Institut
Teknologi
Sepuluh Nopember

**TUGAS AKHIR – TE 141599**

# IMPLEMENTASI METODE *WALL FOLLOWING* BERBASIS KINECT UNTUK ROBOT *MOBILE*

Novem Ardan Rohmadin
NRP 2211100051

Dosen Pembimbing
Ronny Mardiyanto, ST., MT., Ph.D.
Rudy Dikairono, ST., MT.

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


**FINAL PROJECT – TE 141599**

# IMPLEMENTATION OF WALL FOLLOWING METHOD BASED ON KINECT FOR MOBILE ROBOT

Novem Ardan Rohmadin
NRP 2211100051

Advisor
Ronny Mardiyanto, ST., MT., Ph.D.
Rudy Dikairono, ST., MT.

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

# IMPLEMENTASI METODE *WALL FOLLOWING* BERBASIS KINECT UNTUK ROBOT *MOBILE*

## 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, |
|---|---|
| Ronny Mardiyanto, ST., MT., Ph.D. | Rudy Dikairono, ST., MT. |
| NIP : 198101182003121003 | NIP : 198103252005011002 |

SURABAYA
JULI, 2015

# IMPLEMENTASI METODE *WALL FOLLOWING* BERBASIS KINECT UNTUK ROBOT *MOBILE*

**Nama : Novem Ardan Rohmadin**
**Pembimbing I : Ronny Mardiyanto, ST., MT., Ph.D.**
**Pembimbing II : Rudy Dikairono, ST., MT.**

## ABSTRAK

Navigasi robot *mobile* merupakan cara dalam mengatur gerakan robot *mobile*. Pada robot dengan tugas khusus seperti robot pencari benda berbahaya yang digerakkan otomatis di dalam ruangan, robot membutuhkan sistem navigasi yang mampu mengatur gerakan robot untuk menyusuri ruangan. Metode *wall following* adalah suatu metode dalam sistem navigasi robot *mobile* yang dapat digunakan untuk kebutuhan tersebut. Pada metode tersebut, robot akan bergerak berdasarkan jarak dari dinding di sebelah robot. Sehingga, robot membutuhkan sensor jarak yang akurat.

Pada tugas akhir ini, dibuat suatu rancangan yang berjudul "*Implementasi Metode Wall Following berbasis Kinect untuk Robot Mobile*". Sistem yang dirancang memanfaatkan kamera kinect sebagai input sensor jarak. *Kinect* merupakan kamera RGB-D buatan Microsoft. Piksel kedalaman yang dihasilkan kinect dapat diolah untuk menavigasikan robot *mobile*.

Dalam tugas akhir ini, robot akan menelusuri ruangan menggunakan kinect sebagai input jarak, rotari enkoder dan sensor kompas untuk menunjukkan posisi robot. Robot akan terus menelusuri ruangan dengan mengambil referensi jarak dinding di sebelah kiri hingga *user* menghentikan gerakan robot melalui remote kontrol. Pengujian dilakukan dengan menjalankan robot secara otomatis di dalam suatu ruangan. Ruangan akan diberi halangan untuk membagi ruangan menjadi beberapa bagian. Dari pengujian, semakin banyak penambahan halangan untuk membagi ruangan, maka kemampuan telusur robot akan berkurang. Berdasarkan pengamatan dari pengujian pada ruang uji yang didesain, sistem yang dirancang memiliki error rata-rata 7,33% terhadap kemampuan robot dalam memasuki bagian ruangan. Lebar celah ruangan menjadi faktor kegagalan robot dalam memasuki bagian ruangan.

**Kata kunci : Kinect, robot *mobile*, navigasi**

*# Halaman ini sengaja dikosongkan #*

# *IMPLEMENTATION OF WALL FOLLOWING BASED ON KINECT FOR MOBILE ROBOT*

***Name*** ***: Novem Ardan Rohmadin***
***1st Supervisor*** ***: Ronny Mardiyanto, ST., MT., Ph.D.***
***2nd Supervisor*** ***: Rudy Dikairono, ST., MT.***

***ABSTRACT***

*Mobile robot navigation is a way to control the movement of the mobile robot. Robot with special task like dangerous objects searching in the room, robot needs a navigation system that is able to control the movement of the robot to explore the room. Wall-following method is a method in mobile robot navigation system that can be used for such needs. In this method, the robot will move based on the distance from the wall next to the robot. Therefore, robot requires accurate distance sensor.*

*In this final project, made a research entitled "Implementation of Wall Following Method based on Kinect for Robot Mobile". The system is designed utilizing kinect camera as a sensor input range. Kinect is an RGB-D camera made by Microsoft. The resulting pixel depth kinect can be processed to navigate a mobile robot.*

*In this final project, the robot will explore the room using kinect as an input range, rotary encoders and sensor compass to indicate the position of the robot. In automatic mode, robot will continue to explore the room by taking reference wall to the left until the user stops the movement of the robot using remote control. Testing is done by running the robot automatically in a closed room. The room will be an obstacle to divide the room into sections. From the test, the more additional obstacle to divide the room, the search capabilities of robots will be reduced. Based on observations from the testing in test room designed, the system have an average error of 7,33% against the robot's ability to enter the sections of the room. The gap width of the room becomes a factor in the failure of a robot entered the room.*

*Keywords: Kinect, mobile robot, navigation*

*# Halaman ini sengaja dikosongkan #*

# KATA PENGANTAR

*Alhamdulillah*, puji syukur tiada henti penulis haturkan kepada Tuhan semesta alam, Allah SWT, atas seluruh limpahan rahmat, karunia, dan hidayahNya selama ini sehingga penulis mampu menyelesaikan tugas akhir ini.

Tugas Akhir ini dibuat berdasarkan teori-teori yang didapat selama mengikuti perkuliahan, berbagai literatur penunjang dan pengarahan dosen pembimbing dari awal hingga akhir pengerjaan Tugas Akhir ini.

Pada kesempatan ini, penulis ingin berterima kasih kepada pihak-pihak yang membantu pembuatan tugas akhir ini, khususnya kepada:

1. Kedua orang tua tercinta, Bapak Riyanto dan Ibu Siti Musthofiah, yang tidak pernah putus untuk seluruh do'a, nasihat, motivasi, kasih saying dan dukungannya.
2. Bapak Ronny Mardiyanto, ST., MT., Ph. D., selaku dosen pembimbing pertama, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjan penelitian Tugas Akhir ini.
3. Bapak Rudy Dikairono, ST., MT., selaku dosen pembimbing kedua, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjan penelitian tugas akhir ini.
4. Kakak, Novi Rafika Hidayati, dan adik, Harvin Adnin Hidayat, yang selalu memberi semangat dan memotivasi selama pengerjaan penelitian tugas akhir ini.
5. Keluarga besar Hj. Robiyah dan Miratun yang selalu memberi semangat dan memotivasi selama pengerjaan penelitian tugas akhir ini.
6. Teman-teman laboratorium Elektronika ITS yang selalu memberikan semangat dan bantuan dalam menyelesaikan tugas akhir ini.

Surabaya, 2015

Penulis

*# Halaman ini sengaja dikosongkan #*

# DAFTAR ISI

Halaman

*# Halaman ini sengaja dikosongkan #*

# DAFTAR TABEL

Halaman

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Saat ini robot *mobile* telah banyak diaplikasikan dalam kehidupan manusia. Robot telah diaplikasikan dalam bidang hiburan, kebersihan, pencarian benda berbahaya, hingga pencarian korban dalam bencana. Kemampuan robot *mobile* akan selalu dikembangkan untuk memenuhi kebutuhan manusia. Beberapa kemampuan robot *mobile* yang dikembangkan adalah kemampuan robot untuk bergerak secara otomatis. Kemampuan robot *mobile* seperti ini diterapkan pada robot yang bertugas untuk menjelajahi suatu tempat yang tidak dikenali, seperti robot pencari benda berbahaya. Metode yang dapat digunakan dalam menjelajahi ruangan yang tidak dikenali adalah *wall following*. Dengan mengetahui jarak antara robot dengan dinding ruangan, pergerakan robot dapat diatur. Robot akan bergerak mengikuti dinding ruangan sehingga robot mampu memasuki ruangan yang ada.

Penggunaan robot *mobile* dalam mencari benda berbahaya pernah digunakan oleh DENSUS 88[1]. Robot MoroLIPI adalah nama robot tersebut. Kasus dimana robot MoroLIPI digunakan adalah pada saat misi penangkapan teroris Noordin M Top. Robot dengan bentuk seperti tank tersebut bertugas untuk mengetahui situasi didalam ruangan dimana Noordin M Top bersembunyi. Robot bertugas untuk mendeteksi apakah ada benda berbahaya seperti bom yang dapat membahayakan petugas. Pergerakan robot MoroLIPI saat menyusuri ruangan saat itu masih dikendalikan oleh operator secara manual dari jarak jauh. Proses navigasi robot yaitu pergerakan robot *mobile* dalam menyusuri ruangan saat mencari benda berbahaya dapat dikembangkan menjadi sistem otomatis.

Robot *mobile* dengan tujuan menyusuri suatu lingkungan yang tidak dikenal seringkali menggunakan sensor jarak 3 dimensi sebagai input untuk mengatur gerakan robot ataupun hanya untuk pengambilan data. LIDAR (*Light Detecting and Ranging*) adalah salah satu sensor yang sering digunakan sebagai input dalam pemetaan dan navigasi robot. Namun harga LIDAR yang mahal menjadi faktor penghambat bagi para pengembang robotika untuk membeli sensor tersebut.

Dalam kasus ini kinect dapat menjadi solusi untuk memenuhi permasalahan diatas. Kinect adalah sensor jarak 3 dimensi yang

dilengkapi dengan kamera RGB. Sensor buatan Microsoft ini memiliki harga yang lebih murah dibandingkan dengan LIDAR. Kinect dapat menggantikan fungsi LIDAR dalam proses navigasi dan pemetaan.

Oleh karena itu, diajukanlah proposal tugas akhir ini dengan harapan agar dapat mengimplementasikan metode *wall following* berbasis kinect pada robot *mobile* sehingga robot mampu menelusuri ruangan secara otomatis.

## 1.2 Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah :

1. Bagaimana merancang suatu sistem navigasi robot *mobile* untuk menelusuri ruangan yang tidak diketahui menggunakan kinect.
2. Bagaimana merancang sistem yang dapat memandu robot untuk menghindari halangan.

## 1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut :

1. Robot mampu menelusuri ruangan secara otomatis.
2. Robot mampu melewati celah ruangan.
3. Robot mampu menghindari halangan menggunakan kamera kinect pada saat robot bergerak otomatis.

## 1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Tempat pengujian robot berupa bidang datar.
2. Robot tidak mengetahui bentuk ruangan dan letak halangan.
3. Robot tidak memiliki posisi tujuan.
4. Halangan robot berupa benda solid.

## 1.5 Metodologi Penelitian

Langkah-langkah yang dikerjakan pada tugas akhir ini dapat dilihat pada Gambar 1.1.


**Gambar 1. 1** Metodologi Pengerjaan Tugas Akhir

**1. Studi Literatur**

Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan Tugas Akhir. Dasar teori ini dapat diambil dari buku-buku, jurnal, proceeding, dan artikel-artikel di internet.

**2. Perancangan Hardware**

Pada tahap ini dirancang hardware yang terdiri dari Pemodelan desain dari sistem yang terdiri atas kamera kinect, rotari enkoder, sensor kompas, dan remote kontrol yang dihubungkan ke arduino. Arduino dan kamera kinect dihubungkan ke *processing unit* berupa laptop.

**3. Perancangan Software**

Langkah pertama yang dilakukan pada tahap ini adalah pembuatan *source code* untuk menangkap citra kedalaman (*depth*) yang selanjutnya dilakukan konversi data jarak dalam bentuk data kedalaman menjadi bentuk milimeter. Data jarak dalam tugas akhir ini digunakan untuk membuat menavigasikan robot dan membuat peta dua dimensi.

**4. Pengujian Sistem**

Pada tugas akhir ini dilakukan beberapa pengujian, yaitu :

1. Pengukuran jarak obyek dengan sensor kinect.
2. Pengujian kemampuan robot dalam menulusuri ruangan.

**5. Penulisan Laporan Tugas Akhir**

Tahap ini adalah tahapan terakhir dari proses pengerjaan tugas akhir ini. Tahap ini dimulai saat pengambilan data. Laporan tugas akhir ini berisi tentang semua kegiatan yang dilakukan selama mengerjakan tugas akhir.

## 1.6 Sistematika Penulisan

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

- **Bab 1 : Pendahuluan**
  Bab ini meliputi latar belakang, perumusan masalah, tujuan, sistematika penulisan, metodologi, dan relevansi.

- **Bab 2 : Dasar Teori**
  Bab ini menjelaskan tentang berbagai macam teori-teori penunjang dalam pengerjaan tugas akhir ini yang meliputi kinect, kontrol gerakan robot *mobile*, navigasi robot *mobile*.

- **Bab 3: Perancangan Sistem**
  Pada bagian ini berisi perancangan sistem robot, yaitu sistem navigasi mobile robot. Sistem navigasi manual berupa pengaturan gerakan robot menggunakan remote kontrol. Sistem navigasi otomatis berupa pengaturan gerakan robot pada saat menyusuri ruangan.

- **Bab 4 : Pengujian dan Analisis**
  Bab ini menjelaskan data yang didapat dari pengujian keseluruhan sistem beserta analisanya.

- **Bab 5 : Penutup**
  Bagian ini merupakan bagian akhir yang berisikan kesimpulan yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran untuk pengembangannya.

## 1.7 Relevansi

Mata kuliah yang mendukung tugas akhir ini adalah sistem mikroprosessor dan mikrokontroller, perancangan komponen terprogram, serta beberapa reverensi mengenai navigasi robot otomatis dan pembuatan peta dua dimensi berdasarkan data kedalaman kinect.

Hasil dari tugas akhir ini diharapkan dapat menjadi pendukung penelitian pada bidang studi elektronika dalam menggunakan kamera kinect untuk aplikasi penelitian selanjutnya.

*#halaman ini sengaja dikosongkan#*

# BAB II
# TINJAUAN PUSTAKA DAN DASAR TEORI

Dasar teori dalam bab ini menjelaskan tentang teori penunjang yang berhubungan dengan keseluruhan sistem pada tugas akhir ini. Sedangkan tinjauan pustaka dalam bab ini menjelaskan tentang sistem-sistem yang berhubungan dengan tugas akhir ini dan pernah diimplementasikan oleh penulis-penulis sebelumnya

## 2.1 Dasar Teori

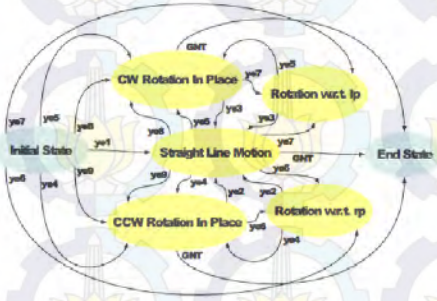
Pada sub bab ini akan dibahas tentang beberapa hal yaitu, navigasi robot, pembacaan posisi robot, kinect, dan pembuatan peta dua dimensi, mikrokontroler arduino, dan *remote* kontrol.

### 2.1.1 Navigasi Robot

Bagi robot *mobile,* kemampuan robot dalam bernavigasi menjadi penting. Robot harus mampu berjalan secara otomatis sesuai dengan perintah sekaligus menghindari dengan obyek lain. Navigasi robot adalah kemampuan robot untuk mengetahui posisi robot dan menentukan gerakan robot selanjutnya dengan aman dan menghindari benturan benda lain[2].

#### *2.1.1.1 Finite State Machine*

Finite State Machines (FSM) adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut: State (Keadaan), Event (masukan) dan action (aksi). Pada satu saat dalam periode waktu yang cukup signifikan, sistem akan berada pada salah satu state yang aktif. Sistem dapat bertransisi menuju state lain jika mendapatkan masukan tertentu (transisi). Transisi keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh sistem ketika menanggapi masukan yang terjadi. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relatif kompleks.

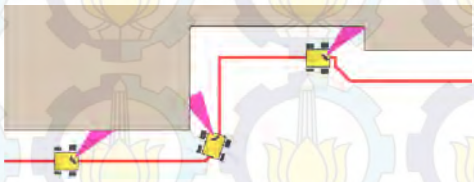



**Gambar 2. 1** Diagram Finite State Machine pada Robot [3]

Dalam robotika, diagram state dapat merepresentasikan strategi robot pada saat berjalan otomatis. State merepresentasikan bentuk gerakan robot. Event merepresentasikan kondisi-kondisi masukan pada robot dalam menentukan gerakan robot. Kondisi-kondisi gerakan robot dapat berupa maju, mundur, belok kanan di tempat, dan belok kiri di tempat.

### *2.1.1.2 Wall Following*

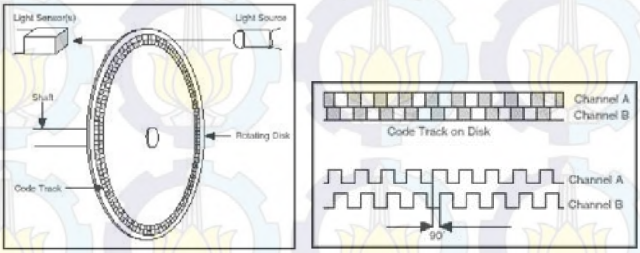
Wall following adalah suatu metode navigasi yang terkenal di kalangan para peneliti robot khususnya di bidang *reactive* robot. Dengan menggunakan sensor jarak, ketika robot terlalu dekat dengan dinding maka robot akan bergerak menjauh ke dinding dan ketika robot terlalu jauh dengan dinding maka robot akan bergerak mendekat ke dinding[4]. Pengimplementasian wall following mampu menggunakan langkah-langkah sederhana seperti maju, lihat, dan belok atau menggunakan tambahan sistem kontrol.



**Gambar 2. 2** Pergerakan Robot Wall Following [5]

### 2.1.2 Pembacaan Posisi dan Arah Robot

#### *2.1.2.1 Rotari Enkoder*




**Gambar 2.3** Rotari Enkoder dengan Perbedaan Fasa dapat mengetahui arah putaran[6]

Rotari enkoder adalah divais elektromekanik yang dapat memonitor gerakan dan posisi. Rotari enkoder umumnya menggunakan sensor optik untuk menghasilkan sinyal pulsa yang dapat diartikan menjadi posisi.

Rotari enkoder tersusun dari suatu piringan tipis yang memiliki lubang-lubang pada bagian lingkaran piringan. LED ditempatkan pada salah satu sisi piringan sehingga cahaya akan menuju ke piringan. Di sisi yang lain phototransistor diletakkan sehingga photo-transistor ini dapat mendeteksi cahaya dari LED yang berseberangan. Apabila posisi piringan mengakibatkan cahaya dari LED dapat mencapai photo-transistor melalui lubang-lubang yang ada, maka photo-transistor akan mengalami saturasi dan akan menghasilkan suatu pulsa gelombang persegi. Semakin banyak deretan pulsa yang dihasilkan pada suatu putaran menentukan akurasi rotari enkoder tersebut.

#### *2.1.2.2 Sensor Kompas*

Kompas adalah alat yang berfungsi untuk menunjukkan arah hadap suatu objek. Kompas merupakan bagian penting dari pembacaan posisi dan arah, sebagai sensor yang dapat mengestimasi arah hadap robot. Dalam bidang navigasi, arah hadap robot merupakan bagian yang penting. Pada sensor kompas terdapat CPU-based dan 2 magnet yang masing-masing kutubnya dihadapkan ke arah mata angin yang berbeda, sehingga arah hadap sensor dapat diketahui.

### *2.1.2.3 Pembacaan Posisi*

Dalam navigasi dan pemetaan menggunakan robot, perlu diketahui posisi dan arah robot dalam bidang kartesian. Konsep basis untuk mengetahui posisi dan arah adalah dengan cara membaca jarak yang telah ditempuh robot setiap roda. Persamaan 1,2,3,4, dan 5[7] digunakan untuk menghitung posisi dan arah robot.

$$\Delta S = \frac{\Delta S_r + \Delta S_l}{2} \quad (1)$$

$$\Delta\theta = \frac{\Delta S_r - \Delta S_l}{B} \quad (2)$$

$$X = X + \Delta S \cos\theta \quad (3)$$

$$Y = Y + \Delta S \sin\theta \quad (4)$$

$$\theta = \theta + \Delta\theta \quad (5)$$

Keterangan: $\Delta S$ = Jarak tempuh terbaru robot

$\Delta S_r$ = Jarak tempuh terbaru roda kanan

$\Delta S_l$ = Jarak tempuh terbaru roda kiri

$\Delta\theta$ = Perubahan sudut terbaru

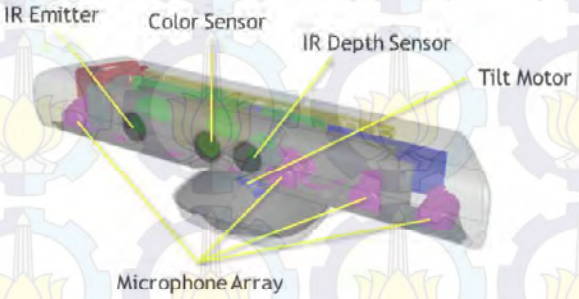
$B$ = Jarak antara roda kiri dan kanan

$X$ = Posisi robot pada bidang x

$Y$ = Posisi robot pada bidang y

$\theta$ = Arah sudut robot

## 2.1.3 Kinect



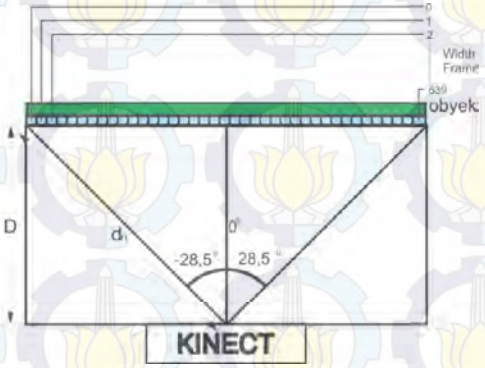

**Gambar 2. 4** Kamera Kinect

Kinect adalah perangkat yang digunakan sebagai pendeteksi gerakan yang awalnya digunakan sebagai kontroler permainan Xbox 360. Kinect dibangun dari teknologi perangkat lunak yang didirikan oleh Rare, anak perusahaan Microsoft Game Studios. Sensor kamera pada kinect dikembangkan oleh perusahaan dari Israel, yaitu Primesense. Kamera Kinect dapat menyediakan data *RGB-Depth* dengan kecepatan fps maksimal 30 Hz dengan resolusi 640 x 480 piksel. Kamera Kinect memiliki sudut pandang 57 derajat pada bidang horizontal dan 43 derajat pada bidang vertikal. Jarak optimal kinect saat digunakan oleh software Xbox adalah 1.2 - 3.5 meter. Apabila jarak yang digunakan melebihi atau kurang dari jarak yang ditentukan, maka obyek tidak akan tertangkap oleh kamera. Pada bagian bawah kinect terdapat *multi-array mic* yang digunakan untuk merekam atau menginputkan suara. Kinect juga dilengkapi dengan *motorized tilt* untuk mengatur sudut kamera, sehingga area yang bisa ditangkap oleh kamera dapat diatur. Untuk mengatur sudut kamera bisa menggunakan program tertentu, dengan jangkauan sudut kurang lebih 27 derajat.

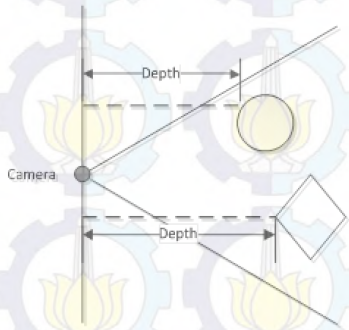
### 2.1.4 Pembuatan Peta Dua Dimensi

Pembuatan peta dua dimensi hasil pemetaan *mobile robot* dengan *tracking mobile robot* pada ruangan yang akan dilewati *mobile robot* membutuhkan data jarak obyek disekitar *mobile robot* dan data posisi *mobile robot* pada bidang kartesian. Data jarak diperoleh dengan menggunakan sensor kinect sedangkan data posisi *mobile robot* diperoleh dengan menggunakan rotari enkoder dan kompas.

Seperti pada gambar 2.5, pengukuran jarak dengan sensor kinect dari titik sensor kinect dengan obyek yang ditangkap memiliki sudut pengukuran vertical (43 derajat) dan horizontal (57 derajat). Jarak pengukuran minimum pengukuran dari sensor kinect adalah 80 cm dan jarak pengukuran maksimal adalah 3,7 meter.



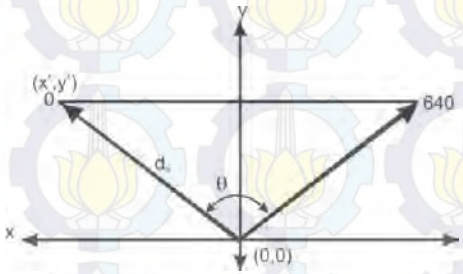

**Gambar 2. 5** *Angle View* Horizontal Kamera Kinect




**Gambar 2. 6** Nilai *depth stream*[9]

Data nilai *depth* yang diberikan kinect adalah jarak tegak lurus antara obyek dengan bidang kamera kinect (*D*) pada setiap pikselnya. Pada umumnya terdapat 640x480 piksel. Untuk menggambarkan kondisi lingkungan di sekitar ruangan dibutuhkan jarak *d*, maka dihitung dengan persamaan segitiga seperti pada persamaan 6. Pengambilan data jarak hasil sensor kinect dimulai pada piksel paling kiri (*d1*) sampai piksel paling kanan (*d640*) sehingga sudut mulai adalah -28,5 derajat. Kemudian besarnya sudut untuk setiap penggambaran jarak pada piksel berikutnya (*dn, dimana n=1 s/d 640*) akan ditambah dengan 0,089 derajat ($57^0/640$), yaitu :

$$d_n = D_n \sin\theta \qquad (6)$$




**Gambar 2. 7** Hubungan posisi dan pengambilan jarak pada Kinect

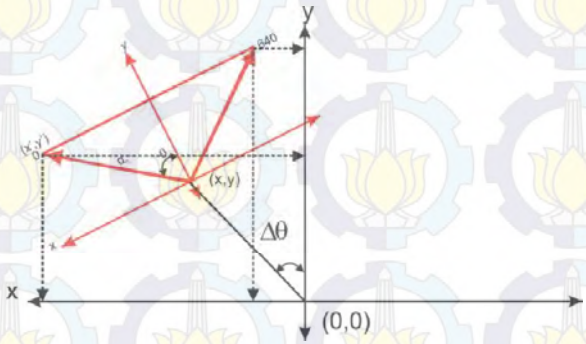
Dengan demikian akan didapat jarak *d* untuk masing-masing piksel (*d1* s/d *d640*) dari titik *A* sampai ke titik. Sesuai dengan ilustrasi pada gambar 2.7, pertama kali pengambilan data, robot akan berada pada posisi (0,0) pada bidang kartesian. Penggambaran jarak hasil sensor kinect pada peta dua dimensi dimulai pada piksel paling kiri (*d1*) sampai piksel paling kanan (*d640*) sehingga sudut mulai adalah -28,5 derajat. Kemudian besarnya sudut untuk setiap penggambaran jarak pada piksel berikutnya (*dn, dimana n=1 s/d 640*) akan ditambah dengan 0,089 derajat.

Gambar 2.8 menunjukkan mengenai cara untuk menggambarkan koordinat piksel yang menunjukkan jarak obyek untuk setiap nilai *d*. Nilai koordinat piksel (*x',y'*) dicari dari *d1* hingga *d640*. Setiap kenaikan *d* maka nilai sudut sebelumnya akan ditambah 0,089 derajat. Untuk mencari Nilai koordinat piksel (*x',y'*) ditunjukkan pada persamaan 10 dan 11.

$$x' = d_n \sin\theta \quad (7)$$

$$y' = d_n \cos\theta \quad (8)$$

Keterangan :
- x' = posisi koordinat x untuk jarak *d*
- y' = posisi koordinat y untuk jarak *d*
- d = jarak obyek dengan pusat kamera
- n = index *d* pada *width frame* (1 s/d 640)
- $\theta$ = besarnya sudut dari index jarak ke n pada frame pada titik 0 derajat




**Gambar 2. 8** Hubungan posisi dan pengambilan jarak pada kinect pada kondisi kinect bergerak

Persamaan 7 dan 8 digunakan jika robot selalu berada pada titik (0,0). Sehingga persamaan 7 dan 8 tidak efektif pada saat sensor kinect bergerak dan berotasi meninggalkan titik (0,0) pada bidang kartesian. Persamaan 7 dan 8 dikembangkan menjadi persamaan 9 dan 10.

$$x' = x + (d_n \sin(\theta + \Delta\theta)) \quad (9)$$

$$y' = y + (d_n \sin(\theta + \Delta\theta)) \quad (10)$$

Keterangan :
- x' = posisi koordinat x untuk jarak *d*
- y' = posisi koordinat y untuk jarak *d*
- x = posisi koordinat x robot saat ini
- y = posisi koordinat y robot saat ini
- d = jarak obyek dengan pusat kamera
- n = index *d* pada *width frame* (1 s/d 640)
- $\theta$ = besarnya sudut dari index jarak ke n pada frame pada titik 0 derajat
- $\Delta\theta$ = besarnya sudut dari index jarak ke n pada frame pada titik 0 derajat

### 2.1.5 Mikrokontroler Arduino


**Gambar 2. 9** Arduino Mega[10]

Arduino merupakan suatu platform dari *physical computing* yang bersifat *open source*. Arduino bukan hanya sebagai modul mikrokontroler, namun arduino adalah kombinasi dari hardware, bahasa pemrograman dan *Integrated Development Environment* (IDE) yang canggih. IDE adalah sebuah *software* yang sangat berperan untuk menulis program, meng-*compile* menjadi kode biner dan meng-upload ke dalam memori mikrokontroler. Ada banyak projek dan alat-alat dikembangkan oleh akademisi dan profesional dengan menggunakan Arduino, selain itu juga ada banyak modul-modul pendukung (sensor, tampilan, penggerak dan sebagainya) yang dibuat oleh pihak lain untuk bisa disambungkan dengan Arduino.

IDE Arduino adalah software yanhg ditulis dengan menggunakan Java. IDE Arduino terdiri dari editor program, sebuah window yang memungkinkan pengguna menulis dan mengedit program dalam bahasa *Processing*. Arduino menggunakan koneksi USB (*Universal Serial Bus*) menggunakan chip FTDI (*Future Technology Devices International*) untuk melakukan pemrograman, dan biasanya pada chip Arduino sudah dimasukkan bootloader, sehingga dapat dilakukan pemrograman langsung ke dalam chip menggunakan software Arduino.

Perangkat lunak arduino merupakan software *Open Source*. Perangkat lunak Arduino IDE dipublikasikan sebagai aplikasi *open source* sehingga dapat dikembangkan lebih lanjut untuk pengembangan lebih lanjut. Bahasa IDE arduino menggunakan C/C++.

### 2.1.6 Radio Kontrol

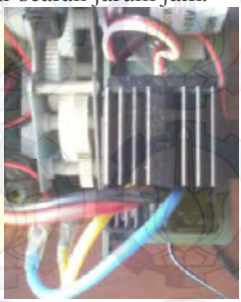
Radio kontrol adalah penggunaan sinyal radio untuk mengontrol alat lain dari jauh. Alat yang dimaksud dapat merujuk ke pengontrolan mobil, kapal dan pesawat model dari sebuah kotak radio kontrol yang dipegang pengguna. Radio kontrol yang saat ini sering digunakan untuk model adalah radio kontrol 2,4 Ghz. Dengan radio kontrol 2,4 Ghz,

transmitter dan receiver radio kontrol mampu mengunci pada frekuensi yang sama sehingga tidak gangguan dari radio lain.


**Gambar 2. 10** Radio Kontrol Turnigy 9x [13]

### 2.1.7 Electronic Speed Control

Electronic Speed Control adalah divais yang berfungsi untuk mengatur kecepatan putar motor menggunakan sinyal PWM dengan frekuensi dan *duty cycle* tertentu. ESC digunakan pada kendaraan radio kontrol. ESC menggunakan tegangan DC sebagai catu dayanya. ESC membaca sinyal PWM yang diinputkan. ESC menggunakan sinyal persegi dengan frekuensi 50 Hz dengan lebar pulsa antara 1ms hingga 2ms. Lebar pulsa sinyal yang dimasukkan pada ESC akan mempengaruhi kecepatan putar motor dan arah putar motor. Sinyal PWM dengan lebar pulsa antara 1ms dan kurang dari 1,5ms akan mengakibatkan motor berputar berlawanan arah jarum jam. Sinyal PWM dengan lebar pulsa 1,5ms akan mengakibatkan motor diam. Sinyal PWM dengan lebar pulsa antara 1,5ms dan kurang dari 2ms akan mengakibatkan motor berputar searah jarum jam.



**Gambar 2.11** Electronic Speed Control Hobywing WP-1040 Brushed

### *2.1.8 Jarak Ecludian*

Jarak *ecludian* adalah perhitungan jarak dari 2 buah titik dalam suatu koordinat. *Euclidean* diperkenalkan oleh Euclid, seorang matematikawan dari Yunani sekitar tahun 300 sebelum mahesi untuk mempelajari hubungan antara sudut dan jarak. *Euclidean* ini berkaitan dengan Teorema Phytagoras. Dalam tugas akhir ini, jarak ecludian digunakan untuk mengukur jarak antar 2 titik dalam bidang kartesian. Pada pengukuran jarak *ecludian* dalam bidang kartesian, dimana dua buah titik, yaitu p1 = (x1, y1) dan p2 = (x2, y2), jaraknya (d) adalah :

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (11)$$

## 2.2 Tinjauan Pustaka

### 2.2.1 VIPeR

VIPeR adalah robot militer yang dikembangkan oleh Perusahaan Israel yang bernama *Elbit Systems* dan digunakan pada lokasi peperangan. Robot ini diresmikan pada bulan Maret 2007. Agar robot dapat bergerak dengan bebas, VIPeR menggunakan roda track *Galilleo* dimana roda ini mampu mengikuti lingkungan. Robot VIPeR digunakan untuk mendeteksi IED (*improvised explosive device*), memeriksa *booby traps*, memeriksa lokasi musuh, dan mengingatkan prajurit jika ada bahaya di depan. Robot dengan bobot 11 kg ini, dilengkapi dengan kamera thermal, pencium bahan peledak, senjata mini uzi, pelepas granat, *gripper,* dan kemampuan untuk memetakan ruangan. Penggunaan khusus dari robot VIPeR digunakan pada lokasi berbahaya, dimana akan sangat berbahaya bagi prajurit untuk masuk secara langsung ke dalam lokasi tersebut. Penggunaannya bias digunakan di goad an terowongan. VIPeR dikendalikan dengan menggunakan *remote control* dan HMD (*Helmet Mounted Display*).

**Gambar 2. 12** Robot VIPeR[16]

### 2.2.2 Microsoft Kinect SDK

**Gambar 2. 13** Logo Kinect for Windows [9]

Microsoft Kinect SDK (Software Development Kit) merupakan sebuah library dari fungsi programming untuk penggunaan divais khusus yaitu Kinect. Library ini dimanfaatkan oleh program-program lainnya (seperti C++, C# dan lain-lain) untuk melakukan pengambilan, pengolahan serta penampilan data gambar, baik dalam bentuk citra dan video maupun *real time* video. Library ini juga sering digunakan dalam hal interaksi manusia dengan mesin sehingga *user* dapat langsung berinteraksi dengan program yang telah dibuat oleh programmer. Library ini juga biasa digunakan dalam pembuatan sebuah game yang tentunya basis dari game tersebut adalah Kinect itu sendiri untuk melakukan kontrol dalam game tersebut.

### 2.2.3 OpenCV

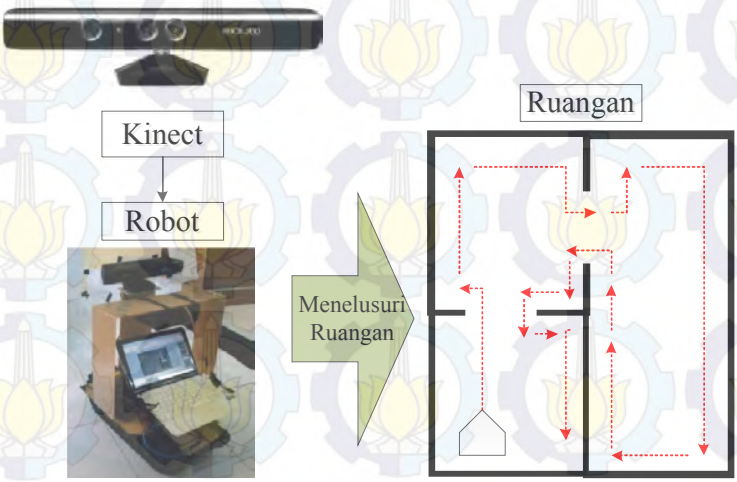

**Gambar 2. 14** Logo OpenCV [11]

OpenCV (*Open Source Vision Computer Library*) adalah sebuah API (*Application Programming Interface*) Library mengenai *computer vision* dan *machine learning*. OpenCV dibangun untuk menyediakan kebutuhan umum untuk aplikasi *computer vision*. OpenCV menyediakan lebih dari 2500 algoritma yang dapat bekerja dengan optimal. Algoritma ini dapat digunakan untuk mendeteksi dan mengenali wajah, mengidentifikasi objek, mengklasifikasikan tindakan manusia dalam video, menghasilkan model 3D dari gambar, menghasilkan *point cloud data* dari kamera stereo, menemukan gambar yang sama dari database gambar, dan lain-lain. OpenCV telah digunakan secara luas di perusahaan dan beberapa kelompok penelitian. Beberapa perusahaan, seperti Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, dan Toyota telah menggunakan *library* OpenCV. Library ini dikembangkan dalam berbagai bahasa pemrogaman seperti C++, C, Python, dan Java. OpenCV dapat digunakan di Windows, Linux, Android dan Mac OS.

*#halaman ini sengaja dikosongkan#*

# BAB III
# PERANCANGAN SISTEM

Secara keseluruhan sistem berupa robot pencari benda berbahaya yang dinavigasikan dalam ruang yang tidak dikenal. Sistem navigasi untuk menelusuri ruangan yang dirancang dalam tugas akhir ini terbagi mejadi beberapa bagian utama, yaitu remote kontrol sebagai alat bantu *user* untuk mengatur gerakan robot, robot dengan sistem mekanik roda track, kamera kinect sebagai input jarak bagi robot agar dapat menyusuri ruangan, dan sensor rotari enkoder serta sensor kompas untuk mengetahui posisi dan arah robot selama penelusuran. Untuk bagian pendeteksian benda berbahaya tidak dijelaskan pada tugas akhir ini.




**Gambar 3. 1** Ilustrasi Sistem

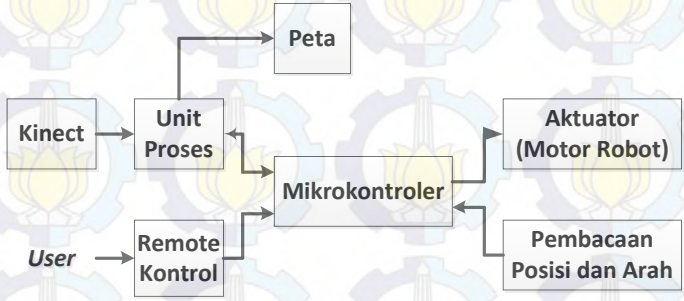
## 3.1 Diagram Blok Sistem

Robot dikendalikan oleh seorang *user* menggunakan radio kontrol. Sinyal dari remote kontrol akan diterima oleh *receiver*. Robot akan membaca sinyal dari *receiver*. Berdasarkan sinyal yang dibaca oleh *receiver*, mode gerak robot akan ditentukan. Terdapat 2 mode, yaitu manual dan otomatis. Saat mode auto, gerakan robot diatur sepenuhnya oleh unit proses. *User* memilih mode melalui saklar pada remote. Rotari enkoder dan kompas membaca kondisi robot, baik posisi maupun arah. Data posisi dan arah dikirim ke unit proses. Kinect membaca kondisi lingkungan berupa jarak obyek di depan robot. Data jarak kinect diambil dan diolah oleh unit proses. Pertama kali robot bergerak, robot akan menelusuri dinding sebelah kiri. Unit proses akan bergerak maju untuk dan selanjutnya memeriksa apakah ada celah disebelah kiri robot. Jika terdapat celah maka robot akan bergerak masuk ke celah, jika tidak maka robot akan kembali maju pada arah sebelumnya.

Selama robot bergerak, unit proses akan menghasilkan peta dua dimensi berdasarkan jarak obyek di depan robot, posisi dan arah robot. Peta berisi kondisi lingkungan di sekitar robot secara 2 dimensi. Peta ini merupakan fitur tambahan bagi robot *mobile*. Hasil ketentuan gerakan robot yang dihasilkan oleh unit proses akan dikirim ke mikrokontroler. Robot akan terus bergerak sesuai dengan intruksi unit proses. Robot akan berjalan secara otomatis hingga terdapat intruksi kembali ke mode manual dari *user* melalu radio kontrol.

Saat mode manual, gerakan robot diatur sepenuhnya oleh *user* melalui joystick remote. Remote kontrol akan disetting sehingga gerakan maju, mundur, belok kanan dan belok kiri robot akan dikendalikan menggunakan joystick remote kontrol. *User* akan mengendalikan robot dari jarak jauh. Penggunaan remote kontrol bertujuan agar robot mampu dikendalikan dari jarak yang aman dari lokasi penggunaan robot.

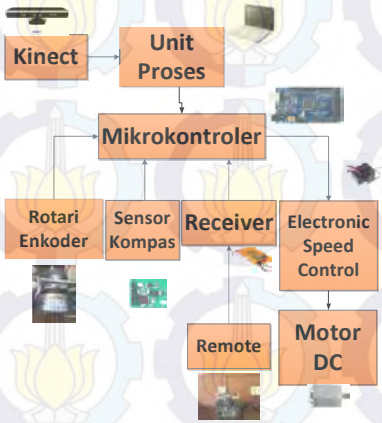
Cara kerja dari sistem secara umum digambarkan pada blok diagram pada gambar 3.2.




**Gambar 3. 2** Diagram Sistem Robot Mobile

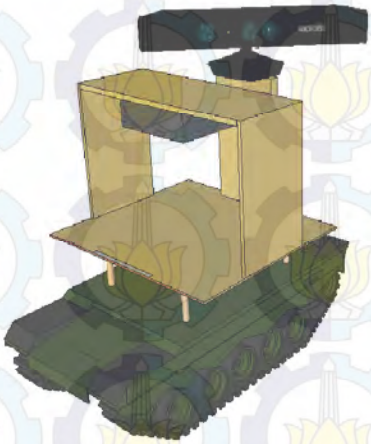
## 3.2 Perancangan Perangkat Keras Robot

Sebagaimana terlihat pada gambar 3.3, perangkat keras yang digunakan pada tugas akhir ini merupakan laptop atau komputer sebagai unit pengambilan data, pemrosesan data dan visualisasi data, kinect sebagai sensor untuk pengambil data *depth*, mikrokontroler sebagai pembaca sinyal pwm dan pembangkit sinyal pwm, rotari enkoder dan sensor kompas sebagai pembaca posisi dan arah, ESC (*electronic speed control*) sebagai pengendali motor dc, *receiver* dan *transmitter* radio kontrol sebagai penghubung *user* dengan robot.

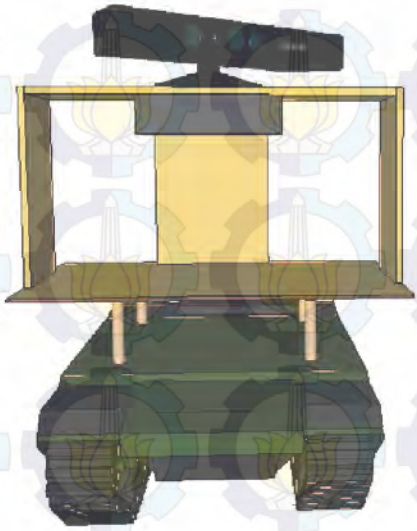



**Gambar 3. 3** Diagram blok perangkat keras sistem navigasi berbasis kinect

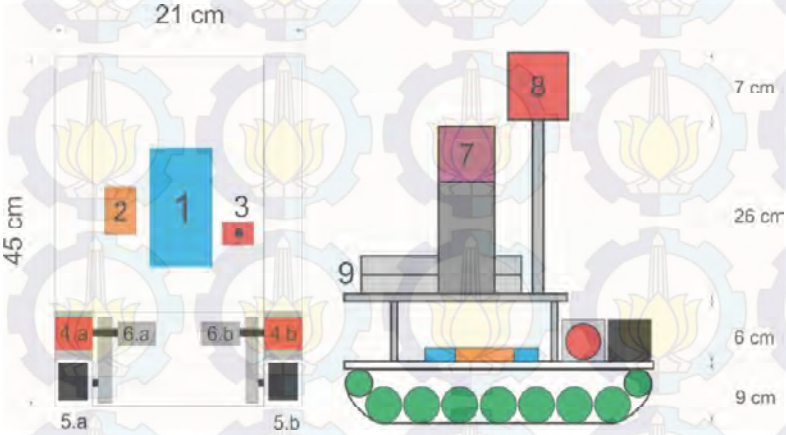
### 3.2.1 Desain Robot



(a)



**Gambar 3. 4** Desain 3d robot (a) tampak atas, (b) tampak depan



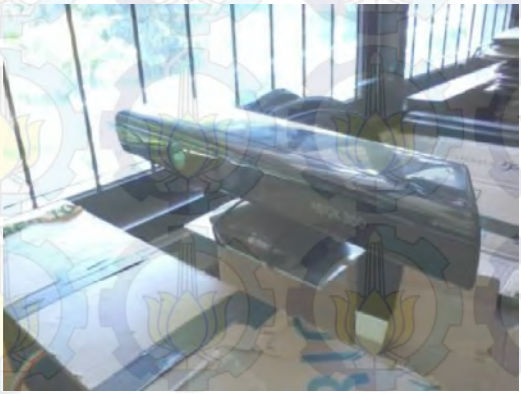

**Gambar 3. 5** Desain Mekanik Robot

Keterangan :

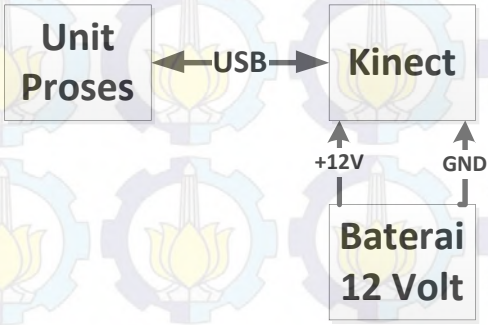
1. Arduino Mega
2. *Receiver*
3. Tombol On/Off
4. Rotari Enkoder : (a) kiri, (b) kanan
5. ESC (*Electronic Speed Controller*) : (a) kiri, (b) kanan
6. Motor DC : (a) kiri, (b) kanan
7. Sensor kompas
8. Kinect
9. Unit Proses (PC)

Pada gambar 3.4, *Mobile* robot yang dirancang, memiliki bentuk dasar seperti tank dengan 2 buah roda track yang dihubungkan dengan motor dc sebagai penggerak utama. Kinect ditempatkan dibagian atas robot agar jarak pandang robot lebih luas. Dengan memiringkan arah hadap kinect $45^0$ ke arah kiri sehingga kamera kinect dapat melihat dinding di sebelah kiri robot .Gambar 3.5, menjelaskan bagian – bagian dari robot.

### 3.2.2 Kinect



**Gambar 3. 6** Kamera Kinect




**Gambar 3. 7** Diagram Blok Kinect dengan Unit Proses

Kinect yang digunakan adalah Kinect generasi pertama atau yang sering disebut dengan Kinect Xbox 360. Kinect ini memiliki 1 buah lensa *depth* dengan spesifikasi :

- Resolusi : 640 x 480 pixels
- *Frame rate* : 30 frame per second
- *Operation Range* : 0,8 – 3,7 m
- Sudut Pandang Vertikal : $43^0$
- Sudut Pandang Horizontal : $57^0$

### 3.2.3 Unit Proses

**Gambar 3. 8** Unit Proses

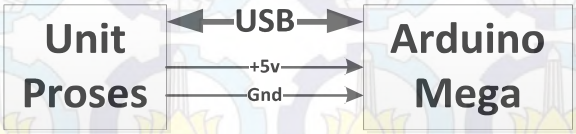
Pada tugas akhir ini digunakan sebuah komputer dengan spesifikasi :

- Merk : Samsung
- CPU : Intel Atom CPU N2600 1.6 GHz
- RAM : 2 GB
- Sistem Operasi : 32-bit

Pemrosesan berupa pengambilan data kinect, pengolahan data kinect, pembuatan peta 2D, pengaturan gerak robot dilakukan pada unit proses.

### 3.2.4 Mikrokontroler Arduino

Mikrokontroler yang digunakan adalah mikrokontroler Arduino Mega. Catu daya arduino mega menggunakan catu daya dari unit proses.



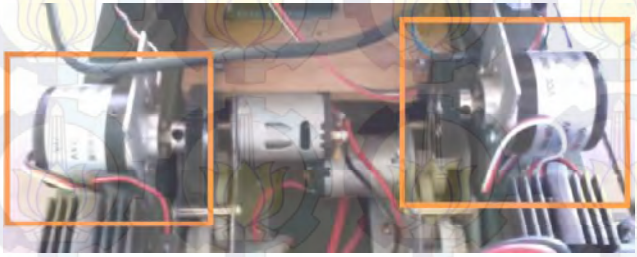

**Gambar 3. 9** Diagram Arduino Mega dengan Unit Proses

Berikut merupakan spesifikasi mikrokontroler Arduino Mega yang digunakan :

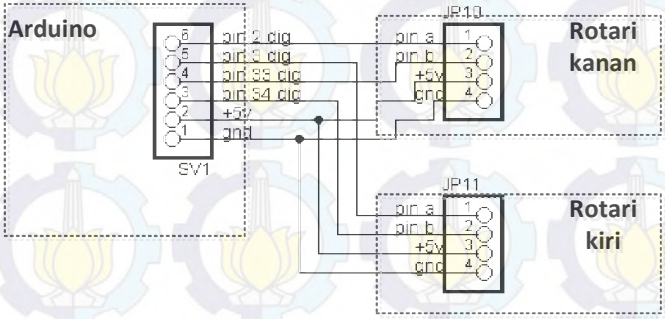
**Tabel 3. 1** Spesifikasi Arduino Mega[10]

| | |
|---|---|
| IC Mikrokontroler | ATMega1280 |
| Tegangan Operasional | 5V |
| Catu daya (rekomendasi) | 7-12V |
| Catu daya (batas) | 6-20V |
| Input Digital | 54 |
| Analog | 16 |
| Arus DC per input | 40mA |
| Flash Memori | 128 KB |
| EEPROM | 4 KB |
| Clock Speed | 16Mhz |

### 3.2.5 Rotari Enkoder



**Gambar 3. 10** Rotari Enkoder




**Gambar 3. 11** Pin diagram arduino mega dengan rotari enkoder

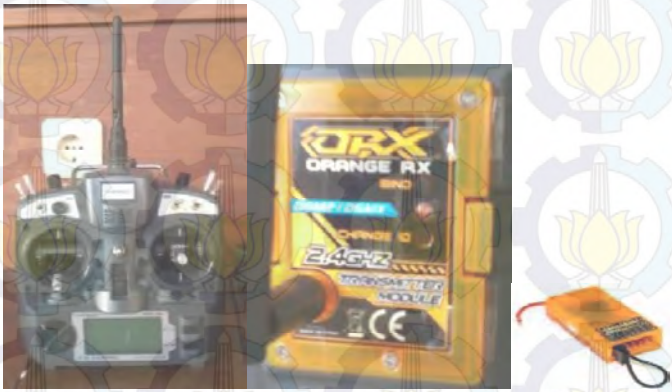
Rotari enkoder yang digunakan adalah Rotari enkoder 2 kanal buatan Sanyou Technology. Berikut merupakan spesifikasi rotari enkoder yang digunakan.

- Tipe : Incremental
- Resolusi : 400 P/R

Pin a rotari kanan dan pin a rotari kiri dihubungkan dengan pin interrupt eksternal dari arduino mega. Pin interrupt eksternal yang digunakan adalah pin 2 dan 3 dari Arduino Mega. Sedangkan pin b rotari enkoder kanan dihubungkan dengan pin digital 33 dan pin b rotari enkoder kiri dihubungkan dengan pin digital 34.

### 3.2.6 Radio Kontrol

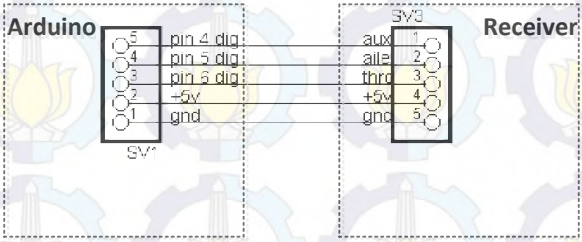
Radio kontrol digunakan sebagai penghubung *user* dengan robot. Modul radio kontrol terdiri dari 3 bagian, yaitu remote, transmitter, dan *receiver*. Remote adalah bagian radio kontrol yang berhubungan langsung dengan *user*. *Transmitter* adalah bagian radio kontrol yang mengirim data dari remote ke *receiver*. *Receiver* adalah bagian yang menerima sinyal transmisi data dari *transmitter*. Remote yang digunakan adalah remote Turnigy 9x 2,4GHz. Modul *transmitter* dan *receiver* yang digunakan adalah OrangeRX 6 channel.




(a) (b) (c)

**Gambar 3. 12** (a)Remote Turnigy 9x, (b)Transmitter Orange RX 2,4Ghz, (c)Receiver Orange RX 2,4Ghz 6 channel


**Gambar 3. 13** Kontrol Gerakan oleh *user* melalui *remote control*

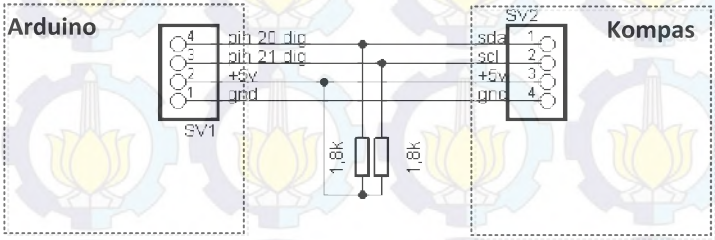



**Gambar 3. 14** Pin Diagram Arduino Mega dengan Receiver

Seperti pada gambar 3.13, radio kontrol telah disetting agar joystick kanan menjadi pengatur gerakan robot secara manual. Dengan menekan saklar auto/manual maka mode robot dapat dijadikan auto atau manual.

### 3.2.7 Sensor Kompas CMPS 11

CMPS 11 adalah modul kompas buatan Devantech. CMPS11 adalah kompas generasi ke-3 Devantech dengan kompensasis *tilt*. CMPS11 menggunakan sensor LSM9DS0 buatan ST electronics. Sensor kompas ini terdiri atas magnetometer 3 sumbu, gyro 3 sumbu, dan accelerometer 3 sumbu.

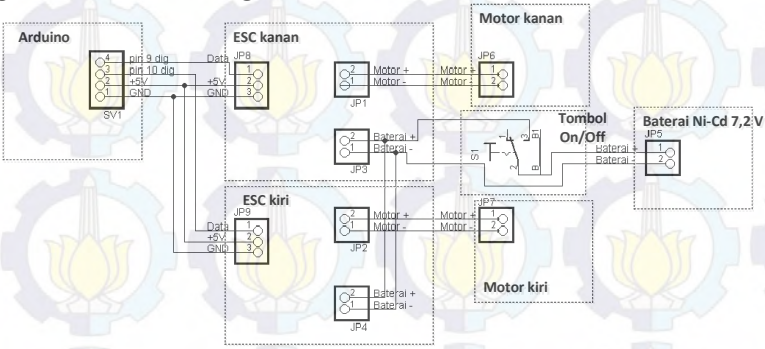



**Gambar 3. 15** Pin diagram CMPS 11 dengan Arduino Mega

Sensor magnetometer adalah sensor yang bekerja dengan pembacaan gaya magnet bumi. Accelerometer adalah sensor yang mengukur percepatan suatu benda. Gyroscope adalah sensor yang mengukur rotasi dari suatu benda. Dengan mengkombinasikan kalman filter pada sensor gyro dan accelerometer, error yang disebabkan oleh proses tilting pada modul sensor dapat dikurangi. CMPS11 membutuhkan tegangan kerja sebesar 3.6-5 V. Data dari CMPS11 diakses menggunakan komunikasi I2C.

### *3.2.8 Electronic Speed Controller*

Driver motor yang digunakan pada robot *mobile* yang didesain adalah ESC Hobbywing WP - 1040 - Brushed Crawler. Pada tugas akhir ini digunakan 2 buah ESC, 1 untuk motor kanan dan 1 untuk motor kiri. Supply yang digunakan untuk *driver* motor menggunakan baterai Ni-Cd 7,2 volt. Untuk menggerakkan motor menggunakan ESC ini, pin data pada ESC perlu diberi sinyal PWM (*Pulse Width Modulation*). Sinyal PWM yang digunakan adalah sinyal persegi dengan frekuensi 50 Hz dan waktu aktif antara 10 hingga 20 ms. Sinyal PWM ini dibangkitkan dari pin PWM Arduino Mega.



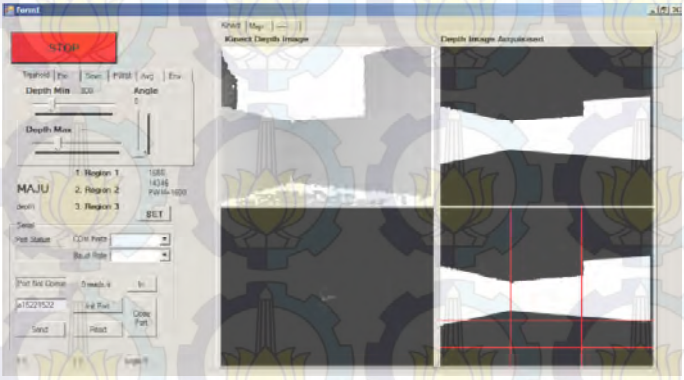

**Gambar 3. 16** Pin Diagram ESC dengan Arduino Mega

## 3.3 Perancangan Perangkat Lunak

Dalam perancangan software, melibatkan *interface* program, algoritma pemrogaman yang digunakan pada unit proses untuk navigasi robot dan pembuatan peta antara lain pengambilan data *depth*, pengolahan data *depth*, rancangan kondisi gerakan robot. Sedangkan untuk pembuatan peta pada unit proses, yaitu pengambilan data *depth*, konversi data *depth*, pembacaan posisi, dan penggambaran peta. Untuk mikrokontroler, antara lain pengambilan data posisi dan arah, dan pembacaan sinyal pwm dari receiver, dan pembangkitan sinyal pwm untuk ESC.

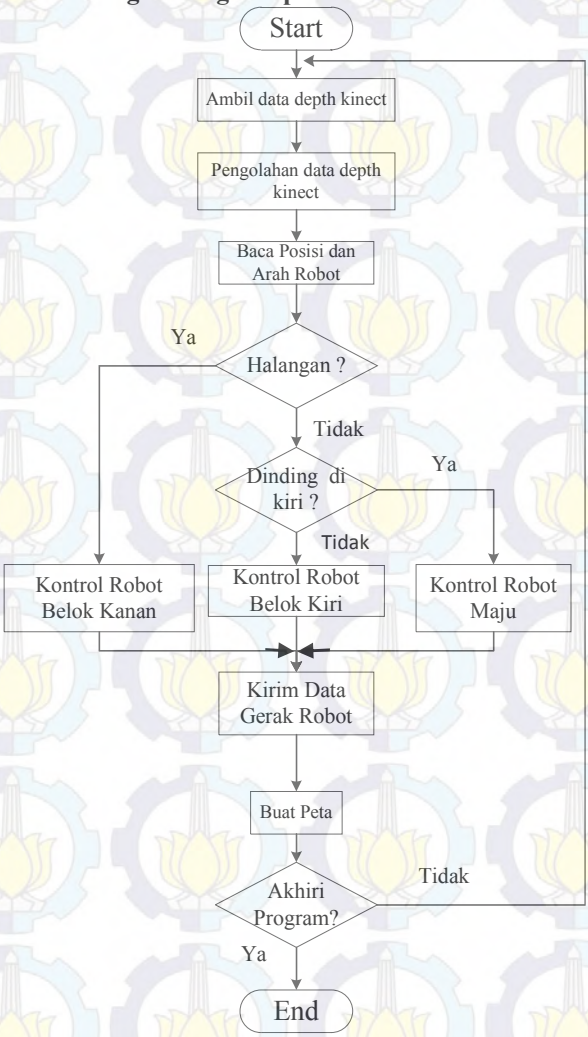
### 3.3.1 Perancangan Tampilan Program pada Unit Proses

Pada tahap ini, dirancang tampilan GUI (*Graphic Interface Unit*) dengan tujuan untuk mempermudah pengguna dalam merubah parameter – parameter robot dan menunjukkan kondisi posisi dan arah robot. Seperti pada gambar 3.17, display gambar *depth* dari Kinect ditampilkan untuk mengetahui keadaan yang dibaca oleh kinect. Kondisi gerakan robot akan ditampilkan pula. Peta yang dihasilkan oleh unit proses dapat disimpan dengan menekan tombol *save image* pada tab *Save*. Posisi dan arah robot ditampilkan pada bagian pojok kiri bawah tampilan program.




**Gambar 3. 17** Tampilan GUI

### 3.3.2 Perancangan Program pada Unit Proses




**Gambar 3. 18** Flowchart Sistem Pergerakan Robot

Pada gambar 3.18, ditampilkan flowchart mengenai pergerakan utama robot. Penjelasan flowchart diatas dijelaskan melalui poin-poin berikut :

1. Robot mula - mula akan mengambil data depth dari kamera kinect. Data depth yang diambil ini berupa matriks 640x1 yang berisi data kedalaman yang ditangkap oleh kamera kinect.
2. Data depth dari kinect akan diolah dan dibagi menjadi beberapa region. Nilai data kedalaman rata-rata per region akan dibandingkan untuk mengetahui apakah robot menemui halangan, celah di kiri robot, atau jalan depan kosong dan robot berada dekat dinding.
3. Jika jalan depan kosong dan robot berada dekat dinding, maka robot akan dikontrol untuk bergerak maju.
4. Jika terdapat dinding di kiri robot, maka robot akan dikontrol untuk bergerak ke maju. Jika tidak ada dinding maka robot akan bergerak ke kiri.
5. Jika terdapat halangan di depan robot maka robot akan dikontrol untuk segera belok ke arah kanan. Aksi ini akan berhenti jika robot menemui keadaan seperti pada urutan nomor 3 dan 4.
6. Saat robot bergerak, robot akan sekaligus menggambar peta lingkungan yang ada di sekitarnya.
7. Robot akan keluar dari mode otomatis dan kembali ke mode manual jika user menghentikan gerakan robot menggunakan remote kontrol.

### 3.3.3 Pengambilan Data Depth dari Kamera Kinect

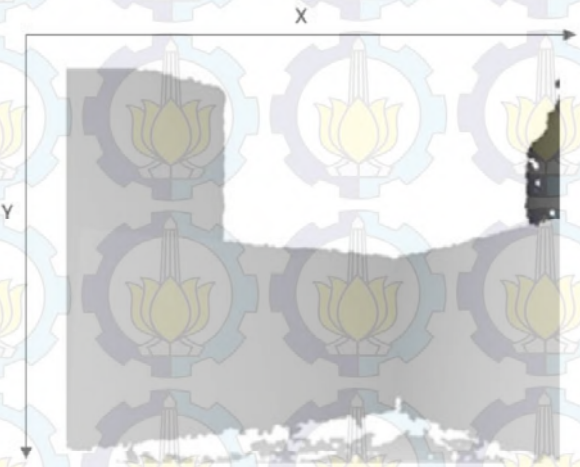
Data *depth* yang diterima oleh unit proses berupa data jarak antara obyek tegak lurus dengan bidang kamera kinect. Dari data yang diambil, terdapat data *depth* dengan ukuran 640x480 piksel. Dalam sistem ini, robot akan megolah data depth untuk 2 kebutuhan, yaitu:

a. Navigasi
b. Pembuatan Peta

#### *3.3.3.1 Inisialisasi Sensor Kinect*

Untuk melakukan pengukuran jarak, maka fungsi sensor *depth* dari Kinect diaktifkan. Pantulan dari sinar inframerah yang dikeluarkan oleh transmitter inframerah dapat ditangkap dan dapat diolah menjadi data milimeter. Fungsi untuk mengaktifkan lensa *depth* yaitu,

```
NuiInitialize(NUI_INITIALIZE_FLAG_USES_DEPTH);
```




**Gambar 3. 19** Pengambilan data *depth* kamera Kinect

Gambar 3.19 adalah hasil dari pengambilan frame *depth* yang diambil dari kamera kinect menggunakan library Kinect SDK. Frame yang diambil merupakan matriks yang memiliki ukuran 640 x 480 elemen. Setiap elemen nya menyimpan data yang berisi kedalaman pada piksel tersebut. Kode program yang digunakan untuk mengakses setiap bagian dari elemen matriks depth adalah sebagai berikut :

```
RGBQUAD Nui_ShortToQuad_Depth(USHORT s)
{
    USHORT realDepth = (s&0xfff8) >> 3;
    BYTE l = 255-(BYTE)(256*realDepth /(0x0fff));
    dalam = realDepth;
    RGBQUAD q;
    q.rgbRed = q.rgbBlue = q.rgbGreen = l;
    return q;
}
```

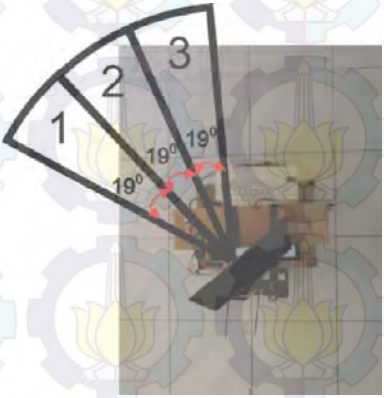
Variabel `dalam` pada kode tersebut menyimpan informasi nilai kedalaman setiap piksel dalam frame *depth*.

### *3.3.3.2 Pengambilan Data Depth Kinect untuk Navigasi*

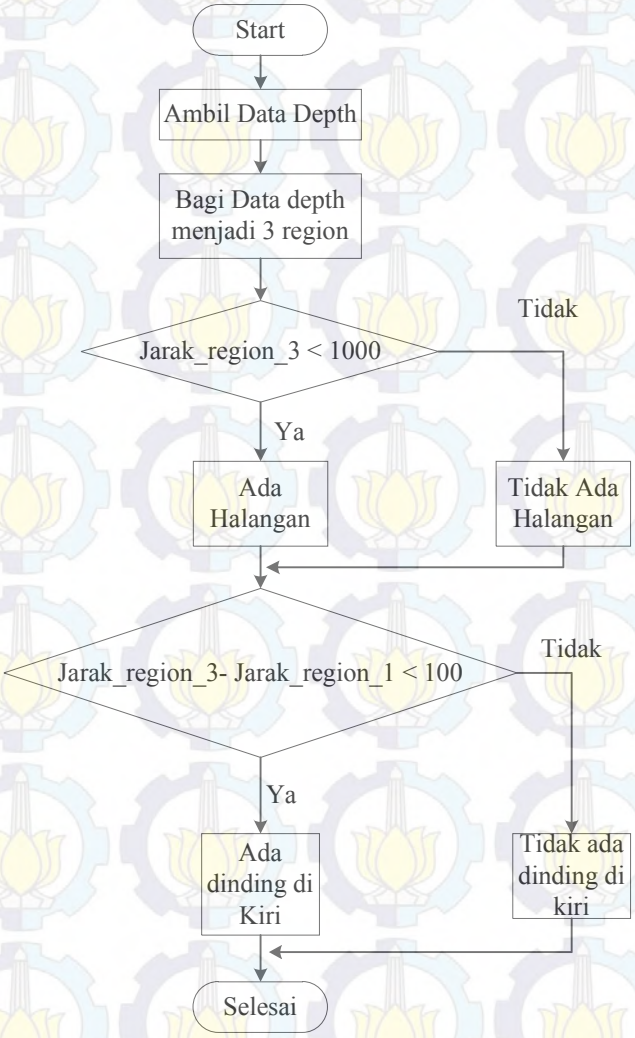
Data depth dalam navigasi digunakan oleh robot untuk mengetahui kondisi lingkungan yang ada di depannya, apakah ada halangan di depan robot atau tidak. Dalam tugas akhir ini, sistem memanfaatkan data kedalaman yang diambil oleh kamera kinect. Data tersebut akan dijadikan acuan pergerakan robot.

Dalam sistem yang dirancang, data frame depth dibagi menjadi beberapa region. Frame *depth* dibagi menjadi 3 region secara vertikal, yaitu region 1 (kiri), region 2 (tengah), dan region 3 (kanan). Setiap region memiliki lebar 213 piksel secara vertikal. Pada bidang vertikal, region 1 pada piksel 0 sampai 212, region 2 pada piksel 213 sampai 415, dan region 3 pada piksel 416 sampai 639. Region ini yang akan digunakan dalam penentuan gerakan robot. Region 1,2 dan 3 dalam program disebut dalam variabel `jarak_region_1`, `jarak_region_2`, dan `jarak_region_3`.

Selanjutnya data kedalaman dari setiap region 1, 2, dan 3 dihitung nilai rata-rata data kedalaman yang telah diambil. Region 1 adalah tangkapan jarak dari bagian kiri robot. Region 2 adalah tangkapan jarak dari bagian kiri depan robot. Region 3 adalah tangkapan jarak dari bagian depan robot.




**Gambar 3. 20** Besar Sudut Tangkapan Jarak oleh Kamera Kinect




**Gambar 3. 21** Flowchart Penentuan halangan dan celah pada pengambilan data depth kinect

Flowchart pada gambar 3.21 menjelaskan tentang pengolahan data *depth* kinect untuk menentukan apakah ada celah di sebelah kiri robot, terdapat halangan di depan robot, atau tidak keduanya sehingga robot dapat maju lurus.

1. Data depth diambil dari kinect menggunakan komunikasi USB melalui unit proses.
2. Data depth memiliki bentuk berupa matriks 640 x 480 1 kanal. Matriks ini berisi data kedalaman yang ada di depan kamera kinect. Frame tersebut dibagi menjadi 3 bagian dengan ukuran yang sama, yaitu region 1, region 2, dan region 3.
3. Nilai jarak di dalam setiap region dihitung nilai rata-ratanya. Nilai rata-rata disimpan pada variabel `jarak_region_1`, `jarak_region_2`, dan `jarak_region_3`.
4. `Jarak_region_3` merepresentasikan daerah di depan robot. Jika nilai pada variabel ini dibawah 1000, hal ini menunjukkan bahwa terdapat halangan sehingga robot perlu mengubah gerakannya menjadi belok kanan.
5. `Jarak_region_1` merepresentasikan daerah di sebelah kiri robot. Jika selisih `jarak_region_3` dan `jarak_region_1` kurang dari 100 maka di sebelah kiri robot tidak ada dinding sehingga robot perlu mengubah gerakannya menjadi belok kiri.
6. Jika kondisi pada poin 5 dan poin 6 tidak terpenuhi berarti robot berada dekat dengan dinding kiri dan di depan robot tidak ada halangan sehingga robot hanya perlu bergerak lurus saja.

#### *3.3.3.2 Pengambilan Data Depth Kinect untuk Pembuatan Peta Lingkungan*

**Gambar 3. 22** Pengolahan data depth untuk peta

Pengambilan data *depth* untuk pembuatan peta dilakukan dengan mengkonversi data *depth*. Data *depth* perlu dikonversi terlebih dahulu sehingga data *depth* yang diambil berada pada posisi yang sejajar dengan kamera kinect, sehingga. Pada setiap baris *frame depth* yang telah dikonversi dan masuk dalam range yang telah ditentukan akan diambil satu nilai yang paling kecil dan yang tidak bernilai 0. Hasil akhir pengolahan konversi data *depth* yang digunakan dalam pembuatan peta berupa data array dengan ukuran 640x1. Data tersebut yang akan digunakan untuk penggambaran peta oleh unit proses.

### 3.3.4 Pengambilan data posisi dan arah

Selama robot bergerak, posisi dan arah robot $(x,y,\theta)$ pada bidang kartesian akan selalu berubah. Posisi dan arah robot diketahui melalui sensor posisi dan arah yang dimiliki. Mengenai cara pembacaan posisi dan arah dengan sensor akan dijelaskan pada bagian mikrokontroler. Posisi dan arah diterima oleh unit proses melalui komunikasi usb dari mikrokontroler arduino.

### 3.3.5 Kontrol Gerakan

Pada unit proses, gerakan robot ditentukan berdasarkan hasil gerak berdasarkan kondisi robot. Terdapat 4 gerakan dasar robot, yaitu putar kanan, putar kiri, diam, dan maju. Untuk gerakan robot putar kanan, putar kiri, dan diam, gerakan robot menggunakan kecepatan yang statis. Sedangkan untuk gerakan robot maju, besar sinyal *Pulse Width Modulation* (PWM) yang akan dibangkitkan oleh mikrokontroler. Jarak dari dinding digunakan sebagai parameter kontrol gerakan robot. Kontrol yang digunakan adalah kontrol proporsional. Jarak actual dari dinding dan nilai jarak set point dibandingkan untuk menentukan besar nilai PWM.

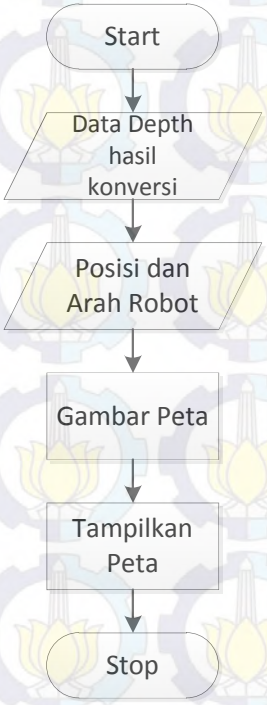
### 3.3.6 Pengiriman Data Gerakan

Kemudian pergerakan *mobile robot* dikontrol, sehingga *mobile robot* akan bergerak. Pengaturan bentuk pergerakan *mobile robot* dilakukan dengan mengirim data ke mikrokontroler yang nantinya mikrontroler yang akan mengendalikan *electronic speed control* motor untuk mengatur pergerakan masing-masing motor.

**Tabel 3. 2** Pengiriman Data Gerakan Robot

| Data | Gerakan |
|---|---|
| axxxxyyyy | Maju<br>(xxxx = pwm motor kanan, yyyy = pwm motor kiri) |
| i | Putar kiri |
| k | Putar kanan |
| d | Diam |

### 3.3.3 Perancangan Program Pembuatan Peta

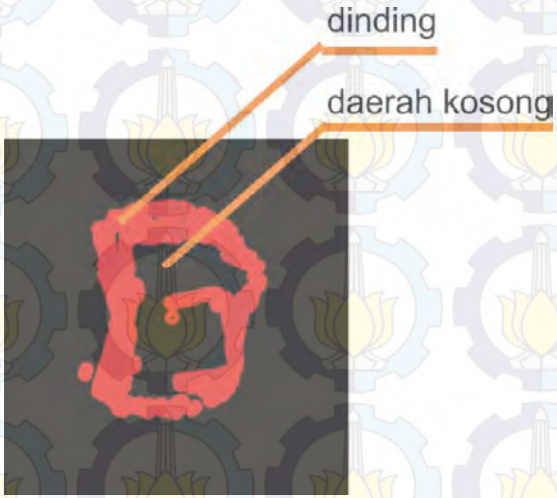



**Gambar 3. 23** Flowchart Pembuatan Peta

Pada saat robot mobile bergerak maka sensor kinect akan membaca jarak obyek disekitar mobile robot sedangkan sistem rotari enkoder dan kompas akan memberikan posisi dan arah robot (x,y, $\theta$) terhadap bidang kartesian secara real time. Data depth hasil konversi merupakan data *depth* yang memiliki posisi di bawah bidang kamera kinect dalam sumbu y dengan kamera kinect sebagai titik pusat.

(a) (b) (c)

**Gambar 3. 24** (a) visualisasi gambar *depth* oleh kamera kinect, (b) hasil konversi data *depth*, dan (c) hasil visualisasi *depth* sebagai peta




**Gambar 3. 25** Peta yang digambar

Untuk menggambar peta seperti pada gambar 3.25, digunakan fungsi dari OpenCV untuk menyimpan dan menampilkan peta dalam format citra. Dengan menggunakan persamaan trigonometri yang sudah dijelaskan pada bab sebelumnya, berikut merupakan kode yang digunakan untuk menggambar peta tersebut.

```
for( int x = 0 ; x < 640 ; x++ ){
if (jarak[x] != 0 )
{
jarak[x]=jarak[x]/10/2;
p_xpos=(500+p2_xpos+(jarak[x]*sin((sudut_buff-45)
       *PI/180.0)));
p_ypos=(500+p2_ypos+(jarak[x]*cos((sudut_buff-45)
       *PI/180.0)));
sudut_buff += 0.089 ;
cvCircle(Peta_dinding,cvPoint((int)(p_xpos),(int)(p_ypos)),
7, CV_RGB(255,0,0),8,0);
}
}
```

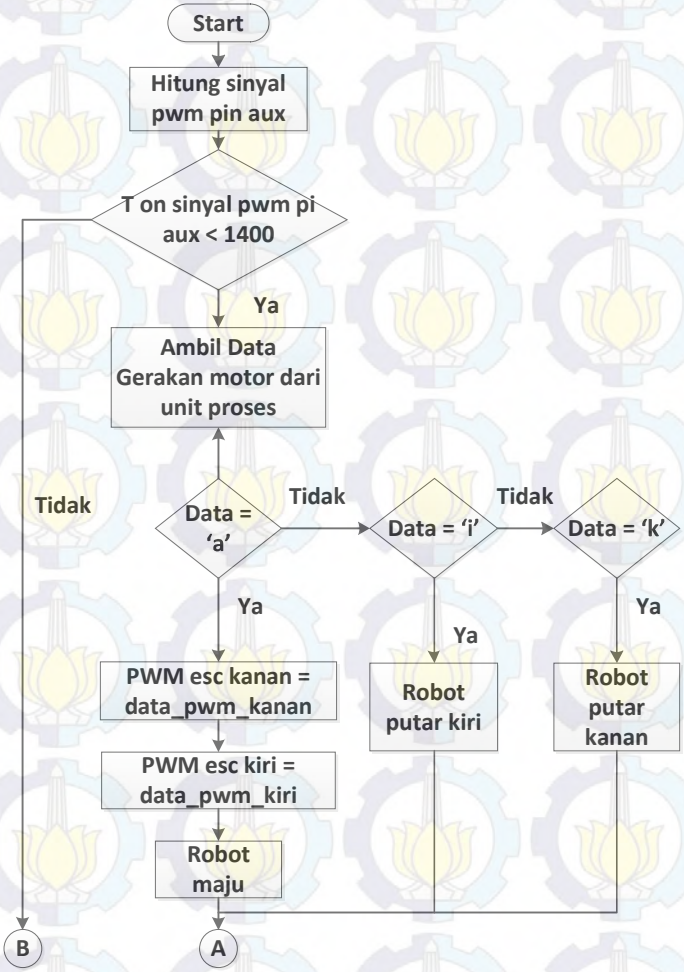
`P2_xpos` dan `p2_ypos` adalah variabel yang menyimpan posisi aktual robot pada saat peta digambar. `Jarak[x]` adalah variabel yang menyimpan data depth hasil konversi untuk peta.Pada penggambaran peta nilai – nilai variabel jarak terukur oleh kinect, posisi x dan y robot pada bidang kartesian, dan arah hadap robot menjadi sangat penting. Kesalahan pada parameter – parameter tersebut akan mengakibatkan peta tidak tergambar sempurna.

### 3.3.4 Perancangan Setting Program pada Remote Kontrol

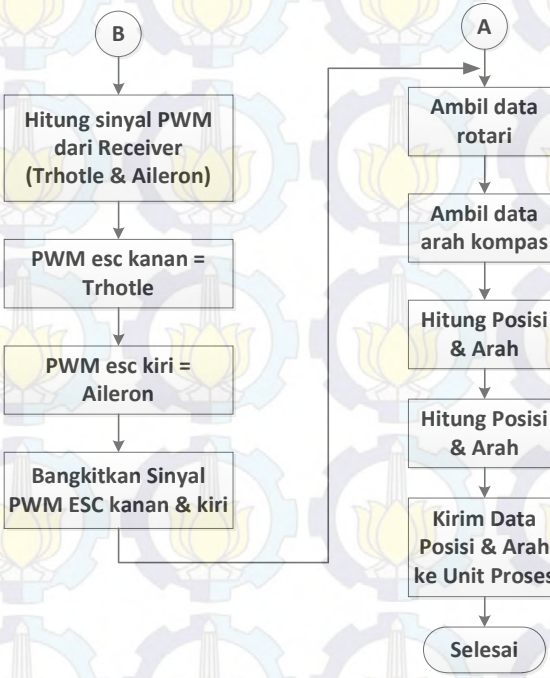
Remote kontrol yang digunakan adalah remote kontrol Turnigy 9x. Seperti pada datasheetnya, remote kontrol ini dapat diprogram sesuai dengan kebutuhan pengguna. Karena kebutuhan pada tugas akhir ini adalah untuk mengatur pergerakan robot. Bentuk robot yang digunakan seperti tank. Model mekanik robot tank sama dengan mekanik robot diferensial. Robot bergerak dengan mengatur kecepatan dua motor. Pada tugas akhir ini disetiing agar satu joystick remote kontrol dapat mengatur kecepatan dua motor secara bersamaan.

Pada menu setting, dipilih menu elevon. Selanjutnya nilai ail1 disetting 100 100. Nilai ail2 disetting 100 100. Nilai ele1 disetting -100. Nilai ele2 disetting 100. Dengan mengatur menu seperti nilai tersebut, robot dapat dikendalikan menggunakan satu joystick saja.

### 3.3.5 Perancangan Program pada Mikrokontroler




**Gambar 3. 26** Flowchart program pada mikrokontroler




**Lanjutan Gambar 3.26** Flowchart program pada mikrokontroler

### *3.3.5.1 Pemilihan Mode*

Pemilihan mode otomatis atau manual dilakukan dengan membaca sinyal PWM yang dihasilkan oleh pin aux pada receiver. Pin aux akan menghasilkan sinyal dengan frekuensi 50 Hz dan Ton antara 1ms dan 2ms. Ketika saklar auto/manual diaktifkan maka pin aux akan menghasilkan sinyal dengan Ton lebih dari 1,4 ms, dan ketika saklar auto/manual dinonaktifkan maka pin aux akan menghasilkan sinyal dengan Ton kurang dari 1,4 ms. Berikut adalah kode yang digunakan untuk membaca sinyal dari pin aux.

```
ken = pulseIn(4, HIGH, 25000);
```

### *3.3.5.2 Mode Manual*

Mode manual adalah mode dimana *user* mengkontrol gerakan robot melalui remote. Kecepatan motor bergantung pada joystick 2 pada remote. Untuk itu, mikrokontroler perlu membaca sinyal pwm dari receiver. Sesuai dengan konfigurasi hardware, pin trho menghasilkan sinyal untuk motor kanan, dan pin aile menghasilkan sinyal untuk motor kiri. Sinyal yang sudah dibaca selanjutnya perlu dibangkitkan kembali sesuai dengan sinyal yang dihasilkan remote. Dikarenakan pembacaan sinyal pada mikrokontroler mengalami delay sehingga perlu sinyal perlu dikalibrasi. Selanjutnya dibangkitkan sinyal dengan frekuensi sinyal yang sama sesuai dengan sinyal yang sudah dikalibrasi tersebut. Berikut adalah kode yang digunakan untuk membaca sinyal dari pin trho dan pin aile.

```
tro = pulseIn(5, HIGH, 25000);
ail = pulseIn(6, HIGH, 25000);
tro=(map(tro, 1039, 1804, 1080, 1750));
ail=(map(ail, 1066, 1857, 1080, 1750));
motor_kanan.writeMicroseconds(tro);
motor_kiri.writeMicroseconds(ail);
```

### *3.3.5.3 Mode Auto*

Mode auto adalah mode dimana unit proses mengkontrol penuh gerakan robot. Pergerakan robot sesuai dengan data yang diterima mikrokontroler dari unit proses. Setiap data yang masuk disimpan pada variabel data. Jika data yang diterima adalah ‘a’ maka robot akan maju dengan kecepatan tertentu. Pada paket data yang sama terdapat data pwm untuk motor kanan dan motor kiri. Mikrokontroler akan membangkitkan sinyal pwm dengan data tersebut. Jika data yang diterima ‘k’, maka robot akan bergerak rotasi ke kanan. Jika data yang diterima ‘i’, maka robot akan bergerak rotasi ke kiri.

### *3.3.5.4 Ambil data rotary*

Setiap motor bergerak maju maupun mundur rotari enkoder akan mengirimkan sinyal pwm. Semakin cepat motor bergerak maka frekuensi sinyal akan semakin cepat. Pergerakan motor dan track robot sebanding, sehingga ketika track bergerak maka rotari enkoder akan menghasilkan sinyal pula. Sinyal dari rotari enkoder akan dihitung setiap

waktunya sehingga diketahui jarak yang telah ditempuh oleh robot. Sinyal dari rotary enkoder dibaca menggunakan interrupt eksternal pada mikrokontroler arduino yang digunakan. Untuk mengaktifasi pin interrupt pada mikrokontroler maka digunakan kode sebagai berikut.

```
pinMode(B_ENCODER_RIGHT, INPUT);
pinMode(B_ENCODER_LEFT, INPUT);
attachInterrupt(0, count_left, FALLING);
attachInterrupt(1, count_right, FALLING);
```

Dikarenakan terdapat 2 rotari enkoder maka digunakan 2 pin interrupt. Ketika terdapat transisi sinyal dari HIGH ke LOW pada pin interrupt, maka mikrokontroler akan menginterupsi program utama dan masuk ke fungsi interrupt yang dibuat. Fungsi interrupt yang dibuat adalah untuk menghitung jumlah sinyal yang masuk. Berikut adalah kode pada fungsi interrupt yang dibuat.

```
void count_left(){
B_ENCODER_LEFT_CONDITION=
digitalRead(B_ENCODER_LEFT);
if(B_ENCODER_LEFT_CONDITION==HIGH){
      COUNT_LEFT_POSITION++;}
else if( B_ENCODER_LEFT_CONDITION==LOW){
     COUNT_LEFT_POSITION--;}}
void count_right(){
B_ENCODER_RIGHT_CONDITION=
digitalRead(B_ENCODER_RIGHT);
if(B_ENCODER_RIGHT_CONDITION==LOW){
COUNT_RIGHT_POSITION++;}
else if(B_ENCODER_RIGHT_CONDITION==HIGH){
COUNT_RIGHT_POSITION--;}
```

Ketika fungsi interrupt teraktifasi maka variabel `COUNT_RIGHT_POSITION` akan ditambah jika sinyal dari kanal B rotari enkoder kanan bernilai LOW dan akan dikurangi jika sinyal dari kanal B rotari enkoder kanan bernilai HIGH. Hal yang sama akan dilakukan pada rotari enkoder kiri, namun `COUNT_LEFT_POSITION` akan ditambah jika sinyal dari kanal B rotari enkoder kiri bernilai LOW dan akan ditambah jika sinyal dari kanal B rotari enkoder kiri bernilai LOW dan

akan ditambah jika sinyal dari kanal B rotari enkoder kiri bernilai HIGH. Hal ini dikarenakan posisi rotari enkoder yang tidak sejajar.

### *3.3.5.5 Ambil data kompas*

Pengambilan data kompas, dilakukan dengan menggunakan komunikasi I2C antara mikrokontroler dan sensor kompas CMPS11. Data yang diterima adalah estimasti arah robot oleh kompas dalam satuan derajat. IDE arduino telah menyediakan library untuk mengakses data dari modul kompas CMPS11.

```
theta=compass.bearing();
```

### *3.3.5.6 Hitung Posisi robot*

Dengan menghitung sinyal masuk dari rotari enkoder, maka jarak yang ditempuh oleh robot dapat diketahui. Berdasarkan pengujian , setiap robot bergerak sejauh 1 cm maka terdapat 201 interupt dari rotari enkoder. Setiap interrupt yang terjadi maka variabel *tick* akan bertambah satu. Maka digunakan persamaan berikut untuk menghitung jarak tempuh robot berdasarkan pembacaan sinyal dari rotari enkoder.

$$\Delta S = \frac{tick}{201} \tag{12}$$

Dengan menggunakan jarak tempuh ($\Delta \boldsymbol{S}$) untuk roda kanan dan roda kiri dan arah robot yang berasal dari kompas. Robot dapat mengetahui posisi dan arah. Untuk mengetahui arah dan posisi, sebenarnya dapat menggunakan pembacaan tick dari rotary enkoder saja, namun dikarenakan drift, yaitu kejadian saat robot tidak bergerak, sedangkan roda robot telah bergerak dapat menyebabkan error terhadap arah hadap robot sehingga penentuan arah menggunakan kompas CMPS11. Untuk perhitungan posisi, berikut merupakan kode yang digunakan.

```
left = COUNT_LEFT/LEFT_CLICKS_PER_INCH;
right=COUNT_RIGHT/RIGHT_CLICKS_PER_INCH;
in = (left + right) / 2.0;
Y_pos += in * cos(theta * PI/180 );
X_pos += in * sin(theta * PI/180 );
X_POS = X_pos;
Y_POS = Y_pos;
```

### *3.3.5.7 Kirim data posisi, arah, dan mode robot*

Posisi, arah, dan mode robot perlu dikirim ke unit proses. Posisi dan arah ini akan digunakan pada unit proses untuk membuat peta. Sedangkan mode dikirm ke unit proses, agar unit proses mengetahui kondisi mode robot. Berikut merupakan kode yang digunakan untuk mengirim posisi, arah, dan mode.

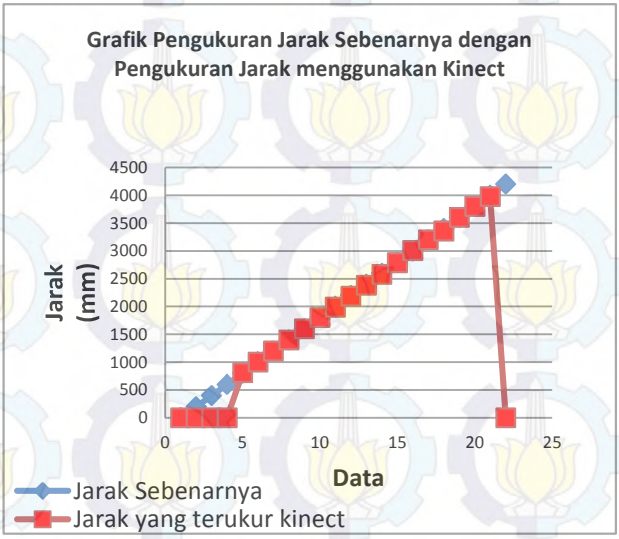
```
sprintf(b,"q%d%d%d%d",X_POS, Y_POS, SUDUT,MODE);
Serial.println(b);
```

# BAB IV
# PENGUJIAN

Pada bab ini akan dibahas mengenai pengujian dari sistem yang telah dirancang untuk mengetahui kinerja dari sistem. Pengujian pada bab ini terdiri dari pengujian pengukuran jarak menggunakan kinect, pengujian jarak kendali robot menggunakan remote kontrol, pengujian pembacaan arah hadap robot, pengujian pembacaan posisi robot, pengujian kemampuan robot dalam mengikuti dinding, dan pengujian kemampuan robot dalam menelusuri ruangan berdasarkan waktu dan berdasarkan daerah yang dilewati.

## 4.1 Pengujian Pembacaan Jarak Obyek dengan Kinect

Pengujian yang dilakukan pada bagian ini adalah untuk mengetahui karakteristik pembacaan jarak estimasi yang dilakukan oleh kinect. Pengujian dilakukan dengan mencatat hasil pembacaan jarak terhadap obyek oleh kinect pada pusat frame *depth*. Pencatatan dilakukan pada jarak 20 cm hingga 4200 cm dari kamera kinect dengan interval 20 cm. Berikut adalah pengujian yang telah dilakukan :

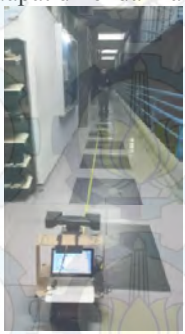



**Gambar 4. 1** Grafik Pengukuran Jarak Sebenarnya dengan Pengukuran Jarak menggunakan Kinect

Dari data pengujian tersebut dapat diketahui bahwa akurasi pengukuran jarak oleh kinect pada jarak kurang dari 80 cm dan lebih dari 4 meter, pengukuran jarak memiliki error rata-rata sebesar 100 %. Sedangkan pada jarak antara lebih dari 80 cm hingga kurang dari 4 meter pengukuran jarak memiliki error rata-rata sebesar 0.197 %. Berdasarkan pengujian ini dapat diketahui bahwa pengukuran jarak menggunakan kamera kinect baik untuk jarak pada range 80 cm hingga 4 meter.

## 4.2 Pengujian Jarak Kendali Robot Menggunakan Remote Kontrol

Robot *mobile* yang didesain akan digunakan oleh *user* dari jarak jauh. Penggunaan jarak jauh bertujuan agar *user* berada pada lokasi aman dari lokasi pencarian. Pengujian ini dilakukan untuk mengetahui kemampuan jarak antara *user* dengan robot menggunakan remote kontrol yang digunakan. Pengujian dilakukan dengan cara mengendalikan robot secara manual pada jarak-jarak tertentu dan diperhatikan apakah robot masih dapat dikendalikan atau tidak.



**Gambar 4. 2** Pengujian jarak dalam pengendalian robot menggunakan remote kontrol

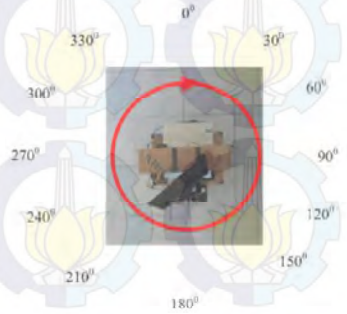
**Tabel 4. 1** Jarak uji dan tingkat keberhasilan kendali robot secara manual

| Jarak (Meter) | Kendali robot manual |
|---|---|
| 7 | Berhasil |
| 14 | Berhasil |
| 21 | Berhasil |
| 28 | Berhasil |
| 35 | Berhasil |
| 42 | Berhasil |
| 49 | Berhasil |

Menurut NCTC (*The National Counter Terrorism Center*), jarak aman bagi petugas yang berkepentingan ketika menangani bahan berbahaya seperti bahan peledak adalah 21 meter untuk bom pipa, 34 meter untuk bom rompi, dan 46 meter untuk bom koper. Sesuai dengan hasil pengujian, robot mampu dikendalikan oleh *user* dari jarak yang aman dari lokasi robot.

## 4.3 Pengujian Pembacaan Arah Robot dengan Sensor Kompas

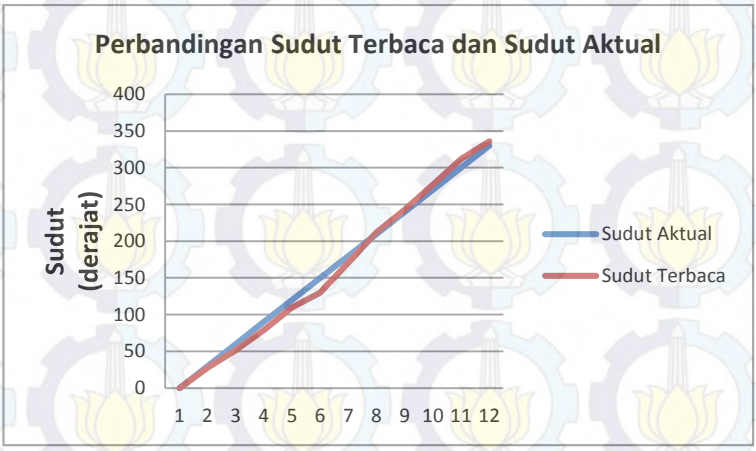
Pada saat robot melakukan perhitungan odometri, yaitu mengetahui posisi robot, robot perlu mengetahui arah hadap robot. Pada pengujian ini, dilakukan analisa terhadap akurasi pembacaan arah oleh kompas dan perbandingan dengan arah hadap aktual robot. Pengujian dilakukan dengan cara memutar robot satu putaran penuh pada posisi yang sama.




**Gambar 4. 3** Pengujian Akurasi Pembacaan Arah Hadap Robot

**Tabel 4. 2** Tabel Perbandingan Arah Sudut Sensor Kompas dengan Arah Sudut Aktual Robot

| Sudut Aktual | Sudut Terbaca | Error (%) |
|---|---|---|
| $0^0$ | $0^0$ | 0 |
| $30^0$ | $28^0$ | 6.6666667 |
| $60^0$ | $52^0$ | 13.3333333 |
| $90^0$ | $79^0$ | 12.2222222 |
| $120^0$ | $110^0$ | 8.3333333 |
| $150^0$ | $130^0$ | 13.3333333 |
| $180^0$ | $171^0$ | 5 |
| $210^0$ | $212^0$ | -0.952381 |
| $240^0$ | $243^0$ | -1.25 |
| $270^0$ | $277^0$ | -2.5925926 |
| $300^0$ | $312^0$ | -4 |
| $330^0$ | $336^0$ | -1.8181818 |




**Gambar 4. 4** Grafik Perbandingan Arah Sudut Sensor Kompas dengan Arah Sudut Aktual Robot

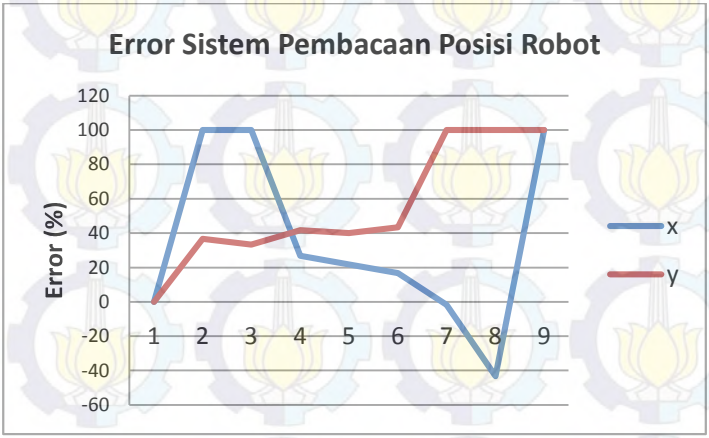
Dari percobaan didapatkan hasil pembacaan sudut arah hadap robot menggunakan sensor kompas CMPS11. Hasil dari percobaan tersebut diketahui bahwa pembacaan arah hadap robot menggunakan sensor kompas CMPS11 memiliki error rata-rata yaitu 4,023%.

## 4.4 Pengujian Pembacaan Posisi Robot menggunakan Rotari Enkoder dan Sensor Kompas

Pengujian sistem pembacaan posisi ini dilakukan untuk mengetahui besarnya nilai presentase error pada sistem pembacaan posisi menggunakan rotari enkoder dan sensor kompas. Nilai yang diukur pada pengujian ini adalah posisi x dan y aktual dan posisi x dan y hasil perhitungan sistem pada bidang kartesian.

**Tabel 4. 3** Pengukuran posisi x dan posisi y menggunakan rotari enkoder dan kompas

| No | Posisi Aktual (cm) | | Posisi Perhitungan (cm) | | Error (%) | |
|---|---|---|---|---|---|---|
| | x | y | x | y | x | y |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 30 | 3 | 19 | 100 | 36.67 |
| 3 | 0 | 60 | 0 | 40 | 100 | 33.33 |
| 4 | 30 | 60 | 22 | 35 | 26.67 | 41.67 |
| 5 | 60 | 60 | 47 | 36 | 21.67 | 40 |
| 6 | 60 | 30 | 50 | 17 | 16.67 | 43.33 |
| 7 | 60 | 0 | 61 | 4 | -1.67 | 100 |
| 8 | 30 | 0 | 43 | 15 | -43.33 | 100 |
| 9 | 0 | 0 | 26 | 20 | 100 | 100 |

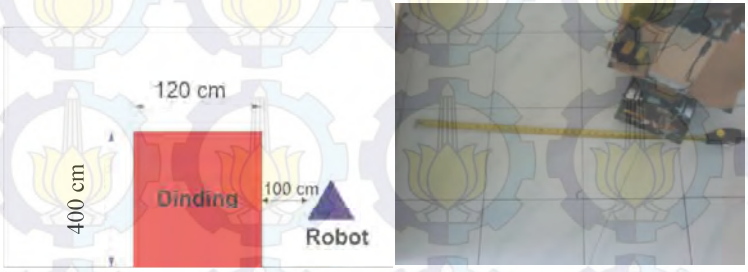



**Gambar 4. 5** Grafik Error Sistem Pembacaan Posisi Robot Menggunakan Rotari Enkoder dan Sensor Kompas

Pengujian ini dilakukan dengan menggerakkan robot secara manual menggunakan remote kontrol. Posisi robot dan nilai pembacaan sistem diukur secara langsung. Dari hasil pengukuran posisi x dan y sistem pembacaan posisi pada bidang kartesian seperti pada tabel 4.3, pada pengujian dengan nilai posisi x maupun y dengan nilai 0, sistem memiliki error sebesar 100%. Sedangkan untuk nilai posisi x maupun y yang tidak bernilai aktual 0, sistem memiliki error absolut dibawah 44%.

## 4.5 Pengujian Kemampuan Robot dalam Mengikuti Dinding

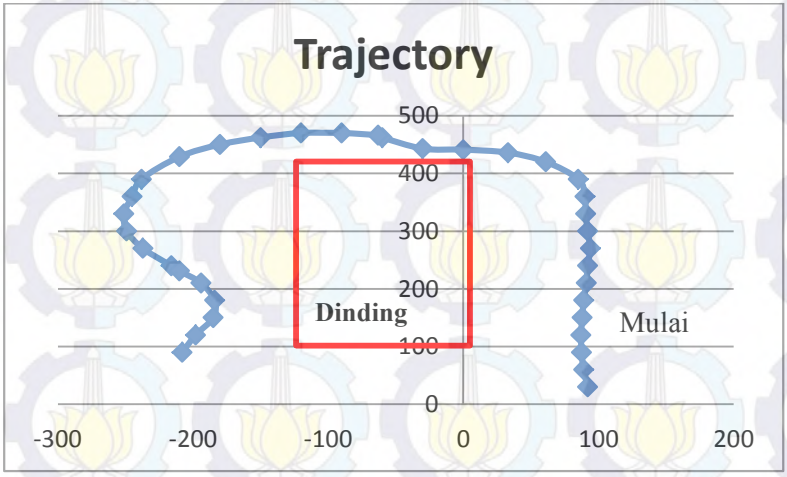
Salah satu dasar teori yang digunakan dalam sistem ini adalah *wall following*. Untuk itu dilakukan pengukuran jarak robot dari dinding ketika robot dalam kondisi berjalan otomatis. Pengukuran ini bertujuan untuk mengetahui kemampuan gerakan robot sesuai dengan teori yang digunakan. Pengujian dilakukan dengan cara menjalankan robot secara otomatis pada halangan yang dianggap sebagai dinding.




(a) (b)

**Gambar 4. 6** (a) Pengujian 1 Robot dalam Menelusuri Dinding, (b) Pengukuran Jarak dari Dinding mengunakan Penggaris

Gambar 4.6 menunjukkan pengujian dan pengambilan data kemampuan robot untuk menelusuri dinding dalam pengujian 1. Robot akan mengikuti dinding yang ada di sebelah kiri. Jarak robot dari dinding diukur menggunakan penggaris sehinga dapat diketahui jarak robot dari dinding. Gambar 4.7 adalah hasil *trajectory* robot ketika robot mengikuti dinding.



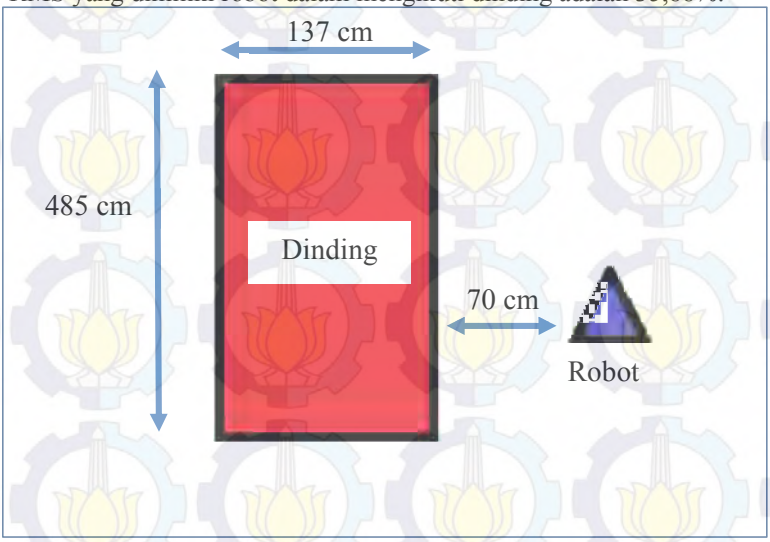

**Gambar 4. 7** Trayektori Robot dalam mengikuti dinding

**Tabel 4. 4** Tabel Pengukuran Jarak Robot dari Dinding

| N | Jarak dari Dinding (cm) | Set Point (cm) | Error (%) |
|---|---|---|---|
| 1 | 92 | 108 | -14.81481481 |
| 2 | 89 | 108 | -17.59259259 |
| 3 | 87.5 | 108 | -18.98148148 |
| 4 | 87 | 108 | -19.44444444 |
| 5 | 88 | 108 | -18.51851852 |
| 6 | 89 | 108 | -17.59259259 |
| 7 | 91 | 108 | -15.74074074 |
| 8 | 92 | 108 | -14.81481481 |
| 9 | 94 | 108 | -12.96296296 |
| 10 | 92 | 108 | -14.81481481 |
| 11 | 90.5 | 108 | -16.2037037 |
| 12 | 90 | 108 | -16.66666667 |
| 13 | 85 | 108 | -21.2962963 |
| 14 | 61 | 108 | -43.51851852 |
| 15 | 33 | 108 | -69.44444444 |
| 16 | 21 | 108 | -80.55555556 |
| 17 | 24 | 108 | -77.77777778 |
| 18 | 42 | 108 | -61.11111111 |
| 19 | 47 | 108 | -56.48148148 |
| 20 | 51 | 108 | -52.77777778 |
| 21 | 50 | 108 | -53.7037037 |
| 22 | 51 | 108 | -52.77777778 |
| 23 | 77 | 108 | -28.7037037 |
| 24 | 95 | 108 | -12.03703704 |
| 25 | 118 | 108 | 9.259259259 |
| 26 | 124 | 108 | 14.81481481 |
| 27 | 132 | 108 | 22.22222222 |

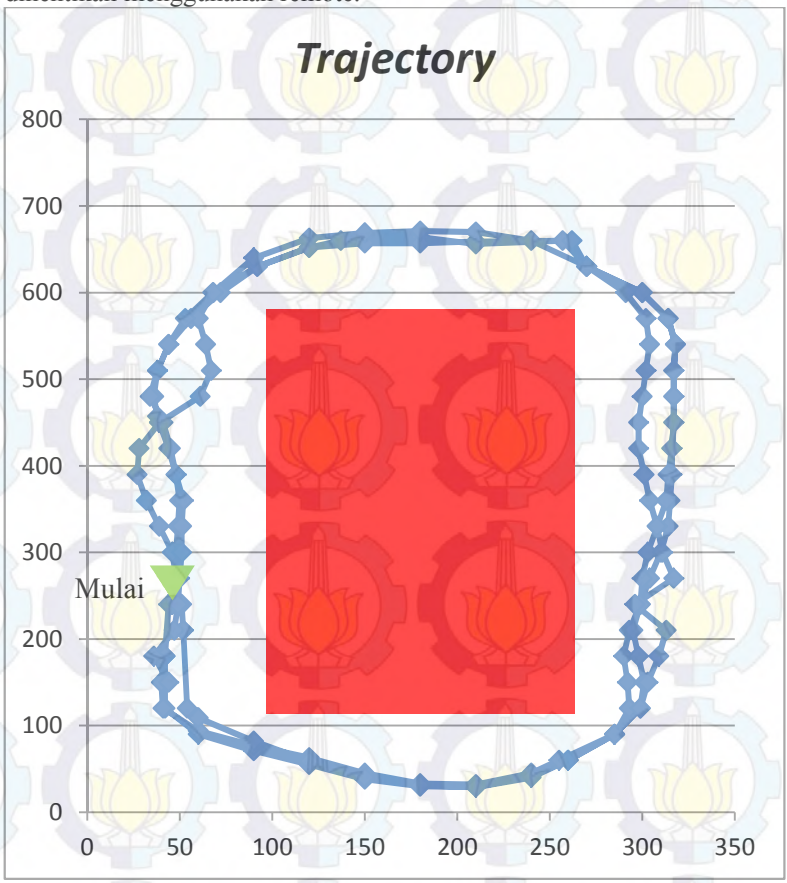
| | | | |
|---|---|---|---|
| 28 | 128 | 108 | 18.51851852 |
| 29 | 117 | 108 | 8.333333333 |
| 30 | 96 | 108 | -11.11111111 |
| 31 | 92 | 108 | -14.81481481 |
| 32 | 73 | 108 | -32.40740741 |
| 33 | 63 | 108 | -41.66666667 |
| 34 | 66 | 108 | -38.88888889 |
| 35 | 79 | 108 | -26.85185185 |
| 36 | 86 | 108 | -20.37037037 |

Dari tabel 4.4, dapat dianalisa bahwa robot selalu menjaga jarak dari dinding. Ketika robot mengestimasi bahwa tidak ada dinding di sebelah kiri maka robot akan segera mengubah gerakannya untuk belok ke kiri. Dengan set point jarak dari dinding 108 cm, robot akan menjaga jarak dari dinding dengan rata-rata jarak 80,36 cm. Sedangkan error RMS yang dimiliki robot dalam mengikuti dinding adalah 35,88%.

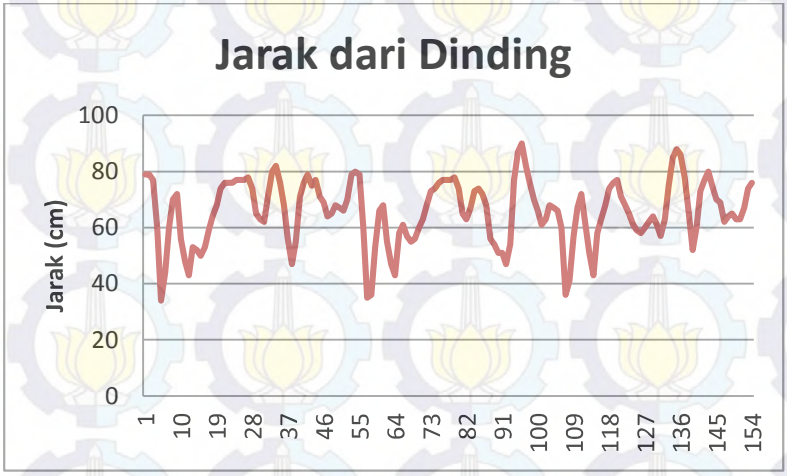



**Gambar 4. 8** Pengujian Jarak dari dinding dalam pengujian 2

Selanjutnya dilakukan pengujian 2. Gambar 4.8 menunjukkan ruang pengujian. Pada pengujian ini, robot akan mengikuti dinding yang ada di sebelah kiri sehingga robot akan terus bergerak mengelilingi ruangan dengan acuan dinding di sebelah kiri. Robot akan terus bergerak mengikuti dinding hingga robot diberi intruksi berhenti oleh pengguna. Pada gambar 4.9, ditunjukkan *trajectory* robot selama mengikuti dinding. Robot terus bergerak mendekati dinding. Dalam pengujian ini robot bergerak memutari dinding sebanyak 3 kali, selanjutnya robot dihentikan menggunakan remote.




**Gambar 4. 9** *Trajectory* Robot dalam mengikuti dinding




**Gambar 4. 10** Jarak dari dinding dalam pengujian 2

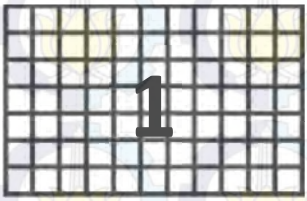
Pada pengujian 2 robot bergerak dengan mengikuti dinding di sebelah kiri. Ketika terdapat tikungan, maka robot segera bergerak ke kiri. Ketika robot terlalu mendekat ke dinding, maka robot akan segera bergerak menjauh dari robot. Setpoint jarak dalam pengujian 2 diatur menjadi 85 cm. Pada saat robot menelusuri dinding di sebelah kiri, robot menjaga jarak dari dinding dengan jarak rata-rata sebesar 65,73 cm. Robot memiliki error rata – rata sebesar 26,3 %. Dari kedua pengujian tersebut diketahui bahwa error rata - rata robot dalam menjaga jarak dengan dinding kiri dengan set point jarak sebesar 31,09 %.

## 4.6 Pengujian Kemampuan Robot dalam Menelusuri Ruangan

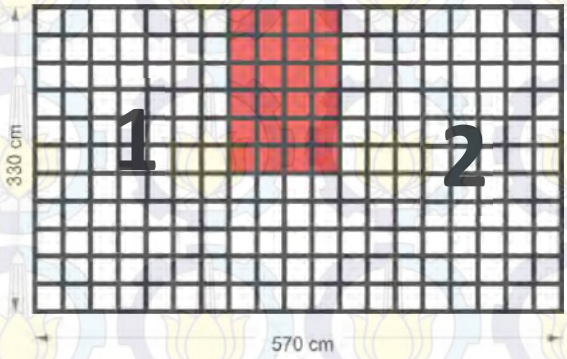
Salah satu tujuan perancangan sistem pada tugas akhir ini adalah agar robot mampu menelusuri ruangan, sehingga perlu dilakukan pengujian untuk mengetahui sejauh mana kemampuan robot dalam menelusuri ruangan. Ruangan akan diberi halangan sehingga ruangan terbagi menjadi beberapa bagian. Pembagian jumlah ruang dilakukan menggunakan sekat sebagai halangan dan pemisah. Robot akan dibiarkan berjalan secara otomatis dan akan dicatat waktu yang dibutuhkan robot untuk memasuki masing-masing bagian ruang.

### 4.6.1 Pembuatan Ruang Uji

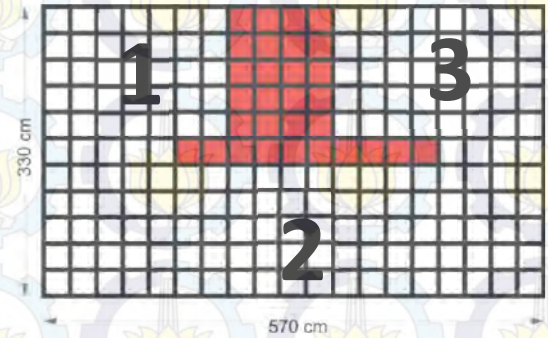
Ruangan akan diberi halangan sehingga ruangan terbagi menjadi beberapa bagian. Pembagian jumlah ruang dilakukan menggunakan sekat. Pada gambar 4.2 merupakan desain ruang yang diujikan. Daerah putih menunjukkan daerah kosong sedangkan daerah merah menunjukkan bagian ruangan yang diberi halangan. Setiap sekat akan membagi ruangan menjadi beberapa bagian. Dalam pengujian, setiap sel ruangan, yaitu kotak-kotak kecil pada setiap desain ruangan, memiliki ukuran 30 cm x 30 cm.



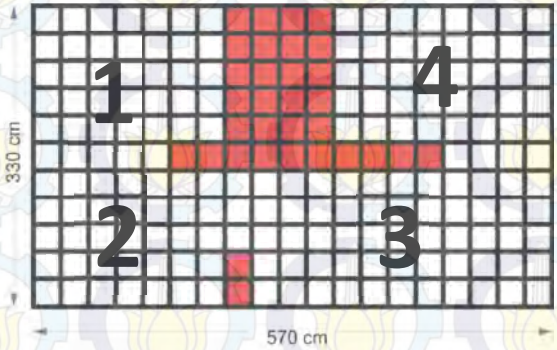

(a)



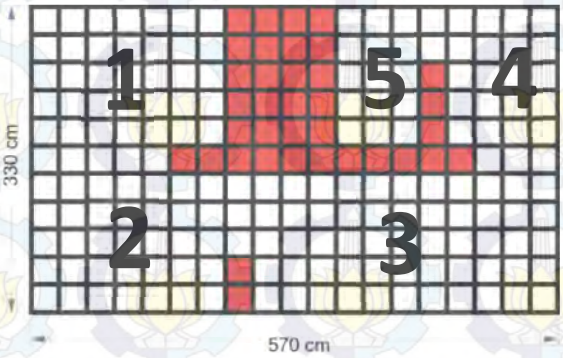

(b)



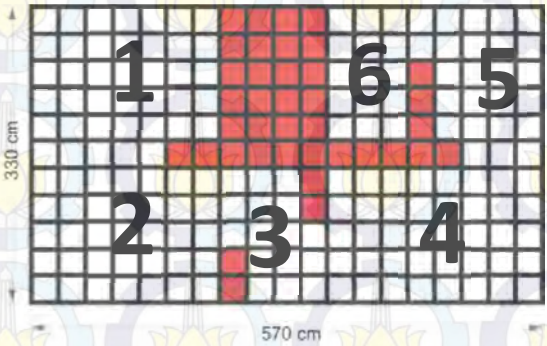

(c)




(d)




(e)



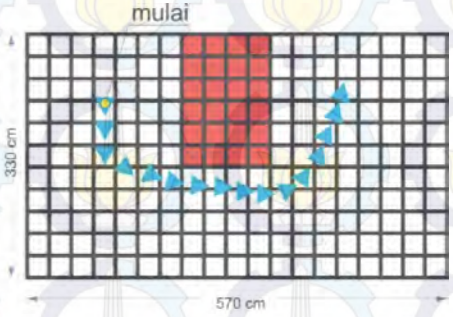

(f)

**Gambar 4. 11** Desain Ruang Uji : (a) Ruang tanpa pembagian, (b) Ruang dengan 2 pembagian, (c) Ruang dengan 3 pembagian, (d) Ruang dengan 4 pembagian , (e) Ruang dengan 5 pembagian, (f) Ruang dengan 6 pembagian

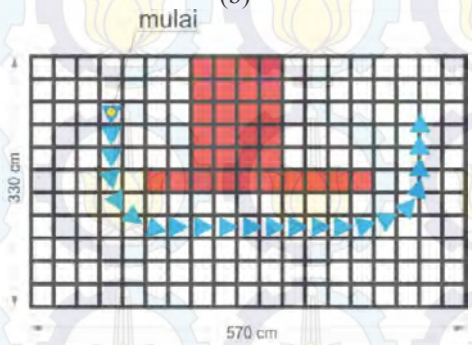
### 4.6.2 Pengujian Kemampuan Telusur Robot

Dengan menggunakan metode yang telah dijelaskan pada bagian perancangan sistem maka dilakukan pengujian berupa pencatatan waktu dan presentase kemampuan robot dalam memasuki bagian dalam ruangan. Pencatatan dihentikan ketika robot telah memasuki semua bagian ruangan atau robot tidak bias memasuki bagian ruangan. Ruang yang digunakan pada pengujian ini menggunakan ruang uji yang didesain pada bagian sebelumnya.

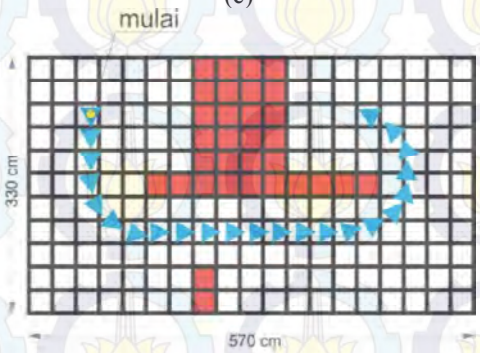

(a)



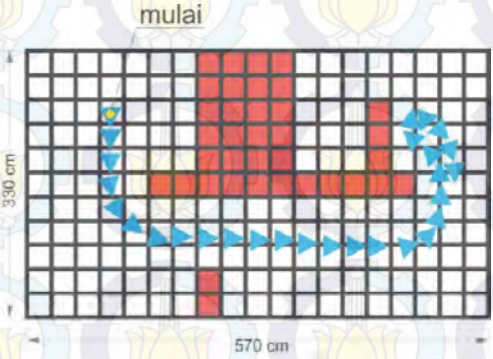

(b)



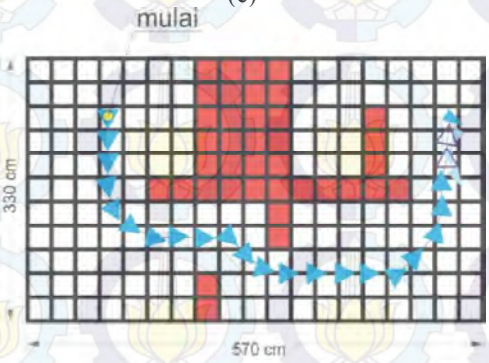

(c)




(d)




(e)




(f)

**Gambar 4. 12** Trajectory Robot dalam menelusuri (a) Ruang tanpa pembagian, (b) Ruang dengan 2 pembagian, (c) Ruang dengan 3 pembagian, (d) Ruang dengan 4 pembagian , (e) Ruang dengan 5 pembagian, (f) Ruang dengan 6 pembagian

**Tabel 4. 5** Hasil Pengujian Kemampuan Telusur Ruang oleh Robot

| Ruang | Bagian | Waktu Per Bagian (detik) | Total Ruang yang Dimasuki (%) |
|---|---|---|---|
| 1 | 1 | 0 | 100 |
| 2 | 1 | 0 | 100 |
| | 2 | 22 | |

| | | | |
|---|---|---|---|
| 3 | 1 | 0 | 100 |
| | 2 | 16 | |
| | 3 | 33 | |
| 4 | 1 | 0 | 100 |
| | 2 | 15 | |
| | 3 | 27 | |
| | 4 | 54 | |
| 5 | 1 | 0 | 80 |
| | 2 | 14 | |
| | 3 | 22 | |
| | 4 | 47 | |
| | 5 | - | |
| 6 | 1 | 0 | 83.33 |
| | 2 | 13 | |
| | 3 | 19 | |
| | 4 | 40 | |
| | 5 | 62 | |
| | 6 | - | |

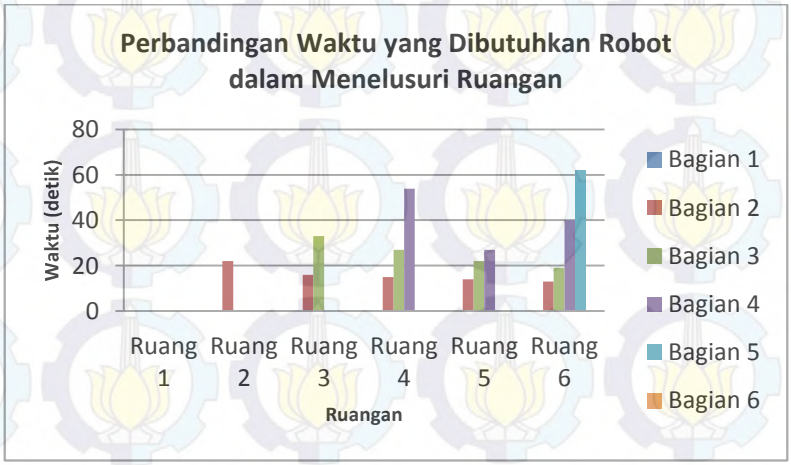
Gambar 4.12 merupakan *trajectory* robot selama robot bergerak menyusuri ruangan. Selama pengujian, robot berusaha mengikuti dinding di sebelah kiri robot. Selanjutnya dianalisa pula kemampuan jelajah robot. Kemampuan jelajah robot per bagian ruang menunjukkan presentase keberhasilan robot dalam menelusuri ruang yang ada. Dari data tersebut dapat diketahui hubungan antara jumlah ruang dengan waktu yang dibutuhkan robot untuk menelusuri seluruh ruangan. Pada kondisi ruangan yang terdiri dari 1 bagian hingga 4 bagian , sistem mampu menelusuri semua bagian. Namun pada ruangan dengan 5 bagian, presentase robot untuk menelusuri robot turun menjadi 80%. Presentase keberhasilan robot untuk memasuki bagian ruangan menjadi 83,33% pada ruangan dengan 6 pembagian.

Dari tabel 4.5, diketahui bahwa robot memiliki kemampuan dalam menjalajahi ruangan sesuai dengan ruang uji sebesar 92,67%. Error disebabkan ukuran ruangan yang semakin kecil sehingga ruang

gerak robot semakin tidak bebas. Sesuai dengan kemampuan pembacaan jarak oleh kamera kinect, robot memerlukan ruangan yang luas. Hal ini dikarenakan kinect hanya mampu membaca jarak antara 80 cm hingga 4 meter.

Pada grafik yang ditampilkan oleh gambar 4.13, ditampilkan waktu yang dibutuhkan oleh robot untuk memasuki ruangan. Setiap ruangan membutuhkan waktu yang berbeda untuk memasuki semua ruangan. Ruang 1 membutuhkan waktu 0 detik, ruang 2 membutuhkan waktu 22 detik, ruang 3 membutuhkan waktu 33 detik, dan ruang 4 membutuhkan waktu 54 detik untuk memasuki semua bagian ruangan. Sedangkan ruang 5 dan ruang 6, robot tidak dapat memasuki semua ruangan. Ruang 5, robot membutuhkan waktu 57 detik untuk memasuki 4 bagian ruangan. Dan ruang 6, robot membutuhkan waktu 62 detik untuk memasuki 5 bagian ruangan.

Dari data-data tersebut dapat dilihat bahwa penambahan bagian dalam suatu ruangan akan menambah waktu yang dibutuhkan robot untuk menelusuri ruangan. Semakin banyak pembagian daerah dalam suatu ruangan, maka waktu yang dibutuhkan oleh robot untuk penelusuran ruangan semakin lama. Dan jika semakin banyak halangan maka kemampuan robot untuk memasuki ruang akan semakin sedikit karena ukuran celah antar ruang yang semakin mengecil.




**Gambar 4. 13** Grafik Waktu Telusur per Ruangan

# BAB V
# PENUTUP

## Kesimpulan

Kesimpulan yang diperoleh pada tugas akhir ini adalah :

1. Dalam menelusuri ruangan, robot menjaga jarak dengan dinding sebelah kiri dengan rata – rata jarak dari dinding dalam set point jarak dari dinding sebesar 108 cm adalah 80,36 cm dengan error rata – rata sebesar 35,88%. Sedangkan dengan set point 85 cm, robot telah mengikuti dinding dengan jarak rata-rata dari dinding sebesar 65,73 cm dengan error rata-rata sebesar 26,3 %. Dari data tersebut diketahui bahwa error rata – rata yang dimiliki robot dalam menjaga jarak dengan dinding kiri sebesar 31,09 % berdasarkan set point jarak yang digunakan.
2. Berdasarkan pengujian menggunakan ruang uji, keberhasilan rata-rata robot untuk menelusuri bagian dari ruangan sebesar 92,67%.
3. Error 7,33% disebabkan oleh ketidakmampuan robot untuk masuk kedalam beberapa bagian ruangan dikarenakan ukuran celah ruangan yang kecil. Pada kondisi seperti itu, maka sistem akan mengaggap celah tersebut sebagai halangan sehingga robot akan mengantisipasi dengan cara menjauhkan diri dari celah tersebut.
4. Penambahan pembagian ruangan pada ukuran yang sama akan semakin menurunkan kemampuan robot dalam menelusuri ruangan. Hal ini disebabkan oleh jarak pembacaan jarak oleh kamera kinect yang terbatas yaitu 80 cm hingga 4 meter.

## Saran

Berdasarkan kesimpulan yang telah diberikan, penulis dapat memberikan saran untuk pengembangan sistem sebagai berikut:

1. Diharapkan menggunakan unit proses yang lebih tinggi kemampuan prosesnya sehingga proses lebih *real time*.
2. Diharapkan dengan metode ini, fitur pada robot dapat ditambah seperti robot pencari bahan berbahaya.
3. Diharapkan robot dapat ditambah dengan sensor jarak *ultrasonic* sehingga dapat menutupi jarak buta pada kinect.
4. Mekanik robot dapat diperkuat sehingga robot dapat digunakan diberbagai medan.

5. Dengan unit proses yang lebih tinggi kemampuan prosesnya, metode dapat dikembangkan dengan menggunakan metode yang digunakan dalam navigasi robot untuk mencari daerah yang belum dijelajahi dengan menganalisa peta yang telah dibuat dan metode untuk mencari jalur terpendek antar satu titik ke titik yang lain berdasarkan peta yang diketahui.

# DAFTAR PUSTAKA


[1] "__", "*Si Robot Mengintai Teroris*". http://informatika.lipi.go.id/ipt/index.php?option=com_content&view=article&id=176:si-robot-mengintai-teroris-&catid=1:latest-news&Itemid=59&lang=in diakses pada 14 Mei 2015.

[2] Correa S. O., Diogo et al. "*Mobile Robots Navigation in Indoor Environments Using Kinect Sensor*". University of Sao Paulo:Brazil, 2012.

[3] Laguna, Guillermo et al. "*Exploration of an Unknown Environment with a Differential Drive Disc Robot*". University of Illinois: United State of America. n.d.

[4] Lee, David. "*The Map Building and Exploration Strategies of a Simple Sonar Equipped Mobile Robot*". Cambridge : United Kingdom. 2003.

[5] "__", "*Wall Follower*". http://www.philohome.com/wallfollower/wallfollower.htm diakses pada 14 Mei 2015.

[6] "__","*Encoder Measurements: How-To Guide*". National Instrument. 2013.

[7] Panich, Surachai. "*Implement Absolute Orientation Instruments for Odometry System Integrated with Gyroscope by Using IKF*". Srinakharinwirot University : Nakhonnayok. 2011.

[8] Khoselham, Kourosh & Sander Oude Elberink. "*Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications*". University of Twente: Netherlands. 2012.

[9] "__" . "*Coordinate Spaces*". https://msdn.microsoft.com/en-us/library/hh973078.aspx diakses 14 Mei 2015.

[10] "__". "*Arduino Mega*". http://www.arduino.cc/en/Main/arduinoBoardMega diakses 14 Mei 2015.

[11] "__". "*Operations with images"*. http://docs.opencv.org/doc/user_guide/ug_mat.html diakses pada 14 Mei 2015.

[12] Anggoro D.A., Janu. "*Perancangan Sistem Pemetaan Otomatis Pada Mobile Robot Menggunakan Sensor Kinect dan Rotari Encoder*".Institut Teknologi Sepuluh Nopember:Surabaya. 2012.

[13] Chevalier, Bernard. "*Turnigy 9X 2.4Ghz radio TGY*". "__" : Prancis. 2010.

[14] "__", "*Si Robot Mengintai Teroris*". http://informatika.lipi.go.id/ipt/index.php?option=com_content&v

iew=article&id=176:si-robot-mengintai-teroris-&catid=1:latest-news&Itemid=59&lang=in diakses pada 14 Mei 2015.

[15] Shiddiq, Ahmad et al."*RANCANG BANGUN ALAT KALIBRASI SENSOR POSISI MENGGUNAKAN METODE ECLUDIAN*", Politeknik Elektronika Negri Surabaya: Surabaya. 2012.

[16] "__", "*Elbit Systems Serves Special Operators*". http://www.soldiermod.com/volume-2-06/elbit-systems.html diakses pada tanggal 14 Mei 2015

# LAMPIRAN

**Kode program Arduino**

```
#include <Wire.h>
#include <Servo.h>
#include <CMPS10.h>

CMPS10 compass;
int dt=0;
unsigned long t;
#define LEFT_CLICKS 201 /* turns out that the wheels
are */
#define RIGHT_CLICKS 201
#define WHEEL_BASE              6.95          /*
distance between wheels, inches */
#define TWOPI                   6.2831853070  /* nice
to have float precision */
#define RADS                    57.2957795    /*
radians to degrees conversion */
#define INCHES_TO_CM            2.54          /* inches
to cm conversion */

float theta          = 0;                     /* bot
heading */
float X_pos          = 0;                     /* bot X
position in inches */
float Y_pos          = 0;                     /* bot Y
position in inches */
float total_inches   = 0;                     /* total
inches traveled */

float left_inches   = 0; float right_inches = 0;

float full_inches = 0;
int L_ticks, R_ticks=0;           /* icc11 typedefs
defined in <int32.h> */
int last_left, last_right=0;
int lsamp, rsamp=0;
float X_POS_CM = 0; float Y_POS_CM = 0;
int BUFFER_X_POS_CM = 0;
int BUFFER_Y_POS_CM = 0;
int DATA_MODE = 0;
```

```
float sudut;

int BUFFER_SUDUT=0;
int ail; int tro; int ken;
int mov_time = 10;        //movement_time (mili
second) , minimal 10 ms
int init_mov_time = 40;    //initial_movement_time
(mili second) for turn and backward , minimal 40 ms
int time_int=0;
int countleft=0; int countright=0;

int left=0; int right=0;
const int B_ENCODER_RIGHT  = 34; const int
B_ENCODER_LEFT   = 35;
int B_ENCODER_RIGHT_CONDITION = 0; int
B_ENCODER_LEFT_CONDITION  = 0;
int COUNT_LEFT_POSITION  = 0; int COUNT_RIGHT_POSITION
= 0;
int data_buffer1a=0; int data_buffer1b=0; int
data_buffer2=0;
boolean gerak_maju = false;
boolean gerak_kiri = false;
boolean gerak_kanan = false;
boolean gerak_mundur = false;
boolean gerak_diam = true;
Servo motor_kanan;
Servo motor_kiri;
int buff_data1; int buff_data2;
int buff_data3; int buff_data4;
int buff_data; int buff_data5;
int buff_data6; int buff_data7;
int buff_data8; int buff_datab;
int pwm_r,pwm_l;
int simpan_x=0;
int simpan_y=0;
int simpan_sudut=0;
int flag_simpan=0;  //0 -> belum simpan, 1->simpan
int buff_sudut;
int flag_sudut_pertama=0;
void diam();
void maju();
void mundur();
void kanan();
void kiri();
```

```
void odometri();

// Main code ----------------------------------------
------------------------
void setup(){
  Serial.begin(9600);
  motor_kanan.attach(9);
  motor_kiri.attach(10);
  motor_kanan.writeMicroseconds(1460);
  motor_kiri.writeMicroseconds(1460);
  Serial.println("\nBISMILLAH....");delay(1000);
  Serial.println("1....");delay(1000);
  Serial.println("2....");delay(1000);
  Serial.println("3....");delay(1000);
  Serial.println("4....");delay(1000);
  Serial.println("5....");delay(1000);
  attachInterrupt(0, count_left, FALLING); //FALLING);
//pin 2 -> KIRI
  attachInterrupt(1, count_right, FALLING); //
FALLING); //pin 3 -> KANAN
  pinMode(B_ENCODER_RIGHT, INPUT);
  pinMode(B_ENCODER_LEFT, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  diam();
}

// Main loop ----------------------------------------
------------------------
void loop(){
  theta=compass.bearing();
  if (flag_sudut_pertama==0){
    buff_sudut=theta;
    flag_sudut_pertama=1;
  }
  theta=theta-buff_sudut;
  if (theta<0)theta=360+theta;
  odometri();
  ken = pulseIn(4, HIGH, 25000);
  if (ken >= 1400)          {kendali=false;}  //false =
auto , true = manual
  else if (ken < 1390)    {kendali=true;}
  //MODE MANUAL
```

```
  if(kendali)    //Choosing Mode ::: False -> Auto |||
True -> Manual
  {
    DATA_MODE=0;  //manual
    tro = pulseIn(5, HIGH, 25000);
    ail = pulseIn(6, HIGH, 25000);
    tro=(map(tro, 1039, 1804, 1080, 1750));  //1039,
1804
    ail=(map(ail, 1066, 1857, 1080, 1750));  //1066,
1857
    motor_kanan.writeMicroseconds(tro);
    motor_kiri.writeMicroseconds(ail);
    ken = pulseIn(4, HIGH, 25000);
  }
  //MODE OTOMATIS
  else
  {
    DATA_MODE=1;   //auto
    if (flag_simpan==0)
      {
        simpan_x      = int(X_POS_CM);
        simpan_y      = int(Y_POS_CM);
        simpan_sudut  = int(theta);
        flag_simpan   = 1;
      }
    if (gerak_maju){maju();}
    else if (gerak_mundur){mundur();}
    else if (gerak_kanan){kanan();}
    else if (gerak_kiri){kiri();}
    else if (gerak_diam){diam();}
  }
 //---------- kirim data [start]
    BUFFER_X_POS_CM = int(X_POS_CM);//-simpan_x ;
    BUFFER_Y_POS_CM = int(Y_POS_CM);//-simpan_y ;
    BUFFER_SUDUT    = int(theta);//-simpan_sudut;
    if ( BUFFER_X_POS_CM >= 0 ) BUFFER_X_POS_CM =
10000 + abs(BUFFER_X_POS_CM) ;
    else                        BUFFER_X_POS_CM =
20000 + abs(BUFFER_X_POS_CM) ;
    if ( BUFFER_Y_POS_CM >= 0 ) BUFFER_Y_POS_CM =
10000 + abs(BUFFER_Y_POS_CM) ;
    else                        BUFFER_Y_POS_CM =
20000 + abs(BUFFER_Y_POS_CM) ;
```

```
    if ( BUFFER_SUDUT >= 0 ) BUFFER_SUDUT = 1000 +
abs(BUFFER_SUDUT) ;
    else                     BUFFER_SUDUT = 2000 +
abs(BUFFER_SUDUT) ;
    char b[16];

sprintf(b,"q%d%d%d%d",BUFFER_X_POS_CM,BUFFER_Y_POS_CM,
BUFFER_SUDUT,DATA_MODE);
    Serial.println(b);
 //---------- kirim data [end]
}
void diam()
{
  motor_kanan.writeMicroseconds(1460);
  motor_kiri.writeMicroseconds(1460);
  delay(mov_time);
}

void maju()
{
  flag_mundur = false;
  flag_kanan = false;
  flag_kiri = false;
  event_moving_left  = true;
  event_moving_right = true;
  motor_kanan.writeMicroseconds(pwm_r);   //1600
  motor_kiri.writeMicroseconds(pwm_l);    //1600
  delay(mov_time);
}

void mundur()
{
  event_moving_left  = false;
  event_moving_right = false;

  if (!flag_mundur)
  {
    motor_kanan.writeMicroseconds(1250);
    motor_kiri.writeMicroseconds(1250);
    delay(init_mov_time);
    motor_kanan.writeMicroseconds(1460);
    motor_kiri.writeMicroseconds(1460);
    delay(init_mov_time);
  }
```

```
  flag_mundur = true;
  motor_kanan.writeMicroseconds(1250);
  motor_kiri.writeMicroseconds(1250);
  delay(mov_time);
}

void kiri()
{
  event_moving_left  = false;
  event_moving_right = true;

  if (!flag_kiri)
  {
    motor_kanan.writeMicroseconds(1250);
//init_mov_time
    motor_kiri.writeMicroseconds(1250);
//init_mov_time
    delay(200);  //init_mov_time
    motor_kanan.writeMicroseconds(1460);
    motor_kiri.writeMicroseconds(1460);
    delay(200);  //init_mov_time
  }
  flag_kiri = true;
  motor_kanan.writeMicroseconds(1560); //1600
  motor_kiri.writeMicroseconds(1290);  //1250
  delay(mov_time);
}

void kanan()
{
  event_moving_left  = true;
  event_moving_right = false;
  if(!flag_kanan)
  {
    motor_kanan.writeMicroseconds(1250);
    motor_kiri.writeMicroseconds(1250);
    delay(200);
    motor_kanan.writeMicroseconds(1460);
    motor_kiri.writeMicroseconds(1460);
    delay(200);
  }
  flag_kanan = true;
  motor_kanan.writeMicroseconds(1250+30);//(1265-10);
//1250
```

```
  motor_kiri.writeMicroseconds(1600-30);
  delay(mov_time);
}
void serialEvent(){
  char newByte = Serial.read();
  if (newByte=='a')  //maju
  {
    char cbuff_data1 = Serial.read();
    char cbuff_data2 = Serial.read();
    char cbuff_data3 = Serial.read();
    char cbuff_data4 = Serial.read();
    char cbuff_data5 = Serial.read();
    char cbuff_data6 = Serial.read();
    char cbuff_data7 = Serial.read();
    char cbuff_data8 = Serial.read();
    buff_data1 =( cbuff_data1 - 48 ) * 1000;
    buff_data2 =( cbuff_data2 - 48 ) * 100;
    buff_data3 =( cbuff_data3 - 48 ) * 10;
    buff_data4 =( cbuff_data4 - 48 ) * 1;
    buff_data = buff_data1 + buff_data2 + buff_data3 +
buff_data4;
    pwm_r = buff_data;
    buff_data5 =( cbuff_data5 - 48 ) * 1000;
    buff_data6 =( cbuff_data6 - 48 ) * 100;
    buff_data7 =( cbuff_data7 - 48 ) * 10;
    buff_data8 =( cbuff_data8 - 48 ) * 1;
    buff_datab = buff_data5 + buff_data6 + buff_data7
+ buff_data8;
    pwm_l = buff_datab;
    gerak_maju    = true;
    gerak_kiri    = false;
    gerak_kanan   = false;
    gerak_mundur  = false;
    gerak_diam    = false;
    flag_uji_coba=true;
  }
  if (newByte=='u')  //mundur
  {
    gerak_maju    = false;
    gerak_kiri    = false;
    gerak_kanan   = false;
    gerak_mundur  = true;
    gerak_diam    = false;
  }
```

```
  if (newByte=='k')  //kanan
  {
    gerak_maju    = false;
    gerak_kiri    = false;
    gerak_kanan   = true;
    gerak_mundur  = false;
    gerak_diam    = false;
  }
  if (newByte=='i')  //kiri
  {
    gerak_maju    = false;
    gerak_kiri    = true;
    gerak_kanan   = false;
    gerak_mundur  = false;
    gerak_diam    = false;
  }

  if (newByte=='d')  //diam
  {
    gerak_maju    = false;
    gerak_kiri    = false;
    gerak_kanan   = false;
    gerak_mundur  = false;
    gerak_diam    = true;
  }
}
void count_left()
{
  B_ENCODER_LEFT_CONDITION =
digitalRead(B_ENCODER_LEFT);

  left++;  //for speed

  if(B_ENCODER_LEFT_CONDITION==HIGH)
    {
      //left++;
      COUNT_LEFT_POSITION++;
    }
   else if( B_ENCODER_LEFT_CONDITION==LOW)
   {
     //left--;
     COUNT_LEFT_POSITION--;
   }
}
```

```
void count_right()
{
  B_ENCODER_RIGHT_CONDITION =
digitalRead(B_ENCODER_RIGHT);

  right++;  //for speed

  if(B_ENCODER_RIGHT_CONDITION==LOW)
  {
    //right++;
    COUNT_RIGHT_POSITION++;
  }
   else if(B_ENCODER_RIGHT_CONDITION==HIGH)
   {
     //right--;
     COUNT_RIGHT_POSITION--;
   }
}
void odometri()
{
  lsamp = COUNT_LEFT_POSITION;
  rsamp = COUNT_RIGHT_POSITION;
  COUNT_LEFT_POSITION  = 0 ;
  COUNT_RIGHT_POSITION = 0 ;
  L_ticks = lsamp ;//- last_left;
  R_ticks = rsamp ;//- last_right;
  last_left = lsamp;
  last_right = rsamp;
   left = COUNT_LEFT/LEFT_CLICKS;
   right=COUNT_RIGHT/RIGHT_CLICKS;
   in = (left + right) / 2.0;
   Y_pos += in * cos(theta * PI/180 );
   X_pos += in * sin(theta * PI/180 );
   X_POS = X_pos;
   Y_POS = Y_pos;
}
```

**Kode Program Unit Proses dengan C++**

```
//////////////////////////////////////////////////
//        BISMILLAHHIROHMAANNIRROHIM
//
// ---------------------------------------
// NAMA       : NOVEM ARDAN ROHMADIN
// NRP : 2211100051
//
//////////////////////////////////////////////////
#include "stdafx.h"
#include <math.h>
#include "variables.h"
#include <cv.h>
#include <highgui.h>
#include <cxcore.h>
#include <windows.h>
#include <NuiApi.h> // Microsoft Kinect SDK
#include <iostream>
using namespace std;
#define PI 3.141592654
//============== VARIABEL ====================//
USHORT dalam;
RGBQUAD m_rgbWk[640*480];
INuiSensor *        m_pNuiSensor;
CvPoint pt; CvPoint ptbuff;
int tekan =0;
IplImage* DepthImage;
IplImage* DepthImage2;
int JARAK_TERDEKAT = 800;
int JARAK_TERJAUH   = 1500;
bool ADA_BENDA = false;
int map_record[1000][1000];
int cekka=0;int cekki=0;int cekx=0;int ceky=0;
int ly;int lx; int lr;
int status=0; int straight=0;
int REGION_1 = 0; int REGION_2 = 0;
int REGION_3 = 0; int REGION_4 = 0;
int REGION_5 = 0;
int reg1=0; int reg2=0;
int reg3=0; int reg4=0;
int reg5=0;
int flag_celah=0;
int jarak_nol[5]; int sudut_target;int sudut_ambil;
int ba; int bi;
```

```
int jarak_nol_dinding[6];
int ca = 0; int ci = 0;
int jarak1=0; int jarak2=0; int jarak3=0;
int batas_atas = 340; int batas_bawah = 420;
bool KONDISI_JARAK = false;
int ADA_BENDA_SEBELUMNYA = 0;
int cek_jarak[640][480];
//------- untuk pemetaan
bool SaveMap =      false;
int batas_peta=170; int jarak[700];
double sudut = 0; int DIAMETER=100;
bool DATA_UPDATE=true;
int X_POS,Y_POS=0; int ANGLE=0;
int sudut0 = 0; int tanda_sudut_masuk=0;
int set_sudut_posisi=0;
int simpan_x,simpan_y,simpan_sudut=0;
int last_x,last_y=500; int B,G,R=0;
float sudut_ambil_jarak=0;
float x_real=0; float y_real[640][480];
int map_weight[1000][1000]; float y_buf;
int angleview[360];
int celahstart[3],celahend[3];
int           setpoint=200;   // mm
int           PWM;
int           jarak_acuan;
int pwm_max_maju = 1600; int pwm_min_maju = 1550;
#pragma once
namespace TugasAkhirKinect {
      using namespace System;
      using namespace System::ComponentModel;
      using namespace System::Collections;
      using namespace System::Windows::Forms;
      using namespace System::Data;
      using namespace System::Drawing;
      using namespace System::IO::Ports;

      /// <summary>
      /// Summary for Form1
      /// </summary>
      public ref class Form1 : public
System::Windows::Forms::Form
      {
      public:
            Form1(void)
```

```
            {
                InitializeComponent();
                findPorts();
            }

        protected:
            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            ~Form1()
            {
                if (components)
                {
                    delete components;
                }
            }
        protected:

        private:
            /// <summary>
            /// Required designer variable.
            /// </summary>


#pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support
- do not modify
            /// the contents of this method with the
code editor.
            /// </summary>

//======== FUNGSI - FUNGSI ===============//
RGBQUAD Nui_ShortToQuad_Depth(USHORT s)
{
        USHORT realDepth = (s&0xfff8) >> 3;
        BYTE l = 255-(BYTE)(256*realDepth /(0x0fff));
            dalam = realDepth;
            RGBQUAD q;
        q.rgbRed = q.rgbBlue = q.rgbGreen = l;
        return q;
}
#pragma endregion
//find available ports
```

```
        private: void findPorts(void)
                 {
array<Object^>^ objectArray =
SerialPort::GetPortNames();
this->comboBox1->Items->AddRange(objectArray);
                 }
        private: void maju(void)
                 {
                              this->label10-
>Text="MAJU"; this->textBox2->Text=String::Concat(
"a",pwm_r,pwm_l);
String^ message = String::Concat( "a",pwm_r,pwm_l);
                 }
        private: void kiri(void)
                 {
this->label10->Text="KIRI"; this->textBox2->Text="i";
String^ message = "i";
                 }
        private: void kanan(void)
                 {
this->label10->Text="KANAN"; this->textBox2->Text="k";
String^ message = "k";
                 }
        private: void diam(void)
                 {
this->label10->Text="DIAM"; this->textBox2->Text="d";
String^ message = "d";
                 }
label51->Text = String::Concat( "",environment[0]);
label52->Text = String::Concat( "",environment[1]);
label53->Text = String::Concat( "",environment[2]);
label54->Text = String::Concat( "",environment[3]);
label55->Text = String::Concat( "",environment[4]);
label56->Text = String::Concat( "",environment[5]);
label57->Text = String::Concat( "",environment[6]);
label58->Text = String::Concat( "",environment[7]);
}
        private: void diferential_drive(void)
                 {
                              Radius = 2.25;
                              L = 17.653;
v_lin = 0 + (30-0) * (((float)PWM-1460)/(1750-1460));
v_ang=0;
```

```
if((environment[0]==0 && environment[1]==0 &&
environment[7]==0)||
(environment[0]==1 && environment[1]==0 &&
environment[7]==1)||
(environment[0]==0 && environment[1]==0 &&
environment[7]==1) ||
(environment[0]==1 && environment[1]==0 &&
environment[7]==0) )
{ v_ang=0; }
else if ( (environment[0]==1 && environment[1]==1 &&
environment[7]==0) ||
(environment[1]==0 && environment[2]==0 &&
environment[3]==0) ||
(environment[1]==0 && environment[2]==0 &&
environment[3]==1) ||
(environment[1]==0 && environment[2]==1 &&
environment[3]==1) )
{
 v_ang=90;
}
else
{
 v_ang=-90;
}
v_ang=v_ang/180*3.14;
v_ang=0;
vr = ( 2*v_lin - L*v_ang ) / (2*Radius);
vl = ( 2*v_lin + L*v_ang ) / (2*Radius);
pwm_r = (int) 1460 + (1750-1460) * ((vr-0)/(30-0));
pwm_l = (int) 1460 + (1750-1460) * ((vl-0)/(30-0));
if (pwm_r >= 1675) pwm_r=1675;
else if (pwm_r<=1550) pwm_r=1550;
if (pwm_l >= 1675) pwm_l=1675;
else if (pwm_l<=1550) pwm_l=1550;
}
//============== PROGRAM UTAMA ================//
private: System::Void button1_Click(System::Object^
sender, System::EventArgs^  e) {
//---- inisialisasi tombol [START] ---------------//
this->button1->BackColor =
System::Drawing::Color::Red;
this->button1->Text = L"STOP";
JARAK_TERDEKAT = trackBar1->Value;
JARAK_TERJAUH = trackBar2->Value;
```

```
//------ inisialisasi tombol [END] -------------//
//------ inisialisasi variabel KINECT [START] --//
HANDLE depthStreamHandle;
HANDLE nextDepthFrameEvent = CreateEvent(NULL, TRUE,
FALSE, NULL);
NuiInitialize(NUI_INITIALIZE_FLAG_USES_DEPTH);
NuiImageStreamOpen(NUI_IMAGE_TYPE_DEPTH,
NUI_IMAGE_RESOLUTION_640x480, 0, 2, NULL,
&depthStreamHandle);
cvNamedWindow("Hello Kinect!", CV_WINDOW_AUTOSIZE);
IplImage* DepthImage = cvCreateImage(cvSize(640,480),
IPL_DEPTH_8U,4);
IplImage* Frame_Kinect =
cvCreateImage(cvSize(320,240), IPL_DEPTH_8U,4);
IplImage* Buffer_DepthImage=
cvCreateImage(cvSize(320,240), IPL_DEPTH_8U,3);
IplImage* Frame_DepthImage =
cvCreateImage(cvSize(320,240), IPL_DEPTH_8U,1);
IplImage* Showing_DepthImage =
cvCreateImage(cvSize(640,480), IPL_DEPTH_8U,3);
IplImage* Buffer_Showing_DepthImage =
cvCreateImage(cvSize(320,240), IPL_DEPTH_8U,3);
IplImage* Buffer_Showing_DepthImage2 =
cvCreateImage(cvSize(320,240), IPL_DEPTH_8U,3);
IplImage* Peta1 = cvCreateImage(cvSize(1000,1000),
IPL_DEPTH_8U,3);
IplImage* Peta2 = cvCreateImage(cvSize(320,240),
IPL_DEPTH_8U,3);
IplImage* gray = 0;
IplImage* Peta_dinding =
cvCreateImage(cvSize(1000,1000), IPL_DEPTH_8U,3);
IplImage* ruangan = cvCreateImage( cvGetSize(Peta1),
8, 3 );
Dst = cvCreateImage( cvGetSize(Peta1), 8, 1 );
color_dst = cvCreateImage( cvGetSize(Peta1), 8, 3 );
//---- inisialisasi variabel OPENCV [END] ---------//
int jarakx0=800; int jarakx1=1500;
        while(1)
        {
//----- inisialisasi ambil data depth KINECT------//
const NUI_IMAGE_FRAME *pImageFrame = NULL;
HRESULT hr =
NuiImageStreamGetNextFrame(depthStreamHandle,1000,
&pImageFrame);
```

```
            if(FAILED(hr))
                {
                        printf("Failed to get a Next
Frame(Kinect Image)\n");
                        continue;
                }
                INuiFrameTexture *pTexture =
pImageFrame->pFrameTexture;
                NUI_LOCKED_RECT LockedRect;
                pTexture->LockRect(0, &LockedRect,
NULL,0);
gray = cvCreateImage(cvSize(DepthImage->width,
DepthImage->height), 8, 1);
                if(LockedRect.Pitch != 0)
                {
BYTE * pBuffer = (BYTE*)LockedRect.pBits;
RGBQUAD * rgbrun = m_rgbWk;
USHORT * pBufferRun = (USHORT*) pBuffer;
ADA_BENDA = false;
ADA_BENDA_REGION_1 = false;
ADA_BENDA_REGION_2 = false;
ADA_BENDA_REGION_3 = false;
ADA_BENDA_REGION_4 = false;
ADA_BENDA_REGION_5 = false;
REGION_1 = 0 ;
REGION_2 = 0 ;
REGION_3 = 0 ;
REGION_4 = 0 ;
REGION_5 = 0 ;
int jarak_minimum = 10000; int jarak_maximum = 0;
titik_x_minimum = 0; int titik_y_minimum = 0;
int titik_x_maximum = 0; int titik_y_maximum = 0;
this->label11->Text="depth";
int min=0; int max=0;
int flag_min=0;int flag_max=0;
int flag_hitung_lebar=0; int lebar=0;
int sudut_x=0;
int j[100];
jarak_acuan=0;
int depth_1, depth_2, depth_3;
int jumlah_depth_1,jumlah_depth_2,jumlah_depth_3;
sudut_ambil_jarak = -28.5;
tsudut=0;
//--------- ambil data depth KINECT-------------//
```

```
for( int y = 0 ; y < 480 ; y++ )
{
        sudut_ambil_jarak = -28.5;
        tsudut=28.5;
        for( int x = 0 ; x < 640 ; x++ )
        {
         RGBQUAD quad = Nui_ShortToQuad_Depth(
*pBufferRun );
         pBufferRun++;\
         *rgbrun = quad;
         rgbrun++;
        cek_jarak[639-x][y]=dalam;
        y_buf=(float)y/480;
        y_real[639-x][y] = (y_buf-0.5)*(cek_jarak[639-
x][y]/10)*0.84024;
        if (y_real[639-x][y]>0 && y_real[639-x][y]<15
&& cek_jarak[639-x][y]!=0)  //15
        {
        gray->imageData[gray->widthStep * y + x*gray-
>nChannels] = 255;
        if ( ( x >= 0 ) && ( x <= 212 ) ){
                //0-212
                depth_3 += dalam;
                jumlah_depth_3++;
                }
                else if ( ( x >= 213 ) && ( x <= 425 )
){              //213-425
                depth_2 += dalam;
                jumlah_depth_2++;
                }
                else if ( ( x >= 426 ) && ( x <= 639 )
){              //426-639
                depth_1 += dalam;
                jumlah_depth_1++;
                }
if(jarak[639-x]<=dalam && dalam!=0)//(jarak[639-
x]<=dalam)
                {
jarak[639-x]= (int)( dalam /
sin(abs(abs(sudut_ambil_jarak)-90)/180*3.14) );
                }
                }
                else
                {
```

```
gray->imageData[gray->widthStep * y + x*gray-
>nChannels] = 0;
}
sudut_ambil_jarak += 0.0890625;
//-------------------------------------------------//
                    }
                   }
int lside; int hal_ki;
int selisih_kiri=0;
if (jumlah_depth_1== 0) jumlah_depth_1= 1;
if (jumlah_depth_2== 0) jumlah_depth_2= 1;
if (jumlah_depth_3== 0) jumlah_depth_3= 1;
if (jumlah_depth_1!= 0 && jumlah_depth_2!= 0 &&
jumlah_depth_3!= 0)
       {
       lside = ( (depth_1/jumlah_depth_1) +
(depth_2/jumlah_depth_2) + (depth_3/jumlah_depth_3)
)/3;
hal_ki = 1000 - lside; //set_point
}
selisih_kiri = (depth_3/jumlah_depth_3)-
(depth_1/jumlah_depth_1);
if (jumlah_depth_1!=0)label36->Text = String::Concat(
"",depth_1/jumlah_depth_1," mm, ",jarak_nol[1]);
//lpath
if (jumlah_depth_2!=0)label37->Text = String::Concat(
"",depth_2/jumlah_depth_2," mm, ",jarak_nol[2]);
if (jumlah_depth_3!=0)label39->Text = String::Concat(
"",depth_3/jumlah_depth_3," mm, ",jarak_nol[3]);  //lr
label42->Text = String::Concat(
"",selisih_kiri,",",y_tujuan);
label47->Text = String::Concat( "",lside," +++
",hal_ki,"  ",(sudut_x-320)*0.089*-1);
jarak_nol[1]=0;jarak_nol[2]=0;jarak_nol[3]=0;
label12->Text = String::Concat( "",time_click);
label13->Text = String::Concat( "",kali_maju);
label14->Text = String::Concat(
"",titik_tengah_x,",",titik_tengah_y);
label14->Text = String::Concat(
"",kali_maju,",",kali_diam);
//---- maju ikut dinding
//---------- cek kiri ---------------------//
if (MODE == 1 && status_aksi==1){
if (kali_kiri<12){
```

```
kiri();
kali_kiri++;
}
else if (kali_kiri>=12){
if (lebar > 300){
straight=0;
}
if (( straight == 0 && ( status == 0 ||  status == 5
))  ) { //|| jarak2/17040 > 900
//maju();
straight=0;
status_aksi=4;
}
lebar=0;
kanan();
kali_kiri++;
if (kali_kiri>=24){
      status_aksi=4;}
      }
}
if (MODE==0 ){
diam();
}
jarak2=0;
timer_awal++;
if ( timer_awal>700 ){
//diam();
x_dis = (float) titik_awal_x - (float) X_POS;
y_dis = (float) titik_awal_y - (float) Y_POS;
dis_target = sqrt( (x_dis*x_dis) + (y_dis*y_dis) );
if (dis_target<40 && flag_sampai==0){
status_aksi=9;
flag_sampai=1;
            }
      }
//------------REVISI PROGRAM
      int r1=0;int r2=0; int r3=0;
      int rata_reg1=depth_1/jumlah_depth_1;
      if ( rata_reg1 > 2000  )  {r1 = 1;}
      else {r1=0;}
      int rata_reg2=depth_2/jumlah_depth_2;
      if ( rata_reg2 < 1000  ) {r2 = 0;}
      else {r2=1;}
      int rata_reg3=depth_3/jumlah_depth_3;
```

```
        if ( rata_reg3 < 100 )  {r3 = 1;}
        else {r3=0;}
        int dinding_kiri=100;  //6-20-2015 -> 2000
        if (MODE==1 && status_aksi==875)
        {
        kanan();
        kirim_data();
        status_aksi=875;
        kali_maju++;
        if (lebar > 300){

        straight=0;
        }
        if ( straight == 0 && ( status == 0 ||  status
== 5 ) ){
        status_aksi=999; //maju();
        kali_maju=0;
        straight=0;
        }
        lebar=0;
        }
if (MODE==1 && status_aksi == 666)
        {
                kanan();
                kirim_data();
        }
if (MODE == 1 && status_aksi == 999)
        {
                if ( (rata_reg3<=1000) )
        {
        status_aksi=875;
        kali_maju=0;
        }
        else if  ( ((selisih_kiri<=dinding_kiri) ||
r3==1 ) && rata_reg3>1500 )
        {
        diam();
        flag_kirim=0;
        kirim_data();
        status_aksi=100;
        kali_maju=0;
        }
        else
        {
```

```
        pwm_r=1565;pwm_l=1565;//1575;pwm_l=1575;
        maju();
        if (flag_kirim==0)
        {
        kirim_data();
        flag_kirim=1;
        }}}
if (MODE == 1 && status_aksi == 100)
        {
                if (flag_kirim==0)
                {
                pwm_r = pwm_kanan;
                pwm_l = pwm_kiri;
                maju();
                kirim_data();
                flag_kirim=1;
                }
if ( kali_maju == delay_belok ) flag_kirim=0;
                kali_maju++;
                if ( kali_maju > delay_belok )
                {
                if (flag_kirim==0)
                {
                pwm_r = 1575; pwm_l = 1575;
                maju();
                kirim_data();
                flag_kirim=1;
                }
if ( kali_maju > delay_belok + delay_maju )
{
        status_aksi = 999;
}
}
if ( (rata_reg3<=1000) && (rata_reg3!=0) )
                {
                status_aksi=875;
                kali_maju=0;
                }
//------- jika ada halangan belok kanan hingga aman
                                                }

//-0-0-0-0-0-0-0-0
jarak1=0;jarak3=0;
reg1=0;reg2=0;reg3=0;
```

```
depth_3=0;depth_2=0;depth_1=0;
jumlah_depth_3=0;jumlah_depth_2=0;jumlah_depth_1=0;
//-0-0-0-0-0-0-0-0
jarak_nol_dinding[0]=0;jarak_nol_dinding[1]=0;jarak_no
l_dinding[2]=0;jarak_nol_dinding[3]=0;jarak_nol_dindin
g[4]=0;jarak_nol_dinding[5]=0;
cvSetData(DepthImage, (BYTE*) m_rgbWk, DepthImage-
>widthStep);
cvShowImage("Hello Kinect!", DepthImage);
cvResize(DepthImage,Frame_Kinect);
cvFlip(Frame_Kinect,Frame_Kinect, 1);
cvLine(Frame_Kinect,cvPoint(0,batas_peta/2),cvPoint(32
9,batas_peta/2),CV_RGB(0,0,250),2,8);
cvLine(Frame_Kinect,cvPoint(0,batas_atas/2),cvPoint(32
9,batas_atas/2),CV_RGB(250,0,0),2,8);
cvLine(Frame_Kinect,cvPoint(0,batas_bawah/2),cvPoint(3
29,batas_bawah/2),CV_RGB(250,0,0),2,8);
cvLine(Frame_Kinect,cvPoint(106/2,0/2),cvPoint(106/2,4
79/2),CV_RGB(250,0,0),2,8);
cvLine(Frame_Kinect,cvPoint(318/2,0/2),cvPoint(318/2,4
79/2),CV_RGB(250,0,0),2,8);
cvLine(Frame_Kinect,cvPoint(532/2,0/2),cvPoint(532/2,4
79/2),CV_RGB(250,0,0),2,8);
cvLine(Frame_Kinect,cvPoint(106/2,0/2),cvPoint(106/2,4
79/2),CV_RGB(250,0,0),2,8);
cvLine(Frame_Kinect,cvPoint(212/2,0/2),cvPoint(212/2,4
79/2),CV_RGB(250,0,0),2,8);
cvLine(Frame_Kinect,cvPoint(425/2,0/2),cvPoint(425/2,4
79/2),CV_RGB(250,0,0),2,8);
cvCircle(Frame_Kinect,cvPoint(sudut_x/2,batas_bawah/2)
, 10, CV_RGB(0,255,0),4,8);
pictureBox1->Image  = gcnew System::Drawing::Bitmap
(Frame_Kinect->width,Frame_Kinect-
>height,Frame_Kinect->widthStep,
System::Drawing::Imaging::PixelFormat::Format32bppRgb,
(System::IntPtr) Frame_Kinect->imageData);
pictureBox1->Refresh();
cvFlip(gray,gray,1);
cvCvtColor(gray, Showing_DepthImage,  CV_GRAY2BGR);
cvLine(Showing_DepthImage,cvPoint(212,0),cvPoint(212,4
79),CV_RGB(250,0,0),2,8);
cvLine(Showing_DepthImage,cvPoint(425,0),cvPoint(425,4
79),CV_RGB(250,0,0),2,8);
```

```
cvLine(Showing_DepthImage,cvPoint(0,batas_atas),cvPoin
t(639,batas_atas),CV_RGB(250,0,0),2,8);
cvResize(Showing_DepthImage,
Buffer_Showing_DepthImage);
cvLine(Buffer_Showing_DepthImage,cvPoint(0,batas_bawah
/2),cvPoint(329,batas_bawah/2),CV_RGB(250,0,0),2,8);

pictureBox4->Image  = gcnew System::Drawing::Bitmap
(Buffer_Showing_DepthImage-
>width,Buffer_Showing_DepthImage-
>height,Buffer_Showing_DepthImage->widthStep,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,
(System::IntPtr) Buffer_Showing_DepthImage-
>imageData);
pictureBox4->Refresh();
DATA_UPDATE=true;
if (DATA_UPDATE)
    {
        double xp, yp , yin, xin  = 0;
        double p2_xpos , p2_ypos = 0;
        double p_xpos , p_ypos = 0;
        environment_check();
        sudut = ANGLE;
        p2_xpos = X_POS/2;
        p2_ypos = Y_POS/2;
        label48->Text = String::Concat( "x:",X_POS);
        label49->Text = String::Concat( "y:",Y_POS);
        label50->Text = String::Concat(
"angle:",ANGLE);
        double sudut_buff=0;
        sudut_buff = sudut - 28.5;
        if (set_sudut_posisi == 1)
        {
                simpan_x            = p2_xpos ;
                simpan_y            = p2_ypos ;
                simpan_sudut  = sudut        ;
                for( int x = 0 ; x < 1000 ; x++ ) {
                for( int y = 0 ; y < 1000 ; y++ ) {
((uchar *)(Peta1->imageData + (int)(y)*Peta1-
>widthStep))[(int)(x)*Peta1->nChannels + 0]=0;
((uchar *)(Peta1->imageData + (int)(y)*Peta1-
>widthStep))[(int)(x)*Peta1->nChannels + 1]=0;  // G
((uchar *)(Peta1->imageData + (int)(y)*Peta1-
>widthStep))[(int)(x)*Peta1->nChannels + 2]=0;  // R
```

```
                }
        }
set_sudut_posisi= 0         ;
        }
        p2_xpos = p2_xpos - simpan_x;
        p2_ypos = p2_ypos - simpan_y;
        sudut = sudut - simpan_sudut;
        cekka=0;cekki=0;
        if ( first_draw == 0 ){
cvLine(Peta1,cvPoint(500,500),cvPoint(500,500),CV_RGB(
250,0,0),3,6);
        first_draw = 1;
        //---- for avoid past
                peta_jalan[10][10]=1;
        }
        else {
if(abs(last_x-500-p2_xpos)<50 && abs(last_y-500-
p2_ypos)<50
)cvLine(Peta1,cvPoint(last_x,last_y),cvPoint((500+p2_x
pos),(500+p2_ypos)),CV_RGB(255,0,0),5,8);
        //---- for avoid past
        if (flag_gambar_jalan==1){
        int buff_x_jalan;int buff_y_jalan;
        buff_x_jalan = (int)(500+p2_xpos)/33;
        buff_y_jalan = (int)(500+p2_xpos)/33;
        peta_jalan[(int)(500+p2_xpos)/33][(int)(500+p2_
ypos)/33]=1;
        }
        }
        titik_x=p2_xpos; titik_y=p2_ypos;
        for( int x = 0 ; x < 640 ; x++ )
          {
            if (jarak[x] != 0 )
        {
        jarak[x]=jarak[x]/10/2;
p_xpos = (500 + p2_xpos + ( jarak[x] *
sin(sudut_buff*PI/180.0) ) ) *( 1);
p_ypos = (500 + p2_ypos + ( jarak[x] *
cos(sudut_buff*PI/180.0) ) ) *( 1);
sudut_buff += 0.089 ;
uchar b,g,r;
buffer_p_xpos = p_xpos;
buffer_p_ypos = p_ypos;
```

```
if ( buffer_p_xpos>=1000 ) buffer_p_xpos=1000; if (
buffer_p_xpos<=0 ) buffer_p_xpos=0;
if ( buffer_p_ypos>=1000 ) buffer_p_ypos=1000; if (
buffer_p_ypos<=0 ) buffer_p_ypos=0;
r=(uchar)Peta1->imageData[Peta1 -
>widthStep*(int)(buffer_p_ypos) +
(int)(buffer_p_xpos)*Peta1 ->nChannels+2];
if (p_xpos < 1000 && p_ypos < 1000 && p_xpos >= 0 &&
p_ypos >= 0 )
{
if ( r != 255 )
{
cvLine(Peta1,cvPoint(p2_xpos+500,p2_ypos+500),cvPoint(
p_xpos,p_ypos),CV_RGB(255,255,255),6,8);
cvRectangle(Peta1,cvPoint((int)(p_xpos),(int)(p_ypos+5
)), cvPoint((int)(p_xpos),(int)(p_ypos-5)),
CV_RGB(255,255,255),3,8,0);
cvCircle(Peta_dinding,
cvPoint((int)(p_xpos),(int)(p_ypos)), 7,
CV_RGB(255,0,0),8,0);
map_weight[(int)p_xpos][(int)p_ypos]++;
                    }
                }
        }
}
{
last_x=500 + p2_xpos;last_y=500 + p2_ypos;
DATA_UPDATE = true;
                                        }
                                    }
for (int i = 0; i<11 ; i++){
cekx0[i]; ceky0[i];
cekx1[i]; ceky1[i];
cektengahx[i]; cektengahy[i];
cekpanjangx[i]; cekpanjangy[i];
}
cvResize(Peta_dinding,Peta2);  //Peta1
pictureBox3->Image  = gcnew System::Drawing::Bitmap
(Peta2->width,Peta2->height,Peta2->widthStep,

System::Drawing::Imaging::PixelFormat::Format24bppRgb,
(System::IntPtr) Peta2->imageData);
pictureBox3->Refresh();
label24->Text = String::Concat( "",jarak_maximum);
```

```
if (SaveMap)
        {
        cvSaveImage("Map_free_space.jpg", Peta1);
        cvSaveImage("Map_wall.jpg", Peta_dinding);
        SaveMap=false;
        }
       }
hr = NuiImageStreamReleaseFrame(depthStreamHandle,
pImageFrame);
if(cvWaitKey(10) == 0x001b)
                {
                        break;
                }

         } // End of while(1)
        cvReleaseImage(&DepthImage);
        cvReleaseImage(&Frame_Kinect);
        cvReleaseImage(&Buffer_DepthImage);
        cvReleaseImage(&gray);
        cvReleaseImage(&Frame_DepthImage);
        cvReleaseImage(&Showing_DepthImage);
        cvReleaseImage(&Buffer_Showing_DepthImage);
        cvReleaseImage(&Buffer_Showing_DepthImage2);
        cvReleaseImage(&Peta2);
        cvReleaseImage(&Peta1);
        cvReleaseImage(&Peta_dinding);
        cvReleaseImage(&dst);
        cvReleaseImage(&color_dst);
        cvReleaseImage(&ruangan);
        cvReleaseImage(&ruangan_buff);
        cvReleaseImage(&tes);
        cvReleaseImage(&ruang_full);

         //return 0;
                        }
//================================================//

//================== KIRIM DATA ==================//
private: void kirim_data(void)
                {
String^ name = this->serialPort1->PortName;
// grab text and store in send buffer
String^ message = this->textBox2->Text;
// write to serial
```

```
if(this->serialPort1->IsOpen)
this->serialPort1->WriteLine(
String::Format("{0:X}\n", message));
 else
this->textBox1->Text="Port Not Opened";
                }
//INIT BUTTON (Open Ports)
private: System::Void button4_Click(System::Object^
sender, System::EventArgs^  e) {
this->textBox2->Text=String::Empty;
if(this->comboBox1->Text==String::Empty || this-
>comboBox2->Text==String::Empty)
this->textBox1->Text="Please Select Port Settings";
else {
try{
if(!this->serialPort1->IsOpen){
this->serialPort1->PortName=this->comboBox1->Text;
this->serialPort1->BaudRate=Int32::Parse(this-
>comboBox2->Text);
this->textBox1->Text="Enter Message Here";
this->serialPort1->Open();
this->progressBar1->Value=100;
}
else
this->textBox2->Text="Port isn't openned";
}
       catch(UnauthorizedAccessException^){
       this->textBox2->Text="UnauthorizedAccess";
                                    }
                                }
                }
private: System::Void
serialPort1_DataReceived(System::Object^  sender,
System::IO::Ports::SerialDataReceivedEventArgs^  e) {
unsigned char data1, data2, data3, data4='0';
unsigned char data5, data6, data7, data8='0';
unsigned char data9, data10, data11, data12='0';
unsigned char data_a,data_b=0;
unsigned char data_mode;
time_click++;
if(serialPort1->ReadByte()=='q')
                                    {
data1=serialPort1->ReadByte();
data2=serialPort1->ReadByte();
```

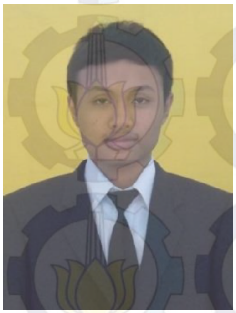
```
data3=serialPort1->ReadByte();
data4=serialPort1->ReadByte();
data_a=serialPort1->ReadByte();
data5=serialPort1->ReadByte();
data6=serialPort1->ReadByte();
data7=serialPort1->ReadByte();
data8=serialPort1->ReadByte();
data_b=serialPort1->ReadByte();
data9=serialPort1->ReadByte();
data10=serialPort1->ReadByte();
data11=serialPort1->ReadByte();
data12=serialPort1->ReadByte();
data_mode=serialPort1->ReadByte();
real_data  = ((data2-48)*1000) + ((data3-48)*100) +
((data4-48)*10) + ((data_a-48)*1) ;
real_data2 = ((data6-48)*1000) + ((data7-48)*100) +
((data8-48)*10) + ((data_b-48)*1) ;
real_data3 = ((data10-48)*100) + ((data11-48)*10) +
((data12-48)*1) ;
mode = data_mode-48;
POS_X=real_data; POS_Y=real_data2; heading=real_data3;
if ( (data1-48) == 2 ) POS_X = real_data  * (-1);
if ( (data5-48) == 2 ) POS_Y = real_data2 * (-1);
if ( (data9-48) == 2 ) heading = real_data3 * (-1);
if (mode==0){
status_aksi = 999; //status_aksi=0;
       time_click=0;
       flag_time_click=0;
       kali_maju=0;
       kali_diam=0;
       flag_kirim=0;
       MODE = mode;
       X_POS = POS_X;
       Y_POS = POS_Y;
       ANGLE = heading;
       data_count++;
        }
 }
//----- for checking is there any data has been update
(TIMER 100 ms) ----//
private: System::Void timer1_Tick(System::Object^
sender, System::EventArgs^  e) {
this->label18->Text=Convert::ToString(data_count*10)+
" reads/s";
```

```
                    data_count=0;
                    time_click++;
                }
//--------------- TOMBOL Saving Map -------------//
private: System::Void button8_Click(System::Object^
sender, System::EventArgs^  e) {
                    SaveMap = true;
                }
//------------ TOMBOL set PWM --------------------//
private: System::Void button10_Click(System::Object^
sender, System::EventArgs^  e) {
String^ buffer_textBox1 = textBox7->Text;
pwm_max_maju =
(int)(Convert::ToDouble(buffer_textBox1));
String^ buffer_textBox2 = textBox8->Text;
pwm_min_maju =
(int)(Convert::ToDouble(buffer_textBox2));
String^ buffer_textBox3 = textBox16->Text;
pwm_kanan = (int)(Convert::ToDouble(buffer_textBox3));
String^ buffer_textBox4 = textBox17->Text;
pwm_kiri  = (int)(Convert::ToDouble(buffer_textBox4));
String^ buffer_textBox5 = textBox18->Text;
delay_belok =
(int)(Convert::ToDouble(buffer_textBox5));
String^ buffer_textBox6 = textBox19->Text;
delay_maju =
(int)(Convert::ToDouble(buffer_textBox6));
String^ buffer_textBox7 = textBox20->Text;
delay_kanan =
(int)(Convert::ToDouble(buffer_textBox7));
 }


//------------
```

*#Halaman ini sengaja dikosongkan#*

# BIODATA PENULIS



Novem Ardan Rohmadin dilahirkan di Surabaya, 22 November 1993. Anak kedua dari tiga bersaudara dari pasangan Riyanto dan Siti Musthofiah. Penulis menyelesaikan pendidikan dasar di SDN Wonorejo 274 Surabaya, dilanjutkan dengan pendidikan menengah di SMPN 12 Surabaya dan SMAN 15 Surabaya. Pada tahun 2011, penulis memulai pendidikan di jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah penulis aktif membantu penyelenggaraan kegiatan dan aktif sebagai asisten laboratorium Elektronika Dasar dan praktikum Elektronika pada semester ganjil 2014-2015, serta aktif dalam Unit Kegiatan Mahasiswa ITS.

Email :

novem11@mhs.ee.its.ac.id

novemrohmadin@gmail.com

# IMPLEMENTASI METODE *WALL FOLLOWING* BERBASIS KINECT UNTUK ROBOT *MOBILE*

Novem Ardan Rohmadin, Ronny Mardiyanto, ST.,MT.,Ph.D[1)], Rudy Dikairono, S.T., M.T.[2)]
Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
*e-mail*: novemrohmadin@gmail.com, ronny@ee.its.ac.id [1)], rudydikairono@ee.its.ac.id[2)]

***Abstrak* – Navigasi robot *mobile* merupakan cara dalam mengatur gerakan robot *mobile*. Pada robot dengan tugas khusus seperti robot pencari benda berbahaya yang digerakkan otomatis di dalam ruangan, robot membutuhkan sistem navigasi yang mampu mengatur gerakan robot untuk menyusuri ruangan. Metode *wall following* adalah suatu metode dalam sistem navigasi robot *mobile* yang dapat digunakan untuk kebutuhan tersebut. Pada metode tersebut, robot akan bergerak berdasarkan jarak dari dinding di sebelah robot. Sehingga, robot membutuhkan sensor jarak yang akurat.**

**Pada tugas akhir ini, dibuat suatu rancangan yang berjudul "*Implementasi Metode Wall Following berbasis Kinect untuk Robot Mobile*". Sistem yang dirancang memanfaatkan kamera kinect sebagai input sensor jarak. *Kinect* merupakan kamera RGB-D buatan Microsoft. Piksel kedalaman yang dihasilkan kinect dapat diolah untuk menavigasikan robot *mobile*.**

**Dalam tugas akhir ini, robot akan menelusuri ruangan menggunakan kinect sebagai input jarak, rotari enkoder dan sensor kompas untuk menunjukkan posisi robot. Robot akan terus menelusuri ruangan dengan mengambil referensi jarak dinding di sebelah kiri hingga *user* menghentikan gerakan robot melalui remote kontrol. Pengujian dilakukan dengan menjalankan robot secara otomatis di dalam suatu ruangan. Ruangan akan diberi halangan untuk membagi ruangan menjadi beberapa bagian. Dari pengujian, semakin banyak penambahan halangan untuk membagi ruangan, maka kemampuan telusur robot akan berkurang. Berdasarkan pengamatan dari pengujian pada ruang uji yang didesain, sistem yang dirancang memiliki error rata-rata 7,33% terhadap kemampuan robot dalam memasuki bagian ruangan. Lebar celah ruangan menjadi faktor kegagalan robot dalam memasuki bagian ruangan.**

**Kata kunci : Kinect, robot *mobile*, navigasi**

## I. PENDAHULUAN

Saat ini robot *mobile* telah banyak diaplikasikan dalam kehidupan manusia. Robot telah diaplikasikan dalam bidang hiburan, kebersihan, pencarian benda berbahaya, hingga pencarian korban dalam bencana. Kemampuan robot *mobile* akan selalu dikembangkan untuk memenuhi kebutuhan manusia. Beberapa kemampuan robot *mobile* yang dikembangkan adalah kemampuan robot untuk bergerak secara otomatis. Kemampuan robot *mobile* seperti ini diterapkan pada robot yang bertugas untuk menjelajahi suatu tempat yang tidak dikenali, seperti robot pencari benda berbahaya. Metode yang dapat digunakan dalam menjelajahi ruangan yang tidak dikenali adalah *wall following*. Dengan mengetahui jarak antara robot dengan dinding ruangan, pergerakan robot dapat diatur. Robot akan bergerak mengikuti dinding ruangan sehingga robot mampu memasuki ruangan yang ada.

Penggunaan robot *mobile* dalam mencari benda berbahaya pernah digunakan oleh DENSUS 88. Robot MoroLIPI adalah nama robot tersebut. Kasus dimana robot MoroLIPI digunakan adalah pada saat misi penangkapan teroris Noordin M Top. Robot dengan bentuk seperti tank tersebut bertugas untuk mengetahui situasi didalam ruangan dimana Noordin M Top bersembunyi. Robot bertugas untuk mendeteksi apakah ada benda berbahaya seperti bom yang dapat membahayakan petugas. Pergerakan robot MoroLIPI saat menyusuri ruangan saat itu masih dikendalikan oleh operator secara manual dari jarak jauh. Proses navigasi robot yaitu pergerakan robot *mobile* dalam menyusuri ruangan saat mencari benda berbahaya dapat dikembangkan menjadi sistem otomatis.

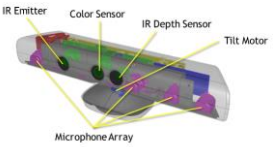
Robot *mobile* dengan tujuan menyusuri suatu lingkungan yang tidak dikenal seringkali menggunakan sensor jarak 3 dimensi sebagai input untuk mengatur gerakan robot ataupun hanya untuk pengambilan data. LIDAR (*Light Detecting and Ranging*) adalah salah satu sensor yang sering digunakan sebagai input dalam pemetaan dan navigasi robot. Namun harga LIDAR yang mahal menjadi faktor penghambat bagi para pengembang robotika untuk membeli sensor tersebut.

Dalam kasus ini kinect dapat menjadi solusi untuk memenuhi permasalahan diatas. Kinect adalah sensor jarak 3 dimensi yang dilengkapi dengan kamera RGB. Sensor buatan Microsoft ini memiliki harga yang lebih murah dibandingkan dengan LIDAR. Kinect dapat menggantikan fungsi LIDAR dalam proses navigasi dan pemetaan.

Oleh karena itu, diajukanlah proposal tugas akhir ini dengan harapan agar dapat mengimplementasikan metode *wall following* berbasis kinect pada robot *mobile* sehingga robot mampu menelusuri ruangan secara otomatis.

## II. TEORI PENUNJANG

### A. *Kamera Kinect*




**Gambar 2. 1** Kamera Kinect[1]

Kinect adalah perangkat yang digunakan sebagai pendeteksi gerakan yang awalnya digunakan sebagai kontroler permainan Xbox 360.Kinect dibangun dari teknologi perangkat lunak yang didirikan oleh Rare, anak perusahaan Microsoft Game Studios. Sensor kamera pada kinect dikembangkan oleh perusahaan dari Israel, yaitu Primesense. Kamera Kinect dapat menyediakan data *RGB-Depth* dengan kecepatan fps maksimal 30 Hz dengan resolusi 640 x 480 piksel. Kamera Kinect memiliki sudut pandang 57 derajat pada bidang horizontal dan 43 derajat pada bidang vertikal. Jarak optimal kinect saat digunakan oleh software Xbox adalah 1.2 - 3.5 meter. Apabila jarak yang digunakan melebihi atau kurang dari jarak yang ditentukan, maka obyek tidak akan tertangkap oleh kamera. Pada bagian bawah kinect terdapat *multi-array mic* yang digunakan untuk merekam atau menginputkan suara. Kinect juga dilengkapi dengan *motorized tilt* untuk mengatur sudut kamera, sehingga area yang bisa ditangkap oleh kamera dapat diatur. Untuk mengatur sudut kamera bisa menggunakan program tertentu, dengan jangkauan sudut kurang lebih 27 derajat.

### B. *Pembacaan Posisi Robot*

Dalam navigasi dan pemetaan menggunakan robot, perlu diketahui posisi dan arah robot dalam bidang kartesian. Konsep basis untuk mengetahui posisi dan arah adalah dengan cara membaca jarak yang telah ditempuh robot setiap roda. Untuk robot dengan mekanik diferensial, yakni kendali gerakan robot berdasarkan pergerakan 2 roda. Persamaan 2.1, 2.2, 2.3, 2.4, dan 2.5 digunakan untuk menghitung posisi dan arah robot.

$$\Delta S = \frac{\Delta S_r + \Delta S_l}{2} \quad (2.1)$$

$$\Delta\theta = \frac{\Delta S_r - \Delta S_l}{B} \quad (2.2)$$

$$X = X + \Delta S \cos\theta \quad (2.3)$$

$$Y = Y + \Delta S \sin\theta \quad (2.4)$$

$$\theta = \theta + \Delta\theta \quad (2.5)$$

Keterangan:

$\Delta S$ = Jarak tempuh terbaru robot
$\Delta S_r$ = Jarak tempuh terbaru roda kanan
$\Delta S_l$ = Jarak tempuh terbaru roda kiri
$\Delta\theta$ = Perubahan sudut terbaru
$B$ = Jarak antara roda kiri dan kanan
$X$ = Posisi robot pada bidang x
$Y$ = Posisi robot pada bidang y
$\theta$ = Arah sudut robot

### C. *Mirkokontroler Arduino*

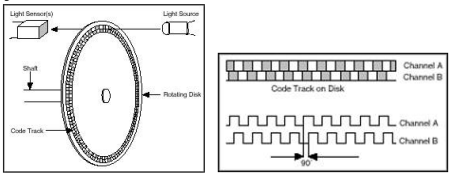

**Gambar 2.2** Arduino Mega [2]

Arduino merupakan suatu platform dari *physical computing* yang bersifat *open source*. Arduino bukan hanya sebagai modul mikrokontroler, namun arduino adalah kombinasi dari hardware, bahasa pemrograman dan *Integrated Development Environment* (IDE) yang canggih. IDE adalah sebuah *software* yang sangat berperan untuk menulis program, meng-*compile* menjadi kode biner dan meng-upload ke dalam memori mikrokontroler. Ada banyak projek dan alat-alat dikembangkan oleh akademisi dan profesional dengan menggunakan Arduino, selain itu juga ada banyak modul-modul pendukung (sensor, tampilan, penggerak dan sebagainya) yang dibuat oleh pihak lain untuk bisa disambungkan dengan Arduino.

### D. *Wall Following*

*Wall following* adalah suatu metode navigasi yang terkenal di kalangan para peneliti robot khususnya di bidang *reactive* robot. Dengan menggunakan sensor jarak, ketika robot terlalu dekat dengan dinding maka robot akan bergerak menjauh dari dinding dan ketika robot terlalu jauh dengan dinding maka robot akan bergerak mendekat ke dinding. Pengimplementasian *wall following* mampu menggunakan langkah-langkah sederhana seperti maju, lihat, dan belok atau menggunakan tambahan sistem kontrol.

### E. *Rotary Encoder*




**Gambar 2.4** Rotari Enkoder[3]

Rotari enkoder adalah divais elektromekanik yang dapat memonitor gerakan dan posisi. Rotari enkoder umumnya menggunakan sensor optik untuk menghasilkan sinyal pulsa yang dapat diartikan menjadi posisi.

Rotari enkoder tersusun dari suatu piringan tipis yang memiliki lubang-lubang pada bagian lingkaran piringan. LED ditempatkan pada salah satu sisi piringan sehingga cahaya akan menuju ke piringan. Di sisi yang lain phototransistor diletakkan sehingga photo-transistor ini dapat mendeteksi cahaya dari LED yang berseberangan. Apabila posisi piringan mengakibatkan cahaya dari LED dapat mencapai photo-transistor melalui lubang-lubang yang ada, maka photo-transistor akan mengalami saturasi dan akan menghasilkan suatu pulsa gelombang persegi.

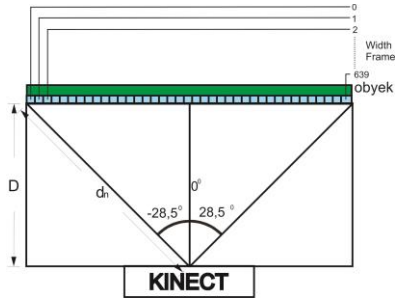
Semakin banyak deretan pulsa yang dihasilkan pada suatu putaran menentukan akurasi rotari enkoder tersebut.

F. *Sensor Kompas*

Kompas adalah alat yang berfungsi untuk menunjukkan arah hadap suatu objek[4]. Kompas merupakan bagian penting dari pembacaan posisi dan arah, sebagai sensor yang dapat mengestimasi arah hadap robot. Dalam bidang navigasi, arah hadap robot merupakan bagian yang penting. Pada sensor kompas terdapat unit proses pusat dan 2 magnet yang masing-masing kutubnya dihadapkan ke arah mata angin yang berbeda, sehingga arah hadap sensor dapat diketahui.

G. *Pembuatan Peta menggunakan Kinect*

Pembuatan peta dua dimensi hasil pemetaan *mobile robot* dengan *tracking mobile robot* pada ruangan yang akan dilewati *mobile robot* membutuhkan data jarak obyek disekitar *mobile robot* dan data posisi *mobile robot* pada bidang kartesian[5]. Data jarak diperoleh dengan menggunakan sensor kinect sedangkan data posisi *mobile robot* diperoleh dengan menggunakan rotari enkoder dan kompas.




**Gambar 2.5** Pengambilan Data Jarak dalam pembuatan peta

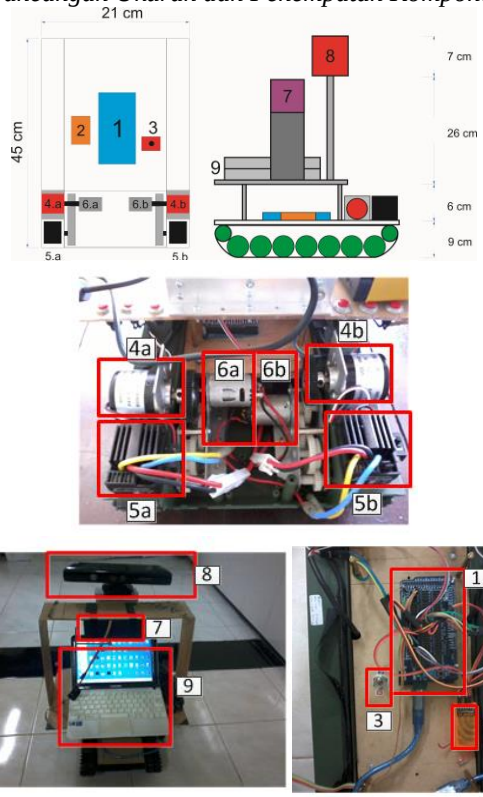
H. *Electronic Speed Controller*

**Gambar 2.6** Electronic Speed Controller

ESC (*Electronic Speed Control*) adalah divais yang berfungsi untuk mengatur kecepatan putar motor menggunakan sinyal PWM dengan frekuensi dan *duty cycle* tertentu. ESC digunakan pada kendaraan radio kontrol. ESC menggunakan sinyal persegi dengan frekuensi 50 Hz dengan lebar pulsa antara 1ms hingga 2ms. Lebar pulsa sinyal yang dimasukkan pada ESC akan mempengaruhi kecepatan putar motor dan arah putar motor. Sinyal PWM dengan lebar pulsa antara 1ms dan kurang dari 1,5ms akan mengakibatkan motor berputar berlawanan arah jarum jam. Sinyal PWM dengan lebar pulsa 1,5ms akan mengakibatkan motor diam. Sinyal PWM dengan lebar pulsa antara 1,5ms dan kurang dari 2ms akan mengakibatkan motor berputar searah jarum jam.

## III. DESAIN SISTEM DAN IMPLEMENTASI

A. *Perancangan Ukuran dan Penempatan Komponen Robot*

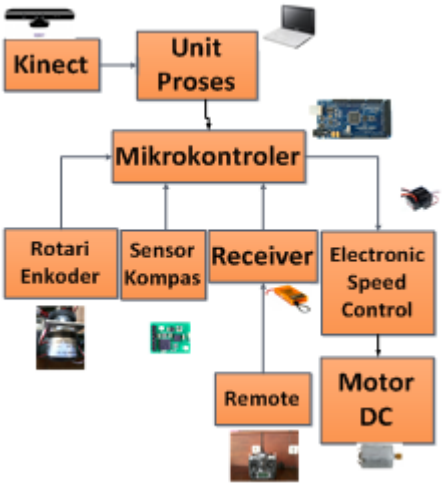



**Gambar 3. 1** Ukuran dan Penempatan Komponen Robot

Keterangan :

1. Arduino Mega
2. *Receiver*
3. Tombol On/Off
4. Rotari Enkoder : (a) kiri, (b) kanan
5. ESC (*Electronic Speed Controller*) : (a) kiri, (b) kanan
6. Motor DC : (a) kiri, (b) kanan
7. Sensor kompas
8. Kinect
9. Unit Proses (PC)

*Mobile* robot yang dirancang, memiliki bentuk dasar seperti kendaraan tank dengan 2 buah roda track yang dihubungkan dengan motor dc sebagai penggerak utama. Kinect ditempatkan dibagian atas robot agar robot mampu membaca jarak obyek dan lantai di depan robot. Pembacaan jarak pada lantai dianggap perlu agar robot mengetahui perbedaan antara ada tidaknya halangan di depan robot.



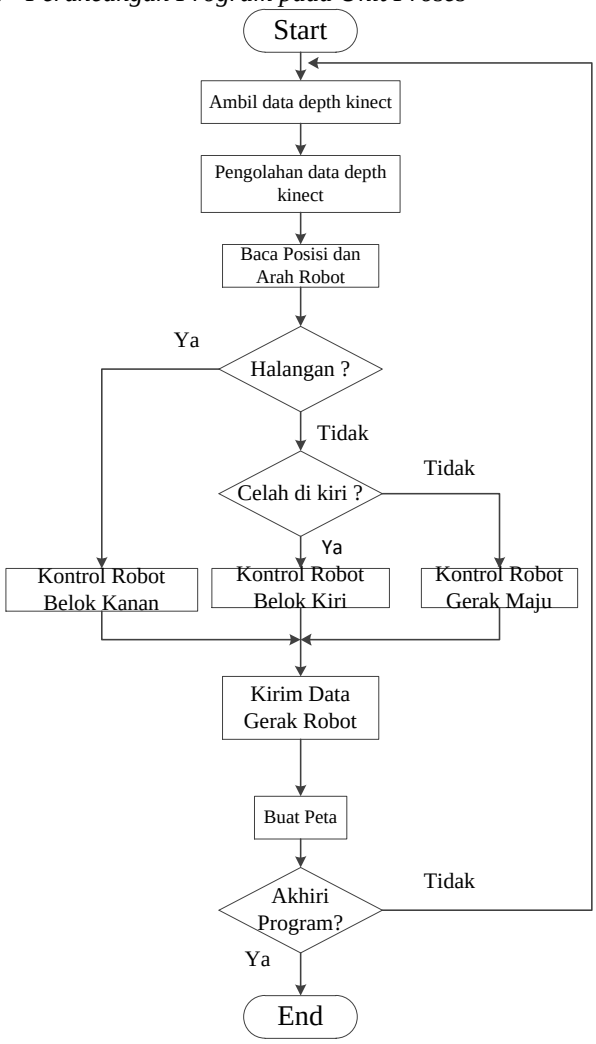

**Gambar 3. 2** Blok Diagram Sistem Robot

Robot dikendalikan oleh seorang *user* menggunakan radio kontrol. Sinyal dari remote kontrol akan diterima oleh *receiver*. Robot akan membaca sinyal dari *receiver*. Berdasarkan sinyal yang dibaca oleh *receiver*, mode gerak robot akan ditentukan. Terdapat 2 mode, yaitu manual dan otomatis. Saat mode auto, gerakan robot diatur sepenuhnya oleh unit proses. *User* memilih mode melalui saklar pada remote. Rotari enkoder dan kompas membaca kondisi robot, baik posisi maupun arah. Data posisi dan arah dikirim ke unit proses. Kinect membaca kondisi lingkungan berupa jarak obyek di depan robot. Data jarak kinect diambil dan diolah oleh unit proses. Pertama kali robot bergerak, robot akan menelusuri dinding sebelah kiri. Unit proses akan bergerak maju untuk dan selanjutnya memeriksa apakah ada celah disebelah kiri robot. Jika terdapat celah maka robot akan bergerak masuk ke celah, jika tidak maka robot akan kembali maju pada arah sebelumnya.

Selama robot bergerak, unit proses akan menghasilkan peta dua dimensi berdasarkan jarak obyek di depan robot, posisi dan arah robot. Peta berisi kondisi lingkungan di sekitar robot secara 2 dimensi. Peta ini merupakan fitur tambahan bagi robot *mobile*. Hasil ketentuan gerakan robot yang dihasilkan oleh unit proses akan dikirim ke mikrokontroler. Robot akan terus bergerak sesuai dengan intruksi unit proses. Robot akan berjalan secara otomatis hingga terdapat intruksi kembali ke mode manual dari *user* melalu radio kontrol.

Saat mode manual, gerakan robot diatur sepenuhnya oleh *user* melalui joystick remote. Remote kontrol akan disetting sehingga gerakan maju, mundur, belok kanan dan belok kiri robot akan dikendalikan menggunakan joystick remote kontrol. *User* akan mengendalikan robot dari jarak jauh. Penggunaan remote kontrol bertujuan agar robot mampu dikendalikan dari jarak yang aman dari lokasi penggunaan robot.

B. *Perancangan Program pada Unit Proses*



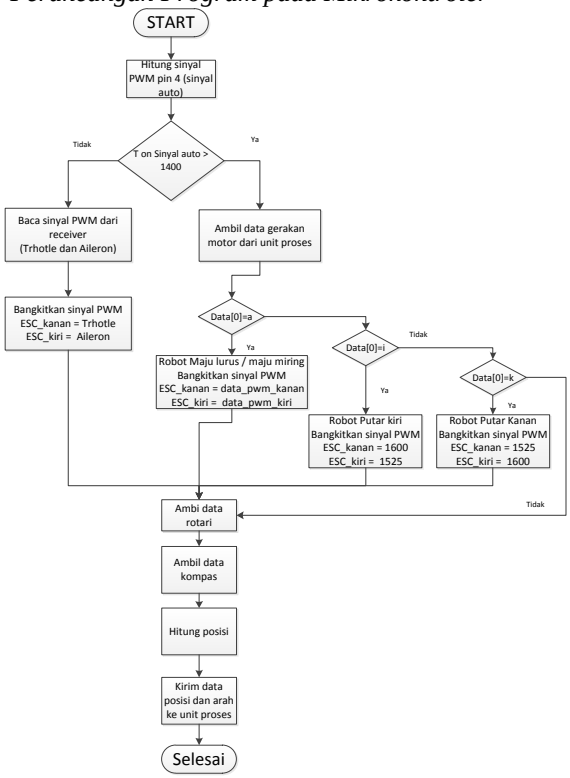

**Gambar 3. 3** Flowchart pada unit proses

Pada gambar 3.17, ditampilkan flowchart mengenai pergerakan utama robot. Penjelasan flowchart diatas dijelaskan melalui poin-poin berikut :

1. Robot mula - mula akan mengambil data depth dari kamera kinect. Data depth yang diambil ini berupa matriks 640x1 yang berisi data kedalaman yang ditangkap oleh kamera kinect.
2. Data depth dari kinect akan diolah dan dibagi menjadi beberapa region. Nilai data kedalaman rata-rata per region akan dibandingkan untuk mengetahui apakah robot menemui halangan, celah di kiri robot, atau jalan depan kosong dan robot berada dekat dinding.
3. Jika jalan depan kosong dan robot berada dekat dinding, maka robot akan dikontrol untuk bergerak maju.
4. Jika terdapat dinding di kiri robot, maka robot akan dikontrol untuk bergerak ke maju. Jika tidak ada dinding maka robot akan bergerak ke kiri.
5. Jika terdapat halangan di depan robot maka robot akan dikontrol untuk segera belok ke arah kanan. Aksi ini akan berhenti jika robot menemui keadaan seperti pada urutan nomor 3 dan 4.

6. Saat robot bergerak, robot akan sekaligus menggambar peta lingkungan yang ada di sekitarnya.
7. Robot akan keluar dari mode otomatis dan kembali ke mode manual jika user menghentikan gerakan robot menggunakan remote kontrol.

C. *Perancangan Program pada Mikrokontroler*




**Gambar 3. 4** Flowchart pada arduino

Pemilihan mode otomatis atau manual dilakukan dengan membaca sinyal PWM yang dihasilkan oleh pin aux pada receiver. Pin aux akan menghasilkan sinyal dengan frekuensi 50 Hz dan Ton antara 1ms dan 2ms.

Mode auto adalah mode dimana unit proses mengkontrol penuh gerakan robot. Pergerakan robot sesuai dengan data yang diterima mikrokontroler. Data ‘a’berarti robot akan maju dengan kecepatan tertentu. Data ‘k’, maka robot akan bergerak rotasi ke kanan. Dan data ‘i’, maka robot akan bergerak rotasi ke kiri.

Pada mode manual, mikrokontroler perlu membaca sinyal pwm dari *receiver*. Sesuai dengan konfigurasi hardware, pin trho menghasilkan sinyal untuk motor kanan, dan pin aile menghasilkan sinyal untuk motor kiri.Sinyal yang sudah dibaca selanjutnya perlu dibangkitkan kembali sesuai dengan sinyal yang dihasilkan remote.

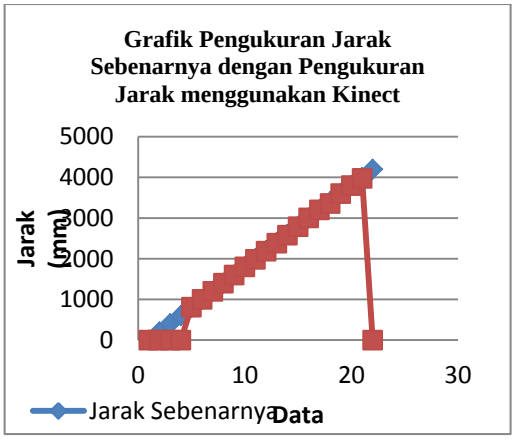
Setiap motor bergerak maju maupun mundur, rotari enkoder akan mengirimkan sinyal pulsa. Semakin cepat motor bergerak maka frekuensi sinyal akan semakin cepat. Pergerakan motor dan track robot sebanding, sehingga ketika track bergerak maka rotari enkoder akan menghasilkan sinyal pula. Sinyal dari rotari enkoder akan dihitung setiap saat sehingga diketahui jarak yang telah ditempuh oleh robot. Sinyal dari rotari enkoder dibaca menggunakan interrupt eksternal pada mikrokontroler arduino.

Pengambilan data kompas, dilakukan dengan menggunakan komunikasi I2C antara mikrokontroler dan sensor kompas CMPS11. Dengan menghitung sinyal masuk dari rotari enkoder, maka jarak yang ditempuh oleh robot dapat diketahui. Dengan menggunakan jarak tempuh (ΔS) untuk roda kanan dan roda kiri dan arah robot yang berasal dari kompas. Robot dapat mengetahui posisi dan arah.

## IV. PENGUJIAN

**A.** *Pengujian Pembacaan Jarak Obyek dengan Kinect*

Pengujian yang dilakukan pada bagian ini adalah untuk mengetahui karakteristik pembacaan jarak estimasi yang dilakukan oleh kinect. Pengujian dilakukan dengan mencatat hasil pembacaan jarak terhadap obyek oleh kinect pada pusat frame *depth*. Pencatatan dilakukan pada jarak 20 cm hingga 4200 cm dari kamera kinect dengan interval 20 cm. Berikut adalah pengujian yang telah dilakukan :




**Gambar 4. 1** Grafik Pengukuran Jarak Sebenarnya dengan Pengukuran Jarak menggunakan Kinect

Dari data pengujian tersebut dapat diketahui bahwa akurasi pengukuran jarak oleh kinect pada jarak kurang dari 80 cm dan lebih dari 4 meter, pengukuran jarak memiliki error rata-rata sebesar 100 %.Sedangkan pada jarak antara lebih dari 80 cm hingga kurang dari 4 meter pengukuran jarak memiliki error rata-rata sebesar 0.197 %. Berdasarkan pengujian ini dapat diketahui bahwa pengukuran jarak menggunakan kamera kinect baik untuk jarak pada range 80 cm hingga 4 meter.

**B.** *Pengujian Kemampuan Robot dalam menelusuri ruangan*

Salah satu tujuan perancangan sistem pada tugas akhir ini adalah agar robot mampu menelusuri ruangan, sehingga perlu dilakukan pengujian kemampuan robot dalam menelusuri ruangan. Selanjutnya ruangan akan dibagi menjadi beberapa ruang.Pembagian jumlah ruang dilakukan menggunakan sekat. Robot akan dibiarkan berjalan secara otomatis dan akan dicatat waktu yang dibutuhkan robot untuk memasuki masing-masing bagian ruang.

**Tabel 4.1** Tabel Pengujian Kemampuan Jelajah Robot

| Ruang | Bagian | Waktu Per Bagian (detik) | Total Ruang yang Dimasuki (%) |
|---|---|---|---|
| 1 | 1 | 0 | 100 |
| 2 | 1 | 0 | 100 |
| | 2 | 22 | |
| 3 | 1 | 0 | 100 |
| | 2 | 16 | |
| | 3 | 33 | |
| 4 | 1 | 0 | 100 |
| | 2 | 15 | |
| | 3 | 27 | |
| | 4 | 54 | |
| 5 | 1 | 0 | 80 |
| | 2 | 14 | |
| | 3 | 22 | |
| | 4 | 47 | |
| | 5 | - | |
| 6 | 1 | 0 | 83.33 |
| | 2 | 13 | |
| | 3 | 19 | |
| | 4 | 40 | |
| | 5 | 62 | |
| | 6 | - | |

Selama pengujian, robot berusaha mengikuti dinding di sebelah kiri robot. Selanjutnya dianalisa pula kemampuan jelajah robot. Kemampuan jelajah robot per bagian ruang menunjukkan presentase keberhasilan robot dalam menelusuri ruang yang ada. Dari data tersebut dapat diketahui hubungan antara jumlah ruang dengan waktu yang dibutuhkan robot untuk menelusuri seluruh ruangan. Pada kondisi ruangan yang terdiri dari 1 bagian hingga 4 bagian, sistem mampu menelusuri semua bagian. Namun pada ruangan dengan 5 bagian, presentase robot untuk menelusuri robot turun menjadi 80%. Presentase keberhasilan robot untuk memasuki bagian ruangan menjadi 83,33% pada ruangan dengan 6 pembagian.

Setiap ruangan membutuhkan waktu yang berbeda untuk memasuki semua ruangan. Ruang 1 membutuhkan waktu 0 detik, ruang 2 membutuhkan waktu 22 detik, ruang 3 membutuhkan waktu 33 detik, dan ruang 4 membutuhkan waktu 54 detik untuk memasuki semua bagian ruangan. Sedangkan ruang 5 dan ruang 6, robot tidak dapat memasuki semua ruangan. Ruang 5, robot membutuhkan waktu 57 detik untuk memasuki 4 bagian ruangan. Dan ruang 6, robot membutuhkan waktu 62 detik untuk memasuki 5 bagian ruangan.

## V. KESIMPULAN

Kesimpulan yang diperoleh pada tugas akhir ini adalah :

1. Dalam menelusuri ruangan, robot menjaga jarak dengan dinding sebelah kiri dengan rata – rata jarak dari dinding dalam set point jarak dari dinding sebesar 108 cm adalah 80,36 cm dengan error rata – rata sebesar 35,88%. Sedangkan dengan set point 85 cm, robot telah mengikuti dinding dengan jarak rata-rata dari dinding sebesar 65,73 cm dengan error rata-rata sebesar 26,3 %. Dari data tersebut diketahui bahwa error rata – rata yang dimiliki robot dalam menjaga jarak dengan dinding kiri sebesar 31,09 % berdasarkan set point jarak yang digunakan.
2. Berdasarkan pengujian menggunakan ruang uji, keberhasilan rata-rata robot untuk menelusuri bagian dari ruangan sebesar 92,67%.
3. Error 7,33% disebabkan oleh ketidakmampuan robot untuk masuk kedalam beberapa bagian ruangan dikarenakan ukuran celah ruangan yang kecil. Pada kondisi ruangan seperti itu, sistem akan mengaggap celah tersebut sebagai halangan sehingga robot akan mengantisipasi dengan cara menjauhkan diri dari celah tersebut.
4. Penambahan pembagian ruangan pada ukuran yang sama akan semakin menurunkan kemampuan robot dalam menelusuri ruangan. Hal ini disebabkan oleh jarak pembacaan jarak oleh kamera kinect yang terbatas yaitu 80 cm hingga 4 meter.

## DAFTAR PUSTAKA


[1] Microsoft Developer Network Library. “Kinect for Windows Sensor Components and Specifications”. <URL: http://msdn.microsoft.com/en-us/library/jj131033.aspx> Desember, 2010.

[2] “__”. “Arduino Mega”. http://www.arduino.cc/en/Main/arduinoBoardMega diakses 14 Mei 2015.

[3] “__”,”*Encoder Measurements: How-To Guide*”. National Instrument. 2013.

[4] Purnomo, Didik Setyo et al. “*Navigation and Control System of Quadrotor Helicopter*". Electronics Engineering Polytechnic Institute of Surabaya (EEPIS): Indonesia. “__”.

[5] Anggoro D.A., Janu et al. “*Perancangan Sistem Pemetaan Otomatis Pada Mobile Robot Menggunakan Sensor Kinect dan Rotari Encoder*”.Institut Teknologi Sepuluh Nopember:Surabaya. 2012.

# IMPLEMENTASI METODE WALL FOLLOWING BERBASIS KINECT PADA ROBOT MOBILE

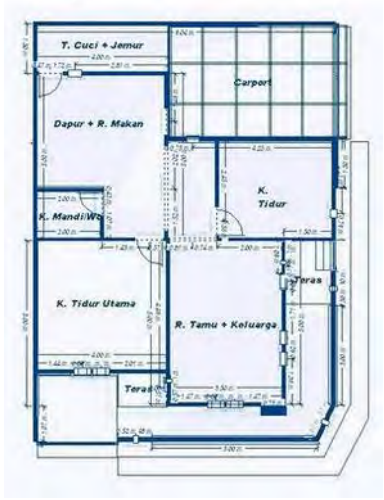
**Novem Ardan Rohmadin - 2211100051**

**Dosen Pembimbing :**
**Ronny Mardiyanto, ST.,MT.,Ph.D**
**Rudy Dikairono, S.T., M.T.**

# Contents

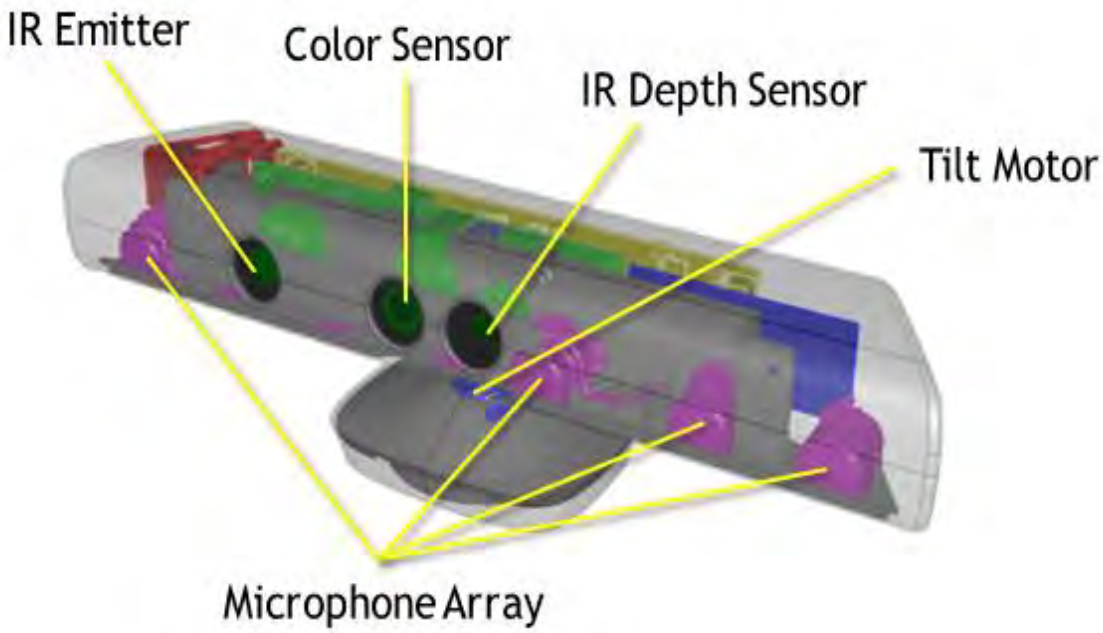
# Latar Belakang

DENSUS 88




PENGGEREBEKAN

# Latar Belakang



# Rumusan Masalah

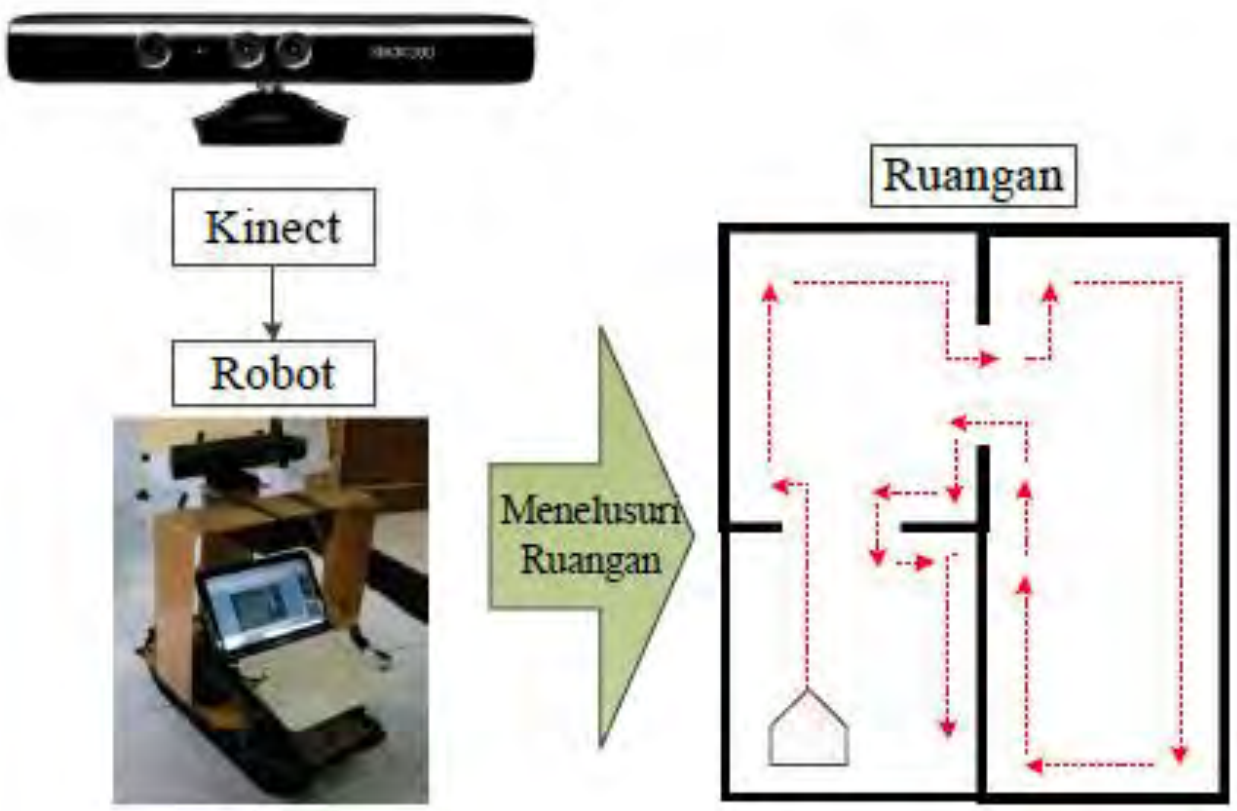
1. Bagaimana merancang suatu sistem navigasi robot *mobile* untuk menelusuri ruangan yang tidak diketahui menggunakan kinect.
2. Bagaimana merancang sistem yang dapat memandu robot untuk menghindari halangan.

# Tujuan Penelitian

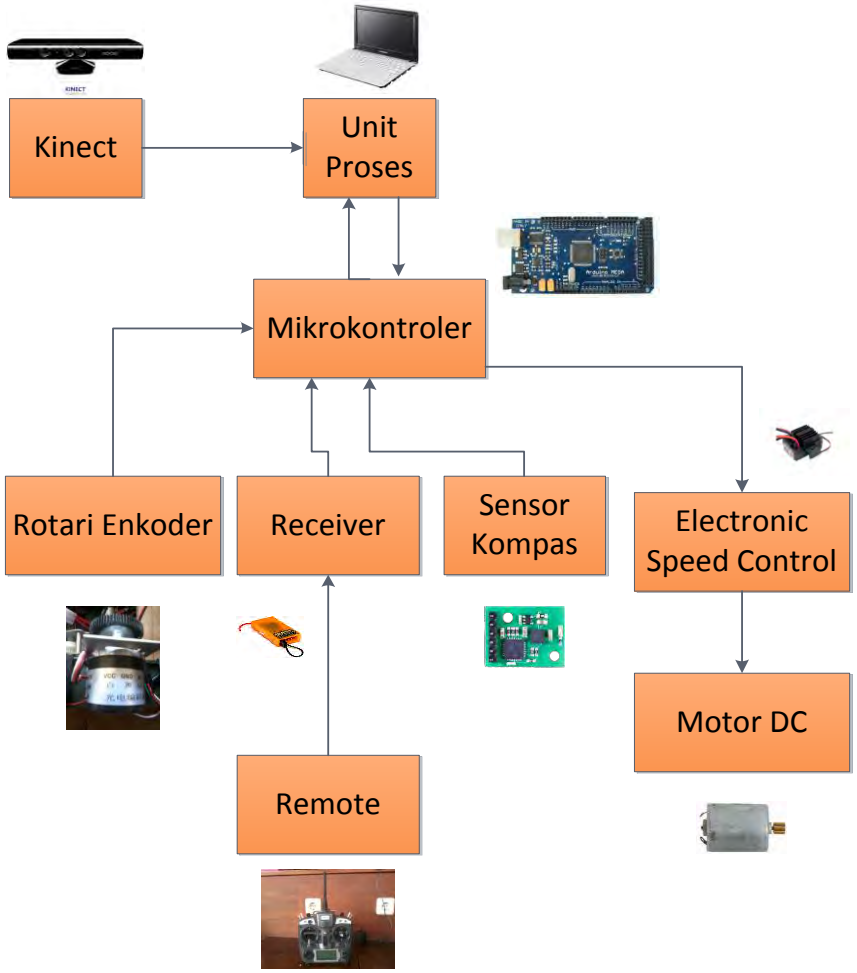
Penelitian pada tugas akhir ini bertujuan sebagai berikut :

1. Robot mampu menelusuri ruangan secara otomatis.
2. Robot mampu melewati celah ruangan.
3. Robot mampu menghindari halangan menggunakan kamera kinect pada saat robot bergerak otomatis.

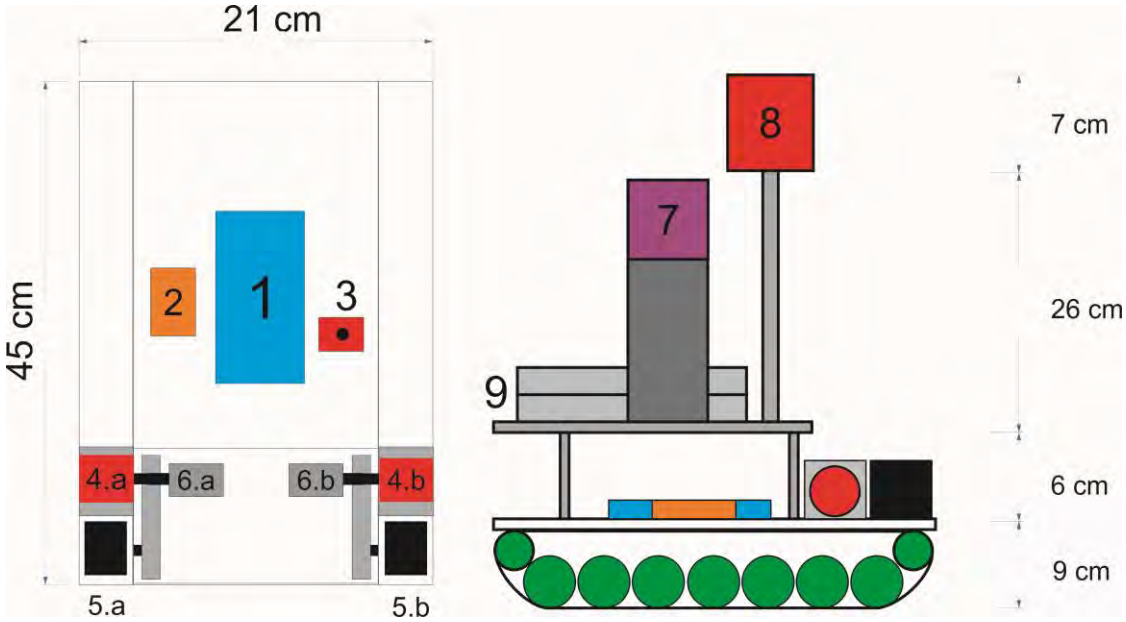
# Ilustrasi Sistem



# Diagram Blok

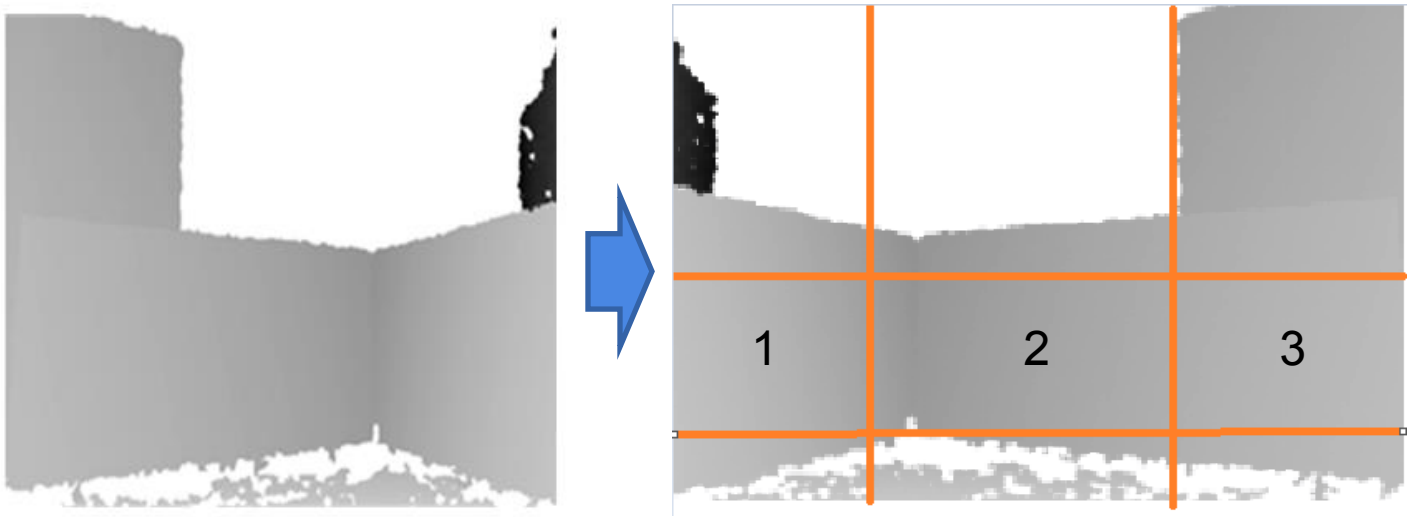


# Desain Robot

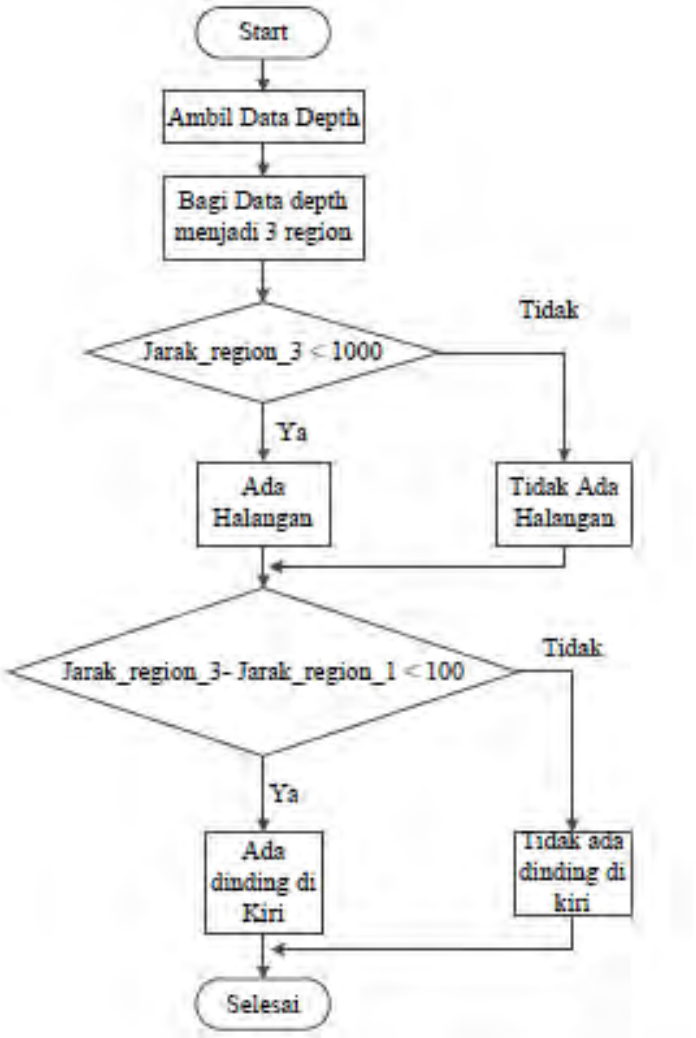


# Pendeteksian Dinding

Pengambilan data depth kinect untuk mendeteksi ada tidaknya halangan

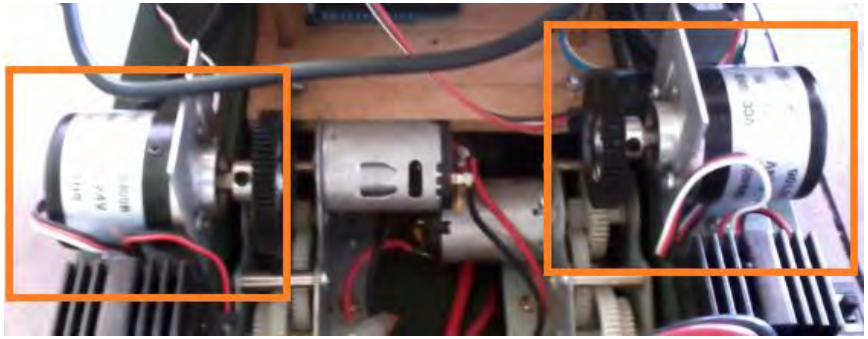


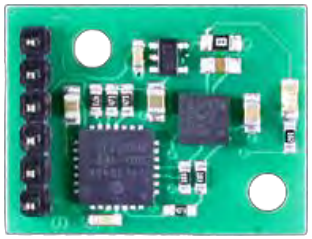
# Pendeteksian Dinding



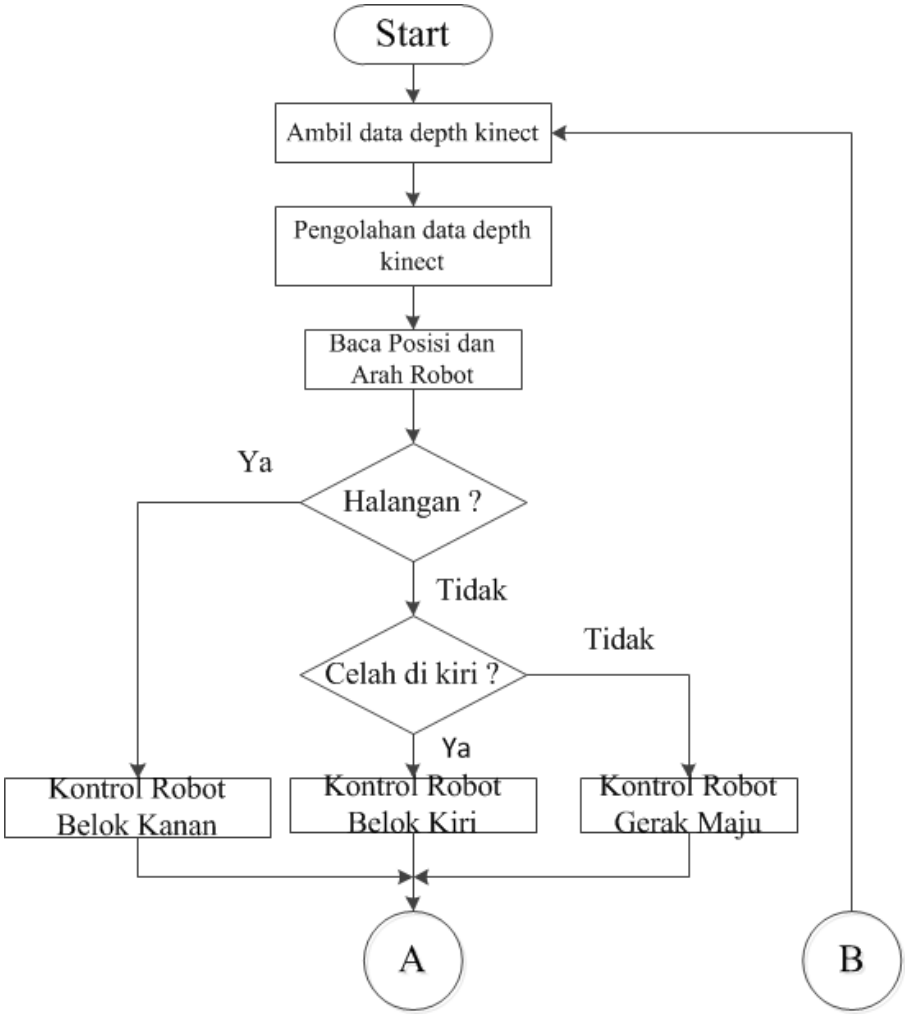
# Pembacaan Posisi dan Arah Robot

$$\Delta S = \frac{\Delta S_r + \Delta S_l}{2}$$
$$\Delta\theta = \frac{\Delta S_r - \Delta S_l}{B}$$
$$X = X + \Delta S \cos\theta$$
$$Y = Y + \Delta S \sin\theta$$
$$\theta = \theta + \Delta\theta$$

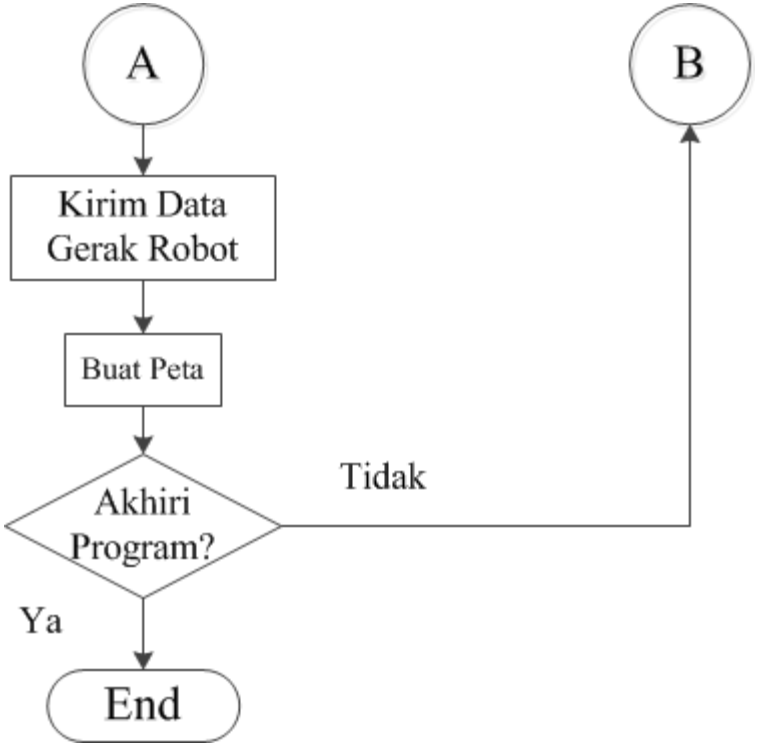




# Algoritma Utama

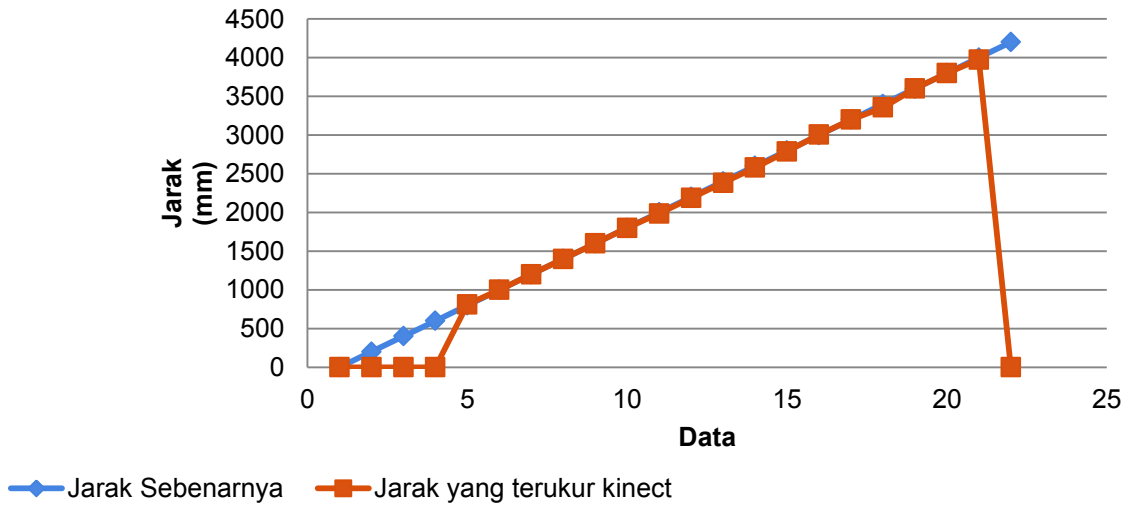


# Algoritma Utama



# PENGUJIAN PENGUKURAN JARAK KINECT

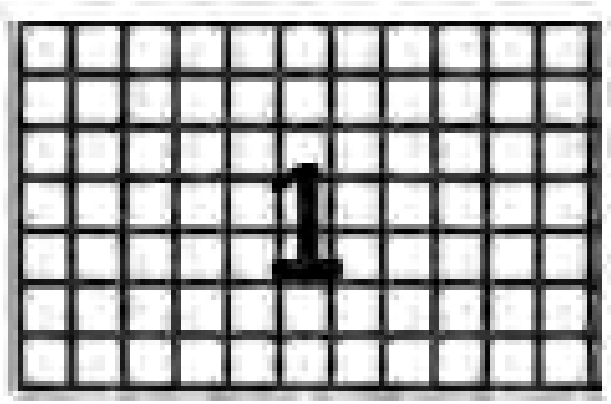
**Grafik Pengukuran Jarak Sebenarnya dengan Pengukuran Jarak menggunakan Kinect**



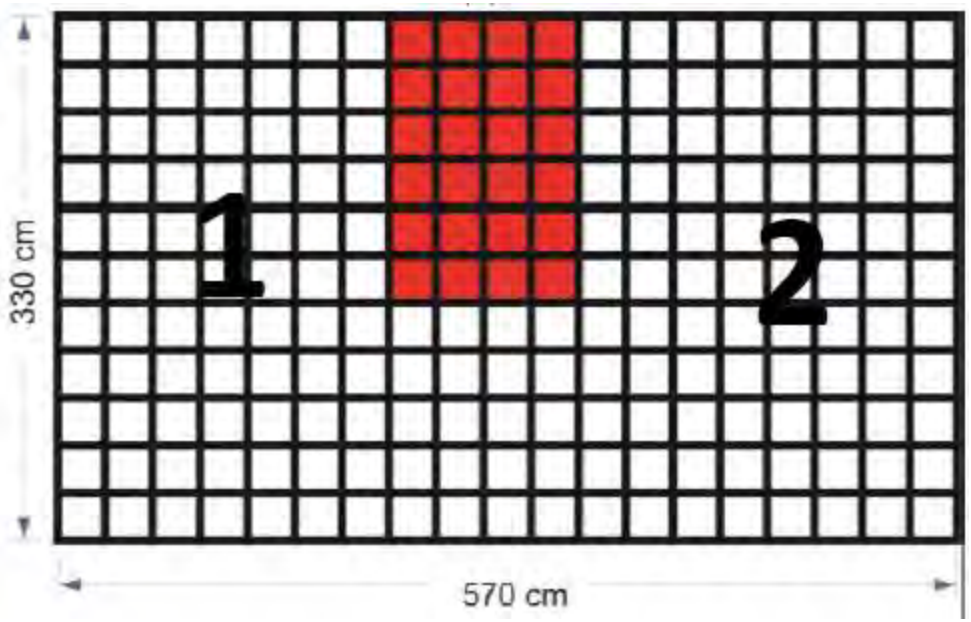

akurasi pengukuran jarak oleh kinect pada jarak **kurang dari 80 cm dan lebih dari 4 meter**, pengukuran jarak memiliki error rata-rata sebesar **100 %**.Sedangkan pada jarak antara **lebih dari 80 cm hingga kurang dari 4 meter** pengukuran jarak memiliki error rata-rata sebesar **0.197 %**.

# PENGUJIAN JELAJAH ROBOT

DESAIN RUANGAN

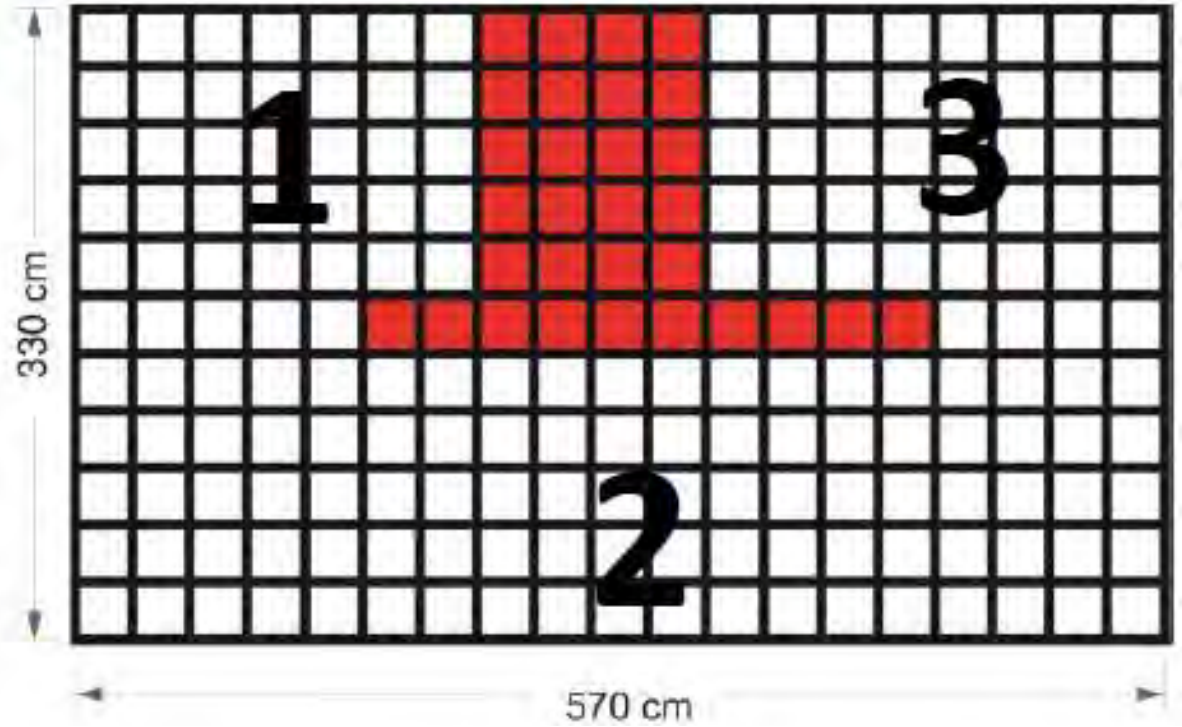



Ruang uji 1



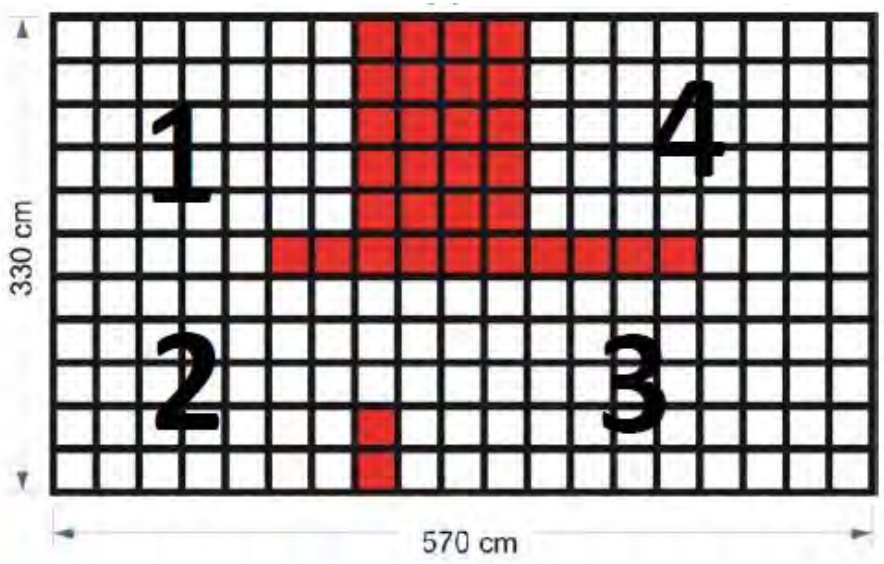

Ruang uji 2

# PENGUJIAN

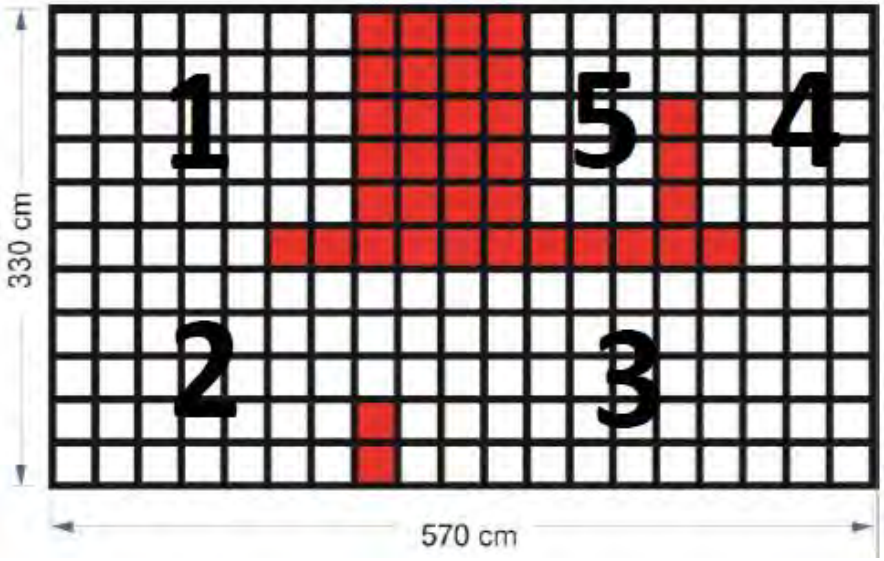



Ruang uji 3

# PENGUJIAN

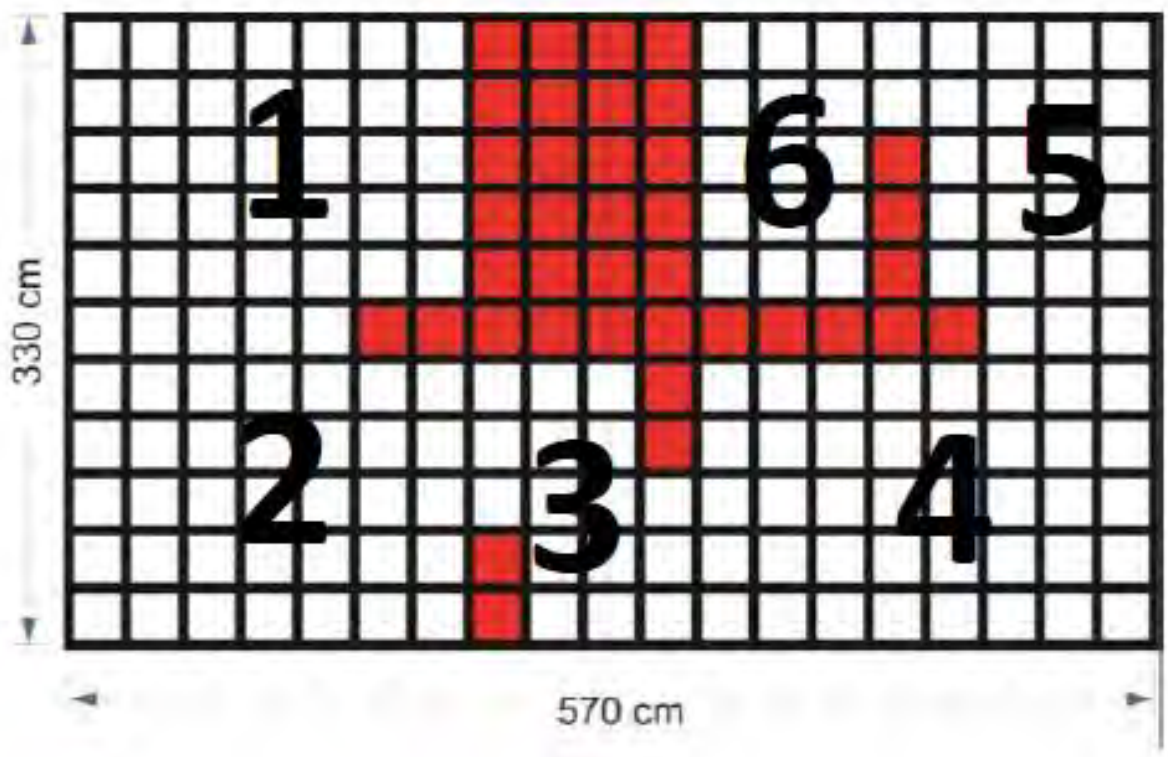



Ruang uji 4




Ruang uji 5

# PENGUJIAN

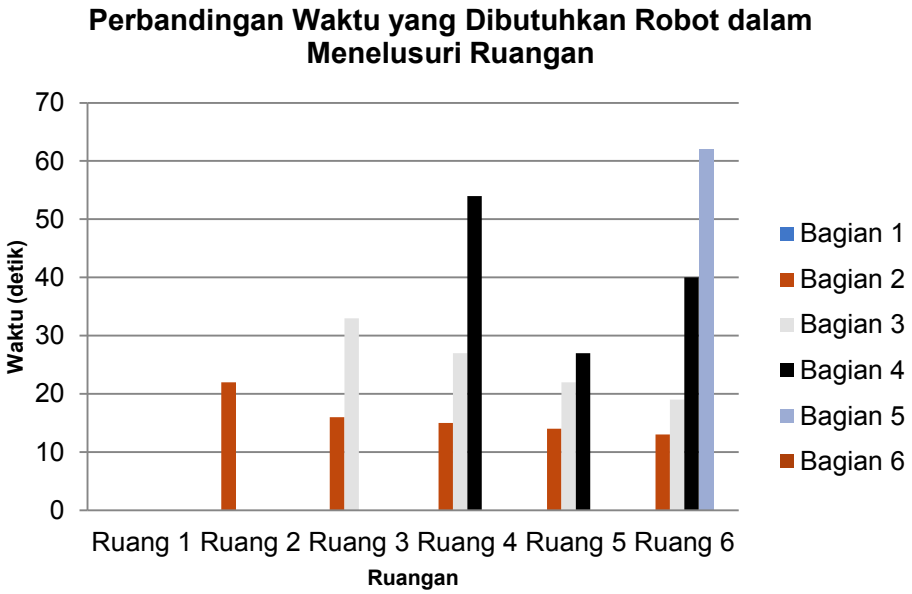



Ruang uji 6

# Hasil Waktu Telusur per Bagian Ruang

| Ruang | Bagian | Waktu Per Bagian (detik) | Total Ruang yang Dimasuki (%) |
|---|---|---|---|
| 1 | 1 | 0 | 100 |
| 2 | 1 | 0 | 100 |
| | 2 | 22 | |
| 3 | 1 | 0 | 100 |
| | 2 | 16 | |
| | 3 | 33 | |
| 4 | 1 | 0 | 100 |
| | 2 | 15 | |
| | 3 | 27 | |
| | 4 | 54 | |
| 5 | 1 | 0 | 80 |
| | 2 | 14 | |
| | 3 | 22 | |
| | 4 | 47 | |
| | 5 | - | |
| 6 | 1 | 0 | 83.33 |
| | 2 | 13 | |
| | 3 | 19 | |
| | 4 | 40 | |
| | 5 | 62 | |
| | 6 | - | |

Kemampuan rata – rata Sistem dalam memasuki Ruang sebesar 92,3%

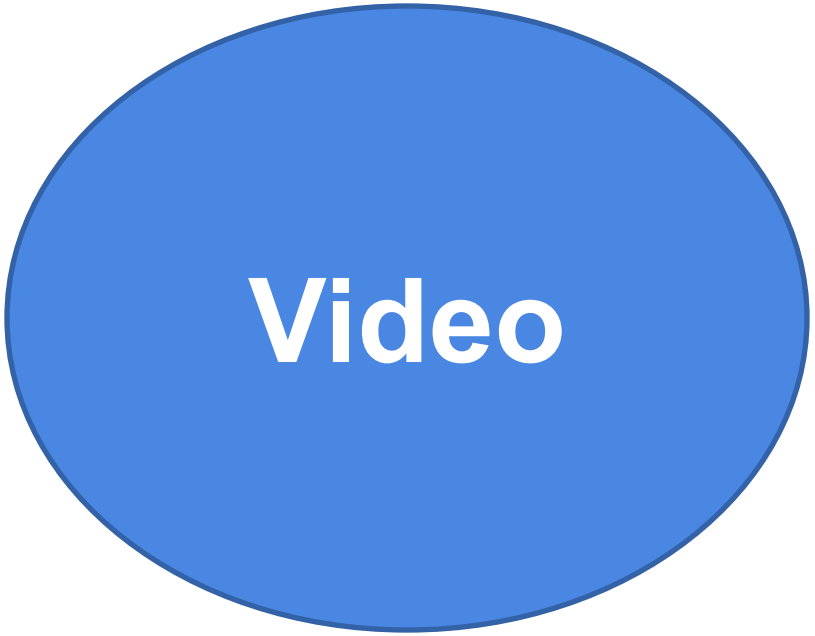


# Kesimpulan

1. Dalam menelusuri ruangan, robot menjaga jarak dengan dinding sebelah kiri dengan rata – rata jarak dari dinding dalam set point jarak dari dinding sebesar 108 cm adalah 80,36 cm dengan error rata – rata sebesar 35,88%. Sedangkan dengan set point 85 cm, robot telah mengikuti dinding dengan jarak rata-rata dari dinding sebesar 65,73 cm dengan error rata-rata sebesar 26,3 %. Dari data tersebut diketahui bahwa error rata – rata yang dimiliki robot dalam menjaga jarak dengan dinding kiri sebesar 31,09 % berdasarkan set point jarak yang digunakan.

# Kesimpulan

2. Berdasarkan pengujian menggunakan ruang uji, keberhasilan rata-rata robot untuk menelusuri bagian dari ruangan sebesar 92,67%.
3. Error 7,33% disebabkan oleh ketidakmampuan robot untuk masuk kedalam beberapa bagian ruangan dikarenakan ukuran celah ruangan yang kecil. Pada kondisi ruangan seperti itu, sistem akan mengaggap celah tersebut sebagai halangan sehingga robot akan mengantisipasi dengan cara menjauhkan diri dari celah tersebut.
4. Penambahan pembagian ruangan pada ukuran yang sama akan semakin menurunkan kemampuan robot dalam menelusuri ruangan. Hal ini disebabkan oleh jarak pembacaan jarak oleh kamera kinect yang terbatas yaitu 80 cm hingga 4 meter.


Video

# Terima Kasih