II. TINJAUAN PUSTAKA

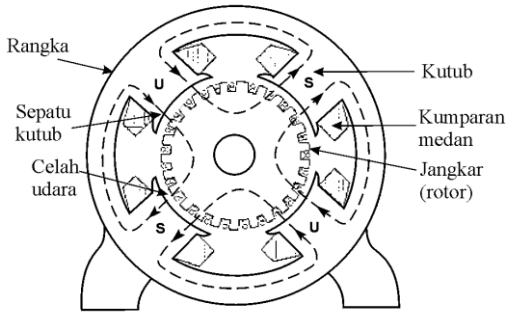
2.1. Motor Arus Searah[8]

Motor arus searah (motor DC) adalah mesin yang merubah enargi listrik arus searah menjadi energi mekanis yang berupa putaran. Berdasarkan fisiknya motor arus searah secara umum terdiri atas bagian yang diam dan bagian yang berputar. Pada bagian yang diam (stator) merupakan tempat diletakkannya kumparan medan yang berfungsi untuk menghasilkan fluksi magnet sedangkan pada bagian yang berputar (rotor) ditempati oleh rangkaian jangkar seperti kumparan jangkar, komutator dan sikat. Motor arus searah bekerja berdasarkan prinsip interaksi antara dua fluksi magnetik. Dimana kumparan medan akan menghasilkan fluksi magnet yang arahnya dari kutub utara menuju kutub selatan dan kumparan jangkar akan menghasilkan fluksi magnet yang melingkar. Interaksi antara kedua fluksi magnet ini akan menimbulkan suatu gaya.

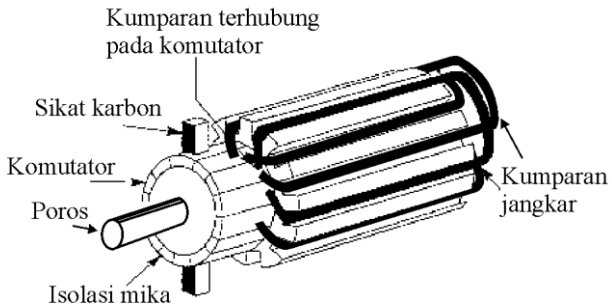
Penggunaan motor arus searah akhir-akhir ini mengalami perkembangan, khususnya dalam pemakaiannya sebagai motor penggerak. Motor arus searah digunakan secara luas pada berbagai motor penggerak dan pengangkut dengan kecepatan yang bervariasi yang membutuhkan respon dinamis dan keadaan *steady-state*. Motor arus searah mempunyai pengaturan yang sangat mudah

dilakukan dalam berbagai kecepatan dan beban yang bervariasi. Itu sebabnya motor arus searah digunakan pada berbagai aplikasi tersebut. Pengaturan kecepatan pada motor arus searah dapat dilakukan dengan memperbesar atau memperkecil arus yang mengalir pada jangkar menggunakan sebuah tahanan.

Konstruksi dari motor arus searah dapat dilihat pada gambar 2.1 berikut,



(a)

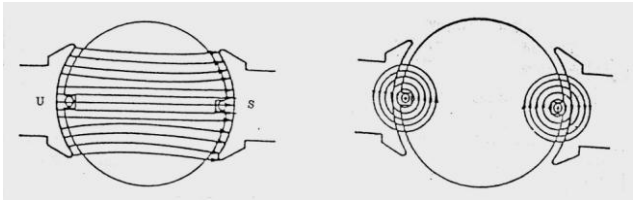


(b)

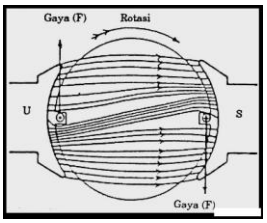
Gambar 2.1 Konstruksi Motor Arus Searah[8]

2.1.1 Prinsip Kerja Motor Arus Searah

Setiap konduktor yang mengalirkan arus mempunyai medan magnet disekelilingnya. Kuat medan magnet yang timbul tergantung pada besarnya arus yang mengalir di dalam konduktor.



(a) (b)



(c)

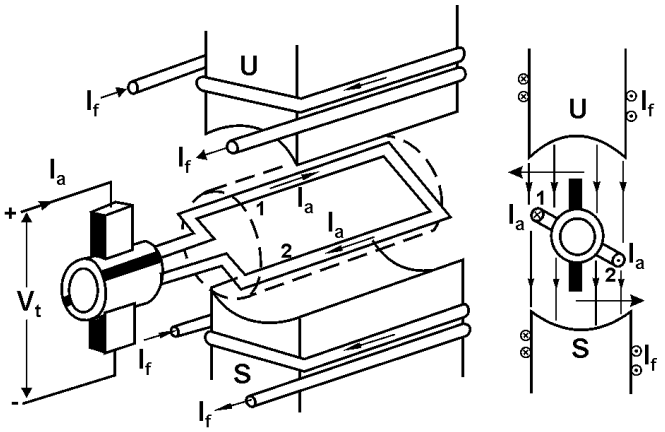
Gambar 2.2 Pengaruh Penempatan Konduktor Pengalir Arus dalam Medan magnet[8]

Pada Gambar 2.2(a) menunjukkan sebuah medan magnet seragam yang dihasilkan oleh kutub-kutub magnet utara dan selatan yang arahnya dari kutub utara menuju kutub selatan.. Sedangkan Gambar 2.2(b) menggambarkan sebuah konduktor yang dialiri arus searah dan menghasilkan medan magnet (garis-garis gaya fluksi) disekelilingnya.

Jika konduktor yang dialiri arus tersebut ditempatkan di dalam medan magnet seragam, maka interaksi kedua medan akan menimbulkan medan yang tidak seragam seperti yang ditunjukkan pada Gambar 2.2(c). Sehingga kerapatan fluksi akan bertambah besar di atas sebelah kanan konduktor (dekat kutub selatan) dan di bawah sebelah kiri konduktor (dekat kutub utara) sedangkan kerapatan fluksi menjadi berkurang di atas sebelah kiri konduktor dan di bawah sebelah kanan konduktor. Kerapatan fluksi yang tidak seragam ini menyebabkan konduktor di sebelah kiri akan mengalami gaya ke atas, sedangkan konduktor di sebelah kanan akan mengalami gaya ke bawah. Kedua gaya tersebut akan menghasilkan

torsi yang akan memutar jangkar dengan arah putaran searah dengan putaran jarum jam.

Selanjutnya prinsip dasar diatas diterapkan pada motor dc. Prinsip kerja sebuah motor arus searah dapat dijelaskan dengan Gambar 2.3 berikut,



Gambar 2.3 Prinsip Kerja Motor Arus Searah[8]

Berdasarkan gambar diatas kedua kutub stator dibelitkan dengan konduktor – konduktor sehingga membentuk kumparan yang dinamakan kumparan stator atau kumparan medan. Misalkan kumparan medan tersebut dihubungkan dengan suatu sumber tegangan, maka pada kumparan medan itu akan mengalir arus medan (I_f). Kumparan medan yang dialiri arus ini akan menimbulkan fluksi utama yang dinamakan fluksi stator. Fluksi ini merupakan medan magnet yang arahnya dari kutub utara menuju kutub selatan (hal ini dapat dilihat dengan adanya garis – garis fluksi). Apabila pada kumparan jangkar mengalir arus yakni arus jangkar, maka dari hukum *Lorenzt* diketahui apabila sebuah konduktor yang dialiri arus ditempatkan pada sebuah medan magnet maka pada konduktor tersebut akan timbul gaya, maka demikian pula

halnya pada kumparan jangkar. Besarnya gaya ini bergantung dari besarnya arus yang mengalir pada kumparan jangkar (I_a), kerapatan fluksi (B) dari kedua kutub dan panjang konduktor jangkar (*l*). Semakin besar fluksi yang terimbas pada kumparan jangkar maka arus yang mengalir pada kumparan jangkar juga besar, dengan demikian gaya yang terjadi pada konduktor juga semakin besar.

Besar gaya yang dihasilkan oleh arus yang mengalir pada konduktor jangkar yang ditempatkan dalam suatu medan magnet adalah,

$F = B.I_a.l$ ..... (2.1)

Dimana,

I_a = Arus jangkar (A)

B = Kerapatan Fluksi (Webber/m^2)

l = Panjang konduktor (m)

Bila kumparan jangkar dari motor berputar dalam medan magnet dan memotong fluksi utama maka sesuai dengan hukum induksi elektromagnetis pada kumparan jangkar akan timbul gaya gerak listrik (ggl) induksi yang arahnya sesuai dengan kaidah tangan kanan, dimana arahnya berlawanan dengan tegangan yang diberikan kepada jangkar atau tegangan terminal. Karena arahnya melawan maka ggl induksi ini disebut ggl lawan.

$E_a = c.n.\phi$ ..... (2.2)

Dimana,

E_a = gaya gerak listrik (volt)

c = konstanta

n = kecepatan putar motor (rpm)

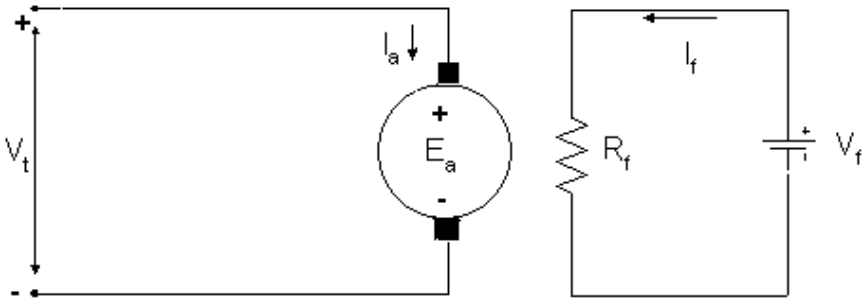
ϕ = fluksi magnetik

2.1.2. Motor Arus Searah Penguatan Bebas

Jenis-jenis motor arus searah dapat dibedakan berdasarkan jenis penguatannya, yaitu hubungan rangkaian kumparan medan dengan kumparan jangkar. Sehingga motor arus searah dibedakan menjadi:

1. Motor arus searah penguatan bebas
2. Motor arus searah penguatan sendiri

Motor arus searah penguatan bebas adalah motor arus searah yang sumber tegangan penguatannya berasal dari luar motor. Dimana kumparan medan disuplai dari sumber tegangan DC tersendiri. Rangkaian ekivalen motor arus searah penguatan bebas dapat dilihat pada gambar 2.4.



Gambar 2.4 Motor Arus Searah Penguatan Bebas[8]

Dari rangkaian tersebut berdasarkan hukum Kirchoff tentang tegangan diperoleh persamaan,

$V_t = E_a + I_a R_a$ ..... (2.3)

$V_f = I_f R_f$ ..... (2.4)

Jika sesuai persamaan 2.2, maka dapat dihasilkan

$n = \frac{V_t - I_a R_a}{c\Phi}$ ..... (2.5)

Dimana,

V_t = tegangan terminal (volt)

E_a = tegangan jangkar (volt)

I_a = arus jangkar (amper)

R_a = tahanan jangkar (ohm)

V_f = tegangan kumparan penguat (volt)

I_f = arus kumparan penguat (amper)

R_f = tahanan kumparan penguat (ohm)

n = kecepatan putar motor (rpm)

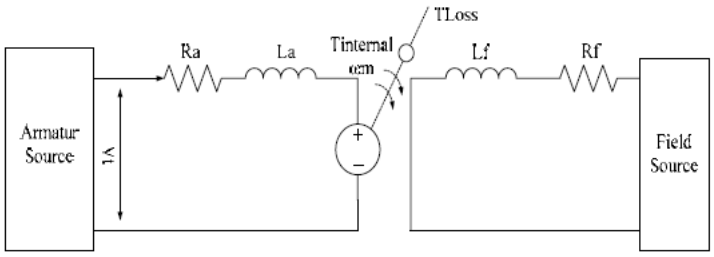
Φ = fluks magnetik

2.1.2.1. Prinsip Kontrol Kecepatan Motor DC Penguatan Bebas[15]

Pada gambar 2.5 menunjukkan rangkaian ekivalen dari motor dc penguat terpisah, dimana pada sumber tegangan kumparan jangkar dan kumparan medan dalam posisi terpisah. Dari rangkaian tersebut diperoleh suatu persamaan berikut,

$V_f = R_f I_f + L_f \frac{di_f}{dt}$ ..... (2.6)

$$V_t = K\Phi\omega + L_a \frac{di_a}{dt} R_a I_a \quad \text{.....(2.7)}$$



Gambar 2.5 Rangkaian Ekivalen Motor DC Penguatan Bebas[15]

Pada keadaan *steady state*, turunan terhadap fungsi waktu adalah nol (0) dan jika variabel I_f, I_a, dan ω konstan, maka diperoleh persamaan sebagai berikut,

$$V_f = R_f I_f \quad \text{.....(2.6)}$$

$$V_t = K\Phi\omega + R_a I_a \quad \text{.....(2.7)}$$

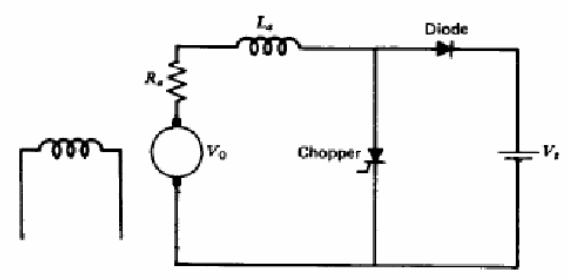
Pada kondisi *steady state*, kecepatan motor dc dapat dikontrol langsung dengan mengatur nilai tegangan terminal jangkar Vt, dapat juga diatur melalui besarnya fluks (Φ) pada kumparan medan dengan cara menambah arus medan (I_f), dari kedua metode ini dapat dikombinasikan untuk mendapatkan *range* pengaturan kecepatan yang lebih baik.

2.1.3. Sistem Pengereman Motor DC

a. Pengereman Regeneratif

Bagan rangkaian pada gambar 2.6 menjelaskan mengenai rangkaian pemenggal yang bekerja sebagai pengerem regeneratif. Vo adalah

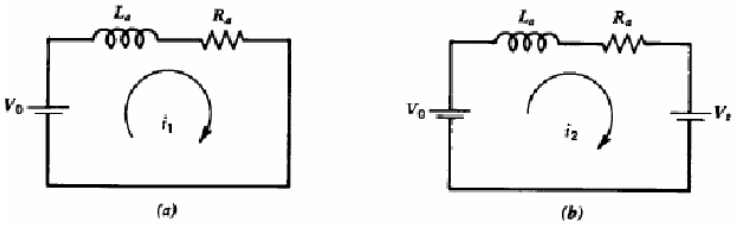
gaya gerak listrik yang dibangkitkan oleh mesin arus searah, sedangkan Vt adalah tegangan sumber bagi motor sekaligus merupakan baterai yang diisi. R_a dan L_a masing-masing adalah hambatan dan induktansi jangkar.



Gambar 2.6 Bagan pengereman regenerative[16]

Prinsip kerja rangkaian diatas ini adalah sebagai berikut :

Ketika saklar pemenggal dihidupkan, maka arus mengalir dari jangkar, melewati skalar dan kembali ke jangkar. Ketika saklar pemenggal dimatikan, maka energi yang tersimpan pada induktor jangkar akan mengalir melewati dioda, baterai dengan tegangan Vt dan kembali ke jangkar. Analogi rangkaian sistem pengereman regeneratif dari gambar di atas dapat dibagi menjadi dua mode.Mode-1 ketika saklar on dan mode ke-2 ketika saklar off seperti ditunjukkan pada gambar di bawah ini.



Gambar 2.7 Rangkaian ekivalen Pengereman Regenaritif untuk (a) saklar On dan (b) saklar Off[16]

b. Pengereman secara Dinamis

Pengereman yang dilakukan dengan melepaskan jangkar yang berputar dari sumber tegangan dan memasangkan tahanan pada terminal jangkar. Oleh karena itu waktu mekanis konstan. Pada dasarnya, T adalah waktu yang diperlukan untuk kecepatan motor jatuh ke 36,8% dari nilai awalnya. Namun, jauh lebih mudah untuk menggambar kurva kecepatan-waktu dengan mendefinisikan konstanta waktu baru T_0 yang merupakan waktu untuk kecepatan dapat berkurang menjadi 50% dari nilai aslinya. Ada hubungan matematis langsung antara konvensional konstanta waktu T dan setengah konstanta waktu T_0.

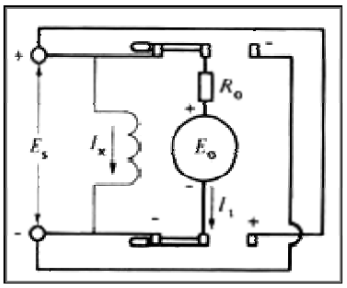
c. Pengereman secara Plugging

Pada pengereman secara plugging mampu menghentikan motor lebih cepat dengan menggunakan metode yang disebut metode plugging. Prinsip kerjanya adalah membalikkan arus jangkar dengan cara membalik terminal sumber (Gambar 2.8).

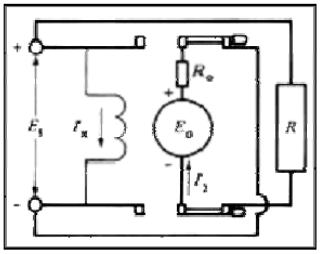
Dibawah kondisi motor normal, jangkar arus I_1dipengeruhi oleh

$I_1=(E_s-E_0)$IR ..... (2.6)

di mana Ro adalah resistansi jangkar. Jika pembalikkan terminal sumber tegangan bekerja, maka pada rangkaian jangkar menjadi (Eo + E s).



Gambar 2.8 Rangkaian Jangkar Terhubung ke Sumber dc (Es)[16]



Gambar 2.9 Rangkaian Jangkar terhubung ke sumber dc (Es) dan resistor (hambatan)[16]

Untuk mencegah suatu hal yang tidak diinginkan, harus ada yang membatasi arus balik dengan menggunakan sebuah resistor dalam seri dengan rangkaian pembalikan (Gambar 2.9). Seperti pada pengereman dinamis, resistor dirancang untuk membatasi pengereman awal arus I_2 sampai sekitar dua kali arus beban penuh. Dengan memasukkan rangkaian, torsi *reverse* akan dikembangkan bahkan ketika jangkar telah berhenti.

2.2. DC-DC *Buck Converter*[9]

Step down converter, atau lebih dikenal sebagai *Buck Converter* terlihat pada gambar 2.9a. Konverter ini terdiri dari tegangan input DC V_s, Saklar kendali *S*, dioda *D*, induktor filter *L*, kapasitor filter *C*, dan beban resistif *R*. Bentuk gelombang dari konverter ini terlihat di gambar 2.9b dimana dengan asumsi bahwa arus induktor selalu positif. Kondisi konverter pada saat arus induktor tidak pernah nol pada semua kondisi disebut kondisi *Continous Conduction Mode* (CCM). Mode ini terjadi pada rangkaian bila saklar *S* diperintahkan pada kondisi hidup, dan dioda akan bias balik. Ketika saklar *S* mati, dioda akan berkonduksi untuk mendukung arus *uninterrupted* dari induktor.

Hubungan antara tegangan input, tegangan keluaran, dan rasio pensaklaran *S*, dapat diketahui, untuk lebih jelas terlihat pada bentuk gelombang tegangan indukto v_L. Menurut hukum Faraday, tegangan-detik dari induktor dihasilkan atas sebuah perioda nol pada kondisi tetap (*steadi-state*). Dimana untuk *buck converter*,

$$(V_s - V_o)DT = -V_o(1 - D)T \quad \text{.....(2.8)}$$

Dimana, fungsi transfer dari tegangan DC didefinisikan sebagai rasio tegangan keluaran terhadap tegangan masukan, yaitu

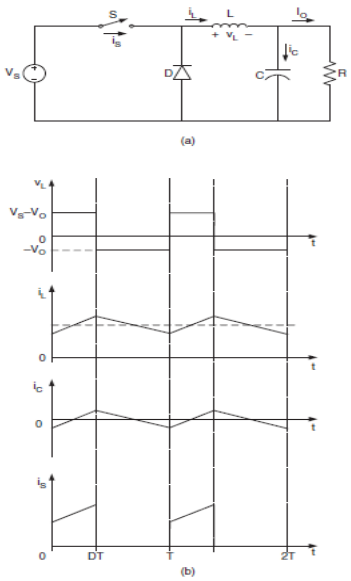
$$M_V \equiv \frac{V_o}{V_s} = D \quad \text{.....(2.9)}$$

Dari persamaan tersebut, maka tegangan keluaran akan selalu lebih kecil dibandingkan tegangan masukan.

DC-DC Converter dapat beroperasi dalam dua kondisi yang berbeda dengan memperhatikan arus induktor i_L. Pada gambar 2.9b menggambarkan mode CCM dimana arus induktor selalu lebih besar dari nol. Jika nilai rata-rata dari arus masukan nol (*R* besar) dan atau frekuensi saklar f kecil maka konverter teramasuk dalam kondisi *Discontinous Conduction Mode* (DCM). Pada kondisi DCM, arus induktor bernilai nol selama bagian dari perioda pensaklaran. Kondisi CCM disukai untuk efisiensi yang tinggi dan memanfaatkan lebih dari saklar semikonduktor dan komponen pasif. Kondisi DCM bisa dipakai pada aplikasi yang meminta kendali secara khusus, dimana dinamisnya konverter dapat dikurangi. Untuk *buck converter*, nilai dari induktansi filter yang membedakan antara DCM dan CCM adalah sebagai berikut,

$$L_b = \frac{(1-D)R}{2f} \quad \text{.....(2.10)}$$

Untuk nilai $L > L_b$, konverter akan beroperasi pada mode CCM.



Gambar 2.10 *Buck Converter* (a) Diagram Rangkaian, (b) Bentuk Gelombang[9]

Arus induktor filter I_L pad CCM terdiri dari sebuah komponen DC I_O dengan komponen ac segitiga yang ter-*superimposed*. Hampir semua dari komponen ac mengiliri kapasitor filter sebagai arus i_c. Arus i_c akan memberikan ripple tegangan yang kecil pada tegangan keluaran V_O. Untuk membatasi nilai *peak-to-peak* dari tegangan ripple dibawah nilai tertentu, kapasitansi filter *C* harus lebih besar dari

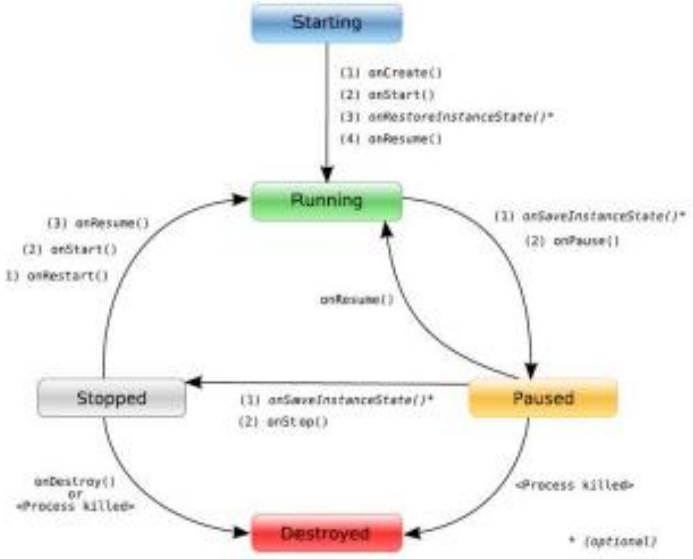
$$C_{min} = \frac{(1-D)V_o}{8V_r Lf^2} \quad \text{..... (2.11)}$$

2.3.Pemrograman Android[5]

Pemrograman android dilakukan dengan menggunakan bahasa pemrograman berbasis java. Pemrograman android ini dilakukan dengan menggunakan program yang bernama eclipse. Dalam program eclipse terdapat *tool* ADT yang diinstal agar program eclipse ini dapat digunakan untuk membuat aplikasi android.

Perangkat berbasis android hanya mempunyai satu layar foreground. Normalnya saat menghidupkan android, yang pertama terlihat adalah *home*. Kemudian sebagai contoh apabila dijalankan sebuah aplikasi catur, *User Interface* (UI) akan menumpuk diatas layar sebelumnya (*home*). Kemudian bila melihat *help*-nya "catur", maka UI help akan menimpa UI sebelumnya (catur), begitu seterusnya. Semua proses diatas direkam di *application stack* oleh sistem *Activity manager*. Menekan tombol back hanya kembali ke halaman sebelumnya, analoginya mirip dengan browser dimana ketika menekan tombol *back* browser akan kembali menampilkan halaman sebelumnya. Setiap *User Interface* diwakili oleh kelas Activity (*Activity class*). Setiap *activity* mempunyai siklus, dimana siklus tersebut

dapat dilihat di gambar 2.10. Sebuah aplikasi dapat terdiri dari satu atau lebih *activity* yang diproses.



Gambar 2.11. Siklus *Activity* Pemrograman Android[5]

Di dalam siklus *activity* dari sebuah aplikasi android terdapat beberapa method yang perlu diperhatikan, antara lain:

a. *onCreate()*: ini adalah method yag dipanggil ketika *activity* pertama kali dilakukan. Method ini memanggil satu parameter, bisa juga null, atau nilai terakhir yang disimpan oleh method onSaveInstanceState().
b. *onStart()*: memulai *activity* ketika antarmuka pengguna ditampilkan.
c. *onPause()*: ini dijalankan di balik layar. Contohnya ketika pengguna menjalankan aplikasi lain dan tampil pada layar pengguna maka otomatis aplikasi sebelumnya akan di-*pause* terlebih dahulu.
d. *onStop()*: method ini dijalankan saat aplikasi sudah tidak dijalankan dan tidak dibutukan.
e. *onRestart*: method ini dijalankan ketika sebuah *activity* dari posisi onStop() dan akan dijalankan kembali.

f. *onDestroy()*: method ini dijalankan saat *activity* di-*destroy* dan aplikasi akan dimatikan.

g. *onSaveInstanteState(bundle)*: method ini akan digunakan aplikasi untuk menyimpan setiap status dari *activity* yang dijalankan dari hal ini secara otomastis telah dilakukan android.

Sebagai seorang programer harus mengetahui beberapa komponen aplikasi yang sangat penting seperti *activities*, *intens*, *service*, dan *content providers*.

a. *Activity*

Normalnya setiap *activity* menampilkan satu buah *user interface* kepada pengguna. Misalnya sebuah *activity* menampilkan daftar menu minuman, kemudian pengguna dapat memilih satu jenis minuman. Contoh lainnya pada aplikasi sms, dimana satu *activity* digunakan untuk menulis pesan, *activity* berikutnya untuk menampilkan nomor kontak tujuan, atau *activity* lainnya digunakan untuk menampilkan pesan-pesan lama. Meskipun *activity-activity* diatas terdapat dalam satu aplikasi sms, namun masing-masing *activity* berdiri sendiri. Untuk pindah dari satu *activity* ke *activity* lainnya dapat melakukan suatu event misalnya tombol diklik atau melalui trigger tertentu.

b. *Service*

Service tidak memliki *user interface*, namun *service* ini berjalan di belakang layar. Misalnya pada pemutar musik, sebuah *activity* digunakan untuk memilih lagu kemudian dimainkan. Agar pemutar musik bisa berjalan dibelakang aplikasi lain maka harus menggunakan *service.*

c. *Intent*

Intent adalah mekanisme untuk menggambarkan sebuah *action* secara detail seperti bagaimana cara mengambil sebuah foto.

d. *Content Provider*

Menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya dalam penggunaan applikasi berbasis peta (MAP). *Activity* membutuhkan cara untuk mengakses data kontak untuk prosedur navigasi. Disinilah peran *content providers*.

2.3.1. Paket Android.Speech[7]

Pada paket Android.Speech ini terdapat satu *interface* dan lima *class* di dalamnya. *Interface* yang terdapat pada paket *speech* ini adalah *RecognitionListener*. *Interface RecognitionListener* ini digunakan untuk menerima notifikasi dari *SpeechRecognizer* selama proses pengenalan berjalan. Semua panggilan balik akan dieksekusi pada thread aplikasi utama. Pada *interface RecognitionListener* ini terdapat *public method* yaitu:

a. *onBeginningOfSpeech()*, digunakan ketika memulai berbicara.
b. *onBufferReceived(byte[] buffer)*, tujuan dari fungsi ini adalah untuk memberikan umpan balik kepada pengguna mengenai audio yang ditangkap.
c. *OnEndOfSpeech()*, digunakan ketika setelah pengguna berhenti berbicara.
d. *onError()*, terjadi ketika ada kesalahan pada jaringan dan pengenalan.

e. *onPartialResult(),* digunakan ketika sebagian pengenalan dapat digunakan.
f. *onReadyForSpeech()*, digunakan ketika titik terakhir telah siap untuk memulai berbicara.
g. *onResult(),* digunakan ketika hasil pengenalan telah didapatkan.

Selain *interface*, pada paket ini juga terdapat *class* yang digunakan, antara lain:

a. *RecognitionService*, *class* ini memberikan sebuah *class* dasar untuk mengimplementasikan layanan pengenalan.
b. *RecognitionService.Callback*, *class* ini menerima panggilan balik dari layanan pengenalan suara dan diteruskan ke pengguna.
c. *RecognizerIntent*, konstanta untuk mendukung pengenalan ucapan dengan memulai sebuah *intent*.
d. *RecognizerResultIntent*, konstanta untuk menguhubungkan *intent* dengan hasil pengenlan ucapan.
e. *SpeechRecognizer*, *class* ini memberikan akses kepada layanan pengenalan ucapan.

2.3.2. Paket Bluetooth Android[7]

Platform android terdapat sebuah *stack* bluetooth yang memungkikan pengguna untuk bertukar data dengan perangkat bluetooth lainnya secara nirkabel. Kerangka aplikasi ini memberikan akses fungsi bluetooth ini melalui APIs bluetooth android. API ini memungkinkan aplikasi nirkabel

terhubung dengan perangkat bluetooth lainnya, yang bisa akses *point-to-point* ataupun *multipoint*.

Dengan menggunakan API bluetooth ini, sebuah android dapat melakukan beberapa hal, antara lain:

- Memindai perangkat bluetooth lainnya
- Meng-*query* perangkat bluetooth lainnya untuk memasangkan perangkat
- Membuat sambungan RFCOMM
- Membangun sebuah sambungan dengan menggunakan pencarian layanan pencarian.
- Mengirimkan data dari dan ke perangkat lainnya
- Mengatur banyak koneksi

Dari semua API bluetooth ini dapat ditemukan dalam paket android.bluetooth. Berikut ini akan dijelaskan *class* dan *interface* yang akan dibutukan untuh membuat sebuah komunikasi bluetooth.

a. BluetoothAdapter

Mempresentasikan adapter *bluetooth* lokal yang ada (bluetooth radio). Dengan menggunakan *BluetoothAdapte*r ini, aplikasi dapat mencari perangkat bluetooth lainnya, meng-*query* daftar perangkat yang terhubung, memberikan sebuah *BluetoothDevice* cara mengetahui *MAC address* dan membuat sebuah *BluetoothServerSocket* untuk berkomunikasi dengan perngkatlainnya.

b. *BluetoothDevice*

BluetoothDevice ini digunakan untuk meminta sebuah sambungan dengan perangkat kontrol melalui sebuah *BluetoothSocket* atau informasi *query* tentang perangkat seperti nama, alamat, *class*, dan *bonding state*.

c. *BluetoothSocket*

Memberikan antarmuka untuk sebuah *socket* bluetooth. *Point* sambungan ini memungkinkan sebuah aplikasi untuk mengubah data dengan perangkat lainnya melalui *outputStream* dan *inputStream*.

d. *BluetoothServerSocket*

Memberikan sebuah *open server socket* untuk melihat permintaan yang masuk. Dalam menghubungkan dua buah perangkat android, salah satu dari perangkat harus membuka *server socket* dengan menggunakan *class* ini.

e. *BluetoothClass*

Mendeskripsikan karakteristik umum dan kapabilitas dari perangkat bluetooth.

f. *BluetoothProfile*

Sebuah antarmuka yang mendefinisikan profil bluetooth.

g. *BluetoothHeadset*

Memberikan dukungan *headset* bluetooth untuk digunakan oleh handphone.

h. BluetoothA2dp

Menjelaskan bagaimana suara dengan kualitas tinggi dapat dikirimkan dari satu perngakat ke perangkat lainnya melalui sambungan bluetooth.

i. BluetoothHealth

Mereprsentasikan sebuah *proxy* profil kesehatan perangkat yang mengontrol layanan bluetooth.

j. BluetoothHealthCallback

Sebuah *class* abstrak yang digunakan mengimplementasikan panggilan balik *bluetoothhealth*.

k. BluetoothHealthAppConfiguration

Merepresentasikan konfigurasi aplikasi *bluetooth health thrid-party* register aplikasi untuk berkomunikasi dengan perangkat kontrol *health bluetooth*.

l. BluetoothProfile.ServiceListener

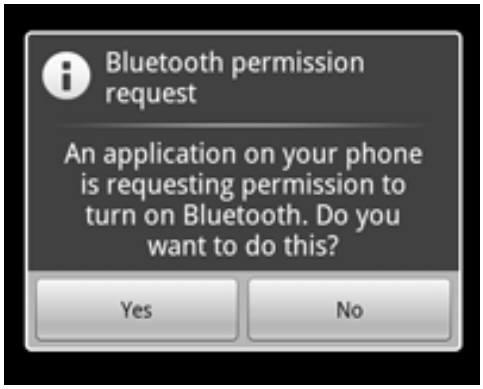
Sebuah antarmuka dimana pemberitahuan profil bluetooth *client* profil pada saat terhubung atau tidak dengan layanan.

2.3.2.1. *Bluetooth Permission*

Dalam menggunakan fitur bluetooth pada aplikasi, aplikasi harus medeklarasikan *bluetooth permission* ini. Aplikasi memperlukan izin untuk menampilkan komunikasi bluetooth, seperti meminta sebuah sambungan, menerima sebuah sambungan, dan transfer data.

Contoh pendeklarasian perizinan bluetooh dapat dilihat pada kode program berikut.

```
<manifest ... >
  <uses-permission
android:name="android.permission.BLUETOOTH" />
  ...
</manifest>
```

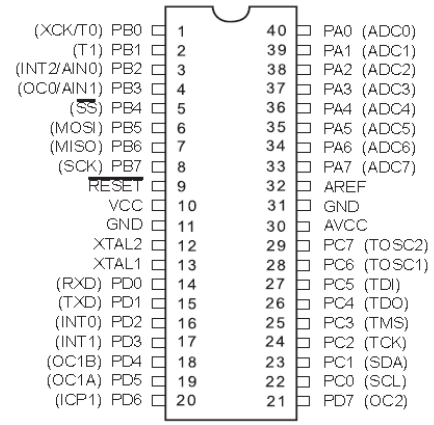


Gambar 2.12 *Bluetooth Permission* untuk mengaktifkan Perangkat bluetooth[7]

2.4. Mikrokontroler ATMega32[2]

Mikrokontroler ATmega32 adalah mikrokontroler 8-bit keluaran Atmel dari keluarga AVR. Pihak Atmel menyatakan bahwa AVR bukanlah sebuah akronim atau singkatan dari suatu kalimat tertentu, perancang arsitektur AVR, Alf-Egil Bogen dan Vegard Wollan tidak memberikan jawaban yang pasti tentang singkatan AVR ini. Mikrokontroler ini dirancang berdasarkan arsitektur AVR RISC (*Reduced Instruction Set Computer*) yang mengeksekusi satu instruksi dalam satu siklus *clock* sehingga dapat mencapai eksekusi instruksi sebesar 1 MIPS (*Million Instruction Per Second*) setiap 1MHZ frekuensi *clock* yang digunakan mikrokontroler tersebut. Frekuensi *clock* yang digunakan dapat diatur

melalui *fuse bits* dan kristal yang digunakan. Gambar 2.12 merupakan susuan pin mikrokontroler ATmega32.



Gambar 2.13 Susunan Pin ATmega32[2]

2.4.1 USART (*Universal Synchronous Asynchronous Receiver Transmitter*)[14]

USART (*Universal Synchronous Asynchronous serial Receiver and Transmitter*) merupakan protokol komunikasi serial yang terdapat pada mikrokontroler AVR. Dengan memanfaatkan fitur ini mikrokontroler dapat berhubungan dengan "dunia luar". Dengan USART, bisa menghubungkan mikrokontroler dengan PC, *handphone*, GPS atau bahkan modem, dan banyak lagi peralatan yang dapat dihubungkan dengan mikrokontroler dengan menggunakan fasilitas USART.

Komunikasi dengan menggunakan USART dapat dilakukan dengan dua cara yaitu dengan mode sinkron dimana pengirim data mengeluarkan pulsa/*clock* untuk sinkronisasi data, dan yang kedua dengan mode asinkron dimana pengirim data tidak mengeluarkan pulsa/*clock* tetapi untuk proses sinkronisasi memerlukan inisialisasi agar data yang diterima

sama dengan data yang dikirimkan. Pada proses inisialisasi ini setiap perangkat yang terhubung harus memiliki *baud rate* (laju data) yang sama. Pada mikrokontroler AVR untuk mengaktifkan dan mengeset komunikasi USART dilakukan dengan cara mengaktifkan register-register yang digunakan untuk komunikasi USART. Register-register yang dipakai antara lain:

a. UDR

   Merupakan register 8 bit yang terdiri dari 2 buah dengan alamat yang sama, yang digunakan sebagai tempat untuk menyimpan data yang akan dikirimkan (TXB) atau tempat data diterima (RXB) sebelum data tersebut dibaca.

b. UCSRA

   Merupakan register 8 bit yang digunakan untuk mengendalikan mode komunikasi USART dan untuk membaca status yang sedang terjadi pada USART.

c. UCSRB

   Merupakan register 8 bit yang digunakan untuk mengendalikan mode komunikasi USART dan untuk membaca status yang sedang terjadi pada USART.

d. UCSC

   Merupakan register 8 bit yang digunakan untuk mengendalikan mode komunikasi USART dan untuk membaca status yang sedang terjadi pada USART.

e. UBRRH dan UBRRL

Merupakan register 16 bit yang digunakan untuk mengatur laju data (baud rate) pada saat mode komunikasi asinkron.

2.4.2. Mode *Fast Pwm Timer0*[2]

Mode *Fast PWM* memberikan pilihan untuk membangkitan bentuk gelombang dengan frekuensi tinggi. *Fast PWM* berbebeda dengan mode PWM lainnya, karena pada mode ini menggunakan operasi *single-slope*. *Counter* akan menghitung dari *BOTTOM* menuju *MAX* dan kembali ke *BOTTOM*. Dengan frekuensi yang tinggi ini, mode *fast pwm* cocok untuk regulator, penyearah dan aplikasi DAC.

Dalam mode *fast pwm*, memungkinkan unit pembangding membangkitan bentuk gelombang pada pin OC0. Pengaturan bit COM1:0 menjadi 2 akan menghasilkan keluaran pwm non-inverting dan pwm inverting apabila bit COM1:0 menjadi 3. Nilai OC0 yang sebenarnya akan terlihat jika PORT tersebut di atur sebagai keluaran. Gelombang PWM dibangkitkan dengan mengatur register OC0 dimana pencocokan perbandingan antara OCR0 dan TCNT0, serta penghapusan register OC0 pada siklus *clock timer* dari *counter* dihapus. Untuk pengaturan frekuensi PWM digunakan persamaan,

$$f_{OCnPWM} = \frac{f_{clk_I/o}}{N.256} \quad \text{..... (2.12)}$$