BAB 2
TINJAUAN PUSTAKA

2.1 Tempat Penelitian

Balai Pengembangan Benih Ikan Air Tawar (BPBIAT) Purwakarta yang berlokasi di Jl. Cipulus, RT.06/03 Desa Nagrog, Kec. Wanasaya, Kab. Purwakarta.

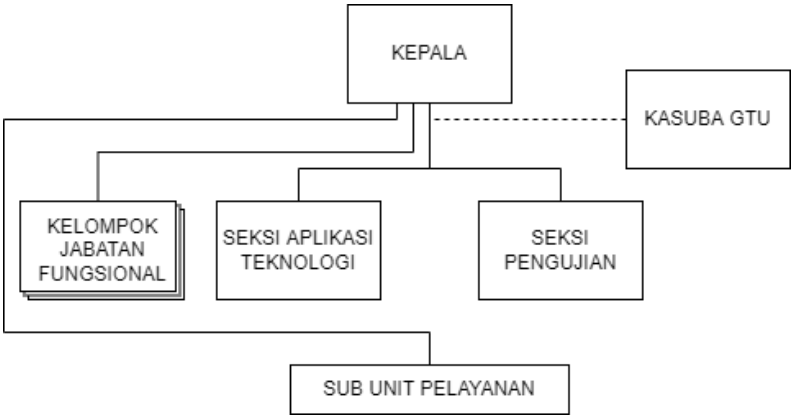
2.1.1 Visi dan Misi

Visi Balai Pemngembangan Benih Ikan Air Tawar (BPBIAT) adalah “Mewujudkan Balai Pemngembangan Benih Ikan Air Tawar (BPBIAT) menjadi tontonan, tuntunan, dan mandiri”.

Misi Balai Pemngembangan Benih Ikan Air Tawar (BPBIAT) adalah:

1. Meningkatkan kualitas dan kuantitas induk unggul ikan nila nirwana dan ikan mas.
2. Meningkatkan peran serta masyarakat dalam pengembangan kawasan pembenihan ikan nila nirwana dan ikan mas.
3. Mewujudkan ikan nila nirwana sebagai komoditas unggulan dan andalan perikanan nasional.

2.1.2 Struktur Organisasi



Gambar 2.1 Struktur Organisasi

2.2 Landasan Teori

Landasan teori ini berfungsi untuk mendukung dalam perancangan sebuah penelitian baik secara teori maupul hal-hal lain yang berkaitan tentang penelitian ini. Di dalam penelitian ini, akan membahas berbagai teori yang berkaitan dengan skripsi ini sebagai bentuk landasan dalam pembuatan penelitian ini.

2.2.1 Definisi Sistem

Pada dasarnya sistem adalah suatu hubungan dari kerangka prosedur-prosedur yang disusun dengan skema menyeluruh dengan tujuan melaksanakan tugas-tugasnya. Menurut Gordon B. Davis menyatakan bahwa sistem ialah sesuatu yang terdiri dari bagian-bagian yang saling berkaitan yang beroperasi bersama untuk mencapai beberapa suatu sasaran dan maksud.

Jadi, secara umum sistem adalah prosedur-prosedur yang saling berhubungan dengan tujuan yang sama dengan tugas yang dibuat sesuai kebutuhannya.

2.2.1.1 Sistem Monitoring

Sistem adalah kumpulan elemen yang berinteraksi untuk mencapai satu tujuan tertentu [10]. Sistem pada dasarnya adalah sekelompok unsur yang erat hubungannya satu dengan yang lain, yang berfungsi bersama-sama untuk mencapai tujuan tertentu [11]. Monitoring atau pengawasan adalah salah satu hal pokok dalam manajemen [12].

Sistem monitoring adalah kumpulan dari beberapa proses yang dijadikan satu kesatuan dan saling terintegerasi untuk dapat mengumpulkan data dari suatu kejadian dan menampilkan data tersebut secara langsung atau data yang sudah mengalami proses pengolahan.

2.2.2 Internet of Things

Internet of things atau yang biasa disebut IoT adalah struktur dimana objek yang disediakan dengan identitas eklsusif dan kemampuan untuk pindah data melalui jaringan tanpa memerlukan dua arah antara manusia ke manusia yaitu sumber ke tujuan atau interkasi manusia ke komputer.

Semakin berkembangnya keperluan manusia tentang teknologi maka semakin banyak penelitian yang hadir, dan *internet of things* menjadi sebuah bidang penelitian tersendiri semenjak berkembangnya teknologi internet, dan *internet of things* merupakan salah satu hasil pemikiran para peneliti yang mengoptimasi beberapa alat pintar seperti media sensor

dan alat pintar lainnya yang memungkinkan manusia untuk mudah berinteraksi dengan semua peralatan yang terhubung dengan jaringan internet [13].

2.2.3 Pengertian Rancang Bangun

1. Rancang

   Menurut Pressman (2009) Perancangan atau rancang merupakan serangkaian prosedur untuk meneterjemahkan hasil analisa dan sebuah sistem ke dalam bahasa pemograman untuk mendeskripsikan dengan detail bagaimana komponen-komponen diimplementasikan. Adapun tujuan dari perancangan ialah untuk memberi gambaran yang jelas lengkap kepada pemogram dan ahli teknik yang terlibat.

   Perancangan adalah sebuah proses untuk mendefinisikan sesuatu akan dikerjakan dengan menggunakan teknik yang bervariasi serta di dalamnya melibatkan dekripsi mengenai arsitektur serta detail komponen dan juga keterbatasan yang akan dialami dalam proses pengerjaannya.

2. Bangun

   Menurut Pressman (2009) pengertian pembangunan atau bangun sistem adalah kegiatan menciptakan sistem baru maupun mengganti atau memperbaiki sistem yang telah ada secara kesuluruhan.

Jadi dapat disimpulkan bahwa rancang bangun adalah penggambaran, perancanaan atau pengaturan dari beberapa elemen yang terpisah menjadi satu kesatuan yang utuh dan berfungsi. Dengan demikian pengertian rancang bangun adalah kegiatan yang meneterjemahkan hasil sebuah analisa ke dalam bentuk perangkat dengan sistem yang sudah ada atau memperbaiki sisyem yang ada.

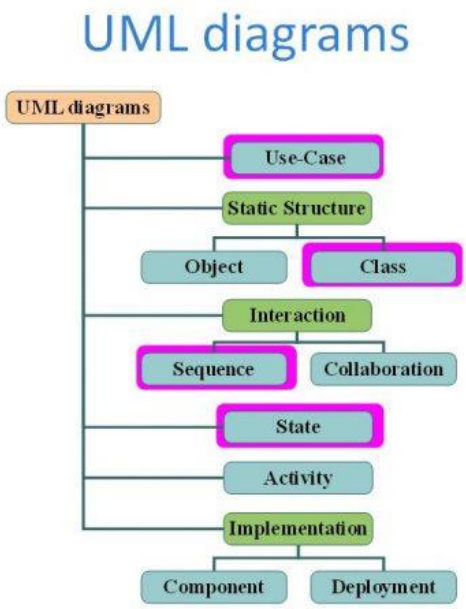
2.2.4 Irigasi

Menurut Kartasapoetra (2004), irigasai merupakan kegiatan penyediaan dan pengaturan air untuk memenuhi kepentingan dengan memanfaatkan air permukaan dan tanah. Jadi irigasi adalah suatu sistem yang mengatur tentang penyediaan air untuk keperluan usaha tani seperti pertanian, perikanan, perekebunan dan sebagainya.

2.2.5 *Unifed Modelling Language*

Unifed Modeling language (UML) adalah satu alat bantu yang sangat berguna di area pengembangan sistem berorientasi objek, karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didomniasi oleh objek dan digambarkan atau dinotasikan dalam

simbol-simbol yang cukup spesifik maka orientasi objek memiliki proses standard dan bersifat independen. UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat rancangan skema yang efektif untuk berbagi dan mengkomunikasikan rancangan tersebut dengan yang lain.



Gambar 2.2 Diagram UML

Ada 4 (empat) prinip dasar dari pemograman berorintasi objek yang menjadi dasar kemunculan UML, yaitu abstraksi, enkapsulasi, modularitas, dan hirarki [14].

1. Abstraksi memfokuskan perhatian pada karakteristik objek yang paling penting dan paling dominan yang bisa digunakan untuk membedakan objek tersebut dari objek lainnya.
2. Ekapsulasi menyembunyikan banyak hal yang terdapat dalam objek yang tidak perlu diketahui oleh objek lain. Dalam praktek pemrograman, enkapsulasi diwujudkan dengan membuat suatu kelas *interface* yang akan dipanggil oleh objek lain, sementara didalam objek yang dipanggil terdapat kelas lain yang mengimplementasikan apa yang terdapat dalam kelas *inreface*.
3. Modularitas membagi sistem yang rumit menjadi bagian-bagian yang lebih kecil yang bisa mempermudah *developer* memahami dan mengelola objek tersebut.
4. Hirarki berhubungan dengan abtraksi dan modularitas, yaitu pembagian berdasarkan urutan dan pengelompokkan tertentu. Misalnya untuk menentukan objek mana yang berada pada kelompok yang sama, objek mana yang merupakan

komponen dari objek yang memiliki hirarki yang lebih tinggi. Semakin rendah hirarki objek berarti semakin jauh abstraksi dilakukan terhadap suatu objek.

UML memiliki sperangkat aturan dan notasi dalam bentuk grafis yang cukup spesifik sehingga para pakar merasa lebih mudah dalam menganalisa dan mendesain atau memodelkan suatu sistem (Sugue J. 2009).

Komponen atau notasi UML diturunkan dari 3 (tiga) notasi yang telah ada sebelumnya yaitu Grady Booch *Object-Oriented Design (OOD),* Jim Rumbaugh *Object Modelling Technique (OMT),* Ivar Jacobson *Object-Oriented Software Engineering (OOSE).*

UML terdiri atas 3 (tiga) kategori dan memiliki 13 (tigabelas) jenis diagram, diantaranya:

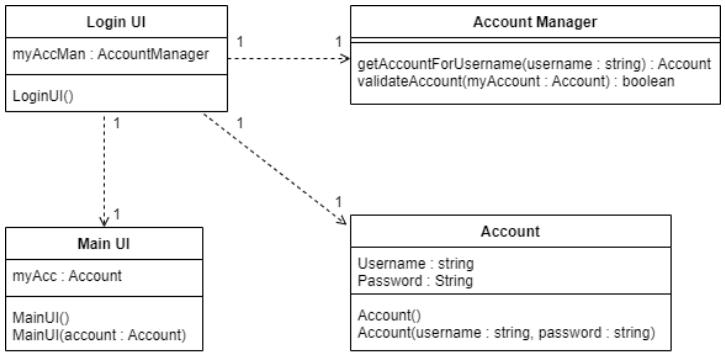
1. **Struktur Diagram**

   Menggambarkan elemen dari spesifikasi dimulai dengan kelas, objek, dan hubungan mereka, dan beralih ke dokumen arsitektur logis dari suatu sistem. Struktur diagram dalam UML terdiri atas:

   1) *Class Diagram*

      Class Diagram menggambarkan struktur statis dari kelas dalam sistem dan menggambarkan atribut, operasi dan hubungan antar kelas. *Class Diagram* membantu dalam menvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. *Class Diagram* memiliki 3 (tiga) area pokok:

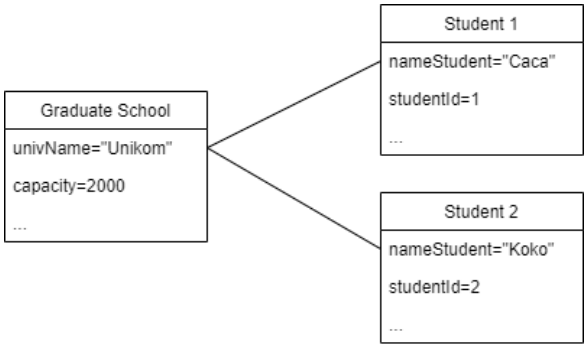
      1. Nama
      2. Atribut
      3. Metoda



Gambar 2.3 Notasi *Class Diagram*

2) *Object Diagram*

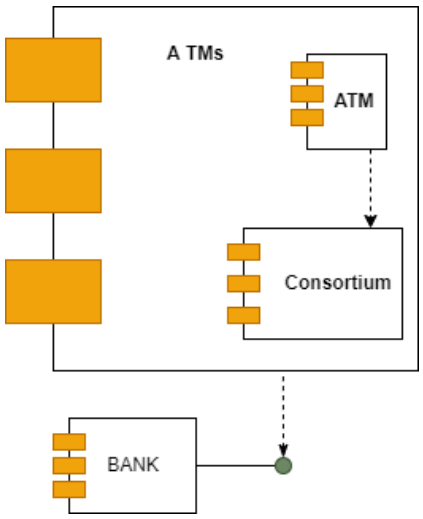
Object Diagram menggambarkan kejelasan kelas dan warisan dan kadang-kadang diambil ketika merencanakan kelas, atau untuk membantu pemamangku kepentingan non-program yang mungkin menemukan diagram kelas terlalu abstrak.



Gambar 2.4 Notasi *Object Diagram*

3) *Component Diagram*

Component Diagram menggambarkan struktur fisik dari kode, pemetaan pandangan logis dari kelas proyek untuk kode aktual dimana logika ini dilaksanakan.

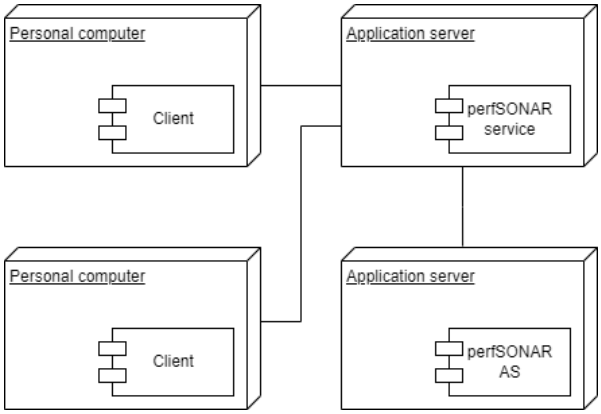


Gambar 2.5 Notasi *Component Diagram*

4) *Deployment Diagram*

Deployment Diagram memberikan gambaran dari arsitektur fisik perangkat lunak, perangkat keras, dan artefak dari sistem. *Deployment diagram* dapat dianggap sebagai ujung spektrum dari kasus penggunaan, menggambrakan bentuk fisik dari

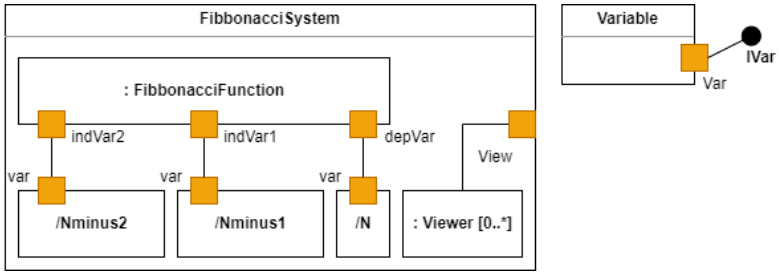
sistem yang bertentangan dengan gambar konseptual dari pengguna dan perangkat berinteraksi dengan sistem.



Gambar 2.6 Notasi *Deployment Diagram*

5) *Composite Structure Diagram*

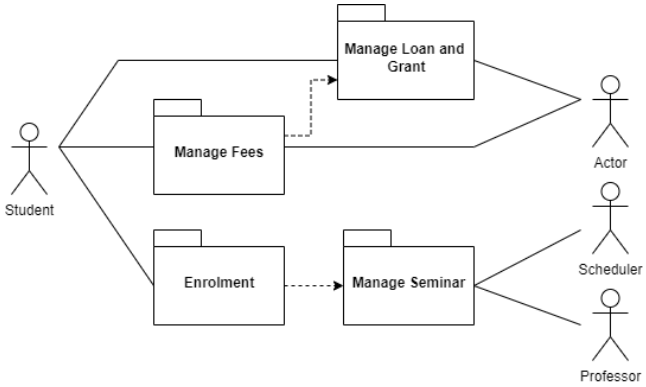
Composite Structure Diagram sebuah diaram struktur komposit mirip dengan diagram kelas, tetapi menggambarkan bagian individu, bukan seluruh kelas. Kita dapat menambahkan konektor untuk menghubungkan dua atau lebih bagian dalam atau ketergantungan hubungan asosiasi.



Gambar 2.7 Notasi *Composite Structure Diagram*

6) *Package Diagram*

Package Diagram biasanya digunakan untuk menggambarkan tingkat organisai yang tinggi dari suatu proyek *software*. Atau dengan kata lain untuk menghasilkan diagram ketergantungan paket untuk setiap paket dalam *Tree Model*.



Gambar 2.8 Notasi *Package Diagram*

2. Behavior Diagram

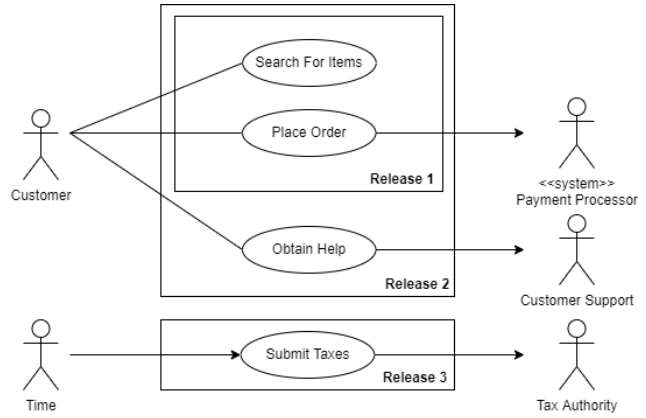
Menggambarkan ciri-ciri *behavior* atau metode fungsi dari sebuah sistem atau *business process*. *Behavior Diagram* dalam UML terdiri atas:

1) *Use Case Diagram*

Use Case Diagram menggambarkan *actor*, *use case*, dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk actor. Sebuhan *use case* digambarkan sebagai elips hosrizontal dalam suatu diagram UML *use case*.

Use Case memiliki 2 (dua) istilah:

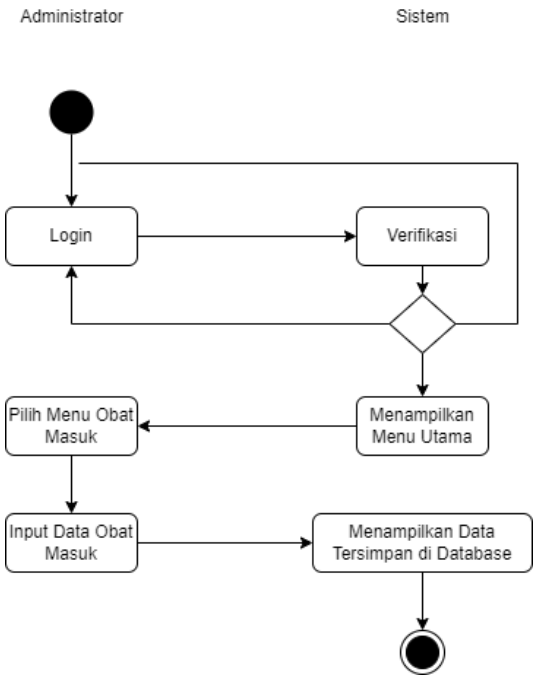
1. *System use case,* interaksi dengan sistem.
2. *Business use case,* interaksi bisbis dengan konsumen atau kejadian nyata.



Gambar 2.9 Notasi *Use Case Diagram*

2) *Activity Diagram*

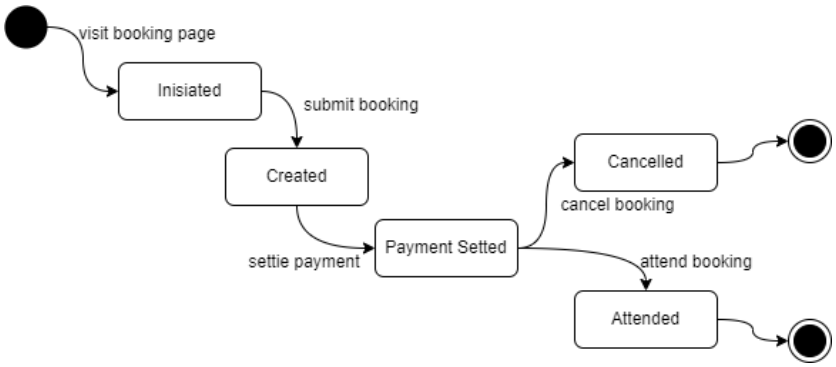
Activity Diagram menggambarkan aktifitas-akifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas.



Gambar 2.10 Notasi *Activity Diagram*

3) *State Machine Diagram*

State Machine Diagram menggambarkan *state,* dan *event.*



Gambar 2.11 Notasi *State Machine Diagram*

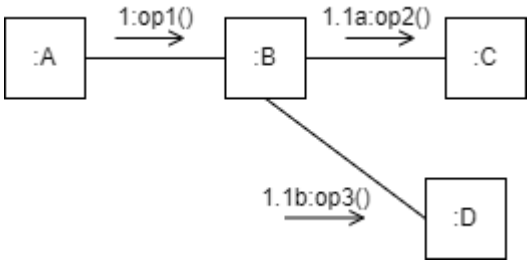
3. Interaction Diagram

Bagian dari *behavior diagram* yang menggambarkan interaksi objek. *Interaction diagram* dalam UML terdiri atas:

1) *Communicaton Diagram*

Communicaton Diagram serupa dengan *sequence diagram*, tetapi diagram ini juga digunakan untuk memodelkan perilaku dinamis dari *use case*. Bila dibandingkan

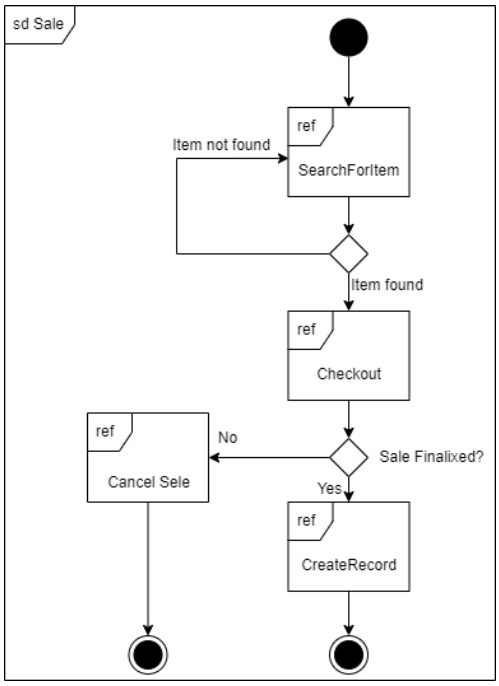
dengan *sequence diagram*, diagram ini lebih terfokus pada menampilkan kolaborasi benda dari pada urutan waktu.



Gambar 2.12 Notasi *Communication Diagram*

2) *Interaction Overview Diagram*

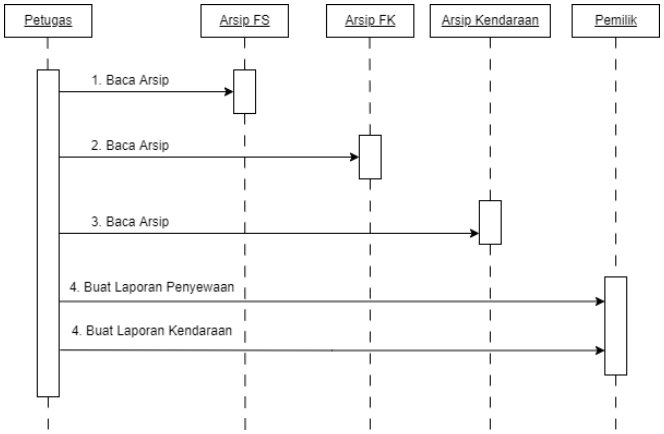
Interaction Overview Diagram berfokus pada gambaran aliran kendali interaksi dimana node adalah interaksi atau kejadian interaksi.



Gambar 2.13 Notasi *Interaction Overview Diagram*

3) *Sequence Diagram*

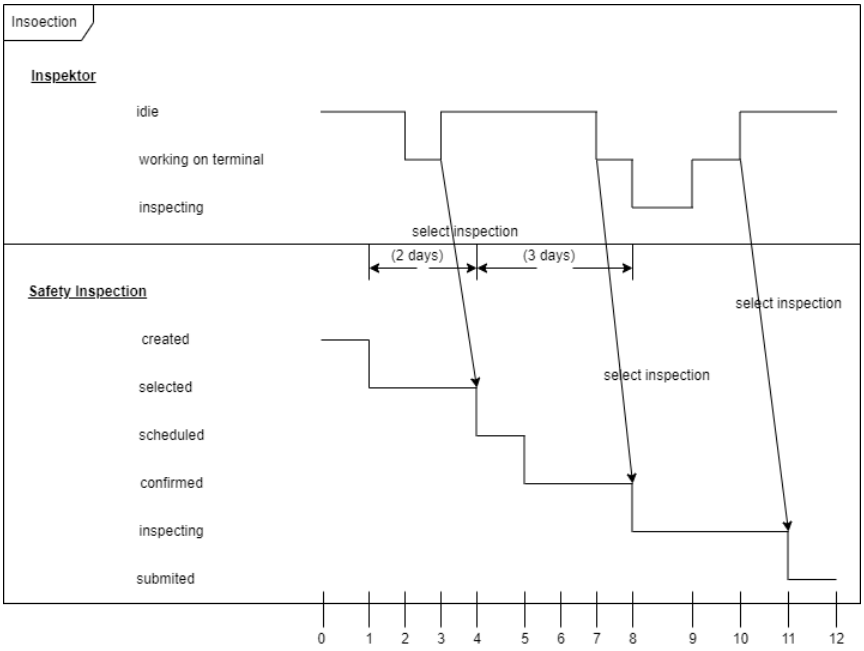
Sequence Diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram*.



Gambar 2.14 Notasi *Sequence Diagram*

4) *Timing Diagram*

Timing Diagram didasarkan pada diagram waktu *hardware* awalnya dikembangkan oleh insinyur listrik.

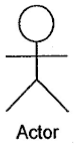


Gambar 2.15 Notasi *Timing Diagram*

Untuk menggambarkan analisa dan desain diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu sruktur diagram, *behavior* diagram, dan *interaction* diagram. Berikut beberapa notasi dalam UML diantaranya:

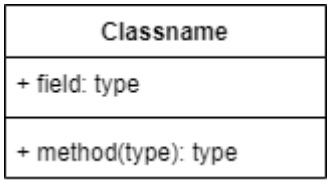
1. *Actor*, menentukan peran yang dimainkan oleh user atau sistem lain yang berinteraksi dengan subjek. Aktor adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya.

Tugas aktor adalah memberikan informasi kepada sistem dan dapat memerintahkan sistem untuk melakukan sesuatu tugas.



Gambar 2.16 Notasi *Actor*

2. *Class Diagram*, notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu *class* beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari sistem berorientasi objek.



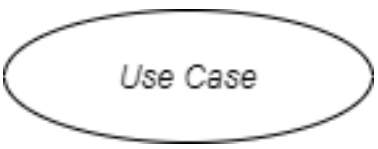
Gambar 2.17 Notasi *Class*

3. *Use Case* dan *Use Case Spesification*

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

   Use case merupakan awal yang sangat baik untuk setiap fase pengembangan berbasis objek, *design, testing*, dan dokumentasi yang menggambarkan kebutuhan sistem dari sudut pandang di luar sistem.

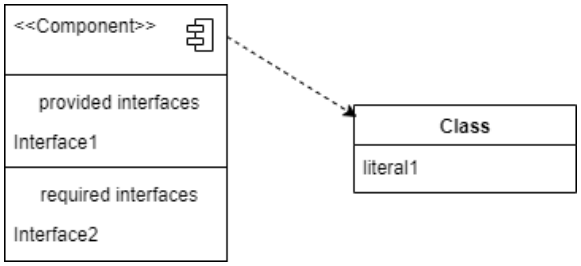
   Perlu diingat bahwa *use case* hanya menetapkan apa yang seharusnya dikerjakan oleh sistem, yaitu kebutuhan fungsional sistem dan tidak untuk menentukan kebutuhan non-fungsional, misalnya: sasaran kinerja, bahasa pemrograman dan lain sebagainya.



Gambar 2.18 Notasi *Use Case*

4. *Realization*

   Realization menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.



Gambar 2.19 Notasi *Realization*

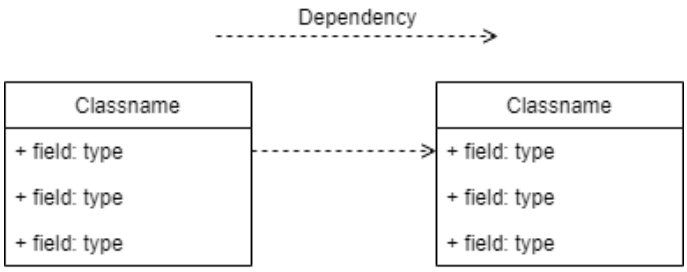
5. *Interaction*

   Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar objek maupun hubungan antar objek.

Gambar 2.20 Notasi *Interaction*

6. *Dependency*

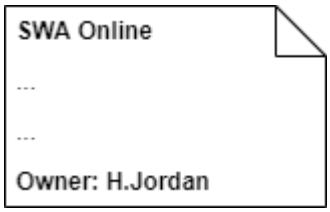
   Dependency merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan ke dalam elemen yang ada di garis dengan panah.



Gambar 2.21 Notasi *Dependency*

7. *Note*

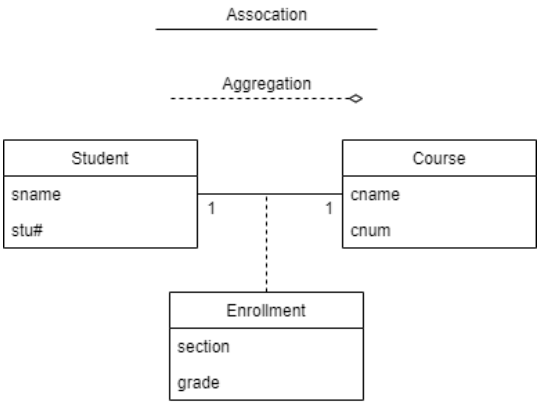
Note digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.



Gambar 2.22 Notasi *Note*

8. *Association*

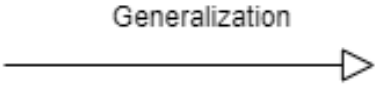
   Association menggambarkan navigasi antar *class* (*navigation*), berapa banyak objek lain yang bisa berhubungan dengan satu objek (*multiplicity* antar *class*) dan apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*).



Gambar 2.23 Notasi *Assocation*

9. *Generalization*

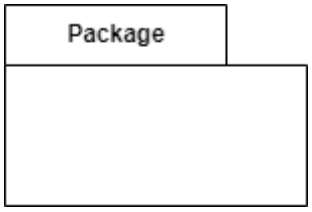
   Generalization menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik.



Gambar 2.24 Notasi *Generalization*

10. *Package*

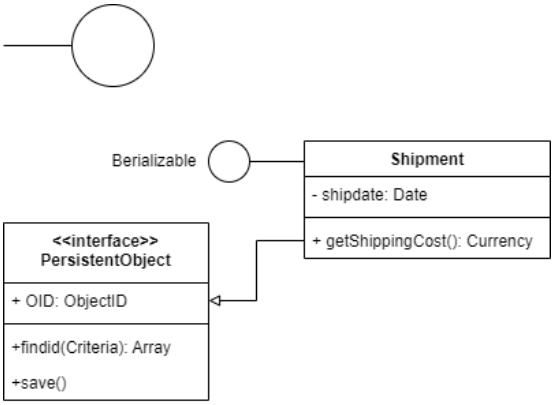
    Package adalah mekanisme pengelompokkan yang dgunakan untuk menandakan pengelompokkan elemen-elemen model.



Gambar 2.25 Notasi *Package*

11. *Interface*

    Interface merupakan kumpulan operasi berupa implementasi dari suatu *class*. Atau dengan kata lain implementasi operasi dalam *interface* dijabarkan oleh operasi di dalam *class*.



Gambar 2.26 Notasi *Interface*

2.2.6 *Object Oriented*

Object oriented adalah sebuah objek memiliki keadaan sesat *(state)* dan perilaku *(behaviour). State* sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam *attribute/properties*. Sedangkan perilaku suatu objek mendefinisikan bagaimana sebuah objek bertindak atau beraksi dan memberikan reaksi. Perilaku sebuah objek dinyatakan dalam *operation*. Menurut Schmuller *attribute* dan *operation* bila disatukan akan memberikan *features.* Himpuan objek-objek yang sejenis disebut *class*. Objek adalah contoh *instance* dari sebuah *class.* Tidak semua aplikasi yang akan dibangun cocok menggunakan *object oriented,* seperti pembangunan aplikasi yang sangat berorientasi ke *database,* karena akan banyak kehilangan manfaat dari penggunaan RDBMS *(Relation 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. Dan juga aplikasi yang membutuhkan banyak algortima. Beberapa aplikasi yang melibatkan perhitungan yang besar dan kompleks [15]. Pada penelitian ini konsep OO

digunakan untuk konsep pengkodean pada sistem yang akan dibangun [16]. Berikut ini adalah konsep-konsep dalam pemograman berorientasi objek:

1. ***Class***

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

2. ***Object***

   Object 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 pengemban perangkat lunak. Orientasi objek merupaka teknik pemodelan sistem rill 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 laporan serta berkomunikasi dengan objek lain, tanpa harus menampakkan kelebihan diterapakan.

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 laiinya 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 implementas untuk sebagian fungsi tertentu.

6. ***Inheritance***

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

2.2.7 JavaScript Object Notation

Format data *JavaScript Object Notation* (JSON) memungkinkan aplikasi untuk berkomunikasi melalui jaringan, biasanya melalui RESTful APIs. JSON adalah teknologi agnostik, *nonproprietary*, dan *portable*. Semua bahasa modern dan *platform* memberikan dukungan yang sangat baik untuk memproduksi (membuat serial) dan mengosumsi data JSON *(deserializing)*. JSON sederhana: terdiri dari kontruksi ramah-pengembang seperti *object, array*, dan pasangan nama / nilai. JSON tidak terbatas pada *Representational State Transfer* (REST).

RESTful Web *Services* sebagai tidak standar, tetapi (seperti HTTP) JSON sebenarnya merupakan standar. Baik *Internet Engineering Task Force* (IETF) dan *Ecma International* (sebelumnya *European Computer Manufactures Association* (ECMA)) telah mengakui JSON sebagai standar. Douglas Crockford awalnya menciptakan JSON pada tahun 2001, dan awalnya melakukannya pada tahun 2006 dibawah RFC 4627 melalui IETF; lihat spesifikasi JSON. Pada musim gugur 2013, Ecma International juga menstandarisasi JSON dibawah Ecma. Dengan pengakuan Ecma, JSON sekarang dianggap sebagau sumber pemrosesan data internasional formal [17]. Pada penelitian ini JSON digunakan digunakan sebagai API untuk mengirim data sensor dari Arduino ke aplikasi web.

2.2.8 MySQL

MySQL adalah sebuah perangkat lunak sisem manajamen basis data SQL *(database management system)* atau DBMS yang *multithread, multi-user,* dengan sekitar 6 juta instalasi diseluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunanya tidak cocok dengan penggunaan GPL [18]. Pada penelitan ini MySQL digunakan sebagai platform penyimpanan data dan

implementasi model rancangan *database* yang sudah di buat. Gambar 2.27 menunjukan logo MySQL.

Gambar 2.27 Logo MySQL

MySQL memiliki banyak fitur seperti sebagai berikut:

1. *Relational Database System*, seperti halnya *software database* lain yang ada di pasaran, MySQL termasuk RDBMS.
2. *Arsitektur Client-Server,* MySQL memiliki arsitektur *client-server* dimana sever *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. *Performance Tuning*, MySQL mempunyai kecepatan uang cukup baik dalam menangani *query-query* sederhana, serta mampu memproses lebih banyak SQL persatuan waktu.
5. *Sub Select.*
6. *Views.*
7. *Stored Proesedured.*
8. *Triggers.*
9. *Replication.*
10. *Foreign Key.*
11. *Security* yang baik.

2.2.9 Web Server

Web server adalah sebuah *software* yang memberikan layanan berbasis data dan berfungsi menerima permintaan dari HTTP atau HTTPS pada klien web server menunggu permintaan dari klien yang menggunakan browser seperti Netscape Navigator, Internet

Expoler, Mozilla, dan program browser lainnya. Jika permintaan dari browser, maka web server akan memproses permintaan itu kemudian memberikan hasil prosesnya berupa data yang diinginkan kembali ke browser. Data ini mempunyai format yang standar, disebut dengan format *Standard General Markup Language* (SGML). Data yang berupa format ini kemudian akan ditampilkan oleh browser sesuai dengan kemampuan browser tersebut.

Web server berfungsi untuk mentransfer berkas melalui protokol komunikasi yang telah ditentukan atas permintaan pengguna. Berkas yang ditransfer dapat berupa teks, gambar, video, dan lainnya yang merupakan elemen sebuah web. Web server saat ini berfungsi pula untuk menjalankan program-program yang memang dirancang untuk berjalan di web server. Bahasa-bahasa tersebut ialah seperti PHP atau ASP. Sehingga web server juga dapat melakukan pengolahan data yang diberikan oleh pengguna. Fitur ini biasa disebut dengan *server site scripting.* Berikut gambar 2.28 menunjukan alur web server.

Gambar 2.28 Web Server

Beberapa jenis web server diantaranya adalah:

1. Apache Web Server / The HTTP Web Server
2. Apache Tomcat
3. Microsoft Windows Server 2008 IIS *(Internet Information Services)*
4. Lighttpd
5. Zeus Web Server
6. Sun Java System Web Server

2.2.10 API *(Application Programming Interface)*

API *(Application Programming Interface)* merupakan sebuah *interface* yang dapat diimplementasikan dengan menggunakan perangkat lunak *(software)* sehingga perangkat lunak tersebut dapat berinteraksi dengan perangkat lunak lainnya, seperti halnya tampilan *interface user* yang memungkinkan *user* untuk berinteraksi dengan komputer (Prasetiadi,

2011). Dengan memanfaatkan API, *developer* dapat memanfaatkan beberapa perangkat lunak untuk melakukan suatu proses. Selain itu, tujuan dari penggunaan API adalah mempercepat proses pengembangan sebuah sistem atau aplikasi dengan menggunakan fungsi-fungsi secara terpisah, sehingga *developer* tidak perlu membuat fungsi atau fitur yang serupa. Cara kerja API dapat dilihat pada gambar 2.29.

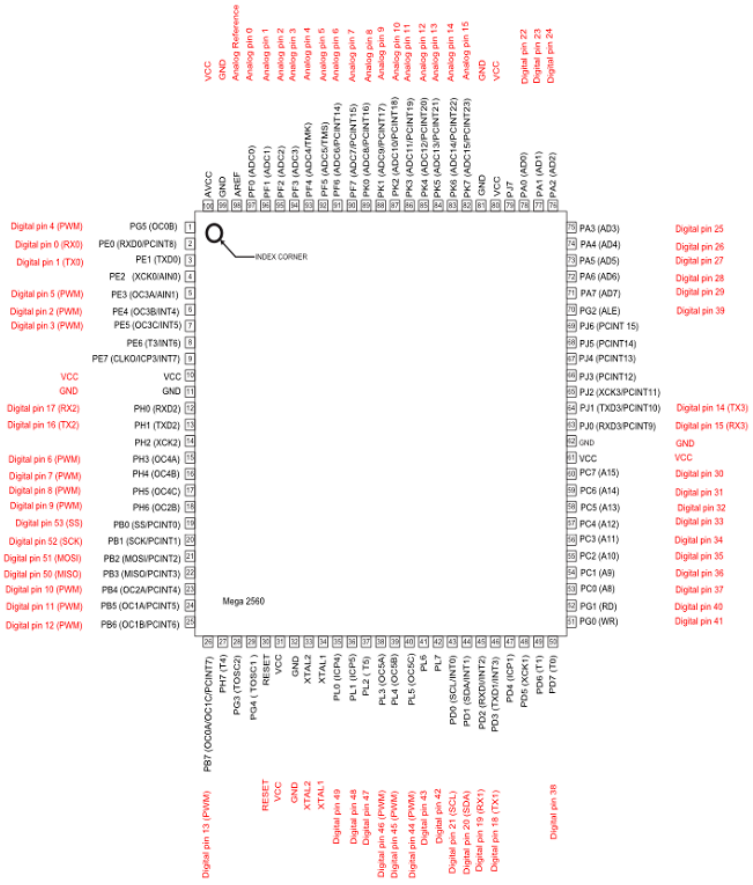
Gambar 2.29 Cara Kerja API

API bekerja dengan cara membantu aplikasi berinteraksi dengan *library* dengan mengikuti serangkaian aturan yang ditentukan sebelumnya oleh API itu sendiri. Pendekatan ini memudahkan *developer* untuk membuat aplikasi yang berkomunikasi dengan berbagai *library* tanpa harus memikirkan kembali strategi yang digunakan selama semua *library* mengikut API yang sama. Kelebihan lain dari metode ini menunjukkan betapa mudahnya menggunakan *library* yang sama dengan bahasa pemrograman yang berbeda. Dalam penelitaian ini API digunakan dalam pertukaran data antara Arduino dengan *database*.

2.2.11 Arduino Mega 2560

Arduino Mega 2560 adalah papan mikrokontroler yang berbasis pada ATmega2560. Memiliki 54 pin *input*/*output* digital, 15 dapat digunakan sebagai *output* PWM, 16 *input* analog, 4 UART (port serial perangkat keras), 16 MHz kristal osilator, koneksi USB, *jack power*, *header* ICSP, dan tombol *reset*. Itu semua dibutuhkan untuk mendukung mikrikontroler, untuk mulai mengktifkan cukup dengan menghubungkan power dari USB ke komputer atau dengan adaptor AC – DC ke *jack* DC. Arduino Mega 2560 juga kompatibel dengan sebagian besar *shield* yang di rancang untuk Arduino *Deumilanove* atau *Diecimila*.

Gambar 2.30 Arduino Mega 2560



Gambar 2.31 Pin Arduino Mega 2560

2.2.12 NodeMCU ESP8266

NodeMCU merupakan papan pengembangan produk *Internet of Things* (IoT) yang berbasiskan *Firmware* eLua dan *y* (SoC) ESP8266-12E. ESP8266 sendiri merupakan *chip* WiFi dengan protokol *stack* TCP/IP yang lengkap.

NodeMCU dapat dianalogikan sebagai papan arduinonya ESP8266. Program ESP8266 sedikit susah karena diperlukan beberapa teknik *wiring* serta tambahan modul USB ke serial untuk mengunduh program. Namun NodeMCU telah me-*package* ESP8266 ke

dalam sebuah board yang kompak dengan berbagai fitur layaknya mikrokontroler kapabilitas akses terhadap WiFi juga *chip* komunikasi USB ke serial. Sehingga untuk memprogramnya hanya diperlukan ekstensi kabel data USB persis yang digunakan *charging smarphone*.

Gambar 2.32 NodeMCU ESP8266

Gambar diatas merupakan kaki pin yang ada pada NodeMCU ESP8266, berikut penjelasan dari pin-pin tersebut:

1. ADC : Analog Digital Converter. Rentang tegangan masukan 0- 1v, dengan skup nilai digital 0-1024.
2. RST : berfungsi mereset modul
3. EN : Chip Enable, Active High
4. IO16 : GPIO16, dapat digunakan untuk membangunkan chipset dari mode deep sleep
5. IO14 : GPIO14; HSPI_CLK
6. IO12 : GPIO12: HSPI_MISO
7. IO13 : GPIO13; HSPI_MOSI; UART0_CTS
8. VCC : Catu daya 3.3V (VDD)
9. CS0 : Chip selection
10. MISO : Slave output, Main input.
11. IO9 : GPIO9
12. IO10 : GPIO10
13. MOSI : Main output slave input
14. SCLK : Clock
15. GND : Ground

16. IO15 : GPIO15; MTDO; HSPICS; UART0_RTS
17. IO2 : GPIO2;UART1_TXD
18. IO0 : GPIO0
19. IO4 : GPIO4
20. IO5 : GPIO5
21. RXD : UART0_RXD; GPIO3
22. TXD : UART0_TXD; GPIO1

2.2.13 Sensor Ultrasonic JSN-SR04T Waterproof

JSN-SR04T merupakan sensor ultrasonik yang dilengkapi dengan waterproof dan memiliki jangkauan pengukuran sebesar 0 sampai 500 cm. Sensor ini beroperasi pada tegangan antara 3V sampai 5V.

Sensor ini bekerja dengan mengirimkan gelombang suara jika terdapat objek depannya maka gelombang ini akan dipantulkan kembali oleh objek tersebut. Untuk mendapatkan nilai jarak antara sensor dengan objek yang terdeteksi perlu untuk menghitung berapa lama waktu antara mengirim dan menerima gelombang suara, dengan rumus sebagai berikut:

Jarak = (Waktu sinyal *high*) * kecepatan suara (340m/s) / 2

Gambar 2.31 dibawah menunjukkan bentuk fisik dari sensor Ultrasonic JSN-SR04T Waterproof.

Gambar 2.33 Sensor Ultrasonic JSN-SR04T Waterproof

2.2.14 Sensor YF-S201 Water Flow

. Water Flow Sensor merupakan sensor yang dapat mengukur debit air yang melalui suatu ruangan terutup. Pada umumnya sensor ini diimplementasikan pada pipa keluaran maupun masukan dalam sebuah jaringan distribusi air, hal ini dikarenakan sensor dapat membantu pengguna dalam melakukan pengukuran debit yang melalui suatu pipa untuk selanjutnya dilakukan pendataan maupun sebatas pemantauan. Salah satu varian yang cukup banyak digunakan adalah water flow sensor tipe YF-S201 seperti pada Gambar 2.34. Sensor ini dipilih karena sudah kompatibel dengan mikrokontroler keluarga AVR dan mudah diimpelementasikan. Sensor ini bekerja dengan cara menghitung jumlah air yang masuk dan keluar melalui katup outlet perbandingan dengan kecepatan pengisian pada tangki air untuk mendapatkan jumlah air yang masuk. Air yang melewati katup akan membuat rotor magnet dengan kecepatan tertentu sesuai dengan air yang mengalir, selanjutnya medan magnet pada rotor akan memberikan efek pada sensor dan menghasilkan sinyal pulsa atau yang biasa dikenal dengan PWM (Pulse Width Modulation). Keluaran dari sinyal pulsa ini akan diolah oleh

Gambar 2.34 Sensor YF-S201 Water Flow

2.2.15 Sensor Raindrop FR-04

Sensor raindrop atau hujan adalah sensor yang difungsikan untuk medeteksi ada tidaknya kondisi rinik hujan. Prinsip kerja modul sensor ini yaitu pada saat air hujan turun

dan mengenai panel sensor maka akan terjadi proses *elektrolisasi* oleh air hujan. Dan karena air hujan termasuk dalam golongan cairan *elektrolit* yang dimana cairan tersebut akan menghantarkan arus listrik. Pada sensor hujan ini terdapat *ic* komparator yang dimana *output* dari sensor ini berupa logika *high* dan *low* (*on* atau *off*). Serta pada modul ini terdapat terdapat *output* yang berupa tegangan pula.

Gambar 2.35 Sensor Raindrop FR-04

2.2.16 Modul Relay

Modul Relay adalah komponen elektronika berupa saklar elektronik yang digerakkan oleh arus listrik. Secara prinsip, relay merupakan tuas saklar dengan lilitan kawat pada batang besi (solenoid) di dekatnya. Ketika solenoid dialiri arus listrik, tuas akan tertarik karena adanya gaya magnet yang terjadi pada solenoid sehingga kontak saklar akan menutup. Pada saat arus dihentikan, gaya magnet akan hilang, tuas akan kembali ke posisi semula dan kontak saklar kembali terbuka. Relay biasanya digunakan untuk menggerakkan arus / tegangan yang besar (misalnya peralatan listrik 4 A / AC 220 V) dengan memakai arus / tegangan yang kecil misalnya 0.1 A / 12 Volt DC) [20].

Gambar 2.36 Modul Relay 2 *Channel*

Relay adalah saklar *(switch)* yang dioperasikan secara listrik dan merupakan komponen *Electromechanical* yang terdiri dari 2 bagian utama yakni Elektromagnet *(Coil)* dan Mekanikal (seperangkat Kontak Saklar/*Switch*) [21]. Relay menggunakan Prinsip Elektromagnetik 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 [22].

Pada dasarnya relay terdiri dari 4 komponen dasar, yaitu:

1. *Electromagnet (Coil)*
2. *Armature*
3. *Switch Contact Point (Saklar)*
4. *Spring*

2.2.17 Motor DC

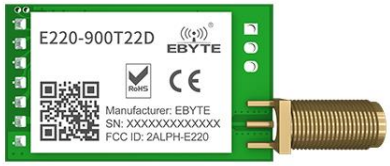
Motor DC menggunakan Silicon Controller Rectifier untuk memfasilitasi kontrol kecepatan pada motor. Pada motor listrik tersebut terjadi proses konversi energi dari energi listrik menjadi energi mekanik. Motor DC itu sendiri memerlukan suplai tegangan searah dari kumparan jangkar sebagai kumparan medan magnit untuk diubah menjadi energi mekanik. Pada motor DC, kumparan medan magnit disebut stator (bagian yang tidak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar). Jika tejadi putaran pada kumparan jangkar dalam pada medan magnit, maka akan timbul tagangan (GGL = Caya Gerak Listrik) yang berubah-ubah arah pada setiap setengah putaran, sehingga merupakan tegangan arus bolakbalik [23].

Gambar 2.37 Motor DC

2.2.18 Modul LoRa EBYTE E220-900T22D

E220-900T22D mengadopsi generasi baru teknologi spread spectrum LoRa dan modul port serial nirkabel (UART) yang dirancang berdasarkan skema chip LLCC68. Ini

memiliki berbagai metode transmisi, bekerja di pita frekuensi (850.125 - 930.125MHz) (default 873.125MHz), output level TTL, kompatibel dengan tegangan port 3.3V dan 5V IO.



Gambar 2.38 Modul EBYTE E220-900T22D

2.2.19 AT-Command

Modul *wireless* ESP8266 yang digunakan pada penelitian ini memiliki *firmware* bawaan pabrik yang mendukung perintah AT-Command. Sekumpulan daftar dari Hayes command merupakan deskripsi dari AT-Command. Hayes command dikembangkan oleh Dennis Hayes pada tahun 1981 sebagai daftar perintah untuk melakukan konfigurasi modem dengan menggunakan jalur serial *interface* [19]. Berikut ini contoh beberapa perintah ATCommand berserta fungsinya pada modul ESP8266.

Tabel 2.1 Daftar AT-Command

AT-Command	Function	Response
AT	Working	OK
AT+RST	Restart	OK [System Ready, Vendor:www.ai-thinker.com]
AT+GMR	*Firmware* Version	AT+GMR 0018000902 OK
AT+CWLAP	List Access Point	AT+CWLAP+CWLAP:(4,"AP1",-38,"70:62:b8:6f:6d:58",1)+CWLAP:(4,"AP2",-83,"f8:7b:8c:1e:7c:6d",1) OK
AT+CWJAP? AT+CWJAP=”SSID”,”PASS”	Join Access Point	Query AT+CWJAP?+CWJAP:"AP1" OK