Implementation of Robot *Operating System* on Autonomous Surface Vehicle for Trajectory Localization with You Only Look Once Method

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] Automation Engineering Study Program of Politeknik Perkapalan Negeri Surabaya

[3] Marine Electrical Engineering Study Program of Politeknik Perkapalan Negeri Surabaya

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

Abstract*—The development of robotics technology, especially in the field of autonomous vehicles, has made rapid progress in recent years. This study focuses on the development of a trajectory detection and localization system on an Autonomous Surface Vehicle (ASV) using the Robot Operating System (ROS) and the You Only Look Once algorithm version five (YOLOv5). ASV is an autonomous surface vehicle used for various applications, such as underwater mapping and environmental monitoring. In this study, ROS is implemented as a hardware and software integration platform to improve the accuracy of object detection and localization, especially the red and green buoys as trajectory boundaries. Testing was carried out in a real environment to assess the performance of the system, which was previously only based on simulation. The results showed that the integration of ROS and YOLOv5 increased the navigation speed of the ASV, with an increase in the average travel time from 1 minute 16.2 seconds to 1 minute 11.2 seconds, and the success of object detection reached 70% out of 50 trials. This study contributes to the development of ASV technology by increasing the accuracy, efficiency, and reliability of the system in detecting and localizing objects in complex trajectory areas.

Kata kunci* – *Autonomous Surface Vehicle (ASV); Robot Operating System (ROS); YOLOv5; Object Localization

I. INTRODUCTION

The development of robotics technology, especially in the field of autonomous vehicles, has experienced rapid progress in recent years [1]. One implementation of this technology is the Autonomous Surface Vehicle (ASV), an autonomous surface vehicle designed to move and operate independently on water [2]. ASVs are widely used in various applications, ranging from underwater mapping to environmental monitoring. In this context, the ability of ASVs to detect and localize objects around them is a crucial aspect that affects their performance and operational safety [3], [4]. One of the main challenges is the detection of objects such as red and green buoys that are often used as boundaries for track areas [5], [6]. In this study, the Robot Operating System (ROS) was implemented on ASV as a platform to integrate various hardware and software components to facilitate the process of object detection and localization [7]. One of the object detection methods used is "You Only Look Once Version 5" (YOLOv5), an algorithm known for its ability to detect objects in real time with high accuracy [8]. The YOLOv5 method enables ASV to detect red and green buoys efficiently, thus assisting vehicles in navigating the test track better [9].

This study focuses on the development of a track detection and localization system using ROS and the YOLO method, with the main objective of evaluating the success rate of the system in detecting buoys as track boundaries [10], [11]. In the previous system, ASV only relied on detection without using area localization and ROS, which caused limitations in terms of system accuracy and efficiency [12]. Therefore, this study aims to improve system performance by integrating ROS as the main platform and adding localization features.

In the previous study, the author conducted a simulation of the detection system using the ROS Gazebo simulator, with the implementation of the area localization system [3]. In this latest study, ASV will be tested in a real environment to assess the extent to which the integration of ROS and YOLOv5 can improve the performance of object detection and localization on the actual test track [13], [14].

Through this study, it is expected to gain a better understanding of the implementation of ROS and the YOLO method in the ASV system for object detection and localization applications in the test track area. In addition, this research is also expected to contribute to the further development of ASV technology, especially in terms of improving the accuracy, efficiency, and reliability of the system in detecting and localizing objects in complex environments.

II. MATERIALS AND METHODS

Research on Autonomous Surface Vehicle (ASV) is now very popular among researchers. Therefore, the author took part in finding ASV research and can contribute to the development of ASV technology. This chapter explains in detail the research flow, ASV hardware design, software and hardware integration and schemes for system testing.

A. Research Flow

This research is a continuation of previous research conducted by the author. In the previous research, the author successfully simulated system performance using Gazebo Simulator software. Therefore, this research focuses on system integration with hardware and testing in a real environment. The following is the research flow in this study presented in Figure 1.

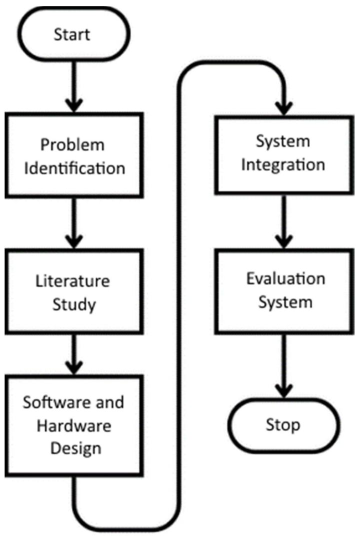


Figure 1. Research Flow Diagram

Based on the research flow diagram, the first step is to formulate the problem and followed by a literature study as a reference when conducting research. The next step is to design the hardware prototype that is implemented. Then, the system integration process is continued, and finally the system is tested against actual conditions.

B. *ASV Hardware Design*

In a system, the hardware design process is a fairly risky process. In this process, adjustments are made to the hardware specifications used according to needs. The following is a hardware design diagram on ASV as shown in Figure 2.

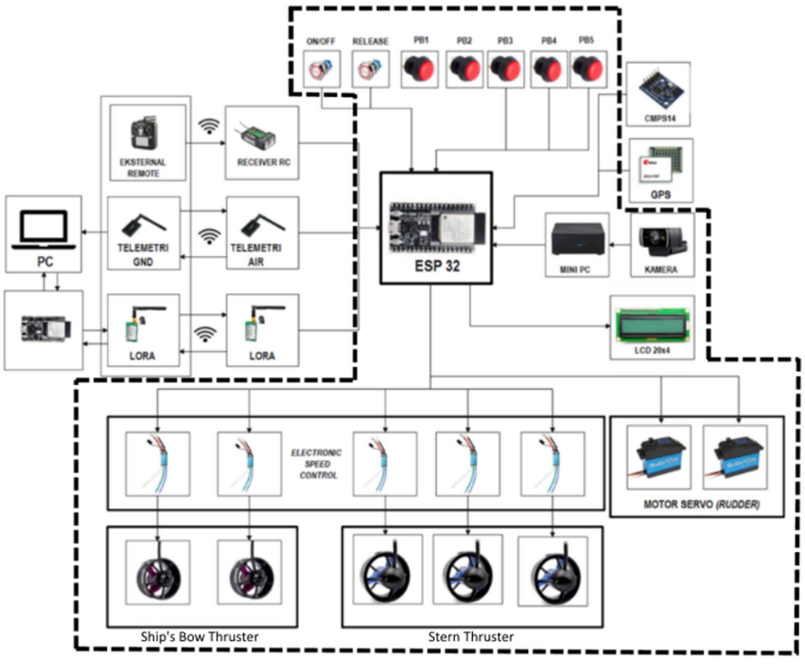
The figure shows the design of the hardware circuit applied in this study. Not all systems are discussed in detail, only the components in the dotted line area will be the focus of the discussion. Each component has a different function, such as a mini-PC that processes images and makes decisions for the prototype, ESP32 which acts as an action executor by providing input data to the Electronic Speed Controller (ESC) and servo motors according to commands from the mini-PC via serial communication. The camera functions as a sensor to capture the processed image, the ESC controls the speed of the thruster motor, the servo motor drives the prototype propulsion, the thruster motor as the prototype driver, and the 20x4 LCD that displays speed data.

C. *Software and Hardware Integration*

Integration is a very crucial process, this process determines whether hardware and software can run according to the desired needs. In this study, the software used is the result of previous research. Therefore, in this study only the software and hardware integration process was carried out.

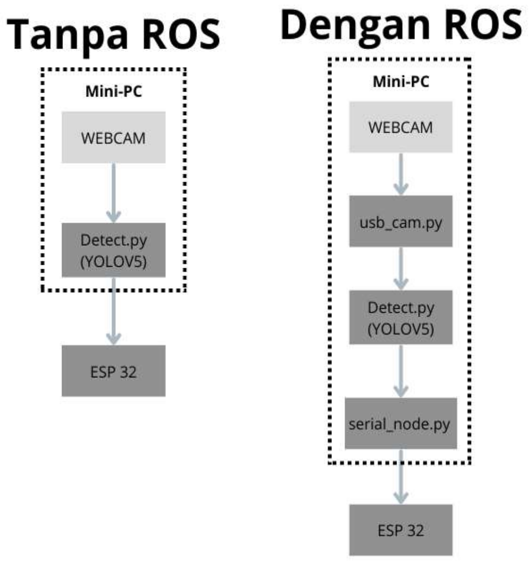
The software used is an object detection program that runs on the ROS ecosystem. The following is a comparison diagram of the system software diagram if using ROS and without ROS presented in Figure 3.

These two systems have striking differences. The conventional system is simpler and more concise compared to the one using the ROS ecosystem. In the conventional system, data from the camera is directly processed by the program for detection and sent to the microcontroller without an intermediary. While in ROS, each process has a specific role, starting from retrieval of camera data by "usb_cam.py", processing with YOLO in "Detect.py", to translation of serial data by "serial_node.py". Although the

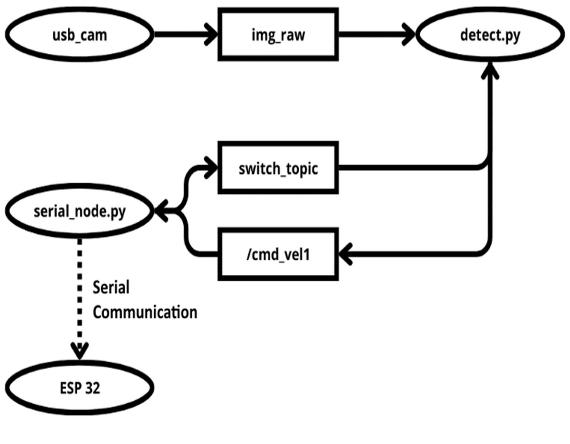


Gambar 2. Overall hardware design of the ASV system.

conventional system looks simple, it is very vulnerable - if an error occurs at one point, the entire system will stop without warning. In contrast, in ROS, only the problematic part stops, and there is an error notification. Adding functions to the conventional system is difficult because data paths can collide, while in ROS, it is enough to use the same topic without the risk of data collision. In this study, ROS was run on a mini-PC for camera data processing, while the microcontroller functions as a motion system controller. The system connection diagram in the ROS environment can be seen in Figure 4.



Gambar 3. Block Diagram Differences between Systems Using ROS and Without ROS.



Gambar 4. Software integration block diagram.

Based on Figure 4, there are three nodes connected by three topics as data delivery paths. The process starts from the "usb_cam" node which takes a digital image from the webcam and sends it via the "img_raw" topic in the "image_smg" format to the "detect.py" node. The "detect.py" node functions to process data using the YOLOv5 method, then the data is sent to the ESP32 via the "cmd_vel1" topic in the "twist" data format, which is connected to the "serial_node.py" node. The "twist" format in the "cmd_vel1" topic has 6 float data slots, but only 3 slots are used to send rudder angle data, buoy detection status, and commands for scanning. Finally, the "serial_node.py" node is responsible for configuring the serial port and connecting communication between the mini PC and the ESP32 via "rosserial communication".

D. *System Testing Scenario*

The system evaluation aims to determine the capabilities and reliability of the prototype that has been created. There are three tests in this study. The first test is carried out by giving a detection object to the system to determine whether the system responds correctly or not. Then the second test is carried out with two different systems to determine the differences in system response in the old and new systems. The old system is not yet equipped with area localization and ROS capabilities, but the latest system has used the area localization and ROS methods. Furthermore, the last one is a test of the system's success rate. This test was carried out 50 times with different lighting conditions so that it is expected to determine the most ideal conditions for ASV to run well.

III. Results and Discussion

This chapter explains all the results of performance testing on the system that has been created. Because testing on the YOLOv5 method has been carried out by researchers in previous studies, in this study the system testing is focused on the real capabilities of the robot and the response of the ASV system.

A. *Initial Hardware and Software Testing Results*

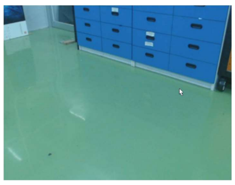
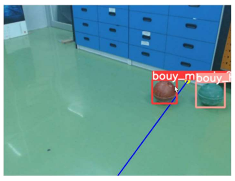
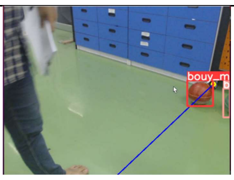
Initial testing of software and hardware aims to determine whether the system has been fully integrated and can function properly or not before direct testing is carried out through the entire track.

To integrate the entire system, a serial communication is used to send servo motor angle data, buoy detection, and commands for scanning test tracks from the mini-PC to the ESP 32 and ship release condition data from the ESP 32 to the mini-PC using the rosserial protocol. In the rosserial protocol, there are two modes that can be used, namely publisher and subscriber modes sent through a topic. Publisher mode is used to send data through a topic while subscriber mode is used to read data that has been published through a particular topic. To send data in this study, 3 topics were used, namely the topic "/cmd_vel1" with the data type "twis" which can accommodate as many as 6 data used to send data from the mini-PC to the ESP32, while for data sent from the ESP 32 to the mini-PC using the topic "switch_topic" with the data type bool and the topic "img_raw" is used to transfer image data from the usb_cam node to the detect.py node.

To determine the movement of the ship, the reference point of the ship's movement is used which is taken from the center point of the nearest object from the results of the mapping of the track area if the buoy is detected in more than one class. The following are the results of the ship's rudder movement response test based on camera data that detects red and blue buoys as shown in Table I.

From the servo motor movement test, the results show that the servo motor movement has an average movement accuracy of 99.56% of the desired command. With error test results below 5%, the system can be said to be feasible for further testing, namely automatic testing on the test track.

TABLE I. RESULTS OF RUDDER MOVEMENT TESTING BASED ON BUOY DETECTION INPUT.

No.	Detection	Data Sent (^{0})	Servo Motor Movement (^{0})	Accuracy (%)
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
Average error				99.56

B. ASV(Autonomous Surface Vehicle) Test Results Automatically on the Test Track

After the integration of hardware and software with good synchronization test results and can run on the gazebo simulator, the next stage is testing the ASV ship automatically through the test track. This test aims to determine whether the ship can determine the location of the track and go through the track properly without hitting and touching the track boundary buoy. An illustration of the test track is available in Figure 5.

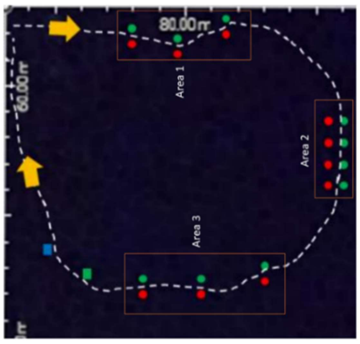


Figure 5. Illustration of Test Track

The evaluation was conducted in two sessions with each session being tested 5 times. The first session was conducted using the program before the application of the track area mapping capability and without the use of ROS. The program used in this test runs on the Windows 10 operating system. From the first session testing, the results presented in Table II were obtained.

TABLE II. AUTOMATIC TESTING RESULTS WITHOUT ROS AND MAPPING

Number of Test	Time Duration	Description
1.	1 minute 18 sec	Succeed
2.	-	Gagal (Track 3)
3.	1 minute 16 sec	Succeed
4.	-	Fail (Track 2)
5.	1 minute 17 sec	Succeed
6.	-	Fail (Track 1)
7.	1 minute 15 sec	Succeed
8.	1 minute 18 sec	Succeed
9.	1 minute 13 sec	Succeed
10.	-	Fail (Track 3)
Average Time	**1 minute 16.2 sec**	

From the test data presented in Table II, it can be seen that the ship had an average travel time of 1 minute 16.2 seconds on the test track with a success rate of 6 (Succeed) compared to 4 (Fail).

The second session testing was conducted using a program running on the ROS ecosystem and equipped with the ability to map the track area. This program runs on the Linux Ubuntu 20.04 operating system. The results of the second session testing are presented in Table III.

TABLE III. AUTOMATIC TESTING RESULTS WITHOUT ROS AND MAPPING

Number of Test	Time Duration	Description
1.	-	Fail (Track 2)
2.	1 minute 9 sec	Succeed
3.	-	Fail (Track 2)
4.	1 minute 9 sec	Succeed
5.	1 minute 6 sec	Succeed
6.	-	Fail (Track 2)
7.	1 minute 12 sec	Succeed
8.	1 minute 12 sec	Succeed
9.	1 minute 12 sec	Succeed
10.	1 minute 19 sec	Succeed
Average Time	**1 minute 11.2 sec**	

From the second session test data presented in Table III, it can be seen that the ship had an average travel time through the test track for 1 minute 11.2 seconds with a success of 7 (Succeed) to 3 (Fail) when using the track area mapping program.

After the two test sessions were carried out, it was seen that by adding the track area mapping capability, ship performance could be increased by up to 5 seconds. In addition to increasing performance, the author also observed that by adding the track area mapping capability, ship movement or ship maneuvers became more stable.

C. *Results of Automatic Ship Success Rate Testing on Test Track*

After the automatic testing is done and the ship can pass the test track well, the next step is to test the consistency of the ship in passing the test track. In the test of the ship's success rate, 5 testing sessions were carried out with 10 trials in each session. The first session was carried out at 07.56 to 08.35. The second session was carried out at 10.02 to 10.36. The third session was carried out at 12.20 to 13.05. The fourth session was carried out at 13.57 to 14.20. The fifth session was carried out at 16.13 to 16.46. The following are all the results of the experiments that have been carried out shown in Tables IV to VIII.

TABLE IV. RESULTS OF SUCCESS LEVEL TESTING IN THE FIRST SESSION

Number of Test	Time Duration	Description
1.	-	Fail
2.	1 minute 4 sec	Succeed
3.	-	Fail
4.	1 minute 4 sec	Succeed
5.	1 minute 5 sec	Succeed
6.	1 minute 7 sec	Succeed
7.	-	Fail
8.	-	Fail
9.	1 minute 3 sec	Succeed
10.	1 minute 6 sec	Succeed
Average Time	**1 minute 4.8 sec**	**Success = 60 %**

TABLE V. RESULTS OF SUCCESS LEVEL TESTING IN THE SECOND SESSION

Number of Test	Time Duration	Description
1.	1 minute 5 sec	Succeed
2.	1 minute 6 sec	Succeed
3.	1 minute 7 sec	Succeed
4.	-	Fail
5.	1 minute 6 sec	Succeed
6.	-	Fail
7.	1 minute 5 sec	Succeed
8.	1 minute 8 sec	Succeed
9.	-	Fail
10.	1 minute 7 sec	Succeed
Average Time	**1 minute 6,3 sec**	**Success = 70 %**

TABLE VI. RESULTS OF SUCCESS LEVEL TESTING IN THE THIRD SESSION

Number of Test	Time Duration	Description
1.	1 minute 5 sec	Succeed
2.	1 minute 9 sec	Succeed
3.	1 minute 6 sec	Succeed
4.	-	Fail
5.	-	Fail
6.	1 minute 4 sec	Succeed
7.	1 minute 6 sec	Succeed
8.	1 minute 7 sec	Succeed
9.	-	Fail
10.	1 minute 9 sec	Succeed
Average Time	**1 minute 6.6 sec**	**Success = 70 %**

TABLE VII. RESULTS OF SUCCESS LEVEL TESTING IN THE FOURTH SESSION

Number of Test	Time Duration	Description
1.	1 minute 7 sec	Succeed
2.	1 minute 9 sec	Succeed
3.	1 minute 7 sec	Succeed
4.	1 minute 16 sec	Succeed
5.	-	Fail
6.	-	Fail
7.	1 minute 5 sec	Succeed
8.	1 minute 12 sec	Succeed
9.	-	Fail
10.	1 minute 9 sec	Succeed
Average Time	**1 minute 7 sec**	**Success = 70 %**

TABLE VIII. RESULTS OF SUCCESS LEVEL TESTING IN THE FIFTH SESSION

Number of Test	Time Duration	Description
1.	-	Fail
2.	1 minute 11 sec	Succeed
3.	1 minute 14 sec	Succeed
4.	1 minute 14 sec	Succeed
5.	1 minute 14 sec	Succeed
6.	1 minute 13 sec	Succeed
7.	1 minute 17 sec	Succeed
8.	1 minute 18 sec	Succeed
9.	1 minute 15 sec	Succeed
10.	-	Fail
Average Time	**1 minute 14.5 sec**	**Success = 80 %**

Based on the success rate test, the best time achieved in this testing process occurred in the first session with a time of 1 minute 4.8 seconds while the longest time occurred in the fifth session and the average travel time of the ship was 1 minute 14.5 seconds. According to the test results data, it can be seen that in the fifth session there was a decrease in ship performance, this happened because the battery power began to weaken, in the fifth session it actually had the highest success rate of 80%. From the test data of the five test sessions presented, the ship was able to pass the track 35 times and failed 15 times from all the trials carried out. Thus, from the test it can be concluded that the ship has a good success rate in passing the test track with a success rate of 70 % of all the trials carried out.

IV. Conclussion

Based on the results of all tests conducted on the ASV ship in this study directly, it can be concluded that the use of ROS and mapping of the track area on the Autonomous Surface Vehicle (ASV) ship has proven effective in improving ship performance, with an average increase in travel time from 1 minute 16.2 seconds to 1 minute 11.2 seconds, and the best time reaching 1 minute 6 seconds. This shows an increase in performance of about 5 seconds faster than the previous system and improvements in ship maneuvers during testing. The ship successfully crossed the test track with a success rate of 70%, namely 35 successes out of 50 attempts in five sessions, with an average travel time of 1 minute 8.57 seconds. The fifth session recorded the highest success because the position of the light source was parallel to the ASV, thus increasing the accuracy of YOLOv5 in detecting track boundary buoy objects.

References

[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.