BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam rancang bangunan purwarupa *smart plant protection* tanaman cabai 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 rancang bangun purwarupa *smart plant protection* tanaman cabai berbasis *internet of things* (IoT).

2.2 Cabai (*Capsicum spp*)

Tanaman cabai (*Capsicum spp*) merupakan salah satu jenis tanaman sayuran yang prospeknya sangat baik untuk dikembangkan sebagai tanaman utama karena mempunyai nilai ekonomis yang tinggi. Buah cabai bermanfaat antara lain sebagai penyedap masakan, penambah selera makan. Tanaman ini juga dibudidayakan oleh masyarakat Indonesia, dan menunjang gizi masyarakat[7].

Dilihat dari kandungan gizi dan manfaat yang dimiliki cabai, maka cabai juga penting dikomsumsi oleh manusia. Untuk itu pembudidayaan tanaman cabai harus diperhatikan agar produksi tanaman cabai meningkat dari tahun ketahun. Dalam satu periode tanam, cabai dapat dipanen beberapa kali bila musim dan perawatannya baik dapat dipanen 15-17 kali, namun umumnya sebanyak 10-12 kali. Perawatan tanaman cabai lebih rumit dibandingkan dengan perawatan tanaman hortikultura lainnya, sehingga biaya perawatannya lebih mahal, rendahnya produksi juga dapat membuat harga cabai meningkat [7].

Rendahnya produksi cabai salah satunya disebabkan oleh adanya serangan hama dan penyakit karena dapat menyebabkan kerugian baik kualitas maupun kuantitas cabai. Salah satu penyakit yang mempengaruhi produksi tanaman cabai di Indonesia adalah penyakit virus yang menyerang cabai yaitu virus kuning dan virus keriting [7]. Virus dapat mempunyai bermacam-macam pengaruh terhadap tumbuhan, karena virus mempunyai daya tular yang tinggi.

2.2.1 Jenis Jenis Tanaman Cabai

Cabai adalah salah satu jenis sayuran yang disukai, setiap jenis cabai mempunyai keistimewaan tersendiri misalnya, cabai rawit dan cabai merah keriting rasanya lebih pedas dibandingkan jenis cabai lainnya, cabai merah besar rasanya agak manis tetapi kurang pedas, paprika rasanya tidak pedas sama sekali, tetapi penggemarnya kian hari semakin banyak[8]. Karena masyarakat di indonesia rata-rata menyukai rasa pedas, sehingga harga jual sangat tinggi untuk tanaman cabai tersebut.

Menurut Ir Nur Tjahjadi, jenis-jenis tanaman cabai antara lain [8]:

1. Cabai Rawit (*Capsicum Frustescens L.*)

   Cabai rawit terdiri dari tiga jenis, yakni:

   a. Cabai jemprit

      Bentuk buah kecil, pendek, dan ada yang bulat. Panjangnya 1-2 cm dan lebarnya 0,5-1 cm [8]. Cabai jemprit ini sangat tahan terhadap bakteri yang menyerang tanaman tersebut, serta sangat cocok untuk di tanaman di dataran rendah ataupun dataran tinggi.

   b. Cabai ceplik

      Bentuk buah besar dan gemuk, lebih besar daripada cabai jemprit. Panjangnya 3-4 cm, lebarnya 1-1,5 cm [8]. Cabai ceplik lebih cocok untuk ditanam di dataran rendah, karena kalau di tanaman di dataran tinggi akan menyebabkan cabai mudah membusuk.

   c. Cabai putih, cabai cengek, atau cabai burung.

      Bentuk buah panjang dan langsing, paling panjang diantara ketiga jenis cabai rawit. Panjang buah 4-6 cm, lebarnya 1-1,5 cm [8]. Tanaman ini sering tumbuh di dataran rendah, seperti di tepi hutan atau di perkebunan.

2. Cabai Merah (*Capsicum Annuum L.*)

   Cabai merah terdiri dari dua jenis, yakni:

   a. Cabai Merah Keriting

      Bentuk buah memanjang, mengikal atau mengeriting, dan bagian ujungnya meruncing. Rasanya pedas, bijinya relatif banyak jika

dibandingkan dengan ukuran buahnya [8]. Biasanya cabai keriting suka dipake untuk kebutuhan masyarakat untuk memasak.

b. Cabai merah besar atau cabai bulat

Bentuk buah pendek sampai panjang, dan bagian ujungnya tumpul atau bulat. Rasa buah kurang pedas dan aga manis. Kulit buah relatif lebih tebal dibandingkan dibandingkan dengan cabai keriting [8]. Cabai merah ini biasanya tiap daerah beda-beda dalam penyebutannya atau penamaan nya.

3. Paprika (*Capsicum longum L. Send*)

Cabai ini sering disebut cabai lonceng (*bell-pepper*). Cabai paprika belum banyak beredar di indonesia, karena harga benihnya mahal dan pemeliharaannya agak sulit [8]. Karena masyarakat indonesia belum terbiasa dalam memelihara cabai paprika tersebut sehingga dirasakan kesulitan untuk merawatnya.

4. Cabai Hias (*Capsicum spp*)

Walaupun dapat dimakan, jenis cabai ini jarang di konsumsi orang. Bentuk buah umumnya bulat atau bulat panjang [8]. Sebagian cabai hias merupakan tanaman penghias halaman atau ruang depan, tanaman cabai hias ini berbentuk buah menarik. Walaupun menarik, tetapi jarang dikonsumsi oleh manusia.

2.2.2 Syarat Tumbuh Tanaman Cabai

Syarat tumbuh tanaman cabai dalam budi daya tanaman cabai adalah sebagai berikut [8]:

1. Iklim

Suhu berpengaruh pada pertumbuhan tanaman, demikian juga terhadap tanaman cabai. Suhu yang ideal untuk budidaya cabai adalah 24-28°C. Pada suhu tertentu seperti 15°C dan lebih dari 32°C akan menghasilkan buah cabai yang kurang baik[8]. bahwa tanaman cabai dapat tumbuh pada musim kemarau apabila dengan pengairan yang cukup dan teratur. Iklim yang dikehendaki untuk pertumbuhannya antara lain [8]:

a. Sinar Matahari

Penyinaran yang dibutuhkan adalah penyinaran secara penuh, bila penyinaran tidak penuh pertumbuhan tanaman tidak akan normal.

b. Curah Hujan

Walaupun tanaman cabai tumbuh baik di musim kemarau tetapi juga memerlukan pengairan yang cukup.

c. Suhu dan Kelembaban

Tinggi rendahnya suhu sangat mempengaruhi pertumbuhan tanaman. Adapun suhu yang cocok untuk pertumbuhannya adalah siang hari 21°C-28°C, malam hari 13°C-16°C, untuk kelembaban tanaman 80%.

d. Angin

Angin yang cocok untuk tanaman cabai adalah angin sepoi-sepoi, angin berfungsi menyediakan gas CO2 yang dibutuhkannya.

2. Ketinggian Tempat

Ketinggian tempat untuk penanaman cabai adalah adalah dibawah 1400 m dpl. Berarti cabai dapat ditanam pada dataran rendah sampai dataran tinggi (1400 m dpl) [8]. Di daerah dataran tinggi tanaman cabai dapat tumbuh, tetapi tidak mampu berproduksi secara maksimal karena tidak cocok untuk tanaman cabai.

3. Tanah

Cabai sangat sesuai ditanam pada tanah yang datar. Dapat juga ditanam pada lereng-lereng gunung atau bukit. Tetapi kelerengan lahan tanah untuk cabai adalah antara 0-10° [8]. Tanaman cabai juga dapat tumbuh dan beradaptasi dengan baik pada berbagai jenis tanah, mulai dari tanah berpasir hingga tanah liat.

2.2.3. Definisi Hama

Hama adalah segala makhluk hidup yang merusak dan menggangu pertumbuhan tanaman, sehingga mengurangi kualitas atau kuantitas hasil tanaman. Sedangkan penyakit tanaman adalah pertumbuhan tanaman yang tidak sesuai dari keadaan normal yang di timbulkan oleh binatang atau agens biotik yang merusak tanaman[9]. Hewan yang dikategorikan hama ditempat satu, belum tentu menjadi hama ditempat yang lain, contohnya "banyaknya populasi kelinci di benua Australia akibatnya kelinci menjadi hama", beda hal nya ketika kita jalan-jalan ke Lembang Bandung Jawabarat, "kelinci dibudidayakan sebagai mata pencarian

penduduk disana". dalam hal ini tidak semua hama menjadi hama, jika ada langkah untuk membudidayakannya.

2.2.3 Jenis-Jenis Hama Pada Tanaman Cabai

Hama tanaman dalam arti luas adalah semua organisme atau binatang yang aktifitas hidupnya menyebabkan kerusakan tanaman sehingga menimbulkan kerugian secara ekonomi bagi manusia. Organisme yang menjadi hama adalah binatang yang menyerang tanaman budidaya sehingga menimbulkan kerugian. Hama tanaman sering disebut serangga hama[10].

Berikut adalah jenis-jenis hama yang sering menyerang tanaman cabai diantaranya adalah [10]:

1. *Trips* (*Thrips Parvispinus*)

   Trips menyerang tanaman cabai sepanjang tahun, serangan hebat umumnya terjadi pada musim kemarau. Serangga dewasa bersayap seperti jumbai (sisir bersisi dua), sedangkan nimfa tidak bersayap [10]. Pada serangan berat menyebabkan daun, tunas atau pucuk menggulung ke dalam dan muncul benjolan seperti tumor, pertumbuhan tanaman terhambat dan kerdil bahkan pucuk tanaman menjadi mati.

2. Lalat Buah (*B. dorsalis*)

   Lalat buah menyebabkan kerusakan pada buah cabai yang masih muda maupun buah yang sudah matang. Di lapangan hama ini merusak buah yang masih segar, dari buah muda sampai dengan buah menjelang masak. Gejala serangan pada buah yang terinfestasi lalat buah ditandai dengan adanya noda-noda kecil bekas tusukan ovipositornya [10]. Buah yang terserang akan membusuk dan kemudian jatuh ke tanah.

3. Ulat Gerayak (*S. litura*)

   Ulat grayak (*S. litura*) merupakan salah satu hama daun yang penting karena mempunyai kisaran inang yang luas meliputi kedelai, kacang tanah, kubis, ubi jalar, tebu, dan tanaman herba lainnya. Pada daun yang terserang oleh larva yang masih kecil terdapat sisasisa epidermis bagian atas dan tulang-tulang daun saja. Larva yang sudah besar merusak tulang daun dan buah. Gejala serangan pada buah ditandai dengan timbulnya lubang

tidak beraturan pada buah cabai [10]. Serangan berat pada umumnya terjadi pada musim kemarau dan menyebabkan *defoliasi* daun yang sangat berat.

4. Kutu Kebul (*B. Tabaci*)

Hama kutu kebul, B. tabaci yang merupakan hama penting pada tanaman cabai. Hama ini pertama kali ditemukan di Indonesia pada tahun 1938 pada tanaman tembakau. Permasalahan hama B. tabaci tidak terbatas hanya di Indonesia, karena hama ini juga menyerang berbagai tanaman di berbagai negara seperti Australia, India, Sudan, Iran, EL Savador, Mexico, Brazil, Turki, Israel, Thailand, Arizona, California (*Horowitz 1986*), Jepang (*Ohto 1990*) dan USA (*Perring et al. 1993*) [10]. Gejala serangan pada daun berupa bercak, disebabkan oleh rusaknya sel-sel dan jaringan daun akibat serangan *nimfa* dan serangga dewasa.

5. *Virus* Kuning(*Gemini Virus*)

Helai daun mengalami *vein clearing* dimulai dari daun pucuk berkembang menjadi warna kuning jelas, tulang daun menebal dan daun menggulung ke atas. Infeksi lanjut dari gemini virus menyebabkan daun mengecil dan berwarna kuning terang, tanaman kerdil dan tidak berbuah [10]. Keberadaan penyakit ini sangat merugikan karena mampu mempengaruhi produksi buah cabai tersebut.

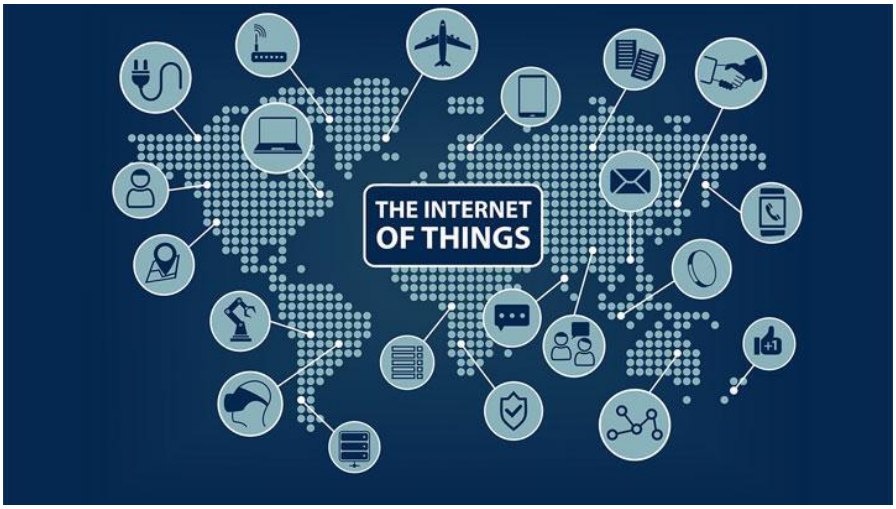
2.3 *Internet Of Things*

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[11]. *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.

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. 1 Internet of Things

Internet Of Things (IoT) juga diidentifikasi sebagai *enabler* untuk interaksi mesin-ke-mesin, manusia-ke-mesin, dan manusia dengan lingkungan. Dengan meningkatnya jumlah *smart device* dan adopsi protocol baru seperti *IPv6*, *Internet Of Things* (IoT) diperkirakan akan bergeser kearah perpaduan *smart networking* dan *autonomous networks* dari objek-objek yang dilengkapi *internet*. *Internet Of Things* (IoT) menawarkan banyak keuntungan untuk berbagai aplikasi, termasuk *emergency management*, *smart farming*, perawatan kesehatan dan lain-lain. Peran penting lain dari *Internet Of Things* (IoT) adalah membangun sistem kolaboratif yang mampu merespons secara efektif terhadap peristiwa yang ditangkap melalui sensor dengan efektif [11]. Dalam penelitian ini konsep *Internet Of Things* (IoT) digunakan sebagai konsep utama manusia dengan tanaman untuk melakukan

monitoring hama pada tanaman cabai dengan menggunakan microcontroller raspberry pi, sensor yang digunakan adalah sensor DHT 22 untuk mendapatkan keadaan suhu dan kelembaban ruangan, sensor LDR untuk pencahayaan otomatis, sensor Soil Moisture untuk mendapatkan keadaan kelembaban tanah dan modul relay untuk melakukan pengontrolan water jet pump.

2.3.1 Sejarah

Menurut(Burange & Misalkar, 2015)Internet of Things (IoT) adalah struktur di mana 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 komputer. *Internet of Things* merupakan perkembangan keilmuan yang sangat menjanjikan untuk mengoptimalkan kehidupan berdasarkan sensor cerdas dan peralatan pintar yang bekerjasama melalui jaringan internet [12]. Sejak mulai dikenalnya internet pada tahun 1989, mulai banyak hal kegiatan melalui internet, Pada tahun 1990 JohnRomkey menciptakan 'perangkat', pemanggang roti yang bisa dinyalakan dan dimatikan melalui Internet. *WearCam* diciptakan padatahun 1994 oleh Steve Mann. Pada tahun 1997 Paul Saffo memberikan penjelasan singkat pertama tentang sensor dan masa depan. Tahun 1999 Kevin Ashton menciptakan *The Internet of Things*, direktur eksekutif *Auto IDCentre*, MIT. Mereka juga menemukan peralatan berbasis RFID (*Radio Frequency Identification*) global yang sistem identifikasi pada tahun yang sama [12]. Penemuan ini disebut sebagai sebuah lompatan besar dalam *commercialisingIoT* pada masa itu.

2.3.2 Cara Kerja

Internet of Things menggunakan beberapa teknologi yang secara garis besar di gabungkan menjadi satu kesatuan diantaranya sensor sebagai pembaca data, koneksi internet dengan bebarapa macam topologi jaringan, *radio frequency identification* (RFID), *wireless* sensor *network* dan teknologi yang terus akan bertambah sesuai dengan kebutuhan [12]. 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*.

2.4 *Analisis Fungsional*

Analisis kebutuhan fungsional akan dimulai setelah tahap analsis terdapat sistem selesai dilakukan, analisis kebutuhan f*ungsional* dapat didefinisikan sebagai penggambaran, perencanaan dan pembuatan *sketsa* atau pengaturan dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi. Pemodelan sistem dilakukan menggunakan UML (Unified Modelling Language) dimana tahap-tahap pemodelan dalam analisis tersebut antara lain Use Case Diagram, Use Case Scenario, Activity diagram, Class diagram dan Sequence diagram [5]. 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.

2.4.1 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*[13]. Pada penelitian ini konsep OO digunakan untuk konsep pengkodean pada sistem yang

akan dibangun. Berikut ini adalah konsep-konsep dalam pemrograman berorientasi *object oriented*:

1. *Class*

   Kelas (*Class*) merupakan penggambaran satu set objek yang memiliki atribut yang sama. class 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.

2.4.2 UML (*Unified Modeling Language*)

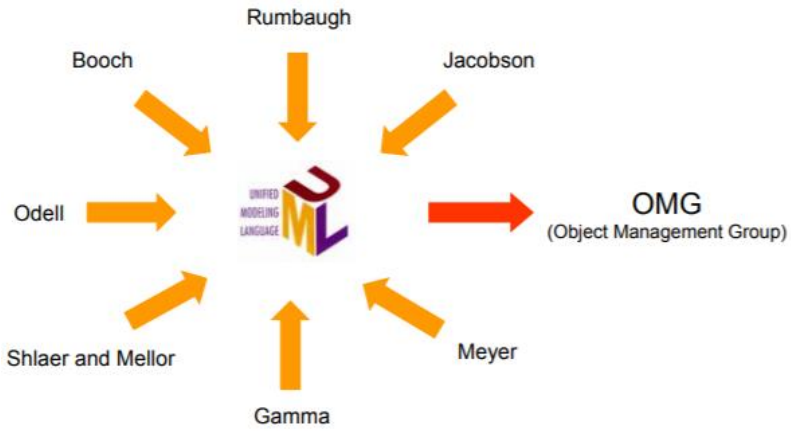
UML merupakan singkatan dari "*Unified Modelling Language*" yaitu suatu metode permodelan secara visual untuk sarana perancangan sistem berorientasi objek, atau definisi UML yaitu sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan dan juga pendokumentasian sistem *software*. 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[14].

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 bahasa-bahasa

berorientasi objek seperti *C++*, *Java, C#* atau *VB.NET*. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C [14].

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*)[15].



Gambar 2. 2 UML

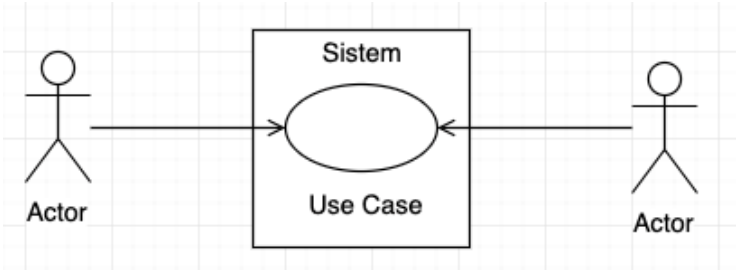
Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan 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 [15]. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

2.4.1.1 *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[16].

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.

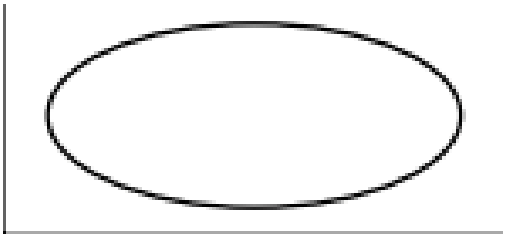


Gambar 2. 3 Use case diagram [16]

Berikut ini adalah bagian dari sebuah *use case diagram*:

a. *Use Case*

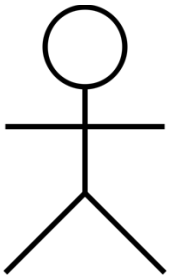
Use case adalah abstraksi dari interaksi antara sistem dengan *actor*. Oleh karena itu sangat penting untuk memilih abstraksi yang cocok. *Use case* dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ‘apa’ yang dikerjakan *software* aplikai, bukan ‘bagaimana’ *software* aplikasi mengerjakannya. 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 2.3 menunjukan bentuk *Use Case* dalam UML.



Gambar 2. 4 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 2.4 menunjukan bentuk *actor* dalam UML.



Gambar 2. 5 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.

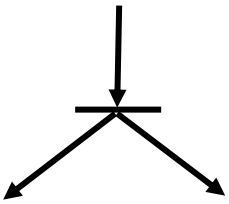
d. *Generalisasi* antara 2 *use case*.

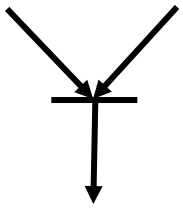
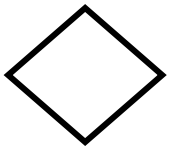
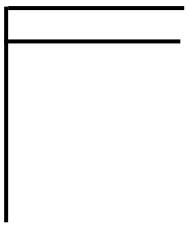
Hubungan *inheritance* (pewarisan) yang melibatkan *use case* yang satu (*the child*) dengan *use case* lainnya(*the parent*). *Generalisasi tipe* ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

2.4.1.2. *Activity Diagram*

Activity diagram merupakan *diagram* yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. *Activity diagram* juga digunakan untuk mendefinisikan urutan atau pengelompokan tampilan dari sistem/user *interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antar muka tampilan serta rancang menu yang ditampilkan pada perangkat lunak [13].

Tabel 2. 1 Daigram Activity

Gambar	Keterangan
	*Start point,* diletakkan pada pojok kiri atas dan merupakan awal aktivitas.
	*End point,* akhir aktivitas
	*Activities,* menggambarkan suatu proses / kegiatan bisnis.
	*Fork*/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.

	*Jioin* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*.
	*Swinlane*, pembagian *activity diagram* untuk menunjukkan siapa melakukan apa.

2.4.1.3. *Class Diagram*

Class adalah sebuah *spesifikasi* yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek[13].

Class menggambarkan keadaan diantaranya :

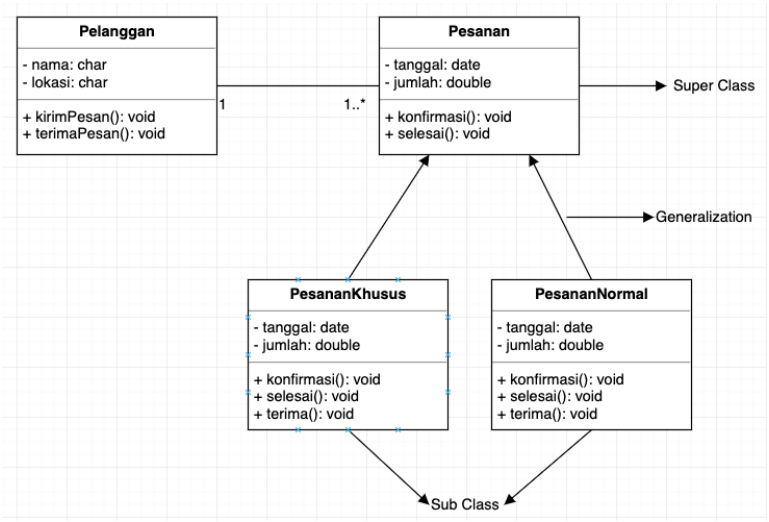
Atribut/properti suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Menggambarkan struktur dan deskripsi class, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

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. 6 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*

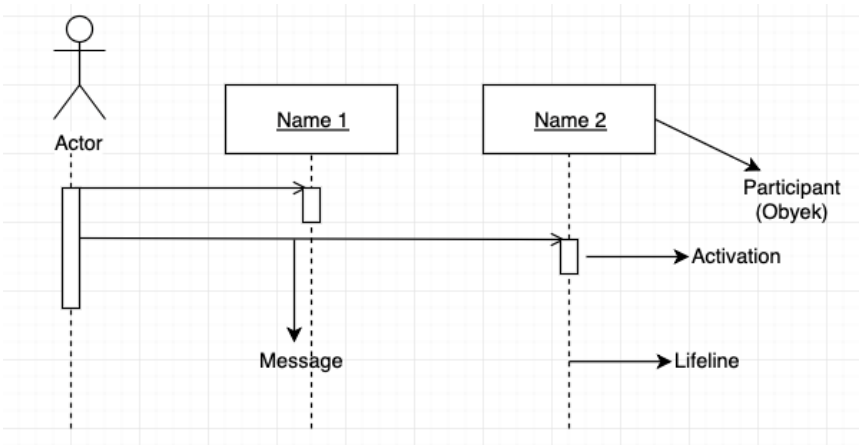
Generalization adalah sebuah hubungan antar *class* yang bersifat dari khusus ke umum.

c. *Constraint*

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

2.4.1.4 *Sequence 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 [13].



Gambar 2. 7 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.

2.5 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 dibawah lisensi GNU General *Public License* (GPL), tetapi mereka juga menjual dibawah *lisensi komersial* untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. *Relational Database Management System* (RDBMS) [17].

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat komersial [17]. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*).

Gambar 2. 8 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.

2.7 *TensorFlow*

TensorFlow adalah perpustakaan perangkat lunak, yang dikembangkan oleh Tim *Google Brain* dalam organisasi penelitian Mesin Cerdas *Google,* untuk tujuan melakukan pembelajaran mesin dan penelitian jaringan syaraf dalam. Merupakan kerangka dasar yang digunakan dalam proses machine learning serta sebagai library khusus untuk machine learning yang dikembangkan oleh google. Arsitektur yang fleksibel memungkinkan penerapan komputasi yang mudah di terapkan di berbagai platform seperti (CPU,GPU,TPU) dengan ekstraksi CUDA untuk komputasi tujuan umum pada unit pemrosesan grafis. Tensorflow ini membuat penelitian secara umum dan terbuka[18]. *TensorFlow* kemudian menggabungkan aljabar komputasi teknik pengoptimalan kompilasi, mempermudah penghitungan banyak ekspresi matematis dimana masalahnya adalah waktu yang dibutuhkan untuk melakukan perhitungan. Fitur utamanya meliputi:

1. Mendefinisikan, mengoptimalkan, dan menghitung secara efisien ekspresi matematis yang melibatkan *array multi dimensi* (tensors).
2. Pemrograman pendukung jaringan syaraf dalam dan teknik pembelajaran mesin
3. Penggunaan GPU yang transparan, mengotomatisasi manajemen dan optimalisasi memori yang sama dan data yang digunakan. *Tensorflow* bisa menulis kode yang sama dan menjalankannya baik di CPU atau GPU. Lebih

khusus lagi, *TensorFlow* akan mengetahui bagian perhitungan mana yang harus dipindahkan ke GPU.

4. Skalabilitas komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar.

2.8 *Analisi Non Fungsional*

Analisis kebutuhan non-fungsional merupakan suatu analisis yang digunakan untuk pengspesifikasian kebutuhan sistem. Analisis kebutuhan non-fungsional terdiri dari kebutuhan perangkat keras, kebutuhan perangkat lunak serta analisis pengguna[19]. 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.

2.8.1 *Teknologi Mikrocontroller*

Mikrokontroler adalah system *mikroprosesor* lengkap yang terkandung di dalam sebuah chip. Mikrokontroler berbeda dari mikroprosesor serbaguna yang digunakan dalam sebuah PC, karena sebuah mikrokontroler umumnya telah berisi komponen pendukung sistem minimal mikroprosesor, yakni memori dan pemrograman *Input-Output*. Mikrokontroler dapat deprogram untuk melakukan penghitungan, menerima input dan menghasilkan output. Mikrokontroler mengandung sebuah inti prosessor, memori dan pemrograman Input-Output[20]. Meskipun kecepatan pengolahan data dan kapasitas memori pada *microcontroller* jauh lebih kecil jika dibandingkan dengan komputer personal, namun kemampuan *microcontroller* sudah cukup untuk dapat digunakan pada banyak aplikasi terutama karena ukurannya yang kompak. *Microcontroller* sering digunakan pada sistem yang tidak terlalu kompleks dan tidak memerlukan kemampuan komputasi yang tinggi.

2.8.2 Raspberry Pi

Raspberry Pi adalah komputer papan tunggal (*Single Board Circuit /SBC*) atau komputer mini yang memiliki ukuran sebesar kartu kredit. *Raspberry Pi* sangat berguna untuk berbagai keperluan, seperti *spreadsheet*, *game*, memutar *video high definition*. *Raspberry Pi* dikembangkan oleh yayasan nirbala yaitu *Raspberry Pi*

Foundation yang dikelola developer dan ahli komputer dari Universitas *Cambridge, Inggris* [5].

Gambar 2. 9 Raspberry pi 3

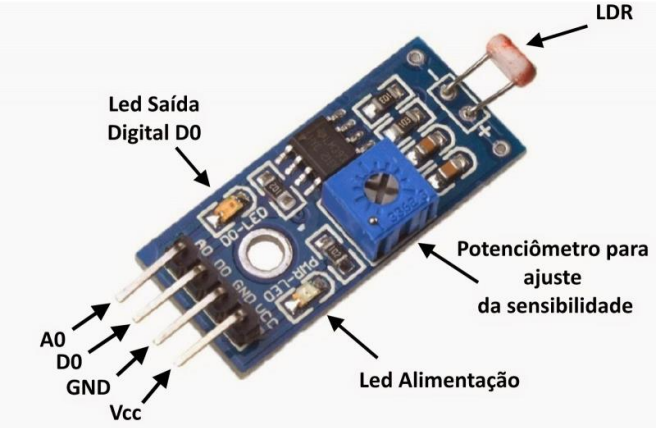
Raspberry pi sudah terbenam USB host yang memungkinkan komunikasi dengan perangkat luar seperti *mouse* atau *keyboard,* selaian itu juga terdapat *port HDMI* dan *Composed* A/V 3,5mm Jack sebagai interface audio video. *Port LAN* dan *wifi*, dan *bluetooth* dapat digunakan untuk terhubung dengan jaringan komunikasi, *Camera Serial Interface* dan *Display serial interface* dapat dijadikan alternatif *interface* kamera maupun untuk monitor [5].

2.8.3 Sensor LDR

LDR atau *light Dependent Resistor* adalah salah satu jenis *resistor* yang nilai hambatannya dipengaruhi oleh cahaya yang diterima olehnya. Besarnya nilai hambatan pada LDR tergantung pada besar kecilnya cahaya yang diterima oleh LDR itu sendiri. Contoh penggunaannya adalah pada lampu taman dan lampu di jalan yang bisa menyala di malam hari dan padam di siang hari secara otomatis. Atau bisa juga kita gunakan di kamar kita sendiri[21]. *Resistor* peka cahaya atau *fotoresistor* adalah komponen elektronik yang resistansinya akan menurun jika ada penambahan intensitas cahaya yang mengenainya. *Fotoresistor* dapat merujuk pula pada *light-dependent resistor* (LDR), atau *fotokonduktor.*

Fotoresistor dibuat dari *semikonduktor* beresistansi tinggi yang tidak dilindungi dari cahaya. Jika cahaya yang mengenainya memiliki frekuensi yang cukup tinggi, foton yang diserap oleh semikonduktor akan menyebabkan elektron

memiliki energi yang cukup untuk meloncat ke pita konduksi [21]. Elektron bebas yang dihasilkan (dan pasangan lubangnya) akan mengalirkan listrik, sehingga menurunkan resistansinya.



Gambar 2. 10 LDR

Sensor LDR dalam alat ini adalah untuk menerangi tanaman cabai dari gangguna serangga ngengat lampu ldr ini juga bisa menyala di malam hari dan padam di siang hari secara otomatis.

2.8.4 *Sensor Pi Camera*

Sistem operasi utama *Raspberry Pi* menggunakan *Debian GNU/Linux* dan bahasa pemrograman *python*. Komputer ini dilengkapi dengan 4 *USB* versi 2.0 *Port* yang dapat dihubungkan dengan perangkat yang menggunakan *port USB* apapun. Komputer dilengkapi juga dengan *ethernet port* untuk sambungan *LAN* (*Local Area Network*), *HDMI port* untuk disambungkan dengan perangkat layar LCD/LED dengan kualitas HD (*High Definition*), *connector* 3,5mm untuk disambungkan ke perangkat *speaker* atau *headset*, dan *USB micro* untuk *power supply*. Selain itu disediakan juga 2 *Port* khusus untuk kamera dan LCD *Raspberry Pi*[22]. Komputer ini menyediakan 40 pin GPIO (*General Purpose Input Output*) agar *Raspberry Pi* dapat digunakan sebagai pengendali rangkaian elektronik.

Gambar 2. 11 pi camera

Kamera yang digunakan untuk melakukan proses akusisi citra pada penelitian ini adalah *Pi Camera* yang merupakan kamera khusus yang didesain untuk *minicomputer Raspberry Pi*. Dalam alat ini *pi camera* digunakan untuk mendeteksi objek ulat yang ada pada tanaman cabai. Seperti terlihat pada Gambar 2.11, dengan ukuran kecil, modul kamera *Raspberry Pi* dapat digunakan untuk mengambil citra dengan kualitas *high definition* memiliki kualitas 5 MP mendukung resolusi video 1080p 30 *fps* ,720p 60 *fps* dan VGA90,dapat bekerja pada semua model *Raspberry Pi* yang terhubung pada *port CSI*.

Gambar 2. 12 antarmuka raspi camera dengan raspi 3 board

2.8.5 Bahasa Pemrograman *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*)[23]. *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 [23]. *Python* dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan diberbagai *platform* sistem operasi.

Gambar 2. 13 Icon Aplikasi Bahasa Python

Beberapa fitur yang dimiliki *Python* adalah[23]:

1. Memiliki kepustakaan yang luas, dalam distribusi *Python* telah disediakan modul-modul siap pakai untuk berbagai keperluan.
2. Memiliki tata bahasa yang jernih dan mudah dipelajari.
3. Memiliki aturan layout kode sumber yang memudahkan pengecekan, pembacaan kembali, dan penulisan ulang kode sumber berorientasi objek.
4. Memiliki sistem pengelolaan memori otomatis (*garbage collection*, seperti *java*)
5. *Modular*, mudah dikembangkan dengan menciptakan modul-modul baru. Modul-modul tersebut dapat dibangun dengan bahasa *Python* maupun *C/C++*.
6. Memiliki fasilitas pengumpulan sampah otomatis. Seperti halnya pada bahasa pemrograman *java*, *Python* memiliki fasilitas pengaturan penggunaan ingatan komputer sehingga para pemrogram tidak perlu melakukan pengaturan ingatan komputer secara langsung.

Python memiliki beberapa macam kelebihan dan kekurangan. Berikut ini beberapa macam kelebihan bahasa *Python* [23]:

1. Tidak ada deklarasi tipe sehingga program menjadi lebih sederhana, singkat dan fleksibel.
2. Manajemen memori otomatis yaitu kumpulan sampah memori sehingga dapat menghindari pencatatan kode.
3. Mendukung program skala besar dan secara modular karena terdapat kelas, modul dan eksepsi.
4. Kecepatan perubahan pada masa pembuatan sistem aplikasi lebih tinggi karena tidak ada tahapan kompilasi dan penyambungan (link).
5. Pemrograman berorientasi objek.
6. Model objek universal kelas satu.
7. Konstruksi pada saat aplikasi berjalan.
8. Interaktif, dinamis dan alamiah.
9. Akses hingga informasi interpreter.

10. Portabilitas secara luas seperti pemrograman antar *platform* tanpa *ports*.
11. Kompilasi untuk portable kode *byte* sehingga kecepatan eksekusi bertambah dan melindungi kode sumber.

2.8.6 Sensor *Moisture Oil Sensor*

Sensor pH tanah merupakan sensor pendeteksi tingkat keasaman (acid) atau kebasaan (alkali) tanah. Skala pH yang dapat diukur oleh sensor pH tanah ini memiliki range 3.5 hingga 8. Sensor ini dapat langsung disambungkan dengan pin analog arduino maupun pin analog mikrokontroller lainnya[24].

Sensor kelembaban tanah terdiri dari dua probe yang digunakan untuk mengukur kandungan volumetric air. Kedua probe memungkinkan arus melewati tanah dan kemudian mendapat nilai resistansi untuk mengukur nilai kelembaban. Ketika ada lebih banyak air, tanah akan melakukan lebih banyak listrik yang berarti bahwa akan ada lebih sedikit perlawanan. Oleh karena itu, tingkat kelembapannya akan lebih tinggi. Tanah kering melakukan listrik dengan buruk, sehingga ketika akan ada lebih sedikit air, maka tanah akan melakukan lebih sedikit listrik yang berarti akan ada lebih banyak perlawanan. Oleh karena itu, tingkat kelembapannya akan lebih rendah[25]. Pada penelitian ini sensor pH tanah digunakan untuk mendapatkan nilai pH tanah, pada pembuatan sistem ini pH tanah akan memonitor nilai pH tanah pada tanaman cabai.

2.8.7 *Mini Water Pump*

Pompa air adalah alat yang digunakan untuk memindahkan air dari ketempat satu ketempat yang lainnya melalui selang. Pada pompa air ada lubang untuk masuknya air dan keluarnya air. Prinsipnya adalah menambahkan energi pada air secara terus menerus sehingga air bisa berpindah dengan kecepatan yang dihasilkan pada pompa air.

Pompa air ini berkerja ketika mendapatkan tegangan 12v sehingga dibutuhkan power supply untuk bisa menggunakan alat ini, bila diberi tegangan langsung ke mikrokontroller alat ini tidak akan kuat dikarekan arus yang dibutuhkan harus besar. Inlet dan outlet yang digunakan untuk selang ukuran 8mm setara dengan

selang pada akuarium. Untuk penggunannya pompa ini di tenggelamkan seluruhnya. Pompa air ini diklaim dapat berkerja 30.000 tetapi penggunannya harus ditenggelamkan di air, jika tidak akan mudah rusak ketika tidak ada air yang ditarik atau dipindahkan[26].

2.9 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. Pengujian adalah proses untuk menemukan error pada perangkat lunak sebelum dikirim kepada pengguna[27]. 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.

2.9.1 *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[28]. 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 dirabilitas. 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 usability 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.

Revision class partitioning adalah sebuah metode *black box* lainnya yang merupakan faktor dasar yang menentukan keberhasilan paket suatu *software*, pelayanan jangka panjang, dan keberhasilan penjualan ke sejumlah besar populasi pengguna [28]. Berkaitan dengan hal tersebut terdapat tiga kelas pengujian revisi sebagai berikut:

1. *Reusability test*

   Reusability menentukan bagian mana dari suatu program (modul, integrasi, dbs) yang akan dikembangkan untuk digunakan kembali pada projet pengembangan *software* lainnya, baik yang telah direncanakan maupun yang belum. Bagian ini harus dikembangkan, disusun, dan didokumentasikan menurut prosedur perpustakaan *software* yang digunakan ulang.

2. *software interoperability test*

   software interoperability berkaitan dengan kemamputan *software* dalam memenuhi perlengkapan dan paket *software* lainnya agar memungkinkan untuk mengoperasikannya bersama dalam satu sistem komputer kompleks.

3. *Equipment interoperability test*

   Equipment interoperability berkaitan dengan perlengkapan *firmware* dalam menghadapi untuk perlengkapan lain dan atau paket *software*, dimana persyaratan mencantumkan *specified interfaces*, termasuk

dengan *interfacing standard*. Pengujian yang relevan harus menguji implementasi dari *interoperability reqirements* dalam sistem.

Keuntungan dan kekurangan dari *black box testing*, beberapa keuntungan dari *black box testing*, diantaranya sebagai berikut:

1. *Black box* testing memungkinkan kita untuk memiliki sebagian besar tingkat pengujian, yang sebagian besarnya dapat diimplementasikan.
2. Untuk tingkat pengujian yang dapat dilakukan baik dengan *white box testing* maupun *black box testing*, *black box testing* memerlukan lebih sedikit sumber dibandingkan dengan yang dibutuhkan oleh *white box testing* pada pake *software* yang sama.

Sedangkan kekurangan dari *black box testing* adalah sebagai berikut:

1. Adanya kemungkinan untuk terjadinya beberapa kesalahan yang tidak disengaja secara bersama-sama akan menimbulkan respon pada pengujian ini dan mencegah deteksi kesalahan (*error*). Dengan kata lain, *black box test* tidak siap untuk mengidentifikasi kesalahan-kesalahan yang berlawanan satu sama lain sehingga menghasilkan output yang benar.
2. Tidak adanya kontrol terhadap *line coverage*. Pada kasus dimana *black box test* diharapkan dapat meningkatkan *line converage*, tidak ada cara yang mudah untuk menspesifikasikan parameter-parameter pengujian yang dibutuhkan untuk meningkatkan *coverage*. Akibatnya, *black box test* dapat melakukan bagian penting dari baris kode, yang tidak ditangani oleh set pengujian.
3. Ketidakmungkinan untuk menguji kualitas pembuatan kode dan pendekatannya dengan standar pembuatan kode.