**

**RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER**

OTNIEL YEHEZKIEL BORNOK HUTABARAT

NRP 5112100212

Dosen Pembimbing

Fajar Baskoro, S.Kom., MT.

Rizky Januar Akbar, S.Kom, M.Eng

JURUSAN TEKNIK INFORMATIKA

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya 2016

TUGAS AKHIR – K141502

*[Halaman ini sengaja dikosongkan]*



**RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER**

OTNIEL YEHEZKIEL BORNOK HUTABARAT

NRP 5112100212

Dosen Pembimbing

Fajar Baskoro, S.Kom, MT.

Rizky Januar Akbar, S.Kom, M.Eng

JURUSAN TEKNIK INFORMATIKA

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya 2016

TUGAS AKHIR – K141502

*[Halaman ini sengaja dikosongkan]*



*[Halaman ini sengaja dikosongkan]*

***DEVELOPING A BUMP DETECTION SYSTEM USING ANDROID SMARTPHONE WITH ACCELEROMETER***

OTNIEL YEHEZKIEL BORNOK HUTABARAT

NRP 5112100212

Dosen Pembimbing

Fajar Baskoro, S.Kom, MT.

Rizky Januar Akbar, S.Kom, M.Eng

JURUSAN TEKNIK INFORMATIKA

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya 2016

FINAL PROJECT – K141502

# LEMBAR PENGESAHAN

**RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER**

**TUGAS AKHIR**

Diajukan Guna Memenuhi Salah Satu Syarat   
Memperoleh Gelar Sarjana Komputer  
pada   
Bidang Studi Algoritma dan Pemrograman

Program Studi S-1 Jurusan Teknik Informatika  
Fakultas Teknologi Informasi  
Institut Teknologi Sepuluh Nopember

Oleh :  
**OTNIEL YEHEZKIEL BORNOK HUTABARAT  
NRP : 5112 100 212**

Disetujui oleh Dosen Pembimbing Tugas Akhir :

|  |  |
| --- | --- |
| Fajar Baskoro, S.Kom., M.T.  NIP: 19740403 199903 1 002 | ................................ (pembimbing 1) |
|  |  |
| Rizky Januar Akbar, S.Kom., M.Eng.  NIP: 19870103 201404 1 001 | ................................ (pembimbing 2) |
|  |  |

**Surabaya  
JUNI 2016**

*[Halaman ini sengaja dikosongkan]*

**RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER**

Nama Mahasiswa : Otniel Yehezkiel Bornok Hutabarat

NRP : 5112 100 212

Jurusan : Teknik Informatika FTIf-ITS

Dosen Pembimbing 1 : Fajar Baskoro, S.Kom, MT.

Dosen Pembimbing 2 : Rizky Januar Akbar, S.Kom, M.Eng

# ABSTRAK

*Seiring semakin meningkatnya jumlah polisi tidur yang ilegal atau tanpa izin, maka hal ini dapat menyebabkan ketidaknyamanan oleh pengguna jalan. Bahkan pemerintah telah membuat sanksi yang ditulis dalam Peraturan Daerah terhadap pembuat pita penggaduh atau polisi tidur yang tidak memiliki izin dari Kepala Dinas Perhubungan.*

*Untuk mengatasi masalah tersebut, pada tugas akhir ini dibangun sebuah sistem pendeteksi bump yang terdiri dari aplikasi Android dan peta digital. Pada aplikasi Android, sistem mengumpulkan data dengan mendeteksi guncangan apabila pengguna melewati polisi tidur, lalu mengirim lokasi tersebut ke server. Deteksi guncangan ini memanfaatkan sensor akselerometer pada Android dan pengiriman lokasi menggunakan GPS. Kemudian pada server data diolah dan ditampilkan dalam bentuk peta digital.*

*Setelah melakukan pengujian, data yang dihasilkan adalah lokasi guncangan (disebabkan oleh polisi tidur atau jalan rusak) dan ditampilkan dalam bentuk peta digital.*

*Dengan adanya sistem ini, diharapkan informasi lokasi guncangan (polisi tidur maupun kerusakan jalan yang terdeteksi) dapat diperoleh secepat mungkin sehingga pemerintah dapat bertindak dengan lebih cepat dan baik.*

***Kata kunci***: ***Deteksi bump, Sensor Akselerometer, Perangkat Bergerak, Monitor jalan.***

***DEVELOPING A SPEED BUMP MAPPING SYSTEM USING ANDROID SMARTPHONE WITH ACCELEROMETER***

Student Name : Otniel Yehezkiel Bornok Hutabarat

Student ID : 5112 100 212

Major : Informatics Department FTIf-ITS

Advisor 1 : Fajar Baskoro, S.Kom, MT.

Advisor 2 : Rizky Januar Akbar, S.Kom, M.Eng

# ABSTRACT

*As an increasing number of bumps that are illegal or unlicensed, then this can cause discomfort to road users. Even the government has made a written sanction local regulation against the makers of rumble strip or bumps that do not have the permission of the Head of the Department of Transportation.*

*To overcome these problems, this final project constructed a mapping system of speed limiting devices consisting of Android apps and digital maps. On Android, the system collects data by detecting vibration which caused when the user passes through bumps, and then send it to the server location. This detection utilizes the accelerometer sensor on Android and delivery location using GPS. Then the server data is processed and displayed in the form of digital maps.*

*After examination, the data produced is the location of the bump (caused by a bump or damaged roads) and displayed in the form of digital maps.*

*With this system, the expected location information shocks (bumps or road damage is detected) can be obtained as soon as possible so that the government* can act more quickly and better.

***Keywords***: ***Bump Detection, Accelerometer sensor, Mobile Sensing, Road Monitoring.***

*[Halaman ini sengaja dikosongkan]*

# KATA PENGANTAR

Segala puji syukur bagi Tuhan Yesus Kristus, yang oleh karena anugrahNya penulis dapat menyelesaikan tugas akhir yang berjudul “RANCANG BANGUN SISTEM PENDETEKSI BUMP MENGGUNAKAN ANDROID SMARTPHONE DENGAN AKSELEROMETER”.

Pengerjaan tugas akhir ini menjadi suatu pengalaman yang baik bagi penulis. Penulis dapat memperoleh banyak pengalaman yang berharga dalam memperdalam dan meningkatkan keilmuan dalam bidang informatika selama perkuliahan di Teknik Informatika ITS.

Tugas akhir ini selesai karena tidak lepas dari bantuan dan dukungan berbagai pihak. Pada kesempatan kali ini penulis ingin mengucapkan terima kasih kepada:

1. Tuhan Yesus. Atas segala kebaikan dan berkatNya bagi penulis.
2. Mama dan Adik penulis, Kezia, Ruth dan Caleb yang memberikan doa, dukungan dan semangat untuk penulis
3. Bapak Fajar Baskoro selaku dosen pembimbing 1 yang telah membantu dan membimbing penulis dalam menyelesaikan tugas akhir ini.
4. Bapak Rizky Januar Akbar selaku dosen pembimbing 2 yang telah membantu dan membimbing penulis dalam menyelesaikan tugas akhir ini.
5. Anak-anak grup *Iluminati* yang selalu menjadi penghibur dan teman nongkrong saat jenuh mengerjakan tugas akhir.
6. Temen-teman admin lab yang membantu dan menemani selama pembuatan tugas akhir ini.
7. Teman-teman angkatan 2012 yang menjadi semangat bagi penulis selama masa perkuliahan di Teknik Informatika ITS.
8. Romasta Hutagaol, yang tidak bosan-bosannya memberi semangat kepada penulis untuk pengerjaan tugas akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juni 2016

Otniel Yehezkiel Bornok Hutabarat

# DAFTAR ISI

[LEMBAR PENGESAHAN vii](#_Toc454535808)

[ABSTRAK ix](#_Toc454535809)

[ABSTRACT xi](#_Toc454535810)

[KATA PENGANTAR xii](#_Toc454535811)

[DAFTAR ISI xiv](#_Toc454535812)

[DAFTAR GAMBAR xviii](#_Toc454535813)

[DAFTAR TABEL xxi](#_Toc454535814)

[DAFTAR KODE SUMBER xxiii](#_Toc454535815)

[BAB I PENDAHULUAN 1](#_Toc454535816)

[1.1. Latar Belakang 1](#_Toc454535817)

[1.2. Tujuan 4](#_Toc454535818)

[1.3. Rumusan Permasalahan 4](#_Toc454535819)

[1.4. Batasan Permasalahan 4](#_Toc454535820)

[1.5. Metodologi 4](#_Toc454535821)

[1.6. Sistematika Penulisan 6](#_Toc454535822)

[BAB II DASAR TEORI 9](#_Toc454535823)

[2.1. Jalan 9](#_Toc454535824)

[2.1.1. Pengelompokan Jalan 9](#_Toc454535825)

[2.1.2. Tipe Jalan Berdasarkan Perkerasan Jalan 11](#_Toc454535826)

[2.1.3. Jenis Kerusakan Jalan 11](#_Toc454535827)

[2.2. Alat Pembatas Kecepatan 11](#_Toc454535828)

[2.3. Android 14](#_Toc454535829)

[2.3.1. Komponen Aplikasi pada Android 14](#_Toc454535830)

[2.3.2. Sensor pada Android 17](#_Toc454535831)

[2.4. PostgreSQL 18](#_Toc454535832)

[2.5. GPS (Global Positioning System) 19](#_Toc454535833)

[2.6. PHP (Hypertext Prepocessor) 21](#_Toc454535834)

[2.7. Python 21](#_Toc454535835)

[2.8. Google Map API 22](#_Toc454535836)

[2.9. CodeIgniter 23](#_Toc454535837)

[2.10. Slim 23](#_Toc454535838)

[2.11. Decision Tree 24](#_Toc454535839)

[2.12. Algoritma *Z-Thresh* 24](#_Toc454535840)

[2.13. BIRCH 25](#_Toc454535841)

[BAB III ANALISIS DAN PERANCANGAN 27](#_Toc454535842)

[3.1. Deskripsi Umum Sistem 27](#_Toc454535843)

[3.1.1. Analisis Masalah 28](#_Toc454535844)

[3.1.2. Analisis Kebutuhan Sistem 28](#_Toc454535845)

[3.2. Perancangan Sistem 35](#_Toc454535846)

[3.2.1. Perancangan Basis Data 35](#_Toc454535847)

[3.2.2. Perancangan *Web Service* 39](#_Toc454535848)

[3.2.3. Perancangan Antarmuka 41](#_Toc454535849)

[3.2.4. Perancangan Proses Data 45](#_Toc454535850)

[BAB IV IMPLEMENTASI 53](#_Toc454535851)

[4.1. Lingkungan Implementasi 53](#_Toc454535852)

[4.1.1. Lingkungan Implementasi Perangkat Keras 53](#_Toc454535853)

[4.1.2. Lingkungan Implementasi Perangkat Lunak 53](#_Toc454535854)

[4.2. Lingkungan Implementasi Antarmuka 54](#_Toc454535855)

[4.2.1. Antarmuka Peta Digital pada *Server* 54](#_Toc454535856)

[4.2.2. Antarmuka Peta pada Android 55](#_Toc454535857)

[4.2.3. Antarmuka Pengumpulan Data pada Android 56](#_Toc454535858)

[4.2.4. Antarmuka Verifikasi Data 58](#_Toc454535859)

[4.3. Implementasi Basis Data 59](#_Toc454535860)

[4.3.1. Implementasi Tabel *Location* 59](#_Toc454535861)

[4.3.2. Implementasi Tabel *Accelerometer* 59](#_Toc454535862)

[4.3.3. Implementasi Tabel Jenis 60](#_Toc454535863)

[4.3.4. Implementasi Tabel *Users* 61](#_Toc454535864)

[4.4. Implementasi *Web* *Service* 61](#_Toc454535865)

[4.4.1. Implementasi *Query User* 62](#_Toc454535866)

[4.4.2. Implementasi *Query* *Event Bump* 62](#_Toc454535867)

[4.4.3. Implementasi Koneksi Basis Data 64](#_Toc454535868)

[4.5. Implementasi Proses Aplikasi 64](#_Toc454535869)

[4.5.1. Implementasi Inisialisasi Sensor 64](#_Toc454535870)

[4.5.2. Implementasi Pengiriman Data 65](#_Toc454535871)

[4.5.3. Implementasi Mengambil Data Sensor 65](#_Toc454535872)

[4.5.4. Implementasi Reorientasi Akselerometer 66](#_Toc454535873)

[4.5.5. Implementasi Deteksi *Event* *Bump* 67](#_Toc454535874)

[4.5.6. Implementasi *Decision tree* 67](#_Toc454535875)

[4.5.7. Implementasi *Clustering* pada Peta Digital 68](#_Toc454535876)

[BAB V PENGUJIAN DAN EVALUASI 71](#_Toc454535877)

[5.1. Lingkungan Uji Coba 71](#_Toc454535878)

[5.2. Dasar Pengujian 71](#_Toc454535879)

[5.3. Uji Coba Fungsionalitas 72](#_Toc454535880)

[5.3.1. Uji Coba Fungsionalitas Melihat Peta Digital 73](#_Toc454535881)

[5.3.2. Uji Coba Fungsionalitas Mengumpulkan Data 75](#_Toc454535882)

[5.3.3. Uji Coba Fungsionalitas Verifikasi Data 76](#_Toc454535883)

[5.4. Analisa Data 77](#_Toc454535884)

[5.4.1. Analisa Data pada Kondisi Normal 78](#_Toc454535885)

[5.4.2. Analisa Data pada Kondisi Jalan Rata 79](#_Toc454535886)

[5.4.3. Analisa Data pada Polisi Tidur 79](#_Toc454535887)

[5.5. Pengujian Akurasi 80](#_Toc454535888)

[5.6. Evaluasi Pengujian 84](#_Toc454535889)

[5.6.1. Evaluasi Pengujian Fungsionalitas 84](#_Toc454535890)

[5.6.2. Evaluasi Pengujian Akurasi 85](#_Toc454535891)

[BAB VI PENUTUP 87](#_Toc454535892)

[6.1. Kesimpulan 87](#_Toc454535893)

[6.2. Saran 87](#_Toc454535894)

[DAFTAR PUSTAKA 89](#_Toc454535895)

[BIODATA PENULIS 93](#_Toc454535896)

# DAFTAR GAMBAR

[Gambar 2.1 Desain Standar Polisi Tidur 13](#_Toc454535528)

[Gambar 2.2 Sistem Koordinat (Relatif Terhadap Perangkat) 18](#_Toc454535529)

[Gambar 2.3 Ide Dasar GPS *Positioning* 20](#_Toc454535530)

[Gambar 2.4 Diagram Alir Algoritma Z-Thresh 25](#_Toc454535531)

[Gambar 2.5 Visualisasi Birch Clustering 26](#_Toc454535532)

[Gambar 3.1 Alur Umum Sistem 28](#_Toc454535533)

[Gambar 3.2 Diagram Kasus Penggunaan 29](#_Toc454535534)

[Gambar 3.3 Diagram Aktivitas Kasus Penggunaan UC-01 31](#_Toc454535535)

[Gambar 3.4 Diagram Alir Kasus Penggunaan UC-01 31](#_Toc454535536)

[Gambar 3.5 Diagram Aktivitas Kasus Penggunaan UC-02 32](#_Toc454535537)

[Gambar 3.6 Diagram Alir Kasus Penggunaan UC-02 33](#_Toc454535538)

[Gambar 3.7 Diagram Aktivitas Kasus Penggunaan UC-03 34](#_Toc454535539)

[Gambar 3.8 Diagram Alir Kasus Kegunaan UC-03 34](#_Toc454535540)

[Gambar 3.9 CDM (*Conceptual Data Model)* pada *web service* 36](#_Toc454535541)

[Gambar 3.10 PDM (*Physical Data Model)* pada *web service* 36](#_Toc454535542)

[Gambar 3.11 Rancangan Antarmuka pada Peta Digital 42](#_Toc454535543)

[Gambar 3.12 Rancangan Antarmuka Peta pada Android 42](#_Toc454535544)

[Gambar 3.13 Antarmuka pada Aplikasi 43](#_Toc454535545)

[Gambar 3.14 Rancangan Antarmuka Verifikasi Data 44](#_Toc454535546)

[Gambar 3.15 Rancangan Halaman Login Verifikasi 45](#_Toc454535547)

[Gambar 3.16 Diagram Alir Proses Data 46](#_Toc454535548)

[Gambar 3.17 Sistem Koordinat Global 47](#_Toc454535549)

[Gambar 3.18 Sistem Koordinat Kendaraan 48](#_Toc454535550)

[Gambar 3.19 Contoh Hasil *Clustering* pada Peta Digital 51](#_Toc454535551)

[Gambar 4.1 Tampilan Peta Digital Hasil Deteksi *Bump* 55](#_Toc454535552)

[Gambar 4.2 Tampilan Halaman List *Bump* 55](#_Toc454535553)

[Gambar 4.3 Tampilan Peta pada Android 56](#_Toc454535554)

[Gambar 4.4 Tampilan Aplikasi Utama 57](#_Toc454535555)

[Gambar 4.5 Tampilan Verifikasi Data 58](#_Toc454535556)

[Gambar 4.6 Halaman Verifikasi Login 58](#_Toc454535557)

[Gambar 5.1 Tampilan Lokasi *Bump* pada Peta Digital 74](#_Toc454535558)

[Gambar 5.2 Tampilan *List Bump* 74](#_Toc454535559)

[Gambar 5.3 Tampilan Aplikasi Android 76](#_Toc454535560)

[Gambar 5.4 Tampilan Halaman Verifikasi Data 77](#_Toc454535561)

[Gambar 5.5 Grafik Data Akselerometer Saat Tidak Bergerak 78](#_Toc454535562)

[Gambar 5.6 Grafik Data Akselerometer pada Jalan Rata 79](#_Toc454535563)

[Gambar 5.7 Grafik Data Akselerometer pada Polisi Tidur 80](#_Toc454535564)

[Gambar 5.8 Lokasi Polisi Tidur untuk Pengujian 82](#_Toc454535565)

[Gambar 5.9 Hasil Pengujian P-10 pada Peta Digital 84](#_Toc454535566)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR TABEL

[Tabel 3.1 Deskripsi Kasus Penggunaan 30](#_Toc454535567)

[Tabel 3.2 Rincian Alur Kasus Penggunaan UC-01 30](#_Toc454535568)

[Tabel 3.3 Rincian Alur Kasus Penggunaan UC-02 32](#_Toc454535569)

[Tabel 3.4 Rincian Alur Kasus Penggunaan UC-03 33](#_Toc454535570)

[Tabel 3.5 Atribut Tabel Data Location 37](#_Toc454535571)

[Tabel 3.6 Atribut Tabel Data Accelerometer 38](#_Toc454535572)

[Tabel 3.7 Atribut Tabel Jenis 38](#_Toc454535573)

[Tabel 3.8 Atribut Tabel Users 39](#_Toc454535574)

[Tabel 3.9 Parameter *Request* Menerima *Event Bump* 40](#_Toc454535575)

[Tabel 3.10 Parameter *Response* Menerima *Event Bump* 40](#_Toc454535576)

[Tabel 3.11 Parameter *Request* Menerima Id *User* 41](#_Toc454535577)

[Tabel 3.12 Parameter *Response* Menerima Id *User* 41](#_Toc454535578)

[Tabel 4.1 Lingkungan Implementasi Perangkat Keras 53](#_Toc454535579)

[Tabel 4.2 Lingkungan Implementasi Perangkat Lunak 54](#_Toc454535580)

[Tabel 5.1 Tabel Skenario Uji Coba Melihat Peta Digital 73](#_Toc454535581)

[Tabel 5.2 Tabel Skenario Uji Coba Mengumpulkan Data 75](#_Toc454535582)

[Tabel 5.3 Tabel Skenario Uji Coba Verifikasi Data 77](#_Toc454535583)

[Tabel 5.4 Tabel Skenario Pengujian 82](#_Toc454535584)

[Tabel 5.5 Hasil Uji Coba 83](#_Toc454535585)

[Tabel 5.6 Rangkuman Hasil Pengujian 84](#_Toc454535586)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR KODE SUMBER

[Kode Sumber 4.1 Implementasi Tabel *Location* 59](#_Toc454535587)

[Kode Sumber 4.2 Implementasi Tabel *Accelerometer* 60](#_Toc454535588)

[Kode Sumber 4.3 Implementasi Tabel Jenis 61](#_Toc454535589)

[Kode Sumber 4.4 Implementasi Tabel *Users* 61](#_Toc454535590)

[Kode Sumber 4.5 Implementasi *Query User* 62](#_Toc454535591)

[Kode Sumber 4.6 Implementasi *Query* *Event* *Bump* 63](#_Toc454535592)

[Kode Sumber 4.7 *Pseudocode* Koneksi Basis Data 64](#_Toc454535593)

[Kode Sumber 4.8 Implementasi Inisialisasi Sensor 64](#_Toc454535594)

[Kode Sumber 4.9 Implementasi Fungsi *addToJSONArray* 65](#_Toc454535595)

[Kode Sumber 4.10 Implementasi Fungsi *postEventData* 65](#_Toc454535596)

[Kode Sumber 4.11 Implementasi Mengambil Data Sensor 66](#_Toc454535597)

[Kode Sumber 4.12 Implementasi Reorientasi Akslerometer 67](#_Toc454535598)

[Kode Sumber 4.13 Implementasi Deteksi *Event Bump* 67](#_Toc454535599)

[Kode Sumber 4.14 Implementasi Klasifikasi *Decision tree* 68](#_Toc454535600)

[Kode Sumber 4.15 Implementasi *Clustering* pada Peta Digital 69](#_Toc454535601)

*[Halaman ini sengaja dikosongkan]*

# BAB I PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

## Latar Belakang

Alat pembatas kecepatan atau disebut juga sebagai polisi tidur adalah bagian jalan yang ditinggikan berupa tambahan aspal atau semen yang dipasang melintang di jalan untuk pertanda meperlambat laju/kecepatan kendaraan. Tujuan utama dari polisi tidur sendiri adalah untuk mencegah terjadinya kecelakaan pada saat aktivitas transportasi. Polisi tidur banyak ditemukan di daerah pemukiman penduduk, perumahan, terminal atau pasar. Namun pada kenyataannya banyak sekali polisi tidur yang dibuat tidak sesuai dengan desain polisi tidur yang diatur berdasarkan Keputusan Menteri Perhubungan No. 3 Tahun 1994 sehingga dapat membahayakan keamanan dan kenyamanan pengguna jalan.

Salah satu pendekatan untuk mengidentifikasi polisi tidur yang memiliki izin adalah adanya laporan dari pihak pembuat polisi tidur dengan pihak yang berwenang. Untuk setiap daerah memiliki peraturan masing-masing untuk menindaklanjuti Keputusan Menteri Perhubungan tersebut. Misalnya untuk daerah jakarta diatur oleh Peraturan Daerah Provinsi DKI yaitu berdasarkan pasal 53 huruf b Perda DKI Jakarta 12/2003, setiap orang tanpa izin dari Kepala Dinas Perhubungan dilarang membuat atau memasang tanggul pengaman jalan dan pita penggaduh (*speed* *trap*). Pelanggaran terhadap ketentuan tersebut adalah kurungan paling lama 3 bulan atau denda sebanyak-banyaknya Rp5.000.000,00 (lima juta rupiah). Oleh karena itu, untuk mempermudah pemerintah mengetahui polisi tidur yang memiliki izin atau ilegal, pada tugas akhir ini saya menawarkan sistem pendeteksi *bump* menggunakan smartphone android dengan pendekatan survei otomatis.

Berdasarkan penelitian sebelumnya yang telah dilakukan adalah Pothole Patrol (P2) [1], menggunakan sensor 3-*axis* *accelerometer* dan GPS diletakkan pada kendaraan taksi untuk memonitor keadaan permukaan jalan. Pothole Patrol dapat mendeteksi lubang dan anomali jalan dengan cara mengolah data dari getaran atau *vibrasi* pada akselerometer dan GPS. Proses pengolahan datanya menggunakan filter pengolahan sinyal untuk menolak *event* selain lubang (lubang got, celah jalan, rel kereta api, *expansion* *joints*). P2 juga menggunakan metode *blacklist* pada lokasi *speedbump* agar tidak terdeteksi sebagai lubang dengan menggunakan peta *speedbump* dari arsip data geografis di daerah penelitian P2. Penelitian ini juga mengklasifikasi data berdasarkan lokasi untuk mengurangi kesalahan dalam. Pada penelitian selanjutnya yaitu CRSM [2] atau disebut juga *crowdsourcing-based road surface monitoring system* mampu mendeteksi lubang dan mengevaluasi tingkat kekasaran permukaan jalan. Namun penelitian ini mengalami kesulitan dalam membedakan *event* *decelerating belts* atau polisi tidur dengan lubang karena memiliki getaran yang mirip. Sehingga CRSM menggunakan informasi geografis keadaan jalan untuk mengeliminasi tipe *event* seperti lubang got dan polisi tidur berdasarkan informasi GPS.

Penelitian yang menggunakan sensor *smartphone* adalah Wolverine [3]. Wolverine menggunakan sensor akselerometer untuk mengumpulkan data untuk mendeteksi *event* bump dan rem. Perangkatnya direorientasi terlebih dahulu sehingga dapat digunakan pada posisi apapun di dalam kendaraan. Proses reorientasinya menggunakan sensor akselerometer dan magnometer. Data akselerometer yang dikumpulkan kemudian diolah menggunakan *Support Vector Machine* (SVM) untuk menentukan kondisi jalan mulus atau bergelombang dan kondisi rem atau tidak. Salah satu penelitian yang mendeteksi lubang secara *real time* adalah Mednis et al., [4] mengusulkan sistem yang menggunakan *smartphone* Android beserta sensor akselerometernya untuk mendeteksi *events* secara *real* *time*. Sistem ini mengumpulkan data secara *off-line post-processing* dengan menggunakan algoritma *Z-Thresh*, *Z-Diff, STDEV(Z)* dan *G-Zero*.

Pendekatan survei otomatis yang diusulkan pada tugas akhir ini dapat dilakukan dengan menggunakan *embedded* *sensing* *devices* atau *smartphone*. Pada tugas akhir ini akan berfokus pada proses data akselerometer untuk mendeteksi polisi tidur menggunakan *smartphone* Android Metode untuk mendeteksi *bump* menggunakan kombinasi metode *real-time* dan *nonreal-time*. Metode *real-time* menggunakan algoritma *Z-Thresh* seperti pada penelitian pada Mednis et al namun perbedaannya adalah pada tugas akhir ini juga menggunakan metode klasifikasi *decision tree* pada data yang terdeteksi menggunakan algoritma *Z-Thresh* di *server* untuk menghasilkan deteksi yang lebih akurat.

Data GPS lokasi polisi tidur akan dikirim dari *smartphone* Android ke *server*. Aplikasi ini diharapkan akan digunakan oleh banyak pengguna jalan raya untuk mempercepat proses pemetaan polisi tidur pada sistem. Lalu pada *server* akan menandai lokasi polisi tidur dari data-data lokasi GPS yang dikirimkan dari *smartphone* pengguna.

Data-data GPS lokasi polisi tidur yang dikirim dari para pengguna diolah untuk mengestimasi lokasi polisi tidur dengan lebih akurat dan ditampilkan dalam bentuk peta digital. Dengan sistem yang *crowdsourced* (menggunakan kumpulan data partisipasi yang terdistribusi) dapat meningkatkan skalabilitas melihat banyaknya jumlah pengguna *smartphone* dan terus meningkat. Harapannya adalah informasi tersebut dapat diperoleh sedini mungkin sehingga dapat digunakan pemerintah untuk mempermudah pengawasan pembuatan polisi tidur maupun perbaikan dan pemeliharaan jalan.

## Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat sistem monitoring yang memetakan lokasi alat pembatas kecepatan dan anomali (kerusakan) jalan.

## Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini antara lain:

1. Bagaimana mendeteksi alat pembatas kecepatan menggunakan *smartphone* Android?
2. Bagaimana menampilkan lokasi alat pembatas kecepatan dalam bentuk peta digital?
3. Bagaimana mengolah estimasi lokasi alat pembatas kecepatan?

## Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, antara lain:

1. Data yang dikirimkan dari Android ke *server* adalah data akselerometer dan lokasi jalan.
2. Smartphone yang digunakan memiliki sensor akselerometer dan Android.
3. Uji coba menggunakan kendaraan sepeda motor.
4. Diujikan pada kendaraan dengan estimasi kecepatan 20-40 km/jam.
5. Aplikasi membutuhkan koneksi internet dan tersambung GPS.
6. Aplikasi yang dibangun adalah aplikasi android dan peta digital.

## Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini yaitu:

1. Studi literatur

Tahap ini merukapan proses pengumpulan informasi yang dibutuhkan untuk mengerjakan tugas akhir. Informasi yang dibutuhkan dapat berupa literatur dan dokumentasi penggunaan. Studi literatur yang akan dipelajari untuk tugas akhir ini adalah sebagai berikut:

1. Mekanisme sensor akselerometer pada *smartphone* Android
2. Pengambilan data training alat pembatas jalan dan penentuannya menggunakan algoritma *Z-Thresh* dan *Decision tree*.
3. Mekanisme pengiriman data lokasi dan data akselerasi dari Android
4. Menampilkan alat pembatas kecepatan pada peta digital
5. Analisis dan Desain Perangkat Lunak

Tahap ini dilakukan analisis terhadap sistem serta perancangan sistem yang akan dibuat. Tujuannya adalah untuk merumuskan solusi dalam pelaksanaan implementasi pada sistem. Secara garis besar, fitur utama yang terdapat pada program ini adalah:

1. Mendeteksi alat pembatas kecepatan

2. Menampilkan pada peta digital

1. Implementasi

Untuk implementasi sistem pemetaan *bump*, implementasi pada tugas akhir ini adalah sebagai berikut:

* 1. Implementasi Peta Digital

Implementasi pada peta digital yang akan menampilkan alat pembatas kecepatan yang akan ditandai pada peta. Bahasa pemogramman yang digunakan adalah HTML, PHP, dan CSS.

* 1. Implementasi *Server*

Pada *server* akan digunakan basis data Postgresql dan bahasa PHP untuk menyimpan dan mengolah data aktivitas dari sensor akselerometer yang dikirim

* 1. Implementasi Aplikasi Android

Aplikasi untuk mendeteksi alat pembatas kecepatan diimplementasikan pada smartphone Android menggunakan bahasa pemogramman Java.

1. Pengujian dan evaluasi

Pengujian untuk tugas akhir ini sebagai berikut:

* 1. Melakukan uji coba apabila beberapa pengguna mengendarai motor melewati alat pembatas jalan maka pada peta digital akan ditampilkan lokasi alat pembatas kecepatan.
  2. Menghitung akurasi dari hasil deteksi pada perangkat lunak
  3. Pada peta digital akan menampilkan hasil lokasi-lokasi alat pembatas kecepatan yang terdekteksi dan di-*cluster* menggunakan algoritma BIRCH.

1. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan proses dokumentasi dan pembuatan laporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

## Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

1. **Pendahuluan**

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

1. **Dasar Teori**

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

1. **Metode Pemecahan Masalah**

Bab ini membahas mengenai Metode yang digunakan untuk memecahkan masalah yang dipaparkan pada rumusan permasalahan.

1. **Analisis dan Perancangan Sistem**

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur dan proses.

1. **Implementasi**

Bab ini berisi implementasi dari perancangan dan implementasi dalam bentuk coding. Bab ini berisi proses pembangunan perangkat lunak

1. **Pengujian dan Evaluasi**

Bab ini membahas tentang pengujian aplikasi berdasarkan skenario yang telah ditentukan. Mengevaluasi hasil uji coba dari perangkat lunak.

1. **Kesimpulan dan Saran**

Bab ini berisi kesimpulan dari proses pengembangan perangkat lunak dan hasil uji coba.

**Daftar Pustaka**

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

**Lampiran**

Merupakan bab tambahan yang berisi daftar istilah atau kode-kode sumber yang penting pada aplikasi

*[Halaman ini sengaja dikosongkan]*

# BAB II DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir.

## Jalan

Jalan merupakan prasarana transportasi darat yang meliputi segala bagian jalan, termasuk bangunan pelengkap dan perlengkapannya yang diperuntukkan bagi lalu lintas, yang berada pada permukaan tanah, di atas permukaan tanah, di bawah permukaan tanah dan/atau air, serta di atas permukaan air, kecuali jalan kereta api, jalan lori, dan jalan kabel. Jalan sebagai infrastruktur publik yang bertujuan menghubungkan suatu tempat ke tempat lain dalam satu daratan.

### Pengelompokan Jalan

Pengelompokan jalan di wilayah Indonesia diatur menurut UU No. 38 Tahun 2004 tentang Jalan, yaitu:

* Klasifikasi berdasarkan sistem jalan adalah sebagai berikut.

Jalan Primer: Distribusi Lingkup Nasional

Jalan Sekunder: Distribusi Lingkup Perkotaan

* Klasifikasi berdasarkan status atau administrasi pemerintahan sebagai berikut.

Jalan Nasional, adalah jalan arteri dan jalan kolektor pada sistem jaringan jalan primer yang menghubungkan antar ibukota provinsi, dan jalan strategis nasional, serta jalan tol.

Jalan Provinsi, merupakan jalan kolektor dan arteri dalam menghubungkan ibukota provinsi dengan ibukota kabupaten/kota, atau antar ibukota kabupaten/kota dan jalan strategis kabupaten.

Jalan Kabupaten, jalan lokal dalam jaringan jalan primer

Jalan Kota, adalah jalan lokal dalam sistem jaringan jalan primer yang menghubungkan antarpusat pelayanan dalam kota, persil dan antarpersil serta antarpusat pemukiman yang berada di dalam kota.

* Klasifikasi berasarkan fungsional jalan adalah sebagai berikut.

Jalan Arteri, yaitu jalan umum yang digunakan oleh angkutan utama (jarak jauh) dengan ciri-ciri perjalanan jarak jauh, kecepatan rata-rata tinggi dan jumlah jalan masuk dibatasi.

Jalan Kolektor, merupakan jalan umum yang digunakan oleh angkutan pengumpul atau pembagi (jarak sedang) dengan ciri-ciri perjalanan jarak sedang, kecepatan rata-rata sedang dan jumlah jalan masuk dibatasi.

Jalan Lokal, yaitu jalan umum yang melayani angkutan setempat (jarak dekat) kecepatan rendah dan jalan masuk tidak dibatasi.

Jalan Lingkungan, merupakan jalan umum yang berfungsi untuk pelayanan angkutan lingkungan dengan ciri perjalanan jarak dekat, dan kecepatan rendah.

* Klasifikasi Jalan berdasarkan beban muatan sumbu atau disebut juga kelas jalan berdasarkan UU No 14 Tahun 1992 tentang Lalu Lintas dan Angkutan Jalan adalah sebagai berikut:

Jalan Kelas I, yaitu jalan arteri yang dapat dilalui kendaraan bermotor lebar muatan tidak melebihi 2.500 milimeter dan panjang 18.000 milimeter, dengan sumbu terberat yang diizinkan lebih dari 10 ton.

Jalan Kelas II, yaitu jalan arteri yang dapat dilalui kendaraan bermotor dengan lebar muatan tidak melebihi 2.500 milimeter dan panjang 18.000 milimeter, dengan sumbu terberat yang diizinkan 10 ton, contoh penggunaan jalan ini adalah yang sesuai dengan angkutan peti kemas.

Jalan Kelas III A, yaitu jalan arteri atau kolektor yang dapat dilalui kendaraan bermotor dengan lebar muatan tidak melebihi 2.500 milimeter dan panjang 18.000 milimeter, dengan sumbu terberat yang diizinkan 8 ton.

Jalan Kelas III B, yaitu jalan kolektor yang dapat dilalui kendaraan bermotor dengan lebar muatan tidak melebihi 2.500 milimeter dan panjang 12.000 milimeter, dengan sumbu terberat yang diizinkan 8 ton.

### Tipe Jalan Berdasarkan Perkerasan Jalan

Tipe jalan berdasarkan perkerasan jalan adalah jenis jalan yang dibedakan berdasarkan proses pembuatan jalan menggunakan beberapa metode yaitu:

1. Perkerasan Kaku (*Rigid* *Pavement*)

Merupakan jenis jalan yang terbuat dari beton semen, yaitu terdiri dari plat beton semen sebagai lapis pondasi di atas tanah dasar.

1. Perkerasan Lentur (Fleksible *Pavement*)

Merupakan jenis jalan yang terbuat dari aspal atau hot mix.

1. Perkerasan Block (Menggunakan *Paving* *block*)

Merupakan jenis jalan yang terbuat dari campuran pasir dan semen atau disebut juga blok beton terkunci.

### Jenis Kerusakan Jalan

Jenis kerusakan jalan terdapat 2 jenis yaitu kerusakan fungsional dan kerusakan struktural.

1. Kerusakan Fungsional

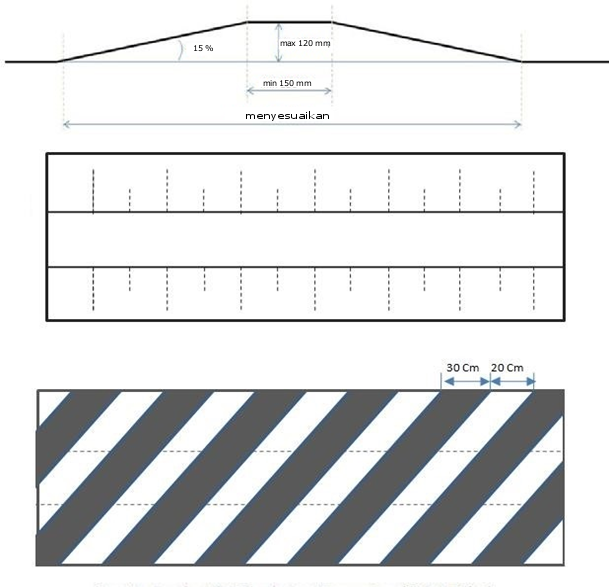
Kerusakan fungsional merupakan kerusakan pada permukaan jalan yang dapat menyebabkan terganggunya fungsi jalan tersebut.

1. Kerusakan Struktural

Kerusakan pada struktur jalan, sebagian atau seluruhnya yang menyebabkan perkerasan jalan tidak lagi menahan beban yang bekerja di atasnya.

## Alat Pembatas Kecepatan

Polisi tidur (*speed bump*) atau disebut juga sebagai alat pembatas kecepatan adalah bagian jalan yang ditinggikan berupa tambahan aspal atau semen yang dipasang melintang dijalan untuk pertanda memperlambat laju atau kecepatan kendaraan [5]. Polisi tidur umumnya mempunyai ukuran dengan tinggi 7,5 cm sampai 15 cm dan lebar 30-90 cm. Dalam pembuatan polisi tidur terdapat ketentuan yang diatur yaitu ketinggian dan rambu-rambu yang memberitahu mengenai adanya polisi tidur, khususnya pada saat malam hari, polisi tidur dilengkapi dengan marka jalan dengan garis serong berwarna putih atau kuning kontras sebagai tanda adanya polisi tidur. Hal ini diperlukan untuk meningkatkan keselamatan dan kenyamanan pengguna jalan. Desain standar polisi tidur diatur dalam Keputusan Menteri Perhubungan No. 3 Tahun 1994 tentang Alat Pengendali dan Pengaman Pemakai Jalan. Desain standar polisi tidur di Indonesia dapat dilihat pada Gambar 2.1.



Gambar 2.1 Desain Standar Polisi Tidur

Alat pembatas kecepatan harus diletakkan pada posisi melintang tegak lurus di jalur lalu lintas. Lokasi penempatan pembatas kecepatan adalah sebagai berikut:

1. Jalan di daerah pemukiman atau Jalan Lingkungan.
2. Pada daerah jalan yang sedang melakukan pekerjaan konstruksi.
3. Pada Jalan Lokal yang memiliki Kelas III C.

Pengaturan ketinggian polisi tidur harus diatur agar tidak membahayakan pemakai jalan karena ketinggian dari polisi tidur berkaitan dengan saat melintas maka beban dan berat tubuh bagian atas akan membuat stres signifikan pada struktur tubuh yang rendah dibagian punggung, terutama pada disk antara *lumbalis* kelima dan *vertebra* *sakral* pertama yang dikenal sebagai L5/S1 *lumbosacral* *disc* atau pengangkatan beban dengan berat beban tubuh bagian atas yang dapat menyebabkan adanya risiko cedera atau berisiko tinggi bagi para penderita osteoporosis. Ketentuan tetang polisi tidur diatur dalam Keputusan Menteri Perhubungan No: KM. 3 Tahun 1994.

## Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler [6]. Pada tugas akhir ini, aplikasi pendeteksi alat pembatas kecepatan akan dibangun pada *smartphone* dengan sistem operasi android. Aplikasi ini akan dibangun menggunakan Android Studio.

### Komponen Aplikasi pada Android

Aplikasi Android menggunakan bahasa pemrograman Java. Android SDK (Software Development Kit) sebagai *compiler* kode aplikasi. Komponen aplikasi adalah bagian penting dalam pengembangan aplikasi Android. Komponen dapat berupa *entry point*, yaitu kontrol penghubung antara sistem operasi dengan program aplikasi. Setiap komponen memiliki ketergantungan dengan komponen yang lain. Berikut ini adalah 4 tipe komponen aplikasi [7].

#### Activities

*Activity* merepresentasikan sebuah tampilan pada antarmuka pengguna. Misalnya, aplikasi email memiliki satu *activity* yaitu untuk membuat email. Namun pada satu aplikasi dapat terdapat beberapa *activity* yang bekerjasama untuk membuat aplikasi yang dapat digunakan oleh user dengan mudah. Sebuah *activity* dapat diimpelementasikan menggunakan subclass Activity.

#### Services

Komponen *service* merupakan komponen yang bekerja pada latarbelakang untuk menjalankan sebuah operasi. *Service* tidak menyediakan antarmuka pengguna. Contoh penggunaan ini adalah menjalankan musik dilatarbelakang (*background*). Sebuah *service* dapat diimpelementasikan menggunakan subclass Services.

#### Content Providers

*Content provider* berguna untuk mengatur seperangkat data aplikasi. Data dapat disimpan dalam sistem, SQLite, dan penyimpanan persisten lainnya. Melalui *content provider,* aplikasi lain dapat mengambil atau memodifikasi data. *Content* *provider* berguna untuk membaca dan menulis data yang rahasia dan tidak boleh dibagikan, misalnya aplikasi notepad menggunakan *content* *provider* untuk menyimpan catatan.

#### Broadcast Receivers

*Broadcast receiver* adalah komponen yang merespon pengunguman *broadcast* ke semua sistem. Contoh *broadcast* yang berasal dari sistem adalah *broadcast* yang muncul ketika layar telah dimatikan, baterai sedikit, dan lain lain. Aplikasi ini juga dapat memulai *broadcast,* misalnya memberitahu aplikasi lain bahwa data telah diunduh dan siap untuk digunakan. Pada umumnya *broadcast receiver* berguna sebagai pintu gerbang untuk komponen lain dan dimaksudkan untuk melakukan jumlah pekerjaan yang sedikit.

#### SensorEventListener

*SensorEventListener* merupakan antarmuka publik yang terdapat pada Android yang berfungsi untuk menerima pemberitahuan pembaharuan data dari *sensorManager*. *Public Method* yang digunakan ada 2 yaitu *onAccuracyChanged* dan *onSensorChanged. Public Method onAccuracyChanged* adalah *listener* aplikasi untuk perubahan akurasi pada sensor yang sedang digunakan, sedangkan *onSensorChanged* adalah *listener* perubahan nilai sensor.

#### LocationListener

*LocationListener* [8]adalah penghubung antara aplikasi dengan GPS. *LocationListener* terdiri dari *LocationManager* dan *LocationListener. LocationManager* adalah layanan akses lokasi pada Android. Salah satu layanan yang tersedia adalah memperbaharui posisi lokasi dimana *smartphone* tersebut sedang digunakan. *Method* ini dipanggil jika *LocationListener* telah terdaftar dengan *LocationManager* menggunakan *requestLocationUpdates()*. Berikut adalah *Public* *Method* yang digunakan.

*onLocationChanged*, fungsi ini dipanggil ketika lokasi berubah. Nilai garis bujur dan garis lintang akan diperbaharui ketika fungsi ini dipanggil dalam bentuk objek *LocationListener*.

*onProviderDisabled,* digunakan ketika provider dinon-aktifkan oleh pengguna.

*onProviderEnabled,* digunakan ketika provider diaktifkan oleh pengguna.

*onStatusChanged,* dipanggil saat ada perubahan status pada provider.

#### Volley

Volley adalah HTTP *library* yang membuat sistem jaringan aplikasi Android menjadi lebih mudah dan cepat. Volley tersedia pada repositori terbuka AOSP [9]. Volley menawarkan berbagai keuntungan yaitu, penjadwalan otomatis pada permintaan jaringan, mendukung banyak koneksi secara bersamaan, mendukung prioritas permintaan, dapat melakukan pemberhentian permintaan, ringan untuk dimodifikasi sesuai kebutuhan.

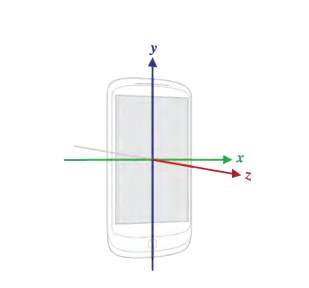
Volley unggul dalam hal operasi bertipe RPC yang digunakan untuk mengisi suatu antarmuka, misalnya mengambil halaman dari suatu pencarian yang terdiri dari data struktur. Volley mudah diintegrasi dengan berbagai protokol dan keluar dari kotak dengan mendukung rangkaian atau *string* mentah, gambar dan JSON. Dengan menyediakan *built-in* dukungan untuk fitur yang dibutuhkan, Volley membebaskan developer untuk menulis kode *boilerplate* dan memperbolehkan developer untuk fokus dalam logika pengembangan aplikasi.

### Sensor pada Android

Kebanyakan perangkat Android telah dilengkapi oleh sensor yang dapat mengukur gerakan, orientasi, dan berbagai jenis keadaan lingkungan sekitarnya. Sensor ini mampu menyediakan data mentah dengan akurat dan presisi. Sensor yang digunakan pada tugas akhir ini adalah sensor akselerometer dan magnometer.

#### Sensor Akselerometer

Sensor akselerometer menghitung percepatan yang terjadi pada device termasuk gaya gravitasi. Akselerometer menggunakan standard sensor koordinat sistem sesuai pada Gambar 2.2. Pada tugas akhir ini, sensor akselerometer pada android akan digunakan untuk memperoleh data akselerasi. Data akslerasi yang digunakan adalah sumbu z yang telah direorientasi dari sumbu koordinat *device* android menjadi koordinat bumi.



Gambar 2.2 Sistem Koordinat (Relatif Terhadap Perangkat)

#### Sensor Magnometer

Sensor magnometer atau disebut juga sensor medan magnet berfungsi untuk melaporkan medan magnet disekitar perangkat *smartphone* yang terukur dalam sensor dengan 3 sumbu. Pengukuran ditampilkan dalam bidang x, y, dan z dalam satuan micro-Tesla (uT). Pada tugas akhir ini sensor magnometer akan digunakan untuk memperoleh nilai matriks rotasi pada *devices*.

## PostgreSQL

PostgreSQL [10] adalah perangkat lunak *open*-*source* object-relational database management system dengan penekanan pada ekstensibilitas dan aturan-standar. PostgreSQL menawarkan kombinasi unik fitur-fitur yang lebih baik jika dibandingakan dengan basis data komersial lainnya seperti Sysbase, Oracle dan DB2. PostgreSQL tidak dimiliki oleh perusahaan manapun. PostgreSQL dikembangkan, dipelihara, diperbaiki oleh grup-grup developer yang sukarelawan di seluruh dunia.

PostgreSQL menawarkan semua kebutuhan fitur dari basis data relasional ditambah beberapa fitur unik. PostgreSQL menawarkan *inheritance*, sehingga pengembang dapat menambahkan tipe data tertentu sesuai yang diinginkan ke PostgreSQL. PostgreSQL juga mendukung tipe data geometris seperti *point*, *line segment, polygon, box* and *circle*. PostgreSQL menggunakan sistem indeks struktur yang membuat tipe data geometris cepat diakses.

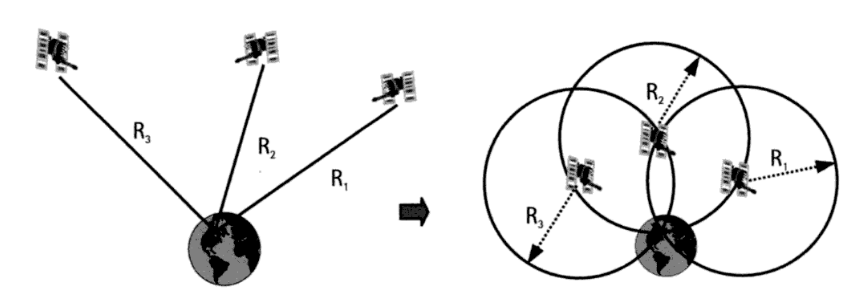
PostgreSQL bersifat *extended* atau dapat dikembangkan (membuat fungsi baru, tipe data baru, operator baru) sesuai dengan bahasa pemrograman yang diinginkan. PostgreSQL dibangun dengan arsitektur *client*/*server*. Aplikasi pada sisi *client* dapat dibangun dengan berbagai bahasa termasuk C, C++, Java, Python, Perl, TCL/Tk dan lain-lain. Pada sisi *server*, PostgreSQL mendukung bahasa prosedural yang baik yaitu PL/pgSQL. Pada tugas akhir ini, basis data PostgreSQL digunakan sebagai penyimpanan data accelerometer, waktu dan posisi koordinat terjadinya *bump*.

## GPS (Global Positioning System)

GPS adalah sistem navigasi berbasis ruang yang menyediakan informasi lokasi dan waktu dalam segala kondisi cuaca, dimanapun di bumi yang tak terhalang pandangan 4 atau lebih satelit GPS. Sistem ini menggunakan 24 satelit yang mengirimkan gelombang mikro ke bumi. Sinyal ini kemudian diterima oleh alat penerima di permukaan bumi. Data yang diterima itu kemudian digunakan untuk menentukaan posisi, kecepatan, arah dan waktu. Satelit GPS mengitari bumi secara sirkular (mendekati bentuk elips) dengan inklinasi sebesar 55 derajat [11].

GPS pada awalanya dikelola oleh US Air Force (USAF) sebagai operator sistem. GPS yang pada awalnya dikembangkan untuk sistem kekuatan tambahan bagi militer memiliki potensi menguntungkan bagi masyarakat sipil. Pada akhirnya GPS menyediakan dua layanan, yaitu layanan Precise Positioning Service (PPS) yang berguna untuk kekuatan militer AS dan layanan Standard Positioning Service (SPS) yang didesain untuk memberikan kemampuan *positioning.*

GPS memiliki 3 segmen, yaitu segmen angkasa, segmen kontrol, dan segmen pengguna. Segmen angkasa terdiri dari 24-satelit yang telah dijelaskan sebelumnya. Setiap satelit GPS secara terus menerus akan mentransmisikan gelombang sinyal radio mikro yang terdiri dari dua *carriers,* dua kode dan pesan navigasi. Ketika GPS *receiver* diaktifkan, GPS *receiver* akan mengambil sinyal melalui antena penerima. Setelah menerima sinyal GPS, sinyal tersebut akan diproses menggunakan software yang ada didalam perangkat tersebut. Secara teori hanya dibutuhkan informasi tiga jarak antara GPS *receiver* dengan satelit, artinya dibutuhkan sebanyak minimal tiga satelit yang melacak secara bersamaan. Pada kasus ini, penerima berada pada persimpangan dari tiga lingkup; masing-masing memiliki jarak radius seperti ditunjukkan pada Gambar 2.3.



Gambar 2.3 Ide Dasar GPS *Positioning*

Pada segmen kontrol sistem GPS terdiri dari stasiun pelacak yang tersebar pada jaringan seluruh dunia, dengan kontrol utamanya berada di negara Amerika Serikat di Colorado Springs, Colorado. Tujuan utama segmen kontrol adalah melacak satelit GPS dengan maksud untuk menentukan dan meprediksi lokasi satelit, integritas satelit, perilaku satelit, jam atom, data atmosfer dan lain-lain. Pada segmen pengguna sendiri terdiri dari seluruh bagian militer dan pengguna sipil. Dengan adanya GPS *receiver* yang tersambung pada antena GPS, pengguna dapat menerima sinyal GPS, yang dapat digunakan untuk menentukan posisi dirinya dimanapun ia berada. GPS sekarang ini tersedia kepada semua pengguna diseluruh dunia dengan gratis.

## PHP (Hypertext Prepocessor)

PHP merupakan bahasa pemrograman *server*-*side* yang digunakan untuk pengembangan web dan bahasa yang digunakan secara luas dalam pengembangan website. PHP dikembangkan pertama kali oleh Rasmus Lerdorf pada tahun 1995. Secara resmi dirilis pada tahun 1997. Fungsionalitas dasar pemrograman berorientasi objek ditambahkan pada PHP 3 dan ditingkatkan pada PHP 4. PHP telah digunakan lebih dari 200 juta website yang berbeda, dari website personal biasa sampai perusahaan raksasa seperti Facebook, Wikipedia, Tumblr, Slack dan Yahoo.

PHP [12] didesain untuk membuat halaman web yang dinamis. PHP berguna sebagai bahasa pemrograman scripting yang dikhususkan untuk pengembangan web *server*-*side*. PHP memiliki peran utama sebagai filter, yaitu mengambil file atau stream yang mengandung teks dan atau instruksi PHP dan menghasilkan data stream lain. PHP dijalankan pada komputer untuk digunakan oleh seorang pengguna, biasanya dijalankan pada web *server* dan diakses oleh banyak pengguna yang menggunakan web browser. PHP memiliki kelebihan yaitu menyembunyikan kompleksitas. Pada tugas akhir ini, PHP akan digunakan untuk membangun aplikasi *web* *service*.

## Python

Python adalah bahasa pemrograman yang digunakan secara luas sebagai bahasa pemrograman yang *high-level*, bertujuan umum, dapat diinterpretasi, dan dinamis [13]. Python didesain dengan filosofi penekanan terhadap kode yang dapat dibaca dan sintaks yang memperbolehkan programmer untuk mengeskpresikan konsep hanya dengan sedikit baris kode dibandingkan pada bahasa lain seperti C++ dan Java. Bahasa pemrograman ini menyediakan gagasan yang memungkinkan untuk pengembangan program dalam skala kecil maupun besar.

Python mendukung berbagai pola pemrograman, termasuk pemrograman berorientasi objek, imperatif, dan pemrograman fungsional atau bergaya prosedural. Python memiliki fitur tipe sistem yang dinamis dan manajemen memori yang otomatis dan memiliki *library* standart yang besar dan komprehensif. Interpreter Python disediakan untuk banyak sistem operasi, sehingga memungkinkan kode Python untuk berjalan dalam sistem yang luas dan bervariasi. Python sangat banyak digunakan untuk pekerjaan kecerdasan buatan. Pada tugas akhir ini, Python digunakan dalam implementasi klasifikasi *decision tree* dan *clustering* BIRCH menggunakan *library sklearn*.

## Google Map API

Google Map API [14] adalah layanan oleh Google yang dapat digunakan untuk menampilkan data dalam bentuk peta digital melalui beberapa model fitur. Google Maps API menyediakan *web service* sebagai antarmuka untuk permintaan data Maps API dari layanan luar dan digunakan pada aplikasi Maps pengguna. Layanan web ini menggunakan HTTP *request* ke URL yang spesifik, meneruskan parameter URL sebagai argumen ke layanan.

Pada umumnya, layanan ini mengembalikan data dengan HTTP *request* berupa JSON atau XML untuk diuraikan dan atau diproses oleh aplikasi. Fitur web service Google Maps API adalah sebagai berikut.

Direction API

Distance Matrixm API

Elevation API

Geolocation API

Road API

Time Zone API

Places API

Fitur yang dapat digunakan pada pengembangan sistem tugas akhir ini adalah menampilkan peta dan menandai peta menggunakan marker. *Library* yang digunakan adalah Google Maps V3 API Biostall dan dikembangkan menggunakan kerangka kerja CodeIgniter [15].

## CodeIgniter

CodeIgniter [16] adalah perangkat lunak sumber terbuka untuk pengengembangan kerangka kerja web secara cepat. CodeIgniter digunakan untuk membangun web site yang dinamis menggunakan PHP. CodeIgniter merupakan kerangka kerja yang menggunakan pola pengembangan *model-view-controller* (MVC). CodeIgniter juga dapat diubah menjadi *Hierarchical Model View Controller* (HMVC) yang berguna untuk pemeliharaan grup modular controller, models dan view yang disusun dalam format satu subdirektori.

CodeIgniter tercatat memiliki kecepatan yang lebih baik dibandingkan dengan kerangka kerja PHP lainnya. CodeIgniter dirilis pertama kali oleh EllisLab pada 28 Februari 2006. Pada 6 Oktober 2014, EllisLab mengumumkan bahwa pengembangan CodeIgniter dilanjutkan oleh British Columbia Institute of Technology. Pada tugas akhir ini CodeIgniter digunakan untuk menampilkan pengolahan data dan peta digital.

## Slim

Slim [17] adalah kerangka kerja mikro berbahasa PHP yang membantu dalam pembuatan web aplikasi dan APIs dengan cepat. Slim menyediakan fitur-fitur yang lengkap dan menggunakan kode yang sederhana. Slim memiliki 4 fitur utama yaitu HTTP *Router*, *Middleware*, PSR-7 *Support* dan *Dependency* *Injection*.

HTTP Router adalah fitur Slim yang menyediakan mekanisme router yang cepat dan *powerful* yang memetakan rute ke HTTP *Method* yang spesifik dan URIs. Slim juga mendukung parameter dan *pattern matching.* Fitur Middleware memungkinkan developer untuk membagun aplikasi yang memudahkan penarikan objek HTTP *request* dan *response.* Fitur PSR-7 adalah Slim mendukung implementasi HTTP *message* sehingga developer dapat mengetahui dan memanipulasi HTTP *message method, status, uri, headers, cookies* dan *body. Dependency Injection* adalah fitur pada Slim yang berguna untuk mempermudah developer dalam mengendalikan alat bantu eksternal dengan menggunakan Container-Interop*.*

## Decision Tree

*Decision tree* adalah metode klasifikasi yang mengubah data menjadi pohon keputusan dan aturan keputusan. *Decision tree* atau pohon keputusan juga memperhatikan faktor-faktor kemungkinan yang mempengaruhi proses pengambilan keputusan tersebut. Pohon keputusan sangat populer karena mudah diinterpretasi oleh manusia. Model prediksi yang digunakan adalah struktur pohon atau struktur hierarki. Algoritma yang digunakan pada *Decision tree* ini adalah CART algorithm. CART *tree* [18] adalah sebuah pohon keputusan *binary* regresi non parametrik untuk klasifikasi atau yang dibangun dengan membagi suatu *node* induk menjadi 2 *node* anak secara berulang dimulai dari *root* *node* berdasarkan hasil *learning* data sampel.

Tahapan yang digunakan dalam membangun pohon keputusan adalah sebagai berikut:

Cari pemisah variabel *predictor* terbaik.

Cari *node* pemisah terbaik. Dari pemisah terbaik pada langkah pertama, pilih satu yang paling baik sebagai pemisah kategori atau kriteria.

Pisahkan *node* menggunakan pemisah terbaik yang ditemukan pada langkah kedua jika hasil belum memuaskan.

## Algoritma *Z-Thresh*

Algoritma *Z-Thresh* merupakan algoritma *thresholding* nilai amplitudo akselerometer pada sumbu z. *Event* direpresentasikan sebagai nilai yang melebihi ambang batas atau *threshold* yang ditentukan. Masukan berupa data akselerometer pada sumbu x, y dan z. Keluaran berupa informasi apakah terdeteksi *bump* atau bukan *bump*. Diagram alir algoritma *z*-*thresh* dapat dilihat pada Gambar 2.4.

C:\Users\Otniel Yeheskiel\Desktop\zthresh.png

Gambar 2.4 Diagram Alir Algoritma Z-Thresh

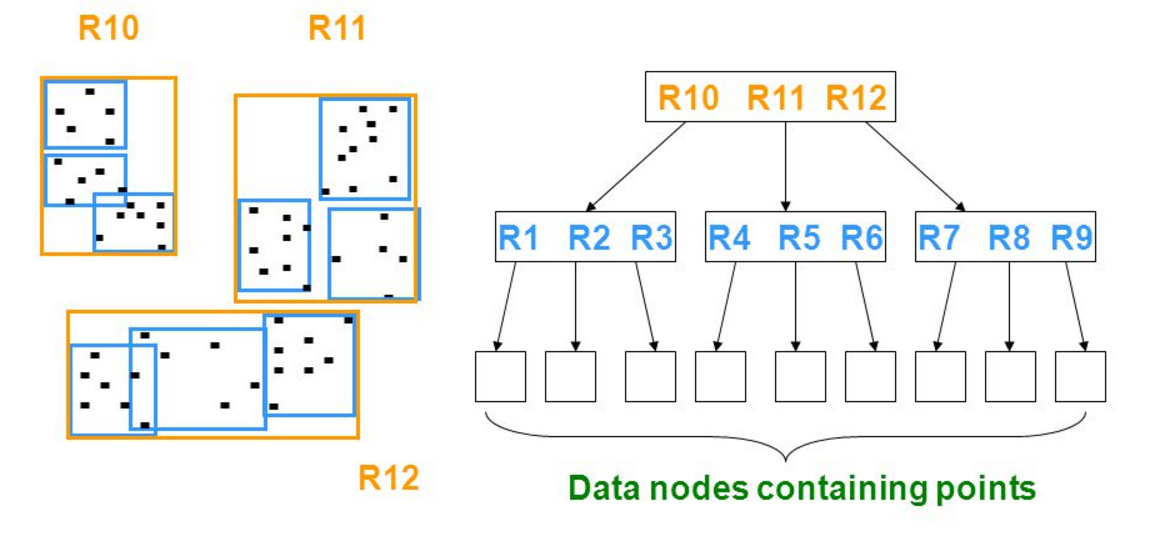
## BIRCH

BIRCH (*Balanced* *Iterative* *Reducing* *and* *Clustering* *using* *Hierarchies*) merupakan algoritma *unsupervised* *machine* *learning* yang melakukan *hierarchical clustering* pada dataset yang besar. Membentuk suatu CF (*Clustering Feature*) *tree* secara meningkat. CF adalah struktur data hirarkikal pada multifase *cluster*. *Clustering* *Feature* terdiri dari 3 fitur yaitu N (jumlah data), LS (*Linear Sum of N data*) dan SS (*Square Sum of N data*).

Fase 1: Pindai data untuk membuat inisial *CF tree*.

Fase 2: Menggunakan algoritma *cluster* yang dinamis untuk melakukan *cluster* *leaf-node*s pada *CF tree*.

*CF tree* memiliki dua parameter yaitu, *branching factor* dan radius threshold. Branching factor jumlah maksimum *CF* *subcluster* pada setiap *node*. Diameter *threshold* adalah nilai radius yang dimiliki *subcluster*. Sampel baru dan *subcluster* terdekat dapat digabung menjadi satu subcluster apabila jaraknya lebih kecil dari *threshold*, sebaliknya (lebih besar dari *threshold*) akan membentuk *subcluster* yang baru. Pada tugas akhir ini algoritma BIRCH diimplementasikan menggunakan *library* scikit-learn Python. Contoh proses *clustering* dapat dilihat pada Gambar 2.5. Pada gambar ditunjukan R10, R11 dan R12 adalah *node*s yang memiliki masing-masing tiga *subcluster* atau *child* yaitu *cluster* R1, R2, dan R3 pada *node*s R10 dan seterusnya.



Gambar 2.5 Visualisasi Birch Clustering

# BAB III ANALISIS DAN PERANCANGAN

Pada bab ini akan membahas tahap analisis dan perancangan sistem yang akan dibangun pada tugas akhir. Perancangan meliputi perancangan data, perancangan proses, perancangan sistem dan perancangan antarmuka perangkat lunak

## Deskripsi Umum Sistem

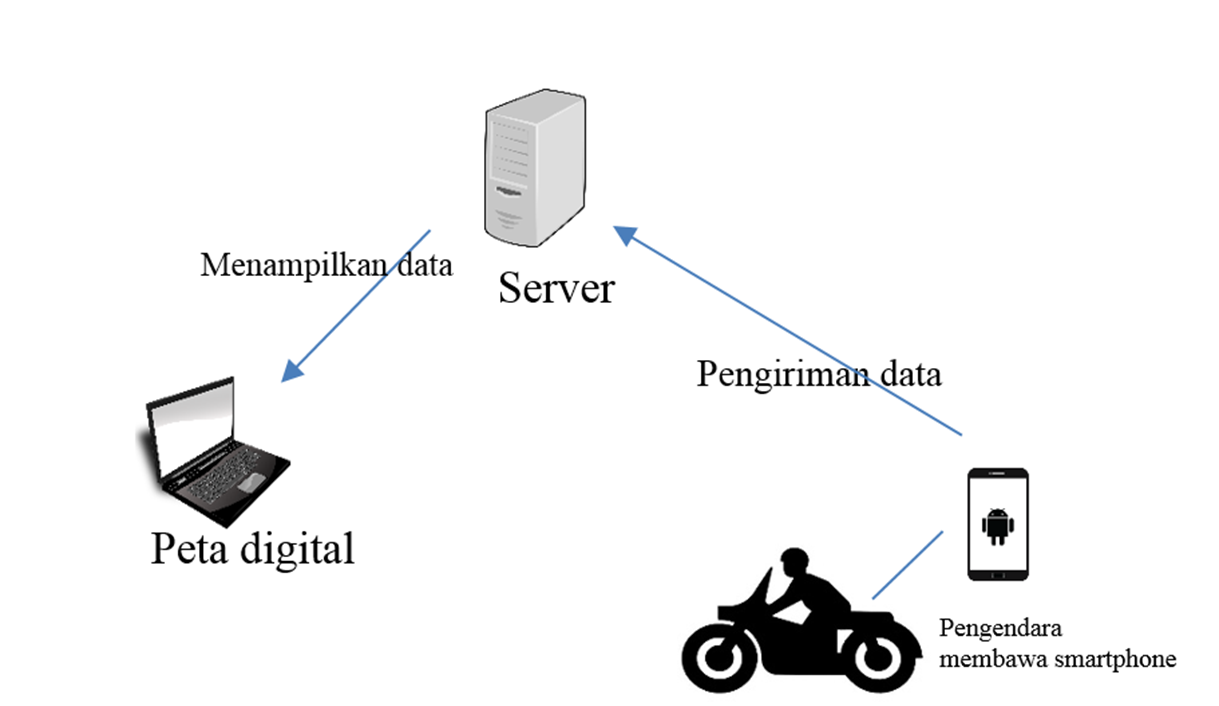
Sistem yang dibangun pada tugas akhir ini merupakan suatu sistem pemetaan *bump* beserta kerusakan jalan yang terdiri dari aplikasi perangkat bergerak berbasis Android sebagai pengirim data dan aplikasi *server* sebagai penerima dan pengolah data lokasi alat pembatas kecepataan lalu menampilkannya pada peta digital.

Aplikasi ini dibangun untuk memetakan lokasi *bump.* Informasi lokasi *bump* ini dapat digunakan oleh pemerintah untuk pertimbangan kebijakan perbaikan dan pemeliharaan jalan. Selain itu dapat digunakan pengguna untuk menghindari jalan yang berbahaya karena jalan rusak.

Tahap awal adalah pengguna akan membawa aplikasi ini kemudian melewati jalan yang terdapat polisi tidur. Alur proses sistem pendeteksi *bump* adalah seperti berikut. Sensor akselerometer pada *smartphone* Android akan diproses untuk mendeteksi *bump* menggunakan algoritma *Z-Thresh*. Apabila terdeteksi sebagai *bump* maka aplikasi Android akan mengirimkan data akselerasi ke *server* beserta lokasi dimana terjadi lonjakan akibat alat pembatas kecepatan.

Data mentah berupa data akselerasi, *timestamp* dan GPS akan diolah lagi pada *server* menggunakan klasifikasi *decision tree* untuk meningkatkan akurasi pada hasil deteksi pada aplikasi Android. Pada *server* data lokasi *bump* akan di-*cluster* pada titik-titik koordinat *bump* yang berdekatan. Kemudian pada *server* akan menampilkan lokasi *bump* yang telah di-*cluster* pada peta digital.

Alur sistem secara keseluruhan ditampilkan pada Gambar 3.1.



Gambar 3.1 Alur Umum Sistem

### Analisis Masalah

Dalam memudahkan untuk memetakan alat pembatas kecepatan atau polisi tidur, maka diperlukan suatu sistem *monitoring* yang dapat memberi lokasi dimana polisi tidur tersebut berada. Identifikasi polisi tidur dapat menggunakan sensor akselerometer yang terdapat pada *smartphone* Android. Data-data akselerasi pengguna android saat melewati polisi tidur dapat dianalisis untuk menentukan adanya suatu *bump* atau tidak.

Permasalahan yang diidentifikasi pada sistem ini adalah bagiamana mengidetifikasi suatu polisi tidur dan mengolah serta menampilkan koordinat lokasi deteksi.

### Analisis Kebutuhan Sistem

Pada subbab ini akan dibahas secara mendalam tentang kebutuhan sistem fungsional dan kebutuhan sistem non fungsional yang diperlukan pada tugas akhir ini.

#### Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan yang memiliki keterkaitan langsung dengan sistem. Kebutuhan fungsional pada sistem sebagai berikut:

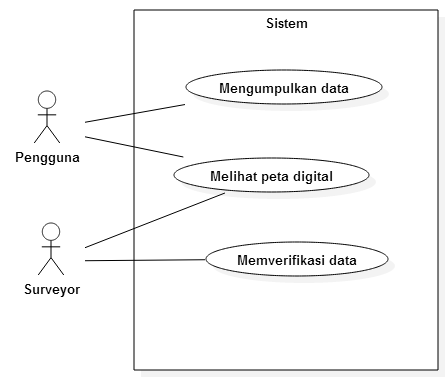
1. Kebutuhan pengguna

* Melihat tampilan peta digital
* Mengumpulkan data akselerometer saat berkendara

1. Kebutuhan surveyor

* Melakukan verifikasi kesesuaian hasil pemetaan pada peta digital

Kebutuhan fungsional sistem digambarkan dalam diagram kasus penggunaan seperti pada Gambar 3.2.



Gambar 3.2 Diagram Kasus Penggunaan

Penjelasan mengenai kasus penggunaan sistem monitoring ini berada pada Tabel 3.1.

Tabel 3.1 Deskripsi Kasus Penggunaan

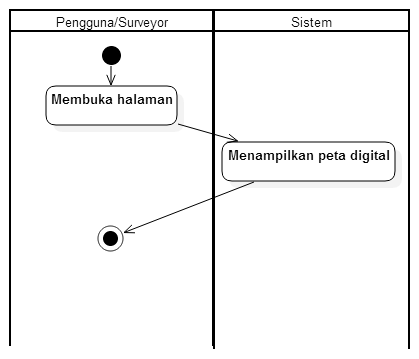
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Kode** | **Nama** | **Keterangan** |
| **1** | UC-01 | Melihat peta digital | Pengguna dan Surveyor dapat melihat informasi pemetaan pada peta digital. |
| **2** | UC-02 | Mengumpulkan data | Pengguna mengumpulkan data akselerometer menggunakan aplikasi pada android. |
| **3** | UC-03 | Memverifikasi data | Surveyor melakukan verifikasi data sesuai pada kondisi di lapangan. |

##### Deskripsi Kasus Penggunaan UC-01

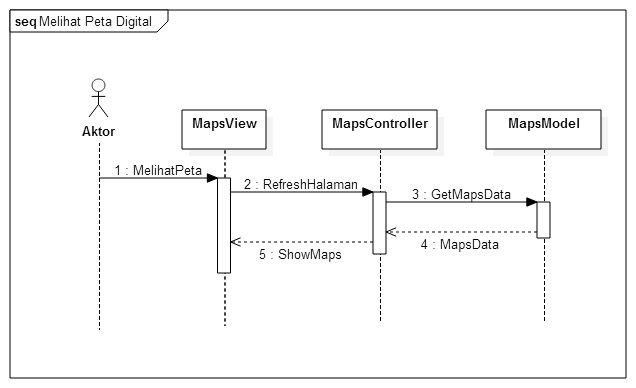
Kasus Penggunaan kode UC-01 adalah kasus penggunaan menampilkan informasi pemetaan pada halaman di peta digital. Detail alur kasus melihat peta digital dijelaskan pada Tabel 3.2 berserta diagram aktivitas kasus yang dijelaskan pada Gambar 3.3. Diagram alir dijelaskan pada Gambar 3.4.

Tabel 3.2 Rincian Alur Kasus Penggunaan UC-01

|  |  |
| --- | --- |
| **Nama *Use Case*** | **Melihat Peta Digital** |
| **Nomor** | UC-01 |
| **Aktor** | Pengguna, Surveyor |
| **Kondisi Awal** | Peta digital belum ditampilkan |
| **Kondisi Akhir** | Peta digital ditampilkan |
| **Alur Normal** | 1. Pengguna/Surveyor membuka halaman peta digital pada browser atau peta pada android. |
| 2. Sistem menampilkan peta digital |



Gambar 3.3 Diagram Aktivitas Kasus Penggunaan UC-01



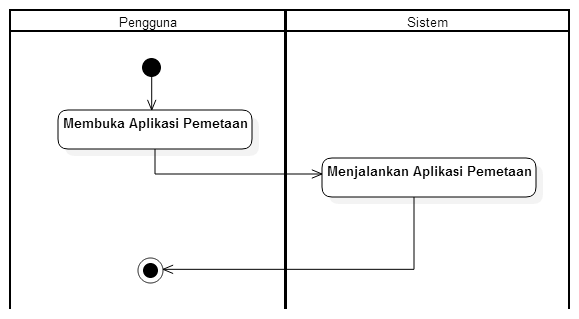
Gambar 3.4 Diagram Alir Kasus Penggunaan UC-01

##### Deskripsi Kasus Penggunaan UC-02

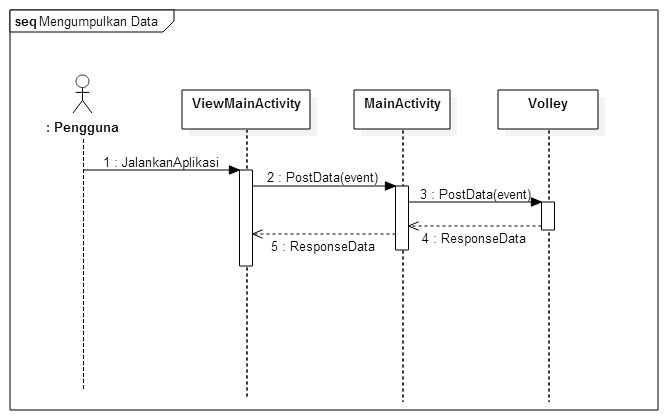
Kasus Penggunaan kode UC-02 merupakan kasus penggunaan mengumpulkan data akselerometer pada perangkat Android. Detail alur kasus melihat peta digital dijelaskan pada Tabel 3.3 berserta diagram aktivitas kasus yang dijelaskan pada Gambar 3.5. Diagram alir dijelaskan pada Gambar 3.6.

Tabel 3.3 Rincian Alur Kasus Penggunaan UC-02

|  |  |
| --- | --- |
| **Nama *Use Case*** | **Mengumpulkan Data** |
| **Nomor** | UC-02 |
| **Aktor** | Pengguna |
| **Kondisi Awal** | Aplikasi pemetaan pada Android belum terbuka |
| **Kondisi Akhir** | Aplikasi pemetaan pada Android berjalan |
| **Alur Normal** | 1. Pengguna membuka aplikasi android |
| 2. Sistem menjalankan aplikasi dan menampilkan antarmuka |



Gambar 3.5 Diagram Aktivitas Kasus Penggunaan UC-02



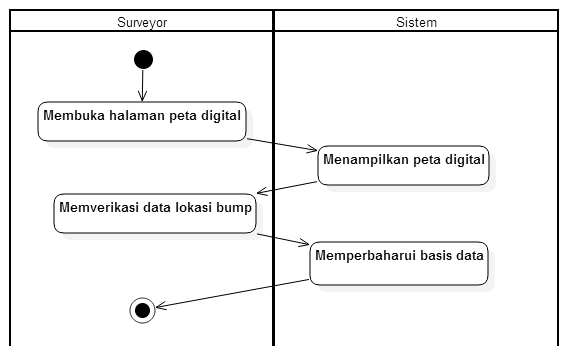
Gambar 3.6 Diagram Alir Kasus Penggunaan UC-02

##### Deskripsi Kasus Penggunaan UC-03

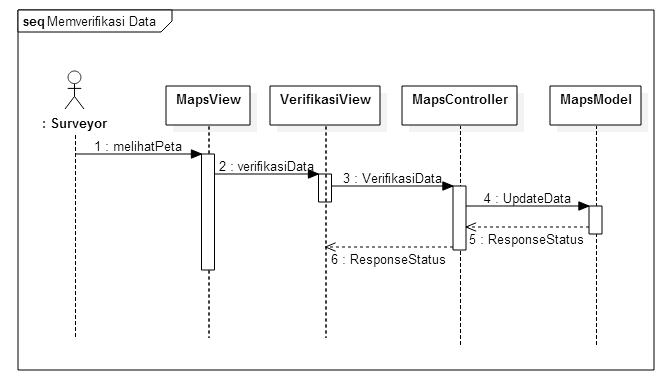
Kasus Penggunaan kode UC-02 merupakan kasus penggunaan melakukan verifikasi data lokasi *bump* yang terdeteksi. Detail alur kasus melihat peta digital dijelaskan pada Tabel 3.4 berserta diagram aktivitas kasus yang dijelaskan pada Gambar 3.7 Diagram alir dijelaskan pada Gambar 3.8.

Tabel 3.4 Rincian Alur Kasus Penggunaan UC-03

|  |  |
| --- | --- |
| **Nama *Use Case*** | **Memverifikasi Data** |
| **Nomor** | UC-03 |
| **Aktor** | Surveyor |
| **Kondisi Awal** | Lokasi data *bump* belum terverifikasi |
| **Kondisi Akhir** | Data lokasi terjadinya *event* *bump* sudah diverifikasi |
| **Alur Normal** | 1. Surveyor membuka halaman verifikasi |
| 2. Sistem menampilkan halaman login |
| 3. Surveyor memverifikasi lokasi yang terdeteksi *bump* |
| 4. Sistem memperbaharui data yang diverifikasi pada basis data |



Gambar 3.7 Diagram Aktivitas Kasus Penggunaan UC-03



Gambar 3.8 Diagram Alir Kasus Kegunaan UC-03

#### Kebutuhan Non Fungsional

Kebutuhan sistem yang diperlukan dalam membangun sistem *monitoring* ini terdiri dari kebutuhan fungsional dan non-fungsional. Kebutuhan non-fungsional yang diperlukan meliputi:

1. Spesifikasi Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk membangun sistem adalah sebagai berikut:

* Sistem Operasi Windows 8.1, Ubuntu 14
* Android 6 Marshmallow
* Android SDK
* Android Studio
* ADT (Android Development Tools)

1. Spesifikasi Perangkat Keras

Pada aplikasi ini, perangkat keras yang digunakan dalam mengembangkan sistem adalah sebagai berikut:

* *Server*: Ubuntu-RAM 512MB - 1CPU
* Evercoss Android One X

Perangkat keras yang digunakan untuk mengambil data berupa *smartphone* berbasis android 6.

## Perancangan Sistem

Tahap ini berupa perancangan basis data sistem, analisis proses sistem, analisis antarmuka peta digital. Pembahasan lebih lanjut akan dibahas sebaai berikut.

### Perancangan Basis Data

Pada subbab ini akan membahas bagaimana rancangan basis data yang digunakan pada sistem pendeteksi *bump*. Basis data yang digunakan adalah postgreSQL. PostgreSQL digunakan untuk menyimpan data akselerometer, jenis, user dan lokasi alat pembatas kecepatan. *Conceptual Data Model* (CDM) dan *Physical Data Model* (PDM) dari basis data dapat dilihat pada Gambar 3.9 dan Gambar 3.10.



Gambar 3.9 CDM (*Conceptual Data Model)* pada *web service*



Gambar 3.10 PDM (*Physical Data Model)* pada *web service*

#### Rancangan Tabel *Location*

Tabel *location* digunakan untuk menyimpan data lokasi terjadinya lonjakan atau polisi tidur pada peta. Tabel ini memiliki relasi dengan tabel lainnya sebagai berikut:

Tabel *Users*

Hubungan dengan tabel ini adalah menyimpan id user setiap kali terjadi *event* lonjakan pada jalan. Kegunaannya adalah untuk mengetahui *device* yang digunakan saat pengambilan data.

Tabel Jenis

Hubungan dengan tabel ini adalah menyimpan id jenis setiap kali melakukan *record* lokasi saat terjadi lonjakan. Manfaatnya adalah untuk mengetahui jenis anomali pada lonjakan.

Tabel *Accelerometer*

Hubungan dengan tabel ini adalah id location tersimpan pada tabel accelerometer setiap kali terjadi lonjakan pada jalan. Manfaatnya adalah untuk mengetahui lokasi data akselerasi.

Detail Atribut Tabel *Location* dijelaskan pada Tabel 3.5.

Tabel 3.5 Atribut Tabel Data Location

|  |  |  |
| --- | --- | --- |
| **Nama Kolom** | **Tipe Data** | **Keterangan** |
| **id** | *long integer* | *Primary Key* pada tabel Location |
| **users\_id** | *long integer* | *Foreign Key* dari tabel Users |
| **jenis\_id** | *integer* | *Foreign Key* dari tabel Jenis |
| **lat** | *decimal* | *Latitude* pada Gmaps |
| **Lon** | *decimal* | *Longitude* pada Gmaps |
| **validasi** | *integer* | Status validasi |

#### Rancangan Tabel *Accelerometer*

Tabel *Accelerometer* digunakan untuk menyimpan data akslerasi yang terjadi setiap kali terjadi lonjakan. Tabel ini memiliki relasi dengan tabel lainnya sebagai berikut:

Tabel *Location*

Hubungan dengan tabel ini adalah menyimpan informasi lokasi setiap kali terjadi *event* lonjakan. Kegunaannya adalah untuk mengetahui informasi lokasi dimana data akslerometer tersebut diambil.

Detail Atribut Tabel *Accelerometer* dijelaskan pada Tabel 3.6.

Tabel 3.6 Atribut Tabel Data Accelerometer

|  |  |  |
| --- | --- | --- |
| **Nama Kolom** | **Tipe Data** | **Keterangan** |
| **id** | *long integer* | *Primary Key* pada tabel Accelerometer |
| **location\_id** | *long integer* | *Foreign Key* dari tabel Location |
| **x** | *decimal* | Data akselerasi yang sudah direorientasi pada sumbu x |
| **y** | *decimal* | Data akselerasi pada sumbu y terhadap koordinat bumi |
| **z** | *decimal* | Data akselerasi pada sumbu z bumi |
| **lat** | *decimal* | *Latitude* pada Gmaps |
| **lon** | *decimal* | *Longitude* pada Gmaps |
| **waktu** | *long integer* | Timemillis (waktu dalam milisecond) saat menyimpan data |

#### Rancangan Tabel Jenis

Tabel Jenis digunakan untuk menyimpan nama jenis anomali pada jalan. Tabel ini memiliki relasi dengan tabel lainnya yaitu sebagai berikut.

Tabel *Location*

Hubungan dengan tabel Location adalah sebagai *foreign key* untuk menyimpan informasi jenis pada lokasi *bump* atau lonjakan yang terdeteksi. Kegunaannya adalah untuk mengetahui informasi nama jenis anomali jalan yang terdeteksi.

Detail Atribut Tabel Jenis dijelaskan pada Tabel 3.7.

Tabel 3.7 Atribut Tabel Jenis

|  |  |  |
| --- | --- | --- |
| **Nama Kolom** | **Tipe Data** | **Keterangan** |
| **id** | *integer* | *Primary Key* pada tabel Jenis |
| **nama** | *varchar(32)* | Nama jenis |

#### Rancangan Tabel Users

Tabel *Users* digunakan sebagai penyimpanan data *user* dan nama *device* yang digunakan. Tabel ini memiliki relasi dengan tabel lainnya yaitu sebagai berikut.

Tabel *Location*

Hubungan dengan tabel ini adalah sebagai *foreign key* untuk menyimpan informasi pengguna yang merekam informasi tersebut. Kegunaannya adalah untuk mengetahui informasi nama *device* yang digunakan saat melakukan pengambilan data.

Detail Atribut Tabel User dijelaskan pada Tabel 3.8.

Tabel 3.8 Atribut Tabel Users

|  |  |  |
| --- | --- | --- |
| **Nama Kolom** | **Tipe Data** | **Keterangan** |
| **id** | *long integer* | *Primary Key* pada tabel Users |
| **nama** | *varchar(32)* | Nama adalah id unik Android yang digunakan |
| **device** | *varchar(32)* | Nama perangkat android (*device*) |

### Perancangan *Web Service*

Pada subbab ini akan membahas desain *web service* yang akan diimplementasikan pada tugas akhir ini. *Web* *service* yang digunakan adalah RESTful *web service* menggunakan kerangka kerja Slim*.* Berikut ini adalah fungsi-fungsi yang akan dibangun pada *web service*.

#### Memasukkan Data *Event Bump*

Fungsi memasukkan data *event bump* adalah fungsi pada *web* *service* yang berguna untuk menerima data berupa JSONArray dari aplikasi Android pada saat terjadi *event bump* dan memasukkan data tersebut ke basis data pada *server*. *Method* yang digunakan pada fungsi ini adalah HTTP *Post*.

Data JSONArray yang dikirim ke *web service* sebagai parameter *request* dijelaskan pada Tabel 3.9.

Tabel 3.9 Parameter *Request* Menerima *Event Bump*

|  |  |
| --- | --- |
| **Nama *Key*** | **Keterangan *Value*** |
| lat | Koordinat lintang terjadinya *event bump* |
| lon | Koordinat bujur terjadinya *event bump* |
| x | Nilai akselerometer pada sumbu x |
| y | Nilai akselerometer pada sumbu y |
| z | Nilai akselerometer pada sumbu z |
| waktu | *Timestamp* terjadinya *event bump* |
| location\_id | id lokasi terjadinya *event bump* |
| jenis\_id | Jenis *bump* |
| user\_id | Id *user* pengguna aplikasi Android |

Parameter *response* yang dikembalikan dari *web service* apabila data telah terkirim dijelaskan pada Tabel 3.10.

Tabel 3.10 Parameter *Response* Menerima *Event Bump*

|  |  |
| --- | --- |
| **Nama *Key*** | **Keterangan *Value*** |
| status | Bernilai “*success*” jika data yang dikirimkan berhasil ditambahkan pada basis data  Bernilai “*fail*” jika data yang dikirimkan gagal ditambahkan pada basis data |

#### Menerima Id *User*

Menerima id *user* adalah fungsi pada *web service* berguna untuk mengirimkan data berupa id *user* dari informasi kode unik Android pada *device* pengguna, apabila id *user* belum terdaftar, maka *web service* akan memasukkan data kode unik Android dan nama *device* penggunake basis data dan *web service* akan mengembalikan nilai id *user*. *Method* yang digunakan adalah HTTP *Post*.

Data JSONObject yang dikirimkan ke *web service* sebagai parameter *request* dijelaskan pada Tabel 3.11.

Tabel 3.11 Parameter *Request* Menerima Id *User*

|  |  |
| --- | --- |
| **Nama *Key*** | **Keterangan *Value*** |
| android\_id | Kode unik Android yang digunakan pengguna |
| device | Nama merek *device* yang digunakan pengguna |

Parameter *response* yang dikembalikan dari *web service* jika data terkirimdijelaskan pada Tabel 3.12.

Tabel 3.12 Parameter *Response* Menerima Id *User*

|  |  |
| --- | --- |
| **Nama *Key*** | **Keterangan *Value*** |
| id\_user | Id pengguna yang akan digunakan pada aplikasi Android |

### Perancangan Antarmuka

Pada subbab ini akan dibahas dengan terperinci dari rancangan antarmuka sistem. Antarmuka sistem terdiri dari antarmuka pada peta digital dan antarmuka pada aplikasi Android. Berikut ini dibahas antarmuka yang terdapat pada sistem yang dibangun untuk tugas akhir ini.

#### Antarmuka pada Peta Digital

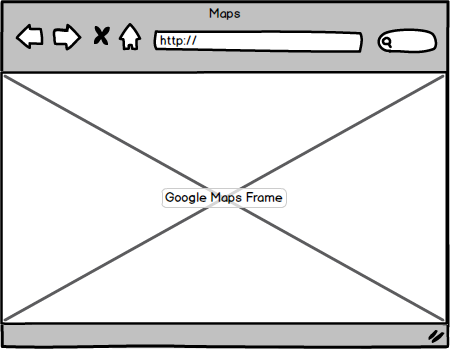
Pada Gambar 3.11 terdapat peta digital beserta *marker* penanda lokasi *bump*. Peta digital ditampilkan menggunakan *library* dari Google Maps API v3 Biostall. Fitur yang akan digunakan Google Maps API adalah sebagai berikut.

Menampilkan peta digital menggunakan Google Maps.

Menandai peta menggunakan *marker.*

Menentukan posisi dan *zoom* pada peta digital.

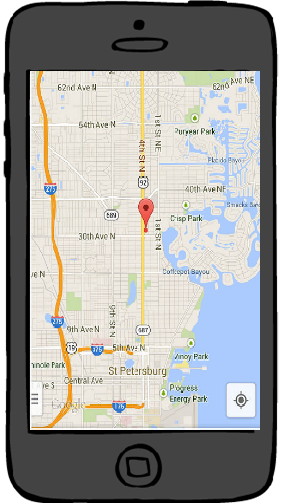
Rancangan antarmuka pada halaman peta digital dapat dilihat pada Gambar 3.11.



Gambar 3.11 Rancangan Antarmuka pada Peta Digital

#### Antarmuka Peta pada Android

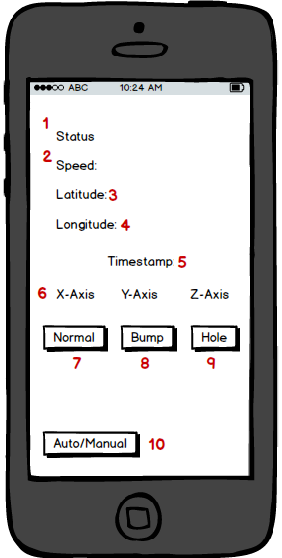
Pada Gambar 3.12 merupakan rancangan antarmuka peta pada aplikasi android. Pada tampilan peta terdapat *marker* yang akan menunjukkan lokasi terjadinya *bump*.



Gambar 3.12 Rancangan Antarmuka Peta pada Android

#### Antarmuka Pengumpulan Data pada Android

Pada Gambar 3.13 merupakan rancangan antarmuka pada aplikasi Android saat pengumpulan data *training*. Komponen-komponen pada aplikasi Android dijelaskan pada nomor berwarna merah.



Gambar 3.13 Antarmuka pada Aplikasi

Keterangan pada Gambar 3.13 berdasarkan nomor-nomor di atas sebagai berikut.

Label untuk status pengambilan data. Statusnya adalah pengiriman auto/manual, sedang mengirim, bump, hole atau kondisi normal.

Label untuk keterangan kecepatan saat pengambilan data.

Label koordinat *latitude*.

Label koordinat *longitude*.

Label timestamp.

Label nilai akselerometer pada X-axis, Y-axis dan Z-axis.

Tombol *training* data pada kondisi jalan Normal.

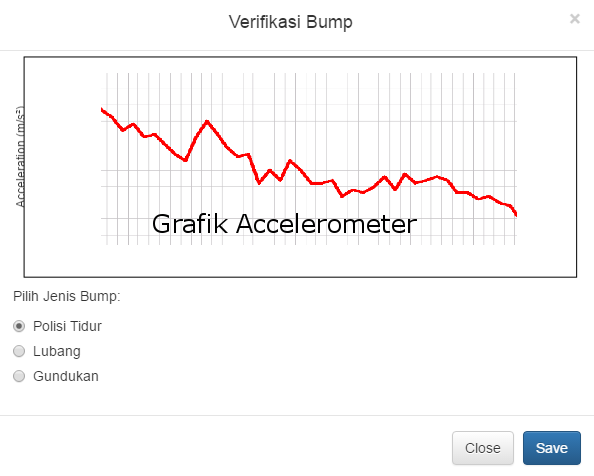
Tombol *training* data pada kondisi melonjak.

Tombol *training* data pada kondisi lubang.

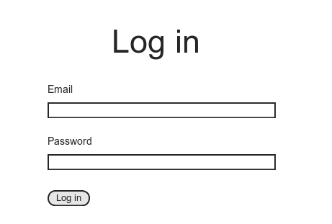
Tombol pengambilan data Auto/Manual.

#### Antarmuka Verifikasi Data

Pada Gambar Gambar 3.14 merupakan rancangan antarmuka verifikasi data pada peta digital. Tampilan verifikasi berupa *modal* Bootstrap 3 yang akan muncul ketika *marker* pada peta digital di-*double* *click*. Antarmuka login verifikasi terlihat pada Gambar 3.15.



Gambar 3.14 Rancangan Antarmuka Verifikasi Data



Gambar 3.15 Rancangan Halaman Login Verifikasi

### Perancangan Proses Data

Pada subbab ini akan dibahas dengan detail rancangan bagaimana pengiriman dan pengolahanan data. Data yang dikirimkan berupa json berisi data akselerometer dan lokasi gps. Pengiriman data JSON dari *smartphone* menuju *server* terjadi setiap kali terjadi *event* yakni ketika pengendara mengalami lonjakan pada jalan.

Data yang dikirim adalah hasil data yang terekam selama 1.5 detik setiap terjadi *event.* Waktu selama 1.5 detik merupakan waktu paling maksimum ketika kendaraan melewati polisi tidur dengan kecepatan yang rendah (kurang dari 10km/jam). Tujuannya adalah pengiriman data yang lebih efisien sehingga tidak perlu menyimpan dan mengirimkan data akselerometer pada jalan normal dan dapat fokus menyimpan data yang menyebabkan *bump.*

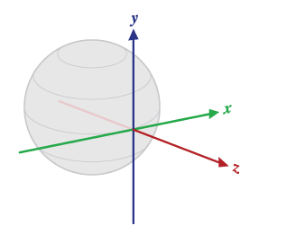
Data tersebut kemudian diprediksi menggunakan klasifikasi dengan metode *decision tree*. Apabila benar maka lokasi data tersebut akan ditampilkan pada peta digital. Pada peta digital koordinat lokasi polisi tidur juga akan dikluster untuk titik-titik yang berdekatan (kurang dari 5-9 meter) menggunakan algoritma BIRCH. Diagram Alir perancangan proses data ditunjukkan pada Gambar 3.16.



Gambar 3.16 Diagram Alir Proses Data

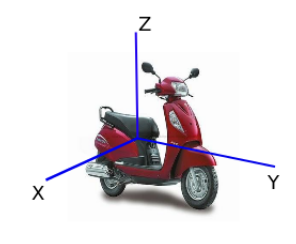
#### Reorientasi Sistem Koordinat Akselerometer

Sensor akselerometer android memiliki sistem koordinat yang relatif terhadap perangkat android itu sendiri. Sistem koordinat akselerometer dapat dilihat pada Gambar 2.2. Apabila perangkat tersebut bergerak atau disorientasi maka dapat mempersulit pengukuran data akselerometer. Untuk itu diperlukan suatu mekanisme yang dapat melakukan reorientasi dari sistem koordinat perangkat menjadi sistem koordinat pada motor. Langkah pertama adalah melakukan reorientasi menjadi sistem koordinat global, baru dari sistem koordinat global, direorientasi lagi menjadi sistem koordinat kendaraan. Tujuannya adalah untuk mengetahui *magnetic* *north*-nya. Hal tersebut dapat dilakukan dengan menggunakan matrix rotasi perangkat android. Matrix rotasi dapat diperoleh dari API sensor android yaitu menggunakan fungsi getRotationMatrix. Setelah memperoleh matrix rotasinya, selanjutnya adalah melakukan multiplikasi matrix rotasi tersebut dengan nilai akselerometer. Sistem Koordinat Global dapat dilihat pada Gambar 3.17.



Gambar 3.17 Sistem Koordinat Global

Setelah memperoleh nilai akselerasi pada sistem koordinat global, selanjutnya adalah melakukan transformasi menjadi sistem koordinat pada kendaraan (Gambar 3.18). Langkah berikutnya adalah menghitung *bearing* yaitu sudut antara 2 titik *latitude*-*longitude* dengan *magnetic* North. *Bearing* dapat diperoleh dengan munggunakan fungsi getBearing pada API GPS Location Android. Selanjutnya setelah mendapatkan nilai *bearing* adalah menemukan sudut yang benar antara *magnetic* North dengan arah kendaraan adalah dengan melakukan pengurangan antara *bearing* dengan *magnetic* *declination*. *Magnetic* *declination* adalah sudut antara *magnetic north* (arah utara pada kompas, sesuai dengan arah garis medan magnet bumi)dengan *true north* (arahsepanjang meridian terhadap kutub utara). Nilai akselerasi yang sesuai dengan sistem koordinat kendaraan dapat diperoleh dengan cara memultiplikasi nilai akselerasi sistem koordinat global dengan matrix rotasi yang merepresentasikan rotasi axis z pada sudut *bearing – declination*.



Gambar 3.18 Sistem Koordinat Kendaraan

#### Pengumpulan Data Training

Proses ini merupakan pengambilan data-data akselerasi pada berbagai kondisi jalan. Kondisi data yang diamati adalah jalan biasa, jalan bergelombang, polisi tidur dan lubang. Data dikirim dari Android dalam bentuk JSON. Proses pengambilan data diambil dalam waktu tertentu saat mengendarai motor. Pengambilan data dimulai ketika pengguna menekan tombol jenis *event*  yang ingin direkam dan berhenti ketika pengguna kembali menekan tombol tersebut. Selama melewati jalan, data akan direcord dalam *JSONArray*, yang kemudian akan dikirim ke *server* setelah selesai mengambil data. Dari data-data tersebut dapat diolah dengan melakukan ekstrak fitur. Fitur ekstrasi yang dipilih adalah standar deviasi, selisih maximum dengan minimum dan mean data akselerasi pada sumbu z.

Selain mengambil data training dengan cara manual (menekan tombol) aplikasi juga dapat melakukan pengumpulan data training dengan otomatis. Pengumpulan data otomatis menggunakan algoritma Z-*thresh*, yaitu apabila data akselerasi pada sumbu z melewati batas *threshold* yang telah ditentukan maka dianggap telah terjadi lonjakan atau *event bump*.

Aplikasi ini akan merekam aktivitas akselerasi pada saat sebelum dan selama 1.5 detik setelah *event bump* terjadi. Untuk memperoleh data akselerasi sebelum terjadinya *event bump* maka data disimpan pada *List* sebagai satu *window*. Ukuran lebar *window* pada tugas akhir ini adalah 20 buah data akselerometer. Isi *window* akan berubah setiap kali data sensor akselerometer masuk. Data ini kemudian akan dikumpulkan sebagai satu jenis data *bump* dan akan dikirimkan ke *server* beserta data lokasi terjadinya *bump*.

#### Pengolahan Data pada *Server*

Pada proses ini, data-data yang telah terkirim saat terjadinya *event bump* akan diolah pada *server*. Data-data yang terkirim berupa data akselerometer pada sumbu x, y, z beserta koordinat *latitude* dan *longitude* pada saat terjadi *bump*. Berikut proses data yang terjadi di *server*.

##### Prediksi *Event* dengan *Decision tree*

Data *event* yang telah dikirimkan ke *server* (data akselerometer) diolah kembali menggunakan algoritma *Decision tree*. Tujuannya adalah untuk mengurangi false positif *bump* yang terdeteksi. Biasanya false positif ini diakibatkan permukaan jalan yang kurang rata sehingga menyebabkan guncangan yang dianggap sebagai *bump*. Namun jenis guncangan seperti ini dapat dikenali berdasarkan aktivitas data akselerometer pada sumbu x dan y. Data akselerometer pada sumbu x dan y tersebut kemudian diekstraksi berdasarkan fitur standart deviasi, rata-rata dan selisih nilai maximum dengan mininum. Fitur tersebut kemudian dipelajari atau di-*training* menggunakan *decision tree*. Setelah data *training* terkumpul, diharapkan *decision tree* ini dapat memprediksi *bump* dengan akurasi yang lebih baik.

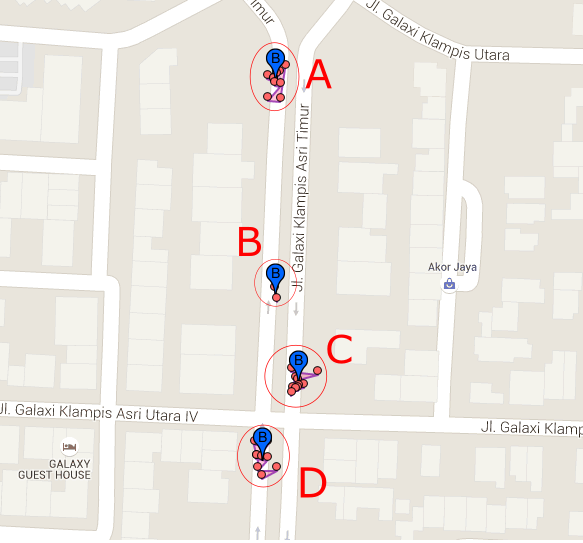
##### Clustering Koordinat pada Peta Digital

Data lokasi *bump* yang telah terkirim ke *server* selanjutnya akan dikelompokkan menjadi satu *subcluster* apabila berdekatan. *Subcluster* pada peta digital adalah titik-titik koordinat yang dikelompokkan menjadi satu. Data lokasi berupa titik-titik koordinat lokasi terjadinya *bump* yang terdeteksi. Titik koordinat *latitude* dan *longitude* yang diperoleh dari hasil deteksi *bump* pada aplikasi Android akan diproses pada *server* untuk di-*cluster.* I

Dalam pengujian, apabila pengendara melewati titik yang sama pada lokasi yang terdapat lonjakan pada pengujian sebelumnya, maka sensor GPS akan memberikan estimasi koordinat dimana lonjakan tersebut terjadi. Namun dikarenakan adanya ketidakakuratan pada GPS, maka titik koordinat yang dihasilkan tidak selalu tepat sama pada satu lokasi lonjakan.

Titik-titik koordinat hasil deteksi yang berdekatan dapat dikluster menjadi satu *subcluster* menggunakan algoritma BIRCH. Algoritma BIRCH adalah algoritma algoritma *unsupervised* (tanpa pengawasan) *machine* *learning* yang melakukan *hierarchical clustering*. Lalu dari titik-titik pada satu *subcluster* itu akan diambil titik *centroid* yang akan menjadi representasi lokasi bump di *subcluster* tersebut.

Pada Gambar 3.19 terdapat lingkaran A, B, C dan D yang merupakan *subcluster* dihasilkan dari titik-titik *bump* (pada marker dot merah). Masing-masing s*ubcluster* tersebut diwakilkan oleh titik sentroidnya (pada marker warna biru).



Gambar 3.19 Contoh Hasil *Clustering* pada Peta Digital

*[Halaman ini sengaja dikosongkan]*

# BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari analisis dan perancangan sistem yang telah dibahas pada Bab III. Namun dalam penerapannya, rancangan tersebut dapat mengalami perubahan minor sewaktu-waktu apabila dibutuhkan.

## Lingkungan Implementasi

Dalam proses perancangan sisitem ini digunakan perangkat pendukung sebagai berikut.

### Lingkungan Implementasi Perangkat Keras

Pada Tabel 4.1 perangkat keras yang digunakan dalam proses pengembangan sistem perangkat lunak adalah Laptop HP, 2.2 Ghz processor Intel Core i3, 4 GB RAM. Perangkat keras yang digunakan sebagai *server* untuk implementasi *web service* adalah *server* dari Digital Ocean dengan 512 MB memory. Perangkat *smartphone* yang digunakan adalah Android One X Evercoss A65, Sistem Operasi Android 6 Marshmallow, Memori Internal 8 GB, RAM 1 GB.

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

|  |  |
| --- | --- |
| *Hardware* | Laptop HP, 2.2 Ghz processor Intel Core i3, 4 GB RAM |
| *Server* | Digital Ocean: 512 MB Memory, Ubuntu 14. |
| *Smartphone* | Android One X Evercoss A65, Sistem Operasi Android 6 Marshmallow, Memori Internal 8 GB, RAM 1 GB. |

### Lingkungan Implementasi Perangkat Lunak

Pada Tabel 4.2 menjelaskan implementasi perangkat lunak yang digunakan dalam proses pengembangan dan implementasi aplikasi ini.

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak

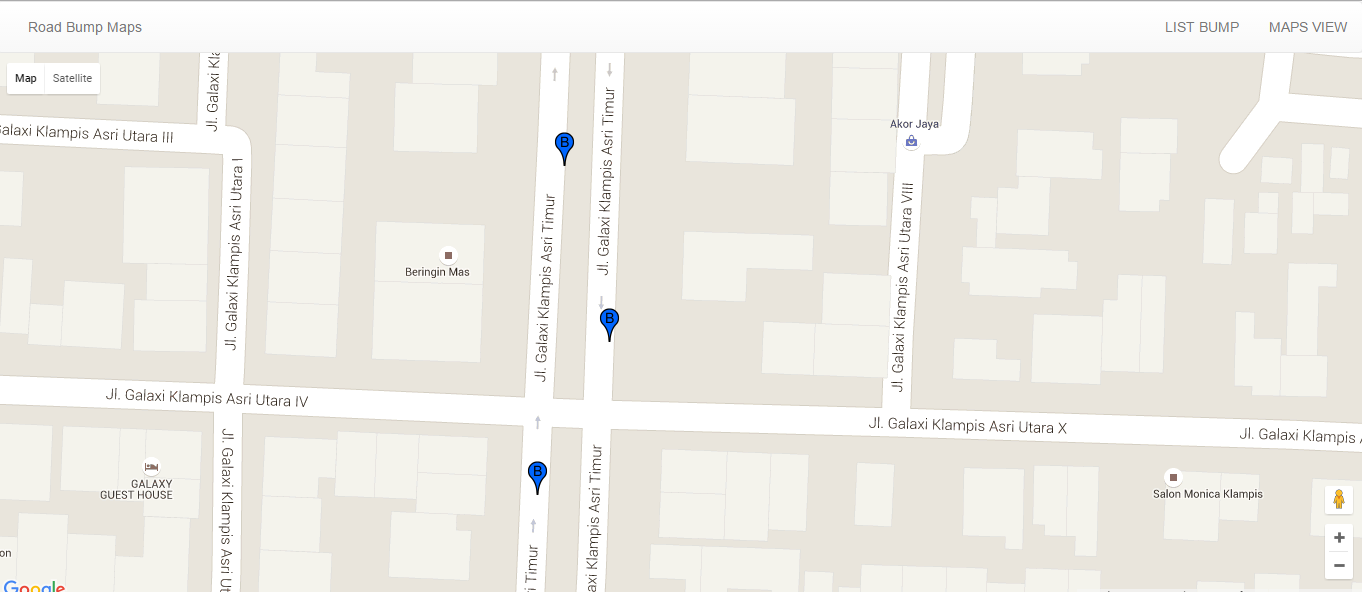
|  |  |
| --- | --- |
| **Nama Aplikasi** | **Kegunaan** |
| Microsoft Windows 8.1 | Sebagai sistem operasi pada perangkat keras pengembangan aplikasi |
| Ubuntu 14 | Sebagai sistem operasi pada *server* |
| PostgreSQL | Sebagai basis data pada pengembangan aplikasi. |
| XAMPP | Sebagai web *server* aplikasi |
| Phppgadmin | Sebagai aplikasi manajemen basis data |
| Power Designer 15 | Sebagai aplikasi perancangan basis data |
| Android Studio | Sebagai IDE untuk implementasi aplikasi |
| Sublime 3 | Sebagai teks editor untuk penulisan kode sumber |
| Putty dan WinSCP | Untuk instalasi dan integrasi web service pada *server* |

## Lingkungan Implementasi Antarmuka

Pada subbab ini dibahas bagaimana implementasi anatrmuka berdasarkan rancangan antar muka yang telah dibahas pada Bab 3.

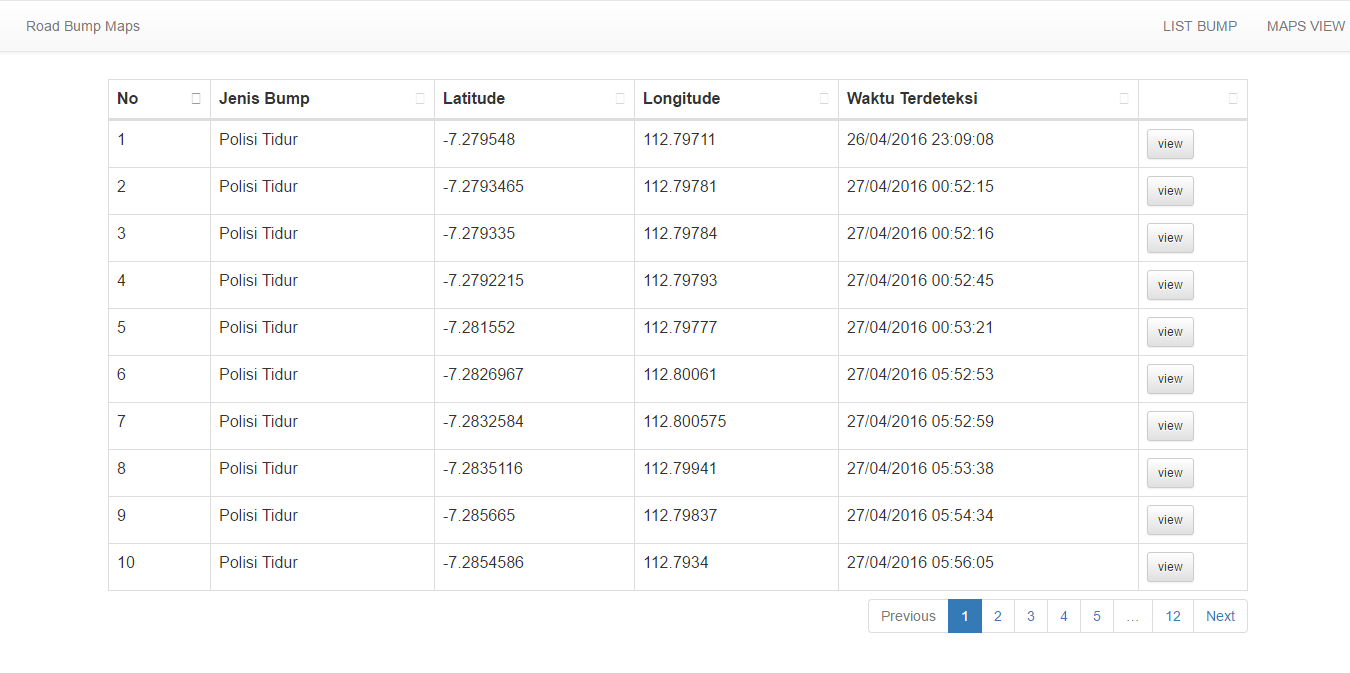
### Antarmuka Peta Digital pada *Server*

Implementasi tampilan peta digital dapat dilihat pada Gambar 4.1. Marker berwarna merah menunjukkan lokasi polisi tidur hasil observasi langsung sedangkan marker berwarna biru menunjukkan *bump* hasil deteksi dari aplikasi.



Gambar 4.1 Tampilan Peta Digital Hasil Deteksi *Bump*

Impelentasi informasi bump juga dapat dilihat pada Gambar 4.2. Daftar *bump* ditampilkan dalam bentuk tabel dengan informasi *latitude*, *longitude*, jenis, dan waktu terdeteksinya.



Gambar 4.2 Tampilan Halaman List *Bump*

### Antarmuka Peta pada Android

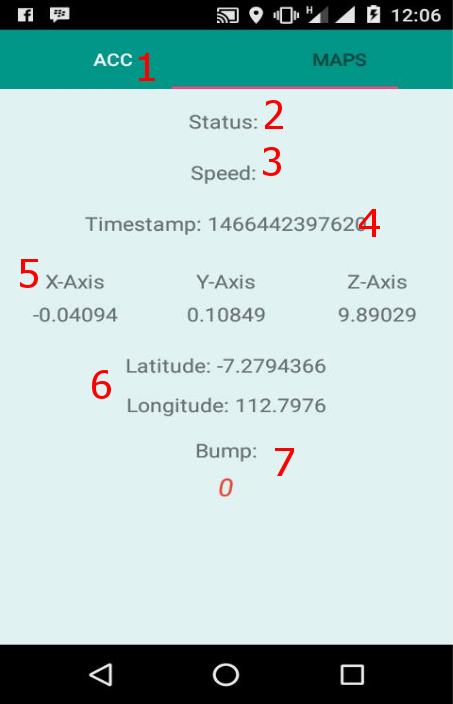
Implementasi peta pada aplikasi Android ditunjukkan pada Gambar 4.3. Keterangan pada peta sebagai berikut. Nomor 1 menunjukkan judul fragment yang sedang aktif atau dibuka. Pada Nomor 2 merupakan tombol untuk memperoleh lokasi pengguna dan pada nomor 3 merupakan marker yang menunjukkan lokasi terjadinya *bump*.



Gambar 4.3 Tampilan Peta pada Android

### Antarmuka Pengumpulan Data pada Android

Implementasi aplikasi Android ditunjukkan pada Gambar 4.4. Aplikasi setelah dijalankan dapat langsung digunakan untuk melakukan pengumpulan data.



Gambar 4.4 Tampilan Aplikasi Utama

Keterangan pada Gambar 4.4 berdasarkan nomor-nomor di atas sebagai berikut.

Label untuk menunjukkan fragment yang aktif.

Label untuk status pengiriman data. Statusnya adalah *bump* atau normal.

Label untuk keterangan kecepatan saat pengambilan data.

Label untuk menunjukkan nilai *timestamp*.

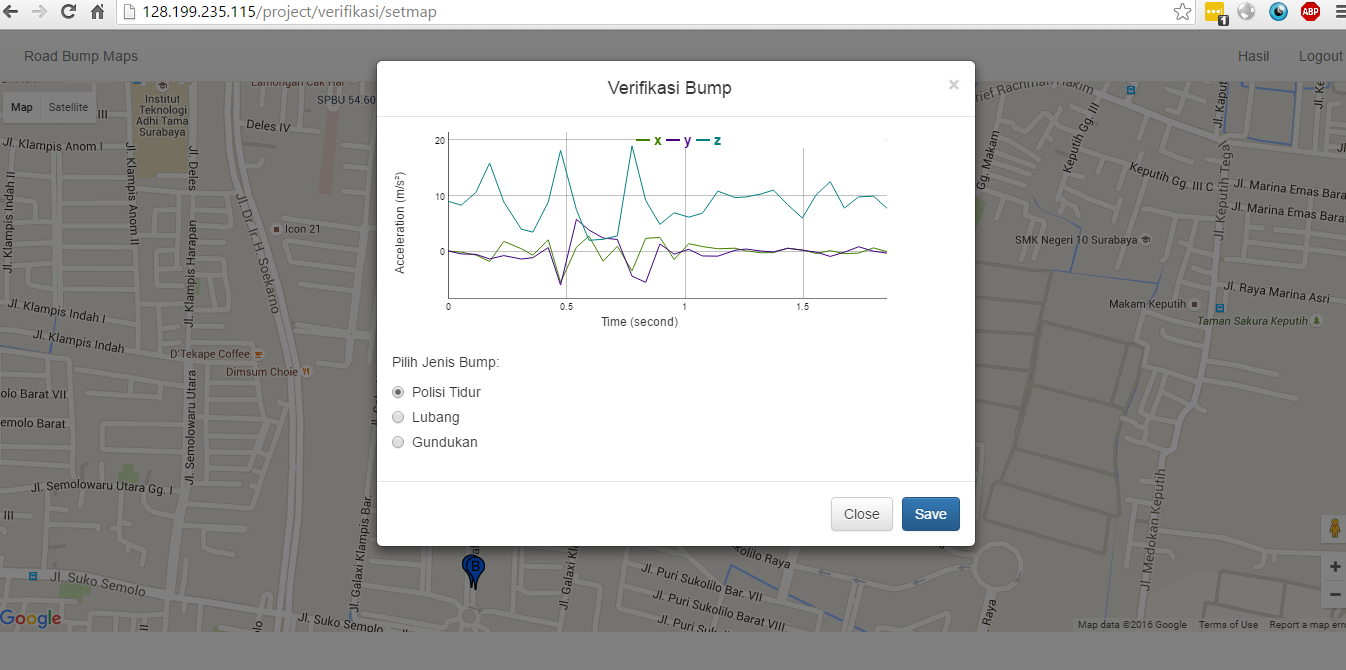
Label nilai akselerometer pada X-axis, Y-axis dan Z-axis.

Label koordinat *latitude* dan *longitude*

Label jumlah *bump* yang terdeteksi.

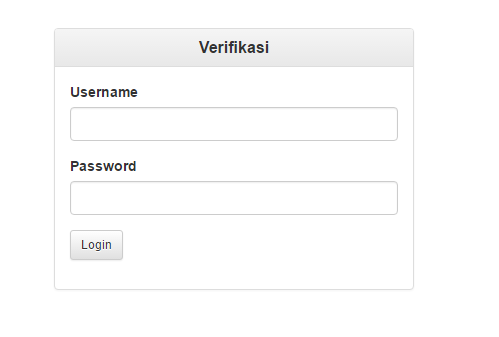
### Antarmuka Verifikasi Data

Implementasi verifikasi data ditunjukkan pada dialog *modal* Bootstrap dapat dilihat pada Gambar 4.5. *Modal* berisi grafik nilai akselerometer pada *bump* dan pilihan jenis *bump* berupa radio *button.*.



Gambar 4.5 Tampilan Verifikasi Data

Halaman login verifikasi ditunjukkan pada Gambar 4.6. Login form berupa username dan password yang akan dikirimkan ke *server*.



Gambar 4.6 Halaman Verifikasi Login

## Implementasi Basis Data

Pada subbab ini akan dibahas secara mendetail rancangan basis data yang telah dipaparkan pada Bab 3. Pada implementasi ini dijelaskan mengenai pembuatan tabel-tabel yang digunakan pada sistem. Implementasi tabel basis data menggunakan sintaks postgreSQL.

### Implementasi Tabel *Location*

Tabel *location* adalah tabel yang digunakan untuk menyimpan data lokasi terjadinya lonjakan atau polisi tidur pada peta. Implementasi tabel *location* dapat dilihat pada Kode Sumber 4.1. Tabel *location* memiliki atribut yaitu *latitude* dan *longitude* sebagai koordinat lokasi terjadinya *bump*. Jenis\_id dan user\_id sebagai *foreign key*. Validasi sebagai status data *location* beserta id tabel sebagai primary key.

|  |
| --- |
| CREATE TABLE location (  lat double precision NOT NULL,  lon double precision NOT NULL,  jenis\_id integer,  id integer NOT NULL,  user\_id integer,  validasi integer DEFAULT 0  );  CREATE SEQUENCE location\_id\_seq  START WITH 1  INCREMENT BY 1  NO MINVALUE  NO MAXVALUE  CACHE 1;  ALTER TABLE ONLY location ALTER COLUMN id SET DEFAULT nextval('location\_id\_seq'::regclass);  ALTER TABLE ONLY location  ADD CONSTRAINT location\_id\_jenis\_fkey FOREIGN KEY (jenis\_id) REFERENCES jenis(id); |
|  |

Kode Sumber 4.1 Implementasi Tabel *Location*

### Implementasi Tabel *Accelerometer*

Tabel *accelerometer* adalah tabel yang digunakan untuk menyimpan data akslerasi yang terjadi setiap kali terjadi lonjakan. Tabel *accelerometer* memiliki beberapa atribut sebagai berikut: id, latatitude dan *longitude*, nilai akselerometer pada sumbu x, y dan z, beserta waktu dan location­\_id sebagai foreign key dari tabel *location*. Implementasi tabel *accelerometer* ditunjukkan pada Kode Sumber 4.2.

|  |
| --- |
| CREATE TABLE accelerometer(  id bigint NOT NULL,  lat double precision NOT NULL,  lon double precision NOT NULL,  x double precision,  y double precision,  z double precision NOT NULL,  waktu bigint,  location\_id integer  ); |
| CREATE SEQUENCE accelerometer\_id\_seq  START WITH 1  INCREMENT BY 1  NO MINVALUE  NO MAXVALUE  CACHE 1; ALTER TABLE ONLY accelerometer ALTER COLUMN id SET DEFAULT nextval(' accelerometer\_id\_seq'::regclass); |
| ALTER TABLE ONLY accelerometer  ADD CONSTRAINT accelerometer\_pkey PRIMARY KEY (id); |

Kode Sumber 4.2 Implementasi Tabel *Accelerometer*

### Implementasi Tabel Jenis

Tabel Jenis merupakan tabel yang digunakan untuk menyimpan nama jenis anomali pada jalan. Data yang disimpan berupa data nama dan id jenis guncangan atau anomali jalan. Atribut pada tabel ini adalah id dan nama jenis. Implementasi ditunjukkan pada Kode Sumber 4.3.

|  |
| --- |
| CREATE TABLE jenis ( |
| id integer NOT NULL, |
| nama character varying(50) ); ALTER TABLE ONLY jenis  ADD CONSTRAINT jenis\_pkey PRIMARY KEY (id); |

Kode Sumber 4.3 Implementasi Tabel Jenis

### Implementasi Tabel *Users*

Tabel *users* merupakan tabel yang digunakan sebagai penyimpanan data *user* dan nama *device* yang digunakan. Atribut pada tabel *users* adalah id, nama dan *device*. Implementasi tabel ini dapat dilihat pada Kode Sumber 4.4.

|  |
| --- |
| CREATE TABLE users ( |
| id integer NOT NULL, |
| nama character varying(50),  device character varying(50)  );  CREATE SEQUENCE users\_id\_seq  START WITH 1  INCREMENT BY 1  NO MINVALUE  NO MAXVALUE  CACHE 1; ALTER TABLE ONLY users   ALTER COLUMN id SET DEFAULT nextval('users\_id\_seq'::regclass); ALTER TABLE ONLY users  ADD CONSTRAINT users\_pkey PRIMARY KEY (id); |

Kode Sumber 4.4 Implementasi Tabel *Users*

## Implementasi *Web* *Service*

Pada subbab ini akan dibahas bagaimana implementasi *web service* dilakukan. *Web service* yang digunakan pada sistem ini menggunakan metode *representational state transfer* (REST) dengan *framework* Slim. Format data yang digunakan dalam proses pertukaran data adalah *JavaScript Object Notation* (JSON). Pada subbab ini akan dibahas implementasi dari proses *query* pada *web service*.

### Implementasi *Query User*

*Query* *User* merupakan *query* yang dipanggil saat aplikasi pada Android pertama kali dijalankan. *Query* ini bertujuan untuk mengambil id user dari tabel *users* berdasarkan nama kode unik android pada *device* tersebut. Kode unik Android adalah id berupa string yang unik dan dimiliki setiap perangkat Android. Apabila kode unik Android belum ada pada tabel *users,* maka kode unik tersebut akan disimpan di kolom namapada tabel *users* sehingga *device* tersebut memiliki id user. Implementasi *query* memperoleh id user dapat dilihat pada Kode Sumber 4.5.

|  |  |
| --- | --- |
| function getUserId(data){ | |
|  | OPEN db\_connection |
|  | SET data to json\_decode(data) |
|  | SET sql\_query to "SELECT id FROM users where nama = ". data->nama |
|  | EXECUTE sql\_query |
|  | SET result to fetch(sql\_query) |
|  |  |
|  | IF result is NULL |
|  | THEN |
|  | SET sql\_statement to "INSERT INTO users (nama,device) VALUES (:nama,:device) RETURNING id " |
|  | PREPARE sql\_statement |
|  | ADD data->nama to bindParam(:nama) |
|  | ADD data->device to bindParam(:device) |
|  | EXECUTE sql\_statement |
|  | SET result to fetch(sql\_statement) |
|  | ENDIF |
|  |  |
|  | CLOSE db\_connection |
|  | RETURN result->id\_user |

Kode Sumber 4.5 Implementasi *Query User*

### Implementasi *Query* *Event Bump*

*Query* *Event* adalah *insert query* yang dipanggil ketika terjadi *event* guncangan atau *bump* pada *device* ketika aplikasi berjalan. *Query* ini akan menerima data berupa JSONArray dari *smartphone* lalu memasukkan data tersebut ke dalam tabel *location* dan *accelerometer*. Pada fungsi insertEvent() terdapat 2 *query* yaitu *query* ketika *insert* ke tabel *location* dan *insert* ke tabel *accelerometer. Query* pertama setelah insert ke tabel *location* akan mengembalikan location\_id yang akan digunakan sebagai *foreign key* saat *query* insert data ke tabel akselerometer. *Pseudocode* impelementasi *query event* dapat dilihat pada Kode Sumber 4.6.

|  |  |
| --- | --- |
| function insertEvent(location,accelerometer){ | |
|  | OPEN db\_connection |
|  | SET data to json\_decode(location) |
|  |  |
|  | SET sql\_query to "INSERT INTO location (lat, lon, jenis\_id,user\_id) VALUES (:lat, :lon, :jenis\_id,:user\_id) RETURNING id" |
|  | PREPARE sql\_query |
|  | ADD data->lat to bindParam(:lat) |
|  | ADD data->lon to bindParam(:lon) |
|  | ADD data->jenis\_id to bindParam(:jenis\_id) |
|  | ADD data->user\_id to bindParam(:user\_id) |
|  | EXECUTE sql\_query |
|  |  |
|  | SET id\_location to fetch(sql\_query) |
|  | SET acc to json\_decode(accelerometer) |
|  |  |
|  | FOR all items IN acc |
|  | DO |
|  | SET sql\_query2 to "INSERT INTO acc\_data (lat, lon, x, y, z, waktu, location\_id) VALUES (:lat, :lon, :x, :y, :z, :waktu, :location\_id)" |
|  | PREPARE sql\_query2 |
|  | ADD acc->lat to bindParam(:lat) |
|  | ADD acc->lon to bindParam(:lon) |
|  | ADD acc->jenis\_id to bindParam(:jenis\_id) |
|  | ADD acc->x to bindParam(:x) |
|  | ADD acc->y to bindParam(:y) |
|  | ADD acc->z to bindParam(:z) |
|  | ADD acc->user\_id to bindParam(:waktu) |
|  | ADD location\_id to bindParam(:location\_id) |
|  | EXECUTE sql\_query2 |
|  | ENDFOR |
|  |  |
|  | CLOSE db\_connection |
|  | RETURN STATUS\_SUCCESS |

Kode Sumber 4.6 Implementasi *Query* *Event* *Bump*

### Implementasi Koneksi Basis Data

Koneksi basis data yang digunakan sebagai penghubung antara *web* dengan basis data. Koneksi ini diperlukan untuk proses pengiriman dan penerimaan data yang dibutuhkan antara *server,* peta digital dan *aplikasi* *smartphone*. Implementasi koneksi basis data dapat dilihat pada *pseudocode* Kode Sumber 4.7.

|  |  |
| --- | --- |
| getDB() | |
|  | SET db\_host |
|  | SET db\_user |
|  | SET db\_pass |
|  | SET db\_name |
|  | Add to db\_connection |
|  | CONNECT db\_connection |
|  | IF db\_connection NOT NULL |
|  | THEN RETURN db\_connection |
|  | ELSE RETURN STATUS\_FAILED |
|  | ENDIF |

Kode Sumber 4.7 *Pseudocode* Koneksi Basis Data

## Implementasi Proses Aplikasi

Subbab ini menjelaskan tentang proses-proses yang terjadi dalam aplikasi. Proses-proses yang terjadi mulai dari inisialisasi sensor, reorientasi sistem koordinat akselerometer hingga pengiriman data dari *smartphone* ke *server*.

### Implementasi Inisialisasi Sensor

Pada Kode Sumber 4.8 adalah kode sumber implementasi inisialisasi *SensorEventListener* untuk sensor *accelerometer* dan *magnetic field.*

|  |  |
| --- | --- |
|  | INITIALIZE SensorManager |
|  | GET SENSOR\_SERVICE |
|  | INITIALIZE sensor TYPE\_ACCELEROMETER |
|  | INITIALIZE sensor TYPE\_MAGNETIC\_FIELD |
|  | REGISTER sensor TYPE\_ACCELEROMETER listener |
|  | REGISTER sensor TYPE\_MAGNETIC\_FIELD listener |

Kode Sumber 4.8 Implementasi Inisialisasi Sensor

### Implementasi Pengiriman Data

Pengiriman data *event* dari *devices* ke *server* menggunakan *library* Volley. Volley adalah HTTP *library* yang biasa digunakan untuk pengiriman jaringan data. Pada Kode Sumber 4.10 dapat dilihat *pseudocode* implementasi pengiriman data *event bump* berupa data *location* dan *accelerometer* setiap terjadi peristiwa lonjakan pada *devices* menggunakan HTTP POST *method* yang diambil dari *library* Volley. Nama fungsi pengirimannya adalah postEventData(). Data yang dikirimkan dalam bentuk format JSONArray. Pada Kode Sumber 4.9 adalah implementasi fungsi addToJSONArray() yaitu fungsi untuk memasukkan data-data yang mau dikirimkan berupa JSONObject kedalam JSONArray. Variabel dataJsonArray adalah variabel global dengan tipe data JSONArray di dalam aplikasi ini.

|  |  |
| --- | --- |
| addToJSONArray(data) | |
|  | INITIALIZE json\_object |
|  | FOR all\_items in data |
|  | DO |
|  | PUT all\_items to json\_object |
|  | ENDFOR |
|  | PUT json\_object to dataJsonArray |

Kode Sumber 4.9 Implementasi Fungsi *addToJSONArray*

|  |  |
| --- | --- |
| postEventData(data\_array) | |
|  | INITIALIZE JsonArrayPost |
|  |  |
|  | ADD PARAM RequestMethod to POST |
|  | ADD PARAM URL |
|  | ADD PARAM Data to data\_array |
|  | ADD PARAM ResponseListener |
|  | ADD PARAM ResponseErrorListener |
|  |  |
|  | SET PARAM to JsonArrayPost |
|  | SEND JsonArrayPost |

Kode Sumber 4.10 Implementasi Fungsi *postEventData*

### Implementasi Mengambil Data Sensor

Kode Sumber 4.11 merupakan pseudocode untuk mengambil data sensor yaitu sensor akselerometer dan magnometer setiap kali sensor mengalami perubahan. Fungsi onSensorChanged adalah fungsi yang dijalankan ketika sensor mengalami perubahan nilai.

|  |  |
| --- | --- |
| onSensorChanged() | |
|  | IF sensor TYPE\_ACCELEROMETER |
|  | THEN |
|  | get values to accelerometers\_array |
|  | ENDIF |
|  | IF sensor TYPE\_MAGNETIC\_FIELD |
|  | THEN |
|  | get values to magnometers\_array |
|  | ENDIF |

Kode Sumber 4.11 Implementasi Mengambil Data Sensor

### Implementasi Reorientasi Akselerometer

Reorientasi data akselerometer dari koordinat *devices* menjadi koordinat kendaraan dapat dilihat pada *pseudocode* Kode Sumber 4.12. Fungsi reorientation() dijalankan dalam fungsi onSensorChanged(). Apabila terjadi perubahan pada sensor akselerometer maka nilai akselerasi akan direorientasi. Metode yang digunakan untuk mengimplementasi program sesuai dengan metode reorientasi yang digunakan pada penelitian aplikasi Wolverine [3]. Fungsi untuk mendapatkan RotationMatrix sistem koordinat bumi adalah getRotationMatrix. RotationMatrix harus *ditranspose* agar dapat dimultiplikasi dengan matriks vektor akselerometer [19].

|  |  |
| --- | --- |
| reorientation() | |
|  | INITIALIZE RotationMatrix |
|  | INITIALIZE global\_accelerometers |
|  | INITIALIZE vehicle\_accelerometers |
|  | GET gravity[] values with HPF(acc) |
|  | RotationMatrix := getRotationMatrix(gravity,magnometers) |
|  | TRANSPOSE RotationMatrix |
|  | global\_accelerometers := MULTIPLY accelerometers with RotationMatrix |
|  | GET bearing FROM LocationManager |
|  | GET declination FROM GeomagneticField |
|  | INITIALIZE HeadingMatrix |
|  | SET Heading := bearing – declination |
|  | GET HeadingMatrix from angle Heading in Z-axis |
|  | vehicle\_accelerometers:= MULTIPLY global\_accelerometers with HeadingMatrix |
|  | RETURN vehicle\_acclerometers |

Kode Sumber 4.12 Implementasi Reorientasi Akslerometer

### Implementasi Deteksi *Event* *Bump*

Proses pendeteksian *event bump* atau guncangan akibat lonjakan menggunakan algoritma *Z-Thresh* [4]. Apabila data akselerasi pada sumbu z melewati nilai *threshold* maka aplikasi akan mendeteksi bahwa *event bump* sedang terjadi. Setelah berhasil dideteksi maka selanjutnya aplikasi akan merekam aktivitas data pada sensor akselerometer selama 1.5 detik. Kumpulan data akselerometer yang terekam selama 1.5 itu kemudian dikirimkan maka data lokasi dan akselerometer akan dikirimkan ke *server*. Nilai *threshold* terdiri dari nilai batas atas dan batas bawah. Implementasi deteksi *event bump* dapat dilihat pada Kode Sumber 4.13.

|  |  |
| --- | --- |
|  | SET x\_acc |
|  | SET y\_acc |
|  | SET z\_acc |
|  |  |
|  | IF z\_acc > threshold\_top |
|  | OR z\_acc < threshold\_bottom |
|  | THEN |
|  | WHILE 1.5s DO |
|  | ADD location to data |
|  | ADD accelerometer to data |
|  | ENDWHILE |
|  | postEventData(data) |
|  | ENDIF |

Kode Sumber 4.13 Implementasi Deteksi *Event Bump*

### Implementasi *Decision tree*

Kode Sumber 4.14 merupakan *pseudocode* implementasi untuk klasifikasi *bump* atau normal menggunakan algoritma *decision tree.* Pada setiap data *event*, data akan diklasifikasi berdasarkan fitur yang diekstraksi yaitu simpangan baku, rata-rata dan *diff* (selisih nilai maximum dan nilai minimum). Fungsi DecisionTreeClasifier yang digunakan adalah fungsi klasifikasi dari library *sklearn.tree* Python [20].

|  |  |
| --- | --- |
|  | INITIALIZE data\_train |
|  | STORE stdZ, diffZ, meanZ, stdY, diffY, meanY to data\_train |
|  | INITIALIZE treeClasifier |
|  | SET treeClasifier to DecisionTreeClassifier(data\_train) |
|  | INITIALIZE event\_bump |
|  | STORE stdZ, diffZ, meanZ, stdY, diffY, meanY to event\_bump |
|  | PREDICT event\_bump using treeClasifier |
|  | GET predict\_result |
|  | OPEN dbConnection |
|  | UPDATE table location SET jenis\_id to predict\_result |
|  | CLOSE dbConnection |

Kode Sumber 4.14 Implementasi Klasifikasi *Decision tree*

### Implementasi *Clustering* pada Peta Digital

Pada Kode Sumber 4.15 dapat dilihat *pseudocode* implementasi *clustering* yang digunakan untuk mengelompokkan titik-titik koordinat *bump*  yang berdekatan pada peta digital. Fungsi Birch merupakan fungsi yang diambil dari *library sklearn.cluster.birch* Python. Parameter threshold pada fungsi Birch merupakan jarak maximum antar koordinat yang boleh dijadikan satu *subcluster*. Data koordinat diperoleh dari database menggunakan fungsi getBumpLocation(). Data tersebut kemudian di*cluster* menggunakan Birch menjadi beberapa kelompok *subcluster*. Setiap subcluster memiliki satu titik pusat luasan atau *centroid* yang mewakili titik-titik pada subcluster tersebut. Titik-titik tersebut akan ditampilkan pada peta digital.

|  |  |
| --- | --- |
|  | INITIALIZE location |
|  | SET location to getBumpLocation() |
|  | INITIALIZE map\_cluster |
|  | INITIALIZE Birch, label |
|  | CLUSTER map\_cluster with Birch(threshold = 9 m) |
|  | SET label to CLUSTER.result |
|  | INITIALIZE marker, map |
|  | FOR all items in label |
|  | DO |
|  | SET marker to centroid(label->position) |
|  | ENDFOR |
|  | ADD marker to map |
|  | LOAD map |

Kode Sumber 4.15 Implementasi *Clustering* pada Peta Digital

*[Halaman ini sengaja dikosongkan]*

# BAB V PENGUJIAN DAN EVALUASI

Pada bab ini dibahas tentang uji coba dan evaluasi pada perangkat lunak yang dibangun pada tugas akhir ini. Pengujian dilakukan pada fungsionalitas sistem perangkat lunak. Pada bab ini juga membahas mengenai dasar pengujian, pengujian fungsionalitas, pengujian akurasi, analisa data dan evaluasi pengujian.

## Lingkungan Uji Coba

Dalam proses pengujian perangkat lunak, dibutuhkan suatu lingkungan uji coba yang sesuai dengan standar kebutuhan. Uji coba aplikasi ini dilakukan dengan menggunakan beberapa smartphone. Berikut ini adalah *smartphone* yang digunakan.

* Perangkat *mobile* Evercoss A65
* Sistem Operasi Android Marshmallow 6.0
* Processor 1.3 GHz ARMv7 rev3 (v71)
* Memory Card Slot up to 16 GB
* Internal Storage 5.8 GB
* RAM 1 GB
* Wi-Fi 802.11 a/b/g/n/ac, dual-sim

## Dasar Pengujian

Pengujian yang dilakukan berupa pengujian fungsionalitas dan pengujian akurasi. Pengujian fungsionalitas dilakukan dengan model *black box* untuk setiap fungsionalitas yang diuji dari aplikasi ini. Pengujian ini dilakukan untuk menguji kebutuhan fungsionalitas yang diidentifikasi telah diimplementasikan dengan baik seperti yang diinginkan.

Pengujian akurasi adalah pengujian dengan melakukan pengumpulan data dan melihat kesesuaian (akurasi) hasil pengumpulan dengan hasil observasi pada keadaan sesungguhnya. Perhitungan akurasi menggunakan *precision* dan *sensivity* karena jumlah *true* *negative* yang banyak sehingga tidak dapat dihitung. Presisi adalah perbandingan antara *true* *positive* dengan total hasil yang terdeteksi positif (*true* *positive* dan *false* *positive*). Formula perhitungan nilai presisi dapat dilihat pada Persamaan 5.1.

|  |  |
| --- | --- |
|  | (5.1) |

Selain presisi, sensitivitas juga dihitung untuk mengetahui nilai perbandingan antara *true* *positive* dengan total keseluruhan polisi tidur (*true* *positive* dan *false negative*). Formula perhitungan sensitivitas dapat dilihat pada Persamaan 5.2.

|  |  |
| --- | --- |
|  | (5.2) |

Langkah pertama yang dilakukan untuk menguji ketepatan aplikasi dalam mendeteksi polisi tidur adalah dengan melakukan observasi pada lokasi dan menandai setiap polisi tidur yang ada pada lokasi pengujian. Cara menandai lokasi polisi tidur hasil pengamatan adalah dengan berhenti pada tepat di atas polisi tidur kemudian mengirimkan koordinat posisi lokasi polisi tidur ke *server*. Saat pengambilan koordinat harus berhenti agar posisi yang dihasilkan GPS lebih akurat. Setelah memperoleh koordinat lokasi polisi tidur pada lokasi pengujian, selanjutnya adalah melakukan pengujian aplikasi dengan berbagai kondisi. Kondisi yang diuji adalah tata cara penempatan *smartphone,* kecepatan kendaraan, threshold Z-axis, metode, dan lokasi pengujian. Dari hasil pendeteksian oleh aplikasi ini dapat dihitung akurasi (presisi) setiap pengujian.

## Uji Coba Fungsionalitas

Uji coba fungsionalitas adalah uji coba fungsi-fungsi yang berjalan pada aplikasi. Pengujian dilakukan oleh pengguna dengan tujuan untuk menjalankan aplikasi pemetaan lokasi *bump* dan melihat hasilnya peta digital. Uji coba fungsionalitas terdiri dari beberapa skenario pengujian.

1. Uji coba melihat peta digital
2. Uji coba mengumpulkan data

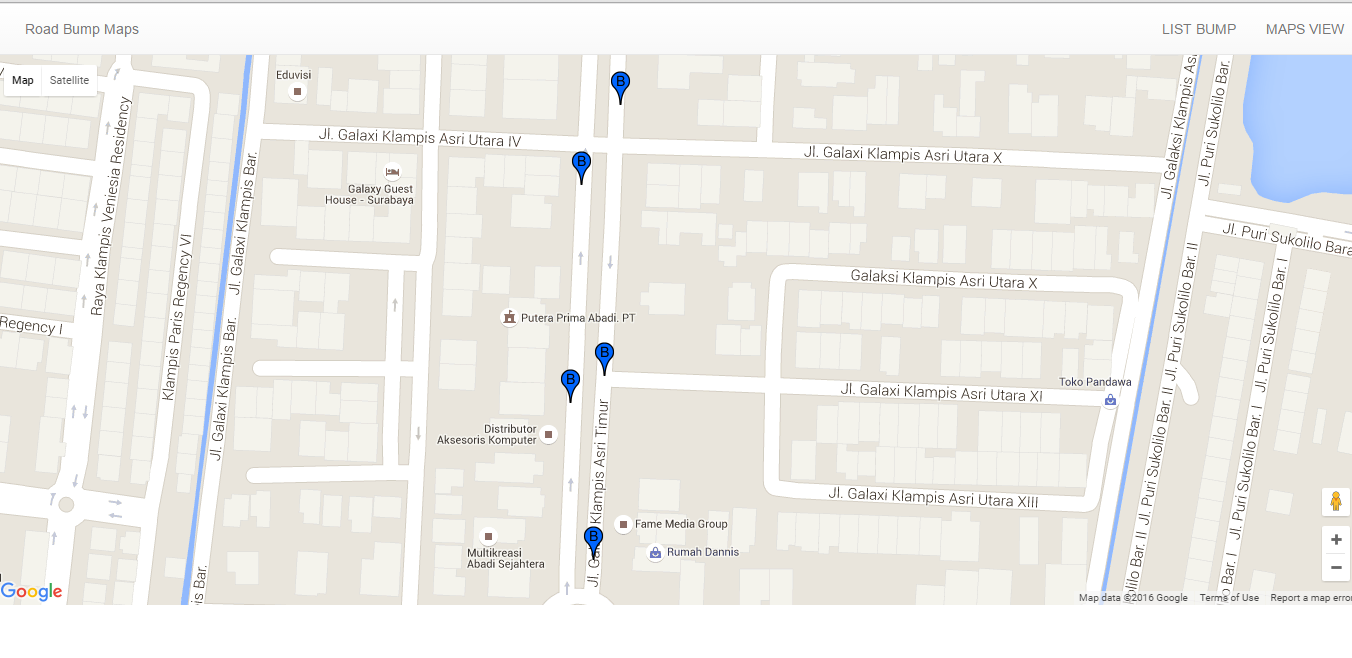
### Uji Coba Fungsionalitas Melihat Peta Digital

Uji coba melihat peta digital dimulai dengan membuka halaman pada *browser* setelah itu memasukkan alamat *website.* Setelah memasukkan alamat, peta digital akan ditampilkan ke layar. Pengguna dapat melihat informasi lokasi-lokasi terjadinya *bump* berdasarkan posisi marker yang ditampilkan pada peta. Tabel skenario uji coba melihat peta digital ditunjukkan pada Tabel 5.1.

Tabel 5.1 Tabel Skenario Uji Coba Melihat Peta Digital

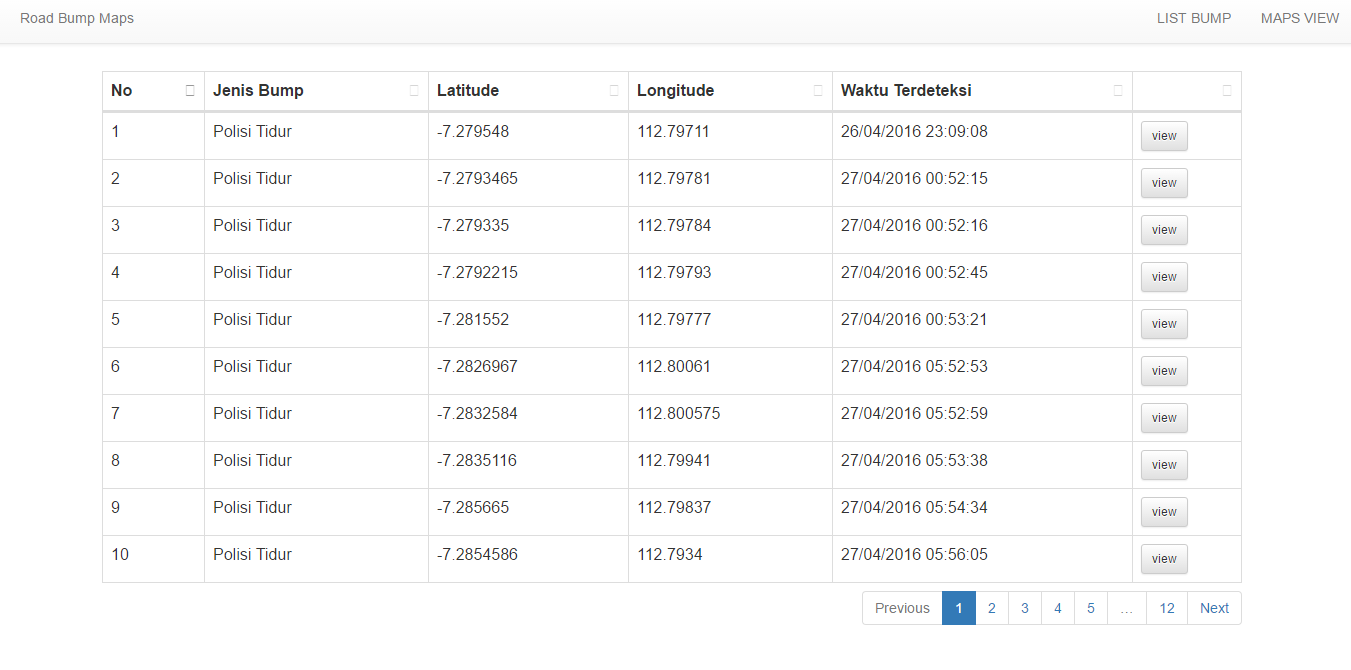
|  |  |
| --- | --- |
| **Nama Uji Coba** | **Melihat Peta Digital** |
| **Kondisi Awal** | Halaman Kosong |
| **Aksi yang dilakukan** | Memasukkan alamat *website* |
| **Hasil yang diharapkan** | 1. Peta digital menampilkan lokasi *bump* |
| **Hasil yang diperoleh** | 1. Peta digital menampilkan lokasi *bump* dengan marker 2. Website dapat menampilkan list data *bump* dalam bentuk tabel |
| **Hasil Uji Coba** | Berhasil |

Pada Gambar 5.1 dapat dilihat lokasi *bump* yang telah di-*cluster* pada peta digital. Lokasi bump ditunjukan pada marker berwarna biru.



Gambar 5.1 Tampilan Lokasi *Bump* pada Peta Digital

Selain melihat peta lokasi *bump,* pengguna juga dapat melihat list *bump* pada halaman *List Bump* dengan memilih tombol pada *navbar* list bump. *List* *Bump* akan ditampilkan dalam bentuk tabel berisi yang berisi informasi jenis *bump*, *latitude*, *longitude* dan waktu terdeteksinya. Tampilan list bump dapat dilihat pada Gambar 5.2.



Gambar 5.2 Tampilan *List Bump*

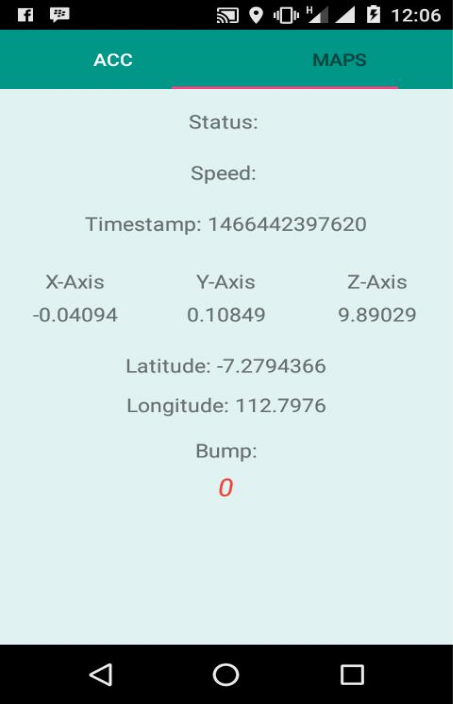
### Uji Coba Fungsionalitas Mengumpulkan Data

Uji coba mengumpulkan data adalah proses pengambilan data oleh pengguna dengan menjalankan aplikasi Android. Aplikasi ini saat pengguna akan melakukan perjalanan menggunakan motor. Hal yang pertama yang dilakukan pengguna adalah membuka aplikasi, kemudian aplikasi akan berjalan dan mengirim data apabila terjadi guncangan ketika berkendara. Skenario uji coba mengumpulkan data ditunjukkan pada Tabel 5.2.

Tabel 5.2 Tabel Skenario Uji Coba Mengumpulkan Data

|  |  |
| --- | --- |
| **Nama Uji Coba** | **Mengumpulkan Data** |
| **Kondisi Awal** | Aplikasi belum dijalankan |
| **Aksi yang dilakukan** | Aplikasi berjalan |
| **Hasil yang diharapkan** | 1. Aplikasi berjalan dan dapat mengumpulkan data *bump* |
| **Hasil yang diperoleh** | 1. Aplikasi berjalan dan mengumpulkan data *bump* |
| **Hasil Uji Coba** | Berhasil |

Pada Gambar 5.3 dapat dilihat tampilan aplikasi ketika dijalankan. Apabila aplikasi telah berjalan maka aplikasi telah siap digunakan untuk mengumpulkan data.



Gambar 5.3 Tampilan Aplikasi Android

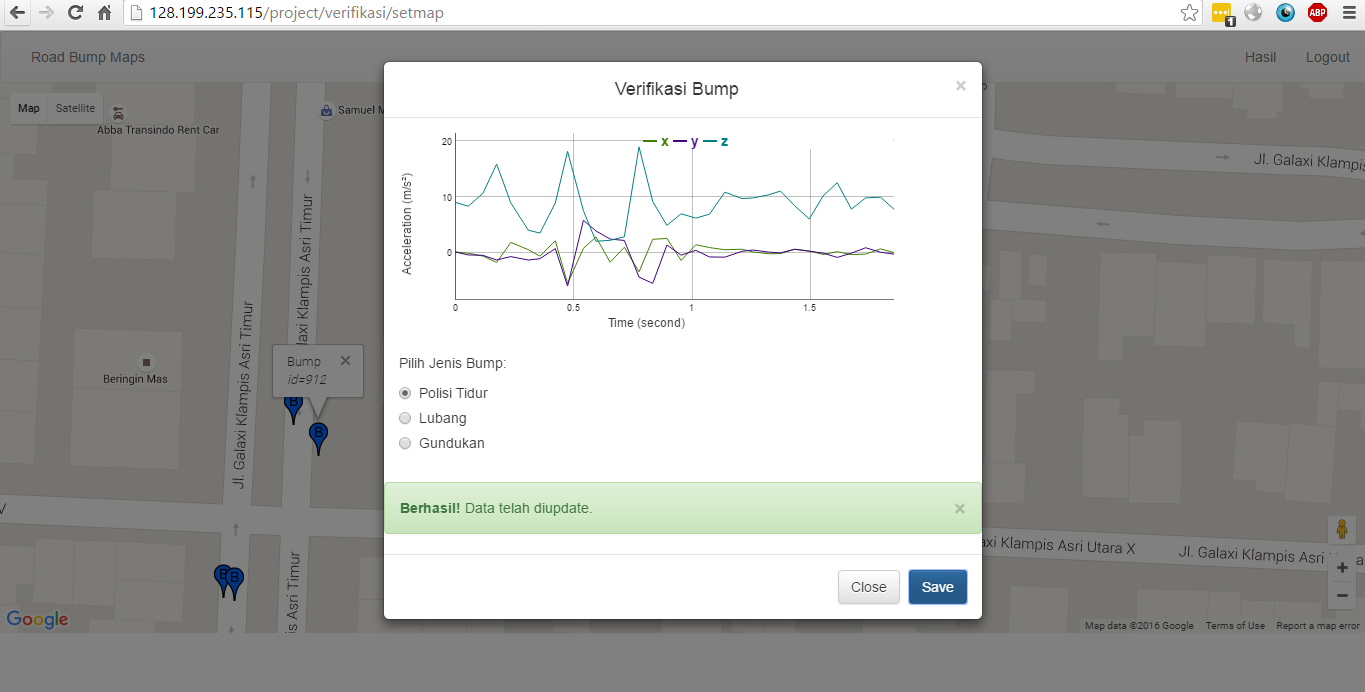
### Uji Coba Fungsionalitas Verifikasi Data

Uji coba verifikasi data merupakan tahap surveyor melakukan verifikasi kesesuaian data dengan hasil observasi lapangan. Hal pertama yang dilakukan adalah membuka halaman verifikasi pada browser, login, kemudian akan menampilkan peta digital, surveyor memilih *marker* yang akan diverifikasi dengan cara ­melakukan *double click* pada marker. *Modal* akan ditampilkan, kemudian pilih jenis *bump*, pilih tombol *save*. Tabel Skenario uji coba verifikasi dapat dilihat pada Tabel 5.3.

Tabel 5.3 Tabel Skenario Uji Coba Verifikasi Data

|  |  |
| --- | --- |
| **Nama Uji Coba** | **Verifikasi Data** |
| **Kondisi Awal** | Aplikasi belum dijalankan |
| **Aksi yang dilakukan** | 1. Login  2. *Double click* pada marker  3. Pilih jenis *bump,* save. |
| **Hasil yang diharapkan** | 1. Data yang diverifikasi diperbaharui |
| **Hasil yang diperoleh** | 1. Data yang diverifikasi diperbaharui |
| **Hasil Uji Coba** | Berhasil |

Pada Tabel 5.3 dapat dilihat verifikasi berhasil dilakukan ditunjukkan oleh dialog *alert* *success* yang muncul pada bagian bawah *Modal*.



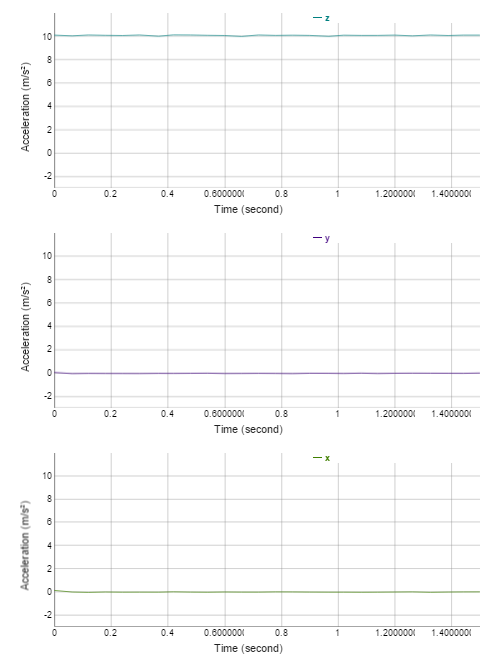
Gambar 5.4 Tampilan Halaman Verifikasi Data

## Analisa Data

Pada subbab ini membahas mengenai analisa data *training* pada berbagai kondisi. Data akselerometer yang dianalisis adalah saat kondisi normal (tidak bergerak), saat berkendara pada jalan normal (rata), dan saat melewati polisi tidur.

### Analisa Data pada Kondisi Normal

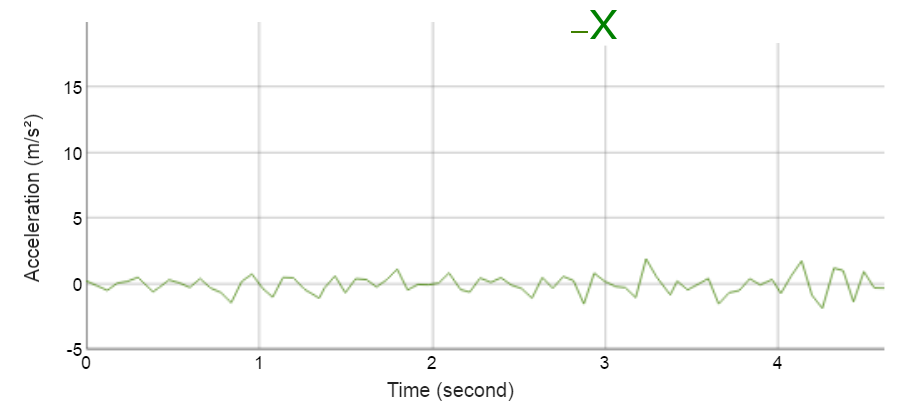
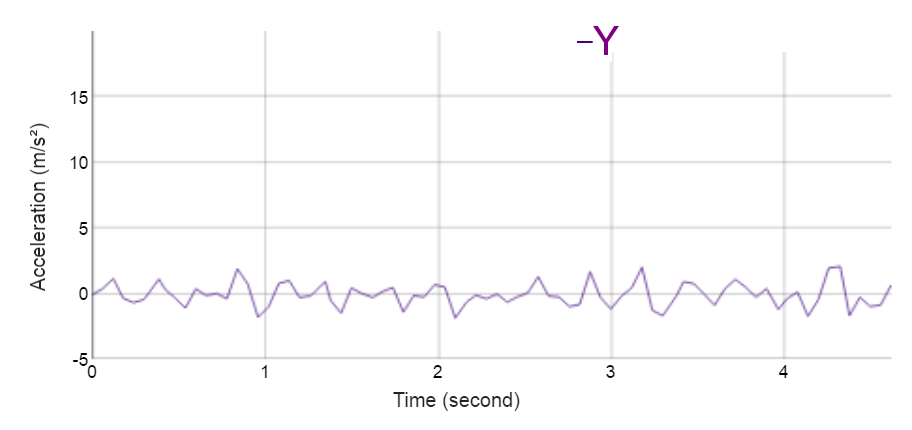
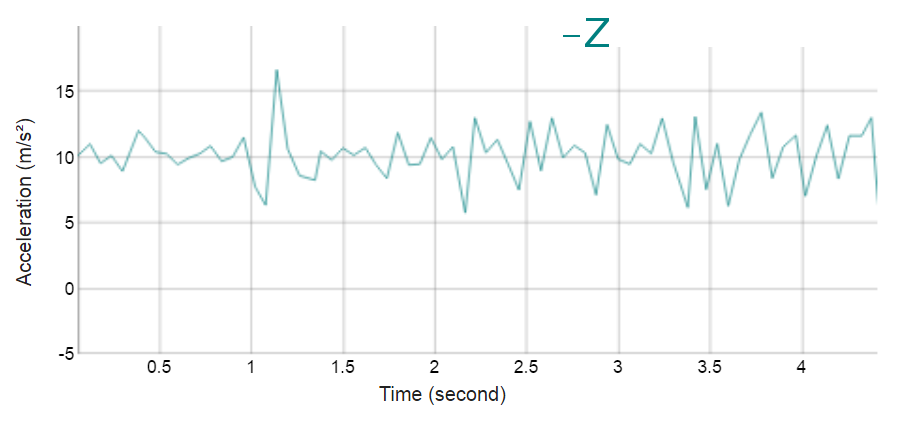
Data yang dianalisis adalah data akselerometer pada sumbu x, y dan z. Pada Gambar 5.5 dapat dilihat grafik nilai akselerometer saat *smartphone* tidak bergerak. Dari grafik tersebut dapat dilihat nilai akselerasi pada sumbu z adalah kurang lebih 10 m/s² karena pengaruh gaya gravitasi bumi, sedangkan pada sumbu x dan y adalah nol karena kondisi *smartphone* yang tidak mengalami pergerakan.



Gambar 5.5 Grafik Data Akselerometer Saat Tidak Bergerak

### Analisa Data pada Kondisi Jalan Rata

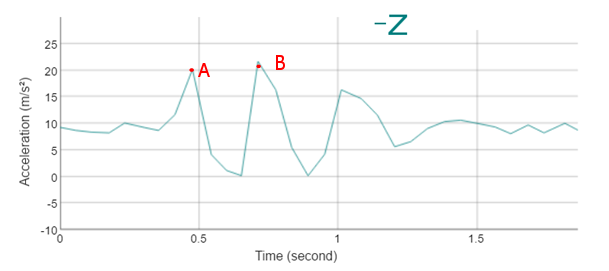
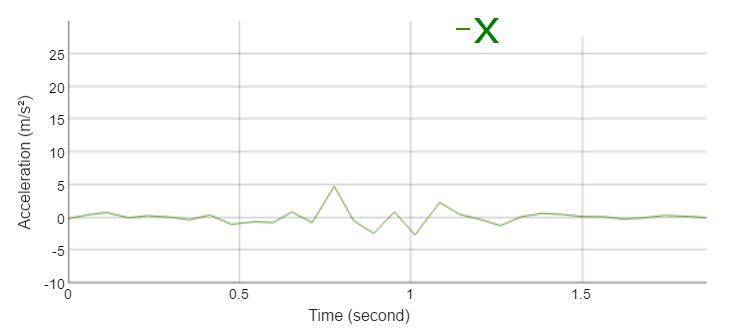
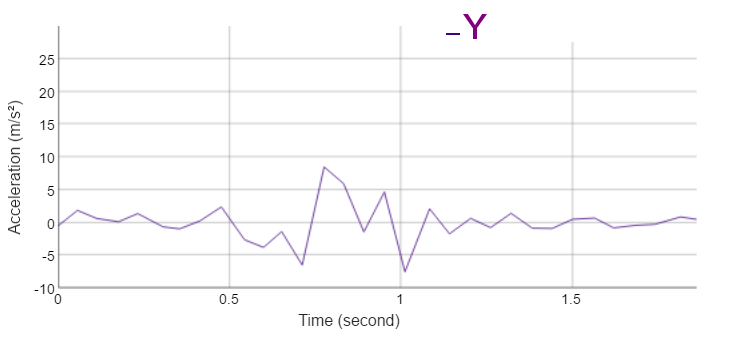
Kondisi akselerometer yang dianalisis adalah keadaan ketika pengguna berkendara melewati jalan yang normal atau rata. Pada Gambar 5.6 ditunjukkan nilai akselerometer pada sumbu x, y dan z di jalan yang rata (dalam kasus ini jalan paving). Pada sumbu z terdapat fluktuasi (naik turun) nilai yang disebabkan vibrasi atau getaran yang muncul karena adanya pergerakan pada kendaraan. Sedangkan pada sumbu x dan y tidak terlalu berpengaruh karena kecepatan kendaraan yang konstan bergerak lurus (sumbu y) sehingga tidak ada percepatan ke arah samping/belok (sumbu x).



Gambar 5.6 Grafik Data Akselerometer pada Jalan Rata

### Analisa Data pada Polisi Tidur

Data akselerometer yang dianalisa adalah nilai yang diamati ketika pengendara melewati polisi tidur. Cara pengambilan data adalah dengan menekan tombol *record* pada aplikasi ketika melewati polisi tidur. Pada Gambar 5.7 ditunjukkan nilai akselerometer pada saat melewati polisi tidur dengan kecepatan konstan 20 km/jam. Dapat dilihat nilai yang paling berpengaruh adalah nilai akselerasi pada sumbu z hal ini disebabkan saat melewati polisi tidur, kendaraan akan mengalami guncangan naik turun sebanyak dua kali. Guncangan pertama pada titik A Gambar 5.7 adalah guncangan akibat roda depan motor mengenai polisi tidur kemudian pada titik B adalah guncangan akibat roda belakang motor mengenai polisi tidur. Berdasarkan analisa ini maka pendeteksian *bump* dapat menggunakan algoritma *Z-Thresh* [4], yaitu mendeteksi terjadinya *bump* ketika nilai akselerometer pada sumbu z telah melewati nilai threshold tertentu. Nilai threshold diambil dari hasil pengamatan uji coba yang akan dibahas pada subbab selanjutnya.



Gambar 5.7 Grafik Data Akselerometer pada Polisi Tidur

## Pengujian Akurasi

Uji coba akurasi deteksi polisi tidur dilakukan untuk mengetahui akurasi pendeteksian menggunakan algoritma Z-*thresh* dan klasifikasi *decision tree.*

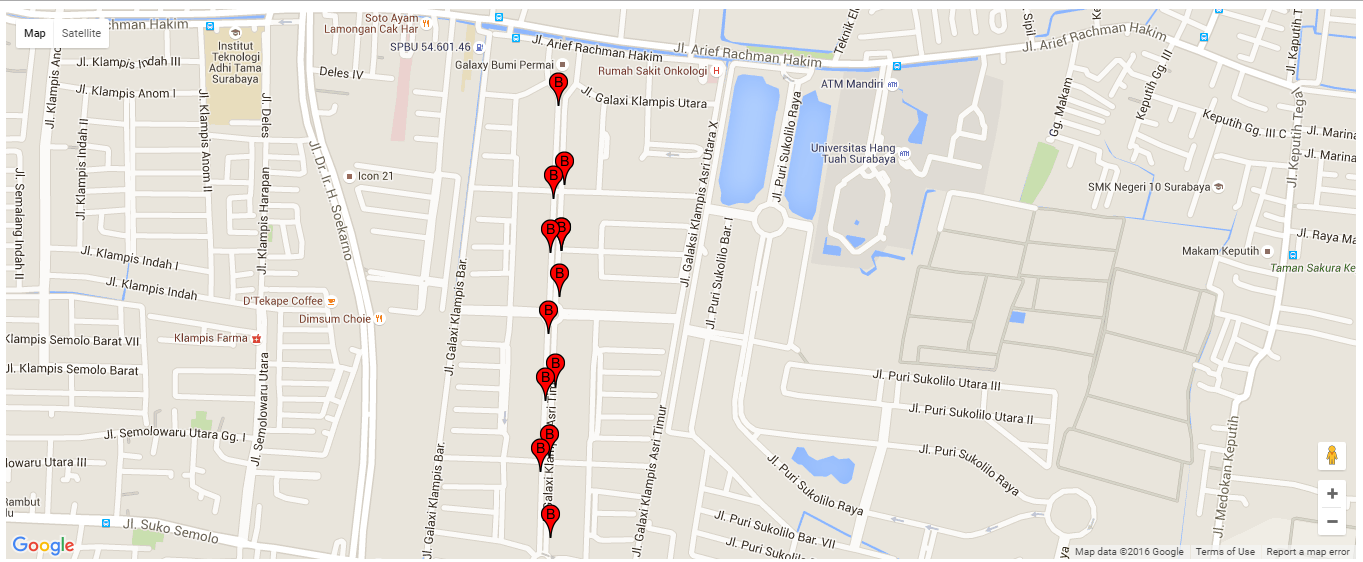
Berikut ini tata cara pengujian akurasi aplikasi dalam mendeteksi *bump*.

* Lokasi Pengujian: Perumahan Galaxi Klampis (2.61 km total tempuh).
* Jenis Jalan: *Paving.*
* Kecepatan Kendaraan: pelan (5-15 km/jam), sedang(15-30 km/jam), cepat (lebih dari 35 km/jam).
* Total Polisi Tidur: 12.
* Peletakan smartphone: kantong baju dan tas.
* Metode yang diuji: *Z-Thresh* (dengan threshold *batas\_atas*/*batas\_bawah*: 19/1, 18/2, 17/3) dan klasifikasi *decision tree*.

Pengujian ini dilakukan di Perumahan Galaxi Klampis karena memiliki jalan yang rata dan terdapat banyak polisi tidur. Setiap lokasi polisi tidur ditandai dengan menggunakan GPS dan datanya dikirimkan ke *server*.

Kecepatan pengujian dibagi menjadi 3 kondisi yaitu pelan, sedang dan cepat. Pada keadaan pelan kecepatan kendaraan adalah minimal 5 km/jam yaitu saat kendaraan mengurangi kecepatannya ketika akan menemui polisi tidur dan sekitar 10 km kecepatan maksimum ketika berjalan pada jalan normal. Sedangkan pada kondisi sedang, kecepatan minimal adalah 20 km/jam dan maksimum 25 km/jam. Pada kondisi cepat, kecepatan maksimum adalah kurang lebih 60 km/jam dan kecepatan minimum (ketika mendekati polisi tidur) adalah 35 km/jam.

Dari informasi tersebut dapat ditentukan *bump* (hasil deteksi dari aplikasi) mana yang benar-benar disebabkan polisi tidur (*true positive*) atau tidak (*false* *positive*). Apabila *bump* hasil deteksi berada didekat lokasi polisi tidur hasil observasi maka *bump* tersebut dianggap *true* *positives*.



Gambar 5.8 Lokasi Polisi Tidur untuk Pengujian

Pada Tabel 5.4 dapat dilihat skenario pengujian yang dilakukan. Pengujian dilakukan untuk mencari threshold-z yang tepat serta menguji metode klasifikasi *decision tree* apakah dapat meningkatkan akurasi atau tidak. Skenario pengujian juga memperhatikan peletakan smartphone untuk mengetahui pengaruh posisi smartphone terhadap aplikasi. Kecepatan yang diujikan berdasarkan batasan masalah adalah sekitar 20-40 km/jam.

Tabel 5.4 Tabel Skenario Pengujian

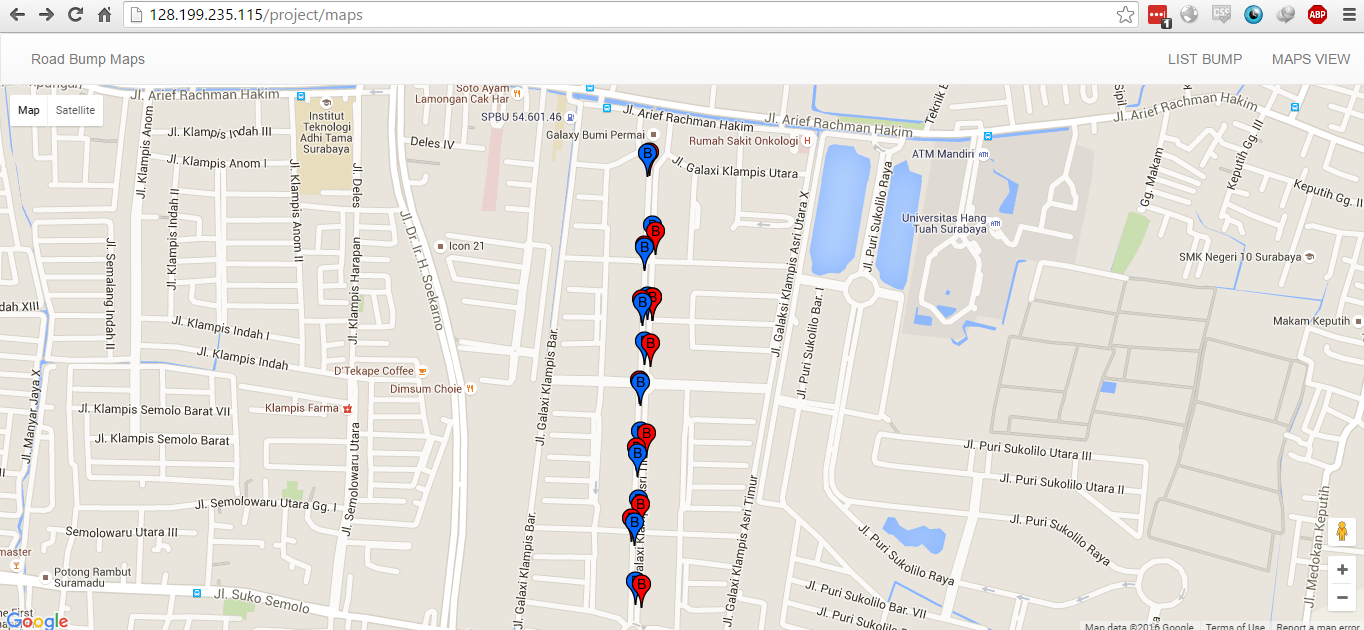
| **Kode Pengujian** | **Lokasi** | **Z-Thresh** | **Klasifikasi *Decision Tree*** | **Kecepatan** | **Peletakan Smartphone** |
| --- | --- | --- | --- | --- | --- |
| P-01 | Galaxi Klampis | 19/1 | - | sedang | kantong baju |
| P-02 | Galaxi Klampis | 19/1 | - | sedang | tas |
| P-03 | Galaxi Klampis | 19/1 | - | sedang | dipegang |
| P-04 | Galaxi Klampis | 18/2 | - | sedang | kantong baju |
| P-05 | Galaxi Klampis | 18/2 | - | sedang | tas |
| P-06 | Galaxi Klampis | 17/3 | - | pelan | kantong baju |
| P-07 | Galaxi Klampis | 17/3 | - | cepat | kantong baju |
| P-08 | Galaxi Klampis | 17/3 | - | sedang | kantong baju |
| P-09 | Galaxi Klampis | 18/2 | ✓ | sedang | kantong baju |
| P-10 | Galaxi Klampis | 18/2 | ✓ | sedang | tas |
| P-11 | Galaxi Klampis | 17/3 | ✓ | sedang | kantong baju |
| P-12 | Galaxi Klampis | 17/3 | ✓ | cepat | kantong baju |

Hasil uji coba dapat dilihat pada Tabel 5.5. Akurasi yang baik (presisi dan sensitivitas yang tinggi) dapat dilihat pada saat pengujian P-06, P-07, dan P-08 memiliki presisi dan sensitivitas mendekati 100%. Presisi yang paling rendah ditemui pada saat pengujian P-06. Sensitivitas paling rendah terjadi pada pengujian P-03. Pada pengujian yang menggunakan metode klasifikasi *decision tree*, pengujian P-09, P-10, dan P-11 memiliki akurasi yang lebih baik dibandingkan pengujian tanpa menggunakan metode klasifikasi *decision tree*.

Tabel 5.5 Hasil Uji Coba

| **Kode Pengujian** | **True Positives** | **False Positives** | **False Negatives** | **Presisi** | **Sensitivitas** |
| --- | --- | --- | --- | --- | --- |
| P-01 | 9 | 1 | 3 | 90.00% | 75.00% |
| P-02 | 8 | 3 | 4 | 72.73% | 66.67% |
| P-03 | 6 | 0 | 6 | 100.00% | 50.00% |
| P-04 | 12 | 1 | 0 | 92.31% | 100.00% |
| P-05 | 11 | 2 | 1 | 84.62% | 91.67% |
| P-06 | 7 | 5 | 5 | 58.33% | 58.33% |
| P-07 | 12 | 6 | 0 | 66.67% | 100% |
| P-08 | 11 | 1 | 0 | 91.67% | 100.00% |
| P-09 | 11 | 0 | 1 | 100.00% | 91.67% |
| P-10 | 12 | 0 | 0 | 100.00% | 100.00% |
| P-11 | 12 | 0 | 0 | 100.00% | 100.00% |
| P-12 | 12 | 0 | 0 | 100.00% | 100.00% |

Contoh hasil dari salah satu pengujian ada pada Gambar 5.9. Pada gambar tersebut (hasil dari pengujian P-010) dapat dilihat bahwa marker biru (*bump* hasil deteksi)yang berdekatan dengan marker merah (lokasi polisi tidur hasil observasi) dianggap sebagai bump yang berhasil terdeteksi sebagai *true positive* oleh sistem.



Gambar 5.9 Hasil Pengujian P-10 pada Peta Digital

## Evaluasi Pengujian

Berdasarkan hasil pengujian fungsionalitas dan pengujian akurasi maka pada subbab ini akan menjelaskan evaluasi pengujian seperti berikut.

### Evaluasi Pengujian Fungsionalitas

Hasil pengujian fungsionalitas dapat dilihat pada subbab sebelumnya. Berdasarkan Tabel 5.6 dapat dilihat hasil uji coba fungsi pada aplikasi dapat dijalankan dengan baik. Sehingga dapat ditarik kesimpulan bahwa fungsionalitas aplikasi telah bekerja sesuai dengan harapan dan menampilkan proses serta hasil yang benar.

Tabel 5.6 Rangkuman Hasil Pengujian

|  |  |
| --- | --- |
| **Nama** | **Hasil** |
| Pengujian melihat peta digital | Berhasil |
| Pengujian mengumpulkan data | Berhasil |
| Pengujian verifikasi data | Berhasil |

### Evaluasi Pengujian Akurasi

Berdasarkan hasil uji coba pada Tabel 5.5 dapat dilihat bahwa pengujian (P-01 sampai P-03) yang menggunakan threshold-z dengan nilai batas atas/bawah 19/1 memiliki akurasi yang rendah. Hal ini disebabkan nilai akselerasi pada sumbu z yang tidak melewati batas threshold ketika pengendara melewati polisi tidur sehingga terjadi *false negative* atau polisi tidur tidak terdeteksi sebagai *bump.* Akurasi yang paling baik dapat ditemui pada pengujian dengan *threshold*-z sebesar 18/2 dan 17/3 memiliki akurasi (presisi dan sensitivitas) rata-rata di atas 85%.

Pengujian akurasi dengan posisi *smartphone* di kantong baju dan tas tidak memiliki perbedaan yang signifikan. Namun untuk pengujian pada *smartphone* dipegang (P-03) memiliki *false* *negative* yang besar. Sensitivitas aplikasi ketika dipegang berkurang karena posisi tangan yang lebih stabil (mempertahankan titik seimbang) saat guncangan terjadi sehingga vibrasi tidak sebesar dibandingkan pengujian aplikasi jika *smartphone* diletakkan di tas atau kantong baju.

Pada pengujian untuk kecepatan pelan (P-06) memiliki sensivitas yang rendah karena pada kecepatan rendah nilai akselerasi yang dihasilkan ketika mengalami guncangan dengan polisi tidur lebih kecil dibandingkan pada kecepatan sedang atau cepat. Hal ini disebabkan nilai akselerometer yang terjadi pada saat guncangan tidak memenuhi threshold pada sumbu z sehingga banyak polisi tidur yang tidak terdeteksi. Presisi juga rendah karena disebabkan oleh jalan yang kurang rata.

Pada pengujian dengan kecepatan cepat (P-07) memiliki sensitivitas yang tinggi namun presisi yang rendah. Hal ini disebabkan banyak *false positive* yang dihasilkan akibat vibrasi/guncangan yang besar pada jalan yang kurang rata akibat kecepatan kendaraan yang tinggi. Sensitivitas tinggi diakibatkan guncangan yang dihasilkan yang melewati threshold pada setiap polisi tidur yang dilewati.

Pada pengujian P-04, P-05 dan P-08 tidak menggunakan metode klasifikasi *decision tree* memiliki false positive yang lebih tinggi. False positive saat pengujian dapat disebabkan adanya permukaan jalan yang tidak rata (gundukan kecil) sehingga mempengaruhi kendaraan dan aplikasi mendeteksi guncangan tersebut sebagai *bump.* Dengan adanya klasifikasi *decision tree* yang membedakan guncangan akibat polisi tidur dan guncangan yang tidak rata. Sehingga pada pengujian P-09, P-10, P-11 dan P-12 akurasi dapat meningkat karena sebagian besar *false positive* berhasil dikenali menggunakan metode klasifikasi *decision tree*. Namun dalam kasus *false* *positive* yang disebabkan lubang yang besar maupun gundukan jalan (bukan polisi tidur) sulit untuk dibedakan karena memiliki vibrasi/guncangan yang serupa dengan polisi tidur. Pada P-12 (kecepatan tinggi) merupakan data pengujian P-07 yang ditambahkan metode klasifikasi *decision tree* danhasilnyaadalah akurasi meningkat menjadi 100% karena *false-positive* yang berhasil dideteksi klasifier *decision tree.*

# BAB VI PENUTUP

Pada bab ini akan dibahas kesimpulan yang diperoleh berdasarkan hasil uji coba aplikasi untuk menjawab rumusan masalah dan saran mengenai pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

## Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, maka dapat disimpulkan sebagai berikut.

1. Sensor pada *smartphone* Android dapat digunakan untuk mendeteksi alat pembatas kecepatan.
2. Peta digital dapat menampilkan lokasi alat pembatas kecepatan dan mengestimasi posisinya menggunakan metode *clustering* pada titik-titik koordinat yang berdekatan*.*
3. Berdasarkan hasil uji coba maka dapat diambil kesimpulan bahwa peletakkan posisi *smartphone* didalam kantong atau tas memiliki akurasi yang lebih baik dibandingkan jika *smartphone* dipegang.
4. Kecepatan ideal berdasarkan uji coba adalah pada kecepatan sedang atau cepat.
5. Algoritma pendeteksi *bump* yang digunakan adalah *Z-Thresh* dengan nilai threshold dengan akurasi terbaik sebesar 17 pada batas atas dan 3 pada batas bawah. Untuk mengurangi *false* *positives* yaitu deteksi *bump* akibat anomali jalan (jalan sedikit bergelombang, tidak rata, lubang kecil, pembatas jalan) menggunakan klasifikasi *decision tree.* Dari kombinasi kedua metode tersebut diperoleh nilai akurasi di atas 91%.

## Saran

Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan. Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang.

1. Pada penelitian selanjutnya dapat dilakukan pada kendaraan mobil dan pada *range* kecepatan yang lebih luas.
2. Aplikasi ini juga dapat digunakan sebagai *services* pada aplikasi (*crowdsourcing*) seperti Waze atau Google Maps dalam pemetaan polisi tidur maupun anomali jalan sehingga data yang dihasilkan lebih banyak dan cepat.

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden and H. Balakrishnan, "The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring," *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008),* pp. 29-39, 2008. |
| [2] | K. Chen, G. Tan, M. Lu and J. Wu, "CRSM: a practical crowdsourcing-based road surface monitoring system," *Wireless Networks,* pp. 765-779, 2016. |
| [3] | R. Bhoraskar, N. Vankadhara, B. Raman and P. Kulkarni, "Wolverine: Traffic and road condition estimation using smartphone sensors," *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012),* pp. 1 - 6, 2012. |
| [4] | A. Mednis, G. Strazdins , R. Zviedris and G. Kanonirs , "Real time pothole detection using Android smartphones with accelerometers," *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS),* pp. 1-6, 2011. |
| [5] | "Polisi Tidur," [Online]. Available: https://id.wikipedia.org/wiki/Polisi\_tidur. [Accessed April 2016]. |
| [6] | "Android\_(operating\_system)," Android, [Online]. Available: https://en.wikipedia.org/wiki/Android\_(operating\_system). [Accessed May 2016]. |
| [7] | "Android\_fundamental," Google, [Online]. Available: https://developer.android.com/guide/components/fundamentals.html. [Accessed May 2016]. |
| [8] | "LocationListener," Google, [Online]. Available: https://developer.android.com/reference/android/location/LocationListener.html. [Accessed May 2016]. |
| [9] | "Volley Library," Google, [Online]. Available: https://developer.android.com/training/volley/index.html. [Accessed May 2016]. |
| [10] | K. Douglas and S. Douglas, "PostgreSQL, Second Edition," in *PostgreSQL*, Sams, 2005. |
| [11] | A. El-Rabanny, "Introduction to GPS The Global Positioning System," Boston, Artech House, 2002. |
| [12] | "PHP," PHP, [Online]. Available: https://en.wikipedia.org/wiki/PHP. [Accessed May 2016]. |
| [13] | [Online]. Available: https://en.wikipedia.org/wiki/Python\_(programming\_language). [Accessed May 2016]. |
| [14] | "Google Maps API," Google, [Online]. Available: https://developers.google.com/maps/web-services/overview#WebServices. [Accessed May 2016]. |
| [15] | "BIOSTALL," Biostall, [Online]. Available: http://biostall.com/demos/google-maps-v3-api-codeigniter-library/. [Accessed May 2016]. |
| [16] | "CodeIgniter," EllisLab, [Online]. Available: https://en.wikipedia.org/wiki/CodeIgniter. [Accessed May 2016]. |
| [17] | "Slim," Slim Framework Team, [Online]. Available: http://www.slimframework.com/. [Accessed May 2016]. |
| [18] | . L. Breiman, J. Friedman, R. Olshen and C. J. Stone, Classification and Regression Tree, Pacific California: Wadsworth & Brooks/Cole Advanced Books & Software, 1984. |
| [19] | "Rotation Matrix," [Online]. Available: https://developer.android.com/reference/android/hardware/SensorManager.html#getRotationMatrix(float[], float[], float[], float[]). [Accessed April 2016]. |
| [20] | "DecisionTreeClassifier," scikit-learn, [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed 30 May 2016]. |
| [21] | "Volley," [Online]. Available: https://developer.android.com/training/volley/index.html. [Accessed 24 May 2016]. |

*[Halaman ini sengaja dikosongkan]*

# BIODATA PENULIS

 Penulis, **Otniel Yehezkiel Bornok Hutabarat**, lahir di Batam, 12 April 1994. Penulis menempuh pendidikan sekolah dasar di SD Kristen Kalam Kudus, Batam. Melanjutkan pendidikan sekolah menengah pertama di SMP Kristen Kalam Kudus Batam dan selanjutnya di SMA Katolik Yos Sudarso Batam. Selanjutnya penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika, Fakultas Teknologi dan Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif menjadi administrator Laboratorium Pemrograman Teknik Informatika, asisten praktikum Dasar Pemrograman, asisten PIKTI dan aktif dalam organisasi tingkat jurusan dan UKM Musik.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Algoritma Pemrograman (AP) dan memiliki ketertarikan dalam bidang *Web and* *Mobile Application Development, Music* dan *Startup.* Penulis dapat dihubungi melalui email: braincreativelife@gmail.com