TESIS – TE142599

SISTEM PEMETAAN MENGGUNAKAN FITUR DEPTH SENSOR KINECT PADA MOBILE ROBOT UNTUK PROSES EVAKUASI KEBAKARAN GEDUNG

ALI UROIDHI
2214204003

DOSEN PEMBIMBING
Ronny Mardiyanto, ST., MT., Ph.D
Ir. Djoko Purwanto, M.Eng., Ph.D

PROGRAM MAGISTER
BIDANG KEAHLIAN ELEKTRONIKA
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

TESIS – TE142599

SISTEM PEMETAAN MENGGUNAKAN FITUR DEPTH SENSOR KINECT PADA MOBILE ROBOT UNTUK PROSES EVAKUASI KEBAKARAN GEDUNG

ALI UROIDHI
2214204003

DOSEN PEMBIMBING
Ronny Mardiyanto, ST., MT., Ph.D
Ir. Djoko Purwanto, M.Eng., Ph.D

PROGRAM MAGISTER
BIDANG KEAHLIAN ELEKTRONIKA
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T)
di
Institut Teknologi Sepuluh Nopember

oleh:

Ali Uroidhi
NRP. 2214204003

Tanggal Ujian : 04 Januari 2017
Periode Wisuda : Maret 2017

Disetujui oleh:

1. **Ronny Mardiyanto, ST., MT., Ph.D** **(Pembimbing I)**
   NIP: 198101118 200312 1 003
2. **Ir. Djoko Purwanto, M.Eng., Ph.D** **(Pembimbing II)**
   NIP: 19651211 199002 1 002
3. **Dr. Tri Arief Sardjono, ST., MT** **(Penguji)**
   NIP: 19700212 199512 1 001
4. **Achmad Arifin, ST., M.Eng., Ph.D** **(Penguji)**
   NIP: 19710314 199702 1 001
5. **Dr. Muhammad Rivai, ST.,MT** **(Penguji)**
   NIP: 19690426 199403 1 003

an. Direktur Program Pascasarjana

Prof. Dr. Ir. Tri Widjaja, M.Eng.
NIP. 19611021 198603 1 001

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**SISTEM PEMETAAN MENGGUNAKAN FITUR DEPTH SENSOR KINECT PADA MOBILE ROBOT UNTUK PROSES EVAKUASI KEBAKARAN GEDUNG**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Januari 2017

Ali Uroidhi

NRP. 2214204003

SISTEM PEMETAAN MENGGUNAKAN FITUR DEPTH SENSOR KINECT PADA MOBILE ROBOT UNTUK PROSES EVAKUASI KEBAKARAN GEDUNG

Mahasiswa Nama : Ali Uroidhi
Mahasiswa ID : 2214204003
Pembimbing : 1. Ronny Mardiyanto, ST., MT., Ph.D
2. Ir. Djoko Purwanto, M.Eng., Ph.D

ABSTRAK

Sistem pemetaan untuk proses evakuasi adalah sistem pemetaan yang membantu tim evakuasi untuk menemukan korban. Kendala tim evakuasi adalah waktu persiapan untuk melakukan evakuasi, waktu persiapan meliputi perjalanan menuju lokasi, mempersiapkan alat, dan melakukan evakuasi. Di Indonesia banyak kawasan-kawasan yang padat akan penduduk, dan memiliki area jalan yang sempit. kedua hal tersebut sangat menghambat perjalanan tim evakuasi. Sehingga waktu persiapan untuk evakuasi menjadi lama. Disela waktu persiapan evakuasi, dapat dimanfaatkan untuk melakukan pemetaan dan pencarian korban. Sehingga ketika persiapan evakuasi telah selesai, tim evakuasi dapat melihat hasil pemetaan beserta posisi korban. Dengan mengetahui posisi korban, maka proses evakuasi diharapkan akan lebih cepat.

Penelitian ini akan dilakukan pengontrolan navigasi untuk pemetaan menggunakan sensor Kinect dan mini komputer (Raspberry PI 2). Karena sensor Kinect memiliki sudut pandang yang sempit (sekitar 57^0), maka di perlukan adanya kemampuan untuk mengingat kondisi sekitar. Kemampuan mengingat kondisi sekitar di inspirasi oleh kemampuan manusia dalam menghindari objek tanpa harus memandang objek secara terus-menerus. Dua sumber kendali navigasi yaitu secara *real-time* (diambil langsung dari sensor Kinect) maupun melalui *database* peta, di harapkan dapat digunakan untuk membentuk kemampuan tersebut.

Penelitian ini mengarah pada evakuasi korban kebakaran di gedung, korban yang ada akan di informasikan pada sebuah dengan rute terbaik menuju korban. Hasil pengujian menunjukkan mobile robot mampu menggambarkan rute korban yang dideteksi dengan baik dengan rata-rata waktu proses 355.71 ms untuk kondisi baterai 100%, dan rata-rata waktu proses 824.34 ms untuk kondisi baterai 40%. Tingkat kesalahan dalam pembuatan peta mencapai sekitar 16.43%. kesalahan ini dipengaruhi oleh eror pembacaan Kinect yang rata-ratanya 4.21%, eror rotary encoder dengan rata-rata eror 0.41%.Dengan dua sumber referensi navigasi mobile robot dapat berjalan dan mencari korban secara autonomous.

Kata kunci : Fitur jalan, kontrol navigasi, Mobile robot, Pemetaan, Penyelamatan

MAPPING SYSTEM USING THE DEPTH SENSOR KINECT FEATURES IN MOBILE ROBOT FOR THE BUILDING FIRE EVACUATION

By : Ali Uroidhi
Student Identity Number : 2214204003
Supervisors : 1. Ronny Mardiyanto, ST., MT., Ph.D
2. Ir. Djoko Purwanto, M.Eng., Ph.D

ABSTRACT

Mapping system for the evacuation process is a mapping system that helps the evacuation teams to find survivors. Tim evacuation constraint is the time to prepare to evacuate, the preparation time included a trip to the location, preparing tools, and evacuation. In Indonesia, many populated areas, and have a narrow street area. This second trip greatly hamper the evacuation team. So the preparation time for evacuation becomes longer. Disconnected evacuation preparation time, can be used to map and search the victim. So when preparing the evacuation has been completed, the evacuation teams can see the results of the mapping along with the position of the victim. By knowing the position of the victim, then the evacuation process is expected to be faster.

This study will be conducted controlling for mapping navigation using Kinect sensor and mini-computer (Raspberry PI 2). Because the Kinect sensor has a narrow viewing angle (approximately 570), need the ability to remember the ambient conditions. The ability to remember about conditions inspired by the human ability to avoid objects regardless of object continuously. Two sources navigation control in real-time (taken directly from the Kinect sensor) or via a map database, expected to be used to build these capabilities.

This research led to the evacuation of victims of a fire in the building, there are victims who will be informed on the best route for the victim. The test results show the phone robot is able to describe these victims detected well with an average of 355.71 ms processing time for the condition of the battery 100%, and the average processing time of 824.34 ms to 40% battery condition. The error rate in mapmaking approximately 16: 43%. This error is affected by an error reading Kinect average 4.21%, the error rotary encoder with an average error of 0.41% .In two menu reference in mobile robot can walk victimizing autonomous.

Key words: Features street, the navigation controls, Mobile robot, Mapping, Rescue

KATA PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kepada Allah SWT, karena atas segala nikmat, dan taufik Nya lah tesis ini dapat diselesaikan. Tesis berjudul "**SISTEM PEMETAAN MENGGUNAKAN FITUR DEPTH SENSOR KINECT PADA MOBILE ROBOT UNTUK PROSES EVAKUASI KEBAKARAN GEDUNG**" ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Magister Teknik (MT) pada Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh November.

Penulis menyadari bahwa dalam penyusunan tesis ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

1. Bapak Ronny Mardiyanto S.T., M.T., Ph.D. selaku Dosen Pembimbing, yang telah banyak memberikan saran, bantuan, serta sabar dalam membimbing penulis.
2. Bapak Ir. Djoko Purwanto, M.Eng., Ph.D. selaku Dosen Pembimbing dan Koordinator Program Pasca Sarjana Jurusan Teknik elektro – FTI – ITS, yang telah banyak memberikan saran, bantuan, serta sabar dalam membimbing penulis.
3. Bapak Dr. Muhammad Rivai, S.T., M.T. selaku Koordinator Bidang Keahlian Teknik Elektronika – Jurusan Teknik Elektro dan selaku Dosen Penguji Ujian Sidang Tesis atas saran dan masukannya.
4. Bapak Dr. Tri Arief Sardjono, S.T., M.T. selaku Wakil Dekan FTI – ITS, dan selaku Dosen Penguji Ujian Sidang Tesis atas saran dan masukannya.
5. Bapak Achmad Arifin, S.T., M.Eng., Ph.D. selaku Ketua Prodi Teknik Biomedik dan selaku Dosen Penguji Ujian Sidang Tesis atas saran dan masukannya.
6. Pimpinan dan civitas akademika Jurusan Teknik Elektro FTI – ITS.
7. Ibu, Ayah, dan saudara, atas segala dukungan dan doanya hingga terselesaikannya tesis ini.

Pada akhirnya, penulis menyadari bahwa tesis ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga tesis ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi serta bagi masyarakat.

Surabaya, 13 Januari 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan berkembangnya teknologi, khususnya teknologi elektronik, peran robot menjadi semakin penting tidak saja di bidang sains, tapi juga di berbagai bidang lainnya, seperti di bidang kedokteran, pertanian, dan militer. Peran robot diharapkan dapat membantu pekerjaan manusia dengan aman dan akurat. Dalam beberapa kasus terdapat pekerjaan manusia yang memiliki risiko keselamatan yang sangat besar, salah satunya adalah pekerjaan penyelamatan. Robot penyelamat dirancang untuk tugas yang berbahaya, misi penyelamatan biasanya berupa penyelamatan manusia maupun aset-aset berharga pada saat terjadi bencana.

Bencana atau musibah bukanlah sesuatu yang dapat dengan mudah diprediksi dan dihindari, berbagai upaya telah dilakukan untuk memperkecil munculnya korban. Ketika bencana telah terjadi aksi penyelamatan atau evakuasi yang aman dan cepat merupakan hal yang terpenting untuk mengurangi jumlah kerugian.

Dalam proses evakuasi terdapat beberapa kendala, salah satu kendala adalah waktu persiapan untuk melakukan evakuasi. Waktu persiapan evakuasi meliputi perjalanan menuju lokasi, mempersiapkan alat, dan melakukan evakuasi. Permasalahan yang sering di jumpai di Indonesia adalah permasalahan perjalanan tim evakuasi. Permasalahan ini terjadi karena banyaknya kawasan-kawasan padat penduduk, dan area jala yang sempit. Hal-hal tersebut menyebabkan waktu persiapan evakuasi menjadi lama.

Dalam proses evakuasi, ketika terdapat korban yang terjebak. Tim evakuasi akan masuk dan menjelajah ruangan untuk menemukan korban-korban tersebut. Ketika terjadi bencana seperti kebakaran, pencarian korban haruslah sangat cepat. Lamanya pencarian korban akan berisiko pada tim evakuasi tersebut. Keselamatan tim evakuasi juga menjadi pertimbangan, sehingga tidak memicu muncul korban baru.

Peranan robot dalam membantu evakuasi di harapkan dapat mempercepat dan memperkecil munculnya korban baru. Berbagai hal dalam evakuasi dapat diwakilkan oleh sebuah robot, di antaranya adalah menemukan korban dan menggambarkan dalam sebuah peta. Banyak penelitian yang meneliti pembuatan peta. Pada jurnal [1] [2], dibuat sistem pemetaan menggunakan sensor lidar dengan kemampuan sudut pandang 240^0-270^0, dan menggunakan sensor sonar. Penggunaan sensor lidar dan sensor sonar memiliki kemampuan yang baik dalam melakukan pemetaan tetapi memiliki harga yang cukup mahal. Pada jurnal [3] [4] menggunakan sensor Kinect, tetapi sensor Kinect hanya digunakan untuk membaca objek, sedangkan sistem gerak masih di bantu menggunakan sensor sonar.

Pada penelitian ini digunakan sensor Kinect karena sensor Kinect merupakan sensor gerakan yang dibuat Microsoft yang memiliki harga rendah [5], kemampuan sensor Kinect tidak hanya digunakan sebagai game komputer, tetapi digunakan dalam beberapa penelitian [6], [7]. Sensor Kinect memiliki sudut pandang yang sempit sekitar 57^0. Karena sudut pandang yang sempit, maka di perlukan teknik khusus untuk melakukan sebuah navigasi.

Pada penelitian ini akan digunakan sistem pemetaan dengan menggunakan dua sumber navigasi, yaitu navigasi secara Online (langsung dari Kinect) dan Offline (data peta). Dalam teknik pemrosesan di perlukan komputer untuk memproses data peta. Penggunaan komputer membutuhkan biaya yang cukup besar, sehingga di pilih mini komputer Raspberry PI 2 yang memiliki harga rendah.

1.2 Rumusan Masalah

Penelitian ini membahas tentang permasalahan evakuasi dalam kebakaran. Dalam proses evakuasi kecepatan dan keamanan proses evakuasi menjadi hal yang paling di perhitungkan. Salah satu alternatif mempercepat sekaligus aman adalah dengan meluncurkan robot untuk membantu evakuasi. Peran robot dalam evakuasi bermacam-macam, di antaranya untuk menemukan posisi korban beserta peta dengan rute yang terbaik. Permasalahan yang muncul dalam sistem pembuatan peta yang menunjukkan korban adalah:

1. Pembuatan peta menggunakan Kinect
2. Membedakan antara penghalang dengan jalan

3. Merekam fitur jalan ke dalam peta
4. Menemukan korban
5. Menggambarkan posisi korban ke dalam peta

1.3 Tujuan dan Manfaat

Adapun tujuan dari penelitian ini adalah membuat sebuah sistem kontrol robot mobile menggunakan Kinect, dengan sistem pemetaan jalan sekaligus merekam hasil pemetaan, kontrol robot dilakukan secara otomatis.

Sistem ini di aplikasikan pada suatu pencarian korban seperti pada bencana kebakaran. Ketika korban ditemukan robot akan menandai posisi korban di dalam peta, peta digunakan oleh tim evakuasi untuk melakukan evakuasi sesuai posisi korban di dalam peta. Dengan sistem ini diharapkan robot dapat bergerak secara otomatis dan dapat mengenali rute dan fitur jalan, tanpa harus di perkenalkan secara manual.

1.4 Batasan Masalah

Dalam penelitian ini dibuat sistem pembantu evakuasi dalam bencana kebakaran, pada tahap ini hanya di rancang sistem pemetaan dan pencarian korban. Sehingga pada penelitian ini diberikan beberapa batasan yaitu:

1. Robot tidak di rancang untuk menjelajah pada suhu tinggi
2. Pemetaan hanya di lakukan di dalam ruangan.
3. Proses evakuasi hanya berupa penunjuk posisi tanpa membawa korban.
4. Pembacaan fitur jalan memiliki keterbatasan seperti tidak mampu membaca penghalang yang terlalu tipis.
5. Jalan yang dilewati memiliki lebar minimal karena keterbatasan sensor Kinect.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

Evakuasi merupakan suatu pemindahan objek (orang atau benda berharga) dari tempat kurang aman ke tempat yang lebih aman, misal dari bahaya perang, bahaya banjir, meletusnya gunung api, kebakaran, dan kebocoran gas ke daerah yang aman. Perencanaan evakuasi dilakukan untuk memastikan waktu evakuasi teraman dan paling efisien bagi semua orang. Ketepatan perencanaan akan menghasilkan beberapa jalan keluar evakuasi, proses evakuasi memiliki beberapa tahapan mulai tahap pendeteksian, pengambilan keputusan, penandaan alarm, reaksi, perpindahan ke area perlindungan, dan transportasi.

Mobile robot adalah mesin otomatis yang mampu bergerak. Mobile robot memiliki kemampuan untuk bergerak di sekitar lingkungan dan tidak tetap untuk satu lokasi. Mobile robot dapat "otonom" yang berarti mampu menavigasi lingkungan yang tidak terkendali tanpa perlu perangkat bimbingan fisik atau elektro-mekanis. Mobile robot dapat mengandalkan perangkat sensor yang memungkinkan mereka untuk melakukan perjalanan rute navigasi yang telah didefinisikan dalam ruang yang relatif terkendali.

Berbagai penelitian banyak dilakukan pada mobile robot di antaranya jurnal [8] telah dirancang sebuah pendekatan untuk mengenali dan mengklasifikasi benda-benda menggunakan Microsoft kinect yang digunakan untuk mobile robot. Dalam mengenali benda-benda dilakukan dengan dua fase yaitu fase *on-line* dan *off-line*. Fase *off-line* terdiri dari segmentasi, grafik konstruksi dan ekstraksi model. Segmentasi memisahkan objek yang dihasilkan oleh sensor kinect dan masing-masing terurai dengan kelompok. Konstruksi grafik merupakan himpunan kelompok dari pemisahan blok dalam grafik, di mana himpunan membentuk hubungan geometris. Hasil dari ekstraksi disimpan dalam database model objek. Fase *on-line* hampir sama dengan fase *off-line*, perbedaannya adalah database model objek digunakan dalam masukkan pengenalan (*recognition*), pengenalan di sini membandingkan objek dari kinect secara *real-time* dengan semua model yang ada dalam database. Hasil dari algoritma ini mampu mendeteksi benda-benda yang

sama dengan yang tersimpan di dalam database tetapi masih belum bisa mengenali model yang berbeda seperti meja bulat dengan meja kotak.

Jurnal [9] merancang robot pelacak yang mampu mendaki trotoar dengan menggunakan kontrol Fuzzy. Untuk mendeteksi tanjakan jurnal ini menggunakan beberapa mode. Mode pertama adalah *searching ramp mode*, *aligning ramp mode*, *closing mode*, dan *climbing mode*. *Searching ramp mode* mendeteksi perbedaan antara *depth* bawah dengan *depth* atas, *aligning ramp mode* mendeteksi *rims* kanan dengan kiri, *closing mode* digunakan untuk menjaga jarak antara robot dengan jalan, *climbing mode* digunakan untuk memastikan kaki depan, dan kaki belakang menempel dengan tanah. Hasil dari metode ini mampu memperkecil eror sesuai dengan batas yang diizinkan dan mampu berjalan secara akurat dan stabil dan dapat mencapai tujuan.

Jurnal [10] dibuat suatu robot yang memetakan suatu tempat yang digunakan sebagai pemandu orang asing. Dalam jurnal ini digunakan kinect sebagai penglihatan, dan sebagai sarana untuk mengkategorikan label-label tempat yang terdeteksi selama robot berjalan, informasi kinect diolah menggunakan cara *hierarchical fusion* untuk merinci adegan yang diamati. Dalam pemetaan dibuat teknik pengelompokan berdasarkan *indoors single scane*, *indoors large scale, outdoors single scene,* dan *outdoors large scale.*

Jurnal [11] dibuat sistem perencana gerak jalur robot dengan menggunakan *Voronoi Fast Marching* (VFM), metode ini memperbaharui dari algoritme diekstrak, metode ini di klaim dapat menentukan dan mempertahankan waktu respons yang baik, pergerakan yang halus dan lintasan yang aman. Ketika terdapat hambatan tidak cembung metode ini dapat menghasilkan lintasan yang baik.

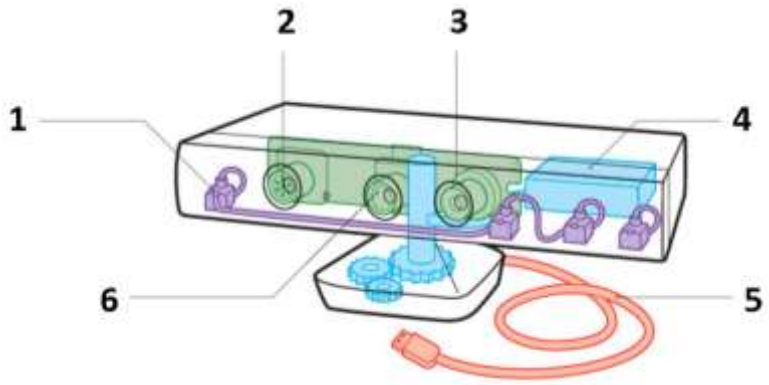
Jurnal [12] di buat sistem yang memodifikasi algoritme A* untuk menghasilkan gerakan yang aman, pada algoritme mencari rute, kebanyakan hasil yang dihasilkan selalu bersanding dengan dinding, sehingga ketika di terapkan robot sering kali terjadi tabrakan. Untuk menghasilkan rute yang aman maka algoritme A* dimodifikasi dengan menambahkan *Direction of Arrival* (DOA) dan masukan ukuran robot, sehingga rute dihasilkan berdasarkan penambahan ukuran robot.

2.1 Sensor Kinect

Kinect adalah "*controller-free gaming*" oleh Microsoft dan Xbox 360 video game platform. Sensor Kinect adalah batang horizontal yang terhubung dengan alas kecil yang memiliki poros yang dapat berputar. Perangkat ini memiliki kamera RGB, sensor kedalaman, dan mikrofon, yang menyediakan kemampuan untuk menangkap gerak secara 3D, mengenali wajah, dan mengenali suara.

Sensor kedalaman terdiri dari proyektor laser infrared dikombinasikan dengan sensor CMOS *monokromatik*, yang merekam data video 3D dalam kondisi pencahayaan apa pun. Pengiriman video RGB menggunakan resolusi VGA (640 x 480 piksel) dengan kedalaman warna 8-bit dengan filter warna Bayer, sedangkan pengiriman video monokrom untuk deteksi kedalaman menggunakan resolusi VGA dengan kedalaman warna 11-bit. Sensor Kinect memiliki area penggunaan 1.2-35 m (3.9-11 kaki), memiliki lebar pandangan angular 57 derajat horizontal dan 43 derajat vertikal, dan poros yang di gerakan oleh motor sampai dengan 27 derajat atas dan bawah. Bidang horizontal dari sensor Kinect pada jarak minimum ~0.8m (2.6 kaki) adalah sekitar ~87 cm (34 inch), dan bidang vertikal adalah ~63 cm (25 inch), menghasilkan resolusi sekitar 1.3 mm (0.051 inch) setiap pikselnya. Fitur mikrofon pada Kinect memiliki empat mikrofon kapsul dan setiap jalur dioperasikan dengan kedalaman suara 16-bit pada kecepatan cuplik 16 kHz.

Gambar 2.1 Gambar Data Kedalaman Sensor Kinect [13]



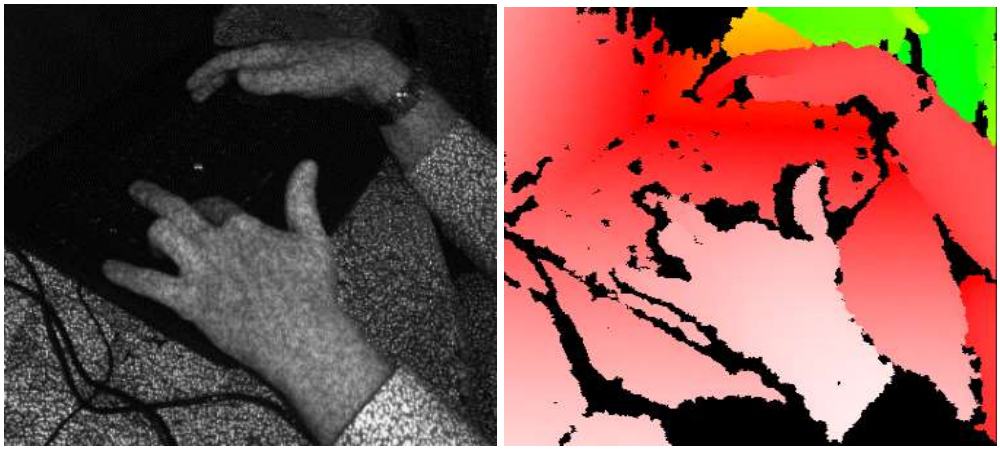
Gambar 2.2 Komponen-Komponen Sensor Kinect [18]

Data kedalaman Kinect dengan warna cerah menunjukkan bahwa objek dekat ke kamera. Namun, karena terlalu dekat akan menyebabkan objek tidak terdeteksi dan ditandai sebagai warna hitam dan objek terjauh akan berwarna gelap, hal ini ditunjukkan pada Gambar 2.1. Output berupa histogram dari akumulatif frekuensi kemunculan setiap nilai. Histogram dibangun berdasarkan nilai-nilai kedalaman setiap pixel [13]

Gambar 2.2 menunjukkan isi dari komponen yang ada pada sensor kinect, kompoen tersebut adalah

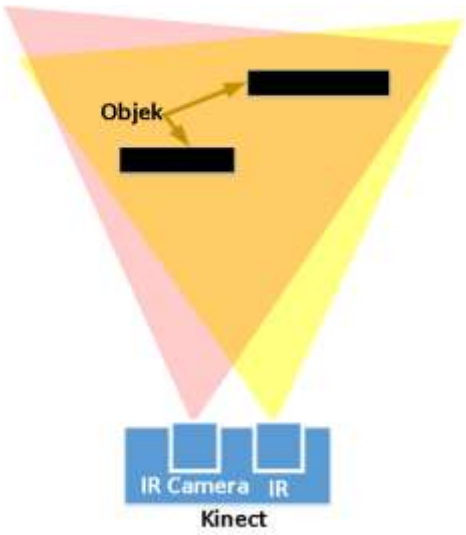
1. Microphone Array
2. IR emitter
3. Depth camera / IR Camera
4. Tilt motor
5. USB cable
6. Color camera

Data depth kinect di dapatkan dari paduan IR Emitter dan IR Camera, prinsip kerja dari kedua perangkat tersebut di ilustrasikan pada Gambar 2.4, IR Emitter dengan IR Camera mimiliki jarak yang konstan. Perbedaan jarak pada kedua perangkat mengakibatkan perbedaan penangkapan objek, sehingga terdapat beberapa titik yang terkena IR Emitter tetapi tidak terbaca oleh kamera, begitu juga sebaliknya. Cahaya yang di pancarkan IR Emitter memiliki pola-pola titik. Ketika jarak benda semakin jauh maka pola-pola titik akan semakin berjauhan dan semakin kecil. Ketika jarak benda semain dekat maka pola-pola titik akan semakin berdekatan dan semakin besar. Berdasarkan pola-pola tersebut IR Camera akan membaca intensitas dari cahaya IR yang di pancarkan. Pancaran sinar IR dan data Depth di tunjukkan pada Gambar 2.3



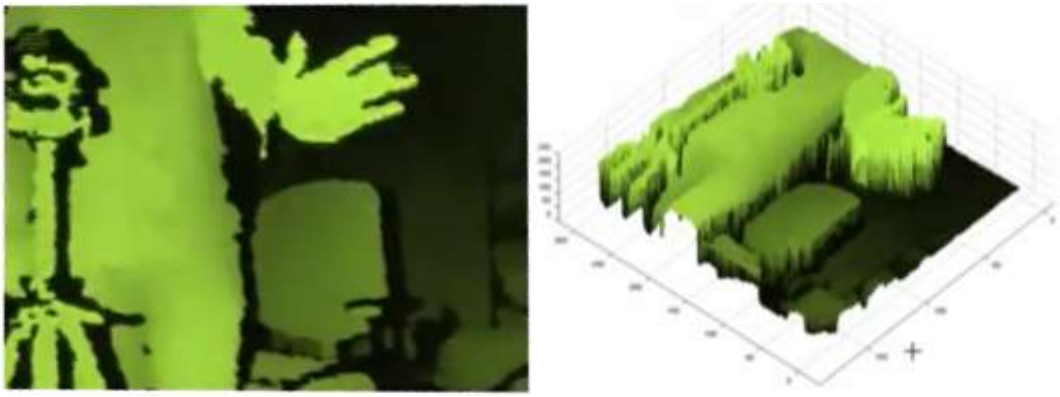
(a) (b)

Gambar 2.3 Data Depth Kinect, (a) IR Kinect, (b) Depth Map Kinect [17]



Gambar 2.4 Prinsip Kerja Kinect

Berdasarkan prinsip di atas data depth kinect mempresentaikan jarak yang di dapat dari itensitas cahaya yang di tangkap oleh IR Camera. Semakin jauh suatu benda itensitas cahaya akan semakin kecil, kecilnya intensitas juga di pengaruhi dari jauhnya pola-pola titik IR pada benda tersebut. Hasil jarak di presentasikan dalam 2D array, nilai dari 2d array dapat mewakili semua jarak, hasil 2d array beserta plot diagram 3d di tunjukkan pada Gambar 2.5. berdasarkan Gambar 2.5, dapat terlihat perbadaan jarak antara orang dengan kursi, dengan dinding, dan dengan tripot kamera malalui plot 3d



(a) (b)

Gambar 2.5 Data Depth Kinect (a) Bentuk 2D Array, (b) Dalam Plot 3D [19]

2.2 Pemetaan Jalan

Data pemetaan jalan didapatkan dari sensor kinect, data kedalaman dari sensor kinect digunakan sebagi arsitektur jalan, arsitektur jalan didapatkan dari pembacaan dari data kedalaman secara bergantian, sehingga bentuk dari rintangan di depan robot akan mampu terbaca dengan baik, adapun proses pembacaan dapat terlihat pada Gambar 2.6.

Gambar 2.6 menunjukkan pergerakan robot dari titik pusat (0,0) menuju ke koordinat (*x*,*y*) pada pergerakan frame dasar (lingkup koordinat (*xb*,*yb*)), perubahan frame yang bergerak disimbolkan dengan koordinat (*xm*,*ym*).

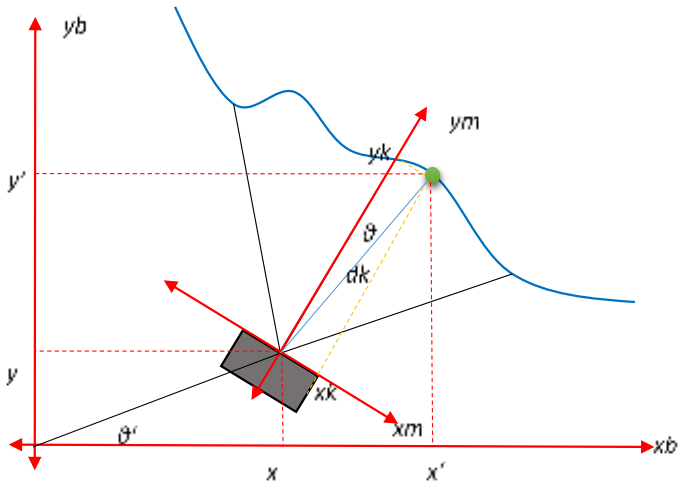
Berdasarkan gambar di atas dengan menggunakan konsep trigonometri didapatkan Persamaan 2.1-Persamaan 2.3 untuk mendapatkan posisi x dan y pada dinding,

$$dk = \sqrt{xk^2 + yk^2} \quad (2.1)$$

$$x' = x + (dk * \sin(\theta' + \theta)) \quad (2.2)$$

$$y' = y + (dk * \cos(\theta' + \theta)) \quad (2.3)$$

dengan melakukan pembacaan data kedalaman secara bergantian maka bentuk dari penghalang akan mampu dibaca, selanjutnya adalah mengonversikan data hasil pembacaan agar mampu disimpan dan dibaca kembali.



Gambar 2.6 Struktur Pembacaan Data Kedalaman Kinect.

2.3 Kontrol Mobile Robot

Berdasarkan pemetaan jalan pada sub bab sebelumnya, telah didapatkan dua frame yaitu frame pergerakan (*xm*,*ym*) dan frame dasar (*xb*,*yb*) yang diilustrasikan pada Gambar 2.6, Postur (sikap robot) pada frame dasar dapat digambarkan dalam Persamaan 2.4, sedangkan rotasi metrik dari frame dasar dengan respons terhadap frame bergerak dapat dituliskan pada Persamaan 2.5.

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad 2.4$$

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 2.5$$

Pengontrolan diferensial menggunakan kontrol masukkan $u = \begin{bmatrix} v \\ \omega \end{bmatrix}$ dimana v adalah kecepatan linier robot dan ω adalah kecepatan sudut robot, pada Gambar 2.8 merupakan ilustrasi pergerakan robot menggunakan kontrol diferensial. Pada Gambar 2.7 terdapat beberapa parameter di antarnya $V_R(t)$ adalah kecepatan linier roda kanan, $V_L(t)$ adalah kecepatan linier roda kiri, r adalah jari-jari roda, dan R adalah jarak ICC ke titik tengah antara dua roda.

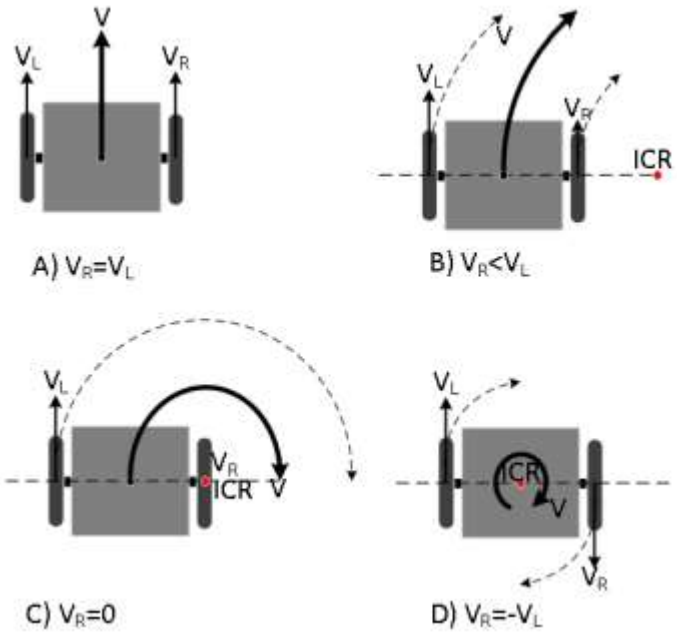
Persamaan kinematika dari kontrol masukan ditunjukkan pada Persamaan 2.6- Persamaan 2.8,

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad 2.6$$

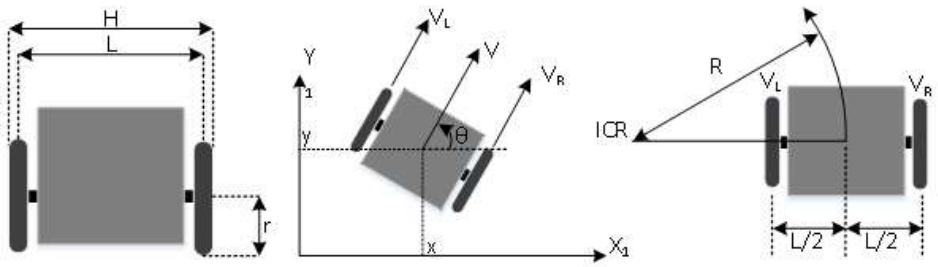
$$\frac{V_R - V_L}{L} = \frac{V_R}{R + \frac{L}{2}} \quad 2.7$$

$$R = \frac{L}{2} * \frac{V_R + V_L}{V_R - V_L} \quad 2.8$$

ketika jari-jari rotasi (R) adalah 0 maka pergerakan $V_R = -V_L$ dan R adalah ∞ maka pergerakan $V_R = V_L$.



Gambar 2.7 Dasar Kontrol Pergerakan.



Gambar 2.8 Kontrol Diferensial.

2.4 Logika Fuzzy

Logika Fuzzy adalah peningkatan dari logika *boolean* yang berhadapan dengan konsep kebenaran sebagian. Saat logika klasik menyatakan bahwa segala hal dapat diekspresikan dalam istilah biner (0 atau 1, hitam atau putih, ya atau tidak), logika Fuzzy menggantikan kebenaran *boolean* dengan tingkat kebenaran.

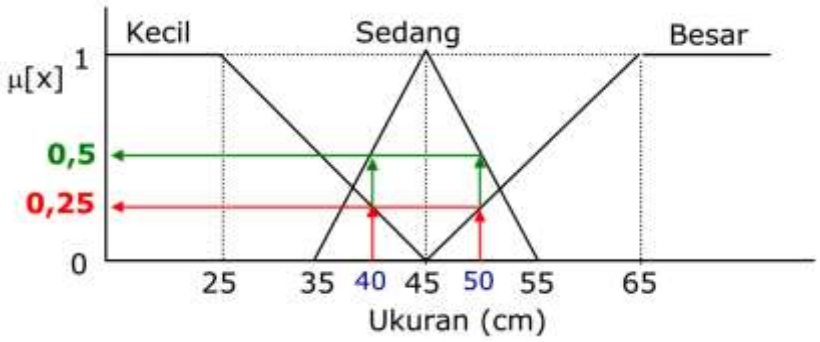
Logika Fuzzy memungkinkan nilai keanggotaan antara 0 dan 1, tingkat ke abuan dan juga hitam dan putih, dan dalam bentuk linguistik, konsep tidak pasti seperti "sedikit", "lumayan", dan "sangat". Logika ini berhubungan dengan set Fuzzy dan teori kemungkinan.

Logika Fuzzy dan logika probabilitas secara matematis sama, keduanya mempunyai nilai kebenaran yang berkisar antara 0 dan 1, namun secara konsep berbeda. Logika Fuzzy berbicara mengenai "derajat kebenaran", sedangkan logika probabilitas mengenai "probabilitas, kecenderungan". Karena kedua hal itu berbeda, logika Fuzzy dan logika probabilitas mempunyai contoh penerapan dalam dunia nyata yang berbeda.

2.4.1 Himpunan Fuzzy

Pada himpunan tegas (*crisp*), nilai keanggotaan suatu item *x* dalam suatu himpunan A, yang sering ditulis dengan μA[x], memiliki 2 kemungkinan, yaitu 0 dan 1, sehingga sekitar terjadi pergeseran sedikit saja akan mengakibatkan perbedaan kategori yang cukup signifikan.

Sedangkan pada himpunan Fuzzy digunakan untuk mengantisipasi hal tersebut. Seberapa besar eksistensinya dalam himpunan tersebut dapat dilihat pada nilai keanggotaannya.



Gambar 2.9 Himpunan Fuzzy.

Pada Gambar 2.9 dapat dilihat untuk ukuran 40 cm, termasuk dalam himpunan kecil (0,25) dan sedang (0,5), begitu juga untuk ukuran 50 cm, termasuk dalam himpunan sedang (0,5) dan besar (0.25). Sehingga himpunan Fuzzy berbeda dengan himpunan *crisp*. Himpunan *crisp* hanya ada 2 kemungkinan, yaitu 0 atau 1, sedangkan dengan himpunan Fuzzy nilai keanggotaan terletak pada rentang 0-1.
Ada beberapa hal yang perlu diketahui dalam sistem Fuzzy, yaitu:

1) Variabel Fuzzy

   Variabel Fuzzy merupakan variabel yang hendak dibahas dalam suatu sistem Fuzzy.

2) Himpunan Fuzzy

   Himpunan Fuzzy merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel Fuzzy. Setiap himpun akan memiliki kemungkinan di antara 0 sampai 1

3) Semesta Pembicaraan

   Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel Fuzzy. Semesta pembicaraan merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun negatif. Adakalanya nilai semesta pembicaraan ini tidak dibatasi batas atasnya.

4) Domain

   Domain himpunan Fuzzy adalah keseluruhan nilai yang diizinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan Fuzzy.

Seperti halnya semesta pembicaraan, domain merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif maupun negatif.

2.4.2 Fungsi Keanggotaan

Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaannya (sering juga disebut dengan derajat keanggotaan) yang memiliki interval antara 0-1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Ada beberapa fungsi yang bisa digunakan di antaranya

2.4.2.1 Representasi Linear

Pada representasi linear, pemetaan input ke derajat keanggotaan digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas.

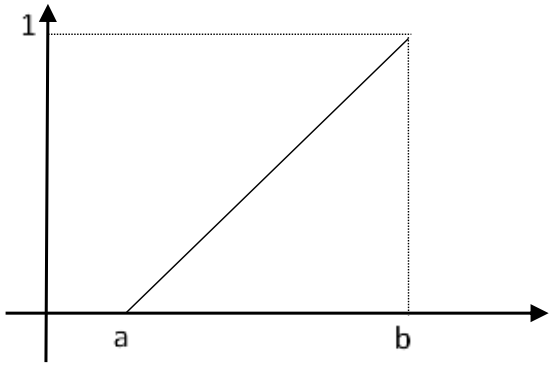
Ada 2 keadaan himpunan Fuzzy yang linear. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi.

Fungsi keanggotaan *linear up* ditunjukkan pada Persamaan 2.9 dengan bentuk grafik pada Gambar 2.10.

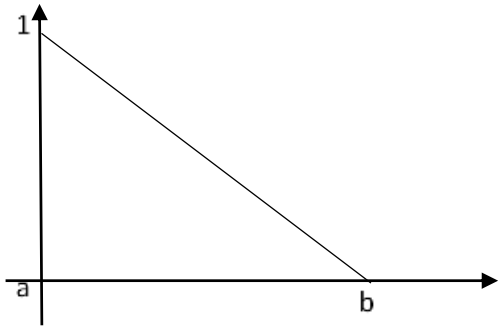
$$\mu[x] = \begin{cases} 0 & x \leq a \\ \dfrac{x-a}{b-a} & a \leq x \leq b \\ 1 & x \geq b \end{cases} \qquad 2.9$$

Fungsi keanggotaan *linear down* ditunjukkan pada Persamaan 2.10 dengan bentuk grafik pada Gambar 2.11.

$$\mu[x] = \begin{cases} 1 & x \leq a \\ \dfrac{b-x}{b-a} & a \leq x \leq b \\ 1 & x \geq b \end{cases} \qquad 2.10$$



Gambar 2.10 Fungsi Keanggotaan Linier Up.

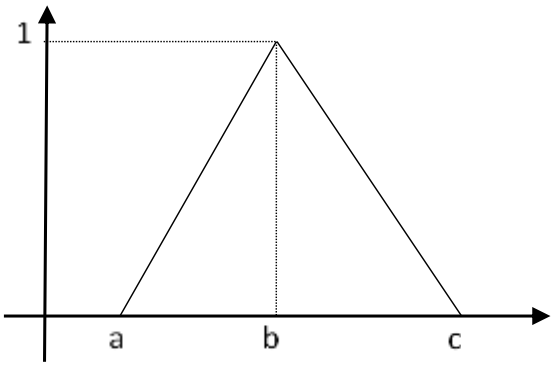


Gambar 2.11 Fungsi Keanggotaan Linier Down.

2.4.2.2 Representasi Kurva Segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara 2 garis (linear), fungsi keanggotaan segitiga ditunjukkan pada Persamaan 2.11 dengan bentuk grafik pada Gambar 2.12,

$$\mu[x] = \begin{cases} 0 & x \leq a \; atau \; x \geq c \\ \dfrac{x-a}{b-a} & a \leq x \leq b \\ \dfrac{b-x}{c-b} & b \leq x \leq c \end{cases} \qquad 2.11$$

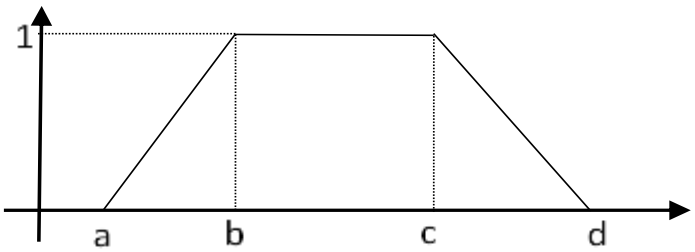


Gambar 2.12 Fungsi Keanggotaan Segitiga.

2.4.2.3 Representasi Kurva Trapesium

Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1, fungsi keanggotaan trapesium ditunjukkan pada Persamaan 2.12 dengan bentuk grafik pada Gambar 2.13,

$$\mu[x] = \begin{cases} 0 & x \leq a \; atau \; x \geq d \\ \dfrac{x-a}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \dfrac{d-x}{d-c} & c \leq x \leq d \end{cases} \qquad 2.12$$



Gambar 2.13 Fungsi Keanggotaan Trapesium.

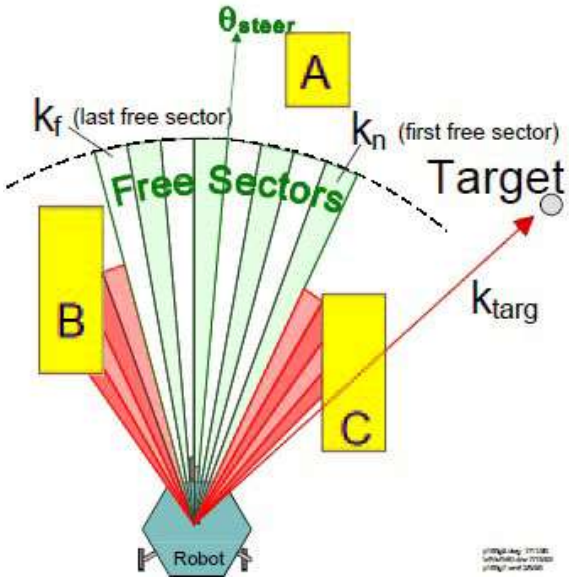
2.5 Vector Field Histogram

Vector Field Histogram (VFH) adalah algoritma *real-time* yang diusulkan oleh Johann Borenstein dan Yoram Koren pada tahun 1991 [14]. Algoritma ini awalnya berdasarkan metode VFF (*Virtual force field-algoritme*). Metode ini akhirnya dikembangkan karena ada kelemahan dalam metode VFF. Metode VFF sebenarnya kurang tahan terhadap persepsi yang keliru dari sensor. Metode VFH

ini juga memperhitungkan pengamatan sebelumnya dengan kecenderungan kesalahan terhadap pengukuran sehingga mempengaruhi arah robot. Pada tahun 1998, VFH disesuaikan dengan Iwan Ulrich dan Johann Borenstein dan berganti nama VFH+ [15] dan kemudian diperbarui untuk VFH* [16].

Dengan menggunakan *polar obstacle density* (POD) akan menentukan daerah yang cocok untuk robot bergerak. Untuk melakukan hal ini, terdapat batas tertentu yang digunakan, jika nilai sektor dari POD menunjukkan nilai di bawah nilai yang ditentukan maka dianggap aman, Sehingga robot akan menemukan beberapa daerah yang aman untuk bergerak dan memilih daerah yang tercepat untuk ke target. Arah θ di mana robot seharusnya maju ditentukan dengan melihat sektor yang paling dekat dengan target (k_n) dan sektor jauh dari targer (k_f) hal ini di tunjukkan pada Gambar 2.14. Ketika robot terlalu dekat atau terlalu jauh dari sektor terdekat, θ akan di adaptasi dan akan mengarahkan robot. Karena jarak maksimum antara sektor terdekat dan sektor akhir dapat bervariasi robot dengan bantuan VFH mampu melewati bagian sempit dan pintu. Arah ditentukan dengan Persamaan 2.13.

$$\theta = \frac{k_n + k_f}{2} \qquad 2.13$$



Gambar 2.14 Contoh Algoritma VFH [14].

2.6 Algoritme A*

Algoritme A* adalah algoritme yang digunakan dalam merintis jalan dan grafik traversal, proses merencanakan jalur efisien traversable antara beberapa poin, yang disebut node. algoritme ini dikenalkan pertama kali pada tahun 1968. Algorima ini merupakan perkembangan dari algoritme Dijkstra, A* memiliki kinerja yang lebih baik karena menggunakan heuristik untuk memandu pencarian.

Secara umum persamaan algoritme A* dapat dituliskan sebagai berikut,

$$f(n) = g(n) + h(n) \quad 2.14$$

di mana n adalah adalah node terakhir di jalan, g(n) adalah biaya jalan dari awal node ke n, dan h(n) adalah heuristik yang memperkirakan biaya dari jalur termurah dari n ke tujuan.

Heuristik adalah penilai yang memberi harga pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan adanya heuristik menjadikan algoritma A* lebih baik.

Tahapan dalam mencari atau menentukan algoritme A* adalah:

1. Tambahkan starting point ke dalam openset.
2. Ulangi langkah berikut:
    a. Carilah biaya F terendah pada setiap simpul dalam open set. Node dengan biaya F terendah kemudian disebut current node.
    b. Masukkan ke dalam closed set.
    c. Untuk setiap 8 simpul (neighbor node) yang berdekatan dengan current node:
        i. Jika tidak walkable atau jika termasuk closed set, maka abaikan.
        ii. Jika tidak ada pada open set, tambahkan ke open set.
        iii. Jika sudah ada pada open set, periksa apakah ini jalan dari simpul ini ke current node yang lebih baik dengan menggunakan biaya G sebagai ukurannya. Simpul dengan baik. Jika demikian, buatlah simpul ini (neighbor node) sebagai came from dari current node, dan menghitung

ulang nilai G dan F dari simpul ini biaya G yang lebih rendah berarti bahwa ini adalah jalan yang lebih.

d. Stop ketika Anda:

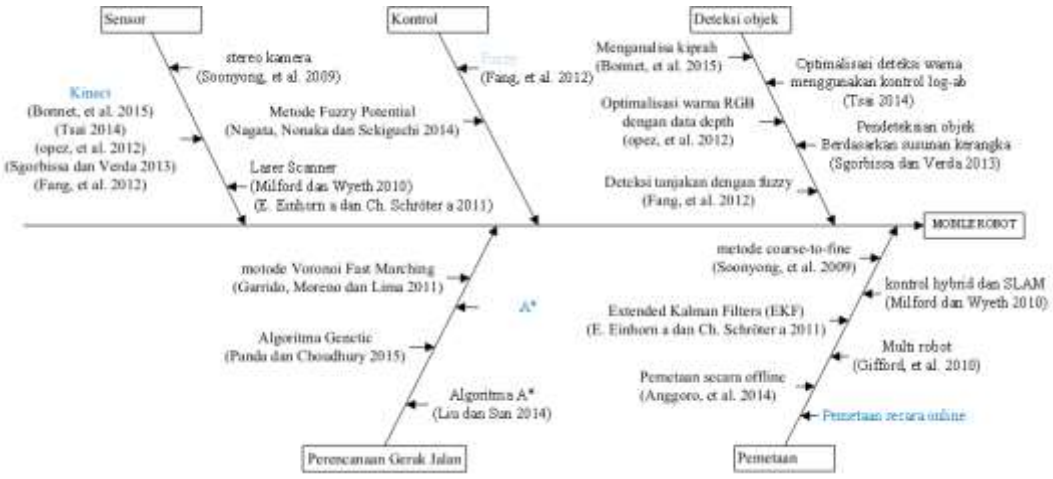
i. menambahkan target point ke dalam closed set, dalam hal ini jalan telah ditemukan, atau

ii. Gagal untuk menemukan target point, dan open set kosong. Dalam kasus ini, tidak ada jalan

3. Simpan jalan. Bekerja mundur dari target point, pergi dari masing-masing simpul ke simpul came from sampai mencapai starting point. Itu adalah jalan Anda.

2.7 Fishbone

Adapun penelitian yang akan dilakukan tergambarkan di dalam diagram fishbone yang ditunjukkan pada Gambar 2.15. Dalam penelitian ini terdapat lima kategori tentang mobile robot, yaitu mengenai sensor, kontrol, deteksi objek, perencanaan gerak jalan, dan pemetaan.



Gambar 2.15 Diagram Fishbone.

BAB 3
METODOLOGI PENELITIAN

Pada penelitian ini akan dilakukan sistem pemetaan untuk membantu proses evakuasi. Secara umum tahapan sebelum dilakukan evakuasi akan ada beberapa tahap seperti:

1. Ketika terjadi bencana alarm darurat akan dibunyikan
2. Informasi alarm akan di terima oleh tim evakuasi
3. Tim evakuasi melakukan persiapan dan meluncur ke lokasi
4. Tim evakuasi mempersiapkan beberapa peralatan, seperti alat keselamatan, alat bantu evakuasi dan lain-lain
5. Tim evakuasi melakukan evakuasi penyelamatan

Pada penelitian ini akan di tambahkan beberapa alur yang dilakukan oleh robot, ketika tim evakuasi menjalankan 5 tahap di atas, mobil robot akan melakukan beberapa tahap yaitu:

1. Ketika terjadi bencana alarm darurat akan dibunyikan
2. Mobil robot akan diluncurkan untuk mencari objek (benda berharga, atau korban)
3. Mobile robot menjelajah ruangan dan mencari korban
4. Tim evakuasi menerima data peta dan posisi objek.
5. Tim evakuasi melakukan evakuasi sesuai dengan peta yang di hasilkan

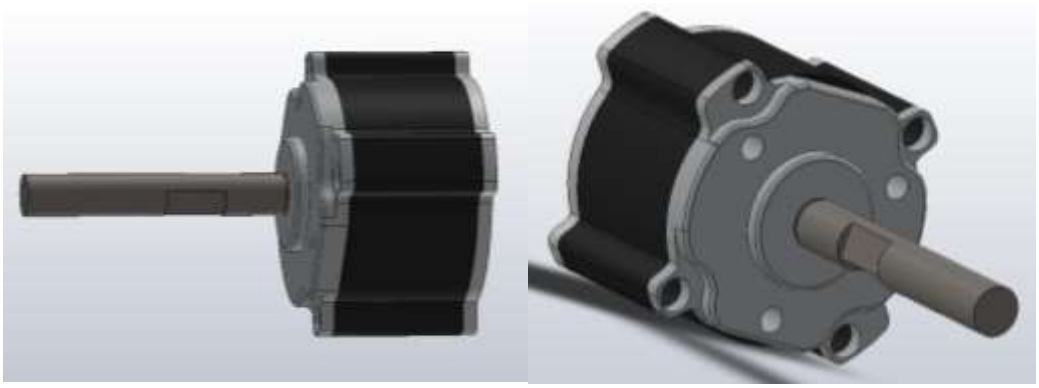
Keseluruhan tahapan digambarkan pada Gambar 3.1, jalur biru merupakan standar tim evakuasi, dan jalur hijau merupakan tahapan yang dilakukan oleh robot.

Gambar 3.1 Tahapan Evakuasi

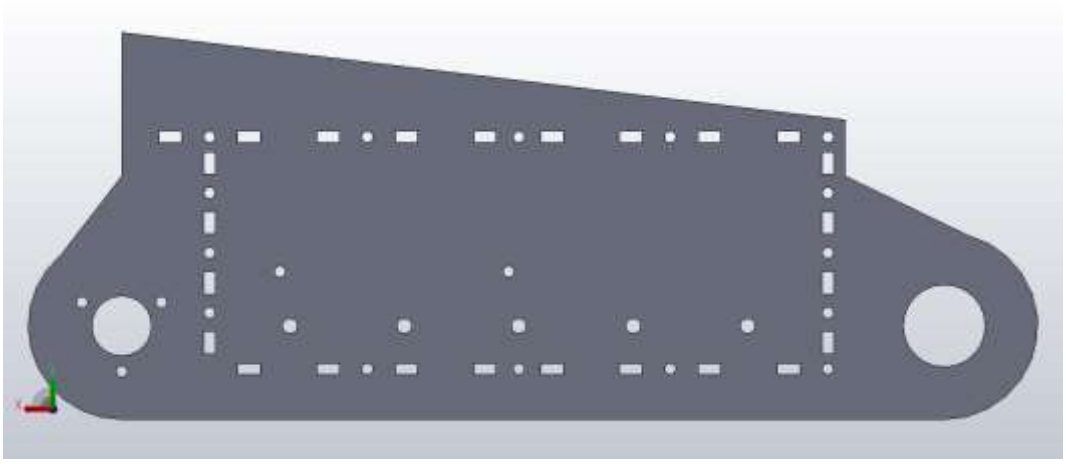
3.1 Perancangan Mobile Robot

Mobile robot yang digunakan adalah tipe tank. Tipe tank dipilih karena memiliki roda yang memberikan kestabilan terhadap jalan yang bergelombang. Perancangan robot meliputi rangkaian elektronik, bentuk badan robot, desain roda.

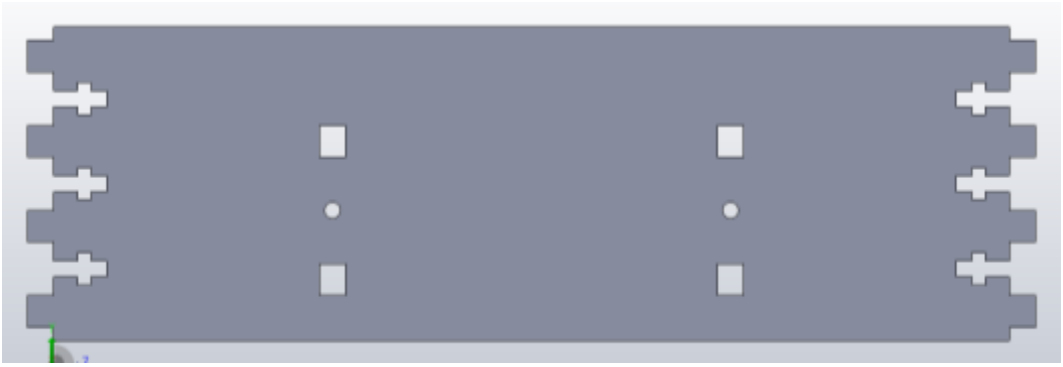
Sebelum membuat badan robot terlebih dahulu melakukan pengukuran komponen yang melekat, seperti luas area untuk rangkaian elektronik, dudukan motor pada badan robot, poros roda dan lain-lain. Untuk memudahkan dalam simulasi desain maka dibuat bentuk motor sesuai dengan skala motor, bentuk motor dapat di lihat pada Gambar 3.2



Gambar 3.2 Bentuk Motor Faulhaber DC 12 V



Gambar 3.3 Bentuk Bodi Bagian Samping

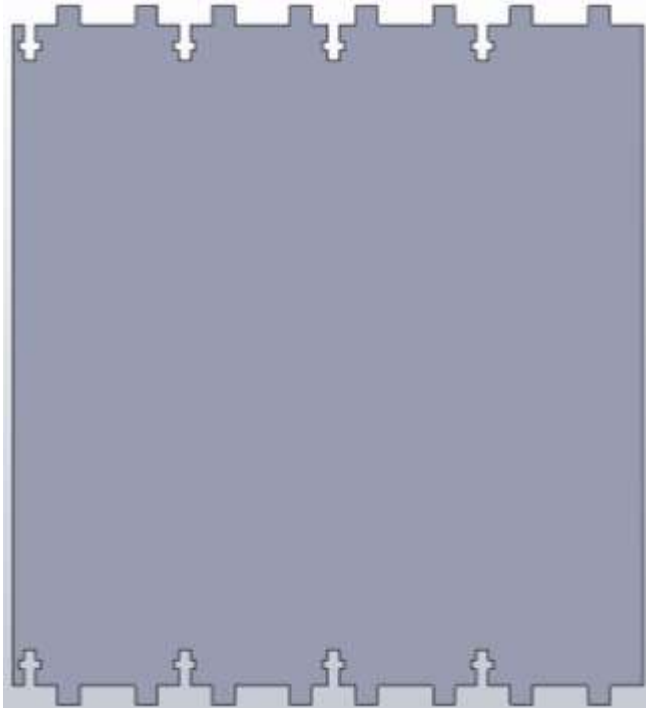


Gambar 3.4 Bentuk Desain Bodi Depan dan Belakang

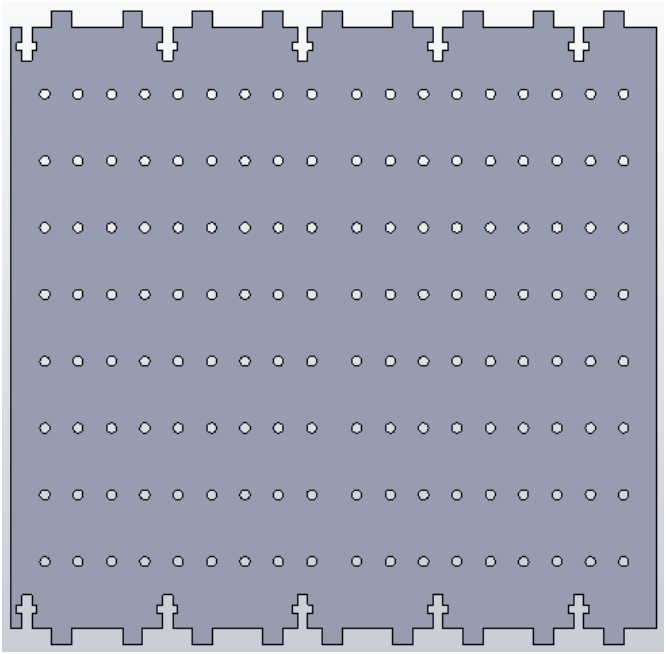
Perancangan bodi robot berikutnya membuat kerangka bagian samping, kerangka ini harus memiliki kekuatan yang cukup sehingga di pilih ketebalan 5mm, pemberian lubang pada bodi samping agar semua bagian bodi bisa terpasang dengan kokoh, bentuk desain bodi samping di tunjukkan pada Gambar 3.3.

Pembuatan bodi berikutnya adalah bagian depan dan belakang, pada bagian samping dari desain harus dapat masuk pada lubang-lubang bodi bagian samping. Pemberian lubang berbentuk + di gunakan sebagai tempat dari baut dan ringnya. adapun bentuk desain bodi depan dan samping di tunjukkan pada Gambar 3.4. Lubang pada bagian tengah digunakan sebagai dudukan Ass roda.

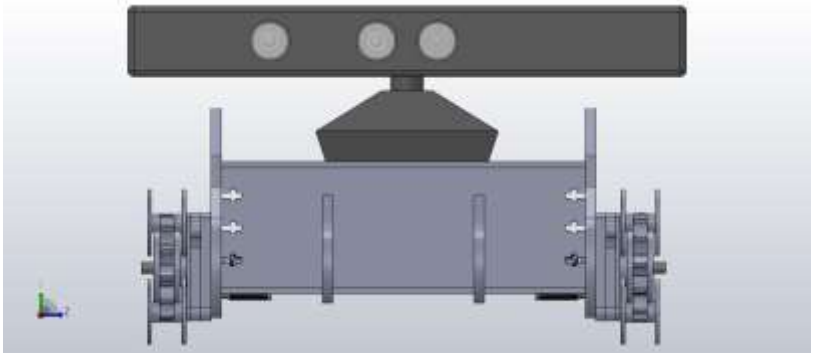
Desain berikutnya adalah penutup bagian bawah dan atas, bentuk penutup bagian atas dan bawa memiliki bentuk yang sama. Tetapi pada penutup bagian atas dibuat berlubang-lubang. Tujuan lubang-lubang pada penutup atas adalah agar udara bisa masuk sebagai pendinginan, sekaligus digunakan untuk merekatkan komponen tambahan, seperti Kinect, antena untuk kompas, dan hal tambahan yang belum diketahui. Adapun bentuk penutup bagian bawah ditunjukkan pada Gambar 3.5. Sedangkan penutup bagian atas di tunjukkan pada gambar Gambar 3.6



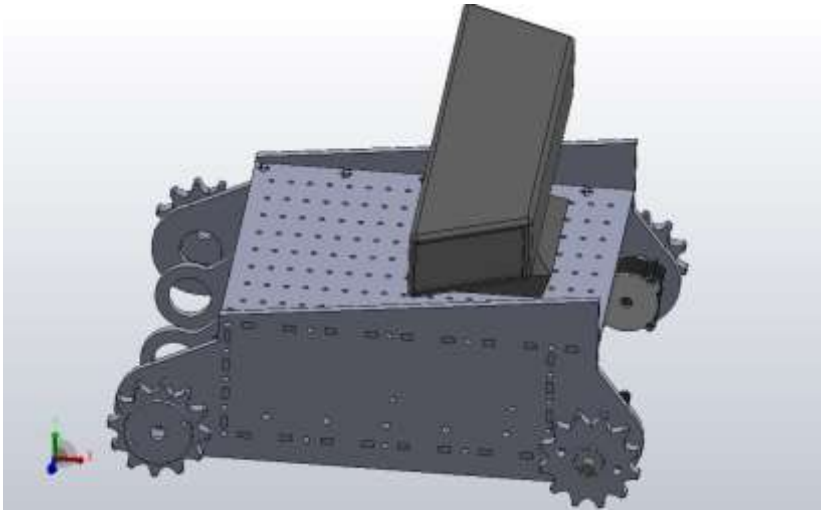
Gambar 3.5 Bentuk Penutup Bagian Bawah



Gambar 3.6 Bentuk Penutup Bagian Atas



Gambar 3.7 Bentuk Desain Robot Keseluruhan Dilihat Dari Depan



Gambar 3.8 Bentuk Desain Robot Keseluruhan Dilihat Dari Samping

Ketika semua bodi telah dibuat maka langkah selanjutnya menyatukan untuk membuktikan bahwa ukuran setiap bagian sesuai dengan bagian yang lainya. Adapun bentuk desain setelah di satukan dapat di lihat pada Gambar 3.7 dan Gambar 3.8

Roda tank di desain menggunakan tipe rantai konveyor *RS-40-1 K-1 Attachment*. Rantai sudah memiliki kuping di sisi kanan dan kiri yang akan digunakan untuk karet roda. Pelat T digunakan untuk melekatkan karet ke rantai.bentuk T dipilih karena bentuk ini memiliki tiga sisi, sisi pertama digunakan untuk menjepit kared, dan dua sisi yang lain digunakan untuk melekatkan ke kuping rantai. Adapun bentuk karet roda dapat di lihat pada Gambar 3.9. Semu karet roda akan di pasang pada rantai.

Gambar 3.9 Desain Karet Rantai Roda Tank.

Pemasangan karet bertujuan untuk mengurangi terjadinya slip pada roda. semua karet rantai di pasang ke kuping rantai, sistem penguncian karet memanfaat kan bentuk T pada pelat. Dengan membengkokkan pelat T ke kuping rantai, maka karet rantai akan terpasang dengan kuat. Adapun hasil dari roda di tunjukkan pada Gambar 3.10.

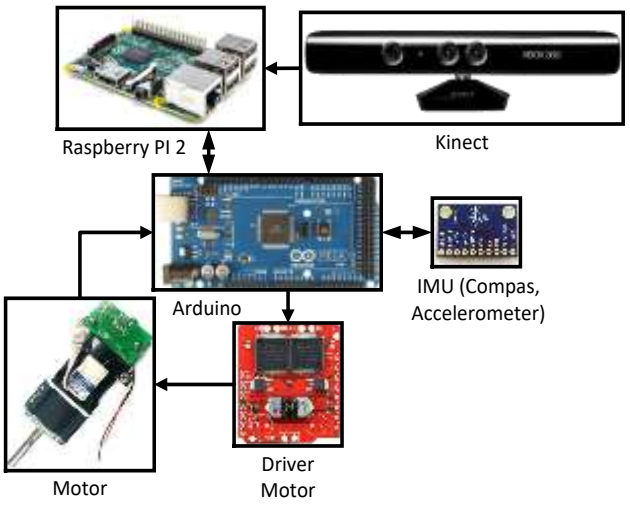
Gambar 3.10 Desain Rantai Roda Tank

Gambar 3.11 Desain Gigi Roda Belakan Untuk Roda Tank.

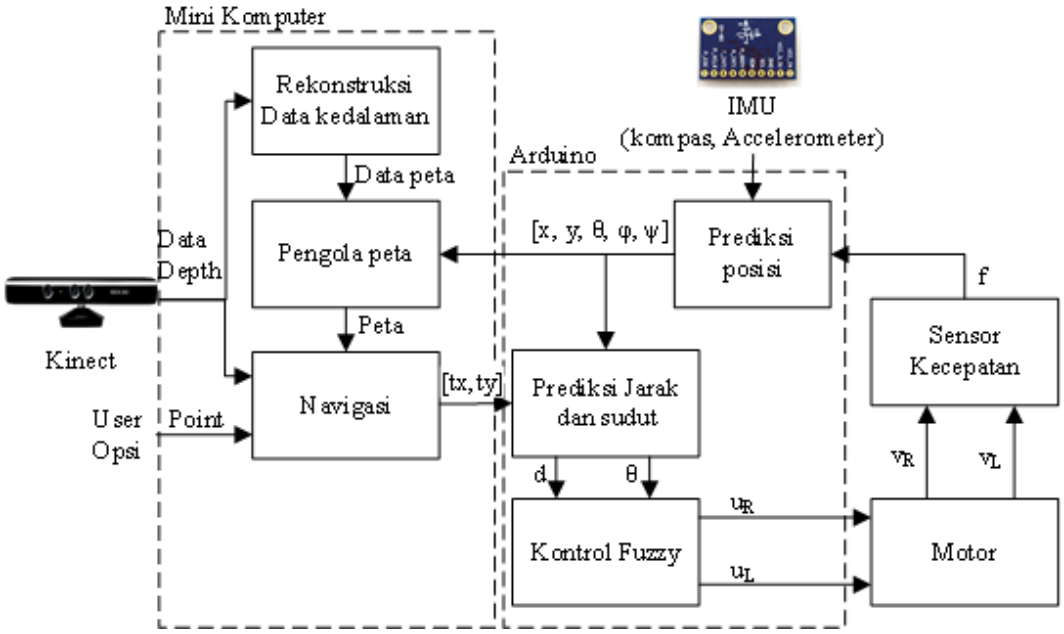
Penggerak roda menggunakan dua motor DC. motor DC menggunakan konsumsi daya 12 V. Sistem gigi roda dibuat menggunakan gir sepeda motor bagian depan. Bentuk desain gir di tunjukkan pada Gambar 3.11.

3.2 Perancangan Sistem

Sistem perancangan yang akan dibuat dalam penelitian ditunjukkan pada Gambar 3.12. komponen yang diguanakn di antaranya Kinect, Raspberry PI 2, Arduino MEGA 2560, IMU GY-521 (*Accelerometer*), IMU GY-271 (kompas), driver motor dual VNH2SP30, dan motor Faulhaber DC 12V,Alur proses setiap komponen di tunjukkan pada Gambar 3.13. Kontrol utama dilakukan oleh sebuah mini komputer (Raspberry PI 2), pengguna memasukkan batas area yang akan dijelajah oleh robot, setelah batas area dimasukkan, komputer akan mengirimkan sinyal kontrol untuk menjelajah area sesuai dengan batas yang dimasukkan. Sinyal kontrol berdasarkan data-data dari sensor kinect dan *database* penyimpanan (jika area sudah pernah dijelajah sebelumnya), berdasarkan keduanya akan diperhitungkan rute yang memiliki jarak paling optimum.



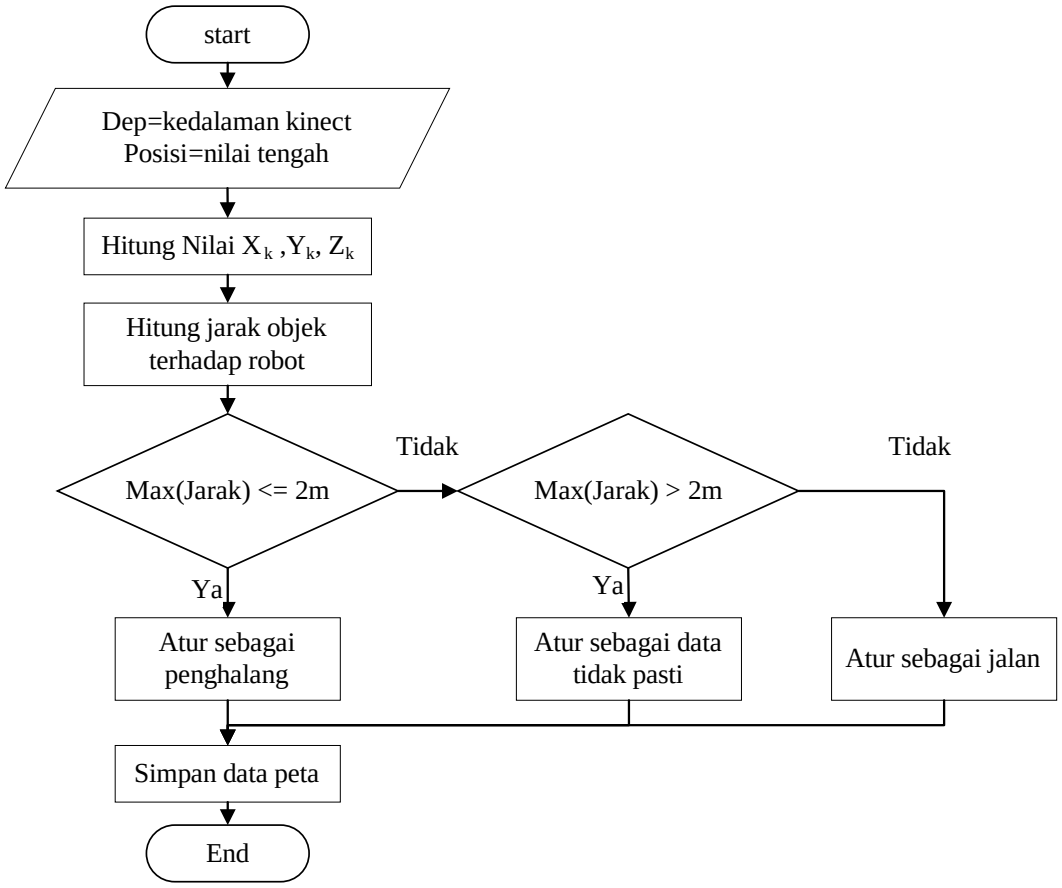
Gambar 3.12 Perancangan Sistem Mobile Robot.



Gambar 3.13 Diagram Blok Perancangan Sistem.

3.2.1 Rekonstruksi data kedalaman(*Depth*)

Rekonstruksi data kedalaman merupakan tahapan mengonversi data *Depth* ke dalam data peta. Adapun alur pendeteksian sesuai dengan diagram Gambar 3.14. alur dari rekonstruksi di mulai dari pengambilan data *Depth* Kinect, kemudian mengambil sampel pada bagian tengah dari data *Depth*. Data sampel akan dihitung dalam koordinat Cartesius. Dari data koordinat Cartesius dapat ditentukan jarak objek di depannya. Berdasarkan jarak objek kita dapat menentukan apakah jarak tersebut menggambarkan sebuah penghalang atau sebuah jalan.



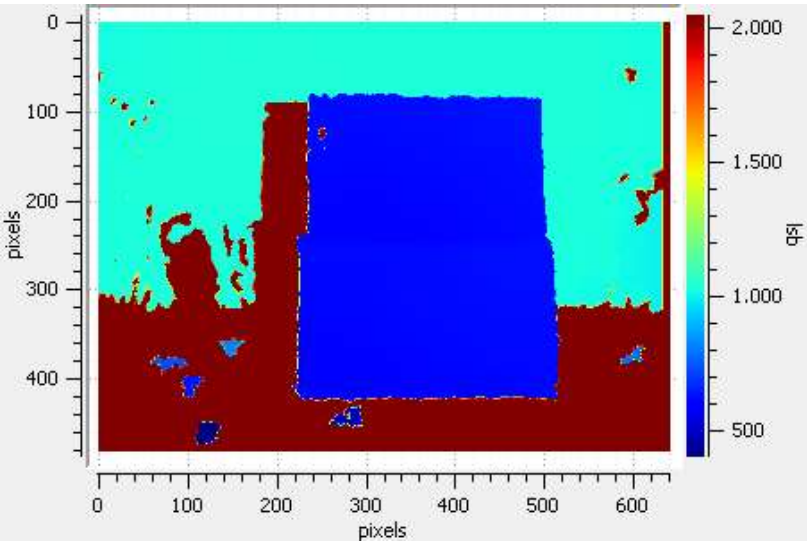
Gambar 3.14 Alur Rekonstruksi Data Kedalaman.

Kinect menghasilkan data-data dalam bentuk array 2d yang di tunjukkan pada Gambar 3.15. setiap nilai *pixel* di presentasikan dalam tingkatan warna atau *Least Significant Bit* (LSB). Setiap *pixel* menggambarkan jarak dan posisi. Data *Depth* Kinect di presentasikan dalam data 11-bit. Adapun persamaan yang di gunakan untuk mengonversi nilai tiap bit ke dalam koordinat Cartesius atau ke dalam X_k,Y_k,Z_k di tunjukkan dalam Persamaan 3.1 – Persamaan 3.2,

$$Z_k(i,j) = 0.1236 * \tan\left(\frac{dept(i,j)}{2842.5} + 1.1863\right) \quad 3.1$$

$$X_k(i,j) = \left(i - \frac{w}{2}\right) * 1.12032 * \frac{Z_k(i,j)}{1000} \quad 3.2$$

$$Y_k(i,j) = \left(j - \frac{h}{2}\right) * 0.84024 * \frac{Z_k(i,j)}{1000} \quad 3.3$$

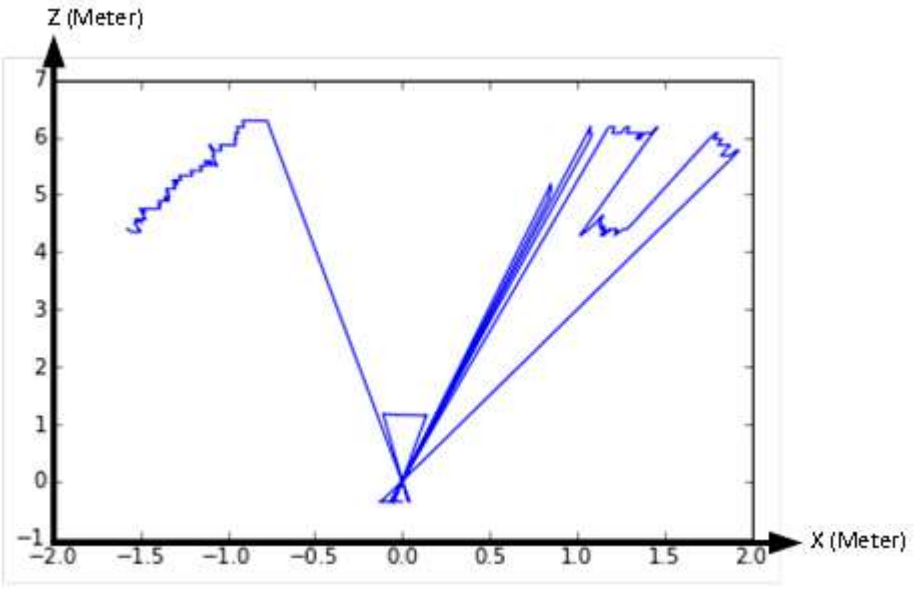


Gambar 3.15 Depth Kinect

Berdasarkan Persamaan 3.1 – Persamaan 3.2, dibuat sebuah fungsi program untuk mengonversi data *Depth* ke dalam koordinat Cartesius. Dalam fungsi yang dibuat di perlukan dua parameter yaitu, data *Depth* dan posisi tengah *Depth*. Data *Depth* akan di potong untuk mempercepat proses perhitungan. Pemotongan data *Depth* menghasilkan data *Depth* pada posisi tengah saja. Adapun fungsi program yang dibuat sesuai dengan persamaan di atas adalah

```
1.  def maping(dep,posisi=240):
2.      wit=640
3.      jarak=stats.mode(dep[posisi:posisi+20,:])[0]
4.      jarak=0.1236 * np.tan(jarak / 2842.5 + 1.1863)
5.      jarak=np.clip(jarak,-2,3.25)
6.      minDistance = -10
7.      scaleFactor = -0.0021
8.
9.      _x=np.zeros(wit)
10.     for x in range(wit-1):
11.         _x[x] = (x-320)*1.12032*jarak[x]/1000
12.     return jarak,_x
```

Gambar 3.16 merupakan contoh hasil perhitungan antara X_k dan Z_k dari Gambar 3.15 untuk j=240. Dari data X_k,Y_k,Z_k yang ada pada *Depth* Kinect, dapat menentukan adanya jalan dan penghalang. Prinsip pendeteksian ini berdasarkan dari kemampuan Kinect dalam membaca jarak maksimal.



Gambar 3.16 Perbandingan X dan Z pada j=240

Berdasarkan data jarak yang sudah didapatkan, maka akan dapat di tentukan mana yang merupakan sebuah penghalang atau merupakan sebuah jalan. Perbedaan tersebut dapat dilihat jelas pada Gambar 3.16. Ketika data kurang dari 2 meter sudah di pastikan posisi tersebut terdapat sebuah penghalang, hal ini dapat di tuliskan dalam program untuk menentukan data termasuk penghalang atau dinding

```
1.  def status(xr,yr,x,y,ujng,mp):
2.      global db_map
3.      x,y=mp
4.      batas=2
5.      if ujng and abs(xr-x)<=batas and abs(yr-y)<=batas:
6.          db_map[leb+x/0.25,leb+y/0.25]=3
7.      elif abs(xr-x)<=batas and abs(yr-y)<batas:
8.          db_map[leb+x/0.25,leb+y/0.25]=2
9.      elif db_map[leb+x/0.25,leb+y/0.25]<=1:
10.         db_map[leb+x/0.25,leb+y/0.25]=1
```

3.2.2 Pengolahan Peta

Pada tahap ini akan dirancang sistem pengolahan peta dari hasil rekonstruksi data kedalaman sensor Kinect. Data jalan hasil rekonstruksi akan diperiksa apakah terdapat jalan yang dapat dilewati oleh mobile robot atau tidak, ketik terdapat data jalan yang dapat dilewati oleh mobile robot, maka data tersebut akan ditambah atau diperbaharui ke *databases* peta, aksi disimpan atau diperbaharui tergantung dari apakah pada koordinat tersebut sudah memiliki data peta atau tidak. Ketika data jalan yang dapat dilewati tidak ditemukan maka akan dilakukan

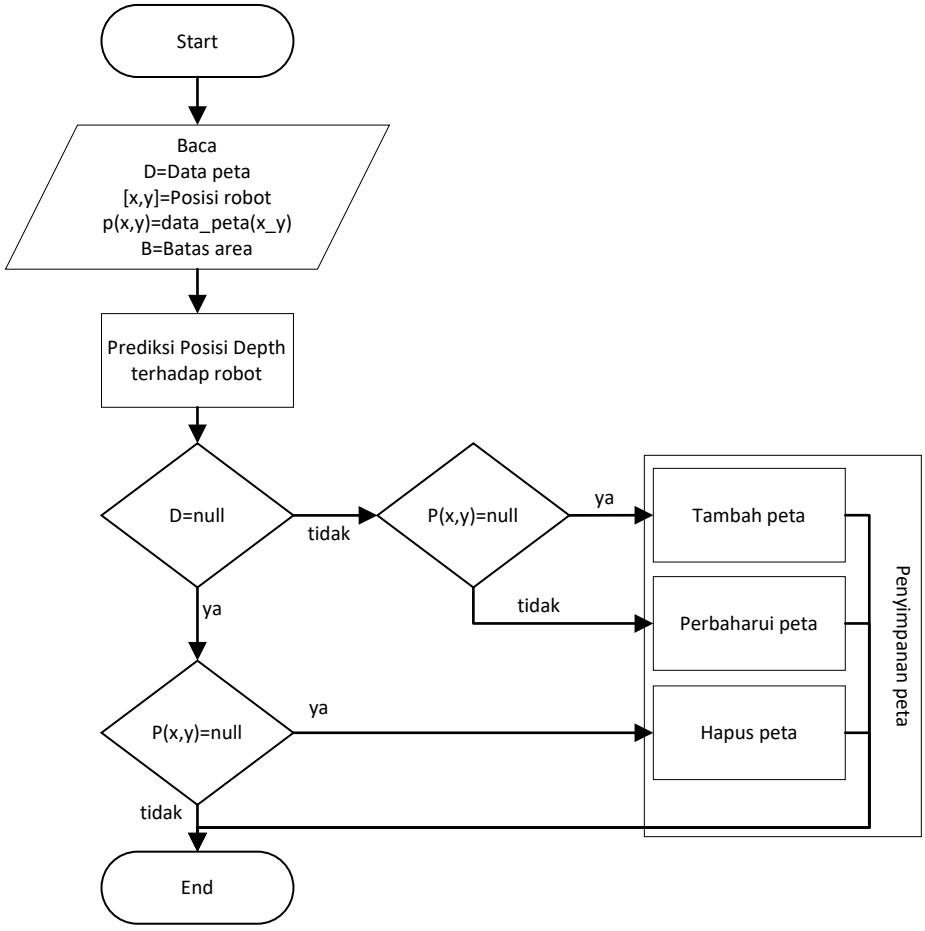
pemeriksaan pada *database*, ketika pada koordinat tersebut pernah ada data peta maka data peta tersebut akan dihapus. Adapun alur dari pengolahan peta di tunjukkan pada Gambar 3.17.

Data X_k,Y_k,Z_k Kinect akan di rubah sesuai dengan posisi robot. Hubungan X_k,Y_k,Z_k terhadap posisi robot ditunjukkan pada Gambar 3.18. Berdasarkan hubungan Gambar 3.18 di dapatkan Persamaan 3.4-Persamaan 3.6.

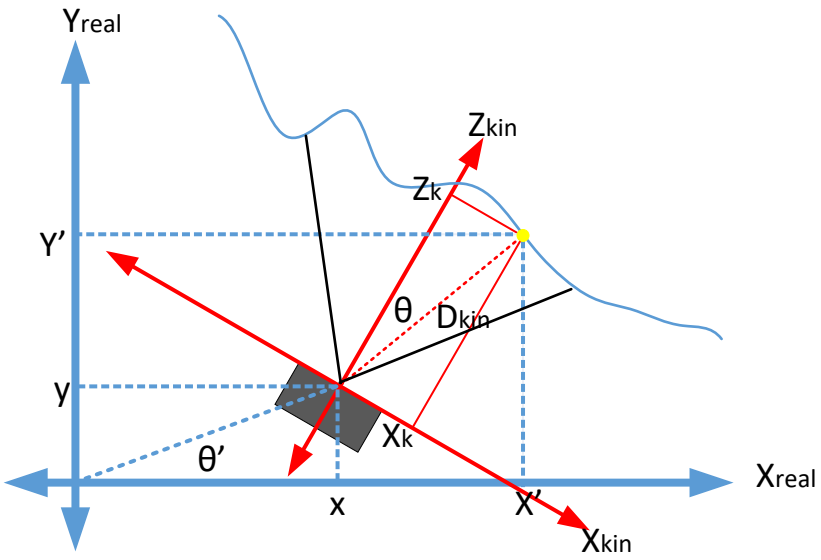
$$Dkin = \sqrt{Xk^2 + Zk^2} \quad 3.4$$

$$X' = x + (Dkin * \sin(\theta' + \theta)) \quad 3.5$$

$$Y' = y + (Dkin * \cos(\theta' + \theta)) \quad 3.6$$



Gambar 3.17 Alur Sistem Pengolahan Peta.



Gambar 3.18 Ilustrasi Pembacaan Data Kinect Terhadap Posisi Robot

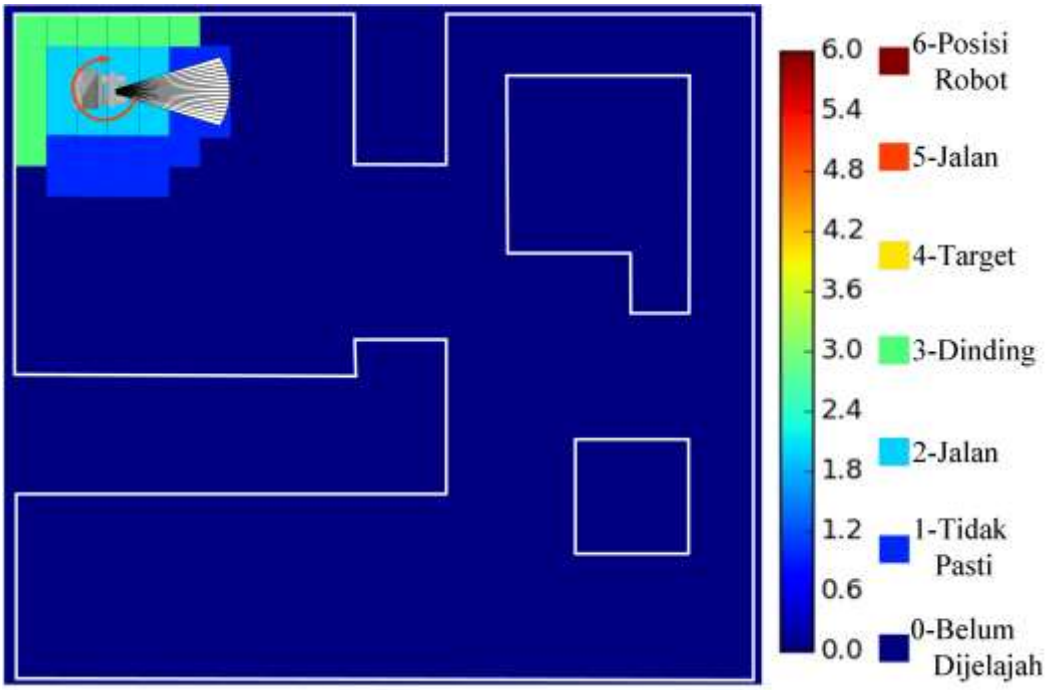
Data peta dihasilkan dari proses rekonstruksi data kedalaman(*Depth)*. Data tersebut sudah di bagi menjadi beberapa tingkatan, di antaranya adalah data dinding, data jalan, dan data tidak pasti. *Database* peta adalah data peta yang sudah disimpan berdasarkan posisi robot. Perubahan posisi data peta ke dalam posisi robot dilakukan pada fungsi program yang berdasarkan Persamaan 3.4-Persamaan 3.6

```
1. def koordinat(dep_x,dep_z,rob_x,rob_y,rob_t):
2.     dl=np.sqrt(pow(dep_x,2)+pow(dep_z,2))
3.     dt=np.pi/2-np.arctan2(dep_z,dep_x)
4.     x=rob_x+dl*np.cos(rob_t-dt)
5.     y=rob_y+dl*np.sin(rob_t-dt)
6.     return x,y
```

Data peta akan menentukan proses apakah *Database* peta akan di tambah, di perbaharui, atau di hapus. *Database* peta akan di tambahkan jika pada data peta menunjukkan sebuah peta sedangkan *database* peta tidak ada sebuah peta. *Database* peta akan di perbaharui ketika data peta berbeda dengan *database* peta. Sedangkan *database* peta akan di hapus ketika data peta menunjukkan bahwa *database* peta sudah tidak ada. Berdasarkan prinsip di atas maka dibuat fungsi program penyimpanan *database* peta dapat dibuat sebagai berikut

```
1.    def simpan(kin_x,kin_z,r_x,r_y,r_tt,r_a,r_b):
2.        global db_map,kotak,leb
3.        sim_lokasi(r_x,r_y,r_tt,r_a,r_b)
4.        kin_x,kin_z=koordinat(kin_x,kin_z,0,0,r_tt)
5.        r_x=r_x/kotak+leb
6.        r_y=r_y/kotak+leb
7.
      x_1,y_1,x_2,y_2,x_3,y_3=simpan_diag(db_map,kin_x/kotak,k
      in_z/kotak,r_x,r_y)
8.        db_map[x_1.tolist(),y_1.tolist()]=1
9.        db_map[x_2.tolist(),y_2.tolist()]=2
10.       db_map[x_3.tolist(),y_3.tolist()]=3
11.
12.   def simpan_diag(db_map,x,z,r_x,r_y):
13.       t=np.arctan2(x,z)
14.       l=np.round(np.sqrt(x**2+z**2))
15.       x_0=[]
16.       y_0=[]
17.       for i in xrange(0,len(l),42):
18.           a=np.arange(l[i])
19.           x_0+=(a*np.sin(t[i])).tolist()
20.           y_0+=(a*np.cos(t[i])).tolist()
21.       x_0=np.array(x_0)
22.       y_0=np.array(y_0)
23.       p=np.round(np.sqrt(x_0**2+y_0**2))
24.       i=np.where(p>20)
25.       x_2=np.int8(np.delete(x_0,i))+r_x
26.       y_2=np.int8(np.delete(y_0,i))+r_y
27.       i=np.append(np.where(l>20),np.where(l<=1))
28.       z_3=np.int8(np.delete(z,i))+r_y
29.       x_3=np.int8(np.delete(x,i))+r_x
30.       i=np.where(p<20)
31.       x_1=np.int8(np.delete(x_0,i))+r_x
32.       y_1=np.int8(np.delete(y_0,i))+r_y
33.       i=np.where(db_map[x_1.tolist(),y_1.tolist()]>0)[0]
34.       x_1=np.delete(x_1,i)
35.       y_1=np.delete(y_1,i)
36.
37.       return x_1,y_1,x_2,y_2,x_3,z_3
```

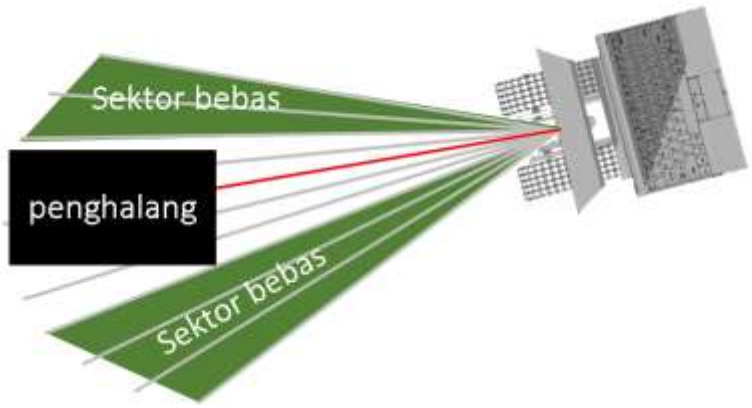
Parameter yang digunakan pada fungsi program di atas adalah data koordinat Cartesius (nilai x dan z), dan posisi robot (x, y, θ, α, β). Dalam fungsi program di atas akan menghasil luas area jalan, luas area penghalang, dan luas area data tidak pasti. Contoh ilustrasi data peta ketika robot berputar untuk pertama kali di tunjukkan pada Gambar 3.19.



Gambar 3.19 Ilustrasi Database Peta Ketika Robot Berputar

3.2.3 Perancangan Gerak Pemetaan

Perancangan gerak robot dipengaruhi oleh penghalang dan data peta yang telah tersimpan. Ketika pada peta terdapat data yang di set sebagai data yang tidak pasti. Maka lokasi data tersebut akan digunakan sebagai tujuan robot berikutnya. Robot akan bergerak sesuai dengan tujuan yang di tetapkan. Ketika terdapat penghalang, robot akan menghindari penghalang dengan membaca sektor yang bebas dari data Kinect dan data peta. Beberapa sektor yang bebas akan diambil nilai tengah, nilai tengah tersebut merupakan sudut referensi belok robot, hal ini diilustrasikan pada Gambar 3.20.



Gambar 3.20 Kondisi Sektor Sensor Kinect Terhadap Penghalang.

Berdasarkan ilustrasi di atas maka dapat di buat sebuah program yang dapat membagi-bagi sesuai dengan sektornya. Sehingga dibuat fungsi utama untuk megatur alur pembagian, fungsi tersebut dapat di tuliskan dalam program

```
1.    def kontrol(db,x,y,t,tx,ty):
2.        j=min(5,int(np.sqrt(np.power(x-tx,2)+np.power(y-
            ty,2))/0.25))
3.        t1=np.arctan2(ty-y,tx-x)
4.        s=data(db,t1,j)
5.        return gerak(s,x,y,t,t1,(j)*0.25)
```

dari program di atas merupakan program utama untuk kendali berdasarkan histogram, sebelum melakukan perencanaan gerak data harus diruba dalam bentuk histogram. Untuk mengubah dara menjadi histrogram maka dibuat dalam program

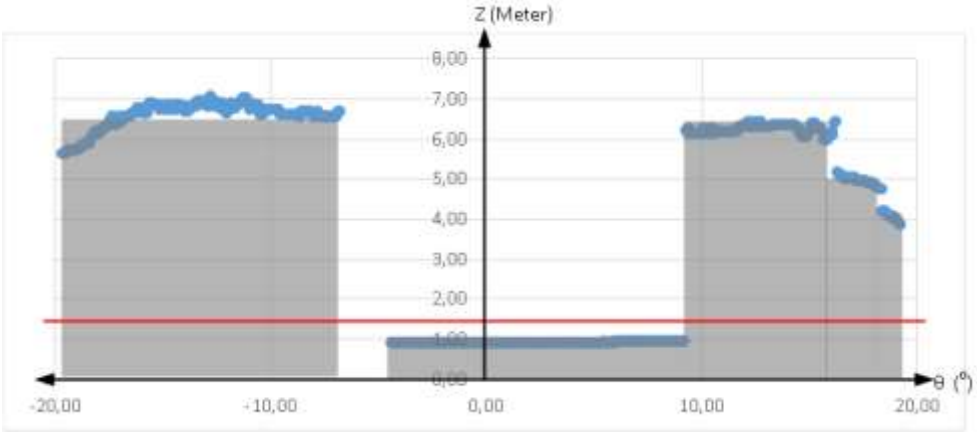
```
1.    def data(data,t,j):
2.        aa=[]
3.        #print j
4.        for sudut in np.arange(np.pi,-np.pi-0.01,-np.pi/8):
5.            h=0
6.            for panjang in range(j):
7.                x=4+round(panjang*np.cos(sudut+t))
8.                y=4+round(panjang*np.sin(sudut+t))
9.                if data[x,y]==3:
10.                   Break
11.               h+=0.25
12.           aa.append(h)
13.       return aa
```

setelah data histogram dibuat maka langkah selanjutnya menentukan arah gerak dari mobail robot. Perencanaan gerak terletak pada fungsi gerak, fungsi ini menghasilkan koordinat baru yang akan di telusuri robot. Data koordinat berdasarkan vektor yang bebas, sesuai ilustrasi pada Gambar 3.20. adapun program

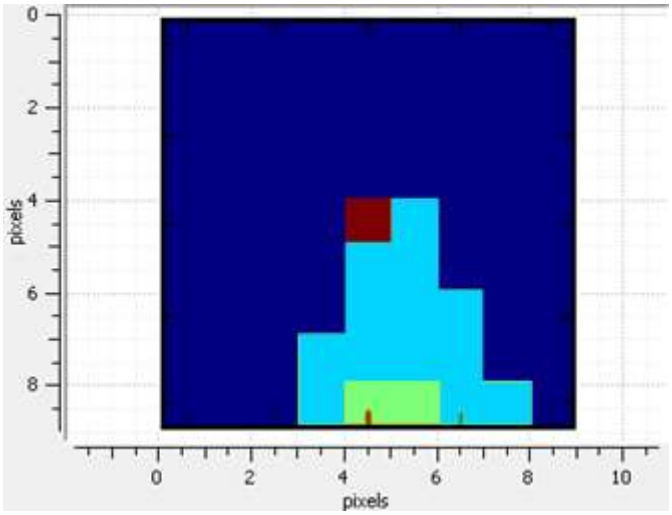
pada fungsi gerak, sesuai dengan perencana gerak dapat dituliskan

```
def gerak(data,x,y,t,t1,j):
    global tem_arah
    ren=2;
    s=np.sin(0)
    kiri,kanan=0,0
    p=8+int(round(s/0.125))
    for i in xrange(p,ren-1,-1):
        if min(data[i:i+2])>=j:
            Break
        kiri=kiri+1

    if p<17-ren:
        for i in xrange(max(ren-1,p),17,1):
            if min(data[i-1:i+1])>=j:
                Break
            kanan=kanan+1

    if min(kanan,kiri)==0:
        a=t1
        j=min(j,data[8])
        tem_arah=0
    elif(kanan<kiri) or (kanan==kiri and tem_arah>=0):
        j=min(j,min(data[6+kanan:8+kanan]))
        kanan=kanan*0.25
        a=t1-np.arctan2(kanan,j)
        tem_arah=1
        j=np.sqrt(j*j+kanan*kanan)
    else:
        j=min(j,min(data[9-kiri:11-kiri]))
        kiri=kiri*0.25
        a=t1+np.arctan2(kiri,j)
        tem_arah=-1
        j=np.sqrt(j*j+kiri*kiri)
    return j*np.cos(a)+x,j*np.sin(a)+y
```

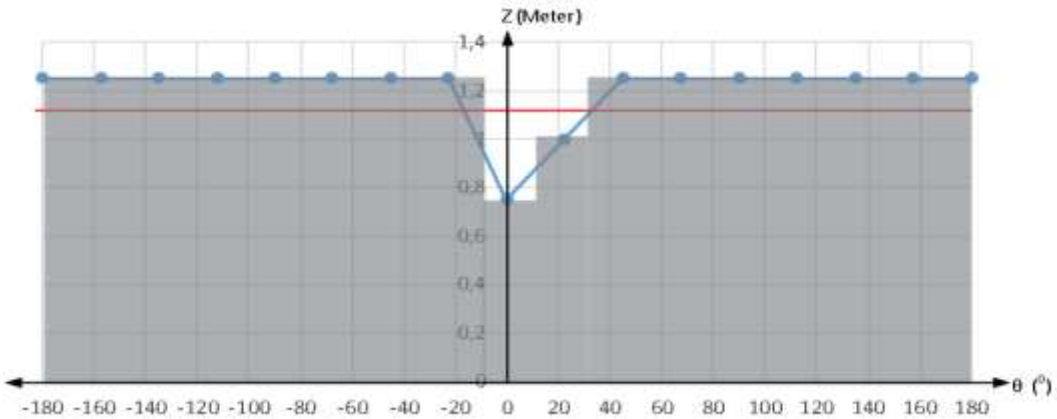
Data *Depth* Kinect dan data peta akan diubah dalam histogram polar. Gambar 3.21 merupakan bentuk histogram polar dari data Kinect. Gambar 3.22 merupakan contoh pengambilan data peta sesuai dengan posisi robot. Data peta yang digunakan untuk membuat histogram polar merupakan sebagian data peta sesuai dengan posisi robot. Gambar 3.23 merupakan data histogram polar dari data peta. Histogram polar *Depth* Kinect dan histogram polar peta akan digabungkan dengan membentuk satu histogram polar yang ditunjukkan pada Gambar 3.24.



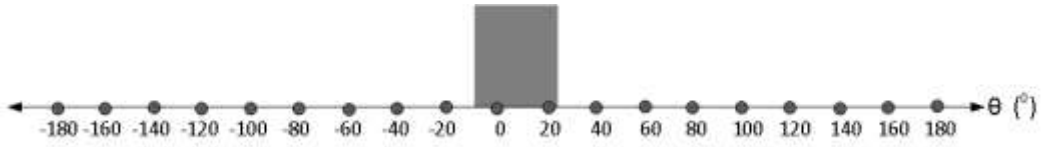
Gambar 3.21 Histogram Polar Data *Deptch* Kinect



Gambar 3.22 Mini Peta Sebagai Referensi Navigasi



Gambar 3.23 Histogram Polar Data Peta



Gambar 3.24 Hasil Histogram Polar

3.2.4 Prediksi posisi

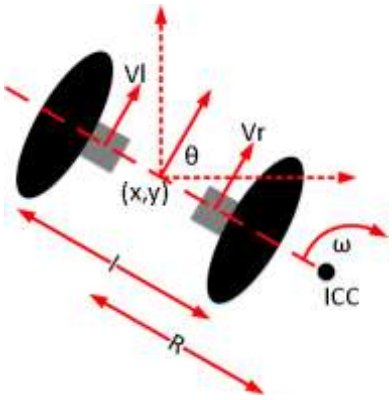
Pada tahap ini dilakukan perhitungan untuk menemukan posisi robot dalam koordinat Cartesius. Sensor yang digunakan adalah rotay encoder. Sensor ini mampu menunjukkan kecepatan dari masing-masing motor. Berdasarkan kecepatan masing-masing motor akan dapat ditentukan posisi robot. Hal ini di ilustrasikan pada Gambar 3.25. Adapun persamaan yang digunakan adalah Persamaan 3.7 – Persamaan 3.10

$$R = \frac{l}{2}\frac{(V_l + V_r)}{V_r - V_l} \tag{3.7}$$

$$ICC = [x - R * \sin\theta\,, y + R * \cos\theta] \tag{3.8}$$

$$\omega = \frac{v_r - v_l}{l} \tag{3.9}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega \end{bmatrix} \tag{3.10}$$



Gambar 3.25 Hubungan Posisi Dengan Kecepatan Motor

Agar proses menghitung dapat berjalan dengan baik, tanpa ada tunda waktu. Maka proses ini akan di lakukan oleh Arduino. Sehingga berdasarkan Persamaan 3.8 – Persamaan 3.10, maka dapat di buat program seperti

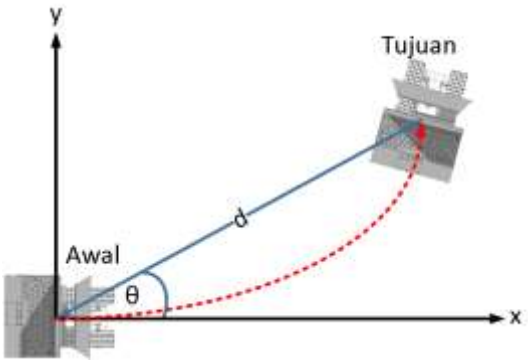
```
1.  void def_kin(double vr,double vl,double x,double y,double t){
2.     double l=jarak_roda;
3.     double R=0,w=0;
4.     if(vr-vl!=0)
5.           R=(l/2)*((vl+vr)/(vr-vl));
6.     w=(vr-vl)/l;
7.     double ICC[2]={x-R*sin(t),y+R*cos(t)};
8.     double b[3][3]={{cos(w),-sin(w),0},{sin(w),cos(w),0},
           {0,0,1}};
9.     double c[3][1]={{x-ICC[0]},{y-ICC[1]},{t}};
10.    double d[3][1]={{ICC[0]},{ICC[1]},{w}};
11.    double a[3][1];//=b*c+d
12.    Matrix.Multiply((float*)b,(float*)c,3,3,1,(float*)a);
13.    Matrix.Add((float*) a, (float*) d, 3, 1, (float*) a);
14.    _x=a[0][0];
15.    _y=a[1][0];
16. }
```

3.2.5 Prediksi jarak dan sudut

Prediksi jarak dan sudut merupakan perhitungan posisi awal dengan posisi target. Dalam prediksi ini akan di hasilkan nilai jarak dan sudut yang didapatkan dari Persamaan 3.11 – Persamaan 3.12 yang berdasarkan ilustrasi Gambar 3.26. Nilai jarak dan sudut akan digunakan sebagai aksi kontrol logika Fuzzy.

$$d = \sqrt{(x - tx)^2 + (y - ty)^2} \qquad 3.11$$

$$\theta = \tan 2^{-1}(x, y) \qquad 3.12$$

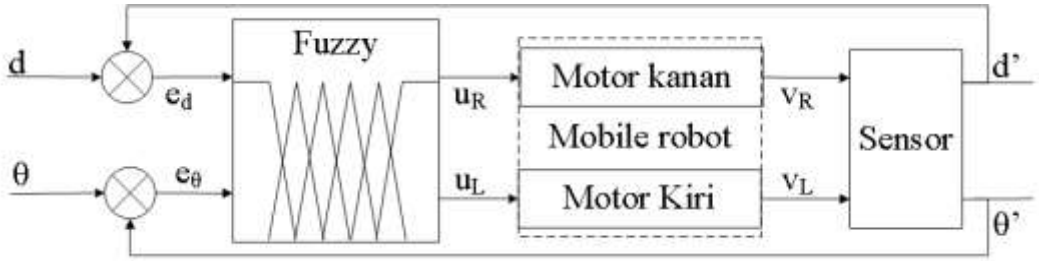


Gambar 3.26 Ilustrasi Jarak dan Sudut Untuk Posisi Awal Dengan Posisi Tujuan

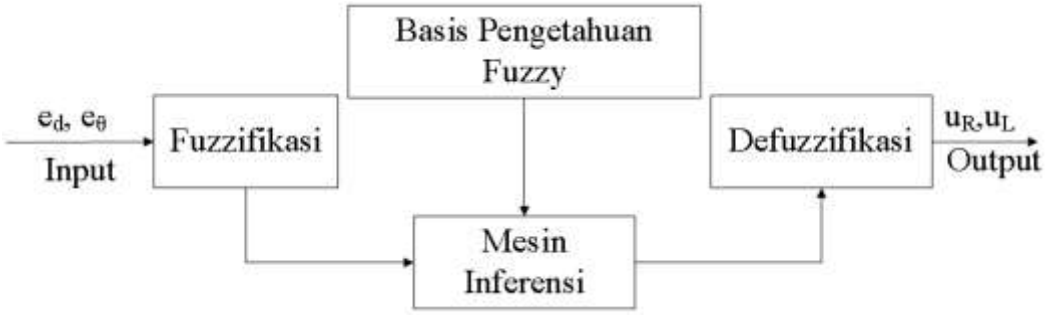
3.2.6 Kontrol Fuzzy

Sistem kontrol yang digunakan dalam penelitian ini adalah Fuzzy, referensi yang digunakan dalam kontrol ini adalah jarak (d) dan sudut (θ). Setiap referensi akan dijumlahkan dengan umpan balik kemudian dikonversikan ke kecepatan motor kanan (v_R) dan motor kiri (v_L) oleh kontrol Fuzzy.

Kontrol Fuzzy diolah menggunakan sistem digital pada Arduino. Arduino akan mendapatkan informasi berupa posisi tujuan robot. Posisi tujuan diberikan ke Arduino berdasarkan data peta dan data Kinect. Berdasarkan posisi tujuan akan di dapatkan nilai jarak dan sudut sebagai input dari kontol Fuzzy. Nilai kecepatan untuk masing-masing motor didapatkan dari aksi kontrol yang ditunjukkan pada Gambar 3.27.



Gambar 3.27 Sistem Kontrol Navigasi Robot.



Gambar 3.28 Blok Diagram Logika Fuzzy.

Sistem logika Fuzzy dapat dilihat pada Gambar 3.28 yang merupakan perancangan blok diagram dari logika Fuzzy. Adapun tahapan penting di dalam perancangan logika Fuzzy pada sistem kecepatan dua motor, di antaranya:

1. Membuat himpunan Fuzzy dari variabel input dan variabel output.

   Variabel input berupa nilai jarak (d) dan sudut (θ),

- Jarak (d) terbagi menjadi 3 himpunan Fuzzy, yaitu : Netral (N), Sedang (M), dan Jauh (F)
- Sudut (θ) terbagi menjadi 3 himpunan Fuzzy, yaitu : tengah (C), Kanan (R), dan Kiri (L)

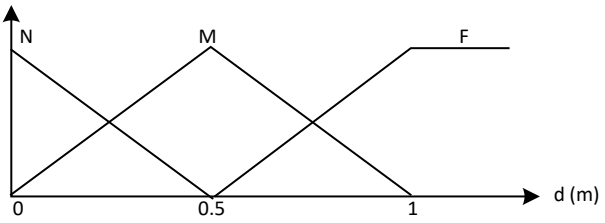
   Variabel output berupa nilai kecepatan motor kanan(v_R) dan motor kiri(v_L)

- Kecepatan motor kanan (v_R) terbagi menjadi 5 himpunan Fuzzy, yaitu : mundur cepat (FB), mundur sedang (MB), lambat (S), maju sedang (MF), maju cepat (FF)
- Kecepatan motor kiri (v_L) terbagi menjadi 5 himpunan Fuzzy, yaitu : mundur cepat (FB), mundur sedang (MB), lambat (S), maju sedang (MF), maju cepat (FF)

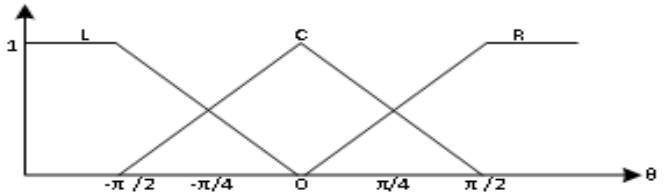
2. Menentukan nilai semesta pembicaraan dari tiap-tiap himpunan Fuzzy

Jarak (d) (m)	: [0 1]
Sudut (ω) (^{0})	: [-180 180]
Kecepatan motor kanan (v_R) (PWM)	: [-255 255]
Kecepatan motor kiri (v_L) (PWM)	: [-255 255]

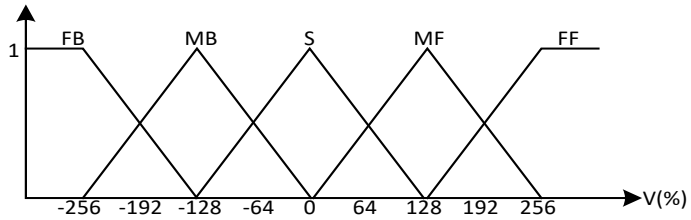
Gambar 3.29, Gambar 3.30, menunjukkan himpunan input jarak dan sudut dan Gambar 3.31, Gambar 3.32 menunjukkan himpunan dari. Aturan logika Fuzzy untuk motor kanan dan kiri ditunjukkan pada Tabel 3.1dan Tabel 3.2



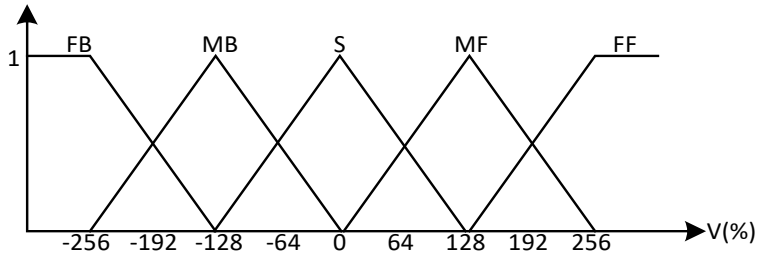
Gambar 3.29 Membership Function Untuk Input Jarak



Gambar 3.30 Membership Function Untuk Input Sudut



Gambar 3.31 Himpunan *Fuzzy* Untuk *Output* Motor Kanan (v_R)



Gambar 3.32 Himpunan *Fuzzy* Untuk *Output* Motor Kiri (v_L)

Tabel 3.1 Perancangan Aturan Dari Kecepatan Motor Kanan

		Distance		
		N	**M**	**F**
Angle	**L**	S	MB	MB
	C	S	FF	FF
	R	S	MF	MF

Tabel 3.2 Perancangan Aturan Dari Kecepatan Motor Kiri

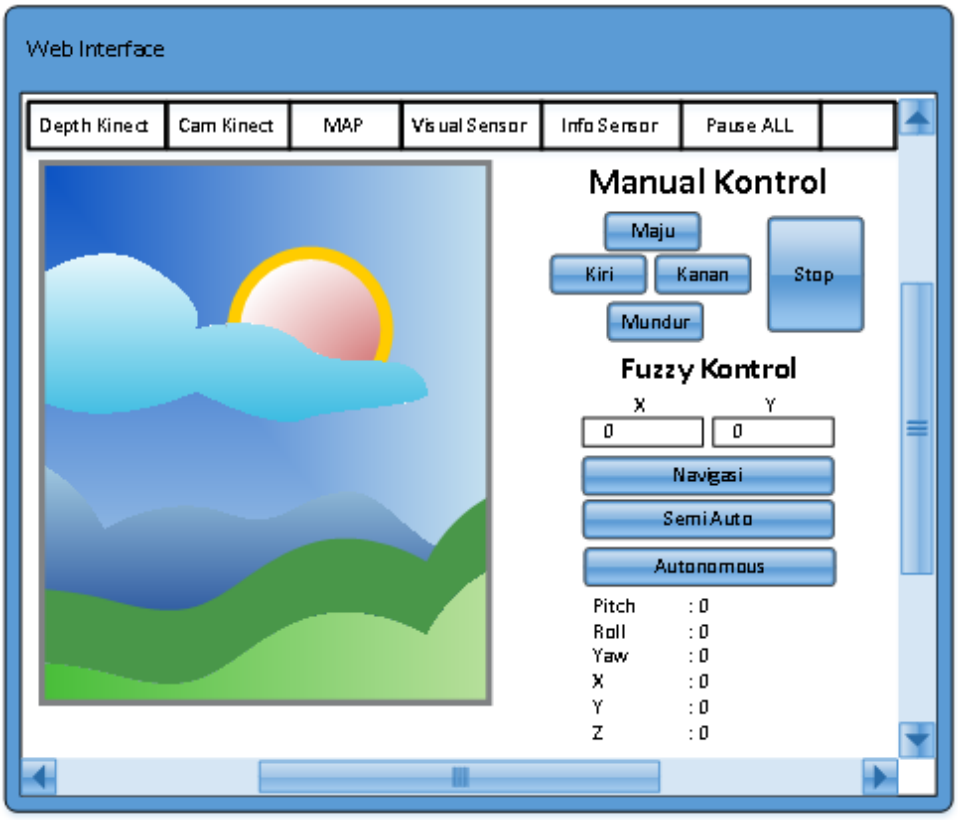
		Distance		
		N	**M**	**F**
Angle	**L**	S	MF	MF
	C	S	FF	FF
	R	S	MB	MB

3.3 Perancangan Perangkat Lunak

Pada tahap ini di rancang pembuatan perangkat lunak untuk mengendalikan robot dan mengambil informasi robot. Perancangan perangkat lunak di mulai dari perancangan *interface* dan pembuatan struktur program. Bahasa pemrograman yang digunakan untuk membuat perangkat lunak adalah bahasa Pyhton 2.7.

3.3.1 Desain interface

Pada desain *interface* digunakan *interface* berbasis web, *interface* web di pilih karena mendukung hampir semua perangkat (Komputer maupun *Handphone*). Desain *interface* diharapkan dapat mempermudah *user* untuk mengendalikan maupun mengambil informasi yang ada pada robot.



Gambar 3.33 Perancangan Desain Interface

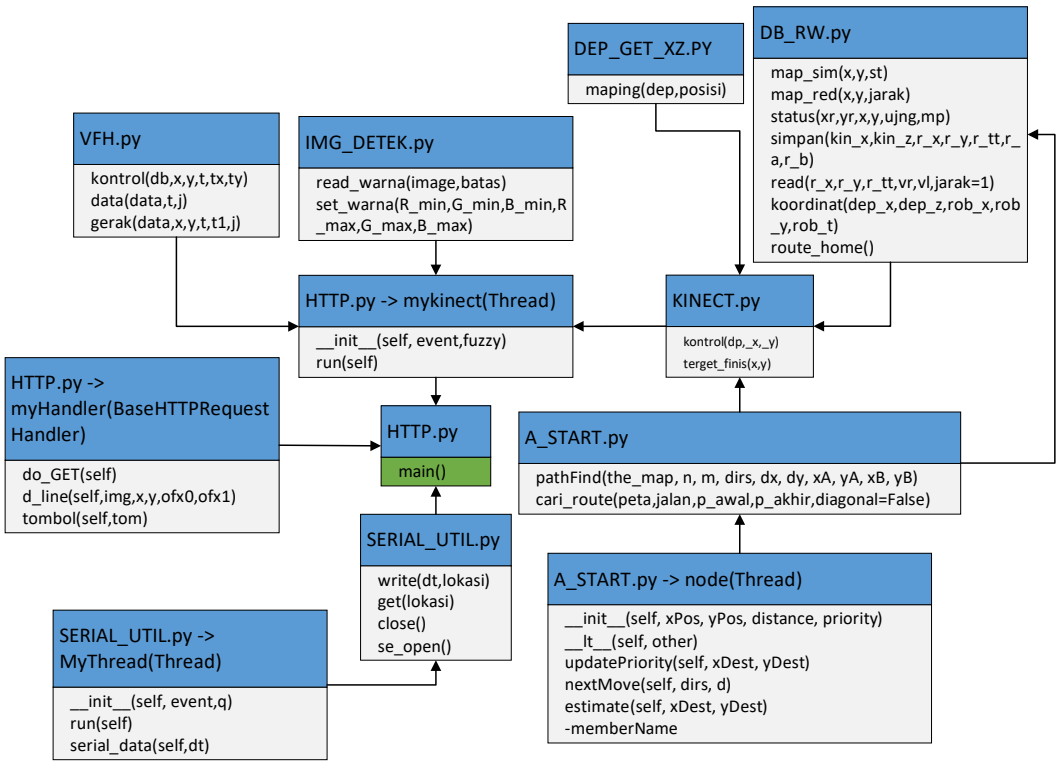
Adapun desain yang dibuat ditunjukkan pada Gambar 3.33. Desain interface dilengkapi dengan beberapa komponen penting, yaitu tiga buah menu (*Home*, Kinect, Sensor, Stop), enam buah tombol (maju, mundur, kanan, kiri, stop, Fuzzy jalan), *image* area, dan label (*Pitch, Roll Yaw*, *X, Y Z*). Adapun fungsi masing-masing komponen adalah

1. **Menu Depth Kinect,** digunakan untuk menampilkan data *depth* Kinect ke dalam *image* area.
2. **Menu Cam Kinect**, digunakan untuk menampilkan data camera Kinect ke dalam *image* area.
3. **Menu MAP**, digunakan untuk menampilkan data peta ke dalam *image* area.
4. **Menua Visaul Sensor**, digunakan untuk menampilkan gambar visual sesuai dengan kondisi sensor (*Pitch, Roll Yaw*).
5. **Menu Info Sensor,** digunakan untuk menampilkan data sensor ke dalam label (*Pitch, Roll Yaw, X, Y Z*)
6. **Menu Pause All,** digunakan untuk menghentikan informasi Kinect dan sensor, ketika menu kinect atau sensor di aktifkan.
7. **Tombol Maju,** digunakan untuk membuat robot bergerak ke arah depan.
8. **Tombol Mundur,** digunakan untuk membuat robot bergerak ke arah belakang.
9. **Tombol Kiri,** digunakan untuk membuat robot bergerak ke arah kiri.
10. **Tombol Kanan,** digunakan untuk membuat robot bergerak ke arah kanan.
11. **Tombol Stop,** digunakan untuk menghentikan pergerakan robot, baik dalam kontrol manual atau otomatis (Fuzzy).
12. **Navigasi**, tombol yang digunakan untuk menggerakkan robot ke suatu lokasi, tanpa melakukan pemetaan dan pendeteksian penghalang
13. **Semi Auto**, tombol yang digunakan untuk menggerakkan robot ke suatu lokasi. Pada mode ini robot melakukan pemetaan, dan menghindari penghalang yang ada di depannya.
14. **Autonomous,** tombol yang digunakan untuk menjalankan robot secara otomatis. Pada mode ini robot dapat menentukan target lokasi secara otomatis, menghindari penghalang yang ada di depannya, melakukan pemetaan, dan robot melakukan pencarian objek berdasarkan warna yang sudah di tentukan sebelumnya.

15. **Image Area,** area yang digunakan untuk menampilkan data *Depth* Kinect, Kamera Kinect, peta area, dan peta rute korban, ke semua data yang ditampilkan pada area ini akan di ubah menjadi gambar terlebih dahulu.
16. **Label,** komponen yang digunakan untuk menampilkan informasi sensor berupa tulisan angka, informasi yang ditampilkan adalah nilai *Pitch*, nilai *Roll*, nilai *Yaw*, nilai *X*, nilai *Y*, nilai *Z*.

3.3.2 Class Struktur

Dalam mempermudah proses pembuatan maupun pengembangan program, dilakukan pengelompokan program berdasarkan fungsinya. Fungsi utama yang ada di dalam program terdiri dari program web *interface*, program pembaca *depth* Kinect, program pengelola data Kinect, program penyimpanan data peta, program pendeteksian objek, dan program pencari rute terpendek.



Gambar 3.34 Diagram Stuktur Class

Setiap struktur *class* terhubung satu sama lainnya, hubungan antar *class* di tunjukkan pada Gambar 3.34. adapun fungsi dari masing-masing *class* adalah

1. **HTTP.py**, merupakan program utama, program ini yang menghubungkan semua *class*. Pada program ini terdapat fungsi yaitu fungsi main()
2. **HTTP.py -> myHandler(BaseHTTPRequestHandler)**, merupakan *class* yang terdapat pada program HTTP.py. *Class* ini berfungsi untuk membentuk dan mengolah data web, pada *class* terdapat tiga fungsi utama yaitu
    a. **do_GET(self)**, merupakan fungsi yang akan dipanggil ketika terdapat permintaan atau pemanggilan URL, data URL akan di kenali dalam metode GET. Ketika ada permintaan maka program akan memeriksa jenis permintaan. Ketika jenis permintaan di kenali maka program akan mengirimkan status sukses (OK 200), dan apabila permintaan tidak di kenali maka program akan mengirimkan status eror (*Not Found* 404).
    b. **d_line(self,img,x,y,ofx0,ofx1)**, merupakan fungsi tambahan, fungsi ini digunakan untuk menggambar dua buah garis ketika terdapat permintaan pengambilan data Kinect, garis merupakan nilai dari sensor kemiringan robot. Fungsi ini akan di panggil pada fungsi do_GET(self). Adapun fungsi dari masing-masing parameter adalah *x* adalah nilai posisi vertikal garis yang akan dibuat (nilai 0 terdapat pada pertengahan gambar), *y* adalah nilai posisi horizontal garis yang akan dibuat (nilai 0 terdapat pada pertengahan gambar), *ofx0* merupakan selisih tambahan pada sumbu x untuk garis pertama,*ofx1* merupakan selisih tambahan pada sumbu x untuk garis kedua.
    c. **Tombol(self,tom)**, fungsi ini digunakan untuk menjalankan permintaan tombol yang di tekan (maju, mundur, kanan, kiri, stop, dan Fuzzy jalan).
3. **HTTP.py → mykinect(Thread)**, merupakan *class* yang mengolah data Kinect. *Class* ini merupakan jenis *Thread* (bekerja di balik layar) sehingga terpisah dengan proses utama. Jenis *thread* di pilih agar tidak mengganggu atau terganggu dengan proses yang lain. Seperti proses saat web *interface* di panggil. Pada class ini terdapat beberapa fungsi di antaranya
    a. **__init__(self, event,fuzzy)**, fungsi ini merupakan parameter awal ketika *class* mykinect(Thread) di buat. Adapun fungsi dari masing-masing

parameter adalah *event* merupakan parameter berjenis *event* yang digunakan untuk menghentikan proses keseluruhan, dan Fuzzy merupakan *event* yang digunakan untuk menghentikan proses menjelajah otomatis.

b. **run(self)**, fungsi ini akan di panggil ketika *class* mykinect(Thread) di jalankan, fungsi ini berisi pengolahan data Kinect mulai dari pembacaan data sampai aksi ke robot.

4. **SERIAL_UTIL.py**, program ini digunakan untuk mengatur komunikasi antara komputer dan Arduino, menyimpan respons dari Arduino. Pada program ini terdapat beberapa fungsi di antaranya
    a. **write(dt,lokasi=0)**, fungsi ini digunakan untuk mengirimkan data ke Arduino. Pada fungsi ini terdapat dua parameter, parameter pertama (dt) merupakan data yang akan di kirim, parameter kedua (lokasi) merupakan alamat penyimpanan dari respons data yang dikirim.
    b. **get(lokasi)**, fungsi ini digunakan untuk mengambil respons dari data yang dikirim sesuai dengan lokasi penyimpanannya. Parameter lokasi digunakan untuk menentukan lokasi penyimpanan yang akan di ambil
    c. **close()**, fungsi ini digunakan untuk menutup komunikasi serial
    d. **se_open()**, fungsi ini digunakan untuk membuka komunikasi serial
5. **SERIAL_UTIL.py →MyThread(Thread)**, merupakan *class* yang mengolah komunikasi serial. *Class* ini merupakan jenis *Thread* (bekerja di balik layar) sehingga terpisah dengan proses utama. Jenis thread di pilih agar tidak mengganggu atau terganggu dengan proses yang lain. *Class* ini selalu memeriksa data yang masuk maupun keluar dari komunikasi serial. Pada *class* ini terdapat beberapa fungsi di antaranya
    a. **__init__(self, event,q)**, fungsi ini merupakan parameter awal ketika class MyThread (Thread) di buat. Adapun fungsi dari masing-masing parameter adalah *event* merupakan parameter berjenis *event* yang digunakan untuk menghentikan proses keseluruhan, dan q merupakan *Queue* atau *dictionary* yang digunakan untuk menjembatani variabel ke proses utama.

b. **run(self)**, fungsi ini akan di panggil ketika *class* MyThread (Thread) di jalankan, fungsi ini berisi pengolahan komunikasi serial seperti mengirim dan menerima data.
c. **serial_data(self,dt)**, fungsi ini merupakan fungsi yang digunakan untuk mengirimkan data, pada fungsi ini mengatur tunda waktu pengiriman untuk menghindari eror dalam komunikasi.

6. **IMG_DETEK.py**, program ini digunakan untuk mendeteksi objek berdasarkan warna. Pada program ini terdapat beberapa fungsi di antaranya
   a. **set_warna(R_min,G_min,B_min,R_max,G_max,B_max)**, fungsi ini digunakan untuk mengatur warna minimal dan maksimal pada objek yang akan di deteksi. Pada fungsi ini terdapat beberapa parameter diantaranya *R_min* = warna merah terendah, *G_min* = warna hijau terendah, *B_min* = warna biru terendah, *R_max* = warna merah tertinggi, *G_max* = warna hijau tertinggi, dan *B_max* = warna biru tertinggi
   b. **read_warna(image,batas)**,Fungsi ini digunakan untuk membaca objek pada gambar sesuai dengan batas warna yang di tentukan pada fungsi sebelumnya. Pada fungsi ini terdapat beberapa parameter di antaranya *image* yang merupakan gambar yang akan di deteksi, batas adalah batasan luas dari area .
7. **KINECT.py**, program ini digunakan untuk mengolah data Kinect. Bentuk pengolahannya di mulai dari pengaturan pembacaan data *depth*, pengaturan penyimpanan data peta, pengaturan pendeteksian jarak pada *databases* peta, dan mengeluarkan mini peta untuk kontrol mobile robot. Pada program ini terdapat beberapa fungsi di antaranya
   a. **kontrol(dp,_x,_y,_theta)**, fungsi ini digunakan untuk mengatur penyimpanan data peta sampai mengeluarkan data mini peta untuk kontrol robot. Adapun fungsi dari masing-masing parameter adalah dp merupakan data *depth* dari sensor Kinect, (*_x* dan *_y*) merupakan posisi robot, dan *_theta* merupakan arah sudut pandang robot.
   b. **target_finis(x,y)**, fungsi ini digunakan untuk memeriksa apakah robot sudah sampai pada posisi yang di tuju atau belum. Parameter *x* dan *y* merupakan posisi robot.

8. **DEP_GET_XZ.py**, program ini digunakan untuk membaca data *depth* yang tersusun dari 2d array di rubah menjadi 1d array. Tujuan konversi dari 2d ke 1d agar mempermudah pengolahan data, sekaligus menghilangkan *noise* yang kecil dari data *depth*. Pada program ini memiliki fungsi yaitu
    a. **maping(dep,posisi=240)**, fungsi ini digunakan untuk mengonversikan array 2d menjadi 1d, sekaligus mengonversi nilai pixel menjadi nilai jarak dalam satuan meter. Parameter *dep* merupakan data *depth* dalam array 2d, dan posisi merupakan nilai tengah yang merupakan posisi dinding.
9. **DB_RW.py**, program ini digunakan untuk mengatur data peta, penyimpanan data peta, dan mengeluarkan data peta. Pada program ini terdapat beberapa fungsi diantaranya
    a. **map_sim(x,y,st)**, fungsi ini digunakan untuk menyimpan data ke database peta. Parameter yang digunakan (*x,y*) merupakan posisi robot, dan *st* merupakan status yang akan disimpan (1 merupakan nilai untuk data yang tidak diketahui, 2 merupakan nilai untuk jalan, dan 3 merupakan nilai untuk dinding).
    b. **map_red(x,y,jarak)**, fungsi ini digunakan untuk mengambil atau membaca data peta. Adapun fungsi dari masing-masing parameter adalah (*x,y*) merupakan posisi peta yang akan dibaca, jarak merupakan luas jari-jari dari peta yang akan di baca.
    c. **read(r_x,r_y,r_tt,jarak=1)**, fungsi ini merupakan pengembangan dari fungsi map_red, perbedaan antara keduanya yaitu, pada fungsi ini di lengkapi parameter sudut. Parameter sudut digunakan untuk memutar data peta. Sehingga posisi atas pada peta yang dibaca merupakan arah yang dipandang oleh robot. Nilai parameter (*r_x,r_y*) merupakan lokasi peta yang akan dibaca, nilai *r_tt* merupa sudut putar peta, jarak merupakan luas jari-jari dari peta yang akan dibaca.
    d. **status(xr,yr,x,y,ujng)**, fungsi ini digunakan untuk menentukan status apakah tergolong sebagai jalan, dinding, atau data yang tidak diketahui. Adapun fungsi dari masing-masing parameter adalah (*xr,yr*) merupakan posisi koordinat robot, (*x,y*) merupakan posisi koordinat dari data *depth*,

ujung merupakan nilai logika (*boolean*) yang menandakan posisi awal data *depth*.

e. **simpan(kin_x,kin_z,r_x,r_y,r_tt,r_a,r_b)**, fungsi ini merupakan fungsi yang digunakan untuk mengatur penyimpanan data peta. Adapun fungsi parameter yang digunakan adalah (*kin_x, kin_z*) merupakan array 1d yang diambil dari array 2d menggunakan program DEP_GET_XZ.py, (*r_x,r_y,r_tt,r_a,r_b*) merupakan posisi koordinat robot beserta nilai *yaw* (*r_tt*), *pitch* (*r_a*), dan *roll* (*r_b*).

f. **koordinat(dep_x,dep_z,rob_x,rob_y,rob_t)**, fungsi ini digunakan untuk mengonversi koordinat *depth* ke dalam koordinat yang sesungguhnya berdasarkan posisi robot. Adapun parameter yang digunakan adalah (*dep_x, dep_z*) merupakan nilai dari koordinat data depth kinect, (*rob_x, rob_y, rob_t*) merupakan posisi koordinat robot beserta sudut *yaw*.

g. **route_home()**, fungsi ini digunakan untuk mengambil data peta yang dilengkapi dengan jalur terpendek untuk mencapai posisi awal (*home position*).

10. **A_START.py**, program ini digunakan untuk mencari rute terpendek dari satu posisi ke posisi tertentu melalui data peta yang sudah di buat. . Pada program ini terdapat beberapa fungsi di antaranya

    a. **pathFind(the_map, n, m, dirs, dx, dy, xA, yA, xB, yB)**, fungsi ini merupakan fungsi yang digunakan untuk menemukan rute peta pada lokasi tertentu. Fungsi masing-masing parameter adalah *the_map* merupakan data peta yang sudah di konversikan menjadi 2 status (jalan dan penghalang), (*n, m*) merupakan luas area dari peta, *dirs* merupakan angka dari sebuah arah yang memungkinkan (jika rute yang di inginkan tidak terdapat diagonal maka arah yang memungkinkan ada 4 (atas, bawah, kanan, kiri), jika di izinkan terdapat data diagonal maka arah yang memungkinkan ada 8), (*xA, yA*) merupakan koordinat posisi awal (titik start), (*xB, yB*) merupakan koordinat posisi tujuan (titik target).

    b. **Cari_router(peta,jalan,p_awal,p_akhir,diagonal=False)**, fungsi ini merupakan penyederhanaan dari fungsi pathFind, pada fungsi ini tidak

perlu melakukan konversi data peta menjadi 2 status (jalan dan penghalang). Adapun fungsi masing-masing parameter yang digunakan adalah peta merupakan data peta, jalan merupakan nilai dari peta yang menunjukkan jalan, p_*awal* merupakan array dari posisi koordinat awal, p_*akhir* merupakan array dari posisi koordinat tujuan, dan diagonal merupakan nilai logika (*boolean*) yang menunjukkan apakah diizinkan menggunakan pembacaan secara diagonal atau tidak.

11. **A_START.py→node()**, merupakan class yang digunakan untuk mengolah setiap *node* yang terbuka dari algoritme A*. Pada *class* ini terdapat beberapa fungsi di antaranya
    a. **__init__(self, xPos, yPos, distance, priority)**, fungsi ini merupakan parameter awal ketika *class* **node** () di buat. Adapun fungsi dari masing-masing parameter adalah (*xPos*, *yPos*) merupakan parameter posisi koordinat sekarang (jika untuk pertama kali maka merupakan koordinat titik start), *distance* merupakan total jarak yang sudah di kunjungi untuk setiap *node*, *priority* merupakan penjumlahan *distance* (jarak) dengan perkiraan jarak yang tersisa.
    b. **__lt__(self, other)**, fungsi ini merupakan fungsi yang akan dipanggil ketika terdapat operator <, contoh ketika pada program di tuliskan *x*<*y* maka fungsi yang akan dipanggil adalah x.__lt__(y).
    c. **updatePriority(self, xDest, yDest)**, fungsi ini digunakan untuk memperbaharui nilai dari *priority*. Adapun fungsi dari masing-masing parameter adalah (*xDest*, *yDest*) merupakan koordinat posisi tujuan.
    d. **nextMove(self, dirs, d)**, fungsi ini akan menghasilkan prioritas yang lebih baik ketika berjalan lurus (bukan diagonal). Adapun fungsi dari masing-masing parameter adalah *dirs* merupakan angka dari sebuah arah yang memungkinkan, d merupakan arah dari pergerakan.
    e. **estimate(self, xDest, yDest)**, fungsi ini digunakan untuk menghitung sisa jarak ke tujuan. Adapun fungsi dari masing-masing parameter adalah (*xDest*, *yDest*) merupakan koordinat posisi tujuan.

12. **VFH.py**, program ini di gunakan untuk menentukan arah gerakan berdasarkan data histogram. Data histogram di hasilkan dari perhitungan mini peta pada program KINECT.py.
    a. **kontrol(db,x,y,t,tx,ty)**, merupakan fungsi yang memberikan aksi kontrol berupa posisi tujuan. Adapun fungsi dari masing-masing parameter adalah *db* merupakan data mini peta yang dihasilkan dari program KINECT.py, (*x*,*y*,*t*) merupakan posisi koordinat robot beserta sudut *yaw* robot, (*tx*,*ty*) merupakan posisi koordinat tujuan.
    b. **data(data,t,j)**, fungsi ini digunakan untuk menghasilkan data histogram berdasarkan data mini peta. Adapun fungsi dari masing-masing parameter adalah data merupakan data mini peta, t merupakan sudut antara posisi robot dengan sudut tujuan, j merupakan jarak dari posisi sekarang ke posisi tujuan.
    c. **gerak(data,x,y,t,t1,j)**, fungsi ini merupakan fungsi yang menghasilkan aksi gerakan berdasarkan data histogram. Adapun fungsi dari masing-masing parameter adalah data merupakan data histogram yang dihasilkan pada fungsi sebelumnya, (*x*,*y*,*t*) merupakan posisi koordinat robot beserta sudut *yaw* robot, t1 merupakan sudut antara posisi robot dengan sudut tujuan, j merupakan jarak dari posisi sekarang ke posisi tujuan.

BAB 4
HASIL DAN PEMBAHASAN

Pengujian mobile robot untuk pemetaan dalam proses evakuasi, diawali dengan pengujian navigasi, pengujian pemetaan, pengujian deteksi objek, dan pengujian rute objek. Teknik pengujian di lakukan dengan menjalankan mobile robot di suatu ruangan, kemudian mobile robot akan bergerak mengikuti fitur jalan. Ketika objek di deteksi mobile robot akan memberikan *feedback* berupa lokasi objek dengan rute terpendeknya.

4.1 Keterbatasan penelitian

Dalam penelitian ini, terdapat beberapa keterbatasan yang mempengaruhi kondisi dari penelitian yang dilakukan. Adapun keterbatasan tersebut adalah:

1. Penelitian ini di fokuskan pada teknik pembuatan peta dan menggambarkan rute dari sebuah objek (yang dimisalkan sebagai korban), sehingga tidak memperhatikan desain robot yang mampu bertahan pada kondisi extrim.
2. Sistem pembacaan kondisi sekitar tidak mampu membedakan benda tipis dengan pixel eror yang ada pada sensor depth Kinect. Sehingga di perlukan media uji untuk menutupi benda-benda tipis yang ada di sekitar.
3. Penelitian ini hanya memanfaatkan pendeteksian warna sebagai pendeteksi objek, dan tidak memperhitungkan eror jarak minimal terhap objek. Sehingga terdapat beberapa objek yang tidak di kenali dengan baik.

4.2 Hasil Perancangan Mobile Robot

Pada tahap ini dilakukan perakitan desain mobile robot yang telah di rancang pada bab sebelumnya. Pengujian dilakukan untuk menunjukkan bahwa hasil desain setiap komponen memiliki kecocokan dengan komponen lainya. Dan hasil dari perakitan menunjukkan bahwa ukuran setiap bagian sudah sesuai dengan bagian yang lain. Hasil dari perakitan dapat di lihat pada Gambar 4.1.

Gambar 4.1 Hasil Perakitan Mobile Robot

Dari hasil perakitan dapat terdapat beberapa selisih jarak, selisih jarak terjadi pada bagian yang berhubungan dengan roda. Dari hasil pengukuran manual didapatkan keliling roda sebesar 0.65 m, dan jarak antar roda 0.24 m. dari kedua nilai tersebut maka kenematik robot dapat dilakukan.

4.3 Pengujian Web *Interface*

Web *interface* berperan penting dalam mengendalikan mobile robot. Web *interface* diharapkan dapat memberikan informasi semua sensor dan memberikan kontrol ke mobile robot. Bentuk hasil desain web *interface* di tunjukkan pada Gambar 4.2 - Gambar 4.5.

Pengujian web *interface* dilakukan untuk memastikan semua fasilitas bekerja dengan benar, dan dapat menampilkan beberapa informasi pending. Gambar 4.2 di ujikan untuk menampilkan *Depth* Kinect, dari pengujian ini dapat terlihat *Depth* Kinect dapat ditampilkan dengan baik. Gambar 4.3 di ujikan untuk menampilkan kamera Kinect, dari pengujian ini dapat terlihat kamera Kinect dapat ditampilkan dengan baik. Gambar 4.4 di ujikan untuk menampilkan data map, pada pengujian ini terlihat data berwarna hitam, dikarenakan belum ada pengujian data peta. Sehingga data peta masih kosong. Gambar 4.5 di ujikan tampilan visual untuk memperlihatkan posisi robot, dari hasil pengujian dapat terlihat visual sensor dapat di tampilkan dengan baik.

Gambar 4.2 Bentuk Web *Interface* Dengan Depth Kinect

Gambar 4.3 Bentuk Web *Interface* Dengan Camera Kinect

Gambar 4.4 Bentuk Web *Interface* Dengan Data Peta

Gambar 4.5 Bentuk Web *Interface* Dengan Visual Sensor

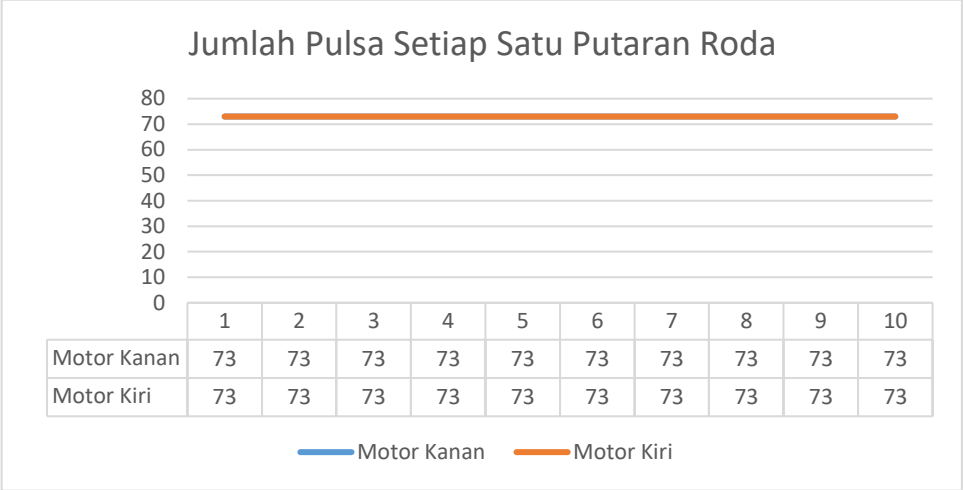
4.4 Pengujian Rotary dan Kinematik Robot

Pada pengujian ini dilakukan untuk menguji apakah sensor kecepatan, dan posisi dapat bekerja dengan baik. Sebelum memperhitungkan posisi terlebih dahulu dilakukan pengujian jumlah pulsa setiap putaran roda. Dari hasil percobaan dapat dilihat hasil yang linier yang ditunjukkan pada Gambar 4.6.

Setelah jumlah pulsa diketahui maka kinematika robot dapat di hitung, perhitungan juga membutuhkan data keliling roda dan jarak antar roda (didapatkan pada bab sebelumnya Hasil Perancangan Mobile Robot). Hasil dari kinematika robot dapat dilihat pada Tabel 4.1. Tabel 4.1 menunjukkan tingkat akurasi dari hasil pembacaan sensor dengan hasil pengukuran manual. Dari beberapa percobaan total eror adalah 0.4% dengan jarak yang ditempuh 16.91 m

Tabel 4.1 Hasil Pengujian Kinematika Robot

No	**Data Sensor (m)**	**Data Asli (m)**	**Selisih (m)**	**Eror (%)**
1	1.28	1.27	0.01	0.79%
2	1.65	1.65	0	0.00%
3	1.8	1.83	0.03	1.64%
4	1.25	1.26	0.01	0.79%
5	1.39	1.4	0.01	0.71%
6	1.65	1.67	0.02	1.20%
7	1.12	1.13	0.01	0.88%
8	1.7	1.71	0.01	0.58%
9	2.68	2.68	0	0.00%
10	2.32	2.31	0.01	0.43%
Rata-rata			**0.011**	**0.70%**
Total	**16.84**	**16.91**	**0.07**	**0.41%**

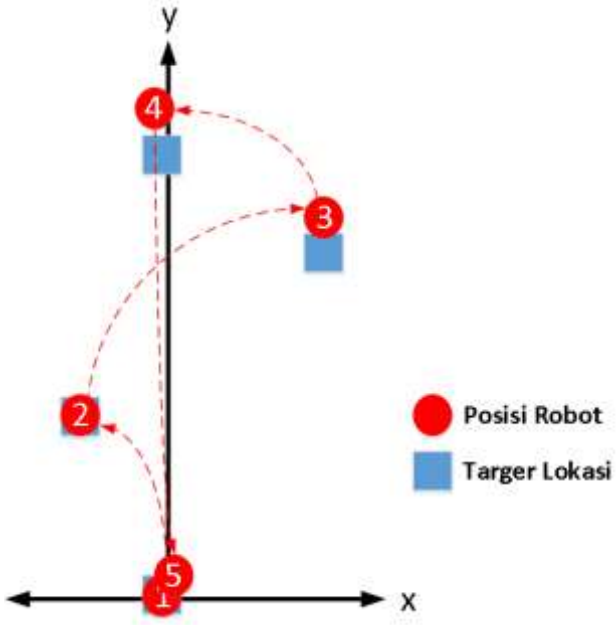


	1	2	3	4	5	6	7	8	9	10
Motor Kanan	73	73	73	73	73	73	73	73	73	73
Motor Kiri	73	73	73	73	73	73	73	73	73	73

Gambar 4.6 Hasil Pengujian Jumlah Pulsa Setiap Satu Putaran Roda

4.5 Hasil Pengujian Navigasi, Poin ke Poin

Pada pengujian ini dilakukan pengujian kemampuan kontrol Fuzzy. Sistem pengujian dilakukan dengan menentukan target lokasi, dan menghitung selisih jarak antara target dengan posisi robot. Data hasil pengujian dapat di lihat pada Tabel 4.2. berdasarkan data Tabel 4.2, didapatkan rata-rata kesalahan adalah 2.33%, adapun ilustrasi posisi pengujian ditunjukkan pada Gambar 4.7.



Gambar 4.7 Ilustrasi Hasil Posisi Pengujian Navigasi

Tabel 4.2 Data Hasil Percobaan Navigasi

No.	Target Lokasi		Posisi Robot		Posisi Sensor		Eror
	X	Y	X	Y	X	Y	
1	0	0	0	0	0	0	0.00%
2	-0.32	2.7	-0.34	2.78	-0.33	2.72	2.92%
3	1.2	3.9	1.1	4.1	1.12	3.91	2.88%
4	0	4.4	-0.1	4.65	0.08	4.41	13.38%
5	0	0	0.1	0.1	0.05	0.07	2.25%

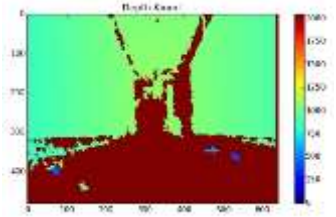
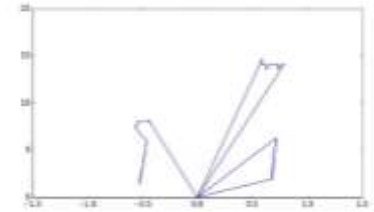
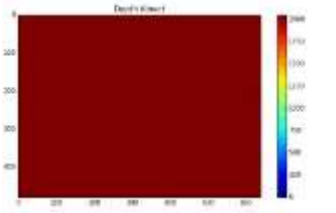
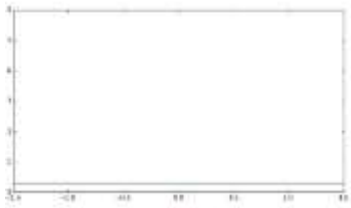
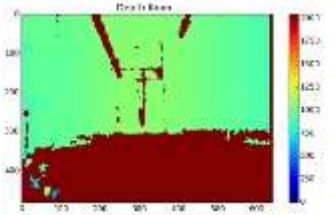
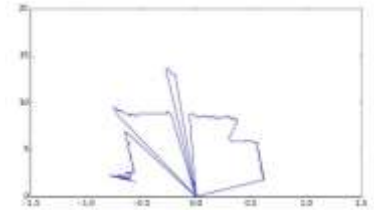
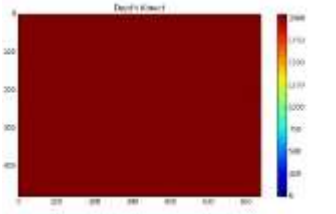
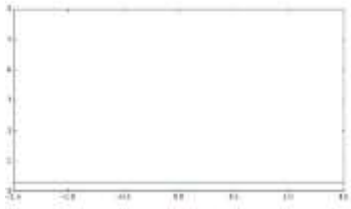
4.6 Pengujian Sensor Kinect

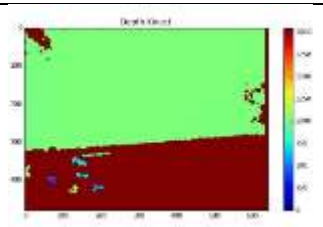
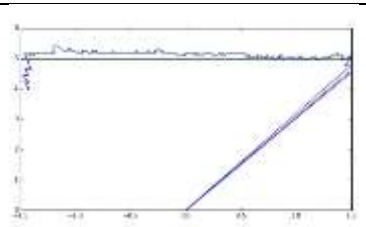
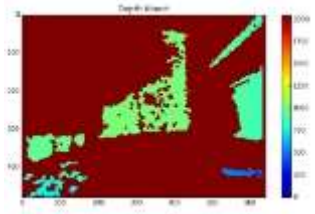
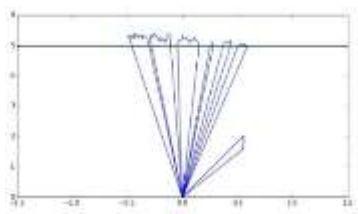
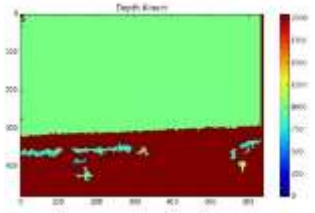
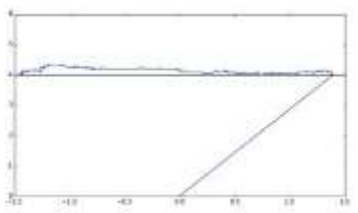
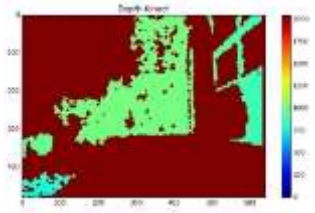
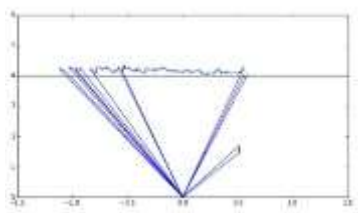
Pada pengujian ini, dilakukan pengujian terhadap kemampuan sensor kinect. Pengujian dilakukan pada 2 tempat yang berbeda yaitu ruang A dan B. Ruang A merupakan ruangan yang tidak terkena sinar matahari dengan intensitas cahaya 3.4 lux dan Ruang B merupakan ruangan yang terkena sinar matahari dengan intensitas cahaya 980 lux. Pengujian juga dilakukan dengan mengukur ketelitian pembacaan jarak sensor Kinect berdasarkan jarak sebenarnya. Hasil pengujian dapat dilihat pada Tabel 4.3.

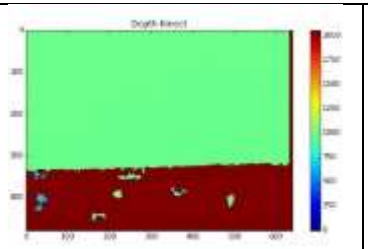
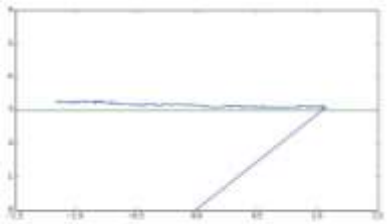
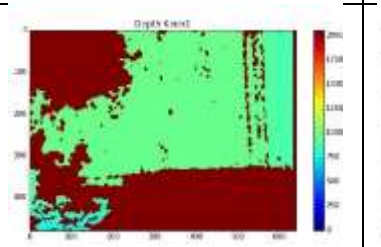
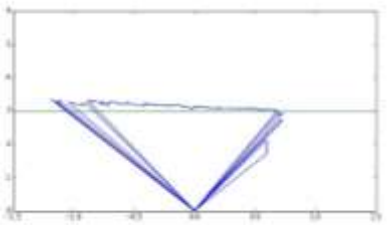
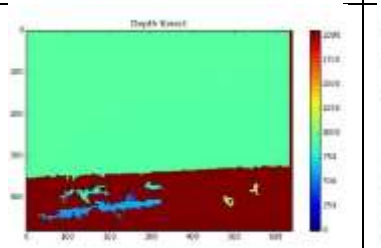
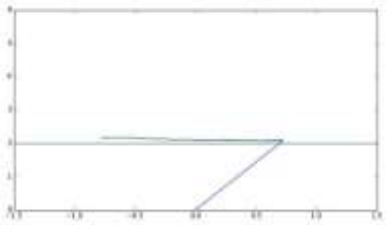
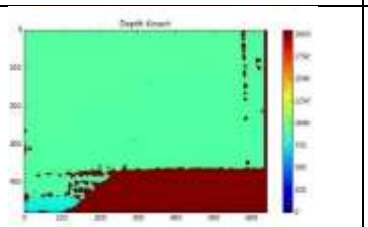
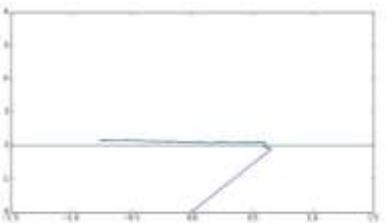
Berdasarkan hasil pengujian, Ruang A dan B, menunjukkan ruang B memiliki eror pixel yang lebih besar dari pada Ruang A, hal ini di sebabkan karena panjang gelombang cahaya IR matahari dapat bercampur dengan cahaya IR pada Kinect sehingga terjadi kesalahan dalam pembacaan. Eror pixel terbesar adalah 76.63% pada ruang B dengan jarak pengukuran 5m. Untuk data lebih dari 8m memiliki eror yang sangat tinggi hingga pencapai 100% pada ruangan B.

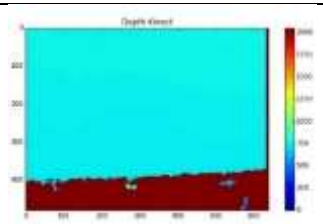
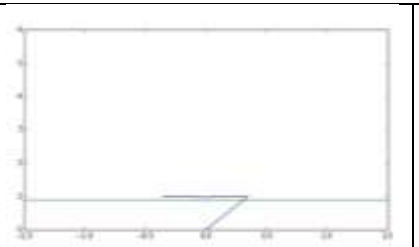
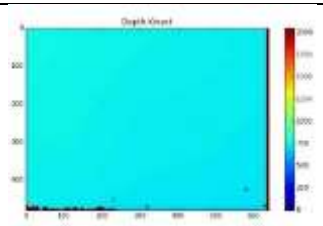
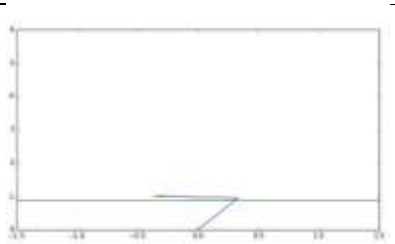
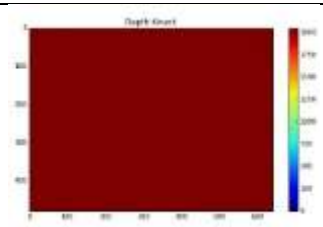
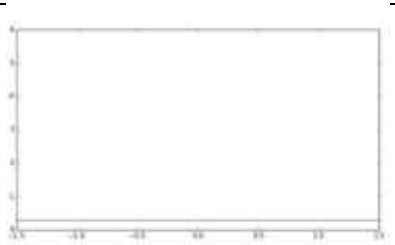
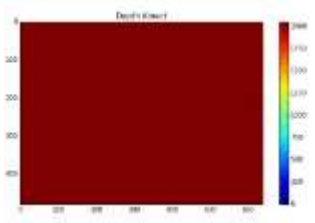
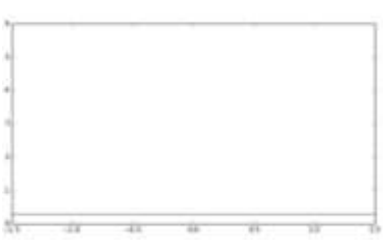
Berdasarkan hasil pengujian, jarak yang memiliki eror yang dapat di toleransi yaitu pada pengukuran di bawah sama dengan 3m, dan jarak yang tidak memiliki eror kecil terjadi pada pengukuran di bawah sama dengan 2m. Oleh sebab itu digunakan nilai 2 meter sebagai nilai yang akan di masukkan ke dalam peta sebagai jalan maupun dinding, sedangkan nilai di antara 2-3 di kategorikan sebagai data yang tidak diketahui.

Tabel 4.3 Data Hasil Pengujian Sensor Kinect

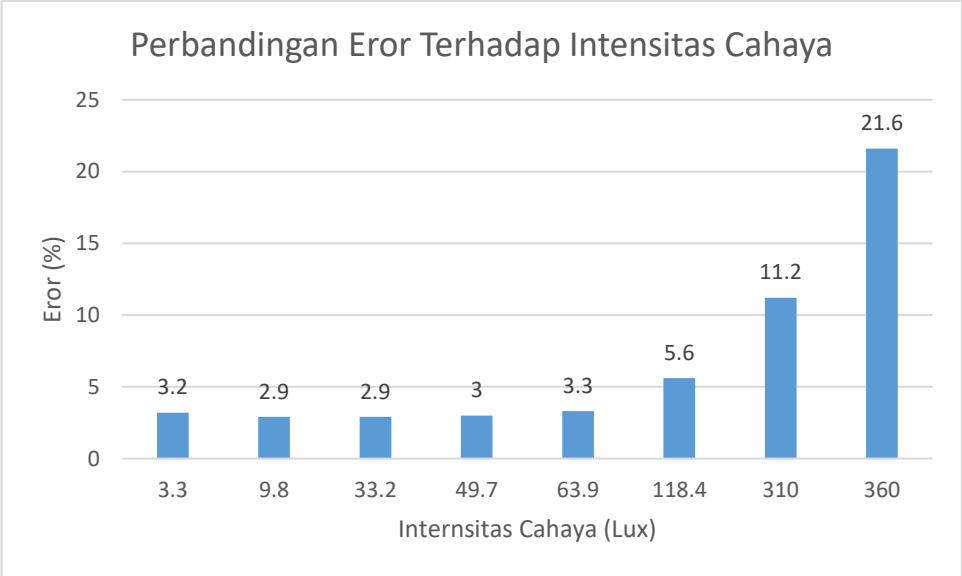
No.	Rungan	Depth	Kurva	Jarak asli	Jarak Kinect	Error jarak	Error Pixel
1	A			15	14.31	4.51%	80%
2	B			10	0	100%	100%
3	A			8	9.52	15.97%	4.12%
4	B			8	0	100%	100%

5	A			5	5.17	3.28%	3.04%
6	B			5	5.32	6.02%	76.63%
7	A			4	4.18	4.31%	0.00%
8	B			4	4.14	3.38%	31.16%

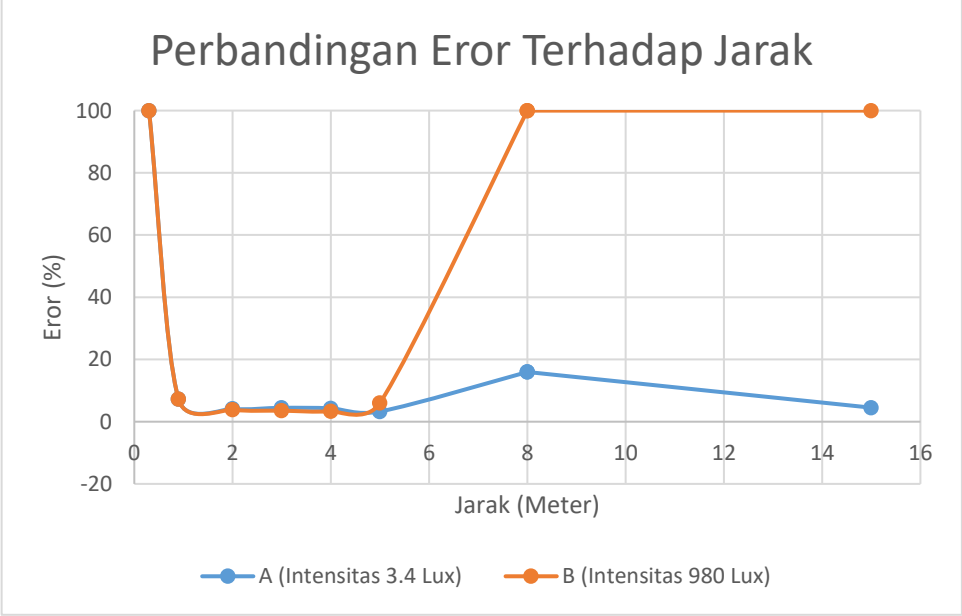
9	A			3	3.14	4.46%	0.00%
10	B			3	3.11	3.54%	10.14%
11	A			2	2.09	4.31%	0.00%
12	B			2	2.08	3.85%	0.00%

13	A			0.9	0.97	7.22%	0.00%
14	B			0.9	0.97	7.22%	0.00%
15	A			0.3	0	100%	100%
16	B			0.3	0	100%	100%

Berdasarkan hasil percobaan, pembacaan sensor Kinect memiliki kondisi eror. Kondisi eror di pengaruhi oleh intensitas cahaya, semakin besar intensitas cahaya eror piksel dan eror pembacaan jakak akan semakin besar. Berdasarkan intensitas cahaya, data eror di gambarkan pada Gambar 4.8 . Sedangkan dampak eror terhadap pembacaan jarak ditunjukkan pada Gambar 4.9.



Gambar 4.8 Perbandingan Eror Terhadap Intensitas Cahaya

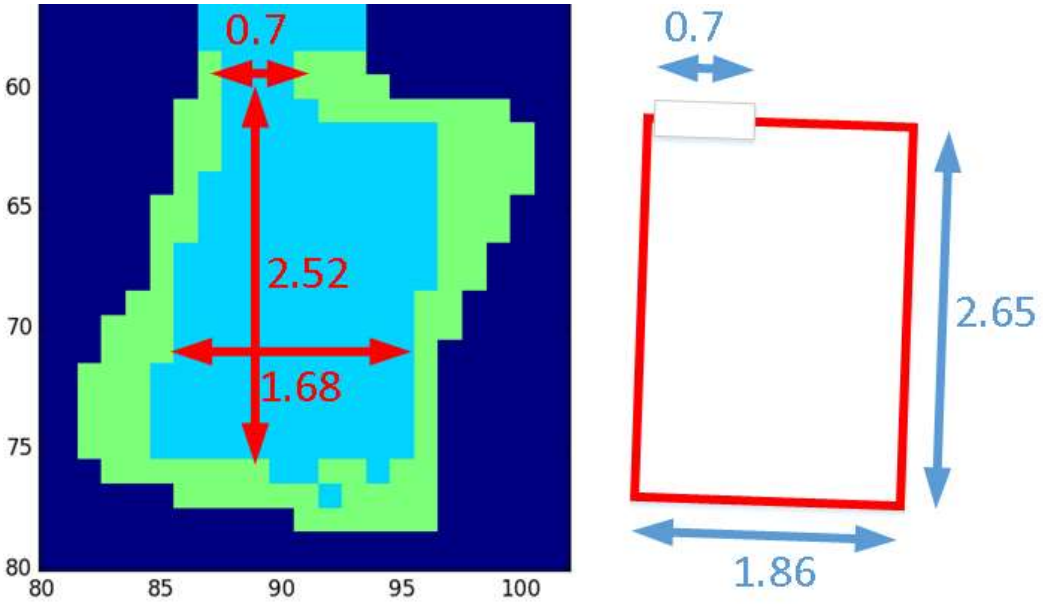


Gambar 4.9 Perbandingan Eror Terhadap Jarak Dengan Variasi Intensitas Cahaya

4.7 Pengujian Akurasi Peta

Pada pengujian ini dilakukan pengujian untuk melihat tingkat akurasi pembuatan peta. Pengujian dilakukan pada ruangan 1.86 x 2.65 m^2, robot gerak memutar untuk melakukan pemetaan. Dalam menentukan skala, digunakan tiga sekala yaitu ¼ dari ukuran robot, ½ dari ukuran robot, dan sesuai ukuran robot. Dari hasil pengujian Tabel 4.4 dipilih tingkat eror dan waktu proses terbaik, dari hasil pengujian digunakan sekala 0.14 m yang memiliki waktu proses 1.3 detik, dan eror 16.43%.

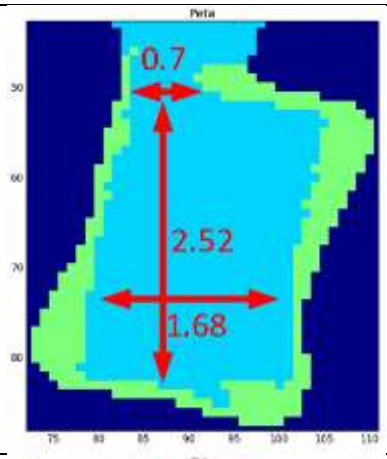
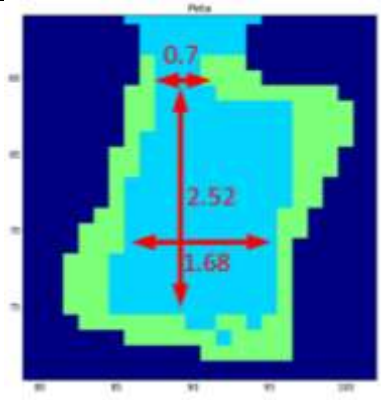
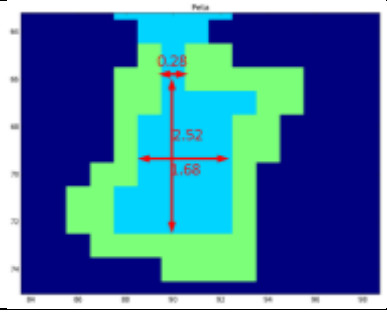
Perbandingan hasil percobaan dengan denah sesungguhnya dapat dilihat pada Gambar 4.10, dari hasil pengujian terdapat beberapa eror, eror di ukur berdasarkan luas area, di mana luas area sesungguhnya adalah 1.86x2.65=4.93 m^2 dan luas area yang terukur pada peta 1.68x2.52=4.23 m^2, sehingga eror pembuatan peta adalah 16.43 %.



(a) (b)

Gambar 4.10 Hasil Pengujian Akurasi Peta (a) Hasil Pembuatan Peta, (b) Ukuran Peta Sesungguhnya

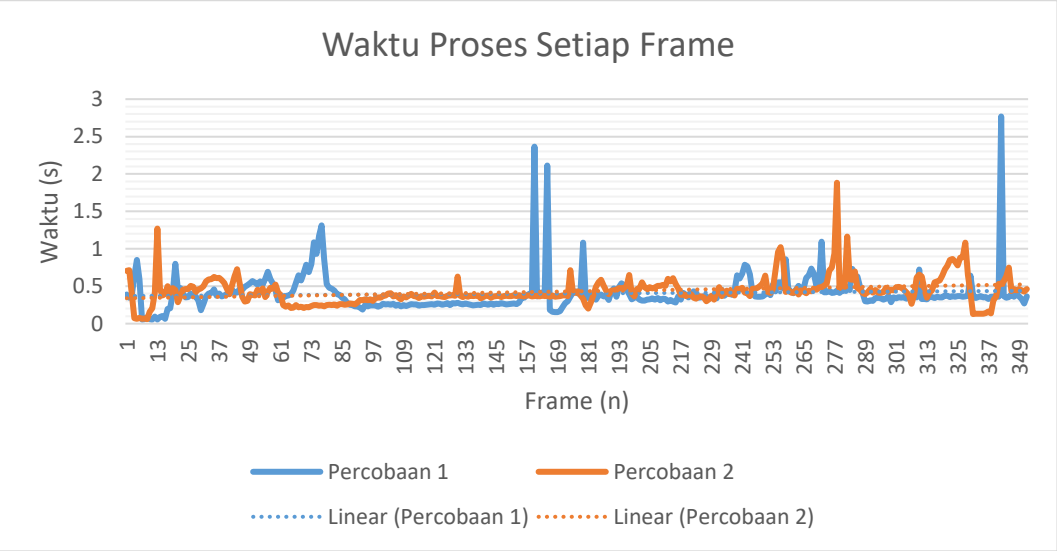
Tabel 4.4Data Hasil Pengujian Peta

No.	Skala/Kotak (m)	Peta	Waktu (detik)	Eror
1	0.07		1.37	16.43 %.
2	0.14		1.30	16.43 %.
3	0.28		1.27	20.3 %.

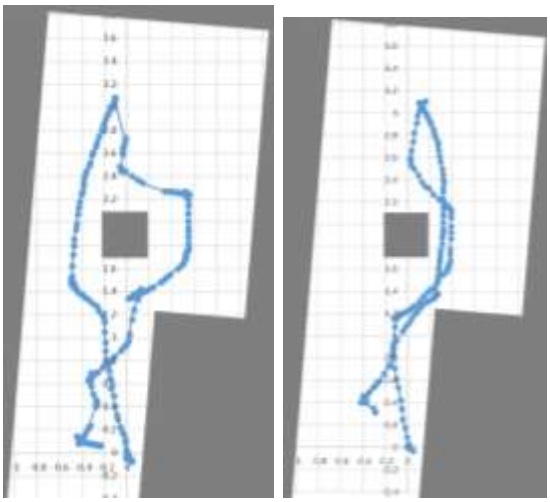
Berdasarkan pengujian data peta, dan sensor Kinect, dapat di tentukan nilai toleransi gerak robot, nilai toleransi bertujuan untuk menghindari terjadinya tabrakan akibat kesalahan pemberian koordinat. Nilai toleransi berasal dari tingkat eror sensor Kinect dengan eror rata-rata eror jarak 0.197 m, karena skala peta adalah 0.14 maka nilai toleransi di bulatkan ke atas menjadi 0.28.

4.8 Pengujian Menghindari Penghalang

Pengujian dalam mengenali penghalang dilakukan dengan memberi penghalang pada jalan Mobile robot. Kemampuan mengingat posisi penghalang di ujikan pada tahap ini. Hasil pergerakan dalam pengujian di tunjukkan pada Gambar 4.12. Hasil pengujian menunjukkan Mobile robot mampu menghindari penghalang dengan baik.



Gambar 4.11 Grafik Waktu Proses Setiap *Frame*

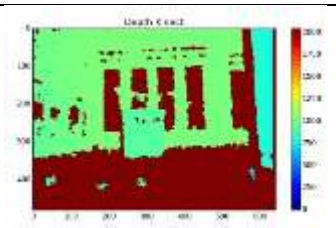
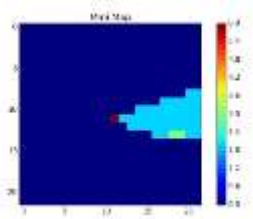
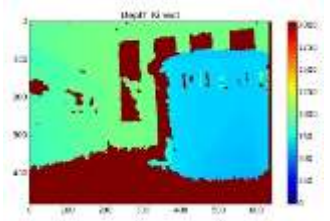
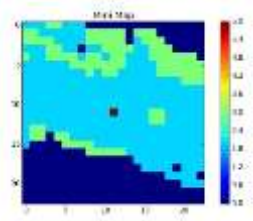
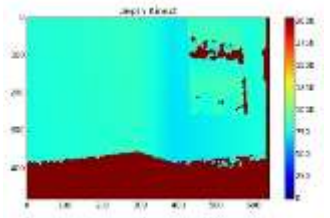
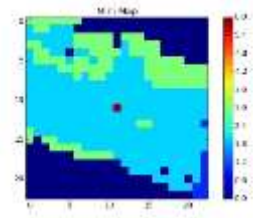
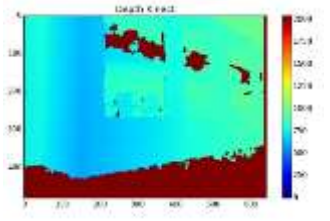
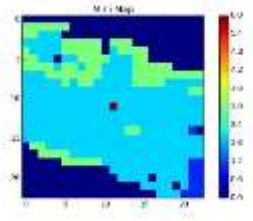


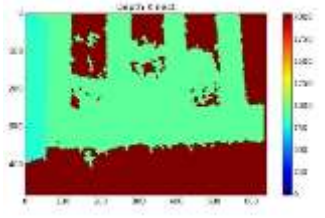
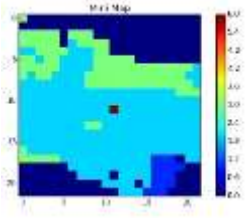
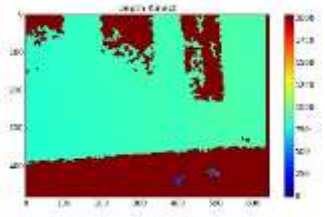
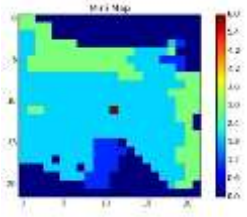
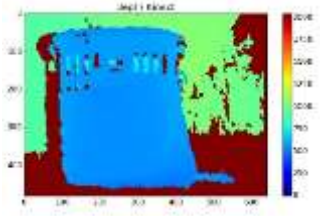
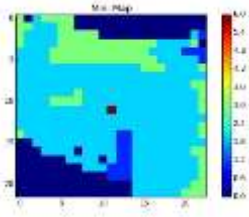
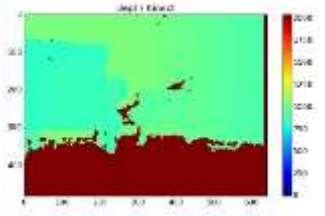
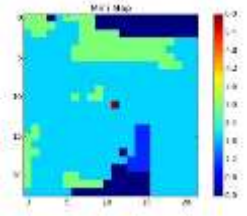
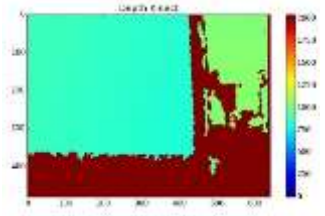
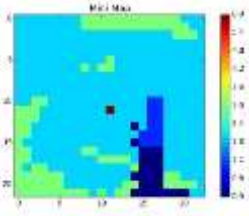
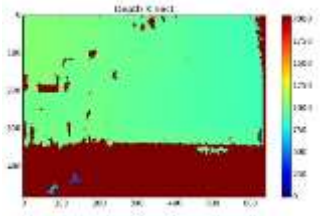
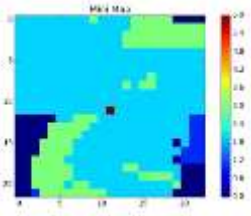
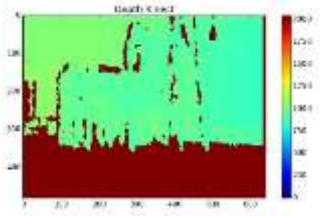
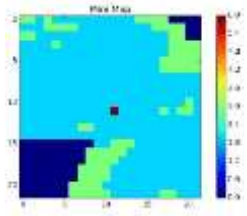
(a) (b)

Gambar 4.12 Rekaman Posisi Sensor Ketika Menghindari Penghalang (a) Percobaan 1, (b) Percobaan 2

Pengujian terhadap waktu proses di lakukan untuk melihat apakah mini komputer yang digunakan mampu menangani kontrol dari mobile robot. Waktu yang tercatat merupakan waktu proses keseluruhan. Waktu proses di mulai dari pengambilan data Kinect, rekonstruksi data Kinect, pemetaan, dan deteksi objek. Grafik waktu proses di tunjukkan pada Gambar 4.11. Tabel 4.5 menunjukkan data yang diproses beserta mini peta yang digunakan untuk navigasi. Rata-rata waktu proses untuk 353 *frame* adalah 430.73 ms. Rata-rata waktu didapatkan ketika sisa baterai sekitar 60%.

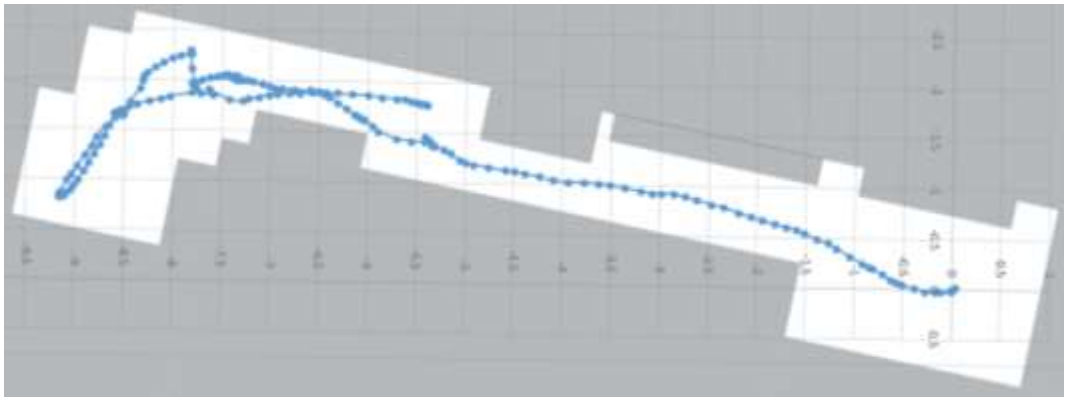
Tabel 4.5 Data Pembacaan Navigasi Pada Percobaan Menghindari Penghalang

No	Frame	Depth Kinect	Mini Map	Waktu Proses (ms)
1	1			347.05
2	54			596.92
3	62			366.94
4	70			788.85

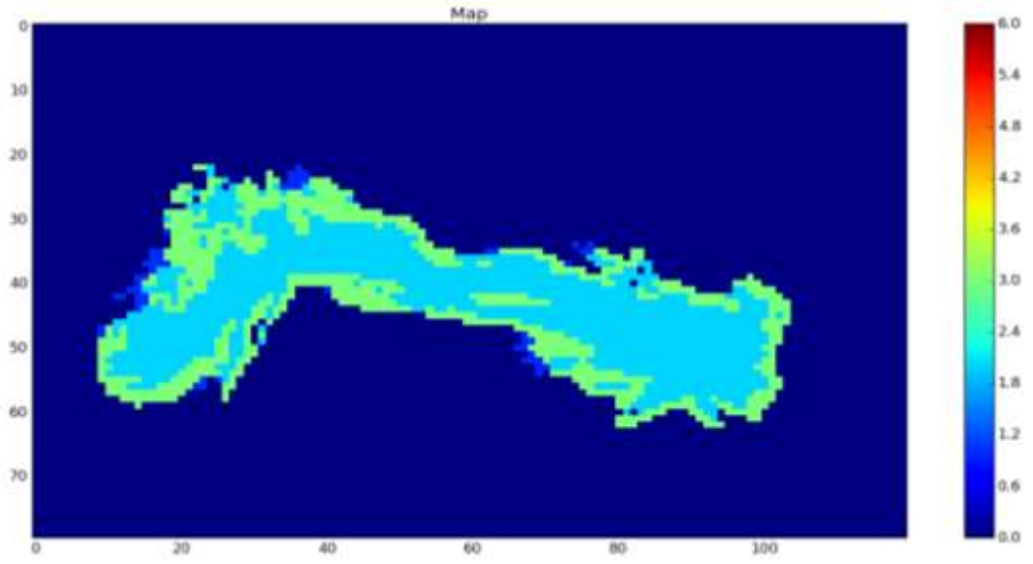
5	86			276.92
6	133			262.92
7	162			411.87
8	178			440.73
9	199			345.04
10	222			393.50
11	249			388.85

4.9 Pengujian Autonomous

Kemampuan dalam menentukan target rute berikutnya di ujikan pada tahap ini. Kemampuan ini digunakan untuk mengetahui apakah mobile robot dapat berjalan tanpa harus di kenalkan terlebih dahulu. Pengujian dilakukan pada sebuah rumah dengan luas 12x5 meter dengan beberapa penghalang seperti kursi, lemari, dan meja. Rute perjalanan mobile robot di tunjukkan pada Gambar 4.13, peta yang dihasilkan ditunjukkan pada Gambar 4.14.



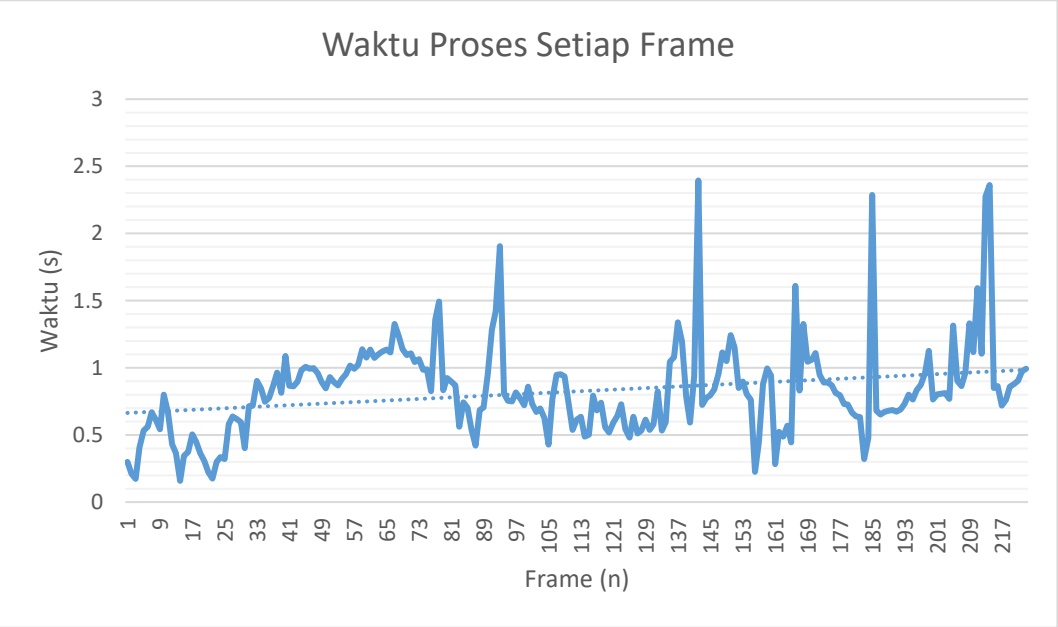
Gambar 4.13 Rute Perjalanan Mobile Robot



Gambar 4.14 Hasil Peta Perjalanan

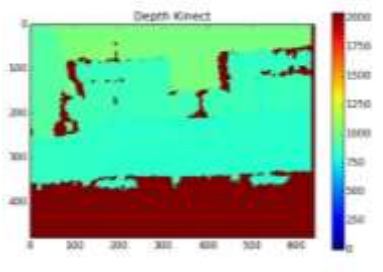
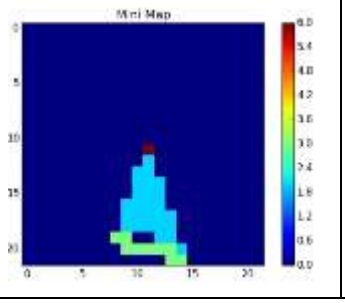
Pengujian terhadap waktu proses di lakukan untuk melihat apakah mini komputer yang digunakan mampu menangani kontrol dari mobile robot. Waktu

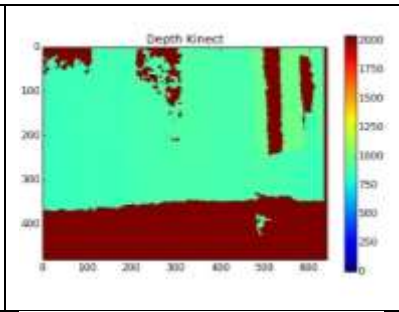
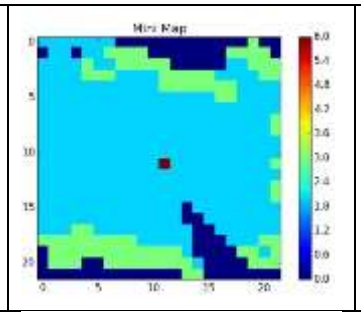
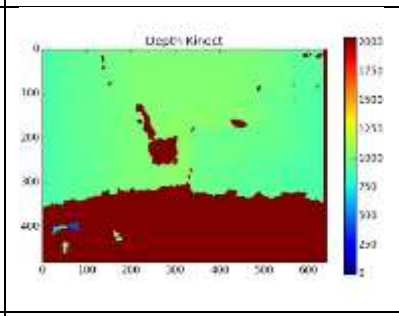
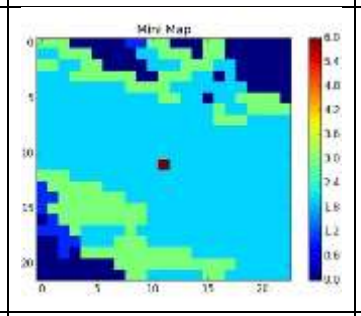
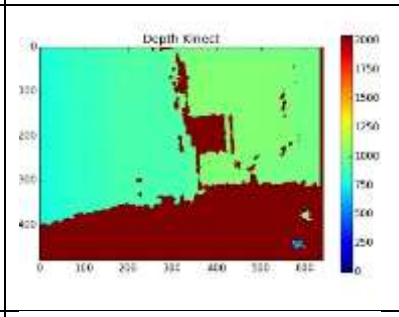
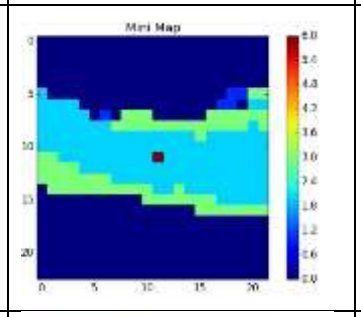
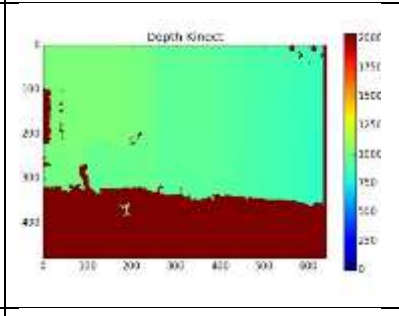
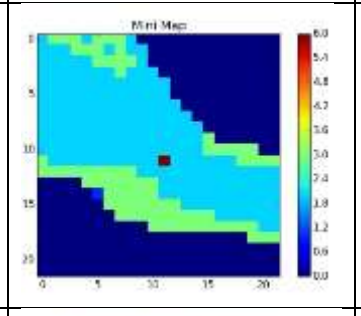
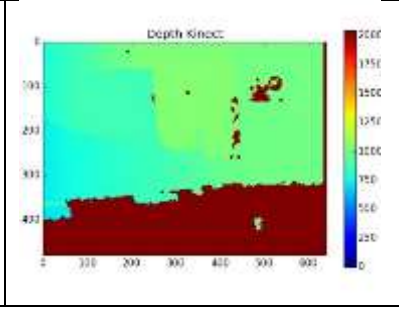
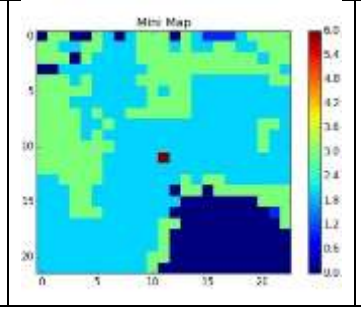
yang tercatat merupakan waktu proses keseluruhan. Waktu proses di mulai dari pengambilan data Kinect, rekonstruksi data Kinect, pemetaan, dan deteksi objek. Grafik waktu proses di tunjukkan pada Gambar 4.15. Tabel 4.4 menunjukkan data yang diproses beserta mini peta yang digunakan untuk navigasi. Rata-rata waktu proses untuk 222 *frame* adalah 824.34 ms. Rata-rata waktu didapatkan ketika sisa baterai sekitar 40%.

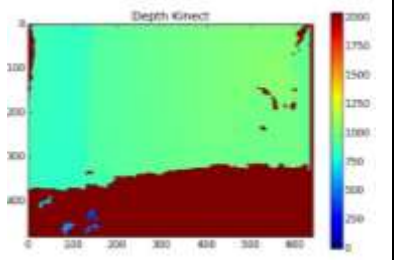
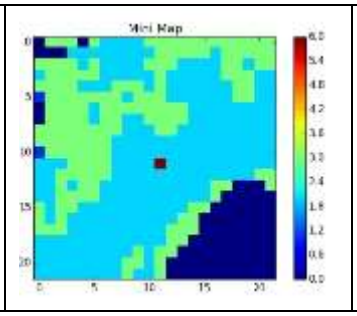


Gambar 4.15 Waktu Proses Pada Pengujian Autonomous

Tabel 4.6 Data Pembacaan Navigasi Pada Percobaan Autonomous

No.	Frame	Detph	Mini Map	Waktu
1	1			299.61

2	25			320.64
3	50			846.49
4	75			987.75
5	100			858.94
6	150			1242.95

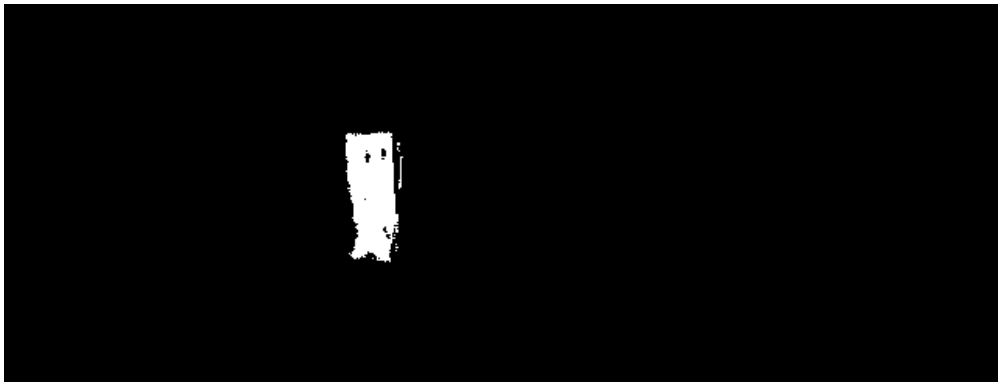
7	200			764.66

4.10 Hasil Pengujian Deteksi Objek

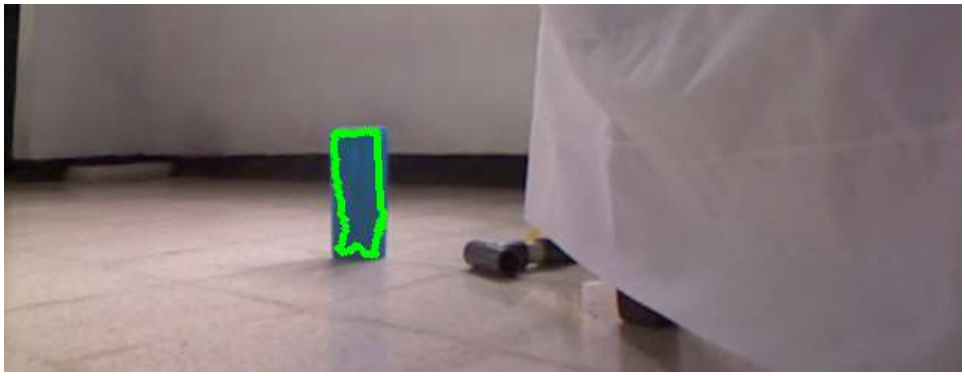
Pengujian untuk mengenali benda di lakukan dengan mengarahkan kamera Kinect ke objek. Pengenalan objek berupa pengenalan warna. Tahap pengenalan objek di lakukan dengan tiga tahap

1. Tahap pertama adalah pengambilan gambar, hasil pada tahap ini di tunjukkan pada Gambar 4.16
2. Tahap kedua adalah menemukan warna dengan rentang tertentu, hasil pada tahap ini di tunjukkan pada Gambar 4.17
3. Tahap terakhir adalah tahap menemukan lokasi dari batas warna yang ditemukan, hasil pada tahap ini di tunjukkan pada Gambar 4.18

Gambar 4.16 Hasil Pengambilan Gambar Kamera Kinect



Gambar 4.17 Hasil Menemukan Warna Dengan Batas Tertentu

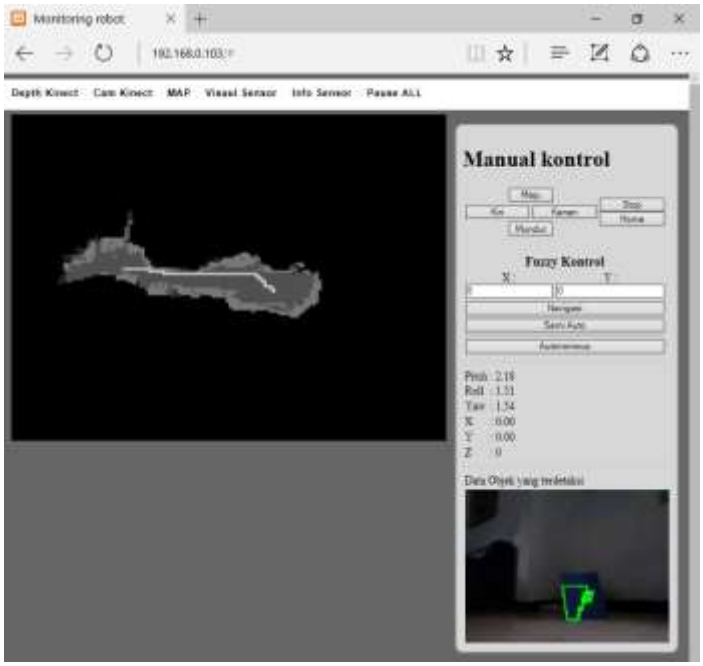


Gambar 4.18 Hasil Menemukan Posisi Batas Warna

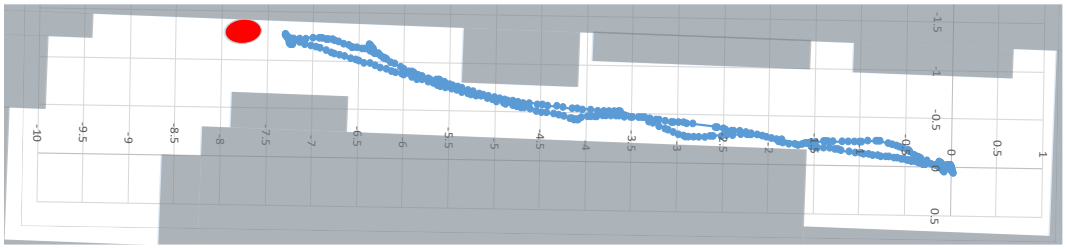
4.11 Hasil Pengujian Mencari Objek

Pada pengujian ini dilakukan percobaan antara deteksi objek dan *autonomous* robot. Ketika objek terdeteksi maka robot akan kembali ke posisi awal. Objek yang terdeteksi akan di tampilkan pada web *interface*, contoh tampilan pada web *interface* di tunjukkan pada Gambar 4.19. Untuk menampilkan rute menuju objek dapat dilakukan dengan klik gambar objek.

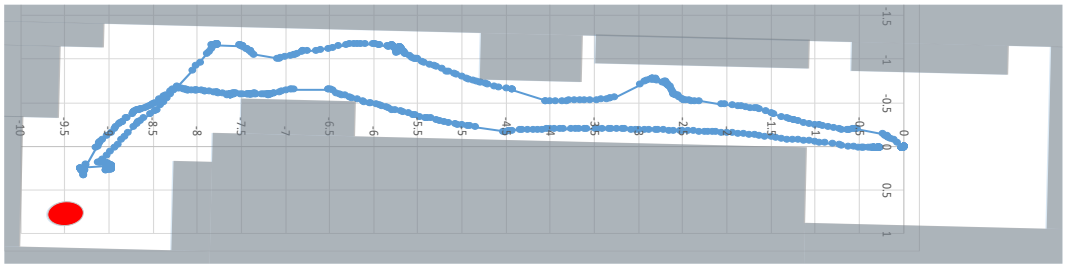
Dalam percobaan dilakukan perekaman posisi robot, dari hasil perekaman posisi robot didapatkan ilustrasi gerakan robot yang ditunjukkan pada Gambar 4.20, pada percobaan ini dilakukan dua percobaan dengan posisi objek yang berbeda. Dari kedua percobaan robot mampu mendeteksi objek dengan baik.



Gambar 4.19 Bentuk Pendeteksian Pada Web *Interface*

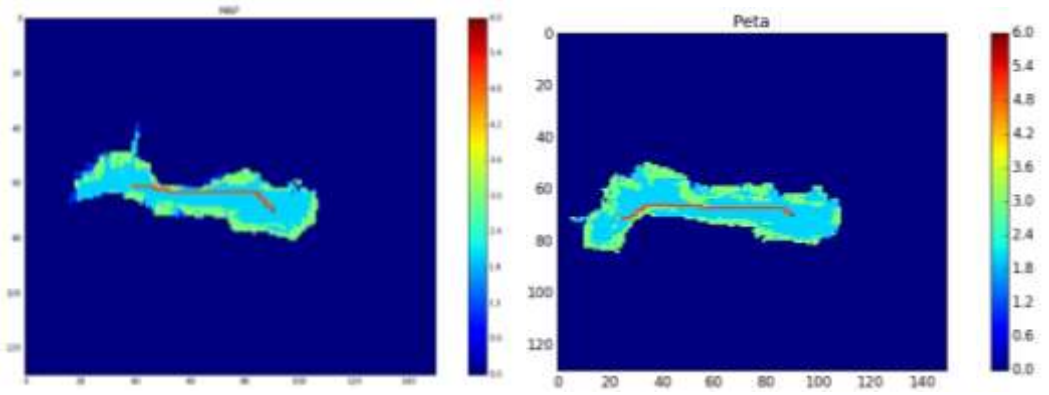


(a)



(b)

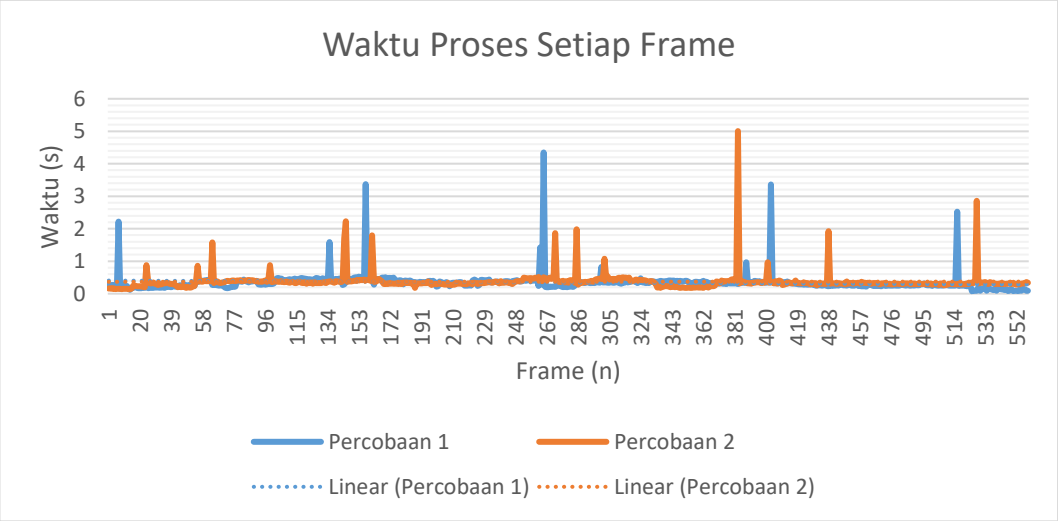
Gambar 4.20 Ilustrasi Posisi Robot dan Posisi Objek (a) Percobaan 1 (b) Percobaan 2



(a) (b)

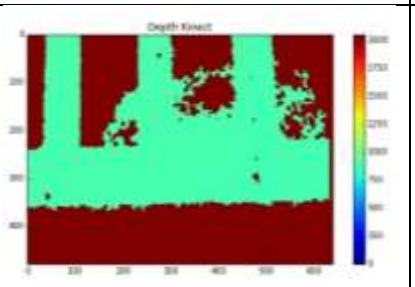
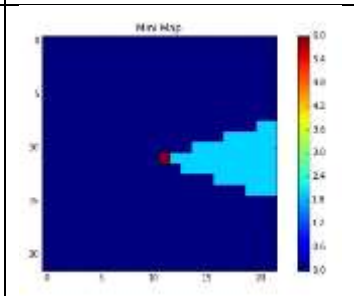
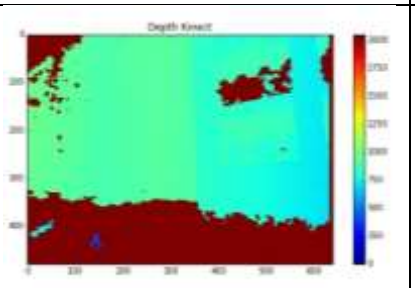
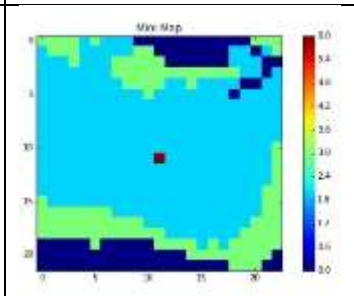
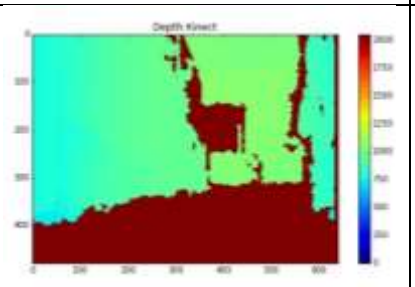
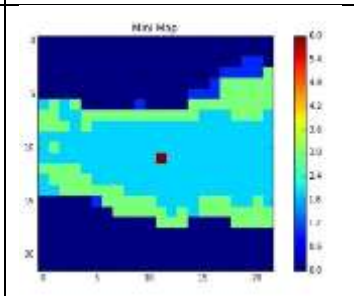
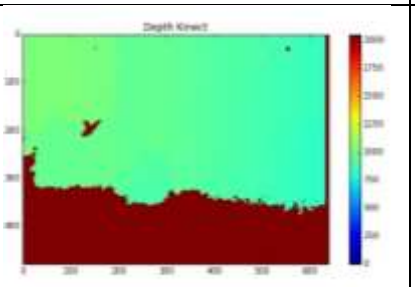
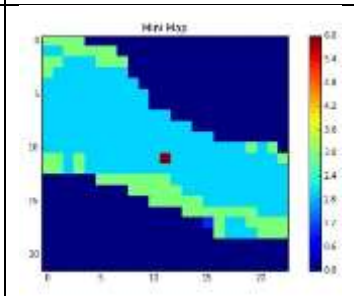
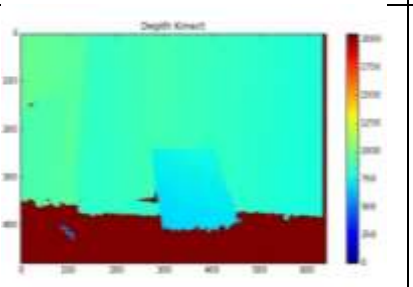
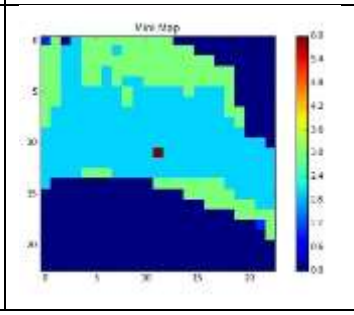
Gambar 4.21 Hasil Peta dan Rute Menuju Objek (a) Hasil Percobaan 1 (b) Hasil Percobaan 2

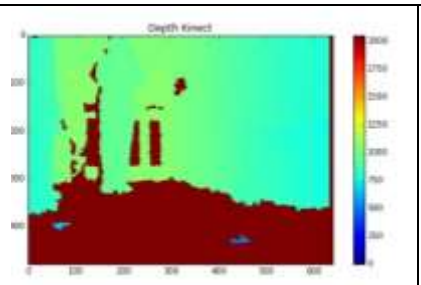
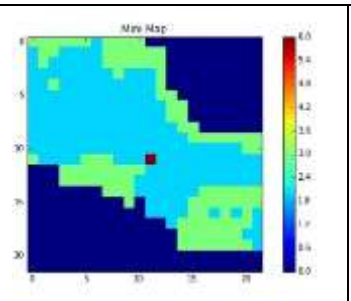
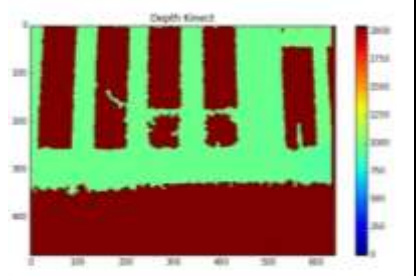
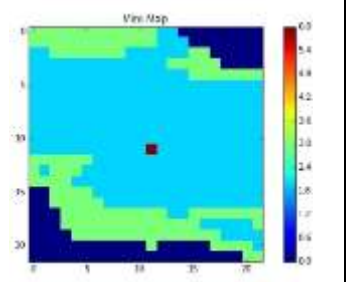
Pengujian terhadap waktu proses di lakukan untuk melihat apakah mini komputer yang digunakan mampu menangani kontrol dari mobile robot. Waktu yang tercatat merupakan waktu proses keseluruhan. Waktu proses di mulai dari pengambilan data Kinect, rekonstruksi data Kinect, pemetaan, dan deteksi objek. Grafik waktu proses di tunjukkan pada Gambar 4.22. Tabel 4.7 menunjukkan data yang diproses beserta mini peta yang digunakan untuk navigasi. Rata-rata waktu proses untuk 552 *frame* adalah 355.71 ms. Rata-rata waktu didapatkan ketika kondisi baterai 100%.



Gambar 4.22 Waktu Proses Untuk Setiap Frame

Tabel 4.7 Data Pembacaan Navigasi Pada Percobaan Pencarian Objek

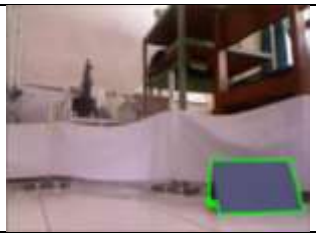
No.	Frame	Depth	Mini Map	Waktu
1	1			259.39
2	125			433.77
3	200			215.05
4	250			379.89
5	300			815.25

6	350			378.49
7	450			263.81

Pada percobaan ini dilakukan pengujian deteksi objek yang memiliki posisi konstan. Robot di ujikan untuk mencari objek dari posisi yang sama. Objek terdeteksi di gambarkan pada peta. Posisi objek yang terdeteksi digambarkan sesuai dengan posisi robot, bukan berdasarkan posisi objek. Sehingga terdapat perbedaan antara objek dengan posisi yang ada di peta. Selisih objek yang sesungguhnya dengan yang ada di peta memiliki selisih jarak antara 0.5 meter – 0.78 meter. Selisih jarak terjadi karena tidak adanya perhitungan eror terhadap posisi objek yang di deteksi. Data hasil percobaan di tunjukkan pada Tabel 4.8.

Tabel 4.8 Data Hasil Posisi Deteksi Objek

No.	**Objek Terdeteksi**	**Posisi Robot (Meter)**		**Posisi Sensor (Meter)**		**Posisi Objek (Meter)**		**Selisih Jarak (Meter)**
		X	**Y**	**X**	**Y**	**X**	**Y**	
1		3.25	0.20	3.27	0.1	3.9	0.15	0.67

2		3.38	0.5	3.43	0.03	3.9	0.15	0.50
3	Tidak Ditemukan	0	0	0	0	3.9	0.15	0
4		3.2	-0.4	3.28	-0.33	3.9	0.15	0.78

BAB 5
KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil pengujian yang dilakukan, sistem pada Raspberry PI di ujikan dengan kondisi baterai sekitar 40% sampai 100%, hasil menunjukkan proses optimal berada pada kondisi diatas 60% dengan rata-rata proses kurang dari 430.73ms.

Sistem pembacaan sensor Kinect di ujikan pada intensitas cahaya antara 3.4 Lux sampai 980 Lux, hasil menunjukkan optimal ketika cahaya kurang dari 118 Lux. Tingkat akurasi jarak juga di uji pada jarak 0.3 meter sampai 15 meter. Ketika intensitas tinggi (980 Lux) jarak yang mampu di baca di bawah 5 meter, sedangkan dengan intensitas rendah (3.4 Lux) mampu membaca sampai 15 meter

Navigasi menggunakan data *online* dan *offline* mampu menghindari penghalang yang ada di depannya. Dan dapat melengkapi sudut pandang yang tidak terjangkau oleh sensor Kinect. Sistem di ujikan untuk melakukan pemetaan ruangan, selisih eror pembuatan peta sekitar 16.43%. Ketika terdapat objek sistem mampu mendeteksi adanya objek dan menunjukkan posisi objek.

5.2 Saran

Pada penelitian ini masih terdapat banyak permasalahan seperti filter Kinect belum di lakukan, sehingga sering terjadi kesalahan saat pemetaan maupun saat navigasi.

Dalam mendeteksi objek tidak cukup dengan menggunakan deteksi warna, pendeteksian objek perlu memperhitungkan jarak minimal robot terhadap objek. Sehingga hasil pendeteksian tidak memiliki kesalahan terhadap objek yang di deteksi.

DAFTAR PUSTAKA

[1] E. Einhorn, C. Schröter dan H. Gross, "Attention-driven monocular scene reconstruction for obstacle detection, robot navigation and map building," *ScienceDirect Robotics and Autonomous Systems,* vol. 59, p. 296–309, 2011.

[2] M. Milford dan G. Wyeth, "Hbrid robot control and SLAM for persistent navigation and mapping," *ScienceDirect Robotics and Autonomous Systems,* vol. 58, pp. 1096-1104, 2010.

[3] V. Bonnet, C. A. Coste, L. Lapierre, J. Cadic, P. Fraisse, R. Zapata, G. Venture dan C. Geny, "Towards an affordable mobile analysis platform for pathological walking assessment," *ScienceDirect Robotics and Autonomous Systems,* vol. 66, pp. 116-128, 2015.

[4] E. Machida, M. Cao, T. Murao dan H. Hashimoto, "Human Motion Tracking of Mobile Robot with Kinect 3D Sensor," dalam *SICE Annual Conference*, Akita Japan, 2012.

[5] E. Machida, M. Cao dan T. Murao, "Human Motion Tracking of Mobile Robot witch Kinect 3D Sensor," dalam *SICE Annual Conference*, Japan, 2012.

[6] M. Cao dan H. Hashimoto, "Specific Person Recognition and Tracking of Mobile Robot with Kinect 3D Sensor," *IEEE,* pp. 8323-8328, 2013.

[7] A. Sgorbissa dan D. Verda, "Structure-based object representation and classification in mobile robotics through a Microsoft Kinect," *Robotics and Autonomous Systems,* no. 61, pp. 1665-1679, 2013.

[8] A. Sgorbissa dan D. Verda, "Structure-based object representation and classification in mobile robotics through a Microsoft Kinect," *Robotics and Autonomous Systems 61 (2013) 1665–1679,* pp. 1665-1679, 2013.

[9] N.-H. Fang, I.-H. Li, W.-Y. Wang, L.-W. Lee dan Y.-H. Chien, "Resarch and Design of Control System for a Tracked Robot with a Kinect Sensor," *International Conference on System Science and Engineering,* pp. 217-222, 2012.

[10] I. Kostavelis dan A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems 66,* pp. 86-103, 2015.

[11] S. Garrido, L. Moreno dan P. U. Lima, "Robot formation motion planning using Fast Marching," *Robotics and Autonomous Systems,* no. 59, pp. 675-683, 2011.

[12] B. M. ElHalawany, H. M. Abdel-Kader, Adly dan Z. B. Nossair, "Modified A* Algorithm for Safer Mobile Robot Navigation," *International Conference on Modelling, Identification & Control,* pp. 74-78, 31 Agusturs 2013.

[13] N. Zainuddin, Y. Mustafah, Y. Shawgi dan N. M. Rashid, "Autonomous Navigation of Mobile Robot Using Kinect Sensor," dalam *International Conference on Computer & Communication Engineering*, Kuala Lumpur, Malaysia, 2014.

[14] J. Borenstein dan Y. Koren, "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots," *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION,* vol. 7, no. 3, pp. 278-288, 1991.

[15] I. Ulrich dan J. Borenstein, "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots," *Intcrnational Conference on Robotics & Automation,* pp. 1572-1577, 1998.

[16] I. Ulrich dan J. Borenstein, "VFH*: Local Obstacle Avoidance with Look-Ahead Verification," *Intcrnational Conference on Robotics & Automation,* pp. 2505-2511, 2000.

[17] "Kinect," Wikipedia, 7 January 2017. [Online]. Available: https://en.wikipedia.org/wiki/Kinect. [Diakses 8 January 2017].

[18] K. Francis dan B. Bird, "KINECT HACKERS ARE CHANGING THE FUTURE OF ROBOTICS," Wired.com, 28 Juni 2011. [Online]. Available: https://www.wired.com/2011/06/mf_kinect/. [Diakses 8 January 2017].

[19] CuriousInventor, "How the Kinect Depth Sensor Works in 2 Minutes," 16 February 2013. [Online]. Available: https://www.youtube.com/watch?v=uq9SEJxZiUg. [Diakses 8 January 2017].

LAMPIRAN

Druvermotor.ino

```
#include <Wire.h>
#include <MPU6050.h>
#include <KalmanFilter.h>
#include <TimerOne.h>
#include <MatrixMath.h>
#include "RunningAverage.h"

MPU6050 mpu;

KalmanFilter kalmanX(0.001, 0.003, 0.03);
KalmanFilter kalmanY(0.001, 0.003, 0.03);
KalmanFilter kalmanZ(0.001, 0.003, 0.03);

bool target=false;

double _x=0;
double _y=0;

float mxpwm=1;
float kalPitch = 0;
float kalRoll = 0;
float kalyaw = 0;

void setup()
{
  Serial.begin(115200);

  motor_set();
  rotari_set();
  mpu_set();
  kompas_set();
  setup_fuzzy();
}

void loop()
{
  mpu_read();
  kompas_read();
}

```

compas.ino

```
#define RAW_DATA_CSV
#define RAW_DATA_ARRAY

#define HMC5883_WriteAddress 0x1E
#define HMC5883_ModeRegisterAddress 0x02
#define HMC5883_ContinuousModeCommand (uint8_t)0x00
#define HMC5883_DataOutputXMSBAddress 0x03
#define compass_rad2degree 57.2957795130823208767981548141 05

#define I2C_TX write
#define I2C_RX read

int regb=0x01;
int regbdata=0x40;
int outputData[6];

const float x_offset = 6.87;
const float y_offset = 45.03;
const float z_offset = -53.47;

const float compass_gain_fact = 1.22;

const float x_scale = 0.94;
const float y_scale = 0.98;
const float z_scale = 0.92;

float comyaw = 0;

double theta=0;

RunningAverage myRA(10);

void kompas_set(){
   Wire.begin();
   theta=kompas();
}

void kompas_read(){
  comyaw=kompas();
  myRA.addValue(comyaw);
  kalyaw = myRA.getFastAverage();
}

float kompas()
{
 int i,x,y,z;
```

```
47.       float angle;
48.       double _x,_y,_z;
49.
50.        Wire.beginTransmission(HMC5883_WriteAddress);
51.        Wire.I2C_TX(regb);
52.        Wire.I2C_TX(regbdata);
53.        Wire.endTransmission();
54.
55.        //delay(1000);
56.        Wire.beginTransmission(HMC5883_WriteAddress);
57.        Wire.I2C_TX(HMC5883_ModeRegisterAddress);
58.        Wire.I2C_TX(HMC5883_ContinuousModeCommand);
59.        Wire.endTransmission();
60.
61.        Wire.beginTransmission(HMC5883_WriteAddress);
62.        Wire.requestFrom(HMC5883_WriteAddress,6);
63.
64.        if(6 <= Wire.available())
65.        {
66.          for(i=0;i<6;i++)
67.          {
68.            outputData[i]=Wire.I2C_RX();
69.          }
70.        }
71.
72.        x=outputData[0] << 8 | outputData[1];
73.        z=outputData[2] << 8 | outputData[3];
74.        y=outputData[4] << 8 | outputData[5];
75.
76.        _x=x*compass_gain_fact*x_scale+x_offset;
77.        _y=y*compass_gain_fact*y_scale+y_offset;
78.        _z=z*compass_gain_fact*z_scale+z_offset;
79.        angle= atan2(_y,_x);
80.
81.        return -angle;
82.       }
83.
```

mpu.ino

```
#define BRAKEVCC 0
#define CW   1
#define CCW  2
#define BRAKEGND 3
#define CS_THRESHOLD 100

int inApin[2] = {7, 4};  // INA: Clockwise input
int inBpin[2] = {8, 9}; // INB: Counter-clockwise input
int pwmpin[2] = {5, 6}; // PWM input
int cspin[2] = {2, 3}; // CS: Current sense ANALOG input
int enpin[2] = {0, 1}; // EN: Status of switches output (Analog pin)

int statpin = 13;

int setmotorr[2]={1,255};
int setmotorl[2]={1,255};

void motor_set(){
  pinMode(statpin, OUTPUT);

  // Initialize digital pins as outputs
  for (int i=0; i<2; i++)
  {
    pinMode(inApin[i], OUTPUT);
    pinMode(inBpin[i], OUTPUT);
    pinMode(pwmpin[i], OUTPUT);
  }
  // Initialize braked
  for (int i=0; i<2; i++)
  {
    digitalWrite(inApin[i], LOW);
    digitalWrite(inBpin[i], LOW);
  }


}

void motorOff(int motor)
{
  // Initialize braked
  for (int i=0; i<2; i++)
  {
    digitalWrite(inApin[i], LOW);
    digitalWrite(inBpin[i], LOW);
  }
  analogWrite(pwmpin[motor], 0);
```

```
47.     }
48.
49.     void motorGo(uint8_t motor, uint8_t direct, uint8_t pwm)
50.     {
51.       if (motor <= 1)
52.       {
53.         if (direct <=4)
54.         {
55.           // Set inA[motor]
56.           if (direct <=1)
57.             digitalWrite(inApin[motor], HIGH);
58.           else
59.             digitalWrite(inApin[motor], LOW);
60.
61.           // Set inB[motor]
62.           if ((direct==0)||(direct==2))
63.             digitalWrite(inBpin[motor], HIGH);
64.           else
65.             digitalWrite(inBpin[motor], LOW);
66.
67.           analogWrite(pwmpin[motor], pwm);
68.         }
69.       }
70.     }
71.
72.     void motordrive(int vr, int vl){
73.       motorOff(0);
74.       motorOff(1);
75.
76.       setmotorr[0]=(vr>0?1:2);
77.       setmotorr[1]=abs(vr)*mxpwm;
78.
79.       setmotorl[0]=(vl>0?1:2);
80.       setmotorl[1]=abs(vl)*mxpwm;
81.
82.       if(setmotorr[1]>50)
83.         motorGo(0, setmotorr[0], setmotorr[1]);
84.       if(setmotorl[1]>50)
85.         motorGo(1, setmotorl[0], setmotorl[1]);
86.
87.     }
88.
```

Rotary.ino

```
#define SUDUTR 0.125109
#define SUDUTL 0.125109
//#define SUDUTL 0.122254
//#define SUDUTR 1
//#define SUDUTL 1

int encoderPin[2] = {18,19};

volatile double encoderPos[2] = {0,0};
volatile double encoderPos1[2] = {0,0};
double lastReportedPos[2] = {0,0};

double kel_roda=0.690;
//double jarak_roda=0.315;
double jarak_roda=0.260;
double sudut[2]= {0,0};



void rotari_set(){
  //kanan
  pinMode(encoderPin[0], INPUT);
  digitalWrite(encoderPin[0], HIGH);
  //kiri
  pinMode(encoderPin[1], INPUT);
  digitalWrite(encoderPin[1], HIGH);

  Timer1.initialize(100000);
  attachInterrupt(4, doEncoderl, RISING);
  attachInterrupt(5, doEncoderr, RISING);
  Timer1.attachInterrupt( kec_rol );
}

void doEncoderr(){
  encoderPos[0] += (setmotorr[0] == 1) ? +SUDUTR : -SUDUTR;
  encoderPos[0] = fmod(encoderPos[0],360.0);
  encoderPos1[0]++;
}

void doEncoderl(){
  encoderPos[1] += (setmotorl[0] == 1) ? +SUDUTL : -SUDUTL;
  encoderPos[1] = fmod(encoderPos[1],360.0);
  encoderPos1[1]++;
}

void kec_rol(){
```

```
47.       Timer1.detachInterrupt();  //stop the timer
48.       Serial.print(_x);
49.       Serial.print(" ");
50.       Serial.print(_y);
51.       Serial.print(" 0 ");
52.       Serial.print(kalPitch);
53.       Serial.print(" ");
54.       Serial.print(kalRoll);
55.       Serial.print(" ");
56.       Serial.println(kalyaw);
57.       get_position(encoderPos[0],encoderPos[1]);
58.       if(target){
59.         go_target(_x,_y,kalyaw,target_x,target_y);
60.         //go_target(_x,_y,theta,target_x,target_y);
61.       }
62.
63.       Timer1.attachInterrupt( kec_rol );
64.     }
65.
66.     void get_position(double sudut_r,double sudut_l){
67.        double vr=kel_roda*del_sudut(sudut[0],sudut_r,180)/(360);
68.        double vl=kel_roda*del_sudut(sudut[1],sudut_l,180)/(360);
69.        sudut[0]=sudut_r;
70.        sudut[1]=sudut_l;
71.        def_kin(vr,vl,_x,_y,kalyaw);
72.        //def_kin(vr,vl,_x,_y,theta);
73.     }
74.
75.     double del_sudut(double s0,double s1,double pi){
76.        double delta=0;
77.        if(abs(s1-s0)>pi){
78.           if(s0>s1)
79.              delta=(pi*2)-abs(s1-s0);
80.           else
81.              delta=abs(s1-s0)-(pi*2);
82.        }
83.        else{
84.           delta=s1-s0;
85.        }
86.
87.        return delta;
88.     }
```

SerialEvent.ino

```
1.  String inputString = "";
2.
3.  void serialEvent() {
4.    while (Serial.available()) {
5.      char inChar = (char)Serial.read();
6.      inputString += inChar;
7.      if (inChar == '\n'){
8.       String com = inputString.substring(0,3);
9.       if(com=="mvr"){//kecepatan motor kanan
10.        setmotorr[1]=inputString.substring(4).toInt();
11.      }
12.      else if(com=="mvl"){//kecepatan motor kiri
13.        setmotorl[1]=inputString.substring(4).toInt();
14.      }
15.      else if(com=="mdr"){//arah motor kanan
16.        setmotorr[0]=inputString.substring(4).toInt();
17.      }
18.      else if(com=="mdl"){//arah motor kiri
19.        setmotorl[0]=inputString.substring(4).toInt();
20.      }
21.      else if(com=="mgo"){//motor bergerak
22.         motorGo(0, setmotorr[0], setmotorr[1]);
23.         motorGo(1, setmotorl[0], setmotorl[1]);
24.      }
25.      else if(com=="sto"){//motor berhenti
26.         motorGo(0, BRAKEGND, 0);
27.         motorGo(1, BRAKEGND, 0);
28.         target=false;
29.      }
30.      else if(com=="ror"){//rotari motor kanan
31.         Serial.println(encoderPos[0]);
32.         //Serial.println(encoderPos1[0]);
33.      }
34.      else if(com=="rol"){//rotari motor kiri
35.         Serial.println(encoderPos[1]);
36.         //Serial.println(encoderPos1[1]);
37.      }
38.      else if(com=="res"){//rotari motor kiri
39.         encoderPos[0]=0;
40.         encoderPos[1]=0;
41.         encoderPos1[0]=0;
42.         encoderPos1[1]=0;
43.         _x=0;
44.         _y=0;
45.         theta=0;
46.         //Serial.println(encoderPos[0]);
```

```
47.             //Serial.println(encoderPos[1]);
48.           }
49.           else if(com=="mpu"){
50.            if(inputString.substring(4,5)=="p"){
51.              Serial.println(kalPitch);
52.            }
53.            else if(inputString.substring(4,5)=="r"){
54.              Serial.println(kalRoll);
55.            }
56.            else if(inputString.substring(4,5)=="y"){
57.              Serial.println(kalyaw);
58.            }
59.            else{
60.              Serial.print(kalPitch);
61.              Serial.print(",");
62.              Serial.print(kalRoll);
63.              Serial.print(",");
64.              Serial.println(kalyaw);
65.            }
66.           }
67.           else if(com=="pos"){
68.            if(inputString.substring(4,5)=="x"){
69.              Serial.println(_x);
70.            }
71.            else if(inputString.substring(4,5)=="y"){
72.              Serial.println(_y);
73.            }
74.            else if(inputString.substring(4,5)=="t"){
75.              Serial.println(theta);
76.            }
77.            else{
78.              Serial.print(_x);
79.              Serial.print(",");
80.              Serial.print(_y);
81.              Serial.print(",0,");
82.              Serial.println(kalyaw);
83.              //Serial.println(theta);
84.            }
85.           }
86.           else if(com=="all"){
87.            Serial.print(_x);
88.            Serial.print(",");
89.            Serial.print(_y);
90.            Serial.print(",0,");
91.            Serial.print(kalPitch);
92.            Serial.print(",");
93.            Serial.print(kalRoll);
```

```
94.             Serial.print(",");
95.             Serial.println(kalyaw);
96.           }
97.           else if(com=="fux"){//koorninat target x
98.             target_x=inputString.substring(4).toFloat();
99.           }
100.          else if(com=="fuy"){//koorninat target y
101.            target_y=inputString.substring(4).toFloat();
102.          }
103.          else if(com=="fut"){//arah motor kiri
104.            target=inputString.substring(4,5)=="1";
105.            motorOff(0);
106.            motorOff(1);
107.          }
108.          else if(com=="mot"){
109.            String text=inputString.substring(4);
110.            int i1 = text.indexOf(",");
111.            //int i2 = text.indexOf(",", i1 + 1);
112.            target=false;
113.            motordrive(text.substring(0,i1).toInt(), text.substring(i1+1).toInt());
114.          }
115.          else if(com=="fuz"){
116.            String text=inputString.substring(4);
117.            int i1 = text.indexOf(",");
118.            //int i2 = text.indexOf(",", i1 + 1);
119.            target_x=text.substring(0,i1).toFloat();
120.            target_y=text.substring(i1+1).toFloat();
121.            target=true;
122.          }
123.          else if(com=="pwm"){
124.            mxpwm=inputString.substring(4).toFloat();
125.          }
126.          inputString="";
127.        }
128.      }
129.    }
```

kinect.py

```
1.  # -*- coding: utf-8 -*-
2.  """
3.  Created on Mon Apr 04 15:08:15 2016
4.
5.  @author: IbnuMuhammadBafagih
6.  """
7.  import numpy as np
8.  import dep_get_xz as mp
9.  #import robot_read_xz as xz
10. import db_rw as dbrw
11.
12. #import cv2
13. import a_start
14.
15. tx,ty=0.0,0.0
16. target_pet=[0,0]
17. kotak=0.10
18. def kontrol(dp,_x,_y,_theta):
19.     global tx,ty
20.     z,x=mp.maping(dp,295)
21.     dbrw.simpan(x,z,_x,_y,_theta,0,0)
22.     vr,vl,db=dbrw.read(_x,_y,_theta,0,0)
23.     return 0,tx,ty,db
24.
```

db_rw.py

```
1.  # -*- coding: utf-8 -*-
2.  import numpy as np
3.  import a_start as ast
4.  import cv2
5.  from scipy import stats
6.
7.
8.  leb=800
9.  db_map=np.zeros((leb*2,leb*2),np.uint8)
10. db_tar=[]
11.
12. #st{1=tidak diketahui; 2= lantai; 3= dinding}
13. lx=-1
14. ly=-1
15. lt=0;
16.
17. # %%
18. def read(r_x,r_y,r_tt,vr,vl,jarak=1):
19.     #return 0,0,map_red(r_x,r_y,jarak)
20.     xa,ya,xb,yb= int(leb+(r_x-jarak)/kotak), int(leb+(r_y-jarak)/kotak),
    int(leb+(r_x+jarak)/kotak), int(leb+(r_y+jarak)/kotak)
21.     return 0,0,db_map[xa:xb+1,ya:yb+1]
22.
23. # %%
24. def koordinat(dep_x,dep_z,rob_x,rob_y,rob_t):
25.     dl=np.sqrt(pow(dep_x,2)+pow(dep_z,2))
26.     dt=np.pi/2-np.arctan2(dep_z,dep_x)
27.     x=rob_x+dl*np.cos(rob_t-dt)
28.     y=rob_y+dl*np.sin(rob_t-dt)
29.     return x,y
30.
31. # %%
32. def sim_lokasi(r_x,r_y,r_tt,r_a,r_b):
33.     global lx,ly
34.     #print r_x,r_y
35.     lx=r_x
36.     ly=r_y
37.
38. # %%
39. def target_baru():
40.     if len(db_tar)>0:
41.         a=db_tar.pop()
42.         return 0,float(a[0]),float(a[1])
43.     else:
44.         return 0,0,0
45.
```

```
46.     # %%
47.     def cek_target():
48.         return len(db_tar)>0
49.
50.     # %%
51.     def target_tambah(x,y):
52.         #db_tar=[]
53.         db_tar.append((x,y))
54.
55.     # %%
56.     def update():
57.         pass
58.
59.     def route_home():
60.         #print lx,ly
61.         #print (int(leb+(lx)/0.25),int(leb+(ly)/0.25)),(leb,leb)
62.         return
        ast.cari_route(db_map.copy(),2,(int(leb+(lx)/kotak),int(leb+(ly)/kotak)),(l
        eb,leb))
```

dep_get_xz.py

```
1.      import numpy as np
2.      from scipy import stats
3.      #import matplotlib.pyplot as plt
4.
5.      kotak=0.10
6.      def maping(dep,posisi=240):
7.
8.          wit=640
9.          jarak=stats.mode(dep[posisi:posisi+20,:])[0][0]
10.         jarak=0.1236 * np.tan(jarak / 2842.5 + 1.1863)
11.         jarak=np.clip(jarak,-2,3.25)
12.
13.         _x=np.zeros(wit)
14.         for x in range(wit-1):
15.             _x[x] = (x-320)*1.12032*jarak[x]/1000
16.         return jarak,_x
```

img_detek.py

```
1.      # -*- coding: utf-8 -*-
2.
3.      import cv2
4.      import numpy as np
5.
6.      lower = np.array([0,0,0], dtype = "uint8")
7.      upper = np.array([255,255,255], dtype = "uint8")
8.
9.      def set_warna(R_min,G_min,B_min,R_max,G_max,B_max):
10.        global lower,upper
11.
12.        lower = np.array([B_min,G_min,R_min], dtype = "uint8")
13.        upper = np.array([B_max,G_max,R_max], dtype = "uint8")
14.
15.     def read_warna(image,batas):
16.        global lower,upper
17.        mask = cv2.inRange(image, lower, upper)
18.        contours,hierarchy = cv2.findContours(mask.copy(), 1, 1)
19.        for cnt in contours:
20.           if cv2.contourArea(cnt)>=batas:
21.              return True
22.        return False
23.
```

serial_util.py

```
1.  # -*- coding: utf-8 -*-
2.  """
3.  Created on Mon May 02 10:00:08 2016
4.
5.  @author: Pavilion 11
6.  """
7.  from threading import Thread,Event,Lock
8.  import Queue
9.  import serial
10. import time
11.
12. lock = Lock()
13. #shared_dict = {1:"0,0,0,0,0,0\n"}
14. shared_dict = {1:["0","0","0","0","0","0"]}
15. class MyThread(Thread):
16.     #ser = serial.Serial('com21', 9600)
17.     def __init__(self, event,q):
18.         Thread.__init__(self)
19.         self.stopped = event
20.         self.q = q
21.         self.thread = Thread(target=self.run)
22.
23.     def run(self):
24.         #self.ser = serial.Serial('com1', 115200)
25.         self.ser = serial.Serial('/dev/ttyUSB0',115200)
26.         lock.acquire()
27.         while not self.stopped.isSet():
28.             #tic = time.clock()
29.             if not workQueue.empty():
30.                 a,lokasi=self.q.get()
31.                 self.ser.flushInput()
32.                 self.ser.flushOutput()
33.                 self.ser.write(a)
34.                 print a
35.
36.             Line = self.ser.readline()
37.             #print Line
38.             data = Line.split()
39.             if len(data) == 6:
40.                 shared_dict[1]=data
41.             #toc = time.clock()
42.             #print toc-tic
43.         self.ser.close()
44.         self.stopped.set()
45.         print "berhenti"
46.
```

```
47.         def serial_data(self,dt):
48.             self.ser.write(dt)
49.             print dt
50.             time.sleep(0.1)
51.             out=""
52.             while self.ser.inWaiting():
53.                 out+=self.ser.read(1)
54.             return out
55.
56.     stopFlag = Event()
57.     workQueue = Queue.Queue(25)
58.     thread = MyThread(stopFlag,workQueue)
59.     def write(dt,lokasi=0):
60.         global workQueue
61.         if not dt in workQueue.queue:
62.             workQueue.put((dt,lokasi))
63.         se_open()
64.
65.     def get(lokasi):
66.         if shared_dict.has_key(lokasi):
67.             return shared_dict[lokasi]
68.         else:
69.             return 0
70.
71.     def close():
72.         global stopFlag
73.         stopFlag.set()
74.
75.     def se_open():
76.         global thread,stopFlag,workQueue
77.         if not thread.is_alive():
78.             stopFlag.clear()
79.             thread = MyThread(stopFlag,workQueue)
80.             thread.start()
```

Indek.html

```
1.  <html>
2.  <head>
3.  <title>Monitoring robot</title>
4.  <script type="text/javascript" src="jquery-1.8.3.min.js"></script>
5.  <script type="text/javascript" src="js.js"></script>
6.  <link rel="stylesheet" type="text/css" href="css.css">
7.  </head>
8.  <body>
9.  <div id="menu">
10. <ul id="menu-h" class="horizontal">
11.   <li><a href="#" onClick="start();">Depth Kinect</a></li>
12.   <li><a href="#" onClick="info();">Info Sensor</a></li>
13.   <li><a href="#" onClick="pause();">Pause</a></li>
14. </ul>
15. </div>
16. <div id="webcam" style="position: absolute; left: 10px; top: 60px;
    width:640px">
17. <noscript>
18. <img src="/bcam.jpg" width="640"/>
19. </noscript>
20. </div>
21. <div id="kontrol" style="position: absolute; left: 655px; top:
    65px;"><h1>Manual kontrol</h1>
22.   <table width="300" id="tombol">
23.     <tr><td width="60" align="center"> </td>
24.       <td colspan="2" align="center"><input type="button"
    value="Maju" onClick="ajaxtombol('1');"></td>
25.       <td width="60" align="center"> </td>
26.       <td rowspan="3"><input type="button" value="Stop"
    onClick="ajaxtombol('0');"></td>
27.     </tr>
28.     <tr>
29.       <td width="60" colspan="2"><input type="button" value="Kiri"
    onClick="ajaxtombol(4);"></td>
30.       <td colspan="2"><input type="button" value="Kanan"
    onClick="ajaxtombol(3);"></td>
31.     </tr>
32.     <tr>
33.       <td align="center"> </td>
34.       <td colspan="2" align="center"><input type="button"
    value="Mundur" onClick="ajaxtombol(2);"></td>
35.       <td align="center"> </td>
36.     </tr>
37.     <tr><td colspan="5"> </td></tr>
38.     <tr><td colspan="5"><h3 align="center">Fuzzy
    Kontrol</h3></td></tr>
```

```
39.         <tr>
40.           <td colspan="3" align="center">X : <input name="in_x"
          type="text" id="in_x" value="0"></td>
41.           <td colspan="2" align="center">Y : <input name="in_y"
          type="text" id="in_y" value="0"></td>
42.         </tr>
43.         <tr>
44.           <td colspan="5" align="center"><input type="button"
          name="button" id="button" value="Fuzzy ON"
          onClick="fuzzy('on')"></td>
45.         </tr>
46.       </table>
47.       <div id="data"></div>
48.     </div>
49.     </body>
50.     </html>
```

RIWAYAT PENULIS

Ali Uroidhi di lahirkan di Bangil pada 24 Agustus 1991, putra ke tiga dari pasangan Muhammad Fauzi Bafagih dan Fatimah Al-Idrus. Penulis memulai pendidikan sekolah dasar di MI Al-Islamiyah pada tahun 1997 - 2003), kemudian menjalankan sekolah menengah pertama di MTs Negeri 1 Bangil pada tahun 2003 - 2006, selanjutnya penulis melanjutkan di sekolah menengah kejuruan di SMK Negeri 1 Bangil pada tahun 2006 – 2009, setelah itu penulis menempuh pendidikan tinggi pada program sarjana di Muhammadiyah Malang pada tahun 2010-2014, dan melanjutkan pendidikan program magister Jurusan Teknik Elektro di Institut Teknologi Sepuluh November Surabaya dengan bidang keahlian Teknik Elektronika pada tahun 2014.