Ketua Penguji	.....	5/6 2013
Djoko Santoso, M.Pd	Sekretaris Penguji	.....	7/6 2013
Ahmad Fatchi M.Pd.	Penguji	.....	12/6 2013

Yogyakarta, Mei 2013
Fakultas Teknik
Universitas Negeri Yogyakarta
Dekan ,

Dr. Moch. Bruri Triyono
NIP. 19560216 198603 1 003

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Proyek Akhir ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar Ahli Madya atau gelar lainya di suatu perguruan tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis oleh orang lain, kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, Mei 2013

Yang Menyatakan,

Febri Catur Prayogo
NIM. 09507131017

PERSEMBAHAN

Rasa syukur yang mendalam Tugas Akhir ini saya persembahan kepada :

1. Allah SWT yang telah memberikan rahmatnya sehingga dapat menyelesaikan Tugas Akhir ini.
2. Ibu dan Bapak tercinta yang telah memberikan semangat dan doa serta fasilitas dalam menyelesaikan Tugas Akhir.
3. Kakak-kakak ku dan Keluarga Besar ku yang selalu mendukung Terimaksih segalanya.
4. Slamet Harimukti, Endri Sujatmiko, Masrur Abdul Nasir, Feri, dan rekan rekan *Study Club* HIMANIKA FT UNY yang telah memberikan seluruh bantuanya.
5. *Big Family* PH HIMANIKA 2011 yang senantiasa memberikan semangat.
6. Seluruh Teman-teman angkatan 2009 Jurusan Pendidikan Teknik Elektronika.

MOTTO

Sedikit bicara banyak berfikir dan bertindak

Mencoba sesuatu yang baru jika itu baik

Semakin pandai seseorang akan semakin banyak masalah.

Masalah yang ada untuk di hadapi bukan untuk di hindari

Berfikir sejenak, merenung masa lalu

adalah permulaan yang baik untuk bertindak

Berusahalah untuk menjadi yang terbaik, tetapi

jangan pernah merasa bahwa kita yang terbaik karna

kita adalah yang terpilih

1000 langkah berawal dari 1 langkah

Kebanyakan berfikir kapan *action*-nya?

PROYEK AKHIR

SERVO KONTROLER SEPAGAI PENGGERAK KAKI ROBOT DENGAN KOMUNIKASI SERIAL BERBASIS MIKROKONTROLER ATMEGA 16

Oleh: Febri Catur Prayogo
09507131017

ABSTRAK

Penulisan proyek akhir ini bertujuan membuat servo kontroler penggerak kaki robot yang dikomunikasikan secara serial terhadap IC utama.

Servo kontroler berbasis ATmega16 dengan komunikasi serial dirancang khusus untuk digunakan pada robot berkaki. Alat ini bekerja sesuai perintah dari chip IC utama yang telah diprogram. Data yang dikirimkan chip IC utama ke chip servo kontroler menggunakan komunikasi serial. Metode yang digunakan dalam membuat servo kontroler berbasis ATmega16 menggunakan metode eksperimental dengan tahap-tahap yaitu: (1) Identifikasi kebutuhan, (2) Analisis Kebutuhan, (3) Perancangan perangkat keras dan perangkat lunak, (4) Pembuatan alat, (5) Pengujian Alat dan (6) Pengoperasian Alat.

Berdasarkan hasil pengujian yang telah dilaksanakan diperoleh kesimpulan bahwa perangkat keras terdiri dari (1) UBEC sebagai penguat arus penyuplai servo, (2) Sistem minimum ATmega16 sebagai pemroses utama, (3) Sistem minimum ATmega16 sebagai pengendali servo, (4) Output motor servo sebagai motor yang di kendalikan. Perangkat lunak terdiri dari (1) Definisi *prosesor*, (2) Penyertaan fungsi, (3) Definisi *Port*, Deklarasi variabel dan (4) Fungsi Utama. Servo kontroler sebagai penggerak kaki robot dapat bekerja sesuai dengan prinsip kerja yang dirancang. Unjuk kerja dari alat ini dengan melihat kinerja penggeraan servo yang banyak.

Kata Kunci :Servo kontroler, Kaki Robot, Serial USART

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, yang telah melimpahkan rahmat, taufik dan karunia-Nya, sehingga dapat menyelesaikan Laporan Proyek Akhir yang berjudul "Servo Kontroler Sebagai Penggerak Kaki Robot Dengan Komunikasi Serial Berbasis Mikrokontroler Atmega 16". Tujuan dari penyusunan Proyek Akhir ini adalah sebagai syarat kelulusan pada program studi Teknik Elektronika D3 Universitas Negeri Yogyakarta.

Penulis menyadari bahwa tanpa bimbingan dan dorongan dari semua pihak, maka penulisan laporan Tugas Akhir ini tidak akan lancar. Oleh karena itu pada kesempatan ini, izinkanlah penulis menyampaikan ucapan terima kasih kepada:

1. Bapak dan Ibu yang selalu memberikan semangat dan doa-doanya.
2. Bapak Dr. Moch. Bruri Triyono, selaku Dekan Fakultas Teknik Universitas Negeri Yogyakarta.
3. Bapak Drs. Muhammad Munir, M.Pd selaku Ketua Jurusan Pendidikan Teknik Elektronika.
4. Bapak Drs. Djoko Santoso, M.Pd. selaku Koordinator Proyek Akhir Pendidikan Teknik Elektronika.
5. Bapak Dr. Eko Marpanaji, selaku Dosen Pembimbing Proyek Akhir.

6. Seluruh Dosen dan Karyawan di Jurusan Teknik Elektronika Fakultas Teknik Universitas Negeri Yogyakarta. Yang telah mendidik dan memotivasi selama kuliah di UNY.
7. Keluarga besar yang telah memberikan kasih sayang dan motivasi selama ini.
8. Slamet Harimukti terimakasih atas ide dan masukan – masukanya, semoga sukses selalu.
9. PH HIMANIKA 2011 yang telah membakar semangat untuk selesainya Proyek Akhir ini.
10. Teman-teman mahasiswa Teknik Elektronika UNY angkatan 2009
11. Semua pihak yang tidak dapat disebutkan satu per satu yang telah membantu dalam penyelesaian laporan ini.

Akhirnya disadari sepenuhnya bahwa dalam penyusunan Proyek Akhir ini masih jauh dari kesempurnaan sehingga saran, masukan, dan kritik sangat diperlukan demi kesempurnaan, semoga penyusunan Proyek Akhir ini dapat memberikan kontribusi bagi semua pihak.

Yogyakarta, Mei 2013

Febri Catur Prayogo

DAFTAR ISI

Halaman

BAB II PENDEKATAN PEMECAHAN MASALAH

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Seiring dengan kemajuan ilmu teknologi khususnya bidang kontrol membuat kebutuhan dunia akan kontroler sangat banyak. Dalam dunia kontrol terdapat banyak bentuk dari yang berupa mainan anak-anak sampai dengan produksi pada pabrik-pabrik. Kontroler banyak digunakan pada robot yang berada di pabrik maupun praktikum yang sering dilakukan oleh mahasiswa dan hobi dengan dunia kontrol.

Kontes Robot Indonesia (KRI), Kontes Robot Pemadam Api Indonesia(KRPAI), Kontes Robot Sepak Bola Indonesia(KRSBI), dan Kontes Robot Seni Indonesia (KRSI) adalah sebuah kegiatan perlombaan tingkat mahasiswa yang di selenggarakan oleh Dikti. Pada kontes Robot Pemadam Api Indonesia terdapat dua devisi yaitu: KRPAI devisi Beroda dan KRPAI devisi Berkaki. Pada KRPAI devisi Berkaki dan Beroda mempunyai lapangan yang sama dan tingkat kesulitan yang sama. Tujuan dari KRPAI adalah menyelesaikan misi memadamkan lilin di track yang sudah di sediakan dan kembali ketempat awal semula saat robot mulai dijalankan. Pada robot berkaki bagian kaki menggunakan servo yang cukup banyak sebagai penggerak.

Kebutuhan robot berkaki membutuhkan beberapa sensor dan perangkat pendukung kecerdasan robot. Berikut ini merupakan kebutuhan port yang

harus disediakan: 10 untuk sensor jarak, 6 untuk LCD, 2 untuk sensor api, 1 untuk indikator LED, 2 untuk kipas, 12 untuk servo, 1 untuk tombol, tiga untuk pengisian dan *sound activator*. Total penggunaan port adalah 41 port, jika menggunakan satu buah *chip* ATmega16 tidak mencukupi karena port I/O yang tersedia 32 port. Solusi permasalahan ini adalah menggunakan dua buah *chip* IC ATmega16 dan yang terpisah adalah servo yang dikendalikan dengan satu *chip* IC.

Cara untuk mengatasi masalah yang terjadi dirancanglah sebuah alat dengan teknologi yang sedang berkembang saat ini, sehingga dapat di amplikasikan untuk meringankan kinerja dari *Chip* IC yang menjadi otak pada robot. Sebuah IC mikrokontroler dapat digunakan sebagai kontrol pada servo yang banyak. Servo kontroler dapat dikatakan sebagai jembatan dari *Chip* IC utama. Rancangan alat ini menggunakan *Chip* IC yang digunaakan adalah ATmega16 yang banyak tersedia di pasaran. ATmega16 terdapat port *serial* yang digunakan untuk berkomunikasi secara *serial* .

Berdasarkan masalah yang terjadi maka diperlukan beberapa solusi, salah satunya membuat servo kontroler yang terpisah dari *Chip* IC utama. Proyek akhir ini akan mengkaji tetang bagaimana membuat servo kontroler terpisah dari *chip* utama menggunakan ATmega16. ATmega16 dapat difungsikan menjadi servo kontroler untuk menggerakan sampai dengan 24 servo. ATmega16 mempunyai port *serial* yang nantinya digunakan sebagai jalur untuk berkomunikasi dengan *Chip* IC utama

sebagai prosesor. Adanya alat ini diharapkan dapat membantu dan mempermudah sistem pergerakan kaki pada robot.

B. Identifikasi Masalah

Uraian latar belakang diatas dapat dibuat suatu identifikasi masalah sebagai berikut:

1. Kaki Robot membutuhkan servo kontroler untuk menggerakan motor servo.
2. Belum adanya servo kontroler yang menggunakan ATmega16 di pasaran.
3. Bagaimana memaksimalkan ATmega16 digunakan sebagai servo kontroler pada kaki robot.
4. Bagaimana mengkomunikasikan chip utama dengan chip servo kontroler untuk mengirimkan perintah ke chip servo kontroler.

C. Batasan Masalah

Berdasarkan latar belakang dan identifikasi masalah di atas, perlu adanya batasan masalah sehingga ruang lingkup masalah menjadi lebih jelas. Adapun batasan masalah yang diambil yaitu pembuatan servo kontroler dengan mikrokontroler ATmega16, yang menggunakan komunikasi *serial* USART untuk menghubungkan chip utama dengan chip servo kontroler. Memaksimalkan mikrokontroler ATmega16 karena mempunyai port yang cukup banyak untuk mengendalikan robot. Selain servo kontroler yang disebutkan tidak dibahas dalam proyek akhir ini

D. Rumusan Masalah

Melihat identifikasi yang ada dapat ditarik beberapa rumusan masalah, yaitu:

1. Bagaimana merancang Sistem *hardware* servo kontroler sebagai penggerak kaki robot dengan komunikasi *serial* USART berbasis Mikrokontroler ATmega16?
2. Bagaimana merancang Sistem *Software* servo kontroler sebagai penggerak kaki robot dengan komunikasi *serial* USART berbasis Mikrokontroler ATmega16?
3. Bagaimana unjuk kerja servo kontroler yang digunakan untuj mengerakan kaki robot dengan komunikasi *serial* USART berbasis Mikrokontroler ATmega16?

E. Tujuan

Adapun tujuan dari pembuatan servo kontroler sebagai kontrol penggerak pada robot berkaki berbasis mikrokontroler ATmega16 yaitu:

1. Merealisasikan rancangan sistem *hardware* servo kontroler sebagai kontrol penggerak kaki robot dengan komunikasi *serial* USART berbasis mikrokontroler ATmega16.
2. Merealisasikan rancangan sistem *Software* servo kontroler sebagai kontrol penggerak kaki robot dengan komunikasi *serial* USART berbasis mikrokontroler ATmega16.

3. Mengetahui unjuk kerja servo kontroler sebagai control penggerak kaki robot dengan komunikasi *serial* USART berbasis mikrokontroler ATmega16.

F. Manfaat

Pembuatan proyek akhir ini diharapkan dapat bermanfaat bagi mahasiswa, lembaga pendidikan, dan industri. Adapun manfaat yang diharapkan dari pembuatan tugas akhir ini antara lain:

1. Bagi mahasiswa
   a. Mengaplikasikan ilmu yang didapat selama di bangku kuliah dan menerapkan ilmunya secara nyata.
   b. Digunakan sebagai bahan referensi atau pembelajaran dan penambah wawasan tentang servo kontroler sebagai kontrol penggerak kaki robot berbasis mikrokontroler ATmega16 serta sebagai kajian untuk pengembangan selanjutnya.
2. Bagi jurusan
   a. Wujud dari perkembangan Ilmu Pengetahuan dan Teknologi (IPTEK).
   b. Parameter kualitas dan kuantitas lulusan mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta.
3. Bagi Dunia Industri
   a. Digunakan sebagai pengembangan produk elektronika yang dapat diaplikasikan sebagai servo kontroler robot berkaki.

b. Penerapan jembatan pengontrol servo pada robot berkaki.

G. Keaslian Gagasan

Gagasan ini asli dari gagasan bersama teman yang terinspirasi dari teman-teman di *study club* Robotika HIMANIKA dan banyaknya kelemahan yang terjadi pada robot berkaki jika menggunakan sebuah *Chip* IC yang menyebabkan tidak maksimalnya dalam perlombaan.

Penulis membuat servo kontroler sebagai kontrol pada robot berkaki dengan konsep yang berbeda dari yang sudah ada.

BAB II

PENDEKATAN PEMECAHAN MASALAH

A. Robot

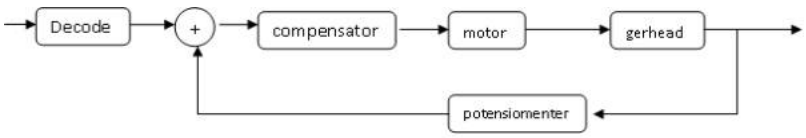
Robot adalah rangkaian elektronika yang terdapat program didalam untuk menggerakannya guna mempermudah kerja manusia. Robot dapat bergerak sesuai dengan program yang telah dimasukan. Cara mengetahui keadaan sekitar menggunakan sensor yang sesuai dengan kebutuhan. Contoh untuk melihat menggunakan sensor kamera.

Kontes Robot Pemadam Api Indonesia(KRPAI) adalah salah satu devisi kegiatan perlombaan robot tingkat mahasiswa yang di selenggarakan oleh Dikti. Pada kontes Robot Pemadam Api Indonesia terdapat dua devisi yaitu: KRPAI devisi Beroda dan KRPAI devisi Berkaki. Pada KRPAI devisi Berkaki dan Beroda mempunyai lapangan yang sama dan tingkat kesulitan yang sama. Tujuan dari KRPAI adalah menyelesaikan misi memadamkan lilin di track yang sudah di sediakan dan kembali ketempat awal semula saat robot mulai dijalankan.

Robot cerdas berkaki merupakan robot yang bergerak menggunakan kaki sebagai penggeraknya. Robot mempunyai kaki tergantung kebutuhan. Robot ini dibuat menggunakan empat kaki karena di sesuaikan dengan kondisi lapangan. Kondisi lapangan berbentuk lorong-lorong dan terdapat ruangan-ruangan. Tujuanya adalah memadamkan api dan kembali ketempat *start.*

B. **Servo**

Motor servo adalah sebuah motor dengan sistem umpan balik tertutup dimana posisi dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada didalam motor servo. Motor ini terdiri dari sebuah motor DC, serangkaian gear, potensiometer dan rangkaian kontrol. Potensiometer berfungsi untuk menentukan batas sudut dari putaran servo. Sedangkan sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor.



Gambar 1. Diagram Blok

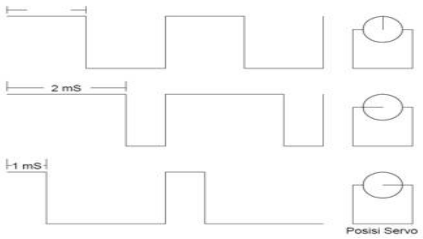
Karena motor DC servo merupakan alat untuk mengubah energi listrik menjadi energi mekanik, maka magnet permanen motor DC servolah yang mengubah energi listrik ke dalam energi mekanik melalui interaksi dari dua medan magnet. Salah satu medan dihasilkan oleh magnet permanen dan yang satunya dihasilkan oleh arus yang mengalir dalam kumparan motor. Resultan dari dua medan magnet tersebut menghasilkan *torsi* yang membangkitkan putaran motor tersebut. Saat motor berputar, arus pada kumparan motor menghasilkan *torsi* yang nilainya konstan.

Secara umum terdapat 2 jenis motor servo, yaitu motor servo standar dan motor servo *Continous*. Servo motor tipe standar hanya mampu berputar 180 derajat. Motor servo standar sering dipakai pada sistim robotika misalnya untuk membuat " Robot Arm" (Robot Lengan).

sedangkan Servo motor *continuous* dapat berputar sebesar 360 derajat. motor servo *Continous* sering dipakai untuk Mobil Robot. Pada badan servo tertulis tipe servo yang bersangkutan (akbarulhuda,2010).

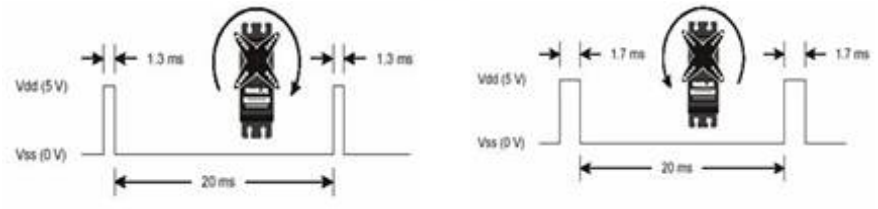
Gambar 2. Servo

Pengendalian gerakan motor servo dapat dilakukan dengan menggunakan metode PWM (*Pulse Width Modulation*). Teknik ini menggunakan system lebar pulsa untuk mengemudikan putaran motor. Sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor servo. Tampak pada gambar dengan pulsa 1.5mS pada periode selebar 20mS maka sudut dari sumbu motor akan berada pada posisi tengah. Semakin lebar pulsa OFF maka akan semakin besar gerakan sumbu ke arah jarum jam dan semakin kecil pulsa OFF maka akan semakin besar gerakan sumbu ke arah yang berlawanan dengan jarum jam.



Gambar 3. Pulsa

Untuk menggerakkan motor servo kekanan atau kekiri, tergantung dari nilai *delay* yang kita berikan. Untuk membuat servo pada posisi *center*, berikan pulsa 1.5ms. Untuk memutar servo ke kanan, berikan pulsa <=1.3ms, dan pulsa>= 1.7ms untuk berputar ke kiri dengan *dela*y 20ms, seperti ilustrasi berikut:



Gambar 4. Gerakan Servo

Tabel 1. Pergerakan Servo Standar

No	Sudut(^{0})	Delay(uS)	No	Sudut(^{0})	Delay(uS)
1	-90	600	11	10	1600
2	-80	700	12	20	1700
3	-70	800	13	30	1800
4	-60	900	14	40	1900
5	-50	1000	15	50	2000
6	-40	1100	16	60	2100
7	-30	1200	17	70	2200
8	-20	1300	18	80	2300
9	-10	1400	19	90	2400
10	0	1500	20		

Tabel 1. menjelaskan tetang pergerakan servo dengan memberikan delay pada port data servo. Servo digerakan dengan jarak delay 10uS menghasilkan 1^0 maka untuk jarak delay 100uS dapat menggerakan sebesar 10^0 .

C. Mikrokontroler AVR

1. Arsitektur Atmega 16

Mikrokontroller dapat dianalogikan seperti sebuah sistem komputer yang dikemas dalam sebuah chip. Dalam sebuah chip mikrokontroller sudah terdapat kebutuhan minimal agar mikroprosessor dapat bekerja, yaitu meliputi mikroprosessor , ROM, RAM, I/O, dan Clock seperti yang dimiliki sebuah Personal Komputer (PC).

Karena Ukurannya yang relative kecil membuat mikrokontroller menjadi lebih fleksibel dan praktis digunakan terutama pada sistem yang tidak terlalu kompleks dan tidak membutuhkan beban komputasi yang tinggi.

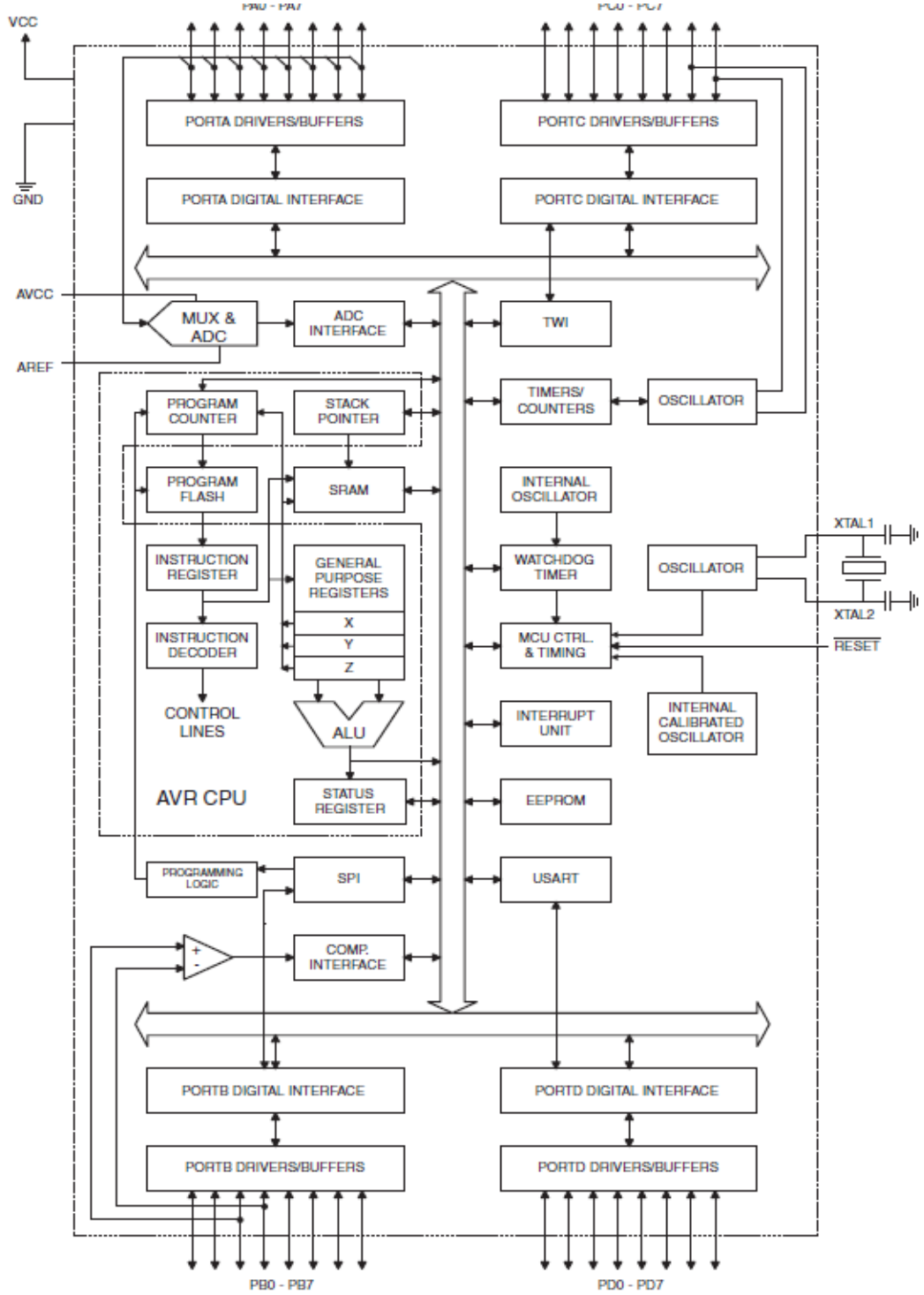
2. Fitur Atmega 16

Fitur – fitur yang dimiliki ATMEGA16 sebagai berikut:

a. Mikrokontroler AVR 8 bit yang memiliki kemampuan tinggi, dengan daya rendah.
b. Arsitektur RISC dengan *throughput* mencapai 16 MIPS pada frekuensi 16MHZ.
c. Memiliki kapasitas *Flash* memori 16Kbyte, EEPROM 512 *Byte* dan SRAM 1 KByte.
d. Saluran I/O sebanyak 32 buah, yaitu Port A, Port B, Port C, dan Port D.
e. CPU yang terdiri atas 32 buah register .

f. Unit interupsi internal dan eksternal.

g. Port USART untuk komunikasi *serial*.



Gambar 5. Blok Diagram AVR ATMega16

h. Fitur *peripheral*

- Tiga buah *Timer/Counter* dengan kemampuan pembandingan.
  - 2 (dua) buah *Timer/Counter* 8 bit dengan *Prescaler* terpisah dan *Mode Compare.*
  - 1 (satu) buah Timer/Counter 16 bit dengan *prescaler* terpisah, Mode *Compare*, dan *Mode Capture.*
- *Real Time Counter* dengan *Oscilator* tersendiri.
- 4 *channel* PMW
- 8 *channel*, 10-bit ADC
  - 8 *Single-ended Channel*
  - 7 *Diferential Channel* hanya pada kemasan TQFP
  - 2 *Diferential Channel* dengan Progammable Gain 1x, 10x, atau 200x.
- *Byte-oriental Two-wire Serial Interface*
- *Progammable* Serial USART
- Antarmuka SPI
- *Watchdog Timer* dengan *oscillator internal.*
- *On-chip Analog Comparator.*

3. Konfigurasi Pin Atmega 16

Gambar 6. Konfigurasi Pin ATMEGA 16

Konfigurasi pin ATmega16 dengan kemasan 40 pin DIP (*Dual In-line Package*) dapat dilihat pada gambar 6. Dari gambar diatas dapat dijelaskan fungsi dari masing-masing pin ATmega16 sebagai berikut:

VCC merupakan pin yang berfungsi sebagai masukan catu daya.

GND merupakan pin Ground.

Port B :(PB7-PB0) port B merupakan Port I/O 8-bit *bi-direktional* (dua arah) dengan resistor *pull-up* internal secara individual. Selain sebagai Port I/O, Port B juga memiliki fungsi *alternative*.

Port D : (PD7-PD0) port D merupakan Port I/O 8-bit *bi-direktional* (dua arah) dengan resistor *pull-up* internal secara individual. Selain sebagai Port I/O, Port D juga memiliki fungsi *alternative*.

Port A: **(PA7-PA0)** sebagai masukan analog untuk ADC. Port A juga bisa digunakan sebagai 8-bit I/O, Port jika A/D *Converter*

tidak digunakan dan masing–masing pin I/O memiliki internal *pull-up.* Pemilihan portA sebagai input analog atau sebagai *Analog to Digital Converter* (ADC) bisa dilakukan melalui pemrograman.

Port C : (PD7-PD0) port D merupakan Port I/O 8-bit *bi-direktional* (dua arah) dengan resistor *pull-up* internal secara individual. Selain sebagai Port I/O, Port D juga memiliki fungsi alternative.

RESET: merupakan input reset yang bekerja pada *level* rendah (*active low*) selama minimal 1,5us.

XTAL1:Input ke penguat *inverting oscillator* dan input ke internal *clock.*

XTAL2 Output dari penguat *inverting oscillator*.

AVCC merupakan catu daya yang digunakan sebagai masukan analog ADC yang terhubung ke Port A.

AREF merupakan tegangan referensi analog untuk ADC.

4. I/O PORT

Semua Port I/O keluarga AVR bersifat *bi-diectional* (dua arah) pada saat berfungsi sebagai port I/O digital. Bahakan masing–masing pin dapat dikonfigurasi tanpa mempengaruhi pin lainnya.

Pengaturan port I/O baik sebagai input atau output otomatis akan akan diikuti dengan pengaturan resistor *pull-up* internal. Meskipun demikian internal *pull-up* resistor bisa di non-aktifkan melalui bit PUD

SFIOR (*Special Function I/O* Register). Jika bit PUD diset “1” maka resistor *pull-up* internal di non-aktifkan.

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Setiap Port I/O terdiri dari tiga register I/O yaitu DDRX, PORTX, dan PINX:

a. Data Register (PORTX)

Port A data register

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B data register

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C data register

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D data register

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Register Portx digunakan untuk 2 keperluan yaitu untuk jalur output atau untuk mengaktifkan resistor *pull-up*.

1) Portx berfungsi sebagai output jika DDRx = 1 maka:

   Portxn = 1 maka pin Pxn akan berlogika *high*.

   Portxn = 0 maka pin Pxn akan berlogika *low*.

2) Portx berfungsi untuk mengaktifkan resistor *pull-up* jika DDRx = 0 maka:

   Portxn = 1 maka pin Pxn sebagai pin input dengan resistor *pull-up*.

   Portxn = 0 maka pin Pxn sebagai output tanpa resistor *pull-up*.

b. DDRX (Data *Direction* Register)

Register DDRx digunakan untuk memilih arah pin. Jika DDRx = 1 maka Pxn sebagai pin output, Jika DDRx = 0 maka Pxn sebagai input.

Port A Data *Direction* Register

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Data *Direction* Register

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Data *Direction* Register

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Data *Direction* Register

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

c. PINX (Port Input Pin *Address*)

Digunakan untuk menyimpan data yang terbaca dari port I/O pada saat dikonfigurasi sebagai input.

Port A Input Pins *Address*

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port B Input Pins *Address*

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port C Input Pins *Address*

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port D Input Pins *Address*

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Setiap Port I/O bersifat *bi-directional* atau dua arah dan masing–masing Port juga memiliki fungsi tambahan *(Alternate Functions)*

Tabel 2. Fungsi Tambahan (*Alternate Function*) PORTB

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

Tabel 3. Fungsi Tambahan (*Alternate Functions*) PORTD

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP1 (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

Tabel 4. Fungsi Tambahan (*Alternate Functions*) PORTA

Port Pin	Alternate Function
PA7	ADC7 (ADC input channel 7)
PA6	ADC6 (ADC input channel 6)
PA5	ADC5 (ADC input channel 5)
PA4	ADC4 (ADC input channel 4)
PA3	ADC3 (ADC input channel 3)
PA2	ADC2 (ADC input channel 2)
PA1	ADC1 (ADC input channel 1)
PA0	ADC0 (ADC input channel 0)

Tabel 5. Fungsi Tambahan (*Alternate Functions*) PORTC

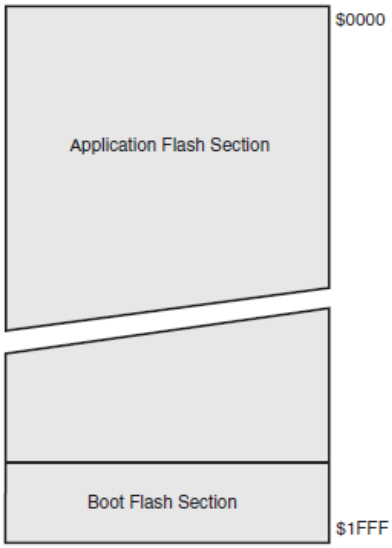
Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC5	TDI (JTAG Test Data In)
PC4	TDO (JTAG Test Data Out)
PC3	TMS (JTAG Test Mode Select)
PC2	TCK (JTAG Test Clock)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC0	SCL (Two-wire Serial Bus Clock Line)

5. Peta Memori

a. Memori Flash

ATmega16 memiliki On-*Chip In-System Reprogrammable* Flash *Memory* untuk menyimpan program. Alasan keamanan, program *memory* dibagi menjadi dua bagian yaitu Boot Flash Section dan Application Flash Section. Boot Flash *Section* digunakan untuk menyimpan program Boot *Loader*, yaitu program yang harus dijalankan pada saat AVR reset atau pertamakali diaktifkan.

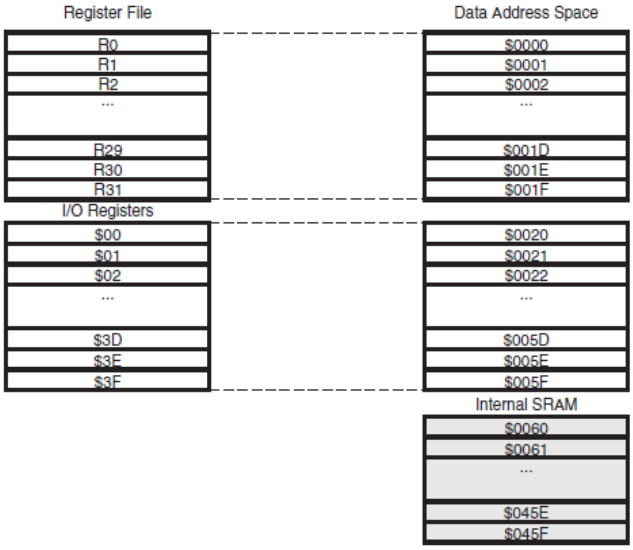
Application Flash *Section* digunakan untuk menyimpan program aplikasi yang dibuat *user*. AVR tidak dapat menjalankan program aplikasi ini sebelum menjalankan program Boot *Loader*.



Gambar 7. Peta Program *memory*

b. Memori Data

Gambar berikut menunjukkan peta memori SRAM pada ATmega16. Terdapat 1120 lokasi address data memori. 96 lokasi *address* digunakan untuk Register File dan I/O *Memory*, selanjutnya 1024 lokasi address lainnya digunakan untuk internal data SRAM. Register File terdiri dari 32 *General Purpose* Register (GPR), I/O register terdiri dari 64 register .



Gambar 8. Peta Data Memori

Dalam organisasi memori AVR, 32 register serbaguna (GPR) menempati *space* data pada alamat terbawah, yaitu $00 sampai $30. Sedangkan register -register khusus untuk penanganan I/O dan control terhadap mikrokontroler, menempati 64 alamat berikutnya merupakan register I/O khusus digunakan untuk melakukan pengaturan fungsi terhadap berbagai perihal mikrokontroler seperti control register, timer/counter, fungsi-fungsi

I/O, ADC, USART, SPI ,EEPROM dan sebagainya. Alamat berikutnya digunakan untuk SRAM *(Static Random Access Memory*) 1 KB.

c. Memory EEPROM

ATMega16 memiliki memori EEPROM (*Electrically Erasable Programable Read Only Memory*) sebesar 512 *byte* yang terpisah dari memori program maupun memori data. Memori EEPROM bisa digunakan untuk menyimpan data yang dapat bertahan atau tersimpan walaupun mikrokontroler tanpa tegangan catu daya atau tahan terhadap gangguan catu daya. Memori EEPROM ini hanya bisa diakses dengan menggunakan register I/O

D. Komunikasi

Secara harafiah, komunikasi berasal dari Bahasa Latin: COMMUNIS yang berarti keadaan yang biasa, membagi, sama atau milik bersama. Dengan kata lain, komunikasi adalah sutu proses di dalam upaya membangun saling pengertian. Berikut ini merupakan definisi komunikasi:.

1. komunikasi adalah kegiatan perilaku atau kegiatan penyampaian pesan atau informasi tentang pikiran atau perasaan (Roben.J.G)
2. Komunikasi adalah suatu proses pertukaran informasi antar individu melalui suatu sistem yang biasa (lazim), baik dengan simbol-simbol, sinyak-sinyal, maupun perilaku atau tindakan(Himstreet & Baty)

3. Komunikasi adalah suatu proses pengiriman dan penerimaan pesan(Bovee).

Komunikasi didalam proyek akhir ini adalah hubungan antara duabuah benda mikrokontroler untuk menyamakan presepsi. Hubungan antara kedua mikrokontroler menggunakan port yang telah tersedia pada mikrokontroler.(indah, 2010)

E. Komunikasi Data Serial

Komunikasi data adalah proses pengiriman dan penerimaan data/informasi dari dua atau lebih device (alat,seperti komputer/laptop/printer/dan alat komunikasi lain)yang terhubung dalam sebuah jaringan. Cara mengkomunikasikan data membututuhkan dua atau lebih perangkat yang akan berkomunikasi. Proyek akhir ini perangakat yang digunakan adalah dua buah mikrokontroler ATmega16. (agung gunawan, 2013)

Komunikasi serial adalah komunikasi yang pengiriman datanya per-bit secara berurutan dan bergantian. Komunikasi ini mempunyai suatu kelebihan yaitu hanya membutuhkan satu jalur dan kabel yang sedikit dibandingkan dengan komunikasi paralel. Pada prinsipnya komunikasi serial merupakan komunikasi dimana pengiriman data dilakukan per bit sehingga lebih lambat dibandingkan komunikasi parallel, atau dengan kata lain komunikasi serial merupakan salah satu metode komunikasi data di mana hanya satu bit data yang dikirimkan melalui seuntai kabel pada suatu waktu tertentu

Komunikasi serial ada dua macam, *asynchronous serial* dan *synchronous serial*. *Synchronous serial* adalah komunikasi dimana hanya ada satu pihak (pengirim atau penerima) yang menghasilkan clock dan mengirimkan clock tersebut bersama-sama dengan data. *Asynchronous serial* adalah komunikasi dimana kedua pihak (pengirim dan penerima) masing-masing menghasilkan clock namun hanya data yang ditransmisikan, tanpa clock(Arif Zakarya, 2012).

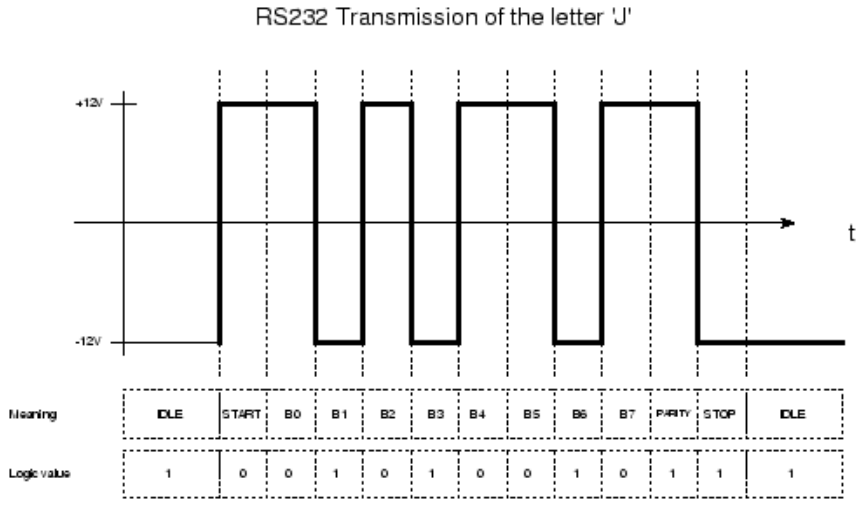
RS-232 adalah standar komunikasi serial yang didefinisikan sebagai antarmuka antara perangkat terminal data (*data terminal equipment* atau **DTE**) dan perangkat komunikasi data (*data communications equipment* atau **DCE**) menggunakan pertukaran data biner secara serial. Komunikasi RS-232 diperkenalkan pada 1962 dan pada tahun 1997, *Electronic Industries Association* mempublikasikan tiga modifikasi pada standar RS-232 dan menamainya menjadi EIA-232. Pada saat itu RS-232 lahir karena muncul dari ide-ide pada sebuah komite (*Electronic Industries Association-EIA)* yang mengembangkan sebuah interface untuk pertukaran data digital antara komputer mainframe yang sebagai pusatnya dengan komputer lain, tetapi perangkat ini dihubungkan dengan jaringan telepon sehingga dibutuhkan modem untuk menerjemahkan sinyal tersebut (Akhmad Zainuri, ST, 2011)

Antarmuka kanal serial menawarkan berapa kelebihan dibandingkan secara paralel, antara lain:

1. Kabel untuk komunikasi serial bisa lebih panjang dibandingkan dengan paralel; data-data dalam komunikasi serial dikirim-kan untuk logika ’1′ sebagai tegangan -3 s/d -25 volt dan untuk logika ’0′ sebagai tegangan +3 s/d +25 volt, dengan demikian tegangan dalam komunikasi serial memiliki ayunan tegangan maksimum 50 volt, sedangkan pada komunikasi paralel hanya 5 volt.

2. Jumlah kabel serial lebih sedikit; bisa menghubungkan dua perangkat dengan hanya 3 kabel, yaitu TXD (saluran kirim), RXD(saluran terima) dan Ground.

3. Untuk teknologi embedded system, banyak mikrokontroler yang dilengkapi dengan komunikasi serial (baik seri RISC maupun CISC) atau Serial Communication Interface (SCI); dengan adanya SCI yang terpadu pada 1C mikrokontroler akan mengurangi jumlah pin keluaran, sehingga hanya dibutuhkan 2 pin utama TxD dan RxD (di luar acuan ground).



Gambar 9. Diagram Sinyal RS 232

F. Serial USART

USART (*Universal Synchronous Asynchronous serial Receiver and Transmitter*) merupakan protokol komunikasi *serial* yang terdapat pada mikrokontroler AVR. Memanfaatkan fitur ini kita dapat berhubungan dengan "dunia luar". USART bisa menghubungkan mikrokontroler dengan mikrokontroler, PC, Handphone, *GPS* atau bahkan *modem*, dan banyak lagi peralatan yang dapat dihubungkan dengan mikrokontroler dengan menggunakan fasilitas USART .

Komunikasi dengan menggunakan USART dapat dilakukan dengan dua cara yaitu dengan *mode* sinkron dimana pengirim data mengeluarkan pulsa/*clock* untuk sinkronisasi data, dan yang kedua dengan mode asinkron, dimana pengirim data tidak mengeluarkan pulsa/*clock*, tetapi untuk proses sinkronisasi memerlukan inisialisasi, agar data yang diterima sama dengan data yang dikirimkan. Pada proses inisialisasi ini setiap perangkat yang terhubung harus memiliki *baud rate* (laju data) yang sama. Pada mikrokontroler AVR untuk mengaktifkan dan mengeset komunikasi USART dilakukan dengan cara mengaktifkan register 2 yang digunakan untuk komunikasi USART . Register *2* yang dipakai antara lain:

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

UDR: merupakan register 8 bit yang terdiri dari 2 buah dengan alamat yang sama, yang digunakan sebagai tempat untuk menyimpan data yang akan dikirimkan (TXB) atau tempat data diterima (RXB) sebelum data tersebut dibaca.

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

UCSRA: merupakan register 8 *bit* yang digunakan untuk mengendalikan *mode* komunikasi USART dan untuk membaca status yang sedang terjadi pada USART .

Bit **RXC** [status]—> akan "1″ bila ada data di UDR (RXB) yang belum terbaca. Dapat digunakan untuk sumber interupsi, dengan mengeset RXCIE

Bit **TXC** [status]—> akan "1″ bila ada data di UDR (TXB) yang sudah dikirimkan. Dapat digunakan untuk sumber interupsi, dengan mengeset TXCIE

Bit **UDRE** [status]—> akan "1″ bila UDR siap untuk menerima data baru.

Bit **U2X** [kendali]—> diisi "1″ bila kecepatan transmisi data ingin dinaikkan 2kali.

Bit **MPCM** [kendali]—>digunakan bila ingin menggunakan komunikasi multiprosesor.

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

UCSRB: merupakan register 8 bit yang digunakan untuk mengendalikan mode komunikasi USART dan untuk membaca status yang sedang terjadi pada USART .

Bit **RXCIE** [kendali]—>digunakan untuk mengaktifkan interupsi yang bersumber dari RXC.

Bit **TXCIE** [kendali]—>digunakan untuk mengaktifkan interupsi yang bersumber dari TXC.

Bit **UDRIE** [kendali]—>digunakan untuk mengaktifkan interupsi yang bersumber dari UDRE.

Bit **RXEN** [kendali]—>digunakan untuk mengaktifkan *receiver*.

Bit **TXEN** [kendali]—>digunakan untuk mengaktifkan *transmitter*.

Bit **UCSZ2** [kendali]—>digunakan untuk menentukan panjang data yang dikirim dalam sekali. Digunakan bersama2 dengan UCSZ1,UCSZ0 pada UCSRC.

Bit **RXB8** [status]—>digunakan sebagai penampung data ke-9 pada penerimaan data dengan 9 bit

Bit **TXB8** [status]—>digunakan sebagai penampung data ke-9 pada transmisi data dengan 9 bit

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

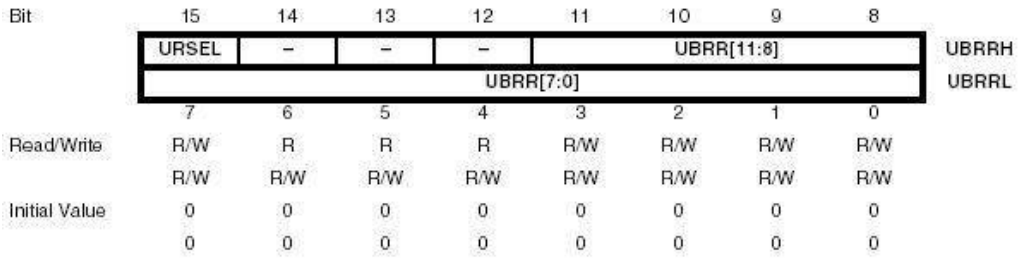
UCSC: merupakan register 8 bit yang digunakan untuk mengendalikan *mode* komunikasi USART dan untuk membaca status yang sedang terjadi pada USART .

Bit **URSEL** [kendali]—> digunakan untuk memilih register pada UCSRC dan UBRRH. Dua register ini memiliki alamat yang sama, sehingga untuk proses penulisan memerlukan bantuan URSEL. Bila URSEL=1, maka register yang diisi adalah UCSRC, sedangkan bila URSEL=0, register yang diisi adalah UBRRH. Tidak semua mikrokontroler AVR memiliki URSEL, karena ada yang memiliki register UBRRH dan UCSRC yang beda alamat

Bit **UMSEL** [kendali]—> bila “1″, maka *mode* yang dipilih adalah asinkron, “0″=sinkron

Bit **USBS** [kendali]—>bila “1″, maka *stop* bit berjumlah 2 bit

Bit **UCSZ1,UCSZ0** [kendali]—> bersama 2 UCSZ2 digunakan untuk menentukan jumlah bit yang akan dikirimkan dalam sekali pengiriman data.



UBRRL,H: merupakan register 16 bit yang digunakan untuk mengatur laju data (*baud rate*) pada saat *mode* komunikasi asinkron (polong, 2008).

G. UBEC (*Universal Battery Elimination Circuit*)

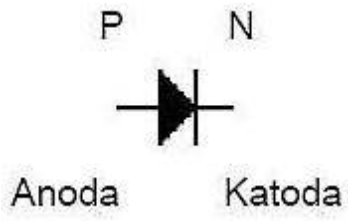
UBEC adalah *regulator switching*. UBEC (*Universal Battery Elimination Circuit)* adalah rangkaian elektronik yang mengambil daya dari *battery* pack atau sumber DC lainnya, dan menurunkannya ke *level* tegangan 5V atau 6V serta menaikan arus 3-5A. Kebutuhan mencatu motor servo atau rangkaian lain yang bekerja pada tingkat tegangan 5V–6V Tegangan input maksimum tergantung pada spesifikasi UBEC.

Gambar 10. UBEC

UBEC biasanya digunakan pada aplikasi yang memerlukan arus lebih tinggi, dan divais mampu men-*deliver* daya dengan efisiensi hingga 92%. UBEC yang digunakan mampu menyupai arus sebesar 5 Ampere maksimal(Christianto Tjahyadi, 2012).

H. Dioda (Penyearah Arus)

Dioda adalah komponen elektronika yang hanya memperbolehkan arus listrik mengalir dalam satu arah sehingga dioda biasa disebut juga sebagai "Penyearah". Dioda terbuat dari bahan semikonduktor jenis *Silicon* dan *germanium* Simbol dioda dalam rangkaian elektronika diperlihatkan pada gambar berikut.



Gambar 11. Simbol Dioda

Dioda terbuat dari penggabungan dua tipe semikonduktor yaitu tipe P (*Positive*) dan tipe N (*Negative*), kaki dioda yang terhubung pada semikonduktor tipe P dinamakan "*Anode*"sedangkan yang terhubung pada semikonduktor tipe N disebut "*Katoda*". Pada bentuk aslinya pada dioda terdapat tanda cincin yang melingkar pada salah satusisinya, ini digunakan untuk menandakan bahwa pada sisi yang terdapat cincin tersebut merupakan kaki *Katoda*.

Gambar 12. Bentuk Dioda

Arus listrik akan sangat mudah mengalir dari anoda ke katoda hal ini disebut sebagai " *Forward-Bias*" tetapi jika sebaliknya yakni dari katoda keanoda, arus listrik akan tertahan atau tersumbat hal ini dinamakan sebagai " *Reverse-Bias*".*)*

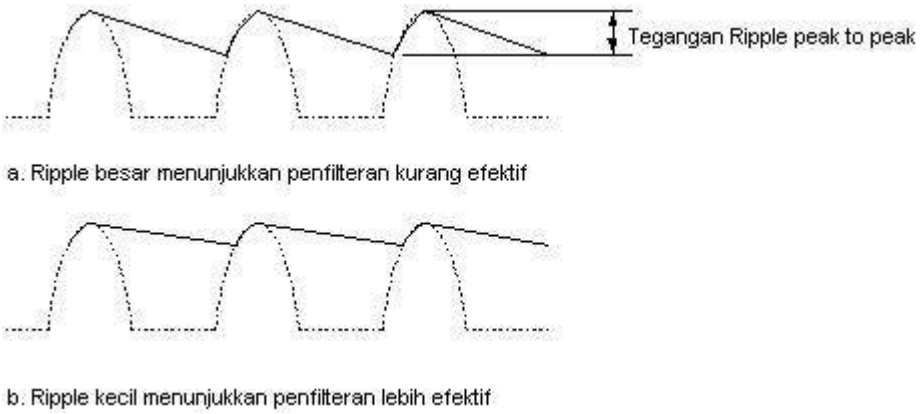
I. Kapasitor

Kapasitor adalah suatu komponen elektronika yang terdiri dari dua buah penghantar yang diberi sekat. Sekat tersebut terbuat dari bahan isolasi (mika, kertas, keramik, udara, dan sebagainya).

Ditemukan pada tahun 1746 di Universitas Leyden, oleh ilmuwan Pieter Van Musschenbroek dengan cara mencoba menyimpan sejumlah

besar muatan listrik. Hasilnya adalah suatu alat yang secara luas dikenal sebagai botol Leyden. (Unggul Prasetya,2010)

Fungsi kapasitor pada rangkaian catu daya adalah berfungsi sebagai *filter* pada rangkaian power supply, yang dimaksud disini adalah kapasitor sebagai *ripple filter*, disini sifat dasar kapasitor yaitu dapat menyimpan muatan listrik yang berfungsi untuk mengecilkan tegangan *ripple*.



Gambar 13. Tegangan Riple

J. Beterai

Baterai adalah peralatan yang mengkonversikan energi kimia menjadi energi listrik. Baterai digunakan untuk menyimpan tenaga listrik arus searah (DC). Sejarah baterai dimulai pada tahun 1800, ketika pertama kali Allexsandro Volta, seorang profesor fisika dari Universitas Pavia, Italia membuat baterai pertamanya yang terbuat dari lembaran-lembaran logam dan papan karton atau kertas yang direndam dalam air garam. Dan sejak saat itu, para ilmuwan berlomba menyempurnakan ide sederhana Volta hingga menjadi seperti baterai yang kita jumpai saat ini. Pada dasarnya baterai terdiri dari tiga komponen utama, yaitu:

- Elektroda Positif (*Katoda*)
- Elektroda Negatif (*Anoda*)
- Bahan Elektrolit (Cair/ Padat)

Elektroda berfungsi sebagai tempat mengalirnya elektron. Sedangkan bahan elektrolit berfungsi sebagai *medium* aliran arus ionik dirangkaian dalam. Elektron dialirkan melalui rangkaian luar dari salah satu elektroda ke elektroda lainnya. Anoda berfungsi untuk melepaskan elektron sedangkan berfungsi untuk menerima arus elektron. Untuk dapat mengalirkan dan menerima elektron, elektroda harus memiliki konduktivitas elektronik tinggi dan konduktivitas ionik yang rendah. Bahan yang digunakan sebagai elektroda haruslah bahan yang stabil dan tidak bereaksi dengan elektrolit (Lis Lestari, 2012).

Berdasarkan bahan elektrolitnya, baterai terbagi menjadi 2, yaitu:

1. Baterai basah (Accu/Aki)
2. Baterai kering (Baterai padat)

K. Perangkat Lunak *(Software)*

Bahasa C adalah bahasa mesin tingkat tinggi. Dimana dapat dengan mudah untuk melakukan pemrograman terhadap mikrokontroler. Dengan instruksi-instruksi yang mudah dipahami dan mudah diakses. Secara umum pemrograman mikrokontroler terdiri atas empat blok, yang stiap blok tersebut mempunyai definisi tersendiri yaitu:

1. Header.

2. Deklarasi kostanta global.
3. Fungsi dan atau prosedur (biasa dibawah program utama).
4. Program utama.

Secara umum pemrograman C paling sederhana dapat dilakukan dengan hanya menuliskan program utamanya saja. Beberapa peraturan yang ada dalam bahasa C adalah:

1. Header

Header berisi include file (.hex), yaitu *library* (pustaka) yang akan digunakan dalam pemrograman. Perhatikan contoh dibawah ini:

```
#include <mega16.h>
#include <delay.h>
#incude <stdio.h>
```

2. Tipe Data

Berikut ini adalah tabel tipe-tipe variabel data yang dapat digunakan di *compiler Code Vision AVR*:

Tabel 6. Type Data

Type	Size (Bits)	Range
Bit	1	0,1
Char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127
Int	16	-32768 to 32767
short int	16	-23768 to32767
unsigned int	16	0 to 65535
signed int	16	-32768 to 32767
long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
signed long int	32	-2147483648 to 2147483647
Float	32	±1.175e-38 to ±3.402e38
Double	32	±1.175e-38 to ±3.402e38

3. Konstanta

Penulisan konstanta adalah sebagai berikut:

a. *Integer* atau *long integer* dapat ditulis dengan format decimal (contoh 1234), biner dengan awalan 0b (contoh 0b101001), heksadesimal dengan awalan 0x (contoh 0xff) atau octal dengan awalan 0 (contoh 0777).
b. *Unsigned integer* ditulis dengan diakhiri U (contoh 10000U).
c. Long integer ditulis dengan diakhiri L (contoh 99L).
d. *Unsigned long integer* ditulis dengan diakhiri UL (contoh 99UL).
e. *Floating* poin ditulis dengan diakhiri F (contoh 1.234F).

Karakter konstanta harus ditulis dalam tanda kutip (contoh ‘a’), sedangkan konstanta string harus dalam tanda kutip dua (contoh “Saya Belajar C”).

4. Label, Variabel, Fungsi

Identifikasi label, variabel dan fungsi dapat berupa huruf (A…Z, a…z) Dan angka (0…9), juga karakter *underscore* (_). Meskipun begitu identikasi hanya bias dimulai dengan huruf atau karakter *underscore*. Yang lebih penting lagi, identifikasi ini *Case is significant*, yaitu huruf besar dan kecil berbeda. Misal Variable1 tidak sama dengan variabel1. Identifikasi bisa memuat sebanyak 32 karakter.

5. Komentar

Komentar diawali dengan tanda ’/*’ dan diakhiri dengan ‘*/’.

Perhatikan contoh dibawah:

```
/* ini komentar */
```

6. Reserved *Keywords*

Berikut ini adalah daftar kata baku yang tidak bias dipakai (*reserfed keywords*) untuk label, identifikasa atau variable:

Break	flash	signed	do	int	typedef
Bib	float	sizeof	double	interrupt	union
Case	for	sfrb	eeprom	long	unsigned
Char	funcused	sfrw	else	register	void
const	goto	static	enum	return	volatile
continue	if	struct	extern	short	while
default	inline	witch			

7. Operator

Suatu intruksi pasti mengandung operator dan operand. Operand adalah variabel atau konstanta yang merupakan bagian pernyataan sedangkan operator adalah suatu simbol yang menyatakan operasi mana yang akan dilakukun oleh operand tersebut. Contoh:

c = a + b ;

Ada tiga operand (a, b dan c) dan dua operator (= dan +). Operator dalam C dibagi menjadi 3

kelompok, yaitu:

a. *unary* operator yang beroperasi pada satu operand, misal: -n
b. *binary* operator yang beroperasi pada dua operand, misal: a-n
c. *ternary* operator memerlukan tiga / lebih operand, missal: a = (b*c) + d

8. Aritmatika

Tabel 7. Simbol dan Aritmatika

simbol	Contoh	aritmatika
+	c = a + b n = n + 2	penjumlahan
-	c = a - b n = n – 2	pengurangan
++	++i	Kenaikan (increment), sama dengan i = i + 1
--	--i	Penurunan (decrement), sama dengan i = i - i
*	c = a * b	perkalian
/	c = a / b	pembagian
	n = n / 2	
%	Sisa = a % b	Menghasilkan sisa dari pembagi. a dan b bilangan bulat
=	a = b	Pemberian nilai
+=	A += 2	Penambahan suatu nilai pada suatu variabel yang sudah ada sebelumnya. Sama dengan a = a + 2
-=	A -= 2	Pengurangan suatu nilai pada suatu variabel yang sudah ada sebelumnya. Sama dengan a = a - 2
*=	A *= 2	Pengalian suatu nilai pada suatu variabel yang sudah ada sebelumnya.nSama dengan a = a * 2
/=	a /= 2	Pembagian suatu nilai pada suatu variabel yang sudah ada sebelumnya.mSama dengan a = a / 2
%=	A %= 2	Sisa dari suatu nilai pada suatu variabel yang sudah ada sebelumnya yang dibagi oleh nilai atau variable lainnya. Sama dengan a = a % 2
*	*pointer	Menunjukkan isi dari pointer

9. Logika

Tabel 8. Simbol dan Pembanding

Simbol	Contoh	Logika pembanding
= =	If (a= = b)	Logika sama dengan, digunakan untuk pembanding. Menghasilkan nilai *true* jika a=b
!=	If (a != b)	Tidak sama dengan. Menghasilkan nilai *true* jika a ≠ b.
<	If (a < b)	Logika lebih kecil dari. Menghasilkan nilai *true* jika a < b.
<=	If (a <= b)	Logika lebih kecil sama dengan dari. Menghasilkan nilai true jika a ≤ b.
>	If (a > b)	Logika lebih besar dari. Menghasilkan nilai *true* jika a > b.
>=	If (a >= b)	Logika lebih beser sama dengan dari. Menghasilkan nilai *true* jika a ≥ b.
!	If (!a)	NOT
&&	If (a = = b && a= = c)	AND
||	If (a = = b || a = = c)	OR

10. Manipulasi Bit

Tabel 9. Manipulasi Bit

~	A = ~ b	*Complement*. b = 1100; a = 0011.
&	c = a & b	AND untuk manupulasi *bit*. a = 1100; b = 1001; c = 1000.
|	c = a | b	OR untuk manipulasi bit. a = 1100; b = 1001; c = 0101.
<<	c = a << n	*Shift left*, manipulasi bit menggeser kekiri sejauh n bit a = 1101; n = 2; maka c = 110100.
>>	c = a >> n	Shift Right, manipulasi bit menggeser kekanan sejauh n bit a = 11010;n = 2; c = 0110.

11. Percabangan

a. *if*

bentuk umum dari percabangan ini adalah:

```
if (kondisi) {
// pernyataan
};
```

b. *if – else*

bentuk umum dari percabangan ini adalah:

```
if (kondisi)
{
// pernyataan a
}
else
{
// pernyataan b
};
```

c. *switch – case*

Pernyataan *switch – case* digunakan jika terjadi banyak percabangan. struktur penulisan pernyataan ini adalah sebagai berikut:

```
…..
switch (ekspresi)
{
pernyataan1
```

```
`break;
case konstanta2:
pernyataan2
break;
……
case konstantaN:
pernyataanN
break;
}
```

d. *switch – case – default*

Pernyataan *switch – case – default* hampir sama dengan *switch – case.* Hal yang membedakannya adalah bahwa dengan adanya *default* maka jika tidak terdapat kondisi *case* yang sesuai dengan ekspresi *switch* maka akan menuju pernyataan yang terdapat pada bagian *default*.

```
switch (ekspresi)
{
case konstanta1
break;
case konstanta2:
pernyataan2
break;
….
case konstantaN:
pernyataanN
break;
default:
Pernyataan-pernyataan;
}
```

e. Perulangan *for*

Pernyataan *for* akan melakukan perulangan berapa kali sesuai yang diinginkan. Struktur penulisan perulangan for adalah sebagai berikut:

```
….
For (mulai ; kondisi ; penambahan atau
pengurangan)
{
Pernyataan-pernyataan;
};
```

f. *while*

Bentuk dari perulangan *while* adalah sebagai berikut:

```
while (kondisi)
{
pernyataan-pernyataan;
}
```

g. *do – while*

Bentuk perulangan ini kebalikan dari *while – do*, yaitu pernyataan dilakukan terlebih dahulu kemudian diuji kondisinya

```
do
{
pernyataan-pernyataan;
} while (kondisi);
```

12. Konversi pola (%)

Karakter % dipakai sebagai operator konversi pola. Konversi pola akan sangat berguna pada saat kita menampilkan hasil ke LCD.

a. %d menampilkan bilangan bulat positif.
   Contoh: Sprintf(buf,”angka%d”,14);
b. %o menampilkan bilangan octal bulat.
c. %x menampilkan bilangan heksadesimal bulat.
d. %u menampilkan bilangan decimal tanpa tanda.
e. %f menampilkan bilangan pecahan.

f. %i menampilkan bilangan integer.

g. %c menampilkan karakter yang ditunjukkan bilangan ASCII.

13. Prosedur

Prosedur adalah suatu kumpulan instruksi untuk mengerjakan suatu keperluan tertentu tanpa mengembalikan suatu nilai.

```
…..
void nama_prosedur (parameter1,
parameter2,……parameterN)
{
}
…..
```

14. Fungsi

Fungsi adalah suatu kumpulan instruksi untuk mengerjakan suatu keperluan tertentu dengan hasil akhir pengembalian nilai dari keperluan tersebut.

```
…..
Type data nama_fungsi (parameter1, parameter2,
….parameterN)
{
Pernyataan-pernyataan;
return variable_hasil;
}
….
```

15. Memasukkan Bahasa *Assembly*

Sering disebut juga dengan *in-line assembly*. Pemrograman dengan bahasa C ini masih dapat memasukkan bahasa *assembly* ke dalam program C. yaitu:

```
…
#asm //dimulai dengan #asm
nop // blok bahasa assembly
```

```
nop //
#endasm // diakhiri dengan #endasm
```

….

Atau jika hanya beberapa instruksi maka kita bisa melakukannya dengan cara:

….

```
#asm("nop\nop\nop")
```

BAB III
KONSEP RANCANGAN

A. Identifikasi kebutuhan

Merancang sebuah servo kontroler sebagai penggerak kaki robot dengan komunikasi serial berbasis ATmega16 dibutuhkan beberapa komponen terdiri dari:

1. Baterai untuk menyuplai daya ke alat.
2. Motor Servo Standart sebagai motor yang dikontrol.
3. Sistem minimum ATMega16 sebagai prosesor utama.
4. Sistem minimum ATMega16 sebagai kontroler servo.
5. UBEC sebagai pembatas tegangan dan penguat arus.

B. Analisis Kebutuhan

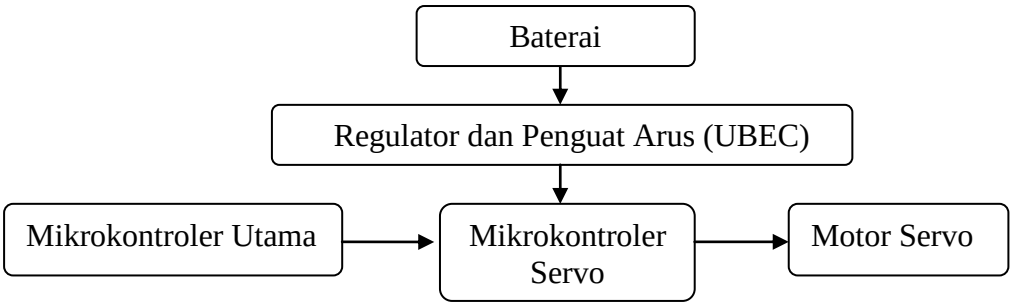
Berdasarkan identifikasi kebutuhan yang ada, maka diperlukan beberapa speifikasi dari komponen atau rangkaian sebagai berikut:

1. Baterai dibutuhkan untuk menyuplai daya ke alat. Tegangan baterai yang digunakan sebesar 7,4 Volt dengan arus 1600MAh. Sebagai pengaman maka di tambahkan sebuah diode
2. Motor Servo pada bagian ini digunakan sebagi motor yang dikontrol. Sesuai dengan fungsinya motor servo digunakan sebagai penggerak lengan-lengan yang ada pada robot berkaki.

3. Alat ini menggunakan dua buah mikrokontroler ATMega16. Satu unit ATmega16 sebagai pengendali utama untuk menangani semua proses sistem dan unit kedua digunakan sebagai pengontrol gerak motor servo
4. UBEC digunakan sebagai pembatas tegangan dan penguat arus untuk menyuplai daya pada rangakaian dan motor servo yang membutuhkan daya yang cukup besar.

C. Perancangan Alat

Rangkaian elektronik servo kontroler ini terdiri atas beberapa blok, diantaranya sebagai berikut:



Gambar 14. Blok Diagram Sistem servo kontroler

Gambar No.12 dapat dilihat alur proses yang terjadi pada servo kontroler yang menggunakan mikrokontroler. Berikut ini merupakan penjelasan dari masing-masing blok:

1. Baterai merupakan penyuplai tegangan yang digunakan untuk menghidupkan rangkaian dan menggerakan motor servo.

2. UBEC merupakan rangakaian regulator dengan sistem *switching* yang dapat membatasi tegangan yang bersumber dari baterai 7,4 Volt menjadi 5 Volt dan membuat arus *continue* untuk menyuplai servo.
3. Mikrokontroler utama merupakan pengendali utama untuk di-*interface*-kan dengan rangkaian yang lain termasuk dengan mikrokontroler servo.
4. Mikrokontroler servo merupakan kontroler servo yang jembatan dari mikrokontroler utama dengan servo. Pada mikrokontroler servo dengan mokrokontroler utama terhubung secara *serial* USART.
5. Motor servo merupakan motor yang nantinya digerakan untuk mendirikan robot dan dapat berjalan.

D. Perencanaan Rangkaian

1. Catu Daya

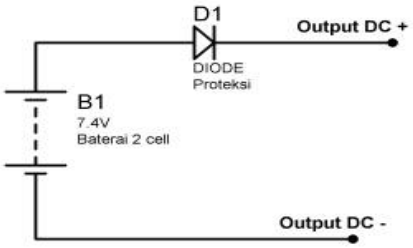
Rangkaian catu daya sangat penting karena tanpa catu daya alat ini tidak dapat bekerja. Rangkain catu daya digunakan untuk menyuplai tenaga ke beberapa rangkaian seperti dua buah sistem minimum ATMega16 dan motor servo.

Rangakaian catu daya terdiri dari dua buah bagian besar yaitu baterai dan UBEC.

a. Baterai

Baterai merupakan salah satu sumber tegangan yang paling baik, karena dapat menyuplai tegangan DC (*direct Current)* secara

continue. Karena output baterai sudah berbentuk tegangan DC maka tidak diperlukan rangkaian penyearah dan rangakaian filter. Jika ditambah rangkaian penyearah maka difungsikan sebagai proteksi agar tidak terjadi hubung singkat jika terjadi kesalahan pemasangan polaritas baterai.



Gambar 15 Rangkaian Baterai dan Proteksi

b. UBEC

Rangkaian UBEC merupakan rangkaian *regulator* buatan pabrik yang dipasarkan dalam kondisi siap pakai. Rangkaian ini bekerja dengan metode *switching* yang dapat memberikan output tegangan 5 Volt atau 6 Volt tergantung penggunaan. Rangakaian ini dapat mengeluarkan arus maksimal 5 Ampere dengan arus *continue* 3A. Pada servo kontroler ini digunakan tegangan 5 volt, sesuai dengan tegangan kerja sistem minimum ATmega16 dan motor servo yang digunakan. Pemilihan tegangan output menggunakan jamper yang sudah tersedia pada rangkaian.

Gambar 16. *Universal Battery Elimination Circuit*

2. Rangkaian Sistem Minimum Mikrokontroler Utama

Kebutuhan robot berkaki membutuhkan beberapa sensor dan perangkat pendukung kecerdasan robot. Berikut ini merupakan kebutuhan port yang harus disediakan: 10 untuk sensor jarak, 6 untuk LCD, 2 untuk sensor api, 1 untuk indikator LED, 2 untuk kipas, 12 untuk servo, 1 untuk tombol, tiga untuk pengisian dan *sound activator*. Total penggunaan port adalah 41 port, jika menggunakan satu buah *chip* ATmega16 tidak mencukupi karena port I/O yang tersedia 32 port. Solusi permasalahan ini adalah menggunakan dua buah *chip* IC ATmega16 dan yang terpisah adalah servo yang dikendalikan dengan satu *chip* IC.

Rangkaian sistem minimum ini digunakan sebagai pusat kendali dari keseluruhan sistem kaki robot. Salah satu yang dikendalikan adalah IC servo kontroler dengan menggunakan PORT D yang terdapat pada kaki IC mikrokontroler ATmega16 menggunakan komunikasi *serial* USART sebagai jalurnya.

Berikut ini merupakan penggunaan port pada IC mikrokontroler utama:

a. Port A

Empat port A digunakan sebagai penampil pengaturan program dengan terhubung pada LCD. Empat port digunakan sebagai pembaca sensor garis menggunakan fungsi ADC-nya.

b. Port B

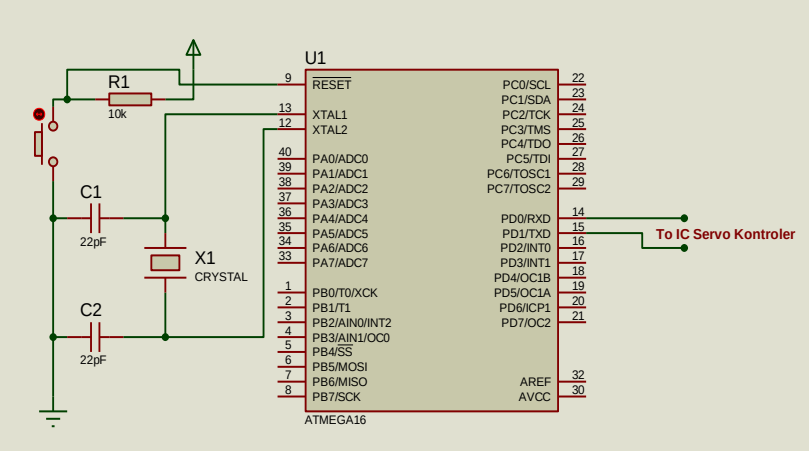
Empat port B digunakan sebagai pengendali sensor jarak. Tiga port digunakan sebagai jalur untuk mengisikan program ke IC mikrokontroler utama dan satu untuk led.

c. Port C

Dua port C digunakan untuk mengendalikan sensor jarak, dua untuk LCD, dua untuk sensor api, satu untuk led dan satu menghidupkan kipas.

d. Port D

Empat port D digunakan untuk mengendalikan sensor jarak, dua komunikasi dengan servo kontroler dan satu digunakan untuk tombol.



Gambar 17. Rangkaian sistem minimum mikrokontroler utama

3. Rangkaian Servo Kontroler

Rangakaian servo kontroler merupakan rangkaian yang terbuat dari mikrokontroler ATmega16 standart yang biasa di program sebagai IC utama. Pada rangkaian ini mikrokontroler ATmega16 digunakan sebagai jembatan pengendali antara motor servo dengan mikrokontroler utama. Rangkaian servo kontroler ini sangat berguna, karena selain sebagai jembatan dapat mengatasi jumlah kebutuhan port pada IC mikrokontroler utama dan tidak mengganggu program utama.

Sesuai dengan fungsinya rangkaian ini hanya digunakan sebagai servo kontroler, maka port hanya terhubung dengan mikrokontroler utama dan motor servo sebagai objeknya. Berikut ini merupakan masing-masing fungsi port I/O:

a. Port A

Port A semua digunakan untuk mengendalikan motor servo.

b. Port B

Port B semua digunakan untuk mengendalikan motor servo.

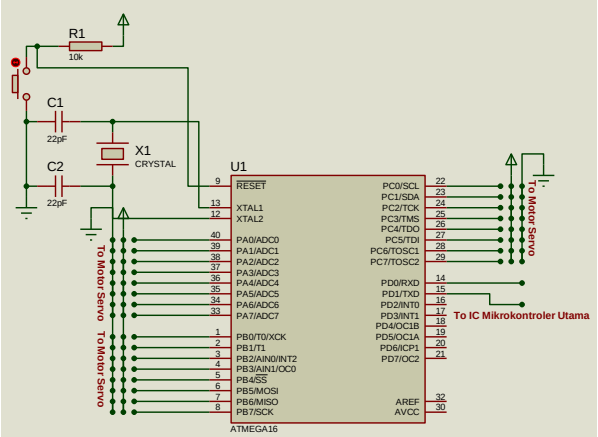
c. Port C

Port C semua digunakan untuk mengendalikan motor servo.

d. Port D

Port D yang digunakan hanya 3 buah yaitu Port D.0, Port D.1 dan Port D.2. Port D.0 berfungsi sebagai Recive data dari Transmit data mikrokontroler utama. Port D.1 berfungsi Sebagai Transmit

data dari servo kontroler. Port D.2 berfungsi sebagai pengontrol data Recive yang di kirim. Selain port diatas tidak digunakan.



Gambar 18. Rangkaian Servo kontroler

E. Langkah Pembuatan Alat

Langkah pembuatan alat pada proyek akhir ini terdiri dari pembuatan PCB, pemasangan komponen, pelarutan dan pengeboran PCB.

1. Pembuatan PCB

   a. Pembuatan Desain PCB

      Langkah awal dari pembuatan PCB adalah menggambar rangkaian dan *layout* dengan perakat lunak yang tersedia. Hasil penggambaran dapat dilihat pada lampiran

   b. Penyablonan PCB

      Desain PCB selesai dibuat, maka langkah berikutnya adalah penyablonan desain ke PCB polos. Langkah pertama dengan mencetak desain PCB pada kertas *glossy* secara mirror.

Langkah kedua menempelkan desain yang terdapat pada kertas *glossy* ke PCB yang polos. Supaya desain dapat tersalin ke PCB maka kertas tersebut disetrika diatas PCB sampai melekat kurang lebih 8 menit.

Setelah kertas melekat pada PCB langkah selanjutnya merendam PCB beserta kertas yang menempel. Setelah beberapa menit hilangkan kertas secara perlahan sampai yang tersisa hanya jalur rangkaian saja.

c. Pelarutan dan Pengeboran PCB

Langkah selanjutnya adalah pelarutan PCB dengan cairan kimia *feri chloride* dan air hangat hingga jalur rangkaian terbentuk. Setelah jalur terbentuk sesuai desain, mengangkat PCB dari cairan *feri chloride* dan membersihkan denan air sampai bersih dan jalur yang terlihat berbentuk tembaga.

2. Pemasangan Komponen

Pada bagian pemasangan komponen dapat dilaksanakan dengan urutan berikut:

a. Menyiapkan komponen yang akan dipasang sesuai dengan desain.
b. Memasang jumper dan komponen yang kecil dahulu baru mulai yang besar.
c. Menyolder kaki komponen sampai semua komponen terpasang.

Sebelum menguji rangkaian dengan member tegangan terlebih dahulu di cek jalur dan soldiranya agar tidak terjadi konsleting saat pengujian dan penggunaan.

3. Pembuatan Kerangka Robot

Pembuatan kerangga menggunakan bahan dasar *acrylic* yang di potong menggunakan *laser cutting*. Pembuatan kerangka melalui beberapa tahapan sebagai berikut:

a. Desain kerangka menggunakan perangkat lunak yang bias untuk menggambar secara akurat yaitu Corel Draw.
b. Pemotongan dengan menggunakan *laser cutting* sesuai dengan desain.
c. Pengeleman dengan lem *acrylic* yang tersedia di pasaran.
d. Penyatuan kerangka perbagian dengan motor servo dan rangkaian.

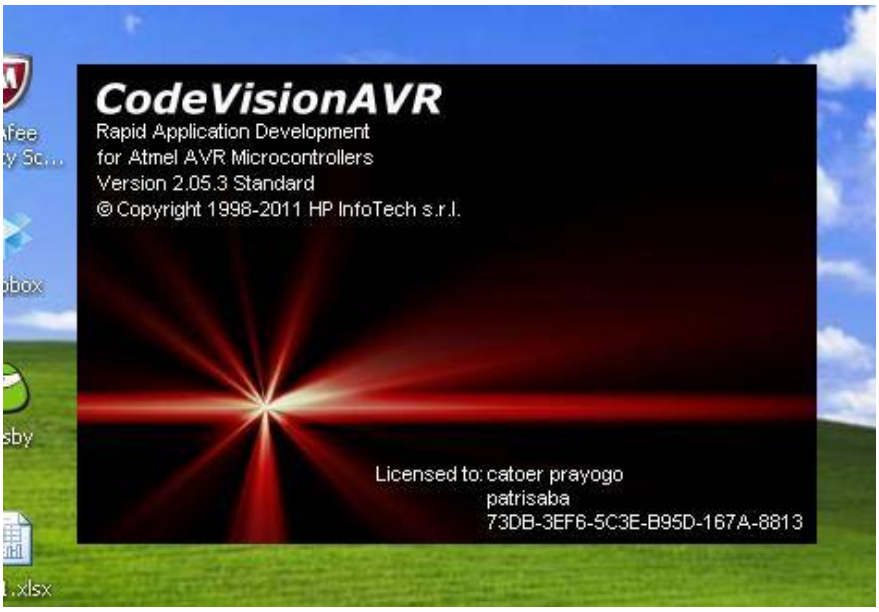
Desain kerangka dapat dilihat pada lampiran akhir laporan.

F. Perancangan Perangkat Lunak *(Software)*

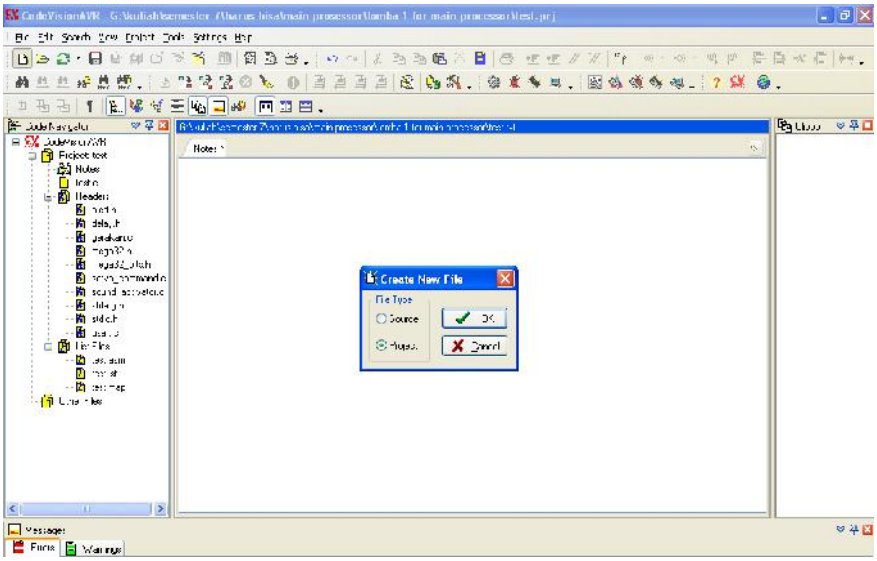
1. Program

Sebelum menulis *software* yang akan diisikan ke-chip, terlebih dahulu menuliskan programnya. Pemrograman bahasa C dalam laporan ini menggunakan *software* yang sudah ada yaitu CV AVR walaupun dapat menggunakan *software* yang lain. *Software* ini maka akan mudah untuk pembuatanya karena menggunakan *compiler* sebagai penguji. Cara memulai pembuatan program terlebih dahulu

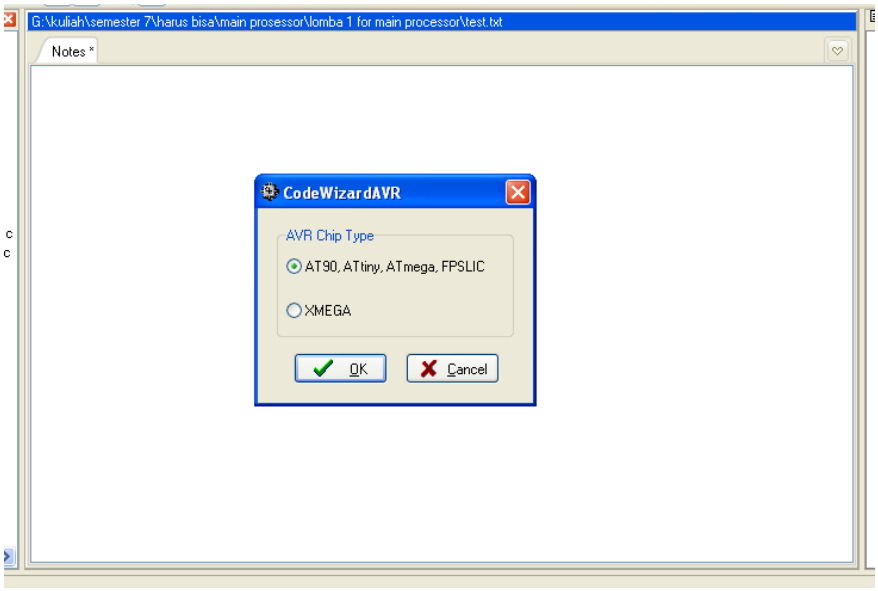
jalankan *software*-nya. Jika sudah berjalan mulai kita membuat *new project*.



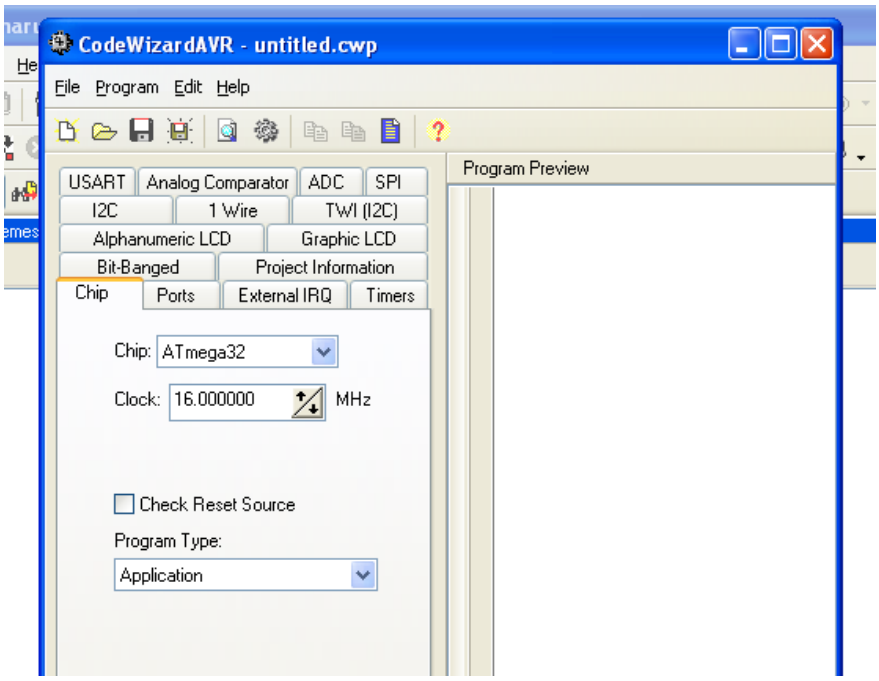
Gambar 19. membuka aplikasi CV AVR



Gambar 20. new project



Gambar 21. Pemilihan Chip Secara Umum

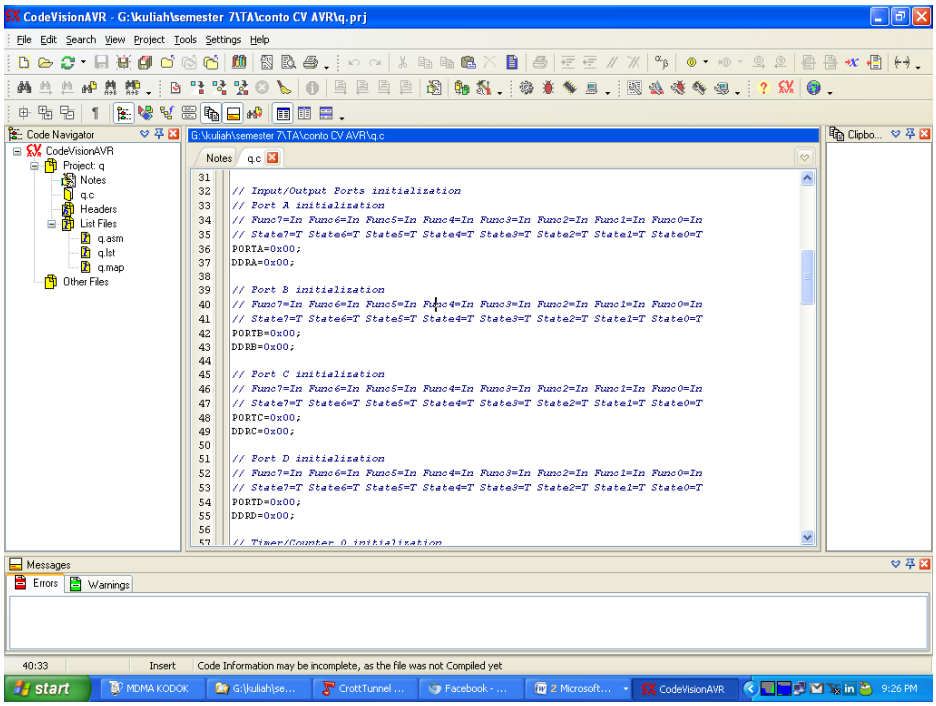


Gambar 22. Code WizardAVR

Gambar 20. merupakan Gambar penentuan beberapa pengaturan. Pengaturan tersebut: Chip, Crystal, Port I/O, Timers, Alphanumeric LCD, dan lain-lain. Setelah pengaturan selesai, maka disimpan dengan

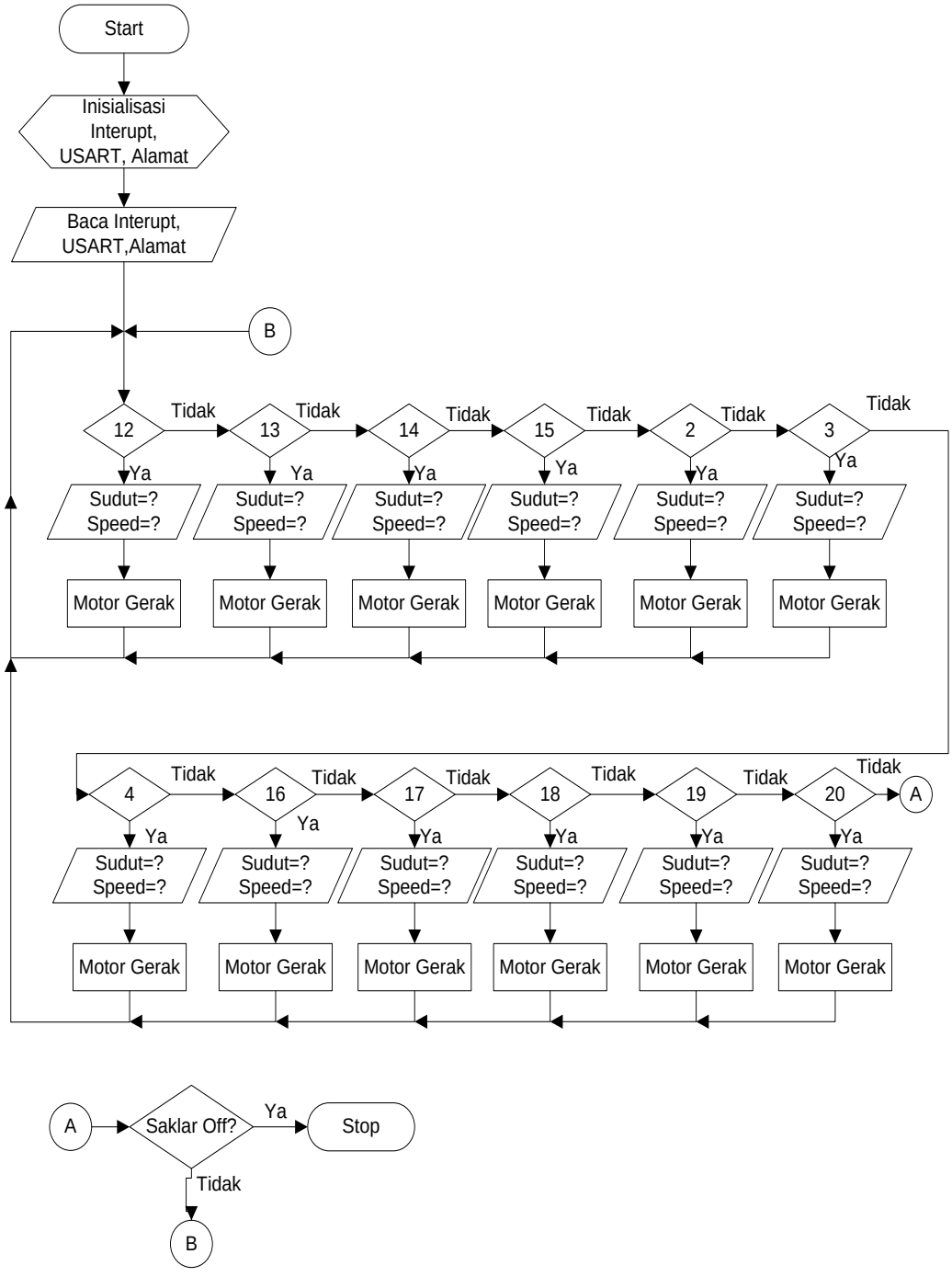
tiga kali penyimpanan dengan memilih generate program, save, and exit.

Berikut ini merupakan program bawaan yang sudah di generate oleh CV AVR dan siap untuk ditambah dengan program lainya.



Gambar 23. aplikasi siap untuk di tambah dengan program lainya

2. Percabangan Flowchart



Gambar 24. Flow Chart

Flowchart diatas menjelaskan tentang alur program yang terdapat pada alat yang dibuat yaitu servo kontroler menggunakan ATmega16. Berikut ini adalah penjelasan dari flowchart:

a. START, menjelaskan mulainya program
b. Inisialisasi Interupt, USART, dan alamat menjelaskan persiapan dari port interrupt untuk menyela program yang berjalan, port USART menerima data dari chip IC utama. Mempersiapkan alamat port yang akan digunakan sebagai control servo berikut:
    1) 12 = 6 = PortA.3
    2) 13 = 7 = PortA.4
    3) 14 = 8 = PortA.5
    4) 15 = 9 = PortA.6
    5) 2 = 10 = PortA.7
    6) 3 = 11 = PortB.2
    7) 4 = 12 = PortB.3
    8) 16 = 13 = PortB.4
    9) 17 =14 = PortC.0
    10) 18 = 15 = PortC.1
    11) 19 = 16 = PortC.2
    12) 20 = 17 = PortC.3
c. Baca interrupt, USART, dan alamat menjelaskan jika terdapat interup yang diterima maka USART akan menerima data dari chip IC utama.

d. 12, 13, 14, 15, 2, 3, 4, 16, 17, 18, 19 dan 20 mejelaskan alamat yang dituju sesuai inisialisasi. Jika alamat sesuai maka akan diteruskan ke proses selanjutnya, tetapi jika tidak akan mencari alamat yang sesuai dulu.
e. Sudut=? dan speed=? terisi jika alamat tujuan tepat, karena data selanjutnya yang dikirim adalah sudut diteruskan dengan kecepatan.
f. Motor Servo akan bergerak sesuai dengan sudut yang diterima dengan kecepatan yang telah ditentukan.
g. Kembali membaca interrupt, USART dan alamat, akan looping seperti ini terus.
h. Berhenti jika saklar power dimatikan.

G. Spesifikasi Alat

servo kontroler sebagai penggerak kaki robot dengan komunikasi *serial* USART berbasis Mikrokontroler ATmega16 mempunyai spesifikasi:

1. Pembuatan rangka robot menggunakan acrylic yang dipotong sesuai desain dan digabung dengan servo menggunakan lem acrylic dan mur baut.
2. Sendi-sendi yang bergerak digerakan oleh servo Hitec dengan seri HS-5625MG. Servo ini mempuyai spesifikasi tegangan kerja 5V, merupakan servo standar dan menggunakan metal gear.

3. Arah gerak terprogram di chip utama yang mengirimkan ke chip servo kontroler.
4. Sistim pengendalian menggunakan dua buah chip yang keduanya merupakan mikrokontroler ATmega16.
5. Tegangan kerja yang digunakan adalah 7,4V 1600mAh yang bersumber dari baterai.
6. Keluaran dari robot berkaki berupa gerakan. Jika tombol power di hidupkan maka robot akan berdiri, untuk mulai menjalankan dengan menekan tombol start yang telah ada. Robot akan berjalan membentuk sebuah kotak. Tombol power dimatikan maka robot baru dapat berhenti

H. Pengujian Alat

Pengujian alat dilakukan untuk mendapatkan data penelitian. Pengujian alat ini dilakukan dengan dua pengujian, yaitu:

1. Uji fungsional

   Pengujian ini dilakukan dengan cara menguji setiap bagian alat berdasarkan karakteristik dan fungsi masing-masing. Pengujian ini dilakukan untuk mengetahui apakah setiap bagian dari perangkat telah bekerja sesuai dengan fungsi dan keinginan.

2. Uji unjuk kerja

   Pengujian unjuk kerja alat dilakukan dengan cara melihat unjuk kerja alat. Hal-hal yang perlu diamati antara lain: rangkaian sistem

minimum dan pergerakan servo yang telah terpasang. Dari pengujian ini akan diketahui kinerja dari alat yang dibuat.

I. Pengoperasian Alat

Pengoprasian alat ini dapat dilakukan dengan cara sebagai berikut:

1. Pastikan alat terhubung dengan tegangan yang bersumber dari baterai dan sudah distabilkan menjadi 5 v dengan UBEC.
2. Pastikan semua kabel terhubung dengan sempurna.
3. Tekan saklar pada posisi ON untuk menghidupkan.
4. Tekan tombol start untuk mulai menjalankan.
5. Tekan saklar pada posisi OFF untuk mematikan.

BAB IV

PENGUJIAN DAN PEMBAHASAN

Tujuan dari pengujian dan pembahasan adalah untuk mengetahui kinerja alat baik secara perbagian blok rangkaian maupun sistem keseluruhan apakah sudah seperti yang diharapkan atau belum. Pengujian ini meliputi:

A. Hasil Pengujian

1. Pengujian Tegangan

   a. Pengujian Tegangan Catu Daya

      Tabel 9. Pengukuran Regulator tegangan UBEC

No	Pengukuran	Tegangan Input (V)	Tegangan Output (V)	Teganagn UBEC	*Error*
1	Tanpa Beban	7,4	5,01	5	0.2%
2	Denagn Beban	7,4	4,96	5	0.8%

      Hasil dari pengukuran tegangan regulator untuk keluaran UBEC saat tanpa beban adalah 5,01V dan saat dengan beban menjadi 4,96V. Kondisi tersebut sesuai dengan tegangan kerja yang dihasilkan oleh UBEC yaitu 5V. *Error* yang terdapat pada rangkaian regulator ini memiliki *presentase* saat tanpa beban 0,2% dan saat dengan beban 0,8%. Data yang dihasilkan diatas maka rangkaian regulator yang terdapat UBEC bekerja dengan maksimal.

   b. Pengujian Tegangan Mikrokontroler

      Tabel 10. Pengukuran Tegangan Mikrokontroler

No	V input	Tegangan Kerja Chip	*Error*
1	4,96	4.92	1,6%

Hasil pengukuran tegangan yang terdapat pada mikrokontroler adalah 4,92V. Kondisi tersebut mempunyai presentase *error* sebesar 1,6%.

c. Pengujian Tegangan Servo

Tabel 11. Pengukuran Tegangan Mikrokontroler

No	V input	Tegangan Kerja Chip	*Error*
1	4,96	4.88	2,4%

Hasil pengukuran tegangan yang terdapat pada mikrokontroler adalah 4,88V. Kondisi tersebut mempunyai presentase *error* sebesar 2,4%.

2. Pengujian Chip Servo kontroler

Pengendalian gerakan motor servo dapat dilakukan dengan menggunakan *delay*. Teknik ini menggunakan sistem lebar pulsa untuk mengemudikan putaran motor. Sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor servo. Cara pemberian pulsa secara high selama 1500uS pada periode selebar 20mS maka sudut dari sumbu motor akan berada pada posisi tengah 0^0. Penggerakan servo dapat dilakukan dengan memberi pulsa 600uS maka akan bergeser berlawanan arah jarum jam -90^0 dan pulsa high selama 2400us akan bergerak searah jarum jam sebesar 90°.

Tabel 12. Pengujian Chip Servo kontroler

No	Kode Program (^{0})	Sudut (^{0})	*Delay* (uS) Standart	Sudut (^{0}) Terukur	*Delay* (uS) Terukur	Selisih (^{0})	*Error (%)*
1	0	-90	600	-84.6	654	-5.4	6
2	10	-80	700	-78	720	-2	2.5
3	20	-70	800	-69	810	-1	1.42
4	30	-60	900	-59.5	905	-0.5	0.83
5	40	-50	1000	-49.7	1003	-0.3	0.6
6	50	-40	1100	-40.1	1099	0.1	0.25
7	60	-30	1200	-30	1200	0	0
8	70	-20	1300	-20.1	1299	0.1	0.5
9	80	-10	1400	-10.1	1399	0.1	1
10	90	0	1500	0	1500	0	0
11	100	10	1600	9.9	1599	0.1	1
12	110	20	1700	19.8	1698	0.2	1
13	120	30	1800	30	1800	0	0
14	130	40	1900	40.1	1901	-0.1	0.25
15	140	50	2000	50.3	2003	-0.3	0.6
16	150	60	2100	59.5	2095	0.5	0.83
17	160	70	2200	69	2190	1	1.42
18	170	80	2300	78.1	2281	1.9	2.37
19	180	90	2400	84.8	2348	5.2	5.77

Penjelasan Tabel 12.

a. Kode Program (^{0}) merupakan cara penulisan pada program di chip IC utama. Contoh 0(^{0}) pada kode program merupakan -90(^{0}) pada gerakan servo.
b. Sudut(^{0}) merupakan besar sudut pada gerakan servo.
c. Delay (uS) Standart merupakan besar delay yang dibutuhkan untuk menggerakan servo. Contoh 600(uS) untuk posisi -90(^{0}).
d. Sudut(^{0}) Terukur merupakan sudut yang terbentuk dari delay yang keluar dari servo kontroler. Contoh 84.6^0 merupakan sudut yang terbentuk dari 654uS.

e. Delay(^{0}) Terukur merupakan delay yang dikeluarkan oleh servo kontroler.

f. Selisih (^{0}) merupakan selisih antara sudut standart dengan sudut yang terukur. Contoh 90^0 dengan 84.6^0 mempunyai selisih 5.4^0.

g. *Error* (%) merupakan presentase dari selisih sudut. Contoh 6% merupakan selisih 5.4 dibagi sudut standart 90^0 dikalikan 100%.

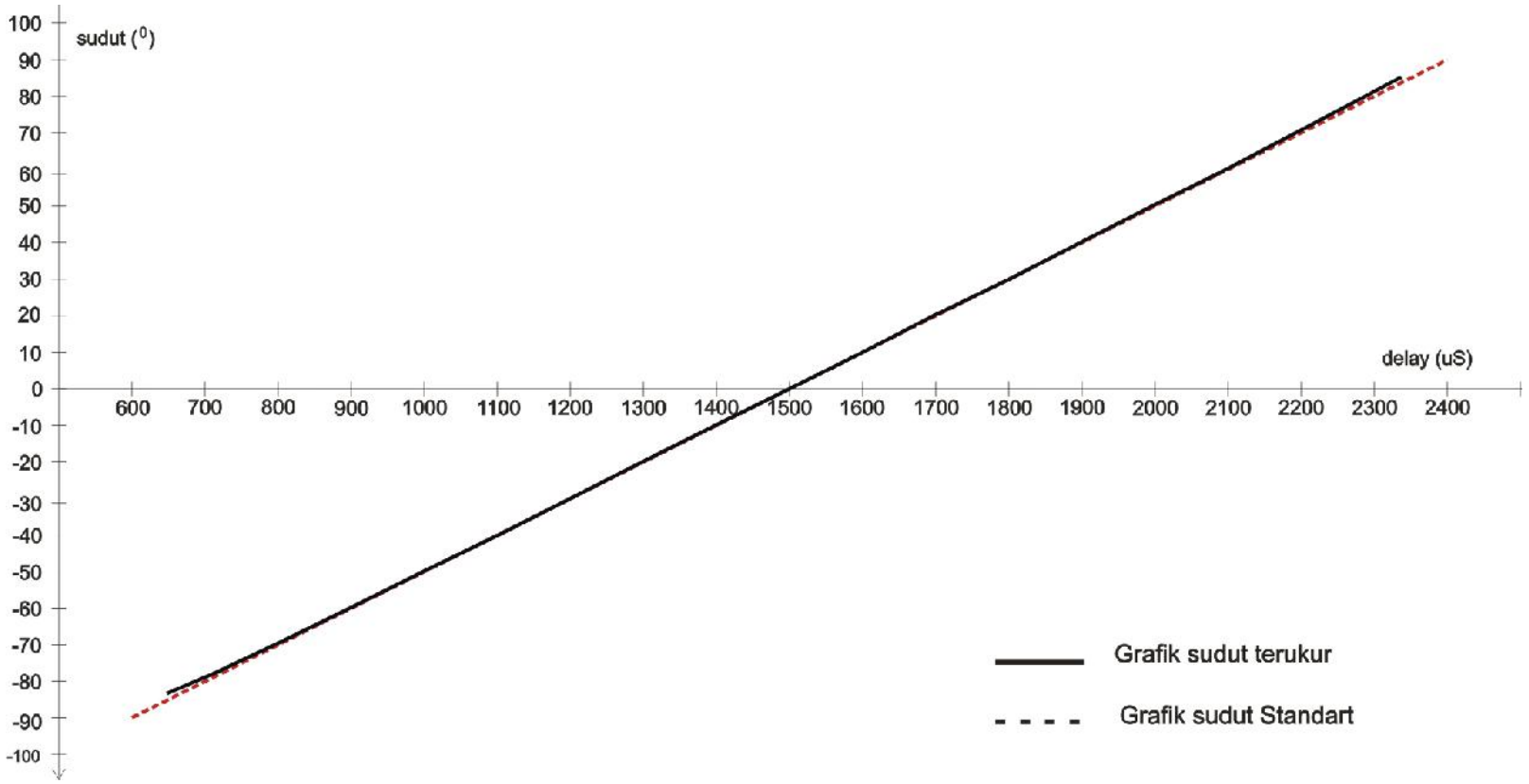
$$\frac{selisih}{sudut} x\ 100\% = error$$

Data Tabel 12. Menunjukan bahwa servo kontroler yang dibuat untuk mengontrol servo dengan jumlah banyak dapat bekerja. Jika lebar pulsa lebih dari batas maksimal akan menyebabkan sudut servo kembali kearah 0^0.

Gambar 17. Grafik karateristik servo kontroler menunjukan kinerja yang hampir sama dengan input servo standart. Karateristik servo diatas dapat diaplikasikan pada robot berkaki, karena kebutuhan pergerakan servo pada robot hanya menggunakan sudut yang masih medekati standart.

Error yang terjadi pada servo kontroler ditunjukan oleh gambar grafik karaeristik servo kontroler. Keadaan standart, pergerakan servo membentuk gambar grafik yang linier. Error terbesar terjadi dikedua ujung pergerakan servo, karena terdapat lengkungan yang menyebabkan tidak linier 100%.

Berikut ini merupakan grafik kinerja servo kontroler:



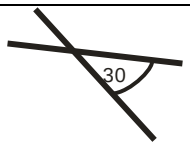
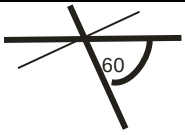
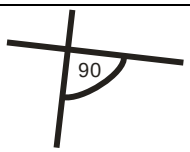
Gambar 25. Grafik karateristik servo kontroler

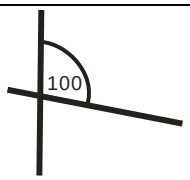
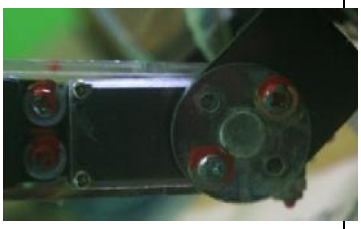
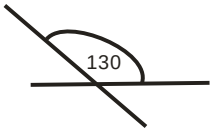
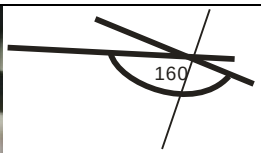
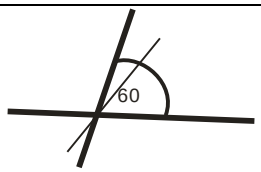
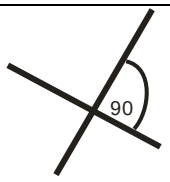
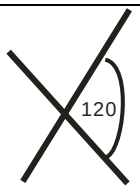
Servo kontroler ini tidak bisa megerakan servo pada kondisi -90^0 atau 90^0, karena servo kontroler tidak mampu mengeluarkan pulsa yang sesuai untuk 90^0 maupun -90^0 yaitu 600uS dan 2400uS. Alat ini mampu mengeluarkan pulsa yang mendekati standart pada -70^0 sampai 70^0. Ketidak mampuan servo kontroler ini tidak mejadi masalah karena pergerakan robot berkaki menggunakan sudut yang masih mendekati standart.

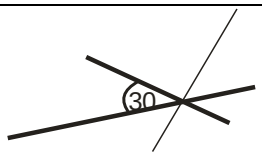
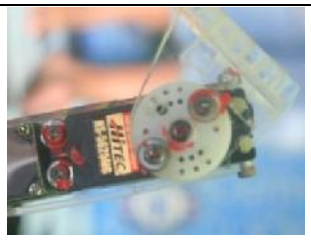
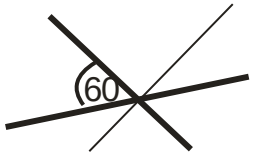
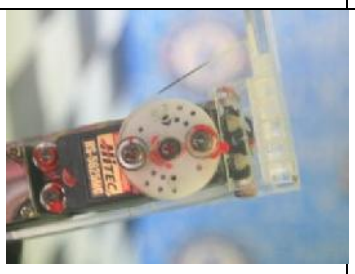
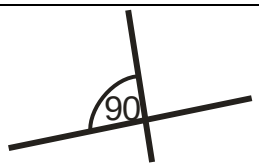
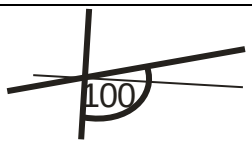
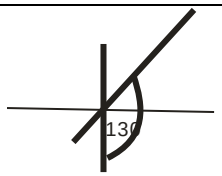
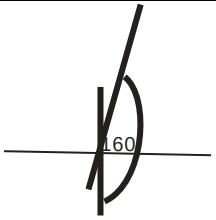
3. Pengujian gerakan servo pada robot

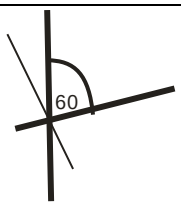
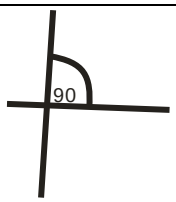
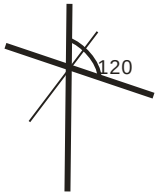
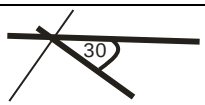
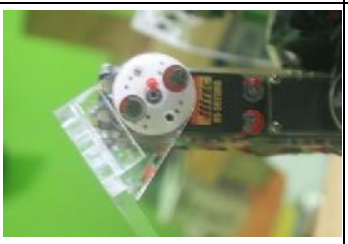
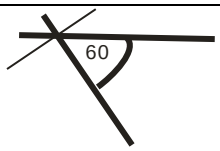
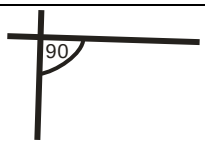
Pengujian gerakan servo dengan menggunakan servo kontroler sebagai kontrol, dapat dilihat pada Tabel 13. Sudut yang diambil merupakan gerakan yang terjadi pada robot.

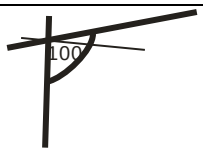
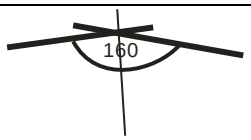
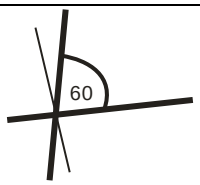
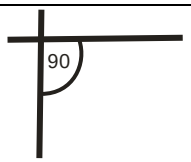
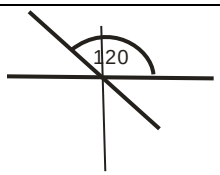
Tabel 13. Pengamatan gerakan servo

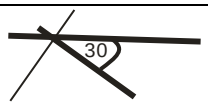
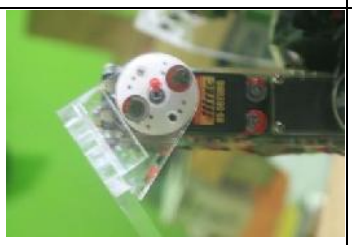
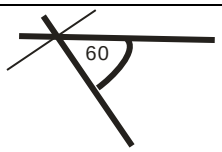
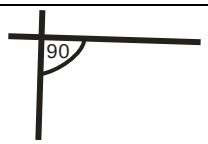
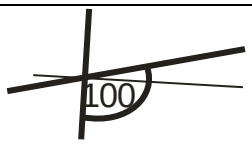
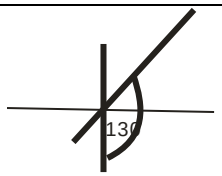
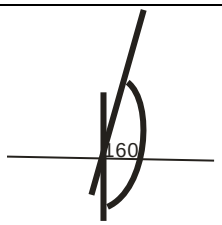
No	Alamat	Sudut(^{0})	Foto	Pembacaan sudut(^{0})
1	A1 depan kanan servo kaki panjang	30		
		60		
		90		

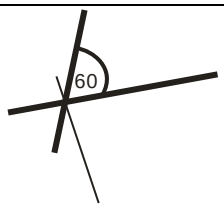
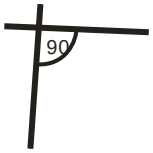
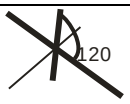
2	A2 depan kanan servo tengah	100		
		130		
		160		
3	A3 depan kanan servo body	60		
		90		
		120		

4	B1 belakang kanan servo kaki panjang	30		
		60		
		90		
5	B2 depan kanan servo tengah	100		
		130		
		160		

6	B3 depan kanan servo body	60		
		90		
		120		
7	C1 belakang kiri servo kaki panjang	30		
		60		
		90		

8	C2 belakang kiri servo tengah	100		
		130		
		160		
9	C3 belakang kiri servo body	60		
		90		
		120		

10	D1 depan kiri servo kaki panjang	30		
		60		
		90		
11	D2 depan kiri servo tengah	100		
		130		
		160		

12	D3 depan kiri servo body	60		
		90		
		120		

Tabel 13. Merupakan foto hasil pergerakan sesuai kode program yang ada pada chip IC utama. Posisi yang difoto merupakan sudut-sudut tertentu pada masing-masing servo. Setiap servo di foto 3 sudut yang berbedad dengan interval 30^0. Pembacaan sudut dilihat dari garis tebal yang terdapat juring. Garis tipis merupakan garis bantu untuk mempermudah pembacaan yang sesuai dengan gambar disampingnya.

B. Pembahasan

1. Hardware

Robot berkaki pada proyek akhir ini hanya membahas tentang servo kontroler. Perangkat pendukung lainya tidak dipasang.

a. Servo

Servo dapat bergerak sesuai dengan yang diinginkan untuk mengerakan kaki robot. Pergerakan servo pada dasarnya diatur oleh *delay* 600uS sampai dengan 2400uS. Adanya servo kontroler dapat diatur langsung dengan menuliskan besaran sudutnya. Besar sudut dikontrol dari chip IC utama.

Sesuai dengan gerak standart servo mulai dari -90^0 sampai dengan 90^0, servo dapat bergerak hampir maksimal yaitu pada $-84{,}6^0$ sampai dengan $84{,}8^0$. Hal ini terjadi karena berdasarkan kemampuan servo kontroler yang dibuat. Servo kontroler dapat digunakan untuk menggerakan kaki pada robot berkaki, karena kebutuhan gerak pada robot cerdas berkaki tidak menggunakan gerakan lebih dari $-84{,}6^0 \leftrightarrow 84{,}8^0$.

b. ATmega16

Atmega16 mempunyai port I/O yang cukup banyak. ATmega16 adalah sebuah mikrokontroler yang didalamnya terdapat RAM, ROM, mikropocesor, port I/O, Clock. Servo kontroler ini hanya memfungsikan sebagian dari kemampuan seluruhnya dari ATmega16. Fungsi yang digunakan meliputi port I/O dan USART. ATmega16 pada alat ini dapat dikatakan sebagai buffer dari chip utama ke servo yang banyak.

ATmega16 mempunyai port I/O sebanyak 32 yang masing-masing mempunyai fungsi khusus. Servo kontroler ini menggunakan 24 port sebagai output dengan fungsi standart dan 3 port sebagai input dengan menggunakan fungsi khususnya sebagai port serial. Penggunaan ATmega16 karena sudah mencukupi untuk pembuatan servo kontroler dengan output 24 servo.

c. UBEC

UBEC dapat bekerja maksimal dengan menggerakan servo sesuai dengan torsinya. Setiap servo mampu menguatkan arus mencapai 3-5A. Penggunaan UBEC pada robot ini menggunakan 2 biji untuk meggerakan 12 servo. Setiap servo membutuhkan arus sebesar 500mA, jika terdapat 12 servo maka dengan dua buah UBEC sudah menyuplai arus untuk kebutuhan servo.

2. Software

Perangkat lunak atau *software* pada alat ini dibuat mengguankan bahasa pemrograman C. Bahasa C adalah bahasa tingkat tinggi yang lebih dimengerti oleh manusia. Pembuatan kode-kode pemrograman dibantu dengan memfungsikan *compiler Code Vision AVR*. Penggunaan *compiler CVAVR* memudahkan pembuatan program. Bagian-bagian pemrograman meliputi:

a. Definisi Prosesor

```
#include <mega16.h>
```

Baris ini menyatakan bahwa chip yang digunakan adalah salah satu bagian keluarga AVR Atmega dengan seri 16. Pernyataan ini berfungsi memanggil *library* yang terdapat pada *compiler* CV AVR.

b. Definisi Pewaktu

```
#include <delay.h>
```

Pernyataan ini memanggil fungsi *delay* yang ada pada *Code Vision AVR* yang terdapat dalam program jika kita akan menggunakan fungsi yang berkaitan dengan waktu.

c. Definisi Port

```
#define servo_on1 PORTB
#define servo_on2 PORTA
#define servo_on3 PORTC
```

Pernyataaan ini digunakan untuk mendefinisikan servo_on1 pada port B, servo_on2 untuk portA, dan servo_on3 untuk portC. Penyataan diatas memudahkan dalam pemrograman untuk mengelola I/O pada port tersebut.

d. Definisi Jumlah Servo

```
#define jml_servo 24
```

Pendefinisian ini digunakan untuk menyatakan jumlah servo yang bisa digunakan sebanyak 24 servo.

e. Definisi *Delay*

```
#define delay_servo delay_us(500);
```

Pendefinisian ini digunakan untuk memudahkan dalam pemrograman dalam menyebut *delay* sebesar500uS dengan menyebut *delay*_servo.

f. Definisi *Clock* dan Buadrate

```
#define F_CLOCK 12000000L
```

Pendefinisian ini merupakan penggunaan *crystal clock* sebagai pembangkit sinyal sebesar 12 MHz.

```
#define BAUD 9600L
```

Pendefinisian ini merupakan pengaturan baudrate yang digunakan sebesar 9600.

g. Penyertaan *Library* Buatan

```
#include "usart.c"
```

Baris ini digunakan untuk mengikutsertakan *library* yang telah dibuat sendiri dengan nama “usart.c”

h. Definisi *Variable*

```
unsigned int n;
unsigned int alamat;
int servo[jml_servo];
int power,indek_servo,set_servo[jml_servo]
int speed[jml_servo];
```

Beberapa baris ini merupakan pendefinisian *variable* yang berbeda-beda sesuai dengan fungsi masing-masing.

i. Definisi Fungsi

```
unsigned int debug[jml_servo]= {0,1,8,9,10,11,12,13,
//0=0,2=8
                                //0 1 2 3 4  5   6  7
                                 14,15,2,3,4,16,17,18,
                                //8 9 1o 11 12 13 14 15
```

```
                                    19,20,21,22,23,5,6,7};
                                  //16 17 18 19 20 21 22 23
```

Pernyataan ini merupakan metode pengurutan alamat servo sesuai dengan hardwarenya. Fungsi ini memudahkan dalam pemrograman karena servo sudah urut.

```
long int set_sudut (int sudut){
    long out;
    out=(long)432*sudut;
    out=out/180;
    return out;
}
```

Pernyataan ini merupakan fungsi untuk pengaturan sudut pergerakan servo, agar dapat diatur dengan menuliskan sudutnya saja.

j. Definisi External Interupt

```
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    alamat=usart_receive();
    if (alamat==66) power=0;
    else if (alamat==88) power=1;
    else alamat=debug[alamat];
    set_servo[alamat]=set_sudut(usart_receive());
    speed[alamat]=usart_receive();
  }
```

Pernyataan ini merupakan penyela program data untuk memasukan data yang akan diterima oleh USART yang berupa alamat, sudut dan speed.

k. Program Utama

```
void main(void)// Yang menyatakan Program Utama
```

```
{
    PORTA=0x00;
    DDRA=0xff;
    PORTB=0x00;
    DDRB=0xFF;
    PORTC=0x00;
    DDRC=0xFF;
    PORTD=0x00;
    DDRD=0x00;   //menyatakan pendefinisian Input dan Output

    // External Interrupt(s) initialization
    GICR|=0x40;   // INT0: On
    MCUCR=0x02; // INT0 Mode: Falling Edge
    GIFR=0x40;   // INT1: Off
    ACSR=0x80;    //Menyatatan pengaturan Interupt External

    usart_init(MYUBRR);

    #asm("sei")  // penyertaan bahasa asembli

    servo_on1 = 0x00;
    servo_on2 = 0x00;
    servo_on3 = 0x00;
    power=0;

    while (1)
    {
        for
(indek_servo=0;indek_servo<jml_servo;indek_servo++){

            if((indek_servo < 8) && power) {
                servo_on1 = 1 << indek_servo;   //penentuan
lokasi bit yang dipakai.
                for(n=0; n < servo[indek_servo]; n++){
                    delay_us(1);
                 };
```

```
                delay_us(595);
                servo_on1 =0x00;
                delay_servo;

            }else if ((indek_servo < 16) && power){
                servo_on2 = 1 << (indek_servo-8);
                for(n=0; n < servo[indek_servo]; n++){
                    delay_us(1);
                 };
                delay_us(595);
                servo_on2 =0x00;
                delay_servo;

            }else if ((indek_servo < 24) && power){
               servo_on3 = 128 >> (indek_servo-16);
                for(n=0; n < servo[indek_servo]; n++){
                    delay_us(1);
                 };
                delay_us(595);
                servo_on3 =0x00;
                delay_servo;
            }

            if(set_servo[indek_servo]>servo[indek_servo]){
                servo[indek_servo]+=speed[indek_servo];
                if                                          (
servo[indek_servo]>set_servo[indek_servo]){

servo[indek_servo]=set_servo[indek_servo];
                    }
                }
            else
if(set_servo[indek_servo]<servo[indek_servo]){
                servo[indek_servo]-=speed[indek_servo];
                if                                          (
servo[indek_servo]<set_servo[indek_servo]){
```

```
servo[indek_servo]=set_servo[indek_servo];
                } } } }     //pergerakan servo dari sudut satu
kesudut lainya dengan kecepatan yang telah diatur.
```

l. *Library* "usart.c"

Berikut ini merupakan program *library* yang dibuat sendiri:

```
#ifndef _STDIO_INCLUDED_
    #include <stdio.h>
#endif

#ifndef _DELAY_INCLUDED_
    #include <delay.h>
#endif

#ifndef F_CLOCK
    #error F_ClOCK
#endif

#ifndef BAUD
    #error BAUD
#endif

#define MYUBRR F_CLOCK/16/BAUD-1

#ifndef TXEN
    #define TXEN 3
    #define RXEN 4
    #define URSEL 7
    #define USBS 3
    #define UCSZ0 1
    #define UDRE 5
    #define RXC 7
#endif

#pragma used+
```

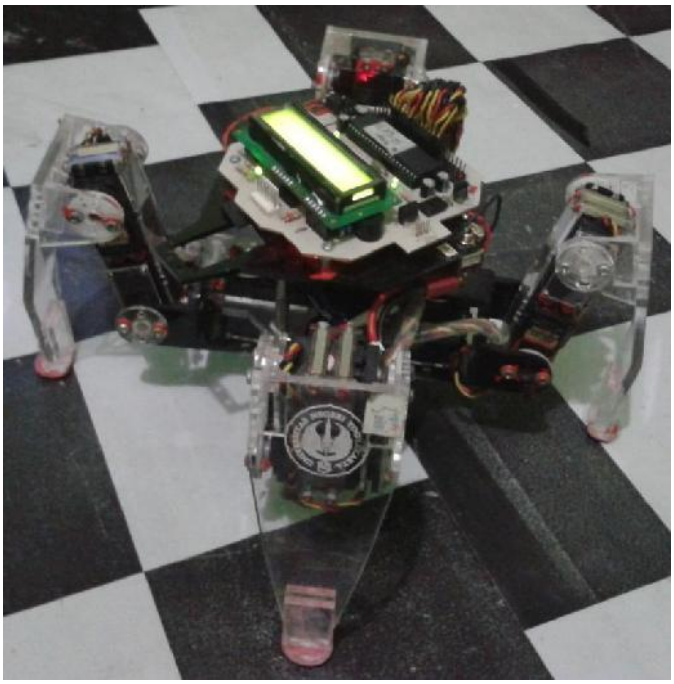
```
void usart_init( unsigned int ubrr);
void usart_transmit( unsigned char data );
unsigned char usart_receive( void );
#pragma used- // agar tidak eror saat di Compile

void usart_init( unsigned int ubrr)
{
    /* Set baud rate */
    UBRRH = (unsigned char)(ubrr>>8);
    UBRRL = (unsigned char)ubrr;
    /* Enable receiver and transmitter */
    UCSRB = (1<<RXEN)|(1<<TXEN);
    /* Set frame format: 8data, 2stop bit */
    UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
    }
//fungsi    usart_init    merupakan    penginisialisasikan
komunikasi usart
void usart_transmit( unsigned char data )
{
    /* Wait for empty transmit buffer */
    while ( !( UCSRA & (1<<UDRE)) );

    /* Put data into buffer, sends the data */
    UDR = data;
    }
    //fungsi untuk transmin atau mengirim

unsigned char usart_receive( void )
{
    /* Wait for data to be received */
    while ( !(UCSRA & (1<<RXC)) );

    /* Get and return received data from buffer */
    return UDR;
    }
      //fungsi untuk recive atau penerimaan dat
```

C. Unjuk Kerja

Unjuk kerja dari alat ini merupakan penerapan dari diagram alir. program utama, yang ada pada chip IC utama yang akan mengarahkan gerak robot sesuai program. Robot saat ini diprogram hanya untuk beberapa gerakakan.

1. Posisi Berdiri

Posisi berdiri dibutuhkan saat *star-up* untuk menunggu perintah dijalankan dengan menekan tombol *start*. Kondisi ini masing-masing servo pada hanya berada pada satu sudut saja. Sudut yang dipakai pada masing-masing kaki sama dari servo luar (x1) 25^0 servo tengah (x2) 165^0 dan servo dalam (x3) 90^0. Berikut ini merupakan gambar kondisi robot berdiri.



Gambar 26.Robot posisi berdiri.

2. Posisi Bergerak

Posisi bergerak merupakan kondisi dimana robot sedang berjalan berpindah tempat. Kondisi bergerak mempunyai beberapa gerakan berikut ini:

a. Gerak Maju

Gerak maju merupakan dimana robot mulai bergerak berjalan kedepan sesuai dengan depannya robot. Kondisi ini servo bergerak secara keseluruhan secara bergantian sesuai program yang telah dibuat. Berikut ini pergerakan dari servo saat bergerak maju:

Tabel 14. Pergerakan gerak maju

No	Kaki	Alamat	Sudut(^{0})
1	A Depan Kanan	A1	35, 55
		A2	135, 165
		A3	90, 135
2	B Belakang kanan	B1	40, 50
		B2	165, 125
		B3	55, 90
3	C Belakang Kiri	C1	40, 50
		C2	165, 125
		C3	90, 135
4	D Depan Kiri	D1	35, 55
		D2	135, 165
		D3	55, 90

Data Tabel 14. menunjukan gerak yang terjadi pada masing-masing servo. Servo pada alamat A1 yang merupakan servo paing luar bergerak 35^0 dan 55^0. Pembacaan tabel selanjutnya sama.

b. Gerak Belok

Gerak belok merupakan dimana robot bergerak berjalan berbelok sesuai dengan arah belok kanan atau kiri robot. Kondisi ini servo bergerak secara keseluruhan secara bergantian sesuai program yang telah dibuat. Gerakan servo pada saat berbelok berbeda dengan gerak maju. Berikut ini pergerakan dari servo saat bergerak berbelok:

Tabel 15. Pergerakan Belok

No	Kaki	Alamat	Sudut(^{0})
1	A Depan Kanan	A1	35, 55
		A2	135, 165
		A3	90, 135/90,100
2	B Belakang kanan	B1	40, 50
		B2	165, 125
		B3	55, 90/80, 90
3	C Belakang Kiri	C1	40, 50
		C2	165, 125
		C3	90, 135/90, 100
4	D Depan Kiri	D1	35, 55
		D2	135, 165
		D3	55, 90/80, 90

Pergerakan yang terjadi pada servo pada kondisi berbelok berbeda dengan kondisi servo bergarak maju, karena saat kondisi berbelok kanan/kiri kaki pada sisi gerak belok dikurangi sudut geraklangkahnya agar dapat berbelok.

Data Tabel 15. menunjukan gerak yang terjadi pada masing-masing servo. Pergerakan servo sebagian besar sama kecuali servo

yang berada alamat A3, B3, C3 dan D3 sebagai penentu arah gerak robot.

Sistem dapat bekerja dengan langkah pertama yang dilakukan adalah menghubungkan sistem kesumber tegangan yang berupa baterai dengan menekan tombol power. Robot akan berdiri stelah tombol start di tekan baru robot akan bergerak. Tekan tombol power untuk mematikanya.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan terhadap "*Servo Kontroler Sebagai Penggerak Kaki Robot dengan Komunikasi Serial Berbasis Mikrokontroler ATmega16*" maka dapat disimpulakan:

1. Perangkat keras Servo Kontroler Sebagai Penggerak Kaki Robot dapat diwujudkan dengan menggabungkan beberapa komponen dari rangkaian yang telah dibuat diantaranya: rangkaian catu daya, servo kontroler, servo. Setiap elemen merupakan komponen untuk mewujudkan servo kontroler oleh chip IC Mikrokontroller ATMEGA16 sebagai kendali
2. Perangkat lunak (software) yang diaplikasikan dalam sistem ini adalah program yang dibangun dengan bahasa pemrograman C. Aplikasi yang digunakan adalah *Compiler* CV AVR. Hasil program yang telah di-*compile* digunakan dua buah mikrokontroler berkomunikasi secara serial USART, antara Chip mikrokontroler utama dan mikrokontroler servo. Berdasarkan pengujian perangkat lunak ini dapat bekerja dengan baik untuk mengerakan kaki-kaki robot berkaki.
3. Unjuk kerja penggerak kaki robot berbasis ATmega16 secara kesuluruan sudah sesuai dengan fungsi yang terapkan, yaitu saat

tombol *start* di tekan maka robot akan berdiri dengan baik dan akan mulai berjalan setelah beberapa waktu.

B. Keterbatasan Alat

Alat yang dibuat masih mempunyai banyak keterbatasan, beberapa keterbatasan antara lain:

1. Servo yang digerakan belum bergerak 180^0 sesuai kemapuan maksimalnya.
2. Gerakan lebih lambat jika dibandingkan dengan satu chip. Gerakan cukup cepat untuk digunakan pada robot berkaki.

C. Saran

Dalam pembuatan Proyek Akhir ini tentunya terdapat kekurangan, sehingga diperlukan penggembanan lebih lanjut. Saran membangun yang dibutuhkan untuk menyempurnakan Proyek Akhir ini, antara lain sebagai berikut:

1. Perubahan sistem pergerakan servo dengan PWM yang lebih tepat untuk menggerakan servo.
2. Penggunaan sensor untuk kecerdasan robot.

DAFTAR PUSTAKA

Atmel Corporation. 2003.*8-bit Microcontroller with 16K Bytes In-System Programmable Flash ATmega16 ATmega16L Preliminary*. Diambil pada tanggal 04 Oktober 2012, dari : http:// www.atmel.com.

Akbarulhuda. 2010. Mengenal Motor Servo. Diambil tanggal 2 Oktober 2012 dari (http://akbarulhuda.wordpress.com/2010/04/01/mengenal-motor-servo/)

Akhmad Zainuri, ST, 2011. Definisi Komunikasi Data. Diambil 10 Mei 2013 dari (http://belajarkomputerlagi.blogspot.com/2013/01/definisi-komunikasi-data.html)

Arif Zakarya, 2012, Komunikasi Seral Mikrokontroler. Diambil tanggal 10 Mei 2013 dari http://arifzakariya.blog.ugm.ac.id/2012/01/09/komunikasi-serial-mikrokontroler/

Christianto. 2011. UBEC. Diambil tanggal 15 Januari 2012 dari (http://christianto.tjahyadi.com/elektronika/ubec.html)

Lis Lestari. 2012. Pengertian Baterai dan Sejarahnya. Diambil tanggal 12 jan 2013 dari (http://kamusq.blogspot.com/2012/09/baterai-adalah-pengertian-dan-sejarah.html)

Polong. 2008. USART pada Mikrokontroler AVR. Diambil tanggal 10 Januari 2012 dari (http://polong.wordpress.com/2008/04/24/USART -pada-mikrokontroler-avr/)

Tim penyusun. 2011. Pendoman proyek Akhir. Fakultas Teknik UNY

Unggul Prasetya.2010. Kapasitor. Diambil 10 Mei 2013 dari, http://fisika12.blogspot.com/2010/08/kapasitor_19.html

Lampiran

Hitec Digital Servos Operation and Interface

Revision 0.4

1st June 2006

Introduction

This is based on the data gathered from the HFP-10 and the following servos:

HS-5475HB (Firmware Version 1.03)

HS-5245MG (Firmware Version 1.04)

HSR-5995TG (No Firmware Version given)

HS-5645MG (Firmware Version 1.04)

This document does not cover the HSR 8498HB. This robot servo is quite different. It is not compatible with HFP-10. It uses a different electrical interface to support multi-drop bus operation.

Inside the Hitec Digital Servo

This is mainly based on the HS-5475HB. The HS-5245MG looks similar, but being smaller the circuit is on two sandwiched boards, so I can't be sure.

The HSR-5995TG and the HS-5645MG have an ATmega8 processor instead of the AT90LS4433. The processors are pin compatible.

Processor

The HS-5475HB servo is based on the Atmel AT90LS4433 processor, which has:

2.7V to 6V
4K Flash not reprogrammable in servo
128 bytes RAM
256 bytes EEPROM
UART
10 bit ADC
4 MHz Clock

The processor data sheet is available on the Atmel web site under mature devices:

http://www.atmel.com/dyn/products/product_card.asp?part_id=1998

I didn't measure the clock; I think it is probably 4MHz

The HSR-5995TG and HS-5645MG are based on the ATmega8 processor which has:

2.7V to 5.5V
8K Flash self reprogrammable
1024 bytes RAM
512 bytes EEPROM
UART
10 bit ADC

4 MHz Clock

The processor data sheet is available on the Atmel web site:

http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf

There does not appear to be any reprogramming of Flash by the HFP10 at least in version 1.02 software

Mechanics

The mechanics are conventional, motor, gear-train, and feedback pot (4K7Ohm). I didn't count the gear ratio.

Obviously the gears are of different material

Surprisingly the same pot seems to be used on all servos, and the quality while maybe acceptable on a $40 servo, is low for a $150 servo

Physical end stops are on output shaft at just over 180 degrees (198 degrees I think)

Mechanical specs (torque, speed, etc.) are on Hitecrcd web page:

http://www.hitecrcd.com
And elsewhere.

Motor Driver

The motor is driven by a pair of Vishay Siliconix Si9958 Complimentary MOSFETS in Bridge configuration.

http://www.vishay.com/doc?70141

The Bridges are driven from transistors and linked together. This has the unfortunate situation that the bridge cannot be turned completely off. So the motor is either driven or braked while power is applied to servo. No current or temperature monitoring is performed on motor.

The MOSFETS heat up pretty quick under stall conditions, and are close to their maximum ratings on the higher power servos.

Control Interface

The control interface is connected to the processor Port D0 and Port D1 pins which are also the Uart TX and RX pins. The Port B0/ICP pin is also connected to the UART RX pin, and likely measures the pulse width in PWM mode

The control input is protected by a 1K resistor and what appears to be a 4V7 zener on the input. The control then goes to what appears to be the emitter of an NPN transistor. The collector goes to the RX input of the processor, and the base is driven by the processor TX through a 10K resistor.

When driven in PWM mode the swing on control is 0 to 4.8 Volts

For serial mode a light pulldown (I used 47K), is needed. The control input is pulled up by the transistor. Too high a value for pulldown prevents a good low, too low reduces the high from the servo.

The HSR-5995TG has a couple of extra diodes on the TX pin; this may be for higher speed operation in HMI.

Other bits

Processor power is regulated to 3.1 Volts by regulator IC.

4K7 Feedback pot feeds to ADC input on processor between Vref and ground.

Servo programmer HFP-10

I didn't spend too much time looking at the programmer. Mine has version 1.02 software.

I made a simple board with two 3 pin headers, and a link on the control pin, to enable isolation.

I used a Tek 318 logic analyser in serial mode on the control Pin. To determine if it was the servo or the HFP sending data, I put fine wires direct to the processor RX and TX pins on the servo.

Servo in PWM Mode

I didn't check out the servo in PWM mode.

I would like to know how the servo reacts to pulses outside the 900 – 2100 uSec range. Do they respond to 800, as if 900, or not at all?

Also do the pulses in the HMI document have any effect?
http://www.hitecrobotics.com/Tony%20information/HMI%20Protocol.pdf

The HSR-5995TG says position feedback on the box

Servo Electrical Interface in Serial Mode

Electrical

I used the following interface from the PC.

This pulls down the control input, allows the PC to send, receives data (including what is sent), and controls the power up to the servo.

State of control input on power up seems to determine the mode (PWM or serial). I do not know if you can send PWM in serial mode. I do know you have to follow the right sequence to get into serial mode.

The data is inverted (Space = High), so can drive a PIC direct on RXD. The TXD must be open collector or open source driver, which inverts, so best to use a processor where TXD can be inverted.

Lampiran Program

Program Servo Kontroler

```
#include <mega16.h>
#include <delay.h>

#define servo_on1 PORTB
#define servo_on2 PORTA
#define servo_on3 PORTC

#define jml_servo 24

#define delay_servo //delay_us(500);

#define F_CLOCK 12000000L
#define BAUD 9600L

#include "usart.c"

unsigned int n;
unsigned int alamat;
int servo[jml_servo];
int power,indek_servo,set_servo[jml_servo],speed[jml_servo];
unsigned char data_diterima=0;
```

```
// debug ini untuk mengurutkan alamat sesuai hardware..
unsigned int debug[jml_servo]= {0,1,8,9,10,11,12,13,       //0=0,2=8
                        //0 1 2 3 4  5   6  7
                         14,15,2,3,4,16,17,18,
                        //8 9 1o 11 12 13 14 15
                          19,20,21,22,23,5,6,7};
                        //16 17 18 19 20 21 22 23



long int set_sudut (int sudut){
   long out;
   out=(long)432*sudut;
   out=out/180;
   return out;
}



// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{

  data_diterima =1 ;
 }

void main(void)
```

```
{
   PORTA=0x00;
   DDRA=0xff;
   PORTB=0x00;
   DDRB=0xFF;
   PORTC=0x00;
   DDRC=0xFF;
   PORTD=0x00;
   DDRD=0x00;

  /*

   while(1){

      PORTA=0x00;
      PORTC=0x00;
      PORTB=0x00;
      delay_ms(20);
      PORTB=0xff;
      PORTC=0xFF;
      PORTA=0XFF;
      delay_us(1300);

   }   */
```

```
// External Interrupt(s) initialization
GICR|=0x40;   // INT0: On
MCUCR=0x02; // INT0 Mode: Falling Edge
GIFR=0x40;   // INT1: Off
ACSR=0x80;



usart_init(MYUBRR);



// Global enable interrupts
#asm("sei")

servo_on1 = 0x00;
servo_on2 = 0x00;
servo_on3 = 0x00;
power=0;

 indek_servo=0;
while (1)
{



  // for (indek_servo=0;indek_servo<jml_servo;indek_servo++){
```

```
if((indek_servo < 8) && power) {
   servo_on1 = 1 << indek_servo;
   for(n=0; n < servo[indek_servo]; n++){
      delay_us(1);
    };
   delay_us(595);
   servo_on1 =0x00;
   delay_servo;

}else if ((indek_servo < 16) && power){
   servo_on2 = 1 << (indek_servo-8);
   for(n=0; n < servo[indek_servo]; n++){
      delay_us(1);
    };
   delay_us(595);
   servo_on2 =0x00;
   delay_servo;

}else if ((indek_servo < 24) && power){
  servo_on3 = 128 >> (indek_servo-16);
   for(n=0; n < servo[indek_servo]; n++){
      delay_us(1);
    };
   delay_us(595);
   servo_on3 =0x00;
```

```
   delay_servo;
}

if(set_servo[indek_servo]>servo[indek_servo]){
   servo[indek_servo]+=speed[indek_servo];
   if ( servo[indek_servo]>set_servo[indek_servo]){
      servo[indek_servo]=set_servo[indek_servo];
      }
   }
else if(set_servo[indek_servo]<servo[indek_servo]){
   servo[indek_servo]-=speed[indek_servo];
   if ( servo[indek_servo]<set_servo[indek_servo]){
      servo[indek_servo]=set_servo[indek_servo];
      }
   }
if (data_diterima){
   alamat=usart_receive();
   if (alamat==66) power=0;
   else if (alamat==88) power=1;
   else alamat=debug[alamat];

   set_servo[alamat]=set_sudut(usart_receive());
   speed[alamat]=usart_receive();
   data_diterima = 0;
}
```

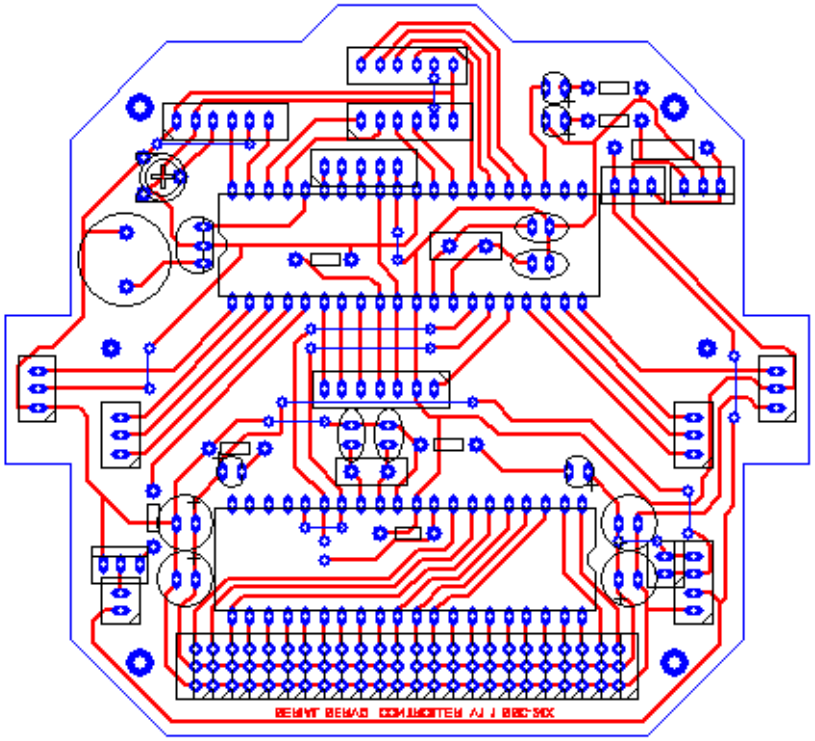
```
    //}

        indek_servo+=1;
        if(indek_servo>=jml_servo){
           indek_servo=0;
        }

  }
```

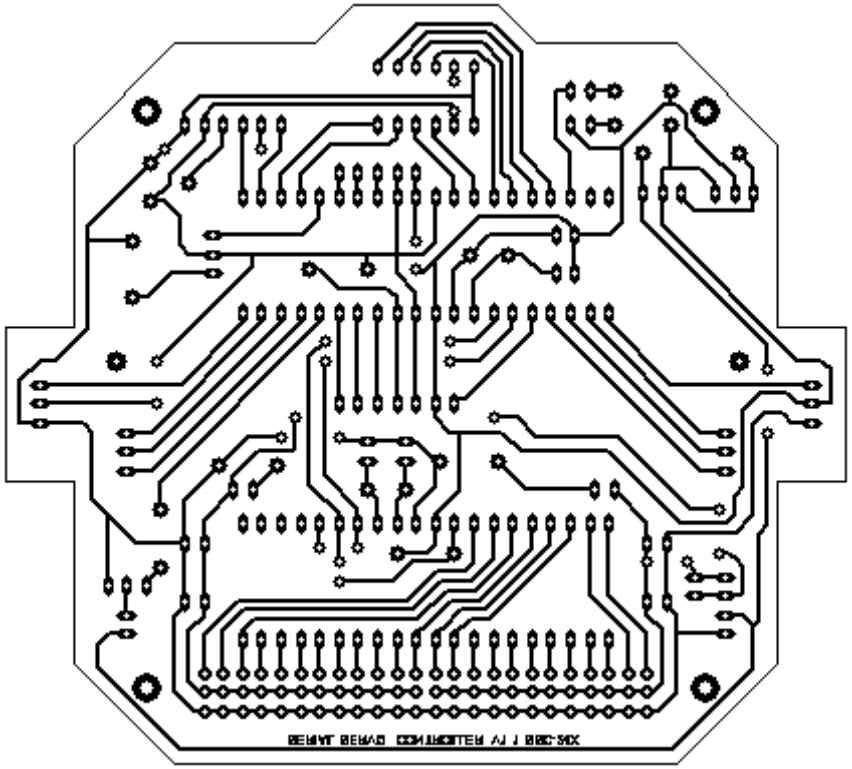
Program Usart.c

```
#ifndef _STDIO_INCLUDED_
  #include <stdio.h>
#endif
#ifndef _DELAY_INCLUDED_
  #include <delay.h>
#endif
#ifndef F_CLOCK
  #error F_ClOCK belum didefinisikan..
#endif
#ifndef BAUD
  #error BAUD belum di definisikan..
#endif
#define MYUBRR F_CLOCK/16/BAUD-1
#ifndef TXEN
  #define TXEN 3
```

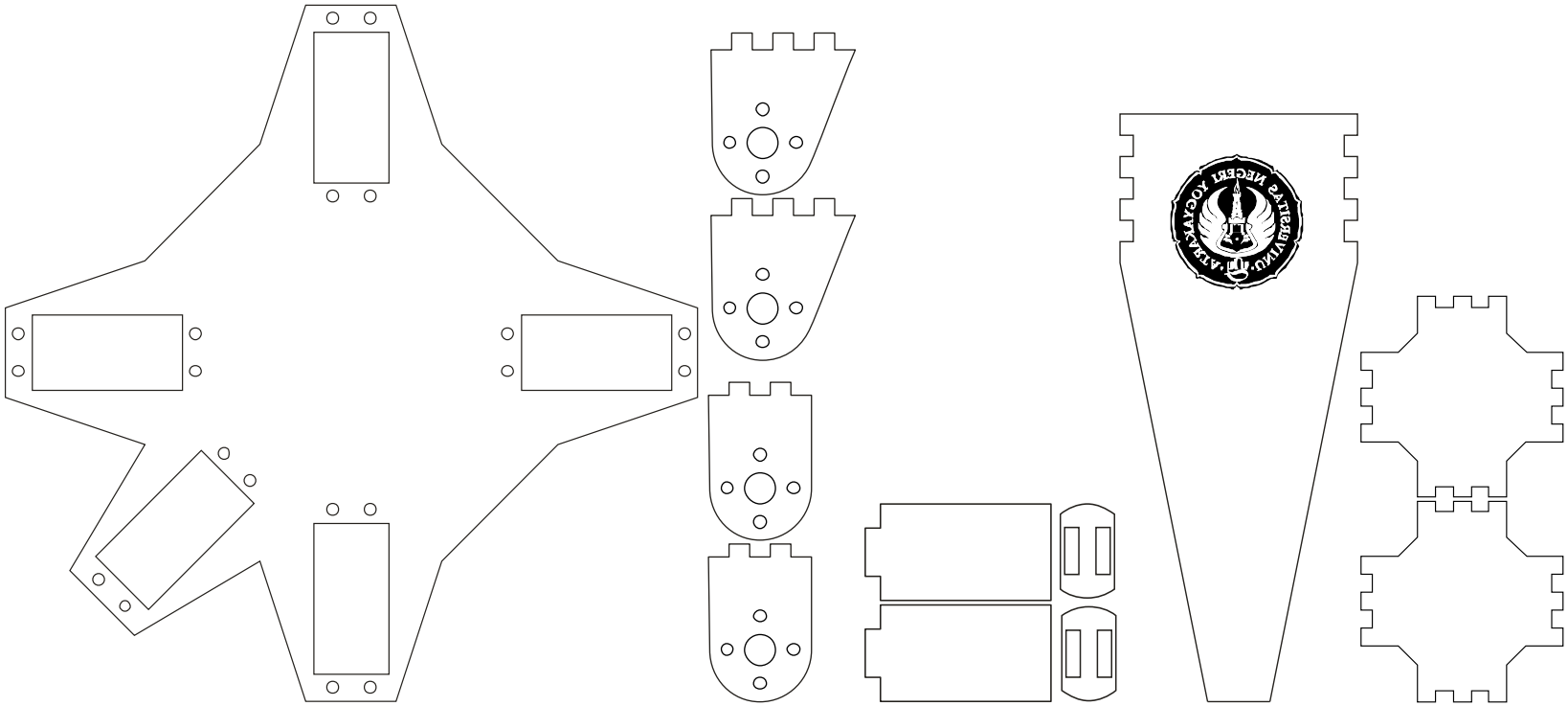
```
   #define RXEN 4
   #define URSEL 7
   #define USBS 3
   #define UCSZ0 1
   #define UDRE 5
   #define RXC 7
#endif

#pragma used+
void usart_init( unsigned int ubrr);
void usart_transmit( unsigned char data );
unsigned char usart_receive( void );
#pragma used-
void usart_init( unsigned int ubrr)
{
   /* Set baud rate */
   UBRRH = (unsigned char)(ubrr>>8);
   UBRRL = (unsigned char)ubrr;
   /* Enable receiver and transmitter */
   UCSRB = (1<<RXEN)|(1<<TXEN);
   /* Set frame format: 8data, 2stop bit */
   UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
   }
void usart_transmit( unsigned char data )
{
```

```
   /* Wait for empty transmit buffer */
   while ( !( UCSRA & (1<<UDRE)) );

   /* Put data into buffer, sends the data */
   UDR = data;
   }
unsigned char usart_receive( void )
{
   /* Wait for data to be received */
   while ( !(UCSRA & (1<<RXC)) );

   /* Get and return received data from buffer */
   return UDR;
   }
```

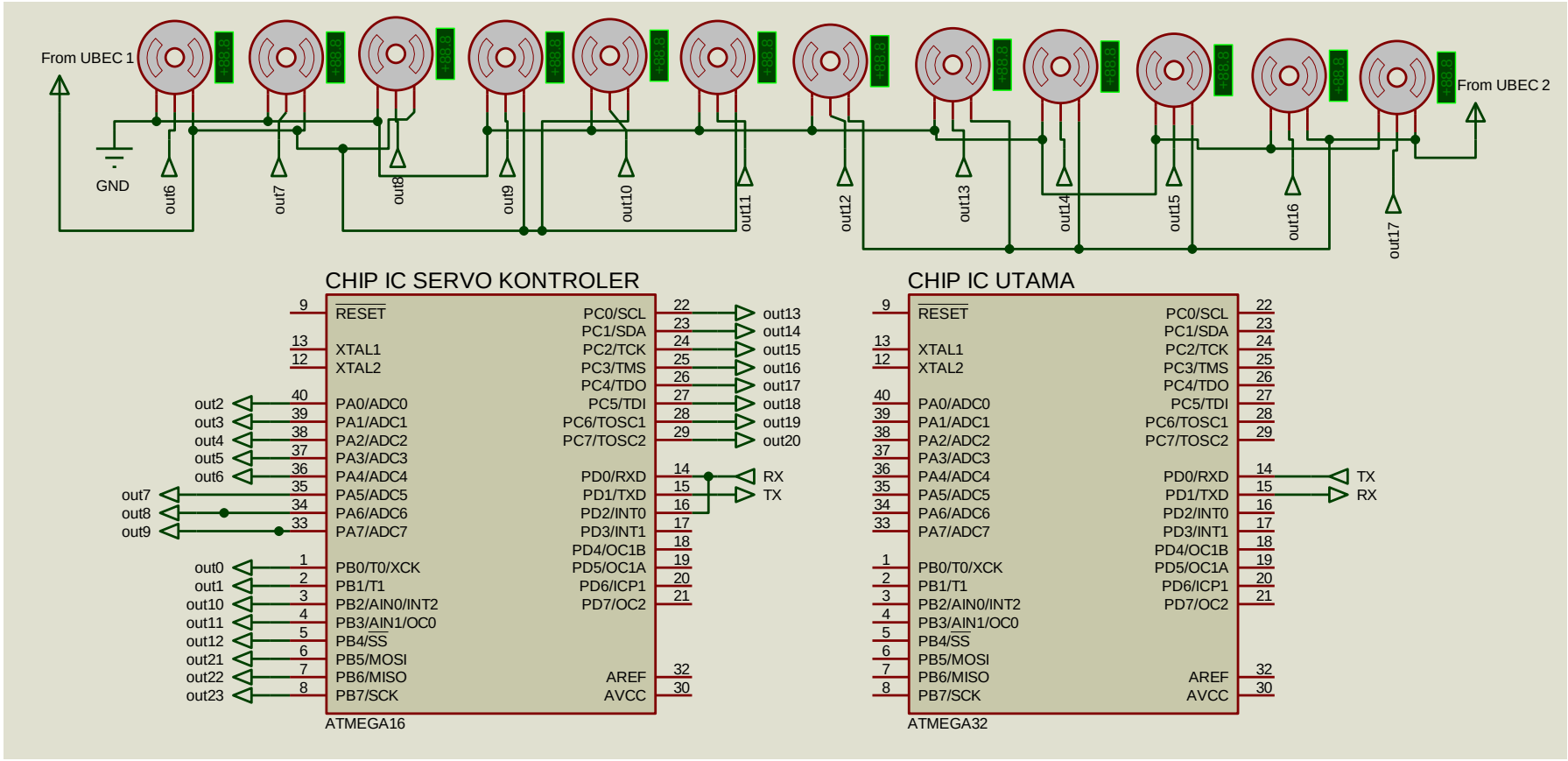




LAYOUT DAN PCB KONTROL KECEPATAN			KETERANGAN
			A4 / No.1
FT UNY	SKALA -	DIG: FEBRI C.P	
	DIP. Dr.Eko M	DIST. Dr.Eko M.	NIM. 09507131017



LAYOUT DAN PCB KONTROL KECEPATAN			KETERANGAN	
FT UNY	SKALA -	DIG: FEBRI C.P	A4	No.1
	DIP. Dr.Eko M	DIST. Dr.Eko M.	NIM. 09507131017	



LAYOUT DAN PCB KONTROL KECEPATAN			KETERANGAN
			A4 / N0.2
FT UNY	SKALA -	DIG: FEBRI C.P	
	DIP. Dr.Eko M	DIST. Dr.Eko M.	NIM. 09507131017

Panduan Pengoprasian Alat

A. Pasangkan baterai pada soket yang ada pada robot.
B. Tekan tomol power ke posisi on untuk menghidupkan.
C. Biarkan robot berdiri dengan semburna dulu.
D. Tekan tombol start untuk mulai menggerakan robot.
E. Robot akan bergerak sendiri dan baterai dibawa robot.
F. Tekan tombol power ke posisi off untuk mematikan.