BAB 2
LANDASAN TEORI

2.1 Longsor

Berdasarkan definisi dari Badan Penanggulangan Bencana Nasional sesuai dengan Undang-Undang, longsor merupakan salah satu jenis gerakan massa tanah atau batuan, ataupun percampuran keduanya, menuruni atau keluar lereng akibat terganggunya kestabilan tanah atau batuan penyusun lereng [10]. Gerakan tanah atau lebih dikenal dengan istilah tanah longsor adalah suatu produk dari proses gangguan keseimbangan lereng yang menyebabkan bergeraknya massa tanah dan batuan ke tempat yang lebih rendah. Gaya yang menahan massa tanah di sepanjang lereng tersebut dipengaruhi oleh sifat fisik tanah, dan sudut dalam tahanan geser tanah yang bekerja di sepanjang lereng. Perubahan gaya-gaya tersebut ditimbulkan oleh pengaruh perubahan alam maupun tindakan manusia. Perubahan kondisi alam dapat diakibatkan oleh gempa bumi, erosi, kelembaban lereng karena penyerapan air hujan dan perubahan aliran permukaan. Pengaruh manusia terhadap perubahan gaya-gaya antara lain adalah penambahan beban pada lereng dan tepi lereng, penggalian tanah di tepi lereng dan penajaman sudut lereng. Tekanan jumlah penduduk yang banyak menempati tanah-tanah berlereng sangat berpengaruh terhadap peningkatan resiko longsor. Faktor-faktor yang mempengaruhi terjadinya gerakan tanah antara lain: tingkat kelerengan, karakteristik tanah, keadaan geologi, keadaan vegetasi, curah hujan/hidrologi dan aktivitas manusia di wilayah tersebut [11]**.**

2.2 Faktor Penyebab Terjadinya Longsor

Beberapa hal berikut ini adalah beberapa faktor yang berkaitan dengan penyebab terjadinya bencana longsor [12]:

1. Curah Hujan

   Intensitas hujan yang semakin meningkat akan memperbesar peluang terjadinya tanah longsor. Pada dasarnya ada dua tipe hujan pemicu

terjadinya longsoran, yaitu hujan deras yang mencapai 70 mm hingga 100 mm per hari [13] dan hujan kurang deras namun berlangsung terus-menerus selama beberapa jam hingga beberapa hari yang kemudian disusul dengan hujan deras sesaat. Seluruh kejadian longsor yang terjadi umunya terjadi setelah hujan hujan turun selama 1-2 jam dan disususl dengan hujan deras [14]**.** Pada saat musim kemarau, air yang terkunci di dalam tanah akan menguap dalam jumlah yang cukup besar. Hal ini kemudian meninggalkan permukaan tanah dalam keadaan berongga atau berpori yang dalam keadaan tertentu akan berujung pada retakan tanah. Pada saat masuk musim penghujan, air kemudian akan memenuhi retakan pada tanah. Selanjutnya tanah akan mengembang/merekah. Namun hal ini kemudian berbuntut pada munculnya gerakan lateral sebab air terakumulasi pada bagian dasar lereng. Kondisi ini rawan longsor terlebih pada lokasi tersebut jarang terdapat pepohonan sebagai penyerap air dan pengikat tanah [15]**.**

2. Lokasi Yang Terjal

Kelerengan menjadi faktor yang sangat penting dalam proses terjadinya tanah longsor. Pembagian zona kerentanan sangat terkait dengan kondisi kemiringan lereng. Kondisi kemiringan lereng lebih 15 derajat perlu mendapat perhatian terhadap kemungkinan bencana tanah longsor dan tentunya dengan mempertimbangkan faktor-faktor lain yang mendukung. Pada dasarnya sebagian besar wilayah di Indonesia merupakan daerah perbukitan atau pegunungan yang membentuk lahan miring. Namun tidak selalu lereng atau lahan yang miring berbakat atau berpotensi longsor. Potensi terjadinya gerakan pada lereng juga tergantung pada kondisi batuan dan tanah penyusun lerengnya, struktur geologi, curah hujan, vegetasi penutup dan penggunaan lahan pada lereng tersebut [15]**.** terdapat 3 tipologi lereng yang rentan untuk bergerak/ longsor, yaitu:

1. Lereng yang tersusun oleh tumpukan tanah gembur dialasi oleh batuan atau tanah yang lebih kompak.
2. Lereng yang tersusun oleh pelapisan batuan miring searah lereng.
3. Lereng yang tersusun oleh blok-blok batuan [15].

Tabel klasifikasi kemiringan dapat dilihat dalam Tabel 2.1 Tabel Klasifikasi Kemiringan Lereng

Tabel 2.1 Tabel Klasifikasi Kemiringan Lereng

Kelas	Kemiringan (%)	Klasifikasi
I	< 2	Datar
II	2 – 15	Kemiringan rendah
III	16 – 25	Kemiringan sedang
IV	26 – 40	Kemiringan tinggi
V	> 40	Curam

3. Getaran

Penyebab tanah longsor lainnya adalah getaran yang bisa saja bersumber dari gempa bumi, mesin, lalulintas kendaraan dan lain-lain. Akibatnya adalah tanah yang kemudian longsor, jalanan yang retak dan lain-lain [14]**.** Getaran dari gempa bumi merupakan faktor yang sering terjadi yang mengakibatkan terjadinya tanah longsor, rambatan getaran dari gempa bumi dapat mempengaruhi kestabilan lereng dan batuan pada lereng, skala getaran gempa bumi dapat digunakan untuk mengakomodir dampak yang diakibatkan, skala getaran gempa bumi dapat dilihat pada Tabel 2.2 Tabel Skala dampak Getaran Gempa Bumi BMKG

Tabel 2.2 Tabel Skala dampak Getaran Gempa Bumi BMKG

Skala SIG-BMKG	Deskripsi Sederhana	Deskripsi Rinci	Skala MMI	PGA (gal)
I	Tidak dirasakan	Tidak dirasakan atau dirasakan hanya oleh beberapa orang tetapi terekam oleh alat.	I – II	< 2,9
II	Dirasakan	Dirasakan oleh orang banyak tetapi tidak menimbulkan kerusakan. Benda-benda ringan yang digantung bergoyang dan jendela kaca bergetar.	III – V	2,9 – 88

Skala SIG-BMKG	Deskripsi Sederhana	Deskripsi Rinci	Skala MMI	PGA (gal)
III	Kerusakan ringan	Bagian non struktur bangunan mengalami kerusakan ringan, seperti retak rambut pada dinding.	VI	89 – 167
IV	Kerusakan sedang	Banyak retakan terjadi pada dinding bangunan sederhana, sebagian roboh, lepas. Hampir sebagian besar atap bergeser ke bawah atau jatuh. Struktur bangunan mengalami kerusakan ringan sampai sedang.	VII – VIII	168 – 564
V	Kerusakan Berat	Sebagian besar dinding bangunan permanen roboh. Struktur bangunan mengalami kerusakan.	IX – XII	> 564

4. Kondisi Tanah dan Jenis Tanah

Kondisi tanah dan jenis tanah pada wilayah longsor sangat berpengaruh terhadap terjadinya longsor di suatu wilayah, faktor ini menjadi pemicu utama terjadinya longsor, nilai permeabilitas tanah akan berpengaruh terhadap kerentan tanah terhadap air dan curah hujan yang berakibat pada perubahan stabilitas tanah penyusun pada lokasi rawan longsor. serta ketinggian dan kemiringan lereng yang berpengaruh pada stabilitas lereng, berikut adalah jenis tekstur tanah penyusun pada wilayah longsor pada Tabel 2.3 Tabel Jenis dan Teksur Tanah

Tabel 2.3 Tabel Jenis dan Teksur Tanah [4]

Tekstur Tanah	Porositas Efektif	Permeabilitas (mm/jam)	Permeabilitas (mm)
Pasir	0.471	235.6	96.2
Pasir Lempungan	0.401	59.8	119.6
Lempung Pasiran	0.412	21.8	215.3
Lempung	0.434	13.2	175.0
Lempung Liatan	0.390	2.0	408.9
Liat Pasiran	0.321	1.2	466.5
Liat Lempungan	0.423	1.0	577.7
Liat	0.385	0.6	622.5

Kemudian faktor kemiringan tanah yang berpengaruh terhadap nilai resapan tanah yang berpengaruh terhadap kedalaman penetrasi air kedalam tanah dapat dilihat pada Tabel 2.4 Tabel Nilai Resapan Tanah

Tabel 2.4 Tabel Nilai Resapan Tanah [4]

Kedalaman Bidang Pembasahan (m)				
Tekstur Tanah	**Kemiringan Lereng (derajat)**			
	10	**20**	**30**	**40**
Pasir	4,9	0,8	0,5	0,4
Lempung	10,2	2,8	1,8	1,4
Liat	79,1	26,4	17,3	14,4

Waktu perubahan stabilitas tanah pada area longsor dapat dilihat pada Tabel 2.5 Tabel Stabilitas Tanah (dalam jam)

Tabel 2.5 Tabel Stabilitas Tanah (dalam jam) [4]

Tekstur Tanah	**Tanah Pasir**				**Tanah Lempung**				**Tanah Liat**			
	Kemiringan Lereng				**Kemiringan Lereng**				**Kemiringan Lereng**			
Intensitas (mm/jam)	**10°**	**20°**	**30°**	**40°**	**10°**	**20°**	**30°**	**40°**	**10°**	**20°**	**30°**	**40°**
10,4	1,929	0,669	0,441	0,369	1,517	0,525	0,345	0,289	1,173	0,405	0,269	0,225
20	1,005	0,349	0,229	0,193	0,789	0,273	0,181	0,153	0,649	0,213	0,141	0,117
50	0,405	0,141	0,093	0,077	0,317	0,109	0,073	0,061	0,445	0,125	0,057	0,049
80	0,253	0,089	0,061	0,049	0,253	0,069	0,045	0,041	0,421	0,129	0,085	0,061
100	0,161	0,057	0,037	0,023	0,217	0,057	0,029	0,025	0,413	0,133	0,093	0,081

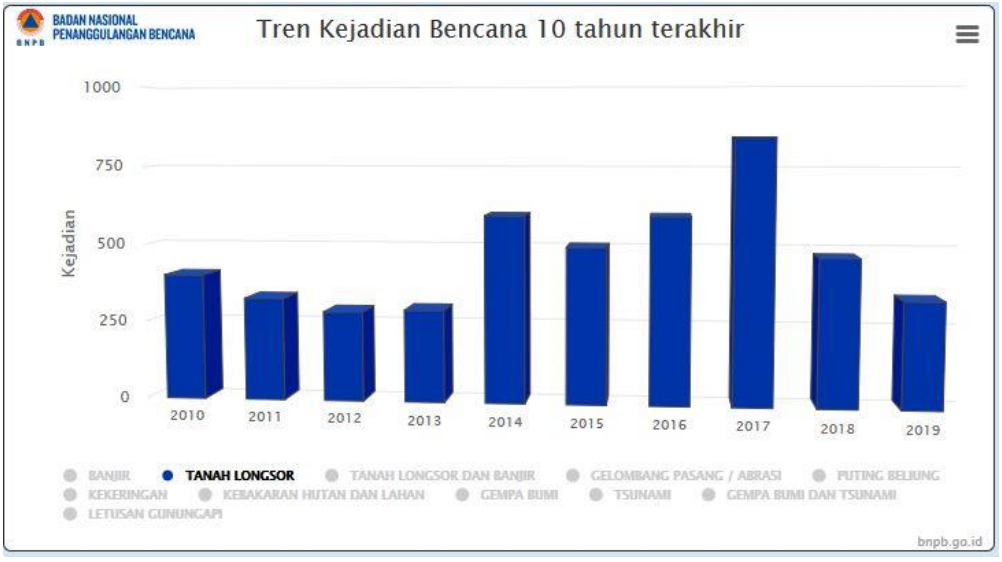
Faktor ketinggian lereng yang berpengaruh terhadap stabilitas lereng pada area rawan longsor dapat dilihat pada Tabel 2.6 Tabel Stabilitas Ketinggian Lereng

Tabel 2.6 Tabel Stabilitas Ketinggian Lereng

Kelas Lereng / *Slope Class* (%)	Ketinggian Lereng / Height (Meter)	Tingkat Stabilitas Lereng / *Slope Stability* (%)
0-2	0-10	20
>2-15	>10-20	30
>15-40	>20-100	50
>40	>100	80

2.3 Kejadian Longsor

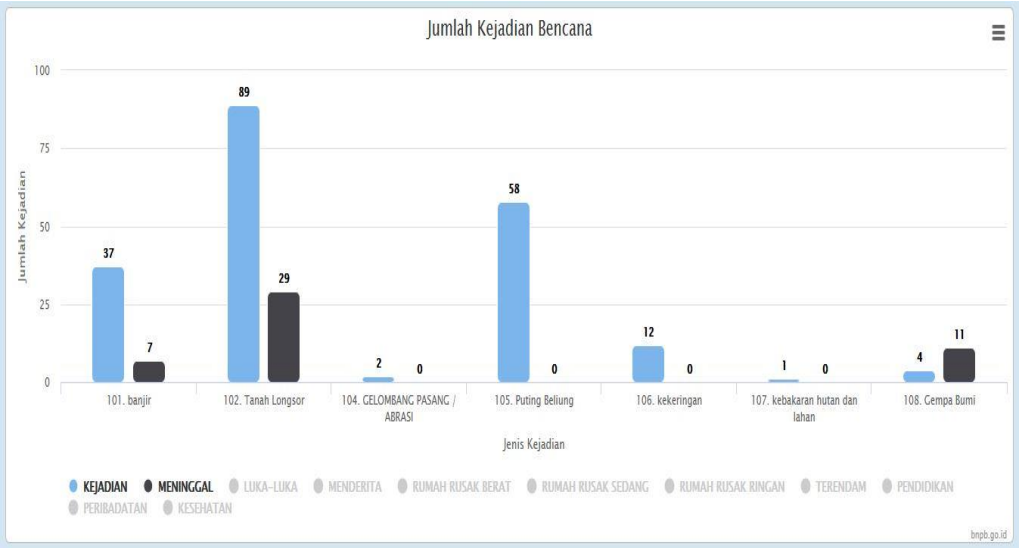
Data kejadian tanah longsor di Indonesia dari Badan Nasional Penanggulangan Bencana (BNPB) menunjukkan bahwa longsor merupakan salah satu kejadian bencana yang cukup sering terjadi di bandingkan dengan kejadian bencana lain. Berikut pada Gambar 2.1 Data kejadian longsor di Indonesia berdasarkan waktu menunjukan jumlah kejadian tanah longsor di Indonesia selama 10 tahun terakhir.



Gambar 2.1 Data kejadian longsor di Indonesia berdasarkan waktu [10]

Kemudian, berdasarkan sebaran wilayah Jawa Barat merupakan salah satu wilayah yang paling sering terjadi kejadian longsor setelah jawa tengah. Longsor yang terjadi di Jawa barat terutama di wilayah Tasikmalaya masih merupakan bencana yang menyebabkan banyak korban dan kerugian. Pada Gambar 2.4

berikut menunjukkan jumlah kejadian bencana dan jumlah koraban akibat bencana yang terjadi di wilayah Tasikmalaya selama 10 tahun terakhir :



Gambar 2.2 Data Kejadian Beserta Data Jumlah Korban dan Kerugian [10].

2.3.1 Status Bencana

Penentuan status bencana dilakukan dengan melakukan pengamatan pada pengujian alat yang akan dilakukan, penentuan status Normal, Siaga Waspada dan Awas, ditentukan dengan nilai besaran pada masing-masing sensor setelah dilakukan pengujian pada komponen perangkat keras, Status normal adalah status dimana keadaan pada kondisi tanah tidak mengalami perubahan dan pada kondidi normal, Status siaga adalah keadaan dimana adanya perubahan kondisi pada tanah sesuai dengan nilai yang telah ditentukan, status waspada adalah status dimana perubahan nilai pada kondisi tanah semakin membesar, dan status Awas adalah status dengan kondisi perubahan yang signifikan pada kondisi tanah, berikut adalah Tabel 2.7 Nilai Status Bencana :

Tabel 2.7 Nilai Status Bencana

Status	Kel. Tanah	Hujan	Getar
Normal	<60%	0%	0
Siaga	60%-69%	10%-55%	0
Waspada	70%-79%	56%-70%	0

Status	Kel. Tanah	Hujan	Getar
Awas	>=80%	>=71%	1

2.3.2 Monitoring

Monitoring adalah proses pengumpulan dan analisis informasi berdasarkan indikator yang ditetapkan secara sistematis dan berkelanjutan tentang kegiatan/program sehingga dapat dilakukan tindakan koreksi untuk penyempurnaan program/kegiatan itu selanjutnya [16]**.** Adapun tujuan dari monitoring adalah :

1. Memantau proses dan perkembangan pelaksanaan aktiftas program dengan mengacu pada indikator dan target yang telah ditetapkan dalam Workplan
2. mengidentifkasi masalah dan kesenjangan pada waktu pelaksanaan aktiftas
3. mengatasi masalah yang teridentifkasi dan mengantisipasi dampak daripermasalahan.

2.3.3 Pendeteksi Dini (Early Warning Detection)

Deteksi dini (*Early Warning*) dalam sistem kebencanaan merupakan bagian dari manajemen mitigasi bencana, Manajemen bencana pada dasarnya berupaya untuk menghindarkan masyarakat dari bencana baik dengan mengurangi kemungkinan munculnya *hazard* maupun mengatasi kerentanan. Terdapat lima model manajemen bencana yaitu:

1. *Disaster management continuum model*.

   Model ini mungkin merupakan model yang paling popular karena terdiri dari tahap-tahap yang jelas sehingga lebih mudah diimplementasikan. Tahap-tahap manajemen bencana di dalam model ini meliputi *emergency*, *relief*, *rehabilitation*, *reconstruction*, *mitigation*, *preparedness*, dan *early warning*.

2. *Pre-during-post disaster model*.

   Model manajemen bencana ini membagi tahap kegiatan di sekitar bencana. Terdapat kegiatan-kegiatan yang perlu dilakukan sebelum

bencana, selama bencana terjadi, dan setelah bencana. Model ini seringkali digabungkan dengan *disaster management continuum model*.

3. *Contract-expand model*.

   Model ini berasumsi bahwa seluruh tahap-tahap yang ada pada manajemen bencana (*emergency*, *relief*, *rehabilitation*, *reconstruction*, *mitigation*, *preparedness*, dan *early warning*) semestinya tetap dilaksanakan pada daerah yang rawan bencana. Perbedaan pada kondisi bencana dan tidak bencana adalah pada saat bencana tahap tertentu lebih dikembangkan (*emergency* dan *relief*) sementara tahap yang lain seperti *rehabilitation*, *reconstruction*, dan *mitigation* kurang ditekankan.

4. *The crunch and release model*.

   Manajemen bencana ini menekankan upaya mengurangi kerentanan untuk mengatasi bencana. Bila masyarakat tidak rentan maka bencana akan juga kecil kemungkinannya terjadi meski *hazard* tetap terjadi.

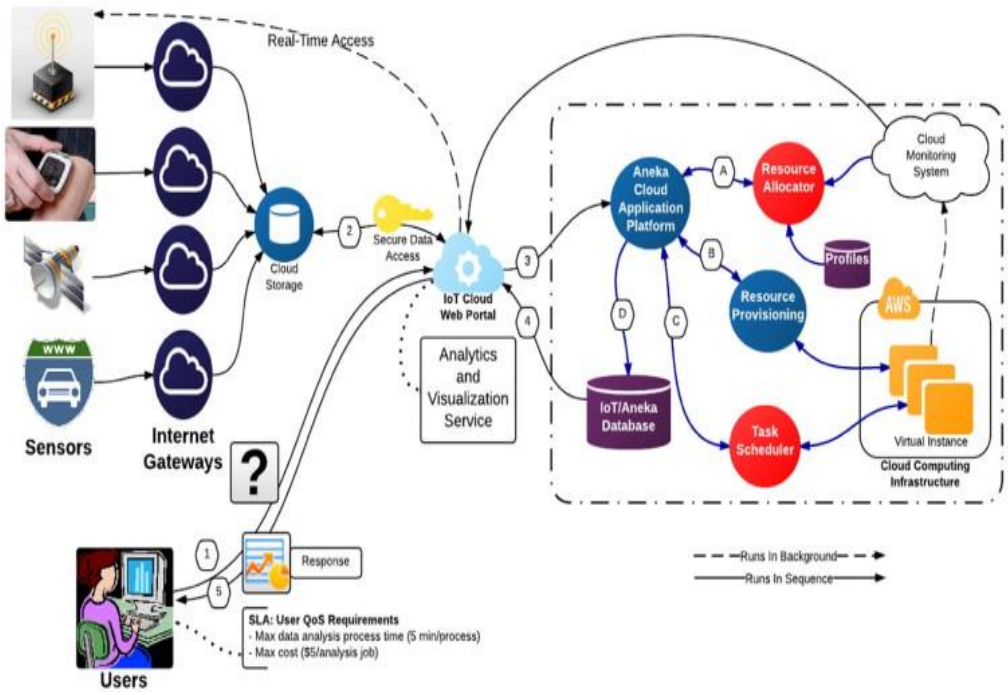
5. *Disaster risk reduction framework*.

   Model ini menekankan upaya manajemen bencana pada identifikasi risiko bencana baik dalam bentuk kerentanan maupun *hazard* dan mengembangkan kapasitas untuk mengurangi risiko tersebut.

2.4 *Internet Of Things*

Internet of Things, atau dikenal juga dengan singkatan IoT, merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang sudah meluas [17]. Metode yang digunakan oleh *Internet of Things* adalah nirkabel atau tanpa media kabel, jadi dapat dikendalikan secara otomatis tanpa mengenal jarak. Pengimplementasian *Internet of Things* sendiri biasanya selalu mengikuti keinginan *developer* atau perancang dalam mengembangkan sebuah aplikasi yang akan ia ciptakan, apabila aplikasinya itu diciptakan guna membantu monitoring sebuah ruangan maka pengimplementasian *Internet of Things* itu sendiri harus mengikuti alur diagram pemrograman mengenai sensor dalam sebuah rumah, berapa jauh jarak agar ruangan dapat dikontrol, dan kecepatan jaringan internet yang digunakan [17].

Internet of things mempunyai arsitektur yang terdiri dari *hardware* khusus, sistem *software*, *web API*, *protocol* yang bersama membuat lingkungan yang rapi dimana dasar alat pintar dapat terhubung ke internet semisal data sensor dapat di akses dan sistem kontrol dapat digerakan melalui internet, alat dapat terhubung ke internet dengan berbagai cara seperti *ethernet*, *WIFI*, *bluetooth*, dan sebagainya, alat mungkin juga tidak terhubung langsung dengan internet, namun di kelompokan dalam kluster. *IoT* sebenarnya *cyber* fisik sistem atau jaringan dari jaringan. Dengan jumlah besar hal / benda dan sensor / aktuator yang terhubung ke internet, besar-besaran dan dalam beberapa kasus aliran data *realtime* akan otomatis dihasilkan oleh hal-hal yang terhubung dan sensor. Dari semua kegiatan yang ada dalam *IoT* adalah untuk mengumpulkan data mentah yang benar dengan cara yang efisien; tapi lebih penting adalah untuk menganalisis dan mengolah data mentah menjadi informasi lebih berharga, berikut skema umum *Internet Of Things* yang ditunjukkan pada Gambar 2.3 Skema jaringan *Internet Of Things* :



Gambar 2.3 Skema jaringan *Internet Of Things* [17]

2.5 *Unified Modeling Language (UML)*

Unified Modeling Language(UML) adalah himpunan struktur dan teknik untuk pemodelan dan desain program berorientasi objek serta aplikasinya. UML adalah suatu bahasa yang digunakan untuk menentukan, menvisualisasikan, membangun, dan mendokumentasikan suatu Sistem Informasi [18]**.** UML (Unified Modeling Language) merupakan bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma (berorientasi objek). Pemodelan (modeling) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami [19]**.**Berdasarkan *Unified Modeling Language* beberapa definisi dan pendapat di atas maka dapat disimpulkan UML (*Unified Modeling Language*) adalah bahasa pemodelan yang digunakan untuk merancang, dan mendokumentasikan sebuh sistem perangkat lunak.

2.5.1 *Diagram UML*

Diagram UML digunakan untuk membangun model suatu sistem yang berbentuk diagram-diagram yang memudahkan baik pengguna maupun programmer dalam memahami urutan dalam sebuah sistem yang dirancang. ada 9 jenis diagram UML yang terdiri dari *Class Diagram, Package Diagram, Use Case Diagram, Sequence Diagram, Communication Diagram, State Chart Diagram, Activity Diagram, Component Diagram, Deployment Diagram*. Namun berdasarkan analisis yang penulis lakukan hanya menggunakan 4 jenis Diagram UML yaitu, *Class Diagram, Use Case Diagram, Sequance Diagram, dan Activity Diagram* [19].berikut adalah penjelasan dari masing-masing diagram :

1. *Use Case Diagram*

   Use case diagram digunakan untuk memodelkan semua bisnis proses berdasarkan perspektif pengguna sistem [20]. Berikut adalah langkah-langkah dari pembuatan use case :

   a. Aktor (*actor*)

      Menggambarkan suatu hal diluar sistem yang berinteraksi dengan sistem.

b. *Use case*

Adalah kegiatan/aktivitas yang disiapkan oleh sistem.Menekankan pada “apa” yang dikerjakan oleh sistem, bukan “bagaimana” sistem itu bekerja.

c. Relationship

Relationship adalah hubungan anatara use case dengan *actor,* jenis relationship dalam use case yaitu :

d. Asosiasi

Asosiasi digunakan untuk menggambarkan interaksi antara aktor dengan setiap *use case* tertentu. Relasi ini digambarkan sebagai garis penghubung aktor terhadap *use case* yang berelasi dengan aktor tersebut.

e. Generalisasi

Suatu relasi antara *use case* umum (induk) dan *use case* yang lebih spesifik (anak). Relasi Generalisasi memungkinkan *use case* yang lebih spesifik memiliki perilaku (*behaviour*) yang sama dengan *use case* yang lebih umum atau bisa disebut *use case* induk. Relasi generalisasi digambarkan dengan anak panah segitiga. *use case* yang terletak di sisi anak panah adalah *use case* induk dan yang terletak di sisi lainnya adalah *use case* yang mewarisi perilaku *use case* induk.

f. *<<include>> relationship*

Relasi include memungkinkan terjadinya penambahan perilaku (behaviour) ke dalam *use case* awal yang pada dasarnya *use case* ini tidak dapat berdiri sendiri tanpa adanya penambahan *use case* , dan *use case* awal tidak akan lengkap tanpa adanya *use case* tambahan ini. *use case* yang berada pada kepala anak panah adalah *use case* awal, dan pada sisi lain adalah *use case* penambah.

g. *<<extend>> relationship*

Relasi extend memungkinkan terjadinya penambahan beberapa behaviour (tingkah laku) ke dalam *use case* awal yang pada dasarnya *use case* tersebut sudah dapat berdiri sendiri tanpa adanya

penambahan. *Extend* digambarkan dengan anak panah yang mempunyai garis putus-putus. *use case* yang berada pada kepala anak panah adalah *use case* awal dan yang berada di lain sisi adalah *use case* tambahan.

2. *Class Diagram*

Class Diagram merupakan himpunan dari objek-objek yang sejenis. Sebuah objek memiliki keadaan sesaat (*state*) dan perilaku (*behavior*). *State* sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam *attribute/properties*. Sedangkan perilaku suatu objek mendefinisikan bagaimana sebuah objek bertindak/beraksi dan memberikan reaksi [21]. *Class* diagram menggambarkan relasi atau hubungan antar kelas dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam *class* diagram:

a. *Association*

Hubungan antar *class* yang statis. *Class* yang mempunyai relasi asosiasi menggunakan *class* lain sebagai atribut pada dirinya

b. *Aggregation*

c. Relasi

Relasi yang membuat *class* yang saling terikat satu sama lain namun tidak terlalu berkegantungan.

d. *Composition*

Relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan

e. *Dependency*

Hubungan antar *class* dimana *class* yang memiliki relasi dependency menggunakan *class* lain sebagai atribut pada method.

a. *Realization*

Hubungan antar *class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya.

f. Realization

Hubungan antar *class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya.

3. *Activity Diagram*

Activity Diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, *activity diagram* memainkan peran mirip diagram alir, tetapi perbedaan prinsip antara notasi diagram alir adalah *activity diagram* mendukung *behavior paralel*. *Node* pada sebuah *activity diagram* disebut sebagai *action*, sehingga diagram tersebut menampilkan sebuah *activity* yang tersusun dari *action* [22]. Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

4. *Sequence Diagram*

Sequence Diagram adalah grafik dua dimensi dimana obyek ditunjukkan dalam dimensi horizontal, sedangkan *lifeline* ditunjukkan dalam dimensi vertikal [21]. *Sequence diagram* menjelaskan secara detil urutan proses yang dilakukandalam sistem untuk mencapai tujuan dari use case. 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.

2.5.2 *Entity Relationship Diagram (ERD)*

Entity Relationship diagram (ERD) merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh *System Analyst* dalam tahap analisis persyaratan proyek pengembangan system. Sementara seolah-olah teknik diagram atau alat peraga memberikan dasar untuk desain *database* relasional yang mendasari sistem informasi yang dikembangkan. ERD bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk *database* [23].Peranan ERD di dalam proses pembuatan suatu sistem basis data sangatlah penting, karena di ERD tersebutlah dijelaskan tentang alur pemrosesan suatu data, mulai dari proses input hingga output.. Dalam pembentukan ERD terdapat 3 komponen yang akan dibentuk, yaitu entitas, relasi, dan atribut :

1. Entitas

   Entitas adalah objek yang menarik di bidang organisasi yang dimodelkan.

2. Relasi

   Suatu relasi atau hubungan adalah hubungan antara dua jenis entitas dan direpresentasikan sebagai garis lurus yang menghubungkan dua entitas.

3. Atribut

   Atribut memberikan informasi lebih rinci tentang jenis entitas. Atribut memiliki struktur internal berupa tipe data. Jenis-jenis atribut :

a. Atribut Key

Atribut Key adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data (*Row/Record*) dalam tabel secara unik. Dikatakan unik jika pada atribut yang dijadikan key tidak boleh ada baris data dengan nilai yang sama. Contoh : Nomor Pokok Mahasiswa (NPM), NIM, dan nomor pokok lainnya

b. Atribut simple

Atribut yang bernilai atomik (tidak dapat dipecah/dipilah lagi). Contoh Alamat, penerbit, tahun terbit, judul buku.

c. Atribut Multivalue

Nilai dari suatu attribut yang mempunyai lebih dari satu nilai (multivalue) dari atrribut yang bersangkutan. Contoh : dari sebuah buku, yaitu terdapat beberapa pengarang.

d. Atribut Composite

Atribut composite adalah suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu yang masih bisa dipecah lagi atau mempunyai *sub attribute*. Contoh : dari entitas nama yaitu nama depan, nama tengah, dan nama belakang

e. Atribut Derivatif

Atribut yang tidak harus disimpan dalam *database* Ex. Total. atau atribut yang dihasilkan dari atribut lain atau dari suatu *relationship*. Atribut ini dilambangkan dengan bentuk oval yang bergaris putus-putus. [23]

Selain itu, dalam ERD juga terdapat kardinalitas. Kardinalitas menjelaskan jumlah maksimum hubungan antara satu entitas dengan entitas lainnya.

1. *One to One* (1:1)

Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.

2. *One to many* (1:M)

Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.

3. *Many to Many* (M:M)

   Setiap entitas A dapat berhubungan dengan banyak entitas himpunan entitas B dan demikian pula sebaliknya. [23]

2.5.3 *Object Oriented (OO)*

Pemrograman berorientasi objek adalah pendekatan yang lebih berfokus pada cara objek berinteraksi untuk berkomunikasi dan berbagi informasi. Metode Ini mengubah kebiasaan pemrograman berorientasi prosedural konvensional di mana fokusnya adalah pada prosedur eksekusi [24]**.**

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas, metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek [25]. Berikut adalah konsep dalam *Object* Oriented :

1. *Class*

   Dalam *object* oriented programming, *class* (kelas) adalah blueprint atau *prototype*, yang mendefinisikan variabel dan method-method pada seluruh *object* tertentu untuk membuat anggota dirinya sendiri mengarah kepada *class* instance, *class object*, instance *object* atau simply *object*. Suatu *class* (kelas) mendefinisikan unsur anggota yang memungkinkan contoh *class* (kelas) ini memiliki perilaku dan kemampuan khusus sendiri.*Class* (Kelas) adalah struktur data yang berisi data-data (konstanta-konstanta dan *field-field*), fungsi-fungsi (method, property,event, indexer, operator, constructor,destructor,dan static constructor) dan *class* dalam *class*.*Class* (Kelas) adalah kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebuah *class* (kelas) adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi *object*. *Class*

(kelas) merupakan sarana pengkapsulan kumpulan data dan kumpulan *method java*, *class* (kelas) digunakan untuk membuat objek, dan berperan sebagai tipe data dari objek. Sebuah *class* (kelas) secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah *class* sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya. Sebuah *class* merupakan dasar dari modularitas dan struktur dalam pemrograman berorientasi *object*.

2. *Object*

Object adalah elemen dasar dari konsep pemrograman, merupakan sesuatu yang memiliki identitas (nama), pada umumnya juga memiliki data tentang dirinya maupun *object* lain dan mempunyai kemampuan untuk melakukan sesuatu dan bisa bekerja sama dengan objek lainnya.

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. *Inheritance*

Inheritance adalah proses pewarisan data dan method dari suatu kelas kepada kelas yang lain. Pewarisan ini bersifat menyeluruh, sehingga semua data dan method yang dimiliki oleh kelas asalnya akan diturunkan kepada kelas baru. Kelas yang mewarisi disebut kelas "SUPER (*super class*)" dan kelas yang diwarisi disebut "SUBKELAS (sub *class*)" [25]**.**

2.6 *MySQL*

MySQL merupakan sebuah perangkat lunak atau *software* sistem manajemen basis data SQL atau *DBMS Multithread* dan *multi user*. *MySQl* sebenarnya merupakan turunan dari salah satu konsep utama dalam *database* untuk pemilihan atau seleksi dan pemasukan data yang memungkinkan pengoperasian data dikerjakan secara mudah dan otomatis. *MySQL* diciptakan oleh Michael "Monty" Widenius pada tahun 1979, seorang programmer komputer asal Swedia yang mengembangkan sebuah sistem *database* sederhana yang dinamakan UNIREG yang menggunakan koneksi *low-level* ISAM *database engine* dengan *indexing*.dalam penelitian ini *MySQL* digunakan sebagai media penyimpanan *database*, berikut adalah kelebihan dari *MySql*:

1. *Free* atau gratis sehingga *MySQL* dapat dengan mudah untuk mendapatkannya.
2. *MySQl* stabil dan tangguh dalam pengoperasiannya
3. *My SQl* mempunyai sistem keamanan yang cukup baik
4. Sangat mendukung transaksi dan mempunyai banyak dukungan dari komunitas
5. Sangat fleksibel dengan berbagai macam program
6. Perkembangan dari *MySQl* sangat cepat

Berikut adalah Gambar 2.4 Logo *MySQL* :

Gambar 2.4 Logo *MySQL*

2.7 *Python*

Python adalah bahasa pemrograman interpreatatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Bahasa pemograman *Python* disebut sebagai bahasa yang kemampuan, menggabungkan kapabilitas, dan sintaksis kode yang sangat jelas, dan juga dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python dibuat dan dikembangkan lebih lanjut oleh Guido Van Rossum, yaitu seorang programmer yang berasal dari Negara Belanda. Dibuat dan dikembangkan di kota Amsterdam, Belanda pada tahun 1990. Pada tahun 1995 *Python* dikembangkan lagi agar lebih kompatibel oleh Guido Van Rossum.

Kemudian di awal tahun 2000 versi *Python* dikembangkan dan diperbaharui lagi sehingga kini Bahasa Pemrograman *Python* telah mencapai Versi 3. Sebenarnya awal mula dari kata ‘ *Python* ‘ Diambil oleh Guido Van Rossum dari sebuah acara televisi yang lumayan terkenal yang bernama Mothy *Python* Flying Circus, Yaitu sebuah acara sirkus yang disukai oleh Guido Van Rossum. *Python* mempunyai beberapa fitur unik di dalam bahasa pemrogramannya diantaranya sebagai berikut:

1. Bahasa pemrograman *python* memiliki tata bahasa dan script yang sangat mudah untuk dipelajari.
2. Bahasa pemrograman *python* memiliki sistem pengelolaan data dan memori yang otomatis.

3. Bahasa pemrograman *python* mempunyai modul yang baru dan selalu diperbaharui.
4. Bahasa pemrograman *python* memiliki banyak fasilitas pendukung dan memudahkan para penggunannya.
5. Bahasa pemrograman *python* sangat mudah dipahami seperti bahasa pemrograman lainnya.

Python merupakan salah satu bahasa pemrograman yang populer digunakan, berikut adalah kelebihan dari *Python* :

1. Mudah Dipelajari

   Mudah dipelajari sudah melekat sebagai salah satu kelebihan bahasa pemrograman *Python* diantara bahasa pemrograman lainnya. Bahasa pemograman *Python* ini memiliki sintaks-sintaks yang cukup sederhana dan mudah dimengerti. *Python* merupakan bahasa yang sangat dinamis, yang dibangun berdasarkan tingkat keterbacaan kode yang tinggi. Filososfi ini menjadikan bahasa pemrograman *Python* memiliki kelebihan seperti yang telah dijelaskan sebelumnya.

2. Mudah Digunakan

   Kelebihan bahasa pemrograman *Python* lainnya adalah bahasa pemrograman ini merupakan bahasa yang mudah untuk digunakan dalam mengembangkan sebuah produk, baik itu web, *software*, aplikasi web, maupun video game. Selain memiliki keterbacaan kode yang tinggi, sehingga kode mudah dipahami, bahasa pemrograman ini memiliki library yang sangat banyak dan luas. Berbagai macam jenis library ini memuat sangat banyak perlengkapan dan fungsionalitas yang dangat luar biasa, sehingga kemudahan membangun program menjadi salah satu yang ditawarkan oleh bahasa pemrograman tersebut.

3. Mendukung *Internet of Things* Dengan Baik

   Kelebihan bahasa pemrograman *Python* salah satunya adalah merupakan bahasa yang mendukung ekosistem *Internet of Things* dengan sangat baik.

Logo python dapat dilihat pada Gambar 2.5 Logo *Python* :

Gambar 2.5 Logo *Python*

2.8 *Javascript*

JavaScript adalah bahasa pemrograman *web* yang bersifat *Client Side Programming Language*. *Client Side Programming Language* adalah tipe bahasa pemrograman yang pemrosesannya dilakukan oleh *client*. Aplikasi *client* yang dimaksud merujuk kepada *web browser* seperti Google Chrome dan Mozilla Firefox. Bahasa pemrograman Client Side berbeda dengan bahasa pemrograman *Server Side* seperti PHP, dimana untuk *server side* seluruh kode program dijalankan di sisi *server*. Untuk menjalankan *JavaScript*, kita hanya membutuhkan aplikasi *text editor* dan *web browser*. *JavaScript* memiliki fitur: *high-level programming language, client-side, loosely tiped* dan berorientasi objek.

2.9 Jaringan Komputer

Jaringan komputer adalah himpunan "*interkoneks*" antara 2 komputer autonomus atau lebih yang terhubung dengan media transmisi kabel atau tanpa kabel (wireless) [26]. Dua komputer dikatakan terkoneksi apabila keduanya bisa saling bertukar data / informasi, berbagai resource yang dimiliki seperti *file*, *printer*, *media* penyimpanan (*hardisk*, *floppydisk*, *cd rom*, *flashdisk*, dll). Data yang berupa teks, audio, maupun video bergerak melalui media kabel atau tanpa kabel sehingga memungkinkan pengguna komputer dalam jaringan komputer yang dapat saling bertukar *file*/data, mencetak pada printer yang sama dan menggunakan *hardware* atau *software* yang terhubung dalam jaringan secara bersama-sama [26], jaringan komputer dibedakan menjadi 3 jenis, yaitu :

1. *LAN (Local Area Network)*

   Local Area Network (LAN) merupakan jaringan yang menghubungkan sejumlah komputer yang ada dalam suatu lokasi dengan area terbatas seperti ruang tamu atau gedung. *LAN* dapat menggunakan media komunikasi seperti kabel dan wireless.

2. *MAN (Metropolitan Area Network)*

   Jaringan yang menghubungkan komputer-komputer dalam satu kota, sampai dengan beberapa puluh kilometer. Cakupan daerahnya lebih luas dibandingkan dengan *LAN.*

3. *WAN (Wide Area Network)*

   Jaringan yang menghubungkan komputer-komputer dalam satu negara atau benua, sampai beberapa ratus kilometer.

2.9.1 *Website*

World Wide Web atau WWW atau juga dikenal dengan *website* adalah *Website* merupakan kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi suara,dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing masing dihubungkan dengan jaringan-jaringan halaman [27].dalam penelitian yang dilakukan website digunakan sebagai media informasi penyampaian data hasil monitoring dan deteksi dini.

2.9.2 *Web Server*

Web Server adalah komputer yang digunakan untuk menyimpan dokumen-dokumen *web*, komputer ini melayani permintaan dokumen *web* dari kliennya. *Web server* merupakan *software* yang memberikan layanan data, berfungsi menerima permintaan HTTP atau HTTPS dari *client* yang dikenal dengan *browser web* dan mengirimkan kembali hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk dokumen HTML [28], konsep *web server* antara lain:

1. *Web server* merupakan mesin aplikasi atau *software* yang beroperasi dalam medistribusikan *web page* ke *user*, tentu saja sesuai dengan permintaan *user* [28].
2. Hubungan antara *web server* dan *browser* internet merupakan gabungan atau jaringan komputer yang berada diseluruh dunia. Setelah terhubung secara fisik, *protocol* TCP/IP (*networking protocol*) yang memungkinkan semua komputer dapat berkomunikasi antar satu dengan lainnya. Pada saat aplikasi *browser* meminta data *web page* ke *server* maka instruksi permintaan data oleh *browser* tersebut dikemas dalam TCP yang merupakan *protocol transport* dan dikirim ke alamat yang merupakan *protocol* berikutnya yaitu *hyper text transfer protocol* (HTTP). Data yang diparsing dari *browser* ke *web server* disebut sebagai HTTP *request* yang meminta halaman *web* dan kemudian *web server* akan mencari data HTML yang dibutuhkan dan dikemas dalam TCP *protocol* kemudian dikirim kembali ke *browser*. Data yang dikirim dari *server* ke *browser* disebut sebagai HTTP *response*. Jika data yang diminta oleh *browser* tidak ditemukan pada *web server* maka akan menampilkan *error* pada *web page* yaitu *Error: 404 Page Not Found* [28].

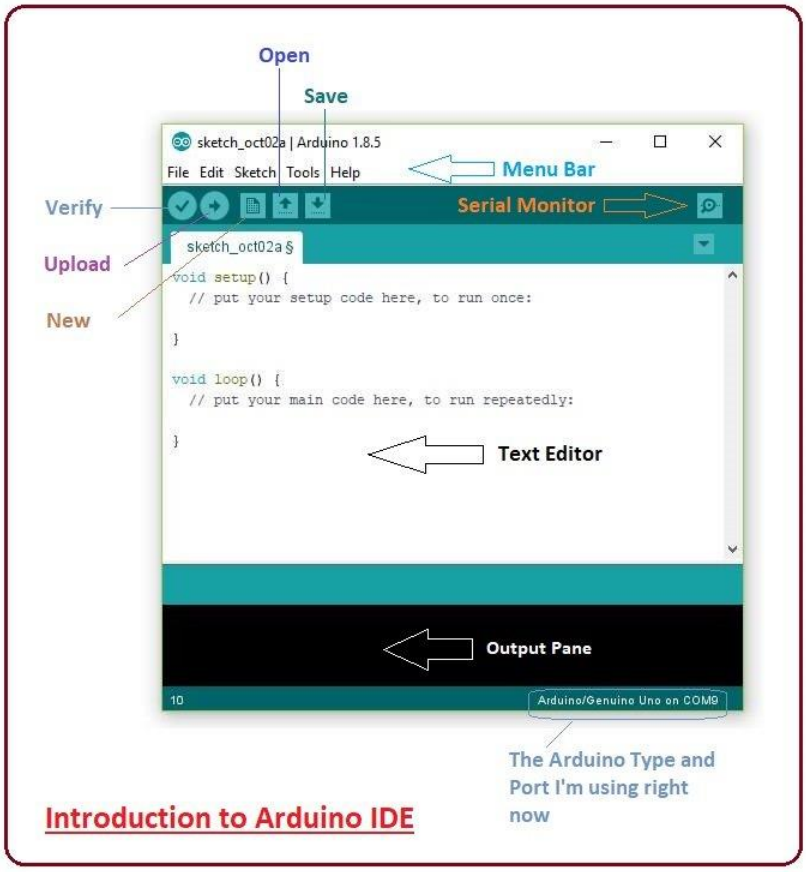
2.10 *Arduino IDE*

IDE merupakan kependekan dari *Integrated Developtment Enviroenment*, atau secara bahasa mudahnya merupakan lingkungan terintegrasi yang digunakan untuk melakukan pengembangan. Disebut sebagai lingkungan karena melalui *software* inilah *Arduino* dilakukan pemrograman untuk melakukan fungsi-fungsi yang dibenamkan melalui sintaks pemrograman. *Arduino* menggunakan bahasa pemrograman sendiri yang menyerupai bahasa C. Bahasa pemrograman *Arduino* (Sketch) sudah dilakukan perubahan untuk memudahkan pemula dalam melakukan pemrograman dari bahasa aslinya. Sebelum dijual ke pasaran, IC microcontroller *Arduino* telah ditanamkan suatu program bernama Bootlader yang berfungsi sebagai penengah antara compiler *Arduino* dengan microcontroller. *Arduino* IDE dibuat dari bahasa pemrograman *JAVA*. *Arduino IDE* juga

dilengkapi dengan library C/C++ yang biasa disebut Wiring yang membuat operasi *input* dan *output* menjadi lebih mudah. *Arduino IDE* ini dikembangkan dari *software Processing* yang dirombak menjadi *Arduino IDE* khusus untuk pemrograman dengan *Arduino*.

Program yang ditulis dengan menggunaan *Arduino Software (IDE)* disebut sebagai sketch. *Sketch* ditulis dalam suatu editor teks dan disimpan dalam file dengan ekstensi .ino. Teks editor pada *Arduino Software* memiliki fitur" seperti cutting/paste dan seraching/replacing sehingga memudahkan kamu dalam menulis kode program.

Pada *Software Arduino IDE*, terdapat semacam message box berwarna hitam yang berfungsi menampilkan status, seperti pesan *error, compile, dan upload* program. Di bagian bawah paling kanan *Software Arduino* IDE, menunjukan *board* yang terkonfigurasi beserta COM *Ports* yang digunakan. Tampilan Arduino IDE dapat dilihat pada Gambar 2.6 Tampilan *Arduino* IDE :



Gambar 2.6 Tampilan *Arduino* IDE

2.11 Mikrokontroller

Mikrokontroller adalah sebuah sistem komputer yang seluruh atau sebagian besar elemennya dikemas dalam satu chip IC, sehingga sering disebut single chip microcomputer. Mikrokontroler merupakan sistem komputer yang mempunyai salah satu atau beberapa tugas yang sangat spesifik [29]. *Microcontroller* dapat di artikan sebagai salah satu komponen terpenting dalam suatu sistem yang membutuhkan pengontrolan, dengan ukuran mikro yang tak ubahnya seperti sebuah komputer dalam ukuran mini, yang terdiri dari sebuah *Central Processing Unit* (CPU) sebagai pengolah data, memori untuk penyimpan data dan sarana input/output (I/O) yang terangkum dalam sebuah chip. Jenis-jenis *microcontroller* terbagi menjadi 3 keluarga besar yaitu keluarga MCS51, AVR dan PIC, pada umumnya jenis AVR yang sering digunakan, karena melimpahnya *tutorial* penggunaannya dan mudah untuk memprogramnya, AVR di kelompokkan menjadi 4 kelas, yaitu ATTiny, AT90Sxx, ATMega dan AT86RFxx. Banyak sistem minimum yang chip pemogramannya menggunakan AVR khususnya jenis ATMega, contohnya *Arduino* yang secara universal sudah banyak yang memakainya untuk *controller* sebuah sistem.

2.11.1 *Arduino*

Arduino adalah kit elektronik atau papan rangkaian elektronik open source yang didalamnya terdapat komponen utama, yaitu sebuah chip mikrokontroller dengan jenis AVR dari perusahaan atmel. Mikrokontroller itu sendiri adalah chip atau IC (*Integrated Circuit*) yang bisa diprogram menggunakan komputer. Tujuan menanamkan program pada mikrokontroller adalah agar rangkaian elektronik dapat membaca input, memproses input tersebut dan kemudian menghasilkan output sesuai yang diinginkan. Jadi mikrokontroler bertugas sebagai otak yang mengendalikan input, proses dan output sebuah rangkaian elektronik [30]. Secara umum, *Arduino* terdiri dari dua bagian, yaitu :

1. *Hardware* berupa papan input/output (I/O) yang open source.
2. *Software Arduino* yang juga open source, meliputi *software Arduino* IDE untuk menulis program dan driver untuk koneksi dengan komputer.

2.11.2 *Arduino Nano*

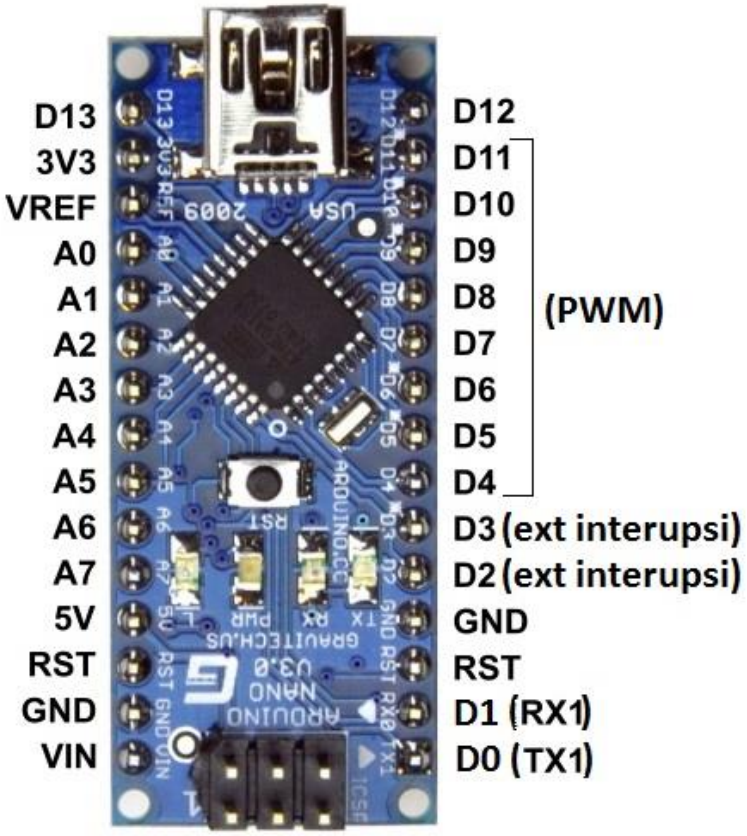
Arduino merupakan sebuah platform dari physical computing yang bersifat open source. *Arduino* tidak hanya sekedar sebuah alat pengembang, tetapi merupakan kombinasi dari *hardware*, bahasa pemrogaman dan *Integrated Development Environment (IDE)* yang canggih IDE adalah sebuah *software* yang berperan untuk menulis program, meng-compile menjadi kode biner dan mengupload ke dalam memory microcontroler *Arduino* Nano adalah salah satu varian dari produk board mikrokontroller keluaran *Arduino*.

Arduino Nano adalah board *Arduino* terkecil, menggunakan mikrokontroller Atmega 328 untuk *Arduino* Nano 3.x dan Atmega168 untuk *Arduino* Nano 2.x. Varian ini mempunyai rangkaian yang sama dengan jenis *Arduino* Duemilanove, tetapi dengan ukuran dan desain PCB yang berbeda [30]. *Arduino* Nano tidak dilengkapi dengan soket catudaya, tetapi terdapat pin untuk catu daya luar atau dapat menggunakan catu daya dari mini USB port. *Arduino* Nano didesain dan diproduksi oleh Gravitech. Berikut adalah Tabel 2.8 Tabel Spesifikasi *Arduino Nano* :

Tabel 2.8 Tabel Spesifikasi *Arduino Nano*

Spesifikasi	Keterangan
Mikrokontroler	Atmel ATmega168 atau ATmega328
Tegangan Operasi	5V
Input Voltage (disarankan)	7-12V
Input Voltage (limit)	6-20V
Pin Digital I/O	14 (6 pin digunakan sebagai output PWM)
Pins Input Analog	8
Arus DC per pin I/O	40 mA
Flash Memory	16KB (ATmega168) atau 32KB (ATmega328) 2KB digunakan oleh Bootloader
SRAM	1 KB (ATmega168) atau 2 KB (ATmega328)
EEPROM	512 byte (ATmega168) atau 1KB (ATmega328)
Clock Speed	16 MHz
Ukuran	1.85cm x 4.3cm

Konfigurasi pin *Arduino Nano. Arduino Nano* memiliki 30 Pin. Berikut penjelasan konfigurasi pin *Arduino Nano* pada Gambar 2.7 Modul *Arduino Nano*:



Gambar 2.7 Modul *Arduino Nano*

Konfigurasi umum pin *Arduino Nano*.

1. VCC merupakan pin yang berfungsi sebagai pin masukan catu daya digital.
2. GND merupakan pin ground untuk catu daya digital.
3. AREF merupakan Referensi tegangan untuk input analog. Digunakan dengan fungsi analogReference().
4. RESET merupakan Jalur LOW ini digunakan untuk me-reset (menghidupkan ulang) mikrokontroler. Biasanya digunakan untuk menambahkan tombol reset pada shield yang menghalangi papan utama Arduino
5. Serial RX (0) merupakan pin yang berfungsi sebagai penerima TTL data serial. 6. Serial TX (1) merupakan pin yang berfungsi sebagai pengirim TT data serial.

6. *External Interrupt* (Interupsi Eksternal) merupakan pin yang dapat dikonfigurasi untuk memicu sebuah interupsi pada nilai yang rendah, meningkat atau menurun, atau perubahan nilai.Output PWM 8-Bitmerupakan pin yang berfungsi untuk analogWrite( ).
7. SPI merupakan pin yang berfungsi sebagai pendukung komunikasi.
8. LED merupakan pin yang berfungsi sebagai pin yang diset bernilai HIGH, maka LED akan menyala, ketika pin diset bernilai LOW maka LED padam. LED Tersedia secara *built-in* pada papan *Arduino Nano*.
9. Input Analog (A0-A7) merupakan pin yang berfungsi sebagi pin yang dapat diukur/diatur dari mulai Ground sampai dengan 5 Volt, juga memungkinkan untuk mengubah titik jangkauan tertinggi atau terendah mereka menggunakan fungsi analogReference().

Berikut adalah Tabel 2.9 Tabel Konfigurasi *Arduino*:

Tabel 2.9 Tabel Konfigurasi *Arduino*

Nomor Pin *Arduino Nano*	Nama Pin *Arduino*
1	Digital Pin 0 (TX)
2	Digital Pin 0 (RX)
3 & 28	Reset
4 & 29	GND
5	Digital Pin 2
6	Digital Pin 3 (PWM)
7	Digital Pin 4
8	Digital Pin 5 (PWM)
9	Digital Pin 6 (PWM)
10	Digital Pin 7
11	Digital Pin 8
12	Digital Pin 9 (PWM)
13	Digital Pin 10 (PWM-SS)
14	Digital Pin 11 (PWM-MOSI)
15	Digital Pin 12 (MISO)
16	Digital Pin 13 (SCK)
18	AREF
19	Analog Input 0
20	Analog Input 1
21	Analog Input 2
22	Analog Input 3
23	Analog Input 4
24	Analog Input 5
25	Analog Input 6
26	Analog Input 7

Nomor Pin *Arduino Nano*	Nama Pin *Arduino*
27	VCC
30	Vin

2.11.3 *Raspberry pi 3*

Raspberry pi 3 adalah generasi ketiga dari *Raspberry pi*, menggantikan *Raspberry pi 2 Mode B* pada Februari 2016. *Raspberry pi 3* memiliki bentuk yang identik dengan *Raspberry pi 2* sebelumnya (dan Pi 1 Model B +) dan memiliki kompatibilitas lengkap dengan *Raspberry pi* 1 dan 2. Pada perangkat terbarunya ini *Raspberry* menambahkan fitur *built-in wireless* dan *proseso*r yang lebih bertenaga yang belum pernah dimiliki pada versi sebelumnya [31]. Bentuk dari *Raspberry pi 3* dapat diligat pada Gambar 2.8 *Raspberry pi 3*:

Gambar 2.8 *Raspberry pi 3* [31]

Berikut adalah Tabel 2.10 Tabel Spesifikasi *Raspberry pi* 3

Tabel 2.10 Tabel Spesifikasi *Raspberry pi* 3

Spesifikasi	Keterangan
Soc	BCM2837
Procesor	1.2GHz 64-bit quad-core ARMv8 CPU
Memory /RAM	1 GB SDRAM 400MHz
GPU	*VideoCore IV 3D graphics core*
Wireless Adapter/LAN	*802.1 In Wireless LAN*
Bluetooth	*Bluetooth 4.1 (built in), Bluetooth Low Energy (BLE)*
GPIO	40 Pin
Port USB	4 USB Ports
Card Stroge	*Micro SD card slot (now push-pull rather than push-push)*
Jaringan	*Ethernet Port*
ExternalAudio and Video	*Full HDMI port, Camera interface (CSI), Display interface (DSI), Combined 3.5mm audio jack and composite video*
Sistem Operasi	*Debian GNU/Linux,Fedora, Arch Linux ARM, RISC OS*

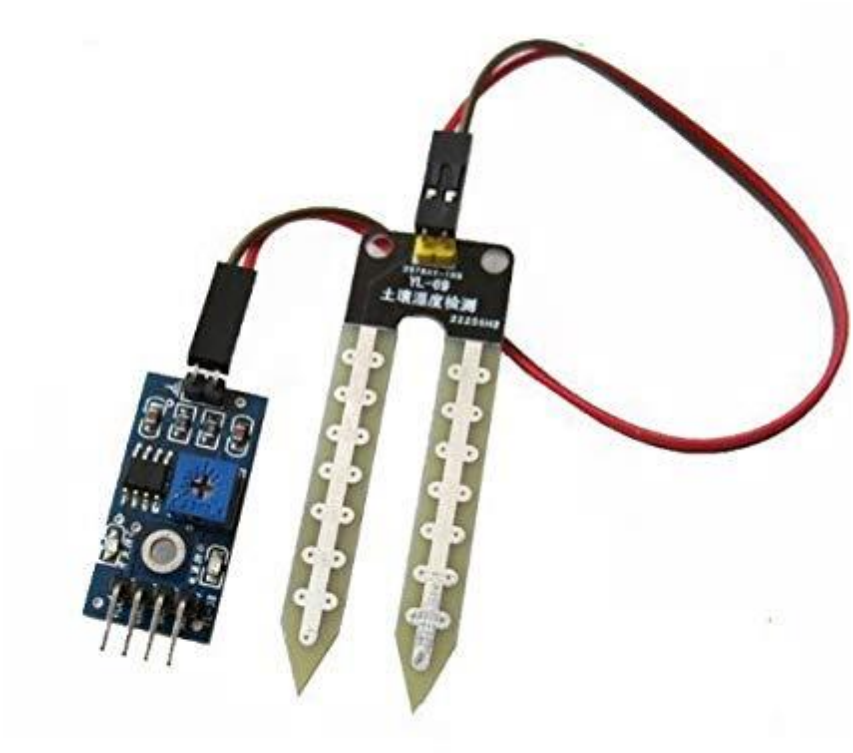
2.12 Sensor

Sensor adalah suatu peralatan yang berfungsi untuk mendeteksi gejala-gejala atau sinyal-sinyal yang berasal dari perubahan suatu energi seperti energi listrik, energi fisika, energi kimia, energi biologi, energi mekanik dan sebagainya [32].

2.12.1 Sensor *Soil Mouisture*

Soil Moisture Sensor adalah sensor yang dapat mendeteksi kelembaban tanah disekitarnya. Sensor ini terdiri dari dua probe untuk melewatkan arus listrik dalam tanah, kemudian membaca resistansinya untuk mendapatkan nilai tingkat kelembaban. Semakin banyak air membuat tanah lebih mudah menghantarkan listrik (resistansi kecil), sedangkan tanah yang kering sangat sulit menghantarkan

listrik (resistansi besar). Bentuk dari sensor soil moisture dapat dilihat pada Gambar 2.9 Sensor *Soil Moisture* :



Gambar 2.9 Sensor *Soil Moisture*

Sensor ini ideal digunakan dalam monitoring dan pendeteksi tanah longsor karena digunakan untuk memantau kondisi tingkat kelembaban pada tanah yang memilki potensi longsor, dengan menghitung tingkat kelembaban makan akan diketahui tingkat kerawanan longsor.berikut adalah Tabel 2.11 Spesifikasi *Soil Moisture* :

Tabel 2.11 Spesifikasi *Soil Moisture*

Parameter	**Spesifikasi**
Power Supply	3.3v or 5v
Output voltage signal	0~4.2v
Current	35mA
Pin definition	1. Analog output 2.GND 3.Power
Size	60x20x5

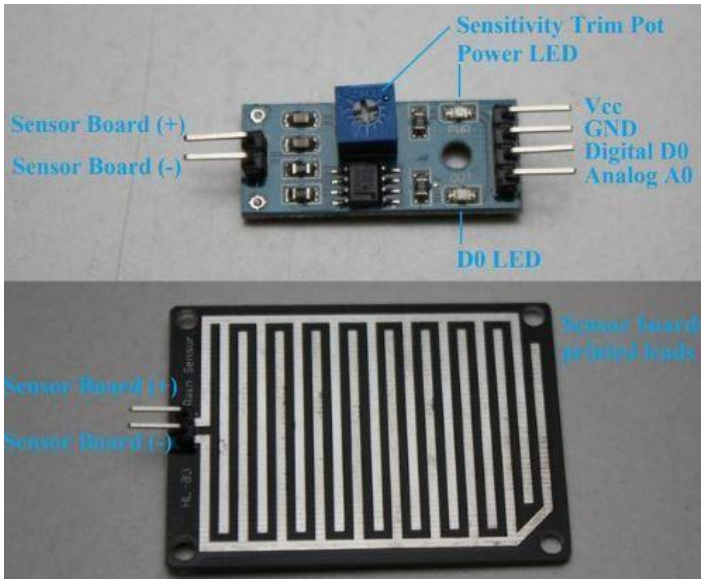
2.12.2 Sensor Hujan

Sensor hujan adalah jenis sensor yang berfungsi untuk mendeteksi terjadinya hujan atau tidak, yang dapat difungsikan dalam segala macam aplikasi dalam kehidupan sehari – hari. Dipasaran sensor ini dijual dalam bentuk modul sehingga hanya perlu menyediakan kabel jumper untuk dihubungkan ke mikrokontroler atau *Arduino*.

Prinsip kerja dari module sensor ini yaitu pada saat ada air hujan turun dan mengenai panel sensor maka akan terjadi proses elektrolisasi oleh air hujan. Dan karena air hujan termasuk dalam golongan cairan elektrolit yang dimana cairan tersebut akan menghantarkan arus listrik.

Pada sensor hujan ini terdapat ic komparator yang dimana output dari sensor ini dapat berupa logika *high* dan *low* (*on* atau *off*). Serta pada modul sensor ini terdapat *output* yang berupa tegangan pula. Sehingga dapat dikoneksikan ke pin khusus *Arduino* yaitu *Analog Digital Converter*.

Dengan singkat kata, sensor ini dapat digunakan untuk memantau kondisi ada tidaknya hujan di lingkungan luar yang dimana output dari sensor ini dapat berupa sinyal analog maupun sinyal digital. Berikut adalah Gambar 2.10 Sensor Hujan



Gambar 2.10 Sensor Hujan

Berikut adalah Tabel 2.12 Spesifikasi Sensor Hujan:

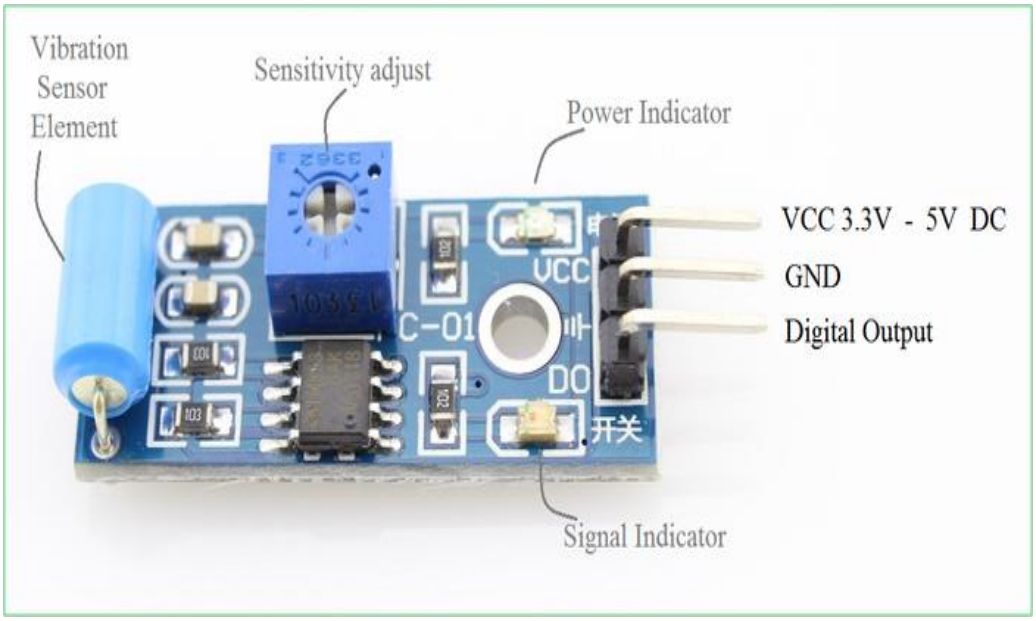
Tabel 2.12 Spesifikasi Sensor Hujan

Parameter	**Spesifikasi**
Power supply voltage	3.3-5V
Dimensions	2.17 in x 1.57 in x 0.31 in ().
Weight	0.28 oz (8 g).
LM393 comparator	*Comparator output signal waveform is good,*
driving ability	15mA
Voltage Rating	5V
Shield PCB Size	30mm x 16mm
Sensor Board Size	54mm x 40mm

2.12.3 Sensor Getar (*Vibration Sensor*)

Sensor getar adalah sensor yang memanfaatkan getaran fisik yang diterima oleh komponen sensor getar yang kemudian diubah menjadi sinyal listrik dan diolah lebih lanjut sesuai kebutuhan, komponen pendeteksi getar yang bereaksi terhadap getaran dari berbagai sudut. Pada kondisi statis/tanpa getaran, sensor ini berfungsi seperti saklar yang berada pada kondisi menutup (*normally closed*) dan bersifat konduktif, sebaliknya pada kondisi terpapar getaran, saklar akan membuka/menutup dengan kecepatan pengalihan (*switching frequency*) proporsional dengan kekerapan getaran. Tingkat sensitifitas pendeteksian dapat diatur dengan memutar potensiometer (*variable resistor*) yang terpasang di sensor ini.

Prinsip Kerja dari sensor ini adalah adalah dengan menggunakan 1 buah pelampung logam yang akan bergetar ditabung yang berisi 2 elektroda ketika modul sensor menerima getaran / shock. Terdapat 2 *output* yaitu digital *output* (0 dan 1) dan analog *output* (tegangan). Gambar dapat dilihat pada Gambar 2.11 Sensor Getaran



Gambar 2.11 Sensor Getaran

Berikut adalah Tabel 2.13 Spesifikasi Sensor Getar :

Tabel 2.13 Spesifikasi Sensor Getar

Parameter	**Spesifikasi**
Chipset	LM393
Type	SW-420
Size	3.2 x 1.4 mm
Voltage	3.3 – 5 V DC

2.12.4 Sensor Kelembaban Udara *DHT11*

Sensor *DHT11* merupakan sensor dengan kalibrasi sinyal digital yang mampu memberikan informasi suhu dan kelembaban. Sensor ini tergolong komponen yang memiliki tingkat stabilitas yang sangat baik, apalagi digandeng dengan kemampuan mikrokontroler *ATmega8*. Produk dengan kualitas terbaik, respon pembacaan yang cepat, dan kemampan anti-interference [33]. *DHT11* adalah sensor digital yang dapat mengukur suhu dan kelembaban udara disekitarnya. Sensor ini sangat mudah digunakan dengan *Arduino*. Memiliki tingkat stabilitas yang sangat baik serta fitur kalibrasi yang sangat akurat.

DHT11 temasuk sensor yang memiliki kualitas baik, dinilai dari respon, pembacaan data yang cepat, dan kemampuan anti-interference. Ukurannya yang kecil, dan dengan transmisi sinyal hingga 20 meter, dengan spesifikasi : *Supply Voltage*: +5 V, *Temperature range* : 0-50 °C *error of* ± 2 °C, *Humidity* : 20-90% RH ± 5% RH *error*, dengan spesifikasi *digital interfacing sytem.* Membuat produk ini cocok digunakan untuk banyak aplikasi-aplikasi pengukuran suhu dan kelembaban [33]**.** Berikut adalah Gambar 2.12 Sensor Kelembaban Udara *DHT11*

Gambar 2.12 Sensor Kelembaban Udara *DHT11*

Berikut adalah Tabel 2.14 Spesifikasi Sensor Kelembaban Udara *DHT11*:

Tabel 2.14 Spesifikasi Sensor Kelembaban Udara *DHT11*

Nomor Pin *Arduino Nano*	**Nama Pin *Arduino***
Model	*DHT11*
Power supply	*3-5.5V DC*
Output signal	*digital signal via single-bus*
Measuring range	*humidity 20-90% RH ± 5% RH error* *temperature 0-50 °C error of ± 2 °C*
Accuracy	*humidity +-4%RH (Max +-5%RH);* *temperature +-2.0 Celsius*

Nomor Pin *Arduino Nano*	Nama Pin *Arduino*
Resolution orSensitivity	*humidity 1%RH; temperature 0.1Celsius*
Repeatability	*humidity +-1%RH; temperature + -1 Celsius*
Humidity hysteresis	*+-1%RH*
Long-term Stability	*+-0.5%RH/year*
Sensing period	*Average: 2s*
Interchangeability	*fully interchangeable*
Dimensions size	*12*15.5*5.5mm*

2.12.5 Modul Step Down LM 2596

Modul LM2596 adalah modul yang digunakan untuk menurunkan tegangan DC maksimal hingga 3A dengan range DC 3.2V-46V dengan selisih minimum *input* - *output* 1.5V DC. Keunggulan modul *step down* LM2596 adalah besar tegangan *output* tidak berubah (stabil) walaupun tegangan input naik turun, Output bisa di atur dengan memutar potensiometer

Modul *stepdown* lm2596 memiliki IC LM2596 sebagai komponen utamanya. IC LM2596 adalah sirkuit terpadu / *integrated circuit* yang berfungsi sebagai *Step-Down DC converter* dengan *current rating* 3A. Terdapat beberapa varian dari IC seri ini yang dapat dikelompokkan dalam dua kelompok yaitu versi *adjustable* yang tegangan keluarannya dapat diatur, dan versi *fixed voltage output* yang tegangan keluarannya sudah tetap / *fixed*. Berikut adalah Gambar 2.13 *Stepdown* LM259 :

Gambar 2.13 *Stepdown* LM259

Berikut adalah Tabel 2.15 Spesifikasi *Stepdown LM2596*:

Tabel 2.15 Spesifikasi *Stepdown LM2596*

Parameter	**Spesifikasi**
Model/name	LM2596S DC-DC Step-Down module
input	3.2-46V DC
output	1.25-35V DC
Max	3A,
Efisiensi step down	92%
Operating Temperature	45 - 85 C
Output ripple	30mV
Switching frequency	65KHz
Dimension	43 x 21 x 14 mm

2.13 LCD 16x2 dengan LCD I2C (Inter integrated Circuit)

Display elektronik adalah salah satu komponen elektronika yang berfungsi sebagai tampilan suatu data, baik karakter, huruf ataupun grafik. *LCD (Liquid Cristal Display)* adalah salah satu jenis *display* elektronik yang dibuat dengan teknologi *CMOS logic* yang bekerja dengan tidak menghasilkan cahaya tetapi memantulkan cahaya yang ada di sekelilingnya terhadap *front-lit* atau mentransmisikan cahaya dari *back-lit*. *LCD (Liquid Cristal Display)* berfungsi sebagai penampil data baik dalam bentuk karakter, huruf, angka ataupun grafik, bentuk dapat dilihat pada Gambar 2.14 LCD 16x2

Gambar 2.14 LCD 16x2

2.13.1 Modul I2C *(Inter integrated circuit)*

Jumlah *IO Port* pada *Arduino* kadang tidak cukup untuk semua sensor, *card reader, relay* dan modul lainnya sehingga tidak cukup untuk layar LCD yang memerlukan 7 *IO Port* untuk pengendalian (4 pin data pada moda 4-bit / 8-pin data pada moda 8-bit + 1 pin RS + *optional* 1 pin untuk R/W + 1 pin *Enable*, di luar pin untuk mengendalikan lampu latar).Dengan pemakaian *Serial Interface* IIC/I2C ini hanya diperlukan 2 *port* saja untuk mengendalikan LCD sehingga menghemat pemakaian *port* pada *Arduino*. Berikut bentuk dari modul pada Gambar 2.15 Modul *Inter Integrated Component* (I2C) :

Gambar 2.15 Modul *Inter Integrated Component* (I2C)

2.14 Kapasitor

Kapasitor (*Capacitor*) atau disebut juga dengan Kondensator (*Condensator*) adalah komponen elektronika pasif yang dapat menyimpan muatan listrik dalam waktu sementara dengan satuan kapasitansinya adalah Farad. Satuan Kapasitor tersebut diambil dari nama penemunya yaitu Michael Faraday (1791 ~ 1867) yang berasal dari Inggris. Namun Farad adalah satuan yang sangat besar, oleh karena itu pada umumnya Kapasitor yang digunakan dalam peralatan Elektronika adalah satuan Farad yang dikecilkan menjadi *pikoFarad, NanoFarad* dan *MicroFarad*. Kapasitor merupakan Komponen Elektronika yang terdiri dari 2 pelat konduktor yang pada umumnya adalah terbuat dari logam dan sebuah Isolator diantaranya

sebagai pemisah. Dalam Rangkaian Elektronika, Kapasitor disingkat dengan huruf "C" [34]**.** Bentuk dari kapasitor dapat dilihat pada Gambar 2.16 Kapasitor :

Gambar 2.16 Kapasitor

Dibawah ini adalah beberapa fungsi daripada Kapasitor dalam Rangkaian Elektronika :

1. Sebagai Penyimpan arus atau tegangan listrik
2. Sebagai Konduktor yang dapat melewatkan arus AC (*Alternating Current*)
3. Sebagai Isolator yang menghambat arus DC (*Direct Current*)
4. Sebagai Filter dalam Rangkaian *Power Supply* (Catu Daya)
5. Sebagai Kopling
6. Sebagai Pembangkit Frekuensi dalam Rangkaian Osilator
7. Sebagai Penggeser Fasa
8. Sebagai Pemilih Gelombang Frekuensi (Kapasitor Variabel yang digabungkan dengan Spul Antena dan Osilator) [34]**.**

2.15 Pengujian Sistem

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan

pengkodean. Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah Pengujian perangkat lunak memiliki urutan-urutan mengenai beberapa hal yang perlu dilakukan. Berikut adalah kategori pengujian perangkat lunak yang disusun secara kronologis [35]**:**

1. *Unit Testing*: Pengujian dilakukan pada setiap modul atau blok kode selama pengembangan. Pengujian ini biasanya dilakukan oleh *developer* yang menulis kode.
2. *Integration Testing*: Pengujian yang dilakukan Sebelum, selama, dan setelah integrasi modul baru ke dalam paket perangkat lunak utama. Pengujian ini melibatkan pengujian setiap modul kode dari masing-masing individu. Satu perangkat lunak dapat berisi beberapa modul yang sering dibuat oleh beberapa *developer* yang berbeda.
3. *System Testing*: Pengujian yang dilakukan oleh agen pengujian profesional pada produk perangkat lunak yang telah selesai sebelum perangkat lunak tersebut diperkenalkan secara umum.
4. *Acceptance Testing*: Pengujian beta dari produk yang dilakukan oleh pengguna akhir yang sebenarnya.

Terdapat beberapa karakteristik dari pengujian sistem perangkat [35], berikut adalah penjelasan dari setiap karakteristik :

1. Pengujian dimulai pada level modul dan bekerja keluar kearah integrasi pada sistem berbasiskan komputer.
2. Teknik pengujian yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3. Pengujian diadakan oleh *software developer* dan untuk proyek yang besar oleh *group testing* yang *independent.*
4. *Testing* dan *Debugging* adalah aktivitas yang berbeda tetapi *debugging* harus diakomodasikan pada setiap strategi *testing.*

Metode pengujian perangkat lunak ada terbagi menjadi 3 jenis, yaitu: [35]

1. *White Box/Glass Box* - pengujian operasi
2. *Black Box* - untuk menguji sistem

3. *Use case* - untuk membuat input dalam perancangan *black box* dan pengujian *statebased.*

2.15.1 *Blackbox Testing*

Black Box testing juga disebut pengujian tingkah laku, memusat pada kebutuhan fungsional perangkat lunak. Teknik pengujian *black box* memungkinkan memperoleh serangkaian kondisi masukan yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Beberapa jenis kesalahan yang dapat diidentifikasi adalah fungsi tidak benar atau hilang, kesalahan antar muka, kesalahan pada struktur data (pengaksesan basis data), kesalahan performasi, kesalahan inisialisasi dan akhir program [36]**.** Pengujian pada Black Box berusaha menemukan kesalahan seperti:

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database* eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi.

Ada beberapa ciri khusus dalam teknik pengujian *blackbox*, yaitu : [35]

1. *Black box testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*.
2. *Black box testing* bukan teknik alternatif daripada *white box testing*. Lebih dari pada itu, ia merupakan pendekatan pelengkap dalam mencakup error dengan kelas yang berbeda dari metode *white box testing*.
3. *Black box testing* melakukan pengujian tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites. juga disebut sebagai *behavioral testing, specification-based testing, input/output testing* atau *functional testing*

Black box Testing memungkinkan untuk melakukan pengujian kebenaran *output*. Selain itu, karena karakteristik spesial dari masing-masing strategi *testing* dan kelas-kelas pengujian yang khusus untuk *White box Testing, Black box*

Testing tidak dapat secara otomatis menggantikan *White box Testing* [37].berikut adalah tipe-tipe pengujian dalam black box testing :

1. *Equivalence classes for output correcness tests*

   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. Equivalence class partitioning* adalah sebuah metode *black box* terarah yang meningkatkan efisisensi dari pengujian dan meningkatkan *coverage* dari *error* yang potensial. Sebuah *equivalence class* adalah sebuah kumpulan dari nilai variabel *input* yang memproduksi *output* yang sama [37]**.**

2. *Documentation tests*

   Documentation *testing* harus dipertimbangkan sebagai *code testing* atau pemeriksaan dokumen rancangan. Manual penggunaan atau manual programer yang salah dapat menimbulkan kesalahan selama operasi dan pemeliharaan program yang dapat mendatangkan kerusakan sebanding dengan keparahan akibat *software bugs*. Komponen utama dari dokumentasi yang disediakan oleh pengembang adalah:

   1. Deskripsi fungsional dari sistem *software*.
   2. Manual instalasi.
   3. *User manual*.
   4. *Programmer manual.*

   Rencana pengujian dokumen harus terdiri dari 3 komponen berikut:

   1. *Document completeness check*
   2. *Document correctness tests.*
   3. *Document style dan editing inspection* [37]**.**

3. *Avaibility Test*

   Availabilitas 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 [37]**.**

4. *Reliability tests*

   Persyaratan reabilitas sistem *software* 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 reliabilitas memiliki efek selama *regular full- capacity* operasi sistem. Harus diperhatikan bahwa penambahan faktor *software reliability test* juga berkaitan dengan perangkat, sistem operasi, dan efek dari sistem komunikasi data [37]**.** Persyaratan reabilitas sistem *software* 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 reliabilitas memiliki efek selama *regular full- capacity* operasi sistem. Harus diperhatikan bahwa penambahan faktor *software reliability test* juga berkaitan dengan perangkat, sistem operasi, dan efek dari sistem komunikasi data [37]**.**

5. *Stress tests*

   Kelas stress tests terdiri dari 2 tipe pengujian, yaitu load test dan durability test. Suatu hal yang mungkin untuk melakukan pengujian-pengujian tersebut setelah penyelesaian sistem *software*. Durability test dapat dilakukan hanya setelah firmware atau sistem informasi *software*

diinstal dan siap untuk diuji [37]**.**

a. Load Test

Load tests berkaitan dengan functional performance system dibawah beban maksimal operasional, yaitu maksimal transaksi per menit, hits per menit ke tempat internet dan sebagainya. Load tests, 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.

b. Durability test

Durability test dilakukan pada kondisi operasi fisik yang ekstrem seperti temperatur yang tinggi, kelembaban, mengendara dengan kecepatan tinggi pada jalan di pedesaan, sebagai detail persyaratan spesifikasi durabilitas. Jadi, durability tests ini dibutuhkan untuk real-time firmware yang diintegrasikan ke dalam sistem seperti sistem senjata, kendaraan transport jarak jauh, dan keperluan meteorologi. Isu ketahanan pada *firmware* terdiri dari respon *firmware* terhadap efek cuaca seperti temperatur panas atau dingin yang ekstrem, debu, kegagalan operasi ekstrem karena kegagalan listrik secara tiba-tiba, loncatan arus listrik, putusnya komunikasi tiba-tiba dan sebagainya. Durability tests *software* sistem operasi berfokus pada kegagalan operasi akibat kegagalan listrik, loncatan arus listrik dan putusnya komunikasi secara tiba-tiba.

6. *Software system security tests*

Komponen keamanan *software* pada sistem sofrware ditujukan untuk mencegah unauthorized access terhadap sistem atau bagiannya, mendeteksi unauthorized access dan aktivitas yang dilakukan melalui

penetrasi dan memperbaiki kerusakan yang disebabkan oleh unauthorized penetration [37].

Sistem keamanan utama yang dilakukan oleh uji ini adalah:

1. Access control.
2. Backup *database* dan file *software* dan perbaikan kegagalan sistem.
3. Logging of transaction, penggunaan sistem, access trials, dsb.

7. *Training usability tests*

   Ketika sejumlah besar pengguna terlibat dalam sistem operasi, training usability requirement ditambahkan dalam agenda pengujian. Lingkup dari training usability tests 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 sistem, tetapi yang lebih penting lagi adalah berdasarkan karakteristik pekerja. Hasil dari pengujian ini harus menginspirasi rencana dari kursus pelatihan dan follow-up serta memperbaiki sistem operasi *software* [37].

8. *Operational usabilty tests*

   Fokus dari pengujian ini adalah produktifitas operator, yang aspeknya tehadap sistem yang mempengaruhi performace dicapai oleh operator sistem. Operational usability test dapat dijalankan secara manual [37].

9. *Revision factor testing classes*

   Revisi yang mudah dari *software* merupakan faktor dasar yang menentukan keberhasilan paket suatu *software*, pelayanan jangka panjang, dan keberhasilan penjualan ke sejumlah besar populasi pengguna [37].

10. *Reusability tests*

    Reusabilitas menentukan bagian mana dari suatu program (modul, integrasi, dbs) yang akan dikembangkan untuk digunakan kembali pada

project pengembangan *software* lainnya, baik yang telah direncanakan maupun yang belum. Bagian ini harus dikembangkan, disusun, dan didokumentasikan menurut prosedur perpustakaan *software* yang digunakan ulang [37].

11. *Equipment interoperability tests*

    Equipment interoperability berkaitan dengan perlengkapan firmware dalam menghadapi unit perlengkapan lain dan atau paket *software*, dimana persyaratan mencantumkan *specified interfaces*, termasuk dengan *interfacing* standard [37]. Pengujian yang relevan harus menguji implementasi dari interoperability requirements dalam sistem.

12. *Software interoperability tests*

    Software interoperability berkaitan dengan kemampuan *software* dalam memenuhi perlengkapan dan paket *software* lainnya agar memungkinkan untuk mengoperasikannya bersama dalam satu sistem komputer kompleks [37].

Black box testing memiliki beberapa kerugian dan kekurangan. 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 dengan *black box tests*.
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 kerugian 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* tests 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 tests* diharapkan dapat meningkatkan *line coverage*, tidak ada cara yang mudah untuk menspesifikasikan *parameter-parameter* pengujian yang dibutuhkan untuk meningkatkan *coverage*. Akibatnya, *black bos tests* 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. [37]