BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Pada penelitian sebelumnya yang di lakukan oleh Andri Saputra, Dwi Febriansyah, dan Haris Kuswara tahun 2014 dalam jurnal yang berjudul "Alat Kendali Lampu Rumah Menggunakan *Bluetooth* Berbasis *Android"* pada penelitian ini *Andorid* di gunakan sebagai *remote control* untuk mengendalikan lampu di mana Arduino sebagai sistem minimum dan *Bluetooth* sebagai media transmisi.

Penelitian ini bertujuan untuk memudahkan kerja manusia dalam menyalakan dan memadamkan lampu rumah tanpa harus berinteraksi langsung dengan saklar listrik.

Cara kerja alat pada penelitian ini pada saat *bluetooth* diaktifkan dari *smartphone* android maka *smartphone* android akan mencari perangkat modul *bluetooth* yang terpasang pada *mikrokontroller.* Maka akan tampil perangkat *bluetooth* yang sudah tersedia kemudian dihubungkan pada modul *bluetooth* yang ada pada *mikrokontroller* dengan memasukan *password* terlebih dahulu. Setelah perangkat sudah terhubung maka untuk menyalakan dan memadampakan lampu rumah dapat dikendalikan melalui *smartphone* Android yang sudah terpasang aplikasi.

Dari penelitian yang dilakukan, dapat di ambil kesimpulan bahwa untuk menyalakan dan memdamkan lampu rumah harus menggunakan *smartphone Android* sebagai *Remote Control* dengan media transmisi *bluetooh*.

Kekurangan alat ini jika di kembangkan untuk kamar pasien di rumah sakit adalah alat ini sangat tidak efektif dan efisien karena jika pasien yang tidak bisa menggerakan tangannya, pasien masih tidak bisa mengendalikan lampu karena untuk mengendalikan lampu pasien harus memegang *smartphone Android* dan di rumah sakit sendiri terdiri dari berbagai kalangan pasien dan ada kemungkinan pasien tidak mengenal *smartphone Android* sehingga pasien tersebut tidak bisa mengendalikan lampu.

2.1.1 Perbedaan dengan penelitian sebelumnya

Dalam laporan akhir yang berjudul "Rancang Bangun Alat Menyalakan dan Memadamkan Lampu Menggunakan Suara". Pada rancang bangun alat ini untuk mengendalikan lampu cukup mengguanakan suara tanpa *remote control* atau saklar listik. Alat ini menggunakan *voice recognition modul* sebagai modul pengelola suara dan arduino sebagai sistem minimum. Pada saat kita memasukan suara ke *microphone* maka suara tersebut akan di proses oleh *voice recognition modul* kemudian akan di teruskan ke arduino. Arduino memberikan aksi untuk menggerakan relay untuk menghasilkan aksi berupa keluaran nyala atau padam lampu tersebut. Apabila suara yang di masukkan adalah "nyala" maka lampu akan menyala dan apabila suara yang di masukkan adalah "padam" maka lampu akan padam.

Pada penelitian ini, biaya tidak terlalu mahal dan cukup efisien dikarenakan tidak membutuhkan *smartphone* Android untuk mengendalikan lampu. Jika seseorang tidak memiliki *smartphone* Android, orang tersebut masih bisa menyalakan dan memadamkan lampu dikarenkan cukup menggunakan suara.

2.2 Sistem Pengenal Ucapan (*Speech Recognition System)*

Sistem pengenal suara atau *speech recognition system* merupakan sebuah system yang menggunakan mikrofon sebagai alat input kemudian mengubah sinyal inputan analog menjadi sinyal digital dan membandingkan pola-pola listrik yang dihasilkan oleh suara atau ucapan (suara input) dengan sekumpulan data pola rekaman terlatih yang tersimpan di komputer. System ini dapat mengerjakan dua tugas, yaitu mengubah pola suara ke tulisan dan mengeluarkan perintah suara untuk mengontrol computer. System pengenalan suara ini harus mampu mengatasi beragam kesulitan, misalnya perbedaan suara, pelafalan dan aksen. (Jeffry,dkk.2014)

Pengenalan suara sendiri terbagi menjadi dua kategori, yaitu, Piranti pengenalan kata dan Piranti pengenalan kalimat.

Piranti pengenalan kata (*word recognation*) yang mampu merespon ucapan-ucapan secara individu atau perintah-perintah yang menggunakan teknik

yang dikenal sebagai *speaker verification*. Disini sistem akan membangkitkan suatu template untuk mengenali suara *user*. Piranti pengenalan kalimat (*speech recognition*) yang mampu mengenali hubungan antar kata terucap di dalam kalimat atau frase. Teknik-teknik statistik dipakai dalam pola perekaman suara yang akan dicocokan dengan kata-kata terucap.

Ada 2 tipe *Speech Recognition* dilhat dari ketergantungan pembicara yaitu, *Independent Speech Recognition* dan *Dependent Speech Recognition.*

Independent Speech Recognition,yaitu sistem pengenal ucapan tanpa terpengaruh dengan siapa yang berbicara, tetapi mempunyai keterbatasan dalam jumlah kosakata. Model ini akan mencocokan setiap ucapan dengan kata yang dikenali dan memilih yang sangat mendekati kata yang cocok.Untuk mendapatkan kecocokan kata yang diucapkan maka digunakan model statistik yang dikenal dengan nama *Hidden Markov Model* (HMM).

Dependent Speech Recognition, yaitu sistem pengenal ucapan yang memerlukan pelatihan khusus dari pembicara, dimana hasil pelatihan dari masing-masing pembicara akan disimpan dalam sebuah profil. Profil inilah yang nantinya digunakan untuk berinteraksi dengan sistem pengenal ucapan dan sistem akan bergantung dengan siapa yang berbicara. Sistem ini biasanya lebih mudah untuk dikembangkan, dimana contoh suara sudah dibuat sebelumnya dan disimpan dalam database (basis data) dan jumlah kosakatanya lebih besar dibandingkan dengan independent *speech recognition*. Proses pengenalan ucapan dengan cara membandingkan ucapan pembicara dengan contoh suara yang ada.

Saat ini sistem pengenal ucapan telah menggunakan model statistik untuk menentukan dengan tepat kata yang dimaksud. Menurut John Garofolo, *Speech Group Manager* pada *Information Technology Laboratory* di *National Institute of Standards* and *Technologi*, model statistik yang sangat mendominasi adalah *Hidden Markov Model* dan *Neutral Networks*. Tetapi yang paling sering digunakan adalah *Hidden Markov Model*. Pada gambar 1 ditunjukan dimana setiap fonem dihubungkan seperti rantai, dan rangkaian rantai yang komplit merupakan kata.

Sumber sinyal yang berasal dari ucapan seseorang dinyatakan sebagai *Speech Waveform.* Sumber sinyal ini kemudian dicari ciri pentingnya yang dapat membedakan antara satu sinyal dengan sinyal lain. Pengungkapan ciri dilakukan dengan pengolahan sinyal. Hasil dari pengolahan dan analisis sinyal adalah deretan vektor ciri ucapan, dinyatakan sebagai *spectral feature vector.* Vektor-vektor ini dilatih menggunakan jaringan syaraf tiruan. Pelatihan dimaksudkan agar pemrosesan vektor dapat dilakukan dengan cepat. Hasil dari pelatihan berupa deretan fonem-fonem yang membentuk data ucapan. Pemodelan fonem ini menggunakan model akustik *Hidden Markov Model* (HMM). Proses pencarian kesamaan data fonem dilakukan dengan menggunakan *N-gram Grammar* dan di kodekan kembali menggunakan algoritma *Viterbi.* Proses ini melihat nilai total penjajaran terbaik. Dari penjajaran ini diperoleh deretan kata yang terpilih dan diasumsikan yang paling benar.

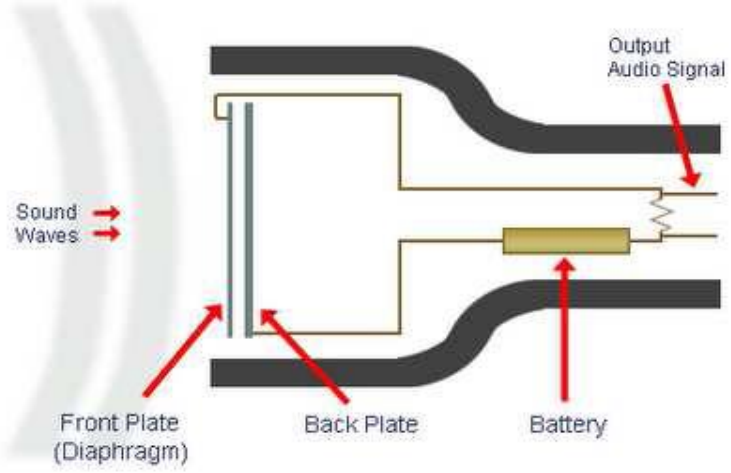
2.3 Sensor Suara

Sensor suara adalah sebuah alat yang mampu mengubah gelombang *Sinusioda* suara menjadi gelombang sinus energi listrik (*Alternating Sinusioda Electric Current).* Sensor suara bekerja berdasarkan besar/kecilnya kekuatan gelombang suara yang mengenai membran sensor yang menyebabkan bergeraknya membran sensor yang juga terdapat sebuah kumparan kecil di balik membran tadi naik dan turun. Oleh karena kumparan tersebut sebenarnya adalah ibarat sebuah pisau yang berlubang-lubang maka pada saat ia bergerak naik-turun, ia juga telah membuat gelombang magnet yang mengalir melewatinya terpotong-potong. Kecepatan gerak kumparan menentukan kuat-lemahnya gelombang listrik yang dihasilkannya. (Jeffry,dkk.2014)

2.4 *Microphone*

Microphone adalah komponen elektronika dimana cara kerjanya yaitu membran yang digetarkan oleh gelombang suara akan menghasilkan sinyal listrik dan lain-lain. Secara umum ada dua jenis *microphone* yaitu *Microphone Condenser* dan *dynamic microphone*. (Jeffry,dkk.2014)

2.4.1 *Microphone Condenser*



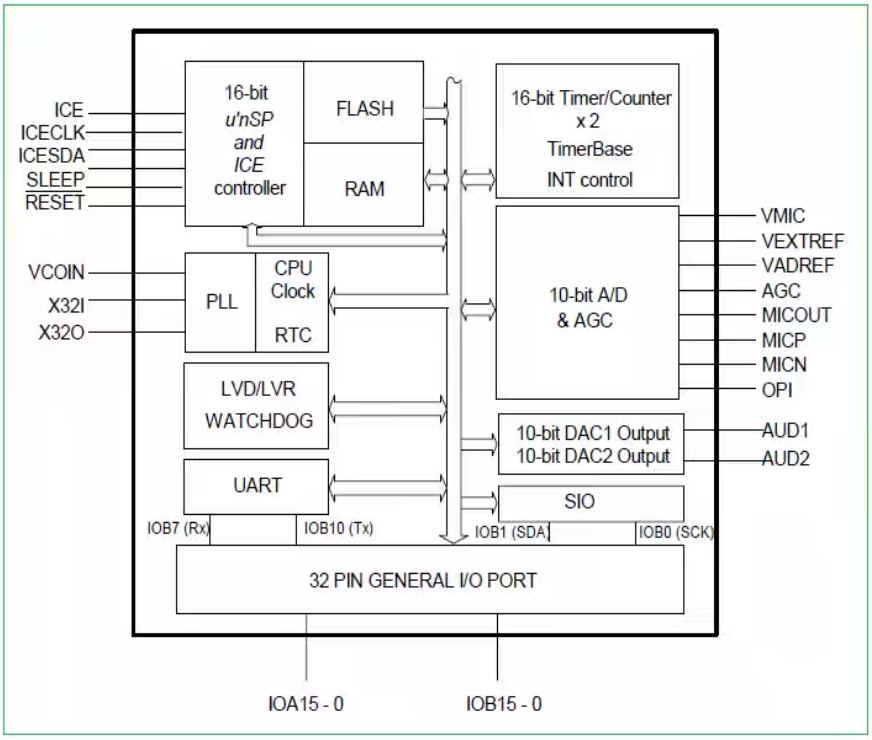
Gambar 2.1 Cara kerja *microphone condenser*
Sumber : Jeffry,dkk.2014

Microphone tipe ini tersusun atas 2 keping plat tipis yang berfungsi untuk menangkap gelombang suara. Cara kerjanya sederhana, gelombang suara yang masuk akan menggetarkan kedua plat ini sehingga membentuk sinyal-sinyal audio yang kemudian diteruskan ke pre-amplifie untuk dikuatkan Karena hanya menggunakan 2 plat yang bisa disesuaikan ukuranya, maka *mic condenserini memiliki ukuran yang kecil* dan ringan. Karena kecil, *mic condenser* banyak digunakan dalam berbagai perangkat elektronik seperti ponsel, handsfree, *headphone*, dan lain-lain. Mic tipe ini harus menggunakan daya dalam pengoperasiannya. Jika tidak maka *mic condenser* tidak akan bekerja. Daya yang digunakan sedikit saja sehingga hemat . (Jeffry,dkk.2014)

2.4.2 *Dynamic Microphone*

Microphone ini tersusun dari gulungan spul (coil) yang mengelilingi sebuah magnet silinder. Penyusun mic tipe ini sama seperti spul loadspeaker, hanya saja, jika *loudspeaker* spulnya sedikit (impedansi 4-16 ohm) maka spulpada mic ini lebih banyak dan lebih panjang gulungannya (impedansi sekitar 600 ohm). (Jeffry,dkk.2014)

2.5 *Voice Recognition Modul*



Gambar 2.2 Diagram Blok *Voice Recognize Modul*

Sumber : www.theorycircuit.com

Modul ini bisa menyimpan 15 buah instruksi suara. Ke 15 buah tersebut dibagi menjadi 3 kelompok, dengan 5 di satu kelompok. Pertama kita harus merekam instruksi suara kelompok demi kelompok. Setelah itu, kita harus mengimpor satu grup dengan perintah serial sebelum bisa mengenali 5 instruksi suara dalam kelompok tersebut. Jika kita perlu menerapkan instruksi pada kelompok lain, kita harus mengimpor grup terlebih dahulu. Modul ini adalah pembicara independen. Jika teman Anda mengucapkan instruksi suara alih-alih Anda, mungkin tidak mengenali instruksinya.

Berikut Penjelasan dari *Voice Recognize Modul* :

1. Parameter

a. Tegangan: 4.5-5.5V

b. Saat ini: <40mA

c. Antarmuka Digital: antarmuka UART tingkat 5V TTL

d. Analog Interface: konektor mikrofon mono-channel 3.5mm + pin pin mikrofon

e. Ukuran: 30mm x 47.5mm

f. Akurasi pengenal: 99% (di bawah lingkungan ideal)

2. *Serial Command*

Modul ini dapat dikonfigurasi dengan mengirimkan perintah melalui port serial. Konfigurasi tidak akan terhapus setelah dimatikan. Antarmukanya adalah 5V TTL. Format data serial: 8 bit data, tidak ada paritas, 1 stop bit. Tingkat baud default adalah 9600 dan baud rate dapat berubah.

Format perintah adalah "Head + Key". "Head" adalah 0xaa, dan "Key" adalah sebagai berikut:

Tabel 2.1 Format Perintah *Serial Command Key* pada *Voice Recognition Modul*

Key (HEX format)	Description	Respond in Common Mode	Respond in Compact Mode
0x00	Enter into "Waiting" state	"Waiting! | n" : successful "ERROR! | n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x01	Delete the instructions of group 1	"Group1 Deleted ! | n" : successful "ERROR! | n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x02	Delete the instructions of group 2	"Group2 Deleted ! | n" : successful "ERROR! | n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x03	Delete the instructions of group 3	"Group3 Deleted ! | n" : successful "ERROR! | n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x04	Delete the instructions of all the 3 groups	" All Groups Deleted !\n " : successful "ERROR! | n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x11	Begin to record instructions of group 1	"ERROR! | n" : Instruction error "START | n" : Ready for recording, you can speak now "No voice | n" : no voice detected "Again | n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud | n" : Too loud to record "Different | n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one | n" : recording one voice instruction successfully "Group1 finished! | n" : finish recording group 1	0xe0 : Instruction error 0x40 : Ready for recording, you can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message 0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x46 : finish recording group 1
0x12	Begin to record instructions of group 2	"ERROR! | n" : Instruction error "START | n" : Ready for recording, you can speak now "No voice | n" : no voice detected "Again | n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud | n" : Too loud to record "Different | n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one | n" : recording one voice instruction successfully "Group2 finished! | n" : finish recording group 2	0xe0 : Instruction error 0x40 : Ready for recording, you can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message 0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x47 : finish recording group 2
0x13	Begin to record instructions of group 3	"ERROR! | n" : Instruction error "START | n" : Ready for recording, you can speak now "No voice | n" : no voice detected "Again | n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud | n" : Too loud to record	0xe0 : Instruction error 0x40 : Ready for recording, you can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message

		"Different | n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one | n" : recording one voice instruction successfully "Group3 finished! | n" : finish recording group 3	0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x48 : finish recording group 3
0x21	Import group 1 and be ready for voice instruction	"Group1 Imported !\n" : Successful "ERROR! | n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed
0x22	Import group 2 and be ready for voice instruction	"Group2 Imported !\n" : Successful "ERROR! | n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed
0x23	Import group 3 and be ready for voice instruction	"Group3 Imported !\n" : Successful "ERROR! | n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed
0x24	Query the recorded group	"Used group:0\n" : No group is recorded "Used group:1\n" : Group 1 is recorded "Used group:2\n" : Group 2 is recorded "Used group:3\n" : Group 3 is recorded "Used group:12\n" : Group 1 and Group 2 are recorded "Used group:13\n" : Group 1 and Group 3 are recorded "Used group:23\n" : Group 2 and Group 3 are recorded "Used group:123\n" : All the 3 groups are recorded "ERROR! | n" : Instruction error	0x00 : No group is recorded 0x01 : Group 1 is recorded 0x02 : Group 2 is recorded 0x04 : Group 3 is recorded 0x03 : Group 1 and Group 2 are recorded 0x05 : Group 1 and Group 3 are recorded 0x06 : Group 2 and Group 3 are recorded 0x07 : All the 3 groups are recorded 0xe0 : Instruction error
0x31	Change the baud rate to 2400bps	"Baud: 2400\n" : Successful "ERROR! | n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x32	Change the baud rate to 4800bps	"Baud: 4800\n" : Successful "ERROR! | n" : Instruction error	
0x33	Change the baud rate to 9600bps	"Baud: 9600\n" : Successful "ERROR! | n" : Instruction error	
0x34	Change the baud rate to 19200bps	"Baud: 19200\n" : Successful "ERROR! | n" : Instruction error	
0x35	Change the baud rate to 38400bps	"Baud: 38400\n" : Successful "ERROR! | n" : Instruction error	
0x36	Switch to Common Mode	"Common Mode\n" : Successful "ERROR! | n" : Instruction error	
0x37	Switch to Compact Mode	"Compact Mode\n" : Successful "ERROR! | n" : Instruction error	
0xbb	Query version information	Version information	No respond

Jika Anda ingin mengubah baud rate serial menjadi 38400, Anda perlu mengirim perintah: 0xaa35. Jika berhasil, itu akan kembali "Baud: 38400\N "(dalam Common Mode) atau 0xcc (dalam mode Compact). Tingkat baud diatur ke 38400.

Perbedaan utama antara Compact Mode dan Common Mode adalah pesan yang kembali. *Respon Common Mode* adalah string panjang sedangkan *Response Compact Mode* adalah byte. Misalnya, setelah mengirim 0xaa04 untuk menghapus semua isi dari 3 grup, di *Common Mode* itu akan kembali "Semua

Grup Dihapus! \ N", namun dalam *Compact Mode* ia akan mengembalikan byte ringkas seperti 0xcc yang berarti Operasi yang sukses.

Untuk penggunaan pertama kali, kita perlu melakukan beberapa konfigurasi:

1. Pilih baud rate serial (standar 9600)
2. Pilih mode komunikasi: Common Mode atau Compact Mode
3. Merekam lima instruksi dari kelompok pertama (atau yang kedua atau ketiga sesuai kebutuhan)
4. Impor grup yang perlu Anda gunakan (hanya mengenali 5 instruksi dalam satu grup sekaligus)

Setelah semua pengaturan di atas, Anda dapat berbicara atau mengirim instruksi suara untuk itu. Jika berhasil diidentifikasi, hasilnya akan dikembalikan melalui port serial dalam format: nomor grup + nomor perintah. Sebagai contoh, return Result: 11 (Compact mode returns 0x11) berarti mengidentifikasi perintah pertama kelompok 1. Jika instruksi suara dicatat, setiap kali Anda menyalakannya, Anda harus mengimpor grup sebelum membiarkannya mengenali suara Instruksi.

3. LED

a. Tahap rekaman

1. Catatan rekaman: D1 (RED) berkedip 3 kali dalam 600ms, lalu mati for400ms, lalu berkedip cepat selama 4 Kali dalam 600ms Sekarang indikasi rekaman selesai.
2. Mulai bicara: D1 (RED) mati selama 400ms, dan kemudian menyala. Suara selama waktu sementara D1 (RED) menyala direkam oleh modul ini.
3. Merekam instruksi suara dengan sukses untuk pertama kalinya: D1 (RED) mati, D2 (ORANGE) menyala 300ms.
4. Merekam instruksi suara dengan sukses untuk pertama kalinya: D1 (RED) mati, D2 (ORANGE) menyala seharga 700ms.
5. Rekaman kegagalan: D2 (ORANGE) berkedip 4 kali dalam 600ms. Jika instruksi suara terdeteksi dua kali tidak cocok, atau suaranya terlalu besar,

atau tidak ada suara, rekaman akan gagal. Anda perlu memulai dari rekaman proses untuk instruksi itu.

b. Mode menunggu :

Dalam mode tunggu, D2 (ORANGE) dimatikan, dan D1 (RED) menyala 80ms setiap 200ms lainnya, berkedip cepat. Dalam mode ini, ia tidak mengenali Perintah suara, hanya menunggu perintah serial.

c. Tahap identifikasi :

Pada tahap identifikasi, D2 (ORANGE) dimatikan, dan D1 (RED) menyala selama 100ms setiap 1500ms lainnya, berkedip lambat. Pada tahap ini, modul ini sedang memproses sinyal suara yang diterima, dan jika cocok, maka akan mengirimkan hasilnya segera melalui port serial.

4. *Recording*

Sebelum menggunakannya, kita harus merekam instruksi suara. Setiap instruksi suara memiliki panjang maksimum 1300ms, yang memastikan itu kebanyakan kata bisa direkam. Begitu Anda mulai merekam, Anda tidak bisa menghentikan proses perekaman sampai Anda menyelesaikan semua 5 suara Instruksi rekaman satu kelompok. Juga, setelah Anda mulai merekam, konten sebelumnya dari grup itu akan dihapus. Dalam rekaman tahap, modul ini tidak membalas perintah serial lainnya.

2.6 *Arduino*

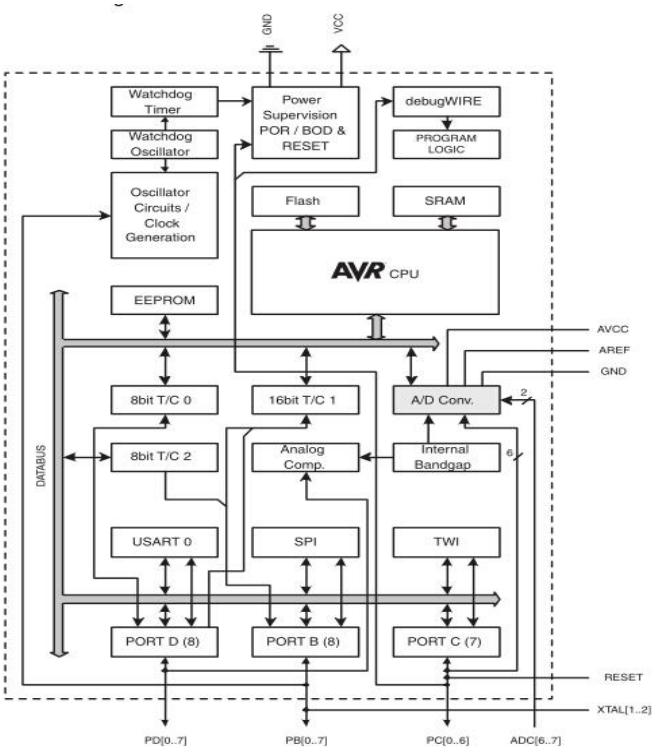
Arduino adalah sebuah platform komputasi fisik *open source* berbasiskan Rangkain *input / output* sederhana (I/O) dan lingkungan pengembangan yang mengimplementasikan bahasa *Processing*. Arduino dapat digunakan untuk mengembangkan obyek interaktif mandiri atau dapat dihubungkan ke perangkat lunak pada komputer anda (seperti Flash, Pengolahan, VVVV, atau Max / MSP). Rangkaiannya dapat dirakit dengan tangan atau dibeli. IDE *(Integrated Development Environment)* Arduino bersifat *open source*. (Steven,dkk.2016)

Beberapa Contoh Arduino diantaranya : Arduino Duemilanove, Arduino UNO, Arduino Leonardo, Arduino Mega, Arduino MEGA 2560 R3, Arduino Nano, Arduino Due, LilyPad Arduino, dan Arduino mikro.

2.6.1 *Arduino Uno R3*

Pada perancangan dan pembuatan tugas akhir ini digunakan jenis papan *Arduino Uno R3*. Arduino Uno adalah sebuah *board* mikrokontroler yang didasarkan pada ATmega328. Dibawah ini adalah Spesifikasi *Arduino Uno R3*:

1. Mikrokontroler : ATmega328
2. Tegangan pengoperasian : 5V Tegangan input yang disarankan: 7-12V
3. Batas tegangan input : 6-20V
4. Jumlah pin I/O digital : 14
5. Jumlah pin input analog : 6
6. Arus DC tiap pin I/O : 40 mA
7. Arus DC untuk pin 3.3V : 50 mA
8. Memori :32 KB (ATmega328), sekitar 0.5 KB digunakan oleh *bootloader*.
9. SRAM : 2 KB (ATmega328)
10. EEPROM : 1 KB (ATmega328)
11. Clock Speed : 16 MHz



Gambar 2.3 Blok Diagram *Arduino Uno R3*

Sumber : indonesian.alibaba.com

2.6.2 Karakteristik *Arduino Uno R3*

Arduino uno memiliki Spesifikasi, adapun spesifikasi dari *Aduino Uno* adalah sebagai berikut :

1. Daya (Power)

Arduino UNO dapat disuplai melalui koneksi USB atau dengan sebuah *power* suplai eksternal. Sumber daya dipilih secara otomatis. Suplai eksternal (non-USB) dapat diperoleh dari sebuah adaptor AC ke DC atau *battery*. Adaptor dapat dihubungkan dengan mencolokkan sebuah *center-positive plug* yang panjangnya 2,1 mm ke *power jack* dari *board*. Kabel lead dari sebuah *battery* dapat dimasukkan dalam *header*/kepala pin *Ground* (Gnd) dan pin Vin dari konektor POWER. *Board Arduino UNO* dapat beroperasi pada sebuah suplai eksternal 6 sampai 20 Volt. Jika disuplai lebih kecil dari 7 V, misalnya pin 5 Volt mensuplai lebih kecil dari 5 Volt dan board *Arduino UNO* bisa menjadi tidak stabil. Jika menggunakan suplai yang lebih besar dari 12 Volt, voltage regulator bisa kelebihan panas dan membahayakan *board Arduino UNO*. Range yang direkomendasikan adalah 7 sampai 12 Volt. (Steven,dkk.2016)

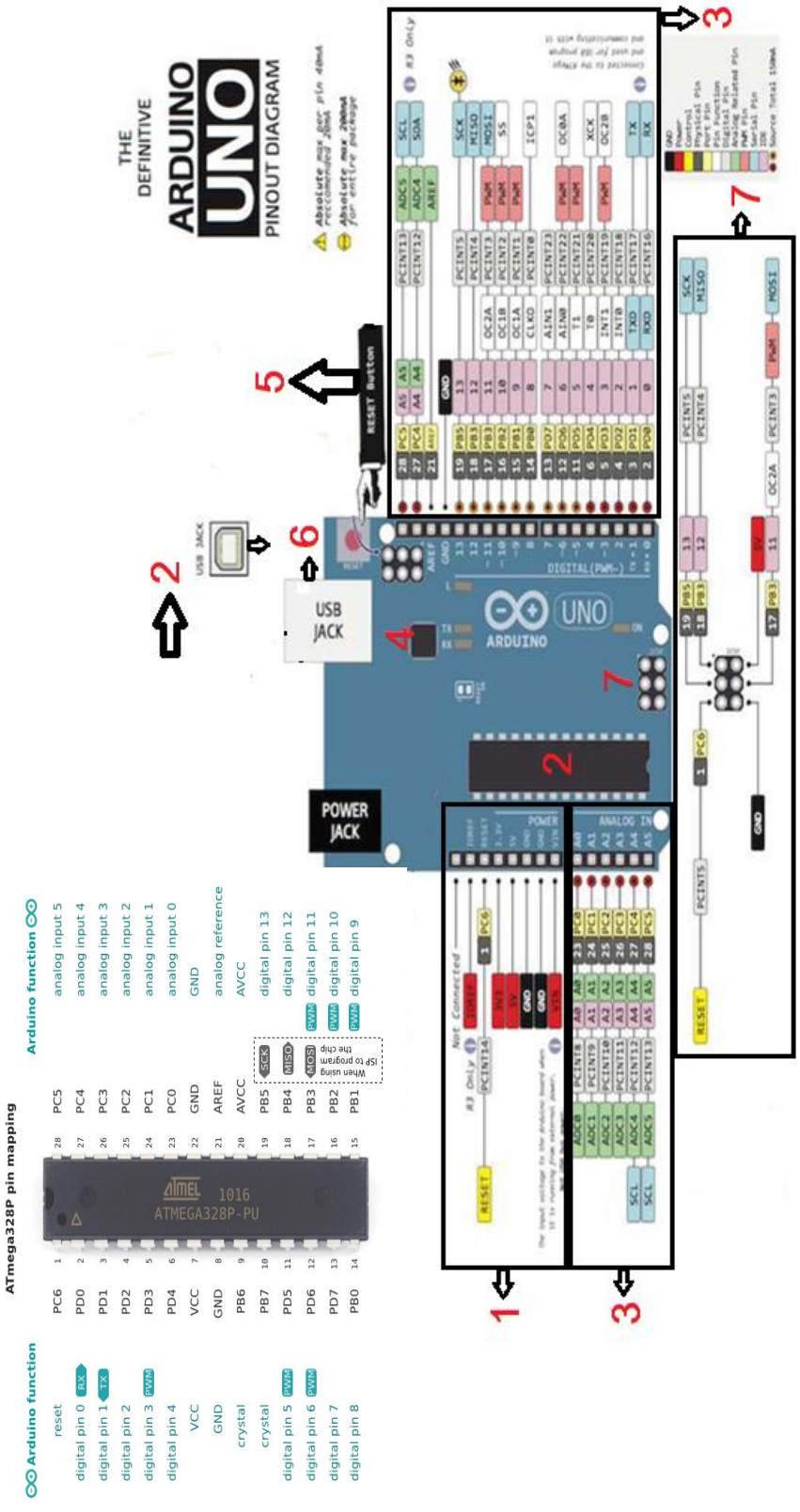
Penjelasan Power PIN:

a. **VIN -** Input voltase board saat anda menggunakan sumber catu daya luar (adaptor USB 5 Volt atau adaptor yang lainnya 7-12 volt), Anda bisa menghubungkannya dengan pin VIN ini atau langsung ke jack power 5V. DC power jack (7-12V), Kabel konektor USB (5V) atau catu daya lainnya (7-12V). Menghubungkan secara langsung power supply luar (7-12V) ke pin 5V atau pin 3.3V dapat merusak rangkaian Arduino ini, jangan salahkan saya ya?!

b. **3V3 -** Pin tegangan 3.3 volt catu daya umum langsung ke board. Maksimal arus yang diperbolehkan adalah 50 mA.

c. **GND -** Pin Ground.

d. **IOREF -** Pin ini penyedia referensi tengangan agar mikrokontrol beroperasi dengan baik. Memilih sumber daya yang tepat atau mengaktifkan penerjemah tegangan pada output untuk bekerja dengan 5V atau 3.3V.



Gambar 2.6 pin *out* diagram pada Arduino UNO R3 sumber : steven, dkk.2016

2. ICAtmega328

Atmega328 merupakan tipe AVR yang telah dilengkapi dengan 8 saluran ADC internal dengan fidelitas 10 bit.Dalam mode operasinya, ADC ATmega328 dapat dikonfigurasi, baik *secara single ended input* maupun *differential input.* Selain itu, ADC ATmega328 memiliki konfigurasi pewaktuan, tegangan referensi, mode operasi, dan kemampuan filter derau yang amat fleksibel, sehingga dengan mudah disesuaikan dengan kebutuhan ADC itu sendiri. (Ferry,dkk.2014)

ATmega328 memiliki 3 modul timer yang terdiri dari dua buah timer/counter 8 bit dan 1 buah timer/counter 16 bit. Ketiga modul timer/counter ini dapat diatur dalam mode yang berbeda secara individu dan tidak saling mempengaruhi satu sama lain.Selain itu, semua timer/counter juga dapat difungsikan sebagai sumber interupsi. Masing-masing timer/counter ini memiliki register tertentu yang digunakan untuk mengatur mode dan cara kerjanya.

Mikrokontroller ATmega328 memiliki arsitektur Harvard, yaitu memisahkan memori untuk kode program dan memori untuk data sehingga dapat memaksimalkan kerja dan *parallelism.*

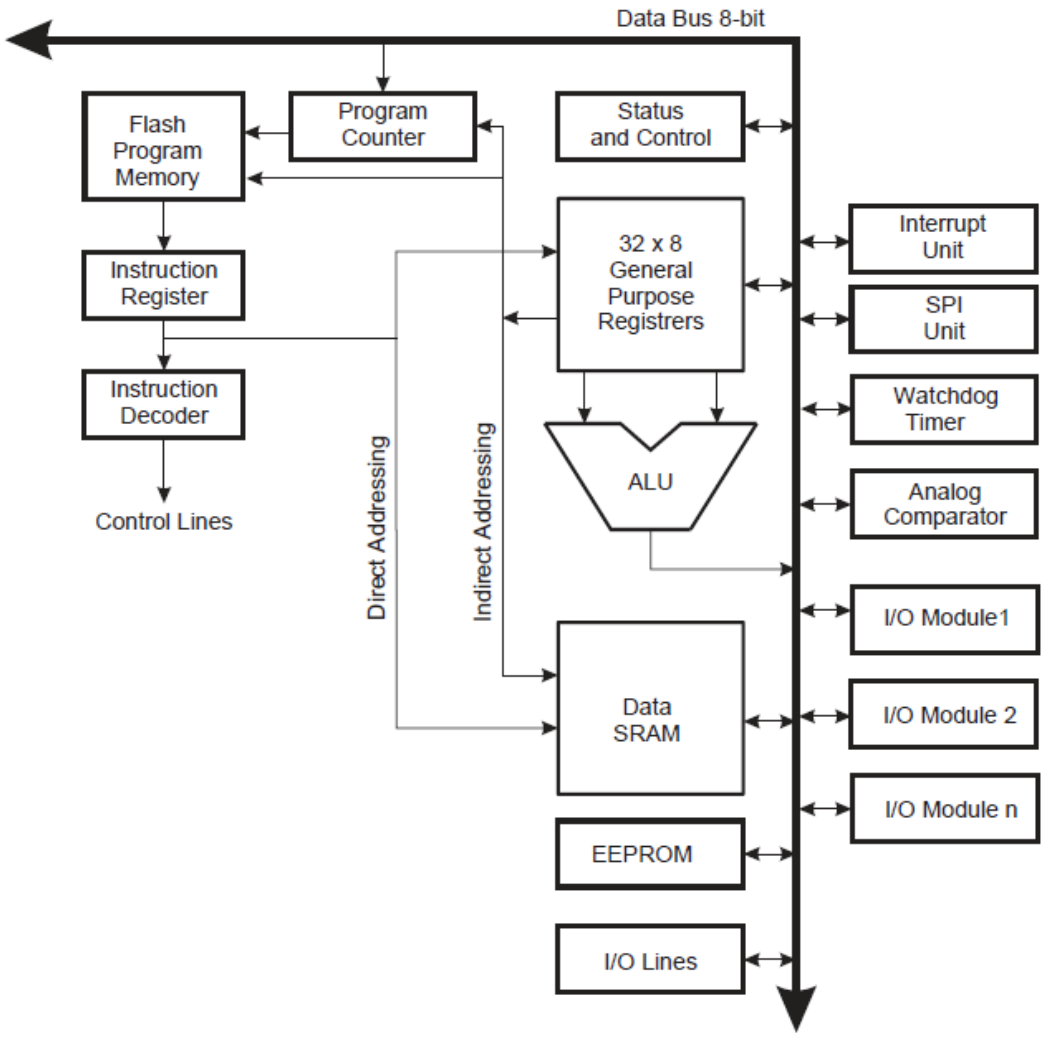
Instruksi – instruksi dalam memori program dieksekusi dalam satu alur tunggal, dimana pada saat satu instruksi dikerjakan instruksi berikutnya sudah diambil dari memori program. Konsep inilah yang memungkinkan instruksi – instruksi dapat dieksekusi dalam setiap satu siklus clock. 32 x 8-bit register serbaguna digunakan untuk mendukung operasi pada ALU (*Arithmatic Logic unit)* yang dapat dilakukan dalam satu siklus. 6 dari register serbaguna ini dapat digunakan sebagai 3 buah register pointer 16-bit pada mode pengalamatan tidak langsung untuk mengambil data pada ruang memori data.

Ketiga register pointer 16-bit ini disebut dengan register X (gabungan R26 dan R27), register Y (gabungan R28 dan R29), dan register Z (gabungan R30 dan R31). Hampir semua instruksi AVR memiliki format 16-bit. Setiap alamat memori program terdiri dari instruksi 16-bit atau 32-bit.

Selain register serba guna di atas, terdapat register lain yang terpetakan dengan teknik *memory mapped* I/O selebar 64 byte. Beberapa register ini digunakan untuk fungsi khusus antara lain sebagai *register control Timer /*

Counter, Interupsi, ADC, USART, SPI, EEPROM, dan fungsi I/O lainnya. Register – register ini menempati memori pada alamat 0x20h – 0x5Fh.

Arsitektur ATmega328 dapat dilihat pada gambar dibawah ini :



Gambar 2.5 Arsitektur ATmega328 Sumber: Ferry,dkk.2014

3. Input dan Output

Setiap 14 pin digital pada *Arduino Uno R3* dapat digunakan sebagai *input* dan *output*, menggunakan fungsi pin *Mode*(), digital *Write*(), dan digital *Read*(). Fungsi-fungsi tersebut beroperasi di tegangan 5 Volt. Setiap pin dapat memberikan atau menerima suatu arus maksimum 40 mA dan mempunyai sebuah resistor pull-up (terputus secara *default*) 20-50 kΩ. (Steven,dkk.2016)

beberapa pin memiliki fungsi kekhususan antara lain:

a. Serial: 0 (RX) dan 1 (TX). Sebagai penerima (RX) dan pemancar (TX) TTL serial data. Pin ini terkoneksi untuk pin korespondensi chip ATmega8U2 USB-toTTL Serial.

b. External Interrupts: 2 dan 3. Pin ini berfungsi sebagai konfigurasi trigger saat interupsi value low, naik, dan tepi, atau nilai value yang berubah-ubah.

c. PWM: 3, 5, 6, 9, 10, dan 11. Melayani output 8-bit PWM dengan fungsi analogWrite().

d. SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Pin yang support komunikasi SPI menggunakan SPI library.

e. LED: 13. Terdapat LED indikator bawaan (built-in) dihubungkan ke digital pin 13, ketika nilai value HIGH led akan ON, saat value LOW led akan OFF.

f. **TWI:** pin A4 atau pin SDA dan and A5 atau pin SCL. Support TWI communication menggunakan Wire library.

g. Uno memiliki 6 analog input tertulis di label A0 hingga A5, masing-masingnya memberikan 10 bit resolusi (1024). Secara asal input analog tersebut terukuru dari 0 (ground) sampai 5 volt, itupun memungkinkan perubahan teratas dari jarak yang digunakan oleh pin AREF dengan fungsi analogReference().

pin lainnya di board UNO:

a. **AREF.** Tegangan referensi untuk input analog. digunakan fungsi analogReference().

b. **Reset**. Meneka jalur LOW untuk mereset mikrokontroler, terdapat tambahan tombol reset untuk melindungi salah satu blok.

4. Komunikasi

Arduino UNO mempunyai sejumlah fasilitas untuk komunikasi dengan sebuah komputer, *Arduino* atau mikrokontroler lainnya. Atmega328 menyediakan

serial komunikasi UART TTL (5V), yang tersedia pada pin digital 0 (RX) dan 1 (TX). Sebuah Atmega 16U2 pada channel board serial komunikasinya melalui USB dan muncul sebagai sebuah port virtual ke software pada komputer. Firmware 16U2 menggunakan driver USB COM standar, dan tidak ada driver eksternal yang dibutuhkan. LED RX dan TX pada board akan menyala ketika data sedang ditransmit melalui chip USB-to-serial dan koneksi USB pada komputer (tapi tidak untuk komunikasi serial pada pin 0 dan 1). Atmega328 juga mensupport komunikasi I2C (TWI) dan SPI. (Steven,dkk.2016)

5. Riset Otomatis

Arduino Uno didesain pada sebuah cara yang memungkinkannya untuk direset dengan *software* yang sedang berjalan pada komputer yang sedang terhubung. (Steven,dkk.2016)

6. USB

Arduino UNO mempunyai sebuah sekring reset yang memproteksi *port* USB komputer dari hubungan pendek dan arus lebih. Jika lebih dari 500 mA diterima port USB, sekring secara otomatis akan memutuskan koneksi sampai hubungan pendek atau kelebihan beban hilang. (Steven,dkk.2016)

USB berfungsi untuk memuat program dari komputer ke dalam papan, komunikasi serial antara papan ke komputer, dan memberi daya listrik kepada papan.

7. ICSP (*In-Circuit Serial Programming)*

Port ICSP memungkinkan pengguna untuk memprogram mikrokontroller secara langsung, tanpa melalui bootloader. Umumnya pengguna Arduino tidak melakukan ini sehingga ICSP tidak terlalu dipakai walaupun disediakan.

2.6.3 Karakteristik Fisik *Arduino Uno R3*

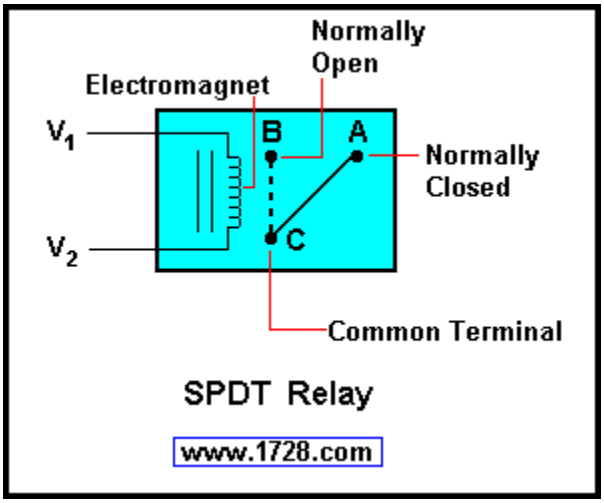
Panjang dan lebar maksimum dari PCB Arduino UNO masing-masingnya adalah 2.7 dan 2.1 inci, dengan konektor USB dan power jack yang memperluas dimensinya. Empat lubang sekrup memungkinkan board untuk dipasangkan ke sebuah permukaan atau kotak. Sebagai catatan, bahwa jarak antara pin digital 7

dan 8 adalah 160 mil. (0.16"), bukan sebuah kelipatan genap dari jarak 100 mil dari pin lainnya. (Steven,dkk.2016)

2.7 Relay

Relay adalah sebuah saklar yang dikendalikan oleh arus. Relay memiliki sebuah kumparan tegangan-rendah yang dililitkan pada sebuah inti. Terdapat sebuah armatur besi yang akan tertarik menuju inti apabila arus mengalir melewati kumparan. Armatur ini terpasang pada sebuah tuas berpegas. Ketika armatur tertarik menuju ini, kontak jalur bersama akan berubah posisinya dari kontak normal-tertutup ke kontak normal-terbuka. (Daniel,2015)

Relay dibutuhkan dalam rangkaian elektronika sebagai eksekutor sekaligus interface antara beban dan sistem kendali elektronik yang berbeda sistem power supplynya. Secara fisik antara saklar atau kontaktor dengan elektromagnet relay terpisah sehingga antara beban dan sistem kontrol terpisah. Bagian utama relay elektro mekanik adalah sebagai berikut. Kumparan elektromagnet Saklar atau kontaktor Swing Armatur Spring (Pegas). (Daniel,2015)



Gambar 2.6 Relay Sumber : www.1728.com

Relay dapat digunakan untuk mengontrol motor AC dengan rangkaian kontrol DC atau beban lain dengan sumber tegangan yang berbeda antara tegangan rangkaian kontrol dan tegangan beban. Rangkaian penggerak relay dapat dilihat pada gambar 2. Diantara aplikasi relay yang dapat ditemui diantaranya adalah : Relay sebagai kontrol ON/OF beban dengan sumber tegang berbeda.

Relay sebagai selektor atau pemilih hubungan. Relay sebagai eksekutor rangkaian delay (tunda) Relay sebagai protektor atau pemutus arus pada kondisi tertentu

2.7.1 Sifat-sifat Relay

Sifat-sifat Relay adalah sebagai berikut :

1. Impedansi kumparan, biasanya impedansi ditentukan oleh tebal kawat yang digunakan serta banyaknya lilitan. Biasanya impedansi berharga 1 – 50 KΩ Guna memperoleh daya hantar yang baik.
2. Daya yang diperlukan untuk mengoperasikan relay besarnya sama dengan nilai tegangan dikalikan arus.
3. Banyaknya kontak-kontak jangkar dapat membuka dan menutup lebih dari satu kontak sekaligus tergantung pada kontak dan jenis relaynya. Jarak antara kontak-kontak menentukan besarnya tegangan maksimum yang diizinkan antara kontak tersebut.

2.8 Sejarah Bahasa C

Akar dari bahasa C adalah bahassa BCPL yang dikembangkan oleh Martin Richard pada tahun 1967. Bahasa ini kemudian dikembangkan oleh Ken Thompson menjadi bahasa B pada tahun 1970. Perkembangan selanjutnya menjadi bahasa C oleh Dennis Richie sekitar 1970-an di Bell Telephone Laboratories (Sekarang adalah AT&T Bell Laboratories). Bahasa C pertama kali digunkana di *Computer Digital Equipment Corporation* PDP-11yang menggunkana system operasi UNIX, (±90v% sistem operasi UNIX ditulis dalam bahasa C) dan sampai sekarang bahasa ini telah dipergunakan secara praktis pada hampir semua sistem operasi. Selain itu, banyak bahasa pemrograman populer seperti PHP dan java menggunakan sintaks dasar yang mirip bahasa C. Pada tahun 1983, American National Standart Institute (ANSI) membentuk sebuah komite X3J11, untuk mengembangkan suatu spesifikasi standart untuk C dan berhasil diselesaikan pada tahun 1989. ANSI C didukung oleh kebanyakan compiler. Banyak kode C yang ditulis sekarang didasarkan pada ANSI C. Semua program ditulis dengan standart C dijamin akan berfungsi dengan baik pada platform lai

yang memiliki C. Tetapi banyak juga program C yang hanya dapat dikompilasi pada platform tertentu dengan compiler tertentu sehubungan dengan library non standart, misalnya untuk graphic. Pada tahun 1986, dikembangkan superset C (kompatibel dengan C, namun dilengkapi dengan kemampuan pemrograman berorientasi objek) oleh Brajne Stroustrup yaitu bahasa C++ (*C with Class*) dan sekarang bahasa yang banyak dipergunkana pada sistem operasi Microsoft Windows, sedangkan C tetap bahasa yang popoler di UNIX. Setelah proses standarisasi oleh ANSI, sepesikasi bahasa C masih relatif statis untuk beberapa saat, sedangkan C ++ terus berepolusi. Revisi standard tahun 1990, mengawali publikasi sebagai ISO 9899:1999 pada tahun1999. Standard ini disebut "C99" telah diadopsi sebagai ANSI standard pada tahun 2000.
(Dian Wirdasari, 2010)

2.8.1 Kelebihan Bahasa C

1. Bahasa C tersedia hampir disemua jenis komputer.
2. Kode bahasa C bersifat portable untuk semua jenis komputer. Suatu program yang ditulis dengan versi bahasa C tertentu akan dapat dikompilasi dengan versi bahasa C yang lain hanya dngan sedikit modifikasi.
3. C adalah bahasa pemrograman yag fleksibel. Dengan bahasa C, kita dapat menuis dan mengembangkan berbagai jenis program mulai dari operatig system, word processor, graphic processor, spreadsheets, ataupun kompiler untuk sutau bahasa pemrograman.
4. Bahasa C hanya menyediakan sedikit kata-kata kunci, hanya terdapat 32 kata kunci.
5. Proses executable program baha c lebih cepat.
6. Dukungan pustaka yang banyak.
7. C adalah bahasa yang terstruktur.
8. Bahasa C termasuk bahasa tingkat menengah.
9. Dibandingkan dnegan assembly, kode bahasa C lebih mudah di baca dan di tulis.

2.8.2 kekurangan Bahasa C

1. banyaknya operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai.
2. Para pemrogram C tingkat pemula umumnya belum pernah mengenal pointer dan tidak terbiasa menggunakannya. Keampuhan C justru terletak pada pointer.

2.9 Bahasa Pemrograman Arduino

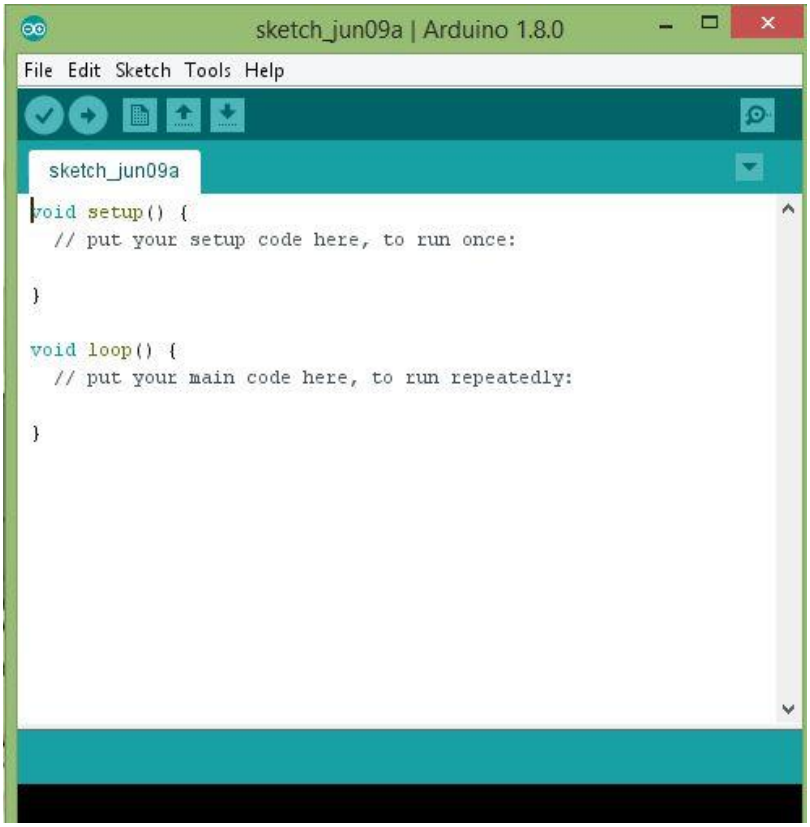
Bahasa Pemrograman Arduino adalah bahasa pemrograman utama yang digunakan untuk membuat program untuk Arduino *board*. Bahasa pemrograman Arduino menggunakan bahasa pemrograman C sebagai dasarnya. Karena menggunakan bahasa pemrograman C sebagai dasarnya, bahasa pemrograman Arduino memiliki banyak sekali kemiripan, walaupun beberapa hal telah berubah.

2.9.1 IDE *Arduino*

Untuk memulai program Arduino (untuk membuatnya melakukan apa yang kita inginkan) kita menggunakan IDE Arduino (*Integrated Development Environtment*), IDE Arduino adalah bagian software opensource yang memungkinkan kita untuk memprogram bahasa Arduino dalam bahasa C. IDE memungkinkan kita untuk menulis sebuah program secara step by step kemudian instruksi tersebut di upload ke papan Arduino.

Ada beberapa menu pilihan pada IDE Arduino yang mempunyai fungsi sebagai berikut:

1. Verify :Cek error dan lakukan kompilasi Kode.
2. Upload :Upload kode anda ke *board*/kontroler.
3. Serial Monitor :Membuka serial port monitor untuk melihat feedback/umpan balik dari *board* anda.



Gambar 2.7 Tampilan Program IDE *(Integrated Development Environtment)*
Sumber : Andi dan Oka, 2013

2.9.2 Kode – kode Dasar Program pada IDE *Arduino*

Seperti yang telah disebutkan sebelumnya bahwa untuk memprogram Arduino kita menggunakan sebuah kode program khusus yang mirip dengan struktur bahasa C.

- *Struktur*

Setiap program Arduino (biasa disebut sketch) mempunyai dua buah fungsi yang harus ada.

void setup( ) { }

Semua kode didalam kurung kurawal akan dijalankan hanya satu kali ketika program Arduino dijalankan untuk pertama kalinya.

void loop( ) { }

- *Syntax*

Berikut ini adalah elemen bahasa C yang dibutuhkan untuk format penulisan.

//(komentar satu baris)

Kadang diperlukan untuk memberi catatan pada diri sendiri apa arti dari kode-kode yang dituliskan. Cukup menuliskan dua buah garis miring dan apapun yang kita ketikkan dibelakangnya akan diabaikan oleh program.

/* */(komentar banyak baris)

Jika anda punya banyak catatan, maka hal itu dapat dituliskan pada beberapa baris sebagai komentar. Semua hal yang terletak di antara dua simbol tersebut akan diabaikan oleh program.

{ }(kurung kurawal)

Digunakan untuk mendefinisikan kapan blok program mulai dan berakhir (digunakan juga pada fungsi dan pengulangan).

;(titik koma)

Setiap baris kode harus diakhiri dengan tanda titik koma (jika ada titik koma yang hilang makaprogram tidak akan bisa dijalankan).

2.10 *Flowchart*

Flowchart merupakan diagram simbol yang menunjukkan arus data dan tahapan operasi dalam sebuah sistem yang digunakan baik oleh editor maupun oleh personal sistem. (Samuel Ratumurun, 2015)

Ada berbagai jenis *flowchart* secara teori, namun *flowchart* yang akan digunakan dalam memecahkan permasalahan distribusi dokumen sistem informasi keuangan penerimaan dan pengeluaran kas pada penulisan ini, adalah gabungan antara *flowchart* analitik, *flowchart* dokumen dan diagram distribusi formulir.Mengingat pemisahan dan pembagian tugas merupakan elemen pengendalian internal, membutuhkan teknik untuk membagi tugas pengolahan data antar personel dan atau departemen/bagian.

2.10.1 Jenis-Jenis *Flowchart*

Adapun jenis-jenis *flowchart* yang digunakan dalam penelitian ini adalah sebagai berikut.

1. *Flowchart* Analitik

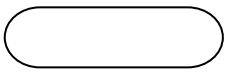
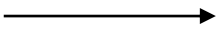
Flowchart Analitik adalah bagan alir yang ditandai dengan penggunaan simbol yang dihubungkan dengan garis. *Flowchart* analitik mengidentifikasi semua proses signifikan pada sebuah aplikasi, dengan penekanan pada pemrosesan tugas.

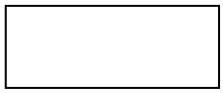
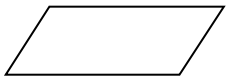
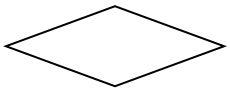
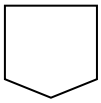
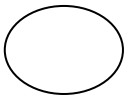
2. *Flowchart* Dokumen

Flowchart Dokumen bagan alir yang hanya terdiri dari simbol-simbol dokumen yang digunakan dalam *flowchart* tersebut. Tetapi, simbol lain pada dasarnya boleh saja digunakan untuk memperjelas suatu *flowchart*. Tujuan dari *flowchart* semacam ini adalah untuk mengetahui setiap dokumen yang digunakan dalam setiap sistem aplikasi dan mengidentifikasi titik awal dokumen, distribusi dokumen serta titik akhir setiap dokumen.

Diagram distribusi formulir, adalah diagram alir yang menggam-barkan distribusi setiap salinan formulir dalam sebuah organisasi. Dalam diagram ini, penekanannya terletak pada siapa yang akan mendapatkan formulir tertentu, bukan pada bagaimana setiap formulir akan diproses.

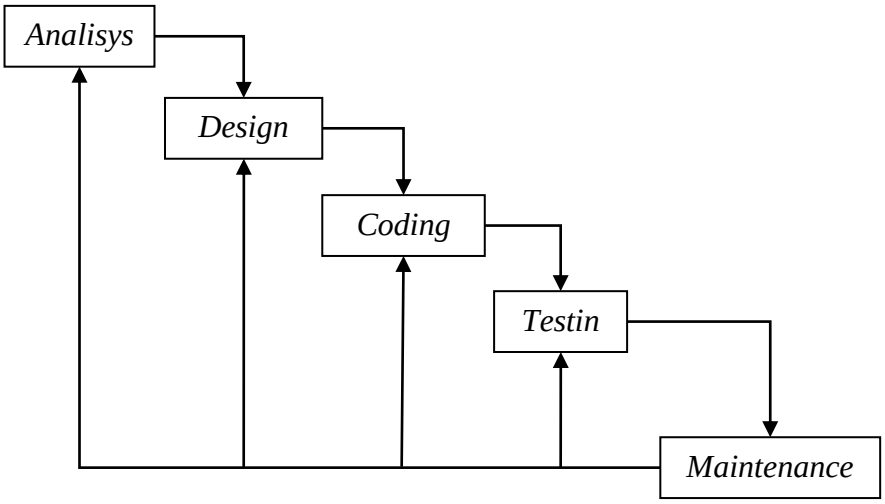
Tabel 2.2 Simbol-simbol yang digunakan pada flowchart

No	Simbol-Simbol	Nama	Keterangan
1.		***Terminator***	Simbol untuk permulaan (*start*) atau akhir (*stop*) dari suatu kegiatan.
2.		***Flow***	Simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain.

3.		***Processing***	Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer.
4.		***Input-Output***	Simbol yang menyatakan proses *input* dan *output* tanpa tergantung dengan jenis peralatannya .
5.		***Decision***	Simbol pemilihan proses berdasarkan kondisi yang ada.
6.		***Connector Symbol***	Simbol untuk keluar masuk atau penyambungan proses pada lembar atau halaman yang berbeda.
7.		***Connector Symbol***	Simbol untuk keluar masuk atau penyambungan proses pada lembar atau halaman yang sama.

2.11 Metode Pengembangan Sistem

Model pengembangan sistem yang digunakan ialah model *waterfall* atau air terjun. Model ini merupakan sebuah desain proses yang sequensial (berurutan) yang dalam progressnya terlihat seperti aliran air terjun (*waterfall*) dari proses perancangan konsep, inisialisasi project, analisis, desain, pembuatan *system* (*coding*), testing, produksi/implementasi dan perawatan (*maintenance*). Model ini adalah model klasik yang bersifat sistematis, berurutan dalam membangun software. Adapun tahapan-tahapan model *waterfall* adalah pada gambar 2.10.



Gambar 2.8 Siklus Pengembangan dengan Model *Waterfall*

1. *Analisys*

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user* . spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. *Design*

   Desain perangkat lunak adalah proses multilangkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data., arsitektur perangkat lunak, representasi antarmuka, prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisi ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan

3. *Coding*

   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. *Testing*

   Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. *Maintenance*

   Tidak menutup sebuah kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user* . perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau pernagkat lunak harus beradaptasi dengan lingkungan baru. Tahap ini dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.12 Pengujian Sistem Perangkat Lunak

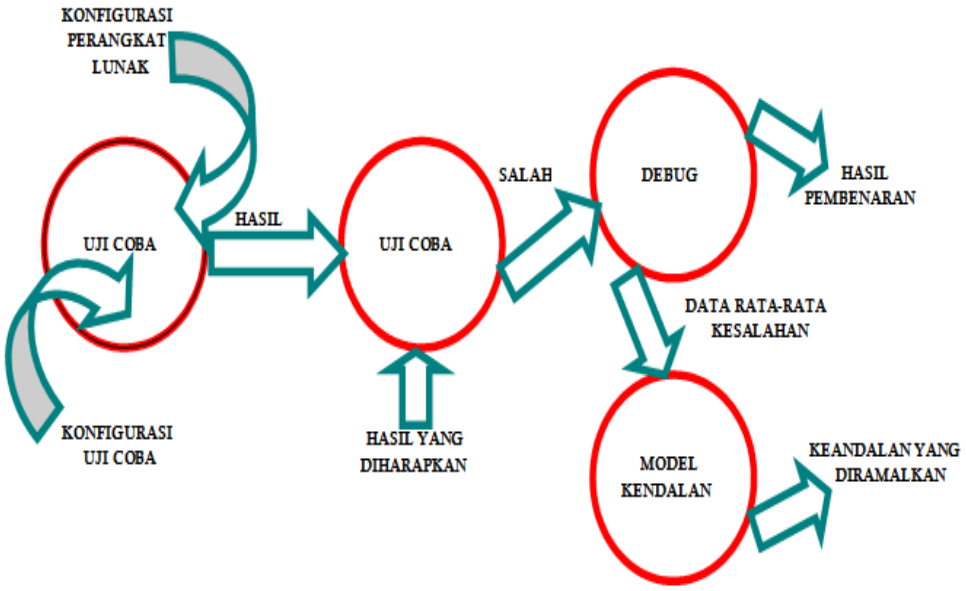
Pengujian adalah proses eksekusi program untuk menemukan kesalahan. Pengujian perangkat lunak *(testing)* merupakan bagian terpenting dalam pengembangan perangkat lunak *(software engineering)*. Pengujianya atau uji coba tersebut bertujuan untuk mencari sebanyak mungkin kesalahan *(bug error)* dan menemukan kesalahan yang sebelumnya tidak ditemukan, serta untuk mengurangi resiko yang terkandung dalam suatu sistem komputer. Suatu pengujian dikatakan berhasil apabila menemukan kesalahan-kesalahan yang belum terdeteksi. Secara umum pengujian dilakukan untuk menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya.

Beberapa aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak, yaitu :

a. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
b. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.

c. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Fase-fase pengujian perangkat lunak digambarkan sebagai berikut (Mediarman, 2009) :



Gambar 2.9 Fase Pengujian

Ada 2 tingkat yang tersedia pada proses uji coba, yaitu :

a. Konfigurasi perangkat lunak yang mencakup spesifikasi keperluan perangkat lunak, spesifikasi perancangan, *test case* dan program sumber.
b. Konfigurasi uji coba yang mencakup rencana dan prosedur uji coba, *test case* dan hasil yang diharapkan.

2.12.1 Strategi Pengujian

Strategi yang dibuat harus mengacu kepada resiko dan proses yang ada yang dapat mengurangi resiko tersebut. Dua komponen dari strategi pengujian adalah :

a. Faktor pengujian, merupakan resiko atau isu-isu yang ada yang perlu menjadi perhatian.
b. Fase pengujian, merupakan siklus pengembangan sistem dimana akan dilakukan proses pengujian.

2.12.2 Faktor Pengujian

Dalam merancang suatu pengujian, faktor-faktor yang menjadi resiko merupakan basis atau tujuan dari proses pengujian. Resiko-resiko yang berhubungan dengan proses pengujian ini disebut dengan faktor pengujian. Faktor-faktor pengujian tersebut merupakan kunci sukses suatu perangkat lunak :

a. *Correctness* (keakuratan dan kelengkapan data).
b. *File integrity* (pengelolaan *file* secara benar).
c. *Authorization* (kesesuaian data yang diproses dengan ketentuan-ketentuan secara manajemen).
d. *Audit trail* (kemampuan untuk mereka ulang suatu proses yang pernah terjadi).
e. *Continuity of processing* (kemampuan untuk selalu berjalan meskipun terjadi suatu problem).
f. *Service level* (pelayanan yang diberikan tidak mengecewakan pemakai).
g. *Access control* (perlindungan dari modifikasi, perusakan, salah penggunaan, yang disengaja atau tidak).
h. *Compliance* (kesesuaian sistem).
i. *Reliability* (ketepatan dan kemampuan dari sistem).
j. *Ease of use* (pemahaman sistem oleh pemakai).
k. *Maintenance* (kemampuan untuk mencari dan memperbaiki kesalahan dalam pengoperasian sistem).
l. *Portability* (kemampuan untuk mentransfer program dari satu konfigurasi perangkat keras dan atau perangkat lunak ke konfigurasi yang lain).
m. *Coupling* (kemampuan untuk menghubungkan komponen dalam sistem aplikasi dengan semua sistem aplikasi dalam suatu lingkungan proses).
n. *Performance* (rata-rata dari sumber daya komputasi dari kode yang diperlukan oleh sistem untuk menjalankan fungsi yang telah ditetapkan).

o. *Ease of operation* (kemampuan untuk mengintegrasikan sistem ke dalam lingkungan operasi dan kemudian mengoperasikan sistem apliakasi, baik yang manual maupun otomatis).

2.12.3 Membuat Srategi Pengujian

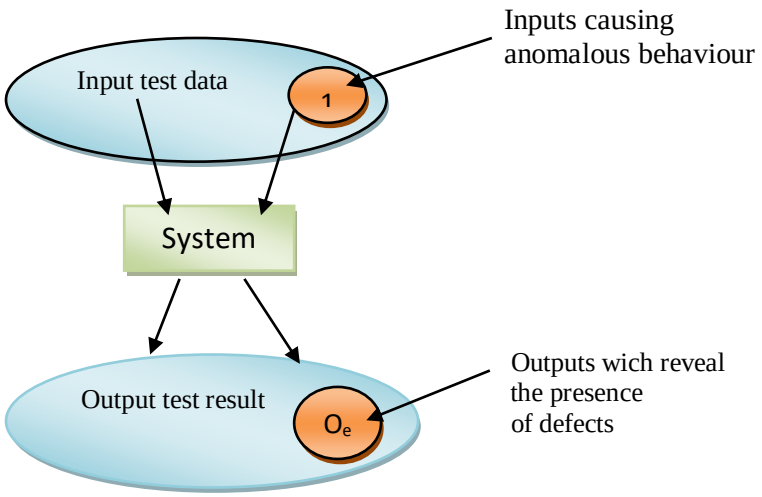
Strategi pengujian disusun berdasarkan resiko-resiko yang telah dipilih. Pada umumnya terdapat empat langkah dalam pembuatan strategi pengujian, sedangkan untuk perangkat lunak yang bersifat khusus memerlukan sedikit modifikasi. Langkah-langkah adalah :

a. Seleksi dan perangkingan faktor pengujian. Dalam banyak kasus, dari faktor-faktor yang ada cukup diambil 3 – 7 faktor pengujian. Proses ini dilakukan dengan cara membuat daftar dalam satu matriks dan diurutkan dari faktor yang paling penting sampai yang kurang penting.
b. Identifikasi fase pengembangan sistem. Biasanya sesuai dengan metodologi pengembangan sistem yang digunakan. Fase-fase yang ada kemudian ditulis menjadi salah satu komponen dalam matriks.
c. Identifikasi resiko-resiko bisnis yang berhubungan dengan sistem yang sedang dikembangkan. Resiko-resiko tersebut kemudian dirangking dari tingkat *high, medium, low.*
d. Menempatkan resiko yang diperoleh ke dalam matriks. Kemudian menentukan fase dan faktor yang berhubungan dengan resiko yang telah dibuat.

2.13 Pengujian Kotak Hitam (*Black Box Testing*)

Menurut (myers,1979), *Black Box Testing* adalah proses menjalankan program dengan maksud menemukan kesalahn.

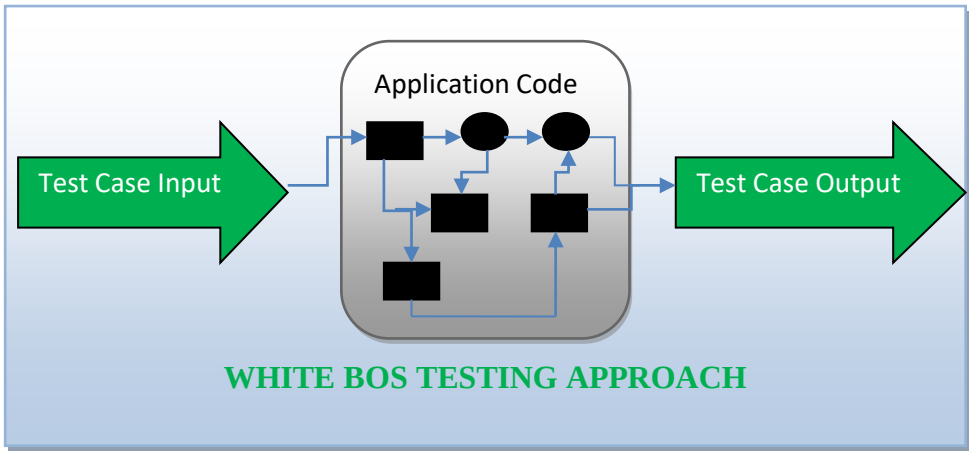
Menurut (IEEE, 1990), *Black Box Testing* adalah pengujian yang mengabaikan mekanisme internal sistem atau komponen dan fokus semata-mata pada output yang dihasilkan yang merepon input yang dipilih dan kondisi eksekusi. Pengujian yang dilakukan untuk mengevaluasi pemenuhan sistem atau komponen dengan kebutuhan fungsional tertentu.



Gambar 2.10 Pengujian Kotak Hitam **(***Black Box Testing*)
Sumber : Umi Gusti Salamah, 2013

2.14 Pengujian Kotak Putih (*White Box Testing*)

White Box Testing merupakan cara pengujian dengan melihat kedalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kasalahan atau tidak. Jika ada modul yang menghasilkan output yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris-baris program, variabel, dan parameter yang terlibat pada unit tersebut akan dicek satu persatu dan diperbaiki, kemudian di-*compile* ulang.



Gambar 2.11 Pengujian Kotak Putih (*White Box Testing*)
Sumber : Umi Gusti Salamah, 2013