BAB II

LANDASAN TEORI

2.1 Tunanetra

Menurut kamus besar bahasa Indonesia, tunanetra diartikan tidak dapat melihat atau buta (KBI, 2012). Sehingga dapat diartikan bahwa tunanetra adalah seseorang yang memiliki hambatan dalam penglihatan atau tidak berfungsinya indera penglihatan, sedangkan *low vision* dapat dikatakan apabila seseorang mengalami kekurangan penglihatan. (SLB Kartini Batam, 2012)

2.1.1 Klasifikasi Tunanetra

Terdapat beberapa klafisikasi tunanetra, yaitu :

1. Berdasarkan waktu terjadinya ketunanetraan.
   a. Tunanetra sebelum dan sejak lahir.
   b. Tunanetra setelah lahir atau pada usia kecil.
   c. Tunanetra pada usia sekolah atau pada masa remaja.
   d. Tunanetra pada usia dewasa.
   e. Tunanetra dalam usia lanjut.
2. Berdasarkan kemampuan daya penglihatan.
   a. Tunanetra ringan.
   b. Tunanetra sedang.
   c. Tunanetra berat.
3. Berdasarkan pemeriksaan klinik.
4. Berdasarkan kelainan-kelainan pada mata.

a. *Myopia* adalah penglihatan jarak dekat, bayangan tidak terfokus dan jatuh di belakang retina.
b. *Hyperopia* adalah penglihatan jarak jauh, bayangan tidak terfokus dan jatuh di depan retina.
c. *Astigmatisme* adalah penyimpangan atau penglihatan kabur yang disebabkan ketidak beresan pada kornea mata.

(SLB Kartini Batam, 2012)

2.1.2 Penyebab Tunanetra

Ada beberapa penyebab tunanetra, antara lain :

1. *Pre-natal*, faktor penyebab ketunanetraan pada masa *pre-natal* sangat erat hubungannya dengan masalah keturunan dan pertumbuhan seorang anak dalam kandungan.
2. *Post-natal*, Faktor penyebab ketunanetraan yang terjadi pada masa *post-natal* dapat terjadi sejak atau setelah bayi lahir, antara lain: kerusakan pada mata atau syaraf mata pada waktu persalinan hamil ibu menderita penyakit *gonorrhoe*, penyakit mata lain yang menyebabkan ketunanetraan, seperti *trachoma* dan akibat kecelakaan.

(SLB Kartini Batam, 2012)

2.2 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis Linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Android menyediakan *platform* yang terbuka bagi para pengembang untuk menciptakan

aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel/*smartphone*. (Nazruddin,2011:1)

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android juga menjadi sistem operasi pada *Tablet* PC. Pesatnya pertumbuhan Android disebabkan oleh platformnya yang sangat lengkap baik dari sistem operasi, aplikasi, dan *tool* pengembangan, *market* aplikasi Android serta dukungan yang sangat tinggi dari komunitas *open source* di dunia.

2.2.1 Keunggulan Android

1. Lengkap (*Complete Platform*): Para desainer dapat melakukan pendekatan yang komprehensif ketika mereka sedang mengembangkan *platform* Android. Android merupakan sistem operasi yang aman dan banyak menyediakan *tools* dalam membangun *software* dan memungkinkan untuk peluang pengembangan aplikasi.
2. Terbuka (*Open Source Platform*): *Platform* Android disediakan melalui lisensi *open source*. Pengembang dapat dengan bebas untuk mengembangkan aplikasi. Android sendiri menggunakan Linux *kernel* 2.6.
3. *Free* (*Free Platform*): Android adalah *platform*/aplikasi yang bebas untuk *developer*. Tidak ada lisensi atau biaya royalti untuk dikembangkan pada *platform* Android. Tidak ada biaya keanggotaan diperlukan. Tidak diperlukan biaya pengujian. Tidak ada kontrak yang diperlukan. Android dapat didistribusikan dan diperdagangkan dalam bentuk apapun.

(Nazruddin,2011:3)

2.2.2 Arsitektur Android

Secara garis besar arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut:

1. *Applications* dan *Widgets*

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

2. *Applications Frameworks*

   Applications Frameworks ini adalah layer dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti *content-providers* yang berupa sms dan panggilan telepon.

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

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

3. *Libraries*

   Libraries ini adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya.

Berjalan diatas *kernel*, *Layer* ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL, serta:

a. *Libraries* media untuk pemutaran media *audio* dan *video.*
b. *Libraries* untuk manajemen tampilan.
c. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
d. *Libraries* SQLite untuk dukungan database.
e. *Libraries* SSL dan *WebKit* terintegrasi dengan *web browser* dan *security*.
f. *Libraries LiveWebcore* mencakup modern *web browser* dengan *engine embedded web view*.
g. *Libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API's.

4. Android *Run Time*

Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya mengggunakan implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian, yaitu:

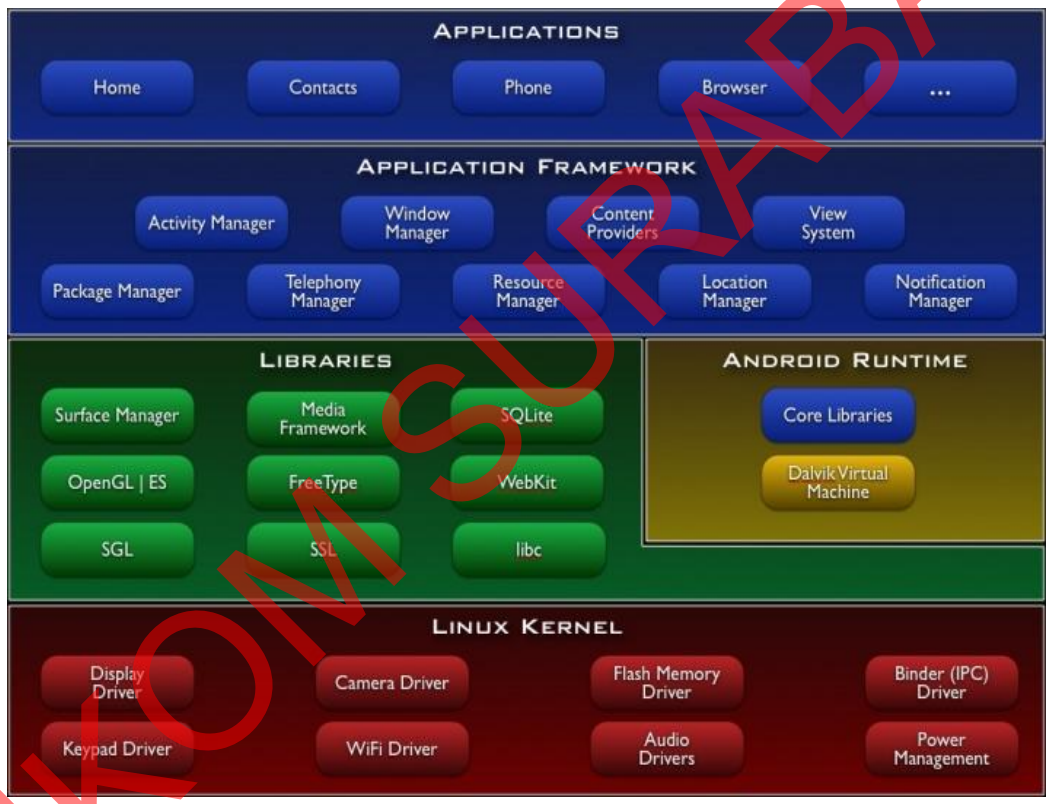
a. *Core Libraries*: Aplikasi Android dibangun dalam bahasa *java*, sementara Dalvik sebagai *virtual* mesinnya bukan *Virtual Machine Java*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menerjemahkan bahasa *Java / C* yang ditangani oleh *Core Libraries*.
b. *Dalvik Virtual Machine*: *Virtual* mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat Linux *kernel* untuk melakukan *threading* dan manajemen tingkat rendah.

5. **Linux *Kernel***

Linux *kernel* adalah layer dimana inti dari *operating system* dari Android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, memori, *resource, drivers,* dan sistem-sistem operasi Android lainnya. Linux *kernel* yang digunakan Android adalah Linux *kernel release 2.6.*

(Nazruddin,2011:6-8)

Berikut merupakan gambar dari arsitektur Android:



Gambar 2.1 Arsitektur Android (Nazruddin,2011:9)

2.2.3 Fundamental Aplikasi Android

Aplikasi Android ditulis dalam bahasa pemrograman *java*. Kode *java* dikompilasi bersama dengan data file *resource* yang dibutuhkan oleh aplikasi, dimana prosesnya di-*package* oleh *tools* yang dinamakan *"apt-tools"* ke dalam

paket Android sehingga menghasilkan file dengan ekstensi apk. File apk itulah yang kita sebut dengan aplikasi dan nantinya dapat di-*install* di perangkat *mobile*.

Ada empat jenis komponen pada aplikasi Android, yaitu:

a. *Activites*

Suatu *activity* akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi Android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut.

b. *Service*

Service tidak memilki GUI, tetapi berjalan secara *background*, sebagai contoh dalam memainkan musik, *service* mungkin memainkan musik atau mengambil data dari jaringan, tetapi setiap *service* harus berada dalam kelas induknya. Misalnya, *media player* sedang memutar lagu dari list yang ada, aplikasi ini akan memiliki dua atau lebih *activity* yang memungkinkan user untuk memilih lagu atau menulis SMS sambil *player* sedang jalan. Untuk menjaga musik tetap dapat dijalankan, *activity player* dapat menjalankan *service*.

c. *Broadcast Receiver*

Broadcast Receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh *broadcast* seperti notifikasi zona waktu berubah, baterai *low*, dll.

Broadcast receiver tidak memiliki UI, tetapi memiliki sebuah *activity* untuk merespon informasi yang mereka terima, atau mungkin menggunakan *Notification Manager* untuk memberitahu kepada pengguna, seperti lampu latar atau getaran perangkat, dan lain sebagainya.

d. *Content Provider*

Content provider membuat kumpulan aplikasi data secara spesifik sehingga bias digunakan oleh aplikasi lain. Data disimpan dalam file sistem seperti database SQLite. *Content provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya ketika kita menggunakan aplikasi yang membutuhkan peta, atau aplikasi yang membutuhkan untuk mengakses data kontak dan navigasi, maka disinilah fungsi *content provider*.

(Nazruddin,2011:9-10)

2.3 Antarmuka atau interface

Antarmuka diperlukan dalam membangun sebuah aplikasi. Dengan adanya antar muka yang *user-friendly*, diharapkan dapat memudahkan para pengguna sebagai user untuk menjalankan sistem tersebut.

Antarmuka juga dikenal dengan nama GUI (*Graphical User Interface*), yaitu program antarmuka yang berbasis grafis, dimana perintah-perintah tidak lagi diketik di *keyboard*, tetapi dengan cara melakukan interaksi secara langsung terhadap apa yang terlihat di layar, yang sebenarnya merupakan suatu abstraksi dari suatu perintah kepada komputer agar komputer mengerjakan apa yang pengguna inginkan.

Terdapat beberapa tipe komunikasi / interaksi antara manusia dengan mesin komputer, yaitu:

1. Dialog berbasis bahasa alami

   Pengguna dapat secara bebas dapat memberikan instruksinya. Dengan kebebasan yang dimiliki pengguna untuk memberikan sembarang instruksi,

komputer harus mampu mengolah bahasa alami. Meskipun demikian, karena bahasa alami sering menimbulkan ambiguitas, maka dialog dengan bahasa alami tidak dapat diimplementasikan secara sempurna.

2. Sistem Menu

   Sistem ini dilakukan dengan memilih pilihan-pilihan yang tersedia pada layar tampilan, atau dengan meng-klik pilihan-pilihan dari menu *pulldown* yang tersedia, maka komputer akan memproses instruksi tersebut.

3. *Form filling dialog*

   Pengguna seolah-olah mengisikan data ke dalam formulis elektronik menggunakan *keyboard*.

4. Dialog berbasis icon

   Tampilan layar menggunakan icon (gambar sederhana yang menunjukkan suatu aktifitas tertentu). Jadi dengan meng-klik gambar tersebut, maka komputer akan mengerjakan perintah dari maksud gambar tersebut.

5. Dialog berbasis jendela

   Tampilan pada layar terdapat jendela, yaitu terdapat bentuk empat persegi panjang dan dibatasi oleh suatu pembatas yang biasanya nampak. Hal ini memungkinkan pengguna untuk melihat banyak jendela yang berisi informasi yang dapat dilihat secara serempak.

6. Manipulasi Langsung

   Pengguna langsung berinteraksi dengan objek yang ada pada layar tampilan dengan mengarahkan *pointer* yang ada di layar, atau menekan tombol-tombol.

7. Interaksi grafis

   Pengguna seolah-olah berdialog dengan grafik yang dibuatnya. Pengguna mempunyai keleluasaan mengubah gambar yang ada pada layar tampilan.

(Insap Santosa, 1994).

2.4 GPS (*Global Positioning System)*

Sistem Kedudukan Sejagat (*Global Positioning System* (GPS)) adalah sistem untuk menentukan posisi di permukaan bumi dengan bantuan sinkronisasi sinyal satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke Bumi. Sinyal ini diterima oleh alat penerima di permukaan, dan digunakan untuk menentukan posisi, kecepatan, arah, dan waktu. Sistem yang serupa dengan GPS antara lain GLONASS Rusia, Galileo Uni Eropa, IRNSS India.

Sistem ini dikembangkan oleh Departemen Pertahanan Amerika Serikat, dengan nama lengkapnya adalah NAVSTAR GPS (kesalahan umum adalah bahwa NAVSTAR adalah sebuah singkatan, ini adalah salah, NAVSTAR adalah nama yang diberikan oleh John Walsh, seorang penentu kebijakan penting dalam program GPS). Kumpulan satelit ini diurus oleh 50th Space Wing Angkatan Udara Amerika Serikat. Biaya perawatan sistem ini sekitar US$750 juta per tahun, termasuk penggantian satelit lama, serta riset dan pengembangan.

GPS *Tracker* atau sering disebut dengan GPS *Tracking* adalah teknologi AVL (*Automated Vehicle Locater*) yang memungkinkan pengguna untuk melacak posisi kendaraan, armada ataupun mobil dalam keadaan *Real-Time*. GPS *Tracking*

memanfaatkan kombinasi teknologi GSM dan GPS untuk menentukan koordinat sebuah obyek, lalu menerjemahkannya dalam bentuk peta digital.

(Parkinson, B.W., 1996)

2.4.1 Arsitektur GPS

Arsitektur GPS terdiri atas 3 segmen, yaitu *space segment*, *control segment*, dan *user segment*. *Space segment* berupa 27 satelit yang terus mengorbit bumi. 24 satelit digunakan untuk operasional, sedangkan 3 satelit sisanya digunakan sebagai cadangan. 24 satelit tersebut dibagi atas kumpulan 4 satelit yang mengorbit pada 6 jalur lintasan. Orbit lintasan ini telah diatur, sehingga setiap titik di bumi ini pasti tercakup dalam LoS (*Line of Sight*) dari setidaknya 6 satelit.

Gambar 2.2 GPS Space Segment

Control segment berupa stasiun di bumi yang memonitor satelit-satelit GPS. Stasiun ini mengontak setiap satelit GPS secara berkala untuk *update*

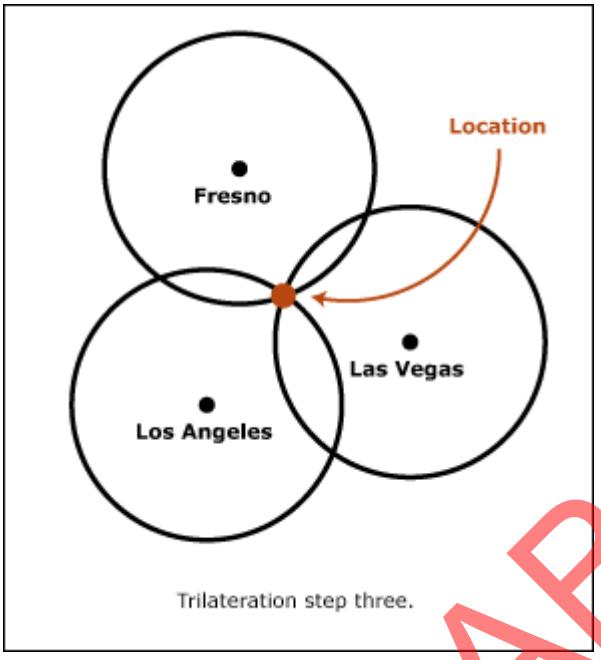
navigasi. *Update* ini berupa sinkronisasi jam atomik satelit dengan satelit lainnya dan mengkoreksi lintasan orbit setiap satelit.

User segment adalah berupa GPS *receiver*. Secara umum, GPS *receiver* terdiri dari sebuah antena yang dapat menangkap frekuensi yang ditransmisikan satelit GPS, sebuah prosesor, dan sebuah jam yang sangat stabil. GPS *receiver* memiliki atribut *channel* yaitu jumlah satelit yang dapat dimonitornya dalam suatu waktu (sekarang umumnya jumlah *channel* berkisar antara 12-20). Kebanyakan GPS *receiver* dapat meneruskan datanya ke perangkat yang lain melalui koneksi serial, USB, atau *Bluetooth* menggunakan protokol NMEA (*National Marine Electronics Association*).

2.4.2 Cara penentuan lokasi pada GPS

Satelit GPS mengorbit bumi dua kali dalam sehari dengan lintasan yang sangat presisi dan mentransmisikan sinyal secara kontinu ke bumi. GPS *receiver* memanfaatkan informasi ini dengan berperan sebagai sebuah alat pengukur yang menghitung jarak antara antena *receiver* dengan berbagai satelit GPS. Kemudian GPS *receiver* mendeduksi posisi melalui posisi trilaterasi dengan mencari perpotongan tiap vektor satelit-satelit tersebut. Jarak antara antena *receiver* dan satelit diukur dengan membandingkan waktu yang terdapat pada sinyal dengan waktu ketika sinyal diterima.

Sebuah GPS *receiver* setidaknya harus dapat menangkap sinyal dari 3 buah satelit untuk menghitung posisi 2 dimensi (*latitude* dan *longitude*) dan pergerakannya. Dengan 4 buah satelit atau lebih, *receiver* dapat menghitung posisi 3 dimensi (*latitude*, *longitude*, dan *altitude*).



Gambar 2.3 Trilaterasi GPS

2.4.3 Akurasi GPS

GPS menyediakan posisi dengan ketepatan akurasi hingga 15 meter, yang berarti jika GPS *receiver* memberikan koordinat terhadap suatu lokasi tertentu, maka boleh diharapkan lokasi sebenarnya berada dalam radius 15 meter dari korrdinat tersebut (El-Rabbany, 2002). Ketepatan GPS bergantung daripada lokasi GPS *receiver*-nya dan halangan terhadap sinyal satelit GPS. Meski secara umum, GPS menawarkan tingkat ketelitian 15 meter, namun akurasi ini dapat ditingkatkan dengan berbagai teknik, seperti *Assisted* GPS (A-GPS), *Differential* GPS (D-GPS), atau *Wide Area Audmentation System* (WAAS).

2.4.4 Sumber Kesalahan Pada GPS

Berikut adalah berbagai faktor yang dapat mengurangi akurasi GPS (El-Rabbany, 2002):

1. *Ionosphere* dan *troposphere error*, dikarenakan sinyal mengalami hambatan ketika melewati atmosfer bumi sehingga menyebabkan delay.
2. *Multipath signyal*, dikarenakan sinyal GPS dapat direfleksikan oleh berbagai benda sebelum mencapai *receiver* sehingga waktu yang dibutuhkan menjadi semakin lama.
3. Jam GPS *receiver error*, dikarenakan jam internal pada GPS *receiver* tidak seakurat jam atomik pada satelit GPS.
4. Orbital atau *ephemeris error*, dikarenakan adanya kesalahan mengenai lokasi satelit.
5. Jumlah satelit yang tertangkap, semakin sedikit satelit yang tertangkap oleh GPS *receiver* semakin rendah akurasinya.
6. *Satellite Geometry error*, dikarenakan posisi satelit tidak ideal dalam perhitungan geometri ketika satelit terlalu berdekatan.

2.5 Google Maps

Google Maps adalah sebuah layanan gratis peta digital dari Google berbasis web yang dapat digunakan dan ditempatkan pada website tertentu dengan menggunakan Google Maps API (Google Inc, 2011).

Google Maps sendiri mempunyai antara lain navigasi peta dengan *dragging mouse*, *zoom in*, dan *zoom out* untuk menunjukkan informasi peta secara detil, member penanda, dan memberi informasi tambahan. *Mode viewing* pada Google Maps berupa "*Map*" (peta topografi dan jalan), "*satellite*" (peta berupa foto satelit dan foto resolusi tinggi dari udara), "*Hybrid*" (peta berupa foto satelit

dan peta jalan berada diatasnya) dan "*Street View*", fasilitas ini secara diperkenalkan oleh Google pada Mei 2007.

2.6 Location Based Service

Location Based Service (LBS), yaitu *service* yang berfungsi untuk mencari dengan teknologi GPS dan Google's *cell-based location*. *Maps* dan layanan berbasis lokasi menggunakan nilai lintang dan bujur untuk menentukan lokasi geografis, namun sebagai user kita membutuhkan alamat atau posisi *realtime* kita. Android menyediakan *geocoder* yang mendukung *forward* dan *reverse geocoding*. Menggunakan *geocoder*, kita dapat mengkonversi nilai lintang bujur menjadi alamat dunia nyata dan sebaliknya.

LBS adalah istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang kita gunakan. Ada dua unsur utama LBS, yaitu :

1. *Location Manager* (API *Maps*)

   Menyediakan *tools/source* untuk LBS, *Application Programming Interface* (API) *Maps* menyediakan fasilitas untuk menampilkan, manipulasi *maps*/peta beserta *feature-feature* lainnya seperti tampilan satelit, *street* (jalan), maupun gabungannya. Paket ini berada pada com.google.Android.maps.

2. *Location Providers* (API *Location*)

   Menyediakan teknolgi pencarian lokasi yang digunakan oleh *device*/perangkat. API *Location* berhubungan dengan data GPS (*Global Positioning System*) dan data lokasi *real-time*. API *Location* berada pada paket Android yaitu dalam paket Android.location. Dengan *Location*

Manager, kita dapat menentukan lokasi kita saat ini, *track* gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahaan.

(Nazruddin, 2011:226)

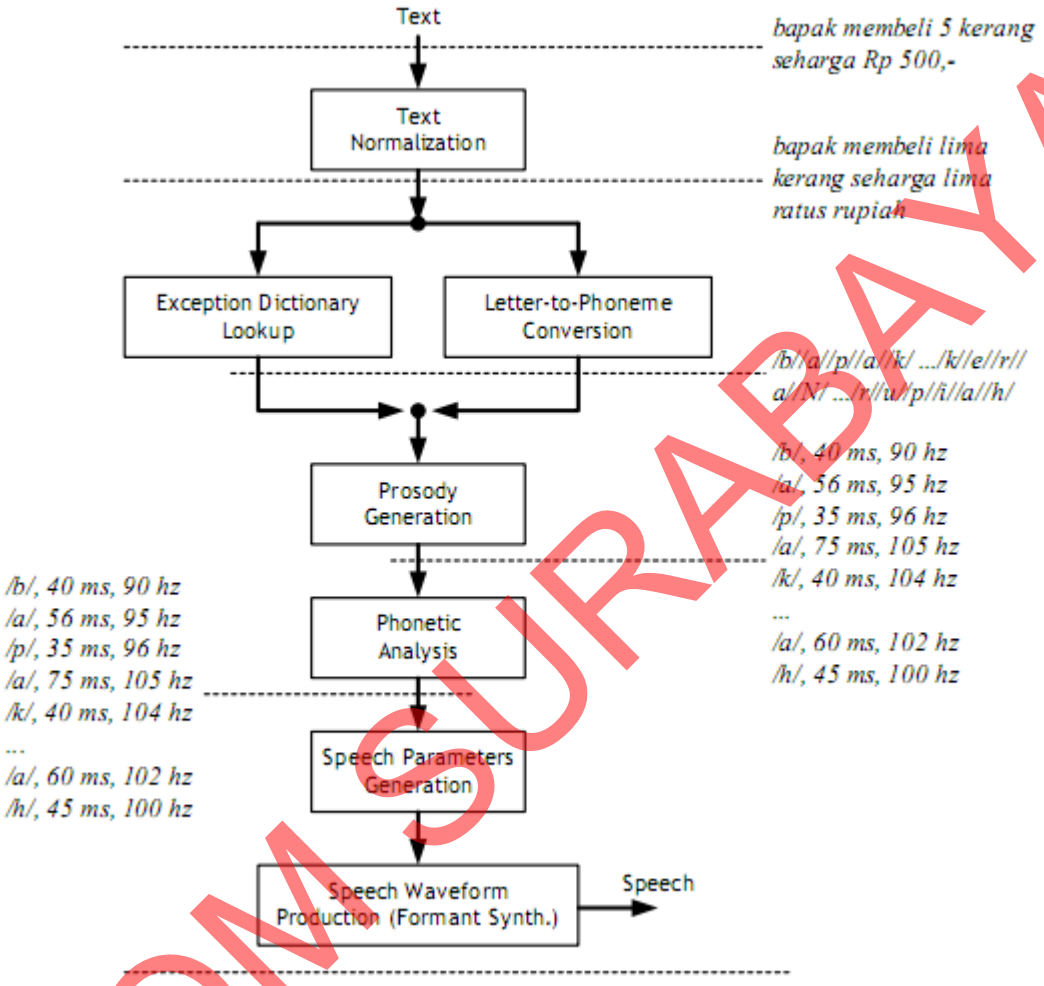
2.7 *Text To Speech* (TTS)

Speech synthesis adalah produksi buatan dari ucapan manusia. Sebuah sistem komputer yang digunakan untuk tujuan ini disebut *speech synthesizer*, dan dapat diimplementasikan dalam sebuah perangkat lunak dan keras. Sebuah sistem *text to speech* mengubah teks bahasa normal menjadi bentuk suara, selain itu membuat representasi *linguistic simbolis fonetis* menjadi bentuk suara.

Suara buatan dapat dibuat dengan menggabungkan pembicaraan rekaman yang disimpan dalam database. Sistem berbeda dalam ukuran unit pidato disimpan, sebuah sistem yang menyimpan telepon atau *diphones* memberikan rentang *output* terbesar, tapi mungkin kurang jelas. Untuk domain penggunaan khusus, penyimpanan seluruh kata-kata atau kalimat memungkinkan untuk *output* berkualitas tinggi, atau *synthesizer* dapat menggabungkan model saluran vokal dan karakteristik suara manusia lainnya untuk membuat benar-benar "sintetik" *output* suara.

Kualitas *synthesizer* suara dinilai oleh kesamaannya dengan suara manusia dan dengan kemampuannya untuk dipahami. Sebuah program *text-to-speech* dipahami memungkinkan orang tunanetra atau cacat membaca untuk mendengarkan karya-karya tulis di komputer rumah. Banyak sistem operasi komputer telah menyertakan alat bicara sejak awal 1980-an."

Berikut gambar dari urutan proses konversi *text to speech*



Gambar 2.4 Urutan Proses *Text To Speech* (Agus Sutomo, 2010)

Sistem *text to speech* pada prinsipnya terdiri atas dua sub sistem, yaitu bagian *text to phoneme converter* dan *phoneme to speech converter*. Bagian *text to phoneme converter* berfungsi untuk mengubah kalimat masukan dalam bahasa tertentu yang berbentuk teks menjadi rangkaian kode-kode bunyi yang biasanya diprensentasikan dengan kode fonem (*phoneme code*), beserta durasi dan *pitch*-nya. Bagian ini bersifat sangat *language dependant*. Maka untuk suatu bahasa baru, bagian ini harus dikembangkan secara lengkap khusus untuk bahasa

tersebut. Bagian *phoneme to speech converter* akan menerima masukan berupa kode-kode fonem, beserta durasi dan *pitch* yang dihasilkan oleh bagian sebelumnya. Berdasarkan kode-kode tersebut maka bagian *phoneme to speech converter* akan menghasilkan bunyi atau sinyal ucapan sesuai dengan kalimat yang ingin diucapkan. (Agus Sutomo, 2010)

TTS pada *platform* Android mendukung sejumlah bahasa, yaitu : Inggris, Perancis, Jerman, Itali, dan Spanyol. Selain itu, tergantung pada sisi dimana user berada. TTS perlu mengetahui bahasa yang digunakan untuk dapat berbicara. TTS API memungkinkan aplikasi untuk melakukan *query platform* untuk ketersediaan *file* bahasa dapat dilakukan *download*. Dengan mengimplemetasikan TextToSpeech.OnInitListener. Maka kita dapat mengatur bahasa yang digunakan dan memeriksa ketersediaannya.

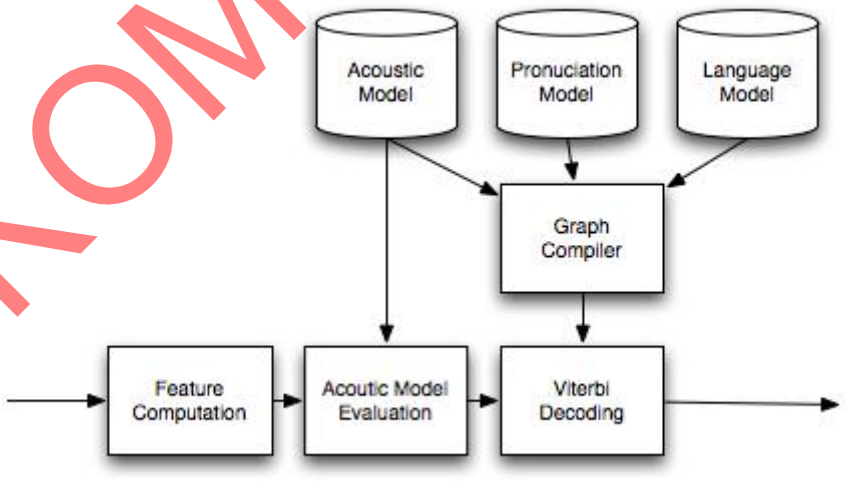
```
public void onInit(int status) {
    // TODO Auto-generated method stub
     // status can be either TextToSpeech.SUCCESS or TextToSpeech.ERROR.
    if (status == TextToSpeech.SUCCESS) {
        // Set preferred language to US english.
        // Note that a language may not be available, and the result will indicate this.
        int result = tts.setLanguage(Locale.US);
        if (result == TextToSpeech.LANG_MISSING_DATA ||
            result == TextToSpeech.LANG_NOT_SUPPORTED) {
           // Language data is missing or the language is not supported.
            Log.e(TAG, "Language is not available.");
        } else {
            // Check the documentation for other possible result codes.
            // For example, the language may be available for the locale,
            // but not for the specified country and variant.

            // The TTS engine has been successfully initialized.
            // Allow the user to press the button for the app to speak again.
            btn.setEnabled(true);
            // Greet the user.
            //sayHello();
        }
    } else {
        // Initialization failed.
        Log.e(TAG, "Could not initialize TextToSpeech.");
    }
}
```

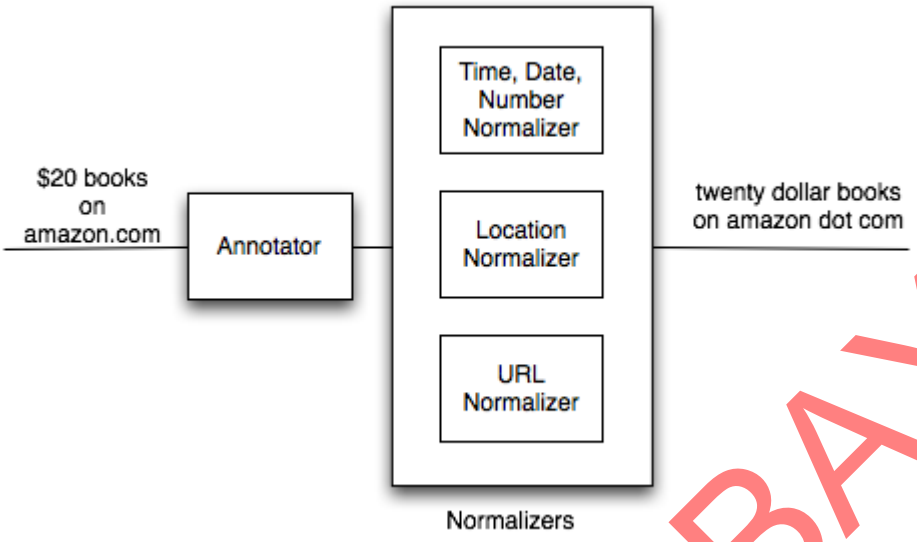
2.8 *Speech Recognition*

Dalam ilmu komputer, *Speech recognition* adalah terjemahan dari kata yang diucapkan menjadi teks. Ia juga dikenal sebagai *automatic speech recognition* (ASR), *computer speech recognition*, *Speech to Text* (STT).

Speech Recognition adalah teknologi yang dapat menerjemahkan kata-kata yang diucapkan menjadi teks. Beberapa sistem *speech recognition* menggunakan "pelatihan" di mana seorang pembicara individu membaca bagian teks ke dalam sistem *speech recognition*. Sistem ini menganalisis suara tertentu orang tersebut dan menggunakannya untuk *fine tune* suara orang itu, sehingga transkripsi lebih akurat. Sistem yang tidak menggunakan pelatihan yang disebut sistem "*Speaker Independent*". Sistem yang menggunakan pelatihan yang disebut sistem "*Speaker Dependent*".



Gambar 2.5 Voice Recognition Basic Block Diagram



Gambar 2.6 Context Specific Text Normalization

Pada gambar 2.6 diatas dapat dilihat proses dari normalisasi teks. Dimana semua angka akan diubah ke bentuk teks. Begitu juga dengan alamat url. Akan diubah menjadi bentuk teks.

Android SDK memudahkan untuk dapat mengintegrasikan *voice recognition* langsung ke dalam aplikasi. Pada penggunaan *speech input* aplikasi harus memeriksa apakah perangkat yang digunakan mampu menjalankan fitur *voice recognition*.

```
PackageManager pm = getPackageManager();
List<ResolveInfo> activities = pm.queryIntentActivities(
        new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
if (activities.size() == 0)
{
    speakButton.setEnabled(false);
    speakButton.setText("Recognizer not present");
}
```

Penggunaan intent diperlukan untuk menjalankan *speech recognition*. Intent sendiri adalah struktur data pasif yang berisikan deskripsi abstrak dari operasi yang akan dilakukan.

```
Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 5);
startActivityForResult(intent, REQUEST_CODE);
```

Untuk mendapatkan hasil dari proses *voice recognition*, maka dapat digunakan onActivityResult(). Salah satu tip penting agar *voice recognition* dapat seakurat mungkin adalah dengan memiliki gagasan tentang kata-kata apa yang mungkin diucapkan oleh *user*.

2.9. Android Calendar

Kalender adalah kelas dasar abstrak untuk mengkonversi antara objek tanggal dan satu set bidang bilangan bulat seperti tahun, bulan, hari, jam, dan sebagainya. (Sebuah objek tanggal dapat menyatakan waktu tertentu dengan presisi mili detik).

(Developers, 2012)

Android *calendar* agar dapat diakses dan digunakan maka kita harus memasukkan *permission* ke dalam file AndroidManifest.xml.

```
<uses-permission android:name="android.permission.READ_CALENDAR" />
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
```

Untuk dapat memasukkan data ke dalam Android *calendar*, maka digunakan C*ontentValues* untuk menampung masukan ke dalam Android *calendar*, kemudian menggunakan URI (“content://com.android.calendar/events” hanya berjalan pada Android versi 2.1 keatas, untuk yang dibawah itu menggunakan “content://calendar/events”) kita dapat memasukkan data ke dalam Android *calendar*. Masukan yang dibutuhkan untuk menambahkan alarm adalah

calendar id, title, description, eventlocation, date start, date end, event status, visibility, transparency, dan *has alarm.* Berikut contohnya :

```
String eventUriString = "content://com.android.calendar/events";
ContentValues eventValues = new ContentValues();

eventValues.put("calendar_id", 1);
eventValues.put("title", title);
eventValues.put("description", addInfo);
eventValues.put("eventLocation", place);

long endDate = startDate + 1000 * 60 * 60;

eventValues.put("dtstart", startDate);
eventValues.put("dtend", endDate);

eventValues.put("eventStatus", status);

eventValues.put("visibility", 3
eventValues.put("transparency", 0);
eventValues.put("hasAlarm", 1);

Uri eventUri =
curActivity.getApplicationContext().getContentResolver().insert(Uri.pars
e(eventUriString), eventValues);
long eventID = Long.parseLong(eventUri.getLastPathSegment());
```

Berikut merupakan contoh untuk dapat menambahkan *reminder*. *Reminder* ini berfungsi untuk memberikan peringatan sebelum tenggat waktu alarm, waktu pengaktifannya sesuai dengan jeda waktu yang telah ditentukan.

```
String reminderUriString = "content://com.android.calendar/reminders";
ContentValues reminderValues = new ContentValues();
reminderValues.put("event_id", eventID);
reminderValues.put("minutes", 5);
reminderValues.put("method", 1);
Uri reminderUri =
curActivity.getApplicationContext().getContentResolver().insert(Uri.pars
e(reminderUriString), reminderValues);
```

2.10. Android Gestures

Android *gestures* berguna untuk berinteraksi dengan aplikasi, dengan melakukan manipulasi gerakan pada layar. Berikut beberapa contoh gerakan dasar yang dikenali.

1. *Touch*

Merupakan fungsi dasar yang digunakan pada kebanyakan *item*. Dilakukan dengan aksi tekan lalu angkat pada layar.

2. *Long Press*

Dilakukan dengan cara menekan lama pada layar,

3. *Swipe*

Dilakukan dengan menggerakkan jari ke arah atas, bawah, kiri, atau kanan pada layar.

4. *Drag*

Cara melakukannya dengan menekan lama, kemudian menggerakkan *item* sesuai dengan arah yang kita mau, kemudian lepas.

5. *Double Touch / Double Tap*

Dilakukan dengan jalan menyentuh layar sebanyak dua kali dalam waktu yang singkat.

6. *Pinch Open*

Cara penggunaannya dengan menaruh dua jari di atas layar, kemudian menggerakkan keduanya berjauhan.

7. *Pinch Close*

Cara penggunaannya dengan menaruh dua jari di atas layar, kemudian menggerakkan keduanya berdekatan.

(Developer,2010)