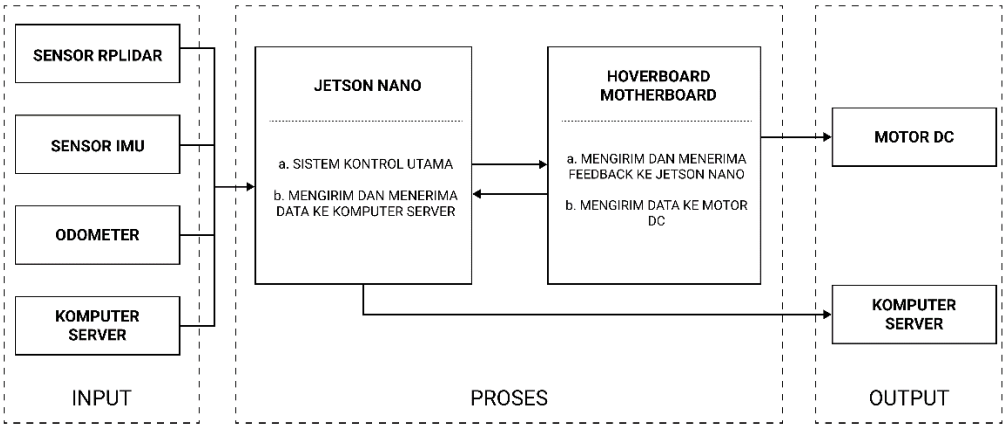
BAB 4 REKAYASA KEBUTUHAN

Dalam bab rekayasa kebutuhan, penulis akan menjelaskan kebutuhan apa saja yang diperlukan untuk mengimplementasikan sistem dalam penelitian ini. Rekayasa kebutuhan mencakup gambaran umum sistem, analisis kebutuhan fungsional dan kebutuhan non-fungsional, kebutuhan perangkat keras, kebutuhan perangkat lunak dan batasan desain sistem. Diharapkan dengan rekayasa kebutuhan ini, sistem yang dibuat bisa berfungsi sesuai dengan tujuan penelitian.

4.1 Gambaran Umum Sistem

Sistem yang dibuat pada penelitian ini merupakan sistem yang mempunyai fungsi untuk membawa barang dari titik awal menuju titik tujuan yang bergerak menggunakan roda serta mendeteksi area sekitar dengan sensor-sensornya. Barang yang dibawa oleh sistem dianalogikan sebagai paket ekspedisi pengiriman asli dengan berat beragam yang nantinya akan dibawa mengelilingi area gudang. Mekanisme yang dilakukan oleh robot dianalogikan sebagai pekerja penyortiran barang. Barang yang disortir berupa paket yang akan dipindahkan dari gudang penyimpanan menuju area distribusi. Setiap informasi yang sudah diselesaikan akan mengirimkan sebuah pesan menuju komputer server.



Gambar 4.1 Blok Diagram Sistem

Dari gambar 4.1 menunjukkan Blok Diagram dari sistem yang memiliki tiga bagian yaitu input, proses, dan output. Bagian input dari sistem akan menggunakan sensor RPLIDAR, IMU dan Odometer. Ketiga sensor ini digabung menjadi satu kesatuan untuk menjalankan proses lokalisasi. Hasil dari sensor RPLIDAR akan digunakan untuk lokalisasi dengan metode AMCL dimana data laser dan map akan di fusi sehingga menghasilkan perkiraan pemosisian. Untuk sensor IMU dan Odometer akan digabungkan sekaligus menggunakan algoritme EKF sehingga menghasilkan odom_combined yang membuat posisi robot selalu terkoreksi dan terfilter. Sedangkan input dari komputer server berupa perintah-perintah *command line* dan input letak koordinat tujuan melalui Rviz atau terminal. Semua bagian input ini akan diteruskan ke bagian proses untuk nantinya diolah oleh Jetson Nano sebagai sistem kontrol utama. Pada bagian proses sistem

menggunakan dua alat yaitu Jetson Nano dan *Hoverboard Motherboard*. Jetson Nano akan menjadi sistem kendali utama dari sistem yang bertugas untuk mencakup seluruh proses pengolahan data sensor, data odometer, dan proses SLAM serta navigasi. Setiap data dari sensor akan dikirimkan ke Jetson Nano yang kemudian akan diproses untuk lokalisasi robot. Hanya sensor RPLIDAR yang mengirimkan data berupa laser scan dengan output yang sudah jadi dan siap digunakan. Jadi tidak perlu melakukan konfigurasi maupun menghitung manual data *raw* sensor. *Hoverboard motherboard* juga mengirimkan data yang sudah jadi berupa odometry, data ini sudah diolah di dalam *motherboard*. Hal ini bisa dilakukan karena kita melakukan *custom* firmware dan memproses data odometer menjadi data odometry yang bisa langsung digunakan tanpa perlu di filtrasi.

Untuk menggerakkan robot, kita hanya perlu memutar motor BLDC. Untuk memutar motor kita perlu memanipulasi nilai pada variabel cmd_vel. Variabel cmd_vel merupakan variabel yang digunakan untuk *trigger* nilai voltase pada *motherboard hoverboard* sehingga motor dapat berputar, hal ini terjadi karena kode biner yang di tanam ke dalam *chip motherboard*. Untuk memanipulasi cmd_vel kita perlu membuat kode di dalam Jetson Nano. Setelah Jetson Nano menentukan nilai cmd_vel maka nilai cmd_vel akan berubah dari kondisi sebelumnya, hal ini juga berlaku pada variabel cmd_vel *motherboard* karena Jetson Nano saling berkomunikasi dengan *motherboard* melalui serial USART. Dengan begitu akan menghasilkan output berupa pergerakan motor, sehingga robot bergerak. Setelah data berubah *motherboard* juga mengirimkan balik nilai Odometer yang terdeteksi ke Jetson Nano untuk mengetahui secara simultan lokasi dari robot. Kemudian Jetson Nano akan mengirimkan semua data pesan yang diterimanya ke dalam Rviz, sehingga kita dapat melihat perubahan yang terjadi.

4.2 Analisis Kebutuhan Sistem

Pada sub bab terkait analisis kebutuhan sistem, proses ini bertujuan agar peneliti tahu segala kebutuhan yang diperlukan oleh sistem pada perangkat keras dan perangkat lunak, baik secara fungsional dan non-fungsional.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan sistem yang di dalamnya berisi proses apa saja yang harus disediakan oleh sistem, seperti bagaimana sistem merespons input dan bagaimana respons sistem pada kondisi tersebut. Kebutuhan fungsional disajikan ke dalam tabel yang dapat diliat pada tabel 4.1.

Tabel 4.1 Kebutuhan Fungsional Sistem

No	Kebutuhan Sistem	Skenario Pengujian
1	Sensor RPLIDAR dapat membaca jarak benda sekitar robot dengan pemindaian pada area seluas 360 derajat. Nilai yang dihasilkan	Dapat mengirimkan data ke mikrokontroler dan pengujian hasil data berupa pengukuran jarak antara robot dengan halangan

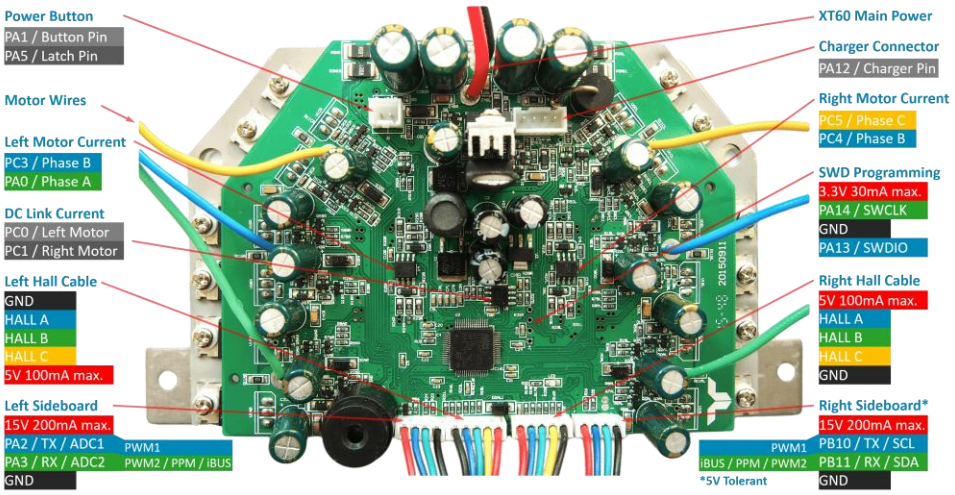
	berupa nilai *distance* atau jarak dari *range* 0.15 meter hingga 6 meter	menggunakan alat pengukur sebenarnya seperti meter ukur
2	Sensor Inertial measurement unit (IMU) dapat membaca nilai derajat kemiringan dan gerak akselerasi pada robot. Berfungsi untuk melakukan Sensing proyeksi terhadap kemiringan dan gerak akselerasi robot	Dapat mengirimkan posisi robot pada mikrokontroler terhadap posisi sebenarnya robot menggunakan pemantauan secara langsung dan proyeksi melalui 3D visual Rviz
3	Motor DC Brushless atau BLDC dapat menggerakkan robot berdasarkan nilai PWM yang diberikan dari *motherboard hoverboard*	Diujinya jumlah putaran motor DC menggunakan *odometry* dan perhitungan secara manual sehingga dipastikan data *odometry* adalah putaran sebenarnya dan robot bergerak sesuai dengan yang didesain
4	Sistem mikrokontroler di dalam robot mampu mengirim dan menerima data sensor IMU untuk dilaksanakan	Diujinya pengiriman data dan penerimaan data ketika diam dan bergerak
5	Sistem mikrokontroler di dalam robot mampu mengirim dan menerima data sensor RPLIDAR untuk dilaksanakan	Diujinya pengiriman data ketika robot melakukan proses pembuatan peta atau *mapping*
6	Mikrokontroler dapat memproses data dari sensor menjadi kesimpulan arah gerak dan kecepatan sistem pada aplikasi ROS	Diujinya navigasi robot menggunakan data yang diberikan oleh sensor
7	Sistem *Motherboard Hoverboard* dapat menggerakkan kedua motor serta memberi tegangan pada semua komponen elektronik.	Diujinya kecepatan berdasarkan perputaran roda melalui encoder motor BLDC menggunakan metode *debugging*
8	Aplikasi Rviz dapat memantau pengontrolan sistem secara otomatis ketika bergerak menggunakan sistem otonom *robot operating system*	Diujinya penggunaan Rviz untuk melihat hasil pengontrolan otonom secara real-time

4.2.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsionalitas pada sistem ini terbagi menjadi dua kebutuhan yaitu perangkat keras dan perangkat lunak yang diperlukan untuk menunjang sistem.

4.2.2.1 Kebutuhan Perangkat Keras

1. *Motherboard Hoverboard*



Gambar 4.2 Perangkat *Motherboard Hoverboard*

Sumber : (aliexpress.com, 2021)

Pada gambar 4.2 dapat dilihat pinout dari motherboard *hoverboard* yang memiliki banyak pin di bagian bawah. *Motherboard Hoverboard* digunakan sebagai *driver* dan *controller* pada motor BLDC. *Hoverboard motherboard* menjadi komponen penting yang menyimpan data firmware yang telah di *flash* sebelumnya. *Motherboard* merupakan komponen wajib yang ada pada penelitian, ini dikarenakan tanpa adanya *motherboard hoverboard* maka tidak akan bisa mengendalikan motor dan akan sangat sulit jika menggunakan motor BLDC *driver* eksternal karena permasalahan biaya dan kompatibilitas dengan firmware yang ada. Berikut adalah spesifikasi yang diambil dari *datasheet* motherboard *hoverboard* yang ditunjukkan pada tabel 4.2.

Tabel 4.2 Spesifikasi *Motherboard Hoverboard*

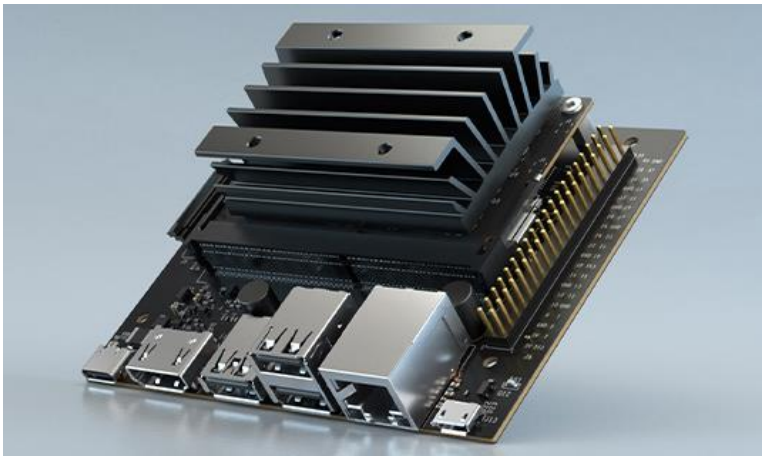
Sumber : (beta.ivc.no dan www.st.com, 2021)

Keterangan	Spesifikasi
Prosesor	GD32F103RCT6 ARM Cortex-M3 32-bit MCU
Flash memory	256 KB sampai 512 KB
SRAM	64 KB
Clock speed	108MHz
Flasher	ST-Link V2 *Programming* Unit
Tegangan operasi	36V
Pengisian daya	42V 2A

Port *Fast* I/O	80 Port (45P, 9ADC, 16PWM, 6RXTX, 4SC, 3.3V, 5V, GND)
Pin Digital I/O motor DC	6 (2xHALL A-C)
Pin Digital I/O sensor	4 (PA2, PA3, PB10, PB11)
Tegangan, arus dan kekuatan maksimal pada Motor DC	36V(1000 r.p.m.) 1-25A 350W
Tegangan dan arus DC Pin I/O motor	5V 100mA
Tegangan dan arus DC Pin I/O sensor	15V 200mA
A/D *converters*	3 x 12-bit
D/A *converters*	2 x 12-bit
Komunikasi	USART, SPI, I2C, I2S, USB, CAN

2. Nvidia Jetson Nano

Nvidia Jetson Nano merupakan pemrosesan inti dari sistem pada penelitian ini, fungsinya sebagai tempat pemrosesan input dari ketiga sensor dan menjalankan fungsi mulai dari lokalisasi, pemetaan dan perencanaan jalur serta komunikasi SSH-VNC pada slave (komputer). Semua metode, algoritme dan komunikasi yang digunakan akan diproses di dalam sini. Alasan dipilihnya Jetson Nano sebagai mikrokomputer dan pemrosesan utama adalah pertimbangan kemampuan dari perangkat ini yang memiliki CPU Quad-core dengan *clock speed* 1.43 GHz dan RAM sebesar 4 GB LPDDR. Gambar 4.3 merupakan bentuk fisik dari Nvidia Jetson Nano



Gambar 4.3 Nvidia Jetson Nano

Sumber: (www.nvidia.com, 2022)

Dengan spesifikasi seperti ini diharapkan Nvidia Jetson Nano mampu melakukan pemrosesan *mapping* dan *navigation.* Proses pemetaan menggunakan sumber daya yang cukup banyak karena terdapat banyak sistem ROS yang berjalan di belakang layar ditambah dengan sistem perencanaan gerak menggunakan Dijkstra yang banyak menguras CPU untuk menemukan jalur terpendek. Memori RAM sebesar 4 GB dirasa sudah mumpuni untuk menjalankan semua proses ini. Walaupun pemrosesan pada Rviz dilakukan di Slave atau Komputer, akan tetapi untuk melakukan *load data* hasil pemetaan juga dibutuhkan sumber daya CPU dan RAM yang besar. Oleh karena itu, kesimpulannya adalah Nvidia Jetson Nano mampu menjalankan sistem secara baik pada penelitian ini. Berikut adalah *datasheet* dari mikrokontroler Jetson Nano 3B yang ditunjukkan pada tabel 4.3.

Tabel 4.3 Spesifikasi Mikrokontroler Jetson Nano 3B

Sumber : (developer.nvidia.com, 2021)

Keterangan	Spesifikasi
CPU	Quad-core ARM A57 @ 1.43 GHz
GPU	NVIDIA Maxwell *architecture with* 128 NVIDIA CUDA® *cores*
Memory	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Storage	*microSD*
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Kamera	2x MIPI CSI-2 DPHY
Konektivitas	*Gigabit Ethernet*, M.2 Key E
Display	HDMI 2.0 *and display port*
USB	4x USB 3.0, USB 2.0 Micro-B
Lainnya	GPIO, I2C, I2S, SPI, USART
Mechanical	69.6 mm x 45 mm 260-pin *edge connector*
Memory External	64GB Mikro SD
Power	*Micro* USB 5V 2.5A *and* DC Barrel Jack 5V 4A

3. Motor DC Brushless dan *build-in encoder*

Gambar 4.4 Perangkat Motor BLDC *Hoverboard*

Sumber : (www.jaxlec.top, 2021)

Motor DC Brushless atau motor BLDC adalah motor sinkron yang menggunakan catu daya listrik arus searah (DC). Ini menggunakan pengontrol loop tertutup elektronik untuk mengalihkan arus DC ke putaran kawat motor yang menghasilkan medan magnet yang secara efektif berputar di ruang angkasa dan yang diikuti oleh rotor magnet permanen. Kontroler menyesuaikan fase dan amplitudo pulsa arus DC untuk mengontrol kecepatan dan torsi motor. Sistem kontrol ini merupakan alternatif dari komutator mekanis (sikat) yang digunakan di banyak motor listrik konvensional. Berikut adalah *datasheet* motor dc *brushless* yang ditunjukkan pada tabel 4.4 dan bentuk fisik dari motor BLDC *hoverboard* ditunjukkan pada gambar 4.4.

Tabel 4.4 Spesifikasi Motor DC Brushless

Sumber : (www.aliexpress.com, 2021)

Keterangan	Spesifikasi
Tipe	Gear Motor, Motor Skuter
Komunikasi	Brushless
Kecepatan	5 - 1500 RPM
Tegangan	36V
Arus	1A - 25A
Power Motor	250 - 350W
Kecepatan Maksimal	26km/h
Kecepatan rata-rata	800rpm
Diameter roda	6.5”
Panjang kabel	26 cm

4. Catu Daya

Gambar 4.5 Baterai *Hoverboard*

Sumber : (ubuy.com, 2020)

Catu daya merupakan hal penting dari sistem yang digunakan untuk penyedia sumber tegangan untuk mengaktifkan keseluruhan sistem. Selain itu, catu daya tergolong jenis yang portabel dengan daya tampung energi listrik yang dapat diisi ulang kembali. Untuk penggunaan catu daya harus dapat menyesuaikan tegangannya yang dibutuhkan oleh sistem (Cahyadi, et al., 2016). Ada dua catu daya yang digunakan pada robot ini, pertama yaitu baterai li-ion dengan besar tegangan 32V dan kuat arus 2A. Catu daya ini digunakan untuk memberikan *power* ke *motherboard hoverboard* dan yang nantinya akan disalurkan lagi ke motor BLDC. Bentuk fisik dari baterai ini dapat dilihat pada gambar 4.5. Kemudian catu daya yang kedua adalah *power* bank ke Jetson Nano dengan besar tegangan 5V dan kuat arus 1A. Berikut adalah tabel spesifikasi Baterai Li-ion 10S2P dan Power bank yang masing-masing ditunjukkan pada tabel 4.5 dan tabel 4.6.

Tabel 4.5 Spesifikasi Catu Daya untuk Suplai *Motherboard Hoverboard*

Sumber : (www. hovertech.co.za, 2021)

Keterangan	Spesifikasi
Teknologi	Litium-Ion
Ukuran Cell	18650 Litium Ion
Tegangan ke *motherboard*	25.2V - 36V
Tegangan maksimal	42V
Arus sistem kerja	1A - 20A
Tenaga	75W hingga 158W
Maksimal pemakaian arus	15A
Arus Pengisian	< 5A
Tegangan Pengisian	43.2V
Kapasitas	4400 mAh
Struktur	10S2P & 10S1P

Tabel 4.6 Spesifikasi Catu Daya untuk Suplai Jetson Nano

Sumber : (www.shopee.co.id, 2021)

Keterangan	Spesifikasi
Teknologi	*Quick Charge* 3.0
Tipe Baterai	*Lithium Polymer*
Output 1	USB A DC 5V/3A, 9V/2A, 12V/1,5A
Output 2	USB A DC 5V/3A, 9V/2A, 12V/1,5A
Output 3	*Type* C 5V/3A, 9V/2,33A, 12V/1,7A
Total Output	20W
Kapasitas	10000 mAh

5. Sensor RPLIDAR A1

Gambar 4.6 Sensor RPLIDAR A1

Sumber : (amazon.com, 2022)

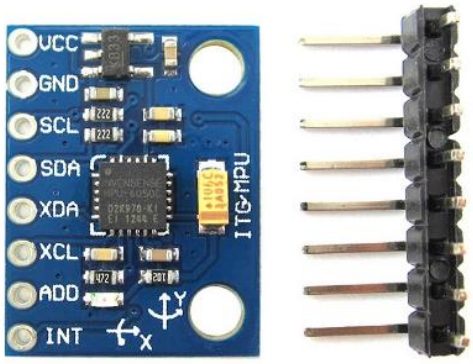
Pada gambar 4.6 ditunjukkan bentuk fisik dari sensor RPLIDAR yang digunakan pada penelitian ini. Terdiri dari sensor RPLIDAR, kabel *socket*, dan *converter port* lidar ke mikro USB. RPLIDAR A1 digunakan sebagai input utama dari sistem berupa tangkapan jarak dari tembakan laser yang nantinya akan diproses oleh sistem. Penggunaan RPLIDAR didasarkan dari kemampuannya mendeteksi objek seluas 360 derajat dan sejauh 6 meter. Selain itu RPLIDAR juga sangat praktis untuk digunakan karena kita tidak perlu menghitung nilai ataupun kalibrasi data karena RPLIDAR memiliki prosesornya sendiri dan mampu mengirimkan data setengah jadi berupa koordinat-koordinat data jarak. Atas semua pertimbangan dan kemampuannya yang *powerful* maka dari itu RPLIDAR dipilih menjadi sensor utama pada penelitian ini. Berikut adalah spesifikasi sensor RPLIDAR yang ditunjukkan pada tabel 4.7.

Tabel 4.7 Spesifikasi RPLIDAR A1

Sumber : (www.slamtec.com, 2021)

Keterangan	Spesifikasi
Measuring Range	0.15 – 12 meter
Angular Field of View	0 - 360°
Angular Resolution:	≤1°
Sampling frequency	4000 - 8000 Hz
Rotational Speed	5.5Hz
Laser wave length	775 - 795nm
Pulse width	110 us
System Voltage	5V
System Current	100mA
Power Consumption	0.5W
Output	USART Serial 3.3 *Voltage level*
Range Resolution	*≤1% of the range （≤12m)* *≤2% of the range （12m～16m)*
Accuracy	*1% of the range （≤3 m）* *2% of the range （3-5 m）* *2.5% of the range （5-25m）*

6. Sensor IMU GY-521



Gambar 4.7 Sensor GY-521 IMU6050

Sumber : (geekbuying.com, 2022)

Sensor GY-521 memiliki 3 DOF akselerometer dan 3 DOF giroskop yang dapat mendeteksi nilai akselerasi dan gerakan roll, pitch atau yaw pada robot. Sensor GY-521 digunakan sebagai input dari sistem berupa mendeteksi posisi robot secara lokal. Hal ini sangat berguna dalam proses lokalisasi, walaupun proses lokalisasi dapat dilakukan hanya dengan menggunakan data dari *Odometer*, akan

tetapi masih perlu dilakukan validasi lebih lanjut lagi menggunakan sensor yang mampu membaca akselerasi robot. Ini dikarenakan terkadang ketika motor BLDC tersangkut atau berputar di tempat maka data yang dikirimkan akan dianggap sedang melakukan suatu gerakan. Tentunya hal ini tidak diinginkan karena dapat membuat robot tidak bergerak semestinya dan posisinya akan berbeda ketika di aplikasi Rviz dan keadaan didunia riil. Maka dari itu atas pertimbangan ini, sensor GY-521 dipilih karena sangat dibutuhkan untuk memvalidasi nilai dari *Odometer* dan membuat robot untuk tetap diposisi semestinya walaupun motor BLDC berputar di tempat. Berikut adalah spesifikasi sensor IMU GY-512 yang ditunjukkan pada tabel 4.8 dan bentuk fisik dari sensor GY-521 yang digunakan ditunjukkan pada gambar 4.7.

Tabel 4.8 Spesifikasi Sensor IMU GY-521 MPU6050

Sumber : (www. berrybase.de, 2022)

Keterangan	Spesifikasi
Chipset	MPU-5060
Tegangan sistem kerja	3.3 - 5V DC
Jalur Komunikasi	I2C *up to* 400kHz
Range Gyroscope	250, 500, 1000, 2000/s
Range Gyroscope	2, 4, 6, 8, 16g
Pin	VCC, RX, TX, GND, RST, B0, SCL, SDA

7. Wi-Fi *Dongle*

Wi-Fi *dongle* digunakan sebagai Wi-Fi *extension* karena Jetson Nano tidak memiliki modul Wi-Fi setelah pabrik. Wi-Fi *dongle* akan menjadi komunikasi utama robot dengan komputer server.

8. Tombol *Power On-Off*

Tombol *power* on-off digunakan untuk menyalakan atau mematikan perangkat robot. Tombol ini berupa *push button* yang kondisinya hanya 1 dan 0.

9. *Adapter dan Port Charger Hoverboard*

Adapter charger baterai digunakan untuk mengisi ulang daya dari baterai *hoverboard*. Adapter ini merupakan adapter AC ke DC yang nantinya ditancapkan pada lubang *port* adapter yang terhubung langsung ke *motherboard hoverboard*.

10. STLink V2

STLink bukanlah bagian dari komponen utama robot, alat ini digunakan dalam proses *flashing motherboard hoverboard*. Tepatnya STlink berfungsi sebagai perantara USB *Flashing* antara komputer dengan *motherboard*. *Flashing* dilakukan untuk *reset* semua kode yang ada di *motherboard* sekaligus menuliskan ulang *motherboard* dengan kode baru yang sudah di modifikasi. Spesifikasi dari STLink yang digunakan pada penelitian ini dapat dilihat pada tabel 4.9.

Tabel 4.9 *Datasheet* STLink V2 mini USB

Sumber : (www.waveshare.com, 2022)

Keterangan	Spesifikasi
SWD voltage range	1.65V - 3.6V
SWIM *voltage range*	1.65V - 5.5V
Supports SWV	No
Debug interfaces	2
LED indicator	Dual color LED
Fuse	YES

11. FTDI mini USB F23RL

FTDI merupakan kepanjangan dari Future Technology Devices International yang merupakan nama perusahaan semiconductor yang mengkhususkan diri dalam teknologi Universal Serial Bus. FTDI digunakan untuk komunikasi serial Jetson Nano dengan *motherboard hoverboard* menggunakan protokol serial USART. FTDI akan terhubung dengan kabel sensor kanan dari *hoverboard* yang memiliki pin RX/TX dan akan ditancapkan dengan *port* USB pada Jetson Nano.

12. Komputer (*Personal* komputer)

Komputer yang dimaksud merupakan sebuah komputer pribadi fleksibel yang dapat dijinjing dengan mudah atau dengan kata lain adalah laptop. Pada penelitian ini, komputer pribadi memiliki peran yang sangat penting dan banyak berkontribusi pada penelitian ini. Di antaranya adalah berguna sebagai perangkat dalam memprogram sistem pada tahap percobaan awal, baik dalam memprogram perangkat keras maupun perangkat lunak. Personal komputer digunakan sebagai komputer server dan seterusnya penulis akan menggunakan kata komputer server sebagai maksud dari komputer pribadi penulis. Komputer server digunakan sebagai *setting up Master - Slave* pada *robot operating system*. Robot bergerak otonom merupakan *slave* sedangkan komputer berperan sebagai *master*. Fungsi *master* (komputer) adalah menerima semua jenis data *feedback* yang dikirim atau di ekspor dari *slave* (robot) yang kemudian data-data tersebut dapat digunakan untuk proses visualisasi menggunakan aplikasi *third party* Rviz. Dengan begini Mikro komputer pada robot tidak perlu banyak menghabiskan Resource untuk melakukan pemetaan sekaligus menampilkan visualisasi dari peta tersebut.

13. *Chassis* papan kayu

Chassis merupakan rangka penyangga suatu struktur. Rangka penyangga pada robot AMR terbuat dari gabungan papan kayu berukuran 40x40 cm sejumlah 4 buah dengan ketebalan 2 cm yang digunakan sebagai bahan pembuatan *chassis* robot. Papan kayu ini nantinya akan menjadi tempat penyimpanan komponen utama seperti Jetson Nano dan bagian atasnya berperan sebagai tutup sekaligus alas untuk tempat menaruh barang.

14. Filamen 3D *print*

Filamen 3D printer sebagai bahan utama tinta 3D printer. *Chassis* robot menggunakan desain *custom* maka dari nantinya akan ada beberapa bagian yang hanya bisa dibuat menggunakan mesin 3D printer. Jenis filamen yang digunakan adalah PETG berwarna hitam sebanyak 2 kilogram.

15. 3D *printer*

3D Printer digunakan untuk mencetak bagian utama robot. Jenis 3D printer yang digunakan pada penelitian ini adalah printer dengan jenis FDM bermerek Creality CR-10S Pro dengan spesifikasi minimal panjang 30 cm, lebar 30 cm dan tinggi 20 cm.

16. Kabel *jumper*

Kabel *jumper* digunakan untuk menghubungkan serial USART sensor kanan *motherboard* dengan FTDI yang terhubung pada Jetson Nano melalui USB mini.

17. Kabel *micro* USB

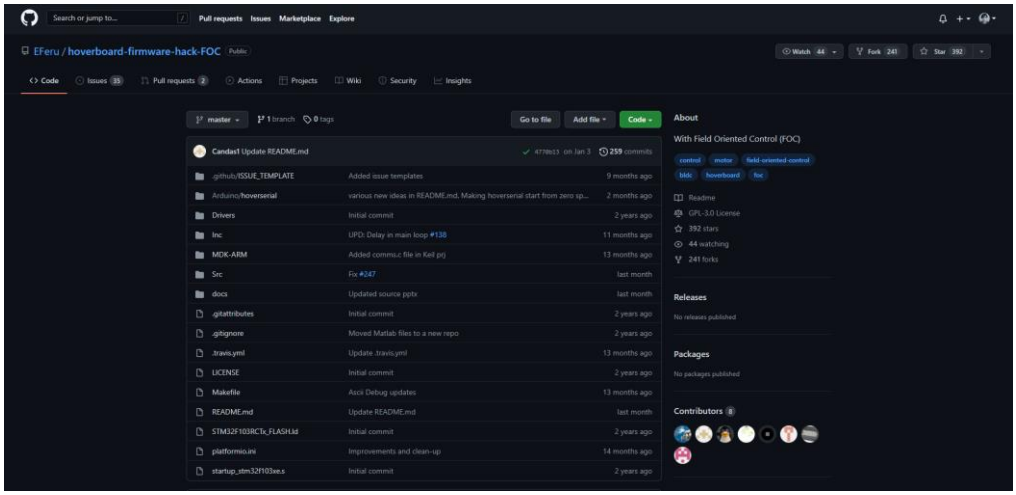
Kabel *micro* USB digunakan untuk menghubungkan sensor RPLIDAR dengan Jetson Nano melalui USB. Selain itu juga digunakan untuk menghubungkan Jetson Nano ke catu daya, yakni *Power bank*.

4.2.2.2 Kebutuhan Non-Fungsional Perangkat Lunak

Kebutuhan non-fungsional adalah batasan dari sistem terhadap layanan yang diberikan. Batasan tersebut dapat berupa waktu, batasan proses pengembangan dan batasan terhadap standar tertentu.

1. *Hoverboard* Firmware

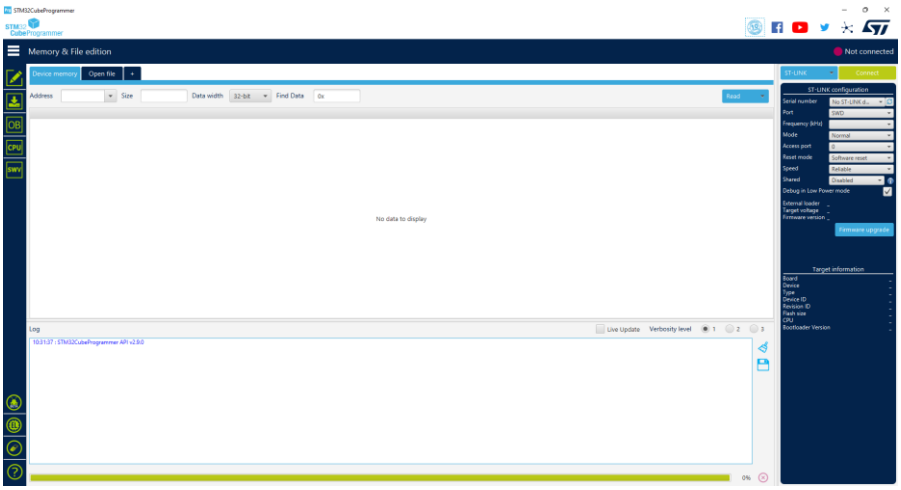
Firmware adalah kelas perangkat lunak komputer tertentu yang menyediakan kontrol tingkat rendah untuk perangkat keras khusus perangkat. Firmware, seperti BIOS komputer pribadi, mungkin berisi fungsi dasar perangkat, dan dapat menyediakan layanan abstraksi perangkat keras ke perangkat lunak tingkat tinggi seperti sistem operasi. Gambar 4.8 merupakan tampilan dari directory *hoverboard*-firmware-hack-FOC yang dikelola oleh Emanuel Feru, pengembang pertama yang membuat firmware *hoverboard* hack ini.



Gambar 4.8 *Hoverboard* Firmware Hack FOC di GitHub

2. STM32Cube Programmer

STM32CubeProgrammer adalah alat perangkat lunak multi-OS all-in-one untuk memprogram produk STM32. Aplikasi ini digunakan untuk melakukan *flashing* pada *motherboard hoverboard*. STM32CubeProgrammer dikirimkan dalam versi GUI (antarmuka pengguna grafis) dan CLI (antarmuka baris perintah). Gambar 4.9 merupakan tampilan GUI dari aplikasi STM32Cube Programmer.

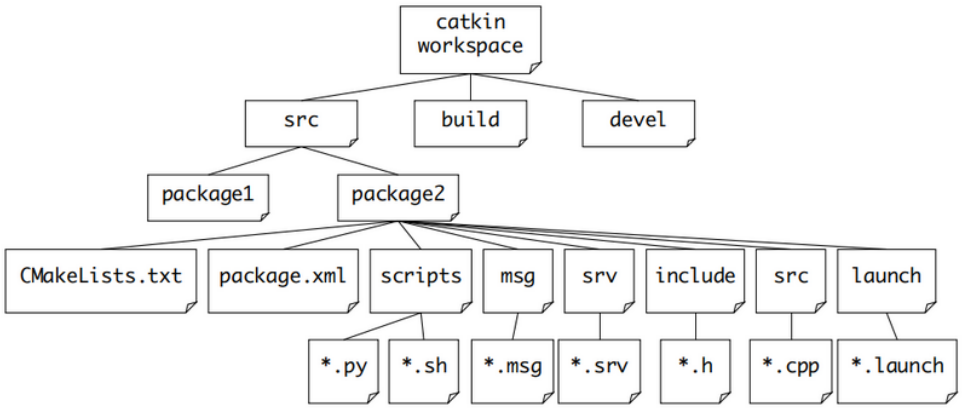


Gambar 4.9 STM32Cube Programmer

3. Catkin workspace

Ruang kerja *catkin* adalah *directory* (*folder*) tempat membuat atau memodifikasi paket *catkin* yang ada. Struktur *catkin* menyederhanakan proses pembuatan dan pemasangan untuk paket ROS. Ruang kerja *catkin* dapat berisi hingga tiga atau lebih *subdirectory* yang berbeda (/build, /devel, dan /src), yang masing-masing memiliki peran berbeda dalam proses pengembangan perangkat lunak. *Catkin* merupakan salah satu metode *workspace* yang ada di dalam ROS, pembuatan *workspace* ini bertujuan untuk memudahkan proses *developing* dan *compiling*, konsep ini sama dengan konsep *webpack* yang ada pada web *developing*. Kemudahan ini terjadi karena untuk menjalankan *middle ware* ROS

kita perlu *build* terlebih dahulu kode ROS yang telah ditulis dan disimpan di dalam folder /src. Kemudian pindahkan posisi terminal pada *directory* /catkin_ws dan ketik $ catkin_make, maka *catkin* akan bekerja dan melakukan *building file* ke folder /*build* dan /devel. Jika sudah selesai maka perintah ROS sudah dapat digunakan dan kita dapat menjalankan sistem berdasarkan pemrograman yang dilakukan di folder /src seperti yang terdapat pada gambar 4.10.



Gambar 4.10 Diagram pohon *directory catkin workspace*

Sumber : (www.medium.com/swlh, 2021)

4. Makefile

Make adalah utilitas untuk membangun dan memelihara grup program (dan jenis *file* lainnya) dari kode sumber. Tujuan dari utilitas *make* adalah untuk menentukan secara otomatis bagian mana dari program besar yang perlu dikompilasi ulang, dan mengeluarkan perintah yang diperlukan untuk mengompilasi ulang (Computer Hope, 2021). *Make* digunakan untuk men install aplikasi dan dependensi ROS, melakukan *compile* firmware, melakukan *flash* firmware dan melakukan *compiling* pada *file* C lainnya dengan hasil output biner.

5. *Robot Operating System* (ROS)

Gambar 4.11 ROS Industrial

Sumber : (rosindustrial.org, 2022)

Robot Operating System atau ROS merupakan packet manager yang dibuat untuk keperluan robotik berisikan kumpulan *library* lengkap untuk membuat perangkat lunak robot. ROS berisikan sekumpulan *library, driver*, algoritme, dan

alat-alat canggih pendukung sistem robot yang sudah siap digunakan. ROS merupakan metode modern untuk membuat sistem robot yang mampu berjalan secara paralel dengan sistem terkontrol. ROS sudah diimplementasi kan pada robot maupun sistem Industri dan kebanyakan berada di luar negeri. Gambar 4.11 merupakan logo ROS yang digunakan untuk Industrial. ROS sangat terkenal dan sistem dengan teknologi ROS digadang akan menjadi teknologi robot dimasa depan.

Pada penelitian ini ROS digunakan sebagai *middleware* utama penyusun perangkat lunak sistem robot. Sistem pada ROS ini terpecah menjadi *node-node* yang dapat disusun bersama dan bisa dijalankan secara paralel. *Library-library* yang ada pada ROS tidak semuanya di install dari internal ROS, melainkan di *install* dari pihak eksternal yang membuatnya di atas lisensi *open-source* MIT. Hal ini membuat pengembang melakukan instalasi pada komputer dan sistem sudah bisa langsung digunakan. Namun tidak semua *library* mendukung perangkat yang digunakan pada setiap pengembang robot. Penulis *library* hanya membuatkan *file-file header* atau sebutannya adalah *driver* yang nantinya bisa langsung digunakan oleh pengembang robot, sehingga tidak perlu mencari dan menulis dari awal *file-file* pendukung dari suatu perangkat elektronik. Saat ini ROS sudah mempunyai 2 *major version* dengan arsitektur yang berbeda, yaitu ada ROS1 dan ROS2. Pada penelitian ini, penulis menggunakan ROS1 dengan versi Melodic pada mikro komputer Jetson Nano dan menggunakan versi Noetic pada komputer server. Perbedaan versi ini di karena kan perbedaan versi Ubuntu pada Jetson Nano dan komputer server, akan tetapi ROS mampu melakukan komunikasi antar perangkat walaupun memiliki versi yang berbeda.

6. Terminator

Aplikasi ini digunakan sebagai terminal utama dalam melakukan penelitian ini, hal ini dikarenakan fiturnya lebih kaya dari pada terminal *default* milik Linux. Kelebihan terminal ini adalah mampu membelah atau membagi tampilan tabnya menjadi beberapa bagian. Dikarenakan proses untuk menjalankan *file* terjadi lewat terminal, untuk itu dibutuhkan terminal yang mampu membelah diri di dalam satu tampilan *Window.*

7. Visual Studio Code

Visual Studio Code atau yang biasa disingkat dengan VSCode merupakan kode editor yang banyak digunakan oleh developer. Kode editor ini digunakan karena kemudahan dalam auto koreksi dan kemudahan navigasi antar *file*. Selain itu fitur *debugging* dalam melakukan *tracking error* akan sangat membantu sekali.

8. Platform IO

PlatformIO merupakan sebuah *extension* yang bisa diintegrasikan dengan IDE favorit atau *text* editor seperti Arduino IDE, Atom, Eclipse, Sublime, VIM, Visual Studio Code dan lain sebagainya. Penggunaan *extension* ini dikarenakan kemudahannya dalam proses *compiling* dan *flash* ke *motherboard hoverboard*. Dengan cukup melakukan 1 kali klik maka proses *compile* dan *upload* akan

sekaligus dilakukan sehingga mempercepat proses pengembangan robot di dalam penelitian ini.

9. Creality Slicer

Creality Slicer merupakan perangkat lunak pengiris digunakan dalam pencetakan 3D untuk memotong model 3D menjadi beberapa lapisan (irisan) dan menggambar jalur pencetakan yang dapat diikuti oleh mesin. Creality Slicer adalah alat pengiris milik Creality, yang juga memproduksi banyak printer 3D populer seperti seri Ender 3. Aplikasi ini dipakai sebelum melakukan pencetakan 3D, fungsinya adalah melakukan *convert* dari *file* ekstensi umum menjadi *gcode*, sehingga 3D printer dapat mengenali *file* tersebut

10. VNC *Viewer & VNC Server*

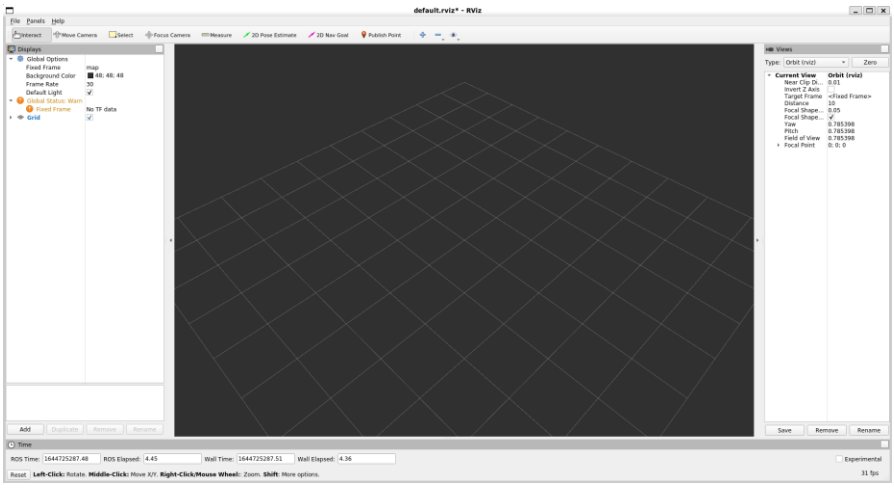
VNC *Viewer* dan VNC Server merupakan software remote-control yang memungkinkan untuk mengontrol komputer lain melalui koneksi *network*. Komputer server akan di *install* VNC *Viewer* sedangkan Jetson Nano sebagai mikro komputer akan di *install* VNC Server. Kegunaan perangkat lunak ini adalah untuk melakukan proses yang membutuhkan tampilan user interface pada Jetson Nano, di karena kan SSH tidak mampu menjalankan aplikasi GUI. Mekanisme penggunaannya kurang lebih input dari *keyboard* dan mouse dikirimkan dari satu komputer server ke Jetson Nano sehingga kita dapat mengontrol mikro komputer tanpa harus menancapkan kabel HDMI ataupun kabel LAN, jadi cukup menggunakan koneksi pada jaringan Wi-Fi yang sama.

11. *Secure Shell Protocol* (SSH)

SSH digunakan sebagai komunikasi remote dengan tampilan antar muka CLI. SSH digunakan pada penelitian ini untuk menjalankan perintah-perintah ROS ataupun sekadar melakukan *editing file*. SSH digunakan karena kecepatannya dalam melakukan proses perintah dan hemat penggunaan *memory*, dibandingkan dengan VNC yang sedikit lambat dan banyak memakan *memory*. SSH hanya digunakan pada saat-saat tertentu ketika VNC tidak memungkinkan atau sangat sulit untuk dilakukan.

12. Rviz

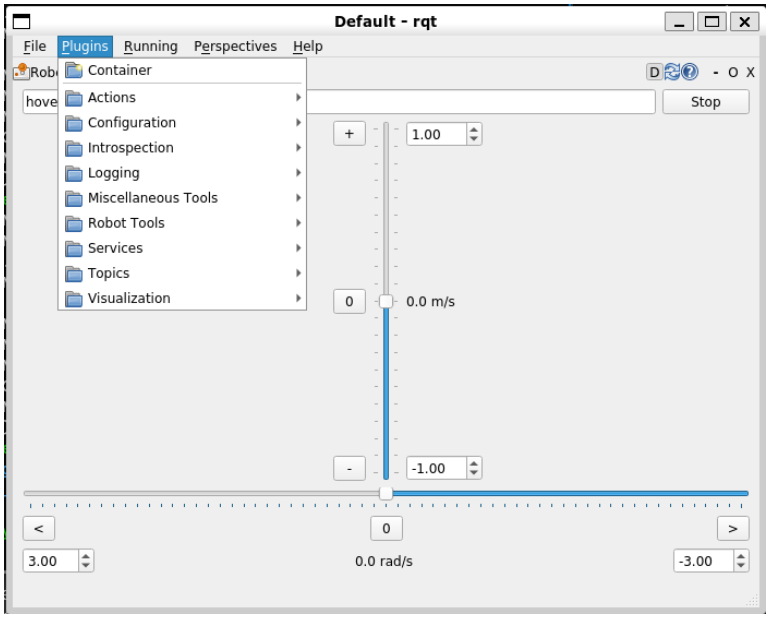
Rviz merupakan aplikasi Linux berbasis GUI yang mendukung visualisasi dari Robot Operating System (ROS). Pada penelitian ini Rviz digunakan sebagai aplikasi untuk melihat sejauh mana pembuatan peta atau mapping bekerja. Hampir setiap pengembangan ROS pasti menggunakan Rviz sebagai aplikasi visualisasinya. Rviz memiliki fitur pendukung utama untuk melakukan navigasi, yakni 2D *Nav Goal*. Fitur ini mengirimkan data koordinat pada peta yang dijadikan *fixed* frame, sehingga ketika robot sudah dilengkapi dengan sistem navigasi maka akan terbuat sebuah path planning sehingga robot akan mengikuti arahan dari jalur yang sudah dibuat dan di visualisasikan melalui Rviz. Gambar 4.12 merupakan tampilan antar muka aplikasi Rviz.



Gambar 4.12 Tampilan Antarmuka Rviz

13. rqt

rqt adalah framework perangkat lunak ROS yang mengimplementasikan berbagai alat GUI dalam bentuk *plugin*. Pengguna dapat menjalankan semua alat GUI yang ada sebagai jendela yang dapat dipasang di dalam rqt. *Tools* ini masih dapat berjalan dalam metode mandiri tradisional, tetapi rqt memudahkan untuk mengelola semua berbagai jendela di layar pada satu tampilan. Gambar 4.13 merupakan tampilan antar muka aplikasi rqt.



Gambar 4.13 Aplikasi rqt Dan *Plugins*

14. *Hoverboard driver*

Hoverboard driver adalah sekumpulan *file* dengan bahasa C yang menjembatani antara perangkat lunak dengan firmware *hoverboard*. *Hoverboard driver* berisikan *library-library* yang dibutuhkan untuk mengendalikan *hoverboard* melalui *node* ROS. *Driver* ini tidaklah ditulis dari awal oleh penulis, melainkan menggunakan *driver* dari pengembang lain yang membuat tim untuk fokus

mengembangkan *driver hoverboard*, *link* yang digunakan sebagai berikut: https://github.com/bambanggunawanid/hoverboard-driver. Driver ini sudah di fork dari sumber aslinya untuk menghindari perubahan kode di masa depan yang membuat hasil berbeda dari penelitian ini.

Dengan menggunakan *driver* ini kita dapat melakukan kontrol pada *hoverboard* dengan mudah, hal ini karena *driver* ini sudah berisikan kode untuk komunikasi firmware dengan komputer. *Hoverboard driver* memiliki beberapa parameter pada paketnya yang digunakan sebagai konfigurasi pada robot. Nilai dari parameter ini disesuaikan berdasarkan data nyata pada pengukuran robot sesungguhnya. Tabel parameter yang digunakan paket *Hoverboard driver* dapat dilihat pada tabel 4.10.

Tabel 4.10 Parameter *Hoverboard Driver*

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**
left_wheel	Nama sambungan roda kiri
right_wheel	Nama sambungan roda kanan
pose_covariance_diagonal	Diagonal matriks kovarians untuk penerbitan pose odometri
twist_covariance_diagonal	Diagonal matriks kovarians untuk penerbitan twist odometri
wheel_separation_multiplier	Ini digunakan untuk menjelaskan perbedaan antara model robot dan robot nyata.
cmd_vel_timeout	Periode yang diizinkan (dalam s) diperbolehkan antara dua perintah kecepatan yang berurutan.
base_frame_id	Digunakan untuk mengisi child_frame_id dari pesan Odometry dan TF
enable_odom_tf	Publikasikan ke TF secara langsung atau tidak
wheel_radius	Jari-jari roda. Diharapkan mereka semua memiliki ukuran yang sama.
wheel_separation	Jarak roda kiri dan kanan.
allow_multiple_cmd _vel_publishers	Menyetel ini ke true akan mengizinkan lebih dari satu penerbit pada topik masukan, ~/cmd_vel. Menyetel ini ke false akan menyebabkan pengontrol mengerem jika ada lebih dari satu penerbit di ~/cmd_vel.
linear/x/min_velocity	Kecepatan linier minimum (dalam m/s).
linear/x/max_velocity	Kecepatan linier maksimum (dalam m/s)

linear/x/max_acceleration	Percepatan linier maksimum (dalam m/s^2)
angular/z/min_velocity	Kecepatan sudut minimum (dalam rad/s). Mengatur ini ke 0,0 akan menonaktifkan rotasi berlawanan arah jarum jam.
angular/z/max_velocity	Kecepatan sudut maksimum (dalam rad/s)
angular/z/min_acceleration	Percepatan sudut minimum (dalam rad/s^2). Saat tidak ditentukan, -max_acceleration digunakan.

15. RPLIDAR ROS *driver*

RPLIDAR *driver* adalah sekumpulan *file* dengan bahasa C, *python* dan sebagainya yang menjembatani antara perangkat lunak dengan perangkat RPLIDAR. *Driver* ini diciptakan langsung oleh tim riset dari perusahaan yang menjual produk RPLIDAR A1, yaitu RoboPeak. Untuk melakukan *download driver* bisa di *download* di GitHub resmi RoboPeak, *link* yang digunakan sebagai berikut: https://github.com/bambanggunawanid/rplidar_ros. *Driver* ini sudah bisa langsung di gunakan tanpa harus melakukan pengaturan yang berarti. Tabel parameter yang digunakan paket RPLIDAR ROS *driver* dapat dilihat pada tabel 4.11.

Tabel 4.11 Parameter RPLIDAR ROS *Driver*

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**
serial_port	nama port serial yang digunakan di sistem Anda.
serial_baudrate	kecepatan baud port serial
frame_id	Frame id untuk perangkat rplidar.
inverted	menunjukkan apakah LIDAR dipasang terbalik.
angle_compensate	menunjukkan apakah pengemudi perlu melakukan kompensasi sudut.
scan_mode	mode pemindaian lidar.

16. MPU6050 dan I2C *driver*

MPU6050 merupakan *driver* sensor yang digunakan untuk menjembatani sensor IMU 6050 dengan mikro komputer atau mikro kontroler. Sedangkan I2C merupakan *library* untuk komunikasi jalur serial I2C antara kontroler dengan sensor IMU. Untuk mendapatkan driver ini, kita bisa mendownload drive MPU6050 dan I2C driver pada link berikut:

- https://github.com/bambanggunawanid/mpu6050_driver

- https://github.com/bambanggunawanid/i2c_device_ros

Driver ini bertujuan untuk membaca nilai percepatan dari sensor akselerometer, giroskop dan magnetometer. Dengan menggunakan *driver* ini kita mampu mengakses sensor IMU dengan mudah. Tabel parameter yang digunakan paket MPU6050 dan I2C *driver* dapat dilihat pada tabel 4.12.

Tabel 4.12 Parameter MPU6050 *Driver*

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**
bus_uri	I2C Bus URI digunakan untuk berkomunikasi dengan perangkat I2C
mpu_address	Alamat I2C dari MPU6050 yang biasanya memiliki alamat default 0x68
pub_rate	Frekuensi dalam Hertz yang data IMU diterbitkan
frame_id	Bingkai jika pesan IMU
axes_offsets	Offset untuk memperbaiki nilai yang salah yang disebabkan oleh misalignment. Nilai ini didapat setelah melakukan kalibrasi
ki	Konstanta PID yang digunakan dalam prosedur kalibrasi.
kp	Konstanta PID yang digunakan dalam prosedur kalibrasi
delta	Proses kalibrasi selesai ketika kesalahan mendekati nol dengan presisi yang ditetapkan oleh delta

17. Hector SLAM *library*

Hector SLAM *library* merupakan sekumpulan paket fungsi yang dipakai untuk membuat pemetaan. *Library* ini bertujuan untuk melakukan pemetaan dan lokalisasi dengan mengandalkan data dari laser. *Library* ini memiliki kemampuan melakukan *mapping* tanpa harus mengetahui kondisi sekitarnya terlebih dahulu. Fungsi-fungsi pada paket ini tersedia pada *directory* GitHub pengembangnya yaitu *Technical University* Darmstats, *link* yang digunakan sebagai berikut: https://github.com/bambanggunawanid/hector_slam. Hector SLAM *library* memiliki dari beberapa *library* lagi di dalamnya, terdiri dari *library* hector mapping, hector Geotiff, hector Trajectory, hector SLAM dan masih banyak lagi. Tabel parameter yang digunakan paket robot pose EKF dapat dilihat pada tabel 4.13.

Tabel 4.13 Parameter Hector Mapping

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**
base_frame	Nama rangka dasar robot. Ini adalah frame yang digunakan untuk pelokalan dan transformasi data pemindaian laser.
map_frame	Nama dari frame peta
odom_frame	Nama dari frame odometry
map_resolution	Resolusi peta [m]. Ini adalah panjang tepi sel grid.
map_size	Ukuran [jumlah sel per sumbu] peta. Peta berbentuk persegi dan memiliki sel kisi (map_size * map_size).
map_start_x	Lokasi asal [0.0, 1.0] bingkai /peta pada sumbu x relatif terhadap peta kisi. 0,5 di tengah.
map_start_y	Lokasi asal [0.0, 1.0] bingkai /peta pada sumbu y relatif terhadap peta kisi. 0,5 di tengah.
map_update _distance_thresh	Ambang batas untuk melakukan pembaruan peta.
map_update _angle_thresh	Ambang batas untuk melakukan pembaruan peta.
map_multi_res_levels	Jumlah tingkat petak multi-resolusi.
update_factor_free	Pengubah pembaruan peta untuk pembaruan sel gratis dalam kisaran.
update_factor_occupied	Pengubah pembaruan peta untuk pembaruan sel yang ditempati dalam kisaran
laser_z_min_value	Tinggi minimum [m] relatif terhadap bingkai pemindai laser untuk titik akhir pemindaian laser yang akan digunakan oleh sistem
laser_z_max_value	Tinggi maksimum [m] relatif terhadap bingkai pemindai laser untuk titik akhir pemindaian laser yang akan digunakan oleh sistem
pub_map_odom_transform	Tentukan apakah transformasi map->odom harus dipublikasikan oleh sistem.

tf_map_scanmatch _transform_frame_name	Nama frame saat memublikasikan scanmatcher ke transformasi peta seperti yang dijelaskan dalam parameter sebelumnya.

18. Robot Pose EKF Localization library

EKF adalah implementasi dari filter Kalman yang diperluas. Ini menggunakan model gerakan *omnidirectional* untuk memproyeksikan keadaan ke depan dalam waktu, dan mengoreksi perkiraan yang diproyeksikan menggunakan data sensor yang dirasakan. EKF digunakan baik dalam proses mapping ataupun navigasi, hal ini dikarenakan kedua proses itu membutuhkan metode untuk menggabungkan sensor agar peta yang dihasilkan menjadi lebih baik. Output dari paket ini adalah sebuah frame bernama odom_combined yang sudah dilakukan koreksi nilai kesalahan data odometry dengan menggunakan data IMU. Tabel parameter yang digunakan paket robot pose EKF dapat dilihat pada tabel 4.14.

Tabel 4.14 Parameter EKF Localization

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**
output_frame	Nama frame dari odometry, dalam hal ini adalah /odom
base_footprint_frame	Nama frame dari base footprint
freq	Frekuensi bernilai nyata dalam Hz, di mana filter menghasilkan perkiraan keadaan
sensor_timeout	Ketika sensor berhenti mengirimkan informasi ke filter, berapa lama filter harus menunggu sebelum melanjutkan tanpa sensor itu.
odom_used	Aktifkan atau nonaktifkan input sensor odometry
imu_used	Aktifkan atau nonaktifkan input sensor IMU
vo_used	Aktifkan atau nonaktifkan input sensor visual odometry
gps_used	Aktifkan atau nonaktifkan input sensor GPS
debug	Parameter yang menentukan apakah akan dijalankan dalam mode debug atau tidak.

19. ROS Navigation Stack: AMCL library

ROS navigation library merupakan sekumpulan paket fungsi yang dipakai untuk proses path planning robot. *Library* termasuk ke dalam paket eksternal dari

ROS yang dapat langsung di *install* menggunakan perintah $ sudo apt install ros-melodic-navigation. Sedangkan AMCL atau *Adaptive Monte Carlo Localization* adalah sistem lokalisasi probabilistik untuk robot yang bergerak dalam 2D. Ini menerapkan pendekatan lokalisasi Monte Carlo adaptif yang menggunakan filter partikel untuk melacak pose robot terhadap peta yang diketahui. AMCL digunakan pada lokalisasi proses navigasi robot, yaitu dengan memanfaatkan data lidar dan hasil pemetaan untuk mendapatkan lokasi dan pose terkini dari robot. Tabel parameter yang digunakan paket AMCL dapat dilihat pada tabel 4.15.

Tabel 4.15 Parameter AMCL *Localization*

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**
odom_frame_id	Frame mana yang digunakan untuk odometri.
odom_model_type	Model mana yang akan digunakan, baik "diff", "omni", "diff-corrected" atau "omni-corrected".
base_frame_id	Bingkai mana yang digunakan untuk basis robot
update_min_d	Jumlah partikel minimum yang diizinkan
update_min_a	Jumlah partikel maksimum yang diizinkan
global_frame_id	Nama kerangka koordinat yang diterbitkan oleh sistem lokalisasi
tf_broadcast	Setel ini ke false untuk mencegah amcl memublikasikan transformasi antara frame global dan frame odometri.
initial_pose_x	Mean pose awal (x), digunakan untuk menginisialisasi filter dengan distribusi Gaussian.
initial_pose_y	Pose awal mean (y), digunakan untuk menginisialisasi filter dengan distribusi Gaussian.
initial_pose_a	Pose mean awal (yaw), digunakan untuk menginisialisasi filter dengan distribusi Gaussian.

20. ROS Navigation Stack: Move base library

Di dalam paket ROS navigation stack terdapat paket *move base* yang merupakan paket utama untuk mengirimkan pesan *twist* yang nantinya di konversi. Di dalam paket ini juga terdapat *base* global planner untuk path planning secara global, DWA local planner untuk path planning local serta dan masih banyak lagi. Semua ini tentunya berisikan fungsi-fungsi yang bertujuan untuk menunjang

sistem navigasi robot. Paket *move base* melakukan *subscribe* pada beberapa topik antara lain, move_base/*goal* yang merupakan sebuah koordinat tujuan untuk di capai robot dan move_base/*cancel* yang merupakan permintaan untuk membatalkan tujuan tertentu. Output dari paket ini adalah sebuah *topic* yang digunakan untuk menggerakkan robot, yaitu cmd_vel. Parameter yang digunakan pada paket ini dapat dilihat pada tabel 4.16

Tabel 4.16 Parameter *Move Base*

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**
base_global_planner	Nama plugin untuk perencana global untuk digunakan dengan move_base
base_local_planner	Nama plugin untuk perencana lokal untuk digunakan dengan move_base
recovery_behaviors	Daftar plugin perilaku pemulihan untuk digunakan dengan move_base
controller_frequency	Laju dalam Hz untuk menjalankan loop kontrol dan mengirim perintah kecepatan ke pangkalan.
planner_patience	Berapa lama perencana akan menunggu dalam hitungan detik dalam upaya untuk menemukan rencana yang valid sebelum operasi pembersihan ruang dilakukan.
controller_patience	Berapa lama pengontrol akan menunggu dalam hitungan detik tanpa menerima kontrol yang valid sebelum operasi pembersihan ruang dilakukan.
conservative_reset_dist	Jarak dari robot dalam meter di mana rintangan akan dibersihkan dari peta biaya saat mencoba mengosongkan ruang di peta
recovery_behavior_enabled	Apakah akan mengaktifkan perilaku pemulihan move_base atau tidak untuk mencoba mengosongkan ruang
clearing_rotation_allowed	Menentukan apakah robot akan mencoba melakukan rotasi di tempat atau tidak saat mencoba mengosongkan ruang
shutdown_costmaps	Menentukan apakah akan mematikan peta biaya *node* atau tidak saat move_base dalam keadaan tidak aktif

oscillation_timeout	Berapa lama dalam detik untuk memungkinkan osilasi sebelum menjalankan perilaku pemulihan
oscillation_distance	Berapa jauh dalam meter robot harus bergerak agar dianggap tidak berosilasi
planner_frequency	Laju dalam Hz untuk menjalankan loop perencanaan global
max_planning_retries	Berapa kali untuk memungkinkan perencanaan ulang sebelum menjalankan perilaku pemulihan

Parameter paket *move base* hanya berisikan variabel untuk menentukan *plugin* dari perencanaan jalur global dan lokal. Terdapat beberapa parameter penting yang menjadi konfigurasi pada proses navigasi. Pada dokumentasi ROS hal ini disebut dengan *Component APIs. Node* move_base berisi komponen yang memiliki ROS API sendiri. Komponen ini dapat bervariasi berdasarkan nilai masing-masing yang terdiri dari global costmap, local *costmap, global planner*, *local planner*, dan *recovery behaviors*. Global costmap adalah semua yang diketahui robot dari kunjungan sebelumnya dan pengetahuan yang tersimpan misalnya peta. Sedangkan Local costmap adalah segala sesuatu yang dapat diketahui dari posisi saat ini dengan sensor yang tepat. Misalnya. orang berjalan dan benda bergerak lainnya, serta setiap dinding yang dapat dilihat.

Parameter yang ada pada costmap memiliki kesamaan hanya berbeda di penggunaan dan topik yang akan di *subscribe*, maka dari itu penjelasan mengenai parameter global costmap dan local costmap akan dijadikan satu tabel saja. Berikut ini akan dijelaskan parameter-parameter apa saja yang ada pada setiap komponen yang sudah disebutkan sebelumnya. Untuk parameter komponen global dan local costmap dapat dilihat pada tabel 4.17, sedangkan untuk komponen global planner dapat dilihat pada tabel 4.18 dan local planner 4.19.

Tabel 4.17 Parameter Konfigurasi *Costmap*

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**
global_frame	Frame global dari costmap yang mau di digunakan. Costmap merupakan halangan pada peta maka dari itu value dari parameter ini adalah /map
robot_base_frame	Nama frame dari base link robot
transform_tolerance	menentukan penundaan dalam transformasi (tf) data yang dapat ditoleransi dalam hitungan detik.

update_frequency	Frekuensi dalam Hz untuk peta yang akan diperbarui
publish_frequency	Frekuensi dalam Hz untuk peta yang akan mempublikasikan informasi tampilan.
max_obstacle_height	Ketinggian maksimum rintangan apa pun yang akan dimasukkan ke dalam peta biaya dalam meter.
obstacle_range	Jarak maksimum default dari robot tempat rintangan akan dimasukkan ke dalam costmap dalam meter.
raytrace_range	Rentang default dalam meter untuk menelusuri rintangan dari peta menggunakan data sensor
inflation_radius	Lebar jari-jari (meter) yang mengembangkan nilai costmap pada peta
<name>/observation_sources	Daftar nama sumber observasi yang dipisahkan dengan spasi.
<name>/ <source_name>/topic	Topik di mana data sensor masuk untuk sumber ini.
<name>/ <source_name>/data_type	Jenis data yang terkait dengan topik, saat ini hanya "PointCloud" dan "LaserScan" yang didukung.
<name>/ <source_name>/clearing	Apakah pengamatan ini harus digunakan untuk mengosongkan ruang kosong atau tidak.
map_topic	Topik yang menjadi langganan costmap untuk peta statis.
rolling_window	Apakah akan menggunakan versi rolling windows pada costmap. Jika parameter static_map disetel ke true, parameter ini harus disetel ke false.
track_unknown_space	Menentukan apakah akan melacak ruang apa di peta biaya yang tidak diketahui, artinya tidak ada pengamatan tentang sel yang terlihat dari sumber sensor mana pun.

Tabel 4.18 Parameter Konfigurasi Algoritme Dijkstra Global *Planner*

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**

allow_unknown	Menentukan apakah akan mengizinkan perencana membuat rencana yang melintasi ruang yang tidak diketahui atau tidak
default_tolerance	Sebuah toleransi pada titik tujuan untuk perencana. Path planner akan mencoba membuat rencana yang sedekat mungkin dengan tujuan yang ditentukan tetapi tidak lebih jauh dari default_tolerance.
visualize_potential	Menentukan apakah akan memvisualisasikan area potensial yang dihitung melalui PointCloud2 atau tidak.
use_dijkstra	Jika benar, gunakan algoritma dijkstra. Jika tidak, A*.
old_navfn_behavior	Jika karena alasan tertentu, kita ingin global_planner benar-benar mencerminkan perilaku navfn, setel ini ke true (dan gunakan default untuk parameter boolean lainnya)
lethal_cost	Biaya dari lethal area pada konfigurasi ulang secara dinamis
neutral_cost	Biaya dari netral area pada konfigurasi ulang secara dinamis
cost_factor	Faktor yang mengalikan beberapa biaya dari biaya peta pada konfigurasi ulang secara dinamis
publish_potential	Publikasikan Peta Biaya Potensial pada konfigurasi ulang dinamis
orientation_mod	Cara mengatur orientasi setiap titik
orientation_window_size	Window mana yang digunakan untuk menentukan orientasi berdasarkan turunan posisi yang ditentukan oleh mode orientasi
outline_map	Menguraikan peta biaya global dengan rintangan mematikan. Untuk penggunaan peta biaya global non-statis (jendela bergulir), ini perlu disetel ke false

Tabel 4.19 Konfigurasi Algoritme DWA *Local Planner*

Sumber: (wiki.ros.org, 2022)

Parameter	**Keterangan**

acc_lim_x	Batas percepatan x robot dalam meter/detik^2
acc_lim_y	Batas percepatan y robot dalam meter/detik^2
acc_lim_th	batas percepatan rotasi robot dalam radian/detik^2
max_vel_trans	Nilai absolut dari kecepatan translasi maksimum untuk robot dalam m/s
min_vel_trans	Nilai absolut dari kecepatan translasi minimum untuk robot dalam m/s
max_vel_x	Kecepatan x maksimum untuk robot dalam m/s.
min_vel_x	Kecepatan x minimum untuk robot dalam m/s, negatif untuk gerak mundur.
max_vel_y	Kecepatan y maksimum untuk robot dalam m/s
min_vel_y	Kecepatan y minimum untuk robot dalam m/s
max_rot_vel	Nilai mutlak kecepatan putar maksimum robot dalam rad/s
min_rot_vel	Nilai mutlak kecepatan putar minimum robot dalam rad/s
yaw_goal_tolerance	Toleransi dalam radian untuk pengontrol dalam yaw/rotasi saat mencapai tujuannya
xy_goal_tolerance	Toleransi dalam meter untuk pengontrol dalam jarak x & y saat mencapai tujuan
latch_xy_goal_tolerance	Jika toleransi tujuan terkunci, jika robot pernah mencapai lokasi tujuan xy, ia hanya akan berputar di tempatnya, bahkan jika itu berakhir di luar toleransi tujuan saat melakukannya.
sim_time	Parameter untuk simulasi. Jumlah waktu untuk meneruskan simulasi lintasan dalam hitungan detik
sim_granularity	Parameter untuk simulasi. Ukuran langkah, dalam meter, untuk mengambil antara titik pada lintasan tertentu
vx_samples	Parameter untuk simulasi. Jumlah sampel yang akan digunakan saat menjelajahi ruang kecepatan x

vy_samples	Parameter untuk simulasi. Jumlah sampel yang akan digunakan saat menjelajahi ruang kecepatan y
vth_samples	Parameter untuk simulasi. Jumlah sampel yang akan digunakan saat menjelajahi ruang kecepatan theta
controller_frequency	Parameter untuk simulasi. Frekuensi di mana pengontrol ini akan dipanggil dalam Hz. Menggunakan "searchParam" untuk membaca parameter dari ruang nama induk jika tidak disetel di ruang nama pengontrol.
path_distance_bias	Pembobotan untuk seberapa banyak pengontrol harus tetap dekat dengan jalur yang diberikan
goal_distance_bias	dia menimbang berapa banyak pengontrol harus berusaha mencapai tujuan lokalnya, juga mengontrol kecepatan
occdist_scale	Pembobotan untuk seberapa banyak pengontrol harus berusaha menghindari rintangan
forward_point_distance	Jarak dari titik pusat robot untuk menempatkan titik penilaian tambahan, dalam meter
stop_time_buffer	Jumlah waktu robot harus berhenti sebelum tabrakan agar lintasan dianggap valid dalam hitungan detik
scaling_speed	Nilai absolut dari kecepatan untuk mulai menskalakan jejak robot, dalam m/s
max_scaling_factor	Faktor maksimum untuk menskalakan jejak robot
publish_cost_grid	Apakah akan mempublikasikan kisi biaya yang akan digunakan perencana saat merencanakan atau tidak
oscillation_reset_dist	Seberapa jauh robot harus berjalan dalam meter sebelum tanda osilasi direset
prune_plan	Menentukan apakah akan memakan rencana saat robot bergerak di sepanjang jalan. Jika disetel ke true, poin akan jatuh dari akhir rencana setelah robot bergerak 1 meter melewatinya.