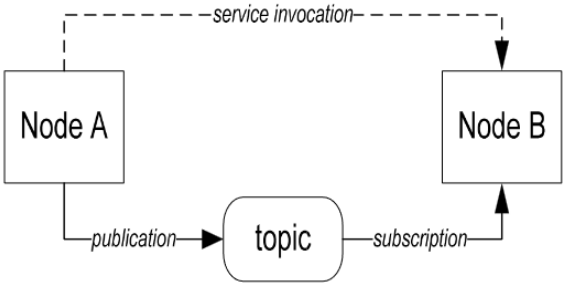
BAB II

TINJAUAN PUSTAKA

2.1 Robot Operating System

Robot telah menjadi topik pembicaraan di banyak industri dan bisnis selama 20 tahun terakhir. Kendaraan darat, operasi militer, analisis infrastruktur pipa, dan sejumlah otomasi industri yang signifikan. Dalam banyak kasus, hardcoding Raspberry Pis dan perangkat mikrokontroler lainnya memberikan dasar untuk manipulasi dan implementasi robot[8].

Robot Operating System (ROS) adalah kerangka kerja perangkat lunak sumber terbuka untuk mengendalikan robot yang sebagian besar didasarkan pada sistem operasi UNIX [7]. Selain abstraksi lapisan perangkat keras, kontrol perangkat tingkat rendah, implementasi fungsionalitas yang sering digunakan, transfer pesan antar proses, dan manajemen paket, ROS menawarkan fitur-fitur yang diharapkan pengguna dari sistem operasi. Selain itu, ROS juga menawarkan alat dan pustaka untuk memperoleh, membuat, menulis, dan mengeksekusi kode pada beberapa platform. Mendukung penggunaan ulang kode dalam penelitian dan pengembangan robotika adalah tujuan utama ROS. ROS adalah sistem terdistribusi dari proses (node) yang memungkinkan penyesuaian runtime dan fleksibilitas eksekusi. Operasi ini dapat diatur ke dalam *Stacks* dan *Packages*, yang mudah didistribusikan dan dibagikan. Selain itu, ROS menyediakan jaringan federasi repositori kode yang memungkinkan kerja sama terdistribusi. Ide dasar ROS digambarkan pada **Gambar 2.1**

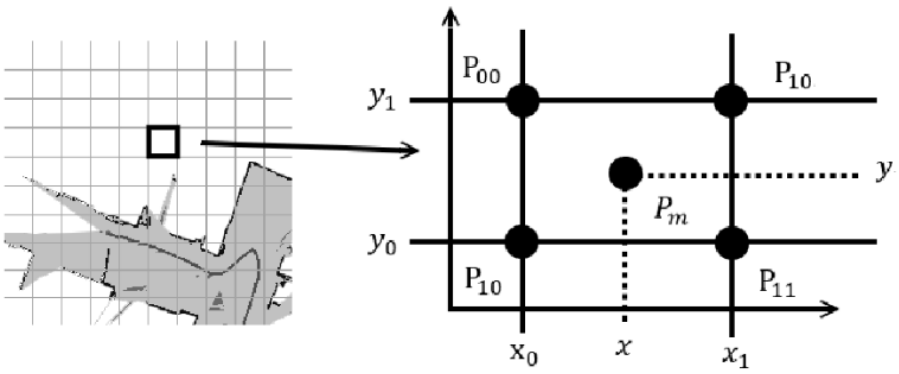


Gambar 2.1. Konsep Dasar ROS [7]

2.2 Hector SLAM

Hector SLAM adalah alat implementasi sumber terbuka dari teknik SLAM 2D yang diusulkan di (Kohlbrecher et al., 2011)[5]. Teknik ini mengandalkan data pemindaian laser 2D untuk mendeteksi landmark dan membangun peta jaringan di sekitarnya. Karena odometer roda memiliki robot bergerak beroda yang terkenal tidak dapat diandalkan karena faktor selip, Hector SLAM dirancang untuk tidak memanfaatkan data odometri, melainkan sepenuhnya bergantung pada pencocokan pemindaian data LIDAR yang cepat dengan kecepatan pembaruan LIDAR (Light Detection and Ranging) penuh. Agar proses pemetaan berhasil, diperlukan algoritma pemetaan Hector Slam yang merupakan Slam berbasis *scan-matching* [9]. Dalam lingkungan pemetaan, kisi hunian dijelaskan oleh Elfes yaitu, "Representasi yang menggunakan pengubinan ruang multidimensi (2D atau 3D) ke dalam sel, di mana setiap sel menyimpan estimasi probabilistik dari keadaannya. Variabel keadaan *s(C)* yang terkait dengan sel *C* dari kisi-kisi hunian didefinisikan sebagai variabel acak diskrit dengan dua keadaan, terisi dan kosong" [9]. Sel kisi hunian ditunjukkan pada **Gambar 2.2**. Untuk membahas *scan-matching*, dalam mempertimbangkan sebuah titik pada peta kontinu, Pm, nilai hunian, *M(Pm)*, gradien akan ditunjukkan padamemiliki bentuk [10]:

$$\nabla M(P_m) = \left(\frac{\partial M(P_m)}{\partial x}, \frac{\partial M(P_m)}{\partial y}\right) \quad (1)$$



Gambar 2.2. Representasi sel kisi hunian menggunakan metode Bilinear [10].

filterisasi bilinear digunakan untuk menginterpolasi sel sub-kisi untuk memperkirakan turunan hunian & probabilitas. Dengan begitu, nilai diskrit dari permukaan peta akan menjadi kontinu pada setiap titik tertentu pada peta [10]. Dan dengan menggunakan metode interpolasi linier dengan koordinat integer terdekat **P_{00}**, **P_{01}**, **P_{10}**, **P_{11}**, dapat ditunjukkan dengan:

$$M(P_m) \approx \frac{y-y0}{y1-y0}\left(\frac{x-x0}{x1-x0}M(P_{11}) + \frac{x-x0}{x1-x0}M(P_{01})\right) \quad (2)$$

$$+\frac{y-y0}{y1-y0}\left(\frac{x-x0}{x1-x0}M(P_{10}) + \frac{x-x0}{x1-x0}\ M(P_{00})\right)$$

Selain itu, turunan dari peta pada titik tertentu dapat ditampilkan sebagai [10]:

$$\frac{\partial M(P_m)}{\partial x} \approx \frac{y-y0}{y1-y0}\left(M(P_{11}) + M(P_{01})\right) + \frac{y-y0}{y1-y0}\left(M(P_{10}) + M(P_{00})\right) \quad (3)$$

$$\frac{\partial M(P_m)}{\partial y} \approx \frac{x-x0}{x1-x0}\left(M(P_{11}) + M(P_{01})\right) + \frac{y-y0}{y1-y0}\left(M(P_{10}) + M(P_{00})\right) \quad (4)$$

Mengoptimalkan penyelarasan titik akhir laser yang relevan dengan peta yang dibuat sejauh ini merupakan dasar untuk prosedur pencocokan pemindaian.

Perkiraan Gauss-Newton digunakan untuk mengantisipasi posisi berikutnya tanpa mencari data yang sesuai di antara titik-titik akhir. Transformasi benda tegar ditunjukkan oleh ekspresi $\xi = (px, py, \psi)^T$, di mana ψ = *yaw*. Tujuan utamanya adalah untuk mengurangi transformasi ini dan mendapatkan perataan optimal antara data pemindaian laser dan peta. Untuk mencapainya, minimalkan:

$$\xi^* = argmin\, \Sigma_{i=1}^{n} [1 - M(S_i(\xi))]^2 \tag{5}$$

Di mana, $\boldsymbol{S_i}$ adalah koordinat dunia dari titik akhir pemindaian, $S_i=(S_{i,}x, S_{i,}y)^T$ [2]. Sehingga, $S_i(\xi)$ dapat didefinisikan sebagai berikut:

$$S_i(\xi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{pmatrix} \begin{pmatrix} S_i\,, x \\ S_i\,, y \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix} \tag{6}$$

Untuk menyelesaikan masalah kuadrat terkecil non-linier, jaringan Gauss-Newton digunakan. Kami mengasumsikan definisi berikut dari fungsi target r_i:

$$r_i = 1 - M(S_i(\xi)) \tag{7}$$

Fungsi $M(Si(\xi))$ mengembalikan nilai peta pada koordinat yang disajikan oleh $Si(\xi)$. Mengingat bahwa estimasi awal adalah ξ, kita perlu menghitung $\nabla\xi$ [11]. Dari metode Newton, relasi perulangan untuk meminimalkan fungsi target adalah:

$$\xi_t = \xi_{t-1} - H^{-1}G \tag{8}$$

G adalah vektor gradien dan H adalah matriks Hessian. Kedua hal tersebut diperoleh dengan mengabaikan suku turunan orde dua. Dengan mengasumsikan posisi robot ($\nabla\xi$) telah berubah dengan gerakan yang sangat kecil yang dimana cukup kecil untuk diabaikan. Maka dapat diperoleh persamaan sebagai berikut:

$$G = \Sigma_{i=1}^{n} \frac{\partial r_i}{\partial \xi} \tag{9}$$

$$= [\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}]^T[1 - M\big(S_i(\xi)\big) - \nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}\ \nabla\xi]$$

$$= [\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}]^T[1 - M\big(S_i(\xi)\big)] \quad (10)$$

Seperti yang telah disebutkan sebelumnya, H dapat didefinisikan sebagai:

$$H = \sum_{i=1}^{n} \frac{\partial ri}{\partial \xi}\frac{\partial ri}{\partial \xi} = \quad (11)$$

$$\sum_{i=1}^{n} [\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}]^T[\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}\ \nabla\xi\ ]$$

Dan turunan dari $Si(\xi)$ dapat direpresentasikan sebagai [10]:

$$\frac{\partial S_i(\xi)}{\partial \xi} = \begin{pmatrix} 1 & 0 & -\sin(\psi) & -\cos(\psi) & S_i\,,x \\ 0 & 1 & \cos(\psi) & \sin(\psi) & S_i\,,y \end{pmatrix} \quad (12)$$

Dengan menyelesaikan $\nabla\xi$, persamaan dalam meminimalkan masalah akan menjadi:

$$\nabla\xi = H{-}1[\nabla MSi\xi\partial Si\xi\partial\xi]T[1{-}MSi\xi] \quad (13)$$

Sehingga didapar algoritma *hector scan matching* seperti pada **Gambar 2.3**.

1: **hector_scan_matching**$(\xi_{t-1}, s_{i,x}, s_{i,y}, M)$**:**
2: $S_i(\xi_{t-1}) = \begin{pmatrix} cos(\psi_{t-1}) & -sin(\psi_{t-1}) \\ sin(\psi_{t-1}) & cos(\psi_{t-1}) \end{pmatrix}\begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix} + \xi_{t-1}$
3: $G = \sum_{i=1}^{n}\left[\nabla M\big(S_i(\xi_{t-1})\big)\frac{\partial S_i(\xi_{t-1})}{\partial \xi}\right]^T\left[1 - M\big(S_i(\xi_{t-1})\big)\right]$
4: $H = \sum_{i=1}^{n}\left[\nabla M\big(S_i(\xi_{t-1})\big)\frac{\partial S_i(\xi_{t-1})}{\partial \xi}\right]^T\left[\nabla M\big(S_i(\xi_{t-1})\big)\frac{\partial S_i(\xi_{t-1})}{\partial \xi}\right]$
5: $\Delta\xi = H^{-1}G$
6: $\xi_t = \xi_{t-1} + \Delta\xi$
7: return ξ_t

Gambar 2.3. Algoritma *Hector Scan Matching*

Keterangan :

ξ_t : pose robot dalam waktu t

M : map 2D yang telah dipetakan

S_i : transformasi dari end-point dalam frame robot ke word frame

r_i : target fungsi pendekatan Gauss-Newton

G : gradien vector dari target fungsi pendekatan Gauss-Newton

H : hessian matriks dari target fungsi pendekatan Gauss-Newton

ψ : *yaw*

$S_i(\xi)$: koodinat titik akhir pemindaian

M(Pm) : nilai okupansi peta kontinue

M(Si(ξ)) : nilai peta di S(ξ) yang merupakan koordinat dunia dari titik akhir pemindaian

Langkah-langkah utama Hector SLAM ditunjukkan pada Algoritma 4 [13]. Langkah-langkah utama Hector SLAM ditunjukkan algoritma pada **Gambar 2.4** dan **Gambar 2.5**. Untuk memperkirakan gerakan robot dan secara bertahap memperbaharui sikap robot saat ini, algoritma ini mencocokkan pemindaian LiDAR masukan dengan kisi hunian peta yang lengkap. Karena menggunakan metode *scan-matching* ke peta yang diwarisi dari Gauss-Newton, apa yang disebut sebagai *dead reckoning* ini rentan terhadap penumpukan kesalahan dan kebisingan dalam pengukuran sensor. Hector SLAM berisi banyak kisi peta dengan berbagai resolusi untuk mengatasi masalah ini. Dengan resolusi yang menurun secara eksponensial mulai dari *r, 2r, 4r, ..., $2^{n-1}r$*, metode ini secara khusus membangun piramida dari n peta grid, $M_0, M_1, ..., M_{n-1}$. Setelah itu, pencocokan pemindaian awal menggunakan teknik penyelarasan kasar ke halus (baris 5-7), di mana estimasi postur tubuh dari tahap sebelumnya digunakan sebagai tebakan yang terdidik pada

tahap berikutnya yang lebih halus. Untuk menyimpan dan memperbarui beberapa peta grid, teknik peta multi-resolusi ini menghasilkan peningkatan penggunaan memori dan biaya komputasi (baris 10) [13].

Algorithm 4 Original Hector SLAM

1: Create grid maps $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_{n-1}$ with exponentially decreasing resolutions $r, 2r, \ldots, 2^{n-1}r$
2: Initialize a current pose: $\boldsymbol{\xi}_0 \leftarrow [0, 0, 0]$
3: **for** $t = 1, \ldots,$ **do**
4: Receive a new LiDAR scan $\mathcal{S}_t$ and remove outliers
▷ **Gauss-Newton scan matching**
5: $\boldsymbol{\xi}_{t,n-1} \leftarrow \boldsymbol{\xi}_{t-1}$
6: **for** $i = n-1, n-2, \ldots, 0$ **do**
7: Perform Gauss-Newton scan matching using $\mathcal{M}_i$. Improve pose estimate $\boldsymbol{\xi}_{t,i}$ according to the cost function $J(\boldsymbol{\xi}_{t,i})$ defined as follows:

$$J(\boldsymbol{\xi}_{t,i}; \mathcal{M}_i, \mathcal{S}) = \sum_{k=1}^{N} \left(1 - \mathcal{M}_i(h(\boldsymbol{\xi}_{t,i}, z_k))\right)^2$$

Resulting pose estimate is used as the initial estimate for the next iteration ($\boldsymbol{\xi}_{t,i-1} \leftarrow \boldsymbol{\xi}_{t,i}$).
8: Update the current pose: $\boldsymbol{\xi}_t \leftarrow \boldsymbol{\xi}_{t,0}$
▷ **Map update**
9: **for** $i = 0, \ldots, n-1$ **do**
10: Update $\mathcal{M}_i$ using scan $\mathcal{S}_t$ and pose $\boldsymbol{\xi}_t$
11: **return** Trajectory $\{\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \ldots\}$, grid map $\mathcal{M}_0$

Gambar 2.4. Algoritma Hector SLAM [13]

Algoritma Asli Hector SLAM

1 : buat peta kisi $M_0, M_1, \ldots, M_{n-1}$ dengan resolusi yang menurun secara eksponensial $r, 2r, \ldots, 2^{n-1}r$
2 : menginisialisasi pose saat ini: $\xi_0 \leftarrow [0,0,0]$
3 : **for** $t = 1, \ldots,$ **do**
4 : menerima pemindaian LiDAR baru S_t dan menghapus outliers
▷ **Gauss-Newton scan matching**
5 : $\xi_{t,n-1} \leftarrow \xi_{t-1}$
6 : **for** $i = n-1, n-2, \ldots, 0$ **do**
7 : lakukan Gauss-Newton scan matching menggunakan M_i perbaiki perkiraan pose $\xi_{t,i}$ sesuai dengan fungsi biaya $J(\xi_{t,i})$ tentukan sebagai berikut:

$$J(\xi_{t,i}, M_i, S) = \sum_{k=1}^{N} (1 - M_i\left(h(\xi_{t,i}, zk)\right))^2$$

perkiraan pose yang dihasilkan digunakan sebagai perkiraan awal untuk iterasi berikutnya $\xi_{t,i-1} \leftarrow \xi_{t,i}$
8 : perbarui pose saat ini: $\xi_t \leftarrow \xi_{t,0}$
▷ **Perbarui peta**
9 : **for** $i = 0, \ldots, n-1$ **do**
10 : Perbarui M_i menggunakan pemindaian S_t dan pose ξ_t
11 : Kembali Trajectory $\{\xi_0, \xi_1, \ldots\}$, peta kisi M_0

Gambar 2.5. Algoritma Hector SLAM Bahasa Indoneisa

1.3 MSE (*Mean Square Error*)

Penaksir metrik yang paling banyak digunakan untuk mengukur kualitas gambar adalah MSE. Nilai mendekati nol adalah nilai terbaik untuk metrik referensi lengkap ini. Ini adalah momen keliru yang kedua. Kesalahan kuadrat rata-rata dicampur dengan bias dan varians penduga. Dalam kasus penduga yang tidak bias, MSE adalah varian penduganya. Satuan pengukurannya sama dengan kuadrat besaran yang dihitung, misalnya varians. MSE, yang juga dikenal sebagai deviasi standar varians, memperkenalkan Root-Mean-Square Error (RMSE) atau Root-Mean-Square Deviation (RMSD)[20].

$$MSE = \frac{1}{mn} \sum_{0}^{m-1} \sum_{0}^{n-1} \|f(i,j) - g(i,j)\|^2 \tag{14}$$

Keterangan:

f : pose robot dalam waktu t

m : map 2D yang telah dipetakan

g : transformasi dari end-point dalam frame robot ke word frame

n : target fungsi pendekatan Gauss-Newton

1.4 PSNR (*Peak Signal-to-Noise Ratio*)

Rasio signal-to-noise puncak adalah metrik yang umum digunakan untuk mengevaluasi kualitas gambar dengan membandingkannya dengan gambar asli, dengan asumsi keduanya memiliki resolusi yang sama. Kekuatan sinyal tertinggi, yang diwakili oleh gambar asli, dibandingkan dengan kekuatan noise, yang didasarkan pada perbedaan antara gambar asli dan gambar yang diproses. Desibel

(dB) digunakan untuk menyatakan nilai PSNR, yang memberikan angka yang menunjukkan seberapa mirip dua gambar satu sama lain. Jika nilai dalam proses PSNR ini memiliki nilai yang tinggi maka dapat dikatakan gambar memiliki *noise* yang rendah, sedangkan jika PSNR memiliki nilai yang rendah maka *noise* yang dihasilkan dalam hasil pembentukan gambar sangat tinggi[18][19]. jika dalam suatu algoritma/sistem memproses data yang direpresentasikan dalam bentuk gambar dalam PSNR dikatakan bahwa semakin tinggi nilai PSNR maka semakin baik proses algoritma/sistem memproses rekonstruksi data menjadi gambar.

$$PSNR = 20 \log_{10} \left(\frac{MAX_f}{\sqrt{MSE}} \right) \tag{15}$$

Keterangan :

MAX f : merupakan nilai sinyal maksimum yang ada pada gambar asli kami yang "dikenal baik"

1.5 SSIM (*Structural Similarity Index*)

Metode indeks SSIM, metrik pengukuran kualitas dihitung, tiga faktor utama yaitu kecerahan, kontras, dan istilah struktural atau korelasi digunakan untuk membuat metrik pengukuran kualitas. Indeks ini merupakan hasil perkalian ketiga faktor tersebut [21]. Tiga klausa berikut dapat digunakan untuk menyatakan metode Indeks Kesamaan Struktural:

$$\mathrm{SSIM}(x,y) = \left[l(x,y) \right]^{\alpha} \cdot \left[c(x,y) \right]^{\beta} \cdot \left[s(x,y) \right]^{\gamma} \tag{16}$$

Di sini, l adalah luminans (digunakan untuk membandingkan kecerahan antara dua im-

usia), *c* adalah kontras (digunakan untuk membedakan rentang antara wilayah paling terang dan paling gelap dari dua gambar) dan *s* adalah struktur (digunakan untuk membandingkan pola luminans lokal antara dua gambar untuk menemukan kesamaan dan ketidaksamaan gambar) dan α, β dan γ adalah konstanta positif.

1.6 Arduino Nano

Mikrokontroler merupakan *chip* atau *integrated circuit* (ICs) yang dapat diprogram menggunakan komputer, Arduino merupakan platform elektronik *open source* yang berisi komponen utama yaitu chip mikrokontroler jenis *Advanced Versatile Risc* (AVR) dari sebuah PC. Tujuan penyematan pogram pada mikrokontroler adalah agar rangkaian elektronik dapat membaca masukan, mengolah masukan dan kemudian menghasilkan keluaran yang diinginkan. Oleh karena itu, mikrokontroler merupakan "otak" dari input, output dan pemorosesan pada rangkaia elektronik.

Mikrokontroler Atmega328P disertakan pada papan pengembangan Arduino Uno atau Arduino Nano. Arduino Nano dari segi bentuk menjadi jenis terkecil, Arduino Mega jenis memiliki pin paling banyak dan Arduino Uno adalah papan yang paling umum digunakan. Bentuk fisik dari development board seperti pada **Tabel 2.1**. di bawah ini. Berikut adalah **Gambar 2.7.** Jenis Arduino

Tabel 2.1 spesifikasi development board

Spesifikasi	**Uno**	**Nano**	**Mega**
Mikrokontroller	ATMEGA328P	ATMEGA328P	ATMEGA2560
Arsitektur	8bit	8bit	8bit

Spesifikasi	Uno	Nano	Mega
Dimensi	68.6mm x 53.3mm	43.1mm x 18.5mm	101.5mm x 53.3mm
Tegangan input	5V/7-12V	5V/7-12V	5V/7-12V
Kecepatan CPU	16 MHZ	16 MHZ	16 MHZ
I/O Analog	6/0	8/0	16/0
I/O Digital / PWM	14/6	14/6	54/15
SRAM	½ kB	½ kB	½ kB
Memori Flash	32 kB	32 kB	32 kB
Spesifikasi	**Uno**	**Nano**	**Mega**
Mikrokontroller	ATMEGA328P	ATMEGA328P	ATMEGA2560

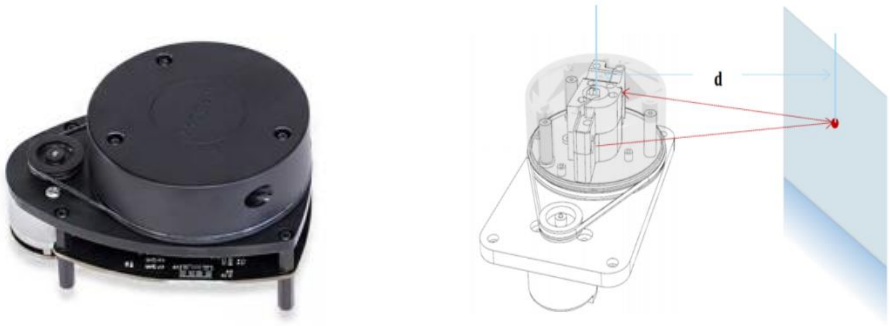
Gambar 2.6 Jenis Arduino

Dengan menimbang dan memperhatikan kebutuhan penelitian dari jumlah pin yang akan digunakan, juga kemampuan proses maka Arduino Nano merupakan mikrokontroler yang tepat untuk digunakan dalam penelitian ini karena jumlah pin

yang cukup digunakan untuk memproses intruksi penggerakan robot dari ROS-master.

1.7 RPLIDAR A1M8

RPLidar A1M8 dibuat oleh Slamtec. Pada **Gambar 2.7** menunjukkan bentuk dan konsep pengukurannya, sedangkan pada **Tabel 2.3** mencantumkan nilai lembar data utamanya. Jarak ke objek yang sedang diukur ditentukan dengan mengambil sampel dan memproses sinyal balik dari pemindai setelah menghasilkan sinyal laser inframerah termodulasi yang dipantulkan oleh objek target. Kumpulan pasangan sudut dan jarak dikembalikan dengan setiap putaran spinner tempat pemindai dipasang karena memiliki skema pengkodean sudut [14].



(a) **(b)**

Gambar 2.7. (a) Sensor RPLidar A1M8, **(b)** Prinsip Pengukuran [14]

Tabel 2.2. Parameter RPLidar A1M8

Parameter	Value (unit)
Dimension	90*70*60 mm
Weight	200 g
Maeasuring range	200-6000 mm
Angular range	360°

Parameter	Value (unit)
Scan frequency	10 Hz
Resolution	< 0.5 mm
Angular resolusion	<1°
Temperature range	0-45℃
Power consumption	2W

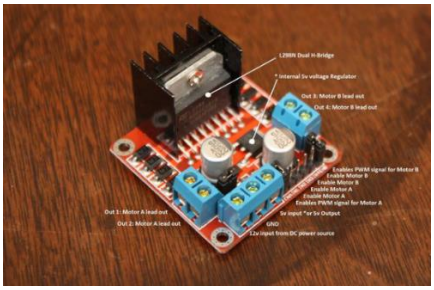
1.8 Motor Driver L298N

Dua motor DC dapat dikontrol kecepatan dan arahnya oleh IC jembatan-H L298N pada **Gambar 2.8** [15]. Motor dengan tegangan DC antara 5 dan 35 volt dan arus puncak hingga 2A dapat digunakan dengan modul ini Modul ini memiliki dua blok terminal sekrup untuk motor A dan B, serta blok terminal sekrup lainnya untuk pin Ground, pin VCC motor, dan pin 5V yang dapat digunakan sebagai input atau output. **Tabel 1** menampilkan penetapan pin untuk Modul H-Bridge ganda L298N. IN1 dan IN2 pada papan L298N menggunakan pin digital yang ditetapkan dari HIGH ke LOW atau LOW ke HIGH untuk mengatur arah.

Tabel 2.3. Penetapan pin untuk L298 [15]

PIN	Keterangan
Out 1	Motor A mengarah keluar
Out 1	Motor A mengarah keluar
Out 1	Motor B mengarah keluar
Out 1	Motor B mengarah keluar
GND	Ground
5V	Input tegangan

PIN	Keterangan
In 1	Mengaktifkan motor A
In 2	Mengaktifkan motor A
In 3	Mengaktifkan motor B
In 4	Mengaktifkan motor B
ENA	Mengaktifkan sinyal PWM untuk motor A
ENB	Mengaktifkan sinyal PWM untuk motor B



Gambar 2.8. Motor Driver L298N [15]

1.9 DC Motor 25GA370 w/Encoder

Untuk menjalankan swarm robot ini menggunakan motor DC dengan sensor encoder yang dihubungkan ke driver motor L298N sebagai pengatur kecepatan dan pengatur arah putaran dari motor DC. Sensor encoder digunakan sebagai umpan balik yang dapat mendeteksi kecepatan motor DC. Kecepatan motor pada motor DC berpengaruh terhadap nilai PWM selain itu untuk menggerakkan roda pada permukaan berlantai dibutuhkan kecepatan motor yang sesuai. Pada motor DC ini kecepatan maksimalnya yaitu sebesar 1000 RPM dan dibutuhkan pengujian untuk melihat berapa nilai kecepatan motor untuk menggerakkan roda pada permukaan berlantai dengan beban komponen lainnya. Berikut ini pada Gambar 3.6 tampak dari Motor DC 25GA370 12V 1000RPM w/ Encoder.

Tabel 2.4. Spesifikasi DC Motor 25GA370 w/Encoder

Parameter	value
Perbandingan Gear	1:46
Kecepatan	1000 rpm
Tegangan Kerja	12V
Shaft diameter	4 mm
Arus Maksimum	1.3 A

Gambar 2.9 DC Motor 25GA370 w/Encoder

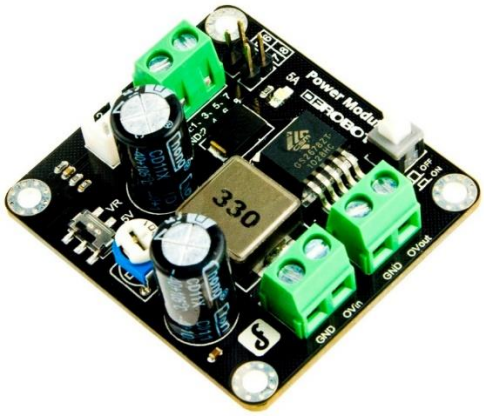
1.10 DFROBOT DFR0205

Power Modul berukuran kecil 5A 350KHz 25V Buck DC to DC Converter. Itu dapat mengubah tegangan DC apa pun antara 3.6V-25V ke tegangan yang dapat dipilih dari 3.3V ke 25V.Anda dapat memilih tegangan output langsung 5V dengan sakelar atau menyesuaikan tegangan output dengan resistor biru & putih. Nyaman bagi Anda untuk memilih tiga antarmuka output yang berbeda. Antarmuka Ovout dapat menampilkan tegangan input asli. sehingga dapat digunakan sebagai kekuatan modul lain. Tombol ON/OFF pada board dapat ditarik tinggi untuk

menghidupkan modul konverter dan rendah untuk mematikannya. Modul ini dipilih karena sudah cukup untik mengakomodoasi semua komponen yang terdapat pada robot(sumber : wiki.dfrobot.com).

Tabel 2.5. Spesifikasi Modul DFR0205

Parameter	value
Input voltage range	3.6V-25V
Output adjustable range	3.3V-25V
output current	5A@5V
Max Output Power	25W
Switching Frequency	350KHZ
Size	46x50x20mm



Gambar 2.10. Modul DFR0205 (sumber : wiki.dfrobot.com)