# BAB 2
# LANDASAN TEORI

## 2.1 Daging

Daging merupakan salah satu bahan dari hasil produksi peternakan yang dikosumsi untuk memenuhi kebutuhan gizi seperti protein, hal ini karena dalam daging mengandung protein yang juga terdapat asam amino esensial yang lengkap didalamnya. Daging didefinisikan sebagai bagian dari hewan potong yang digunakan manusia sebagai makanan, selain memiliki penampilan yang menarik selera, juga merupakan sumber protein hewani berkualitas tinggi. Selain itu, daging juga dapat didefinisikan seluruh bagian dari ternak yang sudah dipotong dari tubuh ternak kecuali tulang, kuku, bulu, dan bagian tanduknya.

### 2.1.1 Ciri-ciri dan karakteristik daging sapi dan daging Kambing

1. Daging sapi

   Dari warnanya, daging ini berwarna merah kecoklatan, merah pucat atau merah keunguan. Namun dapat berubah warna menjadi coklat saat terlalu lama terkena udara. Memiliki serat yang halus namun tidak mudah hancur dan ada sedikit lemak berwarna kuning
2. Daging Kambing

   Memiliki warna merah muda dan berserat halus. Daging ini mengandung banyak lemak yang bertekstur keras dan kenyal yang berwarna putih kekuningan dan dagingnya memiliki bau khas yang menyengat.

### 2.1.2 Nilai Keakuratan Kesegaran Daging Sapi dan Kambing

Menganalisa jurnal sebelumnya, beberapa keakuratan nilai tegangan dari 2 sensor warna dan bau sebagai berikut.

1. Daging Sapi
   a) Daging sapi segar

   Red = 74 – 78

   Green = 25 – 27 ( Sensor Warna )

Blue = 18 – 24

Sapi Segar = 1,78 – 1,90 ( Sensor Bau/Gas )

b) Daging Sapi busuk

Red = 27 – 30

Green = 24 – 17 ( Sensor warna )

Blue = 13 – 17

Sapi Busuk = 2,09 – 2,11 ( Sensor Bau/Gas )

2. Daging Kambing

a) Daging Kambing segar

Red = 79 – 84

Green = 27 – 32 ( Sensor Warna )

Blue = 25 – 29

Kambing Segar = 1,91 – 2,09 ( Sensor Bau/Gas )

b) Daging Kambing Busuk

Red = 27 – 30

Green = 24 – 17 ( Sensor Warna )

Blue = 13 – 17

Kambing Busuk = 2,12 – 2,34 (Sensor Bau/gas)

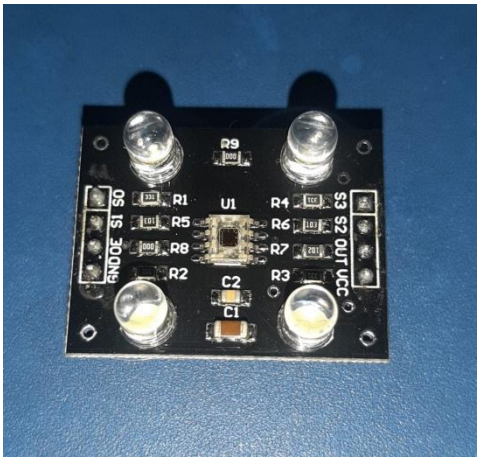
## 2.2 Sensor

Sensor adalah sesuatu yang digunakan untuk mendeteksi adanya perubahan lingkungan fisik atau kimia. Variabel keluaran dari sensor yang diubah menjadi besaran listrik disebut Transduser. Pada saat ini , sensor-sensor sudah dibuat dengan ukuran yang sangat kecil sehingga memudahkan untuk pemakaian dan lebih hemat energi[1].

### 2.2.1.1 TCS 3200

TCS3200 merupakan konverter yang diprogram untuk mengubah warna

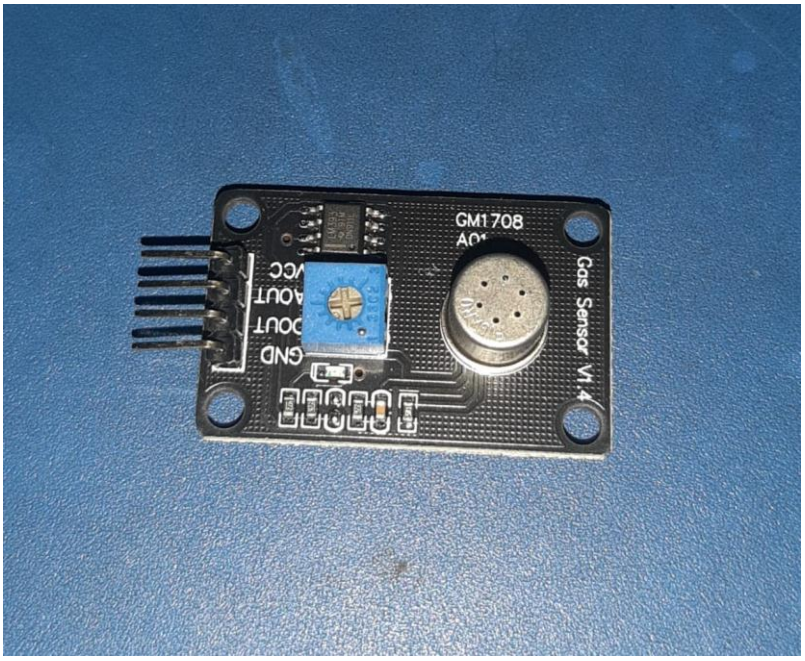
menjadi frekuensi, yang tersusun atas konfigurasi fotodiode silikon dan converter arus ke frekuensi dalam IC CMOS monolithic yang tunggal[2]. Keluaran dari sensor ini adalah gelombang kotak (duty cycle 50%) dengan frekuensi yang berbanding lurus dengan intensitas cahaya (irradiance). Masukan digital dan keluaran digital dari modul sensor ini memungkinkan antarmuka langsung ke mikrokontroler atau sirkuit logika lainnya. Di dalam TCS3200, konverter cahaya ke frekuensi membaca sebuah array fotodiode 8 × 8, 16 fotodiode mempunyai penyaring warna biru, 16 fotodiode mempunyai penyaring warna merah, 16 fotodiode mempunyai penyaring warna hijau, dan 16 fotodiode untuk warna terang tanpa penyaring. Empat tipe warna dari fotodiode diintegrasikan untuk meminimalkan efek ketidakseragaman dari insiden irradiance. Semua fotodiode dari warna yang sama terhubung secara paralel. Pin S2 dan S3 pada modul sensor digunakan untuk memilih grup dari fotodiode (merah, hijau, biru, jernih) yang aktif[3].




**Gambar 2.1 Sensor warna**

### 2.2.1.2 TGS 2602

TGS 2602 adalah Sensor gas untuk mengetahui kadar gas di luar ruang seperti amonia dan H2S yang berasal dari tempat pembuangan. Selain itu sensor juga dapat digunakan untuk memonitor VOC, memiliki sensitivitas tinggi terhadap konsentrasi rendah dari gas-gas yang tidak berbau seperti amonia dan H2 S dihasilkan dari bahan limbah di lingkungan kantor dan rumah[4].

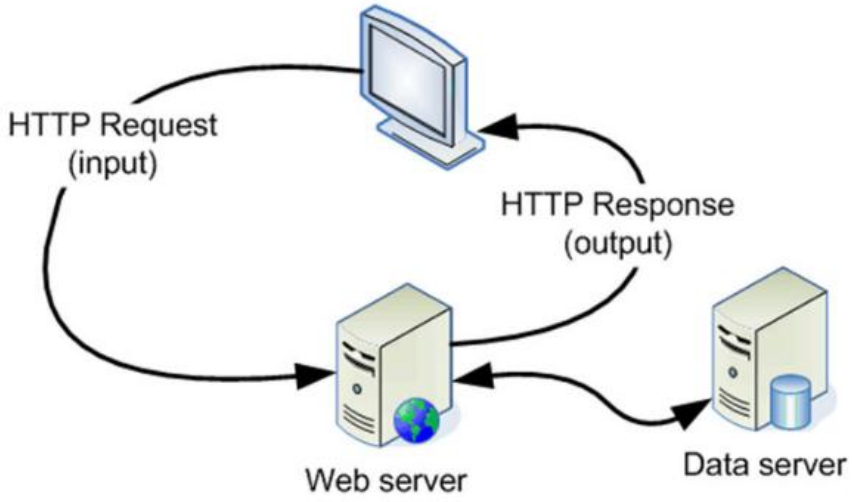



**Gambar 2.2** Gas sensor

## 2.3 Web Server

Web server adalah perangkat lunak yang berfungsi sebagai penerima permintaan yang dikirimkan melalui browser kemudian memberikan tanggapan permintaan dalam bentuk halaman situs web atau lebih umumnya dalam dokumen HTML. Namun, web server dapat mempunyai dua pengertian berbeda, yaitu sebagai bagian dari perangkat keras (hardware) maupun sebagai bagian dari perangkat lunak (software)[5].

Jika merujuk pada hardware, web server digunakan untuk menyimpan semua data seperti HTML dokumen, gambar, file CSS stylesheets, dan file JavaScript. Sedangkan pada sisi software, fungsi web server adalah sebagai pusat kontrol untuk memproses permintaan yang diterima dari browser. (Sumber : https://www.niagahoster.co.id/blog/web-server-adalah/)




**Gambar 2.3 Web server**

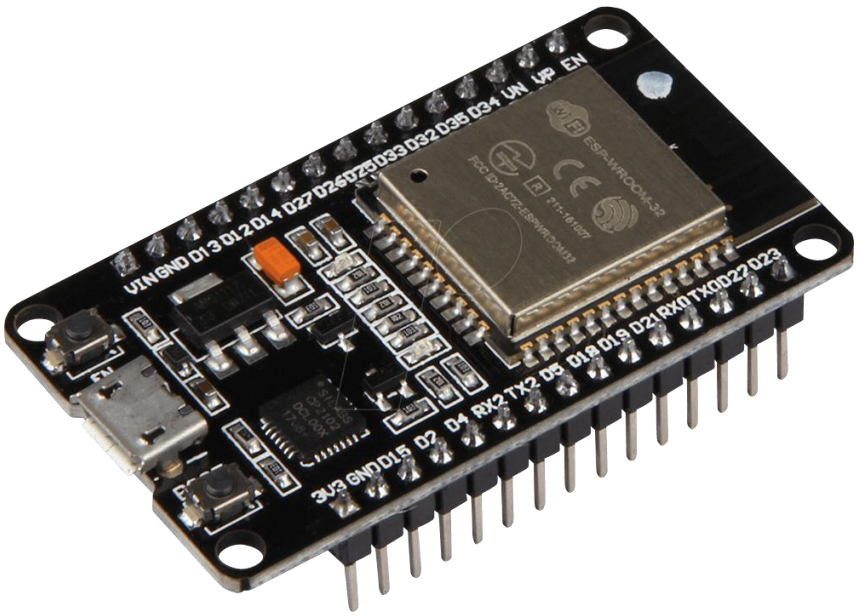
## 2.4 Akses Point

Wireless access point (WAP) yang juga dikenal sebagai access point adalah perangkat keras yang digunakan dalam jaringan area lokal nirkabel untuk mengirim dan menerima data. alur akses menghubungkan pengguna ke pengguna lain dalam jaringan dan juga berfungsi sebagai titik interkoneksi antara WLAN dan jaringan kabel tetap.

Access point terhubung langsung ke jaringan area lokal berkabel, biasanya Ethernet. Access point kemudian menyediakan koneksi nirkabel menggunakan teknologi LAN nirkabel, biasanya Wi-Fi, untuk perangkat lain yang menggunakan koneksi kabel itu. Access point mendukung koneksi beberapa perangkat nirkabel melalui satu koneksi kabelnya.

## 2.5 ESP32-01

ESP32 adalah mikrokontroler yang dikenalkan oleh Espressif System merupakan penerus dari mikrokontroler ESP8266[6]. Pada mikrokontroler ini sudah tersedia modul WiFi dalam chip sehingga sangat mendukung untuk membuat sistem aplikasi Internet of Things, ESP32 ini memiliki tegangan operasi 3.3V[7].




**Gambar 2.4** EPS32-01

(Sumber : https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)

Dengan penggunaan yang cukup sederhana, anda tinggal menghubungkan power dari USB ke PC anda atau melalui adaptor AC/DC ke jack D.

## 2.6 Konsep Perancangan Berorientasi Objek

Pendekatan perancangan berorientasi objek menganalogikan sistem aplikasi seperti kehidupan yang memiliki elemen-elemen objek didalamnya. Dalam membangun sistem yang berorientasi objek, langkah awal yang dilakukan adalah melakukan proses analisis dan perancangan yang berorientasi pada objek-objek yang terlibat didalam sistem. Tujuan dilakukannya proses tersebut adalah untuk mempermudah pengembang dalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut yang kemudian dimodelkan dalam sistem nyata. Perusahaan software Rational Software, telah membentuk konsorsium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai Bahasa standar dalam *Object Oriented Analysist Design* (OOAD).

### *2.6.1* 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.

#### 2.6.1.1 Diagram UML

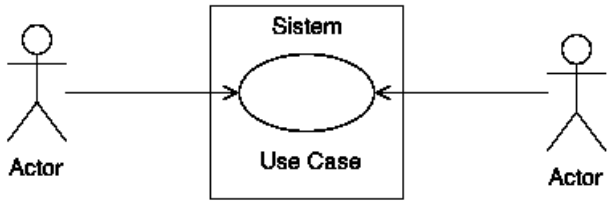
UML dalam mendokumentasikan rancangan menyediakan berbagai macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek. Namun, dalam penelitian ini hanya menggunakan empat macam diagram saja untuk memodelkannya. Empat macam diagram yang digunakan diantaranya yaitu:

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*.



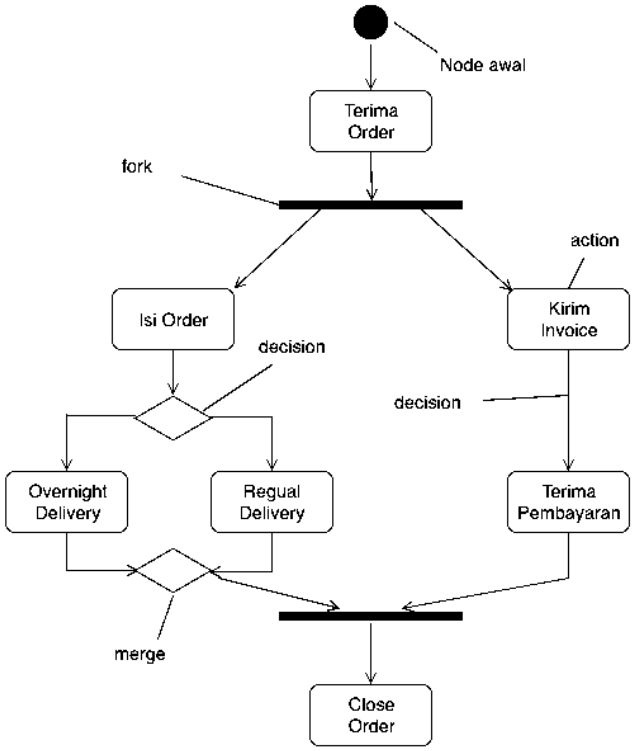

**Gambar 2.5** Use Case Model

2. 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*.

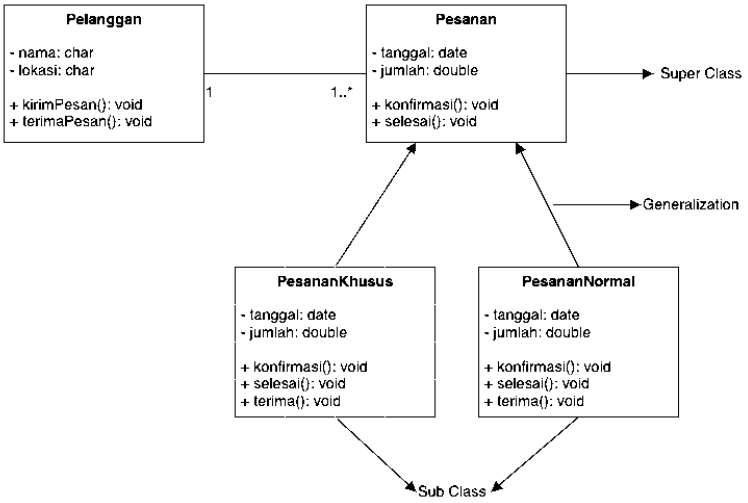



**Gambar 2.6** Contoh activity diagram sederhana

3. 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.




**Gambar 2.7** Contoh class diagram sederhana

## 2.7 Arduino IDE

Arduino IDE (Integrated Development Environment) adalah software yang di gunakan untuk memprogram di arduino, dengan kata lain Arduino IDE sebagai media untuk memprogram board Arduino. Arduino IDE bisa di download secara gratis di website resmi Arduino IDE[8].

Arduino IDE ini berguna sebagai text editor untuk membuat, mengedit, dan juga mevalidasi kode program. bisa juga digunakan untuk meng-upload ke board Arduino. Kode program yang digunakan pada Arduino disebut dengan istilah Arduino "sketch" atau disebut juga source code arduino, dengan ekstensi file source code. ino**.** (sumber : http://allgoblog.com/apa-itu-arduino-ide-dan-arduino-sketch/ ).

## 2.8 Bahasa C

C secara umum merupakan bahasa pemrograman terstruktur. Instruksi - instruksinya terdiri dari istilah ekspresi aljabar, ditambah dengan kata kunci bahasa Inggris tertentu seperti *if*, *else*, *for*, *do* dan *while*. Dalam hal ini C

menyerupai bahasa tingkat tinggi lainnya seperti pascal dan fortran. Bahasa pemrograman C juga mengandung fitur tambahan tertentu yang memungkinkannya digunakan pada level yang lebih rendah, lebih dekat ke bahasa mesin komputer. C pertama kali dikembangkan oleh Dennis Ritchie dan Brian Kerighan di Bell Telephone Laboratories Inc (sekarang AT&T Bel Laboratories) pada 1970-an. C sebagian besar dikembangkan pada Bell Labs hingga 1978, ketika Brian Kerighan dan Dennis Ritchie menerbitkan deskripsi definitif bahasa, "*The C Programming Language, Prentice-Hall*" tahun 1978. Versi C yang sesuai dengan standar ANSI dikenal sebagai ANSI C. Tetapi kebanyakan kompiler C yang kompatibel dengan ANSI juga memiliki fitur-fitur khusus yang ditambahkan, berbeda dengan yang tidak kompatibel dengan ANSI. C memiliki set instruksi yang relatif kecil, 32 kata kunci dalam versi bahasa yang paling umum, tetapi hal ini bisa menjadi komentar dengan fungsi *library* yang luas yang dapat dibangun oleh programmer. Salah satu fitur menarik dari bahasa C adalah bahwa C dapat dikompilasi untuk menghasilkan kode yang dapat dieksekusi dengan sangat cepat dan saling terhubung. Hal ini memungkinkannya digunakan untuk memprogram mikrokontroller, yang biasanya memiliki sedikit memori.

## 2.9 Metode Pengujian

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean. 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.

### 2.9.1 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, 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.

### 2.9.2 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.

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.