BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian serupa yang pernah dilakukan oleh Agus Sulistiyo dan kawan-kawan dalam jurnal Emitor dengan judul "KWH Meter Digital Terkoneksi Personal Computer (PC) Berbasis Mikrokontroler ATMega16". Alat ini dibuat sebagai solusi terhadap suatu permasalahan yang sering terjadi dilingkungan sekitar khususnya pada lingkungan rumah tangga terkait masalah dalam pembayaran tagihan listrik. Hal ini dikarenakan seringnya pelanggan membayar tagihan listrik yang tidak sesuai dengan pemakaian listrik mereka. Hal ini sangat merugikan pelanggan ketika energi yang dikonsumsi dengan yang dibayarkan pada penyedia layanan listrik ternyata berbeda. Dari penelitian yang telah dilakukan hasil pengujian menunjukan bahwa sensor tegangan pada pengukuran 218,4 volt rms *Power Quality Analyzer* diwakili dengan tegangan analog 3,798 volt yang pada akhirnya menunjukan resolusi sensor sebesar 17,5 mv/volt rms. Pengukuran ini dilakukan pada beban induktif dengan beban berupa kipas angin 60 watt, motor mesin jahit 120 watt serta motor mesin jahit 250 watt pada tegangan kerja 220 volt dengan frekuensi 50 hz (Agus, 2010).

Pengujian yang telah dilakukan menunjukkan bahwa setiap kenaikan beban sebesar 1 watt akan menaikkan arus sebesar 0,044 ampere. Korelasi ini ditunjukkan oleh persamaan y = 0,625x – 0,669 dengan y arus yang dihasilkan (Ampere) serta x merupakan 9beban resistif (watt) yang diujikan. Perhitungan energi aktif diambil dari hasil pengukuran daya aktif tiap beban yang diukur pada alat ukur referensi *Power Quality Analyzer*dengan asumsi tiap waktunya mengalami perpindahan energi yang sama dari daya jala-jala listrik ke dalam beban yang diukur. Hasil perhitungan dibandingkan terhadap tampilan alat ukur KWH meter digital tiap satuan waktu yang sama. Pengukuran dilakukan selama satu menit (60 detik) dengan pengukuran daya dalam satuan watt sehingga diperoleh energi tiap watt jam sebesar 0,00278 kali daya terukur (Agus, 2010).

Peneltian kedua yang serupa yang telah dikakukan oleh Sulistyowati dalam jurnal ELTEK dengan judul "Audit Energi untuk Efisiensi Pemakaian Energi Listrik". Audit energi adalah teknik yang dipakai untuk menghitung besarnya konsumsi energi pada bangunan gedung dan mengenali cara-cara untuk penghematannya. Dari data pengukuran sampel daya didapatkan karakteristik selama 2 hari, dimana tren pemakaian daya pada saat beban puncak jam 07:34 sampai dengan jam 14:00 wib sudah mencapai beban maksimal (98,88 %) atau mendekati daya tersambung. Dengan pembebanan yang sudah maksimal terhadap trafo terpasang dapat memperpendek usia pemakaian trafo itu sendiri (Sulistyowati, 2012).

Berdasarkan data pengukuran didapatkan ketidakseimbangan beban dengan range ketidakseimbangan antara 0,31% sampai dengan 45,81%. Menurut standard IEC ketidak seimbangan beban yang diijinkan adalah 5%, dengan tingginya ketidakseimbangan beban berpengaruh terhadap besarnya arus netral, dimana arus netral yang besar mengakibatkan losses bertambah dan kualitas tenaga yang rendah sehingga berpengaruh terhadap kualitas sistem penyaluran tenaga listrik. Dari audit diperoleh hasil bahwa biaya perbaikan dan biaya operasional bulanan untuk kerja paralel PLN dan genset lebih murah, dibandingkan dengan tambah daya. Dibutuhkan biaya investasi baru yang lebih besar untuk tambah daya PLN (Sulistyowati, 2012).

Penelitian ketiga yang serupa yang telah dilakukan oleh Yadi Mulyadi dan kawan-kawan dalam jurnal UPI dengan judul "Analisis Audit Energi untuk Pencapaian Efisiensi Penggunaan Energi di Gedung FPMIPA JICA Universitas Pendidikan Indonesia". Alat ini dibuat untuk mengatasi masalah yang terjadi di lingkungan Universitas Pendidikan Indonesia terkait masalah pembayaran tagihan listrik. Alat ini digunakan sebagai solusi untuk penghematan pemekaian energi listrik di lingkunagn Universitas Pendidikan Indonesia. Sistem kelistrikan di gedung FPMIPA JICA Universitas Pendidikan Indonesia merupakan sistem interkoneksi antara suplaier daya dari PLN dan suplaier daya dari genset. Suplai daya dari PLN di salurkan melalui panel MVMDP dan diteruskan ke transformator, sistem ini merupakan sistem tegangan menengah (20 KV, 3 Phasa,

50Hz). Dari transformator utama menuju panel LVMV (220/380V), kemudian disalurkan ke masing – masing sub panel distribusi. Suplai daya dari PLN untuk gedung FPMIPA JICA Universitas Pendidikan Indonesia adalah sebesar 865 KVA, dengan golongan tarif listrik dan beban tersambung termasuk pada klasifikasi S3 (pelayanan sosial besar tegangan menengah).

Selain itu, gedung FPMIPA JICA Universitas Pendidikan Indonesia juga disuplai genset dengan kapasitas 250 KVA. Suplai daya dari genset disimpan sebagai cadangan pasokan listrik apabila terjadi pemadaman oleh PLN. Untuk mengetahui kualitas daya dari gedung FPMIPA JICA Universitas Pendidikan Indonesia maka dilakukanlah pengukuran pada LVMDP (Low Voltage Main Distribution Panel) sebagai pusat penyalur daya listrik menuju SDP (Sub Distribution Panel) kemudian disambungkan ke beban sesuai kebutuhan. Alat yang dipakai dalam pengukuran ini adalah Power Quality Analyzer. Dari audit energi awal yang telah dilaksanakan dapat disimpulkan bahwa tingkat efisiensi konsumsi energi listrik pada gedung FPMIPA JICA Universitas Pendidikan Indonesia berada pada rata-rata 60119.75 Kwh/m2/tahun. Dan dengan nilai IKE (Intensitas Konsumsi Energi) 3.77 per-tahun. Dengan demikian bisa di katakan nilai IKE sangat efisien untuk sebuah gedung katagori ber-AC (Mulyadi, 2013).

Dari beberapa penelitian yang telah dilakukan sebelumnya, terdapat perbedaan antara alat yang telah dibuat pada penelitian sebelumnya dengan alat yang dibuat oleh penulis. Perbedaannya yaitu pada alat yang dibuat sebelumnya hanya dpat mengetahui besar pemakaian daya listrik yang digunakan dalam KWH, sedangkan alat yang dibuat oleh penulis selain dapat mengetahui besar pemakaian daya listrik dalam KWH, alat ini juga dapat mengetahui berapa besar biaya yang harus dikeluarkan berdasarkan pemakaian daya listrik yang digunakan.

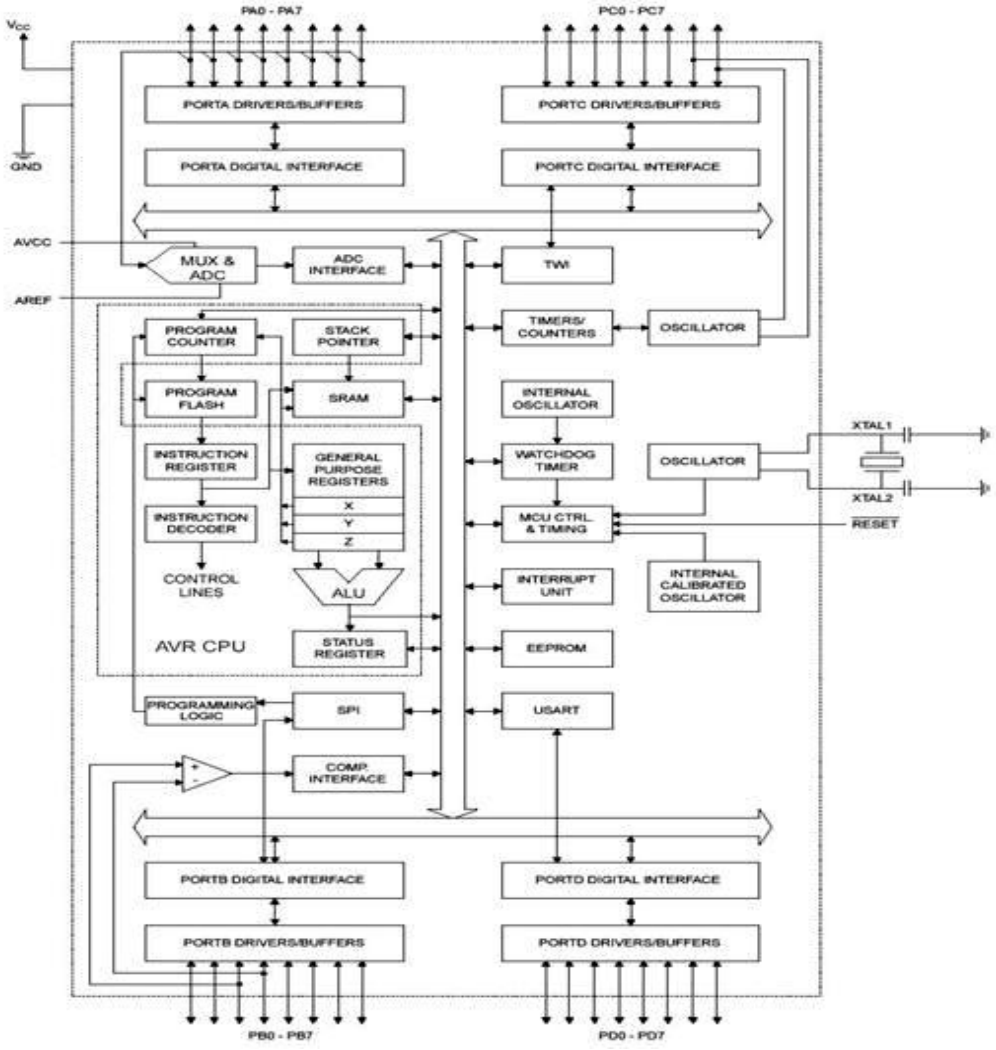
2.2 Mikrokontroler ATMEGA8535

Mikrokontroler merupakan sebuah sistem komputer yang mempunyai satu atau beberapa tugas yang spesifik, berbeda dengan PC yang memiliki beragam fungsi (Marethania, 2011). Atmega8535 merupakan IC CMOS 8-bit yang memiliki daya rendah dalam pengoperasiannya dan berbasis arsitektur *RISC AVR.*

Atmega8535 dapat mengeksekusi satu instruksi dalam sebuah siklus *clock*, dapat mencapai 1 *MIPS* per Mhz, sehingga para perancang dapat mengoptimalkan penggunaan daya rendah dengan kecepatan tinggi (Marethania, 2011).

2.2.1 Arsitektur ATMega8535

Arsitektur ATMega8535 ini dapat dilihat pada blok diagram ATMega8535 seperti yang terlihat pada gambar 2.1 dibawah ini.



Gambar 2.1 Blok Diagram ATMEGA8535

Dari gambar tersebut dapat dilihat bahwa ATMega8535 memiliki bagian sebagai berikut (Lingga, 2006):

1. Saluran I/O sebanyak 32 buah, yaitu Port A, Port B, Port C dan Port D

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

2.2.2 Fitur ATMega8535

Kapabilitas detail dari ATMega8535 (Lingga, 2006) adalah sebagai berikut

1. Sistem mikroprosessor 8 bit berbasis *RSIC* dengan kecepatan maksimal 16 MHz
2. Kapabilitas memori *flash* 8 kb, SRAM sebesar 512 byte dan EEPROM (*Electrically Erasable Programmable Read Only Memory*) sebesar 512 byte
3. ADC internal dengan fidelitas 10 bit sebanyak 8 *channel*
4. Portal komunikasi serial (USART) dengan kecepatan maksimal 2,5 Mbps
5. Enam pilihan mode *sleep* menghemat penggunaan daya listrik

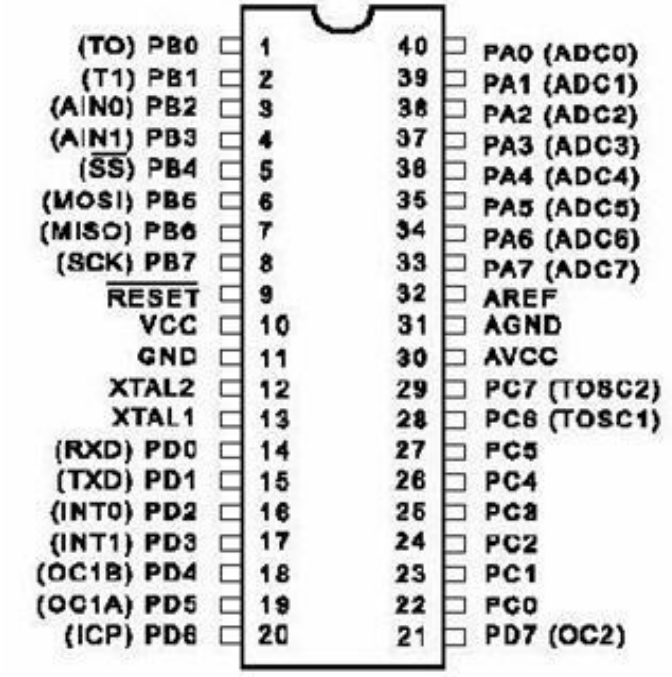
2.2.3 Konfigurasi Pin ATMega8535

Konfigurasi pin ATMega8535 bisa dilihat pada Gambar 2.2. pada gambar tersebut dapat dijelaskan secara fungsional konfigurasi pin ATMega8535 sebagai berikut (Lingga, 2006):

1. VCC merupakan pin yang berfungsi sebagai pin masukan catu daya
2. GND merupakan pin *ground*
3. Port A (PA0...PA7) merupakan pin I/O dua arah dan pin masukan ADC

4. Port B (PB0...PB7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu *Timer/Counter*, komparator analog dan SPI
5. Port C (PC0...PC7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu TWI, komparator analog dan *Timer Oscilator*
6. Port D (PD0...PD7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu komparator analog, interupsi eksternal dan komunikasi serial
7. RESET merupakan pin yang digunakan untuk me-reset mikrokontroler
8. XTAL1 dan XTAL2 merupakan pin masukan *clock* eksternal
9. AVCC merupakan pin masukan tegangan untuk ADC
10. AREF merupakan pin masukan tegangan referensi ADC

Pada gambar 2.2 berikut ini merupakan gambar pin-pin ATMega8535



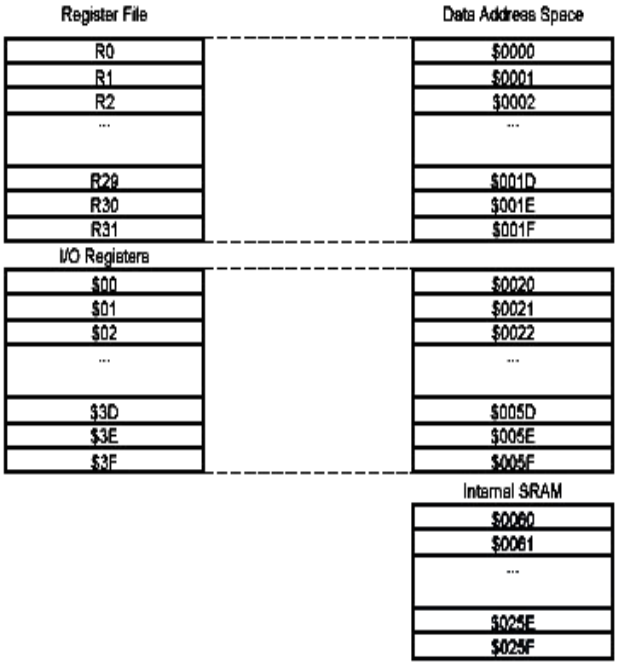
Gambar 2.2 Pin ATMega8535

2.2.4 Peta Memori Mikrokontroler ATMega8535

AVR ATMega8535 memiliki ruang pengalamatan memori data dan memori program yang terpisah. Memori data terbagi menjadi 3 bagian, yaitu 32 buah register umum, 64 buah register I/O dan 512 byte SRAM internal. Register

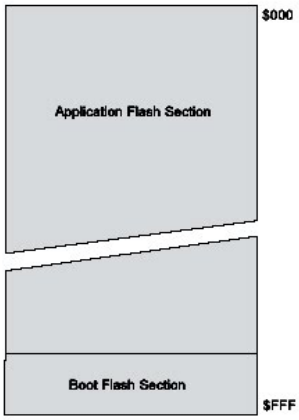
keperluan umum menempati *space* data pada alamat terbawah, yaitu $00 sampai $1F.

Sementara ini, register khusus untuk menangani I/O dan kontrol terhadap mikrokontroler menempati 64 alamat berikutnya, yaitu mulai dari $20 hingga $5F. Register tersebut merupakan register yang khusus digunakan untuk mengatur fungsi terhadap berbagai peripheral mikrokontroler seperti kontrol register, *timer/counter*, fungsi-fungsi I/O dan sebagainya. Alamat memori berikutnya digunakan untuk SRAM 512 byte, yaitu pada lokasi $60 sampai dengan $25F. Konfigurasi memori data dapat ditunjukkan pada gambar 2.3 dibawah ini (Lingga, 2006).



Gambar 2.3 Peta Memori Data ATMega8535

Memori program yang terletak dalam *Flash PEROM* tersusun dalam word atau 2 byte karena setiap instruksi memiliki lebar 16-bit atau 32-bit. AVR ATMega8535 memiliki 4KByte x 16-bit *Flash PEROM* dengan alamat mulai dari $000 sampai $FFF. AVR tersebut memiliki 12-bit *Program Counter* (PC) sehingga mampu mengalamati isi *Flash* (Lingga, 2006). Gambar peta memori program ATMega8535 dapat dilihat pada gambar 2.4 dibawah ini.



Gambar 2.4 Peta Memori Program ATMega8535

Selain itu, AVR ATMega8535 juga memiliki memori data berupa EEPROM 8-bit sebanyak 512 byte. Alamat EEPROM dimulai dari $000 sampai $1FF (Lingga, 2006).

2.2.5 Status Register (SREG) ATMega8535

Status register adalah register berisi status yang dihasilkan pada setiap operasi yang dilakukan ketika suatu instruksi dieksekusi. SREG merupakan bagian dari inti CPU mikrokontroler (Lingga, 2006). Dibawah ini merupakan gambar dari status register ATMega8535.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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	

Gambar 2.5 Status Register ATMega8535

a. Bit 7-I: Global Interrupt Enable

   Bit harus diset untuk meng-enable interupsi. Setelah itu, anda dapat mengaktifkan interupsi mana yang akan anda gunakan dengan cara meng-enable bit kontrol register yang bersangkutan secara individu. Bit akan di *clear* apabila trejadi suatu interupsi yang dipicu oleh *hardware* dan bit tidak akan mengizinkan terjadinya interupsi, serta akan diset kembali oleh instruksi RETI.

b. Bit 6-T: Bit *Copy Storage*

   Instruksi BLD dan BST menggunakan bit-T sebagai sumber atau tujuan dalam operasi bit. Suatu bit dalam sebuah register GPR dapat disalin ke bit T menggunakan instruksi BTS, dan sebaliknya bit-T dapat disalin kembali ke suatu bit dalam register GPR menggunaknan instruksi BLD.

c. Bit 5-H: *Half Carry Flag.*

d. Bit 4-S: Sign Bit

   Bit-S merupakan hasil operasi EOR antara flag –N (negatif) dan flag-V (komplemen dua *overflow*).

e. Bit 3-V: Two's C*omplement Overflow Flag*

   Bit berguna untuk mendukung operasi aritmatika.

f. Bit 2-N: Negative flag

   Apabila suatu operasi menghasilkan bilangan negatif, maka flag-N akan diset.

g. Bit 1-Z: Zero Flag

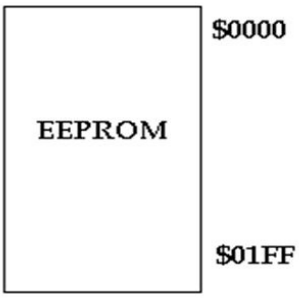
   Bit akan diset akan menghasilkan operasi yang diperoleh adalah nol.

h. Bit 0-C: Carry Flag

   Apabila suatu operasi menghasilkan carry, maka bit akan diset.

2.2.6 EEPROM Data Memory

ATmega8535 memiliki EEPROM 8 bit sebesar 512 byte untuk menyimpan data. Loaksinya terpisah dengan system address register, data register dan control register yang dibuat khusus untuk EEPROM. Alamat EEPROM dimulai dari $000 sampai $1FF seperti yang terlihat pada gambar 2.6 dibawah ini.



Gambar 2.6 EEPROM Data Memory

2.3 *Analog To Digital Converter* (ADC)

ATMega8535 menyediakan fasilitas ADC dengan resolusi 10 bit. ADC ini dihubungkan dengan 8 *channel Analog Multiplexer* yang memungkinkan terbentuk 8 input tegangan *single-ended* yang masuk melalui pin pada Port A. ADC memiliki pin *supply* tegangan analog yang terpisah yaitu AVCC. Besarnya tegangan AVCC adalah ±0.3V dari VCC.

Tegangan referensi ADC dapat dipilih menggunakan tegangan referensi internal maupun eksternal. Jika menggunakan tegangan referensi internal, bisa dipilih *on-chip* internal reference voltage yaitu sebesar 2.56 volt atau sebesar AVCC. Jika menggunakan tegangan referensi eksternal dapat dihubungkan melalui pin AREF.

ADC mengkonversi tegangan input analog menjadi data digital 8 bit atau 10 bit. Data digital tersebut akan disimpan didalam ADC data register yaitu ADCH dan ADCL. Sekali ADCL dibaca, maka akses ke data register tidak bisa dilakukan, dan ketika ADCH dibaca maka akses ke data register kembali *enable* (Heryanto,2008:81).

2.3.1 Inisialisasi ADC

Menurut Heryanto (2008:82) ada beberapa langkah yang harus dilakukan untuk inisialisasi ADC pada mikrokontroler ATMega8535, yaitu penentuan *clock*, tegangan referensi, format data input dan *mode* pembacaan. Inisialisasi ini dilakukan pada register- register berikut :

1. ADMUX

   Pada register ini bertujuan untuk mengatur tegangan referensi yang digunakan ADC, format data output dan saluran ADC. Register ADMUX ini dapat dilihat pada tabel 2.1 dibawah ini.

Tabel 2.1 Register ADMUX

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

a. REF_{0-1} adalah bit- bit pengatur *mode* tegangan referensi ADC. Bit-bit pengatur mode tegangan referensi ini dapat dilihat pada tabel 2.2 dibawah ini.

Tabel 2.2 Bit-Bit pengatur mode tegangan referensi

$REFS_1$	**$REFS_0$**	**Mode Tegangan Referensi**
0	0	Pin Vref
0	1	VCC
1	0	Tidak digunakan
1	1	Vref internal = 2,56 Volt

b. ADLAR adalah bit keluaran ADC.

Jika ADC telah selesai konversi maka data ADC akan diletakkan di 2 register, yaitu ADCH dan ADCL dengan format sesuai ADLAR. Format data ADC dengan ADLAR=0 dapat dilihat pada tabel 2.3 dibawah ini. Sedangkan format data ADC dengan ADLAR=1 dapat dilihat pada tabel 2.4 dibawah ini.

Tabel 2.3 Format data ADC dengan ADLAR = 0

-	-	-	-	-	-	ADC9	ADC8
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

Tabel 2.4 Format data ADC dengan ADLAR = 1

ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADC1	ADC0	-	-	-	-	-	-

c. MUX0-4 adalah bit-bit pemilihan saluran pembacaan ADC.

2. ADCSRA

ADCSRA adalah register 8 bit yang berfungsi untuk melakukan menajemen sinyal kontrol dan status ADC. Register ADCSRA ini dapat dilihat pada tabel 2.5 berikut ini.

Tabel 2.5 Register ADCSRA

ADEN	ADCS	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

a. ADEN merupakan bit pengatur aktivasi ADC. Jika bernilai 1 maka ADC aktif.
b. ADCS merupakan bit penanda dimulainya konversi ADC. Selama konversi berlogika 1 dan akan berlogika 0 jika selesai konversi.
c. ADATE merupakan bit pengatur aktivasi picu otomatis. Jika bernilai 1 maka konversi ADC akan dimulai pada saat tepi positif pada sinyal *trigger* digunakan.
d. ADIF merupakan bit pengatur aktivasi interupsi. Jika bernilai 1 maka interupsi penandaan telah selesai, konversi ADC diaktifkan.
e. $ADPS_{0-2}$ merupakan bit pengatur *clock* ADC. Berikut ini merupakan tabel konfigurasi *clock* ADC

Tabel 2.6 Konfigurasi *clock* ADC

$ADPS_{2-0}$	**Clock ADC**
000-001	f_{osd} 2
010	f_{osd} 4
011	f_{osd} 8
100	f_{osd} 16
101	f_{osd} 32
110	f_{osd} 64
111	f_{osd} 128

3. SFIOR

SFIOR adalah register 8 bit yang mengatur seumber pemicu ADC. Jika bit ADATE pada register ADCSRA bernilai 0 maka $ADTS_{0-2}$ tidak berfungsi. Tabel register SFIOR dapat dilihat pada tabel 2.6 dibawah ini. Sedangkan tabel bit-bit ADATE itu sendiri dapat dilihat pada tabel 2.7 dibawah ini.

Tabel 2.6 Register SFIOR

ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10

Tabel 2.7 Bit-Bit ADATE

ADTS2	ADTS1	ADTS0	Sumber Pemicu
0	0	0	*Free Running Mode*
0	0	1	*Analog Comparator*
0	1	0	*Extenal Interupt Request 0*
0	1	1	*Timer/ Counter 0 Compare Match*
1	0	0	*Timer/ Counter 0 Overflow*
1	0	1	*Timer/ Counter 1 Compare Match B*
1	1	0	*Timer/ Counter 1 Overflow*
1	1	1	*Timer/ Counter 0 Capture Event*

2.4 Code Vision AVR

CodeVisionAVR pada dasarnya merupakan perangkat lunak pemrograman mikrokontroler keluarga AVR berbasis bahasa C. Ada tiga komponen penting yang telah diintegrasikan dalam perangkat lunak ini: *Compiler* C, IDE dan program *generator*. CodeVisionAVR dilengkapi dengan *source code editor*, *compiler*, *linker* dan dapat memanggil Atmel AVR studio dengan debugger nya (Widodo, 2013).

Berdasarkan spesifikasi yang dikeluarkan oleh perusahaan pengembangnya, *Compiler* C yang digunakan hampir mengimplementasikan semua komponen standar yang ada pada bahasa C standar ANSI (seperti struktur program, jenis tipe data, jenis operator, dan *library* fungsi standar berikut penamaannya). Tetapi walaupun demikian, dibandingkan bahasa C untuk aplikasi komputer, *compiler* C untuk mikrokontroler ini memiliki sedikit perbedaan yang disesuaikan dengan arsitektur AVR tempat program C tersebut ditanamkan (*embedded*). Khusus untuk *library* fungsi, disamping *library* standar (seperti fungsi- fungsi matematik, manipulasi *string*, pengaksesan memori dan sebagainya).

CodeVisionAVR juga menyediakan fungsi-fungsi tambahan yang sangat bermanfaat dalam pemrograman antarmuka AVR dengan perangkat luar yang umum digunakan dalam aplikasi kontrol. Beberapa fungsi *library* yang penting diantaranya adalah fungsi-fungsi untuk pengaksesan LCD, komunikasi I2C, IC

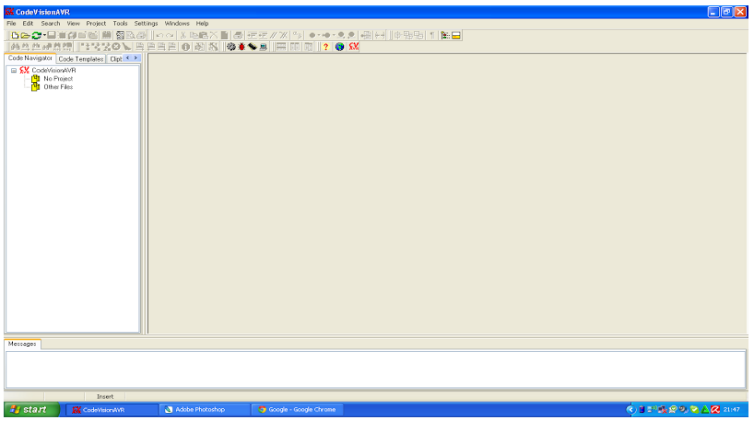
RTC (*Real time Clock*), sensor suhu, SPI (*Serial Peripheral Interface*) dan lain sebagainya. Untuk memudahkan pengembangan program aplikasi, CodeVisionAVR juga dilengkapi IDE yang sangat *user friendly*. Selain menu-menu pilihan yang umum dijumpai pada setiap perangkat lunak berbasis Windows, CodeVisionAVR ini telah mengintegrasikan perangkat lunak *downloader* yang bersifat *In System Programmer* yang dapat digunakan untuk mentransfer kode mesin hasil kompilasi ke dalam sistem memori mikrokontroler AVR yang sedang diprogram (Widodo, 2013).

CodeVisionAVR juga menyediakan sebuah fitur yang dinamakan dengan *Code Generator* atau CodeWizardAVR. Secara praktis, fitur ini sangat bermanfaat membentuk sebuah kerangka program (*template*), dan juga memberi kemudahan bagi *programmer* dalam peng-inisialisasian register-register yang terdapat pada mikrokontroler AVR yang sedang diprogram. Dinamakan *Code Generator*, karena perangkat lunak CodeVision ini akan membangkitkan kode-kode program secara otomatis setelah fase inisialisasi pada jendela CodeWizardAVR selesai dilakukan (Widodo, 2013).

2.4.1 Menjalankan Code Vision AVR

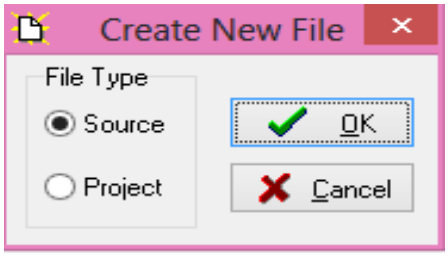
Ada beberapa program yang dapat digunakan sebagai editor dan compiler untuk mikrokontroller AVR, salah satunya yaitu CodeVisionAVR. Untuk menjalankan program tersebut terlebih dahulu menginstall program tersebut. Setelah terinstal maka buka program CodeVision melalui menu Start||All Program||CodeVision| CodeVisionAVR C Compiler atau melalui deskop dengan mengklik lambang codevision. Lambang dari Code Vision AVR dapat dilihat pada gambar 2.7 dibawah ini. Sedangkan gambar tampilan Code Vision AVR saat pertama kali dijalankan dapat dilihat pada gambar 2.8 seperti dibawah ini.

Gambar 2.7 Lambang Code Vision AVR

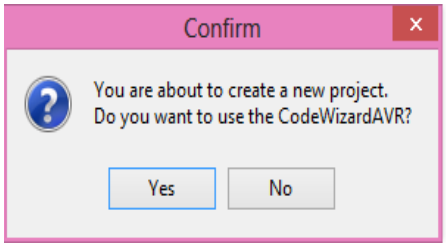


Gambar 2.8 Tampilan Pertama Kali Code Vision AVR Dijalankan

Setelah Code Vision AVR terbuka selanjutnya pilih File|New| pilih File Type dan klik Project lalu klik OK.

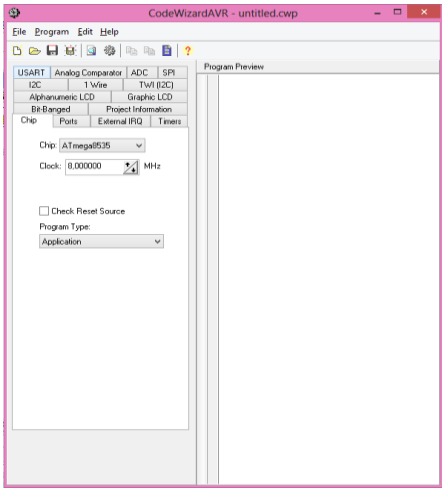


Gambar 2.9 Membuat File Project Baru



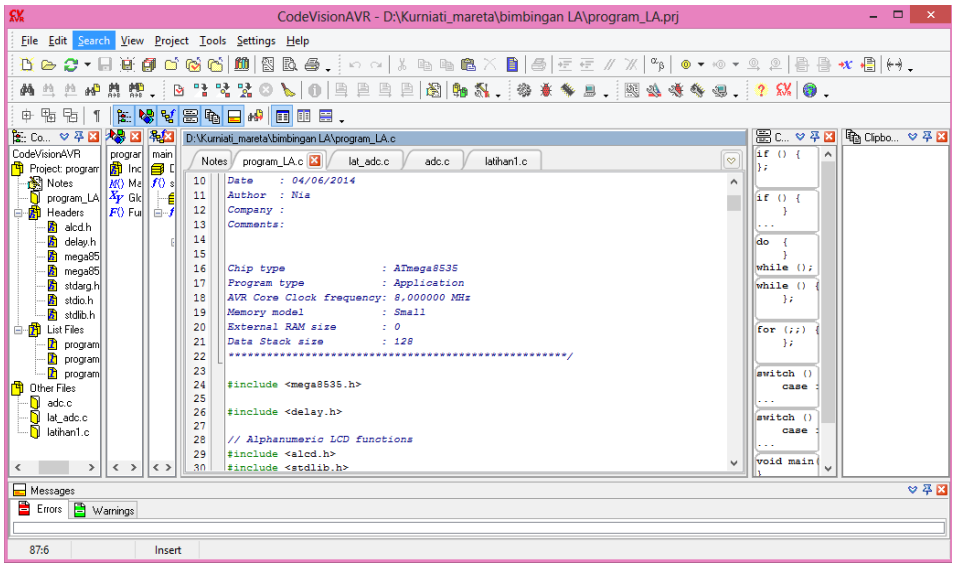
Gambar 2.10 Tampilan Konfirmasi CodeWizardAVR

Pilih tampilan konfirmasi yang menanyakan apakah akan membuat *project* baru atau tidak, kemudian pilih ***yes.*** Setelah itu akan tampil konfigurasi USART, Analog Comparator, ADC, SPI, I2C, 1 wire, 2 wire (I2C), LCD, Bit-Banged, Project Information, chip, Port, External IRQ dan Timer. Selanjutnya tinggal mengatur program yang akan dibuat melalui CodeWizard ini. Misalnya konfigurasi chip yang akan digunakan, pilih chip, lalu isi konfigurasi : chip:ATMega8535, clock: 8.000000 MHz seprti yang terlihat pada gambar dibawah ini.



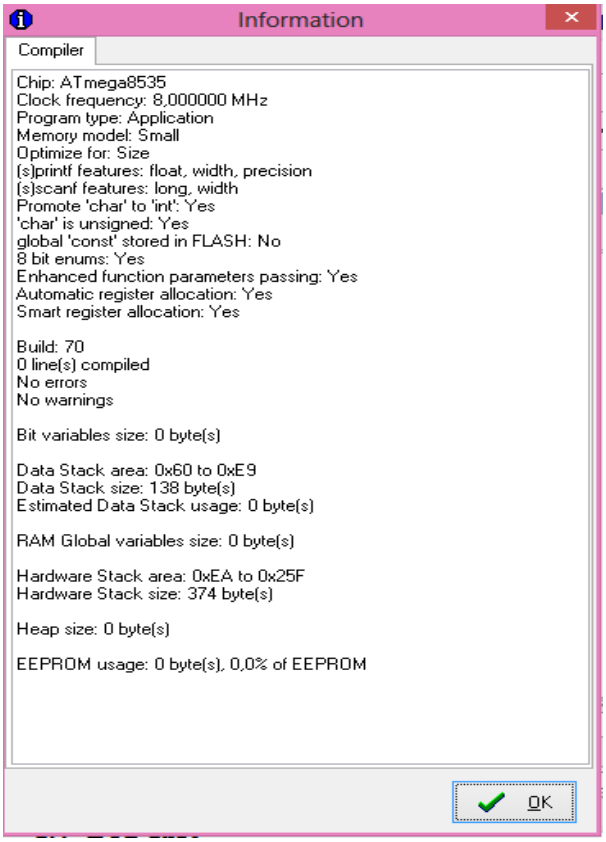
Gambar 2.11 Pengaturan Chip Pada CodeVisionAVR

Kemudian klik PORT untuk untuk memilih dan mensetting PORT yang akan digunakan sebagai input atau output. Setelah pengaturan selesai simpan pengaturan dengan cara pilih file kemudian pilih save as. Berikut ini adalah tampilan setelah menggunakan Code Wizard.



Gambar 2.12 Tampilan Setelah Menggunakan Code Wizard

Jika sudah selesai membuat program ,maka *compile* program , pilih *Project* klik *Compile.* Setelah itu akan muncul tampilan seperti yang ditunjukkan pada gambar 2.13 dibawah ini.



Gambar 2.13 Hasil Proses Kompilasi

2.5 Bahasa Pemrograman C

2.5.1 Pengenalan Bahasa C

C merupakan bahasa universal dalam bidang pegembangan *software* dan banyak digunakan pada mesin-mesin dan komputer. Banyak sekali *software* sistem yang dibuat dengan C karena bahasa C memiliki kemampuan untuk mengakses sistem dari komputer, mulai dari RAM yang sederhana, disk bahkan sampai yang sangat detail dan dalam seperti register dan port-port pada komputer, baik itu PC maupun *mini computer* dan *Mainframe* (Heryanto: 2008).

Dalam pembuatan program yang menggunakan fungsi atau aritmatika, Bahasa C menawarkan kemudahan dengan menyediakan fungsi – fungsi khusus, seperti: pembuatan konstanta, operator aritmatika, operator logika, operator *bitwise* dan operator *Assigment*. Selain itu, bahasa C menyediakan Program kontrol seperti: Percabangan (*if* dan *if…else*), Percabangan *switch*, *Looping* (*for*, *while* dan *do…while*), *Array*, serta fungsi – fungsi lainnya.

2.5.2 Penulisan Program Bahasa C

Untuk dapat memahami bagaiman suatu program ditulis, maka struktur dari program harus dimengerti terlebih dahulu. Tiap bahasa komputer mempunyai struktur program yang berbeda. Jika struktur dari program tidak diketahui, maka akan sulit bagi pemula untuk memulai menulis suatu program dengan bahasa yang bersangkutan. Struktur dari program memberikan gambaran secara luas, bagaimana bentuk dari program secara umum. Selanjutnya dengan pedoman struktur program ini, penulis program dapat memulai bagaimana seharusnya program tersebut ditulis (Heryanto, 2008).

Struktur dari program C dapat dilihat sebagai kumpulan dari sebuah atau lebih fungsi-fungsi. Fungsi pertama yang harus ada di program C sudah ditentukan namanya, yaitu bernama main(). Suatu fungsi di program C dibuka dengan kurung kurawal ({) dan ditutup dengan kurung kurawal tutup (}). Diantara kurung-kurung kurawal dapat dituliskan statemen-statemen program C. Berikut ini adalah struktur dari program C (Heryanto, 2008).

```
main()
{
        Statemen-statemen        fungsi utama
}

Fungsi_fungsi_lain()
{                                fungsi-fungsi lain
        Statemen-statemen        yang ditulis oleh
}                                pemrograman komputer
```

Bahasa C dikatakan sebagai bahasa pemrograman terstruktur, karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagian (*subroutine).* Fungsi-fungsi selain fungsi utama merupakan program-program bagian. Fungsi-fungsi ini dapat ditulis setelah fungsi utama atau diletakkan di file pustaka (*library*). Jika fungsi-fungsi diletakkan di file pustaka dan akan dipakai disuatu program, maka nama file judulnya (*header file)* harus dilibatkan di dalam

program yang menggunakannya dengan *preprocessor directive* #include. (Heryanto,2008).

2.5.3 Struktur Bahasa Pemrograman C

Struktur penulisan pemrograman bahasa C secara umum terdiri atas:

1. Header

   Header berisi include file (.hex), yaitu library (pustaka) yang akan digunakan dalam pemrograman.

   Contoh:

   #include <atmega8535.h>

   #include <delay.h>

   #include <stdio.h>
2. Deklarasi konstanta global atau variabel
3. Fungsi atau prosedur (bisa dibawah program utama)

   Prosedur adalah suatu kumpula instruksi untuk mengerjakan sutau keperluan tertentu tanpa mengembalikan sutau nilai
4. Program utama

Contoh penulisan program bahasa C

```
/*HEADER untuk memanggil library yang akan digunakan*/
#include <atmega8535.h>
#include <stdio.h>

/*Deklarasi konstanta atau variabel global*/
unsigned char dt, xx;
char buf [33];

/*Deklarasi fungsi atau prosedur*/
unsigned char lampu(unsigned char bitn)
{
PORTA=bitn & 0x3C;
}
```

```
/*Program utama*/
Void main (void);

/*Deklarasi konstanta dan atau variabel lokal*/
char data;
PORTA=0x00;
DDRA=0xF0;

while (1)
{
. . .
};
}
```

2.5.4 Tipe Data

Tipe data merupakan bagian yang paling penting karena tipe data mempengaruhi setiap instruksi yang akan dilaksanakan oleh komputer. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien. Tipe data pada bahasa C dapat dilihat pada tabel 2.8 dibawah ini:

Table 2.8 Tipe Data Bahasa C

Tipe Data	Ukuran (byte)	Format	Keterangan
Char	1	%c	Karakter/string
Int	2	%i%d	Bilangan bulat (integer)
Float	4	%f	Bilangan pecahan (float)
Double	8	%If	Pecahan presisi ganda

2.6 LCD 2x16

LCD adalah kepanjangan dari Liquid Crystal Display. *LCD* (*Liquid Crystal Display*) digunakan sebagai tampilan dari sebuah informasi. Ada 2 macam jenis *LCD*, yaitu *LCD* yang mempunyai satu baris dan *LCD* yang mempunyai dua baris. *LCD* yang mempunyai satu baris ini disebut *LCD* 1 x 16 dan *LCD* yang mempunyai dua baris disebut dengan *LCD* 2 x 16. Angka 16 ini menunjukkan

banyaknya karakter yang dapat ditampilkan dalam setiap baris (Marethania, 2011). Pada rancang bangun ini *LCD* yang digunakan mempunyai lebar display 2 baris dan 16 kolom atau biasa disebut dengan *LCD character* 2x16, dengan 16 pin konektor yang didefinisikan seperti pada tabel 2.9 berikut ini.

Tabel 2.9 Fungsi-Fungsi Pin pada LCD

PIN	Nama	Fungsi
1	VSS	Ground
2	VCC	+5 V
3	VEE	Pengatur Kontras
4	RS	Register Select 0 = Register Perintah 1 = Register Data
5	R/W	Read / Write 0 = Write Mode 1 = Read Mode
6	E	Enable 0 = Enable 1 = Disable
7-14	DB0	Data Bus
15-16		Tegangan untuk Menyalakan Lampu LCD

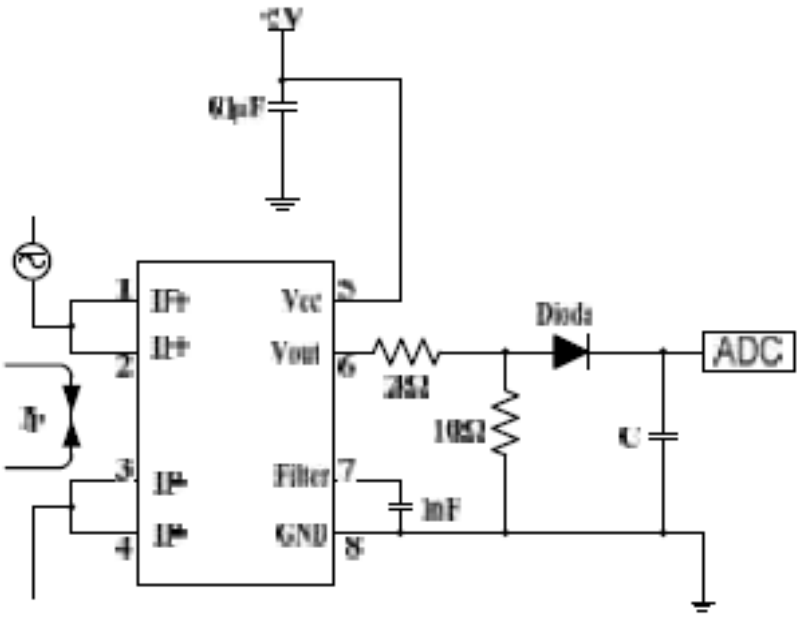
Berikut ini adalah gambar dari tampilan LCD display 2 x 16

Gambar 2.14 Tampilan LCD 2x16

2.7 Sensor Arus ACS712

ACS712 merupakan IC yang berfungsi sebagai sensor arus dan menggantikan trafo arus yang relatif besar dalam bentuk fisiknya. Sensor ACS712 adalah produksi Allegro untuk solusi ekonomis dalam pangukuran arus AC maupun DC. Pada prinsipnya sensor arus ACS712 sama dengan sensor efek hall lainnya yaitu memanfaatkan medan magnetik yang ada disekitar arus yang akan dikonversi menjadi tegangan yang linier dengan perubahan arus. Tegangan yang dihasilkan oleh sensor arus, hasil dari pengukuran arus tersebut berupa tegangan yang variabel. Nilai tegangan yang bervariabel inilah yang akan masuk ke mikrokontroler. Sebelum masuk ke mikrokontroler tegangan yang dihasilkan oleh sensor yang mengukur arus harus dikonversi menjadi tegangan DC, hal ini karena tegangan yang dihasilkan oleh sensor arus tersebut adalah teganan AC.

ACS712 merupakan sensor persisi sebagai sensor arus AC maupun DC pada pembacaan arus didalam dunia industry, pengukuran dan sistem komunikasi. Sensor ini pada umumnya digunakan sebagai pengontrol motor, penghitung beban, dan proteksi arus lebih. Berikut ini merupakan gambar rangkaian sensor arus ACS712 yang dapat dilihat pada gambar 2.15. Sedangkan keterangan fungsi dari masing-masing pin dapat dilihat pada tabel 2.10 berikut ini.



Gambar 2.15 Rangkaian Sensor ACS712

Tabel 2.10 Fungsi Masing-Masing Pin ACS712

No Pin	Nama	Fungsi
1 dan 2	IP+	Terminal untuk arus yang akan disensor
3 dan 4	IP-	Terminal untuk arus yang akan disensor
5	GND	Terminal Ground
6	FILTER	Terminal untuk kapasitor eksternal
7	Vout	Sinyal analog keluaran
8	Vcc	Power Supply

Pada gambar 2.16 berikut ini merupakan gambar dari bentuk fisik sensor arus ACS712

Gambar 2.16 Bentuk Fisik Sensor ACS712

2.8 Catu Daya (*Power Supply*)

Semua peralatan elektronika menggunakan sumber tenaga untuk beroperasi. Sumber tenaga tersebut bermacam-macam. Ada yang menggunakan baterai, Accu dan ada juga yang langsung menggunakan tegangan listrik jala-jala PLN. Untuk konsumsi yang berasal dari tegangan listrik untuk alat-alat elektronika tertentu tidak bisa langsung dikonsumsi, akan tetapi harus disesuaikan dengan tegangan yang diperlukan oleh peralatan tersebut. Penyesuaian tegangan ini dilakukan oleh sebuah alat yang dinamakan *power supply* atau adaptor (Azwar, 2013).

Sebuah *power supply* adalah sebuah perangkat yang memasok energi listrik untuk satu atau lebih beban listrik. Tegangan jala-jala 220 volt dari listrik PLN diturunkan oleh *trafo* atau *transformator* penurun tegangan yang menerapkan perbandingan lilitan, dimana perbandingan lilitan dari suatu *transformator* akan mempengaruhi perbandingan tegangan yang dihasilkan.

Tegangan yang dihasilkan oleh *trafo* masih berbentuk gelombang AC dan harus disearahkan dengan menggunakan penyearah. Rangkaian penyearah yang digunakan memanfaatkan 4 buah dioda yang telah dirancang untuk bisa meloloskan kedua siklus gelombang AC menjadi satu arah saja (Azwar, 2013).

2.9 Transformator

Transformator adalah alat yang berfungsi untuk menaikan tegangan dan menurunkan tegangan dengan frekuensi yang harus sama. Disini *transformator* berperan dalam menyalurkan tenaga atau daya listrik dari tegangan tinggi ke tegangan yang rendah atau sebaliknya. Oleh karena itu pula *transformator* merupakan piranti listrik yang termasuk ke dalam golongan mesin listrik statis (Azwar, 2013). Berikut ini merupakan gambar bentuk fisik dari *transformator*.

Gambar 2.17 Bentuk Fisik *Transformator*

Transformator bekerja berdasarkan prinsip kerja induksi *elektromagnetik, d*imana apabila terjadi suatu perubahan fluks magnet pada kumparan primer, maka akan diteruskan ke kumparan sekunder dan menghasilkan suatu gaya gerak listrik (ggl) induksi dan arus induksi. Rumus atau persamaan *transformator* ideal (Azwar, 2013) sebagai berikut :

$$Pp=Ps$$

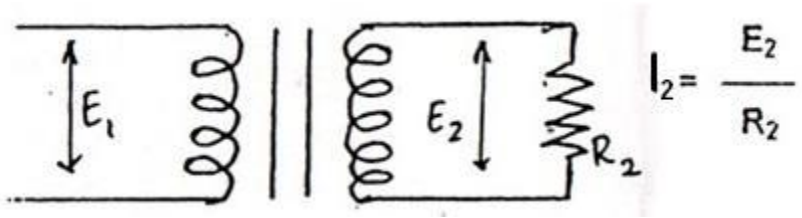
$$Vp \times Ip=Vs \times Is$$

$$\frac{Vp}{Vs}=\frac{Is}{Ip}$$

$$\frac{Is}{Ip}=\frac{Np}{Ns}$$

Keterangan: Pp= Daya primer (watt)
Ps= Daya sekunder (watt)
Vp= Tegangan primer (volt)
Vs= Tegangan sekunder (volt)
Ip= Kuat arus primer (ampere)
Is= Kuat arus sekunder (ampere)
Np= Jumlah lilitan primer
Ns= Jumlah lilitan sekunder

Dalam penggulungan kawat email primer maupun sekunder di dalam t*ransformator* berlaku rumus berikut seperti yang terlihat pada gambar 2.18 dibawah ini (Azwar, 2013):



Gambar 2.18 Ilustrasi Persamaan *Transformator*

Keterangan:
I2 = arus yang mengalir ke beban R2
E1 = tegangan gulungan primer dari PLN atau dari sumber tenaga generator
E2 = tegangan gulungan sekunder

Di dalam perkembangannya terdapat bermacam-macam jenis transformator atau trafo dan mempunyai berbagai fungsi, diantaranya (Azwar, 2013) :

1. *Trafo (Transformator)* Adaptor
2. *Trafo* (*Transformator*) IF (Frekuensi Menengah)
3. *Trafo Step Up* / *Step Down*
4. *Trafo* OT (Output)

Berikut ini contoh fungsi *transformator* yang diaplikasikan dalam kehidupan sehari-hari (Azwar, 2013):

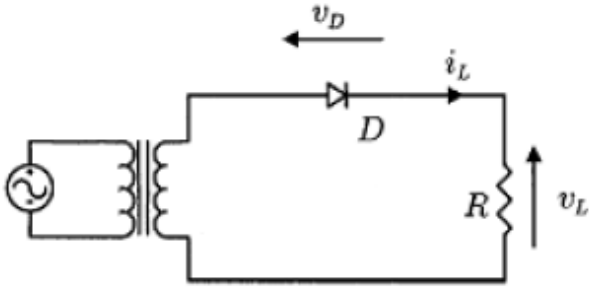
1. *Trafo step up*, fungsi *transformator* ini digunakan untuk menaikkan tegangan AC. Trafo jenis ini dipakai dalam rangkaian-rangkaian pembangkit tegangan pada perangkat elektronika seperti trafo inverter monitor LCD, trafo inverter TV, dll.
2. *Trafo step-down* adalah kebalikannya, fungsi transformator ini untuk menurunkan tegangan AC, contoh pemakaiannya pada adaptor

2.10 Penyearah Gelombang (*Rectifier*)

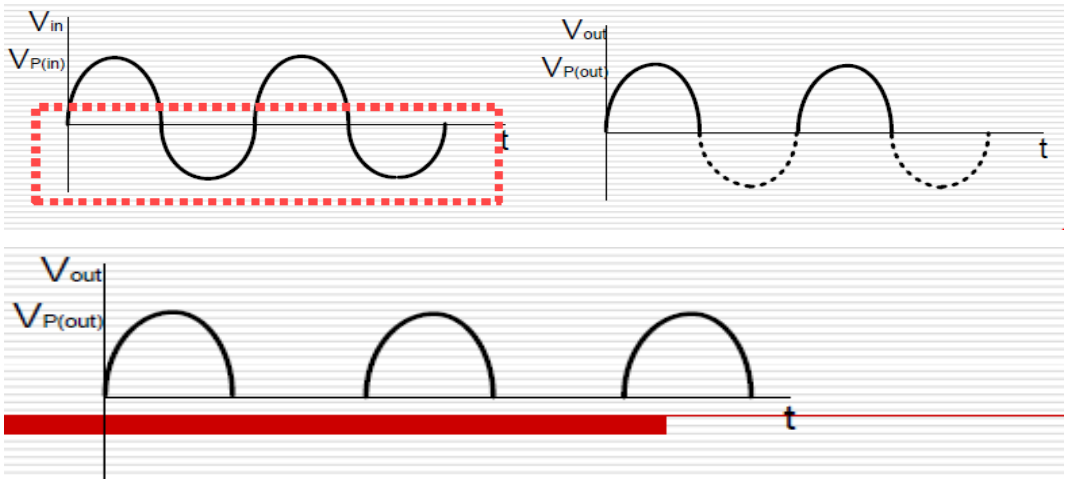
Penyearah gelombang (*rectifier*) adalah bagian dari *power supply* / catu daya yang berfungsi untuk mengubah sinyal tegangan AC (*Alternating Current*) menjadi tegangan DC (*Direct Current*). Komponen utama dalam penyearah gelombang adalah diode yang dikonfigurasikan secara *forward bias*. Dalam sebuah *power supply* tegangan rendah sebelum tegangan AC tersebut diubah menjadi tegangan DC, maka tegangan AC tersebut perlu diturunkan menggunakan *transformator stepdown*. Pada dasarnya konsep penyearah gelombang dibagi dalam 3 jenis, yaitu penyearah setengah gelombang, penyearah gelombang penuh dan penyearah jembatan gelombang penuh (Yusron, 2012).

1. Penyearah setengah gelombang (*Half Wave Rectifier*)

   Pada penyearah setengah gelombang dioda akan berlaku sebagai penghantar selama putaran setengah positif dan tidak berlaku sebagai penghantar pada setengah siklus negatif, sehingga dinamakan sebagai sinyal setengah gelombang. Gambar 2.19 merupakan gambar rangkaian penyearah setengah gelombang (Yusron, 2012). Sedangkan gambar 2.20 merupakan gambar dari sinyal output penyearah setengah gelombang.



Gambar 2.19 Rangkaian Penyearah Setengah Gelombang



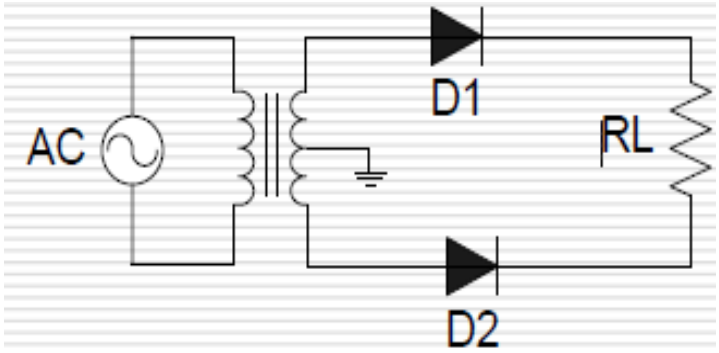
Gambar 2.20 Sinyal Output Penyearah Setengah Gelombang

Nilai rata-rata tegangan output dari penyearah setengah gelombang dapat dihitung dengan persamaan berikut di mana Vp (out) adalah nilai puncak dari tegangan output setengah gelombang:

$$V_{AVG} = \frac{Vp(out)}{\pi}$$

2. Penyearah gelombang penuh (*Full Wave Rectifier*)

   Penyearah gelombnag penuh ini menggunakan *transformator* dengan CT (*Center Tap*). Rangkaian penyearah gelombnag penuh dapat dilihat pada gambar 2.21 dibawah ini (Yusron, 2012).



Gambar 2.21 Rangkaian Penyearah Gelombang Penuh

Prinsip kerja rangkaian penyearah gelombang penuh ini yaitu dioda D1 menghantar ke putaran setengah positif dan Dioda D2 menghantar ke

putaran setengah negatif. Sebagai hasilnya arus beban rectifier mengalir selama setengah putaran bersama-sama. Rangkaian equivalen pada putaran maju stengah siklus positif, D1 merupakan dioda dengan bias maju yang akan menghasilkan sebuah tegangan beban positif yang diindikasikan sebagai *Polarity Plus-Minus* melalui resistor beban. Rangkaian equivalen pada putaran maju stengah siklus negatif. D2 merupakan dioda dengan bias maju yang akan menghasilkan sebuah tegangan beban positif. Selama kedua putaran setengah, tegangan beban mempunyai polaritas yang sama dan arus beban berada dalam satu arah, rangkaian ini disebut sebagai *Rectifier* gelombang penuh, sebab mengganti tegangan masukan AC ke Pulsating (getaran) tegangan keluaran DC (Yusron, 2012).Untuk lebih jelasnya dapat dilihat pada gambar output penyearah gelombang penuh berikut ini.

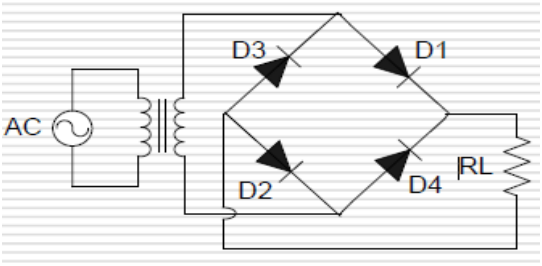
Gambar 2.22 Sinyal Output Penyearah Gelombang Penuh

Nilai rata-rata tegangan output dari penyearah setengah gelombang dapat dihitung dengan persamaan berikut di mana Vp (out) adalah nilai puncak dari tegangan output setengah gelombang:

$$V_{AVG} = \frac{2Vp(out)}{\pi}$$

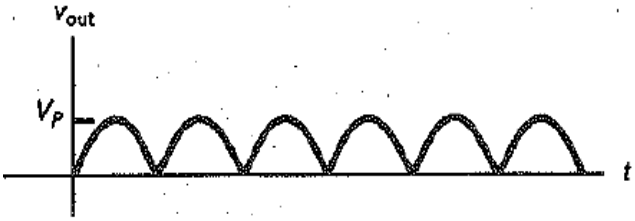
3. Penyearah jembatan gelombang penuh

   Penyearah jembatan gelombang penuh ini menggunakan 4 buah dioda seperti yang terlihat pada gambar 2.23 dibawah ini:



Gambar 2.23 Rangkaian Penyearah Jembatan

Prinsip kerja dari rangkaian penyearah jembatan gelombang penuh diatas dimulai pada saat output *transformator* memberikan level tegangan sisi positif, maka D1 dan D4 berada pada posisi *forward bias*, sedangkan D2 dan D3 berada pada posisi *reverse bias* sehingga level tegangan sisi puncak positif tersebut akan dilewatkan melalui D1 ke D4. Pada saat output *transformator* memberikan level tegangan sisi puncak negatif maka D2 dan D4 berada pada posisi *forward bias*, sedangkan D1 dan D2 berada pada posisi *reverse bias*, sehingga level tegangan sisi negatif tersebut dialirkan melalui D2 dan D4 (Yusron, 2012). Sinyal output penyearah jembatan gelombang penuh dapat dilihat pada gambar 2.24 dibawah ini.

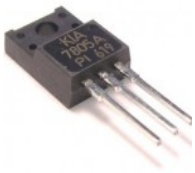


Gambar 2.24 Sinyal Output Penyearah Jembatan

2.11 IC Regulator

IC regulator atau yang sering disebut sebagai regulator tegangan (*voltage regulator*) merupakan suatu komponen elektronik yang melakukan suatu fungsi yang penting dan berguna dalam perangkat elektronik baik digital maupun analog. Hal yang dilakukan oleh IC regulator ini adalah menstabilkan tegangan yang melewati IC tersebut. Setiap IC regulator mempunyai rating tegangannya sendiri-sendiri. Salah satunya IC regulator dengan nomor seri 7805 merupakan regulator tegangan sebesar 5 volt, yang artinya selama tegangan masukan lebih besar dari

tegangan keluaran maka akan dikeluarkan tegangan sebesar 5 volt (Marethania, 2011). IC regulator 7805 ini mempunyai 3 buah kaki, yaitu kaki tegangan masukan yang biasa sering disebut Vin, kaki *ground* (0V) dan yang ketiga adalah kaki tegangan keluaran atau Vout, seperti yang terlihat pada gambar 2.25 dibawah ini (Surjati, 2008).



Gambar 2.25 IC Regulator 7805

2.12 UML (*Unified Modelling Language*)

2.12.1 Pengertian *Unified Modelling Language* (UML)

Unified Modelling Language adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual (Haviluddin: 2011). UML juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek (Haviluddin: 2011). Gambar logo dari UML dapat dilihat pada gambar 2.26 berikut ini.

Gambar 2.26 Logo *Unified Modelling Language* (UML)

2.12.2 Sejarah Singkat *Unified Modelling Language* (UML)

UML dimulai secara resmi pada Oktober 1994 ketika *Rumbaugh* bergabung dengan *Booch* pada *Relation Software Corporation*. Proyek ini memfokuskan pada penyatuan metode *Booch* dan *OMT*. UML versi 0.8 merupakan metode penyatuan yang dirilis pada bulan Oktober 1995. Disini beberapa *partner* yang berkontribusi pada UML 1.0, diantaranya *Digital Equipment Corporation, Hewlett-Packard, I-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse,*

Microsoft, Oracle, Relation, Texas Instruments dan *Unisys*, dari kolaborasi ini dihasilkan UML 1.0 yang merupakan bahasa pemodelan yang ditetapkan secara baik, *expressive*, kuat dan cocok untuk lingkungan masalah yang luas. UML 1.0 ditawarkan menjadi standarisasi dari *Object Management Group (OMG),* dan pada Januari 1997 dijadikan sebagai standar bahasa pemodelan. Pada bulan Januari 1997 ini lahirlah UML versi 1.0. Pada bulan September 1997 UML versi 1.1, dengan 8 buah diagram, yaitu:

1. Use Case diagram
2. Activity diagram
3. Sequence diagram
4. Collaboration diagram
5. Class diagram
6. Statechart diagram
7. Component diagram
8. Deployment diagram

Pada tahun 2002 lahirlah UML versi 2.0, menjadi 13 buah diagram, dengan penambahan dan penggantian yaitu :

1. Use Case Diagram
2. Activity Diagram
3. Sequence Diagram
4. Communication Diagram (Collaboration diagram in versi 1.x)
5. Class Diagram
6. State Machine Diagram (Statechart diagram in versi 1.x)
7. Component Diagram
8. Deployment Diagram
9. Composite Structure Diagram
10. Interaction Overview Diagram
11. Object Diagram
12. Package Diagram
13. Timing Diagram

2.12.3 Diagram-diagram pada UML (*Unified Modelling Language*)

Diagram-diagram yang terdapat didalam UML antara lain:

1. **Use Case Diagram**

   Use case merupakan gambaran fungsionalitas dari suatu system, sehingga pengguna system mengerti kegunaan system yang akan dibangun. Use case diagram adalah penggambaran system dari sudut user, sehingga pembuatan use case lebih dititik beratkan pada fungsionalitas yang ada pada system, bukan berdasarkan alur kegiatan system (Haviluddin: 2011). Komponen-komponen yang terlibat dalam use case diagram :

   a. **Actor**

      Pada dasarnya *actor* bukanlah bagian dari *use case diagram*, namun untuk dapat terciptanya suatu *use case diagram* diperlukan beberapa *actor*. *Actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah *actor* mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima, dan memberi informasi pada sistem. *Actor* hanya berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. *Actor* digambarkan dengan *stick man* . *Actor* dapat digambarkan secara secara umum atau spesifik, dimana untuk membedakannya kita dapat menggunakan *relationship* (Haviluddin: 2011).

   b. **Use Case**

      Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun (Haviluddin: 2011).

2. **Activity Diagram**

   Pada dasarnya diagram Activity sering digunakan pada *flowchart*. Diagram ini berhubungan dengan diagram Statechart. Diagram Statechart

berfokus pada obyek yang dalam suatu proses (atau proses menjadi suatu obyek), diagram Activity berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Komponen-komponen activity diagram ini dapat dilihat pada tabel 2.11 dibawah ini.

Tabel 2.11 Komponen-komponen Activity Diagram:

No	Gambar	Nama	Keterangan
1		Activity	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraski satu sama lain
2		Action	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		Start State	Bagaimana objek dibentuk atau diawali
4		End State	Bagaimana objek dibentuk dan dihancurkan
5		State Transition	State transition menunjukkan kegiatan apa berikutnya setelah suatu kegiatan
6		Fork	Percabangan yang menunjukkan aliran pada activity diagram
7		Join	Penggabungan yang menjadi arah aliran pada activity diagram
8		Decision	Pilihan untuk mengambil keputusan
9		Flow Final	Aliran akhir

3. Sequence Diagram

Diagram sequence merupakan salah satu diagram *Interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan. Diagram ini diatur berdasarkan waktu. Objek-objek yang berkaitan dengan proses berjalannya

operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.

4. Class Diagram

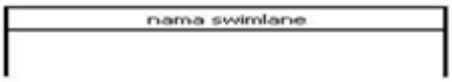
Class adalah sebuah spesifikasi yang akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class diagram* menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. Class menggambarkan keadaan (*atribut / property*) suatu sistem sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut. Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain sepeti *containment*, pewarisan, asosiasi, dan lain-lain. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat (Haviluddin: 2011). Class memiliki tiga area pokok yaitu:

a. Nama
b. Atribut
c. Metoda

5. Object Diagram atau Overview Diagram

Overview Diagram adalah pencangkokan secara bersama antara activity diagram dan sequence diagram. *Interaction Overview Diagram* dapat dianggap sebagai *activity diagram* dimana semua aktivitas diganti dengan sedikit *sequence diagram*, atau bisa juga dianggap sebagai *sequence diagram* yang dirincikan dengan notasi *activity diagram* yang digunakan untuk menunjukkan aliran pengawasan (Haviluddin: 2011). Komponen-komponen Overview Diagram dapat dilihat dengan jelas pada tabel 2.12 berikut ini.

Table 2.12 Komponen-komponen Overview Diagam

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / decision 	Asosiasi percabangan, dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / join	Asosiasi penggabungan, dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimiane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi
Fork 	Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel
Join 	Digunakan untuk menunjukkan kegiatan yang digabungkan

2.12.4 Tujuan UML

Berikut ini tujuan utama dalam desain UML adalah (Sugrue J. 2009 dalam Haviluddin, *Memahami Penggunaan UML* (*Unified Modelling Language*). 2011):

1. Menyediakan bagi pengguna (analisis dan desain sistem) suatu bahasa pemodelan visual yang ekspresif sehingga mereka dapat mengembangkan dan melakukan pertukaran model data yang bermakna.
2. Menyediakan mekanisme yang spesialisasi untuk memperluas konsep inti.
3. Memberikan dasar formal untuk pemahaman bahasa pemodelan.

2.12.5 Cakupan UML

Adapun cakupan UML antara lain:

1. UML menggabungkan konsep *Bocch*, *OMT*, dan *OOSE*, sehingga UML merupakan suatu bahasa pemodelan tunggal yang umum dan digunakan secara luas oleh para *user* ketiga metode tersebut dan bahkan para *user* metode lainnya.
2. UML menekankan pada apa yang dapat dikerjakan dengan metode-metode tersebut.
3. UML berfokus pada suatu bahasa pemodelan standar, bahkan pada proses standar. Meskipun UML harus diaplikasikan dalam konteks sebuah proses dari pengalaman bahwa organisasi dan masalah yang berbeda juga memerlukan proses yang berbeda pula.

2.13 Notasi pada UML

Terdapat beberapa macam notasi pada UML, yaitu diantaranya:

1. Actor

Actor menggambarkan segala pengguna software aplikasi (*user*). Actor memberikan suatu gambaran jelas tentang apa yang harus dikerjakan software aplikasi. Sebagai contoh sebuah actor dapat memberikan input kedalam dan menerima informasi dari software aplikasi, perlu dicatat bahwa sebuah actor berinteraksi dengan use case, tetapi tidak memiliki kontrol atas use case (Haviluddin: 2011). Berikut ini merupakan gambar Notasi *Actor*.

Gambar 2.27 Notasi *Actor*

2. Use Case

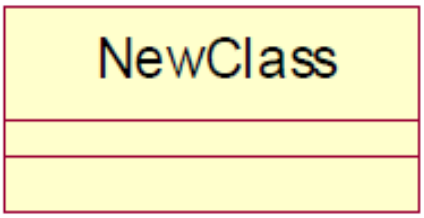
Use case menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, namun *use case* hanya menjelaskan apa yang dilakukan oleh *actor* dan

sistem bukan bagaimana *actor* dan sistem melakukan kegiatan tersebut. Berikut ini merupakan gambar Notasi *Use Case.*

Gambar 2.28 Notasi *Use Case*

3. Class

Class merupakan pembentuk utama dari sistem berorientasi objek, karena class menunjukkan kumpulan objek yang memiliki atribut dan operasi yang sama. Class digunakan untuk mengimplementasikan *interface*. Class digunakan untuk mengabstraksikan elemen-elemen dari sistem yang sedang dibangun. Class bisa mempresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata, dibawah ini merupakan gambar dari Notasi Class.



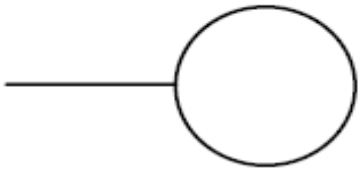
Gambar 2.29 Notasi Class

Notasi class berbentuk persegi panjang yang berisi 3 bagian. Persegi panjang paling atas untuk nama class, persegi panjang paling bawah untuk operasi dan persegi panjang ditengah untuk atribut. Atribut juga digunakan untuk menyimpan informasi. Nama atribut menggunakan kata benda yang bisa dengan jelas merepresentasikan informasi yang tersimpan didalamnya. Operasi menunjukkan sesuatu yang bisa dilakukan oleh objek dan menggunakan kata kerja.

4. Interface

Interface merupakan kumpulan operasi tanpa implementasi dari suatu class. Implementasi operasi dalam *interface* dijabarkan oleh operasi didalam class. Oleh karena itu keberadaan *interface* selalu disertai oleh class yang mengimplementasikan operasinya. *Interface* ini merupakan

salah satu cara mewujudkan *prinsip enkapsulasi* dalam objek. Berikut ini merupakan gambar Notasi *Interface.*



Gambar 2.30 Notasi *Interface*

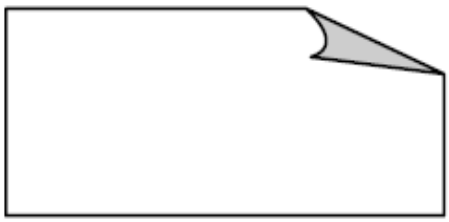
5. Interaction

Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar objek maupun hubungan antar objek. Biasanya *interaction* ini dilengkapi juga dengan teks bernama *operation signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan. Pada gambar 2.31 dibawah ini menunjukkan Notasi *Interaction.*

Gambar 2.31 Notasi *Interaction*

6. Note

Note digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain. Berikut ini adalah gambar dari Notasi *Note.*



Gambar 2.32 Notasi *Note*

7. Dependency

Dependency merupakan relasi yang menunjukkan bahwa perubahan pada salah stu elemen memberi pengaruh pada elemen lain. Elemen yang ada dibagian tanda panah adalah elemen yang tergantung pada elemen yang

ada dibagian tanpa tanda panah. Terdapat 2 *stereotype* dan *dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada digaris dengan panah). *Extend* menunjukkan bahwa sutau bagian dari elemen di garis tanpa panah bisa disisipkan kedalam elemen yang ada digaris dengan panah. Pada gambar 2.33 dibawah ini menunjukkan Notasi *Dependency.*

Gambar 2.33 Notasi *Dependency*

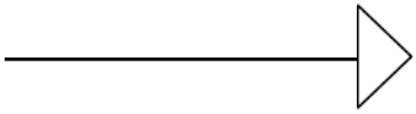
8. Association

Association menggambarkan navigasi antar class (*navigation*), berapa banyak objek lain yang bisa berhubungan dengan satu objek (*multiplicity* antar class) dan apakah suatu class menjadi bagian dari class lainnya (*aggregation*). *Navigation* dilambangkan dengan penambahan tanda panah di akhir garis. *Bidirectional navigation* menunjukkan bahwa dengan mengetahui salah satu class bisa didapatkan informasi dari class lainnya. Sementara *Unidirectional navigation* hanya dengan mengetahui class diujung garis *association* tanpa panah kita bisa mendapatkan informasi dari class di ujung dengan panah, tetapi tidak sebaliknya. Berikut ini merupakan gambar Notasi *Association.*

Gambar 2.34 Notasi *Association*

9. Generalization

Generalization menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization,* class yang lebih spesifik (*subclass)* akan menurunkan atribut dan operasi dari class yang lebih umum (*superclass*) atau "*subclass is superclass"*. Dengan menggunakan notasi *generalization* ini, konsep *inheritance* dari prinsip hirarki dapat dimodelkan. Gambar 2.35 dibawah ini menunjukkan Notasi *Generalization.*



Gambar 2.35 Notasi Generalization

10. Realization

Realization menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah. Misalnya class merealisasikan *package*, komponen merealisasikan class atau *interface.* Pada gambar 2.36 dibawah ini menunjukkan gambar dari Notasi *Realization.*

Gambar 2.36 Notasi *Realization*