TUGAS AKHIR – TE 141599

SISTEM NAVIGASI DAN PENGHINDAR RINTANGAN PADA MOBILE ROBOT MENGGUNAKAN GPS DAN PENGUKUR JARAK ULTRASONIK

Lukas Yulianto
NRP 2212105012

Dosen Pembimbing
Rudy Dikairono, ST., MT.
Dr. Tri Arief Sardjono, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – TE 141599

NAVIGATION AND OBSTACLE AVOIDANCE SYSTEM ON A MOBILE ROBOT USING GPS AND ULTRASONIC RANGE FINDER

Lukas Yulianto
NRP 2212105012

Supervisor
Rudy Dikairono, ST., MT.
Dr. Tri Arief Sardjono, ST., MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

SISTEM NAVIGASI DAN PENGHINDAR RINTANGAN PADA MOBILE ROBOT MENGGUNAKAN GPS DAN PENGUKUR JARAK ULTRASONIK

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I,	Dosen Pembimbing II,
Rudy Dikairono, ST., MT.	Dr. Tri Arief Sardjono, ST., MT.
NIP: 198103252005011002	NIP: 197002121995121001

SURABAYA
JANUARI, 2015

SISTEM NAVIGASI DAN PENGHINDAR RINTANGAN PADA MOBILE ROBOT MENGGUNAKAN GPS DAN PENGUKUR JARAK ULTRASONIK

Nama : Lukas Yulianto
Dosen Pembimbing : Rudy Dikairono, ST., MT
Dr. Tri Arief Sardjono, ST., MT

ABSTRAK

Seiring perkembangan teknologi, teknologi robot juga ikut berkembang. Salah satu perkembangan dalam teknologi robot adalah navigasi *mobile* robot. Sistem navigasi *mobile* robot ini memanfaatkan GPS dan kompas pada android sebagai penentu posisi dan arah, serta ultrasonik sebagai sensor pendeteksi rintangan. Android dihubungkan dengan mikrokontroler sebagai penghasil sinyal kontrol pada *mobile* robot menggunakan sistem komunikasi *bluetooth*. Data dikirimkan melalui *Bluetooth* untuk dikonversi menjadi suatu sinyal kontrol kemudi pada motor. Dengan adanya sistem ini menghasilkan suatu *mobile* robot yang dapat bergerak secara otomatis menuju titik yang telah ditentukan oleh *user*. Data kemudi yang diberikan adalah belok kanan, belok kiri dan maju lurus dengan dua level kecepatan yakni lambat dan cepat. Namun, ketika ultrasonik mendeteksi rintangan, maka perintah dari android akan diabaikan sementara dan laju mobile robot mengikuti perintah kemudi dari hasil pembacaan ultrasonik. Hasil dari 10 kali pengujian tanpa rintangan menunjukkan proses navigasi *mobile* robot mencapai tingkat keberhasilan 60% untuk radius target 5 meter, dan 70% untuk radius target 8 meter. Sedangkan pada pengujian navigasi dengan rintangan mencapai tingkat keberhasilan 50% untuk panjang rintangan 100 cm dengan radius target 5 meter dan 60% untuk panjang rintangan 50cm dengan radius target 8 meter.

Kata kunci : Android, *Bluetooth*, GPS, Navigasi.

 # Halaman ini sengaja dikosongkan #

NAVIGATION AND OBSTACLE AVOIDANCE SYSTEM ON A MOBILE ROBOT USING GPS AND ULTRASONIC RANGE FINDER

Name : Lukas Yulianto
Supervisor : Rudy Dikairono, ST., MT
Dr. Tri Arief Sardjono, ST., MT

ABSTRACT

Along with the development of technology, robot technology also developed. One of the developments in robot technology is mobile robot navigation. The mobile robot navigation system using GPS and compass on android as positioning and direction, as well as the ultrasonic obstacle detection sensors. Android is connected to the microcontroller as the control signal generator on the mobile robot using bluetooth communication system . Data is sent via Bluetooth to be converted into a steering control signal to the motor. With this system resulted in a mobile robot that can move automatically to the point specified by the user . Data supplied steering is turn right , turn left and straight forward with two levels namely slow and fast speeds. However, when the ultrasonic detecting obstacles, then the command of android will be ignored while and rate steering mobile robot following orders from the ultrasonic readings. The results of testing 10 times without hindrance shows a mobile robot navigation process achieved a success rate of 60 % for the 5 meter radius of the target , and 70 % for the target radius of 8 meters. While the navigation test with obstacles achieved a success rate of 50 % to 100 cm long hurdle with a target radius of 5 meters and 60 % for the length 50cm hurdles with a target radius of 8 meters .

Keywords : Android , Bluetooth , GPS , Navigation .

 # Halaman ini sengaja dikosongkan #

KATA PENGANTAR

Puji syukur penulis haturkan kepada Tuhan Yesus Kristus atas segala cinta kasih yang telah diberikan kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul :

SISTEM NAVIGASI DAN PENGHINDAR RINTANGAN PADA MOBILE ROBOT MENGGUNAKAN GPS DAN PENGUKUR JARAK ULTRASONIK

Tugas Akhir ini ditujukan untuk memenuhi salah satu persyaratan dalam menyelesaikan Studi Strata-1 di Jurusan Teknik Elektro, ITS Surabaya. Penulis menyadari bahwa dalam penyelesaian pengerjaan Tugas Akhir ini tidak terlepas dari bantuan dan dukungan dari berbagai pihak. Untuk itu dengan segala hormat, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah berperan serta dalam penyelesaian Tugas Akhir ini dari awal sampai akhir, khususnya pada dosen pembimbing, Rudy Dikairono, ST., MT. dan Dr. Tri Arief Sardjono, ST., MT. di Teknik Elektro ITS, Orang tua tercinta dan seluruh keluarga yang memberikan dukungan penuh kepada penulis serta kepada semua pihak yang telah membantu dalam proses penyelesaian Tugas Akhir ini.

Penulis menyadari masih terdapat kekurangan pada Tugas Akhir ini. Semoga Tugas Akhir ini dapat bermanfaat dalam pengembangan ilmu pengetahuan dan teknologi di kemudian hari.

Surabaya, Januari 2015

Penulis

 # Halaman ini sengaja dikosongkan #

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

 # Halaman ini sengaja dikosongkan #

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Saat ini teknologi *mobile phone* semakin berkembang, salah satu diantaranya yaitu OS Android, banyak *Developer* yang mengembangkan aplikasinya pada OS ini. Android menyediakan *platform* terbuka bagi para pengembang, sehingga pengguna dapat membuat aplikasi baru didalamnya tanpa harus membayar lisensi apapun. Seiring dengan perkembangan teknologi yang semakin maju, teknologi robot pun digunakan sebagai alat bantu manusia yang memiliki kelebihan dan akan terus berkembang. Salah satu perkembangan dalam teknologi robot adalah robot navigasi. Sistem navigasi digunakan dalam penunjuk posisi dan penuntun bagi *mobile* robot, sistem navigasi memiliki peran yang sangat penting pada sistem *mobile* robot. Sistem ini menggunakan GPS (*Global Positioning System*) dan Kompas (*Magnetometer*) sebagai penentu posisi dan arah. Dalam hal ini GPS digunakan sebagai penentu posisi dari *user*, sedangkan Kompas digunakan untuk menentukan arah robot dengan prinsip pembacaan medan magnet bumi. Dengan adanya *smartphone* berbasis OS Android dapat memanfaatkan sensor-sensor yang digunakan untuk navigasi yakni GPS dan kompas (*magnetometer*), sehingga dengan demikian modul GPS dan sensor kompas yang biasanya terpisah dengan harga yang relatif mahal kini dapat direalisasikan dengan menggunakan *smartphone* berbasis OS Android. Pada sistem ini juga akan memanfaatkan ultrasonik yang difungsikan sebagai sensor pendeteksi rintangan didepan, kanan, dan kiri robot. Sehingga robot dapat bergerak menentukan arah untuk menghindari rintangan.

Dari latar belakang ini muncul suatu gagasan untuk mengembangkan suatu ”Sistem Navigasi dan Penghindar Rintangan Pada *Mobile* Robot Menggunakan GPS dan Pengukur Jarak Ultrasonik”. Pada perancangan ini memanfaatkan GPS dan Kompas dari *smartphone* OS Android sebagai sensor pada Navigasi *mobile* robot, serta memanfaatkan sensor ultrasonik sebagai sensor untuk mendeteksi rintangan. *Smartphone* Android dihubungkan dengan mikrokontroler sebagai penghasil sinyal kontrol pada *mobile* robot. Sistem komunikasi antara android dan mikrokontroler dilakukan dengan memanfaatkan

Bluetooth. Data dikirimkan melalui *Bluetooth* untuk dikonversi menjadi suatu sinyal kontrol kemudi pada motor, data kemudi yang diberikan diantaranya adalah maju lurus, belok kanan, belok kiri dan berhenti.

1.2 Perumusan Masalah

Yang menjadi rumusan masalah dalam tugas akhir ini adalah :

a) Bagaimana Sistem komunikasi *bluetooth* antara *device* android dengan mikrokontroller.
b) Bagaimana robot dapat bergerak mencapai target tanpa rintangan.
c) Bagaimana robot dapat bergerak mencapai target dengan adanya rintangan.
d) Bagaimana robot dapat menghindari rintangan yang menghalangi laju robot.

1.3 Batasan Masalah

Batasan masalah dari Tugas Akhir ini adalah :

a) Sensor yang diakses pada android adalah GPS dan Kompas.
b) Sensor yang digunakan sebagai pendeteksi rintangan menggunakan Ultrasonik HC-SR04 dengan radius pembacaan 3-300cm.
c) Komunikasi antara mikrokontroler dan android memanfaatkan bleutooth pada *device* Android dan modul *bluetooth* HC-05.
d) Navigasi yang digunakan adalah *point to point* dengan perintah kemudi lurus, belok kanan, belok kiri,dan berhenti.
e) Navigasi dilakukan pada area terbuka atau diluar gedung.

1.4 Tujuan Penelitian

Tujuan dari tugas akhir ini adalah membuat sebuah modul navigasi yang terintegrasi dengan *smartphone* Android. Serta adanya penambahan sensor ultrasonik untuk mendeteksi rintangan yang menghalangi laju *mobile* robot. Dengan demikian diharapkan akan didapat suatu sistem navigasi *mobile* robot yang murah, handal dan kaya akan fitur.

1.5 Metodologi Penelitian

Metodologi yang digunakan pada penulisan Tugas Akhir ini adalah:

a) Studi Literatur.
Pada tahap ini dilakukan pengumpulan berbagai referensi mengenai hal yang mendukung penelitian tugas akhir ini. Referensi ini dapat diambil dari buku, jurnal, artikel maupun melalui media internet.

b) Perancangan Instrumentasi
Pada tahap ini dilakukan perancangan dari sistem *mobile* robot, seperti akses GPS dan kompas pada android, akses ultrasonik, motor dan mikrokontroler.

c) Eksperimen
Pada tahap eksperimen ini dilakukan proses uji coba pada sistem yang meliputi uji coba penggunaan GPS dan kompas pada android, serta uji coba ultrasonik, mikrokontroler dan motor.

d) Perancangan *Software*
Pada tahap ini dilakukan perancangan *software* pada *smartphone* Android untuk dapat melakukan komunikasi antara Android dan mikrokontroler serta mampu memberikan data GPS dan Kompas kepada mikrokontroler. Perancangan *Software* ini meliputi perancangan *software* pada *device* Android dan mikrokontroler.

e) Perancangan *Hardware*
Pada tahap perancangan *Hardware* ini terdiri atas sebuah *board* mikrokontroler ATMega328 dan sebuah *mobile* robot.

f) Pengujian Sistem
Tahap pengujian sistem ini dilakukan dengan mengkoneksikan *board* mikrokontrolller dengan *device* android. Sistem komunikasi antara mikrokontroler dengan Android menggunakan *Bluetooth.* Sedangkan ultrasonik akan difungsikan sebagai sensor yang akan mendeteksi rintangan yang menghalangi laju robot. Pada pengujian navigasi ini, *mobile* robot diperintahkan untuk bergerak menuju satu posisi yang sudah diinputkan koordinatnya oleh *user*. Ketika robot dapat berpindah tempat sesuai arah yang ditunjukan pada koordinat serta mampu mendeteksi rintangan dan melewati rintangan tersebut, maka pengujian sistem telah tercapai.

g) Penulisan buku Tugas Akhir
Penulisan buku Tugas Akhir akan mengacu pada hasil perancangan serta realisasi dari alat.

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari 5 bab dengan sistematika penulisan sebagai berikut :

- Bab 1 : Pendahuluan
  Pada bab 1 meliputi latar belakang, perumusan masalah, tujuan, s istemaika penulisan, metodologi dan relevansi.
- Bab 2 : Teori Penunjang
  Pada bab 2 menjelaskan tentang dasar-dasar teori yang dibutuhkan dalam pengerjaan tugas akhir ini. Beberapa teori yang akan dibahas pada laporan tugas akhir ini diantaranya adalah tentang GPS dan kompas sistem navigasi, board mikrokontroler, ultrasonik, komunikasi *Bluetooth* dan *smartphone* android.
- Bab 3 : Perancangan Alat
  Pada bab 3 dijelaskan tentang perencanaan sistem baik perangkat keras (*hardware*) maupun perangkat lunak (*Software*) untuk sistem navigasi pada *mobile* robot.
- Bab 4 : Pengujian Alat
  Pada bab 4 dijelaskan tentang hasil yang didapat dari pengujian system, serta hasil analisa sistem tersebut.
- Bab 5 : Penutup
  Pada Bab 5 menjelaskan tentang hasil kesimpulan dari hasil kerja alat menurut analisa yang meliputi kekurangan kekurangan pada alat, serta saran untuk pengembangan alat ke depan.

1.7 Relevansi

Hasil dari tugas akhir ini dapat memberikan solusi pada sistem navigasi *mobile* robot yang lebih handal, murah dan kaya akan fitur dengan memanfaatkan sensor GPS dan Kompas yang terdapat pada *smartphone* Android. Serta dapat memberikan sistem navigasi yang akurat karena adanya sensor ultrasonik yang mampu mendeteksi rintangan yang nantinya dapat memberikan perintah pada *mobile* robot untuk menghindari rintangan tersebut.

BAB II

TINJAUAN PUSTAKA

2.1 Sejarah Android

Android merupakan *subset* perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi *Software Development Kit* dan aplikasi inti yang di *release* oleh Google. Sedangkan Android SDK (*Software Development Kit*) menyediakan *Tools* dan API yang diperlukan untuk mengembangkan aplikasi pada *platform* Android dengan menggunakan bahasa pemrograman Java. Dikembangkan bersama antara Google, HTC, Intel, Motorola, Qualcomm, T-*Mobile*, NVIDIA yang tergabung dalam OHA (*Open Handset Alliance*) dengan tujuan membuat standar terbuka untuk perangkat bergerak(*mobile device*).

Gambar 2.1 Lambang Sistem operasi Android

Pada tahun 2005 Google mengakuisisi Android Inc yang pada saat itu dimotori oleh Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Yang kemudian pada tahun itu juga memulai membangun *platform* Android secara intensif. Kemudian pada tanggal 12 November 2007 Google bersama *Open Handset Alliance* (OHA) yaitu konsorsium perangkat *mobile* terbuka, merilis Google Android SDK. Google bersama dengan OHA merilis paket *Software* SDK yang lengkap untuk mengembangkan aplikasi pada perangkat *mobile* yaitu : Sistem operasi, *Software Development Kit* dan aplikasi utama untuk perangkat *mobile.*

Semua aplikasi yang dibuat untuk Android akan memiliki akses yang setara dalam mengakses seluruh kemampuan *handset*, tanpa membedakan apakah itu merupakan aplikasi inti atau aplikasi pihak

ketiga. Dalam kata lain dengan *platform* Android ini, Programmer dan *Developer* secara penuh akan bisa mengkustom perangkat androidnya. Android *built in* pada Linux Kernel (Open Linux Kernel), dengan sebuah mesin *virtual* yang telah didesain dan untuk mengoptimalkan penggunaan sumberdaya memori dan *hardware* pada lingkungan perangkat *mobile.* Dalvik adalah nama dari Android *Virtual Machine*, yang merupakan *interpreter virtual mesin* yang akan mengeksekusi file kedalam format Dalvik Executeable(*.dex), sebuah format yang telah dirancang untuk ruang penyimpanan yang efisien dan eksekusi memori yang terpetakan. Dalvik *Virtual Machine*(Dalvik VM) berbasis *register*, dan dapat mengeksekusi kelas yang telah terkompilasi pada *compiler* bahasa Java, kemudian di transformasikan ke dalam *native* format dengan menggunakan *tool* "dx" yang telah terintegrasi. Berbeda dengan DalvikVM, JavaVM berbasis *stack*. DalvikVM memiliki keunggulan dengan menggunakan *Registered Based*, ini karena pada prosesor perangkat genggam telah dioptimasi untuk eksekusi berbasis *register*. Android saat ini tidak hanya berjalan pada *handphone*, beberapa vendor menanamkan Android pada Tablet, Internet Tablet, E-Book Reader, Laptop, dan gadget lainnya. Dengan begitu akan sangat berharga sekali mempelajari *platform* ini, dengan arsitekturnya yang terbuka, maka *platform* Android adalah *platform mobile* masa depan.

Beberapa fitur yang terdapat dalam Android yaitu:

1) *Application Framework*, memungkinkan penggunaan dan penggantian komponen.
2) Dalvik *Virtual Machine*, untuk mengoptimalkan perangkat *mobile*.
3) *Integrated Browser*, berbasis *open source WebKit engine*.
4) *Optimized Graphics*, didukung oleh *Library* grafis 2D dan 3D berbasis spesifikasi OpenGL ES 1.0.
5) SQLite, untuk menyimpanan data terstruktur.
6) Media *Support*, untuk audio dan video dengan format (MPEG4, H.264, MP3,AAC, AMR, JPG, PNG, GIF).
7) GSM *Telephony* (tergantung *handset*)
8) *Bluetooth*, EDGE, 3G,HSPA and WiFi (tergantung *handset*).
9) Camera, GPS, Compass, Accelerometer (tergantung *handset*).
10) *Rich Development Environment*, tersedianya dukungan yang penuh untuk para pengembang aplikasi termasuk *emulator*, *tools* untuk *debugging*, memori dan kinerja *profil*, dan *plugin* untuk IDE *Eclipse.*

2.2 GPS

GPS adalah singkatan dari *Global Positioning System* yang merupakan sistem untuk menentukan posisi dan navigasi *global* dengan mengunakan satelit. GPS pertama kali dikembangkan oleh departemen pertahanan Amerika, digunakan untuk kepentingan militer maupun sipil (*survey* dan pemetaan).

Sistem GPS, yang bernama asli NAVSTAR GPS (*NAVIGATION Satelite Timing and Ranging Global Positioning System*) mempunyai tiga segmen yaitu: satelit, pengonrol, dan penerima/pengguna. Satelit GPS yang mengorbit bumi, dengan orbit dan kedudukan yang tetap (koordinatnya pasti), seluruhnya berjumlah 24 buah dimana 21 buah aktif bekerja dan 3 buah sisanya adalah cadangan.

Berikut merupakan ke tiga segmen dari sistem GPS :

1 Satelit, bertugas untuk menerima dan menyimpan data yang ditranmisikan oleh stasiun-stasiun pengontrol. Menyimpan dan menjaga informasi waktu berketelitian tinggi (ditentukan dengan jam atomik di satelit), dan memancarkan sinyal dan informasi secara kontinyu ke pesawat penerima (*receiver*) dari pengguna.
2 Pengontrol, bertugas untuk mengendalikan dan mengtrol satelir dari bumi baik untuk mengecek kesehatan satelit, penentuan dan prediksi orbit waktu, singkronisasi waktu antar satelit, dan mengirim data ke satelit.
3 Penerima (*receiver*), bertugas menerima data dari satelit dan memprosesnya untuk menentukan posisi (posisi tiga dimensi yaitu koordinat di bumi plus ketinggian), arah, jarak dan waktu yang diperlukan oleh pengguna. Ada dua macam penerima (*receiver*) yaitu tipe NAVIGASI dan tipe GEODETIC.

Yang termasuk tipe *receiver* NAVIGASI antara lain : *Trimble Ensign, Trimble Pathfinder, Garmin, Sony* dan lain-lain. Sedangkan tipe GEODETIC antara lain : *Topcon, Leica, Astech, Trimble* seri 4000 dan lain-Lain.

2.2.1 Penentuan Posisi Dengan GPS

Pada dasarnya penentuan posisi dengan GPS adalah pengukuran jarak secara bersama-sama ke beberapa satelit (yang koordinatnya telah diketahui) sekaligus. Untuk menentukan koordinat suatu titik di bumi,

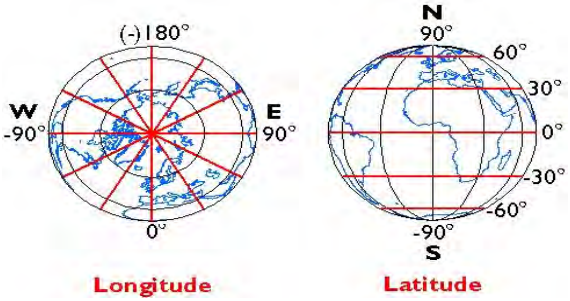
receiver setidaknya membutuhkan 4 posisi atau satelit yang dapat ditangkap sinyalnya dengan baik. Secara *default* koordinat yang diperoleh bereferensi ke *global* datum yaitu *World Geodetic System* 1984 atau disingkat WGS'84.

Secara garis besar penentuan posisi dengan GPS ini dibagi menjadi dua metode yaitu metode *absolut* dan metode relatif.

- Metode *absolut* atau juga dikenal sebagai *point positioning*, menentukan posisi hanya berdasarkan pada sebuah penerima (*receiver*) saja. Ketelitian posisi dalam beberapa meter (tidak berketelitian tinggi) dan umumnya hanya diperuntukan bagi keperluan NAVIGASI.
- Metode relatif atau sering disebut *differential positioning*, menentukan posisi dengan menggunakan lebih dari sebuah penerima (*receiver*). Satu GPS dipasang pada lokasi tertentu di bumi dan secara terus menerus menerima sinyal dari satelit dalam jangka waktu tertentu dijadikan sebagai referensi bagi yang lainnya. Metode ini menghasilkan posisi ketelitian tinggi (umunya kurang dari 1 meter) dan diaplikasikan untuk keperluan survei GEODESI ataupun pemetaan yang memerlukan ketelitian tinggi.

2.2.2 Sistem Koordinat

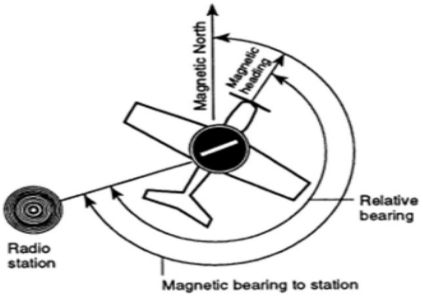
Sistem koordinat *global* yang biasa digunakan dalam sistem GPS disebut sebagai koordinat GEOGRAFI. Koordinat ini diukur dalam lintang dan bujur dalam besaran derajat desimal, derajat menit desimal, atau derajat menit detik. Lintang diukur terhadap ekuator sebagai titik NOL (0^o sampai 90^o *positif* kearah utara dan 0^o sampai 90^o *negatif* kearah selatan). Adapun bujur diukur berdasarkan titik NOL di Greenwich NOL (0^o sampai 180^o kearah timur dan 0^o sampai 180^o kearah barat). Tititk 180^o dari kedua bujur ini berada didaerah Samudra Pasifik. Koordinat geografi ini dapat dipetakan ke koordinat XY dengan sumbu X sebagai bujur dan sumbu Y sebagai lintang.



Gambar 2.2 Garis Lintang dan Bujur

2.2.3 *Heading* dan *Bearing*

Pada sistem navigasi, dikenal istilah *Heading* dan *Bearing*. Heading merupakan istilah yang digunakan untuk merepresentasikan arah menghadap dari suatu benda. Pada sistem navigasi pesawat, yang menjadi patokan dalam menentukan heading adalah hidung pesawat (nose). Disini arah *heading* merupakan representasi arah pesawat dengan sudut referensi 0 derajat pada arah utara (*true north*). Jadi apabila pesawat sedang mengarah ke timur, maka heading dari pesawat tersebut adalah 90 derajat.



Gambar 2.3 Ilustrasi *bearing* dan *heading*

Bearing merupakan istilah lainnya yang juga banyak digunakan pada sistem navigasi pesawat. *Bearing* merupakan derajat yang

merepresentasikan posisi suatu titik terhadap titik lainnya. Sudut bearing umumnya menggunakan sudut referensi arah utara (*true north*) sebagai titik 0 derajatnya.

2.2.4 Navigasi *Waypoint*

Navigasi *waypoint* adalah suatu sistem pergerakan titik dari koordinat titik awal terhadap koordinat titik tujuan pada bidang XY. Pada GPS koordinat titik yang didapat dihasilkan dari koordinat *latitude* (lintang) dan koordinat *longitude* (bujur). Dimana *latitude* (lintang) diukur terhadap ekuator sebagai titik NOL (0^o sampai 90^o positif kearah utara dan 0^o sampai 90^o negatif kearah selatan) bergerak secara vertikal dan pada bidang XY diplot pada sumbu X. Adapun *longitude* (bujur) diukur berdasarkan titik NOL di Greenwich NOL (0^o sampai 180^o kearah timur dan 0^o sampai 180^o kearah barat) bergerak secara horisontal dan pada bidang XY diplot pada sumbu Y.

2.2.5 *Assisted* GPS

Assisted GPS atau biasa disebut A-GPS merupakan perkembangan dari GPS. Umumnya A-GPS digunakan pada telepon selular dan perangkat *mobile* lainnya yang terhubung dengan jaringan internet.

Secara umum, cara kerja A-GPS adalah sama dengan GPS pada umumnya,yakni menangkap sinyal dari GPS dan mengkalkulasi posisi GPS dengan menghitung jarak satelit menggunakan metode triangulasi. Perbedaan paling mendasar antara GPS dan A-GPS adalah bagaimana mengakuisisi data *almanac* dan *epherneris error* satelit. Kedua data ini merupakan data yang digunakan untuk mencari posisi satelit diangkasa. Pada sistem GPS biasa, data ini ditangkap dan diterjemahkan oleh GPS *receiver* dari pancaran *transmisi* satelit GPS dengan *transfer rate global* 50bps. Karena *transfer rate* data yang rendah ini, umumnya GPS biasa membutuhkkan waktu rata-rata 12,5 menit untuk dapat mengunci posisi satelit jika GPS telah di *non*-aktifkan lebih dari 30 menit. Pada A-GPS, data *almanac* dan *epherneris error* tidak di *dowload* langsung melalui satelit, namun di *download* melalui jaringan internet pada *server* yang menyediakan data tersebut, karena umumnya kecepatan internet/selular lebih dari 50 bps, maka kedua data ini akan didapat dengan lebih cepat dibanding pada GPS biasa sehingga penguncian posisi satelit dapat dilakukan dengan lebih cepat. Selain itu, A-GPS juga mampu

melakukan perhitungan posisi tanpa harus terkoneksi dengan satelit GPS. Hal ini dilakukan dengan melakukan triangulasi posisi menggunakan cell tower/BTS dari jaringan yang digunakan. Metode ini dapat menentukan posisi dengan lebih cepat namun dengan akurasi yang sangat rendah jika dibanding dengan triangulasi dari satelit.

Pada *smartphone* Android, dalam keadaan terhubung jaringan, akan digunakan mode A-GPS untuk menentukan posisi dari *user*. Namun apabila *smartphone* tidak terkoneksi jaringan, maka *smartphone* android akan mengubah mode menjadi mode GPS.

2.2.6 Kelemahan GPS

Walaupun memiliki kemampuan navigasi yang cukup luar biasa, GPS juga memiliki beberapa kelemahan sehingga mengganggu akurasi dari navigasi. Berikut ini beberapa kelemahan dari GPS :

1. *Delay* di *ionosphere* dan *troposphere*: sinyal satelit terganggu saat melewati atmosfir bumi lapisan ini terdapat di permukaan bumi pada ketinggian 50 - 500 km. Partikel partikel yang terionisasi pada lapisan ini membuat pengaruh pada GPS sinyal sehingga mengakibatkan salah satu penyebab *error* tertinggi dalam penentuan jarak dan lokasi pada GPS *Receiver*. Sedangangkan lapisan *Troposphere* berada ketinggian 50 Km kebawah sampai dengan permukaan bumi yang selalu mengalami perubahan temperature tekanan awan ,debu, hanya relatif sedikit sebagai mengganggu sinyal *transmisi* dari Satelit GPS yang menjadi penyebab *error* atau kesalahan perhitungan dari GPS *Receiver*.
2. *Signal multipath*: terjadi ketika sinyal GPS dipantulkan oleh gedung tinggi atau permukaan padat seperti pegunungan sebelum sinyal mencapai *receiver*. Hal Ini menambah lama waktu perjalanan sinyal (timing), karena itu menyebabkan *error* pada perhitungan *Receiver* GPS.
3. *Error* pada *clock* di *receiver*. *Built-in Clock* di *receiver* tidak seakurat *atomic clock* yang ada di satelit GPS. Maka dari itu, akan mudah terjadi *error* dalam penentuan waktu.
4. Orbital (*ephemeris*) *error*, hal ini terjadi akibat ketidakakuratan laporan lokasi satelit. Semakin banyak satelit yang bisa di *Access* oleh *receiver*, semakin akurat informasi yang didapat. Bangunan, kontur bumi, interferensi peralatan elektronika atau bahkan rimbun dedaunan, dapat mengganggu penerimaan sinyal yang

menyebabkan kesalahan posisi. *Receiver* biasanya tidak bisa bekerja di dalam ruangan, di dalam air atau di bawah tanah.

5. Geometri satelit: Ini merujuk pada posisi relatif satelit di suatu waktu tertentu. Geometri satelit ideal terjadi ketika satelit berada di sudut yang lebar relatif terhadap satelite lainnya. Geometry yang buruk terjadi ketika satelit berada satu garis atau jarak yang terlalu dekat dengan yang lainnya mengakibatkan melesetnya perhitungan yang dilakukan *receiver* GPS.

2.3 Kompas

Sejak dulu kala, kompas digunakan untuk mengetahui arah mata angin. Kompas ini bekerja berdasarkan medan magnet yang dihasilkan oleh bumi. Seiring dengan kemajuan jaman, telah dikembangkan sebuah rangkaian dan sensor medan magnet yang digunakan untuk mengukur medan magnet bumi sehingga berfungsi sebagai kompas digital.

Kompas merupakan sensor *magnetometer* yang digunakan untuk membaca medan magnet di sekitar sensor tersebut. Secara umum sensor ini digunakan sebagai pedeteksi benda-benda logam (metal detector) dengan membaca perubahan medan magnet yang cukup besar. Selain sebagai *metal detector*, sensor *magnetometer* juga dapat digunakan untuk membaca medan magnet bumi dan menentukan arah medan magnet bumi (utara dan selatan).

Sensor *magnetometer* mengukur besaran medan magnet dalam satuan *nanoTesla* (Nt) dengan melihat seberapa besar medan magnet di arah depan sensor,*user* dapat memperkirakan arah menghadap (*heading*) terhadap arah utara. Hasil pengukuran dari *magnetometer* umumnya sama akurat jika dibandingan dengan kompas analog, dengan syarat disekitar sensor tidak ada medan magnet yang menggangu pembacaan sensor.

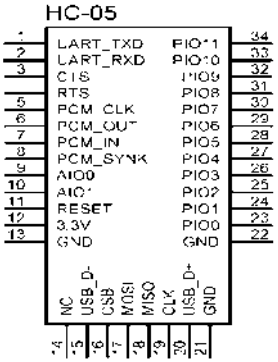
Sensor *magnetometer* umumnya digunakan pada dunia *aeromodeling* yakni digunakan untuk menghitung, *yaw*, *pitch*, dan *roll* dari sebuah model R/C *helicopter* atau pesawat terbang. Untuk sistem ini umumnya R/C *helicopter* menggunakan gabungan dari sensor *magnetometer*, *accelerometer* dan *Gyroscope* untuk menghasilkan pembacaan yang benar.

Pengunaan sensor *magnetometer* sebagai alat navigasi juga memiliki kelemahan. Hal ini juga dialami oleh kompas analog yakni adanya deklinasi. Deklinasi merupakan sebuah fenomena dimana arah

utara yang ditunjukkan oleh kompas bukanlah arah utara yang sebenarnya. Arah utara kompas dan arah utara sebenarnya memiliki perbedaan sudut yang besarnya berbeda untuk posisi yang berbeda pula. Beberapa ahli geografis telah dapat memetakan perbedaan deklinasi dari berbagai tempat di dunia. Perbedaan ini digambarkan dengan menggunakan garis *isogonic*, yakni garis yang mewakili daerah dengan nilai deklinasi yang sama.

2.4 *Bluetooth* HC-05

Bluetooth digunakan untuk komunikasi antara *smartphone* Android dengan mikrokontroler. *Bluetooth* adalah sebuah teknologi antarmuka radio yang beroperasi dalam pita frekuensi 2,4 Ghz *unlincensed* ISM (*Industrial, Scientific an Medical*) yang mampu menyediakan layanan komunikasi data dan suara secara *real-time* antara *host-host Bluetooth* dengan jarak jangkauan layanan yang terbatas (sekitar 10 meter). *Bluetooth* menggunakan sistem *Frequency Hopping Spread Spectrum* (FHSS) yang mempunyai kecepatan maksimum 1Mbps. Pada awalnya teknologi *Bluetooth* dipromosikan untuk pengguna LAN. Namun, mengingat jangkauan maksimum yang tidak terlalu luas, *Bluetooth* kemudian dipromosikan untuk penggunaan dalam *personal area network* (PAN). Pada tugas akhir ini komunikasi yang dilakukan yaitu antara *Bluetooth* yang sudah tersedia pada *device* android dengan modul *Bluetooth* HC-05 yang terhubung pada mikrokontroler.



Gambar 2.4 Modul *Bluetooth* HC-05

2.5 Sensor Ultrasonik

Sensor ultrasonik adalah sensor yang bekerja berdasarkan prinsip pantulan gelombang suara, dimana sensor ini menghasilkan gelombang suara yang kemudian menangkapnya kembali dengan perbedaan waktu sebagai dasar penginderaannya. Perbedaan waktu antara gelombang suara yang dipancarkan dengan gelombang suara yang diterima kembali berbanding lurus dengan jarak atau tinggi objek yang memantulkannya. Gelombang ultrasonik timbul akibat getaran mekanik dengan frekuensi diatas batas ambang pendengaran manusia yakni diatas 20KHz-100KHz. Jika gelombang bolak-balik terjadi terus menerus secara periodik maka akan menghasilkan deretan gelombang periodik dimana pada setiap gerak periodik, partikel-partikel yang berada pada titik-titik yang sama pada gelombang tersebut akan berada dalam fase yang sama.

Gelombang ultrasonik merupakan gelombang longitudinal, dengan demikian gelombang ultrasonik memiliki sifat fisik gelombang longitudinal. Gelombang ultrasonik tidak dapat merambat pada vakum atau ruang hampa, gelombang ini memerlukan medium untuk merambat. Dalam perambatan ini dikenal istilah displacement, yaitu pergeseran partikel-partikel yang dilalui gelombang ultrasonik. Perambatan gelombang terjadi saat adanya perbedaan tekanan antara kedua ujung zat antara udara, logam dan lain-lain, menyebabkan terjadinya pengusikan atom-atom zat antara tersebut. Pengusikan tersebut menghasilkan kecepatan v. Atom atom tidak bergerak terlalu jauh karena terikat oleh gaya ikat antar atom. Tetapi energi dari satu atom akan pindah ke atom lain dengan kecepatan tertentu.

Kecepatan rambat gelombang ultrasonik tergantung pada kerapatan (*density*) dan elastisitas dari medium. Kecepatan rambatan ultrasonik berbeda untuk medium yang berbeda, hal ini dijelaskan dengan rumus:

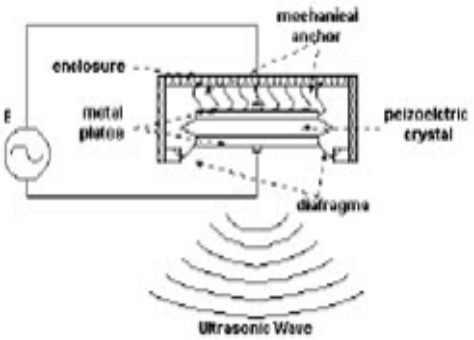
$$\mathbf{V = \lambda f}$$

Dimana: V = kecepatan rambat
λ = panjang gelombang
f = frekuensi

kecepatan rambat gelombang uktrasonik pada medium yang sama akan selalu tetap meskipun frekuensinya dirubah. Sedangkan untuk medium yang berbeda dengan frekuensi yang sama terjadi perubahan

panjang gelombang yang menyebabkan perubahan kecepatan rambat gelombang. Kecepatan rambat gelombang berbeda untuk zat yang berbeda, kecepatan rambat terbesar terjadi dalam zat padat dan kecepatan menurun pada zat cair dan kecepatan paling lambat terjadi pada zat gas.

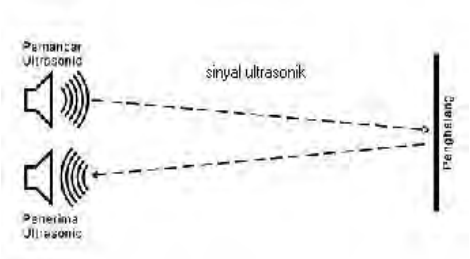
Sensor ultrasonik terdiri dari dua unit yaitu unit pemancar (Tx) dan unit penerima (Rx). Struktur dari unit pemancar dan penerima yaitu sebuah kristal *piezoelectric* dihubungkan dengan mekanik jangkar dan hanya dihubungkan dengan diafragma penggetar. Tegangan bolak-balik yang memiliki frekuensi 40KHz-400KHz diberikan pada plat logam. Struktur atom dari kristal *piezoelectric* akan berkontraksi (mengikat), mengembang atau menyusut terhadap polaritas tegangan yang diberikan, dan ini yang disebut dengan efek *piezoelectric*. Kontraksi yang terjadi diteruskan ke diafragma penggetar sehingga terjadi gelombang ultrasonik yang dipancarkan ke udara (tempat sekitarnya) dan pantulan gelombang ultrasonik akan terjadi bila ada objek tertentu kemudian pantulan gelombang ultrasonik akan diterima kembali oleh unit sensor penerima. Selanjutnya unit sensor penerima akan menyebabkan diafragma penggetar akan bergetar dan efek *piezoelectric* menghasilkan sebuah tegangan bolak-balik dengan frekuensi yang sama.



Gambar 2.5 Prinsip kerja sensor ultrasonik

Besar amplitudo sinyal elekrik yang dihasilkan unit sensor penerima tergantung dari jauh dekatnya objek yang dideteksi serta kualitas dari sensor pemancar dan sensor penerima. Proses *sensing* yang dilakukan

pada sensor ini menggunakan metode pantulan untuk menghitung jarak antara sensor dengan obyek sasaran. Jarak antara sensor tersebut dihitung dengan cara mengalikan setengah waktu yang digunakan oleh sinyal ultrasonik dalam perjalanannya dari rangkaian Tx sampai diterima oleh rangkaian Rx, dengan kecepatan rambat dari sinyal ultrasonik tersebut pada media rambat yang digunakannya, yaitu udara.



Gambar 2.6 Prinsip pemantulan gelombang ultrasonik

Pada Gambar 2.6 diatas dapat dilihat suatu proses pemantulan gelombang ultrasonik, dimana unit pemancar ultrasonik memancarkan sinyal ultrasonik yang kemudian sinyal ultrasonik tersebut di pantulkan oleh penghalang hingga pantulannya kembali lagi dan diterima oleh unit penerima ultrasonik. hasil sinyal yang diterima oleh tranduser penerima akan dikonversikan menjadi jarak. Dengan mengukur selang waktu antara saat pulsa dikirim dan pulsa diterima , jarak antara obyek dengan alat pengukur akan dapat dihitung berdasarkan persamaan :

$$\mathbf{D = V . T}$$

Dimana,V = kecepatan suara (m/ det)
T = selang waktu (detik)
D = jarak antara sumber dengan obyek (m)

2.5.1 Sensor Ultrasonik HC-SR04

Sensor jarak HC-SR04 adalah sebuah *device* transmitter dan receiver ultrasonic dalam 1 paket buatan Devantech yang dapat membaca jarak dengan prinsip sonar.

Gambar 2.7 Bentuk fisik sensor ultrasonik HC-SR04

Sensor ultrasonik HC-SR04 memiliki beberapa karakteristik diantaranya yaitu :

1. Tegangan keja : 5V DC
2. Konsumsi arus : 15mA
3. Frekuensi kerja : 40KHz
4. Sudut jangkauan : 15 derajat
5. Jarak Jangkauan : 2cm - 400cm
6. Input trigger : 10us, level pulsa TTL
7. Dimensi : P x L x T (45 x 20 x 15) mm

Ultrasonik HC-SR04 mendeteksi jarak objek dengan cara memancarkan gelombang ultrasonik (40KHz) selama tBURST (200µs) kemudian mendeteksi pantulannya. Sensor akan memancarkan gelombang ultrasonik sesuai dengan kontrol dari mikrokontroler pengendali (pulsa *trigger* dengan tOUT min. 2 µs). Gelombang ultrasonik ini melalui udara dengan kecepatan 344 meter per detik, mengenai obyek dan memantul kembali ke sensor penerima.

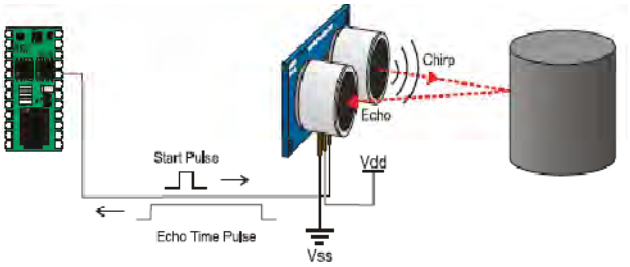
Prinsip kerja HC-SR04 adalah pemancar memancarkan seberkas sinyal ultrasonic (40KHz) yang berbentuk pulsa, kemudian jika di depan HC-SR04 terdeteksi objek padat maka penerima akan menerima pantulan sinyal ultrasonik tersebut. Penerima akan membaca lebar pulsa yang dipantulkan objek dan selisih waktu pemancaran. Dengan pengukuran tersebut, jarak objek di depan sensor dapat diketahui.

Untuk mengaktifkan HC-SR04, mikrokontroler harus mengirimkan pulsa positif minimal 10us melalui pin *trigger*, maka HC-SR04 akan mengeluarkan sinyal ultrasonic sebesar 8 *cycle* dan selanjutnya HC-SR04 akan memberikan pulsa 100us-18ms pada outputnya tergantung pada informasi jarak pantulan objek yang diterima.

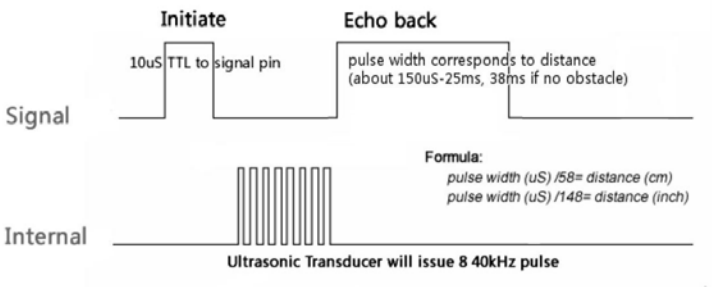
Sensor HC-SR04 mengeluarkan pulsa *output high* pada pin trig setelah memancarkan gelombang ultrasonik dan setelah gelombang pantulan terdeteksi akan membuat *output low* pada pin trig. Lebar pulsa *High* (tIN) akan sesuai dengan lama waktu tempuh gelombang ultrasonik untuk 2x jarak ukur dengan obyek. Selanjutnya mikrokontroler cukup mengukur lebar pulsa tersebut dan mengkonversinya dalam bentuk jarak dengan perhitungan sebagai berikut :

Jarak = [(tIN s x 344 m/s) ÷ 2] (dalam satuan meter)

Keterangan :
tIN = lebar pulsa
344 m/s = kecepatan gelombang ultrasonik melalui udara
2 = 2 kali siklus gelombang ultrasonik



Gambar 2.8 Proses kerja sensor ultrasonik HC-SR04

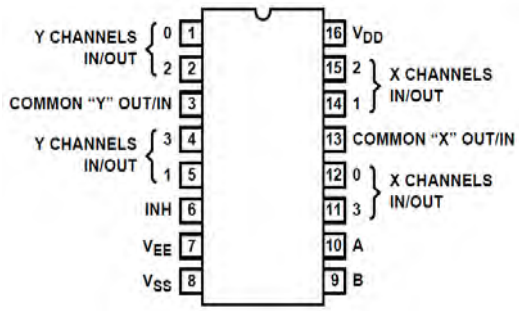


Gambar 2.9 Pulsa Ultrasonic HC-SR04

Pada sensor ultrasonik HC-SR04 terdapat 4 pin yang digunakan yaitu vcc, ground, trig dan echo. Pin trig dan echo dapat langsung dihubungkan dengan mikrokontroler tanpa tambahan komponen apapun. Namun karena pada tugas akhir ini menggunakan 4 sensor ultrasonik yang harus aktif secara bersamaan, sehingga akan membuuhkan banyak ruang pada mikrokontroler, maka digunakan Multiplexer tipe CD4052 sebagai selector.

2.5.2 Multiplexer CD4052

Multiplexer (Mux) berfungsi sebagai selektor / penyeleksi data berdasarkan perintah untuk menampilkan data yang diingkan.



Gambar 2.10 Konfigurasi pin multiplexer CD4052

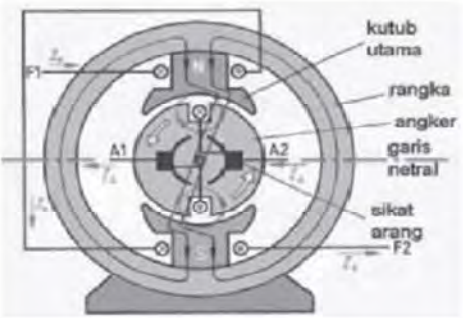
Tabel 2.1 Tabel kebenaran multiplexer CD4052.

INHIBIT	B	A	ON CHANNEL
0	0	0	0x , 0y
0	0	1	1x , 1y
0	1	0	2x , 2y
0	1	1	3x , 3y
1	X	X	None

Dari tabel kebenaran diatas dapat dilihat bahwa untuk mengaktifkan x *channel* dan y *channel* maka perlu di set pada pin A dan B.

2.6 Motor DC

Motor DC merupakan jenis motor yang menggunakan tegangan searah sebagai sumber tenaganya. Dengan memberikan beda tegangan pada kedua terminal tersebut, motor akan berputar pada satu arah, dan bila polaritas dari tegangan tersebut dibalik maka arah putaran motor akan terbalik pula. Polaritas dari tegangan yang diberikan pada dua terminal menentukan arah putaran motor sedangkan besar dari beda tegangan pada kedua terminal menentukan kecepatan motor.



Gambar 2.11 Konstruksi Motor DC

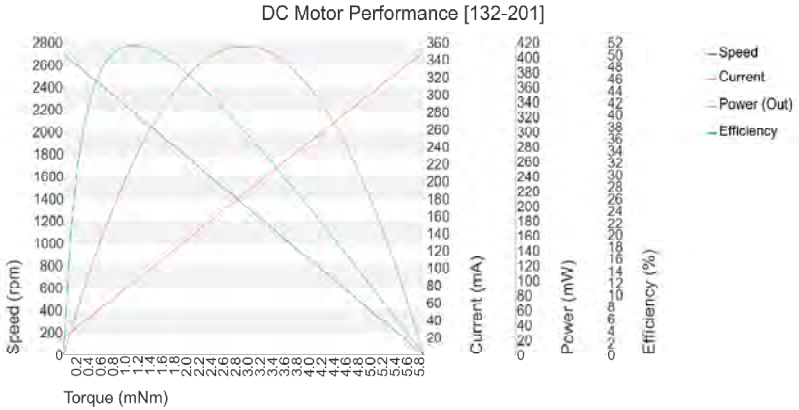
Konstruksi motor DC pada gambar 2.11 memiliki 2 bagian dasar, yaitu :

1. Bagian yang tetap/stasioner yang disebut stator. Stator ini menghasilkan medan magnet, baik yang dibangkitkan dari sebuah koil (elektro magnet) ataupun magnet permanen.
2. Bagian yang berputar disebut rotor. Rotor ini berupa sebuah koil dimana arus listrik mengalir.

Gaya elektromagnet pada motor DC timbul saat ada arus yang mengalir pada penghantar yang berada dalam medan magnet. Medan magnet itu sendiri ditimbulkan oleh megnet permanen. Garis-garis gaya magnet mengalir diantara dua kutub magnet dari kutub utara ke kutub selatan. Menurut hukum gaya Lourentz, arus yang mengalir pada penghantar yang terletak dalam medan magnet akan menimbulkan gaya. Gaya F timbul tergantung pada arah arus I, dan arah medan magnet B.

Gambar 2.12 Motor DC

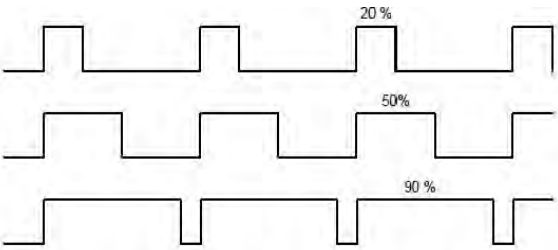
Konstruksi motor DC yang digunakan ini merupakan motor DC yang telah ditambahkan *gearbox* untuk mengendalikan pergerakan motor DC tersebut.



Gambar 2.13 Karakteristik permorma motor DC

Dari gambar 2.13 diatas dapat dilihat karakteristik dari *speed*, *torsi*, arus, power dan efisiensi.

Sistem kontrol pada motor DC dilakukan dengan mengirimkan pulsa PWM (*Pulse Width Modulation*) pada pin *control*-nya. PWM merupakan suatu cara memanipulasi lebar dari pulsa dalam perioda yang konstan untuk mendapatkan tegangan rata-rata yang berbeda.



Gambar 2.14 Sinyal PWM dengan berbagai duty cycle

Pada gambar 2.14 diatas diperlihatkan tiga sinyal PWM yang berbeda. Sinyal yang paling atas menunjukkan sinyal PWM dengan *duty cycle* 20%, artinya sinyal ON selama 20% dari perioda sinyal dan off selama 80% sisanya. Gambar yang lainnya menunjukkan sinyal dengan *duty cycle* 50% dan 90%. Ketiga sinyal PWM tersebut akan menghasilkan sinyal analog yang berbeda.

2.7 Mikrokontroler

Mikrokontroler merupakan sebuah prosesor yang digunakan untuk kepentingan kontrol. Meskipun mempunyai bentuk yang jauh lebih kecil dari suatu komputer pribadi dan komputer *mainframe*, mikrokonroler dibangun dari elemen-elemen dasar yang sama. Seperti pada umumnya komputer, mikrokontroler adalah alat yang mengerjakan instruksi-instruksi yang diberikan kepadanya.

Beberapa fitur yang umumnya ada di dalam mikrokontroler adalah sebagai berikut :

- RAM (*Random Acces Memory*)
  RAM digunakan oleh mikrokontroler untuk tempat penyimpanan variable. Memori ini bersifat *volatile* yang berarti akan kehilangan semua datanya jika tidak mendapatkan catu daya.
- ROM (*Read Only Memory*)
  ROM seringkali disebut sebagai kode memori karena berfungsi untuk tempat penyimpanan program yang akan diberikan oleh *user*.
- *Register*
  Merupakan tempat penyimpanan nilai-nilai yang akan digunakan dalam proses yang telah disediakan oleh mikrokontroler.

- *Special Function Register*
  Merupakan *register* khusus yang berfungsi untuk mengatur jalannya mikrokontroler. *Register* ini terletak pada RAM.
- *Input dan Output*
  Pin *input* adalah bagian yang berfungsi sebagai penerima sinyal dari luar, pin ini dapat dihubungkan ke berbagai media input seperti *keypad*, sensor dan sebagainya. Sedangkan pin *Output* adalah bagian yang berfungsi untuk mengeluarkan sinyal dari hasil proses algoritma mikrokontroler.
- *Interrupt*
  Interrupt merupakan bagian dari mikrokontroler yang berfungsi sebagai bagian yang dapat melakukan interupsi, sehingga ketika program utama sedang berjalan, program utama tersebut dapat di interupsi dan menjalankan program interupsi terlebih dahulu.
  Beberapa interrupt pada umumnya adalah sebagai berikut :
  - *Interrupt Eksternal*
    Interrupt akan terjadi bila ada input dari pin interrupt.
  - *Intterupt Timer*
    Intterupt akan terjadi bila waku tertentu telah tercapai.
  - *Intterupt Serial*
    Intterupt yang akan terjkadi ketika ada penerimaan data dari komunikasi serial.

2.7.1 Mikrokontroler ATMega328

ATMega328 adalah mikrokontroler keluaran dari atmel yang mempunyai arsitektur RISC (*Reduce Instruction Set Computer*) yang dimana setiap proses eksekusi data lebih cepat dari pada arsitektur CISC (*Completed Instruction Set Computer*)

Beberapa fitur dari ATMega328 antara lain adalah :

- 130 macam instruksi yang hampir semuanya dieksekusi dalam satu siklus *clock*.
- 32 x 8-bit *register* serba guna.
- Kecepatan mencapai 16MIPS dengan *clock* 16MHZ.
- 32 KB *flash memory*.
- Memiliki EEPROM (*Electrically Erasable Programmable Read Only Memory*) sebesar 1 KB sebagai tempat penyimpanan

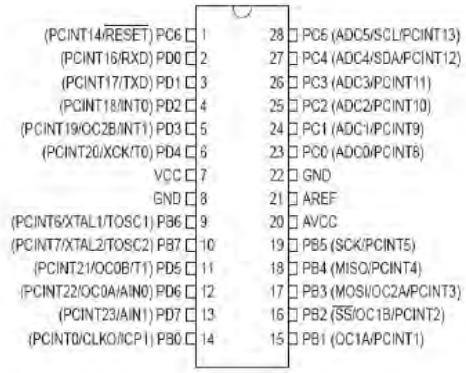
data semi permanen karena EEPROM tetap dapat menyimpan data meskipun catu daya dimatikan.

- Memiliki SRAM (*Static andom Acces Memory*) sebesar 2 KB.
- Memiliki pin I/O digital sebanyak 14 pin, 6 diantaranya merupakan PWM (*Pulse width Modulation*) output.
- *Master/Slave* SPI *Serial interface.*

Mikrokontroler ATMega328 memiliki arsitektur Harvard, yaitu memisahkan memori untuk kode program dan memori untuk data sehingga dapat memaksimalkan kerja dan parallelism. Instruksi-instruksi dalam memori program dieksekusi dalam satu alur tunggal, dimana pada saat satu instruksi dikerjakan instruksi berikutnya sudah diambil dari memori program. Konsep inilah yang memungkinkan instruksi-instruksi dapat dieksekusi dalam setiap satu siklus *clock.*

Hampir semua instruksi AVR memiliki format 16-bit. Setiap alamat memori program terdiri dari instruksi 16-bit atau 32-bit. Selain *register* serba guna di atas, terdapat register lain yang terpetakan dengan teknik *memory mapped* I/O selebar 64 byte. Beberapa *register* ini digunakan untuk fungsi khusus antara lain sebagai *register control Timer/Counter*, interupsi, ADC, USART, SPI, EEPROM, dan fungsi I/O lainnya. *Register-register* ini menempati memori pada alamat 0x20h – 0x5fh.

2.7.2 Konfigusai Pin ATMega328



Gambar 2.15 Konfigurasi Pin ATMega328

ATMega328 memiliki 3 buah PORT utama yaitu PORTB, PORTC, dan PORTD dengan total pin *input/output* sebanyak 23 pin. PORT tersebut dapat difungsikan sebagai *input/output* digital atau difungsikan sebagai periperial lainnya.

1) Port B
   Port B merupakan jalur data 8 bit yang dapat difungsikan sebagai *input/output*. Selain itu PORTB juga dapat memiliki fungsi alternatif seperti di bawah ini.
   a. ICP1 (PB0), berfungsi sebagai *Timer Counter* 1 *input capture* pin.
   b. OC1A (PB1), OC1B (PB2) dan OC2 (PB3) dapat difungsikan sebagai keluaran PWM (*pulse width modulation*).
   c. MOSI (PB3), MISO (PB4), SCK (PB5), SS (PB2) merupakan jalur komunikasi SPI.
   d. Selain itu pin ini juga berfungsi sebagai jalur pemograman serial (ISP).
   e. TOSC1 (PB6) dan TOSC2 (PB7) dapat difungsikan sebagai sumber *clock external* untuk *timer*.
   f. XTAL1 (PB6) dan XTAL2 (PB7) merupakan sumber *clock* utama mikrokontroler.

2) Port C
   Port C merupakan jalur data 7 bit yang dapat difungsikan sebagai *input/output* digital. Fungsi alternatif PORTC antara lain sebagai berikut.
   a. ADC6 *channel* (PC0,PC1,PC2,PC3,PC4,PC5) dengan resolusi sebesar 10 bit. ADC dapat kita gunakan untuk mengubah input yang berupa tegangan analog menjadi data digital.
   b. I2C (SDA dan SDL) merupakan salah satu fitur yang terdapat pada PORTC. I2C digunakan untuk komunikasi dengan sensor atau *device* lain yang memiliki komunikasi data tipe I2C seperti sensor kompas,*accelerometer nunchuck*.

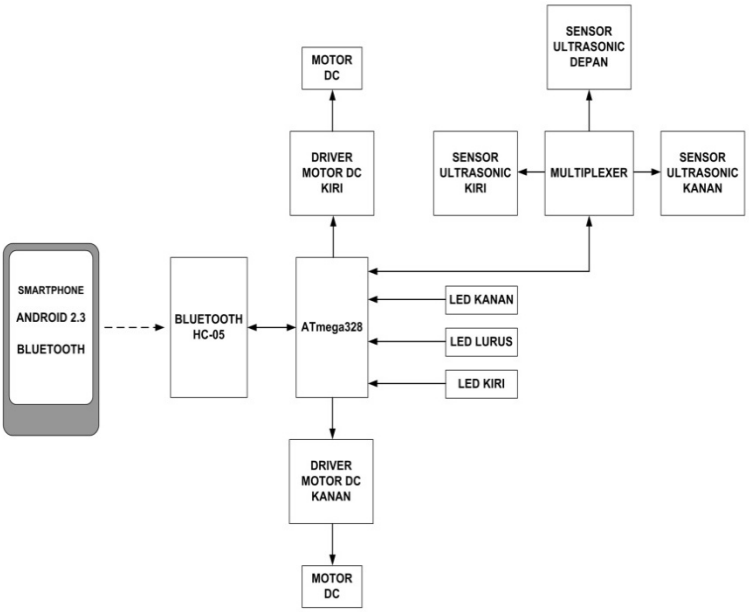
3) Port D
   Port D merupakan jalur data 8 bit yang masing-masing pin-nya juga dapat difungsikan sebagai *input/output*. Sama seperti *Port B* dan *Port C*, *Port D* juga memiliki fungsi alternatif dibawah ini.

a. USART (TXD dan RXD) merupakan jalur data komunikasi serial dengan level sinyal TTL. Pin TXD berfungsi untuk mengirimkan data serial, sedangkan RXD kebalikannya yaitu sebagai pin yang berfungsi untuk menerima data serial.
b. *Interrupt* (INT0 dan INT1) merupakan pin dengan fungsi khusus sebagai interupsi *hardware*. Interupsi biasanya digunakan sebagai selaan dari program, misalkan pada saat program berjalan kemudian terjadi interupsi *hardware /Software* maka program utama akan berhenti dan akan menjalankan program interupsi.
c. XCK dapat difungsikan sebagai sumber *clock external* untuk USART, namun kita juga dapat memanfaatkan *clock* dari CPU, sehingga tidak perlu membutuhkan *external clock.*
d. T0 dan T1 berfungsi sebagai masukan counter external untuk *timer* 1 dan *timer* 0.
e. AIN0 dan AIN1 keduanya merupakan masukan input untuk *analog comparator.*

BAB III

PERANCANGAN SISTEM

Pada bab ini, akan dibahas tentang perancangan secara keseluruhan dari sistern navigasi *mobile* robot ini. Pada perancangan sistem yang akan dibahas ini dibagi dalam 2 sub bab utama yakni perancangan *Software* (Program Android) dan peracangan *Hardware* (mikrokontroler). Pada sub bab perancangan *Software* akan dibahas mengenai bagaimana pembuatan aplikasi Android yang berupa *user interface* (UI) dan akses sistem sensor pada Android serta bagaimana melakukan pengiriman data menggunakan *Bluetooth*. Kemudian pada sub bab selanjutnya perancangan *hardware* akan dibahas mengenai bagaimana implementasi mikrokontroler untuk rnenerima data dari Android dan mengkonversinya menjadi sinyal kontrol untuk motor DC.



Gambar 3.1 Diagram blok sistem

Pada diagram blok diatas dapat dilihat bahwa perancangan sistem ini terdiri dari 2 bagian inti yaitu perancangan pada Android dan perancangan pada Mikrokontroler. Keduanya diharuskan bekerja secara sinkron agar sistem yang dibangun dapat berjalan dengan semestinya.

Android disini difungsikan sebagai Sensor dan Sistem yang berfungsi mengkalkulasi Arah dan Kecepatan untuk *mobile* robot, sedangkan sistem Mikrokontroler digunakan untuk mendapatkan data dan mengkonversikannya ke dalam sinyal yang dapat dimengerti oleh motor DC. Untuk mengkomunikasikan kedua bagian ini, maka diperlukan sebuah protokol komunikasi yang dapat membuat sistem Android dapat mengirimkan data pada mikrokontroler. Pada tugas akhir ini digunakan protokol komunikasi *Bluetooth* Android yang akan dikomunikasikan ke modul *Bluetooth* HC-05.

Dikarenakan sistem yang dirancang mengharuskan untuk mengembangkan sistem untuk dua *environtment*, maka Perancangan sistem secara detail akan dibagi menjadi dua, yakni perancangan Sistem Android dan perancangan Mikrokontroler.

3.1 Perancangan *Software*

Untuk mengembangkan suatu aplikasi android maka dibutuhkan suatu *Software*. *Software* yang digunakan untuk mengembangkan suatu aplikasi android ada dua yaitu dengan menggunakan Eclipse dan dengan menggunakan line Tools. Pada tugas akhir pengembangan aplikasi android menggunakan *Software* Eclipse. Pemilihan Eclipse karena *Software* ini gratis dan open source yang berarti semua orang boleh melihat kode pemrograman perangkat lunak ini. Jenis Eclipse yang digunakan pada tugas akhir ini adalah Eclipse Juno.

Sebelum menggunakan *Software* Eclipse untuk mengembangkan android harus disediakan android SDK manager (*Software* development kid) dan JDK (*java development kid*). Setelah dua perangkat tersebut disediakan maka harus *install* ADT *plugin* (android *development tools*) untuk Eclipse. Terakhir *download* SDK tools dan *platform* android yang akan dikembangkan melalui SDK manager.

Gambar 3.2 Tampilan awal Eclipse Juno

Perancangan *Software* dapat dibagi menjadi 2 bagian, yaitu perancangan sistem untuk akses sensor (GPS, Kompas) yang meliputi sensor apa saja yang akan digunakan, bagaimana pengambilan datanya, bagaimana pengolahan data mentahnya hingga data bisa di manfaatkan. Dan perancangan selanjutnya merupakan perancangan sistem untuk komunikasi antara android dengan mikrokontroler.

Pada tugas akhir ini GPS dan kompas yang digunakan adalah dengan memanfaatkan GPS dan kompas yang ada pada *device* Android.

3.1.1 Parancangan *User Interface* (UI)

Pada perancangan aplikasi Android ini akan memanfaatkan Eclipse sebagai *Software* yang akan digunakan dalam perancangan *user interface.*

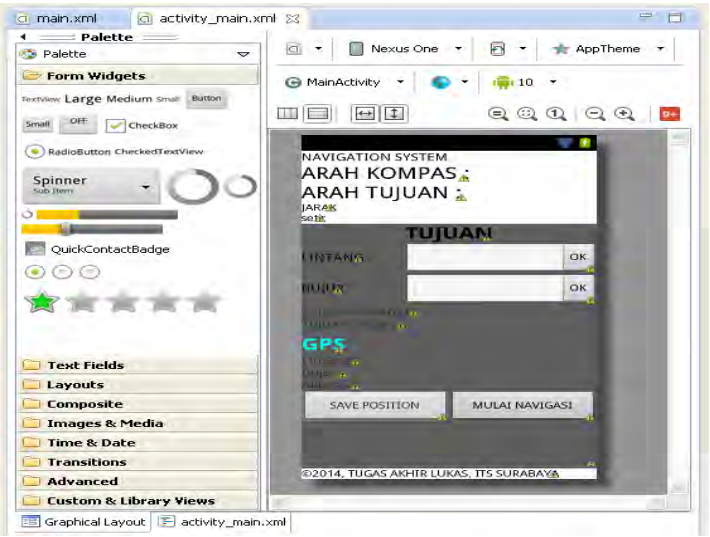
Pada perancangan UI ini akan dibuat sebuah kontroler untuk rnernpermudah *user* dalam mengerti dan memprogram sistem Pada program Eclipse. pembuatan UI ini akan dilakukan dengan meggunakan 2 cara, yakni menggunakan *drag and drop widget* dan pemrograman dengan bahasa xml.

Dalam pembuatan tampilan UI ada beberapa bagian yang akan ditampilkan pada UI, yakni :

- Posisi GPS
- Posisi Target

- Arah kompas
- Arah tujuan
- *Form* Manual *Input* untuk menentukan posisi target
- Perintah/tombol untuk melakukan *save input*
- Perintah/tombol untuk memulai navigasi

Form yang dibutuhkan dalam pembuatan UI dapat diambil dengan melakukan cara *drag and drop widget. Form widget* telah tersedia pada Eclipse seperti yang terlihat pada gambar dibawah ini :



Gambar 3.3 Tampilan UI yang dirancang pada Eclipse

Pada pembuatan UI diatas dilakukan dengan cara *drag and drop widget* yang kemudian ditambahkan pemrograman dengan menggunakan bahasa xml. Terdapat beberapa *icon setting* yang digunakan untuk mensetting nilai variabel-variabel kontrol pada UI diatas. Berikut adalah penjelasan tentang UI diatas :

- Arah kompas :
  Arah kompas menunjukkan arah dari kompas pada *device* android dalam radius 360^{o}.

- Arah Tujuan :
  Arah Tujuan merupakan arah yang akan dituju oleh robot dalam satuan derajat (^{o})
- Jarak :
  Jarak menunjukkan seberapa jauh jarak antara posisi robot dengan target / tujuan dalam satuan meter (m)
- Setir :
  Setir merupakan perintah yang akan dikirimkan ke mikrokontroler, perintah tersebut adalah Lurus,Kanan dan Kiri.
- Tujuan
  - Lintang :
    Merupakan *form manual input* untuk mengisiskan posisi / nilai lintang yang akan dituju.
    Misal : -7.2892740
  - Bujur :
    Sama seperti lintang, bujur juga merupakan *form manual input* untuk mengisiskan posisi / nilai bujur yang akan dituju.
    Misal : 112.7684280
- GPS :
  - Lintang :
    Menunjukkan posisi lintang dari *device* android berada.
  - Bujur :
    Menunjukkan posisi bujur dari *device* android berada.
  - Akurasi :
    merupakan data real yang menyatakan seberapa akurat posisi *user* (dalam meter).

UI yang dirancang pada tugas akhir ini ditargetkan pada *smartphone* Motorola XT-530 dengan OS Android v2.3.4 (Gingerbread) yang memiliki resolusi 320x480 pixel dengan lebar layar 3,5 *inches*.

3.1.2 *Access permission*

Pada sistem yang akan dibuat ini, perlu adanya proses *access permission. Access permission* adalah izin penggunaan fitur yang akan digunakan, fitur-fitur yang akan digunakan tersebut bersifat personal pada *device* Android.

Google ingin melindungi pemilik *device* android dari pencurian dan penyalahgunaan data yang dapat merugikan pemilik *device* serta

melindungi *firmware* dari *device* android yang sedang dikembangkan dari adanya *malware* atau virus yang dapat merusak data. Untuk itu google menyediakan tempat khusus untuk mencatumkan *access permission* untuk menggunakan fitur ftur yang ada. Tempat tersebut dinamakan AndroidManifest.xml dimana file ini telah di *generate* secara otomatis oleh eclipse sebagai *Software* tempat pembuatan aplikasi android. *Access permission* tidak perlu dituliskan untuk seluruh fitur pada *device* android, cukup untuk fitur fitur yang akan digunakan dalam mengembangkan aplikasi android.

Pada sistem yang dirancang ini, ada beberapa fitur yang bersifat personal yang akan dipakai yakni fitur GPS, Internet dan *Bluetooth*, maka untuk mendaftarkannya pada file AndroidManifest.xrnl yaitu dengan menambahkan beberapa *syntax* berikut :

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

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

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

3.1.3 Variable linking

Pada pemrograman Android menggunakan Eclipse IDE bahasa yang digunakan adalah bahasa JAVA. Disini, sebuah *looping* proses yang dijalankan dikatakan sebagai sebuah *Activity*. Pada Eclipse IDE dapat dibuat beberapa macam *Activity* secara sekaligus secara simultan tanpa harus saling menggangu satu sama lainnya. Pada sistem ini akan dirancang sebuah sistem dengan 2 *Activity* untuk memproses data GPS, Kompas dan berhubungan dengan *User Interface*. Sedangkan *Activity* selanjutnya untuk proses komunikasi dari *device* Android ke mikrokontroler melalui *Bluetooth.*

Activity pertama untuk memproses GPS dan kompas akan ditampilkan pada *User Interface* program, maka harus dilakukan suatu *linking* antara variabel yang digunakan pada *User Interface* dengan

Variabel yang akan dijalankan pada *Activity*. Dengan melakukan *linking*, maka dapat dipantau perubahan variabel tersebut pada UI yang dibangun. Sebelum melakukan *linking*, maka harus dipanggil *Library* untuk *linking* dengan *syntax* berikut :

```
import android.widget.Button;

import android.widget.EditText;
```

Kemudian, cara untuk melakukan *linking* adalah dengan menggunakan *syntax* sebagai berikut :

```
Tulis1=(TextView)findViewById(R.id.text1);

Lintang=(TextView)findViewById(R.id.lintang);

Bujur=(TextView)findViewById(R.id.bujur);

Akurasi=(TextView)findViewById(R.id.akurasi);

GPSstat=(TextView)findViewById(R.id.GPSStatus);

SavLintang=(TextView)findViewById(R.id.savLintang);

SavBujur=(TextView)findViewById(R.id.savBujur);

arah=(TextView)findViewById(R.id.arah);

radius=(TextView)findViewById(R.id.jarak);

setir=(TextView)findViewById(R.id.setir);

Edit1=(EditText)findViewById(R.id.editText1);

Edit2=(EditText)findViewById(R.id.editText2);
```

```
button_lintang=(Button)findViewById(R.id.button_lintang);

button_bujur=(Button)findViewById(R.id.button_bujur);

button1=(Button)findViewById(R.id.button1);

button2=(Button)findViewById(R.id.button2);
```

Pada *syntax* diatas, sisi kiri bertulisan biru merupakan variabel yang digunakan didalam *Activity*, sedangkan pada sisi kanan merupakan variabel yang didefinisikan pada main.xml pada rancangan *User Interface*. Terdapat beberapa tipe yang digunakan pada *syntax* diatas, yaitu tipe *TextView, Edit text,* dan *Button*. Tipe tersebut mewakili jenis apakah variabel berikut :

- *TextView* mewakili Tulisan/Text
- *Edit Text* mewakili form yang bisa diisi dan diubah
- *Button* mewakili tombol

3.1.4 Registrasi sensor

Setelah melakukan *variable linking* langkah selanjutnya adalah meregistrasikan pada *Activity* tersebut, sensor apakah yang akan digunakan pada *Activity* tersebut. Langkah ini perlu dilakukan disamping meregistrasikan *permission* pada Android Manifest. Berbeda tujuan dengan *permission* atau izin pada AndroidManifest, registrasi sensor disini berfungsi untuk menggunakan sensor tersebut secara real atau nyata. Sebelum itu, juga harus dipanggil *Library* dari sensor tersebut yakni dengan perintah sebagai berikut :

```
import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;
```

```
import android.hardware.SensorManager;

import android.location.Location;

import android.location.LocationListener

import android.location.LocationManager;
```

Library sensor diatas merupakan *Library* sensor secara umum. Pada *Library* "android.*hardware*.sensor" telah mencakup akses untuk sensor kompas, accelerometer, proximity,dll. Sedangkan *Library* sensor untuk lokasi / GPS pada "android.location.Location". Setelah mengambil *Library* dari sensor yang akan digunakan, maka selanjutnya adalah melakukan registrasi sensor pada program. Hal ini dapat dilakukan dengan memanggil "Sensor Manager" dan "Location Manager" dengan *syntax* sebagai berikut :

```
//registrasi GPS

LocationManager mlocManager =
(LocationManager)getSystemService(Context.LOCATION_SE
RVICE);

mlocManager.requestLocationUpdates(
LocationManager.GPS_PROVIDER, 0, 0, myListener3);


//registrasi kompas
sensorManager=(SensorManager)getSystemService(SENSOR_
SERVICE);

sensorManager.registerListener(myListener1,sensorMana
ger.getDefaultSensor(Sensor.TYPE_ORIENTATION),SensorM
anager.SENSOR_DELAY_FASTEST);
```

3.1.5 Global Positioning System

Sensor GPS menangkap sinyal dari satelit dan melakukan penghitungan untuk memperkirakan dimana posisi *user* berada. Hal yang sangat berpengaruh dalam bidang navigasi adalah tingkat akurasi dari GPS yang digunakan, mengingat semakin tinggi akurasi GPS, maka data posisi yang didapat semakin *valid*.

Dari masalah diatas, maka sensor GPS pada android diharuskan memiliki kemampuan untuk melihat seberapa besar akurasi dari posisi yang sekarang didapat, disamping data posisi lintang dan posisi bujur dari GPS. Hal ini diperlukan karena apabila akurasi GPS lebih besar dari 10 meter, maka navigasi akan sulit untuk dilakukan mengingat bahwa posisi dapat terus berubah sehingga data menjadi tidak *valid*. Untuk itu, maka dibutuhkan minimal 3 jenis data yang harus ditampilkan pada *user interface* yakni data lintang, data bujur, dan data akurasi. Pada *Software* Eclipse IDE for Android data-data diatas dapat dipanggil dengan menggunakan *syntax* sebagai berikut :

```
Lintang1 = loc.getLatitude();
```

loc.get*Latitude* yang berfungsi untuk mendapatkan data koordinat lintang.

```
Bujur1 = loc.getLongitude();
```

loc.get*Longitude* yang berfungsi untuk mendapatkan data koordinat bujur.

```
accuracy = loc.getAccuracy();
```

loc.getAccuracy yang berfungsi untuk mendapatkan data akurasi.

Pada *syntax* yang digunakan diatas, data Lintang dan data Bujur merupakan data derajat dalam bentuk bilangan *real* dengan *range* -90 hingga 90 derajat untuk Lintang dan -180 hingga 180 derajat untuk Bujur. Sedangkan data akurasi merupakan data real yang menyatakan seberapa akurat posisi *user* (dalam meter). Umumnya data akurasi berkisar antara 3 meter (akurasi maksimal) hingga 100 meter atau lebih.

Untuk mendapatkan data hasil input *user*, maka kita memanfaatkan EditText yang telah disiapkan pada UI yang dirancang sebelumnya. Karena EditText merupakan data yang bersifat umum (TEXT/ASCII) dan data yang kita butuhkan merupakan data berupa data *double* maka

kita harus menerjemahkan/menconvert data dari EditText menjadi data double. Kekurangannya adalah apabila ada data yang salah atau tidak *valid* , semisal memasukkan huruf bukannya angka, maka akan menyebabkan aplikasi akan melakukan *force close.* Untuk perlu diperhatikan bahwa data yang dimasukkan haruslah benar yakni berupa angka/bilangan. Data dari Lintang dapat diambil dan dikonversi dengan menggunakan perintah berikut :

```
Lintang2=
Double.parseDouble(Edit1.getText().toString());

SavLintang.setText("Tujuan (Lintang) : "+Lintang2);

Bujur2=
Double.parseDouble(Edit2.getText().toString());

SavBujur.setText("Tujuan (Bujur)  : "+Bujur2);
```

Apabila data telah berhasil didapat, maka Lintang dan Bujur tujuan akan tersimpan dalam variabel Bujur2 dan Lintang2 dan koordinat tujuan pada UI akan terupdate secara otomatis.

Pada tahap ini, telah didapat 2 variabel posisi yakni posisi saat ini dan posisi tujuan. Data yang kita inginkan adalah *Heading* dari posisi awal menuju posisi target. Secara matematis, arah *heading* dari 2 posisi koordinat lintang dan bujur dapat dihitung dengan menggunakan fungsi *atan* ataupun *atan2.* Namun pada Android telah diberikan perintah khusus yang telah dapat digunakan untuk menghitung *heading* dari 2 posisi tersebut.

```
//Current Position
      Location A = new Location("titik A");
      A.setLatitude(Lintang1);
      A.setLongitude(Bujur1);

//Target Position
      Location B = new Location("titik B");
      B.setLatitude(Lintang2);
      B.setLongitude(Bujur2);
```

Keluaran dari perintah khusus tersebut adalah masih berupa sudut -180 hingga +180 derajat. Agar sama levelnya dengan arah mata angin, maka derajat negatif harus dirubah dengan ditambahkan 360. Dengan ini maka data *heading*/*bearing* dari navigasi telah dapat digunakan sebagai patokan.

3.1.6 Sensor Kompas

Setelah data dari GPS telah *valid* dan dapat digunakan, maka selanjutnya adalah mengolah data dari kompas.

Untuk mendapatkan data arah saat ini, maka yang harus dlakukan adalah memberikan *syntax* untuk mengaktifkan sensor kompas pada *device* android. Sensor kompas yang ada pada *device* Android merupakan sensor orientasi 3-axis yang digunakan untuk mengukur *pitch, yaw*, dan *roll. Pitch* adalah sudut putaran menuju sumbu y dan z dengan sumbu x sebagai tumpuan. *Roll* adalah sudut putaran menuju sumbu x dan y dengan sumbu z sebagai tumpuan. *Yaw* adalah sudut putaran menuju sumbu x dan z dengan sumbu y sebagai tumpuan. Untuk mendapatkan data pitch,roll dan yaw dapat dilakukan dengan menuliskan *syntax* sebagai berikut :

```
float azimuth=event.values[0];

float pitch=event.values[1];

float roll=event.values[2];
```

Syntax diatas merupakan *syntax* untuk mendapatkan data yaw dengan menuliskan *syntax* event.values[0] , data pitch dengan menuliskan *syntax* event.values[1], dan data *roll* dengan menuliskan *syntax* event.values[2]. Namun pada tugas akhir ini data yang diingkan hanya data *yaw* karena sensor (*device* android) berada pada posisi sudut x dan z (sejajar dengan permukaan bumi) dan tidak memiliki *pitch* dan *roll* sehingga dapat dikatakan bahwa sudut *yaw* tersebut merupakan orientasi sudut yang merujuk pada medan magnet bumi dengan titik 0^o sebagai acuan arah utara.

Apabila sensor (dalam hal ini adalah *device* Android) diasumsikan sebagai benda dengan posisi sudut x dan z yang tetap atau tidak

memiliki *pitch* dan *roll*, maka kondisi sudut *yaw* dari *device* android dapat dikatakan sebagai data orientasi sudut yang merujuk pada medan magnet utara bumi sebagai titik 0 derajatnya. Dari kondisi ini dapat dikatakan bahwa kondisi sudut yaw pada sensor orientasi Android dapat dijadikan kompas magnetik.

3.1.7 Kalkulasi Data Sensor

Kalkulasi data sensor dilakukan untuk mengolah data yang sudah didapat dari sensor GPS dan sensor kompas lalu dijadikan perintah untuk mengarahkan kemudi *mobile* robot serta mengatur kecepatan motor. Pada tugas akhir ini,terdapat tiga arah kemudi yang digunakan yakni kiri, kanan dan lurus. Sedangkan untuk kecepatan yang digunakan juga ada tiga yakni kecepatan tinggi, kecepatan rendah dan berhenti. Kecepatan motor akan dipengaruhi oleh arah kemudi saat ini.

Tabel 3.1 Pengaruh Arah kemudi terhadap kecepatan Motor

Arah Kemudi	Kecepatan Motor
Kanan	Kecepatan rendah
Kiri	Kecepatan rendah
Lurus	Kecepatan tinggi

Kondisi diatas hanya berlaku ketika tombol mulai navigasi telah ditekan atau dengan kata lain navigasi pada posisi ON. Selama tombol navigasi tidak ditekan atau OFF, maka kecepatan motor akan berada pada kecepatan 0 atau berhenti.

Apabila jarak antara posisi target dan posisi saat ini adalah kurang dari 5 meter, maka navigasi akan otomatis dihentikan. Hal ini dilakukan karena sistem navigasi GPS tidak mampu melakukan deteksi jarak dibawah 5 meter, sehingga apabila dipaksakan, maka navigasi akan terus dilakukan karena dianggap belum mencapai target. Untuk pemrograman pada Eclipse, digunakan perintah sebagai berikut :

```
int selisih = (int) ((hadap)-(kompas));

//Diberi radius 10 derajat arah kanan dan kiri
if (selisih<10 && selisih>-10)
{
      setir.setText("lurus");
```

```
        heading = 90;
        temp_PWM=90; //KECEPATAN TINGGI
        }
else
{
temp_PWM = 85; //KECEPATAN RENDAH
if (hadap<kompas)
{
hadap=hadap+360;
if ((hadap-kompas)<180)
{
setir.setText("kanan");
heading = 40;
}
else
{
        setir.setText("kiri");
        heading = 140;
}
}
        else
{
kompas = kompas+360;
if ((hadap-kompas)>-180)
{
        setir.setText("kiri");
        heading = 140;
}
else
{
        setir.setText("kanan");
    heading = 40;
}
        }
        }

        if (navigasi==0) {
        PWM=80;  //PWM FOR STOPPING
```

```
}
else if (navigasi == 1 ) {
if (dist > 5){
PWM=temp_PWM;
}
else {
PWM=80;
}
```

3.1.8 Komunikasi *Bluetooth*

Pada perancangan selanjutnya adalah perancangan sistem komunikasi yang akan menghubungkan antara *device* android dengan mikrokontroler. Pada tugas akhir ini komunikasi yang akan digunakan adalah dengan memanfaatkan *Bluetooth* pada *device* android.

Pemilihan *Bluetooth* sebagai protokol komunikasi antara android dan mikrokontroler adalah berdasarkan pada sistem navigasi yang akan dirancang pada Tugas Akhir ini, beberapa hal tersebut diantaranya sebagai berikut :

- Komunikasi yang dilakukan adalah jarak dekat (Android diletakkan pada badan *mobile* robot sebagai sensor utama)
- Robot Bersifat *Mobile*
- Data harus *real time* dan cepat

Berdasarkan pada beberapa hal diatas, maka protokol yang paling cocok digunakan yakni *Bluetooth* dan USB. *Bluetooth* dan USB memiliki kelebihan dan kekurangan masing-masing, diantaranya adalah:

Tabel 3.2 Perbandingan antara Bluetooth dan USB

Jenis Komunikasi	Keuntungan	Kekurangan
Bluetooth	Komunikasi Nirkabel Jarak hingga 10 meter. Transfer rate menengah (9600bps – 57600bps). Protokol Komunikasi	Membutuhkan daya lebih.

	cukup mudah (Tx dan Rx) seperti pada Serial.	
USB	Tidak menghabiskan daya (Android dapat di-charge sekaligus saat Transfer data). Transfer rate sangat cepat. Lebih tahan terhadap noise/kesalahan data	Komunikasi masih menggunakan kabel. Jarak maksimal <10 meter (Untuk menghindari *data losses*)

Dari tabel diatas dapat diketahui kelebihan dan kekuangan dari *Bluetooth* dan USB, Namun pada tugas akhir ini komunikasi yang akan digunakan adalah *Bluetooth*. *Bluetooth* dianggap lebih mudah dalam komunikasi (Tx dan Rx) dan tanpa adanya penggunaan kabel dianggap lebih efisien.

Untuk dapat memanfaatkan *Bluetooth* pada *device* android, maka hal pertama yang harus dilakukan adalah dengan pemanggilan *Library* dengan cara menambahkan *syntax* berikut :

```
import android.Bluetooth.BluetoothAdapter;
import android.Bluetooth.BluetoothDevice;
import android.Bluetooth.BluetoothSocket;
```

Setelah mengambil *Library* dari *Bluetooth* yang akan digunakan, maka selanjutnya adalah melakukan registrasi *Bluetooth* tersebut pada program, dengan menambahkan *syntax* berikut :

```
//registrasi Bluetooth
btAdapter = BluetoothAdapter.getDefaultAdapter();
checkBTState();
```

Nantinya sistem mikrokontroler harus menyesuaikan *Bluetooth* pada *device* yang akan digunakan agar dapat melakukan komunkasi ke mikrokontroler. Perintah yang digunakan pada main *Activity* adalah sebagai berikut :

```
private BluetoothSocket
createBluetoothSocket(BluetoothDevice device) throws
IOException {

if(Build.VERSION.SDK_INT >= 10){
try {
final Method  m =
device.getClass().getMethod("createInsecureRfcommSock
etToServiceRecord", new Class[] { UUID.class });

return (BluetoothSocket) m.invoke(device, MY_UUID);
} catch (Exception e) {
Log.e(TAG, "Could not create Insecure RFComm
Connection",e);
}
}

return
device.createRfcommSocketToServiceRecord(MY_UUID);
}
```

Setelah mendefinisikan perintah diatas, maka Android telah membuka jalur komunikasi *Bluetooth* dan siap melakukan komunikasi.

Pengiriman data dari Android ke board mikrokontroler adalah bersifat *Asynchronous* atau tak teratur, bukan bersifat *Synchronous*. Hal ini menyebabkan data baru dikirimkan bila ada suatu kejadian yang men-*trigger* pengiriman data. Maka dari itu, dengan melihat *flowchart* sistem Android maka dapat disimpulkan bahwa komunikasi pengiriman data dilakukan saat terjadi perubahan posisi saat ini dan perubahan arah kompas. Hal ini menekankan bahwa apabila terjadi perubahan posisi dan arah, maka *mobile* robot harus segera melakukan penyesuaian menuju posisi yang diharuskan.

Karena data yang dikirimkan ada dua jenis, yakni data kemudi dan data kecepatan, maka data yang dikirimkan akan dibentuk sebuah *array* yang berurutan. Tujuan penggunaan *array* ini adalah agar data dapat diterima dengan baik oleh mikrokontroler sehingga tidak terjadi kesalahan penerimaan data atau data yang diterima menjadi terbalik.
Perintah yang digunakan adalah sebagai berikut.

```
public void onSensorChanged(SensorEvent event) {

      if(event.sensor.getType()==Sensor.TYPE_ORIENTA
TION){
try
      {
      //Send heading state and PWM state
      server.send(new byte[]
{
(byte)heading,(byte)PWM});
      }
catch (IOException e)
      {
Log.e("Seeeduino ADK", "problem sending TCP message",
e);
      }
      }
```

Pada *Syntax* diatas, variabel *heading* merupakan variabel yang merujuk pada posisi motor DC untuk kemudi sedangkan variabel PWM adalah variabel yang merujuk pada kecepatan.

3.1.9 Pengambilan Data Posisi Target

Pada tugas akhir ini, posisi target tujuan akan diisikan melalui *manual input* pada UI. Untuk mendapatkan posisi target tujuan yang berupa nilai lintang dan bujur, diambil menggunakan aplikasi Google Maps.

Melalui aplikasi google maps yang ditampilkan pada *device* android bisa didapatkan posisi target yang hendak dituju dengan cara men-tab pada layar *device* android ke posisi yang dituju. Sehingga akan ditampilkan *direction* / arah jalan dari posisi *device* android ke posisi tujuan yang telah ditentukan, seperti yang terlihat pada gambar di bawah ini :

Gambar 3.4 Tampilan Lintang bujur pada Google Maps

Dari gambar 3.4 diatas dapat dilihat bahwa telah didapatkan nilai dari lintang dan bujur posisi target tujuan yang hendak dituju, pada kasus diatas terlihat nilai dari lintang dan bujur.

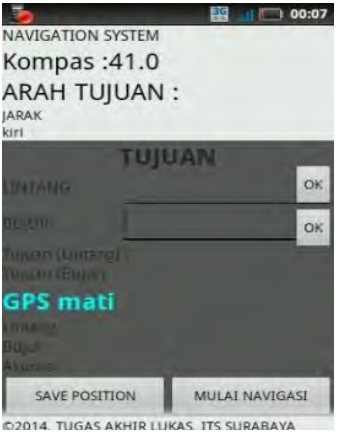
Lintang : -7.2892740
Bujur : 112.7684280

Selain nilai lintang dan bujur, dapat dilihat juga pada gambar diatas ditampilkan jalur yang nantinya akan dilalui oleh *mobile* robot. Namun pada navigasi point to point ini, jalur yang akan diambil oleh mobile robot bukan mengacu pada jalur yang di tentukan oleh google maps, melainkan menetukan jalur terdekat dengan menarik garis lurus..

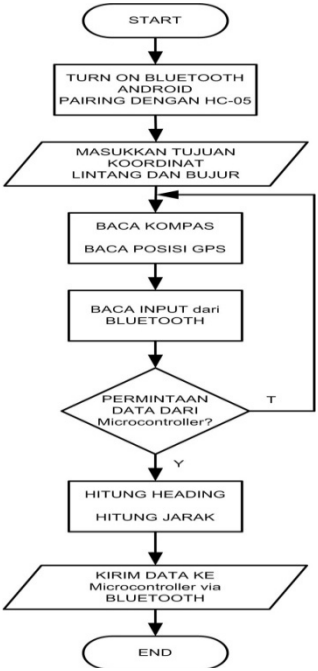
Gambar diatas merupakan tampilan standar dari google maps, namun pada goole maps juga dapat menampilkan maps dengan tampilan satelite. Seperti gambar dibawah :

Gambar 3.5 Tampilan Maps mode satelit pada Google Maps

Setelah didapatkan nilai lintang dan bujur maka selanjutnya nilai tersebut dapat di inputkan pada *form* manual *input* lintang dan bujur pada UI yang telah dibuat.



Gambar 3.6 Tampilan UI untuk menginput lintang dan bujur target



Gambar 3.7 Flowchart Perancangan Sistem pada Android

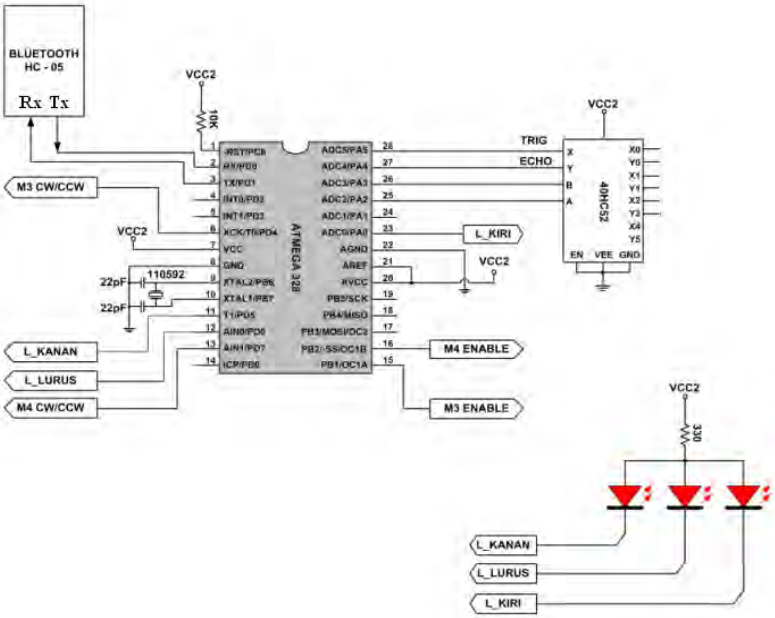
3.2 Perancangan *Hardware*

Pada perancangan *hardware* ini akan meliputi beberapa bagian, diantaranya perancangan sistem mikrokontroler secara *hardware* dan *Software*. Perancangan *Hardware* yang dilakukan adalah meliputi perancangan *board* mikrokontroler yang digunakan dan bagaimana relevansinya dengan tugas akhir yang dibangun saat ini. Sedangkan pada bagian perancangan *Software*, yang dilakukan adalah proses perancangan untuk pemrograman dari mikrokontroler yang bersangkutan.

3.2.1 Perancangan Sistem Mikrokontroler

Hardware yang digunakan pada Tugas akhir ini adalah Mikrokontroler dengan *minimum system* yang digunakan adalah mikrokontroler ATMega328P sebagai CPU utamanya.

ATMega328 merupakan salah satu mikrokontroler 8 bit buatan Atmel untuk keluarga AVR.



Gambar 3.8 Skema perancangan Board Mikrokontroler

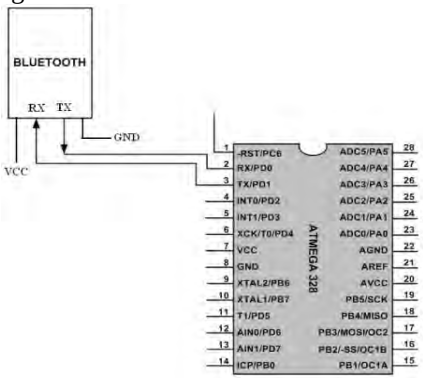
Board mikrokontroler diatas merupakan *minimum system* utama yang berperan penting pada *mobile* robot ini. Pada *minimum system* ini terdiri dari mikrokontroler ATMega328 dengan xtal 11,0592 MHz. Xtal / kristal tersebut digunakan sebagai pembangkit clock (osilator), dimana setiap 1 instruksi/perintah dalam program dieksekusi dalam 1 siklus clock.

Fungsi dari port-port lainnya yang digunakan pada project ini adalah sebagai berikut :

1. Port A2-A5 : Digunakan oleh Multiplexer CD4052 yang nantinya sebagai selector untuk sensor ultrasonk HC-SR04.
2. Port A0 : Digunakan sebagai pin indikator LED kemudi kiri.
3. Port B1 : Digunakan sebagai pin enable motor 3 (Motor kanan).
4. Port B2 : Digunakan sebagai pin enable motor 3 (Motor kanan).
5. Port D0-D1 : Digunakan sebagai pin Tx dan Rx pada modul bluetooth HC-05.
6. Port D4 : Digunakan sebagai pin kontrol rotasi motor (CW (clockwise)/CCW(counter clockwise)).
7. Port D5 : Digunakan sebagai pin indikator LED kemudi kanan.
8. Port D6 : Digunakan sebagai pin indikator LED kemudi kanan.
9. Port D7 : Digunakan sebagai pin kontrol rotasi motor (CW (*clockwise*)/CCW(*counter clockwise*).

3.2.2 Komunikasi *Bluetooth* HC-05

Untuk dapat berkomunikasi dengan *Bluetooth* pada Android, maka mikrokontroler juga harus menyesuaikan sistem kedalam sistem *Bluetooth* yang dalam hal ini memanfaatkan modul *Bluetooth* HC-05. Modul *Bluetooth* HC-05 dapat langsung dihubungkan ke mikrokontroler ATMega328 sebagai berikut.

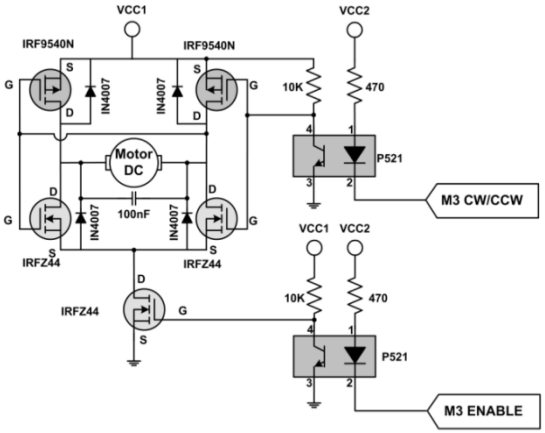


Gambar 3.9 Skema komunikasi *Bluetooth* HC-05 dengan Atmega328

Dari gambar 3.9 diatas terlihat bahwa TX dan RX pada *Bluetooth* HC-05 dapat langsung dihubungkan pada pin TX (port D1) dan RX (pin D0) yang telah tersedia pada ATMega328. Untuk menghubungkan antara TX dan RX pada *Bluetooth* HC-05 dengan TX dan RX pada mikrokontroler dilakukan dengan menghubugkan secara silang, yaitu TX pada *Bluetooth* HC-05 dihubungkan dengan RX pada mikrokontroler, sedangkan RX pada *Bluetooth* HC-05 dihubungkan dengan TX pada mikrokontroler. *Bluetooth* HC-05 memiliki tegangan 5 volt sehingga bisa langsung memanfaatkan *power suppy* pada minimum sistem.

3.2.3 Kontrol motor DC

Fungsi utama dari mikrokontroler pada tugas akhir ini adalah sebagai konverter data dari Android ke *Mobile* robot yang digunakan. Apabila komunikasi USB telah diinisiasikan dengan Android, tugas selanjutnya adalah melakukan penerjemahan data dan konversi dari data dari Android menuju *mobile* robot yang digunakan.



Gambar 3.10 Rangkaian driver motor DC dengan H-Bridge

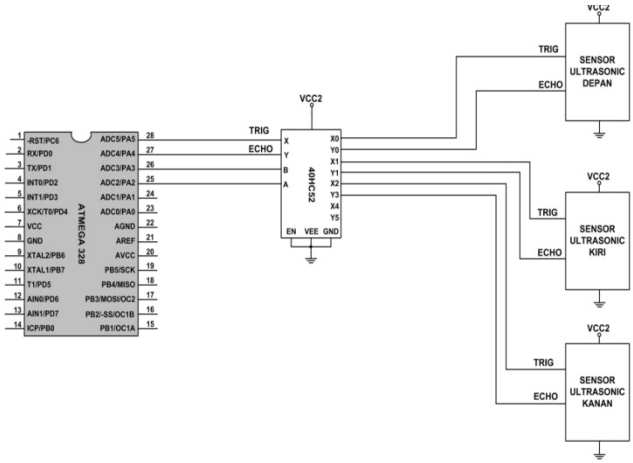
Sistem kontrol motor DC yang digunakan pada sistem ini yaitu dengan H-*Bridge* yang pada pada dasarnya adalah 4 buah transistor yang difungsikan sebagai saklar. Pengaturan motor DC yaitu meliputi

kecepatan dan arah putaran yang dapat di set ke CW (clockwise) yang berarti putarannya searah jarum jam dan CCW (counter clockwise) yang berarti putarannya berlawanan dengan arah jarum jam. Pengaturan arah yaitu dengan cara membalik tegangan logika masukan H-*bridge*. Sedangkan sistem kontrol kecepatan motor DC digunakan prinsip PWM (*Pulse Width Modulator*) yaitu suatu metode pengaturan kecepatan putaran motor DC dengan mengatur lamanya waktu pensaklaran aktif (*Duty Cycle*). Motor DC merupakan sebuah komponen yang memerlukan arus yang cukup besar untuk menggerakannya. Oleh karena itu motor DC biasanya memiliki penggerak tersendiri. Pada tugas akhir ini motor DC akan digerakkan dengan menggunakan PWM yang telah terintegrasi dengan rangkaian H-*Bridge*. Dengan rangkaian H-*Bridge* yang memiliki input PWM ini, maka selain arah kita juga bisa mengendalikan kecepatan putar motor DC tersebut.

3.2.4 Perancangan Sensor Ultrasonik HC-SR04

Perancangan selanjutnya adalah perancangan dari sensor ultrasonik yang nantinya sensor ini difungsikan untuk mendeteksi rintangan yang menghalangi laju *mobile* robot.

Sensor yang digunakan adalah berupa modul sensor ultrasonik HC-SR04, Pada sensor ultrasonik HC-SR04 terdapat 4 pin yang digunakan yaitu vcc, ground,trig dan echo. Pin trig dan echo dapat langsung dihubungkan dengan mikrokontroler tanpa tambahan komponen apapun. Namun karena pada tugas akhir ini menggunakan 4 sensor ultrasonik yang harus aktif secara bersamaan,sehingga akan membuuhkan banyak ruang pada mikrokontroler, maka digunakan Multiplexer tipe CD4052 sebagai selector.



Gambar 3.11 Skema ultrasonik HC-SR04 dengan Mux CD4052

Agar Ultrasonik dapat terbaca atau aktif maka harus dilakukan penyetingan pada selector di pin A dan B, Sebagai berikut:

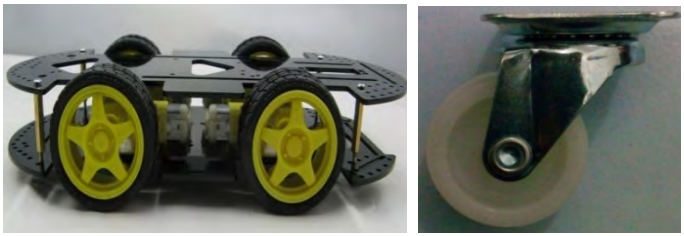
Tabel 3.3 Seting selector Mux CD4052 Untuk Pembacaan ultrasonik

Pin B (mux CD4052)	Pin A (mux CD4052)	Ultrasonik Terbaca
0	0	Ultrasonik depan (1)
0	1	Ultrasonik kanan (2)
1	0	Ultrasonik belakang (3)
1	1	Ultrasonik kiri (4)

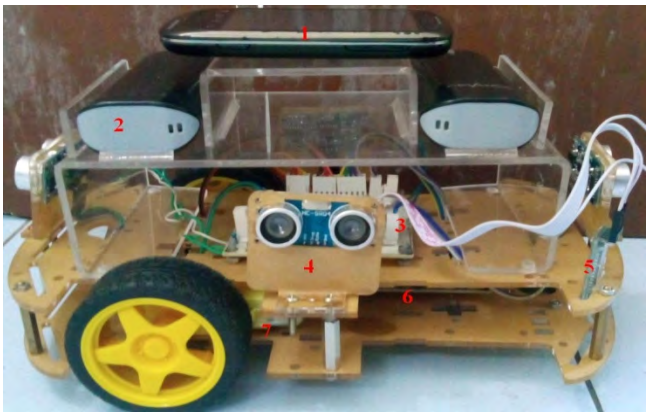
Dari tabel 3.3 diatas menunjukan pengaktifan dari ultrasonik, bahwa ketika pada mux CD4052 pin B di set pada logika 0 dan pin A di set pada logika 1 maka sensor ultrasonik depan (1) akan aktif, begitu juga selanjutnya bahwa ketika pin B di set 0 pin A di set 1 maka ultrasonik kanan (2) aktif, ketika pin B di set 1 pin A di set 0 maka ultrasonik belakang (3) aktif, dan ketika pin B di set 1 pin A di set 1 maka ultrasonik kiri (4) aktif.

3.2.5 Perancangan Mekanik *Mobile* Robot

Mobile robot yang digunakan pada tugas akhir ini adalah 4WD Robot *smart car*. Pada *mobile* robot ini, bagian yang akan dikontrol adalah arah kemudi (*steering*) dan kecepatan (*speed*), oleh karena itu perancangan mekanik dari *mobile* robot ini merupakan hal penting untuk diperhatiakn. Nantinya, akan ada penambahan chassis pada *mobile* robot untuk meletakkan *device* android dan baterai. Selain itu akan ada perubahan pada roda depan, hal ini dikarenakan ketika *mobile* robot tetap menggunakan 4 roda akan sulit dalam hal pengontrolan ketika berbelok. Maka dari itu roda depan akan digantikan dengan roda bebas untuk mempermudah pengontrolan saat belok.



Gambar 3.12 Robot 4WD *smart car* (kiri), Roda bebas (kanan)



Gambar 3.13 Konstruksi *mobile* robot setelah modifikasi

Keterangan :

1. *Device* Android
2. Baterai 5V
3. Board Mikrokontroler
4. Ultrasonik HC-SR04
5. Modul Bluetooth HC-05
6. Driver Motor DC
7. Motor DC

3.2.6 Perancangan *Software* Mikrokontroler

Software yang dirancang pada mikrokontroler ini menggunakan *Software* Bascom AVR. Pada *Software* Bascom AVR ini bahasa pemrograman yang digunakan adalah bahasa Basic.

Setiap bahasa pemrograman memiliki standar penulisan program. Konstruksi dari program bahasa basic harus mengikuti aturan sebagai berikut :

```
$regfile = “header”
‘inisialisasi
‘deklarasi variabel
‘deklarasi konstanta
Do
‘pernyataan-pernyataan
Loop
End
```

```
$regfile = "m328pdef.dat"
$crystal = 11059200
$baud = 9600
```

$regfile = m328pdef.dat berisi deklarasi register dari mikrokontroler ATMega328. Sedangkan $crystal = 11059200 merupakan *crystal clock* (xtal) yang digunakan adalah 11.059600 MHz, dan $baud = 9600 adalah *baudrate* yang digunakan dalam satuan bps (*bit rate persecond*).

3.2.7 Perancangan *Software* Komunikasi *Bluetooth*

Bagian ini merupakan bagian yang digunakan untuk melakukan inisiasi komunikasi dengan *device* android. Untuk melakukan komunikasi Android, maka kita harus mendefinisikan komunikasi antara kedua *device* tersebut.

Untuk mendapatkan data yang dikirimkan oleh Android, maka kita harus menambahkan sebuah prosedur yang fungsinya menunggu data yang dikirimkan oleh Android. *Syntax* yang digunakan adalah sebagai berikut.

```
Baca_bluetooth:
Njupuk = ""
Blue_t(1) = ""
Blue_t(2) = ""
Print "Data"
Do
   If Rx_flag = 1 Then
      Reset Rx_flag
      If Rx_byte = 36 Then
         Njupuk = "" : I = 0
         Do
            If Rx_flag = 1 Then
               Reset Rx_flag
               Select Case Rx_byte
                  Case 44
                     Incr I
                     Blue_t(i) = Njupuk
                     Njupuk = ""
                  Case 35
                     Exit Do
                  Case Else
                     Njupuk = Njupuk + Rx_char
               End Select
            End If
         Loop
         Goto Akhir
      End If
   End If
```

```
Loop
Akhir:

Return
```

3.2.8 Perancangan *Software* Kontrol Motor DC

Setelah data hasil komunikasi dari Android telah didapat, maka yang selanjutnya dilakukan adalah mengkonversi data hasil komunikasi menjadi sinyal yang *usable* bagi motor DC.
Prosedur kontrol motor DC dan PWM ini dapat dilakukan dengan menambahkan *syntax* berikut.

```
Motor_maju_pelan:
   Set M3_putar : Set M4_putar
   Waitms 10
   Pwm1a = 800 : Pwm1b = 800
   Waitms 250
   Return
Motor_maju_cepat:
   Set M3_putar : Set M4_putar
   Waitms 10
   Pwm1a = 1000 : Pwm1b = 1000
   Waitms 250
   Return
Motor_mundur:
   Reset M3_putar : Reset M4_putar
   Pwm1a = 600 : Pwm1b = 600
   Waitms 20
   Return
Motor_kanan:
   Reset M3_putar : Set M4_putar
   Waitms 5
   Pwm1a = 800 : Pwm1b = 800
   Waitms 50
   Return
Motor_kiri:
   Set M3_putar : Reset M4_putar
```

```
   Waitms 5
    Pwm1a = 800 : Pwm1b = 800
   Waitms 50
   Return
Motor_off:
   Pwm1a = 0 : Pwm1b = 0
   Waitms 20
   Return
```

3.2.9 Perancangan *Software* Ultrasonik

Perancangan selanjutnya adalah perancangan software dari ultrasonik HC-SR04. Ultrasonik ini akan di seting agar mampu menghindari rintangan yang menghalangi laju dari *mobile* robot untuk menuju ke target tujuan. Untuk melakukan pembacaan pada ultrasonik dapat dilakukan dengan menggunakan syntax berikut :

```
Baca_sonic1:
   Reset Sel_a : Reset Sel_b
   Waitms 40
   Gosub Baca_sensor
   Sonic1 = Dta
   Waitms 2
   Return
Baca_sonic2:
   Set Sel_a : Reset Sel_b
   Waitms 40
   Gosub Baca_sensor
   Sonic2 = Dta
   Waitms 2
   Return
Baca_sonic3:
   Reset Sel_a : Set Sel_b
   Waitms 40
   Gosub Baca_sensor
   Sonic3 = Dta
   Waitms 2
```

```
   Return
Baca_sonic4:
   Set Sel_a : Set Sel_b
   Waitms 40
   Gosub Baca_sensor
   Sonic4 = Dta
   Waitms 2
   Return
```

Dari syntax diatas dapat dilihat bahwa untuk mengaktifkan sensor 1 (depan), 2 (kanan), 3 (belakang) dan 4 (kiri) pada ultrasonik HC-SR04 ini, dilakukan dengan men set dan reset selector A dan B pada multiplexer CD4052. Set berarti memberikan logika "1" dan reset berarti memberikan logika "0". Dalam kasus ini ketika reset selector A : reset selector B maka baca ultrasonik 1(depan), ketika set selector A : reset selector B maka baca ultrasonik 2 (kanan), reset selector A : set selector B maka baca ultrasonik 3 (belakang), dan ketika set selector A : set selector B maka baca ultrasonik 4 (kiri).

Setelah proses baca sensor berhasil, maka langkah selanjutnya adalah proses pengontrolan dari ultrasonik, dengan menambahkan syntax berikut :

```
Diam:
   Pwm1a = 0 : Pwm1b = 0
   Waitms 20
   Return

Lurus_pelan:
   Gosub Baca_sonic1
   If Sonic1 > 300 Then
      Gosub Motor_maju_pelan
      Waitms 100
   Else
      Gosub Cek_lurus
   End If
   Gosub Motor_off
   Return

Cek_lurus:
```

```
   Gosub Baca_sonic2
   Gosub Baca_sonic4
   If Sonic2 > Sonic4 Then
      Gosub Haluan_kanan
   Else
      Gosub Haluan_kiri
   End If
   Return

Lurus_cepat:
   Gosub Baca_sonic1
   If Sonic1 > 400 Then
      Gosub Motor_maju_cepat
      Waitms 100
   Else
      Gosub Cek_lurus
   End If
   Gosub Motor_off
   Return

Haluan_kanan:
   Gosub Baca_sonic2
   If Sonic2 > 250 Then
      A = 0
      Do
         Incr A
         Gosub Motor_kanan
         If A > 17 Then Goto Haluan_kanan_akhir
         Waitms 20
      Loop
   Else
      Gosub Lurus_pelan
   End If
Haluan_kanan_akhir:
   Gosub Motor_off
   Return

Haluan_kiri:
```

```
   Gosub Baca_sonic4
   If Sonic4 > 250 Then
       A = 0
      Do
         Incr A
         Gosub Motor_kiri
         If A > 15 Then Goto Haluan_kiri_akhir
         Waitms 20
      Loop
   Else
      Gosub Lurus_pelan
   End If
Haluan_kiri_akhir:
   Gosub Motor_off
   Return
```

Sensor Ultrasonik ini akan mendeteksi rintangan di depan, kanan dan kiri. Selain itu ultrasonik juga yang nantinya akan menjadi acuan kecepatan laju robot. Akan ada 2 kontrol kecepatan yaitu pelan dan cepat.

Dari syntax diatas dapat dilihat bahwa ketika ultrasonik 1 (depan) lebih besar dari 30cm maka motor akan maju pelan.

```
Lurus_pelan:
   Gosub Baca_sonic1
   If Sonic1 > 300 Then
      Gosub Motor_maju_pelan
```

ketika ultrasonik 1 (depan) lebih besar dari 40cm maka motor akan maju cepat.

```
Lurus_cepat:
   Gosub Baca_sonic1
   If Sonic1 > 400 Then
      Gosub Motor_maju_cepat
```

BAB IV

PENGUJIAN ALAT DAN ANALISA DATA

Pada bab ini akan dilakukan pengambilan data dan analisa data hasil implementasi dari alat yang telah di rancang. Beberapa pengujian yang akan dilakukan adalah pengujian akurasi kompas, pengujian pengiriman data, dan pengujian navigasi *mobile robot* dan Plot jalur hasil navigasi.

4.1 Pengujian Pengiriman Perintah Kemudi

Pada Pengujian ini bertujuan untuk mengetahui perintah kemudi yang dkirimkan oleh *device* android ke mikrokontroler terhadap arah kompas target yang telah ditentukan. Langkah pengujian yaitu pertama-tama memasukkan nilai lintang dan bujur target yang hendak di tuju, lalu aktifkan navigasi, maka akan didapatkan data arah kompas yang dituju (derajat). Dengan cara membandingkan posisi kompas *device* android dengan posisi kompas target akan diketahui pada sudut berapa perubahan perintah kemudi yang diberikan pada *device* android.

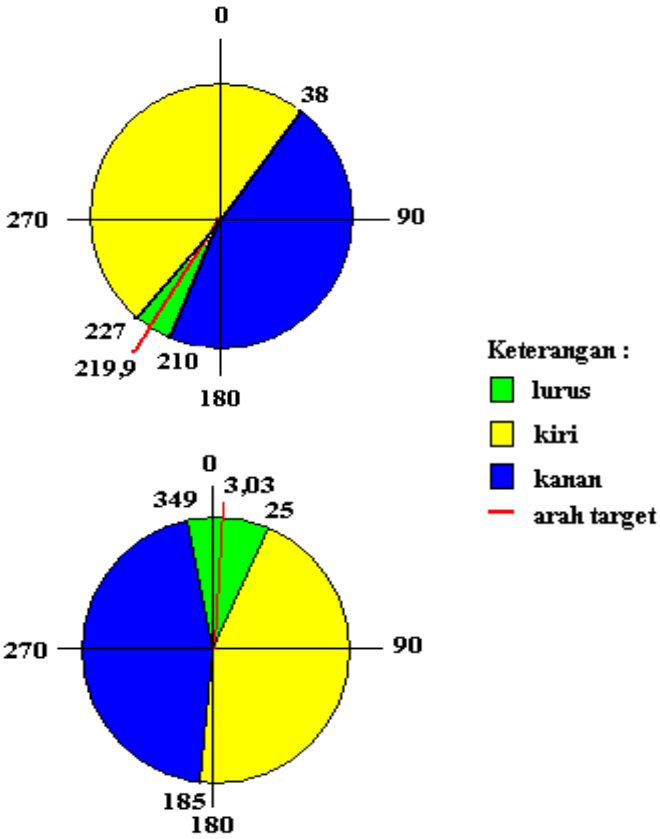
Pada pengujian ini dilakukan 2 kali pengujian, dengan target berbeda :

Pengujian 1 :
Lintang : -7.2895420
Bujur : 112.7697320
Pengujian 2 :
Lintang : -7.2874010
Bujur : 112.7701820

Tabel 4.1 Pengujian perintah kemudi Android ke mikrokontroller.

Pengujian	Arah Target (^{0} derajat)	Arah *device* (^{0} derajat)	Perintah kemudi
1	219,9	210-227	Lurus
	219,9	228-38	Kiri
	219,9	39-209	Kanan
2	3,03	349-25	Lurus
	3,03	26-185	Kiri
	3,03	186-348	Kanan

Dari hasil pengujian pada tabel 4.1 diatas, dapat digambarkan pada suatu grafik lingkar sebagai berikut :



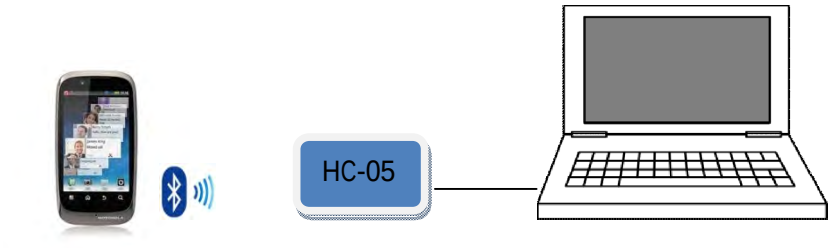
Gambar 4.1 Grafik perintah kemudi dalam sudut mengacu target GPS dan arah kompas target

Dari Gambar 4.1 diatas dapat dilihat bahwa target adalah garis berwarna merah, ketika arah posisi dalam hal ini *device* andoid pada area hijau maka *device* android akan memberikan perintah lurus, namun ketika arah *device* android pada area kuning maka *device* android akan

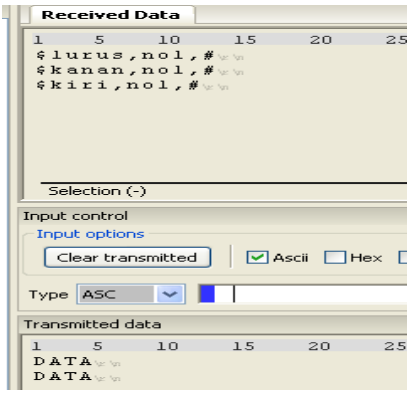
memberikan perintah kiri, sedangkan jika arah *device* android berada pada area biru maka *device* android akan memberikan perintah kanan.

4.2 Pengujian Komunikasi *Bluetooth*

Pada pengujian *Bluetooth* ini akan diuji komunikasi *Bluetooth* antara *Bluetooth* pada *device* android dengan module *Bluetooth* HC-05. Cara pengujiannya adalah dengan mengkomunikasikan *Bluetooth* HC-05 ke PC menggunakan USB TTL. Kemudian *Bluetooth* HC-05 di pair dengan *Bluetooth* pada *device* android. Untuk pengujiannya dalam hal ini memanfaatkan *Software* Hterm.

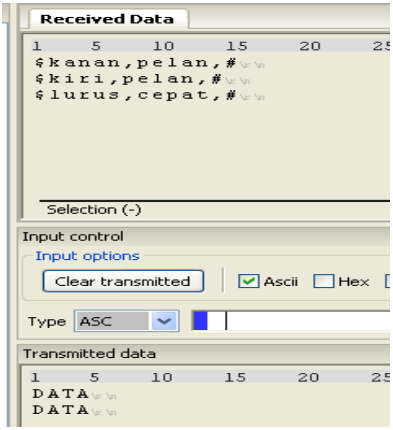


Gambar 4.2 Mekanisme pengujian bluetooth



Gambar 4.3 Pengujian komunikasi *Bluetooth* android dengan *Bluetooth* HC-05 pada posisi navigasi OFF.

Dari gambar 4.3 diatas dapat dilihat bahwa *Bluetooth* HC-05 dapat menerima data yang dikirimkan oleh *device* android. Perintah yang dikirimkan adalah perintah kemudi dan kecepatan. Pengujian diatas dilakukan pada saat posisi navigasi dimatikan / off, sehingga data yang diterima adalah data kemudi (lurus,kanan dan kiri) serta data kecepatan yang bernilai 0 (nol). Namun ketika navigasi pada posisi on / diaktifkan maka data yang dikirimkan berupa data kemudi (lurus,kanan dan kiri) serta kecepatan yang berupa pelan dan cepat. Pada posisi belok (kanan dan kiri) kecepatan akan pelan, sedangkan pada posisi lurus kecepatan akan cepat. Seperti yang terlihat pada gambar dibawah ini :



Gambar 4.4 Pengujian komunikasi *Bluetooth* android dengan *Bluetooth* HC-05 pada posisi navigasi ON.

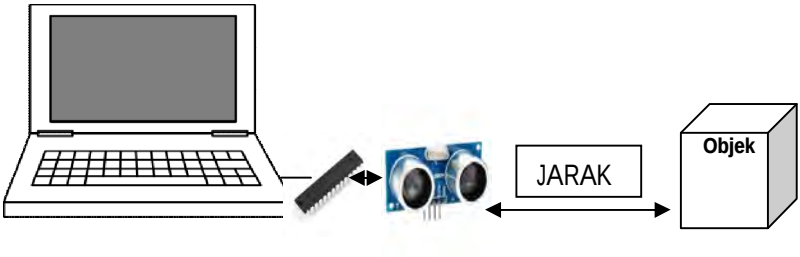
Tabel 4.2 hasil pengujian pengiriman perintah kemudi dari *device* android ke *Bluetooth* HC-05

Navigasi	**Perintah yang dikirim android**	**Perintah yang diterima *Bluetooth* HC-05**	**Keterangan**
OFF	Lurus	Lurus, 0	Sukses
	Kanan	Kanan, 0	Sukses
	Kiri	Kiri, 0	Sukses

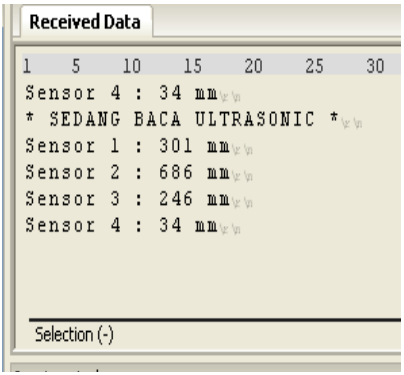
ON	Kanan	Kanan,pelan	Sukses
	Kiri	Kiri,pelan	Sukses
	Lurus	Lurus,cepat	Sukses

4.3 Pengujian Sensor Ultrasonik HC-SR04

Pada bab ini akan diuji hasil pengukuran sensor ultrasonik. Pengujian ini bertujuan untuk mengetahui perbandingan hasil pengukuran ultrasonik dengan hasil pengukuran menggunakan meteran. Pada pengujian ini sensor ultrasonik akan di komunikasikan ke PC untuk dapat melihat hasil pengukuran. Dalam hal ini akan memanfaatkan *Software* hterm untuk menampilkan hasil pengukuran.



Gambar 4.5 Mekanisme pengujian sensor ultrasonic HC-SR04



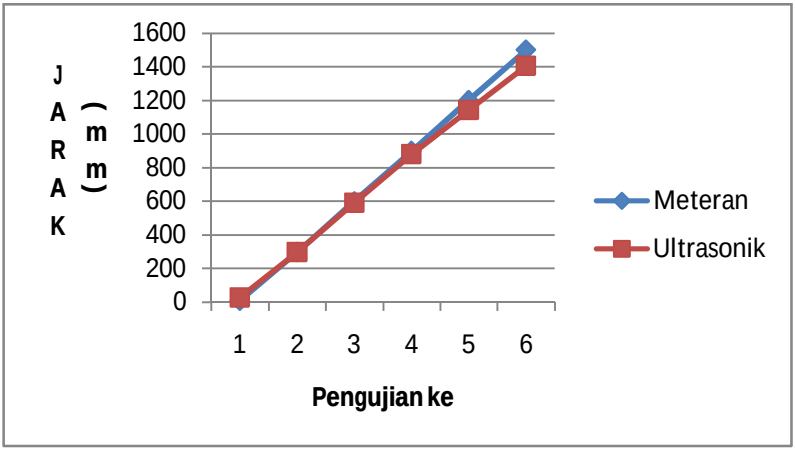
Gambar 4.6 Pengujian pembacaan sensor ultrasonik

Dari hasil pengujian didapatkan nilai hasil pengukuran yang dilakukan pada sensor ultrasonik 1 (depan), pada pengujian didapatkan data sebagai berikut :

Tabel 4.3 hasil pengujian pengukuran sensor ultrasonik

Pengujian	**Data pengukuran real**	**Data pengukuran ultrasonik**	**Error**
1	10 mm	30 mm	20 mm
2	300 mm	301 mm	1 mm
3	600 mm	592 mm	8 mm
4	900 mm	881 mm	19 mm
5	1200 mm	1144 mm	56 mm
6	1500 mm	1405 mm	95 mm
Rata-rata error			33,1667 mm

Dari hasil pengujian pada table 4.3 diatas didapatkan hasil pengukuran sensor ultrasonik yang cukup bagus, karena error yang dihasilkan tidak terlalu besar dengan rata-rata error sebesar 33,1667 mm. Pada pengukuran kurang dari 10 mm,sensor akan tetap membaca pada kisaran 30 mm.



Gambar 4.7 Grafik Data Pengujian Perbandingan Sensor Jarak Dengan Alat Ukur (meteran)

4.4 Pengujian Navigasi *Mobile* Robot

Pengujian navigasi dilakukan untuk melihat bagaimana hasil navigasi yang dilakukan oleh *mobile* robot dengan menggunakan sensor Android.

Pengujian ini nantinya akan dibagi ke dua bagian utama. Pada bagian pertama akan dilakukan pengujian navigasi *mobile* robot tanpa rintangan sedangkan bagian kedua akan dilakukan pengujian navigasi *mobile* robot dengan rintangan. Hal penting yang harus diperhatikan disini adalah lokasi atau area yang digunakan untuk melakukan pengujian. Dalam hal ini pengujian dilakukan pada area luas dan terbuka (lapangan), dan rintangan yang digunakan adalah rintangan buatan / portable.

Pengujian dilakukan dengan memberikan target posisi pada *mobile* robot dan melihat apakah *mobile* robot dapat berjalan menuju posisi yang diinputkan. Posisi target yang diinputkan dari seluruh pengujian adalah sama. Pada pengujian ini diamati pula akurasi dari pembacaan GPS untuk membandingkan responnya terhadap hasil navigasi. Pengujian ini juga akan membandingkan respon robot untuk radius target yang berbeda.

Pada pengujian ini dilakukan pada lapangan futsal teknik elektro, ITS Surabaya.

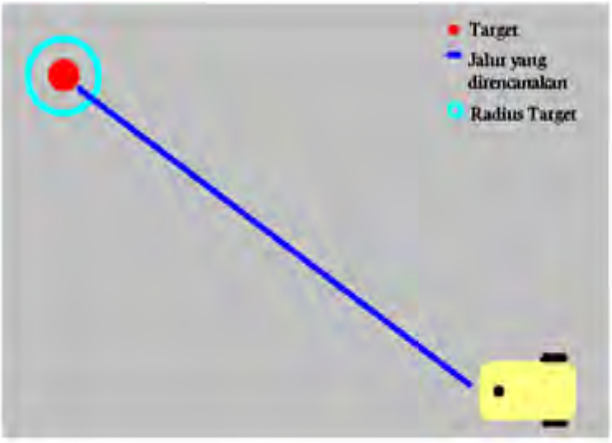
Gambar 4.8 Denah lokasi pengujian

4.4.1 Pengujian tanpa rintangan

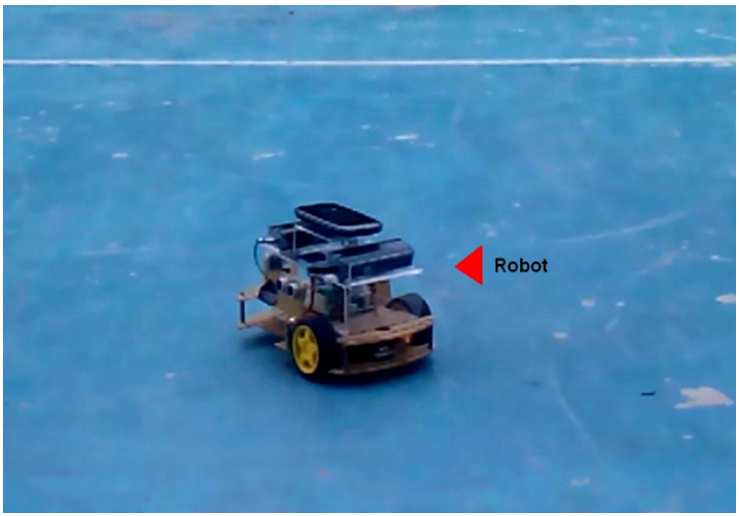
Pada pengujian yang pertama akan dilakukan pengujian navigasi tanpa ada rintangan yang menghalangi laju *mobile* robot. Navigasi dianggap sukses apabila jalur yang diambil oleh *mobile* robot telah mengarah menuju target dan mampu berhenti pada target.

Pada pengujian tanpa rintangan ini akan dibagi menjadi dua bagian pengujian, yaitu yang pertama adalah pengujian *mobile* robot tanpa rintangan dengan radius target 5 meter, kemudian yang kedua adalah pengujian *mobile* robot tanpa rintangan dengan radius 8 meter.

Pengujian dilakukan dengan memasukkan titik koordinat target kemudian menjalankan *mobile* robot dan mencatat hasil respon dari *mobile* robot.



Gambar 4.9 Ilustrasi pengujian *mobile* robot tanpa rintangan



Gambar 4.10 Pengujian di lapangan *mobile* robot tanpa rintangan

Pada pengujian tanpa rintangan yang pertama diberikan radius target 5 meter. Navigasi dianggap sukses apabila jalur yang diambil oleh

mobile robot telah mengarah menuju target dan berhenti pada radius 5 meter dari posisi target.

Tabel 4. 4 Tabel pengujian navigasi *mobile* robot tanpa rintangan dengan radius target 5 meter

Pengujian	Jarak robot ke target (meter)	Akurasi saat start (meter)	Robot mengarah ke target?	Robot berhenti di target?
1	25	5	Ya	Ya
2	25	7	Ya	Ya
3	25	5	Ya	Ya
4	25	6	Ya	Ya
5	25	5	Ya	Ya
6	25	8	Tidak	Tidak
7	25	5	Ya	Ya
8	25	17	Tidak	Tidak
9	25	10	Ya	Tidak
10	25	7	Ya	Tidak

Dari hasil pengujian tabel 4.4 diatas, dapat dilihat bahwa dari pengujian yang dilakukan sebanyak 10 kali didapati hasil 8 kali robot mengarah ke target dan 2 kali robot tidak mengarah ketarget. Dan dari pengujian sebanyak 10 kali tersebut didapatkan hasil bahwa pada radius target 5 meter, robot dapat berhenti pada target sebanyak 6 kali dan 4 kali sisanya tidak dapat mengarah pada target atau dalam persentase keberhasilan sebesar 60%.

Pada pengujian kedua ini navigasi masih tetap dilakukan tanpa adanya rintangan yang menghalangi laju dari *mobile* robot. Namun pada pengujian kedua ini radius target diperbesar dari 5 meter menjadi 8 meter. Navigasi dianggap sukses apabila jalur yang diambil oleh *mobile* robot telah mengarah menuju target dan berhenti pada radius 8 meter dari posisi target.

Tabel 4. 5 Tabel pengujian navigasi *mobile* robot tanpa rintangan dengan radius target 8 meter

Pengujian	Jarak robot ke target (meter)	Akurasi saat start (meter)	Robot mengarah ke target?	Robot berhenti di target?
1	25	5	Ya	Ya
2	25	5	Ya	Ya
3	25	6	Ya	Ya
4	25	5	Ya	Ya
5	25	9	Ya	Ya
6	25	13	Tidak	Tidak
7	25	8	Ya	Ya
8	25	5	Ya	Ya
9	25	10	Tidak	Tidak
10	25	8	Ya	Tidak

Dari hasil pengujian diatas didapati hasil 8 dari 10 kali pengujian, robot mengarah ke target. Dan dari pengujian sebanyak 10 kali tersebut didapatkan hasil bahwa pada radius target 8 meter, robot mampu berhenti sebanyak 7 kali dan 3 sisanya robot tidak dapat berhenti pada target atau dalam persentase keberhasilan sebesar 70%. Dari data diatas dapat disimpulkan bahwa dengan menambah nilai radius target dari 5 meter ke 8 meter memberikan hasil navigasi yg lebih baik. Namun karena pengubahan range ini, jarak antara posisi berhenti robot dan target posisi menjadi lebih jauh.

Pada pengujian ini diamati juga hasil jalur yang direkam oleh *mobile* robot ketika start hingga menuju target. Plot jalur hasil navigasi ini untuk melihat bagaimana perbandingan jalur yang dihasilkan oleh *mobile* robot saat melakukan navigasi dengan jalur yang telah direncanakan. Plot jalur ini memanfaatkan perubahan data lintang dan bujur posisi Android dari saat start hingga mencapai target. Dari data GPS tersebut, dapat diambil suatu jalur yang dilalui robot mulai dari start hingga mencapai target.

Pada pengujian tanpa rintangan dapat di plot jalur yang dilalui oleh robot sebagai berikut :

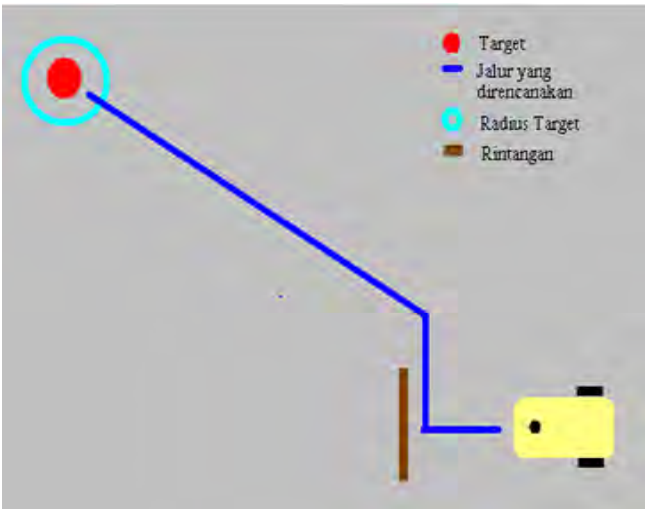
Gambar 4.11 Hasil plot jalur robot pada navigasi tanpa rintangan.

Dari gambar 4.11 diatas dapat dilihat bahwa garis biru merupakan jalur yang direncanakan, namun jalur yang dilalui oleh mobile robot sedikit besimpangan dari jalur yang direncanakan seperti yang ditunjukkan oleh garis hitam. Meski bersimpangan namun nilai dari simpangan tersebut cenderung kecil (< 3 meter). Terjadinya simpangan tersebut dikarenakan pembacaan GPS yang tidak maksimal sehingga mengakibatkan arah navigasi / heading menuju target menjadi tidak akurat sehingga jalur navigasi mobile robot menjadi cenderung menyimpang.

4.4.2 Pengujian Navigasi Dengan Rintangan

Pada pengujian yang kedua akan dilakukan pengujian navigasi dengan rintangan yang menghalangi laju *mobile* robot. Navigasi dianggap sukses apabila jalur yang diambil oleh *mobile* robot telah mengarah menuju target dan mampu berhenti pada target.

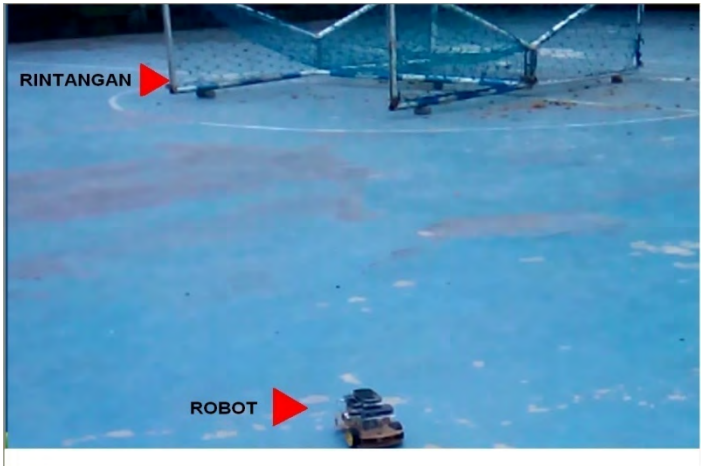
Pada pengujian dengan rintangan ini juga akan dibagi menjadi dua bagian pengujian, yaitu yang pertama adalah pengujian *mobile* robot dengan rintangan dengan radius target 5 meter, kemudian yang kedua adalah pengujian *mobile* robot dengan rintangan dengan radius target 8 meter. Pengujian dilakukan dengan memasukkan titik koordinat target kemudian menjalankan *mobile* robot dan mencatat hasil respon dari *mobile* robot.



Gambar 4.12 Ilustrasi pengujian *mobile* robot dengan rintangan

Pada pengujian pertama dengan rintangan ini *mobile* robot akan dijalankan dengan fungsi mencapai target tujuan dan mampu menghidari rintangan. Navigasi dianggap sukses apabila robot mampu menghindari rintangan, Serta robot juga mampu menuju target dan berhenti pada

radius target 5 meter dari posisi target. Pada pengujian ini rintangan pada sisi kanan, kiri dan depan menggunakan ritangan buatan.



Gambar 4.13 Pengujian di lapangan navigasi dengan rintangan

Tabel 4. 6 Tabel pengujian navigasi *mobile* robot dengan rintangan dengan radius target 5 meter.

Uji ke	Jarak (meter)	Akurasi saat start (meter)	Ada Rintangan, Respon robot?	Robot mengarah ke target?	Robot berhenti di target?
1	25	8	Menghindar	Ya	Tidak
2	25	5	Menghindar	Ya	Ya
3	25	7	Menabrak lalu menghindar	Ya	Ya
4	25	9	Menghindar	Tidak	Tidak
5	25	5	Menghindar	Ya	Ya
6	25	18	Menghindar	Tidak	Tidak

7	25	5	Tidak dapat menghindar	Ya	Ya
8	25	17	Tidak dapat menghindar	Tidak	Tidak
9	25	10	Menghindar	Tidak	Tidak
10	25	7	Menabrak lalu menghindar	Ya	Ya

Dari pengujian yang telah dilakukan sebanyak 10 kali, di dapatkan hasil bahwa 6 kali pengujian robot dapat mengarah ke target dan 5 diantaranya robot dapat berhenti di radius target 5 meter. Hasil pengujian ini lebih rendah dibandingkan pada pengujian navigasi tanpa rintangan dengan radius target yang sama. Hal yang paling berpengaruh dalam hasil pengujian ini adalah akurasi GPS dan bagaimana respon robot ketika harus menghindari rintangan. Beberapa kali robot menabrak bahkan tidak mampu menghindari rintangan. Ketika menghindari rintangan, terkadang robot terlalu melebar jauh dari jalur yang direncanakan sehingga membuat robot sulit mengarah ke target. Panjang dari dari rintangan juga mempengaruhi bagaimana robot dapat menghindar.

Pada pengujian selanjutnya navigasi masih akan tetap di uji dengan rintangan yang menghalangi laju dari *mobile* robot. Namun pada pengujian ini panjang rintangan akan diturunkan dari 100 cm ke 50 cm. hal ini tentu diharapkan mampu mempermudah mobile robot untuk menghindari rintangan sehingga dapat mengarah ke target lebih baik. Selain itu, radius target juga akan diperbesar dari 5 meter menjadi 8 meter. Navigasi dianggap sukses apabila jalur yang diambil oleh *mobile* robot telah mengarah menuju target dan berhenti pada radius 8 meter dari posisi target.

Pada pengujian ini *mobile* robot masih akan dijalankan dengan fungsi mencapai target tujuan dan mampu menghidari rintangan. Navigasi dianggap sukses apabila robot mampu menghindari rintangan , Serta robot juga mampu menuju target dan berhenti pada radius 8 meter dari posisi target.

Tabel 4.7 Tabel pengujian navigasi *mobile* robot dengan radius target 8 meter

Uji ke	Jarak (meter)	Akurasi saat start (meter)	Ada Rintangan, Respon robot?	Robot mengarah ke target?	Robot berhenti di target?
1	25	5	Menabrak lalu Menghindar	Ya	Ya
2	25	5	Menghindar	Ya	Ya
3	25	4	Menabrak lalu Menghindar	Ya	Ya
4	25	8	Menghindar	Ya	Tidak
5	25	6	Menghindar	Ya	Ya
6	25	9	Menabrak lalu Menghindar	Tidak	Tidak
7	25	5	Menabrak lalu menghindar	Ya	Ya
8	25	12	Menghindar	Tidak	Tidak
9	25	10	Menabrak lalu Menghindar	Tidak	Tidak
10	25	8	Menghindar	Ya	Ya

Dari hasil pengujian diatas, dapat dilihat bahwa dengan mengurangi panjang rintangan, robot mampu melewati rintangan lebih baik dibandingkan dengan pengujian sebelumnya. Dari pengujian yang dilakukan sebanyak 10 kali didapati hasil 7 kali robot mengarah ke target dan 3 kali robot tidak mengarah ketarget atau dalam persentase keberhasilan sebesar 70%. Sedangkan dari pengujian sebanyak 10 kali tersebut juga didapatkan hasil bahwa pada radius target 8 meter, robot

dapat berhenti pada target sebanyak 6 kali dan 4 kali sisanya tidak dapat mengarah pada target. Robot juga mampu menghindari rintangan dengan baik meski beberapa kali harus menabrak dahulu sebelum dapat menghindar.

Dari hasil seluruh pengujian didapati bahwa akurasi pembacaan GPS tetap menentukan hasil navigasi yang dilakukan. Semakin akurat pembacaan posisi dari GPS, maka arah *bearing* yang dikalkulasi oleh android semakin *valid*, sebaliknya apabila pembacaan GPS tidak akurat posisi *bearing* yang dikalkulasi akan menjadi tidak tepat karena range *bearing* menjadi besar. Robot juga mampu menghindari rintangan dengan cukup baik meski beberapa kali menabrak rintangan sebelum akhirnya dapat menghindar. Pada pengujian ini robot sering kali menabrak bagian sudut dari rintangan, hal ini dikarenakan adanya keterbatasan pembacaan sudut dari ultrasonic yang tidak lebih dari 30^0. Dengan keterbatasan ini robot tidak mampu mendeteksi adanya rintangan di sudut robot tersebut karena ultrasonic hanya dipasang pada depan bagian tengah dan samping kanan, kiri.

Selanjutnya adalah mengamati hasil plot jalur hasil navigasi mobile robot dengan adanya rintangan yang menghalangi. Hampir sama seperti pada navigasi tanpa rintangan pada pengujian sebelumnya, pada navigasi dengan rintangan ini akan di plot jalur hasil navigasi untuk melihat bagaimana perbandingan jalur yang dihasilkan oleh *mobile* robot saat melakukan navigasi dengan jalur yang telah direncanakan. Namun yang membedakan adalah pada pengujian ini terdapat suatu rintangan.
Plot jalur ini sama dengan plot jalur pada pengujian sebelumnya, yaitu dengan memanfaatkan perubahan data lintang dan bujur posisi Android dari saat start hingga mencapai target. Dari data GPS tersebut, dapat diambil suatu jalur yang dilalui robot mulai dari start hingga mencapai target.

Gambar 4.14 Hasil Plot jalur robot pada navigasi dengan rintangan.

Dari gambar 4.14 diatas dapat dilihat hasil plot dari mobile robot saat start hingga menuju ketarget. Sama seperti pengujian sebelumnya, terjadi simpangan antara jalur yang direncanakan dengan jalur yang direkam oleh mobile robot. Hal ini dipengaruhi oleh pembacaan GPS pada *device* Android yang tidak maksimal sehingga mengakibatkan arah navigasi / heading menuju target menjadi tidak akurat.Namun ketika robot mendeteksi rintangan, plot jalur yang direkam justru hampir sesuai dengan jalur yang direncanakan, hal ini dikarenakan ketika mobile robot mendeteksi rintangan maka ultrasonik yang aktif akan mengirimkan data ke mikrokontroler agar robot menjaga jarak dengan rintangan pada jarak kurang dari 30 cm.

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil perancangan dan pengujian alat pada tugas akhir ini didapatkan beberapa kesimpulan seperti berikut ini :

1. Sistem komunikasi dengan memanfaatkan *Bluetooth* dapat menghasilkan komunikasi data yang *reliable* dengan tingkat ketepatan hingga 100 % pada transfer rate 9600 bps dan jarak < 10 meter.
2. Navigasi tanpa rintangan yang dilakukan memiliki tingkat keberhasilan sebesar 60% pada radius target 5meter dan 70% pada radius target 8meter.
3. Navigasi dengan rintangan yang dilakukan memiliki tingkat keberhasilan sebesar 50% pada radius target 5meter dan 60% pada radius target 8meter.
4. Tingkat keberhasilan navigasi dengan rintangan sangat dipengaruhi oleh panjang rintangan yang menghalangi, semakin pendek rintangan (<50 cm) yang terdeteksi maka akan semakin besar tingkat keberhasilan robot mampu menghindari rintangan.

5.2 Saran

Saran – saran yang dapat diberikan untuk pengembangan alat sistem navigasi dan penghindar rintangan ini adalah sebagai sebagai berikut:

1. Banyak fitur pada *Smartphone* Android yang masih bisa ditambahkan guna untuk pengembangan sistem navigasi, semisal pemanfaatan kamera,WiFi, USB, Accelerometer , Gyroscope ,dll
2. Pengontrolan dasar arah kemudi masih kurang optimal. Perlu adanya penambahan algoritma dalam pengontrolan arah kemudi semisal PID, Fuzzy dan neural network agar hasil pengontrolan lebih optimal.

 # Halaman ini sengaja dikosongkan #

LAMPIRAN

PROGRAM ANDROID

```
package com.Example.coba4;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.reflect.Method;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.UUID;
import java.util.Timer;
import java.util.TimerTask;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.media.AudioManager;
import android.media.ToneGenerator;
import android.os.Build;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.os.Environment;
import android.os.Handler;
import android.telephony.SmsMessage;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

```
import android.widget.Toast;


public class MainActivity extends Activity {
        private BluetoothAdapter btAdapter = null;
        private BluetoothSocket btSocket = null;
        private StringBuilder sb = new StringBuilder();

        static public final char cr = (char) 13;
        static public final char lf = (char) 10;
        protected static final String TAG = null;
        TextView Tulis1, Tulis2, Lintang, Bujur, Akurasi, GPSstat;
        TextView SavLintang, SavBujur, arah, radius,setir;
        EditText Edit1, Edit2;
        Button button_lintang, button_bujur, button1, button2,
button3;
        private SensorManager sensorManager;
        private float x;
        float hadap = 0;
        private double dist=0, Lintang1, Bujur1, Bujur2=0,
Lintang2=0;
        int navigasi = 0;
        int PWM =0;
        float accuracy;
        final String SMS_RECEIVED =
"android.provider.Telephony.SMS_RECEIVED";
        public String fileName;
        public String heading;
        public String speed;
        public String nilai_pwm;

          Handler h;

          final int RECIEVE_MESSAGE = 1;

          private ConnectedThread mConnectedThread;

          private static final UUID MY_UUID =
UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

          private static String address = "98:D3:31:60:0B:6D";


        /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
                        h = new Handler() {
        public void handleMessage(android.os.Message msg) {
switch (msg.what) {
case RECIEVE_MESSAGE:

// if receive massage
byte[] readBuf = (byte[]) msg.obj;
String strIncom = new String(readBuf, 0, msg.arg1);

// create string from bytes array
sb.append(strIncom);

// append string
int endOfLineIndex = sb.indexOf("\r\n");

// determine the end-of-line
if (endOfLineIndex > 0) {

// if end-of-line,
String sbprint = sb.substring(0, endOfLineIndex);

mConnectedThread.write("$"+heading+","+speed+","+"#"+cr+lf);
sb.delete(0, sb.length());
                                        // and clear
}
                                        //Log.d(TAG,
"...String:"+ sb.toString() +  "Byte:" + msg.arg1 + "...");
break;
}
    };
        };

        Tulis1=(TextView)findViewById(R.id.text1);
        Lintang=(TextView)findViewById(R.id.lintang);
        Bujur=(TextView)findViewById(R.id.bujur);
        Akurasi=(TextView)findViewById(R.id.akurasi);
        GPSstat=(TextView)findViewById(R.id.GPSStatus);
        SavLintang=(TextView)findViewById(R.id.savLintang);
        SavBujur=(TextView)findViewById(R.id.savBujur);
        arah=(TextView)findViewById(R.id.arah);
        radius=(TextView)findViewById(R.id.jarak);
        setir=(TextView)findViewById(R.id.setir);
        Edit1=(EditText)findViewById(R.id.editText1);
        Edit2=(EditText)findViewById(R.id.editText2);
        button_lintang=(Button)findViewById(R.id.button_lintang);
        button_bujur=(Button)findViewById(R.id.button_bujur);
        button1=(Button)findViewById(R.id.button1);
```

```
        button2=(Button)findViewById(R.id.button2);


 //registrasi sensor kompas

sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
sensorManager.registerListener(myListener1,sensorManager.getDefaul
tSensor(Sensor.TYPE_ORIENTATION),SensorManager.SENSOR_DELAY_FASTES
T);

        //registrasi tombol
        button1.setOnClickListener(myListener2);
        button2.setOnClickListener(myListener2);
        button_lintang.setOnClickListener(myListener2);
        button_bujur.setOnClickListener(myListener2);

//registrasi GPS
LocationManager mlocManager
=(LocationManager)getSystemService(Context.LOCATION_SERVICE);
mlocManager.requestLocationUpdates( LocationManager.GPS_PROVIDER,
0, 0, myListener3);

        //registrasi bluetooth
        btAdapter = BluetoothAdapter.getDefaultAdapter();
        checkBTState();

    }

private BluetoothSocket createBluetoothSocket(BluetoothDevice
device) throws IOException {
        if(Build.VERSION.SDK_INT >= 10){
            try {
                final Method  m =
device.getClass().getMethod("createInsecureRfcommSocketToServiceRe
cord", new Class[] { UUID.class });
                return (BluetoothSocket) m.invoke(device,
MY_UUID);
            } catch (Exception e) {
                Log.e(TAG, "Could not create Insecure RFComm
Connection",e);
            }
        }
        return  device.createRfcommSocketToServiceRecord(MY_UUID);
    }

    @Override
    protected void onResume() {
            super.onResume();
```

```
sensorManager.registerListener(myListener1,
sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
SensorManager.SENSOR_DELAY_FASTEST);
Log.d(TAG, "...onResume - try connect...");

BluetoothDevice device = btAdapter.getRemoteDevice(address);

        try {
        btSocket = createBluetoothSocket(device);
                } catch (IOException e) {
                        errorExit("Fatal Error", "In onResume()
and socket create failed: " + e.getMessage() + ".");
                }

        btAdapter.cancelDiscovery();

        Log.d(TAG, "...Connecting...");
            try {
              btSocket.connect();
              Log.d(TAG, "....Connection ok...");
            } catch (IOException e) {
              try {
                btSocket.close();
              } catch (IOException e2) {
                errorExit("Fatal Error", "In onResume() and unable
to close socket during connection failure" + e2.getMessage() +
".");
              }
            }

 // Create a data stream so we can talk to server.
            Log.d(TAG, "...Create Socket...");

            mConnectedThread = new ConnectedThread(btSocket);
            mConnectedThread.start();

            }

    @Override
        protected void onPause() {
            super.onPause();
            sensorManager.unregisterListener(myListener1);

            Log.d(TAG, "...In onPause()...");

            try     {
              btSocket.close();
```

```
            } catch (IOException e2) {
              errorExit("Fatal Error", "In onPause() and failed to
close socket." + e2.getMessage() + ".");
            }

          }

        //Syntax untuk GPS
        LocationListener myListener3 = new LocationListener() {

                public void onLocationChanged(Location loc) {
                        Lintang1 = loc.getLatitude();
                        Bujur1   = loc.getLongitude();
                        accuracy = loc.getAccuracy();

                        //Current Position
                        Location A = new Location("titik A");
                                A.setLatitude(Lintang1);
                                A.setLongitude(Bujur1);

                        //Target Position
                        Location B = new Location("titik B");

                                B.setLatitude(Lintang2);
                                B.setLongitude(Bujur2);

                        dist = A.distanceTo(B);
                        hadap = A.bearingTo(B);
                        if (hadap<0) hadap=360+hadap;

                        Lintang.setText("Lintang : "+Lintang1);
                        Bujur.setText("Bujur : "+Bujur1);
                        Akurasi.setText("Akurasi : "+ accuracy);

                        arah.setText("Arah : "+hadap);
                        radius.setText("jarak : "+dist);


                }

                public void onProviderDisabled(String arg0) {
                        GPSstat.setText("GPS mati");

                }

                public void onProviderEnabled(String arg0) {
                        GPSstat.setText("GPS hidup");
```

```
                }

                public void onStatusChanged(String arg0, int arg1,
Bundle arg2) {

                }
        };

        //Syntax untuk Tombol
        OnClickListener myListener2 = new OnClickListener() {

        public void onClick(View v) {

                if(v==button2) {
                Lintang2 = Lintang1;
                Bujur2 = Bujur1;
                SavLintang.setText("Tujuan (Lintang) :
"+Lintang2);
                SavBujur.setText("Tujuan (Bujur)  : "+Bujur2);
                        }

                else if(v==button1){


                if (button1.getText().equals("MULAI NAVIGASI")) {
                button1.setText("MATIKAN NAVIGASI");
                navigasi=1;

                SimpleDateFormat formatter = new
                SimpleDateFormat("yyyy_MM_dd_HH_mm_ss");
                Date now = new Date();
                fileName = "GPSLOG "+formatter.format(now) +
                ".txt";


try {
File root = Environment.getExternalStorageDirectory();
if (root.canWrite()){
File gpxfile = new File(root,"/GPSLOG/"+fileName);
FileWriter out = new FileWriter(gpxfile,true);
out.write("Target :
"+Double.toString(Lintang2)+","+Double.toString(Bujur2)+cr+lf);

out.write(Double.toString(Lintang1)+","+Double.toString(Bujur1)+cr
+lf);
out.close();
 }
} catch (IOException e) {
```

```
         Log.e(TAG, "Could not write file " + e.getMessage());
                }

        Counter1.start();

                }
        else if (button1.getText().equals("MATIKAN NAVIGASI")) {
                button1.setText("MULAI NAVIGASI");
                navigasi=0;
                                        }
                                }
        else if(v==button_lintang){

        Lintang2=Double.parseDouble(Edit1.getText().toString());
        SavLintang.setText("Tujuan (Lintang) : "+Lintang2);
                        }
        else if(v==button_bujur){
        Bujur2= Double.parseDouble(Edit2.getText().toString());
                                SavBujur.setText("Tujuan (Bujur)
: "+Bujur2);
                        }
                }
        };

        //Syntax Untuk data logger
        CountDownTimer Counter1 = new CountDownTimer( 1000 , 1000)
{
        public void onTick(long millisUntilFinished) {

                }

        public void onFinish() {
                if (button1.getText().equals("MATIKAN NAVIGASI")){

                try {
File root = Environment.getExternalStorageDirectory();
if (root.canWrite()){
File gpxfile = new File(root, "/GPSLOG/"+fileName);
FileWriter out = new FileWriter(gpxfile,true);

out.write(Double.toString(Lintang1)+","+Double.toString(Bujur1)+cr
+lf);
 out.close();
        }
                } catch (IOException e) {
         Log.e(TAG, "Could not write file " + e.getMessage());
                }
        Counter1.start();
```

```
                }

        else if (button1.getText().equals("MULAI NAVIGASI")){

        Counter1.cancel();
                }
                }
                };


//Syntax Untuk Sensor kompas
SensorEventListener myListener1 = new SensorEventListener() {
public void onAccuracyChanged(Sensor sensor, int accuracy) {

                }

        public void onSensorChanged(SensorEvent event) {
                //int heading = 0;
                int temp_PWM = 80; //STOP
                speed="nol";

                double SmoothFactorCompass = 0.5;
                double SmoothThresholdCompass = 30.0;
                double oldCompass = 0.0;
                double newCompass;


        if(event.sensor.getType()==Sensor.TYPE_ORIENTATION){

        //if(event.sensor.getType()==Sensor.TYPE_ORIENTATION){
                                x=event.values[0];
                                newCompass = x;

        if (Math.abs(newCompass - oldCompass) < 180) {

if (Math.abs(newCompass - oldCompass) > SmoothThresholdCompass) {
        oldCompass = newCompass;
                }
else {
oldCompass = oldCompass + SmoothFactorCompass * (newCompass -
oldCompass);
                }
                }
else {
if (360.0 - Math.abs(newCompass - oldCompass) >
SmoothThresholdCompass) {
 oldCompass = newCompass;
                }
```

```
else {
if (oldCompass > newCompass) {
oldCompass = (oldCompass + SmoothFactorCompass * ((360 +
newCompass - oldCompass) % 360) + 360) % 360;
                }
else {
oldCompass = (oldCompass - SmoothFactorCompass * ((360 -
newCompass + oldCompass) % 360) + 360) % 360;
}
        }
                }
double kompas=oldCompass;
        Tulis1.setText("Kompas :"+x);
        int selisih = (int) ((hadap)-(kompas));

        if (selisih<10 && selisih>-10) {
        setir.setText("lurus");

//heading = 90;
heading="lurus";
temp_PWM=86; //KECEPATAN TINGGI
nilai_pwm="cepat";
        }
else {
temp_PWM = 85; //KECEPATAN RENDAH
nilai_pwm="pelan";

if (hadap<kompas) {
hadap=hadap+360;
if ((hadap-kompas)<180) {
setir.setText("kanan");

//heading = 70;
heading="kanan";
                                                        }
else {
setir.setText("kiri");

//heading = 105;
heading="kiri";
                }
                        }
else {
kompas = kompas+360;
if ((hadap-kompas)>-180) {
setir.setText("kiri");

//heading = 105;
```

```
heading="kiri";
}
else {
setir.setText("kanan");

//heading = 70;
heading="kanan";
                }
                        }
                                }

if (navigasi==0) {

PWM=80;  //PWM FOR STOPPING
//setir.setText("STOP" + PWM);
speed="nol";
                }
else if (navigasi == 1 ) {
if (dist > 10){ // jika jarak belum tercapai, navigasi berlanjut
PWM=temp_PWM;
speed=nilai_pwm;
        }
        else {
        PWM=80;
        speed="nol";

        button1.setText("MULAI NAVIGASI");
        navigasi=0;

//String peringatan = ;

Toast.makeText(getBaseContext(), "!! JARAK <=1m !!",
Toast.LENGTH_LONG).show();
final ToneGenerator tg = new
ToneGenerator(AudioManager.STREAM_NOTIFICATION, 100);
                                }
                        }
                }
        }
};

private void checkBTState() {

            if(btAdapter==null) {
              errorExit("Fatal Error", "Bluetooth not support");
            } else {
              if (btAdapter.isEnabled()) {
                Log.d(TAG, "...Bluetooth ON...");
```

```
            } else {
//Prompt user to turn on Bluetooth
Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
              startActivityForResult(enableBtIntent, 1);
            }
          }
        }

        private void errorExit(String title, String message){
          Toast.makeText(getBaseContext(), title + " - " +
message, Toast.LENGTH_LONG).show();
          finish();
        }

        private class ConnectedThread extends Thread {
                  private final InputStream mmInStream;
                  private final OutputStream mmOutStream;

                  public ConnectedThread(BluetoothSocket socket)
{
                      InputStream tmpIn = null;
                      OutputStream tmpOut = null;

                      try {
                          tmpIn = socket.getInputStream();
                          tmpOut = socket.getOutputStream();
                      } catch (IOException e) { }

                      mmInStream = tmpIn;
                      mmOutStream = tmpOut;
                  }

public void run() {
byte[] buffer = new byte[256];
int bytes;
while (true) {
try {
bytes = mmInStream.read(buffer);
h.obtainMessage(RECIEVE_MESSAGE, bytes, -1,
buffer).sendToTarget();                                    }
catch (IOException e) {
break;
                    }
        }
}
```

PROGRAM MIKROKONTROLER

```
$regfile = "m328pdef.dat"
$crystal = 11059200
$baud = 9600

Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down ,
Compare B Pwm = Clear Down , Prescale = 64

Pwm1a = 0 : Pwm1b = 0

Config Portc.5 = Output
Config Pinc.4 = Input
Config Portc.3 = Output
Config Portc.2 = Output

Config Portd.4 = Output
Config Portd.7 = Output
Config Portb.1 = Output
Config Portb.2 = Output
Config Pinb.3 = Input

Config Portc.0 = Output
Config Portd.6 = Output
Config Portd.5 = Output

Led_kanan Alias Portd.5
Led_lurus Alias Portd.6
Led_kiri Alias Portc.0

Reset Led_kanan : Reset Led_lurus : Reset Led_kiri

Kunci Alias Pinb.3
M3_putar Alias Portd.4
M4_putar Alias Portd.7

Trig Alias Portc.5
Echoo Alias Pinc.4
Sel_a Alias Portc.3
Sel_b Alias Portc.2

'---sensor---
Dim Dta As Word
Dim Sonic1 As Word
Dim Sonic2 As Word
```

```
Dim Sonic3 As Word
Dim Sonic4 As Word
Dim Time_out As Word
    Dim Data_lawas As Word
    Dim Distance_set As Word
    Dim Distance As Word
    Dim A As Word

    Dim Rx_byte As Byte
    Dim Njupuk As String * 10
    Dim Rx_char As String * 1
    Dim Rx_flag As Bit
    Dim Blue_t(2) As String * 10
    Dim I As Byte
    Dim B1 As Byte

    On Urxc Rec_isr
    Goto Awal

    '***** INISIALISASI SISTEM *****


    'INTERUPSI TERIMA DATA SERIAL
    Rec_isr:
       Rx_byte = Udr
       Rx_char = Chr(rx_byte)
       Set Rx_flag
       Return

    Awal:
    Enable Urxc
    Enable Interrupts
    For B1 = 1 To 3
       Reset Led_kanan : Reset Led_lurus : Reset Led_kiri
       Waitms 100
       Set Led_kanan : Set Led_lurus : Set Led_kiri
       Waitms 350
    Next
    Reset Trig

    Mula:

    Pwm1a = 0 : Pwm1b = 0
    Do
       Gosub Baca_bluetooth
       If Blue_t(1) = "lurus" And Blue_t(2) = "nol" Then
          Set Led_kanan : Set Led_lurus : Set Led_kiri
          Waitms 2
```

```
   Gosub Motor_off
Elseif Blue_t(1) = "kanan" And Blue_t(2) = "nol" Then
   Set Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
   Gosub Motor_off
Elseif Blue_t(1) = "kiri" And Blue_t(2) = "nol" Then
   Set Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
   Gosub Motor_off
Elseif Blue_t(1) = "lurus" And Blue_t(2) = "pelan" Then
   Set Led_kanan : Reset Led_lurus : Set Led_kiri
   Waitms 2
   Gosub Lurus_pelan
   Set Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
Elseif Blue_t(1) = "kanan" And Blue_t(2) = "pelan" Then
   Reset Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
   Gosub Kanan_pelan
   Set Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
Elseif Blue_t(1) = "kiri" And Blue_t(2) = "pelan" Then
   Set Led_kanan : Set Led_lurus : Reset Led_kiri
   Waitms 2
   Gosub Kiri_pelan
   Set Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
Elseif Blue_t(1) = "lurus" And Blue_t(2) = "cepat" Then
   Set Led_kanan : Reset Led_lurus : Set Led_kiri
   Waitms 2
   Gosub Lurus_cepat
   Set Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
Elseif Blue_t(1) = "kanan" And Blue_t(2) = "cepat" Then
   Reset Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
   Gosub Kanan_pelan
   Set Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
Elseif Blue_t(1) = "kiri" And Blue_t(2) = "cepat" Then
   Set Led_kanan : Set Led_lurus : Reset Led_kiri
   Waitms 2
   Gosub Kiri_pelan
   Set Led_kanan : Set Led_lurus : Set Led_kiri
   Waitms 2
End If
Set Led_kanan : Set Led_lurus : Set Led_kiri
Waitms 5
```

```
Loop


Baca_bluetooth:
Njupuk = ""
Blue_t(1) = ""
Blue_t(2) = ""
Time_out = 0
Reset Rx_flag
Print "Data"
Do

   'Incr Time_out
   If Rx_flag = 1 Then
      Exit Do
   Else
      Incr Time_out
   End If
   Waitms 1
   If Time_out > 150 Then Goto Baca_bluetooth
Loop
Time_out = 0
Do
   If Rx_flag = 1 Then
      Reset Rx_flag
      If Rx_byte = 36 Then
         Njupuk = "" : I = 0
         Do
            If Rx_flag = 1 Then
               Reset Rx_flag
               Select Case Rx_byte
                  Case 44
                     Incr I
                     Blue_t(i) = Njupuk
                     Njupuk = ""
                  Case 35
                     Exit Do
                  Case Else
                     Njupuk = Njupuk + Rx_char
               End Select
            End If
         Loop
         Goto Akhir
      End If
   Else
      Incr Time_out
   End If
   If Time_out > 150 Then Exit Do
```

```
Loop
Akhir:
Return

Baca_sonic1:
   Reset Sel_a : Reset Sel_b
   Waitms 40
   Gosub Baca_sensor
   Sonic1 = Dta
   Waitms 2
   Return
Baca_sonic2:
   Set Sel_a : Reset Sel_b
   Waitms 40
   Gosub Baca_sensor
   Sonic2 = Dta
   Waitms 2
   Return
Baca_sonic3:
   Reset Sel_a : Set Sel_b
   Waitms 40
   Gosub Baca_sensor
   Sonic3 = Dta
   Waitms 2
   Return
Baca_sonic4:
   Set Sel_a : Set Sel_b
   Waitms 40
   Gosub Baca_sensor
   Sonic4 = Dta
   Waitms 2
   Return

Baca_sensor:
   Set Trig
   Waitus 10
   Reset Trig
   Do
      Incr Time_out
      If Echoo = 1 Then Exit Do
   Loop Until Time_out = 10000
   Time_out = 0
   Dta = 0
   Config Timer1 = Timer , Prescale = 64
   Timer1 = 0
       Start Timer1
       Do
          Incr Time_out
```

```
      If Echoo = 0 Then
         Dta = Timer1
         Exit Do
      End If
   Loop Until Time_out = 10000
   Time_out = 0
   Stop Timer1
   Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down
, Compare B Pwm = Clear Down , Prescale = 64
   Return

'(
Kanan_pelan:
   Gosub Baca_sonic2
   If Sonic2 > 500 Then
      Gosub Motor_kanan
      Waitms 80
   Else
'      Gosub Motor_maju_pelan
      Gosub Motor_kiri
   End If
   Gosub Motor_off
   Return

Kiri_pelan:
   Gosub Baca_sonic4
   If Sonic4 > 500 Then
      Gosub Motor_kiri
      Waitms 80
   Else
'      Gosub Motor_maju_pelan
      Gosub Motor_kanan
   End If
   Gosub Motor_off
   Return


Lurus_cepat:
   Gosub Baca_sonic1
   If Sonic1 > 500 Then
      Gosub Motor_maju_cepat
      Waitms 80
   Else
'      Gosub Motor_maju_pelan
      Gosub Motor_mundur
End If
Gosub Motor_off
Return
```

```
Lurus_pelan:
   Gosub Baca_sonic1
   If Sonic1 > 500 Then
      Gosub Motor_maju_cepat
      Waitms 80
   Else
'      Gosub Motor_maju_pelan
      Gosub Motor_mundur
   End If
   Gosub Motor_off
   Return


')
Kanan_pelan:
   Gosub Motor_off
   Gosub Baca_sonic2
   If Sonic2 = 0 Or Sonic2 > 500 Then
      Gosub Motor_kanan
      Waitms 80
   'Elseif Sonic2 = 0 Then
   '   Gosub Motor_kanan
   '   Waitms 80
   Else
'      Gosub Motor_maju_pelan
      Gosub Cek_kanan
   End If
   Gosub Motor_off
   Return

Kiri_pelan:
   Gosub Motor_off
   Gosub Baca_sonic4
   If Sonic4 = 0 Or Sonic4 > 500 Then
      Gosub Motor_kiri
      Waitms 80
   Else
      Gosub Cek_kiri
   End If
   Gosub Motor_off
   Return


Cek_kiri:
   Gosub Motor_off
   Do
'      Baca_sonic1
```

```
      Gosub Baca_sonic4
      Gosub Motor_kanan
      If Sonic4 = 0 Or Sonic4 > 500 Then Exit Do
      Waitms 5
   Loop
   Gosub Motor_off
   Do
   '      Baca_sonic1
      Gosub Motor_maju_pelan
      Waitms 20
      Gosub Motor_off
      Gosub Baca_sonic4
      If Sonic4 = 0 Or Sonic4 > 600 Then Exit Do
      Gosub Baca_sonic1
      If Sonic1 > 0 And Sonic1 < 500 Then
         Gosub Haluan_kanan
      End If
      Waitms 5
   Loop
   Gosub Haluan_kiri

   Gosub Motor_off
   Return

Cek_kanan:
   Gosub Motor_off
   Do
'      Baca_sonic1
      Gosub Baca_sonic2
      Gosub Motor_kiri
      If Sonic2 = 0 Or Sonic2 > 500 Then Exit Do
      Waitms 5
   Loop
   Gosub Motor_off
   Do
      Gosub Motor_maju_pelan
      Waitms 20
      Gosub Motor_off
      Gosub Baca_sonic2
'      Gosub Motor_maju_pelan
      If Sonic2 = 0 Or Sonic2 > 600 Then Exit Do
      Gosub Baca_sonic1
      If Sonic1 > 0 And Sonic1 < 500 Then
         Gosub Haluan_kiri
      End If
      Waitms 5
   Loop
   Gosub Haluan_kanan
```

```
   Gosub Motor_off
   Return

Lurus_pelan:
   Gosub Motor_off
   Gosub Baca_sonic1
   If Sonic1 = 0 Or Sonic1 > 500 Then
      Gosub Motor_maju_pelan
      Waitms 100
   Else
      Gosub Cek_lurus
   End If
   Gosub Motor_off
   Return

Cek_lurus:
   Gosub Motor_off
   Gosub Baca_sonic2
   Gosub Baca_sonic4
   If Sonic2 = 0 And Sonic4 = 0 Then
      Gosub Haluan_kiri
      Gosub Motor_off
      Gosub Motor_maju_pelan
      Do
         Gosub Motor_off
         Gosub Baca_sonic2
         If Sonic2 = 0 Or Sonic2 > 500 Then Exit Do
         Gosub Motor_maju_pelan
         Waitms 20
      Loop
      Gosub Haluan_kanan
      Gosub Motor_maju_pelan
   Elseif Sonic2 > 500 And Sonic4 > 500 Then
      Gosub Haluan_kiri
      Gosub Motor_off
      Gosub Motor_maju_pelan
      Do
         Gosub Motor_off
         Gosub Baca_sonic2
         If Sonic2 = 0 Or Sonic2 > 500 Then Exit Do
         Gosub Motor_maju_pelan
         Waitms 20
      Loop
      Gosub Haluan_kanan
      Gosub Motor_maju_pelan
   Elseif Sonic2 > 1 And Sonic2 < 500 Then
      Gosub Haluan_kiri
```

```
      Gosub Motor_off
      Gosub Motor_maju_pelan
      Do
         Gosub Motor_off
         Gosub Baca_sonic2
         If Sonic2 = 0 Or Sonic2 > 500 Then Exit Do
         Gosub Motor_maju_pelan
         Waitms 20
      Loop
      Gosub Haluan_kanan
      Gosub Motor_maju_pelan
   Elseif Sonic4 > 1 And Sonic4 < 500 Then
      Gosub Haluan_kanan
      Gosub Motor_off
      Gosub Motor_maju_pelan
      Do
         Gosub Motor_off
         Gosub Baca_sonic4
         If Sonic4 = 0 Or Sonic4 > 500 Then Exit Do
         Gosub Motor_maju_pelan
         Waitms 20
      Loop
      Gosub Haluan_kiri
      Gosub Motor_maju_pelan
   End If
   Gosub Motor_off
   Return

'(
Cek_lurus:
   Gosub Motor_off
   Gosub Baca_sonic2
   Gosub Baca_sonic4
   If Sonic2 > Sonic4 Then
      Gosub Haluan_kanan
      Gosub Motor_off
      Do
         Gosub Motor_off
         Gosub Baca_sonic4
         If Sonic4 = 0 Or Sonic4 > 500 Then Exit Do
         Gosub Motor_maju_pelan
         Waitms 20
      Loop
      Gosub Haluan_kiri
   Elseif Sonic2 < Sonic4 Then
      Gosub Haluan_kiri
      Gosub Motor_off
      Do
```

```
         Gosub Motor_off
         Gosub Baca_sonic2
         If Sonic2 = 0 Or Sonic2 > 500 Then Exit Do
         Gosub Motor_maju_pelan
         Waitms 20
      Loop
      Gosub Haluan_kanan
   Elseif Sonic2 = 0 Or Sonic2 > 500 Then
      Gosub Haluan_kanan
      Gosub Motor_off
      Do
         Gosub Motor_off
         Gosub Baca_sonic4
         If Sonic4 = 0 Or Sonic4 > 500 Then Exit Do
         Gosub Motor_maju_pelan
         Waitms 20
      Loop
      Gosub Haluan_kiri
   Elseif Sonic4 = 0 Or Sonic4 > 500 Then
      Gosub Haluan_kiri
      Gosub Motor_off
      Do
         Gosub Motor_off
         Gosub Baca_sonic2
         If Sonic2 = 0 Or Sonic2 > 500 Then Exit Do
         Gosub Motor_maju_pelan
         Waitms 20
      Loop
      Gosub Haluan_kanan
   End If
   Gosub Motor_off
   Return
')

Lurus_cepat:
   Gosub Motor_off
   Gosub Baca_sonic1
   If Sonic1 = 0 Or Sonic1 > 500 Then
      Gosub Motor_maju_cepat
      Waitms 100
   Else
      Gosub Cek_lurus
   End If
   Gosub Motor_off
   Return
Haluan_kanan:
   Gosub Motor_off
   Gosub Baca_sonic2
```

```
   If Sonic2 = 0 Or Sonic2 > 500 Then
      A = 0
      Do
         Incr A
         Gosub Motor_kanan
         If A > 4 Then Goto Haluan_kanan_akhir
         Waitms 20
      Loop
   Else
      Gosub Lurus_pelan
   End If
Haluan_kanan_akhir:
   Gosub Motor_off
   Return
Haluan_kiri:
   Gosub Motor_off
   Gosub Baca_sonic4
   If Sonic4 = 0 Or Sonic4 > 500 Then
      A = 0
      Do
         Incr A
         Gosub Motor_kiri
         If A > 4 Then Goto Haluan_kiri_akhir
         Waitms 20
      Loop
   Else
      Gosub Lurus_pelan
   End If
Haluan_kiri_akhir:
   Gosub Motor_off
   Return
Motor_maju_pelan:
   Set M3_putar : Set M4_putar
   Waitms 10
   Pwm1a = 1000 : Pwm1b = 1000
   Waitms 250
   Return
Motor_maju_cepat:
   Set M3_putar : Set M4_putar
   Waitms 10
   Pwm1a = 1000 : Pwm1b = 1000
   Waitms 250
   Return
Motor_mundur:
   Reset M3_putar : Reset M4_putar
   Pwm1a = 600 : Pwm1b = 600
   Waitms 20
   Return
```

```
Motor_kanan:
   Reset M3_putar : Set M4_putar
   Pwm1a = 1000 : Pwm1b = 1000
   Waitms 50
   Return
Motor_kiri:
   Set M3_putar : Reset M4_putar
   Pwm1a = 1000 : Pwm1b = 1000
   Waitms 50
   Return
Motor_off:
   Pwm1a = 0 : Pwm1b = 0
   Waitms 20
   Return
```

DAFTAR PUSTAKA

[1]. Safaat H, Nazarudin. "Pemrograman Aplikasi *Mobile* Smartphone dan Tablet PC Berbasis Android". Informatika, Bandung, 2012.

[2]. Pratama, Widianto. "Tutorial Androd Programing .pdf" *http://greenbel.wordpress.com/*, 2011

[3]. _____.,"Electronic Compass on GPS"., <URL: http://blog.csdn.net/xuing/article/details/1741571>, Maret 2012

[4]. _____.,"What is Declination?" ., <URL: ., <URL: http://www.seattlerobotics.org/guide/servos.html>, Oktober 2014

[5]. _____,"Android (operating system)", <URL: http://en.wikipedia.org/wiki/Android_(operating_system)>, Oktober 2014.

[6]. <URL:http://*Developer*.android.com/reference/android/*hardware*/Sensor.html >, September 2014.

[7]. _____,"Android.location.Location", <URL:http://*Developer*.android.com/reference/android/location/Location.html >, September 2014.

[8]. _____,"SRF04 Ultrasonic Ranger", <http://www.robot-electronics.co.uk/htm/srf04tech.htm>, 2012.

[9]. Firdaus, Nurizal, "Rancang bangun sistem navigasi *mobile* robot berbasis device android"**,** Tugas Akhir, ITS 2012.

[10]. Aji "Rancang bangun sistem navigasi *mobile* robot berbasis android menggunakan peta GPS dengan metode multi waypoint" Tugas Akhir, ITS 2013.

[11]. _____,"datasheet atmega328 .pdf", Atmel, 2009.

[12]. Eko putra, Agfianto “Atmega 16 dan Bascom AVR .pdf”____,2010

[13]. Ableson, Frank. *“Tapping into Android’s Sensor”* ,<URL:*http://www.ibm.com/developerworks/opensource/library/os-android-sensor/Tapping into Android's sensors.htm >*, Oktober 2014.

 # Halaman ini sengaja dikosongkan #

BIODATA PENULIS

Lukas Yulianto, adalah nama lengkap penulis dengan nama panggilan Lukas. Penulis dilahirkan di kotabumi Lampung tanggal 04 Juli 1990, putra kedua dari tiga bersaudara pasangan Bp.Joko Pitono dan Ibu Lilik Suryani. Penulis memulai pendidikannya dari TK Darmawanita Lampung Utara, kemudian melanjutkan studinya di SDN 01 Bungamayang Lampung Utara, SMPN 01 Bungamayang Lampung Utara, dan SMAK Frateran Malang. Setelah menamatkan SMA, penulis melanjutkan studinya di Politeknik Negeri Malang (POLINEMA) tepatnya pada Program Studi Elektronika dan lulus pada Agustus 2011. Pada tahun 2012 penulis sempat bekerja selama 6 bulan sebelum akhirnya melanjutkan studi sarjana di Teknik Elektro ITS (Institut Teknologi Sepuluh Nopember), yang tetap pada bidang studi Elektronika. Pada bulan Januari 2015 penulis mengikuti seminar dan ujian Tugas Akhir sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik Elektro dari Institut Teknologi Sepuluh Nopember Surabaya.

Email : paten26@gmail.com

 # Halaman ini sengaja dikosongkan #