

# IMPLEMENTATION OF GLOBAL POSITIONING SYSTEM ON OBJECT FOLLOWING CAMERA HOLDING ROBOT BASED ON ESP32

Christine Gracia Lubalu<sup>a1</sup>, I Made Agus Dwi Suarjaya<sup>a2</sup>, Kadek Suar Wibawa<sup>b3</sup>

<sup>a</sup>Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana,  
Bukit Jimbaran, Bali, Indonesia

e-mail: <sup>1</sup>[lubaluchristine27@gmail.com](mailto:lubaluchristine27@gmail.com), <sup>2</sup>[agussuarjaya@it.unud.ac.id](mailto:agussuarjaya@it.unud.ac.id), <sup>3</sup>[suar\\_wibawa@unud.ac.id](mailto:suar_wibawa@unud.ac.id)

## Abstrak

Mengabadikan momen melalui foto dan video sering kali sulit dilakukan sendiri. Untuk itu, penelitian ini merancang robot berbasis ESP32 yang mampu mengikuti dan mendekati objek secara otomatis menggunakan teknologi GPS. Robot dilengkapi sensor HMC5883 sebagai penunjuk arah, sensor ultrasonik untuk menghindari hambatan, serta dikendalikan melalui aplikasi mobile berbasis IoT menggunakan protokol MQTT. Sistem ini memungkinkan pertukaran data lokasi secara real-time. Robot digerakkan motor DC yang dikontrol melalui motor shield L298 dan didukung modul stepdown LM2596 untuk kestabilan daya. Hasil pengujian menunjukkan robot mampu mengikuti posisi pengguna secara responsif dan menghindari rintangan. Penelitian ini menghadirkan alternatif solusi otomatisasi untuk pelacakan dan pengikut objek berbasis GPS dan IoT.

**Kata kunci:** Robot otomatis, ESP32, GPS, IoT, MQTT.

## Abstract

Capturing moments through photos and videos is often challenging when done alone. Therefore, this study designs an ESP32-based robot capable of automatically following and approaching objects using GPS technology. The robot is equipped with an HMC5883 sensor for direction indication, an ultrasonic sensor for obstacle avoidance, and is controlled via a mobile application based on IoT technology using the MQTT protocol. This system enables real-time location data exchange. The robot is powered by DC motors controlled through an L298 motor shield and supported by an LM2596 step-down module for power stability. Test results show that the robot can responsively follow the user's position and automatically avoid obstacles. This research offers an alternative automation solution for GPS- and IoT-based object tracking and following systems.

**Keywords :** Automatic robot, ESP32, GPS, IoT, MQTT.

## 1. Introduction

Taking pictures and videos as a form of capturing moments has become an important part of everyday life. These activities not only aim to document personal memories, but also to be shared on social media as a form of self-expression and showing off achievements [1]. However, these activities often require the assistance of other people or additional devices, which can be especially challenging for individuals traveling alone. For people traveling alone, a more practical and efficient way of capturing moments is needed, without dependence on others [2].

Seiring As technology advances, robotics offers increasingly relevant solutions to overcome these obstacles. Robots equipped with Global Positioning System (GPS) technology, ultrasonic sensors and digital compasses now make it possible to capture images and videos without the need for human assistance. Such robots, which can follow objects automatically, are

---

especially useful for solo individuals, such as solo travelers, who want to capture moments without the need for direct human intervention [3]. GPS technology is used to determine the position of the robot and the object being followed, while a digital compass sensor helps to set the robot's direction of movement more accurately.

In addition, Internet of Things (IoT) technology plays an important role in facilitating interaction between the robot and the user. By using MQTT-based communication, the robot can communicate in real-time with the mobile application used by the user, allowing remote control of the robot. This increases efficiency in image and video capture, making it more flexible and practical [4].

The use of GPS technology in this research is based on previous work by Bisma Ferriand et al. (2020), where an object-tracking robot was developed using a GPS sensor, digital compass, and Arduino with an On-Off control method to follow objects within a 30 cm range. Unlike the previous study, the current project extends the tracking distance, applies continuous position updates, and integrates IoT-based real-time communication using MQTT and a mobile application, making the system more dynamic, flexible, and suitable for outdoor use without relying solely on simple position differences [5].

In previous research by Gede et al. (2024), MQTT (Message Queuing Telemetry Transport) was used as a communication protocol to connect an automatic feeding system with a mobile application, utilizing a publish-subscribe model for scheduling feed times and transmitting real-time data with high efficiency and low latency. The system used a broker (HiveMQ) to relay messages between the ESP32 and the app. In contrast, this study adopts a similar MQTT approach but applies it to a mobile-controlled robot that tracks objects using GPS. Rather than transmitting feeding schedules or weight data, the MQTT protocol in this research facilitates continuous, real-time exchange of location coordinates between the robot and the user's mobile application. This implementation allows the robot to adjust its movement dynamically according to the user's position, highlighting a more complex interaction between hardware and software in mobile robotic navigation compared to a static control task [6].

## 2. Research Method / Proposed Method

To achieve the design goals of this GPS and IoT-based tracking robot, a systematic and structured research approach is needed. Therefore, this research uses the Research and Development (R&D) method which consists of several stages, starting from problem identification related to the need to take pictures independently without the help of others. After that, data collection was carried out through literature studies and observations to understand system needs more deeply. The next stage is the system modeling process which includes designing integration between hardware and software.

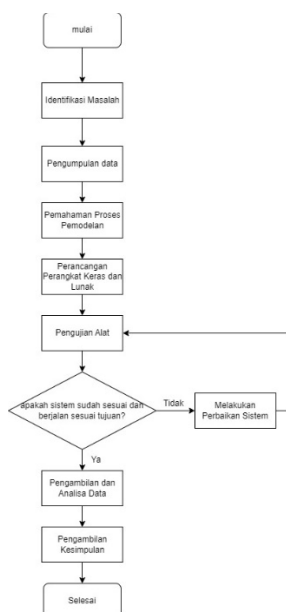


Figure 1. Flowchart of Research and Development (R&D) Method

The hardware used includes ESP32 as a microcontroller, GPS module for position tracking, HMC5883L sensor as a digital compass, ultrasonic sensor for obstacle detection, and implementation of MQTT-based communication for connection with mobile applications. After the system design is complete, testing is carried out to ensure that all components function as needed. If discrepancies are found, repairs and retesting are carried out until the system runs optimally. The test data is then collected and analyzed to evaluate the overall system performance.

### **3. Literature Study**

This section presents a review of several relevant previous studies as a basis for system development in this study. The literature reviewed includes research on the Internet of Things (IoT), object follower robots, utilization of GPS sensors, and integration of ESP32 microcontrollers with Android applications.

#### **3.1 Internet of Things (IoT)**

Internet of Things (IoT) is a concept in which various physical devices are connected through an internet network to exchange data. IoT enables automation, real-time monitoring, and integration of data from various sensors to support digital system-based decision making [7].

#### **3.2 ESP32**

The ESP32 is a microcontroller with a dual-core processor equipped with Wi-Fi and Bluetooth. This microcontroller has low power consumption, supports many communication interfaces such as UART, SPI, and I2C, and has security features such as SSL/TLS encryption. ESP32 is widely used in Internet of Things applications due to its flexibility and high performance [8].

#### **3.3 Neo-7M GPS Receiver Module**

Neo-7M is a GPS module that has high sensitivity and positioning accuracy up to  $\pm 2.5$  meters. It supports multi-GNSS such as GPS, GLONASS, and QZSS, and has fast signal acquisition capabilities, making it ideal for mobile tracking or robot navigation applications [9].

#### **3.4 DC Motor**

DC motors convert direct current electrical energy into mechanical motion energy in the form of rotation. This motor has advantages in ease of speed and direction regulation, as well as fast dynamic response. DC motors are often used in robotics, industrial automation, and control system applications [10].

#### **3.5 HC-SR04 Ultrasonic Sensor**

The HC-SR04 ultrasonic sensor is used to measure distance using ultrasonic waves. It has an effective measurement range of 2 cm to 400 cm, with an accuracy of approximately  $\pm 3$  mm. The HC-SR04 is widely used in robotic navigation systems, obstacle detection, and environmental mapping [11].

#### **3.6 L298N Motor Driver**

The L298N motor driver is a dual H-bridge module used to independently control the speed and direction of two DC motors. It operates using PWM signals and supports input voltages of up to 46V, with a maximum current of 2A per channel [12].

#### **3.7 HMC5883L Digital Compass Sensor**

The HMC5883L is a 3-axis magnetometer used to determine direction based on the earth's magnetic field. This sensor operates with an I2C interface, has high sensitivity, and is widely used in robotic navigation applications, drones, and digital orientation devices [13].

#### **3.8 MQTT and HiveMQ**

---

MQTT is a publish/subscribe communication protocol designed for lightweight communication between IoT devices. HiveMQ is an MQTT broker capable of handling large-scale message exchanges with high security and reliability, supporting real-time device connectivity [14].

### **3.11 Step Down LM2596**

LM2596 is a step-down DC-DC converter module used to step down the input voltage to a lower output voltage efficiently. This module is capable of handling currents up to 3A and has a conversion efficiency of up to 92%, making it ideal for battery-based applications [15].

### **3.13 Buzzer**

Active buzzers produce sound or sound automatically when given a DC voltage without the need for additional input signals. This buzzer is widely used as an alarm, warning, or notification indicator in electronic systems [16].

### **3.14 LCD I2C**

The I2C LCD is an I2C-based display module that allows a reduction in the number of communication cables between the microcontroller and the display. This module is commonly used to display text or system data in robotic projects, IoT, and monitoring systems [17].

## **4. Result and Discussion**

In this research, an ESP32 and Global Positioning System (GPS)-based camera support robot system that is able to follow the user's movement automatically has been successfully implemented.

### **4.1 Robot Design Flowchart**

In the initial stage of the process, the robot system starts with the initialization of all main components, namely ESP32 as a microcontroller, ultrasonic sensor for obstacle detection, Neo-7 GPS for positioning, HMC5883L compass sensor for direction, L298N motor driver to control the DC motor, and 2596 step-down module for power regulation. After that, the system initializes the Wi-Fi connection. If the connection is successful, the command to open the Android app is initialized, followed by the process of initializing the connection to the MQTT broker. If the connection to MQTT is successful, the robot will start receiving coordinate data from two sources, namely from the user's cell phone and from the Neo-7 GPS installed on the robot. These two data are used to determine the relative position of the robot to the user.

---

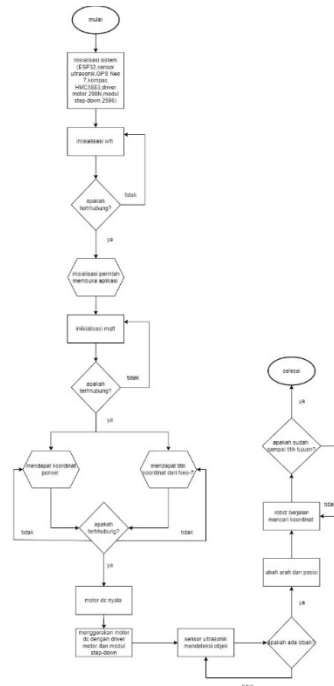


Figure 2. Robot Design Flowchart

Setelah After getting both coordinates, the DC motor will be turned on and the robot starts moving using the motor driver controlled by the ESP32. During the movement, the ultrasonic sensor works to detect the presence of objects or obstacles around the robot's path. If an object is detected, the robot will change its direction and position to avoid collision. The robot continues to walk looking for the destination coordinates until it approaches the user's location. When the robot has reached the intended coordinate point, the system will stop the robot movement and the process is declared complete.

## 4.2 System Overview

One of the implementations of this system is to use an ESP32 microcontroller that is able to connect to the internet via a WiFi network and communicate using the MQTT protocol. This system also utilizes various sensors to support the functionality of the robot, and is equipped with a mobile application to monitor and control it in real-time.

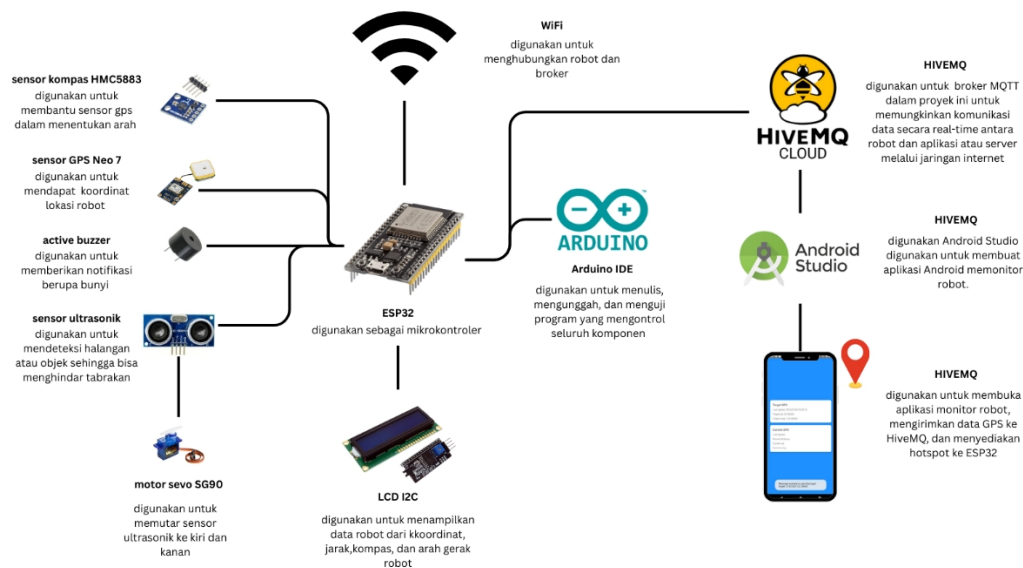


Figure 3. Robot System Overview

This diagram shows how a smart robot that can connect to the internet works. In the center is the ESP32, a kind of little brain that manages all parts of the robot. There is a compass sensor to know the direction, GPS to know the position of the robot on the map, ultrasonic sensors to detect obstacles in front, buzzers to give warning sounds, and small motors to move certain parts. All important information is displayed on a small screen (LCD). The robot is connected to the internet via WiFi and sends or receives data via HiveMQ Cloud. To create the robot program, Arduino IDE is used, while the application on Android phones is made with Android Studio so that we can monitor and control the robot remotely easily.

#### 4.3 Fishbone Diagram of Robot Design

To analyze the source of the problems that occur in the Global Positioning System Implementation system on the ESP32-based Object Following Camera Buffer Robot, the Fishbone Diagram method is used. This diagram helps identify the main factors that have the potential to cause problems in the system work process, starting from the use of tools, methods applied, processes that take place, to the expected output results.

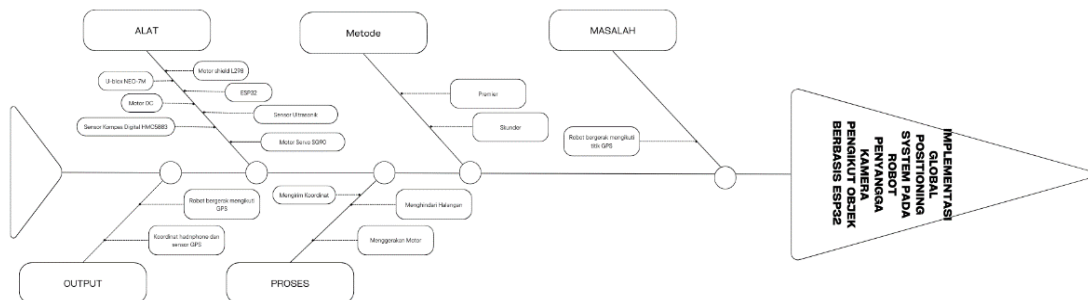


Figure 4. Robot Design Flowchart Diagram

The tool factor includes key components such as Neo-7M GPS sensor, HC-SR04 ultrasonic sensor, ESP32 as the main microcontroller, DC motor, HMC5883L digital compass sensor, and L298N motor driver. All of these tools must function optimally so that the system can run according to design. On the method side, the system requires steps such as setting the

coordinates of the robot and user, controlling the direction and position of the robot, and driving the motor to adjust the direction according to the coordinate data. Problems that arise can stem from problem factors, including GPS signal interference and robot mechanical structure factors that may hinder movement. The robot's work process starts from reading GPS coordinates, moving the robot based on the data, navigating through ultrasonic sensors to avoid obstacles, and adjusting the speed and direction of the motor. All these stages aim to produce output in the form of the robot's ability to follow the user's movements responsively and accurately. This fishbone diagram shows that each factor is interrelated and must be considered as a whole so that the ESP32-based GPS implementation system can function properly according to the research objectives.

#### 4.3 System circuit

The electronic circuit schematics were created using the Fritzing application, a popular software used to design and visualize electronic projects more easily and clearly.

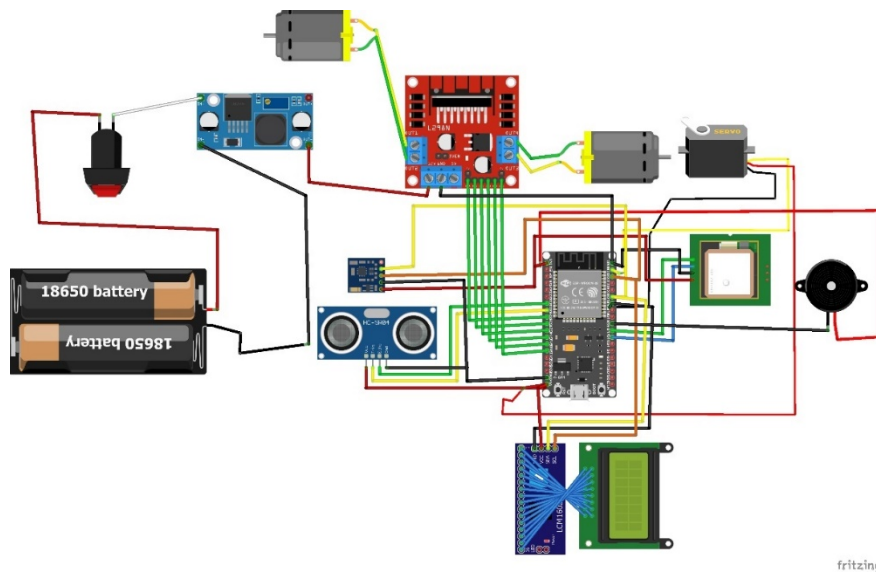


Figure 5. Robot System Circuit

The ESP32-based robot is powered by two 18650 batteries through a switch and a step-down module to regulate the voltage. The ESP32 serves as a control center that manages various components, such as DC motors controlled through L298N motor drivers to move the robot, SG90 servo motors for additional movement, GPS sensors to determine location, HC-SR04 ultrasonic sensors to detect obstacles, HMC5883L compass sensors to determine direction, buzzers to provide sound alerts, and I2C LCDs to display various important information. All components are connected with different colored cables to manage the flow of power and signals, so that the entire system can work together to support automatic robot navigation and control.

#### 4.4 Software System Implementation

The software implementation of this system uses a combination of the ESP32 microcontroller, MQTT broker service (HiveMQ), and Android application. ESP32 plays a role in processing data from the GPS sensor, then sending data on the robot's current position (`esp/robot/gps/current`), destination position (`esp/robot/gps/target`), and distance still to be traveled (`esp/robot/gps/distance`) to HiveMQ using the MQTT protocol. All this data is sent in real-time so that it can be monitored continuously. By using an MQTT server as an intermediary, data communication between the ESP32 and the Android application can take place quickly and efficiently, even remotely.









-8.79786 115.17380	 esp/robot/gps/current	0	
5.73	 esp/robot/gps/distance	0	
-8.79782 115.17380	 esp/robot/gps/target	0	
4.56	 esp/robot/gps/distance	0	

Figure 6. HiveMQ Topic Message

The Android application created with Android Studio serves to receive (subscribe) data from the HiveMQ broker and display the information to the user in a simple interface. The app has two main views, namely Target GPS which shows the destination coordinate information, and Current GPS which shows the robot's current position, remaining distance, as well as the last update time. With this implementation, the user can easily monitor the robot's movement through a smartphone, while ensuring that the robot is traveling in the right direction towards the set location.

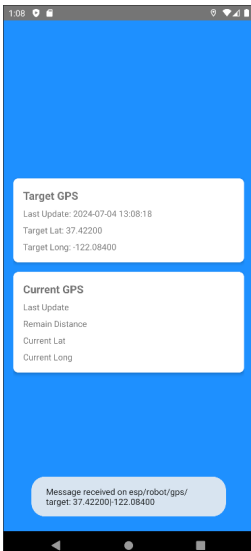


Figure 7. Robot App View

Briefly, this system connects the ESP32, HiveMQ, and Android app to monitor the position of the robot in real-time. GPS data from the robot is sent to the MQTT server and then displayed on the Android app, so that users can easily know the position, destination, and distance that the robot still has to travel remotely.

4.5 System Evaluation

System evaluation is the process of testing and assessing the performance of a system to ensure that all of its components and functions run in accordance with predetermined goals.

Table 1. System Evaluation

No	Testing	Scenario	Expected Result	Actual Result
1	Robot receives target coordinates	The Android application sends the target coordinates via MQTT	The robot correctly receives the target coordinates	Successful
2	Robot reads the current GPS position	The GPS sensor is active and connected to the ESP32	The robot's current location data is read and sent to MQTT	Successful



3	Robot calculates the distance to the target	After receiving the current position and target coordinates	The remaining distance is calculated and sent to the MQTT topic	Successful
4	Data is displayed on the Android application	The application is connected to the HiveMQ broker	Target data, current position, and distance are displayed on the application	Successful
5	Robot updates data in real time	The robot moves or changes location	Position and distance data are continuously updated on the application	Successful
6	Connection to HiveMQ is stable	The robot and Android application are active simultaneously	No connection disruption during data transmission	Successful
7	Robot follows the path toward the target	The robot moves based on GPS data	The robot sometimes deviates significantly, sometimes follows the path well	Sometimes deviates significantly, sometimes follows the path well
8	Robot correctly faces the target direction	The robot uses the compass sensor to adjust its heading	The robot accurately adjusts its orientation toward the target	Successful
9	Robot moves according to user direction	The user sets the direction via the Android application or coordinates	The robot moves following the direction provided by the user	Successful

Table 1 shows the evaluation results of the ESP32-based robotic system that uses GPS, compass, and MQTT connection through HiveMQ broker. Tests were conducted in nine scenarios to ensure system performance, ranging from receiving target coordinates, reading the current GPS position, calculating the distance to the target, to displaying data on the Android app. Overall, eight out of nine tests showed successful results, in accordance with the expected results. The robot was able to receive target coordinates, read GPS data, calculate distance, update data in real-time, maintain a stable connection to HiveMQ, face the target correctly, and move according to user directions. However, in testing the movement of the robot towards the target, the results were inconsistent; sometimes the robot was able to follow the path well, but sometimes it deviated far from the path, which is likely due to the limited accuracy of the GPS. As evidence of the experiments conducted, photo documentation of the robot experiments has been included and is shown in Figure 8. The photo reinforces the validity of the evaluation results that have been presented.



Figure 8. The Robot Follows the Object Well and Aligned

In Figure 8, it can be seen that the robot is able to follow the object well and maintain the direction of movement parallel to the object. The robot and the object move in a straight line without any deviation, indicating that the movement of the robot is quite stable and in accordance with the path intended by the robot so that in Figure 9 there is one other condition of the robot. The condition where the robot is less stable and has a slight deviation from the object's trajectory can be seen in Figure 9. In the figure, there is a small difference between the direction of the robot and the direction of the object, indicating that in certain situations, the stability of the robot's direction still needs to be improved.



Figure 10. Robot Slightly Deviated

Figure 10. shows the condition where the robot experienced a slight deviation from the main path while following the object. Despite the direction deviation, the robot is able to immediately make corrections by re-searching the user's position. After making adjustments, the robot again moves along the correct path and aligns with the direction of the object.

---



Figure 11. Robot Turning to Follow the Object

The robot can turn to follow the direction towards the target location even though the rotation process requires a considerable maneuvering distance. This happens because the robot needs to make adjustments to the position and direction of motion based on the GPS coordinate data obtained. Although the turning path taken is not always direct or narrow, in the end the robot is able to make course corrections and successfully turn in the right direction towards the predetermined target.



Figure 12. Rotating Robot Looking for Connection and Adjusting Coordinates

Under certain conditions when the robot loses connection with the internet network, the robot will perform rotating movements in a circular pattern. This happens because the robot loses real-time GPS coordinate update data. Once the connection is stable again, the robot will automatically readjust its position based on the latest GPS point. Because the data exchange system between the robot and the application uses the MQTT protocol, the stability of the internet connection is a very important factor to maintain the smooth navigation and movement of the robot towards the target.

## 5. Conclusion

The Global Positioning System Implementation Research on ESP32-based Object-Following Camera Buffer Robot successfully developed a robotic system that can automatically follow and approach objects using GPS technology, compass sensors, and ultrasonic, and controlled through IoT-based mobile applications with data communication using MQTT. The test results show that all hardware and software components work as expected, including the robot's ability to read GPS positions, avoid collisions, and display real-time data in the application. This system provides a practical solution in coordinate monitoring and automatic image/video capture, making it easier for users to explore the desired location. However, further development is

needed, especially in improving navigation accuracy and connection stability so that the robot can operate more optimally in various terrains and environmental conditions.

## References

- [1] N. E. Henryan and E. J. Simanjuntak, "Hubungan antara gambaran tubuh, keberhargaan diri, dan aktivitas swafoto di Instagram pada remaja perempuan," *Jurnal Psikologi Ulayat*, 2022, doi: 10.24854/jpu412.
- [2] A. Irvan and W. Wildian, "Rancang Bangun Tripod Kamera Otomatis Pengikut Objek Menggunakan Sensor Ultrasonik," *Jurnal Fisika Unand*, vol. 12, no. 4, 2023, doi: 10.25077/jfu.12.4.690-696.2023.
- [3] M. Marsono, R. Nurmalasari, and Y. Yoto, "PELATIHAN IMPLEMENTASI ROBOTIKA MANUFAKTUR UNTUK PENGUATAN KEMAMPUAN OTOMASI INDUSTRI 4.0 ALUMNI JURUSAN TEKNIK MESIN," *Jurnal Pengabdian Pendidikan dan Teknologi (JP2T)*, vol. 3, no. 1, 2022, doi: 10.17977/um080v3i12022p8-13.
- [4] M. Saiqul Umam, S. Adi Wibowo, and Y. Agus Pranoto, "IMPLEMENTASI PROTOKOL MQTT PADA APLIKASI SMART GARDEN BERBASIS IOT (INTERNET OF THINGS)," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 7, no. 1, 2023, doi: 10.36040/jati.v7i1.6131.
- [5] B. Ferriand, D. M. Putra, I. Y. Prasetya, N. Cholis Basjaruddin, and E. Rakhman, "Robot Pengikut Objek Menggunakan Global Positioning System(GPS)," Bandung, Aug. 2020. doi: <https://doi.org/10.35313/irwns.v11i1.1997>.
- [6] P. Gede, K. Mahadiputra, I. Made, A. D. Suarjaya, and K. S. Wibawa, "Automatic Pet Feeder Rotational Model Using MQTT and Mobile Application," Aug. 2024.
- [7] F. Susanto, N. K. Prasiani, and P. Darmawan, "IMPLEMENTASI INTERNET OF THINGS DALAM KEHIDUPAN SEHARI-HARI," *Jurnal Imagine*, vol. 2, no. 1, 2022, doi: 10.35886/imagine.v2i1.329.
- [8] J. Triyanto, M. P. Pradana, A. T. Permatasari, N. Rezika, and N. K. Daulay, "Monitoring Multi Sensor Esp 32 Secara Realtime Berbasis Website," *Escaf*, vol. 2, no. 1, 2023.
- [9] A. Z. Arfianto *et al.*, "Autopilot unmanned smart boat vehicle (Ausv) communication with lora rfm95," *International Journal on Informatics Visualization*, vol. 4, no. 4, 2020, doi: 10.30630/joiv.4.4.492.
- [10] Moh. N. Yuski, W. Hadi, and A. Saleh, "Rancang Bangun Jangkar Motor DC," *BERKALA SAINSTEK*, vol. 5, no. 2, 2017, doi: 10.19184/bst.v5i2.5700.
- [11] Dkk. Purwanto, H., "Komparasi Sensor Ultrasonik HC-SR04 Dan JSN-SR04T Untuk Apikasi Sistem Deteksi Ketinggian Air," *Jurnal SIMETRIS*, vol. 10, no. 2, 2020.
- [12] Azhari, T. I. Nasution, and P. F. A. Azis, "MPU-6050 Wheeled Robot Controlled Hand Gesture Using L298N Driver Based on Arduino," in *Journal of Physics: Conference Series*, 2023. doi: 10.1088/1742-6596/2421/1/012022.
- [13] F. Affandi, A. Izzuddin, and I. Apriia, "Implementasi Sensor Kompas Sebagai Sistem Navigasi Pada Robot vacuum cleaner," *Energy - Jurnal Ilmiah Ilmu-Ilmu Teknik*, vol. 11, no. 1, 2021, doi: 10.51747/energy.v11i1.1235.
- [14] S. Lin, L. Cui, and N. Ke, "End-to-End Encrypted Message Distribution System for the Internet of Things Based on Conditional Proxy Re-Encryption," *Sensors*, vol. 24, no. 2, 2024, doi: 10.3390/s24020438.
- [15] B. G. Bakshi, "An Integrated Circuit Based, Single-Inductor-Dual-Output Buck LED Driver for Dimmable and Color Temperature Tunable Indoor Lighting Application," *Journal of Circuits, Systems and Computers*, vol. 32, no. 6, 2023, doi: 10.1142/S0218126623501062.
- [16] K. Laili, T. Pangaribowo, and B. Badaruddin, "Robot Pendeteksi Gas Beracun Menggunakan NodeMCU Esp8266 Berbasis IoT," *Jurnal Teknologi Elektro*, vol. 10, no. 3, 2020, doi: 10.22441/jte.v10i3.006.
- [17] O. O. Akinwale, "Design, simulation and implementation of an Arduino microcontroller based automatic water level controller with I2C LCD display," *International Journal of Advances in Applied Sciences*, vol. 9, no. 2, 2020, doi: 10.11591/ijaas.v9.i2.pp77-84.