# BAB II
# TINJAUAN PUSTAKA

## 2.1 Penelitian Terdahulu

Pada penelitian sebelumnya yang dilakukan oleh Zanuar R akhman dan M Ibrahim Ashari tahun 2012 dalam jurnal yang berjudul "Perancangan dan Pembuatan Sistem Proteksi Kebocoran Air Pada Pelanggan PDAM Dengan Menggunakan Selenoid Valve dan Water Pressure Switch Berbasis Atmega 8535". Dalam penelitian ini menjelaskan salah satu masalah yang timbul adalah kebocoran air pada konsumen/ pelanggan Perusahaan Daerah Air Minum (PDAM), hal ini disebabkan karena meter air yang terus berputar meskipun hanya beberapa air yang menetes dari saluran PDAM yang pada dasarnya tanpa kita kehendaki dan mungkin tanpa kita ketahui.

Air yang tetap menetes itu diakibatkan karena pemakaian keran pengaman pada saluan PDAM yang mudah rusak ataupun karena kelalaian penghuni rumah yang lupa mematikan keran kurang rapat, sehingga air tetap mengalir meskipun cuma sedikit. Keran air yang umumnya dipakai oleh pelangan adalah keran yang digerakkan secara manual oleh manusia dengan cara memutar atau menggerakkan keran keatas atau kebawah. Peralatan tersebut sangat mudah mengalami kerusakan dikarenakan kurang bijak dalam mengoperasikannya, sehingga kerusakan dan kelalaian dalam penggunaan keran tersebut akan berdampak pada kebocoran dan pemborosan yang menyebabkan meter air terus berputar, sehingga pelanggan wajib membayar air yang tidak dikehendaki tersebut.

Oleh sebab itu, untuk memudahkan dalam mendeteksi kebocoran air pada pelanggan, khususnya diakibatkan oleh keran air yang menutup kurang rapat ataupun rusak, maka dirancang sebuah alat untuk mendeteksi dan memproteksi kebocoran tersebut. Alat ini dinamakan Sistem proteksi kebocoran air pada pelanggan PDAM dengan menggunakan solenoid valve dan water pressure switch berbasis ATMEGA8535, yang mempunyai komponen inti mikrokontroler ATMEGA8535, sensor tekanan, dan solenoid valve. Sensor tekanan membaca tekanan pada pipa saluran air, tegangan yang dihasilkan akan semakin besar bila

tekanan dalam pipa besar dan keran dalam keadaan tertutup, dan sebaliknya. Saat mikrokontroler menerima data dari sensor, mikrokontroler akan memproses dan menyimpan data dan akan menampilkan pada LCD. Program yang digunakan adalah bascom.

Hasil yang dicapai pada pengujian perancangan dan pembuatan sistem proteksi kebocoran air  Pada saat keran ditutup sepenuhnya, *pressure* meter menunjukkan tekanan 70-80 kPa. Hal tersebut diasumsikan tidak terjadi kebocoran pada keran air .  Dan pada saat keran air dibuka sedikit, hingga seperti dalam keadaan menutup kurang rapat atau bocor (air dialirkan dalam debit kecil). Pada keadaan ini solenoid valve secara otomatis akan menutup saluran air, LCD menampilkan tulisan bocor dan buzzer berbunyi.

Maka kesimpulan yang dapat diambil Dari peoses perancangan dan pembuatan sistem proteksi kebocoran air dan Setelah dilakukan pengujian dan pengukuran peralatan, maka dapat di simpulkan beberapa hal yang berhubungan dengan kinerja peralatan yaitu Sensor MPX5100GP menghasilkan output tegangan (V) berbanding lurus dengan tekanan pada saluran air (kPa). Bila tekanan dalam saluran air besar, maka tegangan  output yang dihasilkan juga besar, begitu pula sebaliknya, bila tekanan pada saluran air kecil, maka tegangan output yang dihasilkan juga kecil. Kemudian  Tegangan keluaran yang dihasilkan Sensor MPX5100GP berada pada range 0.3 volt - 4.7 volt dengan range tekanan 0 kPa – 100 kPa. (menurut teori), sedangkan pada pengujian digunakan tekanan maksimal 70-80 kPa.  Pada percobaan sensor tekanan keseluruhan, rata-rata % kesalahan (error) sebesar 4,4% . serta Selenoid valve akan bekerja (membuka saluran) bila pada input ULN2003 diberi tegangan 5V dan output ULN2003 0V.

Penelitian berikutnya yang dilakukan oleh Ahmad Fajrul Falah dan Triyogatama Wahyu Widodo tahun 2014 dalam jurnal yang berjudul “ Purwarupa Mekanisme Akuisisi Data Rotary Vane Positive Displacement Flowmeter dengan Kompensasi Suhu”. Pada penelitian Purwarupa Mekanisme Akuisisi Data Rotary Vane Positive Displacement Flowmeter dengan Kompensasi Suhu ini menjelaskan bahwa Rotary vane positive displacement (PD) flowmeter termasuk dalam kategori flowmeter yang bekerja menggunakan prinsip operasi volumetric flow.

Perubahan suhu pada fluida akan mengakibatkan perubahan pada volume fluida baik berupa penyusutan maupun pemuaian.

Penelitian ini bertujuan untuk merancang dan mengimplementasikan purwarupa mekanisme akuisisi data rotary vane PD flowmeter dengan turut menyertakan suhu sebagai variabel kompensasi pengukuran. Aliran fluida disimulasikan menggunakan motor DC yang memutar model chamber, sedangkan perubahan suhu lingkungan disimulasikan menggunakan elemen pemanas. Nilai suhu standar yang digunakan sebagai acuan adalah 15°C. Pengujian dilakukan dengan memberikan variasi nilai pengaturan koefisien muai 0,0007/°C dan 0,001/°C, kecepatan aliran 335 L/min, 506 L/min, dan 556 L/min, serta nilai pengaturan suhu 30°C, 35°C, 40°C, 45°C, dan 50°C.

Hasil percobaan didapatkan nilai kompensasi volume tertinggi adalah 65,854 liter sedangkan yang terendah adalah 10,530 liter. Pada nilai volume dan suhu yang sama, semakin tinggi nilai koefisien muai suatu fluida maka semakin tinggi pula nilai kompensasi yang diberikan demikian juga sebaliknya.

Dari penelitian yang telah dilakukan, dapat diambil beberapa kesimpulan. Implementasi purwarupa mekanisme akuisisi data rotary vane positive displacement flowmeter dengan kompensasi suhu telah berhasil dilakukan dan dapat bekerja dengan baik. Persentase selisih total volume dengan variasi suhu dan kecepatan aliran baik pada pengaturan koefisien muai 0,0007/°C maupun 0,001/°C menunjukkan kecenderungan nilai yang sama dalam variasi nilai suhu yang sama.

Hal ini sesuai dengan prinsip kerja kompensasi flowmeter dengan variabel suhu yakni semakin tinggi selisih suhu aktual dengan suhu standar maka semakin tinggi pula nilai kompensasi yang diberikan demikian juga sebaliknya. Nilai selisih total volume hasil dari proses kompensasi yang tertinggi adalah pada percobaan dengan pengaturan nilai koefisien muai 0,001/°C, kecepatan 556 L/min, dan suhu 50°C yakni sebesar 65,854 liter sedangkan yang terendah adalah pada percobaan dengan pengaturan nilai koefisien muai 0,0007/°C, kecepatan 335 L/min, dan suhu 30°C yakni sebesar 10,530 liter.

Kemudian pada penelitian selanjutnya yang dilakukan oleh Suyamto, Sudiono Dan Edy Eko Prasetiyo tahun 2007 dalam jurnal yang berjudul “Rancang

Bangun Sistem Penampil Digital Laju Alir Air Pendingin Sekunder Reaktor Kartini Menggunakan *Paddlewheel Flosensors* Mk 515". Pada penelitian ini menjelaskan bahwa Pada reaktor nuklir seperti reaktor Kartini terdapat komponen-komponen utama dan penunjang agar reaktor dapat dioperasikan sebagaimana mestinya.

Komponen-komponen tersebut di antaranya adalah teras reaktor, moderator, batang kendali, sistem pendingin, sistem pengungkung dan lain-lain. Pada reactor Kartini terdapat 2 macam sistem pendingin, yaitu sistem pendingin primer dan system pendingin sekunder dengan fluida kerja berupa air ringan. Sistem pendingin primer adalah system pendingin yang berhubungan langsung dengan air tangki reaktor. Dalam sistem pendingin primer, air panas dari tangki reaktor dialirkan ke sistem penukar kalor atau *Heat Exchanger* (HE) menggunakan pompa primer. Di dalam HE, air tersebut didinginkan oleh air pendingin sekunder yang terdapat pada sistem pendingin sekunder, selanjutnya di buang ke lingkungan menggunakan Cooling Tower dengan cara hamburan atau *spray* menjadi butiran-butiran air. Besarnya laju aliran air dari sistem
pendingin primer maupun sekunder sangat menentukan proses terjadinya perpindahan atau pertukaran panas dari sistem. Karena hal tersebut merupakan masalah yang sangat penting maka ke dua laju aliran tersebut harus selalu diukur dan dimonitor.

Untuk laju alir sistem pendingin sekunder, selama ini jarang dimonitor secara langsung karena adanya beberapa kesulitan misalnya tidak tersedianya alat ukur yang sesuai dan kalibrasinya mengalami kesulitan. Tujuan dari rancang bangun ini adalah untuk membuat alat ukur laju alir digital air pendingin sekunder reaktor Kartini menggunakan *paddlewheel flosensors* MK 515/415, dimana dengan alat ukur yang telah dibuat tersebut, diharapkan laju alir air pendingin sekunder reaktor Kartini dapat diketahui dan dipantau setiap saat dengan mudah.

Hasil Pengujian dilakukan terhadap rangkaian pengubah frekuensi beserta penguatnya, dan unjuk kerja peralatan untuk mengetahui linearitas alat yang dibuat. Pengujian yang pertama ditujukan untuk mengetahui korelasi antara tegangan keluaran dengan frekuensi masukan. Rangkaian penguat disini adalah rangkaian penguat awal dan rangkaian penguat akhir.

Besarnya tegangan keluaran dari rangkaian dapat diketahui dengan mengalikan tegangan keluaran dari rangkaian FVC terhadap besarnya penguatan yang dihasilkan oleh rangkaian penguat akhir, yaitu sebesar 1,67. Pengujian yang kedua ditujukan untuk mengetahui unjuk kerja peralatan secara lengkap dimana hasil pengukuran laju alir fluida dibandingkan dengan hasil kalibrasi yang tercantum dalam *datasheet* dari transduser dan blok diagram.

Dalam pengujian ini frekuensi keluaran dari *function generator* diset sama dengan frekuensi kalibrasi alat yang terdapat dalam *datasheet test certificate* kemudian dimasukkan ke bagian masukan alat penampil laju alir. Nilai dari laju alir tranduser diperoleh dari nilai kecepatan alir yang dihasilkan oleh transduser pada *datasheet* kemudian dikoreksi dengan dimeter pipa pendingin sekunder yang dipakai di reactor Kartini sebesar 5 inchi atau luas penampangnya sama dengan 126,61. 10-4 m2.. Hal ini terjadi disebabkan oleh beberapa hal diantaranya adalah tegangan pada saat dilakukan pengujian kurang stabil, spesifikasi komponen yang digunakan kurang tepat dan adanya *noise* pada saat pengubahan frekuensi menjadi sinyal tegangan oleh rangkaian FVC.

Dari hal-hal yang telah diterangkan dapat disimpulkan bahwa peralatan penampil laju aliran yang telah dibuat dapat berfungsi dengan baik dan mempunyai linearitas yang tinggi serta sistem penampil laju alir tersebut dapat mengukur laju air pendingin sekunder sampai dengan 4.558,05 liter/menit.

### 2.1.1 Perbedaan dengan Penelitian Sebelumnya

Dalam laporan akhir yang berjudul "**Rancang Bangun Alat Monitoring Volume - Biaya Pemakaian Air PDAM dengan Memori Berbasis Mikrokontroler Menggunakan Sensor Flowmeter**". Pada rancang bangun alat ini menggunakan sensor flowmeter, ketika sensor flowmeter aktif sensor akan segera melakukan pembacaan aliran air, pembacaan aliran air pada sensor dalam bentuk rotasi dan dikonversi menjadi pulsa digital yang mewakili laju alir kecepatan air. Dari data pulsa digital yang telah dikonversi ini dapat ditentukan debit air yg mengalir melalui sensor, dan setelah proses konversi data selesai. Data pulsa digital ini akan dikonversi menjadi debit air sebelum dialirkan menuju mikrokontroler.

Pada mikrokontroler proses konversi akan dilanjutkan dengan menghitung jumlah pembayaran mengacu pada ketentuan tarif yang ditentukan oleh PDAM. Dan hasil perhitungan yang telah di konversi menjadi rupiah kemudian akan ditampilkan pada display lcd dan disimpan kedalam memori agar apabila terdapat gangguan pada baterai data hasil perhitungan tidak akan hilang.

## 2.2 Sensor Flowmeter

Sensor Flowmeter adalah perangkat yang mengukur gerakan jumlah cairan, gas atau uap yang melewati mereka. Sebuah flowmeter standar terdiri dari serangkaian komponen terkait yang mentransmisikan sinyal yang menunjukkan volume, laju aliran, atau volume cairan bergerak melalui saluran tertentu. (Rudi, 2011)

Sensor Flowmeter terdiri dari transduser, perangkat utama dan pemancar. Transduser berfungsi untuk merasakan cairan yang melewati perangkat utama. Pemancar menghasilkan sinyal arus yang dapat digunakan dari sinyal transduser mentah. Komponen-komponen ini sering dikombinasikan, sehingga flowmeter sebenarnya bisa memiliki satu atau mungkin beberapa perangkat. Dapat dilihat Sensor flowmeter pada gambar 2.1 dibawah ini.

**Gambar 2.1 Sensor Flowmeter**

## 2.3 Mikrokontroler Atmega16

Atmega 16 adalah seri mikrokontroler CMOS 8-bit keluaran ATMEL yang berbasiskan RISC arsitektur. (Hadi, 2008)

RISC adalah kepanjangan dari *Reduced Instruction Set Computer* yang memiliki karakteristik:

1. Arsitektur Load-Store
2. Memiliki panjang instruksi yang tetap (biasanya 32 bit)
3. Arsitektrur berpengalamatan 3 operand
4. Instruksi operasi yang mudah

Contoh dari RISC: MIPS, SPARC, IBM/Motorola PowerPC, Compaq Alpha, ARM, SH4, HP-PA. Berbeda dengan RISC, terdapat pula jenis arsitektur yang lain yaitu CISC. CISC adalah kepanjangan dari Complex Instruction Set Computer.

Karakteristik CISC adalah sbb:

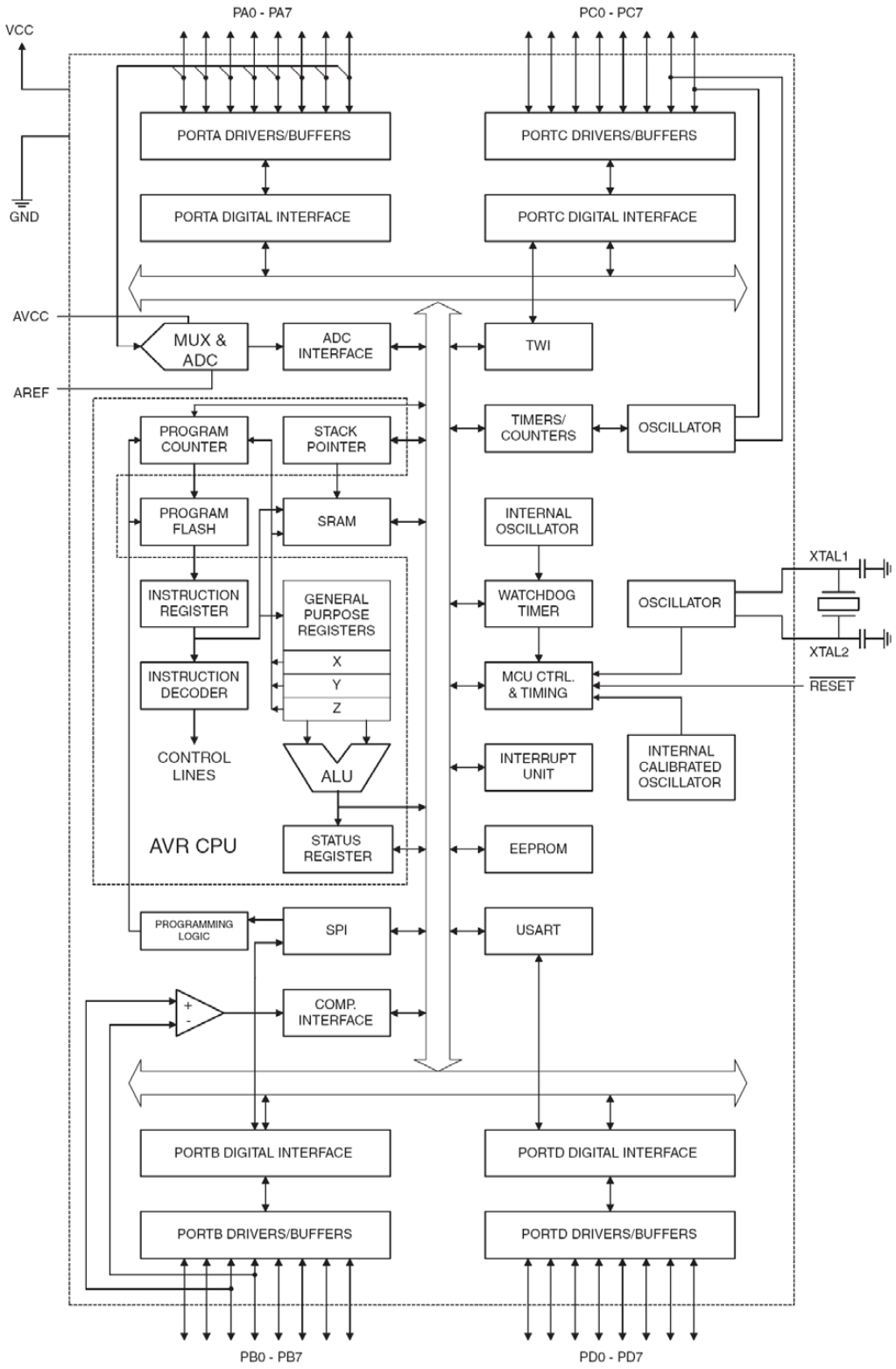
1. Arsitektur Register-memory
2. Panjang instruksi yang bervariabel
3. Instruksi operasi yang kompleks

Contoh dari CISC: Intel 80×86, VAX, IBM 360.

Semua instruksi yang diperintahkan diolah dalam kode 16 bit dan dieksekusi dalam satu single clock. Walaupun Atmega 16 sudah lama keluar akan tetapi masih banyak digunakan hingga saat ini entah untuk robotika maupun percobaan mekatronika di perguruan tinggi maupun sekolah kejuruan. Atmega 16 mempunyai beberapa fitur antara lain sebagai berikut;

1. Mempunyai 16Kb flash program memory
2. 512 EEPROM
3. 1Kb SRAM
4. 32 general-purpose I/O lines
5. 32 general-purpose working register

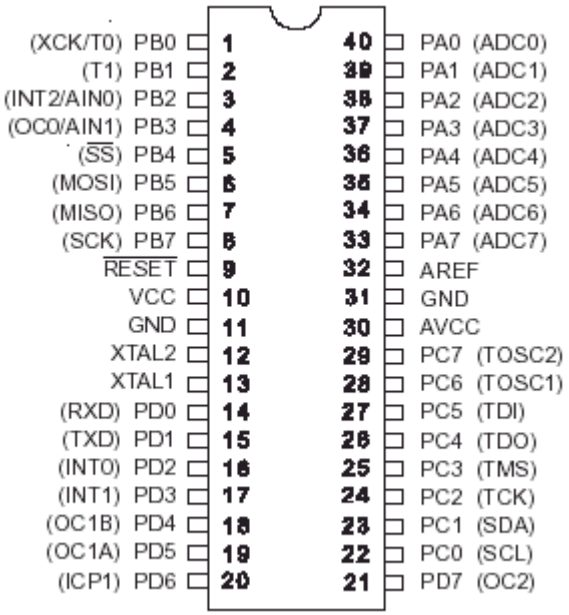
### 2.3.1. Arsitektur Atmega 16




**Gambar 2.2 Blok diagram Atmega 16**

### 2.3.2. Konfigurasi Atmega 16

**Konfigurasi pin Mikrokontroler Atmega 16**




**Gambar 2.3 Pin-pin Atmega 16**

Konfigurasi pin ATMEGA16 dengan kemasan 40 pin DIP (*dual in-line package*). Dari gambar diatas ini dapat dijelaskan fungsi dari masing-masing ATMEGA16 sebagai berikut:

1. VCC merupakan pin yang berfungsi sebai masukan catu daya.
2. GND merupakan pin Ground.
3. Port A (PA0..PA7) merupakan pin input / outut dua arah dan pin masukan ADC.
4. Port B (PB0..PB7) merupaka pin input / output dua arah dan pin fungsi khusus.
5. Port C (PC0..PC7) merupakan pin input/ output dua arah dan pin fungsi khusus
6. Port D (PD0..PD7) merupakan pin input/output dua arah dan pin khusus
7. Reset merupakan pin yang digunakan untuk m me-reset mikrokontroller.
8. XTAL1 dan XTAL2 merupakan pin masukan clock eksternal.

9. AVCC merupakan pin masukan tegangan intuk ADC.
10. AREF merupakan pin masukan tegangan referensi ADC.

## 2.4. Pompa air DC

Jenis pompa air DC disini menggunakan pompa rotari roda gigi-gigi, pompa ini merupakan jenis pompa rotari yang paling sederhana.Apabila gerigi roda gigi pada sisi hisap cairan akan mengisi ruangan yang ada diantara gerigi tersebut,kemudian cairan ini akan dibawa berkeliling dan ditekan keluar apabilka geriginya bersatu lagi. Roda gigi itu dapat berupa gigi helix tunggal, helix ganda atau gigi lurus. (Maulidy, 2006)

Beberapa desain mempunyai lubang fluida yang radial pada roda gigi bebas dari bagian dan atas akar gerigi sampai ke lubang dalam roda gigi ini memungkinkan cairan melakukan jalan pintas atau bypas dari satu gigi ke gigi lain nya. Untuk menghindarkan terjadi nya tekanan berlebih yang akan membebani bantalan secara berlebihan dan menimbulkan kebisingan. Dapat dilihat gambar 2.4 Pompa air DC.

**Gambar 2.4 Pompa Air DC**

## 2.5 LCD ( *Liquid Cristal Display* )

Banyak sekali kegunaan LCD dalam perancangan suatu sistem yang menggunakan mikrokontroler. LCD berfungsi untuk menampilkan suatu nilai hasil sensor, menampilkan teks, atau menampilkan menu pada aplikasi mikrokontroler. LCD yang digunakan adalah jenis *LMB162AFC* yang merupakan

modul LCD dengan tampilan 16 x 2 baris dengan konsumsi daya rendah. (Saputra, 2014)

*LMB162AFC* dapat dioperasikan menjadi 2 mode. Mode 1 interface data 4 bit, dan yang ke dua interface data 8 bit. Jika pengoperasiannya menggunakan data 4 bit maka dibutuhkan 2 kali pengiriman data per-karakter, sedangkan menggunakan pengiriman data 8 bit relatif lebih mudah, karena tidak menghabiskan memori program tapi membutuhkan 4 tambahan jalur I/O.Dalam implementasi *LMB162AFC* secara umum ada 3 cara yang sering digunakan:

1. Interface 8 bit
2. Interface data 4 bit, dengan pengiriman data high nibble pada port
3. Interface data 4 bit, dengan pengiriman data low nibble pada port

LCD ini juga mempunyai tiga sinyal kontrol, diantaranya: Enable (E), Read/Write (R_W), dan register select (RS). Untuk menampilkan suatu huruf atau angka, data yang dikirim harus merupakan kode ASCII dari huruf dan angka tersebut.


**Gambar 2.5 LCD (*Liquid Crystal Display)***

## 2.6 Baterai

**Baterai** adalah alat listrik-kimiawi yang menyimpan energi dan mengeluarkan tenaganya dalam bentuk listrik. (Rahmanisa, 2012)

Sebuah baterai biasanya terdiri dari tiga komponen penting, yaitu:

1. batang karbon sebagai anode (kutub positif baterai)
2. seng (Zn) sebagai katode (kutub negatif baterai)
3. pasta sebagai elektrolit (penghantar)

Baterai yang biasa dijual (*disposable*/sekali pakai) mempunyai tegangan listrik 1,5 volt. Baterai ada yang berbentuk tabung atau kotak. Ada juga yang dinamakan *rechargeable battery*, yaitu baterai yang dapat diisi ulang, seperti yang biasa terdapat pada telepon genggam. Baterai sekali pakai disebut juga dengan baterai primer, sedangkan baterai isi ulang disebut dengan baterai sekunder. Baik baterai primer maupun baterai sekunder, kedua-duanya bersifat mengubah energi kimia menjadi energi listrik.

Baterai primer hanya bisa dipakai sekali, karena menggunakan reaksi kimia yang bersifat tidak bisa dibalik (*irreversible reaction*). Sedangkan baterai sekunder dapat diisi ulang karena reaksi kimianya bersifat bisa dibalik (*reversible reaction*).


**Gambar 2.6 Baterai**

## 2.7 IC Regulator

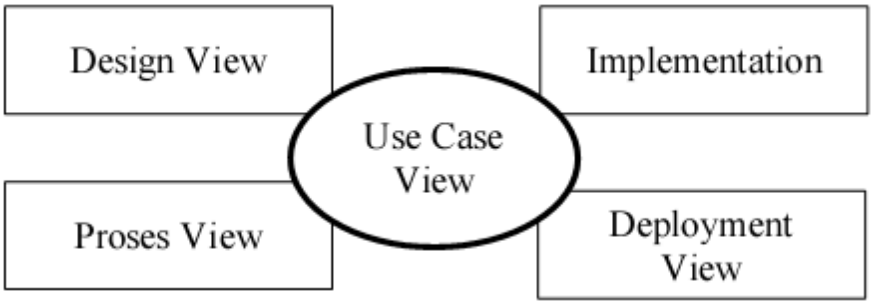
Regulator adalah Suplai daya atau tegangan catu suatu rangkaian elektronik yang berubah-ubah besarnya dapat menyebabkan pengaruh yang sifatnya merusak fungsi kerja rangkaian elektronik yang dicatunya. Catu daya yang stabil dan dapat diatur sering disebut dengan *regulated power supply*. Catu daya ini menggunakan komponen aktif sehingga harganya cukup mahal. Maka dari itu, saat ini banyak digunakan catu daya dalam bentuk IC yaitu IC regulator tegangan. IC regulator tegangan secara garis besar dapat dibagi menjadi dua yakni regulator tegangan tetap (3 kaki) dan regulator tegangan yang dapat diatur (3 kaki atau lebih). (Randy, 2012)


**Gambar 2.7 IC Regulator**

## 2.8 *Unified Modelling Language* (UML)

(*Unified Modeling Language*) adalah metode pemodelan secara visual sebagai sarana untuk merancang dan atau membuat software berorientasi objek. Karena UML ini merupakan bahasa visual untuk pemodelan bahasa berorientasi objek, maka semua elemen dan diagram berbasiskan pada paradigma *object oriented*. UML adalah salah satu *tool* / model untuk merancang pengembangan software yang berbasis object oriented. UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponenkomponen yang diperlukan dalam sistem software. (Nugroho, 2005)

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).




**Gambar 2.8 UML dibangun atas model 4+1 view**

1. *Use Case View*

   Mendefinisikan perilaku eksternal sistem.

2. *Design View*

   Mendefinisikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan *use case* (computer program, kelas-kelas, interkasi)

3. *Implementation View*

   Menjelaskan komponen-komponen fifik dari sistem (*file exe, library, database*).

4. *Process View*

   Berkaitan dengan concurrency di dalam sistem *Deployment View* menjelaskan bagaimana komponen-komponen fisik didistribusikan.

Diagram-diagram yang terdapat pada UML yaitu sebagai berikut:

1. *Diagram Use Case*

*Diagram Use Case* menggambarkan apa saja aktifitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar. yang menjadi persoalan itu apa yang dilakukan bukan bagaimana melakukannya. Diagram Use Case dekat kaitannya dengan kejadian-kejadian. Kejadian (*scenario*) merupakan contoh apa yang terjadi ketika seseorang berinteraksi dengan sistem.

*2. Class Diagram*

*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

*3. Statechart Diagram*

*Statechart* atau biasa disebut dengan state diagram digunakan untuk mendokumentasikan beragam kondisi atau keadaan yang terjadi terhadap sebuah class dan kegiatan apa saja yang dapat merubah keadaan/kondisi tersebut. Pada umumnya statechart diagram menggambarkan class tertentu (satu class dapat memiliki lebih dari satu *statechart diagram*). Transisi antar state umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan. *Action* ( aktifitas: menjalankan atau membuat state berubah) yang dilakukan sebagai akibat dari event ( penyebab terjadinya perubahan ).

*4. Diagram Sequence*

*Diagram Class* dan diagram *Object* merupakan suatu gambaran model statis. Namun ada juga yang bersifat dinamis, seperti *Diagram Interaction*. *Diagram sequence* merupakan salah satu *diagram Interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan; *message* (pesan) apa yang dikirim dan kapan pelaksanaannya.

Tabel 2.1 Simbol-Simbol UML

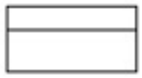
*1.* Simbol *Use Case Diaram*

| No | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1 |  | ***Actor*** | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |

| | | | |
|---|---|---|---|
| 2 |  | ***Dependency*** | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*). |
| 3 |  | ***Generalization*** | Hubungan dimana objek anak (*descendent*) berbagai perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
| 4 |  | ***Include*** | Menspesifikasikan bahwa *use case* sumber secara *eksplisit*. |
| 5 |  | ***Extend*** | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 6 |  | ***Association*** | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7 |  | ***System*** | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8 |  | ***Use Case*** | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor. |
| 9 |  | ***Collaboration*** | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |

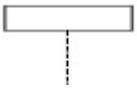
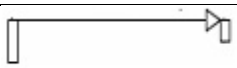
| | | | |
|---|---|---|---|
| 10 |  | ***Note*** | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi. |

*2.* Simbol *Class Diagram*

| No | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1. |  | *Generalization* | Hubungan dimana objek anak (*descendent*) berbagai perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*). |
| 2. |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| 3. |  | *Class* | Himpunan dari objek-objek yang berbagai atribut serta operasi yang sama. |
| 4. |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yag ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor. |
| 5. |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 6. |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*indepenent*) akan mempengaruhi elemen yang |

| | | | |
|---|---|---|---|
| | | | bergantung padanya elemen yang tidak mandiri. |
| 7. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |

*3.* Simbol *Sequence Diagram*

| No | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1. |  | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. |
| 2. |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. |

4. Simbol *Chart Diagram*

| No | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1. |  | *State* | Nilai atribut dan nilai link pada suatu waktu tertentu, yang dimiliki oleh suatu objek. |
| 2. |  | *Initial Pseudo State* | Bagaimana objek dibentuk atau diawali. |
| 3. |  | *Final State* | Bagaimana objek dibentuk dan dihancurkan |
| 4. |  | *Transition* | Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya. |
| 5. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |

| | | | |
|---|---|---|---|
| 6. |  | *Node* | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi. |

*5.* Simbol *Activity Diagram*

| No | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1. |  | *Actifity* | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain. |
| 2. |  | *Action* | State dari sistem yang mencerminkan eksekusi dari suatu aksi. |
| 3. |  | *Initial Node* | Bagaimana suatu objek dibentuk dan diawali. |
| 4. |  | *Actifity Final Node* | Bagaimana objek dibentuk dan dihancurkan. |
| 5. |  | *Fork Node* | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran. |

**2.9 Bahasa Pemograman C**

Bahasa pemrograman C merupakan salah satu bahasa pemrograman komputer. Dibuat pada tahun 1972 oleh Dennis Ritchie untuk Sistem Operasi Unix di Bell Telephone Laboratories. Meskipun C dibuat untuk memprogram sistem dan jaringan komputer namun bahasa ini juga sering digunakan dalam mengembangkan software aplikasi. C juga banyak dipakai oleh berbagai jenis platform sistem operasi dan arsitektur komputer, bahkan terdapat beberepa compiler yang sangat populer telah tersedia. (Rohmanah, 2013)

### 2.9.1 Mengkompilasi Program

Suatu source program C baru dapat dijalankan setelah melalui tahap kompilasi dan penggabungan. Tahap kompilasi dimaksudkan untuk memeriksa source-program sesuai dengan kaidah-kaidah yang berlaku di dalam bahasa pemrograman C. Tahap kompilasi akan menghasilkan relocatable object file. File-file objek tersebut kemudian digabung dengan perpustakaan-fungsi yang sesuai. untuk menghasilkan suatu executable-program.

Shortcut yang digunakan untuk mengkompile:

1. ALT + F9 dipakai untuk melakukan pengecekan jika ada error pada program yang telah kita buat.
2. CTRL + F9 dipakai untuk menjalankan program yang telah kita buat atau bisa juga dengan mengklik tombol debug pada toolbar.

### 2.9.2 Struktur Bahasa Pemograman C

Terdapat banyak struktur dalam Pemrograman C yaitu:

1. Header File

Adalah berkas yang berisi prototype fungsi. definisi konstanta. dan definisi variable. Fungsi adalah kumpulan code C yang diberi nama dan ketika nama tersebut dipanggil maka kumpulan kode tersebut dijalankan.

Contoh :

stdio.h

math.h

conio.h

2. Preprosesor Directive (#include)

Preprosesor directive adalah bagian yang berisi pengikutsertaan file atau berkas-berkas fungsi maupun pendefinisian konstanta.

Contoh:

#include <stdio.h>

#include phi 3.14

3. Void

Artinya fungsi yang mengikutinya tidak memiliki nilai kembalian (return).

4. Main ( )

Fungsi main ( ) adalah fungsi yang pertama kali dijalankan ketika program dieksekusi.tanpa fungsi main suatu program tidak dapat dieksekusi namun dapat dikompilasi.

5. Statement

Statement adalah instruksi atau perintah kepada suatu program ketika program itu dieksekusi untuk menjalankan suatu aksi. Setiap statement diakhiri dengan titik-koma(;).

### 2.9.3 Kata Kunci (Keyword)

Kata kunci-kata kunci yang terdapat di C, sebagai berikut:

Tabel 2.2 Kata Kunci yang Terdapat di Bahasa C

| | | | |
|---|---|---|---|
| auto | break | case | Char |
| const | continue | default | Do |
| double | else | enum | Extern |
| float | for | goto | If |
| int | long | register | Return |
| short | signed | sizeof | static |
| struct | switch | typedef | union |
| unsigned | void | volatile | while |

### 2.9.4 Identifier

Identifier atau nama pengenal adalah nama yang ditentukan sendiri oleh pemrogram yang digunakan untuk menyimpan nilai, misalnya nama variable, nama konstanta, nama suatu elemen (misalnya: nama fungsi, nama tipe data, dll). Identifier punya ketentuan sebagai berikut :

a. Maksimum 32 karakter (bila lebih dari 32 karakter maka yang diperhatikan hanya 32 karakter pertama saja).
b. Case sensitive: membedakan huruf besar dan huruf kecilnya.

c. Karakter pertama harus karakter atau underscore ( _ ) selebihnya boleh angka.
d. Tidak boleh mengandung spasi atau blank.
e. Tidak boleh menggunakan kata yang sama dengan kata kunci dan fungsi.

### 2.9.5 Variabel

Variabel adalah identifier yang nilainya dapat berubah atau diubah selama program berjalan (dieksekusi). Pengubahnya adalah user atau proses.

- Deklarasi variabel (tipe_data nama_variabel;)

  Variabel yang akan digunakan dalam program haruslah dideklarasikan terlebih dahulu. Pengertian deklarasi di sini berarti memesan memori dan menentukan jenisdata yang bisa disimpan di dalamnya.

  Contoh:

  Int a, b, c;

  -Inisialisasi variabel (tipe_data nama_variabel = nilai;)

  Int a=12, b=7, c=0;

### 2.9.6 Konstanta

Konstanta adalah identifier yang nilainya tetap selama program berjalan/dieksekusi. Cara untuk mengubahnya hanya melalui source codenya saja seperti halnya variabel, konstanta juga memiliki tipe. Penulisan konstanta mempunyai aturan
tersendiri, sesuai dengan tipe masing-masing.

1. Konstanta karakter misalnya ditulis dengan diawali dan diakhiri dengan tandapetik tunggal, contohnya : ‘A’ dan ‘@’.
2. Konstanta integer ditulis dengan tanda mengandung pemisah ribuan dan tidakmengandung bagian pecahan, contohnya : –1 dan 32767.
3. Konstanta real (*float* dan *double*) bisa mengandung pecahan (dengan tanda berupa titik) dan nilainya bisa ditulis dalam bentuk eksponensial (menggunakan tanda e), contohnya : 27.5f (untuk tipe *float*) atau 27.5 (untuk tipe *double*) dan2.1e+5 (maksudnya 2,1 x 105 ).

4. Konstanta string merupakan deretan karakter yang diawali dan diakhiri dengantanda petik-ganda (“), contohnya :“Pemrograman Dasar C”.

Contoh :

```
#define phi 3.14
#define max_data 50
```

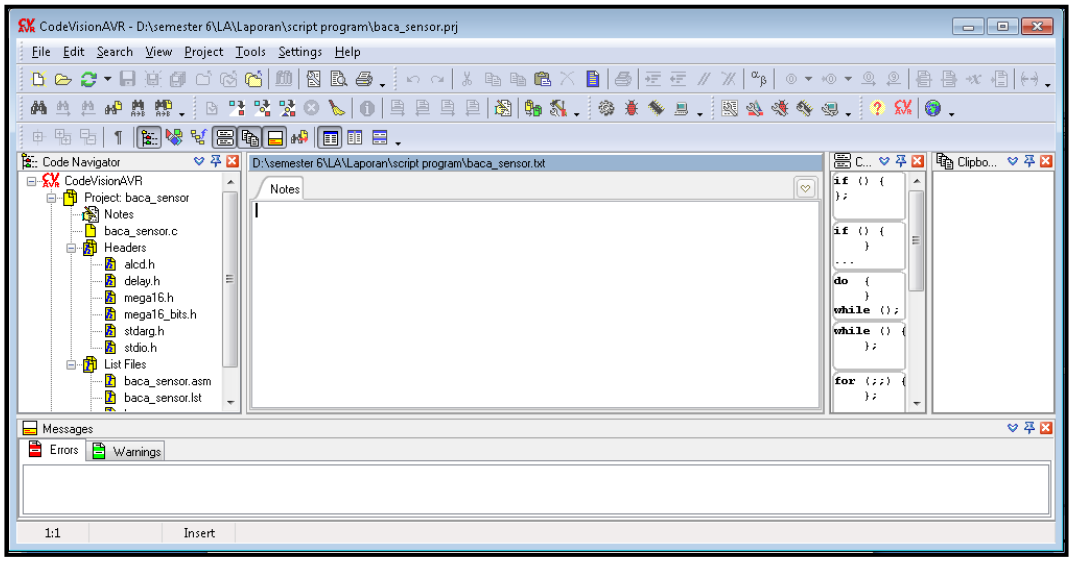
## 2.10 CodeVisionAVR

CodeVisionAVR merupakan sebuah *cross-compiler* C, *Integrated Development Environment* (IDE) dan *Automatic Program Generator* yang didesain untuk mikrokontroller buatan Atmel seri AVR. CodeVisionAVR dapat dijalankan pada *operating system* Windows 95, 98, Me, NT4, 2000, XP, Vista dan 7 (*Seven*). (Hendawan, 2009)

*Cross-compiler* C mampu menjalankan hampir semua perintah dari bahasa ANSI C, sejauh yang diizinkan oleh arsitektur AVR dengan tambahan beberapa fitur untuk mengambil kelebihan khusus dari arsitektur AVR dan kebutuhan pada sistem *embedded*. File *object* COFF hasil kompilasi dapat digunakan untuk keperluan *debugging* pada tingkatan C, dengan pengamatan variabel menggunakan *debugger* Atmel AVR Studio.

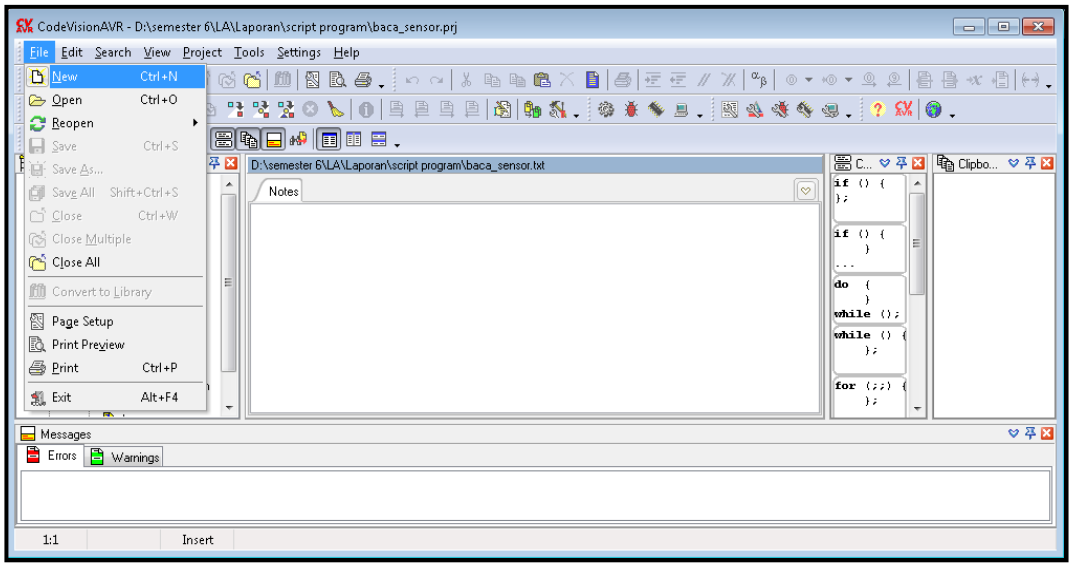
IDE mempunyai fasilitas internal berupa *software* AVR *Chip In-System Programmer* yang memungkinkan untuk melakukan transfer program kedalam *chip* mikrokontroller setelah sukses melakukan kompilasi/asembli secara otomatis. *Software In-System Programmer* didesain untuk bekerja dengan Atmel STK500/AVRISP/AVRProg, Kanada *System* STK200+/300, Dontronics DT006, Vogel Elektronik VTEC-ISP, Futurlec JRAVR dan MicroTronics ATCPU/Mega2000 *programmers/development boards*. Untuk keperluan *debugging* sistem *embedded* yang menggunakan komunikasi serial, IDE mempunyai fasilitas internal berupa sebuah terminal.


**Gambar 2.9 Tampilan Awal *Splash Screen* CodeVisionAVR**



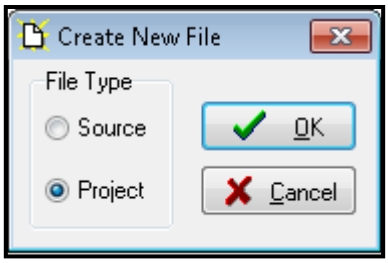

**Gambar 2.10 IDE CodeVisionAVR**

Untuk memulai *project* baru, pilih File > New.



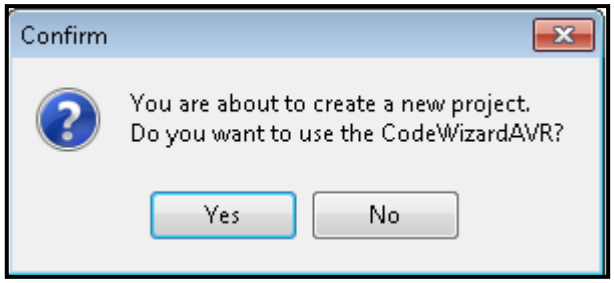

**Gambar 2.11 Membuat File Baru pada CodeVisionAVR**

Buatlah sebuah *project* sebagai induk desain dengan memilih *project* lalu klik tombol Ok.




**Gambar 2.12 Membuat *project* baru**

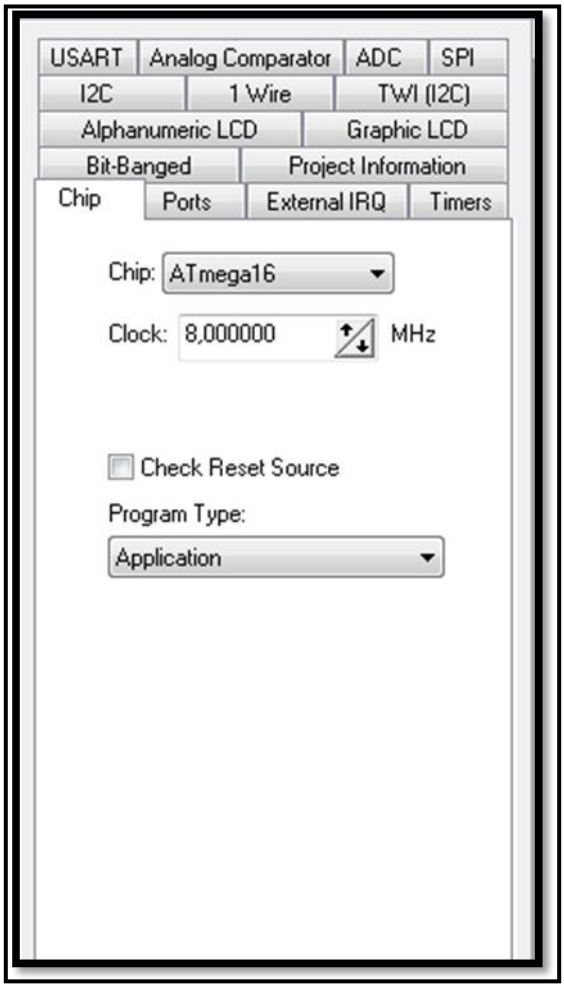
Berikutnya akan ditanya apakah akan menggunakan CodeWizardAVR, lalu pilih tombol Yes.




**Gambar 2.13 Memilih Untuk Menggunakan CodeWizardAVR**

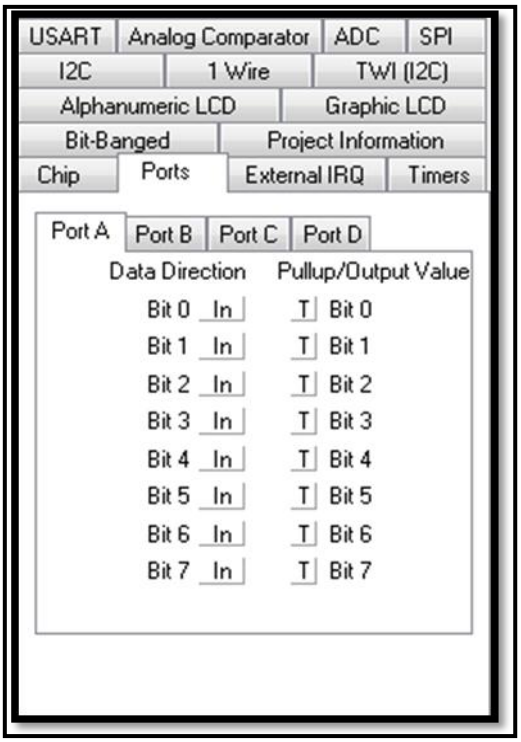
Pilih chip dengan IC yang digunakan. Tab-tab pada CodeWizardAVR menunjukkan fasilitas yang dimiliki oleh chip yang dipilih. Cocokkan pula

frekuensi kristal yang digunakan pada bagian *clock*. Pengisian frekuensi *clock* digunakan oleh software untuk menghitung rutin-rutin seperti *delay* agar diperoleh perhitungan yang akurat.



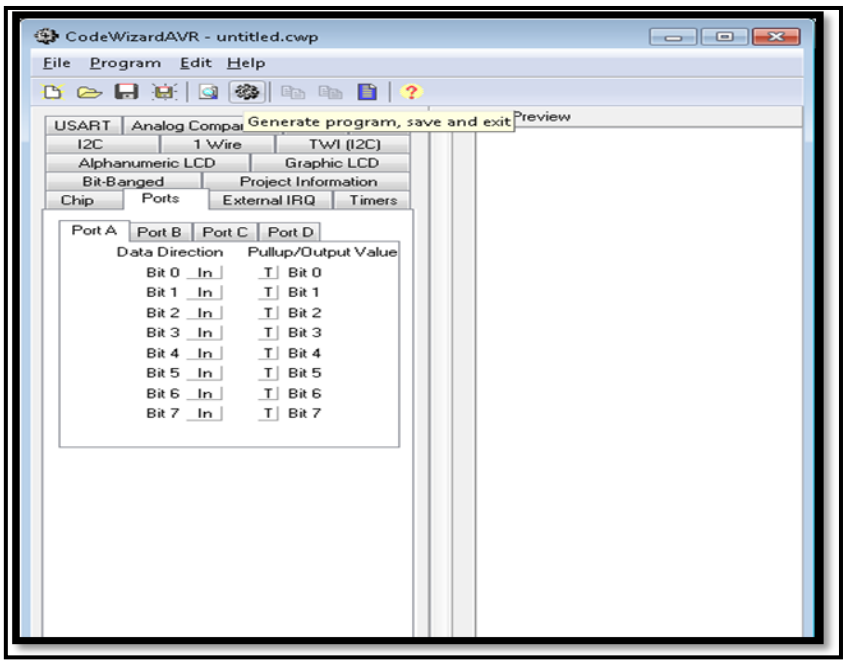

**Gambar 2.14 CodeWizardAVR pada Tab Chip**

Pada bagian ini diberi kesempatan untuk mengatur *ports-ports* yang akan digunakan. Kemudian lakukan inisialisasi *port* yang akan digunakan sebagai *input* dan *output*.



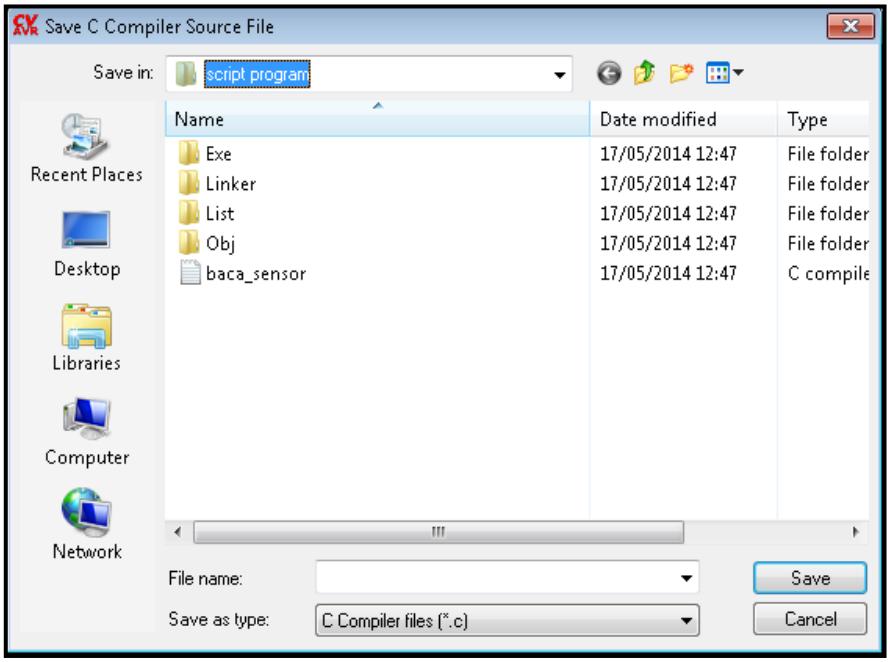

**Gambar 2.15 *Setting Port***

Jika telah selesai, pilih File > Generate, Save and Exit untuk menyimpan *setting* yang telah dibuat pada menu CodeWizardAVR.



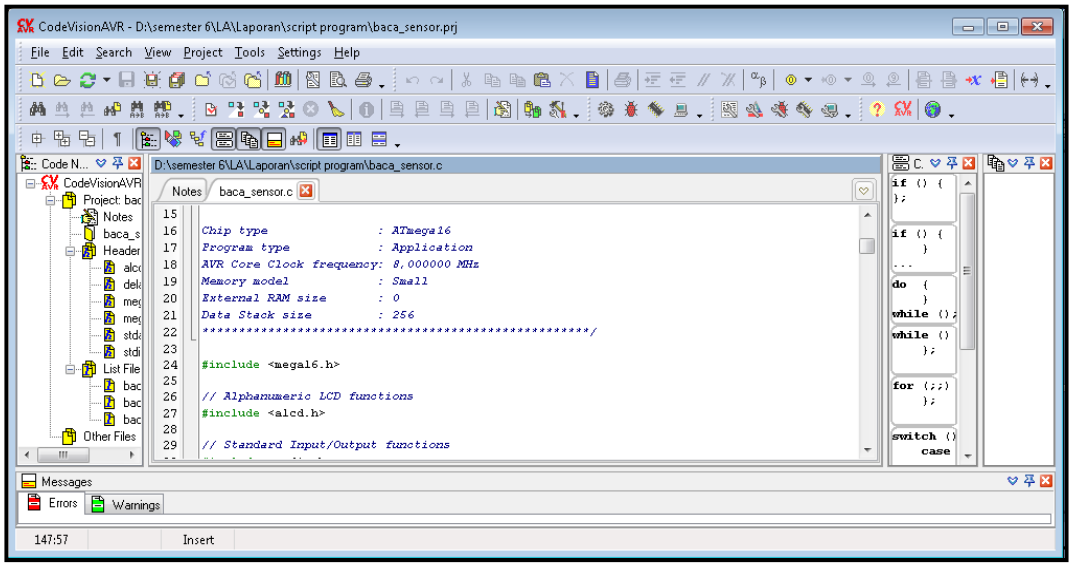

**Gambar 2.16 Menyimpan *setting***

Proses menyimpan file dilakukan sebanyak tiga kali, masing-masing menghasilkan ekstensi *.C, *.prj dan *.cwp. Pilih lokasi penyimpanan dan beri nama *project*.



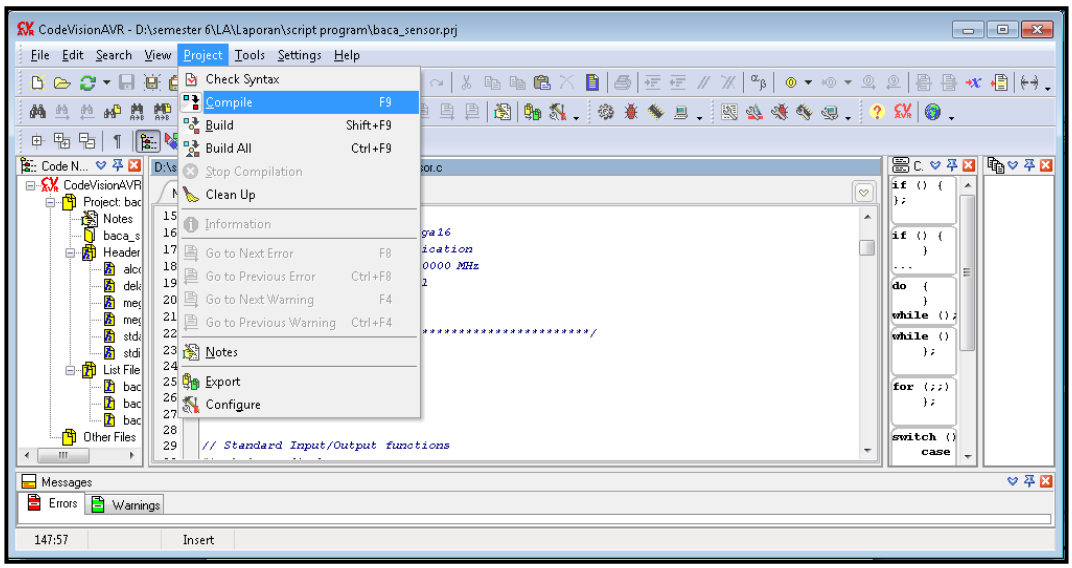

**Gambar 2.17 Menyimpan File**

Setelah proses menyimpan selesai, kemudian tampil seperti gambar dibawah ini.



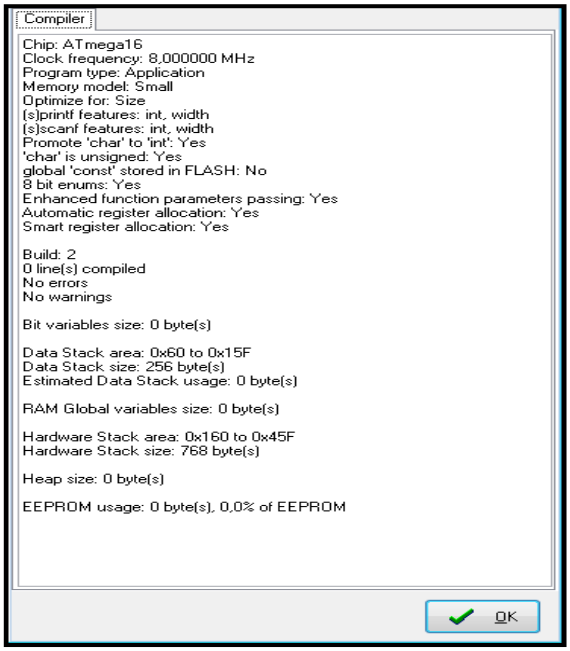

**Gambar 2.18 *Project* baru**

Setelah selesai menuliskan program, selanjutnya klik *compile* untuk menghasilkan ekstensi *.hex.




**Gambar 2.19 Melakukan Proses *Compile***

Kemudian akan tampil jendela informasi seperti dibawah ini:




**Gambar 2.20 Informasi Hasil *Compile***

Program yang telah dibuat siap untuk ditransfer kedalam mikrokontroler. File ini dapat ditemukan didalam folder exe.

Beberapa kelebihan yang dimiliki oleh CodeVisionAVR antara lain (Agus, 2008) :

1. Menggunakan IDE (*Integrated Development Environment*).
2. Fasilitas yang disediakan lengkap (mengedit program, mengkompile program, mendownload program) serta tampilannya terlihat menarik dan mudah dimengerti. Kita dapat mengatur setingan editor sedemikian rupa sehingga membantu memudahkan kita dalam penulisan program.
3. Mampu membangkitkan kode program secara otomatis dengan menggunakan fasilitas CodeVisionAVR.
4. Memiliki fasilitas untuk mendownload program langsung dari CodeVisionAVR dengan menggunakan *hardware* khusus seperti Atmel STK500, Kanada *System* STK200+/300 dan beberapa *hardware* lain yang telah didefinisikan oleh CodeVisionAVR.
5. Memiliki fasilitas *debugger* sehingga dapat menggunakan *software compilere* lain untuk mengecek kode assembler, contohnya AVRStudio.
6. Memiiki terminal komunikasi serial yang terintegrasi dalamCodeVisionAVR sehingga dapat digunakan untuk membantu pengecekan program yang telah dibuat khususnya yang menggunakan fasilitas komunikasi serial UART.