

**PENGEMBANGAN SISTEM PELAPORAN KERUSAKAN JALAN  
OTOMATIS BERBASIS SISTEM EMBEDDED**

**SKRIPSI**

Untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer

Disusun oleh:  
Hanif Irfan Syah  
NIM: 155150201111001



PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BRAWIJAYA  
MALANG  
2019

## PENGESAHAN

PENGEMBANGAN SISTEM PELAPORAN KERUSAKAN JALAN OTOMATIS BERBASIS  
SISTEM EMBEDDED

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer

Disusun Oleh:  
Hanif Irfan Syah  
NIM: 155150201111001

Skripsi ini telah diuji dan dinyatakan lulus pada  
2 Januari 2019

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I



Fajar Pradana, S.ST, M.Eng  
NIP. 19871121 201504 1 004

Dosen Pembimbing II



Bayu Priyambadha, S.Kom, M.Kom  
NIP. 19820909 200812 1 000

Mengetahui

Ketua Jurusan Teknik Informatika



Tri Astoto Kurniawan, S.T, M.T, Ph.D  
NIP. 19710518 200312 1 001

## PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 02 Januari 2019



Hanif Irfan Syah

NIM: 155150201111001

## KATA PENGANTAR

Dengan menyebut nama Allah Yang Maha Pengasih dan Maha Penyayang. Segala puji bagi Allah karena atas berkah, rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi yang berjudul “Pengembangan Sistem Pelaporan Kerusakan Jalan Otomatis Berbasis Sistem Embedded” dengan baik.

Penelitian skripsi ini mendapat banyak bantuan dari berbagai pihak yang terus memberikan bimbingan, saran, dukungan, motivasi maupun doa. Oleh karena itu, penulis mengucapkan terima kasih kepada :

1. Fajar Pradana, S.ST, M.Eng sebagai Dosen Pembimbing pertama, atas segala bimbingan, waktu, kritik dan saran yang diberikan kepada penulis selama penelitian skripsi.
2. Bayu Priyambadha, S.Kom, M.Kom sebagai Dosen Pembimbing kedua, atas segala bimbingan, waktu, kritik dan saran yang diberikan kepada penulis selama penelitian skripsi.
3. Eko Sakti Pramukantoro, S.Kom, M.Kom sebagai Dosen Pembimbing Akademik yang telah memberi dukungan moril maupun masukan selama masa perkuliahan.
4. Seluruh Bapak dan Ibu dosen Fakultas Ilmu Komputer Universitas Brawijaya atas segala bimbingan dan ilmu yang diberikan kepada penulis.
5. Semua pihak yang telah membantu kelancaran penelitian skripsi dan memberikan doa yang tidak dapat disebutkan penulis satu persatu.

Penulis menyadari bahwa penelitian ini masih memiliki banyak kesalahan dan kekurangan yang disebabkan oleh keterbatasan ilmu. Oleh karena itu, penulis sangat membutuhkan kritik dan saran kepada pembaca sebagai perbaikan untuk menyempurnakan skripsi ini pada penelitian selanjutnya. Penulis berharap skripsi ini dapat bermanfaat bagi diri penulis sendiri maupun bagi seluruh pihak.

Malang, 02 Januari 2019

Penulis

hanifirfansyah05@gmail.com

## ABSTRAK

Pembangunan infrastruktur jalan merupakan salah satu program yang sedang digiatkan oleh kementerian Pekerjaan Umum dan Perumahan Rakyat. Mengingat banyaknya manfaat yang mampu dirasakan oleh masyarakat maupun pemerintah. Namun selain pembangunan, juga diperlukan perawatan dan perbaikan terhadap jalan raya tersebut. Kerusakan jalan dapat menyebabkan kecelakaan luka ringan hingga kematian. Pengambilan keputusan perbaikan dan penanganan kerusakan jalan sering tidak tepat sasaran, dikarenakan kesalahan dalam menentukan prioritas yang diakibatkan oleh keterbatasan petugas lapangan. Selain itu penanganan kerusakan jalan terkesan lambat dikarenakan rekapitulasi data yang diperoleh masih menggunakan dokumen kertas, yang mengakibatkan penanganan kerusakan jalan menjadi lama dan kerusakan jalan tidak segera diperbaiki. Pada penelitian ini, sistem yang dikembangkan menggunakan mikrokomputer Raspberry PI dengan menerapkan modul GPS, sensor getar dan sensor ultrasonik yang ditanamkan pada kendaraan bermotor. Sensor ini akan memberikan informasi terkait tingkat kerusakan jalan dan lokasi kerusakan jalan dengan bantuan API Google Maps secara *realtime* kepada petugas terkait. Sehingga pengambilan keputusan dan penanganan perbaikan kerusakan jalan dapat dilakukan dengan tepat sasaran dan segera diperbaiki.

**Kata Kunci:** *kerusakan jalan, raspberry PI, sensor getar, sensor ultrasonik, modul GPS*

## ABSTRACT

*The construction of road infrastructure is one program that is being activated by the Ministry of Public Works and Public Housing. Given the many benefits that can be felt by the community and government. But in addition to development, maintenance and repairs are also needed on the highway. Road damage can cause minor injuries to death. Decision making for repair and handling of road damage is often not on target, due to errors in determining priorities caused by the limitations of field officers. Besides that, the handling of road damage seems slow because the recapitulation of the data obtained still uses paper documents, which results in the handling of road damage being prolonged and road damage not being corrected immediately. In this study, a system developed using Raspberry PI microcomputer by applying the mosul GPS, vibration sensor and ultrasonic sensor implanted in motorized vehicles. This sensor will provide information regarding the level of road damage and the location of road damage with the help of the Google Maps API in real time to the relevant officers. So that decision making and handling repair of road damage can be done on target and immediately corrected.*

**Keywords:** *road damage, raspberry PI, vibration sensor, ultrasonic sensor, GPS module*

## DAFTAR ISI

PENGESAHAN .....	<b>Error! Bookmark not defined.</b>
PERNYATAAN ORISINALITAS .....	iii
KATA PENGANTAR.....	iv
ABSTRAK.....	v
ABSTRACT .....	vi
DAFTAR ISI.....	vii
DAFTAR TABEL.....	x
DAFTAR GAMBAR.....	xii
DAFTAR LAMPIRAN .....	xiii
BAB 1 PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan .....	2
1.4 Manfaat.....	3
1.5 Batasan Masalah.....	3
1.6 Sistematika Pembahasan .....	3
BAB 2 LANDASAN KEPUSTAKAAN .....	5
2.1 Kajian Pustaka .....	5
2.2 Rekayasa Perangkat Lunak .....	6
2.3 Pengembangan Perangkat Lunak .....	6
2.3.1 Siklus Pengembangan Perangkat Lunak (SDLC) .....	7
2.3.2 <i>Object Oriented Programming (OOP)</i> .....	8
2.3.3 Pemodelan Perangkat Lunak .....	8
2.3.4 Strategi Pengujian .....	12
2.4 Perkerasan Lentur.....	13
2.4.1 Pengertian Perkerasan Lentur .....	13
2.4.2 Kerusakan Perkerasan Lentur .....	13
2.4.3 Jenis Kerusakan Perkerasan Lentur.....	13
2.4.4 Lubang Jalan.....	13
2.5 Rekayasa <i>Embedded System</i> .....	14
2.6 <i>Raspberry Pi</i> .....	15

2.7 Modul Sensor Getar SW-18020 .....	16
2.8 Modul Sensor Ultrasonik HC-SR04.....	17
2.9 Modul <i>Raspberry GPS Add-On V2.0</i> .....	17
2.10 <i>Framework CodeIgniter</i> .....	18
2.11 <i>Realtime Database Firebase</i> .....	19
2.12 Bahasa Pemrograman <i>Python</i> .....	20
2.13 Bahasa Pemrograman <i>PHP</i> .....	21
<b>BAB 3 METODOLOGI</b> .....	<b>22</b>
3.1 Tahapan Penelitian .....	22
3.2 Studi Literatur .....	22
3.3 Rekayasa Kebutuhan Sistem .....	23
3.4 Perancangan dan Implementasi .....	23
3.5 Pengujian Sistem.....	23
3.6 Kesimpulan dan Saran .....	24
<b>BAB 4 REKAYASA KEBUTUHAN</b> .....	<b>25</b>
4.1 Deskripsi Umum Sistem .....	26
4.2 Analisis Kebutuhan .....	27
4.2.1 Identifikasi Aktor .....	28
4.2.2 Aturan Penomoran Kode Kebutuhan Perangkat Lunak.....	29
4.2.3 Analisis Kebutuhan Fungsional Sistem.....	30
4.2.4 Analisis Kebutuhan Non-Fungsional Sistem.....	31
4.3 Pemodelan Kebutuhan .....	31
4.3.1 Use Case Diagram .....	31
4.3.2 <i>Use Case Scenario</i> .....	33
<b>BAB 5 PERANCANGAN DAN IMPLEMENTASI</b> .....	<b>39</b>
5.1 Perancangan .....	39
5.1.1 <i>Sequence Diagram</i> .....	40
5.1.2 <i>Class Diagram</i> .....	42
5.1.3 Perancangan Komponen .....	43
5.1.4 Perancangan Antarmuka.....	45
5.1.5 Perancangan Perangkat Keras .....	45
5.1.6 Perancangan Data .....	47

5.2 Implementasi .....	48
5.2.1 Spesifikasi Sistem .....	48
5.2.2 Implementasi Kode Program .....	48
5.2.3 Implementasi Antarmuka .....	51
BAB 6 PENGUJIAN SISTEM .....	53
6.1 Pengujian Unit.....	53
6.1.1 Pengujian unit <i>method get_distance</i> .....	54
6.1.2 Pengujian unit <i>method callback</i> .....	55
6.1.3 Pengujian unit <i>method kategori</i> .....	58
6.2 Pengujian Integrasi .....	59
6.2.1 Pengujian Integrasi <i>Method callback() Class Vibration</i> .....	60
6.3 Pengujian Validasi .....	63
6.3.1 Pengujian validasi menangkap data getar .....	63
6.3.2 Pengujian Validasi Menangkap data jarak .....	64
6.3.3 Pengujian Validasi Menangkap data koordinat .....	65
6.3.4 Pengujian Validasi Mengirim data .....	66
6.3.5 Pengujian Validasi Klasifikasi kerusakan .....	67
6.3.6 Pengujian Validasi Menampilkan data .....	68
6.3.7 Pengujian Validasi Mencetak laporan.....	68
6.3.8 Pengujian Validasi Menampilkan lokasi.....	70
6.3.9 Pengujian validasi kebenaran data ( <i>correctness</i> ).....	70
BAB 7 PENUTUP .....	72
7.1 Kesimpulan.....	72
7.2 Saran .....	72
DAFTAR PUSTAKA.....	73
LAMPIRAN A WAWANCARA Dinas Pekerjaan Umum dan Penataan Ruang (PUPR) Kota Malang .....	75
LAMPIRAN B IMPLEMENTASI SISTEM .....	76
LAMPIRAN C DOKUMENTASI KEGIATAN.....	78

## DAFTAR TABEL

Tabel 2.1 Notasi <i>use case diagram</i> .....	9
Tabel 2.2 Notasi <i>sequence diagram</i> .....	10
Tabel 2.3 Notasi <i>class diagram</i> .....	11
Tabel 2.4 Tingkat kerusakan lubang jalan .....	14
Tabel 2.5 Spesifikasi Raspberry PI 3 Model B v1.2.....	15
Tabel 2.6 Spesifikasi Raspberry PI GPS Add-on.....	18
Tabel 4.1 Analisis dan spesifikasi kebutuhan .....	27
Tabel 4.2 Daftar Aktor .....	28
Tabel 4.3 Spesifikasi kebutuhan fungsional sistem.....	30
Tabel 4.4 Spesifikasi kebutuhan non-fungsional sistem .....	31
Tabel 4.5 Use Case Scenario menangkap data getar .....	33
Tabel 4.6 Use Case Scenario menangkap data jarak .....	33
Tabel 4.7 Use Case Scenario menangkap data koordinat.....	34
Tabel 4.8 <i>Use Case Scenario</i> mengirim data .....	35
Tabel 4.9 <i>Use Case Scenario</i> klasifikasi kerusakan.....	35
Tabel 4.10 <i>Use Case Scenario</i> menampilkan data .....	36
Tabel 4.11 <i>Use Case Scenario</i> mencetak laporan .....	37
Tabel 4.12 <i>Use Case Scenario</i> menampilkan lokasi .....	37
Tabel 5.1 Perancangan komponen <i>method get_distance</i> .....	43
Tabel 5.2 Perancangan komponen <i>method callback</i> .....	44
Tabel 5.3 Perancangan komponen <i>method kategori</i> .....	44
Tabel 5.4 Pin GPIO.....	47
Tabel 5.5 Spesifikasi perangkat keras .....	48
Tabel 5.6 Spesifikasi perangkat lunak .....	48
Tabel 5.7 Kode program <i>method get_distance()</i> .....	48
Tabel 5.8 Penjelasan kode program <i>method get_distance()</i> .....	49
Tabel 5.9 Kode program <i>method callback()</i> .....	49
Tabel 5.10 Penjelasan kode program <i>method callback()</i> .....	50
Tabel 5.11 Kode program <i>method kategori()</i> .....	50
Tabel 5.12 Penjelasan kode program <i>method kategori()</i> .....	51
Tabel 6.1 Perancangan komponen <i>method get_distance</i> .....	54

Tabel 6.2 Hasil pengujian method <i>get_distance</i> .....	55
Tabel 6.3 Perancangan komponen <i>method callback</i> .....	55
Tabel 6.4 Hasil pengujian method <i>callback</i> .....	56
Tabel 6.5 Perancangan komponen <i>method kategori</i> .....	58
Tabel 6.6 Hasil pengujian method kategori .....	59
Tabel 6.7 Pengujian integrasi method <i>callback</i> .....	60
Tabel 6.8 Hasil pengujian method <i>callback</i> .....	61
Tabel 6.9 Pengujian validasi menangkap data getar .....	63
Tabel 6.10 Pengujian validasi menangkap data getar jalur alternatif .....	63
Tabel 6.11 Pengujian validasi menangkap data jarak .....	64
Tabel 6.12 Pengujian validasi menangkap data jarak jalur alternatif.....	65
Tabel 6.13 Pengujian validasi menangkap data koordinat .....	65
Tabel 6.14 Pengujian validasi menangkap data koordinat jalur alternatif.....	66
Tabel 6.15 Pengujian validasi mengirim data .....	66
Tabel 6.16 Pengujian validasi mengirim data jalur alternatif .....	67
Tabel 6.17 Pengujian validasi klasifikasi kerusakan .....	67
Tabel 6.18 Pengujian validasi menampilkan data.....	68
Tabel 6.19 Pengujian validasi mencetak laporan.....	69
Tabel 6.20 Pengujian validasi mencetak laporan jalur alternatif .....	69
Tabel 6.21 Pengujian validasi menampilkan lokasi.....	70
Tabel 6.22 Pengujian validasi kebenaran data.....	70



## DAFTAR GAMBAR

Gambar 2.1 Waterfall Model .....	7
Gambar 2.2 Raspberry PI Model B v1.2 .....	16
Gambar 2.3 Modul Sensor Getar SW-420.....	16
Gambar 2.4 Modul Sensor Ultrasonik HC-SR04.....	17
Gambar 2.5 Raspberry PI GPS Add-on .....	18
Gambar 3.1 Blog Diagram Metodologi Penelitian .....	22
Gambar 4.1 Diagram Pohon Rekayasa Kebutuhan .....	25
Gambar 4.2 Deskripsi Sistem .....	26
Gambar 4.3 Aturan Penomoran.....	29
Gambar 4.4 Use Case Diagram Sistem Embedded .....	32
Gambar 4.5 Use Case Diagram Website .....	32
Gambar 5.1 Diagram pohon perancangan dan implementasi.....	39
Gambar 5.2 Sequence Diagram menangkap data jarak .....	40
Gambar 5.3 Sequence Diagram mengirim data.....	41
Gambar 5.4 Sequence Diagram klasifikasi kategori.....	41
Gambar 5.5 Rancangan Class Diagram Sistem Embedded .....	42
Gambar 5.6 Rancangan Class Diagram Website .....	43
Gambar 5.7 Rancangan Antarmuka Halaman Website .....	46
Gambar 5.8 Rancangan Perangkat Keras.....	47
Gambar 5.9 Implementasi Antarmuka Website .....	52
Gambar 6.1 Diagram Pohon Rekayasa Kebutuhan .....	53
Gambar 6.2 Flow graph method get_distance .....	54
Gambar 6.3 Flow graph method callback .....	57
Gambar 6.4 Flow graph method kategori.....	59
Gambar 6.5 Flow graph method callback .....	62

## DAFTAR LAMPIRAN

LAMPIRAN A WAWANCARA.....	75
LAMPIRAN B IMPLEMENTASI SISTEM .....	76
LAMPIRAN C DOKUMENTASI KEGIATAN.....	78



## BAB 1 PENDAHULUAN

### 1.1 Latar belakang

Keberadaan infrastruktur berupa jalan raya memberikan banyak manfaat bagi masyarakat maupun pemerintah. Manfaat yang diberikan diantaranya memudahkan mobilitas dan aksesibilitas baik kegiatan sosial dan ekonomi (Bernad and Syafaruddin, 2017). Undang-undang Republik Indonesia No. 38 tahun 2004 tentang prasarana jalan menyebutkan bahwa jalan memiliki peranan yang penting dalam mewujudkan perkembangan kehidupan bangsa. Karena manfaat yang begitu besar dan peranan yang sangat penting, menjadikan perhatian dan pertimbangan khusus oleh pemerintah saat ini. Salah satunya dengan program Rencana Strategis (Renstra) Kementerian Pekerjaan Umum dan Perumahan Rakyat tahun 2015-2019. Salah satu program yang digagas adalah dengan pembangunan jalan raya sejauh 1.200 km (Umum and Perumahan, 2019).

Selain pembangunan jalan baru, pemerintah juga perlu melakukan perawatan dan perbaikan terhadap jalan raya. Perawatan tersebut berfungsi untuk mencegah dan memperbaiki kerusakan jalan raya (M et al., 2016). Kerusakan pada jalan raya dapat berupa retak, distorsi, cacat permukaan, pengausan, kegemukan dan penurunan pada bekas penanaman utilitas. Umumnya kerusakan yang timbul disebabkan oleh lebih dari satu faktor. Faktor-faktor kerusakan tersebut disebabkan oleh adanya peningkatan beban lalu lintas, sistem drainase yang tidak baik, iklim suhu udara dan curah hujan serta kondisi tanah dasar yang tidak stabil (Sukirman, 1999).

Adanya kerusakan jalan dapat menyebabkan mobilitas barang dan aksesibilitas untuk menjangkau suatu daerah menjadi terganggu. Dampak lain yang dihadapi akibat kerusakan jalan yaitu menghambat laju perekonomian dan perdagangan masyarakat. Selain faktor dampak ekonomi, kerusakan jalan juga mengakibatkan ketidaknyamanan pengguna jalan serta mengganggu kelancaran arus lalu lintas (Bernad and Syafaruddin, 2017). Kerusakan jalan raya dapat menyebabkan kecelakaan dari luka ringan hingga kematian. Pada tahun 2017 tercatat kecelakaan lalu lintas terjadi sebanyak 2.857 kejadian, dan 622 orang diantaranya meninggal dunia. Berdasarkan kasus tersebut tercatat 159 kecelakaan diakibatkan oleh jalan rusak dan berlubang (Aris, 2017).

Pelaporan kerusakan jalan dilakukan oleh masyarakat maupun instansi kepada Dinas Pekerjaan Umum dan Penataan Ruang. Hal ini menjadi tugas berat bagi dinas terkait untuk mengelola seluruh data kerusakan jalan yang ada di wilayahnya. Penanganan perbaikan jalan masih dilakukan secara manual dengan membandingkan data hasil survei lapangan oleh petugas. Data hasil survei diperoleh dengan cara manual yaitu dengan menggunakan dokumen kertas yang menyebabkan rekapitulasi data berjalan lambat. Lambatnya data yang diperoleh dan banyaknya dokumen yang harus dibandingkan mengakibatkan penanganan jalan yang rusak menjadi lama dan kerusakan jalan tidak segera diperbaiki (Utama, 2013).

Pengambilan keputusan yang didasarkan pada data hasil survei menjadi permasalahan tersendiri. Hasil data terhadap suatu kerusakan sangat dipengaruhi oleh pengamat atau petugas di lapangan. Sehingga pengamat harus menguasai seluruh jenis dan penyebab kerusakan jalan yang ada di lapangan untuk menentukan kategori kerusakan dan prioritas penanganannya. Keterbatasan yang dialami petugas karena subjektivitas ini memberikan dampak terhadap keputusan perbaikan dan penanganan kerusakan, dimana seringkali tidak tepat sasaran. Sedangkan prioritas penanganan tersebut harus dilakukan secara tepat dan sesuai dengan kepentingan (Sukirman, 1999).

Dalam sebuah wawancara (Lampiran A.1) menunjukkan permasalahan terkait proses pelaporan kerusakan jalan di Dinas Pekerjaan Umum dan Penataan Ruang (PUPR) Kota Malang. Pelaporan kerusakan jalan yang berajalan sangat bergantung pada laporan dari berbagai media online maupun cetak seperti website Lapor Kementerian Kominfo dan melalui sosial media. Laporan kerusakan jalan juga bisa berasal dari masyarakat yang mengadu langsung ke dinas maupun dari petugas dinas sendiriyang tanpa sengaja melihat kerusakan jalan. Hal ini membuat petugas sangat bergantung pada laporan yang dilakukan oleh berbagai pihak tersebut.

Mengingat banyaknya data jalan yang perlu direkap serta pengambilan keputusan yang cepat dan tepat. Sangat diperlukan sebuah sistem yang mampu melaporkan kerusakan jalan secara otomatis. Salah satunya dengan memanfaatkan teknologi sistem embedded. Teknologi sistem embedded dapat digunakan untuk melakukan pelaporan secara otomatis dan *realtime*. Sistem tersebut akan melaporkan data terkait kerusakan jalan kepada petugas dinas terkait yang dapat dipantau melalui website.

## 1.2 Rumusan masalah

Berdasarkan latar belakang yang ada, maka rumusan masalah yang dapat dibuat adalah sebagai berikut:

1. Bagaimanakah hasil analisis dan spesifikasi persyaratan sistem pelaporan kerusakan jalan otomatis tersebut?
2. Bagaimanakah hasil rancangan sistem perangkat lunak yang sesuai dengan spesifikasi persyaratan sistem tersebut?
3. Bagaimanakah hasil implementasi sistem perangkat lunak yang sesuai dengan rancangan sistem tersebut?
4. Bagaimanakah hasil pengujian sistem perangkat lunak tersebut?

## 1.3 Tujuan

Dari rumusan masalah di atas, maka tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Mengetahui hasil analisis dan menyusun spesifikasi persyaratan sistem pelaporan kerusakan jalan otomatis tersebut.
2. Mengetahui rancangan sistem perangkat lunak yang sesuai dengan spesifikasi persyaratan sistem tersebut.

3. Mengetahui implementasi sistem perangkat lunak yang sesuai dengan rancangan sistem tersebut.
4. Mengetahui hasil pengujian sistem perangkat lunak tersebut.

#### 1.4 Manfaat

Manfaat dari penelitian ini antara lain:

1. Membantu petugas dalam mengetahui kedalaman dari kerusakan jalan.
2. Membantu petugas dalam mengetahui lokasi kerusakan jalan.
3. Membantu petugas dalam mengetahui tingkat kerusakan jalan.
4. Membantu petugas dalam melakukan perekapan data terkait lokasi dan kategori kerusakan jalan.

#### 1.5 Batasan masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Perangkat lunak dikembangkan dengan menggunakan bahasa pemrograman PHP dan Python.
2. Deteksi sensor dilakukan hanya pada jalan perkerasan lentur.
3. Sistem hanya mampu digunakan di luar ruangan karena spesifikasi dari sensor GPS yang mengguna antena sebagai penangkat sinyal.
4. Kerusakan jalan yang diidentifikasi adalah lubang pada jalan raya.
5. Kendaraan bermotor yang digunakan adalah Yamaha X-Ride 2014.
6. Kecepatan yang digunakan pada kendaraan bermotor adalah 30-40 km/jam.
7. Pengujian hanya fokus pada pengujian sistem, tidak pada pengujian kinerja hardware.
8. Sensor ultrasonik hanya dapat digunakan pada lubang jalan yang tidak tergenang oleh air.

#### 1.6 Sistematika pembahasan

##### 1. BAB 1 PENDAHULUAN

Bab ini membahas latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, serta sistematika pembahasan.

##### 2. BAB 2 LANDASAN KEPUSTAKAAN

Bab ini membahas mengenai dasar teori yang digunakan sebagai acuan dalam melakukan penelitian. Pada bab ini juga membahas penelitian-penelitian mengenai sistem deteksi jalan berlubang yang telah dilakukan sebelumnya. Dasar teori yang digunakan sebagai pedoman dalam melakukan penelitian ini meliputi Rekayasa Perangkat Lunak, Pengembangan Perangkat Lunak, Perkerasan Lentur, Rekayasa *Embedded System*, *Raspberry PI*, Modul Sensor Getar SW-18020, Modul Sensor Ultrasonik HC-SR04, Modul *Raspberry GPS Add-On V2.0*, *Framework CodeIgniter*, *Realtime Database Firebase*, Bahasa Pemrograman *Python*, Bahasa Pemrograman *PHP*.

**3. BAB 3 METODOLOGI**

Bab ini membahas deskripsi alur proses penelitian yang akan dilakukan meliputi identifikasi masalah, studi literatur, pengumpulan data, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian sistem, serta kesimpulan dan saran.

**4. BAB 4 ANALISIS KEBUTUHAN**

Bab ini membahas analisis kebutuhan dan pemodelan kebutuhan sistem. Analisis kebutuhan terdiri atas identifikasi aktor, aturan penomoran, analisis kebutuhan fungsional sistem, analisis kebutuhan non-fungsional sistem. Sedangkan pemodelan kebutuhan terdiri atas *use case diagram* dan *use case scenario*.

**5. BAB 5 PERANCANGAN DAN IMPLEMENTASI**

Bab ini membahas proses yang dilakukan dengan membuat rancangan berdasarkan hasil analisis kebutuhan. Pada tahap ini akan dihasilkan perancangan hardware berupa rangkaian dan konfigurasi alat dan perancangan perangkat lunak yang meliputi perancangan sistem, perancangan data dan perancangan antarmuka. Setelah fase perancangan selesai, maka rancangan tersebut akan diimplementasikan. Pada tahap ini, implementasi sistem dilakukan dengan menggunakan bahasa pemrograman *Python* dan *PHP* serta menggunakan media penyimpanan *cloud realtime database*. Sistem akan dikembangkan dengan metode *Object Oriented Programming*.

**6. BAB 7 PENGUJIAN**

Bab ini membahas proses pengujian yang dilakukan untuk menemukan kesalahan pada sistem. Pengujian dilakukan dengan menggunakan strategi pengujian sistem yang terdiri atas pengujian *unit testing*, *integration testing* dan *validation testing*.

**7. BAB 8 PENUTUP**

Bab ini membahas proses dalam penarikan kesimpulan berdasarkan hasil dari penelitian yang telah dilakukan setelah menyelesaikan semua tahap penelitian dan saran berupa masukan mengenai pengembangan sistem di masa mendatang untuk melengkapi maupun memperbaiki kesalahan yang terjadi dalam penelitian ini.

## BAB 2 LANDASAN KEPUSTAKAAN

### 2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini mengacu pada beberapa penelitian sebelumnya. Pada penelitian *Automatic Detection and Notification of Potholes and Humps on Roads to Aid Drivers*, sistem dikembangkan untuk mendeteksi lubang di jalan dan polisi tidur dengan memanfaatkan sensor *Ultrasonic* dan menandai lokasi dari lubang atau polisi tidur tersebut. Tujuan utama dari penelitian ini adalah untuk memberi peringatan adanya lubang atau polisi tidur di jalan melalui perangkat android kepada pengguna. Pengujian dilakukan dengan meletakkan *microcontroller* pada penyangga sepeda motor yang akan mengirimkan informasi ke database. Kemudian akan diproses dan dikirim kembali sebagai peringatan lokasi lubang atau polisi tidur terdekat kepada pengguna (Madli et al., 2015).

Pada penelitian *Rancang Bangun Smart Vehicle Untuk Mendeteksi Dini Kecelakaan Dengan Pelaporan Visual Pada Google Maps* peneliti membangun sistem untuk mendeteksi tabrakan pada kendaraan. Sistem ini akan mengirimkan koordinat dari lokasi tabrakan tersebut yang ditampilkan dalam *Google Maps*. Selain itu koordinat lokasi tabrakan juga akan dikirimkan melalui SMS menggunakan *SMS Gateway*. Peneliti menggunakan *Microcontroller Arduino* dengan memanfaatkan sensor *Accelerometer* untuk mendeteksi tabrakan. Peneliti menguji sensor pada kondisi tabrakan dari depan, samping dan belakang untuk menentukan nilai terendah ambang batas. Jika terdeteksi melebihi nilai ambang batas, maka sistem akan mengenali data tersebut sebagai kecelakaan dan mengirimkan lokasi tersebut (Adhitia et al., 2016).

Penelitian *IOT Based Pothole Detection and Notification System* peneliti mengembangkan sistem pendeteksi lubang dengan memanfaatkan IoT. Pada penelitian ini, sistem dikembangkan dengan menggunakan sensor getar dan sensor *Accelerometer*. Tujuan dari penelitian ini adalah untuk mendeteksi lubang jalan dengan memanfaatkan nilai getaran pada kendaraan bermotor dengan nilai ambang batas. Peneliti juga menggunakan modul GPS yang berfungsi untuk memberikan koordinat lokasi lubang jalan ke sistem saat nilai yang dihasilkan melebihi nilai ambang batas (Gnanapriya, 2015).

Penelitian yang berjudul *Survey on Road Surface Monitoring System Using Internet of Things* mengembangkan sistem monitoring kondisi jalan dengan memanfaatkan konsep *Internet of Things* untuk mewujudkan *Smart City*. Peneliti menggunakan teknologi pemrosesan gambar dengan model pemasangan pada sepeda. Modul yang digunakan adalah modul sinar laser yang berfungsi untuk memancarkan *grid* cahaya laser. Selain itu juga digunakan modul kamera yang bekerja untuk memotret dan memberi peringatan kepada pengendara mengenai kondisi jalan yang buruk. Pendeteksian dilakukan dengan menggunakan metode pencocokan gambar. Pencocokan ini dilakukan dengan membandingkan template gambar jalan yang baik dengan hasil pengambilan gambar dari kamera secara periodik (S and Muralidoss, 2018).

## 2.2 Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak adalah sebuah disiplin ilmu rekayasa yang berfokus pada seluruh aspek produk perangkat lunak. Aspek tersebut meliputi langkah yang paling awal dari spesifikasi sistem hingga pemeliharaan sistem setelah sistem tersebut dapat digunakan. Rekayasa berperan penting dalam menghasilkan kualitas yang diharapkan sesuai dengan jadwal dan anggaran yang telah ditentukan. Tanpa rekayasa, program hanya bisa dimengerti oleh pemrogram itu sendiri yang berdampak pada waktu yang lama pada pengembangan program. Secara umum, perekrayasa perangkat lunak mengadopsi pendekatan sistematis dan terorganisir untuk menghasilkan perangkat lunak kualitas tinggi (Sommerville, 2011).

Rekayasa perangkat lunak sangat penting dalam pembuatan sebuah sistem. Hal ini dikarenakan baik individu maupun masyarakat sangat bergantung pada sistem perangkat lunak yang canggih. Sehingga perekrayasa perangkat lunak harus mampu menghasilkan sistem yang handal dan terpercaya secara ekonomis dan cepat. Metode dan teknik yang digunakan dalam rekayasa perangkat lunak, lebih murah dibandingkan dengan hanya menulis program seperti proyek pemrograman pribadi. Kebanyakan sistem yang sudah ada, menghabiskan biaya lebih untuk mengubah perangkat lunak setelah digunakan (Sommerville, 2011).

## 2.3 Pengembangan Perangkat Lunak

Pengembangan perangkat lunak merupakan sekumpulan aktivitas yang mengarah pada produksi perangkat lunak. Pengembangan perangkat lunak yang telah digunakan oleh perekrayasa perangkat lunak memiliki banyak model. Namun pada dasarnya, harus mencakup empat aktivitas rekayasa perangkat lunak yang meliputi spesifikasi perangkat lunak, perancangan dan implementasi perangkat lunak, validasi perangkat lunak dan evolusi perangkat lunak. Pengembangan perangkat lunak sangat kompleks dan membutuhkan proses yang kreatif dan cerdas. Hal ini juga harus berdasarkan pada pembuatan keputusan dan penilaian yang dilakukan oleh seorang pengembang (Sommerville, 2011).

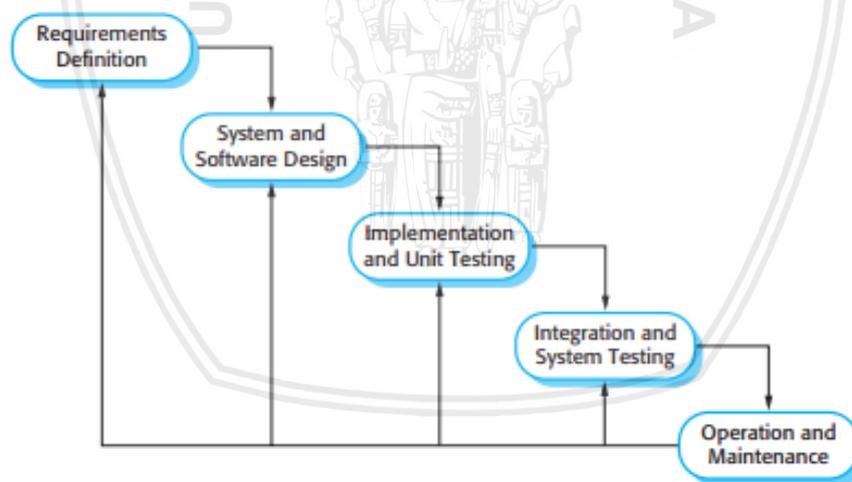
Meskipun tidak ada pengembangan perangkat lunak yang ideal, terdapat keleluasaan untuk meningkatkan pengembangan perangkat lunak pada beberapa organisasi. Pengembangan masih memasukkan teknik yang mungkin tidak memberikan keuntungan pada metode pengembangan perangkat lunak. Adanya pengembangan perangkat lunak mampu meningkatkan standarisasi pengembangan dimana keberagaman pada pengembangan perangkat lunak yang ada di organisasi dapat dikurangi. Hal ini mengarah pada peningkatan komunikasi dan mengurangi waktu pelatihan. Serta dapat membuat otomatisasi proses lebih murah. Standarisasi sangat penting untuk langkah pertama dalam mengenalkan metode serta teknik rekayasa perangkat lunak baru dan praktik pengembangan perangkat lunak yang baik (Sommerville, 2011).

### 2.3.1 Siklus Pengembangan Perangkat Lunak (SDLC)

Model pengembangan perangkat lunak adalah representasi sederhana dari pengembangan perangkat lunak. Setiap model pengembangan merupakan proses dari sebuah perspektif dan hanya mencakup beberapa informasi mengenai pengembangan. Pada model aktivitas proses menunjukkan aktivitas dan rangkaian yang ada di dalamnya, bukan menampilkan aturan dari orang yang terlibat dalam aktivitas tersebut. Ada beberapa model pengembangan perangkat lunak yang umum digunakan diantaranya *Waterfall model*, *Incremental development*, dan *Reuse-oriented software engineering*. Model pengembangan tersebut tidak terikat satu sama lain dan dapat digunakan secara bersamaan, terutama untuk pengembangan sistem besar (Sommerville, 2011).

#### 2.3.1.1 Waterfall Model

Model ini disebut sebagai *Waterfall Model* karena setiap proses pada model ini berkesinambungan antara satu fase dengan fase lainnya. *Waterfall Model* merupakan aktivitas pengembangan paling dasar mulai dari spesifikasi, pengembangan, validasi, dan evolusi. *Waterfall Model* merepresentasikan aktivitas pengembangan sebagai fase proses yang terpisah seperti spesifikasi kebutuhan, perancangan perangkat lunak, implementasi, pengujian dan sebagainya. Sehingga satu fase harus terselesaikan terlebih dahulu sebelum ke fase selanjutnya (Sommerville, 2011). Model ini dapat diilustrasikan dalam Gambar 2.1.



Gambar 2.1 Waterfall Model  
(Sumber : Sommerville, 2011)

#### 2.3.1.2 Fase Waterfall Model

Menurut Sommerville (2011), ada beberapa tahap atau fase pada *Waterfall Model*. Setiap fase dari model ini harus terselesaikan sebelum masuk ke fase selanjutnya. Fase dari SDLC *Waterfall Model* dibagi menjadi 5 bagian utama yaitu:

1. Definisi Kebutuhan, merupakan fase dimana layanan dari sistem harus mampu memenuhi tujuan sepenuhnya dengan melakukan konsultasi

- bersama pengguna sistem. Kebutuhan tersebut akan didefinisikan secara detail dan disajikan menjadi spesifikasi sistem.
2. Perancangan sistem, merupakan proses perancangan sistem yang menyediakan kebutuhan untuk perangkat keras dan perangkat lunak dengan menentukan keseluruhan arsitektur sistem. Perancangan perangkat lunak melibatkan identifikasi dan pendeskripsian abstraksi perangkat lunak secara mendasar serta hubungannya.
  3. Implementasi dan pengujian unit, merupakan fase dimana rancangan perangkat lunak akan diimplementasikan menjadi program atau unit program. Pengujian unit melibatkan verifikasi dari setiap unit yang terdapat dalam spesifikasi kebutuhan.
  4. Integrasi dan pengujian sistem, merupakan proses unit program atau program diintegrasikan dan diuji sebagai kesatuan sistem untuk memastikan kebutuhan perangkat lunak telah terpenuhi. Setelah pengujian, sistem perangkat lunak akan diserahkan kepada pelanggan.
  5. Operasi dan perawatan, proses ini merupakan fase terpanjang dalam siklus perangkat lunak. Perawatan melibatkan pembenahan kesalahan yang tidak ditemukan pada fase sebelumnya dalam siklus perangkat lunak. Proses ini meningkatkan implementasi dari unit sistem dan peningkatan pelayanan sistem sebagai kebutuhan baru yang telah ditemukan.

### **2.3.2 Object Oriented Programming (OOP)**

*Object Oriented Programming* atau Pemrograman Berorientasi Objek merupakan sebuah konsep pengembangan perangkat lunak dengan menggunakan metode berorientasi objek. Metode ini sama halnya seperti proses pengembangan perangkat lunak pada umumnya yang meliputi analisis, perancangan dan pengujian. Teknologi berorientasi objek ini menyediakan *framework* teknikal untuk sebuah proses berdasarkan komponen dengan model rekayasa perangkat lunak. Konsep berorientasi objek menekankan pembuatan kelas yang membungkus kedua data. Algoritma yang digunakan untuk memanipulasi data. Jika rancangan dan implementasi dilakukan dengan benar, kelas berorientasi objek dapat digunakan kembali pada aplikasi yang berbeda dan arsitektur komputer yang berbeda (Pressman, 2001).

### **2.3.3 Pemodelan Perangkat Lunak**

#### **2.3.3.1 Use Case Diagram**

*Use Case Diagram* merupakan salah satu jenis pemodelan *Unified Markup Language* (UML) yang berfokus pada interaksi yang dilakukan oleh aktor maupun sistem serta mampu memberikan deskripsi terkait tipe interaksi yang terjadi. Dalam membuat *use case diagram* dilakukan dengan mengidentifikasi berbagai jenis orang (atau perangkat) yang menggunakan sistem atau produk. Seorang aktor adalah segala sesuatu yang berkomunikasi dengan sistem atau produk. Komunikasi ini dilakukan baik secara eksternal maupun ke sistem itu sendiri (Pressman, 2001). Notasi dari *use case diagram* dapat dilihat pada Tabel 2.1 notasi *use case diagram*.

Tabel 2.1 Notasi *use case diagram*

No.	Simbol	Nama	Keterangan
1		<i>Actor</i>	Menggambarkan pengguna atau sistem lain yang berinteraksi dan terlibat dalam sistem.
2		<i>Dependency</i>	Menggambarkan relasi yang dapat mempengaruhi perubahan antara elemen <i>independent</i> dengan elemen yang tidak <i>independent</i> .
3		<i>Generalization</i>	Menggambarkan relasi dimana semua struktur dan perilaku dari objek yang dimiliki oleh induk ( <i>parent</i> ) akan diwariskan kepada anaknya ( <i>child</i> ).
4		<i>Include</i>	Menggambarkan relasi yang menspesifikkan antara <i>use case</i> dari sumber ke <i>base</i> secara eksplisit.
5		<i>Extend</i>	Menggambarkan relasi yang menspesifikkan antara <i>use case</i> dari sumber untuk memperluas perilaku dari <i>use case</i> kepada target yang diberikan.
6		<i>Assosiation</i>	Menggambarkan relasi yang terjadi dari suatu objek dengan objek lainnya
7		<i>System</i>	Menggambarkan lingkungan yang terdapat dalam <i>use case</i> dan aktor yang terlibat di dalamnya.
8		<i>Use case</i>	Menggambarkan suatu aksi yang dapat dilakukan dan diukur oleh aktor.
9	+++++	<i>Collaboration</i>	Menggambarkan interaksi aturan dan elemen dalam menyediakan perilaku yang lebih besar dan dapat bekerja sama.

**Tabel 2.2 Notasi use case diagram (lanjutan)**

10		<i>Note</i>	Memberikan penjelasan elemen fisik dari sumber daya komputasi yang dijalankan pada aplikasi.
----	---	-------------	--

**2.3.3.2 Sequence Diagram**

*Sequence diagram* merupakan salah satu jenis pemodelan UML yang berfokus pada interaksi yang terjadi antar objek. Interaksi ini dapat terjadi dalam suatu sistem dan antar objek itu sendiri. Interaksi yang terjadi digambarkan berdasarkan urutan waktu dan tahapan untuk menghasilkan sesuatu seperti *use case diagram*. *Sequence diagram* memiliki sintaks yang kaya, yang memungkinkan berbagai jenis interaksi untuk dimodelkan. Sesuai namanya, *sequence diagram* menunjukkan urutan interaksi yang terjadi selama kasus penggunaan tertentu (Pressman, 2001). Notasi dari *sequence diagram* dapat dilihat pada Tabel 2.2 notasi *sequence diagram*.

**Tabel 2.2 Notasi sequence diagram**

No.	Notasi	Nama	Keterangan
1		<i>View</i>	Merepresentasikan suatu bagian yang berfungsi untuk menjadi komunikasi antara <i>user</i> dengan data.
2		<i>Controller</i>	Merepresentasikan suatu bagian yang berfungsi untuk mengatur proses yang terjadi antara <i>model</i> dan <i>view</i> , serta mengatur alur <i>request</i> dan <i>respond</i> kepada user.
3		<i>Model</i>	Merepresentasikan suatu bagian yang berfungsi untuk menangani segala proses yang terdapat di dalam <i>database</i> .
4		<i>Actor</i>	Merepresentasikan suatu bagian yang dapat berkomunikasi dengan objek.
5		<i>Lifeline</i>	Merepresentasikan suatu bagian basis waktu dalam suatu objek.
6		<i>Activation</i>	Merepresentasikan suatu bagian untuk melakukan sebuah aksi pada suatu objek.



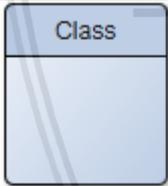
**Tabel 2.2 Notasi *sequence diagram* (lanjutan)**

7		<i>Message</i>	Merepresentasikan relasi antara satu objek dengan objek yang lain.
---	---	----------------	--

**2.3.3.3 Class Diagram**

*Class diagram* merupakan salah satu jenis pemodelan UML yang menggambarkan kelas dan relasi pada suatu sistem yang kemudian akan diimplementasikan dalam pembuatan perangkat lunak. *Class diagram* digunakan ketika mengembangkan model sistem berorientasi objek untuk menunjukkan kelas dalam suatu sistem dan asosiasi antar kelas didalamnya. Kelas objek dapat dianggap sebagai definisi umum dari satu jenis objek sistem. Sedangkan suatu asosiasi adalah hubungan antara kelas yang menunjukkan bahwa ada hubungan antar kelas-kelas yang ada. *Diagram class* dalam UML dapat diekspresikan pada tingkat detail yang berbeda. Ketika perancang sistem mengembangkan sebuah model, tahap pertama biasanya melihat representasi di dunia nyata, mengidentifikasi objek-objek penting, dan menggambarkannya sebagai kelas (Pressman, 2001). Notasi dari *sequence diagram* dapat dilihat pada Tabel 2.3 notasi *class diagram*.

**Tabel 2.3 Notasi *class diagram***

No	Simbol	Nama	Deskripsi
1		Class	Merepresentasikan suatu bagan yang berfungsi untuk membuat suatu objek dan operasi serta atribut yang ada di dalamnya.
3		Asosiasi	Merepresentasikan relasi antar objek yang memungkinkan suatu objek dapat menggunakan operasi atau atribut dari objek lain.
3		Generalisasi	Merepresentasikan relasi antar <i>class</i> yang memungkinkan suatu <i>class</i> dapat mewariskan operasi atau atribut untuk <i>class</i> lain.
4		Dependency	Merepresentasikan ketergantungan suatu <i>class</i> dengan <i>class</i> lain.
5		Agregasi	Merepresentasikan relasi suatu <i>class</i> merupakan bagian dari <i>class</i> lain.

### 2.3.4 Strategi Pengujian

Pengujian sistem bertujuan untuk menemukan sebanyak mungkin kesalahan dalam sebuah kode program. Selain itu juga dengan pengujian sistem mampu memperbaiki kesalahan sistem sebelum program tersebut diserahkan kepada pengguna. Suatu pengujian dikatakan baik jika pengujian tersebut memiliki tingkat kemungkinan yang tinggi dalam menemukan kesalahan sistem. Sebuah strategi pengujian harus mampu menguji dari tingkat rendah hingga tingkat tinggi. Uji tingkat rendah merupakan sebuah proses untuk melakukan verifikasi bahwa kode program telah diimplementasikan dengan benar. Sedangkan uji tingkat tinggi harus mampu melakukan pengujian untuk melakukan validasi fungsionalitas sistem terhadap kebutuhan *stakeholder* terkait. Pressman (2010) menjelaskan beberapa strategi yang digunakan untuk melakukan pengujian pada pendekatan berorientasi objek sebagai berikut:

#### 2.3.4.1 Unit Testing

*Unit testing* atau pengujian unit merupakan pengujian yang dilakukan pertama kali oleh pengembang. Pengujian unit sangat bergantung pada jenis bahasa pemrograman dan pendekatan yang digunakan oleh pengembang sistem. Untuk pendekatan berorientasi objek, pengujian unit berfokus pada *modul*, *class* dan *method*. Pengujian unit dilakukan dengan mengisolasi unit program dan mengujinya secara individu tanpa terlibat dengan unit lain. *Whitebox testing* merupakan teknik yang digunakan dalam pengujian unit. Pengujian ini dilakukan dengan mengacu pada struktur program sehingga membutuhkan kode sumber. *Whitebox testing* bertujuan untuk memastikan pada setiap kode program telah dieksekusi setidaknya satu kali (Pressman, 2001).

#### 2.3.4.2 Integration Testing

*Integration Testing* atau pengujian integrasi merupakan strategi pengujian setelah pengujian unit. Pada pengujian ini berfokus pada integrasi antar unit atau komponen dapat terhubung dengan baik dan benar. Pengujian dilakukan untuk menemukan kecacatan atau kerusakan dalam hubungan antar komponen yang terdapat dalam sistem. Pengujian integrasi terbagi menjadi dua strategi pengujian yaitu *top-down integrations* dan *bottom-up integrations*. Pengujian *top-down integration* dilakukan dimana komponen utama akan memanggil komponen level bawah. Jika komponen bawah belum jadi, maka dapat memanfaatkan *stub* sebagai komponen untuk simulasi. Sedangkan strategi *bottom-up integration* merupakan kebalikan dari strategi *top-down*. Sama halnya dengan strategi sebelumnya, simulasi dapat dilakukan dengan menggunakan *driver* yang berfungsi sama seperti *stub* (Pressman, 2001).

#### 2.3.4.3 Validation Testing

*Validation Testing* atau pengujian validasi merupakan proses terakhir dari tahap pengujian yang dilakukan pada lingkungan yang sebenarnya. Pengujian ini menitikberatkan pada sudut pandang pengguna sistem tanpa mengetahui kode sumber yang ada pada program tersebut. Metode pengujian ini juga disebut sebagai *Blackbox testing*. *Blackbox testing* sendiri merupakan sebuah teknik pengujian yang didasarkan pada spesifikasi kebutuhan sistem. Sehingga pada metode ini tidak akan ditemukan kode sumber untuk diuji (Pressman, 2001).

## 2.4 Perkerasan Lentur

### 2.4.1 Pengertian Perkerasan Lentur

Konstruksi perkerasan lentur merupakan suatu perkerasan jalan raya yang memanfaatkan aspal sebagai bahan baku utama pengikat. Sifat yang dimiliki oleh lapisan perkerasan lentur adalah memikul beban lalu lintas serta menyebarkannya ke tanah dasar. Konstruksi perkerasan lentur terdiri atas beberapa lapisan yang diletakkan pada tanah dasar yang sudah dipadatkan. Perkerasan letur digunakan pada jalan raya sebagai jalan primer (Sukirman, 1999).

### 2.4.2 Kerusakan Perkerasan Lentur

Menurut Sukirman (1999) menerangkan bahwa kerusakan pada perkerasan lentur dapat diakibatkan oleh satu atau lebih faktor yang saling berkaitan. Adapaun faktor yang menjadi penyebab kerusakan perkerasan lentur antara lain:

1. Lalu lintas yang disebabkan oleh peningkatan dan repetisi beban jalan raya.
2. Air yang berasal dari air hujan, drainase jalan yang tidak baik hingga kenaikan air akibat kapilaritas.
3. Material konstruksi yang tidak baik dan pengolahan bahan yang salah.
4. Iklim di Indonesia yang memiliki suhu udara dan curah hujan yang tinggi.
5. Kurang baiknya proses pemadatan yang dilakukan pada lapisan tanah dasar.

### 2.4.3 Jenis Kerusakan Perkerasan Lentur

Menurut Manual Pemeliharaan Jalan Nomor: 03/MN/B/1983 yang dikeluarkan oleh Direktorat Jenderal Bina Marga, kerusakan jalan dapat dibedakan atas:

1. Retak (*cracking*).
2. Distorsi (*distortion*).
3. Cacat permukaan (*disintegration*).
4. Pengausan (*polished aggregate*).
5. Kegemukan (*bleeding or flushing*).
6. Penurunan pada bekas penamaan utilitas.

### 2.4.4 Lubang Jalan

Menurut Manual Pemeliharaan Jalan Nomor: 03/MN/B/1983 yang dikeluarkan oleh Direktorat Jenderal Bina Marga, lubang pada jalan raya masuk dalam kategori jenis kerusakan jalan berupa cacat permukaan (*disintgration*). Lubang (*potholes*), berupa mangkuk atau cekungan yang terdapat di jalan dengan ukuran yang bervariasi. Ukuran tersebut terdiri dari kerusakan kecil hingga kerusakan besar. Lubang jalan ini lah yang nantinya akan menjadi wadah genangan air yang mana dapat menyebabkan semakin parah tingkat kerusakan jalan (Sukirman, 1999).

Menurut Shahin (1994), tingkat kerusakan lubang jalan dapat dikategorikan menjadi 3 kategori utama. Kategori tersebut terdiri atas kategori rusak parah,

rusak sedang dan rusak ringan. Kategori kerusakan ini berdasarkan dari kedalaman lubang jalan. Pada setiap kategori memiliki ukuran yang berbeda-beda. Tingkat kerusakan lubang jalan dapat dilihat pada Tabel 2.4 tingkat kerusakan lubang jalan.

**Tabel 2.4 Tingkat kerusakan lubang jalan**

Kedalaman Maksimum	Diameter rata-rata (mm) (inci)		
	100-200 mm (4 – 8 inci)	200-450 mm (8 – 18 inci)	450-750 mm (18 – 30 inci)
13 mm - $\leq$ 25 mm (1/2 – 1 inci)	L	L	M
> 25 mm - $\leq$ 50 mm (1 – 2 inci)	L	M	H
> 50 mm (2 inci)	M	M	H

(Sumber : Shahin, 1994)

Keterangan :

H = Rusak Parah

M = Rusak Sedang

L = Rusak Ringan

## 2.5 Rekayasa *Embedded System*

*Embedded* sistem adalah sistem komputer dengan fungsi khusus yang tidak dirancang sehingga perlu diprogram ulang seperti unit kontrol mesin, perangkat medis implan, peralatan, dll. Jenis sistem tertanam yang paling umum adalah mikrokontroler, yang merupakan sistem komputer kecil pada satu sirkuit terintegrasi. mikrokontroler ahli melakukan tugas-tugas seperti membaca sensor dan menerapkan hukum kontrol, tetapi penting untuk dicatat bahwa perangkat ini menginterpretasikan data secara diskrit, berbeda dengan dunia nyata di mana data dikenal dalam bentuk analog, sehingga semua yang kita lihat bersifat berkelanjutan. Untuk menyesuaikan ini, mikrokontroler akan memanfaatkan konversi digital ke analog (DAC) untuk berpindah dari nilai biner ke tegangan *output* aktual dan konversi analog ke digital (ADC) untuk berpindah dari sinyal *input* ke data digital yang mikrokontroler dapat digunakan (Lambert, 2018).

Menurut Spichkova (2013) menyebutkan bahwa alasan mengapa sistem *embedded* dapat digunakan secara luas adalah karena mereka memiliki atribut berikut.

### 1. Konsumsi daya rendah

Efisiensi energi komputasional adalah karakteristik utama dari teknologi platform eksekusi. Karena sistem *embedded* biasanya digunakan di beberapa perangkat kecil, sumber dayanya relatif terbatas. Misalnya, jam tangan elektronik. Kebutuhan konsumsi energi yang rendah harus dipertimbangkan pada awal. Tidak ada yang mau mengganti baterai untuk arlojinya setiap bulan.

2. Bisa diandalkan
 

Sistem yang tertanam harus dapat diandalkan dan dapat dipelihara. Keandalan adalah kemungkinan bahwa sistem tidak akan gagal. *Maintainability* adalah probabilitas bahwa sistem yang gagal dapat diperbaiki dalam kerangka waktu tertentu. Sistem *embedded* selalu diintegrasikan ke dalam perangkat. Seluruh perangkat tidak dapat berfungsi jika sistem *embedded* tidak berfungsi atau rusak. Perakitan dan panas Setiap masalah harus dipertimbangkan, sebagai bagian dari perangkat, sistem *embedded* harus dirancang untuk menghindari masalah ini, keandalan dan pemeliharaan sangat penting.
3. Antarmuka pengguna khusus
 

Kebanyakan sistem *embedded* tidak menggunakan *keyboard* atau *monitor* besar untuk antarmuka pengguna mereka. Sebaliknya, ada antarmuka pengguna khusus yang terdiri dari tombol *push*, roda kemudi dll. Pengguna hanya dapat mengontrol perangkat, mereka tidak dapat memodifikasi program. Sisi positifnya adalah, hal tersebut baik untuk keamanan sistem.
4. Efisiensi waktu nyata
 

Sistem *embedded* umumnya memiliki persyaratan *real-time*. Sistem *embedded* biasanya digunakan untuk mengendalikan prosedur yang berupa serangkaian proses. Sistem yang tertanam dapat mengatur kerangka waktu. Suatu proses harus diselesaikan dalam jangka waktu tertentu. Hal tersebut adalah persyaratan *real-time* yang sangat penting dalam aplikasi lini perakitan.

## 2.6 Raspberry Pi

*Raspberry Pi* adalah sebuah mini komputer yang memiliki ukuran menyerupai kartu kredit serta dapat digunakan seperti *PC desktop* dan mampu menjalankan program kantor, permainan komputer hingga pemutar video beresolusi tinggi (Angga, 2017). *Raspberry Pi* juga dikenal sebagai kartu kredit berbasis ARM yang berukuran *Single Board Computer* yang dibuat oleh *Raspberry Pi Foundation*. *Raspberry Pi* mampu menjalankan sistem operasi GNU atau Linux berbasis *Debian Raspbian* dan *port* dari banyak OS lain yang ada. Spesifikasi yang dimiliki oleh *Raspberry Pi 3* dapat dilihat pada Tabel 2.5. Gambar dari *Raspberry Pi Model B v1.2* ditampilkan dalam gambar 2.2.

**Tabel 2.5 Spesifikasi *Raspberry Pi 3 Model B v1.2***

Board	Raspberry Pi 3 Model B
ProsesBrsor	Broadcom BCM2837
CPU Core	Quadcore ARM Cortex-A53, 64Bit
Clock Speed	1.2GHz (Roughly 50% faster than Pi2)
RAM	1 GB
GPU	400 MHz VideoCore IV®

**Tabel 2.5 Spesifikasi Raspberry PI 3 Model B v1.2 (lanjutan)**

Network Connectivity	1 x 10 / 100 Ethernet (RJ45 Port)
Wireless Connectivity	802.11n wireless LAN (WiFi) and Bluetooth 4.1
USB Ports	4 x USB 2.0
GPIOs	2 x 20 Pin Header
Camera Interface	15-pin MIPI
Display Interface	DSI 15 Pin / HDMI Out / Composite RCA
Power Supply (Current Capacity)	2.5 A

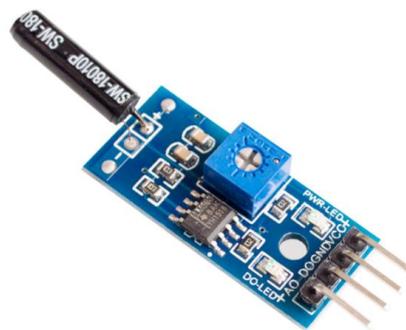
(Sumber: Angga, 2017)



**Gambar 2.2 Raspberry PI Model B v1.2**  
(Sumber: Angga, 2017)

### 2.7 Modul Sensor Getar SW-18020

Sensor Getar SW-18020 adalah perangkat keras yang digunakan untuk mendeteksi getaran (alarm) serta dapat digunakan sebagai alarm kendaraan bermotor hingga alarm pada jendela atau pintu rumah (Tokoteknologi, 2018). Sensor ini bekerja dengan getaran *non-directional* sensitivitas tinggi. Ketika modul stabil, sirkuit dihidupkan dan hasilnya tinggi. Ketika gerakan atau getaran terjadi, rangkaian akan diputus secara singkat dan *output* rendah. Pada saat yang sama, sensitivitas dari getaran yang diterima oleh modul dapat diatur. Sensor Getar SW-18020 dapat dilihat dalam Gambar 2.4 Modul Sensor Getar SW-18020.



**Gambar 2.3 Modul Sensor Getar SW-420**

## 2.8 Modul Sensor Ultrasonik HC-SR04

Sensor ultrasonik menggunakan suara untuk menentukan jarak antara sensor dan objek terdekat di jalurnya. Sensor ultrasonik pada dasarnya merupakan sensor suara, tetapi mereka beroperasi pada frekuensi di atas pendengaran manusia. Sensor mengirimkan gelombang suara pada frekuensi tertentu. Kemudian mendengarkan gelombang suara tertentu untuk memantul objek dan kembali. Sensor melacak waktu antara mengirim gelombang suara dan menerima gelombang suara kembali. Kecepatan suara dapat dihitung berdasarkan berbagai kondisi atmosfer, termasuk suhu, kelembaban, dan tekanan (Morgan, 2014).

Kemampuan sensor untuk mendeteksi objek juga tergantung pada orientasi objek ke sensor. Jika suatu objek tidak menghadirkan permukaan yang datar ke sensor maka mungkin gelombang suara akan memantulkan sinyal ke objek dan tidak kembali ke sensor. Sensor Ultrasonik HCSR04 dapat dilihat dalam Gambar 2.4. Modul Sensor Ultrasonik HC-SR04. Modul sensor ini memiliki 4 pin: VCC, GND, TRIG dan ECHO dengan penjelasan sebagai berikut :

1. VCC adalah catu daya 5v. Ini harus berasal dari mikrokontroler.
2. GND adalah pin ground. Pasang ke tanah di mikrokontroler.
3. TRIG harus dilampirkan ke pin GPIO yang dapat disetel ke HIGH.
4. ECHO sedikit lebih sulit. Output HCSR04 5v.



Gambar 2.4 Modul Sensor Ultrasonik HC-SR04

## 2.9 Modul *Raspberry GPS Add-On V2.0*

*Raspberry Pi GPS Add-on* disesuaikan untuk antarmuka *Raspberry Pi* berdasarkan modul *GPS NEO-6*. Melalui *port serial* pada *Raspberry Pi*, data yang dikembalikan dari modul *NEO-6* dapat diterima, sehingga informasi seperti lokasi dan waktu saat ini dapat dicari. *Raspberry Pi GPS Add-on* dapat dilihat dalam Gambar 2.5 *Raspberry Pi GPS Add-on*. Sedangkan spesifikasi dari modul sensor ini ditampilkan pada Tabel 2.6 Spesifikasi *Raspberry Pi GPS Add-on* (ITEAD, 2016).



**Gambar 2.5 Raspberry PI GPS Add-on**  
(Sumber : ITEAD, 2016)

**Tabel 2.6 Spesifikasi Raspberry PI GPS Add-on**

PCB size	65mm X 56mm X 1.6mm
Operation Level	Digital 3.3V DC
Interface	UART
Baud rate	9600(default)
Weight	106 g
Model	IM150627005
Board Size	65mm*56mm*1.6mm
Version	V2.0
Operation Level	Digital 3.3V

(Sumber : ITEAD, 2016)

### 2.10 Framework CodeIgniter

*Framework* atau kerangka merupakan sebuah pola dengan struktur konseptual dasar dan dapat digunakan untuk memecahkan sebuah permasalahan, dari yang sederhana hingga isu-isu kompleks yang ada.

*Framework* memiliki struktur aplikasi yang baik seperti, *design pattern*, *standard coding*, *common function* dan *best practice*. Sehingga dengan adanya *design pattern* dan *common function* memberikan kemudahan dalam pengembangan aplikasi dengan bantuan *framework*. *Framework* juga memudahkan pengembang lain untuk mempelajari dan mengubah aplikasi dengan adanya *framework*. Hal ini karena *framework* memiliki pola tertentu yang dapat diidentifikasi oleh pengembang dalam memahami kode suatu aplikasi (Daqiqil, 2011).

*CodeIgniter* adalah kerangka pengembangan aplikasi, yang dapat digunakan untuk mengembangkan situs web, menggunakan *PHP*. *CodeIgniter* adalah kerangka kerja *Open Source*. *CodeIgniter* memiliki satu set fungsi yang sangat kaya, yang akan meningkatkan kecepatan pekerjaan pengembangan situs web. *CodeIgniter* memiliki satu set yang sangat kaya perpustakaan (*library*) dan pembantu (*helper*). Dengan menggunakan *CodeIgniter*, dapat menghemat banyak waktu, bagi pemula dalam mengembangkan sebuah situs web dari awal. Tidak hanya itu, situs web yang dibangun di *CodeIgniter* juga aman, seperti kemampuan yang dimilikinya untuk mencegah berbagai serangan yang terjadi melalui situs web (Daqiqil, 2011). Beberapa keunggulan yang dimiliki oleh *codeigniter* dibandingkan *framework* lainnya yaitu:

1. *CodeIgniter* merupakan salah satu *framework PHP* berdasarkan hasil *branchmark*.
2. Modifikasi dalam *framework Codeigniter* sangat mudah karena tidak memerlukan *requirement* dan mudah dalam mengadopsi suatu *library*.
3. Dokumentasi panduan *Codeigniter* telah tersedia meskipun tanpa membutuhkan buku *Codeigniter*.
4. *CodeIgniter* sangat mudah dipelajari. Sehingga sering digunakan oleh pengembang dalam menyamaratakan kemampuan dari anggota tim agar dapat bekerja dan berkomunikasi dengan baik.

### **2.11 Realtime Database Firebase**

Menurut Nicolas (2017) secara intuitif, *database real-time* dapat dilihat sebagai *database* klasik yang mampu menangani beban kerja yang statusnya berubah secara permanen dengan menggunakan pemrosesan *real-time*. *Database Real-Time* adalah *database* yang mengikuti sejumlah waktu (*temporal*). Hal itu merupakan karakteristik *temporal* dari data yang disimpan dalam *database*, waktu dan sasaran kinerja. *Database real-time* kebanyakan berbeda dari *database* konvensional dengan tujuan kinerjanya. Perbedaan yang paling terlihat adalah perbedaan waktu yang berada dalam mikrodetik atau bahkan dalam nanodetik. Kemampuan dari *database real-time* inilah yang digunakan untuk mengevaluasi rata-rata dari transaksi yang terlewat dan biaya yang dikeluarkan dalam transaksi yang terlewat.

*Real-time database* cukup memadai ketika dimanfaatkan untuk memiliki gambaran yang baik tentang keadaan lingkungan saat ini. *Database* ini menggunakan batasan waktu yang tidak didukung dalam basis data konvensional. Tidak seperti *database* konvensional, *Real-time Database System* (RTDBS) menggunakan tenggat waktu untuk menentukan keakuratan nilai yang

terukur. *Database* konvensional memiliki respon yang buruk dan kurangnya prediktabilitas yang merupakan fitur utama dari RTDBS. Akhirnya, data tidak pernah menjadi usang dalam *database* konvensional. Hal ini menjadi keunggulan dari *database* konvensional yang tidak terjadi dalam sistem *database real-time* (Nicolas, 2017).

Lebih jauh lagi, Nicolas (2017) menyebutkan bahwa sebagai basis data *real-time*, sistem ini menyinkronkan basis data dengan setiap klien yang terhubung daripada menggunakan permintaan HTTP. Ini berarti hanya ada satu contoh umum dari *database* untuk setiap pengguna yang terhubung secara otomatis menerima pembaruan. Dengan demikian, tenggat waktu pada transaksi tidak dapat diimplementasikan sebagai sistem *Firebase* karena pemberitahuan informasi baru dikirimkan ke setiap pengguna yang terhubung. Setiap pengguna akan menerima pembaruan sesegera mungkin. Setelah *offline*, setiap perangkat memelihara salinan lokal dari *database* dan menyimpan data baru pada disk lokal sampai pulih konektivitas. Kemudian informasi diunggah ke *database real-time online*, sistem *online* menyelesaikan konflik dengan sendirinya. Dan perangkat pengguna mengambil perubahan yang tidak terjawab saat dia berada di luar.

## 2.12 Bahasa Pemrograman Python

*Python* merupakan sebuah bahasa pemrograman yang mudah untuk dipelajari dan interpretatif yang berfokus pada keterbacaan kode. Alasan tersebut sejalan dengan karakteristik dari bahasa pemrograman *Python*. Karena *Python* memiliki kode pemrograman yang sangat jelas, lengkap dan mudah untuk dipahami. Secara umum, *python* memiliki beberapa bentuk pemrograman seperti pemrograman berorientasi objek, imperatif dan fungsional. *Python* juga memiliki kemampuan untuk dapat berjalan di berbagai platform sistem operasi seperti *Windows* maupun *Linux* (Enterprise, 2017). Beberapa fitur atau kelebihan yang dimiliki *python* antara lain:

1. Memiliki koleksi kepustakaan (*library*) yang banyak. Dengan kata lain *python* memiliki modul-modul yang dapat digunakan langsung oleh pengguna sistem sesuai dengan keperluannya.
2. Memiliki struktur dan tata bahasa yang sederhana, mudah dipelajari dan jelas.
3. Memiliki aturan *layout* kode sumber. *Python* memungkinkan sebuah penulisan ulang kode program dapat dilakukan pengecekan dan pembacaan kode sumber.
4. Mendukung pemrograman berorientasi objek.
5. Memiliki sistem pengelolaan memori dan pengumpulan sampah otomatis. Fitur ini hampir sama seperti yang dimiliki oleh *java*, yaitu *garbage collection*. Fasilitas ini memudahkan pengguna tanpa harus melakukan pengaturan ingatan komputer secara langsung.
6. Modular, memungkinkan pengguna sistem untuk membuat modul baru dengan menggunakan bahasa pemrograman *python* maupun *C* atau *C++*.

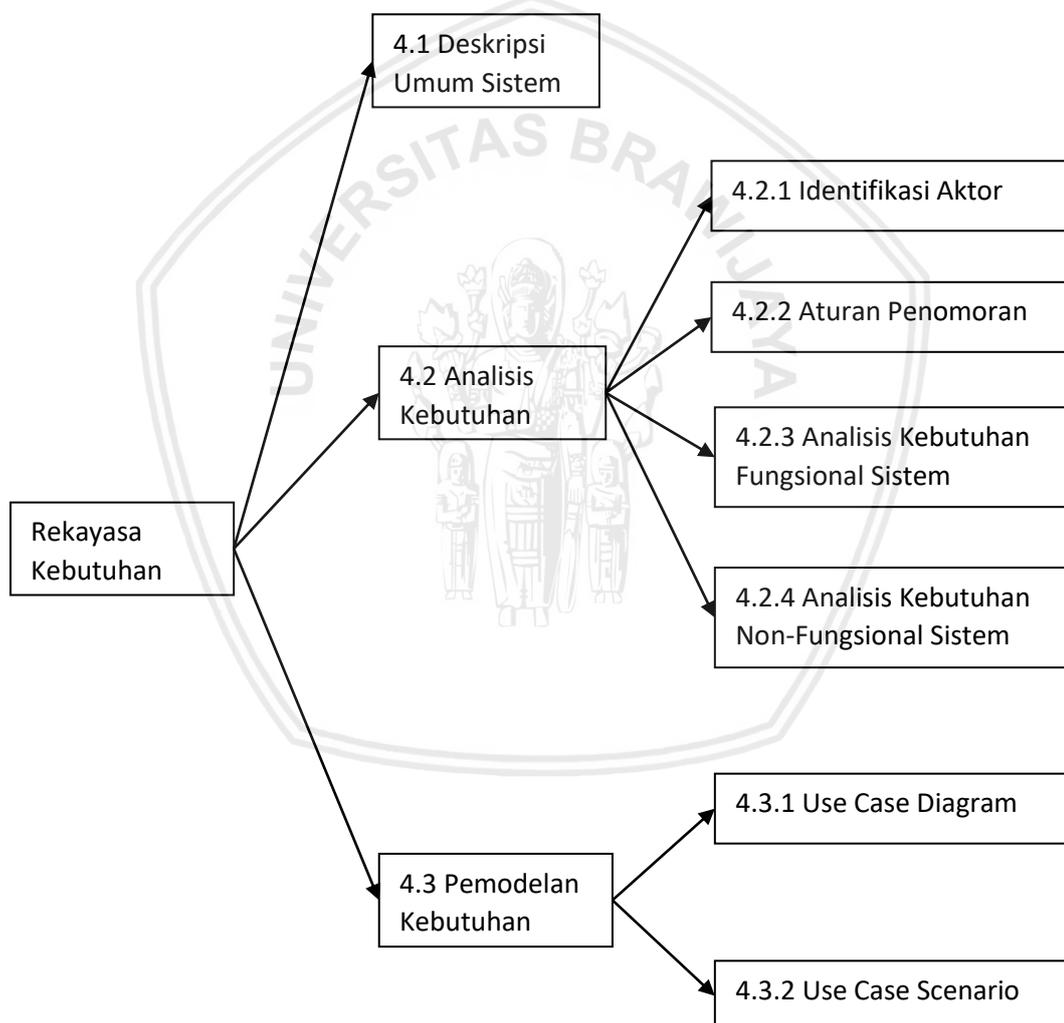
### 2.13 Bahasa Pemrograman *PHP*

*PHP Hypertext Processor* (*PHP*) merupakan salah satu bahasa pemrograman yang digunakan untuk pengembangan sistem berbasis web. Bahasa pemrograman ini dibuat secara khusus untuk membangun aplikasi berbasis web. Pemrograman dengan menggunakan bahasa *PHP* harus dieksekusi dan diproses di dalam *server*. Dengan adanya bahasa *PHP*, memudahkan pengembang sistem untuk membangun sebuah website yang lebih dinamis dan interaktif. Penyimpanan data yang telah dikirim oleh komputer *client* dilakukan pada *database server*. Data yang telah disimpan ke dalam *server* tersebut akan ditampilkan kembali saat data tersebut diakses (Solichin, 2016). Sebagai Bahasa pemrograman, *PHP* memiliki keunggulan diantaranya:

1. Pemrograman *PHP* dapat diunduh dan dipergunakan secara gratis karena karakteristiknya yang *open source*.
2. *PHP* berlisensi *GNU General Public License* (*GPL*). Dengan kata lain, hal ini menjamin bahwa seluruh versi yang dikeluarkan oleh *PHP* telah didistribusikan secara gratis.
3. *PHP* dikenal sangat efisien. Hal ini karena performanya yang dapat diandalkan untuk dapat melayani jutaan akses per hari tanpa biaya yang mahal.
4. Dukungan basisdata. *PHP* mendukung hampir semua perangkat basisdata, mulai dari *MySQL*, *Oracle*, *Informix*, *MariaDB*, *Interbase*, *PostgreSQL*, *Sybase*, hingga *SQLite*.
5. Memiliki *library* atau pustaka bawaan. *PHP* digunakan secara khusus untuk aplikasi berbasis web. Oleh karena itu, pengguna hanya perlu memanggil dan menggunakan *library* yang dapat langsung digunakan.
6. *Cross Platform*. *PHP* mendukung untuk digunakan dalam berbagai sistem operasi seperti *Windows*, *Mac OS*, *Linux*, *FreeBSD*, *Sun Solaris*, *Unix* dan bahkan dapat berjalan di sistem operasi *Android* melalui proyek *DroidPHP*
7. Mudah dipelajari. Bahasa *PHP* memudahkan pengguna dalam menjalankannya, hal ini karena *PHP* mengadopsi beberapa bahasa pemrograman populer seperti bahasa *C/C++*, *Java* dan *Perl*.

## BAB 4 REKAYASA KEBUTUHAN

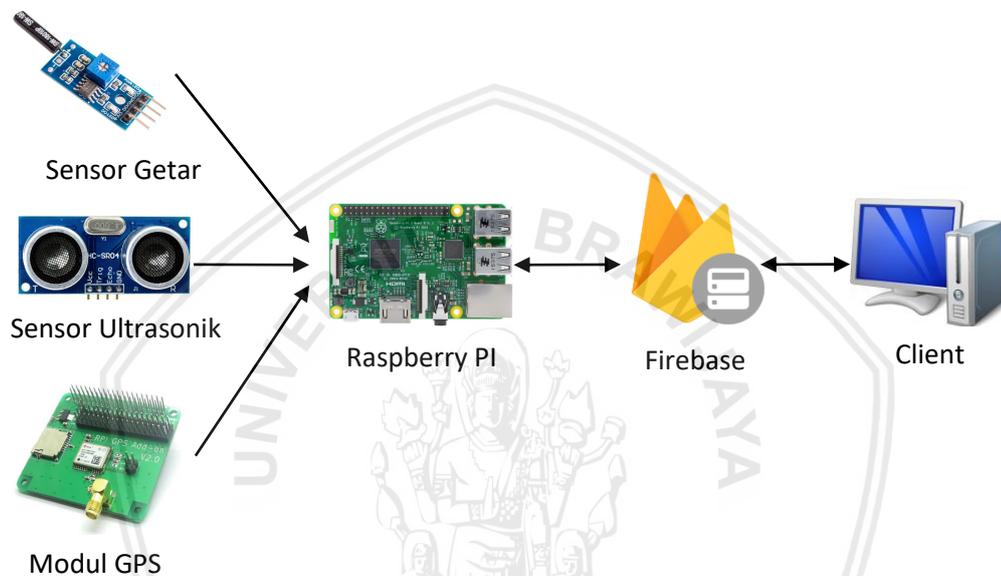
Pada bab ini membahas mengenai rekayasa kebutuhan yang akan dilakukan dalam penelitian dimana kebutuhan pengguna akan dikumpulkan, diulas dan ditetapkan. Hasil dari rekayasa kebutuhan ini akan dijadikan sebagai acuan dalam perancangan Sistem Pelaporan Lubang Jalan Otomatis Berbasis Sistem Embedded. Pembahasan dibagi menjadi deskripsi umum sistem, analisis kebutuhan dan pemodelan kebutuhan. Hasil dari pembahasan bab 4 adalah kebutuhan fungsional dan non-fungsional yang nantinya akan dijadikan acuan untuk membuat perancangan. Pembahasan dari bab ini akan ditampilkan dalam Gambar 4.1 Diagram pohon rekayasa kebutuhan.



Gambar 4.1 Diagram Pohon Rekayasa Kebutuhan

## 4.1 Deskripsi Umum Sistem

Sistem pelaporan kerusakan jalan otomatis merupakan sebuah sistem yang dibangun dengan menggunakan *platform* website. Sistem ini juga menggunakan teknologi sistem *embedded* yang memanfaatkan mikrokomputer *raspberry pi* sebagai inti dari sistem tersebut. Selain itu juga memanfaatkan beberapa modul sensor seperti sensor getar, sensor ultrasonik dan sensor GPS. Sistem ini menerapkan teknologi *Internet of Things* (IoT) dimana pengguna harus terhubung dengan internet. Diagram dari sistem pelaporan kerusakan jalan otomatis dapat dilihat dalam Gambar 4.2 deskripsi sistem.



**Gambar 4.2 Deskripsi Sistem**

Pada penelitian ini, sistem akan ditanamkan pada kendaraan bermotor. Dimana sensor getar diletakkan pada stang roda motor depan dan sensor ultrasonik diletakkan pada bagian bawah kendaraan. Alur dari kerja sistem yaitu sensor akan mengirimkan data kepada *raspberry* dengan format data yang berbeda sesuai dengan fungsi dari setiap sensor. Sensor getar akan mengirimkan data getaran yang diterima alat dalam nilai *True* atau *False*. Sensor ultrasonik menerima dan mengirimkan data kedalaman kerusakan jalan dalam satuan *centimeter* (cm). Serta sensor GPS akan mengirimkan data berupa lokasi yang terdeteksi oleh kedua sensor dalam bentuk koordinat *latitude* dan *longitude*. Data ini akan diterima dan diproses oleh mikrokomputer yang kemudian akan diteruskan dan dikirimkan ke *firebase*. Pada sistem *raspberry*, digunakan bahasa pemrograman *Python* dimana nantinya data yang dikirimkan ke *firebase* berupa data *json*. Kemudian *Client* dengan antarmuka website dikembangkan dengan bahasa pemrograman *PHP* dimana data yang tersimpan pada *firebase* akan diterima dalam bentuk data *json* yang akan ditampilkan ke dalam *dashbord website* berupa tabel data serta memanfaatkan *API Google Maps* untuk melihat lokasi dari kerusakan yang dibaca oleh sensor.

## 4.2 Analisis Kebutuhan

Pada penelitian ini, analisis kebutuhan bertujuan untuk memberikan gambaran mengenai kebutuhan yang harus terpenuhi oleh sistem dan pengguna sistem. Kebutuhan yang didefinisikan mengacu pada studi literatur dan penelitian sebelumnya serta dari hasil wawancara yang menunjukkan permasalahan terkait proses pelaporan kerusakan jalan di Dinas Pekerjaan Umum dan Penataan Ruang (PUPR) Kota Malang. Pelaporan kerusakan jalan yang berajalan diperoleh dari laporan berbagai media *online* maupun cetak seperti website Lapor Kementerian Kominfo dan melalui media sosial. Laporan kerusakan jalan juga bisa berasal dari masyarakat yang mengadu langsung ke dinas maupun dari petugas dinas sendiri yang tanpa sengaja melihat kerusakan jalan. Hal ini membuat petugas sangat bergantung pada laporan yang dilakukan oleh berbagai pihak tersebut.

Penanganan perbaikan jalan dilakukan setelah melakukan survei langsung ke lapangan oleh petugas. Survei dilakukan secara manual dengan menggunakan dokumen kertas. Pihak Dinas PUPR Kota Malang juga menggunakan pihak ketiga (*vendor*) untuk melakukan tugas pengukuran terkait kerusakan jalan. Hal ini menyebabkan petugas lapangan harus melakukan perekapan data yang banyak, yang berakibat pada lambatnya rekapitulasi data. Lambatnya data yang diperoleh dan banyaknya dokumen yang harus direkap terlebih dahulu mengakibatkan penanganan jalan yang rusak menjadi lama, dan kerusakan jalan tidak segera diperbaiki.

Pengambilan keputusan yang didasarkan pada data hasil survei menjadi permasalahan tersendiri. Hasil data terhadap suatu kerusakan sangat dipengaruhi oleh pengamat atau petugas di lapangan. Sehingga pengamat harus menguasai seluruh jenis dan penyebab kerusakan jalan yang ada di lapangan untuk menentukan kategori kerusakan dan prioritas penanganannya. Keterbatasan yang dialami petugas karena subjektifitas ini memberikan dampak terhadap keputusan perbaikan dan penanganan kerusakan, dimana seringkali tidak tepat sasaran. Sedangkan prioritas penanganan tersebut harus dilakukan secara tepat dan sesuai dengan kepentingan. Dari permasalahan yang dihadapi oleh Dinas PUPR Kota Malang, hasil analisis kebutuhan yang diperoleh peneliti dapat dilihat pada Tabel 4.1 Analisis dan spesifikasi kebutuhan.

**Tabel 4.1 Analisis dan spesifikasi kebutuhan**

No	Nama Kebutuhan	Spesifikasi Kebutuhan
1.	Sistem harus mampu mengumpulkan data kerusakan jalan dari sensor.	1. Sistem harus mampu menangkap data dari sensor getar. 2. Sistem harus mampu mengumpulkan data dari sensor ultrasonik dalam bentuk satuan jarak. 3. Sistem harus mampu mengumpulkan data dari modul gps dalam bentuk koordinat lokasi.

**Tabel 4.1 Analisis dan spesifikasi kebutuhan (lanjutan)**

2.	Sistem harus mampu mengirimkan data yang diperoleh dari sensor ke <i>server cloud</i> .	<ol style="list-style-type: none"> <li>1. Sistem harus mampu mengirimkan data yang telah dikumpulkan oleh sensor ke <i>server</i> secara <i>online</i>.</li> <li>2. Data yang dikirimkan ke <i>server</i> berupa satuan <i>centimeter (cm)</i>, serta koordinat <i>latitude</i> dan <i>longitude</i>.</li> </ol>
3.	Sistem harus mampu mengklasifikasikan jenis kerusakan jalan yang terdeteksi oleh sensor.	<ol style="list-style-type: none"> <li>1. Sistem harus mampu menyediakan halaman untuk menampilkan data yang diterima oleh <i>server cloud</i> dalam bentuk tabel yang meliputi, kedalaman kerusakan, lokasi <i>latitude</i> dan <i>longitude</i> serta kategori kerusakan.</li> <li>2. Klasifikasi kerusakan terdiri atas kerusakan ringan, kerusakan sedang dan kerusakan berat.</li> <li>3. Sistem mampu melakukan perekapan data ke dalam bentuk file PDF atau EXEL.</li> </ol>
4.	Sistem harus mampu menampilkan data dari server ke dalam <i>API Google Maps</i> .	<ol style="list-style-type: none"> <li>1. Sistem harus mampu menyediakan halaman untuk menampilkan lokasi kerusakan jalan dengan menggunakan <i>API Google Maps</i> beserta lokasi kerusakan jalan terdeteksi.</li> <li>2. Lokasi kerusakan jalan dapat menampilkan kategori dari kerusakan yang terdeteksi.</li> </ol>

**4.2.1 Identifikasi Aktor**

Aktor merupakan sebuah entitas berupa orang maupun sistem dari luar yang melakukan interaksi dengan sistem yang telah dikembangkan. Aktor memiliki peran yang tidak sama dengan pengguna. Pengguna biasa dapat memainkan sejumlah peran berbeda saat menggunakan sistem. Sedangkan aktor mewakili kelas entitas eksternal yang hanya memainkan satu peran. Aktor yang terlibat dan berinteraksi dengan sistem ini di paparkan pada Tabel 4.2 Daftar Aktor.

**Tabel 4.2 Daftar Aktor**

Aktor	Deskripsi
Admin	Seseorang yang mampu mengelola data yang ada pada website terkait dengan pelaporan kerusakan jalan yang diterima dari <i>server cloud</i> . Admin juga dapat memantau data laporan yang masuk serta melakukan perekapan data.

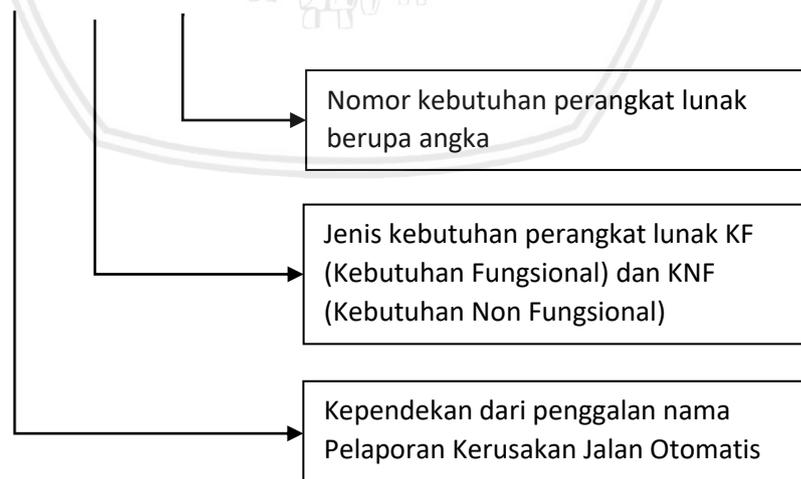
Tabel 4.2 Daftar Aktor (lanjutan)

Sensor Getar	Sebuah alat yang mampu menerima data terkait getaran yang terjadi di jalan dan mengirimkan data tersebut ke mikromputer untuk diproses. Data yang dihasilkan dalam <i>boolean True</i> atau <i>False</i> .
Sensor Ultrasonik	Sebuah alat yang mampu menerima data terkait kedalaman kerusakan jalan dan mengirimkan data tersebut ke mikromputer untuk diproses dan diteruskan kembali ke <i>server cloud</i> . Data yang dihasilkan dalam bentuk satuan centimeter (cm).
Sensor GPS	Sebuah alat yang mampu menerima data terkait lokasi jalan dan mengirimkan data tersebut ke mikromputer untuk diproses dan diteruskan kembali ke <i>server cloud</i> . Data yang dihasilkan dalam bentuk koordinat <i>latitude</i> dan <i>longitude</i> .

#### 4.2.2 Aturan Penomoran Kode Kebutuhan Perangkat Lunak

Aturan penomoran kode kebutuhan perangkat lunak bertujuan untuk memudahkan dalam pemberian nama dari kebutuhan fungsional. Diharapkan dengan aturan penomoran mampu menyelaraskan antara nama kebutuhan fungsional dengan sistem yang dibangun. Aturan penomoran terdiri atas 3 bagian utama yaitu nama dari sistem, jenis kebutuhan dan nomor kebutuhan. Jenis kebutuhan terdiri atas kebutuhan fungsional dan kebutuhan non fungsional. Adapun penulisan penomoran kode kebutuhan perangkat lunak dapat dilihat dalam Gambar 4.3 Aturan Penomoran.

Kode: PKJO-KF/KNF-XXX



Gambar 4.3 Aturan Penomoran

### 4.2.3 Analisis Kebutuhan Fungsional Sistem

Analisis kebutuhan fungsional merupakan salah satu fase dalam rekayasa perangkat lunak. Fase ini bertujuan untuk menentukan kebutuhan fungsional atau fungsi dari suatu sistem yang harus dipenuhi. Kebutuhan fungsional berisi tentang kumpulan proses yang dapat dilakukan oleh sistem. Analisis kebutuhan fungsional yang telah diperoleh dapat dilihat pada Tabel 4.3 spesifikasi kebutuhan fungsional sistem.

**Tabel 4.3 Spesifikasi kebutuhan fungsional sistem**

Kode Fungsi	Kebutuhan Fungsional	Pengguna	Deskripsi
PKJO-KF-001	Menangkap data getar	Sensor Getar	Sistem harus mampu menangkap data getar dari sensor dalam bentuk <i>boolean True</i> atau <i>False</i> .
PKJO-KF-002	Menangkap data jarak	Sensor Ultrasonik	Sistem harus mampu menangkap data jarak dari sensor dalam bentuk satuan <i>centimeter (cm)</i> .
PKJO-KF-003	Menangkap data koordinat	Sensor GPS	Sistem harus mampu menangkap data koordinat lokasi dari sensor dalam data <i>latitude</i> dan <i>longitude</i> .
PKJO-KF-004	Mengirimkan data	Sensor Getar	Sistem harus mampu mengirimkan data yang diperoleh oleh sensor ultrasonik dan sesnsor gps ke <i>server cloud</i> .
PKJO-KF-005	Klasifikasi kerusakan	Sensor Ultrasonik	Sistem harus mampu mengklasifikasikan kategori kerusakan jalan yang terdiri dari kerusakan ringan, sedang dan berat.
PKJO-KF-006	Menampilkan data	Admin	Sistem harus mampu menampilkan data yang telah disimpan oleh <i>cloud</i> dalam bentuk tabel.
PKJO-KF-007	Mencetak laporan	Admin	Sistem harus mampu mencetak laporan data kerusakan jalan dalam format PDF atau EXEL.
PKJO-KF-008	Menampilkan lokasi	Admin	Sistem harus mampu memetakan titik dengan <i>API Google maps</i> .

#### 4.2.4 Analisis Kebutuhan Non-Fungsional Sistem

Analisis kebutuhan non-fungsional merupakan kebutuhan yang harus terpenuhi untuk menunjang jalannya sistem. Kebutuhan non-fungsional mencakup batasan layanan yang terdapat pada sistem. Kebutuhan ini memiliki karakteristik berupa kemampuan sistem untuk mendukung kebutuhan fungsional sistem. Kebutuhan non-fungsional pada sistem ini terdiri atas satu kebutuhan yaitu *correctness*. Analisis kebutuhan non-fungsional yang telah diperoleh dapat dilihat pada Tabel 4.4 spesifikasi kebutuhan non-fungsional sistem.

**Tabel 4.4 Spesifikasi kebutuhan non-fungsional sistem**

Kode Fungsi	Kebutuhan Non-Fungsional	Deskripsi
PKJO-KNF-001	<i>Correctness</i>	Sistem harus mampu diuji kebenarannya terkait data jarak dari sensor ultrasonik, lokasi <i>latitude</i> dan <i>longitude</i> serta penentuan klasifikasi kerusakan jalan.

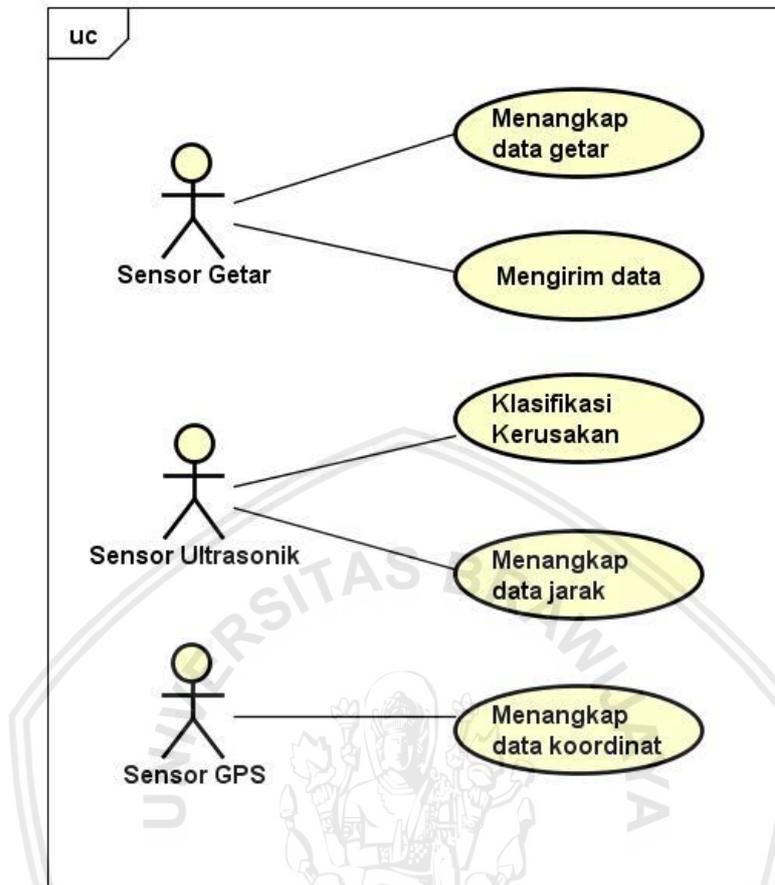
#### 4.3 Pemodelan Kebutuhan

Pemodelan kebutuhan merupakan sebuah cara dalam mendeskripsikan kebutuhan fungsional sistem ke dalam bentuk diagram. Hal ini bertujuan untuk menjembatani antara pengembang dengan pengguna sistem dalam memahami kebutuhan sistem. Selain itu juga untuk mencegah agar tidak terjadi kesalahan pemahaman satu sama lain. Pemodelan kebutuhan pada bab ini menjadi dasar dalam melakukan perancangan sistem. Serta menjadi acuan dalam melakukan validasi kebutuhan.

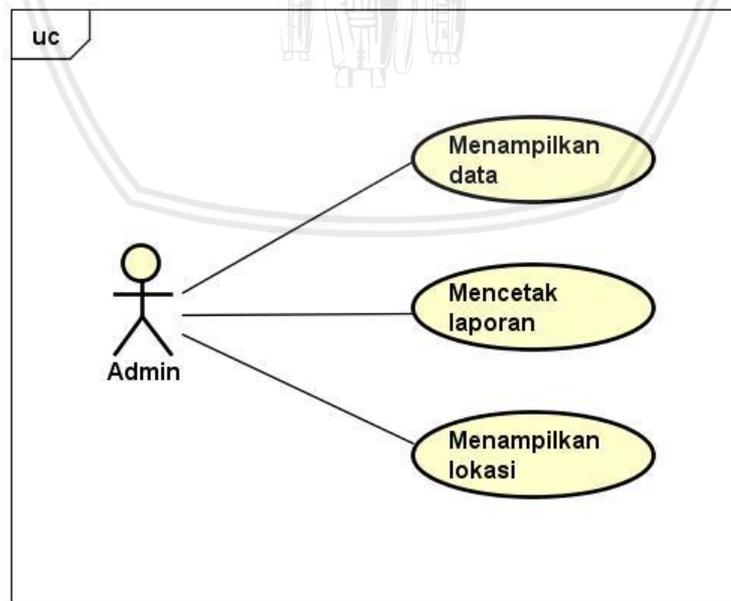
##### 4.3.1 Use Case Diagram

*Use Case Diagram* merupakan salah satu jenis pemodelan *Unified Markup Language* (UML) yang berfokus pada interaksi yang dilakukan oleh aktor maupun sistem serta mampu memberikan deskripsi terkait tipe interaksi yang terjadi. Dalam membuat *use case diagram* dilakukan dengan mengidentifikasi berbagai jenis orang (atau perangkat) yang menggunakan sistem atau produk. Jadi *use case diagram* adalah sebuah diagram yang menggambarkan interaksi yang dilakukan oleh aktor maupun sistem serta mampu memberikan deskripsi terkait tipe interaksi yang terjadi.

Pada diagram ini, terdapat 4 aktor yaitu admin, sensor getar, sensor ultrasonik, dan sensor GPS. Serta terdapat 8 *use case*. Pemodelan *use case diagram* dibagi menjadi dua, yaitu *use case diagram sistem embedded* dan *use case diagram website*. Pada *use case diagram sistem embedded* terdapat lima buah *use case* yang dapat dilakukan oleh tiga aktor sesuai dengan fungsinya. Sedangkan pada *use case diagram website* terdapat tiga buah *use case* yang dapat diperankan oleh aktor admin sesuai dengan fungsinya. *Use case diagram sistem embedded* dan *website* ditampilkan dalam Gambar 4.4 *use case diagram sistem embedded* dan Gambar 4.5 *use case diagram website*.



Gambar 4.4 Use Case Diagram Sistem Embedded



Gambar 4.5 Use Case Diagram Website

### 4.3.2 Use Case Scenario

*Use case scenario* merupakan kumpulan dari skenario yang menggambarkan interaksi pada sebuah *use case*. Skenario ini bertujuan untuk menjelaskan secara detail terkait sistem yang dibangun dan interaksi yang terjadi dengan aktor. *Use case scenario* pada penelitian ini dapat dilihat pada Tabel 4.5 sampai 4.15.

*Use case* menangkap data getar merupakan aktivitas yang dilakukan oleh aktor berupa sensor getar. Sensor getar ini berfungsi untuk mendeteksi getaran yang ada pada jalan raya. Sistem akan menerima data getaran yang ditangkap oleh sensor dalam *boolean True* atau *False*. *Use case scenario* menangkap data getar ditampilkan pada Tabel 4.5.

**Tabel 4.5 Use Case Scenario menangkap data getar**

PKJO-KF-001	<i>Use case</i> menangkap data getar	
Tujuan	Menerima data berupa getaran pada sensor	
Aktor	Sensor Getar	
Kondisi Awal	Sensor sudah terpasang pada kendaraan dan terkoneksi dengan internet	
Skenario Utama	Aktor	Sistem
	1. Sensor getar dijalankan pada kendaraan dan melewati jalan yang rusak. 2. Sensor getar mendeteksi getaran pada jalan dan membaca data getaran.	3. Sistem menerima data sensor berupa <i>boolean True</i> atau <i>False</i> .
Skenario Alternatif	2a. Jika sensor tidak mendeteksi adanya getaran, maka sistem akan menerima data berupa <i>boolean</i> bernilai <i>False</i>	
Kondisi Akhir	Sistem mendeteksi terjadi getaran	

*Use case* menangkap data jarak merupakan aktivitas yang dilakukan oleh aktor berupa sensor ultrasonik. Sensor ultrasonik ini berfungsi untuk mengukur kedalaman dari kerusakan yang ada pada jalan raya. Sistem akan menerima angka data besaran jarak yang ditangkap oleh sensor dalam satuan *centimeter* (cm). *Use case scenario* menangkap data jarak ditampilkan pada Tabel 4.6.

**Tabel 4.6 Use Case Scenario menangkap data jarak**

PKJO-KF-002	<i>Use case</i> menangkap data jarak	
Tujuan	Menerima data berupa jarak dari sensor	
Aktor	Sensor Ultrasonik	
Kondisi Awal	Sensor sudah terpasang pada kendaraan dan terkoneksi dengan internet	

**Tabel 4.6 Use Case Scenario menangkap data jarak (lanjutan)**

Skenario	Aktor	Sistem
Skenario Utama	<ol style="list-style-type: none"> <li>1. Sensor getar mendeteksi getaran pada jalan.</li> <li>2. Sensor ultrasonik melakukan pengukuran kedalaman kerusakan jalan.</li> <li>3. Sensor ultrasonik membaca besar jarak kedalaman kerusakan jalan.</li> </ol>	<ol style="list-style-type: none"> <li>4. Sistem menerima data sensor berupa angka satuan <i>centimeter</i> (cm).</li> </ol>
Skenario Alternatif	1a. Jika sensor getar tidak mendeteksi getaran, maka sensor ultrasonik tidak melakukan aksi apapun	
Kondisi Akhir	Sistem menerima data sensor berupa angka satuan <i>centimeter</i> (cm)	

*Use case* menangkap data koordinat merupakan aktivitas yang dilakukan oleh aktor berupa sensor GPS. Sensor GPS ini berfungsi untuk mendapatkan koordinat lokasi dari kerusakan jalan yang terdeteksi. Sistem akan menerima data koordinat lokasi dalam bentuk data *latitude* dan *longitude*. *Use case scenario* menangkap data koordinat ditampilkan pada Tabel 4.7.

**Tabel 4.7 Use Case Scenario menangkap data koordinat**

PKJO-KF-003	<i>Use case</i> menangkap data koordinat	
Tujuan	Menerima data berupa koordinat lokasi dari sensor	
Aktor	Sensor GPS	
Kondisi Awal	Sensor sudah terpasang pada kendaraan dan terkoneksi dengan sinyal antena	
Skenario	Aktor	Sistem
Skenario Utama	<ol style="list-style-type: none"> <li>1. Sensor getar mendeteksi getaran pada jalan.</li> <li>2. Sensor GPS melakukan pencarian koordinat lokasi dari kerusakan jalan yang terdeteksi</li> <li>3. Sensor GPS membaca koordinat lokasi dari kerusakan jalan.</li> </ol>	<ol style="list-style-type: none"> <li>4. Sistem menerima data sensor berupa koordinat lokasi dalam bentuk <i>latitude</i> dan <i>longitude</i>.</li> </ol>



**Tabel 4.7 Use Case Scenario menangkap data koordinat (lanjutan)**

Skenario Alternatif	1a. Jika sensor getar tidak mendeteksi getaran, maka sensor ultrasonik tidak melakukan aksi apapun
Kondisi Akhir	Sistem menerima data sensor berupa koordinat lokasi dalam bentuk <i>latitude</i> dan <i>longitude</i>

Setelah sensor getar menangkap *trigger* dalam bentuk getaran, maka akan mengembalikan nilai boolean *True*. Ketika *boolean* bernilai *True* maka akan mengirimkan data lokasi kerusakan pada jalan raya ke *server cloud*. Data yang dikirimkan dalam bentuk *array* data yang berisi jarak, *latitude*, *longitude* dan kategori kerusakan yang nantinya akan disimpan ke dalam *server*. Data dari *server* kemudian akan diakses oleh admin. *Use case scenario* mengirim data koordinat ditampilkan pada Tabel 4.8

**Tabel 4.8 Use Case Scenario mengirim data**

PKJO-KF-004	<i>Use case</i> mengirim data	
Tujuan	Mengirim data yang diperoleh sensor ke <i>server cloud</i>	
Aktor	Sensor Getar	
Kondisi Awal	Sensor getar dalam keadaan menyala	
Skenario Utama	Aktor	Sistem
	1. Sensor menerima <i>trigger</i> berupa getaran	2. Sistem mengembalikan nilai boolean <i>True</i> 3. Sistem mengirimkan data yang telah diperoleh oleh sensor ke dalam bentuk data <i>array</i> 4. Data tersimpan di <i>server cloud</i> ke dalam bentuk tabel
Skenario Alternatif	1a. Jika sistem tidak menerima <i>trigger</i> getaran, maka data tidak dikirimkan ke <i>server cloud</i> dan mengembalikan nilai boolean <i>False</i>	
Kondisi Akhir	Data tersimpan di <i>server cloud</i>	

Sistem juga dapat melakukan klasifikasi kerusakan jalan berdasarkan data yang diperoleh oleh sensor. Klasifikasi kerusakan jalan mengacu pada kategori yang terdapat dalam literatur. Model pengklasifikasian dengan menggunakan parameter data kategori yang sudah dijelaskan dalam literatur. *Use case* klasifikasi kerusakan ditampilkan pada Tabel 4.9.

**Tabel 4.9 Use Case Scenario klasifikasi kerusakan**

PKJO-KF-005	<i>Use case</i> klasifikasi kerusakan
Tujuan	Melakukan pengklasifikasian terkait kerusakan jalan
Aktor	Sensor Ultrasonik
Kondisi Awal	Sensor Ultrasonik menangkap data jarak

**Tabel 4.9 Use Case Scenario** klasifikasi kerusakan (lanjutan)

Skenario Utama	Aktor	Sistem
		1. Sistem menerima data dari sensor ultrasonik 2. Sistem melakukan penyeleksian kategori dari kerusakan jalan berdasarkan literatur 3. Sistem memberikan label terkait jenis dari kerusakan dan disimpan kedalam kolom klasifikasi pada tabel 4. Sistem menampilkan hasil klasifikasi di halaman website admin
Skenario Alternatif	-	
Kondisi Akhir	Sistem mengenali klasifikasi kerusakan jalan	

Setelah semua data dari sensor tersimpan dalam *server cloud*, maka data tersebut akan ditampilkan di halaman website yang bisa diakses oleh admin. Pada halaman ini, sistem akan menampilkan data dari *server* ke dalam bentuk tabel. *Use case* menampilkan data dapat ditampilkan pada tabel 4.10.

**Tabel 4.10 Use Case Scenario** menampilkan data

PKJO-KF-006	<i>Use case</i> menampilkan data	
Tujuan	Menampilkan data yang tersimpan dalam <i>server cloud</i> ke halaman website admin	
Aktor	Admin	
Kondisi Awal	Admin membuka aplikasi dan terkoneksi dengan internet	
Skenario Utama	Aktor	Sistem
	1. Admin membuka halaman website	2. Sistem menampilkan data dalam bentuk tabel
Skenario Alternatif	-	
Kondisi Akhir	Admin dapat melihat data yang tersimpan di server cloud pada halaman website	

Selain melakukan peninjauan secara langsung, admin juga dapat melakukan mencetak laporan. Laporan tersebut berisi data dari server yang ditampilkan pada tabel kerusakan jalan. Hal ini bertujuan untuk memudahkan admin dalam merekap data kerusakan jalan. *Use case* mencetak laporan ditampilkan pada tabel 4.11.

**Tabel 4.11 Use Case Scenario** mencetak laporan

PKJO-KF-007	Use case mencetak laporan	
Tujuan	Melakukan perekapan data dengan mengunduh data yang tersimpan pada server cloud	
Aktor	Admin	
Kondisi Awal	Admin membuka membuka halaman dashboard	
Skenario Utama	Aktor	Sistem
	<ol style="list-style-type: none"> <li>1. Admin membuka halaman website.</li> <li>2. Admin menekan tombol cetak laporan PDF atau EXEL.</li> <li>3. Admin memilih direktori lokasi untuk menyimpan file data.</li> <li>4. Admin menekan tombol OK.</li> </ol>	<ol style="list-style-type: none"> <li>3. Sistem menampilkan jendela <i>browse</i> untuk menyimpan file.</li> <li>5. Sistem mengunduh dan menyimpan file data.</li> </ol>
Skenario Alternatif	4a. Jika admin menekan tombol <i>cancel</i> , maka file tidak tersimpan ke dalam PC admin	
Kondisi Akhir	Admin berhasil mengunduh dan menyimpan file data	

Untuk memudahkan peninjauan lokasi yang dilakukan oleh admin secara *online*, juga terdapat peta digital. Peta digital ini memanfaatkan *API Google Maps* untuk menunjukkan titik koordinat lokasi kerusakan jalan. Titik koordinat lokasi ini mengacu pada data yang diperoleh oleh sensor GPS. *Use case* menampilkan lokasi ditampilkan pada Tabel 4.12.

**Tabel 4.12 Use Case Scenario** menampilkan lokasi

PKJO-KF-009	Use case menampilkan lokasi	
Tujuan	Menampilkan titik kerusakan jalan dengan bantuan peta digital	
Aktor	Admin	
Kondisi Awal	Data lokasi dari GPS telah tersimpan dalam <i>database server</i>	
Skenario Utama	Aktor	Sistem
	<ol style="list-style-type: none"> <li>1. Admin membuka halaman website</li> </ol>	<ol style="list-style-type: none"> <li>2. Sistem mengambil data dari server</li> <li>3. Sistem menampilkan lokasi berdasarkan data yang diperoleh dari server dalam bentuk titik lokasi</li> </ol>



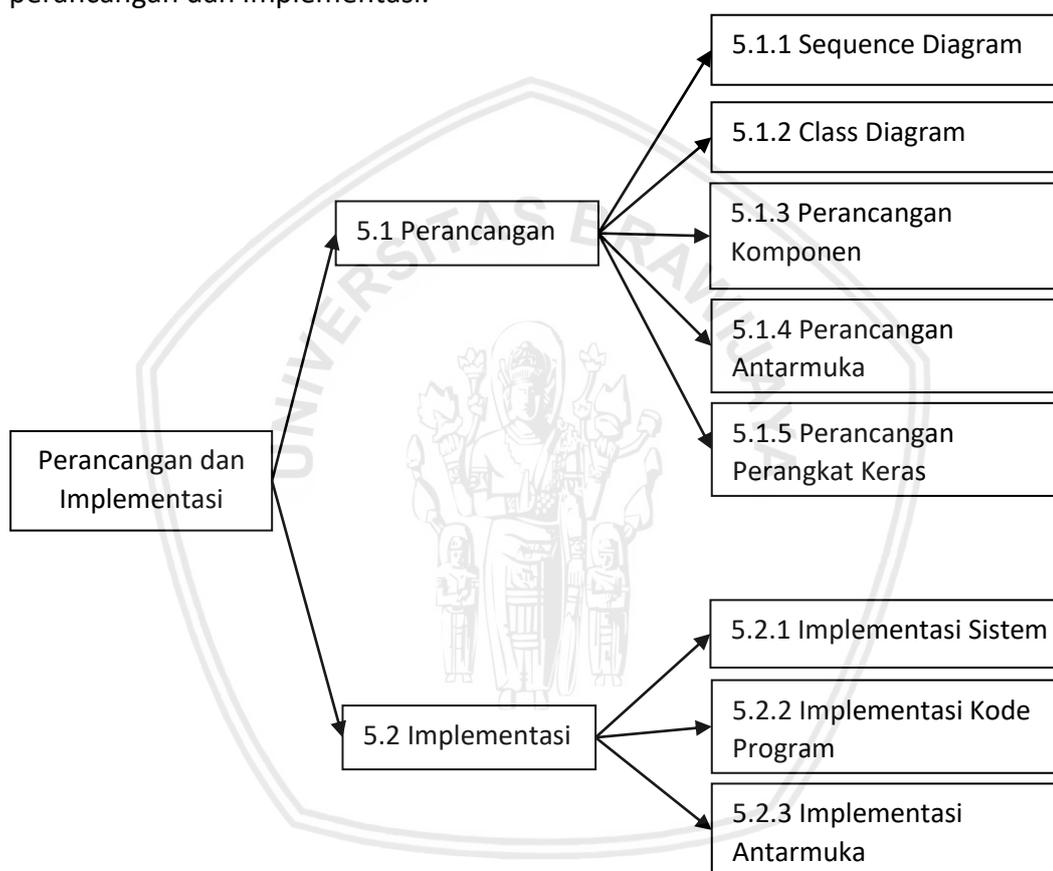
Tabel 4.12 *Use Case Scenario* menampilkan lokasi (lanjutan)

Skenario Alternatif	-
Kondisi Akhir	Admin dapat melihat titik lokasi sesuai dengan data yang tersimpan di <i>server</i>



## BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab 5 membahas mengenai perancangan dan implementasi mengacu berdasarkan hasil rekayasa kebutuhan. Fase ini merepresentasikan rancangan dari sisi sistem berdasarkan kebutuhan fungsional dan non-fungsional dari Sistem Pelaporan Lubang Jalan Otomatis Berbasis Sistem Embedded. Pembahasan dibagi menjadi perancangan dan implementasi. Hasil dari pembahasan bab 5 adalah produk sistem yang akan dilakukan pengujian. Pembahasan dari bab ini akan ditampilkan dalam Gambar 5.1 Diagram pohon perancangan dan implementasi.



Gambar 5.1 Diagram pohon perancangan dan implementasi.

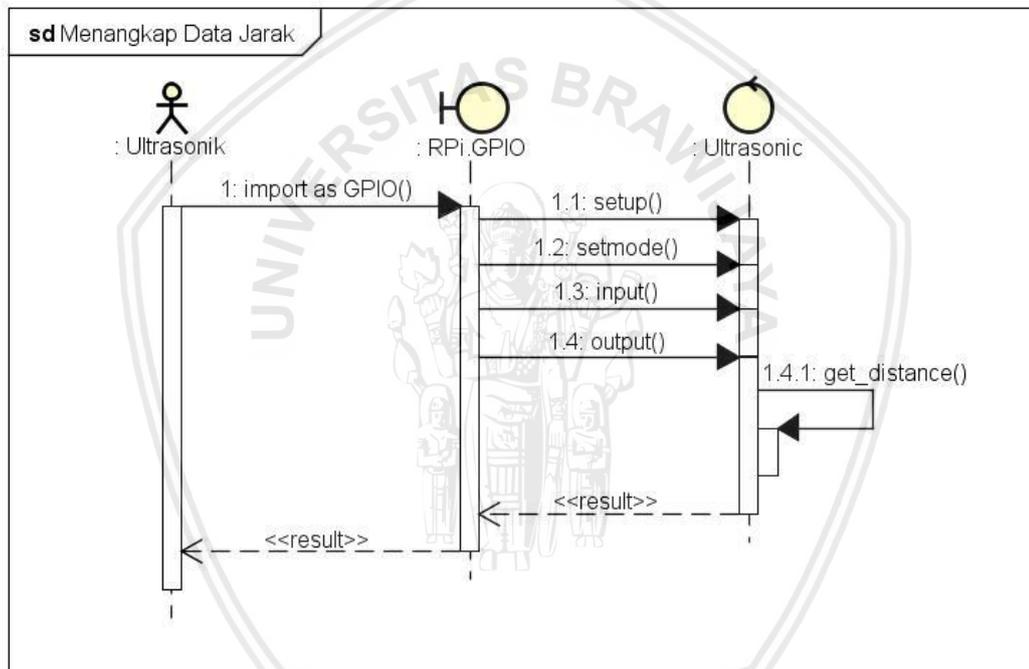
### 5.1 Perancangan

Perancangan sistem pada bab ini akan menjelaskan beberapa diagram yang dibuat berdasarkan pada rekayasa kebutuhan pada bab sebelumnya. Pada tahap ini akan dihasilkan perancangan *hardware* berupa rangkaian dan konfigurasi alat dan perancangan sistem berupa *sequence diagram*, *class diagram*, perancangan komponen, perancangan antarmuka dan perancangan perangkat keras. Perancangan antarmuka bertujuan untuk memberikan gambaran tampilan sistem yang akan dikembangkan.

## 5.1.1 Sequence Diagram

### 5.1.1.1 Sequence Diagram menangkap data jarak

Aktor berupa sensor ultrasonik melakukan penangkapan data. Pada diagram ini terdapat aktor ultrasonik, *boundary RPI.GPIO*, *controller ultrasonic*. Sensor ultrasonik berada pada kondisi menyala dan terhubung dengan daya atau listrik. Kemudian *boundary RPI.GPIO* memanggil *method setup()*, *setmode()*, *input()* dan *output()* untuk menginisialisasi *pin* dan *modul* dari sensor ultrasonik. Kemudian *controller ultrasonic* menjalankan *method get\_distance()* untuk mengirimkan sinyal ketika melewati sebuah objek dan mengembalikan nilai tersebut. Kemudian akan menangkap nilai dari sinyal sensor tersebut dan mengembalikan nilai dalam bentuk angka. *Sequence diagram* yang menggambarkan interaksi objek yang terlibat ditampilkan dalam Gambar 5.2 *Sequence Diagram* menangkap data jarak.



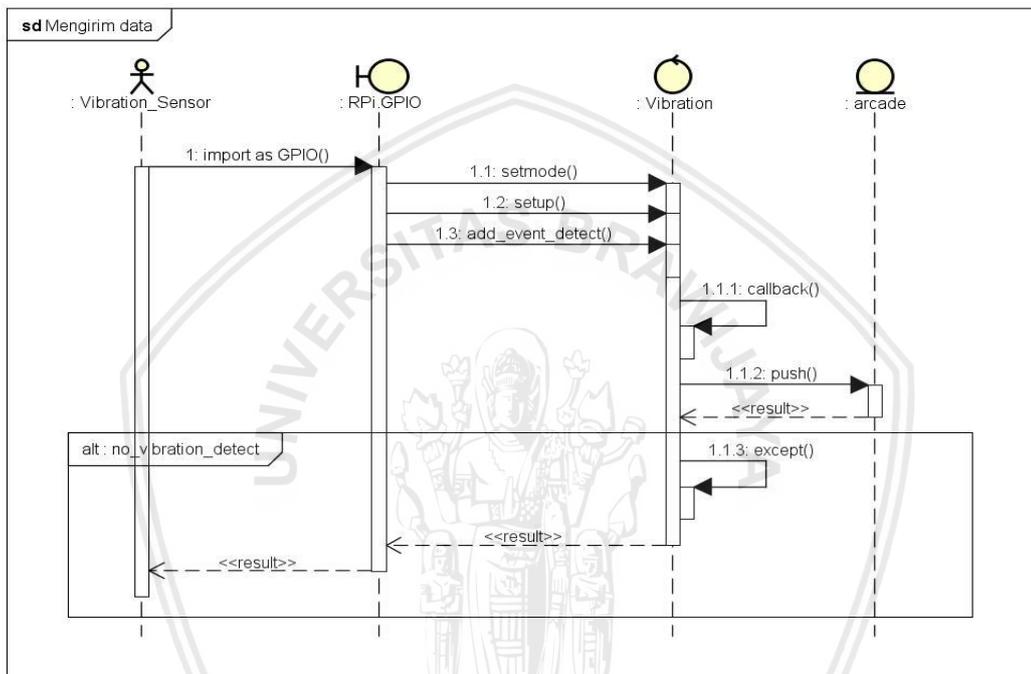
Gambar 5.2 *Sequence Diagram* menangkap data jarak

### 5.1.1.2 Sequence Diagram mengirim data

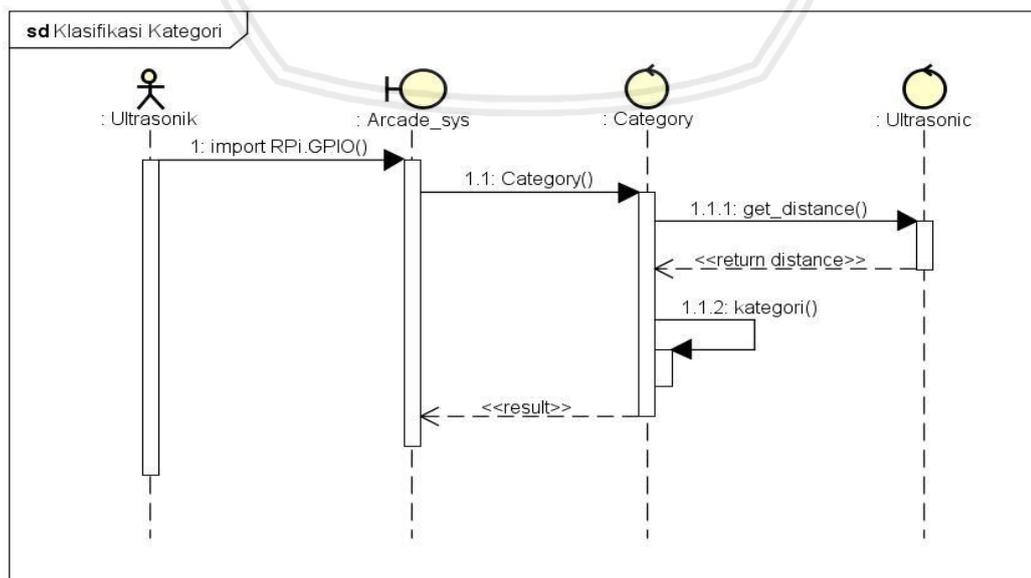
Aktor berupa sensor getar yang melakukan pengiriman data. Pada diagram ini terdapat aktor sensor getar, *boundary RPI.GPIO*, *controller vibration* dan *entity arcade*. *Boundary RPI.GPIO* memanggil *method setup()*, *setmode()* dan *add\_event\_detect()* untuk menginisialisasi *pin* dan *modul* dari sensor getar. Kemudian *controller vibration* memanggil *method callback()* untuk menjalankan sensor getar dan mengirimkan data ke *server cloud*. Terdapat satu alternatif dimana jika sensor tidak mendeteksi getaran, maka tidak mengirimkan data. *Sequence diagram* yang menggambarkan interaksi objek yang terlibat ditampilkan dalam Gambar 5.3 *Sequence Diagram* mengirim data jarak.

### 5.1.1.3 Sequence Diagram klasifikasi kategori

Aktor merupakan sensor ultrasonik yang dapat melakukan klasifikasi kategori. Pada diagram ini terdapat aktor ultrasonik, *boundary arcade\_sys*, *controller category*, dan *controller ultrasonic*. *Boundary arcade\_sys* memanggil *constructor Category()* untuk menjalankan *controller category*. *Controller Category()* akan memanggil *method get\_distance()* dari *controller ultrasonic* untuk mendapatkan nilai dari variabel *distance*. Kemudian *controller category()* akan memanggil *method kategori()* pada kelasnya dan mengembalikan nilai kategori. *Sequence diagram* yang menggambarkan interaksi objek yang terlibat ditampilkan dalam Gambar 5.4 *Sequence Diagram* klasifikasi kategori.



Gambar 5.3 Sequence Diagram mengirim data



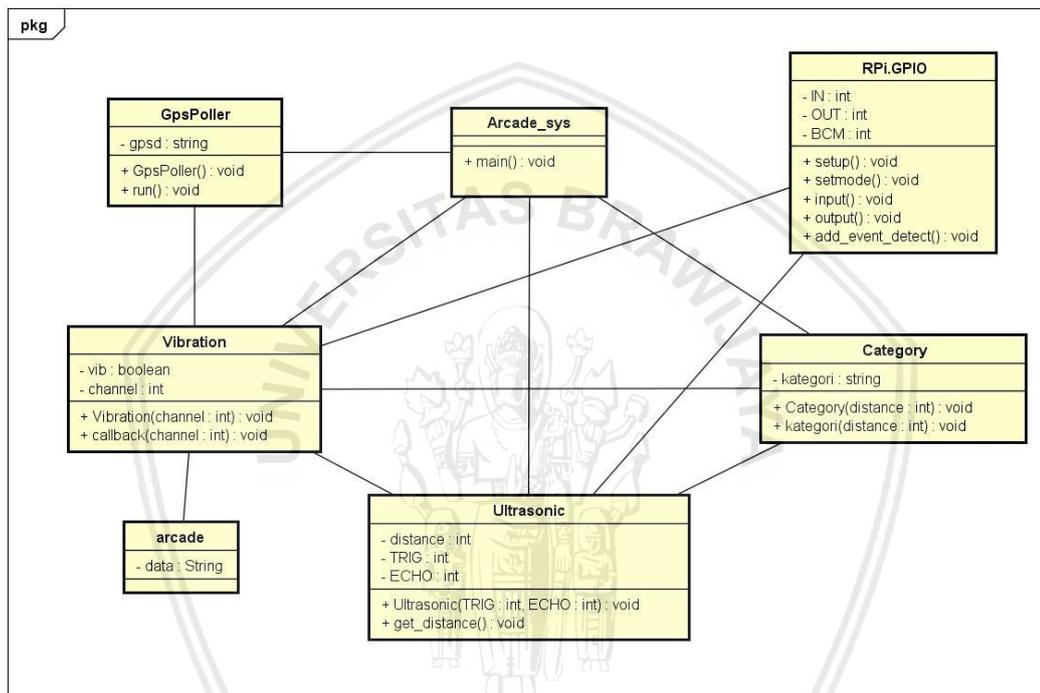
Gambar 5.4 Sequence Diagram klasifikasi kategori



## 5.1.2 Class Diagram

### 5.1.2.1 Class Diagram Sistem Embedded

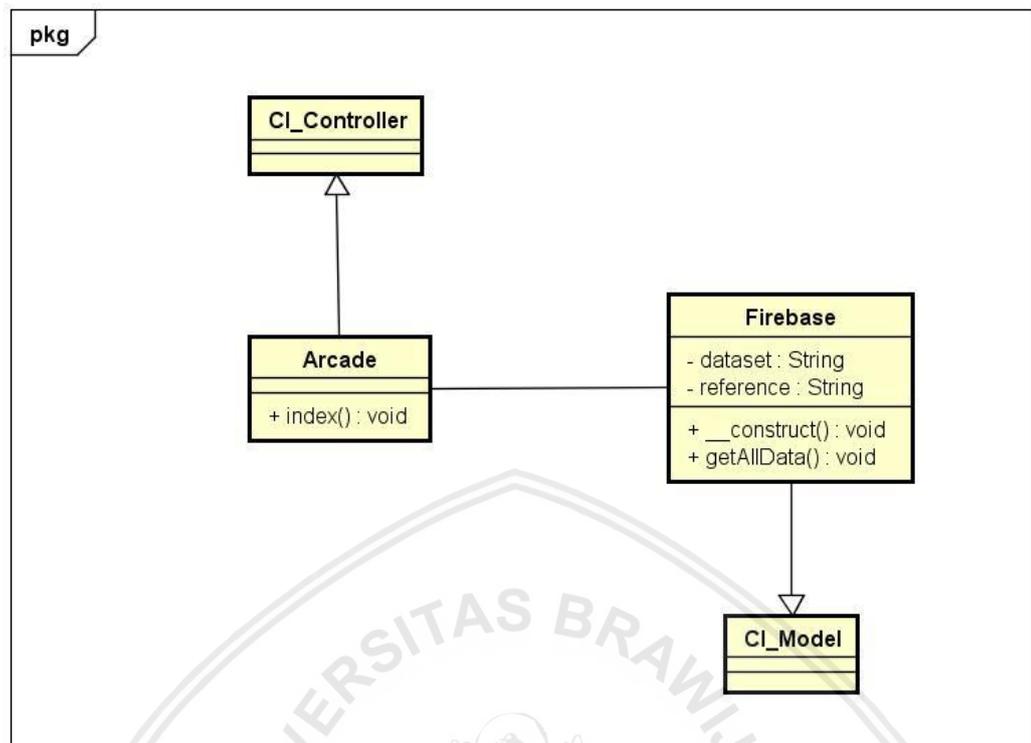
*Class Diagram* merupakan diagram yang menggambarkan kelas dan relasi pada suatu sistem yang kemudian akan diimplementasikan dalam pembangunan suatu sistem. *Class diagram* yang dirancang pada bab ini menerapkan *Object Oriented Programming* (OOP). Pada *class diagram* sistem embedded ini terdapat 7 class yang terdiri atas class *Arcade\_sys*, *RPI.GPIO*, *Ultrasonic*, *GpsPoller*, *Vibration*, *Arcade* dan *Category*. *Class diagram* yang menggambarkan interaksi antar class yang terlibat ditampilkan pada gambar 5.5 Rancangan *Class Diagram* Sistem Embedded.



Gambar 5.5 Rancangan *Class Diagram* Sistem Embedded

### 5.1.2.2 Class Diagram Website

*Class Diagram* merupakan diagram yang menggambarkan kelas dan relasi pada suatu sistem yang kemudian akan diimplementasikan dalam pembangunan suatu sistem. *Class diagram* yang dirancang pada bab ini menerapkan *Object Oriented Programming* (OOP). Pada *class diagram* website ini terdapat 2 class controller yang terdiri atas Class *CI\_Controller* dan *Dashboard*. Serta 2 class model yang terdiri atas *CI\_Model* dan *m\_data*. *Class diagram* yang menggambarkan interaksi antar class yang terlibat ditampilkan pada gambar 5.6 Rancangan *Class Diagram* Website.



Gambar 5.6 Rancangan *Class Diagram* Website

### 5.1.3 Perancangan Komponen

Pada perancangan komponen menjelaskan mengenai rincian dari sistem yang akan akan dibangun pada setiap komponennya. Perancangan komponen berisi mengenai alur dari setiap komponen berdasarkan algoritmanya. Pada penelitian ini terdapat tiga *method* yang akan dibahas. *Method* tersebut antara lain *method get\_distance*, *method callback* dan *method kategori*. Berikut adalah hasil pemaparan perancangan komponen tiga algoritme *method*.

#### 5.1.3.1 Perancangan komponen *method get\_distance*

Nama *class*: *Ultrasonic*

Nama *method*: *get\_distance*

Tabel 5.1 Perancangan komponen *method get\_distance*

Pseudocode <i>get_distance</i>	
1	SET method declaration
2	SET GPIO output TRIG VALUE TRUE
3	SET time.sleep VALUE 0.0001
4	SET GPIO output TRIG VALUE FALSE
5	WHILE GPIO input when ECHO VALUE FALSE:
6	SET start VALUE this time
7	WHILE GPIO input when ECHO VALUE TRUE:
8	SET end VALUE this time
9	SET sig_time VALUE end MIN start
10	SET distance VALUE sig_time CROSS converter_to_cm MIN threshold_distance
11	SET distance VALUE distance with round 4
12	RETURN distance

### 5.1.3.2 Perancangan komponen *callback*

Nama *class*: *Vibration*

Nama *method*: *callback*

**Tabel 5.2 Perancangan komponen *method callback***

Pseudocode method callback	
1	SET method declaration
2	GET objek ultra VALUE class UltraSonic
3	GET objek cat VALUE class Category
4	IF GPIO input GET nothing:
5	PRINT "no!"
6	vib VALUE FALSE
7	ELSE:
8	PRINT "Data terbaca!"
9	vib VALUE TRUE
10	SET array data VALUE jarak, latitude, longitude, kategori
11	IF gpsp not alive:
12	START gpsp
13	SET array data latitude
14	SET array data longitude
15	IF latitude VALUE 0 or longitude VALUE 0 or kategori VALUE "Gundukan":
16	PRINT "Don't Send Data"
17	ELSE:
18	SEND array data to database
19	PRINT "Send Data"
20	END IF
21	ELSE
22	SET gpsp VALUE FALSE
23	JOIN gpsp
24	END IF
25	RETURN vib

### 5.1.3.3 Perancangan komponen kategori

Nama *class*: *Category*

Nama *method*: *kategori*

**Tabel 5.3 Perancangan komponen *method kategori***

Pseudocode method kategori	
1	SET method declaration GET VALUE distance
2	SET kategori VALUE NULL
3	IF distance < 0:
4	PRINT "Gundukan"
5	SET kategori VALUE "Gundukan"
6	ELIF distance > 1.3 AND distance <= 2.5:
7	PRINT "Ringan"
8	SET kategori VALUE "Ringan"
9	ELIF distance > 2.5 AND distace <= 5.0:
10	PRINT "Sedang"
11	SET kategori VALUE "Sedang"
12	ELIF distance > 5.0:
13	PRINT "Berat"
14	SET kategori VALUE "Berat"
15	RETURN kategori

#### 5.1.4 Perancangan Antarmuka

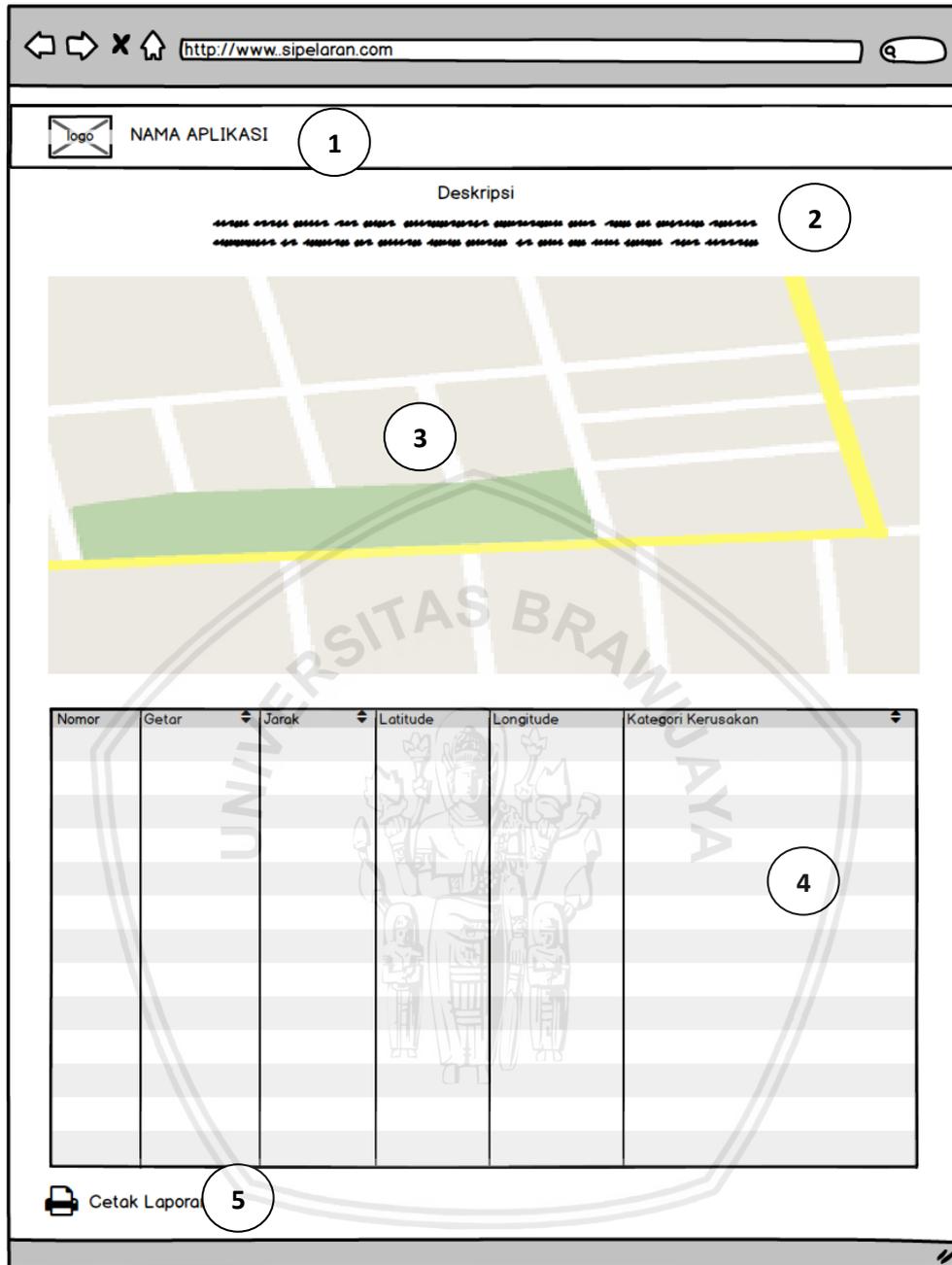
Perancangan antarmuka bertujuan untuk memberikan gambaran terkait tampilan sistem yang akan diimplementasikan. Pada penelitian ini, antarmuka yang digunakan adalah halaman website yang ditampilkan kepada aktor admin. Pada halaman ini terdapat 5 bagian utama, yaitu header, deskripsi, google maps, tabel dan icon print. Perancangan antarmuka halaman website dapat dilihat dalam Gambar 5.6 rancangan antarmuka halaman website.

Rancangan antarmuka halaman dashboard merupakan antarmuka yang dimiliki oleh sistem untuk dapat digunakan oleh admin. Pada halaman ini admin dapat melakukan beberapa aktivitas sesuai dengan peran yang telah didefinisikan dalam kebutuhan fungsional. Pada halaman website akan menampilkan tabel data kerusakan jalan. Selain itu juga menampilkan peta digital dengan memanfaatkan API google maps serta beberapa informasi pendukung. Penjelasan dari rancangan antarmuka halaman dashboard yaitu:

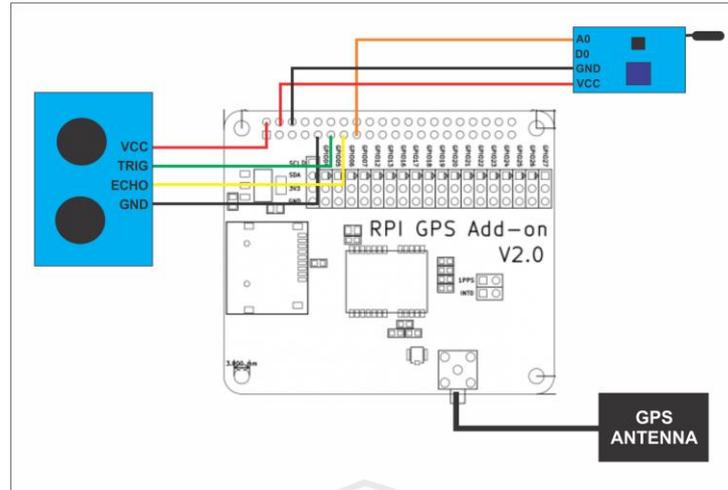
- 1) *Header*, yaitu sebuah kepala dari halaman web yang terletak paling atas. Header berisi logo dan nama dari sistem yang dibangun.
- 2) *Deskripsi*, yaitu berisi penjelasan singkat mengenai sistem yang dibangun. Deskripsi berisi judul dan paragraf singkat penjelasan sistem.
- 3) *Google Maps*, merupakan peta yang digunakan sistem dalam melakukan pemetaan lokasi berdasarkan data yang ditangkap. Pemetaan ini menggunakan *API Google Maps*.
- 4) *Tabel*, yaitu penyajian data kerusakan jalan yang telah ditangkap oleh sensor dan tersimpan dalam server database. Tabel berisi beberapa kolom yang terdiri atas Nomor, Getar, Jarak, Latitude, Longitude, dan Kategori Kerusakan.
- 5) *Icon print*, yaitu sebuah *shortcut* tombol berupa ikon yang berfungsi untuk melakukan cetak laporan.

#### 5.1.5 Perancangan Perangkat Keras

Perancangan perangkat keras bertujuan untuk memberikan gambaran terkait *pin General Input Output (GPIO)* yang digunakan pada mikrokomputer *Raspberry Pi 3*. Pada penelitian ini, *Raspberry Pi* ditanamkan *Raspberry GPS Add-ons* pada seluruh bagian GPIO. Sehingga seluruh *pin* GPIO yang terdapat pada *Raspberry Pi* akan terhubung langsung dengan *pin* GPIO pada *Raspberry GPS*. *Raspberry GPS Add-on* juga menggunakan antenna yang berfungsi untuk menangkap sinyal GPS. *Pin* GPIO pada *Raspberry GPS* akan dihubungkan dengan 2 sensor yaitu sensor ultrasonik dan sensor getar. Perancangan perangkat keras dapat dilihat dalam Gambar 5.7 Rancangan Perangkat Keras, sedangkan pin yang digunakan ditampilkan pada Tabel 5.4 *Pin* GPIO.



Gambar 5.7 Rancangan Antarmuka Halaman Website



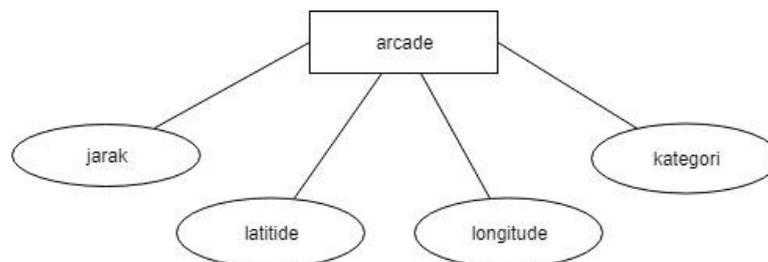
**Gambar 5.8 Rancangan Perangkat Keras**

**Tabel 5.4 Pin GPIO**

No	Sensor	Pin Sensor	Pin Raspberry	Pin GPIO
1.	<i>Ultrasonic</i>	VCC	2	5V PWR
2.	<i>Ultrasonic</i>	GND	9	GND
3.	<i>Ultrasonic</i>	TRIG	11	GPIO 17
4.	<i>Ultrasonic</i>	ECHO	13	GPIO 27
5.	<i>Vibration</i>	VCC	4	5V PWR
6.	<i>Vibration</i>	GND	6	GND
7.	<i>Vibration</i>	A0	15	GPIO 22

### 5.1.6 Perancangan Data

Perancangan data akan digambarkan secara konseptual dengan menggunakan *Entity Realtionship Diagram* (ERD). Perancangan data bertujuan untuk memberikan gambaran terkait data yang akan digunakan dalam sistem. Data ini akan dikodekan ke dalam bentuk data json. Nama entitas dari rancangan data adalah arcade dimana memiliki empat atribut yang meliputi jarak, latitude, longitude dan kategori. Gambaran dari rancangan data dapat dilihat pada Gambar 5.9 Perancangan Data.



**Gambar 5.9 Rancangan Data**

## 5.2 Implementasi

Implementasi sistem pada bab ini akan menjelaskan hasil dari penerapan rancangan sistem yang telah dibuat pada fase sebelumnya. Implementasi sistem menerapkan metode *Object Oriented Programming (OOP)*. Sistem dikembangkan menggunakan dua bahasa pemrograman, yaitu *Python* dan *PHP*. Bahasa *Python* diimplementasikan dalam sistem *embedded Raspberry Pi*. Sedangkan *PHP* diimplementasikan dalam sistem berbasis website.

### 5.2.1 Spesifikasi Sistem

Spesifikasi sistem merupakan deskripsi yang menggambarkan lingkungan dari sistem untuk mengembangkan perangkat lunak. Spesifikasi sistem terdiri atas dua bagian utama, yaitu spesifikasi perangkat keras dan perangkat lunak. Spesifikasi perangkat keras berisi mengenai segala komponen yang digunakan dalam mengembangkan sistem. Sedangkan spesifikasi perangkat lunak berisi mengenai aplikasi yang digunakan selama pengerjaan sistem. Spesifikasi perangkat keras dan perangkat lunak ditampilkan pada Tabel 5.7 dan 5.8.

Tabel 5.5 Spesifikasi perangkat keras

Nama Komponen	Spesifikasi
Prosesor	AMD A8-6410
Memori	500 GB
RAM	4 GB
Mikrokomputer	<i>Raspberry Pi 3</i> model B+
Sensor Getar	SW-18020
Sensor Ultrasonik	HC-SR04
Modul GPS	Raspberry Pi Add-Ons V2.0

Tabel 5.6 Spesifikasi perangkat lunak

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 10 Home 64 bit
Bahasa Pemrograman	PHP dan <i>Python</i>
Framework	Code Igniter
Text Editor	Notepad++
Web Service	XAMPP
Penyimpanan Data Cloud	Realtime Database Firebase
Browser	Google Chrome

### 5.2.2 Implementasi Kode Program

#### 5.2.2.1 Implementasi *method get\_distance()* pada class *Ultrasonic*

*Method get\_distance()* merupakan kode program yang terdapat pada class *ultrasonic*. Method ini berfungsi untuk menangkap data dari sensor ultrasonik. Data yang diperoleh berupa jarak dalam satuan *centimeter*. Kode program *method get\_distance()* ditampilkan pada tabel 5.9. Sedangkan

penjelasan mengenai kode program *method get\_distance()* ditampilkan pada tabel 5.10.

**Tabel 5.7 Kode program method *get\_distance()***

1	<code>def get_distance(self):</code>
2	<code>    GPIO.output(self.TRIG, True)</code>
3	<code>    time.sleep(0.0001)</code>
4	<code>    GPIO.output(self.TRIG, False)</code>
5	<code>    while GPIO.input(self.ECHO) == False:</code>
6	<code>        start = time.time()</code>
7	<code>    while GPIO.input(self.ECHO) == True:</code>
8	<code>        end = time.time()</code>
9	<code>    sig_time = end-start</code>
10	<code>    distance = (sig_time * 17160.5) - 16</code>
11	<code>    distance = round(distance, 4)</code>
12	<code>    return distance</code>

**Tabel 5.8 Penjelasan kode program method *get\_distance()***

1	Deklarasi <i>method get_distance()</i>
2-4	Deklarasi variable TRIG
5-6	Kondisi <i>while</i> dimana GPIO input dengan variabel ECHO bernilai <i>False</i> maka nilai <i>variable start</i> adalah waktu tembak sinyal
7-8	Kondisi <i>while</i> dimana GPIO input dengan variabel ECHO bernilai <i>True</i> maka nilai <i>variable end</i> adalah waktu tembak sinyal
9	Menghitung nilai <i>sig_time = end-start</i>
10	Menghitung nilai <i>distance = (sig_time * 17160.5) - 16</i>
11	Menmbulatkan nilai <i>distance</i> 4 angka dibelakang koma
12	Mengembalikan nilai variabel <i>distance</i>

### 5.2.2.2 Implementasi *method callback()* pada class *Vibration*

*Method callback()* merupakan kode program yang terdapat pada class *vibration*. *Method* ini berfungsi untuk menangkap data dari sensor getar dan mengirimkan data ke *database cloud*. Data yang dikirim akan disimpan dalam bentuk *array* data. Kode program *method callback()* ditampilkan pada tabel 5.11. Sedangkan penjelasan mengenai kode program *method callback()* ditampilkan pada tabel 5.12.

**Tabel 5.9 Kode program method *callback()***

1	<code>def callback(self, channel):</code>
2	<code>    ultra = UltraSonic(17, 27)</code>
3	<code>    cat = Category(ultra)</code>
4	<code>    if GPIO.input(channel):</code>
5	<code>        print "no!"</code>
6	<code>        vib = False</code>
7	<code>    else:</code>
8	<code>        print "Data terbaca!"</code>
9	<code>        vib = True</code>
10	<code>        data = {"jarak":</code> <code>ultra.get_distance(),"latitude": 0, "longitude": 0,</code> <code>"kategori": cat.kategori(ultra.get_distance())}</code>
11	<code>        try:</code>
12	<code>            if not gpssp.isAlive() :</code>
13	<code>                gpssp.start()</code>

**Tabel 5.9 Kode program *method callback()* (lanjutan)**

14	data["latitude"] = gpsd.fix.latitude
15	data["longitude"] = gpsd.fix.longitude
16	except (KeyboardInterrupt, SystemExit):
17	gpsp.running = False
18	gpsp.join()
19	if (gpsd.fix.latitude == 0 or gpsd.fix.longitude == 0 or cat.kategori(ultra.get_distance()) == "Gundukan"):
20	print 'Dont Send Data'
21	else:
22	print 'Send Data'
23	db.child("/arcade").push(data)
24	return vib

**Tabel 5.10 Penjelasan kode program *method callback()***

1	Deklarasi <i>method callback()</i>
2	Deklarasi objek <i>ultra</i> dari <i>class Ultrasonic()</i>
3	Deklarasi objek <i>cat</i> dari <i>class Category()</i>
4-6	Kondisi <i>if</i> dimana GPIO input menerima data dari <i>variable channel</i> maka mengembalikan nilai <i>vib = False</i>
7-9	Kondisi <i>else</i> akan mengembalikan nilai <i>vib = True</i>
10	Deklarasi variabel <i>data</i> yang menerima variabel nilai dari <i>class</i> lain dalam bentuk <i>array</i>
11-15	Kondisi <i>try</i> jika objek <i>gpsp</i> dalam kondisi hidup maka menyimpan data dalam variabel <i>latitude</i> dan <i>longitude</i> ke dalam <i>array data</i>
16-18	Kondisi kecuali jika terjadi gangguan dari <i>keyboard</i> dan menghentikan kondisi <i>try</i>
19-20	Kondisi saat nilai <i>latitude</i> , <i>longitude</i> bernilai 0 atau kategori bernilai "Gundukan" maka data tidak dikirim ke <i>database cloud</i>
21-23	Kondisi <i>else</i> dimana data dikirim ke <i>database cloud</i>
24	Mengembalikan nilai variabel <i>vib</i>

### 5.2.2.3 Implementasi *method kategori()* pada *class Category*

*Method kategori()* merupakan kode program yang terdapat pada *class Category*. *Method* ini berfungsi untuk mengkategorikan tingkat kerusakan jalan berdasarkan data yang diperoleh dari sensor ultrasonik. Data yang diperoleh terdiri dari 4 kategori yaitu gundukan, kerusakan ringan, sedang dan berat. Kode program *method kategori()* ditampilkan pada tabel 5.13. Sedangkan penjelasan mengenai kode program *method kategori()* ditampilkan pada tabel 5.14.

**Tabel 5.11 Kode program *method kategori()***

1	def kategori(self, distance):
2	kategori=""
3	if (distance<0):
4	print "Gundukan"



**Tabel 5.11 Kode program *method kategori()* (lanjutan)**

5	<code>kategori = "Gundukan"</code>
6	<code>elif (distance&gt;1.3 and distance&lt;=2.5):</code>
7	<code>print "Ringan"</code>
8	<code>kategori = "Ringan"</code>
9	<code>elif (distance&gt;2.5 and distance&lt;=5.0):</code>
10	<code>print "Sedang"</code>
11	<code>kategori = "Sedang"</code>
12	<code>elif (distance&gt;5.0):</code>
13	<code>print "Berat"</code>
14	<code>kategori = "Berat"</code>
15	<code>return kategori</code>

**Tabel 5.12 Penjelasan kode program *method kategori()***

1	Deklarasi <i>method kategori()</i>
2	Deklarasi variabel <i>kategori</i> bernilai <i>null</i>
3	Konsisi <i>if</i> jika nilai variabel <i>distance</i> < 0 maka
4	Mencetak teks "Gundukan"
5	Mengembalikan nilai variabel <i>kategori</i> bernilai "Gundukan"
6	Konsisi <i>elif</i> jika nilai variabel <i>distance</i> > 1.3 dan <i>distance</i> <= 2.5 maka
7	Mencetak teks "Ringan"
8	Mengembalikan nilai variabel <i>kategori</i> bernilai "Ringan"
9	Konsisi <i>elif</i> jika nilai variabel <i>distance</i> > 2.5 dan <i>distance</i> <= 5.0 maka
10	Mencetak teks "Sedang"
11	Mengembalikan nilai variabel <i>kategori</i> bernilai "Sedang"
12	Konsisi <i>if</i> jika nilai variabel <i>distance</i> < 0 maka
13	Mencetak teks "Berat"
14	Mengembalikan nilai variabel <i>kategori</i> bernilai "Berat"
15	Mengembalikan nilai variabel <i>kategori</i>

### 5.2.3 Implementasi Antarmuka

Dalam Gambar 5.10 merupakan gambar implementasi antarmuka website. Antarmuka website dibuat berdasarkan perancangan antarmuka pada tahap sebelumnya. Pada *header* terdapat menu sidebar di pojok kanan atas dan logo website di pojok kiri atas. Pada halaman ini, ditampilkan peta digital dengan memanfaatkan *API Google Maps*. Peta digital ini menampilkan lokasi dalam bentuk *icon location*. *Icon location* ini diperoleh dari data tabel yang terdapat di *database firebase*. Pada masing-masing *icon* akan menampilkan kategori kerusakan ketika diklik. Selain itu juga ditampilkan tabel data yang berisi kolom nomor, kedalaman (cm), *latitude*, *longitude* dan kategori. Terdapat tombol untuk menyimpan data ke dalam beberapa format file antara lain Excel, dan PDF. Kolom pencarian berfungsi untuk melakukan pencarian data pada tabel sesuai dengan *keyword* yang dimasukkan. Pada bagian *footer* terdapat media sosial dari dinas terkait.

arcade

HOME / ARCADE

Map Satellite

Search:

Nomor	Kedalaman (cm)	Latitude	Longitude	Kategori
1	4.1528	-7.9546405	112.609282667	Berat
2	4.7706	-7.95464	112.609282667	Sedang
3	5.2329	-7.954640833	112.609281167	Berat
4	3.314	-7.954642667	112.609276167	Sedang
5	3.3468	-7.954667667	112.609288833	Sedang

Showing 1 to 5 of 5 entries

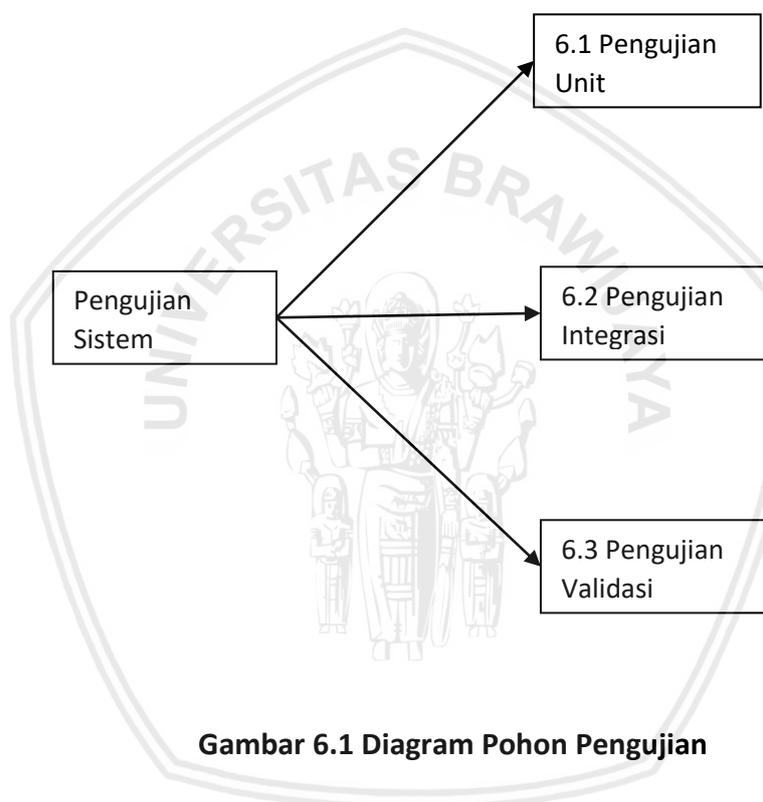
Previous 1 Next

© Untitled. All rights reserved. Design: Dinas Pekerjaan Umum Dan Penataan Ruang Kota Malang

Gambar 5.10 Implementasi Antarmuka Website

## BAB 6 PENGUJIAN SISTEM

Pada bab ini membahas mengenai pengujian sistem yang akan dilakukan dalam penelitian. Pengujian ini didasarkan pada kebutuhan fungsional dan non-fungsional sistem. Hasil dari rekayasa kebutuhaneb pengujian ini akan dijadikan sebagai acuan dalam penarikan kesimpulan dan saran Sistem Pelaporan Lubang Jalan Otomatis Berbasis Sistem Embedded. Pembahasan dibagi menjadi Pengujian unit, pengujian integrasi dan pengujian validasi. Pembahasan dari bab ini akan ditampilkan dalam Gambar 6.1 Diagram pohon pengujian.



Gambar 6.1 Diagram Pohon Pengujian

### 6.1 Pengujian Unit

Pada pengujian unit menjelaskan mengenai rincian pengujian pada setiap komponennya. Pengujian tersebut bertujuan untuk menguji kebenaran komponen dari kode programnya. Metode yang digunakan pada pengujian ini adalah *Whitebox Testing*. Pada penelitian ini terdapat tiga *method* yang akan dibahas. *Method* tersebut antara lain *method get\_distance*, *method callback* dan *method kategori*. Berikut adalah hasil pemaparan perancangan komponen tiga algoritma *method*.

### 6.1.1 Pengujian unit method *get\_distance*

#### 1. Pseudocode

Pengujian unit pada *method get\_distance* dari *class Ultrasonic* akan dijelaskan algoritma yang ditunjukkan pada Tabel 6.1.

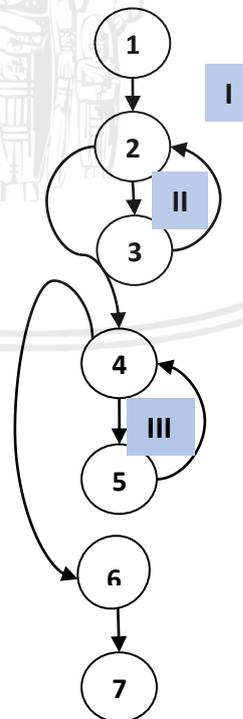
**Tabel 6.1** Pengujian unit *method get\_distance*

Pseudocode <i>get_distance</i>	
1	SET method declaration
2	SET GPIO output TRIG VALUE TRUE
3	SET time.sleep VALUE 0.0001
4	SET GPIO output TRIG VALUE FALSE
5	WHILE GPIO input when ECHO VALUE FALSE: <u>2</u>
6	SET start VALUE this time <u>3</u>
7	WHILE GPIO input when ECHO VALUE TRUE: <u>4</u>
8	SET end VALUE this time <u>5</u>
9	SET sig_time VALUE end MIN start
10	SET distance VALUE sig_time CROSS converter_cm MIN threshold_distance
11	SET distance VALUE distance with round 4
12	RETURN distance <u>7</u>

#### 2. Basis Path Testing

##### 2.1 Flow Graph

Dari algoritma yang dipaparkan akan menjadi dasar pembuatan *flowgraph* untuk *method callback* yang ditunjukkan dalam Gambar 6.2.



**Gambar 6.2** Flow graph *method get\_distance*

### 2.2 Cyclomatic Complexity

Dari gambar *flowgraph* akan diketahui jumlah jalur *independent* melalui kalkulasi *cyclomatic complexity*. Berikut merupakan hasil kalkulasi *cyclomatic complexity* dari *flowgraph method get\_distance*.

$$V(G) = \text{jumlah region} = 3$$

$$V(G) = \text{jumlah edge} - \text{jumlah node} + 2 = 9 - 8 + 2 = 3$$

$$V(G) = \text{jumlah predicate node} + 1 = 2 + 1 = 3$$

### 2.3 Independent Path

Jalur 1: 1-2-4-6-7-8

Jalur 2: 1-2-3-4-6-7-8

Jalur 3: 1-2-3-4-5-6-7-8

Jalur *independent* yang didapat akan dijadikan dasar pembuatan kasus uji. Hasil dari pengujian unit *method get\_distance* ditunjukkan pada Tabel 6.2.

**Tabel 6.2 Hasil pengujian unit *method get\_distance***

Jalur	Kasus uji	Hasil pengujian yang diharapkan	Hasil pengujian	Status
1.	ECHO = FALSE, ECHO = TRUE	Mendapatkan nilai variabel start dan variabel end	Mendapatkan nilai variabel start dan variabel end	Valid
2.	ECHO = TRUE, ECHO = TRUE	Objek berada pada jarak yang jauh dari sensor	Objek berada pada jarak yang jauh dari sensor	Valid
3.	ECHO = TRUE, ECHO = FALSE	Objek berada pada jarak yang jauh dari sensor	Objek berada pada jarak yang jauh dari sensor	Valid

### 6.1.2 Pengujian unit *method callback*

#### 1. Pseudocode

Pengujian unit pada *method callback* dari *class Vibration* akan dijelaskan algoritma yang ditunjukkan pada Tabel 6.3.

**Tabel 6.3 Pengujian unit *method callback***

Pseudocode method callback	
1	SET method declaration
2	GET objek ultra VALUE class UltraSonic } <u>1</u>
3	GET objek cat VALUE class Category } <u>1</u>
4	IF GPIO input GET nothing: <u>2</u>
5	PRINT "no!" } <u>3</u>
6	vib VALUE FALSE } <u>3</u>
7	ELSE:
8	PRINT "Data terbaca!"
9	vib VALUE TRUE
10	SET array data VALUE jarak, latitude, longitude, kategori } <u>4</u>
11	IF gpsp not alive: <u>5</u>
12	START gpsp



**Tabel 6.3 Pengujian unit *method callback* (lanjutan)**

13	SET array data latitude	}	<u>6</u>
14	SET array data longitude		
15	IF latitude VALUE 0 or longitude VALUE 0 or kategori VALUE "Gundukan":		<u>7</u>
16	PRINT "Don't Send Data"		<u>8</u>
17	ELSE:		}
18	SEND array data to database		
19	PRINT "Send Data"		
20	END IF		<u>10</u>
21	ELSE	}	<u>11</u>
22	SET gpsp VALUE FALSE		
23	JOIN gpsp		
24	END IF		<u>12</u>
25	RETURN vib		<u>13</u>
26	END IF		<u>14</u>

2. Basis Path Testing

2.1 Flow Graph

Dari algoritme yang dipaparkan akan menjadi dasar pembuatan *flowgraph* untuk *method callback* yang ditunjukkan dalam Gambar 6.3.

2.1 Cyclomatic Complexity

Dari gambar *flowgraph* akan diketahui jumlah jalur *independent* melalui kalkulasi *cyclomatic complexity*. Berikut merupakan hasil kalkulasi *cyclomatic complexity* dari *flowgraph method callback*.

$$V(G) = \text{jumlah region} = 4$$

$$V(G) = \text{jumlah edge} - \text{jumlah node} + 2 = 16 - 14 + 2 = 4$$

$$V(G) = \text{jumlah predicate node} + 1 = 3 + 1 = 4$$

2.2 Independent Path

Jalur 1: 1-2-3-13-14

Jalur 2: 1-2-4-5-11-12-13-14

Jalur 3: 1-2-4-5-6-7-8-10-12-13-14

Jalur 4: 1-2-4-5-6-7-9-10-12-13-14

Jalur *independent* yang didapat akan dijadikan dasar pembuatan kasus uji. Hasil dari pengujian unit *method callback* ditunjukkan pada Tabel 6.4.

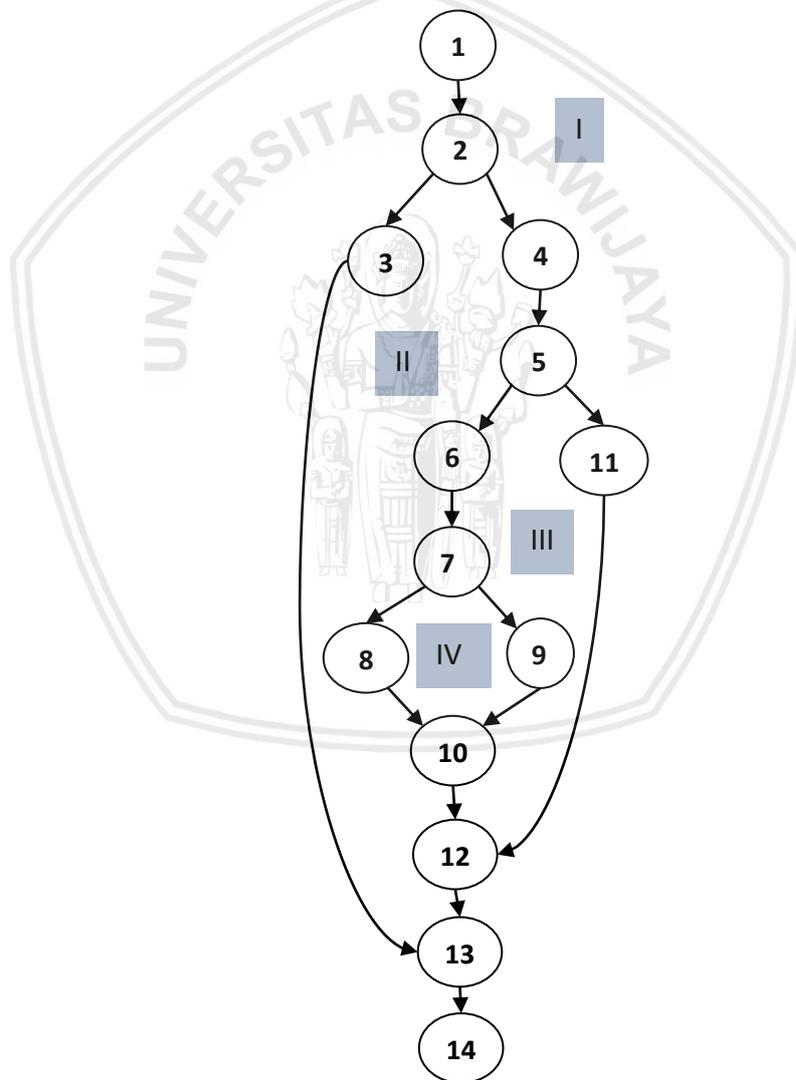
**Tabel 6.4 Hasil pengujian unit *method callback***

Jalur	Kasus uji	Hasil pengujian yang diharapkan	Hasil pengujian	Status
1.	Tidak memberikan getaran pada sensor getar	Mengembalikan variabel vib bernilai <i>False</i>	Mengembalikan variabel vib bernilai <i>False</i>	Valid
2.	Memberikan getaran pada sensor getar dan menekan <i>keyboard</i>	Sistem keluar dan berhenti	Sistem keluar dan berhenti	Valid



**Tabel 6.4 Hasil pengujian unit *method callback* (lanjutan)**

3.	Memberikan getaran pada sensor getar, nilai latitude=0 atau longitude=0 atau kategori = "Gundukam"	Data tidak dikirimkan ke <i>database firebase</i>	Data tidak dikirimkan ke <i>database firebase</i>	Valid
4.	Memberikan getaran pada sensor getar	Data dikirimkan ke <i>database firebase</i>	Data dikirimkan ke <i>database firebase</i>	Valid



**Gambar 6.3 Flow graph *method callback***

### 6.1.3 Pengujian unit *method kategori*

#### 1. Pseudocode

Pengujian unit pada *method kategori* dari class *Category* akan dijelaskan algoritma yang ditunjukkan pada Tabel 6.5.

**Tabel 6.5** Pengujian unit *method kategori*

Pseudocode method kategori	
1	SET method declaration GET VALUE distance
2	SET kategori VALUE NULL <u>1</u>
3	IF distance < 0: <u>2</u>
4	PRINT "Gundukan" }
5	SET kategori VALUE "Gundukan" } <u>3</u>
6	ELSE IF distance > 1.3 AND distance <= 2.5: <u>4</u>
7	PRINT "Ringan" }
8	SET kategori VALUE "Ringan" } <u>5</u>
9	ELSE IF distance > 2.5 AND distace <= 5.0: <u>6</u>
10	PRINT "Sedang" }
11	SET kategori VALUE "Sedang" } <u>7</u>
12	ELSE IF distance > 5.0: }
13	PRINT "Berat" }
14	SET kategori VALUE "Berat" } <u>8</u>
15	END IF <u>9</u>
	RETURN kategori <u>10</u>

#### 2. Basis Path Testing

##### 2.1 Flow Graph

Dari algoritme yang dipaparkan akan menjadi dasar pembuatan *flowgraph* untuk *method kategori* yang ditunjukkan dalam Gambar 6.4.

##### 2.2 Cyclomatic Complexity

Dari gambar *flowgraph* akan diketahui jumlah jalur *independent* melalui kalkulasi *cyclomatic complexity*. Berikut merupakan hasil kalkulasi *cyclomatic complexity* dari *flowgraph method kategori*.

$$V(G) = \text{jumlah region} = 4$$

$$V(G) = \text{jumlah edge} - \text{jumlah node} + 2 = 12 - 10 + 2 = 4$$

$$V(G) = \text{jumlah predicate node} + 1 = 3 + 1 = 4$$

##### 2.3 Independent Path

Jalur 1: 1-2-3-9-10

Jalur 2: 1-2-4-5-9-10

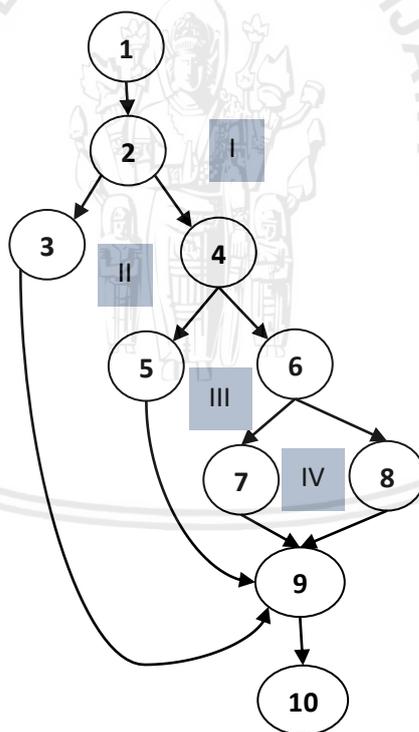
Jalur 3: 1-2-4-6-7-9-10

Jalur 4: 1-2-4-6-8-9-10

Jalur *independent* yang didapat akan dijadikan dasar pembuatan kasus uji. Hasil dari pengujian unit *method kategori* ditunjukkan pada Tabel 6.6.

Tabel 6.6 Hasil pengujian unit *method kategori*

Jalur	Kasus uji	Hasil pengujian yang diharapkan	Hasil pengujian	Status
1.	Memberikan nilai distance -4	Mengembalikan nilai kategori menjadi "Gundukan"	Mengembalikan nilai kategori menjadi "Gundukan"	Valid
2.	Memberikan nilai distance 1.8	Mengembalikan nilai kategori menjadi "Ringan"	Mengembalikan nilai kategori menjadi "Ringan"	Valid
3.	Memberikan nilai distance 3.1	Mengembalikan nilai kategori menjadi "Sedang"	Mengembalikan nilai kategori menjadi "Sedang"	Valid
4.	Memberikan nilai distance 7.3	Mengembalikan nilai kategori menjadi "Berat"	Mengembalikan nilai kategori menjadi "Berat"	Valid



Gambar 6.4 *Flow graph method kategori*

## 6.2 Pengujian Integrasi

Pengujian Integrasi pada penelitian ini bertujuan untuk memastikan bahwa interaksi antar *class* berjalan dengan baik. Pada pengujian Integrasi, digunakan strategi pengujian *top-down* dimana komponen utama akan memanggil

komponen level bawah. Dalam pelaksanaan pengujian integrasi ini memanfaatkan *stub* dimana diperlukan sebuah *class test* yang digunakan untuk menjalankan *method class* yang ingin diuji. Pengujian ini dilakukan untuk menguji interaksi *method callback()* yang ada pada *class Vibration* yang memanggil *method get\_distance()* yang ada pada *Class Ultrasonic*.

### 6.2.1 Pengujian Integrasi *Method callback()* *Class Vibration*

#### 1. Pseudocode

Pengujian integrasi pada *method callback* dari *class Vibration* akan dijelaskan algoritma yang ditunjukkan pada Tabel 6.7.

**Tabel 6.7** Pengujian integrasi *method callback*

Pseudocode method callback	
1	SET method declaration
2	GET objek ultra VALUE class UltraSonic } <u>1</u>
3	GET objek cat VALUE class Category
4	IF GPIO input GET nothing: <u>2</u>
5	PRINT "no!"
6	vib VALUE FALSE } <u>3</u>
7	ELSE:
8	PRINT "Data terbaca!"
9	vib VALUE TRUE } <u>4</u>
10	SET array data VALUE jarak, latitude, longitude, kategori
11	IF gpsp not alive: <u>5</u>
12	START gpsp
13	SET array data latitude
14	SET array data longitude } <u>6</u>
15	IF latitude VALUE 0 or longitude VALUE 0
16	or kategori VALUE "Gundukan": } <u>7</u>
17	PRINT "Don't Send Data" } <u>8</u>
18	ELSE:
19	SEND array data to database } <u>9</u>
	PRINT "Send Data" } <u>10</u>
20	END IF
21	ELSE
22	SET gpsp VALUE FALSE } <u>11</u>
23	JOIN gpsp
24	END IF } <u>12</u>
	RETURN vib } <u>13</u>
	END IF } <u>14</u>

#### 2. Basis Path Testing

##### 2.2 Flow Graph

Dari algoritme yang dipaparkan akan menjadi dasar pembuatan *flowgraph* untuk *method callback* yang ditunjukkan dalam Gambar 6.3.

##### 2.4 Cyclomatic Complexity

Dari gambar *flowgraph* akan diketahui jumlah jalur *independent* melalui kalkulasi *cyclomatic complexity*. Berikut merupakan hasil kalkulasi *cyclomatic complexity* dari *flowgraph method callback*.

$$V(G) = \text{jumlah region} = 4$$

$$V(G) = \text{jumlah edge} - \text{jumlah node} + 2 = 16 - 14 + 2 = 4$$

$$V(G) = \text{jumlah predicate node} + 1 = 3 + 1 = 4$$

2.5 Independent Path

Jalur 1: 1-2-3-4-14

Jalur 2: 1-2-3-5-6-9-10-14

Jalur 3: 1-2-3-5-6-7-8-11-12-14

Jalur 4: 1-2-3-5-6-7-8-11-13-14

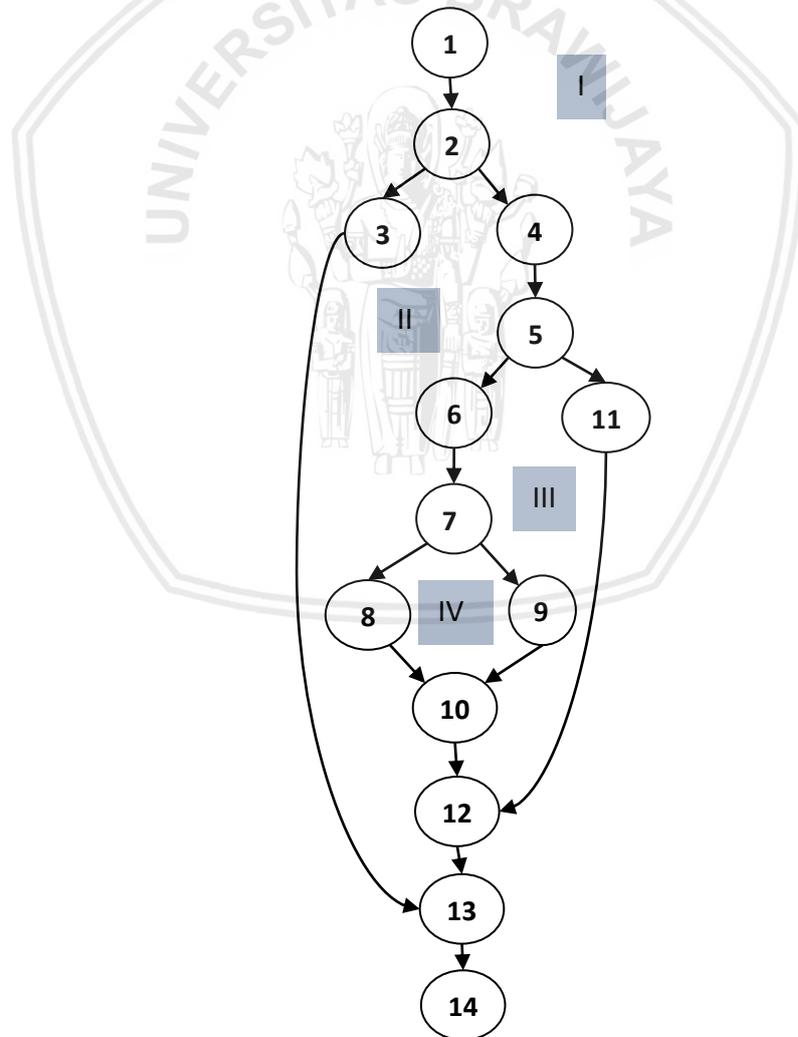
Jalur *independent* yang didapat akan dijadikan dasar pembuatan kasus uji. Hasil dari pengujian unit *method callback* ditunjukkan pada Tabel 6.4.

**Tabel 6.8 Hasil pengujian integrasi *method callback***

Jalur	Kasus uji	Hasil pengujian yang diharapkan	Hasil pengujian	Status
1.	<i>Class Driver</i> memanggil <i>method callbak()</i> yang memanggil <i>method get_distance</i> dari kelas <i>UltraSonic</i> dengan tidak diberi getaran pada sensor getar	Mengembalikan variabel <i>vib</i> bernilai <i>False</i>	Mengembalikan variabel <i>vib</i> bernilai <i>False</i>	Valid
2.	<i>Class Driver</i> memanggil <i>method callbak()</i> yang memanggil <i>method get_distance</i> dari kelas <i>UltraSonic</i> dengan diberi getaran pada sensor getar dan menekan <i>keyboard</i>	Sistem keluar dan berhenti	Sistem keluar dan berhenti	Valid
3.	<i>Class Driver</i> memanggil <i>method callbak()</i> yang memanggil <i>method get_distance</i> dari kelas <i>UltraSonic</i> dengan memberikan getaran pada	Data tidak dikirimkan ke <i>database firebase</i>	Data tidak dikirimkan ke <i>database firebase</i>	Valid



	sensor getar, nilai <i>latitude</i> =0 atau <i>longitude</i> =0 atau kategori = "Gundukan"			
4.	<i>Class Driver</i> memanggil <i>method callbak()</i> yang memanggil <i>method get_distance</i> dari kelas <i>UltraSonic</i> dengan memberikan getaran pada sensor getar	Data dikirimkan ke <i>database firebase</i>	Data dikirimkan ke <i>database firebase</i>	Valid



Gambar 6.5 Flow graph method callback

### 6.3 Pengujian Validasi

Pada pengujian validasi akan dilakukan pengujian pada setiap kebutuhan yang telah didefinisikan sebelumnya. Pengujian ini digunakan untuk memastikan bahwa semua kebutuhan telah diterapkan sesuai skenario yang dibuat. Pengujian validasi akan menguji seluruh kebutuhan baik kebutuhan fungsional dan kebutuhan non-fungsional. Pengujian validasi menggunakan metode *Blackbox testing*. Pada penelitian ini akan menguji 8 kebutuhan fungsional beserta alur alternatif dan satu kebutuhan non-fungsional. Pengujian validasi pada penelitian ini dijelaskan pada sub bab 6.3.1 sampai 6.3.10

#### 6.3.1 Pengujian validasi menangkap data getar

Pengujian validasi menangkap data getar merupakan aktivitas yang dilakukan oleh aktor berupa sensor getar. Pengujian validasi menangkap data getar akan menguji kebutuhan fungsional dengan kode PKJO-KF-001. Pengujian ini terdiri atas satu jalur utama dan satu jalur alternatif. Sensor getar ini berfungsi untuk mendeteksi getaran yang ada pada suatu objek. Sistem akan menerima data getaran yang ditangkap oleh sensor dalam boolean *True* atau *False*. Pengujian validasi menangkap data getar ditampilkan pada Tabel 6.9 dan menangkap data getar jalur alternatif ditampilkan pada Tabel 6.10

**Tabel 6.9 Pengujian validasi menangkap data getar**

Nama kasus uji	Menangkap data getar
Objek uji	Kebutuhan Fungsional PKJO-KF-001
Tujuan pengujian	Memastikan bahwa sensor getar dapat menangkap data berupa getaran.
Data masukan	Sensor menerima trigger berupa getaran
Prosedur uji	<ol style="list-style-type: none"> <li>1. Sensor getar dijalankan pada kendaraan dan melewati jalan</li> <li>2. Sensor getar mendeteksi getaran pada jalan dan membaca besar getaran</li> <li>3. Sistem menerima data sensor berupa angka satuan boolean <i>True</i></li> </ol>
Hasil pengujian yang diharapkan	Sistem menerima data sensor berupa boolean <i>True</i>
Hasil pengujian	Sistem menerima data sensor berupa boolean <i>True</i>
Status	Valid

**Tabel 6.10 Pengujian validasi menangkap data getar jalur alternatif**

Nama kasus uji	Menangkap data getar
Objek uji	Kebutuhan Fungsional PKJO-KF-001
Tujuan pengujian	Memastikan bahwa sensor getar tidak dapat menangkap data berupa getaran.
Data masukan	Sensor tidak menerima trigger berupa getaran

**Tabel 6.10 Pengujian validasi menangkap data getar jalur alternatif (lanjutan)**

Prosedir uji	<ol style="list-style-type: none"> <li>1. Sensor getar dijalankan pada kendaraan dan melewati jalan</li> <li>2. Sensor getar tidak mendeteksi getaran pada jalan dan membaca besar getaran</li> <li>3. Sistem menerima data sensor berupa angka satuan boolean <i>Flase</i></li> </ol>
Hasil pengujian yang diharapkan	Sistem menerima data sensor berupa boolean False
Hasil pengujian	Sistem menerima data sensor berupa boolean Flase
Status	Valid

### 6.3.2 Pengujian Validasi Menangkap data jarak

Pengujian validasi menangkap data jarak merupakan aktivitas yang dilakukan oleh aktor berupa sensor ultrasonik. Pengujian validasi menangkap data getar akan menguji kebutuhan fungsional dengan kode PKJO-KF-002. Pengujian ini terdiri atas satu jalur utama dan satu jalur alternatif. Sensor ultrasonik ini berfungsi untuk mengukur kedalaman dari kerusakan yang ada pada jalan raya. Sistem akan menerima angka data besaran jarak yang ditangkap oleh sensor dalam satuan centimeter (cm). Pengujian validasi menangkap data jarak ditampilkan pada Tabel 6.11 dan menangkap data jarak jalur alternatif ditampilkan pada Tabel 6.12

**Tabel 6.11 Pengujian validasi menangkap data jarak**

Nama kasus uji	Menangkap data jarak
Objek uji	Kebutuhan Fungsional PKJO-KF-002
Tujuan pengujian	Memastikan bahwa sensor mampu menerima data berupa jarak
Data masukan	Data kedalaman yang ditangkap oleh sensor dalam satuan centimeter
Prosedur uji	<ol style="list-style-type: none"> <li>1. Sensor getar mendeteksi getaran pada jalan</li> <li>2. Sensor ultrasonik melakukan pengukuran kedalaman kerusakan jalan</li> <li>3. Sensor ultrasonik membaca besar jarak kedalaman kerusakan jalan</li> <li>4. Sistem menerima data sensor berupa angka satuan centimeter (cm)</li> </ol>
Hasil pengujian yang diharapkan	Sistem menerima data sensor berupa angka satuan centimeter (cm)
Hasil pengujian	Sistem menerima data sensor berupa angka satuan centimeter (cm)
Status	Valid

**Tabel 6.12 Pengujian validasi menangkap data jarak jalur alternatif**

Nama kasus uji	Menangkap data jarak
Objek uji	Kebutuhan Fungsional PKJO-KF-002
Tujuan pengujian	Memastikan bahwa sensor tidak menerima data berupa jarak
Data masukan	Data kedalaman yang ditangkap oleh sensor dalam satuan centimeter
Prosedur uji	<ol style="list-style-type: none"> <li>1. Sensor getar tidak mendeteksi getaran pada jalan</li> <li>2. Sensor ultrasonik tidak melakukan pengukuran kedalaman kerusakan jalan</li> <li>3. Sensor ultrasonik tidak membaca besar jarak kedalaman kerusakan jalan</li> <li>4. Sistem tidak menerima data sensor berupa angka satuan centimeter (cm)</li> </ol>
Hasil pengujian yang diharapkan	Sistem tidak menerima data sensor berupa angka satuan centimeter (cm)
Hasil pengujian	Sistem tidak menerima data sensor berupa angka satuan centimeter (cm)
Status	Valid

### 6.3.3 Pengujian Validasi Menangkap data koordinat

Pengujian validasi menangkap data koordinat merupakan aktivitas yang dilakukan oleh aktor berupa sensor GPS. Pengujian validasi menangkap data koordinat akan menguji kebutuhan fungsional dengan kode PKJO-KF-003. Pengujian ini terdiri atas satu jalur utama dan satu jalur alternatif. Sensor GPS ini berfungsi untuk mendapatkan koordinat lokasi dari kerusakan jalan yang terdeteksi. Sistem akan menerima data koordinat lokasi dalam bentuk data latitude dan longitude. Pengujian validasi menangkap data koordinat ditampilkan pada Tabel 6.13 dan menangkap data koordinat jalur alternatif ditampilkan pada Tabel 6.14.

**Tabel 6.13 Pengujian validasi menangkap data koordinat**

Nama kasus uji	Menangkap data koordinat
Objek uji	Kebutuhan Fungsional PKJO-KF-003
Tujuan pengujian	Memastikan bahwa sensor mampu menerima data berupa koordinat lokasi
Data masukan	Data koordinat lokasi berupa latitude dan longitude
Prosedur uji	<ol style="list-style-type: none"> <li>1. Sensor getar mendeteksi getaran pada jalan</li> <li>2. Sensor GPS melakukan pencarian koordinat lokasi dari kerusakan jalan yang terdeteksi oleh sensor getar</li> <li>3. Sensor GPS membaca koordinat lokasi dari kerusakan jalan</li> <li>4. Sistem menerima data sensor berupa koordinat lokasi dalam bentuk latitude dan longitude</li> </ol>

**Tabel 6.13 Pengujian validasi menangkap data koordinat (lanjutan)**

Hasil pengujian yang diharapkan	Sistem menerima data sensor berupa koordinat lokasi dalam bentuk latitude dan longitude
Hasil pengujian	Sistem menerima data sensor berupa koordinat lokasi dalam bentuk latitude dan longitude
Status	Valid

**Tabel 6.14 Pengujian validasi menangkap data koordinat jalur alternatif**

Nama kasus uji	Menangkap data koordinat
Objek uji	Kebutuhan Fungsional PKJO-KF-003
Tujuan pengujian	Memastikan bahwa sensor tidak menerima data berupa koordinat lokasi
Data masukan	Tidak terdeteksi getaran
Prosedur uji	<ol style="list-style-type: none"> <li>1. Sensor getar tidak mendeteksi getaran pada jalan</li> <li>2. Sensor GPS tidak melakukan pencarian koordinat lokasi dari kerusakan jalan yang terdeteksi oleh sensor getar</li> <li>3. Sensor GPS tidak membaca koordinat lokasi dari kerusakan jalan</li> <li>4. Sistem tidak menerima data sensor berupa koordinat lokasi dalam bentuk latitude dan longitude</li> </ol>
Hasil pengujian yang diharapkan	Sistem tidak menerima data sensor berupa koordinat lokasi dalam bentuk latitude dan longitude
Hasil pengujian	Sistem tidak menerima data sensor berupa koordinat lokasi dalam bentuk latitude dan longitude
Status	Valid

### 6.3.4 Pengujian Validasi Mengirim data

Setelah sensor getar menangkap *trigger* dalam bentuk getaran, maka akan mengembalikan nilai boolean *True*. Ketika boolean bernilai *True* maka akan mengirimkan data lokasi kerusakan pada jalan raya ke *server cloud*. Pengujian validasi menangkap data getar akan menguji kebutuhan fungsional dengan kode PKJO-KF-004. Pengujian ini terdiri atas satu jalur utama dan satu jalur alternatif. Data yang dikirimkan dalam bentuk *array* yang berisi jarak, *latitude*, *longitude* dan kategori kerusakan yang nantinya akan disimpan ke dalam server. Data dari *server* kemudian akan diakses oleh admin. Pengujian validasi mengirim data ditampilkan pada Tabel 6.15 dan mengirim data jalur alternatif ditampilkan pada Tabel 6.16.

**Tabel 6.15 Pengujian validasi mengirim data**

Nama kasus uji	Mengirim data
Objek uji	Kebutuhan fungsional PKJO-KF-004
Tujuan pengujian	Memastikan bahwa data yang diperoleh sensor telah dikirimkan ke <i>server cloud</i>
Data masukan	Data berupa <i>array</i> yang memuat jarak, <i>latitude</i> , <i>longitude</i> dan kategori

**Tabel 6.15 Pengujian validasi mengirim data (lanjutan)**

Prosedur uji	<ol style="list-style-type: none"> <li>1. Sensor menerima <i>trigger</i> berupa getaran</li> <li>2. Sistem mengembalikan nilai boolean <i>True</i></li> <li>3. Sistem mengirimkan data yang telah diperoleh oleh sensor ke dalam bentuk data array</li> <li>4. Data tersimpan di server cloud ke dalam bentuk tabel</li> </ol>
Hasil pengujian yang diharapkan	Data tersimpan di <i>server cloud</i>
Hasil pengujian	Data tersimpan di <i>server cloud</i>
Status	Valid

**Tabel 6.16 Pengujian validasi mengirim data jalur alternatif**

Nama kasus uji	Mengirim data
Objek uji	Kebutuhan fungsional PKJO-KF-004
Tujuan pengujian	Memastikan bahwa data yang diperoleh sensor tidak dikirimkan ke <i>server cloud</i>
Data masukan	Tidak terdeteksi getaran
Prosedur uji	<ol style="list-style-type: none"> <li>1. Sensor tidak menerima <i>trigger</i> berupa getaran</li> <li>2. Sistem mengembalikan nilai boolean <i>False</i></li> <li>3. Sistem tidak mengirimkan data yang telah diperoleh oleh sensor ke dalam bentuk data <i>array</i></li> <li>4. Data tidak tersimpan di <i>server cloud</i> ke dalam bentuk tabel</li> </ol>
Hasil pengujian yang diharapkan	Data tidak dikirimkan ke <i>server cloud</i>
Hasil pengujian	Data tidak dikirimkan ke <i>server cloud</i>
Status	Valid

### 6.3.5 Pengujian Validasi Klasifikasi kerusakan

Pengujian validasi klasifikasi kerusakan merupakan aktivitas yang dilakukan oleh aktor berupa sensor ultrasonik. Pengujian validasi klasifikasi kerusakan akan menguji kebutuhan fungsional dengan kode PKJO-KF-005. Pengujian ini terdiri atas satu jalur utama dan satu jalur alternatif. Sistem juga dapat melakukan klasifikasi kerusakan jalan berdasarkan data yang diperoleh oleh sensor ultrasonik. Klasifikasi kerusakan jalan mengacu pada kategori yang terdapat dalam literatur. Pengujian validasi klasifikasi kerusakan ditampilkan pada Tabel 6.17.

**Tabel 6.17 Pengujian validasi klasifikasi kerusakan**

Nama kasus uji	Klasifikasi kerusakan
Objek uji	Kebutuhan Fungsional PKJO-KF-005
Tujuan pengujian	Memastikan bahwa sistem mampu melakukan pengklasifikasian terkait kerusakan jalan
Data masukan	Data berupa nilai jarak yang diperoleh oleh sensor ultrasonik yang kemudian akan diproses berdasarkan kategori kerusakan jalan

**Tabel 6.17 Pengujian validasi klasifikasi kerusakan**

Prosedur uji	<ol style="list-style-type: none"> <li>1. Sistem menerima data dari sensor ultrasonik</li> <li>2. Sistem melakukan penyeleksian kategori kerusakan jalan</li> <li>3. Sistem memberikan label terkait jenis dari kerusakan dan disimpan kedalam kolom klasifikasi pada tabel</li> <li>4. Sistem menampilkan hasil klasifikasi di halaman website admin</li> </ol>
Hasil pengujian yang diharapkan	Sistem mengembalikan nilai dari kategori kerusakan
Hasil pengujian	Sistem mengembalikan nilai dari kategori kerusakan
Status	Valid

### 6.3.6 Pengujian Validasi Menampilkan data

Pengujian validasi menampilkan data merupakan aktivitas yang dilakukan oleh aktor berupa admin. Pengujian validasi menampilkan data akan menguji kebutuhan fungsional dengan kode PKJO-KF-006. Pengujian ini terdiri atas satu jalur utama. Setelah semua data dari sensor tersimpan dalam *server cloud*, maka data tersebut akan ditampilkan di halaman website yang bisa diakses oleh admin. Pada halaman ini, sistem akan menampilkan data dari *server* ke dalam bentuk tabel. Pengujian validasi menampilkan data ditampilkan pada tabel 6.18.

**Tabel 6.18 Pengujian validasi menampilkan data**

Nama kasus uji	Menampilkan data
Objek uji	Kebutuhan Fungsional PKJO-KF-006
Tujuan pengujian	Memastikan bahwa sistem mampu menampilkan data yang tersimpan dalam <i>server cloud</i> ke halaman website admin
Data masukan	Data yang telah tersimpan di <i>database cloud</i> akan ditampilkan ke dalam bentuk tabel
Prosedur uji	<ol style="list-style-type: none"> <li>1. Admin membuka halaman dashboard</li> <li>2. Sistem menampilkan data dalam bentuk tabel beserta peta lokasi kerusakan jalan</li> </ol>
Hasil pengujian yang diharapkan	Admin dapat melihat data yang tersimpan di <i>server cloud</i> di halaman website
Hasil pengujian	Admin dapat melihat data yang tersimpan di <i>server cloud</i> di halaman website
Status	Valid

### 6.3.7 Pengujian Validasi Mencetak laporan

Pengujian validasi mencetak laporan akan merupakan aktivitas yang dilakukan oleh aktor berupa admin. Pengujian validasi mencetak laporan akan menguji kebutuhan fungsional dengan kode PKJO-KF-007. Pengujian ini terdiri atas satu jalur utama dan satu jalur alternatif. Selain melakukan peninjauan secara langsung, admin juga dapat melakukan mencetak laporan dari data yang

ditampilkan pada tabel kerusakan jalan. Hal ini bertujuan untuk memudahkan admin dalam merekap data kerusakan jalan. Pengujian validasi mencetak laporan ditampilkan pada tabel 6.19 dan mencetak laporan jalur alternatif ditampilkan pada tabel 6.20.

**Tabel 6.19 Pengujian validasi mencetak laporan**

Nama kasus uji	Mencetak laporan
Objek uji	Kebutuhan Fungsional
Tujuan pengujian	Memastikan bahwa sistem dapat melakukan perekapan data dengan mengunduh data yang tersimpan pada <i>database cloud</i>
Data masukan	Data dari kerusakan jalan telah tersimpan ke dalam <i>database cloud</i>
Prosedur uji	<ol style="list-style-type: none"> <li>1. Admin membuka halaman website</li> <li>2. Admin menekan tombol cetak laporan PDF atau EXEL</li> <li>3. Admin memilih direktori lokasi untuk menyimpan file data</li> <li>4. Admin menekan tombol OK</li> <li>5. Sistem menampilkan jendela <i>browse</i> untuk menyimpan file</li> <li>6. Sistem mengunduh dan menyimpan file data</li> </ol>
Hasil pengujian yang diharapkan	Admin berhasil mengunduh dan menyimpan file data
Hasil pengujian	Admin berhasil mengunduh dan menyimpan file data
Status	Valid

**Tabel 6.20 Pengujian validasi mencetak laporan**

Nama kasus uji	Mencetak laporan
Objek uji	Kebutuhan Fungsional
Tujuan pengujian	Memastikan bahwa sistem dapat melakukan perekapan data dengan mengunduh data yang tersimpan pada <i>database cloud</i>
Data masukan	Data dari kerusakan jalan telah tersimpan ke dalam <i>database cloud</i>
Prosedur uji	<ol style="list-style-type: none"> <li>1. Admin membuka halaman website</li> <li>2. Admin menekan tombol cetak laporan PDF atau EXEL</li> <li>3. Admin memilih direktori lokasi untuk menyimpan file data</li> <li>4. Admin menekan tombol <i>Cancel</i></li> <li>5. Sistem menampilkan jendela <i>browse</i> untuk menyimpan file</li> <li>6. Sistem mengunduh dan menyimpan file data</li> </ol>
Hasil pengujian yang diharapkan	Admin tidak mengunduh dan menyimpan file data
Hasil pengujian	Admin tidak mengunduh dan menyimpan file data
Status	Valid

### 6.3.8 Pengujian Validasi Menampilkan lokasi

Pengujian validasi menampilkan lokasi merupakan aktivitas yang dilakukan oleh aktor berupa admin. Pengujian validasi menampilkan lokasi akan menguji kebutuhan fungsional dengan kode PKJO-KF-008. Pengujian ini terdiri atas satu jalur utama. Untuk memudahkan peninjauan lokasi yang dilakukan oleh admin secara *online*, juga terdapat peta digital. Peta digital ini memanfaatkan *API Google Maps* untuk menunjukkan titik koordinat lokasi kerusakan jalan. Titik koordinat lokasi ini mengacu pada data yang diperoleh oleh sensor GPS. Pengujian validasi menampilkan lokasi ditampilkan pada Tabel 6.21.

**Tabel 6.21 Pengujian validasi menampilkan lokasi**

Nama kasus uji	Menampilkan lokasi
Objek uji	Kebutuhan Fungsional PKJO-KF-008
Tujuan pengujian	Menampilkan titik kerusakan jalan dengan bantuan peta digital
Data masukan	Mengakses data yang telah disimpan didalam <i>database</i> dan memetakannya ke dalam peta digital
Prosedur uji	<ol style="list-style-type: none"> <li>1. Admin membuka halaman website</li> <li>2. Sistem mengambil data dari server</li> <li>3. Sistem menampilkan lokasi berdasarkan data yang diperoleh dari server dalam bentuk titik lokasi</li> </ol>
Hasil yang diharapkan	Admin dapat melihat titik lokasi sesuai dengan data yang dipilih
Hasil pengujian	Admin dapat melihat titik lokasi sesuai dengan data yang dipilih
Status	Valid

### 6.3.9 Pengujian validasi kebenaran data (*correctness*)

Pengujian validasi kebenaran data (*correctness*) merupakan suatu pengujian validasi yang bertujuan untuk menguji kebenaran data yang diperoleh oleh sistem. Data yang diuji meliputi data jarak dari sensor ultrasonik, lokasi *latitude* dan *longitude* serta klasifikasi kerusakan jalan. Pengujian ini akan membandingkan antara hasil yang diperoleh oleh sistem dengan penghitungan manual yang ada di lapangan dengan menggunakan alat bantu ukur penggarin dan aplikasi *Google Maps*. Pengujian validasi kebenaran data ditampilkan pada Tabel 6.22.

**Tabel 6.22 Pengujian validasi kebenaran data**

Nama kasus uji	Kebenaran data
Objek uji	Kebutuhan Non-fungsional PKJO-KNF-001
Tujuan pengujian	Menguji kebenaran data yang diterima oleh sistem
Data masukan	Menjalankan sistem pada jalan yang berlubang dan melakukan penghitungan manual terhadap jalan yang berlubang tersebut terkait kedalaman dan lokasinya

**Tabel 6.22 Pengujian validasi kebenaran data (lanjutan)**

Prosedur uji	<ol style="list-style-type: none"><li>1. Melakukan pengukuran terhadap lubang jalan terkait kedalamannya dan lokasi yang tertera di dalam aplikasi <i>Google Maps</i>.</li><li>2. Menjalankan sistem dan di lewatkan pada jalan berlubang tersebut</li><li>3. Mengumpulkan semua data yang telah di hitung secara manual amupun di reap oleh sistem</li></ol>
Hasil yang diharapkan	Sistem dapat memperoleh data secara benar sesuai dengan kondisi lingkungan
Hasil pengujian	Sistem dapat memperoleh data secara benar sesuai dengan kondisi lingkungan
Status	Valid



## BAB 7 PENUTUP

### 7.1 Kesimpulan

Berdasarkan hasil analisis kebutuhan, perancangan, implementasi, dan pengujian yang telah dilakukan, maka dapat disimpulkan bahwa pada tahap rekayasa kebutuhan sistem, diperoleh delapan kebutuhan fungsional dan satu kebutuhan non-fungsional yang dapat membantu proses pelaporan dan *monitoring* kerusakan jalan dengan menggunakan peta digital. Kebutuhan fungsional yang didefinisikan yaitu mengumpulkan data kerusakan jalan dari sensor, mengirimkan data ke database cloud, mengklasifikasikan jenis kerusakan jalan dan menampilkan data dari cloud ke dalam *API Google Maps*. Kebutuhan fungsional yang telah didefinisikan, dimodelkan dalam bentuk *use case diagram* dan *use case scenario*. Pada tahap perancangan yang telah dilakukan diperoleh hasil dari perancangan berupa rancangan *sequence diagram* dan *class diagram*. Perancangan algoritme yang akan digunakan di dalam sistem. Dalam perancangan antarmuka diperoleh rancangan *layout* antarmuka sistem. Dalam perancangan perangkat keras diperoleh rancangan *pin* yang digunakan pada mikrokontroler sistem. Pada tahap implementasi dihasilkan spesifikasi sistem, implementasi kode program menurut rancangan komponen, dan implementasi antarmuka menurut rancangan antarmuka. Pada tahap pengujian yang telah dilakukan diperoleh bahwa pada pengujian unit, pengujian integrasi dan pengujian validasi menghasilkan nilai 100% valid.

### 7.2 Saran

Saran yang diberikan untuk pengembangan lanjut sistem pelaporan kerusakan jalan otomatis berbasis sistem embedded antara lain diperlukan sebuah algoritma yang mampu membantu petugas dinas dalam menentukan *clusterisasi* wilayah kerusakan jalan dan prioritas perbaikan jalan berdasarkan parameter tertentu. Selain itu sistem belum mampu mendeteksi kerusakan jalan dengan kondisi ringan seperti keretakan dan pengausan ruas jalan.

## DAFTAR PUSTAKA

- Adhithia, R., Susetiyo, F., Triyanto, D. and Komputer, J.S., 2016. Rancang Bangun *Smart Vehicle*. 4(3).
- Angga, Muamar. 2017. Apa Itu Raspberry Pi? Dan Apa Perbedaan Versi Lama Dengan Yang Baru. <https://news.ralali.com/apa-itu-raspberry-pi/>. diakses pada 6 Juni 2018.
- Aris, Budi. 2017. 159 Kasus Kecelakaan di Jateng Disebabkan Jalan Berlubang. <https://www.radioidola.com/2017/159-kasus-kecelakaan-di-jateng-disebabkan-jalan-berlubang/>. diakses pada 6 Juni 2018.
- Bernad, R. and Syafaruddin, A.S., 2017. ANALISA KONDISI KERUSAKAN JALAN RAYA PADA LAPISAN PERMUKAAN (Studi Kasus: Jalan Raya Desa Kapur, Desa Kapur, Kecamatan Sungai Raya, Kabupaten Kubu Raya, Provinsi Kalimantan Barat). pp.1–10.
- Daqiqil, Ibnu. 2011. Framework Codeigneter. Online (<https://www.amarbank.co.id/upload/ed69f40abc4d5bb8a9b4857cf461f142.pdf>). Diakses pada 06 Desember 2018.
- Enterprise, Jubilee.2017. otodidak pemrograman phyton. BUKU.
- Gnanapriya, S., 2015. *IOT Based Pothole Detection and Notification System*. 12(3), pp.172–179.
- Lambert, Tyler Ross. 2018. Introduction to Microcontrollers and Embedded Systems. Amerika Serikat: Auburn University
- M, R.G., S, S., KUMAR.H, K., S, B. and R, G., 2016. *Automatic Detection of Potholes on Roads to Aid Drivers. Proceedings of 41st IRF International Conference*, 15(8), pp.67–72.
- M, RAMYA GAYATHRI, SREELAKSHMI S, KRISHNA KUMAR.H, BHARATHI. S, and GEETHA. R. 2016. "Automatic Detection of Potholes on Roads to Aid Drivers." *Proceedings of 41st IRF International Conference* 15 (8): 67–72.
- Madli, R., Hebbar, S., Pattar, P. and Golla, V., 2015. *Automatic Detection and Notification of Potholes and Humps on Roads to Aid Drivers. IEEE Sensors Journal*, 15(8), pp.4313–4318.
- Morgan, Elijah J. 2014. HC-SR04 Ultrasonic Sensor. Online ([http://centmesh.csc.ncsu.edu/ff\\_drone\\_f14\\_finals/Sensor1/files/hcsr04.pdf](http://centmesh.csc.ncsu.edu/ff_drone_f14_finals/Sensor1/files/hcsr04.pdf)). Diakses Pada 06 Desember 2018.
- Nicolas, Baudoux et Bauwin Lucie. 2017. Real-Time Database : Firebase INFO-H-415 : Advanced Database. Belgia: Univeriste Libre de Bruxelles.
- Pressman, Roger S. 2010. *Software Engineering: A Practitioner's Approach*. Seventh Edition. New York: The McGraw-Hill Companies, Inc.,
- S, Jaya Kumar, and Vignesh Muralidoss. 2018. "Survey on Road Surface Monitoring System Using" 6 (Iii): 2254–58.
- Shahin, M. Y. 1994. *Pavement management for airports, roads, and parking lots* (Vol. 501). New York: Springer.
- Solichin, Achmad. 2016. Pemrograman Web dengan PHP dan MySQL. Online (<https://books.google.co.id/books?id=k8-GDAAAQBAJ&printsec=frontcover&dq=bahasa+pemrograman+php+pdf&hl=id>

- &sa=X&ved=0ahUKEwi9ipWbwovfAhUIMo8KHSzBD0kQ6AEIKTAA#v=onepage&q=php&f=false). Diakses pada 06 Desember 2018.
- Sommerville, Ian. 2011. *Software Engineering*. Ninth Edition. Boston: Pearson Education, Inc.,
- Spichkova, Maria. 2013. *Seminar: Embedded Systems*. Melbourne: RMIT University.
- Sukirman, Silvia. 1999. *Perkerasan Lentur Jalan Raya*. Bandung: Nova.
- Tokoteknologi. 2018. Modul SW-420 Motion Sensor Vibration Switch Alarm Getaran. <http://www.tokoteknologi.co.id/modul-sw-420-motion-sensor-vibration-switch-alarm-getaran>. diakses pada 22 Agustus 2018.
- Umum, P. and Perumahan, D.A.N., 2019. Rencana strategis (renstra) kementerian PUPR 2015 – 2019.
- Utama, Y., 2013. Sistem Pendukung Keputusan Untuk Menentukan Prioritas Penanganan Perbaikan Jalan Menggunakan Metode Saw Berbasis Mobile Web. *Jurnal Sistem Informasi (JSI)*, 5(1), pp.566–579.

