

BAB II

DASAR TEORI

Bab ini memuat dasar teori yang mendukung perancangan dan pengujian sistem penggerak serta integrasi LiDAR. Persamaan dibatasi pada perhitungan yang digunakan di Bab III dan analisis hasil di Bab IV, termasuk transformasi koordinat yang diperlukan untuk *local scan matching*.

2.1 Tinjauan Pustaka

Tinjauan pustaka utama pada Tabel 2.1 menggunakan artikel jurnal terbit tahun 2022–2025 yang relevan dengan kendali motor, umpan balik *encoder*, dan robot pemanen. Referensi klasik tetap digunakan pada subbab dasar teori untuk menjelaskan konsep serta asal metode Ziegler–Nichols.

Tabel 2. 1 Tinjauan pustaka penelitian terdahulu

No	Judul Penelitian	Keterangan	Persamaan dan Perbedaan
1	<i>Motion System of a Four-Wheeled Robot Using a PID Controller Based on MPU and Rotary Encoder Sensors (Sagita et al., 2024)</i>	Menerapkan PID, rotary <i>encoder</i> , dan MPU6050 pada robot empat roda omnidireksional. <i>Encoder</i> digunakan untuk odometri, sedangkan MPU6050 mendukung informasi orientasi [6].	Sama-sama menggunakan STM32 dan <i>encoder</i> untuk kendali gerak. Penelitian ini menggunakan dua roda penggerak diferensial dan PI; bukan konfigurasi empat roda omnidireksional.
2	<i>Speed Control of Wheeled Mobile Robot by Nature-Inspired Social Spider Algorithm-Based PID Controller (Khan et</i>	Membahas kendali kecepatan robot beroda dengan penentuan parameter PID berbasis algoritma Social Spider [7].	Sama-sama membahas kendali kecepatan motor. Penelitian ini menggunakan PI dengan parameter awal dari pengujian <i>step</i> manual Ziegler–Nichols pertama, tanpa optimasi Social Spider.

	<i>al., 2023)</i>		
	<i>Optimal PID Control of a Brushed DC Motor with an Embedded Low-Cost Magnetic Quadrature Encoder for Improved Step Overshoot and Undershoot Responses in a Mobile Robot Application (Bitriá dan Palacín, 2022)</i>	Membahas kendali motor DC brushed dengan <i>encoder</i> kuadratur magnetik berbiaya rendah. Optimasi pengendali diarahkan untuk memperbaiki respons <i>overshoot</i> dan <i>undershoot</i> terhadap masukan <i>step</i> [8].	Sama-sama menggunakan umpan balik <i>encoder</i> dan mengevaluasi respons <i>step</i> . Penelitian ini memakai PI pada dua motor penggerak dan penentuan parameter awal secara manual.
3			
	<i>Motor Speed Control of Four-wheel Differential Drive Robots Using a New Hybrid Moth-flame Particle Swarm Optimization (MFPSO) Algorithm (Reda et al., 2025)</i>	Menerapkan optimasi MFPSO untuk parameter PID pada motor DC penggerak empat roda. Kinerja dibandingkan dengan Ziegler–Nichols dan beberapa algoritma optimasi lainnya [3].	Sama-sama membahas kendali motor robot diferensial dan Ziegler–Nichols sebagai pembanding. Penelitian ini menggunakan PI, dua motor penggerak, dan penentuan parameter awal manual tanpa MFPSO.
4			

Keempat penelitian pada Tabel 2.1 menggunakan umpan balik *encoder* dan mengevaluasi respons terhadap masukan *step*, namun penentuan parameternya dilakukan melalui algoritma optimasi. Berbeda dari keempatnya, penelitian ini membatasi kendali pada PI untuk dua motor penggerak. Nilai awal K_p dan K_i dihitung secara manual dari respons *step open loop* menggunakan Ziegler–Nichols metode pertama, kemudian dievaluasi melalui pengujian *closed loop*. Nilai parameter dan hasil uji penelitian lain tidak digunakan sebagai hasil pengujian robot ini.

2.2 Robot Pemanen (*Harvesting Robot*)

Robot pemanen mengintegrasikan platform bergerak, sistem persepsi atau navigasi, manipulator, dan end-effector untuk mendukung proses pemanenan. Droukas et al. [2] mengulas platform robot pertanian beserta teknologi pendukungnya, termasuk persepsi, perencanaan gerak, navigasi, dan *gripper*. Dalam penelitian ini, pembahasan difokuskan pada

sistem penggerak platform sebagai penopang operasi pemetikan, bukan pada pengembangan seluruh subsistem pemanen.

Penerapan navigasi robot di *greenhouse* telah diteliti menggunakan LiDAR dan SLAM, antara lain oleh Jiang et al. [9]. Kondisi lintasan tetap harus ditentukan sesuai lokasi pengujian. Pada penelitian ini, lintasan dibatasi pada lantai rata dengan tanaman hidroponik yang tersusun sepanjang lorong. Platform diarahkan bergerak lurus dengan kecepatan stabil dan berhenti pada posisi pengamatan yang ditentukan. Karena itu, keseragaman kecepatan roda kiri dan kanan menjadi kebutuhan pengujian sistem penggerak.

2.3 Robot *Mobile* Beroda Empat

Robot *mobile* beroda empat merupakan konfigurasi platform yang menggunakan empat roda sebagai tumpuan gerak translasi. Sebagian roda berfungsi sebagai roda penggerak yang menerima torsi dari motor, sedangkan sisanya berfungsi sebagai roda penopang yang berputar bebas mengikuti gerak robot. Siegwart *et al.* [10] menyatakan bahwa konfigurasi roda menentukan tiga aspek yang saling bertukar, yaitu stabilitas, kemampuan manuver, dan kemampuan kendali. Konfigurasi empat roda memberikan luas tumpuan yang lebar sehingga kestabilan robot terhadap gangguan permukaan lebih baik dibandingkan konfigurasi tiga roda.

Pada penelitian ini digunakan konfigurasi penggerak diferensial (*differential drive*), yaitu dua roda penggerak yang masing-masing digerakkan satu motor DC PG45 pada sisi kiri dan sisi kanan robot, sedangkan dua roda lainnya berfungsi sebagai roda penopang. Robot mampu bergerak maju serta berbelok ke kanan dan kiri melalui perbedaan kecepatan kedua motor. Namun, analisis kendali dalam penelitian ini dibatasi pada gerak lurus sehingga fokus pengujian adalah keseragaman kecepatan roda kiri dan kanan.

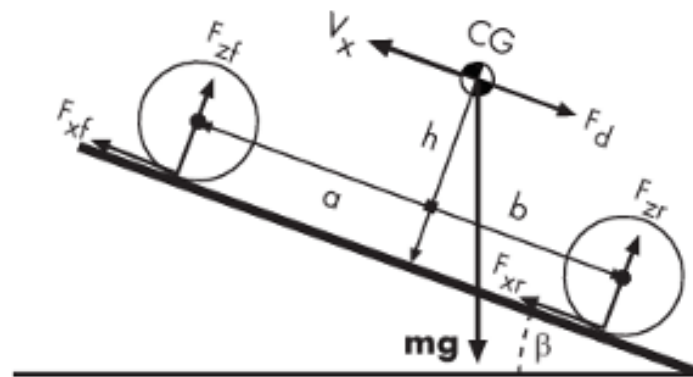
Kebutuhan penyeragaman kecepatan muncul karena kedua motor memiliki toleransi manufaktur yang berbeda pada resistansi jangkar, konstanta torsi, gesekan *gearbox*, dan *duty cycle dead zone*, sehingga *duty cycle* PWM yang identik tidak menjamin kecepatan putar yang sama. Selisih kecepatan kedua sisi menghasilkan kecepatan *yaw* yang tidak dikehendaki. Oleh sebab itu, setiap roda penggerak dilengkapi rotary *encoder* dan dikendalikan oleh satu untai PI tersendiri terhadap nilai acuan yang sama.

Arsitektur pengendali disusun dalam dua tingkat. STM32F411CEU6 bertindak sebagai pengendali tingkat bawah yang menangani pembacaan *encoder*, eksekusi PI, dan pembangkitan PWM. NVIDIA Jetson Orin Nano bertindak sebagai pengendali tingkat atas untuk mengolah data YDLIDAR, menampilkan telemetri, serta mendukung pemetaan dan

odometri. Arah hadap robot diperoleh dari koreksi hasil pencocokan pemindaian LiDAR terhadap peta, sedangkan umpan balik kendali kecepatan PI berasal dari *encoder*.

2.3.1 Dinamika traksi robot

Gambar 2.1 menunjukkan gaya pada model kendaraan dua gandar, meliputi gaya longitudinal roda, gaya normal, berat, dan hambatan aerodinamis [11]. Untuk perhitungan robot pada penelitian ini, kemiringan lintasan ditetapkan nol dan hambatan aerodinamis diabaikan pada kecepatan operasi rendah. Kebutuhan gaya traksi dihitung dari hambat gelinding dan gaya percepatan. Hambat gelinding dinyatakan sebagai hasil kali koefisien hambat gelinding dengan gaya normal, sehingga pada lantai rata diperoleh Persamaan (2.1).



Gambar 2. 1 Diagram gaya pada model kendaraan dua gandar

(sumber: mathworks.com [11])

$$F_{tot} = \mu_r \cdot m \cdot g + m \cdot a \quad (2.1)$$

Keterangan:

F_{tot}	: gaya traksi total (N)
μ_r	: koefisien hambat gelinding (0,01–0,03 untuk roda karet pada lantai keras)
m	: massa total robot (kg)
g	: percepatan gravitasi (9,81 m/s ²)
a	: percepatan robot (m/s ²)

Dengan asumsi pembagian beban merata, gaya traksi total digunakan untuk menentukan kebutuhan torsi pada setiap roda penggerak melalui Persamaan (2.2). Torsi ini berada pada poros keluaran *gearbox* yang terhubung langsung ke roda.

$$\tau_{roda} = \frac{F_{tot} \times r}{n} \quad (2.2)$$

Keterangan:

r : jari-jari roda (m)

n : jumlah roda penggerak

τ_{roda} : kebutuhan torsi tiap roda atau poros keluaran *gearbox* (N·m)

Hubungan antara kecepatan linier robot dan kecepatan putar roda diperoleh dari kondisi menggelinding tanpa *slip*.

$$n_{roda} = \frac{v \times 60}{\pi \times D} \quad (2.3)$$

Keterangan:

n_{roda} : kecepatan putar roda (RPM)

D : diameter roda (m)

v : kecepatan linier robot (m/s)

Persamaan (2.3) digunakan di Bab III untuk menentukan *setpoint* kecepatan roda dari kecepatan linier yang dikehendaki.

2.4 Motor DC dan *Rotary Encoder*

2.4.1 Motor DC PG45 dengan *encoder* 200 PPR

Motor DC PG45 adalah motor arus searah jenis *brushed* berdiameter 45 mm dengan *gearbox* planetari terintegrasi dan *rotary encoder* inkremental pada poros belakang motor. *Gearbox* planetari mereduksi kecepatan sekaligus melipatgandakan torsi dengan susunan roda gigi koaksial, sehingga dimensi motor tetap ringkas meskipun rasio reduksinya besar. Karakteristik ini sesuai bagi robot *mobile* yang memerlukan torsi besar pada kecepatan rendah untuk mengatasi inersia awal dan hambatan gelinding.



Gambar 2. 2 Contoh bentuk fisik motor DC dengan *gearbox planetary* PG45

(sumber: leisonmotor.en.made-in-china.com [12])

Rancangan ini menggunakan parameter *encoder* kuadratur dua kanal sebesar 200 PPR pada poros motor dan rasio *gearbox* 1:71,2. Keduanya merupakan parameter unit yang harus diverifikasi melalui identifikasi motor dan pengujian satu putaran keluaran. Katalog PG45 memuat varian yang berbeda [13], sehingga foto pada Gambar 2.2 tidak digunakan untuk menetapkan resolusi *encoder*. Parameter rancangan dirangkum pada Tabel 2.2.

Tabel 2. 2 Spesifikasi motor DC PG45 dengan *encoder* 200 PPR

Spesifikasi	Keterangan
Tipe motor	Motor DC <i>brushed</i> dengan <i>planetary gearbox</i>
Tegangan operasi	12–24 V DC
Arus kerja	2,5–5 A (bergantung pembebanan)
Daya nominal	60 W
Rasio reduksi <i>gearbox</i>	1 : 71,2
Kecepatan keluaran nominal	± 120 RPM
Torsi efektif / <i>stall</i>	± 70 / ± 100 kgf·cm
Tipe dan resolusi <i>encoder</i>	Inkremental kuadratur dua kanal, 200 PPR (pada poros motor)
Tegangan suplai <i>encoder</i>	3,3 V DC
Jumlah kabel	6 (Motor+, Motor–, VCC, GND, Kanal A, Kanal B)
Jumlah unit yang digunakan	2 buah (roda penggerak kiri dan kanan)

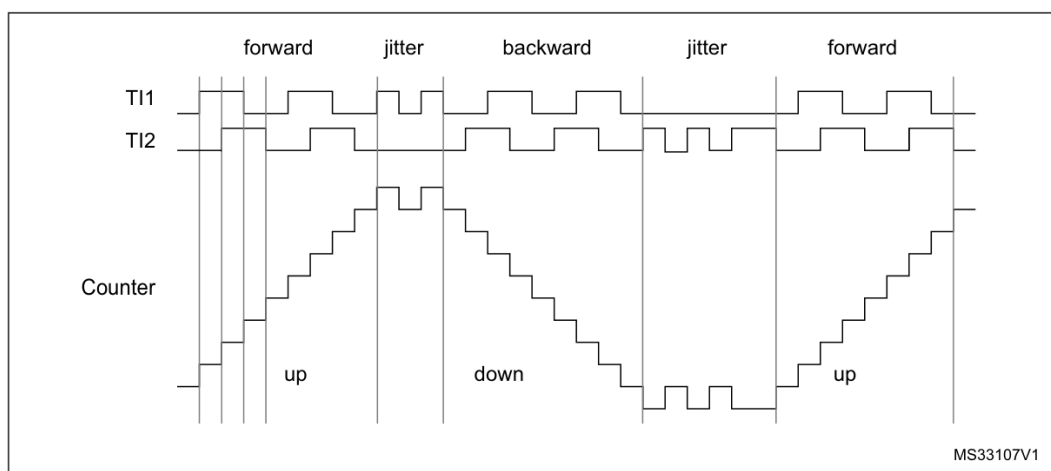
(Sumber: Parameter rancangan penulis; informasi umum keluarga motor PG45 dari Brontoseno [13]. Verifikasi spesifikasi unit sebelum pengujian.)

Varian motor PG45 tersedia dalam beberapa pilihan tegangan nominal. Apabila motor bertegangan nominal 24 V dioperasikan pada catu daya 12 V, kecepatan tanpa beban turun kurang lebih separuhnya sehingga rentang kecepatan operasi dan *headroom* sinyal kendali PI ikut berkurang. Verifikasi lembar data unit yang benar-benar digunakan merupakan langkah wajib sebelum perancangan catu daya.

Respons kecepatan motor terhadap perubahan *duty cycle* digunakan sebagai dasar penalaan PI. Pada penelitian ini karakteristik dinamik diperoleh melalui pengujian *step*, tanpa menurunkan model listrik dan mekanik motor secara terpisah.

2.4.2 Rotary encoder dan perhitungan kecepatan

Rotary encoder inkremental mengubah perpindahan sudut poros menjadi deretan pulsa digital. *Encoder* pada motor PG45 menghasilkan dua deret pulsa berbeda fasa 90°, yaitu kanal A dan kanal B, sehingga selain menghitung perpindahan sudut sistem juga dapat menentukan arah putaran. Urutan fasa kedua kanal menentukan arah pencacahan sesuai konfigurasi masukan *timer*. Gambar 2.3 memperlihatkan sinyal masukan TI1 dan TI2 serta perubahan nilai pencacah pada *encoder* interface mode STM32 [14].



Gambar 2. 3 Pencacahan sinyal kuadratur pada encoder interface mode STM32

(sumber: st.com [14])

Pencacahan sinyal kuadratur dapat dilakukan dengan dua cara. Cara pertama memakai *hardware timer* dalam *encoder mode*, yaitu mode ketika pencacah bertambah atau berkurang otomatis berdasarkan tepi sinyal kedua kanal tanpa intervensi CPU; mode X4 mencacah setiap tepi naik dan turun kedua kanal sehingga resolusinya empat kali jumlah pulsa per putaran. Cara kedua memakai interupsi eksternal pada satu kanal, sedangkan kanal lainnya dibaca sebagai penentu arah. Cara kedua hanya mencacah tepi kanal yang diinterupsi sehingga jumlah tepi

yang dihitung setengah dari *encoder mode* dan beban CPU lebih besar, tetapi tidak memerlukan pasangan pin *timer* tertentu.

$$CPR = 4 \times PPR_{\text{encoder}} \times N \quad (2.4)$$

Keterangan:

CPR : *Counts Per Revolution*, jumlah pencacahan per putaran poros keluaran

PPR_{encoder} : resolusi *encoder* pada poros motor (200 PPR)

N : rasio reduksi *gearbox* (71,2)

Kecepatan putar dihitung dengan metode pencacahan pulsa dalam selang waktu tetap, yaitu membaca selisih nilai pencacah pada setiap periode pencuplikan.

$$n_{\text{roda}} = \frac{\Delta C \times 60}{CPR \times T_s} \quad (2.5)$$

Keterangan:

n_{roda} : kecepatan putar poros keluaran (RPM)

ΔC : selisih pencacahan per periode (*count*)

T_s : periode pencuplikan (detik)

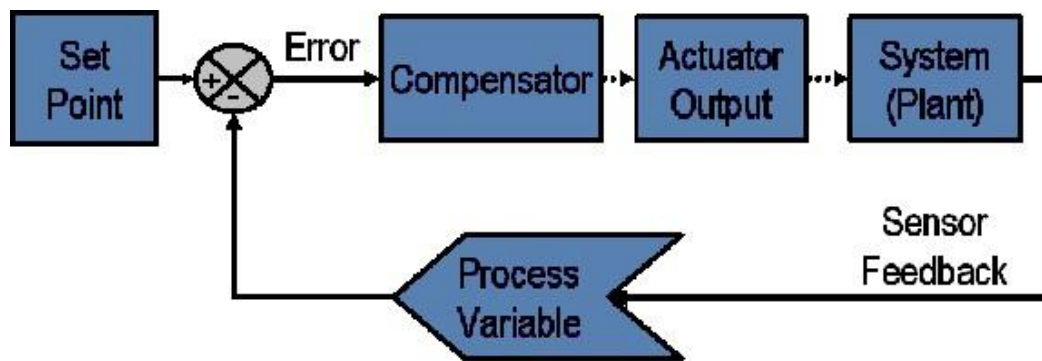
Persamaan (2.5) mengubah selisih pencacahan *encoder* menjadi kecepatan roda sebagai umpan balik PI. Periode pencuplikan dipilih tetap: periode yang terlalu singkat memperbesar fluktuasi karena satu *count* setara dengan kecepatan yang lebih besar, sedangkan periode yang terlalu panjang menunda respons kendali [15].

Metode pencacahan pulsa dalam selang waktu tetap bersifat rata-rata bergerak, sehingga kecepatan hasil perhitungan tertinggal kira-kira setengah periode pencuplikan terhadap kecepatan sesungguhnya. Bila pembacaan masih difilter oleh *low pass filter* digital berorde satu, misalnya $y[k] = (3 y[k-1] + n[k]) / 4$, tertinggalnya bertambah sebesar konstanta waktu filter tersebut. Selain itu periode eksekusi kendali tidak harus sama dengan periode perekaman data ke pengendali tingkat atas. Kedua periode tersebut perlu dibedakan karena waktu tunda yang terbaca pada kurva respons memuat tertinggalnya rantai pengukuran, bukan hanya waktu tunda proses yang dikendalikan.

2.5 Sistem Kendali PI

2.5.1 Sistem kendali *open loop* dan *closed loop*

Sistem kendali dibedakan menjadi *open loop* dan *closed loop*. Pada *open loop*, sinyal kendali diberikan tanpa mempertimbangkan keluaran aktual sehingga sistem tidak mampu mengoreksi diri ketika terjadi gangguan. Sebaliknya, *closed loop* mengukur keluaran aktual, membandingkannya dengan nilai acuan, dan menggunakan selisih keduanya sebagai dasar pembangkitan sinyal kendali [16]. Struktur umpan balik pada sistem *closed loop* ditunjukkan pada Gambar 2.4.



Gambar 2. 4 Diagram blok sistem kendali *closed loop*

(sumber: ni.com [17])

Pada sistem penggerak robot, kendali *open loop* tidak memadai karena hubungan antara *duty cycle* PWM dan kecepatan putar roda bersifat tidak linier dan berubah-ubah akibat penurunan tegangan baterai, variasi pembebanan, perbedaan gesekan *gearbox*, serta adanya *dead-zone* tegangan pada putaran rendah.

Dalam penerapannya pada robot ini, *setpoint* kecepatan dalam RPM dibandingkan dengan kecepatan aktual dari *encoder* untuk menghasilkan *error*. *Error* diolah oleh algoritma PI diskret pada STM32F411CEU6 menjadi *duty cycle* PWM. Sinyal tersebut diterjemahkan ke *register* CCR, dikuatkan oleh *driver* BTS7960, dan diberikan kepada motor PG45. Putaran motor diukur kembali oleh *encoder* dan diumpanbalikkan pada setiap periode pencuplikan.

2.5.2 Algoritma kontroler PI

Kontroler PI merupakan algoritma umpan balik yang menggabungkan aksi proporsional dan integral [16]. Aksi proporsional merespons *error* saat ini, sedangkan aksi integral mengakumulasi *error* untuk mengurangi *steady state error*. Penelitian ini tidak menggunakan aksi derivatif sehingga parameter yang ditentukan hanya K_p dan K_i .

Error adalah selisih antara *setpoint* kecepatan dan RPM aktual. K_p menyatakan penguatan proporsional, sedangkan K_i menyatakan penguatan integral. Keluaran PI berupa *duty cycle* PWM.

Karena kendali dijalankan oleh mikrokontroler, bentuk matematis yang digunakan dibatasi pada PI diskret di subbab 2.5.3. Karakteristik kedua komponen kendali dijelaskan sebagai berikut:

Aksi proporsional memberikan sinyal kendali sebanding dengan *error* saat ini. Peningkatan K_p mempercepat *rise time* dan memperkecil *error*, tetapi nilai yang terlalu tinggi dapat menimbulkan *overshoot* dan osilasi [16].

Aksi integral mengakumulasi *error* sepanjang waktu sehingga *steady state error* dapat dihilangkan. Nilai K_i yang terlalu besar dapat memperbesar *overshoot* dan memicu *integral windup* ketika sinyal kendali tersaturasi [18].

Pada penelitian ini aksi derivatif tidak digunakan. Pengendali hanya memiliki parameter K_p dan K_i sehingga struktur yang diterapkan adalah PI.

2.5.3 Implementasi PI diskret dan *anti-windup*

PI diskret menghitung aksi proporsional dari *error* saat ini dan aksi integral dari akumulasi *error* pada setiap periode pencuplikan. Bentuk yang digunakan pada mikrokontroler dinyatakan pada Persamaan (2.6) [15].

$$u[k] = uff + K_p \times e[k] + K_i \times T_s \times \Sigma e[j] \quad (2.6)$$

Keterangan:

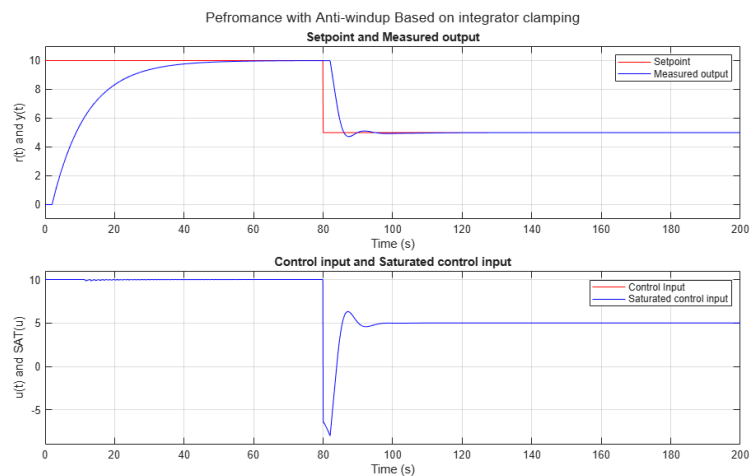
$u[k]$	: keluaran kendali pada cuplikan ke-k dalam <i>duty cycle</i> (%)
uff	: <i>duty cycle</i> awal atau <i>feedforward</i> , yaitu nilai tetap yang menggeser titik kerja keluaran agar tidak berada di daerah mati penggerak (%)
$e[k]$	: <i>setpoint</i> dikurangi RPM aktual pada cuplikan ke-k (RPM)
$\Sigma e[j]$	: akumulasi <i>error</i> sejak integral direset sampai cuplikan ke-k
T_s	: periode pencuplikan (s)
K_p	: penguatan proporsional (%/RPM)
K_i	: penguatan integral (%/(RPM·s))

Nilai integral diperbarui pada setiap periode pencuplikan. Karena kontroler tidak memiliki suku derivatif, program tidak memerlukan perhitungan beda *error* maupun filter derivatif sehingga sensitivitas terhadap derau pembacaan *encoder* berkurang. Suku uff tidak

mengubah struktur pengendali karena nilainya tetap; suku tersebut hanya membuat aksi proporsional dan integral tidak perlu lebih dahulu mengisi daerah mati penggerak sebelum motor bergerak.

Sinyal kendali dibatasi antara 0% dan batas *duty cycle* maksimum yang ditetapkan, dan batas tersebut tidak harus 100% karena dapat diturunkan demi keamanan penggerak. Ketika keluaran tersaturasi, suku integral dapat terus bertambah dan menyebabkan integral *windup*. Penelitian ini menggunakan *clamping*, yaitu membatasi akumulator integral pada rentang tertentu sekaligus membatasi keluaran pada batas *duty* yang berlaku, sehingga akumulasi tidak tumbuh tanpa batas ketika keluaran mencapai saturasi [18].

Contoh simulasi penerapan integrator *clamping* ditunjukkan pada Gambar 2.5 [19]. Panel atas menunjukkan keluaran sistem, sedangkan panel bawah menunjukkan sinyal kendali dan batas saturasinya.



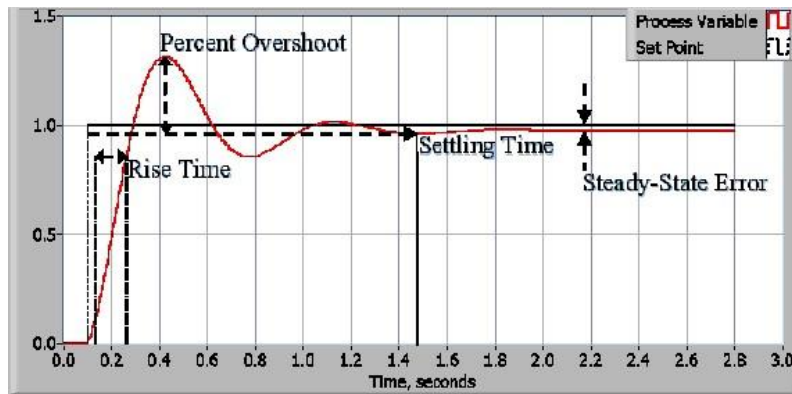
Gambar 2. 5 Contoh respons sistem dengan *anti-windup* integrator *clamping*

(sumber: mathworks.com [19])

2.5.4 Karakteristik respons sistem

Kinerja PI dinilai melalui *rise time*, *maximum overshoot*, *settling time*, dan *steady state error* [20].

Gambar 2.6 memperlihatkan penandaan karakteristik tersebut pada contoh respons sistem kendali [17]. Definisi operasional dan pita toleransi yang digunakan dalam penelitian ini dijelaskan berikut.



Gambar 2. 6 Karakteristik respons tangga sistem kendali

(sumber: ni.com [17])

Rise time (t_r) — selang waktu respons naik dari 10% ke 90% perubahan kecepatan menuju nilai stabil. Nilainya dibaca dari data rekaman respons atau grafiknya.

Maximum overshoot (M_p) — lonjakan maksimum di atas *setpoint*, dinyatakan dalam persen menggunakan Persamaan (2.7).

Settling time (t_s) — waktu sejak perubahan *setpoint* sampai respons masuk dan tetap berada di dalam pita toleransi $\pm 5\%$ terhadap *setpoint* selama sisa rekaman [21]. Pita 5% dipilih, bukan 2%, karena pada *setpoint* kecepatan rendah kuantisasi satu *count encoder* dalam satu periode kendali sudah setara dengan riak beberapa persen, sehingga pita yang lebih sempit hanya akan mengukur derau pembacaan. Pada *setpoint* yang sangat kecil, lebar pita dibatasi agar tidak lebih sempit daripada nilai mutlak tertentu. Bila respons belum masuk pita sampai akhir rekaman, kondisi tersebut dilaporkan.

steady state error — selisih *setpoint* dan kecepatan stabil yss, dinyatakan sebagai persentase menggunakan Persamaan (2.8). Nilai yss dihitung sebagai rata-rata sejumlah cuplikan terakhir rekaman agar riak pembacaan tidak ikut terhitung sebagai *error*. Penilaian dilakukan pada *setpoint* positif dan tidak nol.

$$M_p = \frac{y_{maks} - r}{r} \times 100\% \quad (2.7)$$

$$\%ess = \frac{|r - y_{ss}|}{r} \times 100\% \quad (2.8)$$

Keterangan:

y_{maks} : kecepatan puncak setelah *step* (RPM)

y_{ss} : kecepatan stabil, yaitu rata-rata cuplikan terakhir rekaman (RPM)

r : *setpoint* (RPM)

Persamaan *overshoot* digunakan untuk pengujian naik dari keadaan diam menuju *setpoint* positif; respons yang tidak melampaui *setpoint* dilaporkan memiliki *overshoot* 0%.

2.6 Penentuan Parameter Awal PI dengan Ziegler–Nichols Metode Pertama

Ziegler–Nichols metode pertama atau metode kurva reaksi digunakan untuk menghitung nilai awal K_p dan K_i . Setiap motor diuji secara manual dalam kondisi *open loop* menggunakan masukan *step duty cycle*. Penguatan proses K , waktu tunda L , dan konstanta waktu T diperoleh dari respons kecepatan, kemudian digunakan pada aturan PI [4]. Pemilihan data, perhitungan parameter, dan pengisian *gain* dilakukan oleh peneliti, bukan oleh proses penalaan otomatis.

2.6.1 Prinsip Ziegler–Nichols metode pertama

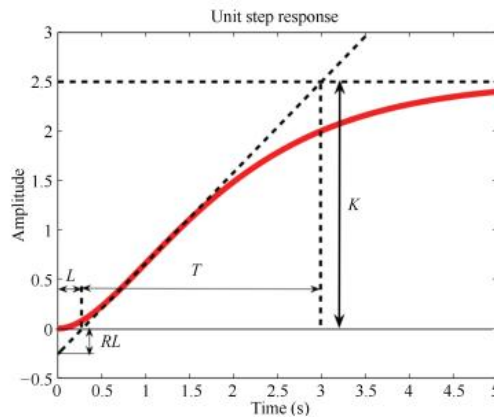
Ziegler–Nichols metode pertama menggunakan kurva reaksi akibat perubahan *duty cycle* dari u_0 ke u_1 untuk memperoleh penguatan proses K , waktu tunda L , dan konstanta waktu T . Nilai u_0 dan u_1 dipilih pada rentang kerja motor yang aman dan dicatat sebagai kondisi pengujian; keduanya tidak dicari secara otomatis oleh program. Contoh pembacaan parameter pada kurva reaksi ditunjukkan pada Gambar 2.7.

Aturan penentuan nilai awal K_p , T_i , dan K_i dituliskan langsung pada Persamaan (2.10)–(2.12) di bagian 2.6.3.

Penguatan proses K dihitung dari perubahan kecepatan stabil terhadap perubahan *duty cycle*. Nilai L dan T dibaca dari garis singgung kurva reaksi. Selanjutnya K_p , T_i , dan K_i dihitung menggunakan aturan Ziegler–Nichols metode pertama.

2.6.2 Pengujian *step* manual dan identifikasi parameter K , L , dan T

Motor dijalankan pada *duty cycle* u_0 hingga kecepatan mencapai nilai stabil y_0 . *Duty cycle* kemudian diubah secara *step* ke u_1 dan dipertahankan sampai kecepatan mencapai nilai stabil y_1 . Waktu, *duty cycle*, dan RPM direkam melalui telemetri. Kurva hasil pengujian digunakan untuk memperoleh K , L , dan T secara manual. Gambar 2.7 menggunakan masukan unit *step*, sehingga perubahan keluaran stabilnya sama dengan penguatan K . Untuk pengujian PWM dengan besar *step* yang berbeda, K tetap dihitung sebagai perbandingan perubahan keluaran terhadap perubahan masukan menggunakan Persamaan (2.9) [22].



Gambar 2. 7 Kurva reaksi unit *step* dan penentuan parameter L, T, serta K

(sumber: ieee-jas.net [22])

Penguatan proses merupakan perbandingan perubahan kecepatan stabil terhadap perubahan *duty cycle*. Nilainya dihitung menggunakan Persamaan (2.9).

$$K = \frac{y_1 - y_0}{u_1 - u_0} \quad (2.9)$$

Keterangan:

y_0 dan y_1 : kecepatan stabil sebelum dan sesudah *step* (RPM)

u_0 dan u_1 : *duty cycle* sebelum dan sesudah *step* (%)

K : penguatan proses (RPM/%)

L : waktu tunda hasil pendekatan kurva reaksi (s)

T : konstanta waktu pendekatan kurva reaksi (s)

Pada kurva respons, garis singgung ditarik pada bagian dengan kemiringan terbesar. Perpotongannya dengan garis kecepatan awal dan akhir ditandai sebagai t_a dan t_b . Nilai L dibaca dari saat *step* t_0 hingga t_a , sedangkan T merupakan selang antara t_a dan t_b [22]. Karena kurva dibaca dari data rekaman yang berperiode tertentu, L yang diperoleh merupakan waktu tunda ekuivalen yang juga memuat tertinggalnya rantai pengukuran dan tidak dapat lebih kecil daripada resolusi perekaman. Apabila L yang terbaca lebih kecil daripada dua kali periode perekaman, nilai tersebut dilaporkan sebagai batas atas, dan K_p serta K_i yang dihitung darinya diperlakukan sebagai nilai awal yang wajib divalidasi melalui pengujian *closed loop* serta penalaan halus, bukan sebagai parameter akhir.

Tabel 2. 3 Data yang dicatat pada identifikasi respons step

No.	Parameter	Cara memperoleh	Satuan
1	y0	RPM stabil pada u0	RPM
2	y1	RPM stabil pada u1	RPM
3	K	Persamaan (2.9)	RPM/%
4	L	Dari saat <i>step</i> to sampai t _a	s
5	T	Selang antara t _a dan t _β	s

Pengambilan K, L, dan T dilakukan terpisah untuk motor kiri dan kanan. Kondisi beban, tegangan baterai, periode eksekusi kendali, periode perekaman data, serta cara pembacaan kurva dicatat agar hasil pengulangan dapat dibandingkan.

2.6.3 Perhitungan nilai awal Kp dan Ki

Kendali PI dinonaktifkan, kemudian motor kiri serta kanan diuji secara terpisah dalam kondisi *open loop*.

Duty cycle awal ketika motor mulai berputar ditandai sebagai udead. Nilai u0 dan u1 dipilih di atas udead dan di dalam batas *duty cycle* yang aman bagi penggerak, dengan selisih yang cukup besar agar perubahan kecepatan stabil terukur jelas namun tetap berada pada daerah kerja yang sama dengan *setpoint* pengujian. Penetapan angkanya diuraikan pada perancangan di Bab III.

Motor dijalankan pada u0 hingga mencapai y0, kemudian diberi masukan *step* ke u1 hingga mencapai y1.

Kurva respons digunakan untuk menghitung K serta membaca L dan T secara manual sesuai subbab 2.6.2.

Setelah K, L, dan T diperoleh, nilai awal Kp, Ti, dan Ki dihitung menggunakan Persamaan (2.10), (2.11), dan (2.12). Faktor K tetap digunakan agar skala penguatan proses diperhitungkan [22].

$$K_p = \frac{0,9 \times T}{K \times L} \quad (2.10)$$

$$T_i = 3,33 \times L \quad (2.11)$$

$$K_i = \frac{K_p}{T_i} \quad (2.12)$$

Ti adalah waktu integral dalam sekon. Dengan masukan *duty cycle* dalam persen dan keluaran RPM, satuan Kp adalah %/RPM dan Ki adalah %/(RPM·s). Aksi derivatif tidak digunakan sehingga Kd ditetapkan 0. Hasil perhitungan merupakan nilai awal yang harus divalidasi pada sistem *closed loop* dan dapat disesuaikan melalui penalaan halus. Cara memasukkan parameter ke pengendali diuraikan pada Bab III.

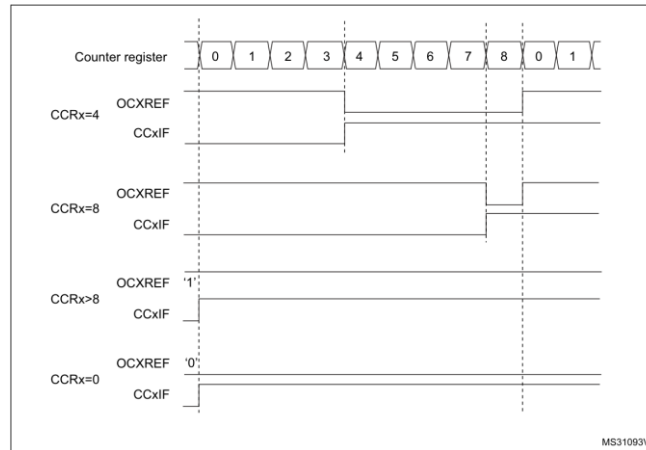
Prosedur tersebut dilakukan pada motor kiri dan kanan secara terpisah agar nilai awal Kp dan Ki mempertimbangkan perbedaan karakteristik dinamik kedua motor.

2.7 Pulse Width Modulation (PWM)

PWM merupakan teknik pengendalian tegangan rata-rata pada beban listrik dengan mengubah lebar pulsa sinyal digital berperiode tetap [23]. Sinyal PWM menyalakan dan mematikan tegangan bergantian pada frekuensi tinggi sehingga beban menerima tegangan rata-rata sebanding dengan perbandingan durasi keadaan menyala terhadap total periode, yang disebut *duty cycle*. Metode ini lebih efisien dibandingkan pengaturan resistif karena elemen pensaklaran hanya berada pada dua kondisi ekstrem sehingga disipasi dayanya minimal.

Duty cycle menyatakan persentase lamanya pulsa aktif dalam satu periode PWM. Pada penelitian ini besaran tersebut diatur melalui *register timer* STM32.

Pada STM32, sinyal PWM dibangkitkan modul *timer* dengan membandingkan nilai pencacah terhadap dua *register*. *Register* ARR (Auto-Reload Register) menentukan nilai maksimum pencacah sebelum diatur ulang, sehingga menentukan periode dan frekuensi PWM. *Register* CCR (Capture Compare Register) menentukan ambang perubahan logika keluaran, sehingga menentukan *duty cycle*. Mekanismenya ditunjukkan pada Gambar 2.8. Contoh pada manual menggunakan ARR = 8 dan beberapa nilai CCRx untuk memperlihatkan perubahan lebar pulsa OCxREF [14].



Gambar 2. 8 Pembangkitan PWM *edge-aligned* pada *timer* STM32 (ARR = 8)

(sumber: st.com [14])

$$ARR = \frac{f_{TIM}}{(PSC + 1) \times f_{PWM}} - 1 \quad (2.13)$$

$$CCR = \frac{d_{PWM}}{100} \times (ARR + 1) \quad (2.14)$$

Keterangan:

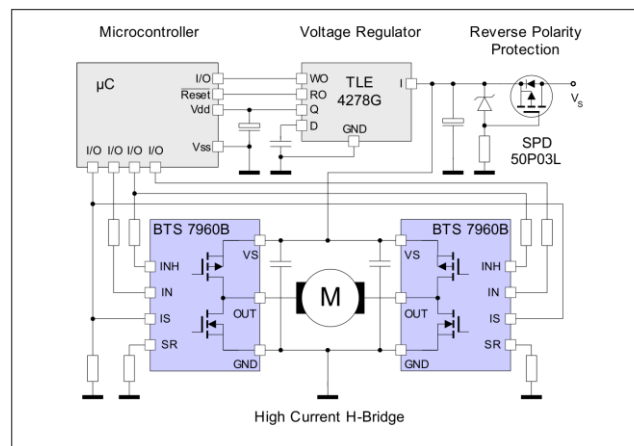
- f_{TIM} : frekuensi *clock* masukan *timer* (maksimum 100 MHz pada STM32F411CEU6)
- f_{PWM} : frekuensi PWM (Hz)
- PSC : *prescaler*
- ARR : nilai muat ulang *timer*
- CCR : nilai pembanding
- dPWM : *duty cycle* (%)

Frekuensi PWM dipilih sesuai batas kemampuan *driver* dan ketersediaan pencacah *timer*. Pada penelitian ini keluaran PI berupa *duty cycle*, lalu dikonversi ke CCR menggunakan Persamaan (2.14). Nilai CCR dibatasi sesuai rentang PWM yang diizinkan.

2.8 Driver Motor BTS7960

BTS7960 merupakan modul *driver* motor jembatan-H berdaya tinggi yang menggunakan sepasang IC BTS7960 produksi Infineon Technologies. Setiap IC merupakan half-bridge *driver* yang mengintegrasikan MOSFET sisi tinggi dan sisi rendah beserta rangkaian penggerakannya dalam satu kemasan NovalithIC. Konfigurasi dua IC membentuk

jembatan-H penuh yang memungkinkan pengendalian motor DC dua arah [24]. Contoh rangkaian aplikasi dua IC dari produsen ditunjukkan pada Gambar 2.9.



Gambar 2. 9 Rangkaian aplikasi jembatan-H menggunakan dua IC BTS7960

(sumber: infineon.com [24])

Pengendalian dilakukan melalui pin RPWM, LPWM, REN, dan LEN. Ketika sinyal PWM diberikan pada RPWM sementara LPWM bernilai rendah, motor berputar maju dengan kecepatan sebanding *duty cycle*; sebaliknya sinyal pada LPWM membalik arah putaran. Pin REN dan LEN harus bernilai tinggi agar modul mengalirkan arus ke motor [24]. Spesifikasi modul ditunjukkan pada Tabel 2.4.

Tabel 2. 4 Spesifikasi modul driver motor BTS7960

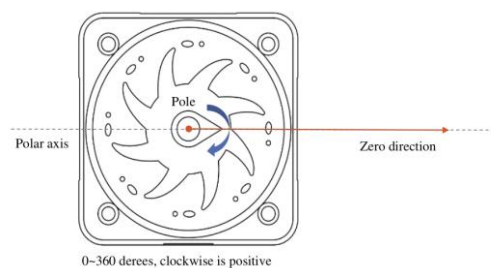
Spesifikasi	Keterangan
IC utama	Infineon BTS7960 ($2 \times \text{half-bridge}$ = jembatan-H penuh)
Tegangan operasi motor	5,5 V – 27 V DC
Pembatasan arus IC	43 A tipikal; bukan rating arus kontinu modul
Frekuensi PWM maksimum	25 kHz
Tegangan logika masukan	3,3 V / 5 V (kompatibel TTL dan CMOS)
Resistansi konduksi ($R_{DS(on)}$)	16 m Ω (tipikal)
Masukan kendali	RPWM, LPWM, REN, LEN
Proteksi	Suhu berlebih, arus berlebih, tegangan berlebih, hubung singkat
Dimensi modul	50 mm $\times$ 50 mm

(Sumber: Karakteristik IC dari Infineon BTS7960 [24]; pin dan dimensi modul harus dicocokkan dengan papan yang digunakan.)

BTS7960 memiliki resistansi konduksi rendah dan proteksi internal. Angka 43 A pada lembar data menunjukkan tingkat pembatasan arus tipikal, bukan jaminan kemampuan arus kontinu modul [24]. Kemampuan modul juga dipengaruhi pendinginan, jalur PCB, dan konektor. Pada penelitian ini digunakan dua modul untuk mengendalikan kedua motor secara independen. Jalur daya motor dan logika menggunakan titik acuan ground bersama dengan penghantar yang disusun agar gangguan arus motor tidak mengganggu mikrokontroler.

2.9 YDLIDAR T-mini Plus 25

YDLIDAR T-mini Plus merupakan sensor LiDAR dua dimensi berputar 360° yang dikembangkan EAI Technology. Sensor ini bekerja berdasarkan prinsip *Time of Flight* (ToF), yaitu pengukuran jarak berdasarkan waktu tempuh pulsa laser inframerah dari pemancar menuju objek dan kembali ke penerima. Sementara pengukuran berlangsung, struktur mekanisnya berputar 360° untuk memperoleh informasi sudut secara berkelanjutan sehingga dihasilkan data awan titik lingkungan sekitar [25]. Sistem koordinat polar sensor ditunjukkan pada Gambar 2.10.



Gambar 2. 10 Sistem koordinat polar YDLIDAR T-mini Plus

(sumber: akizukidenshi.com [25])

[25]Gambar 2.10 memperlihatkan arah nol dan konvensi sudut positif searah jarum jam pada sensor [25]. Setiap hasil pemindaian terdiri atas pasangan jarak dan sudut. Pada penelitian ini data tersebut tidak hanya ditampilkan sebagai awan titik, tetapi juga diubah ke koordinat Kartesian agar dapat dimasukkan ke occupancy grid dan dibandingkan kembali dengan peta pada proses scan matching.

Tabel 2. 5 Spesifikasi sensor YDLIDAR T-mini Plus

Spesifikasi	Nilai	Satuan
Prinsip pengukuran	<i>Time of Flight (ToF)</i>	—
Frekuensi pengukuran jarak	4.000	Hz
Frekuensi pemindaian	6 – 12 (bawaan 6)	Hz
Jangkauan pengukuran	0,05 – 12 (reflektivitas 80%); varian 25 m hingga 25	m
Medan pandang	0 – 360	derajat
<i>Error</i> sistematis	± 20	mm
Resolusi sudut	0,54	derajat
Tegangan suplai / arus kerja	4,8 – 5,2 / 340 (tipikal)	V / mA
Antarmuka komunikasi	UART 3,3 V, 230.400 bps	—
Panjang gelombang laser	905 (tipikal), Class I	nm
Ketahanan cahaya sekitar	60.000	Lux
Dimensi / bobot	38,6 × 38,6 × 33,9 / 45	mm / g

(Sumber: Lembar data YDLIDAR T-mini Plus, EAI Technology [25])

Pada penelitian ini YDLIDAR T-mini Plus 25 berperan sebagai sensor pendukung pada pengendali tingkat atas. Data awan titik digunakan untuk membentuk peta lorong pada tahap *teach*. Pada tahap *repeat*, peta tersebut menjadi acuan *local scan matching* untuk membandingkan pemindaian LiDAR terkini dengan lingkungan yang telah direkam. Hasil pencocokan dapat digunakan untuk memperbaiki perkiraan posisi dan arah hadap robot, sedangkan umpan balik kendali kecepatan PI tetap berasal dari *rotary encoder*. Pendekatan navigasi *teach-and-repeat* berbasis pemindai laser juga telah dilaporkan oleh McManus et al. [31], yang menggunakan LiDAR sebagai sensor utama untuk mengulang rute yang telah diajarkan dan meningkatkan ketahanan terhadap perubahan pencahayaan. Tinjauan Krawciw dan Barfoot [32] menempatkan peta lokal dan estimasi relatif sebagai komponen penting pada navigasi *teach-and-repeat* agar robot dapat mengikuti kembali rute yang telah direkam tanpa bergantung pada peta global yang sepenuhnya konsisten.

Transformasi koordinat digunakan sebelum proses *local scan matching*. LiDAR memberikan data dalam bentuk koordinat polar berupa jarak r dan sudut θ . Agar titik hasil pemindaian dapat digambar pada peta, data polar terlebih dahulu diubah menjadi koordinat

Kartesian pada kerangka sensor. Hubungan keduanya dinyatakan pada Persamaan (2.15) dan Persamaan (2.16).

$$x_L = r \cos(\theta) \quad (2.15)$$

$$y_L = r \sin(\theta) \quad (2.16)$$

Posisi LiDAR pada robot dapat mempunyai *offset* terhadap pusat robot. Jika posisi robot dinyatakan sebagai (x_r, y_r, ψ_r) , sedangkan posisi pemasangan LiDAR terhadap robot adalah (x_m, y_m, ψ_m) , maka *pose* sensor pada kerangka dunia dihitung menggunakan Persamaan (2.17) sampai Persamaan (2.19).

$$x_s = x_r + x_m \cos(\psi_r) - y_m \sin(\psi_r) \quad (2.17)$$

$$y_s = y_r + x_m \sin(\psi_r) + y_m \cos(\psi_r) \quad (2.18)$$

$$\psi_s = \psi_r + \psi_m \quad (2.19)$$

Setelah *pose* sensor diketahui, setiap titik LiDAR diproyeksikan ke kerangka peta. Transformasi yang digunakan ditunjukkan pada Persamaan (2.20) dan Persamaan (2.21).

$$x_w = x_s + r \cos(\theta + \psi_s) \quad (2.20)$$

$$y_w = y_s + r \sin(\theta + \psi_s) \quad (2.21)$$

Pada *scan matching*, *pose* hasil odometri dipakai sebagai tebakan awal. Beberapa kandidat *pose* di sekitar tebakan tersebut diuji dengan memproyeksikan kembali titik LiDAR ke *occupancy grid*. Nilai kecocokan satu kandidat dinyatakan sebagai rata-rata skor sel peta yang dilalui titik *scan*. Secara umum fungsi yang dicari dapat ditulis seperti Persamaan (2.22), yaitu memilih *pose* yang memberikan skor pencocokan terbesar.

$$p^* = \arg \max_p S(p) \quad (2.22)$$

Dalam implementasi perangkat lunak, sel peta yang sudah terisi objek memberikan skor positif, sel bebas memberikan penalti kecil, sedangkan kandidat yang berada di luar peta tidak dipakai. Setelah kandidat terbaik diperoleh, selisih posisi dan sudut terhadap *pose* odometri digunakan sebagai koreksi. Koreksi tidak langsung diterapkan seluruhnya, tetapi dikalikan *gain* posisi dan *gain yaw* agar perubahan *pose* tidak melonjak akibat satu hasil pencocokan yang kurang baik.

Dibandingkan sensor ultrasonik, LiDAR unggul karena resolusi sudut $0,54^\circ$ memungkinkan pembedaan objek yang berdekatan, *error* sistematis ± 20 mm memberikan ketelitian pemosisian yang memadai, dan pemindaian 360° membuat satu sensor tunggal

mencukupi tanpa pemasangan di beberapa sisi robot [25]. Cara sensor ini digunakan di dalam sistem, termasuk pembentukan peta dan penentuan titik berhenti, diuraikan pada Bab III.

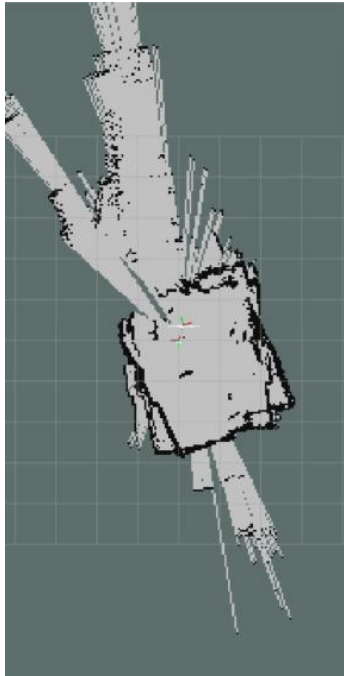
2.9.1 *Simultaneous Localization and Mapping (SLAM)*

Simultaneous Localization and Mapping (SLAM) merupakan metode pada robot bergerak yang menjalankan dua proses secara bersamaan, yaitu membangun representasi lingkungan dan memperkirakan pose robot pada lingkungan yang belum memiliki peta awal. Melalui proses tersebut, robot dapat memanfaatkan informasi sensor untuk membentuk peta sekaligus memperbarui perkiraan posisinya terhadap peta yang sedang dibangun [31].

Pada SLAM berbasis LiDAR, data pemindaian yang diperoleh pada waktu berbeda dibandingkan untuk memperkirakan perubahan posisi dan orientasi robot. Salah satu contoh pendekatan adalah F-LOAM yang memanfaatkan scan-to-scan dan scan-to-map matching untuk memperoleh odometri LiDAR dan memperbarui peta dengan tetap memperhatikan efisiensi komputasi. Pendekatan tersebut menunjukkan bahwa pencocokan data LiDAR dapat digunakan untuk mendukung estimasi gerak dan pemetaan secara real-time [31].

Secara umum, proses SLAM dapat dibagi menjadi front-end dan back-end. Front-end mengolah data sensor dan melakukan pencocokan pemindaian untuk memperoleh estimasi perubahan pose robot serta membentuk representasi lokal lingkungan. Selanjutnya, back-end memanfaatkan informasi pose dan kendala gerak untuk melakukan optimasi sehingga kesalahan yang terakumulasi dapat dikurangi dan peta menjadi lebih konsisten. Proses loop detection digunakan untuk mengenali kembali area yang pernah dilalui dan memberikan kendala tambahan pada proses optimasi [32], [33].

Pengembangan SLAM seperti ASL-SLAM memanfaatkan informasi semantik pada mekanisme loop closure untuk meningkatkan konsistensi lintasan dan peta. Pendekatan tersebut menunjukkan bahwa optimasi graf dan pengenalan kembali area yang telah dilalui dapat membantu mengurangi akumulasi error serta memperbaiki kestabilan hasil pemetaan. Meskipun demikian, penerapan SLAM tetap menghadapi beberapa tantangan, antara lain drift, kebutuhan komputasi yang relatif tinggi, dan perubahan lingkungan yang dapat memengaruhi kestabilan estimasi [32].



Gambar 2. 11 Hasil *mapping* SLAM

(Sumber: Dokumentasi pribadi.)

Gambar tersebut menunjukkan alur umum SLAM berbasis LiDAR 2D. Data pemindaian LiDAR terlebih dahulu diproses pada front-end untuk melakukan pencocokan scan, memperkirakan pose, dan membentuk sub-peta lokal. Informasi tersebut kemudian diteruskan ke back-end untuk proses filtering atau optimasi graf. Ketika sistem mengenali kembali lokasi yang pernah dilalui, loop detection menghasilkan kendala yang dapat digunakan untuk mengurangi drift dan memperbaiki konsistensi peta global. Kerangka ini digunakan sebagai pembanding teoritis terhadap metode Teach-and-Repeat pada CHIMI-Bot dan bukan sebagai arsitektur yang diimplementasikan secara penuh pada penelitian ini [33].

2.9.2 Kalman Filter

Kalman filter merupakan estimator rekursif yang menggabungkan prediksi keadaan dengan pengukuran baru berdasarkan tingkat ketidakpastian masing-masing informasi. Metode ini memperbarui estimasi dan kovarians galat setiap kali pengukuran baru diterima sehingga derau pengukuran dapat diredam tanpa harus menyimpan seluruh riwayat data [34].

Pada penelitian ini, konsep Kalman filter digunakan dalam bentuk skalar untuk memperhalus koreksi yaw hasil local LiDAR scan matching. Besaran yang menjadi pengukuran bukan yaw yang dibaca langsung oleh LiDAR, melainkan raw yaw correction yang diperoleh dari selisih orientasi antara pose hasil scan matching dan pose prediksi odometri. Pengukuran tersebut dinyatakan pada Persamaan (2.23).

$$z_k = \Delta\psi_{raw,k} \quad (2.23)$$

Dengan model perubahan koreksi yaw yang dianggap lambat antar-sampel, tahap prediksi nilai koreksi dan kovarians dapat dinyatakan melalui Persamaan (2.24) dan Persamaan (2.25).

$$\Delta\hat{\psi}_k^- = \Delta\hat{\psi}_{k-1} \quad (2.24)$$

$$P_k^- = P_{k-1} + Q \quad (2.25)$$

Selisih antara pengukuran raw dan prediksi disebut innovation. Karena besaran yang difilter berupa sudut, selisih sudut dinormalisasi menggunakan fungsi wrap agar tetap berada pada rentang sudut yang konsisten. Innovation ditunjukkan pada Persamaan (2.26).

$$v_k = \text{wrap}(z_k - \Delta\hat{\psi}_k^-) \quad (2.26)$$

Besarnya kontribusi pengukuran terhadap estimasi ditentukan oleh Kalman gain. Untuk kasus skalar, Kalman gain dapat dinyatakan dengan Persamaan (2.27).

$$K_k = \frac{P_k^-}{P_k^- + R} \quad (2.27)$$

Nilai koreksi yaw kemudian diperbarui menggunakan Kalman gain dan innovation, sedangkan kovarians estimasi diperbarui untuk siklus berikutnya. Hubungan tersebut ditunjukkan pada Persamaan (2.28) dan Persamaan (2.29).

$$\Delta\hat{\psi}_k = \Delta\hat{\psi}_k^- + K_k v_k \quad (2.28)$$

$$P_k = (1 - K_k)P_k^- \quad (2.29)$$

Koreksi yaw yang telah difilter dapat digunakan bersama yaw odometri encoder untuk membentuk yaw hasil koreksi atau fused yaw. Secara konseptual hubungan tersebut ditunjukkan pada Persamaan (2.30).

$$\psi_{fused,k} = \psi_{odo,k} + \Delta\hat{\psi}_k \quad (2.30)$$

Keterangan:

z_k : raw yaw correction hasil local LiDAR scan matching.

$\Delta\hat{\psi}_k^-$: prediksi koreksi yaw sebelum measurement update.

$\Delta\hat{\psi}_k$: koreksi yaw hasil Kalman filter.

P_k : kovarians galat estimasi.

Q : kovarians gangguan proses.

R : kovarians gangguan pengukuran.

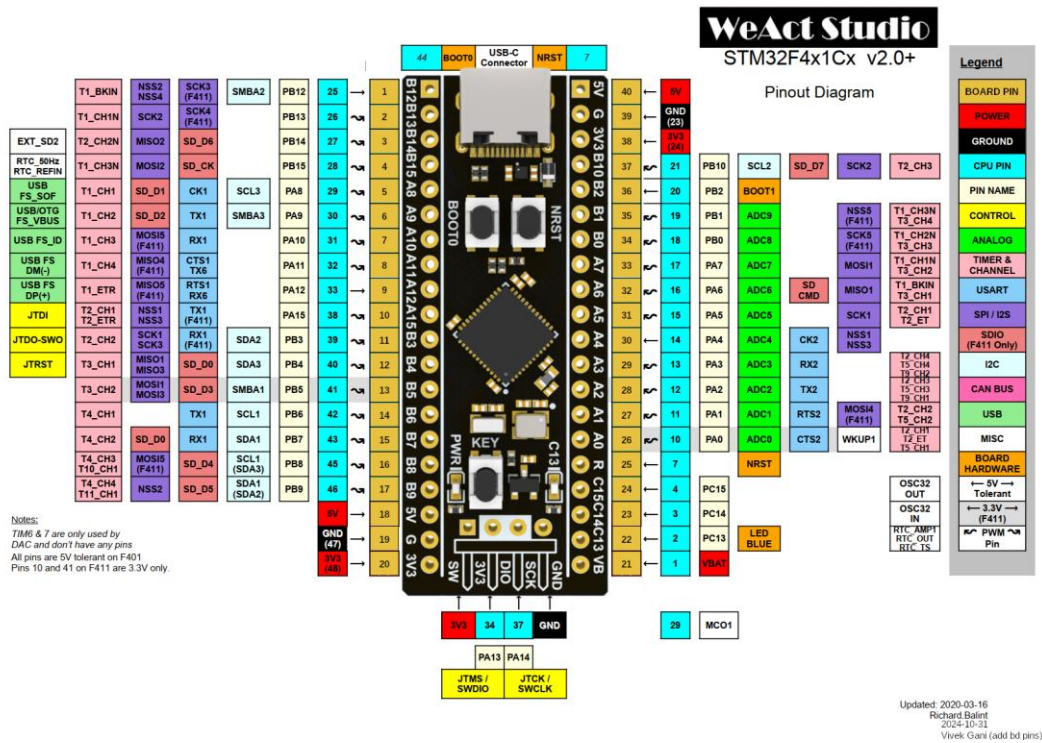
- v_k : innovation atau selisih antara pengukuran dan prediksi.
- K_k : Kalman gain; simbol ini berbeda dengan K penguatan proses pada pembahasan Ziegler–Nichols.
- $\psi_{odo,k}$: yaw yang diperoleh dari odometri encoder.
- $\psi_{fused,k}$: yaw hasil penggabungan odometri dengan koreksi yaw yang telah difilter.

Nilai Q dan R menentukan seberapa besar filter mempercayai model prediksi dan pengukuran. Nilai Q yang lebih besar membuat filter lebih cepat mengikuti perubahan, sedangkan nilai R yang lebih besar membuat filter lebih berhati-hati terhadap pengukuran baru. Pada penelitian ini nilai numerik Q dan R harus mengikuti parameter yang benar-benar digunakan pada perangkat lunak dan tidak ditetapkan hanya dari dasar teori.

Pada penelitian ini, teori SLAM digunakan sebagai dasar pembandingan untuk menjelaskan perbedaan antara full SLAM dan metode pemetaan yang diimplementasikan pada CHIMI-Bot. Pada tahap Teach, occupancy grid dibentuk dengan menempatkan data pemindaian LiDAR berdasarkan pose odometri encoder, sedangkan local LiDAR scan matching digunakan pada tahap Repeat untuk memberikan koreksi terhadap perkiraan pose berdasarkan peta hasil Teach. Dengan demikian, sistem tidak melakukan optimasi pose graph maupun loop closure global seperti pada full SLAM.

2.10 Mikrokontroler STM32F411CEU6

STM32F411CEU6 atau papan Black Pill merupakan mikrokontroler berbasis ARM Cortex-M4 32 bit yang beroperasi hingga 100 MHz dan memiliki FPU. Dukungan FPU membantu pelaksanaan perhitungan PI yang menggunakan bilangan pecahan pada setiap periode pencuplikan [26].



Gambar 2. 12 Konfigurasi pin papan pengembangan STM32F411CEU6 (*Black Pill*)

(sumber: github.com [27])

Mikrokontroler berfungsi sebagai pengendali tingkat bawah yang menangani pembacaan kedua *encoder*, eksekusi PI, pembangkitan PWM untuk kedua *driver* BTS7960, serta komunikasi serial dengan pengendali tingkat atas. *Loop* kendali dijalankan pada periode tetap agar perhitungan RPM dan aksi PI berlangsung teratur, sedangkan pengiriman telemetri dijadwalkan pada periode yang lebih panjang.

Tabel 2. 6 Spesifikasi mikrokontroler STM32F411CEU6

Spesifikasi	Keterangan
Inti prosesor	ARM Cortex-M4 32 bit dengan FPU
Frekuensi <i>clock</i> maksimum	100 MHz
Memori <i>Flash</i> / SRAM	512 KB / 128 KB
Tegangan operasi	2,7 V – 3,6 V
Jumlah GPIO	36 pin
<i>Timer</i>	11 buah (termasuk <i>advanced-control timer</i>)
USART / SPI / I2C	3 / 5 / 3 buah
Keluaran PWM	Hingga 12 kanal melalui <i>timer</i>

Spesifikasi	Keterangan
ADC	12 bit, hingga 16 kanal
Kemasan	48 pin UFQFPN

(Sumber: Lembar data STM32F411xC/xE, STMicroelectronics [26])

Penggunaan *hardware timer* dalam *encoder mode* lebih efisien daripada interupsi eksternal karena pencacahan ditangani langsung oleh perangkat keras dan CPU hanya membaca *register* pencacah setiap periode pencuplikan [14]. Pada penelitian ini kedua *encoder* menggunakan *timer* dalam *encoder mode*, yaitu TIM3 untuk *encoder* motor kiri melalui PA6 dan PA7 serta TIM1 untuk *encoder* motor kanan melalui PA8 dan PA9. Alokasi *timer* dan pemetaan pin yang dipakai pada penelitian ini diuraikan pada Bab III.

2.11 NVIDIA Jetson Orin Nano

NVIDIA Jetson Orin Nano merupakan komputer tepi berbentuk modul *system-on-module* yang dipasang pada papan pembawa dan dirancang untuk aplikasi kecerdasan buatan serta robotika. Berbeda dengan komputer papan tunggal serbaguna, arsitektur Jetson memadukan prosesor ARM Cortex-A78AE enam inti dengan GPU NVIDIA Ampere 1.024 inti CUDA beserta 32 *Tensor Core*, sehingga mampu mengeksekusi operasi matriks dan konvolusi secara paralel dengan efisiensi daya tinggi [28].

Jetson Orin Nano berperan sebagai pengendali tingkat atas yang mengolah data awan titik YDLIDAR T-mini Plus 25, menjalankan antarmuka pemantauan dan pengendalian, menerima telemetri kecepatan serta pencacahan *encoder* dari STM32, dan mengirimkan perintah gerak kepada STM32F411CEU6. Pemisahan tugas diperlukan karena sistem operasi Linux tidak deterministik untuk *loop* kendali berperiode tetap. Estimasi posisi dan arah hadap robot dihitung pada tingkat ini, sedangkan umpan balik kendali kecepatan PI tetap berasal dari *encoder*.

Pemilihan Jetson Orin Nano dibandingkan komputer papan tunggal serbaguna didasarkan pada tiga pertimbangan. Pertama, akselerasi GPU membuka jalan bagi pengembangan lanjutan menuju deteksi buah cabai berbasis penglihatan mesin tanpa mengganti perangkat keras. Kedua, tersedia pustaka pengembangan resmi sensor YDLIDAR untuk Linux pada arsitektur ARM64 beserta dukungan pustaka antarmuka dan grafis, sehingga akuisisi awan titik, pemetaan, dan tampilan operator dapat dijalankan pada satu perangkat.

Ketiga, rentang tegangan masukan papan pembawa sebesar 9 V hingga 20 V mencakup jalur 12 V yang disediakan *buck converter* pada rancangan [28].

Tabel 2. 7 Spesifikasi NVIDIA Jetson Orin Nano (modul 8 GB dan papan pembawa developer kit)

Spesifikasi	Keterangan
CPU	ARM Cortex-A78AE v8.2 enam inti 64 bit, hingga 1,7 GHz
GPU	NVIDIA Ampere 1.024 inti CUDA dengan 32 <i>Tensor Core</i>
Kinerja komputasi AI	40 TOPS (mode standar); hingga 67 TOPS (mode <i>Super</i>)
Memori	8 GB LPDDR5, antarmuka 128 bit
Lebar pita memori	68 GB/s (standar); 102 GB/s (mode <i>Super</i>)
Penyimpanan	microSD dan M.2 Key M untuk NVMe SSD
USB / Ethernet	4 × USB 3.2 Gen2 dan 1 × USB Type-C / Gigabit Ethernet
<i>Header</i> ekspansi	40 pin (UART, SPI, I2C, I2S, GPIO)
Tegangan masukan	9 V – 20 V DC melalui <i>DC jack</i> (adaptor bawaan 19 V)
Anggaran daya	7 W – 15 W (standar); hingga 25 W (mode <i>Super</i>)
Perangkat lunak	NVIDIA JetPack SDK (Linux for Tegra, CUDA, TensorRT)

(Sumber: Dokumentasi produk dan spesifikasi *board* NVIDIA Corporation [28])

2.12 Komunikasi UART

Universal Asynchronous Receiver-Transmitter (UART) merupakan komunikasi serial asinkron untuk pertukaran data antarpengendali [29]. Konfigurasi *baud rate* dan format data harus sama pada kedua perangkat yang terhubung.

Pada sistem ini UART dipakai untuk jalur antara mikrokontroler dan pengendali tingkat atas. Sensor LiDAR juga berkomunikasi secara serial, tetapi melalui sambungan USB pada pengendali tingkat atas. Pemetaan pin, laju data, dan format bingkai jalur tersebut diuraikan pada Bab III. Penerimaan perintah dan pengiriman telemetri dijadwalkan terpisah dari eksekusi kendali agar keteraturan periode *loop* kendali tidak berubah.

2.13 Perangkat Lunak Pengembangan

STM32CubeIDE digunakan untuk mengembangkan *firmware* mikrokontroler dalam bahasa C, mencakup algoritma PI diskret, konversi pencacahan *encoder* menjadi RPM, pembangkitan PWM, penjadwalan *loop* kendali, serta komunikasi dengan pengendali tingkat atas. Konfigurasi awal perangkat keras disiapkan melalui STM32CubeMX, sedangkan fitur *live expression* dan *debugging* SWD digunakan untuk memantau variabel saat pengujian. Dokumentasi pengembangan tersedia pada situs STMicroelectronics [30].

Pada sisi pengendali tingkat atas, perangkat lunak dikembangkan sebagai satu aplikasi C++ dengan pustaka antarmuka Qt yang berjalan di atas sistem operasi Linux pada Jetson Orin Nano. Akuisisi LiDAR, pembentukan peta, odometri, komunikasi serial, dan tampilan antarmuka dijalankan sebagai untai (*thread*) terpisah di dalam aplikasi tersebut, tanpa kerangka kerja robotika seperti ROS. Pemisahan untai membuat pengolahan peta tidak menghambat pertukaran data dengan mikrokontroler serta memudahkan pengujian setiap bagian secara terpisah.

Produsen YDLIDAR menyediakan pustaka pengembangan resmi (YDLidar-SDK) untuk seri T-mini Plus sehingga integrasi sensor dapat dilakukan tanpa mengimplementasikan protokol tingkat bawah dari awal [25]. Antarmuka aplikasi menampilkan telemetri kecepatan dan pencacahan *encoder* kedua motor, menyediakan pengisian parameter PI, serta merekam data pengujian ke berkas CSV. Rincian antarmuka beserta alur penggunaannya diuraikan pada Bab III.