Implementasi *Robot Operating System* Pada *Autonomous Surface Vehicle* untuk Lokalisasi Lintasan dengan Metode *You Only Look Once*

Noorman Rinanto [1*], Anugerah Ekha Gusti Audryadmaja [2], Zindhu Maulana Ahmad Putra [3], Agus Khumaidi [4], Ryan Yudha Adhitya [5], Mat Syaiin [6], Isa Rachman [7]

[1, 2, 4, 5, 6, 7] Program Studi Teknik Otomasi, Politeknik Perkapalan Negeri Surabaya

[3] Program Studi Teknik Kelistrikan Kapal, Politeknik Perkapalan Negeri Surabaya

E-mail: [1*] noorman.rinanto@ppns.ac.id

Abstrak— ***Perkembangan teknologi robotika, terutama dalam bidang kendaraan otonom, telah mengalami kemajuan pesat dalam beberapa tahun terakhir. Pada penelitian ini berfokus terhadap pengembangan sistem deteksi dan pelokalan lintasan pada Autonomous Surface Vehicle (ASV) menggunakan Robot Operating System (ROS) dan algoritma You Only Look Once versi kelima (YOLOv5). ASV merupakan kendaraan permukaan otonom yang digunakan untuk berbagai aplikasi, seperti pemetaan bawah laut dan pemantauan lingkungan. Dalam penelitian ini, ROS diimplementasikan sebagai platform integrasi perangkat keras dan lunak untuk meningkatkan akurasi deteksi dan lokalisasi objek, khususnya buoy merah dan hijau sebagai pembatas lintasan. Pengujian dilakukan dalam lingkungan nyata untuk menilai performa sistem, yang sebelumnya hanya berbasis simulasi. Hasil penelitian menunjukkan bahwa integrasi ROS dan YOLOv5 meningkatkan kecepatan navigasi ASV, dengan peningkatan rata-rata waktu tempuh dari 1 menit 16,2 detik menjadi 1 menit 11,2 detik, serta keberhasilan deteksi objek mencapai 70% dari 50 percobaan. Penelitian ini berkontribusi pada pengembangan teknologi ASV dengan peningkatan akurasi, efisiensi, dan keandalan sistem dalam mendeteksi dan melokalisasi objek di area lintasan yang kompleks.***

Kata kunci – Autonomous Surface Vehicle (ASV); Robot Operating System (ROS); YOLOv5; Lokalisasi Objek

I. PENDAHULUAN

Perkembangan teknologi robotika, terutama dalam bidang kendaraan otonom, telah mengalami kemajuan pesat dalam beberapa tahun terakhir [1]. Salah satu implementasi dari teknologi ini adalah *Autonomous Surface Vehicle* (ASV), sebuah kendaraan permukaan otonom yang dirancang untuk bergerak dan beroperasi secara mandiri di atas air [2]. ASV banyak digunakan dalam berbagai aplikasi, mulai dari pemetaan bawah laut hingga pemantauan lingkungan. Dalam konteks ini, kemampuan ASV untuk mendeteksi dan melokalisasi objek di sekitarnya merupakan aspek krusial yang mempengaruhi performa dan keamanan operasionalnya [3],[4]. Salah satu tantangan utama adalah deteksi objek seperti pelampung (*buoy*) merah dan hijau yang sering digunakan sebagai pembatas area lintasan [5],[6].

Dalam penelitian ini, *Robot Operating System* (ROS) diimplementasikan pada ASV sebagai platform untuk mengintegrasikan berbagai komponen perangkat keras dan lunak guna memfasilitasi proses deteksi dan lokalisasi objek [7]. Salah satu metode deteksi objek yang digunakan adalah "*You Only Look Once Version 5*" (YOLOv5), sebuah algoritma yang dikenal karena kemampuannya melakukan deteksi objek secara *real-time* dengan akurasi yang tinggi [8]. Metode YOLOv5 memungkinkan ASV untuk mendeteksi buoy merah dan hijau secara efisien, sehingga dapat membantu kendaraan dalam menavigasi lintasan uji dengan lebih baik [9].

Penelitian ini berfokus pada pengembangan sistem deteksi dan pelokalan lintasan menggunakan ROS dan metode YOLO, dengan tujuan utama mengevaluasi tingkat keberhasilan sistem dalam mendeteksi buoy sebagai pembatas lintasan[10],[11]. Dalam sistem sebelumnya, ASV hanya mengandalkan deteksi tanpa menggunakan lokalisasi area dan ROS, yang menyebabkan keterbatasan dalam hal akurasi dan efisiensi sistem [12]. Oleh karena itu, penelitian ini bertujuan untuk meningkatkan kinerja sistem dengan mengintegrasikan ROS sebagai platform utama dan menambahkan fitur lokalisasi.

Pada penelitian sebelumnya, penulis melakukan simulasi sistem deteksi menggunakan simulator ROS Gazebo, dengan implementasi sistem lokalisasi area [3]. Dalam studi terbaru ini, ASV akan diuji pada lingkungan nyata untuk menilai sejauh mana integrasi ROS dan YOLOv5 dapat meningkatkan performa deteksi dan pelokalan objek pada lintasan uji yang sebenarnya [13] [14].

Melalui penelitian ini, diharapkan dapat diperoleh pemahaman yang lebih baik mengenai implementasi ROS dan metode YOLO dalam sistem ASV untuk aplikasi deteksi dan lokalisasi objek di area lintasan uji. Selain itu, penelitian ini juga diharapkan dapat memberikan kontribusi terhadap pengembangan lebih lanjut dari teknologi ASV, terutama dalam hal peningkatan akurasi, efisiensi, dan keandalan sistem dalam mendeteksi serta melokalisasi objek di lingkungan yang kompleks.

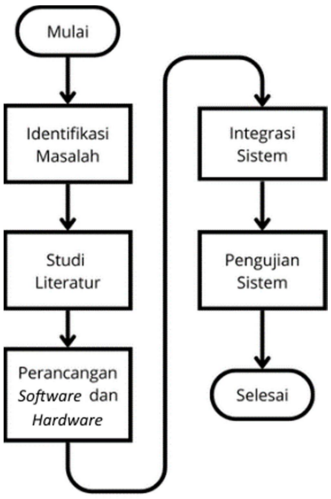
II. BAHAN DAN METODE

Penelitian mengenai *Autonomous Surface Vehicle* (ASV) kini sangat populer di kalangan peneliti. Oleh karena itu penulis ikut ambil bagian dalam menemukan penelitian ASV dan dapat berkontribusi terhadap perkembangan teknologi ASV. Pada bab ini dijelaskan secara detil mengenai alur penelitian, perancangan *hardware* ASV,

integrasi *software* dan *hardware* dan skema untuk pengujian sistem.

A. Alur Penelitian

Penelitian ini merupakan lanjutan dari penelitian sebelumnya yang telah dilakukan oleh penulis. Pada penelitian sebelumnya penulis berhasil melakukan simulasi kinerja sistem dengan baik menggunakan *software* Gazebo Simulator. Oleh karena itu, penelitian ini berfokus pada pengintegrasian sistem dengan *hardware* dan pengujian di lingkungan yang sebenarnya. Berikut merupakan alur penelitian pada penelitian ini disajikan pada Gambar 1.



Gambar 1. Diagam Alur Penelitian

Berdasarkan diagram alur penelitian tersebut, langkah pertama adalah melakukan perumusan masalah dan di ikuti studi literatur sebagai acuan saat melakukan penelitian. Langkah selanjutnya yaitu melakukan perancangan *hardware prototype* yang diimplementasikan. Kemudian, dilanjutkan proses integrasi sistem, dan terakhir merupakan pengujian sistem terhadap kondisi yang sebenarnya.

B. Perancangan Hardware ASV

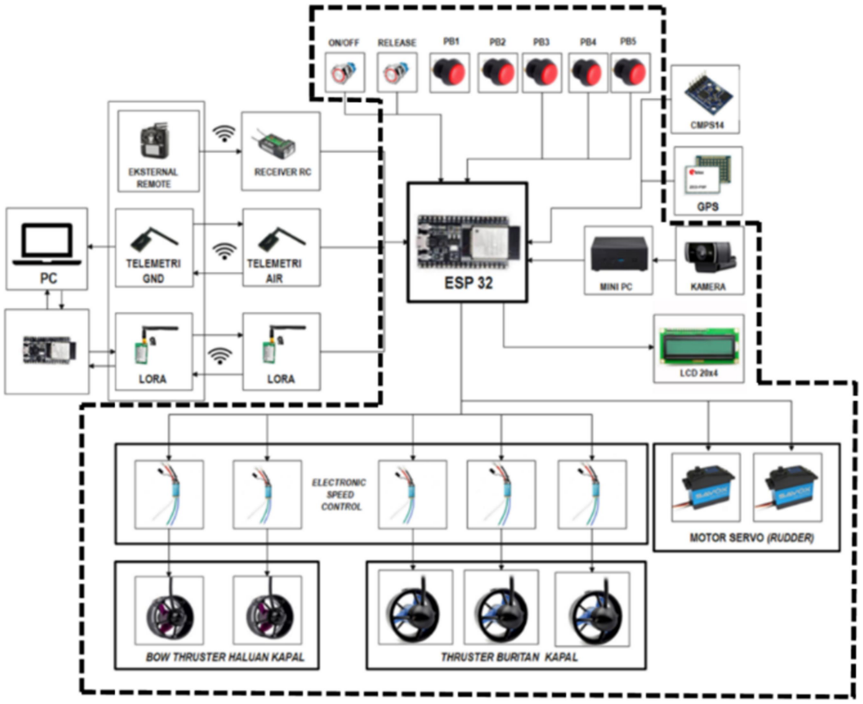
Dalam sebuah sistem proses perancangan *hardware* merupakan proses yang cukup riskan. Pada proses ini dilakukan penyesuaian spesifikasi *hardware* yang digunakan sesuai dengan kebutuhan. Berikut ini merupakan diagram perancangan *hardware* pada ASV seperti ditunjukkan pada Gambar 2.

Gambar tersebut menunjukkan rancangan rangkaian perangkat keras yang diterapkan pada penelitian ini. Tidak semua sistem dibahas secara detail, hanya komponen-komponen yang berada dalam area garis putus-putus yang akan menjadi fokus pembahasan. Setiap komponen memiliki fungsi yang berbeda, seperti mini-PC yang bertugas mengolah citra dan mengambil keputusan untuk prototipe, ESP32 yang berperan sebagai eksekutor tindakan dengan memberikan data masukan pada *Electronic Speed Controller* (ESC) dan motor servo sesuai perintah dari mini-PC melalui komunikasi serial. Kamera berfungsi sebagai sensor untuk menangkap citra yang diproses, ESC mengontrol kecepatan motor *thruster*, motor servo menggerakkan propulsi prototipe, motor *thruster* sebagai pendorong prototipe, serta LCD 20x4 yang menampilkan data kecepatan.

C. Integrasi Software dan Hardware

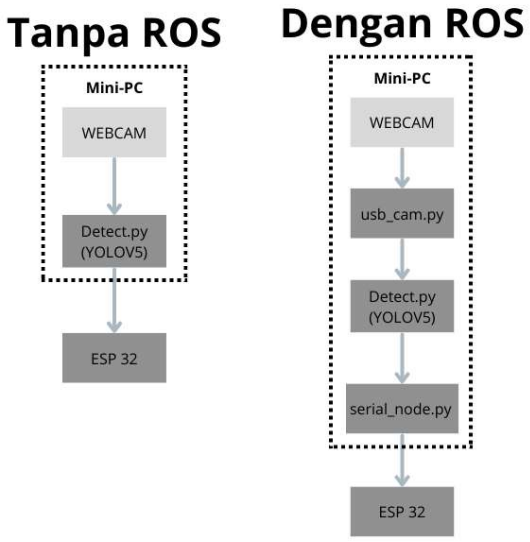
Integrasi merupakan proses yang sangat krusial, proses ini merupakan penentu apakah *hardware* dan *software* dapat berjalan sesuai dengan kebutuhan yang diinginkan. Pada penelitian ini *software* yang dipakai merupakan hasil dari penelitian sebelumnya. Oleh karena itu pada penelitian ini hanya dilakukan proses integrasi *software* dan *hardware*.

Software yang dipakai merupakan sebuah program



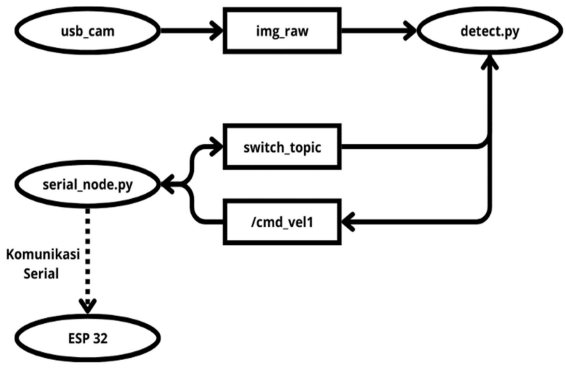
Gambar 2. Rancangan *hardware* keseluruhan sistem ASV.

deteksi objek yang berjalan pada ekosistem ROS. Berikut adalah diagram perbandingan diagram *software* sistem jika menggunakan ROS dan tanpa ROS disajikan pada Gambar 3.



Gambar 3. Perbedaan Diagram Blok Sistem Menggunakan ROS dan Tanpa ROS

Kedua sistem tersebut memiliki perbedaan yang signifikan. Sistem konvensional lebih sederhana dan ringkas dibandingkan dengan yang menggunakan ekosistem ROS. Pada sistem konvensional, data dari kamera langsung diproses oleh program untuk deteksi dan dikirim ke mikrokontroler tanpa perantara. Sementara di ROS, setiap proses memiliki peran khusus, mulai dari pengambilan data kamera oleh "usb_cam.py", pemrosesan dengan YOLO pada "Detect.py", hingga penerjemahan data serial oleh "serial_node.py". Meskipun sistem konvensional tampak sederhana, namun sangat rentan—jika terjadi kesalahan pada satu titik, seluruh sistem akan berhenti tanpa peringatan. Sebaliknya, di ROS, hanya bagian yang bermasalah yang berhenti, dan ada notifikasi kesalahan. Menambah fungsi pada sistem konvensional sulit karena jalur data bisa bertabrakan, sedangkan di ROS, cukup menggunakan topik yang sama tanpa risiko benturan data. Dalam penelitian ini, ROS dijalankan di mini-PC untuk pemrosesan data kamera, sedangkan mikrokontroler berfungsi sebagai pengendali sistem gerak. Diagram koneksi sistem di lingkungan ROS dapat dilihat pada Gambar 4.



Gambar 4. Diagram blok hasil integrasi *software.*

Berdasarkan Gambar 4, terdapat tiga *node* yang dihubungkan oleh tiga topik sebagai jalur pengiriman data. Proses dimulai dari node "usb_cam" yang mengambil citra digital dari webcam dan mengirimkannya melalui topik "img_raw" dalam format “image_smg” menuju node "detect.py". Node "detect.py" berfungsi memproses data dengan metode YOLOv5, kemudian data tersebut dikirim ke ESP32 melalui topik "cmd_vel1" dalam format data "twist", yang terhubung dengan node "serial_node.py". Format "twist" pada topik "cmd_vel1" memiliki 6 slot data float, namun hanya 3 slot yang digunakan untuk mengirim data sudut rudder, status deteksi buoy, dan perintah untuk pemindaian. Terakhir, node "serial_node.py" bertugas mengonfigurasi port serial dan menghubungkan komunikasi antara mini PC dengan ESP32 melalui “rosserial communication”.

D. Skema Pengujian Sistem

Pengujian sistem bertujuan untuk mengetahui bagaimana kemampuan dan kehandalan dari *prototype* yang telah dibuat. Terdapat tiga buah pengujian pada studi ini. Pengujian pertama dilakukan dengan memberi objek deteksi pada sistem guna mengetahui bagaimana respons sistem apakah benar atau tidak. Lalu pengujian kedua dilakukan dengan dua sistem yang berbeda guna mengetahui bagaimana perbedaan respons sistem pada sistem yang lama dan yang baru. Pada sistem yang lama belum di lengkapi dengan kemampuan lokalisasi area dan ROS namun untuk sistem terbaru telah menggunakan metode lokalisasi area dan ROS. Selanjutnya, yang terakhir merupakan pengujian tingkat keberhasilan sistem. Pengujian ini dilakukan sebanyak 50 kali percobaan dengan kondisi pencahayaan yang berbeda beda sehingga diharapkan dapat diketahui kondisi yang paling ideal agar ASV dapat berjalan dengan baik.

III. HASIL DAN PEMBAHASAN

Bab ini menjelaskan seluruh hasil pengujian kinerja terhadap sistem yang telah dibuat. Karena pengujian pada metode YOLOv5 telah dilakukan oleh peneliti pada penelitian sebelumnya, maka pada penelitian ini pengujian sistem dititik beratkan pada kemampuan *real* dari robot dan respons sistem ASV.

A. Hasil Pengujian Awal Hardware dan Software

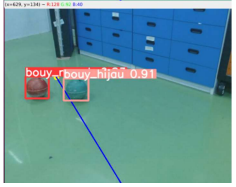
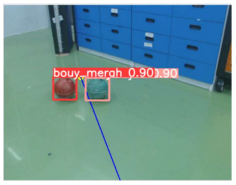
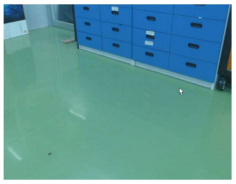
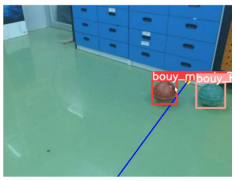
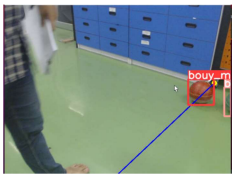
Pengujian awal *software* dan *hardware* bertujuan untuk mengetahui apakah sistem telah terintegrasi secara menyeluruh dan dapat berfungsi dengan baik atau tidak sebelum dilakukan pengujian secara langsung untuk melalui seluruh lintasan.

Untuk mengintegrasikan seluruh sistem digunakan sebuah komunikasi serial untuk mengirim data sudut motor *servo*, deteksi *buoy*, dan perintah untuk *scaning* lintasan pengujian dari mini-PC ke ESP 32 dan data kondisi rilis kapal dari ESP 32 ke mini-PC menggunakan protokol *rosserial*. Pada protokol *rosserial* terdapat dua mode yang dapat digunakan yaitu mode *publiszer* dan *subcriber* yang dikirim melalui sebuah topik. Mode *publiszer* digunakan untuk mengirim data melalui sebuah topik sedangkan untuk mode *subcribe* digunakan untuk membaca data yang telah

dipublikasi melalui topik tertentu. Untuk mengirim data pada penelitian ini digunakan 3 topik yaitu topik "/cmd_vel1" dengan tipe data "*twis"* yang mampu menampung sebanyak 6 data digunakan untuk mengirim data dari mini-PC ke ESP32, sedangkan untuk data yang dikirim dari ESP 32 ke mini-PC menggunakan topik "switch_topic" dengan tipe data bool serta topik "img_raw" digunakan untuk mentransfer data gambar dari node usb_cam menuju node detect.py.

Untuk menentukan pergerakan kapal digunakan titik referensi pergerakan kapal yang di ambil dari titik tengah objek yang terdekat dari hasil pemetaan area lintasan jika *buoy* terdeteksi lebih dari satu kelas. Berikut adalah hasil pengujian respons pergerakan *rudder* kapal berdasarkan data kamera yang mendeteksi *buoy* warna merah dan biru seperti ditunjukkan pada Tabel I.

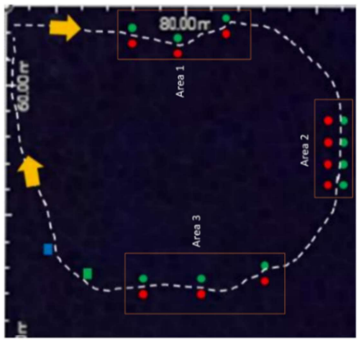
TABEL I. HASIL PENGUJIAN PERGERAKAN *RUDDER* BERDASARKAN MASUKKAN DETEKSI *BUOY*.

No.	Deteksi	Data Yang Dikirim (º)	Pergerakan Motor Servo (º)	Akurasi (%)
1.		120	120	100
2.		110	110	100
3.		105	104	99.04
4.		100	100	100
5.		95	93	97.89
6.		90	90	100
7.		85	85	100
8.		80	80	100
9.		75	76	98.68
10		70	70	100
	Rata-rata Eror			**99.56**

Dari pengujian pergerakan motor *servo* diperoleh hasil bahwa pergerakan motor *servo* memiliki rata-rata akurasi pergerakan sebesar 99,56% dari perintah yang di inginkan. Dengan hasil pengujian eror di bawah 5% maka sistem dapat dikatakan layak untuk dilakukan pengujian selanjutnya yaitu pengujian secara otomatis pada lintasan pengujian.

B. Hasil Pengujian ASV(Autonomous Surface Vessel) Secara Otomatis pada Lintasan Pengujian

Setelah dilakukan integrasi *hardware* dan *software* dengan hasil pengujian sinkronisasi yang baik dan dapat berjalan pada simulator gazebo, maka tahapan selanjutnya adalah pengujian kapal ASV secara otomatis dalam melalui lintasan pengujian. Pengujian ini bertujuan untuk mengetahui apakah kapal dapat menentukan lokasi lintasan serta melalui lintasan dengan baik tanpa menabrak dan menyentuh *buoy* pembatas lintasan. Ilustrasi lintasan pengujian tersedia pada Gambar 5.



Gambar 5. Ilustrasi Lintasan Pengujian

Pengujian dilakukan dengan dua sesi dengan setiap sesi dilakukan pengujian sebanyak 5 kali. Sesi pertama dilakukan dengan menggunakan program sebelum diaplikasikan kemampuan pemetaan area lintasan dan tanpa penggunaan ROS. Program yang digunakan pada pengujian ini berjalan pada operasi sistem Windows 10. Dari pengujian sesi pertama diperoleh hasil yang disajikan pada Tabel II.

TABEL II. HASIL PENGUJIAN SECARA OTOMATIS TANPA ROS DAN PEMETAAN

Pengujian ke-	Waktu Tempuh	Keterangan
1.	1 menit 18 detik	Berhasil
2.	-	Gagal (Lintasan 3)
3.	1 menit 16 detik	Berhasil
4.	-	Gagal (Lintasan 2)
5.	1 menit 17 detik	Berhasil
6.	-	Gagal (Lintasan 1)
7.	1 menit 15 detik	Berhasil
8.	1 menit 18 detik	Berhasil
9.	1 menit 13 detik	Berhasil
10.	-	Gagal (Lintasan 3)
Rata-rata Waktu Tempuh	**1 menit 16,2 detik**	

Dari data pengujian yang tersaji pada Tabel II dapat diketahui bahwa kapal memiliki rata-rata waktu tempuh dalam melalui lintasan pengujian selama 1 menit 16,2 detik dengan keberhasilan 6 (Berhasil) banding 4 (Gagal).

Pengujian sesi kedua dilakukan dengan menggunakan program yang berjalan pada ekosistem ROS dan dibekali kemampuan pemetaan area lintasan. Program ini berjalan pada operasi sistem Linux Ubuntu 20.04. Hasil pengujian sesi kedua disajikan pada Tabel III.

TABEL III. HASIL PENGUJIAN SECARA OTOMATIS TANPA ROS DAN PEMETAAN

Pengujian ke-	Waktu Tempuh	Keterangan
1.	-	Gagal (Lintasan 2)
2.	1 menit 9 detik	Berhasil
3.	-	Gagal (Lintasan 2)
4.	1 menit 9 detik	Berhasil
5.	1 menit 6 detik	Berhasil
6.	-	Gagal (Lintasan 2)
7.	1 menit 12 detik	Berhasil
8.	1 menit 12 detik	Berhasil
9.	1 menit 12 detik	Berhasil
10.	1 menit 19 detik	Berhasil
Rata-rata Waktu Tempuh	**1 menit 11,2 detik**	

Dari data pengujian sesi kedua yang tersaji pada Tabel III dapat diketahui bahwa kapal memiliki rata-rata waktu tempuh dalam melalui lintasan pengujian selama 1 menit 11,2 detik dengan keberhasilan 7 (Berhasil) banding 3 (Gagal) saat menggunakan program pemetaan area lintasan.

Setelah kedua sesi pengujian dilakukan dapat diketahui bahwa dengan penambahan kemampuan pemetaan area lintasan dapat meningkatkan performa kapal hingga 5 detik. Selain peningkatan performa penulis juga mengamati bahwa dengan penambahan kemampuan pemetaan area lintasan pergerakan kapal atau manuver kapal menjadi lebih stabil.

C. Hasil Pengujian Tingkat Keberhasilan Kapal Secara Otomatis pada Lintasan Pengujian

Setelah di lakukan pengujian secara otomatis dan kapal dapat meliwati lintasan pengujian dengan baik langkah selanjutnya ialah pengujian konsistensi kapal dalam melalui lintasan pengujian. Pada pengujian tingkat keberhasilan kapal dilakukan sebanyak 5 sesi pengujian dengan 10 kali percobaan pada setiap sesinya. Sesi pertama dilakukan pada pukul 07.56 sampai pukul 08.35. Sesi kedua dilakukan pada pukul 10.02 sampai pukul 10.36. Sesi ketiga dilakukan pada pukul 12.20 sampai pukul 13.05. Sesi keempat dilakukan pada pukul 13.57 sampai pukul 14.20. Sesi kelima dilakukan pada pukul 16.13 sampai pukul 16.46. Berikut adalah seluruh hasil percobaan yang telah dilakukan ditunjukkan pada Tabel IV sampai VIII.

TABEL IV. HASIL PENGUJIAN TINGKAT KEBERHASILAN PADA SESI PERTAMA

Percobaan ke-	Waktu Tempuh	Keterangan
1.	-	Gagal
2.	1 menit 4 detik	Berhasil
3.	-	Gagal
4.	1 menit 4 detik	Berhasil
5.	1 menit 5 detik	Berhasil
6.	1 menit 7 detik	Berhasil
7.	-	Gagal
8.	-	Gagal
9.	1 menit 3 detik	Berhasil
10.	1 menit 6 detik	Berhasil
Rata-rata Waktu Tempuh	**1 menit 4,8 detik**	**Keberhasilan = 60 %**

TABEL V. HASIL PENGUJIAN TINGKAT KEBERHASILAN PADA SESI KEDUA

Pengujian ke-	Waktu Tempuh	Keterangan
1.	1 menit 5 detik	Berhasil
2.	1 menit 6 detik	Berhasil
3.	1 menit 7 detik	Berhasil
4.	-	Gagal
5.	1 menit 6 detik	Berhasil
6.	-	Gagal
7.	1 menit 5 detik	Berhasil
8.	1 menit 8 detik	Berhasil
9.	-	Gagal
10.	1 menit 7 detik	Berhasil
Rata-rata Waktu Tempuh	**1 menit 6,3 detik**	**Keberhasilan = 70 %**

TABEL VI. HASIL PENGUJIAN TINGKAT KEBERHASILAN PADA SESI KETIGA

Percobaan ke-	Waktu Tempuh	Keterangan
1.	1 menit 5 detik	Berhasil

2.	1 menit 9 detik	Berhasil
3.	1 menit 6 detik	Berhasil
4.	-	Gagal
5.	-	Gagal
6.	1 menit 4 detik	Berhasil
7.	1 menit 6 detik	Berhasil
8.	1 menit 7 detik	Berhasil
9.	-	Gagal
10.	1 menit 9 detik	Berhasil
Rata-rata Waktu Tempuh	**1 menit 6,6 detik**	**Keberhasilan = 70 %**

TABEl VII. HASIL PENGUJIAN TINGKAT KEBERHASILAN PADA SESI KEEMPAT

Percobaan ke-	Waktu Tempuh	Keterangan
1.	1 menit 7 detik	Berhasil
2.	1 menit 9 detik	Berhasil
3.	1 menit 7 detik	Berhasil
4.	1 menit 16 detik	Berhasil
5.	-	Gagal
6.	-	Gagal
7.	1 menit 5 detik	Berhasil
8.	1 menit 12 detik	Berhasil
9.	-	Gagal
10.	1 menit 9 detik	Berhasil
Rata-rata Waktu Tempuh	**1 menit 7 detik**	**Keberhasilan = 70 %**

TABELVIII. HASIL PENGUJIAN TINGKAT KEBERHASILAN PADA SESI KELIMA

Percobaan ke-	Waktu Tempuh	Keterangan
1.	-	Gagal
2.	1 menit 11 detik	Berhasil
3.	1 menit 14 detik	Berhasil
4.	1 menit 14 detik	Berhasil
5.	1 menit 14 detik	Berhasil
6.	1 menit 13 detik	Berhasil
7.	1 menit 17 detik	Berhasil
8.	1 menit 18 detik	Berhasil
9.	1 menit 15 detik	Berhasil
10.	-	Gagal
Rata-rata Waktu Tempuh	**1 menit 14,5 detik**	**Keberhasilan = 80 %**

Berdasarkan pengujian tingkat keberhasilan diperoleh waktu terbaik yang berhasil diraih pada proses pengujian ini terjadi pada sesi pertama dengan waktu 1 menit 4,8 detik sedangkan waktu paling lama terjadi pada sesi kelima dan untuk waktu tempuh rata-rata kapal ialah 1 menit 14,5 detik. Sesuai data hasil pengujian terlihat pada sesi kelima terjadi penurunan performa kapal, hal tersebut terjadi akibat daya baterai mulai melemah, pada sesi kelima justru memiliki tingkat keberhasilan yang paling tingi yaitu 80%. Dari data pengujian kelima sesi pengujian yang disajikan, kapal dapat meliwati lintasan sebanyak 35 kali dan mengalami kegagalan sebanyak 15 kali dari seluruh percobaan yang dilakukan. Dengan demikian dari pengujian dapat disimpulkan bahwa kapal memiliki tingkat keberhasilan yang baik dalam melalui lintasan pengujian dengan keberhasilan mencapai 70% dari seluruh percobaan yang dilakukan.

IV. KESIMPULAN

Berdasarkan hasil seluruh pengujian yang dilakukan terhadap kapal ASV pada penelitian ini secara langsung, maka dapat ditarik kesimpulan pada penelitian ini yaitu Penggunaan ROS dan pemetaan area lintasan pada kapal *Autonomous Surface Vehicle* (ASV) terbukti efektif dalam meningkatkan performa kapal, dengan peningkatan rata-rata waktu tempuh dari 1 menit 16,2 detik menjadi 1 menit 11,2 detik, dan waktu terbaik mencapai 1 menit 6 detik. Hal ini menunjukkan peningkatan performa sekitar 5 detik lebih cepat dari sistem sebelumnya serta perbaikan pada manuver kapal selama pengujian. Kapal berhasil melintasi lintasan pengujian dengan tingkat keberhasilan 70%, yaitu 35 keberhasilan dari 50 kali percobaan dalam lima sesi, dengan waktu tempuh rata-rata 1 menit 8,57 detik. Sesi kelima mencatat keberhasilan tertinggi karena posisi sumber cahaya yang sejajar dengan ASV, sehingga meningkatkan akurasi YOLOv5 dalam mendeteksi objek *buoy* pembatas lintasan.

DAFTAR PUSTAKA

[1] M. J. Hong and M. R. Arshad, "Modeling and motion control of a riverine autonomous surface vehicle (ASV) with differential thrust," *J. Teknol.*, vol. 74, no. 9, pp. 137–143, 2015, doi: 10.11113/jt.v74.4817.

[2] T. Beatriz, "Desain Kendali Sistem Gerak Bow Thruster Pada Autonomous Surface Vehicle Dengan Menggunakan Metode Sliding Mode Control," 2020.

[3] A. Ekha, G. Audryadmaja, J. Endrasmono, Z. Maulana, A. Putra, and L. Subiyanto, "Implementasi ROS dan Optimasi Identifikasi Warna Buoy Dengan Metode YOLOv5 Pada Miniatur Autonomous Surface Vehicle," *Junal Elkolind*, vol. 11, no. 1, pp. 299–308, 2024, [Online]. Available: https://doi.org/10.33795/elkolind.v11i1.5343

[4] K. Azman, M. Arhami, and Azhar, "Metode You Only Look Once (YOLO) dalam Deteksi Physical Distancing dan Wajah Bermasker," *Proceeding Semin. Nas. Politek. Negeri Lhokseumawe*, vol. 6, no. 1, pp. 107–113, 2022, [Online]. Available: https://e-jurnal.pnl.ac.id/semnaspnl/article/view/3446/0

[5] F. Romadloni, J. Endrasmono, Z. M. A. Putra, A. Khumaidi, I. Rachman, and R. Y. Adhitya, "Identifikasi Warna Buoy Menggunakan Metode You Only Look Once Pada Unmanned Surface Vehicle," *J. Tek. Elektro dan Komput. TRIAC*, vol. 10, no. 1, pp. 23–29, 2023, doi: 10.21107/triac.v10i1.19650.

[6] B. Yan, P. Fan, X. Lei, Z. Liu, and F. Yang, "A real-time apple targets detection method for picking robot based on improved YOLOv5," *Remote Sens.*, vol. 13, no. 9, pp. 1–23, 2021, doi: 10.3390/rs13091619.

[7] K. Kisron, B. S. B. Dewantara, and H. Oktavianto, "Improved Performance of Trash Detection and Human Target Detection Systems using Robot Operating System (ROS)," *J. Rekayasa Elektr.*, vol. 17, no. 2, 2021, doi: 10.17529/jre.v17i2.20805.

[8] G. C. Utami, C. R. Widiawati, and P. Subarkah,

"Detection of Indonesian Food to Estimate Nutritional Information Using YOLOv5," *Teknika*, vol. 12, no. 2, pp. 158–165, 2023, doi: 10.34148/teknika.v12i2.636.

[9] A. V. EGA and W. ARDIATNA, "Study on Image Processing Method and Data Augmentation for Chest X-Ray Nodule Detection with YOLOv5 Algorithm," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 11, no. 2, p. 424, 2023, doi: 10.26760/elkomika.v11i2.424.

[10] A. Khumaidi, R. Y. Adhitya, D. Wardani, M. R. Fahmi, S. Utomo, and M. D. Khairansyah, "Design of a Fire Spot Identification System in PT . PAL Indonesia Work Area Using," 2023.

[11] R. Illmawati, "YOLO v5 untuk Deteksi Nomor Kendaraan di DKI Jakarta YOLO V5 for Vehicle Plate Detection in DKI Jakarta," *J. Komput. Abdi Inform.*, vol. 10, pp. 32–43, 2023.

[12] H. Zhang and K. Watanabe, "ROS Based Framework for Autonomous Driving of AGVs," *Conf.E-Jikei.Org*, 2019, [Online]. Available: http://conf.e-jikei.org/ICMEMIS/2019/proceedings/materials/proc_files/IPS/IPS06_Nakazawa/IPS06_4/Camera_ready_Manuscript_ICMEMIS2019_IPS06_A004.pdf

[13] I. Y. Arulampalam Kunaraj, P.Chelvanathan, Ahmad AA Bakar, "PENGEMBANGAN ROBOT MOBIL MENGGUNAKAN ROS Wea," *J. Eng. Res.*, vol. 1, no. 1, pp. 1–14, 2023, [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK558907/

[14] A. Jalil, "Robot Operating System (Ros) Dan Gazebo Sebagai Media Pembelajaran Robot Interaktif," *Ilk. J. Ilm.*, vol. 10, no. 3, pp. 284–289, 2018, doi: 10.33096/ilkom.v10i3.365.284-289.