# NAVIGASI ROBOT BERBASIS *PATH PLANNING* DENGAN PEMULIHAN JALUR OTOMATIS

**SKRIPSI**

Oleh

**Alexander Rafsanjani**

**NIM 131910201111**

**JURUSAN TEKNIK ELEKTRO STRATA 1**
**FAKULTAS TEKNIK**
**UNIVERSITAS JEMBER**
**2017**

**NAVIGASI ROBOT BERBASIS *PATH PLANNING* DENGAN PEMULIHAN JALUR OTOMATIS**

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Teknik Elektro (S1)
dan mencapai gelar Sarjana Teknik

Oleh :
**Alexander Rafsanjani**
**NIM 131910201111**

**JURUSAN TEKNIK ELEKTRO STRATA 1**
**FAKULTAS TEKNIK**
**UNIVERSITAS JEMBER**
**2017**

## PERSEMBAHAN

Dengan ini saya persembahkan skripsi kepada:

1. Allah SWT yang Maha Pengasih lagi Maha Penyayang.
2. Kedua orang tua tercinta, Bapak Ardi Marshanto dan Ibu Siti Rosdiana serta kedua adikku Dhaniar H. T. dan Imantaka R. M. atas kasih sayang, pengorbanan, dan kesabaran yang tiada tara serta doa yang selalu menyertai.
3. Guru – guru mulai SDN Ditotrunan 1 Lumajang, SMPN 2 Lumajang, SMA Negeri 1 Lumajang dan dosen-dosen Teknik Elektro Universitas Jember. Terima kasih untuk ilmu dan pengalaman yang telah diajarkan selama ini.
4. Para anggota Begundal Crew, yang selalu menyemangati dan menghibur ketika sedang kesulitan.
5. Rekan-rekan satu DPU dan DPA yang selalu menemani saya berjuang mulai awal pengerjaan skripsi hingga selesai.
6. Keluarga Intel 2013 yang selalu membantu, menyemangati dan selalu mendampingi saya selama pengerjaan skripsi ini.
7. Almamater Teknik Elektro Universitas Jember.

**MOTTO**

"Maka sesungguhnya bersma kesulitan ada kemudahan. Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari sesuatu urusan), tetaplah bekerja keras (untuk urusan yang lain). Dan hanya kepada Tuhanmulah engkau berharap."

(QS. Al-Insyirah: 6-8)

"Barangsiapa ingin doannya terkabul dan dibebaskan dari kesulitanya hendaaklah dia membebaskan kesulitan orang lain."

(HR. Ahmad)

**PERNYATAAN**

Saya yang bertanda tangan dibawah ini:

Nama: Alexander Rafsanjani

NIM : 131910201111

Menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul ”Navigasi Robot Berbasis *Path Planning* dengan Pemulihan Jalur Otomatis” adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab penuh atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyatan ini tidak benar.

Jember, 14 Agustus 2017

Yang menyatakan,

Alexander Rafsanjani

NIM 131910201111

**SKRIPSI**

# NAVIGASI ROBOT BERBASIS *PATH PLANNING* DENGAN PEMULIHAN JALUR OTOMATIS

**Alexander Rafsanjani**
**NIM 131910201111**

Pembimbing

Dosen Pembimbing Utama : Mohamad Agung Prawira Negara, S.T., M.T.

Dosen Pembimbing Anggota : Dodi Setiabudi, S.T., M.T.

**PENGESAHAN**

Skripsi berjudul ”Navigasi Robot Berbasis *Path Planning* dengan Pemulihan Jaluur Otomatis” Telah diuji dan disahkan pada :

Hari : Senin

Tanggal : 14 Agustus 2017

Tempat : Fakultas Teknik Universitas Jember

Tim Penguji

| Ketua, | Sekretaris, |
|---|---|
| Mohamad Agung P. N., S.T., M.T.<br>NIP 19871217 201212 1 003 | Dodi Setiabudi, S.T., M.T.<br>NIP 19840531 200812 1 004 |
| Anggota I, | Anggota II, |
| Khairul Anam, S.T., M.T., Ph.D<br>NIP 19780405 200501 1 002 | Catur Suko Sarwono, S.T .,M.Si.<br>NIP 19680119 199702 1 001 |

Mengesahkan

Dekan Fakultas Teknik
Universitas Jember,

Dr.Ir. Entin Hidayah M.U.M
NIP 19661215 199503 2 001

# RINGKASAN

**Navigasi Robot Berbasis *Path Planning* dengan Pemulihan Jalur Otomatis: Alexander Rafsanjani, 131910201111: 2017: 123 halaman: Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.**

Pada saat ini perkembangan otomatisasi robot sangat signifikan, mulai dari berkaki dan beroda yang tergolong *mobile* robot maupun yang bukan *mobile* seperti robot lengan dan sebagainya yang tidak berpindah tempat. Sistem navigasi robot tanpa awak terus dikembangkan. Pada sebuah perkantoran sistem antar dalam untuk makanan ringan masih manual, terdapat sebuah ide untuk membuat robot otomatis dengan sebuah *line follower* akan tetapi akan banyak interupsi dari luar, pada umumnya masalah cahaya. Terdapat cara lain untuk mengatasi sistem tersebut salah satunya dengan *Path planning* dengan timbal balik *odometry*.

*Odometry* adalah penggunaan data dari sensor pergerakan untuk memperkirakan perubahan posisi dari waktu ke waktu. *Odometry* ini akan memetakan posisi robot dalam sumbu Cartesian. Sehingga akan didapatkan data posisi berupa titik koordinat (*path*) dan arah hadap (*heading*) dari robot tersebut. David P. Anderson membuat robot yang dalam navigasinya tidak menggunakan referensi dari luar seperti GPS dan lainnya.

Pada penelitian ini robot dituntut untuk melakukan navigasi secara otomatis untuk menempuh *path* atau koordinat yang telah ditentukan. Dengan rumus pemodelan robot *differential drive* yang dikombinasikan dengan pemodelan *unicycle* dari robot yang nantinya didapatkan pemodelan robot untuk menempuh kordinat yang ditentukan serta timbal balik dari odometri yang digunakan untuk mengetahui posisi relatif dari robot. Penambahan sensor kompas yang terpasang diantara kedua roda robot diharapkan mampu untuk menambah keakuratan posisi robot.

Pada pengujian yang dilakukan menggunakan 1 *path*, 2 *path* dan *path planning*. Pada setiap pengujian dilakukan 5 kali percobaan. Sebelum dilakukan

pengujiantersebut dilakukan pengujian pada sensor-sensor yang telah digunakan untuk mengetahui robot menempuh posisi relatifnya. *Rotary Encoder* mempunyai nilai yang dapat dikatakan sama ketika menempuh jarak yangn telah ditentukan dengan rata-rata kesalahan hanya 0,603 % membuat dapat menempuh titik (97, 0) ketika mendapat input (100, 0) dengan selisih sekitar 3 cm. Selanjutnya dilakukan percobaan 5 kali pada setiap pengujian dan robot dapat memulihkan jalurnya secara otomatis dengan adanya PID Ziegler-Nichols meskipun terdapat osilasi sudut terhadap titik tujuan pada gangguan 10 cm dan 15 cm. Didapatkan hasil, Robot dapat melakukan *Path Planning* dengan *Error* paling besar terletak pada *Path Planning* 4 ketika robot menuju titik (10, 30), pada sumbu Y dengan besar rata-rata 80% dibanding X dengan besar 33,333%.. Pengujian selanjutnya dengan tanpa PID akan tetapi dengan hanya menggunakan Kp = 5 dan Kp = 10. Didapatkan hasil, *Path Planning* 4 dengan menggunakan Kp membuat pergerakan robot melengkung akan tetapi tetap dapat menempuh titik yang telah dimasukan dengan *error* paling besar pada *Path Planning* 4 dengan 100% kesalahan pada sisi X dan 33,3333% dengan Kp = 10.

Pada pengujian robot dengan menggunakan Kp = 5 dengan *Input Path Planning* 4 Robot tidak dapat menempuh titik tujuan dengan *error* pada kedua sumbunya sebesar 75% dan Robot terus bergerak kesamping dan berosilasi ke arah kiri atau sumbu Y.

# SUMMARY

**Based Path Planning Robot Navigation with Automatic Route Recovery: Alexander Rafsanjani, 131910201111: 2017: 123 pages: Department of Electrical Engineering, Faculty of Engineering, University of Jember.**

At this time the development of robot automation is very significant, ranging from legged and wheeled belonging to mobile robot and non-mobile like robot arm and so on that do not move. Unmanned robot navigation system continues to be developed. In an in-house inter-office system for snacks is still manual, there is an idea to make an automatic robot with a line follower but it will be a lot of interruptions from outside, generally a matter of light. There are other ways to overcome the system one of them with Path planning with reciprocal odometry.

Odometry is the use of data from motion sensors to estimate position changes over time. This Odometry will map the position of the robot in the Cartesian axis. So that will get the position data in the form of coordinate point (path) and direction of the heap (heading) of the robot. David P. Anderson makes robots that in navigation do not use references from outside such as GPS and others.

In this study the robot is required to navigate automatically to take the path or coordinates that have been determined. With the modeling model of robot differential drive combined with unicycle modeling of the robot that will be obtained robot modeling to pursue the specified coordinates and reciprocal of odometry used to determine the relative position of the robot. The addition of a compass sensor mounted between the two robot wheels is expected to increase the accuracy of the robot position.

In the test conducted using 1 path, 2 path and path planning. In each test 5 experiments were performed. Prior to the examination done testing on the sensors that have been used to know the robot to relative position. Rotary Encoder has a

similar value when traveling a predetermined distance with an average error of only 0.603% making it able to take a point (97,0) when it gets input (100,0) by a difference of about 3 cm. The experiment was then performed 5 times on each test and the robot was able to recover the path automatically in the presence of a Ziegler-Nichols PID although there was an angle oscillation to the destination point at 10 cm and 15 cm disturbance. Obtained results, the Robot can perform the most abundant Path Planning with Error located on Path Planning 4 when the robot to the point (10, 30), on the Y axis with an average average of 80% compared to X with 33.333% .. Further testing with no PID But using only Kp = 5 and Kp = 10. Obtained results, Path Planning 4 using Kp makes the robot movement curve but still can take the point that has been entered with the biggest error on Path Planning 4 with 100% error on the X side And 33.3333% with Kp = 10.

In robot testing using Kp = 5 with Input Path Planning 4 Robot can not reach the destination point with error on both axes by 75% and Robot continues to move sideways and oscillates to the left or Y axis.

# PRAKATA

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Komparasi Performa Panel Surya Monocrystalline dan Polycrystalline pada Atap Gedung CDAST Universitas Jember". Skripsi ini disusun sebagai salah satu syarat untuk menyelesaikan program studi strata satu (S1) Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.Selama penyusunan skripsi ini penulis mendapat bantuan berbagai pihak, untuk itu penulis mengucapkan terima kasih kepada :

1. Ibu Dr.Ir. Entin Hidayah M.U.M selaku dekan Fakultas Teknik Universitas Jember.
2. Bapak Mohamad Agung P. N., S.T., M.T., selaku dosen pembimbing utama dan Bapak Dodi Setiabudi, S.T., M.T., selaku dosen pembimbing anggota yang telah rela meluangkan waktu, pikiran serta motivasi dalam penyusunan skripsi ini.
3. Bapak Samsul Khairul Anam, S.T., M.T., Ph.D , selaku dosen penguji utama dan Bapak Catur Suko Sarwono, S.T ., selaku dosen penguji anggota dan Komisi Bimbingan Program Studi S1 Teknik Elektro yang telah memberikan kritik dan saran yang membangun sehingga sangat membantu terhadap penyempurnaan skripsi ini.
4. Bapak M. Agung Prawira N., S.T.,M.T., selaku dosen pembimbing akademik yang telah membimbing dan menanamkan rasa disiplin dan tanggung jawab dengan apa yang dilakukan selama penulis menjadi mahasiswa.
5. Kedua orang tua tercinta, Bapak Ardi Marshanto dan Ibu Siti Rosdiana serta kedua adikku Dhaniar H. T. dan Imantaka R. M. atas kasih sayang, pengorbanan, dan kesabaran yang tiada tara serta doa yang selalu menyertai dan rekan spesial yang mendukung dalam suka maupun duka Andra Dwi Saputri.
6. Rekan-rekan Asisten Laboratium Sistem Kendali Ahmad Iqbal Nasrudin, M. Binawan Satrio, Erina Dyah Atsari, Ahmad Kusyairi, Arif Fahmi, Aria

Ghosal , Nuharizka, Ahmad Fauzi, Martin Sanjaya dan Arif atas bantuan yang telah diberikan selama ini.

7. Rekan Begundal Crew Intho Nurshauma S., Erwin Setiyandani, Edi Tri Kurniawan, Ade Firmansyah, Dicky Hakim, Nur Wahyu Utomo, Nanda Yudhakawira, Bagus Lintang, M. Binawan Satriyo, Ekky Wahyu Afrian, Fajar Gunawan, Diego Jaka Sundang, Nuh Firmansyah, Mirza Kurnia dan Arif Fahmi Ubaidillah yang selama ini telah saling menguatkan satu sama lain.
8. Rekan-rekan Fakultas Teknik Universitas Jember khususnya rekan-rekan Teknik Elektro Angkatan 2013 yang tidak dapat disebutkan satu per satu, selama ini telah memberikan pengalaman hidup selama penulis menjadi keluarga Fakultas Teknik Universitas Jember.
9. Keluarga baru kelompok KKN 14 yang telah memberikan pengalaman, dukungan, dan motivasi dalam penyelesaian skripsi ini.
10. Rekan-rekan SMAN 1 Lumajang Gombes, Mbah, Fendik, Bondan, Gecol dan teman-teman pendaki gunung lainnya
11. Serta seluruh pihak yang telah membantu dalam mengerjakan skripsi ini yang tidak bisa disebutkan namanya satu persatu.

Penulis juga menerima segala kritik dan saran dari semua pihak demikesempurnaan skripsi ini.Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

Jember, 14 Agustus 2017

Penulis

# DAFTAR ISI

Halaman

# DAFTAR GAMBAR

Halaman

# DAFTAR TABEL

Halaman

# BAB 1. PENDAHULUAN

## 1.1 Latar Belakang

Pada saat ini perkembangan otomatisasi robot sangat signifikan, mulai dari berkaki dan beroda yang tergolong *mobile* robot maupun yang bukan *mobile* seperti robot lengan dan sebagainya yang tidak berpindah tempat. Sistem navigasi robot tanpa awak terus dikembangkan. Pada sebuah perkantoran sistem antar dalam untuk makanan ringan masih manual, terdapat sebuah ide untuk membuat robot otomatis dengan sebuah *line follower* akan tetapi akan banyak interupsi dari luar, pada umumnya masalah cahaya. Terdapat cara lain untuk mengatasi sistem tersebut salah satunya dengan *Path planning* dengan timbal balik *odometry*.

*Odometry* adalah kemapuam robot untuk menemukan posisi relatifnya berdasarkan sensor-sor nyata yang digunakan dalam waktu ke waktu atau dalam keadaan nyata. Robot akan ditempatkan pada sumbu Cartesian yang mempunyai nilai berupa titik cordinat (*path*) dan arah hadap robot (*heading*) akan tetapi Anderson tidak menambahkan sensor GPS atau yang lainnya untuk navigasi dari robotnya (Anderson, 2006).

Judul skripsi ini diambil inspirasi beberapa jurnal mengenai odometri tentang *path tracking* dan odometri dengan pemulihan jalur otomatis. Pada jurnal tersebut penulis tidak nampak menggunakan metode kinematik gerak *differential driv*e pada robot dan sensor tambahan sebagai kompensasi *Error* dari selip roda yang ditimbulkan. Pada robot yang akan diteliti kali ini akan menggunakan metode gerak kinemmatik gerak dari *Differential Drive* dengan menambahkan sensor kompas untuk mengkompensasi selip dari roda.

Sebagai syarat dalam penyelesaian studi di Fakultas Teknik Universitas Jember, penyusun sebagai salah satu mahasiswa Jurusan Teknik Elektro Universitas Jember berusaha mengembangkan suatu bentuk ilmu pengetahuan dan teknologi, berupa pembuatan "**Navigasi Robot Berbasis *Path Planning* dengan Jalur Otomatis**". Penelitian ini bermaksud menavigasi robot untuk mencapai suatu titik tertentu dengan kordinat x dan y serta dapat memulihkan jalur otomatis ketika robot mendapat ganguan dari luar.

## 1.2 Rumusan Masalah

Dari latar belakang yang telah diuraikan di atas, maka pokok permasalahan yang terdapat dalam Skripsi ini sebagai berikut :

1. Bagaimana cara robot melakukan odometri ?
2. Bagaimana cara robot melakukan navigasi *path planning*?
3. Bagaimana cara memulihkan jalur otomatis?

## 1.3 Batasan Masalah

Dari judul yang telah diambil dapat ditarik beberapa rumusan masalah sebagai berikut :

1. Pergerakan robot hanya dilakukan dalam skala kecil.
2. Pergerakan robot dilakukan pada bidang datar.
3. Sensor Kompas HMC5883L untuk mengkompesasi *turning* pada robot ketika berpindah kordinat (*path*).

## 1.4 Tujuan Penelitian

Tujuan dari pembuatan alat ini yaitu :

1. Robot dapat menemukan posisi relatifnya pada bidang cartesian.
2. Mengoptimalkan navigasi robot dengan *path planning* untuk mencapai titik kordinat.
3. Dapat memulihkan jalur secara otomatis ketika mendapatkan gangguan.

## 1.5 Manfaat Penelitian

Manfaat yang diharapkan dengan adanya alat ini yaitu :

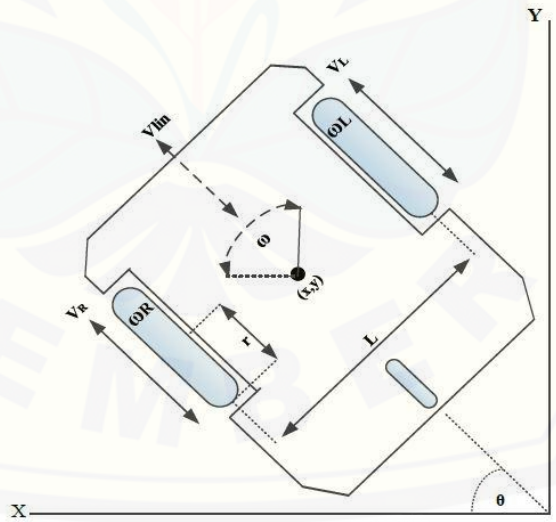
1. Robot menemukan posisi relatifnya melalui odometri.
2. Navigasi robot dapat lebih optimal berdasarkan *path planning.*
3. Robot memulihkan jalur secara otomatis ketika robot mendapat gangguan.

# BAB 2. TINJAUAN PUSTAKA

Untuk menegetahui referensi metode dan dasar-dasar dari komponen yang telah digunakan pada skripsi yang berjudul "*Navigasi Robot Berbasis Path Planning dengan Pemulihan Jalur Otomatis*" ini, maka diperlukan adanya teori yang dapat membantu agar perancangan alat ini dapat bekerja dan berjalan dengan baik, sehingga sesuai dengan keinginan dan mendapat hasil yang maksimal. Komponen yang digunakan dalam perancangan alat ini terdiri dari beberapa komponen, diantaranya:

## 2.1 Kinematik *Differential Wheeled Robot*

*Differential Wheeled Robot* adalah robot yang bergerak berdasarkan dua roda terpisah ditempatkan di kedua sisi tubuh robot. Pergerakan robot bergantung pada putaran rodanya dengan bantuan *free—wheele* dibagian depan atau belakang tanpa da kemudi tambahan.




Gambar 2.1 *Differential Wheeled Robot* dalam Cartesian (Sumber : Mynt, 1991)

Dari Gambar 2.1 konstruksi simpel dari sebuah *differential drive robot*, dengan hanya menggunakan L jarak antar kedua roda dan r sebagai radius dari roda. Pada umumnya ketika Vr = Vl robot akan berjalan maju seiring dan Vr = -Vl atau sebaliknya robot akan berputar seiring dengan perubahan waktu. Hal ini memotivasi penempatan *body-frame* asal mula robot di bagian tengah poros antara roda. Pada pengamatan tersebut dengan syarat tidak terdapat translasi/murni persamaan dari differential drive ketika bergerak maju dan berputar. jika kedua roda robot mempunyai *rate* yang sama meskipun kedua roda mempunyai arah yang berbeda dapat dinyatakan dalam pemodelan sebagai berikut (Steve M. LaValle, 2006).

$$X = \frac{r}{2}(Vr + Vl)\cos\theta \quad \text{.....(2.1)}$$

$$Y = \frac{r}{2}(Vr + Vl)\sin\theta \quad \text{.....(2.2)}$$

$$\Phi = \frac{r}{L}(Vr - Vl) \quad \text{.....(2.3)}$$

Persamaan diatas ditranslasikan dalam bentuk *unicyle* model. Dalam penjelasannya *unicycle* model adalah persamaan ketika robot mempunyai *Input* V (kecepatan roda) dan $\omega$ (kecepatan sudut). Hasil translasi dari persamaan di atas menjadi seperti berikut.

$$X = V\cos\theta \quad \text{.....(2.4)}$$

$$Y = V\sin\theta \quad \text{.....(2.5)}$$

$$\Phi = \omega \quad \text{.....(2.6)}$$

Dari penggabungan rumus tersebut didapatkan rumus sebagai berikut.

$$V = \frac{R}{2}(Vr + Vl) => \frac{2V}{R} = Vr + Vl \quad \text{.....(2.7)}$$

$$\omega = \frac{R}{L}(Vr - Vl) => \frac{\omega L}{R} = Vr - Vl \quad \text{.....(2.8)}$$

Dari persamaan diatas dapat ditarik bentuk Kinematik dari *Differential Drive Robot* menjadi seperti berikut (Morgan, 2011).

$$Vr = \frac{2V + \omega L}{2R} \quad \text{.....(2.9)}$$

$$Vl = \frac{2V - \omega L}{2R} \quad \text{.....(2.10)}$$

## 2.2 *Path Planning* dan *Odometry*

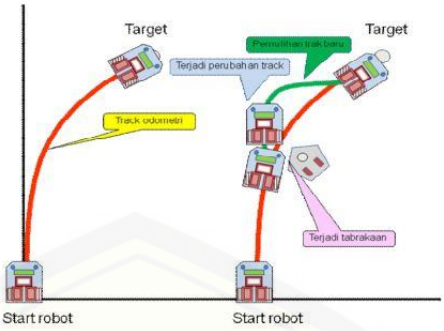
*Path planning* adalah menentukan jalur sebuah robot berdasarkan *path* (kordinat). *Odometry* sendiri adalah adalah kemapuam robot untuk menemukan posisi relatifnya berdasarkan sensor-sor nyata yang digunakan dalam waktu ke waktu atau dalam keadaan nyata. Robot akan ditempatkan pada sumbu Cartesian yang mempunyai nilai berupa titik cordinat (*path*) dan arah hadap robot (*heading*).

Misalkan ketika robot mempunyai posisi awal (0 , 0 , 0) dan akan bergerak lurus dengan kecepatan 2 m/s dan mencapai titik (8, 0, 0) dengan baik itu esensi dari odometri. Pergerakan robot pada odometri diibratkan robot bergerak dengan poros sebuah busur.

Metode ini sensitif terhadap kesalahan karena integrasi pengukuran kecepatan dari waktu ke waktu untuk memberikan perkiraan posisi. Cepat dan akurat pengumpulan data, kalibrasi peralatan, dan pengolahan yang diperlukan dalam kebanyakan kasus untuk *odometry* untuk digunakan secara efektif. *Odometry* disini sebagai umpan balik dalam penentuan beberapa *path* sebagai sebuah *planning* robot utuk menempuh suatu titik lokasi (Bayu, 2011).

## 2.3 Pemulihan Jalur

Dalan penentuan suatu lokasi rlatif dari sebuah robot menggunakan *rotary encoder* (*odometry*) akan selalu didapatkan sebuah *heading* (sudut belok). Dari *heading* tersebut terdapat sebuah *heading error* yang dapat ditambahkan sebuah kontrol untuk dapat kembali ke jalur seperti semula. Seperti contoh ketika terjadi tabrakan robot akan menyimpang dari jalur, simpangan tersebut disebut *heading error* (Jusuf, 2011).

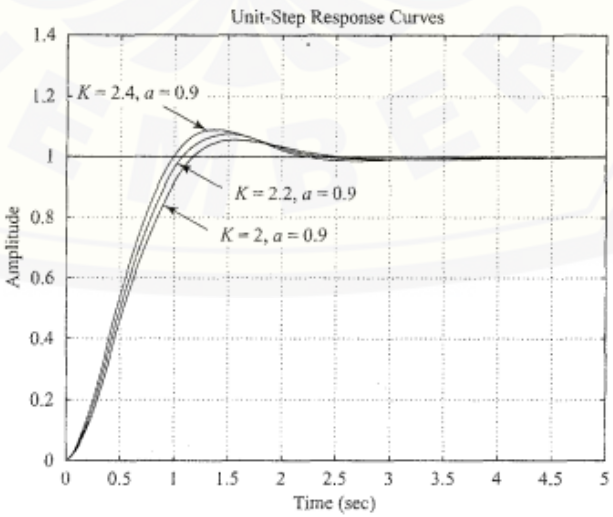



Gambar 2.2 Robot tanpa gangguan dan dengan gangguan (Jusuf, 2011)

Pergerakan robot dicontohkan melalui gambar 2.2 dimana robor bergerak tanpa dan dengan gangguan dari luar.

## 2.4 PID *Controller*

Sebuah kontrol *proporsional-integral-derivatif* (PID) adalah suatu metode kontrol umpan balik dari sebuah sensor. *Error* yang didapatkan dari selisih *Set Point* dan *Point Variable* menjadi acuan penting dalam metode ini. Kontroler ini akan ter menerus menghitung kesalah yang terjadi dari perbandingan kedua variable tersebut hingga mendapatkan respon yang baik terhadap waktu (t).




Gambar 2.3 Bentuk grafik respon PID (Sumber : Ogata, 2002)

### 2.4.1 Metode Ziegler-Nichols

Metode *tuning* Ziegler-Nichols adalah metode kontrol *tuning* dari PID *controller*. metode *tuning* dikembangkan oleh John G. Ziegler dan Nathaniel B. Nichols. Hal ini dilakukan dengan menetapkan I (integral) dan D derivatif) ke nol. "P" (proporsional) gain, Kp kemudian ditingkatkan (dari nol) hingga mencapai *gain* tertinggi Ku, dimana *output* dari kontrol *loop* memiliki osilasi stabil dan konsisten. Ku dan periode osilasi Tu digunakan untuk mengatur P, I, dan D (Ogata, 2002).

Tabel 2.1 Perhitungan Metode Ziegler-Nichols (Sumber : Ogata, 2002)

| Metode Ziegler-Nichols | | | |
|---|---|---|---|
| Tipe Kontrol | Kp | Ti | Td |
| P | 0,5 Kc | - | - |
| PI | 0,45 Kc | Tc/1,2 | |
| PD | 0,8 Kc | - | Tc/8 |
| No Overshoot | 0,2 Kc | TC/2 | Tc/3 |

## 2.5 Arduino Mega

Perkembangan dari Arduino sendiri semakin pesan mulai dari desain minimalis dari sebuah Mikrokontroler jenis Arduino (Arduino Pro Mini) sampai Arduino Due yang mempunyai kecepetan pemrosesan data yang sangat tinggi. Arduino Mega sendiri sendir merupakan perkembangan dari Arduino UNO dengan Arduino Mega mempunya Pin yang lebih banyak dan menggunakan memori yang lebih besar juga dengan mengandalkan ATmega 2560. Spesifikasi dari ATmega 2560 dapa dilihat pada Tabel 2.2.

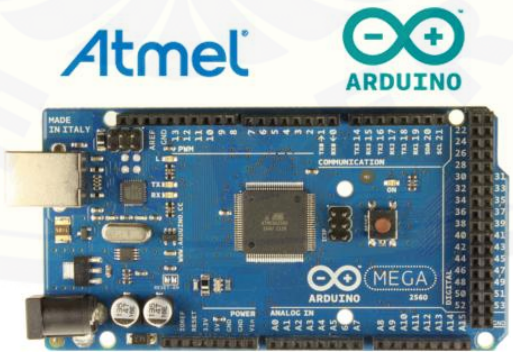
### 2.5.1 Daya (*Power*)

Arduino Mega membutuhkan suplai tegangan sebesar 5V. Suplai tersebut berupa tegangan DC atau tegangan searah. Suplai tegangan dapat disalurkan melalui kabel USB *printer*. Pada Arduino Mega juga terdapat sebuah regulator tegangan 5V dan 3,3V untuk membatasi tegangan yang melebihi spesifikasi

Arduino Mega. Apabila terdapat 2 *Input* tegangan sistem suplai Arduino akan otomatis melakukan pemilihan sumber daya yang sesuai untuk digunakan.

Tabel 2.2 Deskripsi *Arduino Mega*

| | |
|---|---|
| Mikrokontroler | ATMega2560 |
| Tegangan Pengoperasian | 5V |
| Tegangan *Input* yang disarankan | 7 - 12V |
| Batas Tegangan *Input* | 6 – 20V |
| Jumlah pin I/O digital | 54 *pin* digital (15 diantaranya menyediakan keluaran PWM) |
| Jumlah *pin Input analog* | 16 *pin* |
| Arud DC tiap *pin* I/O | 20mA |
| Arus DC untuk pin 3,3V | 50mA |
| *Memori flash* | 256 KB (ATMega 2560) sekitar 8 KB digunakan oleh *bootloader*) |
| SRAM | 8 KB (ATMega 2560) |
| EPROM | 4 KB (ATMega 2560) |
| *Clock Speed* | 16 MHz |




Gambar 2.4 *Board* Arduino Mega (Sumber : Cristy, 2014)

Terdapat toloeansi pada pembatas tegangan (regulator) yang digunakan pada Arduino Mega atau yang lainnya. Maksimum tegangaan yang dapat

digunakan adalah 20V dengan arus maksimuum yang dapat dilewatkan 1A. Apabila tegangan lebih dari 20V regulator akan panas dan rusak sehingga akan berimbas pada IC Arduino sendiri. Berikut adalah ketentuan suplai yang digunakan pada Arduino Mega.

a. Vin pada Arduino dapat berupa *jack power* ketika Arduino Mega mendapatkan tegangan *non*-USB. Pin Vin yang lain terdapat pada Pin I/O.
b. Pin 5V . Pin tersebut pada Arduino Mega mempunyai *output* suplai berupa tegangan 5V. Hal ini digunakan untuk mensuplai sensor-sensor analog yang pada umumnya menggunakan tegangan 5V dengan arus.
c. Pin 3V3. Pin tersebut pada Arduino Mega mempunyai output tegangan 3,3 V dengan arus sebesar 50 mA. Tegangan tersebut biasanya digunkan untuk komunikasi I2C atau Serial dengan *Bluetooth*.
d. Pin GND. Pin tersebut pada Arduino Mega sebagai *grounding*.

### 2.5.2 Memori

Penggunaan ATmega 2560 pada Arduino Mega mempunyai kapasitas 256 KB ( 8 KB digunkan untuk *bootloader*) dengan 8 KB SRAM dan 4KB EEPROM (Penyimpanan data oleh Arduino Mega yanf dapat dipanggil kembali).

### *2.5.3 Input & Output*

Pada I?O Arduino Mega mempunya 54 Pin. Pada Setiap Pin bekerja dengan 5V dengan arus 20 mA. Berikut adalah masing-masing fungsi dari Pin:

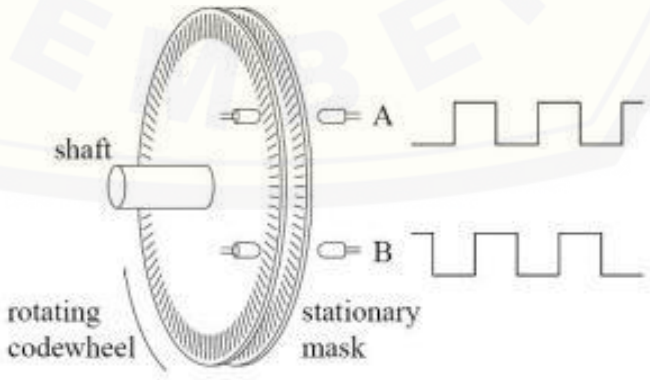
a. Pin Komunikasi Serial : Pin Komunikasi Serial terletak pada Pin 0 (RX) Pin 1 (TX) untuk *Hardware Serial*. Untuk *Software Serial* 14 sampai 19 pin digital.
b. Pin *Interrupt*. Pin tersebut bekerja ketika mendapat perintah *attach.Interrupt()* atau dengan kata lain menyisipkan perintah ditengah perintah yang sedang berjalan. Pin tersebut berada pada pin 2, 3, 18, 19, 20 dan 21.
c. Pin PWM: 1 sampai 14 pada pin digital.

d. Pin Komunikasi SPI : Pin tersebut terletak pada 50 sebgaia SS, 51 sebagai MISO dan 53 sebgai SCK. Pin tersebut pada umumnya berfungsi untuk komunikasi antar Arduino (Dede, 2014).

## *2.6 Rotary Encoder*

Gambar 2.5 *Rotary Encoder* (Sumber : *arduino.cc*)

Sensor putaran yang sering digunakan pada sekarang ini ada *Rotary Encoder*. Jadi sensor tersebut mengubah putaran roda yang linier menjadi sinyal *counter* digital Sensor ini memliki 2 disk yang berlubang didalamnya yang masing-masing mempunyai fungsi yang berbeda. Untuk disk yang pertama berfungssi untuk membangkitkan sinyal digital dari proses transmisi putarannya. Disk kedua berfungsi untuk melengkapi selip dari kode biner dan posisi dari sudut putaran. Ilustrasi dari cara kerja rotary adalah sebagai berikut. (Pitowarno, 2006).




Gambar 2.6 Prinsip Kerja *Rotary Encoder* (Sumber : Angin, 2009)

Disk berlubang yang dimaksudkan adalah sensor optis *channel* A dan *channel* B dimana yang berlubang mempunyai nilai *HIGH* dan yang tidak bernilai *LOW* yang digunakan untuk menetukan arah putar apakah putaran searah jarum jam atau berlawanan jarum jam dan hal tersebut menjadi pertimbangan *counting* dari sensor.

*Rotary Encoder* yang digunakan untuk mendeteksi kecepatan putar atau rotasi gerak dari roda pada umumnya diletakan sejajar dengan *shaft* roda atau dipasangkan secara seri dengan roda. (Agung, 2011).

## 2.7 Motor DC

Motor DC merupakan sebuah aktuator yang mempunyai suplai tegangan. DC atau searah. Tegangan searah tersebut digunakan untuk memutar rotator atau rotor. Dengan membalik fasa tegangan yang mempunyai nilai positif. Pengubahan fasa tersebut dilakukan oleh komutator dengan pembalikan fasa tersebut arus berbalik arah bersama dengan kumparan dalam medan magnet.

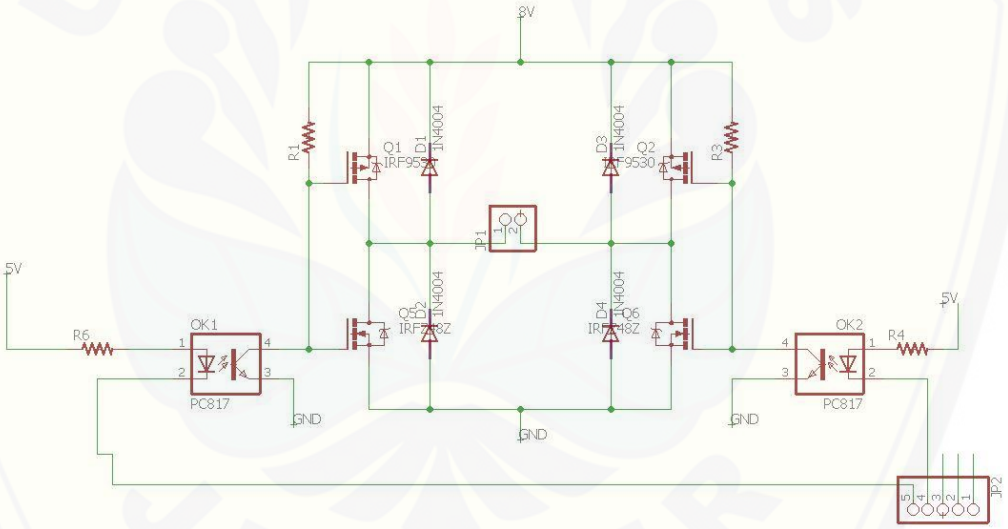
Gambar 2.7 Motor DC (Yusuf, 2015)

Spesifikasi dair Motor DC bermacam-macam berdasarkan tegangan , arus maupun torsi. Torsi pada Motor DC lebih dapat dimanipulasi dengan perbadingan *gear* yang terpasang pada *shaft* dari Motor DC.Penggunaan Motor DC lebih umum pada roda robot dengan tegangan 6 sampai 24 VDC dengan arus yang bermacam-macam tergantung kebutuhan robot. (Yusuf, 2015).

## 2.8 *Driver* H *Bridge* MOSFET

*Driver* H *Bridge* MOSFET merupakan modifikasi dari rangkaian H Bridge dan menggunakan MOSFET sebagai saklar yang mempunyai kemampuan mengalirkan arus yang besar hingga 20 Ampere. *Driver* ini bekerja 5V PWM melalui *Optocopler* sekaligus pengaman yang bekerja pada *Base* dan tegangan *Input* motor pada Kanal P dan Kanal N nya (Firman, 2015).

Pada Prinsip kerjanya, driver motor tipe MOSFET sama dengan Transistor pada umumnya dengan masukan dari sinyal PWM dari Mikrokontroler yang digunakan. Setiap *driver* mempunyai 4 pin, yaitu pin VCC, 5V, GND dan 2 Pin digital yang dihubungkan pada Mikrokontroler. Berikut gambar rangkaian dari *Driver* H *Bridge* MOSFET.




Gambar 2.8 Rangkaian *Driver* H *Bridge* MOSFET (Firman, 2015)

## 2.9 Sensor Kompas HMC5883L

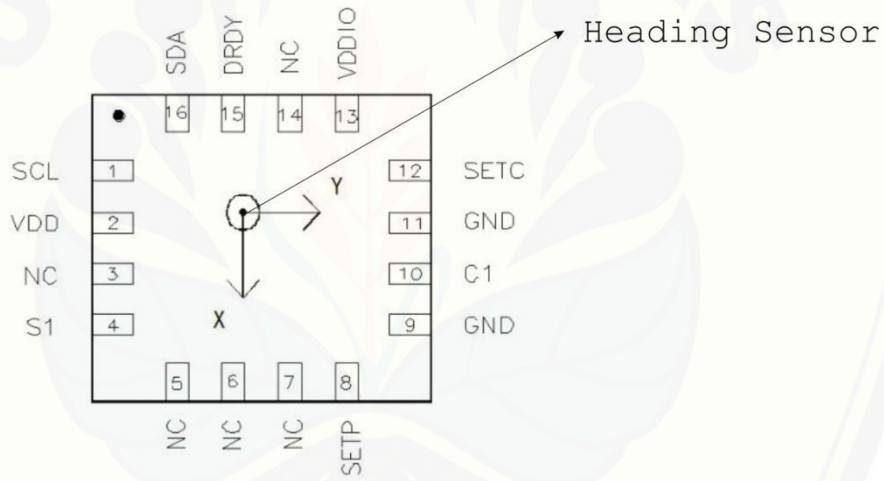
Sensor Kompas yaitu sensor yang memliki respon terhadap rotasi atau putaran, jadi sensor ini akan memiliki nilai yang berbeda saat dia berada dengan posisi hadap yang berbeda, misal jika sensor ini menghadap ke utara dengan ke selatan, maka hasilnya saat posisi menghadap ke utara akan berbeda dengan pada saat sensor menghadap ke posisi selatan, begitulah cara kerja sensor kompas, sensor kompas yang dipakai kali ini adalah sensor compass HMC5883L yang

mana sensor ini jauh lebih murah dibanding sensor kompas yang lain (Yanuar, 2014).

Untuk mendapatkan sebuah *heading* dari dapat melalui persamaan sebagai berikut.

$$\textit{Heading Sensor} = \tan^{-1}\frac{Normal\ Y}{Normal\ X} \quad \text{.....(2.11)}$$

Pada dasarnya sensor kompas membaca nilai X dan Y sensor terhadap medan magnet sehingga didapatkan nilai heading seperti persamaan diatas. Heading sensor dapat digambarkan sebagai berikut.

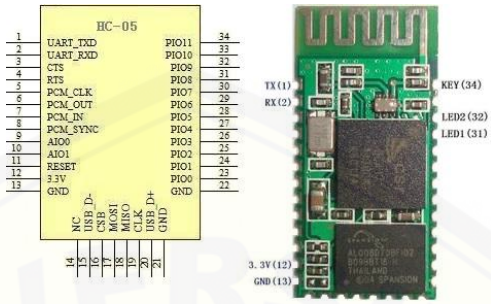



Gambar 2.9 Heading Sensor pada Sensor Kompas HMC5883L (Honeywell, 2013)

## 2.9 Modul *Bluetooth* HC-05

Modul *Bluetooth* HC-05 adalah modul yang dapat diintegrasikan dengan mikrokontrol dan lebih murah dan mudah dalam penggunaanya. Modul ini difungsikan untuk komunikasi nirkabel antar mikrokontroler dengan perangkat lainnya yang sama-sam mempunyai *Bluetooth*. Modul tersebut mempunyai versi Bluettoth V20 + EDR (*Enhanched Data Rate*) Moduulasi sebesar 3 Mbps dan

dilengkapi pengiriman *baseband radio* 2,4 GHz. Pada umumnya modul tersebut bekerja pada *baudrate* 9600 sampai 115200 (iTead, 2015)




Gambar 2.10 Modul *Bluetooth* HC-05 (Sumber : Marchi, 2016)

# BAB 3. METODOLOGI PENELITIAN

## 3.1 Waktu dan Tempat Pembuatan Alat

3.1.1 Waktu

Skripsi *Navigasi Robot Berbasis Path Planning dengan Pemulihan Jalur Otomatis* ini dilakukan selama 6 Bulan, yang di mulai pada bulan Februari sampai dengan Juli 2017 pada tabel 3.1.

Tabel 3.1 Rencana pengerjaan skripsi

| No | Kegiatan | Bulan | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | Studi Kasus | | | | | | |
| 2 | Desain Robot | | | | | | |
| 3 | Pembelian Komponen | | | | | | |
| 4 | Pembuatan dan Observasi pada robot Robot | | | | | | |
| 5 | Pengambilan Data | | | | | | |
| 6 | Penulisan Laporan | | | | | | |

3.1.2 Tempat

Skripsi *Navigasi Robot Berbasis Path Planning dengan dalam Pemulihan Jalur Otomatis* ini dalam pembuatan *hardware* maupun *software* dilakukan di kos dengan alamat Jl. Jawa 6 Gg.4 No. 23 Kel. Sumbersari - Kec. Sumbersari (68121) Jember dan Laboratorium Patrang Kel. Jember Lor – Kec. Patrang Jember.

## 3.2 Tahap Perancangan

Secara garis besar proses perancangan alat dapat dikelompokkan menjadi beberapa tahap, yaitu: Persiapan Desain robot, Persiapan Bahan dan Alat, Persiapan Skema Rangkaian.

### 3.2.1 Persiapan Desain Robot

Desain robot akan dibuat melalui *software* Autodesk Inventor. Desain robot dibuat secara 3 dimensi.

### 3.2.2 Persiapan Bahan dan Alat

Pada tahap ini yang dilakukan adalah mempersiapkan alat dan bahan yang diperlukan sebelum melakukan proses perancangan alat.

a. Bahan

1) Bahan Pembuatan Robot

| | |
|---|---|
| Motor DC | 2 buah |
| Roda | 1 pasang |
| Roda bebas | 1 buah |
| Baterai LiPo 12V | 1 buah |
| IC *Regulator* 7808 | 1 buah |
| MOSFET Kanal-P | 4 buah |
| MOSFET Kanal-N | 4 buah |
| Baut dan mur | Secukupnya |
| Plastik Akrilik | Secukupnya |

3) Bahan Pembuatan Sensor

| | |
|---|---|
| *Rotary Encoder* | 2 buah |
| Sensor Kompas HMC5883L | 1 buah |

4) Bahan *Software*

Microsoft Visual Basic

Proteus 7 Professional

Eagle PCB

Autodesk Inventor

Arduino IDE

5) Bahan pendukung

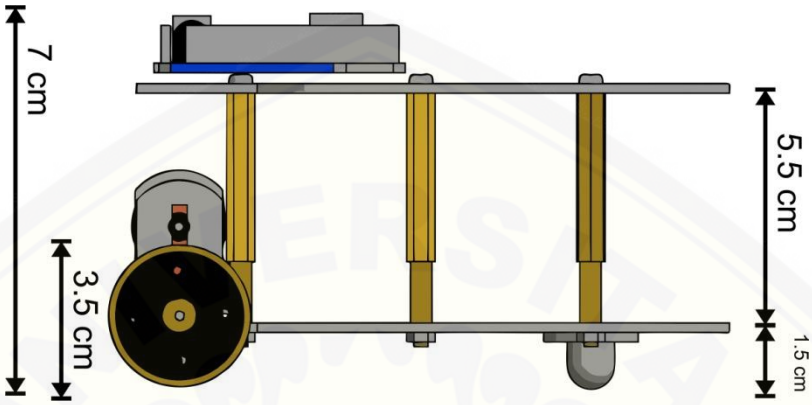
| | |
|---|---|
| Arduino Mega | 1 buah |
| *Bluetooth* HC-05 | 1 buah |
| Saklar | 1 buah |
| Kabel | Secukupnya |
| *Header* | Secukupnya |
| Baut dan mur | Secukupnya |
| Timah | Secukupnya |
| Kabel | Secukupnya |
| PCB | Secukupnya |

b. Alat

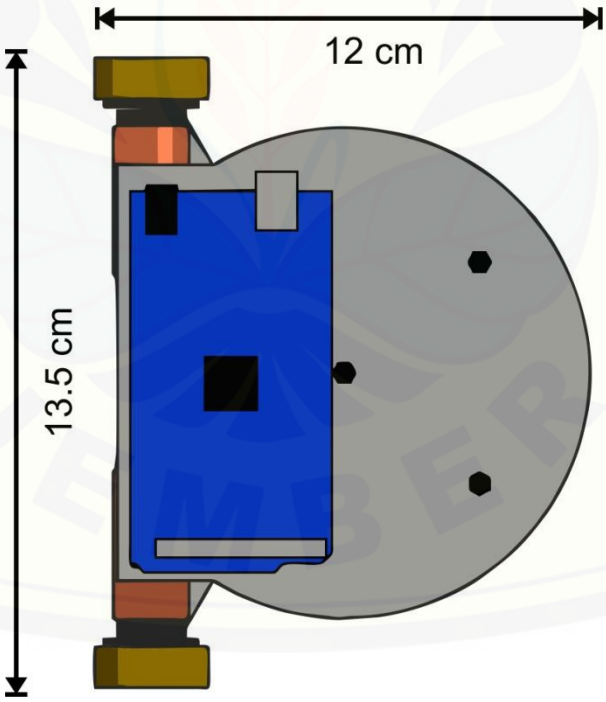
1) Laptop
2) Gergaji
3) Seterika
4) Solder
5) Tang
6) Bor
7) Silet
8) Gunting
9) Obeng

### 3.3 Desain Robot

Pada tahap perancangan desain, robot didesain dengan dimensi pada gambar sebagai berikut.

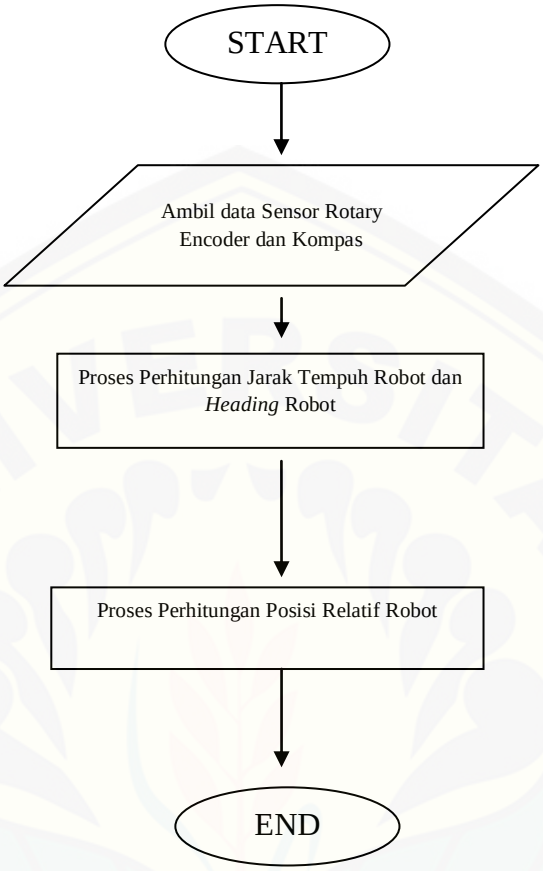



Gambar 3.1 Desain Tampak Samping




Gambar 3.2 Desain Tampak Samping

### 3.4 Parameter *Odometry* dan Kinematik pada *Differential Drive* Robot




Gambar 3.3 *Flowchart Odometry*

Seperti Gambar 3.9 pada dasarnya robot melakukan *odometry* berdasarkan data dari penambahan pulsa dari *rotary encoder* yang didapatkan. Kemudian dilakukan untuk menetukan jarak tempuh robot dengan menggunakan persamaan seperti berikut.

Jarak Tempuh Roda Kiri = (Keliling roda x posA) / PPR……(3.1)

Jarak Tempuh Roda Kanan = (Keliling roda x posB) / PPR…….(3.2)

Jarak Rata – Rata Kedua Roda = (JTRKiri + JTRKanan) / 2…….(3.3)

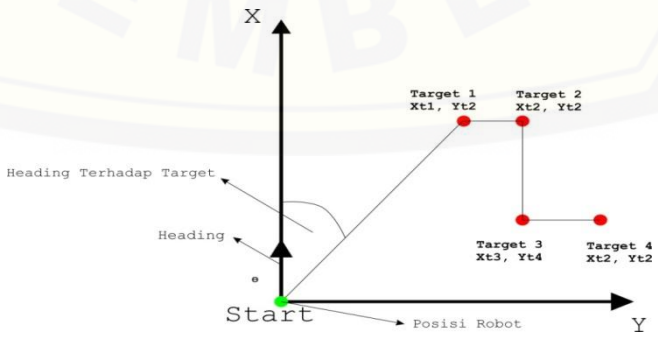
PPR (Pulse Per Rotation) = Jumlah pulsa setiap roda sekali berputar.

Selanjutnya menentukan *heading* robot yan didapatkan dari kombinasi sensor kompas dan *rotary encoder* dan posisi relatif dari robot dari data berupa jarak dan *wheel base* (jarak antar roda robot) robot dengan persamaan seperti berikut.

Posisi X Robot = Jarak Rata – Rata Kedua Roda x cos (tetha)…….…..(3.4)
Posisi Y Robot = Jarak Rata – Rata Kedua Roda x sin (tetha)……...(3.5)
*Heading* = (((JTRKanan - JTRKiri) / *wheel base*) + Sensor) / 2..(3.6)

Setelah didapatkan posisi relatif robot dan *heading* robot, dengan menggunakan PID *regulator* pada kinematik *Differential Drive Robot* menggunakan rumus 2.9 dan 2.10, robot akan bergerak menuju posisi dan otomatis akan memulihkan jalur ketika robot mendapat gangguan yang diberikan melalui persamaan seperti berikut.

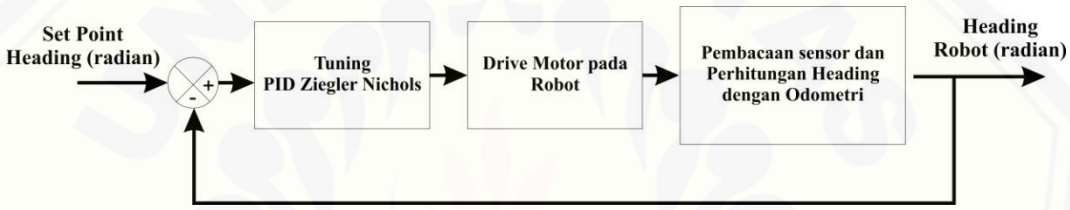
X = X Target - Posisi X Robot…….(3.7)
Y = Y Target - Posisi Y Robot…….(3.8)
*Heading* Terhadap Target = atan2 (X, Y)…….(3.9)
*Error* = *Heading* Terhadap Target (*Set Point) – Heading* (*Point Variable*)..(3.10)
PID = (e) = U……..(3.11)
Roda Kanan = (2 x PWM + U) / Diameter Roda…….(3.12)
Roda Kiri = (2 x PWM – U) / Diameter Roda…….(3.13)




Gambar 3.4 *Path Planning* Robot

### 3.5 Perancangan Diagram Blok Kontrol PID

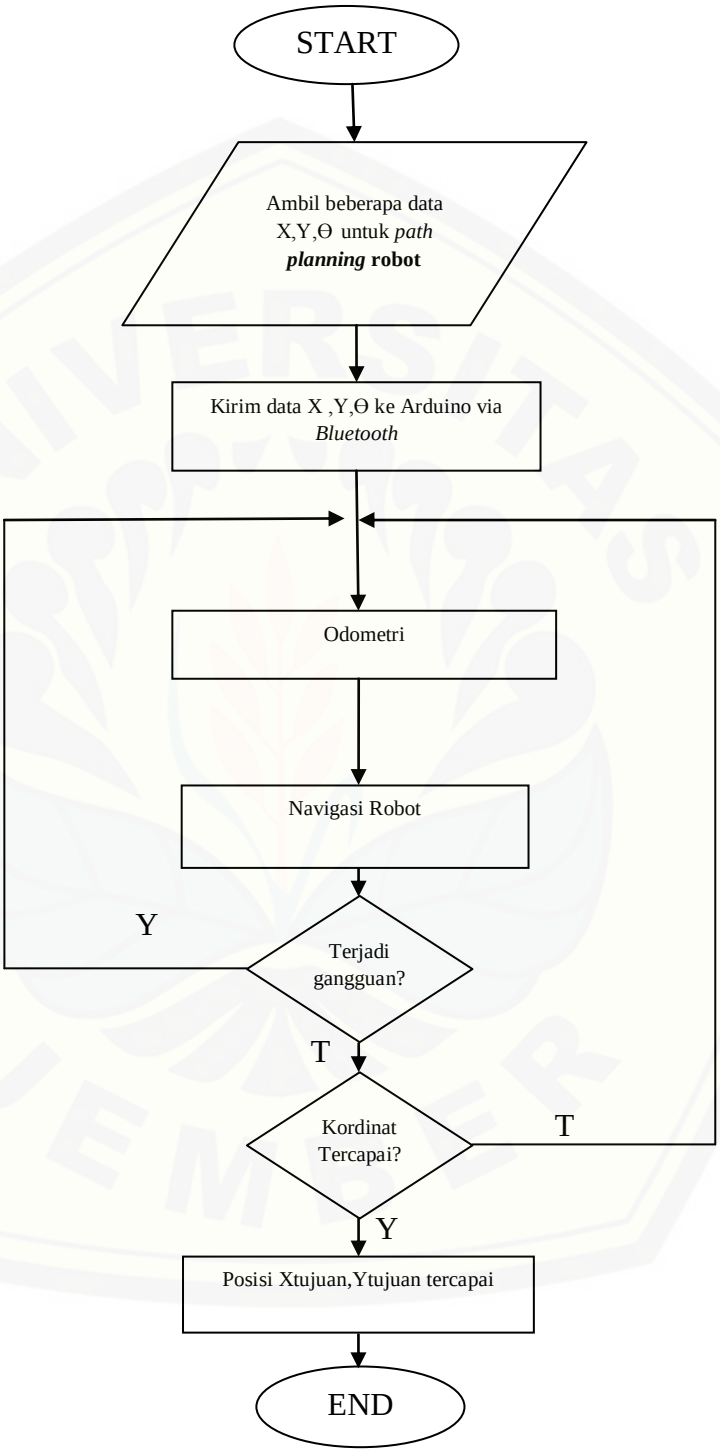
Blok kontrol yang dimaksud pada gambar 3.5 adalah robot mempunya set point berupa sudut (*heading*) terhadap titik yang akan dituju. PID kontroler menggunakan metode *Ziegler Nichols* ini dimaksudkan untuk mengontrol *drive* pada robot. Jadi PID kontroler ini dapat dikatakan menambah atau mengurangi kecepatan motor. Kemudian data yang yang diperoleh setelah *drive* motor pada robot dilakukan perhitungan *heading* robot melalui odometri sehingga didapatkan arah hadap robot dalam satuan radian. Sehingga robot mempunyai sudut yang tepat untuk mencapai kordinat dari tujuan.




Gambar 3.5 Diagram Blok Kontrol Sistem

### 3.6 Pembuatan *Flowchart* Alat

Pada flowchart gambar 3.6 dijelaskan bahwa dilakukan pengambilan beberapa *path* (kordinat) dan arah hadap (Ɵ) sebagai jalur yang akan dilalui sebagai robot itu sendiri dan dikirimkan ke robot melalui *bluetooth*. Setelah pengiriman data selesai dilakukan. Proses perhitungan dan kontrol motor PID mulai dilakukan. Kontrol PID ini dimaksudkan untuk mengontrol putaran motor kanan dan kiri untuk mencapai lokasi yang telah ditentukan melalui timbale balik dari odometri atau kemampuan robot untuk menemukan posisi relatifnya dan dapat memulihkan jalur secara otomatis ketika terjadi tabrakan. Setelah dilakukan perhitungan pada mikro robot mulai menavigasi. Ketika terjadi tabrakan robot akan kembali keposisi perhitungan dan apabila tidak dia akan terus melaju hingga posisi X, Y tercapai. Hal tersebut diulang sehingga robot menempuh *Path* tujuan terakhir.

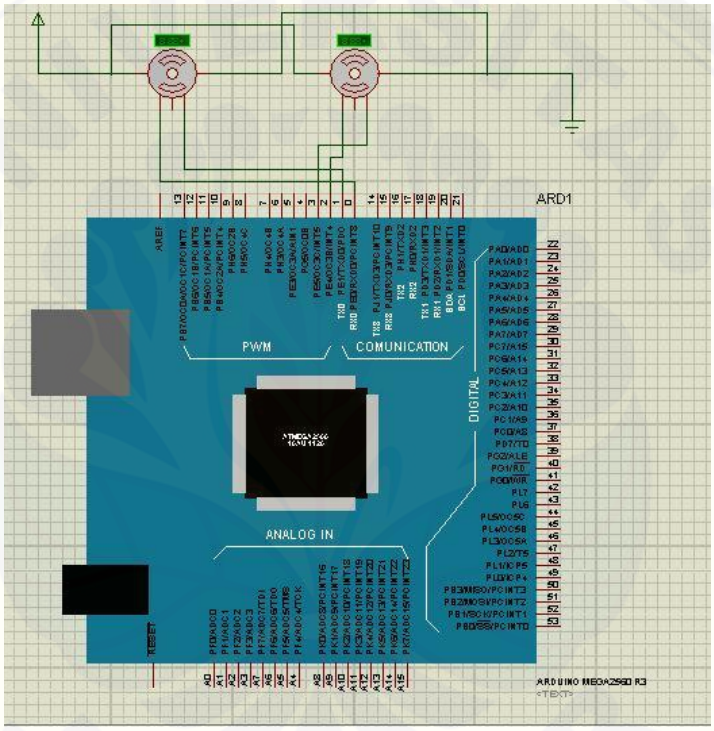



Gambar 3.6 *Flowchart* sistem

### 3.7 Persiapan Skema Rangkaian

Pada tahap ini yang dilakukan sebelum proses pembuatan alat, maka diperlukan skema tiap-tiap rangkaian yang nantinya akan dibuat terpisah dan digambar ulang di laptop. Skema rangkaian yang dipersiapkan yaitu: Skema Rangkaian *Rotary Encoder*, Skema Rangkaian Sensor kompas dan Skema Rangkaian *Bluetooth* HC-05.

#### 3.7.1 Skema Rangkaian *Rotary Encoder*

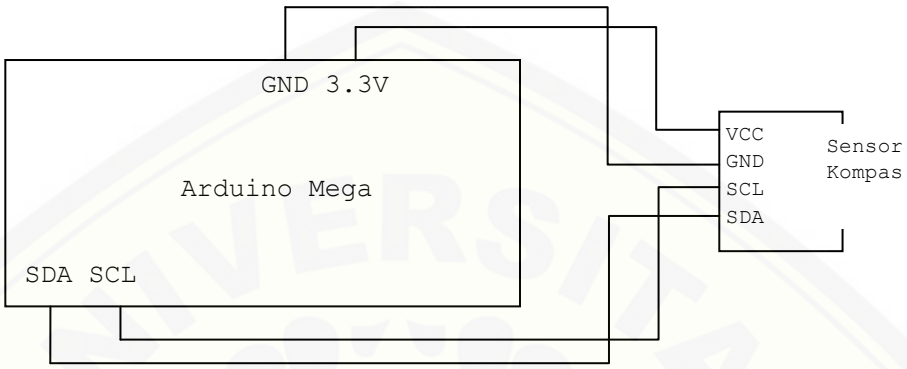



Gambar 3.7 Skema rangkaian *Rotary Encoder*

*Rotary Encoder* yang terpasang dalam arduino ini dihubungkan dengan roda. Ketika roda berputar *rotary encoder* akan mengukur *revolution per minute* (RPM) pada roda robot. *Rotary encoder* kedua terminalnya terpasang pada pin digital Arduino Mega.

3.7.2 Skema Rangkaian Sensor Kompas

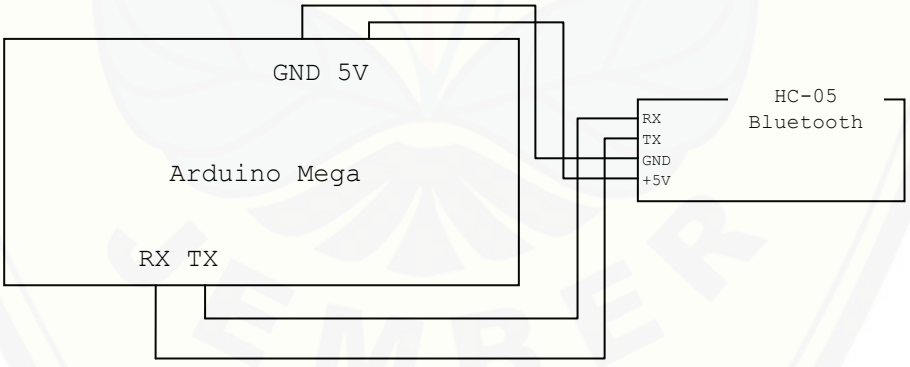
Sensor kompas yang digunakan disambungkan dengan Arduino dengan menggunakan komunikasi I2C. Menggunakan pin D20 (SDA) dan D21 (SCL) pada Arduino Mega. Sensor kompas menggunakan VCC 3,3V.




Gambar 3.8 Rangkaian Sensor KOMPAS pada Arduino

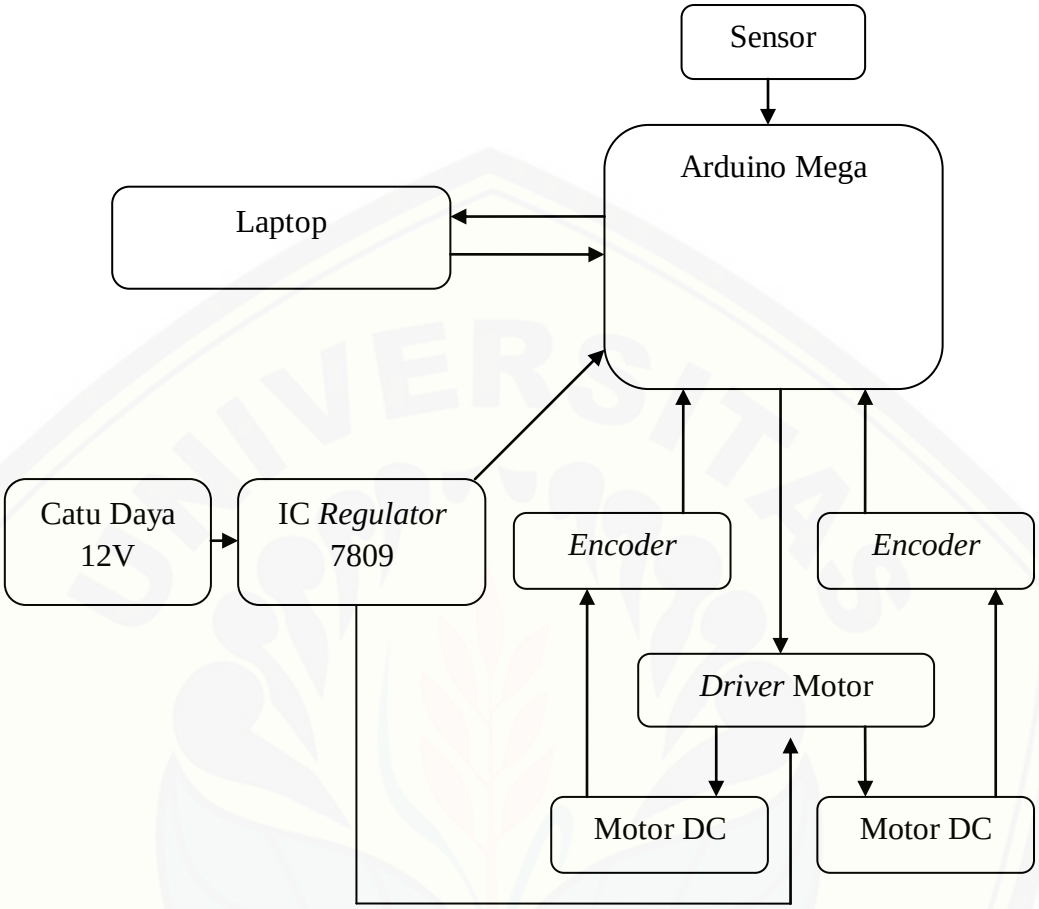
3.7.3 Skema Rangkaian *Bluetooth* HC-05

Rangkaian ini digunakan untuk komunikasi antara PC dengan Arduino melalui *Bluetooth*.




Gambar 3.9 Rangkaian *Bluetooth* HC-05

### 3.8 Perancangan Hardware



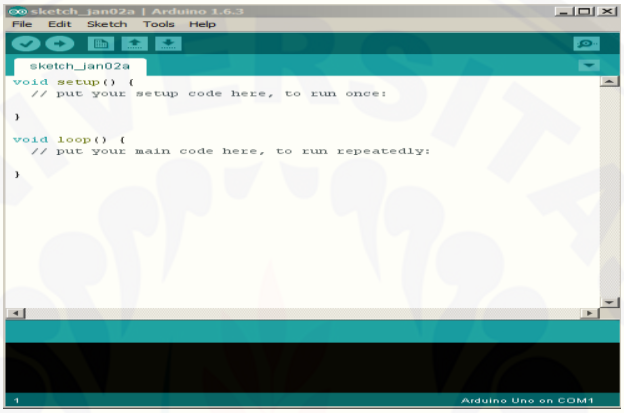

Gambar 3.10 Perancangan *hardware* alat.

Dari blok diagram pada Gambar 3.9 tersebut dijelaskan bahwa Arduino Mega menerima data dari sensor Kompas untuk membaca nilai *turning* dari robot dan mengkompensasi *Error turning* supaya lebih baik, sedangkan data *encoder* yang masuk pada Arduino Mega merupakan hasil *increment* dari resolusi putaran roda robot yang berasal dari penggerak berupa Motor DC. Driver motor mendapatkan catu daya sebesar 12V dari baterai. Arduin Mega juga mendapatkan catu daya dari baterai setelah melewati *Regulator* 7808 untuk diturunkan tegangannya. Laptop memberikan instruksi kordinat kepada Arduino Mega melalui *Bluetooth* dan memantau pergerakan robot.

### 3.9 Perancangan *Software*

3.9.1 Arduino

Program yang dipakai untuk men-*download software* yang telah dirancang menggunakan Arduino IDE. Digunakan program ini berdasarkan pertimbangan bahwa *hardware* sistem mikrokontroler yang digunakan yaitu berbasis Arduino. Berikut contoh tampilan Arduino versi 1.6.3




Gambar 3.11 Arduino 1.6.3

Arduino IDE diciptakan untuk para pemula bahkan yang tidak memiliki basic bahasa pemrograman sama sekali karena menggunakan bahasa C++ yang telah dipermudah melalui *library*. Arduino menggunakan *software processing* yang digunakan untuk menulis program kedalam Arduino. *Processing* sendiri merupakan penggabungan antara bahasa C++ dan Java. *Software* Arduino ini dapat di *install* di berbagai *operating system* (OS) seperti linux, windows. *Software* IDE Arduino terdiri dari 3 bagian :

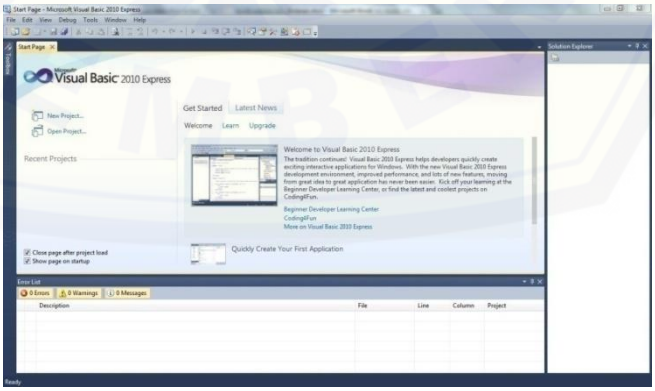
a. *Editing* program, untuk menulis dan mengedit program dalam bahasa *processing*. *Listing* program pada Arduino disebut *sketch*.
b. *Compiler*, modul yang berfungsi mengubah bahasa *processing* (kode program) kedalam kode biner karena kode biner adalah satu-satunya bahasa program yang dipahami oleh mikrokontoler.
c. *Uploader*, modul yang berfungsi memasukkan kode biner kedalam memori mikrokontroler.

Struktur perintah pada Arduino IDE secara garis besar terdiri dari 2 bagian yaitu *void setup* dan *void loop*. *Void setup* berisi perintah yang akan dieksekusi hanya satu kali sejak Arduino dihidupkan, sedangkan *void loop* berisi perintah yang akan dieksekusi berulang-ulang selama Arduino dinyalakan.

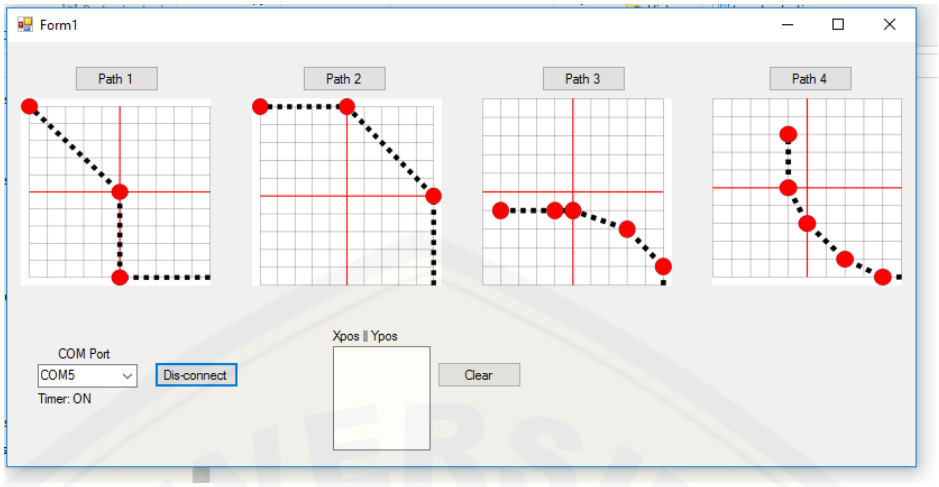
### 3.9.2 *Visual Basic*

*Visual Basic* (VB) adalah RAD (*Rapid Application Development*) *tool*, yang memungkinkan programmer untuk membuat aplikasi Windows dalam waktu yang sangat sedikit. Ini adalah bahasa pemrograman yang paling populer di dunia, dan memiliki programmer lebih dan baris kode dari pada pesaingnya terdekat.

Beberapa bahasa skrip seperti *Visual Basic for Applications* (VBA) dan *Visual Basic Scripting Edition* (VBScript), mirip seperti halnya *Visual Basic*, tetapi cara kerjanya yang berbeda. Para programmer dapat membangun aplikasi dengan menggunakan komponen-komponen yang disediakan oleh Microsoft *Visual Basic* Program-program yang ditulis dengan *Visual Basic* juga dapat menggunakan Windows API, tapi membutuhkan deklarasi fungsi luar tambahan. Dalam pemrograman untuk bisnis, *Visual Basic* memiliki pangsa pasar yang sangat luas. Dalam sebuah survei yang dilakukan pada tahun 2005, 62% pengembang perangkat lunak dilaporkan menggunakan berbagai bentuk *Visual Basic*, yang diikuti oleh C++, *JavaScript*, C#, dan *Java.* (Adi, 2015)




Gambar 3.12 Microsoft Visual Basic 2010




Gambar 3.13 *Software Path Planning* yang telah dibuat

Pada percancangan software menggunakan Microsoft Visual Basic ini, dibuat sebuah bidang dengan piksel 100 x 100 cm yang akan disesuaikan dengan bidang asli pada robot untuk menetukan *path* yang harus dilewati.

## 3.10 Pengujian Alat

Dalam pembuatan diperlukan pengujian yang bertujuan untuk mengetahui kinerja dari sebuah alat yang telah dikerjakan dan dapat menggambarkan respon-respon dari navigasi robot. Pada pengujian alat kali meliputi pengujian per bagian dan pengujian keseluruhan.

### 3.10.1 Pengujian Perbagian

Pada pengujian perbagian meliputi sensor *Rotary Encoder* dan Sensor Kompas HMC5883L serta alat untuk berkomunikasi antara PC dengan robot yaitu Modul *Bluetooth* HC-05.

a. Pengujian *Rotary Encoder*

Pengujian *Rotary Encoder* bertujuan untuk mengetahui respon kedua motor. Diharapkan nilai *increment* pada *encoder* menunjukan hal yang sama antara kedua rodanya sehingga motor dc menjadi layak digunakan dan mempunyai kecepatan yang sama. Pada pengujian ini robot akan

dijalankan lurus searah sumbu X dengan beberapa ketentuan jarak yang berbeda-beda. Robot diuji dengan menempuh jarak 20cm, 40cm , 60cm , 80cm dan 100cm. Data yang dapat diambil dari pengujian tersebut berupa jumlah penjambahan pulsa antara *encoder* kanan dan kiri serta jarak yang telah ditempuh robot.

b. Pengujian Sensor Kompas HMC5883L

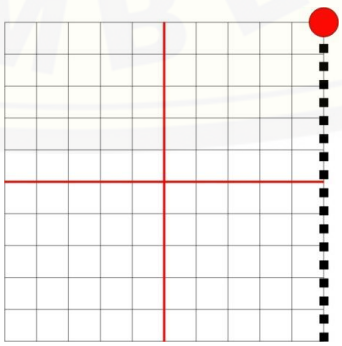
Pengujian pada sensor kompas HMC5883L ini bertujuan untuk mencari tahu apakah sensor tesebut layak digunakan. Pengujian dilakukan dengan menyesuaikan arah hadap robot dengan busur $360^o$. Robot diuji pada sudut $0^o$, $90^o$ dan $180^o$. Pembacaan tersebut dari derajat dikonversikan dalam bentuk radian.

### 3.10.2 Pengujian Keseluruhan

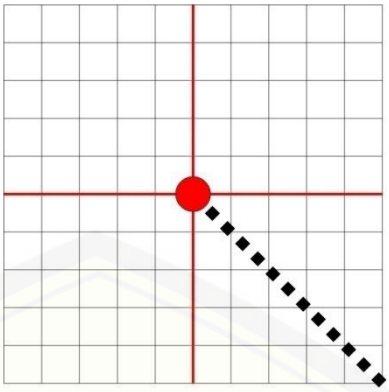
Pada pengujian keseluruhan ini meliputi beberapa *path* (kordinat) yang telah direncanakan sebagai trayektori robotdengan PID Ziegler Nichols dan tanpa PID (hanya dengan Kp = 5 dan Kp =10). Berikut beberapa pengujian yang akan dilakukan dengan mneggunakan 1 *Path*, 2 *Path* dan *Path Planning Robot.*

a. Pengujian dengan menggunakan 1 *Path*

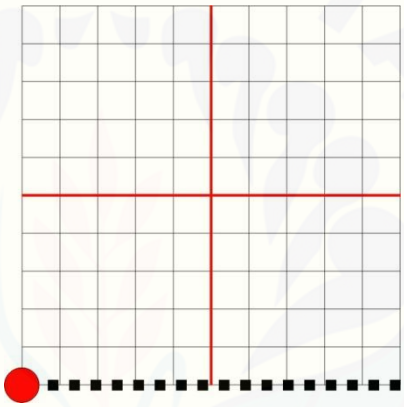
Pada pengujian dengan menggunakan 1 *Path* ini, penguijian dilakukan dengan memberikan *Path* dengan titik (100, 0) untuk pengujian pertama. Pengujian kedua dengan memberikan titk (50, 50) dan pengujian ketiga (0,100).



Gambar 3.14 Pengujian dengan *Path* (100, 0)

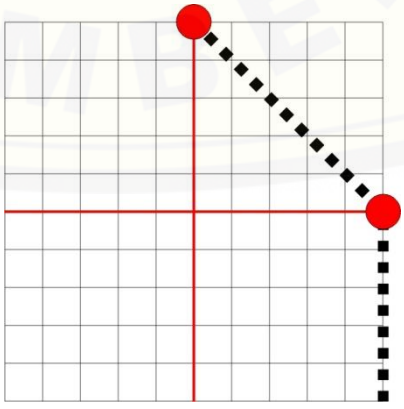


Gambar 3.15 Pengujian dengan *Path* (50, 50)

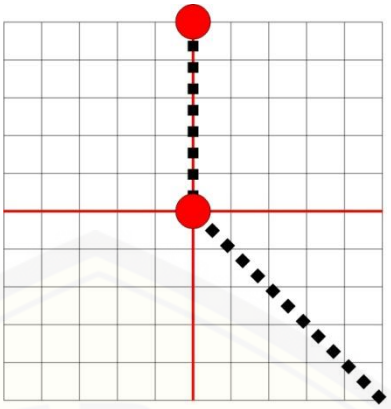


Gambar 3.16 Pengujian dengan *Path* (0, 100)

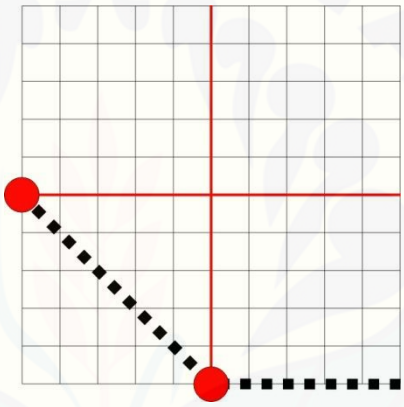
b. Pengujian dengan menggunakan 2 *Path*



Gambar 3.17 Pengujian dengan *Path* (50, 0) dan (100, 50)



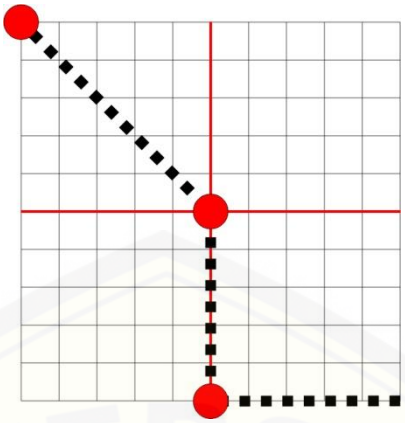
Gambar 3.18 Pengujian dengan *Path* (50, 50) dan (100, 50)



Gambar 3.19 Pengujian dengan *Path* (0, 50) dan (50, 100)

Pengujian dengan menggunakan 2 *Path* ini dengan menambahkan 1 *Path* lagi dari pengujian pertama. Pada pengujian pertama diberikan *Path* dengan titik (50, 0) dan (100, 50). Pengujian kedua diberikan *Path* pada titik (50, 50) dan (100, 50) dan pengujian ketiga dengan memberikan Path pada titik (0, 50) dan (50, 100).

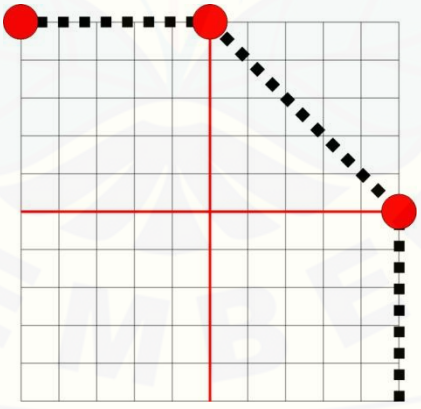
c. *Path Planning* 1



Gambar 3.20 *Path Planning* 1 Robot

Pada *Path Planning* 1 robot robot diberikan kordinat titik (0cm, 0cm) menuju ke titik (0cm, 50cm) selanjutnya ke titik (50cm ,50cm) dan yang terakhir (100cm ,100cm). Dari masukan tersebut data yang diambil dari robot berupa trayektori berdasarkan kordinat robot dan trayektori pada arena robot.

d. *Path Planning* 2

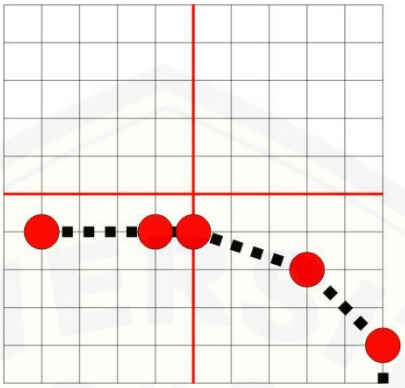


Gambar 3.21 *Path Planning* 2 Robot

Pada *Path Planning* 2 robot berawal dari titik 0 cm, 0 cm menuju ke titik (50cm, 0cm) selanjutnya ke titik (100cm ,50cm) dan yang terakhir (100cm ,100cm). Dari masukan tersebut data yang diambil dari robot

berupa trayektori berdasarkan kordinat robot dan trayektori pada arena robot.

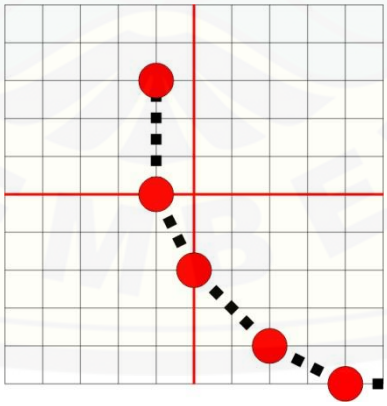
e. *Path Planning* 3



Gambar 3.22 *Path Planning* 3 Robot

Pada *Path Planning* 3 robot berawal dari titik (0cm, 0cm) menuju ke titik (10cm, 0cm) selanjutnya ke titik (30cm ,20 cm) dilanjutkan ke titik (40cm ,50cm) kemudian menuju (50cm, 60cm) dan path yang terakhir (50cm, 90cm). Dari masukan tersebut data yang diambil dari robot berupa trayektori berdasarkan kordinat robot dan trayektori pada arena robot.

f. *Path Planning* 4



Gambar 3.23 *Path Planning* 4 Robot

Pada *Path Planning* 4 robot berawal dari titik (0cm, 0cm) menuju ke titik (0cm, 10cm) selanjutnya ke titik (10cm ,30cm) dilanjutkan ke titik (30cm ,50cm) kemudian (50cm, 60cm) dan yang terakhir (80cm, 60cm).

Dari masukan tersebut data yang diambil dari robot berupa trayektori berdasarkan kordinat robot dan trayektori pada arena robot.

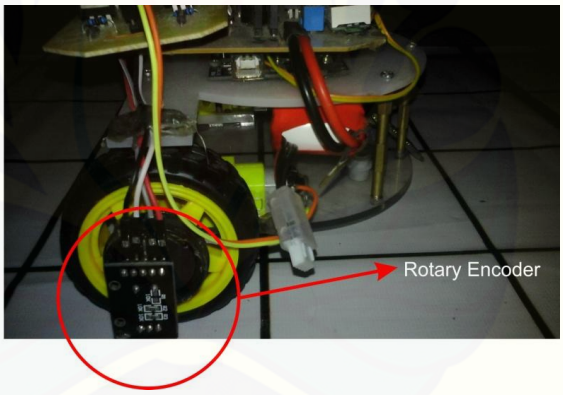
g. Pengujian Pemulihan Jalur

Pada pemulihan jalur robot dilakukan pengujian dengan memberikan gangguan pada roda robot setelah diberikan perintah untuk menempuh salah suatu *path planning*.

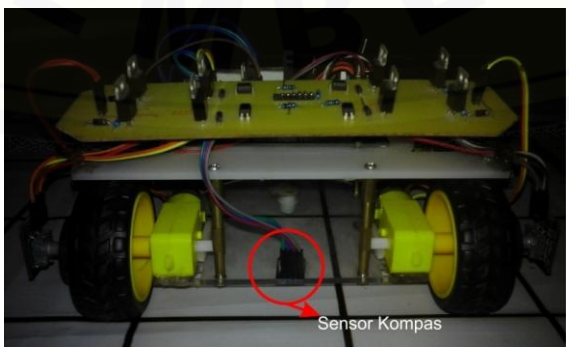
h. Pengujian dengan Kp = 5 dan Kp =10

Pda pengujian ini robot akan diberikan control berupa Kp =5 dan Kp = 10 tanpa danya PID Ziegler-Nichols dengan trayektori pengujianm sama dengan menggunakan PID Ziegler-Nichols.

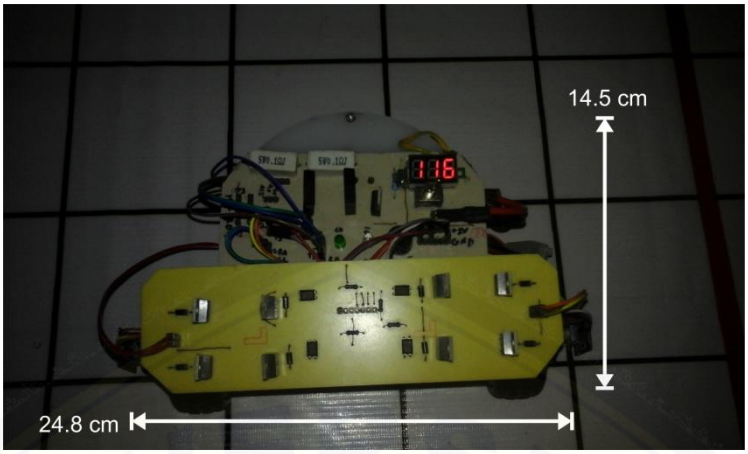
## 3.11 Hasil Perancangan Alat




Gambar 3.24 Peletakan Sensor *Rotary Encoder*

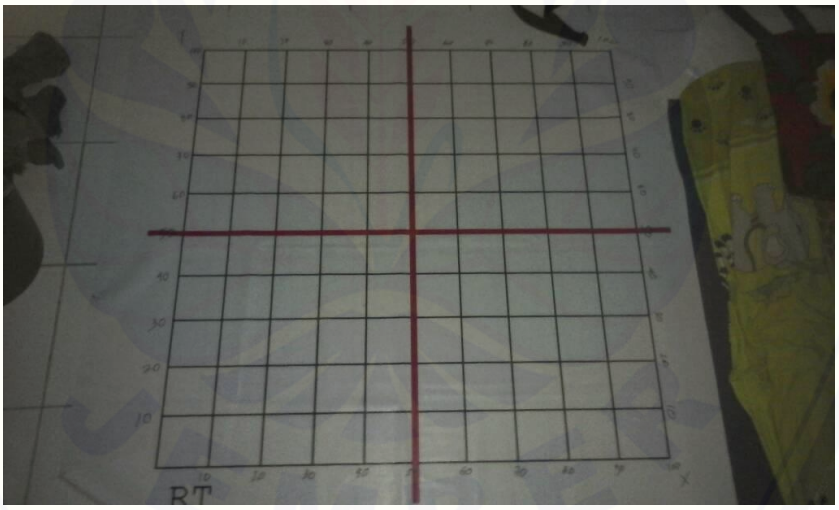



Gambar 3.25 Peletakan Sensor Kompas HMC5883L




Gambar 3.26 Robot tampak atas

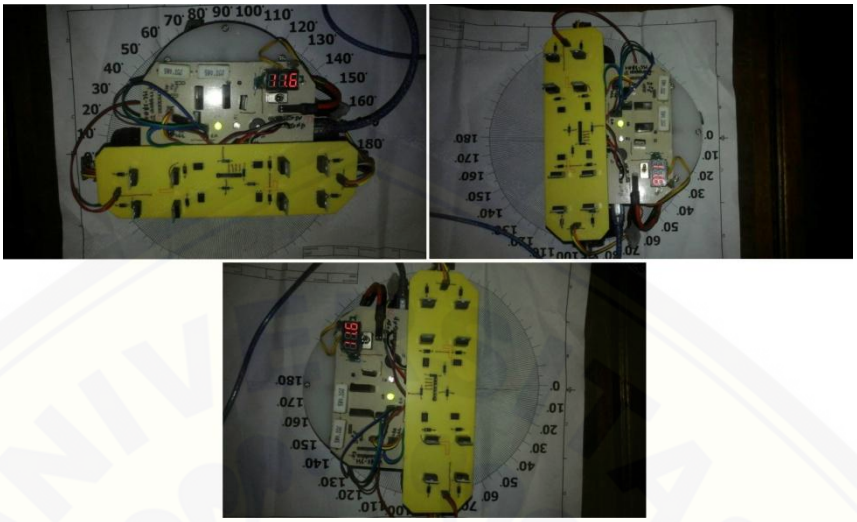
Sedangakan untuk arena yang telah dibuat dengan skala 100cm x 100cm dengan setiap kotak mempunyai ukuraan 10cm x 10cm. Berikut bentuk arena yang telah dibuat.



Gambar 3.27 Bentuk arena robot

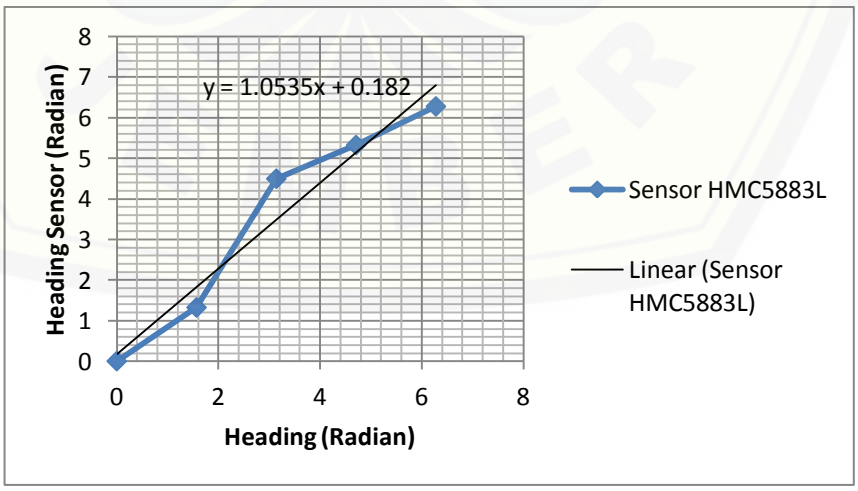
Perancangan dari robot dapat diketahui melalui Gambar 3.24 dimana peletakan *rotary encoder* dipasang secara seri dengan roda. Sedangakan untuk Sensor Kompas pada Gambar 3.25 diletakan pada tengah antara kedua aktuaror dalam hal ini berupa motor dc. Perbedaan dengan desain terjadi karena adanya pemasangan *rotary encoder* secara seri sehingga robot lebih memanjang.

### 3.12 Proses Kalibrasi Sensor Kompas HMC5883L



Gambar 3.28 Proses kalibrasi pada sensor kompas HMC5883L terhadap busur 360$^{o}$

Dari kalibrasi tersebut dengan mengkonversi nilai derajat yang didapatkan sensor menjadi radian sehingga dapat dihitung oleh mikrokontroler, disni mikrokontroler. Nilai 0$^{o}$ pada busur berarti 0, 90$^{o}$ menjadi 1.57, 180$^{o}$ menjadi 3.14, 270$^{o}$ menjadi 4.71 dan 360$^{o}$ menjadi 6.28.




Gambar 3.29 Grafik pengujian sensor kompas HMC5883L terhadap sudut pengukuran dalam satuan radian.

# BAB 5. PENUTUP

## 5.1 Kesimpulan

Dari penilitian yang saya lakukan mengenai Navigasi Robot berbasis *Path Planning* dengan Pemulihan Jalur Otomatis maka dapat diambil kesimpulan:

1. *Rotary Encoder* mempunyai nilai yang dapat dikatakan sama ketika menempuh jarak yang telah ditentukan dengan rata-rata kesalahan hanya 0,603 % membuat dapat menempuh titik (97, 0) ketika mendapat input (100, 0) dengan selisih sekitar 3 cm.
2. Robot dapat melakukan *Path Planning* dengan *Error* paling besar terletak pada *Path Planning* 4 ketika robot menuju titik (10, 30), pada sumbu Y dengan besar rata-rata 80% dibanding X dengan besar 33,333%.
3. Robot dapat memulihkan jalurnya secara otomatis dengan adanya PID Ziegler-Nichols meskipun terdapat osilasi sudut terhadap titik tujuan pada gangguan 10 cm dan 15 cm.
4. *Path Planning* 4 dengan menggunakan Kp membuat pergerakan robot melengkung akan tetapi tetap dapat menempuh titik yang telah dimasukan dengan *error* paling besar pada *Path Planning* 4 dengan 100% kesalahan pada sisi X dan 33,3333% dengan Kp = 10.
5. Pada pengujian robot dengan menggunakan Kp = 5 dengan *Input Path Planning* 4 Robot tidak dapat menempuh titik tujuan dengan *error* pada kedua sumbunya sebesar 75% dan Robot terus bergerak kesamping dan berosilasi ke arah kiri atau sumbu Y.

## 5.2 Saran

Dari peneletian yang dilakukan mengenai Navigasi Robot berbasis *Path Planning* dengan Pemulihan Jalur Otomatis penulis memberikan beberapa saran.

1. Menggunakan aktuator yang memiliki tingkat kepresisian yang lebih tinggi atau aktuator yang didalamnya sudah memiliki kontrol yang baik.

2. Menambahkan jumlah roda pada robot dalam hal ini seperti *Omni-Wheel Drive* dan sebagainya.
3. Menggunakan metode kecerdasan buatan yang lebih baik lagi sehingga robot dalam berjalan dapat melakukan pembelajaran sendiri.
4. Menggunakan sensor dengan kepresisian yang tinggi.
5. Mempresisikan pemasangan roda dan sensor pada robot.

# DAFTAR PUSTAKA

Kompas Sensor, http://id.wikipedia.org

Hendriono, Dede. Artikel *" Mengenal Arduino Mega2560"*. 2014.

Sigit, Riyanto. 2007. "*Robotika, Sensor dan Aktuator"*. Yogyakarta: Graha Ilmu, Vol.63.

Jusuf, 2011.*"Navigasi Mobile Robot Berbasis Trajektori dan Odometry dengan Pemulihan Jalur Otomatis*." Jurnal Teknik Elektronika, Politeknik Eletronika Negeri Surabaya.

Bayu, 2011.*"Path Tracking Mobile Robot dengan Umpan Balik Odometri*." Jurnal Teknik Elektronika, Politeknik Eletronika Negeri Surabaya.

Ogata, Katsuhiko,*"Modern Control Engineering Fourth Edition",*2002

Setiawan Iwan, *" Perancangan dan Implementasi Sistem Kontrol Navigasi Robot Mobile Penjejak Trayektori Bezier "*, Paper, Laboratorium Teknik Kontrol Otomatis, Teknik Elektro Undip

Arras, dkk. 2011.*"Robot Motion Planniing."* Freiburg: Freiburg University.

Mint Cherry dan Win Nu Nu , dkk. 2016*." Position and Velocity control for Two-Wheel Differential Drive Mobile Robot."* Mandalay: *International Journal of Science, Engineering and Technology Research* (IJSETR) Volume 5, Issue 9, September 2016

LaValle, Steve M, 2006., *" Planning Algorithm "*, Cambrige University Press, Cambridge University.

Egerstedt, Morga, 2011., *" Control of Mobile Robot "*, https:// www.youtube.com/channel/UCsGy0masgsIbWaN0po6uYTQ, Georgia Institute of Technology.

Anderson David, " IMU Odometry ", Article, October 2006 Dallas, Texas, 2006.

**LAMPIRAN**

A Lampiran Tabel 1 Respon PID dengan Kcr 150

| Tetha | Set Point | Waktu |
|---|---|---|
| 0 | 0 | 0 |
| -0.3 | 0 | 3247 |
| 0 | 0 | 3273 |
| 0.5 | 0 | 3299 |
| 0 | 0 | 3325 |
| -0.11 | 0 | 3351 |
| 0 | 0 | 3377 |
| 0.12 | 0 | 3403 |
| 0 | 0 | 3429 |
| -0.13 | 0 | 3455 |
| 0 | 0 | 3481 |
| 0.13 | 0 | 3507 |
| 0 | 0 | 3533 |
| -0.13 | 0 | 3559 |
| 0 | 0 | 3585 |
| -0.2 | 0 | 3611 |
| 0 | 0 | 3637 |
| 0.12 | 0 | 3663 |
| 0 | 0 | 3689 |
| -0.13 | 0 | 3715 |
| 0 | 0 | 3741 |
| 0.13 | 0 | 3767 |
| 0 | 0 | 3793 |
| -0.13 | 0 | 3819 |
| 0 | 0 | 3845 |
| 0.15 | 0 | 3871 |
| 0 | 0 | 3897 |
| 0.1 | 0 | 3923 |
| -0.11 | 0 | 3949 |
| 0 | 0 | 3975 |
| 0.3 | 0 | 4001 |
| 0 | 0 | 4027 |
| -0.13 | 0 | 4053 |
| -0.11 | 0 | 4079 |
| -0.11 | 0 | 4105 |
| 0 | 0 | 4131 |
| 0.12 | 0 | 4157 |
| 0.12 | 0 | 4183 |
| 0 | 0 | 4209 |
| -0.13 | 0 | 4235 |
| 0 | 0 | 4261 |
| 0.13 | 0 | 4287 |
| 0 | 0 | 4313 |
| -0.13 | 0 | 4339 |
| 0 | 0 | 4365 |
| 0.15 | 0 | 4391 |
| 0 | 0 | 4417 |
| 0 | 0 | 4443 |
| -0.11 | 0 | 4469 |
| 0 | 0 | 4495 |
| 0.12 | 0 | 4521 |
| 0 | 0 | 4547 |
| -0.13 | 0 | 4573 |
| -0.11 | 0 | 4599 |
| -0.09 | 0 | 4625 |
| 0 | 0 | 4651 |
| 0.12 | 0 | 4677 |

B. Lampiran *Lisitng* Program Robot

```
#include <Encoder.h>
#include <math.h>
#include <Wire.h>
#include <HMC5883L.h>

HMC5883L compass;

Encoder enco1(2, 3);
Encoder enco2(18, 19);


//PID Regulator
float Kp = 90;
float Ki = 1.31386861313868;
float Kd = 1541.25;
float error2  = 0;
float error_P = 0;
float error_I = 0;
float error_D = 0;
float P = 0;
float I = 0;
float D = 0;
float U = 0;
```

```
//Contant Definition
const float phi = 3.141593;
const float wb = 19.2;
const float ppr = 80;
const float r = 3.25;
char command = 0;

//Odometri
long posA = 0;
long posB = 0;
float d_kanan = 0;
float d_kiri = 0;
float jarak = 0;
float tetha = 0;
float bearing = 0;
float Xs = 0;
float Ys = 0;
float tetha_mix;

//Goal Reckoning
float X = 0;
float Y = 0;
float error = 0;
float Td = 0;
float Xt;
```

```
float Yt;


//Motor Drive Go to Goal
float maxspd = 255;
float spd_up = 70;
int pinmot_maka = 4;
int pinmot_muka = 5;
int pinmot_maki = 6;
int pinmot_muki = 7;
float pwm = 180;
float mka = 0;
float mki = 0;
float range = 3;
float q = 1;
float dt = 1 / 10;
float l = 0;


//Buzzer & LED
int led1 = 9;
int led2 = 10;
int buzzer = 11;


//Compass Sensor
float tetha_sensor;
const float declinationAngle = -2.61;
```

```
void setup() {
  // put your setup code here, to run once:
  int_compass();
  Serial.begin(115200);
  pinMode(pinmot_maka, OUTPUT);
  pinMode(pinmot_muka, OUTPUT);
  pinMode(pinmot_maki, OUTPUT);
  pinMode(pinmot_muki, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  digitalWrite(led1, HIGH);



}

long oldPositionA  = -999;
long oldPositionB  = -999;

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available() > 0) // Send data only when you receive data:
  {
    command = Serial.read();
```

```
    if (command == '1') {
      for (;;) {
        execute_path_1();
      }
    }
    if (command == '2') {
      for (;;) {
        execute_path_2();
      }
    }
    if (command == '3') {
      for (;;) {
        execute_path_3();
      }
    }
    if (command == '4') {
      for (;;) {
        execute_path_4();
      }
    }
  }
}

void execute_path_1() {
```

```
while (q == 1) {
  Xt = 50; Yt = 0;
  q = 2;
}

while (q == 3) {
  Xt = 100; Yt = 50;
  q = 4;
}

while (q == 5) {
  Xt = 100; Yt = 100;
  q = 6;
}
while ((Xs > (Xt + range)) || (Xs < (Xt - range)) || (Ys > (Yt + range)) || (Ys < (Yt
- range))) {

  posA = enco1.read();
  posB = enco2.read();

  go_to_goal();
  absolute_V();
  execute_motor();
  delay (dt * 1000);
  Serial.print(Xs);
```

```
  Serial.print("||");
  Serial.println(Ys);
 }

 berhenti();
 delay (2000);
 Serial.println("GET POSITION");
 q++;
}

void execute_path_2() {

 while (q == 1) {
  Xt = 0; Yt = 50;
  q = 2;
 }

 while (q == 3) {
  Xt = 50; Yt = 50;
  q = 4;
 }

 while (q == 5) {
  Xt = 100; Yt = 100;
  q = 6;
```

```
  }
  while ((Xs > (Xt + range)) || (Xs < (Xt - range)) || (Ys > (Yt + range)) || (Ys < (Yt
- range))) {

    posA = enco1.read();
    posB = enco2.read();

    go_to_goal();
    absolute_V();
    execute_motor();
    delay (dt * 1000);
    Serial.print(Xs);
    Serial.print("||");
    Serial.println(Ys);
  }

  berhenti();
  delay (2000);
  Serial.println("GET POSITION");
  q++;
}

void execute_path_3() {

  while (q == 1) {
```

```
  Xt = 10; Yt = 0;
  q = 2;
 }

 while (q == 3) {
  Xt = 30; Yt = 20;
  q = 4;
 }

 while (q == 5) {
  Xt = 40; Yt = 50;
  q = 6;
 }
 while (q == 7) {
  Xt = 40; Yt = 60;
  q = 8;
 }
 while (q == 9) {
  Xt = 40; Yt = 90;
  q = 10;
 }
 while ((Xs > (Xt + range)) || (Xs < (Xt - range)) || (Ys > (Yt + range)) || (Ys < (Yt
- range))) {

  posA = enco1.read();
```

```
  posB = enco2.read();

  go_to_goal();
  absolute_V();
  execute_motor();
  delay (dt * 1000);
  Serial.print(Xs);
  Serial.print("||");
  Serial.println(Ys);
 }

 berhenti();
 delay (2000);
 Serial.println("GET POSITION");
 q++;
}

void execute_path_4() {

 while (q == 1) {
  Xt = 0; Yt = 10;
  q = 2;
 }

 while (q == 3) {
```

```
  Xt = 10; Yt = 30;
  q = 4;
 }

 while (q == 5) {
  Xt = 30; Yt = 40;
  q = 6;
 }
 while (q == 7) {
  Xt = 50; Yt = 60;
  q = 8;
 }
 while (q == 9) {
  Xt = 80; Yt = 60;
  q = 10;
 }
 while ((Xs > (Xt + range)) || (Xs < (Xt - range)) || (Ys > (Yt + range)) || (Ys < (Yt
- range))) {

  posA = enco1.read();
  posB = enco2.read();

  go_to_goal();
  absolute_V();
  execute_motor();
```

```
  delay (dt * 1000);
  Serial.print(Xs);
  Serial.print("||");
  Serial.println(Ys);
 }

 berhenti();
 delay (2000);
 Serial.println("GET POSITION");
 q++;
}

void execute_go_to_goal() {

 Xt = 0; Yt = 100;

 for (l = 0; l < 1000; l++) {

  posA = enco1.read();
  posB = enco2.read();

  go_to_goal();
  absolute_V();
  execute_motor();
  delay(dt * 1000);
```

```
    Serial.print(Xs);
    Serial.print("#");
    Serial.print(Ys);
    Serial.print("#");
    Serial.print(tetha);
    Serial.print("#");
    Serial.print(bearing);
    Serial.print("#");
    Serial.println(error);

    while ((Xs < (Xt + range)) && (Xs > (Xt - range)) && (Ys < (Yt + range)) &&
(Ys > (Yt - range))) {

      berhenti();
      delay (2000);
    }
  }
  berhenti();
  delay (1500);
}

void odometri() {

  Vector norm = compass.readNormalize();
  d_kiri = ((2 * phi * r) * posA) / ppr;
```

```
  d_kanan = ((2 * phi * r) * posB) / ppr;
  jarak = (d_kanan + d_kiri) / 2;
  Xs = jarak * cos (tetha_mix);
  Ys = jarak * sin (tetha_mix);
  tetha_sensor = atan2(norm.YAxis, norm.XAxis);
  tetha_sensor += declinationAngle;
  tetha = (d_kanan - d_kiri) / wb;
  tetha_mix = (tetha_sensor + tetha) / 2;
  tetha_mix = atan2(sin(tetha_mix), cos(tetha_mix));
}

void go_to_goal() {
  odometri();
  X = Xt - Xs;
  Y = Yt - Ys;
  bearing = atan2(Y, X);
  error = atan2(sin(bearing - tetha_mix), cos(bearing - tetha_mix));
  Td = sqrt(pow((float)X, 2) + pow((float)Y, 2));

  error_P = error ;
  P = Kp * error_P ;
  error_I = error + error_I;
  I = Ki * error_I ;
  error_D = error - error2;
  D = Kd * error_D ;
```

```
  error2 = error ;
  U = P + I + D;
  mka = ((2 * pwm + U * wb) / (2 * r)) * 1.3;
  mki = ((2 * pwm - U * wb) / (2 * r)) * 1.3;

}

void execute_motor() {

  if ((mki > 0) && (mka > 0))  {
    maju();
  }
  else if ((mki < 0) && (mka < 0)) {
    mki = 0 - mki + spd_up;
    mka = 0 - mka + spd_up;
    mundur();
  }
  else if ((mki > 0) && (mka < 0)) {
    mka = 0 - mka + spd_up;
    kanan();
  }
  else if ((mki < 0) && (mka > 0)) {
    mki = 0 - mki + spd_up;
    kiri();
  }
```

```
}

void absolute_V() {

  if (mki > maxspd) {
    mki = maxspd;
  }
  else if (mki < -maxspd) {
    mki = -maxspd;
  }
  if (mka > maxspd) {
    mka = maxspd;
  }
  else if (mka < -maxspd) {
    mka = -maxspd;
  }
}

void berhenti() {

  digitalWrite (pinmot_maka, LOW);
  digitalWrite (pinmot_muka, LOW);
  digitalWrite (pinmot_maki, LOW);
  digitalWrite (pinmot_muki, LOW);
}
```

```
void maju() {

  //mki  = map(mki , 0,100,0,255);
  //mka = map(mka, 0,100,0,255);

  analogWrite  (pinmot_maka, mka);
  digitalWrite (pinmot_muka, LOW);
  analogWrite  (pinmot_maki, mki);
  digitalWrite (pinmot_muki, LOW);
}

void mundur() {
  // mki  = map(mki , 0,100,0,255);
  //mka = map(mka, 0,100,0,255);

  digitalWrite (pinmot_maka, LOW);
  analogWrite  (pinmot_muka, mka);
  digitalWrite (pinmot_maki, LOW);
  analogWrite  (pinmot_muki, mki);
}

void kanan() {

  // mki  = map(mki , 0,100,0,255);
```

```
  // mka = map(mka, 0,100,0,255);

  digitalWrite (pinmot_maka, LOW);
  digitalWrite (pinmot_muka, mka);
  analogWrite  (pinmot_maki, mki);
  digitalWrite (pinmot_muki, LOW);
}

void kiri() {

  // mki  = map(mki , 0,100,0,255);
  // mka = map(mka, 0,100,0,255);

  analogWrite  (pinmot_maka, mka);
  digitalWrite (pinmot_muka, LOW);
  digitalWrite (pinmot_maki, LOW);
  analogWrite  (pinmot_muki, mki);
}

void int_compass() {

  while (!compass.begin())
    compass.setRange(HMC5883L_RANGE_1_3GA);
  compass.setMeasurementMode(HMC5883L_CONTINOUS);
  compass.setDataRate(HMC5883L_DATARATE_30HZ);
```

```
  compass.setSamples(HMC5883L_SAMPLES_8);
  compass.setOffset(0, 0);
}

void data() {
  posA = enco1.read();
  posB = enco2.read();
  odometri();
  if (jarak <= 80) {
    analogWrite (pinmot_maka, 180);
    analogWrite (pinmot_muka, 0);
    analogWrite (pinmot_maki, 180);
    analogWrite (pinmot_muki, 0);
  }
  else {
    berhenti();
  }
  Serial.print(posA);
  Serial.print("#");
  Serial.print(posB);
  Serial.print("#");
  Serial.println(jarak);
}
```