

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Pada penelitian sebelumnya yang dilakukan oleh A. Ejah Umraeni Salam dan Cristophorus Yohannes tahun 2011 dalam jurnal yang berjudul “Pengukur Tinggi Badan Dengan Detektor Ultrasonik”. Dalam penelitian ini menjelaskan alat ukur tinggi badan yang sering digunakan saat ini yaitu alat ukur tinggi badan konvensional berupa pita meteran dan segitiga siku. Penggunaan alat ukur ini sangat sederhana yaitu dengan memasang pita meteran pada dinding yang tegak lurus dengan lantai kemudian pembacaan pengukuran dilakukan dengan menggunakan segitiga siku. Maka untuk memberikan kemudahan dibuatlah alat ukur tinggi badan digital yang dapat menghasilkan pembacaan data yang lebih teliti dan mudah dibaca. Dengan menggunakan sensor ultrasonik yang akan mengirimkan pulsa ultrasonik apabila mengenai suatu objek maka pulsa tersebut akan memantul dan diterima kembali oleh *receiver* sensor tersebut. Output dari sensor ultrasonik ini kemudian akan diolah dengan menggunakan mikrokontroler Atmega8535 kemudian diolah menjadi data dan data tersebut dapat dibaca dengan menggunakan alat display berupa Lcd. Hasil yang dicapai pada pengujian pengukuran tinggi badan dilakukan dengan permukaan objek pendeteksi sensor berupa bidang datar dan tidak datar. Untuk bidang datar hasil pengukuran terjadi pergeseran 0.1 – 0.9 cm disebabkan karena pembacaan alat ukur tidak memakai tipe data *float* (bilangan desimal) melainkan memakai tipe data *integer* (bilangan bulat), sehingga tinggi yang nilainya bukan bilangan bulat akan dibulatkan oleh mikrokontroler. Dan pada bidang tidak datar hasil keakuratannya lebih sedikit dibandingkan pada bidang yang datar dikarenakan rata-rata tinggi badan yang bergeser adalah tinggi badan koresponden yang kepalanya tidak datar bersandar penuh pada dinding tempat pengukuran, hal ini disebabkan oleh beberapa hal seperti koresponden perempuan yang mengikat rambutnya kebelakang, koresponden yang bergerak pada saat pengukuran dan perbedaan bentuk kepala serta ketebalan rambut koresponden. Maka kesimpulan yang didapat yaitu

pengukur tinggi badan dengan detektor ultrasonik dapat bekerja dengan optimal dan dapat digunakan untuk mengukur tinggi badan serta untuk mendapatkan pengukuran tinggi badan yang tepat maka pada saat pengukuran objek dialasi dengan permukaan datar yang tipis misalnya penggaris.

Penelitian berikutnya yang dilakukan oleh Wildian Gusrizam Danel tahun 2012 dalam jurnal yang berjudul “Otomatisasi Keran Dispenser Berbasis Mikrokontroler At89S52 Menggunakan Sensor Fotodioda dan Sensor Ultrasonik Ping”. Pada penelitian ini menjelaskan bahwa penggunaan dispenser saat ini masih menyisakan beberapa keterbatasan, antara lain pengguna masih harus mengeluarkan energi untuk menekan keran. Selain itu, pengguna juga harus memusatkan perhatiannya agar air yang dikucurkan ke dalam cangkir tidak melimpah. Berdasarkan hal tersebut maka dibuatlah sebuah alat otomatisasi keran dispenser dengan menggunakan sensor ultrasonik agar pengguna cukup mendorong cangkir kebawah keran, lalu air minum akan mengucur dan kemudian akan berhenti dengan sendirinya saat permukaan air mencapai jarak tertentu dari sensor ultrasonik. Hasil yang diperoleh pada pengujian alat ini yaitu ketika dijalankan, alat sudah dapat bekerja dengan baik. Ketika cangkir diletakkan antara LED dan Fotodioda keran langsung hidup dan air segera mengisi cangkir. Ketika itu sensor PING langsung mengukur ketinggian air dalam cangkir. Ketika jarak antara sensor dan permukaan cangkir telah mencapai 5 cm, maka keran langsung mati. Namun, kadang terdapat masalah pada pembacaan sensor ultrasonik dimana ketika air dalam cangkir terisi pembacaan sensor ultrasonik menjadi tidak stabil. Hal ini menyebabkan kerja *relay* juga tidak stabil. Namun pada saat jarak permukaan air dan sensor sudah mencapai 5 cm, *relay* hidup secara stabil. Maka kesimpulan yang dapat diambil yaitu sistem otomatisasi keran dispenser berbasis mikrokontroler AT89S52 dengan sensor fotodioda dan sensor ultrasonik pada penelitian ini telah diuji dan bekerja dengan baik sesuai dengan yang diharapkan. Dan cahaya dari LED inframerah dapat menembus cangkir kaca dan cangkir plastik yang bening.

Kemudian pada penelitian selanjutnya yang dilakukan oleh Hadijaya Pratama, Erik Haritman dan Tjetje Gunawan tahun 2012 dalam jurnal yang berjudul “Akuisisi Data Kinerja Sensor Ultrasonik Berbasis Sistem Komunikasi

Serial Menggunakan Mikrokontroler Atmega 32”. Pada penelitian ini menjelaskan bahwa didalam dunia industri maupun riset, sistem akuisisi data merupakan ujung terdapan dari proses pengumpulan data secara mentah langsung dari sumbernya dimana sistem ini mengkonversi sinyal fisik menjadi sinyal elektronik dan kemudian mendigitalisasi sinyal tersebut sehingga dapat disimpan, ditransmisikan atau disajikan pada display atau komputer. Maka dari itu dibuatlah suatu perangkat keras sistem akuisisi data yang dapat mengakuisasi lebih dari satu macam besaran dan dapat menyimpan serta mentransmisikannya melalui komputer dengan menggunakan fasilitas komunikasi serial yang terdapat dalam sebuah chip mikrokontroler. Perangkat sistem ini terdiri dari sebuah modul sensor ultrasonik yang memancarkan gelombang ultrasonik setelah menerima trigger dari mikrokontroler. Setelah menerima pantulan gelombang tersebut, modul sensor ultrasonik akan mengirimkan sinyal kembali ke mikrokontroler. Metode dalam penelitian ini dilakukan dengan cara mengukur kinerja sensor ultrasonik terhadap beberapa material, seperti objek benda berwarna hitam, objek benda berwarna putih, kaca dan permukaan objek yang tidak rata. Hasil pengujian terhadap objek benda hitam, putih dan kaca tidak mengalami perubahan yang signifikan sedangkan pengujian terhadap objek dengan permukaan yang tidak rata mengalami pengukuran dengan jarak terjauh dari objek benda tersebut. Dengan hasil penelitian tersebut dapat diambil kesimpulan bahwa sensor ultrasonik dapat mendeteksi objek tanpa terpengaruh perbedaan warna benda ataupun kaca dan akan mendeteksi jarak terjauh dari posisi objek didepan sensor. Secara umum semakin jauh jarak yang diukur, semakin besar persen kesalahan yang terjadi.

2.1.1 Perbedaan dengan Penelitian Sebelumnya

Dalam laporan akhir yang berjudul “Alat Ukur Panjang Bayi Digital Menggunakan Sensor Ultrasonik Berbasis Mikrokontroler Atmega 16 Dengan Visual Basic 6.0”. Pada rancang bangun alat ini menggunakan sensor ultrasonik sebagai sensor utama yang berfungsi sebagai input pembacaan jarak yang menghitung jarak horizontal menuju media pantul yang disediakan melalui pembatas ruang ukur. Sensor ultrasonik akan mengirimkan (*transmitter*) pulsa ultrasonik yang apabila mengenai pembatas ruang ukur maka pulsa tersebut akan memantul dan diterima kembali oleh *receiver* sensor tersebut. Pulsa yang

memantul inilah yang dipakai sebagai indikator jarak pembatas ruang ukur dengan sensor yang kemudian jarak tersebut akan dikurangi dengan jarak antara sensor dan seluruh media ukur. Output dari sensor ultrasonik ini akan diolah dengan menggunakan mikrokontroler Atmega 16 kemudian diolah menjadi data dan data tersebut akan ditampilkan melalui Lcd. Data yang dihasilkan akan disalurkan melalui rangkaian komunikasi serial yang akan terhubung pada PC dan akan tersimpan pada tempat penyediaan database yaitu dengan menggunakan aplikasi Visual Basic 6.0.

2.2 Alat Ukur Panjang Bayi

Pengukuran adalah salah satu hal yang banyak didigitalisasi. Seperti pengukur kecepatan, jarak, tegangan listrik, arus listrik dan banyak lainnya yang semula berjenis analog. Mengukur pada hakekatnya membandingkan suatu besaran yang belum diketahui besarnya dengan besaran lain yang diketahui besarnya (Syaryadhi, 2008).

Alat ukur panjang bayi adalah suatu alat yang berfungsi untuk mengukur nilai suatu besaran panjang atau tinggi bayi berdasarkan satuan tertentu.

Alat ukur panjang bayi digital didefinisikan sebagai piranti mekanik yang mampu mempermudah pekerjaan manusia. Pekerjaan manusia yang dapat dipermudah oleh alat ukur panjang bayi digital tersebut adalah mengukur panjang bayi tanpa adanya bantuan orang lain.

2.3 Sensor Ultrasonik

Ultrasonik sebutan untuk jenis suara diatas batas yang bisa didengar oleh manusia. Jenis suara ini dapat didengar oleh beberapa binatang seperti kelelawar dan lumba-lumba, dan digunakan sebagai pengindra untuk penanda benda yang ada di depannya (Misnawati, 2008).

Gelombang ultrasonik merupakan gelombang longitudinal dengan frekuensi rata-rata diatas 17 kHz. Gelombang ultrasonik termasuk kedalam gelombang bunyi yang dapat merambat melalui medium padat, cair dan gas (Misnawati, 2008).

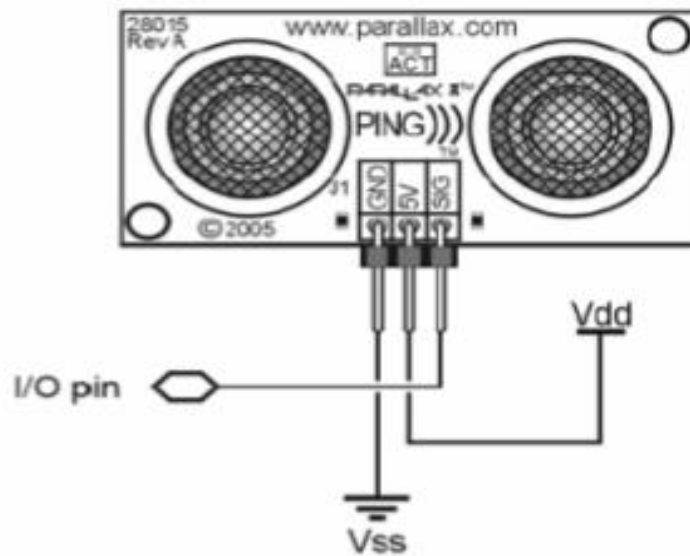
Sensor PING merupakan jenis sensor ultrasonik yang bekerja melalui pemancaran gelombang bunyi dengan frekuensi sumber 40 kHz dan kecepatan

344 m/s. Selanjutnya PING akan menerima pantulan, lalu mengirim sinyal logika (Misnawati, 2008).

Sensor PING bekerja dengan mentransmisikan gelombang ultrasonik dan menghasilkan pulsa keluaran yang sesuai dengan waktu tempuh untuk pemancaran dan pemantulan gelombang. Dengan menghitung waktu tempuh dari pulsa maka jarak sensor dengan target dapat dengan mudah dihitung. Jarak yang terukur oleh gelombang ultrasonik yaitu:

$$h = \frac{v_u}{2} \times \frac{t_{tn}}{2} = \frac{344 \times t_{tn}}{2}$$

Pada persamaan diatas, h adalah jarak yang diukur oleh gelombang ultrasonik. Besaran v_u adalah kecepatan gelombang ultrasonik di udara yaitu 344 meter per detik. Besaran t_{tn} adalah lebar pulsa atau waktu tempuh, gelombang ultrasonik untuk dua kali jarak ukur dengan objek pulang dan pergi. Karena kecepatan udara tetap, maka waktu tempuh gelombang ultrasonik hanya tergantung pada jarak yang diukur (Misnawati, 2008). Sensor jarak ultrasonik dapat dilihat pada gambar 2.1.



Gambar 2.1. Sensor Jarak Ultrasonik

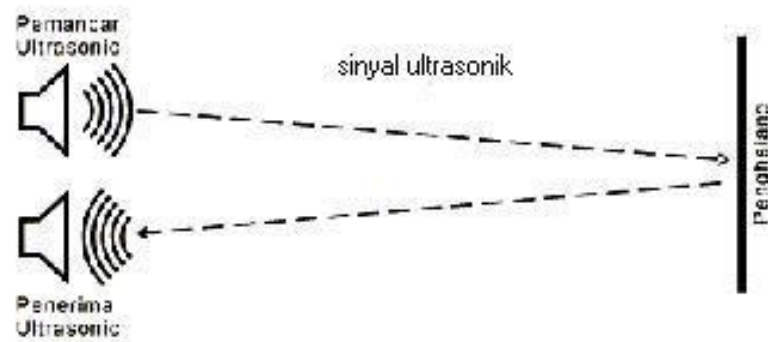
Penjelasan masing-masing pin Sensor jarak ultrasonik adalah sebagai berikut:

1. Vdd merupakan pin masukan catu daya
2. Vss merupakan pin ground
3. SIG merupakan pin I / O

2.3.1 Prinsip Kerja Rangkaian Sensor Ultrasonik

Gelombang ultrasonik adalah gelombang dengan besar frekuensi diatas frekuensi gelombang suara yaitu lebih dari 20 KHz. Seperti telah disebutkan bahwa sensor ultrasonik terdiri dari rangkaian pemancar ultrasonik yang disebut *transmitter* dan rangkaian penerima ultrasonik yang disebut *receiver*. Sinyal ultrasonik yang dibangkitkan akan dipancarkan dari *transmitter* ultrasonik. Ketika sinyal mengenai benda penghalang, maka sinyal ini dipantulkan, dan diterima oleh *receiver* ultrasonik. Sinyal yang diterima oleh rangkaian *receiver* dikirimkan ke rangkaian mikrokontroler untuk selanjutnya diolah untuk menghitung jarak terhadap benda di depannya (bidang pantul).

Prinsip kerja dari sensor ultrasonik dapat ditunjukkan dalam gambar 2.2.



Gambar 2.2. Prinsip Kerja Sensor Ultrasonik

Prinsip kerja dari sensor ultrasonik adalah sebagai berikut :

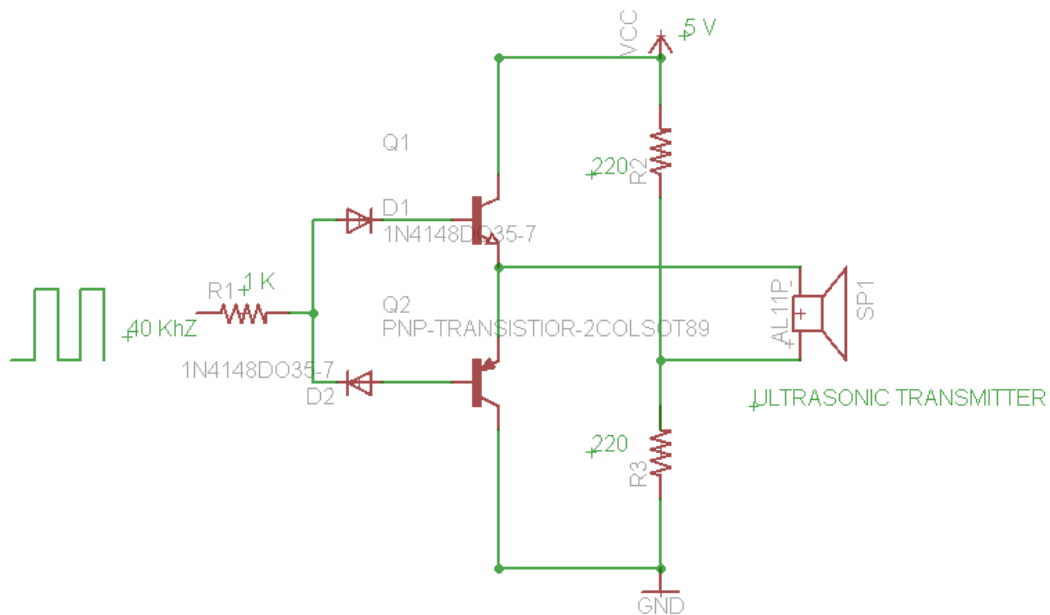
1. Sinyal dipancarkan oleh pemancar ultrasonik. Sinyal tersebut berfrekuensi diatas 20kHz, biasanya yang digunakan untuk mengukur jarak benda adalah 40kHz. Sinyal tersebut di bangkitkan oleh rangkaian pemancar ultrasonik.
2. Sinyal yang dipancarkan tersebut kemudian akan merambat sebagai sinyal / gelombang bunyi dengan kecepatan bunyi yang berkisar 340 m/s. Sinyal tersebut kemudian akan dipantulkan dan akan diterima kembali oleh bagian penerima Ultrasonik.

3. Setelah sinyal tersebut sampai di penerima ultrasonik, kemudian sinyal tersebut akan diproses untuk menghitung jaraknya. Jarak dihitung berdasarkan rumus $S = 340.t/2$

Dimana S adalah jarak antara sensor ultrasonik dengan bidang pantul, dan t adalah selisih waktu antara pemancaran gelombang ultrasonik sampai diterima kembali oleh bagian penerima ultrasonik.

a. Pemancar Ultrasonik (*Transmitter*)

Pemancar Ultrasonik ini berupa rangkaian yang memancarkan sinyal sinusoidal berfrekuensi di atas 20 KHz menggunakan sebuah *transducer transmitter* ultrasonik. Rangkaian pemancar gelombang ultrasonik dapat dilihat pada gambar 2.3.



Gambar 2.3. Rangkaian Pemancar Gelombang Ultrasonik

Prinsip kerja dari rangkaian pemancar gelombang ultrasonik tersebut adalah sebagai berikut :

1. Sinyal 40 kHz dibangkitkan melalui mikrokontroler.
2. Sinyal tersebut dilewatkan pada sebuah resistor sebesar 3kOhm untuk pengaman ketika sinyal tersebut membias maju rangkaian dioda dan transistor.
3. Kemudian sinyal tersebut dimasukkan ke rangkaian penguat arus yang merupakan kombinasi dari 2 buah dioda dan 2 buah transistor.

4. Ketika sinyal dari masukan berlogika tinggi (+5V) maka arus akan melewati dioda D1 (D1 on), kemudian arus tersebut akan membias transistor T1, sehingga arus yang akan mengalir pada kolektor T1 akan besar sesuai dari penguatan dari transistor.
5. Ketika sinyal dari masukan berlogika tinggi (0V) maka arus akan melewati dioda D2 (D2 on), kemudian arus tersebut akan membias transistor T2, sehingga arus yang akan mengalir pada kolektor T2 akan besar sesuai dari penguatan dari transistor.
6. Resistor R4 dan R6 berfungsi untuk membagi tegangan menjadi 2,5 V. Sehingga pemancar ultrasonik akan menerima tegangan bolak – balik dengan $V_{peak-peak}$ adalah 5V (+2,5 V s.d -2,5 V).

b. Penerima Ultrasonik (*Receiver*)

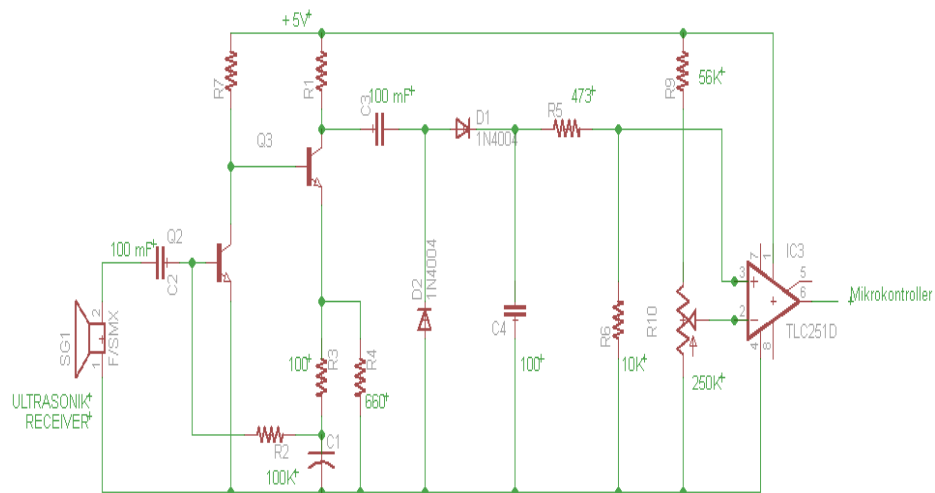
Penerima Ultrasonik ini akan menerima sinyal ultrasonik yang dipancarkan oleh pemancar ultrasonik dengan karakteristik frekuensi yang sesuai. Sinyal yang diterima tersebut akan melalui proses filterisasi frekuensi dengan menggunakan rangkaian *band pass filter* (penyaring pelewat pita), dengan nilai frekuensi yang dilewatkan telah ditentukan. Kemudian sinyal keluarannya akan dikuatkan dan dilewatkan ke rangkaian komparator (pembanding) dengan tegangan referensi ditentukan berdasarkan tegangan keluaran penguat pada saat jarak antara sensor kendaraan mini dengan sekat/dinding pembatas mencapai jarak minimum untuk berbelok arah. Dapat dianggap keluaran komparator pada kondisi ini adalah *high* (logika '1') sedangkan jarak yang lebih jauh adalah *low* (logika '0'). Logika-logika biner ini kemudian diteruskan ke rangkaian pengendali (mikrokontroler).

Prinsip kerja dari rangkaian pemancar gelombang ultrasonik tersebut adalah sebagai berikut :

1. Pertama – tama sinyal yang diterima akan dikuatkan terlebih dahulu oleh rangkaian transistor penguat Q2.
2. Kemudian sinyal tersebut akan di filter menggunakan *High pass filter* pada frekuensi > 40kHz oleh rangkaian transistor Q1.
3. Setelah sinyal tersebut dikuatkan dan di filter, kemudian sinyal tersebut akan disearahkan oleh rangkaian dioda D1 dan D2.

4. Kemudian sinyal tersebut melalui rangkaian filter *low pass filter* pada frekuensi $< 40\text{kHz}$ melalui rangkaian filter C4 dan R4.
5. Setelah itu sinyal akan melalui komparator Op-Amp pada U3.
6. Jadi ketika ada sinyal ultrasonik yang masuk ke rangkaian, maka pada komparator akan mengeluarkan logika rendah (0V) yang kemudian akan diproses oleh mikrokontroler untuk menghitung jaraknya.

Gambar rangkaian penerima gelombang ultrasonik dapat dilihat pada gambar 2.4.



Gambar 2.4. Rangkaian Penerima Gelombang Ultrasonik

2.4 Mikrokontroler

Mikrokontroler adalah sebuah sistem *microprosesor* dimana didalamnya sudah terdapat CPU, ROM, RAM, I/O, *Clock* dan peralatan internal lainnya yang sudah saling terhubung dan terorganisasi (teralamat) dengan baik oleh pabrik pembuatnya dan dikemas dalam satu *chip* yang siap dipakai. Sehingga kita tinggal memprogram isi ROM sesuai aturan penggunaan oleh pabrik yang membuatnya (Nugroho, 2012).

Dibawah ini merupakan blok diagram mikrokontroler secara umum:

a. CPU (*Central Processing Unit*)

CPU adalah suatu unit pengolah pusat yang terdiri dari dua bagian, yaitu unit pengendali (*control unit*) dan unit logika (*logic unit*). Disamping itu juga, CPU mempunyai beberapa simpanan yang berukuran kecil yang

disebut register. Adapun fungsi utama dari unit pengendali ini adalah mengatur dan mengendalikan semua peralatan yang ada pada *system computer* dan juga dapat mengatur kapan alat input menerima data dan kapan data diolah serta ditampilkan pada alat output. Sedangkan unit logika berfungsi untuk melakukan semua perhitungan aritmatika yang terjadi sesuai dengan instruksi program dan dapat juga melakukan keputusan dari operasi logika atau pengambilan keputusan sesuai dengan instruksi yang diberikan kepadanya.

b. Bus Alamat

Bus alamat berfungsi sebagai sejumlah lintasan saluran pengalamatan antara alamat dengan sebuah *computer*. Pengalamatan ini harus ditentukan terlebih dahulu untuk menghindari terjadinya kesalahan pengiriman sebuah instruksi dan terjadinya bentrok antara dua buah alat yang bekerja secara bersamaan.

c. Bus Data

Bus data merupakan sejumlah lintasan saluran keluar masuknya data dalam suatu mikrokontroler pada umumnya saluran data yang masuk sama dengan saluran data yang keluar.

d. Bus Kontrol

Bus kontrol atau bus kendali ini berfungsi menyerempetkan operasi mikrokontroller dengan operasi rangkaian luar.

e. Memori

Di dalam sebuah mikrokontroler terdapat suatu memori yang berfungsi untuk menyimpan data atau program. Ada beberapa jenis memori diantaranya adalah RAM dan ROM serta ada tingkat memori. Registrasi internal adalah memori yang terdapat di dalam ALU. Memori utama adalah memori yang ada pada suatu sistem, waktu akses lebih lambat dibandingkan *register internal*. Sedangkan memori massal dipakai untuk penyimpanan berkapasitas tinggi, yang biasanya berbentuk disket, pita magnetik atau kaset.

f. RAM

RAM adalah memori yang dapat dibaca atau ditulis. Data dalam RAM bersifat volatile dimana isinya akan hilang begitu IC kehilangan catu daya, karena sifat yang demikian RAM hanya digunakan untuk menyimpan data pada saat program bekerja.

2.4.1 Arsitektur Atmega 16

Mikrokontroler ini menggunakan arsitektur Harvard yang memisahkan memori program dari memori data, baik bus alamat maupun bus data, sehingga pengaksesan program dan data dapat dilakukan secara bersamaan (*concurrent*). Secara garis besar mikrokontroler ATMEGA 16 terdiri dari:

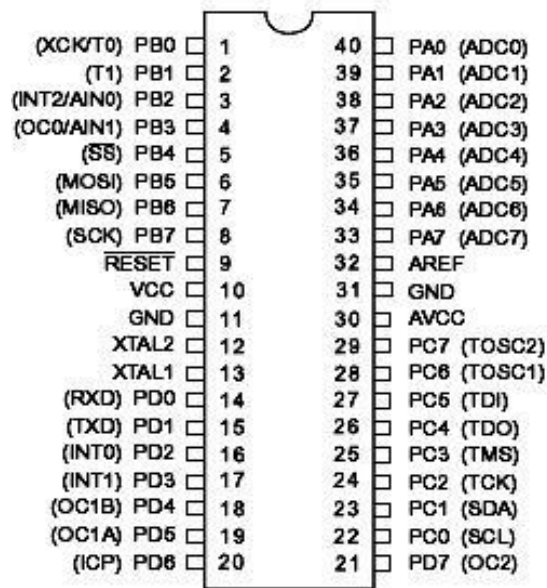
1. Arsitektur RISC dengan through put mencapai 16 MIPS pada frekuensi 16Mhz.
2. Memiliki kapasitas Flash memori 16Kbyte, EEPROM 512 byte dan SRAM 1 Kbyte.
3. Saluran I/O 32 buah, yaitu bandar A, bandar B, bandar C dan bandar D.
4. CPU yang terdiri dari 32 register.
5. User interupsi internal dan eksternal.
6. Bandar antarmuka SPI dan bandar USART sebagai komunikasi serial.

Fitur Peripheral:

1. Dua buah 8-bit *timer/counter* dengan prescaler terpisah dan mode *compare*.
2. Satu buah 16-bit *timer/counter* dengan prescaler terpisah mode *compare* dan mode *capture*.
3. *Real time counter* dengan osilator tersendiri.
4. Empat kanal PWM dan antarmuka komparator analog.
5. 8 kanal, 10 bit ADC
6. *Byte-oriented Two-wire Serial Interface*
7. *Watchdog timer* dengan osilator internal.

2.4.2 Konfigurasi Atmega 16

Konfigurasi pin ATMEGA16 dengan kemasan 40 pin DIP (*dual in-line package*) dapat dilihat pada gambar 2.5. Fungsi dari masing-masing ATMEGA16 dapat dilihat pada gambar 2.5.



Gambar 2.5. Pin-pin ATmega16 kemasan 40-pin

1. VCC merupakan pin yang berfungsi sebagai masukan catu daya.
2. GND merupakan pin Ground.
3. Port A (PA0..PA7) merupakan pin input / output dua arah dan pin masukan ADC.
4. Port B (PB0..PB7) merupakan 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 mereset 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.

2.5 LCD (*Liquid Crystal Display*)

LCD merupakan singkatan dari *Liquid Crystal Display*. Modul LCD ini dapat dengan mudah dihubungkan dengan mikrokontroler. (Nugroho, 2012). 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.

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 implementasinya 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. Bentuk LCD dapat dilihat pada gambar 2.6.



Gambar 2.6. Bentuk LCD

2.6 Komunikasi Data Serial Modul FTDI (Future Technology Device Internationals Ltd)

FTDI (Future Technology Device Internationals Ltd) merupakan IC yang dapat berkomunikasi menggunakan USB akan tetapi jika menggunakan FTDI harus memerlukan software tambahan agar dapat terdeteksi oleh PC, hal tersebut dikarenakan FTDI tidak menggunakan komunikasi kelas HID (*Human Interface Device*). Cara kerja FTDI adalah mengubah komunikasi serial ke komunikasi USB akan tetapi data yang dibaca komputer berupa data serial. Untuk masukan yang diterima FTDI hanya bisa satu tidak bisa lebih (Syahbana, 2012).

2.7 Bahasa Pemograman C

Struktur bahasa pemograman C (Andrian, 2013):

```
<preprosesor directive>
{
<statement>;
<statement>;
}
```

1. Header File

Header File adalah berkas yang berisi *prototype* fungsi, definisi konstanta dan definisi *variable*. Fungsi adalah kumpulan kode C yang diberi nama dan ketika nama tersebut dipanggil maka kumpulan kode tersebut dijalankan.

Contoh : `stdio.h`, `math.h`, `conio.h`

2. Preprosesor Directive

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

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 (;).

2.8 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*). (Andrian, 2013)

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. (Andrian, 2013)

IDE mempunyai fasilitas internal berupa *software AVR Chip In-System Programmer* yang memungkinkan untuk melakukan transfer program kedalam *chip* mikrokontroler setelah sukses melakukan kompilasi/assembly 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.

Beberapa kelebihan yang dimiliki oleh CodeVisionAVR antara lain (Andrian, 2013):

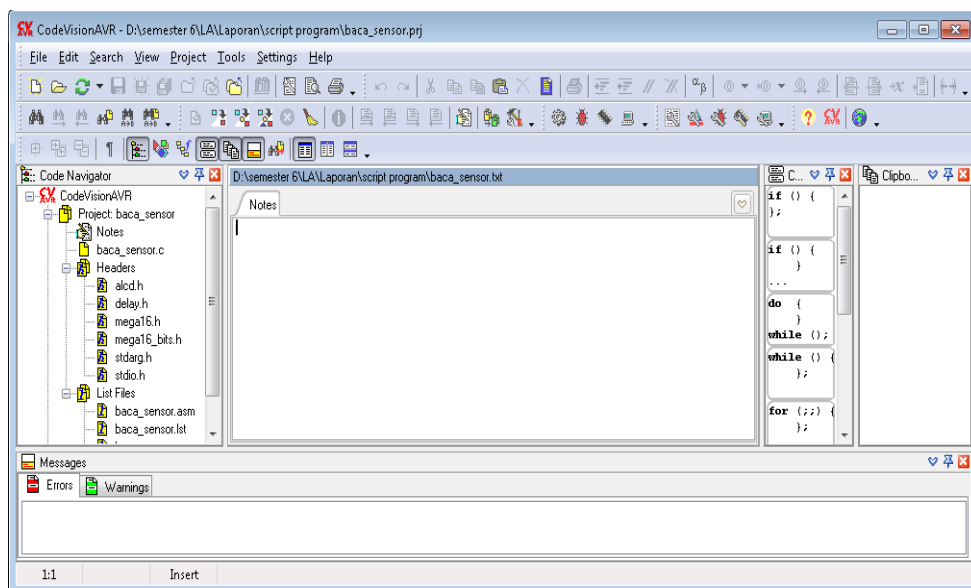
1. Menggunakan IDE (*Integrated Development Environment*).
2. Fasilitas yang disediakan lengkap (mengedit program, mengkompilasi program, mendownload program) serta tampilannya terlihat menarik dan mudah dimengerti. Kita dapat mengatur setingan editor sedemikian rupa sehingga membantu memudahkan kita 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 compiler* lain untuk mengecek kode assembler, contohnya AVRStudio.

Memiliki terminal komunikasi serial yang terintegrasi dalam CodeVisionAVR sehingga dapat digunakan untuk membantu pengecekan program yang telah dibuat khususnya yang menggunakan fasilitas komunikasi serial UART. Tampilan awal CodeVisionAVR dapat dilihat pada gambar 2.7.



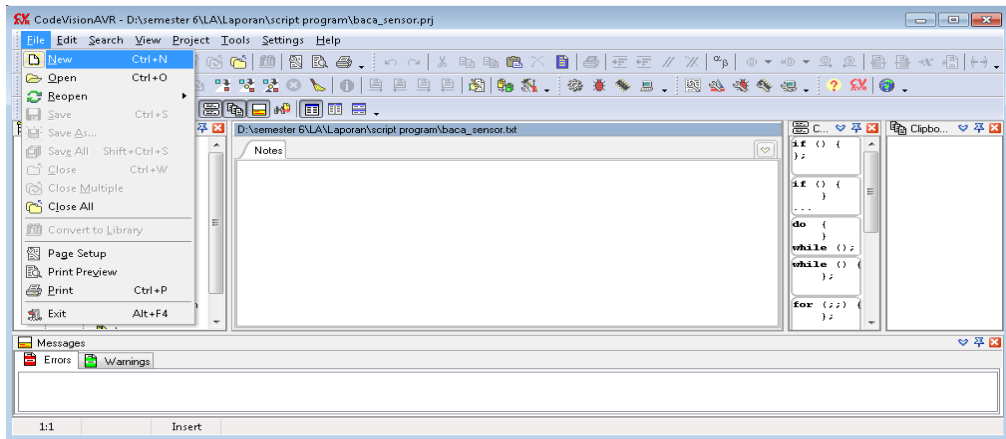
Gambar 2.7 Tampilan Awal *Splash Screen* CodeVisionAVR

Tampilan IDE CodeVisionAVR dapat dilihat pada gambar 2.8.



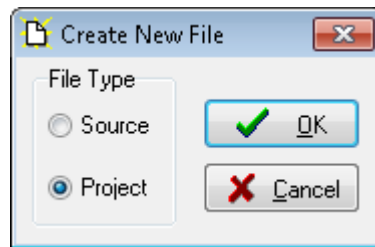
Gambar 2.8 IDE CodeVisionAVR

Untuk memulai *project* baru, pilih File > New, seperti pada gambar 2.9.



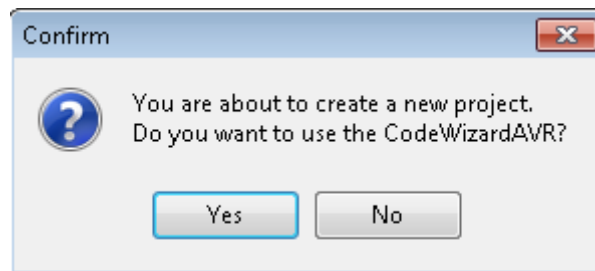
Gambar 2.9 Membuat File Baru pada CodeVisionAVR

Buatlah sebuah *project* sebagai induk desain dengan memilih *project* lalu klik tombol Ok. Membuat *project* baru seperti pada gambar 2.10.



Gambar 2.10 Membuat *project* baru

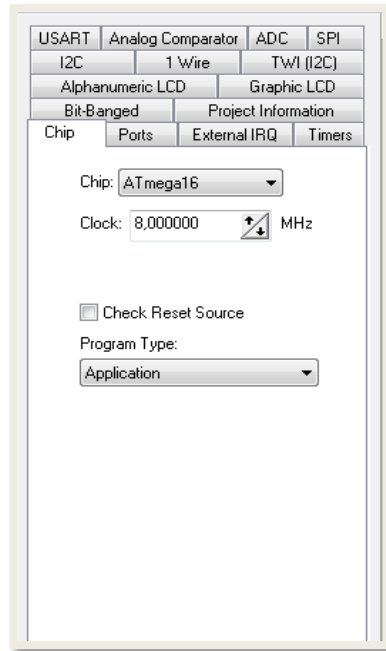
Berikutnya akan ditanya apakah akan menggunakan CodeWizardAVR, lalu pilih tombol Yes, seperti pada gambar 2.11.



Gambar 2.11 Memilih Untuk Menggunakan CodeWizardAVR

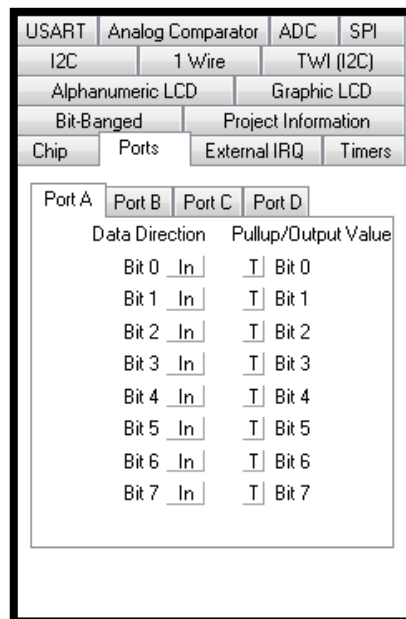
Pilih chip dengan IC yang digunakan. Tab-tab pada CodeWizardAVR menunjukkan fasilitas yang dimiliki oleh chip yang dipilih. Cocokkan pula frekuensi kristal yang digunakan pada bagian *clock*. Pengisian frekuensi *clock* digunakan oleh software untuk menghitung rutin-rutin seperti *delay* agar

diperoleh perhitungan yang akurat. Tampilan CodeWizardAVR seperti pada gambar 2.12.



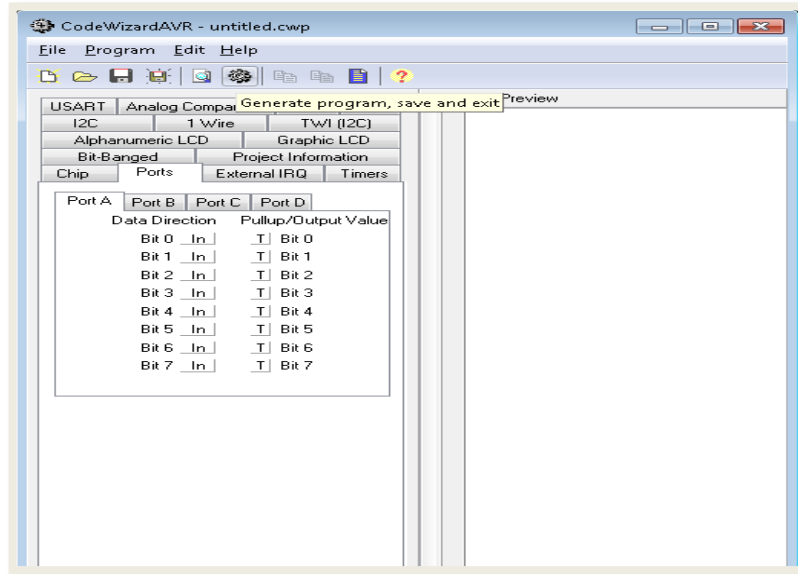
Gambar 2.12 CodeWizardAVR pada Tab Chip

Pada bagian ini diberi kesempatan untuk mengatur *ports-ports* yang akan digunakan. Kemudian lakukan inisialisasi *port* yang akan digunakan sebagai *input* dan *output*, seperti pada gambar 2.13.



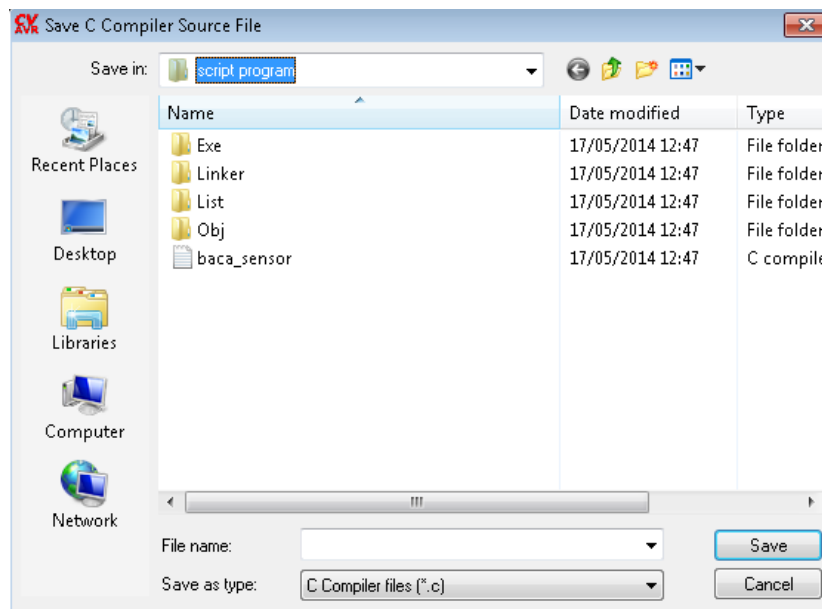
Gambar 2.13 Setting Port

Jika telah selesai, pilih File > Generate, Save and Exit untuk menyimpan *setting* yang telah dibuat pada menu CodeWizardAVR. Tampilan Menyimpan *setting* seperti gambar 2.14.



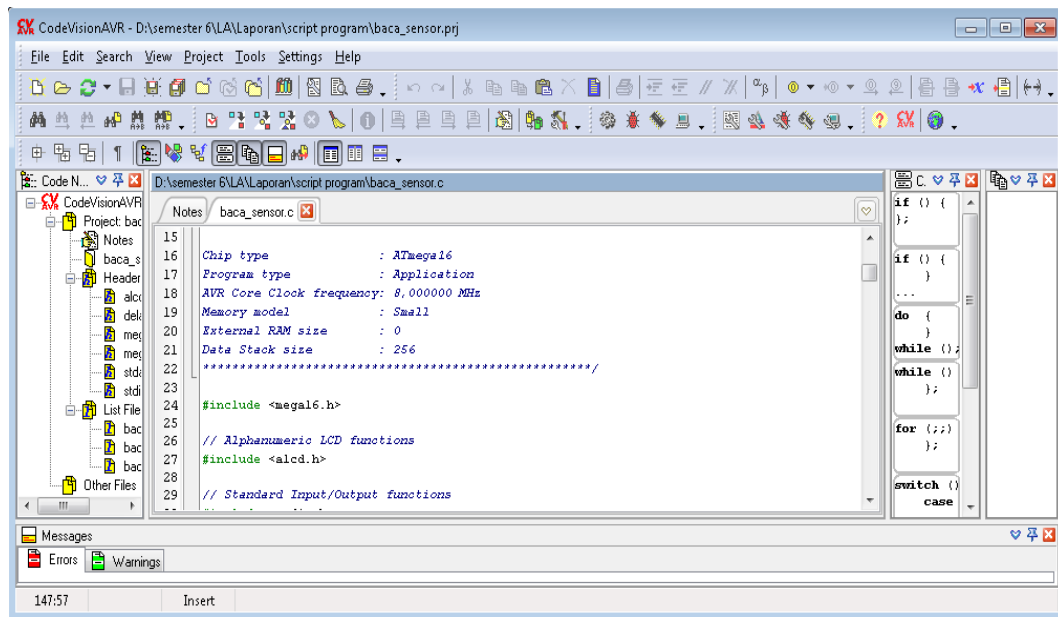
Gambar 2.14 Menyimpan *setting*

Proses menyimpan file dilakukan sebanyak tiga kali, masing-masing menghasilkan ekstensi *.C, *.prj dan *.cwp. Pilih lokasi penyimpanan dan beri nama *project*, seperti pada gambar 2.15



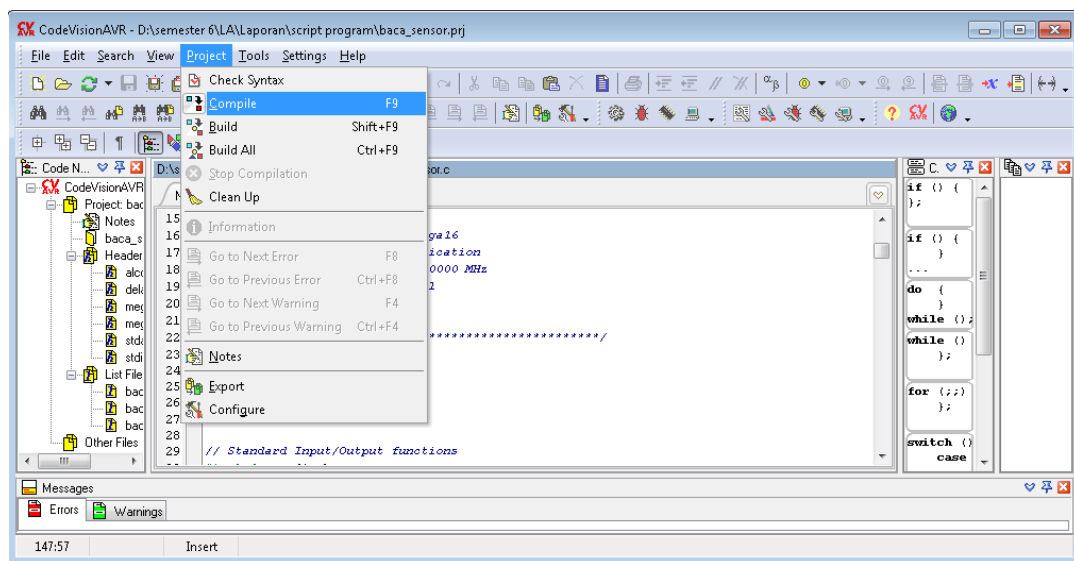
Gambar 2.15 Menyimpan File

Setelah proses menyimpan selesai, kemudian tampil seperti gambar 2.16.



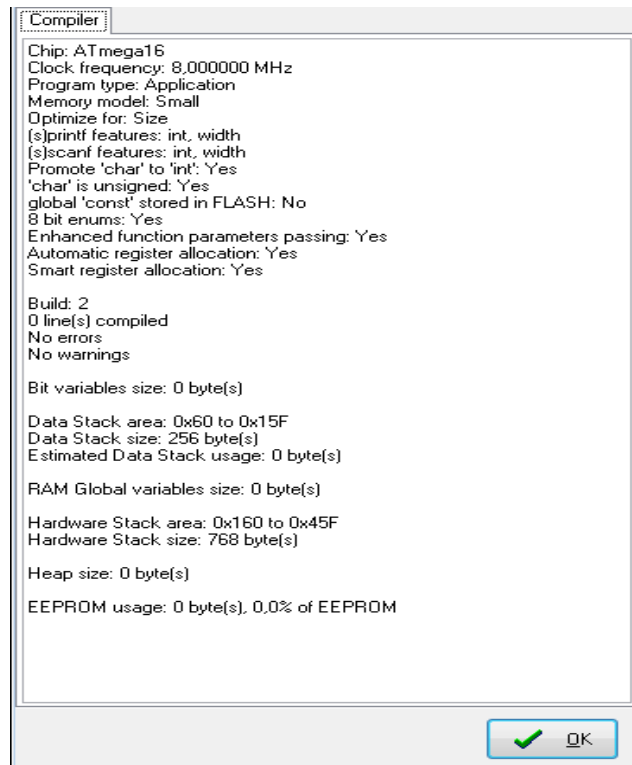
Gambar 2.16 Project baru

Setelah selesai menuliskan program, selanjutnya klik *compile* untuk menghasilkan ekstensi *.hex, seperti pada gambar 2.17.



Gambar 2.17 Melakukan Proses *Compile*

Kemudian akan tampil jendela informasi seperti gambar 2.18.



Gambar 2.18 Informasi Hasil *Compile*

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

2.9 Database

Database merupakan susunan atau kumpulan data operasional lengkap dari suatu organisasi yang dikelola dan disimpan secara terintegrasi dengan menggunakan metode tertentu, yaitu menggunakan komputer sehingga mampu menyediakan informasi yang optimal sesuai yang dibutuhkan pemakai. (Nugroho, 2012)

2.10 Pengenalan Pemrograman Visual Basic

Microsoft Visual Basic merupakan bahasa pemrograman yang berbasis microsoft windows, sebagai bahasa pemrograman yang mutakhir, Microsoft Visual Basic 6.0 didesain untuk dapat memanfaatkan fasilitas yang tersedia dalam Microsoft windows. Microsoft Visual Basic 6.0 juga merupakan bahasa pemrograman Object Oriented Programing (OOP), yaitu pemrograman yang berorientasi objek. Visual Basic merupakan salah satu software untuk membuat program yang cukup sederhana tetapi banyak cakupan yang dapat dikerjakan,

karena visual basic dapat mengakses banyak software seperti Excel, Access dan sebagainya. Visual basic lebih sederhana dari pemrograman yang lain. Kesederhanaan visual basic terletak pada kemudahan membuat bahasa pemrograman dan bentuk tampilan yang dikehendaki. Visual Basic ini merupakan pengembangan bahasa basic yang diterapkan pada program yang berbasis Windows. (Nugroho, 2012)

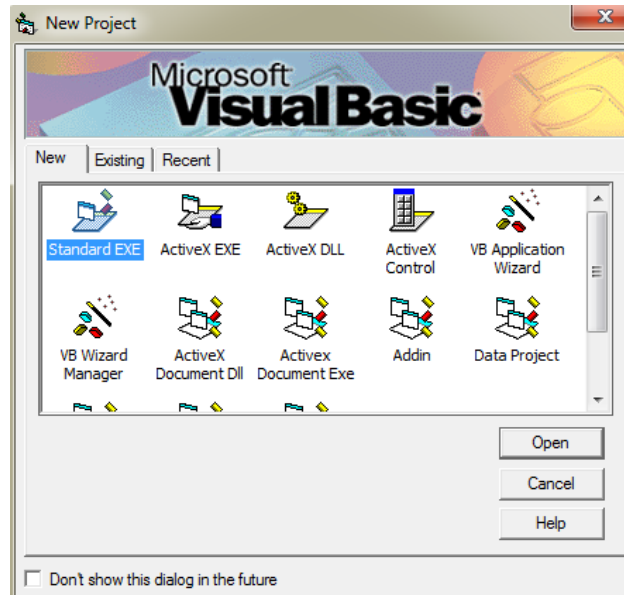
Visual Basic 6.0 adalah salah satu development tools untuk membangun aplikasi dalam lingkungan windows. Dalam pengembangan aplikasi, visual basic menggunakan pendekatan visual untuk merancang user interface atau tampilan dalam bentuk form, sedangkan untuk kodingnya menggunakan bahasa basic yang cenderung mudah dipelajari. Visual basic telah menjadi tools yang terkenal bagi para pemula maupun developer. Visual Basic 6.0 sebetulnya perkembangan dari versi sebelumnya dengan beberapa penambahan komponen yang sedang tren saat ini, seperti kemampuan pemrograman internet dengan DHTML (*Dynamic HyperText Mark Language*), dan beberapa penambahan fitur database dan multimedia yang semakin baik.

Secara umum ada beberapa manfaat yang diperoleh dari pemakaian program Microsoft Visual Basic, diantaranya:

- a. Dipakai dalam membuat program aplikasi berbasis Windows.
- b. Dipakai dalam membuat obyek-obyek pembantu program, seperti fasilitas Help, kontrol ActiveX, aplikasi Internet, dan sebagainya.
- c. Digunakan untuk menguji program (*Debugging*) dan menghasilkan program akhir EXE yang bersifat Executable, atau dapat langsung dijalankan.

2.10.1 Interface Visual Basic 6.0

Untuk memulai pemrograman dengan Visual Basic, jalankan program Microsoft Visual Basic 6.0. Selanjutnya pada tampilan awal akan ditampilkan kotak dialog New Project seperti pada gambar 2.19.



Gambar 2.19 Tampilan Kotak Dialog New Project

Pada kotak dialog tersebut terdapat 3 buah tab yang terdiri dari:

- a. *New* (menampilkan daftar pilihan untuk membuat *project* baru)
- b. *Existing* (untuk *browsing* dan membuka *project*)
- c. *Recent* (untuk membuka *project* yang sering digunakan).

Visual Basic 6.0 menyediakan 13 jenis *project* yang bisa dibuat. Ada beberapa *project* yang biasa digunakan oleh banyak pengguna *Visual Basic*, antara lain:

- a. *Standard EXE*: *Project* standar dalam Visual Basic dengan komponen-komponen standar. Jenis *project* ini sangat sederhana, tetapi memiliki keunggulan bahwa semua komponennya dapat diakui oleh semua unit komputer dan semua user meskipun bukan administrator. Pada buku ini akan digunakan *project Standard EXE* ini, sebagai konsep pemrograman visualnya.
- b. *ActiveX EXE*: *Project* ini adalah *project ActiveX* berisi komponen-komponen kemampuan untuk berinteraksi dengan semua aplikasi di sistem operasi *windows*.
- c. *ActiveX DLL*: *Project* ini menghasilkan sebuah aplikasi *library* yang selanjutnya dapat digunakan oleh semua aplikasi di sistem operasi *Windows*.
- d. *ActiveX Control*: *Project* ini menghasilkan komponen-komponen baru untuk aplikasi Visual Basic yang lain.

- e. *VB Application Wizard: Project* ini memandu pengguna untuk membuat aplikasi secara mudah tanpa harus pusing-pusing dengan perintah-perintah pemrograman.
- f. *Addin: Project* seperti *Standard EXE* tetapi dengan berbagai macam komponen tambahan yang memungkinkan kebebasan kreasi dari pengguna.
- g. *Data project: Project* ini melengkapi komponennya dengan komponen-komponen database. Sehingga bisa dikatakan *project* ini memang disediakan untuk keperluan pembuatan aplikasi database.
- h. *DHTML Application: Project* ini digunakan untuk membuat aplikasi internet pada sisi *client (client side)* dengan fungsi-fungsi DHTML.
- i. *IIS Application: Project* ini menghasilkan aplikasi internet pada sisi *server (server side)* dengan komponen-komponen CGI (*Common Gateway Interface*).

Untuk pembuatan program pertama kali pilih tab New, pilih Standard EXE lalu klik Open. Selanjutnya muncul tampilan utama Visual Basic 6.0 seperti pada gambar 2.20.

1. Title Bar

Title bar merupakan batang jendela dari program visual basic 6.0 yang terletak pada bagian paling atas dari jendela program yang berfungsi untuk menampilkan judul atau nama jendela. Selain itu juga berfungsi untuk memindahkan posisi jendela dengan menggunakan *drag* dan *drop* pada posisi *title bar* tersebut dan untuk mengatur ukuran jendela dari ukuran *minimize* ke ukuran *restore* ataupun sebaliknya dengan melakukan klik ganda pada posisi *title bar* tersebut.

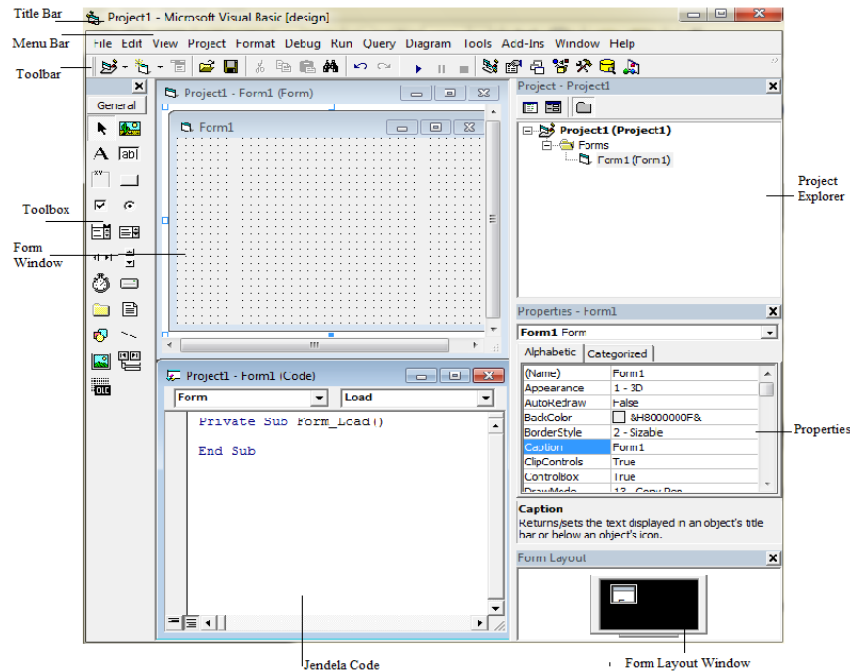
2. Menu Bar

Menu bar merupakan batang menu yang terletak di bawah *title bar* yang berfungsi untuk menampilkan pilihan menu atau perintah untuk mengoperasikan program visual basic.

3. Toolbars

Toolbars merupakan sebuah batang yang berisi kumpulan tombol yang terletak dibagian bawah menu bar atau terdapat didalam Menu Bar (*shortcut*) yang dapat digunakan untuk menjalankan perintah memanipulasi *Project*. Pada

kondisi *default* program visual basic hanya menampilkan *toolbars* standar. Namun dapat pula di-*set* sesuai dengan keinginan kita sendiri. Tampilan IDE Microsoft Visual Basic 6.0 seperti gambar 2.20.



Gambar 2.20 IDE Microsoft Visual Basic 6.0

4. *Project Explorer*

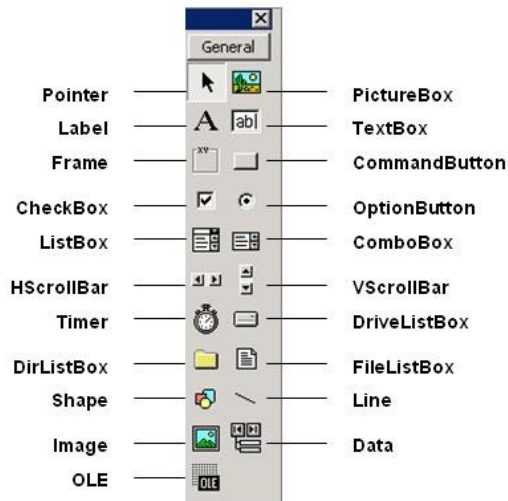
Project Explorer merupakan suatu kumpulan *module* atau merupakan program aplikasi itu sendiri. Dalam visual basic, *file project* disimpan dengan nama *file* berakhiran *vbp*, dimana *file* ini berfungsi untuk menyimpan seluruh komponen program. Apabila membuat suatu program aplikasi baru maka secara otomatis *project* tersebut akan diisi dengan *form* baru.

5. *Form*

Form merupakan *windows* atau jendela di mana akan dibuat *user interface*. Kita dapat menambahkan sebanyak mungkin *form* kedalam aplikasi kita sesuai dengan kebutuhan.

6. *Toolbox* atau kontrol

Merupakan tampilan berbasis grafis yang dimasukkan pada *form* untuk membuat interaksi dengan pemakai. Bentuk *toolbox* visual basic seperti gambar 2.21.



Gambar 2.21 Toolbox Pada Visual Basic 6.0

7. Properties

Properties merupakan nilai yang dimiliki oleh sebuah objek visual basic, merupakan sebuah jendela yang digunakan untuk menampung nama properti dari kontrol yang dipilih.

8. Jendela Code

Jendela Code adalah salah satu jendela yang paling penting dalam visual basic, yang berisi kode-kode program yang merupakan instruksi-instruksi untuk aplikasi visual basic. Setiap objek pada visual basic dapat ditambahkan kode-kode program untuk melaksanakan tugas-tugas tertentu, misalnya membatalkan perintah, menutup aplikasi dan sebagainya.

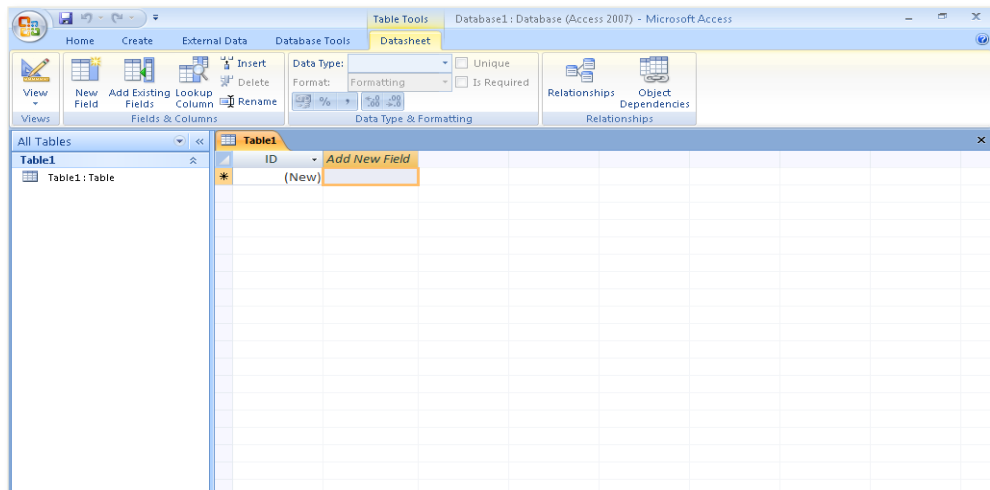
9. Form Layout Window

Form Layout Window merupakan sebuah jendela yang digunakan untuk mengatur posisi dari *form* pada *form* saat program dijalankan. Pada saat mengarahkan *pointer mouse* ke bagian *form*, maka *pointer mouse* akan berubah menjadi anak panah empat arah (*pointer* mengatur posisi) untuk memindah posisi *form* pada *layer monitor* dapat dilakukan dengan proses *drag* dan *drop*.

2.11 Microsoft Access

Microsoft Access atau *Microsoft Office Access* adalah sebuah program aplikasi basis data komputer relasional yang ditujukan untuk kalangan rumahan dan perusahaan kecil hingga menengah. Aplikasi ini merupakan anggota dari

beberapa aplikasi *Microsoft Office*. Aplikasi ini menggunakan mesin basis data *Microsoft Jet Database Engine* dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna. (Nugroho, 2012). Tampilan *Microsoft Access* seperti gambar 2.22.



Gambar 2.22 Tampilan *Microsoft Access*

Microsoft Access dapat menggunakan data yang disimpan di dalam format *Microsoft Access*, *Microsoft Jet Database Engine*, *Microsoft SQL Server*, *Oracle Database* atau semua kontainer basis data yang mendukung standar ODBC. Para pengguna/*programmer* yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para *programmer* yang kurang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang sederhana. *Access* juga mendukung teknik-teknik pemrograman berorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat bantu pemrograman berorientasi objek.

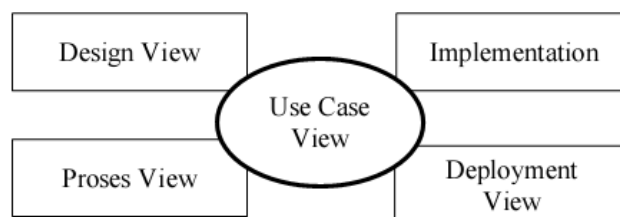
2.12 *Unified Modelling Language (UML)*

Unified Modelling Language (UML) adalah suatu alat yang memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual. Dan juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek (Haviluddin, 2011).

Berikut tujuan utama dalam desain UML adalah:

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. Karena merupakan bahasa pemodelan visual dalam proses pembangunannya maka UML bersifat independen terhadap bahasa pemograman tertentu.
4. Memberikan dasar formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar terhadap penggunaan alat desain sistem yang berorientasi objek (OO).
6. Memiliki integrasi praktik terbaik.

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*) dapat dilihat pada gambar 2.23.



Gambar 2.23 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, interaksi)

3. *Implementation View*

Menjelaskan komponen-komponen fisik 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. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.

5. Diagram Activity







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. Sebagai contoh, perhatikan proses yang terjadi pada pengambilan uang dari bank melalui ATM. Terdapat tiga aktifitas kelas (orang, dan lainnya) yang terkait yaitu: *Costumer*, ATM, dan Bank. Proses berawal dari lingkaran start hitam pada bagian atas dan berakhir di pusat lingkaran stop hitam/putih pada bagian bawah.

Simbol-simbol UML terdiri dari simbol *Use Case Diagram*, *Class Diagram*, *Sequence Diagram*, *Chart Diagram* dan *Activity Diagram* dapat dilihat pada gambar 2.1, 2.2, 2.3, 2.4 dan 2.5.


Tabel 2.1.a Simbol *Use Case Diagram*

No	Simbol	Nama	Keterangan
1		Actor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (<i>independent</i>).
3		Generalization	Hubungan dimana objek anak (<i>descendent</i>) berbagai perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
4		Include	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> .







Tabel 2.1.b Simbol *Use Case Diagram*

5		<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

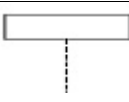

Tabel 2.2.a Simbol *Class Diagram*

No	Simbol	Nama	Keterangan
1.		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagai perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).







Tabel 2.2.b Simbol *Class Diagram*

2.		<i>Nary Association</i>	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		<i>Class</i>	Himpunan dari objek-objek yang berbagai atribut serta operasi yang sama.
4.		<i>Collaboration</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5.		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7.		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.






Tabel 2.3 Simbol *Sequence Diagram*

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

Tabel 2.4 Simbol *Chart Diagram*

No	Simbol	Nama	Keterangan
1.		<i>State</i>	Nilai atribut dan nilai link pada suatu waktu tertentu, yang dimiliki oleh suatu objek.
2.		<i>Initial Pseudo State</i>	Bagaimana objek dibentuk atau diawali.
3.		<i>Final State</i>	Bagaimana objek dibentuk dan dihancurkan
4.		<i>Transition</i>	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya.
5.		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
6.		<i>Node</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2.5. Simbol *Activity Diagram*

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