TUGAS AKHIR – TF 141581

RANCANG BANGUN ALAT UKUR WAKTU TEMPUH PERJALANAN BERDASARKAN KECEPATAN KENDARAAN PADA SPEEDOMETER

OKKY INSAN PAMBUDI
NRP. 02311645000021

Dosen Pembimbing
Dr. Ir. Totok Soehartanto, DEA

DEPARTEMEN TEKNIK FISIKA
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini memang dikosongkan.

FINAL PROJECT – TF 141581

DESIGN OF TRAVEL TIME MEASURING INSTRUMENT BASED ON VEHICLES SPEED ON SPEEDOMETER

OKKY INSAN PAMBUDI
NRP. 02311645000021

Supervisor
Dr. Ir. Totok Soehartanto, DEA

ENGINEERING PHYSICS DEPARTMENT
Industrial Technology Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini memang dikosongkan.

PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan dibawah ini :

Nama	: Okky Insan Pambudi
NRP	: 02311645000021
Departemen / Prodi	: Teknik Fisika / S1 Teknik Fisika
Fakultas	: Fakultas Teknologi Industri
Perguruan Tinggi	: Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir dengan judul **"Rancang Bangun Alat Ukur Waktu Tempuh Perjalanan Berdasarkan Kecepatan Kendaraan Pada Speedometer"** adalah benar karya saya sendiri dan bukan plagiat dari karya orang lain. Apabila di kemudian hari terbukti terdapat plagiat pada Tugas Akhir ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Demikian surat ini saya buat dengan sebenar-benarnya.

Surabaya, 23 Juli 2018
Yang membuat pernyataan,

Okky Insan Pambudi
NRP. 02311645000021

Halaman ini memang dikosongkan.

LEMBAR PENGESAHAN

RANCANG BANGUN ALAT UKUR WAKTU TEMPUH PERJALANAN BERDASARKAN KECEPATAN KENDARAAN PADA *SPEEDOMETER*

TUGAS AKHIR

Oleh:

OKKY INSAN PAMBUDI
NRP 02311645000021

Surabaya, 23 Juli 2018

Mengetahui/Menyetujui

Dosen Pembimbing

Dr. Ir. Totok Soehartanto, DEA

NIP. 19650309 199002 1 001

Kepala Departemen
Teknik Fisika FTI-ITS

Agus Muhamad Hatta, S.T., M.Si., Ph.D.

NIP. 19780902 200312 1 002

Halaman ini memang dikosongkan.

LEMBAR PENGESAHAN
RANCANG BANGUN ALAT UKUR WAKTU TEMPUH PERJALANAN BERDASARKAN KECEPATAN KENDARAAN PADA *SPEEDOMETER*

TUGAS AKHIR
Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Bidang Studi Rekayasa Instrumentasi
Program Studi S-1 Departemen Teknik Fisika
Departemen Teknik Fisika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:
OKKY INSAN PAMBUDI
NRP 02311645000021

Disetujui oleh Tim Penguji Tugas Akhir:

1. Dr. Ir. Totok Soehartanto, DEA ..... (Pembimbing)
2. Dr. Ir. Purwadi Agus Darwito, M.Sc. ..... (Penguji I)
3. Dr. Suyanto, S.T., M.T. ..... (Penguji II)
4. Ir. Zulkifli, M.Sc. ..... (Penguji III)

SURABAYA
JULI, 2018

Halaman ini memang dikosongkan.

RANCANG BANGUN ALAT UKUR WAKTU TEMPUH PERJALANAN BERDASARKAN KECEPATAN KENDARAAN PADA *SPEEDOMETER*

Nama Mahasiswa : OKKY INSAN PAMBUDI
NRP : 02311645000021
Jurusan : Teknik Fisika FTI - ITS
Dosen Pembimbing : Dr. Ir. TOTOK SOEHARTANTO, DEA

Abstrak

Untuk mendapatkan informasi waktu tempuh menuju suatu lokasi, umumnya digunakan aplikasi navigasi dari google maps, namun informasi tersebut hanyalah sebuah estimasi yang didapatkan dari metode crowdsourcing dari waktu perpindahan rata-rata pengguna google maps pada suatu ruas jalan. Nilai estimasi tersebut tidak dapat sepenuhnya menjadi acuan dikarenakan nilai tersebut adalah nilai rata-rata seluruh pengguna google maps yang melewati ruas jalan tersebut, baik itu sepeda motor, mobil maupun kendaraan yang lebih besar yang pastinya memiliki kemampuan mobilitas yang berbeda, terlebih dengan karakteristik pengemudi yang berbeda pula.

Dalam penelitian ini, alat ukur waktu tempuh di rancang menggunakan integrasi rute pada google maps dan kecepatan kendaraan yang terukur pada speedometer, sehingga didapatkan waktu tempuh menuju lokasi tujuan secara real-time berdasarkan kecepatan kendaraan yang terukur pada speedometer. Sistem pengukuran kecepatan menggunakan hall effect sensor untuk menghitung putaran roda sepeda motor, modul GPS U-blox Neo 6M sebagai sensor posisi dan modul ESP 8266 sebagai pengirim data menuju web server berbasis Thingspeak. Sistem pengukuran tersebut kemudian diintegrasikan dengan google maps sebagai penyedia informasi rute dan jarak tempuh menuju suatu lokasi yang disajikan pada aplikasi berbasis android. Untuk mengetahui kinerja alat ini, dilakukan pengujian validasi sensor kecepatan, validasi sensor GPS dan pengujian integrasi sistem dengan kendaraan berjalan pada kondisi lalu lintas real.

Halaman ini memang dikosongkan.

Hasil penelitian menunjukkan, alat ukur waktu tempuh perjalanan ini telah berhasil melakukan pengukuran yang diperbarui secara real-time, berdasarkan perubahan kecepatan kendaraan yang terukur pada sistem pengukuran kecepatan pada speedometer, yang diakibatkan oleh perubahan kondisi lalu lintas yang dilalui. Terdapat selisih sekitar 3 menit dari hasil pengukuran bila dibandingkan dengan estimasi google maps yang menunjukkan rata-rata hasil pengukuran menunjukkan waktu tempuh yang lebih rendah dari google maps. Hasil pengujian sensor kecepatan menunjukkan akurasi sebesar 97,26%, sensitifitas 0,2009 km/h tiap hertz frekuensinya, presentase error sistem pengukuran kecepatan sebesar 4.22% dan presentase keberhasilan sistem pengiriman data sebesar 83.33%.

Kata Kunci: Waktu Tempuh, Speedometer, Global Positioning System, Google Maps, Android

Halaman ini memang dikosongkan.

DESIGN OF TRAVEL TIME MEASURING INSTRUMENT BASED ON VEHICLES SPEEDOMETER

Name	**: OKKY INSAN PAMBUDI**
NRP	**: 02311645000021**
Department	**: Engineering Physics FTI-ITS**
Supervisor	**: Dr. Ir. TOTOK SOEHARTANTO, DEA**

Abstract

To find out information of travel time to a certain location, Google Maps is a commonly used navigation application. However, the information is only an estimation obtained from crowdsourcing method from time of moving average google maps user on a road. The estimated value can not be fully referenced because the value is the average value of all google maps users passing through the road, such as motorcycles, cars or larger vehicles that certainly have different mobility capabilities, with different characteristics of the driver too.

Inthis study, travel time measuring instrument is designed using route integration on google maps and measured vehicle speed on the speedometer, so that the travel time to the destination shown in real-time based on the speed of the vehicle as in speedometer. The speed measurement system uses a hall effect sensor to calculate motorcycle wheel spins, U-blox Neo 6M GPS module as position sensor and ESP 8266 module as a data sender to a Thingspeak-based web server. The measurement system then integrated with google maps as a provider of route information and mileage to a location presented on android based applications. To ascertain the performance of this tool, validation testing of speed sensor, GPS sensor validation and system integration testing with the vehicle running on real traffic conditions were conducted.

The results showed that this travel time measuring instrument has successfully performed the measurement updated in real-time, based on changes in vehicle speed measured on speedometer

Halaman ini memang dikosongkan.

measurement system, which is caused by changes in traffic conditions. There is about 3 minutes differences in measurement results when compared with the estimated google maps that show the average measurement results show a lower travel time than google maps. The results of the speed sensor test shows the accuration is about 97.26%, the sensitivity about 0.2009 km/h/f, the percentage of error measurement system speed of 4.22% and the success rate of data delivery system of 83.33%.

Keyword*: *Travel Time, Speedometer, Global Positioning System, Google Maps, Android

Halaman ini memang dikosongkan.

KATA PENGANTAR

Puji Syukur Alhamdulillah penulis panjatkan pada Allah SWT, atas rahmat dan hidayah-Nya hingga terselesaikannya perancangan dan penyusunan laporan Tugas Akhir yang berjudul RANCANG BANGUN ALAT UKUR WAKTU TEMPUH PERJALANAN BERDASARKAN KECEPATAN KENDARAAN PADA *SPEEDOMETER* dapat diselesaikan tepat waktu.

Penyusunan laporan ini tak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu penulis mengucapkan terimakasih kepada:

1. Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir dan Laporan Tugas Akhir ini.
2. Kedua orang tua penulis yang selalu memberikan dukungan baik moril maupun materiil serta memberikan do'a dan semangat.
3. Bapak Dr. Ir. Totok Soehartanto, DEA, selaku dosen pembimbing yang selama ini telah memberikan bimbingan, dukungan, saran dan kritik dengan penuh semangat, ketulusan dan kebijaksanaan.
4. Kepala dan segenap Dosen Departemen Teknik Fisika-ITS yang telah memberikan bimbingan dan pengetahuan selama proses perkuliahan.
5. Berbagai pihak yang tak bisa penulis sebutkan satu persatu.

Penulis menyadari bahwa penulisan pada tugas akhir ini belum sempurna. Namun, penulis berharap semoga laporan Tugas Akhir yang telah berhasil diselesaikan ini memiliki manfaat yang besar bagi diri sendiri, Jurusan Teknik Fisika ITS, dan pembaca.

Surabaya, 23 Juli 2018

Penulis

Halaman ini memang dikosongkan.

DAFTAR ISI

Halam Judul.....i
Pernyataan Bebas Plagiarisme.....v
Lembar Pengesahan.....vii
Abstrak.....xi
Kata Pengantar.....xix
Daftar Isi.....xxi
Daftar Gambar.....xxv
Daftar Tabel.....xxix
Bab I. Pendahuluan.....1
1.1 Latar Belakang.....1
1.2 Perumusan Masalah.....3
1.3 Batasan Masalah.....3
1.4 Tujuan Tugas Akhir.....4
1.5 Sistematika Laporan.....4
BAB II DASAR TEORI.....5
2.1 *Google Maps*.....5
2.2 Waktu Tempuh Perjalanan.....7
2.3 Sistem Pengukuran Kecepatan.....8
2.4 *Global Positioning System* (GPS).....10
2.5 Mikrokontroller Arduino.....12
2.6 Modul WiFi ESP 8266.....15
2.7 *Thingspeak*.....16
2.8 Android.....18
BAB III METODOLOGI PENELITIAN.....21
3.1 Studi Literatur.....22
3.2 Perancangan Sistem.....22
3.2.1 Perancangan Sistem Pengukuran Kecepatan.....24
3.2.2 Perancangan Sensor GPS.....26
3.2.3 Perancangan Sistem Pengiriman Data.....27
3.3 Pengujian Komponen Sistem Pengukuran.....30
3.3.1 Pengujian Sistem Pengukuran Kecepatan.....30
3.3.2 Pengujian Sistem GPS.....31
3.3.3 Pengujian Sistem Pengiriman Data.....32
3.4 Perancangan Sistem Perhitungan Waktu Tempuh.....32

Halaman ini memang dikosongkan.

3.4.1 Perancangan Algoritma Waktu Tempuh 33
3.4.2 Integrasi Sistem Pengukuran dengan *Google* 35
3.4.3 Desain *Graphical User Interface* (GUI) 37
3.5 Pengujian Sistem Perhitungan Waktu Tempuh 41
3.5.1 Pengujian Masukan Aplikasi Android 42
3.5.2 Pengujian Integrasi Sistem 42
BAB IV PENGUJIAN DAN ANALISA 45
4.1 Pengujian Komponen Sistem 45
4.1.1 Pengujian Sistem Pengukuran Kecepatan 45
4.1.2 Pengujian Sistem GPS 51
4.1.3 Pengujian Sistem Pengiriman Data 53
4.1.4 Pengujian Masukan Aplikasi Android 55
4.2 Pengujian Integrasi Sistem 58
4.3 Analisa Data 81
4.3.1 Analisa Kinerja Komponen Penyusun Sistem 82
4.3.2 Analisa Kinerja Sistem Integrasi 84
BAB V PENUTUP 91
5.1 Kesimpulan 91
5.2 Saran 92
DAFTAR PUSTAKA
LAMPIRAN A
LAMPIRAN B

Halaman ini memang dikosongkan.

DAFTAR GAMBAR

Gambar 2. 1 Tampilan *Google Maps* pada PC 5
Gambar 2. 2 Ilustrasi Sensor *Wheel Speed* 9
Gambar 2. 3 Ilustrasi Satelit *Global Positioning System* (GPS) 11
Gambar 2. 4 Modul GPS U-blox Neo 6M 12
Gambar 2. 5 Mikrokontroller Arduino Uno 13
Gambar 2. 6 Blok Diagram Sistem 14
Gambar 2. 7 Modul Wireless ESP 8266 15
Gambar 2. 8 Hubungan Jaringan Arduino dan Thingspeak 17
Gambar 2. 9 Blok Diagram Sistem 18
Gambar 2. 10 Blok Diagram Aplikasi Android 19
Gambar 3. 1 *Flowchart* Perancangan Sistem 21
Gambar 3. 2 Blok Diagram Sistem 22
Gambar 3. 3 Blok Diagram Sistem 23
Gambar 3. 4 Sensor *Wheel Speed* 25
Gambar 3. 5 Perancangan Sensor Kecepatan 25
Gambar 3. 6 Perancangan GPS 27
Gambar 3. 7 Konfigurasi Pin ESP 8266 pada Arduino 28
Gambar 3. 8 Proses Pembuatan *Channel* pada *Thingspeak* 29
Gambar 3. 9 Tampilan *Field* pada *Thingspeak* 30
Gambar 3. 10 Blok Diagram Aplikasi Android 33
Gambar 3. 11 Diagram Alir Algoritma Waktu Tempuh 34
Gambar 3. 12 Cuplikan Program Perhitungan Waktu 36
Gambar 3. 13 Perancangan Tampilan Pembuka 37
Gambar 3. 14 *Running* Tampilan Halaman Pembuka 38
Gambar 3. 15 Perancangan Halaman *Input* Lokasi Tujuan 39
Gambar 3. 16 *Running* Tampilan Halaman Utama 39
Gambar 3. 17 Perancangan *Depart Page* 40
Gambar 3. 18 *Running* Tampilan *Depart Page* 41
Gambar 4. 1 Grafik Hasil Pengujian Sensitifitas Sensor 47
Gambar 4. 2 Lintasan Pengujian Sensor Kecepatan 48

Halaman ini memang dikosongkan.

Gambar 4. 3 Hasil Validasi GPS pada *Google Maps* 53
Gambar 4. 4 Proses Pengujian Masukan Aplikasi Android 56
Gambar 4. 5 Tampilan Masukan pada Aplikasi Android 57
Gambar 4. 6 Tampilan Awal Pengujian Rute 1 Pertama 59
Gambar 4. 7 Grafik Waktu Tempuh Rute 1 Pertama 52
Gambar 4. 8 Tampilan Saat dibandingkan *Google Maps* 61
Gambar 4. 9 Tampilan Saat Pengujian Rute 1 Kedua 62
Gambar 4. 10 Grafik Waktu Tempuh Rute 1 Kedua 63
Gambar 4. 11 Tampilan Awal Pengujian Rute 2 Pertama 64
Gambar 4. 12 Grafik Waktu Tempuh Rute 2 Pertama 65
Gambar 4. 13 Tampilan Saat dibandingkan *Google Maps* 66
Gambar 4. 14 Tampilan Saat Pengujian Rute 2 Kedua 67
Gambar 4. 15 Grafik Waktu Tempuh Rute 2 Kedua 68
Gambar 4. 16 Tampilan Saat dibandingkan *Google Maps* 69
Gambar 4. 17 Tampilan Saat dibandingkan *Google Maps* 70
Gambar 4. 18 Tampilan Saat dibandingkan *Google Maps* 71
Gambar 4. 19 Tampilan Awal Pengujian Rute 3 Pertama 72
Gambar 4. 20 Grafik Waktu Tempuh Rute 3 Pertama 73
Gambar 4. 21 Tampilan Saat dibandingkan *Google Maps* 74
Gambar 4. 22 Tampilan Saat Pengujian Rute 3 Kedua 75
Gambar 4. 23 Grafik Waktu Tempuh Rute 3 Kedua 76
Gambar 4. 24 Grafik Waktu Tempuh Rute 4 Pertama 77
Gambar 4. 25 Tampilan Saat dibandingkan Google Maps 78
Gambar 4. 26 Grafik Waktu Tempuh Rute 4 Kedua 79
Gambar 4. 27 Tampilan Saat dibandingkan Google Maps 80
Gambar 4. 28 Grafik Hubungan Jarak dan Selisih Waktu 81

Halaman ini memang dikosongkan.

DAFTAR TABEL

Tabel 4. 1 Pengujian Akurasi Sensor Kecepatan 46
Tabel 4. 2 Pengujian Sensitifitas Sensor Kecepatan 47
Tabel 4. 3 Pengujian Sensor pada Kecepatan 20 km/h 49
Tabel 4. 4 Pengujian Sensor pada Kecepatan 40 km/h 50
Tabel 4. 5 Pengujian Sensor pada Kecepatan 60 km/h 51
Tabel 4. 6 Pembacaan Sensor GPS 52
Tabel 4. 7 Penerimaan Data pada Web Server 54

Halaman ini memang dikosongkan.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Lalu lintas adalah sarana untuk bergerak dari satu tempat ke tempat yang lain, oleh karena itu lalu lintas merupakan salah satu masalah penting. Apabila arus lalu lintas terganggu atau terjadi kemacetan, maka berbagai permasalahan dapat timbul karenanya. Data lalu lintas yang dihimpun oleh Dinas Perhubungan Kota Surabaya pada akhir tahun 2015 saja jumlah kendaraan yang melewati wilayah Surabaya terus mengalami peningkatan yang cukup signifikan. Pada semester awal tahun 2015 tercatat lalu lintas harian rata-rata (LHR) sebanyak 21.178 satuan mobil penumpang (SMP) per hari. Sedangkan di semester dua di tahun yang sama meningkat sebanyak 37.319 SMP per hari. Dari data tersebut dapat diperkirakan kemacetan dapat bertumbuh seiring dengan peningkatan lalu lintas tanpa diiringi penambahan volume jalan (Hibban, 2014). Kemacetan itu dapat menimbulkan dampak serius terlebih untuk mobilitas kendaraan yang memiliki urgensi untuk tiba di tempat tujuan tepat pada waktunya, contohnya *ambulance*, pemadam kebakaran ataupun jasa pengiriman.

Untuk mendapatkan informasi waktu tempuh menuju suatu lokasi, umumnya digunakan aplikasi navigasi dari *google maps*. *Google* mengestimasi waktu berdasarkan berbagai hal, tergantung pada data yang tersedia di area tertentu. Hal-hal ini berkisar dari batas kecepatan resmi dan kecepatan yang disarankan, kemungkinan kecepatan yang berasal dari jenis jalan, data kecepatan rata-rata historis selama periode waktu tertentu, waktu perjalanan sebenarnya dari pengguna sebelumnya, dan waktu nyata informasi lalu lintas (Bar-Shalom, 2014). *Google* mencampur data dari sumber manapun yang dimiliki dan menghasilkan prediksi terbaik yang dapat dibuat. Ketika pengguna ponsel cerdas mengaktifkan aplikasi *google maps* mereka dengan lokasi GPS diaktifkan, ponsel mengirim kembali bit data secara anonim ke *google*, yang memungkinkan *google* mengetahui seberapa cepat mobil mereka bergerak. *Google maps* secara terus-menerus menggabungkan data yang berasal dari semua mobil di jalan dan

mengirimkannya kembali dengan garis berwarna pada lapisan lalu lintas. Penting untuk diingat bahwa ketika anda melacak rute melalui *google maps*, semua informasi adalah perkiraan (Bar-Shalom, 2014). Menghitung waktu kedatangan adalah masalah prediksi masa depan dan lalu lintas, dimana hal tersebut mengikuti pola tertentu yang pada dasarnya tidak dapat diprediksi. Bahkan jika terdapat pengetahuan lengkap tentang kondisi lalu lintas saat ini dan perubahan yang diketahui (misalkan perbaikan jalan dimulai atau pertandingan sepak bola), tidak ada yang dapat memprediksi jatuhnya pesawat atau sebagainya. Nilai estimasi tersebut juga tidak dapat sepenuhnya menjadi acuan dikarenakan nilai tersebut adalah nilai rata-rata seluruh pengguna *google maps* yang melewati ruas jalan tersebut, baik itu sepeda motor, mobil maupun kendaraan yang lebih besar yang pastinya memiliki kemampuan mobilitas yang berbeda. Selain itu *google maps* tidak dapat mempertimbangkan kecepatan mengemudi pribadi anda misalnya kecepatan yang disukai untuk memprediksi waktu kedatangan, itulah sebabnya waktu tempuh yang disajikan kepada anda mungkin tidak selalu benar-benar akurat.

Pada penelitian kali ini akan dirancang alat ukur waktu tempuh perjalanan berdasarkan kecepatan kendaraan yang terbaca pada *speedometer*. Alat ukur waktu tempuh perjalanan ini akan dapat menghitung waktu tempuh berdasarkan kecepatan *real* dari kendaraan, sehingga dapat menghitung waktu tempuh yang lebih akurat berdasarkan jenis kendaraan yang digunakan dan kecepatan mengemudi pribadi. Pada perancangan ini, nilai kecepatan kendaraan didapatkan langsung melalui pengukuran yang dilakukan oleh sensor *wheel speed* yang terdapat pada sistem *speedometer* kendaraan, yang merupakan sebuah sensor *hall effect*. Keluaran dari sensor tersebut berupa pulsa yang akan diolah oleh mikrokontroller yang kemudian akan dikirim menuju server internet menggunakan modul wifi yang terkoneksi dengan internet. Kemudian data dari *server* tersebut akan ditampilkan pada sebuah aplikasi android yang mengintegrasikan data kecepatan hasil pembacaan sensor pada kendaraan dengan *google maps* yang akan menyediakan data jarak menuju tempat tujuan, sehingga didapati

waktu tempuh perjalanan yang *real-time* berdasarkan nilai kecepatan dari pengukuran *speedometer* dan diintegrasikan dengan jarak dan rute yang akan dilewati menggunakan *google maps*.

Dengan dirancangnya alat ukur waktu tempuh perjalanan berdasarkan kecepatan kendaraan pada *speedometer* ini, diharapkan dapat memiliki manfaat yang besar terutama untuk kendaraan dengan urgensi ketepatan waktu untuk sampai di tempat tujuan tepat pada waktunya yang memiliki perbedaan kecepatan mengemudi dibandingkan kendaraan pada umumnya. Dengan mendapatkan informasi waktu tempuh perjalanan yang diperbarui secara *real-time,* karena berbasis alat ukur yang terdapat pada kendaraan berdasarkan kecepatan *real* kendaraan tersebut, diharapkan dapat membantu kendaraan sampai pada tempat tujuan tepat pada waktunya .

1.2 Perumusan Masalah

Adapun beberapa permasalahan yang terdapat dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Bagaimana merancang sistem pengukuran kecepatan kendaraan dari *speedometer* yang dapat diolah oleh *handphone* berbasis android?
2. Bagaimana mengintegrasikan informasi jarak dan rute yang akan dilewati dari *google maps* dengan data kecepatan yang terukur agar dapat menjadi informasi waktu tempuh?

1.3 Batasan Masalah

Pada tugas akhir ini, terdapat beberapa batasan yang digunakan antara lain:

1. Sistem operasi *handphone* yang akan digunakan berbasis android.
2. *Speedometer* sepeda motor akan direkayasa agar dapat terhubung dan terbaca kecepatannya oleh *handphone* berbasis android.
3. Data jarak dan rute yang akan dilewati dari *google maps* diintegrasikan dengan kecepatan kendaraan yang terukur dari *speedometer* untuk mendapatkan waktu tempuh.

4. Pada saat kendaraan dalam kondisi belum berjalan atau berhenti, waktu tempuh tertampil hasil estimasi *google maps*.

1.4 Tujuan Tugas Akhir

Adapun tujuan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Merancang sistem pengukuran kecepatan kendaraan dari *speedometer* yang terhubung dengan *handphone* berbasis android.
2. Mengintegrasikan informasi jarak dan rute yang akan dilewati dari google maps dengan data kecepatan yang terukur untuk diolah menjadi informasi waktu tempuh dan ditampilkan pada sebuah aplikasi berbasis android.

1.5 Sistematika Laporan

Secara sistematis, penyusunan laporan tugas akhir ini tersusun dalam 5 Bab dengan penjelasan sebagai berikut:

BAB I Pendahuluan

Bab ini berisi latar belakang, perumusan masalah, batasan masalah , tujuan penelitian, dan sistematika laporan.

BAB II Dasar Teori

Bab ini berisi mengenai teori-teori penunjang yang terkait dalam penulisan tugas akhir.

BAB III Metodologi Penelitian

Bab ini akan dijelaskan mengenai langkah-langkah yang telah dilakukan dalam penelitian.

BAB IV Pengujian dan Analisa

Bab ini akan ditampilkan data dan analisa hasil pengujian beserta pembahasannya.

BAB V Penutup

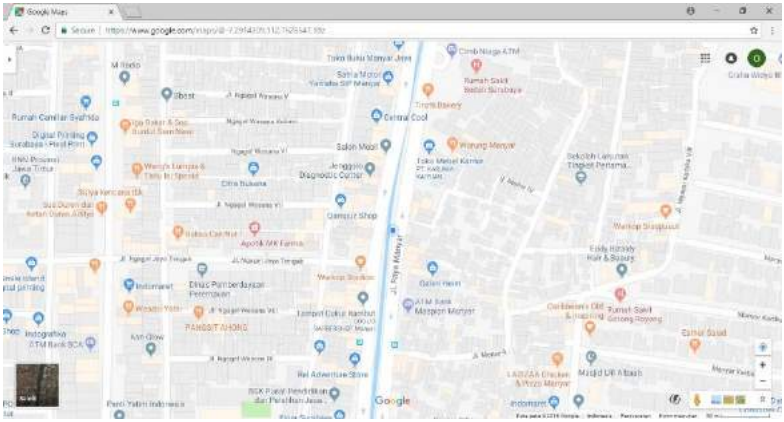
Bab ini berisi tentang kesimpulan pokok dari seluruh rangakaian penelitian yang telah dilakukan dan saran yang dapat dijadikan sebagai pengembangan penelitian selanjutnya.

BAB II
DASAR TEORI

Bab ini berisi teori-teori yang menunjang penelitian terkait *speedometer*, mikrokontroller arduino, android, *google maps* dan perhitungan waktu tempuh.

2.1 *Google Maps*

Google Maps adalah layanan pemetaan yang dikembangkan oleh Google. Layanan ini memberikan citra satelit, peta jalan, panorama 360°, kondisi lalu lintas, dan perencanaan rute untuk bepergian dengan berjalan kaki, mobil, sepeda (versi beta), atau angkutan umum. *Google Maps* menawarkan API (*application programming interface*) yang memungkinkan peta untuk dimasukkan pada situs web pihak ketiga dan menawarkan penunjuk lokasi untuk bisnis perkotaan dan organisasi lainnya di berbagai negara di seluruh dunia. *Google Map Maker* memungkinkan pengguna untuk bersama-sama mengembangkan dan memperbarui pemetaan layanan di seluruh dunia (Bar-Shalom, 2014).



Gambar 2. 1. Tampilan *Google Maps* pada PC

Pada gambar 2.1 menampilkan antarmuka *google maps* pada PC, pada gambar tersebut dapat kita lihat sebagian peta Kota

Surabaya pada *google maps*. *Google maps* dapat menunjukkan tampilan jalan, dari jalan raya hingga jalan yang lebih kecil di pemukiman. Terdapat pula tampilan *landmark* pada *google maps* tersebut, semisal tempat usaha, tempat hiburan umum dan sebagainya. Tampilan jalan tersebut dapat dimanfaatkan pada salah satu fitur *google maps* yaitu *google navigation* sebagai panduan rute menuju suatu lokasi.

Pada fitur navigasi *google*, kita dapat memperoleh estimasi waktu yang dibutuhkan untuk menuju suatu lokasi. *Google* mengestimasi waktu berdasarkan berbagai hal, tergantung pada data yang tersedia di area tertentu. Hal-hal ini berkisar dari batas kecepatan resmi dan kecepatan yang disarankan, kemungkinan kecepatan yang berasal dari jenis jalan, data kecepatan rata-rata historis selama periode waktu tertentu, waktu perjalanan sebenarnya dari pengguna sebelumnya, dan waktu nyata informasi lalu lintas. *Google* mencampur data dari sumber mana pun yang dimiliki dan menghasilkan prediksi terbaik yang dapat dibuat. Ketika pengguna ponsel cerdas mengaktifkan aplikasi *google maps* mereka dengan lokasi GPS diaktifkan, ponsel mengirim kembali bit data secara anonim ke *google*, yang memungkinkan *google* mengetahui seberapa cepat mobil mereka bergerak. *Google Maps* secara terus-menerus menggabungkan data yang berasal dari semua kendaraan di jalan dan mengirimkannya kembali dengan garis berwarna pada lapisan lalu lintas.

Penting untuk diingat bahwa ketika anda melacak rute melalui *google maps*, semua informasi adalah perkiraan. Menghitung waktu kedatangan adalah masalah prediksi masa depan dan lalu lintas, dimana hal tersebut mengikuti pola tertentu yang pada dasarnya tidak dapat diprediksi (Bar-Shalom, 2014). Bahkan jika terdapat pengetahuan lengkap tentang kondisi lalu lintas saat ini dan perubahan yang diketahui (misalkan perbaikan jalan dimulai atau pertandingan sepak bola), tidak ada yang dapat memprediksi jatuhnya pesawat atau sebagainya. Nilai estimasi tersebut juga tidak dapat sepenuhnya menjadi acuan dikarenakan nilai tersebut adalah nilai rata-rata seluruh pengguna *google maps* yang melewati ruas jalan tersebut, baik itu sepeda motor, mobil maupun kendaraan

yang lebih besar yang pastinya memiliki kemampuan mobilitas yang berbeda. Selain itu *google maps* tidak dapat mempertimbangkan kecepatan mengemudi pribadi anda misalnya kecepatan yang disukai untuk memprediksi waktu kedatangan, itulah sebabnya waktu tempuh yang disajikan kepada anda mungkin tidak selalu benar-benar akurat.

2.2 Waktu Tempuh Perjalanan

Waktu tempuh perjalanan (*travel time*) adalah waktu total yang diperlukan untuk melewati suatu panjang jalan tertentu, untuk mencapai lokasi tujuan tertentu (Poerwanto, 2013). Ukurannya dapat dinyatakan dengan satuan detik (sekon), menit, jam, hari, pekan dan seterusnya. Waktu tempuh perjalanan dibutuhkan untuk mengetahui berapa lama waktu yang dibutuhkan untuk mencapai suatu lokasi, atau pukul berapa kita akan tiba di lokasi yang dituju. Untuk menghitung waktu tempuh perjalanan dibutuhkan dua variabel, yaitu jarak dan kecepatan. Persamaan untuk menghitung waktu tempuh yaitu :

$$\boldsymbol{t = \frac{s}{v}} \qquad (2.1)$$

Dimana :

t : waktu (sekon)
s : jarak (m)
v : kecepatan (m/s)

Pada persamaan 2.1, t adalah waktu yang dibutuhkan untuk menempuh perjalanan dari lokasi asal menuju lokasi tujuan, s adalah jarak antara lokasi asal menuju lokasi tujuan dan v adalah kecepatan kendaraan. Sehingga untuk mendapatkan informasi waktu tempuh perjalanan, dibutuhkan nilai kecepatan kendaraan yang diperoleh dari hasil perhitungan mikrokontroller terhadap frekuensi sinyal yang dikirim oleh sensor *wheel speed* pada kendaraan. Sedangkan jarak diperoleh dari informasi rute yang akan dilewati dari *google maps*, berdasarkan posisi kendaraan yang diperoleh dari GPS dan lokasi yang akan dituju

2.3 Sistem Pengukuran Kecepatan

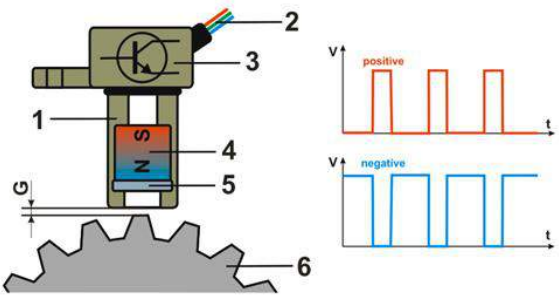
Pengukuran (*measurement*) merupakan kegiatan yang bertujuan untuk menentukan nilai suatu besaran dalam bentuk angka (kuantitatif). Jadi mengukur adalah suatu proses mengaitkan angka secara empiris dan obyektif pada sifat-sifat obyek atau kejadian nyata sehingga angka yang diperoleh tersebut dapat memberikan gambaran yang jelas mengenai obyek atau kejadian yang diukur (Poerwanto, 2013). Kecepatan adalah besaran vektor yang menunjukkan seberapa cepat benda berpindah. Besar dari vektor ini disebut dengan kelajuan dan dinyatakan dalam satuan meter per sekon (m/s atau ms^{-1}). Ada beberapa jenis sensor pengukuran kecepatan, diantaranya :

- Mekanis, adalah perangkat pengukur kecepatan yang dihubungkan langsung dengan roda depan ataupun transmisi dengan menggunakan suatu kabel yang ikut berputar saat kendaraan bergerak, gerakan berputar ini kemudian diubah untuk menggerakkan jarak kecepatan.
- Elektronik, adalah pengukur kecepatan yang bekerja atas dasar sensor yang ditempatkan di poros penggerak kendaraan yang medeteksi jumlah putaran poros untuk selanjutnya data dikirim ke *speedometer* dengan prinsip arus Eddy yang menggerakkan jarum kecepatan ataupun menunjukkan kecepatan secara digital.
- GPS, adalah perangkat pengukur kecepatan yang menggunakan perubahan data posisi koordinat bumi yang diperoleh dari satelit GPS yang diolah oleh prosesor menjadi informasi kecepatan.

2.3.1 *Speedometer* menggunakan *Hall Effect Sensor*

Ada berbagai macam sensor yang dapat digunakan untuk mengukur kecepatan yang dapat diimplementasi sebagai sistem penginderaan *speedometer*, salah satunya adalah sensor *Wheel Speed* (37700-K48-A01) seperti yang terdapat pada sepeda motor honda vario. Sensor *wheel speed* pada dasarnya adalah sebuah sensor *hall effect* yang dirancang untuk dapat mengukur jumlah putaran dari gerigi sepeda motor agar dapat dikonversikan dalam bentuk *speedometer*. Modul sensor ini memiliki keluaran digital

berupa pulsa dan sudah terkalibrasi, sehingga dapat diolah pada mikrokontroller.



Gambar 2. 2. Ilustrasi Sensor *Wheel Speed*

Gambar 2.2 mengilustrasikan sensor *wheel speed.* Saat roda gigi berputar, ujung dari mata gigi yang berdekatan dengan sensor memberikan perubahan medan magnet sehingga menginduksi komponen dalam sensor dan menghasilkan sinyal listrik. Tegangan output yang dihasilkan berada dalam kisaran mili volt (mV) dan diperkuat oleh rangkaian elektronik terpadu yang berada di dalam sensor. Sinyal tegangan keluaran akhir berupa gelombang digital (bentuk persegi). Sinyal keluaran dari sensor dapat berupa positif atau negatif dengan tegangan puncak biasanya sampai 5 V atau 12 V, tergantung pada jenis elektronik dan persyaratan terpadu dari sistem yang digunakan. Amplitudo sinyal output tetap konstan, hanya frekuensi yang meningkat secara proporsional dengan RPM. Sensor dihubungkan dengan tegangan 12 volt yang dibutuhkan untuk rangkaian elektronik penguat (Assidqi N. R. & Soehartanto T., 2014).

Untuk mendapatkan nilai kecepatan kendaraan, dibutuhkan penghitung keluaran sensor tersebut. Nilai kecepatan didapatkan dari perhitungan frekuensi keluaran sensor tersebut per satuan detik. Keluaran dari sensor yang berupa frekuensi tersebut dapat menjadi nilai besaran kecepatan dengan menjadikan keluaran sensor tersebut menjadi masukan pada penghitung frekuensi yang terdapat pada mikrokontroller arduino. Keluaran dari sensor yang

berupa frekuensi tersebut membutuhkan persamaan sehingga dapat menjadi bentuk kecepatan dalam satuan km/h. Karena sensor tersebut menghitung putaran roda maka pertama-tama digunakan persamaan kecepatan *angular* seperti di bawah ini:

$$\omega = 2\,\pi f \qquad (2.2)$$

Dimana :
ω : kecepatan sudut (rad/s)
f : frekuensi (Hz)

Pada persamaan 2.2, ω adalah kecepatan sudut, yaitu kecepatan putaran roda dengan satuan radian per sekon, f adalah frekuensi yang didapat dari keluaran sensor *wheel speed*. Dengan persamaan 2.2 keluaran sensor yang berupa frekuensi telah dapat diubah menjadi bentuk kecepatan putaran roda dalam satuan radian per sekon. Namun karena data yang dibutuhkan adalah kecepatan linier, maka dibutuhkan untuk mengubah kecepatan anguler menjadi kecepatan linier menggunakan persamaan di bawah ini:

$$v = \omega\, r \qquad (2.3)$$

Dimana :
v : kecepatan linier (m/s)
ω : kecepatan sudut (rad/s)
r : jari-jari lingkaran (m)

Pada persamaan 2.3, v adalah kecepatan linier dengan satuan meter per sekon, ω adalah kecepatan sudut, yaitu kecepatan putaran roda dengan satuan radian per sekon, r adalah adalah jari-jari lingkaran atau dalam penelitian ini jari-jari roda dengan satuan meter. Dengan persamaan 2.3 kecepatan putaran roda dalam satuan radian per sekon telah dapat diubah menjadi kecepatan linier dengan satuan meter per sekon.

2.4 *Global Positioning System* (GPS)

Global positioning system (GPS) adalah sistem navigasi yang berbasiskan satelit yang saling berhubungan yang berada di orbitnya. Satelit-satelit itu milik Departemen Pertahanan (*Departemen of Defense*) Amerika Serikat yang pertama kali

diperkenalkan mulai tahun 1978 dan pada tahun 1994 sudah memakai 24 satelit. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS *reciever* yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. Posisi diubah menjadi titik yang dikenal dengan nama *Way-point* nantinya akan berupa titik-titik koordinat lintang dan bujur dari posisi seseorang atau suatu lokasi kemudian di layar pada peta elektronik.

Gambar 2. 3. Ilustrasi Satelit *Global Positioning System* (GPS)

Gambar 2.3 mengilustrasikan satelit-satelit GPS yang mengelilingi bumi. Satelit-satelit tersebut tersebar mengelilingi seluruh posisi terhadap bumi, sehingga tiap lokasi dapat tertangkap sinyalnya oleh satelit-satelit tersebut. Sebuah GPS *receiver* harus mengunci sinyal minimal tiga satelit untuk menghitung posisi 2D (*latitude* dan *longitude*) dan *track* pergerakan. Jika GPS *reciever* dapat menerima empat atau lebih satelit, maka dapat menghitung posisi 3D (*latitude, longitude* dan *altitude*). Jika sudah dapat menentukan posisi *user*, selanjutnya GPS dapat menghitung informasi lain, seperti kecepatan, arah yang dituju, jalur dan tujuan perjalanan, jarak tujuan, matahari terbit dan matahari terbenam dan masih banyak lagi (Assidqi N. R. & Soehartanto T., 2014).

Sebagai sensor posisi pada perancangan ini dibutuhkan sebuah modul yang bekerja sebagai *receiver* sehingga didapati

informasi lokasi. Terdapat beberapa modul GPS yang dapat dimanfaatkan sebagai *receiver*, tiap jenis modul memiliki kemampuan yang berbeda, semakin baik modul yang digunakan akan berpengaruh terhadap kecepatan *locking* posisi dan keakuratan pembacaan posisi. Adapun modul *global positioning system* (GPS) yang digunakan pada penelitian kali ini adalah modul GPS U-blox Neo 6M.

Gambar 2. 4. Modul GPS U-blox Neo 6M

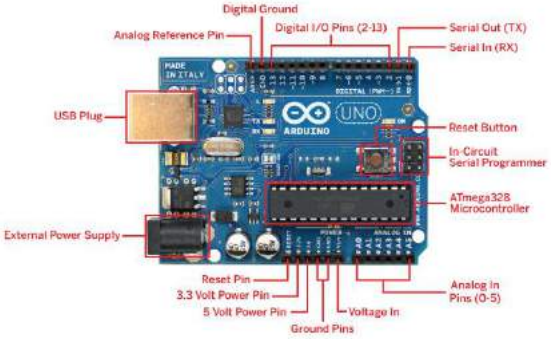
Gambar 2.4 adalah bentuk dari modul GPS U-blox Neo 6M. Modul tersebut digunakan untuk mengetahui letak pada permukaan bumi dengan bantuan sinyal satelit. Ublox neo-6m GPS *module* merupakan modul GPS yang mempunyai karakteristik tegangan masukan *3V-5V*, akurasi posisi *horizontal* 2.5m, dengan antena keramik untuk memperkuat sinyal yang dipancarkan oleh GPS. Keluaran dari modul GPS ini berupa koordinat *latitude* dan *longitude* yang kemudian menjadi masukan bagi mikrokontroller yang akan memanfaatkan informasi koordinat *latitude* dan *longitude* tersebut.

2.5 Mikrokontroller Arduino Uno

Mikrokontroller adalah suatu *integrated circuit* (IC) yang pada umumnya digunakan untuk mengontrol alat tertentu misalnya sistem kontrol mobil atau sistem permesinan serta lain sebagainya. Berbeda dengan mikroprosesor yang pada umumnya digunakan

pada *personal computer* (PC) yang hanya digunakan untuk memproses suatu data masukan atau *input* yang diberikan sesuai dengan namanya yaitu mikroprosesor.

Mikrokontroller arduino uno merupakan sebuah *chip* yang digunakan untuk mengontrol perangkat elektronik. Mikrokontroller arduino uno merupakan mikrokontroller yang bersifat *open source*. Mikrokontroller arduino uno memiliki beberapa fitur yaitu *central processing unit* (CPU), memory, *digital I/O* pin 14, *analog input* pin 6, clock speed 16 MHz, tegangan kerja 5 volt.

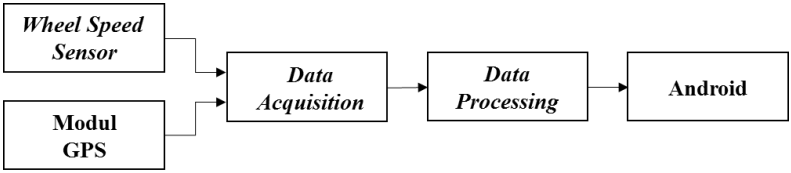


Gambar 2. 5. Mikrokontroller Arduino Uno

Gambar 2.5 adalah *board* arduino uno. Pada *board* terdapat *chip* ATMega sebagai prosesor dari arduino dan terdapat pin yang berfungsi sebagai masukan maupun keluaran. Setiap pin dari mikrokontroller arduino dapat difungsikan sebagai masukan maupun keluaran, namun setiap pin dari arduino tersebut memiliki kemampuan spesifik masing-masing, diantaranya sebagai serial *input*, serial *output*, ADC *input*, digital *input*, dan lain-lain. Mikrokontroller arduino memiliki *software* sendiri yang digunakan untuk menuliskan sebuah program yang dibuat. Program tersebut bersifat *open source* dengan penulisan bahasa menggunakan bahasa pemrograman C, dan *software* pemrograman arduino tersebut dilengkapi dengan adanya *library arduino*, yang digunakan untuk mempermudah pengguna atau *user* dalam

penulisan sebuah program pada lembar kerja mikrokontroler arduino.

Pada penelitian ini, mikrokontroller berfungsi untuk akuisisi data dan pemrosesan data. Akuisisi data yaitu proses pengambilan data yang terukur, misalnya fenomena fisik atau elektrik yang terukur oleh sensor yang kemudian diubah ke dalam nilai digital yang dapat disimulasi ataupun dikomputasi dengan menggunakan komputer. Sedangkan pemrosesan data yaitu merupakan tindakan memanipulasi data agar menjadi bentuk yang lebih informatif atau bermanfaat. Fungsi mikrokontroller pada penelitian ini dapat digambarkan seperti blok diagram di bawah ini.



Gambar 2. 6. Blok Diagram Sistem

Gambar 2.6 merupakan blok diagram penelitian secara umum. Mikrokontroller arduino berfungsi sebagai pengakuisisi data dari sensor dan sebagai pemroses data. Mikrokontroller menghitung frekuensi pulsa keluaran sensor *wheel speed* dan mengambil data *latitude* dan *longitude* dari modul GPS U-blox Neo 6M untuk mendapatkan posisi kendaraan.

Untuk mendpatkan kecepatan kendaraan dalam satuan km/h yang berasal dari keluaran sensor yang berupa frekuensi dalam hertz, digunakan persamaan 2.2 dan 2.3, untuk memudahkan dalam pengkodingan pada arduino maka persamaan 2.2 dan 2.3 disubstitusikan hingga menjadi persamaan di bawah ini.

$$v = 2\,\pi\,f\,r \qquad (2.4)$$

Dimana :

v : kecepatan linier (m/s)
f : frekuensi (Hz)
r : jari-jari lingkaran (m)

Pada persamaan 2.4, v adalah kecepatan linier dengan satuan meter per sekon, f adalah frekuensi yang didapat dari keluaran sensor *wheel speed* dengan satuan hertz, r adalah adalah jari-jari lingkaran atau dalam penelitian ini jari-jari roda dengan satuan meter. Namun dengan persamaan tersebut besaran kecepatan linier masih dalam satuan meter per sekon, sehingga untuk mengubah menjadi satuan kilometer per jam harus dikalikan dengan 3,6. Persamaan 2.4 tersebut kemudian dimasukkan dalam program arduino dengan persamaan seperti di bawah ini.

kecepatan = 2 * 22/7 * frequency * 0.22 * 3.6

Dengan persamaan pada arduino di atas maka data frekuensi masukan dari arduino telah dapat menjadi kecepatan dengan satuan km/h. Nilai 0,22 adalah jari-jari roda yang digunakan dalam penelitian ini yang berukuran 22 cm atau 0,22 m. Sedangkan 3,6 digunakan untuk mengubah nilai besaran kecepatan yang awalnya dalam satuan meter per sekon menjadi kilometer per jam.

2.6 Modul WiFi ESP 8266

Untuk mengirim data yang ada pada mikrokontroller arduino agar dapat diolah pada aplikasi android yang terdapat pada *handphone*, dibutuhkan sebuah modul yang berfungsi sebagai penghubung mikrokontroller ke jaringan internet. Modul tersebut dibutuhkan karena mikrokontroller yang digunakan pada penelitian ini, yaitu arduino uno tidak memiliki fitur untuk terhubung ke jaringan internet, maka digunakanlah modul WiFi ESP 8266 yang dapat menghubungkan mikrokontroller arduino uno dengan jaringan internet agar data yang ada pada mikrokontroller dapat dikirim ke aplikasi yang terdapat pada *handphone* android.

Gambar 2. 7. Modul Wireless ESP 8266

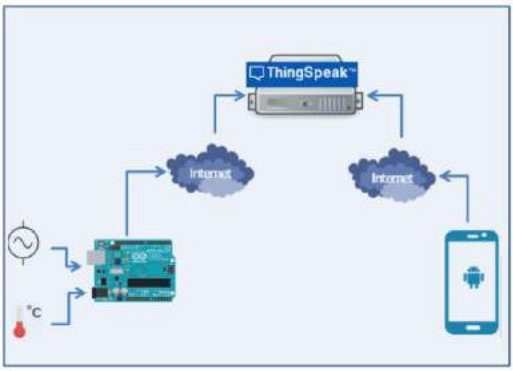
Gambar 2.7 menampilkan *board* modul WiFi ESP 8266. Modul ESP 8266 merupakan modul *low-cost* Wi-Fi dengan dukungan penuh untuk penggunaan TCP/IP. Modul ini diproduksi oleh *Espressif Chinese manufacturer*. Modul *wireless* ESP 8266 yang digunakan pada penelitian ini memiliki *firmware* bawaan pabrik yang mendukung perintah *AT-Command*.

2.7 *Thingspeak*

Pada dasarnya aplikasi android dapat menerima masukan data baik dari masukan langsung dari *handphone*, maupun masukan yang berasal dari luar, misalkan dari sensor yang ada pada sebuah kendaraan seperti pada penelitian ini. Untuk memberi masukan pada aplikasi android yang berasal dari mikrokontroller, maka data perlu dikirim menuju *web server* untuk kemudian *web server*lah yang meneruskan data tersebut menuju aplikasi android. Dalam arti lain *web server* dibutuhkan sebagai penghubung antara mikrokontroller dan aplikasi yang terdapat pada *handphone* berbasis android. Karena dalam penelitian kali ini *web server* hanya dibutuhkan sebagai perantara tanpa adanya proses komputasi pada *web server*, maka digunakanlah *Thingspeak* sebagai *web server*. *Thingspeak* merupakan aplikasi *"Internet of Things"* yang bersifat terbuka (*open source*) dan memiliki antarmuka pemrograman aplikasi yang dapat dimanfaatkan untuk menyimpan dan mengambil data menggunakan HTTP melalui internet atau melalui *Local Area Network*.

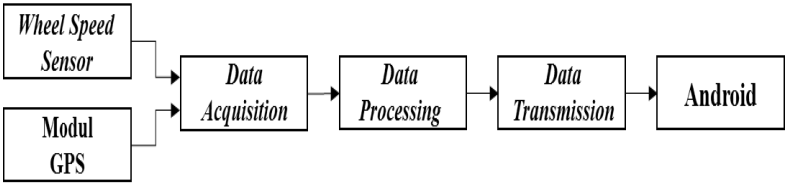
Thingspeak adalah platform IoT yang memungkinkan untuk mengumpulkan, menyimpan, menganalisis, memvisualisasikan dan bertindak atas data dari sensor atau aktuator, seperti *Arduino*, *Raspberry Pi* dan perangkat keras lainnya. Internet of Things (IoT) menyediakan akses ke berbagai perangkat *embedded* dan layanan web. Misalnya, dengan *Thingspeak* kita dapat membuat aplikasi sensor-*logging* ataupun aplikasi pelacakan lokasi. Thingspeak berfungsi sebagai pengumpul data yang mengumpulkan data dari perangkat *node* dan juga memungkinkan data yang akan diambil ke dalam lingkungan perangkat lunak untuk analisis historis data. Unsur utama dari *Thingspeak* adalah saluran (*channel*), yang berisi bidang data, bidang lokasi dan bidang status. Setelah kita membuat

saluran *Thingspeak*, kita dapat menulis data ke saluran, proses dan melihat data dengan kode Matlab dan bereaksi terhadap data dengan *tweet* dan *alert* lainnya. Alur kerja dari *Thingspeak* yaitu pertama membuat saluran (*channel*) dan mengumpulkan data, kemudian menganalisis dan memvisualisasikan data dan kemudian menggunakan salah satu dari beberapa *apps* yang terdapat pada *Thingspeak* bila dibutuhkan, misalnya untuk komputasi menggunakan Matlab.



Gambar 2. 8. Hubungan Jaringan Arduino dan *Thingspeak*

Gambar 2.8 menunjukkan hubungan jaringan arduino dan *thingspeak*. *Thingspeak* dapat menghubungkan arduino dan aplikasi pada *handphone* android. Data yang didapatkan dari sensor dan telah masuk pada arduino dapat dikirim menuju *Thingspeak* melalui koneksi internet dengan bantuan modul WiFi ESP 8266. Setelah diterima oleh *Thingspeak*, kemudian data tersebut diteruskan menuju android melalui koneksi internet pula, sehingga android dapat menerima masukan yang berasal dari pembacaan sensor oleh arduino. Dengan adanya proses pengiriman data yang dibutuhkan pada penelitian ini, dengan menggunakan modul ESP 8266 sebagai penghubung arduino ke jaringan internet dan *Thingspeak* sebagai *web server*, maka blok diagram sistem pada gambar 2.6 menjadi seperti pada gambar di bawah ini.



Gambar 2. 9. Blok Diagram Sistem

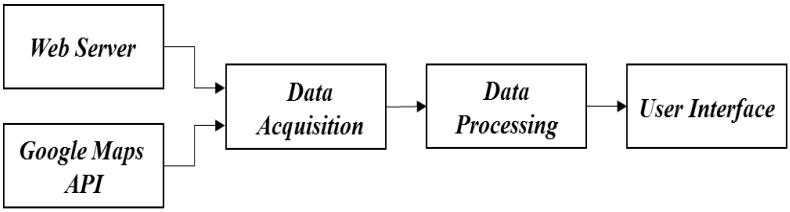
Gambar 2.9 merupakan blok diagram penelitian setelah ditambahkan transmisi data. Mikrokontroller arduino berfungsi sebagai pengakuisisi data dari sensor dan sebagai pemroses data. Mikrokontroller menghitung frekuensi pulsa keluaran sensor wheel speed dan mengambil data latitude dan longitude dari modul GPS U-blox Neo 6M untuk mendapatkan posisi kendaraan. Setelah data diproses oleh arduino, kemudian dilanjutkan dengan proses transmisi data untuk mengirim data ke aplikasi android.

2.8 Android

Android adalah sistem operasi dengan sumber terbuka (*open source*), dan *Google* merilis kodenya di bawah Lisensi Apache. Kode dengan sumber terbuka dan lisensi perizinan pada Android memungkinkan perangkat lunak untuk dimodifikasi secara bebas dan didistribusikan oleh para pembuat perangkat, operator nirkabel, dan pengembang aplikasi. Selain itu, Android memiliki sejumlah besar komunitas pengembang aplikasi (apps) yang memperluas fungsionalitas perangkat, umumnya ditulis dalam versi kustomisasi bahasa pemrograman Java. Pengembangan perangkat lunak Android adalah proses di mana aplikasi baru diciptakan untuk sistem operasi Android (Ardana, 2014). Aplikasi tersebut dikembangkan dalam bahasa pemrograman Java dengan menggunakan *Software Development Kit* (SDK) Android, tetapi perkakas lainnya juga tersedia. Terdapat berbagai macam peruntukan saat membuat aplikasi android, misalnya aplikasi android untuk jual beli, info layanan masyarakat, permainan ataupun hiburan.

Aplikasi android dapat menerima masukan data yang berasal dari *form input* langsung dari *handphone* yang dimasukkan dari *handphone* pengguna. Untuk fungsi tersebut dapat dibuat dari fitur *form input* pada SDK android studio. Selain itu android juga dapat menerima masukan dari *web server* yang artinya android akan menerima data yang dikirimkan dari sebuat *website* melalui komunikasi HTTP. Untuk fungsi tersebut dapat dibuat dari fitur *get data* pada SDK android studio.

Pada penelitian ini apllikasi android akan menerima masukan dari *web server Thingspeak* dan kemudian diintegrasikan dengan *google maps*, oleh karena itu dibutuhkan untuk memasukkan *google maps* pada aplikasi android ini, hal tersebut dapat dilakukan dengan *Google Maps API. Google Maps Application Programming Interface* (API) merupakan sebuah fitur yang disediakan oleh *Google Maps* untuk memudahkan pengguna dalam menggunakan *Google Maps* untuk sebuah situs. Dengan menggunakan *Google Maps API*, anda bisa menyimpan *Google Maps* di sebuah situs baik pada *platform* PC, android maupun IOS. *Google Maps* API dapat digunakan pada aplikasi transportasi online untuk melacak posisi maupun navigasi ataupun untuk menunjukkan lokasi usaha.



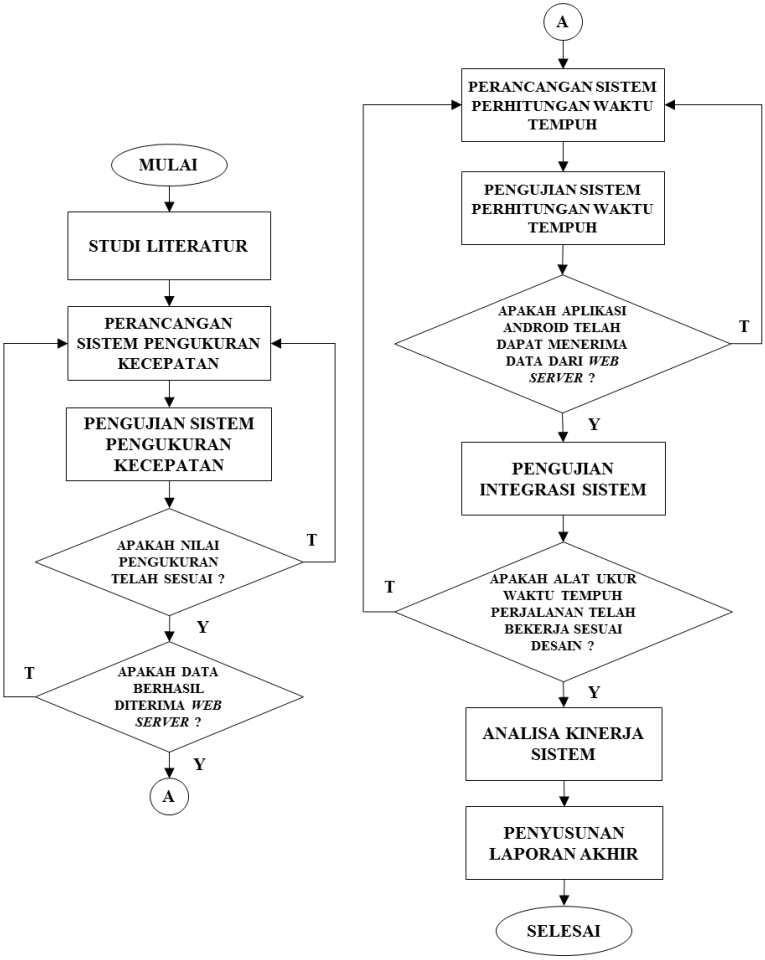
Gambar 2. 10. Blok Diagram Aplikasi Android

Gambar 2.10 merupakan blok diagram aplikasi android yang akan digunakan pada penelitian ini. Aplikasi android akan mengakuisisi data dari *web server* dan *google maps.* Data yang diakuisisi dari *google maps* berupa tampilan peta, jarak dari lokasi

asal ke lokasi tujuan dan rute yang harus dilewati untuk menuju lokasi tujuan. Selain itu aplikasi android juga sebagai pemroses data. Data yang diporses yaitu data kecepatan dan posisi GPS yang didapat dari *web server* dan diintegrasikan dengan jarak dan rute menuju lokasi tujuan untuk kemudian aplikasi android dapat dihitung waktu tempuh yang dibutuhkan untuk mencapai lokasi tujuan. Terakhir yaitu *update user interface* yang berupa tampilan posisi pada peta dan rute yang harus dilewati, informasi kecepatan kendaraan dan waktu tempuh.

BAB III
METODOLOGI PENELITIAN

Bab ini menjabarkan langkah-langkah yang dilakukan dalam perancangan alat ukur waktu tempuh perjalanan berdasarkan kecepatan kendaraan pada *speedometer*. *Flowchart* dari penelitian ini seperti ditunjukkan pada gambar 3.1.



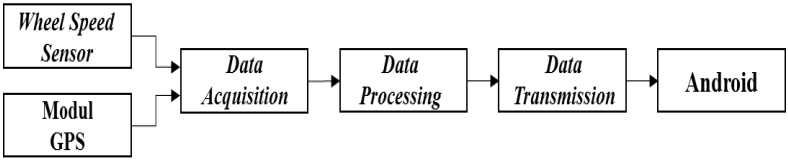
Gambar 3. 1. *Flowchart* Perancangan Sistem

3.1 Studi Literatur

Pada penelitian tugas akhir ini, studi literatur dilakukan untuk memahami perancangan alat ukur waktu tempuh perjalanan berdasarkan kecepatan kendaraan pada *speedometer* dengan pengujiannya. Bahan studi didapatkan dari beberapa jurnal penelitian yang telah dilakukan sebelumnya, maupun buku-buku yang membahas mengenai pengukuran kecepatan, mikrokontroller, android dan *google maps*.

Studi literatur ini bertujuan untuk mengetahui desain rancangan yang dapat diimplementasikan sebagai alat ukur waktu tempuh perjalanan berdasarkan kecepatan kendaraan pada *speedometer*.

3.2 Perancangan Sistem



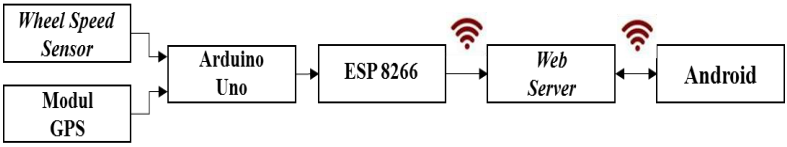
Gambar 3. 2. Blok Diagram Sistem

Gambar 3.2 merupakan blok diagram penelitian secara umum. Mikrokontroller arduino berfungsi sebagai pengakuisisi data dari sensor dan sebagai pemroses data. Mikrokontroller menghitung frekuensi pulsa keluaran sensor *wheel speed* dan mengambil data *latitude* dan *longitude* dari modul GPS U-blox Neo 6M untuk mendapatkan posisi kendaraan. Setelah data diproses oleh arduino dan didapatkan data berupa kecepatan, *latitude* dan *longitude*, kemudian dilanjutkan dengan proses transmisi data untuk mengirim data menuju aplikasi pada *handphone* berbasis android.

Pada bagian perancangan sistem ini bertujuan untuk merancang sistem pengukuran kecepatan kendaraan dari *speedometer* yang dapat terhubung dengan *handphone* berbasis android dan diintegrasikan dengan informasi jarak dan rute yang akan dilewati dari *google maps* untuk diolah menjadi informasi waktu tempuh dan ditampilkan pada sebuah aplikasi berbasis android. Perancangan ini akan menjadi dua bagian, yaitu bagian

pertama pada bagian sistem pengukuran yang meliputi sensor pengukuran kecepatan, sensor GPS dan sistem pengiriman data ke *web server*. Sistem pengukuran ini yang memberikan *input* berupa kecepatan kendaraan dan data posisi GPS serta sistem pengirim data ke *web server.* Bagian kedua yaitu pada bagian sistem perhitungan waktu tempuh yang akan mengolah data kecepatan dan posisi GPS yang telah ada pada *web server* dan diintegrasikan dengan data jarak dan rute pada *google maps* yang kemudian dikalkulasikan untuk menghasilkan waktu tempuh. Selain itu pada bagian ini juga meliputi perancangan desain *Graphical User Interface* (GUI) yang akan menampilkan posisi kendaraan pada peta dan informasi kecepatan dan waktu tempuh pada sebuah aplikasi android.

Agar data yang terbaca pada sensor dapat diolah pada *handphone* berbasis android, maka data harus dikirim ke *web server* sebagai perantara agar dapat selanjutnya dikirim ke aplikasi android yang ada di *handphone*. Sehingga blok diagram pada gambar 3.2 menjadi seperti gambar 3.3 di bawah ini.



Gambar 3. 3. Blok Diagram Sistem

Pada gambar 3.3 merupakan blok diagram sistem dari sistem pengukuran. Input dari sistem ini adalah hasil pembacaan frekuensi putaran roda dari *wheel speed sensor* dan hasil pembacaan koordinat *latitude* dan *longitude* dari modul GPS. Kedua sumber tersebut menjadi masukan bagi arduino uno yang berperan dalam *data acquisition* dan *data processing.* Arduino mengakuisisi data frekuensi keluaran sensor untuk kemudian dihitung frekuensi perdetiknya serta mengakuisisi koordinat *latitude* dan *longitude* hasil pembacaan modul GPS. Kemudian data diolah oleh arduino hingga menjadi data kecepatan dalam satuan km/h dan koordinat *latitude* dan *longitude.* Setelah itu ketiga data tersebut

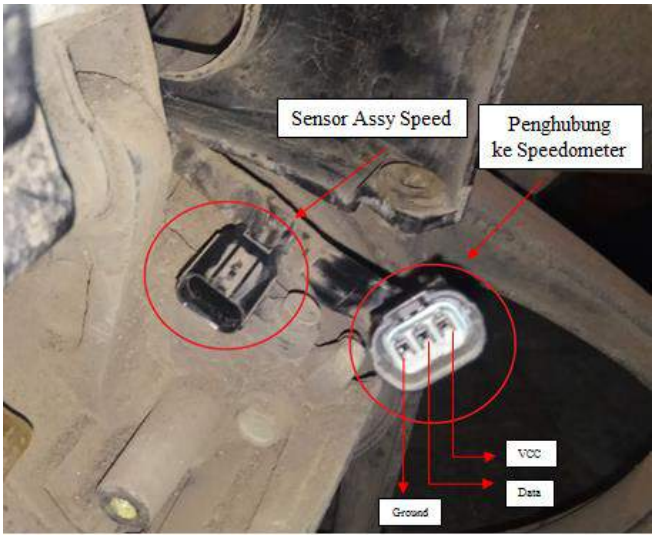
ditransmisikan oleh modul WiFi ESP 8266 melalui koneksi internet untuk dikirim menuju *web server*. Kemudian *web server* akan menunggu permintaan data dari android sebelum mengirimkannya.

Perancangan sistem pengukuran ini bertujuan untuk mendapatkan nilai kecepatan dan posisi kendaraan. Nilai kecepatan didapatkan dari keluaran sensor putaran roda yang telah ada pada sepeda motor honda vario yang digunakan pada penelitian ini. Sensor tersebut merupakan sensor *hall effect* yang memiliki keluaran berupa frekuensi yang telah terkalibrasi sehingga dapat diolah oleh mikrokontroller arduino. Sedangkan posisi kendaraan didapat dari modul GPS U–blox yang diletakkan pada sepeda motor dan dihubungkan dengan mikrokontroller arduino. Kemudian kedua data tersebut dikirim menuju *web server* agar nantinya data tersebut dapat diolah pada aplikasi android. Pengiriman data dari mikrokontroller arduino menuju *server* menggunakan koneksi internet yang terhubung melalui modul WiFi ESP 8266. Sedangkan *web server* sendiri akan menggunakan *server* IoT *Thingspeak*.

Untuk mendapatkan hasil yang optimal maka terlebih dahulu dilakukan studi literatur mengenai GPS beserta modul GPS U-blox dan perhitungan frekuensi sensor pada arduino. Setelah itu menentukan persamaan untuk menghitung kecepatan kendaraan berdasarkan frekuensi keluaran dari sensor tersebut.

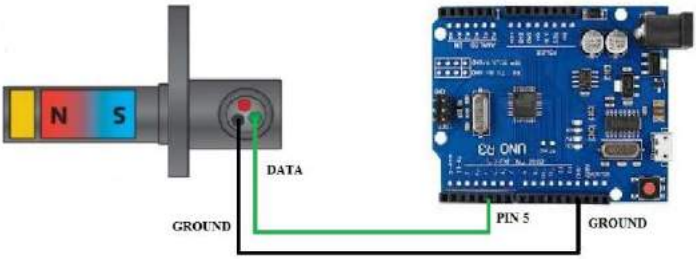
3.2.1 Perancangan Sistem Pengukuran Kecepatan

Perancangan sensor kecepatan ini diawali dengan mempelajari prinsip kerja sensor kecepatan yang telah ada pada sepeda motor yang digunakan. Kemudian dilakukan penyambungan kabel keluaran dari sensor yang terhubung ke *speedometer* untuk disambungkan pada mikrokontroller arduino, sehingga mikrokontroller arduino mendapatkan masukan dari keluaran sensor tersebut.



Gambar 3. 4. Sensor *Wheel Speed* pada Sepeda Motor Vario 125

Pada gambar 3.4 dapat dilihat pada sensor terdapat 3 kabel yaitu vcc, data dan *ground*. Dalam penelitian ini hanya membutuhkan keluaran dari data dan *ground* dari sensor untuk dihitung oleh arduino. Keluaran data pada sensor merupakan sinyal dengan frekuensi yang berbeda berdasarkan kecepatan putaran roda. Kemudian sensor tersebut dihubungkan dengan mikrokontroller arduino. Berikut ini merupakan integrasi keluran sensor dengan mikrokontroller arduino uno.



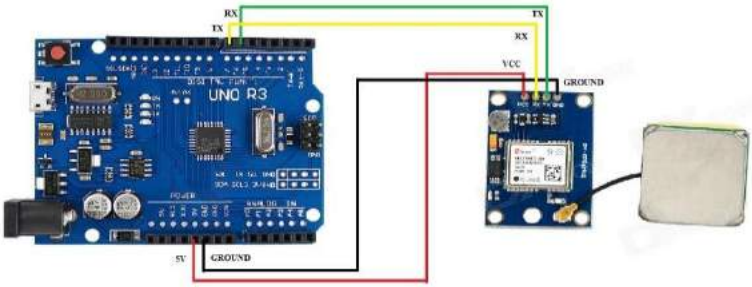
Gambar 3. 5. Perancangan Sensor Kecepatan

Pada gambar 3.5 menunjukkan bahwa terdapat 2 pin pada mikrokontroller yang mendapat masukan dari sensor. Keluaran data dari sensor disambungkan pada pin 5 arduino sebagai penghitung frekuensi keluaran sensor. Sedangkan *ground* (GND) keluaran sensor disambungkan pada *ground* (GND) arduino.

Perancangan sensor kecepatan dimulai dengan mencari letak sensor dan membuat sambungan kabel sebagai *input* arduino. Kemudian dilakukan pengecekan *output* tegangan sensor menggunakan multimeter, ini dilakukan untuk memastikan tegangan yang diambil telah tepat. Setelah itu dilakukan penyambungan kabel data keluaran sensor ke pin 5 arduino sebagai penghitung frekuensi dan penyambungan kabel ground (GND) sensor ke ground (GND) arduino. Setelah penyambungan telah dilakukan, dilanjutkan dengan pengambilan data frekuensi keluaran dengan serial monitor arduino. Setelah data *output* frekuensi didapatkan, selanjutnya dilakukan proses pemrograman pada arduino.

3.2.2 Perancangan Sensor GPS

Perancangan GPS ini bertujuan untuk menerima data dari kendaraan yang berupa koordinat posisi kendaraan tersebut. Pada kendaraan akan dipasang modul *global positioning system* (GPS) yang dapat dikontrol oleh mikrokontroller arduino. Modul GPS yang digunakan adalah Ublox Neo-6M. Modul GPS ini berfungsi mengirimkan letak koordinat. Koordinat yang dikirimkan berupa koordinat bujur (*longitude*) dan koordinat lintang *(latitude*). Karakteristik GPS dapat beroperasi pada tegangan 3-5V, akurasi pada posisi *horizontal* 2.5 m, dan ketinggian maksimal 50.000 m diatas permukaan laut. Berikut ini merupakan integrasi modul GPS yang digunakan dengan mikrokontroller arduino uno.



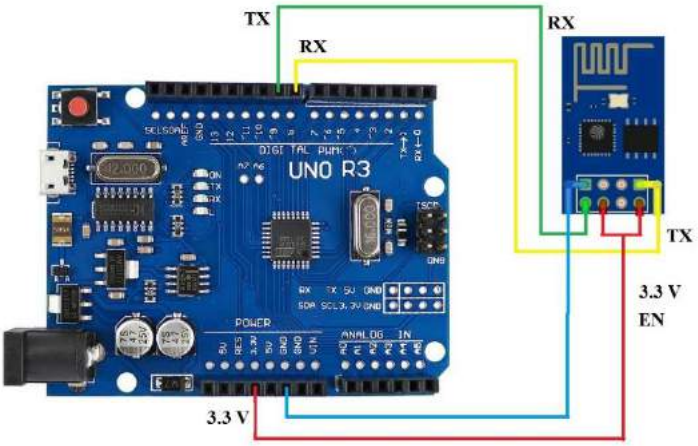
Gambar 3. 6. Perancangan GPS

Gambar 3.6 dapat dilihat bahwa terdapat beberapa pin pada *global positioning system* (GPS) disambungkan pada mikrokontroller. Terdapat Vin yang disambungkan terhadap tegangan 5V tegangan pada mikrokontroller, *ground* (GND) disambungkan pada *ground* (GND) mikrokontroller, *transmitter* (TX) GPS disambungkan pada pin 6 sebagai *receiver* (RX) mikrokontroller, dan *receiver* (RX) GPS disambungkan pada pin 7 sebagai *transmitter* (TX) mikrokontroller. Perancangan dilanjutkan dengan membuat program mikrokontroller. Setelah integrasi berhasil, kemudian dilanjutkan dengan pengambilan data, apabila data dapat diterima maka akan dilanjutkan menuju serial monitor. Apabila tidak diterima maka akan dilakukan perbaikan pada integrasi dengan program mikrokontroller.

3.2.3 Perancangan Sistem Pengiriman Data

Perancangan sistem pengiriman data ini bertujuan untuk dapat mengirimkan data dari sensor menuju *web server* yang nantinya menghubungkan mikrokontroller dan android. *Web server* yang digunakan pada perancangan ini yaitu IoT berbasis *Thingspeak*. Perancangan sistem pengiriman data ini dilakukan dengan dua tahap, yaitu yang pertama menghubungkan arduino menuju koneksi internet dengan menggunakan modul WiFi ESP8266. Kemudian tahap kedua yaitu menghubungkan mikrokontroller dengan *Channel* yang telah dibuat pada *Thingspeak* melalui koneksi internet.

Pada perancangan sistem pengiriman data ini, pin pada modul ESP 8266 dihubungkan dengan pin yang ada pada mikrokontroller arduino agar modul ini dapat terkoneksi dengan benar dan dapat menghubungkan arduino menuju koneksi internet. Kemudian dilanjutkan dengan men-*setting* modul ini agar tersambung dengan *router* yang telah disediakan sebagai penyedia koneksi internet. Pengaturan dilakukan melalui serial monitor IDE arduino menggunakan *AT Command* untuk menyambungkan dengan *router* dan memasukkan *password* dari *router* tersebut.

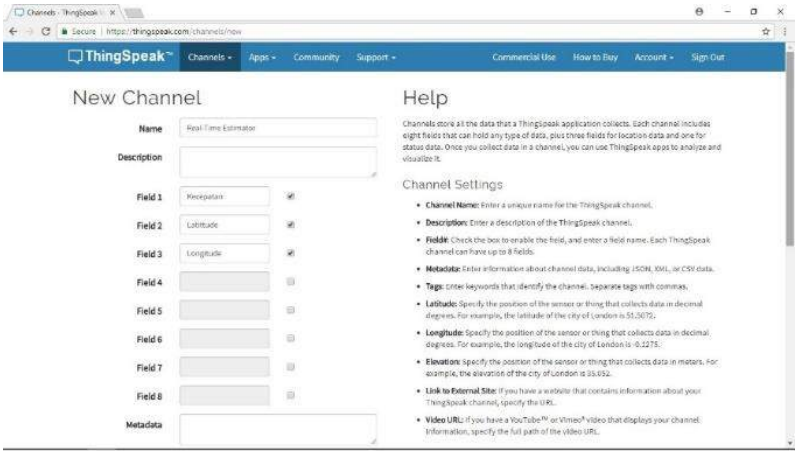


Gambar 3. 7. Konfigurasi Pin ESP 8266 pada Arduino

Pada gambar 3.7 dapat dilihat bahwa terdapat 4 pin pada mikrokontroller yang terkoneksi dengan modul ESP 8266. Pin 3V3 pada mikrokontroller disambungkan dengan pin 3V3 dan EN pada modul ESP 8266. Pin ground (GND) modul ESP 8266 disambungkan dengan ground (GND) arduino. Pin RX modul ESP 8266 disambungkan pada pin 9 arduino sebagai TX, sedangkan pin TX modul ESP 8266 disambungkan pada pin 8 arduino sebagai RX.

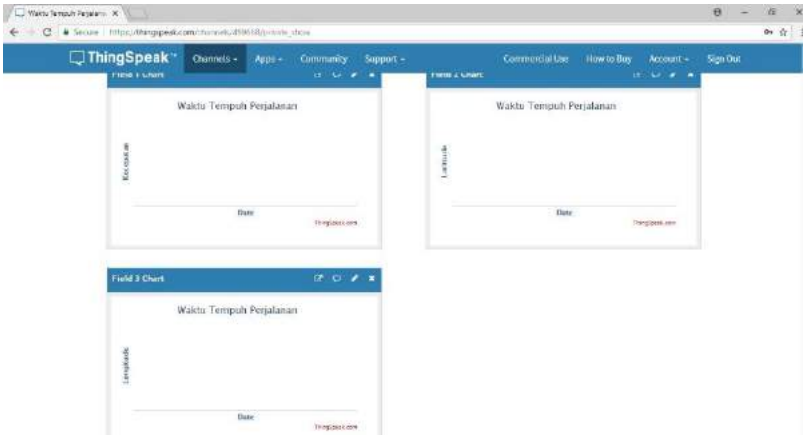
Setelah arduino terkoneksi dengan internet, dilanjutkan dengan menghubungkan arduino dengan *web server*. Perancangan *web server* ini bertujuan untuk menerima data dari arduino, yaitu

data keluaran sensor kecepatan dan sensor GPS. *Web server* ini dibutuhkan sebagai penghubung antara mikrokontroller dan android. Data yang telah diterima oleh *web server* inilah yang nantinya akan dikirim menuju android untuk diolah menjadi informasi waktu tempuh. *Web server* ini dibuat pada *platform* IoT *Thingspeak.*



Gambar 3. 8. Proses Pembuatan *Channel* pada *Thingspeak*

Pada gambar 3.8 dapat dilihat perancangan *web server* pada *Thingspeak* dimulai dengan membuat *channel* baru sesuai dengan kebutuhan. Setelah melakukan konfigurasi akun *thingspeak* kemudian dibuat *channel* sebanyak yang dibutuhkan. Pada perancangan ini dibuat 3 *field* yaitu *field* pertama akan berisi data kecepatan, *field* kedua berisi data *latittude* dan *field* ketiga berisi data *longitude*. Ketiga *filed* tersebut nantinya akan berisi data yang dikirim dari mikrokontroller arduino. Data yang ditampilkan pada *field* berupa grafik, namun *history* data dapat diunduh dalam bentuk file excel. Setelah selesai mengkonfigurasi, maka tampilan *field* nya dapat dilihat pada gambar 3.9.



Gambar 3. 9. Tampilan *Field* pada *Thingspeak*

Pada gambar 3.9 telah dapat dilihat 3 *field* yang telah berhasil dibuat. 3 *field* tersebut yang nantinya akan berisi data kecepatan, *latitude*, dan *longitude.*

3.3 Pengujian Komponen Sistem Pengukuran

Pengujian komponen sistem pengukuran ini akan dibagi menjadi tiga tahap, yaitu tahap pengujian sensor kecepatan, pengujian sensor *global positioning system* (GPS) dan pengujian sistem pengiriman data. Pengujian ini dilakukan dalam tiga tahap dengan tujuan untuk mengetahui apakah tiap fungsi atau komponen penyusun dalam penelitian ini telah bekerja sesuai perancangan.

3.3.1 Pengujian Sistem Pengukuran Kecepatan

Pengujian sensor kecepatan bertujuan untuk mengetahui akurasi sensor, sensitifitas sensor dan keakuratan pengukuran sensor ketika berjalan pada lintasan sebenarnya. Pada pengujian akurasi sensor dan sensitifitas sensor, pengujian dilakukan dengan mengatur kecepatan kendaraan pada kecepatan yang ingin diukur dengan memperhatikan kecepatan yang terukur pada *speedometer*, kemudian dilihat nilai yang terukur pada *serial monitor* arduino. Pada pengujian sensor kecepatan pada lintasan sebenarnya, akan

didapatkan nilai frekuensi keluaran sensor kecepatan dan pembacaan kecepatan dalam satuan km/h. Pengujian dilakukan dengan lintasan lurus sejauh 500 m dengan variasi kecepatan 20 km/h, 40 km/h dan 60 km/h dan dicatat waktu yang dibutuhkan untuk berjalan dari titik awal ke titik akhir dan kemudian dibandingkan dengan perhitungan secara teori. Pengujian dilakukan lima kali tiap variasi kecepatan. Berikut merupakan tahapan pengujian sensor kecepatan yang digunakan pada penelitian:

1. Menyalakan mesin sepeda motor.
2. Memastikan *hardware* telah terpasang dan bekerja dengan baik.
3. Berjalan dari titik awal ke titik akhir sejauh 500 meter dengan variasi kecepatan 20, 40 dan 60 km/jam.
4. Mencatat waktu yang dibutuhkan untuk berjalan dari titik awal ke titik akhir.
5. Membandingkan dengan perhitungan teori.

3.3.2 Pengujian Sistem GPS

Pengujian sensor *global positioning system* (GPS) bertujuan untuk mengetahui apakah pembacaan posisi GPS telah sesuai. Pada pengujian sensor *global positioning system* (GPS) akan mendapatkan keluaran *longitude* dan *latitude*. Pengujian akan dilakukan dengan berjalan pada lintasan lurus sejauh 500 meter, ini bertujuan untuk mendapatkan nilai *longitude* dan *latitute* dari tiap perpindahan pada lintasan. Berikut merupakan tahapan pengujian sensor GPS yang digunakan pada penelitian:

1. Menyalakan mesin sepeda motor.
2. Memastikan *hardware* telah terpasang dan bekerja dengan baik.
3. Memastikan bahwa GPS sudah mendapat sinyal dari satelit, dalam bahasa GPS adalah locking.
4. Berjalan dari titik awal ke titik akhir sejauh 500 meter.
5. Mengambil data rekaman posisi GPS pada *web server*.
6. Memvalidasi dengan *software google maps*.

3.3.3 Pengujian Sistem Pengiriman Data

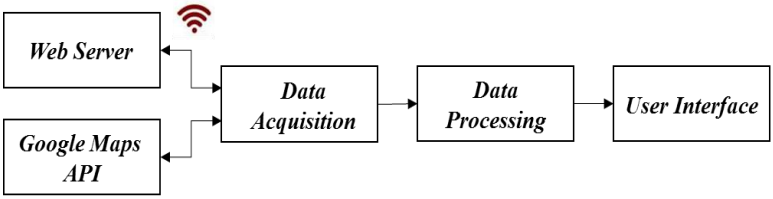
Pengujian sistem pengiriman data ini bertujuan untuk mengetahui tingkat keberhasilan data diterima pada *Thingspeak*. Pada pengujian pengiriman data ini *Thingspeak* akan mendapatkan masukan data kecepatan, data *latitude* dan *longitude* yang dikirim dari mikrokontroller. Pengujian ini juga bertujuan untuk mengetahui berapa presentase keberhasilan pengiriman data yang dilakukan oleh mikrokontroller, sehingga apabila terdapat *loss* data pada pengiriman dapat diatasi.

Berikut merupakan tahapan pengujian sensor kecepatan yang digunakan pada penelitian:

1. Memastikan koneksi mikrokontroller arduino, modul GPS, modul ESP 8266 telah benar.
2. Menyalakan mesin sepeda motor.
3. Berjalan pada lintasan lurus sejauh 500 meter.
4. Mengamati perubahan yang terjadi pada *field Thingspeak*.
5. Melihat hasil perekaman data yang masuk pada *Thingspeak* melalui file excel yang tersedia.
6. Mengetahui presentase keberhasilan pengiriman data dari arduino.

3.4 Perancangan Sistem Perhitungan Waktu Tempuh

Tahap akhir dari penelitian ini berupa perancangan sistem perhitungan waktu tempuh melalui integrasi *speedometer* dan *google maps*. Keluaran dari sistem ini adalah sebuah aplikasi android yang berguna untuk menghitung waktu tempuh kendaraan. Pada aplikasi android tersebut terdapat tampilan *google maps* yang menunjukkan posisi kendaraan, rute yang dilalui untuk mencapai lokasi tujuan dan kecepatan kendaraan secara *real-time* berdasarkan *speedometer*. Aplikasi android akan mengambil data dari *web server*, data tersebut adalah data kecepatan dan posisi GPS yang terukur dari sensor yang terdapat pada kendaraan yang sebelumnya telah dikirimkan dari mikrokontroller arduino ke *web server*.



Gambar 3. 10. Blok Diagram Aplikasi Android

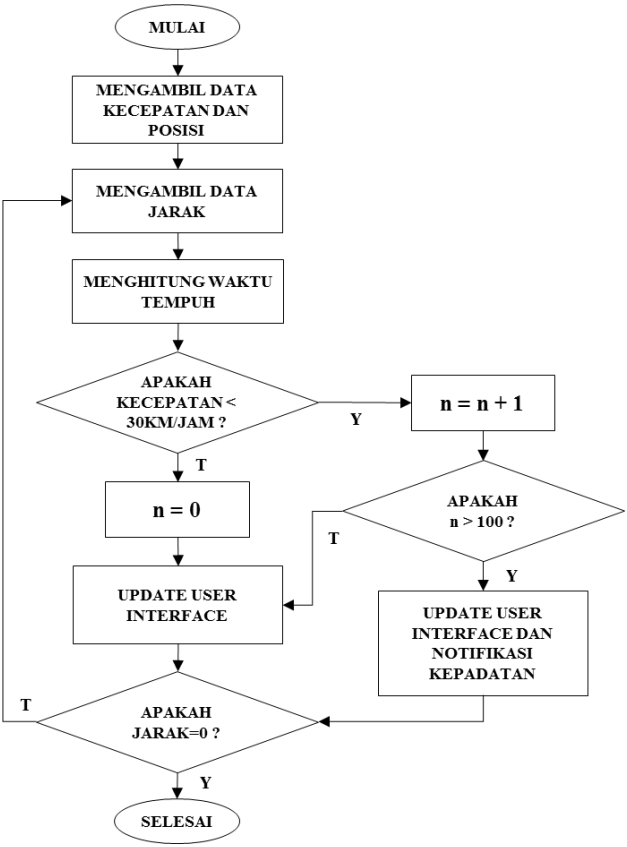
Gambar 3.10 adalah blok diagram aplikasi android. Aplikasi android berperan mengakuisisi data dari *web server* dan melakukan data prosesing untuk menghitung waktu tempuh perjalanan dari data kecepatan yang didapat dari *web server* dan data jarak dari *google maps* API. Setelah waktu tempuh didapatkan, kemudian aplikasi android melakukan *update user interface* yang menampilkan posisi kendaraan pada peta, rute yang dilalui untuk menuju lokasi tujuan dan informasi kecepatan serta waktu tempuh perjalanan.

Perancangan aplikasi android ini bertujuan untuk merealisasikan aplikasi android sebagai alat ukur waktu tempuh kendaraan. Pada penelitian ini perancangan aplikasi android dilakukan menggunakan *software* android studio. Aplikasi yang dirancang ini pada android studio ini diberi judul *smartmover*. Pada perancangan aplikasi android ini dapat dibagi menjadi tiga tahap, yaitu yang pertama menentukan algoritma sebagai perhitungan waktu tempuh. Setelah tahap pertama selesai, dilanjutkan dengan tahap kedua yaitu memberikan masukan data kecepatan dan posisi kendaraan (*latitude* dan *longitude*) dari *web server* menuju aplikasi android yang akan digunakan untuk menghitung waktu tempuh serta memasukkan *google maps* pada aplikasi yang akan dibuat ini dengan *google maps* API. Kemudian dilanjutkan dengan membuat tampilan *Graphical User Interface* (GUI) yang salah satunya sebagai input tujuan dimana *user* akan memasukkan lokasi tujuan.

3.4.1 Perancangan Algoritma Waktu Tempuh

Pada perancangan algoritma waktu tempuh ini dilakukan pembuatan program untuk perhitungan waktu tempuh dan notifikasi kepadatan. Ketika kendaraan melaju stabil, maka nilai

kecepatan akan stabil, jarak terhadap lokasi tujuan akan berkurang secara konstan dan waktu tempuh akan berkurang secara konstan. Sedangkan ketika kendaraan berubah kecepatan baik bertambah maupun berkurang pengaruh dari kondisi lalu lintas maupun keinginan pengemudi, maka nilai kecepatan akan tertampil berubah dan waktu tempuh akan berubah pula mengikuti kecepatan kendaraan sehingga didapat perubahan yang *real-time* berdasarkan perubahan kecepatan kendaraan. Untuk diagram alir algoritmanya dapat dilihat pada gambar 3.11.



Gambar 3. 11. Diagram Alir Algoritma Waktu Tempuh

Gambar 3.11 adalah diagram alir dari algoritma waktu tempuh. Jarak awal akan diambil dari *google maps*, berdasarkan posisi awal kendaraan dan lokasi tujuan. Posisi awal kendaraan dan kecepatan diambil dari data sensor yang telah dikirim ke database *web server*. Informasi waktu tempuh ini akan diperbarui secara *real-time* berdasarkan kecepatan kendaran per tiga detik. Waktu tempuh dihitung menggunakan persamaan 2.1. Ketika kecepatan kendaraan di bawah 30 km/jam, akan ada perhitungan naik untuk menampilkan notifikasi kepadatan. Ketika perhitungan naik telah mencapai 100, dengan kata lain kendaraan berjalan pada kecepatan kurang dari 30 km/jam selama 5 menit, maka notifikasi kepadatan akan muncul. Notifikasi informasi kendaraan sedang berada pada kepadatan lalu lintas akan tampil ketika kendaraan melaju kurang dari 30 km/jam selama lebih dari 5 menit dikarenakan menggunakan asumsi kecepatan rata-rata adalah 40 km/jam. Sehingga apabila kendaraan berjalan dibawah 30 km/jam selama 5 menit dianggap kondisi di luar ideal sehingga dianggap kendaraan sedang berada pada kondisi lalu lintas yang padat. Pada saat kendaraan berhenti di persimpangan jalan dengan *traffic light* maupun sebagainya, maka nilai waktu tempuh akan tertampil hasil estimasi dari *google maps*, sehingga nilai waktu tempuhnya tidak akan menjadi tak berhingga dikarenakan kendaraan sedang berhenti atau kecepatannya adalah 0.

3.4.2 Integrasi Sistem Pengukuran dengan *Google Maps*

Perancang selanjutnya yaitu melakukan integrasi data sensor yang telah dikirim ke *web server* dengan *google maps* untuk menghitung dan menampilkan informasi waktu tempuh. Diawali dengan memasukkan *google maps* pada aplikasi android yang dirancang menggunakan *google maps* API. Perancangan aplikasi android ini menggunakan *software* Android Studio. Untuk menampilkan peta *google maps* dilakukan *input google maps* API pada saat proses koding. Untuk dapat menggunakan *google maps* API sebelumnya dibutuhkan pengaktifan kredensial *google maps* API yang dapat dilakukan menggunakan akun *google*.

Setelah *google maps* berhasil dimasukkan pada aplikasi android yang dirancang, dilanjutkan dengan menyambungkan aplikasi android ini dengan database *web server* agar data dari sensor dapat menjadi masukan untuk perhitungan waktu tempuh yang akan dilakukan pada aplikasi android ini. Hal ini dapat dilakukan dengan menggunakan *library retrofit* untuk berkomunikasi dengan *web server* dengan memasukkan kunci API *server*. Ketika *call* berhasil dan mendapat respon dari *server* untuk datanya, kemudian dilakukan penyimpanan ke database lokal, kalkulasi waktu tempuh, update *user interface* dan kalkulasi kemacetan untuk menghasilkan notifikasi.

Untuk perhitungan waktu tempuh, nilai kecepatan yang didapatkan dari *web server* digunakan untuk mencari nilai untuk kilometer yang didapatkan selama 3 detik dengan kecepatan tersebut. Untuk menghitung informasi kepadatan akan digunakan kondisi bahwa kedapatan adalah kondisi ketika kendaraan tidak dapat melaju lebih dari 30 km/h selama lebih dari 5 menit, hal itu dikarenakan asumsi kecepatan rata-rata kendaraan di jalan raya adalah 40 km/jam. Untuk cuplikan pemrograman waktu tempuh pada aplikasi android dapat dilihat pada gambar 3.12.

Algoritma Kalkulasi :

```
private void calculateEstimation() {
    double tempDistance, tempEstimTime;
    String tempEstimTimeString;
    tempDistance = SPEED / 3600;
    DISTANCE = DISTANCE - tempDistance;
    tvDistance.setText(new DecimalFormat("##.#").format(DISTANCE) + " km");
    tempEstimTime = DISTANCE / SPEED;
    tempEstimTimeString = Constanta.timeConversion((int) tempEstimTime);
    if (tempEstimTimeString.contains("0 h 0 m")) {
       tempEstimTimeString.replace("0 h 0 m", "");
    } else if (tempEstimTimeString.contains("0 h")) {
       tempEstimTimeString.replace("0 h", "");
    }
    tvEstimTime.setText(tempEstimTimeString);
```

Gambar 3. 12. Cuplikan Program Perhitungan Waktu

Gambar 3.12 adalah cuplikan program perhitungan waktu tempuh perjalanan. Waktu tempuh perjalanan akan dihitung

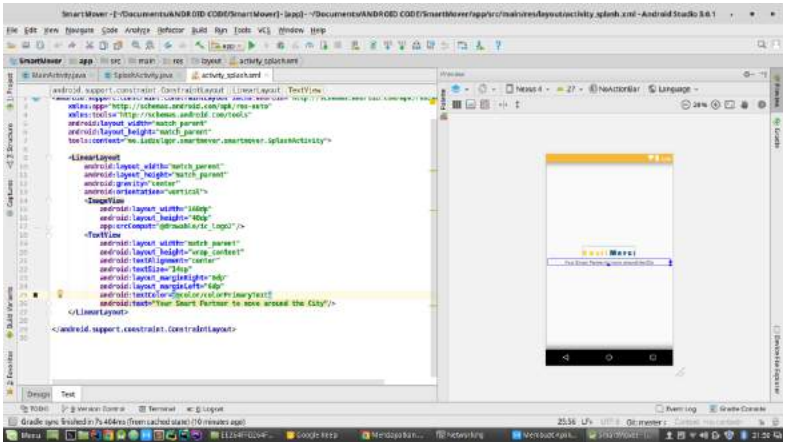
menggunakan persamaan 2.1. Persamaan tersebut dimasukkan pada program aplikasi android sehingga menjadi :

tempEstimTime = DISTANCE / SPEED

dimana *tempEstimTime* adalah waktu tempuh perjalanan, *distance* adalah jarak yang didapatkan dari *google maps* berdasarkan data posisi GPS kendaraan yang didapat dari modul GPS dan *speed* adalah kecepatan kendaraan yang terukur oleh sensor *wheel speed*.

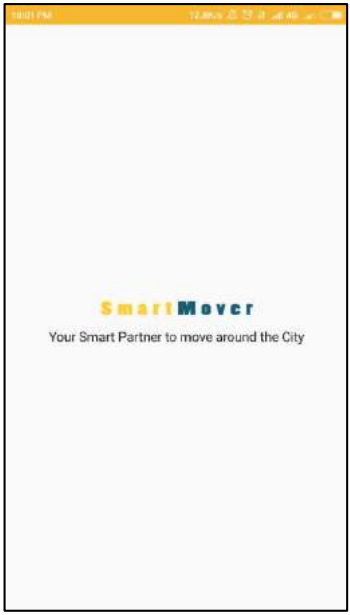
3.4.3 Desain *Graphical User Interface* (GUI)

Pertama dimulai dengan merancang tampilan pembuka aplikasi (*splash screen page*). Tampilan ini adalah tampilan ketika aplikasi pertama kali dibuka. Sebelumnya disiapkan logo atau tulisan yang akan ditampilkan pada tampilan pembuka ini. Pada software android studio tampilan pembuka ini dapat dibuat pada menu *create splash activty*. Tampilan pembuka ini dibuat dengan *delay* selama 3.5 detik sebelum halaman utama terbuka. Tampilan pada android studio pada saat proses perancangan seperti pada gambar 3.13 bawah ini.



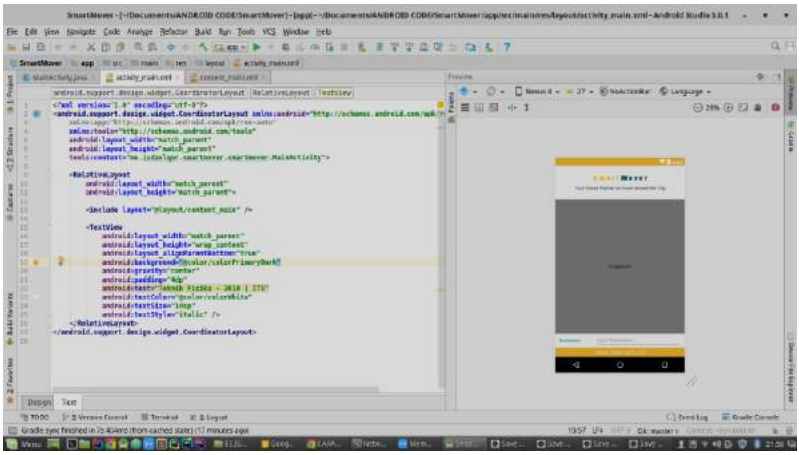
Gambar 3. 13. Perancangan Tampilan Pembuka

Pada gambar 3.13 dapat dilihat cuplikan koding dan *preview* dari tampilan *splash screen* pada saat proses perancangan pada android studio. Setelah *splash screen page* selesai dibuat kemudian dicoba dijalankan pada handphone berbasis android dan hasil tampilannya seperti pada gambar 3.14.



Gambar 3. 14. *Running* Tampilan Halaman Pembuka

Kemudian dilanjutkan dengan perancangan *main page* yang merupakan halaman dimana *user* dapat memberikan *input* berupa lokasi yang dituju. Pada software android studio tampilan halaman ini dapat dibuat pada menu *create main activty*. Dalam merancang halaman ini dimasukkan program pada android studio yang berfungsi untuk melakukan input tujuan, pengaktifan lokasi yang berbasis *google maps* dan *auto complete text* pada saat *user* mengetikkan lokasi tujuan. Tampilan pada android studio pada saat proses perancangan seperti pada gambar 3.15 bawah ini.

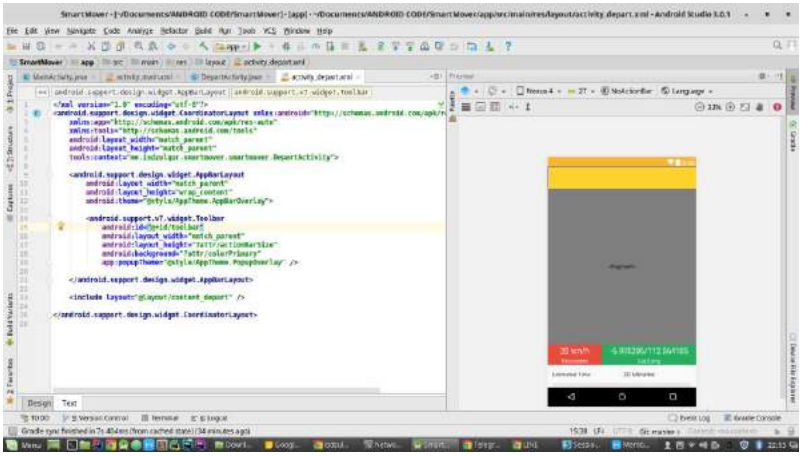


Gambar 3. 15. Perancangan Halaman *Input* Lokasi Tujuan

Pada gambar 3.15 dapat dilihat cuplikan koding dan *preview* dari tampilan *main page* pada saat proses perancangan pada android studio. Setelah *main page* selesai dibuat kemudian dicoba dijalankan pada *handphone* berbasis android dan hasil tampilannya seperti pada gambar 3.16 dibawah ini.

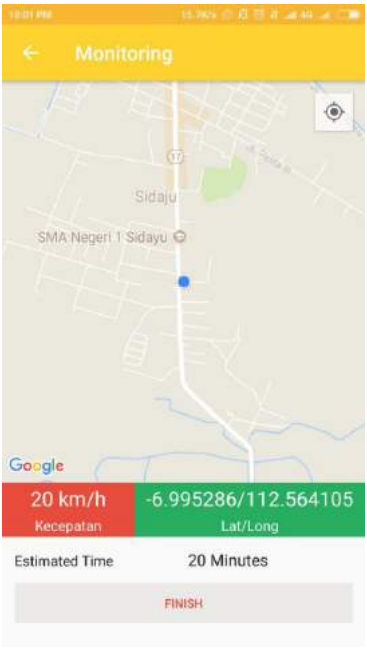
Gambar 3. 16. *Running* Tampilan Halaman Utama

Setelah itu dilanjutkan dengan perancangan *depart page* yang merupakan halaman dimana akan ditampilkan fungsi utama dari aplikasi ini yaitu alat ukur waktu tempuh. Pada halaman ini akan tertampil *google maps* dan posisi kendaraan, kecepatan kendaraan, informasi waktu tempuh dan jarak sisa menuju tempat yang dituju. Tampilan pada android studio pada saat proses perancangan seperti pada gambar 3.17.



Gambar 3. 17. Perancangan *Depart Page*

Pada gambar 3.17 dapat dilihat cuplikan koding dan *preview* dari tampilan *depart page* pada saat proses perancangan pada android studio. *Depart page* inilah yang nantinya akan menjadi keluaran dari penelitian ini. Dalam tampilan *depart page* inilah nantinya dapat dilihat posisi kendaraan pada peta, jalur yang akan dilalui untuk menuju lokasi tujuan, kecepatan kendaraan dan waktu tempuh kendaraan. Setelah *depart page* selesai dibuat kemudian dicoba dijalankan pada handphone berbasis android dan hasil tampilannya seperti pada gambar 3.18 dibawah ini.



Gambar 3. 18. *Running* Tampilan *Depart Page*

3.5 Pengujian Sistem Perhitungan Waktu Tempuh

Pengujian sistem perhitungan waktu tempuh ini akan dibagi menjadi dua tahap, tahap pertama yaitu pengujian masukan aplikasi android dan tahap kedua yaitu pengujian integrasi sistem. Pengujian masukan aplikasi android bertujuan untuk memastikan bahwa aplikasi android telah berhasil menerima masukan data dari *web server*. Kemudian pengujian integrasi sistem bertujuan untuk mengetahui apakah sistem telah dapat mengintegrasikan data yang didapatkan dari *web server* dan informasi rute dan jarak yang didapatkan dari *google maps* sehingga dapat menjadi informasi waktu tempuh dan ditampilkan pada aplikasi android berupa waktu tempuh dan penunjukan lokasi pada peta.

3.5.1 Pengujian Masukan Aplikasi Android

Pengujian masukan aplikasi android bertujuan untuk memastikan bahwa aplikasi android telah berhasil mendapatkan masukan data yang dikirim dari *web server*. Pada pengujian ini mikrokontroller akan mengirimkan data berupa angka 1 hingga 100 tiap 3 detik sekali ke *web server*, kemudian data tersebut akan dikirimkan ke aplikasi android. Berikut merupakan tahapan pengujian masukan aplikasi android yang digunakan pada penelitian:

1. Membuat program pengiriman data pada mikrokontroller ke *web server* berupa angka 1 hingga 100 yang akan dikirimkan tiap 3 detik.
2. Memastikan koneksi mikrokontroller ke *web server* telah benar.
3. Memastikan program pengambilan data dari *web server* pada aplikasi android telah benar.
4. Memulai pengiriman dari mikrokontroller arduino.
5. Mengamati data yang masuk pada aplikasi android.
6. Menganalisa hasil masukan pada *history* data di aplikasi android.

3.5.2 Pengujian Integrasi Sistem

Pengujian integrasi sistem ini dilakukan untuk mengetahui kinerja dari sistem ini pada berbagai kondisi lalu lintas. Pengujian dilakukan pada tiga rute berbeda dengan masing-masing dalam dua waktu yang berbeda. Hal ini dilakukan agar mengetahui kinerja sistem pada rute yang sama namun kondisi lalu lintas yang relatif berbeda. Kemudian dilanjutkan dengan satu rute terakhir yang memiliki kepadatan lalu lintas, rute terakhir dilakukan dua kali pula dengan pengujian pertama kendaraan berjalan dengan kecepatan diatas rata-rata kecepatan kendaraan lainnya dan yang terakhir dengan kecepatan seperti kendaraan disekitarnya. Hal ini ditujukan agar mengetahui kinerja sistem pada kondisi rute yang sama namun dengan kecepatan tempuh yang berbeda. Setelah pengujian dilakukan, kemudian akan dianalisa kinerja sistem berdasarkan data yang didapat saat pengujian. Sehingga diketahui apakah

sistem yang dirancang telah bekerja sesuai yang diharapkan dan untuk mengetahui faktor yang memperngaruhi keberhasilan kinerja dari sistem yang dirancang ini. Berikut merupakan tahapan pengujian integrasi sistem yang digunakan pada penelitian:

1. Menyalakan mesin sepeda motor.
2. Memasukkan lokasi tujuan pada aplikasi android.
3. Berjalan menuju lokasi tujuan.
4. Mengambil rekaman data perjalanan pada *web server*.
5. Menganalisa rekaman perjalanan.

Halaman ini memang dikosongkan.

BAB IV
PENGUJIAN DAN ANALISA

Bab ini membahas mengenai hasil pengujian dan analisa data dari perancangan alat ukur waktu tempuh perjalanan berdasarkan kecepatan kendaraan pada *speedometer*. Data yang didapatkan meliputi kecepatan kendaraan dan posisi GPS, kemudian data yang didapat dikirim menuju *server* untuk kemudian diambil dan diolah pada aplikasi android untuk mendapatkan informasi waktu tempuh kendaraan. Pada tahap akhir dilakukan pengujian integrasi sistem untuk mengetahui kinerja sistem pengukuran waktu tempuh dalam menghitung waktu tempuh kendaraan dan posisi yang tertampil pada *maps* dan kemudian dianalisa kinerjanya.

4.1 Pengujian Komponen Sistem

Pengujian komponen sistem merupakan sebuah tahapan pengujian yang dilakukan untuk mengetahui apakah kinerja setiap komponen penyusun sistem telah sesuai dengan perancangan yang diharapkan. Pengujian ini terdiri dari tiga bagian, yaitu yang pertama pengujian pengukuran kecepatan yang bertujuan untuk mengetahui akurasi sensor, sensitifitas sensor dan pengujian pengukuran sensor pada lintasan sebenarnya. Kemudian pengujian kedua yaitu pengujian sistem GPS dan pengujian ketiga adalah pengujian pengiriman data ke *web server*.

4.1.1 Pengujian Sistem Pengukuran Kecepatan

Pada pengujian pertama sensor kecepatan, bertujuan untuk mengetahui akurasi sensor. Pengujian akurasi sensor dilakukan dengan mengatur kecepatan kendaraan pada kecepatan yang ingin diukur, kemudian diperhatikan kecepatan yang terukur pada *speedometer*, kemudian nilai yang terukur pada *serial monitor* arduino dicatat. Hasil dari pengukuran yang terukur pada *serial monitor* kemudian dibandingkan dengan yang ditampilkan pada *speedometer* untuk mengetahui akurasi dari sensor kecepatan. Hasil pengujiannya seperti pada tabel 4.1.

Tabel 4. 1. Pengujian Akurasi Sensor Kecepatan

No.	Speedometer	Serial Monitor	Akurasi
1	5 km/h	6 km/h	20,00%
2	10 km/h	10 km/h	0,00%
3	15 km/h	16 km/h	6,67%
4	20 km/h	20 km/h	0,00%
5	25 km/h	26 km/h	4,00%
6	30 km/h	31 km/h	3,33%
7	35 km/h	36 km/h	2,86%
8	40 km/h	40 km/h	0,00%
9	45 km/h	46 km/h	2,22%
10	50 km/h	50 km/h	0,00%
11	55 km/h	56 km/h	1,82%
12	60 km/h	60 km/h	0,00%
13	65 km/h	66 km/h	1,54%
14	70 km/h	70 km/h	0,00%
15	75 km/h	76 km/h	1,33%
16	80 km/h	80 km/h	0,00%
Rata-rata			2,74%

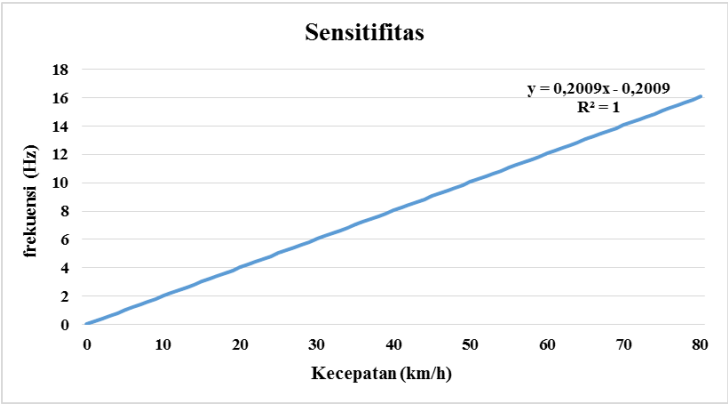
Pada tabel 4.1 dapat dilihat hasil pengujian dari akurasi sensor kecepatan. Pengujian dilakukan pada kecepatan 5 km/h hingga 80 km/h dengan variasi tiap 5 km/h. Dari hasil pengujian menunjukkan akurasi rata-rata sensor kecepatan sebesar 2,74% Dapat dilihat rata-rata hasil pengukuran mendekati grafik rata-rata pengukuran ideal dengan rata-rata error 2,74%, sehingga akurasi dapat dikatakan sangat tinggi yaitu 97,26 %.

Kemudian dilanjutkan dengan pengujian sensitifitas sensor. Pada pengujian sensitifitas sensor, dilakukan dengan mengatur kecepatan kendaraan pada kecepatan yang ingin diukur dan memperhatikan kecepatan yang terukur pada *speedometer*, kemudian keluaran frekuensi yang terukur pada *serial monitor* arduino dicatat. Hasil pengujiannya seperti pada tabel 4.2.

Tabel 4. 2. Pengujian Sensitifitas Sensor Kecepatan

No.	*v* (km/h)	*f* (Hz)
1	0	0,000
2	5	1,004
3	10	2,009
4	15	3,013
5	20	4,017
6	25	5,022
7	30	6,026
8	35	7,031
9	40	8,035
10	45	9,039
11	50	10,044
12	55	11,048
13	60	12,052
14	65	13,057
15	70	14,061
16	75	15,065
17	80	16,070

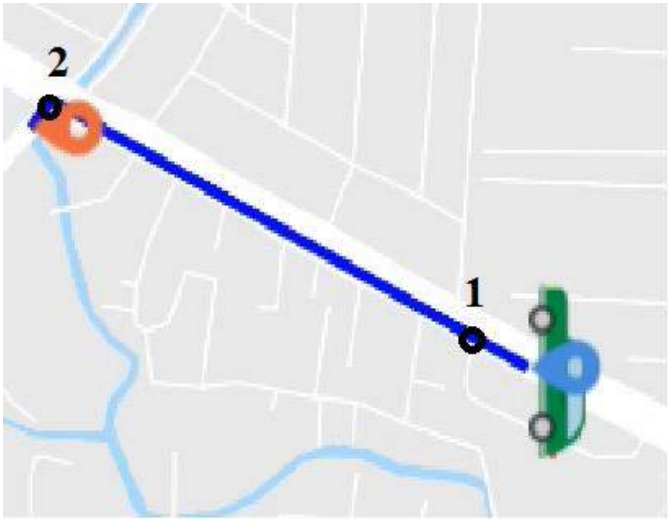
Pada tabel 4.2 dapat dilihat hasil pengujian dari sensitifitas sensor kecepatan. Pengujian dilakukan pada kecepatan 0 km/h hingga 80 km/h dengan variasi tiap 5 km/h. Hasil dari pengujian tersebut kemudian diplot pada grafik dan menghasilkan grafik seperti gambar di bawah ini.



Gambar 4. 1. Grafik Hasil Pengujian Sensitifitas Sensor

Dari grafik hasil pengujian sensitifitas sensor kecepatan didapatkan fungsi transfer yang menginformasikan bahwa sensor kecepatan ini mengkonversi setiap perubahan frekuensi 1 Hz menjadi perubahan kecepatan sebesar 0,2009 km/h dengan kegelinciran 0,2009. Jadi sensitifitasnya adalah 0,2009 km/h tiap hertz frekuensinya

Kemudian pengujian sensor kecepatan yang ketiga bertujuan untuk mengetahui pembacaan sensor pengukuran kecepatan pada kendaraan saat kendaraan berjalan di lintasan sebenarnya, sehingga diketahui nilai presentase *error* dari sensor kecepatan yang dirancang. Pada pengujian sistem pengukuran kecepatan ini dilakukan pengujian dengan mencatat waktu yang dibutuhkan oleh kendaraan saat berjalan dengan kecepatan konstan dari titik awal ke titik akhir sejauh 500 meter, kemudian dibandingkan dengan hasil perhitungan teori. Pengujian sensor kecepatan ini dilakukan dengan variasi kecepatan kendaraan pada 20, 40 dan 60 km/h dengan masing-masing variasi 5 kali pengambilan data.



Gambar 4. 2. Lintasan Pengujian Sensor Kecepatan

Pada pengujian kalibrasi sensor pengukuran kecepatan ini kendaraan akan berjalan dari titik biru pada peta hingga berhenti pada titik jingga dengan jarak 600 meter, namun waktu direkam mulai dari titik 1 hingga titik 2 yaitu sejauh 500 meter, 100 meter pertama mulai dari titik biru hingga titik 1 digunakan untuk mencapai kecepatan yang ditentukan, sehingga ketika pada titik 1 kendaraan telah berjalan pada kecepatan yang ditentukan dan konstan hingga titik 2.

Pengujian dimulai dengan berjalan konstan pada kecepatan 20 km/h. Waktu yang dibutuhkan untuk berjalan dari titik 1 menuju titik 2 sejauh 500 meter dengan kecepatan konstan 20 km/h menurut perhitungan menggunakan persamaan 2.1 sebagai berikut:

$$t = \frac{s}{v}$$

$$= \frac{0{,}5}{20}$$

= 0,025 jam = 90 sekon

Tabel 4. 3. Pengujian Sensor pada Kecepatan 20 km/h

Data ke-	Waktu Pengukuran (s)	Waktu Perhitungan (s)	Error (%)
1	98	90	8.89
2	92	90	2.22
3	89	90	1.11
4	93	90	3.33
5	92	90	2.22

Tabel 4.3 diatas merupakaan data hasil pengujian sensor kecepatan pada saat kecepatan 20 km/h. Dari tabel tersebut didapati nilai presentase *error* dari sensor pengukuran kecepatan dibandingkan dengan perhitungan teori untuk masing-masing pengambilan data, lalu dihitung nilai presentase *error* rata-ratanya. Nilai kesalahan pembacaan rata-rata dari 5 kali pengambilan yaitu sebesar 3,55%.

Pengujian dilanjutkan dengan berjalan konstan pada kecepatan 40 km/h. Waktu yang dibutuhkan untuk berjalan dari

titik 1 menuju titik 2 sejauh 500 meter dengan kecepatan konstan 40 km/h menurut perhitungan menggunakan persamaan 2.1 sebagai berikut:

$$t = \frac{s}{v}$$

$$t = \frac{0{,}5}{40}$$

= 0,0125 jam = 45 sekon

Tabel 4. 4 Pengujian Sensor Pada Kecepatan 40 km/h

Data ke-	Waktu Pengukuran (s)	Waktu Perhitungan (s)	Error (%)
1	48	45	6.67
2	47	45	4.44
3	46	45	2.22
4	47	45	4.44
5	47	45	4.44

Tabel 4.4 merupakaan data hasil pengujian sensor kecepatan pada saat kecepatan 40 km/h. Dari tabel tersebut didapati nilai presentase *error* dari sensor pengukuran kecepatan dibandingkan dengan perhitungan teori untuk masing-masing pengambilan data, lalu dihitung nilai presentase *error* rata-ratanya. Nilai kesalahan pembacaan rata-rata dari 5 kali pengambilan yaitu sebesar 4,44%.

Pengujian dilanjutkan dengan berjalan konstan pada kecepatan 60 km/h. Waktu yang dibutuhkan untuk berjalan dari titik 1 menuju titik 2 sejauh 500 meter dengan kecepatan konstan 60 km/h menurut perhitungan menggunakan persamaan 2.1 sebagai berikut:

$$t = \frac{s}{v}$$

$$t = \frac{0{,}5}{60}$$

= 0,0083 jam = 30 sekon

Tabel 4. 5. Pengujian Sensor pada Kecepatan 60 km/h

Data ke-	Waktu Pengukuran (s)	Waktu Perhitungan (s)	Error (%)
1	31	30	3.33
2	29	30	3.33
3	32	30	6.67
4	31	30	3.33
5	32	30	6.67

Tabel 4.5 diatas merupakaan data hasil pengujian sensor kecepatan pada saat kecepatan 60 km/h. Dari tabel tersebut didapati nilai presentase *error* dari sensor pengukuran kecepatan dibandingkan dengan perhitungan teori untuk masing-masing pengambilan data, lalu dihitung nilai presentase *error* rata-ratanya. Nilai kesalahan pembacaan rata-rata dari 5 kali pengambilan yaitu sebesar 4,67%.

Dari tiga variasi kecepatan yang digunakan pada pengujian ini, kemudian dihitung rata-rata *error* pengukuran dibandingkan dengan perhitungan teori. Pada pengujian dengan kecepatan 20 km/h *error* pengukurannya 3,55%, pada pengujian dengan kecepatan 40 km/h *error* pengukurannya 4,44% dan pada pengujian dengan kecepatan 60 km/h *error* pengukurannya 4,67%, sehingga rata-rata *error*nya didapatkan sebesar 4,22%.

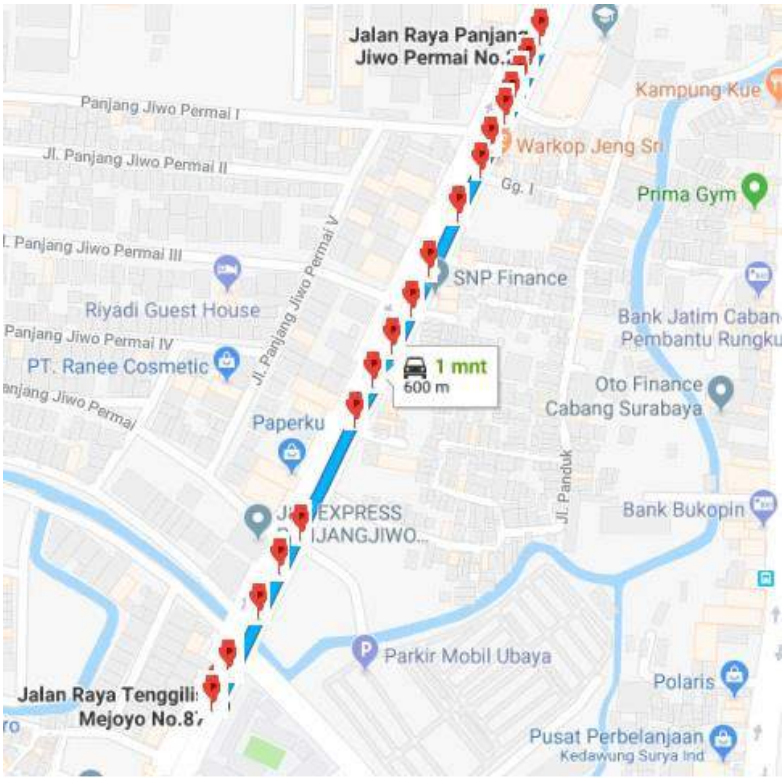
4.1.2 Pengujian Sistem GPS

Pengujian dilanjutkan dengan pengujian sensor GPS. Pengujian sistem GPS ini dilakukan untuk mengetahui apakah sensor GPS telah dapat mengirim lokasi dengan tepat. Pengujian akan dilakukan dengan berjalan pada lintasan lurus sejauh 500 meter, ini bertujuan untuk mendapatkan nilai *longitude* dan *latitude* dari tiap perpindahan pada lintasan. Pada pengujian ini diambil sebanyak 20 data yang masuk beruntun pada *web server*. Kemudian data yang masuk tersebut akan divalidasi dengan *google maps* untuk mengetahui lokasinya pada tampilan peta. Adapun hasil pengujian dapat dilihat pada tabel 4.6.

Tabel 4. 6. Pembacaan Sensor GPS

Data ke-	Waktu	Kecepatan (km/h)	Latitude	Longitude
1	2018-06-22 15:50:42 UTC	0	-7.310135	112.76765
2	2018-06-22 15:50:48 UTC	0	-7.310135	112.76765
3	2018-06-22 15:50:51 UTC	15	-7.310233	112.76760
4	2018-06-22 15:50:54 UTC	28	-7.310323	112.76758
5	2018-06-22 15:50:57 UTC	40	-7.310460	112.76750
6	2018-06-22 15:51:00 UTC	41	-7.310613	112.76743
7	2018-06-22 15:51:03 UTC	40	-7.310873	112.76729
8	2018-06-22 15:51:06 UTC	40	-7.311097	112.76717
9	2018-06-22 15:51:09 UTC	40	-7.311370	112.76706
10	2018-06-22 15:51:15 UTC	40	-7.311906	112.76668
11	2018-06-22 15:51:18 UTC	39	-7.312160	112.76668
12	2018-06-22 15:51:21 UTC	39	-7.312421	112.76656
13	2018-06-22 15:51:24 UTC	40	-7.312677	112.76641
14	2018-06-22 15:51:27 UTC	39	-7.312987	112.76627
15	2018-06-22 15:51:36 UTC	41	-7.313791	112.76586
16	2018-06-22 15:51:39 UTC	40	-7.314077	112.76572
17	2018-06-22 15:51:42 UTC	40	-7.314336	112.76557
18	2018-06-22 15:51:45 UTC	27	-7.314887	112.76532
19	2018-06-22 15:51:48 UTC	0	-7.314993	112.76527
20	2018-06-22 15:51:51 UTC	0	-7.314993	112.76527

Tabel 4.6 merupakaan titik koordinat hasil pembacaan sensor GPS saat kendaraan sedang berjalan. Kemudian dilakukan validasi melalui *google maps* apakah data pembacaan sensor GPS telah sesuai. Kemudian dilakukan ploting hasil validasi titik koordinat pada *google maps* seperti ditunjukkan pada gambar 4.3.



Gambar 4. 3. Hasil Validasi Koordinat GPS pada *Google Maps*

Dari gambar 4.3 dapat dilihat posisi kendaraan yang terbaca oleh sensor GPS. Setiap data yang terdapat pada tabel 4.6 ditunjukkan oleh pin berwarna merah pada gambar 4.3, data pertama hingga 20 ditunjukkan dari pin teratas hingga terbawah pada gambar 4.3. Data yang didapat telah sesuai dengan posisi kendaraan pada saat itu.

4.1.3 Pengujian Sistem Pengiriman Data

Pada pengujian sistem pengiriman data ini bertujuan untuk mengetahui data sensor kecepatan dan sensor GPS telah dapat diterima pada *thingspeak*.

Tabel 4. 7. Penerimaan Data pada *Web Server*

Data ke-	Waktu	Kecepatan (km/h)	Latitude	Longitude
1	2018-06-22 15:50:42 UTC	0	-7.310135	112.76765
2	2018-06-22 15:50:48 UTC	0	-7.310135	112.76765
3	2018-06-22 15:50:51 UTC	15	-7.310233	112.76760
4	2018-06-22 15:50:54 UTC	28	-7.310323	112.76758
5	2018-06-22 15:50:57 UTC	40	-7.310460	112.76750
6	2018-06-22 15:51:00 UTC	41	-7.310613	112.76743
7	2018-06-22 15:51:03 UTC	40	-7.310873	112.76729
8	2018-06-22 15:51:06 UTC	40	-7.311097	112.76717
9	2018-06-22 15:51:09 UTC	40	-7.311370	112.76706
10	2018-06-22 15:51:15 UTC	40	-7.311906	112.76668
11	2018-06-22 15:51:18 UTC	39	-7.312160	112.76668
12	2018-06-22 15:51:21 UTC	39	-7.312421	112.76656
13	2018-06-22 15:51:24 UTC	40	-7.312677	112.76641
14	2018-06-22 15:51:27 UTC	39	-7.312987	112.76627
15	2018-06-22 15:51:36 UTC	41	-7.313791	112.76586
16	2018-06-22 15:51:39 UTC	40	-7.314077	112.76572
17	2018-06-22 15:51:42 UTC	40	-7.314336	112.76557
18	2018-06-22 15:51:45 UTC	27	-7.314887	112.76532
19	2018-06-22 15:51:48 UTC	0	-7.314993	112.76527
20	2018-06-22 15:51:51 UTC	0	-7.314993	112.76527

Pada pengujian pengiriman data ini, *thingspeak* mendapatkan masukan data kecepatan, data *latitude* dan *longitude* yang dikirim dari mikrokontroller. Pada saat kendaraan berjalan melalui lintasan pengujian diambil sebanyak 20 data yang hasilnya seperti tersaji pada tabel 4.7. Rentang waktu pengiriman dari mikrokontroller pada kondisi ideal adalah 3 sekon. Dari 20 data yang berhasil diterima oleh *thingspeak* ketika kendaraan berjalan dapat dilihat pada data pertama dan kedua terdapat rentang waktu 6 sekon yang berarti terdapat 1 data yang tidak berhasil diterima oleh *thingspeak*.

Begitu pula pada data ke-9 da ke-10 yang terdapat rentang waktu 6 sekon yang berarti terdapat 1 data lagi yang tidak berhasil diterima oleh *thingspeak*. Sedangkan antara data ke-14 dan ke-15 terdapat rentang waktu 9 sekon yang berarti terdapat 2 data yang tidak berhasil diterima oleh *thingspeak*. Dari 20 data yang diterima oleh *thingspeak* ternyata terdapat 4 data yang tidak berhasil diterima dalam rentang waktu tersebut, sehingga presentase keberhasilan pengiriman pada saat kendaraan berjalan melalui lintasan pengujian dapat dihitung sebagai berikut:

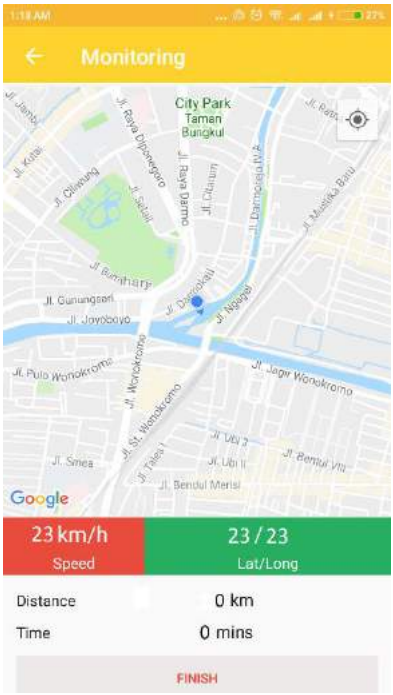
$$\frac{20}{24}\ x\ 100\ \% = 83{,}33\ \%$$

Pada pengujian pengiriman data ini didapati *delay* dari pembacaan data pada sensor hingga diterima oleh *thingspeak* sebesar 7 detik.

4.1.4 Pengujian Masukan Aplikasi Android

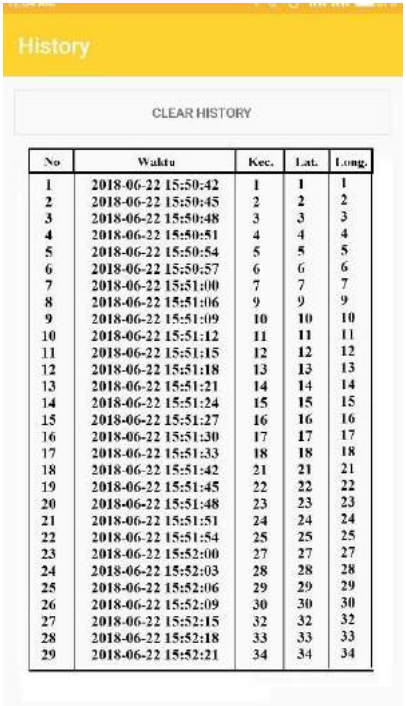
Pengujian selanjutnya yaitu pada sisi perancangan aplikasi android. Pengujian ini yaitu pengujian masukan aplikasi android. Pengujian masukan aplikasi android ini bertujuan untuk mengetahui presentase keberhasilan aplikasi android dalam menerima masukan data kecepatan, *latitude*, dan *longitude* dari *web server*. Pengujian ini penting untuk dilakukan guna memastikan data yang dikirimkan oleh mikrokontroller berhasil diterima sebelum nantinya diproses dan ditampilkan pada aplikasi android.

Tahapan pengujian ini akan dilakukan dengan mengirim data *dummy* berupa angka 1 hingga 100 tiap 3 detik dari mikrokontroller ke *web server*, kemudian *web server* akan mengirimkan ke aplikasi android. Presentase keberhasilan akan dihitung dengan perbandingan data yang berhasil diterima dibanding data yang dikirim. Oleh karena itu akan dilakukan pengecekan data yang dikeluarkan oleh mikrokontroller, data yang masuk ke *web server* dan data yang masuk pada aplikasi android.



Gambar 4. 4. Proses Pengujian Masukan pada Aplikasi Android

Pada gambar 4.4 menunjukkan tampilan aplikasi android saat pengujian *input*. Data yang diterima berupa angka 1 hingga 100 yang diperbarui tiap 3 detik. Setelah seluruh data telah dikirim oleh mikrokontroller, kemudian dilakukan pengecekan *history* masukan pada aplikasi android. Terdapat beberapa data yang dikirim dari mikrokontroller yang tidak diterima oleh aplikasi android, kemudian dilakukan pengecekan pada *web server* apakah data yang tidak masuk pada aplikasi android tersebut masuk pada *web server* atau tidak. Setelah dilakukan pengecekan pada *web server* didapati bahwa data yang tidak masuk pada android juga tidak masuk pada *web server* sehingga dapat disimpulkan data *loss* terjadi antara mikrokontroller dan *web server*, bukan pada *web server* dan aplikasi android. Seluruh data yang ada pada *web server* telah berhasil masuk pada aplikasi android, sehingga didapati presentase keberhasilan masukan data pada aplikasi android adalah 100%.



No	Waktu	Kec.	Lat.	Long.
1	2018-06-22 15:50:42	1	1	1
2	2018-06-22 15:50:45	2	2	2
3	2018-06-22 15:50:48	3	3	3
4	2018-06-22 15:50:51	4	4	4
5	2018-06-22 15:50:54	5	5	5
6	2018-06-22 15:50:57	6	6	6
7	2018-06-22 15:51:00	7	7	7
8	2018-06-22 15:51:06	9	9	9
9	2018-06-22 15:51:09	10	10	10
10	2018-06-22 15:51:12	11	11	11
11	2018-06-22 15:51:15	12	12	12
12	2018-06-22 15:51:18	13	13	13
13	2018-06-22 15:51:21	14	14	14
14	2018-06-22 15:51:24	15	15	15
15	2018-06-22 15:51:27	16	16	16
16	2018-06-22 15:51:30	17	17	17
17	2018-06-22 15:51:33	18	18	18
18	2018-06-22 15:51:42	21	21	21
19	2018-06-22 15:51:45	22	22	22
20	2018-06-22 15:51:48	23	23	23
21	2018-06-22 15:51:51	24	24	24
22	2018-06-22 15:51:54	25	25	25
23	2018-06-22 15:52:00	27	27	27
24	2018-06-22 15:52:03	28	28	28
25	2018-06-22 15:52:06	29	29	29
26	2018-06-22 15:52:09	30	30	30
27	2018-06-22 15:52:15	32	32	32
28	2018-06-22 15:52:18	33	33	33
29	2018-06-22 15:52:21	34	34	34

Gambar 4. 5 Tampilan Masukan pada Aplikasi Android

Gambar 4.5 di atas adalah tampilan *history* masukan yang terdapat pada aplikasi android. Pada gambar dapat dilihat bahwa data yang dikirim oleh mikrokontroller, yaitu yang berupa data *dummy* baik pada data kecepatan, *latitude*, maupun *longitude* yang berupa angka 1 hingga 100 yang dikirim tiap 3 detik telah dapat diterima pada aplikasi android. Seluruh data yang ada pada *web server* telah dapat diterima pada aplikasi android dengan dilakukannya pengecekan pada *history* masukan data. Pada pengujian pengiriman data ini didapati *delay* dari pengiriman data dari mikrokontroller hingga diterima oleh aplikasi android selama 10 detik.

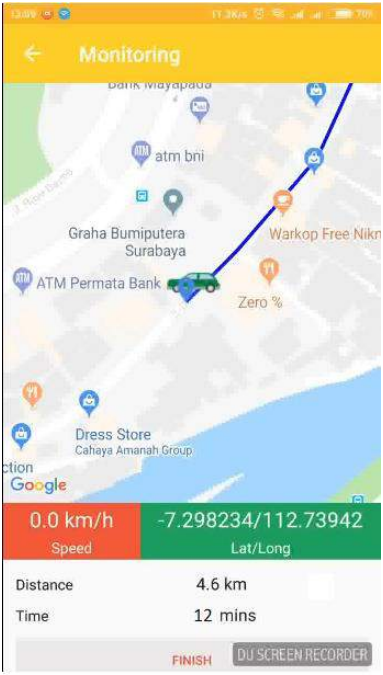
4.2 Pengujian Integrasi Sistem

Pengujian integrasi sistem ini dilakukan untuk mengetahui kinerja dari sistem ini pada berbagai kondisi lalu lintas. Pengujian dilakukan pada tiga rute berbeda dengan masing-masing dalam dua waktu yang berbeda. Hal ini dilakukan agar mengetahui kinerja sistem pada rute yang sama namun kondisi lalu lintas yang relatif berbeda. Kemudian dilanjutkan dengan satu rute terakhir yang memiliki kepadatan lalu lintas, rute terakhir dilakukan dua kali pula dengan pengujian pertama kendaraan berjalan dengan kecepatan diatas rata-rata kecepatan kendaraan lainnya dan yang terakhir dengan kecepatan seperti kendaraan disekitarnya. Hal ini ditujukan agar mengetahui kinerja sistem pada kondisi rute yang sama namun dengan kecepatan tempuh yang berbeda.

Pengujian integrasi sistem perhitungan waktu tempuh yang dilakukan pada tiga rute pertama dilakukan di dua waktu yang berbeda yaitu siang (pukul 14.00-16.00) dan malam (pukul 19.00-21.00). Pada pengujian rute ke empat dilakukan pada pukul 17.00 dengan pengujian pertama kendaraan berjalan dengan kecepatan di atas kecepatan rata-rata kendaraan pada umumnya dan pengujian kedua dilakukan dengan kecepatan kendaraan mengikuti kecepatan kendaraan rata-rata pada kondisi lalu lintas saat itu. Hal tersebut dilakukan untuk mengetahui kemampuan sistem ketika diterapkan pada kendaraan yang membutuhkan urgensi untuk segera tiba di tempat tujuan, semisal kendaraan pemadam kebakaran ataupun *ambulance*.

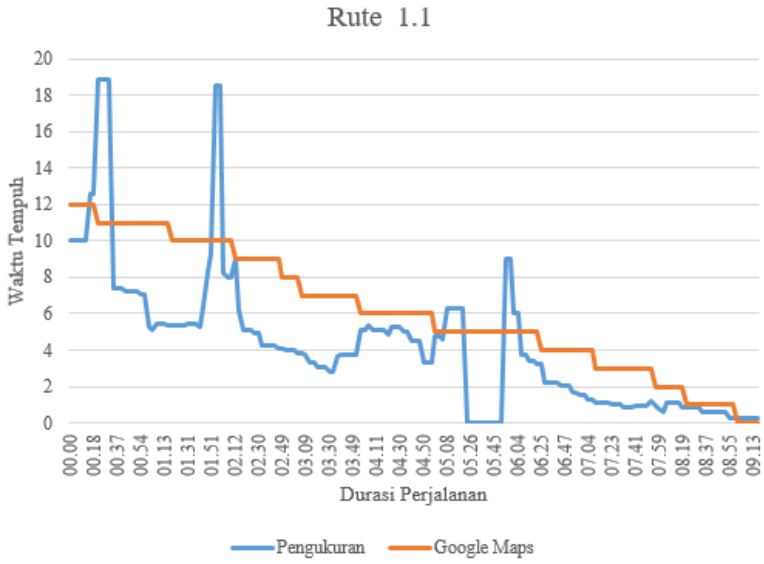
- Rute Pengujian 1

Rute pengujian pertama dilakukan dari lokasi asal di Jl. Darmokali menuju Surabaya Plaza pada pukul 14.00 WIB. Rute yang ditempuh sejauh 4,5 km dengan estimasi waktu awal berdasarkan estimasi *google maps* yaitu 12 menit untuk sampai di lokasi tujuan. Tampilan aplikasi saat perjalanan akan dimulai seperti ditampilkan pada gambar 4.6.



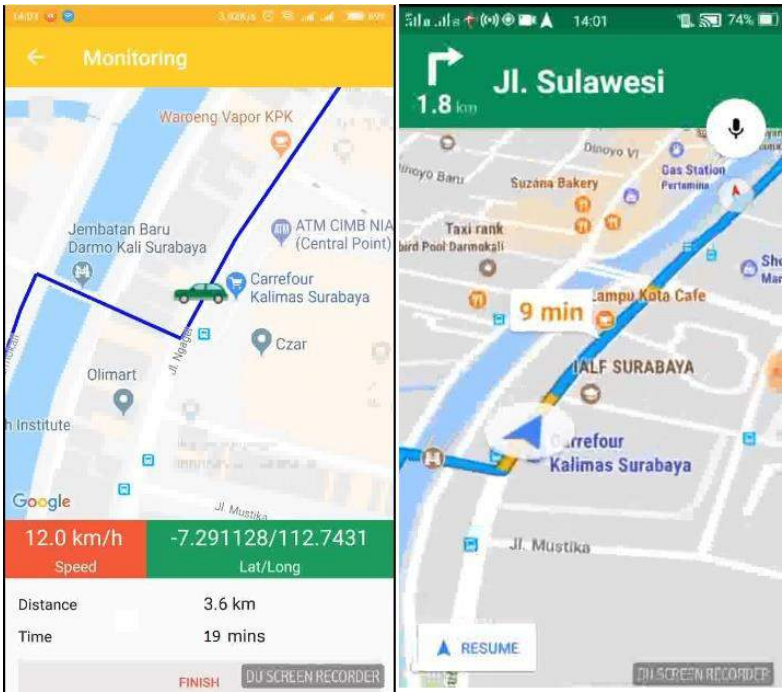
Gambar 4. 6. Tampilan Awal Pengujian Rute 1 Pertama

Pada gambar 4.6 menampilkan tampilan aplikasi saat akan memulai pengujian. Kecepatan masih menunjukkan 0 km/h karena kendaraan belum berjalan dan waktu menampilkan estimasi awal *google maps* yaitu 12 menit. Rute yang ditempuh relatif lancar dengan beberapa kali berhenti di *traffic light*. Pada saat berhenti di *traffic light*, waktu tempuh yang tertampil adalah estimasi waktu *google maps*, hal ini untuk menghindari nilai waktu tempuh tak terhingga dikarenakan nilai kecepatan terbaca sensor adalah 0 saat kendaraan sedang berhenti. Rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 1 yang pertama dapat dilihat pada gambar 4.7.



Gambar 4. 7. Grafik Pengukuran Waktu Tempuh pada Pengujian Rute 1 Pertama

Dari grafik pengukuran waktu tempuh pada pengujian rute 1 yang pertama dapat dilihat terdapat fluktuasi pengukuran waktu tempuh. Fluktuasi tersebut dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui. Terdapat 2 kali kenaikan signifikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga harus mengurangi kecepatan. Pada saat pengujian berjalan sekitar menit ke dua, terjadi kenaikan hasil pengukuran waktu tempuh, hal tersebut didapati karena kendaraan sedang berada pada kondisi lalu lintas yang cenderung padat sehingga kecepatan kendaraan menurun dan perhitungan waktu tempuh meningkat. Pada saat kendaraan sedang berada pada kondisi tersebut dapat dilihat pada gambar 4.8.



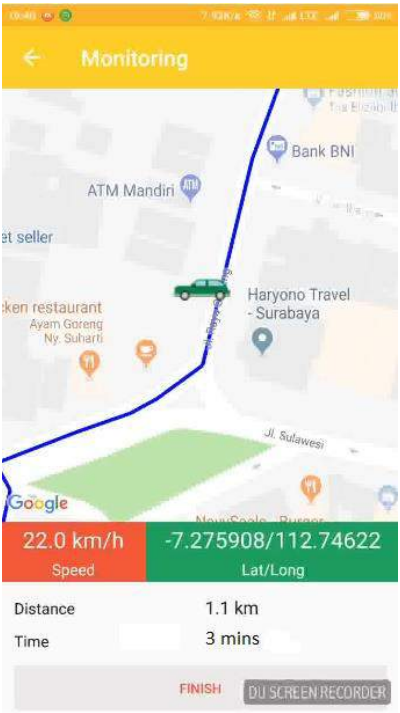
Gambar 4. 8. Tampilan Pengujian Saat dibandingkan dengan *Google Maps*

Gambar 4.8 menunjukkan kondisi ketika terjadi kenaikan waktu tempuh perjalanan yang ternyata hasil tampilan *google maps* menunjukkan kendaraan memang berada pada kondisi lalu lintas yang cenderung padat yang ditampilkan dengan warna rute berwarna jingga. Terdapat nilai 0 pada saat pengujian berjalan sekitar menit ke-5 dikarenakan kendaraan sedang berhenti pada *traffic light.* Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 9 menit 16 detik, yang berarti lebih cepat 3 menit dari estimasi *google maps*.

Dari hasil pengujian pertama rute 1 kendaraan berjalan dengan kecepatan rata-rata 32 km/h dengan kecepatan maksimal 56 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian

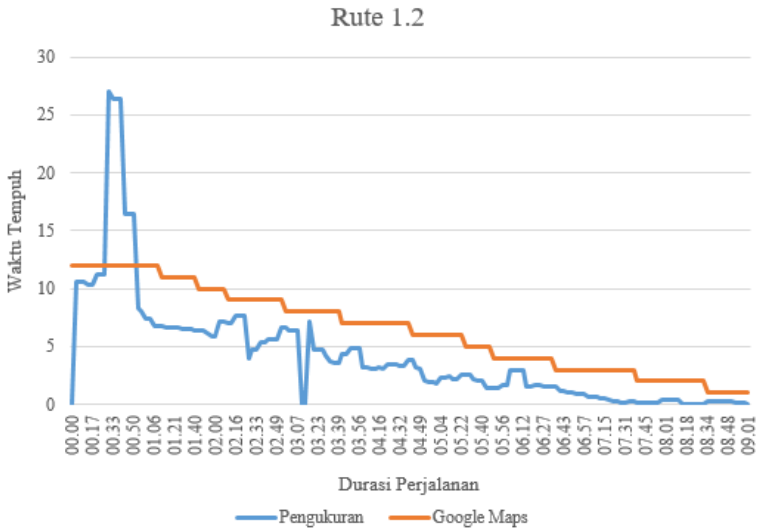
pertama rute 1 ini memiliki selisih rata-rata 2 menit lebih cepat bila dibandinganken dengan hasil estimasi *google maps*.

Kemudian pada pengujian kedua dilakukan pada rute yang sama namun pada waktu yang berbeda, yaitu pukul 19.30 WIB. Estimasi waktu awal berdasarkan estimasi *google maps* yaitu 12 menit untuk sampai di lokasi tujuan. Tampilan aplikasi saat perjalanan seperti ditampilkan pada gambar 4.9.



Gambar 4. 9. Tampilan Saat Pengujian Rute 1 Kedua

Pada gambar 4.9 menampilkan tampilan aplikasi saat pengujian rute 1 yang kedua. Kondisi lalu lintas pada rute yang ditempuh relatif sama dengan kondisi pada pengujian waktu pertama. Rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 1 yang kedua dapat dilihat pada gambar 4.10.



Gambar 4. 10. Grafik Pengukuran Waktu Tempuh pada Pengujian Rute 1 Kedua

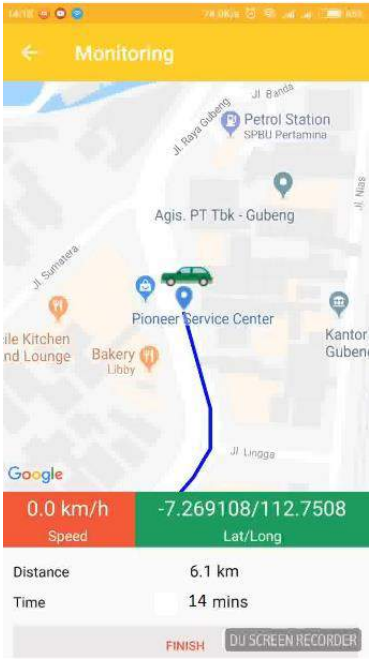
Dari grafik pengukuran waktu tempuh pada pengujian rute 1 yang kedua dapat dilihat terdapat fluktuasi pengukuran waktu tempuh. Fluktuasi tersebut dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui. Terdapat satu kenaikan signifikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga kecepatan kendaraan jauh di bawah rata-rata dan perhitungan waktu tempuhnya meningkat. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 9 menit 1 detik, yang berarti kembali lebih cepat 3 menit dari estimasi *google maps*.

Dari hasil pengujian kedua rute 1 kendaraan berjalan dengan kecepatan rata-rata 34 km/h dengan kecepatan maksimal 62 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 2 menit lebih cepat bila dibandingankan dengan hasil estimasi *google maps*. Hasil pengujian rute 1 yang pertama dan kedua tidak memiliki perbedaan

yang signifikan karena kondisi lalu lintas saat dilakukan kedua pengujian tersebut relatif sama.

- Rute Pengujian 2

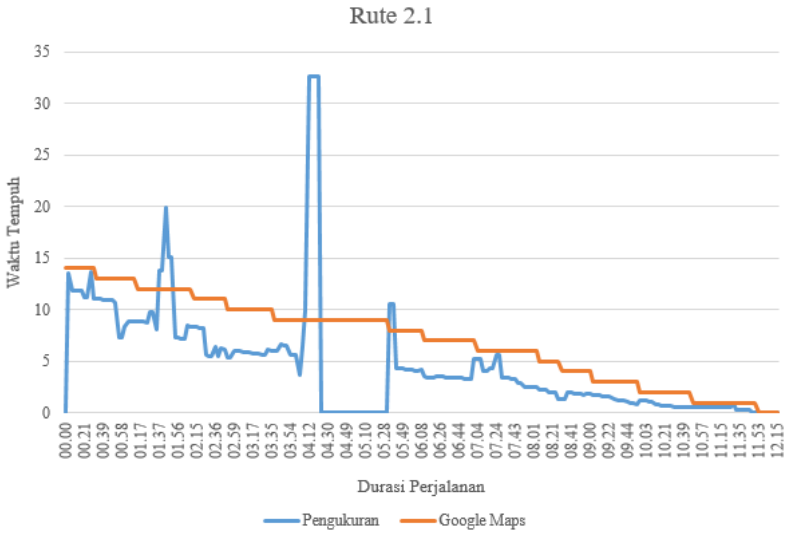
Rute pengujian kedua dilakukan dari lokasi asal di Jl. Raya Gubeng menuju Taman Alumni ITS pada pukul 14.30 WIB. Rute yang ditempuh sejauh 6,1 km dengan estimasi waktu awal berdasarkan estimasi *google maps* yaitu 14 menit untuk sampai di lokasi tujuan. Tampilan aplikasi saat perjalanan akan dimulai seperti ditampilkan pada gambar 4.11.



Gambar 4. 11. Tampilan Awal Pengujian Rute 2 Pertama

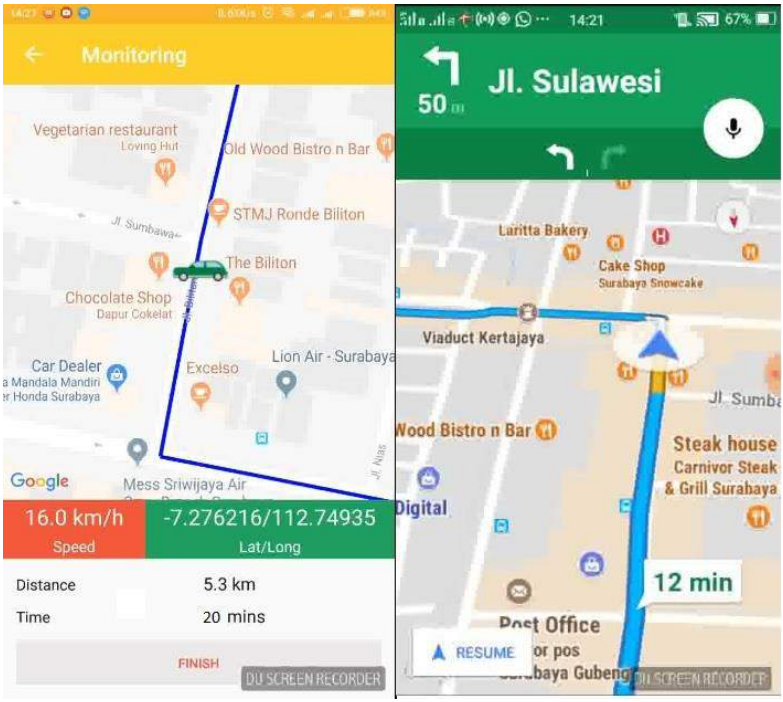
Pada gambar 4.11 menampilkan tampilan aplikasi saat akan memulai pengujian. Kecepatan masih menunjukkan 0 km/h karena kendaraan belum berjalan dan waktu menampilkan estimasi awal *google maps* yaitu 14 menit. Rute yang ditempuh kali ini lebih panjang dari rute pengujian 1. Namun, kondisi lalu lintas relatif lancar dengan satu kali berhenti di *traffic light*. Rekaman data

pengukuran waktu tempuh selama perjalanan pada pengujian rute 2 yang pertama dapat dilihat pada gambar 4.12.



Gambar 4. 12. Grafik Pengukuran Waktu Tempuh pada Pengujian Rute 2 Pertama

Dari grafik pengukuran waktu tempuh pada pengujian rute 2 yang pertama dapat dilihat terdapat fluktuasi pengukuran waktu tempuh. Fluktuasi tersebut dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui. Terdapat 2 kali kenaikan signifikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga harus mengurangi kecepatan. Pada saat pengujian berjalan sekitar 1 menit 40 detik, terjadi kenaikan hasil pengukuran waktu tempuh, hal tersebut didapati karena kendaraan sedang berada pada kondisi lalu lintas yang cenderung padat sehingga kecepatan kendaraan menurun dan perhitungan waktu tempuh meningkat. Pada saat kendaraan sedang berada pada kondisi tersebut dapat dilihat pada gambar 4.13.

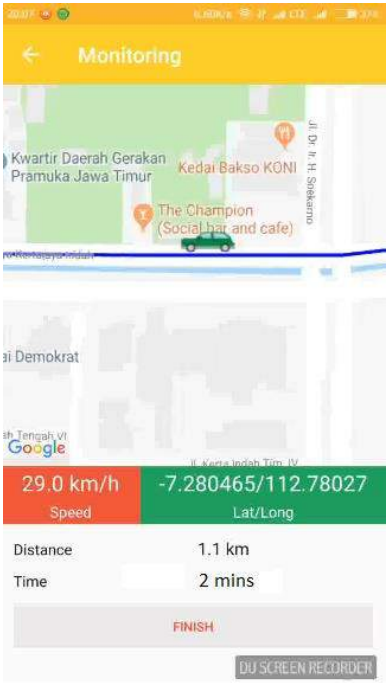


Gambar 4. 13. Tampilan Pengujian saat dibandingkan dengan *Google Maps*

Gambar 4.13 menunjukkan kondisi ketika terjadi kenaikan waktu tempuh perjalanan yang ternyata hasil tampilan *google maps* menunjukkan kendaraan memang berada pada kondisi lalu lintas yang cenderung padat yang ditampilkan dengan warna rute berwarna jingga. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 12 menit 15 detik, yang berarti lebih cepat 2 menit dari estimasi *google maps*.

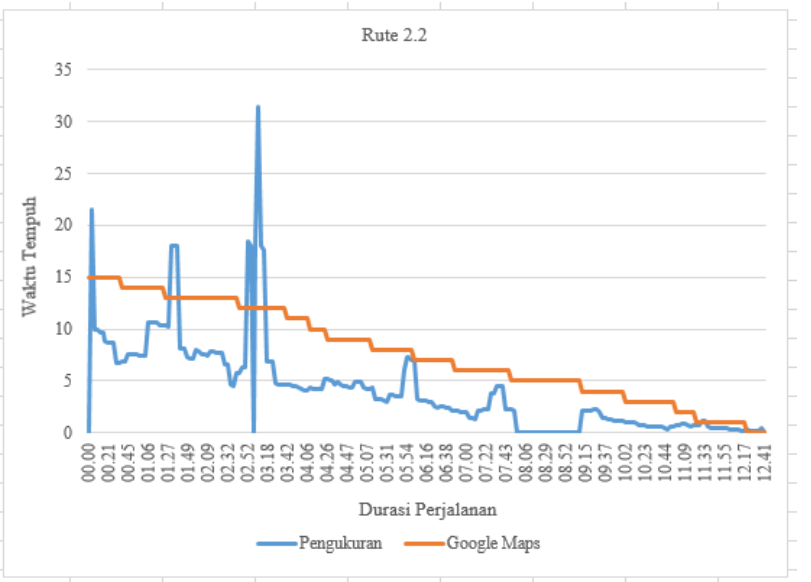
Dari hasil pengujian pertama rute 2 kendaraan berjalan dengan kecepatan rata-rata 35 km/h dengan kecepatan maksimal 80 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 2 menit lebih cepat bila dibandinganken dengan hasil estimasi *google maps*.

Kemudian pada pengujian kedua dilakukan rute yang sama namun pada waktu yang berbeda, yaitu pukul 20.00 WIB. Estimasi waktu awal berdasarkan estimasi *google maps* yaitu 15 menit untuk sampai di lokasi tujuan. Tampilan aplikasi saat perjalanan seperti ditampilkan pada gambar 4.14.



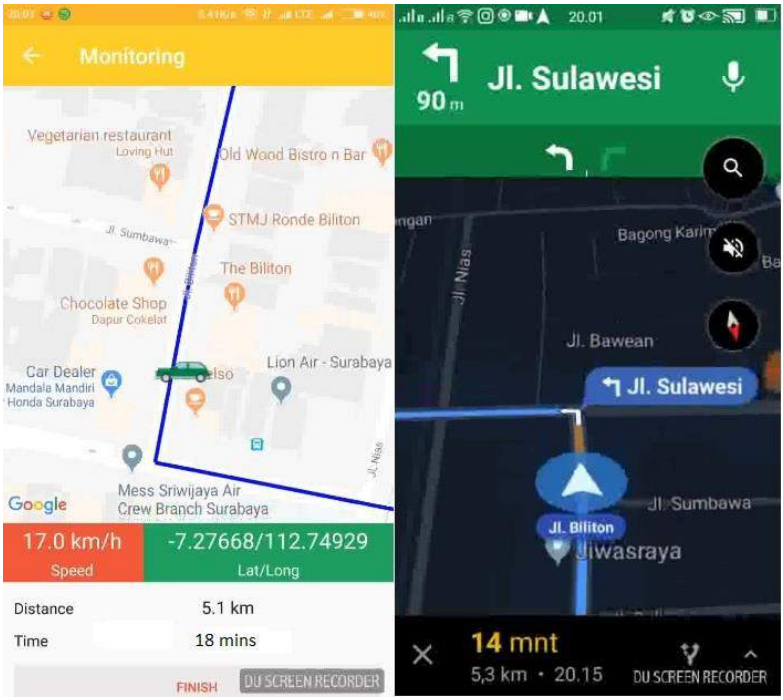
Gambar 4. 14. Tampilan Saat Pengujian Rute 2 Kedua

Pada gambar 4.14 menampilkan tampilan aplikasi saat pengujian rute 2 yang kedua. Kondisi lalu lintas pada rute yang ditempuh relatif sama dengan kondisi pada pengujian waktu pertama. Rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 2 yang kedua dapat dilihat pada gambar 4.15.



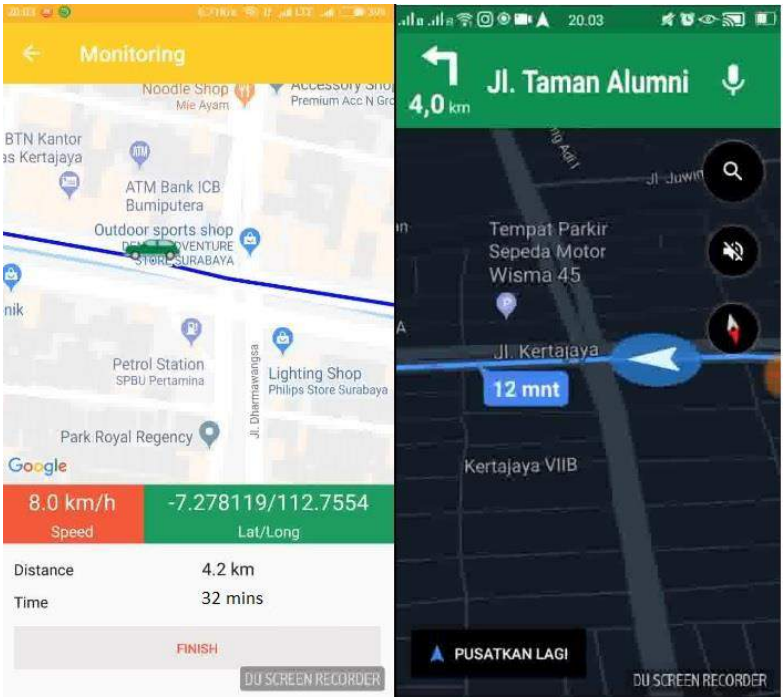
Gambar 4. 15. Grafik Pengukuran Waktu Tempuh pada Pengujian Rute 2 Kedua

Dari grafik pengukuran waktu tempuh pada pengujian rute 2 yang kedua dapat dilihat terdapat fluktuasi pengukuran waktu tempuh. Fluktuasi tersebut dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui. Pada pengujian rute 2 yang kedua ini juga terdapat satu kenaikan signifikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga kecepatan kendaraan jauh di bawah rata-rata dan perhitungan waktu tempuhnya meningkat. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 12 menit 41 detik, yang berarti kembali lebih cepat 2 menit dari estimasi *google maps*.



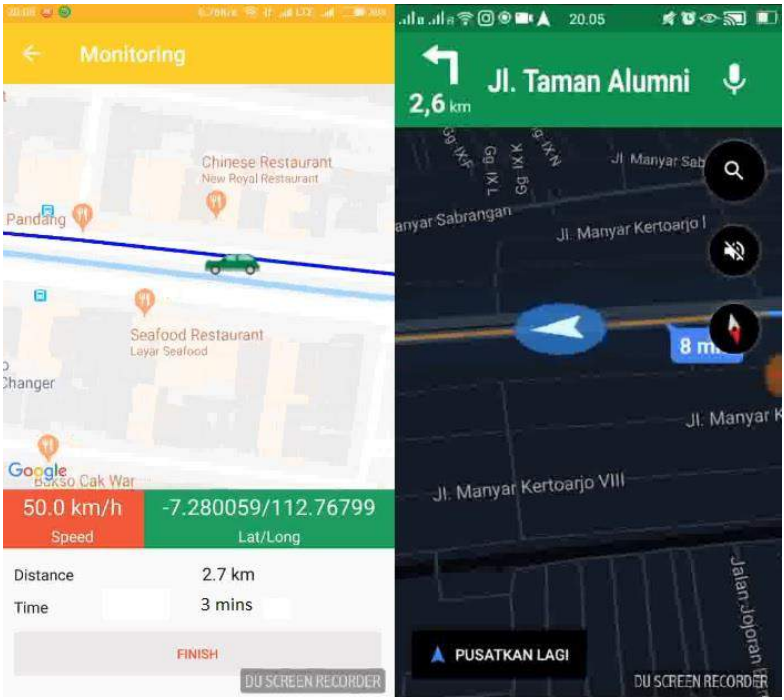
Gambar 4. 16. Tampilan Pengujian saat dibandingkan dengan *Google Maps*

Gambar 4.16 menunjukkan kondisi ketika terjadi kenaikan waktu tempuh perjalanan yang ternyata hasil tampilan *google maps* menunjukkan kendaraan memang berada pada kondisi lalu lintas yang cenderung padat yang ditampilkan dengan warna rute berwarna jingga. Namun tidak selalu ketika *google maps* menunjukkan kendaraan sedang berada pada rute berwarna jingga maupun merah akan meningkatkan hasil pengukuran waktu tempuh secara signifikan. Hal tersebut karena alat ukur waktu tempuh bekerja berdasarkan kecepatan kendaraan, walaupun kondisi lalu lintas yang dilalui padat, namun jika kendaraan dapat berjalan di tengah kepadatan dengan kecepatan di atas rata-rata kendaraan lain, maka hasil pengukuran tidak akan meningkat secara signifikan.



Gambar 4. 17. Tampilan Pengujian saat dibandingkan dengan *Google Maps*

Gambar 4.17 menampilkan ketika terjadi peningkatan hasil pengukuran waktu tempuh, hal tersebut dikarenakan kendaraan harus mengurangi kecepatan saat akan melewati persimpangan jalan. Peningkatan hasil pengukuran terjadi meskipun pada sisi sebelah kanan gambar yang menunjukkan *google maps* menunjukkan bahwa kendaraan tidak sedang berada pada rute yang padat menurut *google maps.* Hal tersebut kembali menunjukkan bahwa alat ukur waktu tempuh ini lebih dapat menyesuaikan dengan karakteristik pengemudi saat berkendara. Apabila pengemudi ingin menurunkan kecepatan kendaraannya maka hasil pengukuran waktu tempuh akan meningkat, begitu pula sebaliknya.



Gambar 4. 18. Tampilan Pengujian saat dibandingkan dengan *Google Maps*

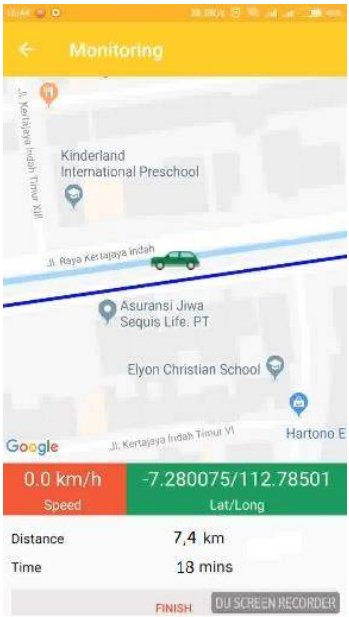
Gambar 4.18 bagian kanan menampilkan ketika kendaraan sedang berada pada lalu lintas yang cenderung padat yang ditunjukkan dengan warna jingga pada jalur yang dilalui pada *google maps.* Namun pada hasil pengukuran waktu tempuh tidak terjadi peningkatan, hal tersebut dikarenakan kendaraan dapat berjalan di atas kecepatan rata-rata kendaraan lainnya saat melewati kondisi lalu lintes tersebut. Tidak terjadi peningkatan hasil pengukuran meskipun pada sisi sebelah kanan gambar yang menunjukkan *google maps* menunjukkan kendaraan sedang berada pada rute yang cenderung padat menurut *google maps.* Hal tersebut kembali menunjukkan bahwa alat ukur waktu tempuh ini lebih dapat menyesuaikan dengan karakteristik pengemudi saat berkendara. Apabila pengemudi ingin menurunkan kecepatan

kendaraannya maka hasil pengukuran waktu tempuh akan meningkat, begitu pula sebaliknya.

Dari hasil pengujian kedua rute 2 kendaraan berjalan dengan kecepatan rata-rata 34 km/h dengan kecepatan maksimal 64 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 4 menit lebih cepat bila dibandinganken dengan hasil estimasi *google maps*. Hasil pengujian rute 2 yang pertama dan kedua juga tidak memiliki perbedaan yang signifikan karena kondisi lalu lintas saat dilakukan kedua pengujian tersebut relatif sama.

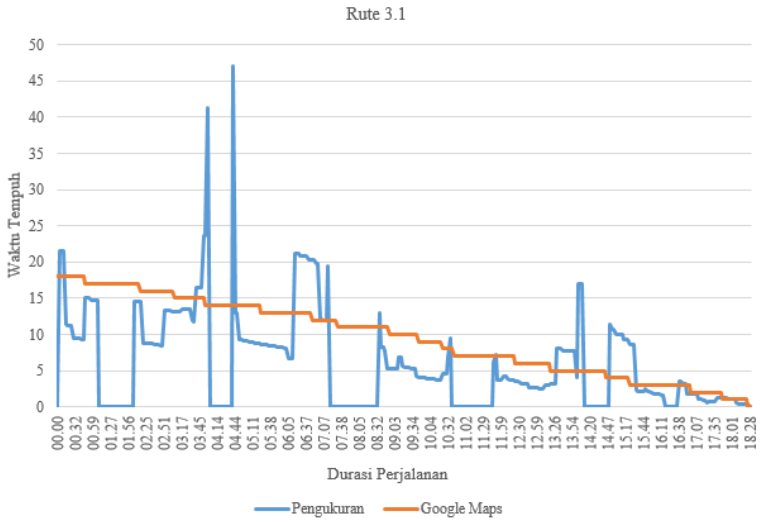
- Rute Pengujian 3

Rute pengujian ketiga dilakukan dari lokasi asal di Jl. Raya Kertajaya Indah menuju Universitas Surabaya Tenggilis pada pukul 15.30 WIB. Rute yang ditempuh sejauh 7,4 km dengan estimasi waktu awal berdasarkan estimasi *google maps* yaitu 18 menit untuk sampai di lokasi tujuan. Tampilan aplikasi saat perjalanan akan dimulai seperti ditampilkan pada gambar 4.19.



Gambar 4. 19. Tampilan Awal Pengujian Rute 3 Pertama

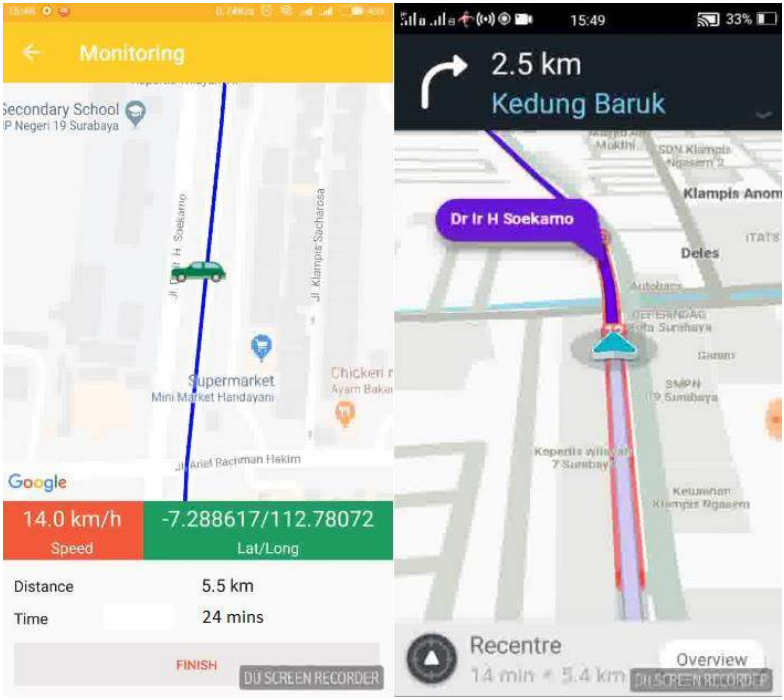
Pada gambar 4.19 menampilkan tampilan aplikasi saat akan memulai pengujian. Kecepatan masih menunjukkan 0 km/h karena kendaraan belum berjalan dan waktu menampilkan estimasi awal *google maps* yaitu 18 menit. Rute yang ditempuh pada pengujian ketiga ini lebih panjang dari kedua rute sebelumnya. Rute ini melalui beberapa *traffic light* yang dapat menimbulkan kepadatan lalu lintas. Rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 3 yang pertama dapat dilihat pada gambar 4.20.



Gambar 4. 20. Grafik Pengukuran Waktu Tempuh pada Pengujian Rute 3 Pertama

Dari grafik pengukuran waktu tempuh pada pengujian rute 3 yang pertama dapat dilihat lebih banyak terdapat fluktuasi pengukuran waktu tempuh. Fluktuasi tersebut dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui jauh lebih padat dari 2 rute sebelumnya. Terdapat beberapa kali kenaikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga harus mengurangi kecepatan. Pada saat pengujian berjalan sekitar 3 menit 50 detik,

terjadi kenaikan hasil pengukuran waktu tempuh, hal tersebut didapati karena kendaraan sedang berada pada kondisi lalu lintas yang cenderung padat sehingga kecepatan kendaraan menurun dan perhitungan waktu tempuh meningkat. Pada saat kendaraan sedang berada pada kondisi tersebut dapat dilihat pada gambar 4.21.

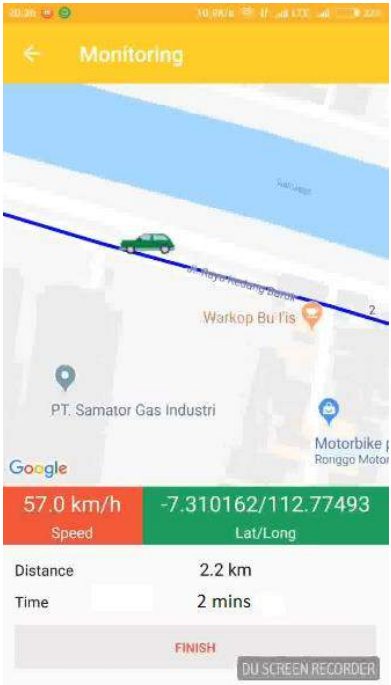


Gambar 4. 21. Tampilan Pengujian saat dibandingkan dengan *Google Maps*

Gambar 4.21 menunjukkan kondisi ketika terjadi kenaikan waktu tempuh perjalanan yang ternyata hasil tampilan *google maps* menunjukkan kendaraan memang berada pada kondisi lalu lintas yang padat yang ditampilkan dengan warna rute berwarna merah. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 18 menit 28 detik, yang berarti sama dengan estimasi *google maps*. Dari hasil pengujian pertama rute 3 kendaraan berjalan dengan kecepatan rata-rata 22 km/h dengan kecepatan maksimal 62 km/h.

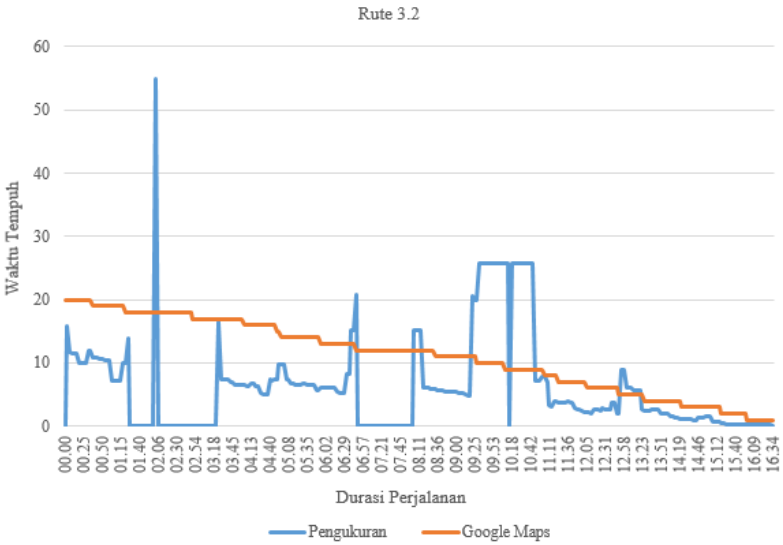
Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 3 ini memiliki selisih rata-rata 4 menit lebih cepat bila dibandingankan dengan hasil estimasi *google maps*.

Kemudian pada pengujian kedua dilakukan rute yang sama namun pada waktu yang berbeda, yaitu pukul 20.30 WIB. Estimasi waktu awal berdasarkan estimasi *google maps* yaitu 20 menit untuk sampai di lokasi tujuan. Tampilan aplikasi saat perjalanan seperti ditampilkan pada gambar 4.22.



Gambar 4. 22. Tampilan Saat Pengujian Rute 3 Kedua

Pada gambar 4.22 menampilkan tampilan aplikasi saat pengujian rute 3 yang kedua. Kondisi lalu lintas pada rute yang ditempuh lebih lancar dibanding dengan kondisi pada pengujian waktu pertama. Rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 3 yang kedua dapat dilihat pada gambar 4.23.



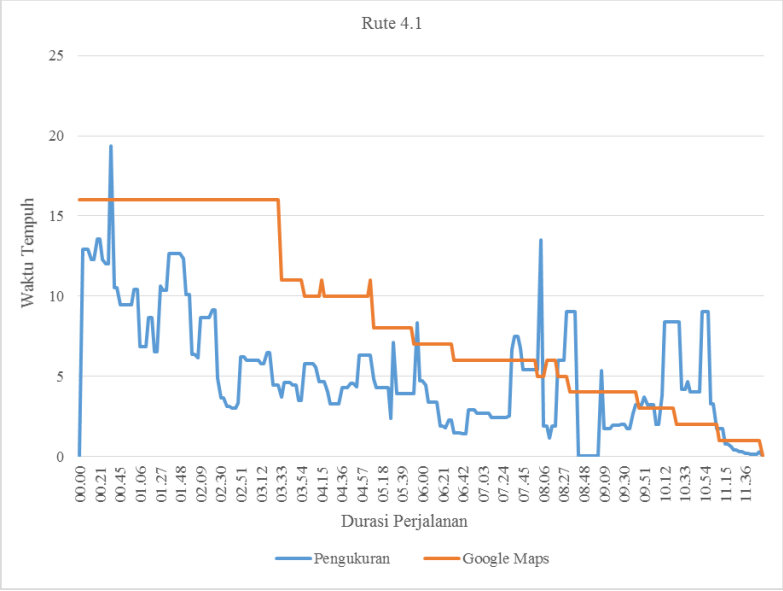
Gambar 4. 23. Grafik Pengukuran Waktu Tempuh Pada Pengujian Rute 3 Kedua

Dari grafik pengukuran waktu tempuh pada pengujian rute 3 yang kedua dapat dilihat terdapat fluktuasi pengukuran waktu tempuh. Fluktuasi tersebut dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui. Pada pengujian ini melalui beberapa *traffic light* yang terdapat kepadatan lalu lintas. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 16 menit 34 detik, yang berarti 3 menit lebih cepat dibanding dengan estimasi *google maps*. Dari hasil pengujian kedua rute 3 kendaraan berjalan dengan kecepatan rata-rata 27 km/h dengan kecepatan maksimal 64 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 5 menit lebih cepat bila dibandinganken dengan hasil estimasi *google maps*.

- Rute Pengujian 4

Pada pengujian rute 4 ini dilakukan 2 kali pengujian dari lokasi asal di Jl. Gubeng menuju Jl. Raya Darmo dengan jarak 4,3 km pada pukul 17.00 WIB. Pengujian pertama akan dilakukan dengan kecepatan diatas rata-rata kecepatan kendaraan lainnya dan yang terakhir dengan kecepatan seperti kendaraan disekitarnya. Hal ini ditujukan agar mengetahui kinerja sistem pada kondisi rute yang sama namun dengan kecepatan tempuh yang berbeda.

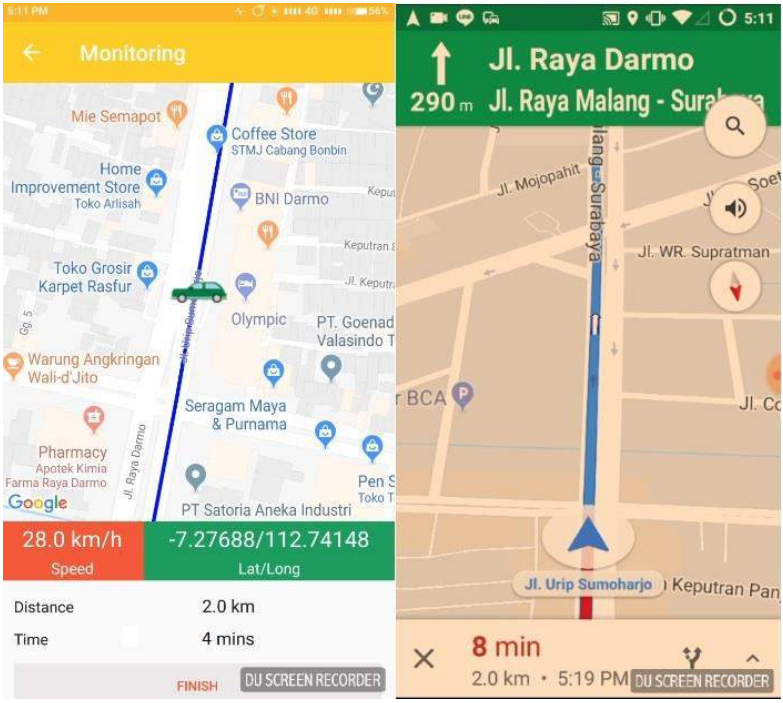
Pengujian pertama dilakukan dengan estimasi waktu awal berdasarkan estimasi *google maps* yaitu 16 menit untuk sampai di lokasi tujuan. Rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 1 yang pertama dapat dilihat pada gambar 4.24.



Gambar 4. 24. Grafik Pengukuran Waktu Tempuh pada Pengujian Rute 4 Pertama

Dari grafik pengukuran waktu tempuh pada pengujian rute 4 yang pertama dapat dilihat terdapat fluktuasi pengukuran waktu tempuh dikarenakan kondisi lalu lintas yang relatif padat. Terlihat

pada grafik bahwa meskipun hasil pengukuran berfluktuasi karena jalur yang dilewati padat, namun rata-rata hasil pengukuran masih lebih rendah dibanding estimasi *google maps.* Hal tersebut dikarenakan kendaraan berjalan di atas rata-rata kecepatan kendaraan lainnya yang juga melalui rute yang sama di saat bersamaan.

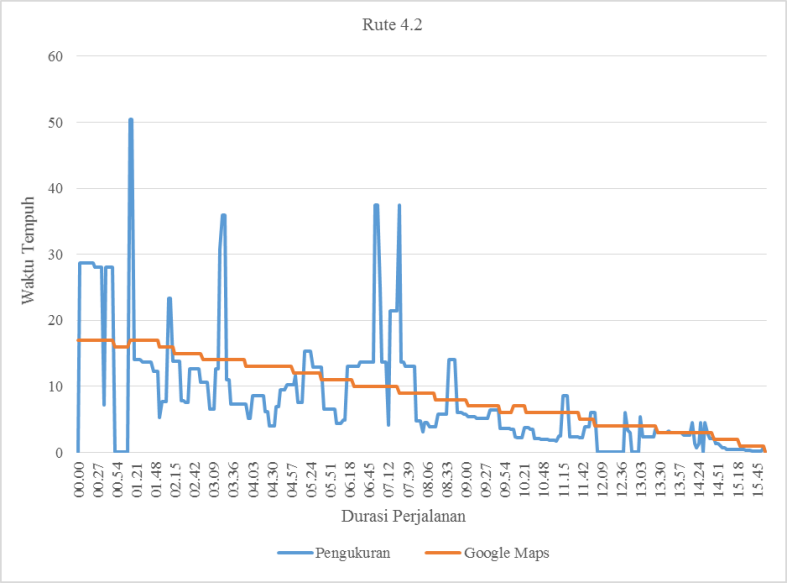


Gambar 4. 25. Tampilan Pengujian saat dibandingkan dengan *Google Maps*

Gambar 4.25 menampilkan kecepatan kendaraan 28 km/h. Kecepatan tersebut di atas kecepatan rata-rata pengendara lain yang melalui rute tersebut pada waktu bersamaan. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 11 menit 54 detik, yang berarti lebih cepat 4 menit dari estimasi *google maps*. Dari hasil pengujian pertama rute 4 kendaraan berjalan dengan kecepatan rata-rata 26 km/h dengan kecepatan maksimal 64 km/h.

Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 4 ini memiliki selisih rata-rata 4 menit lebih cepat bila dibandingankan dengan hasil estimasi *google maps*.

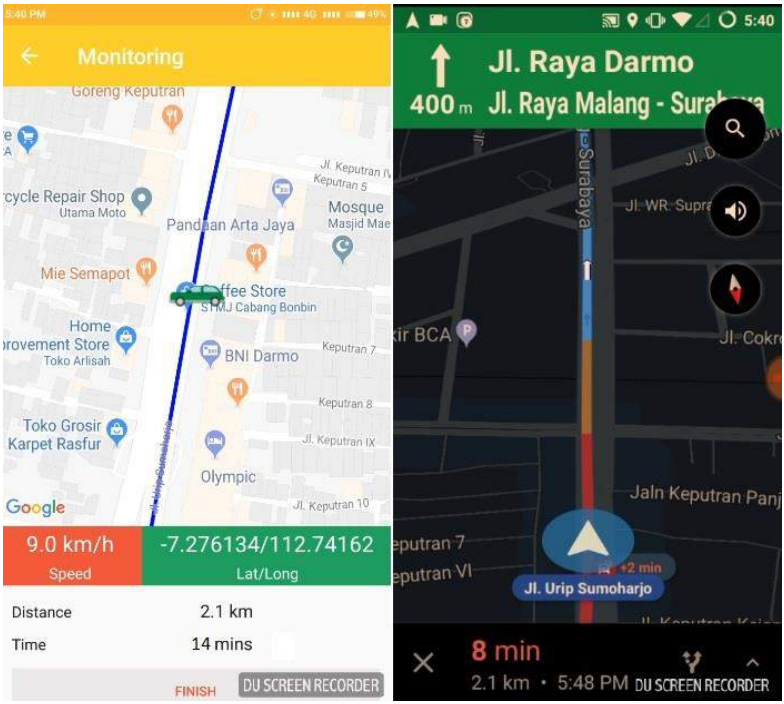
Kemudian pada pengujian kedua dilakukan pada rute yang sama namun dengan kecepatan mengikuti kecepatan rata-rata kendaraan di sekitar. Estimasi waktu awal berdasarkan estimasi *google maps* yaitu 17 menit untuk sampai di lokasi tujuan. Rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 1 yang kedua dapat dilihat pada gambar 4.26.



Gambar 4. 26. Grafik Pengukuran Waktu Tempuh pada Pengujian Rute 4 Kedua

Dari grafik pengukuran waktu tempuh pada pengujian rute 4 yang kedua dapat dilihat terdapat fluktuasi pengukuran waktu tempuh dikarenakan kondisi lalu lintas yang relatif padat. Terlihat pada grafik bahwa rata-rata hasil pengukuran lebih tinggi dibanding pengujian sebelumnya. Hal tersebut dikarenakan kendaraan berjalan di mengikuti rata-rata kecepatan kendaraan lainnya yang juga melalui rute yang sama di saat bersamaan atau

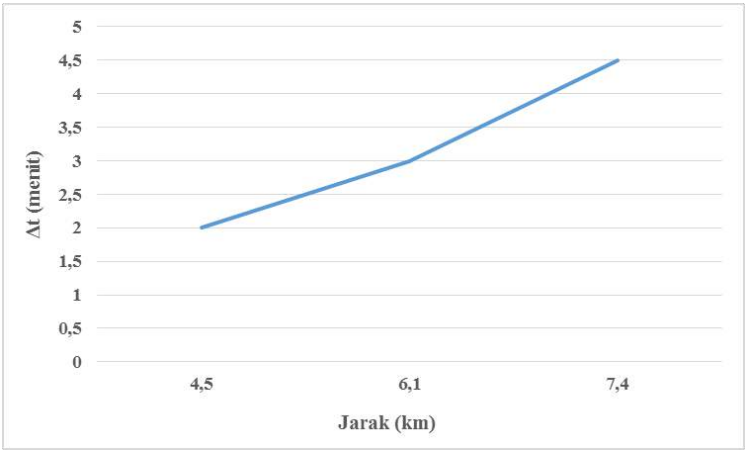
lebih rendah dari kecepatan kendaraan pada saat pengujian sebelumnya.



Gambar 4. 27. Tampilan Pengujian saat dibandingkan dengan *Google Maps*

Gambar 4.27 menampilkan kecepatan kendaraan 9 km/h. Kecepatan tersebut mengikuti kecepatan rata-rata pengendara lain yang melalui rute tersebut pada waktu bersamaan. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 15 menit 57 detik, yang berarti masih lebih cepat 1 menit dari estimasi *google maps*. Dari hasil pengujian kedua rute 4 kendaraan berjalan dengan kecepatan rata-rata 20 km/h dengan kecepatan maksimal 59 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 4 ini memiliki selisih rata-rata hanya 1 menit lebih cepat bila dibandingankan dengan hasil estimasi *google maps*.

Dari hasil pengujian pada rute 1 hingga 3, didapati data selisih waktu rata-rata hasil pengukuran dibandingkan dengan estimasi *google maps* seperti pada grafik di bawah ini.



Gambar 4. 28. Grafik Hubungan Jarak dan Selisih Waktu

Dari gambar 4.28 dapat dilihat bahwa semakin jauh jarak tempuh yang dilakukan saat pengujian, maka selisih waktu rata-rata hasil pengukuran dibandingkan dengan estimasi *google maps* semakin besar. Rata-rata waktu hasil pengukuran lebih rendah dibanding *google maps* karena pada pengujian ini yang digunakan adalah jenis kendaraan sepeda motor, sedangkan estimasi *google* adalah hasil dari berbagai jenis kendaraan baik sepeda motor, mobil maupun kendaraan lainnya. Hal tersebut menunjukkan pula bahwa alat ukur waktu tempuh ini akan memiliki kinerja lebih baik bila diaplikasikan pada kendaraan yang berjalan di atas kecepatan kendaraan umumnya, misalkan kendaraan pemadam kebakaran ataupun *ambulance.*

4.3 Analisa Data

Berdasarkan data pada pengujian yang telah dilakukan kemudian dilakukan analisa dari hasil pengujian tersebut. Analisa dilakukan terhadap kinerja masing-masing komponen penyusun sistem, kemudian dilanjutkan dengan analisa kinerja dari integrasi

sistem pengukuran dan *google maps* menjadi sistem informasi waktu tempuh yang tertampil pada aplikasi *handphone* berbasis android.

4.3.1 Analisa Kinerja Komponen Penyusun Sistem

Pada pengujian pertama sensor kecepatan yang bertujuan untuk mengetahui akurasi sensor, dilakukan pada kecepatan 5 km/h hingga 80 km/h dengan variasi tiap 5 km/h. Dari hasil pengujian menunjukkan kesalahan akurasi rata-rata sensor kecepatan sebesar 2,74%, sehingga akurasi dapat dikatakan sangat tinggi yaitu 97,26 %. Kemudian dilanjutkan dengan pengujian sensitifitas sensor. Pengujian dilakukan pada kecepatan 0 km/h hingga 80 km/h dengan variasi tiap 5 km/h. Dari grafik 4.1 hasil pengujian sensitifitas sensor kecepatan didapatkan fungsi transfer yang menginformasikan bahwa sensor kecepatan ini mengkonversi setiap perubahan frekuensi 1 Hz menjadi perubahan kecepatan sebesar 0,2009 km/h dengan kegelinciran 0,2009. Jadi sensitifitasnya adalah 0,2009 km/h tiap hertz frekuensinya. Kemudian pengujian sensor kecepatan yang ketiga bertujuan untuk mengetahui pembacaan sensor pengukuran kecepatan pada kendaraan saat kendaraan berjalan di lintasan sebenarnya, sehingga diketahui nilai presentase error dari sensor kecepatan yang dirancang. Pengujian sistem pengukuran kecepatan ini dilakukan dengan mencatat waktu yang dibutuhkan oleh kendaraan saat berjalan dengan kecepatan konstan dari titik awal ke titik akhir sejauh 500 meter, kemudian dibandingkan dengan hasil perhitungan teori. Pengujian sensor kecepatan ini dilakukan dengan variasi kecepatan kendaraan pada 20, 40 dan 60 km/h dengan masing-masing variasi 5 kali pengambilan data. Hasil pengujian sensor kecepatan pada saat kecepatan 20 km/h yang dapat dilihat pada tabel 4.1 menunjukkan kesalahan pembacaan rata-rata dari 5 kali pengambilan yaitu sebesar 3,55%. Hasil pengujian sensor kecepatan pada saat kecepatan 40 km/h yang dapat dilihat pada tabel 4.2 menunjukkan kesalahan pembacaan rata-rata dari 5 kali pengambilan yaitu sebesar 4,44%. Hasil pengujian sensor kecepatan pada saat kecepatan 40 km/h yang dapat dilihat pada tabel 4.3 menunjukkan kesalahan pembacaan

rata-rata dari 5 kali pengambilan yaitu sebesar 4,67%. Dari tiga variasi kecepatan yang digunakan pada pengujian ini, kemudian dihitung rata-rata error pengukuran dibandingkan dengan perhitungan teori, sehingga rata-rata errornya didapatkan sebesar 4,22%.

Pada pengujian sensor GPS yang dilakukan untuk mengetahui apakah sensor GPS telah dapat mengirim lokasi dengan tepat, dilakukan pengujian dengan berjalan pada lintasan lurus sejauh 500 meter, kemudian diambil sebanyak 20 data yang masuk beruntun pada web server. Data yang masuk tersebut kemudian divalidasi dengan *google maps* untuk mengetahui lokasinya pada tampilan peta. Adapun hasil pembacaan sensor GPS dapat dilihat pada tabel 4.6 dan hasil validasi melalui *google maps* ditunjukkan pada gambar 4.3. Dari gambar 4.3 dapat dilihat posisi kendaraan yang terbaca oleh sensor GPS. Setiap data yang terdapat pada tabel 4.6 ditunjukkan oleh pin berwarna merah pada gambar 4.3, data pertama hingga 20 ditunjukkan dari pin teratas hingga terbawah pada gambar 4.3. Data yang didapat telah sesuai dengan posisi kendaraan pada saat itu. Pada *datasheet* modul yang digunakan didapati data akurasi pembacaan posisi *latitude* dan *longitude* yang dilakukan memiliki akurasi secara horizontal 2,5 meter, akurasi kecepatan 0,1 meter / detik dan akurasi arah (*heading accuracy*) 0,5°. Akurasi ini dipengaruhi pula oleh kekuatan antena modul GPS tersebut, kondisi cuaca dan peletakan modul GPS tersebut.

Berdasarkan data yang didapat pada pengujian sistem pengiriman data yang memanfaatkan modul WiFi ESP 8266 pada mikrokontroller arduino dan menggunakan koneksi internet *smartfren* untuk mengirim data ke *web server thingspeak* dilakukan dengan pengambilan data sebanyak 20 pada saat kendaraan melakukan pengujian pada lintasan sejauh 500 meter. Setelah dilakukan pengujian didapatkan hasil pembacaan sistem pengiriman data ditunjukkan pada Tabel 4.5. Setelah dilakukan perhitungan didapatkan presentase keberhasilan pengiriman data rata-rata sebesar 83.33% dengan waktu pengiriman data selama 7 detik mulai dari data terbaca oleh sensor hingga diterima oleh *web server*. Terdapat kegagalan data terkirim disebabkan *traffic* dari

koneksi internet yang digunakan saat mikrokontroller arduino meminta pengiriman data atau *web server* sedang *timeout*, sehingga teradapat data tidak berhasil diterima oleh *web server*.

Pada pengujian masukan aplikasi android yang bertujuan untuk mengetahui presentase keberhasilan aplikasi android dalam menerima masukan data kecepatan, latitude, dan longitude dari web server, dilakukan pengiriman data *dummy* berupa angka 1 hingga 100 tiap 3 detik dari mikrokontroller ke *web server*, kemudian *web server* akan mengirimkan ke aplikasi android. Presentase keberhasilan dihitung dengan perbandingan data yang berhasil diterima dibanding data yang dikirim. Setelah dilakukan pengecekan data yang dikeluarkan oleh mikrokontroller, data yang masuk ke *web server* dan data yang masuk pada aplikasi android. Dilakukan pengecekan *history* masukan pada aplikasi android. Terdapat beberapa data yang dikirim dari mikrokontroller yang tidak diterima oleh aplikasi android, kemudian dilakukan pengecekan pada web server apakah data yang tidak masuk pada aplikasi android tersebut masuk pada web server atau tidak. Setelah dilakukan pengecekan pada web server didapati bahwa data yang tidak masuk pada android juga tidak masuk pada web server sehingga dapat disimpulkan data *loss* terjadi antara mikrokontroller dan *web server*, sedangkan seluruh data yang ada pada *web server* telah berhasil masuk pada aplikasi android, sehingga didapati presentase keberhasilan masukan data pada aplikasi android adalah 100%. Seluruh data yang ada pada *web server* telah dapat diterima pada aplikasi android dengan dilakukannya pengecekan pada *history* masukan data. Pada pengujian pengiriman data ini didapati *delay* dari pengiriman data dari mikrokontroller hingga diterima oleh aplikasi android selama 10 detik.

4.3.2 Analisa Kinerja Sistem Integrasi

Berdasarkan rekaman data pengukuran waktu tempuh selama pengujian rute 1 yang pertama pada gambar 4.7, dapat dilihat terdapat 2 kali kenaikan signifikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga harus mengurangi kecepatan. Pada gambar 4.8 dapat dilihat pada saat pengujian berjalan sekitar menit ke dua,

terjadi kenaikan hasil pengukuran waktu tempuh yang dikarenakan kendaraan berada pada kondisi lalu lintas yang cenderung padat sehingga kecepatan kendaraan menurun dan perhitungan waktu tempuh meningkat. Ketika dibandingkan dengan hasil tampilan google maps menunjukkan kendaraan memang berada pada kondisi lalu lintas yang cenderung padat yang ditampilkan dengan warna rute berwarna jingga. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 9 menit 16 detik. Dari hasil pengujian pertama rute 1 kendaraan berjalan dengan kecepatan rata-rata 32 km/h dengan kecepatan maksimal 56 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 2 menit lebih cepat bila dibandingankan dengan hasil estimasi google maps. Pada pengujian kedua dilakukan pada rute yang sama namun pada waktu yang berbeda, kondisi lalu lintas pada rute yang ditempuh relatif sama dengan pengujian waktu pertama. Dari gambar 4.10, dapat dilihat terdapat fluktuasi pengukuran waktu tempuh yang dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui. Terdapat satu kenaikan signifikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga kecepatan kendaraan jauh di bawah rata-rata dan perhitungan waktu tempuhnya meningkat. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 9 menit 1 detik. Dari hasil pengujian kedua rute 1 kendaraan berjalan dengan kecepatan rata-rata 34 km/h dengan kecepatan maksimal 62 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 2 menit lebih cepat bila dibandingankan dengan hasil estimasi google maps. Hasil pengujian rute 1 yang pertama dan kedua tidak memiliki perbedaan yang signifikan karena kondisi lalu lintas saat dilakukan kedua pengujian tersebut relatif sama.

Berdasarkan gambar 4.12, dapat dilihat terdapat pula fluktuasi pengukuran waktu tempuh yang dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui. Terdapat 2 kali kenaikan signifikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga

harus mengurangi kecepatan. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 12 menit 15 detik. Dari hasil pengujian pertama rute 2 kendaraan berjalan dengan kecepatan rata-rata 35 km/h dengan kecepatan maksimal 80 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 2 menit lebih cepat bila dibandinganken dengan hasil estimasi google maps. Pada pengujian kedua yang dilakukan pada rute yang sama, berdasarkan gambar 4.15 yang menampilkan rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 2 yang kedua dapat dilihat terdapat satu kenaikan signifikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga kecepatan kendaraan jauh di bawah rata-rata dan perhitungan waktu tempuhnya meningkat. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 12 menit 41 detik. Gambar 4.16 menunjukkan kondisi ketika terjadi kenaikan waktu tempuh perjalanan yang ternyata hasil tampilan google maps menunjukkan kendaraan memang berada pada kondisi lalu lintas yang cenderung padat yang ditampilkan dengan warna rute berwarna jingga. Namun tidak selalu ketika google maps menunjukkan kendaraan sedang berada pada rute berwarna jingga maupun merah akan meningkatkan hasil pengukuran waktu tempuh secara signifikan. Hal tersebut karena alat ukur waktu tempuh bekerja berdasarkan kecepatan kendaraan, walaupun kondisi lalu lintas yang dilalui padat, namun jika kendaraan dapat berjalan di tengah kepadatan dengan kecepatan di atas rata-rata kendaraan lain, maka hasil pengukuran tidak akan meningkat secara signifikan. Gambar 4.17 menampilkan ketika terjadi peningkatan hasil pengukuran waktu tempuh, hal tersebut dikarenakan kendaraan harus mengurangi kecepatan saat akan melewati persimpangan jalan. Peningkatan hasil pengukuran terjadi meskipun pada sisi sebelah kanan gambar yang menunjukkan google maps menunjukkan bahwa kendaraan tidak sedang berada pada rute yang padat menurut google maps. Gambar 4.18 bagian kanan menampilkan ketika kendaraan sedang berada pada lalu lintas yang cenderung padat yang ditunjukkan dengan

warna jingga pada jalur yang dilalui pada google maps. Namun pada hasil pengukuran waktu tempuh tidak terjadi peningkatan, hal tersebut dikarenakan kendaraan dapat berjalan di atas kecepatan rata-rata kendaraan lainnya saat melewati kondisi lalu lintes tersebut. Tidak terjadi peningkatan hasil pengukuran meskipun pada sisi sebelah kanan gambar yang menunjukkan google maps menunjukkan kendaraan sedang berada pada rute yang cenderung padat menurut google maps. Hal tersebut kembali menunjukkan bahwa alat ukur waktu tempuh ini lebih dapat menyesuaikan dengan karakteristik pengemudi saat berkendara. Apabila pengemudi ingin menurunkan kecepatan kendaraannya maka hasil pengukuran waktu tempuh akan meningkat, begitu pula sebaliknya. Dari hasil pengujian kedua rute 2 kendaraan berjalan dengan kecepatan rata-rata 34 km/h dengan kecepatan maksimal 64 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 4 menit lebih cepat bila dibandinganken dengan hasil estimasi google maps. Hasil pengujian rute 2 yang pertama dan kedua juga tidak memiliki perbedaan yang signifikan karena kondisi lalu lintas saat dilakukan kedua pengujian tersebut relatif sama.

Berdasarkan gambar 4.20, dapat dilihat lebih banyak terdapat fluktuasi pengukuran waktu tempuh yang dikarenakan kondisi lalu lintas yang dilalui jauh lebih padat dari 2 rute sebelumnya. Terdapat beberapa kali kenaikan dari hasil pengukuran waktu tempuh dikarenakan kendaraan sedang berada pada titik kepadatan lalu lintas sehingga harus mengurangi kecepatan. Gambar 4.21 menunjukkan ketika pengujian berjalan sekitar 3 menit 50 detik, terjadi kenaikan hasil pengukuran waktu tempuh, hal tersebut didapati karena kendaraan sedang berada pada kondisi lalu lintas yang cenderung padat sehingga kecepatan kendaraan menurun dan perhitungan waktu tempuh meningkat. Setelah dilakukan pengecekan pada google maps ternyata hasil tampilan google maps menunjukkan kendaraan memang berada pada kondisi lalu lintas yang padat yang ditampilkan dengan warna rute berwarna merah. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 18 menit 28 detik. Dari hasil pengujian pertama rute 3 kendaraan

berjalan dengan kecepatan rata-rata 22 km/h dengan kecepatan maksimal 62 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 3 ini memiliki selisih rata-rata 4 menit lebih cepat bila dibandingankan dengan hasil estimasi google maps. Kemudian pengujian kedua pada rute yang sama, pada gambar 4.23 ditampilkan grafik pengukuran waktu tempuh pada pengujian rute 3 yang kedua, dapat dilihat terdapat fluktuasi pengukuran waktu tempuh yang dikarenakan kondisi lalu lintas yang ada pada rute yang dilalui. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 16 menit 34 detik. Dari hasil pengujian kedua rute 3 kendaraan berjalan dengan kecepatan rata-rata 27 km/h dengan kecepatan maksimal 64 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 1 ini memiliki selisih rata-rata 5 menit lebih cepat bila dibandingankan dengan hasil estimasi google maps.

Pada pengujian rute 4 yang pertama, dilakukan pengujian dengan kecepatan diatas rata-rata kecepatan kendaraan lainnya. Berdasarkan rekaman data pengukuran waktu tempuh selama perjalanan pada pengujian rute 1 yang pertama, pada gambar 4.24 dapat dilihat pada grafik bahwa meskipun hasil pengukuran berfluktuasi karena jalur yang dilewati padat, namun rata-rata hasil pengukuran masih lebih rendah dibanding estimasi google maps. Hal tersebut dikarenakan kendaraan berjalan di atas rata-rata kecepatan kendaraan lainnya yang juga melalui rute yang sama di saat bersamaan. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 11 menit 54 detik. Dari hasil pengujian pertama rute 4 kendaraan berjalan dengan kecepatan rata-rata 26 km/h dengan kecepatan maksimal 64 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 4 ini memiliki selisih rata-rata 4 menit lebih cepat bila dibandingankan dengan hasil estimasi google maps. Pada pengujian rute 4 yang keuda, dilakukan pengujian dengan kecepatan mengikuti kecepatan rata-rata kendaraan di sekitar. Dapat dilihat pada gambar 4.26 bahwa rata-rata hasil pengukuran lebih tinggi dibanding pengujian sebelumnya. Hal tersebut dikarenakan kendaraan berjalan mengikuti rata-rata kecepatan kendaraan lainnya yang juga melalui

rute yang sama di saat bersamaan atau lebih rendah dari kecepatan kendaraan pada saat pengujian sebelumnya. Waktu yang dibutuhkan hingga sampai pada lokasi tujuan yaitu 15 menit 57 detik. Dari hasil pengujian kedua rute 4 kendaraan berjalan dengan kecepatan rata-rata 20 km/h dengan kecepatan maksimal 59 km/h. Hasil pengukuran waktu tempuh perjalanan pada pengujian pertama rute 4 ini memiliki selisih rata-rata hanya 1 menit lebih cepat bila dibandingankan dengan hasil estimasi google maps.

Pada pengujian integrasi sistem ini diiringi pula dengan rekaman data estimasi dari *google maps* saat dilakukan pengujian. Setelah dibandingkan dengan estimasi yang dilakukan oleh *google maps* terdapat perbedaan dari pengukuran waktu tempuh dan estimasi dari *google maps*, hal ini dikarenakan karena sistem ini melakukan pengukuran langsung berdasarkan kecepatan kendaraan pada speedometer. Dengan dilakukannya pengukuran langsung ini akan didapati nilai pengukuran waktu tempuh yang berbeda-beda tiap kendaraan, hal ini menjadi perbedaan dibandingkan sistem estimasi *google maps* karena setiap pengemudi memiliki karakteristik dan cara mengemudi masing-masing, begitu pula dengan perbedaan jenis kendaraan yang dikemudikan tentunya akan mengakibatkan waktu tempuh yang berbeda pula yang dapat diakomodasi pula oleh sistem pengukuran langsung berbasis pengukuran speedometer ini. Pada sistem estimasi google maps akan menemui perbedaan antara estimasi awal dengan hasil waktu tempuh sebenarnya. Selain dikarenakan setiap pengemudi memiliki karakteristik mengemudi masing-masing dan perbedaan jenis kendaraan, perubahan kondisi jalan dari estimasi awal juga mempengaruhi perbedaan itu. Waktu tempuh sebenarnya dapat menjadi lebih cepat dikarenakan pengemudi berjalan dengan kecepatan yang diatas kecepatan rata-rata pengemudi lainnya. Waktu tempuh sebenarnya juga dapat menjadi lebih lambat dari estimasi awal *google maps* dikarenakan apabila pengemudi terjebak kemacetan dan berjalan lebih lambat dari kecepatan rata-rata hasil estimasi google maps maka waktu estimasi *google maps* tidak dapat langsung memperbarui estimasinya.

Dari hasil pengujian pada rute 1 hingga 3, didapati data selisih waktu rata-rata hasil pengukuran dibandingkan dengan estimasi *google maps* seperti pada grafik pada gambar 4.27 dapat dilihat bahwa semakin jauh jarak tempuh yang dilakukan saat pengujian, maka selisih waktu rata-rata hasil pengukuran dibandingkan dengan estimasi *google maps* semakin besar. Rata-rata waktu hasil pengukuran lebih rendah dibanding *google maps* karena pada pengujian ini yang digunakan adalah jenis kendaraan sepeda motor, sedangkan estimasi *google* adalah hasil dari berbagai jenis kendaraan baik sepeda motor, mobil maupun kendaraan lainnya. Hal tersebut menunjukkan pula bahwa alat ukur waktu tempuh ini akan memiliki kinerja lebih baik bila diaplikasikan pada kendaraan yang berjalan di atas kecepatan kendaraan umumnya, misalkan kendaraan pemadam kebakaran ataupun *ambulance.*Pada pengujian sistem perhitungan waktu tempuh ini terjadi *delay* data yang terbaca sensor kecepatan hingga tertampil di aplikasi android selama 10 detik. Ini dikarenakan delay dari pengiriman data dari mikrokontroller ke *web server* selama 7 detik dan *delay request* dari android untuk meminta data ke *web server* selama 3 detik.

BAB V
PENUTUP

5.1 Kesimpulan

Setelah dilakukan penelitian dapat diambil kesimpulan sebagai berikut:

1. Sistem pengukuran kecepatan kendaraan dari *speedometer* yang dapat diolah oleh *handphone* berbasis android telah berhasil direalisasikan menggunakan sensor *hall effect* yang terdapat pada sistem *speedometer* sepeda motor dengan mikrokontroller arduino sebagai pemroses sinyal, modul GPS U-blox 6M sebagai sensor posisi dan modul ESP 8266 sebagai penghubung ke *web server* sebagai database untuk menyajikan data menuju aplikasi yang terdapat pada *handphone* berbasis android .
2. Akurasi sensor kecepatan sebesar 97,26 % dengan sensitifitas 0,2009 km/h tiap hertz frekuensinya. Nilai presentase *error* rata-rata sistem pengukuran kecepatan dengan sensor *hall effect* sebagai penghitung frekuensi putaran roda sepeda motor pada penelitian ini sebesar 4.22%.
3. Presentase rata – rata keberhasilan sistem pengiriman data memanfaatkan modul WiFi ESP 8266 pada mikrokontroller arduino dan menggunakan koneksi internet *smartfren* untuk mengirim data ke *web server thingspeak* yang digunakan pada penelitian sebesar 83,33% dengan waktu pengiriman data selama 7 detik. Adanya kegagalan data terkirim disebabkan *traffic* dari koneksi internet yang digunakan saat mikrokontroller arduino meminta pengiriman data.
4. Alat ukur waktu tempuh perjalanan berdasarkan kecepatan kendaraan pada *speedometer* ini telah berhasil melakukan pengukuran dengan baik berdasarkan perubahan kecepatan yang terukur pada sensor kecepatan kendaraan sehingga waktu tempuh perjalanan dapat berubah secara *real-time*.
5. Selisih waktu rata-rata hasil pengukuran dibandingkan dengan estimasi *google maps* seperti pada penelitian ini berada pada kisaran 3 menit. Semakin jauh jarak tempuh yang dilakukan

saat pengujian, maka selisih waktu rata-rata hasil pengukuran dibandingkan dengan estimasi *google maps* semakin besar.

6. Rata-rata waktu hasil pengukuran lebih rendah dibanding *google maps* karena pada pengujian ini yang digunakan adalah jenis kendaraan sepeda motor, sedangkan estimasi *google* adalah hasil dari berbagai jenis kendaraan baik sepeda motor, mobil maupun kendaraan lainnya. Hal tersebut menunjukkan pula bahwa alat ukur waktu tempuh ini akan memiliki kinerja lebih baik bila diaplikasikan pada kendaraan yang berjalan di atas kecepatan kendaraan umumnya, misalkan kendaraan pemadam kebakaran ataupun *ambulance.*
7. Pada pengujian alat ukur waktu tempuh ini terdapat *delay* data yang terbaca sensor kecepatan hingga tertampil di aplikasi android selama 10 detik. Ini dikarenakan *delay* dari pengiriman data dari mikrokontroller ke *web server* selama 7 detik dan delay request dari android untuk meminta data ke web server selama 3 detik.

5.2 Saran

Adapun saran yang dapat diberikan setelah dilakukan penelitian ini adalah sebagai berikut:

1. Dapat digunakan koneksi internet yang lebih cepat untuk meminimalisir *delay* pengiriman data.
2. Protokol *web server* MQTT dapat digunakan pada pengembangan penelitian untuk meminimalisir waktu *request* data dari android ke *web server*.
3. Sistem ini dapat diaplikasikan pada transportasi dengan urgensi ketepatan waktu untuk mencapai lokasi tujuan seperti kendaraan pemadam kebakaran, *ambulance*, maupun jasa *delivery*.

DAFTAR PUSTAKA

Ardana, Y. M. (2014). *Pemrograman Android Black Box.* Surabaya: Jurusan Teknik Informatika - STIKOM.

Assidqi, N. R., & Soehartanto, T. (2014). Rancang Bangun Sistem Monitoring Kecepatan Kendaraan Bermotor Berbasis Android.

Bar-Shalom, Y. (2001). *Estimation with Applications to Tracking.*

Hibban, H. (2014). Sistem Pengukuran dan Monitoring Kecepatan Gerak Kendaraan Bermotor.

Nova, A. T. (2011). Pengembangan Sistem Peringatan Ganti Oli Pada Sepeda Motor. *Jurnal Teknik Lektro Vol.3 No.1*.

Poerwanto. (2013). *Instrumentasi & Alat Ukur.* Jakarta: Graha Ilmu.

Sudhakar, K. N. (2013). Predicting the Bus Arrival Time Using GPS and GSM Tecnology. *International Journal of Science and Research (IJSR)*, ISSN: 2319-7064.

Halaman ini memang dikosongkan.

LAMPIRAN A

LISTING PROGRAM ARDUINO UNO

```
String apiKey = "24Y0F1RQTDVCOCLY";
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

SoftwareSerial serial_gps(6, 7);      // RX = pin 3, TX = pin 4
SoftwareSerial esp8266(8, 9);
TinyGPSPlus gps;
float latittude, longitude;
const int pulsePin = 5;               // Input signal connected to Pin 5 of Arduino

int pulseHigh;                        // Integer variable to capture High time of the incoming
pulse
int pulseLow;                         // Integer variable to capture Low time of the incoming pulse
float pulseTotal;                     // Float variable to capture Total time of the incoming pulse
float frequency;                      // Calculated Frequency
int kecepatan;
int lastkecepatan;
int margin;

void setup() {
  Serial.begin(19200);
  pinMode(pulsePin,INPUT);
  Serial.println("START IN 5 sec");
  delay(5000);
}

void loop() {
  serial_gps.begin(9600);
  for (unsigned long start = millis(); millis() - start < 1000;) {
    while(serial_gps.available()) {
      gps.encode(serial_gps.read());
    }
  }
  pulseHigh = pulseIn(pulsePin,HIGH);
  pulseLow = pulseIn(pulsePin,LOW);
  pulseTotal = pulseHigh + pulseLow;          // Time period of the pulse in microseconds
  frequency=1000000/pulseTotal;               // Frequency in Hertz (Hz)

  if (pulseTotal == 0)
    { frequency = 0;}

  kecepatan = 2*22/7*frequency*0.22*3.6;

  if (kecepatan < 0)
  { kecepatan = lastkecepatan;}
  if (kecepatan > 85)
  { kecepatan = lastkecepatan;}
```

```
  margin=kecepatan-lastkecepatan;
  if (margin >= 30)
  { kecepatan = lastkecepatan;}

  latittude = gps.location.lat();
  longitude = gps.location.lng();
  Serial.print(latittude, 6);
  Serial.print(",");
  Serial.print(longitude, 6);
  Serial.print(",");
  Serial.print(frequency);                    //menampilkan data dari serial
  Serial.print(" Hz,");
  Serial.print(kecepatan);                    //menampilkan data dari serial
  Serial.print(" km/h");
  Serial.println();

  kirim();

  lastkecepatan = kecepatan;
}

void kirim(){
  //ESP KIRIM
  esp8266.begin(115200);

  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com
  cmd += "\",80";
  esp8266.println(cmd);
  Serial.println(cmd);
  delay(200);

  if(esp8266.find("OK")){
    delay(300);

    String getStr = "GET https://api.thingspeak.com/update?api_key=";
    getStr += apiKey;
    getStr +="&field1=";
    getStr += String(kecepatan);
    getStr +="&field2=";
    getStr += String(latittude,6);
    getStr +="&field3=";
    getStr += String(longitude,6);
    getStr += "\r\n\r\n";

    cmd = "AT+CIPSEND=";
    cmd += String(getStr.length()*2);
    esp8266.println(cmd);
    Serial.println(cmd);

    delay(250);
    if(esp8266.find(">")){
      esp8266.print(getStr);
```

```
    delay(250);
    esp8266.print(getStr);
    Serial.println(getStr);
    delay(250);
    esp8266.println("AT+CIPCLOSE");
    Serial.println("AT+CIPCLOSE");
   }
   else{
    esp8266.println("AT+CIPCLOSE");
    Serial.println("AT+CIPCLOSE");

    String cmd = "AT+CIPSTART=\"TCP\",\"";
    cmd += "184.106.153.149"; // api.thingspeak.com
    cmd += "\",80";
    esp8266.println(cmd);
    Serial.println(cmd);
    delay(200);

    String getStr = "GET https://api.thingspeak.com/update?api_key=";
    getStr += apiKey;
    getStr +="&field1=";
    getStr += String(kecepatan);
    getStr +="&field2=";
    getStr += String(latittude,6);
    getStr +="&field3=";
    getStr += String(longitude,6);
    getStr += "\r\n\r\n";

    cmd = "AT+CIPSEND=";
    cmd += String(getStr.length()*2);
    esp8266.println(cmd);
    Serial.println(cmd);
    delay(250);
    esp8266.print(getStr);
    esp8266.print(getStr);
    Serial.println(getStr);
    delay(250);
    esp8266.println("AT+CIPCLOSE");
    Serial.println("AT+CIPCLOSE");
   }
  }
  else {
   delay(200);
   esp8266.println("AT+CIPCLOSE");
   Serial.println("AT+CIPCLOSE");
  }
}
```

Halaman ini memang dikosongkan.

LAMPIRAN B

LISTING PROGRAM ANDROID STUDIO

Splash Screen :

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_splash);

    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            startActivity(new Intent(SplashActivity.this, MainActivity.class));
            finish();
        }
    }, 3500);

}
```

Activity :

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="me.isdzulqor.smartmover.smartmover.SplashActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical">
        <ImageView
            android:layout_width="160dp"
            android:layout_height="40dp"
            app:srcCompat="@drawable/ic_logo2"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAlignment="center"
            android:textSize="14sp"
            android:layout_marginRight="6dp"
```

```
        android:layout_marginLeft="6dp"
        android:textColor="@color/colorPrimaryText"
        android:text="Your Smart Partner to move around the City"/>
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

Main Page :

```
package me.isdzulqor.smartmover.smartmover;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.WindowManager;
import android.view.inputmethod.EditorInfo;
import android.widget.AdapterView;
import android.widget.AutoCompleteTextView;
import android.widget.TextView;
import android.widget.Toast;

import com.anthonycr.grant.PermissionsManager;
import com.anthonycr.grant.PermissionsResultAction;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.PendingResult;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.places.AutocompletePrediction;
import com.google.android.gms.location.places.Place;
import com.google.android.gms.location.places.PlaceBuffer;
import com.google.android.gms.location.places.Places;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;
```

```
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import me.isdzulqor.smartmover.smartmover.adapter.CustomInfoWindowAdapter;
import me.isdzulqor.smartmover.smartmover.adapter.PlaceAutocompleteAdapter;
import me.isdzulqor.smartmover.smartmover.model.PlaceInfo;

public class MainActivity extends AppCompatActivity implements
OnMapReadyCallback, GoogleApiClient.OnConnectionFailedListener {
    @BindView(R.id.atv_search)
    AutoCompleteTextView atvSearch;
    @OnClick(R.id.bt_depart) void btDepartClick(){
        startActivity(new Intent(MainActivity.this, DepartActivity.class));
    }

    private static final LatLngBounds LAT_LNG_BOUNDS = new LatLngBounds(
            new LatLng(-40, -168), new LatLng(71, 136));
    private static final float DEFAULT_ZOOM = 15f;

    //map
    private SupportMapFragment mapFragment;
    //

    //vars
    private Boolean mLocationPermissionsGranted = false;
    private GoogleMap mMap;
    private FusedLocationProviderClient mFusedLocationProviderClient;
    private PlaceAutocompleteAdapter mPlaceAutocompleteAdapter;
    private GoogleApiClient mGoogleApiClient;
    private PlaceInfo mPlace;
    private Marker mMarker;

    private static final String TAG = SplashActivity.class.getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
        init();
    }

    private void init(){

        //map
        mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        //

        atvSearch.setOnItemClickListener(mAutocompleteClickListener);

        mGoogleApiClient = new GoogleApiClient
                .Builder(this)
                .addApi(Places.GEO_DATA_API)
                .addApi(Places.PLACE_DETECTION_API)
                .enableAutoManage(this, this)
                .build();

        mPlaceAutocompleteAdapter = new PlaceAutocompleteAdapter(this,
mGoogleApiClient,
                LAT_LNG_BOUNDS, null);

        atvSearch.setAdapter(mPlaceAutocompleteAdapter);

        atvSearch.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int actionId, KeyEvent
keyEvent) {
                if(actionId == EditorInfo.IME_ACTION_SEARCH
                        || actionId == EditorInfo.IME_ACTION_DONE
                        || keyEvent.getAction() == KeyEvent.ACTION_DOWN
                        || keyEvent.getAction() == KeyEvent.KEYCODE_ENTER){

                    //execute our method for searching
                    geoLocate();
                }

                return false;
            }
        });
    }

    @SuppressLint("MissingPermission")
```

```
    @Override
    public void onMapReady(final GoogleMap googleMap) {
        mMap = googleMap;

        PermissionsManager.getInstance().requestPermissionsIfNecessaryForResult(this,
                new String[]{
                        Manifest.permission.ACCESS_FINE_LOCATION },
                new PermissionsResultAction() {
                    @Override
                    public void onGranted() {
                        mLocationPermissionsGranted = true;

                        mMap.setMyLocationEnabled(true);
                        mMap.getUiSettings().setMyLocationButtonEnabled(true);
                        Location current = mMap.getMyLocation();
//                        Toast.makeText(MainActivity.this, current.getLatitude()+"
"+current.getLongitude(), Toast.LENGTH_SHORT).show();
//                        LatLng latLng = new LatLng(current.getLatitude(),
current.getLongitude());
//                        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng,
DEFAULT_ZOOM));
                        getDeviceLocation();
                    }


                    @Override
                    public void onDenied(String permission) {
                        Log.e(TAG, "Unable to get location without permission");
                    }
                });
    }
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {

        PermissionsManager.getInstance().notifyPermissionsChange(permissions,
grantResults);
    }

    /*
      ---------------------------- google places API autocomplete suggestions -----------------
    */

    private AdapterView.OnItemClickListener mAutocompleteClickListener = new
AdapterView.OnItemClickListener() {
```

```
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
      hideSoftKeyboard();

      final AutocompletePrediction item = mPlaceAutocompleteAdapter.getItem(i);
      final String placeId = item.getPlaceId();

      PendingResult<PlaceBuffer> placeResult = Places.GeoDataApi
          .getPlaceById(mGoogleApiClient, placeId);
      placeResult.setResultCallback(mUpdatePlaceDetailsCallback);
    }
  };

  private ResultCallback<PlaceBuffer> mUpdatePlaceDetailsCallback = new
ResultCallback<PlaceBuffer>() {
    @Override
    public void onResult(@NonNull PlaceBuffer places) {
      if(!places.getStatus().isSuccess()){
        Log.d(TAG, "onResult: Place query did not complete successfully: " +
places.getStatus().toString());
        places.release();
        return;
      }
      final Place place = places.get(0);

      try{
        mPlace = new PlaceInfo();
        mPlace.setName(place.getName().toString());
        Log.d(TAG, "onResult: name: " + place.getName());
        mPlace.setAddress(place.getAddress().toString());
        Log.d(TAG, "onResult: address: " + place.getAddress());
//          mPlace.setAttributions(place.getAttributions().toString());
//          Log.d(TAG, "onResult: attributions: " + place.getAttributions());
        mPlace.setId(place.getId());
        Log.d(TAG, "onResult: id:" + place.getId());
        mPlace.setLatlng(place.getLatLng());
        Log.d(TAG, "onResult: latlng: " + place.getLatLng());
        mPlace.setRating(place.getRating());
        Log.d(TAG, "onResult: rating: " + place.getRating());
        mPlace.setPhoneNumber(place.getPhoneNumber().toString());
        Log.d(TAG, "onResult: phone number: " + place.getPhoneNumber());
        mPlace.setWebsiteUri(place.getWebsiteUri());
        Log.d(TAG, "onResult: website uri: " + place.getWebsiteUri());

        Log.d(TAG, "onResult: place: " + mPlace.toString());
      }catch (NullPointerException e){
        Log.e(TAG, "onResult: NullPointerException: " + e.getMessage() );
```

```
            }

            moveCamera(new LatLng(place.getViewport().getCenter().latitude,
                    place.getViewport().getCenter().longitude), DEFAULT_ZOOM, mPlace);

            places.release();
        }
    };

    private void moveCamera(LatLng latLng, float zoom, PlaceInfo placeInfo){
        Log.d(TAG, "moveCamera: moving the camera to: lat: " + latLng.latitude + ",
lng: " + latLng.longitude );
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, zoom));

        mMap.clear();

        mMap.setInfoWindowAdapter(new
CustomInfoWindowAdapter(MainActivity.this));

        if(placeInfo != null){
            try{
                String snippet = "Address: " + placeInfo.getAddress() + "\n" +
                        "Phone Number: " + placeInfo.getPhoneNumber() + "\n" +
                        "Website: " + placeInfo.getWebsiteUri() + "\n" +
                        "Price Rating: " + placeInfo.getRating() + "\n";

                MarkerOptions options = new MarkerOptions()
                        .position(latLng)
                        .title(placeInfo.getName())
                        .snippet(snippet);
                mMarker = mMap.addMarker(options);

            }catch (NullPointerException e){
                Log.e(TAG, "moveCamera: NullPointerException: " + e.getMessage() );
            }
        }else{
            mMap.addMarker(new MarkerOptions().position(latLng));
        }

        hideSoftKeyboard();
    }

    private void moveCamera(LatLng latLng, float zoom, String title){
        Log.d(TAG, "moveCamera: moving the camera to: lat: " + latLng.latitude + ",
lng: " + latLng.longitude );
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, zoom));
```

```
        if(!title.equals("My Location")){
            MarkerOptions options = new MarkerOptions()
                    .position(latLng)
                    .title(title);
            mMap.addMarker(options);
        }

        hideSoftKeyboard();
    }

    private void hideSoftKeyboard(){

this.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_ST
ATE_ALWAYS_HIDDEN);
    }

    private void geoLocate(){
        Log.d(TAG, "geoLocate: geolocating");

        String searchString = atvSearch.getText().toString();

        Geocoder geocoder = new Geocoder(MainActivity.this);
        List<Address> list = new ArrayList<>();
        try{
            list = geocoder.getFromLocationName(searchString, 1);
        }catch (IOException e){
            Log.e(TAG, "geoLocate: IOException: " + e.getMessage() );
        }

        if(list.size() > 0){
            Address address = list.get(0);

            Log.d(TAG, "geoLocate: found a location: " + address.toString());
            //Toast.makeText(this, address.toString(), Toast.LENGTH_SHORT).show();

            moveCamera(new LatLng(address.getLatitude(), address.getLongitude()),
DEFAULT_ZOOM,
                    address.getAddressLine(0));
        }
    }

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

    }
```

```
    private void getDeviceLocation(){
        Log.d(TAG, "getDeviceLocation: getting the devices current location");
        Toast.makeText(MainActivity.this, "function getDeviceLocation()",
Toast.LENGTH_SHORT).show();

        mFusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);

        try{
            if(mLocationPermissionsGranted){

                final Task location = mFusedLocationProviderClient.getLastLocation();
                location.addOnCompleteListener(new OnCompleteListener() {
                    @Override
                    public void onComplete(@NonNull Task task) {
                        if(task.isSuccessful()){
                            Log.d(TAG, "onComplete: found location!");
                            Location currentLocation = (Location) task.getResult();

                            moveCamera(new LatLng(currentLocation.getLatitude(),
currentLocation.getLongitude()),
                                    DEFAULT_ZOOM,
                                    "My Location");
                            Toast.makeText(MainActivity.this, "current location",
Toast.LENGTH_SHORT).show();

                        }else{
                            Log.d(TAG, "onComplete: current location is null");
                            Toast.makeText(MainActivity.this, "unable to get current location",
Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            }
        }catch (SecurityException e){
            Log.d(TAG, "getDeviceLocation: getting the devices current location");
            Log.e(TAG, "getDeviceLocation: SecurityException: " + e.getMessage() );
        }
    }
}
```

Activity :

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="me.isdzulqor.smartmover.smartmover.MainActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <include layout="@layout/content_main" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:background="@color/colorPrimaryDark"
            android:gravity="center"
            android:padding="4dp"
            android:text="Teknik Fisika - 2018 | ITS"
            android:textColor="@color/colorWhite"
            android:textSize="14sp"
            android:textStyle="italic" />
    </RelativeLayout>
</android.support.design.widget.CoordinatorLayout>

<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.NestedScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="me.isdzulqor.smartmover.smartmover.MainActivity"
    tools:showIn="@layout/activity_main">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:gravity="center"
        android:orientation="vertical">

        <ImageView
            android:layout_marginTop="16dp"
```

```
    android:layout_width="160dp"
    android:layout_height="40dp"
    app:srcCompat="@drawable/ic_logo2" />

  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="6dp"
    android:layout_marginRight="6dp"
    android:text="Your Smart Partner to move around the City"
    android:layout_marginBottom="24dp"
    android:textAlignment="center"
    android:textColor="@color/colorPrimaryText"
    android:textSize="14sp" />

  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="400dp">

    <fragment
      android:id="@+id/map"
      android:name="com.google.android.gms.maps.SupportMapFragment"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_gravity="center" />

  </RelativeLayout>

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="12dp"
    android:layout_marginRight="12dp"
    android:layout_marginBottom="6dp"
    android:orientation="horizontal">
    <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_weight="7"
      android:textColor="@color/colorSuccess"
      android:textStyle="bold"
      android:textSize="12sp"
      android:text="Destination"/>
    <AutoCompleteTextView
      android:id="@+id/atv_search"
      android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_weight="3"
            android:textColor="@color/colorSecondaryText"
            android:hint="Input Destination..."
            android:textSize="14sp"
            android:imeOptions="actionSearch"/>
        </LinearLayout>
        <Button
            android:id="@+id/bt_depart"
            style="@style/customButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="12dp"
            android:layout_marginRight="12dp"
            android:layout_marginBottom="38dp"
            android:minHeight="0dp"
            android:paddingTop="7dp"
            android:paddingBottom="7dp"
            android:textSize="12sp"
            android:background="@drawable/ripple_warning"
            android:text="DEPART"/>
    </LinearLayout>
</android.support.v4.widget.NestedScrollView>
```

Mengambil Data Dari *Thingspeak* :

```
compile 'com.squareup.retrofit2:retrofit:2.3.0'
compile 'com.squareup.retrofit2:converter-gson:2.3.0'
public class ApiService {
    public static final String THINGSPEAK_API_KEY= "24Y0F1RQTDVCOCLY";
    public static final String API_MAP_URL =
"https://maps.googleapis.com/maps/api/";
    public static final String API_THINGSPEAK =
"https://api.thingspeak.com/channels/";

    public interface ApiMap {
        @GET("distancematrix/json")
        Call<MatrixDistanceModel> getMatrixMap(
                @Query("origins") String origin,
                @Query("destinations") String destination,
                @Query("key") String key
        );
        @GET("directions/json")
        Call<DirectionModel> getDirectionMap(
                @Query("origin") String origin,
                @Query("destination") String destination,
                @Query("key") String key
        );
    }
```

```
    public interface ApiThingSpeak {
        @GET("499668/feeds.json")
        Call<ResponseThinkspeak> getSensors(
                @Query("api_key") String apiKey,
                @Query("results") String result
        );
    }
}

private void getSensorsRequest() {
    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(ApiService.API_THINGSPEAK)
            .addConverterFactory(GsonConverterFactory.create())
            .build();
    ApiService.ApiThingSpeak service = retrofit
            .create(ApiService.ApiThingSpeak.class);
    Call<ResponseThinkspeak> listCall = service.getSensors(
            ApiService.THINGSPEAK_API_KEY,
            "1"
    );
    listCall.enqueue(new Callback<ResponseThinkspeak>() {
        @Override
        public void onResponse(Call<ResponseThinkspeak> call,
Response<ResponseThinkspeak> response) {
            final ResponseThinkspeak responseThinkspeak = response.body();
            if (responseThinkspeak != null) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        // save to local
                        // set latitude/longitude and kecepatan
                        // presave local
                        Feed feedTemp = responseThinkspeak.getFeeds().get(0);

                        try {

feedTemp.setCreatedAt(Constanta.convertDate(feedTemp.getCreatedAt()));
                        } catch (ParseException e) {
                            e.printStackTrace();
                        }
                        FeedDao feedDao = new FeedDao(feedTemp.getEntryId(),
feedTemp.getCreatedAt(),
                                Double.valueOf(feedTemp.getField1()),
Double.valueOf(feedTemp.getField2()),
                                Double.valueOf(feedTemp.getField3()), UNIQUE_ID);
                        saveFeed(feedDao);
                    }
                });
                Log.e(TAG, "responseThinkspeak : " + responseThinkspeak.toString());
```

```
            } else {
                Log.e(TAG, "responseThinkspeak : null bos");
            }
        }

        @Override
        public void onFailure(Call<ResponseThinkspeak> call, Throwable t) {
            Log.e(TAG, "getSensorRequest error : " + t.getLocalizedMessage());
        }
    });
}
```

Algoritma Kalkulasi :

```
private void calculateEstimation() {
        double tempDistance, tempEstimTime;
        String tempEstimTimeString;
        tempDistance = SPEED / 3600;
        DISTANCE = DISTANCE - tempDistance;
        tvDistance.setText(new DecimalFormat("##.#").format(DISTANCE) + " km");
        tempEstimTime = DISTANCE / SPEED;
        tempEstimTimeString = Constanta.timeConversion((int) tempEstimTime);
        if (tempEstimTimeString.contains("0 h 0 m")) {
            tempEstimTimeString.replace("0 h 0 m", "");
        } else if (tempEstimTimeString.contains("0 h")) {
            tempEstimTimeString.replace("0 h", "");
        }
        tvEstimTime.setText(tempEstimTimeString);
    }
```

Polylines di Jalur yang Dilewati :

```
####
if (polylineList != null)
    removeAllPolylines();
polylineList = new ArrayList<Polyline>();
for (int i = 0; i < steps.size(); i++) {
    List<LatLng> points =
Constanta.decodePolyLine(steps.get(i).getPolyline().getPoints());
    drawPolylines(points);
}
####
private void drawPolylines(List<LatLng> latLngList) {
    PolylineOptions polylineOptions = new PolylineOptions().
            geodesic(true).
            color(Color.BLUE).
            width(10);

    for (int i = 0; i < latLngList.size(); i++) {
        polylineOptions.add(latLngList.get(i));
    }
    Polyline line = mMap.addPolyline(polylineOptions);
    polylineList.add(line);
}
```

Save History :

```
classpath "io.realm:realm-gradle-plugin:5.1.0"
public class FeedDao extends RealmObject {
    @PrimaryKey
    private Integer entryId;
    private String createdAt;
    private double field1;
    private double field2;
    private double field3;
    private long uniqueId;

    public FeedDao() {
    }

    public FeedDao(Integer entryId, String createdAt, double field1, double field2, double
field3, long uniqueId) {
        this.entryId = entryId;
        this.createdAt = createdAt;
        this.field1 = field1;
        this.field2 = field2;
        this.field3 = field3;
        this.uniqueId = uniqueId;
    }
    public long getUniqueId() {
        return uniqueId;
    }
    public void setUniqueId(long uniqueId) {
        this.uniqueId = uniqueId;
    }
    public String getCreatedAt() {
        return createdAt;
    }
    public void setCreatedAt(String createdAt) {
        this.createdAt = createdAt;
    }
    public Integer getEntryId() {
        return entryId;
    }
    public void setEntryId(Integer entryId) {
        this.entryId = entryId;
    }
    public double getField1() {
        return field1;
    }
    public void setField1(double field1) {
        this.field1 = field1;
    }
    public double getField2() {
        return field2;
```

```
    }
    public void setField2(double field2) {
        this.field2 = field2;
    }
    public double getField3() {
        return field3;
    }
    public void setField3(double field3) {
        this.field3 = field3;
    }
    @Override
    public String toString() {
        return "Feed{" +
                "createdAt='" + createdAt + '\'' +
                ", entryId=" + entryId +
                ", field1='" + field1 + '\'' +
                ", field2='" + field2 + '\'' +
                ", field3='" + field3 + '\'' +
                '}';
    }
}
```

Class Model diatas merujuk pada response API dari ThingSpeak

Method untuk menyimpan ke database realm

```
private void saveFeed(final FeedDao feedDao) {
    realm.executeTransaction(new Realm.Transaction() {
        @Override
        public void execute(Realm realm) {
            realm.insertOrUpdate(feedDao);
            updateUI(feedDao);
        }
    });
}
private List<HistoryDao> getHistories(){
    List<HistoryDao> data = realm.where(HistoryDao.class).findAll();
    return data;
}
```