Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Remote Validation of UAV and UGV Companion Computer Software using an Esp32-Based Mavlink Bridge

Volodymyr Nazarenko
Department of Specialized Computer Systems
Lviv Polytechnic National University
Lviv, Ukraine

Abstract - The development of autonomous unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) increasingly relies on companion computers capable of executing computationally intensive tasks, including computer vision, artificial intelligence, sensor fusion, and autonomous navigation. However, experimental validation of such systems using physical vehicles introduces several challenges, including the risk of damaging expensive computing hardware, difficulties associated with prototype integration, and potential loss of experimental data. This paper proposes a remote companion computer testing architecture based on an ESP32-based MAVLink communication bridge running DroneBridge firmware. The proposed approach relocates the companion computer from the vehicle to a stationary position on the ground while maintaining bidirectional communication with the vehicle through an ESP-NOW long-range (LR) wireless link. The architecture was implemented and experimentally evaluated using an iFlight Chimera7 Pro V2 Analog multirotor UAV equipped with a BLITZ F722 flight controller running ArduPilot. The experimental setup also included two Waveshare ESP32-C6-Zero communication devices, a Raspberry Pi 5 companion computer, and a Skydroid 5.8 GHz 150-channel USB OTG video receiver. A ROS 2-based application running on the ground-based companion computer processed video and vehicle telemetry data to generate high-level control commands. Experimental results demonstrated successful bidirectional MAVLink communication and remote UAV operation over an approximate communication distance of 1 km. During the experiment, the UAV operated at an altitude of approximately 100 m above the take-off location. Although the proposed architecture is applicable to both UAV and UGV platforms, its experimental validation in this study was performed exclusively on a UAV. The proposed solution provides a compact and cost-effective approach to experimental validation of companion computer software without requiring the computing hardware to be installed directly on the unmanned vehicle.

Keywords - Unmanned Aerial Vehicle, Unmanned Ground Vehicle, Companion Computer, MAVLink, ESP32, DroneBridge, Remote Validation, Embedded Systems.

I. INTRODUCTION

In recent years, the development and practical application of unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) have accelerated significantly due to advances in embedded computing, artificial intelligence, computer vision, sensor technologies, and wireless communication. Modern autonomous robotic platforms incorporate companion computers responsible for executing computationally intensive tasks that exceed the capabilities of conventional flight controllers (FC) or vehicle control units (VCU). A companion computer typically communicates with the vehicle's low-level control system through the MAVLink (Micro Air Vehicle Link) protocol. This communication architecture enables the companion computer to receive vehicle telemetry, process sensor information, and transmit high-level control commands to the autopilot [1], [2].

The primary tasks performed by companion computers include real-time image processing, object detection and classification, sensor fusion, simultaneous localization and mapping (SLAM), autonomous navigation, and intelligent decision-making. The integration of artificial intelligence and edge computing into unmanned platforms has become an important research direction, enabling autonomous data processing close to the source of information and reducing dependence on remote computing infrastructure [3]. Depending on their intended applications, companion computers may incorporate multicore processors, graphics processing units (GPUs), neural processing units (NPUs), and other specialized hardware accelerators.

Although Software-in-the-Loop (SITL) and Hardware-in-the-Loop (HITL) simulation environments provide effective means of testing control algorithms and software components, physical experiments remain essential for evaluating the interaction between real hardware, communication interfaces, sensors, and the surrounding environment under conditions representative of actual operation [4], [5]. In particular, hardware-dependent timing characteristics, communication delays, and real-world sensor behavior cannot always be adequately reproduced by simulation.

However, installing experimental companion computer hardware directly on an unmanned vehicle introduces several practical challenges:

- Risk of hardware damage
- Physical integration limitations
- Potential loss of experimental data

These challenges motivate the investigation of alternative testing architectures in which the companion computer is physically separated from the unmanned vehicle and connected through a dedicated wireless communication channel. Such an approach requires a communication solution that provides sufficient range, bandwidth, and reliability while maintaining low implementation costs and compatibility with commonly available low-level controllers and embedded platforms.

This paper presents a compact and cost-effective communication architecture based on two ESP32 devices running the open-source DroneBridge firmware. The proposed solution establishes a bidirectional MAVLink communication link between an unmanned vehicle and a stationary ground-based companion computer, enabling remote software testing

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

during physical vehicle operation. Where required, additional separate wireless communication channels can be implemented to provide the ground-based companion computer with application-specific data, including video streams and information from onboard sensors.

II. PROBLEM STATEMENT AND OBJECTIVES

Experimental validation of autonomous robotic systems using physical hardware introduces several challenges when companion computers must be installed directly on the target vehicle. The main challenges addressed in this research are:

1. Risk of damaging expensive computing hardware. Companion computers frequently utilize high-performance computing platforms equipped with GPUs or NPUs. During the early stages of evelopment, researchers may also use experimental hardware prototypes manufactured in limited quantities. Installing such equipment on an unmanned vehicle exposes it to potential damage caused by collisions, crashes, or environmental factors.

2. Physical incompatibility with the target platform. Development hardware does not necessarily satisfy the mechanical and electrical requirements of the final vehicle. Prototype boards may have unsuitable dimensions, excessive power consumption, insufficient resistance to vibration and mechanical shock, or inadequate mechanical protection. Consequently, integrating experimental hardware into an existing UAV or UGV may require additional mechanical modifications and electrical interfaces.

3. Difficulties in recovering experimental data. Companion computers generally maintain application logs, diagnostic information, and experimental datasets on local storage devices, such as SD cards or eMMC modules. If the vehicle or computing hardware is damaged, recovering the stored information may require physical access to the equipment and, in some cases, hardware repair.

The identified problems can be mitigated by separating the companion computer from the vehicle and relocating it to a stationary ground-based location. In this architecture, the unmanned vehicle retains its conventional low-level control system, while communication between the vehicle and the remote companion computer is established through a dedicated wireless link. Vehicle telemetry is transmitted to the companion computer, and high-level control commands are returned to the flight controller or vehicle control unit. Additional data inputs for the companion computer may also be provided through additional wireless channels.

The main objectives of this research are:

- Develop and investigate a wireless communication approach for remote companion computer operation based on a bidirectional MAVLink bridge.
- Implement the proposed architecture using commercially available hardware and open-source software.
- Validate the proposed communication architecture through a physical UAV experiment and assess its applicability to UGV platforms.

III. COMMON VEHICLE SYSTEM ARCHITECTURE

A typical autonomous unmanned vehicle equipped with a companion computer consists of several interconnected subsystems. The main components include:

- Remote control receiver (RX). Receives pilot commands from the remote control transmitter.

- Video transmission system. Provides a wireless video stream from onboard cameras to the ground station.

- Flight controller or vehicle control unit. Performs low-level vehicle control, processes sensor measurements, and generates actuator commands.

- Companion computer. Executes high-level application logic, including computer vision, artificial intelligence, sensor fusion, and autonomous navigation.

- Sensors and peripheral devices. Include cameras, GNSS receivers, LiDAR sensors, IMUs, and other devices required by the autonomous application.

In a conventional architecture, the companion computer is installed directly on the vehicle, communicates with the flight controller through a serial MAVLink interface, and may also be connected directly to sensors and peripheral devices. This conventional arrangement of the main vehicle subsystems is illustrated in Fig. 1.

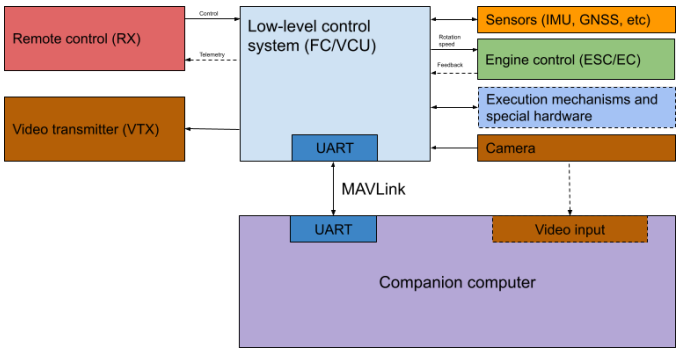


Fig. 1. Conventional architecture of an unmanned vehicle equipped with a companion computer

The flight controller or vehicle control unit is responsible for real-time stabilization, low-level motion control, and actuator management, while the companion computer performs computationally intensive tasks and generates high-level control commands. This division of responsibilities is commonly used in autonomous UAV systems and is supported by established autopilot platforms such as ArduPilot [1], [2]. A similar architecture can be implemented on UGV platforms equipped with a compatible vehicle control unit and MAVLink communication interface.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed architecture relocates the companion computer from the unmanned vehicle to a stationary ground-based location and is designed to support both UAV and UGV platforms. In both cases, the companion computer communicates with the onboard control system through a bidirectional wireless MAVLink bridge. For UAVs, the bridge interfaces with a flight controller, whereas for UGVs, it

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

interfaces with a vehicle control unit. The ESP32-based communication bridge is intended for MAVLink telemetry and control messages rather than high-bandwidth data transmission. Therefore, video streams and other high-bandwidth sensor data must be delivered through separate communication channels. The overall structure of proposed system, including the bidirectional MAVLink communication bridge and independent video transmission channel, is illustrated in Fig 2.

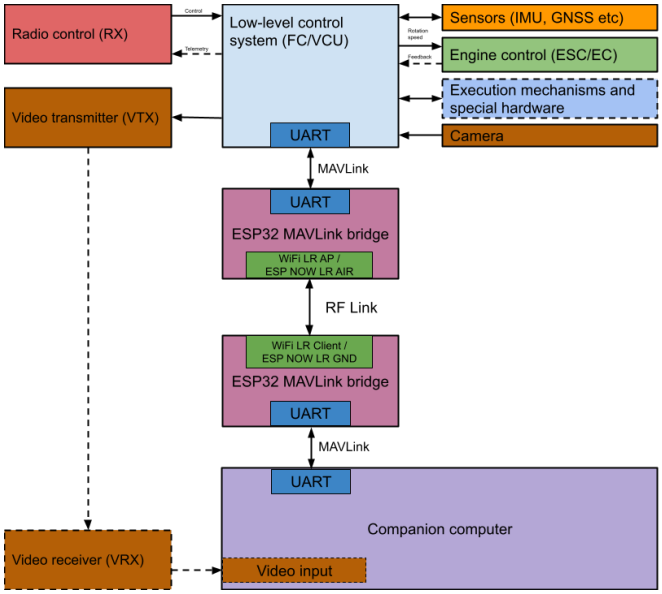


Fig. 2. Proposed remote companion computer architecture using an ESP32-based MAVLink bridge and an independent video transmission channel

The proposed system consists of the following components:

- Unmanned vehicle subsystem. The vehicle retains its original flight controller or vehicle control unit, remote control receiver, sensors, actuators, and other onboard equipment. The vehicle controller communicates with the first ESP32 device through a serial UART interface configured to transmit MAVLink messages.
- Wireless communication subsystem. Two ESP32 devices running DroneBridge firmware establish a bidirectional communication channel using ESP-NOW LR mode. The onboard ESP32 device receives MAVLink data from the flight controller and forwards it through the wireless link. The ground-based ESP32 device receives the transmitted data and provides it to the companion computer through a serial UART interface. The reverse communication path operates in the same way, allowing control commands generated by the companion computer to reach the flight controller. In the UAV implementation, a separate video transmission system provides the ground-based companion computer with the onboard camera feed. The video receiver captures the analog signal transmitted by the vehicle's video transmitter (VTX) and provides it to the companion computer through a USB interface. Other sensor data can be delivered through additional communication channels, depending on the requirements of the application.
- Ground-based companion computer subsystem. The companion computer is installed at a stationary location and connected to the ground-based ESP32 device through a UART interface. It executes the autonomous application, processes received telemetry and sensor data, performs the required computational operations, and generates high-level vehicle control commands. The application can be developed, tested, and debugged without requiring physical access to the vehicle-mounted hardware.
- Operator control subsystem. The remote control system remains available as an independent means of vehicle control. This allows the operator to retain manual control when necessary.

The proposed architecture is intended for experimental validation and testing. However, it does not eliminate the need for onboard safety mechanisms or a reliable manual control path.

V. ESP32-BASED MAVLINK COMMUNICATION BRIDGE

The proposed architecture uses two ESP32-based devices running DroneBridge firmware to establish a bidirectional wireless communication channel between the unmanned vehicle and a remote companion computer. The firmware provides transparent serial-to-wireless data forwarding and supports MAVLink communication, allowing the wireless link to replace the conventional direct UART connection between the flight controller and the companion computer [6].

The communication link is implemented using the ESP-NOW Long Range (LR) mode provided by DroneBridge. In this configuration, one ESP32 device operates as the airborne unit (AIR), while the second operates as the ground unit (GND). The devices exchange data using connectionless ESP-NOW broadcast packets, eliminating the need for a Wi-Fi access point or external network infrastructure. The DroneBridge implementation uses AES-256-GCM encryption to protect transmitted data [6]. The LR configuration prioritizes communication range over data throughput. According to the DroneBridge documentation, the nominal data rate in this mode is reduced to approximately 0.25 Mbit/s. Consequently, the solution is intended primarily for telemetry and low-rate control communication rather than high-bandwidth data transmission [6]. Its suitability for the proposed architecture is based on the relatively low data volume required for MAVLink telemetry and high-level vehicle control messages.

In the proposed implementation, the AIR and GND devices provide transparent bidirectional forwarding between the vehicle controller and the remote companion computer through their respective UART interfaces.

Since the bridge is designed for telemetry and control traffic, video streams and other high-bandwidth sensor data must be transmitted through separate communication channels.

Since the bridge relies on serial communication and MAVLink, the same approach can be applied to UAV and UGV platforms equipped with compatible vehicle controllers.

VI. IMPLEMENTATION

The iFlight FPV Chimera7 Pro V2 Analog was selected as the UAV platform to evaluate the proposed approach. The BLITZ F722 flight controller installed on the platform was configured

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

to run ArduPilot firmware, providing flight control functionality and MAVLink communication. One of the available UART ports was configured for MAVLink communication at a baud rate of 921600 bit/s.

The Raspberry Pi 5 was selected as the ground-based companion computer for remote high-level UAV control. It was configured to run a ROS 2-based application implementing the high-level control logic and executing autonomous missions based on computer vision (CV) object recognition..

The wireless communication bridge between the UAV and the companion computer was implemented using two Waveshare ESP32-C6-Zero development boards equipped with external omnidirectional antennas featuring a nominal gain of 9 dBi. Both devices were programmed with DroneBridge firmware, providing bidirectional wireless forwarding of MAVLink telemetry and control messages. The onboard ESP32 device was connected to the UAV flight controller through a UART interface. The ground-based ESP32 device was connected to the Raspberry Pi 5 through its UART interface. The DroneBridge configuration is provided in the Table I:

TABLE I. DRONEBRIDGE CONFIGURATION SETTINGS

Configuration parameter	Configured value
Wireless mode on UAV side	ESP-NOW LR AIR
Wireless mode on ground side	ESP-NOW LR GND
Channel	6
UART serial protocol	MAVLink
UART baud	921600

The UART baud rate of 921600 bit/s defines the serial communication speed between each ESP32 device and its connected equipment. It should not be interpreted as the actual wireless data rate, which is determined by the ESP-NOW LR communication mode and its associated protocol overhead.

The Skydroid 5.8 Ghz 150-channel USB OTG video receiver was connected to the Raspberry Pi 5 via USB port, providing the control application with the video feed from the UAV.

A full list of selected hardware and software components is provided in the Table II:

TABLE II. SELECTED HARDWARE AND SOFTWARE COMPONENTS

Component	Selected hardware/software
Wireless bridge hardware	Waveshare ESP32-C6-Zero
Wireless bridge software	DroneBridge v2.4.1
Antennas	Beyondoor BY-2400-05-9 (2.4 GHz, 9 dBi, omnidirectional)
UAV platform	iFlight FPV Chimera7 Pro V2 Analog
Flight controller	BLITZ F722
Flight controller firmware	ArduPilot v4.7.3
Companion computer	Raspberry Pi 5, 16 GB
Companion computer software	ROS 2 Lyrical Luth (Patch Release 2)
Video transmitter	iFlight BLITZ Whoop 5.8 GHz 2.5 W VTX
Video receiver	Skydroid 5.8 Ghz, 150-channel USB OTG video receiver

VII. EXPERIMENTAL PROCEDURE

The experimental procedure was designed to validate the feasibility of operating a companion computer remotely while maintaining bidirectional communication with a physically operating vehicle.

The experiment consisted of the following steps:

1. Initialize autonomous vehicle and ground-based companion computer.

2. Verify the operation of the RC and MAVLink wireless communication links.

3. Verify the availability of the analog video stream on the ground-based companion computer.

4. Launch the vehicle using the conventional remote control system.

5. Verify that telemetry data from the vehicle control subsystem are successfully received by the ground-based companion computer.

6. Start the ROS 2 application on the companion computer.

7. Transfer vehicle control to the autonomous application by activating the appropriate operational mode.

8. Execute the autonomous mission using control commands generated by the remote companion computer.

9. Monitor mission execution and continuous communication availability.

10. Upon mission completion, return control to the remote operator or terminate the experiment.

The experiment was conducted using a UAV equipped with the proposed ESP32-based MAVLink bridge and a ground-based Raspberry Pi 5 companion computer. No experiments were conducted on a UGV as part of this study. Although the proposed communication architecture can also be applied to UGV platforms with compatible MAVLink interfaces, its practical applicability to ground vehicles requires separate experimental validation.

During the experiment, the UAV operated at an altitude of approximately 100 m above the take-off location. The communication distance between the UAV and the ground-based equipment reached approximately 1 km. During the experiment, the flight controller remained responsible for low-level stabilization and actuator control. The remote companion computer performed high-level processing of video and telemetry data, and generated vehicle control commands through MAVLink.

The remote control system remained available to the operator throughout the experiment.

VIII. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed ESP32-based MAVLink communication bridge was successfully implemented and integrated with the selected UAV hardware.

Bidirectional communication between the BLITZ F722 flight controller and the ground-based companion computer was established using two ESP32-C6 devices operating in ESP-NOW LR mode. The companion computer successfully received telemetry information from the flight controller and transmitted high-level control commands through the wireless communication link.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

The ROS 2-based application was executed on the stationary Raspberry Pi 5 and successfully performed remote UAV control during the experiment. The application received vehicle telemetry through the MAVLink bridge and video data through the separate analog video reception system.

Successful operation of the proposed communication architecture was demonstrated at a distance of approximately 1 km. During the experiment, the UAV operated at an altitude of approximately 100 m above the take-off location.

These results demonstrate the feasibility of using the proposed ESP32-based communication solution as a practical means of enabling remote companion computer testing without installing the computing hardware directly on the unmanned vehicle, while providing several advantages for experimental development:

- Reduced risk of hardware damage. Keeps expensive experimental computing equipment physically separated from the vehicle, reducing its exposure to mechanical damage during field testing.
- Simplified hardware integration. Eliminates the need for mechanical mounting and electrical integration of the companion computer into the vehicle, while preserving the existing onboard hardware configuration.
- Improved accessibility and maintainability. Provides direct physical access to the companion computer and its storage devices, allowing software development, debugging, and maintenance to be performed independently of the vehicle.
- Platform flexibility and cost efficiency. Allows different ground-based computing platforms to be used without modifying the vehicle's onboard hardware, potentially reducing the cost and complexity of early-stage development and testing.

A comprehensive quantitative evaluation of the communication channel requires additional measurements, including end-to-end latency, packet loss, effective throughput, signal strength, and link stability under different environmental conditions.

These parameters are particularly important when the remote companion computer is responsible for time-sensitive control or perception tasks.

IX. CONCLUSIONS

This paper presented an architecture for remote companion computer validation on UAV and UGV platforms using an ESP32-based communication bridge running DroneBridge firmware. The proposed architecture relocates the companion computer from the unmanned vehicle to a stationary ground-based workstation while maintaining bidirectional communication with the vehicle through an ESP-NOW LR wireless link. The architecture can be applied to both UAV and UGV platforms, provided that the onboard flight controller or vehicle control unit supports MAVLink communication through a serial interface.

The prototype was implemented and experimentally tested using an iFlight Chimera7 Pro V2 Analog multirotor UAV equipped with a BLITZ F722 flight controller running ArduPilot v4.7.3. The system included two ESP32-C6 communication devices, a Raspberry Pi 5 ground-based companion computer, and a Skydroid 5.8 GHz 150-channel USB OTG video receiver. A ROS 2-based application performed high-level vehicle control using video and telemetry data.

Experimental validation demonstrated successful bidirectional MAVLink communication and remote UAV operation over a distance of approximately 1 km, with the UAV operating at an altitude of approximately 100 m above the take-off location.

The experimental results confirm the feasibility of using an ESP32-based MAVLink bridge for remote testing of companion computer software during physical UAV experiments. Although the proposed architecture is also applicable to UGV platforms, its experimental validation on a ground vehicle remains a subject of future research.

The main advantage of the proposed architecture is the separation of expensive and potentially vulnerable computing hardware from the unmanned vehicle. This simplifies experimental hardware integration, reduces the exposure of expensive computing equipment to potential damage, and provides convenient access to diagnostic information and software development tools. Future work will focus on quantitative evaluation of communication performance, including end-to-end latency, packet loss, effective throughput, and link stability. Additional research will investigate the implementation and experimental validation of the proposed architecture on UGV platforms and its application to more complex autonomous systems involving distributed sensors and multiple computing nodes, potential limits of the data rate.

REFERENCES

[1] ArduPilot Development Team, “MAVLink Basics,” ArduPilot Developer Documentation. [Online]. Available: https://ardupilot.org/dev/docs/mavlink-basics.html.

[2] ArduPilot Development Team, “Companion Computers,” ArduPilot Developer Documentation. [Online]. Available: https://ardupilot.org/dev/docs/companion-computers.html.

[3] P. McEnroe, S. Wang, and M. Liyanage, “A Survey on the Convergence of Edge Computing and AI for UAVs: Opportunities and Challenges,” IEEE Internet of Things Journal, vol. 9, no. 17, pp. 15435–15459, September 2022, doi: 10.1109/JIOT.2022.3176400.

[4] A. Augello, S. Gaglio, G. L. Re, and D. Peri, “Simulation and Test of UAV Tasks With Resource-Constrained Hardware in the Loop,” 2021 IEEE International Conference on Smart Computing (SMARTCOMP), Irvine, CA, USA, 2021, pp. 347-352, doi: 10.1109/SMARTCOMP52413.2021.00071.

[5] M. Xu, W. Liu, C. Xu, Y. Zhang, K. Zhang, Y. Feng, Y. Xia, and D. Huang, “Implementing Hardware-in-the-Loop Protocol Simulation for UAV Networks,” 2023 28th Asia Pacific Conference on Communications (APCC), Sydney, Australia, 2023, pp. 172-177, doi: 10.1109/APCC60132.2023.10460600.

[6] DroneBridge Project, “DroneBridge Overview,” Dronebridge Docs. [Online]. Available: https://dronebridge.gitbook.io/docs/dronebridge-for-esp32.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)