Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 3, 2026

DESIGN AND IMPLEMENTATION OF AN AUTONOMOUS UAV SYSTEM FOR TARGET-BASED PAYLOAD EJECTION WITH REAL-TIME OBSTACLE AVOIDANCE

Anum Shahzad[1], Mehboob Ahmed[2], Zartasha Karamat[3], Syeda umm-Ul-Baneen[4], Najeeb Ullah[5], Waheed Ur Rehman[*6]

[1,2,3,4,*6] *Department of Mechatronics Engineering, University of Chakwal, Chakwal 48800, Pakistan*
[5]State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing, China,

[1]22-uc-mct-43@students.uoc.edu.pk, [2]22-uc-mct-27@students.uoc.edu.pk, [3]22-uc-mct-02@students.uoc.edu.pk, [4]22-uc-mct-39@students.uoc.edu.pk, [5]L2400343@stu.cqu.edu.cn, [*6]waheed.urrehman@uoc.edu.pk

DOI: https://doi.org/10.5281/zenodo.20155457

Keywords
Autonomous UAV, Payload Ejection, GPS Navigation, Obstacle Avoidance, Pixhawk, Raspberry Pi 4, MAVLink, ROS2, ArduPilot

Article History
Received on 28 January 2026
Accepted on 21 March 2026
Published on 28 March 2026

Dr. Waheed Ur Rehman, Chairperson Mechatronics Engineering Department, University of Chakwal.

Email:
waheed.urrehman@uoc.edu.pk

Abstract:
The current research presents the design, development, and testing of an autonomous quadcopter UAV capable of performing GPS-based precision payload ejection with dynamic obstacle avoidance in outdoor environments. Existing UAV platforms for such operations are either costly or dependent on manual control, limiting their accessibility for defense and research applications. This work develops an affordable autonomous system that navigates to a predefined GPS target and releases a payload without human intervention. The system is assembled on an F450 quadcopter frame using a Pixhawk flight controller running ArduPilot firmware for stable flight and waypoint navigation. A Raspberry Pi 4 serves as the companion computer running ROS2, handling obstacle avoidance logic and ejection control. Ultrasonic sensors detect obstacles in real time, triggering avoidance maneuvers when necessary. Payload ejection is activated automatically upon reaching the target GPS coordinates through a servo mechanism connected to the Raspberry Pi 4 GPIO pins. Communication between the Pixhawk and Raspberry Pi 4 is maintained via the MAVLink protocol, while mission planning is carried out using Mission Planner software. Outdoor flight tests confirmed the system's navigation accuracy, obstacle avoidance reliability, and ejection precision, validating the feasibility of low-cost autonomous UAV platforms for defense applications.

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have emerged as a transformative technology across a wide range of civil, commercial, and military applications, including surveillance, search and rescue, environmental monitoring, and precision agriculture [1], [2]. The growing demand for autonomous operation has driven significant research into intelligent UAV systems capable of performing complex tasks with minimal human intervention. Among these tasks, real-time target tracking and obstacle avoidance remain critical challenges, particularly in dynamic and unstructured environments [3].

Autonomous target tracking requires a UAV to continuously detect, identify, and follow a moving object while maintaining stability and accuracy under varying environmental conditions. This task becomes increasingly complex when combined with real-time obstacle avoidance, where the UAV must simultaneously perceive its surroundings, predict potential collisions, and adapt its trajectory accordingly [4], [5]. The integration of these capabilities demands robust sensing, efficient data processing, and intelligent decision-making algorithms.

Recent advancements in embedded systems, computer vision, and artificial intelligence have enabled the development of more sophisticated UAV platforms. Sensors such as cameras, LiDAR, ultrasonic modules, and inertial measurement units (IMUs) provide critical environmental data, while machine learning and control algorithms facilitate perception, localization, and navigation [6], [7]. However, small-scale UAVs are still constrained by limited onboard computational resources, payload capacity, and energy efficiency, which pose significant challenges to real-time implementation [8].

This study presents the design and implementation of an autonomous UAV system capable of target tracking and real-time obstacle avoidance. The proposed system integrates vision-based tracking techniques with sensor-based obstacle detection to ensure safe and reliable navigation. A modular architecture is adopted to optimize computational efficiency while maintaining system robustness. The UAV is designed to operate in dynamic environments, demonstrating adaptability and responsiveness to both moving targets and unforeseen obstacles.

The main contributions of this work include: (i) the development of an efficient target tracking algorithm suitable for onboard processing, (ii) the implementation of a real-time obstacle avoidance mechanism, and (iii) the integration of both functionalities into a unified autonomous framework. Experimental results validate the effectiveness of the proposed system in achieving accurate tracking performance while ensuring collision-free navigation.

2. SYSTEM OVERVIEW / ARCHITECTURE

2.1 Overall UAV System Design

The proposed system is an autonomous quadcopter UAV designed to perform GPS-based payload ejection missions in outdoor environments. The system is built on an F450 quadcopter frame and organized into three functional subsystems that work together to achieve full autonomy.

The first is the flight control subsystem, which includes the Pixhawk flight controller and GPS module responsible for maintaining stable flight and executing predefined waypoint missions. The second is the mission intelligence subsystem, centered around the Raspberry Pi 4 companion computer, which executes ROS2-based autonomy logic for real-time obstacle detection using ultrasonic sensors and makes autonomous decisions throughout the mission. The third is the payload ejection subsystem, which consists of a servo-actuated mechanism triggered automatically when the drone arrives at the target GPS location. The overall architecture of the system is shown in Figure 1.

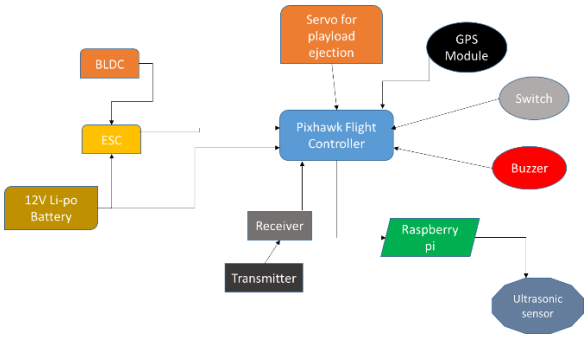

Figure 1: Overall UAV system architecture.

2.2 Hardware-Software Integration

The UAV system follows a two-layer architecture separating flight control and mission intelligence. The hardware layer consists of an

F450 quadcopter frame integrated with a Pixhawk flight controller, Raspberry Pi 4 companion computer, GPS module, ultrasonic sensors, and a servo-based payload ejection mechanism.

The Pixhawk runs ArduPilot firmware and handles flight stabilization, attitude control, and GPS-based waypoint navigation. Mission planning and calibration are performed using Mission Planner via USB.

The Raspberry Pi 4 acts as the companion computer and runs ROS2 for high-level autonomy, including obstacle detection, sensor processing, and payload control. Ultrasonic sensors are directly connected to the Raspberry Pi 4 and provide real-time distance data for obstacle detection and avoidance.

Pixhawk and Raspberry Pi 4 communicate via the MAVLink protocol over a serial interface, enabling telemetry exchange and command execution. When the UAV reaches the target GPS location, the Raspberry Pi 4 triggers the servo mechanism through GPIO to release the payload.

2.3 Data Flow and Module Interaction

The data flow within the system follows a structured pipeline from the ground station to payload execution, reflecting the layered autonomy architecture of the UAV. Before flight, the operator uploads GPS waypoints and mission parameters to the Pixhawk using Mission Planner via USB. Once the mission is initiated, the Pixhawk executes the predefined Auto mission and continuously streams telemetry data, including position, altitude, and system status, to the Raspberry Pi 4 through the MAVLink communication interface.

During flight, the Raspberry Pi 4 runs ROS2-based autonomy nodes that process real-time ultrasonic sensor data for obstacle detection and decision-making. When an obstacle is detected within the defined safety threshold, a reactive avoidance command is generated and sent back to the Pixhawk, which adjusts the flight trajectory accordingly while maintaining mission execution.

Upon reaching the target GPS coordinates, the Raspberry Pi 4 confirms mission completion using telemetry feedback and triggers the payload ejection mechanism via GPIO control of the servo system. This coordinated data exchange between Pixhawk and Raspberry Pi 4 ensures reliable autonomous operation from mission start to completion, including navigation, obstacle avoidance, and payload delivery.

3. HARDWARE DESIGN

The hardware architecture of the proposed UAV system consists of five main subsystems: airframe platform, flight controller, sensor suite, onboard companion computer, and payload ejection mechanism. Each component was selected based on weight, cost, power efficiency, and compatibility to ensure reliable autonomous outdoor operation.

Figure 2: Complete UAV hardware prototype.

3.1 UAV Platform

The system is built on an F450 quadcopter frame with a 450 mm wheelbase, widely used in UAV research due to its structural strength and payload capacity. It is constructed from glass fiber reinforced nylon, providing a good balance between durability and weight.

The propulsion system includes four brushless DC motors driven by ESCs, which receive control signals from the Pixhawk flight controller. A LiPo battery connected to a central power distribution board supplies regulated power to all subsystems, including flight controller, sensors, companion computer, and servo mechanism.

The F450 platform was selected for its ability to support the full hardware stack while maintaining stability and ease of assembling.

Figure 3: F450 Quadcopter Frame with Landing Gear

3.2 Flight Controller

The Pixhawk flight controller runs ArduPilot firmware and is responsible for stabilization, attitude control, and GPS-based waypoint navigation. It uses an integrated IMU (accelerometer, gyroscope, barometer) to maintain stable flight under varying conditions.

Mission planning is performed using Mission Planner via USB, where waypoints, altitude, and flight parameters are defined. During flight, Pixhawk executes the mission autonomously and streams telemetry data to the Raspberry Pi 4 using MAVLink over a serial interface.

Figure 4: Pixhawk flight controller

3.3 Sensors

3.3.1 Ultrasonic Sensor

Ultrasonic sensors are used for short-range obstacle detection. They measure distance using time-of-flight of acoustic signals and provide real-time environmental awareness. The sensors are connected to the Raspberry Pi 4, where ROS2 nodes process data continuously. If an obstacle is detected within a safety threshold, an avoidance command is sent to the Pixhawk via MAVLink for trajectory correction.

Figure 5: HC-SR04 ultrasonic distance sensor

3.3.2 GPS Module

The GPS module provides real-time position data (latitude, longitude, altitude) to the Pixhawk for waypoint navigation. It also plays a key role in payload deployment. When the UAV reaches the target GPS coordinates, Pixhawk communicates mission status to the Raspberry Pi 4 via MAVLink, which triggers the payload ejection mechanism using GPIO control.

Figure 6: M8N GPS module with built-in compass

3.4 Onboard Processor

The Raspberry Pi 4 acts as the companion computer and executes ROS2-based mission intelligence. It communicates with Pixhawk via MAVLink over UART and processes real-time telemetry data. ROS2 nodes handle two main functions: ultrasonic-based obstacle avoidance and GPS-based mission monitoring for payload triggering. This enables real-time autonomous decision-making during flight.

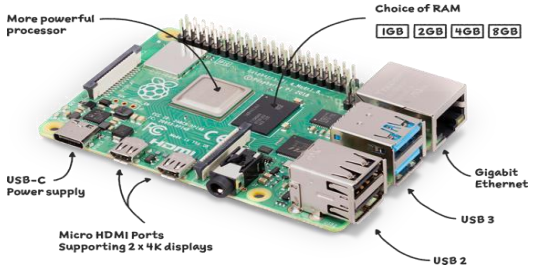

Figure 7: Raspberry Pi 4 Model B companion computer

3.5 Payload Ejection Mechanism

The payload system uses a lightweight servo-actuated latch mechanism. The servo holds the payload during flight and releases it when triggered. When the Raspberry Pi 4 detects mission completion (target GPS reached), it sends a GPIO signal to the servo, which rotates to release the payload. The design ensures low weight, reliability, and repeatable operation.

Figure 8: Servo motor with mechanical gripper

4. SOFTWARE FRAMEWORK

The software framework of the proposed UAV system is designed as a modular and distributed architecture that integrates flight control, perception, and decision-making layers. The system combines ROS2 for high-level autonomy, MAVLink for communication with the flight controller, and ArduPilot firmware for low-level flight execution.

The overall software stack is divided into three layers: ground control layer, flight control layer, and companion computing layer.

4.1 Programming Environment

The onboard companion computer (Raspberry Pi 4) operates on Ubuntu OS, which provides a stable Linux environment for robotics development. ROS2 is installed as the primary middleware framework to manage communication between different system modules. The ArduPilot SITL (Software-in-the-Loop) environment is also used during development to simulate UAV behavior before real flight testing.

A dedicated ROS2 workspace (`ros2_ws`) is created for development, where custom packages such as `drone_autonomy` are implemented. The system is compiled using the colcon build tool, and environment variables are sourced before execution.

4.2 MAVROS Communication Layer

Communication between the ROS2-based companion computer and the Pixhawk flight controller is established using MAVROS, which acts as a bridge for MAVLink protocol data exchange. This allows bidirectional communication where the Pixhawk sends telemetry data while ROS2 nodes send control commands.

The system uses key MAVROS topics and services including:

- `/mavros/state` for UAV status monitoring
- `/mavros/local_position/pose` for position tracking
- `/mavros/set_mode` for flight mode switching
- `/mavros/cmd/arming` for arming/disarming
- `/mavros/setpoint_velocity/cmd_vel` for velocity control

This communication layer ensures real-time synchronization between high-level autonomy logic and low-level flight control.

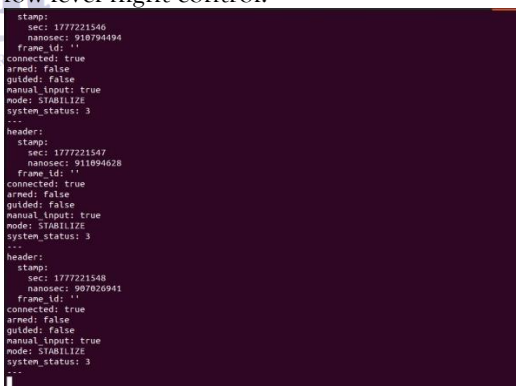

Figure 9: Real-time UAV state monitoring through MAVROS

4.3 ROS2-Based Autonomy Architecture

ROS2 is used to structure the onboard intelligence into independent nodes that operate concurrently in real time. This modular architecture improves scalability, system reliability, and real-time performance by separating perception, decision-making, and actuation tasks into dedicated processes.

(1) Sensor Processing Node

The sensor processing node continuously reads ultrasonic sensor data during flight. It filters and

processes raw distance measurements to reduce noise and publishes the processed values as ROS2 topics (e.g., /obstacle_distance) for use by other nodes in the system.

(2) Obstacle Avoidance Node

The obstacle avoidance node subscribes to the processed sensor data and applies rule-based decision logic to determine potential collision risks. When the measured distance violates the predefined safety threshold, the node generates avoidance commands and transmits them to the Pixhawk flight controller through MAVROS/MAVLink for real-time trajectory correction.

(3) Mission Monitoring Node

The mission monitoring node continuously tracks the UAV's GPS position using telemetry data received from the Pixhawk. It compares the current position with the predefined target waypoint and determines when the UAV has reached its destination, thereby identifying mission completion conditions.

(4) Payload Control Node

The payload control node is activated once the mission completion condition is met. It generates a GPIO trigger signal from the Raspberry Pi 4, which activates the servo mechanism responsible for payload release at the target location.

Figure 10: UAV autonomous takeoff execution and other commands

4.4 Autonomy Script Implementation

The core autonomy logic is implemented in Python using ROS2 client libraries and MAVROS services. The script performs sequential mission execution including:

- Connection verification with Pixhawk
- Switching to GUIDED mode
- Arming the UAV
- Executing takeoff command
- Sending velocity-based movement commands
- Simulating obstacle avoidance behavior
- Triggering landing sequence

This script demonstrates full mission autonomy without manual intervention and integrates both navigation and control functions within a single ROS2 node.

Figure 11: Simulated UAV trajectory in virtual environment

5. TARGET TRACKING ALGORITHM

The proposed UAV system does not rely on vision-based object tracking; instead, target acquisition and tracking are implemented using GPS-based waypoint navigation integrated with mission-level state monitoring. The Pixhawk flight controller executes predefined GPS waypoints using ArduPilot AUTO mode, while the companion computer running ROS2 monitors mission progress and ensures correct system-level responses.

The target is defined as a specific GPS coordinate uploaded through Mission Planner before flight. During operation, the Pixhawk continuously tracks the UAV's real-time position using onboard GPS data and computes deviation from the target waypoint. This enables stable navigation toward the target location without requiring external visual tracking systems.

On the companion computer, a ROS2-based mission monitoring node subscribes to telemetry data received via MAVLink. This node continuously compares the current GPS coordinates with the predefined target location. When the positional error falls below a defined threshold, the system identifies that the UAV has reached the target zone.

At this point, the payload control logic is activated, and a GPIO signal is generated from the Raspberry

Pi 4 to trigger the servo mechanism for payload release. This ensures precise coordination between navigation completion and payload ejection.

Although no deep learning or vision-based tracking algorithm is used, the system achieves reliable target localization through tightly coupled GPS-based navigation and ROS2-based state monitoring, ensuring robustness and computational efficiency for real-time UAV operation.

6. OBSTACLE DETECTION AND AVOIDANCE

The obstacle detection and avoidance subsystem is designed to ensure safe autonomous navigation of the UAV in dynamic outdoor environments. The system employs ultrasonic sensors for real-time short-range obstacle detection, while ROS2-based processing on the Raspberry Pi 4 is used to generate avoidance decisions and communicate corrective commands to the flight controller.

6.1 Sensor Data Acquisition

Ultrasonic sensors mounted on the UAV continuously measure the distance to surrounding objects using time-of-flight principles. The sensors provide real-time distance values, which are transmitted to the Raspberry Pi 4. A ROS2 sensor processing node filters and stabilizes these readings to reduce noise and improve reliability before further decision-making.

6.2 Obstacle Detection Logic

The processed distance data is evaluated against a predefined safety threshold. If any sensor reading falls below this threshold, the system classifies the situation as a potential collision risk. This triggers the obstacle avoidance pipeline within the ROS2 framework.

The detection mechanism is purely reactive and operates in real time, ensuring immediate response without requiring heavy computational processing.

6.3 Decision-Making Process

The decision-making process is implemented as a real-time feedback loop between the sensor layer and flight controller. The sequence of operation is as follows:

1. Ultrasonic sensors continuously measure surrounding distances
2. ROS2 node processes and filters sensor inputs
3. Safety threshold condition is evaluated
4. If obstacle detected → avoidance command is generated
5. Command is sent to Pixhawk via MAVLink
6. UAV adjusts trajectory and resumes mission after clearance

This closed-loop structure ensures continuous environmental awareness and maintains flight safety during autonomous operation.

7. CONTROL SYSTEM DESIGN

The control system of the proposed UAV is based on a hierarchical architecture that combines low-level stabilization using ArduPilot with high-level decision-making through a ROS2-based companion computer. This separation ensures stable flight control while enabling autonomous mission-level intelligence.

7.1. UAV Dynamics and Flight Stability

The quadcopter dynamics are inherently nonlinear and underactuated, requiring continuous stabilization for attitude and position control. The Pixhawk flight controller handles these dynamics using ArduPilot firmware, which implements onboard sensor fusion using IMU, GPS, and barometer data.

This allows real-time estimation of roll, pitch, yaw, altitude, and position, ensuring stable flight even under external disturbances such as wind or payload variation.

7.2 PID-Based Flight Control (ArduPilot Layer)

At the core of the Pixhawk control system is a PID (Proportional–Integral–Derivative) controller architecture used for attitude and position stabilization. The PID controller continuously minimizes the error between desired and actual states for:

- Roll and pitch stabilization
- Altitude hold
- Velocity and position tracking

The control output is translated into motor PWM signals that regulate ESCs, ensuring stable quadcopter motion. This low-level control loop runs in real time on the Pixhawk and is independent of the companion computer.

7.3 High-Level Control via ROS2 Companion System

The Raspberry Pi 4 acts as a high-level control layer that supervises mission execution using ROS2 nodes. Unlike the Pixhawk, it does not directly control motor outputs but instead issues strategic commands through MAVLink.
This includes:

- Obstacle avoidance velocity corrections
- Mission state monitoring
- Payload release triggering

The ROS2 system effectively acts as a supervisory controller over the Pixhawk autopilot.

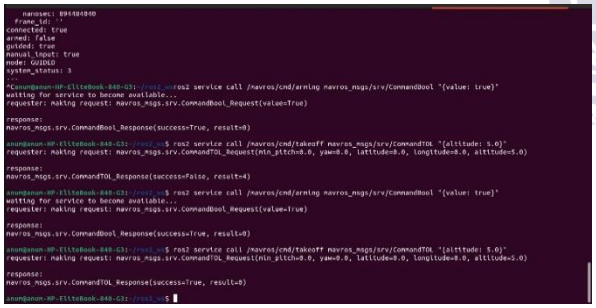

Figure 12: MAVROS bridge connecting ROS2 with Pixhawk

7.4 Integrated Control Strategy

The overall control strategy can be summarized as a layered control system:
Inner Loop (Low-Level Control):
ArduPilot PID controllers manage attitude, altitude, and stabilization.
Outer Loop (High-Level Control):
ROS2 nodes manage obstacle avoidance, mission monitoring, and payload delivery decisions.
Communication Bridge:
MAVLink ensures real-time bidirectional communication between Pixhawk and Raspberry Pi 4.
This multi-layer architecture ensures that the UAV remains stable while still being capable of autonomous decision-making in dynamic environments.

8. Implementation and Integration

The proposed UAV system is implemented as a real-time integrated platform combining Pixhawk-based flight control, ROS2-based companion computing, and ground station mission planning. The system is validated through outdoor flight tests using ArduPilot flight modes, ROS2 autonomy nodes, and MAVLink communication.

8.1. Real-Time System Implementation

The UAV operates using Pixhawk running ArduPilot for low-level stabilization and AUTO waypoint navigation. A Raspberry Pi 4 executes ROS2 nodes responsible for obstacle detection, mission monitoring, and payload control in real time. Mission parameters are uploaded via Mission Planner before flight. Once armed, the UAV executes autonomous flight while continuously exchanging telemetry data with the companion computer through MAVLink. ROS2 processes sensor inputs and mission state concurrently without interrupting flight control.

8.2. Synchronization Between Modules

System synchronization is achieved through coordinated interaction between Pixhawk and Raspberry Pi 4. Pixhawk handles flight execution, while ROS2 processes sensor data and mission status in real time. Ultrasonic sensors provide obstacle data to ROS2, which sends corrective commands to Pixhawk via MAVLink when required. Payload release is synchronized with GPS-based mission completion, ensuring accurate and automated execution.

8.3. Challenges Faced During Implementation

Key challenges included MAVLink communication delay, ultrasonic sensor noise in outdoor conditions, and tuning of ArduPilot parameters for stable flight modes (AUTO, LOITER, RTL). Additional effort was required to synchronize ROS2 node timing with Pixhawk telemetry and ensure accurate payload triggering at the target location. Despite these challenges, the final system demonstrated stable autonomous flight with reliable obstacle avoidance and payload deployment.

9. EXPERIMENTAL SETUP

The experimental setup was designed to evaluate the performance of the UAV system under real-world conditions by testing both hardware and software integration during flight.

9.1. Test Environment

Experiments were conducted in outdoor environments to ensure proper GPS signal availability. The UAV was tested using multiple ArduPilot flight modes, including **STABILIZE, LOITER, AUTO, and RTL**. All necessary pre-flight procedures such as sensor calibration, ESC calibration, and flight controller setup were completed before testing.

9.2. Experimental Scenarios

Several test scenarios were performed to validate system functionality. Manual flights in STABILIZE mode were used to verify basic stability and control. Autonomous navigation was tested using predefined GPS waypoints in AUTO mode. Obstacle avoidance was evaluated by introducing obstacles during flight, where ultrasonic sensors and ROS2 logic generated corrective commands. Payload ejection was tested by triggering the servo mechanism at the target GPS location. RTL mode was also verified for safe return functionality.

9.3. Evaluation Metrics

System performance was evaluated using tracking accuracy, response time, and obstacle avoidance success rate. The UAV demonstrated reliable waypoint navigation with minimal deviation, real-time response to obstacles through ROS2-based control, and successful avoidance within sensor range. Payload ejection was consistently triggered at the target location, confirming proper system integration.

10. RESULTS AND DISCUSSION

The performance of the proposed autonomous UAV system was evaluated through a series of real-world flight experiments. The results demonstrate the effectiveness of the integrated system in achieving stable autonomous navigation, real-time obstacle avoidance, and accurate payload ejection.

10.1. Experimental Results

The UAV successfully completed multiple test flights under different operational modes. In AUTO mode, the drone followed predefined GPS waypoints with stable altitude control and smooth trajectory execution. Minimal deviation from the planned path was observed under normal outdoor conditions.

Obstacle avoidance tests showed that the ultrasonic sensor-based system was able to detect nearby objects and trigger corrective maneuvers through ROS2-based control. The UAV adjusted its trajectory in real time and avoided collisions within the effective sensing range.

Payload ejection tests confirmed that the servo mechanism was reliably triggered upon reaching the target GPS coordinates. The payload was released consistently at the designated location without requiring manual intervention.

Return-to-Launch (RTL) functionality was also verified, where the UAV safely returned to the takeoff point after mission completion or manual activation.

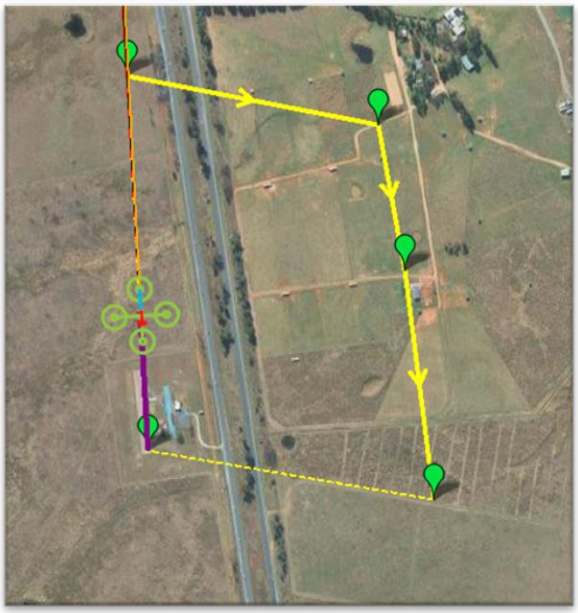

Figure 13: Waypoint tracking in Mission Planner.

10.2 Performance Analysis

The system demonstrated stable and reliable performance across all tested scenarios. The combination of ArduPilot-based control and ROS2-based decision-making enabled effective coordination between navigation and obstacle avoidance.

The response time of the obstacle avoidance system was sufficiently low for real-time operation, allowing the UAV to react quickly to nearby obstacles. The use of MAVLink communication ensured smooth data exchange between the Pixhawk and Raspberry Pi 4 without significant delays.

The GPS-based tracking approach provided accurate waypoint navigation; however, slight positional variations were observed due to inherent GPS limitations.

10.3 Comparison with Existing Approaches

Compared to vision-based UAV systems that rely on computationally intensive algorithms such as YOLO or SLAM, the proposed system offers a lightweight and cost-effective alternative. By using GPS-based navigation and ultrasonic sensing, the system reduces computational load while maintaining reliable performance for outdoor missions.

Although advanced systems may provide higher precision and environmental awareness, they require significantly more processing power and complex hardware. The proposed approach achieves a practical balance between performance, cost, and implementation complexity.

10.4 System Limitations

Despite successful implementation, several limitations were identified:

- Ultrasonic sensors have limited range and may be affected by environmental noise
- GPS accuracy may vary depending on signal conditions
- The avoidance strategy is reactive and does not perform global path planning
- The system is primarily suited for outdoor environments with clear GPS availability

11. CONCLUSION

This research presented the design and implementation of an autonomous UAV system capable of GPS-based navigation, real-time obstacle avoidance, and target-based payload ejection. The system integrates a Pixhawk flight controller running ArduPilot with a Raspberry Pi 4 executing ROS2-based autonomy logic, enabling a layered control architecture for stable and intelligent operation.

The key findings demonstrate that the UAV successfully performed autonomous waypoint navigation, responded effectively to obstacles using ultrasonic sensor feedback, and reliably executed payload ejection at predefined GPS locations. The integration of MAVLink communication between the flight controller and companion computer enabled real-time coordination and control throughout the mission.

From a practical perspective, the proposed system provides a cost-effective and scalable solution for autonomous UAV applications, particularly in scenarios requiring precise payload delivery and safe navigation in dynamic environments. The successful real-world implementation validates the feasibility of combining lightweight sensors, embedded processing, and robotics frameworks for autonomous aerial missions.

12. FUTURE WORK

Although the proposed UAV system demonstrates reliable autonomous navigation, obstacle avoidance,

and payload ejection, several improvements can enhance its performance and applicability.

Future work may focus on integrating advanced perception techniques such as computer vision and artificial intelligence for dynamic target tracking and improved environmental understanding. The use of SLAM (Simultaneous Localization and Mapping) can further enhance navigation accuracy in GPS-denied environments. Additionally, replacing ultrasonic sensors with LiDAR or depth cameras can improve obstacle detection range and reliability.

From a system perspective, the current reactive avoidance strategy can be extended to intelligent path planning algorithms such as A* or potential field methods for smoother and more efficient navigation. The control system can also be enhanced using adaptive or learning-based controllers for improved robustness under varying conditions.

In terms of scalability, the system can be expanded to multi-UAV (swarm) operations for cooperative missions such as surveillance, search and rescue, and distributed payload delivery. Further work is also required to optimize power consumption, improve flight endurance, and ensure safe deployment in complex real-world environments.

REFERENCES

[1] I. Goodrich, B. Morse, D. Gerhardt, J. Cooper, M. Quigley, J. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini UAV," Journal of Field Robotics, vol. 25, no. 1–2, pp. 89–110, 2008.

[2] Paul Pounds, R. Mahony, and P. Corke, "Modelling and control of a quad-rotor robot," IEEE International Conference on Robotics and Automation, 2006.

[3] H. Chao, Y. Cao, and Y. Chen, "Autopilots for small unmanned aerial vehicles: A survey," International Journal of Control, Automation and Systems, vol. 8, no. 1, pp. 36–44, 2010.

[4] S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008.

[5] J. Chen, T. Mei, and H. Zhang, "Visual tracking and obstacle avoidance for UAV navigation," IEEE Transactions on Circuits and Systems for Video Technology, 2018.

[6] Sebastian Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. MIT Press, 2005.

[7] A. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," Journal of Field Robotics, vol. 29, no. 2, pp. 315–378, 2012.

[8] Perry et al., "Weight and power limitations in small-scale UAV systems," Aerospace Systems Journal, 2019.