



International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 10, October 2026)

AerialEYE – AI-Based Smart Road Inspection and Environmental Monitoring Drone

Guided By: Mohini Kontamwar

Rahul Tembhare¹, Mohit Puneekar², Harshal Meshram³, Niraj Bhagat⁴

Abstract—Unmanned aerial vehicles (UAVs) provide a flexible platform for infrastructure inspection, but practical inspection systems require more than a flight controller or an image-processing model alone. They must coordinate mission planning, telemetry, flight commands, operator interaction, and inspection data within a dependable software architecture. This paper presents AerialEYE, an AI-based smart road inspection and environmental monitoring drone centered on a Flutter-based Android application, ArduPilot Software-in-the-Loop (SITL), MAVProxy, and MAVLink communication. The platform is designed to support mission creation and validation, telemetry monitoring, flight-command interaction, and future integration of computer vision and geospatial detection. The present evaluation focuses on the software and simulation layer because physical flight hardware, onboard camera, companion computer, and field deployment have not yet been validated. The implemented prototype successfully establishes the SITL–MAVProxy–MAVLink–Flutter communication path and demonstrates heartbeat reception, telemetry exchange, ARM, DISARM, GUIDED operation, and a simulated 5 m takeoff. The work therefore contributes a modular and safety-aware foundation for an aerial inspection workflow while explicitly separating validated functionality from future physical and AI validation. YOLO11-based pothole detection and GPS-based geotagging are retained as planned inspection extensions and are not presented as experimentally validated results.

Keywords—AerialEYE, ArduPilot, Flutter, MAVLink, MAVProxy, mission planning, pothole detection, Software-in-the-Loop, UAV

I. INTRODUCTION

AerialEYE is an autonomous/assisted drone inspection platform built around a camera-equipped multirotor and a Flutter Android operator application. The project is intended for smart road inspection and environmental monitoring, with the target use cases of pothole, stray-animal and waste detection. The operator application combines mission planning, live telemetry, mapping and inspection results in one workflow, while flight-critical stabilization and failsafe functions remain with the ArduPilot flight controller.

However, an inspection application is not defined only by its detection algorithm. A deployable workflow must also coordinate mission definition, vehicle state, communication, operator control, and inspection outputs. ArduPilot provides an established autopilot environment, while MAVLink provides a communication protocol for telemetry and command exchange between flight controllers and external systems. Companion-computer and ground-station architectures based on ArduPilot and MAVLink have been investigated for trajectory management and UAV control [4]. ArduPilot documentation also provides SITL facilities for testing vehicle behavior and connecting multiple ground-control components through MAVProxy and network interfaces [5].

AerialEYE addresses this system-integration problem through a mobile-centered architecture. The Android application is responsible for operator interaction, mission planning, monitoring, and high-level workflow, whereas ArduPilot remains the flight-control layer. MAVProxy provides the intermediate routing and development interface used during simulation. This separation is intended to reduce coupling between the user interface and flight-critical control functions and to allow later addition of computer vision, GPS association, and inspection reporting.

The current work is deliberately scoped to the implemented software and simulation system. The physical flight controller, GPS, companion computer, camera, and field-validation stages are not yet available in the present prototype. Likewise, the YOLO11 pothole model has not yet been trained and experimentally evaluated within this study. Consequently, this paper does not claim real-flight autonomy, GPS geolocation accuracy, or AI detection performance. Instead, it reports the architecture, implementation, and simulation-based validation that are currently available.

The main contributions of this work are: (1) a modular architecture separating mobile operator functions from flight-critical control; (2) a Flutter-based mission and monitoring interface integrated with MAVLink; (3) a simulation-based validation workflow using ArduPilot SITL and MAVProxy; and (4) an extensible inspection data architecture designed to accommodate future computer-vision and geospatial detection modules.

II. RELATED WORK

The project definition identifies three closely related research directions: edge-AI pothole detection on UAV/edge hardware, UAV-based pothole detection from aerial imagery, and UAV-based waste detection. These studies motivate the use of onboard or edge computer vision for road and environmental inspection. AerialEYE extends this direction at the system level by combining the inspection concept with mission planning, MAVLink communication, live mapping and operator control.

For pothole detection, Asad et al. evaluated several deep-learning approaches and reported strong performance from YOLO-family models, including an edge-oriented evaluation using an OAK-D and Raspberry Pi [2]. The work shows the relevance of lightweight detection for road inspection, while also noting limitations such as missed detections at long distances. Zeng and Zhong later proposed an improved YOLOv8n-based road-damage detector, demonstrating continued development of deep-learning methods for road-surface defects [3]. AerialEYE is designed to use a future detection model as one module within a broader mission-to-inspection workflow rather than treating the detector as the complete system.

At the flight-control and communication level, Pita and Saavedra described an ArduPilot-based UAV trajectory management system using a companion computer and MAVLink communication with a ground station [4]. Their work supports the use of a layered UAV architecture in which flight control and higher-level processing communicate through standardized interfaces. MAVLink itself defines telemetry and command mechanisms and uses heartbeat messages to advertise system presence and detect connection loss [6], [7].

The literature therefore demonstrates mature solutions for individual components—UAV vision, pothole detection, ArduPilot control, and MAVLink communication. The focus of AerialEYE is the integration of these components around a mobile operator workflow.

The current contribution is not a claim that each technology is individually novel; rather, it is the implementation and simulation validation of a modular architecture that can be extended toward physical aerial inspection.

III. PROPOSED AERIALEYE SYSTEM

A. System Architecture

The target AerialEYE architecture consists of five principal elements: a Flutter Android application, a Raspberry Pi Zero 2 W companion computer, a SpeedyBee F405-class ArduPilot flight controller, a GPS module, and an onboard camera with the propulsion system. In the present development stage, the Flutter application communicates with ArduPilot SITL through MAVProxy and MAVLink/UDP. The physical companion computer, flight controller, GPS and camera shown in the project definition remain deployment-stage components.

AerialEYE Validated Software Architecture

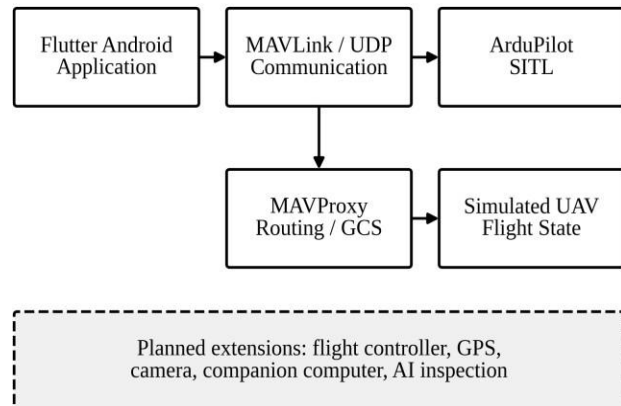


Fig. 1. Validated AerialEYE software architecture and planned physical/inspection extensions.

B. Mobile Application Layer

The Android application is implemented using Flutter. Its role is to provide the user with mission-related functions, vehicle connection information, telemetry, and flight-control actions. The project uses Riverpod for application state management and separates presentation components from communication and data-handling logic. The architecture is intended to support simulation and later physical operation without requiring a redesign of the complete user interface.

C. Flight-Control Layer

ArduPilot is retained as the flight-critical control layer. This design means that high-level application functions do not directly replace the autopilot's stabilization and vehicle-control responsibilities. In the current prototype, ArduPilot SITL provides the simulated vehicle and flight state. ArduPilot documentation supports SITL as a method for testing vehicle behavior and connecting external ground-control components [5].

D. MAVLink Communication Layer

MAVLink provides the interface between the application-side software and the simulated vehicle. The heartbeat mechanism is particularly important because it allows components to identify active systems and infer connection loss when expected heartbeats are no longer received [6]. AerialEYE uses MAVLink messages for vehicle discovery, telemetry reception, mode interaction, and command exchange. The current simulation forwards the MAVLink stream through MAVProxy to a UDP endpoint used by the Flutter application.

IV. SYSTEM METHODOLOGY

A. Development and Simulation Workflow

The development process begins with ArduPilot SITL, followed by MAVProxy initialization and UDP forwarding. The Flutter application then establishes a UDP socket and processes received data. Incoming traffic is initially checked so that non-MAVLink packets do not incorrectly enter the vehicle-state processing pipeline. Once a valid heartbeat is detected, the application identifies the simulated vehicle and begins processing telemetry.

B. Mission Planning

The mission workflow follows the project definition: connect to the vehicle, monitor live telemetry and the map, create waypoints or a survey-grid mission, validate the mission, upload it, explicitly start execution, and finally process inspection data. The present prototype validates the software-side mission and control workflow in simulation; real-camera capture and physical mission execution remain future stages.

C. Flight Command Workflow

The command workflow requires an explicit user action for critical operations. The current validated sequence includes connection, heartbeat detection, ARM, mode selection, and a simulated takeoff.

DISARM is also supported. The project design includes stale-heartbeat handling, mission validation, and recovery actions such as RTL/LAND for future physical deployment.

D. Future Inspection Pipeline

The intended inspection pipeline covers potholes, stray animals and waste. Camera frames are to be processed by an on-device computer-vision module, after which each detection can be associated with drone GPS position, timestamp and confidence and presented as map markers, gallery items and mission history. These AI and geotagging functions are part of the target system definition but are not claimed as experimentally validated in the present paper.

V. IMPLEMENTATION

A. Software Components

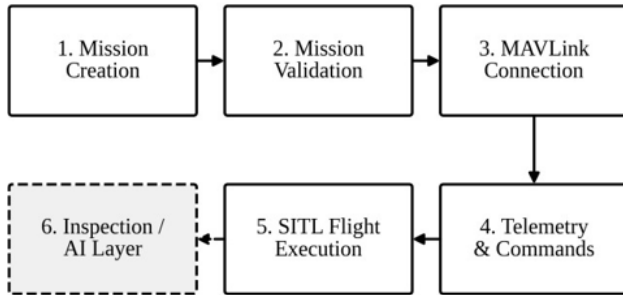
TABLE I
PRINCIPAL TECHNOLOGIES USED IN THE CURRENT PROTOTYPE

Component	Technology	Role
Mobile application	Flutter / Dart	Operator interface, mission planning and monitoring
State management	Riverpod	Application state and module coordination
Map layer	flutter_map / latlong2	Map-oriented mission and position visualization
Vehicle control	ArduPilot SITL	Simulation of autopilot and vehicle state
Routing	MAVProxy	MAVLink routing and simulation-side control
Communication	MAVLink / UDP	Telemetry and command exchange
AI inspection	YOLO11 (planned)	Planned pothole/road-defect detection

B. Physical Deployment Status

The project definition specifies a SpeedyBee F405 ArduPilot-compatible flight controller, GPS module, Raspberry Pi Zero 2 W companion computer, onboard camera, ESCs, motors and frame, battery/power distribution, RC receiver, and Android smartphone. These are the target implementation components. The present paper reports only the software and SITL validation because the physical flight stack has not yet been integrated.

Mission-to-Inspection Workflow



Dashed box: planned (not yet validated)

Fig. 2. Mission-to-inspection workflow represented by the current architecture.

VI. EXPERIMENTAL METHODOLOGY

A. Simulation Connectivity Test

The first experiment evaluated whether the Flutter application could receive and interpret the simulated UAV communication stream. ArduPilot SITL was launched and connected to MAVProxy. MAVProxy forwarded the telemetry stream through UDP to the application. The application then checked incoming packets and identified MAVLink heartbeat traffic.

B. Flight-Control Test

The second experiment evaluated essential control interactions. The tested operations were ARM, DISARM, GUIDED mode selection, and takeoff to approximately 5 m in the simulated environment. The purpose of this experiment was functional validation rather than measurement of real-aircraft flight performance.

C. Mission-Planning Test

The mission-planning module was evaluated functionally through mission creation and management in the mobile application. Because physical UAV hardware is not yet available, this test does not establish real-world waypoint tracking accuracy or navigation error.

D. AI Evaluation Scope

YOLO11 is intentionally excluded from the present experimental results. Although the project architecture reserves an inspection layer for computer vision, the model has not yet undergone the training and testing procedure required to report precision, recall, mAP, F1-score, inference time, or field detection performance. These measurements are therefore reserved for future work.

VII. RESULTS AND DISCUSSION

A. Communication and Vehicle Identification

The communication experiment successfully established the intended development path from ArduPilot SITL through MAVProxy and UDP to the Flutter application. The application was able to distinguish received traffic and process valid MAVLink information. Heartbeat reception provided the basis for identifying the simulated vehicle and establishing vehicle-state processing.

B. Flight-Command Validation

TABLE II
FUNCTIONAL VALIDATION OF THE CURRENT SIMULATION CONTROL PATH

Operation	Status	Interpretation
MAVLink connection	PASS	Application communicated with the simulated vehicle
Heartbeat	PASS	Vehicle presence was detected
ARM	PASS	Simulated vehicle responded to arm command
DISARM	PASS	Simulated vehicle responded to disarm command
GUIDED + 5 m takeoff	PASS	Simulated guided flight and takeoff were demonstrated

C. Mission-Planning Discussion

The mobile mission-planning functionality provides the operator workflow required for future autonomous inspection missions. However, the present evaluation does not establish physical waypoint tracking, cross-track error, GPS accuracy, or mission completion time. Those measurements require a physical aircraft or a deliberately instrumented simulation experiment with recorded telemetry.

D. Interpretation

The results demonstrate that the principal communication and control interfaces can be integrated without placing flight-critical control logic inside the mobile interface. This supports the proposed modular architecture. The main limitation is that simulation validates software interaction and command flow but cannot by itself establish real-world communication reliability, aerodynamic behavior, GPS accuracy, camera performance, or field inspection effectiveness.

VIII. LIMITATIONS AND FUTURE WORK

The current prototype has several limitations. First, it has been validated using ArduPilot SITL rather than a physical flight controller and aircraft.



International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 10, October 2026)

Second, the planned GPS, companion computer, camera, and propulsion hardware have not yet been integrated. Third, no real flight or field inspection campaign has been conducted. Fourth, the YOLO11 inspection model has not yet been experimentally trained and tested as part of this study. Finally, GPS-based geotagging and map markers for detected targets remain planned functions rather than validated experimental outputs.

Future work will proceed in stages. The first stage is physical flight-controller integration and safe bench testing. The second stage is GPS, companion-computer, and camera integration. The third stage is YOLO11 training and quantitative evaluation on a labeled pothole dataset. The fourth stage is integration of image detections with timestamped UAV position and mission identifiers. The final stage is controlled field testing, including communication reliability, mission completion, detection performance, geotagging error, and operational safety.

IX. CONCLUSION

This paper presented AerialEYE, a modular UAV inspection platform that integrates a Flutter mobile application, MAVLink communication, MAVProxy, and ArduPilot SITL into a unified operator workflow. The current prototype successfully demonstrates the communication path, heartbeat-based vehicle identification, telemetry reception, ARM/DISARM operations, GUIDED mode interaction, and a simulated 5 m takeoff. The architecture also provides a structured foundation for mission planning and future inspection functions.

The present work intentionally distinguishes simulation-validated capabilities from functions that require physical deployment.

In particular, real UAV flight, GPS geotagging, onboard camera processing, and YOLO11 detection performance are not claimed as completed experimental results. This limitation defines a clear path for subsequent development and quantitative evaluation. The resulting platform provides a practical software foundation from which a complete aerial inspection system can be developed and experimentally validated.

REFERENCES

- [1] R. Fan, J. Jiao, J. Pan, H. Huang, S. Shen, and M. Liu, "Real-Time Dense Stereo Embedded in a UAV for Road Inspection," in Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 535–543, doi: 10.1109/CVPRW.2019.00079.
- [2] M. H. Asad, S. Khaliq, M. H. Yousaf, M. O. Ullah, A. Ahmad, and Q. Chen, "Pothole Detection Using Deep Learning: A Real-Time and AI-on-the-Edge Perspective," *Advances in Civil Engineering*, vol. 2022, Art. no. 9221211, 2022, doi: 10.1155/2022/9221211.
- [3] J. Zeng and H. Zhong, "YOLOv8-PD: an improved road damage detection algorithm based on YOLOv8n model," *Scientific Reports*, vol. 14, Art. no. 12052, 2024, doi: 10.1038/s41598-024-62933-z.
- [4] J. L. Pita and F. O. Saavedra, "UAV Trajectory Management: Ardupilot Based Trajectory Management System," *Proceedings*, vol. 21, no. 1, Art. no. 8, 2019, doi: 10.3390/proceedings2019021008.
- [5] ArduPilot Development Team, "Using SITL for ArduPilot Testing," *ArduPilot Developer Documentation*, accessed Sep. 24, 2026.
- [6] MAVLink Development Team, "MAVLink Protocol Overview," *MAVLink Guide*, accessed Sep. 24, 2026.
- [7] MAVLink Development Team, "Heartbeat/Connection Protocol," *MAVLink Guide*, accessed Sep. 24, 2026.
- [8] B. K. S. B. Kumar, G. S., K. Manyam, R. Santhosh, et al., "Real-time Pothole Detection using YOLOv5 Algorithm: A Feasible Approach for Intelligent Transportation Systems," in Proc. 2nd Int. Conf. Electronics and Renewable Systems (ICEARS), 2023, doi: 10.1109/ICEARS56392.2023.10085336.