BAB 2
TINJAUAN PUSTAKA

2.1 Tentang Tempat Penelitian

Daerah Irigasi Ciherang dibangun pada tahun 1919 dan di rehabilitasi pada tahun 1990/1991. Bendung Ciherang memiliki tiga saluran yaitu saluran primer, saluran sekunder, dan saluran tersier. Saluran primer adalah bagian dari jaringan irigasi yang terdiri dari bangunan utama yang nantinya akan dibagi ke saluran tersier dan saluran sekunder. Saluran irigasi sekunder bagian dari jaringan irigasi yang terdiri dari saluran sekunder, saluran pembuang, bangunan bagi, bangunan bagi sadap, bangunan sadap dan bangunan pelengkapnya. Saluran irigasi tersier adalah jaringan irigasi yang berfungsi sebagai prasarana pelayanan irigasi untuk mengairi petak pertanian. Bendung Ciherang mengambil air dari kali Cisangkuy yang bermata air di Gunung Wayang dan mendapatkan suplesi dari Situ Cileunca dan dari pembuangan PLTA. Daerah tangkapan air sungai Cisangkuy sekitar 294 km^2. Letak Bendung Ciherang berada di kampung Singkur Desa Jatisari Kec. Cangkuang Kab. Bandung jarak dari kantor Sub Unit Pelayanan Cimahi ke lokasi bendung ±3 km. Daerah yang diair dari jaringan irigasi Ciherang seluas 2.235,77 Ha. 1 saluran induk, 7 saluran sekunder, dan terdiri dari 47 petak tersier. Di Ciherang meliputi 5 Kecamatan yaitu : Kec. Cangkuang, Banjaran, Pameungpeuk, Katapang dan Baleendah. Terdiri dari 16 Desa, 1 Kelurahan.

2.1.1 Jaringan Irigasi

Berikut ini adalah macam-macam saluran jaringan irigasi daerah irigasi Ciherang, dapat dilihat pada tabel 2.1.

Tabel 2.1 Jaringan Irigasi

Saluran	Luas
Saluran Induk/Primer	7,853 Km (651,00 Ha)
Saluran Sekunder Hantap	3,522 Km (124,72 Ha)
Saluran Sekunder Cisalak	2,800 Km (192,96 Ha)
Saluran Sekunder Tanjung	1,024 Km (209,69 Ha)
Saluran Sekunder Bj. Kunci	0,291 Km (160,50 Ha)

Saluran Sekunder Ranca Tungku	0,622 Km (254,90 Ha)
Saluran Ranca Mulya	1,542 Km (182,00 Ha)
Saluran Sekunder Baleendah	6,584 Km (460,00 Ha)
Jumlah	**24,014 Km (2235,77 Ha)**

Berikut ini adalah investarisasi bangunan air daerah irigasi Ciherang, tabel 2.2 menunjukan inventarisasi bangunan air.

Tabel 2.2 Inventarisasi Bangunan

Bendung	1 Buah (3 Orang)
Bangunan Bagi Sadap	6 Buah (2 Orang)
Bangunan Sadap	27 Buah (9 Orang)
Bangunan Ukur	2 Buah
Bangunan Pelengkap	37 Buah

2.1.2 Pengelola Operasi dan Pemeliharaan Daerah Irigasi Ciherang

Daerah Irigasi Ciherang termasuk wilayah kerja Satuan Unit Pelayanan Cirasea Ciwidey pengelolaannya oleh Provinsi yaitu BPSDA (Balai Pengelola Sumber Daya Air) terdiri dari :

1. Juru/Mantri : 2 Orang
2. Penjaga Bendung : 2 Orang
3. Penjaga Pintu Air : 15 Orang

2.1.3 Stuktur Organisasi

Gambar 2.1 Struktur Organisasi

2.2 Purwarupa

Purwarupa adalah bentuk awal (contoh) atau standar ukuran dari sebuah entitas. Dalam bidang desain, sebuah prototype dibuat sebelum dikembangkan atau justru dibuat untuk pengembangan sebelum dibuat dalam skala sebenarnya atau sebelum diproduksi secara massal [4].

Berdasarkan website widuri.raharja.info/ adalah model kerja dasar dari pengembangan sebuah program (software) atau perangkat lunak. Prototipe dalam Bahasa Inggris “prototype” disebut juga dengan purwarupa. Prototipe biasanya dibuat sebagai model untuk tujuan demonstrasi atau sebagai bagian dari proses pengembangan atau pembuatan sebuah software [5].

Dalam bidang desain, Prototipe atau purwarupa atau disebut juga dengan *arketipe* adalah bentuk awal sebagai contoh atau standar ukuran dari sebuah entitas. Sebuah Prototipe dibuat sebelum dikembangkan atau justru dibuat khusus untuk pengembangan sebelum dibuat dalam skala sebenarnya atau sebelum diproduksi secara massal. Prototype adalah sebuah Javascript Framework yang dibuat untuk lebih memudahkan proses dalam membangun aplikasi berbasis web. Metode protyping sebagai suatu paradigma baru dalam pengembangan sistem informasi, tidak hanya sekedar suatu evolusi dari metode pengembangan sistem informasi yang sudah ada, tetapi sekaligus merupakan revolusi dalam pengembangan sistem informasi manajemen [5].

Keuntungan dari purwarupa :

1. Menghasilkan syarat yang lebih baik dari produksi yang dihasilkan oleh metode ‘spesifikasi tulisan’.
2. User dapat mempertimbangkan sedikit perubahan selama masih bentuk prototipe.
3. Memberikan hasil yang lebih akurat dari pada perkiraan sebelumnya, karena fungsi yang diinginkan dan kerumitannya sudah dapat diketahui dengan baik.
4. User merasa puas. Pertama, user dapat mengenal melalui komputer. Dengan melakukan prototipe (dengan analisis yang sudah ada), user belajar mengenai komputer dan aplikasi yang akan dibuatkan untuknya. Kedua, user terlibat langsung dari awal dan memotivasi semangat untuk mendukung analisis selama proyek berlangsung.

2.3 Pertanian

Pertanian adalah kegiatan pemanfaatan sumber daya hayati yang dilakukan manusia untuk menghasilkan bahan pangan, bahan baku industri, atau sumber energi, serta untuk mengelola lingkungan hidupnya [1].

Indonesia merupakan salah satu negara agraris dimana, sebagian besar penduduknya tinggal di perdesaan dengan mata pencaharian sebagai petani. Penduduk Indonesia pada umumnya mengkonsumsi hasil pertanian untuk makanan

pokok mereka. Pertanian di Indonesia perlu ditingkatkan produksinya semaksimal mungkin menuju swasembeda pangan akan tetapi, tantangan untuk mencapai hal tersebut sangat besar karena luas wilayah pertanian yang semakin lama semakin sempit, penyimpangan iklim, pengembangan komoditas lain, teknologi yang belum modern, dan masalah yang satu ini adalah masalah yang sering meresahkan hati para petani yaitu hama dan penyakit yang menyerang tanaman yang dibudidayakan. Luas pertanian di Indonesia yang semakin menyempit hal inilah yang menjadi tantangan terbesar saat ini yang harus dihadapi akan tetapi, ada cara yang dapat dilakukan untuk mengantisipasinya yaitu dengan cara melakukan pembangunan sektor pertanian [1].

Bentuk-bentuk lahan pertanian di Indonesia yaitu diantaranya sawah, tegalan, pekarangan, ladang berpindah dan lainnya. Hasil pertanian di Indonesia sangatlah beragam diantaranya adalah beras, avage, avokad, kopi, jagung, bawang, cengkeh, kakao, kacang-kacangan, kapas, kapuk, karet, kayu manis, kedelai, kelapa, kelapa sawit, kentang, ketela, ubi jalar, sagu dan lainnya [1].

2.4 Sawah

Sawah adalah tanah berlumpur di lahan datar dengan tekstur tanah berlempung yang keras di bagian dalam sehingga dapat menampung genangan air. Sawah biasanya di buat berpetak-petak yang antara petak yang satu dengan yang lain di batasi oleh pematang. Sawah biasanya di gunakan sebagai lahan untuk menanam padi dan palawija. Pengolahan tanah sawah tergantung dari jenis tanaman yang akan di tanam. Jika yang di tanam adalah padi sawah, maka sawah perlu di genangi air. Tetapi jika yang di tanam adalah sayur-mayur dan palawija, maka sawah akan di keringkan untuk mengurangi kadar airnya [6].

2.4.1 Fungsi dan Manfaat Sawah

Sawah merupakan salah satu jenis lahan pertanian yang berfungsi untuk menanam padi. Sawah yang baik akan menghasilkan hasil panen yang baik, akan tetapi yang paling penting adalah bibit dan perawatannya. Sebenarnya menanam padi tidak hanya dilakukan disawah saja, tetapi dapat juga dilakukan di darat. Namun bagi petani yang umumnya berada pada lingkungan pedesaan atau daerah

transmigrasi, maka menanam padi sawah menjadi pilihan yang tepat dalam bercocok tanam padi. Manfaat sawah ini sangat penting sekali bagi kehidupan para petani dan juga bagi manusia lainnya. Manfaat sawah sangat penting, karena petani bisa menanam padi untuk kepentingan pribadi dan banyak orang [6].

Selain bermanfaat untuk menanam padi, ternyata manfaat sawah ini juga berguna untuk hal lainnya, seperti:

1. Menghasilkan bahan pangan yaitu beras. Beras adalah bahan pangan pokok yang umum digunakan oleh masyarakat seluruh indonesia bahkan mancanegara. Dengan adanya sawah ini maka para petani dapat membudidayakan padi yang menghasilkan beras sebagai bahan pokok pangan.
2. Menghasilkan tenaga kerja bagi masyarakat. Karena dengan adanya sawah dan petani yang menanam padi, maka lapangan kerja jadi bertambah, sebagai contohnya yaitu misalkan seorang petani akan menggarap sawah, maka dia membutuhkan orang untuk membantu mengerjakannya dengan memberikan upah. Ini merupakan salah satu contoh manfaat adanya sawah tersebut.
3. Mempertahankan budaya tradisional dan kerakyatan bangsa. Sawah merupakan lahan tempat para petani bercocok tanam dari sejak dulu hingga sekarang dan sudah membudaya. Masyarakat pedesaan yang mempergunakan sawah berarti mereka sudah mempertahankan budaya tradisional.
4. Sebagai tempat untuk menumbuhkan sifat gotong royong antar warga. Sudah jelas bahwa dengan adanya sawah ini dapat menumbuhkan sifat gotong royong antar warga. Saling membantu dalam proses pengolahan sawah secara bergantian.
5. Sebagai sumber pendapatan masyarakat. Sumber pendapatan para petani padi adalah berasal dari sawah. Tanpa adanya sawah ini maka pendapatan mereka jadi susah untuk didapatkan.
6. Berguna sebagai sirkulasi air, khususnya pada musim kemarau. Dengan adanya sawah ini maka sirkulasi air dapat terkendali. Parit-

parit akan hidup dan air akan terus mengalir selama ada petani yang menanam padi sawah.

7. Meminimalisir terjadinya banjir. Dengan adanya sawah ini juga dapat mengurangi potensi banjir.

2.5 Irigasi

Irigasi adalah semua atau segala kegiatan yang mempunyai hubungan dengan usaha untuk mendapatkan air guna keperluan pertanian. Usaha yang dilakukan tersebut dapat meliputi: perencanaan, pembuatan, pengelolaan, serta pemeliharaan sarana untuk mengambil air dari sumber air dan membagi air tersebut secara teratur dan apabila terjadi kelebihan air dengan membuangnya melalui saluran drainasi [7].

Irigasi merupakan salah satu faktor penting dalam kegiatan usaha tanidalam arti luas. Sejalan dengan era reformasi dan otonomi daerah, maka saat ini telah ada pengaturan baru yang mengatur tentang irigasi, yaitu pengelolaan diserahkan kepada petani. Namun demikian pemerintah tetap berkewajiban untuk membantu petani terutama dalam bimbingan teknis dan keuangan sampai mampu mengelolanya secara mandiri. Irigasi didefinisikan sebagai suatu cara pemberian air, baik secara alamiah ataupun buatan kepada tanah dengan tujuan untuk memberi kelembaban yang berguna bagi pertumbuhan tanaman [7].

Secara garis besar, tujuan irigasi dapat digolongkan menjadi 2 (dua) golongan, yaitu: Tujuan Langsung, yaitu irigasi mempunyai tujuan untuk membasahi tanah berkaitan dengan kapasitas kandungan air dan udara dalam tanah sehingga dapat dicapai suatu kondisi yang sesuai dengan kebutuhan untuk pertumbuhan tanaman yang ada di tanah tersebut. Tujuan Tidak Langsung, yaitu irigasi mempunyai tujuan yang meliputi : mengatur suhu dari tanah, mencuci tanah yang mengandung racun, mengangkut bahan pupuk dengan melalui aliran air yang ada, menaikkan muka air tanah, meningkatkan elevasi suatu daerah dengan cara mengalirkan air dan mengendapkan lumpur yang terbawa air, dan lain sebagainya [7].

Sesuai dengan definisi irigasinya, maka tujuan irigasi pada suatu daerah adalah upaya rekayasa teknis untuk penyediaaan dan pengaturan air dalam menunjang proses produksi pertanian, dari sumber air ke daerah yang memerlukan serta mendistribusikan secara teknis dan sistematis.

2.5.1 Manfaat Sistem Irigasi

Adapun manfaat dari suatu sistem irigasi, adalah :

1. Untuk membasahi tanah, yaitu pembasahan tanah pada daerah yang curah hujannya kurang atau tidak menentu.
2. Untuk mengatur pembasahan tanah, agar daerah pertanian dapat diairi sepanjang waktu pada saat dibutuhkan, baik pada musim kemarau maupun musim penghujan.
3. Untuk menyuburkan tanah, dengan mengalirkan air yang mengandung lumpur & zat – zat hara penyubur tanaman pada daerah pertanian tersebut, sehingga tanah menjadi subur.
4. Untuk kolmatase, yaitu meninggikan tanah yang rendah / rawa dengan pengendapan lumpur yang dikandung oleh air irigasi.

2.5.2 Jenis Irigasi di Indonesia

Jenis-jenis irigasi di Indonesia adalah :

1. Irigasi permukaan : Mengambil air dari sumber-sumber yang ada, lalu membuat bangunan penangkapnya, kemudian mengalirkannya melalui saluran primer dan sekunder ke petak-petak sawah.
2. Irigasi tambak : Mengatur tata air dari sumber irigasi yang sudah ada melalui system drainase (menahan dan mengairi padi)
3. Irigasi air tanah : Mengambil air tanah kemudian memompa dan mendistribusikannya ke petak-petak sawah.
4. Irigasi pompa : Diutamakan untuk areal persawahan di dataran tinggi

2.6 *Smart Irrigation*

Smart Irrigation adalah sebuah sebuah teknologi yang memungkinkan petani untuk menjadwalkan dengan tepat kapan tanaman perlu disiram dan berapa banyak air yang dibutuhkan tanaman. Dengan menggunakan sensor teknologi

Internet of Things, para petani dapat memantau tingkat kelembaban tanah di sekitar tanaman mereka, tingkat keasaman tanah, serta memantau kondisi cuaca atau suhu, sehingga mereka dapat menggunakan air dengan lebih efisien dan efektif [6].

2.7 *Internet of Things*

Internet of Things itu sebuah konsep dimana konektifitas internet dapat bertukar informasi dengan benda-benda yang ada disekelilingnya. IoT sangat mendukung integrasi, pengiriman dan analisis data yang dihasilkan oleh perangkat yang sudah saling terhubung dan menggunakan sensor. *Internet of Things* membayangkan sebuah perangkat yang terhubung dengan kegiatan manusia yang mempunyai tujuan untuk meningkatkan kualitas hidup. Mengatur sesuatu. Dan meringankan kebutuhan pengguna [8].

Internet of Things sudah banyak diterapkan di beberapa bidang keilmuan dan industri, seperti dalam bidang ilmu kesehatan, informatika, geografis dan beberapa bidang ilmu lain, berikut beberapa penelitian sudah dilakukan. Dalam dunia kesehatan kebutuhan informasi yang cepat dan tepat pun semakin dibutuhkan. Terutama yang dilakukan oleh Ihsan Fakhruzzaman menerapkan Internet of Things untuk Sport Science yang membantu pelatih dalam melakukan penilaian pada setiap tes yang dilaksanakan, kemudian memberikan informasi performa pemain selama tes berlangsung serta memberikan informasi hasil tes kelincahan [8].

2.8 Mikrokontroller

Menurut Ihsan Fakhruzzaman mikrokontroller merupakan sebuah piranti yang dirancang untuk kebutuhan umum yang berfungsi untuk mengontrol kerja mesin atau sistem yang memakai program dan disimpan dalam ROM biasa disebut dengan mikrokontroler [8].

Mikrokontroler pada dasarnya adalah komputer dalam satu chip, yang di dalamnya terdapat mikroprosesor, memori, jalur Input/Output (I/O) dan perangkat pelengkap lainnya. Kecepatan pengolahan data pada mikrokontroler lebih rendah jika dibandingkan dengan PC. Pada PC kecepatan mikroprosesor yang digunakan saat ini telah mencapai orde GHz, sedangkan kecepatan operasi mikrokontroler

pada umumnya berkisar antara 1 – 16 MHz. Begitu juga kapasitas RAM dan ROM pada PC yang bisa mencapai orde Gbyte, dibandingkan dengan mikrokontroler yang hanya berkisar pada orde byte/Kbyte [9].

Sistem yang menggunakan mikrokontroler sering disebut sebagai embedded system atau dedicated system. Embeded system adalah sistem pengendali yang tertanam pada suatu produk, sedangkan dedicated system adalah sistem pengendali yang dimaksudkan hanya untuk suatu fungsi tertentu. Sebagai contoh, printer adalah suatu embedded system karena di dalamnya terdapat mikrokontroler sebagai pengendali dan juga dedicated system karena fungsi pengendali tersebut berfungsi hanya untuk menerima data dan mencetaknya. Hal ini berbeda dengan suatu PC yang dapat digunakan untuk berbagai macam keperluan, sehingga mikroprosesor pada PC sering disebut sebagai general purpose microprocessor (mikroprosesor serba guna). Pada PC berbagai macam software yang disimpan pada media penyimpanan dapat dijalankan, tidak seperti mikrokontroler hanya terdapat satu software aplikasi [9].

Penggunaan mikrokontroler antara lain terdapat pada bidang-bidang berikut ini.

1. Otomotif : Engine Control Unit, Air Bag, fuel control, Antilock Braking System, sistem pengaman alarm, transmisi automatik, hiburan, pengkondisi udara, speedometer dan odometer, navigasi, suspensi aktif.
2. perlengkapan rumah tangga dan perkantoran : sistem pengaman alarm, remote control, mesin cuci, microwave, pengkondisi udara, timbangan digital, mesin foto kopi, printer, mouse.
3. pengendali peralatan di industri.
4. robotika.

2.9 Arduino Uno

Dalam jurnal yang ditulis oleh Andi Adriansyah dan Oka Hidyatama Arduino UNO adalah sebuah board mikrokontroler yang didasarkan pada ATmega328. Arduino UNO mempunyai 14 pin digital input/output (6 di antaranya dapat digunakan sebagai output PWM), 6 input analog, sebuah osilator Kristal 16 MHz, sebuah koneksi USB, sebuah power jack, sebuah ICSP header, dan sebuat

tombol reset. Arduino UNO memuat semua yang dibutuhkan untuk menunjang mikrokontroler, mudah menghubungkannya ke sebuah computer dengan sebuah kabel USB atau mensuplainya dengan sebuah adaptor AC ke DC atau menggunakan baterai untuk memulainya [10].

Menurut buku yang ditulis oleh Abdul Kadir Arduino adalah nama keluarga papan mikrokontroler yang awalnya dibuat oleh perusahaan Smart Project. Salah satu contoh tokoh penciptanya adalah Massimo Banzi. Papan ini merupakan perangkat keras yang bersifat open source sehingga boleh dibuat oleh siapapun. Arduino dibuat dengan tujuan untuk memudahkan berbagai peralatan yang berbasis mikrokontroler, misalnya :

1. Pemantauan ketinggian air di waduk
2. Pelacakan lokasi mobil
3. Penyiraman tanaman otomatis
4. Otomatisasi akses pintu ruangan
5. Pendeteksi keberadaan orang untuk pengambilan keputusan

Berbagai jenis kartu Arduino tersedia, antara lain adalah Arduino Uno, Arduino Diecimila, Arduino Duemilanove, Arduino Leonardo, Arduino Mega, dan Arduino Nano. Walaupun terdapat berbagai jenis kartu Arduino, secara prinsip pemrograman yang diperlukan menyerupai. Hal yang membedakan adalah kelengkapan fasilitas dan pin-pin yang perlu digunakan [11].

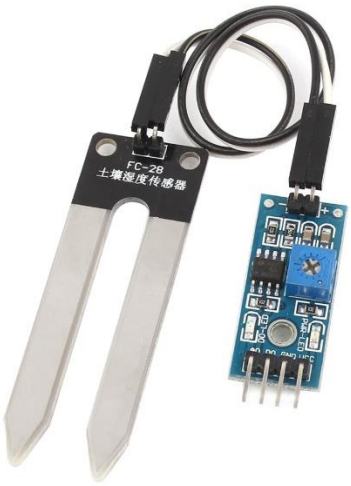
Gambar 2.2 Arduino Uno
(Sumber : https://www.arrow.com/en/products/a000073/arduino-corporation)

2.10 Sensor

Menurut buku yang ditulis oleh Fajar Wicaksono, Sensor merupakan sebuah alat yang digunakan untuk mendeteksi adanya perubahan lingkungan fisik maupun kimia. Variabel keluaran yang diubah sensor adalah Tranduser yang merupakan pengubah besaran mekanis, magnetis, panas, sinar dan kimia [12].

2.10.1 *Soil Moisture* Sensor

Sensor Kelembaban tanah atau hygrometer biasanya digunakan untuk mendeteksi kelembaban tanah [12]. Modul *soil moisture* sensor FC-28 terdiri dari dua bagian, yaitu bagian probe dan bagian papan elektronik seperti yang ditunjukan pada gambar 2.3.



Gambar 2.3 *Soil Moisture* Sensor [12]

Modul Sensor ini sudah dilengkapi dengan potensiometer yang berguna untuk mengatur sensitivitas sensor dan threshold pada keluaran digital [12]. Tabel 2.3 dibawah ini menunjukan spesifikasi dari *soil moisture* sensor FC-28.

Tabel 2.3 Spesifikasi *Soil Moisture* Sensor FC-28

Spesifikasi	Keterangan
Input Tegangan	3.3-5V
Output Tegangan	0-4.2V
Input Arus	35mA
Sinyal Output	Analog dan Digital

Untuk melakukan pengukuran kelembaban tanah, probe pada sensor ini dimasukan ke dalam tanah. Probe ini bertindak sebagai resistor variabel. Output tegangan dari modulsensor ini tergantung dari kandungan air dalam tanah. Ketika air semakin sedikit maka output tegangan dari modul sensor ini akan meningkat, sedangkan ketika kandungan air didalam tanah semakin banyak maka output tegangan dari sensor akan menurun [12]. Berikut ini adalah perhitungan untuk mengolah data dari sensor:

```
Kelembaban Tanah = (100 – ((HasilPembacaan/1023.00)*100));
```

2.10.2 Motor Servo

Motor Servo adalah jenis motor yang memiliki tiga kabel. Masing-masing digunakan sebagai catu daya, ground, dan kontrol. Kabel kontrol digunakan untuk menentukan motor dalam memutar rotor kearah posisi tertentu. Biasanya, rotor hanya berputar hingga 200^{o}. Namun, ada pula yang mampu berputar sebesar 360^{o}. Motor servo biasa digunakan untuk menggerakkan lengan robot atau memutar pada alat ukur yang bersifat analog [11].

Gambar 2.4 Motor Servo [11]

Motor ini dilengkapi dengan tiga kabel berwarna merah, oranye, dan cokelat. Dalam hal ini, kabel merah dihubungkan dengan catu daya 5V, kabel orange dihubungkan ke pin digital yang mendukung PWM, dan Kabel cokelat dihubungkan ke ground [11].

Tabel 2.4 Spesifikasi Motor Servo

No	Spesifikasi	Keterangan
1	Power	3.0 - 7.2 V
2	Ukuran	32.3 x 12.5 x 29.7 mm
3	Kecepatan Operasi	0.12sec/60 derajat
4	Berat	9 gram

2.10.3 *Rain Sensor*

Rain sensor merupakan sensor yang berfungsi untuk mendeteksi hujan turun atau tidak. Intinya sensor ini jika terkena air pada papan sensornya maka resistansinya akan berubah, semakin banyak semakin kecil dan sebaliknya. Untuk pengaplikasiannya sensor ini dapat digunakan untuk jemuran otomatis jadi ketika hujan turun sensor mendeteksi dan akan memberikan peringatan atau untuk tambahan dapat digunakan penutup yang dapat melindungi jemuran pada saat hujan. Untuk jenisnya di pasaran terdapat FC-37 dan YL-83.

Gambar 2.5 *Rain Sensor* [12]

Tabel berikut menjelaskan spesifikasi dari *rain sensor*, Tabel 2.5 menunjukan spesifikasi dari *rain sensor*.

Tabel 2.5 Spesifikasi *rain sensor*

No	Status	Keterangan
1	Ukuran	54x40x1.5mm
2	Power	3.3V - 5V
3	Sensor tipe	Analog
4	IC comparator	LM393

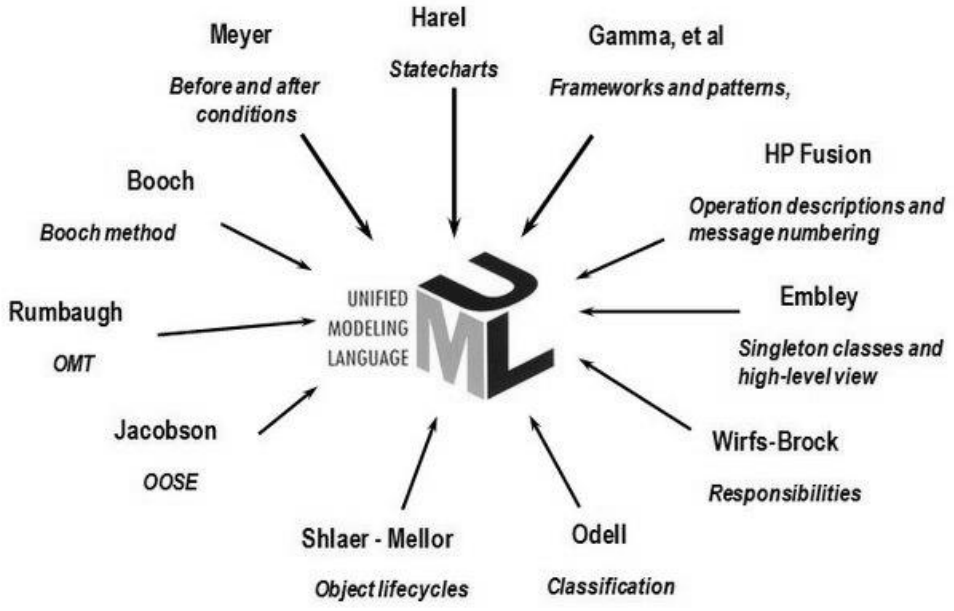
2.11 Unified Modeling Language (UML)

Menurut buku yang ditulis oleh Munawar *Unified Modeling Language* (UML) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa permodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atau svisi mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancagan mereka dengan yang lain [13].

Unified Modeling Language (UML) merupakan kesatuan dari Bahasa permodelan yang dikembangkan booch, *Object Modeling Technique* (OMT) dan *Object Orented Software Engineering* (OOSE) [13]. Metode ini menjadikan proses analisis dan design kedalam empat tahapan iterative, yaitu identifikasi kelas-kelas, dan objek-objek, identifikasi semantic dari hubungan objek dan kelas tersebut,

perincian *interface* dan implementasi. Keunggulan metode Booch adalah pada detail dan kayanya dengan notasi dan elemen, permodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan entity-relationship. Tahapan utama dalam metodologi ini adalah nalisis, design sistem, design objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung konsep OO. Metode OOSE dari Jacobson lebih memberi menekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, design dan implementasi, dan model pengujian. Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya [13]. Gambar 2.6 berikut adalah unsur-unsur yang membentuk UML.



Gambar 2.6 Unsur-unsur pembentuk UML [13]

Sebagai sebuah notasi grafis yang relative sudah dibakukan (*open standard*) dan dikontrol oleh OMG (*Object Management Group*) mungkin lebih dikenal sebagai badan yang berhasil membakukan CORBA (Common Object Request

Broker Architecture), UML menawarkan banyak keistimewaan. UML tidak hanya dominan dalam penotasian di lingkunan OO tetapi juga popular di luar lingkungan OO. Paling tidak ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan Bahasa pemograman. Sebagai sebuah sketsa, UML bisa berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dari sistem. Dengan demikian semua anggota tim akan mempunyai gambaran yang sama tentang suatu sistem. UML bisa juga berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detil. Dengan cetak biru ini maka akan bisa diketahui informasi detail tentang coding program (*forward engineering*) atau bahkan membaca program dan menginterpretasikannya kembali kedalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana code program yang tidak terdokumentasi asli hilang atau bahkan elum dibuat sama sekali. Sebagai bahasa pemograman, UML dapat menterjemahkan diagram yang ada di UML menjadi code program yang siap untuk di jalankan [13].

2.11.1 Diagram UML

UML menyediakan berbagai macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek [13]. Namun, pada penelitian ini hanya menggunakan 4 macam diagram saja untuk memodelkannya. Yaitu:

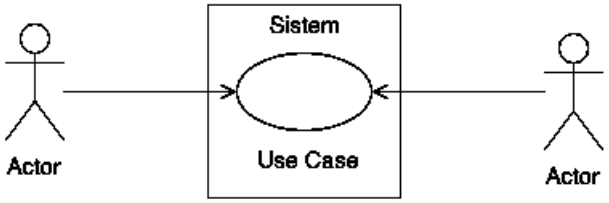
1. *Use Case Diagram*

   Use case diagram adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, system yang lain, perangkat keras atau urutan waktu [13]. Dengan demikian secara singkat bisa dikatan *use case* adalah serangkaian *scenario* yang digabungkan Bersama-sama oleh tujuan umum pengguna.

   Use case adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu system dari sudut pandangnya.

Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud menggunakan sebuah system. Karena system pengembangan tradisional sering cerboh dalam melakukan analisis, akibatnya pengguna seringkali susah menjawabnya tatkala dimintai masukan tentang sesuatu [13].

Diagram use case menunjukan 3 aspek dari sistem yaitu : *actor*, *use case* dan *sistem* atau *sub sistem boundary*. *Actor* mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan use case. Gambar 2.7 mengilustrasikan *actor, use case* dan *boundary*.

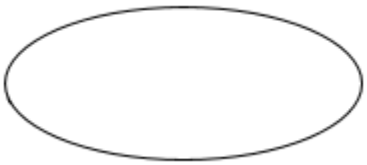


Gambar 2.7 *Use Case Model* [13]

Berikut ini adalah bagian dari sebuah use case diagram :

a. Use Case

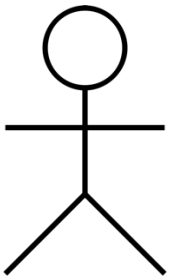
Use case adalah abstraksi dari interaksi antarasistem dengan *actor*. Oleh karena itu sangat penting untuk memilih abstraksi yang cocok. *Use case* dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ‘apa’ yang dikerjakan software aplikai, bukan ‘bagaimana’ software aplikasi mengerjakannya. Setiap *user case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama. Gambar 2.8 menunjukan bentuk *Use Case* dalam UML.



Gambar 2.8 *Use Case* [13]

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 2.9 menunjukan bentuk *actor* dalam UML.



Gambar 2.9 *Actor* [13]

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.

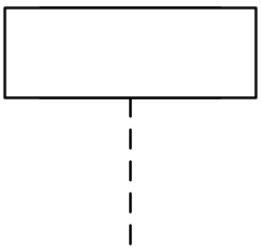
d. Generalisasi antara 2 *use case*.

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. *Sequance Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukan sejumlah contoh obyek dan *message* (pesan) yang diletakan diantara obyek-obyek ini di dalam use case. Komponen utama *sequence diagram* terdiri atas objek yang ditulisan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukan dengan *progress vertical* [13].

Objek diletakan di detak bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram, istilah objek dikenal juga dengan *participant*, setiap *participant* terhubung dengan garis titik-titik yang disebut *lifeline*. Sepanjang *lifeline* ada kotak yang disebut *activation*, *activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation* [13].



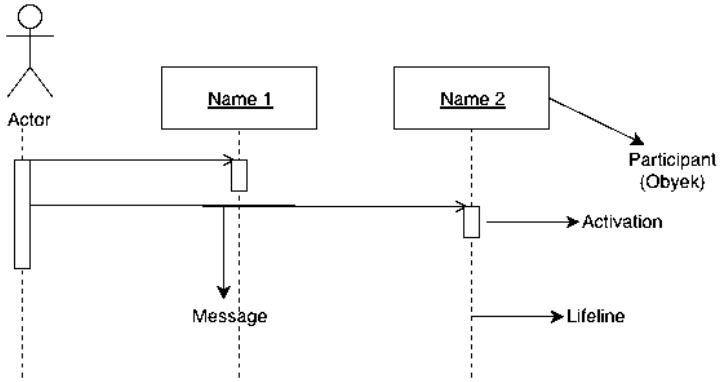
Gambar 2.10 *Participant* pada sebuah *sequence diagram* [13]

Sebuah *message* bergerak dari satu *participant* ke *participant* yang lain dan dari satu *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri. Sebuah *message* bisa jadi simple, *synchronous* atau *asynchronous*. *Message* yang simple adalah sebuah perpindahan (*transfer*) *control* dari satu *participant* ke *participant* yang lainnya. Jika sebuah *participant* mengirimkan sebuah message *synchronous*, maka jawaban atas *message* tersebut akan ditunggu sebelum

diproses dengan ururssannya. Namun jika message message *asynchronous* yang dikirimkan, maka jawaban atas *message* tersebut tidak perlu ditunggu.

Gambar 2.11 Simbol-simbol *message* [13]

Time adalah diagram yang mewakili waktu pada arah *vertical*. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijadikan terlebih dahulu dibanding *message* yang lebih dekat ke bawah. Gambar 2.12 menunjukan esensi *symbol* dari *sequence diagram* dan *symbol* kerjanya secara bersama-sama.



Gambar 2.12 Simbol-simbol yang ada pada *sequence diagram* [13]

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.

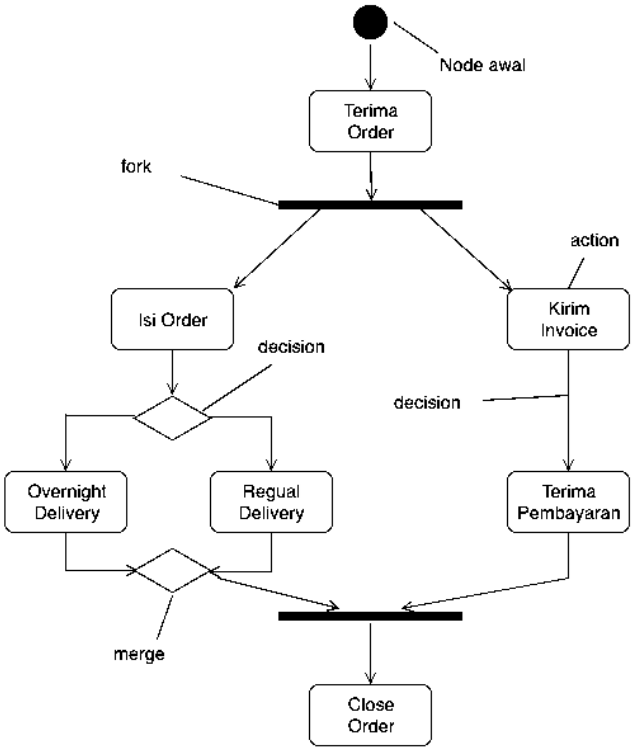
g. *Loop*

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

3. *Activity Diagram*

Activity diagram adalah bagian penting dari UML yang menggambarkan aspek dinamis dari sistem. Logika procedural, proses bisnis dan aliran kerja suatu bisnis bisa dengan mudah dideskripsikan dalam *activity diagram*. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukug perilaku pararel sedangkan *flowchart* tidak bisa. Tujuan dari activity diagram adalah untuk menangkap tingkah laku dinamis dari sistem dengan cara menunjukan aliran pesan dari satu aktifitas ke aktifitas lainya. Secara umum activity diagram digunakan untuk menggambarkan diagram alir yang terdiri dari banyak aktifitas dalam sistem dengan beberapa fungsi tambahan seperti percabangan, aliran pararel, swim lane, dan sebagainya. Sebelum menggambarkan sebuah sctivity diagram, perlu adanya pemahaman yang jelas tentang elemen yang akan digunakan di activity diagram. Elemen tuama dalam activity diagram adalah aktifitas itu sendiri. Aktifitas adalah fungsi yang dilakukan oleh sistem Setelah akitifitas terindentifikasi, selanjutnya yang perlu diketahui adalah bagaimana semua elemen tersebut berasosiasi dengan constraint dan kondisi. Langa selanjutnya perlu penjabaran tata letak dari keseluruhan

aliran agara bisa ditransofrmasikan ke *activity diagram* [13]. Gambar 2.13 menunjukan contoh *activity diagram* sederhana.



Gambar 2.13 Contoh *activity diagram* sederhana [13]

Berikut ini merupakan komponen dalam *activity diagram*, yaitu :

a. *Activity node*

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

b. *Activity edge*

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

c. *Initial state*

   Bentuk lingkaran berisi penuh melambangkan awal dari suatu proses.

d. *Decision*

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

e. *Fork*

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

f. *Join*

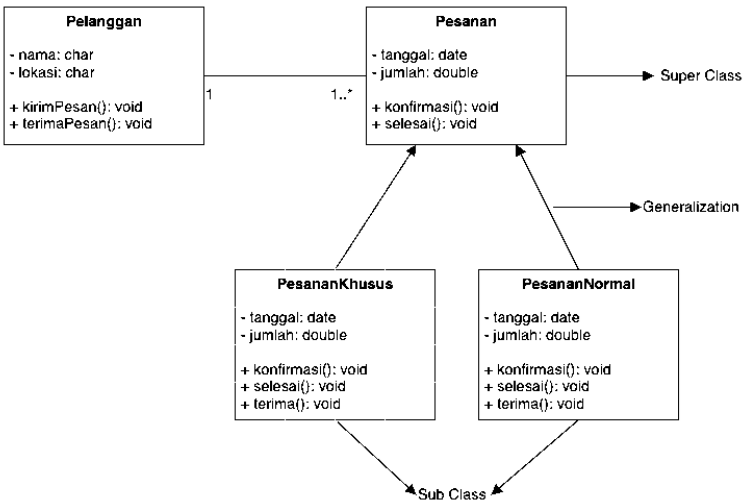
Satu bar hitam dengan dua atau lebih *activity node* yang masuk beserta satu *activity node* yang keluar, tercatat pada akhir dari proses secara bersamaan. Semua *actions* yang menuju *join* harus lengkap sebelum proses dapat berlanjut.

g. *Final state*

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

4. *Class Diagram*

Class diagram adalah diagram statis. Ini mewakili pandangan statis dari suatu aplikasi. Class diagram tidak hanya digunakan untuk memvisualisasikan, menggambarkan,o dan mendokumentasikan berbagai aspek sistem tetapi juga membangun kode eksekusi dari aplikasi perangkat lunak. *Class diagram* menggambarkan *atribut*, *operation* dan juga *constraint* yang terjadi pada sistem. *Class diagram* banyak digunakan dalam pemodelan sistem OO karena mereka adalah satu-satunya diagram UML, yang dapat dipetakan langsung dengan bahasa berorientasi objek. Class diagram menunjukan koleksi Class, antarmuka, asosiasi, kolaborasi, dan *constraint*. Dikenal juga sebagai diagram structural [13]. Gambar 2.14 menunjukan contoh *class diagram* sederhana.



Gambar 2.14 Contoh *class diagram* sederhana [13]

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.

2.12 Android

Android adalah sistem operasi untuk perangkat bergerak (mobile) yang awalnya dikembangkan oleh Android Inc. Salah satu pencipta dari Android adalah Andy Rubin, yang kini sering disebut sebagai “Bapak Android”. Pada tahun 2005, Google secara resmi telah membeli Android. Sehingga sejak saat itu, pengembangan Android sepenuhnya berada di tangan Google hingga saat ini.

Namun Google tetap merilis kode sumber (source code) secara terbuka, sehingga Android termasuk dalam software open source. Yang artinya, setiap orang di seluruh dunia juga dapat berkontribusi untuk mengembangkan Android. Jadi jika kita simpulkan, pengertian Android menurut para ahli adalah sebuah sistem operasi yang dikembangkan khusus untuk perangkat smartphone dan tablet [14]. Hingga kini Android masih terus dikembangkan dan berikut ini adalah versi-versi Android dari yang paling lama hingga yang terbaru:

- Android Cupcake
- Android Donut
- Android Eclair
- Android Froyo
- Android Gingerbread
- Android Honeycomb
- Android Ice Cream Sandwich
- Android Jelly Bean
- Android KitKat
- Android Lollipop
- Android Marshmallow
- Android Nougat
- Android Oreo
- Android Pie

Jika diperhatikan, semua nama versi Android diatas menggunakan nama-nama makanan penutup. Dan secara berurutan secara alfabet. Ini memang sudah jadi “tradisi” bagi Google dalam pemberian nama versi untuk Android.

2.12.1 Sejarah Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, middleware dan aplikasi. Android menyediakan *platform* terbukabagi parapengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34

perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android Bersama Open Handset Alliance menyatakan mendukung pengembangan open source pada perangkat mobile. Di lain pihak, Google merilis kode-kode Android dibawah lisensi Apache, sebuah lisensi perangkat lunak dan open platform perangkat seluler.

Didunia ini terdapat dua jenis sistributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari google atau google mail service (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD).

Sekitar September 2007 Google mengenalkan *Nexus one* salah satu jenis smartphone yang menggunakan Android sebagai sistem operasinya. Telepon seluler ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2010. Pada 9 Desember 2008, diumumkan anggota baru yang tergabung dalam program kerja Android ARM Holdings, Atheros Communications, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka, Android, perangkat mobile yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru.

Pada masa saat ini sebagian besar vendor-vendor smartphone sudah memproduksi smarphone berbasis android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, WayteQ, Sony Ericsson, LG, Acer, Philips, T-Mobile, Nexian, IMO, Asus dan masih banyak lagi vendor smartphone di dunia yang memproduksi android. Hal ini, karena android itu adalah sistem operasi yang open source sehingga bebas didistribusikan dan dipakai oleh vendor manapun.

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android menjadi pesaing utama dari Apple pada sistem operasi Table PC. Pesatnya pertumbuhan Android selain faktor yang disebutkan di atas adalah karena Android itu sendiri adalah *platform* yang sangat lengkap baik itu sistem operasinya, Aplikasi

dan Tool Pengembangan, Market aplikasi android serta dukungan yang sangat tinggi dari komunitas *Open Source* di dunia, sehingga android terus berkembang pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia [15].

2.12.2 Kelebihan Android

Android memiliki beberapa kelebihan yang menjadikannya populer dan banyak digunakan [14]. Berikut ini adalah beberapa kelebihan Android:

- Antarmuka yang mudah digunakan

  Tidak butuh waktu yang lama bagi seorang pemula untuk dapat menguasai penggunaan ponsel berbasis Android. Hal ini tentu menjadi kelebihan tersendiri bagi Android.
- Tampilan yang dapat diubah

  Tidak seperti *IOS*, tampilan Android dapat dengan mudah dikustomisasi. Mulai dari ikon, tema, wallpaper, dan masih banyak lagi. Hal ini menjadikan Android sebagai sistem operasi yang tidak membosankan.
- Open Source

  Android bersifat *open source*. Sehingga dapat dikembangkan lebih jauh sesuai dengan kebutuhan para penggunanya.
- Terus Diperbarui

  Karena dikembangkan langsung oleh Google, Android secara rutin mendapatkan pembaruan atau update. Baik itu berupa penambahan fitur baru, pembaruan keamanan, dan masih banyak lagi.
- Banyak Aplikasi Dan Game Gratis

  Banyaknya aplikasi dan game yang bisa di download gratis dari Google Play Store.

2.12.3 Kekurangan Android

Selain memiliki kelebihan, Android juga memiliki kekurangan [14]. Berikut ini adalah beberapa kekurangan yang dimiliki oleh Android:

- Tidak Semua Smartphone Android Mendapatkan Update

  Kekurangan pertama yang sering dirasakan pengguna Android adalah tidak semua smartphone mendapatkan update. Karena walaupun Google rajin

memperbarui Android, semua update smartphone kembali lagi pada pabrikan.

- Terlalu Banyak Merk Dan Tipe

  Karena terlalu banyak tipe dan merk, membuat penggunaannya jadi tidak konsisten. Tidak seperti iPhone yang hanya memiliki 1 tipe saja dan dikembangkan oleh 1 pabrikan, yaitu Apple.

- Lag Dan Lemot

  Karena banyak merk dan tipe *smartphone* Android, maka spesifikasinya juga berbeda-beda. Smartphone Android yang memiliki spesifikasi rendah biasanya akan lebih mudah lag dan lemot.

2.12.4 Arsitektur Android

Secara garis besar Arsitektur android dapat dijelaskan dan digambarkan sebagai berikut [15]:

- *Applications dan Widgets*

  Applications dan Widgets ini adalah layer dimana kita berhubungan dengan aplikasi saja, di mana biasanya kita download aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di layer terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman java.

- *Applications Frameworks*

  Android adalah *"Open Development Platform" yaitu* Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi resources, menjalankan service background, mengatur alarm, dan menambahkan status notifications, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah kembali dapat menggunakan komponen yang sudah digunakan *(reuse).*

Sehingga bisa kita simpulkan *Applications Frameworks* ini adalah layer di mana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti *contentproviders* yang berupa sms dan panggilan telepon.

Komponen-komponen yang termasuk di dalam *Applications Frameworks* adalah sebagai berikut:

a) *Views*
b) *Content Provider*
c) *Resource Manager*
d) *Notification Manager*
e) *Activity Manager*

- Libraries

  Libraries ini adalah layer di mana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan di atas kernel, Layer ini meliputi berbagai library C/C++ inti seperti Libc dan SSI, serta:

  - Libraries media untuk pemutaran media audio dan video
  - Libraries untuk manajemen tampilan
  - Libraries Graphics mencakup SGL dan OpenGL untuk grafis 2D dan 3D
  - Libraries SQLite untuk dukungan database
  - Libraries SSL dsn WebKit terintegrasi dengan web browser dan security
  - Libraries LiveWebcore mencakup modern web browser dengan engine embeded web view
  - Libraries 3D yang mencakup implementasi OpenGL ES 1.0 API's

- Android Run Time

  Layer yang membuat aplikasi Android dapat dijalankan di mana dalam prosesnya menggunakan Implemantasi Linux. *Davlik Virtual Machine*

(DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian yaitu:

- Core Libraries: Aplikasi Android dibangun dalam bahasa java, sementara Davlik sebagai virtual mesinnya bukan Virtual Machine Java, sehingga diperlukan sebuah libraries yang berfungsi untuk menerjemahkan bahasa java/c yang ditangani oleh Core Libraries.
- Davlik Virtual Machine: Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, di mana merupakan pengembangan yang mampu membuat linux kernel untuk melakukan theading dan manajemen tingkat rendah.

- Linux Kernel

  Linux kernel adalah layer di mana inti dari operating sistem dari Android itu berada. Berisi file-file system yang mengatur sistem processing, memory, resource, drivers, dan sistem-sistem operasi android lainnya. Linux kernel yang digunakan android adalah linux kernel 2.6.

2.13 MIT APP Inventor

App Inventor for Android atau Google App Inventor, itulah namanya, aplikasi berbasis web open source yang awalnya dikembangkan oleh Google, dan saat ini dikelola oleh Massachusetts Institute of Technology (MIT). App Inventor memungkinkan pengguna baru untuk memprogram komputer untuk menciptakan aplikasi perangkat lunak bagi sistem operasi Android. App Inventor ini menggunakan antarmuka grafis, serupa dengan antarmuka pengguna pada Scratch, yang memungkinkan pengguna men-drag-and-drop objek visual untuk menciptakan aplikasi yang bisa dijalankan pada perangkat Android. Begitupun dengan coding, kita tidak perlu menulis kode program yang amat sangat panjang, cukup dengan men-drag-and-drop seperti halnya menyusun puzzle [16].

2.13 *ThingSpeak*

ThingSpeak adalah platform open source *Internet of Things* (IoT) aplikasi dan API untuk menyimpan dan mengambil data dari hal menggunakan protokol HTTP melalui Internet atau melalui Local Area Network. *ThingSpeak* memungkinkan pembuatan aplikasi sensor logging, aplikasi lokasi pelacakan, dan jaringan sosial hal dengan update status. *ThingSpeak* awalnya diluncurkan oleh ioBridge pada tahun 2010 sebagai layanan untuk mendukung aplikasi IoT. *ThingSpeak* telah terintegrasi dukungan dari numerik komputasi perangkat lunak MATLAB dari *MathWorks*. Memungkinkan *ThingSpeak* pengguna untuk menganalisis dan memvisualisasikan data yang diunggah menggunakan Matlab tanpa memerlukan pembelian lisensi Matlab dari MathWorks [17].

ThingSpeak memiliki hubungan dekat dengan MathWorks, Inc. Bahkan, semua dokumentasi *ThingSpeak* dimasukkan ke situs dokumentasi Matlab yang MathWorks dan bahkan memungkinkan terdaftar MathWorks akun pengguna login sebagai valid di situs *ThingSpeak*. Persyaratan layanan dan kebijakan privasi dari *ThingSpeak*.com adalah antara pengguna setuju dan MathWorks, Inc. Didalam *Thingspeak* nantinya data akan ditampilkan dalam bentuk grafik. Data tersebut didapat dari arduino yang telah dipasang sebagai monitor suatu keadaan seperti monitoring kenaikan dan penurunan suhu. Arduino nantinya akan mengirimkan data ke *web server (Thingspeak)* secara otomatis dengan bantuan jaringan internet [17].

1.14 Metode Pengujian

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

2.14.1 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*). 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. 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 [18]. 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* [18]. 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 durabilitas. 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 tiba-tiba, loncatan arus listrik dan putusnya komunikasi secara tiba-tiba.

4. *Training usuability test*

   Ketika sejumlah besar pengguna terlibat dalam sistem operasi, *training usability requirement* ditambahkan dalam agenda pengujian. Lingkup dari *training usability test* ditentukan oleh sumber yang dibutuhkan untuk melatih pekerja baru untuk memperoleh level perkenalan dengan sistem yang ditentukan atau untuk mencapai tingkat produksi tertentu. Detail dari pengujian ini, sama halnya dengan yang lain, didasarkan pada karakteristik pekerja. Hasil dari pengujian ini harus menginspirasi

rencana dari kursus pelatihan dan follow-up serta memperbaiki sistem operasi *software*.

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 [18]. 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 projek 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.