IOT FOR BEGINNER

Penyusun

Koordinator:
Ni Komang Widyasanti

Tim Program:
Gede Widya Dharma (Koordinator)
I Putu Arya Putra Wibawa
Muhammad Raihan Obbiansyah Amri
Lydia Emeralda S
Viona Dewi Ayunitami
I GBN Satya Wibawa
Sulya Arya Wasika
Pande Bagus Narendra Mahaputra
Gede Jorghi Saputra
I Gede Wahyu Pratama
Agus Ghana Putra P Y
Altry David Purba
Putu Irvan Arya Purwadana
Ni Ketut Pradani G
I Komang Liska Bagiartha
I Putu Yogi Dana Saputra

Tim Webservice:
I GBN Satya Wibawa (Koordinator)
Kadek Meliantari

Tim Mobile:
I GD N Bayu Darmawan (Koordinator)
Putri Isma Oktawiani
I Gede Lanang Ari Praditha
Joysun Agape Sianturi
Veranita L Sihombing

Tim Laporan:
Ni Made Surasmitha Dewi (Koordinator 1)
Komang Devi Tripika Dewi (Koordinator 2)
Ni Putu Tias Amarwati

Chatarina Indah Kristina Dewi
Ida Ayu Rini Dharmayani
Tania Maria O S

Cover, Ilustrasi & Gambar:
Ni Komang Widyasanti
Muhammad Raihan Obbiansyah Amri

Disain & Layout:
Ni Komang Widyasanti

IOT FOR BEGINNER | SMART CAR

KATA PENGANTAR

Saat ini banyak yang membicarakan tentang *Internet of Things* dalam berbagai bidang. *Internet of Things* atau dikenal juga dengan singkatan IoT merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas Internet yang tersambung secara terus-menerus.

Penulis sangat tertarik untuk mempelajari lebih dalam mengenai *Internet of Things.* Penulis menerapkan konsep *Internet of Things* dengan memanfaatkan sensor-sensor yang ada dan merancang sebuah aplikasi yaitu Aplikasi SmartCar. Aplikasi ini nantinya dapat terhubung ke Internet dan dapat menjalankan fitur-fitur yang tersedia pada aplikasi.

Buku ini disusun untuk menyajikan teori dan konsep tentang *Internet of Things* agar pembaca dapat memperluas ilmu mengenai *Internet of Things*, serta memahami tentang aplikasi SmartCar. Penulis mengucapkan terima kasih pada semua pihak yang telah membantu dalam menyelesaikan buku SmartCar. Penulis menyadari masih terdapat kekurangan dalam penyusunan buku ini, maka kritik dan saran sangat diperlukan untuk melakukan penyempurnaan.

Denpasar, Juni 2017

Penulis

DAFTAR ISI

DAFTAR KODE PROGRAM

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

Internet of Things

Pendahuluan
Pengenalan Internet of Things
Sejarah Internet of Things
Layer pada Internet of Things
Integrasi Layer pada Fitur SmartCar
Teknologi Internet of Things
Penerapan Internet of Things
Sistem operasi internet of things

1.1 Pendahuluan

Saat ini dunia telah diperkenalkan dengan teknologi maupun perangkat *Internet of Things* yang telah banyak dikembangkan oleh berbagai *vendor*. *Internet of Things* merupakan teknologi yang membuat suatu benda dapat terhubung dengan benda lainnya melalui jaringan Internet. Penghubung utama dalam interaksi benda adalah Internet dan manusia hanya berperan sebagai pengatur serta pemantau perangkat *Internet of Things* secara langsung.

Internet of Things digunakan tidak hanya sebatas untuk perangkat rumah tangga saja, tetapi untuk berbagai keperluan seperti yang berhubungan dengan lingkungan, pangan, penelitan, kesehatan, tata kota, pekerjaan, dan masih banyak lagi. Suatu benda dapat ditanamkan sensor dan dibuat selalu aktif sehingga terhubung secara luas, baik itu menggunakan Internet dengan jaringan lokal maupun global, sehingga menjadi perangkat *Internet of Things* yang lebih cerdas dan memudahkan kehidupan orang banyak. Hal ini terlihat dengan telah disediakannya berbagai macam program untuk membantu pengembangan produk berbasis *Internet of Things*. Beberapa diantaranya adalah program milik Microsoft dengan Windows Developer Program for IoT dan Intel dengan IoT Developer Program. (Adhitya, 2015)

Mobil merupakan benda yang tidak asing lagi bagi masyarakat, hal ini dapat dilihat dengan semakin meningkatnya angka penjualan mobil setiap tahun dan banyaknya transportasi seperti mobil yang dapat dilihat di jalanan saat ini. Berdasarkan data yang diperoleh dari Badan Pusat Statistik (BPS), jumlah pengguna kendaraan bermotor di Indonesia mencapai 114.209.266

IOT FOR BEGINNER | SMART CAR

pada tahun 2014. Penggunaan mobil menimbulkan banyak masalah bagi pengendara, seperti pengendara yang baru belajar menggunakan mobil tidak mampu mengendalikan kecepatan mobil, pengendara yang mengantuk saat berkendara sehingga melajukan mobil secara tidak sadar, kurangnya pengetahuan pengendara akan minimum atau maksimum kecepatan untuk area atau jalan yang berbeda, dan kekhawatiran orang tua terhadap anaknya saat sedang berkendara.

Mengingat kebutuhan teknologi yang semakin meningkat dan dengan adanya teknologi *Internet of Things*, maka dapat dijadikan sebagai pedoman dalam membuat strategi yang efektif untuk membuat sebuah aplikasi yang dapat mengatasi masalah yang ada. Dibuat sebuah aplikasi dengan memanfaatkan teknologi *Internet of Things* yaitu "SmartCar"*.* Aplikasi SmartCar merupakan aplikasi yang dapat digunakan oleh semua orang untuk memantau keadaan ketika sedang mengendarai mobil. Fitur yang tersedia pada aplikasi SmartCar yaitu, pelacakan lokasi, mengetahui jarak aman, ramalan cuaca, menentukan suhu dan kelembapan, menentukan getaran, promosi dan notifikasi, serta tombol panik.

Fitur pelacakan lokasi digunakan untuk mengetahui dimana lokasi kendaraan sedang berada. Fitur mengetahui jarak aman yaitu untuk mengetahui jarak ideal suatu kendaraan terhadap benda yang ada disekitarnya. Fitur ramalan cuaca merupakan fitur untuk mengetahui cuaca di lokasi kendaraan berada. Fitur menentukan suhu dan kelembapan yaitu untuk mengetahui suhu dan kelembapan di dalam maupun di luar mobil. Fitur menentukan getaran yaitu untuk mengetahui besar getaran atau guncangan yang terjadi pada mobil. Fitur promosi dan notifikasi digunakan untuk memberitahu tentang promosi yang sedang ada pada lokasi tertentu.

Fitur tombol panik digunakan megirimkan notifikasi pada saat pengendara dalam keadaan mendesak.

Berdasarkan penerapan fitur-fitur yang terdapat pada aplikasi SmartCar, maka diharapkan dapat membantu dan menangani masalah yang ada, sehingga aplikasi SmartCar ini menjadi solusi yang bermanfaat bagi penggunanya.

1.2 Pengenalan *Internet of Things*

CASAGRAS (*Coordination and Support Action for Global RFID-related Activities and Standardisation*) mendefinisikan *Internet of Things*, sebagai sebuah infrastruktur jaringan global, yang menghubungkan benda-benda fisik dan virtual melalui eksploitasi *data capture* dan kemampuan komunikasi. Infrastruktur terdiri dari jaringan yang telah ada dan Internet berikut pengembangan jaringannya. Semua ini akan menawarkan identifikasi obyek, sensor, dan kemampuan koneksi sebagai dasar untuk pengembangan layanan dan aplikasi kooperatif yang independen. Ditandai dengan tingkat otonom *data capture* yang tinggi, *event transfer*, konektivitas jaringan, dan interoperabilitas.

Internet of Things atau dikenal juga dengan singkatan IoT, merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas Internet yang tersambung secara terus-menerus. Kemampuan seperti berbagi data, *remote control*, dan sebagainya termasuk juga pada benda di dunia nyata. Contohnya bahan pangan, elektronik, serta peralatan termasuk benda hidup yang semuanya tersambung ke jaringan lokal dan global melalui sensor yang tertanam dan selalu aktif. *Internet of Things*

mengacu pada benda yang dapat diidentifikasikan secara unik sebagai representasi virtual dalam struktur berbasis *Internet*. Istilah *Internet of Things* awalnya disarankan oleh Kevin Ashton pada tahun 1999 dan mulai terkenal melalui Auto-ID Center di MIT. (David, 2016)

1.3 Sejarah *Internet of Things*

Menurut (Burange & Misalkar, 2015) *Internet of Things* (IOT) adalah struktur dimana obyek, orang disediakan dengan identitas eksklusif, dan kemampuan untuk pindah data melalui jaringan tanpa memerlukan dua arah antara manusia ke manusia yaitu sumber ke tujuan atau interaksi manusia ke komputer.

Tahun 1999 Kevin Ashton menciptakan *The Internet of Things*, direktur eksekutif Auto IDCentre, MIT. Ditemukannya juga peralatan berbasis RFID (*Radio Frequency Identification*) global yang sistem identifikasi pada tahun yang sama. Penemuan ini disebut sebagai sebuah lompatan besar dalam commercializing IoT. Pada tahun 2008 FCC menyetujui penggunaan "*White Space Spectrum*". Akhirnya peluncuran IPv6 di tahun 2011 memicu pertumbuhan besar di bidang *Internet of Things*, perkembangan ini didukung oleh perusahaan raksasa seperti Cisco, IBM, Ericson yang mengambil inisiatif banyak dari pendidikan dan komersial dengan IoT teknologi dapat hanya dijelaskan sebagai hubungan antara manusia dan komputer.

Perkembangan *Internet of Things* menyebabkan semua peralatan yang digunakan dalam kehidupan sehari-hari dapat dikendalikan dan dipantau menggunakan sebuah teknologi. Mayoritas proses yaitu dilakukan dengan bantuan sensor pada IoT. Sensor dikerahkan dimana-mana dan

sensor ini mengkonversi data fisik mentah menjadi sinyal digital dan mengirimkan mereka ke pusat kontrol. Cara ini bisa memonitor perubahan lingkungan jarak jauh dari setiap bagian dari dunia melalui Internet. Arsitektur sistem ini akan didasarkan pada konteks operasi dan proses dalam skenario *real-time*. (Prasada, 2017)

1.4 *Layer Internet of Things*

Arsitektur Internet dengan protokol TCP/IP yang digunakan untuk IoT, tidak bisa menangani jaringan sebesar IoT yang menyebabkan kebutuhan untuk arsitektur baru yang dapat mengatasi berbagai keamanan dan *Quality of Service* (QoS). Pengembangan lebih lanjut dari IoT, sejumlah arsitektur keamanan *multi-layer. Layer* pada *Internet of Things* yang digunakan terbagi atas enam bagian yaitu *coding, perception, network, middleware, application,* dan *business layer.* Masing-masing *layer* dijelaskan bersama dengan penerapan yang akan dilakukan pada SmartCar sebagai berikut.

1.4.1 *Coding Layer*

Coding Layer adalah sebuah lapisan mendasar dalam arsitektur IoT yang menyediakan identifikasi terhadap obyek yang menarik. Lapisan ini memberikan setiap objek sebuah ID unik yang dapat memudahkan untuk membedakan obyek. *Coding Layer* pada umumnya merupakan suatu alat berupa benda fisik atau ID unik untuk mengenali suatu obyek agar dapat didefinisikan untuk proses selanjutnya. *Coding layer* yang terdapat pada aplikasi dan perangkat SmartCar berupa ID pengendara, dimana dengan ID ini dapat mengenali kendaraan yang terdaftar pada sistem.

1.4.2 *Perception Layer*

Perception layer membahas tentang pengumpulan informasi, persepsi obyek dan kontrol obyek. Lapisan persepsi dapat dibagi menjadi dua bagian yaitu persepsi simpul (sensor atau pengendali, dan lain sebagainya) dan persepsi jaringan yang berkomunikasi dengan jaringan transportasi. Persepsi simpul digunakan untuk akuisisi data dan kontrol data, jaringan persepsi mengirimkan data ke *gateway* dikumpulkan atau mengirimkan instruksi kontrol ke *controller*. Teknologi lapisan persepsi mencakup RFID, WSNs, RSN, GPS, dan lain sebagainya.

Pemodelan *Perception Layer* pada aplikasi SmartCar yaitu menggunakan GPS *module*, sensor ultrasonik, sensor getaran, dan sensor suhu dan kelembapan. GPS *module* digunakan untuk melakukan pelacakan lokasi, menentukan kecepatan dan prediksi cuaca pada daerah tertentu. Sensor ultrasonik digunakan untuk menentukan jarak aman kendaraan terhadap benda tertentu. Sensor getaran digunakan untuk menentukan besar guncangan yang terjadi pada kendaraan. Sensor suhu digunakan untuk menentukan suhu di dalam dan di luar kendaraan.

Secara singkat, *perception layer* mendefinisikan sensor-sensor apa saja yang perlu digunakan. Data yang diperoleh dari sensor akan diolah dari *coding layer* sebelum akhirnya dilempar ke *network layer*.

1.4.3 *Network Layer*

Tujuan dari *layer network* ini adalah menerima informasi yang berguna dalam bentuk sinyal digital dari *Perception Layer* dan mengirimkan

ke sistem pengolahan di *Middleware Layer* melalui media transmisi seperti WiFi, Bluetooth, WiMax, Zigbee, GSM, 3G dan lainnya dengan protokol seperti IPv4, IPv6, MQTT, serta DDS.

Pada sistem SmartCar, *controller* yang digunakan yaitu Rasberry Pi 3, dimana sudah dilengkapi dengan Wi-Fi 802.11n dan Bluetooth versi 4.1. Sehingga pengguna yang ingin Raspberry Pi-nya terhubung dengan jaringan Internet tidak perlu lagi menambahkan modul atau menggunakan slot RJ45. Berdasarkan hal tersebut, metode yang dapat digunakan untuk membangun sebuah komunikasi data antar sensor ke *controller* dapat menggunakan koneksi Wi-Fi atau Bluetooth yang sudah terintegrasi langsung dengan *controller* Raspberry Pi.

1.4.4 *Middleware Layer*

Middleware adalah *software* penghubung yang berisi sekumpulan layanan yang memungkinkan beberapa proses dapat berjalan pada satu atau lebih mesin untuk saling berinteraksi pada suatu jaringan. *Middleware* sangat dibutuhkan untuk migrasi dari aplikasi *mainframe* ke aplikasi *client server* dan juga untuk menyediakan komunikasi antar *platform* yang berbeda. Dalam dunia teknologi informasi, terminologi *middleware* merupakan istilah umum dalam pemrograman komputer yang digunakan untuk menyatukan, sebagai penghubung, ataupun untuk meningkatkan fungsi dari dua buah program atau aplikasi yang telah ada. Perangkat lunak *middleware* merupakan perangkat lunak yang terletak diantara program aplikasi dan pelayanan-pelayanan yang ada di sistem operasi.

1.4.5 *Application Layer*

Application layer merupakan lapisan yang berisi aplikasi IoT. Terdapat berbagai aplikasi dari sektor industri yang dapat memanfaatkan IoT yaitu aplikasi yang dapat digunakan khusus untuk sektor industri tertentu dan aplikasi lainnya seperti *Fleet Management*, *Asset Tracking*, dan *Surveillance* yang dapat di beberapa sektor industri (ITU-T, 2012). Lapisan ini merupakan aplikasi dari IoT untuk semua jenis industri berdasarkan data yang diproses karena aplikasi mempromosikan pengembangan IoT sehingga *layer* ini sangat membantu dalam pengembangan skala besar jaringan IoT. Aplikasi IoT yang saling berhubungan menciptakan rumah pintar, transportasi cerdas, dan planet pintar.

1.4.6 Integrasi *Layer*

Internet of Things menggunakan bahasa pemrograman tingkat rendah (bahasa mesin), sehingga memudahkan komunikasi antara perangkat lunak komputer (aplikasi) dengan perangkat keras (*hardware*). Terdapat empat buah integrasi dalam *Internet of Things*. Keempat integrasi tersebut beserta dengan elemennya masing-masing akan dijelaskan sebagai berikut.

1. Integrasi Benda Fisik (*Things Integration*)

Integrasi benda fisik (*things integration*) terjadi antara dunia digital (*digital world*) dan dunia nyata (*real world*). Dunia digital berupa alam semesta di bumi ini yang menjadi lingkungan nyata sehari-hari. Dunia digital mencakup segala hal terkait dengan komputer, proses komputerisasi, dan pemrosesan secara digital yang terjadi didalamnya.

2. Integrasi Data (*Data Integration*)

Integrasi data (*data integration*) terjadi diantara dunia digital dengan dunia jaringan komputer. Jaringan komputer mencankup semua komputer dan perangkat lain yang saling terhubung. Keterhubungan tersebut membentuk komputer terbesar didunia yaitu internet.

3. Integrasi Semantik (*Semantic Integration*)

Inetgrasi semantik (*semantic integration*) terjadi diantara dunia jaringan komputer atau Internet (*computer network world*) dan dunia nyata (*real world*).

4. Integrasi Pengetahuan (*Knowledge Integration*)

Inetgrasi pengetahuan (*knowledge integration*) terjadi antara pengguna, masyarakat, dan komunitas yang mengembangkan dan memanfaatkan *Internet of Things*.

1.5 Integrasi *Layer* pada Fitur SmartCar

Aplikasi SmartCar memiliki enam fitur, dimana antara keenam fitur tersebut melakukan integrasi dari *layer Internet of Things* yang telah ada. Integrasi yang dilakukan pada masing-masing fitur dapat dilihat sebagai berikut.

1.5.1 Fitur Pelacakan Lokasi

Fitur pelacakan lokasi adalah fitur yang dapat digunakan untuk melacak lokasi kendaraan pengguna kemudian hasil pelacakan akan tercatat pada aplikasi kemanapun kendaraan melaju. Integrasi yang dilakukan dalam fitur ini adalah sebagai berikut.

1. Integrasi Benda Fisik (*Things Integration*): GPS *Module*

Fitur pelacakan lokasi berfungsi untuk melakukan pelacakan lokasi kemanapun kendaraan pengguna melaju. GPS *module* akan menangkap data *latitude*, *longitude*, dan kecepatan kemudian data tersebut akan dikirimkan setiap 2 detik ke *webservice*.

2. Integrasi Data (*Data Integration*): Raspberry Pi

GPS *module* akan mengirimkan data ke *webservice* melalui Raspberry Pi. Data pada *webservice* akan diolah menjadi informasi yang akan ditampilkan pada *website* dan aplikasi *mobile*.

3. Integrasi Semantik (*Semantic Integration*): *Webcam*

Integrasi semantik yang terjadi yaitu *webcam* akan menangkap keadaan yang terjadi pada kendaraan pengguna berupa foto kemudian akan tersimpan pada *webservice* lalu diolah untuk menjadi informasi yang ditampilkan pada *website* dan aplikasi *mobile*.

4. Integrasi Pengetahuan (*Knowledge Integration*)

Pengguna akan menerima informasi lokasi kendaraan yang terlacak yang ditampilkan pada *website* dan aplikasi *mobile*

1.5.2 Fitur Jarak Aman

Fitur jarak aman merupakan fitur untuk menentukan jarak aman kendaraan. Integrasi yang dilakukan dalam fitur ini adalah sebagai berikut.

1. Integrasi Benda Fisik (*Things Integration*): Sensor Ultrasonik

Fitur jarak aman berfungsi untuk mendeteksi jarak kendaraan bagian depan dan belakang dengan kendaraan lain atau benda yang terdeteksi. Sensor ultrasonik akan mendeteksi benda yang menghalangi sensor kemudian mengirim data ke *webservice*.

2. Integrasi Data (*Data Integration*): Raspberry Pi

Sensor ultrasonik akan mengirimkan data yang dihasilkan melalui jaringan ke Raspberry Pi untuk diolah menjadi informasi yang dibutuhkan.

3. Integrasi Semantik (*Semantic Integration*)

Tidak terdapat integrasi semantik pada aplikasi SmartCar.

4. Integrasi Pengetahuan (*Knowledge Integration*): *Web*, aplikasi *mobile* dan pengguna aplikasi.

Pengguna dan pengendara menerima informasi dari aplikasi mengenai jarak depan dan belakang kendaraan apabila sensor terdeteksi atau terhalang oleh suatu benda.

1.5.3 Fitur Suhu dan Kelembapan

Fitur deteksi suhu dan kelembapan merupakan fitur pengawasan suhu dan kelembapan udara. Integrasi yang dilakukan dalam fitur ini adalah sebagai berikut.

1. Integrasi Benda Fisik (*Things Integration*): Tidak Ada

Fitur deteksi suhu dan kelembapan berdiri sendiri, tidak terintegrasi dengan benda fisik lainnya seperti dengan sensor lainnya maupun GPS.

2. Integrasi Data (*Data Integration*): Raspberry Pi

Sensor GPS akan mengirimkan data yang dihasilkan melalui jaringan ke Raspberry Pi untuk diolah menjadi informasi yang diinginkan. Raspberry Pi juga mengirimkan data dan informasi ke aplikasi *mobile*.

3. Integrasi Semantik (*Semantic Integration*): *Speaker*

Raspberry Pi mengirimkan perintah atau data ke *speaker* untuk melakukan notifikasi ke pengemudi apabila suhu dan kelembapan di dalam kedaraan lebih tinggi dibandingkan di luar kendaraan.

4. Integrasi Pengetahuan (*Knowledge Integration*): Aplikasi *mobile* dan pengguna aplikasi.

Pengguna *mobile* menerima informasi dari aplikasi mengenai suhu dan kelembapan udara kendaraan, serta menerima notifikasi apabila suhu dan kelembapan di dalam kendaraan lebih tinggi dibandingkan di luar kendaraan.

1.5.4 Fitur Deteksi Getaran

Fitur deteksi getaran yaitu fitur yang terdapat dalam aplikasi SmartCar dengan fungsi sebagai pendeteksi getaran. Integrasi *layer* pada fitur deteksi getaran adalah sebagai berikut.

1. Integrasi Benda Fisik (*Things Integration*): Sensor Getaran

Sensor getaran berfungsi untuk mendeteksi adanya getaran yang terdapat pada suatu tempat/komponen. Sensor ini mampu menangkap jika terjadinya getaran atau goncangan dan dilanjutkan ke proses selanjutnya.

2. Integrasi Data (*Data Integration*): Raspberry Pi

Sensor getaran ini terhubung dengan komponen Raspberry Pi, dimana Raspberry Pi ini untuk mengolah suatu data menjadi informasi yang diinginkan. Raspberry pada aplikasi SmartCar juga melakukan transfer data ke *webservice* dan *mobile*.

3. Integrasi Semantik (*Semantic Integration*)

Tidak terdapat integrasi semantik pada aplikasi SmartCar.

4. Integrasi Pengetahuan (*Knowledge Integration*): *Web Service*, Aplikasi *mobile* dan Pengguna Aplikasi.

Pengguna *webservice* dan *mobile* akan menerima informasi dari Raspberry Pi mengenai getaran mobil.

1.5.5 Fitur Promosi dan Notifikasi

Fitur promosi dan notifikasi adalah fitur yang berfungsi untuk memberikan informasi mengenai promosi ketika pengendara melewati suatu tempat tertentu serta peringatan tentan. Integrasi *layer* pada fitur promosi dan notifikasi adalah sebagai berikut.

1. Integrasi Benda Fisik (*Things Integration*): Sensor Ultrasonik, Sensor Suhu dan Kelembapan, GPS *Module*.

   Sensor Ultrasonik berfungsi untuk mendeteksi jarak aman antara bagian depan dan belakang kendaraan, lalu sensor suhu dan kelembapan berfungsi untuk mendeteksi cuaca yang terhubung dengan sensor GPS, dan GPS module berfungsi untuk mendeteksi lokasi yang terdapat promosi serta data mengenai cuaca.

2. Integrasi Data (*Data Integration*): Raspberry Pi

   Sensor ultrasonik, sensor suhu dan kelembapan, dan GPS module ini terhubung dengan komponen Raspberry Pi, dimana Raspberry Pi akan mengolah data menjadi informasi yang diinginkan. Raspberry Pi pada aplikasi SmartCar juga melakukan transfer data ke web *service* dan *mobile*.

3. Integrasi Semantik (*Semantic Integration*): *Speaker*

   Raspberry Pi mengirimkan perintah atau data ke *speaker* untuk melakukan notifikasi berupa promosi ke pengemudi apabila melewati lokasi yang terdapat promosi, lalu melakukan notifikasi apabila suhu dan kelembapan di dalam kendaraan lebih tinggi dibandingkan di luar kendaraan serta notifikasi mengenai cuaca daerah sekitar.

4. Integrasi Pengetahuan (*Knowledge Integration*): *Web*, aplikasi *mobile* dan pengguna Aplikasi.

Pengguna akan menerima informasi dari Raspberry Pi mengenai promosi, suhu dan kelembapan antara dalam mobil dan luar mobil, serta ramalan cuaca pada *web* dan aplikasi *mobile*

1.5.6 Fitur Tombol Panik

Fitur deteksi suhu dan kelembapan merupakan fitur pengawasan suhu dan kelembapan udara. Integrasi yang dilakukan dalam fitur ini adalah sebagai berikut.

1. Integrasi Benda Fisik (*Things Integration*): *Tactile Switches Push Button*

   Fitur tombol panik digunakan ketika pengguna mengalami keadaan mendesak. Data ketika tombol ditekan akan dikirim ke *webservice* untuk ditampilkan sebagai notifikasi ke *website* dan aplikasi *mobile*.

2. Integrasi Data (*Data Integration*): Raspberry Pi

   Sensor GPS akan mengirimkan data yang dihasilkan melalui jaringan ke Raspberry Pi untuk diolah menjadi informasi yang diinginkan. Raspberry Pi juga mengirimkan data dan informasi ke aplikasi mobile.

3. Integrasi Semantik (*Semantic Integration*): Speaker

   Tidak terdapat integrasi semantik pada aplikasi SmartCar.

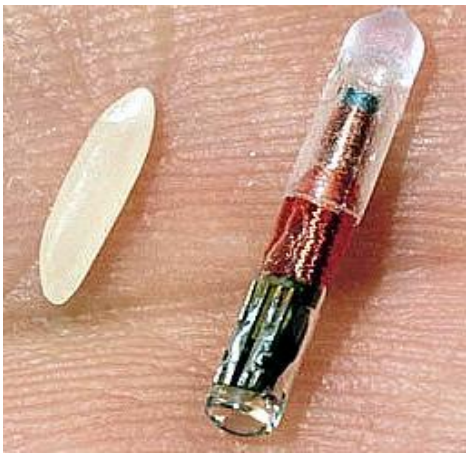
4. Integrasi Pengetahuan (*Knowledge Integration*): *Web*, aplikasi mobile dan pengguna aplikasi.

   Pengguna *mobile* menerima informasi dari aplikasi serta *web* mengenai keadaan pengendara berupa notifikasi dengan menampilkan informasi berupa foto, jenis kendaraan serta plat nomor bahwa pengendara telah menekan tombol.

1.6 Teknologi *Internet of Things*

Teknologi pada *Internet of Things* merupakan hal-hal yang mendukung sehingga aplikasi IoT dapat berjalan sesuai dengan yang diinginkan. Berikut beberapa teknologi yang digunakan sebagai pendukung *Internet of Things.*

1.6.1 RFID (*Radio Frequency Identification*)



Gambar 1.1 RFID *Chip*
(http://www.sosladridos.es)

RFID adalah suatu metode yang mana bisa digunakan untuk menyimpan atau menerima data secara jarak jauh dengan menggunakan suatu piranti yang bernama RFID *tag* atau *transponder*. Suatu RFID *tag* adalah sebuah benda kecil, misalnya berupa stiker adesif, dan dapat ditempelkan pada suatu barang atau produk. RFID *tag* berisi antena yang memungkinkan untuk menerima dan merespon terhadap suatu *query* yang dipancarkan oleh suatu RFID *transceiver*. (Fitriana, 2011)

1.6.2 IP (Internet *Protocol*)

IP adalah suatu aturan atau protokol yang mengatur suatu komunikasi data dalam jaringan Internet. Internet protokol ini akan memberikan penukaran data dari suatu komputer menuju ke komputer lainnya. Protokol atau aturan ini berdiri atas beberapa kumpulan protokol

atau aturan lainnya. Dalam pemakaian Internet sendiri, mungkin hal ini tidak terlalu diperhatikan oleh pengguna Internet. (Helen, 2015)

1.6.3 EPC (*Electronic Product Code*)

EPC adalah sebuah *chip* komputer sebesar kira-kira 320 *micron*, dilengkapi dengan antena dan bersifat pasif, *chip* ini menyimpan data identifikasi untuk suatu produk. EPC adalah standar sistem identifikasi suatu produk yang diperdagangkan dalam jalur perdagangan *supply chain*, digunakan untuk efesiensi dalam pertukaran data elektronis guna *tracking* dan *tracing*. Komputer *chip* ini ditempel atau ditanam dalam kemasan produk yang terkait. Teknologi EPC maka semua barang dapat dideteksi keberadaannya dengan menggunakan perangkat *reader*. (Helen, 2015)

1.6.4 *Barcode*

Gambar 1.2 *Scan Barcode*
(http://www.olsera.com)

Barcode adalah susunan garis cetak vertikal hitam putih dengan lebar berbeda untuk menyimpan data-data spesifik seperti kode produksi, nomor identitas, dll sehingga sistem komputer dapat mengidentifikasi dengan mudah, informasi yang dikodekan dalam *barcode*.

1.6.5 WIFI (*Wireless Fidelity*)

Gambar 1.3 *Router* Wifi
(https://www.alltechbuzz.net)

WIFI merupakan media penghantar komunikasi pada jaringan komputer tanpa menggunakan kabel. WIFI menggunakan sinyal radio yang bekerja pada frekuensi tertentu sehingga transfer data dan program melalui jaringan ini bisa sangat cepat. Umumnya WIFI digunakan untuk mentransfer *Internet* ke pengguna, sehngga dengan adanya WIFI ini semua perangkat yang terhubung bisa terkoneksi dengan *Internet.* (Indra, 2015)

1.6.6 *Bluetooth*

Gambar 1.4 Bluetooth
(http://monitor.espec.ws)

Bluetooth adalah suatu peralatan media komunikasi yang dapat digunakan untuk menghubungkan sebuah perangkat komunikasi dengan perangkat komunikasi lainnya.

Bluetooth umumnya digunakan di *handphone*, komputer atau pc, tablet, dan lain-lain. Fungsi *bluetooth* yaitu untuk mempermudah berbagi atau *sharing file*, audio, menggantikan penggunaan kabel dan lain-lain. (Rahayu, 2015)

1.6.7 ZigBee

Gambar 1.5 Module ZigBee (http://home.roboticlab.eu)

ZigBee adalah standar dari IEEE 802.15.4 untuk komunikasi data pada alat konsumen pribadi maupun untuk skala bisnis. Perangkat ZigBee biasa digunakan untuk mengendalikan sebuah alat lain maupun sebagai sebuah sensor yang *wireless*. ZigBee memliki fitur dimana mampu mengatur jaringan sendiri, maupun mengatur pertukaran data pada jaringan. (Winardi, 2017)

1.6.8 NFC (*Near Field Communication*)

Gambar 1.6 *Near Field Communication* (http://www.dadroidrd.com)

NFC adalah teknologi komunikasi nirkontak yang menggunakan gelombang radio dengan menyentuhkan atau mendekatkan perangkat terkait dalam jarak dekat. Teknologi ini kompatibel dengan infrastruktur kartu cerdas nirkontak dan pembaca kartu cerdas nirkontak, dan telah memiliki spesifikasi yang ditetapkan oleh ISO/IEC, ECMA, ETSI, dan/atau NFC Forum.

1.7 Penerapan *Intenet of Things*

Internet of Things saat ini sudah banyak sekali dikembangan, sehingga penerapan dalam IOT juga terdapat banyak pilihan. Berikut beberapa penerapan dan implementasi *Internet of Things* pada kehidupan saat ini.

1.7.1 *Smart City*

Gambar 1.7 *Smart City*
(https://www.the-iot-marketplace.com/libelium-sofia2-solution-kit)

Secara kasar, definisi dari *Smart City* itu begitu luas mencakup berbagai macam keseluruhan teknologi digital yang dapat meningkatkan kualitas kehidupan, mengurangi biaya dan sumber konsumsi, serta dapat

IOT FOR BEGINNER | SMART CAR

meningkatkan interaksi aktif antara kota dan warganya secara efektif. Cakupan teknologi digital yang dapat diterapkan untuk pengembangan *Smart City* sangat luas dan tidak dibatasi. Penerapan dan aplikasi dari teknologi tersebut juga sangat bervariasi dan dapat diterapkan di semua bidang selama tujuan akhirnya tersebut tercapai. *Internet of Things* mempunyai hubungan yang erat dengan terbentuknya *Smart City* karena IoT merupakan salah satu alat teknologi yang dapat digunakan untuk pengembangan aplikasi *Smart City*. Contoh penggunaan IoT pada *Smart City* yang sudah diterapkan di Indonesia adalah sebagai berikut.

1. Pada aplikasi Informasi Banjir *Online*, selain mengandalkan laporan warga, sensor-sensor banjir yang dapat mengukur ketinggian air secara *real-time* disebarkan ke seluruh wilayah kota sehingga informasi dapat diinformasikan ke *Command Center* secara cepat dan selanjutnya langsung tertangani oleh Dinas terkait.
2. Sistem Notifikasi Gempa dan Tsunami. Beberapa kejadian bencana alam di Indonesia memakan korban jiwa begitu banyak. Jumlah korban jiwa dapat dikurangi secara signifikan apabila *Early Warning System* diterapkan secara benar dan tepat sasaran. Sensor-sensor yang ditempatkan di daerah rawan bencana alam dapat memberikan informasi secara langsung kepada warga sekitar lokasi rawan gempa, longsor, atau tsunami dalam hitungan detik.

1.7.2 *Smart Traffic*

Gambar 1.8 *Smart Traffic*
(https://www.intel.eu/content/www/eu/en/it-managers/iot-smart-cities-russia.html)

Smart Traffic merupakan teknolgi yang digunakan untuk menganalisa lalu lintas kendaraan bermotor di jalan, mulai dari tingkat kemacetan di jalan, kecepatan rata-rata kendaraan bermotor, jalan alternatif jika ada kemacetan, dan sebagainya.

1.7.3 *Smart Environment*

Gambar 1.9 *Smart Environment*
(https://www.the-iot-marketplace.com/solutions/smart-environment)

Smart environment merupakan sebuah teknologi cerdas yang digunakan untuk membuat lingkungan hidup menjadi lebih nyaman. Adanya *smart environment* bertujuan untuk mewujudkan lingkungan yang sehat dan aman, dimana manusia dan benda yang digunakan dapat terhubung melalui jaringan Internet secara terus-menerus. Contoh penerapannya yaitu dengan adanya alat deteksi kebakaran hutan, polusi udara, deteksi dini gempa bumi dan tsunami, dan berbagai bencana alam lain.

1.7.4 *Smart Water*

Gambar 1.10 *Smart Water*
(https://www.the-iot-marketplace.com/libelium-eagle-io-smart-water-solution-kit)

Smart water merupakan teknologi cerdas yang digunakan untuk dapat melakukan manajemen terhadap air. *Smart water* dapat membantu mendapatkan air yang sehat, bersih, bebas polusi, bebas pencemaran bahan kimia di sungai, di laut, maupun di pipa-pipa air, dan deteksi dini terhadap banjir.

1.7.5 *Security and Emergencies*

Gambar 1.11 *Security and Emergencies*
(https://www.redwall.us/)

Smart security dapat membantu meningkatkan keamanan dan membantu dalam situasi darurat. Misalnya seperti mendeteksi manusia di suatu area, mendeteksi cairan, radiasi, gas-gas yang bisa meledak. Perangkat yang digunakan serta sensor yang tertanam akan bekerja secara terus-menerus melalui jaringan Internet sehingga dapat lebih mudah untuk melakukan pengamanan ataupun pencegahan.

1.7.6 *Smart Agriculture*

Gambar 1.12 *Smart Agriculture*
(https://www.the-iot-marketplace.com/libelium-cumulocity-smart-agriculture-solution-kit)

Smart agriculture adalah pemanfaatan teknologi cerdas dalam bidang pertanian. Adanya teknologi ini digunakan untuk mendorong usaha

pertanian, sehingga pihak yang membutuhkan merasa sangat terbantu dalam melakukan pekerjaannya. Contohnya seperti dapat mendeteksi kelembapan tanah, udara, ukuran batang pohon, cuaca, suhu, dan sebagainya.

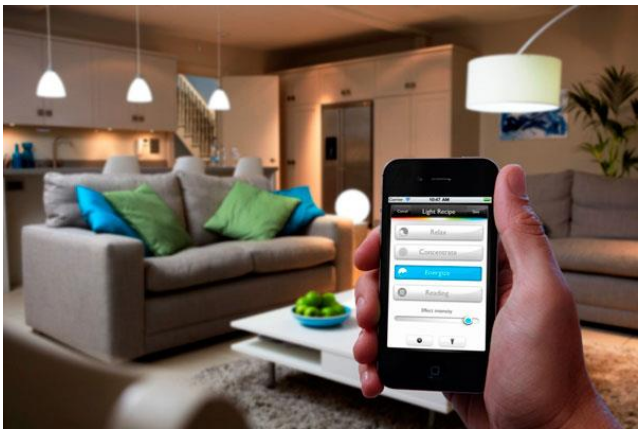
1.7.7 *Smart Animal Farming*

Gambar 1.13 *Smart Animal Farming*
(http://www.thetrentonline.com/4-smart-investment-opportunities/)

Smart animal farming merupakan teknologi cerdas yang dimanfaatkan dalam bidang peternakan. Penerapannya seperti mendeteksi keberadaan ternak, mendeteksi gas beracun, sehingga bisa digunakan untuk mengontrol kembangbiak ternak, melacak keberadaan ternak, dan memastikan usaha peternakan berkembang dengan maksimal.

IOT FOR BEGINNER | SMART CAR

1.7.8 *Home Automation*



Gambar 1.14 *Home Automation*
(https://yeltech.com/applications/home-automation-3/)

Home automation yaitu memanfaatkan teknologi untuk melakukan pengendalian dan pengawasan terhadap segala hal yang terdapat dalam ruangan khususnya rumah pribadi. *Home automation* dapat digunakan untuk memonitor penggunaan energi, air, mendeteksi pintu dan jendela terbuka atau tertutup, mendeteksi keberadaan manusia atau binatang, sehingga mewujudkan rumah yang hemat energi dan aman.

1.7.9 *E-Health*

Gambar 1.15 *E-Health*
(https://www.the-iot-marketplace.com/mysignals-sw-ehealth-medical-biometric-complete-kit)

E-Health merupakan teknologi yang digunakan untuk melakukan pengawasan terhadap kesehatan. Teknologi ini memungkinkan perangkat-perangkat *wearable* sampai *tablet* (pil) bisa saling tersambung. Hal tersebut akan mendorong industri *wearable* sensor, sampai *tablet* (pil), dan sensor yang bisa ditanam di dalam tubuh manusia.

1.8 Sistem Operasi *Internet of Things*

Sistem operasi pada perangkat IoT sering disebut *juga Embeded operating system* atau *embeded software*. Jenis dari sistem operasi yang digunakan dalam *embeded device* bertipe *Real Time Operating System* (RTOS) disebut RTOS karena sistem operasi ini secara *real time*, menerima data, memproses, dan menghasilkan keluaran secara *real time*.

1.8.1 RIOT OS

RIOT adalah sistem operasi yang dikembangkan oleh komunitas *opensource* yang dimulai sjak tahun 2008, RIOT dapat berjalan di berbagai *platform* termasuk *embeded device*. RIOT OS dikenal juga akan efesiensi dalam penggunaan *power* dan membutuhkan spesifikasi yang sangat minim.

1.8.2 Windows 10 for IoT

Versi terakhir dari sistem operasi untuk *embeded device* milik Microsoft adalah Windows 10 for IoT. Ada 3 jenis 3 varian dari Windows for embeded OS: yang pertama windows 10 for IoT Mobile dimana men-*support* ARM *architecture*. Berikutnya Windows 10 for IoT Core yang dapat berjalan di Rasberry Pi dan *inter* Atom dan yang terakhir adalah Windows 10 for IoT enterprise. Windows 10 for IoT sangat baru sehingga masih tertinggal untuk

dukungan *support* dan *framework*, tetapi pada *developer* sistem operasi ini sangat potensial untuk mengembangkan aplikasi *smart house*.

1.8.3 WindRiver VxWorks

WindRiver VxWorks menjadi RTOS (*Real Time Operating System*) komersial yang banyak digunakan saat ini, karena pengemebangan *embeded* OS ini sudah mengembangkan *security* yang sangat kritis ketika implementasi perangkat IoT dalam projek besar dan vital. Sistem operasi ini banyak digunakan dalam bidang medis, penerbangan dan industri.

1.8.4 Google Birllo

Brillo merupakan *embeded* OS besutan Google berbasis Android. Brillo menggunakan protokol khusus untuk berkomunikasi yang disebut *Weave* dimana *embeded device* tidak harus diinstall sistemoperasi berbasis android agar bisa berkomunikasi. Selama perangkat iot menggunakan *protocol* Weave semua perangkat bisa saling berkomunikasi. Ini menjadi terobosan besar ketika antar perangkat IoT dengan *vendor* yang berbeda mengalami masalah kompatibilitas dalam berkomuniksi.

1.8.5 ARM Mbed OS

ARM membuat embeded OS sendiri yang mereka namakan Mbed OS, OS ini hanya kompatibel dengan *embeded device* yang menggunakan ARM arsitektur. OS ini hanya didesain dengan *single traded* bukan *multi traded*. ARM menganggap OS ini akan dapat berjalan di *embeded* OS yang sangat kecil dengan kebutuhan *power* yang sangat kecil pula sebagai terobosan masa depan, dan pada akhirnya semua IOT *device* menggunakan OS ini.

1.8.6 Embedded Apple iOS and OS X

Ketika IoT market mulai ramai dan menjanjikan Apple mengadopsi *platform* iOS untuk membuat Perangkat IoT seperti Apple TV, CarPlay dan Apple Watch. Untuk kedpan nya Apple akan menggunakan iOS dan memodifikasi OS X agar dapat digunakan di embeded *device* dengan efisiensi yang tinggi.

1.8.7 Nucleus RTOS

Nucleus RTOS adalah *embedded* OS dikembangkan oleh Mentor Graphics yang mengklaim bahwa *software*-nya telah digunakan lebih dari 3 Milyar perangkat. Nucleus RTOS menawarkan *power management framework* yang dapat mengoptimalkan penggunaan daya pada perangkat *Internet of Things*.

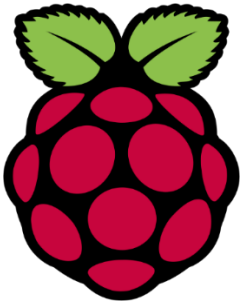
BAB 2

Komponen IoT SmartCar

Raspberry Pi
Sensor Ultrasonik
Sensor Getaran
Sensor Suhu dan Kelembapan
Speaker
Webcam
Button
Kabel Jumper
Resistor
Breadboard
Adaptor
Global Positioning System (GPS)
Android
Web Service

2.1 Raspberry Pi

Raspberry Pi merupakan sebuah komputer mini yang memiliki fitur yang hampir sama dengan komputer pada umumnya, seperti membuat program, *office*, menonton video resolusi tinggi, dan lain sebagainya. Raspberry Pi menyediakan *port* USB, LAN, *jack audio*, serta HDMI untuk *input* dan *output* dengan menghubungkan *mouse*, *keyboard*, dan TV/monitor, maka Raspberry Pi akan bekerja seperti sebuah komputer. Konsumsi daya yang digunakan hanya sekitar 10 watt. (Tandy, 2012)



Gambar 2.1 Logo Raspberry Pi

Nama Raspberry berdasarkan pernyataan raspberrypi.org terinspirasi dari berbagai vendor terkenal yang menggunakan nama buah-buahan (seperti Apple, BlackBerry, dan Apricot). Nama Pi merupakan kepanjangan dari Python, yaitu sebuah bahasa pemrograman interpretatif serbaguna. Raspberry Pi, pada awalnya memiliki konsep sebuah komputer sederhana yang dapat diprogram dengan menggunakan bahasa Python.

2.1.1 Sejarah Raspberry Pi

Raspberry Pi Foundation, perusahaan yang membuat Raspberry Pi (Raspi) didirikan oleh Eben Upton dkk pada tahun 2006. Dikutip dari halaman tentang dari raspberrypi.org, dalam artikel yang berjudul "The Making of Pi" disebutkan ide dibalik dibuatnya komputer super kecil dan terjangkau untuk anak-anak muncul pada tahun 2006, ketika Eben Upton, Rob Mullins, Jack Lang dan Alan Mycroft, dari Laboratorium Komputer di Universitas Cambridge, menjadi khawatir mengenai penurunan level kemampuan komputer dari tahun ke tahun para siswa yang tertarik untuk mempelajari Ilmu Komputer (Jaseman, 2012).

Papan prototipe pertama, pada awalnya hanyalah sebesar *flashdisk*, yang memiliki 1 *port* USB dan *port* HDMI. Papan prototipe juga memiliki *slot* MicroSD untuk menyimpan OS Linux. Papan prototipe terlalu kecil untuk menambahkan *port* LAN, *port* GPIO, atau *output* audio, yang artinya papan *Prototype* kurang cocok untuk dikembangkan. Sehingga diputuskan untuk membuat papan sebesar kartu kredit, dan bekerja sama dengan Linux untuk membuat sistem operasi yang memungkinkan untuk berkerja pada processor 700MHz ARM1 176JZF-S (Jaseman, 2012).

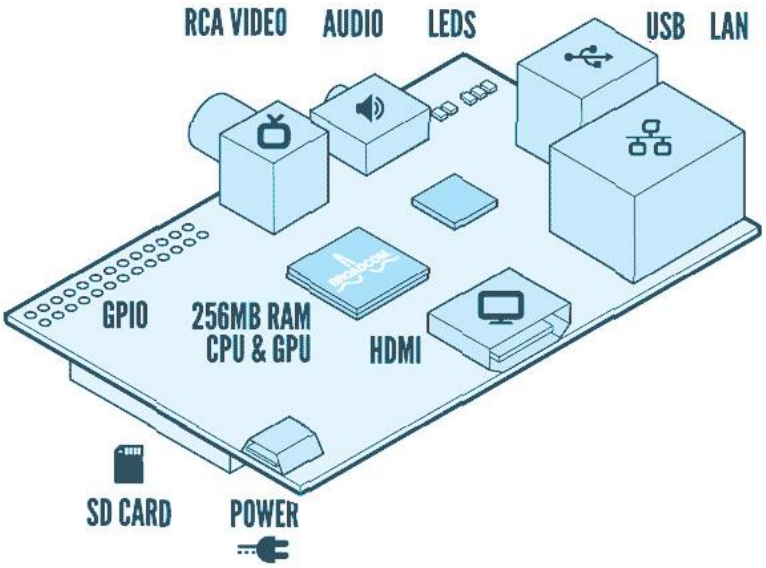
Sebanyak 50 unit Raspberry Pi model Alpha diproduksi pada bulan Agustus 2011. Kemudian pada Bulan Desember 2011, Raspberry Pi model Beta mampu memainkan video HD 1080p menggunakan *onboard Videocore* IV GPU. Raspberry Pi model Beta memiliki konektor daya tipe *micro-usb*, yang memungkinkan para pengguna untuk menghidupkan perangkat tersebut menggunakan *charger* ponsel biasa (Jaseman, 2012).

Raspberry Pi Foundation menjual beberapa prototipenya di Ebay pada bulan Januari 2012. Diumumkan juga sebanyak 10.000 unit komputer Raspberry Pi sedang diproduksi di Cina. Raspberry Pi tersebut dijual melalui komponen industri suplier "RS Components International" dan "Premier Farnell" dengan desain tambahan untuk para *engineer* yang dirancang oleh "Element 14" (Jaseman, 2012).

Raspberry Pi akhirnya siap untuk dijual secara *pre-order* dan peluncuran resmi diadakan pada tanggal 29 Februari 2012 Pukul 06.00 waktu setempat setelah melalui rintangan-rintangan tersebut. Raspberry Pi kini telah diproduksi dengan berbagai model lainnya, seperti Raspberry Pi 2 model B, dan yang terbaru adalah Raspberry Pi 3 model B yang dirilis pada Februari 2016 (Jaseman, 2012).

2.1.2 Komponen Raspberry Pi

Perangkat Raspberry Pi terlihat seperti *motherboard*, dengan dipasang *chip* dan *port* terbuka. Raspberry Pi memiliki semua komponen yang dibutuhkan untuk menghubungkan *input*, *output,* dan perangkat penyimpanan dan komputasi. Model perangkat Raspberry Pi ada dua yaitu, Model A dan Model B. Satu-satunya perbedaan yang nyata antara kedua model tersebut adalah penambahan *Ethernet* dan *port* USB tambahan pada Model B yang lebih mahal.



Gambar 2.2 Diagram Raspberry Pi
(Sumber: http://computer.howstuffworks.com/raspberry-pi2.html)

Gambar 2.2 menunjukkan diagram Raspberry Pi yang pada umumnya memiliki berbagai komponen seperti berikut.

1. ARM CPU/GPU, ini adalah sistem *Broadcom* BCM2835 pada *Chip* (SoC) yang terdiri dari ARM *Central Processing Unit* (CPU) dan *VideoCore* 4 *Graphics Processing Unit* (GPU). CPU menangani semua perhitungan yang membuat komputer bekerja (mengambil *input*, melakukan perhitungan dan menghasilkan *output*), dan GPU menangani keluaran grafis.
2. GPIO memiliki tujuan utama sebagai koneksi *input*/ *output* umum yang memungkinkan para pecinta perangkat keras untuk bereksperimen.

3. RCA merupakan sebuah RCA *jack* yang memungkinkan koneksi dari TV analog dan perangkat *output* serupa lainnya.
4. Audio *out* adalah *jack* standar 3,55 milimeter untuk koneksi dari perangkat *output* audio seperti *headphone* atau *speaker*.
5. LED (*Light-Emitting Diodes*) digunakan untuk semua kebutuhan lampu indikator.
6. USB merupakan *port* koneksi umum untuk perangkat periferal dari semua jenis (termasuk *mouse* dan *keyboard*). Model A memiliki satu, dan Model B memiliki dua. Hub USB dapat digunakan untuk memperluas jumlah *port* atau *plug mouse* ke *keyboard* jika memiliki *port* USB sendiri.
7. HDMI merupakan konektor yang digunakan untuk menghubungkan televisi definisi tinggi atau perangkat lain yang kompatibel menggunakan kabel HDMI.
8. Power adalah konektor *power* USB Micro 5V untuk dipasang *power supply* yang kompatibel.
9. SD *cardslot* adalah slot kartu SD berukuran penuh. SD *card* dengan sistem operasi dipasang diperlukan untuk melakukan *booting* pada perangkat.
10. *Ethernet* adalah konektor yang digunakan untuk akses jaringan kabel dan hanya tersedia pada Model B.

    Banyak fitur yang belum ada, seperti WiFi dan audio *in*, dapat ditambahkan dengan menggunakan *port* USB atau hub USB.

2.1.3 Model Raspberry Pi

Raspberry Pi kini memiliki 8 model, Senin, 2 Februari 2015 Raspberry Pi Foundation merilis Raspberry Pi Versi 2 dengan spesifikasi QuadCore ARMv7, dan RAM sebesar 1GB plus didukung oleh Windows 10 secara cuma-cuma (gratis). Dilanjutkan dengan Raspberry Pi Zero yang dirilis pada 26 Nopember 2015 dan Raspberry Pi 3 yang dilengkapi dengan Wi-Fi dan Bluetooth *built-in* pada tanggal 29 Februari 2016. Raspberry Pi sejak dirilis pada tahun 2012 telah memiliki lima model, empat diantara dapat digunakan oleh orang umum namun satu untuk tujuan pengembangan. Ulasan dari model-model Raspberry Pi yaitu sebagai berikut.

1. Model Raspberry Pi Model A

Gambar 2.3 Raspberry Pi Model A
(Sumber: https://tutorkeren.com)

Model A adalah perangkat yang paling dasar, dengan satu buah USB port dan 256MB SDRAM. *Port* pada *board*-nya terdiri dari:

1) *Full size* SD *card*
2) HDMI *output port*
3) *Composite video output*
4) 1 USB *port*

5) 26 pin *header* GPIO, I2C dll
6) 3.5mm *audio jack*
7) 1 *Camera interface port* (CSI-2)
8) 1 LCD *display interface port* (DSI)
9) 1 mircoUSB *power connector* untuk menyalakan perangkat

2. Model Raspberry Pi Model A

Gambar 2.4 Raspberry Pi Model A+
(Sumber: https://tutorkeren.com)

Dirilis pada November 2014, Raspberry Pi model A+ adalah varian 'plus' dari model A. Memiliki 40 GPIO pin, satu USB *board*, tanpa ethernet dan 256MB SDRAM. Juga memiliki form factor yang lebih kecil dari model yang lain dengan panjang 65mm.

3. Model Raspberry Pi Model B

Gambar 2.5 Raspberry Pi Model B
(Sumber: https://tutorkeren.com)

Raspberry Pi model B adalah perangkat yang memiliki spesifikasi tinggi. Memiliki dua port USB, dan RAM sebesar 512MB SDRAM. Memiliki Port tambahan yang tidak terdapat pada model A, yaitu satu buah *port* ethernet dan satu buah *port* USB sehingga total memiliki dua buah *port* USB.

4. Model Raspberry Pi Model B+

Gambar 2.6 Raspberry Pi Model B+
(Sumber: https://tutorkeren.com)

Dirilis pada Juli 2014, model B+ adalah pembaharuan revisi dari model B. Terdapat penambahan jumlah USB *port* menjadi 4 dan jumlah pin *header* GPIO menjadi 40. Sebagai tambahan, model ini memiliki sirkuit power *supply* yang lebih baik, yang memungkinkan perangkat USB yang memerlukan daya besar untuk digunakan pada Raspberry dengan *mode hot-plugged*. *Composite video connector* yang menonjol besar telah dihilangkan dan digantikan dengan jack audio/video 3.5mm. SD *Card full size* juga diganti dengan versi yang lebih robust yaitu slot microSD. Berikut adalah daftar rinci beberapa peningkatan model B+ dari model B yaitu:

1) Monitor arus pada *port* USB yang berarti model B+ sekarang telah mendukung *hot-plugging*.
2) Pembatas arus pada sumber daya 5V untuk HDMI yang berarti semua VGA konverter yang menggunakan daya dari kabel HDMI bisa digunakan.
3) 14 pin GPIO tambahan.
4) Dukungan EEPROM readout untuk papan ekspansi baru HAT.
5) Kapasitas drive yang lebih tinggi untuk audio out analog, dari regulator terpisah, yang berarti kualitas audio DAC yang lebih baik.
6) Tidak ada lagi masalah dengan *backpowering* (daya lain masuk dari USB port bukan dari port power), karena pembatas arus USB yang juga mencegah aliran balik.
7) *Composite video out* dipindahkan ke jack 3.5mm.
8) Konektor sekarang dipindahkan ke dua sisi papan ketimbang menggunakan empat sisi papan. 4 lobang pasang yang ditaruh

dengan posisi segi panjang sehingga memudahkan untuk pemasangan pada *casing*.

5. Model *Compute* Modul

Compute Modul diperuntukan bagi penggunaan industri, merupakan versi potongan yang hanya menyertakan *chip* BCM2835, 512MB SDRAM dan 4GB eMMC *flash* memori, dalam *form factor* berukuran kecil. Modul ini dihubungkan dengan papan dasar menggunakan konektor 200 pin DDR2 SODIMM yang telah dimodifikasi dan bukan merupakan perangkat yang kompatibel dengan SODIMM, namun hanya menggunakan konektor yang sama dengan SODIMM. Semua fitur dari BCM2835 dipaparkan melalui konektor SODIMM, termasuk dua buah kamera dan LCD *port*, sementara model A dan B hanya memiliki satu.

Gambar 2.7 Raspberry Pi Model *Compute* Modul
(Sumber: https://tutorkeren.com)

Compute modul diharapkan dapat digunakan oleh perusahaan yang berharap untuk dapat mempercepat proses pengembangan dari produk baru, berarti hanya *board* dasarnya saja yang perlu dibuat, dengan periperal

yang sesuai dan dengan *Compute* modul yang menyediakan CPU, memori dan penyimpanan dengan perangkat lunak yang teruji dan terpercaya.

2.1.4 Harga Raspberry Pi 3 Model B

Harga dari Raspberry Pi 3 Model B cukup terjangkau untuk sebuah Mini PC dengan perfoma diatas rata-rata. Harga Raspberry Pi 3 Model B adalah Rp. 600.000,00.

2.1.5 Instalasi Raspbian

Instalasi Raspbian pada Raspberry Pi termasuk sangat sederhana dan mudah, tidak seperti instalasi sistem operasi pada komputer atau laptop biasanya. Beberapa hal yang perlu dipersiapkan mulai dari perangkat keras maupun perangkat lunak. Perangkat keras yang perlu dipersiapkan yaitu:

1. Monitor LCD, LED, dan lain sebagainya yang mendukung *input*-an HDMI.
2. Kabel HDMI
3. *Keyboard* USB
4. *Mouse* USB
5. MicroSD Card kosong minimal 8GB (Class 10)
6. *Card Reader*
7. Raspberry Pi 3 Model B
8. Raspberry Pi *adaptor* 5V
9. Komputer atau Laptop yang memiliki sistem operasi Windows

Perangkat lunak yang harus dipersiapkan untuk dapat melakukan instalasi Raspbian pada Raspberry Pi adalah sebagai berikut.

1. Sistem Operasi Raspbian Jessie with Pixel terbaru yang dapat di-*download* pada halaman https://www.raspberrypi.org/downloads/raspbian/
2. Winrar, Winzip, atau program yang dapat melakukan ekstraksi partisi Zip.
3. *Software* Win32 Disk Imager (windows), yang dapat di-*download* pada halaman https://sourceforge.net/projects/win32diskimager/.

Perangkat keras dan perangkat lunak tersebut sangatlah penting dan diperlukan untuk instalasi awal Raspbian pada Raspberry Pi. Setelah perangkat-perangkat tersebut siap, maka selanjutnya adalah proses *wirte* sistem operasi Raspbian ke MicroSD Card yang dapat dilakukan menggunakan komputer atau laptop windows, berikut adalah langkah-langkahnya:

1. Instal Win32 *Disk Imager* pada komputer atau laptop
2. Unzip sistem operasi Raspbian pada direktori tertentu di komputer atau laptop sehingga Raspbian bertipe file Disc Image File (.img).
3. Masukan MicroSD Card ke komputer atau laptop menggunakan *Card Reader.*
4. Format MicroSD Card
5. Jalankan *software* Win32 Disk Imager

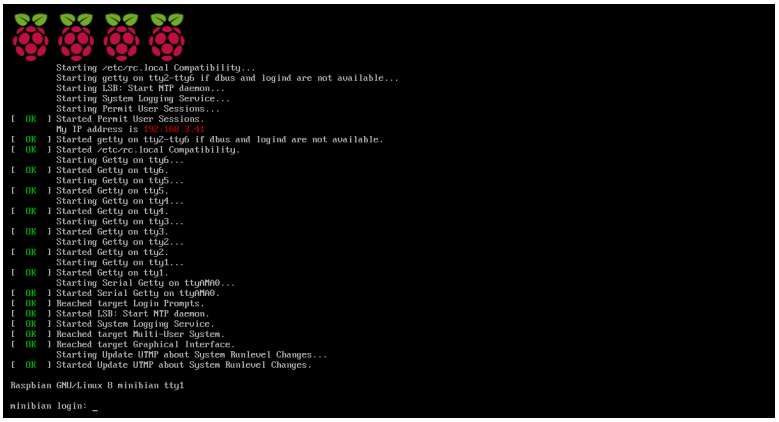
Gambar 2.8 Pilih Direktori MicroSD *Card* dan OS

6. Pertama pilih *Device* atau direktori MicroSD Card. Kedua, pilih tempat atau direktori letak *file image* sistem operasi Raspbian disimpan.

Gambar 2.9 *Write* OS Raspbian ke MicroSD *Card*

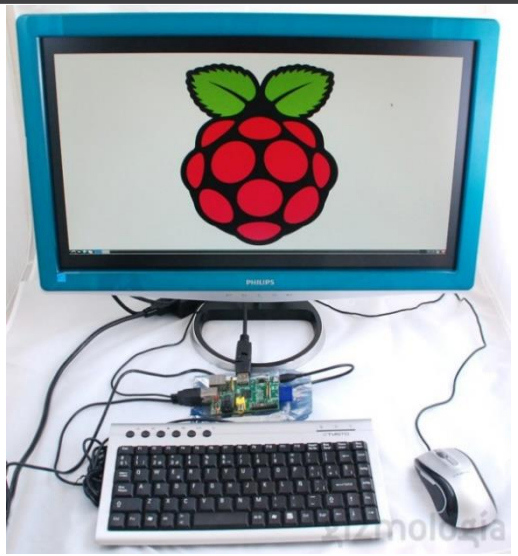
7. Ketiga, klik tombol "Write". Tunggu proses selesai sehingga MicroSD Card telah berisi sistem operasi Raspbian.

8. Pasang perangkat keras *Keyboard* USB, *Mouse* USB, Monitor menggunakan kabel HDMI, dan *Adaptor* ke masing-masing port Raspberry Pi 3 Model B, serta yang paling penting adalah MicroSD Card yang sebelumnya sudah di-*write* sistem operasi Raspbian. Sehingga kini Raspberry Pi benar-benar berfungsi sebagai komputer.
9. Raspberry Pi dapat dihidupkan dengan cara menghubungkan *Adaptor* Raspberry Pi dengan sumber listrik.



Gambar 2.10 Instalasi Raspbian

10. Sistem operasi Raspbian akan langsung terinstal pada Raspberry Pi ketika Raspberry Pi dihidupkan.



Gambar 2.11 Konfigurasi Awal Perangkat Keras pada Raspberry
(Sumber: https://pccontrol.wordpress.com/2014/06/17/pengetahuan-dasar-dan-pemrograman-raspberry-p/)

11. Raspberry Pi kini dapat digunakan seperti sistem operasi komputer pada biasanya.

2.1.6 *Remote Raspberry*

Raspberry Pi sebagai komputer mini dapat di-*remote* atau dikendalikan menggunakan komputer lain. Tujuan melakukan *remote* ini adalah agar Raspberry Pi tidak selalu menggunakan perangkat keras monitor, *keyboard*, maupun *mouse* untuk melakukan fungsinya. Misalnya dalam IOT, Raspberry yang terlah terhubung ke alat atau sensor tidak dimungkinkan selalu menggunakan monitor, *mouse*, *keyboard* untuk melakukan konfigurasi, pengaturan, dan proses lainnya.

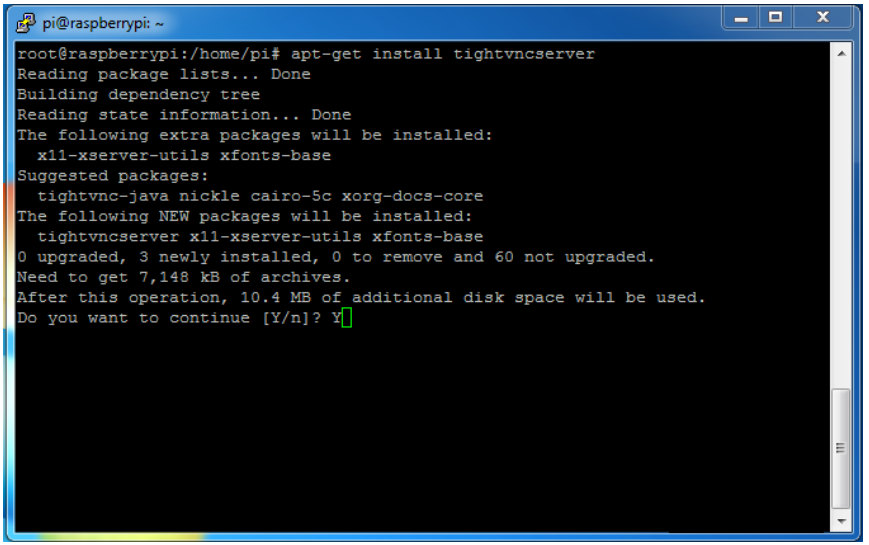
Terdapat beberapa cara melakukan *remote* Raspberry Pi, namun yang paling banyak digunakan adalah TightVNC *Server*. *Virtual Network Computing* yang biasa disingkat dengan VNC adalah suatu aplikasi sistem yang biasa digunakan untuk me-*remote* komputer dengan modus grafis. Perangkat lunak yang perlu disiapkan pada komputer atau laptop yaitu:

1. *Advance* IP *Scanner*, dapat di-*download* pada halaman http://www.*Advance*d-ip-*Scanner*.com/. *Advance* IP *Scanner* ini berfungsi untuk mencari IP *Address* dari perangkat yang berhubung dalam satu jaringan.
2. PuTTY, dapat di-*download* pada halaman http://www.putty.org/. PuTTY adalah *software remote console*/*terminal* yang digunakan untuk me-*remote* komputer dengan terhubungnya menggunakan *port* SSH atau sebagainya.
3. VNC *Server*, dapat di-*didownload* pada halaman http://www.tightvnc.com/. TightVNC *Server* merupakan *software* untuk melakukan *remote* komputer secara grafis.

Langkah-langkah dalam menggunakan TightVNC *Server* untuk melakukan *remote* Raspberry Pi. Pertama instalasi TightVNC *Server* pada Raspberry Pi sebagai berikut:

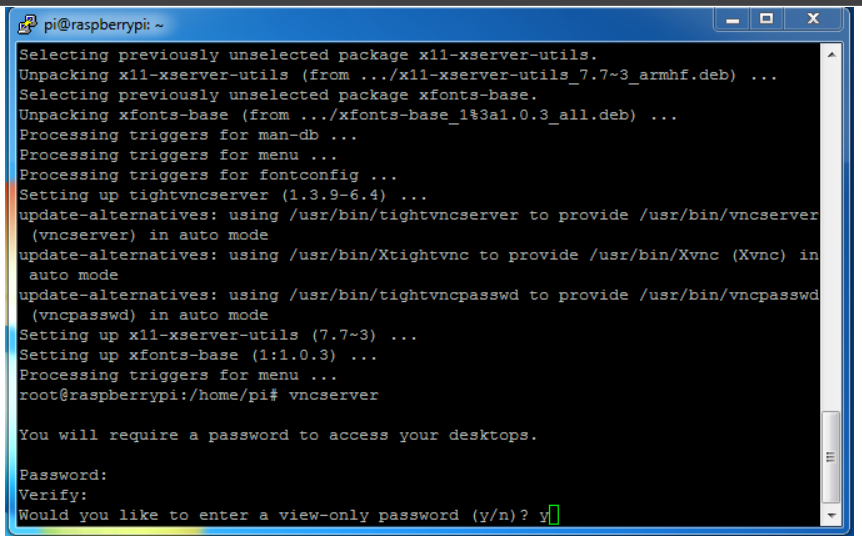
1. Hubungkan Raspberry Pi ke jaringan internet menggunakan WiFi, modem, ataupun ethernet. Lebih baik mencatat IP *Address* Raspberry Pi yang telah terkoneksi internet, karena kemungkinan IP *Address* relatif jarang berubah. Caranya dengan mengetik perintah pada terminal `sudo ip adrr show`. Apabila menggunakan koneksi WiFi,

biasanya Raspberry Pi akan selalu mengingat setiap *boot* SSID dan *Password* Wifi yang dikonfigurasi.



Gambar 2.12 Instalasi TightVNC *Server* pada Raspberry Pi

2. Instal TightVNC *Server* pada Raspbian Raspberry Pi menggunakan perintah `sudo apt-get install tightvncServer` pada terminal Raspbian di Raspberry Pi.



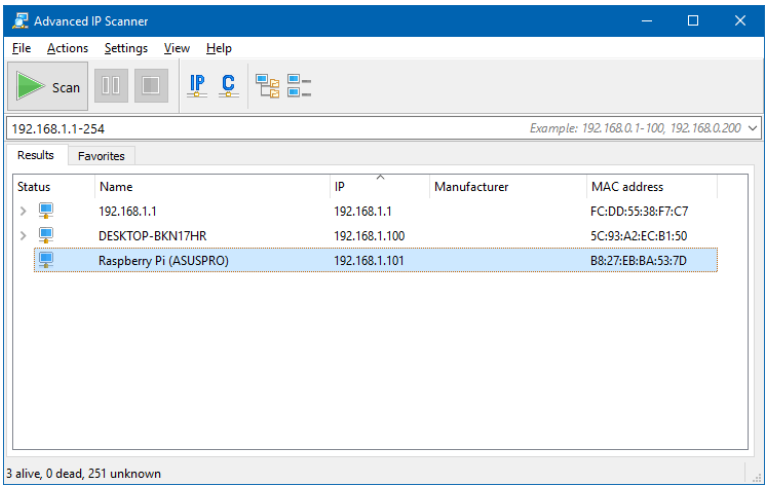
Gambar 2.13 Konfigurasi *Password* TightVNC *Server*

3. Tunggu beberapa saat, kemudian akan muncul konfigurasi *password* yang digunakan untuk *remote*.
4. Cara menjalankan *service* TightVNC *Server* adalah dengan *command* `vncServer :1`.

TightVNC *Server* kini sudah terpasang pada Raspberry Pi sebagai komputer yang akan di-*remote.* Langkah selanjutnya adalah melakukan instalasi Tight VNC *Server* pada komputer atau laptop yang digunakan untuk me-*remote* Raspberry Pi sebagai berikut:

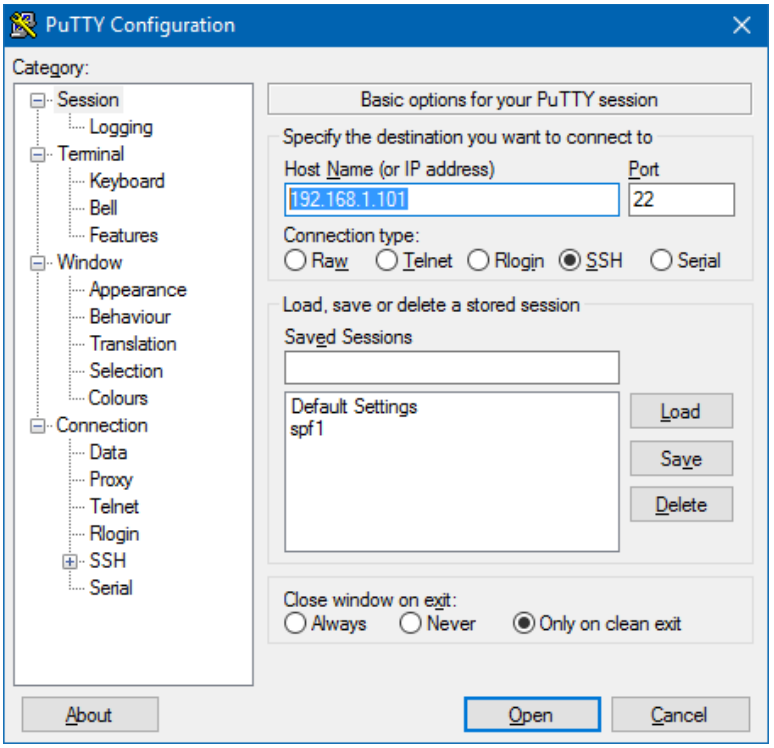
1. Instal TightVNC *Server* untuk desktop windows pada komputer atau laptop.
2. Instal *Advance* IP *Scanner* untuk desktop windows pada komputer atau laptop.
3. Instal PuTTY untuk desktop windows pada komputer atau laptop.

4. Melakukan *remote* hanya dapat dilakukan ketika Raspberry Pi dan komputer atau laptop yang digunakan untuk *remote* berada pada satu jaringan.
5. Jalankan *Advance* IP *Scanner* untuk mendeteksi IP *Address* yang berada pada jaringan yang sama dan untuk menemukan IP *Address* dari Raspberry Pi. Apabila sebelumnya IP *Address* sempat untuk dicatat, besar kemungkinan IP *Address* tersebut merupakan IP *Address* dari Raspberry Pi karena biasanya IP *Address* relatif jarang berubah pada jaringan yang sama setelah pertamakali terkoneksi.



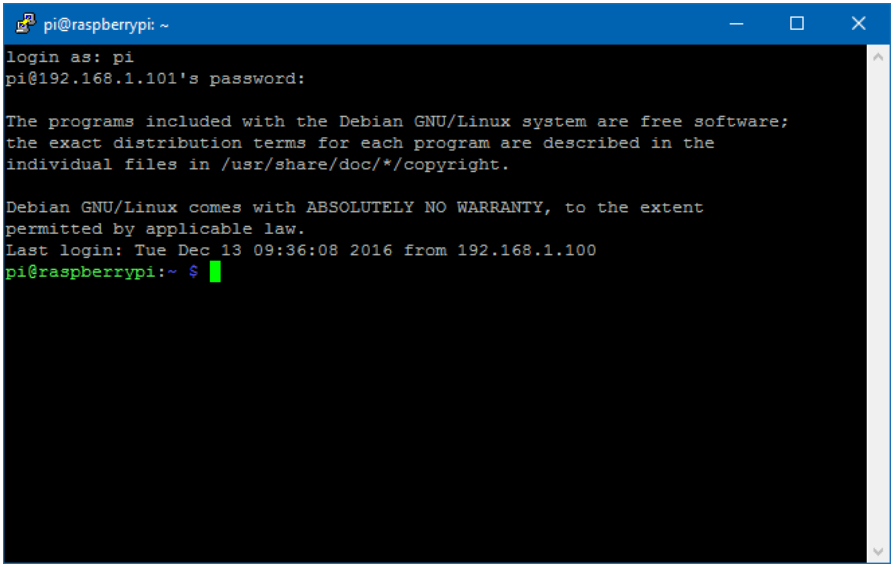
Gambar 2.14 *List* IP *Address* hasil *Advance* IP *Scanner*

6. Gambar 2.14 merupakan tampilan hasil *scan Advance* IP *Scanner* menunjukan bahwa Raspberry Pi memiliki IP *Address* 192.168.1.101. IP *Address* ini kemudian digunakan untuk melakukan *remote* menggunakan TightVNC *Server*.
7. Jalankan PuTTY kemudian masukan IP *Address* Raspberry Pi yang di dapat dari hasil scan *Advance* IP *Scanner* seperti yang ditunjukan pada Gambar 2.15.



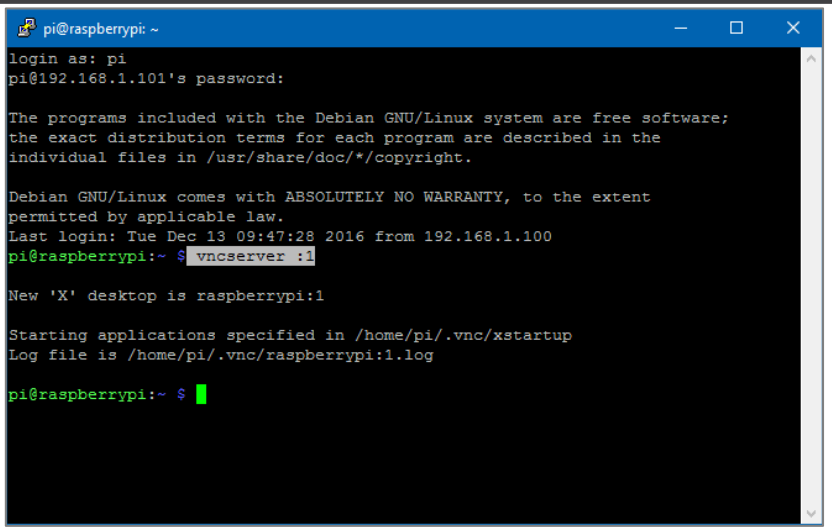
Gambar 2.15 *Remote* SSL Raspberry Pi Menggunakan PuTTY

8. Klik tombol *Open* maka akan muncul tampilan seperti Gambar 2.16.



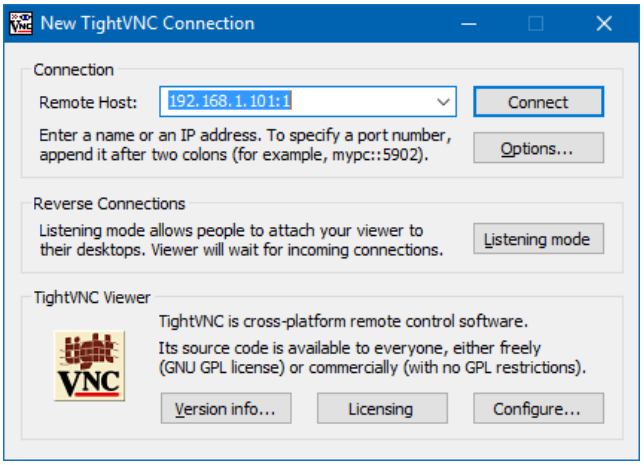
Gambar 2.16 *Login* ke Raspberry Pi menggunakan PuTTY

9. Masukan *login* as dan *password* Raspberry Pi. Secara *default, username* Raspberry Pi adalah *pi* dan password Raspberry Pi adalah *raspberry*.
10. Langkah selanjutnya setelah masuk ke Raspberry Pi adalah menjalankan VNC *Server* dengan *command* pada PuTTY VNC *Server* seperti yang ditunjukan pada Gambar 2.17.



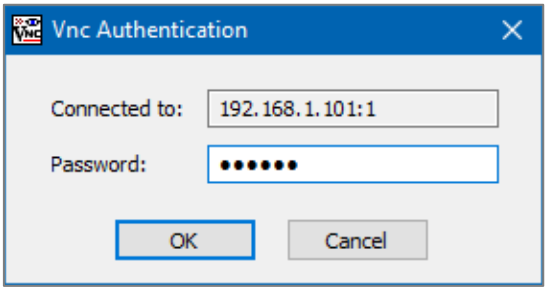
Gambar 2.17 Menjalankan TightVNC *Server*

11. Gambar 2.17 menunjukan bahwa *service* vnc *Server* sudah berjalan, selanjutnya jalankan program TightVNC *Server* pada komputer atau Laptop sehingga muncul jendela seperti yang ditunjukan pada Gambar 2.18.



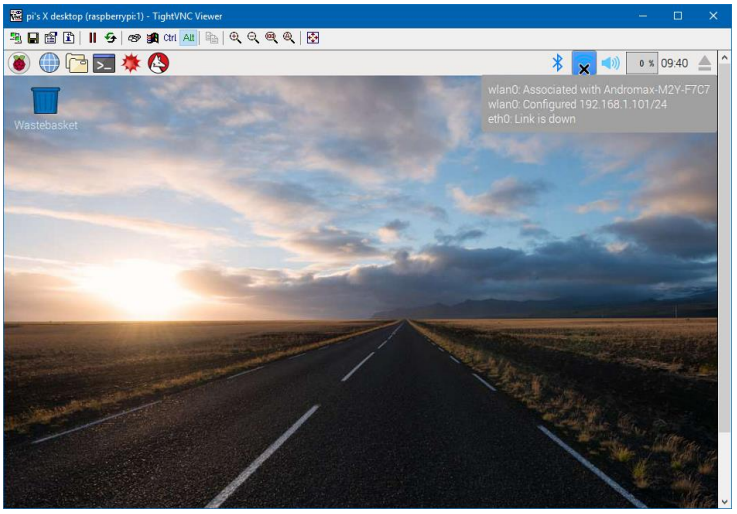
Gambar 2.18 Koneksi Menggunakan IP *Address* Raspberry TightVNC *Server* Desktop

12. Masukan IP *Address* dari Raspberry Pi disertai nomor *Port* yang digunakan lalu tekan tombol *Connect*, sehingga muncul tampilan jendela seperti yang ditunjukan pada Gambar 2.19.



Gambar 2.19 Vnc Authentication

13. Tekan tombol OK setelah memasukan *password* vnc *Server* ketika melakukan instalasi dan konfigurasi vnc *Server* pada Raspberry Pi, maka akan muncul jendela hasil *remote* Raspberry Pi seperti yang ditunjukan pada Gambar 2.20.



Gambar 2.20 Hasil Remote Raspberry Pi

14. Gambar 2.20 merupakan tampilan sistem operasi Raspbian hasil *remote* Raspberry Pi. Hasil *remote* memungkinkan untuk dilakukan semua proses seperti halnya menggunakan Raspberry Pi secara langsung.

2.2 *Sensor Ultrasonik*

Sensor ultrasonik adalah sebuah sensor yang berfungsi untuk mengubah besaran fisis (bunyi) menjadi besaran listrik dan sebaliknya. Cara kerja sensor ultrasonik didasarkan pada prinsip dari pantulan suatu gelombang suara sehingga dapat dipakai untuk menafsirkan eksistensi (jarak) suatu benda dengan frekuensi tertentu. Disebut sebagai sensor ultrasonik karena sensor ini menggunakan gelombang ultrasonik (bunyi ultrasonik). (Sumardi, 2013:113)

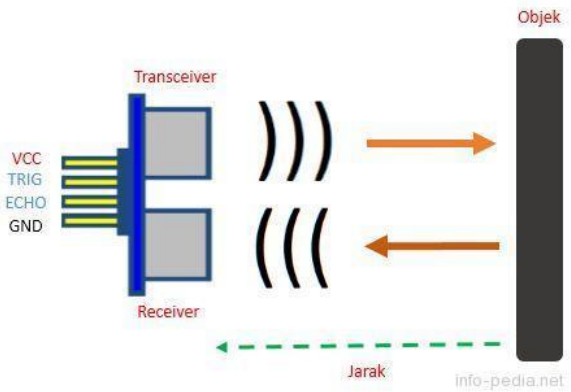
Gambar 2.21 Sensor Ultrasonik
(Sumber: http://arduino-info.wikispaces.com)

Cracknell, A.P, 1980 menjabarkan bahwa gelombang ultrasonik adalah gelombang bunyi yang mempunyai frekuensi sangat tinggi yaitu

20.000 Hz. Bunyi ultrasonik tidak dapat didengar oleh telinga manusia. Bunyi ultrasonik dapat didengar oleh anjing, kucing, kelelawar, dan lumba-lumba. Bunyi ultrasonik dapat merambat melalui zat padat, cair dan gas. Reflektivitas bunyi ultrasonik di permukaan zat padat hampir sama dengan reflektivitas bunyi ultrasonik di permukaan zat cair. Akan tetapi, gelombang bunyi ultrasonik akan diserap oleh tekstil dan busa. Harga sensor ultrasonik di berbagai tempat relatif sama dan cukup murah yaitu Rp. 25.000,00 untuk satu buah sensor.

2.2.1 Cara Kerja Sensor Ultrasonik

Suryono, Kusminarto, Suparta G.B. pada tahun 2010 menguraikan cara kerja sensor ultrasonik dan menyimpulkan gelombang ultrasonik dibangkitkan melalui sebuah alat yang disebut dengan piezoelektrik dengan frekuensi tertentu. Piezoelektrik akan menghasilkan gelombang ultrasonik (umumnya berfrekuensi 40kHz) ketika sebuah *osilator* diterapkan pada benda tersebut. Secara umum, alat ini akan menembakkan gelombang ultrasonik menuju suatu area atau suatu target. Setelah gelombang menyentuh permukaan target, maka target akan memantulkan kembali gelombang tersebut. Gelombang pantulan dari target akan ditangkap oleh sensor, kemudian sensor menghitung selisih antara waktu pengiriman gelombang dan waktu gelombang pantul diterima.



Gambar 2.22 Cara Kerja Sensor Ultrasonik

Secara detail, cara kerja sensor ultrasonik adalah sebagai berikut.

1. Sinyal dipancarkan oleh pemancar ultrasonik dengan frekuensi tertentu dan dengan durasi waktu tertentu. Sinyal tersebut berfrekuensi di atas 20kHz. Untuk mengukur jarak benda (sensor jarak), frekuensi yang umum digunakan adalah 40kHz.
2. Sinyal yang dipancarkan akan merambat sebagai gelombang bunyi dengan kecepatan sekitar 343 m/s. Ketika menumbuk suatu benda, maka sinyal tersebut akan dipantulkan oleh benda tersebut.
3. Setelah gelombang pantulan sampai di alat penerima, maka sinyal tersebut akan diproses untuk menghitung jarak benda tersebut. Jarak benda dihitung berdasarkan rumus S = 343.t/2, dimana S merupakan jarak antara sensor ultrasonik dengan benda (bidang pantul), dan t adalah selisih antara waktu pemancaran gelombang oleh *transmitter* dan waktu ketika gelombang pantul diterima *receiver*.

Secara umum, dapat disimpulkan untuk mendapatkan kecepatan yang ditempuh gelombang ke objek, sensor ultrasonik mengandalkan prinsip dengan rumus berikut.

$$34300 = \frac{Distance}{Time / 2}$$

$$17150 = \frac{Distance}{Time}$$

$$17150 \times Time = Distance$$

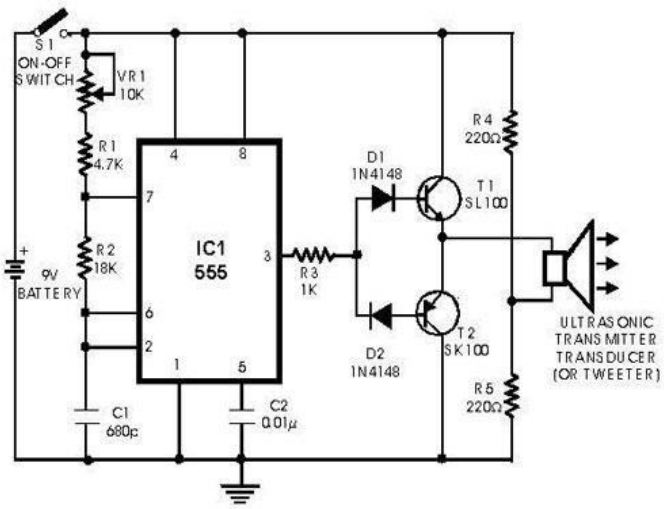
1. Piezoelektrik

Piezoelektrik berfungsi untuk mengubah energi listrik menjadi energi mekanik. Bahan piezoelektrik adalah material yang memproduksi medan listrik ketika dikenai regangan atau tekanan mekanis. Sebaliknya, jika medan listrik diterapkan, maka material tersebut akan mengalami regangan atau tekanan mekanis. Jika rangkaian pengukur beroperasi pada mode pulsa elemen piezoelektrik yang sama, maka dapat digunakan sebagai *transmitter* dan *receiver*. Frekuensi yang ditimbulkan tergantung pada *osilator*-nya yang disesuiakan frekuensi kerja dari masing-masing *transduser*. Keunggulannya inilah maka tranduser piezoelektrik lebih sesuai digunakan untuk sensor ultrasonik.

2. *Transmitter*

Transmitter adalah sebuah alat yang berfungsi sebagai pemancar gelombang ultrasonik dengan frekuensi tertentu (misal, sebesar 40 kHz)

yang dibangkitkan dari sebuah *osilatorp*. Usaha untuk menghasilkan frekuensi 40 KHz, harus dibuat sebuah rangkaian *osilator* dan keluaran dari *osilator* dilanjutkan menuju penguat sinyal. Besarnya frekuensi ditentukan oleh komponen RLC/kristal tergantung dari disain *osilator* yang digunakan.



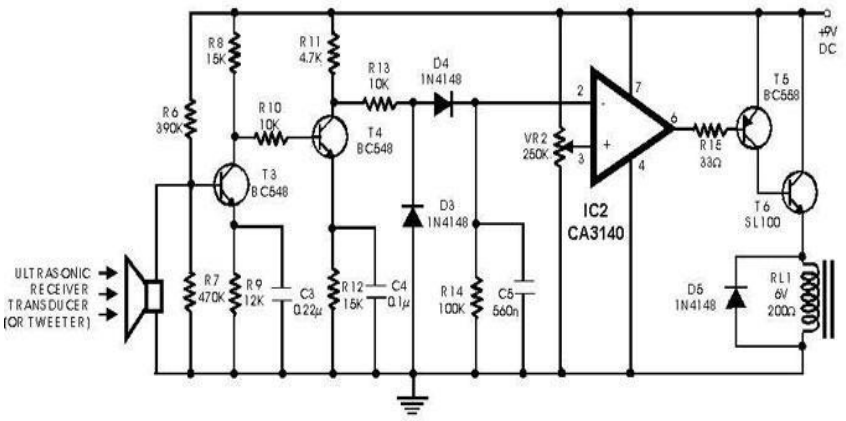
Gambar 2.23 Rangkaian Dasar dari Transmitter Ultrasonik

Penguat sinyal akan memberikan sebuah sinyal listrik yang diumpankan ke piezoelektrik dan terjadi reaksi mekanik sehingga bergetar dan memancarkan gelombang yang sesuai dengan besar frekuensi pada *osilator*.

3. *Receiver*

Receiver terdiri dari *transduser* ultrasonik menggunakan bahan piezoetektrik, yang berfungsi sebagai penerima gelombang pantulan yang berasal dart transmitter yang dikenakan pada permukaan suatu benda atau

gelombang langsung LOS (*Line of Sight*) *dart transmitter*. Oleh karena bahan piezoelektrik memiliki reaksi yang *reversible*, elemen keramik akan membangkitkan tegangan listrik pada saat gelombang datang dengan frekuensi yang resonan dan akan menggetarkan bahan piezoelektrik tersebut.



Gambar 2.24 Rangkaian Dasar dari *Receiver* Sensor Ultrasonik

2.2.2 Aplikasi yang Memanfaatkan Sensor Ultrasonik

Dalam bidang kesehatan, gelombang ultrasonik bisa digunakan untuk melihat organ-organ dalam tubuh manusia seperti untuk mendeteksi tumor, liver, otak dan menghancurkan batu ginjal. Gelombang ultrasonik juga dimanfaatkan pada alat USG (*ultrasonografi*) yang biasa digunakan oleh dokter kandungan.

Dalam bidang industri, gelombang ultrasonik digunakan untuk mendeteksi keretakan pada logam, meratakan campuran besi dan timah, meratakan campuran susu agar homogen, mensterilkan makanan yang

diawetkan dalam kaleng, dan membersihkan benda benda yang sangat halus. Gelombang ultrasonik juga digunakan untuk mendeteksi keberadaan mineral maupun minyak bumi yang tersimpan di dalam perut bumi.

Dalam bidang pertahanan, gelombang ultrasonik digunakan sebagai radar atau navigasi, di darat maupun di dalam air. Gelombang ultrasonik digunakan oleh kapal pemburu untuk mengetahui keberadaan kapal selam, dipasang pada kapal selam untuk mengetahui keberadaan kapal yang berada di atas permukaan air, mengukur kedalaman palung laut, mendeteksi ranjau, dan menentukan posisi sekelompok ikan.

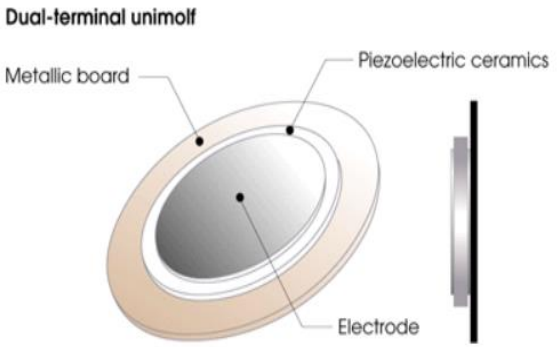
2.3 *Sensor Getaran (Vibration)*

Sensor getaran merupakan salah satu sensor yang dapat mengukur getaran suatu benda yang nantinya dimana data tersebut akan diproses untuk kepentingan percobaan ataupun digunakan untuk mengantisipasi sebuah kemungkinan adanya bahaya. Sensor getaran memiliki beberapa jenis tertentu, adapun jenis-jenis sensor getaran, yaitu sebagai berikut.

1. *Ceramic Piezoelectric Sensor or* Accelerometer

Pengukuran Getaran atau percepatan paling sering meggunakan sensor piezoelektrik keramik atau accelerometer. Tiga faktor utama membedakan sensor getaran yaitu frekuensi natural, koefisien redaman, dan faktor skala. Faktor skala berhubungan *output* ke *input* akselerasi dan terkait dengan sensitivitas. Paramter frekuensi natural dan koefisien redaman menentukan tingkat akurasi dari sensor getaran. Suatu sistem yang terdiri dari pegas dan *massa* terpasang, jika ditarik *massa* kembali menjauh dari keseimbangan dan melepaskannya, *massa* akan bergetar maju (masa

keseimbangan) dan mundur hingga diam. Gesekan yang membawa *massa* untuk beristirahat didefinisikan oleh koefisien redaman, dan tingkat di mana massa bergetar maju dan mundur adalah frekuensi natural. (Rafiuddin Syam, 2013)



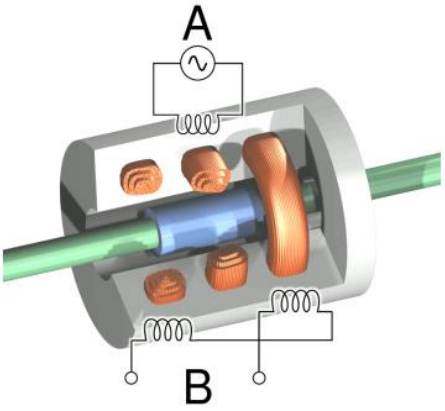
Gambar 2.25 Strukur Sensor Keramik Piezoelectric
(Sumber: Rafiuddin Syam, 2013)

Sensor getaran piezoelektrik keramik adalah sensor yang paling umum digunakan karena jenis ini paling serbaguna. Sensor getaran ini dapat digunakan dalam pengukuran syok (ledakan dan tes gagal), pengukuran frekuensi tinggi, dan lambat frekuensi rendah pengukuran getaran. Hal ini ditunjukkan oleh sensor piezoelektrik keramik yang lebih tinggi daripada frekuensi natural rata-rata. Sensor ini memiliki *output* di kisaran *millivolt* dan membutuhkan *high-input*-impedansi, detektor suara rendah untuk menafsirkan tegangan dari kristal piezoelektriknya. (Rafiuddin Syam, 2013)

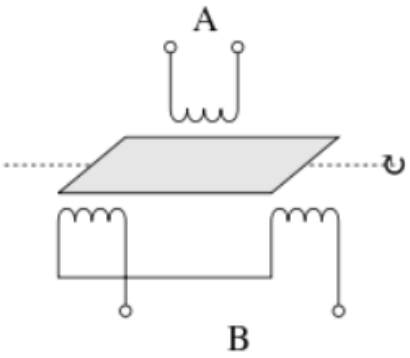
2. *Proximity Probes and Linear Variable Differential Transformers*

Proximity Probes dan *Linear Variable Differential Transformers* (LVDTs) adalah dua sensor yang serupa. Keduanya terbatas percepatan

mapan atau pengukuran getaran frekuensi rendah. Sensor LVDT getaran memiliki frekuensi alami sedikit lebih tinggi, yang berarti bahwa ia dapat menangani/mendeteksi getaran yang agak tinggi. *Proximity Probe* hanyalah sebuah pegas massa yang melekat pada wiper dari potensiometer. (Rafiuddin Syam, 2013)



Gambar 2.26 Struktur Sensor LVDT
(Sumber: Rafiuddin Syam, 2013)



Gambar 2.27 Prinsip Kerja Sensor LVDT
(Sumber: Rafiuddin Syam, 2013)

Sebuah sensor getaran variabel *Reluctance* menggunakan magnet permanen dan gerakan melalui kumparan untuk mengukur gerakan dan getaran. Ini adalah sensor getaran khusus karena *output* keluaran hanya ketika massa itu mengukur adalah dalam gerakan. Hal ini membuatnya sangat berguna dalam studi goncangan gempa dan eksplorasi minyak untuk mengambil getaran tercermin dari *strata rock underground*. (Rafiuddin Syam, 2013)

Gambar 2.28 Jenis Sensor Variable Reluctance
(Sumber: Rafiuddin Syam, 2013)

2.4 *Sensor Suhu dan Sensor Kelembapan*

Sub bab berikut ini akan menjelaskan detail dari sensor suhu dan sensor kelembapan.

2.4.1 Sensor Suhu

Suhu adalah besaran numerik untuk mengetahui derajat panas atau dingin pada suatu benda. Suhu juga dapat didefinisikan sebagai suatu besaran termodinamika yang menunjukkan besarnya energi kinetik translasi rata-rata molekul dalam sistem gas. Adapun beberpa jenis sensor suhu, yaitu sebagai berikut.

1. Termokopel (*Thermocouple*)

Termokopel hingga saat ini merupakan sensor suhu paling populer, efektif dalam aplikasi yang memerlukan kisaran suhu yang besar. Sensor ini dikenal murah, harganya mulai dari $ 1 sampai $ 50 USD, dan memiliki respon waktu sepersekian detik. Karena sifat material dan faktor lainnya, suhu akurasi kurang dari 1° C. (Rafiuddin Syam, 2013)

2. Sensor RTDs

Sensor RTDs hampir sepopuler termokopel dan dapat mempertahankan untuk membaca suhu stabil selama bertahun-tahun. Berbeda dengan termokopel, RTDs memiliki rentang suhu yang lebih kecil yaitu antara -200 hingga 500° C, memerlukan arus eksitasi, dan memiliki waktu respon yang lebih lambat yaitu sekitar 2,5-10 s. Sensor RTDs adalah terutama digunakan untuk pengukuran suhu yang akurat (± 1,9 persen) dalam aplikasi yang tidak waktu kritis. Harga sensor RTDs dapat diperoleh dengan biaya antara $ 25 dan $ 1.000 USD. (Rafiuddin Syam, 2013)

3. Termistor (*Thermistors*)

Termistor memiliki rentang suhu yang lebih kecil (-90 sampai 130° C) dari sensor yang disebutkan sebelumnya (Termokopel dan RTDs). Sensor termistor memiliki akurasi terbaik (± 05° C), tetapi lebih rapuh, mudah rusak dari termokopel atau RTDs. Termistor memerlukan eksitasi seperti RTD; namun, termistor membutuhkan tegangan eksitasi daripada arus eksitasi. Sebuah termistor biasanya dijual dengan harga sekitar antara $ 2 dan $ 10 USD. (Rafiuddin Syam, 2013)

4. *Fiber Optics*

Alternatif lain adalah penggunaan serat optik untuk mengukur suhu. Sensor Suhu serat optik efektif untuk lingkungan yang berbahaya atau di mana mungkin ada interferensi elektromagnetik.

Sensor serat *optic* adalah *nonconductive*, elektrik pasif, kebal terhadap interferensi elektromagnetik (EMI)-induksi karena kebisingan, dan mampu mengirimkan data lebih panjang jarak dengan sedikit atau tanpa kehilangan integritas sinyalnya. (Rafiuddin Syam, 2013)

Sensor Temperatur	Signal Conditioning Required	Akurasi	Sensitivity	Comparison
Thermocouple	■■ Amplification ■■ Filtering ■■ Cold-Junction Compensation	Baik	Baik	■■ Self-Powered ■■ Inexpensive ■■ Rugged ■■ Large Temperature Range
RTD	■■ Amplification ■■ Filtering ■■ Current Excitation	Sangat baik	Lebih baik	■■ Sangat akurat ■■ Sangat stabil
Thermistor	■■ Amplification ■■ Filtering ■■ Voltage Excitation	Lebih baik	Sangat baik	■■ High Resistance ■■ Low Thermal Mass
Fiber Optics	■■ Little or No Amplification ■■ Filtering	Sangat baik	Sangat baik	■■ Good for Hazardous Environments ■■ Good for Long Distances ■■ Immune to Electromagnetic Interference (EMI)-Induced Noise ■■ Small, Lightweight

Gambar 2.29 Perbandingan Sensor Suhu
(Sumber: Rafiuddin Syam, 2013)

2.4.2 Sensor Kelembapan

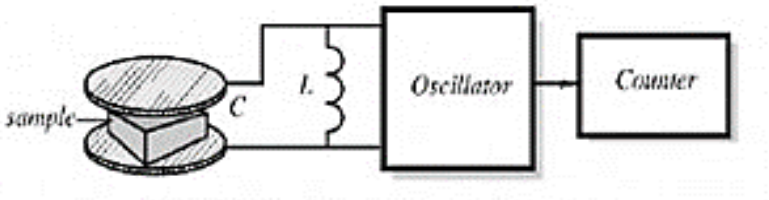
Kelembapan adalah ukuran jumlah uap air di udara, jumlah uap air mempengaruhi proses-proses fisika, kimia dan biologi di alam, oleh karena itu akan mempengaruhi lingkungan. Kandungan uap air melebihi atau kurang

dari kebutuhan yang diperlukan maka akan menimbulkan gangguan atau kerusakan. Alat ukur kelembapan yang telah dikembangkan, salah satu yang biasa digunakan adalah alat untuk mengukur kelembapan udara yang disebut higrometer. Kebutuhan akan kecepatan, keakuratan dan ketelitian hasil pengukuran yang lebih tinggi maka diperlukan pengembangan alat ukur baru. Oleh karena itu dikembangkan sensor kelembapandengan kekurangan dan kelebihannya masing-masing.

Sensor kelembapan adalah suatu alat ukur yang digunakan untuk membantu dalam proses pengukuran atau pendefinisian yang suatu kelembapan uap air yang terkandung dalam udara. Jenis-jenis sensor kelembapan diantaranya *Capacitive Sensors*, *Electrical conductivity Sensors*, *Thermal Conductivity Sensors*, *Optical Hygrometer*, dan *Oscillating Hygrometer*.

1. *Capacitive Sensors*

Sebuah kapasitor air-filled/terisi-udara dibuat sebagai suatu sensor kelembapan relative karena uap dalam atmosfer merubah permivitas elektrik udara. (Dirgantara Made, 2012)



Gambar 2.30 *Capacitive Sensors*
(Sumber: Dirgantara Made, 2012)

Prinsip kerjanya adalah dengan memanfaatkan perubahan kapasitif dimana dengan adanya perubahan posisi bahan dielektrik diantara kedua

keeping, pergeseran posisi salah satu keeping dan luas kepingnya berhadapan langsung, atau dengan perubahan jarak antara kedua keeping. Salah satu contoh sensor kelembapan capasitive adalah sensor Relative Humidity HS-15P. (Dirgantara Made, 2012)

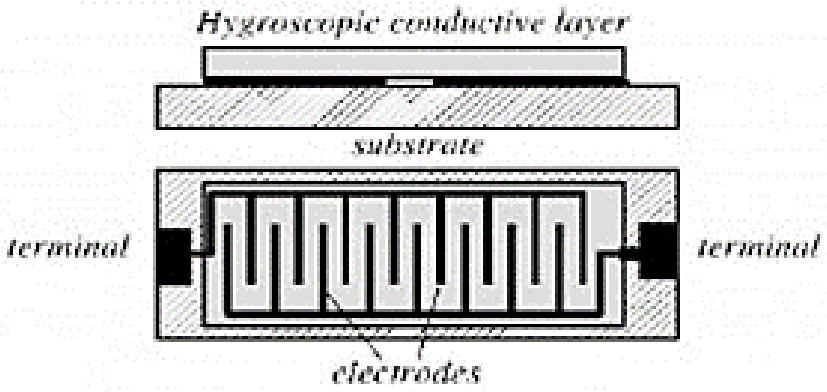
Aplikasi sensor ini dalam bidang industri adalah sebagai sistem pengendalian suhu dan kelembapan serta untuk mesin pengering kertas. Pada prinsipnya cara kerja sensor ini adalah dengan mendeteksi besarnya kelembapan relatif udara di sekitar sensor tersebut. Sensor HS15P yang mendeteksi kelembapan udaran disekitarnya akan merubah frekuensi osilator dan akan mengirimkan data ke mikrokontroler slave. *Mikro slave* akan dilanjutkan ke mikro master, kemudian selanjutnya mikro ini akan menganalisis data dengan cara membandingkan data yang dikirim dan data masukan. Kelembapan yang ditentukan dibawah atau diatas dari data yang dikirim sensor maka alat akan bekerja untuk menyesuaikan kelembapan menjdi sesuai dengan yang diharapkan. (Dirgantara Made, 2012)

Karakteristik sensor HS15P adalah bekerja pada rating temperature 0°C-50°C, bekerja pada rating kelembapan 20%-100% RH, tegangan kerja adalah tegangan AC 1 Vrms, frekuensi kerjanya 50 Hz-1 KHz, konsumsi dayanya sebesar 0,3 mW, dengan perubahan temperature dengan kenaikan 5°C maka kurva karakteristik Relative Humidity akan bergeser berbanding terbalik dengan perubahan impedansi. (Dirgantara Made, 2012)

2. *Electrical Conductivity Sensors*

Sensor kelembapan konduktivitas adalah disebut dengan "*Pope element*", yang terdiri dari *polystyrene* yang dilakukan/diperlakukan

dengan asam sulfur untuk memperoleh karakteristik surface-resistivitas yang diinginkan. (Dirgantara Made, 2012)



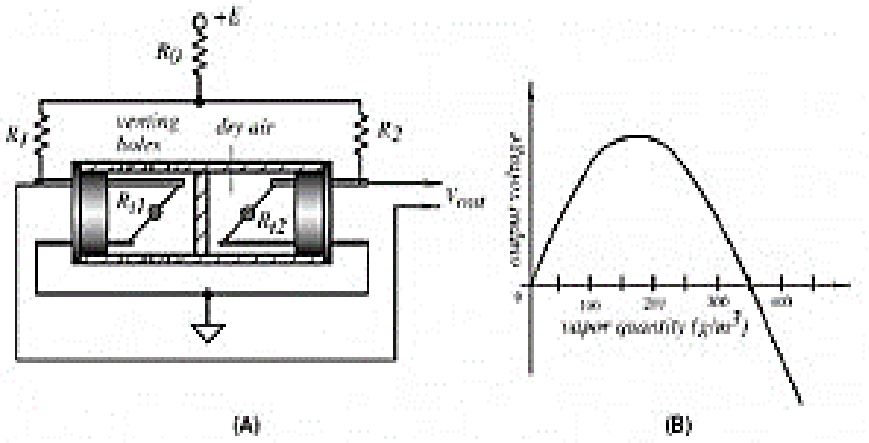
Gambar 2.31 *Electrical Conductivity Sensors*
(Sumber: Dirgantara Made, 2012)

Contoh sensor *conductivity* adalah sensor ABS-300. Sensor ABS-300 dapat diaplikasikan dalam bidang industry sebagai indikator pengering pada mesin cuci. Prinsip kerjanya adalah sensor ini terdiri dari film tipis polimer oksida logam antara 2 elektroda konduktif, dimana permukaan penginderaan/sensornya dilapisi dengan logam berpori elektroda dari bahan kaca/keramik/silicon untuk melindungi dari kontaminasi, sehingga perubahan dalam konstanta dielektrik sensor kelembaman kapasitif hamper berbanding lurus dengan kelembapan relatif lingkungan sekitarnya. (Dirgantara Made, 2012)

Karakteristik sensor ini antara lain memiliki perubahan kapasitansi o.2-o.5 pF untuk RH 1%, kapasitansinya antara 100-500 pF sebesar 50% RH pada 25°C, dan rentang waktu responnya antara 30-60 untuk perubahan RH 63%. (Dirgantara Made, 2012)

3. *Thermal Conductivity Sensor*

Penggunaan konduktivitas thermal dari gas untuk mengukur kelembapan dapat di ukur oleh sebuah sensor thermistor. (Dirgantara Made, 2012)



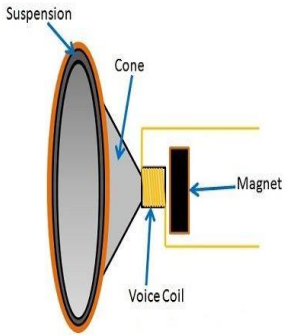
Gambar 2.32 *Thermal Conductivity Sensor*
(Sumber: Dirgantara Made, 2012)

Contoh sensor ini adalah sensor TCG-3880. Sensor TCG-3880 dapat diaplikasikan sebagai vacum sensor pada industri misalnya sebagai mesin pengering. Prinsip kerjanya yakni terdiri dari 2 ruang masing-masing dengan sebuah sensor identik konduktivitas termal, satu ruang yang ditutup dan diisi dengan gas referensi sedangkan yang lainnya menerima gas sampel, perbedaan konduktivitas termal dari gas referensi akan diterjemahkan ke dalam angka konsentrasi oleh sirkuit mikroprosesor dalam unit elektronik. (Dirgantara Made, 2012)

2.5 *Speaker*

Transduser yang dapat mengubah sinyal listrik menjadi Frekuensi Audio (sinyal suara) yang dapat didengar oleh telinga manusia dengan cara mengetarkan komponen membran pada Speaker tersebut sehingga terjadilah gelombang suara. (Kho, 2014)

Seperti pada Gambar 2.33, dapat dilihat bahwa pada dasarnya *Speaker* terdiri dari beberapa komponen utama yaitu *Cone*, *Suspension*, *Magnet Permanen*, *Voice Coil* dan juga Kerangka *Speaker*. Dalam rangka menterjemahkan sinyal listrik menjadi suara yang dapat didengar, *Speaker* memiliki komponen Elektromagnetik yang terdiri dari Kumparan yang disebut dengan *Voice Coil* untuk membangkitkan medan magnet dan berinteraksi dengan Magnet Permanen sehingga menggerakan *Cone Speaker* maju dan mundur.



Gambar 2.33 Speaker
(Sumber: teknikelektronika.com)

Voice Coil adalah bagian yang bergerak sedangkan Magnet Permanen adalah bagian *Speaker* yang tetap pada posisinya. Sinyal listrik yang melewati *Voice Coil* akan menyebabkan arah medan magnet berubah secara

cepat sehingga terjadi gerakan "tarik" dan "tolak" dengan Magnet Permanen. Dengan demikian, terjadilah getaran yang maju dan mundur pada *Cone Speaker*.

Cone adalah komponen utama *Speaker* yang bergerak. Pada prinsipnya, semakin besarnya *Cone* semakin besar pula permukaan yang dapat menggerakan udara sehingga suara yang dihasilkan *Speaker* juga akan semakin besar.

Suspension yang terdapat dalam *Speaker* berfungsi untuk menarik *Cone* ke posisi semulanya setelah bergerak maju dan mundur. Suspension juga berfungsi sebagai pemegang *Cone* dan *Voice Coil*. Kekakuan (*rigidity*), komposisi dan desain *Suspension* sangat mempengaruhi kualitas suara *Speaker* itu sendiri.

Speaker bisa didapatkan dengan harga beragam, mulai dari Rp. 6.000,00, Rp. 45.000,00, hingga lebih tergantung jenis.

2.6 *Webcam*

Webcam (singkatan dari *web camera*) adalah sebutan bagi kamera *real-time* (bermakna keadaan pada saat ini juga) yang gambarnya bisa diakses atau dilihat melalui *World Wide Web*, program *instant messaging*, atau aplikasi video call. (Darma Putra, 2016)

Gambar 2.34 Kamera
(Sumber: http://techallianz.info)

Webcam adalah sebuah kamera video digital kecil yang dihubungkan ke komputer melalui port USB ataupun port COM dan hingga sekarang webcam sudah lebih maju dan tertanam lansung dilaptop tanpa menggunakan port USB. (Julfikar, 2016)

2.7 *Button*

Tombol *button* merupakan tombol yang digunakan untuk mengirimkan pesan ketika pengendara merasa perlu mengirimkan kondisi pengendara saat itu. Tombol *button* bisa didapatkan dengan harga beragam, mulai dari Rp. 15.000,00.

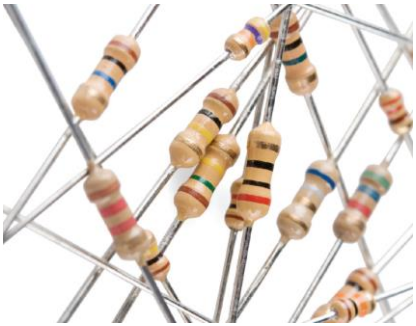
Gambar 2.35 *Button*

2.8 *Kabel Jumper*

Gambar 2.36 Kabel *Jumper*
(Sumber: tokopedia.com)

Kabel *jumper* (atau yang dikenal sebagai kawat *jumper*, kabel *jumper*, kawat DuPont, kabel DuPont) adalah kabel yang berdiameter kecil untuk menghubungkan dua titik atau lebih rangkaian komponen-komponen elektronika. Kabel *jumper* sangat dibutuhkan untuk menyusun sebuah sirkuit yang menggunakan bread board sehingga tidak perlu men-solder suatu perangkat. Kabel *jumper* bisa didapatkan mulai dari harga Rp. 9.000,00.

2.9 *Resistor*



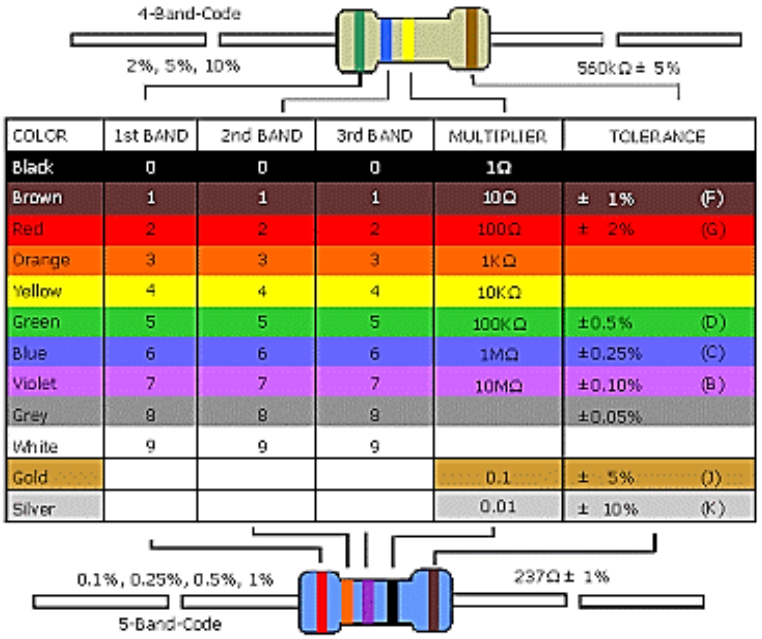
Gambar 2.37 *Resistor*
(Sumber: learn.sparkfun.com)

Resistor adalah komponen elektronik yang memiliki spesifikasi tertentu. *Resistor* juga memiliki resistensi listrik yang tak berubah atau

bersifat tetap. Resistor membatasi aliran listrik sesuai dengan resistensi bawaannya dan dilakukan melalui sirkuit. *Resistor* merupakan komponen pasif, yang berarti hanya mengkonsumsi daya (tidak dapat menghasilkan daya). *Resistor* biasanya ditambahkan ke sirkuit di mana mereka melengkapi komponen aktif seperti *op-amp*, *mikrokontroler*, dan sirkuit terpadu lainnya. Umumnya resistor digunakan untuk membatasi arus, membagi tegangan, dan *pull-up* baris I/O. Harga satu buah resistor cukup murah kurang lebih yaitu Rp. 200,00.

Cara kerja resistor yaitu dari hambatan listrik dari sebuah resistor diukur dalam satuan ohm. Simbol untuk ohm adalah Yunani modal omega: Ω. Definisi 1Ω adalah perlawanan antara dua titik di mana 1 volt (1V) energi potensial diterapkan akan mendorong 1 Ampere (1A) dari saat ini.

Nilai-nilai yang lebih besar atau lebih kecil dari ohm dapat dicocokkan dengan awalan seperti kilo-, mega, atau giga-, membuat nilai-nilai yang besar mudah dibaca. Hal ini sangat umum untuk melihat resistor di kilohm (kW) dan megaohm (MQ) Kisaran (jauh kurang umum untuk melihat miliohm (MQ) resistor).



COLOR	1st BAND	2nd BAND	3rd BAND	MULTIPLIER	TOLERANCE	
Black	0	0	0	1Ω		
Brown	1	1	1	10Ω	± 1%	(F)
Red	2	2	2	100Ω	± 2%	(G)
Orange	3	3	3	1KΩ		
Yellow	4	4	4	10KΩ		
Green	5	5	5	100KΩ	±0.5%	(D)
Blue	6	6	6	1MΩ	±0.25%	(C)
Violet	7	7	7	10MΩ	±0.10%	(B)
Grey	8	8	8		±0.05%	
White	9	9	9			
Gold				0.1	± 5%	(J)
Silver				0.01	± 10%	(K)

Gambar 2.38 Representasi Warna pada *Resistor*

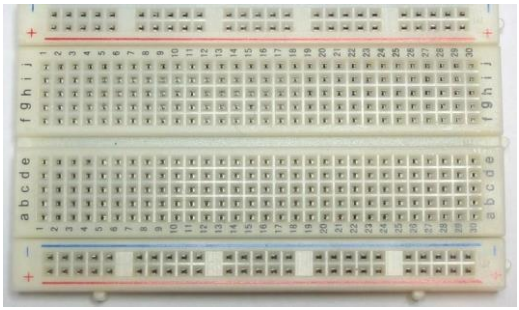
Sebagai contoh, resistor 10000 akan dinamakan 1kΩ, sebuah resistor 4,7000Ω setara dengan resistor 4.7kΩ, dan resistor 5,600,0000Ω dapat ditulis sebagai 5,600kΩ atau (lebih umum sebagai) 5.6MΩ. Gambar 2.18 merupakan tabel dari masing-masing warna yang mewakili nilai, multiplier atau toleransi resistor.

2.10 *Breadboard*

Menciptakan sebuah rangkaian pada elektronika itu membutuhkan waktu yang cukup lama, mulai dari analisis, pembuatan skema, pembuatan prototipe, dan berbagai proses lainnya. Semua tahapan proses memiliki cerita dan catatan masing-masing. Diantara proses pembuatan rangkaian tersebut ada proses yang cukup menarik yaitu proses pembuatan Prototipe.

Proses ini merupakan proses awal untuk menganalisis kinerja dari rangkaian dan proses pencarian kekurangan serta kesalahan rangkaian. Pembuatan Prototipe membutuhkan beberapa perangkat wajib yang harus tersedia. Salah satu komponen yang wajib ada adalah bread board. *Breadboard* adalah papan khusus yang digunakan untuk membuat Prototipe atau rangkaian elektronik yang bersifat percobaan.

Project Board atau yang sering disebut sebagai *Breadboard* adalah dasar konstruksi sebuah sirkuit elektronik dan merupakan prototipe dari suatu rangkaian elektronik. Di jaman modern istilah ini sering digunakan untuk merujuk pada jenis tertentu dari papan tempat merangkai komponen, dimana papan ini tidak memerlukan proses men-solder (langsung tancap). Karena papan ini *solderless*, papan Paket *Breadboard* dapat digunakan kembali, dan dengan demikian dapat digunakan untuk prototipe sementara serta membantu dalam bereksperimen desain sirkuit elektronika. *Breadbroad* adalah papan berlubang yang biasa digunakan untuk membuat simulasi terlebih dahulu sebelum membuat rangkaian yang sesungguhnya. *Breadboard* memiliki beberapa ukuran dari yang kecil, sedang, dan besar. Pada dasarnya ketiga papan tersebut konsepnya sama, hanya berbeda ukurannya saja. Paket *Breadboard* memiliki banyak jenisnya, mulai dari Paket *Breadboard* dengan jumlah terminal yang sedikit, hingga Paket *Breadboard* dengan terminal banyak. Harga untuk 1 Paket *Breadboard* adalah Rp.100.000,00.



Gambar 2.39 Paket *Breadboard*

Terminal pada *breadboard* yaitu untuk berbagai sistem elektronik dapat di prototipe-kan dengan menggunakan *Breadboard*, mulai dari sirkuit analog dan digital kecil sampai membuat unit pengolahan terpusat (CPU).

Dua pasang jalur atas dan bawah terhubung secara horisontal sampai ke bagian tengah dari *Breadboard*. Biasanya jalur tersebut digunakan sebagai jalur *power* atau jalur sinyal yang umum digunakan seperti clock atau jalur komunikasi. Lima lubang komponen di tengah merupakan tempat merangkai komponen. Jalur ke 5 lubang ini terhubung vertikal sampai bagian tengah dari *Breadboard*. Sedangkan pembatas tengah *Breadboard* biasanya digunakan sebagai tempat menancapkan komponen *Integrated Circuit* (IC).

2.11 *Adaptor*

Adaptor adalah sebuah perangkat berupa rangkaian elektronika untuk mengubah tegangan listrik yang besar menjadi tegangan listrik lebih kecil, atau rangkaian untuk mengubah arus bolak-balik (AC) menjadi arus searah (DC). *Adaptor* adalah perangkat elektronik yang dapat merubah tegangan listrik yang tinggi menjadi tegangan listrik yang rendah, tetapi ada juga adaptor yang dapat merubah tegangan listrik yang rendah menjadi tegangan listrik yang tinggi. Adaptor merupakan salah satu contoh penyuplai

daya (*power supply*). Salah satu keuntungan dari *adaptor* adalah sangat praktis berhubungan dengan ketersediaan tegangan, karena *adaptor* dapat diambil dari sumber tegangan AC yang ada di rumah dan mempunyai jangka waktu yang tidak terbatas.

Secara umum *adaptor* adalah alat elektronika yang dapat menyesuaikan atau merubah tegangan listrik, maksudnya adalah merubah sumber tegangan listrik utama yaitu dari PLN menjadi tegangan listrik yang dapat digunakan untuk disesuaikan dengan perangkat elektronik yang akan dipakai, misalnya seperti televisi, radio, *gadget* dan lainnya. Adaptor bisa didapat dari kisaran Rp. 60.000,00 hingga Rp. 100.000,00.

Gambar 2.40 *Adaptor*

2.12 *Global Positioning System (GPS)*

Global Positioning System (GPS) merupakan suatu sistem navigasi atau penentu posisi berbasis satelit. Sistem ini didesain untuk memberikan posisi dan informasi mengenai waktu, secara kontinyu di seluruh dunia tanpa tergantung waktu dan cuaca. Penentuan posisi GPS digambarkan dengan

menggunakan nilai koordinat X dan Y atau garis bujur dan garis lintang. Sistem ini digunakan untuk menentukan posisi pada permukaan bumi dengan bantuan sinkronisasi sinyal satelit, menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal ini diterima oleh alat penerima yang ada di bumi, dan digunakan untuk menentukan posisi, kecepatan, arah, dan waktu (Okky Pratiwi Suherman, 2014).

Data dikirim dari satelit berupa sinyal radio dengan data digital. Dimanapun posisi saat ini, maka GPS bisa membantu menunjukan arah, selama masih terlihat langit. Sistem GPS, yang nama aslinya adalah NAVSTAR GPS (*Navigation Satellite Timing and Ranging Global Positioning System*), mempunyai tiga segmen yaitu: satelit, pengontrol, dan penerima/pengguna. Satelit GPS yang mengorbit bumi, dengan orbit dan kedudukan yang tetap (koordinatnya pasti), seluruhnya berjumlah 24 buah dimana 21 buah aktip bekerja dan 3 buah sisanya adalah cadangan.

2.12.1 Cara Kerja GPS

Setiap daerah di atas permukaan bumi ini minimal terjangkau oleh 3-4 satelit. Pada prakteknya, setiap GPS terbaru bisa menerima sampai dengan 12 channel satelit sekaligus. Kondisi langit yang cerah dan bebas dari halangan membuat GPS dapat dengan mudah menangkap sinyal yang dikirimkan oleh satelit. Semakin banyak satelit yang diterima oleh GPS, maka akurasi yang diberikan juga akan semakin tinggi.

Bagian yang paling penting dalam sistem navigasi GPS adalah beberapa satelit yang berada di orbit bumi atau yang sering sebut di ruang angkasa. Satelit GPS saat ini berjumlah 24 unit yang semuanya dapat memancarkan sinyal ke bumi yang lalu dapat ditangkap oleh alat penerima

sinyal tersebut atau GPS *Tracker*. Selain satelit terdapat 2 sistem lain yang saling berhubungan, sehingga jadilah 3 bagian penting dalam sistem GPS. Ketiga bagian tersebut terdiri dari: GPS *Control Segment* (Bagian Kontrol), GPS *Space Segment* (bagian angkasa), dan GPS *User Segment* (bagian pengguna).

1. GPS *Control Segment*

Control segment GPS terdiri dari lima stasiun yang berada di pangkalan Falcon Air Force, Colorado Springs, Ascension Island, Hawaii, Diego Garcia dan Kwajalein. Kelima stasiun ini adalah mata dan telinga bagi GPS. Sinyal-sinyal dari satelit diterima oleh bagian kontrol, kemudian dikoreksi, dan dikirimkan kembali ke satelit. Data koreksi lokasi yang tepat dari satelit ini disebut data ephemeris, yang kemudian nantinya dikirimkan ke alat navigasi yang kita miliki.

2. GPS *Space Segment*

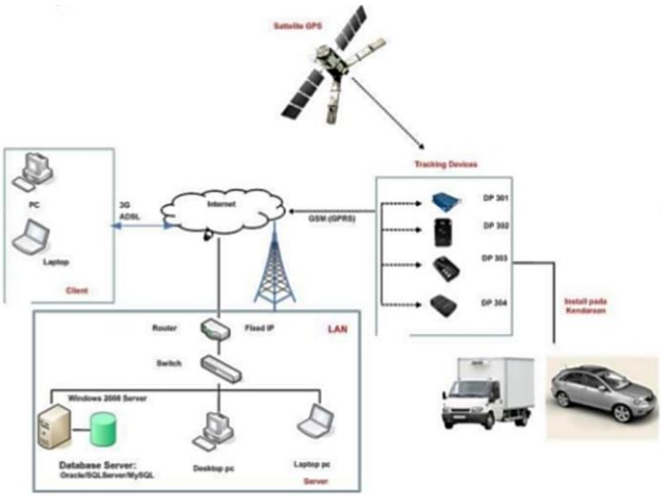
Space Segment adalah terdiri dari sebuah jaringan satelit yang tediri dari beberapa satelit yang berada pada orbit lingkaran yang terdekat dengan tinggi nominal sekitar 20.183 km di atas permukaan bumi. Sinyal yang dipancarkan oleh seluruh satelit tersebut dapat menembus awan, plastik dan kaca, namun tidak bisa menembus benda padat seperti tembok dan rapatnya pepohonan. Terdapat 2 jenis gelombang yang hingga saat ini digunakan sebagai alat navigasi berbasis satelit. Masing-masingnya adalah gelombang L1 dan L2, dimana L1 berjalan pada frequensi 1575.42 MHz yang bisa digunakan oleh masyarakat umum, dan L2 berjalan pada frequensi 1227.6 Mhz dimana jenis ini hanya untuk kebutuhan militer saja.

3. GPS *User Segment*

User segment terdiri dari antena dan *prosesor receiver* yang menyediakan positioning, kecepatan dan ketepatan waktu ke pengguna. Bagian ini menerima data dari satelit-satelit melalui sinyal radio yang dikirimkan setelah mengalami koreksi oleh stasiun pengendali (GPS Control Segment).

2.12.2 *GPS Tracker*

GPS *Tracking* adalah teknologi AVL yang memungkinkan pengguna untuk melacak posisi kendaraan, armada ataupun mobil dalam keadaan Real-Time. GPS Tracking memanfaatkan kombinasi teknologi GSM dan GPS untuk menentukan koordinat sebuah objek, lalu menerjemahkannya dalam bentuk peta digital (Okky Pratiwi Suherman, 2014).



Gambar 2.41 Arsitektur GPS *Tracker*
(https://id.wikipedia.org)

Cara Kerja GPS *Tracker* sebagai berikut GPS *Tracking Device* menerima sinyal GPS dari beberapa satelit GPS. Berdasarkan sinyal-sinyal tersebut, GPS *Tracking* menghitung posisinya. GPS *Tracking* mengirim data posisinya tersebut serta secara *online* ke *Tracking Server. Tracking System* melalui jaringan GSM. Pemakai dapat menggunakan *web browser* untuk melakukan pelacakan dan pemantauan kendaraan yang akan ditampilkan pada peta digital melalui jaringan internet. *Web Browser* dibuatkan aplikasi *Tracking* untuk pelacakan secara *Real Time* (nyata) dengan mengkonfigurasikan waktu data *upload* dari satelit, dan memungkingkan diperlukan adanya paket data (di Indonesia disebut paket *internet*) terhadap *provider* GSM dengan kartu SIM yang dimasukkan di dalam perangkat GPS *Tracker*-nya. Pembacaan data *online* dari satelit ke jaringan GSM berbentuk angka-angka koordinat yang bisa dibaca posisi/peta lokasinya melalui aplikasi mapping (umumnya Google Maps).

2.13 *Android*

Menurut buku nazrudin safaat dalam bukunya yang berjudul ANDROID Pemprograman Aplikasi Mobile Smartphone dan Table PC Berbasis Android: "Android adalah sistem operasi untuk perangkat mobile berbasis linux yang mencakup system operasi, *middleware* dan aplikasi (Nazrudin safaat, 2011: 1)". Android dikembangkan oleh *Google* bersama *Open Handset Allience* (OHA) yaitu aliansi perangkat selular terbuka yang terdiri dari 47 perusahaan Hardware, Software dan perusahaan telekomunikasi ditujukan untuk mengembangkan standar terbuka bagi perangkat selular.

2.13.1 Sejarah Android

Pada mulanya terdapat berbagai macam sistem operasi pada perangkat selular, iantaranya sistem operasi *Symbian*, *Microsoft Windsos Mobile, Mobile Linux, iPhone,* dan sistem operasi lainnya. Namun diantara sistem operasi yang ada belum mendukung standar dan penerbitan API yang dapat dimanfaatkan secara keseluruhan dan dengan biaya yang murah. Kemudian *Google* ikut berkecimpung didalamnya dengan platform Android, yang menjanjikan keterbukaan, keterjangkauan, open source, dan *framework* berkualitas.

Pada tahun 2005, Google mengakuisisi perusahaan Android Inc. untuk memulai pengembangan platform Android. Dimana terlibat dalam pengembangan ini Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Pada pertengahan 2007 sekelompok pemimpin industri bersama-sama membentuk aliansi perangkatselular terbuka, *Open Handset Alliance* (OHA). Bagian dari tujuan aliansi ini adalah berinovasi dengan cepat dan menanggapi kebutuhan konsumen dengan lebih baik, dengan produk awalnya adalah platform Android. Dimana Android dirancang untuk melayani kebutuhan operator telekomunikasi, manufaktur handset, dan pengembang aplikasi. OHA berkomitmen untuk membuat Android open source dengan lisensi Apache versi II.0.

Android pertama kali diluncurkan pada 5 November 2007, dan *smartphone* pertama yang menggunakan sistem operasi Android dikeluarkan oleh T-Mobile dengan sebutan G1 pada bulan September 2008. Hingga saat ini Android telah merilis beberapa versi Android untuk menyempurnakan versi sebelumnya. Selain berdasarkan penomoran, pada setiap versi Android

terdapat kode nama berdasarkan nama-nama kue. Hingga saat ini sudah terdapat beberapa versi yang telah diluncurkan, diantaranya: versi 1.5 dirilis pada 30 April 2009 diberi nama *Cupcake*, versi 1.6 dirilis pada 15 September 2009 diberi nama *Donut*, dan versi terakhir II.0 dirilis pada 26 Oktober 2009 diberi nama *Eclair*.

2.13.2 Arsitektur Android

Secara garis besar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut.

1. *Application* dan *Widget*

Application dan *Widgets* ini adalah *layer* dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita download aplikasi kemudian lakukan instalasi dan jalankan aplikasi tersebut. Terdapat aplikasi inti pada *layer* termasuk klien email, program sms, kalender, peta, browser, kontak, dan lain lain. Semua aplikasi tersebut di tulis menggunakan bahasa pemprograman java.

2. *Application Framework*

Android *Application Frameworks* adalah *open development platform* yaitu android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi, *resources*, menjalankan *service background*, mengatur alarm, dan menambahkan status *notifications,* dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi

kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*). Sehingga bisa kita simpulkan *Applications Frameworks* ini adalah *layer* dimana para pembuat aplikasi melakukan pengembangan atau pembuatan aplikasi yang akan dijalankan disistem operasi android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat, seperti content providers yang berupa sms dan panggilan telepon. Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut.

1) *Views*
2) *Content Provider*
3) *Resource Manager*
4) *Notification Manager*

3. *Libraries*

Libraries ini adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan diatas kernel, *Layer* ini meliputi berbagai library C/C++ inti seperti Libe dan SSL, yaitu:

1) Libraries media untuk pemutaran media audio dan video
2) Libraries untuk manajemen tampilan
3) Libraries graphics mencakup SGL dan OpenGL untuk grafis 2D dan 3D
4) Libraries SQLite untuk dukungan *database*
5) Libraries SSL dan Wevkit terintegrasi dengan *web browser* dan *security*

6) Libraries LiveWebcore mencakup modern *web browser* dengan *engine embedded web view*
7) Libraries 3D yang mencakup implementasi OpenGL ES 1.0 API's

4. Android Run Time

Layer yang membuat aplikasi Android dapat dijalankan dimana didalam prosesnya menggunakan implementasi linux, dalvik virtual machine (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi android. Di dalam android run time dibagi menjadi dua bagian yaitu:

1) Core libraries merupakan aplikasi android dibangun dalam bahasa java, sementara Dalvik sebagai virtual mesinnya buka *virtual machine* java, sehingga diperlukan sebuah libraries yang berfungsi untuk menterjemahkan bahasa java/c yang ditangani oleh core libraries.
2) Dalvik *Virtual Machine* merupakan Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat linux kernel untuk melakukan *threading* dan manajemen rendah.

5. Linux Kernel

Linux kernel adalah *layer* dimana inti dari operating *system* dari android itu berada. Berisi *file system* yang mengatur *system processing, memory, resource, drivers,* dan sistem-sistem operasi android lainnya, linux kernel yang digunakan android adalah linux kernel release II.6

2.14 *Web Service*

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. Web service digunakan sebagai suatu fasilitas yang disediakan oleh suatu web site untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (service) yang disediakan oleh suatu sistem yang menyediakan web service. Web service menyimpan data informasi dalam format XML, sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa compiler.

Web service bertujuan untuk meningkatkan kolaborasi antar pemrogram dan perusahaan, yang memungkinkan sebuah fungsi di dalam *Web Service* dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detil pemrograman yang terdapat di dalamnya. Beberapa alasan mengapa digunakannya web service adalah sebagai berikut.

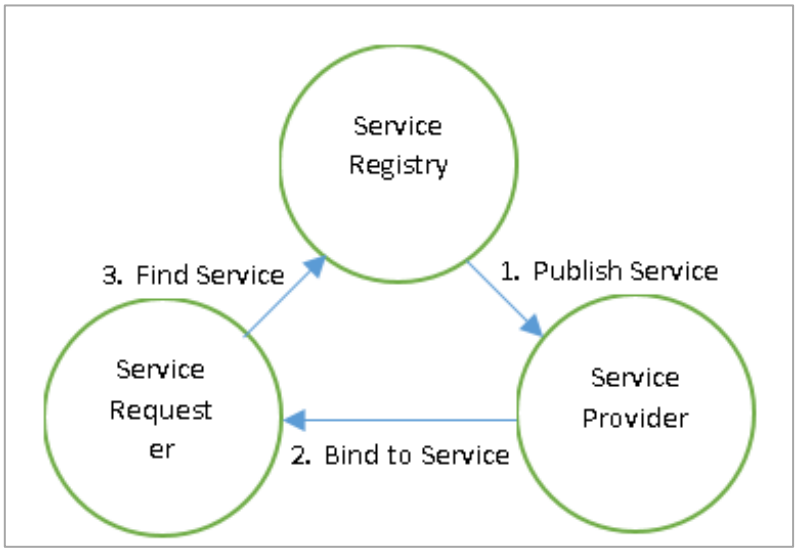
1. *Web service* dapat digunakan untuk mentransformasikan satu atau beberapa bisnis logic atau class dan objek yang terpisah dalam satu ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.
2. *Web service* memiliki kemudahan dalam proses deployment-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. Web service cukup di-upload ke web *Server* dan siap diakses oleh pihak-pihak yang telah diberikan otorisasi.

3. *Web servic*e berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian web service tidak memerlukan konfigurasi khusus di sisi firewall.

2.14.1 Arsitektur *Web Service*

Web service memiliki tiga entitas dalam arsitekturnya, yaitu:

1. *Service Requester* (peminta layanan)
2. *Service Provider* (penyedia layanan)
3. *Service Registry* (daftar layanan)



Gambar 2.42 Arsitektur *Web Sevice*

1) *Service Provider*: Berfungsi untuk menyediakan layanan/service dan mengolah sebuah registry agar layanan-layanan tersebut dapat tersedia.

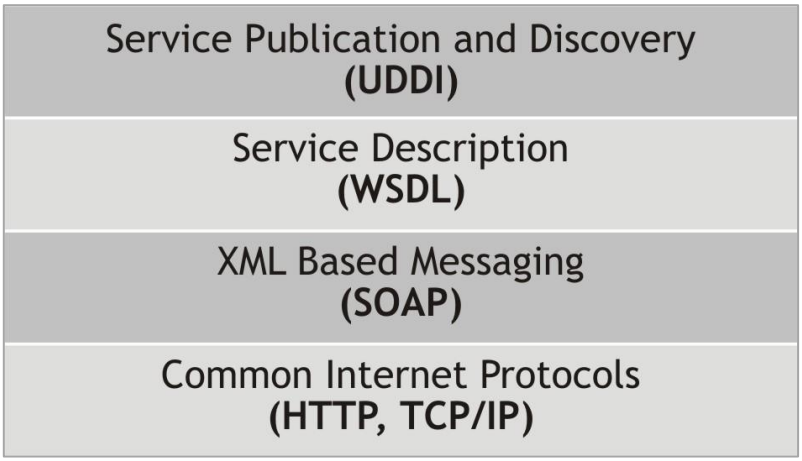
2) *Service Registry*: Berfungsi sebagai lokasi central yang mendeskripsikan semua layanan/service yang telah di-register.
3) *Service Requester*: Peminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut.

Secara umum, *web service* memiliki tiga operasi yang terlibat di dalamnya, yaitu:

1) *Publish/Unpublish*: Menerbitkan/menghapus layanan ke dalam atau dari registry.
2) *Find: Service requestor* mencari dan menemukan layanan yang dibutuhkan.
3) *Bind*: *Service requestor* setelah menemukan layanan yang dicarinya, kemudian melakukan binding ke service provider untuk melakukan interaksi dan mengakses layanan/service yang disediakan oleh service provider.

2.14.2 Komponen *web Service*

Web service memiliki beberapa komponen, Gambar 2.43 merupakan gambaran susunan dari komponen *web service*.



Gambar 2.43 Komponen *Web Sevice*

Web service secara keseluruhan memiliki empat *layer* komponen seperti pada gambar di atas, yaitu:

1. *Layer* 1, protokol internet standar seperti HTTP, TCP/IP
2. *Layer* 2, *Simple Object Access Protocol* (SOAP), merupakan protokol akses objek berbasis XML yang digunakan untuk proses pertukaran data/informasi antar layanan.
3. *Layer* 3, *Web Service Definition Language* (WSDL), merupakan suatu standar bahasa dalam format XML yang berfungsi untuk mendeskripsikan seluruh layanan yang tersedia.

BAB 3

Bahasa Pemrograman

Python
PHP
Java

3.1 *Python*

Python merupakan bahasa pemrograman yang berorientasi obyek dinamis, dapat digunakan untuk bermacam-macam pengembangan perangkat lunak. Python menyediakan dukungan yang kuat untuk integrasi dengan bahasa pemrograman lain dan alat bantu lainnya. Python dapat berjalan di banyak platform atau sistem operasi seperti Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, dan telepon genggam Nokia. Saat ini Python juga telah di-*porting* ke dalam mesin virtual Java dan .NET. Python didistribusikan dibawah lisensi *Open Source* yang disetujui OSI (*Open Source Initiatives*), sehingga Python bebas digunakan, gratis digunakan, bahkan untuk produk-produk komersil. Beberapa fitur yang dimiliki Python adalah sebagai berikut.

1. Memiliki kepustakaan yang luas, dalam distribusi Python telah disediakan modul-modul.
2. Memiliki tata bahasa yang jernih dan mudah dipelajari.
3. Memiliki aturan *layout* kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber.
4. Berorientasi obyek.
5. Dapat dibangun dengan bahasa Python maupun C/C++.

3.1.1 Sejarah Python

Python dikembangkan oleh Guido van Rossum pada tahun 1990 di CWI, Amsterdam sebagai lanjutan dari bahasa pemrograman ABC. Versi terakhir yang dikeluarkan CWI adalah 1.2. Tahun 1995, Guido pindah ke CNRI sambal terus melanjutkan pengembangan Python. Versi terakhir yang

dikeluarkan adalah 1.6. tahun 2000, Guido dan para pengembang inti Python pindah ke BeOpen.com yang merupakan sebuah perusahaan komersial dan membentuk BeOpen PythonLabs. Python 2.0 dikeluarkan oleh BeOpen dan setelah mengeluarkan Python 2.0, Guido dan beberapa anggota tim PythonLabs pindah ke DigitalCreations.

Pengembangan Python saat ini terus dilakukan oleh sekumpulan pemrogram yang dikoordinir Guido dan *Python Software Foundation*. *Python Software Foundation* sebagai pemegang hak cipta intelektual Python sejak versi 2.1 dan dengan demikian mencegah Python dimiliki oleh perusahaan komersial. Distribusi Python saat ini sudah mencapai versi 2.6.1 dan versi 3.0. Nama Python dipilih oleh Guido sebagai nama bahasa ciptaannya karena kecintaan Guido pada acara televisi *Monty Python's Flying Circus*. (Noprianto, 2002).

3.1.2 Keunggulan Python

Python sendiri memiliki beberapa keunggulan apabila dibandingkan dengan bahasa pemrograman lain yaitu sebagai berikut:

1. Sintaks yang digunakan mudah dibaca.
2. Kemampuan melakukan pengecekan sintaks yang tepat.
3. Bersifat *open source.*
4. Berorientasi objek secara intuitif.
5. *Library* standar dapat diperluas dan modul dari pihak ketiga dapat dibuat secara virtual untuk setiap kebutuhan.
6. Ekstensi dan modul dapat secara mudah ditulis dalam C, C++ (atau Java untuk Jython atau .NET untuk IronPython).

Keunggulan lain yang dimiliki oleh Python adalah kemampuan untuk mendeskripsikan pustaka-pustaka standar. Python juga dapat membuat *server web* hanya dalam 3 baris kode. Python juga dapat berintegrasi dengan *Component Object Model* (COM), .NET dan objek-objek *Common Object Request Broker Architecture* (COBRA). Python juga memiliki keunggulan yaitu tersedia untuk sistem operasi yang banyak digunakan seperti Windows, Unix/Linux, OS/2, Mac dan sistem operasi lainnya (Noprianto, 2002).

3.1.3 Konsep Dasar Pemrograman Python

Konsep dasar pemrograman python yang menjadi dasar dalam pemrograman python yaitu *if-else,* perulangan *for* dan *while*, *time delay* dan *function*.

1. *If-Else* (Pemilihan Kondisi)

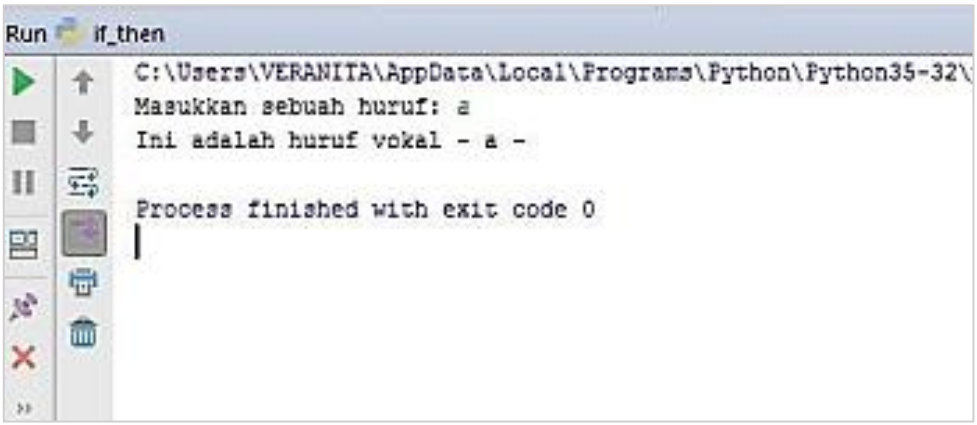
If-else merupakan *statement* yang digunakan untuk melakukan penyeleksian kondisi dimana jika kondisi bernilai benar maka program akan mengeksekusi *statement* pertama namun jika nilai kondisi bernilai salah maka *statement* kedua yang akan dieksekusi. Contoh implementasi dart perintah *if-else* dalam perograman Python adalah seperti Kode Program 3.1.

```
Huruf = input (str("Masukkan sebuah huruf : "))

If (huruf == 'a'):
print ("Ini huruf vocal – a –")
elseif (huruf == 'i'):
print ("Ini huruf vocal – i –")
elseif (huruf == 'u'):
print ("Ini huruf vocal – u –")
elseif (huruf == 'e'):
```

```
print ("Ini huruf vocal – e –")
elseif (huruf == 'o'):
print ("Ini huruf vocal – o –")

else:
print ("Ini bukan huruf vokal")
```

Kode Program 3.1 Implementasi Perintah *if-else*

Kode Program 3.1 merupakan implementasi perintah *if-else* untuk menentukan huruf vokal atau bukan merupakan huruf vokal. Dengan menggunakan 2 kondisi yaitu jika jika huruf yang diinputkan adalah huruf a, i, u, e dan o maka hasil akan menunjukkan bahwa *input*-an adalah huruf vokal. Jika huruf yang di-*input* tidak merupakan lima huruf di atas maka hasil *input*-an akan menunjukkan bahwa *input*-an tersebut adalah bukan huruf vocal. Hasil eksekusi dari program tersebut adalah seperti Gambar 3.1



Gambar 3.1 Hasil Eksekusi Perintah *if-else*

Gambar 3.1 merupakan hasil dari eksekusi perintah *if-else* menunjukkan bahwa huurf yang di-*input*-kan yaitu huruf a merupakan huruf vokal.

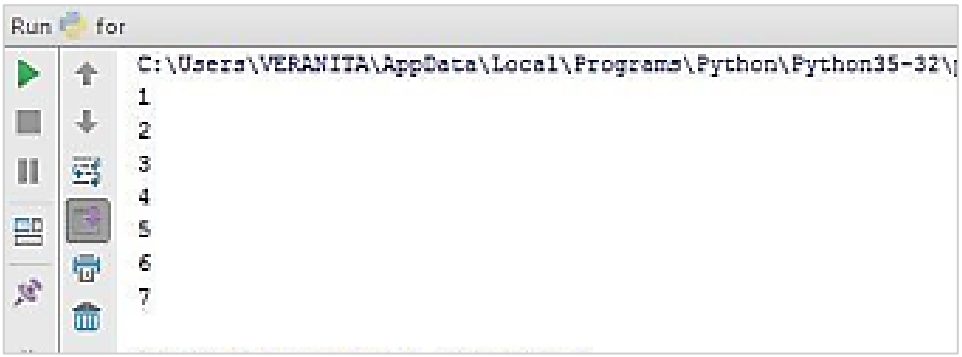
2. *For* (Perulangan)

For merupakan perintah yang digunakan untuk melakukan perulangan dan menghentikan perulangan dalam kondisi yang ditentukan. Contoh implementasi dari perintah *for* dalam perograman Python adalah seperti Kode Program 3.2.

```
Angka = [1,2,3,4,5,6,7,8,9]
For  x in angka:
      If x==0:
            Continue
      If x>7:
            Break
      Print (x)
Else:
Print ("Perulangan selesai")
```

Kode Program 3.2 Implementasi Perintah *For*

Kode Program 3.2 merupakan implementasi perintah *for*. Angka dideklarasikan yaitu angka = [1,2,3,4,5,6,7,8,9] merupakan rentang nilai yang akan di *looping* dengan kondisi jika nilai x adalah 0 maka akan diteruskan untuk melakukan looping sampai nilai x bernilai 7 maka proses *looping* akan berhenti. Hasil eksekusi dari program tersebut seperti pada Gambar 3.2.



Gambar 3.2 Hasil Eksekusi Perintah *for*

Gambar 3.2 merupakan hasil *run* perintah *for* yang menujukkan bahwa bilangan yang di *looping* untuk ditampilkan adalah 1,2,3,4,5,6,7 berdasarkan kondisi yang telah ditentukan dalam kode program.

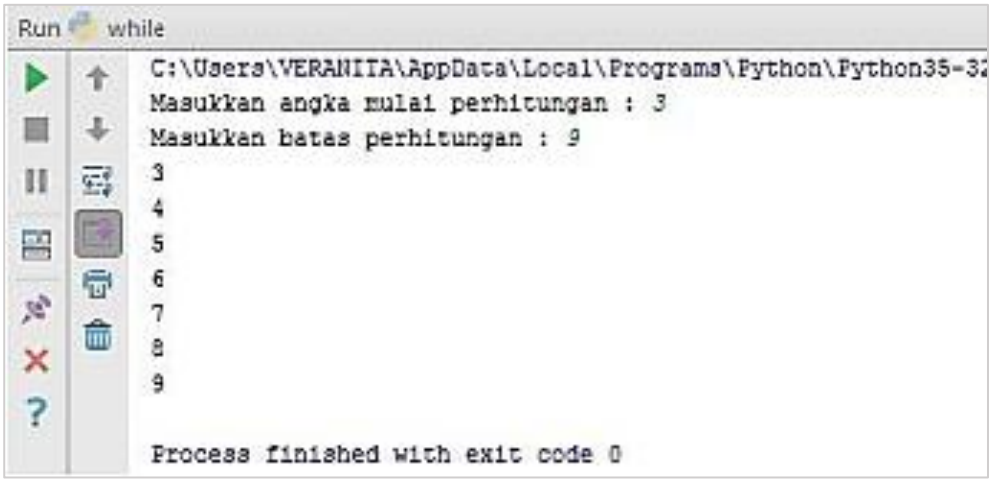
7. *While* (Eksekusi *Statement*)

While merupakan perintah yang digunakan dalam melakukan *iterasi* dengan konsep akan mengeiksekusi *statement* dalam blok *while* selama kondisinya benar dan tidak melakukan eksekusi blok *statement* jika nilai kondisinya salah. Contoh implementasi dari perintah *while* dalam pemrograman Python seperti pada Kode program 3.3.

```
mulai = int(input("Masukkan angka mulai perhitungan : "))
batas = int(input("Masukkan batas perhitungan : "))

while mulai <=batas:
      print(mulai)
      mulai = mulai+1
```

Kode Program 3.3 Implementasi Perintah *while*

Kode Program 3.3 merupakan implementasi perintah *while*. Deklarasikan mulai dan batas untuk dapat mengetahui batasan perhitungan yaitu mulai untuk memulai perhitungan dan batas untuk selesai *looping*. Hasil eksekusi dari program *while* adalah seperti Gambar 3.3.



Gambar 3.3 Hasil Eksekusi Perintah *while*

Gambar 3.3 merupakan hasil *run* yang menunjukkan bahwa hasil *looping* berdasarkan *input*-an nilai mulai dari batas yaitu 3 dan 9 akan melakukan *looping* dan menampilkan angka 3 sampai batas yang di-*input*-kan yaitu 9.

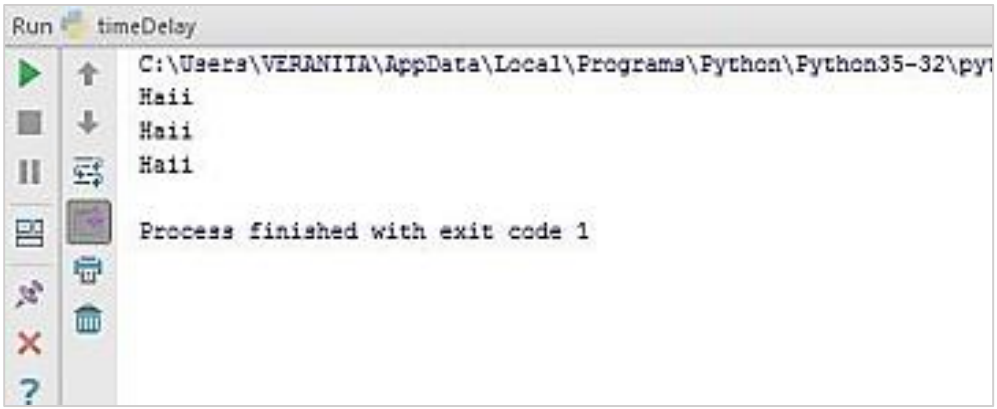
8. *Time delay* (Jeda Waktu)

Time delay merupakan perintah pada Python untuk memberikan sebuah jeda waktu. Contoh implementasi dari perintah *time delay* dalam pemrograman Python dapat dilihat pada Kode Program 3.4.

```
import time
while True:
      print ("Haii")
      time.sleep(5)
```

Kode Program 3.4 Implementasi Perintah *time delay*

Kode Program 3.4 merupakan implementasi perintah *time delay*. Baris kode *import time* berfungsi untuk meng-*import* modul waktu. Kode `time.sleep(5)` adalah *time delay* untuk mendefinisikan lama penundan eksekusi dengan parameter berupa angka yang menyatakan detik. Hasil eksekusi dari program *time delay* seperti Gambar 3.4.



Gambar 3.4 Hasil Eksekusi Perintah *time delay*

Gambar 3.4 merupakan basil dari eksekusi kode program *time delay* yang menunjukkan bahwa setiap 5 detik akan menampilkan *output* "Haii" dan seterusnya berulang.

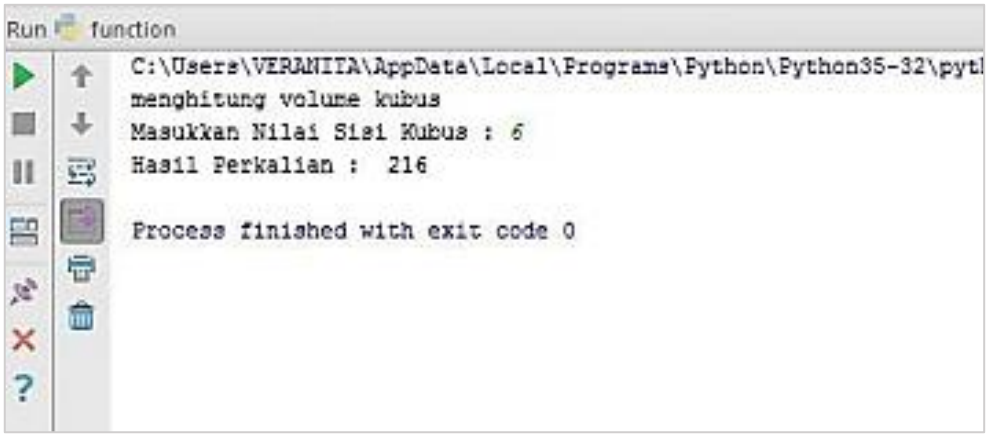
9. Fungsi (*Function*)

Fungsi merupakan program kecil umuk memproses sebagian dari pekerjaan program utama. Contoh implementasi dari perintah fungsi dalam pemrograman Python adatah seperti Kode Program 3.5.

```
Print ("menghitung volume kubus")
Sisi = int(input("Masukkan nilai sisi kubus : "))

Def luaskubus (x):
      Hitung = x*x*x
      Return hitung
Hitung = luskubus(sisi)
Print ("hasil perkalian : ", hitung)
```

Kode Program 3.5 Implementasi Perintah *function*

Kode program 3.5 merupakan implementasi Fungsi. Deklarasikan sebuah variabel yaitu sisi yang merupakan hasil *input*-an nilai sisi kubus. Nilai *input*-an akn dihitung dengan rumus hitung x*x*x. hasil eksekusi dari kode program 3.10 dapat dilihat pada gambar 3.5



Gambar 3.5 Hasil Eksekusi Perintah *function*

Gambar 3.5 merupakan hasil eksekusi fungsi dengan *input*-an nilai sisi kubus yaitu 6, sehingga dengan rumus x*x*x = 216 didapatkan hasil nilai kubus yaitu 216 yang diperoleh dari perkalian nilai sisi.

3.2 *PHP*

PHP merupakan singkatan *recursive* dari *Hypertext Processor*. Bahasa Pemrograman PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1994. PHP merupakan bahasa pemrograman berbasis *web* yang terintegrasi dengan HTML dan memiliki kemampuan untuk memproses data dinamis. Aplikasi-aplikasi yang dibangun oleh PHP pada umumnya memberikan hasil pada *web browser*, tetapi prosesnya secara keseluruhan dijalankan di *server* (Solichin, A., 2009).

PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1994. Pada waktu itu PHP masih bernama FI (*Form Interpreted*), yang wujudnya berupa sekumpulan *script* yang digunakan untuk pengolahan data *form* dari *web*. *Hypertext Preprocessing*/*Form Interpreter* (PHP/FI) dirilis oleh Rasmus untuk umum. Sifat dari PHP sendiri adalah *open source* (Solichin, A., 2009).

3.2.1 Kelebihan PHP

Kelebihan yang dimiliki PHP dari bahasa pemrograman lain adalah sebagai berikut (Solichin, A., 2009).

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang dalam penggunaannya tidak perlu sebuah kompilasi.
2. *Web Server* yang *support* PHP dapat ditemukan dimana-mana dari mulai IIS sampai dengan Apache, dengan konfigurasi relatif mudah.

3. Pengembangannya lebih mudah karena banyaknya sumber dan *developer*.
4. Pemahamannya yang tergolong mudah karena PHP adalah bahasa *scripting* yang paling banyak referensinya.
5. PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (Linux, Mac OS, Windows) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat eksekusi perintah-perintah sistem.

3.2.2 Tata Cara Penulisan *Script* PHP

PHP bersifat *case sensitif* yang artinya semua penulisannya harus sesuai dengan kamus data yang tersedia (Solichin, A., 2009). Struktur dari PHP adalah sebagai berikut.

```
<?php
//ini contoh komentar dalam php
echo "<h2>Contoh sederhana penulisan PHP</h2>";
$nama="Made";
echo "<p>$nama</p>";
$nilai=10;
$hasil=$nilai+10;
echo $hasil;
?>
```

Kode Program 3.6 Contoh sederhana struktur PHP

Kode Program 3.6 adalah contoh sederhana struktur PHP dengan penjelasan sebagai berikut (Solichin, A., 2009).

1. Awal kode harus diawali dengan "`<?php`" dan diakhiri dengan "?>". Dua perintah tersebut harus ada.
2. Barisan perintah PHP, pengguna bebas dalam penyisipan komentar dengan diawali tanda "//". Komentar digunakan sebagai pengingat

kumpulan baris dengan proses yang sama.

3. Kata "echo" digunakan sebagai media pengirim hasil ke *browser*, sehingga yang dikirimkan adalah bagian yang berada setelah perintah tersebut.
4. Variabel harus diawali dengan tanda dolar "$" dan dapat dipanggil kapanpun dalam halaman yang sama.
5. Satu *statement* (perintah) biasanya diakhiri dengan titik-koma (;).
6. *Case-sensitive* untuk nama *identifier* yang dibuat oleh *user* (berupa variabel, konstanta, fungsi dan lain-lain), namun "tidak *case-sensitive*" untuk *identifier built-in* dari PHP.

3.2.3 Variabel dalam PHP

Variabel digunakan untuk menyimpan sebuah *value*, data, atau informasi. Nama variabel diawali dengan tanda `$` dengan panjang tidak terbatas. Setelah tanda `$` diawali oleh huruf atau *underscore* (`_`). Karakter berikutnya bisa terdiri dari huruf, angka, dan karakter tertentu yang diperbolehkan (karakter ASCII dari 127 – 255). Variabel pada PHP bersifat *case-sensitive*, tidak perlu dideklarasikan, dan tidak boleh mengandung spasi (Solichin, A., 2009). Penulisan variabel yang benar pada PHP dapat dilihat pada Tabel 3.1.

Tabel 3.1 Penulisan Variabel PHP

Variabel Benar	Variabel Salah
- `$_name` - `$first_name` - `$name3` - `$name_3`	- `$3name` - `$name?` - `$first+name` - `$first.name` - `$first name`

Tabel 3.1 merupakan tabel yang berisi penulisan variabel PHP. Terdapat contoh penulisan variabel benar dan variabel salah.

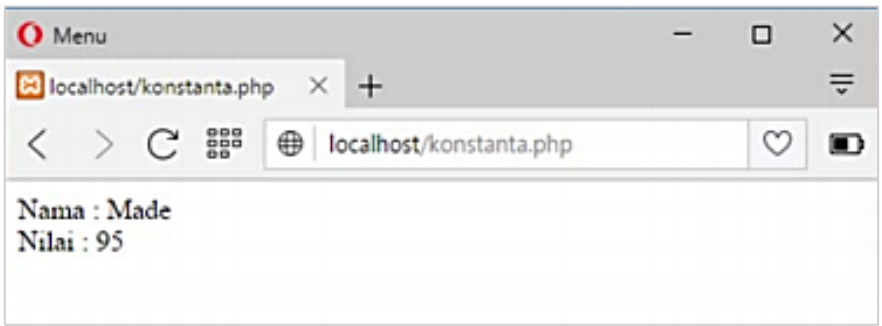
3.2.4 Konstanta pada PHP

Konstanta merupakan variabel konstan yang nilainya tidak berubah-ubah. Untuk mendefinisikan konstanta dalam PHP, menggunakan fungsi `define()` (Solichin, A., 2009).

```
<?php
define ("NAMA", "Made");
define ("NILAI", 95);
//NAMA = "Muhammad";//akan menyababkan error
echo "Nama : " . NAMA;
echo "<br>Nilai : " . NILAI;
?>
```

Kode Program 3.7 Contoh sederhana struktur PHP

Kode Program 3.7 merupakan contoh penggunaan konstanta dalam PHP. Hasil dari Kode Program 3.7 berupa nama dan nilai yang dapat dilihat pada Gambar 3.6. Hasil yang didapatkan yaitu Nama dan Nilai.



Gambar 3.6 Hasil dari Penggunaan Konstanta

3.2.5 Tipe Data pada PHP

Tipe data variabel tidak didefinisikan oleh *programmer* pada PHP, tetapi secara otomatis ditentukan oleh intepreter PHP. Namun demikian, PHP mendukung 8 (delapan) buah tipe data primitive (Solichin, A., 2009), yaitu:

1. *Boolean*

Tipe *Boolean* adalah tipe data pada PHP yang paling sederhana dalam bahasa pemrograman apapun karena tipe data ini hanya memiliki dua nilai yaitu *true* dan *false*. Tipe data *Boolean* sering kali digunakan pada operasi logika seperti kondisi *if* dan *looping*.

2. *Integer*

Integer adalah tipe data pada PHP yang berupa angka bulat seperti 1, 22, 100, 1000, tipe data ini sangat umum digunakan di bahasa pemrograman khusunya berkaitan dengan angka bulat. Nilai *integer* bisa bernilai negatif atau positif dan jika tidak diberi tanda (–) maka diasumsikan sebagai nilai positif.

3. *Float*

Float atau nama lainya adalah *floating point* atau *real number* adalah tipe data pada php yang memiliki bagian desimal di akhir angka contohnya adalah 3,21 atau 4,5. Penulisan tipe data *float* didalam php bukan menggunakan koma (,) tetapi menggunakan titik (.).

4. *String*

String adalah tipe data pada php yang berisi *text* dan karakter dimana bentuknya bisa kata atau kalimat. Dan dalam PHP untuk penulisan jenis tipe data ini ada 4 cara yaitu *Single Quoted, Double Quoted, Heredoc* dan *Nowdoc*.

5. *Array*

Array merupakan tipe data terstruktur yang berguna untuk menyimpan sejumlah data yang bertipe sama. Bagian yang menyusun *array* disebut elemen *array*, yang masing-masing elemen dapat diakses tersendiri melalui *index array*. *Index array* dapat berupa bilangan *integer* atau *string*.

6. *Object*

Object adalah tipe data yang memiliki kombinasi struktur data/atribut dan beberapa fungsi/*method*. Tipe data *object* pada PHP adalah untuk mendukung pemrograman berorientasi *object*.

7. *Resource*

Tipe data *resource* bukanlah suatu tipe data yang sebenarnya atau bukan tipe data aktual, karena tipe data ini hanya meyimpan referensi ke fungsi dan *resource* eksternal untuk PHP. Contoh umum penggunaan tipe data *resource* adalah pemanggilan *database*.

8. NULL

NULL menyatakan bahwa suatu variabel tidak memiliki nilai. NULL hanya merupakan nilai mungkin dari tipe NULL yang telah diperkenalkan pada PHP 4, dan keyword NULL adalah *case sentsitive*. Suatu variabel dianggap NULL, jika suatu variabel diberi nilai konstanta `NULL` atau suatu variabel belum pernah diisi nilai dan atau suatu variabel telah dikenakan fungsi `un_set()`.

3.3 Java

Java adalah bahasa pemrograman dan *platform* komputasi yang pertama kali dirilis oleh Sun Microsystems pada tahun 1995. Java adalah

bahasa pemrograman yang banyak digunakan, dirancang secara tegas untuk digunakan di lingkungan terdistribusi internet. Java merupakan bahasa pemrograman yang paling populer untuk aplikasi *smartphone* Android dan termasuk yang paling disukai untuk *edge device* dan *internet of things* (IoT). Java dirancang untuk memiliki tampilan serta nuansa seperti Bahasa C++ namun penggunannya lebih mudah daripada C++ dan menerapkan model Pemrograman Berorientasi Objek (OOP).

3.3.1 Karakteristik Java

Java memiliki macam-macam karakteristik. Penjelasan dari beberapa karakteristik Java yaitu sebagai berikut.

1. Sederhana

   Bahasa pemrograman Java menggunakan sintaks yang mirip dengan bahasa C++ namun sintaks pada Java telah banyak diperbaiki, terutama dengan menghilangkan *pointer* yang rumit dan *multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *garbage collection*.

2. Berorientasi Objek

   Java merupakan bahasa pemrograman berorientasi objek yang memungkinkan program untuk dibuat secara modular dan digunakan kembali.

3. Terdistribusi

   Java dibuat untuk memudahkan distribusi aplikasi dengan adanya *networking libraries* yang terintegrasi dalam Java.

4. *Interpreted*

   Program Java dijalankan menggunakan program *Interpreter*, yaitu Java Virtual Machine (JVM). Hal ini menyebabkan *source code* Java yang telah dikompilasi menjadi *bytecodes* dapat dijalankan pada berbagai *platform*.

5. *Robust*

   Java mempunyai reliabilitas yang tinggi. Kompiler pada Java mempunyai kemampuan mendeteksi *error* yang lebih baik dibandingkan bahasa pemrograman yang lain. Java mempunyai *Runtime Exception Handling* untuk membantu mengatasi *error* pada pemrograman.

6. *Secure*

   Sebagai bahasa pemrograman aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga agar aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7. *Architecture Neutral*

   Program Java tidak bergantung pada *platform* dimana program akan dijalankan. Cukup dibuat satu program yang dapat dijalankan pada berbagai *platform* dengan Java Virtual Machine.

8. *Portable*

   Source code maupun program Java dapat dengan mudah dibawa ke berbagai *platform* berbeda tanpa harus dikompilasi ulang.

9. *Performance*

   Kinerja Java sering kali dikatakan kurang, namun kinerja Java dapat ditingkatkan menggunakan *compiler* Java lain seperti buatan Inprise, Microsoft maupun Symantec yang menggunakan Just In Time Compilers (JIT).

10. *Multithreaded*

    Java dapat membuat suatu program yang mampu melakukan beberapa pekerjaan secara sekaligus dan simultan.

11. *Dynamic*

    Java dapat didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan suatu *class* dengan menambahkan *properties* ataupun metode dapat dilakukan tanpa menggangu program yang menggunakan *class* tersebut.

3.3.2 Dasar-dasar Pemrograman Java

Bahasa Pemrograman Java memiliki dasar-dasar untuk penggunaannnya. Macam-macam dasar tersebut yaitu:

1. Menulis Program Java

Penulisan program Java dapat dijelaskan sebagai berikut dengan menggunakan contoh untuk menampilkan kata "*Hello World*" pada Kode Program 3.8.

```
public class HelloWorld {
public static void main (String[] args) {
System.out.println("Hello World");
}}
```

Kode Program 3.8 Contoh Sintaks Program "Hello World"

Kode Program 3.8 merupakan sebuah program sederhana yang menampilkan tulisan "Hello World" pada *console*. Terdapat beberapa aturan dalam membuat program dalam Java yaitu:

1) Nama *file* harus sama dengan nama kelas program. Misal pada kode diatas nama kelasnya adalah HelloWorld, maka nama *file* harus HelloWorld.java.
2) Hanya boleh terdapat satu kelas *public* pada sebuah *file*.
3) Kelas yang menjadi program harus memiliki metode `public static void main(string[] args).`
4) Terminal pada Java menggunakan tanda ; (titik koma).

2. Tipe Data

Tipe data dalam Java dapat dijelaskan sebagai berikut dengan penjelasan tipe data *primitive* yaitu sebagai berikut.

Tabel 3.2 Tipe Data Primitive pada Java

Tipe Data	Keterangan
boolean	True atau false
char	Karakter
byte	-128 – 127
short	-32768 -32767
int	-2147483648 – 2147483647
long	-9223372036854775808 - 9223372036854775807
double	4.9E-324 – 1.7976931348623157E308
float	1.4E-45 – 3.4028235E38

Tabel 3.2 menunjukkan tipe data primitive pada Java. String bukan tipe data pada Java, String merupakan *Object*. Namun string memiliki keunikan yaitu String dapat langsung dibuat tanpa harus membuat *Object*.

3. Variabel

Variabel merupakan sesuatu yang digunakan untuk menampung sebuah data. Sebuah variable harus ada dalam sebuah kelas atau metode. Pembuatan sebuah variabel di Java dapat dilihat pada Kode Program 3.9.

```
int nilai;
char indexNilai;
```

Kode Program 3.9 Contoh Pembuatan Variabel

Kode program 3.9 merupakan contoh pembuatan variabel pada Java. Penambahan nilai ke sebuah variabel maka dapat menggunakan tanda "=" (sama dengan). Syarat-syarat penamaan variabel yaitu sebagai berikut.

1) Harus diawali dengan huruf.
2) Tidak boleh terdapat karakter unik seperti @, #, % dan lain-lain.
3) Tidak boleh mengandung karakter putih (spasi, enter, tab).

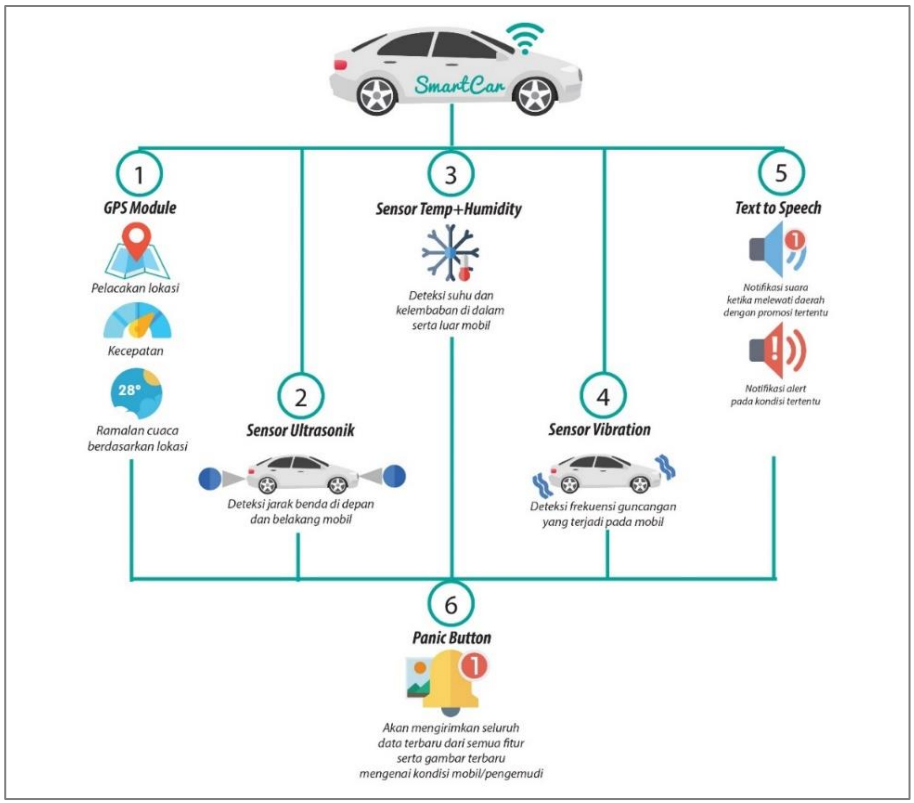
BAB 4

Rancangan Aplikasi SmartCar

Gambaran Umum
Pengertian Fitur Aplikasi SmartCar
Perancangan Sistem

4.1 *Gambaran Umum Aplikasi SmartCar*

Aplikasi SmartCar merupakan sebuah aplikasi (sistem) yang dirancang untuk membantu pengendara mobil dalam hal keselamatan mengemudi di jalan raya agar terhindar dari kecelakaan baik skala kecil maupun besar, terhindar dari tindakan kriminal serta memudahkan pengguna lainnya (seperti orang tua, saudara, atau kerabat) untuk memantau keadaan atau kondisi dari mobil yang menjadi obyek dari aplikasi SmartCar. Berikut merupakan gambaran umum dari aplikasi SmartCar.



Gambar 4.1 Gambaran Umum Aplikasi SmartCar

Gambar 4.1 merupakan gambaran secara umum dari aplikasi SmartCar yang terdiri dari enam fitur. Fitur yang tersedia diantaranya yaitu pelacakan lokasi, jarak aman, suhu dan kelembapan, deteksi getaran, notifikasi dan promo, serta tombol panik pada mobil yang menjadi obyek kemudian akan menangkap data dalam bentuk gambar lalu mengirimkannya beserta dengan seluruh data terbaru di setiap fitur pada aplikasi *web* maupun *mobile*.

4.2 Pengertian Fitur-Fitur Aplikasi SmartCar

Aplikasi SmartCar memiliki enam fitur utama yang telah diimplementasikan dalam aplikasi. Sub-bab dibawah ini menjelaskan tentang deskripsi dari setiap fitur yang ada pada aplikasi SmartCar.

4.2.1 Fitur GPS Tracking

Fitur GPS *Tracking* (pelacakan lokasi kendaraan) merupakan fitur yang berfungsi untuk melacak lokasi kendaraan kemanapun kendaraan melaju yang tercatat pada aplikasi *web* atau *mobile*. Sistem koordinat adalah metode *numeric* untuk merepresentasikan lokasi pada permukaan bumi. Derajat *latitude* dan *longitude* sering digunakan dalam penerapan pada sebuah sistem atau aplikasi. *Latitude* atau garis lintang adalah garis maya yang melingkari bumi ditarik dari arah barat hingga ke timur atau sebaliknya, sejajar dengan *equator* (garis khatulistiwa). Garis lintang terus melingkari bumi dari *equator* hingga ke bagian kutub utara dan kutub selatan bumi. *Longitude* atau garis bujur adalah garis maya yang ditarik dari kutub utara hingga ke kutub selatan atau sebaliknya. Alat-alat canggih kini telah berkembang pesat seiring juga dengan berkembangnya ilmu pengetahuan

dan teknologi yang dapat membantu dalam perkembangan tersebut. GPS atau *Global Positioning System* merupakan salah satu dari alat canggih tersebut. GPS merupakan sebuah alat atau sistem yang dapat digunakan untuk memberikan informasi mengenai keberadaan lokasi (secara global) di permukaan bumi yang berbasiskan satelit. Data dikirim dari satelit berupa sinyal radio dengan data digital. Layanan GPS tersedia gratis kecuali membeli GPS *reciever*.

Fitur pelacakan lokasi dibuat dengan beberapa komponen utama yaitu GPS *Module* yang telah tersedia pada Raspberry Pi 3 Tipe B, GPS NEO 6M, kabel jumper dan *breadboard*. *Breadboard* berfungsi sebagai penghubung atau perantara antara seluruh komponen yang dibutuhkan dalam aplikasi. GPS *Module* akan menangkap data *latitude*, *longitude* dan kecepatan saat seluruh komponen telah terhubung. Data *latitude*, *longitude* dan kecepatan akan dikirim setiap 2 detik. *Webservice* akan menyimpan data tersebut. Data *latitude* dan *longitude* digunakan untuk mengetahui lokasi mobil pada aplikasi SmartCar.

4.2.2 Fitur Jarak Aman

Fitur jarak aman merupakan fitur yang digunakan untuk menentukan jarak ideal pada mobil. Mobil yang melebihi jarak aman yang telah ditentukan, maka akan diberikan sebuah notifikasi. Fitur ini dibangun menggunakan sensor ultrasonik HC-SR04. Sensor ultrasonik akan menangkap jarak antara mobil dengan benda yang berada di depan atau belakang mobil, jika terjadi gesekan ataupun tabrakan maka sensor akan menangkap hasil kemudian hasil tersebut akan diolah ke proses selanjutnya.

4.2.3 Fitur Suhu dan Kelembapan

Fitur suhu dan kelembapan merupakan fitur yang digunakan untuk mengetahui suhu dan kelembapan udara mobil. Sensor yang digunakan pada fitur ini yaitu sensor DHT22. Sensor DHT22 merupakan sensor suhu dan kelembapan digital yang menghasilkan sinyal digital yang dikalibrasi. Fitur suhu dan kelembapan menggunakan dua sensor DHT22 yang diletakkan di luar dan di dalam mobil. Sensor akan mendeteksi suhu dan kelembapan udara mobil yang ada di dalam dan di luar, kemudian menghasilkan data berupa informasi suhu dan kelembapan udara. Data tersebut akan dikirim ke aplikasi *web* dan *mobile*. Raspberry Pi pertama kali akan membaca dan mengirim data dari sensor yang berada di luar mobil, lalu membaca dan mengirim data dari sensor yang berada di dalam mobil. Data sensor tersebut dibaca dan dikirim tiap 1 detik sekali. Proses selanjutnya yaitu suhu dan kelembapan di luar dan di dalam mobil akan dihitung selisihnya, jika selisih suhu dan kelembapan di luar dan di dalam mobil sama dengan kurang dari -10 itu artinya suhu dan kelembapan di dalam mobil lebih tinggi dibandingkan di luar, maka aplikasi akan memberikan notifikasi kepada pengendara untuk menghidupkan AC.

4.2.4 Fitur Deteksi Getaran

Fitur deteksi getaran pada SmartCar merupakan sebuah fitur yang menggunakan *Sw420 vibration sensor module* yang berfungsi untuk mendeteksi getaran yang terjadi pada mobil baik itu karena tabrakan, guncangan atau benturan dan hal lainnya. Cara kerja sensor ini adalah dengan menggunakan 1 buah pelampung logam yang akan bergetar didalam

tabung yang berisi 2 elektroda ketika modul sensor menerima getaran atau guncangan.

Penerapan sensor *vibration* pada aplikasi SmartCar yaitu dengan cara menghitung getaran setiap detiknya yang akan dikategorikan menjadi 3 kategori yaitu rendah, sedang dan tinggi. Kategori rendah jika getaran hanya mempunyai nilai kurang dari 75, kategori sedang jika getaran mempunyai nilai 75 – 150 dan kategori tinggi jika getaran melebihi nilai 150.

4.2.5 Fitur Notifikasi dan Promo

Fitur notifikasi dan promo merupakan salah satu fitur dari aplikasi SmartCar dengan memberikan notifikasi dan promo yang berada pada lokasi tertentu berupa audio yang dihasilkan speaker. Fitur notifikasi dan promo menggunakan GPS *Module* untuk mendeteksi *latitude* dan *longitude* dari suatu lokasi menggunakan *server* serta menghubungkan dengan *database* dan program Python untuk membuat sebuah *engine* yang apabila terdapat promo pada lokasi tertentu maka menghasilkan notifikasi berupa audio pada mobil. Fungsi dari fitur notifikasi dan promo adalah memudahkan pengguna untuk mengetahui promo dan notifikasi yang ada tanpa harus membaca pada saat berkendara, cukup dengan mendengarkan audio dari speaker pengendara akan mengetahui promo tersebut.

4.2.6 Fitur Tombol Panik

Fitur tombol panik merupakan salah satu fitur yang terdapat pada aplikasi SmartCar yang dapat digunakan ketika pengguna mengendarai mobil kemudian mengalami keadaan mendesak, bahaya, atau terjadi

permasalahan saat perjalanan seperti kriminalitas, ban bocor, mesin mati, kecelakaan dengan cara menekan tombol yang terdapat pada rangkaian alat kemudian akan mengirimkan notifikasi kepada kerabat pengguna yang telah menggunakan aplikasi SmartCar.

Fitur tombol panik pada aplikasi SmartCar menggunakan beberapa komponen utama yaitu *Tactile Switches Push Button* dengan ukuran 6x6 mm yang dihubungkan dengan Raspberry Pi 3 Tipe B, kabel jumper, dan *breadboard*. Komponen utama pada fitur ini dihubungkan dengan Raspberry Pi dan *breadboard* menggunakan kabel jumper sebagai penghubung. Tombol yang telah ditekan akan menghasilkan data yang kemudian akan dikirimkan untuk diolah kembali oleh *webservice* menjadi notifikasi yang akan dikirimkan kepada kerabat pengendara yang dituju.

4.3 Perancangan Sistem Aplikasi SmartCar

Penjelasan mengenai perancangan sistem pada aplikasi SmartCar adalah sebagai berikut.

4.3.1 Alat dan Bahan

Alat dan anggaran biaya dalam pembuatan aplikasi SmartCar dapat dilihat pada Tabel 4.1.

Tabel 4.1 Alat dan Bahan

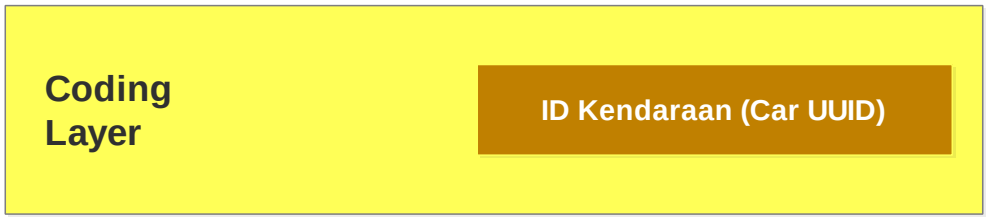
No.	BARANG	JUMLAH	HARGA
1	Raspberry Pi 3 Tipe B	1	Rp. 600,000
2	Sensor Ultrasonic	2	Rp. 50,000
3	*Push Button*	1	Rp. 12,500
4	Resistor 220Ω x 10	1	Rp. 2,500
5	*Breadboard* 830	1	Rp. 40,000
6	Dupon 40pm M-F	1	Rp. 30,000
7	Modul Penerima GPS	1	Rp. 295,000
8	Kamera	1	Rp. 65,000
9	*Casing* Raspy	1	Rp. 90,000
10	Resistor 1k 1% 1/4W x10	1	Rp. 2,500
11	Jumper F-f	1	Rp. 5,000
12	Adaptor	1	Rp. 62,500
13	*Power supply*	1	Rp. 17,500
14	Memori	1	Rp. 50,000
15	Lem Silikon	1	Rp. 22,500
16	Tupperware	1	Rp. 35,000
17	Isolasi listrik	3	Rp. 3,000
18	Cat pilox	1	Rp. 30,000
19	*Speaker*	1	Rp. 65,000
20	EPIO Ex board 40 pin	1	Rp. 90,000
21	*Break glass*	1	Rp. 150,000
22	*Temperature*	2	Rp. 100,000
23	*Tactile Switches* 6x6 mm	1	Rp. 2,000
24	Sensor *Vibration*	1	Rp. 15,000
25	Sinar searah elektronik	4	Rp. 6,000
TOTAL			**Rp. 1,841,000**

4.3.2 Arsitektur Sistem

Keterkaitan elemen pada projek dengan posisinya disetiap arsitektur Internet of Things pada masing-masing *layer* adalah sebagai berikut:

1. *Coding Layer*

Coding layer memberikan setiap mobil sebuah ID unik yang dapat memudahkan untuk membedakan obyek. *Coding layer* pada aplikasi SmartCar yaitu dengan mengimplementasikan ID unik untuk mobil (Car UUID). ID mobil akan mendefinisikan sebagai ID khusus untuk masing-masing mobil. Berikut ini merupakan gambaran arsitektur sistem dari *coding layer*.



Gambar 4.2 Arsitektur Sistem *Coding Layer*

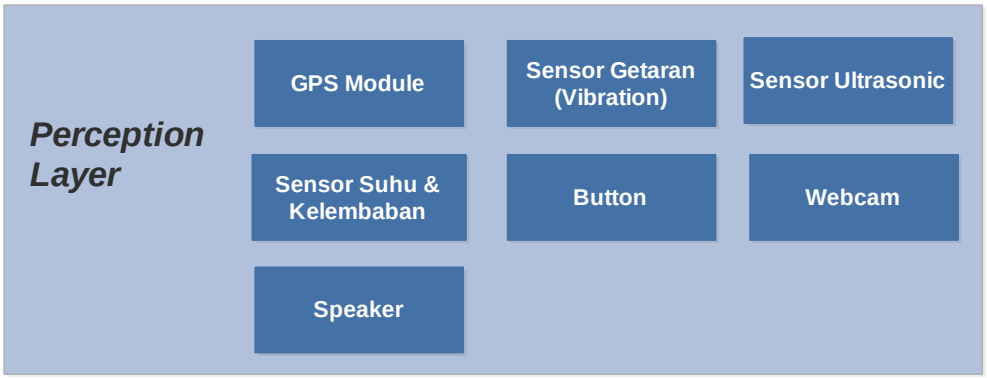
2. *Perception Layer*

Perception layer yang diimplementasikan pada aplikasi SmartCar yaitu terdapat komponen GPS Neo 6 M, WebCam, Sensor Ultrasonik, Sensor Suhu, Sensor *Vibration*, *Button*, *Breadboard* dan Kabel Jumper. Berikut merupakan implementasi dari setiap komponen *coding layer*.

1) GPS *Module* berfungsi untuk melakukan pelacakan lokasi dari pengendara mobil yang dapat mengetahui keberadaan dan membaca

letak lokasi dari pengendara mobil berdasarkan *latitude* dan *longitude*.

2) Sensor *Vibration* merupakan sensor yang berfungsi untuk mendeteksi getaran pada mobil.
3) Sensor Ultrasonik berfungsi untuk mendeteksi jarak mobil dengan obyek atau benda lain yang ada di depan dan belakang mobil. Sensor ultrasonik ini diimplementasikan pada aplikasi SmartCar untuk mengatasi terjadinya kecelakaan atau tubrukan.
4) Sensor Suhu & Kelembapan merupakan sensor yang berfungsi untuk mendeteksi suhu di dalam dan di luar mobil.
5) *Button* merupakan komponen fisik yang digunakan bagi pengendara dalam keadaan mendesak, dimana komponen ini akan terhubung dengan komponen lain yang terkait kemudian selanjutnya di proses.
6) WebCam berfungsi untuk mengambil gambar dengan format ekstensi JPG pada aplikasi SmartCar. WebCam akan mengambil gambar sesuai dengan fungsi kerjanya seperti implementasi pada komponen pada tombol panik, GPS dll.
7) Alat pendukung berupa *Speaker* pada aplikasi SmartCar digunakan sebagai pendukung dari fitur notifikasi.



Gambar 4.3 Arsitektur Sistem *Perception* Layer

3. *Network Layer*

Network layer yang diimplementasikan pada aplikasi SmartCar berfungsi untuk menerima *(receiving)* dan mengirim *(delivery)* data berupa informasi. Komponen yang termasuk dalam *network layer* yaitu Raspberry Pi 3 dengan model yang digunakan yaitu tipe B.

Gambar 4.4 Arsitektur Sistem *Network Layer*

4. *Middleware Layer*

Middleware layer merupakan layanan pendukung yang disediakan pada aplikasi SmartCar. Layanan pendukung yang dimaksud dan dijalankan

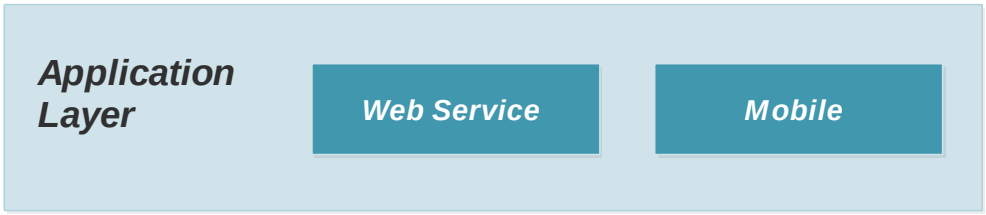
pada aplikasi SmartCar yaitu berupa jaringan internet sebagai sinyal digital yang nantinya menghubungkan antara *layer-layer* yang ada.

1) Kabel Jumper berfungsi sebagai penghubung pada setiap alat atau komponen pada aplikasi SmartCar.
2) *Breadboard* yaitu papan khusus yang digunakan untuk membuat prototipe atau rangkaian dari komponen atau alat aplikasi SmartCar.
3) *Internet* sebagai layanan sinyal yang menjadi keperluan utama dalam membangun aplikasi SmartCar.

Gambar 4.5 Arsitektur Sistem *Middleware Layer*

6. *Application Layer*

Application layer merupakan *layer* yang memberikan suatu aplikasi berupa antar muka *(interface)* yang nantinya dapat digunakan oleh pengguna.



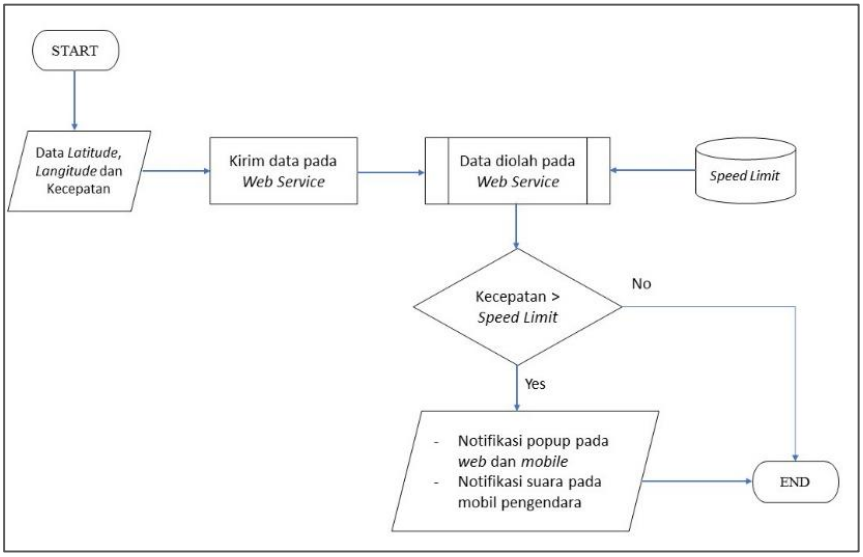
Gambar 4.6 Arsitektur Sistem *Application Layer*

4.3.3 *Flowchart* Fitur Aplikasi SmartCar

Flowchart pada sub-bab ini menjelaskan alur proses dari masing-masing fitur yang ada pada aplikasi SmartCar. Berikut merupakan *flowchart* dari masing-masing fitur.

1. *Flowchart* Pelacakan Lokasi

Flowchart untuk fitur pelacakan lokasi dengan menggunakan komponen GPS *module* dapat dilihat pada Gambar 4.7.



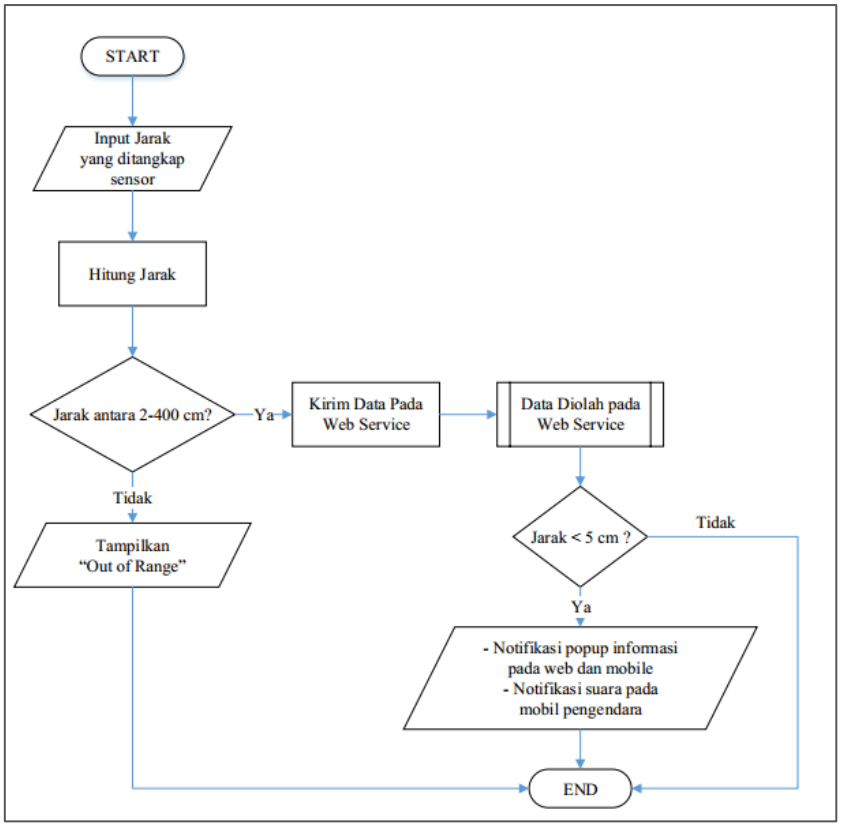
Gambar 4.7 *Flowchart* Fitur Pelacakan Lokasi dan Notifikasi Kecepatan

Gambar 4.7 merupakan *flowchart* dari fitur pelacakan lokasi dengan langkah awal yaitu mengambil data *latitude*, *longitude* dan kecepatan pada GPS *Module* Raspberry Pi, kemudian data tersebut disimpan dan diolah pada *webservice*. Apabila kecepatan melebihi *speed limit* atau batas kecepatan yang telah ditentukan pada *database* maka akan mengirimkan notifikasi

berupa suara atau audio serta notifikasi berupa *pop up* pada *web* dan *mobile* serta notifikasi suara pada mobil pengendara.

2. *Flowchart* Fitur Jarak Aman

Flowchart untuk fitur jarak aman dengan menggunakan sensor ultrasonik dapat dilihat pada Gambar 4.8.

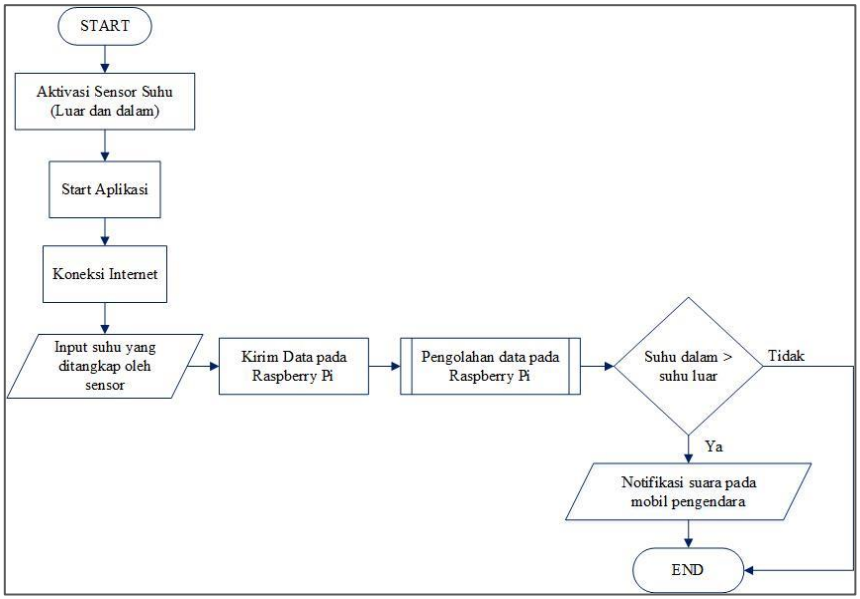


Gambar 4.8 *Flowchart* Fitur Jarak Aman

Gambar 4.8 merupakan alur atau *flowchart* dari cara kerja fitur jarak aman pada aplikasi SmartCar. Sensor ultrasonik yang terdapat pada aplikasi SmartCar terdapat pada bagian depan dan belakang mobil. Proses pertama yaitu melakukan aktivasi pada kedua sensor tersebut, kemudian proses selanjutnya yaitu dengan menghidupkan koneksi internet. *Input* data berupa jarak yang ditangkap oleh sensor ultrasonic kemudian dilakukan proses perhitungan jarak. Jika jarak diluar dari *range* 2-400 cm, maka akan menampilkan pesan "*Out of Range*". Jika jarak berada pada *range* 2-400 cm, maka data akan dikirim pada *web service*. Data yang telah ditangkap akan diolah pada *web service*. Jika jarak mobil dengan benda kurang dari 5 cm, maka jarak dikatakan tidak aman dan terdapat notifikasi berupa *popup* informasi pada *web* dan *mobile* serta notifikasi suara pada mobil pengendara. Apabila jarak mobil dengan benda lebih dari 5 cm, maka jarak dikatakan aman.

3. *Flowchart* Fitur Suhu dan Kelembapan

Flowchart untuk fitur suhu dan kelembapan dengan menggunakan sensor suhu dan kelembapan dapat dilihat pada Gambar 4.9.

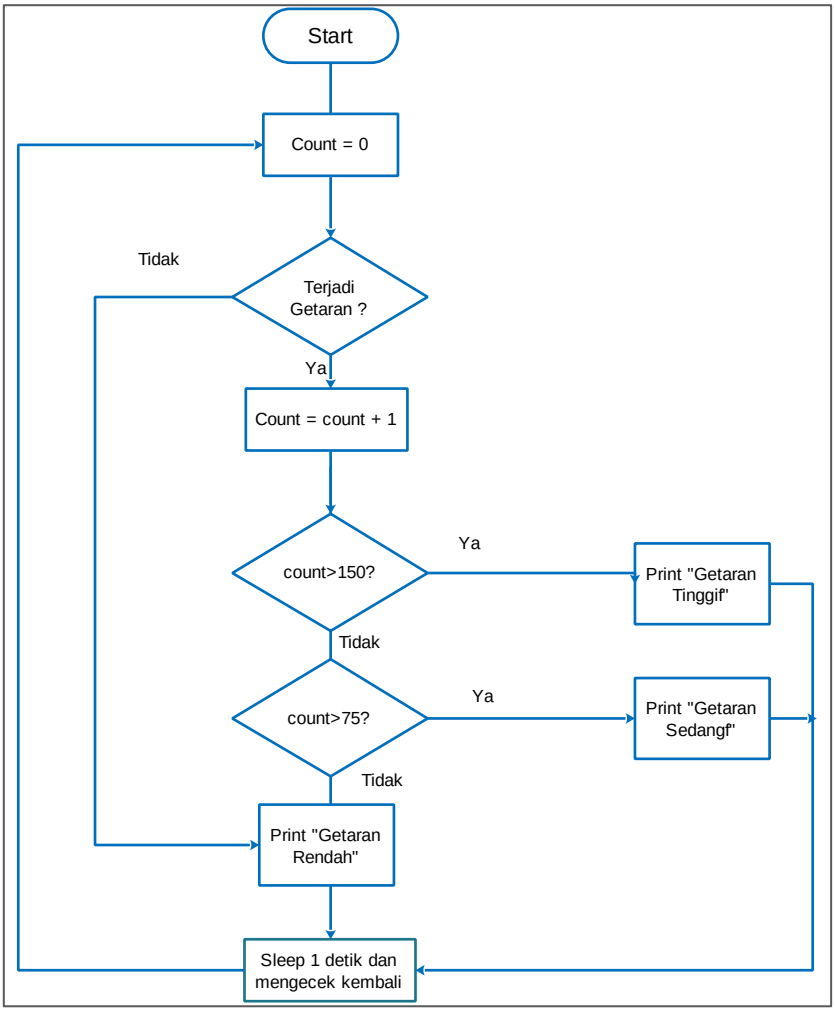


Gambar 4.9 *Flowchart* Fitur Suhu dan Kelembapan

Gambar 4.9 merupakan *flowchart* untuk sensor suhu dimana sensor dipasang pada bagian dalam dan luar mobil. Proses yang dilakukan setelah melakukan aktivasi pada kedua sensor yaitu menjalankan aplikasi dengan menghidupkan koneksi internet. Data suhu yang diperoleh sensor akan di *input* dan di kirim pada Raspberry Pi kemudian diolah. Jika suhu di dalam mobil lebih tinggi daripada di luar, maka terdapat notifikasi suara pada mobil pengendara.

4. *Flowchart* Fitur Deteksi Getaran

Flowchart untuk fitur deteksi getaran dengan menggunakan sensor *vibration* dapat dilihat pada Gambar 4.10.



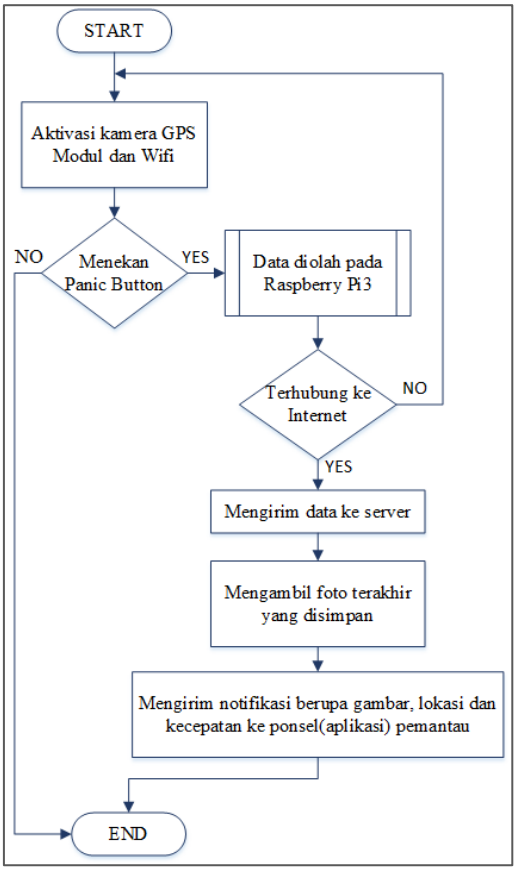
Gambar 4.10 *Flowchart* Fitur Deteksi Getaran

Gambar 4.10 menampilkan *flowchart* dari fitur deteksi getaran yang menjelaskan proses ketika mobil mengalami guncangan atau getaran. Proses pertama yaitu terdapat variabel dengan nama "count" untuk menampung jumlah getaran yang terjadi dengan nilai awal 0. Proses selanjutnya adalah sensor akan mendeteksi apakah terjadi getaran atau tidak, apabila terjadi

maka banyaknya getaran dalam 1 detik akan dihitung dan ditampung dalam variabel "count". Jumlah getaran tersebut dikelompokan dengan kategori rendah, sedang, tinggi. Jika nilai melebihi 150 maka akan menampilkan getaran tinggi, jika nilai diantara 75 dan 150 maka akan menampilkan getaran sedang, jika nilai kurang dari 75 maka akan menampilkan getaran rendah. Proses tersebut akan terus berulang setiap 1 detik, kemudian program akan menghitung kembali proses sebelumnya dan melakukan set dengan mengembalikan nilai "count" menjadi 0 agar tidak terjadi penumpukan data.

5. *Flowchart* Fitur Tombol Panik

Flowchart untuk fitur tombol panik ketika pengendara mengalami kondisi darurat dapat dilihat pada Gambar 4.11.

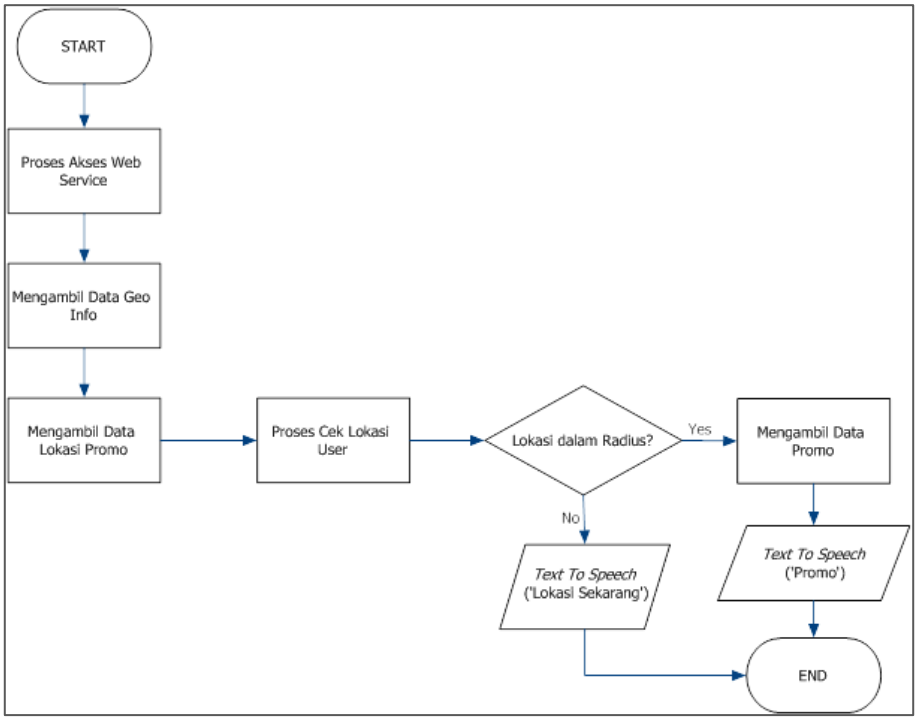


Gambar 4.11 *Flowchart* Fitur Tombol Panik

Alur pada Gambar 4.11 yaitu pertama mengaktifkan kamera, GPS *Module* serta Wifi kemudian setelah itu ketika tombol panik ditekan maka data akan diolah pada Raspberry Pi. Kondisi saat terhubung dengan internet dan data telah diolah menjadi informasi maka akan proses selanjutnya dilakukan pengiriman data ke *webservice* dengan mengambil gambar terakhir yang telah tersimpan serta notifikasi berupa gambar, lokasi, dan kecepatan ke aplikasi *web* dan *mobile*, apabila tidak terhubung dengan internet maka proses berhenti.

6. *Flowchart* Fitur Notifikasi dan Promo

Flowchart atau alur kerja untuk fitur notifikasi dan promo dapat dilihat pada Gambar 4.12.



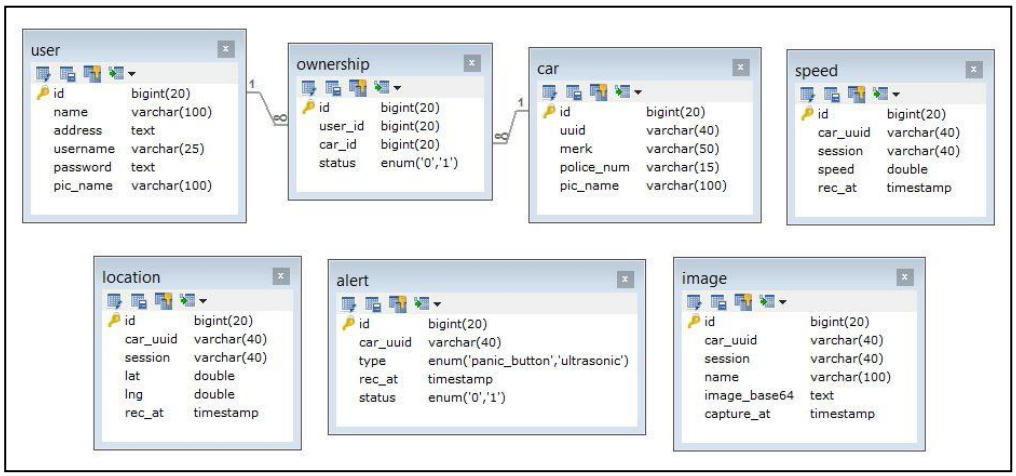
Gambar 4.12 *Flowchart* Fitur Notifikasi dan Promo

Gambar 4.12 menampilkan *flowchart* dari fitur notifikasi dan promo yang menjelaskan proses yang terjadi dalam fitur tersebut. Proses pertama adalah melakukan akses ke *webservice* untuk mengambil data yang diperlukan dari *database*. Proses selanjutnya yaitu mengambil data *geo info* dimana *geo info* menampilkan data lokasi pengendara berada. Proses selanjutnya adalah mengambil data lokasi promo dari *webservice* untuk

dilakukan proses cek apakah lokasi pengendara berada dalam radius dari lokasi promo, jika lokasi pengendara berada dalam radius maka proses dilanjutkan dengan mengambil data promo dari *webservice* dan dari data tersebut akan menghasilkan *output* suara berupa nama perusahaan dan promo yang ditawarkan dengan *text to speech*. Hasil dari proses cek lokasi pengguna yang tidak berada dalam radius maka hanya akan menghasilkan *output* suara sesuai lokasi pengendara berada dengan *text to speech*.

4.3.4 Rancangan *Database*

Rancangan *database* pada aplikasi SmartCar merupakan tahap perancangan untuk *database* sebagai alur tabel data dan penyimpanan data. Gambar 4.13 merupakan rancangan *database* dari aplikasi SmartCar.



Gambar 4.13 Rancangan *Database* Aplikasi SmartCar

Gambar 4.13 menampilkan tabel pada *database* yang digunakan aplikasi SmartCar yang terdiri dari tujuh buah tabel yaitu user, ownership,

car, speed, location, alert dan image. Tabel user memiliki enam buah *field* yaitu id, name, address, username, password dan pic_name. *Field* id digunakan untuk menyimpan *id* dari data *login* pengguna yang telah tersimpan pada *database* dengan tipe data *bigint* dan panjang 20. *Field* name digunakan untuk menyimpan data nama dari pengguna dengan tipe data *varchar* dan panjang 100. *Field* address digunakan untuk menyimpan data alamat pengguna dengan tipe data *text*. *Field* username digunakan untuk menyimpan *username* dari data *login* pengguna dengan tipe data *varchar* dan panjang 25. *Field* password digunakan untuk menyimpan data *password* pengguna dengan tipe data *text*. *Field* pic_name digunakan untuk menyimpan data nama gambar atau foto yang berhasil ditangkap oleh WebCam dengan tipe data *varchar* dan panjang 100.

Tabel ownership pada aplikasi SmartCar memiliki empat buah *field* yaitu id, user_id, car_id dan status. *Field* id digunakan untuk menyimpan *id* dari data *ownership* dengan tipe data *bigint* dan panjang 20. *Field* user_id digunakan untuk menyimpan *id user* pengguna dengan tipe data *bigint* dan panjang 20. *Field* car_id digunakan untuk menyimpan *id* mobil pengendara dengan tipe data *bigint* dan panjang 20. *Field* status digunakan untuk menyimpan *status* dengan tipe data *enum*.

Tabel car pada Aplikasi *Smart Car* ini digunakan untuk menyimpan data mobil yang terdiri dari lima *field* yaitu id, uuid, merk, police_num dan pic_name. *Field* id digunakan untuk menyimpan *id* dari data mobil dengan tipe data *bigint* dan panjang 30. *Field* uuid digunakan untuk menyimpan data uuid dengan tipe data *varchar* dan panjang 40. *Field* merk digunakan untuk menyimpan data merk mobil dengan tipe data *varchar* dan panjang 50. *Field*

police_num digunakan untuk menyimpan data nomor kendaraan dengan tipe data *varchar* dan panjang 15. *Field pic_name* digunakan untuk menyimpan data nama gambar atau foto dengan tipe data *varchar* dan panjang 100.

Rancangan *database* Aplikasi *Smart Car* memiliki tabel speed yang terdiri dari lima buah *field* yaitu id, car_uuid, session, speed dan rec_at. Tabel location yang memiliki enam buah *field* yaitu id, car_uuid, session, lat, lng, rec_at. Tabel alert yang digunakan untk menyimpan segala pemberitahuan. Tabel alert terdiri dari lima *field* yaitu id, car_uuid, type, rec_at dan status. Tabel image terdiri dari enam *field* yaitu id, car_uuid, session, name, image_base64 dan capture_at.

BAB 5

Kode Program SmartCar

Kode Program Aplikasi
Kode Program pada Web Service
Kode Program pada Mobile

5.1 Kode Program Aplikasi

Kode program aplikasi membahas mengenai kode program yang digunakan untuk menjalankan implementasi sistem dari aplikasi SmartCar yaitu Kode Program pada Raspberry Pi, Kode Program Fitur Jarak Aman, Kode Program Deteksi Suhu dan Kelembapan, Kode Program Deteksi Getaran, Kode Program Tombol Panik, Kode Program *Text to Speech*, Kode Program *Web Service*, dan Kode Program *Mobile*. Aplikasi SmartCar terdiri dari kode program untuk rangkaian Raspberry Pi dengan menggunakan bahasa pemrograman Python, *web service* dengan menggunakan bahasa PHP *(Hypertext Prepocessor)*, dan *mobile* menggunakan bahasa Java.

5.1.1 Kode Program pada Raspberry Pi

Kode program pada Raspberry Pi merupakan kode program yang digunakan untuk membangun sistem dari aplikasi SmartCar yang terdiri dari rangkaian komponen-komponen yang saling terhubung pada Raspberry Pi. Kode program Raspberry Pi pada aplikasi SmartCar menggunakan bahasa pemrograman Python yang dipisahkan berdasarkan fitur yang tersedia pada aplikasi.

1. Kode Program Fitur *Tracking* Lokasi (*Global Positioning System*)

Kode program fitur *Tracking* Lokasi pada aplikasi SmartCar terdiri dari kode program yang dirancang untuk proses data gambar serta kode program untuk mengambil dan mengirim data nilai *latitude, longitude* dan kecepatan.

1) Kode Program Data Gambar

Kode program data gambar merupakan kode program Python yang digunakan untuk pemrosesan data berupa gambar.

```
import os
import time, requests

url                                                    =
"https://topsus.digiforest.web.id/api/car/set_image"
car_uuid    = "8e1930b9-180c-4397-bbce-88a05be06de8"

while True:
    try:
#mengambil target
        os.system('sudo fswebcam --no-banner target.jpg')
        response                                       =
requests.post(url,data={'car_uuid':car_uuid},
files={'photo':open('target.jpg','rb')} )   #mendeklarasikan
file
        print response  # memberi respon

    except requests.exeptions.ConnectionError as e:
        print e
    except ValueError as e:
        print e
    time.sleep(300)
```

Kode Program 5.1 `Camera.py`

Kode Program 5.1 merupakan kode program yang digunakan untuk mengambil dan menyimpan gambar yang tertangkap pada *webcam*. *Library* Python yang digunakan yaitu os, *time* dan *request*. Variabel `url` digunakan untuk menyimpan hasil gambar ketika tombol ditekan. Hasil gambar tersebut disimpan pada laman https://topsus.digiforest.web.id/api/car/set_image. Varibel `car_uuid` merupakan identitas yang digunakan untuk

mengidentifikasi kendaran yang akan dipantau. Proses pengambilan gambar dilakukan setiap 5 menit.

2) Kode Program Mengambil Data Nilai *Latitude, Longitude* dan Kecepatan

Kode program 5.2 merupakan kode program Python yang digunakan untuk proses pengambilan data berupa nilai *latitude*, *longitude*, dan kecepatan.

```
#!/usr/bin/python
from sys        import argv
import gps
import gpxpy
import gpxpy.gpx
import os

os.system('stty -F /dev/ttyAMA0 9600')
os.system('sudo gpsd dev/ttyS0 -F /var/run/gpsd.sock')
session = gps.gps("localhost", "2947")
session.stream(gps.WATCH_ENABLE | gps.WATCH_NEWSTYLE)
loop = True
while loop:
#menggunakan session
    try:
        report = session.next()
            # wait for a 'TPV' report and display the current
time
        if report['class'] == 'TPV':
            # Abrir ficheiro para guardar as coordenadas
            coords = open ("coords.txt","w")
            coords.write(str(report.lat))
            coords.write(" ")
            coords.write(str(report.lon))
            coords.write(" ")
            coords.write(str(report.speed * gps.MPS_TO_KPH))
```

```
            coords.close()
    except KeyError:
        pass
    except KeyboardInterrupt:
        quit()
    except AttributeError:
        print ("Data tidak lengkap")
    except StopIteration:
        session = None
        print ("GPSD has terminated")
```

Kode Program 5.2 `Coordinate.py`

Kode program 5.2 merupakan kode program yang digunakan untuk mengambil nilai *latitude*, *longitude* dan kecepatan. *Library* Python yang digunakan yaitu gps, gpxpy, gpxpy.gpx dan os. Baris kode program `coords.write(str(report.lat))` digunakan untuk mengambil nilai *latitude* pada GPS *module* Raspberry Pi 3. Baris kode program `coords.write(str(report.lon))` digunakan untuk mengambil nilai *longitude* pada GPS *module* Raspberry Pi 3. Baris kode program `coords.write(str(report.speed * gps.MPS_TO_KPH))` digunakan untuk mengambil kecepatan. Apabila terjadi kesalahan atau gangguan maka akan menampilkan *output* "Data tidak lengkap".

3) Kode Program Mengirim Data Nilai *Latitude, Longitude* dan Kecepatan

Kode program 5.3 merupakan kode program Python yang digunakan untuk proses pengiriman data berupa nilai *latitude*, *longitude*, dan kecepatan.

```
import time, requests
url                                                              =
"https://topsus.digiforest.web.id/api/car/set_gps"
car_uuid    = "8e1930b9-180c-4397-bbce-88a05be06de8"
while True:
    try:
        coords  = open ("coords.txt","rb").read()
        print coords
        coords  = str(coords).split(' ')
        if(len(coords)>1):
          #data yang dikirim
            data    = {
                    'car_uuid'  : car_uuid,
                    'lat'       : coords[0],
                    'lng'       : coords[1],
                    'speed'     : coords[2]
            }
#respon data menggunakan j.son
            response = requests.post(url,data=data).json()
            print response
    except requests.exceptions.ConnectionError as e:
        print e
    except ValueError as e:
        print e
#looping time
    time.sleep(2)
```

Kode Program 5.3 `Geoloc.py`

Kode Program 5.3 merupakan kode program yang digunakan untuk mengirim nilai *latitude*, *longitude* dan kecepatan. Data yang dikirim ke *web service* yaitu UUID kendaraan, *latitude*, *longitude* dan kecepatan setiap 2 detik. *Library* Python yang digunakan yaitu *time* dan *request*.

5.1.2 Kode Program Fitur Jarak Aman (Sensor Ultrasonik)

Kode program fitur jarak aman merupakan kode program untuk mengolah data sensor ultrasonik yang digunakan untuk menentukan jarak

aman suatu kendaraan terhadap benda. Terdapat dua jenis kode program untuk mengolah data sensor ultrasonik yaitu sensor ultrasonik yang diletakkan pada bagian depan dan belakang kendaraan. Kode program sensor ultrasonik dapat dibedakan dengan kode `'type':'front'` untuk bagian depan, sedangkan sensor ultrasonik bagian belakang menggunakan kode `'type':'back'`.

1. Sensor Ultrasonik Bagian Depan

Sensor ultrasonik bagian depan digunakan untuk menentukan jarak aman pada bagian depan kendaraan.

```
#!/usr/bin/python
import RPi.GPIO as GPIO          #Import GPIO library
import time                      #Import time library
import requests
from datetime import datetime

url         =
"https://topsus.digiforest.web.id/api/car/set_ultrasonic"
car_uuid    = "8e1930b9-180c-4397-bbce-88a05be06de8"

GPIO.setmode(GPIO.BCM)                       #Set GPIO pin
numbering
TRIG = 23                                    #Associate pin
23 to TRIG
ECHO = 24                                    #Associate pin
24 to ECHO

print ("Distance measurement in progress")
GPIO.setup(TRIG,GPIO.OUT)                    #Set pin as GPIO
out
GPIO.setup(ECHO,GPIO.IN)                     #Set pin as GPIO
in

while True:
  try:
    GPIO.output(TRIG, False)                  #Set TRIG as
```

```
LOW
    print ("Waitng For Sensor To Settle")
    time.sleep(2)                                   #Delay of 2
seconds

    GPIO.output(TRIG, True)                         #Set TRIG as
HIGH
    time.sleep(0.00001)                             #Delay of
0.00001 seconds
    GPIO.output(TRIG, False)                        #Set TRIG as
LOW

    while GPIO.input(ECHO)==0:                      #Check whether
the ECHO is LOW
      pulse_start = time.time()                     #Saves the
last known time of LOW pulse

    while GPIO.input(ECHO)==1:                      #Check whether
the ECHO is HIGH
      pulse_end = time.time()                       #Saves the
last known time of HIGH pulse

    pulse_duration = pulse_end - pulse_start #Get pulse
duration to a variable

    distance = pulse_duration * 17150               #Multiply
pulse duration by 17150 to get distance
    distance = round(distance, 2)                   #Round to two
decimal points

    if distance > 2 and distance < 400:             #Check whether
the distance is within range
      print ("Distance:",distance - 0.5,"cm")  #Print
distance with 0.5 cm calibration
      data       = {
                       'car_uuid'   : car_uuid,
                       'distance'   : str(distance - 0.5),
                       'type'       : 'front',
                       'rec_at'     :
datetime.now().strftime('%Y-%m-%d %H:%i:%s')
                     }
      data        = requests.post(url,data=data).json()
      print (data)
```

```
    else:
      print ("Out Of Range")                    #display out
of range
      data       = {
                      'car_uuid'   : car_uuid,
                      'distance'   : 999,
                      'type'       : 'front',
                      'rec_at'     :
datetime.now().strftime('%Y-%m-%d %H:%i:%s')
                    }
      data        = requests.post(url,data=data).json()
      print (data)
  except requests.exceptions.ConnectionError as e:
    print (e)
```

Kode Program 5.4 `Ultrasonic.py`

Kode program 5.4 `Ultrasonic.py` merupakan kode program yang digunakan untuk mengolah data sensor ultrasonik yang diletakkan pada bagian depan kendaraan. GPIO pin yang digunakan pada sensor ultrasonik bagian depan yaitu pin 23 dan 24 yang dapat ditunjukkan pada kode `TRIG = 23` dan `ECHO = 24`.

2. Sensor Ultrasonik Bagian Belakang

Sensor ultrasonik bagian belakang digunakan untuk menentukan jarak aman pada bagian belakang kendaraan.

```
#!/usr/bin/python
import RPi.GPIO as GPIO          #Import GPIO library
import time                      #Import time library
import requests
from datetime import datetime

url         =
"https://topsus.digiforest.web.id/api/car/set_ultrasonic"
```

```
car_uuid    = "8e1930b9-180c-4397-bbce-88a05be06de8"

GPIO.setmode(GPIO.BCM)                          #Set GPIO pin
numbering
TRIG = 19                                       #Associate pin
19 to TRIG
ECHO = 26                                       #Associate pin
26 to ECHO

print ("Distance measurement in progress")
GPIO.setup(TRIG,GPIO.OUT)                       #Set pin as GPIO
out
GPIO.setup(ECHO,GPIO.IN)                        #Set pin as GPIO
in

while True:
  try:
    GPIO.output(TRIG, False)                      #Set TRIG as
LOW
    print ("Waitng For Sensor To Settle")
    time.sleep(2)                                 #Delay of 2
seconds

    GPIO.output(TRIG, True)                       #Set TRIG as
HIGH
    time.sleep(0.00001)                           #Delay of
0.00001 seconds
    GPIO.output(TRIG, False)                      #Set TRIG as
LOW

    while GPIO.input(ECHO)==0:                    #Check whether
the ECHO is LOW
      pulse_start = time.time()                   #Saves the
last known time of LOW pulse

    while GPIO.input(ECHO)==1:                    #Check whether
the ECHO is HIGH
      pulse_end = time.time()                     #Saves the
last known time of HIGH pulse

    pulse_duration = pulse_end - pulse_start #Get pulse
duration to a variable

    distance = pulse_duration * 17150           #Multiply
```

```
pulse duration by 17150 to get distance
    distance = round(distance, 2)             #Round to two
decimal points

    if distance > 2 and distance < 400:       #Check whether
the distance is within range
      print ("Distance:",distance - 0.5,"cm")  #Print
distance with 0.5 cm calibration
      data       = {
                      'car_uuid'   : car_uuid,
                      'distance'   : str(distance - 0.5),
                      'type'       : 'back',
                      'rec_at'     :
datetime.now().strftime('%Y-%m-%d %H:%i:%s')
                    }
      data        = requests.post(url,data=data).json()
      print (data)

    else:
      print ("Out Of Range")                    #display out
of range
      data       = {
                      'car_uuid'   : car_uuid,
                      'distance'   : 9999,
                      'type'       : 'back',
                      'rec_at'     :
datetime.now().strftime('%Y-%m-%d %H:%i:%s')
                    }
      data        = requests.post(url,data=data).json()
      print (data)
  except requests.exceptions.ConnectionError as e:
    print (e)
```

Kode Program 5.5 `Ultrasonic2.py`

Kode program 5.5 merupakan kode program yang digunakan untuk mengolah sensor ultrasonik yang diletakkan pada bagian belakang kendaraan. GPIO pin yang digunakan pada sensor ultrasonik bagian belakang yaitu pin 19 dan 26 yang dapat ditunjukkan pada kode `TRIG = 19` dan `ECHO = 26`. Jarak dapat dikatakan aman apabila jarak kendaraan dengan

benda melebihi 5 cm, jika jarak kendaraan dengan benda berada dibawah 5 cm maka kendaraan berada diluar jarak ideal yang telah ditentukan. Notifikasi jarak aman kendaraan diatur agar dapat muncul setiap 2 detik sekali selama kendaraan masih berada pada jarak aman.

5.1.3 Kode Program Deteksi Suhu dan Kelembapan (Sensor *Humidity)*

Kode program deteksi suhu dan kelembapan pada aplikasi SmartCar digunakan untuk menjalankan fungsi dari sensor *humidity* serta mendapatkan data suhu serta kelembapan yang ada didalam kendaraan dan diluar kendaraan.

```
import Adafruit_DHT           //library sensor suhu DHT22
import time, requests  //library time delay dan webservice


//deklarasi variabel url dan car_uuid
url                                                    =
"https://topsus.digiforest.web.id/api/car/set_temp_humid"
car_uuid    = "8e1930b9-180c-4397-bbce-88a05be06de8"

while True:
    try:
        //------ SUHU LUAR MOBIL ------------
        //deklarasi variable humidity dan temperature untuk
menggunaan librarry Adafruit_DHT untuk mendeteksi suhu luar
mobil
        humidity, temperature = Adafruit_DHT.read_retry(22,
22)
        if(humidity is not None and temperature is not None):

            //menampilkan   value   variable   humidity   dan
temperature
```

```
            print ("Humidity = {} %; Temperature = {}
C".format(humidity, temperature))

            //medifinisikan variable array data
            data    = {
                    'car_uuid'    : car_uuid,
                    'temp'        : temperature,
                    'humid'       : humidity,
                    'type'        : 'out'
            }


            response = requests.post(url,data=data).json()
            print response

        //----------------- SUHU DALAM MOBIL ---------------
------
       //deklarasi variable humidity dan temperature untuk
menggunaan librarry Adafruit_DHT untuk mendeteksi suhu luar
mobil
        humidity, temperature = Adafruit_DHT.read_retry(22,
21)
        if(humidity is not None and temperature is not None):
            //menampilkan value variable humidity dan
temperature
            print ("Humidity2 = {} %; Temperature2 = {}
C".format(humidity, temperature))

            //medifinisikan variable array data
            data    = {
                    'car_uuid'    : car_uuid,
                    'temp'        : temperature,
                    'humid'       : humidity,
                    'type'        : 'in'
            }

            //mengirim ke webservice
            response = requests.post(url,data=data).json()
            print response

    // kondisi jika terjadi error/gagal
```

```
    except ValueError as e:
        print e
    except requests.exception.ConnetionError as e:
        print e
    time.sleep(1)       //timedelay
```

Kode Program 5.6 Kode Program Deteksi Suhu dan Kelembapan

Kode program 5.6 merupakan kode program yang memiliki fungsi untuk mendeteksi suhu dan kelembapan pada aplikasi SmartCar dengan melakukan *import library* `Adafruit_DHT` yang terdapat dalam sensor. Alur kerja program yaitu sensor *humidity* akan mendeteksi suhu dan kelembapan udara, kemudian memberikan data berupa ID kendaraan, suhu, kelembapan, dan tipe data tersebut dengan keterangan '*out*' berarti sensor yang berada di luar kendaraan sedangkan '*in*' berarti sensor yang berada di dalam kendaraan. Data tersebut akan dikirim ke *website* aplikasi dengan menghubungkan url dari aplikasi SmartCar kemudian data tersebut akan ditampilkan. Raspberry Pi pertama kali membaca dan mengirim data berdasarkan sensor '*out*', kemudian setelah itu data berdasarkan sensor '*in*'. Jika suhu dan kelembapan udara di dalam kendaraan lebih tinggi dibandingkan di luar kendaraan maka *website* aplikasi akan memberikan notifikasi kepada pengguna untuk menghidupkan AC.

5.1.4 Kode Program Deteksi Getaran (Sensor *Vibration*)

Kode program deteksi getaran pada aplikasi SmartCar digunakan untuk menjalankan fungsi dari sensor *vibration.*

```
import RPi.GPIO as GPIO        #library GPIO
import math as m          #library aritmatika
```

```
import time, requests   #librari webservice

#deklarasi variabel url dan car_uuid
url         =
"https://topsus.digiforest.web.id/api/car/set_vibration"
car_uuid    = "8e1930b9-180c-4397-bbce-88a05be06de8"

#membuat class Sw40 untuk mendeklarasikan sensor
class Sw40(object):
    """docstring for Senal"""
    def __init__(self, pin , led):
        self.led = led
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.led,GPIO.OUT)
        self.pin = pin
        GPIO.setup(self.pin, GPIO.IN,
pull_up_down=GPIO.PUD_DOWN)

        GPIO.add_event_detect(self.pin, GPIO.RISING,
callback=self.callback, bouncetime=1)
        self.count = 0

    #fungsi untuk menghitung frekunsi getarana
    def callback(self , pin):
        self.count += 1

    #fungsi jika terdapat getaran
    def LedOn(self):
        GPIO.output(self.led , 1)

    //fungsi jika tidak terdapat getaran
    def LedOff(self):
        GPIO.output(self.led , 0)

#fungsi main program
def main():
    #deklarasi sensor dengan class Sw40
    sensor = Sw40(17,20)
```

```
    try:
        while True:

            time.sleep(1) #time delay

            #menampilkan jumlah getaran
            print "vibration count : "+str(sensor.count)
            #mendifinisikan variable array 'data'
            data    = {
                    'car_uuid'  : car_uuid,
                    'mount'       : sensor.count
            }
            #mengirim ke webservice
            response = requests.post(url,data=data).json()
            print response
            sensor.count = 0

   #error traping
    except KeyboardInterrupt:
        GPIO.cleanup()
    except requests.exception.ConnectionError as e:
        print e

#memanggil fungsi main
if __name__ == '__main__':
    main()
```

Kode Program 5.7 `Vibration.py` Deteksi Getaran (Sensor *Vibration*)

Kode Program 5.7 merupakan kode program yang digunakan untuk mendeteksi getaran pada aplikasi SmartCar dengan melakukan *import* pin GPIO pada Raspberry Pi kemudian data diproses pada *web service* dan terhubung dengan *website* dengan link https://topsus.digiforest.web.id/api/car/set_vibration dan ID kendaraan yang digunakan yaitu 8e1930b9-180c-4397-bbce-88a05be06de8. Kode

Program 5.7 terdapat variabel yang telah di deklarasikan untuk *self, pin dan led*. Variabel *pin* merupakan variabel yang digunakan untuk *input* data sedangkan variabel *led* merupakan variabel yang digunakan untuk *output* data. Hasil data ditampilkan dalam bentuk diagram garis. Kode program deteksi getaran juga memberi validasi berupa nilai kondisi, jika terjadi getaran *(Led On)* maka bernilai 1 *(true)* dengan fungsi kode program `GPIO.output(self.led, 1)` dan jika tidak terjadi getaran *(Led Off)* maka bernilai 0 dengan fungsi kode program `GPIO.output(self.led , 0)`. Data yang masuk akan diperbaharui sesuai waktu yang ditentukan dengan melakukan *looping* data dengan waktu 1 menit.

5.1.5 Kode Program Tombol Panik

Kode program tombol panik digunakan untuk memproses data ketika tombol di tekan oleh pemilik kendaraan.

```
import RPi.GPIO as GPIO        #import library GPIO
import time,requests, json     #library      time      delay,
webservice
GPIO.setmode(GPIO.BCM) #setting mode GPIO

#url webservice
url                                                          =
"https://topsus.digiforest.web.id/api/car/set_panic_alert"

#setup GPIO dengan number pin 27
GPIO.setup(27, GPIO.IN, pull_up_down=GPIO.PUD_UP)

#deklarasi variabel
prev_input = 0
first_time = True
```

```
while True:
 # deklarasi variabl input untuk menyimpan GPIO input dengan
numbir pin 27
  input = GPIO.input(27)
  #validasi menekan tombol (mencegah spaming)
  if ((not prev_input) and input and not first_time):
    try:
     # mengirim ke webservice
      response          = requests.post(url,data={'car_uuid'
: "8e1930b9-180c-4397-bbce-88a05be06de8"}).json()
      print (response)

    #error traping
    except requests.exceptions.ConnectionError as e:
        print (e)
    except ValueError as e:
        print (e)

  #deklarasi untuk mengubah variabel
  prev_input = input
  first_time = False
  time.sleep(0.05)     #timedelay
```

Kode Program 5.8 Tombol.py

Kode Program 5.8 merupakan kode program yang digunakan untuk mengatur proses ketika tombol ditekan menggunakan bahasa pemrograman Python. Kode program pada bagian `import RPi.GPIO as GPIO` berfungsi untuk memanggil *file library* untuk modul GPIO pada Raspberry Pi. Kode Program bagian `import time, requests, json` berfungsi untuk memanggil *library* waktu yang digunakan pada *time delay* dan koneksi dengan *web service*. Kode program bagian `GPIO.setmode(GPIO.BCM)` berfungsi untuk melakukan pengaturan modul GPIO untuk Raspberry Pi dengan metode BCM yang menggunakan *Pin Name*. Bagian kode program `url = "https://topsus.digiforest.web.id/api/car/set_panic_alert"`

memiliki fungsi untuk memberikan peringatan berupa notifikasi kepada pengguna ketika tombol ditekan pada halaman tersebut. GPIO yang digunakan oleh tombol untuk Raspberry Pi yaitu GPIO 27. Variabel `prev_input` digunakan untuk memberi nilai 0 sedangkan variabel `first_time` digunakan untuk memberi nilai *true*. `input` merupakan variabel yang digunakan untuk mendeklarasikan *input* GPIO yang digunakan oleh tombol. Proses pada kode program ini yaitu terdapat kondisi ketika tombol ditekan maka akan mencetak respon dari *web service* dengan memanggil laman yang telah dideklarasikan, namun jika koneksi dengan *web service* terjadi *error* maka akan menampilkan pesan *error*.

5.1.6 Kode Program *Text to Speech*

Fitur *Text to Speech* memanfaatkan teknologi TTS *(Text To Speech)* yang digunakan pada implementasi dari fitur notifikasi dan fitur promo.

1. Fitur Notifikasi

Kode program fitur notifikasi digunakan untuk menjalankan sistem aplikasi SmartCar yang ditampilkan dalam bentuk audio dengan memberikan sebuah notifikasi.

```
from gtts import gTTS
import os
import time, requests

requests.packages.urllib3.disable_warnings()
#url webservice
url                                                    =
      "https://topsus.digiforest.web.id/api/car/get_notif?
      car_uuid=8e1930b9-180c-4397-bbce-88a05be06de8"
car_uuid     = "8e1930b9-180c-4397-bbce-88a05be06de8"
last_id      = None
```

```
response    = requests.get(url).json() #mengambil data pada
      j.son
notifs      = response['data']  #deklarasi respon data
for notif in reversed(notifs):
    last_id = notif['id']

while True:
    try:
        response    = requests.get(url).json()
        notifs      = response['data']
        for notif in reversed(notifs):
#fungsi notif
            if (notif['id'] > last_id):
                last_id = notif['id']
                alert   = notif['message']
                tts     = gTTS(text=alert, lang='id')
                tts.save('tts.mp3')
                print alert
                os.system('omxplayer  -o  both  tts.mp3')
      #hasil menggunakan output tts format mp3
    except ValueError as e:
        print e
    except requests.exception.ConnectionError as e:
        print e
time.sleep(5) #time sleep
```

Kode Program 5.9 Kode Program Notifikasi

Kode Program 5.9 merupakan kode keseluruhan dari fitur notifikasi. Kode progam ini yang nantinya akan menampilkan notifikasi-notifikasi yang ada dalam aplikasi SmartCar. Baris kode program https://topsus.digiforest.web.id/api/car/get_notif?car_uuid=8e1930b9-180c-4397-bbce-88a05be06de8 merupakan kode untuk memperoleh informasi yang dihasilkan dari *web service* dengan url. Baris kode program `last_id = None` merupakan kode program yang menyatakan saat pertama kali notifikasi belum muncul, sehingga ID terakhir masih dalam status *none* atau tidak ada. Baris kode program `response = requests.get(url).json()`

merupakan kode program yang menyatakan sebagai *respond* atau timbal balik ketika notifikasi akan muncul maka data dari url akan di ambil dan notifikasi muncul sesuai dengan notifikasi yang akan dihasilkan. Baris kode program `notifs = response['data']` merupakan kode program yang menyatakan notifikasi dengan keterangan bahwa data telah terambil melalui *response* yang telah diperoleh melalui url. Maka `last_id = notif['id']` merupakan id terakhir dari notifikasi yang telah dihasilkan.

Baris Kode program `if (notif['id'] > last_id)` menyatakan jika ID notifikasi lebih besar dari id terakhir, maka `last_id = notif['id']` merupakan id notifikasi. Kode `alert = notif['message']` berfunsi untuk menghasilkan *message* atau pesan dari notifikasi. Baris kode program `tts = gTTS(text=alert, lang='id')` untuk membentuk pesan notifikasi menjadi audio menggunakan bahasa Indonesia. Baris kode program `tts.save('tts.mp3')` akan menyimpan *text to speech* dengan tipe file .mp3.

2. Fitur Promo

Kode program fitur notifikasi digunakan untuk menjalankan sistem aplikasi SmartCar yang ditampilkan dalam bentuk audio dengan memberikan informasi sebuah promo pada suatu tempat.

```
from gtts import gTTS
import os
import time, requests
requests.packages.urllib3.disable_warnings()
#url webservice
url                                                              =
"https://topsus.digiforest.web.id/api/car/get_info?car_uui
d=8e1930b9-180c-4397-bbce-88a05be06de8"
```

```
car_uuid    = "8e1930b9-180c-4397-bbce-88a05be06de8"
while True:
#mengambil data dari geo_info dan url serta daftar promo
    try:
        response = requests.get(url).json()
        geo_info = response['data']['geo_info']
        promos    = response['data']['promo']
        current_post                =    "Daerah    "    +
geo_info['administrative_area_level_4'] + " Kecamatan " +
geo_info['administrative_area_level_3']
        old_post        = ""
        if (current_post != old_post):
            response        =  '''Anda  sedang  berada  di
'''+current_post
            old = response
            tts = gTTS(text=response, lang='id')
            tts.save('tts.mp3')
            print response
            os.system('omxplayer -o both tts.mp3')
        response = ''''''
        for promo in promos:
            temp                      ='''Promo   dari   %s,
%s'''%(promo['company'],promo['promo'])
            tts = gTTS(text=temp, lang='id')
            tts.save('tts.mp3')
            print temp
            os.system('omxplayer -o both tts.mp3')
    except ValueError as e:
        print e
    except requests.exception.ConnectionError as e:
        print e
    time.sleep(60)
```

Kode Program 5.10 Informasi Promo

Kode Program 5.10 merupakan kode program keseluruhan untuk menjalankan fitur promo dalam aplikasi SmartCar. Kode Program informasi promo akan menghasilkan audio ketika tedapat promo atau diskon pada suatu tempat yang dilewati oleh pengguna aplikasi SmartCar.

Baris kode program dengan alamat url https://topsus.digiforest.web.id/api/car/get_info?car_uuid=8e1930b9-180c-4397-bbce-88a05be06de8 digunakan untuk mengambil data dari *web service*, kemudian dari alamat url tersebut akan diambil beberapa data agar dapat menampilkan fitur promo. Baris kode program `response = requests.get(url).json()` berfungsi untuk mengambil data dari url *web service* tersebut. Data yang akan diambli dari url tersebut dengan baris kode program `geo_info = response['data']['geo_info']` yaitu akan mengambil data mengenai lokasi dimana kendaraan berada, baris kode program `promos = response['data']['promo']` untuk mengambil data berupa promo apa saja yang sedang tersedia di sekitar atau *range* radius lokasi kendaraan berada.

Baris kode Program `if (current_post != old_post): response = '''Anda sedang berada di '''+current_post old = response` berarti `current_post` menyatakan lokasi terakhir kendaraan berada, `old_post` menyatakan lokasi sebelum lokasi terakhir atau lokasi lama. Jika lokasi terakhir tidak sama dengan lokasi lama, maka aplikasi akan menampilkan lokasi terakhir atau `current_post` pada layar. Baris kode program yang diucapkan oleh fitur *Text-to-Speech* yaitu `"Anda Sedang Berada di +current_post (tempat terakhir)"`. Baris kode program `tts = gTTS(text=response, lang='id')` berfungsi untuk mengaktifkan suara dari *text* yang di tuliskan pada baris kode program `lang='id'` yaitu dengan menggunakan bahasa Indonesia. Kemudian hasil *Text-to-Speech* tersebut akan disimpan dengan nama tts.mp3 .

Baris pada kode program `temp ='''Promo dari %s, %s'''%(promo['company'],promo['promo'])` yang berfungsi untuk

menampilkan promo yang tersedia pada lokasi tertentu. Baris pada kode program `tts = gTTS(text=response, lang='id')` berfungsi untuk mengaktifkan suara dari *text* yang di tuliskan pada kode program `lang='id'` yaitu dengan berbahasa Indonesia. Kemudian hasil *Text-to-Speech* tersebut akan disimpan dengan nama *file* tts.mp3.

5.2 Kode Program pada *Web Service*

Kode program pada *web service* digunakan untuk mengambil data yang dibutuhkan aplikasi SmartCar pada *database* melalui *web service*.

1. Fungsi Mengambil Data Kendaraan

Fungsi dari mengambil data kendaraan yaitu untuk mengambil data kendaraan pada *database*.

```
public function get_car($car_uuid)
{
$this->db->where('car_uuid',$car_uuid);
return $this->db->get('cars')->row_array();
}
public function get_owners($car_uuid)
{
$this->db->where('car_uuid', $car_uuid);
$this->db->where('status',"1");
return $this->db->get('ownership')->result();
}
```

Kode Program 5.11 Fungsi `get_car`

Kode Program 5.11 merupakan kode program berupa fungsi untuk mengambil data kendaraan yang terdapat pada *database*. Data yang terambil yaitu berdasarkan UUID kendaraan pengguna yang telah tersimpan. Selain kode program fungsi `get_car` terdapat pula fungsi `get_owners` yang

digunakan untuk mengambil data pemilik kendaraan pengguna yang telah tersimpan pada *database*.

2. Mengambil Data GPS *Module*

Fungsi dari mengambil data GPS *Module* yaitu untuk mengambil data lokasi kendaraan.

```
public function get_last_gps($car_uuid)
{
$this->db->where('car_uuid',$car_uuid);
$this->db->order_by('id','DESC');
$this->db->limit(1);
$gps    = $this->db->get('gps')->row_array();

if($gps)
{
$data   = array(
      "lat"   => $gps['lat'],;
      "lng"   => $gps['lng'],
      "speed" => $gps['speed'],
      "at"    => $gps['rec_at']
);
}
else{
$data   = array(
      "lat"   => 0,
      "lng"   => 0,
      "speed" => 0,
      "at"    => "--"
);
}
return $data;
}
```

Kode Program 5.12 Fungsi `get_last_gps`

Kode Program 5.12 merupakan kode program berupa fungsi untuk mengambil data terakhir pada GPS *module*. Kode Program `$this->db->where('car_uuid',$car_uuid);` berfungsi untuk mengambil data dari *database* berdasarkan UUID dari kendaraan yang digunakan. Kode Program `$this->db->order_by('id','DESC');` berfungsi sebagai *descendence* untuk mengurutkan data berdasarkan urutan yang terbesar. Data yang dihasilkan oleh GPS *module* yang kemudian tersimpan pada *database* yaitu *latitude*, *longitude*, kecepatan, dan waktu. Kode program `$data = array("lat" => $gps['lat'],; "lng" => $gps['lng'], "speed" => $gps['speed'], "at" => $gps['rec_at'] );` berfungsi untuk memperoleh nilai terakhir yang didapatkan dari GPS *module*. Status yang akan ditampilkan jika data tidak tersimpan yaitu dengan menampilkan nilai 0 atau tidak terdapat data. Data yang terambil berdasarkan urutan terakhir pada *database*.

3. Mengambil Data Sensor Ultrasonik

Terdapat dua kode program untuk mengambil data dari sensor ultrasonik yaitu mengambil data terakhir pada sensor ultrasonik dan mengambil data sensor ultrasonik berdasarkan ID kendaraan.

1) Data pada Sensor Ultrasonik

Kode Program 5.13 yaitu kode program yang digunakan untuk mengambil data terakhir yang masuk pada sensor ultrasonik.

```
public function get_last_ultras($car_uuid)
{
$this->db->select('`type`, MAX(id) as id');
$this->db->where('car_uuid',$car_uuid);
$this->db->group_by('type');
```

```
$ultras = $this->db->get('ultrasonic')->result_array();
$status = 0;
if(count($ultras)>=2)
{
$data = array();
foreach($ultras as $ultra) {
      if($ultra['type']=='front')
      {
            $front =$this->get_ultrasonic_id($ultra['id']);
            $data['front'] = $front['distance'];
            $data['front_update_at']= $front['rec_at'];
            if($front['distance']<5)
            {
                  $status = $status + 1;
            }
      }
      else if($ultra['type']=='back')
      {
            $back = $this->get_ultrasonic_id($ultra['id']);
            $data['back']          = $back['distance'];
            $data['back_update_at']= $back['rec_at'];
            if($back['distance']<5)
            {
                  $status = $status + 1;
            }
      }}}
else{
$data = array(
      "front"              => "--",
      "front_update_at"    => "--",
      "back"               => "--",
      "back_update_at"     => "--");
}
if($status == 0)
{$data['status'] = 'Aman';
}
else
{$data['status'] = 'Tidak aman';
}
return $data;}
```

Kode Program 5.13 Fungsi `get_last_ultras`

Data sensor ultrasonik digunakan untuk mengetahui jarak aman yang terdeteksi oleh sensor berdasarkan UUID kendaraan pengguna. Terdapat 2 data yang tersimpan pada *database* yaitu data sensor ultrasonik bagian depan dan sensor ultrasonik bagian belakang. Fungsi `get_ultrasonic_id($ultra['id']);` merupakan fungsi untuk memperoleh data dari sensor ultrasonik. Data yang terambil pada masing-masing sensor yaitu jarak dan waktu. Data yang tidak tersimpan pada *database* maka tidak ditampilkan. Terdapat kondisi dimana jika variabel `$status` memiliki nilai sama dengan nol maka status kendaraan aman, sedangkan jika variabel `$status` tidak sama dengan nol maka status kendaraan menjadi tidak aman.

2) Data Sensor Ultrasonik Berdasarkan ID Kendaraan

Data sensor ultrasonik berdasarkan ID kendaraan berfungsi untuk mengambil data pada sensor ultrasonik dimana data tersebut telah dikelompokan berdasarkan Car UUID (ID kendaraan).

```
private function get_ultrasonic_id($id)
{
$this->db->where('id',$id);
return $this->db->get('ultrasonic')->row_array();
}
```

Kode Program 5.14 Fungsi `get_ultrasonic_id`

Kode Program 5.14 merupakan kode program berupa fungsi yang digunakan untuk mengambil data pada sensor ultrasonik berdasarkan ID yang tersimpan pada *database*. Kode Program `$this->db->where('id',$id);` berfungsi untuk mengambil data berdasarkan ID dalam *database*.

4. Data Gambar

Data gambar berfungsi untuk mengambil data terakhir yang masuk pada sistem berupa gambar.

```
public function get_last_image($car_uuid)
{
$this->db->where('car_uuid',$car_uuid);
$this->db->order_by('id','DESC');
$this->db->limit(1);
$image  = $this->db->get('image')->row_array();

if($image)
{
$data = array(
      "dir"   => "".$image['dir'],
      "name"  => "".$image['name']
);
}
else{
$data = array(
      "dir"   => base_url("assets/images"),
      "name"  => "inside_car.jpg"
);
}
return $data;}
```

Kode Program 5.15 Fungsi `get_last_image`

Kode Program 5.15 merupakan kode program berupa fungsi yang digunakan untuk mengambil data terakhir berupa gambar atau foto. Kode program `"dir" => "".$image['dir']` berfungsi untuk menentukan *direction* atau *path* dimana gambar disimpan. Kode Program `"name" => "".$image['name']` berfungsi untuk memberikan nama pada gambar terakhir yang telah diambil. Data gambar akan tersimpan berdasarkan UUID pada kendaraan pengguna. Data yang terambil yaitu berdasarkan urutan terakhir pada *database*.

5. Data Notifikasi

Data notifikasi yang berfungsi untuk mengambil data terakhir yang masuk pada notifikasi.

```
public function get_last_alert($car_uuid, $type)
{
$this->db->where('car_uuid', $car_uuid);
$this->db->where('type', $type);
$this->db->order_by('id','DESC');
$this->db->limit(1);
return $this->db->get('alert')->row_array();
}
```

Kode Program 5.16 Fungsi `get_last_alert()`

Kode program 5.16 merupakan fungsi yang digunakan untuk mengambil data notikasi, dimana data pada *database* yang diambil yaitu UUID dari kendaraan. Kode Program `$this->db->where('type', $type);` berfungsi untuk memperoleh tipe notifikasi yang diperoleh oleh kendaraan. Data yang diambil diurutkan berdasarkan data notifikasi terakhir.

6. Data Sensor Getaran (*Vibration*)

Data sensor getaran *(vibration)* berfungsi untuk mengambil data terakhir yang masuk pada sensor getaran.

```
public function get_last_vibration($car_uuid)
{
$this->db->where('car_uuid',$car_uuid);
$this->db->order_by('id','DESC');
$this->db->limit(1);
$vibration = $this->db->get('vibration')->row_array();
if($vibration)
{
if($vibration['mount']<75)
{
```

```
        $category = 'Rendah';
}
else if($vibration['mount']<150)
{
        $category = 'Sedang';
}
else if($vibration['mount']>150)
{
        $category = 'Tinggi';
}
$data = array(
        "mount"     => "".$vibration['mount'],
        "at"        => "".$vibration['rec_at'],
        "category"  => $category
);
}
else{
$data = array(
        "mount"     => "--",
        "at"        => "--",
        "category"  => '--'
);
}
return $data;
}
```

Kode Program 5.17 Fungsi `get_last_vibration()`

Kode Program 5.17 menjelaskan mengenai fungsi `get_last_vibration()` yang digunakan untuk mengambil data terakhir dari sensor getaran. `if($vibration['mount']<75)` berarti getaran yang memiliki nilai <75 maka dikategorikan rendah, `else if($vibration['mount']<150)` berarti getaran <150 dikategorikan sedang dan `else if($vibration['mount']>150)` berarti getaran >150 makan dikategorikan tinggi. Hasil getaran yang diperoleh kemudian akan ditampilkan pada aplikasi SmartCar.

7. Data Sensor Suhu dan Kelembapan

Data sensor suhu dan kelembapan berfungsi untuk mengambil data terakhir yang masuk pada sensor suhu dan kelembapan.

```
public function get_last_temp_humid($car_uuid)
{
$this->db->select('type, MAX(id) as id');
$this->db->where('car_uuid',$car_uuid);
$this->db->group_by('type');
$temps = $this->db->get('temp_humid')->result_array();
if(count($temps)>=2)
{
$data = array();
foreach($temps as $temp){
      if($temp['type']=='in'){
            $in = $this->get_temp_humid_id($temp['id']);
            $data['inside_temp']    = $in['temperature'];
            $data['inside_humid']   = $in['humidity'];
            $data['inside_rec_at']  = $in['rec_at'];
      }
      else if($temp['type']=='out') {
            $out = $this->get_temp_humid_id($temp['id']);
            $data['outside_temp']    = $out['temperature'];
            $data['outside_humid']   = $out['humidity'];
            $data['outside_rec_at']  = $out['rec_at'];
      }
}
}
else{
$data = array(
      "inside_temp"    => "--",
      "inside_humid"   => "--",
      "inside_rec_at"  => "--",
      "outside_temp"   => "--",
```

```
        "outside_humid"  => "--",
        "outside_rec_at" => "--"
);
}
return $data;
}
```

Kode Program 5.18 Fungsi `get_last_temp_humid()`

Kode Program 5.18 merupakan kode program untuk sensor suhu. Kode program `if($temp['type']=='in')` berfungsi untuk sensor suhu di keadaan suhu dalam kendaraan, sedangkan `else if($temp['type']=='out')` berfungsi untuk sensor suhu yang mengukur keadaan suhu diluar kendaraan. Fungsi `get_last_temp_humid()` yang digunakan untuk mengetahui data terakhir dari sensor suhu dan kelembapan.

8. Data Kendaraan

Kode program untuk mengambil data dari kendaraan terdapat dua jenis yaitu menampilkan data kendaraan dan menampilkan data kendaraan yang dikelompokan berdasarkan Car UUID (ID Kendaraan).

1) Data Kendaraan

Kode program data kendaraan yaitu digunakan untuk menampilkan data dari kendaraan yang telah terdaftar pada sistem.

```
public function get_dashboard_info($car_uuid)
{
$this->db->where('car_uuid',$car_uuid);
$car = $this->db->get('cars')->row_array();
$data = array(
"car"    => $car,
"gps"    => $this->get_last_gps($car_uuid)
```

```
);
return $data;  }
```

Kode Program 5.19 Fungsi `get_dashboard_info()`

Kode Program 5.19 merupakan fungsi `get_dashboard_info()` yang digunakan untuk menampilkan data mobil yang digunakan dan lokasi kendaraan. Kode program `"car" => $car,` berfungsi untuk memperoleh data mengenai kendaraan tertentu. Kode program `"gps" => $this->get_last_gps($car_uuid)` berfungsi untuk memperoleh informasi terakhir dari GPS berdasarkan UUID.

2) Data Kendaraan Berdasarkan Car UUID

Kode program yang digunakan pada data kendaraan berdasarkan Car UUID berfungsi untuk mengambil data yang dikelompokan berdasarkan Car ID (ID Kendaraan).

```
public function get_notification($car_uuid)
{
$this->db->where('car_uuid',$car_uuid);
$this->db->order_by('id','DESC');
$this->db->limit(10);
$data  = $this->db->get('alert')->result_array();
return $data;   }
```

Kode Program 5.20 Fungsi `get_notification`

Kode Program 5.20 merupakan kode program berupa fungsi `get_notification` yang digunakan untuk mengambil notifikasi berdasarkan UUID kendaraan pengguna. Data notifikasi diurutkan berdasarkan data terakhir yang tersimpan pada *database*.

9. Data Informasi Promo

Kode program data informasi promo merupakan kode program yang digunakan untuk mengambil data dari informasi promo.

```
public function get_promo($car_uuid)
{
$gps = $this->get_last_gps($car_uuid);
$lat = $gps['lat'];
$lng = $gps['lng'];
$this->db->select('*, (6371 * ACOS(COS( RADIANS('.$lat.')) *
COS( RADIANS(lat) )
                        *      COS(      RADIANS(lng)      -
RADIANS('.$lng.')) + SIN(RADIANS('.$lat.'))
                        *     SIN(     RADIANS(lat))))     AS
distance');
$this->db->from('promo');
$this->db->where('ABS(lat) <= ABS('.($lat - 0.005).')');
$this->db->where('ABS(lat) >= ABS('.($lat + 0.005).')');
$this->db->where('ABS(lng) >= ABS('.($lng - 0.005).')');
$this->db->where('ABS(lng) <= ABS('.($lng + 0.005).')');
$this->db->where('NOw() BETWEEN date_start AND date_end');
$this->db->having('distance <= 0.5');
$this->db->order_by('distance','ASC');
$this->db->limit(2);
$data = $this->db->get()->result_array();
return $data;
}
```

Kode Program 5.21 Fungsi `get_promo`

Kode Program 5.21 digunakan untuk mendapatkan pemberitahuan tentang promo yang ada di tempat tertentu. Data promo diambil menggunakan berdasarkan lokasi yang diperoleh dari GPS *module*. Untuk memperoleh data promo berdasarkan GPS tersebut, data yang digunakan adalah *longitude* dan *latitude* dari tempat promo berada. Notifikasi promo

nantinya akan memberikan notifikasi jika kendaraan memiliki selisih o.oo5 dari *longitude* dan *latitude* dimana tempat promo berada.

10. Pemrosesan Data

Kode program pemrosesn data yaitu kode program yang digunakan untuk menampung dan menampilkan serta menambahkan data yang tersedia pada sistem.

```
public function set_alert($data){
$gps    = $this->get_last_gps($data['car_uuid']);
$ultra  = $this->get_last_ultras($data['car_uuid']);
$image  = $this->get_last_image($data['car_uuid']);
$data['last_location']      = $gps['lat'].",".$gps['lng'];
$data['last_speed']         = $gps['speed'];
$data['last_front_distance']= $ultra['front'];
$data['last_back_distance'] = $ultra['back'];
$data['last_image']                                    =
$image['dir']."/".$image['name'];
$this->db->insert('alert', $data);}
```

Kode Program 5.22 Fungsi `set_alert`

Kode Program 5.22 digunakan untuk memuat atau menambah data lokasi, kecepatan, jarak aman kendaraan, dan gambar *update* terakhir yang diperoleh dari sensor – sensor dan disimpan pada *database*.

11. Pengelolaan Data Pengguna pada *Database*

Pengelolaan data *user* berfungsi untuk melakukan pengelolaan data pengguna yang terdapat pada *database*.

```
<?php

class Users_model extends CI_Model
{
    function __construct()
      {
            parent::__construct();
            $this->load->database();
      }
    public function sign_in($username, $password)
    {
        $this->db->where('username', $username);
            $this->db->where('password', $password);
        $this->db->limit(1);
        $this->db->select('id, name, address, username, dir,
pic_name');
        return $this->db->get('users')->row_array();
    }
    public function sign_up($data)
    {
        $this->db->insert('users', $data);
        $id = $this->db->insert_id();

        $this->db->where('id',$id);
        $this->db->select('id,    name,    username,    dir,
pic_name');
        return $this->db->get('users')->row_array();
    }
    public function checkUsername($username)
    {
        $this->db->where('username', $username);
        $query = $this->db->get('users');
        if ($query->num_rows() > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

```
    public function get_ownership($user_id)
    {
        $this->db->where('user_id',$user_id);
        $this->db->where('status',"1");
        return $this->db->get('ownership')->result();
    }
    public function get_cars($user_id)
    {
        $ownerships = $this->get_ownership($user_id);
        $cars       = array();
        foreach($ownerships as $ownership)
        {
            $this->db->where('car_uuid',$ownership-
>car_uuid);
            $car = $this->db->get('cars')->row_array();
            array_push($cars, $car);
        }
        return $cars;
    }
    public function get_alerts($user_id)
    {
        $this->db->where('user_id', $user_id);
        $this->db->where('status', '0');
        return $this->db->get('alert')->result();
    }
    public function get_user($user_id)
    {
        $this->db->where('id', $user_id);
        $this->db->select('id, dir, pic_name, name, address,
username');
        return $this->db->get('users')->row_array();
    }
}
```

Kode Program 5.23 `Users_model.php`

Kode Program 5.23 digunakan untuk mengolah data yang berhubungan dengan pengguna pada *database*. Terdapat beberapa fungsi yaitu, `function sign_in`, `function sign_up`, `function checkUsername`,

`function get_ownership`, `function get_cars`, `function get_alerts`, dan `function get_user`. Fungsi `function sign_in` merupakan fungsi yang digunakan untuk mengambil data pengguna yang memiliki *username* dan *password*. Fungsi `function sign_up` digunakan untuk menambah data pengguna dan mengambil kembali data pengguna pada *database*. Fungsi `function checkUsername` merupakan fungsi untuk melakukan cek data pengguna pada *database*.

5.3 Kode Program pada *Mobile*

Kode program pada *mobile* pada aplikasi SmartCar berbasis Android yang nantinya digunakan untuk membangun suatu *interface* lain selain *web service* dengan fungsi yang sama yaitu untuk melihat fitur dan notifikasi kondisi dari SmartCar. Berikut merupakan kode program utama yang terdapat pada *mobile*.

1. *Grandle* Aplikasi

```
Dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    // OneSignal SDK
    //compile 'com.onesignal:OneSignal:[3.5.3,4.0.0)'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    compile

'org.jbundle.util.osgi.wrapped:org.jbundle.util.osgi.wrappe
d.org.apache.http.client:4.1.2'
    compile 'com.mcxiaoke.volley:library-aar:1.0.0'
    compile 'com.google.android.gms:play-services-
appindexing:8.4.0'
    compile 'com.github.bumptech.glide:glide:4.0.0-RC0'
    compile 'com.google.android.gms:play-services-
maps:10.2.6'
```

```
    compile 'com.android.support.constraint:constraint-
layout:1.0.0-alpha7'
    testCompile 'junit:junit:4.12'
}
```

Kode Program 5.24 *Grandle* Aplikasi

Kode Program 5.24 merupakan *grandle* dari aplikasi *mobile* SmartCar. Aplikasi Android dibangun menggunakan *Open Source Gradle Build System*. Secara *default* **Gradle** adalah *build tools* yang digunakan pada Android Studio untuk meng-*compile* menjalankan *project* aplikasi.

2. Mengijinkan Hak Akses *Internet* pada `Mainfest.java`

```
<uses-permission android:name="android.permission.INTERNET"
/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Kode Program 5.25 Kode Program `Mainfest.java` untuk Hak Akses *Internet*

Kode program 5.25 merupakan `manifest.java` dari file Android Studio, dimana memberikan *permission internet* untuk hak akses *internet* karena aplikasi menggunkan *web service* untuk *passing* data.

3. AppController.java

```
package topik.smartcar;
/**
 * Created by windows8 on 4/26/2017.
 */
import android.app.Application;
import android.text.TextUtils;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.Volley;
import com.onesignal.OneSignal;
```

```
public class AppController extends Application {
    public static final String TAG =
AppController.class.getSimpleName();
    private RequestQueue mRequestQueue;

    private static AppController mInstance;

   @Override
    public void onCreate() {
        super.onCreate();
        mInstance = this;
        super.onCreate();
        OneSignal.startInit(this)

.inFocusDisplaying(OneSignal.OSInFocusDisplayOption.Notific
ation)

.unsubscribeWhenNotificationsAreDisabled(true)
                .init();
         //String userEmail = "bayudarmawan134@gmail.com";
        // Call syncHashedEmail anywhere in your app if you
have the user's email.
        // This improves the effectiveness of OneSignal's
"best-time" notification scheduling feature.
        //OneSignal.syncHashedEmail(userEmail);
    }
    public static synchronized AppController getInstance()
{
        return mInstance;
    }
    public RequestQueue getRequestQueue() {
        if (mRequestQueue == null) {
            mRequestQueue =
Volley.newRequestQueue(getApplicationContext());
        }
        return mRequestQueue;
    }
    public <T> void addToRequestQueue(Request<T> req,
String tag) {
        req.setTag(TextUtils.isEmpty(tag) ? TAG : tag);
        getRequestQueue().add(req);
    }
    public <T> void addToRequestQueue(Request<T> req) {
```

```
        req.setTag(TAG);
        getRequestQueue().add(req);
    }
    public void cancelPendingRequests(Object tag) {
        if (mRequestQueue != null) {
            mRequestQueue.cancelAll(tag);
    }}}
```

Kode Program 5.26 AppController.java

Kode program 5.26 merupakan file pada aplikasi *mobile* yaitu AppController yang berada dalam *package app* dan tambah. Fungsi dari AppControlle**r** ini yaitu *class* tunggal yang menginisialisasi *class global* yang diperlukan.

4. AppConvig.java

```
public class AppConfig {
    //URL Login
    public static final String URL_LOGIN =
"https://topsus.digiforest.web.id/mobile/auth/sign_in?usern
ame=";
            // URL for Login GIS Project
            //"http://gis.sigjalan.com/web-services-
db.php?flag=fCekLogin&username=";

    //URL Register for GIS Project
    public static final String URL_LIHAT_INFO =
"https://topsus.digiforest.web.id/mobile/user/get_cars_info
?user_id=";
}
```

Kode Program 5.27 AppConfig.java

Kode Program 5.27 merupakan file AppConfig pada aplikasi *mobile* SmartCar. AppConfig ini sebagai data untuk *link* sumber data *web service* api yang ditampilkan pada aplikasi

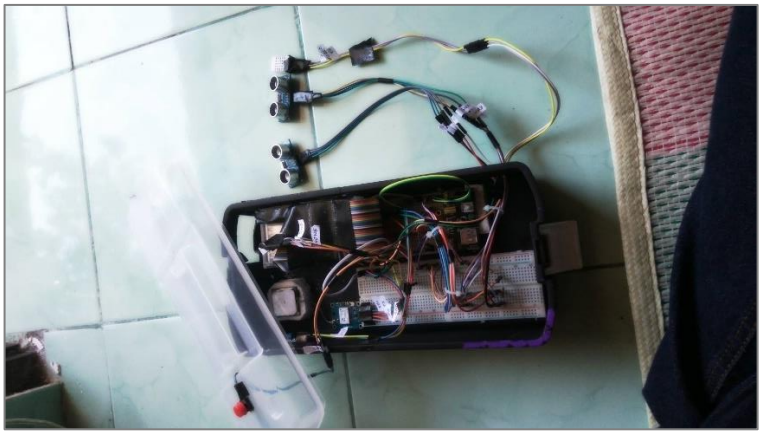
BAB 6

Aplikasi SmartCar

Rangkaian Alat & Bahan Aplikasi SmartCar
Aplikasi SmartCar

6.1 Rangkaian Alat & Bahan Aplikasi SmartCar

Rangkaian komponen aplikasi SmartCar terdapat alat dan bahan yang utama yaitu Raspberry Pi B, *Bread Board*, GPS *Module*, Kabel *Jumper* dan sensor-sensor pendukung aplikasi SmartCar.



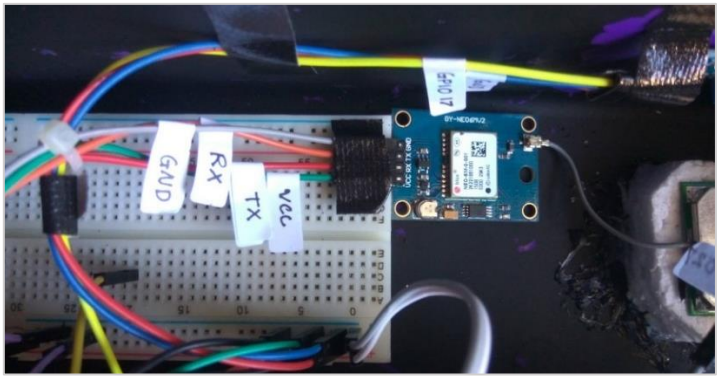
Gambar 6.1 Rangkaian Alat & Bahan Aplikasi SmartCar

Gambar 6.1 merupakan rangkaian keseluruhan dari alat dan bahan yang digunakan untuk membangun aplikasi SmartCar. Rangkaian Gambar 6.1 terdiri dari beberapa alat seperti Raspberry Pi Model B yang berfungsi sebagai sebuah komputer yang menjadi pusat berjalannya aplikasi SmartCar. *Bread Board* merupakan prototipe rancangan sebagai penghubung antara Raspberry Pi dengan sensor-sensor yang digunakan. Kabel *jumper* digunakan untuk menghubungkan ketiga alat tersebut yaitu Raspberry Pi, *Bread Board* dan Sensor. GPS *Module* digunakan sebagai *Tracker*, *antenna* untuk menangkap sinyal pada GPS, *Power Supply* untuk memberikan daya ketersediaan *power* atau daya pada alat. Rancangan aplikasi SmartCar ini

juga dilengkapi dengan sensor atau alat pendukung lainnya seperti sensor ultrasonic, sensor getaran (*vibration)*, sensor suhu dan kelembapan, tombol *(button)* dan speaker. Berikut merupakan rangkaian dari masing-masing sensor dan alat yang terdapat pada aplikasi SmartCar.

6.1.1. Rangkaian Sensor GPS *Tracking*

Sensor GPS atau GPS *Module* digunakan untuk memperoleh infomasi *longitude* dan *latitude* dari suatu tempat, mengukur kecepatan dan kondisi cuaca dari lokasi kendaraan.



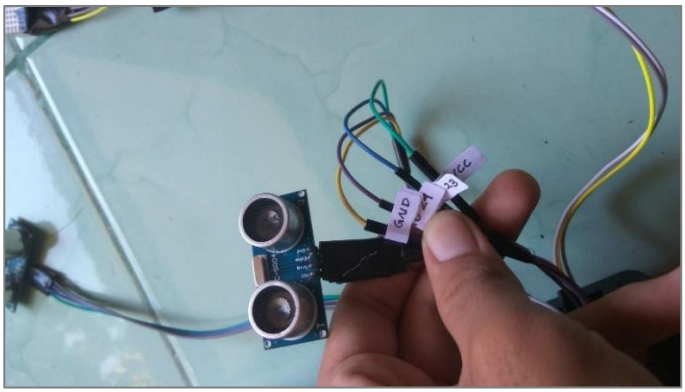
Gambar 6.2 Rangkaian Sensor GPS *Tracking*

Gambar 6.2 merupakan gambar dari rangkaian Sensor GPS *Module*. Sensor GPS berfungsi untuk memperoleh informasi mengenai *longitude* dan *latitude*, kecepatan dan cuaca pada suatu tempat. Sensor GPS menggunakan kabel *jumper* untuk mengghubungkan dengan GPS *Module*. Kabel *jumper* ini terhubung ke *Bread Board* dengan kabel VCC berwarna merah dan memberikan daya bernilai positif, GND atau *Ground* untuk memberikan daya bernilai negatif. Kabel RX berfungsi untuk mengirim dan menerima data RX

sebagai *receiver* dan kabel TX menerima data TX sebagai *transmite* untuk mengirim data. Sensor GPS *Module* juga terhubung dengan *antenna* yang berguna untuk memperoleh sinyal sehingga data yang didapatkan bisa bersifat *real-time*.

6.1.2 Rangkaian Sensor Ultrasonik

Rangkaian Sensor Ultrasonik yang digunakan untuk mengukur jarak kendaraan dengan benda lain yang berada disekitarnya dapat dilihat sebagai berikut.



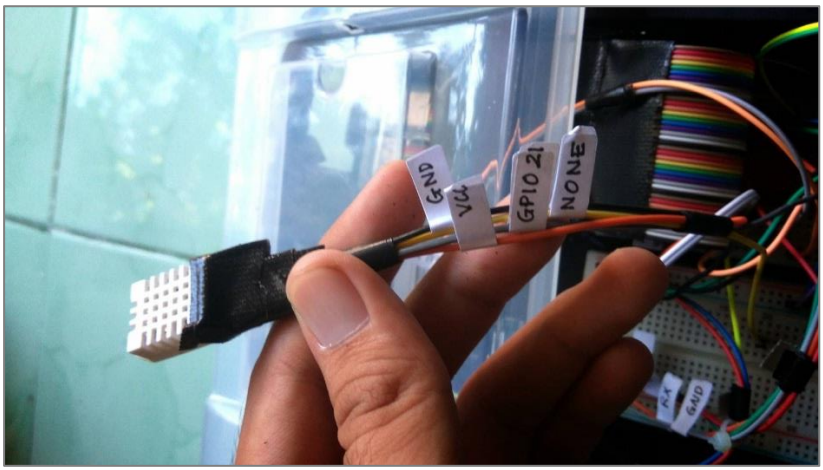
Gambar 6.3 Rangkaian Sensor Ultrasonic Aplikasi SmartCar

Gambar 6.3 merupakan rangkaian dari Sensor Ultrasonik untuk aplikasi SmartCar. Sensor Ultrasonik tersebut dihubungkan dengan kabel *jumper* yang terhubung ke Raspberry Pi dan *Bread Board*. Kabel *Jumper* VCC berwarna merah bernilai positif dan GND berwarna hitam bernilai negatif akan terhubung dengan *Bread Board* yang berfungsi untuk memperoleh

daya, Kabel *Jumper* GPIO 23 dan GPIO 24 terhubung ke Raspberry Pi yang berfungsi sebagai *input* dan *output* dari program.

6.1.3 Rangkaian Sensor Suhu dan Kelembapan

Rangkaian sensor suhu dan kelembapan terdiri dari sensor suhu dan kabel *jumper* terhubung ke Raspberry Pi dan *Bread Board*.

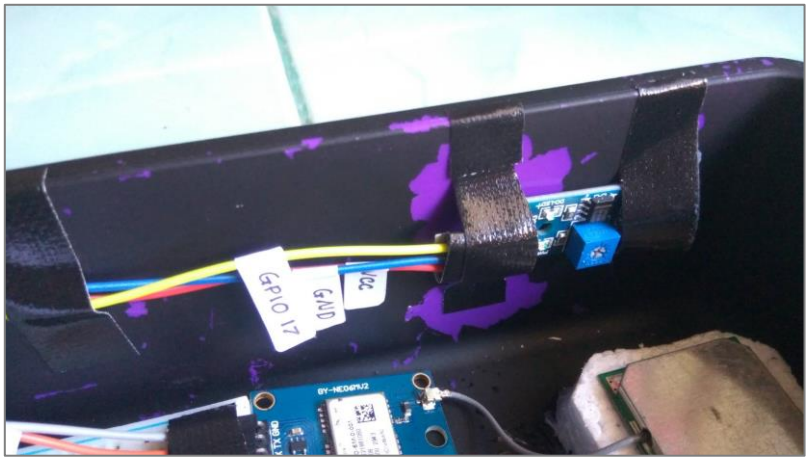


Gambar 6.4 Rangkaian Sensor Suhu Aplikasi SmartCar

Gambar 6.4 merupakan rangkaian alat dari sensor suhu yang berfungsi untuk mengukur suhu antara bagian dalam dan luar kendaraan untuk mencegah terjadinya kaca kendaraan yang pecah akibat perbedaan suhu. Sensor suhu menggunakan kabel *jumper* sebagai penghubung ke Raspberry Pi dan *Bread Board*. Kabel *Jumper* yang digunakan yaitu VCC berwarna merah yang bernilai positif dan GND berwarna hitam yang bernilai negatif, dan Kabel *Jumper* GPIO 21 untuk terhubung ke Raspberry Pi.

6.1.4 Rangkaian Sensor Getar *(Vibration)*

Sensor getaran atau *Vibration* digunakan untuk mendeteksi jika terjadinya getaran pada kendaraan. Sensor getaran berfungsi untuk mendeteksi terjadinya benturan atau tabrakan pada kendaraan.

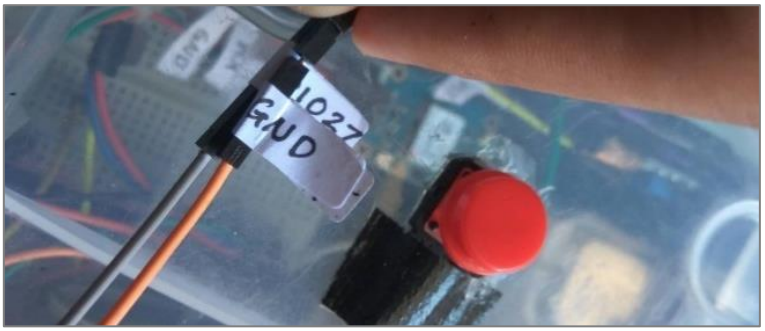


Gambar 6.5 Rangkaian Sensor *Vibration* Aplikasi SmartCar

Gambar 6.5 merupakan gambar dari rangkaian sensor getaran atau *vibration* yang berfungsi untuk mendeteksi jika terjadinya getaran atau benturan pada kendaraan. Sensor getaran dihubungkan dengan kabel *Jumper* VCC untuk memberikan daya yang bernilai positif dan GND atau Ground untuk memberikan daya yang bernilai negatif kemudian terhubung ke *Bread Board*. Kabel *Jumper* GPIO 17 akan terhubung ke Raspberry Pi.

6.1.5 Rangkaian Tombol Panik

Rangkaian Tombol Panik berfungsi sebagai tombol yang digunakan ketika pengguna mengalami keadaan darurat atau mendesak.



Gambar 6.6 Rangkaian Sensor *Panic Button* Aplikasi SmartCar

Gambar 6.6 merupakan gambar rangkaian dari fitur tombol panik dengan alur proses ketika tombol tersebut di tekan maka Raspberry Pi akan mengirimkan notifikasi ke *website* dan aplikasi SmartCar untuk mengetahui pengguna sedang dalam keadaan darurat. Tombol panik dihubungkan dengan kabel *jumper* GND atau *Ground* yang bernilai negatif untuk memberikan daya dan GPIO 27 berfungsi untuk membuat tombol terhubung dengan Raspberry Pi.

6.2 Aplikasi SmartCar

Aplikasi SmartCar dapat dijlankan pada dua jenis *platform* yaitu *mobile* Android dan *website (web service).*

6.2.1 *Mobile*

Aplikasi SmartCar yang dijalankan pada *mobile* terdiri dari beberapa tahapan, yaitu dapat dilihat sebagai berikut.

1. *Login*

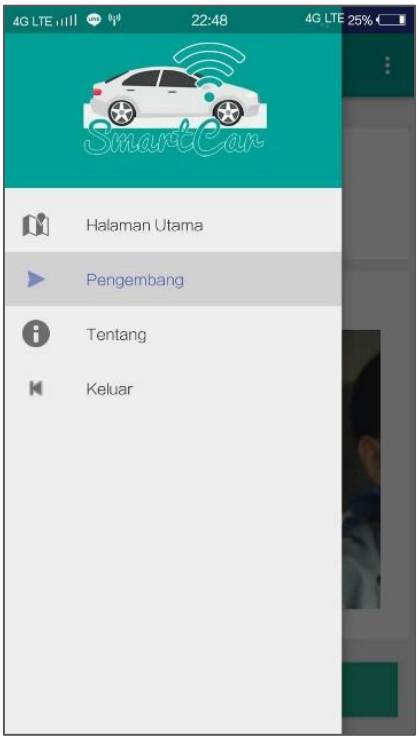
Halaman *Login* merupakan tahap pertama yang dilakukan setelah membuka aplikasi *mobile* SmartCar seperti pada Gambar 6.7.

Gambar 6.7 Halaman *Login Mobile*

Gambar 6.7 menunjukkan halaman *login* dari aplikasi *mobile* SmartCar dimana pengguna harus memasukkan *username* dan *password* yang telah terdaftar kemudian dapat melakukan *login*.

2. Menu

Terdapat tiga menu utama yang dapat digunakan dalam aplikasi *mobile* SmartCar yaitu Halaman Utama, Pengembang, dan Tentang.

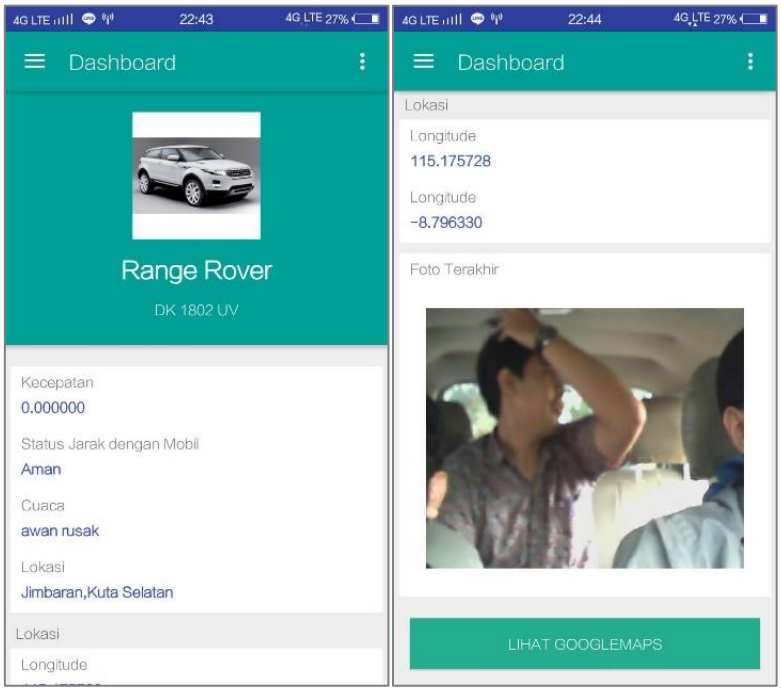


Gambar 6.8 Menu Utama

Gambar 6.8 merupakan menu utama pada aplikasi *mobile* SmartCar dengan menampilkan menu akses dari aplikasi *mobile* yang terdiri dari menu "Pengembang" adalah informasi mengenai profil pengembang yang telah membuat aplikasi SmartCar dan Menu "Tentang" merupakan deskripsi aplikasi SmartCar dan info kontak yang dapat dihubungi.

3. Halaman Utama

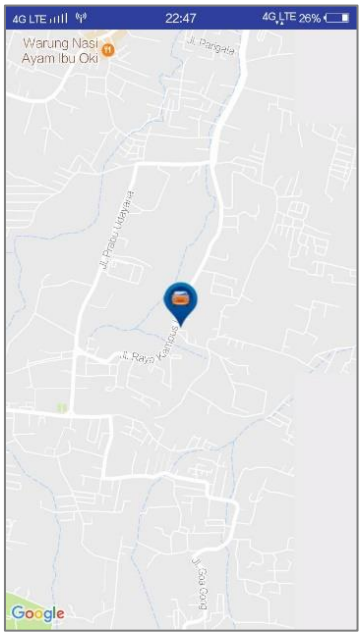
Halaman Utama merupakan halaman yang akan ditampilkan setelah pengguna melakukan *login* untuk dapat masuk ke aplikasi.



Gambar 6.9 Halaman Utama *Mobile*

Gambar 6.9 menunjukkan halaman utama dalam aplikasi *mobile* SmartCar. Informasi yang didapatkan di halaman utama ini yaitu kecepatan, status jarak dengan mobil, cuaca, lokasi, dan foto terakhir. Kecepatan akan memberikan informasi berupa kecepatan mobil yang sedang melaju dengan menampilkan angka. Status jarak dengan mobil akan memberikan informasi jarak mobil dengan mobil lain, ketika status yang diberikan adalah aman maka dapat dipastikan bahwa mobil tidak berjarak kurang dari 5 cm dengan

mobil lain dan sebaliknya jika status jarak adalah bahaya maka jarak mobil dengan mobil lain adalah lebih kecil dari 5 cm. Info cuaca akan memberikan informasi keadaan cuaca pada daerah sekeliling mobil. Info cuaca ini diukur oleh sensor suhu yang berada di dalam dan di luar mobil. Info lokasi akan memberikan informasi mengenai keberadaan mobil dengan mengambil nilai *longitude* yang di dapat dari sensor GPS. Pengguna dapat melihat keberadaan mobil dengan *map* dengan menekan tombol "LIHAT GOOGLEMAPS". Info foto terakhir merupakan foto yang dikirimkan oleh kamera yang ada di dalam mobil kemudian di kirim ke *server* setelah pengguna menekan tombol panik.



Gambar 6.10 Google Maps *Mobile*

Gambar 6.10 merupakan tampilan tombol "LIHAT GOOGLEMAPS" yang ada di halaman utama aplikasi *mobile* SmartCar. Keberadaan mobil dapat di lihat pada google maps. Informasi ini di ambil dari sensor GPS yang ada di dalam mobil.

4. Menu Tentang

Halaman "Tentang" merupakan halaman yang akan ditampilkan setelah pengguna menekan info Tentang di menu utama.

Gambar 6.11 Halaman Tentang

Menu "Tentang" merupakan informasi aplikasi *mobile* SmartCar berupa deskripsi, informasi kontak dan versi android yang digunakan didalam implementasi.

6. Menu Pengembang

Halaman Pengembang merupakan halaman yang akan ditampilkan setelah pengguna menekan info Pengembang di menu utama.

Gambar 6.12 Halaman Pengembang

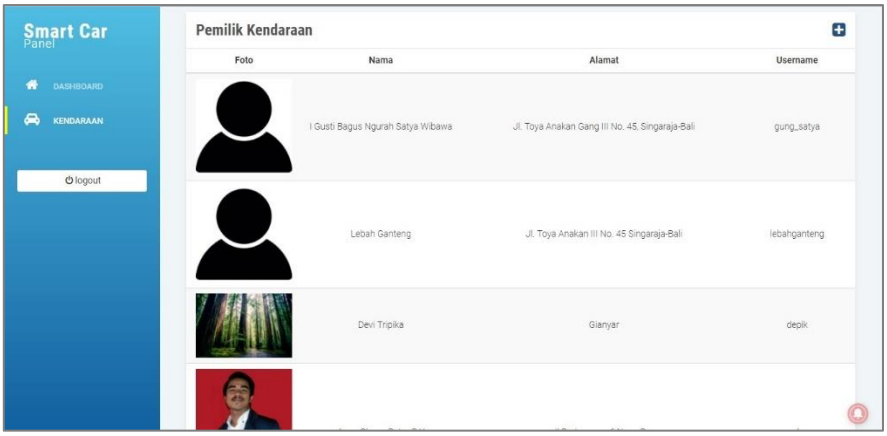
Menu "Pengembang" merupakan informasi tentang pengembang aplikasi SmartCar. Informasi yang diberikan berupa foto pengembang yang dapat di geser ke samping untuk dapat melihat pengembang lainnya.

6.2.2 *Website*

Sistem aplikasi SmartCar yang dijalankan pada *web service* terdiri dari beberapa tampilan. Tampilan dari *web service* dapat dilihat sebagai berikut.

1. Data Pemilik Kendaraan

Halaman pada *website* "Pemilik Kendaraan" yaitu menampilkan detail dari kendaraan.

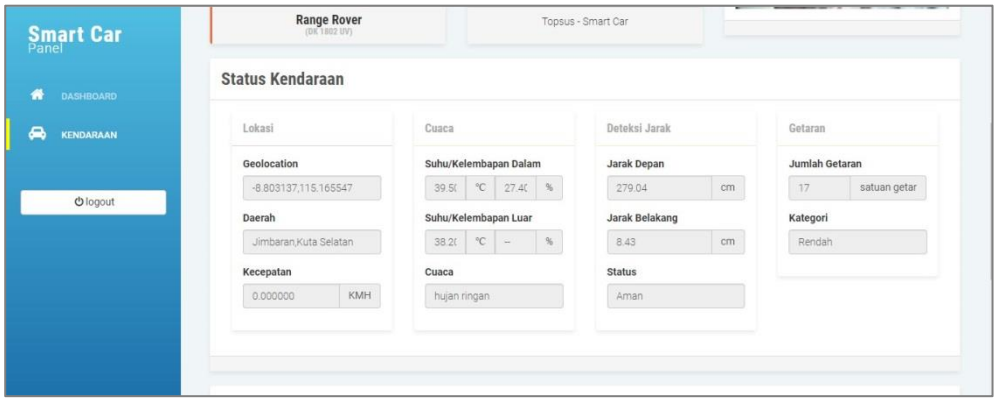


Gambar 6.13 Halaman *Website* Pemilik Kendaraan

Gambar 6.13 merupakan tampilan *website* yang disediakan untuk menampilkan data pemilik kendaraan yang terdaftar pada sistem. Sistem akan menampilkan nama, alamat dan *username* agar dapat terlihat oleh admin. Pengguna akan dapat memantau pengendara ataupun sebagai pemilik pengendara sesuai dengan ID mobil yang terdaftar pada masing-masing pengguna sistem.

2. Informasi Fitur

Halaman "Informasi Fitur" yaitu menampilkan detail informasi dari masing-masing fitur atau sensor secara *real-time*.

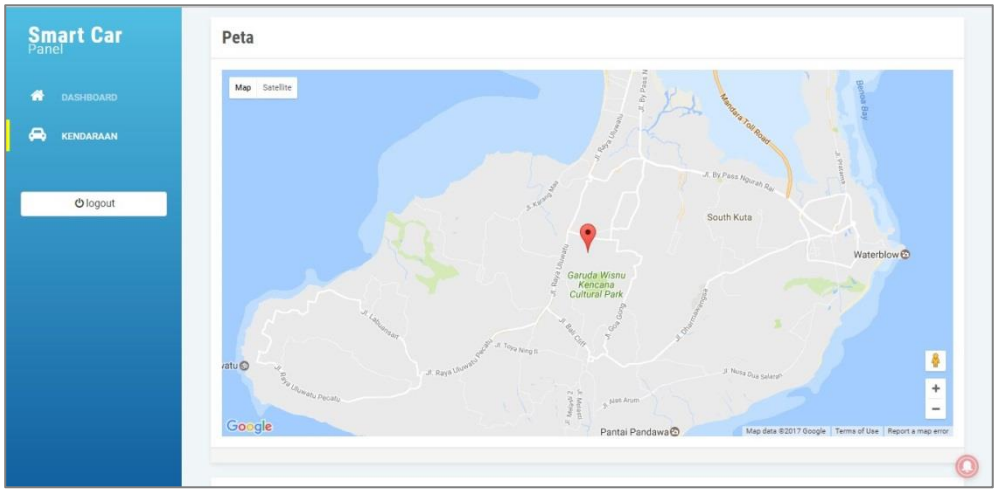


Gambar 6.14 Halaman *Website* untuk Informasi Fitur

Gambar 6.14 merupakan halaman informasi masing-masing sensor pada *website* yang digunakan untuk dapat melihat perubahan status kendaraan. Status kendaraan pada bagian lokasi menampilkan data yang masuk dalam sistem berupa *Geolocation*, Daerah dan Kecepatan. Status kendaraan juga menampilkan pada bagian Suhu Kelembapan Dalam, Suhu Kelembapan Luar dan Cuaca. Bagian status kendaraan yaitu Deteksi jarak menampilkan data berupa Jarak Depan, Jarak Belakang dan Status dari keamanan kendaraan. Bagian pada status kendaraan terakhir yaitu Getaran yang menampilkan kondisi getaran yang dialami oleh kendaraan dengan menyimpan data jumlah Satuan Getaran dan Kategori getaran.

3. Lokasi Kendaraan pada Peta

Halaman "Lokasi Kendaraan pada Peta" yaitu menampilkan detail informasi lokasi kendaraan secara *real-time*.

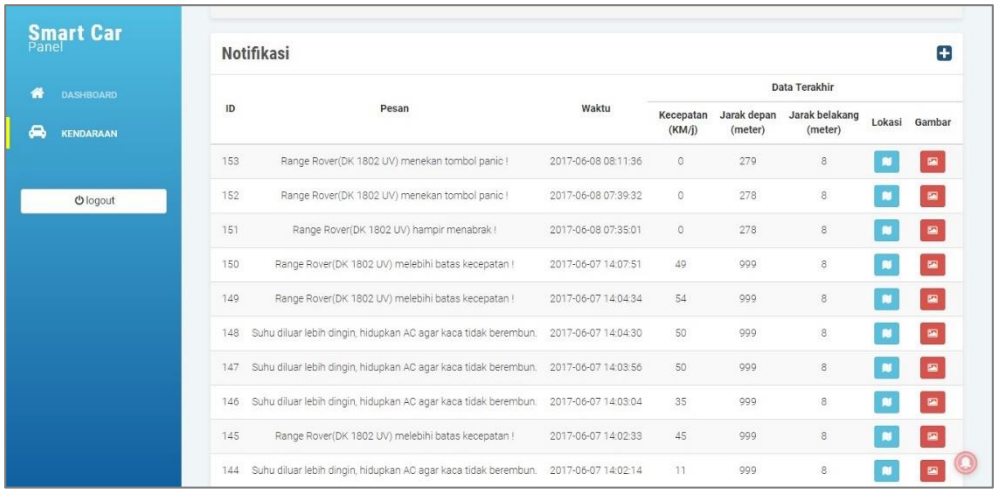


Gambar 6.15 Halaman *Website* untuk Melacak Lokasi Kendaraan pada Peta

Gambar 6.15 merupakan halaman *website* untuk melacak lokasi kendaraan pada peta. Komponen atau alat yang berfungsi untuk halaman ini yaitu sensor GPS *Module* dengan mencari jalur lokasi keberadaan kendaraan. Pengguna dengan ID kendaraan tertentu akan ditandai pada maps *website* aplikasi Smart Car.

4. *History* Notifikasi

Halaman "*History* Notifikasi" yaitu menampilkan notifikasi pada semua fitur yang tersedia pada aplikasi SmartCar.

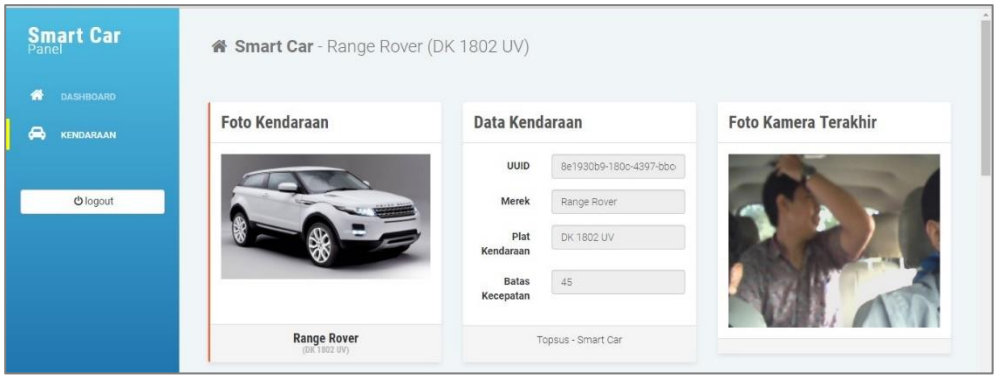


Gambar 6.16 Halaman *Website* untuk History Notifikasi

Gambar 6.16 merupakan tampilan dari halaman *website* untuk *history* notifikasi yang menampilkan pemberitahuan untuk semua fitur yang tersedia dalam aplikasi SmartCar. Notifikasi yang ditampilkan nantinya akan sesuai dengan keadaan, kejadian dan kondisi dari kendaraan. Halaman notifikasi terdiri dari bagian "ID" sebagai identitas notifikasi yang telah muncul, "Pesan" yaitu menampilkan pesan kejadian atau kondisi yang terjadi pada kendaraan, bagian "Waktu" merupakan waktu data terkirim, Kecepatan, Jarak Depan, Jarak Belakang, Lokasi dan Gambar.

5. Penerimaan Informasi Kendaraan

Halaman "Penerimaan Informasi Kendaraan" yaitu menampilkan detail informasi fitur ketika terjadi proses pada rangkaian komponen.

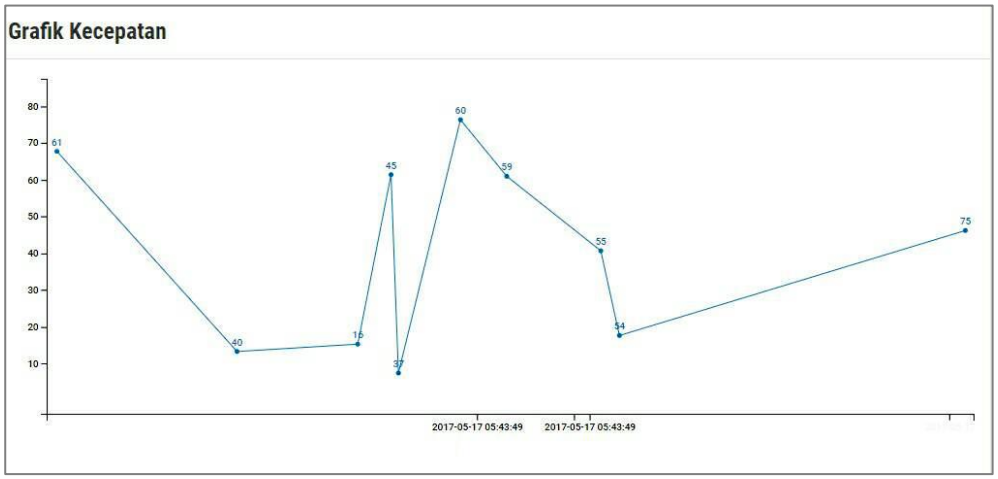


Gambar 6.17 Halaman *Website* Penerimaan Informasi Kendaraan

Gambar 6.17 Halaman penerimaan informasi kendaraan pada *website* merupakan halaman untuk menampilkan informasi berupa notifikasi sesuai dengan kondisi kendaraan. Data yang ditampilkan dalam halaman ini berupa data detail kendaraan, foto kendaraan, dan foto pengambilan kamera terakhir.

6. Grafik Kecepatan

Halaman "Grafik Kecepatan" yaitu menampilkan grafik kecepatan dari fitur *Tracking* Lokasi.



Gambar 6.18 Halaman *Website* untuk Grafik Kecepatan

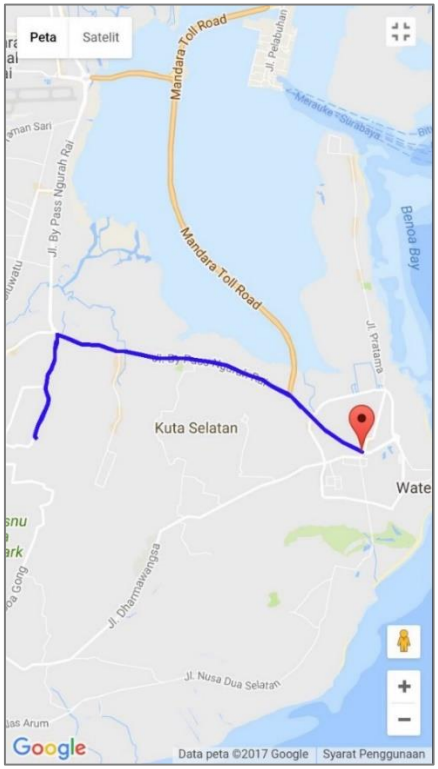
Gambar 6.18 merupakan tampilan dari halaman grafik kecepatan pada *website* dengan menampilkan dalam bentuk grafik garis dan jumlah kecepatan dengan memberi notifikasi setiap detik dari deteksi kecepatan. Halaman ini untuk mengetahui seberapa cepat kendaraan tersebut melaju.

6.2.3 Fitur Aplikasi Smart Car

Uji coba dilakukan untuk mengetahui sejauh mana fitur pada aplikasi SmartCar berfungsi. Point-point dibawah ini akan menjelaskan uji coba pada masing-masing fitur SmartCar.

1. Fitur *Tracking* Lokasi (GPS *Tracking*)

Fitur *Tracking Lokasi* menunjukkan informasi melalui *website* berupa lokasi yang telah ditempuh oleh pengguna mobil.



Gambar 6.19 *Tracking* Lokasi

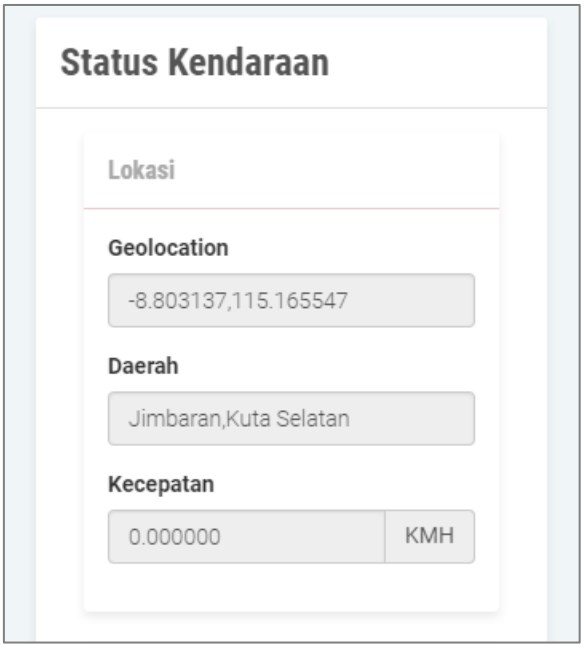
Gambar 6.19 merupakan tampilan yang menunjukan hasil dari percobaan fitur *Tracking* Lokasi. Hasil percobaan dari fitur *Tracking* Lokasi yaitu berupa garis merah yang berarti bahwa garis merah tersebut adalah lokasi yang sudah ditempuh oleh pengguna mobil.

2. Fitur Deteksi Kecepatan

Deteksi kecepatan adalah bagian dari salah satu fitur yang berfungsi untuk memberikan notifikasi kepada pengguna apabila mengendarai mobil hingga melebihi kecepatan yang sudah ditentukan.

Gambar 6.20 Sensor Kecepatan

Gambar 6.20 merupakan tampilan dari deteksi kecepatan. *Website* akan menampilkan informasi berupa kecepatan pengguna ketika mengendarai mobil tersebut serta menampilkan grafik yang menunjukan *history* dari kecepatan pengguna. Notifikasi yang ditampilkan adalah berupa suara dimana menggunakan API dari Google.



Gambar 6.21 Deteksi Kecepatan

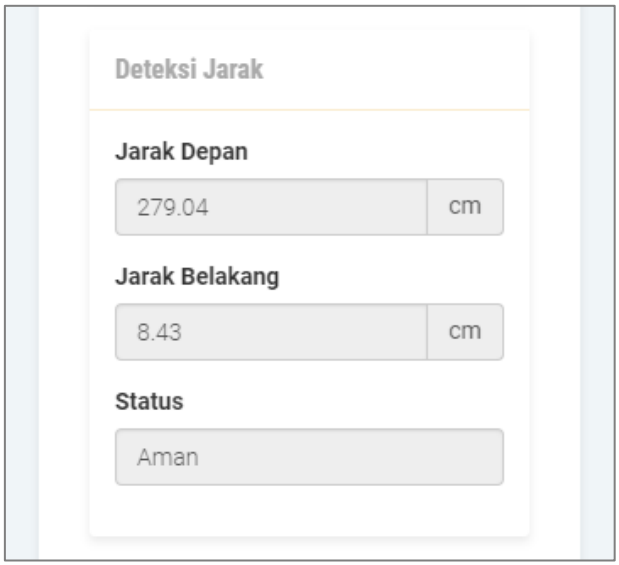
Gambar 6.21 merupakan tampilan dari deteksi kecepatan yang memberikan informasi lokasi mobil, daerah, dan kecepatan yang sedang berjalan dengan kecepatan melebihi 40km per jam maka sistem akan memberi notifikasi kepada user bahwa kecepatan berkendara melebihi batas.

3. Fitur Jarak Aman (Sensor Ultrasonik)

Sensor Ultrasonik merupakan alat yang digunakan untuk mengetahui jarak aman depan dan belakang mobil terhadap benda didekatnya dimana ketika benda dideteksi dengan jarak dibawah 5 cm maka akan di berikan notifikasi.

Gambar 6.22 Jarak pada Kendaraan dan Object Tertentu

Gambar 6.22 merupakan tampilan dari sensor ultrasonic yang diletakkan pada mobil. Tampilan pada Gambar 6.22 diasumsikan di belakang mobil tersebut terdapat penghalang atau benda yang jaraknya kurang dari 5 cm dan tidak kurang dari 2 cm serta pada bagian depan mobil tidak ada benda yang menghalangi mobil.



Gambar 6.23 Fitur Jarak Aman (Sensor Ultrasonik)

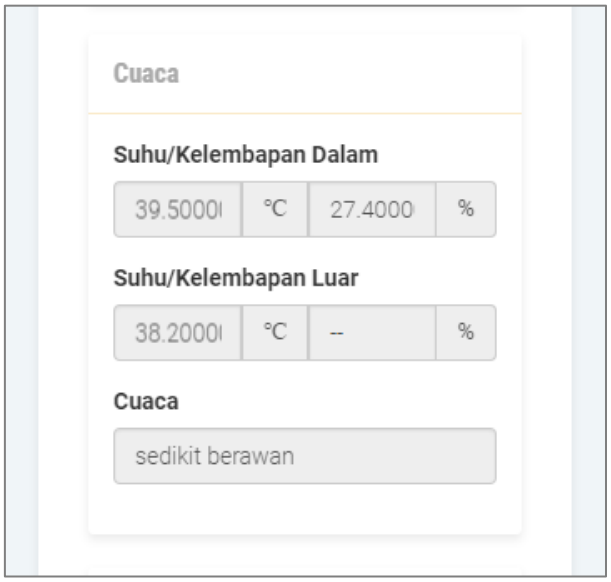
Gambar 6.23 merupakan tampilan fitu jarak aman menggunakan sensor ultrasonik. Hasil yang didapat pada sensor ultrasonik yaitu jarak benda yang ada di depan mobil dan di belakang mobil. Bagian mobil pada jarak depan tidak ada penghalang yang terdapat pada jangkauan di antara 2 cm hingga 400 cm maka akan menampilkan hasil yaitu 999, sedangkan pada sensor ultrasonik pada bagian belakang mobil mendapatkan hasil sebanyak 8,43 cm.

4. Fitur Suhu dan Kelembapan (Sensor Suhu)

Fitur suhu dan kelembapan merupakan fitur yang digunakan untuk memberikan notifikasi berupa suara kepada pengguna apabila suhu diluar mobil lebih tinggi daripada suhu yang ada di dalam mobil.

Gambar 6.24 Hasil Percobaan Sensor Suhu

Gambar 6.24 merupakan tampilan dari hasil sensor suhu. Sensor suhu dipasang pada bagian luar mobil. Sensor suhu memberikan informasi berupa tinggi suhu yang ada di luar mobil dan juga suhu yang ada di dalam mobil pada *website*. Informasi suhu yang ditampilkan memiliki satuan derajat celcius.



Gambar 6.25 Deteksi Suhu

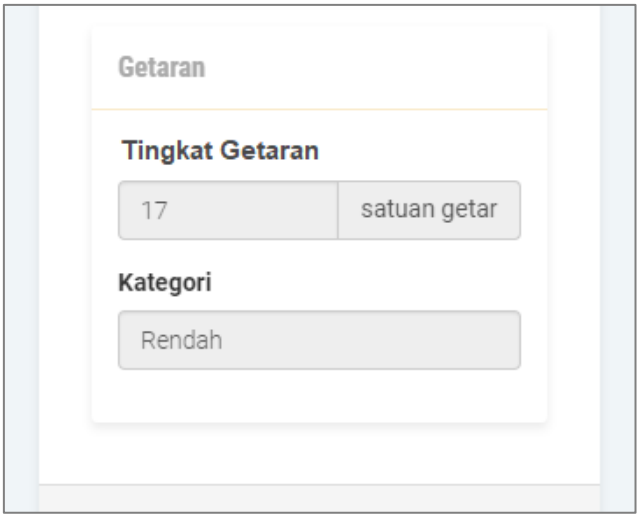
Gambar 6.25 merupakan tampilan dari deteksi suhu dimana terdapat informasi suhu/kelembapan didalam dan luar mobil dimana ketika suhu didalam mobil lebih rendah dari luar mobil maka akan terdapat notifikasi untuk menghidupkan AC agar kaca tidak berembun. Suara yang dikeluarkan apabila suhu diluar mobil lebih tinggi daripada suhu yang ada di dalam mobil yaitu ”Suhu diluar mobil lebih dingin harap hidupkan AC karena dapat membuat kaca ber embun”. Sensor juga memberikan informasi bahwa cuaca yang sedang terjadi adalah awan terpecah.

5. Fitur Deteksi Getaran (Sensor *Vibration*)

Fitur deteksi getaran merupakan proses pengujian dari komponen yang digunakan untuk mendeteksi getaran. Hasil uji deteksi getaran ini akan menampilkan jumlah getaran dan kategori getaran.

Gambar 6.26 Sensor Getaran pada Kendaraan

Gambar 6.26 merupakan tampilan dari sensor getaran pada kendaraan. Percobaan diasumsikan mobil sedang melewati penghalang berupa polisi tidur yang akan menghasilkan getaran yang akan terdeteksi oleh sensor



Gambar 6.27 Hasil Sensor Getaran

Gambar 6.27 merupakan tampilan dari hasil sensor getaran. Hasil yang didapat pada sensor getaran yaitu kuatnya getaran yang diterima oleh kendaraan. Kondisi ketika kendaraan melewati polisi tidur atau jalan yang rusak maka sensor akan mendeteksi jenis getaran yang dialami oleh kendaraan. Sensor akan mengirim data ke Raspberry Pi dan melanjutkan untuk memberi notifikasi pada *website* dan *mobile*.

6. Fitur Tombol Panik

Fitur tombol panik bekerja apabila tombol tersebut ditekan oleh pengguna. Proses yang terjadi yaitu setelah tombol panik ditekan maka kamera mengambil foto kemudian data-data tersebut diolah oleh Raspberry Pi 3, setelah itu data tersebut dikirim ke *server* dan kamera langsung mengambil foto terakhir. Proses terakhir dengan memberikan notifikasi berupa foto, lokasi dan kecepatan mobil yang digunakan oleh pengguna kepada pemantau.

Gambar 6.28 Uji Coba Tekan Tombol Panik

Gambar 6.28 merupakan tampilan dari proses saat menekan tombol panik pada rangkaian alat. Setelah menekan tombol panik maka data tersebut akan dikirim pada *website* dan *mobile* aplikasi SmartCar.

143	Range Rover(DK 1802 UV) menekan tombol panik	2017-06-07 14:02:00	12	999	8

Gambar 6.29 Notifikasi Fitur Tombol Panik pada *Web Service*

Gambar 6.29 menunjukan hasil dari fitur tombol panik dimana terdapat notifikasi pada *website* bahwa mobil "Range Rover" menekan tombol panik.

Gambar 6.30 Hasil Tombol Panik

Gambar 6.30 menunjukkan foto dari hasil dari tombol panik. Percobaan yang terjadi setelah tombol panik ditekan adalah sistem langsung mengirim pada pengguna dan mengeluarkan *output* berupa suara.

7. Fitur Promo

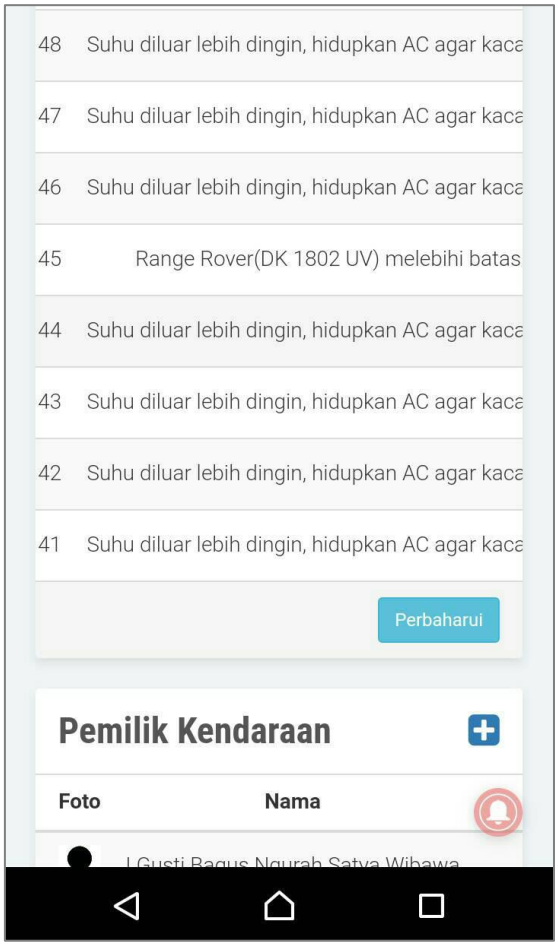
Fitur promosi merupakan fitur yang memberikan notifikasi berupa suara kepada pengguna apabila berada pada wilayah yang terkena radius dari lokasi promo, apabila lokasi dari pengguna tidak berada dalam radius dari lokasi promo maka notifikasi yang diberikan adalah lokasi berupa suara.

8. Notifikasi

Speaker pada aplikasi *Smart Car* berfungsi sebagai media *output* untuk notifikasi wilayah dan notifikasi peringatan untuk pengguna *Smart Car*. *Speaker* dipasang ditengah mobil didekat pengemudi agar suara yang dihasilkan dapat terdengar jelas. Implementasi *Speaker* menggunakan metode *Text To Speech*, pemberitahuan dari *Smart Car* akan dikonversi menjadi *file* suara yang nantinya akan dijalankan *Smart Car*.

Gambar 6.31 *Speaker* pada Kendaraan

Gambar 6.31 merupakan tampilan dari *speaker* yang digunakan untuk memberi informasi pada aplikasi *Smart Car* pada mobil. *Speaker* akan menghasilkan *output* berupa suara dengan memberikan notifikasi pada pengguna mobil.



Gambar 6.32 Notifikasi Aplikasi SmartCar

Gambar 6.32 merupakan notifikasi yang menampilkan informasi semua fitur yang ada dalam aplikasi SmartCar. Notifikasi aplikasi ini dapat diperbaharui setiap saat oleh pengguna.

9. Tampilan Fisik Rangkaian Aplikasi SmartCar

Tampilan fisik rangkaian aplikasi SmartCar berbentuk seperti kotak *tissue*, dimana didalam kotak tersebut terdapat komponen-komponen yang menjalankan alat tersebut seperti Raspberry Pi, sensor untuk masing-masing fitur, kabel *jumper*, *breadboard* dan lain-lain. Hasil dokumentasi untuk tampilan rangkaian SmartCar secara fisik terdapat pada Gambar 6.33.

IOT FOR BEGINNER | SMART CAR

Gambar 6.33 Tampilan Fisik Rangkaian SmartCar

DAFTAR PUSTAKA

Anonim (2015). "Internet of Things, Ketika Dunia Kita Terkoneksi". http://indomedia.com.au/internet-of-things-ketika-dunia-kita-terkoneksi/

Anonim (2017). "Internet of Things, Sejarah, Teknologi, dan Penerapannya: Review". http://docplayer.info/34022090-Internet-of-things-sejarah-teknologi-dan-penerapannya-review.html

Anonim (2016). "Fungsi Dari Middleware yang Harus Kamu Ketahui". https://www.lintaspintas.com/2016/06/21/fungsi-dari-middleware-yang-harus-kamu-ketahui/

Arie (2016). "Pengertian Internet of Things dan implementasi IoT". https://www.tembolok.id/pengertian-internet-of-things-implementasi-dan-contoh-perangkat-iot/

Andre, Julfikar Ali (2016). "Sistem Security Webcam Dengan Menggunakan Microsoft Visual Basic (6.0)".

Damanik. Internet of Things. http://www.academia.edu/30171064/Internet_Of_Things_IoT

Darma Putra (2006). "Pengolahan Citra Digital. Penerbit Andi, Yogyakarta".

David Wong (2016). "Cara Kerja Internet of Things". http://www.progresstech.co.id/blog/internet-of-things/

Dirgantara, Made (2012). "Sensor Kelembaban". https://www.scribd.com/doc/97160205/Tugas-Sensor-Kelembaban.

DosenIT (2016). "Fungsi Application Layer dan Protokol yang Bekerja pada Application Layer". http://dosenit.com/jaringan-komputer/teknologi-jaringan/fungsi-application-layer

Fitriana (2011). "RFID (Radio Frequency Identification)". http://terminaltechno.blog.uns.ac.id/2011/03/13/rfid-radio-frequency-identification/

Hanu, Agung, Umar. "Implementasi Aplikasi Remote Lock Door Berbasis Raspberry Pi Sebagai Sub Sistem Kunci Otomatis Untuk Ruang Dosen Telkom University. Bandung".

Helen (2015). "Pengertian dan Penggunaan Internet Protokol". http://www.helenturvey.com/pengertian-dan-penggunaan-internet-protokol/

Indra (2015). Pengertian Wifi Dan Fungsinya. http://www.indraservicelaptop.com/2014/04/pengertian-wifi-dan-fungsinya.html

Kemasan. "Mengenal Apa Itu Barcode". http://www.galerikemasan.com/mengenal-apa-itu-barcode

Kho, Dickson (2014). "Pengertian speaker dan prinsip kerjanya" http://teknikelektronika.com/fungsi-pengertian-speaker-prinsip-kerja-speaker/#comment-154

Kominfo (2015). "Persyaratan Teknis Perangkat Near Field Communication". https://web.kominfo.go.id/sites/default/files/users/1536/DRAFT%20RPM%20NFC%20KONSULTASI%20%20PUBLIK.pdf

Prasada (2017). "Kelebihan dan Kekurangan Internet of things (IoT)". https://student.unud.ac.id/prasadab/news/32428

Prapanca, Aditya (2015). "Raspberry Pi". http://if.unesa.ac.id/blog/aditya/2015/11/18/198/

Rafiuddin Syam (2013). "Dasar Dasar Teknik Sensor".

Rahayu, Fitri (2015). "Jaringan Wireless". http://www.academia.edu/19133911/Jaringan_weriless

Robot Edukasi (2016). "Mengenal Papan Proyek (ProjectBoard)". http://www.robotedukasi.com/mengenal-papan-proyek-projectboard/

Santosa, Hari. "Cara Kerja Sensor Ultrasonik, Rangkaian, & Aplikasinya".http://www.elangsakti.com/2015/05/sensor-ultrasonik.html

Tandy (2012). "Sejarah Raspberry Pi". http://hmte.ft.uns.ac.id/artikel/sejarah-raspberry-pi/

Wibawa Putra, Adhitya (2015). "Internet of Things-Era Baru Semua Benda Dikendalikan Melalui Jaringan Internet." https://teknojurnal.com/internet-of-things-era-baru-semua-benda-dikendalikan-melalui-jaringan-internet/

Wijaya, Ketut Krisna (2016). "Harga Sama, Raspberry Pi 3 Kini Dilengkapi dengan Wi-Fi dan Bluetooth". https://id.techinasia.com/versi-baru-raspberry-pi-3-wi-fi-bluetooth

Winardi (2017). Mengenal Teknologi ZigBee Sebagai Standart Pengiriman Data Secara Wireless. http://comp-eng.binus.ac.id/files/2012/06/Mengenal-Teknologi-ZigBee-Sebagai-Standart-Pengiriman-Data-Secara-Wireless-Winardi.pdf

Zona Elektro (2014). "Resistor, Karakteristik, Nilai Dan Fungsinya". http://zonaelektro.net/resistor-karakteristik-nilai-dan-fungsinya/

Dewasa ini peran internet dalam kehidupan masyarakat modern menjadi sesuatu yang tidak dapat terpisahkan sehingga lahirnya konsep mengenai Internet of Things (IoT). IOT merupakan sebuah konsep dimana suatu objek fisk dapat terhubung langsung ke internet. SmartCar merupakan salah satu penerapan dari konsep Internet of Things. SmartCar adalah aplikasi untuk kendaraan roda empat yang berfungsi melacak lokasi kendaraan, notifikasi kecepatan, mendeteksi jika terjadi guncangan atau benturan, notifikasi jika ada objek yang terlalu dekat, deteksi perbedaan suhu diluar dan dalam kendaraan, ramalan cuaca, notifikasi promo dan panic button.

Buku ini berisikan pemahaman mengenai Internet of Things yang terdiri dari teori – teori mengenai Internet of Things dan implementasi Interenet of Things dalam aplikasi SmartCar yaitu alat dan bahan, bahasa pemrograman dan uji coba. Sensor yang digunakan pada SmartCar diantaranya ultrasonik, suhu dan temperatur, getaran, dan GPS module.

IOT FOR BEGINNER: SMART CAR

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK UNIVERSITAS UDAYANA