BAB 2

TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam sistem monitoring dan kontroling ruang perawatan ulat hongkong berbasis *internet of things*(IOT). Beberapa teori terkait dengan pembangunan sistem ini yang di dalamnya berhubungan dengan perangkat lunak, perangkat keras, dan bahasa pemrograman yang dibutuhkan dalam sistem monitoring dan kontroling ruang perawatan ulat hongkong berbasis *internet of things*(IOT).

2.2 Ulat Hongkong

Ulat hongkong merupakan larva serangga dari jenis Tenebrio molitor yang sekarang ini banyak dibudidayakan sebagai pakan burung, ikan, reptile, pangan, dan sebagai bahan baku kosmetik. Namun tidak banyak orang mengetahui bahwa Ulat hongkong juga memiliki kandungan protein yang lebih tinggi dan kandungan lemak yang lebih rendah dibandingkan dengan danging sapi dan telur ayam (Ghaly and Alkoaik, 2009). Dalam masa hidupnya, ulat hongkong melewati beberapa siklus yaitu telur, larva, kepompong (pupa), dan kepik / serangga. Tingginya kandungan protein pada ulat hongkong membuat ulat hongkong menjadi salah satu hewan yang dapat menjadi sumber protein alternatif di masa depan

Dilihat dari kandungan gizi dan manfaat yang dimiliki ulat hongkong, ulat hongkong merupakan salah satu usaha potensial untuk dikembangkan menjadi usaha peternakan rakyat. Selain karena cara budidaya yang mudah, peternakan ulat hongkong juga mempunyai peluang bisnis yang cukup menjanjikan mengingat pangsa pasar yang sangat kondusif di Indonesia. Penggunaan ulat hongkong yang semakin meluas menyebabkan permintaan ulat hongkong mengalami peningkatan. Berdasarkan hasil survei yang dilakukan, kebutuhan ulat hongkong untuk wilayah Jakarta, Bogor, Depok, Tangerang, dan Bekasi (Jabodetabek) sekitar 73 ton/bulan yang dirinci dari kebutuhan di Bogor 12 ton/bulan, Jakarta 20 ton/bulan, Bekasi 15 ton/bulan, Depok dan Tangerang yang masing-masing mencapai 13 ton/bulan (Yusdira et al. 2016)

Menurunya Produktivitas Ulat Hongkong salah satunya disebabkan oleh Suhu dan kelembaban yang yang terlalu panas atau terlalu rendah akan menyebabkan pembentukan ulat

dewasa menjadi tidak serempak sehingga ulat yang lambat berkembang akan mengalami kematian akibat diinjak-injak atau dimakan oleh ulat yang sudah berubah menjadi kepik.

Gambar 2. 1 Ulat Hongkong

2.2.1 Syarat Perawatan Ulat Hongkong

Syarat tumbuh ulat hongkong dalam budidaya ulat hongkong adalah sebagai berikut:

1. Iklim

   Suhu berpengaruh pada pertumbuhan Ulat Hongkong. Suhu yang ideal untuk budidaya Ulat Hongkong adalah 26,5-27,5 oC dengan kelembaban sekitar 75,5% Sedangkan pada musim kemarau suhu cenderung meningkat [Apriani, 2006]. Serangga sangat sensitive terhadap suhu tinggi dan menghindari tempat yang panas [Sitompul, 2006].

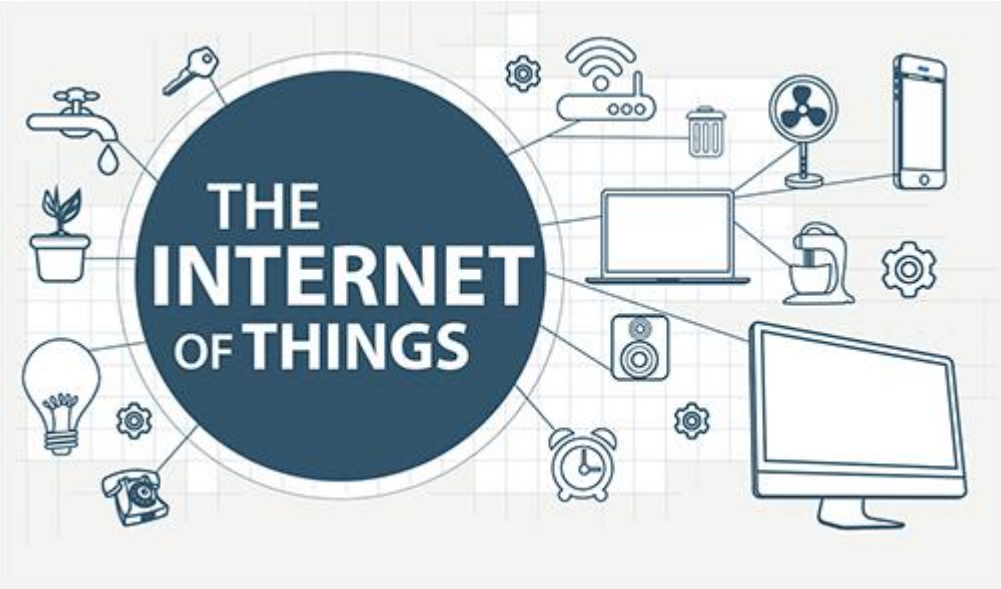
2.3 *Internet Of Things*

Internet of Things adalah sebuah teknologi yang memungkinkan kita untuk menghubungkan mesin, peralatan, dan benda fisik lainnya dengan sensor jaringan dan aktuator

untuk memperoleh data dan mengelola kinerjanya sendiri. Internet of Things mampu memperluas jangkauan teknologi informasi. *Internet of Things* (IoT) merupakan jaringan dari benda-benda yang saling terhubung satu sama lain melalui internet, dan berkomunikasi secara mandiri tanpa campur tangan manusia [Tony Haryanto, 2016].

Berdasarkan pendapat yang telah dikemukakan oleh Tony Haryanto maka dapat disimpulkan bahwa *Internet of Things* (IoT) adalah sebuah teknologi yang memungkinkan terbentuknya suatu jaringan dari benda-benda yang saling terhubung satu sama lain melalui internet dan berkomunikasi secara langsung tanpa campur tangan manusia. Dengan adanya *Internet of Things* (IoT) hal tersebut mampu memperluas jaringan dari teknologi informasi.

Teknologi *Internet of Things* sangat luar biasa. Jika sudah direalisasikan, teknologi ini tentu akan sangat memudahkan pekerjaan manusia. Manusia tidak akan perlu lagi mengatur mesin saat menggunakannya, tetapi mesin tersebut akan dapat mengatur dirinya sendiri dan berinteraksi dengan mesin lain yang dapat berkolaborasi dengannya. Hal ini membuat mesin-mesin tersebut dapat bekerja sendiri dan manusia dapat menikmati hasil kerja mesin-mesin tersebut tanpa harus repot-repot mengatur mereka.



Gambar 2. 2 Internet Of Things

2.3.1 Sejarah

istilah *Internet of Things* Pertama kali muncul yaitu pada tahun 1999. Istilah tersebut diperkenalkan oleh Kevin Ashton, seorang entrepreneur yang fokus pada teknologi asal UK. Kevin merupakan salah satu cofounder dari AutoID Center. Ia menjelaskan IoT sebagai sistem dimana benda-benda fisik terhubung ke internet melalui sensor yang ada di mana-mana [Toni Haryanto, 2016].

IOT(*Internet Of Things*) memungkinkan pengguna untuk mengelola dan mengoptimalkan elektronik dan peralatan listrik yang menggunakan internet. Hal ini berspekulasi bahwa di sebagian waktu dekat komunikasi antara komputer dan peralatan elektronik mampu bertukar informasi di antara mereka sehingga mengurangi interaksi manusia. Hal ini juga akan membuat pengguna internet semakin meningkat dengan berbagai fasilitas dan layanan internet.

Tantangan utama dalam IOT adalah menjembatani kesenjangan antara dunia fisik dan dunia informasi. Seperti bagaimana mengolah data yang diperoleh dari peralatan eletronik melakui sebuah interface antara pengguna dan peralatan itu. Sensor mengumpulkan data mentah fisik dari scenario real time dan mengkonversikan ke dalam mesin format yang dimengerti sehingga akan mudah dipertukarkan antara berbagai bentuk format data (*Thing*) [Suresh, Daniel, &Aswathy,2014].

Pada hakekatnya, benda Internet atau *Internet of Things* mengacu pada benda yang dapat di identifikasi secara unik sebagai representasi virtual dalam struktur berbasis Internet. Andaikan semua benda, makhluk maupun insan dalam kehidupan sehari-hari dapat diidentifikasi secara elektronik, maka mereka bisa dikelola dan diinventarisasi oleh komputer.

IOT muncul sebagai isu besar di Internet. diiharapkan bahwa miliaran hal fisik atau benda akan dilengkapi dengan berbagai jenis sensor terhubung ke internet melalui jaringan serta dukungan teknologi seperti tertanam sensor dan aktualisasi, jaringan sensor nirkabel, real-time dan layanan web, IOT sebenarnya cyber fisik sistem atau jaringan dari jaringan. Dengan jumlah besar hal / benda dan sensor / aktuator yang terhubung ke internet, besar-besaran dan dalam beberapa kasus aliran data real-time akan otomatis dihasilkan oleh hal-hal yang terhubung dan sensor. Dari semua kegiatan yang ada dalam IOT adalah untuk mengumpulkan data mentah yang benar dengan cara yang efisien; tapi lebih penting adalah untuk menganalisis dan mengolah data mentah menjadi informasi lebih berharga [C. Wang et al., 2013].

Perkembangan Internet of Things, semua peralatan yang kita gunakan dalam kehidupan kita sehari hari dapat dikendalikan dan dipantau menggunakan IOT. Mayoritas proses dilakukan dengan bantuan sensor di IOT. Sensor dikerahkan di mana mana dan sensor ini mengkonversi data fisik mentah menjadi sinyal digital dan mengirimkan mereka ke pusat kontrol. Dengan cara ini kita bisa memonitor perubahan lingkungan jarak jauh dari setiap bagian dari dunia melalui internet. Arsitektur sistem ini akan didasarkan pada konteks operasi dan proses dalam skenario real-time. Di otomasi rumah setiap kotak saklar listrik akan terhubung dengan ponsel pintar (atau kadang-kadang remote) sehingga itu bisa dioperasikan

dari jarak jauh. Tapi skenario seperti itu tidak perlu prosesor dan perangkat penyimpanan dipasang di setiap kotak saklar. Hanya dibutuhkan sensor untuk menangkap sinyal dan proses itu (kebanyakan beralih ON / OFF). Jadi arsitektur sistem ini bervariasi tergantung pada konteks penerapannya [Suresh et al., 2014].

2.3.2 Cara Kerja

Cara kerja dari IoT yaitu setiap benda harus memiliki sebuah alamat Internet Protocol (IP).Alamat Internet Protocol (IP) adalah sebuah identitas dalam jaringan yang membuat benda tersebut bisa diperintahkan dari benda lain dalam jaringan yang sama. Selanjutnya, alamat Internet Protocol (IP) dalam benda-benda tersebut akan dikoneksikan ke jaringan internet.

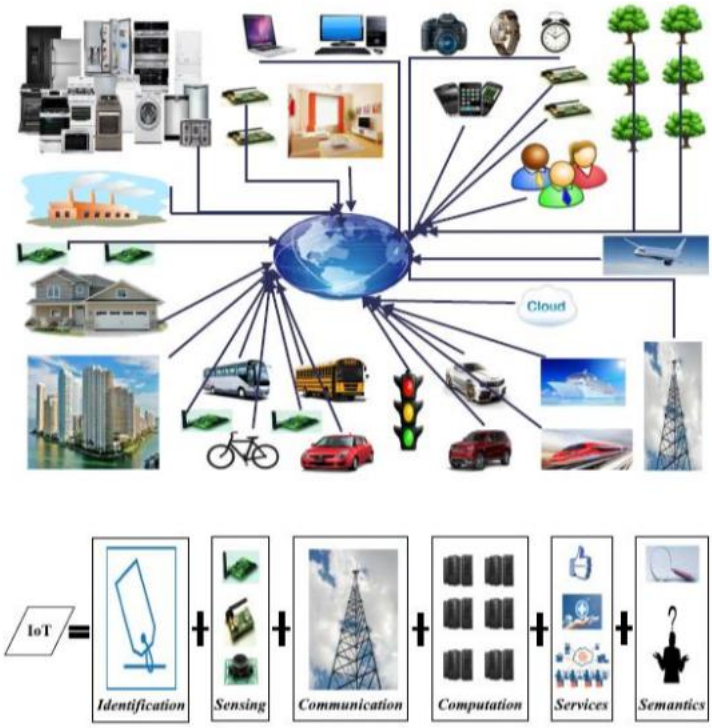
Saat ini koneksi internet sudah sangat mudah didapatkan. Dengan demikian penggunadapat memantau benda bahkan memberi perintah (*remote control*) kepada benda tersebutdengan koneksi internet. Setelah sebuah benda memiliki alamat IPdan terkoneksi dengan internet, pada benda tersebut juga dipasang sebuah sensor. Sensor pada benda memungkinkan benda tersebut memperoleh informasi yang dibutuhkan. Setelah memperoleh informasi, benda tersebut dapat mengolah informasi itu sendiri, bahkan berkomunikasi dengan benda-benda lain yang memiliki alamat IPdan terkoneksi dengan internet juga. Terjadi pertukaran informasi dalam komunikasi antara benda-benda tersebut. Setelah pengolahan informasi selesai, benda tersebut dapat bekerja dengan sendirinya, atau bahkan memerintahkan benda lain juga untuk ikut bekerja. Hal ini merupakan kelebihan dari IoT .

IoT mampu menghubungkan miliaran atau triliun benda-benda yang memiliki IP melalui internet, sehingga ada kebutuhan kritis akanarsitektur berlapis fleksibel. Semakin banyak jumlah arsitektur yang diajukan belum terkonvergensi menjadi model referensi. Sementara itu, ada beberapa proyek seperti Internet of Things (IoT-A) yang mencoba merancang arsitektur bersama berdasarkan analisis kebutuhan peneliti dan industri.

Teknologi nirkabel mewakili daerah pertumbuhan dan kepentingan yang berkembang pesat untuk menyediakan akses ke jaringan yang ada di berbagai tempat. WLAN berdasarkan standar IEEE 802.11 sedang diimplementasikan terus-menerus di rumah dan Broadband Wireless (BW) juga merupakan teknologi nirkabel yang sedang berkembang yang bersaing dengan Digital Subscriber Line(DSL).

Beberapa elemen IoT seperti RFID (*Radio Frequency Identification*), WSN (*Wireless Sensor Network*), WPAN (*Wireless Personal Area Network*), WBAN (*Wireless Body Area Network*), HAN (*Home Area Network*), NAN (*Neighborhood Area Network*), M2M (*Machine to Machine*), CC (*Cloud Computing*), dan DC (*Data Center*) memiliki pengaruh dalam

kehidupan seperti proses penginderaan IoT berarti mengumpulkan data dari benda-benda terkait di dalam jaringan dan mengirimkannya kembali ke warehouse, databaseatau cloud Elemen IoT ini merupakan bagian dari Internet Communication Technology untuk melakukanidentifikasi, penginderaan, komunikasi dan perhitungan.



Gambar 2. 3 Contoh Iot dalam kehidupan dan elemenya

2.3.3 Manfaat IOT

Pada abad ke-21, komputer pribadi dan ponsel digabungkan, menciptakan smartphonesalah satu platformpaling sukses sepanjang masa. Sekitar hampir ratusan miliar perangkat terhubung diramalkan pada tahun 2020, dimana sekitar 50 miliar akan terkait dengan IoT seperti terlihat pada Gambar 2.3.

Tugas kritis untuk mengembangkan kebijakan keamanan cyber untuk IoT memiliki urgensi tertentu karena penggabungan domain fisik dan digital di IoT bisa meningkatkan konsekuensi serangan cyber. Kekhawatiran cyber security pengguna IoT yaitu konsumen, perusahaan, atau pemerintah memerlukan kemudahan untuk mengidentifikasi dan mengeksplorasi masalah keamanan IoT. Misalnya, perusahaan dan pemerintah dapat mengidentifikasi integritas data sebagai perhatian utama, sementara konsumen mungkin paling peduli melindungi informasi pribadi. Industri dapat membangun keamanan dalam pengembangan dan implementasi perangkat IoT dan infrastruktur. Karena pengguna mengandalkan perangkat yang terhubung untuk membuat hidup lebih baik dan mudah, maka

keamanan harus diperhatikan dari setiap aspek. Semua perangkat di ekosistem IoT memiliki tanggung jawab untuk keamanan perangkat, data dan solusi. Ini berarti bahwa produsen perangkat, pengembang aplikasi, konsumen, operator, integrator dan bisnis perusahaan semuanya berperan untuk mengikuti praktik terbaik. Keamanan IoT memerlukan pendekatan berlapis-lapis. Dari sudut pandang perangkat, hal itu harus dipertimbangkan pada tingkat cetak biru yang dimulai dengan desain dan pengembangan dan membuat perangkat keras, firmware/perangkat lunak dan data menjadi aman. Pendekatan yang samaberlaku jika seorang analis keamanan atau personil operasi yang bertanggung jawab atas solusi IoT. Untuk mengaktifkan potensi penuh dari IoT, tantangan keamanan harus ditangani melalui kombinasi antara interoperabilitas dan desain yang baik dengan mengambil pendekatan proaktifakan menghasilkan produk dan solusi yang lebih baik. Block chain memainkan peran utama di IoT, dengan meningkatkan keamanan, membuat transaksi menjadi lebih mulus dan menciptakan efisiensi dalam rantai pasokan. Perusahaan mulai memanfaatkan block chain dalam tiga cara utama, yaitu membangun kepercayaan,mengurangi biaya dan mempercepat transaksi.

2.3.4 Analisis Fungsional

Analisis kebutuhan fungsional akan dimulai setelah tahap analsis terdapat sistem selesai dilakukan, analisis kebutuhan fungsional dapat didefinisikan sebagai penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi. Tahapan ini menyangkut mengkonfigurasi dari komponen-komponen perangkat lunak dan perangkat keras dari suatu sistem sehingga setelah instalasi dari sistem akan benar-benar memuaskan dari rancang bangun yang telah ditetapkan pada akhir tahap analisis sistem. Untuk menjelaskan bagaimana suatu masukan diproses pada sistem maka digunakan pada spesifikasi proses dan kamus data untuk mengetahui aliran data yang mengalir pada sistem.

2.4 Perancangan Sistem

2.4.1 UML (*Unified Modeling Language*)

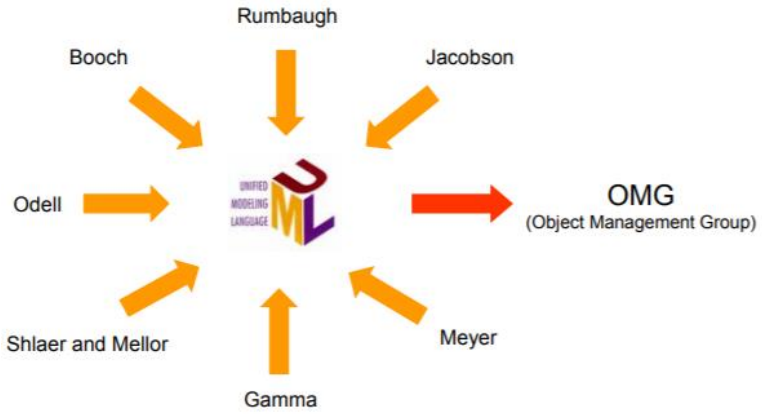
UML adalah UML merupakan singkatan dari “Unified Modelling Language” yaitu suatu metode permodelan secara visual untuk sarana perancangan sistem berorientasi objek, atau definisi UML yaitu sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan dan juga pendokumentasian sistem *software*. Saat ini UML sudah menjadi bahasa standar dalam penulisan blue print *software*.

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch [1], *metodologi coad* [2], *metodologi OOSE* [3], *metodologi OMT* [4], *metodologi shlaer-mellor* [5], *metodologi wirfs-brock* [6], dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan.



Gambar 2. 4 UML

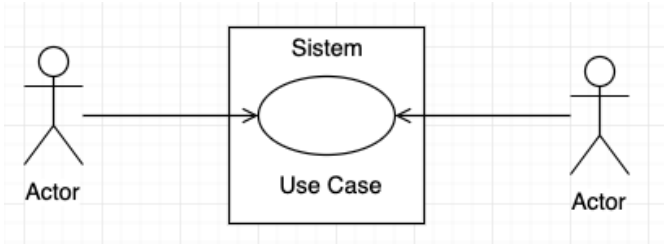
Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999 [7] [8] [9]. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

2.4.2 Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem.Use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu.Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya.

Use case diagram dapat digunakan untuk :

1. Menyusun requirement sebuah sistem
2. Mengkomunikasikan rancangan dengan klien, dan
3. Merancang test case untuk semua feature yang ada pada sistem.



Gambar 2. 5 Use case diagram

Berikut ini adalah bagian dari sebuah use case diagram :

a. Use Case

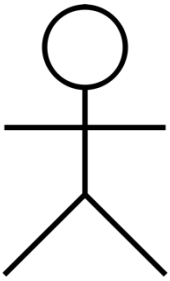
Use case adalah abstraksi dari interaksi antarasistem dengan *actor*. Oleh karena itu sangat penting untuk memilih abstraksi yang cocok. *Use case* dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ‘apa’ yang dikerjakan software aplikai, bukan ‘bagaimana’ software aplikasi mengerjakannya. Setiap *user case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case*

yang memiliki nama yang sama. Gambar 2.5 menunjukan bentuk *Use Case* dalam UML.

Gambar 2. 6 Use case

b. *Actors*

Actors adalah *abstraction* dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Orang atau sistem bisa muncukl dalam beberapa peran. Bahwa actor berinteraksi dengan *use case*, tetapi tidak memiliki control atas *use case*. Gambar 2.6 menunjukan bentuk *actor* dalam UML.



Gambar 2. 7 Actor

c. *Relationship*

Relationship adalah hubungan antara *use cases* dengan *actors*. *Relationship* dalam *use case* diagram meliputi:

a. Asosiasi antara *actor* dan *use case*.

Hubungan antara *actor* dan *use case* yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis lurus dari actor menuju *use case* baik dengan menggunakan mata panah terbuka ataupun tidak.

b. Asosiasi antara 2 *use case*

Hubungan antara *use case* yang satu dan *use case* lainnya yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis putus-putus/garis lurus dengan mata panah terbuka di ujungnya.

c. Generalisasi antara 2 *actor*

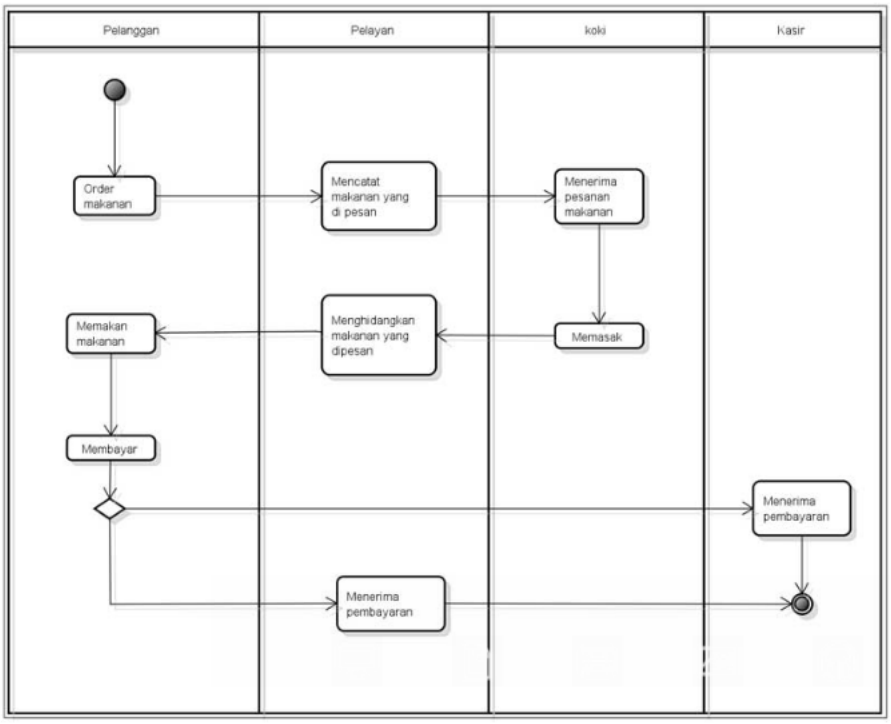
Hubungan *inheritance* (pewarisan) yang melibatkan *actor* yang satu (*the child*) dengan *actor* lainnya (*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

d. Generalisasi antara 2 *use case*.

Hubungan *inheritance* (pewarisan) yang melibatkan *use case* yang satu (*the child*) dengan use case lainnya(*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

2.4.3 Activity Diagram

Activity diagram menggambarkan aliran fungsionalitas dalam suatu sistem informasi. Secara lengkap, activity diagram mendefinisikan dimana *workflow* dimulai, dimana berhentinya, aktifitas apa yang terjadi selama *workflow*, dan bagaimana urutan kejadian aktifitas tersebut. *Activity* diagram juga menyediakan pendekatan untuk proses pemodelan paralel. Bagi mereka yang akrab dengan analisis dan desain struktur tradisional, diagram ini menggabungkan ide-ide yang mendasari diagram alir data dan diagram alur sistem.



Gambar 2. 8 Contoh activity diagram

Berikut ini merupakan komponen dalam activity diagram, yaitu :

a. *Activity node*

Activity node mengGambarkan bentuk notasi dari beberapa proses yang beroperasi dalam kontrol dan nilai data

b. *Activity edge*

Activity edge mengGambarkan bentuk *edge* yang menghubungkan aliran aksi secara langsung ,dimana menghubungkan *input* dan *output* dari aksi tersebut

c. *Initial state*

Bentuk lingkaran berisi penuh melambangkan awal dari suatu proses.

d. *Decision*

Bentuk wajib dengan suatu *flow* yang masuk beserta dua atau lebih *activity node* yang keluar. *Activity node* yang keluar ditandai untuk mengindikasikan beberapa kondisi.

e. *Fork*

Satu bar hitam dengan satu *activity node* yang masuk beserta dua atau lebih *activity node* yang keluar.

f. *Join*

Satu bar hitam dengan dua atau lebih *activity node* yang masuk beserta satu *activity node* yang keluar, tercatat pada akhir dari proses secara bersamaan. Semua *actions* yang menuju *join* harus lengkap sebelum proses dapat berlanjut.

g. *Final state*

Bentuk lingkaran berisi penuh yang berada di dalam lingkaran kosong, menunjukkan akhir dari suatu proses.

2.4.4 Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.Class menggambarkan keadaan diantaranya :

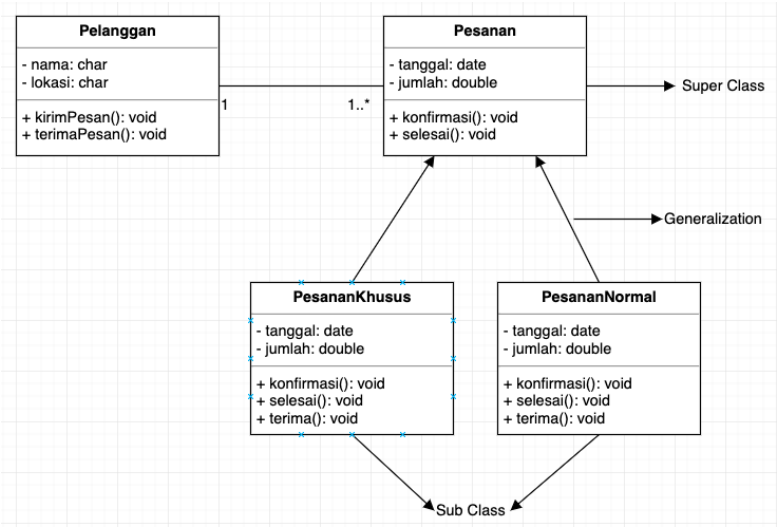
Atribut/properti suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. Private, tidak dapat dipanggil dari luar class yang bersangkutan
2. Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
3. Public, dapat dipanggil oleh siapa saja



Gambar 2. 9 Contoh class diagram

Class diagram mempunyai 3 relasi dalam penggunaanya, yaitu :

a. *Assosiation*

Assosiation adalah sebuah hubungan yang menunjukkan adanya interaksi antar *class*. Hubungan ini dapat ditunjukkan dengan garis dengan mata panah terbuka di ujungnya yang mengindikasikan adanya aliran pesan dalam satu arah.

b. *Generalization*

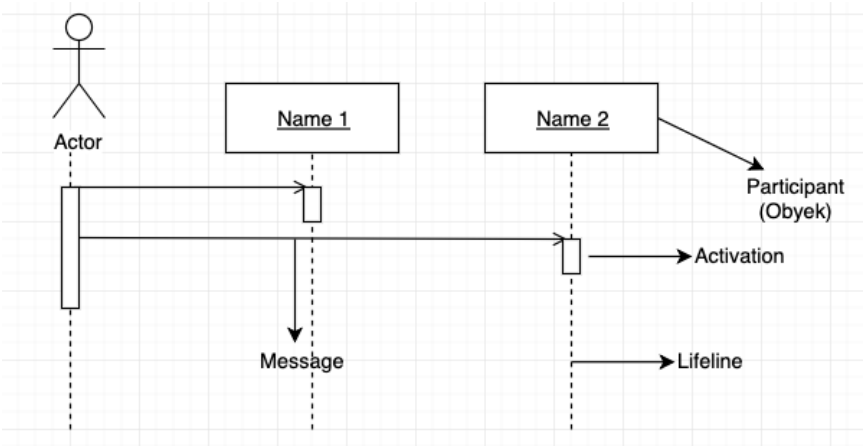
Generalization adalah sebuah hubungan antar *class* yang bersifat dari khusus ke umum

c. *Constraint*

Constraint adalah sebuah hubungan yang digunakan dalam sistem untuk memberi batasan pada sistem sehingga didapat aspek yang tidak fungsional.

2.4.5 Squence Diagram

Sequence diagram yaitu salah satu jenis diagram pada UML yang menjelaskan interaksi objek yang berdasarkan urutan waktu, sequence diagram juga dapat menggambarkan urutan atau tahapan yang harus dilakukan untuk dapat menghasilkan sesuatu seperti pada *use case* diagram.



Gambar 2. 10 Contoh sequence diagram

Berikut ini merupakan komponen dalam *sequence diagram* :

a. *Activations*

Activations menjelaskan tentang eksekusi dari fungsi yang dimiliki oleh suatu objek.

b. *Actor*

Actor menjelaskan tentang peran yang melakukan serangkaian aksi dalam suatu proses.

c. *Collaboration boundary*

Collaboration boundary menjelaskan tentang tempat untuk lingkungan percobaan dan digunakan untuk memonitor objek.

d. *Parallel vertical lines*

Parallel vertical lines menjelaskan tentang suatu garis proses yang menunjuk pada suatu *state*.

e. *Processes*

Processes menjelaskan tentang tindakan/aksi yang dilakukan oleh aktor dalam suatu waktu.

f. *Window*

Window menjelaskan tentang halaman yang sedang ditampilkan dalam suatu proses.

g. *Loop*

Loop menjelaskan tentang model logika yang berpotensi untuk diulang beberapa kali.

2.4.6 Objek Oriented

Object oriented adalah sebuah obyek memiliki keadaan sesaat (*state*) dan perilaku (*behaviour*). State sebuah obyek adalah kondisi obyek tersebut yang dinyatakan dalam *attribute*/*properties*. State sebuah obyek adalah kondisi obyek tersebut yang dinyatakan dalam *attribute*/*properties*. Sedangkan perilaku suatu obyek mendefinisikan bagaimana sebuah obyek

bertindak/beraksi dan memberikan reaksi. Perilaku sebuah objek dinyatakan dalam operation. Menurut schmuller, *attribute* dan *operation* bila disatukan akan memberikan *fitur*/*features*. Himpunan objek-objek yang sejenis disebut class. Objek adalah contoh *instance* dari sebuah class. Ada dua macam aplikasi yang tidak cocok dikembangkan dengan metode OO. Yang pertama adalah aplikasi yang sangat berorientasi ke database. Aplikasi yang sangat berorientasi ke penyimpanan dan pemanggilan data sagnat tidak cocok dikembangkan dengan OO karena akan banyak kehilangan manfaat dari penggunaan RDBMS (Relational Database Management System) untuk penyimpanan data. Akan tetapi RDBMS juga mempunyai keterbatasan dalam penyimpanan dan pemanggilan struktur data yang kompleks seperti multimedia, data spasial dan lain-lain. Bentuk aplikasi lain yang kurang cocok dengan pendekatan OO adalah aplikasi yang membutuhkan banyak algoritma. Beberapa applikasi yang melibatkan perhitungan yang besar dan kompleks. Pada penelitian ini konsep OO digunakan untuk konsep pengkodean pada sistem yang akan dibangun. Berikut ini adalah konsep-konsep dalam pemrograman berorientasi objek :

1. *Class*

Kelas (*Class*) merupakan pengGambaran satu set objek yang memiliki atribut yang sama. Kelas mirip dengan tipe data ada pemrograman non objek, akan tetapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Kelas baru dapat dibentuk lebih spesifik dari kelas ada umumnya.kelas merupakan jantung dalam pemrograman berorientasi objek.

2. *Object*

Objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan teknik yang efektif dalam menemukan cara yang tepat dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak. Orientasi objek merupakan teknik pemodelan sistem riil yang berbasis objek.

3. *Abstraction*

Kemampuan sebuah program untuk melewati aspek informasi yang diolah adalah kemampuan untuk fokus pada inti permasalahan. Setiap objek dalam sistem melayani berbagai model dari pelaku abstrak yang dapat melakukan kerja, laporan dan perubahan serta berkomunikasi dengan objek lain dalam sistem, tanpa harus menampakkan kelebihan diterapkan.

4. *Encapsulation*

Encapsulation adalah proses memastikan pengguna sebuah objek tidak dapat menggantikan keadaan dari sebuah objek dengan cara yang tidak sesuai prosedur. Artinya, hanya metode yang terdapat dalam objek tersebut yang diberi izin untuk mengakses keadaan yang diinginkan. Setiap objek mengakses *interface* yang menyabutkan bagaimana objek lainnya dapat berintegrasi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut

5. *Polimorfism*

Polimorfism merupakan suatu fungsionalitas yang diimplikasikan dengan berbagai cara yang berbeda. Pada program berorientasi objek, pembuat program dapat memiliki berbagai implementasi untuk sebagian fungsi tertentu.

6. *Inheritance*

Seperti yang sudah diuraikan di atas, objek adalah contoh / *instance* dari sebuah *class*. Hal ini mempunyai konsekuensi yang penting yaitu sebagai *instance* sebuah *class*, sebuah objek mempuyai semua karakteristik dari classnya. Inilah yang disebut dengan *Inheritance* (pewarisan sifat). Dengan demikian apapun *attribute* dan *operation* dari *class* akan dimiliki pula oleh semua obyek yang disinherit/diturunkan dari class tersebut. Sifat ini tidak hanya berlaku untuk objek terhadap *class*, akan tetapi juga berlaku untuk *class* terhadap *class* lainnya.

2.4.7 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: database management system) atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. Pada penelitian ini MySQL digunakan sebagai platform penyimpanan data dan implementasi model rancangan database yang sudah di buat. Gambar 2.11 menunjukan logo *MySQL*.

Gambar 2. 11 MySQL

MySQL memiliki banyak fitur. fitur dimiliki *MySQL* sebagai berikut :

a. *Relational Database System,* Seperti halnya software database lain yang ada di pasaran, MySQL termasuk RDBMS.

b. *Arsitektur Client-Server, MySQL* memiliki arsitektur client-server dimana *server database MySQL* terinstal di *server*. Client *MySQL* dapat berada di komputer yang sama dengan *server*, dan dapat juga di komputer lain yang berkomunikasi dengan server melalui jaringan bahkan *internet.*

c. *Mengenal perintah SQL standar, SQL (Structured Query Language)* merupakan suatu bahasa standar yang berlaku di hampir semua software database. *MySQL* mendukung *SQL* versi *SQL:2003*.

d. *Performace Tuning, MySQL* mempunyai kecepatan yang cukup baik dalam menangani *query-query* sederhana, serta mampu memproses lebih banyak *SQL* per satuan waktu.

e. *Sub Select*.

f. *Views*.

g. *Stored Prosedured* (SP).

h. *Triggers.*

i. *Replication*.

j. *Foreign Key*.

k. Security yang baik.

2.4.8 Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional dilakukan untuk menghasilkan spesifikasi kebutuhan non fungsional. Analisis ini diperlukan untuk menentukan keluaran yang akan dihasilkan oleh sistem, masukan yang diperlukan sistem, lingkup proses yang digunakan untuk mengolah masukan menjadi keluaran, volume data yang akan ditangani sistem, jumlah pemakai serta kontrol terhadap sistem.

2.4.9 Mikrokontroler

Mikrokontroler adalah suatu chip berupa IC (*Integrated Circuit*) yang dapat menerima sinyal input, mengolahnya dan memberikan sinyal output sesuai dengan program yang diisikan ke dalamnya. Sinyal input *mikrokontroler* berasal dari sensor yang merupakan informasi dari lingkungan sedangkan sinyal output ditujukan kepada aktuator yang dapat memberikan efek ke lingkungan. Jadi secara sederhana mikrokontroler dapat diibaratkan sebagai otak dari suatu perangkat/produk yang mempu berinteraksi dengan lingkungan sekitarnya.

Mikrokontroler pada dasarnya adalah komputer dalam satu chip, yang di dalamnya terdapat mikroprosesor, memori, jalur Input/Output (I/O) dan perangkat pelengkap lainnya.

Kecepatan pengolahan data pada mikrokontroler lebih rendah jika dibandingkan dengan PC. Pada PC kecepatan mikroprosesor yang digunakan saat ini telah mencapai orde GHz, sedangkan kecepatan operasi mikrokontroler pada umumnya berkisar antara 1 – 16 MHz. Begitu juga kapasitas RAM dan ROM pada PC yang bisa mencapai orde Gbyte, dibandingkan dengan mikrokontroler yang hanya berkisar pada orde byte/Kbyte.

Meskipun kecepatan pengolahan data dan kapasitas memori pada mikrokontroler jauh lebih kecil jika dibandingkan dengan komputer personal, namun kemampuan mikrokontroler sudah cukup untuk dapat digunakan pada banyak aplikasi terutama karena ukurannya yang kompak. Mikrokontroler sering digunakan pada sistem yang tidak terlalu kompleks dan tidak memerlukan kemampuan komputasi yang tinggi.

Sistem yang menggunakan mikrokontroler sering disebut sebagai embedded system atau dedicated system. Embeded system adalah sistem pengendali yang tertanam pada suatu produk, sedangkan dedicated system adalah sistem pengendali yang dimaksudkan hanya untuk suatu fungsi tertentu. Sebagai contoh, printer adalah suatu embedded system karena di dalamnya terdapat mikrokontroler sebagai pengendali dan juga dedicated system karena fungsi pengendali tersebut berfungsi hanya untuk menerima data dan mencetaknya. Hal ini berbeda dengan suatu PC yang dapat digunakan untuk berbagai macam keperluan, sehingga mikroprosesor pada PC sering disebut sebagai general purpose microprocessor (mikroprosesor serba guna). Pada PC berbagai macam software yang disimpan pada media penyimpanan dapat dijalankan, tidak seperti mikrokontroler hanya terdapat satu software aplikasi. [elektronika dasar, 2010]

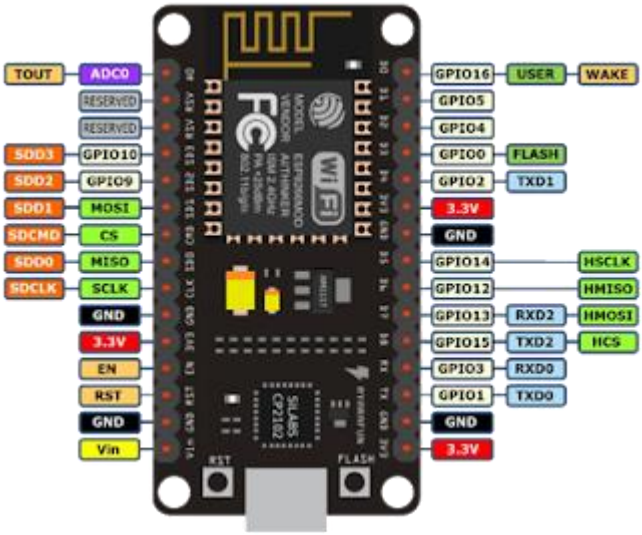
2.4.9.1 NodeMCU ESP8266

NodeMCU merupakan sebuah open source platform IoT dan pengembangan kit yang menggunakan bahasa pemrograman Lua untuk membantu dalam membuat prototype produk IoT atau bisa dengan memakai sketch dengan adruino IDE. Pengembangan kit ini didasarkan pada modul ESP8266, yang mengintegrasikan GPIO, PWM (Pulse Width Modulation), IIC, 1-Wire dan ADC (Analog to Digital Converter) semua dalam satu board. GPIO NodeMCU ESP8266.

Gambar 2. 12 Nodemcu Esp 8266

NodeMCU bisa dianalogikaan sebagai board arduino yang terkoneksi dengan ESP8622. NodeMCU telah me-package ESP8266 ke dalam sebuah board yang sudah terintergrasi dengan berbagai feature selayaknya microkontroler dan kapalitas ases terhadap wifi dan juga chip komunikasi yang berupa USB to serial. Sehingga dalam pemograman hanya dibutuhkan kabel data USB. Spesifikasi yang dimliki oleh NodeMCU sebagai berikut :

1. Board ini berbasis ESP8266 serial WiFi SoC (Single on Chip) dengan onboard USB to TTL. Wireless yang digunakan adalah IEE 802.11b/g/n. 2.2
2. tantalum capasitor 100 micro farad dan 10 micro farad.
3. 3.3v LDO regulator.
4. Blue led sebagai indikator.
5. Cp2102 usb to UART bridge.
6. Tombol reset, port usb, dan tombol flash.
7. Terdapat 9 GPIO yang di dalamnya ada 3 pin PWM, 1 x ADC Channel, dan pin RX TX
8. pin ground.
9. S3 dan S2 sebagai pin GPIO
10. S1 MOSI (Master Output Slave Input) yaitu jalur data dari master dan masuk ke dalam slave, sc cmd/sc.
11. S0 MISO (Master Input Slave Input) yaitu jalur data keluar dari slave dan masuk ke dalam master. 12. SK yang merupakan SCLK dari master ke slave yang berfungsi sebagai clock.
12. Pin Vin sebagai masukan tegangan.
13. Built in 32-bit MCU



Gambar 2. 13 Board Nodemcu

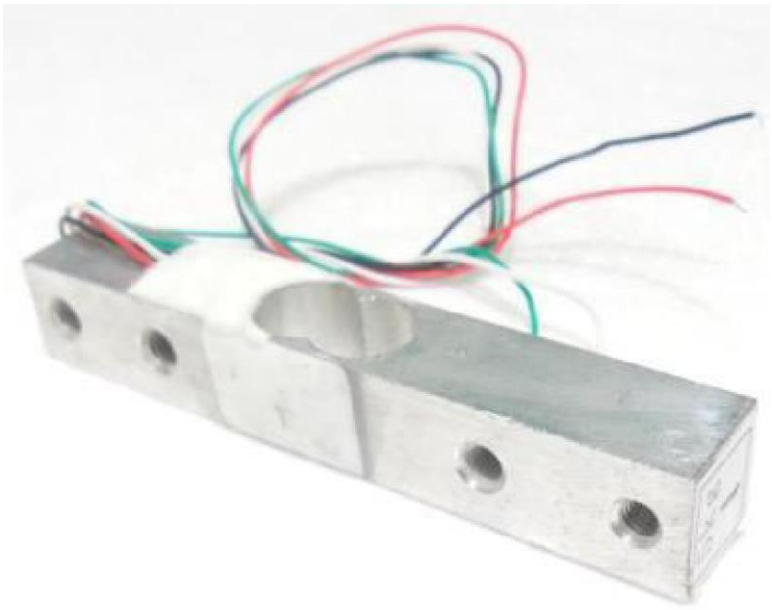
Fungsi part pada nodeMCU

1. RST : berfungsi mereset modul
2. ADC: Analog Digital Converter. Rentang tegangan masukan 0-1v, dengan skup nilai digital 0-1024 3. EN: Chip Enable, Active High
3. IO16 :GPIO16, dapat digunakan untuk membangunkan chipset dari mode deep sleep
4. IO14 : GPIO14; HSPI_CLK
5. IO12 : GPIO12: HSPI_MISO
6. IO13: GPIO13; HSPI_MOSI; UART0_CTS 5
7. VCC: Catu daya 3.3V (VDD)
8. CS0 :Chip selection
9. MISO : Slave output, Main input
10. IO9 : GPIO9
11. IO10 GBIO10
12. MOSI: Main output slave input
13. SCLK: Clock
14. GND: Ground
15. IO15: GPIO15; MTDO; HSPICS; UART0_RTS
16. IO2 : GPIO2;UART1_TXD
17. IO0 : GPIO0
18. IO4 : GPIO4
19. IO5 : GPIO5

20. RXD : UART0_RXD; GPIO3
21. TXD : UART0_TXD; GPIO1

2.4.9.2 Sensor Berat (Load Cell)

Sensor *load cell* merupakan sensor yang dirancang untuk mendeteksi tekanan atau berat sebuah beban, sensor load cell umumnya digunakan sebagai komponen utama pada sistem timbangan digital dan dapat diaplikasikan pada jembatan timbangan yang berfungsi untuk menimbang berat dari truk pengangkut bahan baku, pengukuran yang dilakukan oleh Load Cell menggunakan prinsip tekanan. (www.ricelake.com Load Cell and Weight (AmericaModule H : 2010)



Gambar 2. 14 Load Cell

Keterangan Gambar :

1. Kabel merah adalah input tegangan sensor
2. Kabel hitam adalah input ground sensor
3. Kabel hijau adalah output positif sensor
4. Kabel putih adalah output ground sensor

Sensor *load cell* memiliki spesifikasi kerja sebagai berikut :

1. Kapasitas 2 Kg
2. Bekerja pada tegangan rendah 5 –10 VDC atau 5-10 VAC
3. Ukuran sensor kecil dan praktis
4. Input atau output resistansi rendah 3

5. Nonlineritas 0.05%
6. Range temperatur kerja -10°C - +50°C

2.4.9.2.1 Karakteristik Sensor Berat (*Load Cell*)

Mekanik	
Bahan Dasar	*Alumunium Alloy*
Load Cell Type	*Strain Gauge*
Kapasitas	2kg
Dimensi	55.25x12.7x12.7mm
Lubang Pemasangan	M5 (ukuran baut)
Panjang Kabel	550mm
Ukuran Kabel	30 AWG (0.2mm)
No. Urutan Kabel	4

Tabel 2. 1 Mekanik load cell

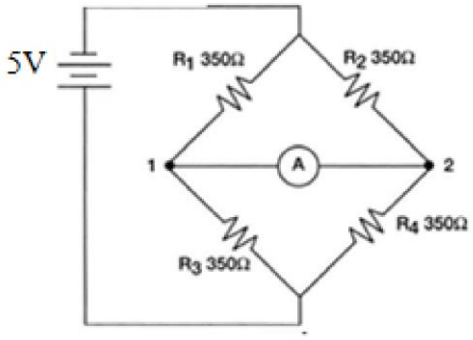
Elektrik	
Presisi	0.05%
Rata – Rata Output	1.0±0.15mv/V
Non-Linieritas	0.05% FS
Hysteresis	0.05% FS
Non-Pengulangan	0.05% FS
Creep (per 30 menit)	0.1% FS
Efek Temperatur Pada Nol (per 10°C)	0.05% FS
Efek Temperatur Pada *Span* (per 10°C)	0.05% FS
Keseimbangan Nol	±1.5% FS

Input Impedansi	1130±10 Ohm
Output Impedansi	1000±10 Ohm
Hambatan Isolasi (dibawah 50VDC)	≥5000 MOhm
Kebutuhan Voltase	5 VDC
Toleransi Jarak Temperatur	-10 to ~ +40°C
Pengoperasian Jarak Temperatur	-20 to ~ +55°C
Safe Overload	120% Kapasitas
Ultimate Overload	150% Kapasitas

Tabel 2. 2 Elektrik load cell

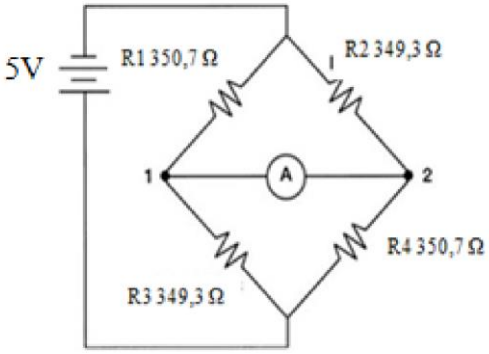
2.4.9.2.2 Prinsip Kerja Sensor Berat (*Load Cell*)

Selama proses penimbangan akan mengakibatkan reaksi terhadap elemen logam pada *load cell* yang mengakibatkan gaya secara elastis. Gaya yang ditimbulkan oleh regangan ini dikonversikan kedalam sinyal elektrik oleh *strain gauge* (pengukur regangan) yang terpasang pada *load cell.* Prinsip kerja load cell berdasarkan rangkaian Jembatan *WheatstoneI*



Gambar 2. 15 Pengukuran Wheatstonel

Pada gambar 2.15 nilai R = 350 Ω, arus yang mengalir pada R1 dan R3 = arus yang mengalir di R2 dan R4, hal ini dikarenakan nilai semua resistor sama dan tidak ada perbedaan tegangan antara titik 1 dan 2, oleh karena itu rangkaian ini dikatakan seimbang.



Gambar 2. 16 Pengukuran Wheatstonel

Jika rangkaian jembatan *Wheatstone* diberi beban, maka nilai R pada rangkaian akan berubah, nilai R1 = R4 dan R2 = R3. Sehingga membuat sensor *load cell* tidak dalam kondisi yang seimbang dan membuat beda potensial. Beda potensial inilah ynag menjadi outputnya. Untuk menghitung Vout atau A sperti pada gambar 2.16, maka rumus yang digunakan adalah sebagai berikut :

$$Vo = \left(Vs\ x \left(\frac{R1}{R1+R4}\right)\right) - \left(Vs\ x\ \left(\frac{R2}{R2+R3}\right)\right)$$

$$Vo = \left(10\ x \left(\frac{349,3}{349,3+350,7}\right)\right) - \left(10\ x\ \left(\frac{350,7}{350,7+349,3}\right)\right)$$

$$Vo = (10\ x(0,499)) - (10\ x\ (0,501))$$

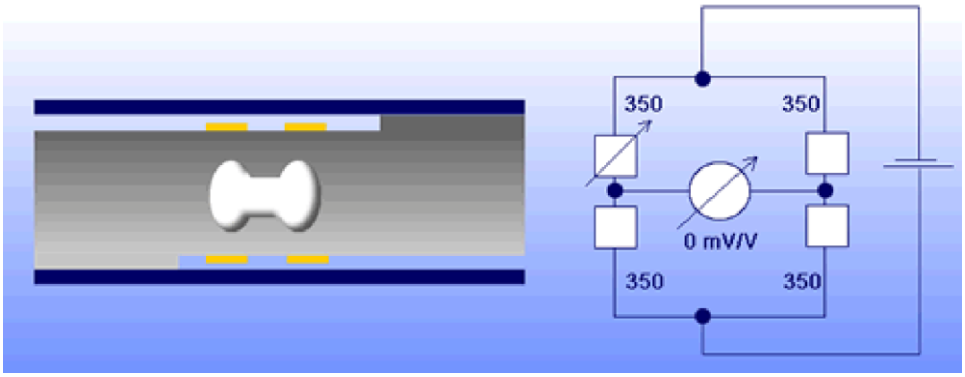
$$Vo = 4,99\text{-}5,01$$

$$Vo = -0,02 \text{ x } 10 = 2 \text{ mV}$$

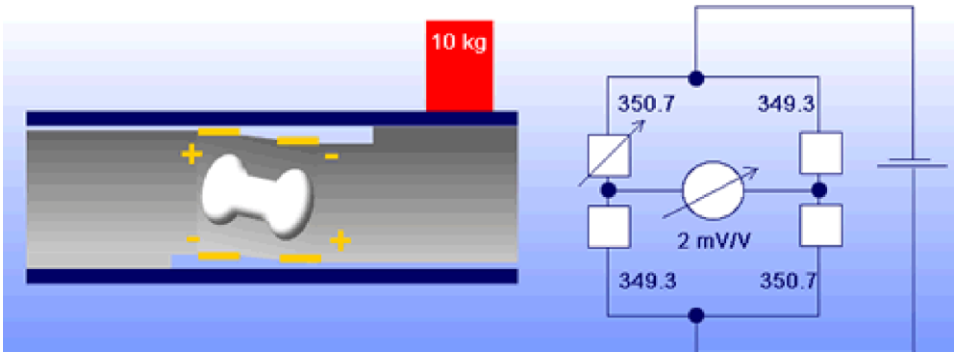
Secara teori, prinsip kerja *load cell* berdasarkan pada jembatan *Wheatstone*

dimana saat *load cell* diberi beban terjadi perubahan pada nilai resistansi, nilai resistansi R1 dan R3 akan turun sedangkan nilai resistansi R2 dan R4 akan naik.

Ketika posisi setimbang, Vout *load cell* = 0 volt, namun ketika nilai resistansi R1 dan R3 naik maka akan terjadi perubahan Vout pada *load cell*. Pada *load cell* output data (+) dipengaruhi oleh perubahan resistansi pada R1, sedangkan output (-) dipengaruhi oleh perubahan resistansi R3. (Rebby Fudi Alexander.2013. *Aplikasi Sensor Berat Load Cell Pada Alat Pengering Herbal)*



Gambar 2. 17 Rangkaian load cell tanpa beban



Gambar 2. 18 Rangkaian load cell diberi beban

2.4.9.3 Modul Penguat HX711

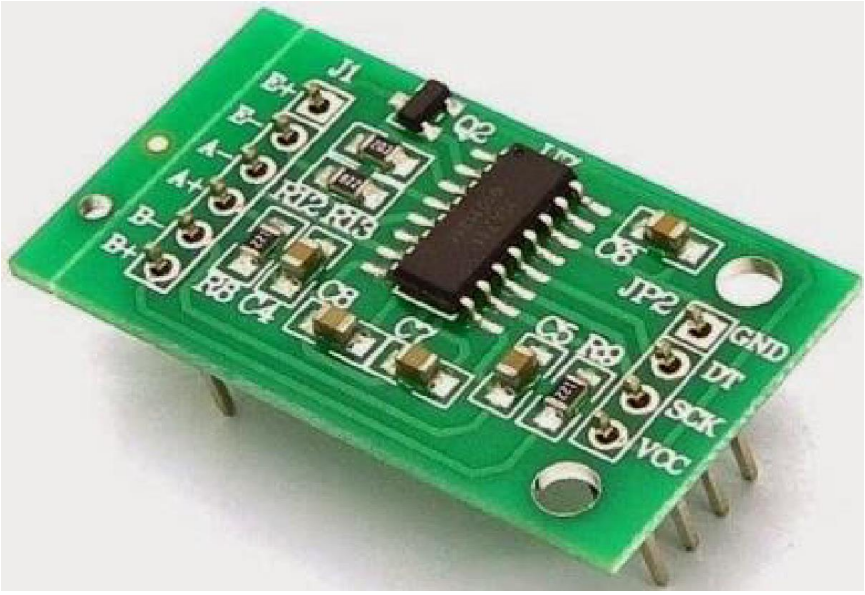
HX711 adalah sebuah komponen terintegrasi dari “AVIA SEMICONDUCTOR”, HX711 presisi 24-bit *analog to digital conventer* (ADC) yang didesain untuk sensor timbangan digital dal industrial control aplikasi yang terkoneksi sensor jembatan.

HX711 adalah modul timbangan, yang memiliki prinsip kerja mengkonversi perubahan yang terukur dalam perubahan resistansi dan mengkonversinya ke dalam besaran tegangan melalui rangkaian yang ada. Modul melakukan komunikasi dengan computer/mikrokontroller melalui TTL232. Struktur yang sederhana, mudah dalam

penggunaan, hasil yang stabil dan reliable, memiliki sensitivitas tinggi, dan mampu mengukur perubahan dengan cepat.

Sensor load cell memiliki spesifikasi kerja sebagai berikut : HX711 biasanya digunakan pada bidang aerospace, mekanik, elektrik, kimia, konstruksi, farmasi dan lainnya, digunakan untuk mengukur gaya, gaya tekanan,perpindahan, gaya tarikan, torsi, dan percepatan. Spesifikasinya adalah sebagai dibawah berikut :

1. Differential input voltage: ±40mV(Full-scale differential input voltage ± 40mV)
2. Data accuracy: 24 bit (24 bit A / D converter chip.)
3. Operating Voltage : 5V DC
4. Operating current : <10 mA
5. Size:38mm*21mm*10mm



Gambar 2. 19 Modul Hx711

Untuk rumus perhitungan konversi input analog ke digital yang berbentuk heksadesimal dapat digunakan rumus sebagai berikut :

$$Out = \frac{input-(-40)}{80} \; x \; 2^{24}$$

Contoh :

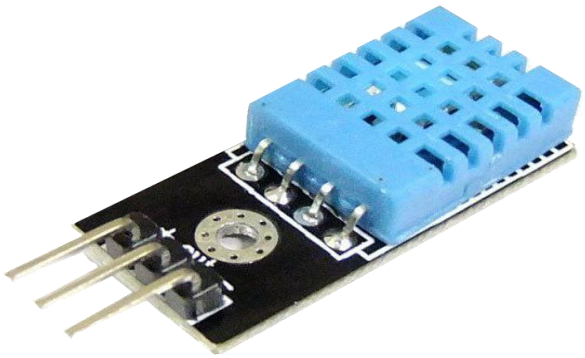
$Out = 0{,}3-(-40)80 \; x \; 16777216$

Out = 8451522 heksadesimal

Bilangan heksadesimal diatas lah yang kemudian yang dapat diolah mikrokontroler yang kemudian dikonversikan kembali menjadi satuan berat.

2.4.9.4 Sensor DHT11

Sensor Suhu & Kelembaban DHT11 dilengkapi dengan sensor suhu & kelembaban kompleks dengan output sinyal digital yang dikalibrasi. Dengan menggunakan akuisisi sinyal digital eksklusif teknik dan teknologi penginderaan suhu & kelembaban, ini memastikan keandalan yang tinggi dan stabilitas jangka panjang yang sangat baik. Sensor ini mencakup pengukuran kelembaban tipe resistif komponen dan komponen pengukuran suhu NTC, dan terhubung ke kinerja tinggi Mikrokontroler 8-bit, menawarkan kualitas luar biasa, respons cepat, anti-interferensi kemampuan dan efektivitas biaya.



Gambar 2. 20 Sensor DHT 11

Nama	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT 11	20-80%RH 0-50 ℃	±5％RH	±2℃	1	4 Pin Single Row

Tabel 2. 3 Spec DHT 11

2.4.9.4.1 Komunikasi DHT 11

Komunikasi dan sinyal Data bus tunggal digunakan untuk komunikasi antara MCU dan DHT22, dengan waktu 5mS untuk satu kalikomunikasi.Data terdiri dari bagian integral dan desimal, berikut ini adalah rumus untuk data.

DATA = 16 bit data RH + 16 bit Data suhu + 8 bit check-sum

MCU telah menerima data 40 bit dari AM2302:

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110

16bit data RH 16bit data T 8bit Jumlah cek

di sini pengubahan16 bit data RH dari sistem biner ke sistem desimal,

0000 0010 1000 1100 → 652

Sistem biner Sistem desimal

RH = 652/10 = 65,2% RH

di sini pengubahan16 bit data T dari sistem biner ke sistem desimal,

0000 0001 0101 1111 → 351

Sistem biner Sistem desimal

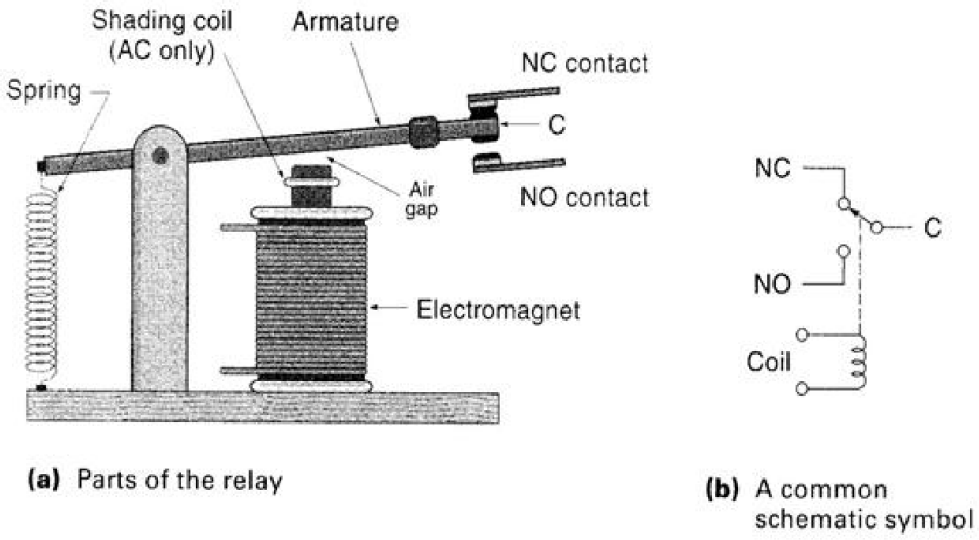
T = 351/10 = 35,1 ℃

Sum = 0000 0010 + 1000 1100 + 0000 0001 + 0101 1111 = 1110 1110

Check-sum = 8 bit terakhir dari Sum = 1110 1110^2

2.4.9.5 Relay

Relay adalah Saklar (Switch) yang dioperasikan secara listrik dan merupakan komponen Elektromekanikal (Electromechanical) yang terdiri dari 2 bagian utama yakni Elektromagnet (Coil) dan Mekanikal (seperangkat Kontak Saklar/Switch). Relay menggunakan Prinsip Elektro magnetik untuk menggerakkan Kontak Saklar sehingga dengan arus listrik yang kecil (low power) dapat menghantarkan listrik yang bertegangan lebih tinggi. Sebagai contoh, dengan Relay yang menggunakan Elektromagnet 5V dan 50 mA mampu menggerakan Armature Relay (yang berfungsi sebagai saklarnya) untuk menghantarkan listrik 220V 2A. coil adalah gulungan kawat yang mendapat arus listrik, sedangkan contact adalah sejenis saklar yang pergerakannya tergantung dari ada tidaknya arus listrik di coil. Contact ada 2 jenis : Normally Open (kondisi awal sebelum diaktifkan open), dan Normally Closed (kondisi awal sebelum diaktifkan close). Secara sederhana berikut ini prinsip kerja dari relay : ketika Coil mendapat energi listrik (energized), akan timbul gaya elektromagnet yang akan menarik armature yang berpegas, dan contact akan menutup.



Gambar 2. 21 Skema relay elektromekanik

Keterangan Gambar 2.21 :

1. Rangkaian listrik (hardware)
2. Program (software)
3. Beberapa fungsi Relay yang telah umum diaplikasikan kedalam peralatan Elektronika diantaranya adalah :
4. Relay digunakan untuk menjalankan Fungsi Logika (Logic Function)
5. Relay digunakan untuk memberikan Fungsi penundaan waktu (Time Delay Function)
6. Relay digunakan untuk mengendalikan Sirkuit Tegangan tinggi dengan bantuan dari Signal Tegangan rendah.
7. Ada juga Relay yang berfungsi untuk melindungi Motor ataupun komponen lainnya dari kelebihan Tegangan ataupun hubung singkat (Short).

2.4.9.6 Exhaust Fan

Exhaust berfungsi untuk menghisap udara di dalam ruang untuk dibuang ke luar, dan pada saat bersamaan menarik udara segar di luar ke dalam ruangan. Selain itu exhaust juga bisa mengatur volume udara yang akan disirkulasikan pada ruang. Supaya sehat setiap ruang butuh sirkulasi udara berbeda sesuai dengan fungsinya.Misalnya, ruang tidur butuh pergantian udara 2 – 4 kali per jam, kamar mandi 6 – 10 kali, dan dapur 10 – 15 kali.Untuk ruangan ber-AC, exhaust fan adalah pasangan yang saling melengkapi.

2.5 Bahasa Pemrograman

2.5.1 Python

Python merupakan bahasa pemograman tingkat tinggi (high level language) yang dikembangkan oleh Gido van Rossum pada tahun 1989 dan diperkenalkan untuk pertama kalinya pada tahun 1991. Python lahir atas dasar keinginan untuk mempermudah programer dalam menyelesaikan tugas-tugasnya dengan cepat. Python dirancang untuk memberikan kemudahan yang sangat luar biasa kepada programer baik dari segi efisiensin waktu, maupun kemudahan dalam pengembangan program dan dalam hal kompatibilitas dengan sistem. Python bisa digunakan untuk membuat program standalone dan pemrograman script (scripting programming).

Phyron adalah bahasa pemrograman interpretatif multi guna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. *Python* di klaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python mendukung multi paradigma pemrograman, utamanya namun tidak dibatasi pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan diberbagai platform sistem operasi.

Saat ini kode python dapat dijalankan diberbagai platform sistem operasi, beberapa diantaranya adalah: Linux/Unix, Windows, MacOSX, Java Virtual Machine, OS/2, Amiga, Palm, Symbian (untuk produk-produk Nokia).

2.6 Metode Pengujian

Pengujian perangkat lunak merupakan proses eksekusi program atau perangkat lunak dengan tujuan mencari kesalahan atau kelemahan dari program tersebut. Proses tersebut dilakukan dengan mengevaluasi atribut dan kemampuan program. Suatu program yang diuji akan dievaluasi apakah keluaran atau output yang dihasilkan telah sesuai dengan yang diinginkan atau tidak. Ada berbagai macam metode pengujian, teknik black box dan teknik

white box merupakan metode pengujian yang telah dikenal dan banyak digunakan oleh pengembang perangkat lunak.

2.6.1 Black Box Testing

Black box testing merupakan metode pengujian dengan pendekatan yang mengasumsikan sebuah sistem perangkat lunak atau program sebagai sebuah kotak hitam (*black box*). Metode pengujian *black box* merupakan metode pengujian dengan pendekatan yang mengasumsikan sebuah sistem perangkat lunak atau program sebagai sebuah kotak hitam (*black box*). Pendekatan ini hanya mengevaluasi program dari output atau hasil akhir yang dikeluarkan oleh program tersebut. Struktur program dan kode-kode yang ada di dalamnya tidak termasuk dalam pengujian ini. Keuntungan dari metode pengujian ini adalah mudah dan sederhana. Namun, pengujian dengan metode ini tidak dapat mendeteksi kekurangefektifan pengkodean dalam suatu program. Ciri-ciri *black box* testing adalah sebagai berikut:

1. *Black box* testing berfokus pada kebutuhan fungsional pada *software,* berdasarkan pada spesifikasi kebutuhan dari *software*.

2. Merupakan pendekatan pelengkap dalam mencangkup error dengan kelas yang berbeda dari metode *white box testing*.

3. Melakukan pengujian tanpa pengetahuan detil struktur internal dari sistem atau komonen yang dites. Juga disebut sebagai *behavioural testing, specification-based testing, input/ouput testing* atau functional testing.

4. Terdapat jenis test yang dapat dipilih berdasarkan pada tipe testing yang digunakan.

5. Kategori *error* yang akan diketahuai melalui *black box testing* seperti fungsi yang hilang atau tidak benar, *error* dari antar-muka, *error* dari struktur data atau akses eksternal *database*, *error* dari kinerja dan *error* dari inisialisasi.

Equivalence class partitioning adalah sebuah metode *black box* terarah yang meningkatkan efisiensi dari pengujian dan meningkatkan *coverage* dari *error* yang potensial. Sebuah *equivalence class* adalah sebuah kumpulan dari nilai *variable input* yang memprduksi output yang sama. Selanjutnya *output correctness test* merupakan pengujian yang memakan sumber daya paling besar dari pengujian. Pada kasus yang sering terjadi dimana hanya output *correctness* test yang dilakukan, maka sumber daya pengujian akan digunakan semua. Implementasi dari kelas-kelas pengujian lain tergantung dari sifat produk *software* dan pengguna selanjutnya dan juga prosedur dan keputusan pengembang. *Output correctness test* mengaplikasikan konsep dari test case. Pemilihan test case yang baik dapat dicapai dengan efesiensi dari penggunaan *equivalence class partitioning*. Jenis-jenis *test case* sebagai berikut:

1. Availability test

Availability didefinisikan sebagai waktu reaksi yaitu waktu yang dibutuhkan untuk mendapatkan informasi yang diminta atau waktu yang dibutuhkan oleh *firmware* yang diinstal pada perlengkapan komputer untuk bereaksi. *Availability* adalah yang paling penting dalam aplikasi online sistem informasi yang sering digunakan. Kegagalan *firmware software* untuk memenuhi persyaratan ketersediaan dapat membuat perlengkapan tersebut tidak berguna.

2. Realiability test

Realiability berkaitan dengan fitur yang dapat diterjemahkan sebagai kegiatan yang terjadi sepanjang waktu seperti waktu rata-rata antara kegagalan (misalnya 500 jam), waktu rata-rata untuk *recovery* setelah kegagalan sistem (misalnya 15 menit) atau *average downtime* per bulan (misalnya 30 menit per bulan). Persyaratan realiabilitas memiliki efek selama regular full-capacity operasi sistem. Harus diperhatikan bahwa penambahan faktor *software realiability* juga berkaitan dengan perangkat, sistem operasi, dan efek dari sistem komunikasi data.

3. Stress test

Stress test terdiri dari 2 tipe pengujian yaitu load test dan *durability test*. Suatu hal yang mungkin untuk melakuakn pengujian-pengujian tersebut setelah penyelesaian sistem *software. Durability test* dapat dilakukan hanya setelah *firmware* atau sistem informasi *software* diinstall dan siap untuk diuji. Pada load test berkaitan dengan functional performance system dibawah beban maksimal operasional, yaitu maksimal transaksi per menit, hits per menit ke tempat internet dan sebagainya. Load test, yang biasanya dilakukan untuk beban yang lebih tinggi dari yang diindikasikan spesifikasi persyaratan merupakan hal yang penting untuk sistem software yang rencananya akan dilayani secara simultan oleh sejumlah pengguna. Pada sebagian besar kerja sistem *software*, beban maksimal menggambarkan gabungan beberapa tipe transaksi. Selanjutnya durability test dilakukan pada kondisi operasi fisik yang ekstrem seperti temperatur yang tinggi, kelembaban, mengendara dengan kecepatan tinggi, sebagai detail persyaratan spesifikasi d rabilitas. Jadi, dibutuhkan untuk *real-time firmware* yang diintegrasikan ke dalam sistem seperti sistem senjata, kendaraan transport jarak jauh, *Internet Of Things* (IOT) dan keperluan meterologi. Isu ketahan pada firmware terdiri dari respon firmware terhadap efek cuara seperti temperatur panas atau dingin yang ekstrem, debu, kegagalan operasi ekstrem karena kegagalan listrik secara tibatiba, loncatan arus listrik dan putusnya komunikasi secara tiba-tiba.

4. Training usuability test

Ketika sejumlah besar pengguna terlibat dalam sistem operasi, training *usability requirement* ditambahkan dalam agenda pengujian. Lingkup dari training usability test

ditentukan oleh sumber yang dibutuhkan untuk melatih pekerja baru untuk memperoleh level perkenalan dengan sistem yang ditentukan atau untuk mencapai tingkat produksi tertentu. Detail dari pengujian ini, sama halnya dengan yang lain, didasarkan pada karakteristik pekerja. Hasil dari pengujian ini harus menginspirasi rencana dari kursus pelatihan dan *follow-up* serta memperbaiki sistem operasi *softwar*.

5. Operational usability test

Fokus dari pengujian ini adalah produktifitas operator, yang aspeknya terhadap sistem yang mempengaruhi performance dicapai oleh operator sistem. Operational usability test dapat dijalankan secara manual.