BAB 2
TINJAUAN PUSTAKA

2.1.Tempat Penelitian

Laksana motor merupakan sebuah bengkel kendaraan bermotor khususnya sepeda motor atau kendaraan roda dua. aksana motor terletak di kota Subang pada Jl. Ion Martasasita No. 3 bertepatan seberang balai desa Rancasari di kecamatan Pamanukan, seperti bengkel pada umumnya laksana motor menjual orderdil dan memiliki mekanik untuk melakukan perawatan kendaraan roda dua. Namun, tak hanya fokus pada penjualan dan perawatan saja, bengkel ini memiliki alat-alat teknik sebagai pendukung perawatan sepeda motor seperti mesin *bubut,* mesin *korter*, mesin *las* dan alat pendukung lainnya.

Bengkel laksana motor telah berdiri lebih dari dua puluh tahun, lebih tepatnya didirikan pada tahun 1995 oleh Bapak Asep Sugiarto selaku pemilik bengkel yang artinya telah beroperasi selama dua puluh empat tahun. Pada awal beroperasi benkel ini hanya berfokus pada penjualan onderdil kendaraaan roda dua saja secara grosir maupun eceran. Namun, setelah beberapa tahun beroperasi, pemilik memperlebar usahanya dengan menambahkan jasa perawatan kendaraan roda doa dan jasa teknik pendukung lainnya.

2.1.1. Visi dan Misi

Sebgai upaya untuk terus meningkatkan kualitas serta pelayanan, laksana motor pamanukan memiliki visi serta misi sebagai berikut :

Visi

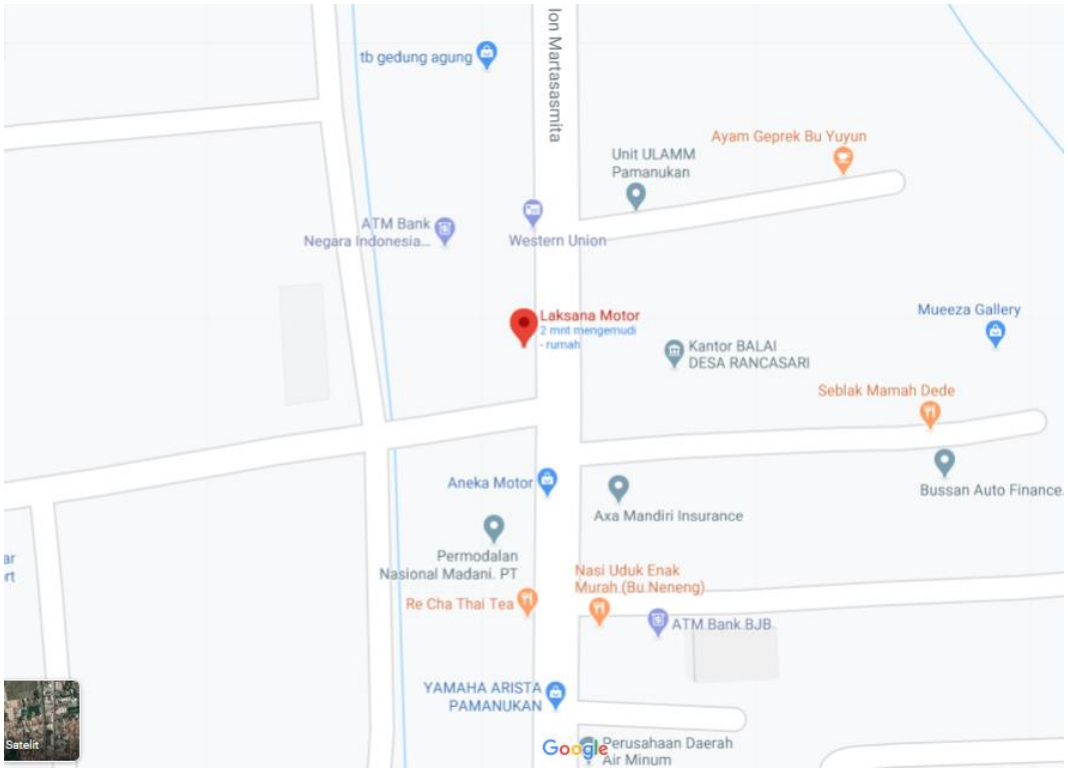
Menjadi pusat perawatan, reparasi serta modifikasi sepeda motor yang menyediakan onderdil dan jasa perawatan, reparasi dan modifikasi dengan mengutamakan pada tingkat kepuasan pelanggan yang didukung dengan peralatan yang sesuai dan tenaga ahli yang kompeten sehingga menghasilkan pelayanan yang optimal dan terepercaya.

Misi

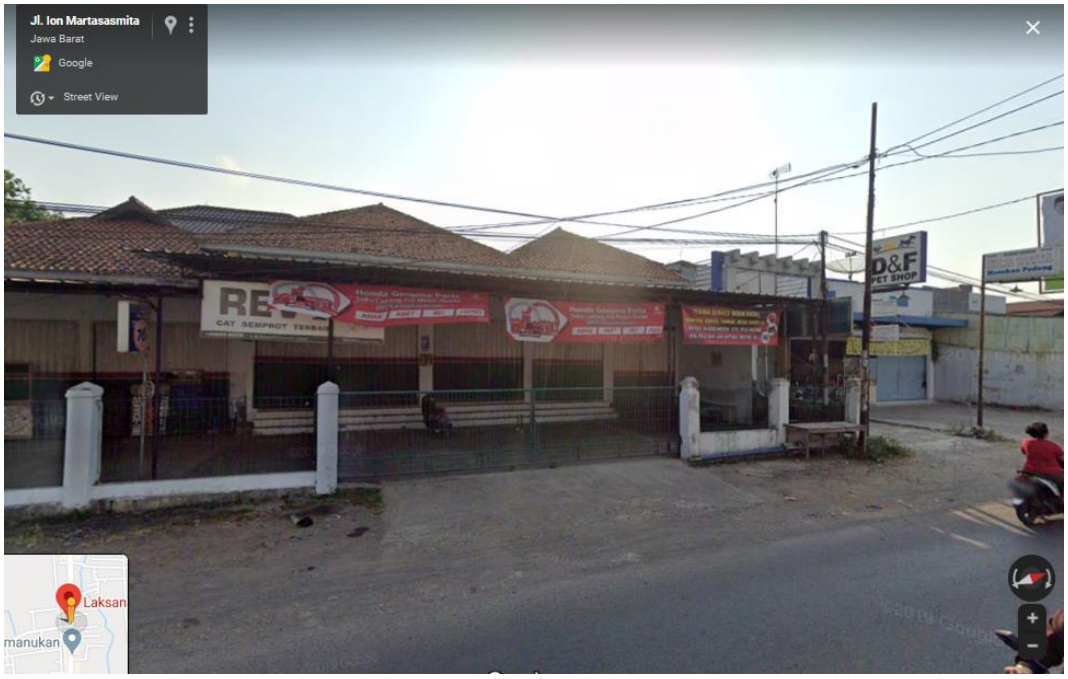
1. Memberikan solusi terbaik untuk perawatan, reparasi serta modifikasi.
2. Memberikan pelayanan terbaik dengan standar kerja yang tersusun rapi sehingga tercapai kepuasan pelanggan.
3. Mengikuti perkembangan ilmu dan terknologi yang terdapat pada sepeda motor agar dapat memcahkan setiap permasalahan yang terjadi pada sepeda motor serta mengimplementasikan dengan cara yang benar.
4. Meningkatkan motivasi dan semangat kerja karyawan agar mendapatkan hasil kerja yang optimal dan sesuai yang diharapkan.

2.1.2. Letak Geografis

Laksana Motor terletak di Jl. Ion Martasasmita No.3 Pamanukan. Titik lokasi dapat dilihat pada gambar 2.1.



Gambar 2.1. Letak Tempat Penelitian Maps



Gambar 2.2. Lokasi Tempat Penelitian

2.2.Landasan Teori

Pada landasan teori, berisi tentan teori-teori maupun konsep-konsep yang berkaitan dengan penelitian ini, adapun beberapa teori yang digunakan sebagai pendukung sebagai berikut.

2.2.1. Sistem Keamanan

Sistem keamanana merupakan sistem yang digunakan untuk memberikan rasa aman tentram serta bebas dari bahaya ataupun ancaman, shingga individu tidak merasa takut, resah atau gelisah terhadap barang berharga yang ditinggalkan. Sistem keamanan dapat mengetehui kemungkinan adanya tanda-tanda bahaya akan terjadinya tindak pencurian terhadap barang berharga[6].

2.2.2. Kunci Kontak

Berdasarkan buku sistem pengapian, kunci kotak pada sepeda motor dibagi menjadi dua yaitu AC dan DC, berikut penjelasannnya.

1. Kunci Kontak DC

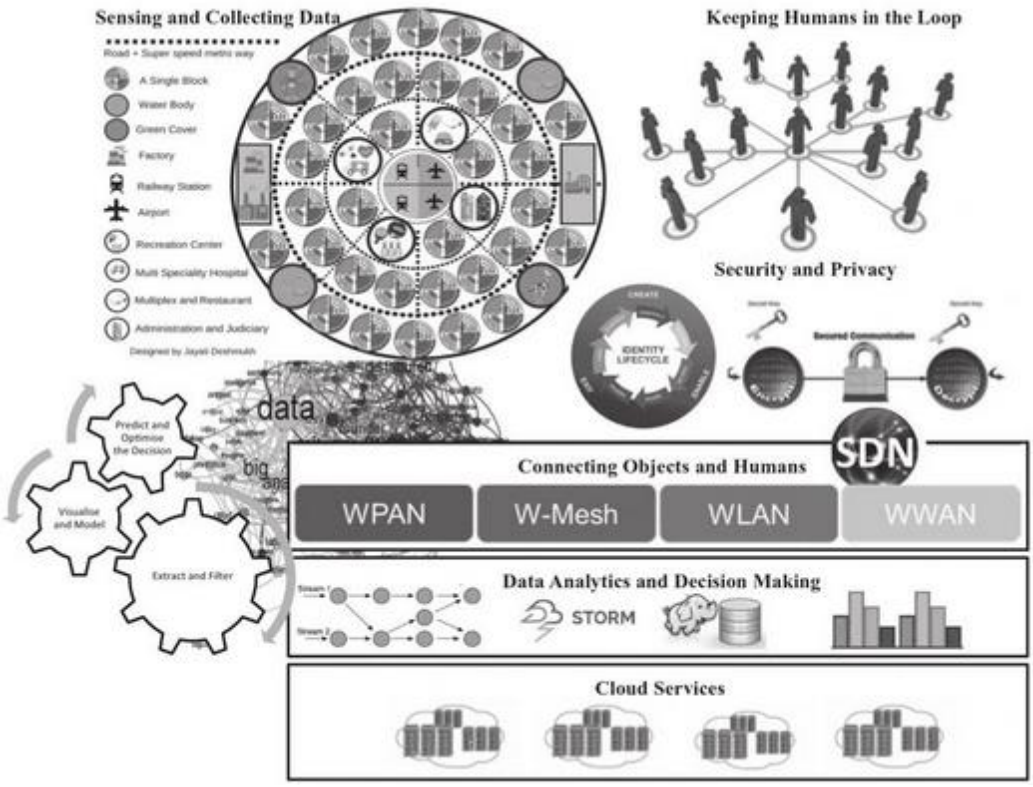
Pada posisi ON, kunci kontak menghubungkan tegangan (+) baterai ke seluruh sistem kelistrikan (termasuk sistem pengapian) untuk mengoperasikan seluruh sistem kelistrikan yang ada. Pada posisi OFF dan LOCK, kunci kontak memutuskan hubungan kelistrikan dari sumber tegangan (terminal (+) baterai) yang dibutuhkan oleh seluruh sistem kelistrikan, sehingga seluruh sistem kelistrikan tidak dapat dioperasikan[7].

2. Kunci Kontak AC

Pada posisi *OFF* dan *LOCK*, kunci kontak membelokkan tegangan dari sumber tegangan (*alternator*) yang dibutuhkan oleh sistem pengapian ke massa melalui terminal IG dan E kunci kontak, sehingga sistem pengapian tidak dapat bekerja. Di sisi lain pada posisi *OFF* dan *LOCK* kunci kontak juga memutuskan hubungan tegangan (+) baterai (terminal BAT dan BAT 1) sehingga seluruh sistem kelistrikan tidak dapat dioperasikan. Pada posisi *ON*, kunci kontak memutuskan hubungan terminal IG dan E, sehingga tegangan yang dihasilkan oleh *alternator* diteruskan ke sistem pengapian. Sistem pengapian dapat dioperasikan, disamping itu hubungan terminal BAT dan BAT 1 terhubung sehingga seluruh sistem kelistrikan dapat dioperasikan[7]. Pada penelitian ini digunakan sistem kunci kontak agar pemutusan atau penyambungan ignition coil (kunci kontak) lebih mudah dilakukan.

2.2.3. Internet of Things

Menurut Buku Internet of Things: Principles and Paradigms bahwa *Internet of Things* atau yang biasa disebut IoT terdiri dari dua pilar utama yaitu "*Internet*" dan "*Things*", jadi untuk objek apapun yang dapat terhubung ke internet akan dikategorikan sebgai "*Things*" seperti mencakup seperangkat entitas yang lebih umum seperti *smartphone*, *sensors*, manusia dan objek lainnya. Konteksnya mampu berkomunikasi dengan entitas lain, membuatnya dapat diakses kapan saja, dimana saja. Secara garis besar dengan *Internet Of Things* (IOT) objek harus dapat diakses tanpa batasan waktu atau tempat[8]. Untuk ekositem IoT dapat dilihat pada gambar 2.3.



Gambar 2.1.Ekosistem IoT[8]

Pada penelitian ini konsep *Internet Of Things* (IOT) digunakan sebagai konsep utama android dengan sepeda motor dengan menggunakan microcontroller *raspberry pi,* sensor yang digunakan adalah sensor *vibration* untuk mendapatkan informasi getaran dan modul relay untuk melakukan pengontrolan kunci kontak dan *electical starter*.

2.2.4. Web Server

Web server amerupakan suatu perangkat lunak yang dijalankan pada komputer server dan berfungsi agar dokumen internet server yang mampu untuk melayani koneksiperpindahan data dalam protokol http *web server* disamping *e-mail*. *Middleware* adalah perangkat lunak yang bekerja sama dengan web server dan berfungsi menterjemahkan kode - kode tertentu, menjalankan kode - kode tersebut dan memungkinkan berinteraksi dengan basis data. Dikarenakan web server dirancanguntuk menampilkan data, dimulai dari teks, *hypertext*, gambar yang merupakan eunggulan dari web sehingga web tidak hanya dapat diterima di

universitas tetapidi seluruh perusahaan komersial yang dapat menampilkan datanya dalam internet.Macam-macam web server antara lain Apache (*Open Source*), Xitami , IIS, PWS.Sedangkan, Web Browser adalah salah satu perangkat lunak disisi client yang digunakan untuk mengakses informasi web[9].

2.2.5. MQTT

MQTT atau *Message Queue Telemetry Transport* merupakan protokol sangat sederhana dan ringan dengan arsitektur *publish*/*subscribe* yang dirilis oleh IBM pada tahun 1999. MQTT direncanakan untuk mengirim data secara akurat di tanpa *delay* jaringan yang lama dengan kondisi jaringan bandwidth rendah.

A. Konsep Dasar MQTT

1. Publish* / *Subscribe

Dalam protokol MQTT, *publisher* mengirimkan pesan dan *client* yang berlangganan topik yang biasa disebut sebagai model *Publishing* / *Subscribing*. *Subscriber* berlangganan ke topik tertentu yang berhubungan dengan mereka dan dengan yang menerima setiap pesan dipublikasikan ke topik tersebut . Di sisi lain, klien dapat mempublikasikan pesan ke topik, sedemikian rupa sehingga memungkinkan semua pelanggan untuk mengakses pesan dari topik tersebut.

2. Topic* dan *Subscription

Pada MQTT, *publisher* mempublikasikan pesan ke topik yang dianggap sebagai subjek pesan. *Subscriber*, dengan demikian, berlangganan topik untuk mendapatkan pesan tertentu. Berlangganan topik dapat diungkapkan, yang membatasi data yang dikumpulkan untuk topik tertentu. Topik berisi dua tingkat *wildcard* topik.

3. Quality of Services level

Pada MQTT terdapat tiga tingkatan QoS antara lain.

1. *QoS0 (At most once)* : Dalam tingkat kualitas layanan ini, pesan dikirim paling banyak sekali dan tidak memberikan jaminan pengiriman pesan

2. *QoS1 (At least once)* : Dalam tingkat kualitas layanan ini, data dikirim setidaknya sekali dan dimungkinkan untuk mengirimkan pesan lebih dari sekali dengan menetapkan nilai flag duplikat sebesar 1.
3. *QoS2 (Exactly once)* : Dalam tingkat kualitas layanan ini, pesan dikirim tepat sekali dengan menggunakan *handshaking* 4 arah[10].

B. Arsitektur MQTT

Sistem umum MQTT pada umumnya membutuhkan dua komponen, yaitu *MQTT Client* dan *MQTT Broker*.

1. MQTT Client

MQTT Client merupakan sebuah perangkat lunak yang akan dipasang pada perangkat yang akan melakukan proses *publish/subscribe*, pada raspberry biasanya menggunakan pustaka mqtt.js.

2. MQTT Broker

MQTT Broker merupakan sebuah broker yang dapat menangani *publish/subscribe* data. Unutk broker mqtt ini sendiri terdapat beberapa broker yang biasa digunakan seperti, mosquito, cloudmqtt, HiveMQ, PubNub dll.

MQTT memiliki beberapa sinyal kontrol namun hanya 5 sinyal utama yang dipakai langsung oleh klien seperti *CONNECT*, *PUBLISH*, *SUBSCRIBE*, *UNSUBSCRIBE* dan *DISCONNECT*. Berikut sinyal kontrol tersebut.

1. CONNECT (*Client request to connect to Server)*
2. PUBLISH (*A message which represents a new/separate publish*)
3. SUBSCRIBE (*A message used by clients to subscribe to specific topics*)
4. UNSUBSCRIBE (*A message used by clients to unsubscribe from specific topics*)
5. DISCONNECT (*Graceful disconnect message sent by clients before disconnecting*)

Dalam MQTT terdapat istilah *topic* yaitu berupa UTF-8 string yang perannya hampir sama seperti topik pada chat hanya saja lebih sederhana dan berfungsi

sebagai filter untuk broker dalam mengirimkan pesan ke tiap klien yang terhubung atau bisa diumpamakan *topic* adalah kanal bagi klien untuk *subscribe*[10].

Pada MQTT klien tak hanya bisa menjadi subscriber namun dapat menjadi subscriber. Jadi misalnya pada peneitian ini perangkat *mobile* dan Raspberry Pi dua-duanya bisa menjadi *subscriber* satu sama lain ataupun menajdi *publisher* satu sama lain hanya saja keduanya terhubung dengan topic tertentu melalui broker. Pada Penelitian ini Protokol MQTT digunakan untuk transaksi data antara alat yang terpasang pada sepeda motor dengan perangkat *smartphone*.

2.2.6. Tracking

Tracking memiliki arti mengikuti jalan, atau dalam arti bebasnya adalah suatu kegiatan untuk mengikuti jejak suatu obyek. Pengertian tracking atau pemantauan dalam hal ini adalah kegiatan untuk memantau keberadaan anak berdasarkan posisi yang di dapatkan dari smartphone[11].

2.2.7. Android

Android adalah software platform yang open source untuk mobile device.Android berisi sistem operasi, middleware dan aplikasi-aplikasi dasar. Basis OSAndroid adalah kernel linux 2.6 yang telah dimodifikasi untuk mobile device[2]. Pada penelitian ini menggunakan android sebagai pengganti kunci kontak (ignition coil).

2.2.8. Android Studio

Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat open source atau gratis. Peluncuran Android Studio ini diumumkan oleh Google pada 16 mei 2013 pada event Google I/O *Conference* untuk tahun 2013. Sejak saat itu, Android Studio mengantikan *Eclipse* sebagai IDE resmi untuk mengembangkan aplikasi *Android*[12].

Android studio sendiri dikembangkan berdasarkan IntelliJ IDEA yang mirip dengan *Eclipse* disertai dengan ADT plugin (*Android Development Tools*). *Android studio* memiliki fitur :

1. Projek berbasis pada *Gradle Build*.
2. *Refactory* dan pembenahan bug yang cepat.
3. Tools baru yang bernama “Lint” dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.
4. Mendukung *Proguard And App-signing* untuk keamanan.
5. Memiliki GUI aplikasi android lebih mudah.
6. Didukung oleh *Google Cloud Platfrom* untuk setiap aplikasi yang dikembangkan.

2.2.9. JDK (*Java Development Kit*)

Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat kamu gunakan untuk mengembangkan perangkat lunak yang berbasis Java, sedangkan JRE adalah sebuah implementasi dari Java Virtual Machine yang benarbenar digunakan untuk menjalankan program java. Baisanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber compiler java, bundling, debuggers, development libraries dan lain sebagainya[12].

2.2.10. Object Oriented (OO)

Menurut buku Analisis Perancangan Sistem Berorientasi Objek dengan UML, *Object oriented* merupakan suatu obyek memiliki keadaan sesaat (*state*) dan perilaku (*behaviour*). State sebuah obyek adalah kondisi obyek tersebut yang dinyatakan dalam *attribute*/*properties*. State sebuah obyek adalah kondisi obyek tersebut yang dinyatakan dalam *attribute*/*properties*. Sedangkan perilaku suatu obyek mendefinisikan bagaimana sebuah obyek bertindak/beraksi dan memberikan reaksi. Perilaku sebuah objek dinyatakan dalam operation. Menurut schmuller, *attribute* dan *operation* bila disatukan akan memberikan *fitur*/*features*. Himpunan objek-objek yang sejenis disebut class. Objek adalah contoh *instance* dari sebuah class. Ada dua macam aplikasi yang tidak cocok dikembangkan dengan metode OO. Yang pertama adalah aplikasi yang sangat berorientasi ke database. Aplikasi yang sangat berorientasi ke penyimpanan dan pemanggilan data sagnat tidak cocok dikembangkan dengan OO karena akan banyak kehilangan manfaat dari penggunaan RDBMS (Relational Database Management System) untuk

penyimpanan data. Akan tetapi RDBMS juga mempunyai keterbatasan dalam penyimpanan dan pemanggilan struktur data yang kompleks seperti multimedia, data spasial dan lain-lain. Bentuk aplikasi lain yang kurang cocok dengan pendekatan OO adalah aplikasi yang membutuhkan banyak algoritma. Beberapa applikasi yang melibatkan perhitungan yang besar dan kompleks[13]. Pada penelitian ini konsep OO digunakan untuk konsep pengkodean pada sistem yang akan dibangun. Berikut ini adalah konsep-konsep dalam pemrograman berorientasi objek :

1. ***Class***

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

2. ***Object***

   Objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan teknik yang efektif dalam menemukan cara yang tepat dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak. Orientasi objek merupakan teknik pemodelan sistem riil yang berbasis objek.

3. ***Abstraction***

   Kemampuan sebuah program untuk melewati aspek informasi yang diolah adalah kemampuan untuk fokus pada inti permasalahan. Setiap objek dalam sistem melayani berbagai model dari pelaku abstrak yang dapat melakukan kerja, laporan dan perubahan serta berkomunikasi dengan objek lain dalam sistem, tanpa harus menampakkan kelebihan diterapkan.

4. ***Encapsulation***

Encapsulation adalah proses memastikan pengguna sebuah objek tidak dapat menggantikan keadaan dari sebuah objek dengan cara yang tidak sesuai prosedur. Artinya, hanya metode yang terdapat dalam objek tersebut yang diberi izin untuk mengakses keadaan yang diinginkan. Setiap objek mengakses *interface* yang menyabutkan bagaimana objek lainnya dapat berintegrasi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut

5. ***Polimorfism***

   Polimorfism merupakan suatu fungsionalitas yang diimplikasikan dengan berbagai cara yang berbeda. Pada program berorientasi objek, pembuat program dapat memiliki berbagai implementasi untuk sebagian fungsi tertentu.

6. ***Inheritance***

   Seperti yang sudah diuraikan di atas, objek adalah contoh / *instance* dari sebuah *class*. Hal ini mempunyai konsekuensi yang penting yaitu sebagai *instance* sebuah *class*, sebuah objek mempuyai semua karakteristik dari classnya. Inilah yang disebut dengan *Inheritance* (pewarisan sifat). Dengan demikian apapun *attribute* dan *operation* dari *class* akan dimiliki pula oleh semua obyek yang disinherit/diturunkan dari class tersebut. Sifat ini tidak hanya berlaku untuk objek terhadap *class*, akan tetapi juga berlaku untuk *class* terhadap *class* lainnya[13].

2.2.11. UML

Unified Modeling Language (UML) Merupakan salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan karena UML menyediakanbahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat blueprint atas visi mereka dalam bentuk yang baku,mudah dipahami serta dilengkapi dengan mekanisme yang efektif untuk berbagi dan mengkomunikasikan perancangan mereka dengan yang lain.

Unified Modeling Language (UML) merupakan kesatuan dari Bahasa permodelan yang dikembangkan booch, *Object Modeling Technique* (OMT) dan *Object Orented Software Engineering* (OOSE) Metode ini menjadikan proses analisis dan design kedalam empat tahapan iterative, yaitu identifikasi kelas-kelas, dan objek-objek, identifikasi semantic dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode Booch adalah pada detail dan kayanya dengan notasi dan elemen, permodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan entity-relationship. Tahapan utama dalam metodologi ini adalah nalisis, design sistem, design objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung konsep OO[13].

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.

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].

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

1. *Scenario*, menggambarkan interaksi diantara objek dan diantara proses. *Scenario* ini digunakan untuk dientifikasi elemen arsitektur, ilustrasi dan validasi disain arsitektur serta sebagai titik awal untuk pengujian prototype arsitektur. *Scenario* ini bisa juga disebut dengan *use case view*. *Use case view* ini mendefinisikan kebutuhan sistem karena mengantdung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari rancangan sistem. Itulah sebabnya *use case view* menajdi pusat peran dan sering dikatakan yang mengdrive proses pengembangan perangkat lunak.
2. *Development View*, menjelaskan sebuah sistem dari perspektif programmer dan terkonsentrasikan ke manajemen perangkat lunak. *View* ini dikenal juga sebagai *implementation view*. Diagram UML yang termasuk dalam development *view* di antaranya adalah *component diagram* dan *package diagram*.

3. *Logical View*, terkait dengan fungsionalitas sistem yang dipersiapkan untuk pengguna akhir. *Logical view* mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi *object diagram, class diagram, state machine diagram* dan *composite structure diagram*.
4. *Physical View*, menggambarkan sistem dari perspektif *sistem engineer*. Fokus dari *physical view* adalah *topologi* sistem perangkat lunak. *View* ini dikenal juga sebagai deployment view. Yang termasuk dalam *physical* view ini adalah *deployment diagram* dan *timing diagram*.
5. *Process View*, berhubungan erat dengan aspek dinamis dari sistem, proses yang terjadi di sistem dan bagaimana komunikasi yang terjadi di sistem serta tingkah laku sistem saat dijalankan. *Process view* menjelaskan apa itu *concurrency, distribusi integrasi, kinerja* dan lain-lain. Yang termasuk dalam process *view* adalah *activiy diagram, communication diagram, sequence diagram* dan *interaction overview diagram*.

Mesikpun UML sudah cukup banyak menyediakan diagram yang bisa membantu mendefinisikan sebuah aplikasi, tidak berarti bahwa semua diagram tersebut akan bisa menjawab persoalan yang ada. Dalam banyak kasus, diagram lain selain UML sangat banyak membantu. Pada penelitian ini konsep UML digunakan untuk menggambarkan bagaimana sistem bekerja, hubungan antar class, memberi gambaran kepada user sistem yang akan di gunakan dan memberikan penjelasan detail pemanggilan fungsi-fungsi atau method dalam class[13].

Pada penelitian ini *Unified Modeling Language* digunakan untuk merancang dan membangun sistem keamanan sepeda motor berbasis *internet of things* dan protokol mqtt

2.2.12. Diagram UML

UML menyediakan 4 macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek. 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. Dengan demikian secara singkat bisa dikatan *use case* adalah serangkaian *scenario* yang digabungkan Bersama-sama oleh tujuan umum pengguna. 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[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.

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*.

c. *Relationship*

Relationship adalah hubungan antara *use cases* dengan *actors*. *Relationship* dalam *use case* diagram meliputi:

a. Asosiasi antara *actor* dan *use case*.

   Hubungan antara *actor* dan *use case* yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis lurus dari actor menuju *use case* baik dengan menggunakan mata panah terbuka ataupun tidak.

b. Asosiasi antara 2 *use case*

   Hubungan antara *use case* yang satu dan *use case* lainnya yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis putus-putus/garis lurus dengan mata panah terbuka di ujungnya.

c. Generalisasi antara 2 *actor*

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

d. Generalisasi antara 2 *use case*.

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

2. *Class Diagram*

Class diagram 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].

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.

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].

Berikut ini merupakan komponen dalam activity diagram, yaitu :

a. *Activity node*

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

b. *Activity edge*

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

c. *Initial state*

   Bentuk lingkaran berisi penuh melambangkan awal dari suatu proses.

d. *Decision*

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

e. *Fork*

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

f. *Join*

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

g. *Final state*

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

4. *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukan sejumlah contoh obyek dan

message (pesan) yang diletakan diantara obyek-obyek ini di dalam use case. Komponen utama *sequence diagram* terdiri atas objek yang ditulisan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukan dengan *progress vertical*. Objek diletakan di detak bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram, istilah objek dikenal juga dengan *participant*, setiap *participant* terhubung dengan garis titik-titik yang disebut *lifeline*. Sepanjang *lifeline* ada kotak yang disebut *activation*, *activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*. Sebuah *message* bisa jadi simple, *synchronous* atau *asynchronous*. *Message* yang simple adalah sebuah perpindahan (*transfer*) *control* dari satu *participant* ke *participant* yang lainnya. Jika sebuah *participant* mengirimkan sebuah message *synchronous*, maka jawaban atas *message* tersebut akan ditunggu sebelum diproses dengan ururs annya. Namun jika message message *asynchronous* yang dikirimkan, maka jawaban atas *message* tersebut tidak perlu ditunggu. *Time* adalah diagram yang mewakili waktu pada arah *vertical*. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijadikan terlebih dahulu dibanding *message* yang lebih dekat ke bawah[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.

2.2.13. Python

Dalam buku *Learning Python : Powerfull Object Oriented Programming*, Python merupakan bahasa pemrograman berorientasi objek. Python dapat didefinisikan memadukan dukungan *oriented object programing* secara keseluruhan terhadap peran baris koding. Dapat dikatakan bahwa python mungkin lebih dikenal sebagai bahasa pemrograman tujuan umum yang memadukan paradigma prosedural, fungsional, dan object-oiented[14].

2.2.14. Raspberry Pi 3B

Raspberry Pi 3 Model B merupakan generasi ketiga dari keluarga *raspberry pi*. *raspberry pi* 3 memiliki *RAM* 1GB dan grafis *Broadcom VideoCore* IV pada frekuensi *clock* yang lebih tinggi dari sebelumnya yang berjalan pada 250MHz. *Raspberry Pi* 3 juga memiliki 4 *USB port*, 40 *pin GPIO*, *Full HDMI port*, *Port Ethernet*, *Combined 3.5mm audio jack and composite video*, *Camera interface* (CSI), *Display interface* (DSI), slot kartu *Micro SD* (Sistem tekan-tarik, berbeda dari yang sebelum nya ditekan-tekan), dan V*ideoCore* IV *3D graphics core*[15]. Pada penelitian ini digunakan sebagai mikrokontroller yang dipasang pada sepeda motor unutk mengontrol kunci kontak maupun *starter*.

2.2.15. Relay

Relay merupakan komponen elektronika yang memiliki fungsi bekerja sebagai saklar mekanik yang digerakkan oleh energi listrik. *Relay* menggunakan gaya elektromagnetik untuk memutuskan atau menghubungkan suatu rangkaian elektronika yang satu dengan rangkaian elektronika yang lainnya. *Relay* terdiri dari *coil* dan *contact*. *Coil* adalah gulungan kawat yang mendapat arus listrik, sedangkan *contact* adalah sejenis saklar yang pergerakannya tergantung dari ada tidaknya arus listrik di *coil*. *Contact* ada 2 jenis yaitu *normally open* (kondisi awal sebelum diaktifkan *open*), dan *normally closed* (kondisi awal sebelum diaktifkan *closed*)[16]. Pada penelitian ini menggunakan relay 2 CH sebagai penghubung dan pemutus arus listrik pada motor.

2.2.16. Modem

Modem adalah singkatan dari modulator demodulator. Komunikasi data bisa berupa analog maupun digital. Komunikasi/transmisi digital terdiri dari dua status sinyal On dan Off, 1 dan 0, hitam dan putih, sedangkan transmisi analog terdiri dari nilai-nilai dalam suatu daerah lebar[17]. Pada penelitian ini menggunakan modem dongle sebagai sumber jaringan untuk alat yang terpasang pada sepeda motor.

2.2.17. Sensor Getar

Merupakan salah satu sensor yang dapat mengukur getaran suatu benda yang nantinya dimana data tersebut akan diproses untuk kepentingan percobaan ataupun di gunakan untuk mengantisipasi sebuah kemungkinan adanya mara bahaya. Salah satu jenis sensor getaran yang saat ini sering di gunakan adalah accelerometer, alat ini merupakan alat yang dapat berfungsi untuk mengukur percepatan dari sebuah benda. Percepatan tersebut di ukur bukan dengan menggunakan koordinat dari percepatan tersebut, melainkan dengan mengukur percepatan berdasarkan fenomena pergerakan benda yang di hubungkan dengan perubahan massa yang terjadi di dalam alat pengukur tersebut[18]. Pada penelitian ini menggunakan sensor getar untuk pemicu peringatan kepada pemilik bahwa kendaraan dalam upaa pencurian.

2.2.18. DC to DC Converter

DC Stepdown converter merupakan peralatan yang dapat mengkonversikan energi listrik dari tegangan searah tertentu ke tegangan searah yang diinginkan atau teregulasi dengan nilai voltase yang lebih kecil dari catu daya masukan[2]. Pada penelitian ini *DC Converter* yang digunakan pada penelitian ini yang bertipe *stepdown* atau penurun tegangan sumber dari accu.

2.2.19. 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[19]. Pada penelitian ini digunakan untuk pengetesan perangkat lunak, aga mengetahui kesesuaian dengan konsep dan fungsi.