BAB 2
TINJAUAN PUSTAKA

2.1 Pengenalan Antena

Antena adalah suatu perangkat yang mentransformasikan sinyal elektromagnetik dari saluran transmisi ke dalam bentuk sinyal radiasi gelombang elektromagnetik dalam ruang bebas, dan juga menangkap dan mengumpulkan gelombang elektromagnetik dari ruang bebas dan mentransformasikannya ke dalam bentuk saluran transmisi [1]. Dari defenisi tersebut dapat diartikan bahwa antena merupakan suatu transformator struktur transmisi antara gelombang terbimbing (saluran transmisi) dengan gelombang ruang bebas berupa suatu gelombang elektromagnetik (GEM) atau sebaliknya.

Antena merupakan *interface* antara daerah transmisi dengan udara bebas. Antena berfungsi sebagai pelepas energi elektromagnetik ke udara ruang bebas dan penerima energi elektromagnetik dari ruang bebas. Karakteristik antena sebagai pemancar juga berlaku pada antena sebagai penerima sesuai dengan Teorema Resiprositas Carson.

Pengetahuan dan subbab tentang pengenalan antena diperlukan karena pada penelitian ini fokus permasalahan utamanya berkaitan dengan pointing antena dan instalasinya. Tentunya dengan adanya pembahasan ini dapat menjadi landasan pemahaman dari apa itu antena dan fungsinya. Pembahasan tentang antena ini juga akan diperdalam lagi dengan pola radiasi antena, polarisasi dan jenis-jenis antena.

2.1.1 Pola Radiasi

Pola radiasi adalah representasi intensitas pancaran antena sebagai fungsi dari parameter koordinat bola (θ, φ). Komponen-komponen dalam gambaran pola radiasi adalah mainlobe, sidelobe, dan backlobe. Mainlobe adalah lobe yang terdapat nilai gain maksimum dimana didalam mainlobe bisa didapatkan parameter penting dalam suatu antenna yaitu *Half Power Beamwidth* (HPBW) dan *First Null Beamwidth* (FNBW).

Pola radiasi antena diukur pada daerah medan jauh antena, karena pada daerah tersebut gelombang elektromagnetik yang terpancar bersifat transversal penuh dan antena tidak dipengaruhi oleh benda di sekelilingnya. Terdapat 3 jenis pola radiasi pada antena [1]:

1. Pola Radiasi *Isotropic*

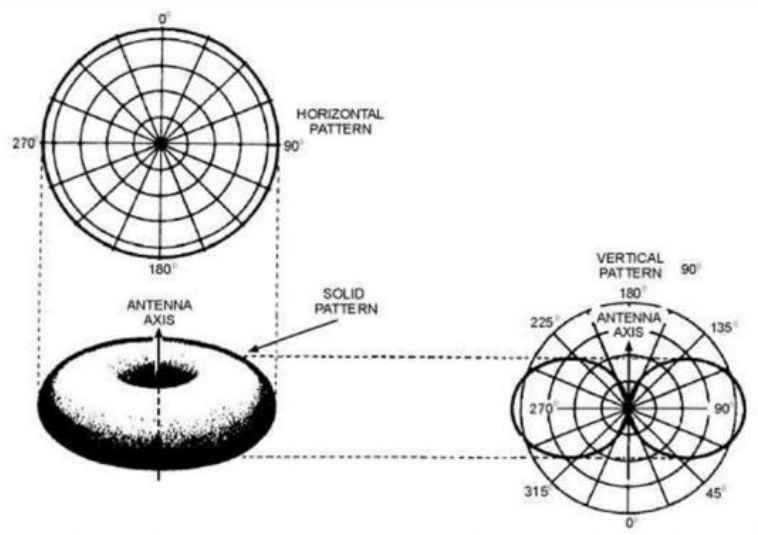
   Radiasi *isotropic* adalah teori antena yang memancarkan kekuatan secara seragam ke segala arah. Oleh karena itu, kadang-kadang disebut radiator non-arah. Kemampuan radiasi tiga dimensi semacam itu dimungkinkan hanya dengan radiator titik. Meski murni hanya teoritis, radiasi *isotropic* ini paling dikenal karena penggunaannya sebagai referensi. Radiator isotropik digunakan sebagai antena referensi untuk menentukan gain arah antena. Sebagai praktik umum, nilai gain antena dinyatakan sebagai "x dB di atas radiator isotropik" dan diberi label sebagai "x dBi".

2. Pola Radiasi *Omnidirectional*

   Antena omnidirectional adalah antena yang memancarkan daya ke segala arah, dan bentuk pola radiasinya digambarkan seperti bentuk donat (doughnut) dengan pusat berimpit. Antena ini ada dalam kenyataan, dan dalam pengukuran sering digunakan sebagai pembanding terhadap antena yang lebih kompleks. Contoh antena ini adalah antena dipole setengah panjang gelombang.

   Pola radiasi dari antena ini desebut toroid (yaitu donat) atau gambar angka delapan, dengan poros toroid sejajar dengan sumbu antena. Pada sebagian besar sistem radio yang menggunakan antena *omnidirectional*, bidang gain puncak adalah bidang horizontal sejajar dengan permukaan bumi setempat. ERP dari antena omnidirectional mengacu pada kekuatan gelombang yang terpancar ke segala arah di bidang referensi. *Effective (or equivalent) Radiated Power* (ERP) dapat diperoleh dengan mengalikan daya terpancar total, P_{rad}, dengan referensi gain puncak ke antena dipole, g_{dipole}.

$$ERP = P_{rad} \times g_{dipole}$$

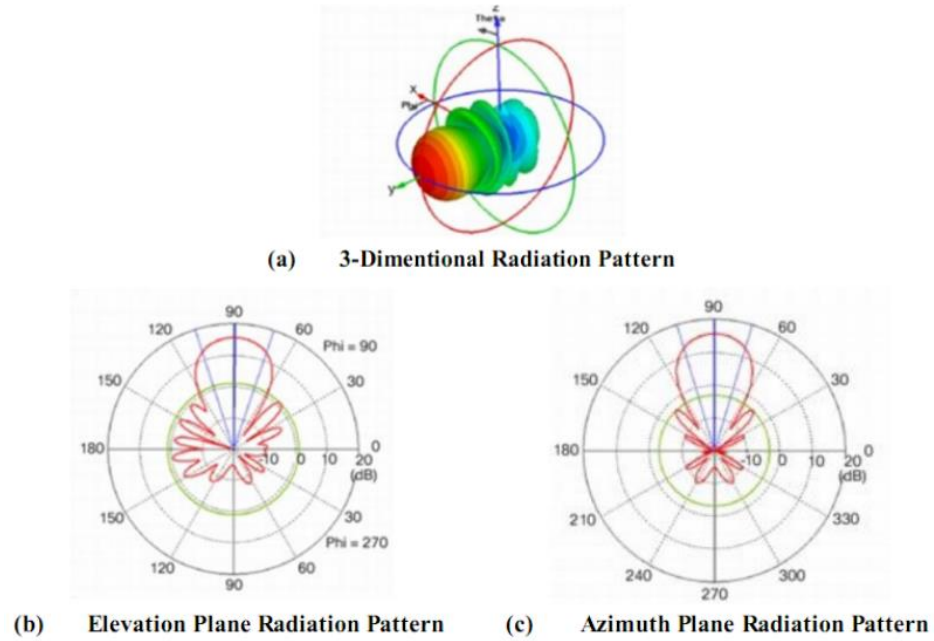


Gambar 2.1 Pola Radiasi *Omnidirectional Antenna*

3. Pola Radiasi *Directional*

   Antena directional merupakan antena yang memancarkan daya ke arah tertentu. Gain antena ini relatif lebih besar dari antena omnidirectional. Contoh, suatu antena dengan gain 10 dBi (kadang-kadang dinyatakan dengan "dBic" atau disingkat "dB" saja). Artinya antena ini pada arah tertentu memancarkan daya 10 dB lebih besar dibanding dengan antena isotropis. *Effective (or equivalent) Isotropic Radiated Power* (EIRP) antena directional mengacu pada kekuatan gelombang yang terpancar ke arah poros mainbeam. EIRP dapat diperoleh dengan mengalikan daya terpancar total, P_{rad}, dengan gain daya puncak, $g_{p.}$

$$EIRP = P_{rad} \times g_p$$



Gambar 2.2 Pola Radiasi *Directional Antenna*

Pengetahuan dan subbab tentang pola radiasi antena diperlukan karena pola radiasi dari suatu antenna dapat mempengaruhi hasil pointing antenna. Pointing antenna hanya dilakukan untuk antenna yang memiliki pola radiasi *directional* yang berfokus untuk meberikan *coverage* pada daerah-daerah tertentu. Tentunya dengan adanya pembahasan ini dapat menjadi landasan pemahaman dari apa itu pola radiasi antenna dan jenis-jenisnya.

2.1.2 Polarisasi Antena

Polarisasi antena dapat didefinisikan jejak kurva yang dibentuk oleh kuat medan listrik, yang diamati pada bidang konstan tegak lurus arah perambatan. Biasanya arah yang dimaksud adalah arah gain maksimum [1]. Polarisasi dapat dikelompokkan menjadi polarisasi eliptik, sirkular dan linier. Polarisasi medan perlu diketahui agar pemasangan antena penerima sesuai dengan antena pemancar atau sebaliknya sehingga tidak ada daya yang hilang disebabkan oleh *polarization loss factor* (PLF). Selain itu dengan polarisasi bisa digunakan untuk mencegah interferensi sinyal antena lain.

Polarisasi antena mengacu pada polarisasi gelombang *far-field* EM yang dihasilkan oleh antena, sementara polarisasi gelombang EM mengacu pada polarisasi medan listrik gelombang EM. Gelombang EM bisa linear, sirkular, atau elips terpolarisasi. Untuk gelombang EM terpolarisasi linier, arah medan listriknya konstan terhadap arah perambatan gelombang. Untuk gelombang EM yang terpolarisasi secara elips, arah medan listriknya berputar sesuai dengan arah propagasi gelombang, dan dapat berupa *Right-Hand Circular Polarization* (RHCP) atau *Left-Hand Circular Polarization* (LHCP). Polarisasi melingkar adalah kasus khusus polarisasi elips dimana sumbu mayor dan minor rotasinya sama.

Dalam komunikasi radio, polarisasi linier ditentukan sesuai dengan hubungan polarisasi ke Bumi, dan biasanya horizontal atau vertikal. Untuk layanan radio terestrial, sinyal radio dipolarisasi secara horizontal bila medan listriknya sejajar dengan bidang horizontal (atau ground), dan dipolarisasi secara vertikal saat medan listriknya tegak lurus terhadap bidang horizontal. Untuk layanan radio luar angkasa, sinyal radio dipolarisasi secara horizontal saat medan listrik sejajar dengan bidang ekuator (atau arah timur-barat), dan terpolarisasi secara vertikal saat medan listriknya tegak lurus terhadap bidang ekuator (atau irisan utara-selatan).

Karakteristik lingkungan propagasi dan operasi sistem radio dapat mendikte pemilihan polarisasi. Misalnya, dalam kasus lingkungan propagasi, polarisasi vertikal adalah pilihan yang lebih disukai untuk sistem relay radio terestrial atau sistem penyiaran karena karakteristik kerugian dari tanah. Atau, polarisasi melingkar adalah pilihan yang lebih disukai untuk sistem ruang angkasa yang beroperasi pada pita frekuensi yang tunduk pada rotasi Faraday di ionosfer. Dalam kasus operasi, polarisasi melingkar adalah pilihan yang lebih disukai untuk sistem satelit bergerak karena sulit untuk menyelaraskan linier polarisasi terminal stasiun bergerak bumi.

Seperti halnya pola radiasi antenna, pengetahuan dan subbab tentang polarisasi antena diperlukan karena pada saat melakukan pointing antenna pemasangan antara antenna pemancar dan antenna penerima harus sesuai agar tidak terjadi interferensi dari sinyal lain. Dengan lebih memahami polarisasi antenna juga dapat mengetahui apakah ada daya yang hilang ketika melakukan pemasangan antenna tersebut akibat dari *polarization loss factor*. Tentunya dengan adanya pembahasan ini dapat menjadi landasan pemahaman dari apa itu polarisasi antena.

2.1.3 Jenis-jenis Antena

Antena memiliki dalam banyak struktur, bentuk, dan ukuran. Jenis antena yang paling umum *adalah wire antennas, aperture antennas, fractal antenna,* dan *microstrip antennas*. Tipe umum lainnya adalah antena array yang terdiri dari beberapa elemen pemancar yang diatur dan dioperasikan dalam korelasi tertentu. Beberapa jenis antena yang umum adalah sebagai berikut [1]:

1. *Wire Antenna*

   Wire Antenna adalah antena yang dibangun dengan kabel atau kawat. Bentuk kabel bisa lurus, melingkar, heliks, dan variasi lainnya. Terkadang kabelnya bisa diratakan menjadi pelat lama. Beberapa antena kawat biasa, yaitu antena *dipole* dan variasinya, antena *loop*, antena heliks, antena Yagi-Uda, dan antena belah ketupat.

   a. *Dipole Antenna*

      Antena dipol terdiri dari dua jalur listrik lurus yang dipisahkan oleh celah kecil di tengahnya, dan daya diberi makan di celah tengah. Saluran listrik harus sangat tipis, yaitu dimensi linier penampangnya jauh lebih kecil dari panjang garis. Ini adalah antena omnidirectional, dan polarisasinya ditentukan oleh orientasi garis listrik yang bisa horizontal, vertikal, atau miring.

b. *Loop Antenna*

Antena loop adalah lingkaran dari saluran listrik dengan dua ujung yang menghubungkan ke dua konduktor pada saluran transmisi. Loop mungkin berbentuk asal-asalan seperti melingkar, persegi, segitiga, atau belah ketupat, dan gain antena sebanding dengan daerah yang tertutup oleh lingkaran. Antena loop lebih kebal kebisingan karena tidak memerlukan *ground plane*.

c. *Helix Antenna*

Antena *helix* adalah jalur listrik yang dililitkan secara heliks yang dipasang pada bidang tanah. Antena heliks dapat beroperasi dalam mode aksial atau normal. Polarisasi umum bersifat elips, namun polarisasi linear dan melingkar dapat dicapai dalam kondisi khusus. Orientasi polarisasi, RHCP atau LHCP, ditentukan oleh orientasi koil.

d. *Yagi-Uda Antenna*

Antena Yagi-Uda, yang kadang disebut antena Yagi, terdiri dari antena dipol dan elemen pasif tambahan. Antena dipol adalah elemen tengah dan radiator. Antena yagi secara teoritis yaitu sejenis antena yang terdiri dari 3 macam elemen. Dimana 3 macam yang memegang peranan penting dalam konstruksi antena yagi yaitu reflektor, dipole dan direktori dalam peng-implementasianya sebuah antena yagi dapat dibuat dari elemen berbentuk pararel silindris.

e. *Rhombic Antenna*

Rhombic antenna memiliki empat saluran listrik yang dihubungkan dalam bentuk berlian "belah ketupat", dengan masing-masing garis biasanya satu λ panjang. Pada pola vertikal, null pada bidang horizontal adalah hasil *ground lossy* yang mempengaruhi pola radiasi. Arah balok menuju ujung resistor, dan gain bergantung pada geometri belah ketupat.

2. *Aperture Antenna*

   Aperture antenna meliputi antena *waveguide antennas, slot antennas* dan *horn antennas*.

   a. *Waveguide Antenna*

      Antena Waveguide adalah waveguide dengan ujung terbuka baik di ruang kosong atau dipasang pada bidang ground.

   b. *Slot Antenna*

      Antena slot adalah pelat logam dengan lubang atau slot yang dipotong. Slot bisa memancarkan gelombang EM saat piring membawa arus listrik. Bentuk dan ukuran slot menentukan pola radiasi dan gain.

   c. *Horn Antenna*

      Horn antenna adalah waveguide dengan ujung terbuka yang berkobar seperti tanduk. Flare bisa berbentuk persegi, persegi panjang, atau melingkar. Sebuah suar persegi panjang atau persegi membentuk tanduk piramid, dan suar melingkar membentuk tanduk berbentuk kerucut. Gain dari antena tanduk sebanding dengan ukuran aperture. Antena tanduk menawarkan keuntungan moderat dalam kemasan kecil dan kasar tanpa perlu penyesuaian.

3. *Fractal Antenna*

   Fraktal adalah bentuk geometris yang dapat dibagi menjadi beberapa bagian yang bentuknya adalah salinan ukuran dikurangi dari keseluruhan. Fraktal umumnya mirip diri dan tidak bergantung pada skala. Antena fraktal menggunakan fraktal untuk merancang bagian yang memancar. Menggabungkan properti fraktal dan sifat penskalaan pada teori antena, antena fraktal dapat beroperasi pada beberapa pita frekuensi atau pita frekuensi lebar. Antena fraktal kompak, dan banyak digunakan pada perangkat mobile seperti telepon seluler.

4. *Microstrip Antenna*

   Antena microstrip, yang juga dikenal sebagai antena patch atau antena cetak, terdiri dari tiga lapisan: bidang tanah metalik di bagian bawah, substrat dielektrik, dan radiator logam. Radiator diberi makan dengan microstrip di lapisan atas atau garis koaksial di bawah radiator melalui bidang tanah dan antena microstrip substrat A adalah antena narrowband, wide-beam. Ini memiliki profil rendah, sangat ringan, tidak mahal untuk diproduksi, dan mudah digabungkan dengan elektronik yang menyertainya. Karena fitur ini, telah menjadi populer untuk telepon seluler atau receiver sistem penentuan posisi global (GPS).

5. *Lens Antenna*

   Antena lensa adalah perangkat dielektrik atau kaca dengan bentuk paraboloid konveks yang menyebabkan cahaya menyatu atau menyimpang. Ini adalah perangkat *optic*. Oleh karena itu, cocok untuk mentransmisikan dan menerima gelombang EM dalam rentang frekuensi optik. Antena lensa digunakan sebagai alat bantu sensor optik meteorologis, teleskop radio, dan astronomi radio.

6. *Reflector Antenna*

   Antena reflektor memiliki radiator dan setidaknya satu permukaan yang memantulkan. Radiator, yang juga disebut pengumpan, bisa berupa antena linier atau antena *horn*. Antena reflektor yang paling umum adalah *corner reflector antenna*, *parabolic antenna*, *paraboloidal antenna*, dan *Cassegrain antenna*. Antena *paraboloidal* dan *Cassegrain* juga disebut *dish antenna*.

7. Array Antenna

   Antena *array* terdiri dari sekelompok radiator yang disusun dalam konfigurasi geometris khusus dan dioperasikan secara koheren sebagai satu kesatuan. Radiator individual bisa berupa antena apapun, seperti *wired*, *aperture*, *fractal* atau antena *microstrip*. Format *array* biasanya linier atau planar. Dengan menghubungkan amplitudo eksitasi dan fase

radiator, antena array dapat melakukan pemindaian balok dan fungsi lainnya secara elektronik dan bukan mekanis. Beberapa antena array khas, yaitu *log-periodic antenna, phased array, collinear array,* dan *slotted array*.

Pengetahuan dan subbab tentang jenis-jenis antena diperlukan karena tidak semua jenis antenna perlu dilakukan pointing, hanya beberapa jenis antenna tertentu saja yang perlu dilakukan pointing. Yaitu jenis-jenis antenna yang memiliki pola radiasi *directional*. Tentunya dengan adanya pembahasan jenis-jenis antenna ini dapat menjadi landasan pemahaman dari apa saja jenis-jenis antenna yang ada saat ini.

2.2 Pointing Antenna

Dalam instalasi antenna, khususnya untuk komunikasi antara *directional antenna* diperlukan suatu proses pointing (biasa disebut dengan pointing antenna) agar arah dari kedua antenna yang berkomunikasi tersebut memiliki *Line Of Sight* (LOS) yang sesuai. Hal ini akan berpengaruh pada kinerja dan propagasi antenna, semakin akurat pointing dari suatu antenna maka akan semakin baik juga kinerja dan propagasinya. Proses pointing antenna ini dapat dilakukan sebelum ataupun sesudah instalasi antenna untuk memperbaiki antenna yang sudah terpasang.

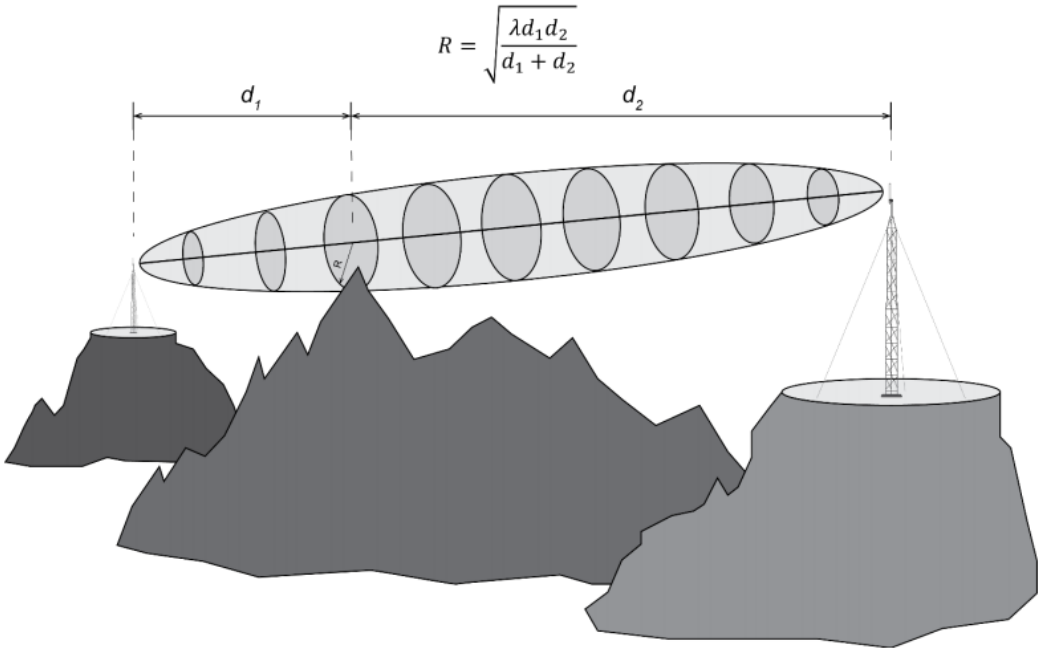
Proses pointing antenna berkaitan erat dengan perhitungan sudut azimuth. Sudut azimuth merupakan sudut yang dimulai dari arah utara kutub bumi berputar searah jarum jam dan diakhiri pada ujung objektif titik bidik. Dengan alat dititik A dan titik B maka dapat di ukur Azimut AB. Sudut azimuth ini hanya berlaku untuk *directional antenna*, bukan *omni antenna*. [2]

Berdasarkan penjelasan contoh kedua antenna direction diatas dapat diketahui bahwa pemahanan dan landasan teori mengenai antenna pointing sangat diperlukan. Pointing antenna diperlukan untuk jenis antenna *directional* yang hanya memiliki pola radiasi ke wilayah tertentu saja. Tentunya dengan adanya pembahasan pointing antenna ini dapat menjadi landasan pemahaman dari apa tujuan dan kapan dibutuhkannya antenna pointing.

2.2.1 *Fresnel Zone*

Sebelum memperkenalkan *fresnel zone,* terlebih dahulu kita perlu mengetahui apa itu *wireless*. *Wireless*, dalam bahasa Indonesia berarti jaringan tanpa kabel (nirkabel) yang menggunakan udara sebagai media transmisinya untuk menghantarkan gelombang elektromagnetik. *Wireless* telah lama berkembang, beberapa ilmuan pun telah membuktikan keberadaan wireless secar ilmiah dengan adanya penemuan-penemuan seperti radio, radar dan lain sebagainya. Dari masa ke masa penggunaan wireless semakin luas dan bertambah. Yang sering digunakan adalah wireless pada lokal area (WLAN). Yaitu LAN Yang menggunakan media transmisi wireless bisa berupa frekuensi radio, inframerah ataupun sinar laser untuk berkomunikasi di antara work station dan file server ataupun hub.

Sinyal nirkabel dipengaruhi oleh beberapa kondisi dan posisi. Salah satu metode untuk menentukan pengaruh kekuatan sinyal nirkabel adalah *fresnel zone*. *Fresenel zone* merupakan besaran tak terhingga dari pola radiasi yang terpancar keluar dari circular aperture. *Fresnel zone* ini adalah hasil pola difraksi dari circular aperture. Fresnel zone ini digunakan sebagai media rambat frekuensi dari gelombang elektromagnetik yang nantinya menghasilkan sebuah jaringan nirkabel. Fresnel zone adalah suatu daerah pada suatu lintasan transmisi gelombang mikro yang digambarkan berbentuk elips yang menunjukkan interferensi gelombang RF jika terdapat blocking [2]. Untuk lebih jelasnya dapat dilihat pada Gambar 2.3 dibawah ini.



Gambar 2.3 Ilustrasi Perhitungan *Fresnel Zone*

Keterangan:

R = radius dari fresnel zone

λ = panjang gelombang

d1 = jarak (km) pertama

d2 = jarak (km) pertama

Dan untuk menghitung panjang gelombang terkait dengan frekuensi sinyal propagasi adalah sebagai berikut:

$$\lambda = \frac{c}{f}$$

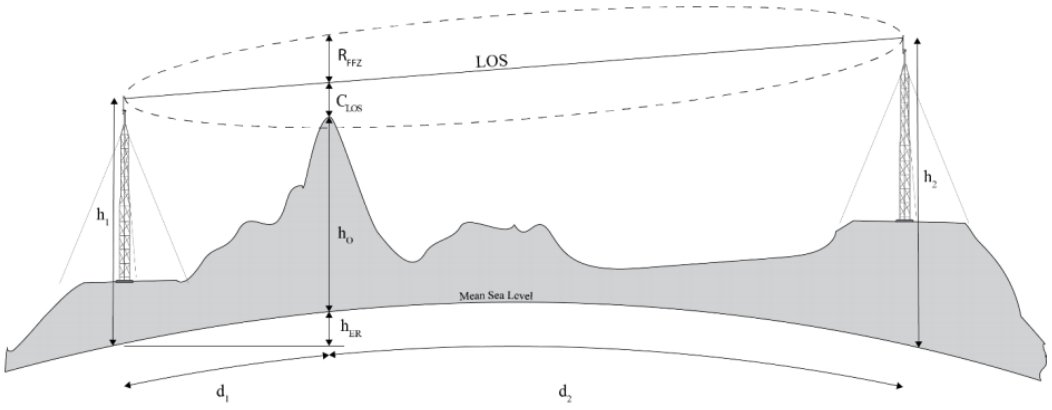
Keterangan:

λ = panjang gelombang

f = fekuensi (MHz)

c = kecepatan cahaya dalam meter per second (299.792 . 10^6 m/s)

Untuk menentukan apakah fresnel zone pertama pada beberapa titik di sepanjang jalur propagasi bebas dari penghalang atau tidak, harus dibangun *path profile* yang secara akurat menggambarkan lokasi dan elevasi potensi penghalang.



Gambar 2.4 *Path Profile* dalam *Fresnel Zone*

Keterangan:

LOS = jalur langsung propagasi LOS antar antena.
R_{FFZ} = radius zona Fresnel pertama di titik lokasi.
C_{LOS} = LOS clearance relatif terhadap obstruksi.
h_1 = tinggi antena dalam meter yang dikaitkan dengan jarak, d1.
h_2 = tinggi antena dalam meter yang terkait dengan jarak, d2.
h_O = tinggi dalam meter obstruksi.
d = jarak total jalur (d1 + d2) dalam kilometer.
H_{ER} = tinggi dalam meter kelengkungan permukaan di titik lokasi sebagai akibat dari *effective earth radius*.

Untuk H_{ER} dapat dihitung dengan rumus berikut, dimana k = *correction factor*:

$$h_{ER} = \frac{d_1 \cdot d_2}{12.74 \cdot k}$$

LOS clearance relatif terhadap obstruksi, $C_{LOS,}$ dapat dihitung dengan rumus:

$$C_{LOS} = h_1 + \frac{h_2 - h_1}{d} \cdot d_1 - h_{ER} - h_O$$

Karena dalam penelitian ini menggunakan *samples* maka h_{er} atau ketinggian *effective radius* tidak digunakan.

$$C_{LOS} = h1 \ + \frac{h2 - h1}{d} \ . d_1 - ho$$

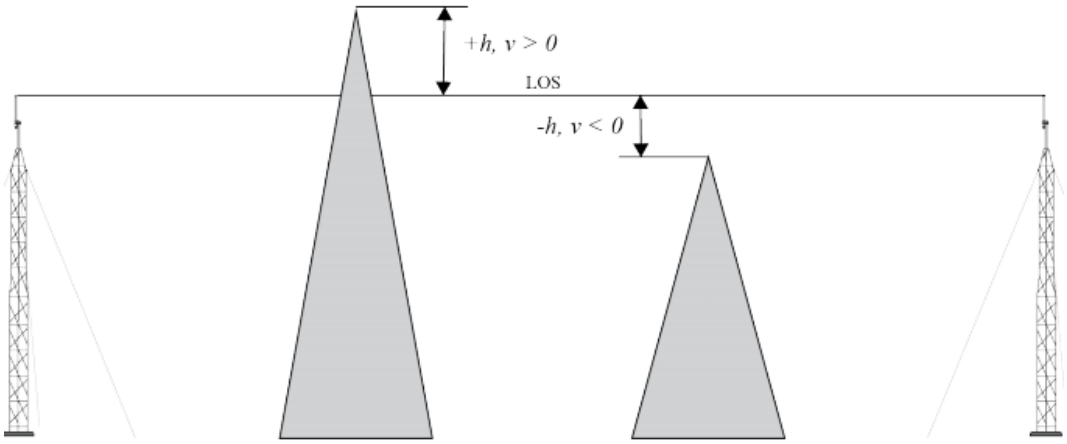
Kemudian untuk menghitung ketinggian LOS di suatu titik samples adalah penjumlahan dari C_{LOS} dengan ho:

$$H_{LOS} = C_{LOS} \ + \ ho$$

Sehingga jika didefinisikan dengan detail rumusnya menjadi:

$$H_{LOS} = h1 \ + \frac{h2 - h1}{d} \ . d_1$$

Nilai positif menunjukkan bahwa obstruksi berada di bawah LOS, sedangkan nilai negatif menunjukkan bahwa obstruksi berada di atas LOS.



Gambar 2.5 *Diffraction Parameter*

Ketinggian obstruksi relatif terhadap LOS dapat dihitung dengan persamaan:

$$h = h_O + h_{ER} - h_1 - \frac{h_2 - h_1}{d} \cdot d_1$$

Parameter difraksi Fresnel-Kirchhoff sekarang dapat dihitung sebagai berikut:

$$v = h \cdot \sqrt{\frac{2(d_1 + d_2)}{\lambda \cdot d_1 \cdot d_2}}$$

Untuk signal loss berikut persamaan yang dapat digunakan jika nilai $v \geq -0.7$

$$Loss_{Diff}(v) = 6.9 + 20\,log\left(\sqrt{(v - 0.1)^2 + 1} + v - 0.1\right)$$

Secara singkatnya, poin-poin berikut dapat menjadi acuan dari setiap rumus perhitungan *fresnel zone*:

1. Untuk jalur propagasi di mana *fresnel zone* pertama bebas dari penghalang (v ≤-1), kehilangan sinyal karena pantulan atau difraksi akan dapat diabaikan.
2. Untuk jalur propagasi di mana lebih dari 60% dari radius *fresnel zone* pertama terhalang oleh obstruksi jenis *knife-edge*, tetapi LOS langsung tidak terhalang (–1 ≤ v ≤ – 0), kehilangan sinyal karena pantulan dapat hingga 6 dB.
3. Untuk jalur propagasi di mana obstruksi jenis *knife-edge* tunggal menyumbat LOS langsung ditambah lebih dari 60% dari radius *fresnel zone* pertama di atas LOS (0 ≤ v ≤ 1), kehilangan sinyal karena difraksi dapat mencapai 14 dB.
4. Untuk jalur propagasi di mana obstruksi jenis *knife-edge* tunggal menyumbat LOS langsung ditambah *fresnel zone* pertama yang lengkap (1 ≤ v ≤ 2.4), kehilangan sinyal karena difraksi dapat lebih besar dari 20 dB.

Pengetahuan dan subbab tentang *fresnel zone* diperlukan karena pada proses pointing antenna ini berkaitan erat dengan Fresnel zone. Pointing antenna dikatakan baik apabila Fresnel zone antara 2 titik antenna tersebut tidak terganggu atau terhalang objek lain. Tentunya dengan adanya pembahasan Fresnel zone ini dapat menjadi landasan pemahaman dari apa itu Fresnel zone dan kegunaannya dalam melakukan pointing antenna.

2.3 *Android*

Android adalah sistem operasi mobile yang didasarkan pada versi dimodifikasi dari Linux. Android pada awalnya dikembangkan oleh startup dengan nama yang sama, Android, Inc. Pada tahun 2005, sebagai bagian dari strategi untuk memasuki ruang mobile, Google membeli Android dan mengambil alih pekerjaan pembangunan (beserta tim pengembangan).

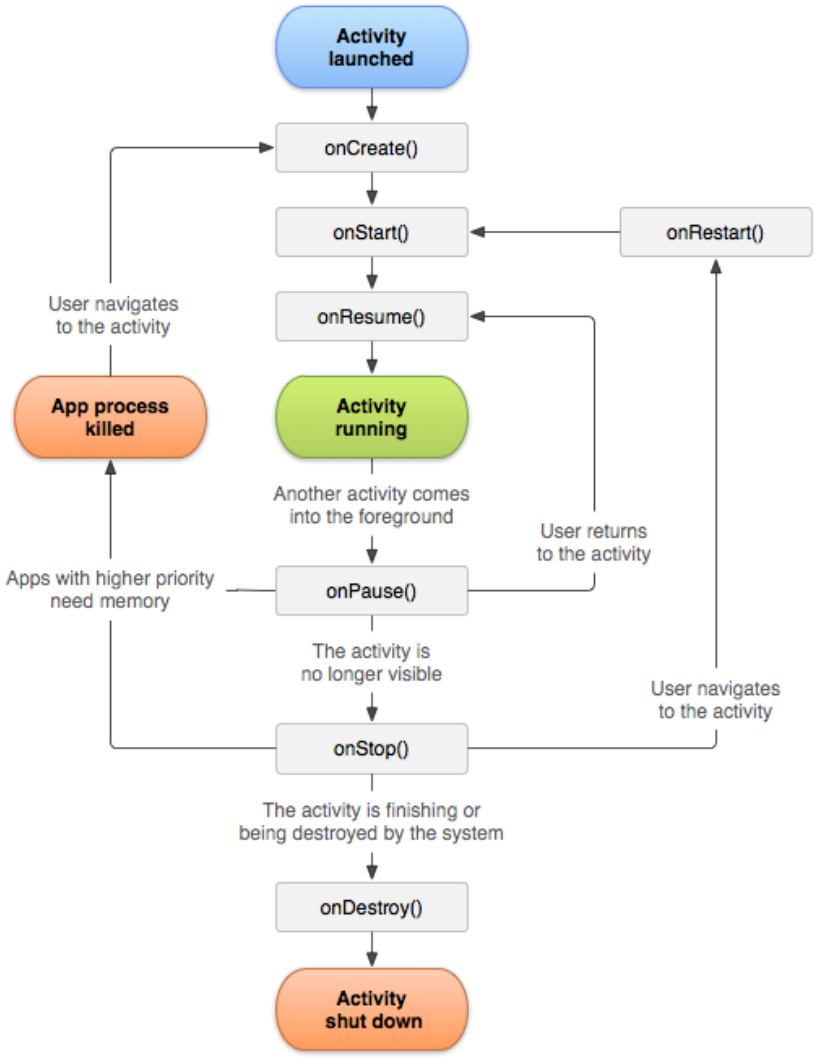
Google ingin *Android* menjadi terbuka dan bebas. Oleh karena itu, sebagian besar kode Android dirilis di bawah open source Apache License, yang berarti bahwa siapapun yang ingin menggunakan *android* dapat melakukannya dengan mengunduh penuh kode sumber *android*. Selain itu, vendor (biasanya produsen hardware) dapat menambahkan ekstensi milik mereka sendiri untuk Android dan menyesuaikan Android untuk membedakan produk mereka dari orang lain. Model pembangunan yang sederhana ini membuat Android sangat menarik dan telah mengusik minat banyak vendor. Keuntungan utama dari mengadopsi *android* adalah menawarkan pendekatan terpadu untuk pengembangan aplikasi. Pengembang hanya perlu mengembangkan untuk *android*, dan aplikasi mereka dapat dijalankan pada berbagai perangkat yang berbeda, asalkan perangkat yang didukung menggunakan *Android* [4].

Pengetahuan dan subbab tentang android diperlukan karena pada penelitian ini aplikasi yang dibangun berbasiskan *platform* android. Tentunya dengan adanya pembahasan ini dapat menjadi landasan pemahaman dari apa itu android dan sejarahnya. Pembahasan tentang android ini juga akan diperdalam lagi dengan *life cycle* atau siklus hidup dari android.

2.3.1 *Android Life Cycle*

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana aplikasi yang dijalankan pada fungsi main(), sistem android menjalankan kode dalam method Activity dengan menerapkan metode callback tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap

aplikasi yang berjalan dalam sistem operasi android memiliki siklus hidup yang berbeda dengan aplikasi desktop atau web. Hal ini dikarenakan aplikasi mobile memiliki tingkat interupsi proses yang lumayan tinggi seperti ketika handling panggilan masuk aplikasi diharuskan menghentikan proses sementara. Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna [4].



Gambar 2.6 ***Android Life Cycle***

Terdapat beberapa state dalam siklus hidup android yang terjadi seperti diilustrasikan pada Gambar 2.6 *Android Life Cycle*, akan tetapi hanya beberapa dari state tersebut yang menjadi statis diantaranya [4]:

1. *Resumed*

   Terjadi ketika aplikasi berjalan setelah state paused. State ini akan menjalankan perintah program yang ditulis pada method onResume().

2. *Paused*

   Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apapun ataupun menampilkan apapun dalam state ini.

3. *Stopped*

   Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan service dibackground. State lain seperti Created dan Started bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode life cycle callback berikutnya. Artinya, setelah sistem OnCreate() dipanggil, dengan cepat sistem akan memanggil method OnStart(), kemudian diikuti oleh onResume().

Pengetahuan dan subbab tentang android *life cycle* diperlukan karena dalam membangun aplikasi berbasis android peneliti harus mengetahui siklus hidup dari aplikasi android tersebut. Siklus hidup dari aplikasi android berbeda dengan siklus hidup aplikasi desktop karena aplikasi android memiliki tingkat interupsi yang tinggi. Tentunya dengan adanya pembahasan ini dapat menjadi landasan pemahaman dari bagaimana siklus hidup aplikasi android dan apa saja komponennya.

2.3.2 GPS

Global Positioning System (GPS) menggunakan sistem satelit yang mengorbit planet untuk membantu penerima (handset Android dalam kasus ini) menentukan lokasi saat ini. Istilah GPS mengacu pada keseluruhan sistem GPS, yang terdiri dari satelit, penerima, dan stasiun kontrol yang memantau dan menyesuaikannya. Penerima ada pada telepon tidak berguna tanpa kinerja dari keseluruhan sistem GPS. [5]

Secara umum, penerima GPS menggunakan informasi dari satelit GPS yang mengorbit bumi untuk menghitung lokasinya saat ini. Sistem GPS berisi 27 satelit yang terus mengorbit bumi, mengirimkan informasi ke calon penerima. Setiap satelit mengikuti jalur yang ditentukan, memastikan bahwa setidaknya empat satelit "*visible*" dari titik mana pun di bumi pada waktu tertentu. Mampu memiliki "*Line of Sight*" ke setidaknya empat satelit diperlukan untuk menentukan lokasi menggunakan GPS. [5]

Untuk menghitung lokasinya, penerima GPS harus dapat menentukan jaraknya dari beberapa satelit. Ini dilakukan dengan menggunakan data ephemeris. Termasuk dalam data yang ditransmisikan dari satelit, bersama dengan data posisi, adalah waktu di mana transmisi dimulai. Setiap satelit GPS berisi mekanisme ketepatan waktu yang sangat akurat yang memungkinkan satelit untuk menjaga waktu sinkron dengan sisa satelit. Untuk menghasilkan perhitungan lokasi yang akurat, satelit GPS dan penerima GPS harus disinkronkan dengan jam mereka. Bahkan perbedaan sedikit waktu dapat menyebabkan kesalahan besar saat menghitung lokasi.

Pengetahuan dan subbab tentang sensor android GPS diperlukan karena pada aplikasi android yang dibangun akan banyak memanfaatkan sensor GPS. Sensor GPS ini nantinya akan digunakan untuk menentukan *current location* dari pengguna dan proses pointing antenna pada aplikasi yang dibangun. Tentunya dengan adanya pembahasan ini dapat menjadi landasan pemahaman dari apa itu sensor GPS pada android dan bagaimana proses penggunaannya.

2.3.3 *Magnetic Field*

Sensor *magnetic field* di *smartphone* dapat beroperasi dengan berbagai metode yang berbeda. Hal ini tergantung dari pabrikan dan arsitektur smartphone. Biasanya sensor ini menggunakan efek Hall, suatu material dengan magneto-resistive atau gaya Lorentz. Sensor efek hall saat ini memiliki pangsa magnetometer terbesar dan mereka bekerja dengan cara yang paling efisien, hanya perlu melewatkan arus melalui kawat. Kemudian komponen medan magnet yang tegak lurus terhadap kawat tersebut menyebabkan elektron memiliki kerapatan yang lebih tinggi pada satu sisi daripada yang lain dari kawat. Hal ini menyebabkan tegangan melintasi lebar kawat yang sebanding dengan medan magnet. Sensor gaya Lorentz hampir serupa, namun mengukur defleksi mekanis dari kawat daripada voltase melintasi lebar kawat. Terlepas dari mekanisme fisik, sensor *magnetic field* akan melaporkan medan magnet di x, y, dan z (dengan memiliki tiga sensor terpisah, satu sejajar sepanjang setiap sumbu). [5]

Salah satu contoh penggunaan dari sensor magnetic field ini adalah *digital compass. Smartphone* dapat berfungsi sebagai kompas dengan sensor ini. Dan dalam implementasinya dengan aplikasi yang akan dibangun, sensor *magnetic field* akan digunakan sebagai kompas yang akan menunjuk langsung pointing antenna terhadap tower lawan. Dengan begitu proses pointing antenna akan lebih akurat dan sesuai dengan perhitungan sudut azimuth.

Pengetahuan dan subbab tentang sensor android *magnetic field* diperlukan karena pada aplikasi android yang akan dibangun memanfaatkan sensor *magnetic field* untuk fitur kompasnya. Sensor *magnetic field* akan membaca arah utara dan selatan seperti kompas, kemudian dikombinasikan dengan sudut azimuth dari titik tower yang dituju dan nantinya akan menghasilkan penunjuk arah ke titik lokasi tujuan. Tentunya dengan adanya pembahasan ini dapat menjadi landasan pemahaman dari apa itu sensor *magnetic field* pada android dan bagaimana proses penggunaannya.

2.4 *Object Oriented Programming*

Object Oriented Programming (OOP) atau yang dikenal dengan pemrograman berorientasi objek adalah metode implementasi di mana program disusun sebagai kumpulan objek yang kooperatif, yang masing-masing mewakili sebuah instance dari beberapa kelas, dan kelas mana yang menjadi anggota hirarki kelas yang disatukan melalui hubungan warisan (Grady Booch, 1998:35). Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. [6]

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri. Pemrograman orientasi-objek menekankan konsep berikut: [6]

1. *Class* (Kelas)

   Kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh *'Class of laptop*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam jenis/karakteristik dari laptop. Sebuah Class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah *Class* 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 seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

2. *Property* (Properti)

   Property (atau disebut juga dengan atribut) adalah data yang terdapat dalam sebuah *class*. Melanjutkan analogi tentang laptop, *property* dari laptop bisa berupa merk, warna, jenis processor, ukuran layar, dan lain-lain. *Property* ini sebenarnya hanyalah variabel yang terletak di dalam class. Seluruh aturan dan tipe data yang biasa diinput kedalam variabel, bisa juga diinput kedalam property. Aturan tata cara penamaan property sama dengan aturan penamaan variabel.

3. *Method* (Metode)

   Method adalah tindakan yang bisa dilakukan didalam *class*. Jika menggunakan analogi class laptop kita, maka contoh *method* adalah: menghidupkan laptop, mematikan laptop, mengganti cover laptop, dan berbagai tindakan lain. *Method* pada dasarnya adalah *function* yang berada di dalam *class*. Seluruh fungsi dan sifat *function* bisa diterapkan kedalam *method*, seperti argumen/parameter, mengembalikan nilai (dengan *keyword return*), dan lain-lain.

4. *Object* (Objek)

   Object atau Objek adalah hasil cetak dari *class*, atau hasil ‘konkrit’ dari *class*. Jika menggunakan analogi *class* laptop, maka objek dari *class* laptop bisa berupa: laptop_andi, laptop_anto, dan lain-lain. Objek dari *class* laptop akan memiliki seluruh ciri-ciri laptop, yaitu *property* dan method-nya. Objek membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program berorientasi objek.

5. *Abstract* (Abstraksi)

   Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat 23 melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

6. *Encapsulation* (Enkapsulasi)

   Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

7. *Polymorphism* (Polimorfisme)

   Melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

8. *Inheritance* (Inheritas)

   Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).

Sebagaimana tercantum pada batasan masalah mengenai desain model aplikasi yang akan digunakan adalah OOP *(Object Oriented Programming)*, maka subbab tentang pembahasan OOP perlu digunakan pada landasan teori ini. Tentunya dengan adanya pembahasan OOP ini dapat lebih meningkatkan pemahaman definisi dan konsep dasar dasar dari OOP. Dan dengan adanya pembahasan ini juga pembangunan aplikasi antenna pointing dapat diterapkan dengan cara *Object Oriented Programming*.

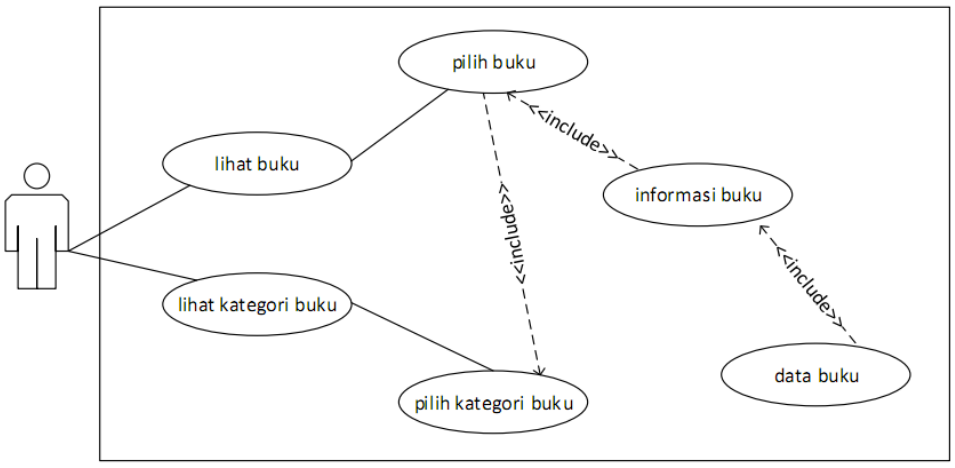
2.5 *Unified Modelling Language*

UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem. UML (*Unified Modeling Language*) merupkan himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. [7]. UML juga dapat diartikan metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP.

UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM. UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. [7]. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan system. UML menyediakan beberapa macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu: [7]

1. *Usecase Diagram*

   Use case diagram adalah gambaran graphical dari beberapa atau semua Aktor, *usecase*, dan interaksi diantara komponen-komponen tersebut yang memperkenalkan suatu sistem yang akan dibangun. *Usecase diagram* menjelaskan manfaat suatu sistem jika dilihat menurut pandangan orang yang berada di luar sistem. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar.

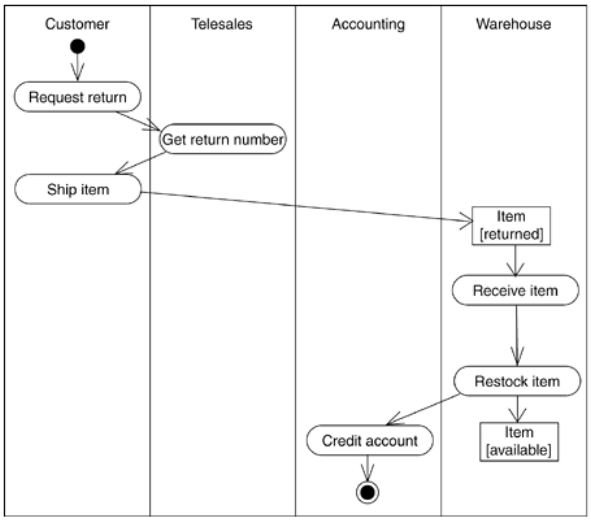


Gambar 2.7 Contoh *Use Case Diagram*

Usecase diagram dapat digunakan selama proses analisis untuk menangkap *requirement* sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *usecase diagram* berperan untuk menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa use-case diagram. Kebutuhan atau *requirements system* adalah fungsionalitas apa yang harus disediakan oleh sistem kemudian didokumentasikan pada model *usecase* yang menggambarkan fungsi sistem yang diharapkan (*use case*), dan yang mengelilinginya (*Aktor*), serta hubungan antara Aktor dengan use case (*use case diagram*) itu sendiri.

2. *Activity Diagram*

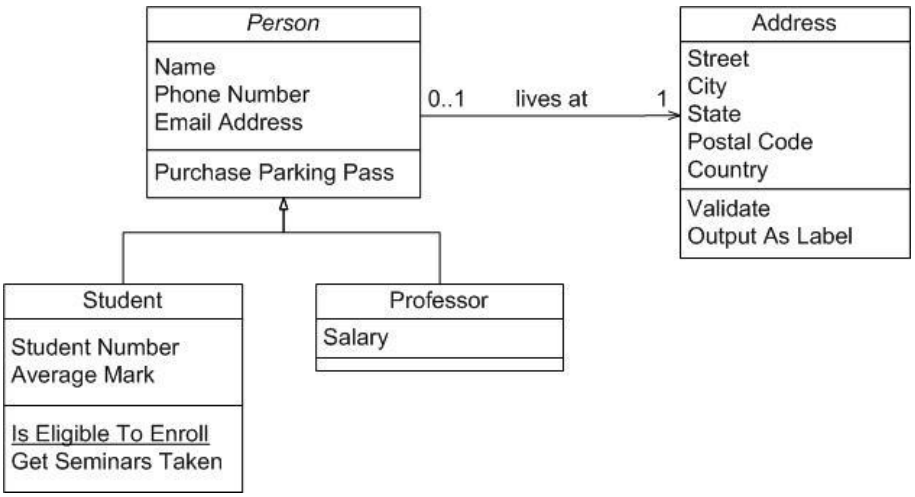
Activity Diagram memiliki pengertian yaitu lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis. Memiliki struktur diagram yang mirip *flowchart* atau data *flow diagram* pada perancangan terstruktur. Memiliki pula manfaat yaitu apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. *Activity diagram* dibuat berdasarkan sebuah atau beberapa *use case*.



Gambar 2.8 Contoh *Activity Diagram*

3. *Class Diagram*

   Class diagram adalah sebuah *Class* yang menggambarkan struktur dan penjelasan *Class*, paket, dan objek serta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class diagram* juga menjelaskan hubungan antar *Class* dalam sebuah sistem yang sedang dibuat dan bagaimana caranya agar mereka saling berkolaborasi untuk mencapai sebuah tujuan.

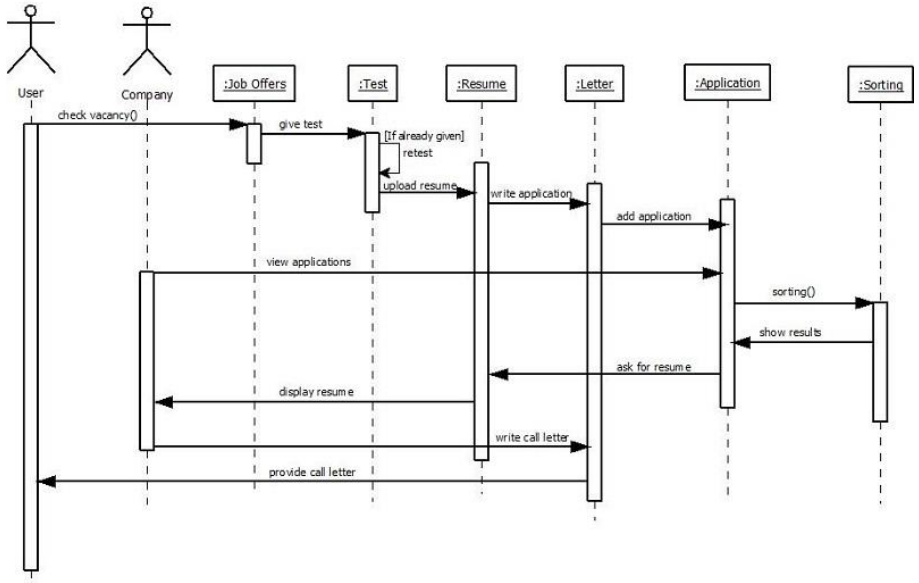


Gambar 2.9 Contoh *Class Diagram*

4. *Sequence Diagram*

   Sequence Diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence Diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek- objek yang terkait). *Sequence Diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang *men-trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem

untuk mencapai tujuan dari use case: interaksi yang terjadi antar class, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi



Gambar 2.10 Contoh *Sequence Diagram*

Dikarenakan desain aplikasi antenna pointing yang akan dibangun berorientasi objek *(Object Oriented Programming)* maka subbab tentang pembahasan UML (*Unified Modeling Language*) perlu digunakan pada landasan teori ini. Tentunya dengan adanya pembahasan UML ini dapat lebih meningkatkan pemahaman definisi dan bentuk-bentuk diagram pemodelan pada UML. Dan dengan adanya pembahasan tentang UML ini dapat menjadi dasar atau landasan dalam membuat perancangan aplikasi antenna pointing.

2.6 *Application Programming Interface* (API)

Application Programming Interface, atau API, adalah rangkaian simbol yang diekspor dan tersedia bagi pengguna *library* untuk menulis aplikasinya. Desain API ini bisa dibilang bagian paling penting dari perancangan *library*, karena dapat mempengaruhi desain aplikasi yang dibangun di atas mereka. [8]

API memungkinkan programmer untuk menggunakan fungsi standar untuk berinteraksi dengan system operasi. API atau Application Programming Interface juga merupakan suatu dokumentasi yang terdiri dari antar muka, fungsi, kelas, struktur untuk membangun sebuah perangkat lunak.

Dengan adanya API, maka memudahkan seorang programmer untuk membongkar suatu software untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya. Suatu rutin standar yang memungkinkan developer menggunakan system function. Proses ini dikelola melalui operating system. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya untuk saling berinteraksi. Keuntungan dengan menggunakan API adalah sebagai berikut [8]:

1. Portabilitas

   Developer yang menggunakan API dapat menjalankan programnya dalam sistem operasi mana saja asalkan sudah terinstal API tersebut.
2. Lebih Mudah Dimengerti

   API menggunakan bahasa yang lebih terstruktur dan mudah dimengerti daripada bahasa system call. Hal ini sangat penting dalam hal editing dan pengembangan.

Dikarenakan aplikasi antenna pointing yang akan dibangun menggunakan berbagai macam API maka subbab tentang pembahasan API perlu digunakan pada landasan teori ini. Tentunya dengan adanya pembahasan API ini dapat lebih meningkatkan pemahaman definisi dan keuntungan menggunakan API. Dan dengan adanya pembahasan tentang API ini dapat menjadi dasar atau landasan dalam membuat perancangan aplikasi antenna pointing. Untuk detail dari API yang digunakan dalam membangun aplikasi antenna pointing ini adalah sebagai berikut:

2.6.1 OpenCelliD API

OpenCelliD adalah proyek komunitas kolaborasi terbesar di dunia yang mengumpulkan posisi GPS dari cell tower, digunakan secara gratis, untuk banyak tujuan komersial dan pribadi. Lebih dari 75.000 kontributor telah terdaftar dan menyumbangkan jutaan pengukuran baru setiap hari ke database OpenCelliD. OpenCelliD bertujuan untuk mendokumentasikan semua cell tower dan WiFi AP di seluruh dunia berikut dengan lokasinya. *Database* cell tower ini dibuat oleh komunitas dan tersedia gratis untuk diunduh atau digunakan melalui API. [9]

Terdapat banyak keuntungan jika kita menggunakan API ini untuk mencari lokasi cell tower. Berikut adalah beberapa keuntungan dari penggunaannya [9]:

a. Peningkatan signifikan untuk *Time To First Fix* (TTFF). GPS dapat memakan waktu cukup lama untuk mengirimkan posisi GPS pertamanya setelah dinyalakan. Sementara itu, data OpenCellID tersedia secara instan.
b. Peningkatan dramatis dalam masa pakai baterai karena berkurangnya penggunaan GPS. Penggunaan GPS dapat menguras baterai secara signifikan, bahkan di telepon yang paling baru sekalipun. Tidak seperti GPS, pengaruh OpenCellID pada konsumsi daya aplikasi Anda hampir tidak terdeteksi. Untuk banyak kasus penggunaan, posisi yang dikirimkan oleh OpenCellID adalah pengganti posisi GPS yang sempurna, hanya untuk disempurnakan dengan data GPS dalam kasus yang jarang terjadi.
c. Pelokalan yang efektif, bahkan di dalam gedung. Posisi telepon saat ini dapat ditentukan bahkan di area di mana GPS tidak tersedia (misalnya di dalam gedung dan di bawah tanah).

Pengetahuan dan subbab tentang API OpenCelliD diperlukan karena API ini dibutuhkan untuk membangun aplikasi antenna pointer. Dengan adanya API OpenCelliD pengguna dapat mengetahui lokasi cell tower terdekat untuk melakukan kegiatan pointing antenna. Tentunya dari pembahasan API OpenCelliD ini dapat menjadi landasan pemahaman dari definisi dan kegunaan API OpenCelliD.

2.6.2 Google Maps API

Google Maps adalah sebuah jasa peta global virtual gratis dan *online* yang disediakan oleh perusahaan Google. Google Maps menawarkan peta yang dapat diseret dan gambar satelit untuk seluruh dunia. Google Maps juga menawarkan pencarian suatu tempat dan rute perjalanan. Google Maps API adalah sebuah layanan yang diberikan oleh Google kepada para pengguna untuk memanfaatkan Google Map dalam mengembangkan aplikasi. Google Maps API menyediakan beberapa fitur untuk memanipulasi peta, dan menambah konten melalui berbagai jenis services yang dimiliki, serta mengijinkan kepada pengguna untuk membangun aplikasi enterprise di dalam websitenya.

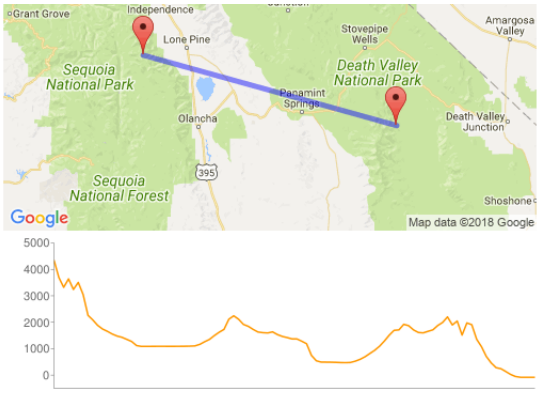
Cara kerja dari google maps mungkin terlihat seperti suatu sistem ajaib yang sangat kompleks dan rumit, namun sebenarnya konten dari Google Maps hanyalah HTML, CSS, dan JavaScript yang bekerja bersama. Setiap petak pada maps merupakan gambar yang dimuat melalui *ajax call* dan kemudian dimasukkan ke dalam *tag* di halaman HTML. Saat melakukan navigasi peta, API mengirim informasi dari koordinat baru dan tingkat zoom peta dalam *ajax call* yang mengembalikan gambar baru. Sedangkan API itu sendiri pada dasarnya terdiri dari file JavaScript yang berisi *class* dengan metode dan properti yang dapat digunakan untuk menginformasikan bagaimana maps dimuat. [10]

Dikarenakan aplikasi antenna pointing yang dibangun akan banyak memanfaatkan layanan Google Maps maka subbab tentang pembahasan Google Maps perlu digunakan pada landasan teori ini. Layanan Google Maps ini akan digunakan sebagai pemetaan lokasi tower dan penentuan titik-titik pointing antenna. Tentunya dengan adanya pembahasan Google Maps ini dapat lebih meningkatkan pemahaman apa itu Google Maps dan bagaimana cara kerjanya.

2.6.3 Google Elevation API

Google Maps Elevation API menyediakan data elevasi untuk semua lokasi di permukaan bumi, termasuk lokasi kedalaman di dasar lautan (yang mengembalikan nilai negatif). Dan untuk kasus tertentu di mana Google tidak memiliki pengukuran elevasi tepat di lokasi tepat yang minta, maka layanan Google Eleveation akan melakukan interpolasi dan mengembalikan nilai rata-rata menggunakan empat lokasi terdekat, nilai elevasi ini dinyatakan relatif terhadap *Local Mean Sea Level* (LMSL). Dengan Google Maps Elevation API, *developer* dapat mengembangkan aplikasi *hiking* atau aplikasi survei resolusi rendah. [11]

Google Maps Elevation API dapat diakses melalui antarmuka HTTP. Pengguna Google Maps JavaScript API juga dapat mengakses API ini secara langsung. Permintaan Google Maps Elevation API ini dibuat dalam bentuk URL. API mengembalikan data elevasi untuk lokasi di bumi. Dan untuk penggunaanya dapat melakukan *request* satu set dari suatu lokasi atau lebih, atau bahkan dapat melakukan *request* dari path antara dua titik pada maps. [11]



Gambar 2.11 Contoh penggunaan Google Elevation API

.

Pengetahuan dan subbab tentang Google Elevation API diperlukan karena API ini dibutuhkan untuk perhitungan *fresnel zone*. Dengan adanya Google Elevation API pengguna dapat mengetahui elevasi dan jalur *fresnel zone* dari dua titik pointing atnenna. Tentunya dari pembahasan Google Elevation API ini dapat menjadi landasan pemahaman dari definisi dan kegunaan Google Elevation API.

2.7 JSON

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-404 Edisi ke 2 – Desember 2017. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. [12]

Dalam penggunaannya, *value* atau sktruktur data dari JSON dapat berbentuk sebagai berikut:

1. *Object*

   Objek adalah sepasang nilai yang tidak terurutkan. Objek dimulai dengan kurung kurawal buka ‘{‘ dan diakhiri dengan kurung kurawal tutup ‘}’. Setiap nama diikuti dengan titik dua ‘:’ dan setiap pasangan nilai dipisahkan oleh koma ‘,’. Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON.

2. *Array*

   Array atau larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan kurung kotak buka ‘[‘ dan diakhiri dengan kurung kotak tutup ‘]’. Setiap nilai dipisahkan oleh koma ‘,’. Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat.

3. *String*

   String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Pada string dapat digunakan backslash escapes ‘\’ untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.

4. *Number*

   Angka adalah urutan digit desimal tanpa nol yang berlebihan. Angka dapat memiliki tanda minus (U + 002D) atau memiliki bagian pecahan yang diawali oleh titik desimal (U + 002E). Angka juga dapat memiliki eksponen, diawali oleh e (U + 0065) atau E (U + 0045) dan opsional + (U + 002B) atau - (U + 002D). Angka-angka adalah poin kode U + 0030 melalui U + 0039.

5. *Boolean*

   Boolean adalah suatu tipe data yang hanya mempunyai dua nilai yaitu *true* (benar) atau *false* (salah).

Dikarenakan aplikasi antenna pointing yang menggunakan API dan pertukaran datanya menggunakan JSON maka subbab tentang pembahasan JSON perlu digunakan pada landasan teori ini. JSON berfungsi sebagai format pertukaran data antara aplikasi android antenna pointer dengan API. Tentunya dengan adanya pembahasan JSON ini dapat lebih meningkatkan pemahaman apa itu JSON dan bentuk-bentuk struktur datanya.

2.8 Java

Java adalah bahasa pemrograman serbaguna. Java dapat digunakan untuk membuat suatu program sebagaimana membuatnya dengan bahasa seperti Pascal atau C++. Yang lebih menarik, Java juga mendukung sumber daya internet yang saat ini populer, yaitu *World Wide Web* atau yang sering disebut web. Java juga mendukung aplikasi klient/server, baik dalam jaringan local maupun jaringan berskala luas. Java dikembangkan oleh Sun Microsystems pada Agustus 1991, dengan nama semula Oak. Konon Oak adalah pohon semacam Jati yang terlihat dari jendela tempat pembuatnya, James Gosling, bekerja. Ada yang mengatakan bahwa Oak adalah singkatan dari "Object Application Kernel", tetapi ada yang menyatakan hal itu muncul setelah nama Oak diberikan. Pada Januari 1995, karena nama Oak dianggap kurang komersial, maka diganti menjadi Java. [13]

Program java bersifat tidak bergantung pada platform, artinya, Java dapat dijalankan pada sembarang komputer dan bahkan pada sembarang sistem operasi. Ketidak bergantungan pada platform sering dinyatakan dengan istilah portabilitas. Yang menarik, tingkat portabilitas java tidak hanya sebatas pada program sumber source code), melainkan juga pada tingkat kode biner yang disebut bytecode. Dengan demikian bila telah mengkompilasi program java pada komputer berbasis operasi Windows, dapat juga menjalankan hasil kompilasi pada Macintosh secara langsung, tanpa perlu mengkompilasi ulang.

Pengetahuan dan subbab tentang Java diperlukan karena Bahasa pemrogramman yang digaunakan dalam membangun aplikasi antenna pointer adalah Java. Bahasa pemrogramman dengan menggunakan Java cocok diimplementasikan pada aplikasi yang dibangun diatas *platform* Android. Tentunya dari pembahasan Java ini dapat menjadi landasan pemahaman dari apa itu Java *programming language* dan sejarahnya.

2.9 MySQL

MySQL merupakan server basis data dimana pemprosesan data terjadi di *server*, dan *client* hanya mengirimkan data serta meminta data. Oleh karena pemprosesan terjadi di server sehingga pengaksesan data tidak terbatas (Solihin, 2010:10). MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang cepat dan mudah digunakan yang digunakan. Kecepatan adalah fokus utama pengembang MySQL sejak awal. Demi kecepatan, pengembang MySQL membuat keputusan untuk menawarkan fitur yang lebih sedikit daripada pesaing utama mereka (seperti *Oracle* dan *Sybase*). Namun, meskipun MySQL kurang berfitur lengkap daripada pesaing komersialnya, namun fitur ini hampir semua fitur yang dibutuhkan oleh pengembang database. Lebih mudah untuk menginstal dan menggunakan daripada pesaing komersialnya, dan selisih harga sangat kuat dengan dukungan MySQL. [14]

MySQL termasuk dalam kategori manajemen basis data yaitu basis data yang terstruktur dalam pengolahan dan penampilan data. Ada beberapa alasan

mengapa MySQL menjadi program yang database yang sangat popular dan digunakan oleh banyak orang. Kelebihan dari MySQL adalah sebagai berikut: [14]

1. MySQL merupakan basis data yang memiliki kecepatan yang tinggi dalam melakukan pemprosesan data, dapat diandalkan dan mudah digunakan serta mudah dipelajari.
2. MySQL mendukung banyak pemrograman seperti C, C++, Perl, Phyton, Java, Visual Basic dan PHP.
3. MySQL dapat menangani basis data dengan skala yang sangat bese dengan jumlah record mencapai lebih dari 50 juta.
4. MySQL merupakan software basis data yang bersifat bebas atau gratis tanpa bayaran, jadi kita tidak perlu susahsusah mengeluarkan uang untuk sekedar membayar lisensi.

Subbab pembahasan mengenai MySQL diperlukan karena pada pembangunan aplikasi antenna pointer menggunakan MySQL sebagai tempat penyimpanan data pada API. Lebih tepatnya untuk menyimpan data histori dari pointing antenna yang sudah dilakukan oleh pengguna. Tentunya dengan adanya pembahasan ini dapat lebih meningkatkan pemahaman definisi dan kelebihan dari MySQL.

2.10 Android Studio

Android Studio adalah sebuah IDE untuk Android Development yang diperkenalkan google pada acara Google I/O 2013. Android Studio merupakan pengembangan dari Eclipse IDE, dan dibuat berdasarkan IDE Java populer, yaitu IntelliJ IDEA. [15] Android Studio merupakan IDE resmi untuk pengembangan aplikasi Android. Sebagai pengembangan dari Eclipse, Android Studio mempunyai banyak fitur-fitur baru dibandingkan dengan Eclipse IDE. Berbeda dengan Eclipse yang menggunakan Ant, Android Studio menggunakan Gradle sebagai build environment. Android studio juga memiliki kelebihan fitur-fitur yang lebih unggul sebagai berikut:

1. Menggunakan *Gradle-based build system* yang fleksibel.

2. Bisa *build* multiple APK.
3. *Template* support untuk Google Services dan berbagai macam tipe perangkat.
4. *Layout* editor yang lebih bagus.
5. *Built-in* support untuk Google Cloud Platform, sehingga mudah untuk integrasi dengan Google Cloud Messaging dan App Engine.
6. Import library langsung dari Maven repository.

Android Studio adalah kolaborasi antara JetBrains dan Google. Android Studio dibangun di atas IntelliJ JetBrain, dan fungsinya merupakan superset dari IntelliJ. Kebanyakan yang dapat Anda lakukan dengan IntelliJ, Anda juga dapat melakukannya di Android Studio. Android Studio bersifat revolusioner karena menyederhanakan proses pengembangan Android dan membuat pengembangan Android jauh lebih mudah diakses daripada sebelumnya. Android Studio sekarang adalah IDE resmi untuk Android. [15]

Dikarenakan aplikasi antenna pointing yang dibangun menggunakan Android Studio sebagai IDE dan Android *development* maka subbab tentang pembahasan Android Studio perlu digunakan pada landasan teori ini. Pemilihan Android Studio sebagai IDE untuk *development* ini berdasarkan fitur-fiturnya yang mampu mencukupi kebutuhan penulis dalam membangun aplikasi antenna pointer. Tentunya dengan adanya pembahasan Android Studio ini dapat lebih meningkatkan pemahaman apa itu Android Studio, sejarahnya dan fitur-fiturnya.