



TUGAS AKHIR - EE184801

LOKALISASI DAN PEMETAAN PADA *INDOOR DOMESTIC SERVICE ROBOT* DENGAN MENGGUNAKAN IMU DAN KAMERA RGB-D

Paltzky Ainurrafi Hidayat
NRP 07111640000104

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020



ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - EE184801

LOKALISASI DAN PEMETAAN PADA *INDOOR DOMESTIC SERVICE ROBOT* DENGAN MENGGUNAKAN IMU DAN KAMERA RGB-D

Paltzky Ainurrafi Hidayat
NRP 07111640000104

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan



FINAL PROJECT - EE184801

**LOCALIZATION AND MAPPING FOR INDOOR DOMESTIC
SERVICE ROBOT USING IMU AND RGB-D CAMERA**

Paltzky Ainurrafi Hidayat
NRP 07111640000104

Supervisor

Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul “Lokalisasi dan Pemetaan pada *Indoor Domestic Service Robot* dengan Menggunakan IMU dan kamera RGB-D” adalah hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan 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, Juli 2020

Paltzky Ainurrafi Hidayat
NRP. 07111640000104

Halaman ini sengaja dikosongkan

LOKALISASI DAN PEMETAAN PADA INDOOR DOMESTIC SERVICE ROBOT DENGAN MENGGUNAKAN IMU DAN KAMERA RGB-D

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I


Dr. Ir. Hendra Kusuma, M.Eng.Sc.
NIP. 196409021989031003

SURABAYA
JULI, 2020

Halaman ini sengaja dikosongkan

**LOKALISASI DAN PEMETAAN PADA INDOOR
DOMESTIC SERVICE ROBOT DENGAN
MENGGUNAKAN IMU DAN KAMERA RGB-D**

TUGAS AKHIR

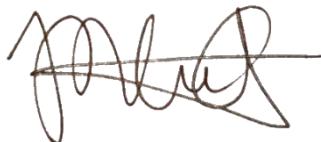
Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing II



Muhammad Atamimi, B.Eng., M.Eng., Ph.D.
NIP. 198503272019031006

**SURABAYA
JULI, 2020**

Halaman ini sengaja dikosongkan

LOKALISASI DAN PEMETAAN PADA INDOOR DOMESTIC SERVICE ROBOT DENGAN MENGGUNAKAN IMU DAN KAMERA RGB-D

Nama : Paltzky Ainurrafi Hidayat
Dosen Pembimbing :
1. Dr. Ir. Hendra Kusuma, M.Eng.Sc.
2. Muhammad Attamimi, B.Eng., M.Eng.,
Ph.D.

ABSTRAK

Lokalisasi (*localization*) dan pembentukan peta (*mapping*) pada robot bergerak merupakan masalah yang penting dan mendasar di dalam penelitian robotika. Persepsi lingkungan dan identifikasi posisi serta orientasi robot terhadap sebuah koordinat global merupakan prasyarat dasar untuk navigasi otonom pada robot bergerak. Pada keadaan ketika robot bergerak otonom harus beroperasi pada sebuah lingkungan yang belum diketahui sebelumnya dibutuhkan lokalisasi dan pemetaan yang dilakukan secara simultan. Sehingga, simultaneous localization and mapping (SLAM) pada robot bergerak di lingkungan yang tidak diketahui sebelumnya menjadi topik yang banyak dibahas dikarenakan nilai teoritis dan kemungkinan penerapannya yang penting. Metode ini dianggap sebagai kunci realisasi robot bergerak otonom yang sesungguhnya. Hal tersebut juga berlaku pada domestic service robot (DSR). DSR beroperasi pada sebuah lingkungan yang dinamis sehingga keadaan disekitarnya selalu berubah-ubah. Agar dapat memenuhi tugasnya dalam lingkungan yang dinamis ini, DSR harus dapat mengenali lingkungan serta berbagai perubahan yang terjadi di sekitarnya. Penelitian ini akan berpusat pada SLAM visual menggunakan kamera RGB-D, yang belakangan ini menjadi topik hangat pada komunitas robotika dikarenakan kemampuannya untuk memperoleh informasi jarak dari suatu lingkungan dengan mudah. Selain itu, akan digunakan juga sebuah *inertial measurement unit* (IMU). Hasil penelitian yang dilakukan menunjukkan hasil pemetaan dengan tingkat eror ukuran sebesar 2%. Eror ini dihasilkan dari sensor kamera RGB-D yang digunakan. Namun untuk bentuk dan karakteristik lainnya menyerupai bentuk aslinya. Untuk penelitian selanjutnya lebih baik digunakan gabungan sensor lain seperti LIDAR untuk mendapatkan hasil yang lebih baik.

Kata Kunci : *Simultaneous localization and mapping*, Robot bergerak, Kamera RGB-D.

Halaman ini sengaja dikosongkan

LOCALIZATION AND MAPPING FOR INDOOR DOMESTIC SERVICE ROBOT USING IMU AND RGB-D CAMERA

Name : Paltzky Ainurrafi Hidayat
Supervisor :
1. Dr. Ir. Hendra Kusuma, M.Eng.Sc.
2. Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ABSTRACT

Localization and mapping on mobile robot are basic and important problem in robotics research. Environment perception, position identification, and orientation identification of the robot to a global coordinate is a basic requirement for autonomous robot to navigate and move around. On a condition where otonomous mobile robot needs to operate in an unknown environment, it requires to do localization and mapping simultaneously. Based on this case, Simultaneous Localization and Mapping (SLAM) on mobile robot that navigate in an unknown environment becomes a hot topic for its theoretical value and possibilities of its application. This method is the key to the realization of the true autonomous robot. This case is also applicable on a Domestic Service Robot (DSR) that runs autonomously or semi-autonomously. DSR operate in a dynamic environment where the condition of the environment around it always changing constantly. To operate on this dynamic environment, DSR must be able to recognize the environment and any changes around it. This research focus on visual SLAM using RGB-D camera, which has become a hot topic in robotics community for its ability to get depth data of the environment easilly. An inertial measuremrnt unit (IMU) is also used to get the posisitin and orientation information of the robot. The result of this research shows that the map we built has 2% error in its dimension. This error is caused by the RGB-D camera. But its form and other characteristics is identical with the real condition. For future research, it's better if we add another sensor like LIDAR to help get better result.

Keyword : *Simultaneous localization and mapping, Mobile robot, RGB-D Camera.*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan YME yang telah memberikan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **“Lokalisasi dan Pemetaan pada Indoor Domestic Service Robot dengan Menggunakan IMU dan kamera RGB-D”** dengan semestinya serta tepat waktu.

Pada kesempatan ini penulis menyampaikan terima kasih atas segala bantuan dan dukungannya yang telah diberikan selama proses pembuatan tugas akhir ini kepada:

1. Keluarga tercinta yang selalu memberikan do'a dan dukungan yang sangat berarti dalam keadaan apapun.
2. Bapak Dr. Ir. Hendra Kusuma, M.Eng.Sc. dan Bapak Muhammad Attamimi, B.Eng., M.Eng., Ph.D. selaku Dosen Pembimbing atas segala bimbingan ilmu, moral, dan spiritual dari awal hingga terselesaiannya tugas akhir.
3. Seluruh Staff/Karyawan/Dosen Departemen Teknik Elektro yang telah memberikan banyak ilmu dan menciptakan suasana belajar yang mendukung.
4. Teman-teman e56 yang telah berjuang bersama semenjak awal masuk perkuliahan.
5. Semua pihak yang telah membantu penulis dalam menyelesaikan penelitian tugas akhir ini.

Surabaya, 15 Juli 2020

Paltzky Ainurrafi Hidayat

Halaman ini sengaja dikosongkan

DAFTAR ISI

HALAMAN JUDUL

PERNYATAAN KEASLIAN

LEMBAR PENGESAHAN

ABSTRAK	I
ABSTRACT	III
KATA PENGANTAR.....	V
DAFTAR ISI.....	VII
DAFTAR GAMBAR.....	XI
DAFTAR TABEL	XV
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	1
1.3 Batasan Masalah.....	2
1.4 Tujuan Penelitian.....	2
1.5 Metode Penelitian.....	2
1.5.1 Studi Literatur	2
1.5.2 Perancangan Sistem.....	2
1.5.3 Pengujian dan Penyempurnaan Sistem.....	3
1.5.4 Penyusunan Buku Tugas Akhir	3
1.6 Sistematika Penulisan	3
BAB 2 TEORI PENUNJANG	5
2.1 Robot	5
2.2 Robot <i>Omnidirectional</i>	6
2.3 Posisi dan Orientasi	6
2.4 Lokalisasi.....	8
2.5 STM32F4 Discovery	9
2.6 Nvidia Jetson Nano Developer Kit	10
2.7 Intel Realsense D435i.....	12

2.8	Robot Operating System (ROS)	13
2.9	Madgwick Filter	14
BAB 3 PERANCANGAN SISTEM	17	
3.1	Diagram Blok Sistem.....	17
3.2	Rancangan Perangkat Keras	18
3.2.1	Rancangan Mekanik.....	18
3.2.2	Rancangan Elektronik	20
3.3	Perancangan Perangkat Lunak.....	23
3.3.1	Program Kendali Manual <i>Base</i> Robot.....	23
3.3.2	Algoritma Lokalisasi dan Pemetaan.....	25
3.3.3	Algoritma RTabMap	26
3.4	Pengambilan Data Berbasis Informasi Visual dan IMU	27
3.5	Proses Pemetaan dan Lokalisasi	28
BAB 4 PENGUJIAN DAN ANALISIS.....	31	
4.1	Pengujian Pada Base Robot	31
4.1.1	Pengujian Pergerakan Robot	31
4.1.2	Pengujian Kecepatan Optimal	43
4.1.3	Analisa Hasil Percobaan Pada <i>Base</i> Robot	44
4.2	Pengujian Kamera RGB-D Intel Realsense D435i	44
4.2.1	Kalibrasi IMU	44
4.2.2	Kalibrasi <i>depth</i>	45
4.2.3	Pengujian <i>depth</i>	46
4.2.4	Analisa Hasil Pengujian Kamera Intel Realsense D435i.	47
4.3	Pengujian Algoritma RtabMap	47
4.3.1	Pengujian Bentuk Benda	47
4.3.2	Ilustrasi Pengujian Pemetaan.....	53
4.3.3	Tempat Pengujian Pemetaan	53
4.3.4	Hasil Pengujian Pemetaan	57
4.3.5	Analisa Hasil Pengujian RTabMap	60
BAB 5 PENUTUP	63	
5.1	Kesimpulan.....	63

5.2 Saran	63
DAFTAR PUSTAKA	65
LAMPIRAN.....	67
Program Kalibrasi IMU Realsense D435i.....	67
Hasil Kalibrasi IMU pada Realsense D435i.....	82
Program <i>Wireless Controller</i> pada Arduiono UNO	86
Program Utama pada STM32F4Discovery Master	88
Program Utama pada STM32F4Discovery <i>Slave</i>	99
BIODATA PENULIS.....	109

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1 Pergerakan robot <i>omnidirectional</i>	6
Gambar 2.2 Sistem koordinat kartesian	7
Gambar 2.3 Papan STM32F4-Discovery	9
Gambar 2.4 Nvidia Jetson Nano Developer Kit.....	10
Gambar 2.5 Intel Realsense D435i.....	12
Gambar 2.6 Logo ROS.....	14
Gambar 2.7 Perhitungan matematis madgwick filter.....	15
Gambar 3.1 Diagram blok keseluruhan sistem	17
Gambar 3.2 Model 3 dimensi kerangka robot tampak depan	18
Gambar 3.3 model 3 dimensi kerangka robot tanpa pelindung	19
Gambar 3.4 Model 3 dimensi kerangka robot tampak bawah.....	20
Gambar 3.5 Rancangan elektronik pada <i>head module</i> robot.....	21
Gambar 3.6 Rancangan elektronik pada <i>base</i> robot.....	22
Gambar 3.7 Diagram blok aliran data program kendali manual	23
Gambar 3.8 <i>Wireless controller</i> yang digunakan.....	24
Gambar 3.9 Diagram blok algoritma lokalisasi dan pemetaan.....	25
Gambar 3.10 Diagram blok algoritma RtabMap	26
Gambar 3.11 Data RGB	27
Gambar 3.12 Data <i>depth</i>	27
Gambar 3.13 Landmark yang tertangkap pada suatu <i>frame</i> data	29
Gambar 3.14 Peta yang terbentuk dengan posisi robot pada peta	29
Gambar 4.1 Xiaomi Duka	31
Gambar 4.2 Tempat pengujian <i>base</i> robot	32
Gambar 4.3 Pemberian perintah gerak maju pada <i>controller</i>	33
Gambar 4.4 Pose awal dan pose setelah bergerak maju sejauh 3 meter	33
Gambar 4.5 Pemberian perintah mundur pada <i>controller</i>	34
Gambar 4.6 Pose awal dan pose setelah bergerak mundur sejauh 3 meter	34
Gambar 4.7 Pemberian perintah gerak kanan pada <i>controller</i>	35
Gambar 4.8 Pose awal dan pose setelah bergerak ke kanan sejauh 3 meter	36
Gambar 4.9 Pemberian perintah gerak kiri pada <i>controller</i>	36
Gambar 4.10 Pose awal dan pose setelah bergerak ke kiri sejauh 3 meter	37
Gambar 4.11 Pemberian perintah gerak diagonal kanan depan pada <i>controller</i>	38

Gambar 4.12 Pemberian perintah gerak diagonal kiri depan pada <i>controller</i>	38
Gambar 4.13 Pemberian perintah gerak diagonal kanan belakang pada <i>controller</i>	39
Gambar 4.14 Pemberian perintah gerak diagonal kiri belakang pada <i>controller</i>	40
Gambar 4.15 Pemberian perintah gerak putar kanan pada <i>controller</i> ...	41
Gambar 4.16 Pose awal dan pose setelah berputar ke kanan sebanyak 6 kali	41
Gambar 4.17 Pemberian perintah gerak putar kiri pada <i>controller</i>	42
Gambar 4.18 Pose awal dan pose setelah berputar ke kiri sebanyak 6 kali	42
Gambar 4.19 Posisi kalibrasi IMU	45
Gambar 4.20 Kalibrasi <i>on-chip</i> menggunakan Intel Realsense Viewer	46
Gambar 4.21 Benda A.....	47
Gambar 4.22 Posisi benda A dan robot saat pengamatan	48
Gambar 4.23 Hasil pemodelan Benda A	48
Gambar 4.24 Benda B	49
Gambar 4.25 Posisi benda B dan robot saat pengamatan	49
Gambar 4.26 Hasil pemodelan benda B	50
Gambar 4.27 Benda C	50
Gambar 4.28 Posisi benda C dan robot saat pengamatan	51
Gambar 4.29 Hasil pemodelan benda C	51
Gambar 4.30 Posisi benda C dan robot setelah robot bergerak	52
Gambar 4.31 Hasil pemodelan benda C setelah robot bergerak.....	52
Gambar 4.32 Ilustrasi proses pemetaan.....	53
Gambar 4.33 Ruang A	54
Gambar 4.34 Ruang B.....	54
Gambar 4.35 Ruang C	55
Gambar 4.36 Denah ruang A, B, dan C	56
Gambar 4.37 Hasil pemetaan dan lokalisasi pada ruangan A	57
Gambar 4.38 Hasil pada ruangan A dari sisi yang berbeda	57
Gambar 4.39 Konversi peta ruangan A menjadi 2 dimensi.....	58
Gambar 4.40 Hasil pemetaan dan lokalisasi pada ruangan B	58
Gambar 4.41 Konversi peta ruangan B menjadi dua dimensi	59
Gambar 4.42 Hasil pemetaan dan lokalisasi pada ruangan C	59
Gambar 4.43 Konversi peta ruangan C menjadi dua dimensi	60
Gambar 4.44 <i>Landmark</i> yang terdeteksi pada ruangan B	61
Gambar 4.45 <i>Lost track</i> yang terjadi pada ruangan B	61

Gambar 4.46 Database yang dihasilkan oleh RtabMap..... 62

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 2.1 Spesifikasi Nvidia Jetson Nano Developer Kit	11
Tabel 2.2 Spesifikasi Intel Realsense D435i	13
Tabel 4.1 Hasil pengujian gerak maju.....	34
Tabel 4.2 Hasil pengujian gerak mundur	35
Tabel 4.3 Hasil pengujian gerak kanan	36
Tabel 4.4 Hasil pengujian gerak kiri	37
Tabel 4.5 Hasil pengujian gerak putar kanan.....	41
Tabel 4.6 Hasil pengujian gerak putar kiri.....	43
Tabel 4.7 Hasil pengujian kecepatan optimal	43
Tabel 4.8 Hasil pengujian depth.....	46

Halaman ini sengaja dikosongkan

BAB 1

PENDAHULUAN

Dalam bab ini akan dipaparkan tentang latar belakang penelitian, permasalahan yang akan diselesaikan, tujuan dari penelitian, metodologi yang digunakan, sistematika penulisan serta relevansi penelitian ini terhadap penelitian sejenis atau lebih rumit di masa mendatang.

1.1 Latar Belakang

Ketika robot memasuki sebuah lingkungan yang belum diketahui sebelumnya, robot tersebut perlu untuk mendekripsi sekeliling untuk menentukan posisinya dan membentuk sebuah peta lingkungan tersebut secara bersamaan. Hal tersebut bisa dicapai dengan menggunakan metode *simultaneous localization and mapping* (SLAM). Permasalahan yang dihadapi oleh SLAM menanyakan kemungkinan sebuah robot bergerak dapat diletakkan pada sebuah lokasi yang belum diketahui sebelumnya dengan lingkungan yang tidak diketahui pula [1].

Pada *domestic service robot* kemampuan untuk memahami lingkungan disekitarnya sangat dibutuhkan untuk memenuhi tugasnya sebagai robot semi-otonom atau otonom pembantu tugas manusia dalam kehidupan sehari-hari. Pemilihan metode SLAM dengan menggunakan Inertial Measurement Unit (IMU) dan kamera RGB-D ini dilakukan karena metode ini memberikan hasil yang lebih detail serta presisi dan akurat. Informasi yang didapat dari metode ini juga dapat digunakan untuk mengembangkan fungsi dari robot tersebut. Untuk penelitian tugas akhir ini, robot yang dirancang diharapkan dapat melakukan pemetaan lingkungan disekitarnya serta melakukan lokalisasi guna mendapatkan koordinat robot itu sendiri, benda, dan halangan yang berada di sekitarnya pada peta yang sedang dibentuk. Hal ini dilakukan agar *domestic service robot* tersebut dapat melakukan berbagai tugas yang akan diberikan secara optimal dengan lebih memahami karakteristik lingkungan disekitarnya.

1.2 Rumusan Masalah

Perumusan masalah pada Tugas Akhir ini adalah :

1. Bagaimana cara menentukan posisi dan orientasi dari sebuah *domestic service robot*?
2. Bagaimana cara sebuah *domestic service robot* mengetahui dan menyimpan bentuk serta karakteristik lingkungan disekitarnya?

3. Apakah lokalisasi dan pemetaan pada *domestic service robot* dapat dilakukan secara simultan?

1.3 Batasan Masalah

Perlu diperhatikan batasan dalam melakukan penelitian ini, antara lain sebagai berikut:

- a. Robot uji coba digerakkan secara manual menggunakan *remote control*.
- b. Lingkungan tempat robot dioperasikan merupakan sebuah ruangan tertutup.
- c. Hasil pemetaan dan lokalisasi berupa data PCL 3 dimensi mentah yang dapat dikonversikan menjadi peta 2 dimensi serta harus diolah kembali agar dapat digunakan untuk navigasi robot otonom.

1.4 Tujuan Penelitian

Tujuan usulan penelitian ini dapat dijabarkan sebagai berikut:

1. Melakukan lokalisasi dan pemetaan secara simultan menggunakan SLAM visual.
2. Menerapkan SLAM visual pada *indoor domestic service robot*.
3. Menghasilkan sebuah peta lingkungan disekitar robot.

1.5 Metode Penelitian

Penelitian ini akan dilaksanakan dengan melalui beberapa tahap proses yang telah dirancang sebagai berikut :

1.5.1 Studi Literatur

Dalam tahap Studi Literatur, akan dipelajari mengenai pengoperasian Linux pada Jetson Nano, penggunaan kamera RGB-D Realsense D435i, konsep *simultaneous localization and mapping* (SLAM) visual, serta penggunaannya pada sistem robot yang dirancang.

1.5.2 Perancangan Sistem

Dalam tahap Perancangan Sistem, pertama akan dilakukan perancangan sistem mekanik dan elektronik dari robot yang digunakan. Kemudian akan dilanjutkan dengan merancang perangkat lunak yang akan mengatur sistem gerak robot tersebut. Terakhir akan dilakukan

perancangan perangkat lunak dari algoritma SLAM yang digunakan untuk melakukan pemetaan dan lokalisasi pada robot tersebut.

1.5.3 Pengujian dan Penyempurnaan Sistem

Untuk dapat memastikan sistem telah memenuhi tujuan yang dimiliki, perlu dilakukan beberapa pengujian yang terukur. Beberapa pengujian ini meliputi pengujian gerak *base* robot, pengujian tingkat akurasi data kamera Intel Realsense D435i, hingga pengujian hasil pemetaan dan lokalisasi yang dilakukan.

1.5.4 Penyusunan Buku Tugas Akhir

Dalam tahap Penyusunan Buku Tugas Akhir, laporan terkait hasil proses yang telah dilakukan pada penelitian ini akan disusun dan disajikan dalam bentuk buku.

1.6 Sistematika Penulisan

BAB 1 PENDAHULUAN

Dalam pendahuluan akan dijelaskan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, serta relevansi dari penelitian.

BAB 2 TEORI PENUNJANG

Dalam teori penunjang dijelaskan mengenai teori dasar dari penelitian yang dilakukan. Teori dasar yang mendasari konsep dari SLAM visual serta beberapa *library* pendukung yang digunakan untuk proses kalibrasi serta pengolahan data.

BAB 3 PERANCANGAN SISTEM

Perancangan sistem mencakup penjelasan dari sistem yang dirancang. Penjelasan ini dapat berupa diagram blok, *flowchart*, hingga kode pendukung.

BAB 4 PENGUJIAN DAN ANALISIS

Pada Pengujian dan Analisis akan dilakukan percobaan pada sistem yang dirancang dalam menyelesaikan beberapa masalah yang disebutkan pada bagian rumusan masalah. Hasilnya akan berupa peta 3 dimensi dan 2 dimensi. Data tersebut kemudian akan digunakan sebagai bahan analisis.

BAB 5 PENUTUP

Hasil dari percobaan yang dilakukan akan diterangkan dalam beberapa poin yang merepresentasikan semua percobaan yang telah dilakukan. Kemudian akan diberikan beberapa saran yang dapat digunakan untuk penelitian selanjutnya agar dapat mengatasi masalah-masalah yang terjadi pada penelitian ini.

BAB 2

TEORI PENUNJANG

Pada bab ini akan dijelaskan beberapa teori dasar yang mendukung percobaan yang akan dilakukan. Teori dasar tersebut akan menjelaskan tentang dasar dari pengolahan citra hingga algoritma SLAM visual yang digunakan.

2.1 Robot

Istilah robot pertama kali muncul di dalam sebuah drama fiksi Ceko pada tahun 1920 yang berjudul "*Rossum's Universal Robots*" oleh Karel Capek. Robot yang dimaksud pada drama tersebut merupakan manusia buatan atau biasa disebut "*android*". Drama ini kemudian menginspirasi beberapa cerita lainnya yang membahas moral dan interaksi antara manusia dan robot. Drama dan cerita ini yang kemudian membentuk persepsi manusia terhadap robot dan mendorong mereka untuk berkreasi dalam mewujudkan konsep robot tersebut.

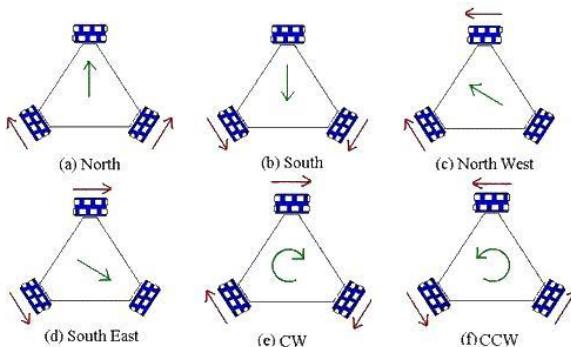
Robot memiliki banyak definisi, namun secara garis besar robot adalah sebuah mesin yang berorientasi terhadap tujuan yang dapat merasa, membuat rencana, dan melakukan aksi. Sebuah robot dapat merasakan dan mendeteksi lingkungannya, dan dengan sebuah tujuan, untuk merencanakan sebuah aksi. Aksi ini dapat berupa menggerakkan lengan robot untuk menggenggam sesuatu hingga bergerak dari satu titik ke titik lainnya. Berdasarkan penggunaannya robot dapat dibagi menjadi dua kategori, yaitu *factory robot* dan *field robot*. *Factory robot* merupakan robot yang digunakan dalam membantu proses produksi pada pabrik. Sedangkan *field robot* merupakan robot yang tidak digunakan pada proses produksi dan biasanya merupakan robot bergerak. *Field robot* yang berinteraksi dengan manusia untuk membantu manusia dalam menyelesaikan tugasnya disebut *service robot*.

Field robot dan *service robot* menghadapi suatu tantangan yang spesifik dan signifikan. Tantangan pertama yang harus dihadapi adalah robot tersebut harus beroperasi dan bergerak di dalam lingkungan yang kompleks. Banyak benda dan halangan yang terdapat pada lingkungan tersebut. Robot tersebut harus dapat mengenali berbagai benda dan halangan itu agar dapat mencapai tujuannya dengan baik. Tantangan kedua adalah robot tersebut harus dapat beroperasi dengan aman di dalam lingkungan yang terdapat banyak manusia. Kedua hal tersebut membuat *sensing* menjadi hal yang penting pada robot. Untuk melakukan *sensing*

ini robot menggunakan berbagai sensor untuk mendapatkan berbagai informasi tentang lingkungan disekitarnya.

2.2 Robot *Omnidirectional*

Robot *omnidirectional* adalah robot dengan sistem pergerakan yang secara langsung dapat bergerak kesegala arah dengan konfigurasi apapun. Roda *omni* adalah roda yang memiliki rol terpasang di pinggiran roda sedemikian rupa sehingga sumbu roda adalah tegak lurus terhadap sumbu roda dasar. Untuk gerak yang halus, rol dipasang di atas bantalan. Pada gambar 2.1 terlihat tiga roda *omni* dipasang di atas sebuah dasar berbentuk segitiga. Roda ini terpasang dengan sudut 120 derajat satu sama lain. Inilah keuntungan dari arah Omni roda di mana tanpa menerapkan mekanisme kemudi masih bisa berputar dan memindahkan tubuh ke berbagai arah. Roda ini bekerja berdasarkan prinsip *parallelogram*. Dua gaya bertindak pada sudut tertentu satu sama lain, sehingga arah gerak akhirnya adalah arah vektor yang dihitung menggunakan hukum *parallelogram*. Robot beroda *omni* dapat bergerak di sudut mana pun dan ke segala arah, tanpa harus berputar sebelumnya [2].

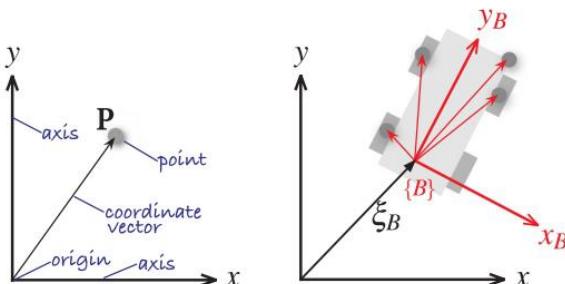


Gambar 2.1 Pergerakan robot *omnidirectional*

2.3 Posisi dan Orientasi

Angka adalah bagian penting dari matematika. Kita menggunakan angka untuk menghitung, misalnya ada 2 apel. Kita menggunakan angka denominasi, angka plus satuan, untuk menentukan jarak, misalnya objek berjarak 2 m. Kita juga menyebut nomor tunggal ini sebagai skalar. Kita

menggunakan vektor, angka denominasi plus arah, untuk menentukan lokasi: objek 2 m di utara. Kita mungkin juga ingin mengetahui orientasi objek: objek 2 m di utara dan menghadap ke barat. Kombinasi posisi dan orientasi ini kita sebut pose. Titik dalam ruang adalah konsep yang akrab dari matematika dan dapat dijelaskan oleh vektor koordinat, seperti yang ditunjukkan pada gambar 2.1. Vektor mewakili perpindahan titik sehubungan dengan beberapa kerangka koordinat referensi – kita sebut ini vektor terikat karena tidak dapat dipindahkan secara bebas. Bingkai koordinat, atau Sistem koordinat Cartesian, adalah seperangkat sumbu ortogonal yang berpotongan pada suatu titik dikenal sebagai asal. Vektor dapat dijelaskan dalam hal komponennya, linear kombinasi vektor satuan yang sejajar dengan sumbu bingkai koordinat. Perhatikan bahwa titik dan vektor adalah jenis objek matematika yang berbeda masing-masing dapat dideskripsikan dengan angka *tuple*. Kita dapat menambahkan vektor tetapi menambahkan poin tidak masuk akal. Perbedaan dua titik adalah vektor, dan kita dapat menambahkan vektor ke titik untuk mendapatkan titik lain [3].



Gambar 2.2 Sistem koordinat kartesian [3]

Suatu titik adalah abstraksi matematika yang menarik, tetapi objek nyata terdiri banyak poin yang tak terhingga. Objek, tidak seperti titik, juga memiliki orientasi. Jika kita melampirkan bingkai koordinat ke objek, seperti yang ditunjukkan pada gambar 2.1, kita bisa menggambarkan setiap titik dalam objek sebagai vektor konstan sehubungan dengan bingkai itu. Sekarang kita bisa menggambarkan posisi dan orientasi dari kerangka koordinat tersebut sehubungan dengan kerangka koordinat referensi. Untuk membedakan berbagai bingkai yang kami beri label

mereka dan dalam hal ini bingkai koordinat objek diberi label {B} dan sumbunya diberi label x_B dan y_B , mengadopsi label frame sebagai subskrip mereka.

Untuk sepenuhnya menggambarkan pose benda kaku di dunia 3 dimensi yang kita butuhkan adalah 6 dimensi, 3 untuk menggambarkan posisinya dan 3 untuk menggambarkan orientasinya. Pada dimensi ini memiliki perilaku yang sangat berbeda. Jika kita meningkatkan nilai salah satu posisi, dimensi objek akan bergerak terus menerus dalam garis lurus, tetapi jika kita meningkatkan nilai salah satu dimensi orientasi objek akan berputar dalam beberapa cara dan segera kembali ke orientasi aslinya, dimensi ini melengkung. Kita jelas perlu memperlakukan dimensi posisi dan orientasi dengan sangat berbeda.

2.4 Lokalisasi

Lokalisasi robot adalah proses menentukan di mana robot bergerak berada sehubungan dengan lingkungannya. Lokalisasi adalah salah satu kompetensi paling mendasar yang diperlukan oleh robot otonom karena pengetahuan tentang lokasi robot itu sendiri merupakan dasar yang penting untuk membuat keputusan tentang tindakan di masa depan. Dalam skenario lokalisasi robot yang khas, peta lingkungan tersedia dan robot dilengkapi dengan sensor yang mengamati lingkungan serta memantau gerakannya sendiri. Masalah lokalisasi kemudian menjadi salah satu estimasi posisi robot dan orientasi dalam peta menggunakan informasi yang dikumpulkan dari sensor ini. Teknik pelokalan robot harus mampu menangani *noise* pengamatan dan menghasilkan tidak hanya estimasi lokasi robot tetapi juga ukuran ketidakpastian estimasi lokasi. Dua teknik probabilistik yang paling umum, filter Kalman yang diperluas dan filter partikel, yang dapat digunakan untuk menggabungkan informasi dari sensor untuk menghitung perkiraan lokasi robot [3].

Robot bergerak yang dilengkapi dengan sensor untuk memantau gerakannya sendiri, misalnya menggunakan *encoder* roda dan sensor inersia, dapat menghitung perkiraan lokasinya relatif terhadap di mana ia dimulai jika model matematika dari gerakan tersebut tersedia. Ini dikenal sebagai perhitungan odometri atau *dead reckoning*. Kesalahan yang ada dalam pengukuran sensor dan model gerak membuat perkiraan lokasi robot yang diperoleh dari perhitungan *dead reckoning* semakin tidak dapat diandalkan saat robot menavigasi di lingkungannya. Kesalahan dalam perkiraan perhitungan *dead reckoning* dapat diperbaiki ketika

robot dapat mengamati lingkungannya menggunakan sensor dan mampu menghubungkan informasi yang dikumpulkan oleh sensor ini dengan informasi yang terkandung dalam peta.

2.5 STM32F4 Discovery



Gambar 2.3 Papan STM32F4-Discovery [4]

Papan evaluasi STM32F4-Discovery dari ST Microelectronics adalah papan berbiaya rendah untuk mengevaluasi kisaran STM32F4 dari mikrokontroler ARM Cortex-M4 32-bit. Satu-satunya perangkat keras yang diperlukan untuk mulai menggunakan papan adalah kabel USB tipe A ke mini-B dan port USB pada PC. Platform Windows diperlukan untuk menjalankan salah satu rangkaian alat pengembangan untuk mikrokontroler STM32. *Toolchain* dari Atollic, IAR dan Keil tersedia untuk dibeli. *Toolchain* ini juga memiliki versi evaluasi terbatas. Sebagai alternatif, *open source* GNU ARM *toolchain* dapat digunakan untuk memprogram mikrokontroler STM32, salah satu contohnya adalah YAGARTO untuk Windows [4].

Disolder di papan tersebut terdapat sebuah mikrokontroler STM32F407VG yang dikemas dalam LQFP 100-pin (aket *Flat Quad* profil rendah). Mikrokontroler ini berisi inti ARM Cortex-M4 dengan

FPU (*Floating Point Unit*). Beberapa fitur mikrokontroler STM32F407VG adalah :

- 1M Byte Flash memory
- 192k Bytes RAM
- On-chip RC oscillator
- Powered by a single supply of 1.8V to 3.6V
- Up to 168MHz operation
- I/O pins multiplexed with many internal peripherals
- USB OTG HS/FS
- Ethernet
- Static memory controller supporting Compact Flash, SRAM, PSRAM, NOR and NAND memories
- LCD parallel interface

ST-LINK tertanam disertakan pada papan evaluasi STM32F4-Discovery yang dapat digunakan untuk memprogram mikrokontroler STM32F407VG on-board. ST-LINK yang tertanam juga dapat digunakan untuk langkah-tunggal melalui kode sumber dan men-debug mikrokontroler target. Dengan melepas dua *jumper* pada board, ST-LINK yang tertanam dapat digunakan sebagai *programmer* atau *debugger* untuk memprogram atau men-debug mikrokontroler ST pada papan lain melalui kabel dari konektor SWD. ST-LINK yang disematkan terbatas pada pemrograman mikrokontroler STM32F melalui tautan SWD saja. JTAG tidak didukung oleh ST-LINK yang disematkan.

2.6 Nvidia Jetson Nano Developer Kit



Gambar 2.4 Nvidia Jetson Nano Developer Kit [5]

Nvidia Jetson Nano adalah kit pengembang, yang terdiri dari SoM (*System on Module*) dan papan pembawa referensi. Perangkat ini ditargetkan untuk menciptakan sistem tertanam yang membutuhkan daya pemrosesan tinggi untuk *machine learning*, *machine vision*, dan aplikasi pemrosesan video. Jetson Nano ditenagai oleh CPU quad-core ARM A57 1,4 GHz, GPU Nvidia Maxwell 128-core, dan RAM 4 GB. Ini memiliki empat port USB tipe-A, termasuk salah satunya USB 3.0, HDMI dan *DisplayPort* untuk video, dan konektor Ethernet gigabit. Ada slot kamera yang terpasang, namun kita juga dapat menghubungkan kamera melalui USB. Port micro-USB terhubung ke daya, tetapi ada juga konektor barel yang dapat kita gunakan dengan catu daya opsional lebih besar yang menyediakan 4 ampere untuk tugas yang lebih intensif. CPU dilengkapi dengan *heatsink* di atasnya, tetapi kita dapat memasang kipas di atas *heatsink* jika untuk tugas-tugas intensif prosesor yang membutuhkan lebih banyak pendinginan [5].

Tabel 2.1 Spesifikasi Nvidia Jetson Nano Developer Kit

Spesifikasi	Jetson Nano Dev Kit
Harga	99 USD
CPU	64-bit Quad-core ARM A57 (1.43 GHz)
GPU	128-Core Nvidia Maxwell
RAM	4GB DDR4
Connectivity	Ethernet 10/100/1000
Ports	3x USB 2.0, 1x USB 3.0, HDMI, DisplayPort, Camera Connector, M.2
Pins	40-Pin GPIO

2.7 Intel Realsense D435i



Gambar 2.5 Intel Realsense D435i [6]

Kamera kedalaman D435i adalah bagian dari seri kamera Intel® RealSense™ D400, jajaran yang membawa perangkat keras dan perangkat lunak pengindraan kedalaman terbaru Intel dan mengemasnya menjadi produk yang mudah diintegrasikan. Sempurna untuk pengembang, pembuat, dan inovator yang ingin menghadirkan penginderaan mendalam pada perangkat, kamera seri Intel® RealSense™ D400 menawarkan integrasi sederhana dan memungkinkan generasi baru dari solusi cerdas yang dilengkapi dengan penglihatan cerdas. Menambahkan IMU memungkinkan aplikasi untuk meningkatkan kesadaran kedalamannya dalam situasi apa pun di mana kamera bergerak. Ini membuka pintu untuk SLAM yang belum sempurna dan aplikasi pelacakan yang memungkinkan penyelesaian *point cloud* yang lebih baik. Perangkat ini juga memungkinkan peningkatan kesadaran lingkungan untuk robotika dan drone. Penggunaan IMU membuat registrasi dan kalibrasi lebih mudah untuk kasus penggunaan sistem pemindaian genggam dan juga penting dalam bidang-bidang seperti *virtual* atau *augmented reality* dan *drone*. Unit pengukuran inersia (IMU) digunakan untuk mendeteksi gerakan dan rotasi dalam 6 derajat kebebasan (6DoF). IMU menggabungkan berbagai sensor dengan giroskop untuk mendeteksi rotasi dan gerakan dalam 3 sumbu, serta pitch, yaw, dan roll. Kombinasi

bidang pandang yang luas dan sensor rana global pada D435i menjadikannya solusi yang disukai untuk aplikasi seperti navigasi robot dan pengenalan objek. Bidang pandang yang lebih luas memungkinkan satu kamera untuk menutupi lebih banyak area yang menghasilkan lebih sedikit “titik buta”. Sensor rana global menyediakan sensitivitas cahaya rendah yang memungkinkan robot untuk menavigasi ruang dengan lampu mati [6].

Tabel 2.2 Spesifikasi Intel Realsense D435i

Spesifikasi		Intel Realsense D435i
Harga		99 USD
Lingkungan		<i>Indoor/Outdoor</i>
		10 meter
Depth		<i>Active IR Stereo, minimum 0.105 meter, 87°±3° × 58°±1° × 95°±3°, 1280 × 720 active stereo depth resolution 90 fps.</i>
RGB		1920 × 1080 30 fps, 69.4° × 42.5° × 77° (±3°)
Dimensi		90 mm × 25 mm × 25 mm
Penghubung		USB-C 3.1 Gen 1

2.8 Robot Operating System (ROS)

ROS adalah kerangka kerja yang fleksibel untuk menulis perangkat lunak robot. Kerangka kerja ini adalah kumpulan alat, perpustakaan, dan konvensi yang bertujuan untuk menyederhanakan tugas menciptakan perilaku robot yang kompleks dan kuat di berbagai platform robot. Membuat perangkat lunak robot serba guna yang benar-benar kuat sangat

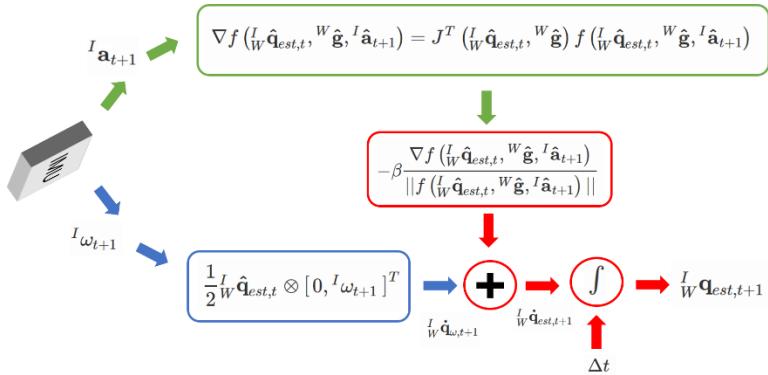
sulit. Dari perspektif robot, masalah yang tampaknya sepele bagi manusia sering sangat bervariasi antara contoh tugas dan lingkungan. Menangani variasi-variasi ini sangat sulit sehingga tidak ada satu individu, laboratorium, atau institusi yang dapat berharap untuk melakukannya sendiri. Sebagai hasilnya, ROS dibangun dari bawah ke atas untuk mendorong pengembangan perangkat lunak robotika kolaboratif. Misalnya, satu laboratorium mungkin memiliki pakar dalam memetakan lingkungan dalam ruangan, dan dapat berkontribusi sistem kelas dunia untuk menghasilkan peta. Kelompok lain mungkin memiliki ahli dalam menggunakan peta untuk menavigasi, dan kelompok lain mungkin telah menemukan pendekatan visi komputer yang bekerja dengan baik untuk mengenali benda-benda kecil dalam kekacauan. ROS dirancang khusus untuk kelompok-kelompok seperti ini untuk berkolaborasi dan membangun karya masing-masing, seperti yang dijelaskan di seluruh situs ini [7].



Gambar 2.6 Logo ROS [7]

2.9 Madgwick Filter

Dalam filter Madgwick, satu-satunya parameter yang dapat diubah adalah trade off parameter β yang menentukan kapan gyro harus mengambil alih accelerometer, pengguna perlu menentukan perkiraan awal dari sikap, bias dan waktu pengambilan sampel. Sikap awal dapat dianggap nol jika perangkat diam atau harus diperoleh dari sumber eksternal seperti sistem tangkap gerak atau kamera. Bias dihitung dengan mengambil rata-rata sampel dengan IMU saat istirahat dan menghitung nilai rata-rata. Bias ini berubah seiring waktu dan filter akan mulai terjadi drift seiring waktu. Waktu pengambilan sampel adalah kebalikan dari frekuensi operasi IMU dan ditentukan secara umum pada driver [8].



Gambar 2.7 Perhitungan matematis madgwick filter [8]

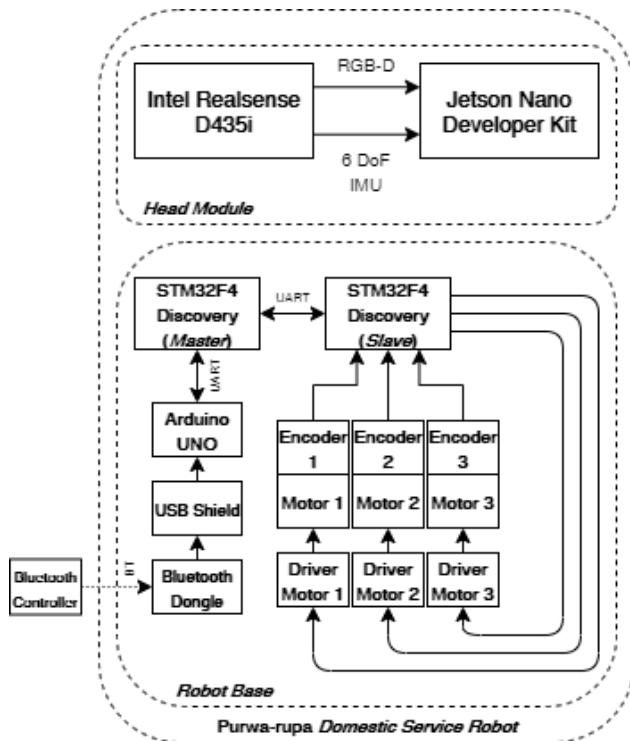
Halaman ini sengaja dikosongkan

BAB 3

PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai rancangan sistem yang dibangun, meliputi rancangan perangkat keras dan rancangan perangkat lunak. Rancangan perangkat keras meliputi rancangan mekanik dan rancangan elektronik. Sedangkan rancangan perangkat lunak meliputi program kendali pada base robot dan program algoritma SLAM yang digunakan pada *head module*.

3.1 Diagram Blok Sistem

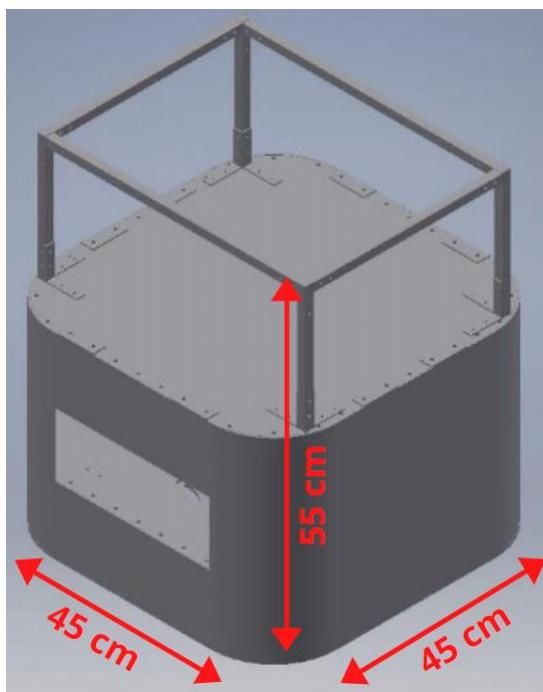


Gambar 3.1 Diagram blok keseluruhan sistem

Pada gambar 3.1 terlihat diagram blok keseluruhan sistem. Secara umum sistem ini memiliki 4 masukan. Masukan pertama adalah perintah arah dari PS3 Bluetooth controller melalui bluetooth yang digunakan untuk menghasilkan pertintah kecepatan guna menggerakkan robot secara manual. Masukan kedua hingga keempat adalah data RGB, *depth*, dan 6 DoF IMU dari Intel Realsense D435i yang akan diolah oleh Jetson Nano Developer Kit untuk menghasilkan keluaran berupa peta. Keluaran dari sistem yang dirancang ini adalah sebuah peta dari lingkungan dimana domestic service robot tersebut beroperasi. Terdapat dua jenis peta yang akan dihasilkan yaitu peta 3 dimensi berbentuk point cloud dan peta 2 dimensi hasil konversi dari peta 3 dimensi sebelumnya.

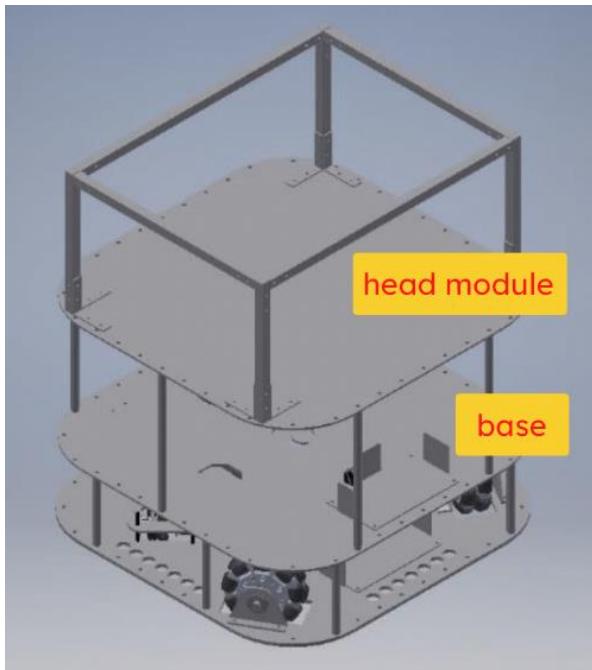
3.2 Rancangan Perangkat Keras

3.2.1 Rancangan Mekanik



Gambar 3.2 Model 3 dimensi kerangka robot tampak depan

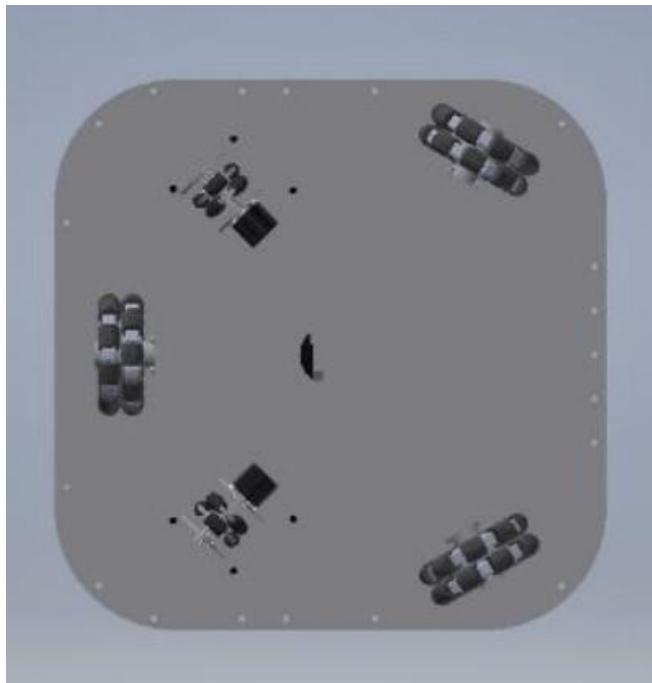
Pada gambar 3.2 terlihat model 3 dimensi kerangka robot tampak depan. Dimensi kerangka robot yang dirancang adalah 45x45x55 cm. Pada perancangan kerangka ini memperhatikan jumlah dan dimensi perangkat elektronik yang akan diletakkan pada kerangka tersebut.



Gambar 3.3 model 3 dimensi kerangka robot tanpa pelindung

Pada gambar 3.3 terlihat kerangka model tanpa pelindung sehingga memperlihatkan bagian dalam robot tersebut. Kerangka robot tersebut terbagi menjadi dua bagian utama, yaitu *head module* dan *base*. Terdapat kerangka kubus pada bagian *head module* yang akan digunakan untuk meletakkan kamera Realsense D435i. Jetson Nano Developer Kit akan diletakkan pada bagian bidang kosong pada head module. Pada kerangka *base*, akan diletakkan berbagai perangkat elektronik lainnya seperti STM32F4Discovery Board, Arduino UNO, driver motor, motor,

roda, dan berbagai alat elektronik lainnya yang berguna sebagai sistem penggerak robot tersebut.



Gambar 3.4 Model 3 dimensi kerangka robot tampak bawah

Bagian bawah model robot memperlihatkan tiga roda yang digunakan oleh robot tersebut. Robot ini menggunakan tiga buah roda *omnidirectional* yang membuat robot tersebut dapat bergerak kesegala arah tanpa menggunakan sistem kemudi.

3.2.2 Rancangan Elektronik

Gambar 3.5 menunjukkan rancangan elektronik *head module* robot. Pada bagian ini tedapat Jetson Nano Developer Kit yang menjalankan algoritma pembentukan peta dan lokalisasi. Dalam menjalankan algortima ini dibutuhkan data RGB-D dan 6 DoF IMU yang diperoleh dari Intel Realsense D435i. Jetson Nano Developer Kit dan Intel

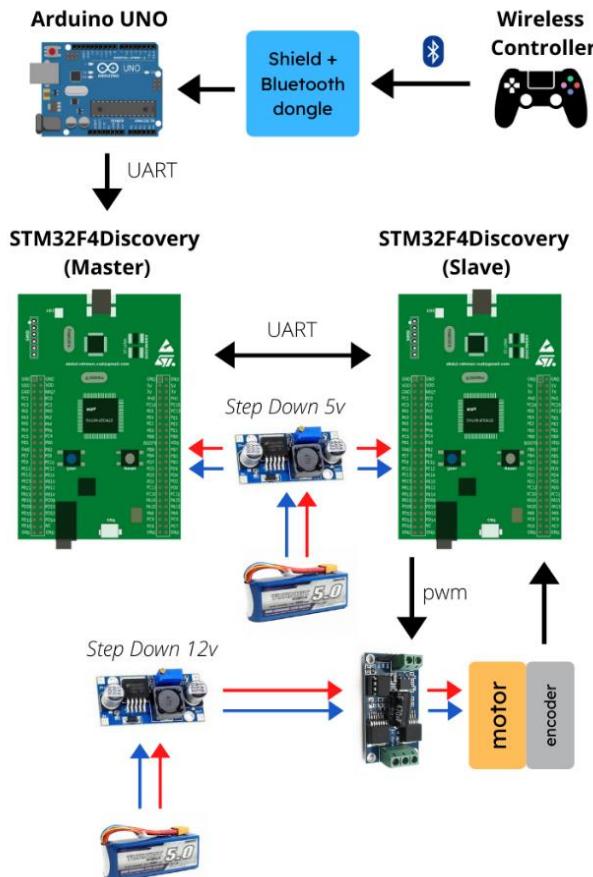
Realsense D435i terhubung oleh kabel *male* USB tipe A ke USB tipe C. Kabel ini digunakan sebagai jalur mengirim data ke Jetson Nano Developer Kit serta memberikan catu daya untuk Intel Realsense D435i. Sebuah baterai lipo digunakan untuk memberikan catu daya kepada Jetson Nano Developer Kit. Untuk menurunkan tegangan yang diterima oleh Jetson Nano Developer Kit sebuah *buck converter* digunakan. Pada *buck converter* ini kita juga bisa memeriksa tegangan pada baterai.



Gambar 3.5 Rancangan elektronik pada *head module* robot

Pada gambar 3.6 terlihat rancangan elektronik sistem pada base robot yang digunakan. Terdapat dua mikrokontroller utama yaitu STM32F4-Discovery yang terhubung secara UART sebagai slave dan master. STM32F4Discovery yang dijadikan master terhubung dengan sebuah Arduino UNO secara UART. Arduino UNO yang digunakan ini terhubung dengan sebuah USB shield yang berguna untuk menghubungkan Arduino UNO tersebut dengan sebuah bluetooth dongle. Sebuah koneksi bluetooth dibutuhkan untuk menghubungkan Arduino UNO dengan wireless controller yang digunakan sebagai pengendali arah robot. STM32F4Discovery yang menjadi slave terhubung kepada driver

motor dan memiliki tugas sebagai pemberi sinyal pwm kepada driver motor tersebut. Kemudian encoder pada motor itu akan memberikan datanya kepada slave itu juga. Dua baterai lipo digunakan sebagai catu daya yang terhubung pada dua buah buck converter. Penggunaan dua buah buck converter ini dilakukan karena sistem membutuhkan dua tegangan yang berbeda, yaitu 5 volt dan 12 volt.

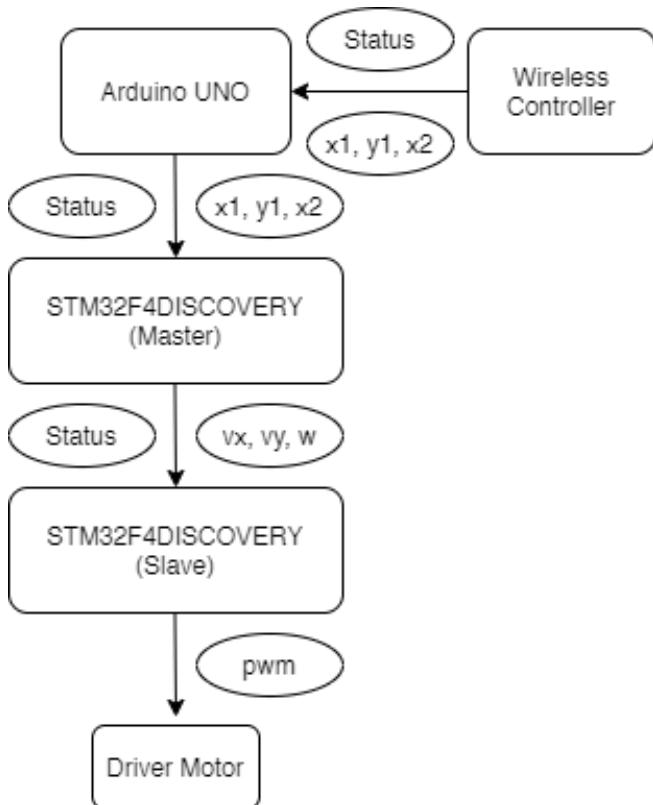


Gambar 3.6 Rancangan elektronik pada *base* robot

3.3 Perancangan Perangkat Lunak

Perangkat lunak yang penulis rancang pada penilitian kali ini terbagi menjadi dua bagian. Pertama, penulis merancang program yang digunakan untuk mengendalikan *base domestic service robot* secara manual. Kedua, penulis melanjutkan dengan perancangan algoritma yang digunakan untuk menghasilkan keluaran peta.

3.3.1 Program Kendali Manual *Base Robot*



Gambar 3.7 Diagram blok aliran data program kendali manual

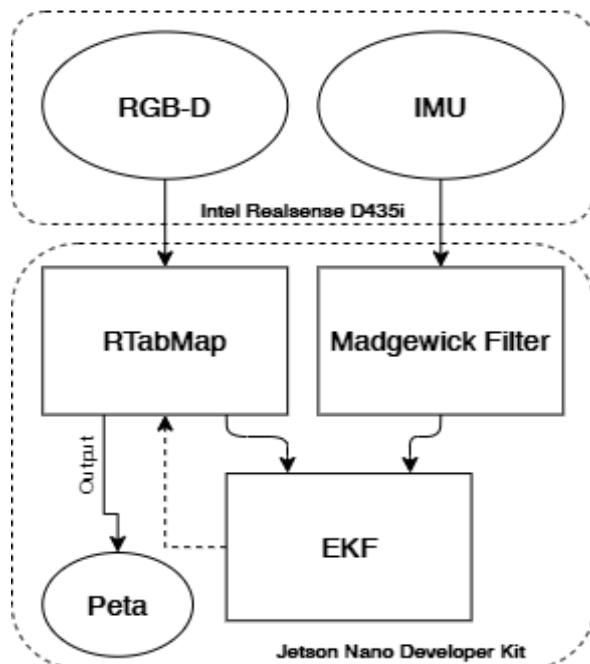
Terlihat pada gambar 3.7 diagram blok aliran data program kendali manual pada *base* robot. Pengguna memberikan masukan arah melalui sebuah *wireless controller*. *Wireless controller* ini berkomunikasi melalui *bluetooth* dengan Arduino UNO dan mengirimkan empat data kepada Arduino Uno tersebut. Keempat data tersebut adalah x_1 , y_1 , x_2 , dan status. Data x_1 dan y_1 merupakan koordinat x dan y dari tombol L3, sedangkan x_2 merupakan koordinat x dari tombol R3. Ketiga data tersebut memiliki nilai -255 hingga 255 bergantung pada arah tombol tersebut digerakkan. Sedangkan data status merupakan data dari tombol *play* yang digunakan untuk menghentikan pergerakan robot ketika data *play* bernilai 0, dan akan bergerak ketika bernilai 1. Kemudian Arduino Uno akan mengirimkan data-data tersebut menuju STM32F4Discovery yang bertindak sebagai *master*. Di sini, data x_1 , y_1 , dan x_2 akan diubah menjadi vx , vy , dan w . Setelah diubah, keempat data tersebut dikirim ke STM32F4Discovery yang bertindak sebagai *slave*. Data vx , vy , dan w ini akan dijadikan *set point* kontrol kecepatan setiap motor dengan *feedback* dari *encoder* masing-masing motor. Keluaran dari kontrol kecepatan tersebut adalah sinyal *pwm* yang akan dikirim ke masing-masing driver motor.



Gambar 3.8 *Wireless controller* yang digunakan

3.3.2 Algoritma Lokalisasi dan Pemetaan

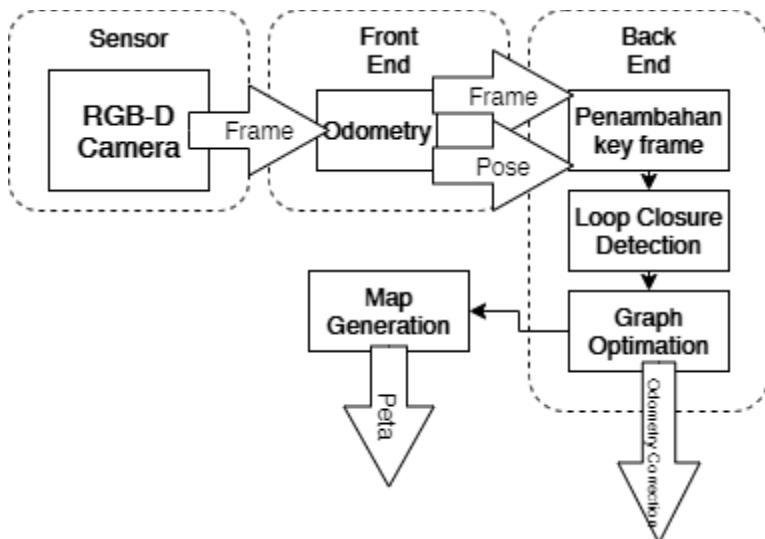
Diagram blok pada gambar 3.9 menggambarkan alur algoritma pembentukan peta. Terdapat dua masukan pada algoritma ini yaitu data RGB-D dan data IMU yang didapatkan dari Intel Realsense D435i. Data tersebut kemudian diolah pada Jetson Nano Developer Kit. Data RGB-D diolah oleh algoritma *open source Real-Time Appearance-Based Mapping* (RtabMap) [9]. Data *accelerometer* dan *gyroscope* dari IMU kemudian akan diolah menggunakan Madgewick Filter untuk mendapatkan estimasi *pose* dari robot. Odometri dari RtabMap dan data IMU hasil dari madgewick filter akan dilewaskan sebuah *extended Kalman Filter*. Dengan ini hasil akhir akan diperbaiki secara berkala oleh IMU dan akan terus diperbaiki juga oleh data odometri visual dari RtabMap.



Gambar 3.9 Diagram blok algoritma lokalisasi dan pemetaan

3.3.3 Algoritma RTabMap

RTabMap adalah pendekatan SLAM berbasis RGB-D, Stereo, dan Lidar berdasarkan detektor penutupan loop berbasis penambahan-penampilan. Detektor penutupan loop menggunakan pendekatan bag-of-words untuk menentukan seberapa besar kemungkinan gambar baru berasal dari lokasi sebelumnya atau lokasi baru. Ketika hipotesis penutupan loop diterima, kendala baru ditambahkan ke grafik peta, kemudian pengoptimal grafik meminimalkan kesalahan dalam peta. Pendekatan manajemen memori digunakan untuk membatasi jumlah lokasi yang digunakan untuk deteksi penutupan loop dan optimisasi grafik, sehingga kendala waktu nyata pada lingkungan skala besar selalu dipenuhi.



Gambar 3.10 Diagram blok algoritma RtabMap

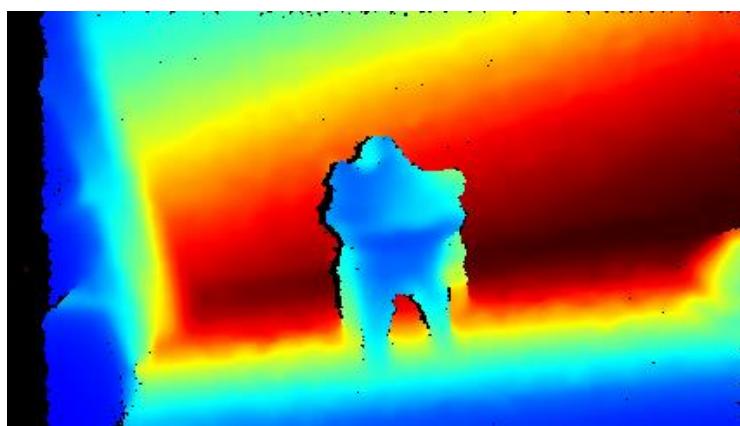
Algoritma ini menggunakan gambar kedalaman dengan RGB untuk membangun peta. Grafik dibuat di sini, di mana setiap *node* berisi RGB dan gambar kedalaman dengan pose odometri yang sesuai. *Linknya* adalah transformasi antara setiap *node*. Ketika grafik diperbarui, RTAB-Map membandingkan gambar baru dengan semua yang sebelumnya

dalam grafik untuk menemukan penutupan *loop*. Ketika penutupan *loop* ditemukan, optimisasi grafik dilakukan untuk memperbaiki pose dalam grafik. Untuk setiap node dalam grafik, dihasilkan *point cloud* dari gambar RGB dan kedalaman. *Point cloud* ini ditransformasikan menggunakan pose dalam node, peta 3D kemudian dibuat [10].

3.4 Pengambilan Data Berbasis Informasi Visual dan IMU



Gambar 3.11 Data RGB



Gambar 3.12 Data depth

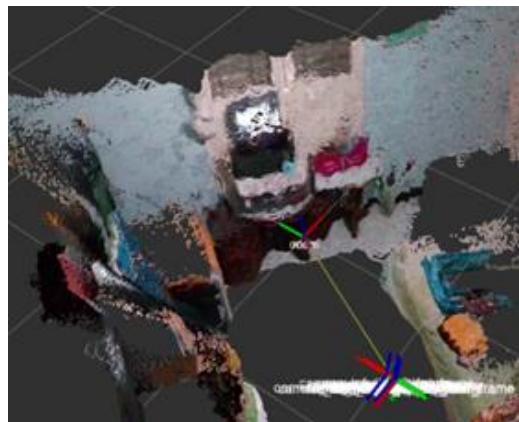
Pengambilan data merupakan tahapan pertama yang penting pada sistem ini. Data utama yang diambil menggunakan kamera Realsense D435i merupakan data yang berisikan informasi visual. Data-data tersebut berupa informasi visual warna dan data depth. Data ini akan digunakan sebagai masukan program algoritma SLAM yang digunakan untuk melakukan pemetaan dan lokalisasi. Gambar 3.11 dan 3.12 menunjukkan visualisasi dari data-data tersebut. Resolusi dari data RGB dan depth yang diterima dari kamera Realsense D435i ini memiliki resolusi 640x480 pixel pada 30 fps. Bila dilihat pada gambar 3.11 dan 3.12, kedua gambar tersebut terlihat tidak terlihat sama. Gambar data RGB terlihat lebih dekat dari pada gambar data depth. Hal ini disebabkan kedua sensor yang digunakan untuk mendapatkan data tersebut memiliki field of view yang berbeda. Untuk mengatasi masalah ini maka akan dilakukan alignment pada dua data tersebut sehingga kedua data tersebut memiliki prespektif yang sama. Selain itu akan juga diambil data IMU yang akan diproses untuk membantu memberikan pose robot. Proses pengambilan data tersebut, pengolahan, hingga proses *alignment* akan dilakukan menggunakan library Intel Realsense SDK 2.0 (ROS wrapper).

3.5 Proses Pemetaan dan Lokalisasi

Pemetaan dan lokalisasi pada penelitian ini akan menggunakan algoritma SLAM berbasis visual dengan masukan data citra RGB-D dan IMU. Teknik SLAM ini digunakan untuk membentuk peta pada sebuah lingkungan yang tidak diketahui sebelumnya, lalu memperbarui data peta yang sudah diketahui sebelumnya. Kedua hal tersebut dilakukan sambil menentukan posisi robot pada peta tersebut secara bersamaan. Proses pemetaan dan lokalisasi pada SLAM ini terdiri dari beberapa tahap, yaitu *landmark extraction*, *data association*, *state estimation*, *state update*, *landmark update*, dan *map generation*. Hal pertama yang perlu dilakukan adalah mengambil data RGB-D lingkungan sekitar robot untuk mencari *landmark* yang dapat digunakan sebagai acuan. Gambar 3.13 memperlihatkan *landmark* yang tertangkap pada suatu *frame* data. *Landmark* ini akan disimpan dan dicocokkan satu sama lainnya sebagai acuan penentuan posisi dan orientasi lalu dirangkai menjadi sebuah peta yang merepresentasikan lingkungan asli disekitar robot. Gambar 3.14 memperlihatkan hasil pemetaan yang dilakukan serta posisi dan orientasi robot hasil lokalisasi menggunakan algoritma SLAM berbasis visual RtabMap. Peta tersebut merupakan peta tiga dimensi berupa *point cloud* yang bisa diolah menjadi peta dua dimensi setelah proses selesai.



Gambar 3.13 Landmark yang tertangkap pada suatu *frame* data



Gambar 3.14 Peta yang terbentuk dengan posisi robot pada peta

Halaman ini sengaja dikosongkan

BAB 4

PENGUJIAN DAN ANALISIS

Untuk mengetahui hasil dari sistem yang telah dirancang maka perlu dilakukan pengujian. Pada tahap ini akan terlihat apakah tujuan dari penelitian kali ini tercapai. Pengujian ini akan dilanjutkan dengan analisis untuk menghasilkan kesimpulan yang dapat memperbaiki hasil penelitian selanjutnya. Proses pengujian pada penelitian ini terbagi menjadi dua, yaitu pengujian *base* robot dan pengujian algoritma SLAM menggunakan RtabMap dengan tujuan pembentukan peta 3 dimensi lingkungan sekitar robot.

4.1 Pengujian Pada Base Robot

Pengujian pertama yang dilakukan adalah pengujian pada *base robot*. *Base* robot digunakan sebagai penggerak badan robot dengan menggunakan tiga buah motor dan tiga buah roda *omnidirectional*. Penggunaan roda *omnidirectional* ini membuat robot bisa bergerak ke segala arah tanpa menggunakan sebuah sistem kemudi. Pergerakan robot akan merubah posisi dan orientasi dari robot tersebut, sehingga dapat bergerak menjelajahi sebuah ruangan dan mengambil data citra RGB dan *depth* dari setiap sudut ruangan tersebut.

4.1.1 Pengujian Pergerakan Robot



Gambar 4.1 Xiaomi Duka

Pengujian ini dilakukan untuk menguji apakah sistem pergerakan yang dimiliki robot sudah sesuai dengan perancangan yang dilakukan sebelumnya. Sesuai rancangan, karena robot yang digunakan adalah sebuah *three wheeled omnidirectional robot* maka seharusnya robot dapat bergerak ke segala arah tanpa menggunakan sistem kemudi. Robot akan dikendalikan menggunakan sebuah *wireless controller* menuju berbagai

arah, lalu hasil pergerakan robot tersebut kemudian akan dicatat sebagai hasil pengujian. Pengukuran jarak akan dilakukan dengan menggunakan sebuah pengukur jarak berbasis laser, yaitu Xiaomi Duka, yang bisa dilihat pada gambar 4.1.



Gambar 4.2 Tempat pengujian *base* robot

a. Gerak Maju



Gambar 4.3 Pemberian perintah gerak maju pada *controller*



Gambar 4.4 Pose awal dan pose setelah bergerak maju sejauh 3 meter

Perintah gerak maju diberikan dengan menggerakkan tombol L3 pada *wireless controller* ke arah atas. Hal ini akan membuat *controller* mengirimkan data x_1 bernilai positif kepada Arduino UNO melalui komunikasi *bluetooth*. Data x_1 ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data vx yang menentukan sinyal *pwm* yang akan dikirim ke masing-masing *driver motor*.

Tabel 4.1 Hasil pengujian gerak maju

Jarak Gerak (cm)	Penyimpangan
50	0,43 cm ke kiri
100	1,05 cm ke kiri
150	1,6 cm ke kiri
200	2,24 cm ke kiri
250	2,72 cm ke kiri
300	3,35 cm ke kiri

b. Gerak Mundur



Gambar 4.5 Pemberian perintah mundur pada *controller*



Gambar 4.6 Pose awal dan pose setelah bergerak mundur sejauh 3 meter

Perintah gerak mundur diberikan dengan menggerakkan tombol L3 pada *wireless controller* ke arah bawah. Hal ini akan membuat controller mengirimkan data x_1 bernilai negatif kepada Arduino UNO melalui komunikasi bluetooth. Data x_1 ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data v_x yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.

Tabel 4.2 Hasil pengujian gerak mundur

Jarak Gerak (cm)	Penyimpangan
50	0,6 cm ke kiri
100	1,12 cm ke kiri
150	1,63 cm ke kiri
200	2,25 cm ke kiri
250	2,80 cm ke kiri
300	3,47 cm ke kiri

c. Gerak Kanan



Gambar 4.7 Pemberian perintah gerak kanan pada controller

Perintah gerak kanan diberikan dengan menggerakkan tombol L3 pada *wireless controller* ke arah kanan. Hal ini akan membuat controller mengirimkan data y_1 bernilai positif kepada Arduino UNO melalui komunikasi bluetooth. Data y_1 ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data v_y yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.



Gambar 4.8 Pose awal dan pose setelah bergerak ke kanan sejauh 3 meter

Tabel 4.3 Hasil pengujian gerak kanan

Jarak Gerak (cm)	Penyimpangan
50	0,37 cm ke belakang
100	0,89 cm ke belakang
150	1,43 cm ke belakang
200	1,82 cm ke belakang
250	2,47 cm ke belakang
300	2,65 cm ke belakang

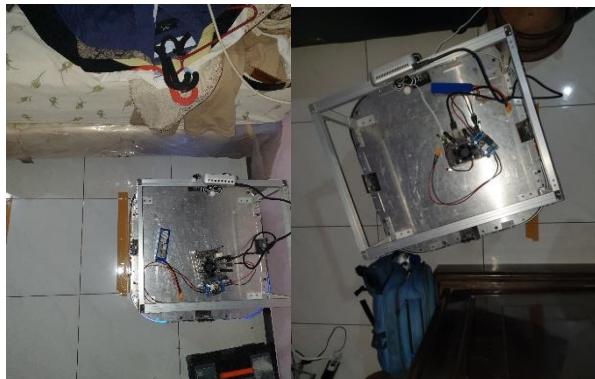
d. Gerak Kiri



Gambar 4.9 Pemberian perintah gerak kiri pada controller

Perintah gerak kiri diberikan dengan menggerakkan tombol L3 pada *wireless controller* ke arah kiri. Hal ini akan membuat controller mengirimkan data y_1 bernilai negatif kepada Arduino UNO melalui

komunikasi bluetooth. Data y_1 ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data v_y yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.



Gambar 4.10 Pose awal dan pose setelah bergerak ke kiri sejauh 3 meter

Tabel 4.4 Hasil pengujian gerak kiri

Jarak Gerak (cm)	Penyimpangan
50	0,41 cm ke belakang
100	0,91 cm ke belakang
150	1,24 cm ke belakang
200	1,75 cm ke belakang
250	2,23 cm ke belakang
300	2,51 cm ke belakang

e. Gerak Diagonal Kanan Depan

Perintah gerak diagonal kanan depan diberikan dengan menggerakkan tombol L3 pada *wireless controller* ke arah diagonal kanan atas. Hal ini akan membuat controller mengirimkan data x_1 dan y_1 bernilai positif kepada Arduino UNO melalui komunikasi bluetooth. Data x_1 dan y_1 ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data v_x dan v_y yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.



Gambar 4.11 Pemberian perintah gerak diagonal kanan depan pada *controller*

Karena keterbatasan ruang pengujian, pengujian gerak diagonal tidak bisa dilakukan sejauh 3 meter seperti pengujian gerak pada arah lainnya. Maka untuk pengujian arah diagonal hanya akan dilakukan pengamatan visual apakah gerak diagonal sudah sesuai. Hasil dari pengujian ini sudah sesuai dengan perintah arah yang diberikan.

f. Gerak Diagonal Kiri Depan



Gambar 4.12 Pemberian perintah gerak diagonal kiri depan pada *controller*

Perintah gerak diagonal kiri depan diberikan dengan menggerakkan tombol L3 pada *wireless controller* ke arah diagonal kiri atas. Hal ini akan membuat controller mengirimkan data $x1$ bernilai positif dan $y1$ bernilai negatif kepada Arduino UNO melalui komunikasi

bluetooth. Data x_1 dan y_1 ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data v_x dan v_y yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.

Karena keterbatasan ruang pengujian, pengujian gerak diagonal tidak bisa dilakukan sejauh 3 meter seperti pengujian gerak pada arah lainnya. Maka untuk pengujian arah diagonal hanya akan dilakukan pengamatan visual apakah gerak diagonal sudah sesuai. Hasil dari pengujian ini sudah sesuai dengan perintah arah yang diberikan.

g. Gerak Diagonal Kanan Belakang



Gambar 4.13 Pemberian perintah gerak diagonal kanan belakang pada *controller*

Perintah gerak diagonal kanan belakang diberikan dengan menggerakkan tombol L3 pada *wireless controller* ke arah diagonal kanan bawah. Hal ini akan membuat controller mengirimkan data x_1 bernilai negatif dan y_1 bernilai positif kepada Arduino UNO melalui komunikasi bluetooth. Data x_1 dan y_1 ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data v_x dan v_y yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.

Karena keterbatasan ruang pengujian, pengujian gerak diagonal tidak bisa dilakukan sejauh 3 meter seperti pengujian gerak pada arah lainnya. Maka untuk pengujian arah diagonal hanya akan dilakukan

pengamatan visual apakah gerak diagonal sudah sesuai. Hasil dari pengujian ini sudah sesuai dengan perintah arah yang diberikan.

h. Gerak Diagonal Kiri Belakang



Gambar 4.14 Pemberian perintah gerak diagonal kiri belakang pada controller

Perintah gerak diagonal kiri belakang diberikan dengan menggerakkan tombol L3 pada *wireless controller* ke arah diagonal kiri bawah. Hal ini akan membuat controller mengirimkan data $x1$ dan $y1$ bernilai negatif kepada Arduino UNO melalui komunikasi bluetooth. Data $x1$ dan $y1$ ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data vx dan vy yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.

Karena keterbatasan ruang pengujian, pengujian gerak diagonal tidak bisa dilakukan sejauh 3 meter seperti pengujian gerak pada arah lainnya. Maka untuk pengujian arah diagonal hanya akan dilakukan pengamatan visual apakah gerak diagonal sudah sesuai. Hasil dari pengujian ini sudah sesuai dengan perintah arah yang diberikan.

i. Gerak Putar Kanan

Perintah gerak putar kanan diberikan dengan menggerakkan tombol R3 pada *wireless controller* ke arah kanan. Hal ini akan membuat controller mengirimkan data w bernilai positif kepada Arduino UNO melalui komunikasi bluetooth. Data w ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data vw yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.



Gambar 4.15 Pemberian perintah gerak putar kanan pada controller



Gambar 4.16 Pose awal dan pose setelah berputar ke kanan sebanyak 6 kali

Tabel 4.5 Hasil pengujian gerak putar kanan

Jumlah Putaran	Penyimpangan
1	5 cm ke kanan dan 3 cm ke depan
2	7 cm ke kanan dan 7 cm ke depan
3	11,5 cm ke kanan dan 6 cm ke depan
4	14 cm ke kanan dan 7 cm ke depan
5	16,5 cm ke kanan dan 9 cm ke depan
6	17 cm ke kanan dan 10 cm ke depan

j. Gerak Putar Kiri



Gambar 4.17 Pemberian perintah gerak putar kiri pada controller



Gambar 4.18 Pose awal dan pose setelah berputar ke kiri sambanyak 6 kali

Perintah gerak putar kiri diberikan dengan menggerakkan tombol R3 pada wireless controller ke arah kiri. Hal ini akan membuat controller mengirimkan data w bernilai negatif kepada Arduino UNO melalui komunikasi bluetooth. Data w ini kemudian akan diolah oleh STM32F4Discovery dan diubah menjadi data vw yang menentukan sinyal pwm yang akan dikirim ke masing-masing driver motor.

Tabel 4.6 Hasil pengujian gerak putar kiri

Jumlah Putaran	Penyimpangan
1	4 cm ke kanan dan 4 cm ke belakang
2	7 cm ke kanan dan 6 cm ke belakang
3	11 cm ke kanan dan 6 cm ke belakang
4	13 cm ke kanan dan 8 cm ke belakang
5	15 cm ke kanan dan 8 cm ke belakang
6	15,5 cm ke kanan dan 9 cm ke belakang

4.1.2 Pengujian Kecepatan Optimal

Beberapa batasan terdapat pada program pemetaan yang dimiliki. Batasan-batasan tersebut antara lain FPS data RGB dan *depth* yang diberikan oleh kamera Intel Realsense D435i relatif rendah, yaitu hanya 30 FPS. Resolusi data RGB dan depth yang dikirim juga hanya 640x480 pixel. Selain itu *refresh rate* IMU yang dimiliki oleh Intel Realsense D435i juga tergolong relatif rendah. Kemampuan komputasi pada Jetson Nano Developer Kit juga terbatas sehingga memerlukan waktu untuk memproses data yang diterima dan mengolahnya menjadi keluaran peta. Maka dari itu, ketika robot bergerak pada kecepatan yang terlalu tinggi, maka proses pemetaan yang dilakukan tidak bisa dilakukan dengan baik. Ketika hal tersebut terjadi, maka akan terjadi *drift* dan *lost tracking*.

Setelah dilakukan percobaan dengan metode *trial and error* pada kecepatan robot dan hasil pemetaan. Didapatkan hasil bahwa kecepatan robot yang optimal adalah 20 cm/detik. Dimana pada kecepatan ini, proses pemetaan dapat dilakukan dengan baik tanpa gangguan dari gerak robot yang terlalu cepat.

Tabel 4.7 Hasil pengujian kecepatan optimal

Kecepatan	Hasil
5 cm / detik	Tidak ada gangguan
10 cm / detik	Tidak ada gangguan
15 cm / detik	Tidak ada gangguan
20 cm / detik	Tidak ada gangguan
25 cm /detik	Terjadi gangguan
30 cm / detik	Terjadi gangguan

4.1.3 Analisa Hasil Percobaan Pada *Base* Robot

Berdasarkan hasil yang didapatkan pada dua pengujian yang dilakukan pada *base* robot, dapat terlihat bahwa sistem gerak robot dapat beroperasi sesuai dengan perintah yang diberikan. Terdapat *error* yang kecil pada posisi dan orientasi akhir robot pada pengujian gerak robot. Hal ini menunjukkan bahwa sistem penggerak yang ada pada robot ini tidak sempurna. Kemungkinan terdapat perbedaan karakteristik mekanik pada tiga roda yang digunakan atau tiga motor yang digunakan tidak memiliki respon yang identik terhadap masukan yang diberikan. Hal ini menyebabkan bila robot tersebut digunakan dengan sistem kendali otonom, maka harus dilakukan koreksi posisi dan orientasi pada setiap pergerakannya. Hal ini bisa dilakukan dengan memanfaatkan *feedback* dari *wheel odometry*, IMU, *visual odometry*, dan sensor lainnya. Sementara itu hasil pengujian kecepatan optimal robot menunjukkan bahwa program pemetaan yang digunakan pada sistem yang dirancang ini memiliki hanya dapat beroperasi pada kecepatan robot tertentu. Bila Kecepatan robot melebihi batasan tersebut, maka hasil pemetaan yang didapatkan akan tidak sempurna atau bahkan tidak bisa dilakukan sama sekali.

4.2 Pengujian Kamera RGB-D Intel Realsense D435i

4.2.1 Kalibrasi IMU

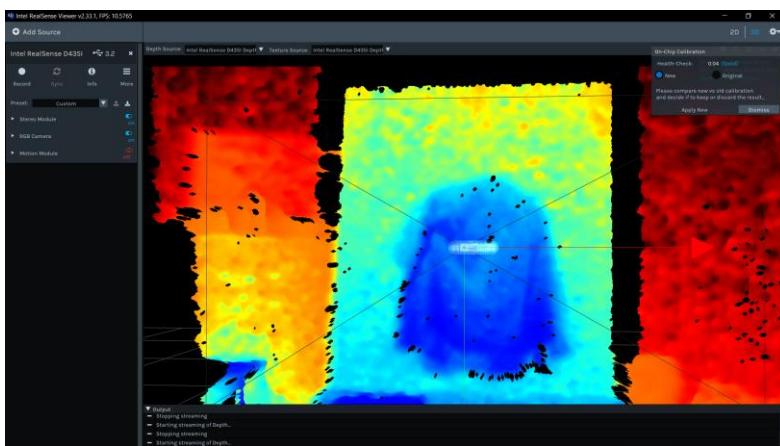
Untuk menentukan penyimpangan kebenaran nilai pada data IMU dan menjamin hasil-hasil pengukuran sesuai dengan standar, maka dilakukan proses kalibrasi IMU pada Intel Realsense D435i. Proses kalibrasi ini dilakukan secara otomatis menggunakan kode yang dirancang dalam bahasa pemrograman python. Program ini akan mengukur dan mencatat nilai data IMU pada posisi-posisi tertentu. Setelah data pada posisi tersebut didapatkan, maka data tersebut akan dikirim ke Intel Realsense D435i sebagai acuan baru untuk menyesuaikan pengukuran data IMU. Program dan hasil kalibrasi dapat dilihat pada bagian lampiran.



Gambar 4.19 Posisi kalibrasi IMU

4.2.2 Kalibrasi *depth*

Kalibrasi *depth* pada Intel Realsense D435i dilakukan secara otomatis dengan menggunakan program Intel Realsense Viewer. Kalibrasi *on-chip* Intel RealSense memungkinkan menjalankan pemeriksaan kesehatan yang tepat pada sistem, untuk memastikan sensor bekerja secara optimal. Data *depth* kamera dapat dipulihkan ke performa mendekati sempurna. Hal ini dapat dilakukan tanpa target, dinding pola *checker*, dan pengaturan rumit seperti jalur gerak khusus. Proses kalibrasi ini sangat sederhana, cukup panggil fungsi *on-chip*. Semua fitur ini disematkan pada *firmware vision processor* kamera, tanpa membutuhkan kemampuan komputasi dari perangkat yang terhubung ke kamera tersebut. Jika proses pemeriksaan kesehatan menentukan bahwa data kalibrasi baru lebih unggul daripada yang lama, data kalibrasi yang baru dan lebih baik kemudian akan ditulis langsung ke memori *flash*.



Gambar 4.20 Kalibrasi *on-chip* menggunakan Intel Realsense Viewer

4.2.3 Pengujian *depth*

Untuk mengetahui kualitas data *depth* yang akan digunakan sebagai data masukan pada program pembentukan peta, maka akan dilakukan pengujian terhadap data *depth* pada kamera Intel Realsense D435i yang digunakan. Pengukuran akan dilakukan dengan menggunakan Xiaomi Duka sebagai data bandingan.

Tabel 4.8 Hasil pengujian *depth*

Jarak Asli (cm)	Jarak Terbaca (cm)	Error (%)
20	20,09	0,45
40	40,29	0,72
60	61,16	1,93
80	80,54	0,67
100	100,23	0,23
120	121,27	1,05
140	140,25	0,17
160	162,23	1,39
180	181,27	0,70
200	202,26	1,13
220	221,81	0,82
240	242,22	0,92

260	261,85	0,71
280	283,26	1,16
300	302,27	0,75
Rata-rata		0,85

4.2.4 Analisa Hasil Pengujian Kamera Intel Realsense D435i

Setelah dilakukan proses kalibrasi pada kamera Intel Realsense D435i Terlihat pada tabel 4.8 hasil pengujian *depth*. Terdapat eror pada data pembacaan *depth* Intel Realsense D435i. Semakin jauh bendanya maka error yang terjadi juga semakin besar. Hal ini sesuai dengan data yang tercantum pada dokumen resmi Intel Realsense D435i [11]. Eror pada pembacaan *depth* ini akan mempengaruhi hasil pemetaan yang dilakukan. Rasio peta dan lingkungan aslinya akan terpengaruh karena jarak yang didapatkan oleh Intel Realsense D435i berbeda dengan aslinya.

4.3 Pengujian Algoritma RtabMap

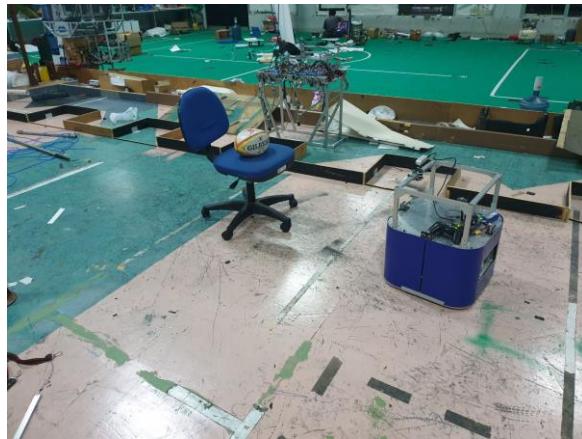
4.3.1 Pengujian Bentuk Benda

Untuk menguji kemampuan algoritma RtabMap dalam menangkap karakteristik lingkungan disekitarnya dengan baik, akan dilakukan pengujian terhadap beberapa benda yang memiliki bentuk berbeda-beda. Benda tersebut akan dimodelkan menggunakan algoritma RtabMap menjadi model tiga dimensi berupa *point cloud*. Kemudian akan dibandingkan dengan bentuk aslinya.

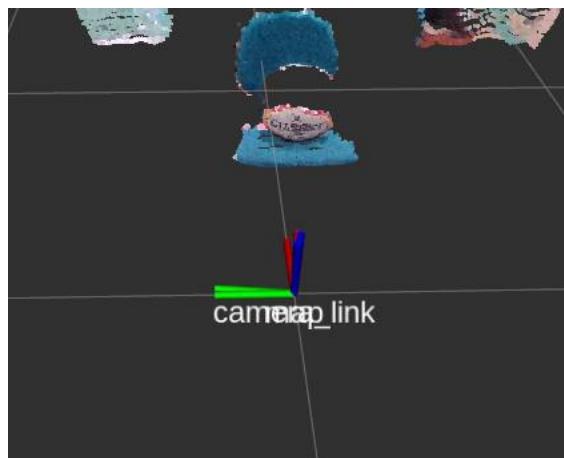


Gambar 4.21 Benda A

Benda A merupakan sebuah bola rugby berbentuk elips dengan *major axis* sebesar 25 cm dan *minor axis* sebesar 12,5 cm. Benda A ini kemudian diletakkan pada sebuah kursi dan robot akan mengamati benda tersebut dari jarak sekitar 50 cm.



Gambar 4.22 Posisi benda A dan robot saat pengamatan



Gambar 4.23 Hasil pemodelan Benda A

Hasil pemodelan benda A dapat dilihat pada gambar 4.23. Bentuk elips yang dimiliki oleh benda A dapat ditangkap dengan cukup baik sesuai dengan karakteristik aslinya.

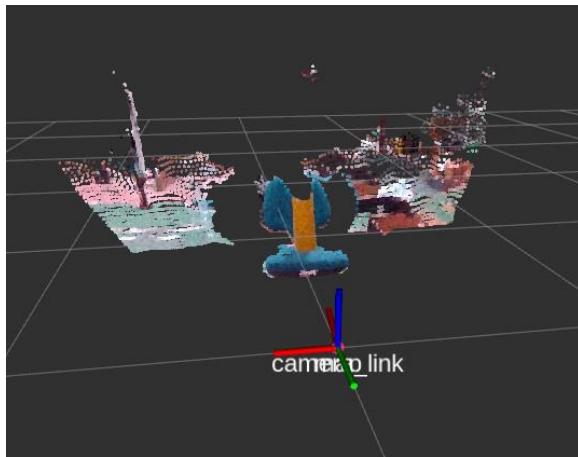


Gambar 4.24 Benda B

Benda B merupakan sebuah tabung silinder besi berdiameter 11 cm dengan tinggi 30 cm. Benda C ini kemudian diletakkan pada sebuah kursi dan robot akan mengamati benda tersebut dari jarak sekitar 50 cm.



Gambar 4.25 Posisi benda B dan robot saat pengamatan



Gambar 4.26 Hasil pemodelan benda B

Hasil pemodelan benda B dapat dilihat pada gambar 4.26. Bentuk tabung silinder yang dimiliki oleh benda B dapat ditangkap dengan cukup baik sesuai dengan karakteristik aslinya.

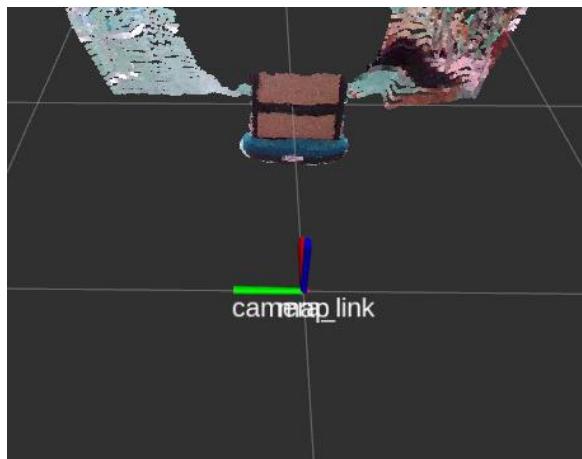


Gambar 4.27 Benda C

Benda C merupakan sebuah kardus dengan dimensi 37,5x30x35 cm. Benda C ini kemudian diletakkan pada sebuah kursi dan robot akan mengamati benda tersebut dari jarak sekitar 50 cm.

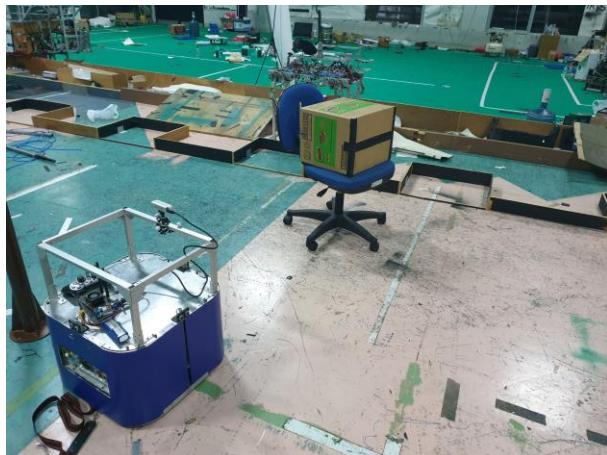


Gambar 4.28 Posisi benda C dan robot saat pengamatan

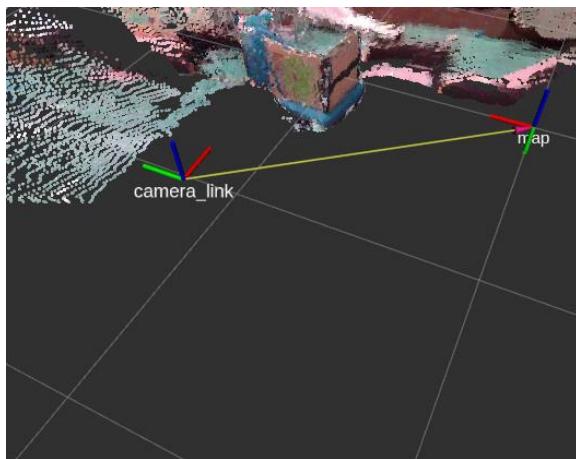


Gambar 4.29 Hasil pemodelan benda C

Hasil pemodelan benda C dapat dilihat pada gambar 4.29. Bentuk persegi yang dimiliki oleh benda C dapat ditangkap dengan cukup baik sesuai dengan karakteristik aslinya.



Gambar 4.30 Posisi benda C dan robot setelah robot bergerak

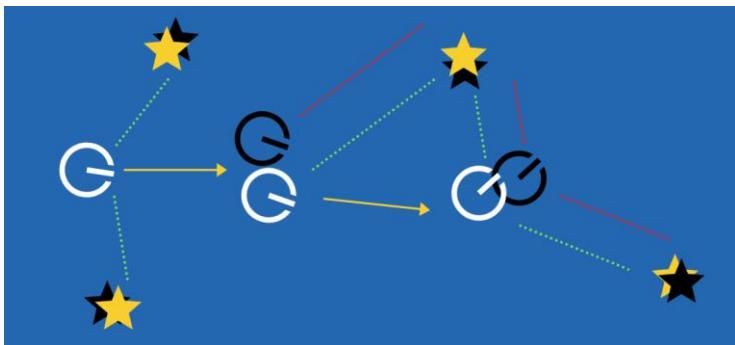


Gambar 4.31 Hasil pemodelan benda C setelah robot bergerak

Kemudian robot bergerak disekitar benda C tersebut untuk mendapatkan model benda C dari prespektif yang berbeda. Bisa dilihat posisi robot setelah bergerak pada gambar 4.30 dan hasil pemodelannya pada gambar 4.31. Setelah bergerak, robot dapat memodelkan benda B

dari sisi lainnya sehingga bentuk tiga dimensi dari benda tersebut terlihat. Terlihat pula posisi dan orientasi robot relatif terhadap titik awal ketika pemetaan dimulai.

4.3.2 Ilustrasi Pengujian Pemetaan



Gambar 4.32 Ilustrasi proses pemetaan

Pada gambar 4.32 terlihat ilustrasi proses pemetaan yang akan dilakukan. Benda bulat pada ilustrasi tersebut menggambarkan robot dan garis didalamnya merepresentasikan posisi dan orientasi dari robot tersebut. Bintang pada ilustrasi tersebut menggambarkan fitur atau *landmark* lingkungan di sekitar robot yang tertangkap oleh kamera Intel Realsense D435i. Landmark ini dapat berdasarkan warna, kontur, hingga *depth*. Garis putus menandakan bahwa robot tersebut dapat melakukan observasi terhadap *landmark* tersebut. Sedangkan garis kuning dengan tanda panah menandakan pergerakan robot tersebut.

Awalnya robot akan menangkap *landmark* yang terlihat pada kamera yang ia miliki. Lalu dilakukan pencocokan antar *landmark* yang ditangkap tersebut. Bersamaan dengan gerak robot, *landmark* yang telah terlihat itu akan dirangkai satu sama lainnya membentuk sebuah peta secara keseluruhan. Disaat yang sama, *landmark* tersebut akan digunakan sebagai acuan posisi dan orientasi robot tersebut. Untuk mendapatkan hasil yang lebih *robust* maka sebuah IMU digunakan juga sebagai data tambahan.

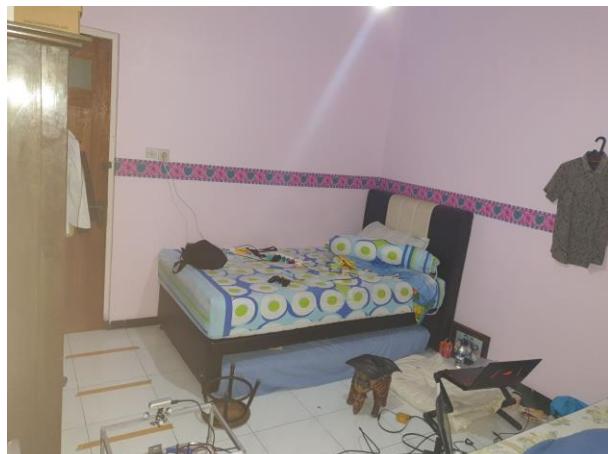
4.3.3 Tempat Pengujian Pemetaan

Pada pengujian ini dilakukan pemetaan dan lokalisasi menggunakan algoritma SLAM visual RtabMap pada tiga ruangan yang

berbeda. Ruangan-ruangan tersebut dapat dilihat pada gambar 4.33, 4.34, dan 4.35.



Gambar 4.33 Ruang A

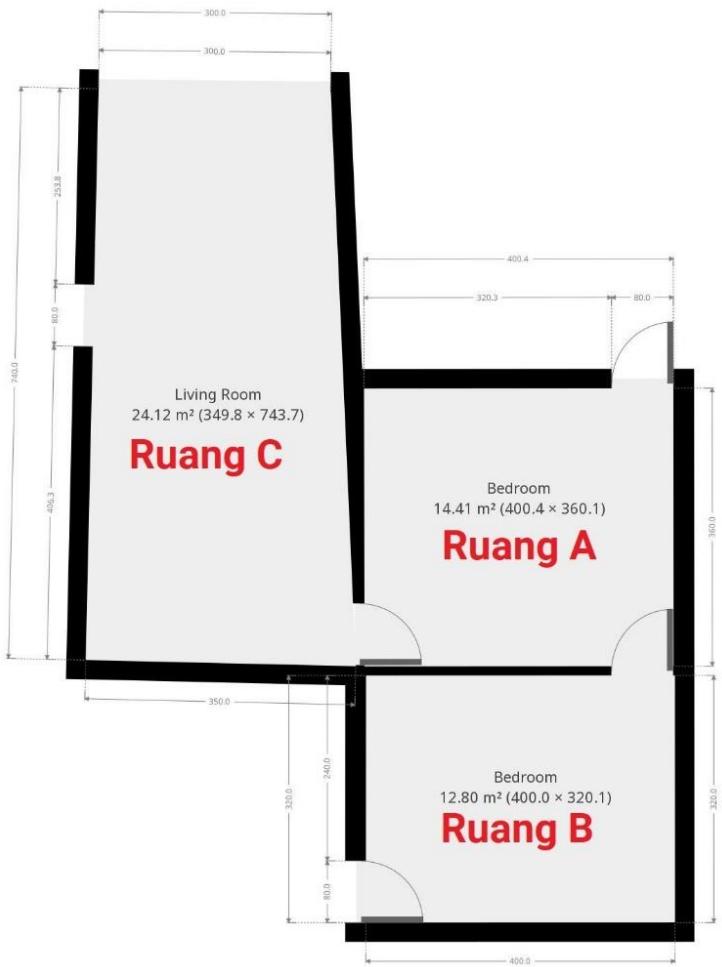


Gambar 4.34 Ruang B



Gambar 4.35 Ruang C

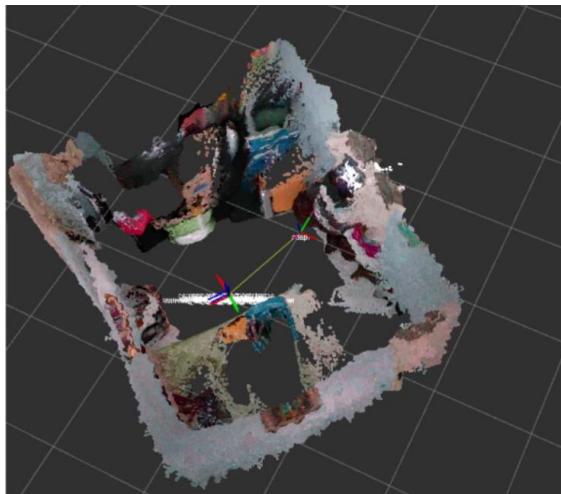
Pada gambar 4.36 terlihat denah dari tiap ruangan yang digunakan. Ruang B merupakan sebuah kamar tidur dengan dimensi 400x360 cm. Pada sekeliling ruang tersebut terdapat wallpaper pada tembok ruangan tersebut yang terlihat identik. Ruang A juga merupakan sebuah kamar tidur yang tidak dipakai sehingga dijadikan gudang tempat penyimpanan barang dengan dimensi 400x320 cm. Sedangkan ruang C merupakan ruang keluarga dengan dimensi 350x745 cm dan tidak berbentuk persegi yang sempurna. Sedangkan ruang A dan B berbentuk persegi panjang. Robot akan bergerak pada ketiga ruang tersebut lalu akan dilakukan analisa terhadap hasil peta dari ruang yang didapat. Bentuk dari ruangan, benda, serta halangan yang terdapat pada ruangan tersebut akan dibandingkan dengan model peta tiga dimensi yang dihasilkan oleh sistem. Karena robot tidak bisa bergerak bebas melalui semua ruangan, maka proses pemetaan akan dilakukan secara terpisah untuk tiap ruangan. Enam buat peta akan dihasilkan dari tiga buah ruangan tersebut. Sebuah peta 3 dimensi dan sebuah peta 2 dimensi untuk tiap ruangan tersebut.



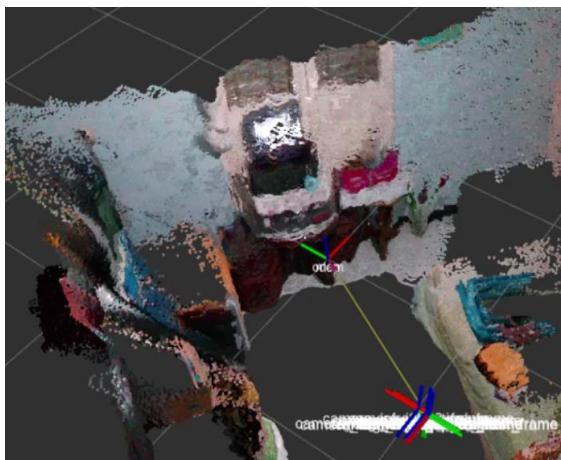
Gambar 4.36 Denah ruang A, B, dan C

4.3.4 Hasil Pengujian Pemetaan

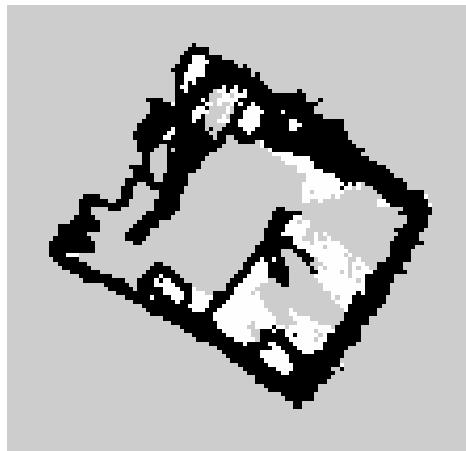
a. Hasil Ruangan A



Gambar 4.37 Hasil pemetaan dan lokalisasi pada ruangan A

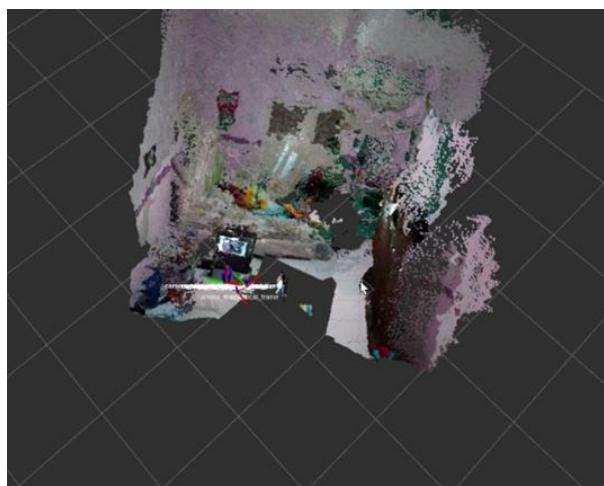


Gambar 4.38 Hasil pada ruangan A dari sisi yang berbeda



Gambar 4.39 Konversi peta ruangan A menjadi 2 dimensi

b. Hasil Ruangan B

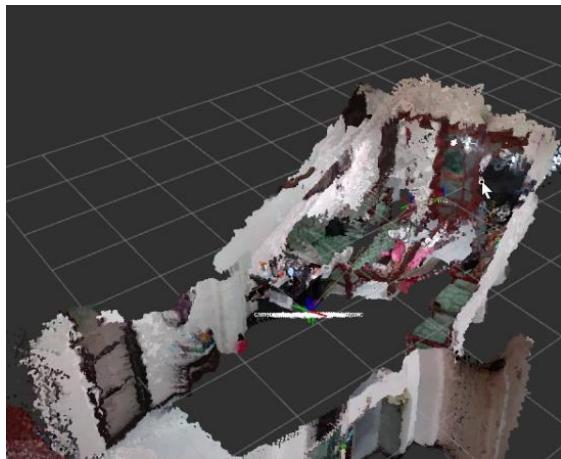


Gambar 4.40 Hasil pemetaan dan lokalisasi pada ruangan B

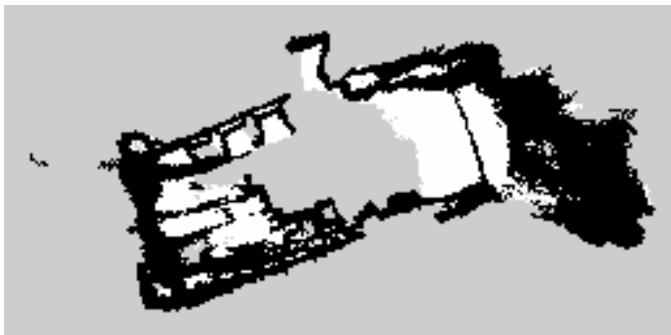


Gambar 4.41 Konversi peta ruangan B menjadi dua dimensi

c. Hasil Ruangan C



Gambar 4.42 Hasil pemetaan dan lokalisasi pada ruangan C



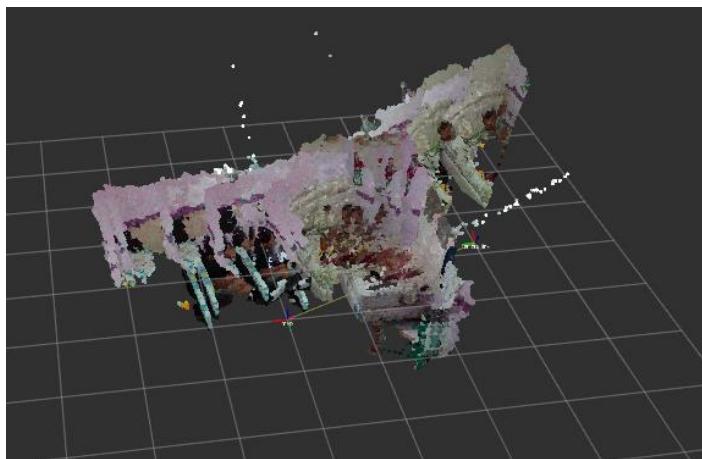
Gambar 4.43 Konversi peta ruangan C menjadi dua dimensi

4.3.5 Analisa Hasil Pengujian RTabMap

Pengujian bentuk benda yang dilakukan menunjukkan bahwa karakteristik bentuk benda dapat ditangkap dengan baik sesuai dengan karakteristik aslinya. Pengujian pada ruangan A menghasilkan peta yang cukup baik. Lingkungan pada ruangan A memiliki *landmark* visual yang memadai sehingga proses pemetaan serta lokalisasi dapat berjalan dengan lancar. Permasalahan terjadi saat terdapat cermin atau kaca pada lingkungan tersebut. Data *depth* yang ditangkap oleh kamera Intel Realsense D435i menjadi kacau ketika terdapat cermin atau kaca. Sehingga pada saat pengujian, seluruh kaca dan cermin ditutup dengan kain agar mendapatkan hasil yang baik. Pemetaan pada ruangan B tidak bisa dilakukan secara penuh. Hal ini dikarenakan landmark yang terdapat pada ruangan ini terlalu identik satu sama lainnya. Terdapat wallpaper pada dinding sekeliling ruangan yang terlihat identik. Gambar 4.44 memperlihatkan landmark pada wallpaper yang terdeteksi pada dinding ruangan ini. Hal ini menyebabkan terjadinya lost tracking yang bisa dilihat pada gambar 4.45. Pengujian pada ruangan C menunjukkan hasil yang serupa dengan pengujian pada ruangan A. Ruangan C yang relatif lebih besar dari ruangan lainnya mengakibatkan kesalahan pembacaan data *depth* yang terakumulasi terlihat sehingga bentuk dari peta ruangan C tidak terlalu sempurna. Terlihat pada salah satu pojok ruangan tidak menutup secara sempurna. Hal ini seharunya bisa diatas oleh *loop closure* pada program RtabMap, namun setelah penulis coba untuk menemukan titik *loop closure* hal tersebut tidak berhasil dilakukan.



Gambar 4.44 *Landmark* yang terdeteksi pada ruangan B



Gambar 4.45 *Lost track* yang terjadi pada ruangan B

Permasalahan juga mungkin akan terjadi bila data citra RGB dan *depth* yang ditangkap oleh kamera pada suatu frame mengarah kepada suatu

benda yang polos seperti tembok saja. Hal ini akan membuat tidak tertangkapnya *landmark* apapun sehingga tidak ada yang bisa dijadikan acuan. Namun pada ruangan pengujian yang penulis pilih tidak terdapat kejadian tersebut. Citra yang tertangkap pada setiap *framanya* selalu terdapat *landmark* lain yang dapat membantu kepolosan tembok yang ada disekitar.

Selain peta 2 dimensi dan 3 dimensi, terdapat juga *file* database yang dihasilkan dari proses program RtabMap yang dijalankan. Pada *file* ini kita bisa melihat citra RGB berserta *depth* untuk setiap frame yang ditangkap pada proses pemetaan dan lokalisasi. Terdapat pula informasi setiap *landmark* pada tiap *framanya*. Selain itu juga terdapat informasi posisi dan orientasi kamera yang dapat memperlihatkan pose robot pada setiap *framanya* relatif terhadap titik awal saat program dijalankan. *File* data base ini juga bisa digunakan untuk lokalisasi robot secara *offline*.



Gambar 4.46 Database yang dihasilkan oleh RtabMap

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis pada sistem yang dirancang yang telah dibahas pada bab 4, penulis dapat menyimpulkan beberapa hasil dari penelitian ini :

1. Metode SLAM visual menggunakan algoritma RtabMap dengan kamera Realsense D435i pada Jetson Nano Developer Kit dapat diterapkan pada *domestic service robot* dengan beberapa batasan.
2. Batasan utamanya adalah ketergantungan algoritma pada fitur gambar yang menyebabkan terjadinya *drift* dan *lost track* pada pembentukan peta ketika berada pada *scene* yang polos seperti tembok polos atau bila terdapat fitur yang identik.
3. Terdapat juga permasalahan pada IMU Realsense D435i yang memiliki *refresh rate* rendah sehingga mengakibatkan terjadinya *drift* pada pembentukan peta ketika bergerak pada kecepatan tinggi.
4. Jetson Nano developer Board yang digunakan mampu menyediakan kemampuan komputasi yang dibutuhkan untuk membentuk peta dengan algoritma yang dirancang namun tidak akan bertahan bila beroperasi dalam waktu yang lama atau bila ditambah tugas lainnya.
5. Peta 2 dimensi dan 3 dimensi yang dihasilkan masih kasar sehingga dibutuhkan pengolahan sebelum dapat digunakan untuk keperluan lainnya (misalnya navigasi).

5.2 Saran

Berdasarkan hasil pengujian dan analis pada sistem yang dirancang yang telah dibahas pada bab 4, penulis menemukan beberapa kekurangan yang dapat diperbaiki untuk mendapatkan hasil yang lebih baik. Beberapa hal tersebut adalah :

1. Untuk mengatasi masalah *drift* dan *lost track* yang disebabkan oleh kurangnya fitur pada *scene* maka bisa digunakan kamera dengan *field of view* yang lebih lebar agar lebih banyak fitur tertangkap oleh kamera.
2. Masalah *drift* yang disebabkan oleh *refresh rate* IMU juga dapat diperbaiki dengan menggunakan IMU dengan kualitas yang lebih baik.
3. Untuk meringankan proses pada Jetson Nano Developer Kit, dapat ditambahkan kamera Realsense T265 yang memiliki

kemampuan untuk memproses visual odometri pada *internal boardnya*. Sehingga Jetson Nano hanya tinggal menerima estimasi *pose* tanpa harus memprosesnya lagi.

4. Solusi lainnya untuk mengatasi masalah kemampuan komputasi pada Jetson Nano Developer Kit adalah dengan menggunakan seri NVIDIA Jetson yang lainnya seperti Jetson TX2 atau Jetson Xavier NX.

DAFTAR PUSTAKA

- [1] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” *IEEE Robot. Automat. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006, doi: 10.1109/MRA.2006.1638022.
- [2] J. Parmar and C. Savant, “Selection of Wheels in Robotics,” *International Journal of Scientific & Engineering Research*, vol. 5, no. 10, pp. 339–343, Oct. 2014.
- [3] P. Corke, *Robotics, Vision and Control*, vol. 118. Cham: Springer International Publishing, 2017.
- [4] “STM32F4 Discovery Review.” <https://startingelectronics.org/reviews/evaluation-boards/STM32F4-discovery-review/> (accessed Jun. 20, 2020).
- [5] “Jetson Nano Developer Kit,” *NVIDIA Developer*, Mar. 06, 2019. <https://developer.nvidia.com/embedded/jetson-nano-developer-kit> (accessed Jul. 15, 2020).
- [6] “Which Intel RealSense device is right for you? (Updated June 2020),” *Intel® RealSense™ Depth and Tracking Cameras*, Mar. 21, 2019. <https://www.intelrealsense.com/which-device-is-right-for-you/> (accessed Jun. 20, 2020).
- [7] “ROS.org | About ROS.” <https://www.ros.org/about-ros/> (accessed Jun. 20, 2020).
- [8] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, “Estimation of IMU and MARG orientation using a gradient descent algorithm,” in *2011 IEEE International Conference on Rehabilitation Robotics*, Zurich, Jun. 2011, pp. 1–7, doi: 10.1109/ICORR.2011.5975346.
- [9] “RTAB-Map | Real-Time Appearance-Based Mapping.” <http://introlab.github.io/rtabmap/> (accessed Jul. 15, 2020).
- [10] N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J.-Y. Ertaud, “Benchmark of Visual SLAM Algorithms: ORB-SLAM2 vs RTAB-Map*,” in *2019 Eighth International Conference on Emerging Security Technologies (EST)*, Colchester, United Kingdom, Jul. 2019, pp. 1–6, doi: 10.1109/EST.2019.8806213.
- [11] A. Grunnet-Jepsen, J. N. Sweetser, and J. Woodfill, “Best-Known-Methods for Tuning Intel® RealSense™ D400 Depth Cameras for Best Performance,” p. 10.

Halaman ini sengaja dikosongkan

LAMPIRAN

Program Kalibrasi IMU Realsense D435i

```
#!/usr/bin/python
from __future__ import print_function
import numpy as np
import sys
import json
import ctypes
import os
import binascii
import struct
import pyrealsense2 as rs
import ctypes
import time
import enum
import threading

is_data = None
get_key = None
if os.name == 'posix':
    import select
    import tty
    import termios

    is_data = lambda : select.select([sys.stdin], [], [], 0) == ([sys.stdin], [], [])
    get_key = lambda : sys.stdin.read(1)

elif os.name == 'nt':
    import msvcrt
    is_data = msvcrt.kbhit
    get_key = lambda : msvcrt.getch()

else:
    raise Exception('Unsupported OS: %s' % os.name)

if sys.version_info[0] < 3:
    input = raw_input

max_float = struct.unpack('f',b'\xff\xff\xff\xff\xff\xff')[0]
max_int = struct.unpack('i',b'\xff\xff\xff\xff\xff\xff')[0]
max_uint8 = struct.unpack('B', b'\xff')[0]
```

```

g      = 9.80665 # SI Gravity page 52 of
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication330e2008.pdf

COLOR_RED  = "\033[1;31m"
COLOR_BLUE = "\033[1;34m"
COLOR_CYAN = "\033[1;36m"
COLOR_GREEN = "\033[0;32m"
COLOR_RESET = "\033[0;0m"
COLOR_BOLD  = "\033[1m"
COLOR_REVERSE = "\033[;7m"

class imu_wrapper:
    class Status(enum.Enum):
        idle = 0,
        rotate = 1,
        wait_to_stable = 2,
        collect_data = 3

    def __init__(self):
        self.pipeline = None
        self.imu_sensor = None
        self.status = self.Status(self.Status.idle) # 0 - idle, 1 - rotate to position,
        2 - wait to stable, 3 - pick data
        self.thread = threading.Condition()
        self.step_start_time = time.time()
        self.time_to_stable = 3
        self.time_to_collect = 2
        self.samples_to_collect = 1000
        self.rotating_threshold = 0.1
        self.moving_threshold_factor = 0.1
        self.collected_data_gyro = []
        self.collected_data_accel = []
        self.callback_lock = threading.Lock()
        self.max_norm = np.linalg.norm(np.array([0.5, 0.5, 0.5]))
        self.line_length = 20
        self.is_done = False
        self.is_data = False

    def escape_handler(self):
        self.thread.acquire()
        self.status = self.Status.idle
        self.is_done = True

```

```

        self.thread.notify()
        self.thread.release()
        sys.exit(-1)

def imu_callback(self, frame):
    if not self.is_data:
        self.is_data = True

    with self.callback_lock:
        try:
            if is_data():
                c = get_key()
                if c == '\x1b':      # x1b is ESC
                    self.escape_handler()

            if self.status == self.Status.idle:
                return
            pr = frame.get_profile()
            data = frame.as_motion_frame().get_motion_data()
            data_np = np.array([data.x, data.y, data.z])
            elapsed_time = time.time() - self.step_start_time

            ## Status.collect_data
            if self.status == self.Status.collect_data:
                sys.stdout.write('\r %15s' % self.status)
                part_done      =      len(self.collected_data_accel) / float(self.samples_to_collect)
                # sys.stdout.write(': %-.3lf (secs)' % (self.time_to_collect - elapsed_time))

                color = COLOR_GREEN
                if pr.stream_type() == rs.stream.gyro:

                    self.collected_data_gyro.append(np.append(frame.get_timestamp(), data_np))
                    is_moving = any(abs(data_np) > self.rotating_threshold)
                    else:
                        is_in_norm = np.linalg.norm(data_np - self.crnt_bucket) < self.max_norm
                        if is_in_norm:

                            self.collected_data_accel.append(np.append(frame.get_timestamp(),
data_np))
                            else:

```

```

        color = COLOR_RED
        is_moving = abs(np.linalg.norm(data_np) - g) / g >
self.moving_threshold_factor

        sys.stdout.write(color)
        sys.stdout.write(['+', int(part_done * self.line_length) +
'*int((1-part_done)*self.line_length) + ']')
        sys.stdout.write(COLOR_RESET)

    if is_moving:
        print('WARNING: MOVING')
        self.status = self.Status.rotate
        return

    # if elapsed_time > self.time_to_collect:
    if part_done >= 1:
        self.status = self.Status.collect_data
        sys.stdout.write('\n\nDirection data collected.')
        self.thread.acquire()
        self.status = self.Status.idle
        self.thread.notify()
        self.thread.release()
        return

    if pr.stream_type() == rs.stream.gyro:
        return
    sys.stdout.write('\r % 15s' % self.status)
    crnt_dir = np.array(data_np) / np.linalg.norm(data_np)
    crnt_diff = self.crnt_direction - crnt_dir
    is_in_norm = np.linalg.norm(data_np - self.crnt_bucket) <
self.max_norm

    ## Status.rotate
    if self.status == self.Status.rotate:
        sys.stdout.write(': %35s' % (np.array2string(crnt_diff,
precision=4, suppress_small=True)))
        sys.stdout.write(': %35s' % (np.array2string(abs(crnt_diff) <
0.1)))
    if is_in_norm:
        self.status = self.Status.wait_to_stable
        sys.stdout.write('\r+' '*90)
        self.step_start_time = time.time()
        return

```

```

## Status.wait_to_stable
if self.status == self.Status.wait_to_stable:
    sys.stdout.write(': %-.3lf (secs)' % (self.time_to_stable - elapsed_time))
    if not is_in_norm:
        self.status = self.Status.rotate
        return
    if elapsed_time > self.time_to_stable:
        self.collected_data_gyro = []
        self.collected_data_accel = []
        self.status = self.Status.collect_data
        self.step_start_time = time.time()
        return
    return
except Exception as e:
    print('ERROR?' + str(e))
    self.thread.acquire()
    self.status = self.Status.idle
    self.thread.notify()
    self.thread.release()

def get_measurements(self, buckets, bucket_labels):
    measurements = []
    print('-----')
    print('*** Press ESC to Quit ***')
    print('-----')
    for bucket,bucket_label in zip(buckets, bucket_labels):
        self.cmnt_bucket = np.array(bucket)
        self.cmnt_direction = np.array(bucket) / np.linalg.norm(np.array(bucket))
        print("\nAlign to direction: ", self.cmnt_direction, ' ', bucket_label)
        self.status = self.Status.rotate
        self.thread.acquire()
        while (not self.is_done and self.status != self.Status.idle):
            self.thread.wait(3)
            if not self.is_data:
                raise Exception('No IMU data. Check connectivity.')
        if self.is_done:
            raise Exception('User Abort.')
        measurements.append(np.array(self.collected_data_accel))
    return np.array(measurements), np.array(self.collected_data_gyro)

```

```

def enable_imu_device(self, serial_no):
    self.pipeline = rs.pipeline()
    cfg = rs.config()
    cfg.enable_device(serial_no)
    try:
        self.pipeline.start(cfg)
    except Exception as e:
        print('ERROR: ', str(e))
        return False

# self.sync_imu_by_this_stream = rs.stream.any
active_imu_profiles = []

active_profiles = dict()
self imu_sensor = None
for sensor in self.pipeline.get_active_profile().get_device().sensors:
    for pr in sensor.get_stream_profiles():
        if pr.stream_type() == rs.stream.gyro and pr.format() == rs.format.motion_xyz32f:
            active_profiles[pr.stream_type()] = pr
            self imu_sensor = sensor
        if pr.stream_type() == rs.stream.accel and pr.format() == rs.format.motion_xyz32f:
            active_profiles[pr.stream_type()] = pr
            self imu_sensor = sensor
    if self imu_sensor:
        break
if not self imu_sensor:
    print('No IMU sensor found.')
    return False
print("\n".join(['FOUND %s with fps=%s' % (str(ap[0]).split('.')[1].upper(), ap[1].fps()) for ap in active_profiles.items()]))
active_imu_profiles = list(active_profiles.values())
if len(active_imu_profiles) < 2:
    print('Not all IMU streams found.')
    return False
self imu_sensor.stop()
self imu_sensor.close()
self imu_sensor.open(active_imu_profiles)
self imu_start_loop_time = time.time()
self imu_sensor.start(self imu_callback)

# Make the device use the original IMU values and not already calibrated:

```

```

if self.imu_sensor.supports(rs.option.enable_motion_correction):
    self.imu_sensor.set_option(rs.option.enable_motion_correction, 0)
return True

class CHeader:
    def __init__(self, version, table_type):
        self.buffer = np.ones(16, dtype=np.uint8) * 255
        self.buffer[0] = int(version[0], 16)
        self.buffer[1] = int(version[1], 16)
        self.buffer.dtype=np.uint16
        self.buffer[1] = int(table_type, 16)

    def size(self):
        return 16

    def set_data_size(self, size):
        self.buffer.dtype=np.uint32
        self.buffer[1] = size

    def set_crc32(self, crc32):
        self.buffer.dtype=np.uint32
        self.buffer[3] = crc32 % (1<<32) # convert from signed to unsigned 32
bit

    def get_buffer(self):
        self.buffer.dtype=np.uint8
        return self.buffer

def bitwise_int_to_float(ival):
    return struct.unpack('f', struct.pack('i', ival))[0]

def bitwise_float_to_int(fval):
    return struct.unpack('i', struct.pack('f', fval))[0]

def parse_buffer(buffer):
    cmd_size = 24
    header_size = 16

    buffer.dtype=np.uint32
    tab1_size = buffer[3]
    buffer.dtype=np.uint8
    print('tab1_size (all_data): ', tab1_size)

```

```

tab1 = buffer[cmd_size:cmd_size+tab1_size] # 520 == eeprom++
tab1.dtype=np.uint32
tab2_size = tab1[1]
tab1.dtype=np.uint8
print('tab2_size (calibration_table): ', tab2_size)

tab2 = tab1[header_size:header_size+tab2_size] # calibration table
tab2.dtype=np.uint32
tab3_size = tab2[1]
tab2.dtype=np.uint8
print('tab3_size (calibration_table): ', tab3_size)

tab3 = tab2[header_size:header_size+tab3_size] # D435 IMU Calib Table
tab3.dtype=np.uint32
tab4_size = tab3[1]
tab3.dtype=np.uint8
print('tab4_size (D435_IMU_Calib_Table): ', tab4_size)

tab4 = tab3[header_size:header_size+tab4_size] # calibration data
return tab1, tab2, tab3, tab4

def get_D435_IMU_Calib_Table(X):
    version = [0x02, 0x01]
    table_type = '0x20'
    header = CHeader(version, table_type)

    header_size = header.size()
    data_size = 37*4 + 96
    size_of_buffer = header_size + data_size # according to table "D435 IMU
    Calib Table" here: https://user-
    images.githubusercontent.com/6958867/50902974-20507500-1425-11e9-
    8ca5-8bd2ac2d0ea1.png
    assert(size_of_buffer % 4 == 0)
    buffer = np.ones(size_of_buffer, dtype=np.uint8) * 255

    use_extrinsics = False
    use_intrinsics = True

    data_buffer = np.ones(data_size, dtype=np.uint8) * 255
    data_buffer.dtype = np.float32

    data_buffer[0] = bitwise_int_to_float(np.int32(int(use_intrinsics)) << 8 |
```

```

    np.int32(int(use_extrinsics)))

intrinsic_vector = np.zeros(24, dtype=np.float32)
intrinsic_vector[:9] = X[:3,:3].T.flatten()
intrinsic_vector[9:12] = X[:3,3]
intrinsic_vector[12:21] = X[3:,:3].flatten()
intrinsic_vector[21:24] = X[3:,3]

data_buffer[13:13+X.size] = intrinsic_vector
data_buffer.dtype = np.uint8

header.set_data_size(data_size)

header.set_crc32(binascii.crc32(data_buffer))
buffer[:header_size] = header.get_buffer()
buffer[header_size:] = data_buffer
return buffer

def get_calibration_table(d435_imu_calib_table):
    version = ['0x02', '0x00']
    table_type = '0x20'

    header = CHeader(version, table_type)

    d435_imu_calib_table_size = d435_imu_calib_table.size
    sn_table_size = 32
    data_size = d435_imu_calib_table_size + sn_table_size

    header_size = header.size()
    size_of_buffer = header_size + data_size # according to table "D435 IMU
    Calib Table" in
    "https://sharepoint.ger.ith.intel.com/sites/3D_project/Shared%20Documents/
    Arch/D400/FW/D435i_IMU_Calibration_eeprom_0_52.xlsx"
    assert(size_of_buffer % 4 == 0)
    buffer = np.ones(size_of_buffer, dtype=np.uint8) * 255

    data_buffer = np.ones(data_size, dtype=np.uint8) * 255
    data_buffer[:d435_imu_calib_table_size] = d435_imu_calib_table

    header.set_data_size(data_size)
    header.set_crc32(binascii.crc32(data_buffer))

```



```

cmd.dtype = np.uint16
cmd[0] += DC_MM_EEPROM_SIZE
cmd.dtype = np.uint32
cmd[3] = DC_MM_EEPROM_SIZE # command 1 = 0x50
    # command 2 = 0
    # command 3 = size
cmd.dtype = np.uint8
buffer[:len(cmd)] = cmd
buffer[len(cmd):len(cmd)+eprom.size] = eeprom

debug = get_debug_device(serial_no)
if not debug:
    print('Error getting RealSense Device.')
    return
# tab1, tab2, tab3, tab4 = parse_buffer(buffer)

rcvBuf = debug.send_and_receive_raw_data(bytarray(buffer))
if rcvBuf[0] == buffer[4]:
    print('SUCCESS: saved calibration to camera.')
else:
    print('FAILED: failed to save calibration to camera.')
    print(rcvBuf)

def get_debug_device(serial_no):
    ctx = rs.context()
    devices = ctx.query_devices()
    found_dev = False
    for dev in devices:
        if len(serial_no) == 0 or serial_no ==
dev.get_info(rs.camera_info.serial_number):
            found_dev = True
            break
    if not found_dev:
        print('No RealSense device found' + str('.' if len(serial_no) == 0 else ' with
serial number: '+serial_no))
        return 0

    # set to advance mode:
    advanced = rs.rs400_advanced_mode(dev)
    if not advanced.is_enabled():
        advanced.toggle_advanced_mode(True)

```

```

# print(a few basic information about the device)
print(' Device PID: ', dev.get_info(rs.camera_info.product_id))
print(' Device name: ', dev.get_info(rs.camera_info.name))
print(' Serial number: ', dev.get_info(rs.camera_info.serial_number))
print('                                     Firmware           version:      ',
dev.get_info(rs.camera_info.firmware_version))
debug = rs.debug_protocol(dev)
return debug

def check_X(X, accel, show_graph):
    fdata = np.apply_along_axis(np.dot, 1, accel, X[:3,:3]) - X[3,:]
    norm_data = (accel**2).sum(axis=1)**(1./2)
    norm_fdata = (fdata**2).sum(axis=1)**(1./2)
    if show_graph:
        import pylab
        pylab.plot(norm_data, 'b')
        #pylab.hold(True)
        pylab.plot(norm_fdata, 'g')
        pylab.show()
    print ('norm (raw data ): %f' % np.mean(norm_data))
    print ('norm (fixed data): %f' % np.mean(norm_fdata), "A good calibration
will be near %f" % g)

def main():
    if any([help_str in sys.argv for help_str in ['-h', '--help', '?']]):
        print("Usage:", sys.argv[0], "[Options]")
        print
        print('[Options]:')
        print('-i : /path/to/accel.txt [/path/to/gyro.txt]')
        print('-s : serial number of device to calibrate.')
        print('-g : show graph of norm values - original values in blue and
corrected in green.')
        print
        print('If -i option is given, calibration is done using previously saved files')
        print('Otherwise, an interactive process is followed.')
        sys.exit(1)

    try:
        accel_file = None
        gyro_file = None
        serial_no =
        show_graph = '-g' in sys.argv

```

```

for idx in range(len(sys.argv)):
    if sys.argv[idx] == '-i':
        accel_file = sys.argv[idx+1]
        if len(sys.argv) > idx+2 and not sys.argv[idx+2].startswith('-'):
            gyro_file = sys.argv[idx+2]
    if sys.argv[idx] == '-s':
        serial_no = sys.argv[idx+1]

buckets = [[0, -g, 0], [ g, 0, 0],
           [0, g, 0], [-g, 0, 0],
           [0, 0, -g], [ 0, 0, g]]

buckets_labels = ["Upright facing out", "USB cable up facing out",
                  "Upside down facing out", "USB cable pointed down",
                  "Viewing direction facing down", "Viewing direction facing up"]

gyro_bais = np.zeros(3, np.float32)
old_settings = None
if accel_file:
    if gyro_file:
        #compute gyro bais

        #assume the first 4 seconds the device is still
        gyro = np.loadtxt(gyro_file, delimiter=",")
        gyro = gyro[gyro[:, 0] < gyro[0, 0]+4000, :]

        gyro_bais = np.mean(gyro[:, 1:], axis=0)
        print(gyro_bais)

    #compute accel intrinsic parameters
    max_norm = np.linalg.norm(np.array([0.5, 0.5, 0.5]))

    measurements = [[], [], [], [], [], []]
    import csv
    with open(accel_file, 'r') as csvfile:
        reader = csv.reader(csvfile)
        rnum = 0
        for row in reader:
            M = np.array([float(row[1]), float(row[2]), float(row[3])])
            is_ok = False
            for i in range(0, len(buckets)):
                if np.linalg.norm(M - buckets[i]) < max_norm:
                    is_ok = True

```

```

        measurements[i].append(M)
        rnum += 1
        print('read %d rows.' % rnum)
    else:
        print('Start interactive mode:')
        if os.name == 'posix':
            old_settings = termios.tcgetattr(sys.stdin)
            tty.setcbreak(sys.stdin.fileno())

    imu = imu_wrapper()
    if not imu.enable_imu_device(serial_no):
        print('Failed to enable device.')
        return -1
    measurements,      gyro      =      imu.get_measurements(buckets,
buckets_labels)
    con_mm = np.concatenate(measurements)
    if os.name == 'posix':
        termios.tcsetattr(sys.stdin, termios.TCSADRAIN, old_settings)

    header = input('\nWould you like to save the raw data? Enter footer for
saving files (accel_<footer>.txt and gyro_<footer>.txt)\nEnter nothing to not
save raw data to disk. >')
    print('\n')
    if header:
        accel_file = 'accel_%s.txt' % header
        gyro_file = 'gyro_%s.txt' % header
        print('Writing files:\n%s\n%s' % (accel_file, gyro_file))
        np.savetxt(accel_file, con_mm, delimiter=',', fmt='%s')
        np.savetxt(gyro_file, gyro, delimiter=',', fmt='%s')
    else:
        print('Not writing to files.')
    # remove times from measurements:
    measurements = [mm[:,1:] for mm in measurements]

    gyro_bais = np.mean(gyro[:, 1:], axis=0)
    print(gyro_bais)

    mlen = np.array([len(meas) for meas in measurements])
    print(mlen)
    print('using %d measurements.' % mlen.sum())

    nrows = mlen.sum()
    w = np.zeros([nrows, 4])

```

```

Y = np.zeros([nrows, 3])
row = 0
for i in range(0, len(buckets)):
    for m in measurements[i]:
        w[row, 0] = m[0]
        w[row, 1] = m[1]
        w[row, 2] = m[2]
        w[row, 3] = -1
        Y[row, 0] = buckets[i][0]
        Y[row, 1] = buckets[i][1]
        Y[row, 2] = buckets[i][2]
    row += 1
np_version = [int(x) for x in np.version.version.split('.')]
rcond_val = None if (np_version[1] >= 14 or np_version[0] > 1) else -1
X, residuals, rank, singular = np.linalg.lstsq(w, Y, rcond=rcond_val)

print(X)
print("residuals:", residuals)
print("rank:", rank)
print("singular:", singular)
check_X(X, w[:, :3], show_graph)

calibration = {}
calibration["device_type"] = "D435i"
calibration["imus"] = list()
calibration["imus"].append({})
calibration["imus"][0]["accelerometer"] = {}
calibration["imus"][0]["accelerometer"]["scale_and_alignment"] = X.flatten()[:9].tolist()
calibration["imus"][0]["accelerometer"]["bias"] = X.flatten()[9:].tolist()
calibration["imus"][0]["gyroscope"] = {}
calibration["imus"][0]["gyroscope"]["scale_and_alignment"] = np.eye(3).flatten().tolist()
calibration["imus"][0]["gyroscope"]["bias"] = gyro_bais.tolist()
json_data = json.dumps(calibration, indent=4, sort_keys=True)

directory = os.path.dirname(accel_file) if accel_file else ''

with open(os.path.join(directory, "calibration.json"), 'w') as outfile:
    outfile.write(json_data)

#concatinante the two 12 element arrays and save
intrinsic_buffer = np.zeros([6,4])

```

```

intrinsic_buffer[:3,:4] = X.T
intrinsic_buffer[3:,:3] = np.eye(3)
intrinsic_buffer[3:,3] = gyro_bais

#                                     intrinsic_buffer
#((np.array(range(24),np.float32)+1)/10).reshape([6,4]) =



d435_imu_calib_table = get_D435_IMU_Calib_Table(intrinsic_buffer)
calibration_table = get_calibration_table(d435_imu_calib_table)
eeprom = get_eeprom(calibration_table)

with open(os.path.join(directory,"calibration.bin"), 'wb') as outfile:
    outfile.write(eeprom.astype('f').tostring())

is_write = input('Would you like to write the results to the camera\'s
eeprom? (Y/N)')
is_write = 'Y' in is_write.upper()
if is_write:
    print("Writing calibration to device.")
    write_eeprom_to_camera(eeprom, serial_no)
    print("Done.")
else:
    print('Abort writing to device')
except Exception as e:
    print ('\nDone. %s' % e)
finally:
    if os.name == 'posix' and old_settings is not None:
        termios.tcsetattr(sys.stdin, termios.TCSADRAIN, old_settings)

"""

wtw = dot(transpose(w),w)
wtwi = np.linalg.inv(wtw)
print(wtwi)
X = dot(wtwi, Y)
print(X)
"""

if __name__ == '__main__':
    main()

```

Hasil Kalibrasi IMU pada Realsense D435i

Start interactive mode:

```
FOUND GYRO with fps=200
FOUND ACCEL with fps=63
-----
*** Press ESC to Quit ***
-----
Align to direction: [ 0. -1. 0.] Upright facing out
Status.collect_dataWARNING: MOVING      ] [0;0mm
Status.collect_dataWARNING: MOVING      ] [0;0mm
Status.collect_dataWARNING: MOVING.... ] [0;0mm
Status.collect_dataWARNING: MOVING.... ] [0;0mm
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.
Align to direction: [1. 0. 0.] USB cable up facing out
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.
Align to direction: [0. 1. 0.] Upside down facing out
Status.collect_dataWARNING: MOVING..... ] [0;0mm
Status.collect_dataWARNING: MOVING      ] [0;0mm
Status.collect_dataWARNING: MOVING      ] [0;0mm
    Status.rotate: [-0.0079 0.0124 0.157 ]: [ True True False]
Done. User Abort.

C:\Users\paltz>python rs-imu-calibration.py
Start interactive mode:
FOUND GYRO with fps=200
FOUND ACCEL with fps=63
-----
*** Press ESC to Quit ***
-----
Align to direction: [ 0. -1. 0.] Upright facing out
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.
Align to direction: [1. 0. 0.] USB cable up facing out
Status.collect_dataWARNING: MOVING      ] [0;0mm
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.
Align to direction: [0. 1. 0.] Upside down facing out
```

```
Status.collect_data [0;32m[.....] [0;0mm
```

Direction data collected.

Align to direction: [-1. 0. 0.] USB cable pointed down

```
Status.collect_data [0;32m[.....] [0;0mm
```

Direction data collected.

Align to direction: [0. 0. -1.] Viewing direction facing down

```
Status.collect_data [0;32m[.....] [0;0mm
```

Direction data collected.

Align to direction: [0. 0. 1.] Viewing direction facing up

```
Status.collect_data [0;32m[.....] [0;0mm
```

Direction data collected.

Would you like to save the raw data? Enter footer for saving files
(accel_<footer>.txt and gyro_<footer>.txt)

Enter nothing to not save raw data to disk. >n

Writing files:

accel_n.txt

gyro_n.txt

[-0.00072566 0.00039485 -0.00046901]

[1000 1000 1000 1000 1000]

using 6000 measurements.

[[1.01458391 -0.00451168 -0.03412694]

[0.01477121 1.02558431 0.04932103]

[-0.00208471 -0.00277754 1.01538195]

[0.18540034 0.39344496 0.04772792]]

residuals: [7.63765368 6.01097048 124.21942761]

rank: 4

singular: [444.61328109 429.46066677 419.21606161 77.22158551]

norm (raw data): 9.638820

norm (fixed data): 9.805317 A good calibration will be near 9.806650

Would you like to write the results to the camera's eeprom? (Y/N)N

Abort writing to device

```
C:\Users\paltz>python rs-imu-calibration.py
```

Start interactive mode:

FOUND GYRO with fps=200

FOUND ACCEL with fps=63

*** Press ESC to Quit ***

Align to direction: [0. -1. 0.] Upright facing out
Status.collect_dataWARNING: MOVING...] [0;0mm
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.

Align to direction: [1. 0. 0.] USB cable up facing out
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.

Align to direction: [0. 1. 0.] Upside down facing out
Status.collect_dataWARNING: MOVING] [0;0mm
Status.collect_dataWARNING: MOVING] [0;0mm
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.

Align to direction: [-1. 0. 0.] USB cable pointed down
Status.collect_dataWARNING: MOVING
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.

Align to direction: [0. 0. -1.] Viewing direction facing down
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.

Align to direction: [0. 0. 1.] Viewing direction facing up
Status.collect_dataWARNING: MOVING] [0;0mm
Status.collect_data [0;32m[.....] [0;0mm

Direction data collected.

Would you like to save the raw data? Enter footer for saving files
(accel_<footer>.txt and gyro_<footer>.txt)
Enter nothing to not save raw data to disk. >y

Writing files:
accel_y.txt
gyro_y.txt
[-0.00052787 0.00034586 -0.00039177]
[1000 1000 1000 1000 1000]
using 6000 measurements.

```

[[ 1.01462541 -0.00427014 -0.02399263]
[ 0.00876685  1.02472511  0.00676223]
[-0.00470463 -0.00230396  1.01516632]
[ 0.1580268  0.37954637  0.13526645]]
residuals: [ 2.93017932  6.59215818 128.40087669]
rank: 4
singular: [438.60529783 428.89474594 425.99957124 77.22679305]
norm (raw data ): 9.638156
norm (fixed data): 9.805458 A good calibration will be near 9.806650
Would you like to write the results to the camera's eeprom? (Y/N)Y
Writing calibration to device.
Device PID: 0B3A
Device name: Intel RealSense D435I
Serial number: 845112071078
Firmware version: 05.12.03.00
SUCCESS: saved calibration to camera.
Done.

```

Program *Wireless Controller* pada Arduiono UNO

```

#include "PS3BT.h"
#include "SPI.h"
#include "usbhub.h"
#include "avr/wdt.h"

USB Usb;
BTB Btd(&Usb);
PS3BT PS3(&Btd);

char joystick_data[6];
char joystick_status = 0;

char analog = 1;
char ps = 0;

void setup(void)
{
    Serial.begin(38400);

    digitalWrite(A0, HIGH);
    digitalWrite(13, HIGH);
    pinMode(A0, OUTPUT);

```

```

pinMode(13, OUTPUT);

delay(2000);

        Usb.Init();
}

void loop(void)
{
    Usb.Task();

    if (PS3.PS3Connected)
    {
        joystick_status = 1;

        joystick_data[0] = 255 - PS3.l2capinbuf[11];
        joystick_data[1] = 255 - PS3.l2capinbuf[12];
        joystick_data[2] = PS3.l2capinbuf[17];
        joystick_data[3] = PS3.l2capinbuf[18];
        joystick_data[4] = PS3.l2capinbuf[15];
        joystick_data[5] = PS3.l2capinbuf[16];

        ps = PS3.getButtonClick(PS);

        if (ps && analog == 1) { analog = 0;
PS3.setLedOn(LED2); }
            else if (ps && analog == 0) { analog = 1;
PS3.setLedOff(LED2); }

        if (analog == 1)
        {
            joystick_data[2] = 255;
            joystick_data[3] = 255;
            joystick_data[4] = 255;
            joystick_data[5] = 255;
        }

        Serial.write('i');
        Serial.write('t');
        Serial.write('s');
        Serial.write(joystick_data[0]);
        Serial.write(joystick_data[1]);
        Serial.write(joystick_data[2]);

```

```

        Serial.write(joystick_data[3]);
        Serial.write(joystick_data[4]);
        Serial.write(joystick_data[5]);

        Serial.flush();
    }

    if(joystick_status == 0 && millis() > 10000)
    {
        digitalWrite(A0, LOW);
    }
}

```

Program Utama pada STM32F4Discovery Master

```

/* Includes -----*/
#include "main.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "math.h"
#include "odometry.h"
#include "joystick.h"
/* USER CODE END Includes */

/* Private typedef -----*/
/* USER CODE BEGIN PTD */
UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;
UART_HandleTypeDef huart3;
UART_HandleTypeDef huart6;
DMA_HandleTypeDef hdma_usart1_rx;
DMA_HandleTypeDef hdma_usart1_tx;
DMA_HandleTypeDef hdma_usart2_rx;

```

```

DMA_HandleTypeDef hdma_usart2_tx;
DMA_HandleTypeDef hdma_usart3_rx;
DMA_HandleTypeDef hdma_usart3_tx;
DMA_HandleTypeDef hdma_usart6_rx;
DMA_HandleTypeDef hdma_usart6_tx;
/* USER CODE END PTD */

/* Private define -----*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----*/
/* USER CODE BEGIN PV */
//=====Master-Slave=====
unsigned char data_kirim2[23] = {'i','t','s'}; //Mengirim data ke STM32
Slave UART2
    //char data_terima2[7];
        //Menerima data SRF dari STM32 Slave
    char data_terima1[7];
        //Menerima data Gyro dari Arduino UART1 (WARNING: PIN
PA9 Tx UART1 tidak dapat digunakan hanya PA10 RX yang berfungsi)

//=====Serial PC=====
unsigned char kirim_serial_pc[63] = {'i','t','s'};
unsigned char terima_serial_pc[63];

//=====Kinematik=====
//=====Kecepatan Awal Robot=====
short int kecepatan_x = 0;
short int kecepatan_y = 0;
short int kecepatan_sudut = 10;

//=====Buzzer=====
char buzzer_status = 0;
short int buzzer_iterasi = 0;
short int buzzer_waktu = 0, buzzer_jumlah = 0;

```

```

//=====Misc=====
unsigned char tombol_display;
char srf[3];
char status_control;
/* USER CODE END PV */

/* Private function prototypes -----
*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
//=====Buzzer=====
void buzzer(short int waktu, short int jumlah);
/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/* MCU Configuration-----*/
/* */

/* Reset of all peripherals, Initializes the Flash interface and the
Systick.*/
    HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

```

```

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DMA_Init();
MX_TIM3_Init();
MX_TIM4_Init();
MX_TIM6_Init();
MX_TIM7_Init();
MX_USART1_UART_Init();
MX_USART2_UART_Init();
MX_USART3_UART_Init();
MX_USART6_UART_Init();
/* USER CODE BEGIN 2 */
//=====INTERRUPT=====//
HAL_TIM_Base_Start_IT(&htim6);
HAL_TIM_Base_Start_IT(&htim7);
__HAL_UART_ENABLE_IT(&huart6,UART_IT_RXNE);
//=====ODOMETRY=====//
HAL_TIM_Encoder_Start(&htim4,TIM_CHANNEL_ALL);
HAL_TIM_Encoder_Start(&htim3,TIM_CHANNEL_ALL);
//=====SERIAL=====//
HAL_UART_Transmit_DMA(&huart2, (uint8_t*) data_kirim2,
23);//=====>>>/UART2 Kirim Slave
//HAL_UART_Receive_DMA(&huart2, (uint8_t*) data_terima2,
7);//=====>>>/UART2 Terima SRF
HAL_UART_Receive_DMA(&huart1, (uint8_t*) data_terima1,
7);//=====>>>/UART1 Terima Gyro

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

```

```

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    __HAL_RCC_PWR_CLK_ENABLE();

    _HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /** Initializes the CPU, AHB and APB busses clocks
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 8;
    RCC_OscInitStruct.PLL.PLLN = 336;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 7;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }
    /** Initializes the CPU, AHB and APB busses clocks
    */
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
}

```

```

        if ((HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_5) != HAL_OK)
    {
        Error_Handler();
    }
}

/* USER CODE BEGIN 4 */
//=====Timer Interrupt=====
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef
*htim)
{
//=====Timer Interrupt 7=====
if(htim->Instance == TIM7)
{

}

//=====Timer Interrupt 6=====
if(htim->Instance == TIM6)
{
    //=====Joystick
Routine=====/
    if (!joystick_eks)
        status_control = 0;
    if (!joystick_start)
        status_control = 1;

    if (status_control == 0)
    {
        if (joystick_r1)
        {
            kecepatan_x = joystick_x1 * -0.1;
            kecepatan_y = joystick_y1 * -0.1;
            kecepatan_sudut = joystick_x2 * -
0.04;
        }
        if (joystick_r1 == 0)
        {
            kecepatan_x = joystick_x1 * -0.25;
            kecepatan_y = joystick_y1 * -0.25;
            kecepatan_sudut = joystick_x2 * -0.1;
        }
    }
}
}

```

```

        }

//=====Buzzer
Routine=====
        if (buzzer_status == 1 && buzzer_jumlah > 0 &&
buzzer_iterasi++ == buzzer_waktu)
        {
            HAL_GPIO_WritePin(buzzer_GPIO_Port,
buzzer_Pin, GPIO_PIN_RESET);
            buzzer_iterasi = 0;
            buzzer_status = 0;
            buzzer_jumlah--;
        }
        else if (buzzer_status == 0 && buzzer_jumlah > 1 &&
buzzer_iterasi++ == buzzer_waktu)
        {

            HAL_GPIO_WritePin(buzzer_GPIO_Port,buzzer_Pin,
GPIO_PIN_SET);
            buzzer_iterasi = 0;
            buzzer_status = 1;
            buzzer_jumlah--;
        }

//=====Odometry
Routine=====
        odometry_VectorKinematic();
    }

/*
 * Fungsi dieksekusi jika pengiriman Tx UART Selesai
 */

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
/*
 * Tx UART 2 Selesai
 */
if(huart->Instance == USART2)
{
/*
 * Kirim Data Slave

```

```

        */
        memcpy(data_kirim2 + 3, &posisi_x, 4);
        memcpy(data_kirim2 + 7, &posisi_y, 4);
        memcpy(data_kirim2 + 11, &gyro_derajat, 4);
        memcpy(data_kirim2 + 15, &kecepatan_x, 2);
        memcpy(data_kirim2 + 17, &kecepatan_y, 2);
        memcpy(data_kirim2 + 19, &kecepatan_sudut, 2);
        memcpy(data_kirim2 + 21, &status_control, 1);

        HAL_UART_Transmit_DMA(&huart2,           (uint8_t*)
data_kirim2, 23);
    }
}

/*
 * Fungsi dieksekusi jika pengiriman data Rx UART Selesai
*/
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
/*
 * Rx UART 1 Selesai
*/
if(huart->Instance == USART1)
{
/*
 * Blink LED Penanda
 */
static int rx_uart1;
if (rx_uart1 > 20)
{
    HAL_GPIO_TogglePin(LED_ORANGE_GPIO_Port,
LED_ORANGE_Pin);
    rx_uart1 = 0;
}
rx_uart1++;
/*
 * Buzzing Penanda Terima Data Gyro
 */
if(gyro_status == RESET)
{
    buzzer(30,30);
}
}
}

```

```

        gyro_status = SET;
    }

    if(data_terima1[0]=='i' && data_terima1[1]=='t' &&
data_terima1[2]=='s')
    {
        memcpy(&gyro_buffer, data_terima1 + 3, 4);
        receive_gyro_serial();
        HAL_UART_Receive_DMA(&huart1,
(uint8_t*)data_terima1, 7);
    }
    else
    {
        HAL_UART_Receive_DMA(&huart1,
(uint8_t*)data_terima1, 7);
    }
}

/*
 *Rx UART 2 Selesai
 */

/*
if(huart->Instance == USART2)
{
    if(data_terima2[0]=='i' && data_terima2[1]=='t' &&
data_terima2[2]=='s')
    {
        memcpy(&srf, data_terima2 + 3, 3);
        HAL_UART_Receive_DMA(&huart2,
(uint8_t*)data_terima2, 7);
    }
    else
    {
        HAL_UART_Receive_DMA(&huart2,
(uint8_t*)data_terima2, 7);
    }
}

/*
 *Rx UART 6 Selesai
 */
if(huart->Instance == USART6)

```

```

{
    joystick_phase_10;
    joystick_phase_20;

    if(joystick_status == 0)
        joystick_status = 1;

    HAL_DMA_Abort_IT(&hdma_usart6_rx);
    HAL_UART_Receive_IT(&huart6,           (uint8_t*)
&usart6_data, 1);
    HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port,
LED_GREEN_Pin);
}
}

/*
 * Fungsi dieksekusi jika pengiriman Rx UART Setengah Selesai
 */

void      HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef
*huart)
{
/*
 * Melakukan pengecekan data header dari setengah paket data yang
telah diterima Rx UART1
*/
if(huart->Instance == USART1)
{
    if(!(data_terima1[0]=='i' && data_terima1[1]=='t' &&
data_terima1[2]=='s'))
    {
        HAL_UART_AbortReceive(&huart1);
        HAL_UART_Receive_DMA(&huart1,
(uint8_t*)data_terima1, 7);
    }
}

/*
 * Melakukan pengecekan data header dari setengah paket data yang
telah diterima Rx UART2
*/

```

```

/*
    if(huart->Instance == USART2)
    {
        if(!(data_terima2[0]=='i' && data_terima2[1]=='t' &&
data_terima2[2]=='s'))
        {
            HAL_UART_AbortReceive(&huart2);
            HAL_UART_Receive_DMA(&huart2,
(uint8_t*)data_terima2, 7);
        }
    }
*/

/*
 * Jika Komunikasi Serial gagal, maka akan diaktifkan kembali
*/
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance == USART1)
    {
        HAL_UART_Receive_DMA(&huart1,
(uint8_t*)data_terima1, 7);
    }

    if(huart->Instance == USART2)
    {
        HAL_UART_Transmit_DMA(&huart2,      (uint8_t*)
data_kirim2, 23);           //HAL_UART_Receive_DMA(&huart2,      (uint8_t*)
data_terima2, 7);
    }

    if(huart->Instance == USART6)
    {
        HAL_UART_Receive_DMA(&huart6,
(uint8_t*)joystick_terima, sizeof(joystick_terima));
    }
}

/*
 * Buzzer Routine

```

```

*/
void buzzer(short int waktu, short int jumlah)
{
    buzzer_status = 1;
    buzzer_iterasi = 0;
    buzzer_waktu = waktu;
    buzzer_jumlah = jumlah * 2;

    HAL_GPIO_WritePin(buzzer_GPIO_Port,          buzzer_Pin,
GPIO_PIN_SET);
}

```

Program Utama pada STM32F4Discovery Slave

```

/* Includes -----*/
#include "main.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "math.h"
#include "lcd.h"
#include "motor.h"
/* USER CODE END Includes */

/* Private typedef -----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -----*/
/* USER CODE BEGIN PM */

```

```

/* USER CODE END PM */

/* Private variables -----*/
/* USER CODE BEGIN PV */
//=====LCD=====
char lcd[20];

//=====Buzzer=====//
char buzzer_status = 0;
short int buzzer_iterasi = 0;
short int buzzer_waktu = 0, buzzer_jumlah = 0;

//==Master-Slave==//
unsigned char data_terima2[23];
char data_kirim2[7] = {'i','t','s'};
char data_terima3[7];

//=====Kinematik=====//
short int kecepatan_x;
short int kecepatan_y;
short int kecepatan_sudut;
float posisi_x;
float posisi_y;
float gyro_derajat;

//=====Misc=====//
char status_control;
//char srf[3];
//unsigned char tombol_buffer;
/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
//=====Buzzer=====//
void buzzer(short int waktu, short int jumlah);
/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */

```

```

/* USER CODE END 0 */

<��
* @brief The application entry point.
* @retval int
*/
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_DMA_Init();
    MX_TIM1_Init();
    MX_TIM2_Init();
    MX_TIM3_Init();
    MX_TIM6_Init();
    MX_TIM7_Init();
    MX_TIM9_Init();
    MX_TIM10_Init();
    MX_USART1_UART_Init();
    MX_USART2_UART_Init();
    MX_USART3_UART_Init();
    /* USER CODE BEGIN 2 */
    //=====LCD=====

```

```

lcd_init(20,4);

//=====INTERRUPT=====
HAL_TIM_Base_Start_IT(&htim6);
HAL_TIM_Base_Start_IT(&htim7);

//=====PWM MOTOR=====
HAL_TIM_PWM_Start(&htim9,TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim9,TIM_CHANNEL_2);
HAL_TIM_PWM_Start(&htim10,TIM_CHANNEL_1);

//=====ENCODER MOTOR=====
HAL_TIM_Encoder_Start(&htim1,TIM_CHANNEL_ALL);
HAL_TIM_Encoder_Start(&htim2,TIM_CHANNEL_ALL);
HAL_TIM_Encoder_Start(&htim3,TIM_CHANNEL_ALL);

//=====UART=====
HAL_UART_Receive_DMA(&huart2,          (uint8_t*)data_terima2,
23);//==>>>/UART2 Terima Master
//HAL_UART_Transmit_DMA(&huart2,          (uint8_t*)data_kirim2,
7);//==>>>/UART2 Kirim SRF
//HAL_UART_Receive_DMA(&huart3,          (uint8_t*)data_terima3,
7);//==>>>/UART3 Terima SRF

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
lcd_print(3,0,"LULUS TA AMIN");
lcd_print(1,1,"BASE SERVICE ROBOT");
lcd_print(7,2,"MARK I");
lcd_print(2,3,"NO PAIN NO GAIN");
buzzer(20,10);
HAL_Delay(3000);
lcd_clear();

while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */

lcd_print(14,0,"MODE:");

```

```

        if(status_control == 0)
        {
            lcd_print(14,1,"MANUAL");
        }
        else
        {
            lcd_print(14,1,"AUTO ");
        }

        sprintf(lcd, "X=%+04d", (int)posisi_x);
        lcd_print(0,0	lcd);
        sprintf(lcd, "Y=%+04d", (int)posisi_y);
        lcd_print(0,1	lcd);
        sprintf(lcd, "A=%+04d", (int)gyro_derajat);
        lcd_print(0,2	lcd);

        sprintf(lcd, "VX=%+03d", (int)kecepatan_x);
        lcd_print(7,0	lcd);
        sprintf(lcd, "VY=%+03d", (int)kecepatan_y);
        lcd_print(7,1	lcd);
        sprintf(lcd, "W=%+03d", (int)kecepatan_sudut);
        lcd_print(7,2	lcd);

        sprintf(lcd, "w0=%3d", motor_velo[0]);
        lcd_print(0,3	lcd);
        sprintf(lcd, "w1=%3d", motor_velo[1]);
        lcd_print(7,3	lcd);
        sprintf(lcd, "w2=%3d", motor_velo[2]);
        lcd_print(14,3	lcd);

    }
    /* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

```

```

/** Configure the main internal regulator output voltage
*/
__HAL_RCC_PWR_CLK_ENABLE();

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

/** Initializes the CPU, AHB and APB busses clocks
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLM = 8;
RCC_OscInitStruct.PLL.PLLN = 336;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
RCC_OscInitStruct.PLL.PLLQ = 7;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}
/** Initializes the CPU, AHB and APB busses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

if ((HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_5) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM7)

```

```

{
    motor_VeloControl();
}

if(htim-> Instance == TIM6)
{

    motor_VectorKinematic(kecepatan_x,kecepatan_y,kecepatan_sudu
t);

    if (buzzer_status == 1 && buzzer_jumlah > 0 &&
buzzer_iterasi++ == buzzer_waktu)
    {
        HAL_GPIO_WritePin(buzzer_GPIO_Port,
buzzer_Pin, GPIO_PIN_RESET);

        buzzer_iterasi = 0;
        buzzer_status = 0;
        buzzer_jumlah--;
    }
    else if (buzzer_status == 0 && buzzer_jumlah > 1 &&
buzzer_iterasi++ == buzzer_waktu)
    {
        HAL_GPIO_WritePin(buzzer_GPIO_Port,
buzzer_Pin, GPIO_PIN_SET);

        buzzer_iterasi = 0;
        buzzer_status = 1;
        buzzer_jumlah--;
    }
    //      motor_status = !(tombol_buffer >> 7 & 1);
}
}

/*
 * Fungsi dipanggil jika pengiriman data Tx UART Selesai
 */
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
    /*
     * Tx USART 2 Selesai Transmit

```

```

        */

/*      if(huart->Instance == USART2)
    {
        memcpy(data_kirim2 + 3, &srfl, 3);
        HAL_UART_Transmit_DMA(&huart2,
(uint8_t*)data_kirim2, 7);
    }
*/
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance == USART2)
    {
        if(data_terima2[0]=='i'  &&  data_terima2[1]=='t'  &&
data_terima2[2]=='s')
        {
            memcpy(&posisi_x, data_terima2 + 3, 4);
            memcpy(&posisi_y, data_terima2 + 7, 4);
            memcpy(&gyro_derajat, data_terima2 + 11, 4);
            memcpy(&kecepatan_x , data_terima2 + 15, 2);
            memcpy(&kecepatan_y , data_terima2 + 17, 2);
            memcpy(&kecepatan_sudut , data_terima2 +
19, 2);
            memcpy(&status_control , data_terima2 + 21,
1);

            HAL_UART_Receive_DMA(&huart2,
(uint8_t*)data_terima2, 23);
        }
        else
        {
            HAL_UART_Receive_DMA(&huart2,
(uint8_t*)data_terima2, 23);
        }
    }

/*      if(huart->Instance == USART3)
    {
        if(data_terima3[0]=='i'  &&  data_terima3[1]=='t'  &&
data_terima3[2]=='s')
        {

```

```

        memcpy(&data_kirim2, data_terima3, 7);
        memcpy(&srf, data_terima3 + 3, 3);

        HAL_UART_Receive_DMA(&huart3,
(uint8_t*)data_terima3, 7);
    }
    else
    {
        HAL_UART_Receive_DMA(&huart3,
(uint8_t*)data_terima3, 7);
    }
*/
}

void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance == USART2)
    {
        if(!(data_terima2[0]=='i' && data_terima2[1]=='t' &&
data_terima2[2]=='s'))
        {
            HAL_UART_AbortReceive(&huart2);
            HAL_UART_Receive_DMA(&huart2,
(uint8_t*)data_terima2, 23);
        }
    }

    if(huart->Instance == USART3)
    {
        if(!(data_terima3[0]=='i' && data_terima3[1]=='t' &&
data_terima3[2]=='s'))
        {
            HAL_UART_AbortReceive(&huart3);
            HAL_UART_Receive_DMA(&huart3,
(uint8_t*)data_terima3, 7);
        }
    }
}

/*
 * Jika Komunikasi Serial gagal, maka akan diaktifkan kembali
 */

```

```

void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance == USART2)
    {
        HAL_UART_Receive_DMA(&huart2,
(uint8_t*)data_terima2, 23);
        HAL_UART_Transmit_DMA(&huart2,
(uint8_t*)data_kirim2, 7);
    }

    if(huart->Instance == USART3)
    {
        HAL_UART_Receive_DMA(&huart3,
(uint8_t*)data_terima3, 7);
    }
}

/*
 * Buzzer Routine
 */
void buzzer(short int waktu, short int jumlah)
{
    buzzer_status = 1;
    buzzer_iterasi = 0;
    buzzer_waktu = waktu;
    buzzer_jumlah = jumlah * 2;

    HAL_GPIO_WritePin(buzzer_GPIO_Port,          buzzer_Pin,
GPIO_PIN_SET);
}

```

BIODATA PENULIS



Paltzky Ainurrafi Hidayat lahir di Surabaya pada tanggal 6 Oktober 1998. Penulis memulai kehidupan perkuliahan pada tahun 2016 di Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya. Selama masa perkuliahan, penulis aktif dalam berbagai kegiatan kepanitiaan dan organisasi khususnya sebagai asisten laboratorium elektronikamikro dan sistem tertanam dan juga menjabat sebagai koordinator praktikum pada laboratorium tersebut. Penulis dapat dihubungi melalui email: paltzky98@gmail.com