BAB 2

LANDASAN TEORI

2.1. Konsep Dasar Kontrol

Suatu sistem kontrol otomatis dalam suatu proses kerja berfungsi mengendalikan proses tanpa adanya campur tangan manusia atau otomatis. Konsep dasar pengontrolan sudah ada sejak abad-18 yang dipelopori James Watt yang membuat kontrol mesin uap, Nyquis (1932) membuat sistem pengendali uang tertutup, Hazem (1943) membuat servo mekanik dan masih banyak yang lainnya. Kontrol otomatis mempunyai peran penting dalam dunia industri modern saat ini. Seiring perkembangan kemajuan ilmu pengetahuan dan teknologi, sistem kontrol otomatis telah mendorong manusia untuk berusaha mengatasi segala permasalahan yang timbul di sekitarnya dengan cara yang lebih mudah, efisien dan efektif. Adanya kontrol otomatis secara tidak langsung dapat menggantikan peran manusia dalam meringankan segala aktifitasnya[1].

Sedangkan untuk fungsi kendali itu sendiri meliputi :

1. Menerima input dan output referensi (sesuai dengan tingkah laku sistem yang diinginkan).
2. Menerima informasi output melalui elemen baik dan membandingkan dengan output mengambil suatu keputusan melalui perhitungan-perhitungan yang cukup rumit.

Dilihat dari prinsipnya, fungsi dasar suatu kendali adalah mencakup operasi pengukuran, perbandingan, perhitungan dan koreksi. Dimana pengukuran merupakan operasi otomatisasi penafsiran mengenal suatu proses dikontrol oleh sistem. Perbandingan merupakan pengujian kesetaraan antara nilai yang diukur dengan yang diharapkan. Perhitungan akan memberikan keyakinan yang menunjukkan seberapa besar perbedaan antara nilai yang diukur dengan nilai yang diharapkan.

2.2 Jaringan GSM

Global System for Mobile communication (GSM) adalah sebuah standar global untuk komunikasi bergerak digital. GSM adalah nama dari sebuah group standarisasi telpon bergerak selular di Eropa yang beroperasi pada daerah frekunensi 900 MHz. GSM saat ini banyak digunakan di negara-negara di dunia[2].

2.2.1 *Short Message Service (*SMS*)*

Short Message Service (SMS) adalah sebuah layanan yang banyak diaplikasikan pada sistem komunikasi tanpa kabel *wireless*, yang memungkinkan pengguna untuk melakukan pengiriman pesan dalam bentuk alphanumeric antara terminal pelanggan dengan sistem eksternal seperti *e-mail*, *paging*, *voice mail*, dan lain-lain.

SMS mulai diperkenalkan di eropa sejak tahun 1991 dengan adanya standardisasi dalam bidang wireless digital yang disebut *Global System for Mobile Communication* (GSM). GSM adalah sistem pelopor seluler yang dikembangkan secara universal oleh *European Telecomunication Standards Institute* (ETSI) dan dengan GSM inilah aplikasi SMS dapat dijalankan.

Mekanisme dalam sistem SMS adalah melakukan pengiriman *short message* dari terminal pelanggan ke terminal lain. Layanan SMS merupakan sebuah layanan yang bersifat *nonreal time* dimana sebuah *short message* dapat di -submit ke suatu tujuan, tidak peduli apakah tujuan tersebut aktif atau tidak. Bila dideteksi bahwa tujuan tidak aktif, maka sistem akan menunda pengiriman ke tujuan hingga tujuan aktif kembali.

Prinsip dasar sistem SMS akan menjamin *delivery* dari *short message* hingga sampai tujuan. Kegagalan pengiriman yang bersifat sementara seperti tujuan tidak aktif akan selalu teridentifikasi sehingga pengiriman ulang *short message* akan selalu dilakukan kecuali bila aturan bahwa *short message* yang telah melampaui batas waktu tertentu harus dihapus dan dinyatakan gagal dikirim.

Karakteristi utama SMS adalah SMS merupakan sebuah sistem pengiriman data dalam paket yang bersifat *out-of-band* dengan bandwidth yang kecil. Dengan karakteristik ini, pengiriman dengan suatu *burst* data yang pendek dapat dilakukan dengan efesiensi yang sangat tinggi[2].

2.2.2 *Short Message Service Centre* (SMSC)

Short Message Service Centre (SMSC) adalah kombinasi perangkat keras dan perangkat lunak yang bertanggung jawab memperkuat, menyimpan dan meneruskan antara SME dan piranti bergerak.

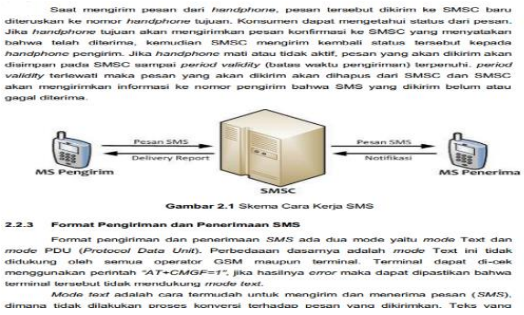
SMSC harus memiliki kehandalan, kapasitas pelanggan dan throughput pesan yang sangat tinggi. Selain itu SMSC juga harus dapat diskalakan dengan mudah untuk mengakomodasi peningkatan permintaan SMS dalam jaringan yang ada.

SMSC merupakan sebuah entitas yang bertanggung jawab untuk menyimpan *routing short message* dari satu titik ke titik yang lain yang merupakan tujuan. Sebuah SMSC harus mempunyai keandalan yang tinggi, kapasitas yang cukup, dan throughput yang memadai dalam menangani trafik short message. Sistem harus bersifat fleksibel dan scalabel agar dapat mengakomdasi pertumbuhan permintaan layanan SMS[2].

2.2.3 Cara Kerja SMS

Implementasi layanan SMS, operator menyediakan apa yang disebut sebagai *SMS Center* (SMSC). Secara fisik SMSC dapat berwujud sebuah PC biasa yang mempunyai interkoneksivitas dengan jaringan GSM. SMSC inilah yang akan melakukan manajemen pesan SMS, baik untuk pengiriman, pengaturan, antrian SMS, penerimaan SMS.

Saat mengirim pesan dari *handphone*, pesan tersebut dikirim ke SMSC baru diteruskan ke nomor *handphone* tujuan. Konsumen dapat mengetahui status dari pesan. Jika *handphone* tujuan akan mengirimkan pesan konfirmasi ke SMSC yang menyatakan bahwa telah diterima, kemudian SMSC mengirim kembali status tersebut kepada handphone pengirim. Jika *handphone* mati atau tidak aktif, pesan yang akan dikirim akan disimpan pada SMSC sampai *period validity* (batas waktu pengiriman) terpenuhi. *period validity* terlewati maka pesan yang akan dikirim akan dihapus dari SMSC dan SMSC akan mengirimkan informasi ke nomor pengirim bahwa SMS yang dikirim belum atau gagal diterima[2]..



Gambar 2.1 Skema Cara Kerja SMS

2.2.3 Format Pengiriman dan Penerimaan SMS

Format pengiriman dan penerimaan *SMS* ada dua mode yaitu *mode* Text dan *mode* PDU (*Protocol Data Unit*). Perbedaaan dasarnya adalah *mode* Text ini tidak didukung oleh semua operator GSM maupun terminal. Terminal dapat di-cek menggunakan perintah "*AT+CMGF=1*", jika hasilnya *error* maka dapat dipastikan bahwa terminal tersebut tidak mendukung *mode text*.

Mode text adalah cara termudah untuk mengirim dan menerima pesan (*SMS*), dimana tidak dilakukan proses konversi terhadap pesan yang dikirimkan. Teks yang

***Gambar** 2.1 cara kerja SMS*

2.2.4 AT Command

Perintah *Hayes AT Command* (AT) digunakan untuk berkomunikasi dengan terminal (modem) melalui gerbang serial pada komputer. Dengan penggunaan perintah AT, dapat diketahui atau dibaca kondisi dari terminal, seperti mengetahui kondisi sinyal, kondisi baterai, mengirim pesan, membaca pesan, menambah item pada daftar telepon, dan sebagainya. Pada tabel 2.1 diperlihatkan beberapa jenis perintah AT yang berhubungan dengan penanganan pesan-pesan SMS.

Komunikasi data antara telepon seluler dengan *peripheral* lain seperti mikrokontroler dilakukan secara serial menggunakan perintah-perintah *Hayes AT Command* (AT). Dengan mengirimkan telepon seluler untuk melakukan apa yang diperintahkan[2].

Pada tabel 2.1 merupakan perintah – perintah yang digunakan untuk mengirim SMS:

Perintah	Fungsi
AT+CMGC	Mengirim sebuah perintah SMS
AT+CMGD	Menghapus sebuah SMS
AT+CMGF	Memori
AT+CMGR	SMS format
AT+CMGS	Membaca dalam sebuah SMS
AT+CMGL	Daftar SMS yang terdapat di HP
AT+CSCA	Mengirim sebuahh SMS alamat dari pusat SMS servis
AT+CSCA	Alamat dari sebuah SMS *service center*
AT+CMNI	Menamplkan pesan yang masuk

Tabel 2.1 *AT Command* pada SMS

2.3 Mikrokontroler ATMega 328P

ATMega 328P adalah mikrokontroller keluaran dari Atmel yang mempunyai arsitektur *Reduce Instruction Set Computer* (RISC) yang dimana setiap proses eksekusi data lebih cepat dari pada arsitektur *Completed Instruction Set Computer* (CISC). Mikrokontroller ATmega 328P 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. Mikrokontroler ATmega328P beroperasi pada frekuensi *clock* sampai 16 Mhz. ATmega328P memiliki dua *Power Saving Mode* yang dapat dikontrol melalui *software*, yaitu *Idle Mode* dan *Power Down Mode*. Pada Idle Mode, CPU tidak aktif sedangkan isi RAM tetap dipertahankan dengan *timer/counter*, *serial port* dan *interrupt system* tetap berfungsi. Pada *Power Down Mode*, isi RAM akan disimpan tetapi osilatornya tidak akan berfungsi sehingga semua fungsi dari chip akan berhenti sampai mendapat reset secara *hardware*[3]. Pada table 2.2 merupakan karakteristik dari mikrokontroler ATMega 328P.

Mikrokontroler	ATMega 328
Operasi Voltage	5 V
Input Voltage	7 – 12 V (rekomendasi)
Input Voltage	6 – 20 V (limit)
I/O	14 pin (6 pin untuk PWM)
Arus	50 Ma
Flash Memory	32 KB
Bootloader	SRAM 2 KB
EEPROM	1 KB
Kecepatan	16 MHz

Tabel 2.2 Karakteristik Mikrokontroler ATMega 328P

2.4 Bahasa Pemrograman Arduino

Arduino menggunakan pemrograman dengan bahasa C. Berikut ini adalah penjelasan tentang karakter dari bahasa C, yaitu:

1. Struktur

   Setiap program Arduino (biasa disebut *sketch*) mempunyai dua buah fungsi yang harus ada, yaitu :

   a. void setup( ) { }

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

b. void loop( ) { }

Fungsi ini akan dijalankan setelah setup (fungsi *void setup*) selesai. Setelah dijalankan satu kali fungsi ini akan dijalankan lagi, dan lagi secara terus menerus sampai catu daya (*power*) dilepaskan.

2. Syntax

Berikut ini adalah elemen bahasa C yang dibutuhkan untuk format penulisan, yaitu:

a. //(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.

b. **/* */**(komentar banyak baris)

Jika memerlukan 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.

c. **{ }**(kurung kurawal)

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

3. Variabel

Sebuah program secara garis besar dapat didefinisikan sebagai instruksi untuk memindahkan angka dengan cara yang cerdas. Variabel inilah yang digunakan untuk memindahkannya.

a. int (*integer*)

Digunakan untuk menyimpan angka dalam 2 byte (16 bit). Tidak mempunyai angka desimal dan menyimpan nilai dari -32,768 dan 32,767.

b. long (*long*)

Digunakan ketika integer tidak mencukupi lagi. Memakai 4 byte (32 bit) dari memori (RAM) dan mempunyai rentang dari -2,147,483,648 dan 2,147,483,647.

c. boolean (boolean)

Variabel sederhana yang digunakan untuk menyimpan nilai *TRUE* (benar) atau *FALSE* (salah). Sangat berguna karena hanya menggunakan 1 bit dari RAM.

d. *float* (*float*)

Digunakan untuk angka desimal (floating point). Memakai 4 byte (32 bit) dari RAM dan mempunyai rentang dari -3.4028235E+38 dan 3.4028235E+38.

e. char (*character*)

Menyimpan 1 karakter menggunakan kode ASCII (misalnya 'A' = 65). Hanya memakai 1 byte (8 bit) dari RAM.

4. Operator Aritmatika

Operator yang digunakan untuk memanipulasi angka (bekerja seperti matematika yang sederhana). Pada tabel 2.3 merupakan operator dari matematika.

Tabel 2.3 Operator Aritmatika[5]

Operator Aritmatika	Keterangan
=	Membuat sesuatu menjadi sama dengan nilai yang lain (misalnya: x = 10 * 2, x sekarang sama dengan 20).
%	Menghasilkan sisa dari hasil pembagian suatu angka dengan angka yang lain (misalnya: 12 % 10, ini akan menghasilkan angka 2).
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian

5. Operator Pembanding

Operator pembandng ini digunakan untuk membandingkan nilai logika. Pada tabel 2.4 merupakan tabel dari operator pembanding.

Tabel 2.4 Operator Pembanding[5]

Operator Kondisi	Keterangan
==	Sama dengan (misalnya: 12 == 10 adalah FALSE (salah) atau 12 == 12 adalah TRUE (benar))
!=	Tidak sama dengan (misalnya: 12 != 10 adalah TRUE (benar) atau 12 != 12 adalah FALSE (salah))
<	Lebih kecil dari (misalnya: 12 < 10 adalah FALSE (salah) atau 12 < 12 adalah FALSE (salah) atau 12 < 14 adalah TRUE (benar))
>	Lebih besar dari (misalnya: 12 > 10 adalah TRUE (benar) atau 12 > 12 adalah FALSE (salah) atau 12 > 14 adalah FALSE (salah))

1. Struktur Pengaturan

Program sangat tergantung pada pengaturan apa yang akan dijalankan berikutnya, berikut ini adalah elemen dasar pengaturan.

a. if..else

```
if (kondisi) { }
else if (kondisi) { }
else { }
```

Dengan struktur seperti diatas program akan menjalankan kode yang ada di dalam kurung kurawal jika kondisinya TRUE, dan jika tidak (FALSE) maka akan diperiksa apakah kondisi pada *else if* dan jika kondisinya FALSE maka kode pada *else* yang akan dijalankan.

b. For

```
for (int i = 0; i < #pengulangan; i++) { }
```

Digunakan bila anda ingin melakukan pengulangan kode di dalam kurung kurawal beberapa kali, ganti #pengulangan dengan jumlah pengulangan yang diinginkan. Melakukan penghitungan ke atas dengan *i*++ atau ke bawah dengan *i*–.

2. Digital

a. pinMode(pin, mode)

Digunakan untuk menetapkan mode dari suatu pin, *pin* adalah nomor pin yang akan digunakan dari 0-19 (pin analog 0-5 adalah 14-19). Mode yang bisa digunakan adalah *INPUT* atau *OUTPUT*.

b. digitalWrite(pin, value)

Ketika sebuah pin ditetapkan sebagai *OUTPUT*, pin tersebut dapat dijadikan *HIGH* (ditarik menjadi 5 volts) atau *LOW* (diturunkan menjadi ground).

c. digitalRead(pin)

Ketika sebuah pin ditetapkan sebagai INPUT maka dapat digunakan kode ini untuk mendapatkan nilai pin tersebut apakah *HIGH* (ditarik menjadi 5 volts) atau *LOW* (diturunkan menjadi *ground*).

3. Analog

a. analogWrite(pin, value)

Beberapa pin pada Arduino mendukung *Pulse Width Modulation* (PWM) yaitu pin 3, 5, 6, 9, 10, 11. Ini dapat merubah pin hidup (*on*)atau mati (*off*) dengan sangat cepat sehingga membuatnya dapat berfungsi layaknya keluaran analog. *Value* (nilai) pada format kode tersebut adalah angka antara 0 (0% duty cycle ~ 0V) dan 255 (100% duty cycle ~ 5V).

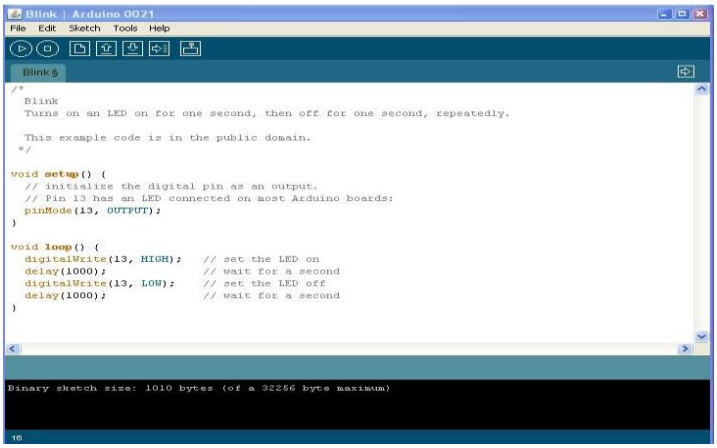
b. analogRead(pin)

Ketika pin analog ditetapkan sebagai INPUT anda dapat membaca keluaran voltase-nya. Keluarannya berupa angka antara 0 (untuk 0 volts) dan 1024 (untuk 5 volts)[5].

2.5 Arduino IDE

Integrated Development Environment (IDE) Arduino adalah *software* yang dijalankan dengan menggunakan Java dan terdiri dari beberapa fitur seperti *editor* program, *compiler*, dan *uploader*. *Editor* program adalah sebuah window yang memungkinkan pengguna menulis dan

mengedit program dalam bahasa *Processing*. *Compiler* berfungsi mengubah kode program (bahasa C Arduino) menjadi Bahasa mesin dalam bentuk file *.hex (*hexadecimal*). *Uploader* adalah sebuah modul yang memuat kode biner dari komputer ke dalam memori didalam Board Arduino[4].



Gambar 2.2 menunjukkan tampilan Arduino IDE.

2.6 SMS Gateway

Pada prinsipnya, SMS *Gateway* adalah sebuah perangkat lunak yang menggunakan bantuan komputer danmemanfaatkan teknologi seluler yang diint egrasikan guna mendistnbusikan pesan-pesan yang di-generate lewat sistem informasi melalui media SMS yang di-handle oleh jaringan seluler. Modul GSM SIM 800L ini merupakan modul GSM yang dapat untuk project mikrokontroler sepert monitoring lewat SMS, menyalakan atau kontrol nyala listrik lewat SMS[6].

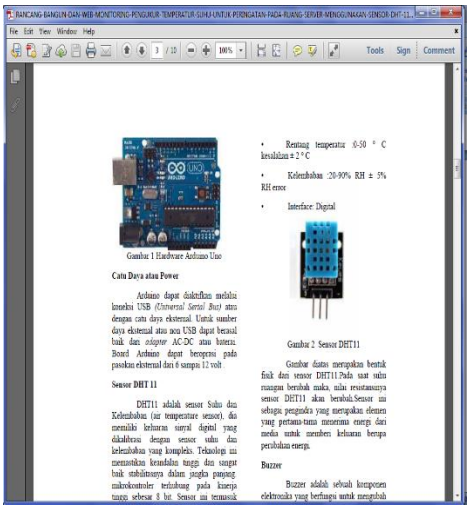
2.7 DHT22

DHT22 adalah sensor Suhu dan kelembaban (*air temperature sensor*), yang memiliki keluaran sinyal digital yang dikalibrasi dengan sensor suhu dan kelembaban yang kompleks. Teknologi ini memastikan keandalan tinggi dan sangat baik stabilitasnya dalam jangka panjang. mikrokontroler terhubung pada kinerja tinggi sebesar 8 bit. Sensor ini termasuk elemen resistif dan perangkat pengukur suhu NTC. Memiliki kualitas yang sangat baik, respon cepat, kemampuan anti-gangguan dan keuntungan biaya tinggi kinerja[6]. Setiap sensor DHT22 memiliki fitur

kalibrasi sangat akurat dari kelembaban ruang kalibrasi. Koefisien kalibrasi yang disimpan dalam memori program OTP, sensor internal mendeteksi sinyal dalam proses, atau koefisien kalibrasi. Sistem antarmuka tunggal-kabel serial terintegrasi untuk menjadi cepat dan mudah. Kecil ukuran, daya rendah, sinyal transmisi jarak hingga 20 meter, sehingga berbagai aplikasi dan bahkan aplikasi yang paling menuntut. Produk ini 4-pin pin baris paket tunggal[7].

Spesifikasi :

1. Pasokan Voltage: 5 V
2. Rentang temperatur :0-50 ° C kesalahan ± 2 ° C
3. Kelembaban :20-90% RH ± 5% RH error
4. Interface: Digital

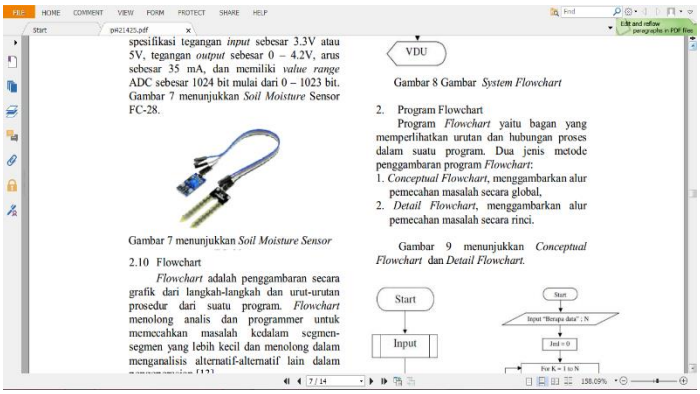


Gambar 2.3 Sensor DHT 22

2.8 Soil Moisture Sensor FC-28

Soil Moisture Sensor FC-28 adalah sensor kelembaban yang dapat mendeteksi kelembaban dalam tanah. Sensor ini sangat sederhana, tetapi ideal untuk memantau taman kota, atau tingkat air pada tanaman pekarangan. Sensor ini terdiri dua *probe* untuk melewatkan arus melalui tanah, kemudian membaca resistansinya untuk mendapatkan nilai tingkat kelembaban. Semakin banyak air membuat tanah lebih mudah menghantarkan listrik (*resistansi* kecil), sedangkan tanah yang kering sangat sulit menghantarkan listrik (*resistansi* besar). Sensor ini sangat membantu untuk mengingatkan tingkat kelembaban pada tanaman atau memantau kelembaban tanah. *Soil Moisture* Sensor FC-28 memiliki spesifikasi tegangan *input* sebesar 3.3V atau 5V, tegangan *output* sebesar

0 – 4.2V, arus sebesar 35 mA, dan memiliki *value range* ADC sebesar 1024 bit mulai dari 0 – 1023 bit[8]. Gambar 2.4 menunjukkan *Soil Moisture* Sensor FC-28.


spesifikasi tegangan *input* sebesar 3.3V atau 5V, tegangan *output* sebesar 0 – 4.2V, arus sebesar 35 mA, dan memiliki *value range* ADC sebesar 1024 bit mulai dari 0 – 1023 bit. Gambar 7 menunjukkan *Soil Moisture* Sensor FC-28.

Gambar 7 menunjukkan *Soil Moisture Sensor*

2.10 Flowchart

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam

VDU

Gambar 8 Gambar *System Flowchart*

2. Program Flowchart

Program *Flowchart* yaitu bagan yang memperlihatkan urutan dan hubungan proses dalam suatu program. Dua jenis metode penggambaran program *Flowchart*:

1. *Conceptual Flowchart*, menggambarkan alur pemecahan masalah secara global,
2. *Detail Flowchart*, menggambarkan alur pemecahan masalah secara rinci.

Gambar 9 menunjukkan *Conceptual Flowchart* dan *Detail Flowchart*.

Start

Input

Gambar 2.4 *Soil Moisture Sensor FC-28*[8]

2.8.1 Kategori Kondisi Kelembaban Tanah

Berdasarkan pembacaan nilai data sensor, nilai range pembacaan sensor berkisar dari angka 0 – 1023 bit yang menunjukan nilai kelembaban suatu tanah. Pembacaan nilai yang semakin tinggi dari sensor menunjukan bahwa semakin kering kondisi kelembaban tanah dan sebaliknya semakin rendah nilai yang dibaca oleh sensor maka semakin lembab kondisi kelembaban tanah. Dalam hal ini untuk lebih mempermudah penelitian dilakukan perubahan nilai sensor menjadi nilai persen (%). Adapun perubahan nilai tersebut menggunakan Persamaan (1).

$$Nilai\ Persen = \frac{1023 - nilai\ sensor}{1023} \times 100\% \quad (1)$$

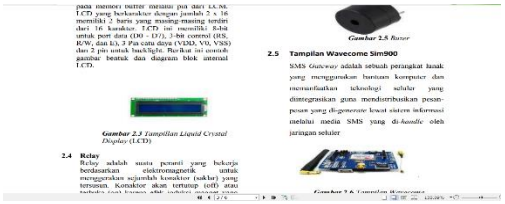
Persamaan (1) menjelaskan nilai sensor yang diperoleh dikurangkan dengan nilai range sensor yang berjumlah 1023 dan dikalikan 100%, sehingga didapatkan nilai sebesar 0,1023 untuk setiap 0,01 nilai persen. Maksud dari perubahan nilai ini agar alat dapat langsung mendeteksi persentase kelembaban tanah. yang dapat diartikan semakin rendah persentase yang dideteksi oleh alat maka semakin kering kondisi kelembaban tanah sedangkan semakin tinggi persentase kelembaban maka semakin lembab kondisi kelembaban tanah[8]. Adapun penentuan kategori kondisi kelembaban tanah ditunjukkan oleh Tabel 2.5.

Tabel 2.5 Kategori kondisi kelembaban tanah[8]

NO	Kelembaban Tanah(%)	Kategori kondisi kelembaban tanah
1	0 – 50%	Kering
2	51 – 100%	Lembab

2.9 Liquid Crystal Display (LCD)

Liquid Crystal Display (LCD) adalah sebuah alat peraga untuk menampilkan suatu karakter yang dibangun dari dot pattern pada permukaan yang terdiri dari *liquid crystal fluid* diantara dua lapisan gelas (*two of glass whose inner surface*). Pada saat diberi tegangan, cahaya diserap oleh *crystal fluid* dan kemudian membentuk *dot pattern*. Pembentukan pola disesuaikan dengan pemberian tegangan masing-masing dot pattern. Sedangkan *Liquid Crystal Display Module* (LCM) adalah sebuah LCD yang telah dilengkapi pengontrol dan memori sebagai penyimpan pola karakter. LCM memberikan kemudahan dalam pemakaiannya. Untuk menampilkan sebuah karakter, pemakai hanya mengisikan data dan kontrol pada memori buffer melalui pin dari LCM. LCD yang berkarakter dengan jumlah 2 x 16 memiliki 2 baris yang masing-masing terdiri dari 16 karakter. LCD ini memiliki 8-bit untuk port data (D0 - D7), 3-bit control (RS, R/W, dan E), 3 Pin catu daya (VDD, V0, VSS) dan 2 pin untuk backlight[9]. Pada gambar 2.5 merupakan bentuk dan diagram blok internal LCD.



Gambar 2.5 *Licuid Crystal Display* (LCD)

Tabel 2.6 Pin LCD dan Fungsinya[9]

PIN	Nama Pin	Fungsi
1	VSS	*Ground Voltage*
2	VCC	+5V
3	VEE	*Contrast Voltage*
4	RS	*Register Select* 0 = Instruction Register 1 = Data Register
5	R/W	*Read/Write, to choose write or read mode* 0 = write mode 1 = read mode
6	E	*Enable* 0 = start to lacht data to LCD character 1 = disable
7	DB0	Data bit ke-0 (LSB)
8	DB1	Data bit ke-1
9	DB2	Data bit ke-2
10	DB3	Data bit ke-3
11	DB4	Data bit ke-4
12	DB5	Data bit ke-5
13	DB6	Data bit ke-6
14	DB7	Data bit ke-7
15	BPL	Back Plane Light
16	GND	Ground Voltage

2.10 Relay

Relay adalah komponen elektronika berupa saklar elektronik yang digerakkan oleh arus listrik. Secara prinsip, *Relay* merupakan tuas saklar dengan lilitan kawat pada batang besi (*solenoid*) di dekatnya. Ketika *solenoid* dialiri arus listrik, tuas akan tertarik karena adanya gaya magnet yang terjadi pada *solenoid* sehingga kontak saklar akan menutup. Pada saat arus

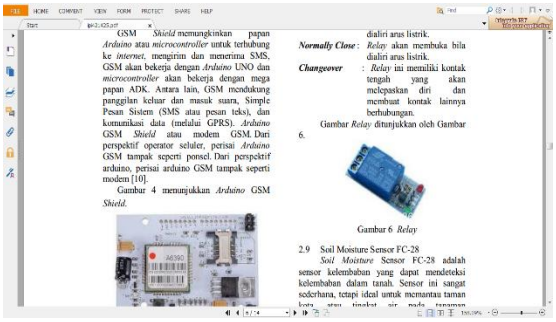
dihentikan, gaya magnet akan hilang, tuas akan kembali ke posisi semula dan kontak saklar kembali terbuka[8].

Susunan kontak pada *Relay* adalah :

1. Normally Open : *Relay* akan menutup bila dialiri arus listrik.

2. Normally Close : *Relay* akan membuka bila dialiri arus listrik.

3. Changeover : *Relay* ini memiliki kontak tengah yang akan melepaskan diri dan membuat kontak lainnya berhubungan[8].



Gambar 2.6 Relay[8]

1. Unified Modelling Language (UML)

Unified Modeling Language (UML) adalah salah satu alat bantu pembangunan perangkat lunak yang memiliki keunggulan dengan konsep pengembangan sistem yang berorientasi objek. UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem membuat cetak biru atau mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancangan mereka dengan unsur yang lain. UML dalam sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan artifact dari sistem software, untuk memodelkan bisnis, dan sistem non-software lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD (*Object Oriented Analysis and Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, konstruksi dan mendokumentasikan artifact yang terdapat dalam sistem. Artifact adalah potongan informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software. Artifact dapat berupa model, deskripsi atau software [11].

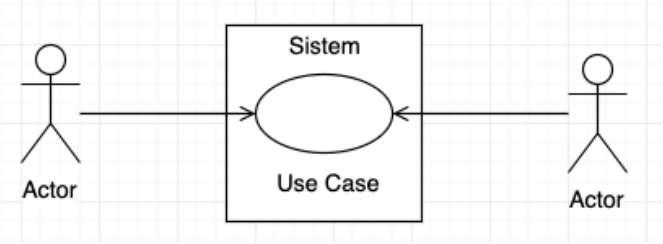
1. Diagram UML

UML dalam mendokumentasikan rancangan menyediakan berbagai macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek [11]. Namun, dalam penelitian ini hanya menggunakan empat macam diagram saja untuk memodelkannya. Empat macam diagram yang digunakan diantaranya yaitu:

1. Use Case Diagram

   Use case diagram adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi

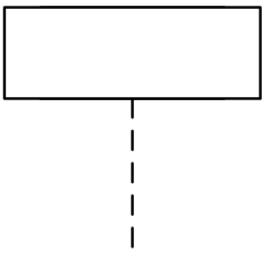
oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa dikatan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh tujuan umum pengguna. *Use case* merupakan alat bantu guna menstimulasi pengguna potensial untuk mengatakan tentang suatu sistem dari sudut pandangnya. Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud menggunakan sebuah sistem. Karena sistem pengembangan tradisional sering ceroboh dalam melakukan analisis, akibatnya pengguna seringkali sulit menemukan jawaban tatkala dimintai masukan tentang sesuatu. Diagram *use case* menunjukan 3 aspek dari sistem yaitu : *actor*, *use case* dan *sistem* atau *sub sistem boundary*. *Actor* mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan *use case*.



Gambar 2.7 Use Case Model [11]

2. Sequance Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukan sejumlah contoh objek dan *message* (pesan) yang diletakan diantara objek-objek ini di dalam *use case*. Komponen utama *sequence diagram* terdiri atas objek yang ditulisan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukan dengan *progress vertical*. Objek diletakan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram, istilah objek dikenal juga dengan *participant*, setiap *participant* terhubung dengan garis titik-titik yang disebut *lifeline*. Sepanjang *lifeline* ada kotak yang disebut *activation*, *activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*.

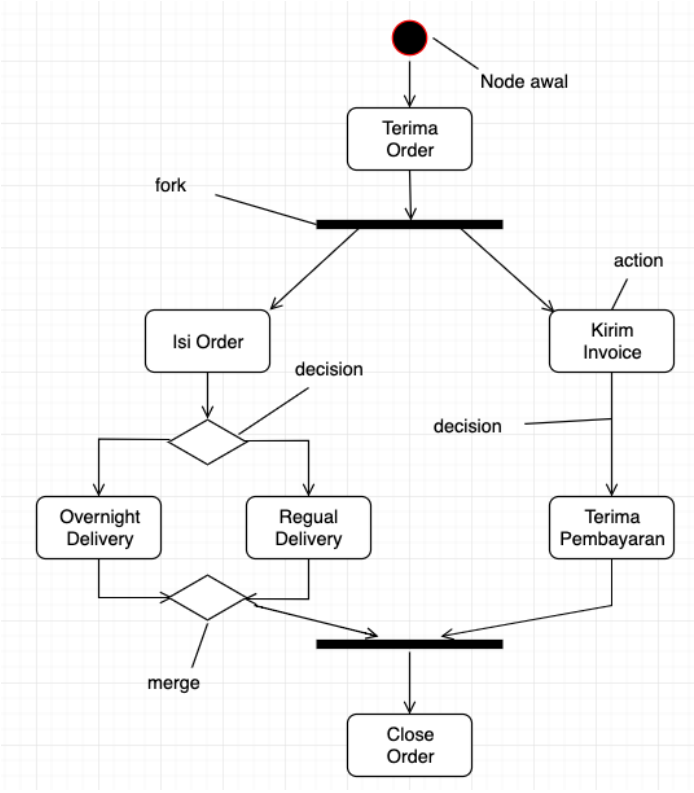


Gambar 2.8 Participant pada sebuah sequence diagram [11]

3. Activity Diagram

Activity diagram adalah bagian penting dari UML yang menggambarkan aspek dinamis dari sistem. Logika prosedural, proses bisnis dan aliran kerja suatu bisnis bisa dengan mudah dideskripsikan dalam *activity diagram*. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku pararel sedangkan *flowchart* tidak bisa. Tujuan dari *activity diagram* adalah untuk menangkap tingkah laku dinamis dari sistem dengan cara

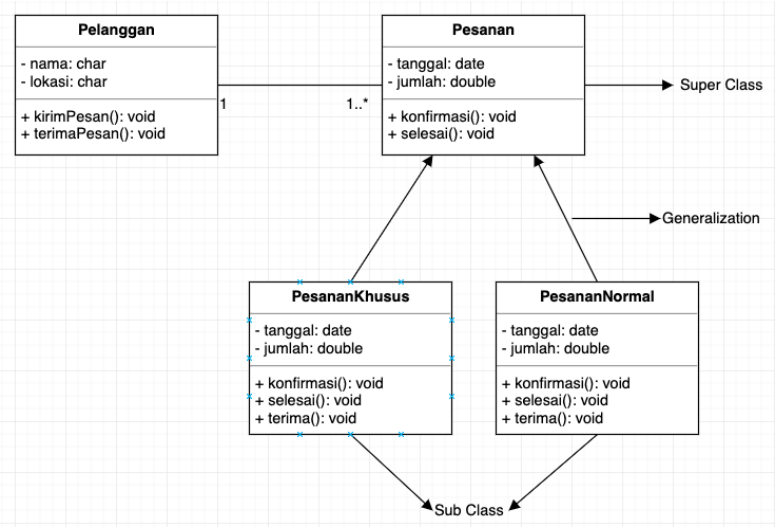
menunjukan aliran pesan dari satu aktifitas ke aktifitas lainya. Secara umum *activity diagram* digunakan untuk menggambarkan diagram alir yang terdiri dari banyak aktifitas dalam sistem dengan beberapa fungsi tambahan seperti percabangan, aliran pararel, *swim lane*, dan sebagainya. Sebelum menggambarkan sebuah *activity diagram*, perlu adanya pemahaman yang jelas tentang elemen yang akan digunakan dalam *activity diagram*. Elemen utama dalam *activity diagram* adalah aktifitas itu sendiri. Aktifitas adalah fungsi yang dilakukan oleh sistem setelah akitifitas terindentifikasi, selanjutnya yang perlu diketahui adalah bagaimana semua elemen tersebut berasosiasi dengan *constraint* dan kondisi lalu perlu penjabaran tata letak dari keseluruhan aliran agar bisa ditransofrmasikan ke *activity diagram*.



Gambar 2.9 Contoh activity diagram sederhana [11]

4. Class diagram

Class diagram adalah diagram statis. Diagram ini mewakili pandangan statis dari suatu aplikasi. *Class diagram* tidak hanya digunakan untuk memvisualisasikan, menggambarkan dan mendokumentasikan berbagai aspek sistem tetapi juga membangun kode eksekusi dari aplikasi perangkat lunak. *Class diagram* menggambarkan *atribut*, *operation* dan juga *constraint* yang terjadi pada sistem. *Class diagram* banyak digunakan dalam pemodelan sistem OO karena mereka adalah satu-satunya diagram UML yang dapat dipetakan langsung dengan bahasa berorientasi objek. *Class diagram* menunjukan koleksi *class*, antarmuka, asosiasi, kolaborasi, dan *constraint*, dikenal juga sebagai diagram structural [10].



Gambar 2.10 Contoh class diagram sederhana [11]

2. Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh *System Analyst* dalam tahap analisis persyaratan proyek pengembangan sistem. ERD merupakan alsat peraga yang memberikan dasar untuk desain *database* relasional yang mendasari sistem informasi yang dikembangkan. ERD bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk *database* [12].

Dalam *Entity Relationship Diagram* (ERD) terdapat beberapa komponen, yaitu sebagai berikut :

1. Entitas (*Entity*)
   Entitas adalah suatu objek yang memiliki hubungan dengan objek lain. Dalam ERD digambarkan dengan bentuk persegi panjang.
2. Hubungan (Relationship)
   Merupakan keterangan entitas dapat berhubungan dengan entitas lain, hubungan ini disebut dengan entity relationship yang digambarkan dengan garis. Terdapat 4 tipe relasi dalam *database* yaitu :

a. One-to-One
   Artinya satu data memiliki satu data pasangan di tabel lain. Jadi, jika dua tabel berelasi one-to-one artinya setiap *record* di entitas pertama hanya akan berhubungan dengan satu *record* di entitas kedua begitu pula sebaliknya. Conrohnya relasi antara tabel pegawai dan alamat pegawai. Satu dokumen pegawai hanya berhubungan dengan satu *record* alamat pegawai begitu pula sebaliknya.

b. One-to-Many
   Artinya satu data memiliki beberapa data pasangan di tabel lain. Jadi, misalkan terdapat relasi antara tabel satu dan tabel dua yang berarti satu dokumen pada tabel satu boleh berelasi (mempunyai) dengan banyak dokumen pada tabel dua. Namun, satu dokumen pada tabel dua hanya boleh berelasi dengan satu dokumen saja pada tabel satu.

c. Many-to-One

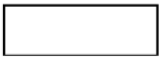
Artinya beberapa data memiliki satu data pasangan di tabel lain. Jadi, misalkan terdapat relasi antara tabel satu dengan tabel dua, dapat diartikan bahwa satu dokumen pada tabel satu hanya boleh berelasi (mempunyai) dengan satu dokumen pada tabel dua. Tetapi, satu dokumen pada tabel dua boleh berelasi dengan banyak dokumen pada tabel satu.

d. Many-to-Many
Artinya beberapa data memiliki beberapa data pasangan di tabel lain. Jadi, jika tabel satu berelasi dengan tabel dua dengan relasi many-to- many berarti bahwa ada banyak dokumen di entitas satu dan entitas dua yang saling berhubungan satu sama lain.

3. Atribut

Atribut adalah elemen dari entitas yang berfungsi sebagai deskripsi karakter entitas dan digambarkan dengan bentuk elips [12].

Tabel 2.1 Entity Relationship Diagram (ERD) [12]

No	Simbol	Keterangan
1		Entitas
2		Atribut
3		Hubungan
4		Garis

1. *Metode Pengujian*

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean [16]. Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. Test case yang baik adalah test case yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Karakteristik umum dari pengujian perangkat lunak adalah sebagai berikut :

1. Pengujian dimulai pada level modul dan bekerja keluar kearah integrasi pada sistem berbasiskan komputer.
2. Teknik pengujian yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3. Pengujian diadakan oleh software developer dan untuk proyek yang besar oleh group testing yang independent.

4. Testing dan Debugging adalah aktivitas yang berbeda tetapi debugging harus diakomodasikan pada setiap strategi testing [17].

Metode pengujian perangkat lunak ada 3 jenis, yaitu :

1. *White Box/Glass Box* - pengujian operasi.
2. *Black Box* - untuk menguji sistem.
3. *Use case* - untuk membuat input dalam perancangan black box dan pengujian *statebased* [17].

1. White Box Testing

Pengujian *white box* adalah pengujian yang meramalkan cara kerja perangkat lunak secara rinci, karenanya logikal *path* (jalur logika) perangkat lunak akan di-test dengan menyediakan test case yang akan mengerjakan kumpulan kondisi atau pengulangan secara spesifik. Secara sekilas dapat diambil kesimpulan *white box* testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

Menurut Pressman [10], pengujian *White box* atau *Glass box* adalah metode *test case* desain yang menggunakan struktur kontrol desain *procedural* untuk memperoleh *test-case*. Dengan menggunakan metode pengujian *white box*, analis sistem akan dapat memperoleh *test case* yang:

a. Memberikankan jaminan bahwa semua jalur *independent* pada suatu modul telah digunakan paling tidak satu kali.
b. Menggunakan semua keputusan logis dari sisi *true* dan *false.*
c. Mengeksekusi semua batas fungsi *loops* dan batas operasionalnya.
d. Menggunakan struktur *internal* untuk menjamin validitasnya.

Ujicoba basis *path* adalah teknik uji coba *white box* yang diusulkan Tom Mc Cabe. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunkan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. *Test case* yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

Terdapat beberapa proses yang harus di lakukan dalam uji coba basis *path* yaitu diantaranya :

1. Notasi Diagram Alir

   Sebelum metode basis *path* diperkenalkan, terlebih dahulu akan dijelaskan mengenai notasi sederhana dalam bentuk diagram alir (grafik alir). Diagram alir menggambarkan aliran kontrol logika yang menggunakan notasi.

2. Kompleksitas Siklomatis

   Kompleksitas siklomatis adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian basis *path*, maka nilai yang terhitung untuk kompleksitas siklomatis menentukan jumlah jalur independen dalam basis set suatu pemrograman memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua statemen telah dieksekusi sedikitnya satu kali. Jalur independen adalah jalur yang memlalui progarm yang mengintroduksi sedikitnya satu rangkaian statemen proses baru atau suatu kondisi baru. Bila dinyatakan dengan terminologi grafik alir, jalur independen harus bergerak sepanjang paling tidak satu *edge* yang tidak dilewatkan sebelum jalur tersebut ditentukan.

3. Melakukan Test Case

   Metode uji coba basis *path* juga dapat diterapkan pada perancangan prosedural rinci atau program sumber. Pada bagian ini akan dijelaskan langkah-langkah uji coba basis *path*.

4. Matriks Grafis

Prosedur untuk mendaptakan grafik alir dan menentukan serangkaian basis *path*, cocok dengan mekanisasi. Untuk mengembangkan peranti perangkat lunak yang membantu pengujian basis *path*, struktur data yang disebut matriks grafis dapat sangat berguna. Matriks garfis adalah matriks bujur sangkar yang ukurannya sama dengan jumlah simpul pada grafik alir. Masing-masing baris dan kolom sesuai dengan simpul yang diidentifikasikan dan *entry* matriks sesuai dengan *edge* diantara simpul.

2. Black Box Testing

Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian *black box* atau bisa disebut pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan menggunakan perangkat lunak, serta dilihat apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *black box testing* harus dibuat dengan kasus benar dan kasus salah. *Black box* testing juga disebut pengujian tingkah laku, memusat pada kebutuhan fungsional perangkat lunak. Teknik pengujian *black box* memungkinkan memperoleh serangkaian kondisi masukan yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Beberapa jenis kesalahan yang dapat diidentifikasi adalah fungsi tidak benar atau hilang, kesalahan antar muka, kesalahan pada struktur data (pengaksesan basis data), kesalahan performasi, kesalahan inisialisasi dan akhir program. *Equivalence Partitioning* merupakan metode *black box* testing yang membagi domain masukan dari program kedalam kelas-kelas sehingga *test case* dapat diperoleh. *Equivalence Partitioning* berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan, dan mengurangi jumlah kasus uji yang harus dibuat. Kasus uji yang didesain untuk *equivalence partitioning* berdasarkan pada evaluasi dari kelas ekuivalensi untuk kondisi masukan yang menggambarkan kumpulan keadaan yang valid atau tidak. Kondisi masukan dapat berupa spesifikasi nilai numerik, kisaran nilai, kumpulan nilai yang berhubungan atau kondisi boolean [16].

Pengujian *black box* berusaha menemukan kesalahan dalam kategori :

1. Fungsi – fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi [17].