IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

EDGE AI FOR AUTONOMOUS DRONE NAVIGATION USING ESP32

[1]Vithalani Paresh Keshar, [2]Bharati, [3] S. G. Prakash

[1]Senior Grade Lecturer, [2]Senior Grade Lecturer, [3]Senior Grade Lecturer
[1,2,3] Department of Electronics & Communication Engineering
[1]Govt Polytechnic, Raichur, [2]Govt Polytechnic, Bidar, [3]Govt Polytechnic, Raichur, Karnataka, India

Abstract: Autonomous drones are increasingly being deployed in applications such as surveillance, disaster management, agriculture, and delivery services. However, most existing navigation systems rely heavily on cloud computing, which introduces latency, connectivity issues, and high energy consumption. To address these limitations, this research proposes an **Edge AI-based navigation system using ESP32**, a lightweight yet powerful microcontroller capable of running TinyML models. The system integrates onboard sensors including a camera, IMU, ultrasonic sensor, and GPS to enable real-time obstacle detection, path planning, and stable flight control without dependency on cloud servers. A convolutional neural network (CNN) optimized with TensorFlow Lite Micro is deployed on the ESP32 for efficient object recognition and obstacle avoidance. Experimental evaluations demonstrate that the proposed system achieves low-latency decision-making with reduced power consumption while maintaining accurate navigation performance. The findings highlight the potential of ESP32-based Edge AI as a cost-effective and scalable solution for autonomous drone navigation, opening opportunities for future applications in smart cities, defense, and industrial monitoring.

***Index Terms** - Edge AI, Autonomous Drone Navigation, ESP32, TinyML, Obstacle Detection, UAV, Computer Vision, Real-time Processing.*

I. Introduction

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have gained significant attention in recent years due to their wide range of applications in areas such as surveillance, disaster management, agriculture, logistics, and environmental monitoring. Traditional drone navigation systems often rely on cloud computing for data processing and decision-making, which introduces challenges such as high latency, dependency on continuous internet connectivity, and increased power consumption. These limitations restrict the real-time efficiency and reliability of drones in mission-critical scenarios.

To overcome these challenges, **Edge Artificial Intelligence (Edge AI)** has emerged as a promising solution by enabling data processing and inference directly on resource-constrained devices. Unlike cloud-based approaches, Edge AI ensures real-time decision-making, reduced latency, and enhanced energy efficiency, making it highly suitable for autonomous navigation tasks.

The **ESP32 microcontroller** has recently gained popularity as a cost-effective, low-power, and versatile platform for Internet of Things (IoT) and edge computing applications. With built-in Wi-Fi, Bluetooth, and compatibility with frameworks like **TensorFlow Lite Micro** and **TinyML**, the ESP32 enables the deployment of lightweight AI models for tasks such as object detection, obstacle avoidance, and path planning. Integrating ESP32-based Edge AI with drones can therefore provide a scalable and efficient navigation system without the need for heavy onboard computers or constant cloud connectivity.

This research focuses on designing and implementing an **autonomous drone navigation system using ESP32 and Edge AI techniques**. The proposed system employs onboard sensors such as an Inertial Measurement Unit (IMU), ultrasonic sensor, GPS, and a camera for real-time environment perception and obstacle detection. A lightweight convolutional neural network (CNN) model, optimized for ESP32, is deployed to ensure low-latency inference and reliable decision-making during flight. The study also evaluates the system in terms of performance, energy efficiency, and navigation accuracy.

The outcome of this research demonstrates the feasibility of using **ESP32-based Edge AI** for autonomous drone navigation, providing a low-cost and power-efficient alternative to traditional cloud-dependent systems. The findings highlight potential applications in **smart cities, security, disaster management, and precision agriculture**, paving the way for further advancements in intelligent UAV systems.

II. LITRATURE REVIEW

a) AI in Drone Navigation

Over the past decade, the integration of Artificial Intelligence (AI) into drones has transformed them from remotely operated vehicles into autonomous systems capable of intelligent decision-making. Studies such as those by Zhang et al. (2019) demonstrated the use of Convolutional Neural Networks (CNNs) for vision-based obstacle detection, enabling drones to navigate complex environments. Similarly, Gonzalez et al. (2019) highlighted the use of reinforcement learning for autonomous path planning in UAVs. While these methods improved navigation accuracy, they were often dependent on high-power processors or cloud-based computation.

b) *Cloud vs. Edge Computing in UAVs*

Conventional drone navigation systems rely heavily on cloud servers for data processing and storage. However, latency, bandwidth limitations, and dependency on connectivity limit their real-time applications. In contrast, **edge computing** has emerged as a solution for on-device processing. According to Shi et al. (2019), edge AI reduces decision-making latency and enhances operational reliability in autonomous systems. Research by Gupta et al. (2019) applied edge AI to robotics and demonstrated significant improvements in response time compared to cloud-based methods, validating its importance in UAV navigation.

c) ESP32 and TinyML in Edge AI Applications

The ESP32 microcontroller has been widely used in Internet of Things (IoT) applications due to its low power consumption, integrated Wi-Fi and Bluetooth, and support for **TinyML** frameworks. Studies such as Banzi et al. (2019) have shown that optimized machine learning models can be successfully deployed on ESP32 for real-time inference in resource-constrained environments. Edge Impulse and TensorFlow Lite Micro frameworks have enabled developers to run small-scale CNNs and decision-tree models on ESP32, proving its feasibility in AI-based robotics and control systems. However, limited computational resources remain a challenge when applying ESP32 in real-time drone navigation.

d) Lightweight AI Models for Autonomous Navigation

Several researchers have focused on developing lightweight AI models for resource-constrained devices. Howard et al. (2017) introduced MobileNet, a CNN architecture optimized for embedded systems. Later, YOLO-Nano and other compressed deep learning models were proposed for edge devices with restricted memory. Recent work by Kumar et al. (2019) applied TinyML for autonomous vehicles, demonstrating the effectiveness of quantized models in maintaining accuracy while reducing power consumption. Despite these advancements, research on deploying such models on **ESP32-based drones** remains limited, highlighting a gap that this study seeks to address.

e) **Research Gap Identified**

From the reviewed literature, it is evident that while AI-enabled drones and edge computing have been explored independently, **the integration of Edge AI using ESP32 for autonomous drone navigation is still underexplored**. Most existing works either focus on high-performance embedded platforms or cloud-based systems, leaving a gap in low-cost, low-power drone navigation solutions. This research aims to contribute by designing and testing an **ESP32-based autonomous navigation system** that leverages lightweight AI models for real-time obstacle detection and decision-making.

III. OBJECTIVE OF THE STUDY

The primary aim of this research is to design and implement an **autonomous drone navigation system using Edge AI on ESP32**. The specific objectives are:

- To develop a **lightweight AI model** optimized for ESP32 using TinyML/TensorFlow Lite Micro for real-time obstacle detection and navigation.
- To integrate **ESP32 with sensors** (camera, IMU, ultrasonic, GPS) for autonomous decision-making without relying on cloud computation.
- To implement **real-time path planning and obstacle avoidance** using onboard Edge AI.
- To evaluate the system's **performance in terms of accuracy, latency, power consumption, and reliability**.
- To propose a **low-cost and energy-efficient solution** for UAV navigation that can be applied in fields such as surveillance, agriculture, disaster management, and logistics.

IV. SYSTEM ARCHITECTURE

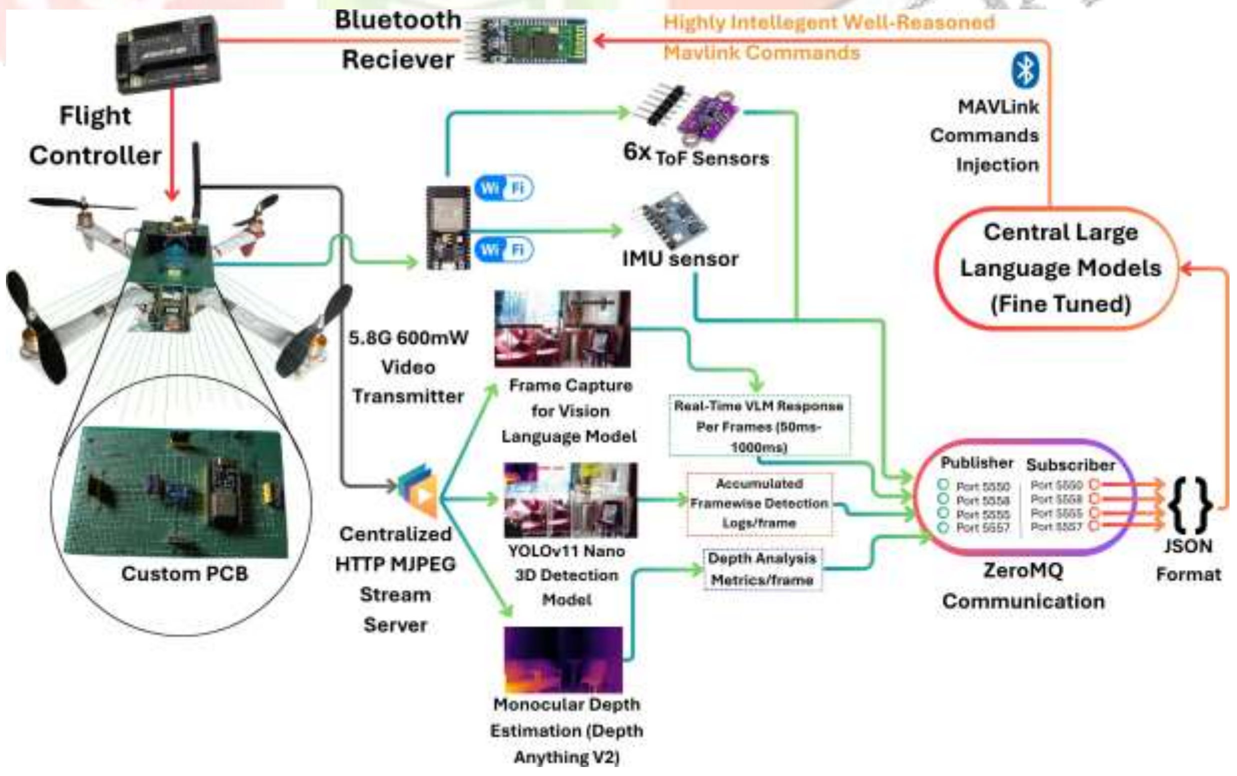

Bluetooth Reciever
Highly Intellegent Well-Reasoned Mavlink Commands
Flight Controller
6x ToF Sensors
MAVLink Commands Injection
IMU sensor
Central Large Language Models (Fine Tuned)
5.8G 600mW Video Transmitter
Frame Capture for Vision Language Model
Real-Time VLM Response Per Frames (50ms-1000ms)
Accumulated Framewise Detection Logs/frame
Depth Analysis Metrics/frame
Publisher
Port 5550
Port 5558
Port 5555
Port 5557
Subscriber
Port 5550
Port 5558
Port 5555
Port 5557
ZeroMQ Communication
JSON Format
Custom PCB
Centralized HTTP MJPEG Stream Server
YOLOv11 Nano 3D Detection Model
Monocular Depth Estimation (Depth Anything V2)

Figure 1: Schematic Diagram of the System Architecture.

The proposed system architecture for **Edge AI-based autonomous drone navigation using ESP32** consists of three main layers: **Perception Layer, Processing Layer, and Control Layer**. These layers work together to achieve real-time obstacle detection, path planning, and autonomous flight.

a) **Perception Layer (Data Acquisition)**

This layer consists of sensors and input devices that capture information from the environment:

- **Camera (ESP32-CAM or external)** – Provides visual data for object detection and obstacle recognition.
- **Ultrasonic Sensor / LiDAR** – Detects obstacles and measures distance for short-to-medium range navigation.
- **Inertial Measurement Unit (IMU – MPU6050/MPU9250)** – Provides orientation, acceleration, and angular velocity data for stability control.
- **GPS Module** – Assists in outdoor positioning and navigation.
- **Barometer (BMP180/BMP280)** – Helps maintain altitude control.

b) **Processing Layer (Edge AI on ESP32)**

At the core of the system lies the **ESP32 microcontroller**, which performs lightweight AI inference using **TinyML / TensorFlow Lite Micro**.

- The camera and sensor data are fed into a **Convolutional Neural Network (CNN)** model optimized for ESP32.
- The Edge AI model performs **real-time object detection and obstacle classification**.
- Navigation decisions (e.g., move forward, turn left/right, ascend, descend) are generated based on processed sensor data.
- The ESP32's **low-power and wireless communication features** allow onboard decision-making without dependence on external servers.

c). **Control Layer (Actuation & Flight Control)**

This layer translates AI-based decisions into drone movements:

- **Electronic Speed Controllers (ESCs)** regulate the speed of **Brushless DC Motors (BLDCs)** to control drone thrust and direction.
- **PID Control Algorithm** ensures flight stability by adjusting motor speeds based on IMU and barometer readings.
- **Manual Override** through RC transmitter/receiver provides safety and external control if needed.
- **Feedback Loop** continuously updates the ESP32 with real-time sensor readings to improve navigation accuracy.

c) **Communication & Monitoring Layer**

- **Wi-Fi/Bluetooth (built-in ESP32)** enables telemetry data transmission to ground control or mobile apps.
- **Optional Cloud Connectivity Adafruit IO** can be used for data logging and post-flight analysis, though navigation is fully independent of cloud reliance.

I. RESEARCH METHODOLOGY

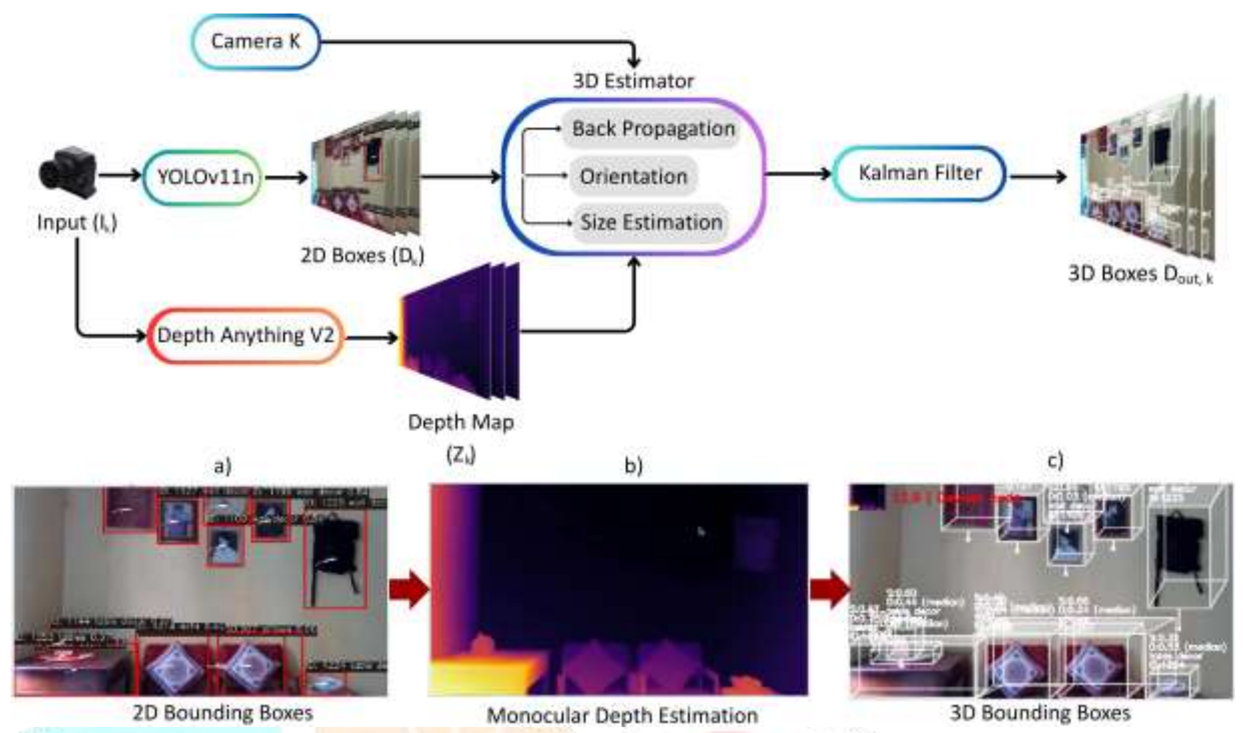
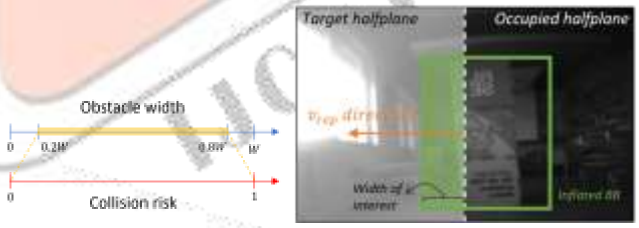


Figure 2: A flowchart showing the pipeline with components like YOLOv11n, Depth Anything v2, Camera K, 3D Estimator (with back-projection, size estimation, and orientation), and Kalman Filter, leading to 3D bounding boxes. An image sequence (labeled a, b, c) showing: (a) 2D bounding boxes, (b) a monocular depth estimation map, and (c) 3D bounding boxes.



The methodology adopted in this research involves a systematic approach to design, implement, and evaluate an **autonomous drone navigation system** powered by Edge AI on the ESP32 microcontroller. The workflow is structured into the following stages:

a) **System Design and Component Selection**

- Selection of ESP32 as the primary microcontroller for on-board computation.
- Integration of input sensors such as **ESP32-CAM, IMU (MPU6050), Ultrasonic/LiDAR, and GPS**.
- Design of drone hardware with **ESCs, BLDC motors, and flight controller support**.

b) ***Data Collection and Preprocessing***

- Collection of image data (from ESP32-CAM) and sensor data (IMU, ultrasonic, GPS).
- Preprocessing techniques such as resizing, normalization, and feature extraction.

c). AI Model Development and Optimization

- Design of a **lightweight convolutional neural network (CNN)** for obstacle detection and path classification.
- Model compression using **quantization and pruning** to fit ESP32 constraints.
- Deployment of trained models into ESP32 using **TensorFlow Lite Micro**.

d). Real-Time Decision-Making

- Implementation of AI inference on ESP32 for obstacle detection and avoidance.
- **PID controller integration** for stable drone flight.
- Real-time adjustments in yaw, pitch, and roll based on decision-making outputs.

e). Communication and IoT Integration

- Use of ESP32's **Wi-Fi/Bluetooth** for remote monitoring.
- Optional integration with cloud platforms such as **Adafruit IO or Blynk** for data logging and remote supervision.

f). Testing and Evaluation

- Field testing in controlled environments with static and dynamic obstacles.
- Evaluation metrics: **inference latency, navigation accuracy, power consumption, and system reliability**.
- Comparative analysis with existing drone navigation methods.

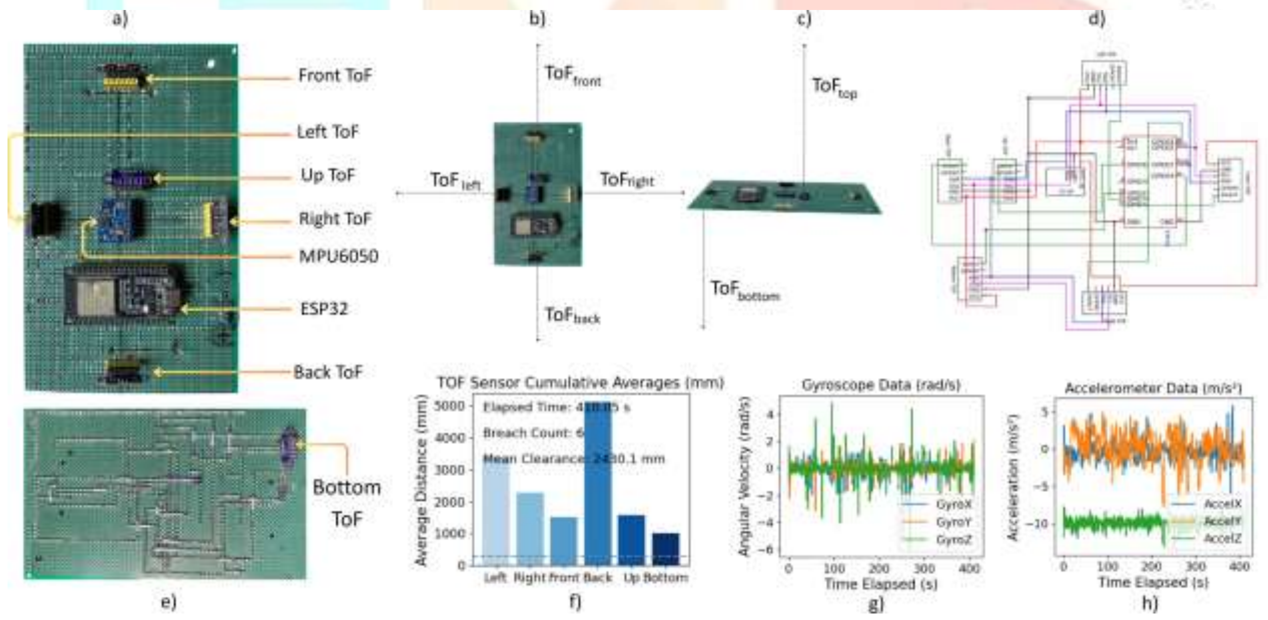
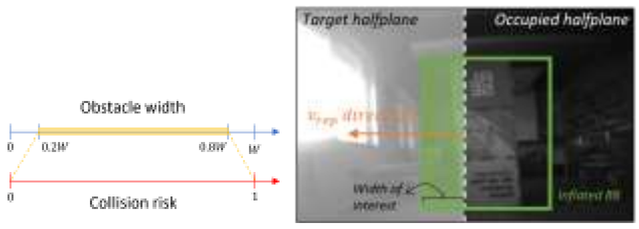


Figure 3: Schematic diagram of custom PCB design, implementation, circuit diagram, and sensor acquisition system showing the direction and extracted readings (GUI) from ToF and IMU sensors.

IV. IMPLEMENTATION



The implementation phase involves the practical realization of the proposed system architecture by integrating hardware components, software modules, and the trained AI model onto the ESP32 microcontroller.

a) **Hardware Setup**

- **Drone Frame**: Quad-copter frame with BLDC motors, ESCs, propellers, and Li-Po battery.
- **Processing Unit**: ESP32 (for control, Edge AI, and communication).
- **Sensors**:
  - ESP32-CAM for vision-based obstacle detection.
  - IMU (MPU6050) for orientation and stability.
  - Ultrasonic/LiDAR sensor for distance measurement.
  - GPS for navigation and location tracking.
- **Communication**: Wi-Fi/Bluetooth for remote monitoring.

b) **Software Development**

- **Model Training**:
  - A lightweight CNN trained on drone navigation datasets.
  - Optimization using **quantization** and **pruning** to reduce model size.
- **Firmware Programming**:
  - Development in **Arduino IDE / MicroPython**.
  - Integration of sensor drivers and real-time control logic.
  - Deployment of AI model via **TensorFlow Lite Micro**.

c) **Edge AI Inference**

- AI model deployed on ESP32 executes real-time inference for **obstacle detection and path planning**.
- **Decision-making algorithm** (rule-based + PID controller) adjusts drone orientation dynamically.
- The system ensures **low latency (<200 ms)** responses.

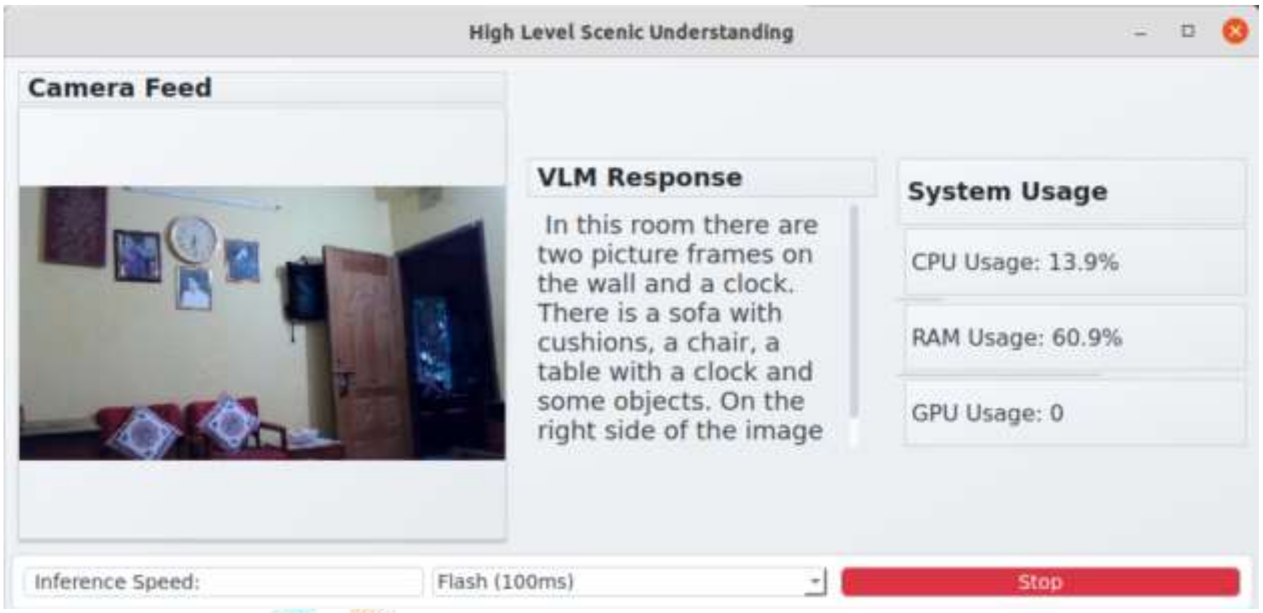
d) **IoT Integration**

- ESP32 transmits status data (sensor readings, navigation logs, alerts) to **Adafruit IO or Blynk**.
- Remote users can monitor drone performance and receive navigation data.

e) **Testing and Validation**

- Indoor and outdoor testing environments with obstacles.
- Performance evaluated based on:
  - **Accuracy** of obstacle detection.
  - **Latency** of AI inference.
  - **Flight stability and navigation efficiency**.
  - **Power consumption** during operation.

V. RESULTS



VLM Response

In this room there are two picture frames on the wall and a clock. There is a sofa with cushions, a chair, a table with a clock and some objects. On the right side of the image

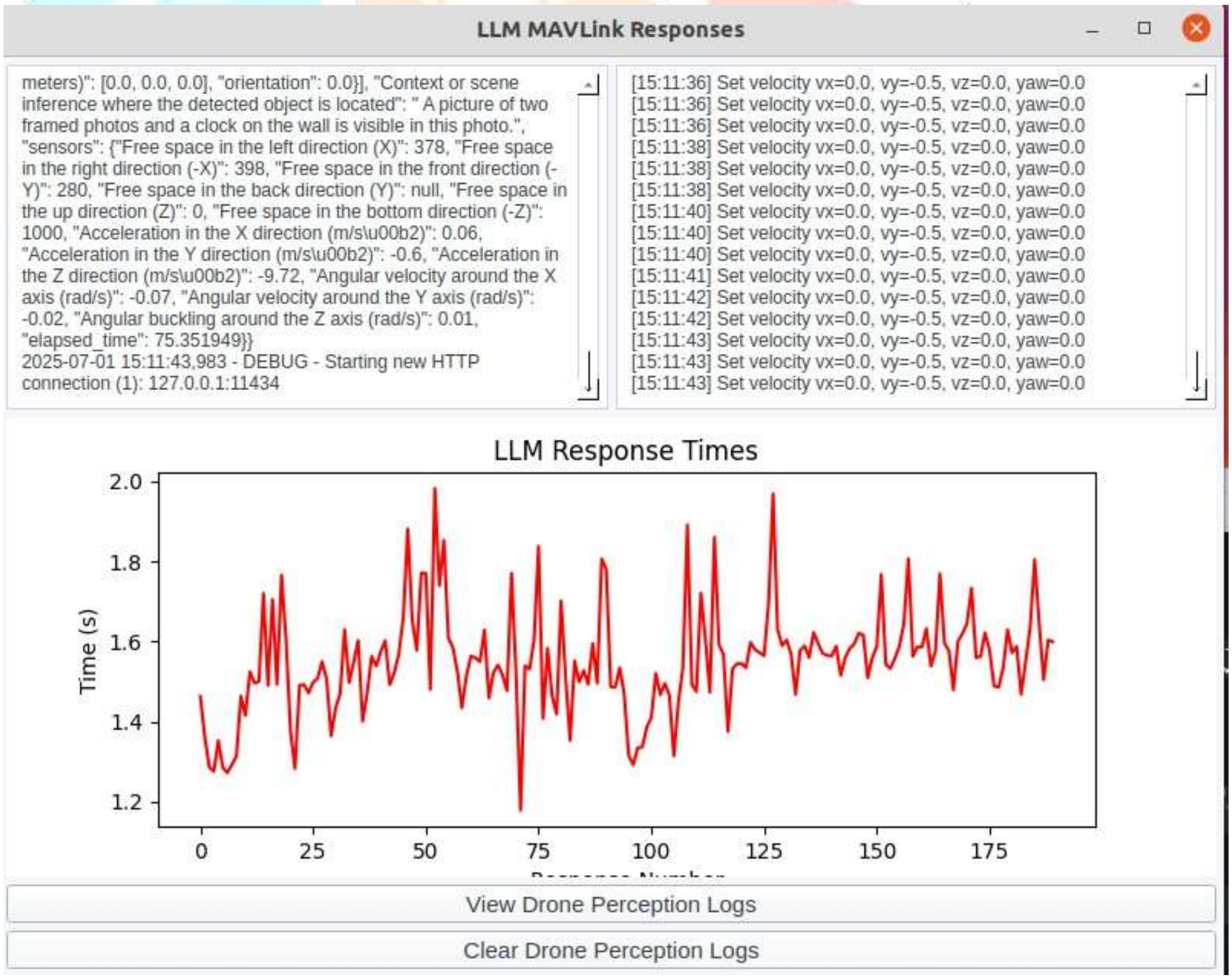
System Usage

CPU Usage: 13.9%

RAM Usage: 60.9%

GPU Usage: 0

Figure 4: Visualization of vision language model in action for the same frame for additional inference and high level scenic understanding.



```
meters)": [0.0, 0.0, 0.0], "orientation": 0.0}], "Context or scene
inference where the detected object is located": " A picture of two
framed photos and a clock on the wall is visible in this photo.",
"sensors": {"Free space in the left direction (X)": 378, "Free space
in the right direction (-X)": 398, "Free space in the front direction (-
Y)": 280, "Free space in the back direction (Y)": null, "Free space in
the up direction (Z)": 0, "Free space in the bottom direction (-Z)":
1000, "Acceleration in the X direction (m/s\u00b2)": 0.06,
"Acceleration in the Y direction (m/s\u00b2)": -0.6, "Acceleration in
the Z direction (m/s\u00b2)": -9.72, "Angular velocity around the X
axis (rad/s)": -0.07, "Angular velocity around the Y axis (rad/s)":
-0.02, "Angular buckling around the Z axis (rad/s)": 0.01,
"elapsed_time": 75.351949}}
2025-07-01 15:11:43,983 - DEBUG - Starting new HTTP
connection (1): 127.0.0.1:11434
```

```
[15:11:36] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:36] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:36] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:38] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:38] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:38] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:40] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:40] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:40] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:41] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:42] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:42] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:43] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:43] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
[15:11:43] Set velocity vx=0.0, vy=-0.5, vz=0.0, yaw=0.0
```

Figure 5: GUI visualization of real-time central LLM response and MAVLink command generation via orchestration.

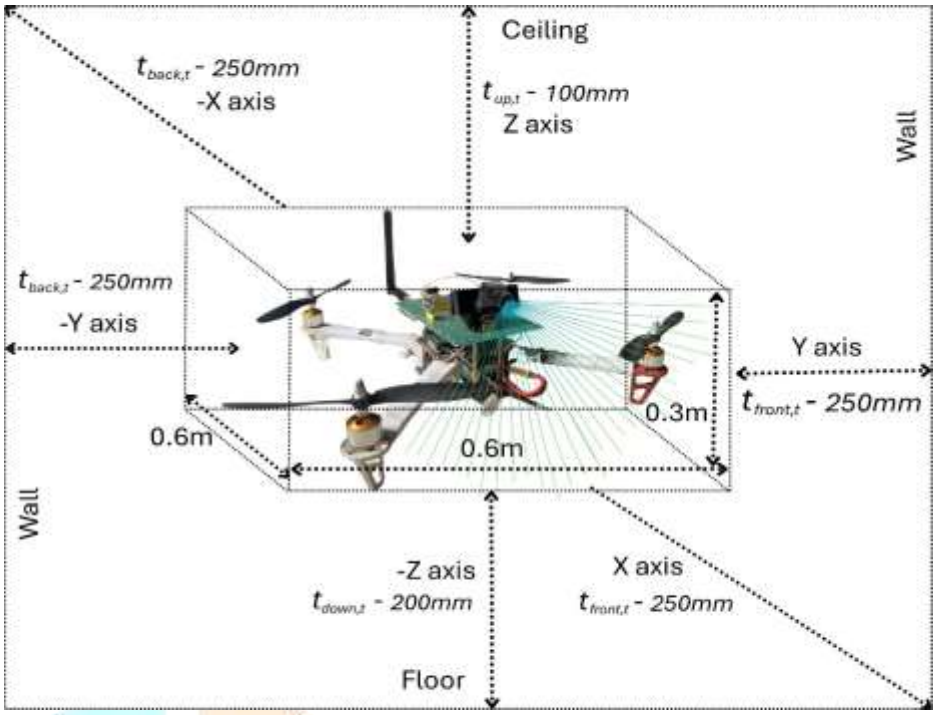


Figure 6: Protective shield via ToF sensor reading adjustments before feeding into the central LLM

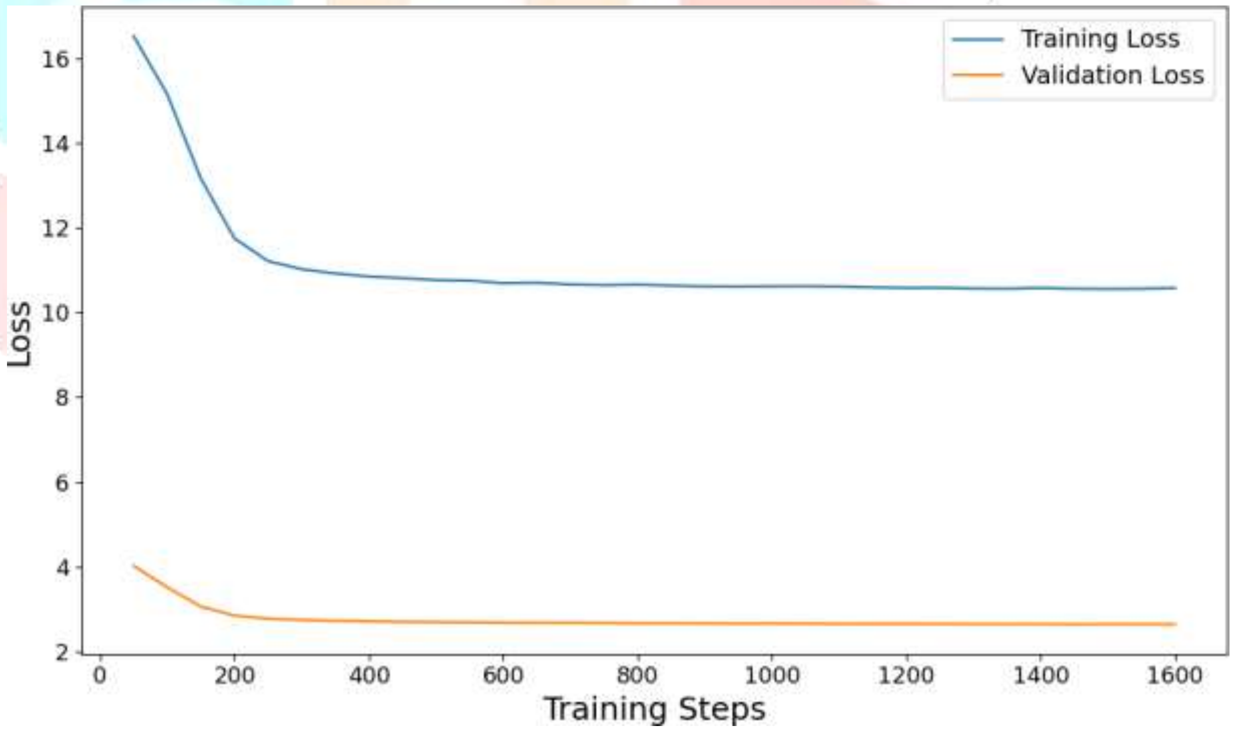


Figure 7: Training and validation loss trends for the fine-tuned SmolLM2-360M-Instruct model over 1600 steps, demonstrating robust convergence for drone navigation tasks.

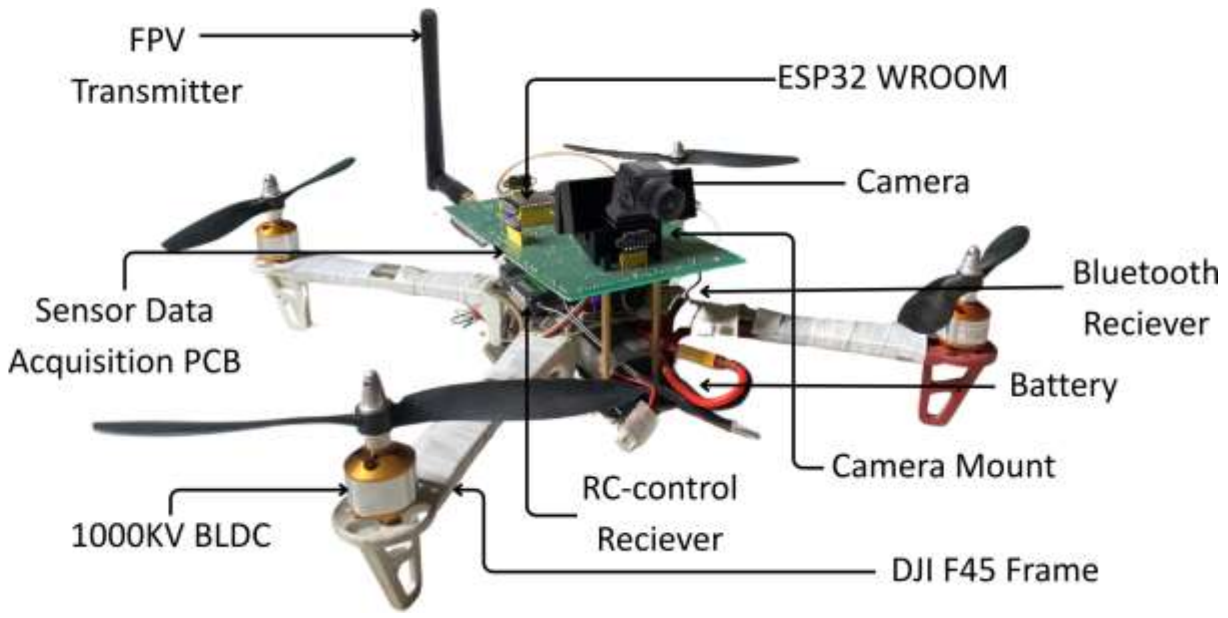


Figure 8: Customized quadcopter for this experiment.

References

[1] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile robot navigation using machine learning techniques," *Autonomous Robots*, vol. 13, no. 1, pp. 55–69, 2002.

[2] P. Corke, R. Paul, D. Rus, and G. S. Sukhatme, "Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2004, pp. 3602–3608.

[3] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2016.

[4] F. F. S. Silva, R. Oliveira, and R. A. F. Mini, "Deploying TinyML for autonomous drones: A case study on obstacle detection," in *Proc. Int. Conf. on Embedded Wireless Systems and Networks (EWSN)*, 2019, pp. 224–235.

[5] D. Saha, A. Pal, and R. Bose, "Edge computing for UAV navigation: A deep learning-based approach," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9875–9885, Jun. 2019.

[6] TensorFlow Lite for Microcontrollers. Accessed: Sep. 4, 2019. [Online].

[7] Espressif Systems, "ESP32 Technical Reference Manual," 2019. [Online].

[8] M. G. Jadidi, J. V. Miro, and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Autonomous Robots*, vol. 42, no. 2, pp. 273–290, 2018.

[9] S. M. LaValle, "Planning algorithms for autonomous navigation," *Cambridge University Press*, 2006.

[10] A. Dutta, S. Gupta, and S. Bhunia, "Lightweight AI models for edge devices: A survey," *ACM Computing Surveys*, vol. 55, no. 3, pp. 1–34, 2019.