TUGAS AKHIR (AE43250)

PENGEMBANGAN *PATH PLANNING* A* PADA ROBOT KRSBI-B UNTUK *DYNAMIC OPPONENT AVOIDANCE* BERBASIS ROS DAN GAZEBO

Muhammad Jardin Saputra
NRP. 0922040086

Dosen Pembimbing
Agus Khumaidi, S.ST., M.T.
Riko Satrya Fajar Jaelani Putra, S.T., M.T.

PROGRAM STUDI D4 TEKNIK OTOMASI
JURUSAN TEKNIK KELISTRIKAN KAPAL
POLITEKNIK PERKAPALAN NEGERI SURABAYA
SURABAYA
2026

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR (AE43250)

PENGEMBANGAN *PATH PLANNING* A* PADA ROBOT KRSBI-B UNTUK *DYNAMIC OPPONENT AVOIDANCE* BERBASIS ROS DAN GAZEBO

Muhammad Jardin Saputra
NRP. 0922040086

Dosen Pembimbing
Agus Khumaidi, S.ST., M.T.
Riko Satrya Fajar Jaelani Putra, S.T., M.T.

PROGRAM STUDI D4 TEKNIK OTOMASI
JURUSAN TEKNIK KELISTRIKAN KAPAL
POLITEKNIK PERKAPALAN NEGERI SURABAYA
SURABAYA
2026

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

TUGAS AKHIR

PENGEMBANGAN *PATH PLANNING* A* PADA ROBOT KRSBI-B UNTUK *DYNAMIC OPPONENT AVOIDANCE* BERBASIS ROS DAN GAZEBO

Disusun Oleh :
Muhammad Jardin Saputra
0922040086

Diajukan Untuk Memenuhi Salah Satu Syarat Kelulusan
Program Studi D4 Teknik Otomasi
Jurusan Teknik Kelistrikan Kapal
POLITEKNIK PERKAPALAN NEGERI SURABAYA

Disetujui oleh Tim penguji Tugas Akhir Tanggal Ujian : 27 Juli 2026
Periode Wisuda : Oktober 2026

Menyetujui,

Dosen Penguji	NIDN/NUPTK	Tanda Tangan
1. Ryan Yudha Adhitya, S.ST., M.T.	(0016069101)	(.....)
2. Evi Nafiatus Sholikhah, S.Tr.T., M.Tr.T.	(0730069801)	(.....)
3. Dr. Mat Syai'in, S.T., M.T., Ph.D.	(0014117707)	(.....)
4. Agus Khumaidi, S.ST., M.T.	(0017089303)	(.....)

Dosen Pembimbing	NIDN/NUPTK	Tanda Tangan
1. Agus Khumaidi, S.ST., M.T.	(0017089303)	(.....)
2. Riko Satrya Fajar Jaelani Putra, S.T., M.T.	(6859776677130162)	(.....)

Menyetujui
Ketua Jurusan,

09.10.26

Isa Rachman, S.T., M.T.
NIP. 198008162008121001

Mengetahui
Koordinator Program Studi,

Agus Khumaidi, S.ST., M.T.
NIP. 199308172020121004

(Halaman ini sengaja dikosongkan)

PERNYATAAN BEBAS PLAGIAT

PPNS	**PERNYATAAN BEBAS PLAGIAT**	No. : F.WD I. 021 Date : 3 Nopember 2015 Rev. : 01 Page : 1 dari 1

Yang bertanda tangan dibawah ini :

Nama : Muhammad Jardin Saputra

NRP. : 0922040086

Jurusan/Prodi : Teknik Kelistrikan Kapal / D4 Teknik Otomasi

Dengan ini menyatakan dengan sesungguhnya bahwa :

Tugas Akhir yang akan saya kerjakan dengan judul :

Pengembangan *Path Planning* A* Pada Robot KRSBI-B Untuk *Dynamic Opponent Avoidance* Berbasis ROS dan Gazebo

Adalah **benar karya saya sendiri** dan **bukan plagiat dari karya orang lain.**

Apabila dikemudian hari terbukti terdapat plagiat dalam karya ilmiah tersebut, maka saya bersedia menerima **sanksi** sesuai ketentuan peraturan yang berlaku.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab.

Surabaya, 18 Agustus 2026

ang membuat pernyataan,

Muhammad Jardin Saputra
NRP. 0922040086

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

بسم الله الرحمن الرحيم

Puji syukur kehadirat Allah SWT atas segala rahmat, taufik, hidayah, serta karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "PENGEMBANGAN *PATH PLANNING* A* PADA ROBOT KRSBI-B UNTUK *DYNAMIC OPPONENT AVOIDANCE* BERBASIS ROS DAN GAZEBO" dengan baik dan tepat pada waktunya. Shalawat serta salam semoga senantiasa tercurah kepada junjungan Nabi Besar Muhammad SAW yang telah membawa umat manusia dari zaman kegelapan menuju zaman yang penuh dengan ilmu pengetahuan dan cahaya Islam.

Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Terapan pada Program Studi D-4 Teknik Otomasi, Jurusan Teknik Kelistrikan Kapal, Politeknik Perkapalan Negeri Surabaya.

Dalam penyusunan Tugas Akhir ini, penulis menyadari bahwa keberhasilan penyelesaiannya tidak terlepas dari doa, bantuan, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis menyampaikan rasa hormat dan ucapan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua serta seluruh keluarga tercinta yang selalu memberikan doa, kasih sayang, dukungan, semangat, dan motivasi yang tiada henti kepada penulis selama menempuh pendidikan hingga terselesaikannya Tugas Akhir ini.
2. Bapak Rachmad Tri Soelistijono, S.T., M.T., selaku Direktur Politeknik Perkapalan Negeri Surabaya.
3. Bapak Isa Rachman, S.T., M.T., selaku Ketua Jurusan Teknik Kelistrikan Kapal Politeknik Perkapalan Negeri Surabaya.
4. Bapak Agus Khumaidi, S.ST., M.T., selaku Koordinator Program Studi Teknik Otomasi serta Dosen Pembimbing I yang telah memberikan bimbingan, arahan, ilmu pengetahuan, motivasi, serta masukan yang sangat berarti selama proses penyusunan Tugas Akhir ini.
5. Bapak Riko Satrya Fajar Jaelani Putra, S.T., M.T., selaku Dosen Pembimbing II yang telah memberikan bimbingan, saran, perhatian, serta motivasi sehingga

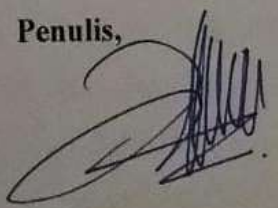
penulis dapat menyelesaikan Tugas Akhir ini dengan baik.

6. Bapak Ryan Yudha Adhitya, S.ST., M.T., selaku Koordinator Tugas Akhir Program Studi Teknik Otomasi yang telah memberikan arahan dan dukungan selama pelaksanaan Tugas Akhir.
7. Seluruh dosen, staf, dan tenaga kependidikan Politeknik Perkapalan Negeri Surabaya yang telah memberikan ilmu, pelayanan, dan pengalaman selama penulis menempuh pendidikan.
8. Seluruh dosen Program Studi Teknik Otomasi yang telah membekali penulis dengan ilmu pengetahuan, pengalaman, dan wawasan yang sangat bermanfaat.
9. Seluruh keluarga besar Teknik Otomasi angkatan 2022 yang telah menjadi teman belajar, berdiskusi, dan berjuang bersama selama masa perkuliahan.
10. Tim Gerhana Dewaruci, yang telah memberikan pengalaman, ilmu, serta kesempatan kepada penulis untuk berkembang di bidang robotika dan kompetisi Robot Sepak Bola Indonesia Beroda.
11. Seluruh teman-teman Kontrakan SPR yang selalu memberikan dukungan, semangat, serta doa selama penyusunan Tugas Akhir.
12. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang telah memberikan bantuan, doa, dan dukungan selama proses penyusunan Tugas Akhir ini.

Semoga Allah SWT membalas segala kebaikan, bantuan, dan dukungan yang telah diberikan dengan balasan yang berlipat ganda. Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan penelitian ini di masa mendatang. Akhir kata, penulis berharap semoga Tugas Akhir ini dapat memberikan manfaat bagi perkembangan ilmu pengetahuan dan teknologi, khususnya di bidang otomasi, dan robotika.

Surabaya, 12 Juli 2026

Penulis,



Muhammad Jardin Saputra

PENGEMBANGAN *PATH PLANNING* A* PADA ROBOT KRSBI-B UNTUK *DYNAMIC OPPONENT AVOIDANCE* BERBASIS ROS DAN GAZEBO

Muhammad Jardin Saputra

ABSTRAK

Kompetisi Robot Sepak Bola Beroda Indonesia (KRSBI-B) membutuhkan sistem navigasi yang mampu beradaptasi terhadap perubahan posisi robot lawan selama pertandingan. Metode perencanaan jalur a* sebelumnya hanya menghasilkan jalur satu kali, sehingga kurang efektif pada lingkungan yang dinamis. Penelitian ini mengusulkan algoritma perencanaan jalur A* dinamis untuk meningkatkan navigasi robot penyerang. Sistem dikembangkan menggunakan ROS 2 Humble, Gazebo, dan YOLOv5s dengan kamera omnidirectional untuk mendeteksi robot lawan. Posisi rintangan yang terdeteksi direpresentasikan dalam occupancy grid, sehingga algoritma A* dapat menghitung ulang jalur ketika terjadi perubahan lingkungan. Metode ini dievaluasi melalui pengujian simulasi dan robot nyata. Pada simulasi Gazebo, robot berhasil mencapai target pada seluruh 10 skenario dengan tingkat keberhasilan 100%. Waktu tempuh berkisar antara 25,31 hingga 56,17 detik, sedangkan panjang jalur berkisar antara 3,16 hingga 7,62 meter tanpa terjadi tabrakan. Pengujian robot nyata dilakukan dalam 8 skenario, dan robot penyerang berhasil mencapai Goal 1 (3000, 10800) atau Goal 2 (5200, 10800) pada setiap percobaan. Seluruh percobaan berhasil dengan waktu tempuh 22,84–33,27 detik dan panjang jalur 4,96–7,46 meter. Hasil tersebut menunjukkan bahwa A* dinamis mampu menyediakan navigasi yang adaptif dan andal bagi robot KRSBI-B pada lingkungan dinamis.

Kata kunci: Robot Sepak Bola, Path Planning A*, Dynamic Replanning, ROS 2, YOLOV5S.

(Halaman ini sengaja dikosongkan)

DEVELOPMENT OF A PATH PLANNING ON KRSBI-B FOR DYNAMIC OPPONENT AVOIDANCE BASED ON ROS AND GAZEBO*

Muhammad Jardin Saputra

ABSTRACT

The Indonesian Wheeled Soccer Robot Contest (KRSBI-B) requires a navigation system that can adapt to changes in opponent robot positions during a match. Conventional A path planning methods generate a path only once, making them less effective in dynamic environments. This study proposes a dynamic A* path planning algorithm to improve attacker robot navigation. The system was developed using ROS 2 Humble, Gazebo, and YOLOV5S with an omnidirectional camera for opponent detection. Detected obstacle positions were represented in an occupancy grid, allowing the A* algorithm to recalculate the path when the environment changed. The method was evaluated through simulation and real robot experiments. In the Gazebo simulation, the robot reached the target in all 10 scenarios with a 100% success rate. Travel times ranged from 25.31 to 56.17 seconds, while path lengths ranged from 3.16 to 7.62 meters without collisions. Real robot tests were conducted in 8 scenarios, and the attacker reached Goal 1 (3000, 10800) or Goal 2 (5200, 10800) in every trial. All eight trials were successful. Travel times ranged from 22.84 to 33.27 seconds, while path lengths ranged from 4.96 to 7.46 meters. These results show that Dynamic A* provides adaptive and reliable navigation for KRSBI-B robots in dynamic environments.*

Keywords: *Soccer Robot, A* Path Planning, Dynamic Replanning, ROS 2, YOLOV5S.*

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR NOTASI

r	: Jari-jari roda robot (mm)
K_{roda}	: Keliling roda robot (mm)
Res_{enc}	: Resolusi rotary encoder, yaitu jumlah pulsa dalam satu putaran penuh (pulse per revolution)
$Pulse$	: Jumlah pulsa yang dihasilkan oleh rotary encoder
$\frac{Pulse}{mm}$	: Jumlah pulsa encoder per satuan jarak tempuh (pulse/mm)
s_i	: Jarak tempuh roda ke-i hasil pembacaan rotary encoder (mm)
θ_i	: Sudut orientasi roda ke-i terhadap sumbu referensi robot (rad)
$Xrobot$	: Perubahan posisi robot pada sumbu X sistem koordinat kartesian
$Yrobot$	: Perubahan posisi robot pada sumbu Y sistem koordinat kartesian
θ	: Sudut orientasi robot terhadap sumbu X global (rad)
ϕ	: Sudut roll robot
θ_g	: Sudut pitch robot
ψ	: Sudut yaw robot
ω	: Kecepatan sudut hasil pengukuran gyroscope (rad/s)
Δt	: Interval waktu pengambilan data (s)
Δs	: Perubahan jarak translasi robot (mm)
x, y	: Koordinat posisi objek atau robot pada bidang citra atau peta
$(x, y)_{img}$	: Koordinat objek pada sistem koordinat citra
$N\ x\ N$	: Ukuran grid citra pada algoritma YOLOV5S
$f\ (n)$	: nilai evaluasi total pada node n, yang digunakan untuk menentukan prioritas pencarian jalur.
$g(n)$	: nilai perpindahan dari posisi awal menuju node n.
$h(n)$	: nilai heuristik berupa estimasi jarak dari node $n$menuju titik tujuan.
f	: Panjang fokus kamera

(Halaman ini sengaja kosongkan)

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Gerhana Dewaruci adalah tim robotika unggulan dari Politeknik Perkapalan Negeri Surabaya (PPNS) yang secara aktif mengikuti ajang Kontes Robot Indonesia (KRI), khususnya pada kategori Kontes Robot Sepak Bola Indonesia Beroda (KRSBI-B). Berdasarkan pada penelitian sebelumnya yang dilakukan oleh (Yaqin, 2025), metode *path planning* menggunakan algoritma A* telah diterapkan sebagai solusi untuk menentukan lintasan pergerakan robot sepak bola beroda. Namun, sistem yang dikembangkan pada penelitian tersebut masih menggunakan pendekatan sekali ambil atau *single planning***,** di mana proses perencanaan jalur hanya dilakukan satu kali pada kondisi awal tanpa adanya pembaruan lintasan secara berkelanjutan. Jalur yang dihasilkan kemudian dijadikan acuan tetap bagi robot dalam mencapai tujuan, tanpa mempertimbangkan perubahan kondisi lingkungan yang terjadi selama robot bergerak.

Pendekatan tersebut menyebabkan sistem navigasi hanya bekerja secara optimal pada lingkungan yang statis, di mana posisi rintangan dan objek di sekitar robot diasumsikan tidak mengalami perubahan yang signifikan. Pada kondisi pertandingan KRSBI-B yang sesungguhnya, asumsi ini tidak dapat dipenuhi karena robot lawan bergerak secara aktif mengikuti strategi permainan, baik untuk bertahan maupun menyerang. Pergerakan lawan yang dinamis tersebut secara langsung mengubah konfigurasi lingkungan lapangan, sehingga jalur yang telah direncanakan sebelumnya berpotensi menjadi tidak aman dan berisiko menyebabkan tabrakan antar robot.

Selain itu, sistem pada penelitian (Yaqin, 2025) belum memiliki mekanisme untuk melakukan pembaruan jalur secara *real-time* berdasarkan informasi terbaru dari sensor penglihatan. Akibatnya, robot tidak mampu menyesuaikan lintasan pergerakannya ketika terjadi perubahan posisi lawan di lapangan. Keterbatasan ini menjadikan sistem sulit diterapkan dalam lingkungan pertandingan lomba yang menuntut respons cepat, adaptif, dan berkelanjutan terhadap kondisi lapangan yang selalu berubah.

Berdasarkan permasalahan tersebut, penelitian tugas akhir ini diusulkan untuk mengembangkan sistem navigasi robot sepak bola beroda yang adaptif terhadap perubahan kondisi lingkungan. Sistem yang dikembangkan tidak hanya melakukan perencanaan jalur menggunakan algoritma A*, tetapi juga melakukan pembaruan lintasan secara kontinu berdasarkan perubahan posisi robot lawan yang terdeteksi selama pertandingan berlangsung. Dengan pendekatan ini, robot diharapkan mampu menghindari lawan secara dinamis, menjaga kelancaran pergerakan menuju target, serta meningkatkan efektivitas strategi permainan dalam kondisi pertandingan yang sebenarnya.

Penggunaan algoritma A* sebagai metode path planning dinilai efektif karena mampu menghasilkan lintasan berdasarkan fungsi evaluasi yang mempertimbangkan kondisi lingkungan. Sementara itu, *Robot Operating System 2* (ROS 2) menyediakan platform yang fleksibel dan modular untuk mengintegrasikan berbagai fungsi robot seperti pemrosesan data sensor, manajemen komunikasi antar modul, serta eksekusi algoritma navigasi. Untuk mendukung proses pengujian dan validasi sistem navigasi yang dikembangkan, penelitian ini memanfaatkan Gazebo sebagai lingkungan simulasi tiga dimensi yang terintegrasi dengan ROS 2. Gazebo memungkinkan pemodelan lapangan KRSBI-B, robot tim, serta robot lawan yang bergerak secara dinamis, sehingga pengujian algoritma *path planning* A* dengan mekanisme pembaruan jalur secara *real-time* dapat dilakukan secara aman dan terkontrol. Gambar 1.1 merupakan ajang lomba di KRI 2024.

Gambar 1. 1 Lomba KRI
(Penulis, 2026)

1.2 Rumusan Masalah

Rumusan Masalah pada penelitin ini antara lain :

1. Bagaimana merancang sistem deteksi posisi robot lawan yang mampu bekerja secara kontinu pada lingkungan pertandingan KRSBI-B yang dinamis?
2. Bagaimana penerapan metode path planning A* untuk penghindaran robot lawan secara dinamis selama proses navigasi?
3. Bagaimana integrasi ROS 2 dengan sistem deteksi dan *path planning* dalam strategi navigasi robot sepak bola beroda?

1.3 Batasan Penelitian

Batasan pada penelitian ini adalah :

1. Pengujian deteksi objek menggunakan kamera omnidirectional dilakukan pada rentang jarak 500–5000 mm. Informasi jarak dan posisi yang diperoleh dari sistem bersifat estimasi relatif terhadap robot dan tidak digunakan sebagai pengukuran posisi absolut.
2. Area pengujian mengacu pada standar lapangan pertandingan KRSBI-B berukuran 12 × 8 meter. Namun, dalam pelaksanaannya, pengujian dilakukan pada lapangan berukuran 5 × 8 meter yang disesuaikan dengan keterbatasan ruang penelitian. Meskipun demikian, skenario pengujian tetap dirancang secara proporsional terhadap kondisi sebenarnya, dan sistem yang dikembangkan berbasis koordinat relatif, sehingga perbedaan skala tidak mempengaruhi prinsip kerja algoritma yang digunakan.
3. Penerapan algoritma perencanaan lintasan A* difokuskan pada skenario permainan dengan satu robot aktif, sehingga interaksi dan koordinasi multi-robot belum menjadi cakupan dipenelitian ini.

1.4 Tujuan Penelitian

Tujuan penelitian ini sebagai berikut :

1. Mengembangkan sistem deteksi dan estimasi posisi relatif robot lawan untuk mendukung navigasi pada lingkungan pertandingan KRSBI-B

yang dinamis.

2. Mengimplementasikan metode *path planning* A* dengan mekanisme *dynamic replanning* untuk melakukan penghindaran robot lawan secara dinamis.
3. Mengintegrasikan ROS dengan sistem deteksi dan metode *path planning* sebagai strategi navigasi robot penyerang sepak bola beroda yang adaptif secara otonom.

1.5 Manfaat Penelitian

Manfaat penelitian ini antara lain:

1. Robot penyerang memiliki kemampuan navigasi yang lebih adaptif, aman, dan efisien dalam menghadapi lawan yang bergerak dinamis.
2. Meningkatkan performa sistem navigasi robot penyerang pada kompetisi KRSBI-B, khususnya dalam situasi pertandingan nyata.
3. Menjadi referensi pengembangan sistem *path planning real-time* berbasis ROS dan algoritma A* untuk penelitian robotika sepak bola selanjutnya.

BAB 2
TINJAUAN PUSTAKA

2.1 Studi Penelitian Terdahulu

Penelitian ini disusun berdasarkan hasil kajian dari penelitian-penelitian sebelumnya yang relevan dan selanjutnya dikembangkan dalam penelitian ini. Ringkasan kajian penelitian terdahulu yang digunakan disajikan pada Tabel 2.1.

Tabel 2. 1 Tinjauan Pustaka Terdahulu

Penulis (Tahun)	Judul	Kelebihan dan Kekurangan	Perbedaan dan Pengembangan
Muhammad Ainul Yaqin (2025)	PENERAPAN *PATH PLANNING* A* UNTUK STRATEGI ROBOT SEPAK BOLA BERODA MENGGUNAKAN *ROBOT OPERATING SYSTEM* (ROS)	Pada penelitian terdahulu, *path planning* A* telah berhasil diintegrasikan dengan *Robot Operating System 1* (ROS 1) untuk mendukung penghindaran robot lawan pada sistem robot sepak bola beroda. Namun, perencanaan jalur masih dilakukan secara sekali (*single planning*), sehingga sistem belum mampu beradaptasi secara optimal terhadap perubahan posisi lawan yang bergerak dinamis selama	Penelitian ini mengembangkan sistem sebelumnya dengan menambahkan mekanisme pembaruan jalur *(replanning)* dan mendeteksi lawan bergerak secara *real-time* berbasis kamera 360 dan diintegrasikan dengan ros 2, sehingga robot penyerang mampu beradaptasi terhadap pergerakan lawan secara dinamis.

Tabel 2.1 Tinjauan Pustaka Terdahulu (Lanjutan)

Penulis (Tahun)	Judul	Kelebihan dan Kekurangan	Perbedaan dan Pengembangan
		pertandingan jalur optimal. Namun, pengujian masih dilakukan pada lingkungan statis tanpa melibatkan rintangan bergerak..	
Adi Rahmad Ramadhan (2025)	ESTIMASI JARAK NYATA ROBOT SEPAK BOLA BERODA MENGGUNAKAN KAMERA Stereo Vision DENGAN METODE YOLOV5S	Penelitian ini menerapkan metode YOLOV5S dan kamera stereo vision untuk mendeteksi objek dan mengestimasi jarak nyata. Sistem mampu memberikan informasi jarak secara akurat, namun belum mengintegrasikan hasil estimasi tersebut ke dalam perencanaan jalur dan penghindaran lawan secara dinamis.	Penelitian ini mengembangkan studi sebelumnya dengan memanfaatkan informasi posisi dan jarak robot lawan sebagai input dalam algoritma A* sehingga robot penyerang mampu melakukan penghindaran lawan secara *real-time* pada lingkungan pertandingan yang dinamis.
Wahyu Darmawan (2023)	OPTIMASI STRATEGI KEPUTUSAN ROBOT PENYERANG	Penelitian ini berhasil menerapkan metode *Decision Tree* untuk meningkatkan kemampuan robot	Penelitian ini mengembangkan penelitian sebelumnya dengan menambahkan deteksi dan penghindaran robot

Tabel 2.1 Tinjauan Pustaka Terdahulu (Lanjutan)

Penulis (Tahun)	Judul	Kelebihan dan Kekurangan	Perbedaan dan Pengembangan
	MENGGUNAKAN *METODE DECISION* TREE PADA ROBOT SEPAK BOLA BERODA	penyerang dalam mengambil keputusan berdasarkan posisi bola, robot kawan, dan gawang lawan. Sistem mampu meningkatkan persentase keberhasilan mencetak gol, namun belum mempertimbangkan pergerakan robot lawan secara dinamis dalam proses pengambilan keputusan dan navigasi	Lawan secara dinamis melalui integrasi sistem persepsi dengan metode *path planning A*,* sehingga robot penyerang mampu menyesuaikan lintasan dan strategi secara *real-time* pada lingkungan pertandingan KRSBI-B.
Luluk Indah Safitri (2023)	Implementasi Algoritma Path Planning A* Pada Base Station Robot Sepak Bola Beroda	Penelitian ini mengimplementasikan algoritma *path planning* A* pada *base station* robot sepak bola beroda dalam lingkungan simulasi statis. Namun, sistem yang dikembangkan belum mampu melakukan pembaruan jalur secara *real-time*, belum	Penelitian ini mengintegrasikan sistem persepsi visual berbasis YOLOV5S untuk memperoleh informasi posisi robot lawan secara *real-time*. Selain itu, algoritma A* untuk memungkinkan penghindaran rintangan dan pembaruan jalur secara dinamis, serta diintegrasikan dalam

Tabel 2.1 Tinjauan Pustaka Terdahulu (Lanjutan)

Penulis (Tahun)	Judul	Kelebihan dan Kekurangan	Perbedaan dan Pengembangan
		mempertimbangkan pergerakan robot lawan yang bersifat dinamis, serta belum terintegrasi dengan robot secara langsung.	arsitektur ROS2 dan divalidasi melalui simulasi Gazebo agar lebih sesuai dengan kondisi pertandingan KRSBI-B yang dinamis.

Penelitian yang dilakukan oleh Muhammad Ainul Yaqin (2025) dengan judul "Penerapan *Path Planning A** untuk Strategi Robot Sepak Bola Beroda Menggunakan Robot Operating System (ROS)", evaluasi kinerja sistem difokuskan pada parameter waktu tempuh robot dalam mencapai target. Algoritma A* diintegrasikan dengan ROS 1 untuk melakukan perencanaan jalur berdasarkan peta grid lapangan, di mana posisi robot dan robot lawan direpresentasikan dalam koordinat (x, y). Setiap sel grid diberi status bebas atau terhalang berdasarkan posisi robot lawan pada satu kondisi waktu tertentu, sehingga jalur yang dihasilkan merupakan jalur optimal pada saat perencanaan awal dilakukan. Namun, perencanaan jalur pada penelitian tersebut masih bersifat *single planning*, yaitu perhitungan jalur hanya dilakukan sekali di awal tanpa mekanisme pembaruan ketika terjadi perubahan posisi robot lawan. Akibatnya, meskipun waktu tempuh awal robot relatif efisien, performa sistem menjadi kurang optimal ketika robot lawan bergerak secara dinamis selama pertandingan, karena jalur yang digunakan tidak lagi merepresentasikan kondisi lapangan terkini. Hal ini menyebabkan potensi peningkatan waktu tempuh, manuver yang tidak efisien, serta risiko tabrakan dengan rintangan bergerak.

Berdasarkan keterbatasan tersebut, penelitian ini dikembangkan untuk meningkatkan performa navigasi robot, khususnya dalam menghadapi rintangan dinamis, dengan tetap menggunakan algoritma A* sebagai metode utama perencanaan jalur. Pengembangan dilakukan dengan menambahkan mekanisme pembaruan status rintangan dan perencanaan ulang jalur *(dynamic replanning)*

secara berkala berdasarkan informasi visual yang diperoleh selama proses navigas.

Pada sistem yang diusulkan, posisi robot lawan diperbarui secara kontinu melalui hasil pengolahan citra dari kamera omni 360°, kemudian direpresentasikan kembali pada peta grid sebagai rintangan dinamis. Setiap perubahan signifikan pada posisi robot lawan akan memicu proses perencanaan ulang jalur A*, sehingga robot penyerang mampu menyesuaikan lintasan pergerakannya secara adaptif terhadap kondisi lapangan terkini. Selain itu, sistem diintegrasikan dengan ROS 2 yang mendukung komunikasi terdistribusi dan eksekusi node secara real-time, sehingga proses pembaruan jalur dapat dilakukan dengan latensi yang lebih rendah dibandingkan ROS 1. Dengan pendekatan ini, robot tidak hanya menggunakan jalur hasil perencanaan awal, tetapi juga dapat melakukan pembaruan lintasan ketika kondisi lingkungan berubah. Oleh karena itu, pengembangan yang diusulkan diharapkan menghasilkan sistem navigasi yang lebih adaptif, lebih optimal secara waktu, dan lebih andal dalam menghadapi rintangan dinamis hingga robot berhasil mencapai target dan melakukan aksi shooting ke gawang lawan.

Penelitian yang dilakukan oleh (Ramadhan et al., 2025) dengan judul "Estimasi Jarak Nyata Robot Sepak Bola Beroda Menggunakan Kamera Stereo Vision dengan Metode YOLOV5S" berfokus pada pengembangan sistem persepsi visual robot sepak bola beroda. Penelitian ini menerapkan metode YOLOV5S untuk mendeteksi objek robot lawan dan memanfaatkan kamera stereo vision untuk mengestimasi jarak nyata berdasarkan perbedaan sudut pandang kedua kamera. Informasi posisi dan jarak robot lawan diperoleh melalui proses deteksi dan perhitungan disparitas citra secara visual. Hasil pengujian menunjukkan bahwa sistem mampu memberikan estimasi jarak robot lawan dengan tingkat akurasi yang cukup baik pada berbagai kondisi pengujian. Namun, penelitian ini masih terbatas pada aspek persepsi dan estimasi jarak, sehingga informasi posisi dan jarak yang diperoleh belum dimanfaatkan secara langsung dalam sistem perencanaan jalur dan penghindaran robot lawan secara dinamis. Berdasarkan keterbatasan tersebut, penelitian ini dikembangkan dengan mengintegrasikan informasi posisi robot lawan hasil deteksi visual ke dalam algoritma A* sebagai dasar pembaruan status rintangan dan perencanaan ulang jalur secara *real-time*.

Penelitian yang dilakukan oleh (Darmawan et al., 2023) dengan judul

"Optimasi Strategi Keputusan Robot Penyerang Menggunakan Metode *Decision Tree* pada Robot Sepak Bola Beroda" berfokus pada perancangan sistem pengambilan keputusan robot penyerang sepak bola beroda dengan menerapkan metode *Decision Tree*. Sistem ini memanfaatkan berbagai informasi kondisi permainan, seperti posisi bola, posisi robot kawan, dan posisi gawang lawan, untuk menentukan aksi robot penyerang secara strategis. Hasil penelitian menunjukkan bahwa penerapan metode *Decision Tree* mampu meningkatkan kualitas pengambilan keputusan robot penyerang dan meningkatkan peluang keberhasilan mencetak gol. Namun, penelitian ini belum membahas secara spesifik aspek perencanaan jalur dan penghindaran robot lawan yang bergerak secara dinamis, serta belum mengintegrasikan sistem keputusan dengan mekanisme navigasi adaptif. Oleh karena itu, penelitian ini dikembangkan dengan mengombinasikan sistem pengambilan keputusan penyerang dengan mekanisme *path planning* A* yang mampu memperbarui jalur secara dinamis berdasarkan perubahan posisi robot lawan selama pertandingan.

Selanjutnya, penelitian yang dilakukan oleh (Safitri et al., 2023) dengan judul "Implementasi Algoritma Path Planning A pada Base Station Robot Sepak Bola Beroda" mengimplementasikan algoritma A* sebagai metode perencanaan jalur pada sistem robot sepak bola beroda yang dijalankan pada base station. Lingkungan lapangan direpresentasikan dalam bentuk peta grid, di mana setiap sel merepresentasikan area bebas atau terhalang berdasarkan kondisi lingkungan simulasi. Algoritma A* digunakan untuk menghitung jalur terpendek dari posisi awal robot menuju target dengan mempertimbangkan rintangan yang terdapat pada peta simulasi. Hasil pengujian menunjukkan bahwa algoritma A* mampu menghasilkan jalur pergerakan yang optimal dan efisien dalam lingkungan yang telah didefinisikan. Namun, sistem yang dikembangkan masih bersifat statis, di mana perencanaan jalur hanya dilakukan satu kali tanpa mekanisme pembaruan jalur secara *real-time*. Selain itu, penelitian ini belum mempertimbangkan pergerakan robot lawan yang bersifat dinamis dan belum terintegrasi dengan sistem persepsi visual serta middleware robotik. Berdasarkan keterbatasan tersebut, penelitian ini dikembangkan dengan mengintegrasikan sistem persepsi visual dan mekanisme dynamic replanning berbasis ROS 2 agar lebih sesuai dengan kondisi

pertandingan yang bersifat dinamis.

2.2. ROS *(Robot Operating System)*

Pada penelitian ini digunakan *Robot Operating System* 2 (ROS 2) *Humble Hawksbill*, yaitu distribusi ROS 2 dengan status *Long-Term Support* (LTS) yang berjalan pada sistem operasi Ubuntu 22.04. ROS 2 Humble dipilih karena menyediakan arsitektur yang lebih modern, mendukung komunikasi berbasis *Data Distribution Service (DDS)*, serta memiliki stabilitas dan dukungan jangka panjang dari komunitas pengembang (Bonci et al., 2023). Selain itu, ROS 2 Humble telah dilengkapi dengan berbagai pustaka perangkat lunak robotika yang mendukung pengembangan sistem robotik secara lebih andal, skalabel, dan efisien. Gambar 2.1 menunjukan logo ROS 2 Humble.

Gambar 2. 1 Logo ROS 2 *Humble*
(ROS, 2023)

Robot Operating System 2 (ROS 2) merupakan platform perangkat lunak robotika berbasis open source yang dikembangkan sebagai penerus Robot Operating System (ROS) dengan peningkatan pada aspek komunikasi, keandalan, keamanan, dan dukungan terhadap sistem terdistribusi. ROS 2 berfungsi sebagai middleware yang memungkinkan komunikasi dan integrasi antarperangkat lunak maupun antara perangkat lunak dengan perangkat keras robot secara modular dan fleksibel. Berbeda dengan ROS generasi sebelumnya yang menggunakan protokol komunikasi khusus, ROS 2 memanfaatkan Data Distribution Service (DDS) sebagai lapisan komunikasi, sehingga mendukung pertukaran data secara real-time, memiliki kualitas layanan (Quality of Service atau QoS) yang dapat dikonfigurasi,

serta lebih andal pada lingkungan dengan banyak node dan perangkat (Sucipto et al., 2021). Melalui arsitektur tersebut, setiap komponen sistem dapat dikembangkan, dijalankan, dan dikelola secara terpisah, namun tetap saling berkomunikasi menggunakan mekanisme publish-subscribe, service, dan action. Penggunaan ROS 2 bertujuan untuk menyederhanakan proses pengembangan perangkat lunak robot dengan memanfaatkan berbagai paket (package) yang telah tersedia, sehingga pengembang tidak perlu membangun seluruh sistem dari awal. Selain itu, ROS 2 mendukung pengembangan sistem yang bersifat modular, mudah dikembangkan (scalable), serta mempermudah integrasi berbagai sensor, aktuator, dan algoritma dalam satu sistem robot. (Marinho, 2023).

2.2.1 Fitur pada ROS 2

ROS 2 memiliki berbagai keunggulan yang menjadikannya banyak digunakan pada bidang industri maupun akademik. Penerapan kerangka kerja pemrosesan terdistribusi memungkinkan pengembangan sistem yang bersifat skalabel, baik pada arsitektur prosesor tunggal maupun multi-prosesor. Ketersediaan pustaka yang bersifat *open source* dalam jumlah besar mendukung pengembangan sistem robotika secara cepat dan efisien. Selain itu, ROS 2 juga menyediakan beragam perangkat bantu berkinerja tinggi, seperti simulasi, visualisasi, serta perencanaan jalur, yang mempermudah proses perancangan dan pengujian sistem robotika (Mobaiyen & Ashjaei, 2022). Selain *library,* ROS 2 juga dilengkapi dengan berbagai alat berkinerja tinggi, seperti alat simulasi, visualisasi, dan perencanaan jalur. Beberapa fitur utama ROS 2 antara lain :

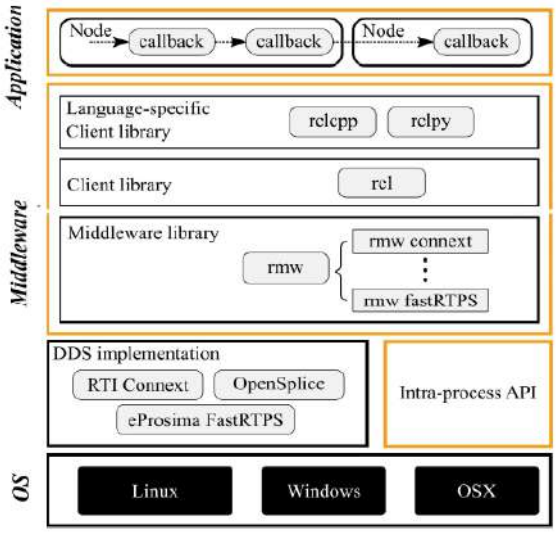
- Mekanisme *discovery*, transport, dan serialisasi data berbasis *Data Distribution Service* (DDS). DDS *(Data Distribution Service)* merupakan middleware komunikasi yang digunakan oleh ROS 2 untuk pertukaran data antar node secara terdistribusi.
- Komunikasi *publish/subscribe* melalui topic.
- Pengaturan *Quality of Service* (QoS) untuk menangani kondisi jaringan yang tidak ideal. Fungsi utama QoS: Mengatur reliabilitas data (reliable / best effort), Mengatur riwayat data (history & depth), Menentukan durability (data lama bisa dikirim ke subscriber baru), Mengatur deadline dan latency
- Dukungan keamanan melalui DDS-*Security.*

- Sistem *launch* untuk mengoordinasikan eksekusi beberapa node.
- Dukungan awal terhadap pemrograman *real-time.*

2.2.2 ROS 2 Arsitektur

ROS 2 dirancang menggunakan struktur berlapis yang terdiri dari beberapa tingkatan sistem. Pada lapisan aplikasi, ROS 2 menyediakan *client library* yang disesuaikan dengan bahasa pemrograman tertentu, seperti C++ dan Python. *Client library* ROS (rcl) berfungsi untuk memastikan keseragaman dan kompatibilitas antar aplikasi yang dikembangkan menggunakan bahasa pemrograman yang berbeda melalui antarmuka pemrograman aplikasi (API) (Mobaiyen & Ashjaei, 2022).

Pada lapisan berikutnya, *ROS middleware library* (rmw) bertugas menjembatani komunikasi antara rcl dan *Data Distribution Service* (DDS). DDS merupakan standar komunikasi industri yang mendukung pertukaran data secara *real-time* dan digunakan dalam ROS 2 untuk memenuhi kebutuhan komunikasi dengan batasan waktu yang ketat, khususnya pada mekanisme komunikasi publisher–subscriber. Dengan arsitektur pada Gambar 2.2, ROS 2 mampu mendukung sistem robotika yang andal, *real-time*, dan mudah dikembangkan (Mobaiyen & Ashjaei, 2022).

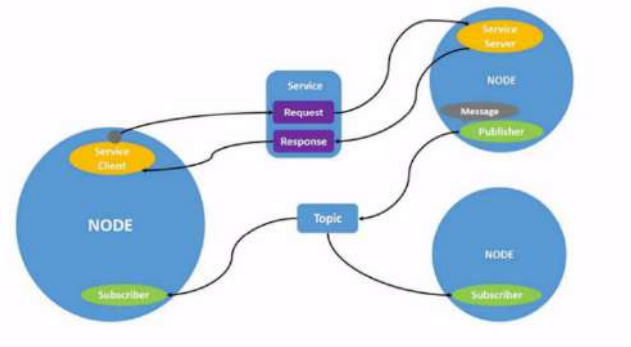


Gambar 2. 2 Arsitektur ROS 2 *Humble*
(Choi et al., 2021)

2.2.3 Abstraksi ROS 2

ROS 2 merupakan sebuah middleware yang tersusun atas komponen *Node,*

Topic, dan *Service* yang memungkinkan terjadinya transmisi pesan antar proses ROS yang berbeda melalui mekanisme *publish/subscribe* Gambar 2.3. Komponen utama dalam setiap sistem ROS 2 adalah ROS *graph*, yang merepresentasikan jaringan node beserta hubungan komunikasi di antara node-node tersebut (Mobaiyen & Ashjaei, 2022).



Gambar 2. 3 Komunikasi antar Node
(Mobaiyen & Ashjaei, 2022)

a. Node pada ROS 2

Node pada ROS 2 berfungsi sebagai unit komputasi yang menjalankan tugas tertentu dalam sistem robotika. Setiap node dirancang untuk menangani satu fungsi spesifik, seperti pengendalian aktuator, pemrosesan data sensor, atau pengolahan citra. Komunikasi antar node dapat dilakukan melalui beberapa mekanisme, antara lain *topic, service, action*, dan parameter (Mobaiyen & Ashjaei, 2022).

b. Topic pada ROS 2

Topic merupakan media komunikasi pada ROS 2 yang digunakan untuk pertukaran data dalam sistem terdistribusi. Melalui topic, sebuah node dapat mengirimkan data sebagai publisher maupun menerima data sebagai subscriber. Mekanisme ini memungkinkan satu node berkomunikasi dengan banyak node lainnya secara simultan tanpa ketergantungan langsung antar node (Mobaiyen & Ashjaei, 2022).

c. Service pada ROS 2

Service pada ROS 2 menyediakan mekanisme komunikasi berbasis permintaan dan respons antara node. Berbeda dengan komunikasi melalui topic yang berlangsung secara kontinu, service hanya aktif ketika terdapat

permintaan dari *node client* kepada *node server*. Mekanisme ini umumnya digunakan untuk fungsi yang membutuhkan respons langsung atau pengambilan keputusan tertentu (Mobaiyen & Ashjaei, 2022).

d. Parameter pada ROS 2

Parameter digunakan sebagai media konfigurasi untuk mengatur perilaku node dalam ROS 2. Nilai parameter dapat disimpan dalam berbagai tipe data, seperti bilangan bulat, bilangan desimal, nilai logika, teks, maupun daftar data (Mobaiyen & Ashjaei, 2022).

2.3. Pengolahan Citra Digital

Pemrosesan dan pengolahan citra merupakan serangkaian metode yang digunakan untuk menerapkan berbagai operasi pada suatu gambar dengan tujuan meningkatkan kualitas tampilan citra atau mengekstraksi informasi penting yang terkandung di dalamnya (Marpaung et al., 2022). Pengolahan citra merupakan salah satu bidang penting yang banyak diterapkan pada berbagai aplikasi nyata, antara lain pengenalan pola, penginderaan jauh melalui satelit dan wahana udara, serta pengembangan sistem *machine vision* (Marpaung et al., 2022).

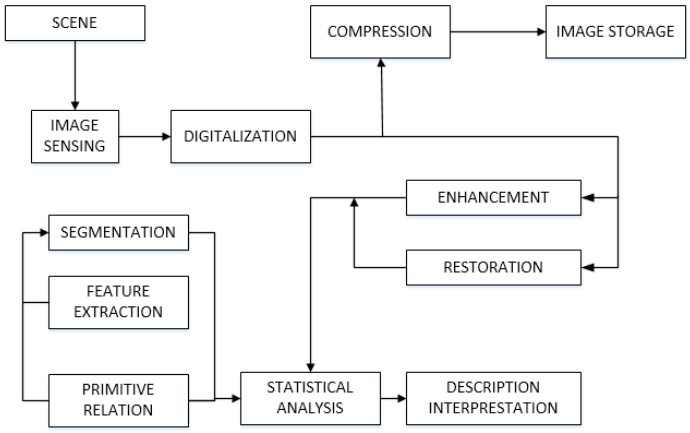
Citra digital dapat didefinisikan sebagai representasi gambar dalam bentuk kumpulan nilai numerik dua dimensi yang tersusun dalam sebuah array digital, di mana setiap nilai merepresentasikan tingkat intensitas atau kecerahan pada setiap piksel penyusun citra (Romadloni et al., 2023). Tujuan utama pengolahan citra digital adalah mengubah citra mentah menjadi bentuk yang lebih informatif dan mudah dianalisis oleh manusia atau sistem komputer dalam berbagai aplikasi nyata (Mursalim & Aprilia, 2024). Dalam proses pengolahan citra visual, terdapat beberapa tahapan dan teknik yang umum diterapkan, antara lain:

1. ***Image Preprocessing*** **–** Tahapan awal dalam pengolahan citra yang bertujuan untuk memperbaiki kualitas visual citra sehingga lebih optimal untuk proses analisis selanjutnya, sehingga noise dapat dikurangi dan kontras citra dapat diperbaiki agar informasi visual yang terkandung di dalam citra menjadi lebih jelas.
2. ***Image Enhancement*** **–** Teknik yang digunakan untuk memperbaiki kualitas visual citra, seperti peningkatan kontras, penajaman tepi, dan perataan

histogram. Metode ini bertujuan untuk menonjolkan fitur-fitur penting agar lebih mudah diamati dan dianalisis.

3. ***Image Segmentation*** **–** Proses pemisahan objek-objek penting dari latar belakang citra sehingga objek tersebut dapat dianalisis lebih lanjut. Segmentasi merupakan tahap penting dalam sistem pengenalan pola dan identifikasi objek.
4. ***Image Analysis and Feature Extraction*** **–** Tahap yang dilakukan setelah proses segmentasi, di mana dilakukan identifikasi dan pengukuran ciri-ciri objek seperti bentuk, warna, dan tekstur untuk membantu sistem dalam mengenali dan membedakan objek secara lebih akurat.
5. ***Image Restoration and Reconstruction*** **–**untuk memperbaiki citra yang mengalami kerusakan atau keburaman agar mendekati kondisi aslinya, termasuk teknik untuk mengurangi distorsi akibat noise maupun efek blur..
6. ***Image Compression*** **–** Proses pengurangan ukuran data citra guna meningkatkan efisiensi penyimpanan atau transmisi data tanpa menghilangkan informasi penting yang terkandung di dalam citra.

Untuk penjelasan yang lebih rinci, dapat dilihat pada diagram atau skema umum pengolahan citra yang menggambarkan tahapan serta teknik yang digunakan dalam proses analisis. Gambar 2.4 merupakan Skema *image processing analysis*



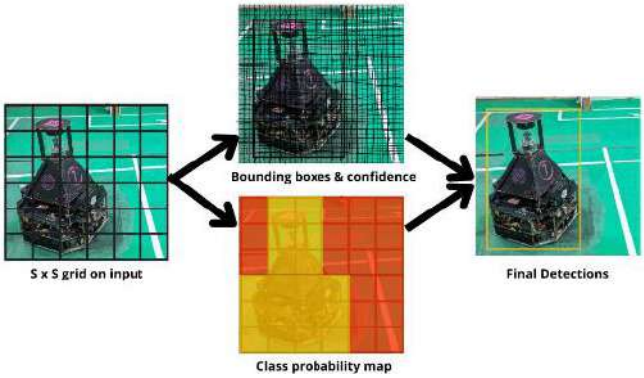
Gambar 2. 4 *Skema Image Processing Analysis*
(Mursalim & Aprilia, 2024)

2.4. *You Only Look Once* (YOLOV5S)

Pada penelitian ini, menggunakan YOLOV5S, YOLOV5S merupakan

algoritma deteksi objek yang dirancang dengan memanfaatkan arsitektur *Convolutional Neural Network* (CNN) untuk melakukan proses deteksi secara efisien dalam satu tahap. dan dirancang untuk mendukung proses pengenalan objek secara waktu nyata. Algoritma ini mampu melakukan pemrosesan secara cepat dengan kecepatan hingga 45 *frame* per detik. YOLOV5S menerapkan pendekatan deteksi menggunakan mekanisme klasifikasi dan pelokalan secara langsung pada citra input, sehingga mampu mengidentifikasi objek pada berbagai posisi dan skala dalam satu proses pemindaian (Adi Rahmad Ramadhan et al., 2025). Area citra yang mengandung objek akan menghasilkan skor tertinggi sebagai indikator keberhasilan proses deteksi objek. (Romadloni et al., 2023). YOLOV5S menawarkan peningkatan efisiensi jaringan, kemampuan deteksi multi-skala, serta optimalisasi sistem anchor, dengan arsitektur yang terdiri dari *backbone, neck,* dan *head* (Adi Rahmad Ramadhan et al., 2025).

Dengan pendekatan YOLOV5S ini memungkinkan deteksi objek dengan terpercaya dan efisien, karena menggunakan satu jaringan saraf tunggal untuk memproses keseluruhan citra. Pada prosesnya, citra dibagi ke dalam beberapa wilayah berbentuk grid, kemudian pada setiap wilayah dilakukan prediksi kotak pembatas (bounding box) beserta nilai probabilitasnya. Secara umum, gambar dibagi menjadi grid berukuran N×N, di mana setiap sel grid memprediksi bounding box, tingkat kepercayaan, serta probabilitas kelas secara simultan. Skema sistem deteksi objek menggunakan YOLOV5S ditunjukkan pada Gambar 2.5

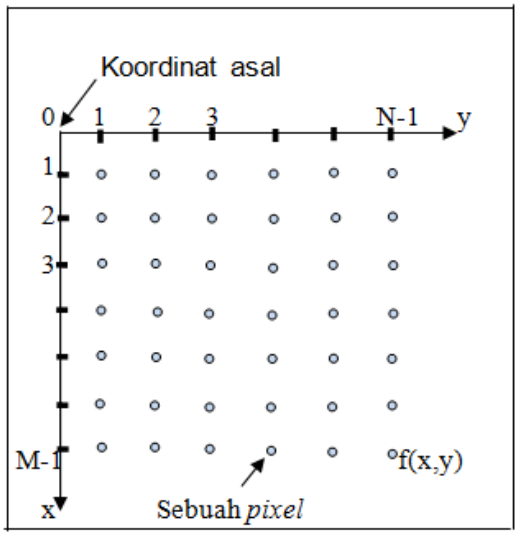


Gambar 2. 5 *Object Detection* YOLOV5S
(Ikram et al., 2024)

2.5 Piksel, Intensitas, dan Resolusi Citra

Sebuah citra tersusun atas sejumlah besar titik kecil yang disebut piksel, di

mana setiap piksel merepresentasikan nilai warna tertentu. Kombinasi dan susunan piksel-piksel tersebut membentuk pola visual yang selanjutnya menghasilkan suatu gambar yang utuh (Ramadhan et al., 2025). Setiap piksel dalam citra memiliki koordinat tertentu yang dapat direpresentasikan dalam sistem koordinat gambar. Nilai warna yang dimiliki oleh setiap piksel berpengaruh terhadap intensitas citra secara keseluruhan. Intensitas warna tersebut dapat dikelompokkan ke dalam beberapa jenis, antara lain citra dengan 256 warna, *high color*, *true color* yang memiliki hingga 16 juta variasi warna, citra skala abu-abu *(grayscale)*, serta citra hitam putih *(black and white).* Semakin tinggi jumlah variasi warna yang digunakan dalam pembentukan citra, maka semakin baik pula kualitas visual yang dihasilkan. Batas maksimum jumlah warna yang dapat ditampilkan pada suatu citra ditentukan oleh format atau jenis *file* gambar yang digunakan. Sebagai contoh, format JPG mampu menampilkan hingga 16 juta warna, sedangkan format GIF hanya mendukung hingga 256 warna (Ramadhan et al., 2025). Gambar 2.6 menunjukkan Koordinat *pixel.*



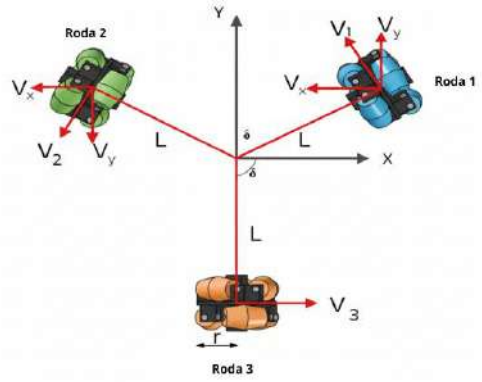
Gambar 2. 6 Koordinat pixel
(Musakkir et al., 2024)

2.6 Odometry

Odometry adalah metode yang digunakan dalam robotika dan sistem navigasi untuk memperkirakan posisi dan orientasi sebuah robot berdasarkan pengukuran pergerakan dari roda atau aktuator lainnya.

Odometri bekerja dengan memanfaatkan sensor rotasi roda *(encoder)* untuk menghitung jarak yang ditempuh oleh robot. Setiap roda robot dilengkapi dengan

encoder yang mengukur jumlah putaran atau pulsa. Dengan mengetahui diameter roda dan jumlah pulsa dari encoder, dapat dihitung jarak linear yang ditempuh roda. Representasi penempatan rotary encoder yang digunakan untuk mendukung perhitungan odometri ditunjukkan pada Gambar 2.7. Setiap putaran roda menghasilkan sejumlah pulsa yang dapat dihitung menggunakan Persamaan 2.1 dan 2.2, yang menjadi dasar utama dalam estimasi posisi robot secara terus-menerus.



Gambar 2. 7 Konfigurasi 3 *Rotary Encoder*
(Penulis, 2026)

Keliling roda menyatakan jarak tempuh roda dalam satu putaran penuh dan dihitung menggunakan persamaan berikut :

$$K_{\text{roda}} = 2\pi r \tag{2.1}$$

di mana r adalah jari-jari roda dalam satuan milimeter. Nilai keliling ini digunakan sebagai dasar untuk mengubah rotasi roda menjadi jarak linier yang dilalui oleh robot (Khumaidi et al., 2025).

Setiap putaran roda menghasilkan sejumlah pulsa yang dideteksi oleh rotary encoder. Jumlah pulsa per satuan jarak dihitung berdasarkan resolusi encoder dan keliling roda, sebagaimana ditunjukkan pada persamaan berikut.:

$$\frac{\text{Pulse}}{\text{mm}} = \frac{Res_{\text{enc}}}{K_{\text{roda}}} \tag{2.2}$$

di mana Res_{enc} adalah jumlah pulsa yang dihasilkan oleh encoder dalam satu putaran penuh *(pulse per revolution).* Rasio ini digunakan untuk mengonversi jumlah pulsa menjadi jarak tempuh aktual dalam satuan milimeter (Khumaidi et al., 2025).

Penempatan 3 *rotary encoder* untuk odometri ditunjukkan pada Gambar 2.8. Pada robot *omni-directional* dengan tiga roda, koordinat posisi (X, Y) dapat

ditentukan dengan memanfaatkan mekanisme pengaturan kecepatan masing-masing roda melalui sistem kinematika robot (Yaqin, 2025). Sistem ini mempertimbangkan gerak *omni-directional* dari ketiga roda serta jarak yang telah ditempuh oleh setiap roda. Pergerakan translasi robot pada sumbu lokal dihitung berdasarkan kontribusi jarak tempuh masing-masing roda, sehingga posisi translasi (S_x, S_y) (Khumaidi et al., 2025). Dapat dirumuskan sesuai kinematika robot *omni-directional* tiga roda sebagai berikut.

$$S_x = S_3 - S_1 \cos(\theta_1) - S_2 \cos(\theta_2) \quad (2.3)$$

$$S_y = S_1 \cos(\theta_1) - S_2 \cos(\theta_2) \quad (2.4)$$

dengan :

S_1, S_2, S_3 : jarak tempuh dari masing-masing *rotary*.

θ_1, θ_2 : sudut posisi roda terhadap sumbu referensi.

Persamaan ini digunakan untuk menghitung perubahan posisi translasi robot berdasarkan kombinasi vektor gerak dari tiap roda omni-directional (Khumaidi et al., 2025). Untuk diterapkan di sistem koordinat global lapangan, posisi robot diperoleh dengan melakukan transformasi rotasi dari sistem koordinat lokal ke sistem koordinat global menggunakan persamaan 2.5 berikut :

$$\begin{matrix} X_{pos} \\ Y_{pos} \end{matrix}] = [\begin{matrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{matrix}] \cdot [\begin{matrix} S_x \\ S_y \end{matrix}] \quad (2.5)$$

di mana θ merupakan sudut orientasi robot terhadap sumbu X global. Transformasi ini memastikan bahwa posisi robot dapat dipetakan secara akurat pada koordinat lapangan meskipun robot mengalami perubahan orientasi selama bergerak (Khumaidi et al., 2025). Odometri rentan terhadap berbagai jenis kesalahan. Kesalahan sistematis dapat disebabkan oleh perbedaan diameter roda, bentuk roda yang tidak benar-benar bulat, ketidakpastian titik kontak roda dengan permukaan, keterbatasan resolusi sensor *encoder*. Selain itu, terdapat pula kesalahan non-sistematis, yang muncul akibat kondisi permukaan lantai yang tidak rata, selip roda, atau adanya hambatan tak terduga seperti objek yang menghalangi pergerakan robot (Yaqin, 2025).

2.7 Teori *Gyroscope*

Gyroscope adalah sebuah perangkat yang berfungsi untuk mengukur atau

mempertahankan orientasi dan gerak rotasi suatu objek dengan mengandalkan sebuah roda atau cakram yang berputar pada porosnya (F. Kurniawan et al., 2021). Ketika cakram tersebut berputar, gyroscope memiliki kecenderungan untuk mempertahankan arah sumbu rotasinya akibat hukum kekekalan momentum sudut, sehingga orientasi awalnya tetap stabil selama tidak ada torsi eksternal yang signifikan. Prinsip ini memungkinkan gyroscope digunakan untuk mendeteksi perubahan orientasi dan kecepatan sudut suatu benda terhadap tiga sumbu utama, yaitu sumbu x, y, dan z, yang masing-masing merepresentasikan sudut *roll (*φ), *pitch* (θ), dan *yaw* (ψ). Dalam implementasi modern, gyroscope banyak direalisasikan dalam bentuk sensor MEMS (*Micro-Electro-Mechanical Systems*) yang memanfaatkan efek Coriolis untuk mengukur kecepatan sudut secara elektronik, sehingga dapat diaplikasikan secara luas pada sistem navigasi, robotika, dan stabilisasi gerak (Alfian et al., 2021).

Gyroscope menghasilkan keluaran berupa kecepatan sudut dalam interval waktu tertentu sebagai parameter utama pengukurannya. Nilai kecepatan sudut tersebut selanjutnya diolah dengan cara mengintegrasikannya terhadap waktu untuk memperoleh besar sudut orientasi suatu objek (Yaqin, 2025).

2.8 *Gyrodometry*

Gyrodometry merupakan metode estimasi posisi dan orientasi robot yang menggabungkan data dari odometri roda dan sensor *gyroscope*. Odometri memanfaatkan *rotary encoder* untuk menghitung jarak tempuh dan posisi robot dalam koordinat kartesian, namun metode ini rentan terhadap kesalahan akibat selip roda dan kondisi permukaan yang tidak rata (Riansyah et al., 2023). Untuk meningkatkan akurasi navigasi, data kecepatan sudut dari gyroscope diintegrasikan terhadap waktu guna menghitung sudut orientasi robot sebagai koreksi arah gerak. Kombinasi data translasi dari odometri dan orientasi sudut dari gyroscope terbukti mampu meningkatkan stabilitas dan akurasi estimasi pose robot pada sistem navigasi robot mobile (Leadtree & Needs, 2025).

Dari sisi komputasi, metode *gyrodometry* memisahkan proses estimasi antara posisi translasi dan orientasi arah hadap robot. Odometri digunakan untuk merekam jumlah putaran roda yang kemudian dikonversi menjadi perubahan jarak

rata-rata (Δs), sedangkan sensor *gyroscope* mengukur laju perubahan sudut (ω) yang selanjutnya diintegrasikan terhadap waktu (dt) untuk memperoleh sudut orientasi (θ) secara berkelanjutan. Pada setiap interval waktu, posisi robot diperbarui dengan cara memproyeksikan nilai jarak tempuh Δs ke dalam sistem koordinat global berdasarkan sudut orientasi yang diperoleh dari *gyroscope* (Sutrisno et al., 2024) . Metode *gyrodometry* dapat dirumuskan sebagai berikut :

1. Nilai kecepatan sudut tersebut diintegrasikan terhadap waktu sehingga menghasilkan sudut orientasi robot pada waktu ke-t, yang dirumuskan pada persamaan 2.6 sebagai berikut:

   $$\theta_t = \theta_0 + \int w \times dt \quad (2.6)$$

2. Perhitungan posisi robot pada sumbu x dan y dirumuskan pada persamaan 2.7 dan 2.8 :

   $$x_t = x_{t-1} + \Delta_s \times \cos(\theta_t) \quad (2.7)$$

   $$y_t = y_{t-1} + \Delta_s \times \mathrm{si\,n}(\theta_t) \quad (2.8)$$

   dengan keterangan sebagai berikut:

θ_t	: Sudut orientasi robot saat ini dari *Gyroscope.*
w	: Kecepatan sudut *(Angular Velocity).*
x_t, y_t	: Posisi robot saat ini.
x_{t-1}, y_{t-1}	: Posisi robot sebelumnya.
Δs	: Perubahan jarak tempuh rata – rata dari *Encoder*.

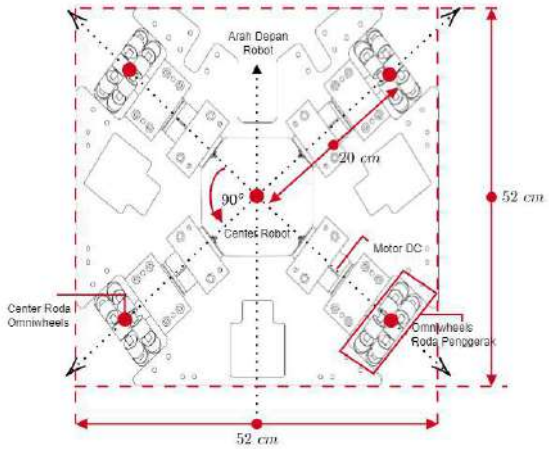
Penerapan persamaan tersebut bertujuan untuk menjaga akurasi perhitungan posisi robot meskipun terjadi gangguan pada sistem penggerak roda. Pada kondisi tertentu, seperti saat robot mengalami selip roda akibat permukaan yang tidak rata, pembacaan data dari *rotary encoder* dapat menyebabkan kesalahan dalam penentuan arah gerak. Namun, karena nilai sudut orientasi (θ) diperoleh dan dipertahankan oleh sensor gyroscope yang relatif tidak terpengaruh oleh gesekan permukaan, sistem navigasi mampu melakukan koreksi terhadap vektor pergerakan robot. Dengan demikian, robot tetap dapat mempertahankan lintasan geraknya dan tidak menyimpang dari jalur yang telah ditentukan di lapangan (Sutrisno et al., 2024).

2.9 Mekanisme Penggerak *Omni-Wheel* Empat Arah

Omni-wheels merupakan jenis roda khusus yang dirancang untuk

memungkinkan robot bergerak secara *omnidirectional,* yaitu mampu bergerak ke segala arah pada bidang datar tanpa harus mengubah orientasi tubuh robot terlebih dahulu. Berbeda dengan roda konvensional, *omni-wheels* memiliki rol-rol kecil yang dipasang di sekeliling keliling roda utama, sehingga roda tidak hanya menghasilkan gaya dorong searah putaran, tetapi juga memungkinkan terjadinya gerakan lateral. Pada sistem *Four Omni-Wheels Drive*, robot menggunakan empat buah omni-wheels yang dipasang pada setiap sisi rangka robot dengan konfigurasi tertentu (umumnya bersudut 90° atau 45°). Konfigurasi ini memungkinkan robot untuk bergerak maju, mundur, ke samping kanan-kiri, diagonal, serta berotasi di tempat hanya dengan mengatur kecepatan dan arah putaran masing-masing motor roda (Rijalusalam & Iswanto, 2021).

Prinsip kerja omni-wheels didasarkan pada kombinasi gaya dorong dari masing-masing roda yang dipengaruhi oleh arah dan kecepatan putaran motor. Rol pada omni-wheels memungkinkan roda bergerak bebas ke arah samping, sehingga resultan kecepatan robot dapat dibentuk pada sumbu X, Y, serta rotasi terhadap sumbu Z. Dengan mengatur kecepatan tiap roda, robot dapat bergerak maju, menyamping, diagonal, maupun berotasi di tempat secara halus dan kontinu (Rijalusalam & Iswanto, 2021). Berikut Gambar 2.8 yang merupakan konfirugasi dan penempatan 4 posisi motor beserta sudutnya :



Gambar 2. 8 *Four Omni-Wheel Drive*
(Salsabilla et al., 2024)

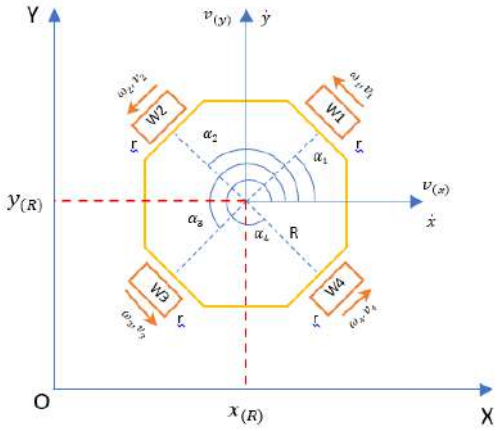
Untuk memahami perpindahan robot dari posisi awal secara menyeluruh, diperlukan pemahaman terhadap keterkaitan antar variabel yang terlibat, sehingga pengendalian pergerakan robot serta pengaturan sudut dan kecepatan pada setiap

roda dapat dilakukan secara optimal dan terkoordinas (Yaqin, 2025). Dapat dilihat pada gambar 2.9, untuk mendapatkan persamaan kinematika pengontrolan sebagai berikut :

$$V_x = -V_1 \cos(\theta_1) - V_2 \cos(\theta_2) + V_3 \cos(\theta_3) + V_4 \cos(\theta_4) \quad (2.9)$$

$$V_y = V_1 \cos(\theta_1) - V_2 \cos(\theta_2) - V_3 \cos(\theta_3) + V_4 \cos(\theta_4) \quad (2.10)$$

$$V_\theta = \frac{V_1}{R} + \frac{V_2}{R} + \frac{V_3}{R} + \frac{V_4}{R} \quad (2.11)$$



Gambar 2. 9 Kinematika *Four Omni-Wheel*
(Rijalusalam & Iswanto, 2021)

2.10 *Path Planning*

Path planning merupakan proses perencanaan lintasan yang menentukan jalur terbaik dari posisi awal menuju tujuan bagi robot, dengan mempertimbangkan kondisi lingkungan dan kehadiran rintangan. Tujuan utama dari path planning adalah menghasilkan jalur yang aman, efisien, dan feasible, sehingga robot dapat bergerak dari titik awal ke titik tujuan tanpa bertabrakan dengan objek lain di sekitarnya. Dalam sistem navigasi robot, *path planning* menjadi bagian krusial karena lintasan yang dihasilkan akan menjadi dasar bagi robot untuk mengeksekusi pergerakan sesuai tujuan yang telah ditetapkan dalam lingkungan operasi (Wahid Hasyim et al., 2025).

Dalam sistem navigasi robot bergerak, *path planning* berfungsi sebagai tahap awal sebelum proses pengendalian gerak dilakukan. Hasil dari path planning berupa lintasan atau serangkaian titik tujuan (*waypoint*) yang menjadi referensi bagi sistem kontrol untuk menentukan arah, kecepatan, dan orientasi pergerakan robot. Oleh karena itu, kualitas path planning sangat mempengaruhi ketepatan dan efisiensi navigasi robot secara keseluruhan.

Secara umum, path planning dapat diklasifikasikan menjadi dua jenis, yaitu *global path planning* dan *local path planning*. Global path planning dilakukan dengan memanfaatkan informasi lingkungan secara menyeluruh untuk menghasilkan lintasan dari titik awal menuju tujuan berdasarkan kondisi lingkungan yang diketahui. Sementara itu, *local path planning* bekerja secara *real-time* dengan mempertimbangkan kondisi lingkungan di sekitar robot, terutama untuk menghindari rintangan yang bersifat dinamis (Salam, 2023).

Proses ini melibatkan penerapan algoritma pengambilan keputusan tingkat tinggi yang mampu beradaptasi terhadap perubahan kondisi serta merespons kejadian yang tidak terduga, seperti penurunan daya baterai atau adanya informasi baru mengenai lingkungan. Dalam perencanaan lintasan, digunakan perencana yang diperluas dengan memanfaatkan peta berbasis kisi sel yang dibangun dari peta *global, local costmap,* dan *global costmap*. Secara khusus, algoritma A* menentukan jalur dengan meminimalkan fungsi evaluasi (Salam, 2023).

$$f(n) = g(n) + h(n) \tag{2.12}$$

Pada persamaan 2.12, di mana n merupakan simpul berikutnya pada lintasan, $g(n)$ menyatakan nilai jalur dari simpul awal menuju n, dan $h(n)$ merupakan fungsi heuristik yang mengestimasi nilai minimum dari n ke tujuan. Algoritma A* menjamin diperolehnya lintasan dengan nilai evaluasi total terendah dari titik awal hingga tujuan (Salam, 2023).

2.11 Sensor dan Aktuator

Dalam penelitian ini membutuhkan beberapa sensor dan aktuator untuk mendukung jalannya sistem, antara lain :

2.11.1 Kamera *Omni-Directional* 360

Kamera utama yang digunakan pada robot sepak bola beroda merupakan sistem kamera omnidirectional. Perangkat yang digunakan adalah kamera ELP USB8MP02G-SFV yang dikombinasikan dengan cermin tele Vstone tipe VS-C450MR, sebagaimana ditampilkan pada Gambar 2.10. Konfigurasi ini memanfaatkan cermin hiperbolik sehingga sistem kamera mampu menangkap citra dengan cakupan sudut pandang 360° pada bidang horizontal. Selain itu, kamera memiliki bidang pandang vertikal sekitar 10–15° pada sisi atas dan hingga 55° pada

sisi bawah, yang tersusun sejajar dengan sumbu cermin untuk mendukung pengamatan lingkungan di sekitar robot secara menyeluruh (Ramadhan et al., 2025).

Gambar 2. 10 Kamera Omni 360
(Penulis, 2026).

Sistem kamera omnidirectional tersusun atas beberapa komponen utama, yaitu cermin, adaptor lensa tele, dan kamera CCD sebagai sensor citra. Keunggulan utama dari sistem ini terletak pada kemampuannya dalam memperoleh bidang pandang yang jauh lebih luas dibandingkan kamera konvensional. Dengan karakteristik tersebut, kamera omnidirectional sangat efektif digunakan sebagai sensor utama pada robot yang membutuhkan kemampuan deteksi objek dan pemantauan lingkungan dari berbagai arah secara simultan (Irwansyah et al., 2023).

2.11.2 *Incremental Rotary Encoder*

Penentuan koordinat posisi $x$dan y dilakukan dengan memanfaatkan *rotary encoder* sebagai sensor umpan balik gerak. Rotary encoder merupakan perangkat elektronik yang berfungsi mengonversi pergerakan sudut atau rotasi poros menjadi sinyal listrik, baik dalam bentuk analog maupun digital. Berdasarkan karakteristik sinyal keluarannya, rotary encoder dikelompokkan menjadi dua jenis utama, yaitu *rotary encoder inkremental* dan *rotary encoder* absolut (Bayu et al., 2024).

Dalam penelitian ini, digunakan incremental rotary encoder sebagai sensor umpan balik gerak, yang menghasilkan sinyal berupa pulsa gelombang kotak setiap kali poros berputar. Pulsa tersebut menunjukkan perubahan posisi relatif terhadap titik awal, sehingga sistem mampu menghitung perpindahan robot secara akurat pada koordinat dua dimensi. *Encoder* tipe ini dipilih karena konstruksinya yang

sederhana, Tabel 2.2 merupakan spesfikasi dari rotary yang digunakan.

Tabel 2. 2 Spesifikasi *Rotary Encoder Incremental.*

Atrribute	*Value*
Pulse per Revolution (PPR)	800
Encoder Technology	*Incremental*
Maximum Revolution	5000 rpm
Output Signal Type	*NPN Open Collector*
Shaft Type	*Radial, Thrust*
Supply Voltage	5-24 VDC
Operating Temperature	40 °C - 85 °C
Overall Height	51 mm
Current Consumption	30 mA
Electrical Response Frequency	20 kHz

Gambar 2. 11 ***Rotary Encoder***
(Penulis, 2026)

sebagaimana *rotary encoder* ditunjukkan pada Gambar 2.11, digunakan untuk mengukur perpindahan posisi robot melalui pembacaan rotasi roda *omni-directional* yang terpasang pada poros encoder. Informasi rotasi tersebut selanjutnya dikonversi menjadi data perpindahan, sehingga sistem dapat menentukan perubahan posisi robot secara akurat.

2.11.3 Sensor HWT101CT

Dalam penelitian ini, untuk menentukan arah robot atau sudut robot menggunakan Sensor HWT101CT-TTL seperti pada gambar 2.12, yaitu sensor kemiringan satu sumbu yang memanfaatkan teknologi MEMS *(Microelectromechanical System).* Sensor ini mampu mengukur sudut kemiringan hingga 180° dengan resolusi 0,01°, sehingga memungkinkan pembacaan posisi yang sangat presisi. Selain itu, sensor ini telah dilengkapi dengan giroskop internal,

yang memungkinkan pengukuran perubahan sudut secara *real-time*, sehingga data orientasi dapat diperoleh secara kontinu dan akurat. Dengan karakteristik ini, sensor HWT101CT-TTL cocok digunakan sebagai sensor sudut pada sistem robotik atau platform bergerak untuk memantau orientasi dan kemiringan secara *real-time*, termasuk dalam sistem navigasi, stabilisasi, dan kontrol gerak robot (Kaladewa & Santoso, 2022). Spesifikasi lengkap ditunjukkan pada tabel 2.3.

Gambar 2. 12 Sensor HWT101CT
(WiMotion, 2024)

Tabel 2. 3 Spesifikasi Sensor HWT101CT

Atribute	*Value*
Voltage	9-36 V
Current	<25 mA
Range	Z : 180º
Stability	0.01^0
Accuracy	0.1^0
Return Rate	0.1 – 1000 Hz (default 10 Hz)
Baud Rate	4800 – 921600 (default 9600)

2.11.4 Sensor *Proximity*

Pada penelitian ini, sensor proximity berfungsi untuk mendeteksi bola apakah bola sudah dalam jangkauan penggiring atau belum.

Gambar 2. 13 Sensor *Proximity*
(Penulis, 2026)

Sensor *Proximity* adalah perangkat elektronik yang mendeteksi objek di dekatnya tanpa kontak fisik, dengan memancarkan dan mengukur perubahan medan energi seperti elektromagnetik, ultrasonik, atau cahaya inframerah. Sensor *Proximity* terbagi menjadi beberapa jenis, yaitu induktif, kapasitif, ultrasonik, dan optik, seperti yang terlihat pada gambar 2.13 (ade agung Kurniawan et al., 2024). Sensor proximity inframerah digunakan dalam penelitian ini untuk mengidentifikasi kondisi bola apakah telah berada pada penggiring robot, yang selanjutnya digunakan sebagai dasar pengambilan keputusan dalam melanjutkan strategi permainan. Spesifikasi sensor *proximity* disajikan pada Tabel 2.4.

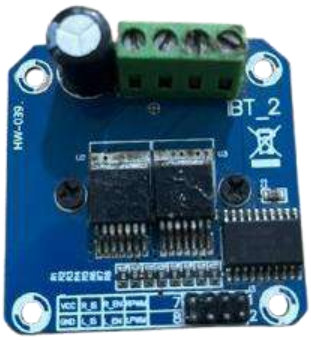
Tabel 2. 4 Spesifikasi Sensor *Proximity*

Parameter	Spesifikasi
Jarak Deteksi	3 – 20 cm
Tegangan	5 VDC
Arus	< 50 mA
Resolusi	Mm
Waktu Respons	< 1 ms
Interface	Konektor

2.11.5 Motor *Driver H-Bridge* IBT – 2 BTS7960

Pada penelitian ini, motor DC dikendalikan menggunakan *driver* H-Bridge IBT-2, sebuah modul *driver* motor yang dilengkapi dengan dua chip BTS7960 dan dirancang untuk menangani arus tinggi. Seperti ditunjukkan pada Gambar 2.14, driver ini mampu mengendalikan motor dengan tegangan 6 hingga 27 VDC. Pengendalian motor dilakukan menggunakan metode *Pulse Width Modulation* (PWM). Driver IBT-2 H-Bridge digunakan untuk mengoperasikan motor PG45 sebagai penggerak robot sekaligus pengiring bola (Darmawan et al., 2023b). Untuk

tabel 2.5 merupakan spesifikasi dari *driver* motor BTS-7960.



Gambar 2. 14 Driver Motor BTS-7960
(Penulis, 2026)

Tabel 2. 5 Spesifikasi BTS-7960

Karakteristik	Keterangan
Tegangan input	6 – 27 V
Arus Maks.	43 A
Input Level	3.3 – 5 V
Control Mode	PWM atau Level

2.11.6 Motor PG - 45

Motor DC PG45 merupakan motor arus searah (*DC gear motor*) yang dilengkapi dengan *gearbox* planetar untuk menghasilkan torsi besar pada kecepatan putar rendah, sehingga sesuai untuk aplikasi sebagai penggerak robot mobile. Motor ini memiliki kemampuan torsi yang relatif besar dengan kecepatan rendah yang sesuai untuk aplikasi penggerak robot, termasuk robot sepak bola beroda, di mana torsi dan stabilitas gerak menjadi faktor krusial (Pambudi et al., 2021). Gambar 2.15 menunjukkan motor PG-45 dan pada tabel 2.6 merupakan spesifikasi dari motor PG-45.

Gambar 2. 15 Motor DC PG-45
(Penulis, 2026)

Tabel 2. 6 Spesifikasi Motor DC PG-45

Spesifikasi	Keterangan
Built in	*Planetary Gearbox* PG45
Tegangan	DC 24V
Arus	4 A
Daya	60 W
Kecepatan	500 rpm
Torsi	25 kgf-cm

2.11.7 Motor PG - 36

Motor DC tipe PG36 adalah motor listrik arus searah (*Direct Current*) yang dilengkapi dengan *gearbox* planetar berdiameter 36 mm seperti pada Gambar 2.16, yang dirancang untuk menghasilkan torsi tinggi pada kecepatan putar yang rendah hingga sedang dibandingkan dengan motor PG - 45. Konstruk *gearbox* planetar pada motor ini memungkinkan motor menghasilkan gaya dorong yang kuat sekaligus stabil, sehingga sering dijadikan pilihan aktuator pada robot-robot mobile yang memerlukan kemampuan navigasi dan pengendalian gerak yang baik, seperti robot beroda dengan *mecanum wheels*. Adapter *gearbox* juga membuat output kecepatan motor lebih sesuai dengan kebutuhan aplikasi robotika, di mana kecepatan tidak selalu tinggi tetapi torsi cukup besar untuk melaju dan membawa beban tertentu (Yuniawan et al., 2021). Untuk spesfikasi motor PG-36 dapat dilihat pada tabel 2.7.

Gambar 2. 16 Motor DC PG-36
(Penulis, 2026)

Tabel 2. 7 Spesifikasi Motor DC PG-36

Spesifikasi	Keterangan
Built in	*Planetary Gearbox* PG45
Tegangan	DC 24V
Arus	5 A
Kecepatan	600 rpm
Torsi	10 kgf-cm

2.11.8 Solenoid *Kicker*

Solenoid *kicker* adalah komponen yang memanfaatkan prinsip elektromagnetik untuk menendang bola, seperti terlihat pada Gambar 2.17. Solenoid merupakan kumparan kawat yang menghasilkan medan magnet saat dialiri arus listrik. Pada sistem kicker, solenoid biasanya dipasang di bagian depan tengah robot dan terhubung dengan mekanisme pemukul. Saat arus listrik mengalir, medan magnet yang terbentuk menggerakkan plunger (piston) di dalam solenoid, sehingga memungkinkan robot menendang bola dengan kekuatan yang cukup besar (As'ari et al., 2024).

Gambar 2. 17 Solenoid Coil
(Penulis, 2026)

Gerakan solenoid digunakan untuk mendorong bola, dengan bagian penendang dan penggiring terpasang di basis tengah robot. Proses penendangan dilakukan dengan pemberian tegangan sebesar 450 V pada solenoid (Ramadhan et al., 2025).

2.11.9 Solenoid *Driver*

Driver *solenoid* adalah rangkaian yang dirancang untuk mengendalikan solenoid penendang menggunakan aktuator agar bola dapat ditendang. Arduino Mega 2560 digunakan sebagai mikrokontroler yang menyediakan logika untuk mengisi kapasitor atau melepaskan energi dari kapasitor ke kumparan solenoid. Saat kapasitor sedang diisi, sinyal dari Arduino mengaktifkan *optocoupler* dan *driver* motor sehingga modul pengisian kapasitor dapat bekerja, dan proses ini secara otomatis berhenti ketika kapasitor sudah penuh. Ketika Arduino memberikan sinyal untuk menendang, optocoupler diaktifkan sehingga membuka gerbang pada IGBT, menyalurkan daya dari kapasitor ke kumparan dalam hitungan milidetik. Hal

ini membuat kumparan menghasilkan medan magnet sementara yang menarik plunger besi (Ramadhan et al., 2025). Gambar 2.18 merupakan gambar driver dan booster untuk solenoid.

Gambar 2. 18 *Booster* dan *Driver Solenoid Kicker*
(Penulis, 2026)

2.11.10 LCD 20x4

Pada penelitian ini, LCD 20x4 digunakan untuk menampilkan berbagai informasi terkait robot, termasuk koordinat x dan y, sudut orientasi robot, jarak terhadap objek, serta data lainnya. LCD (*Liquid Crystal Display*) 20x4 adalah komponen elektronik yang berfungsi untuk menampilkan karakter, baik huruf maupun angka, dengan kemampuan menampilkan 20 karakter secara horizontal (kolom) dan 4 baris secara vertikal (Yaqin, 2025). Sebagaimana ditunjukkan pada Gambar 2.19.

Gambar 2. 19 Tampilan LCD 20x4
(Penulis, 2026)

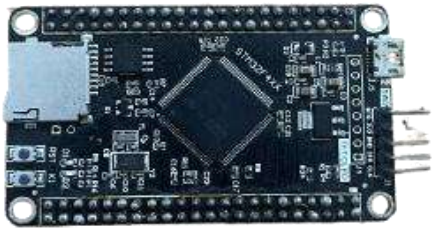
2.12 *Controller*

Pada penelitian ini, kontroler yang digunakan berperan dalam mengatur dan

memantau berbagai parameter operasional robot. Dengan adanya kontroler ini, setiap variabel penting dapat dikendalikan secara efektif, sehingga memastikan data yang diperoleh dari sistem robot tetap akurat dan konsisten selama pengujian.

2.12.1 STM32F4VGTx ARM

STM32F4VGTx ARM adalah mikrokontroler berbasis ARM Cortex-M4 yang dikembangkan oleh *STMicroelectronics,* dapat dilihat pada gambar 2.20. Mikrokontroler ini menawarkan performa tinggi dengan kecepatan hingga 168 MHz, dilengkapi unit pemrosesan sinyal digital (DSP) dan *Floating Point Unit* (FPU) untuk mendukung perhitungan *real-time*. STM32F4VGTx memiliki memori *Flash* hingga 1 MB dan RAM hingga 192 KB, serta berbagai periferal seperti ADC, DAC, UART, SPI, I2C, PWM, dan timer canggih yang memungkinkan pengendalian motor dan komunikasi dengan sensor secara efisien dan juga mikrokontroler ini memiliki pin sejumlah 82 pin *input/output*. Dengan konsumsi daya rendah dan kemampuan mode *sleep*, mikrokontroler ini ideal untuk aplikasi robotika, pengolahan sensor, dan kontrol aktuator, sehingga sangat cocok digunakan sebagai kontroler utama pada robot yang memerlukan pengolahan data dan respons cepat. Spesifikasi lengkap dapat dilihat pada tabel 2.8.



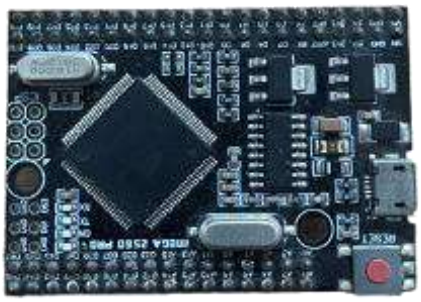
Gambar 2. 20 STM32F4VGTx
(Penulis, 2026)

Tabel 2. 8 Spesifikasi STM32F4

Atribute	*Value*
Core	ARM Cortex M4
Speed	168 mHz
Flash	1 MB
Internal RAM	192 Kb
Timer	2 x WDG, RTC, 24-bit *down counter*
Backup RAM	4 Kb
Sources	3.3 VDC

2.12.2 Arduino Mega 2560 Pro Mini

ATmega 2560 adalah mikrokontroler berbasis arsitektur AVR yang dikembangkan oleh Atmel. Mikrokontroler ini dilengkapi dengan memori flash 256 KB, RAM 8 KB, dan EEPROM 4 KB, serta memiliki 86 pin input/output yang mendukung koneksi dengan berbagai perangkat eksternal. ATmega 2560 juga menawarkan fitur seperti timer 16-bit, PWM, ADC, serta komunikasi serial melalui USART, SPI, dan I2C, sehingga cocok untuk aplikasi yang memerlukan pengolahan data kompleks dan banyak koneksi periferal (Darmawan et al., 2023b). Mikrokontroler ini banyak digunakan dalam proyek robotika dan otomasi, dengan gambar chip ditunjukkan pada Gambar 2.21 dan spesifikasinya tercantum pada Tabel 2.9.



Gambar 2. 21 Arduino Mega 2560 Pro Mini
(Penulis, 2026)

Tabel 2. 9 Spesifikasi Arduino Mega 2560 Pro Mini

Atribute	*Value*
Operating Voltage	5V
Digital I/O pin	54 (*of which* 14 *provide* PWM output)
Analog Input pin	16 pin
Flash Memory	256KB *of which* 8KB *used by bootloader*
SRAM	8KB
EEPROM	4KB
Clock Speed	16MHz

2.12.3 Laptop / *Device*

Laptop merupakan perangkat komputer portabel yang memiliki fungsi serupa dengan komputer desktop, namun dirancang lebih ringkas, ringan, dan mudah dibawa. Laptop tersedia dalam berbagai spesifikasi sesuai kebutuhan pengguna. Pada Tugas Akhir ini, peneliti menggunakan HP VICTUS 15 (Gambar

2.22) untuk melakukan pengolahan citra yang diterima dari kamera melalui koneksi USB. Hasil pengolahan citra kemudian dikirim ke mikrokontroler ARM STM32F407VGTx menggunakan komunikasi serial untuk pengendalian sistem robot.

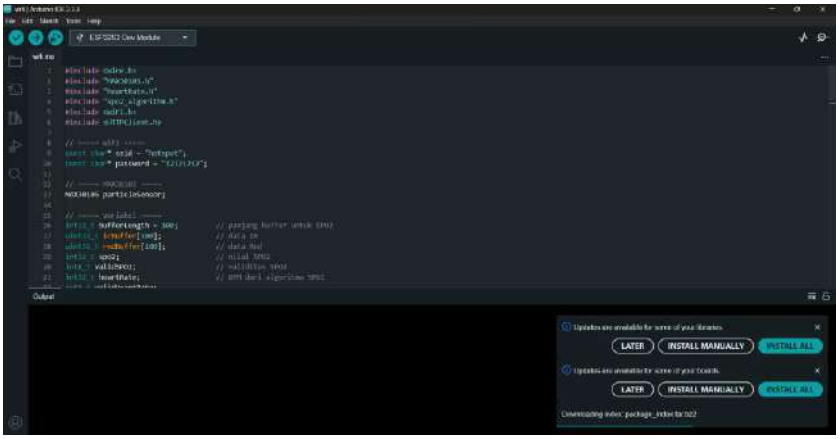
Gambar 2. 22 Laptop HP Victus 15
(Penulis, 2026)

2.13 *Software*

Dalam penelitian tugas akhir ini, peneliti berusaha memanfaatkan sejumlah perangkat lunak yang berfungsi sebagai sarana pendukung, antara lain Arduino IDE, STM32CubeMX, Keil uVision, dan *Visual Studio Code*.

2.13.1 Arduino IDE

Arduino IDE merupakan lingkungan pengembangan terintegrasi yang digunakan untuk membuat, mengedit, dan mengunggah kode program ke papan mikrokontroler Arduino. Aplikasi ini menyediakan antarmuka yang sederhana namun lengkap, sehingga memudahkan mahasiswa maupun peneliti dalam mengembangkan sistem berbasis mikrokontroler. Arduino IDE mendukung penggunaan bahasa pemrograman C/C++ serta menyediakan beragam pustaka (library) yang memudahkan proses komunikasi dan pengendalian berbagai perangkat keras, seperti sensor, aktuator, dan modul elektronik lainnya (Rohman et al., 2021). Pada penelitian ini, Arduino IDE digunakan sebagai perangkat lunak untuk mengembangkan algoritma pemrograman, termasuk pembacaan data dari sensor gyroscope dan pengendalian mekanisme penendang bola pada robot, sebagaimana diperlihatkan pada Gambar 2.23.

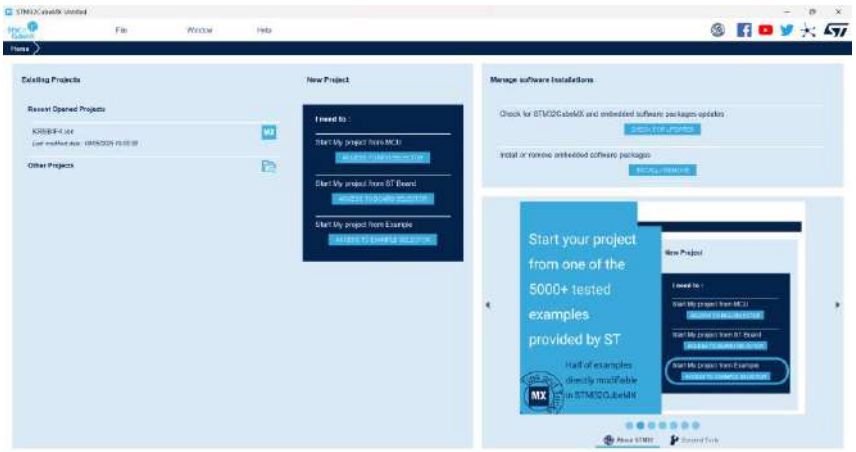


Gambar 2. 23 Tampilan Arduino IDE
(Penulis, 2026)

2.13.2 STM32-*Cube* MX

STM32CubeMX adalah perangkat lunak dari STMicroelectronics yang digunakan untuk mempermudah konfigurasi dan inisialisasi mikrokontroler STM32 melalui antarmuka grafis (GUI) yang intuitif. Software ini memungkinkan pengguna mengatur pin, mengelola peripheral, dan menghasilkan kode awal secara otomatis, sehingga mempercepat pengembangan firmware. STM32CubeMX sering digunakan bersamaan dengan Keil uVision atau STM32CubeIDE untuk membangun sistem embedded pada bidang robotika, otomasi, dan IoT.

Pada Tugas Akhir ini, perangkat lunak STM32CubeMX, seperti yang ditampilkan pada Gambar 2.24, digunakan untuk melakukan konfigurasi seluruh pin pada ARM Development Board STM32F407VGT6, sehingga setiap pin dapat berfungsi sesuai dengan perannya (Yaqin, 2025).

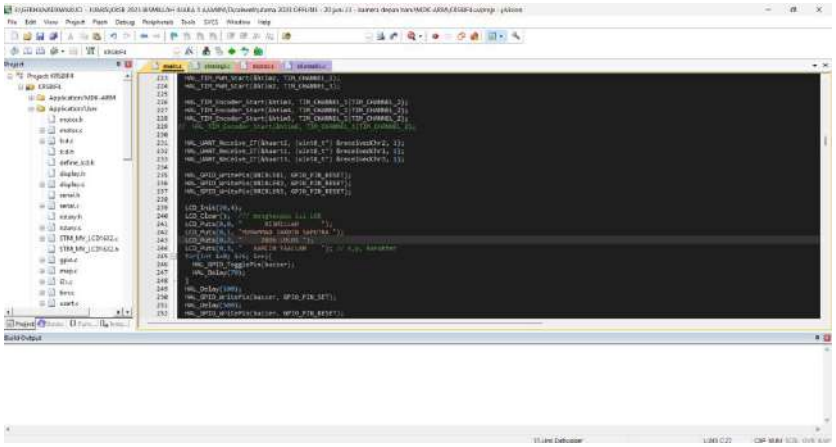


Gambar 2. 24 Tampilan STM32-CubeMX
(Penulis, 2026)

2.13.3 Keil u-Vision

Keil uVision adalah sebuah *Integrated Development Environment* (IDE) yang dirancang untuk pengembangan perangkat lunak sistem *embedded*, terutama pada mikrokontroler berbasis ARM dan arsitektur populer lainnya. IDE ini menyediakan lingkungan terpadu yang mencakup *editor* kode, *compiler*, *debugger*, dan simulator, sehingga memudahkan penulisan, kompilasi, serta debugging program sebelum diunggah ke mikrokontroler. Keil uVision dikenal karena antarmukanya yang ramah pengguna serta kemampuan lengkap dalam menangani berbagai proses pengembangan aplikasi embedded. Penggunaan Keil uVision juga tercatat dalam berbagai penelitian tugas akhir sebagai alat utama dalam perancangan dan pengembangan firmware pada platform ARM seperti STM32, karena kemampuannya dalam mengelola proyek, menyediakan *library* perangkat, dan mendukung simulasi serta debugging secara langsung (Zhu et al., 2025).

Pada Tugas Akhir ini**,** Keil uVision, seperti terlihat pada Gambar 2.25, digunakan untuk merancang algoritma kinematika gerak robot serta memproses data yang dikirim dari laptop melalui komunikasi serial. Pemakaian Keil mempermudah pengujian dan penyempurnaan program secara langsung pada perangkat keras (Yaqin, 2025).

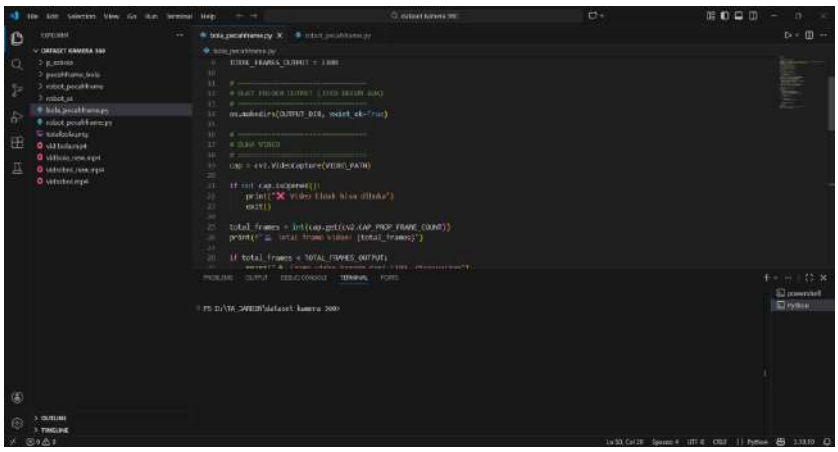


Gambar 2. 25 Tampilan Keil u-Vision
(Penulis, 2026)

2.13.4 *Visual Studio Code*

Visual Studio Code adalah editor kode ringan dengan fitur lengkap yang dapat digunakan pada sistem operasi Windows, macOS, dan Linux. Perangkat lunak ini memiliki dukungan bawaan terhadap bahasa pemrograman seperti

JavaScript, TypeScript, dan Node.js, serta dapat diperluas untuk mendukung bahasa lain, antara lain C++, C#, Java, Python, PHP, dan Go, termasuk berbagai runtime pengembangan seperti .NET dan Unity. Visual Studio Code juga menyediakan kemudahan melalui integrasi dengan sistem kontrol versi seperti Git dan layanan *Source Control Management* (SCM) lainnya. Selain itu, editor ini didukung oleh Microsoft Azure yang memungkinkan proses penyebaran, hosting, serta penyimpanan aplikasi dengan dukungan terhadap berbagai *framework*, antara lain *React, Angular, Vue, Node.js*, dan Python (Yaqin, 2025). Pada program akhir ini, Visual Studio Code digunakan sebagai editor teks, sementara pengolahan citra dilakukan dengan menggunakan bahasa pemrograman C++ dan Python (Ramadhan et al., 2025). Pada Gambar 2.26 merupakan tampilan dari VsCode.



Gambar 2. 26 Tampilan VSCode
(Penulis, 2026)

2.13.5 Python

Python adalah bahasa pemrograman interpretatif yang mendukung berbagai paradigma dan dapat dijalankan secara langsung pada berbagai platform, termasuk Windows, Linux, dan macOS. Dengan sintaks yang ringkas namun ekspresif, Python memadukan paradigma prosedural, berorientasi objek, dan fungsional, sehingga memberikan fleksibilitas tinggi dalam pengembangan aplikasi kompleks. Keunggulan lainnya adalah ekosistem pustaka (*libraries*) yang luas, mendukung beragam bidang mulai dari pengolahan data, kecerdasan buatan, hingga pengembangan sistem embedded, menjadikannya salah satu bahasa pemrograman favorit dalam penelitian dan industri (Rahman et al., 2023). Gambar 2.27 menunjukkan logo dari python.

Gambar 2. 27 Logo Python
(Python, 2022)

Karena fleksibilitasnya yang luas dalam berbagai bidang, seperti pengembangan aplikasi web, aplikasi desktop, dan *Internet of Things* (IoT), Python semakin banyak digunakan oleh programmer dan peneliti sebagai bahasa pemrograman pilihan. Popularitas Python di kalangan peneliti disebabkan kemampuannya dalam mengelola data dalam skala besar serta melakukan perhitungan matematika yang kompleks. Selain itu, kemampuan Python untuk membaca, memproses, dan memodifikasi berbagai jenis file menjadikannya sangat cocok untuk prototyping dan pengembangan perangkat lunak secara cepat dan handal (Rahman et al., 2023).

2.13.6 PyTorch

PyTorch adalah pustaka (*library*) open-source berbasis Python yang digunakan untuk *machine learning* dan *deep learning*. Dikembangkan oleh Facebook's AI Research (FAIR), PyTorch menyediakan *tensor computation* yang fleksibel serta kemampuan *autograd* untuk melakukan *automatic differentiation*, sehingga memudahkan pelatihan model neural network. PyTorch mendukung pembangunan dan pelatihan model secara dinamis (*dynamic computation graph*), yang memudahkan eksperimen dan debugging dibandingkan pustaka lain yang bersifat statis. PyTorch banyak digunakan dalam penelitian kecerdasan buatan, pengolahan citra (*computer vision*), pemrosesan bahasa alami (*natural language processing*), dan aplikasi deep learning lainnya karena kemudahan penggunaan, fleksibilitas, dan integrasi yang baik dengan ekosistem Python (Rozemberczki et al., 2021). Gambar 2.28 menunjukkan logo pytorch.

Gambar 2. 28 Logo Pytorch
(Pytorch, 2023)

Tensor sebagai struktur data utama dalam PyTorch, merupakan *array* multidimensi yang memiliki kemiripan dengan *array* pada pustaka *NumPy*. Oleh karena itu, berbagai fitur dikembangkan untuk mendukung perancangan dan pelatihan arsitektur jaringan saraf yang baru, serta untuk mempercepat proses pengembangan dan eksekusi proyek dengan memanfaatkan perangkat lunak dan perangkat keras terkini. Penggunaan tensor memungkinkan peningkatan kecepatan operasi matematika. Selain itu, PyTorch menyediakan fasilitas untuk pelatihan terdistribusi, pengelolaan proses paralel guna memaksimalkan pemanfaatan sumber daya dalam satu mesin, serta berbagai utilitas tambahan yang mendukung fungsi-fungsi umum dalam *Deep Learning* (Hendri et al., 2021).

2.13.7 *Open CV*

Open Computer Vision berfungsi sebagai pustaka yang berkaitan dengan pengolahan citra digital, yang memungkinkan proses akuisisi, pembentukan, serta manipulasi citra agar dapat dimanfaatkan dalam pemrograman, seperti pada proses deteksi objek. Adapun visi komputer merupakan cabang ilmu dari pengolahan citra yang bertujuan untuk memberikan kemampuan kepada komputer dalam menginterpretasikan dan memahami informasi visual, menyerupai cara kerja penglihatan pada manusia atau makhluk hidup (Romadloni et al., 2023). Gambar 2.29 menunjukkan logo dari OpenCV.

Gambar 2. 29 Logo OpenCV
(OpenCV, 2022)

OpenCV merupakan pustaka bersifat open source yang dirancang untuk mendukung pengolahan citra dan visi komputer secara waktu nyata. Pustaka ini

menyediakan beragam fungsi serta algoritma untuk pemrosesan gambar dan video, meliputi deteksi objek, segmentasi citra, pelacakan objek *(tracking),* hingga ekstraksi fitur. OpenCV memiliki dukungan terhadap beragam bahasa pemrograman, di antaranya C++, Python, dan Java, serta dapat dijalankan pada berbagai platform sistem operasi seperti Windows, Linux, dan macOS. Kelengkapan fitur yang tersedia serta kinerjanya yang optimal menjadikan OpenCV banyak digunakan dalam aktivitas penelitian dan pengembangan aplikasi, khususnya pada bidang robotika, pemrosesan citra digital, dan kecerdasan buatan.

2.13.8 CUDA *(Compute Unified Device Architecture)*

CUDA adalah sebuah platform komputasi paralel beserta model pemrograman yang dikembangkan oleh NVIDIA untuk mengoptimalkan pemanfaatan kemampuan pemrosesan *Graphics Processing Unit* (GPU). Melalui CUDA, pengembang dapat menjalankan komputasi umum *(general-purpose computing)* secara paralel pada GPU, sehingga proses perhitungan yang bersifat kompleks dapat diselesaikan lebih cepat dibandingkan dengan pemrosesan berbasis CPU. Teknologi ini banyak diaplikasikan pada bidang komputasi intensif, seperti deep learning, pengolahan citra, simulasi numerik, serta pemrosesan data berskala besar, karena kemampuannya dalam meningkatkan efisiensi dan performa komputasi.. Gambar 2.30 merupakan logo dari CUDA.

Gambar 2. 30 Logo CUDA
(CUDA, 2025)

CUDA berperan sebagai platform komputasi paralel yang memungkinkan pengembang perangkat lunak memanfaatkan kemampuan pemrosesan grafis secara terintegrasi dengan teknologi CUDA. Dalam model CUDA, tersedia fungsi untuk melakukan alokasi memori pada perangkat serta mekanisme peluncuran *kernel* guna menjalankan proses komputasi, sementara unit pemrosesan grafis digunakan

untuk mengolah data secara paralel. Berdasarkan referensi tersebut, CUDA dimanfaatkan dalam penelitian ini untuk mendukung kinerja TensorFlow dalam memproses data grafis yang terdapat pada suatu citra (Normalisa et al., 2022).

2.13.9 Google Colab

Google Colaboratory (Google Colab) adalah layanan komputasi awan yang disediakan oleh Google untuk menjalankan lingkungan pemrograman Python secara interaktif melalui peramban web. Platform ini memungkinkan pengguna untuk menulis, mengeksekusi, serta membagikan kode dalam format notebook tanpa memerlukan instalasi perangkat lunak secara lokal. Selain itu, Google Colab menyediakan akses ke berbagai sumber daya komputasi, seperti CPU, GPU, dan TPU, sehingga banyak digunakan dalam kegiatan penelitian machine learning, deep learning, serta pengolahan data dalam skala besar (Ulfi & Savira Putri Ayu, 2024). Gambar 2.31 menunjukkan logo google colab.

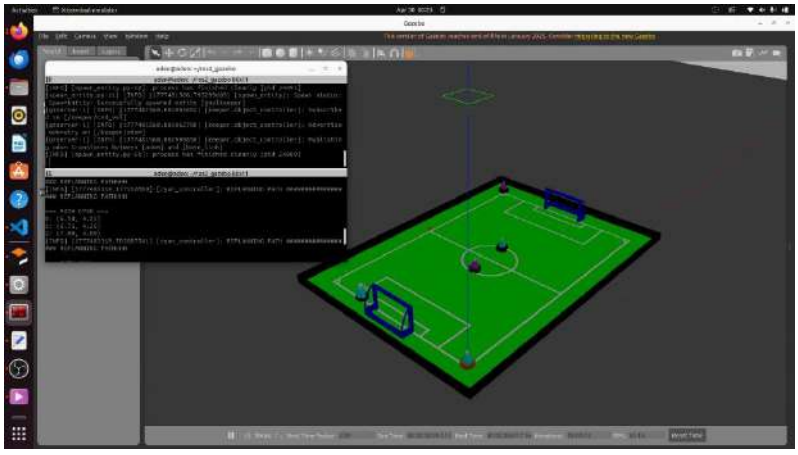
Gambar 2. 31 Logo Google Colab
(Colab, 2025)

Langkah pertama dalam pelatihan model YOLOV5S pada platform Google Colab dimulai dengan mengimpor *library* Python yang diperlukan, kemudian *repository* YOLOV5S diunduh ke lingkungan runtime aktif. Agar dataset pelatihan dapat digunakan sepanjang proses pelatihan, data diunggah ke Google Drive dan drive tersebut dipasang (*mount*) pada runtime Colab. Selanjutnya, file *train.py* dan *data.yaml* disesuaikan untuk mencocokkan lokasi dataset dan pengaturan *hyperparameter* model. Untuk memulai pelatihan, skrip *train.py* dijalankan, dan proses pelatihan dipercepat dengan memanfaatkan GPU gratis yang disediakan oleh Colab. Setelah pelatihan selesai, model YOLOV5S yang telah terlatih akan

tersimpan secara otomatis di Google Drive, sehingga siap digunakan untuk inferensi dan deteksi objek pada gambar maupun video menggunakan skrip *detect.py*. Dengan menggunakan Google Colab, pelatihan model YOLOV5S dapat dilakukan tanpa memerlukan GPU pada perangkat lokal, sehingga memberikan solusi komputasi yang efisien (Yaqin, 2025).

2.13.10 Gazebo

Gazebo merupakan perangkat lunak simulasi robotika berbasis *open-source* yang digunakan untuk memodelkan, mensimulasikan, dan memvisualisasikan sistem robot dalam lingkungan 3D. Perangkat lunak ini mendukung integrasi dengan *Robot Operating System* (ROS), serta memungkinkan penggunaan berbagai jenis sensor virtual seperti kamera, sebagaimana dapat dilihat pada gambar 2.32. Oleh karena itu, Gazebo banyak dimanfaatkan dalam penelitian dan pengembangan di bidang robotika, otomasi, dan sistem cerdas untuk mengurangi risiko kesalahan dan pengujian pada tahap implementasi nyata.



Gambar 2. 32 Tampilan Gazebo
(Penulis, 2026)

Gazebo mampu melakukan simulasi robot pada lingkungan yang kompleks, baik pada skenario dalam ruangan yang melibatkan berbagai objek dan rintangan, maupun pada lingkungan luar ruangan dengan beragam kondisi. Selain itu, Gazebo memiliki keunggulan dalam menjalankan simulasi secara *real-time*, yang memberikan umpan balik secara langsung terhadap setiap perubahan pada model maupun algoritma yang sedang diuji (Yaqin, 2025).

BAB 3
METODE PENELITIAN

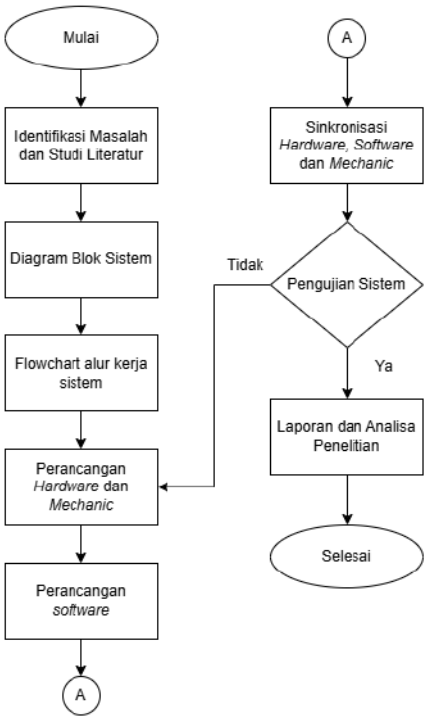
Pada bagian ini dipaparkan rangkaian tahapan yang digunakan untuk mendukung pelaksanaan penelitian, mulai dari identifikasi permasalahan, analisis kebutuhan sistem, hingga teknik perancangan yang diterapkan. Pembahasan mencakup perancangan perangkat keras yang berfokus pada komponen fisik pendukung sistem, serta perancangan perangkat lunak yang berfungsi untuk mengelola proses operasional robot. Selain itu, dilakukan pula analisis terhadap data yang diperoleh selama proses pengujian, guna memberikan pemahaman yang lebih menyeluruh dalam pengembangan dan evaluasi kinerja sistem yang diusulkan.

3.1. Kerangka Penelitian

Pada sub bagian ini memberikan penjelasan menyeluruh tentang setiap langkah yang diambil untuk melakukan penelitian selama penyusunan tugas akhir. Ini dimulai dengan menentukan masalah utama yang akan digunakan sebagai dasar penelitian. Kemudian, masalah tersebut diuraikan menjadi fokus penelitian dengan cara yang lebih spesifik dan terfokus. Setelah itu, untuk memastikan bahwa rancangan yang dikembangkan mampu memenuhi kebutuhan operasional dalam situasi kompetisi robot sepak bola beroda, dilakukan analisis kebutuhan sistem yang mencakup elemen perangkat lunak dan perangkat keras. Selain itu, bagian ini memberikan penjelasan menyeluruh tentang proses perancangan sistem. Ini mencakup penjelasan tentang komponen yang digunakan, fungsi masing-masing elemen, dan alur komunikasi antar bagian yang diperlukan untuk sistem berfungsi secara terpadu.

Selain itu, subbab ini juga memaparkan langkah-langkah teknis dalam proses perancangan dan pembangunan perangkat keras, mulai dari pemilihan komponen yang sesuai, integrasi sensor dan aktuator, hingga pengujian awal setiap modul setelah dirakit. Pada sisi perangkat lunak, pembahasan difokuskan pada pengembangan logika algoritmik yang mencakup deteksi objek menggunakan OpenCV maupun YOLOV5S sebagai alternatif pendekatan, pengolahan data posisi hasil deteksi, serta perhitungan lintasan menggunakan algoritma path planning A*.

Seluruh rangkaian proses tersebut disajikan dalam bentuk flowchart penelitian pada Gambar 3.1, yang berfungsi untuk memberikan gambaran visual dan sistematis mengenai alur penelitian dari tahap awal hingga selesai. Bagian ini juga menguraikan proses analisis data dan pembahasan yang dilakukan untuk meninjau keterkaitan antar variabel, menginterpretasikan temuan penelitian, serta menyusun kesimpulan yang objektif dan mendukung validitas hasil penelitian.



Gambar 3. 1 Kerangka Penelitian
(Penulis, 2026)

3.2. Dasar dan Konsep Pengembangan

Dalam penelitian Tugas Akhir ini, dilakukan perancangan sistem robotika cerdas yang mencakup pengembangan terpadu pada aspek perangkat keras *(hardware)* dan perangkat lunak *(software)*. Tahap awal perancangan diawali dengan penyusunan diagram blok sistem secara detail, yang bertujuan untuk merepresentasikan alur kerja setiap komponen secara sistematis, mulai dari akuisisi data sensor hingga keluaran berupa pergerakan robot. Perangkat keras yang dikembangkan meliputi beberapa sensor utama, antara lain kamera 360° OmniVision yang digunakan untuk mendeteksi objek lawan dan bola, *rotary encoder* untuk pemantauan posisi dan kecepatan roda, sensor *proximity* sebagai

pendeteksi keberadaan bola pada mekanisme penggiring, serta sensor sudut HWT101CT untuk memperoleh informasi orientasi dan arah hadap robot. Seluruh komponen perangkat keras tersebut diintegrasikan menggunakan sistem mikrokontroler STM32F407VGT6 serta unit pemrosesan citra pada laptop, yang saling terhubung melalui mekanisme komunikasi serial.

Pada sisi perangkat lunak, peneliti mengembangkan algoritma deteksi bola berbasis metode YOLOV5S yang mampu mengidentifikasi serta menentukan posisi objek secara *real-time*. Informasi hasil deteksi tersebut digunakan sebagai masukan utama dalam proses perencanaan lintasan *(path planning)* menggunakan algoritma A*. Algoritma A* dipilih karena kemampuannya dalam menghasilkan lintasan berdasarkan fungsi evaluasi serta menyesuaikan perencanaan terhadap perubahan lingkungan, termasuk keberadaan lawan yang diperlakukan sebagai rintangan dinamis. Seluruh proses pengolahan data dan pengambilan keputusan dijalankan serta dikelola menggunakan *Robot Operating System* (ROS) sebagai platform komunikasi antar node, sehingga setiap komponen sistem dapat saling terintegrasi dan beroperasi secara sinkron. Dengan pendekatan ini, sistem dirancang untuk mendukung pengambilan keputusan robot penyerang secara otonom dan strategis dalam menghadapi dinamika pertandingan yang kompleks.

3.2.1 Diagram Blok Sistem

Gambar 3.2 menggambarkan alur kerja sistem penelitian secara keseluruhan yang terbagi ke dalam tiga komponen utama, yaitu masukan (*input*), pemrosesan (*process*), dan keluaran (*output*). Ketiga komponen tersebut membentuk suatu rangkaian proses yang saling terhubung dan terkoordinasi, sehingga sistem mampu mengolah informasi lapangan secara terpadu untuk menghasilkan pergerakan robot penyerang yang adaptif dan responsif terhadap perubahan kondisi lapangan selama proses navigasi.

Pada tahap masukan (*input*), sistem memperoleh data dari empat jenis sensor utama, yaitu kamera OmniVision 360^{o}, *rotary encoder*, sensor proximity, serta sensor orientasi HWT101CT. kamera 360° *omnidirectional* berperan sebagai sistem penglihatan utama robot yang bertugas menangkap informasi visual lingkungan pertandingan secara menyeluruh dalam satu bidang pandang. Sensor ini memungkinkan robot memperoleh citra dua dimensi yang mencakup seluruh area

sekitar robot secara simultan, sehingga tidak hanya digunakan untuk mengenali elemen penting lapangan seperti garis lapangan, titik penalti, dan sudut corner, tetapi juga untuk mendeteksi keberadaan robot lawan serta objek permainan lainnya yang berpotensi menjadi hambatan pada jalur pergerakan robot. Citra yang dihasilkan oleh kamera OmniVision 360º diproses pada laptop menggunakan kombinasi pustaka OpenCV dan algoritma YOLOV5S versi s. OpenCV digunakan pada tahap pra-pemrosesan citra, meliputi konversi ruang warna, segmentasi objek, penyaringan *noise*, serta ekstraksi fitur visual. Selanjutnya, YOLOv5s melakukan proses deteksi objek secara kontinu dengan menghasilkan informasi bounding box, kelas objek, dan tingkat kepercayaan (confidence score) yang digunakan sebagai dasar pengambilan keputusan navigasi dan penghindaran rintangan. Berdasarkan informasi tersebut, sistem dapat memprediksi posisi spasial objek terhadap bidang koordinat global yang telah ditetapkan sesuai dengan dimensi lapangan (panjang 12 x 8 (meter)).

Selanjutnya, rotary encoder dimanfaatkan untuk memperoleh estimasi perpindahan robot berdasarkan perubahan posisi roda. Data dari tiga rotary encoder yang dipasang dengan konfigurasi sudut 120° dikonversikan menjadi perpindahan translasi robot melalui pendekatan inverse kinematics. Data perpindahan tersebut digunakan sebagai sumber informasi odometri untuk memperbarui estimasi posisi robot selama bergerak.

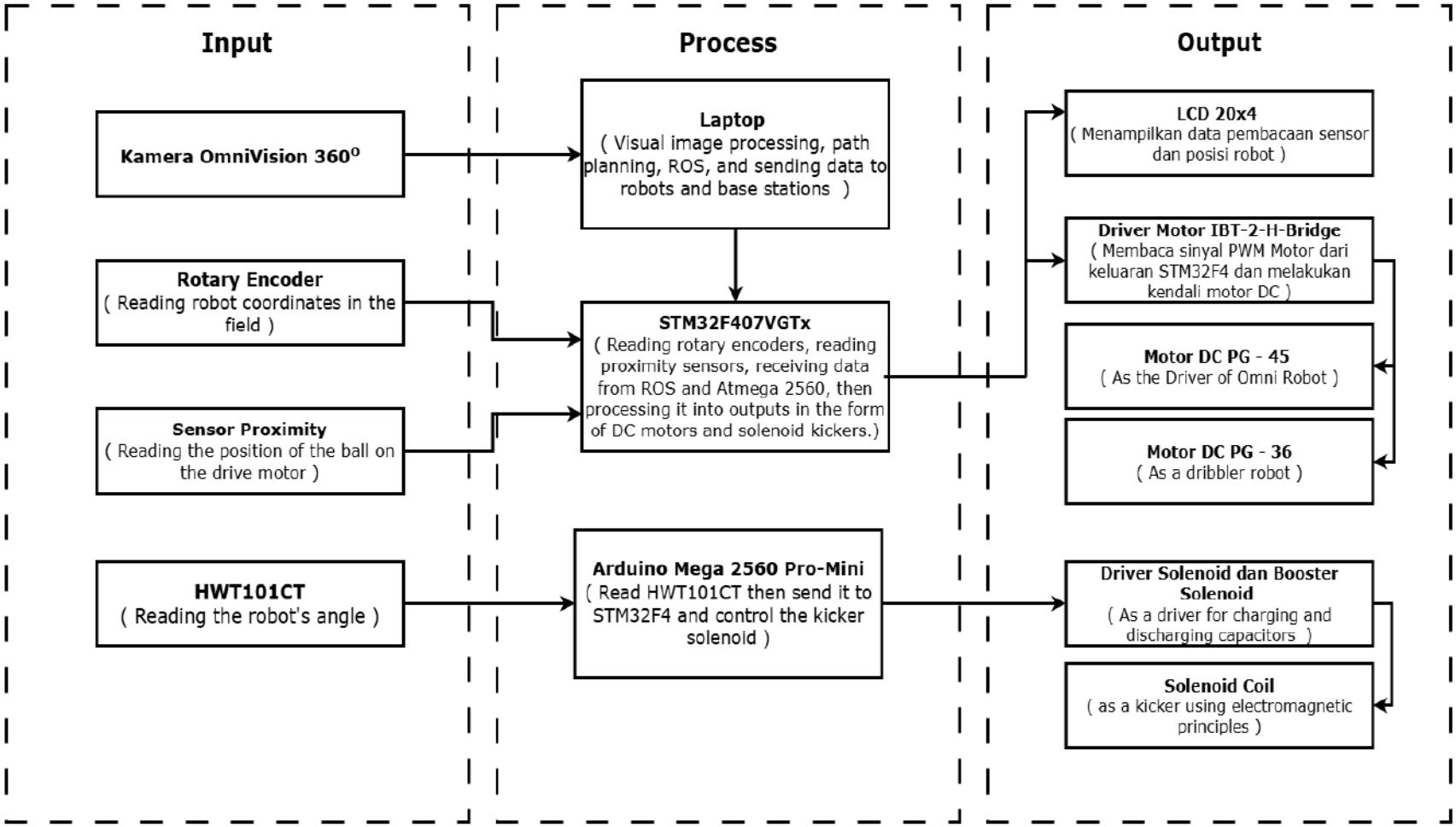
Sensor *proximity* yang bekerja berdasarkan prinsip pemantulan sinyal inframerah dipasang pada bagian mekanisme penggiring bola robot. Sensor ini berfungsi untuk mendeteksi keberadaan bola ketika berada dalam jarak deteksi efektif robot. Informasi yang dihasilkan berupa sinyal logika yang menunjukkan ada atau tidaknya bola di area penguasaan, kemudian dikirimkan ke mikrokontroler STM32 untuk diproses bersama data dari sensor lainnya.

Sensor HWT101CT, yang termasuk dalam kategori *Attitude and Heading Reference System* (AHRS), digunakan untuk memperoleh informasi orientasi sudut robot pada sumbu Z. Sensor ini mampu menghasilkan data sudut dalam rentang −180° hingga 180° dengan laju pengambilan data yang tinggi serta tingkat akurasi yang baik. Data orientasi dibaca oleh mikrokontroler ATmega2560 melalui komunikasi serial TTL, kemudian diproses dan

disesuaikan dengan kebutuhan sistem kendali arah sebelum dikirimkan ke STM32F407VGT6 sebagai pengendali utama robot.

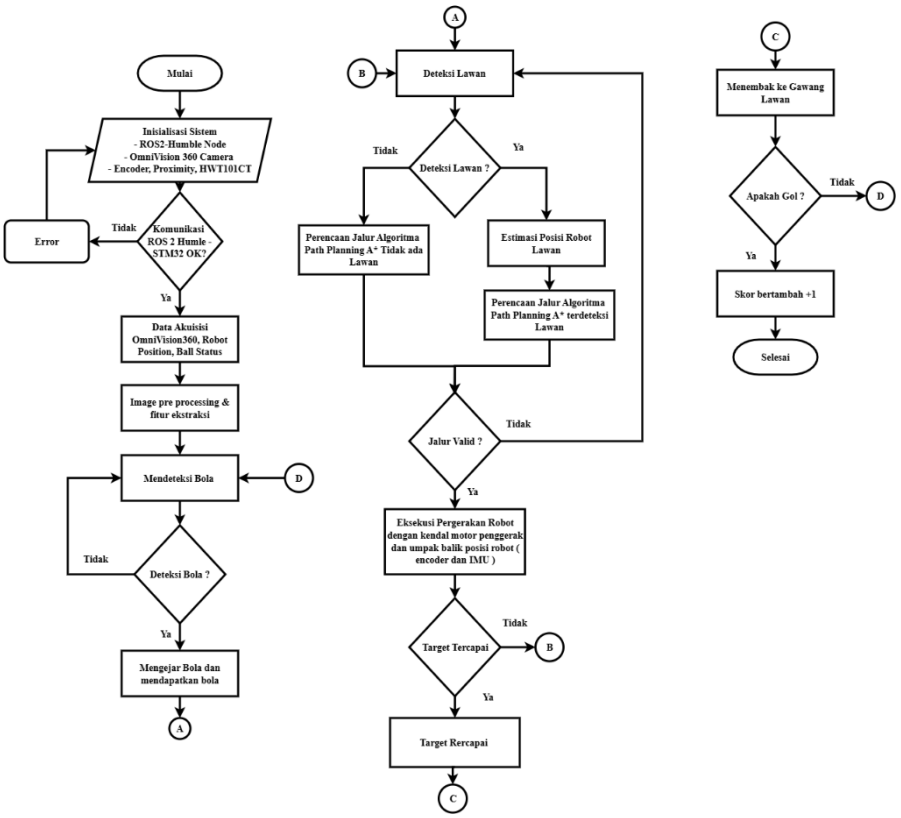
Proses *path planning* dilakukan menggunakan algoritma A* dengan mempertimbangkan posisi robot, lokasi rintangan, serta titik tujuan yang telah ditentukan. Dalam proses ini, robot lawan dimodelkan sebagai zona larangan berbentuk lingkaran sehingga jalur yang dihasilkan merupakan lintasan terpendek yang aman dan bebas dari tabrakan. Setelah jalur dengan nilai fungsi evaluasi yang sesuai diperoleh, koordinat target dikirimkan ke mikrokontroler STM32F407VGT6 melalui komunikasi serial USB.

STM32F407VGT6 kemudian berfungsi sebagai pusat eksekusi perintah gerak robot berdasarkan jalur yang dihasilkan oleh sistem *path planning* serta masukan dari sensor lokal. Sensor *proximity* digunakan untuk memastikan bola tetap berada dalam jangkauan penguasaan, sedangkan data dari rotary encoder dimanfaatkan untuk memperkirakan posisi robot selama proses navigasi dan melakukan koreksi pergerakan apabila terjadi penyimpangan dari jalur yang telah direncanakan. Sebagaimana diagram blok sistem ditunjukkan pada gambar 3.2.



Gambar 3. 2 Diagram Blok Sistem
(Penulis, 2026)

3.2.2 *Flowchart* Kerja Sistem



Gambar 3. 3 Flowchart Kerja Sistem
(Penulis, 2026)

Prinsip operasional dari penelitian ini dijelaskan melalui diagram alir yang menggambarkan tahapan kerja sistem robot penyerang sepak bola beroda secara sistematis. Diagram alir tersebut memberikan visualisasi yang jelas mengenai urutan proses yang dilakukan oleh sistem, mulai dari tahap inisialisasi hingga robot mencapai target yang ditentukan. Dengan adanya diagram alir ini, setiap proses yang terlibat dalam sistem dapat dipahami secara terstruktur, sehingga memudahkan analisis serta implementasi prosedur kerja yang telah dirancang.

Berdasarkan diagram alir yang ditunjukkan pada Gambar 3.3, prinsip operasional sistem robot penyerang sepak bola beroda dapat dijelaskan sebagai berikut.

1. Proses diawali dengan inisialisasi sistem yang meliputi aktivasi ROS 2 Humble, kamera OmniVision 360°, rotary encoder, sensor proximity, serta sensor orientasi HWT101CT. Selanjutnya dilakukan pengecekan komunikasi antara ROS 2 Humble dengan mikrokontroler STM32. Apabila komunikasi gagal,

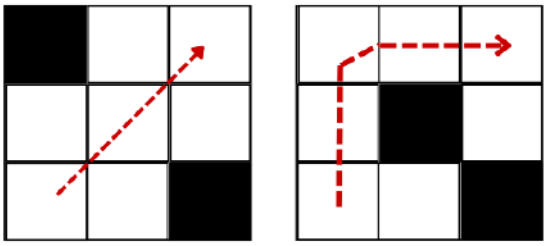
sistem akan masuk ke kondisi error.

2. Setelah komunikasi berhasil, sistem melakukan akuisisi data secara *real-time* yang meliputi citra dari kamera OmniVision 360°, posisi robot, serta status sensor yang digunakan selama proses navigasi.
3. Data citra yang diperoleh kemudian melalui tahap image preprocessing dan ekstraksi fitur menggunakan OpenCV sebagai persiapan sebelum dilakukan proses deteksi objek.
4. Sistem melakukan proses deteksi bola menggunakan algoritma YOLOV5S. Apabila bola belum terdeteksi, sistem akan terus mengulangi proses deteksi hingga bola berhasil ditemukan.
5. Setelah bola terdeteksi, robot bergerak untuk mengejar bola hingga berhasil memperoleh penguasaan bola *(ball possession)*.
6. Selanjutnya sistem melakukan deteksi robot lawan *(opponent)* menggunakan hasil deteksi dari kamera. Apabila robot lawan tidak terdeteksi, sistem langsung melakukan perencanaan jalur menggunakan algoritma A* tanpa mempertimbangkan keberadaan lawan.
7. Apabila robot lawan berhasil terdeteksi, sistem terlebih dahulu melakukan estimasi posisi robot lawan. Informasi posisi tersebut kemudian digunakan sebagai masukan pada proses *path planning* menggunakan algoritma A* sehingga lintasan yang dihasilkan mampu menghindari posisi robot lawan.
8. Jalur hasil perencanaan kemudian divalidasi. Jika jalur tidak valid, sistem kembali melakukan proses deteksi lawan dan perencanaan jalur hingga diperoleh lintasan yang layak untuk dilalui.
9. Apabila jalur dinyatakan valid, robot mengeksekusi lintasan tersebut menggunakan kendali motor penggerak. Selama robot bergerak, posisi robot terus diperbarui menggunakan data rotary encoder dan IMU sebagai umpan balik navigasi.
10. Sistem kemudian memeriksa apakah target telah tercapai. Jika target belum tercapai, proses kembali ke tahap deteksi robot lawan sehingga lintasan dapat diperbarui apabila terjadi perubahan posisi lawan. Jika target telah tercapai, robot melanjutkan proses menembak ke gawang lawan.
11. Setelah proses penembakan dilakukan, sistem mengevaluasi apakah bola

berhasil masuk ke gawang. Apabila berhasil (gol), skor bertambah satu dan proses dinyatakan selesai. Sebaliknya, apabila bola tidak berhasil masuk ke gawang, sistem kembali ke tahap deteksi bola untuk memulai kembali siklus permainan.

3.2.3 Algoritma *Path Planning* A*

Algoritma A* merupakan algoritma pencarian jalur *(path planning)* yang digunakan untuk menentukan lintasan yang layak dan aman dari posisi awal menuju posisi tujuan berdasarkan fungsi evaluasi pada setiap node. Fungsi evaluasi tersebut menggabungkan nilai perpindahan dari titik awal menuju suatu node dengan estimasi jarak menuju titik tujuan menggunakan fungsi heuristik. Dengan mekanisme tersebut, algoritma mampu memilih lintasan yang sesuai berdasarkan kondisi lingkungan sehingga proses navigasi dapat berlangsung secara efektif. Keunggulan algoritma A* terletak pada penggunaan fungsi heuristik untuk mengarahkan proses pencarian menuju target. Pada setiap iterasi, algoritma mengevaluasi node-node di sekitar posisi saat ini, kemudian membandingkan nilai evaluasi masing-masing node untuk menentukan node yang akan diproses berikutnya. Dengan cara tersebut, proses pencarian menjadi lebih terarah dibandingkan metode pencarian tanpa heuristik (Safitri et al., 2023). Dengan mekanisme tersebut, algoritma A* mampu menghasilkan lintasan yang aman, layak dilalui, dan adaptif terhadap perubahan kondisi lingkungan. Pada penelitian ini algoritma A* digunakan untuk menentukan lintasan robot menuju target dengan mempertimbangkan posisi robot lawan sebagai obstacle dinamis. Setiap node dievaluasi menggunakan fungsi evaluasi A*, sedangkan estimasi jarak menuju target dihitung menggunakan metode Euclidean Distance sebagaimana ditunjukkan pada Persamaan (3.1).



Gambar 3. 4 Simulasi Path Planning A*
(Penulis, 2026)

Persamaan utama pada algoritma A* adalah sebagai berikut:

$$f(n) = g(n) + h(n) \quad (3.1)$$

dengan keterangan :

- $f(n)$: nilai evaluasi total pada node n, yang digunakan untuk menentukan prioritas pencarian jalur.
- $g(n)$: nilai perpindahan dari posisi awal menuju node n.
- $h(n)$: nilai heuristik berupa estimasi jarak dari node $n$menuju titik tujuan.

Dalam konteks lingkungan dinamis, nilai $g(n)$ dan $h(n)$ dapat diperbarui secara *real-time* apabila terjadi perubahan kondisi jalur atau munculnya hambatan baru. Nilai $f(n)$digunakan sebagai dasar pemilihan node pada proses pencarian jalur. Node yang memiliki nilai evaluasi terkecil akan diprioritaskan untuk dikembangkan hingga lintasan menuju target berhasil diperoleh. Pada lingkungan yang bersifat dinamis, nilai evaluasi setiap node dapat berubah mengikuti perubahan posisi obstacle sehingga algoritma dapat menyesuaikan lintasan secara *real-time*.

Dalam penelitian ini, estimasi jarak menuju target dihitung menggunakan metode Euclidean Distance, karena mampu memberikan estimasi jarak geometris yang sesuai untuk pergerakan horizontal, vertikal, maupun diagonal. Persamaan matematis untuk perhitungan heuristik menggunakan Euclidean Distance dapat dinyatakan pada Persamaan 3.2.

$$h(n) = \sqrt{(x_n - x_t)^2 + (y_n - y_t)^2} \quad (3.2)$$

dengan keterangan:

- $h(n)$: nilai heuristik pada node n
- (x_n, y_n) : koordinat *node* saat ini
- (x_t, y_t) : koordinat titik tujuan

Nilai heuristik tersebut digunakan sebagai estimasi jarak geometris antara node yang sedang dievaluasi dengan titik tujuan sehingga proses pencarian menjadi lebih terarah.

Pada implementasi penelitian ini, nilai perpindahan $g(n)$tidak hanya ditentukan oleh perpindahan antar node, tetapi juga dipengaruhi oleh kondisi lingkungan di sekitar robot. Oleh karena itu, fungsi evaluasi dimodifikasi dengan menambahkan nilai *occupancy grid*, penalti kedekatan terhadap *obstacle*, dan

penalti perubahan arah. Modifikasi tersebut bertujuan agar lintasan yang dihasilkan tidak hanya mengarah menuju target, tetapi juga menjaga jarak aman terhadap robot lawan.

Berbeda dengan algoritma A* konvensional yang hanya memperhitungkan nilai perpindahan, penelitian ini memodifikasi fungsi evaluasi dengan menambahkan *occupancy grid*, penalti kedekatan terhadap obstacle, dan penalti perubahan arah. Persamaan nilai perpindahan dinyatakan sebagai berikut, Persamaan (3.3) :

$$g(n) = g(nparent) + Cmove + Cfield + Pobs + \delta Pturn \quad (3.3)$$

Dengan :

- $g(nparent)$: nilai perpindahan pada node sebelumnya.
- C_{move} : nilai perpindahan antar node.
- C_{field} : nilai pembobotan berdasarkan occupancy grid.
- P_{obs} : penalti kedekatan obstacle.
- P_{turn} : penalti perubahan arah robot.
- δ : $\begin{cases} 1, & \text{apabila arah perpindahan node berubah} \\ 0, & \textit{apabila arah perpindahan tetap} \end{cases}$

Nilai pada setiap komponen pada Persamaan (3.3) dinyatakan sebagai nilai evaluasi *(cost value)* yang tidak memiliki satuan fisik *(dimensionless).* Nilai tersebut merupakan bobot numerik yang digunakan untuk menentukan prioritas pemilihan node selama proses pencarian jalur.

Nilai perpindahan antar node (C_{move}) ditentukan berdasarkan jenis perpindahan robot. Perpindahan horizontal maupun vertikal diberikan nilai **1**, sedangkan perpindahan diagonal diberikan nilai $\sqrt{2}$ sebagaimana ditunjukkan pada Persamaan (3.4).

$$C_{move} = \begin{cases} 1, & \text{horizontal/vertikal} \\ \sqrt{2}, & \text{diagonal} \end{cases} \quad (3.4)$$

Selanjutnya setiap node juga diberikan nilai pembobotan berdasarkan occupancy grid. Semakin tinggi nilai occupancy pada suatu node, semakin besar pengaruhnya terhadap nilai evaluasi sehingga algoritma akan cenderung menghindari area tersebut. Nilai pembobotan occupancy grid dihitung

menggunakan Persamaan (3.5)

$$C_{field} = 6 \times grid(n) \tag{3.5}$$

Dengan :

- $grid(n)$: nilai occupancy pada node.

Konstanta pengali sebesar 6 merupakan parameter implementasi yang digunakan untuk meningkatkan pengaruh occupancy grid terhadap nilai evaluasi sehingga algoritma lebih menghindari area di sekitar robot lawan. Nilai konstanta tersebut diperoleh melalui proses penyesuaian (*parameter tuning*) selama implementasi sehingga pengaruh *obstacle* terhadap nilai aktual cukup besar tanpa menyebabkan jalur menjadi terlalu konservatif. Selain *occupancy grid*, penelitian ini juga menambahkan penalti berdasarkan kedekatan node terhadap obstacle. Semakin dekat node dengan obstacle, semakin besar nilai penalti yang diberikan sehingga lintasan yang dipilih memiliki jarak aman terhadap robot lawan. Nilai penalti tersebut dihitung menggunakan Persamaan (3.6).

$$P_{obs} = \sum_{i=1}^{N} \frac{grid(i)}{d_i + 0.1} \tag{3.6}$$

dengan

- $grid(i)$: nilai occupancy obstacle di sekitar node
- d_i : jarak node terhadap obstacle.
- N : jumlah sel occupancy grid yang berada pada daerah pencarian (search window) di sekitar node.

Untuk mengurangi perubahan arah yang terlalu tajam, penelitian ini juga memberikan penalti perubahan arah (*turn penalty*) sebesar 0,3. Nilai penalti sebesar 0,3 merupakan parameter implementasi yang digunakan untuk mengurangi perubahan arah secara berlebihan sehingga lintasan yang dihasilkan menjadi lebih halus tanpa mengubah karakteristik pencarian jalur secara signifikan. Sebagaimana ditunjukkan pada Persamaan (3.7).

$$P_{turn} = \begin{cases} 0.3, & \text{arah perpindahan node berturut-turut berubah} \\ 0, & \text{jika arah perpindahan node tetap} \end{cases} \tag{3.7}$$

Penambahan penalti tersebut menghasilkan lintasan yang lebih halus (*smooth path*) sehingga pergerakan robot menjadi lebih stabil.

Pada implementasi penelitian ini, algoritma A* dijalankan menggunakan mekanisme *dynamic replanning*. Posisi robot, target, dan robot lawan diperoleh secara periodik melalui komunikasi *publish–subscribe* pada ROS 2. Informasi posisi robot diperoleh dari hasil odometry, sedangkan posisi robot lawan diperoleh dari sistem persepsi berbasis kamera omnidirectional. Data tersebut digunakan untuk memperbarui *occupancy grid* sehingga kondisi lingkungan selalu merepresentasikan keadaan lapangan terkini. Setelah *occupancy grid* diperbarui, algoritma A* dijalankan kembali untuk menghasilkan lintasan baru menuju target. Dengan mekanisme tersebut, apabila robot lawan berpindah dan menutupi lintasan yang telah direncanakan, sistem akan secara otomatis menghasilkan lintasan baru tanpa menghentikan proses navigasi robot. Proses perencanaan ulang dilakukan secara periodik setiap 0,5 detik sesuai interval *timer* pada node *A* Replanner* sehingga robot mampu beradaptasi terhadap perubahan lingkungan secara *real-time*.

Pada sistem robot bergerak, keadaan robot secara umum direpresentasikan oleh posisi dan orientasi, yaitu:

$$q = (x, y, \theta) \tag{3.8}$$

dengan:

x : posisi robot pada sumbu x;

y : posisi robot pada sumbu y;

θ : orientasi robot.

Pada penelitian ini, algoritma A* melakukan proses pencarian jalur berdasarkan posisi robot pada bidang dua dimensi menggunakan koordinat (x, y). Sementara itu, orientasi robot (θ) tetap digunakan pada tahap pengendalian gerak (*motion control*) untuk mengarahkan robot mengikuti waypoint hasil *path planning*. Dengan demikian, lintasan yang dihasilkan algoritma A* berupa urutan perpindahan translasi antar node pada *occupancy grid*.

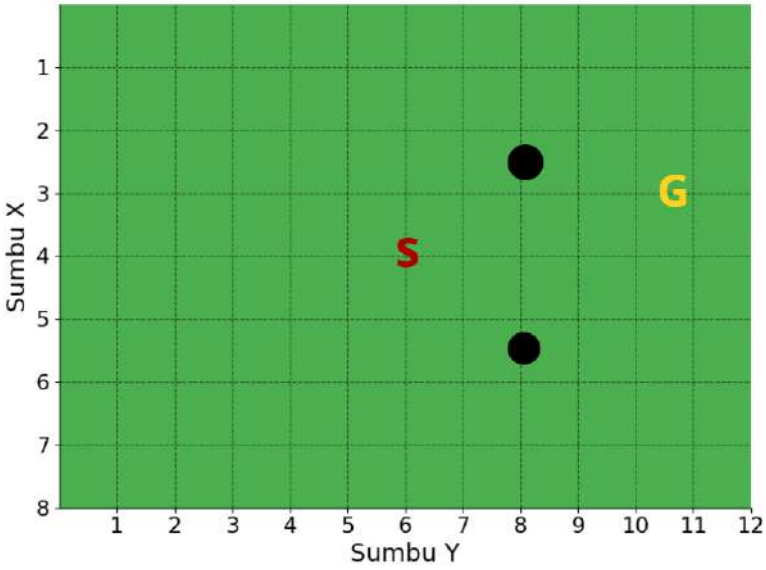
Selanjutnya, *motion controller* menghitung kecepatan translasi (v_x, v_y) dan orientasi (θ) yang diperlukan agar robot dapat mengikuti waypoint secara akurat. Nilai tersebut kemudian dikonversi menggunakan model kinematika robot omni menjadi kecepatan sudut masing-masing roda. Dengan mekanisme tersebut, hasil

perencanaan lintasan oleh algoritma A* dapat direalisasikan menjadi pergerakan robot pada lapangan secara akurat. Pendekatan ini memisahkan proses perencanaan jalur dan pengendalian gerak sehingga komputasi pencarian lintasan tetap efisien, sementara orientasi robot tetap dipertimbangkan selama proses navigasi.

3.2.4 Perhitungan *Dynamic Opponent Avoidance* dengan Algoritma A*

Pada bagian ini disajikan contoh perhitungan algoritma Path Planning A berdasarkan metode yang diimplementasikan pada penelitian ini. Perhitungan dilakukan untuk memperlihatkan proses penentuan lintasan robot dari posisi awal (*start*) menuju titik tujuan (*goal*) dengan mempertimbangkan nilai perpindahan antar node *(move)*, nilai *occupancy grid*, penalti kedekatan terhadap obstacle *(obstacle penalty),* serta penalti perubahan arah (*turn penalty*). Perhitungan mengikuti fungsi evaluasi yang telah dimodifikasi pada Persamaan (3.3), yaitu dengan menggabungkan nilai perpindahan, *occupancy grid*, penalti kedekatan terhadap obstacle, penalti perubahan arah, dan fungsi heuristik Euclidean Distance. Modifikasi tersebut bertujuan agar lintasan yang dihasilkan tidak hanya mengarahkan robot menuju target, tetapi juga menjaga jarak aman terhadap robot lawan. Berbeda dengan algoritma A* konvensional yang hanya mempertimbangkan nilai perpindahan dan fungsi heuristik, algoritma yang diusulkan pada penelitian ini menambahkan occupancy grid, penalti kedekatan terhadap obstacle, dan penalti perubahan arah pada fungsi evaluasi sehingga lintasan yang dihasilkan lebih adaptif terhadap kondisi lingkungan yang dinamis.

Untuk mempermudah proses perhitungan manual, koordinat lapangan direpresentasikan dalam bentuk grid dua dimensi berukuran 8 × 12, dengan setiap satu grid mewakili 1000 mm pada lapangan sebenarnya, sehingga *obstacle* dengan radius penghindaran 1000 mm memengaruhi node-node yang berada dalam satu grid di sekitarnya. Dengan representasi tersebut, koordinat lapangan dapat langsung dinyatakan dalam bentuk pasangan grid tanpa mengubah konsep implementasi pada sistem sebenarnya yang menggunakan satuan milimeter. Ilustrasi lingkungan simulasi yang digunakan pada contoh perhitungan ditunjukkan pada Gambar 3.5.



Gambar 3. 5 Visualisasi lapangan perhitungan Path Planning A*
(Penulis, 2026)

Pada contoh perhitungan ini, robot berada pada posisi awal (Start) di koordinat (4,6) atau (4000,6000) mm. Titik tujuan yang dipilih adalah Goal 1 pada koordinat (3,11) atau (3000,10800) mm, sedangkan posisi robot lawan diasumsikan berada pada koordinat (2.5,8) atau (2500,8000) mm dan (5.5,9) atau (5500,9000) mm. Posisi setiap objek pada contoh perhitungan ditunjukkan pada Tabel 3.1.

Tabel 3. 1 Posisi awal kondisi visualisasi

Objek	Grid	Koordinat Lapangan (mm)
Start	(4,6)	(4000,6000)
Goal 1	(3,11)	(3000,10800)
Lawan 1	(2.5,8)	(2500,8000)
Lawan 2	(5.5,9)	(5500,9000)

Selanjutnya dilakukan perhitungan nilai perpindahan $g(n)$, nilai heuristik $h(n)$, dan nilai fungsi evaluasi $f(n)$ pada setiap node yang dipilih oleh algoritma A* hingga mencapai titik tujuan. Perhitungan tersebut disajikan secara bertahap untuk menunjukkan proses evaluasi setiap node dalam menentukan lintasan yang paling sesuai menuju titik tujuan.:

- **Titik awal (4,6)**

  $g \quad = 0$

  $h \quad = \sqrt{(3-4)^2 + (11-6)^2} = \sqrt{1+25} = \sqrt{26} = 5.10$

$f \quad = g + h = 0 + 5.10 = 5.10$

- **Titik (4,7)**

Perpindahan vertical,

$C_{move} \quad = 1$

Node masih cukup jauh dari obstacle,

$C_{field} \quad = 0$

$P_{obs} \quad = 0$

Belum terjadi perubahan arah,

$P_{turn} \quad = 0$

Maka,

$g \quad = 0 + 1 + 0 + 0 + 0 = 1$

Heuristik,

$h \quad = \sqrt{(3-4)^2 + (11-7)^2} = \sqrt{17} = 4.12$

$f \quad = 1 + 4.12 = 5.12$

- **Titik (4,8)**

Perpindahan vertikal,

$C_{move} \quad = 1$

Node masih berada di luar radius pengaruh obstacle,

$C_{field} \quad = 0$

$P_{obs} \quad = 0$

Tidak terjadi perubahan arah,

$P_{turn} \quad = 0$

Sehingga,

$g \quad = 1 + 1 + 0 + 0 + 0 = 2$

Heuristik,

$h \quad = \sqrt{(3-4)^2 + (11-8)^2} = \sqrt{10} = 3.16$

$f = 2 + 3.16 = 5.16$

- **Titik (3,9)**

Perpindahan diagonal,

$C_{move} \quad = \sqrt{2} = 1.414$

$C_{field} \quad = 0$

$P_{obs} \quad = 0$

Terjadi perubahan arah,

$P_{turn} \quad = 0.3$

Sehingga,

$g \quad = 2 + 1.414 + 0 + 0 + 0.3 = 3.714$

Heuristik,

$h \quad = \sqrt{(3-3)^2 + (11-9)^2} = 2$

$f \quad = 3.714 + 2 = 5.714$

- **Titik (3,10)**

$C_{move} \quad = 1$

$C_{field} \quad = 0$

$P_{obs} \quad = 0$

$P_{turn} \quad = 0$

$g \quad = 3.714 + 1 = 4.714$

$h \quad = \sqrt{(3-3)^2 + (11-10)^2} = 1$

$f \quad = 4.714 + 1 = 5.714$

- **Titik tujuan (3,11)**

$C_{move} \quad = 1$

$C_{field} \quad = 0$

$P_{obs} \quad = 0$

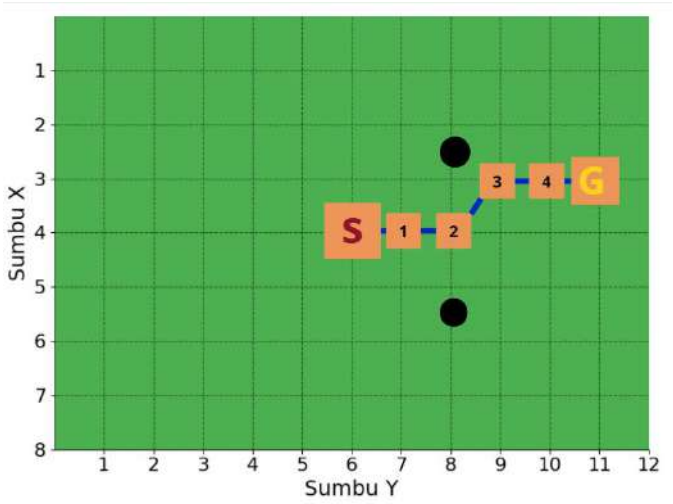
$P_{turn} \quad = 0$

$g \quad = 4.714 + 1 = 5.714$

$h \quad = 0$

$f \quad = 5.714$

Hasil perhitungan menunjukkan bahwa algoritma A* berhasil menentukan lintasan menuju Goal 1 berdasarkan nilai fungsi evaluasi $f(n)$terkecil. Dengan mempertimbangkan nilai perpindahan antar node, nilai occupancy grid, penalti kedekatan terhadap obstacle, dan penalti perubahan arah, lintasan yang dihasilkan adalah: (4,6) → (4,7) → (4,8) → (3,9) → (3,10) → (3,11). Lintasan tersebut dapat dilihat pada visualisasi Gambar 3.6.



Gambar 3. 6 Visualisasi Jalur Hasil dari perhitungan
(Penulis, 2026)

Pada kondisi berikutnya diasumsikan robot telah bergerak hingga mencapai node (4,7). Selanjutnya robot lawan 1 bergerak mendekati lintasan robot sehingga berada pada koordinat (4,8) atau (4000,8000) mm. Ketika robot mencapai node (4,7), sistem persepsi mendeteksi bahwa robot lawan berpindah ke koordinat (4,8). Informasi posisi terbaru tersebut diterima melalui mekanisme *publish–subscribe* ROS 2 sehingga occupancy grid diperbarui secara real-time. Akibatnya node (4,8) berubah menjadi area yang harus dihindari dan algoritma A* melakukan proses *dynamic replanning* menggunakan posisi robot saat ini sebagai node awal yang baru.

Tabel 3. 2 Posisi objek setelah obstacle berpindah

Objek	Grid	Koordinat Lapangan (mm)
Start Baru	(4,7)	(4000,7000)
Goal 1	(3,11)	(3000,10800)
Lawan 1	(4,8)	(4000,8000)
Lawan 2	(5.5,9)	(5500,9000)

- **Titik awal baru (4,7)**

  $g \quad = 0$

  $h \quad = \sqrt{(3-4)^2 + (11-7)^2} = \sqrt{17} = 4.12$

  $f \quad = 0 + 4.12 = 4.12$

- **Titik (3,7)**

Perpindahan horizontal,

$C_{move} \quad = 1$

Node (3,7) masih berada di luar radius penghindaran obstacle sehingga occupancy grid dan obstacle penalty bernilai nol,

$C_{field} \quad = 0$

$P_{obs} \quad = 0$

Belum terjadi perubahan arah.

$P_{turn} \quad = 0$

Sehingga,

$g \quad = 0 + 1 + 0 + 0 + 0 = 1$

Heuristik

$h \quad = \sqrt{(3-3)^2 + (11-7)^2} = 4$

$f \quad = 1 + 4 = 5$

- **Titik (2,8)**

Perpindahan diagonal

$C_{move} \quad = \sqrt{2} = 1.414$

Node (2,8) berada pada area transisi di sekitar radius penghindaran obstacle sehingga mulai dipengaruhi oleh nilai occupancy grid dan obstacle penalty. Berdasarkan hasil perhitungan occupancy grid, node (2,8) berada pada daerah transisi sehingga nilai field cost meningkat menjadi 0,5. Selain itu node tersebut masih berada pada zona pengaruh obstacle sehingga obstacle penalty bernilai 0,2

$C_{field} \quad = 0.5$

Penalty obstacle

$P_{obs} \quad = 0.2$

Karena arah berubah

$P_{turn} \quad = 0.3$

Sehingga,

$g \quad = 1 + 1.414 + 0.5 + 0.2 + 0.3 = 3.414$

Heuristik,

$h \quad = \sqrt{(3-2)^2 + (11-8)^2} = \sqrt{10} = 3.16$

$f \quad = 3.414 + 3.16 = 6.574$

- **Titik (2,9)**

  Perpindahan vertical,

  $C_{move} \quad = 1$

  Masih berada di sekitar obstacle, Karena node (2,9) mulai menjauhi obstacle, maka pengaruh occupancy grid dan obstacle penalty berkurang dibandingkan node sebelumnya.

  $C_{field} \quad = 0.3$

  Penalty *obstacle,*

  $P_{obs} \quad = 0.1$

  Arah tetap,

  $P_{turn} \quad = 0$

  Sehingga,

  $g \quad = 3.414 + 1 + 0.3 + 0.1 = 4.814$

  Heuristik,

  $h \quad = \sqrt{(3-2)^2 + (11-9)^2} = \sqrt{5} = 2.24$

  $f \quad = 4.814 + 2.24 = 7.054$

- **Titik (3,10)**

  Perpindahan diagonal,

  $C_{move} \quad = \sqrt{2} = 1.414$

  Sudah keluar dari area obstacle,

  $C_{field} \quad = 0$

  $P_{obs} \quad = 0$

  Terjadi perubahan arah,

  $P_{turn} \quad = 0.3$

  Sehingga,

  $g \quad = 4.814 + 1.414 + 0 + 0 + 0.3 = 6.528$

  Heuristik,

  $h \quad = \sqrt{(3-3)^2 + (11-10)^2} = 1$

  $f \quad = 6.528 + 1 = 7.528$

- **Titik tujuan (3,11)**

  Perpindahan vertical,

$C_{move} \quad = 1$

$C_{field} \quad = 0$

$P_{obs} \quad = 0$

$P_{turn} \quad = 0$

Sehingga,

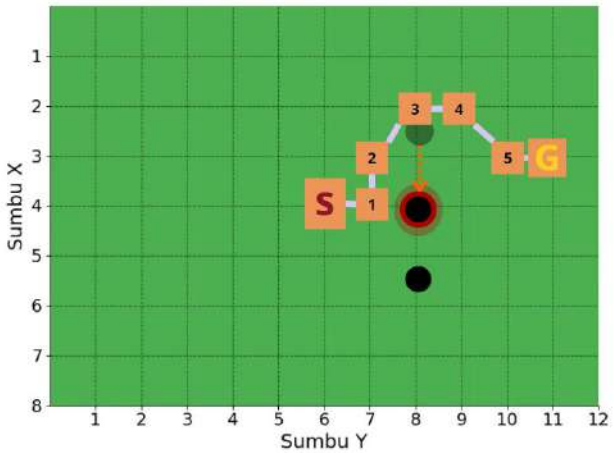
$g \quad = 6.528 + 1 = 7.528$

Heuristik,

$h \quad = 0$

$f \quad = 7.528$

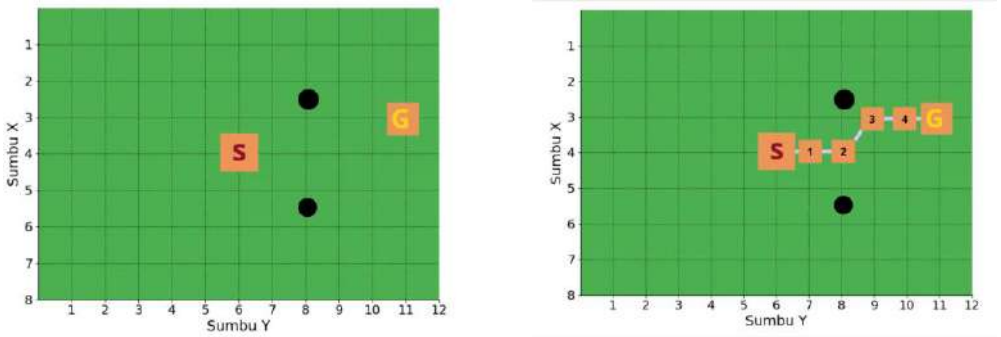
Setelah obstacle berpindah ke koordinat (4,8), node tersebut berada tepat pada lintasan yang sebelumnya telah direncanakan. Dengan radius penghindaran sebesar 1000 mm, node-node di sekitar obstacle memperoleh nilai occupancy grid dan penalti kedekatan terhadap obstacle yang lebih tinggi sehingga nilai fungsi evaluasi $f(n)$pada lintasan sebelumnya ikut meningkat. Kondisi tersebut menyebabkan algoritma A* memilih lintasan alternatif yang memiliki nilai fungsi evaluasi lebih kecil, meskipun lintasan yang ditempuh sedikit lebih panjang. Oleh karena itu, algoritma A* melakukan *dynamic replanning* dan menghasilkan lintasan baru sebagai berikut: (4,7) → (3,7) → (2,8) → (2,9) → (3,10) → (3,11). Dapat dilihat pada gambar 3.7.



Gambar 3. 7 Visualisasi hasil jalur setelah lawan bergerak
(Penulis, 2026)

Berdasarkan hasil perhitungan tersebut dapat disimpulkan bahwa mekanisme dynamic opponent avoidance mampu memperbarui lintasan robot

secara otomatis ketika terjadi perubahan posisi robot lawan. Jalur hasil replanning tidak lagi melewati node yang berada dalam radius penghindaran sebesar 1000 mm, sehingga robot tetap dapat mencapai titik tujuan melalui lintasan alternatif dengan nilai fungsi evaluasi $f(n)$yang paling kecil. Hasil ini menunjukkan bahwa integrasi occupancy grid, obstacle penalty, dan turn penalty pada algoritma A* mampu meningkatkan keamanan navigasi robot pada lingkungan yang dinamis tanpa menghilangkan kemampuan algoritma dalam mencari lintasan dengan nilai fungsi evaluasi yang rendah.



a) Kondisi awal

b) Path A* Sebelum lawan menghalangi jalur

c) Kondisi lawan menghalangi jalur

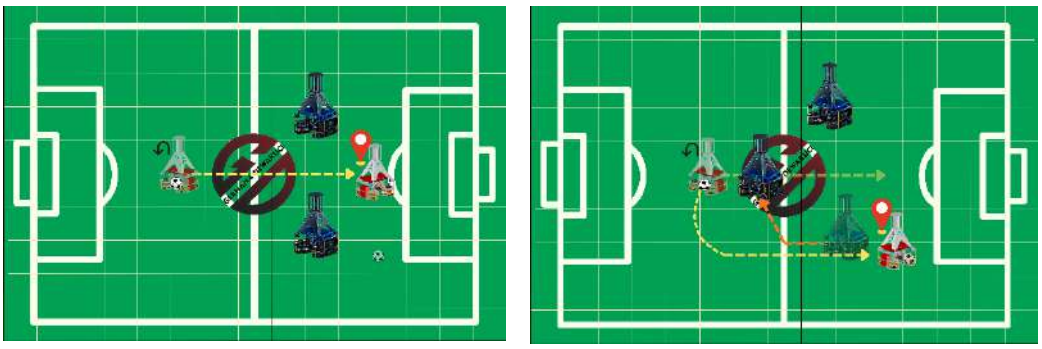
d) Hasil *dynamic replanning*

Gambar 3. 8 Visualisasi jalur path planning A*
(Penulis, 2026)

3.2.5 Model Kinematika Pergerakan Robot Penyerang

Model kinematika pergerakan robot penyerang dirancang untuk menentukan lintasan gerak robot di lapangan dengan mempertimbangkan keberadaan rintangan berupa robot lawan. Proses ini terdiri atas beberapa tahapan utama. Tahap pertama diawali dengan deteksi objek menggunakan sistem penglihatan, yang selanjutnya digunakan untuk melakukan estimasi posisi robot penyerang, robot lawan, serta target di lapangan. Tahap kedua melibatkan proses

perencanaan jalur menggunakan algoritma A* berdasarkan hasil estimasi posisi objek tersebut. Dalam perhitungan algoritma A*, lapangan permainan direpresentasikan dalam bentuk peta grid dengan ukuran sel tertentu untuk mempermudah penentuan jalur dan evaluasi nilai lintasan. Setiap sel grid diklasifikasikan sebagai area bebas atau terhalang sesuai dengan posisi robot lawan yang terdeteksi. Tahap ketiga adalah konversi titik-titik lintasan hasil perencanaan jalur menjadi parameter kecepatan motor, sehingga robot mampu bergerak sesuai lintasan yang telah ditentukan. Pola pergerakan robot penyerang akan menyesuaikan kondisi lapangan dan konfigurasi rintangan yang ada, yang meliputi pergerakan lurus, pergerakan memutar, serta pergerakan zig-zag. Contoh pola pergerakan tersebut ditunjukkan pada Gambar 3.5 hingga Gambar 3.8.

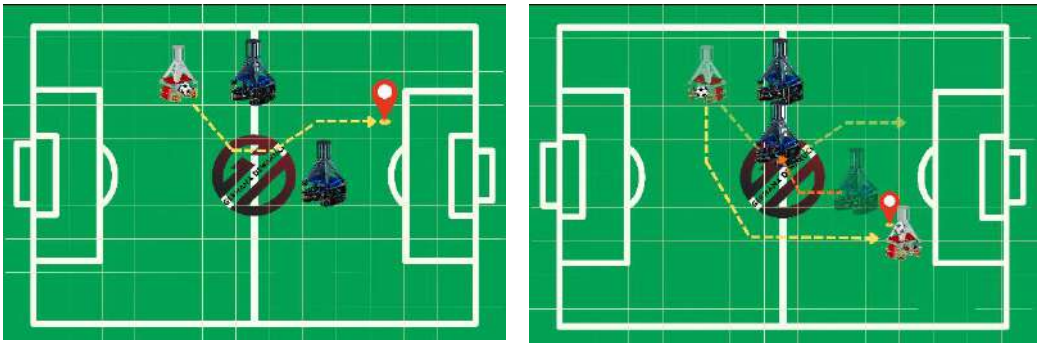


a) Pergerakan Robot Ketika *Opponent* belum bergerak

b) Pergerakan Robot Ketika *Opponent* sudah bergerak

Gambar 3. 9 Simulasi *Path Planning* A* Strategi 1
(Penulis, 2026)

Gambar 3.5 memperlihatkan simulasi pergerakan robot penyerang pada kondisi jalur menuju target relatif terbuka. Pada subgambar (a), robot lawan belum bergerak dan posisinya tidak menutup jalur langsung menuju target, sehingga algoritma A* menghasilkan lintasan berupa garis lurus menuju target karena nilai evaluasi pada jalur tersebut masih menjadi yang paling rendah. Pada subgambar (b), meskipun robot lawan telah bergerak, perubahan posisi tersebut tidak menghalangi lintasan utama sehingga algoritma tidak melakukan perubahan lintasan dan robot tetap bergerak lurus menuju target.

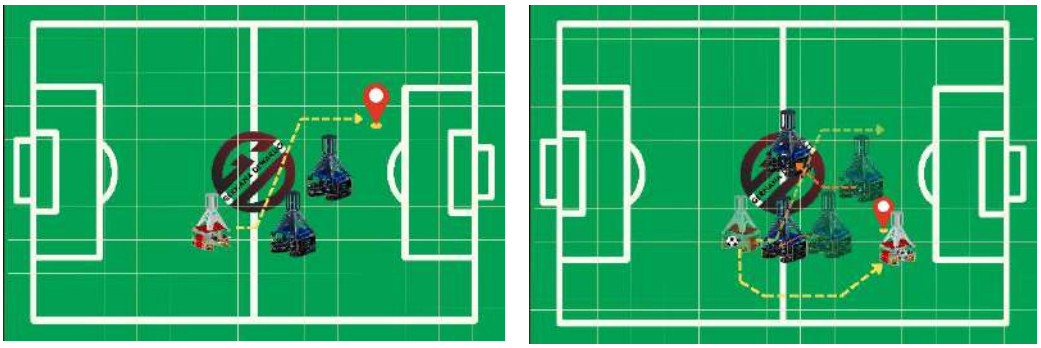


a) Pergerakan Robot Ketika *Opponent* belum bergerak

b) Pergerakan Robot Ketika *Opponent* sudah bergerak

Gambar 3. 10 Simulasi *Path Planning* A* Strategi 2
(Penulis, 2026)

Gambar 3.6 menunjukkan kondisi ketika salah satu robot lawan berada pada jalur menuju target. Pada subgambar (a), posisi robot lawan berada pada lintasan utama sehingga algoritma A mengevaluasi area tersebut sebagai daerah yang harus dihindari. Berdasarkan hasil evaluasi setiap node, algoritma kemudian memilih lintasan alternatif yang memutar di sekitar robot lawan agar robot tetap dapat bergerak menuju target dengan aman. Pada subgambar (b), ketika robot lawan mulai bergerak dan semakin mendekati jalur utama, sistem melakukan penyesuaian lintasan dengan memperbesar sudut belok, sehingga robot penyerang tetap dapat bergerak menuju target tanpa berpotensi terjadi tabrakan.



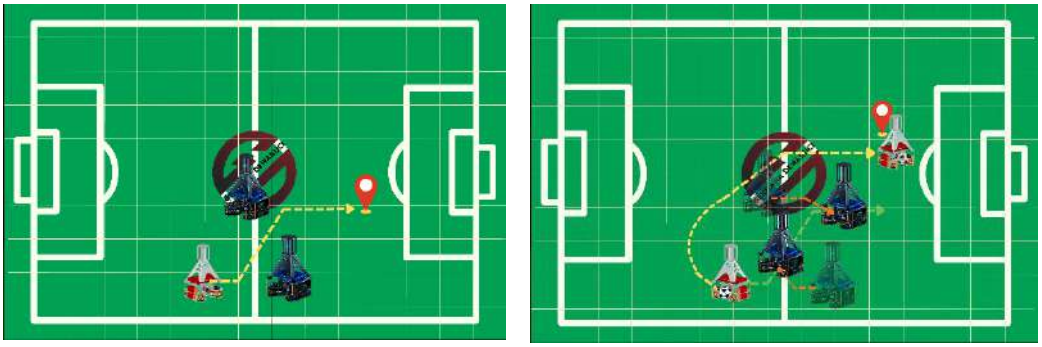
a) Pergerakan Robot Ketika *Opponent* belum bergerak

b) Pergerakan Robot Ketika *Opponent* sudah bergerak

Gambar 3. 11 Simulasi *Path Planning* A* Strategi 3
(Penulis, 2026)

Gambar 3.7 menggambarkan kondisi lapangan dengan lebih dari satu robot lawan yang menghalangi jalur menuju target. Pada subgambar (a), posisi robot lawan membentuk area occupancy yang harus dihindari sehingga jalur lurus menuju target tidak lagi menjadi pilihan. Berdasarkan hasil evaluasi node pada algoritma

A*, sistem menghasilkan lintasan alternatif berbentuk zig-zag untuk menjaga jarak aman terhadap robot lawan sambil tetap menuju target. Pada subgambar (b), ketika robot lawan bergerak dan mengubah konfigurasi hambatan, lintasan zig-zag diperbarui untuk menyesuaikan posisi terbaru rintangan agar robot penyerang tetap dapat melewati celah yang tersedia.



a) Pergerakan Robot Ketika *Opponent* belum bergerak

b) Pergerakan Robot Ketika *Opponent* sudah bergerak

Gambar 3. 12 Simulasi *Path Planning* A* Strategi 4
(Penulis, 2026)

Gambar 3.8 menunjukkan skenario dengan tingkat kompleksitas yang lebih tinggi, di mana beberapa robot lawan bergerak secara dinamis dan membentuk hambatan yang berubah-ubah. Pada subgambar (a), algoritma A* menghasilkan lintasan awal berbentuk zig-zag untuk menghindari rintangan statis. Selanjutnya pada subgambar (b), perubahan posisi robot lawan menyebabkan sistem melakukan pembaruan lintasan secara berkala, sehingga jalur yang dihasilkan menjadi lebih adaptif. Pergerakan zig-zag lanjutan ini memungkinkan robot penyerang mempertahankan lintasan yang aman dan efisien hingga mencapai target.

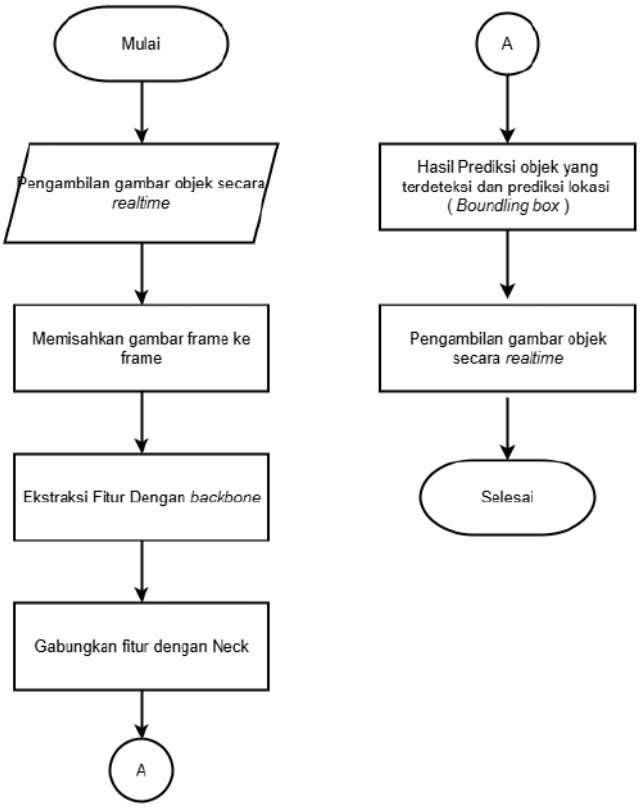
3.2.6 Arsitektur dan Proses Klasifikasi Objek pada YOLOV5S

YOLOV5S merupakan versi lanjutan dari algoritma deteksi objek *You Only Look Once* (YOLOV5S) yang dikembangkan dengan berbagai penyempurnaan dibandingkan generasi sebelumnya. Pengembangan tersebut meliputi peningkatan efisiensi pada struktur jaringan utama, kemampuan pendeteksian objek pada berbagai ukuran melalui mekanisme multi-skala, serta optimalisasi penggunaan anchor dalam proses prediksi.

Secara arsitektural, YOLOV5S terdiri atas tiga komponen utama, yaitu *backbone, neck*, dan *head*. Komponen backbone bertugas melakukan ekstraksi fitur

dari citra input, sementara bagian neck berperan dalam menggabungkan dan meningkatkan kualitas fitur yang dihasilkan dari berbagai lapisan jaringan sebelum diteruskan ke tahap selanjutnya. Selanjutnya, bagian *head* memanfaatkan fitur hasil penyempurnaan tersebut untuk menghasilkan prediksi berupa kotak pembatas *(bounding box),* probabilitas kelas objek, serta tingkat kepercayaan untuk setiap objek yang terdeteksi.

Dengan jumlah sekitar 105 lapisan, YOLOV5S dirancang untuk menghasilkan kinerja deteksi objek yang cepat dan akurat, sehingga banyak digunakan dalam berbagai aplikasi pengolahan citra dan visi komputer (Safitri et al., 2023). Sebagaimana flowchart sistem YOLOv5s ditunjukkan pada gambar 3.9.



Gambar 3. 13 Flowchart system Kerja YOLOV5S
(Penulis, 2026)

Pada Gambar 3.10 menunjukkan arsitektur dasar jaringan deteksi objek modern, seperti YOLOV5S, yang terdiri dari tiga bagian utama yaitu *Backbone, Neck*, dan *Head*. Arsitektur ini dirancang untuk mengekstraksi informasi visual dari citra masukan, menggabungkan fitur pada berbagai skala, dan menghasilkan prediksi objek secara akurat.

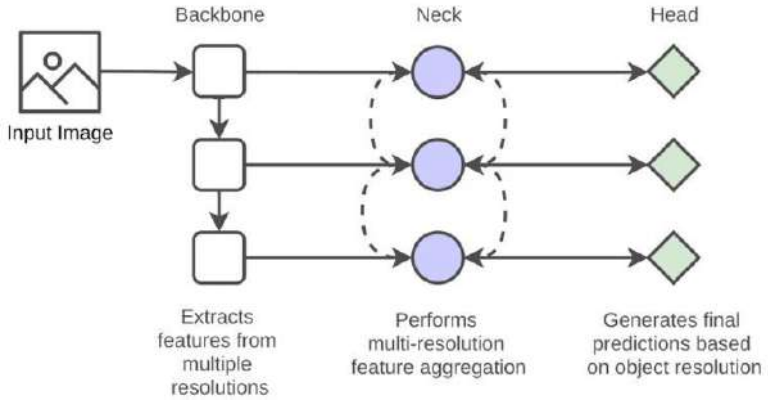
Bagian Input Image merupakan citra mentah yang diperoleh dari kamera, dalam penelitian robot sepak bola beroda biasanya berasal dari kamera OmniVision 360. Citra ini menjadi masukan awal bagi sistem untuk diproses lebih lanjut oleh jaringan saraf konvolusional.

Backbone berfungsi sebagai pengekstraksi fitur *(feature extractor)*. Pada tahap ini, jaringan mempelajari pola-pola penting dalam citra seperti tepi, warna, tekstur, dan bentuk objek. *Backbone* mengekstraksi fitur dari berbagai resolusi melalui beberapa lapisan konvolusi sehingga mampu menangkap informasi objek baik yang berukuran besar maupun kecil. Contoh backbone yang umum digunakan pada YOLOV5S adalah CSPDarknet.

Selanjutnya, *Neck* bertugas menggabungkan fitur-fitur yang dihasilkan oleh *backbone* dari berbagai tingkat resolusi. Proses ini disebut *multi-resolution feature aggregation*. Dengan mengombinasikan fitur resolusi tinggi dan rendah, neck membantu model mendeteksi objek kecil dan besar secara lebih akurat.

Bagian terakhir adalah *Head*, yang berfungsi menghasilkan keluaran akhir berupa prediksi objek. *Head* melakukan perhitungan *bounding box*, klasifikasi objek, dan nilai kepercayaan *(confidence score)* pada setiap skala fitur yang diberikan oleh *neck*. Hasil dari tahap ini berupa koordinat posisi objek, kelas objek (misalnya bola atau robot lawan), serta tingkat kepercayaan deteksi.

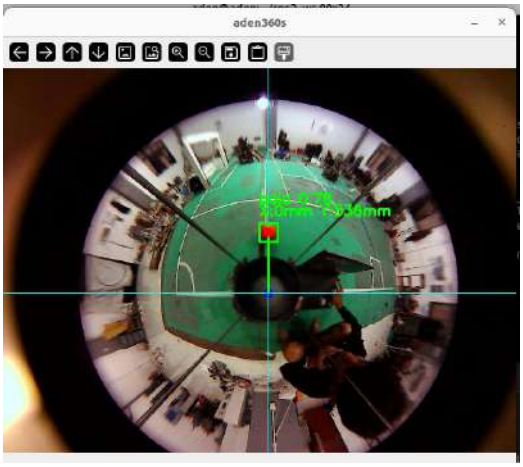
Secara keseluruhan, arsitektur *Backbone–Neck–Head* memungkinkan sistem deteksi objek seperti YOLOV5S bekerja secara efisien dan *real-time*. *Backbone* menangkap fitur penting dari citra, neck menggabungkan informasi lintas skala, dan head menghasilkan prediksi akhir yang siap digunakan untuk sistem navigasi dan perencanaan jalur robot.



Gambar 3. 14 Arsitektur klasifikasi YOLOV5S
(Penulis, 2026)

3.2.7 Sistem Persepsi Visual dan Estimasi Posisi Menggunakan Kamera OmniVision 360

Pada penelitian ini, sistem persepsi visual robot penyerang menggunakan kamera OmniVision 360 untuk memperoleh informasi lingkungan secara menyeluruh di sekitar robot. Kamera memiliki sudut pandang 360° sehingga mampu mendeteksi bola maupun robot lawan di seluruh area sekitar robot. Akuisisi citra dilakukan menggunakan OpenCV, sedangkan proses deteksi objek menggunakan algoritma YOLOV5S yang menghasilkan koordinat *bounding box* setiap objek, seperti pada Gambar 3.11.



Gambar 3. 15 Tampilan Kamera OmniVision360
(Penulis, 2026)

- Titik pusat kamera dinyatakan sebagai

$$(x_c, y_c) \tag{3.9}$$

- Sedangkan titik tengah objek hasil deteksi YOLOV5S dinyatakan sebagai

(x_{obj}, y_{obj}) (3.10)

- Koordinat objek terhadap pusat kamera dihitung menggunakan

$dx = x_{\text{obj}} - x_c$ (3.11)

$dy = y_c - y_{\text{obj}}$ (3.12)

Dengan :

- dx : selisih koordinat horizontal objek terhadap pusat kamera.
- dy : selisih koordinat vertikal objek terhadap pusat kamera.

- Perhitungan Radius Piksel

- Jarak objek terhadap pusat kamera dihitung menggunakan Persamaan Euclidean:

$r_{pixel} = \sqrt{dx^2 + dy^2}$ (3.13)

Dengan :

- r_{pixel} : jarak objek terhadap pusat kamera dalam satuan piksel.

- Konversi Piksel menjadi Jarak Aktual

Karena nilai r_{pixel} masih berupa satuan piksel, maka dilakukan proses kalibrasi menggunakan polynomial orde tiga yang diperoleh dari proses pengukuran beberapa titik referensi.

- Hubungan tersebut dinyatakan sebagai

$r_{mm} = f(r_{pixel})$ (3.14)

Dengan :

$f(r_{pixel})$ = `np.polyval(poly_coef,r_{pixel})`

$poly\_coef = np.polyfit(pixel\_points, distance\_points, 3)$

Dimana :

- r_{mm} : jarak objek dalam satuan milimeter.

- Faktor Skala

- Setelah diperoleh jarak aktual, faktor skala dihitung menggunakan

$k = \frac{r_{mm}}{r_{pixel}}$ (3.15)

Dengan :

- k : faktor konversi dari piksel ke milimeter.

- Posisi Relatif Objek

- Koordinat relatif objek terhadap robot dihitung menggunakan

$$X = dx \times k \quad (3.16)$$

$$Y = dy \times k \quad (3.17)$$

Dengan :

- X : posisi objek terhadap robot pada sumbu horizontal (mm)
- Y : posisi objek terhadap robot pada sumbu vertikal (mm)

- Transformasi ke Koordinat Global

- Koordinat relatif kemudian ditransformasikan ke sistem koordinat lapangan menggunakan posisi robot hasil odometri dan orientasi robot, sehingga posisi global objek dihitung menggunakan :

$$X_{global} = X_{robot} + X\cos\theta - Y\sin\theta \quad (3.18)$$

$$Y_{global} = Y_{robot} + X\sin\theta + Y\cos\theta \quad (3.19)$$

Dengan :

- X_{robot}, Y_{robot} : posisi robot pada lapangan berdasarkan odometri.
- X, Y : posisi relatif objek terhadap robot.
- θ : orientasi robot terhadap sumbu lapangan.

Posisi global tersebut selanjutnya digunakan sebagai masukan pada algoritma path planning A* untuk menentukan lintasan robot menuju target dengan mempertimbangkan posisi bola, robot lawan, dan kondisi lingkungan secara dinamis.

3.2.8 Prosedur Pengambilan Data

Dalam proses pengambilan data, dilakukan sebanyak 8 kali percobaan pada kondisi lapangan yang bersifat dinamis, di mana posisi robot lawan dapat berubah selama robot penyerang bergerak menuju target. Variasi kondisi pengujian meliputi robot lawan yang bergerak sejajar dengan robot tim, robot lawan yang berpindah ke salah satu sisi lintasan robot tim, serta robot lawan yang tidak sejajar dan membentuk celah yang berubah-ubah di antara kedua robot lawan.

Seluruh proses pengujian diawali oleh basestation yang berperan sebagai operator untuk memulai pergerakan robot penyerang. Basestation mengirimkan perintah awal kepada robot, kemudian robot melakukan proses persepsi visual, estimasi posisi, dan perencanaan jalur secara mandiri menggunakan algoritma A*.

Pengambilan data ini bertujuan untuk mengevaluasi kinerja sistem path

planning dalam menyesuaikan jalur pergerakan robot secara *real-time*. Parameter yang diamati berupa waktu tempuh pergerakan robot sebelum dan setelah diterapkannya metode *path planning* A*. Perbandingan hasil pengujian digunakan untuk mengetahui tingkat efisiensi waktu serta kemampuan adaptasi robot terhadap perubahan kondisi lingkungan yang dinamis.

3.3. Perencanaan dan Sistem Arsitektur

Pada subbab ini dijelaskan secara sistematis tahapan perencanaan dan perancangan sistem yang diterapkan dalam penelitian ini, khususnya pada pengembangan robot penyerang sepak bola beroda untuk kompetisi KRSBI-B. Proses perancangan diawali dengan penyusunan desain sistem yang optimal dengan mempertimbangkan aspek ekonomi, keandalan, dan efisiensi guna mendukung performa robot selama pertandingan. Selanjutnya dilakukan pemodelan sistem sebagai kebutuhan utama penelitian yang berfungsi sebagai dasar dan acuan dalam pembuatan perangkat keras dan pengembangan perangkat lunak robot.

Pemodelan tersebut mencakup perancangan berbagai komponen utama robot, seperti sistem penggerak roda, sistem kendali, sistem sensor, serta sistem visi dan pengolahan citra yang saling terintegrasi dan mendukung fungsi robot sebagai penyerang. Selain itu, dibahas pula proses integrasi antar subsistem, baik integrasi antar perangkat keras maupun integrasi antara perangkat keras dan perangkat lunak. Proses integrasi ini bertujuan untuk memastikan seluruh komponen robot dapat bekerja secara selaras, responsif, dan efisien dalam menjalankan strategi penyerangan serta mencapai tujuan penelitian yang telah ditetapkan.

3.3.1 Analisis Kebutuhan Perangkat Sistem

Setelah dilakukan perhitungan serta pertimbangan secara menyeluruh terhadap kebutuhan penelitian ini, tahap awal yang dilakukan adalah menyusun analisis kebutuhan perangkat sistem yang akan dikembangkan. Analisis ini bertujuan untuk mengidentifikasi dan merumuskan secara jelas kebutuhan-kebutuhan utama yang harus dipenuhi agar sistem dapat bekerja sesuai dengan fungsi dan tujuan yang telah ditetapkan.

Proses analisis kebutuhan perangkat sistem dilakukan untuk memperoleh gambaran yang komprehensif mengenai aspek-aspek penting yang mendukung

kinerja sistem, baik dari sisi teknis maupun operasional. Dengan adanya analisis ini, setiap komponen yang diperlukan dalam proses pengembangan sistem dapat direncanakan dan dipersiapkan secara sistematis sehingga mampu menghasilkan sistem yang efektif, efisien, dan andal. Hasil dari analisis kebutuhan perangkat sistem ini disajikan dalam Tabel 3.1.

Tabel 3. 3 Kebutuhan Perangkat Sistem

Komponen Kebutuhan Sistem	Jumlah
Kamera Omni-vision 360	1 buah
Laptop	1 buah
STM32F407VGT6	1 buah
Atmega 2560	1 buah
Sensor HWT101CT-TTL	1 buah
Sensor *Rotary Encoder*	3 buah
Sensor *Proximity*	2 buah
Driver H-*Bridge* IBT 2	6 buah
Motor DC PG-45	4 buah
Motor DC PG-36	2 buah
Liquid Crystal Diode (LCD) 20x4	1 buah

3.3.2 Desain Struktur dan Mekanisme Robot

Pada tahap perancangan sistem mekanis, desain robot disusun dengan mengacu pada peraturan Kontes Robot Sepak Bola Beroda (KRSBI-B) tingkat nasional yang ditetapkan oleh Pusprenas. Regulasi tersebut menetapkan batasan dimensi dan berat robot, meliputi :

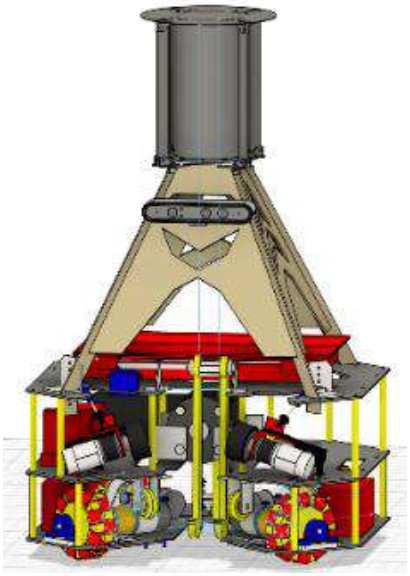
- Panjang Maks. : 520 mm
- Lebar Maks. : 520 mm
- Tinggi Maks. : 800 mm
- Berat Maks. : 40 kg

Berdasarkan ketentuan tersebut, perancangan bentuk dan struktur robot dilakukan menggunakan perangkat lunak Autodesk Fusion 360. Proses perancangan mencakup pengembangan sistem mekanik secara menyeluruh, yang terdiri dari struktur utama robot, sistem penggerak, mekanisme pengiring bola, mekanisme penendang, serta berbagai komponen pendukung lainnya. Ilustrasi

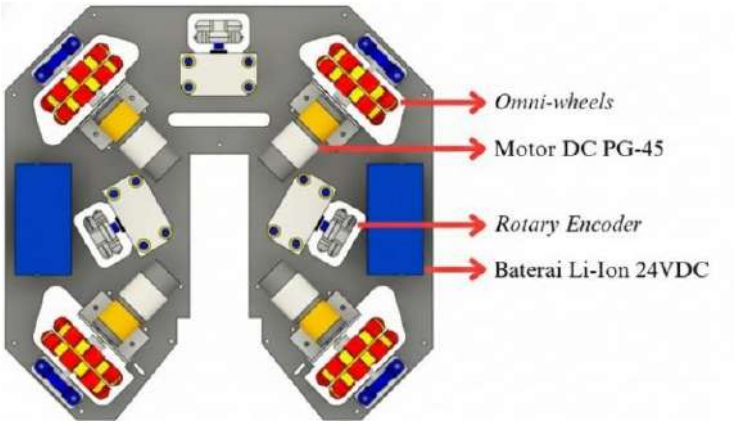
rancangan robot disajikan dalam beberapa sudut pandang, seperti Gambar 3.11.

Gambar 3.11 : - a) menunjukkan tampak depan robot
- b) menampilkan rancangan bangunan lantai 1
- c) menggambarkan rancangan bangunan lantai 2
- d) memperlihatkan tampak atas

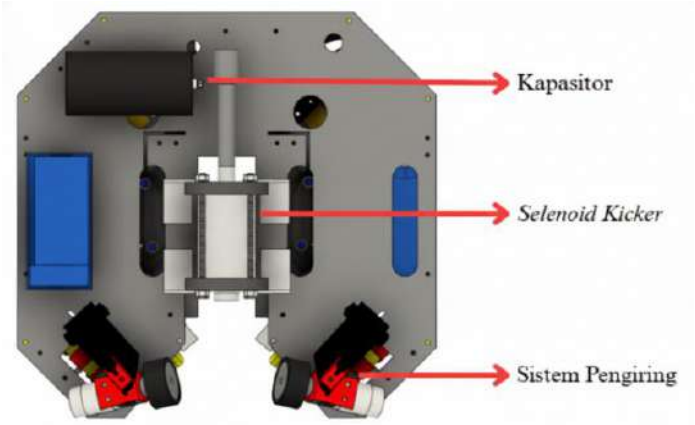
Perancangan ini dilakukan untuk memastikan bahwa robot yang dikembangkan memenuhi seluruh spesifikasi yang telah ditetapkan dalam peraturan lomba. Perhatian utama diarahkan pada pengembangan sistem penggerak, integrasi sensor, serta penerapan algoritma pengolahan data guna menghasilkan kinerja robot yang optimal. Selain itu, aspek perancangan juga mempertimbangkan kemampuan robot dalam menyesuaikan diri terhadap dinamika kondisi lapangan selama pertandingan. Dengan penerapan teknologi terkini dan pemilihan komponen yang sesuai, robot diharapkan mampu bergerak secara efisien, melakukan pengambilan keputusan secara tepat, serta berkompetisi secara maksimal dalam ajang perlombaan.



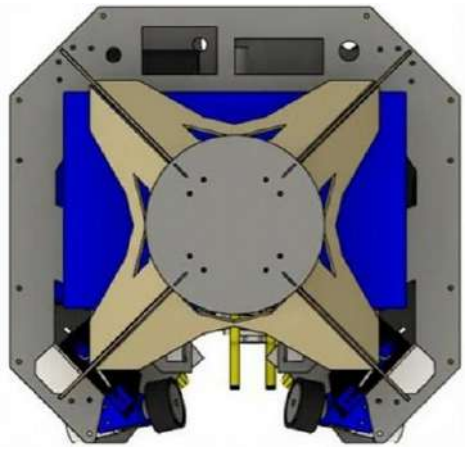
a) Robot tampak depan



b) Robot tampak depan



c) Robot tampak lantai 2



d) Robot tampak atas

Gambar 3. 16 Desain Robot Penyerang
(Penulis, 2026)

3.3.3 Arsitektur Sistem Elektrik Robot

Pada penelitian ini, sistem kelistrikan robot sepak bola beroda dirancang secara terintegrasi untuk menunjang seluruh fungsi operasional robot selama pertandingan, sebagaimana ditunjukkan pada Gambar 3.14. Sistem elektrik ini mengoordinasikan berbagai komponen sensor, aktuator, dan unit pemrosesan data agar dapat bekerja secara terpadu dan saling mendukung.

Sebagai sistem persepsi visual, robot menggunakan kamera omni 360. kamera omni 360 berfungsi memberikan pandangan menyeluruh (360 derajat) terhadap lingkungan sekitar robot. Data visual dari kedua kamera tersebut dikirimkan ke laptop yang berperan sebagai unit pemrosesan tingkat tinggi. Laptop menjalankan algoritma deteksi objek berbasis YOLOV5S serta modul estimasi dan prediksi posisi, sehingga informasi visual dapat diolah menjadi data koordinat dan referensi navigasi bagi robot.

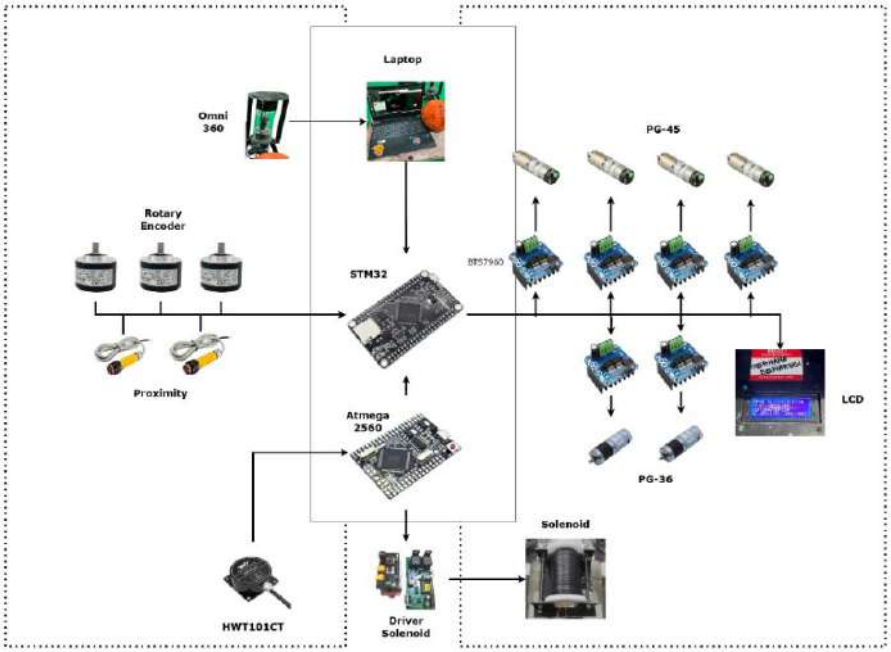
Untuk mengetahui posisi dan orientasi robot secara lokal, sistem dilengkapi dengan *rotary encoder* dan sensor HWT101CT. Rotary encoder digunakan untuk membaca pergerakan roda robot yang selanjutnya dimanfaatkan dalam perhitungan posisi relatif pada sumbu koordinat $x$dan y. Sementara itu, sensor HWT101CT digunakan untuk mengukur sudut orientasi robot pada sumbu z(yaw). Pembacaan sudut dilakukan oleh mikrokontroler ATmega 2560, kemudian data tersebut dikirimkan ke STM32F407VGT6 melalui komunikasi serial UART. STM32F407VGT6 berfungsi sebagai pengendali utama yang mengonsolidasikan data posisi dan orientasi robot secara keseluruhan.

Selain sensor posisi, robot juga dilengkapi dengan sensor proximity yang dipasang pada sistem penggiring bola. Sensor ini digunakan untuk mendeteksi keberadaan bola di area penggiring, sehingga sistem dapat menentukan kondisi logika tertentu, seperti kapan robot siap melakukan penendangan.

Pada sisi aktuator, robot menggunakan empat motor PG-45 yang terhubung ke roda omni dan dikendalikan melalui driver motor BTS7960. Konfigurasi ini memungkinkan robot melakukan manuver secara fleksibel, termasuk bergerak ke segala arah, bergerak diagonal, serta berputar pada porosnya. Untuk mekanisme penggiring bola, digunakan dua motor PG-36 yang ditempatkan di bagian depan robot. Sementara itu, sistem penendang bola menggunakan solenoid yang

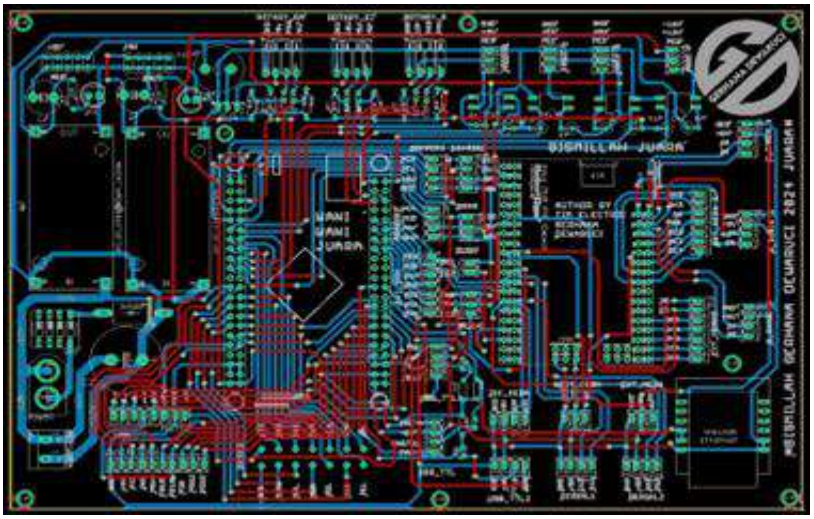
dikendalikan melalui driver khusus dengan suplai tegangan tinggi untuk menghasilkan gaya dorong yang cukup kuat saat melakukan tendangan.

Sebagai antarmuka informasi, robot dilengkapi dengan LCD 20x4 yang digunakan untuk menampilkan parameter sistem seperti sudut orientasi, status keberadaan bola, serta mode operasi robot. Sebagaimana diagram elektrikal dapat dilihat pada Gambar 3.13.



Gambar 3. 17 Diagram Komponen Sistem Elektrik Robot
(Penulis, 2026)

Dalam penelitian ini, dirancang pula sebuah *Printed Circuit Board* (PCB) khusus yang berfungsi sebagai media penghubung antara komponen-komponen utama sistem, mencakup bagian masukan, pemrosesan, hingga keluaran. Perancangan PCB ini dilakukan untuk menata koneksi antar perangkat elektronik secara lebih terstruktur. Proses perancangan PCB dilakukan menggunakan perangkat lunak Eagle PCB, yang mendukung pembuatan skematik rangkaian serta pengaturan tata letak jalur secara akurat. Hasil akhir desain PCB yang telah dirancang ditampilkan pada Gambar 3.14.



Gambar 3. 18 Rancangan PCB Robot
(Penulis, 2026)

3.3.4 Perancangan dan Perencanaan Arsitektur ROS 2 Humble

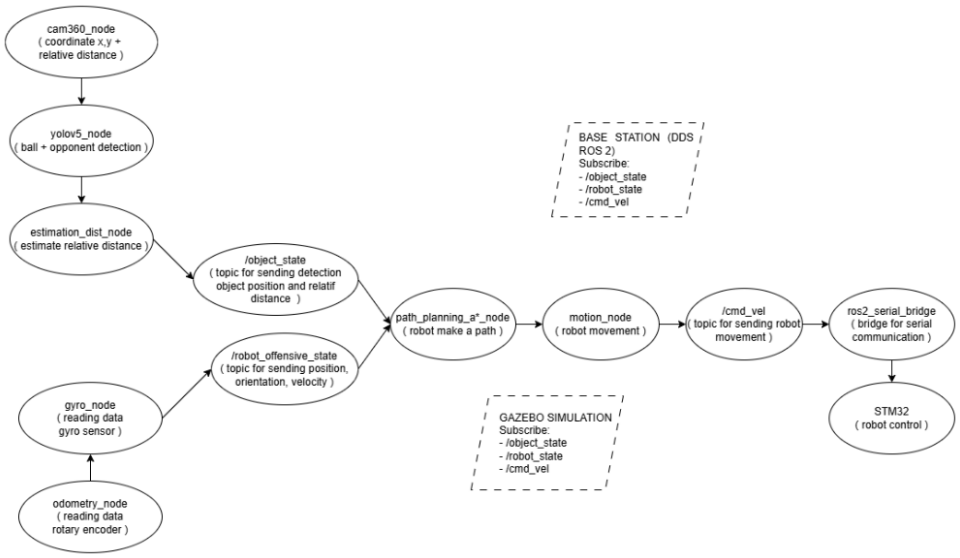
Dalam pengembangan sistem robot sepak bola beroda dengan tingkat kompleksitas tinggi, seperti pada penelitian ini, perancangan dan pembagian node pada *Robot Operating System 2 (ROS 2) Humble* menjadi aspek fundamental yang berperan penting dalam menjamin kinerja sistem yang efisien dan responsif. ROS 2 Humble mengadopsi arsitektur modular berbasis mekanisme *publish-subscribe* yang dibangun di atas *Data Distribution Service* (DDS), sehingga memungkinkan setiap node menjalankan fungsi spesifik secara mandiri namun tetap terhubung dalam sistem komunikasi yang andal dan terstruktur (Al-Batati et al., 2025).

Perancangan struktur node yang optimal memungkinkan proses komputasi berjalan secara paralel dan mendukung komunikasi real-time dengan latensi rendah, sekaligus meningkatkan kemudahan dalam proses pengujian (*debugging*), pemeliharaan sistem, serta pengembangan dan integrasi fitur baru tanpa mengganggu komponen lain yang telah berjalan. Sebagai contoh, node pengolahan citra dapat difokuskan pada proses deteksi objek visual secara independen tanpa memengaruhi kinerja node navigasi maupun node pengendali motor.

Selain itu, pendekatan modular pada ROS 2 Humble memberikan fleksibilitas tinggi dalam melakukan pembaruan atau penggantian perangkat keras dan perangkat lunak secara terpisah, termasuk dukungan terhadap sistem terdistribusi dan *Quality of Service (QoS)* yang dapat disesuaikan dengan kebutuhan aplikasi. Dalam konteks kompetisi robot sepak bola seperti KRSBI-B, yang

menuntut respons cepat terhadap perubahan kondisi lapangan dan pergerakan lawan secara dinamis, pengelolaan node yang efisien dan deterministik menjadi faktor kunci dalam pengambilan keputusan yang cepat dan akurat. Oleh karena itu, perancangan struktur node yang optimal pada ROS 2 Humble merupakan landasan utama untuk memastikan sistem robot mampu beroperasi secara adaptif, stabil, dan terukur dalam berbagai skenario pertandingan.

Arsitektur ini membentuk kerangka utama sistem persepsi, navigasi, dan kontrol pada robot sepak bola beroda, dengan pendekatan modular berbasis mekanisme *publish-subscribe* yang berjalan di atas *Data Distribution Service (DDS)*. Sebagaimana ditunjukkan pada Gambar 3.15.



Gambar 3. 19 ROS 2 Humble Architecture
(Penulis, 2026)

Proses sistem dimulai dari node kamera OmniVision 360 yang berfungsi sebagai sensor visual utama untuk memantau kondisi lapangan secara menyeluruh. Kamera *omni-directional* ini mampu menangkap citra 360 derajat di sekitar robot, sehingga memberikan keunggulan dalam mendeteksi objek penting seperti bola dan robot lawan tanpa memerlukan pergerakan kamera maupun robot secara aktif. Informasi visual yang diperoleh bersifat global dan kontinu, sehingga sangat mendukung kebutuhan persepsi cepat pada pertandingan robot sepak bola beroda.

Data citra dari kamera OmniVision 360 selanjutnya diteruskan ke node deteksi objek berbasis YOLOV5S, yang bertugas mengidentifikasi objek seperti

bola dan robot lawan. Hasil deteksi ini kemudian diproses oleh node estimasi jarak untuk memperoleh informasi posisi objek beserta jarak relatifnya. Informasi posisi objek ditransformasikan dari koordinat citra ke koordinat lapangan menggunakan kerangka referensi yang konsisten, sehingga sistem navigasi memiliki representasi posisi yang mendekati kondisi nyata di lapangan.

Informasi posisi dan jarak relatif objek tersebut dipublikasikan melalui topik ROS 2 sebagai object state dan digunakan sebagai masukan bagi node perencanaan jalur berbasis algoritma A. Pada tahap ini, algoritma A* digunakan untuk menghasilkan lintasan menuju target dengan mempertimbangkan keberadaan robot lawan sebagai obstacle dinamis. Posisi robot lawan direpresentasikan pada *occupancy grid* dengan radius keamanan tertentu sehingga node-node di sekitar robot lawan memiliki nilai evaluasi yang lebih tinggi. Dengan pendekatan tersebut, lintasan yang dihasilkan tidak hanya mengarah menuju target, tetapi juga menjaga jarak aman terhadap robot lawan sehingga navigasi robot dapat berlangsung secara aman dan adaptif terhadap perubahan lingkungan.

Selain melakukan perencanaan jalur awal, node Algoritma A* juga mendukung mekanisme *replanning* secara berkala selama proses navigasi. Replanning dilakukan ketika terdeteksi perubahan posisi robot lawan atau kondisi lingkungan yang signifikan. Dengan demikian, robot mampu beradaptasi terhadap dinamika pertandingan dan menghindari potensi tabrakan selama bergerak menuju target.

Hasil perencanaan jalur selanjutnya dikonversi menjadi perintah gerak robot dan dipublikasikan melalui topic /cmd_vel. Data ini dikirim ke dua tujuan utama. Pertama, ke simulator Gazebo, yang digunakan untuk memvisualisasikan pergerakan robot dalam lingkungan simulasi. Visualisasi ini berfungsi sebagai sarana evaluasi untuk mengamati perilaku navigasi robot serta melakukan pengujian dan validasi algoritma sebelum diterapkan pada sistem nyata. Kedua, perintah gerak diteruskan ke node komunikasi serial yang berperan sebagai penghubung antara sistem ROS 2 dengan perangkat keras robot.

Melalui mekanisme komunikasi tersebut, data perintah gerak dikirim ke STM32, yang berfungsi sebagai pengendali utama sistem motor dan aktuator robot. Mikrokontroler mengeksekusi perintah kecepatan dan arah motor secara *real-time*

sesuai lintasan yang telah direncanakan. Sebaliknya, STM32 juga mengirimkan data umpan balik berupa informasi posisi dan status pergerakan robot kembali ke sistem ROS 2, sehingga terbentuk komunikasi dua arah yang sinkron antara perangkat lunak dan perangkat keras.

Secara keseluruhan, arsitektur sistem antar-node ini dirancang secara modular dan paralel untuk mendukung proses navigasi robot. Pendekatan ini memungkinkan integrasi yang efisien mulai dari persepsi visual berbasis kamera OmniVision 360, estimasi posisi objek, perencanaan jalur adaptif menggunakan Algoritma A* dengan *replanning*, hingga eksekusi kendali motor. Arsitektur tersebut mendukung *real-time decision making* yang sangat krusial dalam kompetisi (KRSBI-B), di mana kondisi lapangan dan pergerakan lawan dapat berubah dengan cepat dan tidak terduga.

3.4. Integrasi dan Instalasi Sistem Pendukung

Pada tahap ini dilakukan implementasi dan instalasi perangkat lunak yang digunakan untuk mendukung pengembangan sistem navigasi robot penyerang berbasis *Robot Operating System* 2 (ROS 2). Tahapan ini meliputi instalasi ROS 2 Humble sebagai middleware komunikasi antar node, instalasi YOLOV5S beserta library pendukung untuk proses deteksi robot lawan secara *real-time*, pembuatan *workspace* dan *package* ROS 2 sebagai lingkungan pengembangan perangkat lunak, serta instalasi Gazebo sebagai media simulasi sistem robot sepak bola. Seluruh perangkat lunak tersebut diintegrasikan sehingga membentuk sistem yang mampu melakukan deteksi robot lawan menggunakan kamera omnidirectional 360°, membangun representasi lingkungan dalam bentuk *occupancy grid*, melakukan proses *path planning* menggunakan algoritma A*, serta mengirimkan jalur hasil perencanaan kepada robot penyerang secara *real-time*.

3.4.1 Instalasi YOLOv5s dan *Software* pendukung

Langkah awal dalam instalasi YOLOV5S dan *library* pendukung dilakukan dengan melakukan *clone* repositori YOLOV5S dari GitHub, kemudian menginstal seluruh dependensi menggunakan perintah *pip install -r requirements.txt*. Selain itu, dilakukan instalasi PyTorch, OpenCV, dan PySerial untuk mendukung proses deteksi objek, pengolahan citra, serta komunikasi serial antara komputer dan robot.

Agar proses inferensi berjalan lebih cepat, juga dilakukan instalasi CUDA Toolkit sehingga YOLOV5S dapat memanfaatkan akselerasi GPU. Tahapan instalasi dilakukan sebagai berikut:

1. Install Python versi 3.10 atau lebih tinggi

```
sudo apt install pyhton3 (Enter)
```

2. Clone Repositori YOLOV5S

```
git clone https://github.com/ultralytics/YOLOv5s.git cd YOLOv5s (Enter)
```

3. Install Dependensi

```
pip3 install -r requirements.txt (Enter)
```

4. Install pyserial

```
pip3 install pyserial (Enter)
```

5. Install json

```
pip3 install json (Enter)
```

6. Install PyTorch

```
pip install torch==2.3.0 torchvision==0.18.0 torchaudio==2.3.0 --index-url
https://download.pytorch.org/whl/cu118 (Enter)
```

7. Install CUDA Toolkit

```
wget
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x8
6_64/cuda-ubuntu2204.pin && \
sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-repository-pin-
600 && \
sudo apt-key adv --fetch-keys
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x8
6_64/3bf863cc.pub && \
sudo add-apt-repository "deb
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x8
6_64/ /" && \
sudo apt update && \
sudo apt install -y cuda-11-8 (Enter)
```

8. Cek instalasi CUDA

nvidia-smi (*Enter*)

9. Menjalankan YOLOV5S

python detect.py –source 0 (*Enter*)

3.4.2 Instalasi *Robot Operating System* 2 *Humble* (ROS 2-*Humble*)

Pada tahap ini dilakukan instalasi *Robot Operating System* 2 (ROS 2) *Humble* sebagai *middleware* utama yang menghubungkan seluruh modul pada sistem robot penyerang. ROS 2 Humble dipilih karena kompatibel dengan sistem operasi Ubuntu 22.04 LTS serta mendukung komunikasi *real-time* berbasis *Data Distribution Service* (DDS). Setelah proses instalasi selesai, dilakukan konfigurasi *environment* dan pengujian menggunakan demo *publisher-subscriber* untuk memastikan seluruh komponen ROS 2 telah terpasang dan berfungsi dengan baik. Tahapan instalasi dilakukan sebagai berikut :

1. Melakukan cek konfigurasi pada locale sistem

locale (*Enter*)

2. Menginstal dan memastikan package *locales* tersedia

sudo apt update && sudo apt install locales (*Enter*)

3. Melakukan generate locale UTF-8

sudo locale-gen en_US en_US.UTF-8 (*Enter*)

4. Mengatur environment locale

sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8 (*Enter*) export LANG=en_US.UTF-8 (*Enter*)

5. Verifikasi pengaturan locale

locale (*Enter*)

6. Memastikan repository universe aktif

apt-cache policy | grep universe (*Enter*)

7. Menginstal curl sebagai dependency

sudo apt update && sudo apt install curl -y (*Enter*)

8. Mengambil versi terbaru ROS apt source dari GitHub

export ROS_APT_SOURCE_VERSION=$(curl -s https://api.github.com/repos/ros-infrastructure/ros-apt-source/releases/latest

```
| grep -F "tag_name" | awk -F\" '{print $4}') (Enter)
```

9. Mengunduh package ros2-apt-source

```
curl -L -o /tmp/ros2-apt-source.deb "https://github.com/ros-infrastructure/ros-apt-source/releases/download/${ROS_APT_SOURCE_VERSION}/ros2-apt-source_${ROS_APT_SOURCE_VERSION}.$(. /etc/os-release && echo ${UBUNTU_CODENAME})_all.deb" (Enter)
```

10. Menginstal ros2-apt-source

```
sudo dpkg -i /tmp/ros2-apt-source.deb (Enter)
```

11. Memperbarui daftar repository

```
sudo apt update (Enter)
```

12. Memperbarui sistem

```
sudo apt upgrade (Enter)
```

13. Menginstal ROS2-Humble pada Desktop

```
sudo apt install ros-humble-desktop (Enter)
```

14. Mengaktifkan environment ROS2

```
source /opt/ros/humble/setup.bash (Enter)
```

Setelah proses instalasi ROS 2 Humble selesai, dilakukan pengujian untuk memastikan seluruh komponen telah terpasang dengan benar. Pengujian dilakukan menggunakan demo *publisher-subscriber* sederhana serta simulasi **Turtlesim** untuk memverifikasi bahwa mekanisme komunikasi antar *node* pada ROS 2 berjalan dengan baik.

15. Menjalankan demo publisher (*talker*)

```
ros2 run demo_nodes_cpp talker (Enter)
```

16. Menjalankan demo subcriber (*listener*)

```
ros2 run demo_nodes_py listener (Enter)
```

17. Menjalankan simulasi turtlesim

```
ros2 run turtlesim turtlesim_node (Enter)
```

18. Mengontrol turtle sim menggunakan keyboard

```
ros2 run turtlesim turtle_teleop_key (Enter)
```

3.4.3 Membuat *Workspace*

Workspace merupakan direktori utama pada ROS 2 yang digunakan sebagai lingkungan pengembangan perangkat lunak. Di dalam *workspace* tersimpan berbagai *package*, *library*, serta berkas konfigurasi yang mendukung proses pengembangan dan integrasi sistem robot. Pada penelitian ini, *workspace* dibuat sebagai tempat pengembangan modul deteksi robot lawan, *path planning* A*, serta komunikasi antar *node* pada ROS 2. Tahapan pembuatan *workspace* dilakukan sebagai berikut :

1. Membuat folder *workspace* beserta folder src sebagai tempat penyimpanan package

mkdir -p ~/ros2_ws/src (*Enter*)

2. Masuk ke direktori *workspace*

cd ~/ros2_ws (*Enter*)

3. Melakukan *build workspace* menggunakan *colcon build*

colcon build (*Enter*)

4. Mengaktifkan *workspace* dengan melakukan *source* pada file setup

source install/setup.bash (*Enter*)

5. Menambahkan konfigurasi *workspace* agar aktif secara otomatis saat terminal di jalankan

echo "source ~/robot_ws/install/setup.bash" >> ~/.bashrc (*Enter*)

6. Memuat ulang konfigurasi bash

source ~/.bashrc (*Enter*)

3.4.4 Membuat *Package*

Package merupakan komponen utama dalam pengembangan perangkat lunak pada sistem ROS 2 yang berfungsi sebagai wadah untuk menyimpan program, *library*, serta berkas konfigurasi yang digunakan dalam menjalankan fungsi tertentu pada robot. Setiap *package* dapat terdiri atas satu atau lebih *node* yang bertugas melakukan pemrosesan data, komunikasi antar modul, dan pengendalian sistem robot. Setelah *workspace* berhasil dibuat, langkah selanjutnya

adalah membuat *package* pada direktori src sebagai tempat pengembangan perangkat lunak. Pada penelitian ini, *package* digunakan untuk mengembangkan modul deteksi robot lawan menggunakan YOLOV5S, sistem *path planning* berbasis algoritma A*, serta komunikasi antar *node* yang terintegrasi dalam arsitektur ROS 2 *Humble*.

1. Masuk ke direktori src pada *Workspace*

cd ~/ros2_ws/src (*Enter*)

2. Membuat *package* baru menggunakan perintah ros2 pkg create

ros2 pkg create robot_pkg --build-type ament_python --dependencies rclpy std_msgs (*Enter*)

3. Kembali ke direktori *workspace*

cd ~/ros2_ws (*Enter*)

4. Melakukan *build package* menggunakan colcon

colcon build (*Enter*)

5. Mengaktifkan *package* hasil *build*

source install/setup.bash (*Enter*)

6. Melakukan cek *package* telah berhasil dibuat

ros2 pkg list | grep robot_pkg (*Enter*)

(Halaman ini sengaja dikosongkan)

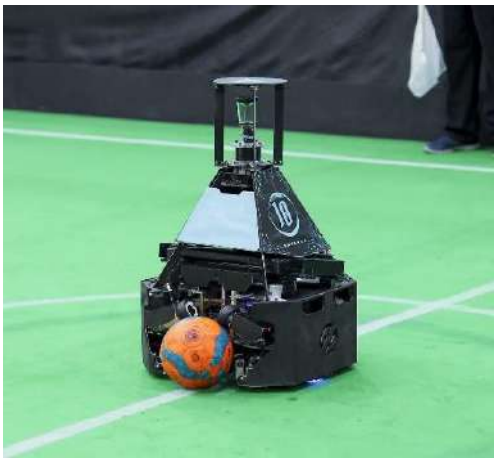
BAB 4
HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil perancangan, pengujian dan pembahasan seluruh komponen *hardware* maupun *software* yang digunakan pada Tugas Akhir ini.

4.1. Hasil Perancangan *Hardware*

4.1.1 Hasil Perancangan Mekanik

Setelah tahap perancangan mekanik selesai dilakukan pada pembahasan sebelumnya, langkah selanjutnya adalah realisasi dalam bentuk fisik serta pengujian sistem mekanik guna memastikan setiap komponen berfungsi sesuai dengan yang dirancang. Hasil perancangan tersebut telah diwujudkan dalam bentuk robot dengan tetap mempertahankan dimensi sesuai regulasi, yaitu panjang 520 mm, lebar 520 mm, dan tinggi 800 mm.



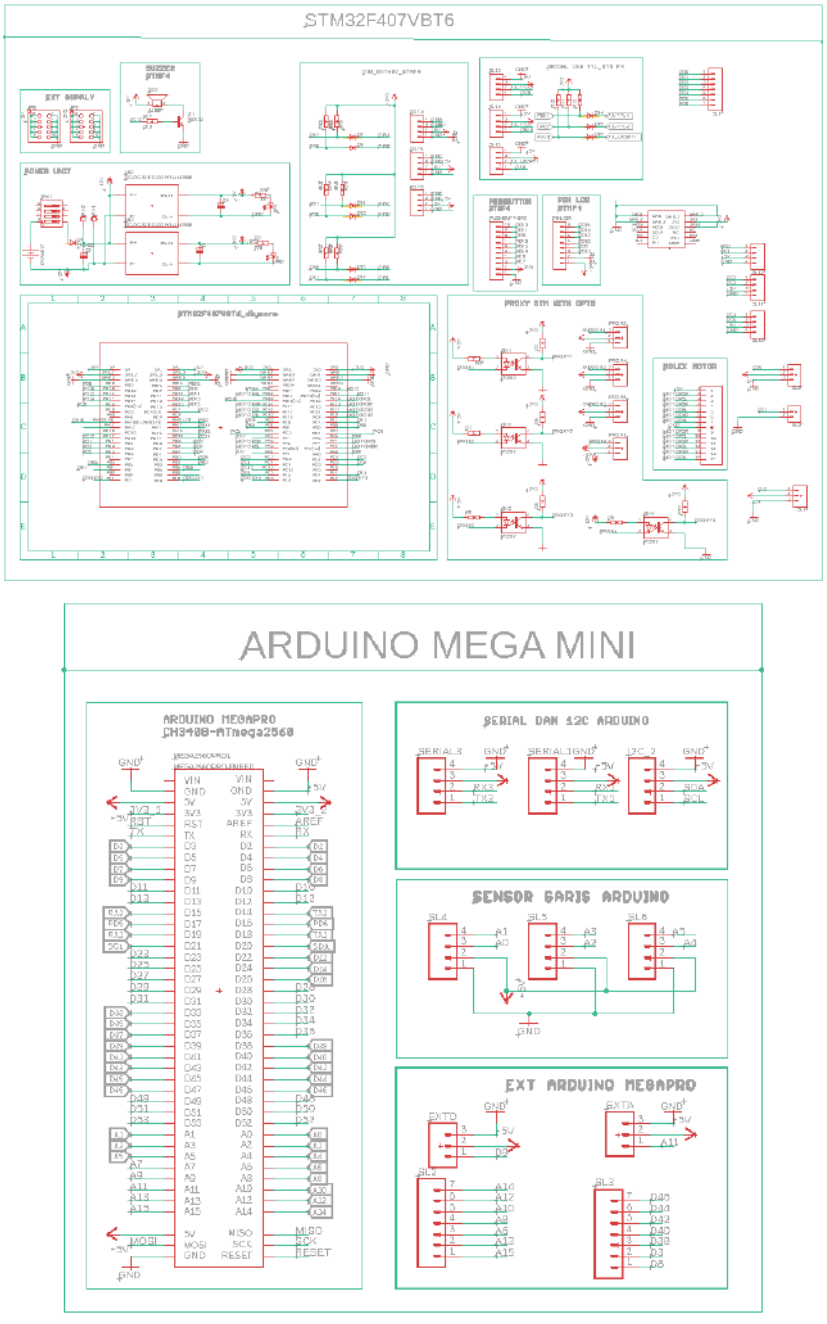
Gambar 4. 1 Robot Penyerang
(Penulis, 2026)

Pada Gambar 4.1 ditunjukkan tampak depan robot yang memperlihatkan posisi pemasangan kamera, struktur pelindung, serta susunan roda omni yang simetris. Kamera ditempatkan pada bagian atas depan robot untuk memperoleh cakupan pandangan yang lebih luas. Sementara itu, sistem penendang *(kicker)* dan mekanisme penggiring bola berada pada bagian bawah depan robot, sehingga memudahkan proses penguasaan dan penendangan bola selama pergerakan.

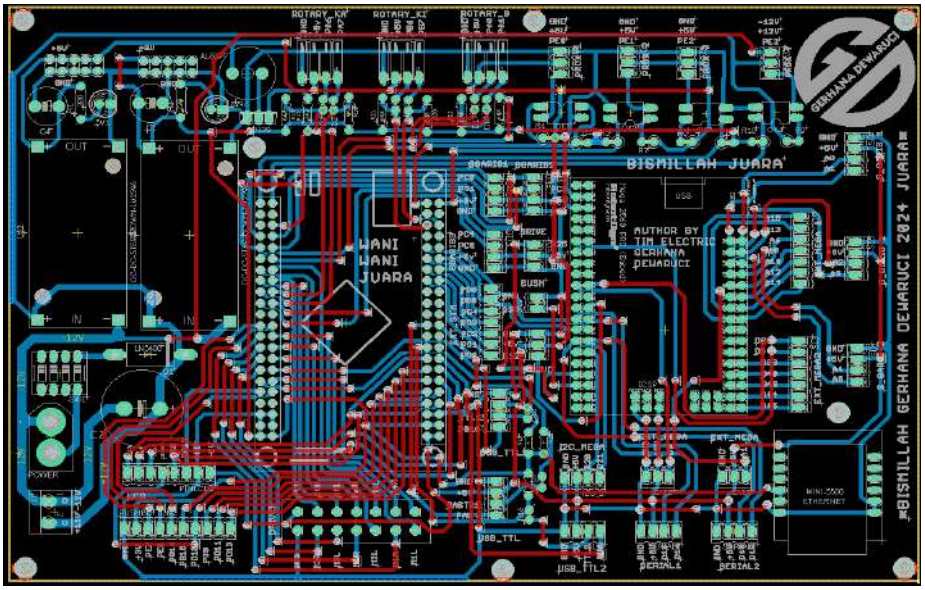
4.1.2 Hasil Perancangan Sistem Elektrik

Sistem elektrik merupakan komponen krusial dalam robot karena berfungsi

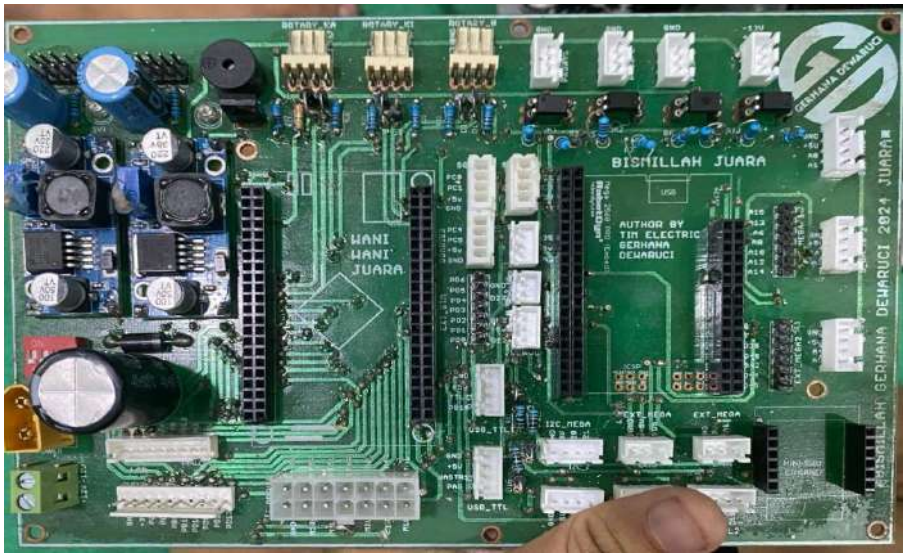
sebagai penghubung antara sensor, aktuator, dan unit pemrosesan. Sistem ini dirancang agar robot mampu bergerak secara otomatis, mendeteksi kondisi lingkungan, serta mengambil keputusan berdasarkan data yang diperoleh dari sensor. Setelah tahap perancangan selesai, sistem elektrik kemudian direalisasikan dalam bentuk fisik dan dilakukan pengujian untuk memastikan seluruh rangkaian beroperasi sesuai dengan fungsi yang diharapkan.



Gambar 4. 2 Rangkaian Skematik *Main* PCB Robot Penyerang
(Penulis, 2026)



Gambar 4. 3 Hasil *routing* PCB Robot Penyerang
(Penulis, 2026)



Gambar 4. 4 *Main* PCB Robot Penyerang
(Penulis, 2026)

Gambar 4.2 menampilkan rancangan Printed Circuit Board (PCB) yang digunakan untuk mengintegrasikan mikrokontroler, driver motor, sensor, serta komponen pendukung lainnya. Desain PCB ini dibuat untuk meningkatkan kerapian, kestabilan sistem, serta memudahkan proses perakitan dan pemeliharaan.

Secara keseluruhan, sistem elektrik terdiri dari kamera 360 sebagai sumber input utama yang diproses oleh laptop untuk pengambilan keputusan berbasis visual. Mikrokontroler STM32F407VGT6 berperan sebagai pusat kendali yang mengatur motor dan aktuator, serta menerima data dari sensor rotary encoder,

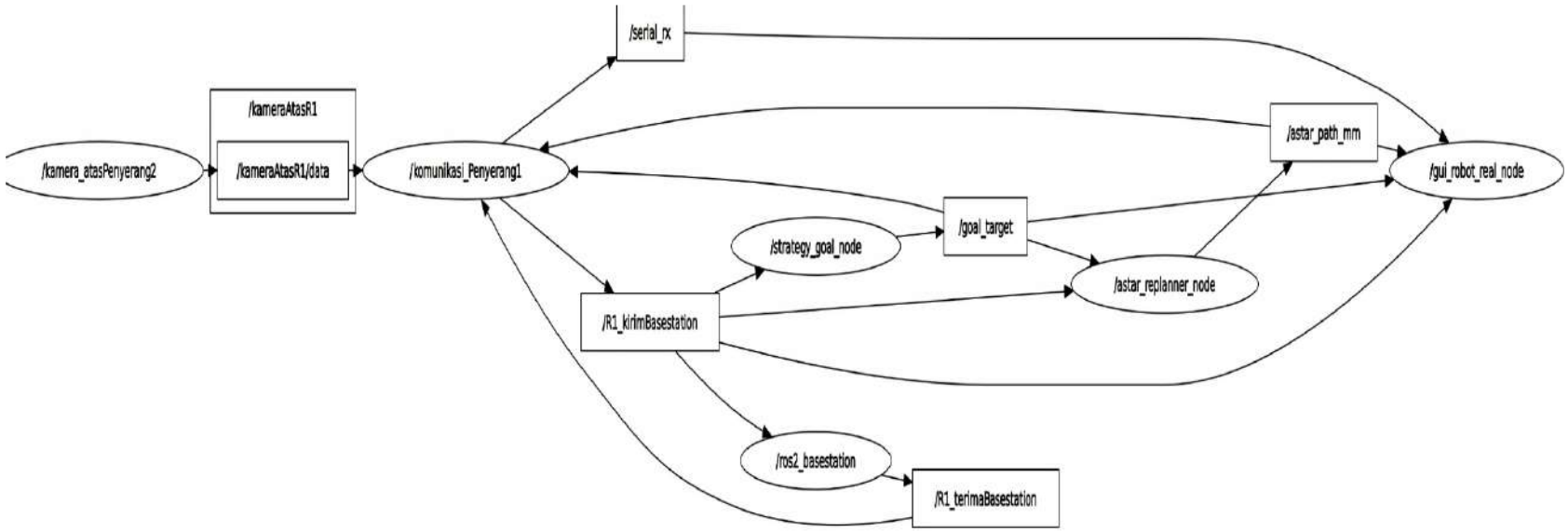
sensor gyroscope HWT101CT, dan sensor proximity. Mikrokontroler tersebut mengendalikan empat motor DC PG-45 sebagai penggerak utama melalui driver IBT-2. Selain itu, dua motor DC PG-36 digunakan pada mekanisme penggiring bola, sedangkan solenoid kicker dikontrol langsung oleh STM32 untuk melakukan penendangan.

Berdasarkan hasil pengujian, seluruh komponen sistem elektrik dapat berfungsi sesuai dengan yang diharapkan. Motor mampu merespons pengaturan arah dan kecepatan dengan baik, kamera dapat mendeteksi objek secara akurat, dan sensor proximity mampu mengidentifikasi posisi bola dengan tepat. Selain itu, solenoid kicker menghasilkan respons yang cepat dan gaya tendang yang cukup kuat. Dengan demikian, sistem yang dirancang mampu mendukung kinerja robot secara responsif dan stabil saat beroperasi di lapangan.

4.1.3 Hasil Perancangan ROS 2 *Humble*

Penelitian ini dibangun menggunakan arsitektur Robot Operating System 2 (ROS 2) Humble yang menerapkan komunikasi antar *node* melalui mekanisme *topic-based communication*. Setiap *node* memiliki fungsi yang berbeda, seperti akuisisi data sensor, komunikasi dengan *basestation*, Melalui mekanisme *publisher-subscriber*, setiap *node* dapat saling bertukar informasi secara terdistribusi sehingga sistem menjadi lebih modular dan mudah dikembangkan. Selain itu, penggunaan middleware berbasis Data Distribution Service (DDS) pada ROS 2 memungkinkan komunikasi data berlangsung dengan latensi rendah sehingga sesuai untuk kebutuhan robot sepak bola.

Alur komunikasi antar *node* divisualisasikan menggunakan RQT Graph, yang menampilkan hubungan antara seluruh *node* beserta *topic* yang digunakan selama sistem beroperasi. Visualisasi ini digunakan untuk memverifikasi bahwa setiap *node* telah terhubung sesuai dengan rancangan sistem, mulai dari akuisisi data kamera omnidirectional, komunikasi dengan *basestation*, penerimaan data serial, penentuan *goal target*, proses *path planning* menggunakan algoritma A*, hingga visualisasi hasil perencanaan lintasan. Dengan arsitektur tersebut, seluruh data dari sensor dan masukan pengguna dapat diproses secara sinkron selama proses navigasi, sehingga sistem mampu melakukan navigasi secara adaptif terhadap perubahan posisi robot lawan *(opponent)* selama pertandingan.



Gambar 4. 5 ROS 2 Flow Graph
(Penulis, 2026)

Gambar 4.5 menunjukkan alur komunikasi antar *node* pada sistem ROS 2 yang digunakan dalam penelitian ini. Proses diawali dari node /kameraAtasPenyerang1 yang berfungsi sebagai sumber data citra dari kamera omnidirectional. Data hasil deteksi kemudian dipublikasikan oleh node /kameraAtasR1 melalui topik /kameraAtasR1/data, yang berisi informasi hasil pengolahan citra seperti posisi robot lawan (*opponent*) dan objek lain yang diperlukan dalam sistem navigasi.

Data tersebut selanjutnya diteruskan ke node /komunikasi_Penyerang1, yang berperan sebagai pusat komunikasi antar modul. Node ini mendistribusikan data hasil deteksi ke beberapa *node* lain yang memerlukan informasi tersebut, termasuk modul komunikasi serial, *basestation*, serta sistem *path planning*.

Pada sisi komunikasi, node /serial_rx menerima data dari sistem komunikasi serial robot, sedangkan node /R1_kirimBasestation mengirimkan informasi kondisi robot menuju *basestation*. Selanjutnya, data dipublikasikan melalui node /ros2_basestation dan diterima kembali oleh node /R1_terimaBasestation sebagai media pertukaran informasi antara robot dan *basestation*.

Informasi dari sistem komunikasi kemudian digunakan oleh node /strategy_goal_node untuk menentukan posisi tujuan (*goal target*) yang akan dicapai robot. Posisi tujuan tersebut dipublikasikan melalui topik /goal_target dan diteruskan ke node /astar_replanner_node sebagai masukan dalam proses perencanaan lintasan.

Node /astar_replanner_node menerapkan algoritma A* untuk menghasilkan lintasan menuju *goal target*. Apabila terjadi perubahan posisi robot lawan (*opponent*) yang memengaruhi jalur pergerakan robot, node ini akan melakukan *replanning* sehingga lintasan yang dihasilkan dapat diperbarui berdasarkan kondisi permainan selama proses navigasi. Hasil perencanaan lintasan kemudian dipublikasikan melalui topik /astar_path_mm.

Seluruh informasi yang meliputi data komunikasi serial, posisi tujuan, serta lintasan hasil *path planning* diterima oleh node /gui_robot_real_node. Node ini berfungsi sebagai antarmuka visual untuk menampilkan kondisi robot, posisi *goal target*, dan lintasan hasil algoritma A* sehingga memudahkan proses pemantauan

selama sistem berjalan.

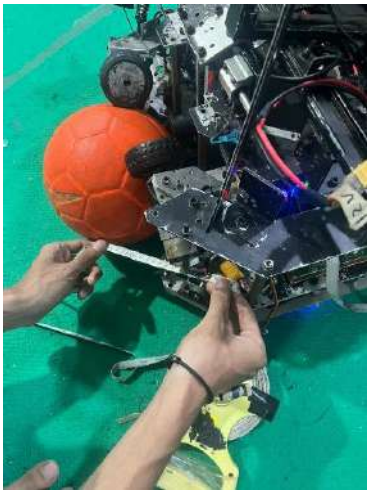
Dengan arsitektur komunikasi tersebut, setiap *node* dapat saling bertukar informasi melalui mekanisme *topic* pada ROS 2 sehingga proses deteksi, komunikasi, penentuan tujuan, perencanaan lintasan, hingga visualisasi sistem dapat berlangsung secara terintegrasi selama proses navigasi.

4.2. Pengujian *input* dan *output*

Pengujian sensor-sensor pada penelitian ini dilakukan beberapa langkah sebagai berikut :

4.2.1 Pengujian Kinerja Sensor Proximity

Sensor proximity pada penelitian ini dimanfaatkan untuk mendeteksi keberadaan bola yang berada di sekitar area mekanisme pengiring robot. Peran sensor ini krusial dalam memastikan bahwa bola benar-benar berada dalam jangkauan penggiring serta tidak terlepas saat robot melakukan pergerakan. Sensor bekerja dengan membaca jarak kedekatan bola terhadap robot secara real-time dan menghasilkan sinyal ketika bola telah memasuki area pengiring. Untuk menilai tingkat akurasi sensor, dilakukan pengujian dengan cara membandingkan hasil pembacaan jarak dari sensor proximity terhadap pengukuran manual yang menggunakan penggaris atau meteran sebagai acuan. Pengujian dilaksanakan pada beberapa variasi jarak dekat guna merepresentasikan kondisi nyata dalam pertandingan. Gambar 4.3 dan Gambar 4.4 menunjukkan skema pengujian secara visual saat bola didekatkan ke mekanisme pengiring robot. Sementara itu, hasil pembacaan sensor beserta perbandingannya dengan jarak aktual disajikan pada Tabel 4.1. Berdasarkan pengujian tersebut, dapat dilakukan analisis terhadap tingkat akurasi dan konsistensi sensor proximity dalam mendeteksi keberadaan bola, sehingga sistem mampu bekerja secara responsif dan menjaga bola tetap berada dalam kendali robot selama pergerakan.



Gambar 4. 6 Pengujian sensor Proximity
(Penulis, 2026)

Gambar 4. 7 Pembacaan Sensor Proximity
(Penulis, 2026)

Tabel 4. 1 Hasil pengujian sensor proximity

Pengujian Ke -	Jarak Bola (cm)	Hasil deteksi sensor proximity
1	4	Terdeteksi
2	5	Terdeteksi
3	6	Terdeteksi
4	7	Terdeteksi
5	8	Terdeteksi
6	9	Terdeteksi
7	10	Terdeteksi
8	11	Terdeteksi
9	12	Tidak Terdeteksi
10	13	Tidak Terdeteksi

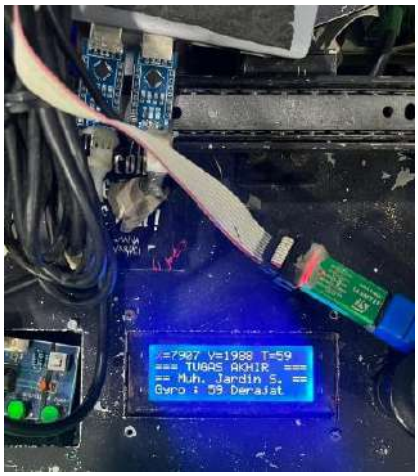
Tabel 4.1 menyajikan hasil pengujian sensor proximity dalam mendeteksi keberadaan bola berdasarkan variasi jarak antara bola dan sensor. Berdasarkan 10 kali pengujian yang dilakukan, diperoleh bahwa sensor mampu mendeteksi bola secara optimal pada rentang jarak 4 cm hingga 11 cm, yang ditunjukkan dengan status “Terdeteksi”. Sementara itu, pada jarak 12 cm dan 13 cm, sensor tidak lagi memberikan respons deteksi, yang mengindikasikan bahwa batas maksimum jangkauan efektif sensor proximity berada di sekitar 11 cm.

Hasil tersebut menunjukkan bahwa sensor memiliki kinerja yang andal pada jarak dekat, sehingga sangat sesuai digunakan sebagai sistem konfirmasi keberadaan bola di dalam area penggiring robot. Namun demikian, sensor ini tidak efektif untuk mendeteksi objek yang berada di luar jangkauan tersebut.

4.2.2 Pengujian Kinerja Sensor HWT101CT

Pada penelitian tugas akhir ini, sensor *gyroscope* HWT101CT digunakan sebagai komponen utama untuk menentukan orientasi atau sudut arah robot. Sensor ini berfungsi dalam mendeteksi perubahan arah secara *real-time* selama robot bergerak. Untuk menilai tingkat keakuratan sensor HWT101CT, dilakukan pengujian dengan membandingkan hasil pembacaan sensor terhadap alat referensi berupa kompas digital pada smartphone, yang digunakan sebagai acuan nilai sudut sebenarnya.

Proses pengujian dilakukan dengan cara memutar robot ke beberapa arah tertentu, kemudian mencatat nilai sudut yang dihasilkan oleh sensor serta kompas. Gambar 4.5 dan Gambar 4.6 menampilkan ilustrasi skema pengujian yang dilakukan, sedangkan hasil perbandingan antara pembacaan sensor HWT101CT dan kompas disajikan secara sistematis pada Tabel 4.2. Pengujian ini bertujuan untuk memastikan bahwa sensor mampu memberikan data sudut yang akurat dan stabil, sehingga dapat menduung kinerja sistem navigasi robot secara keseluruhan.



Gambar 4. 8 Pembacaan Gyroscope
(Penulis, 2026)

Gambar 4. 9 Pembacaan Kompas
(Penulis, 2026)

Tabel 4. 2 Hasil pengujian gyroscope HWT

Pengujian Ke -	Kompas (derajat)	Gyro HWT101CT (derajat)	Error (%)
1	0	0	0
2	30	29	3,33
3	60	59	1,6
4	90	90	0
5	120	120	0

Tabel 4.2 Hasil pengujuan gyroscope HWT (Lanjutan)

Pengujian Ke -	Kompas (derajat)	Gyro HWT101CT (derajat)	Error (%)
6	150	150	0
7	180	179	0,55
8	210	210	0
9	240	240	0
10	270	270	0
11	300	300	0
12	330	331	0,30
13	360	359	0,27
Rata – Rata Nilai Error			0,46

Tabel 4.2 menyajikan hasil pengujian akurasi sensor gyroscope HWT101CT dalam menentukan sudut orientasi robot, dengan membandingkan nilai pembacaan sensor terhadap nilai referensi yang diperoleh dari kompas digital. Pengujian dilakukan pada 13 variasi sudut dengan interval 30°, dimulai dari 0° hingga 360°. Kolom pertama menunjukkan nilai sudut referensi dari kompas, sedangkan kolom kedua menampilkan hasil pembacaan sensor gyroscope pada sudut yang sama.

Kolom ketiga memuat nilai error dalam persentase yang dihitung berdasarkan selisih absolut antara nilai sensor dan nilai referensi, kemudian dibagi dengan nilai referensi dan dikalikan 100%. Berdasarkan data yang diperoleh, error terbesar terjadi pada sudut 30°, yaitu sebesar 3,33%. Sementara itu, pada beberapa sudut seperti 0°, 90, 120, 150, 210°, 240°, 270, dan 300 sensor menunjukkan akurasi sempurna dengan nilai error 0%.

Secara keseluruhan, rata-rata error pembacaan sensor HWT101CT sebesar 0,46%, yang menunjukkan bahwa sensor memiliki tingkat akurasi dan kestabilan yang sangat baik dalam mendeteksi orientasi robot. Nilai ini sudah memenuhi kebutuhan aplikasi robotika, khususnya dalam sistem navigasi dan penentuan arah gerak robot baik pada kondisi simulasi maupun implementasi nyata.

4.2.3 Pengujian Kinerja Solenoid Kicker

Pengujian solenoid kicker pada Gambar 4.6 dilakukan untuk mengevaluasi kinerja mekanisme penendang pada robot dalam merespons sinyal aktivasi berbasis

waktu. Sistem kicker memanfaatkan aktuator solenoid yang dikendalikan melalui driver bertegangan tinggi, dengan sumber tegangan input berada pada kisaran 100 hingga 400 VDC. Aktivasi solenoid dilakukan dengan memberikan sinyal pulsa PWM yang memiliki variasi durasi waktu aktif (*on-time*) antara 5 ms hingga 20 ms.

Tujuan dari pengujian ini adalah untuk menganalisis pengaruh variasi durasi sinyal terhadap kekuatan tendangan yang dihasilkan, serta memastikan bahwa tegangan keluaran dari driver mampu menggerakkan solenoid secara optimal. Setiap percobaan dilakukan dengan mengamati dan mencatat respons solenoid terhadap perubahan durasi sinyal serta variasi tegangan yang diterapkan.

Gambar 4. 10 Pembacaan Tegangan Output pada VDC
(Penulis, 2026)

Tabel 4. 3 Hasil pengujian solenoid kicker

Percobaan ke -	Waktu (ms)	Tegangan Output driver (volt)
1	6	136.3
2	7	153.4
3	8	182.4
4	10	206.5
5	13	274.6
6	15	313.4
7	17	387.2
8	20	403.2

Tabel 4.3 menyajikan hasil pengujian hubungan antara durasi waktu aktif sinyal (dalam milidetik) terhadap tegangan output yang dihasilkan oleh driver

solenoid kicker. Pengujian dilakukan pada rentang waktu 6 ms hingga 20 ms dengan variasi durasi yang meningkat secara bertahap. Berdasarkan data yang diperoleh, terlihat bahwa peningkatan waktu aktif sinyal berbanding lurus dengan kenaikan tegangan output driver.

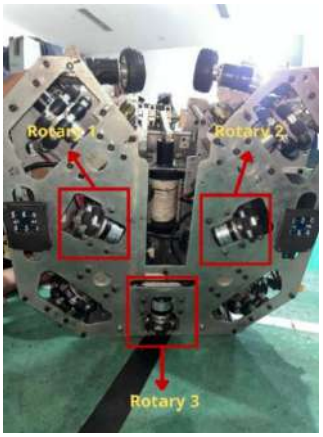
Sebagai contoh, pada durasi 6 ms, tegangan output yang dihasilkan sebesar 136,3 VDC. Ketika durasi dinaikkan menjadi 10 ms, tegangan meningkat menjadi 206,5 VDC. Nilai tegangan terus bertambah hingga mencapai 403,2 VDC pada durasi maksimum 20 ms. Pola ini menunjukkan bahwa sistem driver memiliki respons yang konsisten terhadap perubahan durasi sinyal, dengan kecenderungan peningkatan tegangan yang hampir linier.

Hasil pengujian ini menunjukkan bahwa pengaturan waktu aktif sinyal PWM dapat digunakan untuk mengontrol besar tegangan yang diberikan ke solenoid, sehingga kekuatan mekanisme penendang dapat diatur sesuai kebutuhan. Hal ini menjadi penting dalam mendukung performa robot, khususnya dalam menentukan kekuatan tendangan bola pada berbagai kondisi permainan.

4.2.4 Pengujian Kinerja Rotary Encoder

Pada penelitian ini, rotary encoder digunakan untuk mengukur perpindahan robot terhadap titik acuan awal (titik nol) sebagai salah satu sumber data odometri. Sensor ini bekerja dengan mendeteksi jumlah putaran roda selama robot bergerak. Data jumlah putaran tersebut kemudian dikonversi menjadi nilai jarak tempuh yang merepresentasikan pergerakan robot.

Dalam implementasinya, digunakan tiga buah rotary encoder yang dipasang dengan konfigurasi sudut 120° satu sama lain. Konfigurasi ini memungkinkan pengukuran pergerakan robot secara lebih akurat pada berbagai arah. Gambar 4.8 menunjukkan tata letak perangkat keras sensor rotary encoder yang terpasang dan dikopel langsung dengan roda omni. Dan Gambar 4.9 merupakan pembacaan dari rotary encoder.



Gambar 4. 11 Tata letak *rotary encoder*
(Penulis, 2026)

Gambar 4. 12 Pembacaan Rotary encoder
(Penulis, 2026)

Tabel 4. 4 Hasil pengukuran Rotary Encoder

Pengujian Ke-	Data Meteran (mm)	Hasil Pengukuran Rotary (mm)	Error (%)
1	500	506	**1,20**
2	1000	1007	**0,70**
3	1500	1503	**0,20**
4	2000	2016	**0,80**
5	2500	2505	**0,20**
6	3000	3052	**1,73**
7	3500	3510	**0,29**
8	4000	4050	**1,25**
Rata-rata Nilai Error			**0.80%**

Berdasarkan hasil pengujian pada Tabel 4.4, pembacaan jarak menggunakan rotary encoder menunjukkan tingkat akurasi yang baik dengan nilai rata-rata error sebesar 0,80% terhadap hasil pengukuran menggunakan meteran. Error terbesar terjadi pada pengujian ke-6, yaitu sebesar 1,73%, sedangkan error terkecil diperoleh pada pengujian ke-3 dan ke-5, yaitu 0,20%. Perbedaan hasil pengukuran tersebut dipengaruhi oleh beberapa faktor, seperti toleransi mekanis pada pemasangan rotary encoder, kemungkinan terjadinya selip roda (*wheel slip*), serta ketidaksempurnaan permukaan lintasan. Meskipun demikian, nilai error yang diperoleh masih relatif kecil, sehingga rotary encoder dinilai mampu memberikan

informasi perpindahan robot dengan tingkat akurasi yang memadai dan dapat digunakan sebagai sumber data odometri dalam sistem navigasi robot sepak bola.

4.2.5 Pengujian Kinerja Motor PG-45 Penggerak

Pada penelitian tugas akhir ini, motor DC PG-45 digunakan sebagai aktuator utama untuk menggerakkan robot ke berbagai arah. Motor ini dipilih karena memiliki torsi yang relatif besar serta kecepatan putar yang stabil, sehingga sesuai untuk diaplikasikan pada sistem penggerak robot beroda omni yang memerlukan kemampuan manuver yang tinggi. Konfigurasi ini bertujuan untuk mengurangi kehilangan daya serta meningkatkan efisiensi dalam pergerakan robot. Dapat dilihat pada Gambar 4.10 pengujian motor penggerak.

Gambar 4. 13 Pengujian motor PG-45 Penggerak
(Penulis, 2026)

Tabel 4. 5 Hasil pengukuran Motor PG45

Pengujian Ke -	Tegangan Input (VDC)	RPM Datasheet (RPM)	Hasil Tachometer (RPM)	Error (%)
1	24	500	505.2	1.04
2	24	500	490.2	1.96
3	24	500	513.6	2.72
4	24	500	514.0	2.80

Tabel 4. 5 Hasil pengujian motor PG-45 Penggerak (Lanjutan)

Pengujian Ke -	Tegangan Input (VDC)	RPM Datasheet (RPM)	Hasil Tachometer (RPM)	Error (%)
5	24	500	507.9	1.58
6	24	500	488.8	2.24
7	24	500	514.3	2.86
8	24	500	514.9	2.98
9	24	500	488.6	2.28
10	24	500	490.0	2.00
Rata – Rata Nilai Error				**2.24**

Tabel 4.5 menyajikan hasil pengujian kecepatan putar motor DC PG-45 berdasarkan pembacaan menggunakan tachometer. Pada pengujian ini, tegangan input dijaga konstan sebesar 24 VDC, dengan nilai acuan kecepatan motor sebesar 500 RPM sesuai dengan spesifikasi sistem.

Kolom “Hasil Tachometer (RPM)” menunjukkan kecepatan aktual motor pada setiap percobaan. Dari data yang diperoleh, terlihat bahwa nilai kecepatan motor mengalami variasi, di mana sebagian hasil mendekati nilai referensi, sementara beberapa lainnya menunjukkan deviasi yang cukup signifikan. Salah satu deviasi terbesar terjadi pada pengujian ke-2 dengan nilai 545.0 RPM, serta pada pengujian ke-9 dan ke-10 yang masing-masing mencapai 537.1 RPM dan 536.3 RPM.

Kolom “Error (%)” dihitung berdasarkan selisih antara nilai referensi dan hasil pengukuran, kemudian dinyatakan dalam bentuk persentase. Nilai error terkecil sebesar 1.04% terjadi pada pengujian ke-1, sedangkan error terbesar sebesar 9.00% terjadi pada pengujian ke-2. Secara keseluruhan, rata-rata error yang diperoleh adalah sebesar 4.07%, yang menunjukkan bahwa sistem penggerak motor masih mengalami variasi kecepatan yang cukup besar terhadap nilai target.

4.2.6 Pengujian Kinerja Motor PG-36 Penggiring

Pada penelitian tugas akhir ini, motor DC PG-36 digunakan sebagai aktuator utama pada sistem penggiring bola robot. Motor ini dipilih karena memiliki dimensi

yang lebih kompak, namun tetap mampu menghasilkan torsi yang cukup untuk menggerakkan bola secara konsisten sesuai arah yang diinginkan. Selain itu, kestabilan kecepatan putar motor PG-36 memungkinkan bola tetap berada dalam kendali mekanisme penggiring tanpa mudah kehilangan arah saat robot bergerak.

Motor ini dihubungkan langsung dengan mekanisme penggiring melalui sistem transmisi sederhana guna memperoleh respons gerakan yang lebih presisi dan meminimalkan keterlambatan (*delay*). Dengan konfigurasi tersebut, kinerja robot dalam mempertahankan penguasaan bola selama proses manuver di lapangan dapat ditingkatkan. Gambar 4.11 merupakan pengujian untuk motor penggiring.

Gambar 4. 14 Pengujian Motor PG 36 Penggiring
(Penulis, 2026)

Tabel 4. 6 Hasil pengukuran Motor PG36

Pengujian Ke -	Tegangan Input (VDC)	RPM Datasheet (RPM)	Hasil Tachometer (RPM)	Error (%)
1	24	600	603.9	0.65
2	24	600	596.0	0.67
3	24	600	601.9	0.32
4	24	600	603.6	0.60
5	24	600	597.0	0.50
6	24	600	604.7	0.78

Tabel 4.6 Hasil pengukuran PG36 (Lanjutan)

Pengujian Ke -	Tegangan Input (VDC)	RPM Datasheet (RPM)	Hasil Tachometer (RPM)	Error (%)
7	24	600	597.3	0.45
8	24	600	601.6	0.27
9	24	600	598.7	0.22
10	24	600	598.9	0.18
Rata – Rata Nilai Error				**0.46**

Tabel 4.6 menyajikan hasil pengujian performa motor DC PG-36 yang digunakan sebagai sistem penggiring bola pada robot. Pengujian dilakukan dengan memberikan tegangan input konstan sebesar 24 VDC dan nilai acuan kecepatan putaran motor sebesar 600 RPM sesuai spesifikasi sistem. Setiap data menunjukkan hasil pengukuran kecepatan aktual motor menggunakan tachometer.

Berdasarkan hasil pengujian, nilai RPM yang diperoleh berada sangat dekat dengan nilai referensi 600 RPM. Selisih antara nilai referensi dan hasil pengukuran dihitung sebagai error dalam bentuk persentase. Sebagai contoh, pada pengujian ke-1 diperoleh nilai 603.9 RPM dengan error sebesar 0.65%, sedangkan pada pengujian ke-10 nilai RPM sebesar 598.9 dengan error hanya 0.18%.

Secara keseluruhan, nilai error pada seluruh pengujian berada dalam rentang yang kecil dan relatif stabil. Hal ini menunjukkan bahwa motor DC PG-36 memiliki performa yang konsisten dalam mempertahankan kecepatan putar sesuai dengan nilai target. Nilai rata-rata error yang diperoleh sebesar 0.46% semakin menegaskan bahwa motor ini memiliki tingkat akurasi yang tinggi. Dengan performa tersebut, motor PG-36 sangat sesuai digunakan pada sistem penggiring bola yang membutuhkan kestabilan dan ketepatan kecepatan, sehingga mampu mendukung pengendalian bola secara optimal dalam pergerakan robot.

4.3. Hasil Pengujian Kamera dengan Sistem Deteksi Berbasis YOLOv5s

Pengujian sistem deteksi berbasis YOLOv5s pada penelitian ini dilakukan melalui beberapa tahapan untuk mengevaluasi kemampuan model dalam mendeteksi objek secara akurat dan konsisten.

4.3.1 Pengujian Kamera OmniVision 360

Kamera merupakan salah satu sensor utama dalam sistem persepsi robot, khususnya untuk mendukung proses deteksi objek secara visual. Pada penelitian ini digunakan kamera omnivision 360° yang mampu menangkap citra lingkungan secara menyeluruh di sekitar robot. Kamera ini dipasang pada bagian atas robot untuk memperoleh bidang pandang penuh, sehingga robot dapat mendeteksi objek penting seperti bola dan robot lawan dari berbagai arah tanpa keterbatasan sudut pandang. Dengan cakupan pandangan yang luas, kamera ini sangat sesuai untuk aplikasi robotika dinamis seperti robot sepak bola.

Data visual yang diperoleh kemudian diproses menggunakan bahasa pemrograman Python pada sistem operasi Ubuntu 22.04. Untuk meningkatkan efisiensi komputasi, citra yang ditangkap diolah dengan resolusi tertentu dan diproses secara kontinu. Sistem deteksi berbasis YOLOv5s digunakan untuk mengenali objek seperti bola dan robot lawan, sehingga memungkinkan robot memperoleh informasi posisi objek secara cepat dan akurat.

Dengan pemanfaatan kamera omnivision 360° serta pengolahan citra yang terstruktur, robot diharapkan memiliki kemampuan visual yang optimal dalam mendeteksi dan melacak objek di sekitarnya. Hal ini sangat penting untuk meningkatkan respons robot terhadap pergerakan bola maupun lawan, serta mendukung pengambilan keputusan selama proses navigasi. Contoh hasil tangkapan kamera dapat dilihat pada Gambar 4.12.

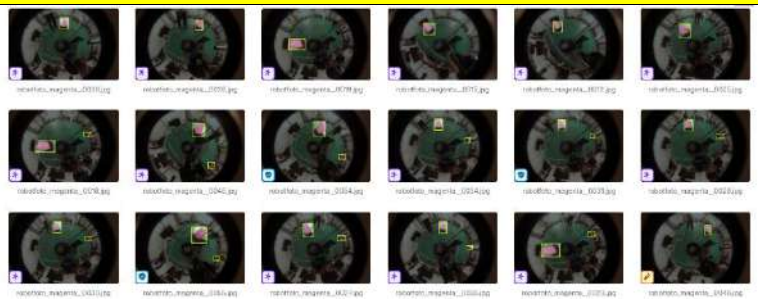
Gambar 4. 15 Tampilan dari kamera OmniVision 360
(Penulis, 2026)

4.3.2 Pengambilan dan Pengumpulan *Datasets*

Dataset merupakan kumpulan data yang disusun secara sistematis untuk keperluan analisis, pelatihan model, maupun evaluasi sistem dalam suatu penelitian. Pada penelitian ini, dataset yang digunakan berupa citra yang merepresentasikan dua objek utama, yaitu bola dan robot lawan. Data tersebut dikelompokkan berdasarkan karakteristik visual seperti warna dan bentuk. Objek bola diidentifikasi berdasarkan warna dominan dan bentuk bulat, sedangkan robot lawan diklasifikasikan berdasarkan bentuk fisik serta ciri pembeda tertentu, seperti warna atau atribut khusus yang membedakannya dari robot tim sendiri.

Setiap gambar dalam dataset telah melalui proses pelabelan secara manual untuk menentukan kelas objek yang terdapat di dalamnya, sehingga dapat digunakan dalam pelatihan model deteksi objek berbasis YOLOv5s. Dataset ini bertujuan untuk melatih sistem agar mampu mengenali dan membedakan objek bola serta robot lawan secara akurat dalam kondisi lingkungan yang bervariasi. Oleh karena itu, pemilihan data mempertimbangkan variasi posisi, pencahayaan, dan latar belakang. Klasifikasi serta contoh kelas dataset dapat dilihat pada Tabel 4.7.

Tabel 4. 7 *Class* dan jumlah *datasets*

Class	Gambar d*atasets*	Jumlah *datasets*
Robot		3193
Bola		2051
Jumlah total datasets		5244

Pada penelitian ini, dataset yang digunakan terdiri dari dua kelas utama, yaitu robot dan bola, dengan total keseluruhan sebanyak 5244 citra. Dari jumlah tersebut, sebanyak 3193 citra merupakan data untuk kelas robot, sedangkan 2051 citra lainnya merupakan data untuk kelas bola. Dataset ini kemudian dibagi menjadi beberapa bagian, yaitu data pelatihan (*train set*), data validasi (*validation set*), dan data pengujian (*test set*) untuk mendukung proses pelatihan dan evaluasi model.

Pembagian dataset dilakukan dengan proporsi sekitar 70% untuk data pelatihan, 20% untuk data validasi, dan 10% untuk data pengujian. Tujuan dari pembagian ini adalah untuk memastikan model memiliki jumlah data yang cukup untuk mempelajari karakteristik visual dari masing-masing objek, sekaligus menyediakan data yang memadai untuk menguji performa dan kemampuan generalisasi model.

Dengan jumlah dataset yang cukup besar dan variasi data yang beragam, sistem diharapkan mampu mengenali objek bola dan robot lawan secara akurat dalam berbagai kondisi lingkungan. Pendekatan ini juga bertujuan untuk menjaga konsistensi hasil deteksi di seluruh area lapangan, sehingga kinerja sistem tetap optimal selama robot beroperasi.

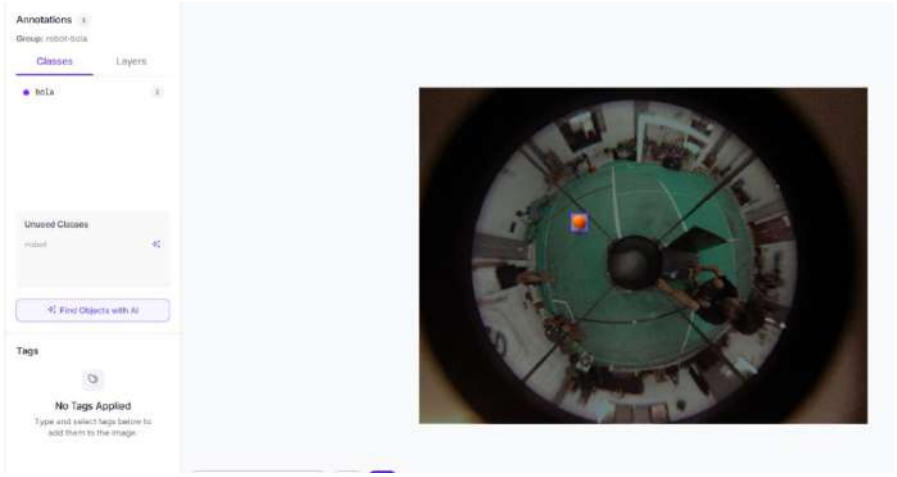
4.3.3 Pelabelan *Datasets*

Tahap pelabelan pada penelitian ini merupakan proses yang sangat penting dalam penyusunan dataset, karena berpengaruh langsung terhadap kualitas data yang digunakan dalam pelatihan model deteksi objek. Setiap citra yang diperoleh dari kamera diberi anotasi berdasarkan jenis objek yang terdapat di dalamnya, sehingga sistem dapat melakukan identifikasi dan klasifikasi secara tepat selama proses pelatihan.

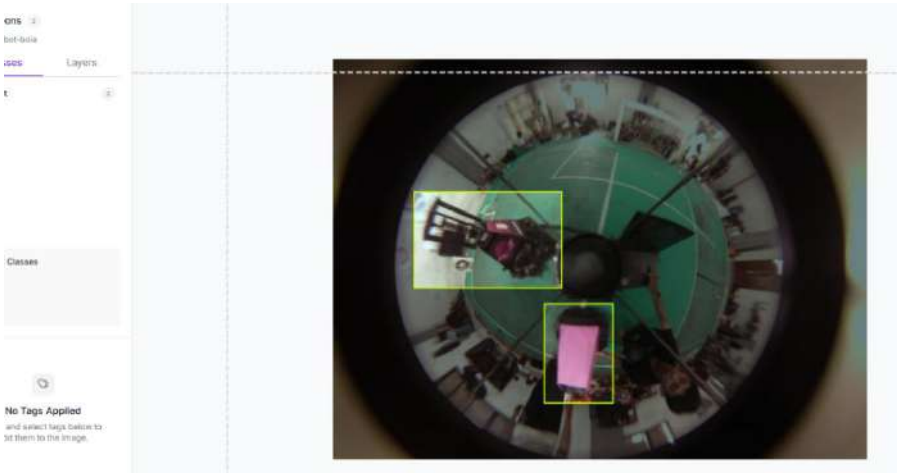
Pada penelitian ini, pelabelan dilakukan menggunakan metode *bounding box*, yaitu dengan memberikan kotak pembatas pada area objek yang akan dideteksi untuk menunjukkan posisi dan ukuran objek dalam citra. Setiap objek yang telah diberi batas kemudian diklasifikasikan sesuai dengan kelasnya, yaitu robot sebagai class 1 dan bola sebagai *class* 0.

Untuk mempermudah proses pelabelan secara manual, digunakan aplikasi Roboflow yang mendukung anotasi gambar secara interaktif serta menyediakan hasil ekspor dalam format yang kompatibel dengan YOLOV5S. Penggunaan

Roboflow memungkinkan proses pelabelan dilakukan secara lebih efisien dan konsisten pada seluruh dataset, baik untuk data pelatihan maupun validasi. Hasil anotasi ini kemudian digunakan sebagai dasar dalam melatih model deteksi agar mampu mengenali objek robot lawan dan bola secara akurat pada kondisi lingkungan yang dinamis. Contoh hasil pelabelan bounding box ditunjukkan pada Gambar 4.16 untuk robot dan Gambar 4.17 untuk bola.



Gambar 4. 16 Proses pelabelan class bola
(Penulis, 2026)



Gambar 4. 17 Proses pelabelan class robot
(Penulis, 2026)

Hasil dari proses pelabelan menghasilkan file anotasi yang memuat informasi koordinat posisi objek pada citra, meliputi nilai batas kotak pembatas pada sumbu horizontal (X1 dan X2) serta vertikal (Y1 dan Y2). Data tersebut disimpan dalam format teks yang terstruktur, sehingga memudahkan proses

pengolahan dan penggunaan pada tahap selanjutnya, seperti pelatihan model deteksi objek.

Pada penelitian ini, proses anotasi dilakukan secara teliti dan konsisten untuk memastikan kualitas dataset yang dihasilkan tetap terjaga dan mampu mendukung kinerja sistem secara optimal. Contoh hasil anotasi citra dari proses pelabelan ditampilkan pada Tabel 4.8 untuk objek robot dan Tabel 4.9 untuk objek bola.

Tabel 4. 8 Hasil anotasi *class* bola

No	Class	X1	Y1	X2	Y2
1	Bola (0)	0.491406	0.208333	0.019531	0.028125
2	Bola (0)	0.488281	0.210416	0.010937	0.020833
3	Bola (0)	0.498437	0.211458	0.014062	0.019791
4	Bola (0)	0.510937	0.214583	0.013281	0.019791
5	Bola (0)	0.516406	0.214583	0.0125	0.017708
6	Bola (0)	0.5375	0.219791	0.011718	0.01875
7	Bola (0)	0.5421875	0.219791	0.016406	0.026041
8	Bola (0)	0.5585937	0.227083	0.023437	0.027083
9	Bola (0)	0.5828125	0.238541	0.010937	0.019791
10	Bola (0)	0.5609375	0.2291667	0.02109375	0.0302083

Tabel 4. 9 Hasil anotasi *class* robot

No	Class	X1	Y1	X2	Y2
1	Robot (1)	0.438281	0.245833	0.071875	0.146875
2	Robot (1)	0.718752	0.386458	0.062533	0.059375
3	Robot (1)	0.489843	0.241666	0.083593	0.154167
4	Robot (1)	0.743753	0.481253	0.067187	0.066666
5	Robot (1)	0.503125	0.246875	0.079687	0.155208
6	Robot (1)	0.747656	0.501041	0.067187	0.057297
7	Robot (1)	0.503125	0.252083	0.082812	0.163547
8	Robot (1)	0.747656	0.501041	0.079687	0.055208
9	Robot (1)	0.747656	0.508333	0.067187	0.054166
10	Robot (1)	0.5203125	0.262533	0.0882812	0.173958

Setelah proses pelabelan pada dataset citra selesai dilakukan, khususnya untuk objek bola, diperoleh file anotasi yang berisi informasi koordinat *bounding box* beserta label kelas objek. Anotasi tersebut, seperti yang ditampilkan pada Tabel 4.10 dan Tabel 4.11, selanjutnya diproses menjadi *ground-truth box* yang digunakan sebagai acuan dalam proses evaluasi hasil deteksi.

Ground-truth box kemudian dibandingkan dengan *predicted box* yang dihasilkan oleh model deteksi menggunakan pendekatan evaluasi berbasis *confusion matrix*. Dalam tahap evaluasi ini digunakan beberapa metrik utama, seperti *Intersection over Union* (IoU), *Precision*, *Average Precision* (AP), dan *Mean Average Precision* (mAP), untuk mengukur tingkat akurasi model dalam mendeteksi dan mengklasifikasikan objek.

Keempat metrik tersebut menjadi indikator penting dalam menilai kinerja sistem, khususnya dalam menghadapi variasi kondisi pencahayaan. Oleh karena itu, pengolahan dataset serta analisis hasil yang dilakukan secara sistematis dan teliti sangat berperan dalam memastikan sistem deteksi bola memiliki tingkat akurasi yang tinggi, andal, dan siap diimplementasikan pada kondisi nyata.

Sebelum proses pelatihan model deteksi objek, dataset dibagi menjadi tiga bagian, yaitu data pelatihan (*train*), validasi (*validation*), dan pengujian (*test*). Dari total 5.244 citra, sebanyak 70% (3.678 gambar) digunakan untuk pelatihan, 20% (1.041 gambar) untuk validasi, dan 10% (525 gambar) sebagai data uji. Data pelatihan berfungsi untuk melatih model mengenali objek bola dan robot, sementara data validasi digunakan untuk mengevaluasi proses pembelajaran dan mengoptimalkan parameter model. Adapun data uji digunakan untuk mengukur kemampuan generalisasi model terhadap data baru yang belum pernah dilihat sebelumnya. Pembagian ini bertujuan agar sistem tidak hanya mampu mengenali objek yang telah dilatih, tetapi juga tetap akurat dalam berbagai kondisi lingkungan saat diterapkan di lapangan. Tabel 4.10 merupakan pembagian jumlah datasets.

Tabel 4. 10 Tabel pembagian jumlah datasets

Class	*Train* (70%)	*Valid* (20 %)	*Test* (10 %)
Bola	1.436	410	205
Robot	2.235	639	319

- **Train (70%)** : Digunakan untuk melatih model YOLOV5S agar dapat mengenali objek dan menentukan posisi serta kelasnya berdasarkan data berlabel.
- **Valid (20%)** : Digunakan untuk mengevaluasi kinerja model selama pelatihan dan mencegah overfitting dengan memantau metrik seperti mAP.
- **Test (10%)** : Digunakan untuk menguji kemampuan model pada data baru yang belum pernah dilihat sebelumnya guna menilai performa secara keseluruhan.

4.3.4 Hasil *training* Model YOLOV5S

Proses pelatihan data dilaksanakan menggunakan platform Google Colab Pro. Adapun parameter yang diterapkan selama proses pelatihan ditunjukkan pada Tabel 4.11.

Tabel 4. 11 Hyperparameter Pelatihan Model YOLOV5S

Parameter	Value
Image Size	640
Batch Size	16
Epochs	150
Optimizer	SGD dan Adam

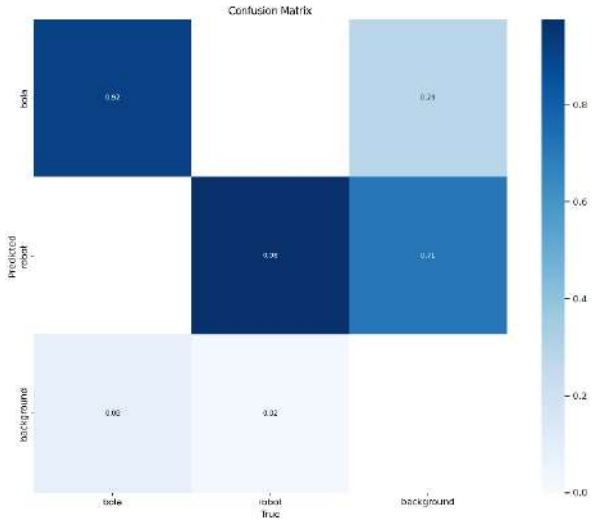
Berdasarkan konfigurasi parameter pada Tabel 4.11, proses training menunjukkan hasil optimal setelah melewati lebih dari 150 epoch, yang ditandai dengan kecenderungan menurunnya nilai *train/cls_loss* dan *val/cls_loss*. Untuk mengetahui tingkat kinerja model, dilakukan pengujian menggunakan beberapa metrik evaluasi sebagai berikut:

- ***Precision****:* Digunakan untuk mengetahui tingkat ketepatan model dalam memberikan prediksi positif, yaitu perbandingan antara prediksi positif yang benar dengan seluruh prediksi positif yang dihasilkan model.
- ***Recall (Sensitivity)***: Berfungsi untuk mengukur kemampuan model dalam mengenali seluruh data positif yang sebenarnya, sehingga menunjukkan seberapa efektif model dalam mendeteksi kasus positif.
- ***F1-Score***: Merupakan kombinasi harmonis antara nilai precision dan recall. Metrik ini sangat berguna terutama pada kondisi distribusi data yang tidak seimbang karena mempertimbangkan aspek ketepatan dan kelengkapan

prediksi.

4.3.4.1 Hasil Training Menggunakan Optimizer SGD

Proses training model YOLOv5s menggunakan optimizer *Stochastic Gradient Descent* (SGD) sebagai optimizer default pada YOLOV5S. Optimizer SGD merupakan metode optimasi yang digunakan untuk memperbarui bobot model berdasarkan nilai gradient loss pada setiap iterasi training. Proses pembaruan bobot dilakukan secara bertahap untuk meminimalkan nilai error sehingga model mampu menghasilkan prediksi yang lebih akurat.



Gambar 4. 18 *Confusion Matrix* SGD
(Penulis, 2026)

Berdasarkan hasil confusion matrix dan proses training model YOLOv5s, diperoleh nilai parameter evaluasi yang menunjukkan performa deteksi objek berjalan dengan baik. Optimizer ini digunakan untuk memperbarui bobot model selama proses training agar menghasilkan performa deteksi yang optimal.

Confusion matrix pada YOLOv5s digunakan sebagai visualisasi untuk melihat distribusi hasil prediksi model terhadap kelas bola, robot, dan background. Nilai yang ditampilkan pada confusion matrix merupakan nilai ternormalisasi sehingga tidak digunakan secara langsung sebagai nilai True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN) untuk menghitung accuracy seperti pada klasifikasi biner. Evaluasi performa model object detection pada penelitian ini selanjutnya mengacu pada precision, recall, F1-score, Average Precision (AP), dan mean Average Precision (mAP). Visualisasi confusion matrix

pada penelitian ini ditampilkan pada Gambar 4.18.

Tabel 4. 12 Pembacaan Nilai Berdasarkan *Confusion Matrix* SGD

Predicted / True	Bola	Robot	Background
Bola	0.92	0	0.29
Robot	0	0.98	0.71
Background	0.08	0.02	0

Pada confusion matrix yang ditunjukkan pada Tabel 4.12, terdapat tiga kategori utama, yaitu bola, robot dan *backgorund*.

- TP (*True Positive*) : objek berhasil terdeteksi dengan benar
- TN (*True Negative*) : data bukan objek berhasil dikenali dengan benar
- FP (*False Positive*) : kesalahan deteksi objek
- FN (*False Negative*) : objek gagal terdeteksi

Berdasarkan *confusion matrix* tersebut, dilakukan tahapan perhitungan beberapa metrik evaluasi model sebagai berikut :

- Perhitungan *precision* untuk kelas bola :

$$Precision_{bola} = \frac{TP}{TP+FP} \quad (4.1)$$

$$Precision_{bola} = \frac{0.92}{0.92+0.29} = 0.76 \text{ atau } 76\% \quad (4.2)$$

- Perhitungan *recall* untuk kelas bola :

$$Recall = \frac{TP}{TP+FN} \quad (4.3)$$

$$Recall_{bola} = \frac{0.92}{0.92+0.08} \quad (4.4)$$

$$Recall_{bola} = \frac{0.92}{1.00} = 0.92 \text{ atau } 92\% \quad (4.5)$$

- Perhitungan *Precision* robot dilakukan menggunakan persamaan :

$$Precision_{robot} = \frac{TP}{TP+FP} \quad (4.6)$$

$$Precision_{robot} = \frac{0.98}{0.98+0.71} = 0.58 \text{ atau } 58\ \% \quad (4.7)$$

- Perhitungan *recall* robot dilakukan menggunakan persamaan :

$$Recall_{robot} = \frac{0.98}{0.98+0.02} \quad (4.8)$$

$$Recall_{robot} = \frac{0.98}{1.00} = 0.98 \text{ atau } 98\% \quad (4.9)$$

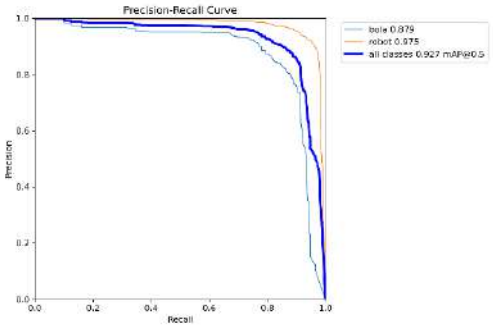
Tabel 4. 13 Rekapitulasi Hasil Evaluasi Model YOLOv5s Menggunakan Optimizer SGD

Kelas	*Precision (%)*	*Recall (%)*
Bola	76	92
Robot	58	98

Berdasarkan Tabel 4.13, model YOLOv5s menggunakan optimizer SGD menunjukkan performa deteksi yang cukup baik pada kelas bola dan robot. Pada kelas bola diperoleh nilai precision sebesar 76% dan recall sebesar 92%, sedangkan pada kelas robot diperoleh nilai precision sebesar 58% dan recall sebesar 98%. Nilai recall yang relatif tinggi pada kedua kelas menunjukkan bahwa sebagian besar objek pada data pengujian berhasil terdeteksi oleh model. Tingginya nilai recall pada kedua kelas menunjukkan bahwa sebagian besar objek berhasil dideteksi oleh sistem, meskipun masih terdapat beberapa kesalahan klasifikasi terhadap background yang menyebabkan nilai precision menjadi lebih rendah.

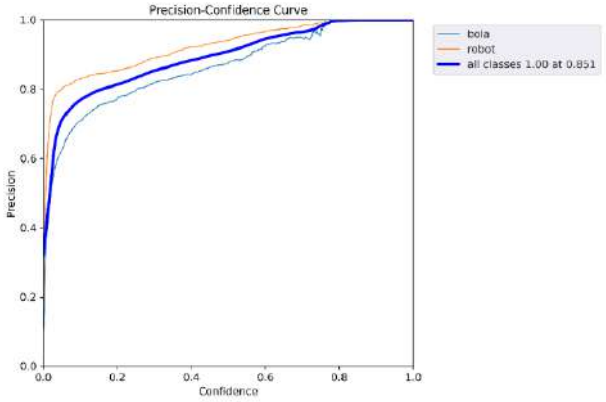
Selain itu, performa deteksi juga dipengaruhi oleh intensitas pencahayaan (lux) lingkungan. Pada kondisi pencahayaan yang baik, nilai confidence cenderung lebih tinggi karena citra yang dihasilkan memiliki kontras dan detail objek yang lebih jelas sehingga fitur visual lebih mudah dikenali oleh model YOLOv5s. Sebaliknya, pada kondisi pencahayaan rendah, kualitas citra menurun akibat noise dan distorsi dari kamera omnidirectional 360°, sehingga meningkatkan kemungkinan terjadinya false positive maupun false negative pada confusion matrix. Meskipun demikian, secara keseluruhan model deteksi mampu mengenali objek bola dan robot pada kondisi pengujian yang dilakukan.

Proses pelatihan (*training*) model berlangsung selama kurang lebih 4 jam. Setelah pelatihan selesai, sistem menghasilkan kurva *Precision-Recall* yang ditunjukkan pada Gambar 4.19.



Gambar 4. 19 *Precision-Recall Curve* SGD
(Penulis, 2026)

Berdasarkan Gambar 4.19, diperoleh nilai *mean Average Precision* (mAP@0.5) sebesar 92,7%, dengan rincian nilai *Average Precision* untuk kelas bola sebesar 87,9% dan kelas robot sebesar 97,5%. Nilai tersebut menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mendeteksi dan mengklasifikasikan objek sesuai dengan kelasnya. Tingginya nilai mAP mengindikasikan bahwa model mampu menghasilkan prediksi yang akurat dengan keseimbangan yang baik antara *precision* dan *recall*. Selain itu, perbedaan nilai antar kelas menunjukkan bahwa model lebih optimal dalam mendeteksi robot dibandingkan bola, yang kemungkinan dipengaruhi oleh variasi bentuk dan ukuran objek. Secara keseluruhan, hasil ini menunjukkan bahwa model deteksi objek yang telah dilatih memiliki performa yang baik dan dapat diandalkan untuk digunakan dalam sistem deteksi bola dan robot pada lingkungan pertandingan yang dinamis.

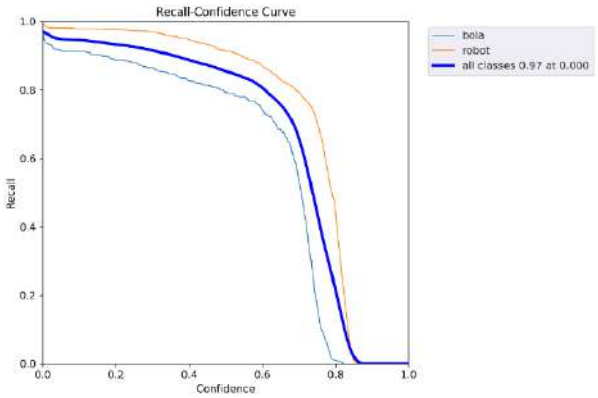


Gambar 4. 20 *Precision-Confidence Curve* SGD
(Penulis, 2026)

Gambar 4.20 menampilkan *Precision-Confidence Curve*, yaitu grafik yang menunjukkan hubungan antara nilai *confidence* (tingkat keyakinan model terhadap prediksi) dengan *precision* (tingkat ketepatan prediksi). Dari kurva tersebut terlihat bahwa baik untuk kelas bola maupun robot, model memiliki nilai precision yang tinggi pada sebagian besar rentang *confidence*.

Nilai *precision* terus meningkat seiring bertambahnya confidence dan mencapai nilai maksimum sebesar 1.00 pada confidence sekitar 0.851, sebagaimana ditunjukkan oleh kurva *all classes*. Hal ini menunjukkan bahwa pada tingkat *confidence* tersebut, model menghasilkan prediksi dengan tingkat precision yang tinggi pada *confidence* tersebut. Secara keseluruhan, hasil ini mengindikasikan

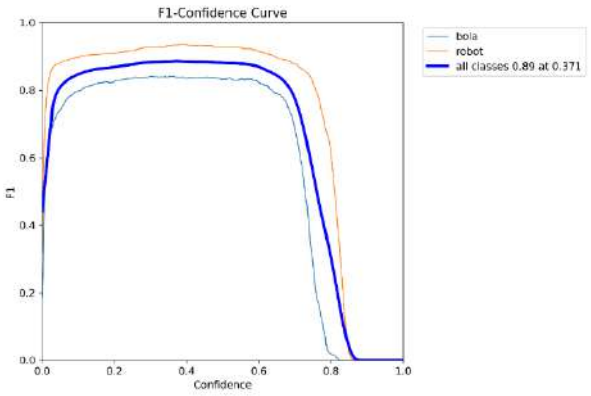
bahwa model memiliki performa deteksi yang sangat baik, khususnya dalam menjaga tingkat ketepatan prediksi. Dengan demikian, model dapat diandalkan untuk digunakan dalam sistem deteksi objek pada robot sepak bola yang membutuhkan akurasi tinggi dan respons yang stabil.



Gambar 4. 21 *Recall - Confidence Curve* SGD
(Penulis, 2026)

Gambar 4.21 menampilkan *Recall-Confidence Curve*, yaitu grafik yang menunjukkan hubungan antara nilai *confidence* dan *recall* dari model deteksi objek. *Recall* menggambarkan kemampuan model dalam mendeteksi seluruh objek yang seharusnya teridentifikasi. Berdasarkan grafik, nilai *recall* untuk kelas bola dan robot berada pada tingkat yang tinggi pada rentang *confidence* rendah hingga menengah. Hal ini menunjukkan bahwa model mampu mendeteksi sebagian besar objek dengan baik pada berbagai tingkat keyakinan. Namun, seiring meningkatnya nilai *confidence*, *recall* mulai mengalami penurunan yang cukup signifikan, terutama ketika mendekati nilai di atas 0.7 hingga 0.8.

Kondisi ini menunjukkan adanya *trade-off* antara *recall* dan *precision*, di mana semakin tinggi *confidence* yang digunakan, model menjadi lebih selektif sehingga beberapa objek tidak terdeteksi (*false negative*). Meskipun demikian, secara keseluruhan kurva ini menunjukkan bahwa model memiliki performa yang baik dan konsisten dalam mendeteksi objek, terutama pada rentang confidence yang memberikan keseimbangan precision dan recall yang baik untuk penggunaan sistem.



Gambar 4. 22 F-*Score Curve* SGD
(Penulis, 2026)

Gambar 4.22 menampilkan *F1-Confidence Curve*, yaitu grafik yang menunjukkan hubungan antara nilai *confidence* dan *F1-score* dari model deteksi objek. *F1-score* merupakan metrik yang menggabungkan *precision* dan *recall*, sehingga mencerminkan keseimbangan antara ketepatan dan kelengkapan dalam proses deteksi. Berdasarkan grafik, nilai *F1-score* untuk kelas bola dan robot berada pada tingkat yang tinggi pada rentang *confidence* rendah hingga menengah. Nilai maksimum untuk seluruh kelas tercapai sebesar 0.89 pada *confidence* sekitar 0.371, sebagaimana ditunjukkan oleh kurva *all classes*. Hal ini menunjukkan bahwa pada tingkat *confidence* tersebut, model memiliki performa yang paling optimal dalam menyeimbangkan *precision* dan *recall*.

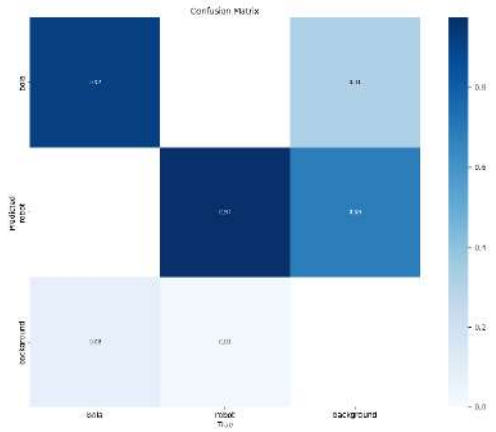
Seiring meningkatnya nilai *confidence*, *F1-score* mulai mengalami penurunan, terutama ketika mendekati nilai tinggi di atas 0.7 hingga 0.8. Hal ini mengindikasikan bahwa model menjadi lebih selektif, sehingga meskipun *precision* meningkat, nilai *recall* menurun dan berdampak pada penurunan *F1-score*.

4.3.4.2 Hasil Training Menggunakan Optimizer Adam

Pada penelitian ini, selain menggunakan optimizer SGD, proses training model YOLOv5s juga dilakukan menggunakan optimizer Adam. Optimizer Adam (*Adaptive Moment Estimation*) merupakan metode optimasi yang menggabungkan konsep *momentum* dan *adaptive learning rate* sehingga proses pembaruan bobot model dapat berlangsung lebih cepat dan stabil selama pelatihan.

Hasil training menggunakan optimizer Adam ditunjukkan pada confusion matrix, *Precision-Recall Curve, Precision-Confidence Curve*, dan *Recall-*

Confidence Curve. Berdasarkan confusion matrix, terdapat tiga kelas utama yang digunakan dalam proses klasifikasi, yaitu bola, robot, dan background.



Gambar 4. 23 *Confusion Matrix Adam*
(Penulis , 2026)

Tabel 4. 14 Pembacaan Nilai dari *Confusion Matrix* Adam

Predicted / True	Bola	Robot	Background
Bola	0.92	0	0.31
Robot	0	0.97	0.69
Background	0.08	0.03	0

Pada confusion matrix yang ditunjukkan pada Tabel 4.14, terdapat tiga kategori utama, yaitu bola, robot dan *backgorund.* Berdasarkan *confusion matrix* tersebut, dilakukan tahapan perhitungan beberapa metrik evaluasi model sebagai berikut :

- Perhitungan *precision* untuk kelas bola :

$$Precision_{bola} = \frac{TP}{TP+FP} \quad (4.10)$$

$$Precision_{bola} = \frac{0.92}{0.92+0.31} = 0.75 \text{ atau } 75\% \quad (4.11)$$

- Perhitungan *recall* untuk kelas bola :

$$Recall = \frac{TP}{TP+FN} \quad (4.12)$$

$$Recall_{bola} = \frac{0.92}{0.92+0.08} \quad (4.13)$$

$$Recall_{bola} = \frac{0.92}{1.00} = 0.92 \text{ atau } 92\% \quad (4.14)$$

- Perhitungan *Precision* robot dilakukan menggunakan persamaan :

$$Precision_{robot} = \frac{TP}{TP+FP} \quad (4.15)$$

$$Precision_{robot} = \frac{0.97}{0.97+0.69} = 0.58 \text{ atau } 58\ \% \quad (4.16)$$

- Perhitungan *recall* robot dilakukan menggunakan persamaan :

$$Recall_{robot} = \frac{0.97}{0.97+0.02} \quad (4.17)$$

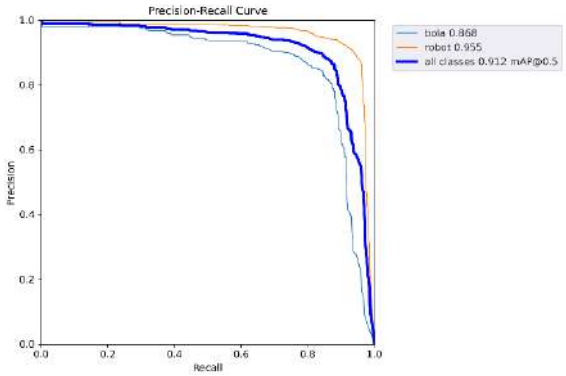
$$Recall_{robot} = \frac{0.97}{1.00} = 0.97 \text{ atau } 97\% \quad (4.18)$$

Tabel 4. 15 Rekapitulasi Hasil Evaluasi Model YOLOv5s Menggunakan Optimizer Adam

Kelas	*Precision (%)*	*Recall (%)*
Bola	75	92
Robot	58	97

Berdasarkan Tabel 4.15, model YOLOv5s menggunakan optimizer Adam menghasilkan nilai precision sebesar 75% dan recall sebesar 92% pada kelas bola, sedangkan pada kelas robot diperoleh precision sebesar 58% dan recall sebesar 97%. Nilai recall yang relatif tinggi menunjukkan bahwa sebagian besar objek pada data pengujian berhasil dideteksi oleh model. Namun nilai precision yang lebih rendah mengindikasikan masih terdapat false positive pada proses deteksi.

Kondisi tersebut dipengaruhi oleh bentuk objek robot yang lebih kompleks, variasi sudut pengambilan citra, serta pengaruh pencahayaan dan distorsi kamera omnidirectional 360°. Secara keseluruhan, model YOLOv5s menggunakan optimizer Adam tetap mampu menunjukkan performa deteksi yang cukup baik dan stabil untuk diterapkan pada sistem robot sepak bola berbasis ROS 2 secara *real-time.*

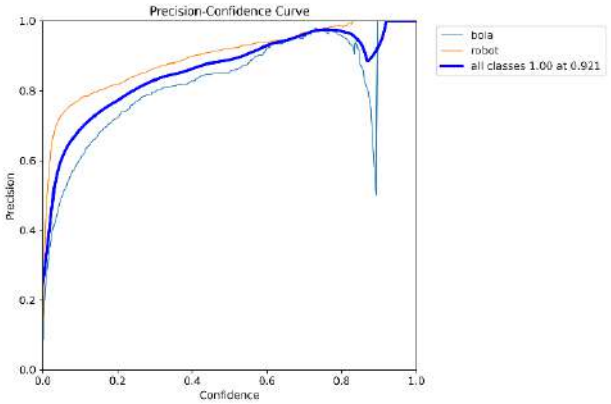


Gambar 4. 24 *PR - Curve Adam*
(Penulis, 2026)

Gambar 4.24 menunjukkan Precision-Recall Curve hasil training model YOLOv5s menggunakan optimizer Adam. Kurva tersebut digunakan untuk mengevaluasi hubungan antara nilai precision dan recall pada berbagai threshold

confidence selama proses deteksi objek. Berdasarkan grafik, diperoleh nilai *Average Precision* (AP) untuk kelas bola sebesar 0,868 atau 86,8%, sedangkan kelas robot memperoleh nilai AP sebesar 0,955 atau 95,5%. Selain itu, nilai *mean Average Precision* (mAP@0.5) untuk seluruh kelas mencapai 0,912 atau 91,2%.

Nilai AP yang lebih tinggi pada kelas robot menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mendeteksi objek robot pada berbagai tingkat recall. Sementara itu, nilai AP pada kelas bola sedikit lebih rendah karena objek bola memiliki ukuran yang lebih kecil dan lebih dipengaruhi oleh kondisi pencahayaan serta distorsi kamera omnidirectional 360°.

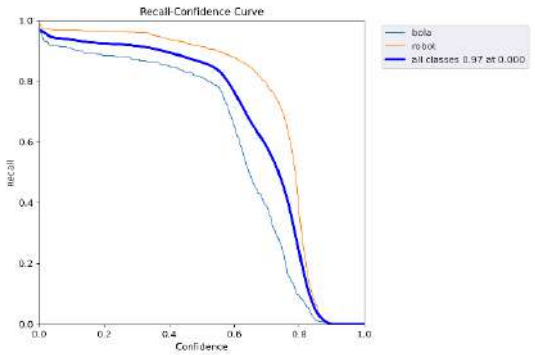


Gambar 4. 25 *Precision Curve* Adam
(Penulis, 2026)

Gambar 4.25 menunjukkan Precision-Confidence Curve hasil training model YOLOv5s menggunakan optimizer Adam. Kurva tersebut digunakan untuk menunjukkan hubungan antara nilai precision terhadap confidence *threshold* pada proses deteksi objek. Berdasarkan grafik, terlihat bahwa nilai precision cenderung meningkat seiring bertambahnya confidence threshold. Pada kelas robot, nilai precision menunjukkan performa yang lebih stabil dan lebih tinggi dibandingkan kelas bola. Hal ini menunjukkan bahwa model memiliki tingkat ketepatan yang baik dalam mendeteksi objek robot dengan kesalahan deteksi (*false positive*) yang relatif rendah pada confidence *threshold* tertentu. Sementara itu, pada kelas bola terlihat adanya penurunan precision pada threshold mendekati 0,9 yang dipengaruhi oleh ukuran objek bola yang lebih kecil, variasi pencahayaan, serta distorsi dari kamera omnidirectional 360°.

Selain itu, kurva keseluruhan (*all classes*) menunjukkan nilai precision

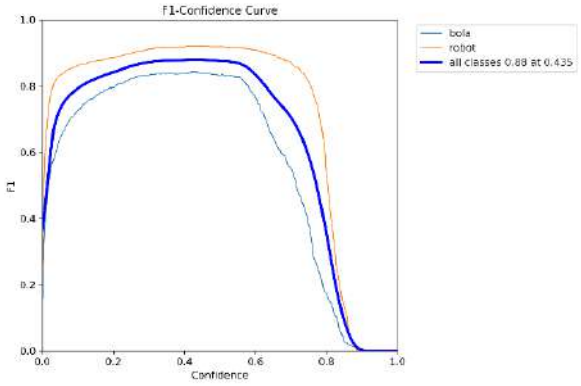
maksimum sebesar 1,00 pada confidence threshold 0,921. Hal ini menunjukkan bahwa pada threshold tersebut model mampu menghasilkan prediksi dengan tingkat ketepatan yang sangat tinggi, meskipun jumlah objek yang terdeteksi menjadi lebih sedikit.



Gambar 4. 26 *Recall Curve Adam*
(Penulis, 2026)

Gambar 4.26 menunjukkan Recall-Confidence Curve hasil training model YOLOv5s menggunakan optimizer Adam. Berdasarkan grafik, nilai recall cenderung menurun seiring meningkatnya confidence threshold. Hal ini menunjukkan bahwa semakin tinggi nilai confidence yang digunakan, maka jumlah objek yang berhasil terdeteksi akan semakin berkurang.

Pada kelas robot, nilai recall terlihat lebih tinggi dan stabil dibandingkan kelas bola, yang menunjukkan bahwa model lebih baik dalam mendeteksi objek robot pada berbagai threshold confidence. Sementara itu, penurunan recall pada kelas bola dipengaruhi oleh ukuran objek yang lebih kecil, kondisi pencahayaan, serta distorsi kamera omnidirectional 360°.



Gambar 4. 27 *F-Score Curve Adam*
(Penulis, 2026)

Gambar 4.27 menunjukkan F1-Confidence Curve hasil training model

YOLOv5s menggunakan optimizer Adam. Berdasarkan grafik, diperoleh nilai F1-score maksimum sebesar 0,88 pada confidence threshold 0,435. Hal ini menunjukkan bahwa model memiliki keseimbangan yang baik antara precision dan recall pada threshold tersebut.

4.3.4.3 Perbandingan Performa Optimizer SGD dan Adam

Pada penelitian ini dilakukan perbandingan performa model YOLOv5s menggunakan optimizer SGD dan Adam untuk mengetahui optimizer yang memberikan performa deteksi lebih baik pada dataset penelitian. Perbandingan dilakukan berdasarkan nilai AP, precision, recall, F1-score, serta mAP@0.5 yang diperoleh dari hasil training dan validasi model.

Rekapitulasi hasil evaluasi model berdasarkan confusion matrix pada masing-masing optimizer ditunjukkan pada Tabel 4.16. Perbandingan dilakukan untuk mengetahui pengaruh penggunaan optimizer SGD dan Adam terhadap performa deteksi objek bola dan robot menggunakan model YOLOv5s.

Tabel 4. 16 Perbandingan Performa Model YOLOv5s Menggunakan Optimizer SGD dan Adam

Optimizer	AP Bola (%)	AP Robot (%)	Precision Bola (%)	Recall Bola (%)	Precision Robot (%)	Recall Robot (%)	mAP@0.5	F-1 Score
SGD	87,9	97,5	76	92	58	98	0.927	0.87
Adam	86,8	95,5	75	92	58	97	0.912	0.88

Berdasarkan Tabel 4.16, performa kedua optimizer menunjukkan hasil yang relatif berdekatan pada beberapa metrik. Pada kelas bola, optimizer SGD menghasilkan precision sebesar 76% dan recall sebesar 92%, sedangkan optimizer Adam menghasilkan precision sebesar 75% dan recall sebesar 92%. Pada kelas robot, kedua optimizer menghasilkan precision yang sama, yaitu sebesar 58%. Sementara itu, recall optimizer SGD sebesar 98% sedikit lebih tinggi dibandingkan Adam sebesar 97%.

Berdasarkan nilai Average Precision (AP), optimizer SGD menghasilkan AP sebesar 87,9% pada kelas bola dan 97,5% pada kelas robot. Sementara itu, optimizer Adam menghasilkan AP sebesar 86,8% pada kelas bola dan 95,5% pada

kelas robot. Nilai mAP@0.5 yang diperoleh optimizer SGD sebesar 92,7%, sedangkan optimizer Adam sebesar 91,2%.

Meskipun F1-score optimizer Adam sebesar 0,88 sedikit lebih tinggi dibandingkan SGD sebesar 0,87, optimizer SGD memberikan nilai AP dan mAP@0.5 yang lebih tinggi pada dataset penelitian. Selain itu, SGD menghasilkan recall robot sebesar 98%, sedangkan Adam sebesar 97%. Karena deteksi robot lawan merupakan aspek penting dalam proses obstacle avoidance, recall pada kelas robot menjadi salah satu pertimbangan utama dalam pemilihan optimizer. Berdasarkan pertimbangan tersebut, optimizer SGD dipilih sebagai optimizer yang digunakan pada implementasi sistem.

Berdasarkan hasil evaluasi tersebut, model YOLOv5s dengan optimizer SGD digunakan pada tahap integrasi sistem. Model menghasilkan mAP@0.5 sebesar 92,7%, dengan AP sebesar 87,9% pada kelas bola dan 97,5% pada kelas robot. Model terpilih selanjutnya digunakan pada tahap integrasi dengan sistem estimasi posisi dan path planning A* dengan mekanisme dynamic replanning.

4.3.5 Deteksi Robot dengan YOLOv5s pada Variasi Intensitas Cahaya

Deteksi robot secara andal merupakan salah satu aspek penting dalam sistem robotika berbasis visi komputer. Namun, implementasi sistem deteksi objek berbasis *deep learning* memiliki tantangan utama berupa perubahan kondisi pencahayaan lingkungan yang dapat mempengaruhi performa model. Oleh karena itu, pada penelitian ini dilakukan pengujian performa deteksi robot menggunakan model YOLOv5s pada berbagai variasi intensitas cahaya. Pengujian ini bertujuan untuk mengetahui tingkat akurasi dan kestabilan model dalam mendeteksi objek robot pada kondisi pencahayaan yang berbeda, mulai dari kondisi terang hingga redup.

Melalui pengujian tersebut, diharapkan dapat diperoleh pemahaman mengenai kemampuan serta keterbatasan sistem deteksi robot berbasis YOLOv5s terhadap perubahan lingkungan pencahayaan. Proses pengujian dilakukan di dalam ruangan dengan intensitas cahaya yang dapat diatur. Robot ditempatkan pada area pengamatan kamera, kemudian dilakukan pengambilan data pada beberapa kondisi pencahayaan berbeda sebagaimana ditunjukkan pada Gambar 4.28.

Gambar 4. 28 Pengujian Deteksi Robot berdasarkan Lux
(Penulis, 2026)

Pengambilan data dilakukan dengan menempatkan robot target pada posisi tetap dengan jarak sekitar 5 meter dari kamera. Intensitas cahaya diatur menggunakan lampu LED dan diukur menggunakan lux meter.

Kategori pencahayaan pada pengujian terdiri dari :

1. Cahaya terang (>300 lux), menyerupai kondisi lapangan pertandingan.
2. Cahaya sedang (100–300 lux), menggunakan pencahayaan ruangan tanpa lampu tambahan.
3. Cahaya redup (<100 lux), menggunakan cahaya alami dari luar ruangan.

Tabel 4. 17 Pengujian deteksi berdasarkan Lux

Pengujian Ke -	Kondisi Cahaya	Lux	Hasil
1	Terang	330	Terdeteksi
2	Terang	312	Terdeteksi
3	Terang	301	Terdeteksi
4	Sedang	238	Terdeteksi
5	Sedang	233	Terdeteksi
6	Sedang	183	Terdeteksi
7	Sedang	115	Terdeteksi
8	Sedang	102	Terdeteksi
9	Redup	98	Terdeteksi
10	Redup	63	Tidak Terdeteksi

Berdasarkan data pada Tabel 4.17, performa deteksi robot menggunakan

model YOLOv5s menunjukkan hasil yang baik pada kondisi cahaya terang (>300 lux) dan sedang (100–300 lux), di mana seluruh pengujian berhasil mendeteksi objek robot. Hal ini menunjukkan bahwa sistem deteksi mampu bekerja secara stabil ketika intensitas cahaya cukup untuk menampilkan fitur visual objek dengan jelas pada kamera.

Namun, pada kondisi cahaya redup (<100 lux), performa deteksi mulai mengalami penurunan. Pada intensitas cahaya 98 lux robot masih dapat terdeteksi, sedangkan pada 63 lux objek robot tidak berhasil terdeteksi. Kondisi tersebut menunjukkan bahwa pencahayaan yang terlalu rendah dapat mengurangi kualitas citra dan mengganggu proses ekstraksi fitur oleh model YOLOv5s.

Secara keseluruhan, hasil pengujian menunjukkan bahwa intensitas cahaya berpengaruh terhadap performa deteksi robot. Sistem bekerja lebih optimal pada kondisi pencahayaan sedang hingga terang, sehingga pencahayaan lingkungan menjadi faktor penting dalam mendukung kestabilan deteksi objek selama proses navigasi.

4.3.6 Deteksi Bola dengan YOLOv5s pada Variasi Intensitas Cahaya

Dalam sistem robotika berbasis visi komputer, kemampuan mendeteksi bola secara akurat merupakan bagian penting pada robot sepak bola otomatis. Bola digunakan sebagai objek utama untuk mendukung navigasi, strategi permainan, dan pengambilan keputusan robot selama proses navigasi. Namun, performa deteksi bola dipengaruhi oleh kondisi pencahayaan lingkungan yang berubah-ubah. Model YOLOv5s sangat bergantung pada fitur visual objek, seperti bentuk dan warna, sehingga intensitas cahaya yang terlalu rendah atau terlalu terang dapat mempengaruhi keakuratan deteksi bola. Dokumentasi proses pengujian ditunjukkan pada Gambar 4.29.

Gambar 4. 29 Pengujian deteksi bola berdasarkan Lux
(Penulis, 2026)

Pengambilan data dilakukan dengan menempatkan bola sebagai objek utama pada posisi tetap dengan jarak sekitar 5 meter dari kamera. Intensitas cahaya diatur menggunakan lampu LED, kemudian diukur menggunakan lux meter untuk memperoleh kategori pencahayaan tertentu. Pengujian dilakukan pada tiga kondisi pencahayaan, yaitu:

1. Cahaya terang (>300 lux), menyerupai kondisi lapangan pertandingan.
2. Cahaya sedang (100–300 lux), menggunakan pencahayaan ruangan tanpa lampu tambahan.
3. Cahaya redup (<100 lux), menggunakan cahaya alami dari luar ruangan.

Tabel 4. 18 Pengujian deteksi bola berdasarkan Lux

Pengujian Ke -	Kondisi Cahaya	Lux	Hasil
1	Terang	338	Terdeteksi
2	Terang	321	Terdeteksi
3	Terang	312	Terdeteksi
4	Sedang	262	Terdeteksi
5	Sedang	242	Terdeteksi
6	Sedang	225	Terdeteksi
7	Sedang	196	Terdeteksi
8	Sedang	203	Terdeteksi
9	Redup	73	Tidak Terdeteksi
10	Redup	63	Tidak Terdeteksi

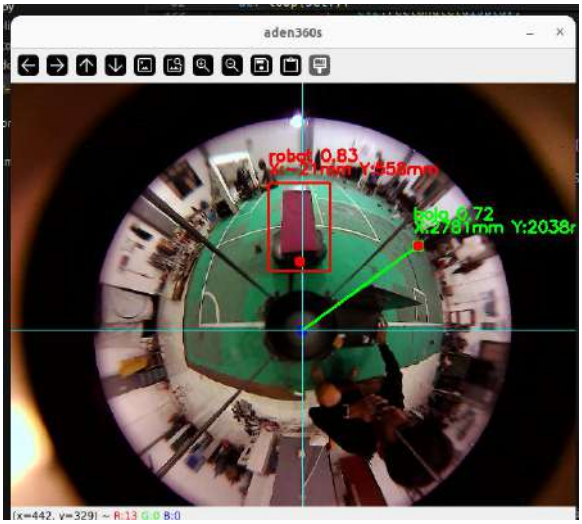
Berdasarkan data pada Tabel 4.18, performa deteksi bola menggunakan model YOLOv5s menunjukkan hasil yang baik pada kondisi cahaya terang (>300

lux) dan sedang (100–300 lux), di mana seluruh pengujian berhasil mendeteksi objek bola. Hal ini menunjukkan bahwa model mampu mengenali objek bola secara stabil ketika intensitas cahaya cukup dan fitur visual objek terlihat jelas pada hasil citra kamera. Namun, pada kondisi cahaya redup (<100 lux), objek bola tidak berhasil terdeteksi pada seluruh pengujian. Pada intensitas cahaya 73 lux dan 63 lux, sistem gagal mengenali objek bola karena kualitas citra mengalami penurunan akibat kurangnya pencahayaan.

Secara keseluruhan, hasil pengujian menunjukkan bahwa intensitas cahaya sangat mempengaruhi performa deteksi bola. Sistem bekerja optimal pada kondisi pencahayaan sedang hingga terang, sehingga pencahayaan lingkungan menjadi faktor penting untuk menjaga kestabilan deteksi objek bola selama proses navigasi.

4.3.7 Pengujian Deteksi Robot dengan YOLOv5s

Pengujian deteksi robot dilakukan untuk mengevaluasi kemampuan sistem dalam mengenali robot lawan menggunakan model YOLOv5s. Parameter utama yang digunakan adalah jarak antara kamera dan objek, dengan pengujian dilakukan pada beberapa variasi jarak mulai dari dekat hingga batas maksimum deteksi. Tujuan pengujian ini adalah untuk mengetahui kemampuan model dalam mendeteksi robot secara akurat serta menganalisis pengaruh jarak terhadap performa deteksi. Hasil pengujian dievaluasi berdasarkan nilai *confidence* dan keberhasilan *bounding box* dalam mengidentifikasi objek robot.



Gambar 4. 30 Deteksi robot menggunakan YOLOv5s
(Penulis, 2026)

Gambar 4.30 menampilkan hasil pengujian deteksi objek robot lawan

menggunakan sistem berbasis YOLOv5s. Pada pengujian ini, robot ditempatkan dalam kondisi statis dengan jarak sekitar 2 meter dari kamera. Citra yang diperoleh diproses secara kontinu oleh sistem untuk mendeteksi objek berdasarkan model yang telah dilatih sebelumnya. Hasil deteksi menunjukkan bahwa objek robot berhasil dikenali dengan baik, ditandai dengan *bounding box* berwarna merah yang mengelilingi objek, serta label "robot" dengan nilai *confidence* sebesar 0.83, yang menunjukkan tingkat keyakinan model yang cukup tinggi.

Selain itu, sistem juga menampilkan informasi posisi objek dalam bentuk koordinat relatif terhadap kamera, yaitu sekitar X: -17 mm dan Y: 958 mm, yang ditunjukkan pada tampilan. Informasi ini dapat digunakan sebagai referensi dalam menentukan arah dan posisi objek untuk keperluan navigasi robot.

Pengujian dilakukan pada lingkungan dalam ruangan dengan kondisi pencahayaan yang cukup, menyerupai kondisi lapangan pertandingan. Meskipun terdapat berbagai objek lain di sekitar, sistem tetap mampu mendeteksi robot dengan stabil tanpa terganggu oleh latar belakang. Hal ini menunjukkan bahwa model memiliki ketahanan yang baik terhadap *noise* visual dan mampu fokus pada objek utama yang dideteksi.

Tabel 4. 19 Hasil pengujian deteksi Robot

Pengujian Ke -	Jarak Robot (mm)	Intesitas Cahaya (Lux)	Hasil Deteksi	Nilai *Confidence*
1	500	98	Terdeteksi	0.83
2	1000	102	Terdeteksi	0.83
3	1500	119	Terdeteksi	0.81
4	2000	139	Terdeteksi	0.80
5	2500	183	Terdeteksi	0.81
6	3000	233	Terdeteksi	0.80
7	3500	238	Terdeteksi	0.78
8	4000	200	Terdeteksi	0.75
9	4500	299	Terdeteksi	0.61
10	5000	285	Terdeteksi	0.61
Rata – Rata Nilai Confidence				**0.763**

Tabel 4.19 menyajikan hasil pengujian sistem deteksi robot lawan menggunakan algoritma YOLOv5s pada variasi jarak antara robot dan kamera. Pengujian dilakukan sebanyak 10 kali dengan jarak mulai dari 500 mm hingga 5000 mm. Pada setiap pengujian turut dicatat intensitas cahaya dalam satuan lux sebagai

informasi kondisi pencahayaan saat pengambilan data.

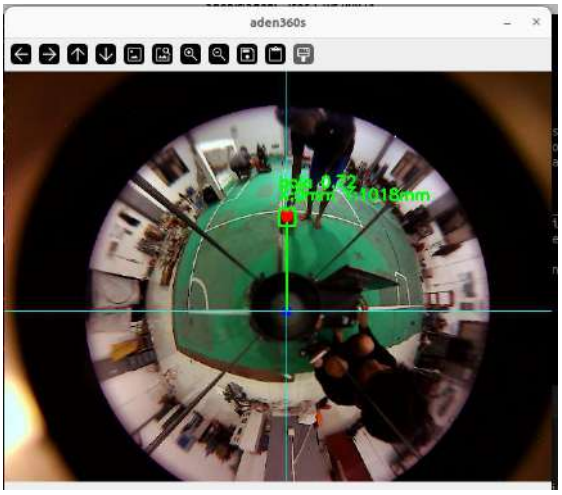
Berdasarkan hasil pengujian, sistem mampu mendeteksi objek robot pada seluruh variasi jarak yang diuji, yang ditunjukkan dengan status "Terdeteksi" pada setiap percobaan. Hasil tersebut menunjukkan bahwa objek robot masih dapat dikenali oleh model pada rentang jarak 500–5000 mm dalam kondisi pengujian yang dilakukan.

Pada kolom nilai *confidence*, terlihat adanya variasi nilai antara 0.83 hingga 0.61. Nilai *confidence* yang tinggi terdapat pada jarak dekat hingga menengah (500 mm – 3000 mm), yaitu berkisar antara 0.80 hingga 0.83. Pada jarak 4000–5000 mm, nilai *confidence* cenderung lebih rendah dibandingkan beberapa pengujian pada jarak dekat hingga menengah. Kondisi tersebut dapat berkaitan dengan ukuran objek yang semakin kecil pada citra dan berkurangnya detail visual yang tersedia bagi model. Namun, karena intensitas pencahayaan juga mengalami perubahan selama pengujian, pengaruh jarak dan pencahayaan tidak dapat dipisahkan sepenuhnya dari data tersebut. Dengan demikian, hasil pengujian ini digunakan untuk menunjukkan hubungan pengamatan antara variasi jarak, intensitas pencahayaan, dan nilai confidence, bukan untuk menentukan pengaruh masing-masing variabel secara independen.

Secara keseluruhan, rata-rata nilai confidence yang diperoleh sebesar 0,763. Nilai tersebut menunjukkan bahwa pada kondisi pengujian yang dilakukan, model tetap memberikan tingkat keyakinan deteksi yang cukup baik dan seluruh objek robot pada variasi jarak yang diuji berhasil terdeteksi.

4.3.8 Pengujian Deteksi Bola menggunakan YOLOv5s

Pengujian deteksi bola dilakukan untuk mengevaluasi kemampuan sistem dalam mengenali objek bola secara akurat pada berbagai variasi jarak dan kondisi pencahayaan. Parameter utama yang digunakan meliputi jarak antara kamera dan bola serta nilai *confidence* dari hasil deteksi. Sistem berbasis YOLOv5s yang telah dilatih diuji untuk mendeteksi bola sebagai objek target dalam pertandingan robot sepak bola. Tujuan pengujian ini adalah untuk mengetahui kemampuan deteksi bola berdasarkan keberhasilan deteksi dan nilai confidence pada berbagai jarak, serta menentukan rentang jarak pengamatan berdasarkan hasil deteksi yang diperoleh.



Gambar 4. 31 Deteksi Bola menggunakan YOLOv5s
(Penulis, 2026)

Gambar 4.31 menampilkan hasil pengujian deteksi bola menggunakan kamera omnivision 360° dan sistem berbasis YOLOv5s pada jarak sekitar 2 meter. Pada tampilan tersebut, objek bola berhasil dikenali oleh sistem dan ditandai dengan *bounding box* berwarna hijau, disertai label "bola" dengan nilai *confidence* sekitar 0.80, yang menunjukkan tingkat keyakinan model yang cukup baik dalam mendeteksi objek. Selain itu, sistem juga menampilkan informasi posisi bola dalam bentuk koordinat relatif, yaitu sekitar X: -89 mm dan Y: 1557 mm, yang dapat digunakan sebagai acuan dalam menentukan arah dan posisi bola terhadap robot. Informasi ini penting untuk mendukung proses navigasi dan pengambilan keputusan dalam pergerakan robot.

Hasil visual menunjukkan bahwa sistem mampu mendeteksi objek bola dengan cukup akurat meskipun berada dalam lingkungan dengan latar belakang yang kompleks. Deteksi dilakukan secara kontinu dan bounding box yang dihasilkan cukup stabil mengikuti posisi objek. Secara keseluruhan, hasil ini menunjukkan bahwa model deteksi memiliki performa yang baik dalam mengenali bola pada jarak menengah, sehingga dapat mendukung kinerja robot dalam kondisi pertandingan yang dinamis.

Tabel 4. 20 Pengujian hasil deteksi bola

Pengujian Ke -	Jarak Bola (mm)	Intesitas Cahaya (Lux)	Hasil Deteksi	Nilai *Confidence*
1	500	82	Terdeteksi	0.76
2	1000	109	Terdeteksi	0.72
3	1500	182	Terdeteksi	0.81

Tabel 4.20 Pengujian hasil deteksi bola (Lanjutan)

Pengujian Ke -	Jarak Bola (mm)	Intesitas Cahaya (Lux)	Hasil Deteksi	Nilai *Confidence*
4	2000	204	Terdeteksi	0.76
5	2500	271	Terdeteksi	0.75
6	3000	196	Terdeteksi	0.73
7	3500	263	Terdeteksi	0.74
8	4000	225	Terdeteksi	0.73
9	4500	242	Terdeteksi	0.74
10	5000	203	Terdeteksi	0.72
Rata – Rata Nilai Confidence				**0.746**

Tabel 4.20 menyajikan hasil pengujian sistem deteksi bola menggunakan model YOLOv5s pada variasi jarak antara bola dan kamera. Pengujian dilakukan sebanyak 10 kali dengan jarak mulai dari 500 mm hingga 5000 mm. Pada setiap pengujian turut dicatat intensitas cahaya dalam satuan lux sebagai informasi kondisi pencahayaan saat pengambilan data. Berdasarkan hasil pengujian, sistem berhasil mendeteksi bola pada seluruh variasi jarak yang diuji, yang ditunjukkan dengan status “Terdeteksi” pada setiap percobaan. Hasil tersebut menunjukkan bahwa objek bola berhasil dideteksi pada seluruh variasi jarak yang diuji dalam kondisi pencahayaan yang teramati selama pengambilan data.

Nilai *confidence* yang diperoleh berada pada rentang 0,72 hingga 0,81. Pada beberapa pengujian jarak dekat hingga menengah, nilai confidence relatif lebih tinggi, sedangkan pada jarak yang lebih jauh nilai confidence cenderung berada pada nilai yang lebih rendah. Kondisi tersebut dapat berkaitan dengan semakin kecilnya ukuran objek bola pada citra dan berkurangnya detail visual yang dapat dikenali oleh model. Meskipun seluruh objek bola pada variasi jarak 500–5000 mm berhasil terdeteksi, nilai confidence pada beberapa pengujian jarak jauh cenderung lebih rendah dibandingkan jarak dekat hingga menengah. Namun, karena intensitas pencahayaan pada setiap pengujian tidak dijaga konstan, penurunan nilai confidence tersebut belum dapat dikaitkan secara khusus dengan peningkatan jarak. Oleh karena itu, hasil pengujian ini menunjukkan kemampuan deteksi pada kombinasi jarak dan kondisi pencahayaan yang teramati, bukan sebagai pengukuran pengaruh jarak secara independen.

4.3.9 Estimasi Posisi Objek / Kalibrasi Kamera

Kalibrasi dilakukan untuk memperoleh hubungan antara radius objek pada citra dalam satuan piksel dan jarak objek terhadap kamera dalam satuan milimeter. Hubungan tersebut dimodelkan menggunakan persamaan polinomial sebagai berikut: $r_m m = a_3 rpixel^3 + a_2 rpixel^2 + a_1 rpixel + a_0$ (4.19)

dengan r_mm merupakan estimasi jarak objek dalam satuan milimeter, r_pixel merupakan radius objek pada citra dalam satuan piksel, sedangkan a_0, a_1, a_2, dan a_3 merupakan koefisien hasil proses kalibrasi.

Data kalibrasi digunakan untuk memperoleh hubungan antara radius objek pada citra dan jarak aktual objek terhadap kamera. Data pada Tabel 4.21 dan Tabel 4.22 digunakan untuk memperoleh hubungan antara radius objek pada citra dan jarak aktual objek terhadap kamera. Hasil estimasi jarak kemudian dibandingkan dengan jarak aktual untuk mengetahui besarnya penyimpangan hasil kalibrasi. Perbandingan antara jarak hasil estimasi kamera dan jarak aktual objek dilakukan dalam satuan milimeter untuk mengetahui besarnya error estimasi pada setiap titik pengukuran. Nilai R^2 digunakan untuk menunjukkan tingkat kesesuaian model polinomial terhadap data kalibrasi, sedangkan nilai Mean Absolute Error (MAE) dan Root Mean Square Error (RMSE) digunakan untuk mengetahui besarnya kesalahan estimasi pada titik pengukuran.

Tabel 4. 21 Hasil Validasi Estimasi Jarak Relatif Kamera pada Objek Robot Lawan

No	Jarak Aktual (mm)	Radius Piksel	Jarak Hasil Kalibrasi (mm)	Error (mm)	Error (%)
1	500	71,3	558	58	11,60
2	1000	108,12	958	42	4,20
3	1500	121	1416	84	5,60
4	2000	140,27	1880	120	6,00
5	2500	146	2225	275	11,00
6	3000	156	2761	239	7,97

Berdasarkan Tabel 4.21, hasil estimasi jarak objek robot lawan menunjukkan adanya perbedaan antara jarak aktual dan jarak hasil kalibrasi. Nilai error absolut berada pada rentang 42–275 mm, dengan error terbesar terjadi pada jarak aktual 2500 mm sebesar 275 mm atau 11,00%. Sementara itu, error terkecil terjadi pada jarak 1000 mm sebesar 42 mm atau 4,20%. Berdasarkan seluruh titik pengukuran, diperoleh nilai MAE sebesar 136,33 mm dan RMSE sebesar 162,96

mm. Hasil tersebut menunjukkan bahwa estimasi jarak masih memiliki penyimpangan terhadap jarak aktual, terutama pada beberapa jarak pengukuran yang lebih jauh.

Tabel 4. 22 Hasil Validasi Estimasi Jarak Relatif Kamera pada Objek Bola

No	Jarak Aktual (mm)	Radius Piksel	Jarak Hasil Kalibrasi (mm)	Error (mm)	Error (%)
1	500	71,3	536	36	7,20
2	1000	108,12	1018	18	1,80
3	1500	121	1557	57	3,80
4	2000	140,27	2074	74	3,70
5	2500	146	2575	75	3,00
6	3000	156	3039	39	1,30

Berdasarkan Tabel 4.22, hasil estimasi jarak objek bola memiliki selisih terhadap jarak aktual pada seluruh titik pengukuran. Nilai error absolut berada pada rentang 18–75 mm. Error terbesar terjadi pada jarak aktual 2500 mm sebesar 75 mm atau 3,00%, sedangkan error terkecil terjadi pada jarak 1000 mm sebesar 18 mm atau 1,80%. Berdasarkan seluruh titik pengukuran, diperoleh nilai MAE sebesar 49,83 mm dan RMSE sebesar 53,99 mm. Hasil tersebut menunjukkan bahwa penyimpangan estimasi jarak pada objek bola lebih kecil dibandingkan dengan objek robot lawan pada data pengukuran yang digunakan.

Berdasarkan perbandingan hasil kalibrasi pada kedua objek, estimasi jarak pada objek bola menghasilkan nilai kesalahan yang lebih rendah dibandingkan dengan objek robot lawan. Nilai MAE pada objek bola sebesar 49,83 mm, sedangkan pada objek robot lawan sebesar 136,33 mm. Demikian pula, nilai RMSE pada objek bola sebesar 53,99 mm lebih rendah dibandingkan dengan objek robot lawan sebesar 162,96 mm. Perbedaan tersebut menunjukkan bahwa karakteristik objek yang terdeteksi dapat memengaruhi hasil estimasi jarak berdasarkan radius objek pada citra. Hasil estimasi posisi relatif ini selanjutnya digunakan sebagai informasi posisi objek dalam proses perencanaan jalur robot.

4.4. Hasil Pengujian *Path Planning* A* pada Gazebo

Pengujian ini dilakukan untuk menilai efektivitas pergerakan robot dengan membandingkan dua pendekatan algoritma path planning A*, yaitu A* konvensional yang bersifat statis dan A* yang dikembangkan dalam penelitian ini dengan mekanisme dynamic replanning. Tujuan dari pengujian ini adalah untuk

mengetahui kemampuan dynamic replanning dalam meningkatkan adaptivitas navigasi dan keberhasilan penghindaran obstacle terhadap perubahan lingkungan.

Pada pendekatan A* konvensional, jalur menuju titik tujuan dihitung satu kali di awal berdasarkan kondisi lingkungan saat itu, kemudian robot akan mengikuti jalur tersebut hingga mencapai tujuan tanpa melakukan pembaruan. Metode ini memiliki keterbatasan dalam menghadapi perubahan lingkungan, karena robot tidak mampu menyesuaikan jalur ketika terdapat pergerakan rintangan di sepanjang lintasan yang telah direncanakan. Akibatnya, robot berpotensi menabrak robot lawan atau melewati zona yang seharusnya dihindari.

Sebaliknya, pada pendekatan A* dengan dynamic replanning yang digunakan dalam penelitian ini, jalur tidak bersifat statis, melainkan dihitung dan diperbarui secara berkala selama robot bergerak. Algoritma secara aktif mempertimbangkan perubahan posisi robot lawan sebagai *obstacle* dinamis, serta menghitung ulang jalur berdasarkan kondisi *obstacle* terbaru ketika terdeteksi adanya potensi konflik pada lintasan yang sedang dilalui. Dengan demikian, robot mampu menyesuaikan arah geraknya berdasarkan perubahan posisi robot lawan selama proses navigasi, sehingga menghasilkan pergerakan yang lebih adaptif, aman, dan terkontrol.

Beberapa indikator yang diamati dalam proses pengujian ini meliputi durasi waktu tempuh, panjang lintasan yang dilalui, serta tingkat keberhasilan dalam menghindari robot lawan. Waktu tempuh diukur mulai dari saat robot mulai bergerak dari posisi awal hingga mencapai titik tujuan, yang menunjukkan efisiensi waktu dari masing-masing metode. Jarak tempuh dihitung berdasarkan total lintasan pergerakan robot, baik pada A* konvensional maupun A* dengan dynamic replanning. Semakin pendek jarak yang ditempuh, maka semakin efisien rute tersebut secara spasial, meskipun dalam kondisi tertentu jalur yang lebih panjang dapat terjadi akibat manuver penghindaran terhadap rintangan. Sementara itu, tingkat keberhasilan diukur berdasarkan kemampuan robot dalam mencapai titik tujuan tanpa menyentuh atau memasuki zona larangan di sekitar robot lawan. Apabila robot berhasil menyelesaikan tugas tanpa pelanggaran, maka percobaan dinyatakan berhasil.

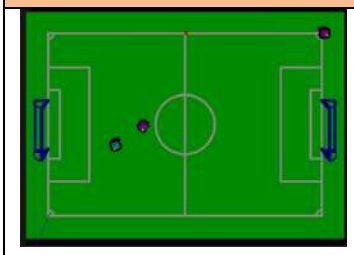
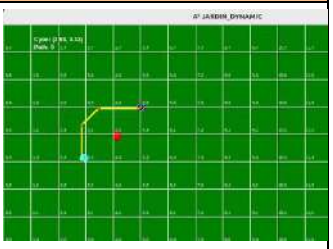
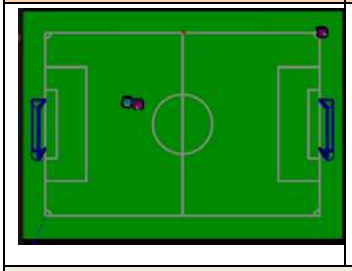
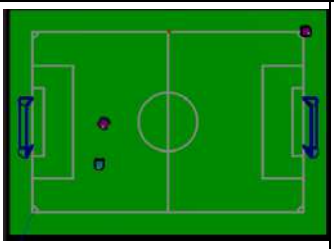
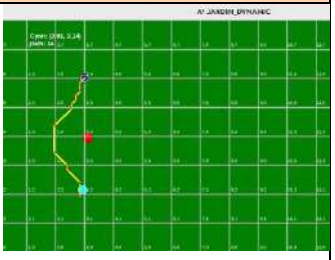
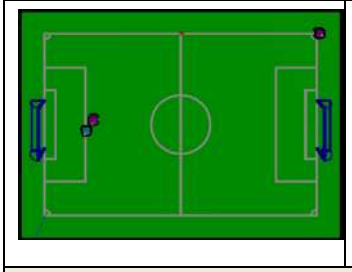
Dalam pengujian ini, dilakukan beberapa percobaan dengan variasi posisi

awal robot, titik tujuan, serta posisi robot lawan sebagai obstacle dinamis. Variasi ini bertujuan untuk mengevaluasi konsistensi performa sistem dalam menghadapi perubahan kondisi lingkungan. Pada setiap percobaan, algoritma A* dengan dynamic replanning akan terus memperbarui jalur berdasarkan kondisi terbaru di lapangan, sementara A* konvensional hanya menggunakan jalur awal tanpa penyesuaian. Dengan pendekatan ini, diharapkan dapat diperoleh gambaran yang lebih komprehensif mengenai keunggulan dynamic replanning dalam meningkatkan kemampuan navigasi robot, khususnya dalam skenario kompetitif seperti pada ajang KRSBI-B yang memiliki lingkungan dinamis dan tidak terprediksi.

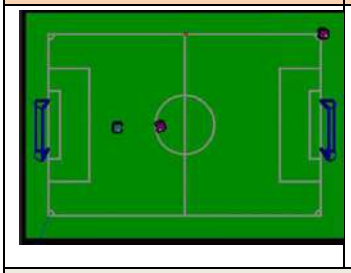
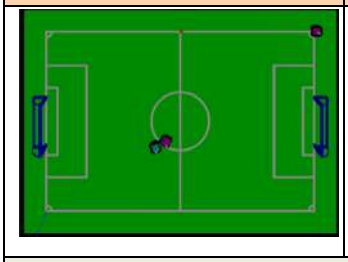
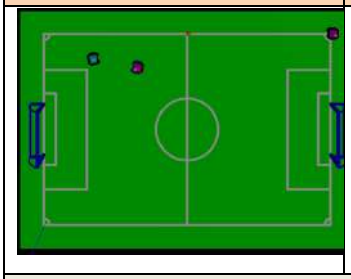
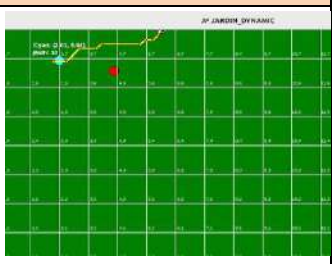
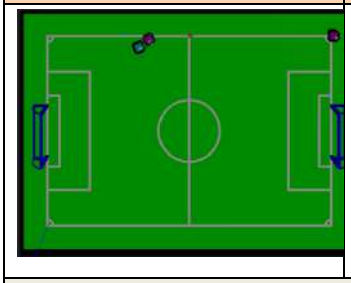
Tabel 4. 23 Hasil pengujian *path planning* A* Terdahulu

Percobaan Ke -	Titik awal (y,x)	Titik Tujuan (y,x)	Titik halangan (y,x)	Titik Halangan Bergerak (y,x)	Waktu (s)	Jarak (m)
1	(3, 3)	(5, 5)	(4, 4)	(4, 5)	24,22	2.83
2	(3, 2)	(3, 6)	(3, 4)	(2, 4)	23,81	4.00
3	(3, 4)	(7, 4)	(5, 4)	(5, 3.5)	26,75	4.00
4	(2, 7)	(5.5, 8)	(4, 6.5)	(4, 8)	25,65	3.64
5	(6, 5.5)	(4, 8)	(5, 7)	(4, 7)	21,51	3.20
6	(9, 4)	(5, 8)	(7.5, 6)	(6, 6)	23,32	5.66
7	(5, 4)	(5, 9)	(5, 7)	(4, 7)	23,19	5.00
8	(3, 7)	(7, 7)	(4.4, 6.4)	(4.4, 7.5)	27,51	4.00
9	(4.5, 4)	(6, 8)	(5, 6)	(6, 6)	28,79	4.27
10	(0.5, 1.1)	(4, 4.9)	(2.5, 3.3)	(1, 3)	24,75	5.17

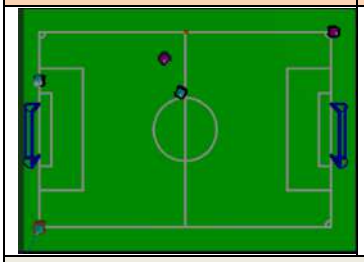
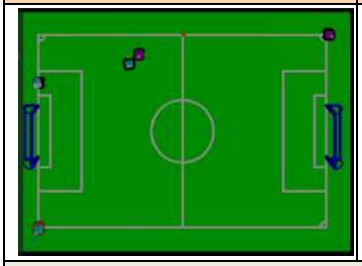
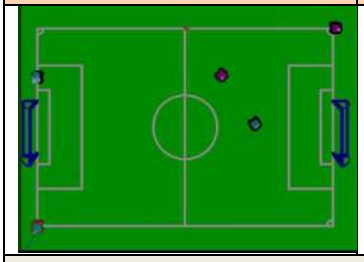
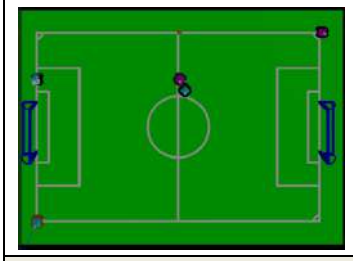
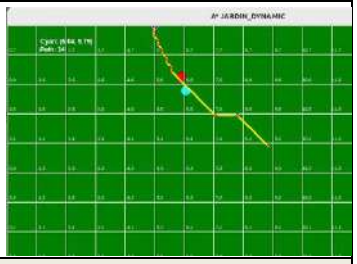
Tabel 4. 24 Hasil pengujian path planning A* Terdahulu pada simulasi Gazebo

Ke	Gambar	Jalur Path Planning (y,x)	Gambar Path
1	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (2.85, 3.00) 1 : (2.85, 4.35) 2 : (3.45, 4.95) 3 : (4.95, 4.95) 4 : (5.00, 5.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (2.85, 3.00) 1 : (2.85, 4.35) 2 : (3.45, 4.95) * 3 : (4.95, 4.95) *	
	Tidak Berhasil (Menabrak)		
2	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (2.85, 1.95) 1 : (2.85, 2.40) 2 : (2.40, 2.85) 3 : (2.40, 3.80) 4 : (1.95, 3.45) 5 : (1.95, 4.35) 6 : (2.55, 4.95) 7 : (2.55, 5.10)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (2.85, 1.95) 1 : (2.85, 2.40) 2 : (2.40, 2.85) 3 : (2.40, 3.80) * 4 : (1.95, 3.45) *	
	Tidak Berhasil (Menabrak)		

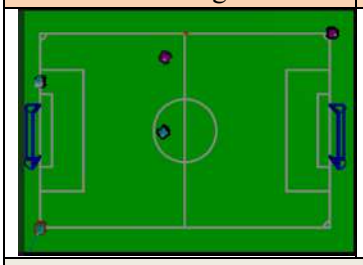
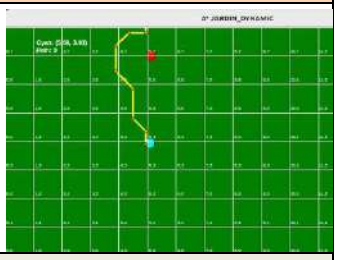
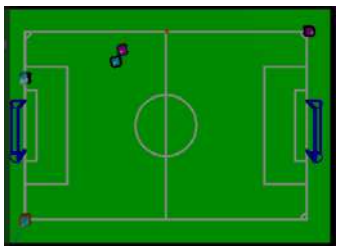
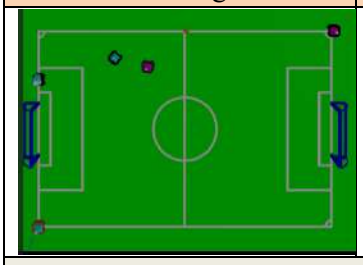
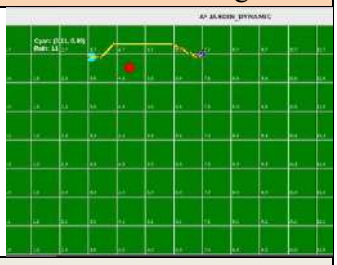
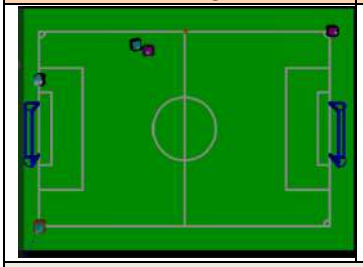
Tabel 4.24 Hasil pengujian *path planning* A* Terdahulu pada simulasi Gazebo (Lanjutan)

Ke	Gambar	Jalur Path Planning (x,y)	Gambar Path
3	**Gambar Sebelum Robot Lawan Bergerak**	**Jalur sebelum robot lawan bergerak**	**Gambar Path sebelum Robot Lawan Bergerak**
		0 : (2.85, 3.75) 1 : (3.30, 3.75) 2 : (3.75, 3.30) 3 : (4.85, 3.30) 4 : (6.45, 3.75) 5 : (6.75, 3.75) 6 : (6.90, 3.90) 7 : (7.00, 4.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	**Jalur robot lawan sudah bergerak**	**Gambar Path Robot Lawan Sudah Bergerak**
		0 : (2.85, 3.75) 1 : (3.30, 3.75) 2 : (3.75, 3.30) 3 : (4.85, 3.30) * 4 : (4.50, 2.85) *	
	Tidak Berhasil (Menabrak)		
4	**Gambar Sebelum Robot Lawan Bergerak**	**Jalur sebelum robot lawan bergerak**	**Gambar Path sebelum Robot Lawan Bergerak**
		0 : (1.80, 6.90) 1 : (2.25, 6.90) 2 : (2.70, 7.35) 3 : (3.30, 7.35) 4 : (5.25, 7.65) 5 : (5.40, 7.80) 6 : (5.50, 8.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	**Jalur robot lawan sudah bergerak**	**Gambar Path Robot Lawan Sudah Bergerak**
		0 : (1.80, 6.90) 1 : (2.25, 6.90) 2 : (2.70, 7.35) 3 : (3.30, 7.35) 4 : (3.45, 7.50) * 5 : (4.80, 7.50) *	
	Tidak Berhasil (Menabrak)		

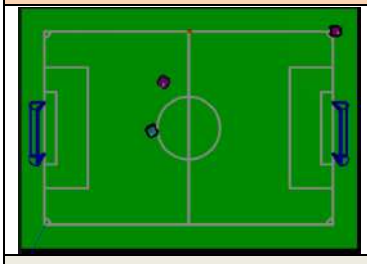
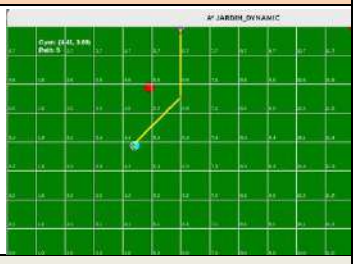
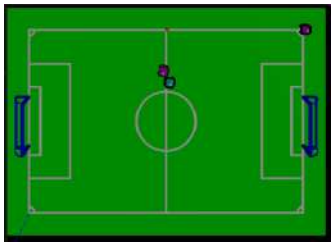
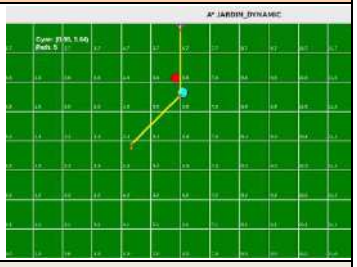
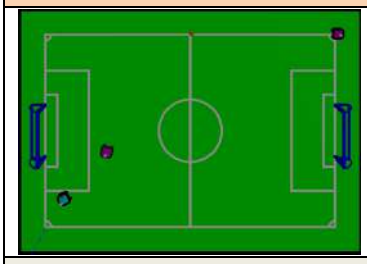
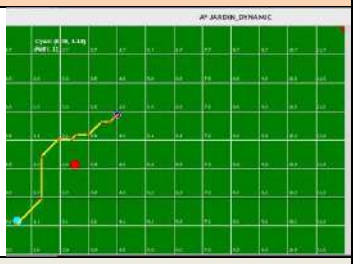
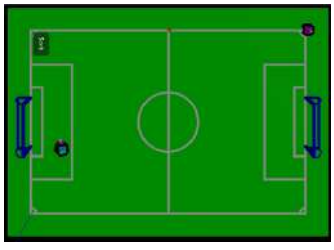
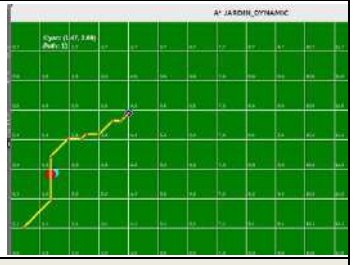
Tabel 4.24 Hasil pengujian *path planning* A* Terdahulu pada simulasi Gazebo (Lanjutan)

Ke -	Gambar	Jalur Path Planning (x,y)	Gambar Path
5	**Gambar Sebelum Robot Lawan Bergerak**	**Jalur sebelum robot lawan bergerak**	**Gambar Path sebelum Robot Lawan Bergerak**
		0 : (5.85, 5.40) 1 : (5.70, 5.55) 2 : (5.55, 5.55) 3 : (5.25, 5.85) 4 : (4.50, 5.85) 5 : (3.90, 6.45) 6 : (3.90, 7.80) 7 : (4.00, 8.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	**Jalur robot lawan sudah bergerak**	**Gambar Path Robot Lawan Sudah Bergerak**
		0 : (5.85, 5.40) 1 : (5.70, 5.55) 2 : (5.55, 5.55) 3 : (5.25, 5.85) * 4 : (4.50, 5.85) *	
	Tidak Berhasil (Menabrak)		
6	**Gambar Sebelum Robot Lawan Bergerak**	**Jalur sebelum robot lawan bergerak**	**Gambar Path sebelum Robot Lawan Bergerak**
		0 : (8.85, 3.90) 1 : (7.80, 4.95) 2 : (7.05, 4.95) 3 : (5.55, 6.45) 4 : (5.55, 6.60) 5 : (5.40, 6.75) 6 : (5.10, 7.50) 7 : (4.95, 7.65)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	**Jalur robot lawan sudah bergerak**	**Gambar Path Robot Lawan Sudah Bergerak**
		0 : (8.85, 3.90) 1 : (7.80, 4.95) * 2 : (7.05, 4.95) *	
	Tidak Berhasil (Menabrak)		

Tabel 4.24 Hasil pengujian *path planning* A* Terdahulu pada simulasi Gazebo (Lanjutan)

Ke -	Gambar	Jalur Path Planning (x,y)	Gambar Path
7	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (4.95, 3.90) 1 : (4.95, 4.35) 2 : (4.50, 4.80) 3 : (4.50, 5.85) 4 : (3.90, 6.45) 5 : (4.95, 7.80) 6 : (5.00, 9.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (4.95, 3.90) 1 : (4.95, 4.35) 2 : (4.50, 4.80) *	
	Tidak Berhasil (Menabrak)		
8	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (3.00, 6.90) 1 : (3.45, 6.90) 2 : (6.00, 7.35) 3 : (6.15, 7.20) 4 : (6.30, 7.20) 5 : (6.75, 6.90) 6 : (6.90, 6.90) 7 : (7.00, 7.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (3.00, 6.90) 1 : (3.45, 6.90) 2 : (3.90, 7.35) 3 : (6.00, 7.35) * 4 : (6.15, 7.20) *	
	Tidak Berhasil (Menabrak)		

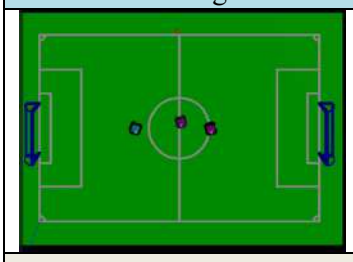
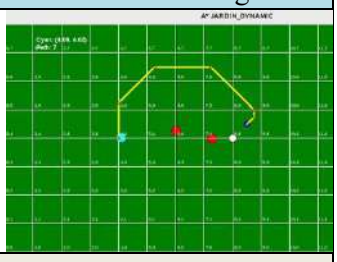
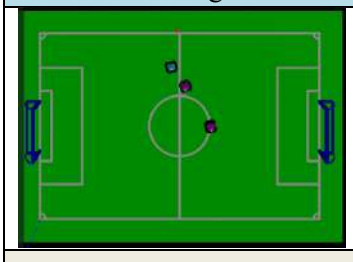
Tabel 4.24 Hasil pengujian *path planning* A* Terdahulu pada simulasi Gazebo (Lanjutan)

Ke -	Gambar	Jalur Path Planning (x,y)	Gambar Path
9	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (4.35, 3.75) 1 : (4.35, 3.90) 2 : (6.00, 5.55) 3 : (6.00, 7.80) 4 : (6.00, 8.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (4.35, 3.75) 1 : (4.35, 3.90) 2 : (6.00, 5.55) *	
	Tidak Berhasil (Menabrak)		
10	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (0.45, 1.05) 1 : (1.35, 1.95) 2 : (1.35, 3.45) 3 : (3.45, 4.65) 4 : (3.75, 4.65) 5 : (3.98, 4.80) 6 : (4.80, 4.90)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (0.45, 1.05) 1 : (1.35, 1.95) 2 : (1.35, 3.45) 3 : (1.95, 4.05) 4 : (2.48, 4.05) 5 : (2.55, 4.20) * 6 : (3.08, 4.20) *	
	Tidak Berhasil (Menabrak)		

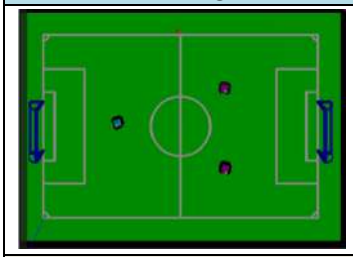
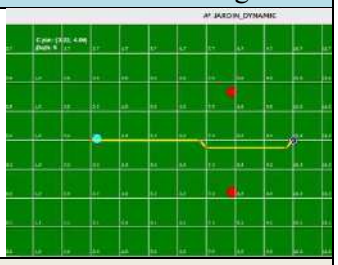
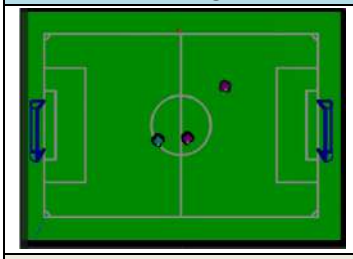
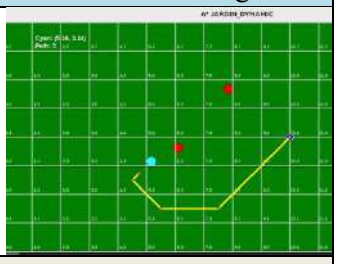
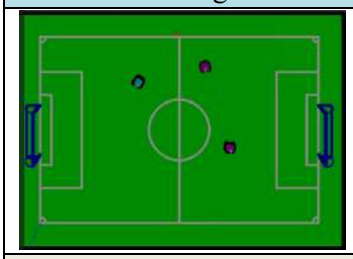
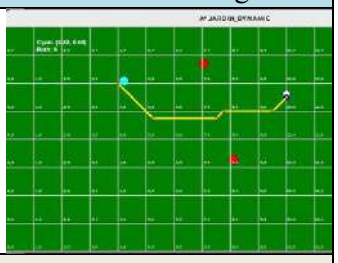
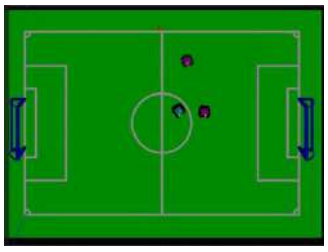
Tabel 4. 25 Hasil Pengujian *Path Planning* A* *Dynamic Opponent*

Percobaan ke -	Awal (x,y)	Tujuan (x,y)	Halangan 1 (x,y)	Halangan 2 (x,y)	Halangan Bergerak (x,y)	Waktu	Jarak
1	(4,4)	(9,4)	(6,4)	(7,4)	(6,6)	25,54	**5.00**
2	(3,4)	(10,4)	(8,5.5)	(8,2)	(6,4)	25,31	**7.00**
3	(4,6)	(10,5.6)	(7,6.5)	(8,3.5)	(8,5)	25,36	**6.01**
4	(4,2)	(10,4)	(7,2)	(8,4)	(5,3)	38,51	**6.32**
5	(4,1)	(10,3)	(7,1)	(6,4)	(8,3)	38,22	**6.32**
6	(1,1)	(8,4)	(3,2)	(3,3)	(6,4)	35	**7.62**
7	(3,7)	(8,4)	(4,7)	(3,5.5)	(3,3)	27,51	**5.83**
8	(10,7)	(9,4)	(10,6)	(11,6)	(8,6)	56,17	**3.16**
9	(10,1)	(10,5)	(11,2)	(10,2)	(9,3)	30,21	**4.00**
10	(4,4)	(9,3)	(6,3)	(7,2)	(6,4)	27,59	**5.10**

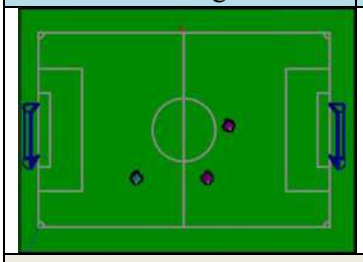
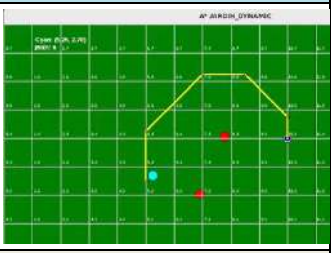
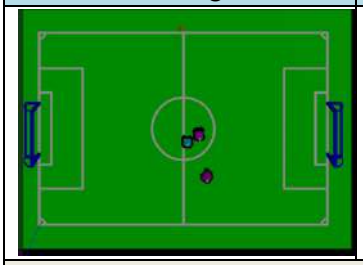
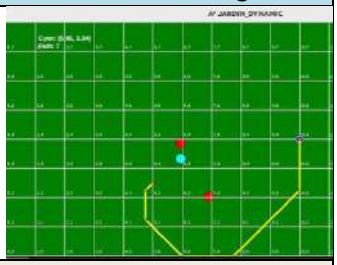
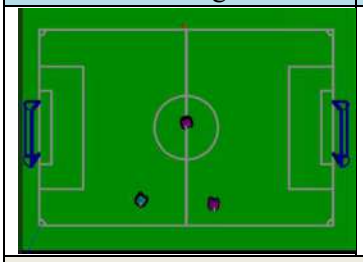
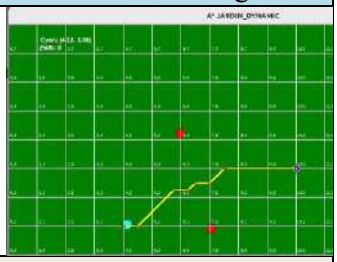
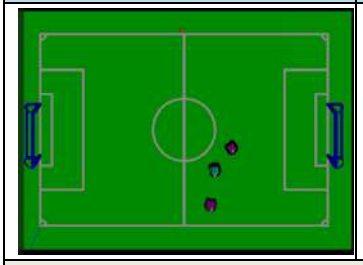
Tabel 4. 26 Hasil Pengujian Path Planning A* Dynamic Opponent pada simulasi Gazebo

Ke -	**Gambar**	**Jalur Path Planning (x,y)**	**Gambar Path**
1	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (4.08, 4.08) 1 : (4.08, 5.25) 2 : (5.25, 6.50) 3 : (7.25, 6.50) 4 : (8.75, 5.00) 5 : (8.75, 4.75) 6 : (8.50, 4.50)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (5.25, 6.50) 1 : (5.00, 6.75) 2 : (5.00, 7.00) 3 : (5.75, 7.75) 4 : (6.75, 7.75) 5 : (8.75, 4.75) 6 : (8.50, 4.50)	
	Berhasil		

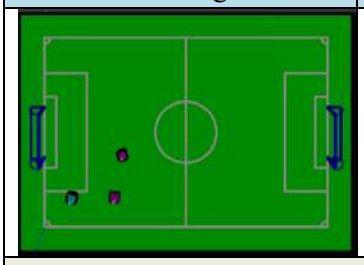
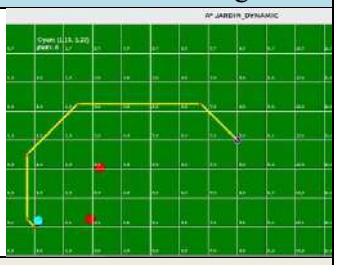
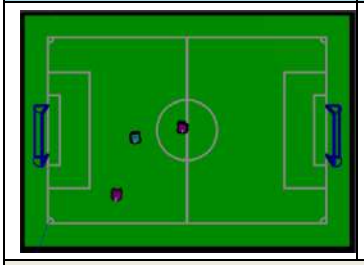
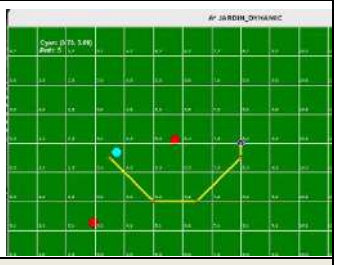
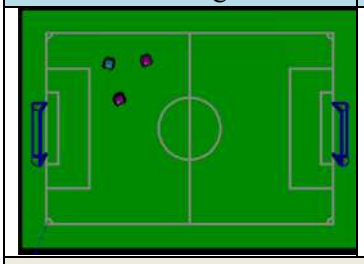
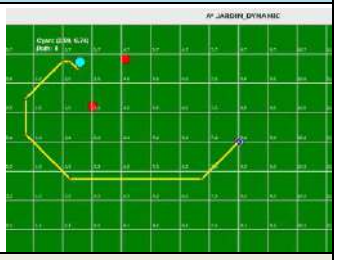
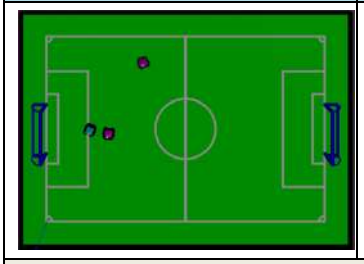
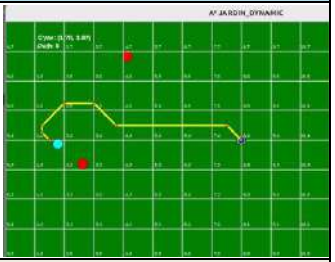
Tabel 4.26 *Hasil Pengujian Path Planning A* Dynamic Opponent* pada simulasi Gazebo (Lanjutan)

Ke -	Gambar	Jalur Path Planning (x,y)	Gambar Path
2	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (3.00, 4.00) 1 : (6.75, 4.80) 2 : (7.00, 3.75) 3 : (9.75, 3.75) 4 : (10.00, 4.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (4.75, 2.75) 1 : (4.50, 2.50) 2 : (5.50, 1.50) 3 : (7.50, 1.50) 4 : (10.00, 4.00)	
	Berhasil		
3	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (4.80, 6.00) 1 : (5.25, 4.75) 2 : (7.50, 4.75) 3 : (7.75, 5.00) 4 : (9.50, 5.00) 5 : (10.00, 5.50)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (6.25, 4.75) 1 : (5.75, 4.25) 2 : (5.75, 3.25) 3 : (7.00, 2.80) 4 : (8.50, 2.80) 5 : (10.00, 3.50) 6 : (10.00, 5.50)	
	Berhasil		

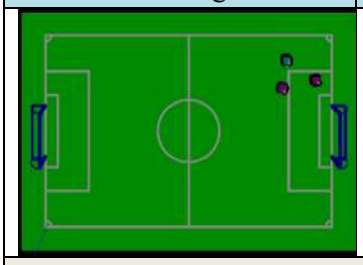
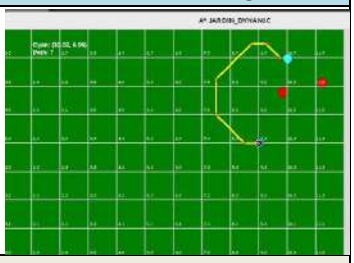
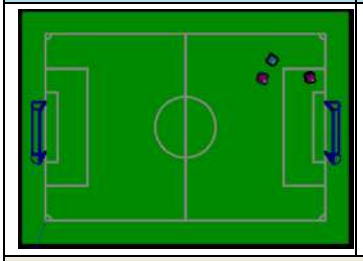
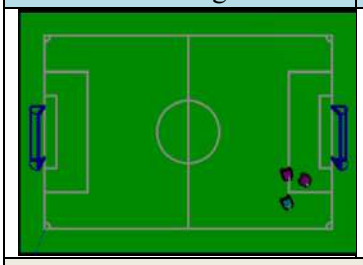
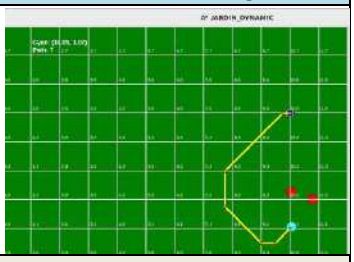
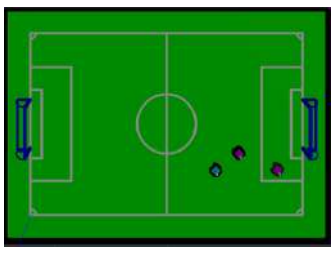
Tabel 4.26 *Hasil Pengujian Path Planning A* Dynamic Opponent* pada simulasi Gazebo
(Lanjutan)

Ke -	Gambar	Jalur Path Planning (x,y)	Gambar Path
4	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (5.00, 2.50) 1 : (5.00, 4.25) 2 : (7.00, 6.25) 3 : (8.50, 6.25) 4 : (10.00, 4.75) 5 : (10.00, 4.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (5.00, 2.50) 1 : (4.75, 2.25) 2 : (4.75, 1.25) 3 : (6.00, 0.00) 4 : (7.75, 0.00) 5 : (10.00, 2.25) 6 : (10.00, 4.00)	
	Berhasil		
5	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (4.08, 1.08) 1 : (4.50, 1.08) 2 : (5.75, 2.25) 3 : (6.25, 2.25) 4 : (6.58, 2.50) 5 : (7.00, 2.50) 6 : (7.58, 3.00) 7 : (10.00, 3.80)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (7.00, 2.25) 1 : (6.50, 2.25) 2 : (5.75, 3.00) 3 : (5.75, 4.25) 4 : (7.00, 5.50) 5 : (8.50, 5.50) 6 : (10.00, 4.00) 7 : (10.00, 3.80)	
	Berhasil		

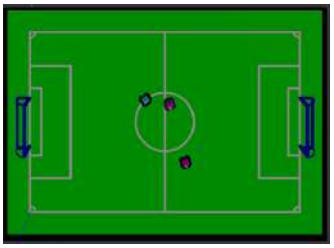
Tabel 4.26 *Hasil Pengujian Path Planning A* Dynamic Opponent* pada simulasi Gazebo (Lanjutan)

Ke -	Gambar	Jalur Path Planning (x,y)	Gambar Path
6	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (1.00, 1.00) 1 : (0.75, 1.25) 2 : (0.75, 3.50) 3 : (2.50, 5.25) 4 : (6.75, 5.25) 5 : (8.00, 4.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (3.50, 3.50) 1 : (5.00, 2.00) 2 : (5.50, 2.00) 3 : (8.00, 3.50) 4 : (8.00, 4.00)	
	Berhasil		
7	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (2.50, 6.50) 1 : (2.25, 6.75) 2 : (2.80, 6.75) 3 : (6.75, 5.50) 4 : (7.25, 4.25) 5 : (6.75, 2.75) 6 : (8.00, 4.80)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (1.50, 4.80) 1 : (1.25, 4.25) 2 : (1.25, 4.50) 3 : (2.80, 5.25) 4 : (3.80, 5.25) 5 : (3.75, 4.50) 6 : (7.50, 4.50) 7 : (8.00, 4.80)	
	Berhasil		

Tabel 4.26 *Hasil Pengujian Path Planning A* Dynamic Opponent* pada simulasi Gazebo
(Lanjutan)

Ke -	Gambar	Jalur Path Planning (x,y)	Gambar Path
8	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (9.75, 7.80) 1 : (9.25, 7.50) 2 : (8.75, 7.50) 3 : (7.50, 6.25) 4 : (7.50, 5.80) 5 : (8.50, 4.80) 6 : (9.80, 4.80)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (9.00, 7.00) 1 : (9.50, 6.50) 2 : (9.50, 4.50) 3 : (9.80, 4.80)	
	Berhasil		
9	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (10.00, 1.00) 1 : (9.50, 0.50) 2 : (9.80, 0.50) 3 : (7.75, 1.75) 4 : (7.75, 3.80) 5 : (9.75, 5.80) 6 : (10.00, 5.00)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (8.50, 1.50) 1 : (8.25, 1.25) 2 : (8.80, 1.25) 3 : (7.25, 2.00) 4 : (7.25, 3.50) 5 : (8.75, 5.80) 6 : (10.00, 5.00)	
	Berhasil		

Tabel 4.26 *Hasil Pengujian Path Planning A* Dynamic Opponent* pada simulasi Gazebo (Lanjutan)

Ke -	Gambar	Jalur Path Planning (x,y)	Gambar Path
10	Gambar Sebelum Robot Lawan Bergerak	Jalur sebelum robot lawan bergerak	Gambar Path sebelum Robot Lawan Bergerak
		0 : (4.00, 4.00) 1 : (4.80, 4.25) 2 : (5.25, 5.50) 3 : (6.50, 5.50) 4 : (9.80, 3.80)	
	Berhasil		
	Gambar Robot Lawan Sudah Bergerak	Jalur robot lawan sudah bergerak	Gambar Path Robot Lawan Sudah Bergerak
		0 : (4.75, 4.50) 1 : (4.25, 5.80) 2 : (4.25, 5.50) 3 : (5.50, 6.75) 4 : (6.75, 6.75) 5 : (9.80, 4.50) 6 : (9.00, 3.80)	
	Berhasil		

Tabel 4.23 menyajikan hasil pengujian sistem navigasi robot menggunakan algoritma A* konvensional, di mana jalur pergerakan ditentukan hanya satu kali di awal berdasarkan kondisi lingkungan saat itu. Pada metode ini, robot akan mengikuti jalur yang telah direncanakan tanpa melakukan pembaruan meskipun terjadi perubahan posisi rintangan, khususnya robot lawan yang bersifat dinamis.

Berdasarkan hasil pengujian pada Tabel 4.24, seluruh percobaan menunjukkan bahwa robot mampu mencapai titik tujuan ketika kondisi lingkungan masih sesuai dengan perencanaan awal. Waktu tempuh yang diperoleh berada pada kisaran 21,51 hingga 28,79 detik, dengan jarak tempuh antara 2,83 hingga 5,66 meter, yang menunjukkan bahwa algoritma ini cukup efisien dalam menghasilkan jalur. Namun, ketika robot lawan bergerak dan mengubah posisinya, jalur yang telah ditentukan tidak lagi sesuai dengan kondisi aktual. Karena tidak adanya mekanisme pembaruan jalur, robot tetap mengikuti lintasan awal sehingga pada beberapa percobaan terjadi kegagalan berupa tabrakan atau jalur yang melewati area tidak aman. Hal ini menunjukkan bahwa A* konvensional memiliki keterbatasan yang signifikan dalam menghadapi lingkungan yang dinamis.

Sebaliknya, Tabel 4.25 menunjukkan hasil pengujian sistem navigasi robot menggunakan algoritma A* dengan pendekatan dynamic replanning. Pada metode ini, jalur pergerakan tidak bersifat tetap, melainkan diperbarui secara berkala selama robot bergerak. Algoritma secara aktif mempertimbangkan perubahan posisi robot lawan sebagai obstacle dinamis dan melakukan perhitungan ulang jalur ketika terdapat potensi konflik pada lintasan.

Tabel 4.26 menunjukkan hasil pengujian bahwa seluruh 10 percobaan berhasil mencapai titik tujuan (100% keberhasilan) tanpa terjadi tabrakan. Waktu tempuh pada metode ini berada pada kisaran 25,31 hingga 56,17 detik, dengan jarak tempuh antara 3,16 hingga 7,62 meter. Meskipun nilai waktu dan jarak tempuh lebih besar dibandingkan A* konvensional, hal ini merupakan konsekuensi dari proses penyesuaian jalur selama navigasi berlangsung untuk menghindari rintangan. Jalur yang dihasilkan menjadi lebih fleksibel, adaptif, dan mampu meningkatkan kemampuan robot dalam menghindari obstacle pada lingkungan yang dinamis.

Dengan demikian, pada skenario pengujian yang dilakukan, A* dengan *dynamic replanning* menunjukkan keunggulan dalam kemampuan beradaptasi terhadap perubahan posisi obstacle dan berhasil menghindari tabrakan pada seluruh percobaan. Namun, kemampuan tersebut disertai dengan peningkatan waktu tempuh dan panjang lintasan dibandingkan A* konvensional. Hasil ini menunjukkan adanya *trade-off* antara kemampuan adaptasi terhadap *obstacle* dinamis dengan efisiensi waktu dan panjang lintasan.

4.5. Pengujian *Dynamic Path Planning* A* pada Robot Penyerang

Pada subbab ini dilakukan pengujian terhadap implementasi sistem *Dynamic Path Planning A** pada robot penyerang mengevaluasi kemampuan sistem dalam menghasilkan lintasan dan melakukan perencanaan ulang *(dynamic replanning)* ketika terjadi perubahan kondisi lingkungan akibat pergerakan robot lawan. Pengujian dilakukan pada sistem yang telah terintegrasi dengan ROS 2, estimasi posisi relatif objek berbasis kamera omnidirectional 360°, serta mekanisme komunikasi publish-subscribe sebagai media pertukaran data antar node.

Tahapan pengujian diawali dengan implementasi sistem yang meliputi

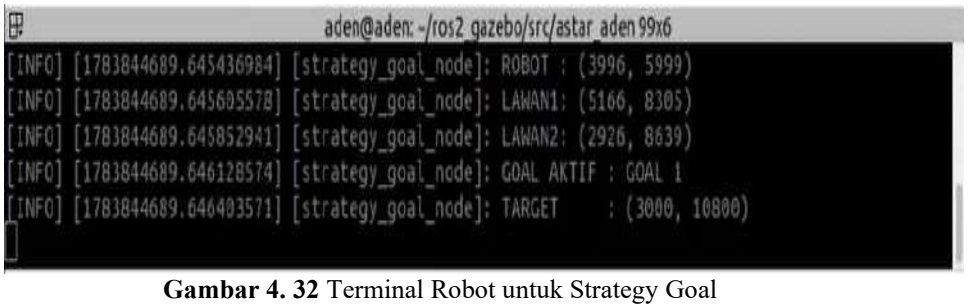
strategi penentuan goal (*strategy goal*), mekanisme *A dynamic replanner**, serta antarmuka grafis (GUI) untuk memvisualisasikan waypoint dan lintasan hasil perencanaan. Selanjutnya dilakukan pengujian terhadap kemampuan sistem dalam memperbarui jalur secara dinamis, diikuti dengan analisis performa lintasan berdasarkan perubahan jalur, keberhasilan robot mencapai tujuan, serta performa sistem selama proses navigasi.

4.5.1. Integrasi Sistem *Dynamic Path Planning* A* pada Robot Penyerang

a) Integrasi Strategy Goal

Setelah informasi posisi robot, bola, dan robot lawan diperoleh dari sistem estimasi posisi relatif berbasis kamera omnidirectional, tahap selanjutnya adalah menentukan koordinat tujuan (*goal*) yang akan dicapai robot penyerang. Proses ini dilakukan oleh node Strategy Goal yang berfungsi sebagai pengambil keputusan awal sebelum algoritma A* melakukan perencanaan lintasan. Penentuan goal dilakukan dengan mempertimbangkan posisi kedua robot lawan sehingga robot dapat memilih sisi gawang yang memiliki peluang serangan lebih besar dan risiko tabrakan yang lebih kecil.

Pada penelitian ini digunakan dua kandidat koordinat tujuan, yaitu Goal 1 pada koordinat (3000, 10800) dan Goal 2 pada koordinat (5200, 10800)**.** Node Strategy Goal menerima data posisi robot sendiri dan kedua robot lawan melalui mekanisme komunikasi *publish-subscribe* pada ROS 2. Berdasarkan data tersebut, sistem melakukan evaluasi terhadap kondisi lapangan untuk menentukan goal yang paling sesuai. Hasil pemilihan goal kemudian dipublikasikan ke node A* *Replanner* sebagai target navigasi robot.



Gambar 4. 32 Terminal Robot untuk Strategy Goal
(Penulis, 2026)

Gambar 4.32 memperlihatkan bahwa robot berada pada koordinat (3996, 5999), sedangkan robot lawan pertama dan kedua masing-masing berada pada

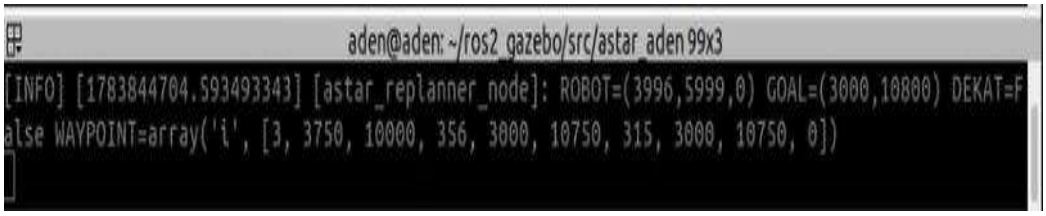
koordinat (5166, 8305) dan (2926, 8639). Berdasarkan konfigurasi tersebut, sistem memilih Goal 1 pada koordinat (3000, 10800) sebagai target navigasi karena jalur menuju koordinat tersebut memiliki potensi hambatan yang lebih kecil dibandingkan Goal 2. Informasi goal yang telah dipilih selanjutnya dikirimkan ke node A* Replanner untuk proses pembentukan lintasan berdasarkan fungsi evaluasi algoritma A*.

b) Integrasi A* Replanner

Setelah node **Strategy Goal** berhasil menentukan koordinat tujuan (*goal*), tahap berikutnya adalah melakukan perencanaan lintasan menggunakan node *A* Replanner*. Node ini bertugas menghasilkan lintasan dengan nilai minimum dari posisi robot menuju titik tujuan berdasarkan fungsi evaluasi algoritma A* yang mempertimbangkan jarak, obstacle, dan perubahan arah. Informasi posisi robot, posisi robot lawan, serta koordinat goal yang diterima melalui komunikasi *publish-subscribe* pada ROS 2 digunakan sebagai masukan dalam proses perencanaan lintasan.

Pada tahap ini, node A* *Replanner* membangun *occupancy grid* yang merepresentasikan kondisi lapangan berdasarkan posisi robot dan robot lawan. Robot lawan dimodelkan sebagai *dynamic obstacle*, sedangkan koordinat goal menjadi tujuan akhir navigasi. Berdasarkan peta tersebut, algoritma A* menghitung lintasan dengan nilai minimum berdasarkan fungsi evaluasi yang digunakan, yang terdiri atas beberapa waypoint sebagai acuan pergerakan robot menuju target. Selama robot bergerak, node A* *Replanner* terus memantau perubahan posisi robot lawan. Apabila salah satu robot lawan memasuki lintasan yang sedang dilalui, sistem akan secara otomatis melakukan *dynamic replanning*, yaitu menghitung ulang lintasan berdasarkan kondisi lingkungan terbaru sehingga robot tetap dapat menghindari rintangan tanpa menghentikan proses navigasi.

Gambar 4.33 menunjukkan keluaran (*output*) dari node A* *Replanner* setelah menerima informasi posisi robot dan koordinat goal dari node *Strategy Goal.*



Gambar 4. 33 Tampilan A* *Replanner* untuk Waypoint Robot
(Penulis, 2026)

Berdasarkan Gambar 4.33, terlihat bahwa node A* *Replanner* menerima posisi robot pada koordinat (3996, 5999) dengan target navigasi Goal 1 pada koordinat (3000, 10800). Berdasarkan informasi tersebut, algoritma A* menghasilkan tiga waypoint sebagai lintasan yang akan dilalui robot, yaitu WP1 (3750, 10000), WP2 (3000, 10750), dan WP3 (3000, 10800). Waypoint tersebut kemudian dipublikasikan ke node pengendali robot dan GUI lokal sebagai acuan navigasi selama robot bergerak menuju titik tujuan. Dengan mekanisme ini, sistem mampu memperbarui lintasan secara dinamis apabila terjadi perubahan posisi robot lawan selama proses navigasi.

c) Integrasi GUI Lokal *Dynamic Path Planning*

Setelah node Strategy Goal menentukan koordinat tujuan dan node A* Replanner menghasilkan lintasan dengan nilai minimum berdasarkan fungsi evaluasi A*, tahap selanjutnya adalah mengintegrasikan seluruh informasi tersebut ke dalam GUI lokal. Seluruh data yang ditampilkan pada GUI diperoleh melalui mekanisme komunikasi *publish-subscribe* pada ROS 2 sehingga setiap perubahan posisi robot, bola, robot lawan, maupun lintasan hasil perencanaan dapat diperbarui secara langsung.

GUI lokal berfungsi sebagai antarmuka pemantauan (*monitoring interface*) yang mengintegrasikan informasi dari berbagai node dalam sistem Dynamic Path Planning A*. Data posisi robot sendiri dan robot lawan diperoleh dari sistem persepsi berbasis kamera omnidirectional, sedangkan koordinat goal dan waypoint diperoleh dari node *Strategy Goal* dan *A Replanner**. Seluruh informasi tersebut kemudian divisualisasikan dalam bentuk peta lapangan sehingga memudahkan pengguna untuk mengamati proses navigasi robot, perubahan lintasan, serta mekanisme *dynamic replanning* ketika terjadi perubahan posisi robot lawan. Gambar 4.34 menunjukkan tampilan GUI lokal yang

digunakan pada implementasi sistem *Dynamic Path Planning* A*.

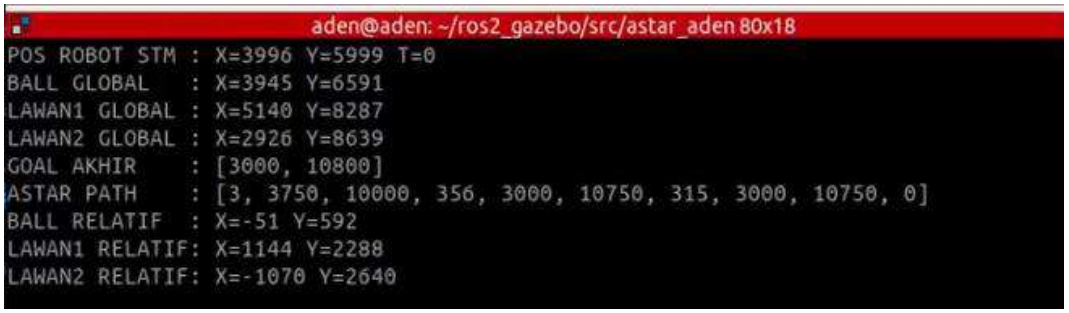
Gambar 4. 34 Tampilan GUI Lokal pada Robot Penyerang
(Penulis, 2026)

Berdasarkan Gambar 4.34, GUI menampilkan informasi posisi robot penyerang, bola, dan dua robot lawan selama proses navigasi. Robot berada pada koordinat (3996, 5999), bola pada (3945, 6591), sedangkan robot lawan berada pada koordinat (5140, 8287) dan (2908, 8700). GUI juga menampilkan Goal 1 pada koordinat (3000, 10800) sebagai target navigasi yang dipilih oleh node Strategy Goal.

Berdasarkan goal tersebut, algoritma A* menghasilkan tiga waypoint, yaitu WP1 (3750, 10000), WP2 (3000, 10750), dan WP3 (3000, 10800), yang divisualisasikan dengan garis berwarna kuning sebagai lintasan menuju target. Apabila posisi robot lawan berubah, node A* *Replanner* akan menghitung ulang lintasan sehingga waypoint dan jalur pada GUI diperbarui ketika terjadi perubahan posisi robot lawan. Hal ini menunjukkan bahwa proses integrasi antara sistem persepsi, Strategy Goal, *A* Replanner*, dan GUI lokal telah berjalan dengan baik.

Gambar 4.35 dibawah ini menunjukkan keluaran terminal yang berisi ringkasan data hasil integrasi seluruh node pada sistem Dynamic Path Planning A*. Informasi yang ditampilkan meliputi posisi robot (POS_ROBOT_STM), posisi bola (BALL_GLOBAL), posisi kedua robot lawan (LAWAN1_GLOBAL dan LAWAN2_GLOBAL), koordinat goal aktif (GOAL_AKHIR), waypoint hasil

algoritma A* (ASTAR_PATH), serta koordinat relatif objek terhadap robot (BALL_RELATIF, LAWAN1_RELATIF, dan LAWAN2_RELATIF). Data tersebut merupakan hasil komunikasi antar-node melalui mekanisme *publish-subscribe* pada ROS 2 yang selanjutnya digunakan sebagai masukan bagi sistem navigasi robot.



Gambar 4. 35 Tampilan Komunikasi antara Mikro dan Vision
(Penulis, 2026)

Berdasarkan Gambar 4.35, hasil pengujian seluruh informasi yang ditampilkan pada GUI telah sesuai dengan data yang diterima dari masing-masing node. Posisi robot, bola, robot lawan, koordinat goal, dan waypoint memiliki nilai yang konsisten sehingga menunjukkan bahwa proses pertukaran data antar-node berlangsung dengan baik. Hal ini membuktikan bahwa integrasi antara sistem persepsi, node Strategy Goal, node *A* Replanner*, dan GUI lokal telah berhasil diimplementasikan, sehingga seluruh informasi dapat dimanfaatkan selama proses navigasi sebagai dasar pengambilan keputusan robot penyerang.

4.5.2. Hasil Implementasi Pengujian *Dynamic Path Planning* A* pada Robot Penyerang

Setelah algoritma A* dengan mekanisme *dynamic replanning* berhasil diuji pada lingkungan Gazebo, tahap selanjutnya adalah implementasi dan pengujian pada robot penyerang di lapangan sebenarnya. Pengujian ini dilakukan untuk mengevaluasi kemampuan algoritma dalam menghasilkan lintasan yang adaptif terhadap perubahan posisi robot lawan.

Pada implementasi ini, sistem navigasi memperoleh informasi posisi robot sendiri, robot lawan, dan target melalui sistem visi berbasis kamera omnidirectional. Seluruh informasi koordinat dipublikasikan menggunakan ROS 2 dan diproses oleh node *path planning* untuk membentuk peta lingkungan

(*occupancy grid*). Robot lawan diperlakukan sebagai *dynamic obstacle*, sedangkan titik tujuan (*goal*) ditentukan berdasarkan strategi penyerangan menuju salah satu sisi gawang lawan. Dalam penelitian ini digunakan dua koordinat tujuan, yaitu Goal 1 pada koordinat (3000, 10800) dan Goal 2 pada koordinat (5200, 10800). Pemilihan goal dilakukan berdasarkan hasil evaluasi posisi robot lawan sehingga robot dapat memilih jalur dengan risiko tabrakan yang lebih kecil.

Selama robot bergerak menuju titik tujuan, algoritma A* secara terus-menerus memantau perubahan posisi robot lawan. Apabila salah satu robot lawan memasuki lintasan yang sedang dilalui, sistem akan melakukan dynamic replanning, yaitu menghitung kembali lintasan dengan nilai minimum berdasarkan kondisi lingkungan terbaru. Mekanisme tersebut memungkinkan robot menghindari rintangan tanpa menghentikan proses navigasi sehingga pergerakan menuju target tetap berlangsung secara aman dan efisien.

Parameter yang diamati pada pengujian ini meliputi posisi awal robot, koordinat tujuan (goal), posisi kedua robot lawan, jumlah waypoint, waktu tempuh robot menuju target, serta tingkat keberhasilan robot mencapai tujuan tanpa mengalami tabrakan. Pengujian dilakukan sebanyak delapan kali dengan variasi posisi awal robot dan posisi robot lawan untuk mengevaluasi konsistensi performa algoritma pada berbagai kondisi lapangan. Hasil pengujian tersebut dapat dilihat pada Gambar 4.36 dan Gambar 4.37.

Gambar 4. 36 Implementasi Pengujian Path Planning A* *Dynamic Opponent*
(Penulis, 2026)

Gambar 4. 37 Tampilan Pergerakan Robot
(Penulis, 2026)

Tabel 4. 27 Implementasi Path Planning A* Dynamic Replanning pada Robot Penyerang

Pengujian	Posisi Awal Robot (x,y) (mm)	Goal (x,y) (mm)	Robot Lawan 1 (x,y) (mm)	Robot Lawan 2 (x,y) (mm)	Jumlah Replanning	Interval replanning (ms)	Jarak (m)	Waktu Tempuh (s)	Status
1	(4000, 6000)	(3000, 10800)	(3500, 6200)	(2800, 8500)	3	0.5	5.18	24,58	Berhasil
2	(4000, 6000)	(5200, 10800)	(4700, 6800)	(5000, 9000)	3	0.5	5.02	23,91	Berhasil
3	(4000, 6000)	(3000, 10800)	(3000, 5900)	(3300, 7900)	3	0.5	5.76	26,47	Berhasil
4	(4000, 6000)	(5200, 10800)	(3900, 7000)	(4700, 9300)	3	0.5	5.54	25,76	Berhasil
5	(4000, 6000)	(5200, 10800)	(4800, 6000)	(4500, 8500)	3	0.5	4.96	22,84	Berhasil
6	(4000, 6000)	(3000, 10800)	(3300, 6500)	(2900, 9100)	3	0.5	5.61	24,95	Berhasil
7	(2000, 4000)	(3000, 10800)	(4000, 6200)	(4400, 8200)	3	0.5	7.08	31.82	Berhasil
8	(2000, 4000)	(5200, 10800)	(3600, 6100)	(4200, 8900)	3	0.5	7.46	33.27	Berhasil

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 4.27, robot penyerang berhasil mencapai koordinat tujuan pada seluruh skenario pengujian

tanpa mengalami tabrakan dengan robot lawan, sehingga tingkat keberhasilan sistem mencapai 100%. Waktu tempuh robot berada pada rentang 22,84–33,27 detik, sedangkan jarak tempuh berkisar antara 4,96–7,46 meter. Variasi waktu dan jarak tempuh tersebut dipengaruhi oleh perbedaan konfigurasi posisi kedua robot lawan pada setiap skenario, yang menyebabkan algoritma A* menghasilkan lintasan yang berbeda. Pada setiap skenario pengujian, mekanisme dynamic replanning dilakukan sebanyak tiga kali dan menghasilkan tiga waypoint sebagai acuan navigasi. Posisi waypoint yang terbentuk tidak selalu sama pada setiap skenario sehingga panjang lintasan yang ditempuh robot juga bervariasi. Mekanisme dynamic replanning memiliki interval pembaruan sebesar 0,5 ms selama proses navigasi. Interval tersebut menunjukkan periode pemanggilan mekanisme *replanning*.

Pada setiap skenario, robot bergerak dari posisi awal yang telah ditentukan menuju salah satu koordinat tujuan, yaitu Goal 1 (3000,10800) atau Goal 2 (5200,10800). Enam skenario pertama menggunakan posisi awal (4000,6000), sedangkan dua skenario terakhir menggunakan posisi awal (2000,4000). Apabila robot lawan berada di sekitar lintasan utama, algoritma A* akan membentuk waypoint yang mengarahkan robot melewati jalur alternatif untuk menghindari potensi tabrakan. Kondisi tersebut menyebabkan jarak dan waktu tempuh menjadi lebih besar dibandingkan ketika posisi robot lawan tidak menghalangi lintasan utama. Sebaliknya, apabila robot lawan berada di luar jalur yang direncanakan, robot dapat bergerak melalui lintasan yang lebih langsung sehingga menghasilkan waktu tempuh dan jarak yang lebih efisien.

Hasil implementasi pada robot nyata memperlihatkan bahwa algoritma tetap mampu beroperasi pada berbagai kondisi pengujian meskipun dipengaruhi oleh faktor yang tidak dijumpai pada lingkungan simulasi, seperti keterlambatan komunikasi ROS 2, ketidakakuratan lokalisasi, fluktuasi hasil deteksi kamera omnidirectional, serta karakteristik respons aktuator robot. Informasi posisi kedua robot lawan digunakan sebagai masukan dalam proses pembentukan lintasan sehingga robot dapat mengikuti waypoint yang telah direncanakan hingga mencapai titik tujuan tanpa mengalami tabrakan.

Secara keseluruhan, hasil pengujian menunjukkan bahwa algoritma A*

berhasil diimplementasikan pada robot penyerang dan mampu menyesuaikan lintasan terhadap perubahan posisi robot lawan pada skenario pengujian yang dilakukan. Seluruh skenario pengujian berhasil mencapai salah satu titik tujuan dengan tingkat keberhasilan sebesar **100%**. Selain itu, perubahan posisi robot lawan terbukti memengaruhi bentuk lintasan, jarak tempuh, dan waktu tempuh yang dihasilkan. Meskipun pada beberapa kondisi robot harus menempuh lintasan yang lebih panjang sebagai konsekuensi dari proses penghindaran rintangan, sistem tetap mampu mempertahankan keberhasilan navigasi tanpa terjadi tabrakan. Temuan tersebut menunjukkan bahwa metode yang diusulkan efektif diterapkan pada robot sepak bola KRSBI-B yang beroperasi pada lingkungan dinamis dengan posisi rintangan yang dapat berubah secara terus-menerus.

Gambar 4. 38 Proses Pengujian dari berbagai posisi robot lawan
(Penulis, 2026)

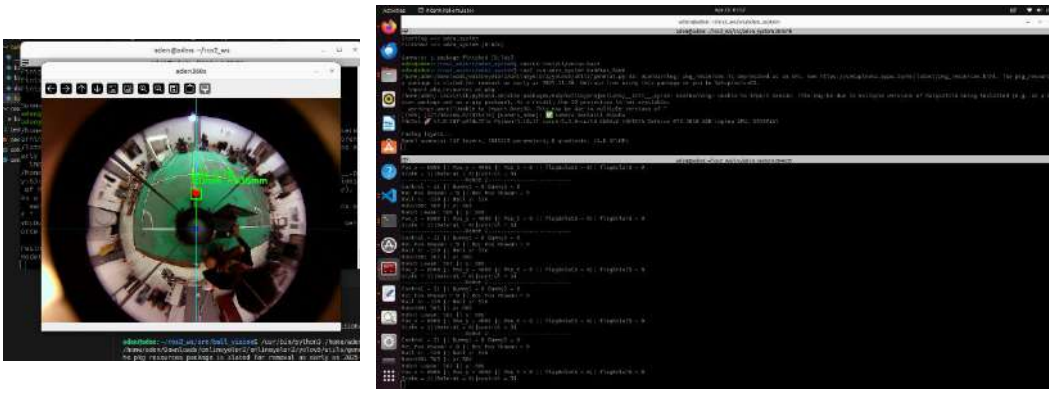
4.6. Integrasi Komunikasi ROS 2 dengan YOLOv5s

Gambar 4.32 menyajikan hasil implementasi sistem deteksi objek berbasis YOLOv5s yang telah terintegrasi dengan komunikasi ROS 2 pada robot sepak bola. Kamera omnidirectional 360° digunakan sebagai sumber masukan citra secara real-time, kemudian citra tersebut diproses menggunakan model YOLOv5s untuk mengidentifikasi objek berupa bola dan robot lawan.

Hasil deteksi ditampilkan dalam bentuk *bounding box* pada citra, lengkap dengan label objek dan nilai confidence. Selain itu, sistem juga menghitung posisi objek dalam bentuk koordinat relatif terhadap kamera, seperti posisi X dan Y yang ditampilkan pada layar. Informasi ini kemudian diproses lebih lanjut untuk menentukan arah dan jarak objek terhadap robot.

Data hasil deteksi tersebut selanjutnya dikirim melalui komunikasi ROS 2 dalam bentuk data numerik yang ditampilkan pada terminal, seperti koordinat bola, posisi robot, serta status tertentu yang digunakan dalam pengambilan keputusan.

Komunikasi ini memungkinkan setiap node dalam sistem untuk saling bertukar informasi selama proses deteksi, pengolahan data, dan pergerakan robot sehingga seluruh proses dapat berjalan secara terintegrasi, sehingga proses deteksi, pengolahan data, dan pergerakan robot dapat berjalan secara terintegrasi..



Gambar 4. 39 Ujicoba integrasi YOLOV5S dengan ROS 2 Humble
(Penulis, 2026)

Hasil pengujian menunjukkan bahwa sistem mampu mendeteksi objek dan mengirimkan data hasil deteksi ke node lain melalui komunikasi ROS 2 selama proses pengujian. Hal ini menunjukkan bahwa arsitektur berbasis ROS 2 mampu mendukung komunikasi data yang stabil dan efisien dalam sistem robotika.

Dengan adanya integrasi ini, robot tidak hanya mampu mengenali objek di sekitarnya, tetapi juga dapat memanfaatkan informasi tersebut untuk proses navigasi dan pengambilan keputusan secara otomatis. Dengan demikian, integrasi algoritma YOLOv5s dan middleware ROS 2 mampu membentuk sistem persepsi yang adaptif terhadap perubahan kondisi lapangan selama proses navigasi. Kombinasi tersebut memungkinkan robot memperoleh informasi posisi objek sebagai dasar pengambilan keputusan dalam proses navigasi dan strategi permainan pada lingkungan yang dinamis.

4.7. Integrasi Sistem Robot Penyerang dengan *Base Station*

Setelah sistem Dynamic Path Planning A* berhasil diintegrasikan pada robot penyerang, tahap selanjutnya adalah mengintegrasikan robot dengan Base Station sebagai pusat operator dan pengendali strategi selama pertandingan. Integrasi dilakukan menggunakan mekanisme komunikasi publisher–subscriber pada ROS 2 sehingga Base Station dapat mengirimkan informasi strategi

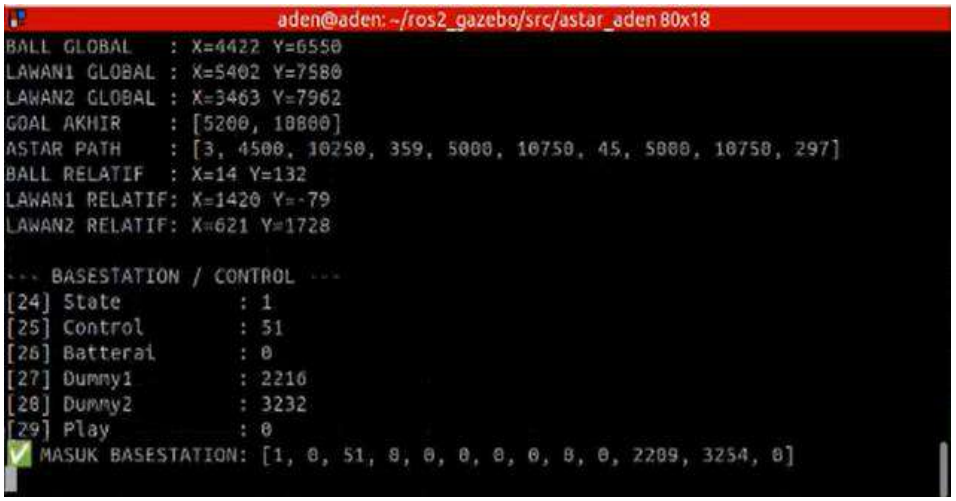
permainan, seperti kick-off, positioning, dan status permainan kepada robot selama proses komunikasi. Selain itu, robot juga mengirimkan informasi posisi, status navigasi, dan kondisi sistem kembali ke Base Station sehingga operator dapat memantau serta memvalidasi kesesuaian data yang ditampilkan pada GUI lokal robot dan GUI Base Station. Dengan mekanisme tersebut, komunikasi dua arah antara robot penyerang dan Base Station dapat berlangsung selama proses pertukaran data selama pengujian.

Gambar 4. 40 Tampilan GUI Lokal
(Penulis, 2026)

Gambar 4. 41 Tampilan *Base Station*
(Penulis, 2026)

Validasi komunikasi dari robot menuju Base Station dilakukan dengan

membandingkan informasi yang ditampilkan pada GUI lokal robot (*A* Dynamic Replanning*) dan GUI Base Station. Data yang diamati meliputi estimasi posisi robot, hasil deteksi posisi bola, posisi robot lawan, target *goal* aktif, serta informasi *waypoint* yang dikirimkan melalui node komunikasi ROS. Berdasarkan hasil pengujian, terlihat bahwa seluruh data yang ditampilkan pada Base Station memiliki kesesuaian nilai secara presisi dengan GUI lokal robot, di antaranya posisi robot sebesar X = 5273 mm dan Y = 5873 mm dengan orientasi T = 297°, koordinat posisi bola sebesar X = 4422 mm dan Y = 6380 mm, posisi robot lawan 1 sebesar X = 4947 mm dan Y = 9362 mm, serta target *goal* aktif sebesar X = 5200 mm dan Y = 10800 mm dengan total 3 titik *waypoint*. Kesesuaian nilai tersebut menunjukkan bahwa proses publikasi data (*publisher*) dari robot menuju Base Station Kesesuaian nilai tersebut menunjukkan bahwa proses publikasi data *(publisher)* dari robot menuju Base Station berhasil dilakukan selama pengujian.



Gambar 4. 42 Data Komunikasi dengan Base Station Posisi *Kick Off*
(Penulis, 2026)

Gambar 4. 43 Tampilan Status *Kick Off* pada *Base Station*
(Penulis, 2026)

Validasi komunikasi dari robot penyerang menuju Base Station dilakukan dengan membandingkan informasi yang ditampilkan pada terminal robot penyerang (*A* Dynamic Replanning*) dan GUI Base Station. Data yang diamati meliputi posisi bola global, posisi robot lawan, target *goal* akhir, jalur lintasan *A* Path*, serta status kontrol dari Base Station. Berdasarkan hasil pengujian, seluruh data terkonfirmasi sesuai secara presisi, di antaranya posisi bola global sebesar X = 4422 mm dan Y = 6550 mm, posisi robot lawan 1 sebesar X = 5402 mm dan Y = 7580 mm, posisi robot lawan 2 sebesar X = 3463 mm dan Y = 7962 mm, target *goal* akhir sebesar X = 5200 mm dan Y = 10800 mm, serta *A* Path* yang menghasilkan 3 titik *waypoint*.

Selanjutnya, pengujian penerimaan data (*subscriber*) dilakukan dengan memberikan perintah kontrol/*kick-off* dari GUI Base Station. Keberhasilan proses penerimaan data ditunjukkan pada terminal robot penyerang dengan berubahnya parameter State menjadi 1, nilai Control sebesar 51, serta terbacanya larik data MASUK BASESTATION: [1, 0, 51, 0, 0, 0, 0, 0, 0]. Hasil pengujian menunjukkan bahwa komunikasi dua arah berbasis mekanisme publisher–subscriber pada ROS 2 antara robot penyerang dan Base Station berhasil dilakukan, yang ditunjukkan oleh data yang diterima pada terminal robot sesuai dengan perintah yang dikirimkan dari Base Station. Kesesuaian nilai tersebut menunjukkan bahwa proses publikasi data dari robot menuju Base Station berhasil dilakukan selama pengujian.

(Halaman ini sengaja dikosongkan)

BAB 5
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem, maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Sistem estimasi jarak relatif berbasis kamera omnidirectional dan YOLOv5s berhasil diimplementasikan untuk memperoleh informasi posisi relatif bola dan robot lawan selama proses navigasi. Informasi posisi relatif diperoleh melalui proses deteksi objek dan estimasi jarak, kemudian digunakan sebagai masukan pada algoritma A* dengan mekanisme *dynamic replanning*. Hasil pengujian menunjukkan bahwa model YOLOv5s dengan optimizer SGD memiliki Average Precision (AP) sebesar 87,9% untuk deteksi bola dan 97,5% untuk deteksi robot lawan, dengan nilai mAP@0.5 sebesar 92,7%. Informasi hasil deteksi dan estimasi posisi relatif tersebut digunakan sebagai masukan pada algoritma A* dengan mekanisme dynamic replanning untuk mendukung proses navigasi robot..
2. Algoritma A* dengan mekanisme dynamic replanning berhasil diimplementasikan dan diuji baik pada lingkungan simulasi Gazebo maupun robot penyerang nyata. Pada pengujian simulasi, sistem berhasil mencapai titik tujuan pada seluruh 10 skenario dengan tingkat keberhasilan 100%, waktu tempuh 25,31–56,17 detik, dan panjang lintasan 3,16–7,62 meter. Selanjutnya, pada implementasi robot nyata dilakukan 8 skenario pengujian, di mana robot berhasil mencapai Goal 1 (3000,10800) maupun Goal 2 (5200,10800) dengan tingkat keberhasilan 100%, waktu tempuh 22,84–33,27 detik, dan jarak tempuh 4,96–7,46 meter tanpa mengalami tabrakan dengan robot lawan. Hasil tersebut menunjukkan bahwa algoritma mampu menyesuaikan lintasan terhadap perubahan posisi obstacle pada skenario pengujian robot nyata.
3. Integrasi ROS 2 dengan sistem deteksi objek dan estimasi jarak relatif, algoritma A*, dan strategi navigasi robot berhasil diimplementasikan menggunakan mekanisme komunikasi publisher–subscriber. Data posisi robot, robot lawan, target, waypoint, dan perintah kontrol berhasil dipertukarkan

antar-node selama pengujian sistem terintegrasi. Hasil implementasi menunjukkan bahwa seluruh komponen sistem dapat bekerja secara terintegrasi dalam mendukung proses navigasi robot pada lingkungan pertandingan yang dinamis.

5.2. Saran

Berdasarkan hasil penelitian yang telah dilakukan, beberapa saran untuk pengembangan penelitian selanjutnya adalah sebagai berikut.

1. Penelitian selanjutnya dapat mengembangkan sistem *path planning* menggunakan algoritma yang lebih adaptif, seperti *D Lite**, Hybrid A*, atau algoritma lainnya, sehingga proses pencarian jalur pada lingkungan yang lebih dinamis dapat dilakukan dengan lebih cepat.
2. Sistem deteksi robot lawan dapat dikembangkan dengan memanfaatkan LiDAR sebagai sensor tambahan atau pengganti kamera omnidirectional. Penggunaan LiDAR diharapkan mampu meningkatkan akurasi dan kestabilan deteksi *obstacle*, terutama pada kondisi pencahayaan yang kurang baik atau ketika terjadi occlusion pada kamera.
3. Penelitian selanjutnya dapat mengintegrasikan metode sensor fusion antara kamera omnidirectional, LiDAR, Gyro, dan encoder roda untuk meningkatkan akurasi lokalisasi robot serta kualitas informasi lingkungan yang digunakan pada proses path planning.
4. Pengembangan berikutnya dapat menambahkan prediksi pergerakan robot lawan *(opponent motion prediction)* menggunakan metode *machine learning* sehingga robot dapat mengantisipasi pergerakan obstacle sebelum melakukan proses *replanning.*

DAFTAR PUSTAKA

Adi Rahmad Ramadhan, Khumaidi, A., Mustika Kurnia Mayangsari, Mat Syai'in, Imam Sutrisno, & Aulia Rahma Annisa. (2025). Penerapan Harris Corner Detection dan YOLOV5S pada Kamera Stereo Vision untuk Estimasi Jarak Robot Sepak Bola Beroda KRSBI-B. *Jurnal Elektronika Dan Otomasi Industri*, *12*(1), 111–122. https://doi.org/10.33795/elkolind.v12i1.7254

Al-Batati, A. S., Koubaa, A., Abdelkader, M., Gabr, K., & Aloqaily, H. (2025). *ROS 2 in a Nutshell: A Survey*. *V3*(October). https://doi.org/10.20944/preprints202410.1204.v2

Alfian, R. I., Ma'Arif, A., & Sunardi, S. (2021). Noise reduction in the accelerometer and gyroscope sensor with the Kalman filter algorithm. *Journal of Robotics and Control (JRC)*, *2*(3), 180–189. https://doi.org/10.18196/jrc.2375

Ananda, D. C. (2023). *SIMULATOR ROBOT SEPAK BOLA BERODA BERBASIS ROBOT OPERATING SYSTEM* (Vol. 7, Issue 2). http://repository.radenintan.ac.id/11375/1/PERPUS PUSAT.pdf%0Ahttp://business-law.binus.ac.id/2015/10/08/pariwisata-syariah/%0Ahttps://www.ptonline.com/articles/how-to-get-better-mfi-results%0Ahttps://journal.uir.ac.id/index.php/kiat/article/view/8839

As'ari, N. H. A., Agus Khumaidi, Rini Indarti, Afif Zuhri Arfianto, Ryan Yudha Adhitya, Hendro Agus Widodo, Zindhu Maulana Ahmad Putra, & Dimas Pristovani Riananda. (2024). Pengendalian Kecepatan Tendangan Robot Sepak Bola Beroda Menggunakan Metode Decision Tree. *Jurnal Elektronika Dan Otomasi Industri*, *11*(2), 628–639. https://doi.org/10.33795/elkolind.v11i2.5192

Bayu, M., Maulana, R., Dewatama, D., & Fauziyah, M. (2024). *Kontrol Kecepatan Motor Induksi 3 Fasa Berbasis Arduino Mega 2560 Pada Lift 4 Lantai*. *2*(6), 71–80. https://ejournal.warunayama.org/kohesi

Bonci, A., Gaudeni, F., Giannini, M. C., & Longhi, S. (2023). Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey. In *Applied Sciences (Switzerland)* (Vol. 13, Issue 23). https://doi.org/10.3390/app132312796

Choi, H., Xiang, Y., & Kim, H. (2021). PiCAS: New design of priority-driven chain-aware scheduling for ROS2. *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, *2021-May*, 251–263. https://doi.org/10.1109/RTAS52030.2021.00028

Colab. (2025). *Logo Google Colab*. https://colab.research.google.com/

CUDA (2025). https://developer.nvidia.com/cuda/wsl

Darmawan, W., Basuki Rahmat, M., Khumaidi, A., Yudha Adhitya, R., & Pristovani Riananda, D. (2023a). Perancangan Strategi Keputusan Robot Sepak Bola Beroda menggunakan Metode Decision Tree. In *Jurnal Elektronika dan Otomasi Industri* (Vol. 10, Issue 2). https://doi.org/10.33795/elkolind.v10i2.3020

Darmawan, W., Basuki Rahmat, M., Khumaidi, A., Yudha Adhitya, R., & Pristovani Riananda, D. (2023b). Perancangan Strategi Keputusan Robot Sepak Bola Beroda menggunakan Metode Decision Tree. *Jurnal Elektronika*

Dan Otomasi Industri, *10*(2), 175–182. https://doi.org/10.33795/elkolind.v10i2.3020

Hendri, H., Hoki, L., Agusman, V., & Aryanto, D. (2021). Penerapan Machine Learning Untuk Mengategorikan Sampah Plastik Rumah Tangga. *Jurnal TIMES*, *10*(1), 1–5. https://doi.org/10.51351/jtm.10.1.2021645

Ikram, F. D., Khumaidi, A., Rahmat, M. B., Endrasmono, J., Syai, M., & Pristovani, D. (2024). *Deteksi Objek di Lapangan pada Robot Sepakbola Beroda Menggunakan Metode*. *11*, 604–611.

Irwansyah, A., Putra, A. Z., & Akbar, R. (2023). Omnidirectional Camera untuk Positioning Robot Soccer dengan Metode 2-Fixed-Point. *Techné : Jurnal Ilmiah Elektroteknika*, *22*(2), 273–284. https://doi.org/10.31358/techne.v22i2.374

Kaladewa, Y., & Santoso, K. A. (2022). Implementasi Sensor Kemiringan Sudut Untuk Alat Bantu (Grab) Gantry Luffing Crane (Glc). *Jurnal Kajian Teknik Elektro*, *6*(2), 62–69. https://doi.org/10.52447/jkte.v6i2.5726

Khumaidi, A., Yaqin, M. A., Adhitya, R. Y., Irsyad, S. M., & Saputra, M. J. (2025). *Penerapan Rotary Encoder sebagai Sistem Odometry untuk Navigasi Robot Sepak Bola Beroda*.

Kurniawan, ade agung, Hermanto, & Rahmawati, S. (2024). *Jurnal KomtekInfo Smart Tong Sampah Pendeteksi Otomatis Sampah Organik &*. *11*(3), 163–172. https://doi.org/10.35134/komtekinfo.v11i3.564.163

Kurniawan, F., Erdata Nasution, M. R., Dinaryanto, O., & Lasmadi, L. (2021). Penentuan Orientasi dan Translasi Gerakan UAV menggunakan Data Fusion berbasis Kalman Filter. *Avitec*, *3*(2), 99–115. https://doi.org/10.28989/avitec.v3i2.890

Leadtree, W., & Needs, W. C. (2025). *MITOR : Jurnal Teknik Elektro*. 39–46.

Marinho, M. M. (2023). (Murilo's) ROS2 Tutorial. *Github*. https://ros2-tutorial.readthedocs.io/en/latest/index.html

Marpaung, F., Aulia, F., & Nabila, R. C. (2022). *Computer Vision Dan Pengolahan Citra Digital*. www.pustakaaksara.co.id

Mobaiyen, S., & Ashjaei, M. (2022). *Systematic Gap Analysis of Robot Operating System (ROS 2) in Real-time Systems*.

Mursalim, M., & Aprilia, T. (2024). Segmentasi Citra Digital berbasis histogram: Systematic Literature Review. *Jurnal Teknik Informatika Dan Desain Komunikasi …*, *3*, 58–66.

Musakkir, M., Nurtanio, I., & Zainuddin, Z. (2024). Estimasi Bobot Ikan Bandeng Menggunakan Segmentasi Citra Biner. *Jurnal Ilmiah Sistem Informasi Dan Teknik Informatika (JISTI)*, *7*(1), 24–35. https://doi.org/10.57093/jisti.v7i1.181

Normalisa, Rachmaniar, A., Diana, D., Saefudin, M., & Parulian, R. (2022). *Application Of Computer Vision Detection Of Apples And Oranges Using Python Language*. Journal of Information System, Informatics and Computing (JISICOM). https://doi.org/10.52362/jisicom.v6i2.946

OpenCV. (2022). *Logo OpenCV*. https://opencv.org/

Pambudi, K., Winarno, T., & Adhisuwignjo, S. (2021). Motion Planning Defense Robot Dalam Mengikuti Throw Robot Menggunakan Kontrol Kinematik. *Jurnal Elektronika Dan Otomasi Industri*, *8*(3), 198.

https://doi.org/10.33795/elk.v8i3.259
Python. (2022). *Logo Python*. https://www.python.org/
Pytorch. (2023). *Pytorch Logo*. https://pytorch.org/
Rahman, S., Sembiring, A., Siregar, D., Khair, H., Prahmana, G., Puspadini, R., & Zen, M. (2023). Python : Dasar Dan Pemrograman Berorientasi Objek Tahta Media Group. In *Python: Dasar Dan Pemrograman Berorientasi Objek*.
Ramadhan, A. R., Kelistrikan, Perkapalan, P., & Surabaya, N. (2025). *ESTIMASI JARAK NYATA ROBOT SEPAK BOLA BERODA MENGGUNAKAN KAMERA Stereo Vision DENGAN METODE*.
Riansyah, M. I. R., Ardiansyah Al Farouq, & Putu Duta Hasta Putra. (2023). Design and Implementation of Sensor Systems for Localization of the Autonomous Robot in a Building Area. *Jurnal Nasional Teknik Elektro*, *2*. https://doi.org/10.25077/jnte.v12n2.1093.2023
Rijalusalam, D. U., & Iswanto, I. (2021). Implementation kinematics modeling and odometry of four omni wheel mobile robot on the trajectory planning and motion control based microcontroller. *Journal of Robotics and Control (JRC)*, *2*(5), 448–455. https://doi.org/10.18196/jrc.25121
Rohman, A. A. N., Hidayat, R., & Ramadhan, F. R. (2021). Pemrograman Mesin Smart Bartender Menggunakan Saftware Arduini IDE Berbasis Microcontroller ATmega2560. *Prosiding Seminar Nasional Teknik Elektro*, *6*, 14–21.
Romadloni, F., Endrasmono, J., Putra, Z. M. A., Khumaidi, A., Rachman, I., & Adhitya, R. Y. (2023). Identifikasi Warna Buoy Menggunakan Metode You Only Look Once Pada Unmanned Surface Vehicle. *Jurnal Teknik Elektro Dan Komputer TRIAC*, *10*(1). https://doi.org/10.21107/triac.v10i1.19650
ROS. (2023). *Robot Operating System*. https://www.ros.org/
Rozemberczki, B., Scherer, P., He, Y., Panagopoulos, G., Riedel, A., Astefanoaei, M., Kiss, O., Beres, F., López, G., Collignon, N., & Sarkar, R. (2021). PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *International Conference on Information and Knowledge Management, Proceedings* (pp. 4564–4573). https://doi.org/10.1145/3459637.3482014
Safitri, L. I., Sahertian, J., & Widodo, D. W. (2023). Implementasi Algoritma Path Planning A* Pada Base Station Robot Sepak Bola Beroda. *Generation Journal*, *7*(3), 56–63.
Salam, A. (2023). RANCANG BANGUN SISTEM NAVIGASI ROBOT OTONOM WAITER-BOT BERBASIS ROBOT OPERATING SYSTEM. *Bussiness Law Binus*, *7*(2), 33–48. http://repository.radenintan.ac.id/11375/1/PERPUS PUSAT.pdf%0Ahttp://business-law.binus.ac.id/2015/10/08/pariwisata-syariah/%0Ahttps://www.ptonline.com/articles/how-to-get-better-mfi-results%0Ahttps://journal.uir.ac.id/index.php/kiat/article/view/8839
Salsabilla, F. A., Siradjuddin, I., & Winarno, T. (2024). Position Based Visual Servoing untuk Robot Sepak Bola Beroda Menggunakan Kamera Omni. *Jurnal Elektronika Dan Otomasi Industri*, *11*(1), 14–23. https://doi.org/10.33795/elkolind.v11i1.3467
Sucipto, A. D. I., Dewanto, R. S., & Pramadihanto, D. (2021). *Gerak Robot Berkaki*

Dua menggunakan ROS dan RViz sebagai Visualisasi Interaktif. *9*(1), 43–57.
Sutrisno, I., Harlinanda, D. A., Ahmad, Z. M., Priyonggo, P., Mantau, A. J., Santosa, A. W. B., Khalil, M., & Umasangadji, F. (2024). Four Wheel Omni Trajectory with Gyrodometry Method in the Indonesia 2022 Ash Robot Contest. *Indonesian Journal of Innovation Multidisipliner Research*, *2*(2), 206–217. https://doi.org/10.69693/ijim.v2i2.130
Ulfi, M., & Savira Putri Ayu, T. (2024). Implementasi Deep Learning dengan Convolutional Neural Network untuk Tingkat Akurasi Citra Image Hama Sawi Hijau Menggunakan Google Colab. *Riau Journal of Computer Science*, *10*(2), 116–125.
Wahid Hasyim, Siradjuddin, I., & Kamajaya, L. (2025). Robot Beroda Pemindah Barang otomatis Dengan Metode Path Planning. *Jurnal Elektronika Dan Otomasi Industri*, *12*(2), 244–253. https://doi.org/10.33795/elkolind.v12i2.4014
WiMotion. (2024). *HWT101CT*. https://witmotion-sensor.com/
Yaqin, M. A. (2025). *PENERAPAN PATH PLANNING A * UNTUK STRATEGI ROBOT SEPAK BOLA BERODA MENGGUNAKAN ROBOT OPERATING SYSTEM (ROS)*.
Yuniawan, A., Rois, M., Sulistijono, I. A., Barakbah, A. R., & Arief, Z. (2021). Sistem Navigasi dari Holonomic Mobile Robot untuk Membantu Tenaga Kesehatan dalam Pengiriman Logistik kepada Pasien. *INOVTEK Polbeng - Seri Informatika*, *6*(2), 170. https://doi.org/10.35314/isi.v6i2.1989
Zhu, X., Li, W., Huang, K., Cao, S., Lin, B., Li, R., & Xu, W. (2025). Research on the Design and Application of Multi-Port Energy Routers. *Energies*, *18*(4), 96–101. https://doi.org/10.3390/en18040866

LAMPIRAN

Lampiran 1. Biodata Mahasiswa

BIODATA MAHASISWA

1. Nama : Muhammad Jardin Saputra
2. NRP : 0922040086
3. Program Studi : D4 Teknik Otomasi
4. Agama : Islam
5. Status : Belum Menikah
6. Alamat Asal : Perumahan Cerme Indah Blok E17 RT01 RW 03, Kec. Cerme, Kab. Gresik.
7. Nomor Telepon : 081231350079
8. Jenis Kelamin : Laki - Laki
9. Email : muhammadjardin@student.ppns.ac.id
10. Tempat, Tanggal Lahir : Gresik, 20 Februari 2003
11. Nama Orang Tua/Wali : Sunarto
12. Alamat Orang Tua.wali : Gresik
13. Telepon Orang Tua/Wali : 0821-4037-0745
14. Riwayat Pendidikan :

Tabel Lampiran 1. Biodata Mahasiswa

PENDIDIKAN FORMAL			
Pendidikan	**Tabun**	**Tempat Pendidikan**	**Jurusan**
Diploma 4	2022 - 2026	Politeknik Perkapalan Negeri Surabaya	Teknik Otomasi
SMK	2019 – 2022	SMK Negeri 1 Cerme	Teknik Komputer dan Jaringan
SMP	2016 - 2019	SMP Negeri 1 Cerme	-
SD	2010 - 2016	SD Negeri Cerme Lor	-

Lampiran 2. Jurnal

Jurnal Elkolind Volume 13, Nomor 1, Mei 2026
DOI: http://dx.doi.org/10.33795/elkolind.v13i1.9813

Implementasi Estimasi Jarak Relatif Objek Berbasis ROS2 pada Robot Sepak Bola Menggunakan YOLO dan Kamera Omnidirectional 360°

Muhammad Jardin Saputra[1*)], Agus Khumaidi[2], Riko Satrya Fajar Jaelani Putra[3], Ryan Yudha Adhitya[4], Mat Syai'in[5], Evi Nafiatus Sholikhah[6]

[1,2,3,4,5]Jurusan Teknik Kelistrikan Kapal, Program Studi Teknik Otomasi, Politeknik Perkapalan Negeri Surabaya, Indonesia

[6]Jurusan Teknik Kelistrikan Kapal, Program Studi Teknik Kelistrikan Kapal, Politeknik Perkapalan Negeri Surabaya, Indonesia

Penulis Korespondensi, e-mail: muhammadjardin@student.ppns.ac.id

Received: 06/04/2026 *Revised: 16/05/2026* *Accepted:16/05/2026*

ABSTRAK

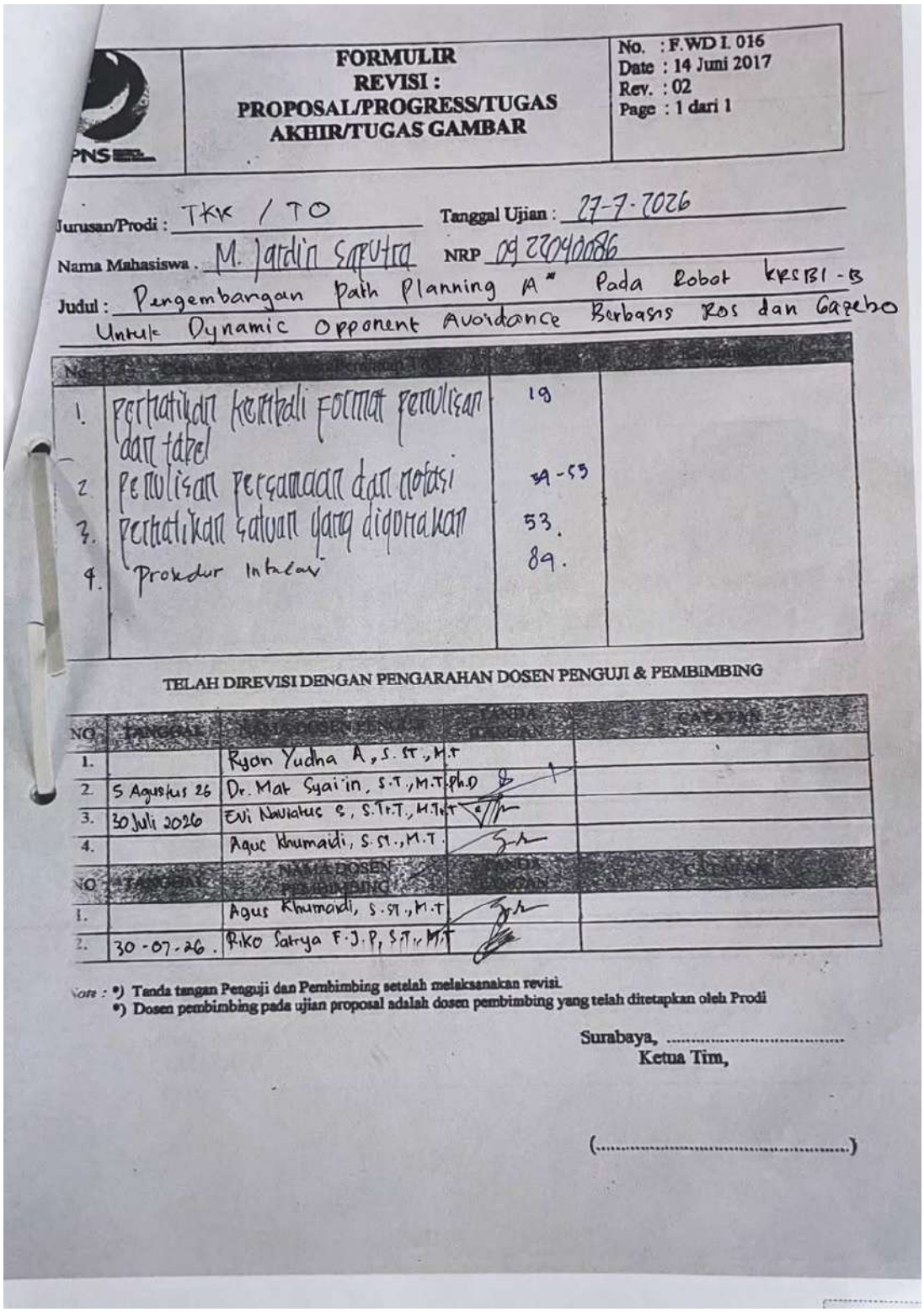
Penelitian ini bertujuan untuk mengimplementasikan sistem estimasi posisi relatif objek pada robot sepak bola menggunakan kamera omnidirectional 360° berbasis ROS 2 dengan metode deteksi objek YOLOv5s. Sistem dirancang untuk mendeteksi objek berupa bola dan robot secara real-time serta mengestimasi posisi objek dalam bentuk koordinat (X,Y) dalam satuan milimeter berdasarkan titik pusat kamera dan proses konversi koordinat piksel menggunakan faktor kalibrasi. Tahap preprocessing citra dilakukan menggunakan metode fisheye correction untuk mengurangi distorsi akibat lensa omnidirectional. Proses komunikasi data antar-node diimplementasikan menggunakan mekanisme publish-subscribe pada ROS 2 Humble. Dataset pelatihan terdiri dari 5.244 citra objek bola dan robot yang dilatih menggunakan framework PyTorch dengan model YOLOv5s. Hasil pengujian menunjukkan bahwa sistem mampu mengestimasi posisi objek dengan tingkat akurasi yang baik, dengan rata-rata error sebesar 2,83% untuk objek bola dan 8,25% untuk objek robot. Selain itu, peningkatan jarak objek menyebabkan kenaikan error, sedangkan peningkatan intensitas pencahayaan meningkatkan nilai confidence deteksi. Pengujian komunikasi ROS 2 menunjukkan bahwa sistem mampu bekerja secara real-time dengan frame rate sebesar 24–30 frame/s, latency komunikasi sebesar 7–19 ms, serta packet loss pada rentang 0–10%. Dengan demikian, sistem yang dikembangkan mampu mendukung proses navigasi dan persepsi lingkungan robot sepak bola secara efektif dan real-time.

Kata Kunci: Jarak Relatif Objek, Omnidirectional 360°, YOLOv5, ROS 2, Computer Vision

ABSTRACT

This study aims to implement a relative object position estimation system for a soccer robot using a 360° omnidirectional camera based on ROS 2 and the YOLOv5s object detection method. The system is designed to detect objects such as balls and robots in real time and estimate object positions in the form of (X,Y) coordinates in millimeters based on the camera center point and pixel coordinate conversion using a calibration factor. Image preprocessing was

Lampiran 3. Form Revisi Sidang Akhir



FORMULIR REVISI : PROPOSAL/PROGRESS/TUGAS AKHIR/TUGAS GAMBAR

No. : F.WD I. 016
Date : 14 Juni 2017
Rev. : 02
Page : 1 dari 1

Jurusan/Prodi : TKK / TO Tanggal Ujian : 27-7-2026

Nama Mahasiswa : M. Jardin Saputra NRP 09 22040086

Judul : Pengembangan Path Planning A* Pada Robot KRSRI-B Untuk Dynamic Opponent Avoidance Berbasis ROS dan Gazebo

No.			
1.	Perhatikan kembali format penulisan dan tabel	19	
2.	Penulisan persamaan dan notasi	49-53	
3.	Perhatikan satuan yang digunakan	53.	
4.	Prosedur Intersect	89.	

TELAH DIREVISI DENGAN PENGARAHAN DOSEN PENGUJI & PEMBIMBING

NO.	TANGGAL	NAMA DOSEN PENGUJI	TANDA TANGAN	CATATAN
1.		Ryan Yudha A, S.ST., M.T		
2.	5 Agustus 26	Dr. Mat Syai'in, S.T., M.T., Ph.D		
3.	30 Juli 2026	Evi Nawatus S, S.T.T., M.T.		
4.		Agus Khumaidi, S.ST., M.T.		
NO.	TANGGAL	NAMA DOSEN PEMBIMBING	TANDA TANGAN	CATATAN
1.		Agus Khumaidi, S.ST., M.T		
2.	30-07-26.	Riko Satrya F.J.P, S.T., M.T		

Note : *) Tanda tangan Penguji dan Pembimbing setelah melaksanakan revisi.
*) Dosen pembimbing pada ujian proposal adalah dosen pembimbing yang telah ditetapkan oleh Prodi

Surabaya, ...
Ketua Tim,

(.....)