

IMPLEMENTASI KENDALI JARAK JAUH ROBOT BERODA

BERBASIS *WEB SERVER*

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat

Untuk Memperoleh Gelar Sarjana Teknik Elektro



Disusun Oleh :

Nama : FRISTA ASRORI PRAWIRADIRGA

No. Mahasiswa : 11 524 004

**JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA**

2015

HALAMAN PENGESAHAN PEMBIMBING

IMPLEMENTASI KENDALI JARAK JAUH ROBOT BERODA

BERBASIS *WEB SERVER*

ISLAM

TUGAS AKHIR

Disusun Oleh :

Nama : Frista Asrori Prawiradirga

No. Mahasiswa : 11 524 004

Yogyakarta, Mei 2015

Dosen Pembimbing



Sisdarmanto Adinandra, Ph.D.

LEMBAR PERNYATAAN KEASLIAN

Saya yang bertanda tangan di bawah ini.

Nama : Frista Asrori Prawiradirga

No. Mahasiswa : 11 524 004

Menyatakan dengan jujur bahwa tugas akhir ini adalah hasil pekerjaan saya sendiri, tanpa adanya niat untuk menjiplak atau melakukan plagiat dari karya orang lain. Adapun bagian-bagian tertentu dari sumber referensi yang saya ambil, sesuai dengan ketentuan dan tata tertib yang berlaku dalam penulisan karya ilmiah yang lazim. Jika terbukti ada sesuatu hal yang melanggar, saya siap bertanggung jawab sepenuhnya.

Yogyakarta, Mei 2015

Frista Asrori Prawiradirga

NIM 11 524 004

LEMBAR PENGESAHAN PENGUJI

**IMPLEMENTASI KENDALI JARAK JAUH ROBOT BERODA
BERBASIS *WEB SERVER***

TUGAS AKHIR

Oleh :

Nama : Frista Asrori Prawiradirga

No. Mahasiswa : 11 524 004

**Telah Dipertahankan Di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Fakultas Teknologi Industri Universitas Islam Indonesia**

Yogyakarta, Mei 2015

Tim Penguji,

Sisdarmanto Adinandra, Ph.D.

Ketua

Dwi Ana Ratna Wati, S.T., M.Eng.,

Anggota I

Firdaus, S.T., M.T.,

Anggota II



Mengetahui,

Ketua Jurusan Teknik Elektro

Universitas Islam Indonesia



Hendra Setiawan, S.T., M.T., Ph.D.

HALAMAN PERSEMBAHAN

Puji syukur terhaturkan kepada Allah SWT tuhan semesta alam. Oleh karena nikmat dan karunia-Nya sehingga kita selalu sehat, semangat dan tetap bersahaja. Sholawat serta salam terhaturkan pula kepada Nabi Muhammad SAW.

Segenap kasih, sayang, dan cinta, skripsi ini penulis persembahkan kepada baginda Ayah dan Ibu yang selalu mendoakan, mendukung, serta membantu dalam segala hal apapun dalam tahap pendidikan tinggi ini.

Skripsi ini pula penulis persembahkan kepada teman-teman Teknik Elektro UII 2011, Adit, Putra, Latif, Dwi, Rendy, Wahid, Rizky, Agung dan teman-teman Teknik Elektro UII yang lainnya. Kemudian kepada teman-teman UII lainnya Ufi, Hamzah, Doddy, Ligar, Rudy, Ahsan, serta semua pihak yang telah banyak membantu dalam penyusunan skripsi yang telah hadir dalam hidupku dan menggoreskan tulisan pengalaman, canda dan tawa.

Skripsi ini pula penulis persembahkan kepada semua tenaga pendidik yang telah mengantarkan penulis sampai ke tahap ini.

HALAMAN MOTTO

“Kuliah tidak semata untuk mencari nilai, bahkan cuma kesuksesan, tapi kamu harus tahu apa itu nikmat Allah, Syukuri itu dan ingat selalu niat semata kepada Allah untuk kuliah. Kamu harus selalu mengikut sertakan Allah dalam setiap tindakanmu.

“Student Today, Leader Tomorrow”.

“Thinking Out Of The Box”.

“Man Behind The Gun adalah ungkapan bagi siapa saja yang ingin menguasai suatu urusan, Senjata itu adalah ilmu pengetahuan dan iman, kuasai itu maka kamulah yang akan jadi pemimpin”.

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillah, rasa syukur dan terima kasih terhaturkan pada-Mu ya Rabb atas karunia nikmat yang telah diberikan sehingga tugas akhir yang berjudul “Implementasi Kendali Jarak Jauh Robot Beroda Berbasis *Web Server*” telah selesai dengan baik dan lancar. Tak lupa sholawat dan salam tercurah kepada Rasulullah Muhammad SAW. yang menjadi teladan baik bagi kita.

Rasa syukur yang tidak pernah berhenti, atas selesainya laporan skripsi yang merupakan tugas akhir bagi setiap mahasiswa tingkat S1, semoga karya ini bisa memberikan manfaat bagi seluruh pembaca untuk ke depannya. Banyak kesan suka dan duka yang mengiringi dalam proses pengerjaan tugas akhir ini.

Rasa syukur terima kasih juga terhaturkan kepada seluruh pihak yang terkait dalam proses pengerjaan tugas akhir ini. Atas dukungan, bimbingan, kerja sama, dan fasilitas, diucapkan terima kasih kepada :

1. Ayah, ibu, dan adik-adik yang selalu memberikan inspirasi dan semangat serta dukungan yang tak henti.
2. Bapak Sisdarmento Adinandra, Ph.D. selaku pembimbing tugas akhir dan selalu memberikan motivasi dan masukan terkait.
3. Seluruh dosen Jurusan Teknik Elektro UII, terima kasih atas bimbingan selama mengikuti kegiatan perkuliahan.
4. Argian Winingrum yang selalu mendukung pengerjaan tugas akhir.
5. Teman-teman Teknik Elektro UII 2011 yang memberikan motivasi.
6. Dan banyak pihak lain yang tidak dapat disebutkan seluruhnya yang telah membantu dalam penyelesaian tugas akhir ini.

Adanya kekurangan dalam penyusunan tugas akhir ini karena keterbatasan ilmu yang dimiliki oleh penyusun. Untuk itu, adanya kritik dan saran yang membangun dapat membantu kesempurnaan dari tugas akhir ini. Semoga apa yang telah didapat dari tugas akhir ini dapat memberikan manfaat yang sangat baik bagi para pembaca dan penggunanya.

Wasaalamualaikum Warahmatullahi Wabarakatuh.

Yogyakarta, Mei 2015

Penulis

DAFTAR ISI

HALAMAN JUDUL.....	i
HALAMAN PENGESAHAN PEMBIMBING	ii
LEMBAR PERNYATAAN KEASLIAN	iii
LEMBAR PENGESAHAN PENGUJI	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTTO	vi
KATA PENGANTAR	vii
DAFTAR ISI.....	ix
DAFTAR GAMBAR	xi
DAFTAR TABEL	xii
ABSTRAK	xiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah.....	2
1.3 Batasan Masalah	2
1.4 Tujuan dan Manfaat Penelitian	3
BAB II TINJAUAN PUSTAKA.....	4
2.1 Robot Beroda	4
2.2 Perangkat Lunak Android	5
2.2.1 Bagian-bagian dalam Aplikasi Android.....	5
2.3 <i>Google Cloud Messaging</i> (GCM).....	8
2.3.1 Cara Kerja <i>Google Cloud Messaging</i> (GCM).....	8
2.4 Google Maps	12
2.4.1 Cara Kerja “ <i>My Location</i> ” Google Maps.....	13
2.5 <i>Web Server</i>	14
2.5.1 <i>Hyper Text Markup language</i> (HTML)	15
2.5.2 PHP	15
2.5.3 <i>Database MySQL</i>	16
2.6 IOIO-OTG untuk perangkat Android	17
2.7 Penelitian Sejenis	18

BAB III PERANCANGAN SISTEM	20
3.1 Perancangan <i>Web Server</i>	23
3.1.1 Sublime Text 3	23
3.1.2 Wamp Server.....	24
3.1.3 Gerak Navigasi Kendali dan Fungsi	25
3.1.4 Hasil Foto dari Perangkat Android	27
3.1.5 Pembaharuan Peta Lokasi Android.....	28
3.2 Perancangan Aplikasi Android	30
3.2.1 Tampilan Aplikasi Android	31
3.2.2 Proses Kerja Aplikasi Android.....	32
3.3 Perancangan Perangkat Keras pada Robot Beroda.....	39
BAB IV PENGUJIAN, ANALISIS, DAN PEMBAHASAN.....	42
4.1 Pengujian Perangkat Keras Robot Beroda.....	43
4.2 Pengujian Aplikasi Android.....	44
4.2.1 Pembahasan Aplikasi Android.....	45
4.3 Pengujian <i>Web Server</i>	47
4.3.1 Pembahasan <i>Web Server</i>	48
4.4 Pengujian Keseluruhan Sistem	49
4.4.1 Pengujian <i>Indoor</i>	52
4.4.2 Pengujian <i>Outdoor</i>	55
4.4.3 Analisa Keseluruhan Sistem	59
BAB V PENUTUP.....	62
4.1 Kesimpulan	62
4.2 Saran	62
DAFTAR PUSTAKA	64
LAMPIRAN.....	66

DAFTAR GAMBAR

Gambar 2. 1 (a) Robot beroda dua, (b) robot beroda <i>caterpillar</i>	4
Gambar 2. 2 Implementasi <i>subclass</i> dari <i>activities</i>	6
Gambar 2. 3 Contoh implementasi subclass <i>service class</i>	7
Gambar 2. 4 Implementasi subclass <i>broadcast reciever</i>	7
Gambar 2. 5 Implementasi subclass <i>content provider</i>	8
Gambar 2. 6 Proses pengambilan data pada GCM.	9
Gambar 2. 7 Penggunaan kunci API untuk menampilkan Google Maps.	13
Gambar 2. 8 Tampilan layanan google maps.....	13
Gambar 2. 9 Contoh penerapan database pada <i>file</i> PHP.....	16
Gambar 2. 10 Sinyal PWM.	18
Gambar 3. 1 Diagram rancangan sistem kendali jarak jauh robot beroda berbasis <i>web server</i>	20
Gambar 3. 2 Tampilan gambar localhost menggunakan wamp server.	24
Gambar 3. 3 Tampilan <i>web server</i> setelah di <i>host</i> ke internet.	25
Gambar 3. 4 Diagram alir proses penerimaan perintah kendali gerak robot beroda melalui <i>web server</i> ke Android.	27
Gambar 3. 5 Basis data untuk menampilkan posisi keberadaan Android.....	29
Gambar 3. 6 Proses pengambilan dan pengiriman data lokasi perangkat Android.	30
Gambar 3. 7 Tampilan aplikasi Android.....	32
Gambar 3. 8 Diagram alir aplikasi Android.....	33
Gambar 3. 9 Servis yang dapat diakses oleh aplikasi Android di penelitian ini...	40
Gambar 3. 10 Rangkaian skematik robot beroda.	41
Gambar 4. 1 Penghapusan pesan oleh <i>server</i> GCM.....	47
Gambar 4. 2 Proses pengiriman data dari pengirim (<i>laptop</i>) ke robot beroda.....	51
Gambar 4. 3 Grafik hubungan lama pengiriman dengan jarak pengiriman pesan pada operator A dalam pengujian <i>indoor</i>	52
Gambar 4. 4 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator B dalam pengujian <i>indoor</i>	53
Gambar 4. 5 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator C dalam pengujian <i>indoor</i>	54
Gambar 4. 6 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator A dalam pengujian <i>outdoor</i>	56
Gambar 4. 7 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator B dalam pengujian <i>outdoor</i>	57
Gambar 4. 8 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator C dalam pengujian <i>outdoor</i>	58

DAFTAR TABEL

Tabel 4. 1 Nilai PWM pada kecepatan gerak robot beroda.	43
Tabel 4. 2 Spesifikasi <i>laptop</i> yang digunakan untuk pengujian.....	51
Tabel 4. 3 Spesifikasi Android yang digunakan untuk pengujian.	51
Tabel 4. 4 Data pengujian <i>indoor</i> pada operator A.....	53
Tabel 4. 5 Data pengujian <i>indoor</i> pada operator B.	54
Tabel 4. 6 Data pengujian <i>indoor</i> pada operator C.	55
Tabel 4. 7 Data pengujian <i>outdoor</i> pada operator A.....	56
Tabel 4. 8 Data pengujian <i>outdoor</i> pada operator B.	58
Tabel 4. 9 Data pengujian <i>outdoor</i> pada operator C.	59
Tabel 4. 10 Rata-rata total waktu yang dibutuhkan untuk pengiriman pada kondisi <i>indoor</i> dan <i>outdoor</i>	60
Tabel 4. 11 Hubungan data rate komponen terhadap waktu tunda (<i>delay</i>) pengiriman perintah.....	60

ABSTRAK

Implementasi komunikasi pada sistem robot terbagi dalam dua jenis, yaitu komunikasi lokal dan komunikasi global. Komunikasi lokal adalah sistem komunikasi yang mencakup wilayah kecil untuk pertukaran data, sedangkan komunikasi global adalah sistem komunikasi yang mencakup wilayah yang sangat luas untuk pertukaran data. Cara yang sering digunakan adalah komunikasi lokal, karena komunikasi ini murah dan handal. Komunikasi lokal memiliki kelemahan pada jarak yang terbatas sehingga dibutuhkan komunikasi global. Pada penelitian ini digunakan sistem robot dengan komunikasi kendali global. Komunikasi yang digunakan adalah internet yang diakses melalui *web server* dan aplikasi Android. Komponen pada sistem ini meliputi robot beroda, *web server*, aplikasi Android, dan GCM (Google Cloud Messaging). *Web server* digunakan untuk mengirim data perintah ke aplikasi Android, pada halaman *web server* terdapat tiga bagian yaitu navigasi gerak robot, tampilan foto, dan tampilan peta. Aplikasi Android sebagai jembatan data perintah yang diberikan *web server* untuk menggerakkan robot. Aplikasi Android pada sistem ini memiliki beberapa fungsi yaitu penerima data dan pengirim data ke *web server*, mengambil foto, mengaktifkan *flash*, pemutar suara, dan mengakses GPS. GCM digunakan sebagai jembatan transfer data dari *web server* ke aplikasi Android. Fitur ini digunakan sebagai layanan pengiriman pesan ke peranti Android, pesan yang dilewatkan ke GCM adalah pesan yang digunakan untuk menjalankan perintah gerak robot beroda dan mengaktifkan fitur pada aplikasi Android. Sistem yang sudah berjalan selanjutnya diuji pada seluruh komponen sistem meliputi robot beroda, *web server*, dan aplikasi Android. Pada pengujian keseluruhan sistem digunakan 3 operator internet yang berbeda untuk dilihat lama pengiriman pesan terhadap jarak pesan yang dikirim. Dari pengujian tersebut disimpulkan bahwa robot dapat dikendalikan jarak jauh serta bertukar data dengan media *web server* tanpa terbatas jarak jangkauan.

Kata kunci : internet robot, GCM, web server, Android.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi robot beroda memiliki sejarah yang cukup panjang. Prototipe pertama yang dapat berfungsi dengan baik diciptakan tahun 1966 – 1972 di Lab penelitian Stanford oleh Shakey [1]. Robot tersebut memiliki kemampuan terbatas untuk memahami dan memodelkan lingkungannya, dia bisa melakukan tugas yang diperlukan seperti mencari rute dan mengambil objek sederhana.

Teknologi kendali jarak jauh sudah lama dikenal oleh manusia, namun terbatas dalam jarak tergantung pada media komunikasi yang digunakan. Kebanyakan yang beredar dalam pasar adalah sistem kendali dengan media Wi-Fi atau *bluetooth*, atau untuk media yang lebih jauh lagi digunakan modul GSM dan ethernet. Semua media tersebut mempunyai kelebihan dan kekurangan masing-masing. Peranti yang digunakan dalam penelitian ini adalah komunikasi internet yang tersedia dalam peranti Android dan juga akses di dalam *web server* ditambah dengan letak lokasi robot berada menggunakan layanan *Google Maps* dan *frame* hasil foto bidikan kamera Android yang bisa dilihat pada *web server*.

Robot sebelumnya milik Pangestu [2], yang sudah pernah di implementasikan memiliki kelemahan dalam jangkauan jarak kendali. Kendali pada robot sebelumnya berjarak sekitar 90 meter dengan menggunakan media *Wi-Fi* / akses poin sebagai media transfer data. Penelitian ini bertujuan untuk memaksimalkan jarak kendali robot sebelumnya. Guna memaksimalkan jarak kendali tersebut maka digunakan sistem *web server* agar media kendali memiliki jarak yang lebih jauh.

Sistem *web server* yang digunakan pada penelitian ini adalah sistem yang dapat menghubungkan Android dengan *web server* dan bisa saling bertukar data. Tampilan pada *web* meliputi perintah navigasi gerak robot dan tombol fungsi lainnya, peta keberadaan robot berada, *log* sejarah perintah dari *user*, dan *frame* hasil bidikan kamera Android. *Web server* diakses oleh robot menggunakan media internet pada peranti Android, sehingga antara *web* dan robot dapat saling mengirim data baik berupa sinyal kendali atau data visual yang tertangkap oleh kamera Android.

1.2 Rumusan Masalah

Berdasarkan pengamatan dan deskripsi latar belakang implementasi kendali jarak jauh robot berbasis *web server*, maka perlu adanya rumusan masalah yang akan dikerjakan dalam penelitian ini. Rumusan masalah tersebut adalah bagaimana membuat sistem robot yang dapat diakses serta dapat bertukar data melalui media *web server*.

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah :

1. Robot yang digunakan adalah robot beroda dengan menggunakan media komunikasi data internet.
2. *Web server* digunakan dalam media transfer sinyal kendali dan data dari kamera robot. Pembuatan *web* ini menggunakan media yang umum digunakan untuk membuat halaman *web* seperti PHP, HTML, dll. *Web*

server memiliki halaman yang berisi tombol gerak robot dan tampilan baca kamera dari robot serta tambahan peta lokasi keberadaan robot.

3. Gerakan robot beroda meliputi gerak maju, mundur, belok kanan, belok kiri, dan berhenti.

1.4 Tujuan dan Manfaat Penelitian

Tujuan penelitian ini adalah mengoptimalkan jarak kendali robot pada penelitian sebelumnya oleh Pangestu [2] dengan jarak maksimal sekitar 90 meter, sehingga menjadi robot yang dapat dikendalikan jarak jauh serta bertukar data dengan media *web server* tanpa terbatas jarak jangkauan.

Manfaat pada penelitian ini adalah mengimplementasikan sistem robot yang tak terbatas jangkauan kendali selama terdapat koneksi internet dan sebagai aplikasi terapan di berbagai bidang baik pendidikan dan keamanan yang menggunakan media rekayasa elektronik.

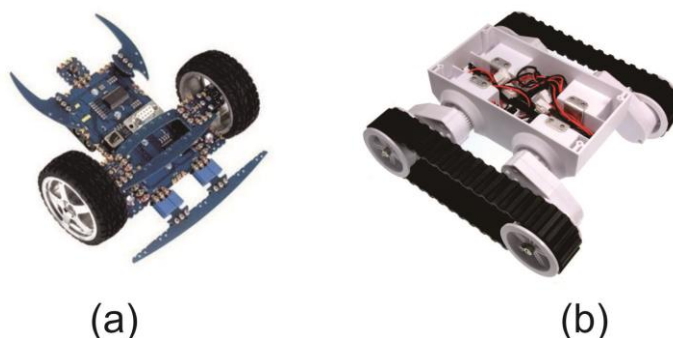
BAB II

TINJAUAN PUSTAKA

2.1 Robot Beroda

Robot beroda merupakan sebuah robot yang dapat bergerak dengan leluasa karena memiliki alat gerak untuk berpindah posisi. Teknik menggunakan roda adalah teknik tertua, paling mudah, dan paling efisien untuk menggerakkan robot melintasi permukaan datar. Roda dipilih karena memberikan tenaga tarik (*traction*) yang bagus, mudah diperoleh dan dipakai, serta mudah dipasang pada robot [3].

Tenaga tarik (*traction*) merupakan variabel dari material roda dan permukaan yang dilintasi oleh roda. Material roda yang lebih lembut memiliki koefisien tenaga tarik (*traction*) yang besar. Koefisien tenaga tarik (*traction*) yang besar ini memberikan gesekan yang besar, sehingga memperbesar daya yang dibutuhkan untuk menggerakkan motor. Jumlah roda yang digunakan pada robot beragam sesuai dengan selera pembuat robot. Robot dapat dibangun dengan menggunakan berbagai macam roda, misalnya beroda dua, beroda empat, beroda enam, atau beroda *caterpillar*.



Gambar 2. 1 (a) Robot beroda dua, (b) robot beroda *caterpillar* [16].

2.2 Perangkat Lunak Android

Android adalah sebuah sistem operasi untuk perangkat lunak *mobile* berbasis Linux yang mencakup sistem operasi, *middleware*, dan aplikasi.

Android merupakan *platform mobile* yang memberikan kesempatan kepada pengembang untuk melakukan pengembangan. Sistem operasi yang mendasari Android merupakan lisensi di bawah naungan GNU yang biasa dikenal dengan istilah *copyleft*, istilah ini merupakan lisensi perbaikan oleh pihak ketiga.

Pengembang memiliki beberapa pilihan dalam membuat aplikasi Android. Aplikasi yang sering digunakan adalah Eclipse dan Android Studio sebagai IDE (*integrated development environment*) untuk merancang aplikasi Android. Hal tersebut dikarenakan Eclipse dan Android Studio mendapat dukungan langsung dari Google untuk menjadi IDE pengembangan aplikasi Android.

Android dapat digunakan sebagai kontroller untuk perangkat keras dengan komunikasi yang sudah tersedia pada perangkat, sebagai contoh komunikasi *bluetooth* dan *Wi-Fi* untuk diterjemahkan *mikrokontroller* sebagai eksekutor. Android memiliki berbagai peranti di dalamnya berupa kamera, *speaker*, mikrofon, GPS, *gyroscope*, *accelerometer* dan berbagai media komunikasi yang dapat diakses di dalamnya, sehingga sebuah sistem Android sudah mewakili kontroller, komunikasi, dan sensor.

2.2.1 Bagian-bagian dalam Aplikasi Android

Bagian dalam aplikasi Android mencakup berbagai macam komponen yang saling terkait, sehingga komponen satu tidak bisa berjalan tanpa komponen lainnya. Aplikasi Android ditulis dalam bahasa pemrograman java. *File* java di *compile*

dengan data *resources* yang dibutuhkan oleh aplikasi. Proses *compile* menghasilkan sebuah *file* berekstensi *.apk*. *File* ini yang disebut dengan aplikasi dan nantinya dapat dipasang dan dijalankan pada perangkat Android.

Komponen utama pada aplikasi Android adalah :

1. *Activities*

Activity merepresentasikan sebuah layar dengan *user interface*. Sebagai contoh sebuah aplikasi email memiliki satu *activity* yang menunjukkan daftar email baru, *activity* lain untuk menulis email, dan *activity* lain untuk membaca email. Jika aplikasi ini memiliki lebih dari satu *activity*, maka salah satu dari mereka harus ditandai sebagai *activity* yang disajikan saat aplikasi diluncurkan.

Sebuah *activity* diimplementasikan sebagai *subclass* dari *class activity* sebagai berikut :

```
public class MainActivity extends Activity {  
}
```

Gambar 2. 2 Implementasi *subclass* dari *activities*.

2. *Service*

Service tidak memiliki GUI dan berjalan pada *background*, sebagai contoh dalam memainkan musik atau mengambil data dari jaringan, *service* tersebut harus berada dalam kelas induknya. Misalnya *media player* sedang memutar sebuah lagu dari daftar lagu yang ada, aplikasi ini akan memiliki dua atau lebih *activity* yang memungkinkan *user* untuk memilih lagu misalnya, atau menulis pesan bersamaan dengan *media player* yang sedang berjalan.

Sebuah *service* diimplementasikan dengan *subclass* dari *service class* sebagai berikut :

```
public class MyService extends Service {
```

Gambar 2. 3 Contoh implementasi *subclass service class*.

3. *Broadcast Reciever*

Komponen ini berfungsi menerima dan menyampaikan pemberitahuan. Contoh *broadcast* adalah pemberitahuan zona waktu telah berubah, baterai lemah, gambar telah selesai diambil oleh kamera, dll. Komponen ini tidak memiliki UI tapi memiliki sebuah *activity* untuk merespon informasi yang diterima atau menggunakan *notification manager* untuk memberi tahu kepada pengguna seperti lampu latar, suara, dan getaran.

Broadcast reciever diimplementasikan sebagai *subclass* dari *class broadcast reciever* dan setiap pesan di siarkan sebagai *intent object*.

```
public class MyReceiver extends BroadcastReceiver {
}
```

Gambar 2. 4 Implementasi *subclass broadcast reciever*.

4. *Content Provider*

Komponen ini membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam *file* sistem seperti basis data. Komponen ini menyediakan cara untuk mengakses data yang dibutuhkan oleh sebuah *activity*. misalnya ketika menggunakan aplikasi yang membutuhkan peta atau aplikasi yang membutuhkan untuk mengakses data kontak dan navigasi.

Content provider diimplementasikan sebagai *subclass* dari *class content provider* dan harus menerapkan satu set standar API (*application programming interface*) yang memungkinkan aplikasi lain untuk melakukan akses [4].

```
public class MyContentProvider extends ContentProvider {
}
```

Gambar 2. 5 Implementasi *subclass content provider*.

2.3 Google Cloud Messaging (GCM)

GCM (*google cloud messaging*) adalah sebuah layanan yang dapat membantu seorang pengembang aplikasi Android untuk mengirim pesan dari *server* ke aplikasi Android. Layanan GCM menangani semua aspek antrean pesan dan pengiriman ke target aplikasi Android yang sedang berjalan pada perangkat target [5].

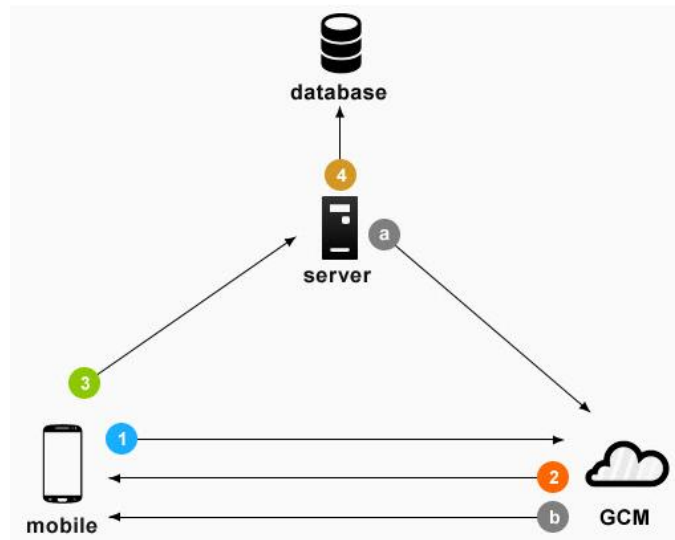
Layanan dari *Google console developer* yang digunakan pada penelitian ini meliputi *Google Cloud Messaging for Android* dan *Google Maps JavaScript API v3*. *GCM for Android* digunakan dalam mengirim pesan, dari pesan tersebut dilanjutkan Android untuk memberikan perintah ke *mikrokontroller* sebagai kendali gerak robot. *Google Maps JavaScript API v3* digunakan sebagai media untuk membuka dan menggunakan layanan Google Maps pada *web server*.

2.3.1 Cara Kerja Google Cloud Messaging (GCM)

GCM memiliki sistem kerja berupa pesan ringan yang memberi tahu aplikasi Android bahwa ada data baru yang harus diambil dari *server* (misalnya film atau gambar yang baru saja singgah) atau berupa pesan berisi 4 KB muatan data

(agar aplikasi seperti pemesanan instan dapat langsung mengonsumsi pesan) [6].

Cara kerja GCM dapat dilihat pada Gambar 2.6.



Gambar 2. 6 Proses pengambilan data pada GCM [17].

Penjelasan :

1. Perangkat Android mengirimkan *id* aplikasi ke server GCM untuk registrasi,
2. Setelah registrasi berhasil, selanjutnya server GCM menerbitkan *id* registrasi kepada perangkat Android,
3. Setelah mendapatkan *id* registrasi, selanjutnya peranti Android mengirimkan *id* registrasi ke server,
4. Server menyimpan *id* registrasi pada database untuk penggunaan selanjutnya,
 - a. Setiap kali notifikasi *push* dibutuhkan, server akan mengirimkan pesan ke server GCM beserta *id* registrasi perangkatnya,

- b. *Server* GCM mengirimkan pesan ke perangkat Android yang bersangkutan menggunakan *id* registrasi.

Berikut adalah karakteristik utama dari GCM :

- Memungkinkan *server* aplikasi pihak ke-3 untuk mengirim pesan ke aplikasi Android mereka.
- Sebuah aplikasi Android tidak perlu berjalan terus-terusan untuk menerima pesan. Sistem akan menjalankan aplikasi melalui *intent broadcast* ketika pesan masuk, selama aplikasi diatur dengan *broadcast reciever* dan perizinan yang tepat.
- GCM hanya melewati pesan data mentah yang diterima langsung ke aplikasi Android.
- GCM membutuhkan perangkat Android versi 2.2 atau yang lebih tinggi yang juga memiliki Google Play, atau *emulator* yang menjalankan Android 2.2 dengan API Google.

Pada GCM dikenal dengan *key concepts*, hal ini merangkum istilah kunci dan konsep yang terlibat dalam GCM. *key concepts* dibagi menjadi dua kategori:

1. *Components*

Components adalah entitas yang memainkan peran utama dalam GCM. komponen ini dibagi menjadi tiga bagian yaitu:

a. *Client App*

GCM hanya bisa bekerja pada perangkat Android dengan versi 2.2 ke atas dan memiliki satu akun Google yang aktif.

b. *3rd-party Application Server*

Aplikasi server yang ditulis sebagai bagian dari implementasi GCM. Server aplikasi pihak ke-3 mengirimkan data ke aplikasi Android pada perangkat melalui koneksi server GCM.

c. *GCM Connection Servers*

Server Google yang disediakan terlibat dalam mengambil pesan dari server aplikasi pihak ke-3 dan mengirim data ke perangkat Android.

2. *Credentials*

a. *Sender ID*

ID pengirim digunakan dalam proses pendaftaran untuk mengidentifikasi server aplikasi pihak ke-3 yang diizinkan untuk mengirim pesan ke perangkat Android.

b. *Application ID*

Aplikasi Android diidentifikasi dengan nama paket dari *manifest*. Hal ini memastikan bahwa pesan-pesan yang ditargetkan untuk aplikasi Android benar.

c. *Registration ID*

Registration ID adalah ID yang dikeluarkan oleh server GCM untuk aplikasi Android untuk menerima pesan. Setelah aplikasi Android memiliki ID pendaftaran, ia mengirimkannya ke server aplikasi pihak ke-3, yang menggunakannya untuk mengidentifikasi setiap perangkat yang telah terdaftar untuk menerima pesan untuk aplikasi Android yang diberikan, dengan kata

lain ID pendaftaran terkait dengan aplikasi Android tertentu yang berjalan pada perangkat tertentu.

d. *Google User Account*

Akun Google diperlukan untuk bisa menjalankan GCM, jika perangkat menjalankan versi Android lebih rendah dari Android versi 4.0.4.

e. *Sender Auth Token*

Sender Auth Token adalah kunci API yang disimpan pada *server* aplikasi pihak ke-3 yang memberikan aplikasi *server* akses untuk layanan Google. Kunci API termasuk dalam *header* permintaan POST yang mengirimkan pesan [7].

2.4 Google Maps

Google Maps *Mapping Service* adalah sebuah *online tools* yang memberikan *user* berbagai fitur-fitur peta. Fitur-fitur tersebut berupa tampilan *street map*, arahan kemudi *point-to-point*, posisi, dan jalur-jalur untuk mencari lokasi bisnis di berbagai kota. Dengan tambahan *street map* dan *terrain view*, *Satellite* atau *serial views* dapat memberikan tampilan yang mudah dipahami *user* dan dapat diakses siapa saja melalui *online connection*.

Untuk menampilkan peta dari Google Maps, diperlukan Google Maps API. Cara penggunaannya adalah memasukkan kunci API ke dalam *syntax* program yang akan digunakan. Penggunaan kunci API bisa dilihat pada Gambar 2.7.

```

<style type="text/css">
  html, body, #map-canvas { height: 100%; margin: 0; padding: 0;}
</style>
<script type="text/javascript"
  src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAeTxvMHA8aBC0hsHBe1_IVy4upk7mnx2E">
</script>
<script type="text/javascript">

```

Gambar 2. 7 Penggunaan kunci API untuk menampilkan Google Maps.

Google Maps memberikan sebuah tampilan berupa peta lokasi jalan atau *street map*. Pada Google Maps terdapat sebuah tanda untuk mengetahui lokasi saat diakses, tanda ini dinamakan dengan *marker*. Berikut ini adalah tampilan dari layanan Google Maps.



Gambar 2. 8 Tampilan layanan google maps.

2.4.1 Cara Kerja “My Location” Google Maps

Salah satu layanan pada Google Maps adalah mengetahui posisi *user* yang mengakses layanan ini. Google Maps *My Location* bisa diakses dengan atau tanpa GPS.

Penentuan lokasi yang menggunakan GPS mengandalkan sinyal satelit, perangkat yang menggunakan GPS akan mentransmisikan sinyal dari satelit ke perangkat. GPS membutuhkan transmisi dari tiga satelit untuk mendapatkan informasi dua dimensi (lintang dan bujur), dan empat satelit untuk tiga dimensi

(lintang, bujur, dan ketinggian). Setelah didapatkan posisi *user* yang mengakses maka data tersebut dimasukkan ke dalam layanan Google Maps sehingga posisi *user* dapat diketahui.

Penentuan lokasi tanpa GPS mengandalkan informasi yang diberikan BTS yang sedang digunakan. *By default* menurut standar GSM yang dikeluarkan oleh ETSI, setiap MS (*Mobile Subscriber*) yang menggunakan *voice All*, *SMS*, dan *GPRS/data access* akan mengirimkan informasi berupa MCC (*Mobile country code*), MNC (*Mobile Network Code*), LAC (*Location Access Code*), CI (*Cell Id*). Jika keempat elemen digabungkan akan menjadi CGI (*Cell Group Information*).

Ketika perangkat melakukan akses data menggunakan layanan Google Maps, maka secara otomatis perangkat akan mengirimkan informasi CGI tersebut ke Google untuk di-*query* di basis data Google untuk mencari koordinat BTS yang sedang melakukan layanan pada perangkat yang digunakan. Selanjutnya Google Maps bisa mengetahui posisi perangkat yang sedang digunakan dengan estimasi beberapa meter dari BTS, estimasi tersebut berdasar pada kekuatan sinyal yang sedang diakses oleh perangkat *user*.

2.5 Web Server

Web Server adalah sebuah aplikasi *server* yang melayani permintaan HTTP atau HTTPS dari *browser* dan mengirimkannya kembali dalam bentuk halaman-halaman *web*. Halaman-halaman *web* yang dikirim oleh *web server* biasanya berupa *file-file* HTML dan PHP yang diparsing atau ditata oleh *browser* sehingga menjadi halaman-halaman *web* yang bagus dan mudah dibaca [8].

Server adalah pelayanan yang memiliki hak untuk mengatur. *Server* disebut sebagai pelayanan dikarenakan fungsi *server* secara keseluruhan adalah memberi layanan (*service*) kepada *client* yang saling terhubung satu sama lain dalam satu jaringan. Fungsi dari *server* adalah mengatur dalam memberikan hak akses terhadap *client* yang terhubung dengan *server* tersebut [9].

Web server dan sistem robot terhubung dalam satu jaringan internet. Jaringan tersebut menggunakan *cloud* milik Google *service*. Jaringan internet menjadikan kedua komponen dapat bertukar data tanpa terbatas jarak. Tugas *web server* pada penelitian ini adalah mengirim, mengambil, dan menampilkan data.

2.5.1 *Hyper Text Markup language (HTML)*

Hyper Text Markup language (HTML) adalah bahasa dari *Word Wide Web (www)* yang dipergunakan untuk menyusun dan membentuk dokumen untuk ditampilkan pada program *web browser*. HTML dapat disebut sebagai protokol yang digunakan untuk mentransfer data atau dokumen dari *web server* ke *browser*. HTML inilah yang menjadi dasar saat menjelajah internet dan melihat halaman *web server*.

2.5.2 *PHP*

PHP merupakan suatu bahasa pemrograman yang difungsikan untuk membangun suatu *website* dinamis. PHP menyatu dengan HTML, maksudnya adalah HTML digunakan sebagai pembangun atau fondasi dari kerangka *layout web*, sedangkan PHP difungsikan sebagai proses. Pemrograman PHP dikenal

sebagai pemrograman *server-side*, artinya untuk menjalankan PHP, wajib adanya *web server* [10].

2.5.3 Database MySQL

MySQL merupakan perangkat lunak basis data yang paling populer di lingkungan Linux, hal ini didukung karena MySQL paling cepat, dan jarang bermasalah dan tersedia di Windows.

Untuk menghubungkan PHP dengan MySQL digunakan `mysqli_connect()`, cara ini digunakan untuk melakukan uji dan koneksi kepada *server database* MySQL dengan sintaks:

```
$conn = mysqli_connect("host","username basis data","password", "nama basis data")
```

`$conn` adalah nama variabel penampung status hasil koneksi kepada *database*, `host` adalah nama host atau alamat *server database* MySQL, `Username` adalah nama *user* yang telah diberi hak untuk dapat mengakses *server database*, dan `password` adalah kata sandi untuk *username* agar dapat masuk ke dalam *database* [9].

Contoh penerapan MySQL pada *file* PHP dapat dilihat pada Gambar 2.9, berikut ini:

```
<?php
$conn = mysqli_connect('mysql.idhostinger.com', 'u719981931_super', 'arta241112', 'u719981931_jarum');

$lat=$_POST["lat"];
$lon=$_POST["lon"];
//untuk menjalankan query insert
mysqli_query($conn, "INSERT INTO location(lat,lon) value('$lat', '$lon')");
mysqli_close($conn);

?>
```

Gambar 2. 9 Contoh penerapan *database* pada *file* PHP.

2.6 IOIO-OTG untuk perangkat Android

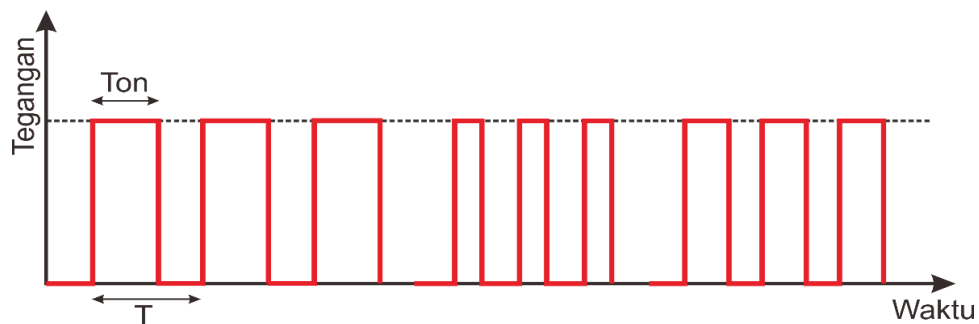
IOIO-OTG adalah sebuah modul *input/output* untuk peranti Android dan tablet. Hal ini memudahkan pengguna untuk menggunakan perangkat ini dengan mengakses melalui USB peranti Android. Pada IOIO dengan *firmware* terbaru pengguna dapat mengakses menggunakan modul *bluetooth* [11].

IOIO-OTG memiliki fungsi yang sama dengan *mikrokontroller* pada umumnya. Namun demikian, modul ini di desain khusus untuk Android sehingga dapat dihubungkan langsung ke peranti Android tanpa tambahan modul lainnya. Pada penelitian menggunakan dua fungsi IOIO-board yaitu *digital output* dan PWM (*Pulse Width Modulation*). *Digital output* digunakan dalam menentukan logika gerakan robot untuk maju, mundur, berbelok, dan berhenti. Selanjutnya, PWM digunakan dalam mengatur kecepatan motor yang menggerakkan robot.

Digital output adalah sebuah fungsi yang dimiliki semua *mikrokontroller* untuk mengeluarkan logika 1 atau 0 pada pin *mikrokontroller* yang dituju. Selanjutnya, sinyal PWM dapat diartikan sebagai sinyal kotak yang lebar pulsanya dapat diatur. Sinyal PWM ditunjukkan pada Gambar 2.10. PWM memiliki amplitudo yang tetap, namun lebar pulsa pada PWM bervariasi. Pada penelitian ini nilai PWM digunakan sebagai pengatur kecepatan putaran motor DC, semakin besar nilai PWM maka semakin cepat pula putaran motor DC.

Cara kerja PWM diatur dengan cara mengubah lebar pulsa atau T_{on} dalam satu periode gelombang. Pada penelitian ini PWM yang digunakan bernilai 120, 160, 162, dan 170 nilai tersebut adalah nilai T_{on} dan nilai T sebesar 256. Pada setiap nilai PWM tersebut didapatkan nilai *duty cycle* dengan rumus $\frac{T_{on}}{T} \times 100\%$. Untuk mendapatkan tegangan masukan pada motor DC, maka nilai *duty cycle* dikalikan

dengan *supply* tegangan. Sebagai contoh untuk menggerakkan robot maju dengan mengatur nilai PWM kanan dan kiri sebesar 160, sehingga didapatkan nilai *duty cycle* sebesar 62,5 % pada masing-masing keluaran pin PWM yang digunakan di IOIO-board. Selanjutnya pada *driver* motor DC disupply dengan tegangan 11,1 v, sehingga tegangan keluaran pada *driver* motor DC adalah $62,5 \% \times 11,1 \text{ v} = 6,9375 \text{ v}$. Nilai tegangan yang dikeluarkan oleh *driver* selanjutnya digunakan sebagai masukan ke motor DC untuk menggerakkan robot.



Gambar 2. 10 Sinyal PWM.

2.7 Penelitian Sejenis

Dari penelitian yang sudah ada sebelumnya beberapa di antaranya memiliki kesinambungan dengan penelitian ini, berikut adalah beberapa penelitian yang sudah ada:

Penelitian pertama oleh Alfian (2010), mengenai “*Web Server* Berbasis Embedded Ethernet Untuk Sistem Kendali Dan Monitoring Jarak Jauh Pada Ruang Penyimpan Barang-Barang Berharga”. Penelitian ini menggunakan *mikrokontroller* ATmega16 dan sebuah modul *ethernet controller* tipe NM7010 A-LF sebagai kontrol jaringan yang menangani komunikasi menggunakan protokol TCP/IP. Penelitian ini digunakan untuk sistem kendali dan monitoring suhu serta keamanan ruang menggunakan media *web server* [12].

Penelitian Selanjutnya oleh Pangestu (2014), mengenai “Pengendali Robot Jarak Jauh Berbasis *Smartphone* Android”. Pada penelitian ini digunakan robot dengan pengendali *smartphone* yang juga digunakan sebagai media penampil data baca sensor kamera yang *built-in* pada *smartphone* Android dengan proses *streaming*. Media yang digunakan untuk transfer data menggunakan akses poin *Wi-Fi*. *Frame size* dan *frame rate* yang digunakan pada kamera berpengaruh terhadap besar *bandwidth*, karena semakin tinggi *frame size* dan *frame rate* berbanding lurus dengan besar *file* yang dikirimkan [2].

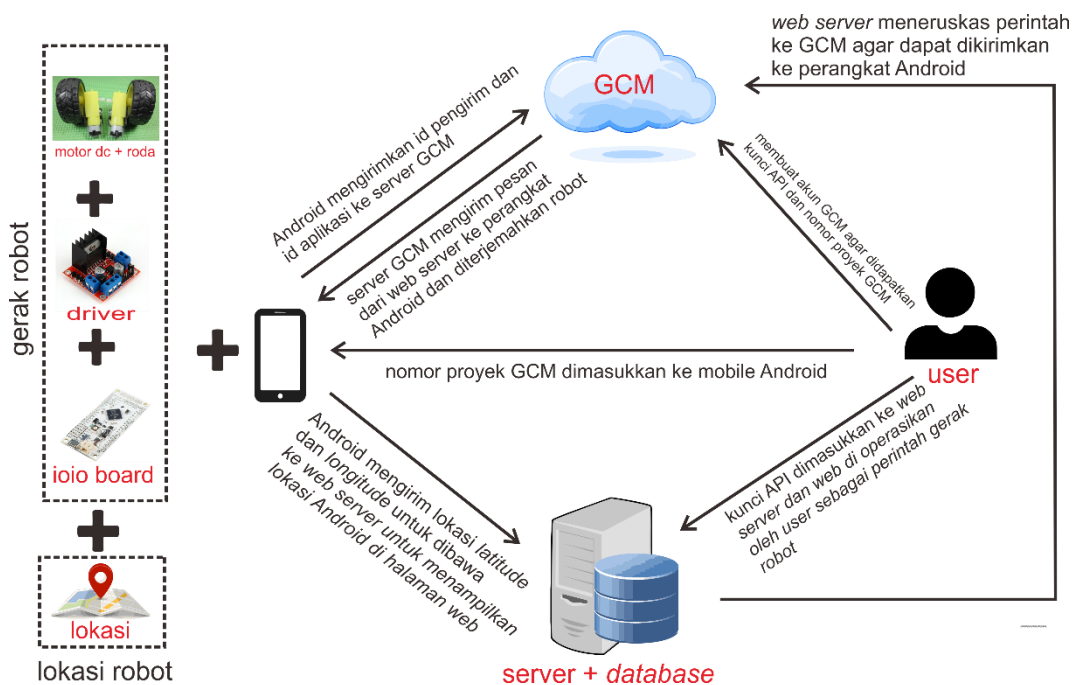
Penelitian yang ketiga oleh Karomah (2014), mengenai “Rancang Bangun Sistem kendali jarak jauh *on/off* lampu dan air conditioner (AC) berbasis Arduino melalui internet”. Pada penelitian ini digunakan Arduino Uno sebagai kontroller dan Ethernet sebagai modul untuk menambahkan fungsi LAN/Ethernet. Sistem pengendalian yang digunakan adalah kendali *on/off* untuk menyalakan dan mematikan lampu dan AC [13].

Penelitian yang keempat oleh Gorat (2013), mengenai “Perancangan Sistem Pemantauan Energi Listrik Pada Bangunan Secara Waktu Nyata Berbasis Web”. Pada penelitian ini digunakan mikrokontroller Arduino, modul Ethernet, sensor arus SCT-013-030 dan sensor tegangan listrik MW41-0901000UA. Data dari masing-masing sensor disimpan pada komputer server untuk ditampilkan pada sebuah *web* sebagai antar muka pengguna [14].

BAB III

PERANCANGAN SISTEM

Perancangan sistem pengendali robot jarak jauh berbasis *web server* meliputi perancangan *hardware*, *software*, dan *web*. Sistem ini bertujuan agar robot dapat dikendalikan dari jarak jauh dengan sambungan internet menggunakan antarmuka berupa halaman *web*. Rancangan sistem bisa dilihat pada Gambar 3.1.



Gambar 3. 1 Diagram rancangan sistem kendali jarak jauh robot beroda berbasis *web server*.

Sistem robot yang digunakan adalah sistem robot beroda. Robot terdiri dari *body robot*, IOIO-board, driver motor DC, dua motor DC, dan modul *bluetooth*. Robot digerakkan oleh IOIO-board yang terhubung melalui perangkat Android. IOIO-board ini menerjemahkan perintah yang diberikan oleh perangkat Android melalui komunikasi *bluetooth*, selanjutnya IOIO-board mengirimkan

sinyal berupa *digital output* dan sinyal PWM ke aktuator untuk membentuk arah dan gerakan robot beroda.

Sistem robot beroda terdiri dari robot beroda yang terhubung dengan *smartphone* Android. Peranti Android yang digunakan wajib memiliki GPS, *bluetooth*, data internet, kamera, dan *flash* kamera. GPS digunakan untuk mengirim lokasi robot secara *real time* kepada *web server* melalui akses internet dari *smartphone* tersebut. *Bluetooth* digunakan untuk komunikasi peranti Android dan *IOIO-board*. Data internet sebagai media komunikasi antara peranti Android dan *web server*. Kamera untuk mengambil foto dan menampilkan ke halaman *web server*. *Flash* kamera digunakan untuk menyalakan lampu *flash* yang berada pada Android. Peranti Android berperan sebagai penerima data yang dikirim oleh *web server*.

Web server berfungsi sebagai media antar muka untuk mengirim sinyal kendali gerak robot, menampilkan lokasi yang dikirimkan GPS Android secara *real time*, dan menampilkan foto yang diambil oleh peranti Android. Pengiriman data dari *web server* ke peranti Android merujuk ke alamat yang disediakan oleh *google service*. Alamat tersebut berupa *registration id* yang terdapat pada *web server* dan aplikasi Android yang tersimpan dalam basis data pada masing-masing komponen. Dengan demikian pengiriman data dari *web server* dapat sampai ke aplikasi Android yang dituju. Pertukaran data yang dikirimkan dari *web server* ke Android atau sebaliknya, melewati layanan *cloud* yang diberikan oleh *google service* sebagai penyedia. Layanan ini disebut GCM (*Google Cloud Messaging*), layanan ini pula menyediakan *server Google Maps* yang dapat diakses oleh perangkat Android dan *web server*. Layanan ini dapat digunakan dengan cara memasukkan kunci API dan

nomor proyek yang diberikan saat melakukan registrasi di akun *console google developer* pada masing-masing komponen baik di *web server* maupun di aplikasi Android.

Perancangan ini dibutuhkan empat komponen utama yaitu *sender*, *server*, *receiver*, dan *robot*. Pengirim atau *sender* adalah semua perangkat yang dapat mengakses koneksi internet yang memuat *web browser* untuk menjalankan UI yang tersedia dalam *web server*. *Web browser* yang digunakan adalah jenis *web browser* yang dapat memuat beberapa elemen yang dibutuhkan seperti HTML, PHP, CSS, JavaScript, dan Ajax. Semua elemen tersebut wajib dijalankan oleh *web browser* yang akan digunakan dalam pengiriman pesan kepada perangkat Android melalui GCM karena dalam perancangan *web server* terdapat elemen-elemen tersebut.

Komponen selanjutnya adalah *server*, *server* berfungsi sebagai penerima perintah dari *user*. *Server* yang digunakan adalah server dengan pemrograman PHP yang menggunakan *database* MySQL. Perintah yang dikirim *user* ke pada server selanjutnya diteruskan ke *receiver* menggunakan layanan GCM (*Google Cloud Messaging*).

Komponen selanjutnya adalah *receiver*, *receiver* berfungsi sebagai penerima perintah dari *web server*. *receiver* yang digunakan dalam perancangan ini adalah aplikasi Android. Aplikasi Android menggunakan pemrograman *mobile* java dan GCM sebagai layanan pertukaran data.

Komponen terakhir yang digunakan adalah *robot*. Robot dalam perancangan ini meliputi dua bagian yaitu bagian perangkat Android dan IOIO-board, kedua bagian ini tersambung melalui komunikasi *bluetooth*. IOIO-board tersambung dengan *driver* motor dan juga motor DC sebagai penggerak utama

robot. Ketika perangkat Android menerima perintah dari *server* maka perintah tersebut selanjutnya diteruskan atau diterjemahkan oleh Android ke *IOIO-board*. Dengan demikian, *IOIO-board* menggunakan perintah yang sudah diterima atau diterjemahkan oleh Android untuk menentukan arah gerakan robot ke mana robot tersebut harus bergerak.

Data kendali gerak robot yang diterima peranti Android dan data yang dikirim oleh peranti Android adalah jenis data *string*. Data yang dikirim peranti Android ke server berisi data *latitude* dan *longitude* posisi peranti Android berada. Jenis data *string* digunakan karena basis data yang digunakan berformat *varchar* atau karakter.

3.1 Perancangan Web Server

Perancangan *web server* meliputi pemrograman PHP, pembuatan basis data, dan melakukan *host* ke internet. Pemrograman *web* yang digunakan adalah PHP, untuk penulisan program digunakan Sublime Text 3 sebagai editor *syntax* PHP. Pembuatan basis data dilakukan pada Wamp Server sebagai aplikasi yang mengubah komputer menjadi *server* lokal. *Host* internet yang dipergunakan pada penelitian ini adalah *host* gratis milik idhostinger.com.

3.1.1 Sublime Text 3

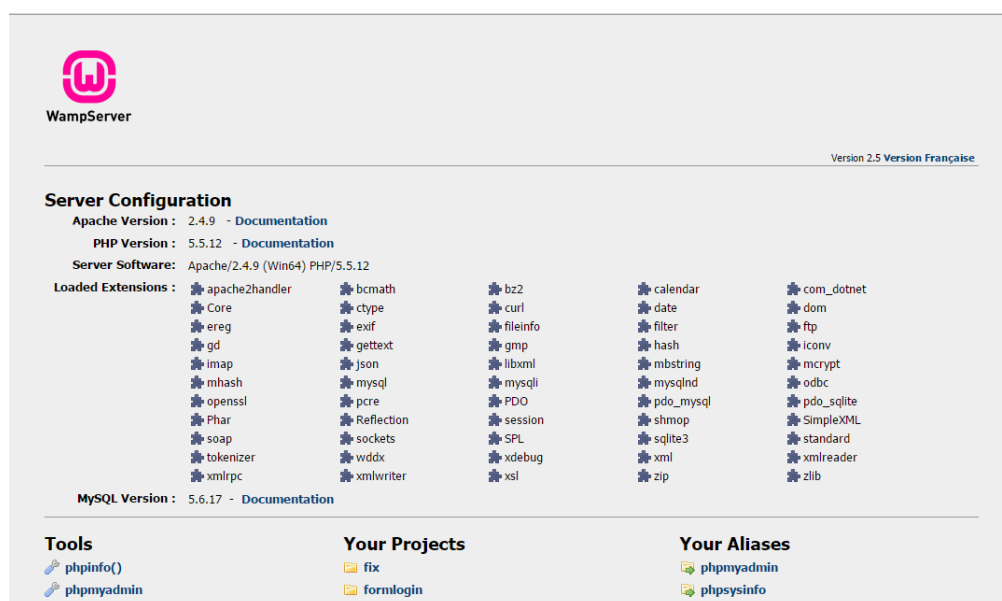
Sublime Text merupakan editor HTML, PHP, JavaScript, dll yang profesional mendesain, menulis kode program dan mengembangkan *website*, halaman *website*, dan aplikasi *web*. Di dalam penelitian ini Sublime Text digunakan untuk menuliskan tiga ekstensi bahasa pemrograman yaitu PHP, Java Script dan

CSS. Penggunaan HTML tidak digunakan karena PHP sudah bisa menggantikan HTML dan bisa dimasukkan fungsi JavaScript.

3.1.2 Wamp Server

Wamp server merupakan aplikasi yang dapat mengubah komputer menjadi sebuah *server* lokal. Wamp *server* merupakan wadah pengembangan *web* berbasis Windows, Wamp *server* ini mampu membuat aplikasi *web* dengan menggunakan Apache, PHP, dan MySQL *database*. Beberapa keuntungan dari Wamp *server* ini adalah cepat waktu instalasi, mudah untuk diatur, bisa di instalasi pada *flash drive*, gratis dan memerlukan spesifikasi komputer yang kecil.

Tampilan dari Wamp Server bisa dilihat pada Gambar 3. 2.



Gambar 3. 2 Tampilan gambar *localhost* menggunakan wamp server.

Tampilan yang ada di Wamp akan sama dengan tampilan yang dihasilkan setelah semua kode program dan basis data di *host* ke penyedia layanan. Penelitian

ini menggunakan layanan *host* milik idhostinger.com sehingga tampilan setelah di *host* bisa dilihat pada Gambar 3. 3.



Gambar 3. 3 Tampilan *web server* setelah di *host* ke internet.

Tampilan dari *web server* yang dibuat dalam penelitian ini meliputi tiga hal yaitu gerak navigasi beserta tombol fungsi, tampilan hasil foto dari kamera Android, dan peta lokasi perangkat Android berada.

3.1.3 Gerak Navigasi Kendali dan Fungsi

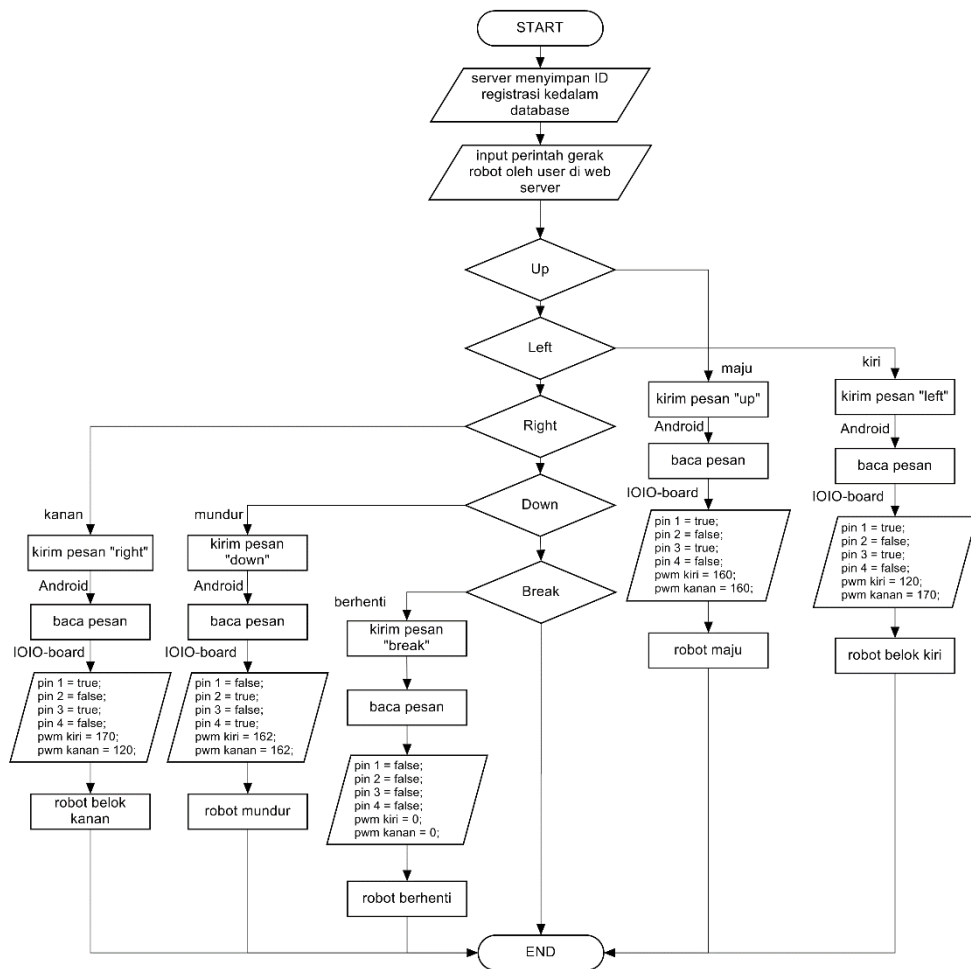
Gerak navigasi dan fungsi ini adalah sebuah tombol yang berfungsi untuk mengirimkan data *string* ke peranti Android. Data yang dikirim dari *web server* dikirimkan ke *cloud* Google dengan servis *google cloud messaging* GCM, selanjutnya data tersebut dikirimkan GCM ke target aplikasi Android yang memiliki *registration id* yang sama dengan milik *web server*.

Data yang diterima oleh Android selanjutnya dikirim Android ke *IOIO-board* dan diteruskan ke aktuator motor DC untuk menggerakkan robot maju, mundur, dan berbelok. Data yang diterima pula akan digunakan untuk membuka

fitur yang ada pada Aplikasi Android seperti pemutar suara, mengambil foto, menyalakan lampu *flash*, dan aktivasi lokasi robot berada untuk ditampilkan di halaman *web server*.

Metode yang digunakan untuk mengaktifkan gerak navigasi robot serta fungsi-fungsi tombol yang ada di *web server* agar bisa diterima dan difungsikan bisa dilihat pada Gambar 3. 4 dengan rincian penjelasan sebagai berikut:

1. Aplikasi Android yang dibuka akan mendaftarkan dirinya untuk mendapatkan *registration id* oleh GCM dan Android akan mengirimkan *registration id* tersebut ke *database* milik *web server* di tabel *regid*,
2. *Web server* yang sudah memiliki *registration id* tersebut sudah bisa melakukan pengiriman dan penerimaan data,
3. *User* melakukan *input* data gerak atau fungsi dengan mengklik tombol yang tersedia di tampilan *web server* untuk mengirimkan pesan ke Android,
4. Pesan yang dikirimkan *web server* dikirim ke GCM dan kemudian dikirimkan ke Android berupa pesan dengan format *string*,
5. Pesan yang diterima oleh Android diproses sesuai dengan jenis pesan yang masuk,
6. Setelah pesan gerak navigasi diterima Android, pesan tersebut dikirimkan ke *IOIO-board* melalui komunikasi *bluetooth*, contoh pesan tersebut adalah “up”, yang terkirim ke *IOIO-board* adalah pin 1 = *true*, pin 2 = *false*, pin 3 = *true*, pin 4 = *false*, pin 34 = *pwm* sebesar 160, pin 35 = *pwm* sebesar 160,
7. Tombol yang tersedia juga meliputi fungsi menyalakan *flash*, pemutar suara, dan pengambilan foto.



Gambar 3. 4 Diagram alir proses penerimaan perintah kendali gerak robot beroda melalui *web server* ke Android.

3.1.4 Hasil Foto dari Perangkat Android

Tampilan foto yang ada pada halaman utama *web server* adalah foto yang diambil oleh peranti Android setelah mendapatkan pesan perintah “take” dari *web server*. Perintah tersebut diberikan *user* dengan mengklik ikon kamera () pada *web server*, selanjutnya foto yang diambil oleh peranti Android dikirimkan ke folder /uploads yang ada di host dengan nama upload.jpg.

Foto yang tersimpan dalam folder /uploads dengan format upload.jpg, selanjutnya akan ditampilkan ke halaman utama *web server*. Bilamana foto kembali

diambil, maka foto sebelumnya akan otomatis tergantikan oleh foto yang baru, namun foto yang sebelumnya akan terus disimpan pada perangkat Android.

Berikut adalah *syntax* untuk menampilkan foto yang sudah tersimpan pada direktori /uploads/upload.jpg di index.php.

```
<div id      ="photo">
<img src     ="uploads/upload.jpg"/>
</div>
```

3.1.5 Pembaharuan Peta Lokasi Android

Peta lokasi yang berada di halaman utama *web server* adalah layanan peta yang diambil dari Google Maps dengan menggunakan servis dari Google Maps JavaScript API v3 yang ada di layanan Google *console developer*.

Layanan peta yang digunakan ini menggunakan *latitude* dan *longitude* untuk menunjukkan posisi perangkat Android berada saat ini. *Latitude* dan *logitude* ini otomatis di perbaharui setelah Android mendapatkan pesan “on” setelah *user* mengklik tombol ikon lokasi (📍).

Pembaharuan lokasi Android berada saat ini diambil dari basis data yang tersimpan dengan nama lat dan lon, lat untuk merujuk ke *latitude* dan lon untuk merujuk ke *longitude*. Lat dan lon ini dikirimkan oleh Android untuk mengisi basis data yang ada di *host web server* pada tabel “*location*”.

Berikut ini adalah gambar basis data *location* untuk membaharui lokasi keberadaan Perangkat Android yang digunakan.

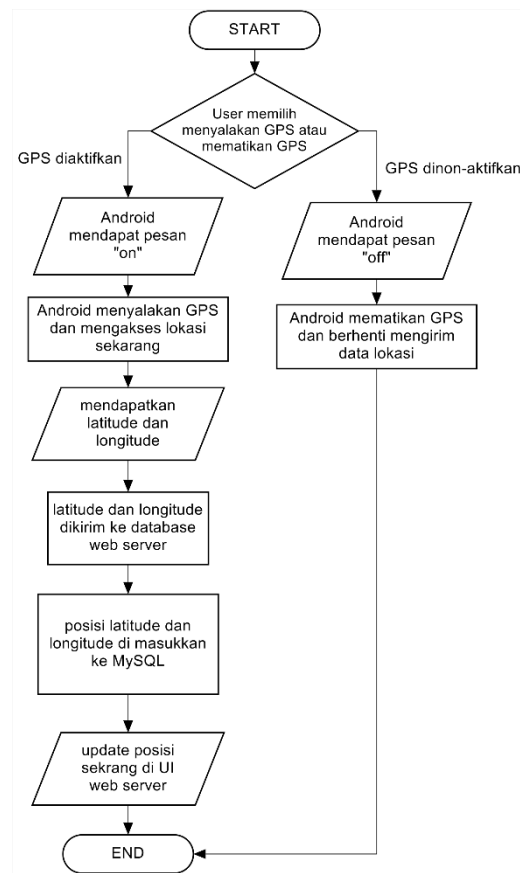
+ Opsi

				lat	lon	id
☐	Edit	Salin	Hapus	-7.687229	110.418448	17
☐	Edit	Salin	Hapus	-7.6968002	110.4206974	18
☐	Edit	Salin	Hapus	-7.6968002	110.4206964	19
☐	Edit	Salin	Hapus	-7.6968002	110.4206956	20
☐	Edit	Salin	Hapus	-7.6968002	110.420695	21
☐	Edit	Salin	Hapus	-7.6968002	110.4206945	22
☐	Edit	Salin	Hapus	-7.6968002	110.4206963	23
☐	Edit	Salin	Hapus	-7.6968001	110.4206979	24
☐	Edit	Salin	Hapus	-7.6968001	110.4206993	25
☐	Edit	Salin	Hapus	-7.6968002	110.4206987	26
☐	Edit	Salin	Hapus	-7.6968002	110.4206982	27

Gambar 3. 5 Basis data untuk menampilkan posisi keberadaan Android.

Metode yang digunakan untuk mengaktifkan lokasi bisa dilihat pada Gambar 3. 6 dengan penjelasan sebagai berikut:

1. *Web server* mengirimkan pesan “on”, pesan ini digunakan untuk mengaktifkan GPS dan mengambil lokasi Android,
2. Android mengakses lokasi saat ini untuk mendapatkan *latitude* dan *longitude*, selanjutnya data tersebut dikirimkan ke *web server* dan disimpan pada *database* di tabel *location*,
3. Android mengirimkan lokasi selama GPS aktif, pengiriman data lokasi ke *database* dilakukan setiap 1 detik sampai mendapatkan pesan “off”,
4. *Web server* melakukan pembaharuan lokasi saat ini dengan mengambil data lokasi yang dikirim Android di *database* tabel *location*,
5. Data yang diambil adalah data terakhir yang ada di tabel *location* untuk melakukan pembaharuan lokasi ke tampilan *web server*.



Gambar 3. 6 Proses pengambilan dan pengiriman data lokasi perangkat Android.

3.2 Perancangan Aplikasi Android

Perancangan aplikasi Android meliputi dua macam pemrograman, yang pertama adalah tampilan aplikasi Android dan selanjutnya adalah fungsi dari aplikasi Android tersebut.

Pemrograman Android terdapat dua bahasa pemrograman yang akan digunakan yaitu Java dan xml. Pemrograman bahasa java digunakan sebagai kode program dalam melakukan proses untuk menjalankan aplikasi Android. Pemrograman xml meliputi dua tahap, tahap satu adalah pemrograman xml *layout* atau tampilan aplikasi Android ketika dijalankan. Tahap kedua adalah

AndroidManifest.xml, *file* ini penting karena semua servis dan perizinan penggunaan layanan yang ada di Android diprogram di sini.

Pada pemrograman Android digunakan ADT resmi dari Google yaitu Android Studio. Android Studio berfungsi untuk menulis dan mendesain program Android menjadi sebuah aplikasi Android.

3.2.1 Tampilan Aplikasi Android

Aplikasi pada penelitian ini memiliki tampilan tunggal, tampilan ini ditulis di folder layout dengan nama *activity_my.xml*. Tampilan ini terdapat 3 macam komponen.

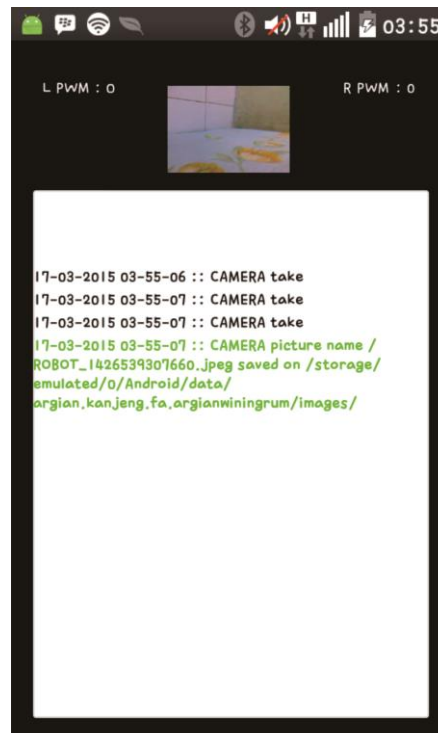
Komponen pertama adalah TextView, komponen ini digunakan untuk menampilkan nilai PWM kanan dan kiri yang akan dikirimkan Android ke *IOIO-board*.

Komponen kedua adalah ScrollView, komponen ini digunakan untuk menampilkan *log* perintah yang diterima oleh Android dari *web server*.

Komponen ketiga adalah FrameLayout, komponen ini digunakan untuk menampilkan gambar yang ditangkap oleh kamera Android. Komponen ini berfungsi untuk membuka sistem kamera Android dan mengambil foto untuk ditampilkan ke halaman utama *web server*.

Semua tampilan pada aplikasi Android ini bersifat pasif, karena aplikasi ini memiliki fungsi utama sebagai *receiver* dan tidak diberikan aksi langsung oleh *user*.

Tampilan aplikasi Android bisa dilihat pada Gambar 3. 7.



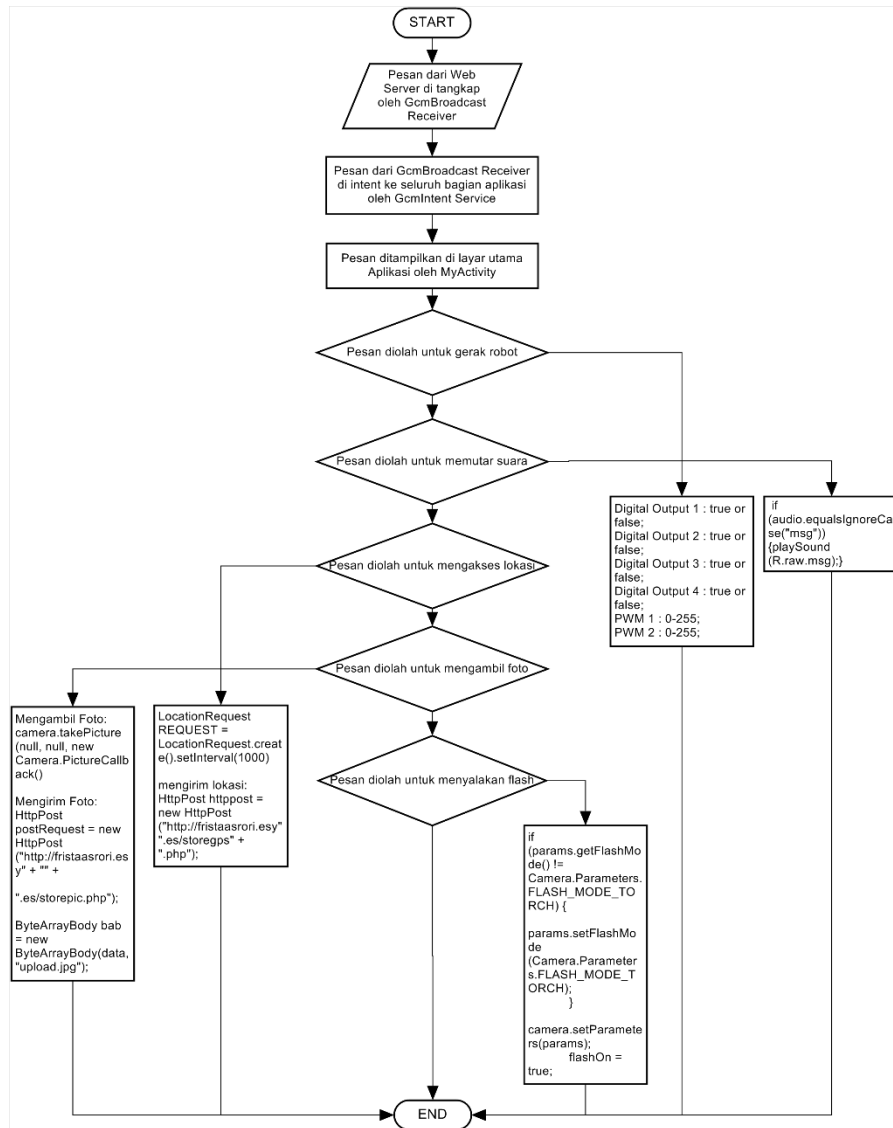
Gambar 3. 7 Tampilan aplikasi Android.

3.2.2 Proses Kerja Aplikasi Android

Setelah perancangan tampilan selesai, proses selanjutnya adalah pembuatan *syntax* java. *Syntax* ini berfungsi sebagai proses kerja dan fungsi-fungsi aplikasi Android. Penelitian ini memiliki beberapa proses kerja, proses kerja tersebut meliputi kendali navigasi robot, akses lokasi menggunakan GPS, pengambilan foto menggunakan kamera, pengaktifan *flash* kamera, dan pemutar suara. Cara kerja aplikasi Android dapat dilihat pada Gambar 3.8.

Proses kerja aplikasi Android dapat diuraikan dalam beberapa *class* java dalam aplikasi ini, masing-masing *class* tersebut adalah *MyActivity.java*, *LocationHelper.java*, *Constant.java*, *GcmBroadcastReceiver.java*, *Preview.java*, *registrar.java*, *CameraHelper.java*, *GcmIntentService.java*, dan tambahan

AndroidManifest.xml sebagai tempat perizinan penggunaan kamera, pemutar suara, GPS, dll.



Gambar 3. 8 Diagram alir aplikasi Android.

- Android Manifest.xml

AndroidManifest.xml adalah sebuah *file* bawaan ketika membuat proyek baru pada Android Studio. *File* ini menyajikan informasi kepada sistem Android tentang aplikasi yang dibuat, menginformasikan apa saja yang dibutuhkan sistem sebelum menjalankan apapun *code* yang ada pada aplikasi.

File ini dapat di modifikasi sesuai keperluan aplikasi yang dibuat dengan menambah *permission*, *reciever*, *feature*, dan *service* yang dibutuhkan dalam aplikasi. *AndroidManifest.xml* ini penting karena hal pertama yang dijalankan oleh aplikasi adalah *file* ini.

Penambahan *permission*, *reciever*, *feature*, dan *service* yang ada pada penelitian ini, antara lain:

1. *Permission*

- a. Internet
- b. *Wake Lock*
- c. *Get Account*
- d. *Access Network State*
- e. *Google Cloud Recieve*
- f. *Read Google Service*
- g. *Access Coarse Location*
- h. *Access Fine Location*
- i. *Camera*
- j. *Flashlight*
- k. *Read External Storage*
- l. *Write External Storage*
- m. *Bluetooth*

2. *Receiver = GcmBroadcastReceiver*

3. *Service = GcmIntentService*

4. *Feature*

- a. *Camera Flash*

b. *Hardware Camera*

AndroidManifest memiliki tugas untuk melakukan hal-hal berikut ini:

1. Menamakan paket aplikasi.
2. Mendeskripsikan komponen-komponen aplikasi seperti *activity*, *service*, *broadcast receiver*, dan penyedia *content* aplikasi.
3. Menentukan proses yang dijalankan komponen-komponen aplikasi.
4. Mendeklarasikan tingkat minimum dari API Android yang dibutuhkan aplikasi.

Pada penggunaan *permission*, *service*, *feature*, dan *receiver* dalam AndroidManifest.xml terdapat berbagai *tag*, *tag* tersebut antara lain `<uses-permission>`, `<application>`, `<activity>`, `<receiver>`, `<service>`, dan `<intent-service>`.

Tag `<uses-permission>` berfungsi untuk mendeklarasikan sebuah izin keamanan yang bisa digunakan untuk menetapkan komponen atau fitur dari aplikasi. Izin yang digunakan adalah izin untuk menggunakan akses internet, tetap terbangun untuk menerima data dari internet, akses lokasi dengan GPS, penggunaan kamera, *flash* kamera, akses *bluetooth*, dan membaca serta menulis pada penyimpanan Android.

Tag `<application>` berfungsi sebagai deklarasi dari aplikasi. Elemen ini mengandung sub elemen yang mendeklarasikan masing-masing komponen aplikasi dan memiliki atribut yang bisa mempengaruhi semua komponen.

Tag `<activity>` mendeklarasikan sebuah aktivitas yang mengimplementasikan bagian dari antarmuka pengguna aplikasi. Semua aktivitas harus direpresentasikan oleh elemen `<activity>`, aktivitas yang tidak dideklarasikan di dalam `AndroidManifest.xml` tidak akan dilihat oleh sistem dan tidak akan pernah berjalan.

Tag `<receiver>` mendeklarasikan sebuah penerima *broadcast* sebagai salah satu dari komponen aplikasi. Penerima *broadcast* memungkinkan aplikasi untuk menerima *intent* yang disiarkan oleh sistem atau aplikasi lainnya.

Tag `<service>` mendeklarasikan suatu servis sebagai salah satu komponen aplikasi. Servis tidak memiliki antarmuka visual, servis ini digunakan untuk mengimplementasikan operasi komunikasi API yang dipanggil oleh aplikasi lain.

Tag `<intent-filter>` menentukan tipe *intent* yang akan merespon sebuah aktivitas, servis atau penerima *broadcast*. Sebuah *intent-filter* mendeklarasikan kemampuan dari komponen induknya, aktivitas atau servis apa yang bisa dilakukan, dan tipe *broadcast* apa yang bisa ditangani oleh penerima.

- `MyActivity.java`

`MyActivity.java` adalah proses utama yang dijalankan aplikasi Android. `MyActivity.java` akan terhubung kepada semua *file class* lainnya. `MyActivity.java` mencakup *library IOIO-board* untuk melakukan pemrograman *mikrokontroller*.

File `MyActivity.java` menjadi tempat untuk meneruskan data yang diterima oleh `GCMIntentService.java` menjadi sebuah perintah untuk mengambil gambar, memutar musik, menyalakan lampu *flash* dan membuka GPS untuk mengirim lokasi keberadaan peranti Android.

- GcmBroadcastReceiver.java

File ini memiliki komponen *broadcast Receiver* yang berfungsi menangani interaksi sistem operasi Android dengan aplikasi lain. *Broadcast receiver* yang ada dalam penelitian ini bertugas untuk menerima pesan yang dikirimkan oleh *user* melalui tombol dari *web server* yang berisi pesan berformat *string*.

- GcmIntentService.java

Pada *file .java* ini memiliki *intent service*, *intent* merupakan *object* dari *package Android content*, *intent* bisa digunakan sebagai jembatan interaksi antar komponen dalam sistem operasi Android [15].

Intent biasa digunakan untuk interaksi antar *activity* dalam satu aplikasi maupun interaksi antar *activity* antar beberapa aplikasi. Pada penelitian ini *intent* digunakan untuk berinteraksi dengan *MyActivity.java* untuk mengambil pesan agar bisa diolah selanjutnya.

File ini juga digunakan untuk mengirimkan pemberitahuan berupa *notification* yang akan muncul pada jendela pemberitahuan yang terdapat di peranti Android. Pemberitahuan berupa pesan yang terkirim melalui *web server*. Metode yang digunakan ini sama dengan aplikasi perpesanan yang ada di Android.

- CameraHelper.java

Sesuai dengan namanya kelas ini bertujuan untuk membuka fungsi kamera dari sistem operasi Android. Kelas ini memiliki beberapa fungsi yaitu membuka kamera dari sistem, membuka *flash* dari sistem, mengambil foto hasil bidikan

kamera untuk disimpan di direktori penyimpanan Android, dan yang terakhir untuk mengirimkan *file* hasil bidikan ke direktori *web server* pada *host*.

- Preview.java

Kelas ini digunakan untuk menampilkan *preview* kamera yang dipinjam dari sistem Android ke dalam *FrameLayout*. Kelas ini terdapat beberapa fungsi lain, fungsi tersebut adalah *zoom in/out* kamera dengan cara menyentuh *preview* kamera dengan dua jari kemudian menggerakkannya mendekat untuk *zoom out* dan menjauh untuk *zoom in* secara bersamaan.

- LocationHelper.java

Kelas ini berfungsi untuk berkomunikasi dengan Google Api Client untuk mengakses peta lokasi keberadaan Android. Lokasi yang digunakan adalah lokasi hasil tangkapan GPS yang ada pada peranti Android, sehingga didapatkan *latitude* dan *longitude* saat diakses.

Lokasi di perbaharui setiap 1 detik, dengan memperbaharui lokasi terus menerus dan dikirimkan ke *web server* untuk di letakkan ke basis data.

- Registrar.java

Kelas ini adalah tempat untuk menaruh nomor proyek yang didapatkan saat mendaftar di *developer Google*, dari kelas ini *web server* spesifik mengirim pesan kepada aplikasi ini karena sudah terdaftar di *cloud* milik Google.

Cara kerja dari kelas ini yang pertama adalah melakukan pengecekan terhadap identitas dari aplikasi ini apakah sudah terdaftar di *cloud* Google atau belum, bila belum terdaftar maka aplikasi Google akan mengirimkan *registration*

id kepada aplikasi ini untuk disimpan di dalam aplikasi, *registration id* ini juga dikirimkan oleh aplikasi ke *web server* untuk disimpan ke basis data agar untuk mengidentifikasi target pesan akan dikirim dan diterima.

- Constant.java

Kelas ini bertujuan untuk mengklasifikasi jenis pesan dari *web server* yang masuk.

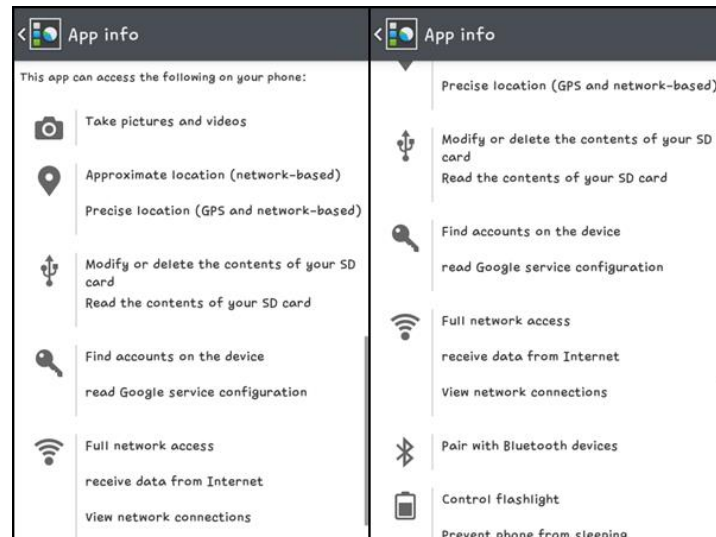
Kategori ini dibedakan dari 1 sampai 5, masing-masing kategorinya adalah navigasi dengan id 1, lokasi dengan id 2, *flash* dengan id 3, Kamera dengan id 4, dan suara dengan id 5. Kategori ini nantinya dibawa ke *MyActivity.java* untuk di proses dalam pengklasifikasian pesan yang masuk.

3.3 Perancangan Perangkat Keras pada Robot Beroda

Perancangan perangkat keras meliputi pemilihan perangkat keras yang digunakan dan menggabungkan menjadi sebuah sistem robot beroda. Bagian perangkat keras yang digunakan adalah *smartphone* Android, *IOIO-board*, baterai *Li-Po*, *driver* motor DC, dan motor DC.

Peranti Android yang digunakan pada sistem ini adalah LG L7 II P-713, dengan perangkat lunak Android versi 4.4.2, versi ini mendukung servis GCM, karena *requirement minimum* GCM hanya bisa digunakan oleh Android versi 2.2 ke atas. Perangkat Android digunakan sebagai pengirim dan penerima data masukan dari dan ke *web server*. Servis lainnya yang bisa digunakan oleh perangkat Android ini adalah *bluetooth*, GPS, kamera, *flash* kamera, dan speaker. *Bluetooth* digunakan untuk komunikasi dari Android ke *IOIO board* sehingga mengurangi penggunaan kabel dalam robot beroda. GPS digunakan dalam pengambilan *latitude*

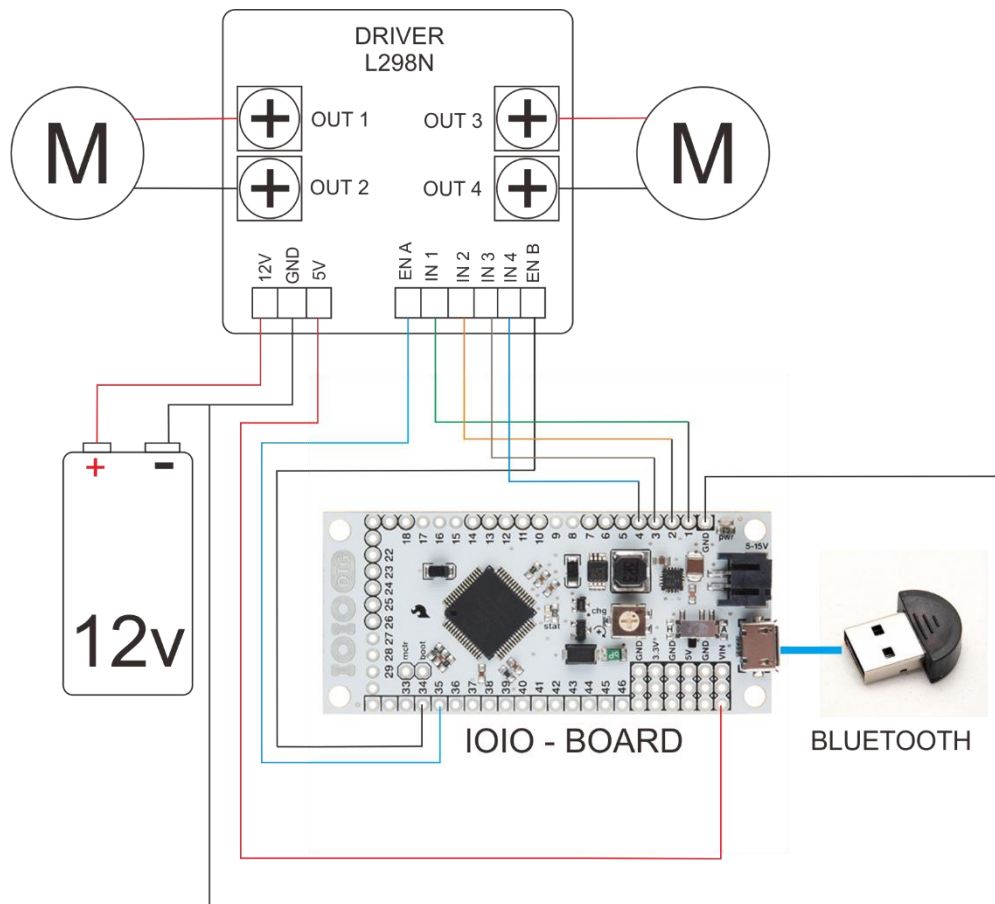
dan *longitude* lokasi robot diakses. Kamera dan *flash* kamera digunakan untuk mengambil foto dan menyimpan foto yang sudah diambil.



Gambar 3. 9 Servis yang dapat diakses oleh aplikasi Android di penelitian ini.

Perangkat keras yang lainnya adalah *IOIO-board* perangkat ini yang mengatur keluaran digital, gerak motor, dan komunikasi *bluetooth* untuk menerima data dari Android. Modul ini *compatible* dengan Android sehingga pemrograman *mikrokontroller* langsung pada Android tanpa ada pemrograman *on chip* seperti *mikrokontroller* yang lain.

Berikut ini adalah gambar skematik pada rangkaian perangkat keras robot beroda:



Gambar 3. 10 Rangkaian skematik robot beroda.

Untuk dapat menggerakkan motor DC dibutuhkan beberapa pin dari IOIO yang terhubung dengan *driver* motor. Pin tersebut digunakan sebagai masukan pada *driver* motor DC. Beberapa pin yang digunakan di sini adalah pin 1,2,3,4, dan 34, 35. Pin 1,2,3,4 terhubung dengan IN 1, IN 2, IN 3, IN 4, pin ini berfungsi mengeluarkan sinyal digital untuk mengatur gerak maju, mundur, belok kanan atau kiri. Selanjutnya pin 34 dan 35 masing-masing terhubung ke EN A dan EN B, pin ini berfungsi sebagai pin keluaran PWM sebagai pengatur kecepatan motor DC.

BAB IV

PENGUJIAN, ANALISIS DAN PEMBAHASAN

Penelitian ini memiliki 4 pengujian. Pengujian tersebut adalah pengujian perangkat keras robot, pengujian aplikasi Android, pengujian *web server*, dan pengujian sistem pada kondisi *indoor* dan *outdoor* menggunakan 3 operator jaringan internet yang berbeda.

Pengujian perangkat keras robot adalah pengujian yang dilakukan setelah robot selesai dirancang. Pengujian tersebut meliputi pengujian gerak robot, kecepatan robot, dan komunikasi robot ke aplikasi Android. Selanjutnya, Pengujian aplikasi Android adalah pengujian yang dilakukan pada komunikasi aplikasi ke robot, pengujian komunikasi ke *web server*, dan pengujian aplikasi Android setelah menerima data dari *web server*. Selanjutnya, Pengujian *web server* pada penelitian ini meliputi pengujian *off line* dan pengujian setelah di *host* atau pengujian *on line*. Semua pengujian *web server* menggunakan basis data baik pada pengujian *off line* maupun *on line*. Pengujian yang terakhir adalah pengujian keseluruhan sistem pada kondisi *indoor* dan *outdoor*. Pengujian ini dilakukan dengan menggunakan 3 operator yang berbeda yaitu operator A, B, dan C sebagai penyedia layanan internet peranti Android. Pengujian *indoor* dilakukan di dalam gedung FTI UII dengan jarak antara pengirim dan penerima data sekitar 100 meter. Kemudian pengujian *outdoor* dikenakan jarak yang sama dengan pengujian *indoor* yaitu sekitar 100 meter antara pengirim dan penerima data. Menurut kedua pengujian tersebut dianalisis bagaimana hubungan antara kecepatan transfer data terhadap jarak maksimal pengiriman data dari *web server* ke peranti Android.

4.1 Pengujian Perangkat Keras Robot Beroda

Pengujian perangkat keras robot beroda meliputi 3 pengujian, pengujian yang pertama adalah pengujian gerak robot. Pengujian ini dilakukan untuk melihat pergerakan robot baik maju, mundur, berbelok kanan atau kiri. Pengujian dilakukan dengan cara memberikan perintah langsung ke robot melalui peranti Android. Pengiriman data dilakukan dengan komunikasi *bluetooth* untuk bisa menghubungkan peranti Android dengan IOIO-board.

Pengujian berikutnya adalah pengujian kecepatan robot. Pengujian ini dengan cara menaruh perintah kecepatan PWM yang diletakkan di aplikasi Android. Kecepatan robot yang digunakan pada penelitian ini adalah kecepatan robot yang rendah, sehingga didapatkan nilai PWM sesuai dengan Tabel 4. 1.

Tabel 4. 1 Nilai PWM pada kecepatan gerak robot beroda.

	Nilai PWM	
	PWM kiri	PWM kanan
Maju	160	160
Mundur	162	162
Belok kanan	170	120
Belok kiri	120	170
Berhenti	0	0

Pengujian yang terakhir adalah komunikasi robot beroda ke aplikasi Android. Komunikasi yang digunakan adalah komunikasi *bluetooth* dari IOIO-board ke aplikasi Android. Antara IOIO-board dan aplikasi sudah tersambung digunakan indikator nilai PWM, sehingga dapat diketahui aplikasi sudah tersambung dengan IOIO-board atau belum.

Pada pengujian ini dihasilkan sebuah robot beroda yang dapat dikendalikan melalui komunikasi *bluetooth* oleh peranti Android. Pengujian ini juga

menghasilkan sebuah robot yang dapat bergerak maju, mundur, dan berbelok dengan kecepatan yang berdasar pada besar nilai PWM yang dikenakan padanya.

4.2 Pengujian Aplikasi Android

Pengujian aplikasi Android meliputi 3 pengujian. Pengujian yang pertama adalah pengujian komunikasi aplikasi ke robot beroda. Pengujian ini meliputi komunikasi mengirim data dari aplikasi Android ke sistem robot. Data yang dikirim akan dilihat apakah perintah bisa dieksekusi sesuai perintah oleh sistem robot atau tidak.

Pengujian selanjutnya adalah pengujian komunikasi aplikasi Android dengan *web server*. Komunikasi pada pengujian ini adalah komunikasi dua arah, yaitu aplikasi Android menerima dan mengirim data dari dan ke *web server*. Pengujian penerimaan data dilihat bahwa data yang dikirimkan oleh *web server* apakah sampai ke aplikasi Android dan sesuai dengan data yang dikirimkan *web server*. kemudian pengiriman data dari aplikasi Android ke *web server* disimpan ke basis data yang ada di *host*. Pengujian dilihat apakah data *registration id* dan data *latitude*, *longitude* sudah bisa terkirim ke basis data *web server* dan sesuai dengan data yang dikirim.

Pengujian yang terakhir adalah pengujian aplikasi Android setelah menerima data dari *web server*. Pengujian ini dilakukan setelah Android dapat menerima data dari *web server*, data tersebut diolah Android untuk melakukan eksekusi terhadap perintah yang diberikan padanya. Eksekusi tersebut adalah menyalakan *flash* kamera, mengambil gambar, memutar suara, dan mengaktifkan

GPS. Pengujian dilakukan satu per satu terhadap semua perintah sampai perintah mampu dieksekusi oleh aplikasi Android.

4.2.1 Pembahasan Aplikasi Android

Hal pertama yang dilakukan aplikasi Android ketika dibuka adalah mendaftarkan nomor proyek yang didapat dari console google developer. Pendaftaran ini digunakan untuk mendapatkan *registration id* yang akan dikirimkan ke basis data *web server* dan juga disimpan dalam basis data aplikasi. *Registration id* adalah *id* yang digunakan untuk mengirim dan menerima data dari dan ke target pengiriman dan penerimaan data tersebut.

Aplikasi yang sudah bisa menyambung ke *cloud* sudah siap untuk menerima pesan yang dikirim dari *web server*. Pesan yang masuk dikelompokkan menurut *id* masing-masing yang sudah ditentukan di kedua aplikasi baik Android maupun *web server*.

Sebelum aplikasi dibuka harus diaktifkan beberapa fitur dari Android yaitu *bluetooth*, GPS, dan data. *Bluetooth* digunakan untuk menyambungkan aplikasi ke *mikrokontroller* yang ada di robot, GPS digunakan untuk mengambil lokasi Android saat diakses dan data sebagai sarana komunikasi internet untuk menyambung aplikasi Android dengan *web server*.

Beberapa fungsi yang digunakan pada aplikasi Android ini adalah pemutar suara, pengambilan foto, dan penggunaan GPS untuk mencari lokasi Android berada. Fungsi ini akan aktif setelah menerima pesan yang dikirim oleh *web server*.

Fungsi pengambilan foto aktif setelah Android menerima pesan “take” dari *web server*. Foto yang diambil akan dikompres dengan level paling rendah agar

lebih cepat untuk mengirimkan *file* gambar yang diambil. Setelah *file* gambar diambil, kemudian *file* tersebut dikirimkan ke folder uploads dengan format upload.jpg. Bila ada pengambilan foto lagi maka foto sebelumnya yang ada di *web server* akan tergantikan dengan *file* foto yang baru. Foto yang diambil sebelumnya tetap disimpan di dalam direktori aplikasi Android.

Fungsi pengambilan data GPS aktif setelah Android menerima pesan “on” dari *web server* di dalam pengelompokan pesan di *array* 2. Lokasi yang dikirimkan Android di perbaharui setiap 1000ms atau 1 detik, jadi pengiriman data lokasi ke basis data dilakukan setiap 1 detik ketika fungsi GPS sudah diaktifkan.

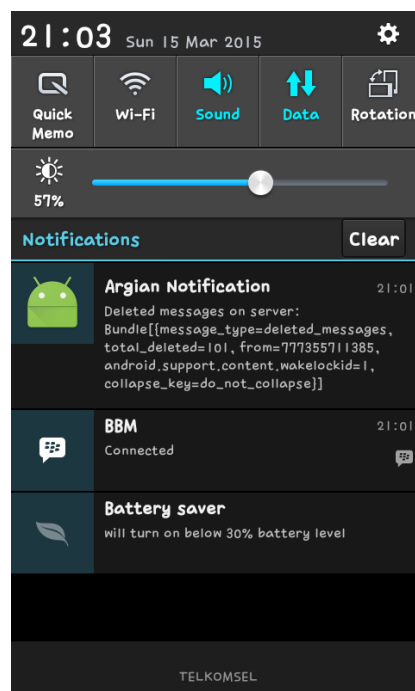
Fungsi pemutar suara pada Android aktif ketika pesan pada setiap jenis suara didapatkan, contoh suara akan aktif ketika menerima pesan “start” dari *web server*. dari pesan tersebut maka akan diteruskan untuk memutar *file* suara yang sudah disisipkan dalam aplikasi Android.

Fungsi gerak navigasi robot sudah dijelaskan dalam diagram alir pada Gambar 3.4, ketika Android sudah menerima pesan maka IOIO-board akan langsung mengirim sinyal perintah ke aktuator berupa motor DC, selanjutnya aktuator menjalankan perintah yang diterima oleh IOIO-board.

Data dikirim *user* melalui *web server*, pesan tersebut kemudian diteruskan ke Google sebagai penyedia layanan GCM. Pesan yang dikirimkan diantrekan oleh GCM untuk dikirimkan ke target aplikasi yang memiliki *registration id* yang sama oleh pengirim. Pesan tersebut akan diantrekan bila penerima belum mendapatkan pesan yang dikirim oleh *web server*.

Antrean pesan memiliki jangka waktu beberapa jam sebelum pesan dihapus oleh *server* GCM. Percobaan antrean pesan dilakukan dengan cara mengirim pesan

dari *web server* dan peranti Android di matikan setelah 12 jam. Akibat dari penundaan dan penumpukan pesan yang dikirimkan tanpa ada penerima maka pesan tersebut dihapus oleh GCM. Penundaan pesan tersebut bisa dilihat pada Gambar 4.1. Pesan yang ditunda dengan interval waktu 1 sampai 2 jam masih bisa dikirimkan dan masih bisa terbaca oleh target aplikasi.



Gambar 4. 1 Penghapusan pesan oleh server GCM.

4.3 Pengujian Web Server

Pengujian *web server* meliputi 2 pengujian. Pengujian yang pertama adalah pengujian *offline* atau pengujian dengan simulasi server lokal. Pengujian ini bertujuan untuk menyimulasikan *web server* sebelum di *host*. Pengujian dilakukan untuk mendapatkan hasil sebuah *web server* yang dapat membaca dan menampilkan data lokasi dari basis data lokal.

Pengujian selanjutnya adalah pengujian *web server* setelah di lakukan *host* atau pengujian *on line*. Pengujian ini bertujuan untuk membaca dan menampilkan data serta mengirimkan data ke aplikasi Android. Pembacaan dan penampilan data sama dengan cara lokal, namun cukup mengganti nama basis data, *host*, *username*, dan *password* yang sudah disediakan oleh *host*. Pengiriman pesan dilakukan pengujian sampai pesan yang dikirimkan oleh *user* bisa terbaca dan dieksekusi oleh aplikasi Android.

4.3.1 Pembahasan *Web Server*

Web server yang digunakan pada penelitian ini adalah layanan *web server* yang di *host* ke idhostinger.com dengan akun gratis. Diberikan dua buah sub domain, dua basis data MySQL, data transfer 100GB, kapasitas 200MB dan *support* PHP serta layanan FTP.

Layanan dari *web server* ini cukup handal meskipun diberikan oleh penyedia dengan bebas. Layanan akan lebih cepat bila menggunakan akun berbayar karena transfer data yang lebih besar pula. Layanan ini menggunakan sebuah basis data dalam menjalankan *web server*.

Basis data tersebut memiliki tiga buah tabel, masing-masing tabel adalah admin, location, dan regid. Tabel admin berfungsi untuk menaruh *user id* dan *password* untuk akses masuk ke halaman utama *web server*. Tabel location berfungsi sebagai tempat menaruh lokasi kiriman dari Android untuk memperbaharui peta lokasi keberadaan Android. Tabel terakhir adalah regid, tabel ini berfungsi sebagai tempat penyimpanan *registration id* yang dikirimkan oleh

Google kepada Android sebagai alamat pengiriman pesan dari *web server* ke target aplikasi.

Halaman utama *web server* memiliki 3 tampilan utama, tampilan yang pertama adalah tampilan foto yang diambil oleh aplikasi Android. Foto tersebut dikirimkan oleh aplikasi Android ke *folder* uploads milik *host web server* dengan nama *file* upload.jpg untuk ditampilkan di halaman utama *web server*.

Tampilan selanjutnya adalah tampilan peta lokasi yang diambil dari Google Maps. Peta lokasi yang ditampilkan adalah lokasi terakhir yang ada di basis data tabel *location* yang ada di *host* internet. Sehingga lokasi keberadaan Android bisa di *tracking* menggunakan data dari basis data tersebut. Data lokasi ini berisi *latitude* dan *longitude* tempat akses Android.

Tampilan yang terakhir adalah tombol navigasi gerak robot, lokasi, *flash*, kamera, dan suara. Fungsi dari tombol tersebut adalah mengirimkan pesan, setiap pesan memiliki pengelompokan tersendiri, sehingga pesan yang masuk ke peranti Android lebih tertata. Pengelompokan pesan itu sendiri dibagi menjadi 5 yaitu navigasi, lokasi, *flash* kamera, kamera, dan pemutar suara.

4.4 Pengujian Keseluruhan Sistem

Pengujian penelitian ini mencakup keseluruhan sistem pada kondisi *indoor* dan *outdoor*. Pengujian dilakukan dengan menggunakan 3 operator yang berbeda sebagai penyedia layanan internet peranti Android. Pengujian *indoor* dilakukan di dalam gedung FTI UII dengan jarak antara pengirim dan penerima data sekitar 100 meter. Kemudian pengujian *outdoor* dilakukan di luar gedung FTI UII dan dikenakan jarak yang sama dengan pengujian *indoor* yaitu sekitar 100 meter antara

pengirim dan penerima data. Jarak pada pengujian *indoor* maupun *outdoor* dengan kisaran 100 meter mengacu pada sampel jarak, namun jarak sesungguhnya pada sistem ini tidak terbatas selama mendapatkan akses internet pada masing-masing komponen. Menurut kedua pengujian tersebut, selanjutnya dianalisis bagaimana hubungan antara kecepatan transfer data terhadap jarak pengiriman data dari *web server* ke peranti Android.

Pengujian yang dilakukan memiliki karakteristik yang berbeda pada setiap pengujian, tergantung pada operator penyedia layanan jaringan internet. Untuk memastikan pengujian terbaik dibutuhkan *baseline* sebagai pembanding pada setiap pengujian. *Baseline* yang digunakan meliputi satu unit *laptop* yang dapat mengakses jaringan internet dengan lancar, kecepatan rata-rata pada *laptop* sekitar 150 KB/detik dan layanan internet yang digunakan Android adalah HSDPA atau menggunakan jaringan *Wi-Fi* dengan kecepatan rata-rata sekitar 150 KB/detik. *Baseline* tidak membatasi jarak akses, selama ketersediaan akses internet pada masing-masing peranti memadai maka pengiriman dan penerimaan data bisa berjalan dengan lancar. Rata-rata waktu tunda transfer data pada *baseline* sekitar 2 detik.

Pengujian ini menggunakan *laptop* sebagai pengirim pesan ke peranti Android. *Laptop* yang digunakan tersambung dengan jaringan *Wi-Fi* FTI UII. Pengujian ini dilakukan pada malam hari sehingga penggunaan jaringan *Wi-Fi* FTI-UII untuk mengakses kecepatan internet yang lebih maksimal. Hal ini dikarenakan pengguna jaringan ini cukup sedikit. Adapun spesifikasi *laptop* yang digunakan bisa dilihat pada Tabel 4.2. Peranti Android yang digunakan pada pengujian ini bisa

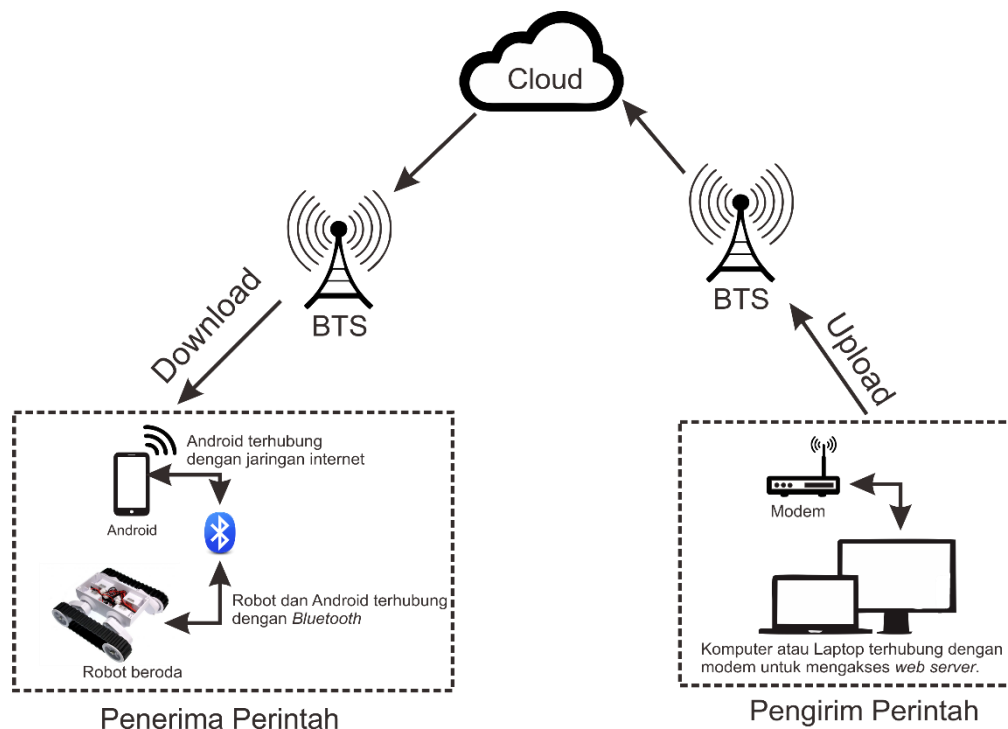
dilihat pada Tabel 4.3, dengan akses internet menggunakan 3 operator yang berbeda.

Tabel 4. 2 Spesifikasi laptop yang digunakan untuk pengujian.

Spesifikasi Laptop	
Jaringan	Wi-Fi a/b/g/n
Memori	4 GB
Prosesor	Intel Core-i5-460M
Grafik	Intel HD Grafik
Kecepatan Prosesor	2.5 GHz
OS	Windows 7 x64 SP1

Tabel 4. 3 Spesifikasi Android yang digunakan untuk pengujian.

Spesifikasi Peranti Android	
Jaringan	GSM/EDGE/3G/HSDPA
Memori	4GB, RAM 768 MB
OS	Android v4.4.2 API 19
CPU	Dual Core Qualcomm Snapdragon 200
Arsitektur	2x ARM Cortex-A5
Kecepatan Prosesor	1.01 GHz

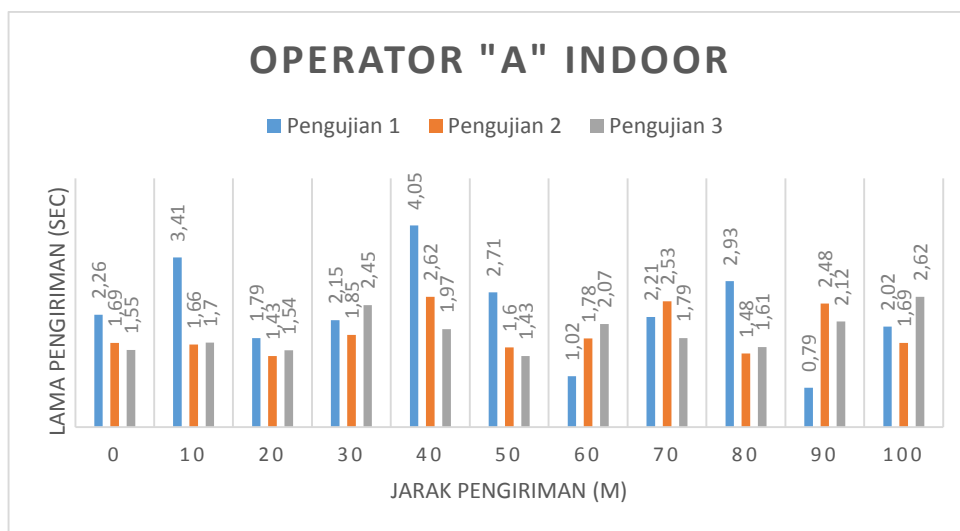


Gambar 4. 2 Proses pengiriman data dari pengirim (*laptop*) ke robot beroda.

4.4.1 Pengujian Indoor

Pengujian dilakukan untuk mengamati lama pengiriman pesan dari *web server* ke peranti Android dengan jarak 0 sampai 100 meter dengan interval jarak 10 meter. Pengujian ini dilakukan 3 kali dengan cara yang sama pada setiap pengujian. Pada pengujian ini digunakan 3 operator yang berbeda yaitu operator A, B, dan C. Berikut ini adalah hasil dari pengujian *indoor*. Pengujian *indoor* pada operator A, layanan internet yang diperoleh adalah HSDPA. Hasil pengujian bisa dilihat pada Gambar 4.3.

Dari grafik pada Gambar 4.3 dapat dilihat bahwa lama pengiriman pesan dari *web server* ke Android bervariasi. Pada grafik tersebut ditunjukkan bahwa jarak bukanlah pengaruh utama pada kecepatan penerimaan pesan. Rata-rata total kecepatan yang dihasilkan pada pengujian *indoor* dengan menggunakan operator A adalah 2,0302 detik.

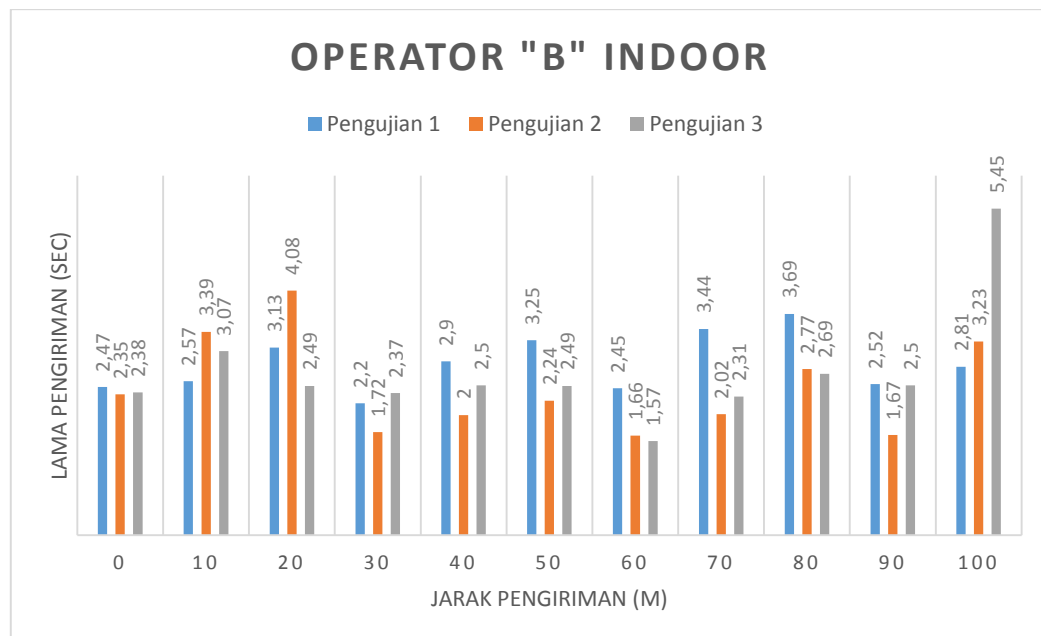


Gambar 4. 3 Grafik hubungan lama pengiriman dengan jarak pengiriman pesan pada operator A dalam pengujian *indoor*.

Tabel 4. 4 Data pengujian *indoor* pada operator A.

Jarak (m)	Lama Pengiriman Pesan		
	Pengujian 1	Pengujian 2	Pengujian 3
0	2,26	1,69	1,55
10	3,41	1,66	1,7
20	1,79	1,43	1,54
30	2,15	1,85	2,45
40	4,05	2,62	1,97
50	2,71	1,6	1,43
60	1,02	1,78	2,07
70	2,21	2,53	1,79
80	2,93	1,48	1,61
90	0,79	2,48	2,12
100	2,02	1,69	2,62
Rata-rata	2,3036	1,8918	1,8954

Pengujian selanjutnya adalah pengujian *indoor* pada operator B. Pada operator B didapatkan layanan internet HSDPA. Hasil pengujian bisa dilihat pada Gambar 4.4.



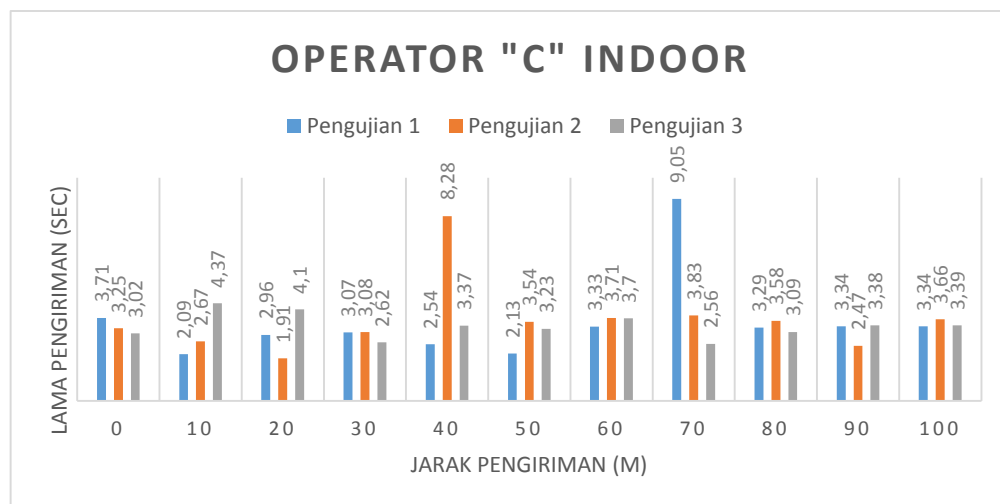
Gambar 4. 4 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator B dalam pengujian *indoor*.

Tabel 4. 5 Data pengujian *indoor* pada operator B.

Jarak (m)	Lama Pengiriman Pesan (Sec)		
	Pengujian 1	Pengujian 2	Pengujian 3
0	2,47	2,35	2,38
10	2,57	3,39	3,07
20	3,13	4,08	2,49
30	2,2	1,72	2,37
40	2,9	2	2,5
50	3,25	2,24	2,49
60	2,45	1,66	1,57
70	3,44	2,02	2,31
80	3,69	2,77	2,69
90	2,52	1,67	2,5
100	2,81	3,23	5,45
Rata-rata	2,8572	2,4663	2,7109

Dari grafik pada Gambar 4.4 dapat dilihat bahwa lama pengiriman pesan dari *web server* ke Android bervariasi. Pada grafik tersebut ditunjukkan bahwa lama pengiriman tidak tergantung pada jarak pengirim pesan ke penerima. Rata-rata lama pengiriman yang dihasilkan dari pengujian *indoor* ini sebesar 2,6781 detik.

Pengujian *indoor* yang terakhir adalah pengujian *indoor* pada operator C. Pada pengujian ini layanan internet yang diperoleh adalah EDGE. Hasil pengujian bisa dilihat pada Gambar 4.5.



Gambar 4. 5 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator C dalam pengujian *indoor*.

Tabel 4. 6 Data pengujian *indoor* pada operator C.

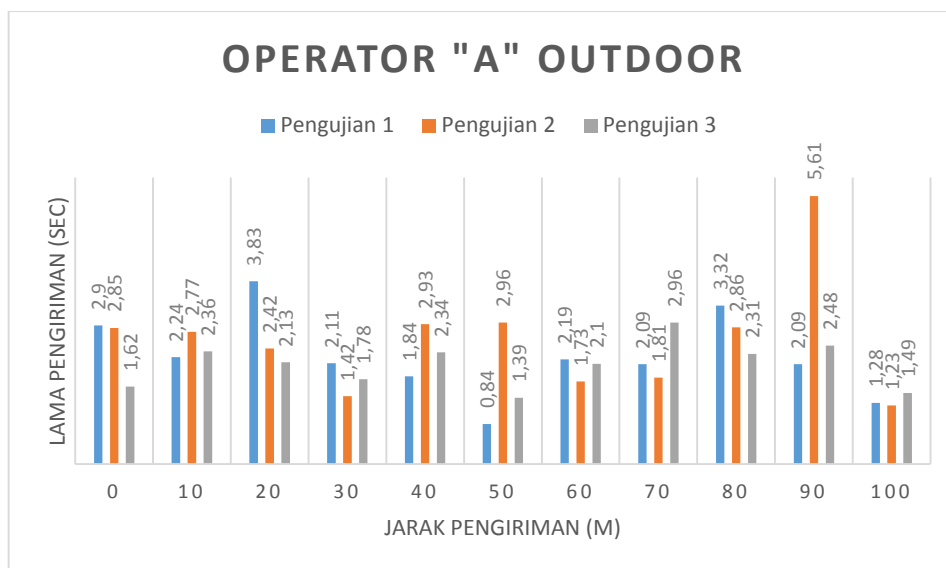
Jarak (m)	Lama Pengiriman Pesan (Sec)		
	Pengujian 1	Pengujian 2	Pengujian 3
0	3,71	3,25	3,02
10	2,09	2,67	4,37
20	2,96	1,91	4,1
30	3,07	3,08	2,62
40	2,54	8,28	3,37
50	2,13	3,54	3,23
60	3,33	3,71	3,7
70	9,05	3,83	2,56
80	3,29	3,58	3,09
90	3,34	2,47	3,38
100	3,34	3,66	3,39
Rata-rata	3,5318	3,6345	3,3418

Dari grafik pada Gambar 4.5 dapat dilihat bahwa lama pengiriman pesan dari *web server* ke Android bervariasi. Pada grafik tersebut terdapat data pencilan pada jarak 70 meter pada pengujian pertama dengan waktu terlama yaitu 9,05 detik dan jarak 40 meter pada pengujian kedua dengan waktu 8,28 detik. Pencilan ini didapatkan karena faktor jaringan internet yang lambat sehingga pesan yang dikirimkan *delay* sebesar 9,05 detik dan 8,28 detik. Selebihnya data yang didapatkan tidak jauh dari rata-rata. Rata-rata total pada pengujian ini sebesar 3,5027 detik.

4.4.2 Pengujian *Outdoor*

Pengujian dilakukan untuk mengamati lama pengiriman pesan dari *web server* ke peranti Android dengan jarak 0 sampai 100 meter dengan interval jarak 10 meter. Pengujian ini dilakukan pada luar gedung FTI UII dengan 3 kali pengujian yang sama. Pada pengujian ini digunakan 3 operator yang berbeda, yaitu operator A, B, dan C. Berikut ini adalah hasil dari pengujian *outdoor*.

Pengujian pertama adalah *outdoor* pada operator A. Pada pengujian ini layanan internet yang diperoleh adalah HSDPA. Hasil pengujian bisa dilihat pada Gambar 4.6.



Gambar 4. 6 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator A dalam pengujian *outdoor*.

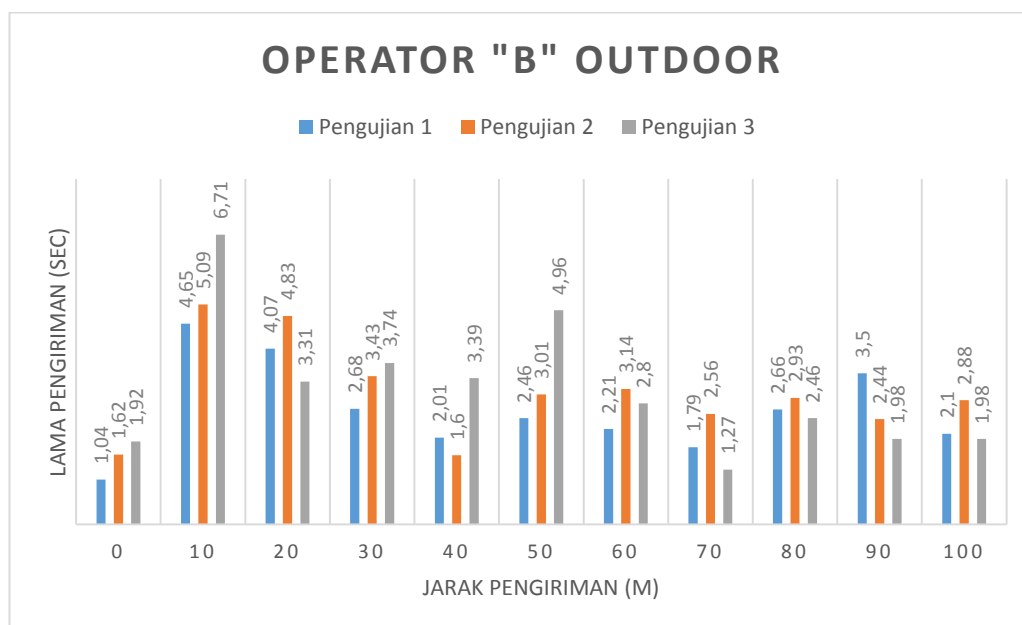
Tabel 4. 7 Data pengujian *outdoor* pada operator A.

Jarak (m)	Lama Pengiriman Pesan (Sec)		
	Pengujian 1	Pengujian 2	Pengujian 3
0	2,9	2,85	1,62
10	2,24	2,77	2,36
20	3,83	2,42	2,13
30	2,11	1,42	1,78
40	1,84	2,93	2,34
50	0,84	2,96	1,39
60	2,19	1,73	2,1
70	2,09	1,81	2,96
80	3,32	2,86	2,31
90	2,09	5,61	2,48
100	1,28	1,23	1,49
Rata-rata	2,2481	2,5990	2,0872

Dari grafik pada Gambar 4.6 dapat dilihat bahwa lama pengiriman pesan dari *web server* ke aplikasi Android bervariasi. Rata-rata lama pengiriman pesan dari *web server* ke peranti Android sebesar 2,3115 detik. Pengujian di lingkungan *outdoor* lebih cepat 0,28 detik dari pengujian *indoor*, perbedaan yang didapatkan tidak terlalu besar dari kedua pengujian ini.

Pengujian selanjutnya adalah pengujian menggunakan layanan pada operator B. Pada pengujian ini didapatkan layanan internet HSDPA. Hasil pengujian bisa dilihat pada Gambar 4.7.

Dari grafik pada Gambar 4.7 dapat dilihat bahwa lama pengiriman pesan dari *web server* ke aplikasi Android bervariasi. Rata-rata total dari lama pengiriman pesan untuk sampai pada peranti Android sebesar 2,9460. Dari pengujian ini operator B pada kondisi *outdoor* lebih lambat 0,26 detik dari pengujian *indoor*.

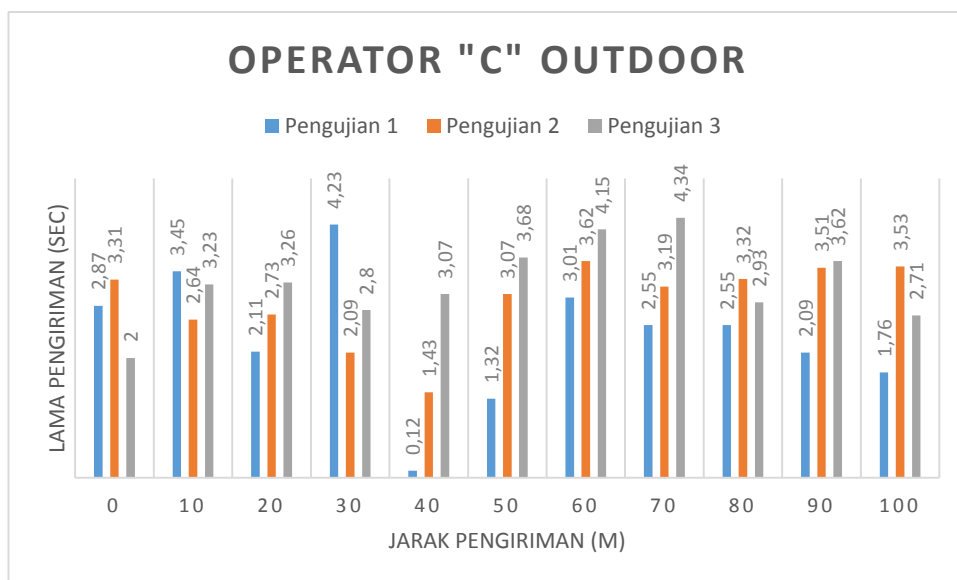


Gambar 4. 7 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator B dalam pengujian *outdoor*.

Tabel 4. 8 Data pengujian *outdoor* pada operator B.

Jarak (m)	Lama Pengiriman Pesan (Sec)		
	Pengujian 1	Pengujian 2	Pengujian 3
0	1,04	1,62	1,92
10	4,65	5,09	6,71
20	4,07	4,83	3,31
30	2,68	3,43	3,74
40	2,01	1,6	3,39
50	2,46	3,01	4,96
60	2,21	3,14	2,8
70	1,79	2,56	1,27
80	2,66	2,93	2,46
90	3,5	2,44	1,98
100	2,1	2,88	1,98
Rata-rata	2,651818182	3,048181818	3,138181818

Pengujian *outdoor* yang terakhir adalah pengujian menggunakan layanan dari operator C. Pada pengujian ini didapatkan jaringan internet EDGE. Hasil pengujian bisa dilihat pada Gambar 4.8.



Gambar 4. 8 Grafik hubungan lama pengiriman terhadap jarak pengiriman pesan pada operator C dalam pengujian *outdoor*.

Tabel 4. 9 Data pengujian *outdoor* pada operator C.

Jarak (m)	Lama Pengiriman Pesan (Sec)		
	Pengujian 1	Pengujian 2	Pengujian 3
0	2,87	3,31	2
10	3,45	2,64	3,23
20	2,11	2,73	3,26
30	4,23	2,09	2,8
40	0,12	1,43	3,07
50	1,32	3,07	3,68
60	3,01	3,62	4,15
70	2,55	3,19	4,34
80	2,55	3,32	2,93
90	2,09	3,51	3,62
100	1,76	3,53	2,71
Rata-rata	2,3690	2,9490	3,2536

Dari grafik pada gambar 4.8 dapat dilihat bahwa lama pengiriman pesan dari *web server* ke aplikasi Android bervariasi. Pengujian paling cepat pada operator C ini adalah pengujian pertama pada jarak 40 meter sebesar 0,12 detik. Hal ini sangat bergantung pada kecepatan internet yang digunakan, meskipun jaringan yang didapatkan EDGE namun pesan bisa sampai dengan cepat. Rata-rata total kecepatan pengiriman pesan dari *web server* ke peranti Android pada pengujian ini sebesar 2,8572 detik. Pengujian *outdoor* ini lebih cepat 0,64 detik dari pengujian *indoor* menggunakan operator yang sama.

4.4.3 Analisa Keseluruhan Sistem

Pengujian yang dilakukan memiliki karakteristik masing-masing. Hal ini dipengaruhi oleh faktor kecepatan pengiriman dan penerimaan data dari *web server* ke peranti Android. Faktor yang diujikan adalah perbedaan kecepatan layanan dari masing-masing operator terhadap jarak dan lokasi yang diujikan.

Jarak bukan faktor linier terhadap kecepatan pengiriman dan penerimaan pesan. Faktor yang menentukan pada pengujian ini adalah lokasi akses internet dan kecepatan internet dari layanan masing-masing operator.

Tabel 4. 10 Rata-rata total waktu yang dibutuhkan untuk pengiriman pada kondisi *indoor* dan *outdoor*.

Kondisi Pengujian	Rata-rata total waktu yang dibutuhkan untuk pengiriman (sec)		
	Operator A	Operator B	Operator C
Indoor	2,0303	2,6781	3,5021
Outdoor	2,3115	2,946	2,8572

Menurut data dari pengujian di atas, transfer data paling cepat pada pengujian *indoor* adalah pengujian menggunakan operator A, selanjutnya adalah operator B dan C. Pada pengujian *outdoor* transfer data paling cepat menggunakan operator A, selanjutnya adalah operator C dan B. Data yang dikirimkan *web server* ke aplikasi sebesar 4 KB, hal ini menunjukkan bahwa jaringan internet EDGE masih bisa digunakan dengan baik. Berikut adalah hubungan *data rate* masing-masing komponen terhadap *delay* yang didapatkan.

Tabel 4. 11 Hubungan *data rate* komponen terhadap waktu tunda (*delay*) pengiriman perintah.

<i>Data Rate Laptop (upload)</i>	<i>Data Rate Android (download)</i>	<i>Delay Laptop ke web server</i>	<i>Delay Android ke web server</i>	<i>Delay Total</i>
0,37 Mbps	1,76 Mbps	163 ms	169 ms	332 ms
0,34 Mbps	1,37 Mbps	164 ms	174 ms	338 ms
0,37 Mbps	1,62 Mbps	163 ms	169,5 ms	332,5 ms
0,36 Mbps	1,21 Mbps	163,5 ms	191,5 ms	355 ms
0,37 Mbps	1,18 Mbps	163 ms	196 ms	359 ms
0,37 Mbps	1,19 Mbps	163,5 ms	178,5 ms	342 ms

Pengambilan data pada Tabel 4.11 diambil pada masing-masing komponen yaitu *laptop* dan peranti *Android*. Pada *laptop*, *data rate* diambil dengan cara mengakses situs <http://www.speedtest.net/>. Situs tersebut berfungsi untuk melihat *data rate* internet yang digunakan oleh *laptop*. Pada situs tersebut terdapat tiga pengukuran yaitu *ping* (ISP ke *host*), *data rate download*, dan *data rate upload*. Selanjutnya *data rate* yang diambil adalah *data rate upload* yang terukur oleh *web* tersebut. Kemudian waktu tunda *laptop* ke *web server* dicari dengan melakukan *ping* ke alamat IP *web server* dengan menggunakan *tool* command prompt yang tersedia di OS Windows 7. Pada proses *ping* didapatkan nilai *rtt* (*round trip times*), nilai yang diambil adalah setengah dari nilai *rtt* (*round trip times*), nilai tersebut adalah waktu tunda pengiriman perintah dari *laptop* ke *web server*.

Pengambilan data pada peranti *Android* diambil menggunakan dua aplikasi tambahan yaitu *SpeedTest* dan *Ping & DNS*. *SpeedTest* mengacu pada situs *web* <http://www.speedtest.net/> yang digunakan untuk mengukur *data rate* internet yang dapat diakses peranti *Android*. Pada pengukuran menggunakan aplikasi *SpeedTest* didapatkan tiga hasil yaitu *ping* (ISP ke *host*), *data rate download*, dan *data rate upload*. Selanjutnya *data rate* yang diambil adalah *data rate download* yang terukur oleh aplikasi tersebut. Aplikasi Selanjutnya adalah *Ping & DNS* digunakan untuk melakukan *ping* *Android* ke *web server* untuk mendapatkan *rtt* (*round trip times*) *web server* ke *Android*. Nilai yang diambil adalah setengah dari *rtt* (*round trip times*), nilai tersebut adalah nilai waktu tunda penerimaan perintah dari *web server* ke peranti *Android*.

BAB V

PENUTUP

4.1 Kesimpulan

1. Sistem kendali jarak jauh yang diimplementasikan adalah sistem yang menggunakan komunikasi internet untuk bertukar data.
2. Dari perancangan dihasilkan sebuah sistem robot yang dapat dikendalikan jarak jauh serta bertukar data dengan media *web server* tanpa terbatas jarak jangkauan.
3. Pengujian yang dilakukan adalah mengamati lama pengiriman pesan terhadap jarak pengirim pada kondisi *indoor* dan *outdoor*. Pengujian dibedakan pada penggunaan jenis operator penyedia layanan internet, yaitu operator A,B, dan C.
4. Berdasarkan data pengujian, operator A unggul dalam kondisi *indoor* maupun *outdoor* dengan kecepatan rata-rata pada kondisi *indoor* 2,0302 detik dan 2,3115 detik pada kondisi *outdoor*. Selanjutnya, pada operator B dan C memiliki selisih waktu yang tidak jauh berbeda dari operator A.
5. Berdasarkan data pengujian, kecepatan internet bersifat linier terhadap waktu tunda pengiriman dan penerimaan data sehingga semakin besar *bandwidth internet* yang bisa diakses maka semakin cepat pengiriman dan penerimaan pesan.

4.2 Saran

1. Dibutuhkan tambahan program untuk mengubah hasil foto menjadi *video streaming* dan penggunaan tombol navigasi kendali robot menjadi tombol

yang bisa mengirim pesan navigasi ketika tombol yang berada di *web server* diklik dan menghentikan robot ketika tombol dilepas.

2. Dibutuhkan tambahan *pan* dan *tilt* untuk peranti Android yang berada di robot beroda. Tambahan tersebut digunakan untuk mengubah arah bidikan foto tanpa mengubah arah robot untuk membidik lokasi sekitar.

DAFTAR PUSTAKA

- [1] Robot-alive, *Introduction to Mobile Robotics*, <http://www.slideshare.net/robotsalive/introduction-to-mobile-robotics>, 2010. [Online].
- [2] W. A. Pangestu, *Pengendali Robot Jarak Jauh Berbasis Smartphone Android*. Skripsi, tidak diterbitkan, Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta, 2014.
- [3] B. M. Anggoro, *Desain Pemodelan Kinematik dan Dinamik Humanoid Robot*, Faculty Engineering of Diponegoro University, Semarang, 2013.
- [4] M. M., *Tutorialspoint*, 13 january 2015. [Online]. Available: <http://www.tutorialspoint.com/android/index.html>.
- [5] G. Inc., *Google Cloud Messaging For Android*, google Inc., 14 1 2015. [Online]. Available: <https://developer.android.com/google/gcm/index.html>.
- [6] G. Inc., *Google Play Support*, 2015. [Online]. Available: <https://support.google.com/googleplay/android-developer/answer/2663268?hl=id>. [Diakses 14 Januari 2015].
- [7] G. Inc., *Developer Android GCM*, 2015. [Online]. Available: <https://developer.android.com/google/gcm/gcm.html#key>.
- [8] R. K. Rohman, *Pengertian Web Server*, 2014. [Online]. Available: <http://rizalkurnia.com/kuliah-2/tugas/pengertian-web-server/>. [Diakses Oktober 2014].

- [9] UMM, *Paduan Lengkap Membangun Server berbasis GUI Menggunakan Linux SuSE*, 2010. [Online]. Available: [directory.umm.ac.id/Operating%20System %20Ebook/SUSE/suse6.pdf](http://directory.umm.ac.id/Operating%20System%20Ebook/SUSE/suse6.pdf).
- [10] B. Sidik, *Pemrograman Web dengan PHP*, Bandung: Informatika Bandung, 2012.
- [11] S. Monk, *Making Android Accessories with IOIO*, USA: O'Reilly Media Inc., 2012.
- [12] R. M. Alfiyan, *Web Server Berbasis Embedded Ethernet Untuk Sistem Kendali dan Monitoring Jarak Jauh Pada Ruang Penyimpanan Barang-Barang Berharga*, *Jurnal Tugas Akhir*, vol. I, pp. 1-13, 2010.
- [13] Karomah, *Rancang Bangun Sistem Kendali Jarak Jauh on/off Lampu dan Air Conditioner (AC) Berbasis Arduino Melalui Internet*, *Jurnal Penelitian*, 2014.
- [14] R. A. P. Gorat, *Perancangan Sistem Pemantauan Energi Listrik pada Bangunan Secara Waktu Nyata Berbasis Web*, *Tugas Akhir*, 2013.
- [15] M. Tutorial, *Mengenal Intent dan Kegunaannya*, 06 Februari 2015. [Online]. Available: <http://www.mediatutorial.web.id/2015/02/praktek-mengirim-data-antar-activity.html>. [Diakses 04 April 2015].
- [16] [Online]. Available: <http://ashabulyamin5315072405.blogspot.com/> & <https://solarbotics.com/product/50858/>.
- [17] [Online]. Available: <http://www.hermantolle.com/class/2014/11/notifikasi-push-android-menggunakan-google-cloud-messaging-gcm-php-dan-mysql/>.

LAMPIRAN 1

LAMPIRAN PROGRAM WEB SERVER

1. Break.php

```

<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
try {
    $conn = new pdo("mysql:host=$host;
    dbname=$db;charset=utf8", $user, $password, array(PDO::MYSQL_ATTR_INI
    T_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
    echo "Not Connected :". $e->getMessage();
}

$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}

$message = "break";

$url = 'https://android.googleapis.com/gcm/send'; // don't change
this url :)

$fields = array(
    'registration_ids' => $registrationIDs,
    'data' => array( "type" =>
1, "message" => $message ),
);

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
);

$ch = curl_init();

curl_setopt( $ch, CURLOPT_URL, $url );

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

```

LAMPIRAN 1

```
$result = curl_exec($ch);
```

```
curl_close($ch);
```

```
?>
```

2. Location.php

```
<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
try {
    $conn = new pdo("mysql:host=$host;
dbname=$db;charset=utf8",$user,$password,array(PDO::MYSQL_ATTR_INI
T_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
echo "Not Connected :". $e->getMessage();
}
$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}

if($_GET['show']=="on"){
    $message = "on";
}
else{
    $message = "off";
}

$url = 'https://android.googleapis.com/gcm/send';

$fields = array(
    'registration_ids' => $registrationIDs,
    'data' => array( "type" =>
2,"message" => $message ),
);

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
);

$ch = curl_init();

curl_setopt( $ch, CURLOPT_URL, $url );
```

LAMPIRAN 1

```

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

$result = curl_exec($ch);

curl_close($ch);

mysql_close($db);
?>

```

3. Right.php

```

<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
try {
    $conn = new pdo("mysql:host=$host;
    dbname=$db;charset=utf8", $user, $password, array(PDO::MYSQL_ATTR_INI
    T_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
    echo "Not Connected :". $e->getMessage();
}
$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}

$message = "right";

$url = 'https://android.googleapis.com/gcm/send';

$fields = array(
    'registration_ids' => $registrationIDs,
    'data' => array("type" => 1, "message"
=> $message ),
);

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
);

$ch = curl_init();

```

LAMPIRAN 1

```

curl_setopt( $ch, CURLOPT_URL, $url );

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

$result = curl_exec($ch);

curl_close($ch);
?>

```

4. Camera.php

```

<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
try {
    $conn = new pdo("mysql:host=$host;
    dbname=$db;charset=utf8",$user,$password,array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
    echo "Not Connected :". $e->getMessage();
}
$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}

$message = "take";

$url = 'https://android.googleapis.com/gcm/send';

$fields = array(
    'registration_ids' => $registrationIDs,
    'data' => array("type" => 4,
    "message" => $message ),
);

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
);

$ch = curl_init();

```

LAMPIRAN 1

```

curl_setopt( $ch, CURLOPT_URL, $url );

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

$result = curl_exec($ch);

```

```

curl_close($ch);?>

```

5. Down.php

```

<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
try {
    $conn = new pdo("mysql:host=$host;
    dbname=$db;charset=utf8",$user,$password,array(PDO::MYSQL_ATTR_INI
    T_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
echo "Not Connected :". $e->getMessage();
}

$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}

$message = "down";

$url = 'https://android.googleapis.com/gcm/send';

$fields = array(
    'registration_ids' => $registrationIDs,
    'data' => array("type" => 1,
    "message" => $message ),
);

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
);

$ch = curl_init();

```

LAMPIRAN 1

```

curl_setopt( $ch, CURLOPT_URL, $url );

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

$result = curl_exec($ch);

curl_close($ch);?>

```

6. Flash.php

```

<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
try {
    $conn = new pdo("mysql:host=$host;
    dbname=$db;charset=utf8", $user, $password, array(PDO::MYSQL_ATTR_INI
    T_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
    echo "Not Connected :". $e->getMessage();
}

$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}
if($_GET['flash']=="on"){
    $message = "on";
}
else{
    $message = "off";
}

$url = 'https://android.googleapis.com/gcm/send';

$fields = array(
    'registration_ids' => $registrationIDs,
    'data' => array( "type" => 3,
    "message" => $message ),
);

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
);

```

LAMPIRAN 1

```

$ch = curl_init();

curl_setopt( $ch, CURLOPT_URL, $url );

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

$result = curl_exec($ch);

curl_close($ch);?>

```

7. Get_location.php

```

<?php

$db=mysql_connect("mysql.idhostinger.com","u325875635_devil","arta
241112") or die (mysql_error());
mysql_select_db("u325875635_satan",$db);

$result=mysql_query("SELECT lat FROM location WHERE id=0");
$row=mysql_fetch_assoc($result);

echo $row["lat"];

$result=mysql_query("SELECT lon FROM location WHERE id=0");
$row=mysql_fetch_assoc($result);
echo ":";
echo $row["lon"];
echo "*";

mysql_close($db);?>

```

8. Index.php

```

<html>
<head>
<title>Kanjeng - Fris_Bot</title>
<link href="img2/Airport.ico" rel="shortcut icon" />

<!-- including jquery library from Google servers-->
<script type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"><
/script>

<!-- two scripts used to display Google maps -->
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDQ73Wy46uiXw
vQs8dg--sukdYM7wjT5jw&sensor=false"></script>

```

LAMPIRAN 1

```

<!-- <script type="text/javascript" language="javascript"
src="res/location.js"></script> -->

<link rel="stylesheet" href="res/experiment-styles.css" />
<link rel="stylesheet" type="text/css" href="res/style.css" />
<link rel="stylesheet" type="text/css" href="res/layout.css" />

</head>

<body onload = "initialize()">
<?php
$conn = mysqli_connect('mysql.idhostinger.com',
'u325875635_devil', 'arta241112', 'u325875635_satan');

$a = mysqli_query($conn, "SELECT * FROM location");
//mengulang semua baris yang ada pada query diatas
//otomatis membaca data query terakhir untuk update maps
while($data=mysqli_fetch_array($a)){
    $lats=$data['lat'];
    $lons=$data['lon'];
}

?>

<script type="text/javascript" >

var lat = <?php echo $lats ?>;
var lon = <?php echo $lons ?>;
var myCenter = new google.maps.LatLng(lat,lon);

function initialize()
{

//letakkan disini

var mapProp = {
    center:myCenter,
    zoom:17,
    mapTypeId: google.maps.MapTypeId.ROADMAP
};

var map=new google.maps.Map(document.getElementById('googleMap')
,mapProp);

var marker=new google.maps.Marker({
    position:myCenter,
    title: 'FrisBot disini',
    map: map
});

marker.setMap(map);
}

xmlhttp=new XMLHttpRequest();

```

LAMPIRAN 1

```

xmlhttp.onreadystatechange=function()
{
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
    var response=xmlhttp.responseText;
    var n=response.indexOf(":");
    var r=response.indexOf("*");
    lat=parseFloat(response.substring(0, n));
    lon=parseFloat(response.substring(n+1,r));

myCenter=new google.maps.LatLng(lat,lon);

//atau disini
  }
xmlhttp.open("GET","get_location.php?q=",true);
xmlhttp.send();
setTimeout(initialize,5000);
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>

<div id="titlebar"></div>
<div id="cc" ></div>
<div id ="mapback">
</div>
<div id="glass"></div>
<div id="tie"></div>
<div id="circles"></div>

<div id="map"><div id="googleMap"
style="width:500px;height:320px;"></div></div>
<div id="textarea" ><textarea id="log" readonly rows="5"
cols="60">
jejak-jejak perintah panjenengan :
-----
</textarea></div>

<aside id="example" class="modal">
  <div>
    <h3>sounds list</h3>
    <a href="JavaScript:void(0);"
onclick="sound('jigsaw1');" >1- jigsaw1</a><br>
    <a href="JavaScript:void(0);"
onclick="sound('jigsaw2');" >2- jigsaw2</a><br>
    <a href="JavaScript:void(0);"
onclick="sound('ghosts');" >3- ghosts</a><br>
    <a href="JavaScript:void(0);"
onclick="sound('horn');" >4- horn</a><br>
    <a href="JavaScript:void(0);"
onclick="sound('start');" >5- car start up</a>
    <a href="#close" title="Close">Close</a>
  </div>
</aside>

```

LAMPIRAN 1

```

<div onmouseover="tooltip.show('Maju');"
onmouseout="tooltip.hide();" id="up"><a href="JavaScript:void(0);"
onmouseover="img1.src='img2/upover.png';"
onmouseout="img1.src='img2/up.png';" onclick="up();" >
   </a></div>

```

```

<div onmouseover="tooltip.show('belok kiri');"
onmouseout="tooltip.hide();" id="left"><a
href="JavaScript:void(0);"
onmouseover="img2.src='img2/leftover.png';"
onmouseout="img2.src='img2/left.png';" onclick="left();" >
   </a></div>

```

```

<div onmouseover="tooltip.show('belok kanan');"
onmouseout="tooltip.hide();" id="right"> <a
href="JavaScript:void(0);"
onmouseover="img3.src='img2/rightover.png';"
onmouseout="img3.src='img2/right.png';" onclick="right();" >
   </a> </div>

```

```

<div onmouseover="tooltip.show('mundur');"
onmouseout="tooltip.hide();" id="down"> <a
href="JavaScript:void(0);"
onmouseover="img4.src='img2/downover.png';"
onmouseout="img4.src='img2/down.png';" onclick="down();" >
   </a> </div>

```

```

<div onmouseover="tooltip.show('berhenti');"
onmouseout="tooltip.hide();" id="break"> <a
href="JavaScript:void(0);"
onmouseover="img5.src='img2/breakover.png';"
onmouseout="img5.src='img2/break.png';" onclick="break1();" >
   </a> </div>

```

```

<div onmouseover="tooltip.show('play/stop sound file');"
onmouseout="tooltip.hide();" id="bell"> <a href="#example"
onmouseover="img6.src='img2/onoff/bell-on.png';"
onmouseout="img6.src='img2/onoff/bell-off.png';" >
   </a> </div>

```

```

<div onmouseover="tooltip.show('Take Picture');"
onmouseout="tooltip.hide();" id="camera"> <a
href="JavaScript:void(0);" onclick="camera();" >
   </a> </div>

```

```

<div onmouseover="tooltip.show('flashlight on/off');"
onmouseout="tooltip.hide();" id="flash"> <a
href="JavaScript:void(0);" onclick="flash();" >
   </a> </div>

```

LAMPIRAN 1

```

<div onmouseover="tooltip.show('location map show/hide');"
onmouseout="tooltip.hide();" id="location"> <a
href="JavaScript:void(0);" onclick="location2();">
   </a> </div>

  <!-- script used when a command is sent -->
<script type="text/javascript" language="javascript"
src="res/commands.js"></script>

<!-- script used when displaying a hint -->
<script type="text/javascript" language="javascript"
src="res/hint.js"></script>

<div id="textView">
<h2>HASIL FOTO</h2>
</div>

<div id="photo">

</div>

</body>
</html>

```

9. Left.php

```

<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
try {
$conn = new pdo("mysql:host=$host;
dbname=$db;charset=utf8",$user,$password,array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
echo "Not Connected :". $e->getMessage();
}

$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}

$message = "left";

$url = 'https://android.googleapis.com/gcm/send';

$fields = array(
    'registration_ids' => $registrationIDs,

```

LAMPIRAN 1

```

        'data'                                => array("type" => 1,
"message" => $message ),
    );

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
);

print_r($fields);
$ch = curl_init();

curl_setopt( $ch, CURLOPT_URL, $url );

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

$result = curl_exec($ch);

curl_close($ch);?>

```

10. Sound.php

```

<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
try {
    $conn = new pdo("mysql:host=$host;
dbname=$db;charset=utf8",$user,$password,array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
    echo "Not Connected :". $e->getMessage();
}

$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}
$sf=$_GET['file'];
switch($sf){
case "jigsaw1":$message = "jigsaw1";break;
case "jigsaw2":$message = "jigsaw2";break;
case "ghosts":$message = "ghosts";break;
case "horn":$message = "horn";break;

```

LAMPIRAN 1

```

case "start":$message = "start";break;
}

$url = 'https://android.googleapis.com/gcm/send';

$fields = array(
    'registration_ids' => $registrationIDs,
    'data'              => array( "type" => 5,
    "message" => $message ),
);

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
);

$ch = curl_init();

curl_setopt( $ch, CURLOPT_URL, $url );

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

$result = curl_exec($ch);

curl_close($ch);
?>

```

11. Storegps.php

```

<?php
$conn = mysqli_connect('mysql.idhostinger.com',
'u325875635_devil', 'arta241112', 'u325875635_satan');

$lat=$_POST['lat'];
$lon=$_POST['lon'];
mysqli_query($conn, "INSERT INTO location(lat,lon) value('$lat',
'$lon') " , $db);
mysqli_close($db);
?>

```

12. Storepic.php

```

<?php

// Path to move uploaded files
$target_path = "uploads/";

if (isset($_FILES['image']['name'])) {

```

LAMPIRAN 1

```

        $target_path = $target_path .
basename($_FILES['image']['name']);
        unlink('uploads/upload.jpg');
        try {
            // Throws exception incase file is not being moved
            if (!move_uploaded_file($_FILES['image']['tmp_name'],
$target_path)) {
                // make error flag true
            }
        } catch (Exception $e) {

        }
    } else {
        // File parameter is missing
    }
}
?>

```

13. Storeregid.php

```

<?php

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan";
    try {
        $conn = new pdo("mysql:host=$host;
dbname=$db;charset=utf8", $user, $password, array(PDO::MYSQL_ATTR_INI
T_COMMAND => "SET NAMES 'utf8'"));
    }
    catch (PDOException $e) {
        echo "Not Connected :". $e->getMessage();
    }
    $regid=$_GET['regid'];

    $stmt = $conn->prepare("INSERT INTO regid(reg
value(:reg)");
    $stmt->execute(array("reg"=>$regid));
?>

```

14. Up.php

```

<?php
$apiKey = "AIzaSyDQ73Wy46uiXwvQs8dg--sukdYM7wjT5jw";

$host = "mysql.idhostinger.com";
$user = "u325875635_devil";
$password = "arta241112";
$db = "u325875635_satan"; try {
    $conn = new pdo("mysql:host=$host;
dbname=$db;charset=utf8", $user, $password, array(PDO::MYSQL_ATTR_INI
T_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e) {
    echo "Not Connected :". $e->getMessage();
}

```

LAMPIRAN 1

```

}

$registrationIDs[] = array();
$stmt = $conn->prepare("SELECT reg FROM regid");
$stmt->execute();
while ($arr = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $registrationIDs[] = $arr["reg"];
}

$message = "up";

$url = 'https://android.googleapis.com/gcm/send';

$fields = array(
    'registration_ids' => $registrationIDs,
    'data'              => array("type" => 1,
    "message" => $message ),
    );

$headers = array(
    'Authorization: key=' . $apiKey,
    'Content-Type: application/json'
    );

$ch = curl_init();

curl_setopt( $ch, CURLOPT_URL, $url );

curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );

$result = curl_exec($ch);

curl_close($ch);
?>

```

LAMPIRAN 2

Source code Android

1. MyActivity.java

```

public class MyActivity extends IOIOActivity implements
Registrar.RegistrarListener,
    MediaPlayer.OnPreparedListener {
    private final SimpleDateFormat formatter = new
SimpleDateFormat("dd-MM-yyyy " +
        "hh-mm-ss");
    private int kec_kiri, kec_kanan;
    boolean A, B, C, D;
    private String msg = new String();
    private DigitalOutput kir_maju;
    private DigitalOutput kir_mundur;
    private DigitalOutput kan_maju;
    private DigitalOutput kan_mundur;
    private PwmOutput pwm_kiri;
    private PwmOutput pwm_kanan;
    private RequestAction requestAction;
    private ViewGroup container;
    private Registrar registrar;
    static final String TAG = "GCMDemo";
    private CameraHelper cameraHelper;
    private LocationHelper locationHelper;
    private MediaPlayer mediaPlayer;

    @Override
    public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (registrar == null) {
            registrar = new Registrar(this, this);
        }
        if (cameraHelper == null) {
            cameraHelper = new CameraHelper(this, (FrameLayout)
findViewById(R.id
                .camera_container));
        }
        if (locationHelper == null) {
            locationHelper = new LocationHelper(this);
        }

        if (requestAction == null) {
            requestAction = new RequestAction();
            IntentFilter filter = new
IntentFilter(PROCESS_RESPONSE);
            filter.addCategory(Intent.CATEGORY_DEFAULT);
            container = (ViewGroup) findViewById(R.id.container);
            try {
                registerReceiver(requestAction, filter);
            }

```

LAMPIRAN 2

```

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    ((LocationManager)
getSystemService(Context.LOCATION_SERVICE))
        .addGpsStatusListener(mGPSStatusListener);
    }

    public GpsStatus.Listener mGPSStatusListener = new
GpsStatus.Listener() {
        public void onGpsStatusChanged(int event) {
            switch (event) {
                case GpsStatus.GPS_EVENT_STARTED:
                    Log.i("GPS Status", "GPS_EVENT_STARTED");
                    break;
                case GpsStatus.GPS_EVENT_STOPPED:
                    Log.i("GPS Status", "GPS_EVENT_STOPPED");
                    break;
                case GpsStatus.GPS_EVENT_FIRST_FIX:
                    Log.i("GPS Status", "GPS_EVENT_FIRST_FIX");

                    break;
                case GpsStatus.GPS_EVENT_SATELLITE_STATUS:
                    Log.i("GPS Status",
"GPS_EVENT_SATELLITE_STATUS");
                    break;
            }
        }
    };

    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(requestAction);
        if (cameraHelper != null) {
            cameraHelper.releaseCamera();
        }
    }

    @Override
    public void onGooglePlayAvailable() {

    }

    @Override
    public void onGooglePlayUnavailable() {

    }

    @Override
    public void onRegisterSuccess(String message) {

    }

    @Override
    public void resultId(String id) {

```

LAMPIRAN 2

```

    }

    public class IOIOLoop extends BaseIOIOLooper {

        @Override
        protected void setup() throws ConnectionLostException {
            // TODO Auto-generated method stub
            kan_maju    = ioio_.openDigitalOutput(1, true);
            kan_mundur  = ioio_.openDigitalOutput(2, true);
            kir_maju    = ioio_.openDigitalOutput(3, true);
            kir_mundur  = ioio_.openDigitalOutput(4, true);

            pwm_kiri    = ioio_.openPwmOutput(34, 4000);
            pwm_kanan   = ioio_.openPwmOutput(35, 4000);
        }

        @Override
        public void loop() throws ConnectionLostException,
        InterruptedException,
        ConnectionLostException {
            // TODO Auto-generated method stub
            if ("up".equals(msg)) {
                A = true;
                B = false;
                C = true;
                D = false;
                kec_kiri = 200-40;
                kec_kanan = 200-40;
            }

            if ("left".equals(msg)) {
                A = false;
                B = true;
                C = true;
                D = false;
                kec_kiri = 150-30;
                kec_kanan = 200-30;
            }

            if ("right".equals(msg)) {
                A = true;
                B = false;
                C = false;
                D = true;
                kec_kiri = 200-30;
                kec_kanan = 150-30;
            }

            if ("down".equals(msg)) {
                A = false;
                B = true;
                C = false;
                D = true;
                kec_kiri = 150+12;
                kec_kanan = 150+12;
            }
        }
    }

```

LAMPIRAN 2

```

        if ("break".equals(msg)) {
            A = false;
            B = false;
            C = false;
            D = false;
            kec_kiri = 0;
            kec_kanan = 0;
        }
        kir_maju.write(A);
        kir_mundur.write(B);
        kan_maju.write(C);
        kan_mundur.write(D);
        pwm_kiri.setPulseWidth(kec_kiri);
        pwm_kanan.setPulseWidth(kec_kanan);
        setText(kec_kiri, kec_kanan);
        Thread.sleep(10);
    }
}

private void setText(final int kiri, final int kanan) {
    runOnUiThread(new Runnable() {

        @Override
        public void run() {
            // TODO Auto-generated method stub
            ((TextView)
MyActivity.this.findViewById(R.id.textkiri)).setText
                (getString(R.string.text_kiri) + kiri);
            ((TextView)
MyActivity.this.findViewById(R.id.textkanan)).setText
                (getString(R.string.text_kanan) + kanan);
        }
    });
}

@Override
protected IOIOLooper createIOIOLooper() {
    return new IOIOLoop();
}

public static final String PROCESS_RESPONSE =
RequestAction.class.getSimpleName()
    + "intent .action.PROCESS_RESPONSE";

public class RequestAction extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        TextView textView = new
TextView(getApplicationContext());
        textView.setTextColor(Color.BLACK);
        String action =
intent.getStringExtra(GcmIntentService.ACTION_TYPE);
        if (action.equals(Constant.MESSAGE_TYPE_NAVIGATION)) {

```

LAMPIRAN 2

```

        msg =
intent.getStringExtra(GcmIntentService.ACTION_MESSAGE);
        textView.setText(getTime() + " :: NAVIGATION " +
intent

.getStringExtra(GcmIntentService.ACTION_MESSAGE));
    } else if
(action.equals(Constant.MESSAGE_TYPE_CAMERA)) {
        textView.setText(getTime() + " :: CAMERA " +
intent.getStringExtra
        (GcmIntentService.ACTION_MESSAGE));
        cameraHelper.takePicture(new
CameraHelper.CameraHelperListener() {

@Override
public void cameraUnavailable(String error) {
            errorLog(" :: CAMERA " + error);
        }

        @Override
        public void onSuccess(String filename, String
path) {
            successLog(" :: CAMERA picture name " +
filename + " " +
                "saved on" +
                " " + path);
        }
    });
    } else if (action.equals(Constant.MESSAGE_TYPE_FLASH))
{
        textView.setText(getTime() + " :: FLASH " +
intent.getStringExtra
        (GcmIntentService.ACTION_MESSAGE));
        if
(intent.getStringExtra(GcmIntentService.ACTION_MESSAGE)
        .equalsIgnoreCase("on")) {
            cameraHelper.flashOn(new
CameraHelper.FlashHelperListener() {
                @Override
                public void flashUnavailable(String error)
{
                    errorLog(" :: CAMERA " + error);
                }
            });
        } else {
            cameraHelper.flashOff();
        }
    } else if
(action.equals(Constant.MESSAGE_TYPE_LOCATION)) {
        textView.setText(getTime() + " :: LOCATION " +
intent.getStringExtra
        (GcmIntentService.ACTION_MESSAGE));
        if
(intent.getStringExtra(GcmIntentService.ACTION_MESSAGE)
        .equalsIgnoreCase("on")) {
            locationHelper.connect();
        } else {

```

LAMPIRAN 2

```

        locationHelper.disconnect();
    }
    } else if (action.equals(Constant.MESSAGE_TYPE_SOUND))
    {
        textView.setText(getTime() + " :: SOUND " +
intent.getStringExtra
        (GcmIntentService.ACTION_MESSAGE));
        String audio =
intent.getStringExtra(GcmIntentService.ACTION_MESSAGE);
        try {
            if (audio.equalsIgnoreCase("ghosts")) {
                playSound(R.raw.ghosts);
            } else if (audio.equalsIgnoreCase("horn")) {
                playSound(R.raw.horn);
            } else if (audio.equalsIgnoreCase("jigsaw1"))
        {
                playSound(R.raw.jigsaw1);
            } else if (audio.equalsIgnoreCase("jigsaw2"))
        {
                playSound(R.raw.jigsaw2);
            } else if (audio.equalsIgnoreCase("start")) {
                playSound(R.raw.start);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    container.addView(textView);
}

}

private void playSound(int src) {
    try {
        if (mediaPlayer != null) {
            mediaPlayer.stop();
            mediaPlayer.reset();
        }
    } catch (Exception e) {
    }
    mediaPlayer = MediaPlayer.create(this, src);
    mediaPlayer.setOnPreparedListener(this);
    try {
        mediaPlayer.prepare();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@Override
public void onPrepared(MediaPlayer mp) {
    mediaPlayer.start();
}

private void errorLog(String arg0) {
    TextView textView = new TextView(getApplicationContext());
    textView.setTextColor(Color.RED);
    textView.setText(getTime() + arg0);
}

```

LAMPIRAN 2

```

        container.addView(textView);
    }

    private void successLog(String arg0) {
        TextView textView = new TextView(getApplicationContext());
        textView.setTextColor(Color.GREEN);
        textView.setText(getTime() + arg0);
        container.addView(textView);
    }

    private String getTime() {
        return formatter.format(new
Date(Calendar.getInstance().getTimeInMillis()));
    }
}

```

2. Constant.java

```

public class Constant {
    public static final String MESSAGE_TYPE_NAVIGATION = "1";
    public static final String MESSAGE_TYPE_LOCATION = "2";
    public static final String MESSAGE_TYPE_FLASH = "3";
    public static final String MESSAGE_TYPE_CAMERA = "4";
    public static final String MESSAGE_TYPE_SOUND = "5";
}

```

3. CameraHelper.java

```

public class CameraHelper {

    private static String PATH;
    private final Context context;
    private Camera camera;
    private boolean flashOn;
    private boolean takingPicture;
    private boolean cameraReady;
    private boolean flashReady;
    private final FrameLayout preview;

    public CameraHelper(Context context, FrameLayout preview) {
        this.context = context;
        takingPicture = false;
        flashOn = false;
        this.preview = preview;
        cameraReady = checkCameraHardware();
        flashReady = isFlashAvailable();
        if (cameraReady) {
            initCamera();
        }
    }

    public void releaseCamera() {
        if (camera != null) {
            try {
                camera.stopPreview();
            } catch (Exception e) {

```

LAMPIRAN 2

```

        }
        try {
            camera.release();
        } catch (Exception e) {
        }
        camera = null;
    }
}

public void flashOn(FlashHelperListener listener) {
    if (flashReady && ! flashOn) {
        Camera.Parameters params = camera.getParameters();
        if (params.getFlashMode() !=
Camera.Parameters.FLASH_MODE_TORCH) {
params.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
        }
        camera.setParameters(params);
        flashOn = true;
    } else if (! flashReady && ! flashOn) {
        Log.i("", "Flash not available");
        listener.flashUnavailable("Flash not available in your
device");
    }
}

public void flashOff() {
    if (flashOn) {
        if (camera != null) {
            Camera.Parameters params = camera.getParameters();
            if (params.getFlashMode() ==
Camera.Parameters.FLASH_MODE_TORCH ||
                params.getFlashMode() ==
Camera.Parameters.FLASH_MODE_ON) {
                Log.i("", "flashOn2");
            }
            params.setFlashMode(Camera.Parameters.FLASH_MODE_OFF);
            camera.setParameters(params);
        }
        releaseCamera();
        initCamera();
    }
    flashOn = false;
}

private static void open(Context context) {
    PATH = Environment.getExternalStorageDirectory() +
"/Android/data/" + context
        .getPackageName();
    File file = new File(PATH);
    if (! file.exists()) {
        file.mkdir();
    }
    PATH += "/images/";
    file = new File(PATH);
    if (! file.exists()) {
        file.mkdir();
    }
}

```

LAMPIRAN 2

[illegible]

LAMPIRAN 2

```

listener.cameraUnavailable("Error accessing " +
                           "file: " +
e.getMessage());
    }
    Bitmap bitmapPicture =
BitmapFactory
        .decodeByteArray(data,
0, data.length);
    Matrix matrix = new Matrix();
    matrix.postRotate(90);
    int height =
bitmapPicture.getHeight();
    int width =
bitmapPicture.getWidth();
    Bitmap scaledBitmap =
        (bitmapPicture,
height, width, true);
    new
Uploader().executeOnExecutor(AsyncTask
        .THREAD_POOL_EXECUTOR,
Bitmap.createBitmap(scaledBitmap, 0, 0,
scaledBitmap.getWidth(),
scaledBitmap.getHeight(), matrix,
                           true));
    releaseCamera();
    initCamera();
    }
    });
    } catch (Exception e) {
        e.printStackTrace();
        takingPicture = false;
    }
    }
    if (flashOn) {
        releaseCamera();
    }
    } catch (Exception e) {
    }
    } else {
        listener.cameraUnavailable("Camera unavailable, "
+
            "" + "Maybe you need restart the device");
    }
    } else {
        listener.cameraUnavailable("Camera unavailable in your
device");
    }
    }

    private class Uploader extends AsyncTask<Bitmap, String,
String> {

```

LAMPIRAN 2

```

@Override
protected String doInBackground(Bitmap... params) {
    try {
        ByteArrayOutputStream bos = new
ByteArrayOutputStream();
        HttpClient httpClient = new DefaultHttpClient();
        params[0].compress(Bitmap.CompressFormat.JPEG,
100, bos);
        byte[] data = bos.toByteArray();
        HttpPost postRequest = new
HttpPost("http://fristaasrori.esy" + "" +
        ".es/storepic.php");
        ByteArrayBody bab = new ByteArrayBody(data,
"upload.jpg");
        MultipartEntity reqEntity = new
MultipartEntity(HttpMultipartMode
        .BROWSER_COMPATIBLE);
        reqEntity.addPart("image", bab);
        reqEntity.addPart("photoCaption", new
StringBody("sfsdfsdf"));
        postRequest.setEntity(reqEntity);
        HttpResponse response =
httpClient.execute(postRequest);
        BufferedReader reader = new BufferedReader(new
InputStreamReader
            (response.getEntity().getContent(), "UTF-
8"));
        String sResponse;
        StringBuilder s = new StringBuilder();

        while ((sResponse = reader.readLine()) != null) {
            s = s.append(sResponse);
        }
        System.out.println("Response: " + s);
    } catch (Exception e) {
        // handle exception here
        Log.e(e.getClass().getName(), e.getMessage());
    }
    return null;
}

private boolean checkCameraHardware() {
    if
(context.getPackageManager().hasSystemFeature(PackageManager
        .FEATURE_CAMERA)) {
        // this device has a camera
        return true;
    } else {
        // no camera on this device
        return false;
    }
}

private Camera getCameraInstances() {
    Camera camera = null;
    int cameraCount = 0;

```

LAMPIRAN 2

```

        try {
            camera = Camera.open();
            Preview cameraPreview = new Preview(context);
            preview.removeAllViews();
            cameraPreview.setCamera(camera);
            preview.addView(cameraPreview);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return camera; // returns null if camera is unavailable
    }

    public interface CameraHelperListener {
        public void cameraUnavailable(String error);

        public void onSuccess(String filename, String path);
    }

    public interface FlashHelperListener {
        public void flashUnavailable(String error);
    }
}

```

4. GcmBroadcastReceiver.java

```

public class GcmBroadcastReceiver extends WakefulBroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i("", "GcmBroadcastReceiver");
        ComponentName comp = new
ComponentName(context.getPackageName(),
                GcmIntentService.class.getName());
        startWakefulService(context, (intent.setComponent(comp)));
        setResultCode(Activity.RESULT_OK);
    }
}

```

5. GcmIntentService.java

```

public class GcmIntentService extends IntentService {
    public static final int NOTIFICATION_ID = 1;
    private NotificationManager mNotificationManager;
    NotificationCompat.Builder builder;
    static final String TAG = "GcmIntentService";
    public static final String ACTION_TYPE = "action_type";
    public static final String ACTION_MESSAGE = "message";
    public String message = "";

    public GcmIntentService() {
        super("GcmIntentService");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        Bundle extras = intent.getExtras();
    }
}

```

LAMPIRAN 2

```

        GoogleCloudMessaging gcm =
        GoogleCloudMessaging.getInstance(this);

        String messageType = gcm.getMessageType(intent);

        if (! extras.isEmpty()) {
            if
            (GoogleCloudMessaging.MESSAGE_TYPE_SEND_ERROR.equals(messageType))
            {
                sendNotification("Send error: " +
                extras.toString());
            } else if
            (GoogleCloudMessaging.MESSAGE_TYPE_DELETED.equals(messageType)) {
                sendNotification("Deleted messages on server: " +
                extras.toString());
            } else if
            (GoogleCloudMessaging.MESSAGE_TYPE_MESSAGE.equals(messageType)) {
                message = intent.getStringExtra("message");
                sendNotification("Received: " + message);
                Intent broadcastIntent = new Intent();

                broadcastIntent.setAction(MyActivity.PROCESS_RESPONSE);

                broadcastIntent.addCategory(Intent.CATEGORY_DEFAULT);
                broadcastIntent.putExtra(ACTION_MESSAGE, message);
                broadcastIntent.putExtra(ACTION_TYPE,
                intent.getStringExtra("type"));
                sendBroadcast(broadcastIntent);
            }
            GcmBroadcastReceiver.completeWakefulIntent(intent);
        }

        private void sendNotification(String msg) {
            mNotificationManager = (NotificationManager)
            this.getSystemService(Context
            .NOTIFICATION_SERVICE);

            PendingIntent contentIntent =
            PendingIntent.getActivity(this, 0,
            new Intent(this, MyActivity.class),
            PendingIntent.FLAG_UPDATE_CURRENT);

            NotificationCompat.Builder mBuilder = new
            NotificationCompat.Builder(this).
            setSmallIcon(R.mipmap.ic_launcher).
            setContentTitle("Argian Notification").
            setStyle(new
            NotificationCompat.BigTextStyle().bigText(msg)).
            setContentText(msg);

            mBuilder.setContentIntent(contentIntent);
            mNotificationManager.notify(NOTIFICATION_ID,
            mBuilder.build());
        }
    }
}

```

LAMPIRAN 2

6. LocationHelper.java

```

public class LocationHelper implements
    GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
    LocationListener {
    private GoogleApiClient googleApiClient;

    private static final LocationRequest REQUEST =
    LocationRequest.create()
        .setInterval(1000) // 1 seconds
        .setFastestInterval(16)
        .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

    public LocationHelper(Context context) {
        googleApiClient = new
        GoogleApiClient.Builder(context).addApi

        (LocationServices.API).addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this).build();
    }

    public void connect() {
        Log.i("", "location connecting");
        googleApiClient.connect();
    }

    public void disconnect() {
        googleApiClient.disconnect();
    }

    @Override
    public void onLocationChanged(Location location) {
        Log.i(location.getProvider(), location.getLatitude() + " ,
" +
            "" + location.getLongitude());
        List<NameValuePair> longlat = new
        ArrayList<NameValuePair>(1);
        String lat = "";
        String lon = "";

        lat = Double.toString(location.getLatitude());
        lon = Double.toString(location.getLongitude());

        longlat.add(new BasicNameValuePair("lat", lat));
        longlat.add(new BasicNameValuePair("lon", lon));
        new
        Uploader().executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR,
        longlat);
        Log.v("gps", lat + "," + lon);
    }

    private class Uploader extends AsyncTask<List<NameValuePair>,
    Boolean, Boolean> {

        @Override
        protected Boolean doInBackground(List<NameValuePair>...
        params) {

```

LAMPIRAN 2

```

        try {
            HttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new
HttpPost("http://fristaasrori.esy" +
            ".es/storegps" + ".php");
            httpPost.setEntity(new
UrlEncodedFormEntity(params[0]));
            HttpResponse response =
httpClient.execute(httpPost);
            Log.i("postData",
response.getStatusLine().toString());
        } catch (Exception e) {
            Log.e("location", "Error in http connection " +
e.toString());
        }
        return null;
    }

    @Override
    public void onConnected(Bundle bundle) {

LocationServices.FusedLocationApi.requestLocationUpdates(googleApi
Client,
        REQUEST, this);
    }

    @Override
    public void onConnectionSuspended(int i) {
    }

    @Override
    public void onConnectionFailed(ConnectionResult
connectionResult) {
    }
}

```

7. Preview.java

```

public class Preview extends ViewGroup implements
SurfaceHolder.Callback {
    private final String TAG = "Preview";

    private SurfaceView mSurfaceView;
    private SurfaceHolder mHolder;
    private Size mPreviewSize;
    protected Size mPictureSize;
    private List<Size> mSupportedPreviewSizes;
    protected List<Size> mPictureSizeList;
    private Camera mCamera;
    private float mDist;

    private static final String CAMERA_PARAM_ORIENTATION =
"orientation";

```

LAMPIRAN 2

```

    private static final String CAMERA_PARAM_LANDSCAPE =
"landscape";
    private static final String CAMERA_PARAM_PORTRAIT =
"portrait";
    protected Activity mActivity;

    public Preview(Context context) {
        super(context);
        mActivity = (Activity) context;
        mSurfaceView = new SurfaceView(context);
        addView(mSurfaceView);

        // Install a SurfaceHolder.Callback so we get notified
when the
        // underlying surface is created and destroyed.
        mHolder = mSurfaceView.getHolder();
        mHolder.addCallback(this);
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    public void setCamera(Camera camera) {
        mCamera = camera;
        if (mCamera != null) {
            mSupportedPreviewSizes =
mCamera.getParameters().getSupportedPreviewSizes();
            mPictureSizeList =
mCamera.getParameters().getSupportedPictureSizes();
            requestLayout();
        }
    }

    public void switchCamera(Camera camera) {
        setCamera(camera);
        try {
            camera.setPreviewDisplay(mHolder);
        } catch (IOException exception) {
            Log.e(TAG, "IOException caused by
setPreviewDisplay()", exception);
        }
        Camera.Parameters parameters = camera.getParameters();
        parameters.setPreviewSize(mPreviewSize.width,
mPreviewSize.height);
        requestLayout();

        camera.setParameters(parameters);
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int
heightMeasureSpec) {
        // We purposely disregard child measurements because act
as a
        // wrapper to a SurfaceView that centers the camera
preview instead
        // of stretching it.
        int widthMode = MeasureSpec.getMode(widthMeasureSpec);
        int widthSize = MeasureSpec.getSize(widthMeasureSpec);
        int heightMode = MeasureSpec.getMode(heightMeasureSpec);

```

LAMPIRAN 2

```

        int heightSize = MeasureSpec.getSize(heightMeasureSpec);

        int size;
        if (widthMode == MeasureSpec.EXACTLY && widthSize > 0) {
            size = widthSize;
        } else if (heightMode == MeasureSpec.EXACTLY && heightSize
> 0) {
            size = heightSize;
        } else {
            size = widthSize < heightSize ? widthSize :
heightSize;
        }

        int finalMeasureSpec = MeasureSpec.makeMeasureSpec(size,
MeasureSpec.EXACTLY);
        super.onMeasure(finalMeasureSpec, finalMeasureSpec);

        if (mSupportedPreviewSizes != null) {
            mPreviewSize = getOptimalSize(mSupportedPreviewSizes,
widthSize,
                heightSize);
        }
        if (mPictureSizeList != null) {
            mPictureSize = getOptimalSize(mPictureSizeList,
widthSize, heightSize);
        }
    }

    @SuppressWarnings("ClickableViewAccessibility")
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        // TODO Auto-generated method stub
        if (mCamera == null) {
            return true;
        }
        Camera.Parameters params = mCamera.getParameters();
        int action = event.getAction();

        if (event.getPointerCount() > 1) {
            // handle multi-touch events
            if (action == MotionEvent.ACTION_POINTER_DOWN) {
                mDist = getFingerSpacing(event);
            } else if (action == MotionEvent.ACTION_MOVE &&
params.isZoomSupported()) {
                mCamera.cancelAutoFocus();
                handleZoom(event, params);
            }
        } else {
            // handle single touch events
            if (action == MotionEvent.ACTION_UP) {
                handleFocus(event, params);
            }
        }
        return true;
    }

    @Override

```

LAMPIRAN 2

```

        protected void onLayout(boolean changed, int l, int t, int r,
int b) {
            if (changed && getChildCount() > 0) {
                final View child = getChildAt(0);

                final int width = r - l;
                final int height = b - t;

                int previewWidth = width;
                int previewHeight = height;
                if (mPreviewSize != null) {
                    previewWidth = mPreviewSize.width;
                    previewHeight = mPreviewSize.height;
                }

                // Center the child SurfaceView within the parent.
                if (width * previewHeight > height * previewWidth) {
                    final int scaledChildWidth = previewWidth * height
/ previewHeight;
                    child.layout((width - scaledChildWidth) / 2, 0,
                                (width + scaledChildWidth) / 2, height);
                } else {
                    final int scaledChildHeight = previewHeight *
width / previewWidth;
                    child.layout(0, (height - scaledChildHeight) / 2,
width,
                                (height + scaledChildHeight) / 2);
                }
            }
        }

        protected void configureCameraParameter(Camera.Parameters
cameraParams,
                                                boolean portrait) {
            if (Build.VERSION.SDK_INT < Build.VERSION_CODES.FROYO) {
// for 2.1 and
                // before
                if (portrait) {
                    cameraParams.set(CAMERA_PARAM_ORIENTATION,
CAMERA_PARAM_PORTRAIT);
                } else {
                    cameraParams.set(CAMERA_PARAM_ORIENTATION,
CAMERA_PARAM_LANDSCAPE);
                }
            } else { // for 2.2 and later
                int angle;
                Display display =
mActivity.getWindowManager().getDefaultDisplay();
                switch (display.getRotation()) {
                    case Surface.ROTATION_0: // This is display
orientation
                        angle = 90; // This is camera orientation
                        break;
                    case Surface.ROTATION_90:
                        angle = 0;
                        break;
                    case Surface.ROTATION_180:
                        angle = 270;

```

LAMPIRAN 2

```

        break;
    case Surface.ROTATION_270:
        angle = 180;
        break;
    default:
        angle = 90;
        break;
    }
    Log.v(TAG, "angle: " + angle);
    mCamera.setDisplayOrientation(angle);
}

        cameraParams.setPreviewSize(mPreviewSize.width,
mPreviewSize.height);
        cameraParams.setPictureSize(mPictureSize.width,
mPictureSize.height);
        cameraParams.setPictureFormat(ImageFormat.JPEG);
        Log.v(TAG, "Preview Actual Size - w: " +
mPreviewSize.width + ", " +
            "h: " + mPreviewSize.height);
        Log.v(TAG, "Picture Actual Size - w: " +
mPictureSize.width + ", " +
            "h: " + mPictureSize.height);

        mCamera.setParameters(cameraParams);
    }

    public void surfaceCreated(SurfaceHolder holder) {
        // The Surface has been created, acquire the camera and
        tell it where
        // to draw.
        try {
            if (mCamera != null) {
                mCamera.setPreviewDisplay(holder);
            }
        } catch (IOException exception) {
            Log.e(TAG, "IOException caused by
setPreviewDisplay()", exception);
        }
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        // Surface will be destroyed when we return, so stop the
        preview.
        if (mCamera != null) {
            try {
                mCamera.stopPreview();
            } catch (Exception e) {
                // TODO: handle exception
            }
        }
    }

    private Size getOptimalSize(List<Size> sizes, int w, int h) {
        final double ASPECT_TOLERANCE = 0.1;
        double targetRatio = (double) w / h;
        if (sizes == null) {
            return null;

```

LAMPIRAN 2

```

    }

    Size optimalSize = null;
    double minDiff = Double.MAX_VALUE;

    int targetHeight = h;

    // Try to find an size match aspect ratio and size
    for (Size size : sizes) {
        double ratio = (double) size.width / size.height;
        if (Math.abs(ratio - targetRatio) > ASPECT_TOLERANCE)
        {
            continue;
        }
        if (Math.abs(size.height - targetHeight) < minDiff) {
            optimalSize = size;
            minDiff = Math.abs(size.height - targetHeight);
        }
    }

    // Cannot find the one match the aspect ratio, ignore the
    requirement
    if (optimalSize == null) {
        minDiff = Double.MAX_VALUE;
        for (Size size : sizes) {
            if (Math.abs(size.height - targetHeight) <
minDiff) {
                optimalSize = size;
                minDiff = Math.abs(size.height -
targetHeight);
            }
        }
    }
    return optimalSize;
}

public void surfaceChanged(SurfaceHolder holder, int format,
int w, int h) {
    // Now that the size is known, set up the camera
    parameters and begin
    // the preview.
    if (mCamera != null) {
        Camera.Parameters parameters =
mCamera.getParameters();
        configureCameraParameter(parameters, false);
        requestLayout();
        mCamera.startPreview();
    }
}

private void handleZoom(MotionEvent event, Camera.Parameters
params) {
    int maxZoom = params.getMaxZoom();
    int zoom = params.getZoom();
    float newDist = getFingerSpacing(event);
    if (newDist > mDist) {
        // zoom in
        if (zoom < maxZoom) {

```

LAMPIRAN 2

```

        zoom++;
    }
    } else if (newDist < mDist) {
        // zoom out
        if (zoom > 0) {
            zoom--;
        }
    }
    mDist = newDist;
    params.setZoom(zoom);
    mCamera.setParameters(params);
}

public void handleFocus(MotionEvent event, Camera.Parameters
params) {

    List<String> supportedFocusModes =
params.getSupportedFocusModes();
    if (supportedFocusModes != null &&
supportedFocusModes.contains(Camera.Parameters.FOCUS_MODE_AUTO)) {
        mCamera.autoFocus(new Camera.AutoFocusCallback() {
            @Override
            public void onAutoFocus(boolean b, Camera camera)
{
                // currently set to auto-focus on single touch
            }
        });
    }
}

/**
 * Determine the space between the first two fingers
 */
private float getFingerSpacing(MotionEvent event) {
    // ...
    float x = event.getX(0) - event.getX(1);
    float y = event.getY(0) - event.getY(1);
    return (float) Math.sqrt(x * x + y * y);
}
}

```

8. Registrar.java

```

public class Registrar {
    private static final String TAG =
Registrar.class.getSimpleName();
    private String SENDER_ID = "777355711385";
    public static final String PROPERTY_REG_ID =
"registration_id";
    private static final String PROPERTY_APP_VERSION =
"appVersion";
    private static final int PLAY_SERVICES_RESOLUTION_REQUEST =
9000;
    private final Activity activity;
    private GoogleCloudMessaging gcm;
    private String regid;
    private final RegistrarListener listener;

```

LAMPIRAN 2

```

    public Registrar(Activity activity, RegistrarListener
listener) {
        this.listener = listener;
        this.activity = activity;
        if (checkPlayServices()) {
            listener.onGooglePlayAvailable();
            gcm = GoogleCloudMessaging.getInstance(activity);
            regid = getRegistrationId(activity);
            if (regid.isEmpty()) {
                registerInBackground();
            } else {
                listener.resultId(regid);
            }
        } else {
            listener.onGooglePlayUnavailable();
            ;
            Log.i(TAG, "No valid Google Play Services APK
found.");
        }
    }

    private void registerInBackground() {
        new RegisterInBackground().execute(null, null);
    }

    private String getRegistrationId(Context context) {
        final SharedPreferences prefs =
getGCMPreferences(context);
        String registrationId = prefs.getString(PROPERTY_REG_ID,
"");
        if (registrationId.isEmpty()) {
            Log.i(TAG, "Registration not found.");
            return "";
        }

        int registrationVersion =
prefs.getInt(PROPERTY_APP_VERSION, Integer.MIN_VALUE);
        int currentVersion = getAppVersion(context);
        if (registrationVersion != currentVersion) {
            Log.i(TAG, "App Was changed.");
            return "";
        }
        return registrationId;
    }

    private static int getAppVersion(Context context) {
        try {
            PackageInfo packageInfo = context.getPackageManager().
getPackageInfo(context.getPackageName(), 0);
            return packageInfo.versionCode;
        } catch (PackageManager.NameNotFoundException e) {
            throw new RuntimeException("Could not get package
name: " + e);
        }
    }

    private SharedPreferences getGCMPreferences(Context context) {

```

LAMPIRAN 2

```

        //MODE_PRIVATE means that only the application creating
        the shared preference can read write the preference
        return context.getSharedPreferences(TAG,
        Context.MODE_PRIVATE);
    }

    private boolean checkPlayServices() {
        int resultCode =
        GooglePlayServicesUtil.isGooglePlayServicesAvailable(activity);
        if (resultCode != ConnectionResult.SUCCESS) {
            if
            (GooglePlayServicesUtil.isUserRecoverableError(resultCode)) {
                GooglePlayServicesUtil.getErrorDialog(resultCode,
                activity,
                PLAY_SERVICES_RESOLUTION_REQUEST).show();
            } else {
                Log.i(TAG, "This device is not supported");
            }
            return false;
        }
        return true;
    }

    private class RegisterInBackground extends AsyncTask<String,
    Void, String> {

        @Override
        protected String doInBackground(String... urls) {
            String msg = "";
            try {
                if (gcm == null) {
                    gcm =
                    GoogleCloudMessaging.getInstance(activity);
                }
                regid = gcm.register(SENDER_ID);
                msg = "Device registered, Registration ID = " +
                regid;

                sendRegistrationIdToBackend();
                storeRegistrationID(activity, regid);
            } catch (IOException ex) {
                msg = "Error: " + ex.getMessage();
            }
            return msg;
        }

        @Override
        protected void onPostExecute(String msg) {
            listener.onRegisterSuccess(msg);
        }
    }

    private void sendRegistrationIdToBackend() {
        new
        RequestTask().execute("http://fristaasrori.esy.es/storeregid.php?r
        egid="
            + regid);
    }

```

LAMPIRAN 2

```

    }

    private void storeRegistrationID(Context context, String
regId) {
        final SharedPreferences prefs =
getGCMPreferences(context);
        int appVersion = getAppVersion(context);
        Log.i(TAG, "Saving regId on app version" + appVersion);
        SharedPreferences.Editor editor = prefs.edit();
        editor.putString(PROPERTY_REG_ID, regId);
        editor.putInt(PROPERTY_APP_VERSION, appVersion);
        editor.commit();
    }

    class RequestTask extends AsyncTask<String, String, String> {

        @Override
        protected String doInBackground(String... uri) {
            HttpClient httpClient = new DefaultHttpClient();
            HttpResponse response;
            String responseString = null;
            try {
                response = httpClient.execute(new
HttpGet(uri[0]));
                StatusLine statusLine = response.getStatusLine();
                if (statusLine.getStatusCode() ==
HttpStatus.SC_OK) {
                    ByteArrayOutputStream out = new
ByteArrayOutputStream();
                    responseString = out.toString();
                    out.close();
                } else {
                    response.getEntity().getContent().close();
                    throw new
IOException(statusLine.getReasonPhrase());
                }
            } catch (ClientProtocolException e) {
                //TODO Handle problems
            } catch (IOException e) {
                //TODO Handle problems
            }
            return responseString;
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);
        }
    }

    public interface RegistrarListener {
        public void onGooglePlayAvailable();

        public void onGooglePlayUnavailable();

        public void onRegisterSuccess(String message);

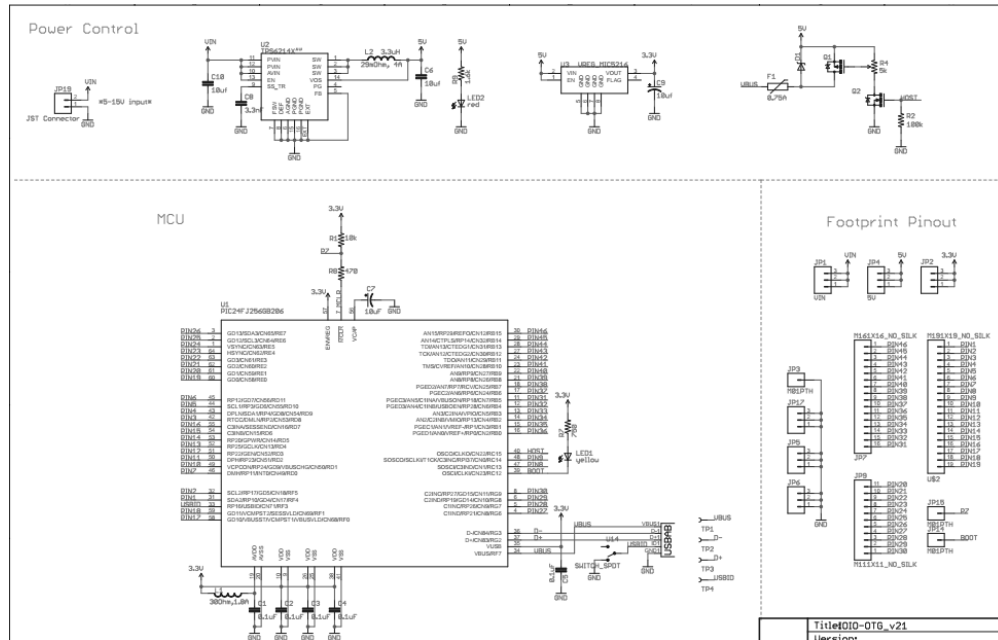
        public void resultId(String id);}}

```

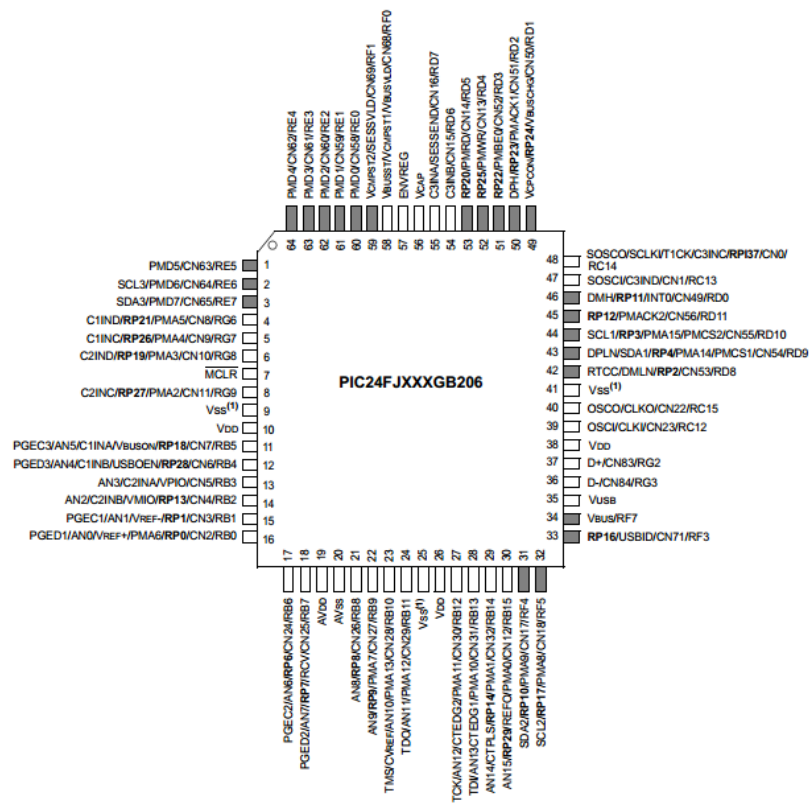
LAMPIRAN 3

Datasheet dan skematik

Skematik IOIO-board



Datasheet PIC24FJ256GB210



LAMPIRAN 3

Skematik modul driver motor DC L298:

