BAB 2
TINJAUAN PUSTAKA

2.1 Internet Of Things

Internet of things (IOT) adalah struktur dimana objek, orang disediakan dengan identitas eksklusif dan kemampuan untuk pindah data melalui jaringan tanpa memerlukan dua arah antara manusia ke manusia yaitu sumber ke tujuan atau interaksi manusia ke computer.

Internet of things merupakan perkembangan keilmuan yang sangat menjajikan untuk mengoptimalkan kehidupan berdasarkan sensor cerdas dan peralatan pintar yang berkerjasama melalui jaringan internet.[4]

2.2 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa permodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atau svisi mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancagan mereka dengan yang lain. *Unified Modeling Language* (UML) merupakan kesatuan dari Bahasa permodelan yang dikembangkan booch, *Object Modeling Technique* (OMT) dan *Object Orented Software Engineering* (OOSE) Metode ini menjadikan proses analisis dan design kedalam empat tahapan iterative, yaitu identifikasi kelas-kelas, dan objek-objek, identifikasi semantic dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode Booch adalah pada detail dan kayanya dengan notasi dan elemen, permodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan entity-relationship. Tahapan utama dalam metodologi ini adalah nalisis, design sistem, design objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung konsep OO.

Metode OOSE dari Jacobson lebih memberi menekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis,

design dan implementasi, dan model Useran. Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak. Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya.

Sebagai sebuah notasi grafis yang relative sudah dibakukan (*open standard*) dan dikontrol oleh OMG (*Object Management Group*) mungkin lebih dikenal sebagai badan yang berhasil membakukan CORBA (Common Object Request Broker Architecture), UML menawarkan banyak keistimewaan. UML tidak hanya dominan dalam penotasian di lingkunan OO tetapi juga popular di luar lingkungan OO. Paling tidak ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan Bahasa pemograman. Sebagai sebuah sketsa, UML bisa berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dari sistem. Dengan demikian semua anggota tim akan mempunyai Gambaran yang sama tentang suatu sistem. UML bisa juga berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detil. Dengan cetak biru ini maka akan bisa diketahui informasi detail tentang coding program (*forward engineering*) atau bahkan membaca program dan menginterpretasikannya kembali kedalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana code program yang tidak terdokumentasi asli hilang atau bahkan elum dibuat sama sekali. Sebagai bahasa pemograman, UML dapat menterjemahkan diagram yang ada di UML menjadi code program yang siap untuk di jalankan [14].

Struktur sebuah sistem dideskripsikan dalam 5 view diman asalah satu diantaranya scenario, scenario ini memegang peran khusus untuk mengintegrasikan content ke view yang lain. Kelima view tersebut berhubungan dengan diagram yang dideskripsikan di UML. Setiap view berhubungan dengan perspektif tertentu di mana sistem akan diuji. View yang erbeda akan menekankan pada aspek yang berbeda dari siste yang mewakili ketertarikan sekelompok stakeholder tertentu. Penjelasan lengkap tentang sistem bisa dibentuk dengan

menggambungkan informasi-informasi yang ada pada kelima view tersebut sebagai berikut :

1. *Scenario*, mengGambarkan interaksi diantara objek dan diantara proses. *Scenario* ini digunakan untuk dientifikasi elemen arsitektur, ilustrasi dan validasi disain arsitektur serta sebagai titik awal untuk Useran prototype arsitektur. *Scenario* ini bisa juga disebut dengan *use case view*. *Use case view* ini mendefinisikan kebutuhan sistem karena mengantdung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari rancangan sistem. Itulah sebabnya *use case view* menajdi pusat peran dan sering dikatakan yang mengdrive proses pengembangan perangkat lunak.
2. *Development View*, menjelaskan sebuah sistem dari perspektif programmer dan terkonsentrasikan ke manajemen perangkat lunak. *View* ini dikenal juga sebagai *implementation view*. Diagram UML yang termasuk dalam development *view* di antaranya adalah *component diagram* dan *package diagram*.
3. *Logical View*, terkait dengan fungsionalitas sistem yang dipersiapkan untuk User akhir. *Logical view* mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi *object diagram*, *class diagram*, *state machine diagram* dan *composite structure diagram*.
4. *Physical View*, mengGambarkan sistem dari perspektif *sistem engineer*. Fokus dari *physical view* adalah *topologi* sistem perangkat lunak. *View* ini dikenal juga sebagai deployment view. Yang termasuk dalam *physical* view ini adalah *deployment diagram* dan *timing diagram*.
5. *Process View*, berhubungan erat dengan aspek dinamis dari sistem, proses yang terjadi di sistem dan bagaimana komunikasi yang terjadi di sistem serta tingkah laku sistem saat dijalankan. *Process view* menjelaskan apa itu *concurrency*, *distribusi integrasi*, *kinerja* dan lain-lain. Yang termasuk dalam process *view* adalah *activiy diagram*,

communication diagram, *sequence diagram* dan *interaction overview diagram*.

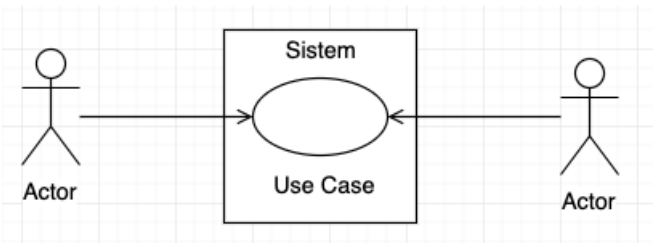
Mesikpun UML sudah cukup banyak menyediakan diagram yang bisa membantu mendefinisikan sebuah aplikasi, tidak berarti bahwa semua diagram tersebut akan bisa menjawab persoalan yang ada. Dalam banyak kasus, diagram lain selain UML sangat banyak membantu. Pada penelitian ini konsep UML digunakan untuk mengGambarkan bagaimana sistem bekerja, hubungan antar class, memberi Gambaran kepada user sistem yang akan di gunakan dan memberikan penjelasan detail pemanggilan fungsi-fungsi atau method dalam class.[5]

2.2.1 **Diagram UML**

UML menyediakan 4 macam diagram untuk menodelkan aplikasi perangkat lunak ber orientasi objek. Yaitu:

1. **Use Case Diagram**

   Use case diagram adalah deskripsi fungsi dari sebuah sistem dari perspektif User. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara user (User) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara User 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 User. 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.[5]



Gambar 2.1 Use Case Diagram

Berikut ini adalah bagian dari sebuah use case diagram :

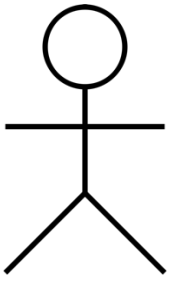
a. Use Case

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.[5]

Gambar 2.2 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 2.3 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. *Class Diagram*

Class diagram memebrikan pandangan secara luas dari suatu sistem dengan menunjukan kelas-kelasnya dan hubungan mereka. Diagram class bersifat statis, menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan.

Diagram class mempunyai 3 macam relationships (hubungan), sebagai berikut :

1. Association

Hubungan antara bagian dari dua kelas. Terjadi *association* antara dua kelas jika salah satu bagian dari kelas menegetahui yang lainnya dalam melakukan suatu kegiatan. Didalam diagram, sebua association adalah penghubung yang menghubungkan dua kelas.

2. Aggregation

   Associaion dimana salah satu kelasnya merupakan bagian dari suatu kumpulan. Aggregation memiliki titik pusat yang mencakup keseluruhan bagian. Sebagai comtoh : OrderDetail merupakan kumpulan dari Order.

3. Generalization

   Hubungan turunan dengan mengasumsikan suatu kelas merupakan *superClass* (kelas super) dari kelas yang lain. *Generalization* memiliki tingkatan yang berpusar pada *superClass.* Contoh: Payment adalah *superClass* dari Cash, Check, dan Credit.

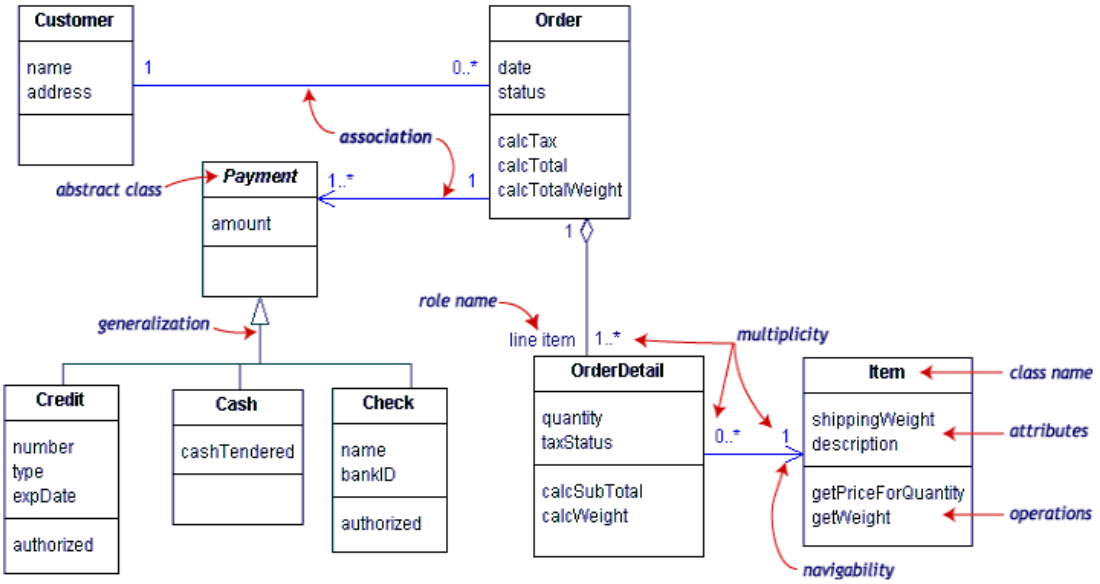
Untuk tambahan bahwa association mempunyai 2 titik. Salah satu titik bisa memiliki label untuk menjelaskan association tersebut, contoh : OrderDetail adalah line. Item untuk setiap permintaan. Panah navigability (pengatur alur arah) dalam suatu association menggambarkan arah mana association dapat ditransfer atau disusun. Seperti dalam contoh : OrderDetail dapat disusun dari item-nya, namun tidak bisa sebaliknya. Panah ini juga menjelaskan siapa "memiliki" implementasi dari association; dalam kasus ini OrderDetail memiliki Item. Association tanpa arah panah merupakan bidirectional (bolak-balik). Multiplicity dari suatu titik association adalah angka kemungkinan bagian dari hubungan kelas dengan single instance (bagian) pada titik yang lain. Multiplicity berupa single number (angka tunggal) atau range number (angka batasan). Pada contoh, hanya bisa satu 'Customer' untuk setiap 'Order', tapi satu 'Customer' hanya bisa memiliki beberapa 'Order'.[6]

Tabel 2.1 Tabel Multiplicity

Multiplicities	Artinya
0..1	Nol atau satu bagian. Notasi ***n . . m*** menerangkan ***n*** sampai ***m*** bagian.
0..* *or* ****	Tak hingga pada jangkauan bagian (termasuk kosong).
1	Tepat satu bagian.

1..*	Sedikitnya hanya satu bagian.

Setiap diagram Class memiliki Class (kelas), association, dan multiplicity. Sedangkan navigability (alur arah) dan role (kegiatan) merupakan optional (tidak diharuskan). [6]

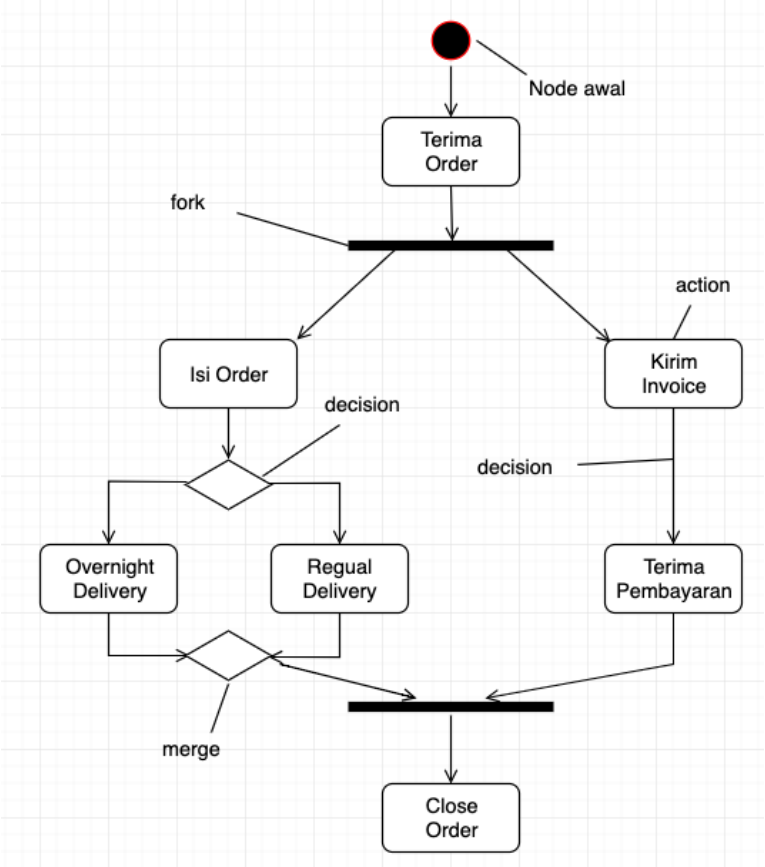


Gambar 2.4 Contoh *Class Diagram* [6]

3. *Activity Diagram*

Activity diagram adalah bagian penting dari UML yang mengGambarkan aspek dinamis dari sistem. Logika procedural, 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 mendukug 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 sctivity diagram, perlu adanya

pemahaman yang jelas tentang elemen yang akan digunakan di activity diagram. Elemen tuama 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. Langa selanjutnya perlu penjabaran tata letak dari keseluruhan aliran agara bisa ditransofrmasikan ke activity diagram.[5]



Gambar 2.5 *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.

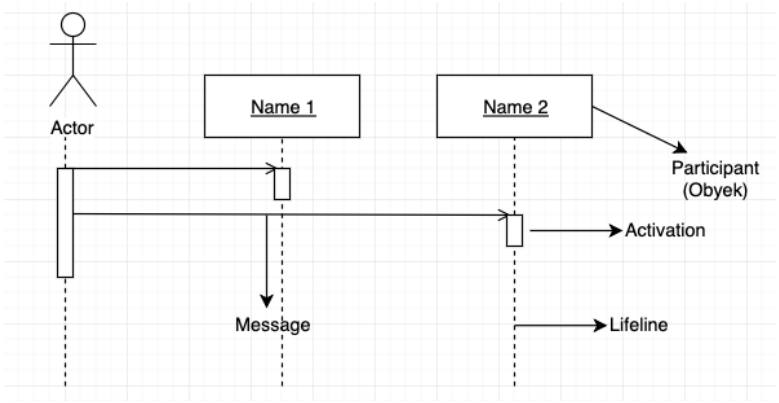
g. *Final state*

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

4. *Sequence Diagram*

Sequence diagram digunakan untuk mengGambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukan sejumlah contoh obyek dan *message* (pesan) yang diletakan diantara obyek-obyek 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 detak 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*. Sebuah *message* bisa jadi simple, *synchronous* atau *asynchronous*. *Message* yang simple adalah sebuah perpindahan (*transfer*) *control* dari satu *participant* ke *participant* yang lainnya. Jika sebuah *participant* mengirimkan sebuah message *synchronous*, maka jawaban atas *message* tersebut akan ditunggu sebelum diproses dengan ururssannya. Namun jika message message *asynchronous* yang dikirimkan, maka jawaban atas *message* tersebut tidak perlu ditunggu. *Time* adalah diagram yang mewakili waktu pada arah *vertical*. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijadikan terlebih dahulu dibanding *message* yang lebih dekat ke bawah.[5]



Gambar 2.6 *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.

2.3 Object Oriented (OO)

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 Useran RDBMS (Relational Database Management

Sistem) 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.[5] 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 User 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 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) adalah model teknik pendekatan yang menyatakan atau mengGambarkan hubungan suatu model. Didalam hubungan ini tersebut dinyatakan yang utama dari ERD adalah menunjukan objek data (*Entity*) dan hubungan (*Relationship*), yang ada pada Entity berikutnya. Proses memungkinkan analis menghasilkan struktur basis data dapat disimpan dan diambil secara efisien.[7] Pada penelitian ini konsep ERD digunakan untuk

merancang database yang akan di gunakan dalam sistem. Simbol-simbol dalam *Entity Relationship Diagram* (ERD) adalah sebagai berikut:

1. Entitas, suatu yang nyata atau bastrak yang mempunyai karakteristik dimana kita akan menyimpan data.
2. Atribut, ciri umum semua atau sebagian besar instansi pada entitas tertentu.
3. Relasi, hubungan alamiah yang terjadi antara satu atau lebih entitas.
4. Link, garis penghubung atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi.

Hubungan kardinalitas relasi dalam *Entity Relationship Diagram* (ERD) adalah sebagai berikut:

1. Satu ke satu (One to One)

   Setiap elemen dari Entitas A berhubungan paling banyak dengan elemen pada Entitas B. Demikian juga sebaliknya setiap elemen B berhubungan paling banyak satu elemen pada Entitas A.
2. Satu ke banyak (One to Many)

   Setiap elemen dari Entitas A berhubungan dengan maksimal banyak elemen pada Entitas B. Dan sebaliknya setiap elemen dari Entitas B berhubungan dengan paling banyak satu elemen di Entitas A.
3. Banyak ke satu (Many to One)

   Setiap elemen dari Entitas A berhubungan paling banyak dengan satu elemen pada Entitas B. Dan sebaliknya setiap elemen dari Entitas B behubungan dengan maksimal banyak elemen di entitas A.
4. Banyak ke banyak (Many to Many)

   Setiap elemen dari Entitas A berhubungan maksimal banyak elemen pada Entitas B demikian sebaliknya.

2.5 Python

Python diciptakan oleh Guido van Rossum di Belanda pada tahun 1990 dan namanya diambil dari acara televisi kesukaan Guido *Monty Python's Flying Circus*. Van Rossum mengembangkan *Python* sebagai hobi, kemudian *Python*

menjadi bahasa pemrograman yang dipakai secara luas dalam industri dan pendidikan karena sederhana, ringkas, sintaks intuitif dan memiliki pustaka yang luas. Python saat ini dikembangkan dan dikelola oleh tim relawan yang besar dan tersedia secara gratis dari Python Software Foundation. Python termasuk dari jajaran bahasa pemograman tingkat tinggi seperti bahasa pemograman C, C++, Java, Perl dan Pascal, dikenal juga bahasa pemograman tingkat rendah, yang dikenal sebagai bahasa mesin yaitu bahasa pemograman assembly, kenyataanya komputer hanya dapat mengeksekusi bhasa tingkat rendah, jadi bahasa tingkat tingkat tinggi harus melewati beberapa proses untuk diubah ke bahasa pemograman tingkat rendah, hal tersebut merupakan kelemahan yang tidak berarti bahasa pemograman tingkat tinggi. Tetapi kekurangna tersebut tidak sebanding dengan kelebihanya. Pertama, lebih mudah memprogram sebuah aplikasi dengan bahasa tingkat tinggi. Lebih cepat, lebih mudah dimengerti menulis program komputer dengan bahasa tingkat tinggi, dan juga kesalahan dalam penulisan program cenderung tidak mengalami kesalahan yang berarti. Kedua bahasa pemograman tingkat tinggi lebih porTabel dalam arti bisa digunakan untuk menulis di berbagai jenis arsitektur komputer yang berlainan dengan sedikit modifikasi ataupun tidak memerlukan modifikasi sama sekali. Bahasa pemograman tingkat rendah hanya dapat berjalan di satu jenis arsitektur komputer dan harus ditulis ulang untuk menjalankannya di lain mesin, hal ini diakrenakan karena perbedan urutan register dan services – servicesnya. Pada penelitian ini bahasa pemograman python digunakan sebagai komponen utama untuk pembanguan website dan konfigurasi *raspberry pi*.[8]

Gambar 2.7 Logo *Phyton*

Python digunakan di berbagai bidang pengembangan. Berikut beberapa implementasi bahasa python yang paling popular:

1. Website

   Bahasa pemograman python dapat digunakan sebagai server side yang diintegrasikan dengan berbagai internet *protocol* misalnya JSON, *Email Processing*, FTP, dan IMAP. Selain itu python juga mempunyai library pendukung untuk pengembangan website seperti Django, Flask, Pyramid dan Bottle.

2. Penelitian

   Python dapat digunakan juga untuk melakukan riset ilmiah untuk mempermudah perhitungan numerik. Misalnya penerapan algoritma KNN, Naïve Bayes dan lain-lain. Beberapa library yang sering digunakan untuk riset seperti Pandas, Numpy, Mathplotlib dan lain-lain.

3. Media Pembelajaran

   Python dapat digunakan sebagai media pembelajaran di universitas. Python sangat mudah dan hemat untuk dipelajari sebagai Object Oriented Programming dibandingkan bahasa lainnya seperti MATLAB, C++, dan C#.

4. Graphical User Interface (GUI)

   Python dapat digunakan untuk membangun interface sebuah aplikasi. Tersedia library untuk membuat GUI menggunakan python, misalnya Qt, win32extension, dan GTK+.

5. Internet Of Things (IOT)

   Bahasa pemrograman Python mendukung ekosistem *Internet of Things* (IOT) dengan sangat baik. *Internet Of Things* (IOT) merupakan sebuah teknologi yang menghubungkan benda-benda di sekitar kita ke dalam sebuah jaring-jaring yang saling terhubung.

2.6 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: database management sistem) 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 Userannya tidak cocok dengan Useran GPL [18]. Pada penelitian ini MySQL digunakan sebagai platform penyimpanan data dan implementasi model rancangan database yang sudah di buat.[9]

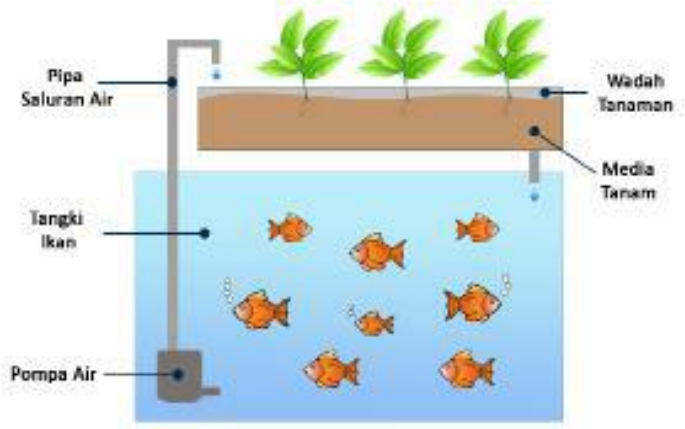
Gambar 2.8 Logo MySQL

MySQL memiliki banyak fitur. fitur dimiliki MySQL sebagai berikut :

1. *Relational Database Sistem,* Seperti halnya software database lain yang ada di pasaran, MySQL termasuk RDBMS.
2. *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.*
3. *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*.
4. *Performace Tuning*, *MySQL* mempunyai kecepatan yang cukup baik dalam menangani *query-query* sederhana, serta mampu memproses lebih banyak *SQL* per satuan waktu.
5. *Sub Select*.
6. *Views*.
7. *Stored Prosedured* (SP).
8. *Triggers.*
9. *Replication*.
10. *Foreign Key*.
11. Security yang baik.

2.7 Akuaponik

Akuaponik dapat digambarkan sebagai penggabungan antara sistem budidaya akuakultur (budidaya ikan) dengan hidroponik (budidaya tanaman/sayuran tanpa media air). Sistem kerja akuaponik sangat sederhana. Air beserta kotoran yang berasal dari budidaya ikan disalurkan kepada tanaman karena mengandung banyak nutrisi yang dibutuhkan oleh tanaman. Tanaman akan menyerap nutrisi yang berasal dari air dan kotoran ikan tadi. Sebagai gantinya , tanaman akan memberikan oksigen kepada ikan melalui air yang sudah tersaring oleh media tanaman.[1]



Gambar 2.9 Sistem Kerja Akuaponik[1]

2.2.1 Akuakultur

Akuakultur berasal dari Bahasa inggris *aquaculture (aqua* = perairan, *culture* = budidaya) dan diterjemahkan ke dalam Bahasa Indonesia menjadi budidaya perairan atau budidaya perikanan. Budidaya perikanan sebagai campur tangan atau upaya-upaya manusia untuk meningkatkan produktivitas perairan melauli kegiatan budidaya. Kegiatan budidaya yang dimaksud adalah usaha pemeliharaan untuk mempertahankan kelangsungan hidup (survival), menumbuhkan (growth) dan memperbanyak (reproduction) biota akuatik. Definisi ini berkembang dengan memperhatikan evolusi produksi yang berlangsung di dalam perikanan.[10]

Tabel 2.2 Jenis Ikan Akuaponik

No.	Jenis Ikan	No.	Jenis Ikan Hias

1.	Patin	1.	Koi
2.	Lele	2.	Oscar
3.	Nila	3.	Arwana
4.	Bawal	4.	Maskoki
5.	Guramai	5.	Louhan

2.2.2 Hidroponik

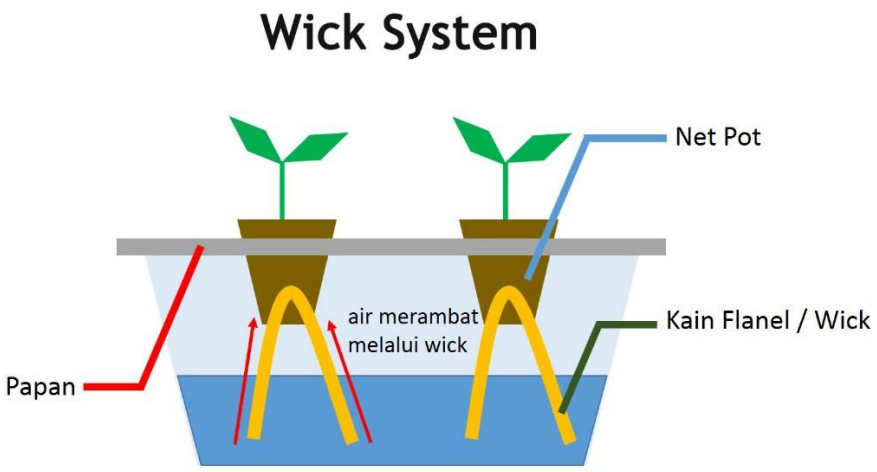
Pada awal tahun 1930 di Berkley California, William Frederick Gericke mempelopori sistem hidroponik, yaitu sistem budidaya menggunakan air yang mengandung nutrisi dan mineral tanpa air. Saat ini pertanian menggunakan hidroponik telah diterapkan secara luas dan memiliki beberapa keunggulan dibandingkan dengan sistem budidaya konvensional, yaitu mengurangi risiko atau masalah budidaya yang berhubungan dengan air seperti gangguan serangga, jamur dan bakteri yang hidup di air. Sistem ini juga lebih mudah dalam pemeliharaan seperti tidak melibatkan proses penyiangan dan pengolahan air dalam budidaya tanamannya. Selanjutnya proses budidaya dilakukan dalam kondisi lebih bersih tanpa menggunakan pupuk kotoran hewan. [11]

2.2 Jenis Hidroponik

Beberapa jenis hidroponik yang umum digunakan antara lain :

1. *Wick Sistem*

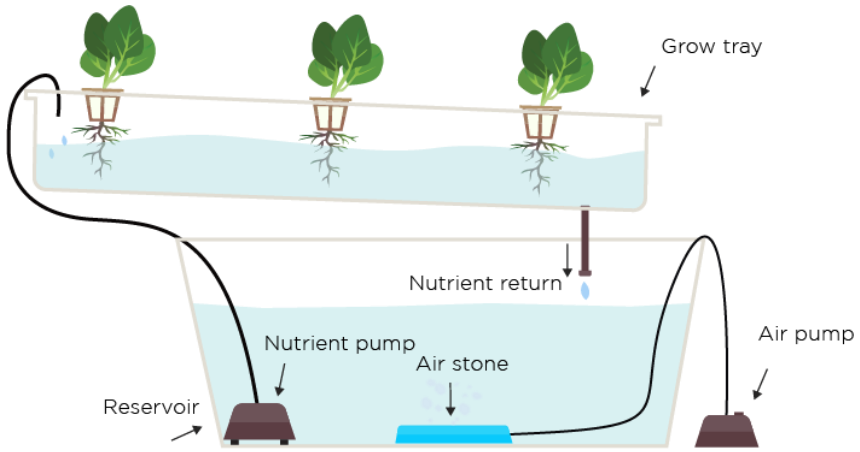
   Sistem ini merupakan model hidroponik yang paling sederhana, yaitu menggunakan sumbu yang menghubungkan pot tanaman dengan media larutan nutrisi.



Gambar 2.10 ***Wick Sistem Hidroponik***[11]

2. *Nutrient Film Technique (NFT)*

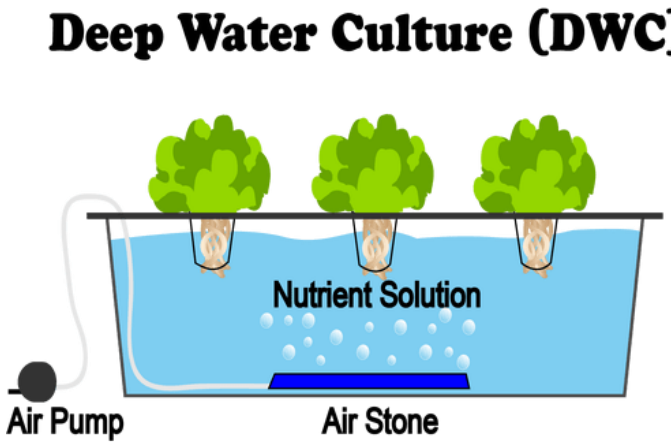
   Larutan nutrisi secara terus menerus dialirkan mengenai akar tanaman menggunakan pipa PVC menggunakan ompa dengan teknik resirkulasi.



Gambar 2.11 Nutrient Film Technique (NFT) Hidroponik[11]

3. *Deep Water Culture (DWC)*

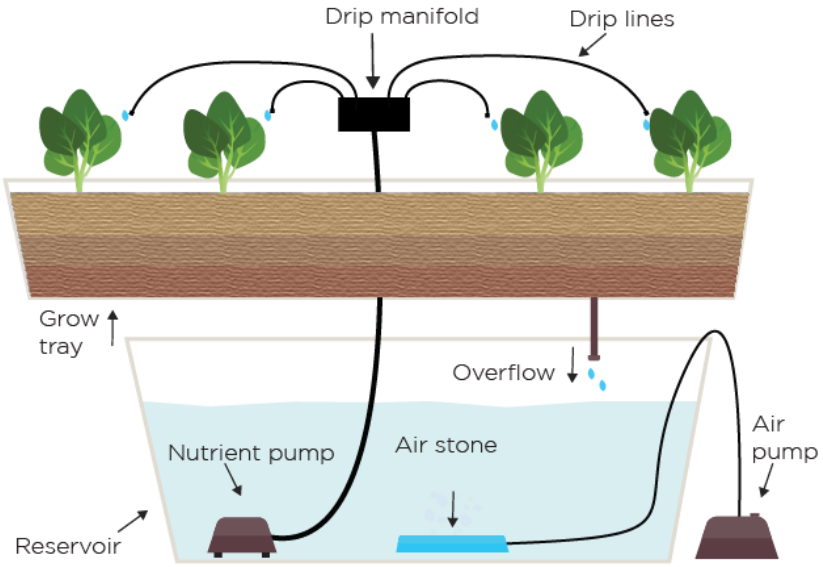
   Tanaman dibuat mengapung pada larutan nutrisi sehingga akar tanaman terendam terus menerus. Useran pompa habya untuk menghasilkan oksigen di dalam larutan nutrisi.



Gambar 2.12 Deep Water Culture (DWC) Hidroponik[11]

4. *Drip Sistem*

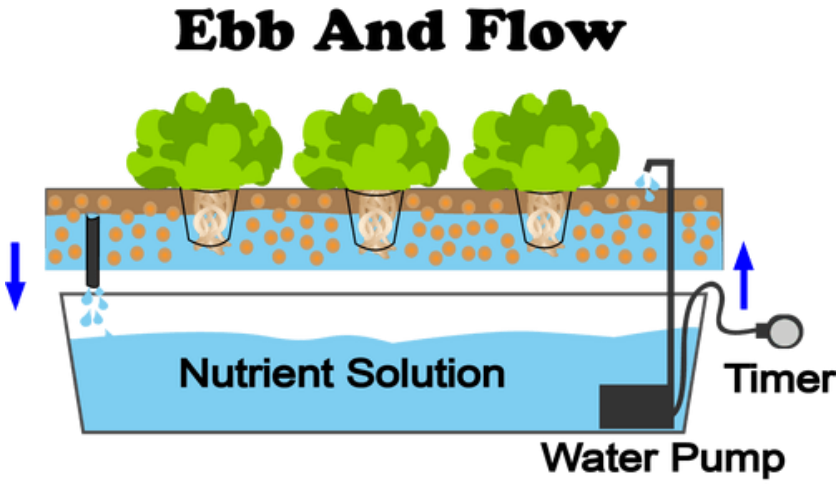
   Sistem ini menggunakan 2 (dua) buah kontaoner terpisah yaitu nagian atas dan bawah. Container atas untuk tanaman dan yang bawah untuk larutan nutrisi. Larutan nutrisi dipompa naik dan menyiram batang tanaman dan larutan sisa akan turun ke container bawah setelah melewati media tanam dan akar tanaman.



Gambar 2.13 Drip Sistem Hidroponik[11]

5. *Ebb and Flow Sistems (Flood and Drain Sistem)*

Pengaturannya mirip dengan sistem infus, di mana ada dua container, yang satu di atas berisi tanaman dalam pot dengan substrat dan yang ada di bagian bawah yang mengandung larutan nutrisi. Pemberian nutrisi untuk tanaman dilakkan dengan sistem pasang surut, yaitu bergantian memenuhi container atas dengan larutan nutrisi dan kemudian mengosongkan larutan nutrisi dan kembali ke container bawah.[11]



Gambar 2.14 Ebb anf Flow Sistems Hidroponik[11]

2.3 Media Tanam Hidroponik dan Tanaman Hidroponik

Macam-macam media tanam hidroponik dapat dilihat pada tabel di bawah ini.[12]

Tabel 2.3 Media Tanam Hidroponik

ORGANIK	Non-Organik
Arang Sekam	Perlit
Serbuk Gergaji	Rockwool
Serabut Kelapa	Clay Granular
Akar Pakis	Sand
Vermikulit	Gravel
Gambut	Batu (Apung, Bata, Karang)

Macam-macam tanaman untuk budidaya hidroponik dapat dilihat pada tabel di bawah ini.[12]

Tabel 2.4 Tanam Untuk Hidroponik

Sayuran	Buah	Tanaman Hias
Selada	Melon	Krisan
Sawi	Tomat	Gerbera
Pakchoi	Mentimun	Anggrek
Tomat	Semangka	Kaladium
Wortel	Strawberi	Kaktus
Asparagus	paprika	
Brokoli		
Cabai		
Seledri		
Bawang merah		
Bawang putih		
Bawang daun		
Terong		

2.3 Mikrokontroler

Suatu kontroler digunakan untuk mengontrol suatu proses atau aspek-aspek dari lingkungan. Satu contoh aplikasi dari mikrokontroler adalah untuk memonitor rumah kita. Ketika suhu naik kontroler membuka jendela dan sebaliknya. Pada masanya, kontroler dibangun dari komponen-komponen logika secara keselruhan, sehingga menjadikannya besar dan berat. Setelah itu barulah dipergunakan mikroprosesor sehingga keseluruhan ontroler masuk kedalam PCV yang cukup kecil. Hingga saat ini masih mering kita lihat kontroler yang dikendalikan oleh mikroprosesor biasa (Zilog Z80, Intel 8088, Motorola 6809, dsb).

Prose pengecilan komponen terus berlangsun, semua komponen diperlukan guna membangun suatu kontroler dapat dikemas dalam satu keeping. Maka lahirlah computer keeping tunggal (*one chip microcomputer)* atau disebut juga mikrokontroler. Mikrokontroler adalah suatu IC dengan kepadatan yang sangat

tinggi, dimana semua bagan yang diperlukan untuk suatu kontroler sudah dikemas dalam satu keeping, biasanya terdiri dari :

1. CPU *(Central Processing Unit)*
2. RAM (*Random Access Memory)*
3. EEPROM/EPROM/PROM/ROM
4. I/O, Serial dan Paralel
5. Timer
6. *Interupt Controller.*

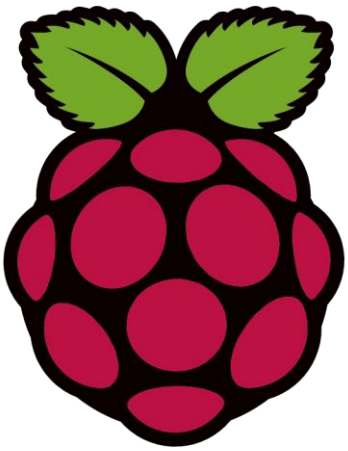
Rata-rata mikrokontroler memiliki intruksi manipulasi *bit,* akses ke I/O secara langsung dan mudah, dan proses *interrupt* yang cepat dan efisien. Dengan kata lain mikrokontroler adalah "Solusi satu Chip" yang secara drastic mengurangi jumlah komponen dan biaya desain.[13]

2.4 *Raspberry pi*

Raspberry pi, sering disingkat dengan nama Raspi, adalah komputer papan tunggal/*single-board circuit* (SBC) yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresousi tinggi. *Raspberry pi* dikembangkan oleh yayasan nirlaba, Rasberry Pi Foundation, yang digawangi sejumlah pengembang dan ahli komputer dari Universitas Cambridge, Inggris.

Ide dibalik *Raspberry pi* diawali dari keinginan untuk mencetak pemrogram generasi baru. Seperti disebutkan dalam situs resmi *Raspberry pi* Foundation, waktu itu Eben Upton, Rob Mullins, Jack Lang, dan Alan Mycroft, dari Laboratorium Komputer Universitas Cambridge memiliki kekhawatiran melihat kian turunnya keahlian dan jumlah siswa yang hendak belajar ilmu komputer. Mereka lantas mendirikan yayasan *Raspberry pi* bersama dengan Pete Lomas dan David Braben pada 2009. Tiga tahun kemudian, *Raspberry pi* Model B memasuki produksi massal. Dalam peluncuran pertamanya pada akhir Febuari 2012 dalam beberapa jam saja sudah terjual 100.000 unit. Pada bulan Februari 2016, *Raspberry pi* Foundation mengumumkan bahwa mereka telah menjual 8 juta

perangkat Raspi, sehingga menjadikannya sebagai perangkat paling laris di Inggris.[14]



Gambar 2.15 Lambang *Raspberry*

2.2 Sistem Operasi *Raspberry pi*

Untuk menggunakan *Raspberry pi* kita memerlukan operating sistem (contoh OS: windows, linux,mac ,Unix dst) yg dijalankan dari SD card pada board Raspberry tidak seperti pada board microcontroller AVR yg selama ini kita pakai tanpa OS. Operating sistem yang banyak dipakai antara lain Linux distro Raspbian. OS disimpan di SD card dan saat proses boot OS hanya bisa dari SD card tdk dari lokasi lain.

OS yang bisa di jalankan di Raspberry board antara lain :

1. Arch Linux ARM
2. Debian GNU/Linux
3. Gentoo
4. Fedora
5. FreeBSD
6. NetBSD
7. Plan 9
8. Inferno
9. Raspbian OS

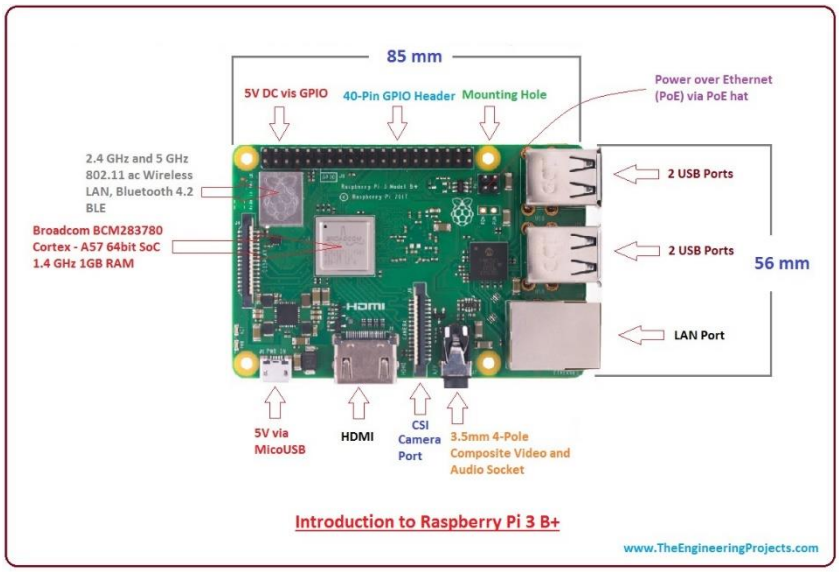
10. RISC OS

11. Slackware Linux.

Jadi dalam menggunakan *microcomputer Raspberry pi* ini kita seperti menggunakan PC yg berbasis linux yg mempunyai input output digital spt yg ada di board microcontroller.[15]

2.3 *Raspberry pi* 3 Model B+

Raspberry pi 3 Model B+ adalah produk terbaru dalam jajajran *raspberry pi* generasi ke 3. Apabila dibandingkan dengan model sebelumnya (*Raspberry pi* 3 Model B) yang hadir dua tahun lalu, *Raspberry pi* 3 Model B+ kini diperkuat prosesor Broadcom BCM2873B0 yang lebih cepat (quad-core Cortex-A53 64-bit 1.4 GHz). Sektor konektivitasnya juga semakin mumpuni berkat dukungan standar *Wi-Fi* 802.11ac (dual band, 2,4 GHz dan 5 GHz) dan *Bluetooth* 4.2 (Pi sebelumnya masih Bluetooth 4.1).[16]

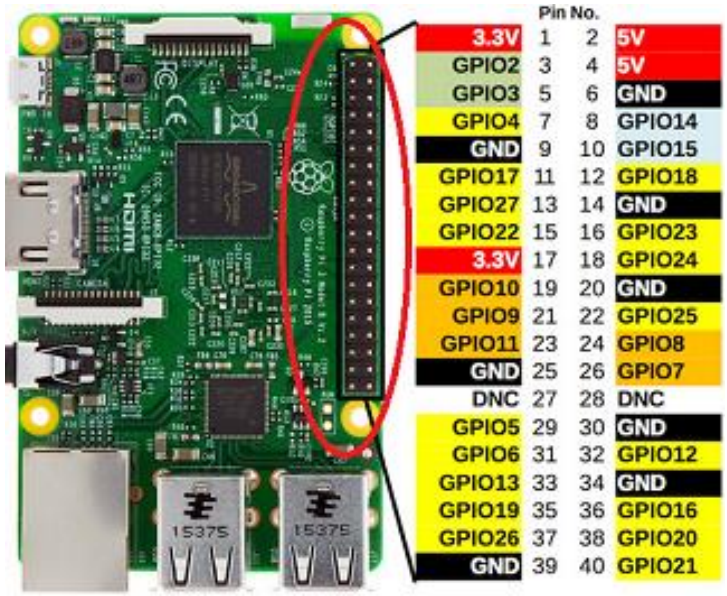


Gambar 2.16 Board *Raspberry pi* 3 Model B+

2.4 General Purpose Input Output (GPIO)

General Purpose Input Output (GPIO) adalah pin atau tempat yang dapat digunakan sebagai input atau output. Salah satu alasan *Raspberry pi* sangat populer digunakan adalah karena memiliki 40-pin GPIO yang memungkinkan User untuk menghubungkan komponen elektronik dan mengendalikan mereka

dengan sebuah program. Kode program untuk proyek komputasi fisik biasanya ditulis dengan Python, akan menjadi jauh lebih mudah jika kita menggunakan library yang sudah disediakan oleh GPIO *Raspberry pi*.[14]



Gambar 2.17 GPIO pada *Raspberry pi*

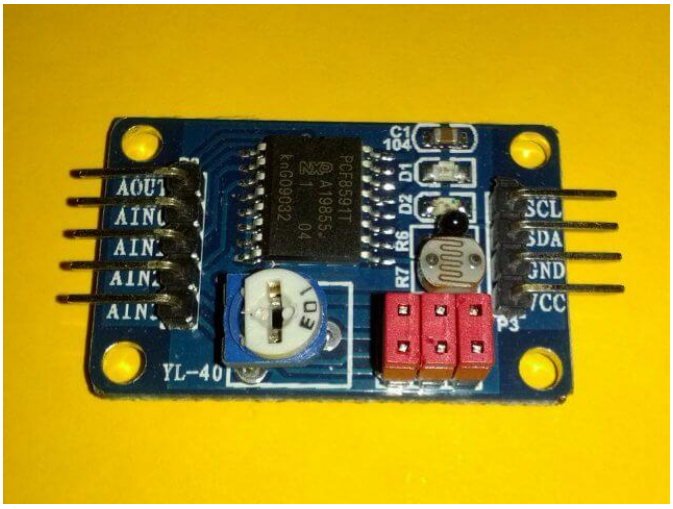
2.5 Bahasa Pemrograman *Python*

Python adalah bahasa pemograman tingkat tinggi yang dapat digunakan secara luas di berbagai bidang. Python diciptakan pertama kali oleh Guido van Rossum pada tahun 1991. Sintaks dan fungsi pada Python dipengaruhi oleh beberapa bahasa seperti C, C++, Lisp, Perl dan Java. Oleh karena itu, kita dapat menemui konsep pemograman prosedural, fungsional dan object-oriented di Python. Python relatif mudah dipelajari bila dibandingkan dengan C++, Java dan PHP karena sintaks Python lebih singkat, lebih jelas dan mudah dipahami oleh programmer pemula.

Meskipun di Indonesia Python tidak sepopuler C/C++ ataupun Java, Python sangat patut untuk dipelajari karena banyak pilihan library Python yang bisa kita gunakan yang biasanya tidak dapat kita temui di bahasa lain. Pada bagian ini akan dijelaskan dasar-dasar dari pemograman Python. Buka terminal, lalu ketik perintah python untuk membuka console Python.[14]

2.6 Analog to Digital Converter PCF8591T

PCF8591T adalah perangkat akusisi data CMOS 8-bit berdaya rendah satu catu daya tunggal dengan empat input analog, satu output analog dan antarmuka bus serial I2C. Tiga pin alamat A0, A1 dan A2 digunakan untuk memprogram alamat perangkat keras, memungkinan Userkan sehingga delapan perangkat yang terhubung ke bus-I2C tanpa perangkat keras tambahan. Alamat, control dan data ked an dari perangkat ditransfer secara serial melalui dua jaur dua arah bus I2C. Fungsi perangkat ini meliputi multiplexing input analog, fungsi on-chip track and hold, konversi analog-ke-digital 8-bit dan konversi digital-ke-analog 8-bit. Tingkat konversi maksimum diberikan oleh kecepatan maksimum bus-I2C. [17]



Gambar 2.18 PCF8591T

2.7 Sensor

Sensor adalah alat untuk mendeteksi/mengukur sesuatu, yang digunakan untuk mengubah variasi mekanis, magnetis, panas, sinar, dan kimia menjadi tegangan dan arus listrik. Sensor berfungsi mengubah besaran fisik (misalnya : temperature, gaya, kecepatan putaran) menjadi besaran listrik.

2.7.1 Sensor pH Air

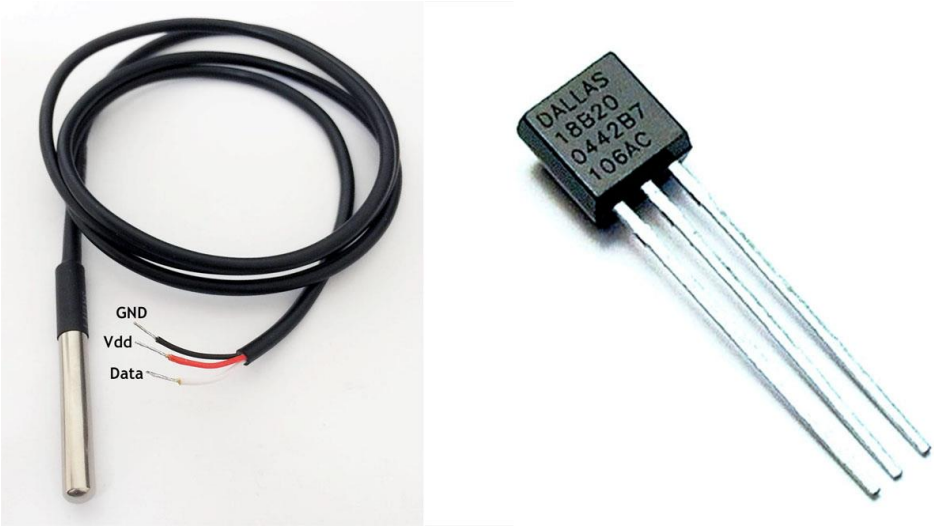
Sensor pH adalah sensor untuk mengukur ke asaman atau basa dari larutan air. Prinsip kerja utama pH meter adalah terletak pada sensor probe berupa

elektrode kaca (glass electrode) dengan jalan mengukur jumlah ion H_3O^+ di dalam larutan.[18]

Gambar 2.19 Sensor pH[18]

2.7.2 Sensor Suhu DS18B20

DS18B20 adalah sensor suhu digital seri terbaru dari maxim IC. Sensor ini mampu membaca suhu dengan ketelitian 9 hinggal 12-bit, rentang -55ºC hinggal 125ºC dengan ketelitian (+/- 0.5ºC).[19]



Gambar 2.20 Sensor Suhu DS18B20[19]