IMPLEMENTATION OF A DRONE-BASED INFORMATION GATHERING SYSTEM

Tianrui Hu, Yifan Pan, Anshuman Dash, Matthew Tran,
Zhiwen Wu, Phil Tokumaru and Yogananda Isukapalli
University of California, Santa Barbara
Santa Barbara, CA 93106

ABSTRACT

This paper demonstrates the implementation of the Personal Information Gathering System (PIGS) for information gathering. This system is intended for several specially-designed drones, a mobile phone, a base station, and a router. Traditionally in combat situations, humans must risk themselves to gain information and identify potential threats. The PIGS system ensures users gain comprehensive information autonomously, while safe from threats. The operator can use the mobile device to remotely command the drones to obtain information, explore different regions, and perform other information-gathering-related tasks. With 802.11ac Wi-Fi and a lightweight computer vision model, PIGS allows the operator to interface with the drones through high-level commands and receive visual information with optional computer vision analysis. The proposed system offers a safer and more efficient way to gather information in dangerous environments.

INTRODUCTION

Navigating through intricate landscapes such as hills or cities often poses a challenge for humans when it comes to gathering relevant data and spotting potential hazards. Traditionally, information gathering would involve finding a high vantage point to obtain information, but this approach has its limitations. The introduction of portable drones has made information gathering easier, given their ability to be remotely operated by the user and capture visual data through onboard cameras. Drones can function in many environments, unaffected by restrictions of terrain or elevation. However, managing a drone typically necessitates the use of a specially designed controller and other hardware, which causes difficulty for the user. Moreover, a single drone may not be reliable in complex conditions, where strong winds or obstacles could cause interference or even destroy the drone. Consequently, a system that incorporates multiple drones and is controlled through abstract high-level commands holds significant potential in addressing these challenges.

To address this, our group proposes PIGS: a personal information gathering system. Through a network of two or more drones, a user is able to command them through various modes to scan the surrounding area, explore different areas, or perform other information-gathering-related tasks. Our group aims to provide a proof of concept for this tool, with a ground operator who interfaces with the drones through a mobile app, while the drones autonomously stream video back to the operator. User control is abstracted to high-level commands, while the drones are intelligently dispatched to tasks and maneuver accordingly. We also provide manual control for finer control for more specific flights.

PROJECT OVERVIEW

The P.I.G.S system encompasses hardware design, communication networking, and drone control logic, with careful consideration given to the selection of hardware and networking components that can effectively optimize system performance. The user can remotely control the drones using high-level commands, enabling the drones to operate in various modes, which can do tasks like orbit the user, explore a target location, and track and relay detected objects.

As presented in Fig.1, the hardware configuration comprises two drones equipped with several different components. Each drone utilizes a Cube Orange with RTK GPS capability which is connected to a Raspberry Pi as a communication proxy. The other devices consist of a phone running the P.I.G.S application, a laptop equipped with an RTK base station serving as the centralized processing for the drones and the phone, and a router that supports 2.4 GHz Wi-Fi. A communication of drone-laptop-phone is built into this network, where the user can operate the network of drones through their phone.

The network operates within a 2.4 GHz Wi-Fi environment, facilitating communication among drones, laptops, and phones through various protocols. These protocols include HTTPS, UDP, TCP, and a file-sharing system. Lightweight protocols like UDP are employed for drone communication to ensure fast communication and streaming for low-power devices. High-reliability protocols such as TCP are utilized for important messages such as high-level commands from the phone. The network forwards commands and user locations from phone to drone and vice versa.

To enable high-level abstract control of the two drone systems, multiple algorithms have been implemented. Internally, the drones utilize a state machine that facilitates transitions between different modes. Each state within the state machine corresponds to one or multiple modes. Various flight control and routing algorithms are implemented within these states to ensure the drones exhibit the desired behaviors safely and consistently.

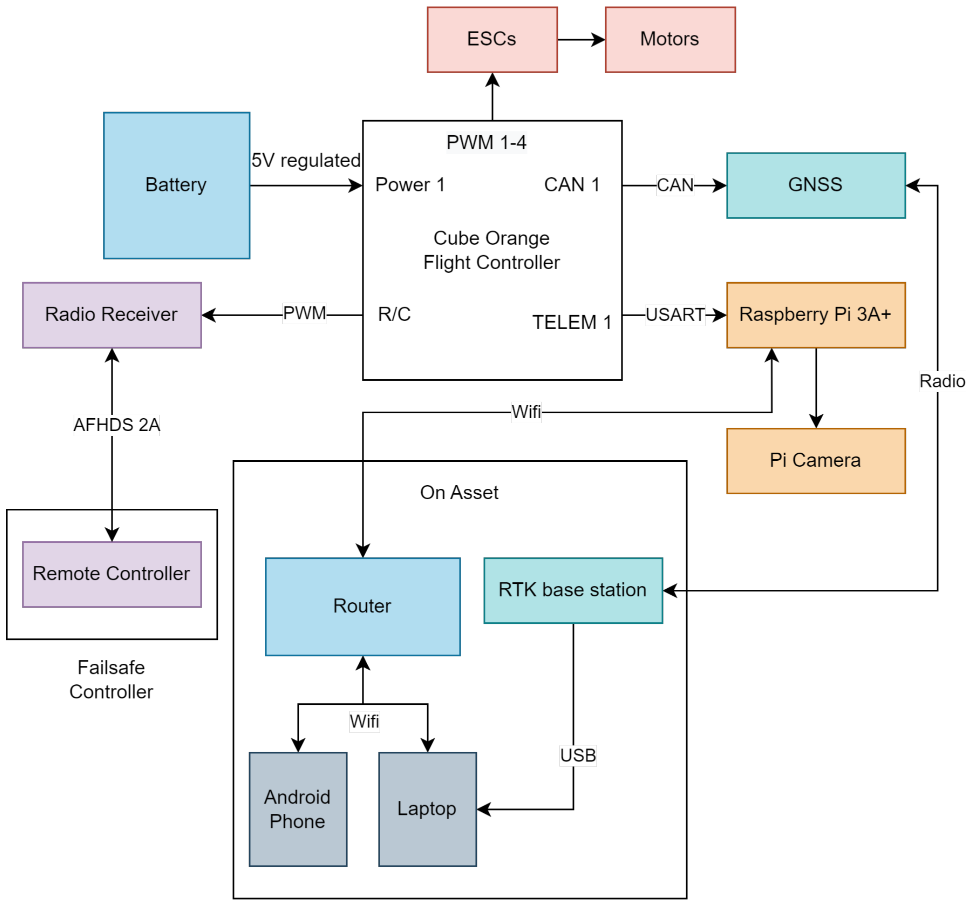

Figure 1: Hardware Block Diagram

DRONE DESIGN

Overview

Shown in Fig.2, the foundation of our drone relies on the robust S500 Quadcopter Drone Frame 2, a decision influenced by our need for a more substantial and steady platform to facilitate high-quality video capture. In order to ensure sufficient propulsion for flight, our drone is equipped with ReadyToSky 40A Electronic Speed Controllers (ESCs) which meticulously regulate the drone's motor speed. The motors are FlashHobby D3530 units with a 1100KV rating, providing a perfect balance of power, efficiency, and responsiveness. This system is further complemented by Master Airscrew 11x4.5 propellers, recognized for their efficiency and reduced noise output. A 6500mAh 4S Lithium Polymer (LiPo) battery is powering this sophisticated array of components. This choice of battery, renowned for its high energy density and lightweight characteristics, ensures that the drone can sustain approximately 15 minutes of uninterrupted flight time. This endurance allows our drone to execute various tasks reliably and consistently over a reasonable distance and time frame.

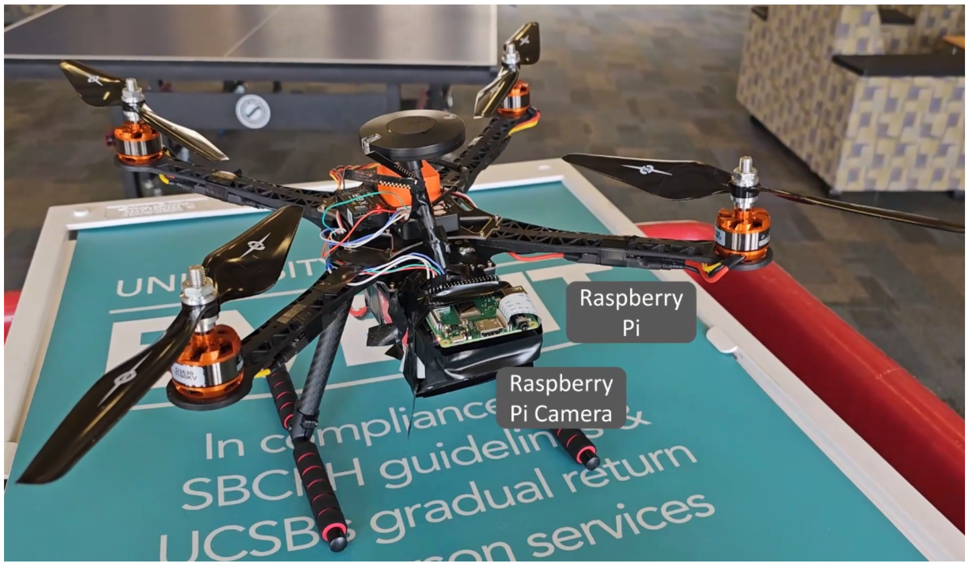

Figure 2: Image of the drone.

Cube Orange

Cube Orange is the central module for the control of the drone. It contains an STM32H753 processor and an STM32F103 failsafe co-processor which gathers and processes data from internal and external sensors. For the internal sensors, it has two vibration-isolated IMUs and an ICM 20649 integrated accelerometer to estimate the flying state. It also has an MS5611 barometer for altitude estimation. The Cube Orange also has various interfaces that are used to connect to the radio, GNSS, and Raspberry Pi 3A+. All of this data is processed on the Cube Orange using the ArduPilot autopilot software.

Here3 GNSS and Here+ RTK Base

Here3 GNSS is an accurate global navigation satellite system (GNSS) designed to optimize the positional accuracy of the drone. It integrates the STM32F302 processor and ICM20948 IMU sensor to measure the drone's specific force, angular rate, and orientation. It is capable of connecting to various satellite constellations such as GPS L1C/A, GLONASS L1OF, and BeiDou B1l to provide the latitude and longitude of the drone. Compatible with Here3 GNSS, the Here+ RTK GPS is a crucial element of the drone's positioning system designed to provide real-time kinematic corrections[1]. These corrections are sent to the drone through the drone's connection to the base station and increase the accuracy of the drone's position estimate.

Raspberry Pi 3 A+

The Raspberry Pi 3 Model A+ is the computer on the drone that is responsible for sending telemetry data to the ground station. It is a single-board computer that brings high-performance computing capability into a compact model. It contains a Cortex-A53 64-bit processor and 512MB LPDDR2 SDRAM, which provides reliable computational power when running multiple tasks simultaneously. The board also contains 802.11 b/g/n/ac wireless LAN which supports 2.4GHz and 5GHz WiFi connections.

Raspberry Pi Camera Module 2

We opted for the Raspberry Pi Camera Module 2 as our camera of choice, primarily because of its compatibility with the Raspberry Pi. The module features a Sony IMX219 sensor, which is capable of recording video at 1080p30 and 720p60. With its 8-megapixel sensor, this camera enabled us to capture high-quality video that was utilized for object identification through machine-learning techniques.

COMMANDS AND ALGORITHMS

The objective of this project was to develop a drone system that incorporates user-friendly control capabilities. Our approach enables users to issue high-level commands to the drone system directly from their smartphones, without the need to specify commands for each individual drone. These commands would then be received by a laptop, where they would be parsed into corresponding instructions for each drone. In order to facilitate various flying modes, smooth transitions, and the execution of the failsafe command at any given time, we designed a hierarchical state machine for the drone system as shown in Fig. 3

A. Command and Telemetry Flow

To ensure efficient data transmission, we divided the process into two stages: phone-to-laptop and laptop-to-drones. When the user interacts with the application on their phone, the phone transmits command data to the laptop including the command type, the current GPS location of the phone, and the target position. Subsequently, the laptop utilizes this information to update the state machines for both drones. The laptop then sends appropriate commands to the drones, accompanied by timestamps. These commands contain actions such as altering Ardupilot modes, initiating takeoff, directing the drones to specific locations, and modifying yaw speed. Not receiving a new timestamp for a given time, a failsafe mechanism prompts the drone to initiate an autonomous landing.

By implementing the state machine on the laptop, we effectively address the coordination of both drones during orbiting. This is achieved by ensuring that the target positions are situated at antipodal points along the trajectory, thereby minimizing the risk of collisions. Furthermore, this approach eliminates the need for one drone to have knowledge of the other drone's status, resulting in enhanced convenience and reduced communication latency.

At each clock cycle, the drone transmits its telemetry data, which includes latitude, longitude, yaw, altitude, and the current timestamp, to the laptop. If the incoming timestamp is determined to be outdated, the state machine will change the state of both drones to `disconnected` state to prevent incoming commands. Additionally, the laptop relays the received telemetry data, along with the corresponding drone identifier, to the phone. This information is then displayed on the

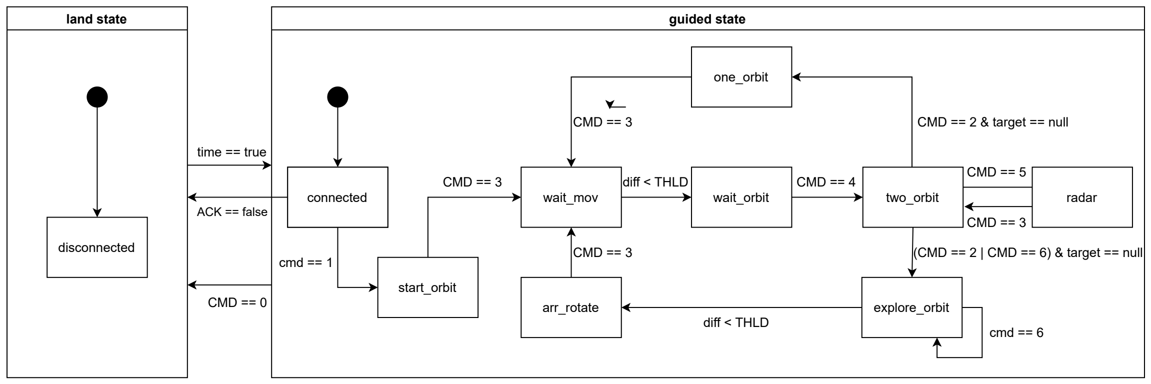

table of CMD	
1	land
2	explore mode
3	orbit mode
4	both drones ready
5	radar mode
6	bird-eye mode

Figure 3: The hierarchical state machine for the drone system implemented in the laptop. CMD represents the high-level command sent from the phone.

phone's map interface, providing real-time updates on the drone's position.

B. Hierarchical State Machine in Laptop

All the drone starts with the `disconnected` state on the ground in Ardupilot land mode. Once receiving a new timestamp from the drone, the connection to the Raspberry Pi is established, and it will change flying mode to Ardupilot-guided mode for future state transition commands. If one drone fails in pre-arm checks, it reverts to the `disconnected` state as a safety precaution.

In the `guided` state, we implemented different flying modes for the drone system. Once receiving the takeoff command, both drones will ascend to a preset height and enter a loiter mode, maintaining a given altitude. Then, the laptop will utilize the current location of the phone as the center point to generate a series of circular trajectories. Both drones will be dispatched to two antipodal points on the circular trajectory by issuing the orbit command. When the distance between the current position of a drone and its target position is smaller than a predefined threshold, the drone will transition into the `wait_orbit` state and awaits the arrival of the other drone.

After both drones enter the `wait_orbit` state, they will automatically transition to the `two_orbit` state, where the laptop will keep sending commands to direct the drones to the next point on the trajectory. If the user is moving, target locations will adjust to orbit around the user. From `two_orbit` state, the drone system can handle bird-eye, explore, and radar commands. Upon receiving an explore command, the laptop directs one drone closer to the destination while instructing the other drone to `one_orbit` state to keep orbiting. If an orbit command is received, the laptop guides both drones back to the antipodal points with the shortest distance, thereby going back to `two_orbit` state.

At any given time during the `guided` state, the user has the option to send a land command to both drones, prompting them to transition back to the `disconnected` state. This command serves as a failsafe, interrupting the current command and initiating a controlled landing. Which ensures safety in the event of emergencies.

COMMUNICATION PROTOCOL

As depicted in Fig.4, the selection of communication protocols is pivotal in the project's networking architecture, where the reliability of communication is paramount. Ensuring the secure delivery of commands is vital for drone safety, while also maintaining network efficiency and scalability. Given these prerequisites, several protocols have been adopted for communication between drones, laptops, and phones.

For communication between the drone and the laptop, the UDP protocol is employed. Throughout the flight, each drone onboard Raspberry Pi consistently transmits video footage and drone-related information to the laptop. Simultaneously, the laptop dispatches commands containing location data to the drone, also utilizing UDP. The protocol's resource efficiency and simplicity make it suitable for a low-energy Raspberry Pi. Moreover, in complex environments where the connection to distant drones may not be stable, the UDP protocol's capacity to endure package loss helps preserve essential communication[2].

In addition to the UDP protocol, a file-sharing system serves as an auxiliary communication method between the drone and the laptop. This system operates by continuously writing commands in a file located within the drone, allowing the drone to receive instructions from this file. While not as efficient as UDP, it does not necessitate an independent communication thread, making it a practical low-power solution for drone communication. This method proves beneficial in situations with low power availability or in unstable environments.

For communication between the phone and the laptop, which are usually in proximity to the user, the TCP protocol is utilized. As this protocol guarantees data packet delivery in the same sequence they were sent and includes error detection and correction mechanisms, it ensures data accuracy and precise location calculations.

Furthermore, for laptop-to-phone communication, an HTTPS server is employed. The laptop sets up a webpage containing the feed from each drone camera. The phone accesses this feed by visiting the web page's URL, thereby allowing the laptop to stream multiple feeds simultaneously and ensuring system scalability. This method also enables efficiency as the phone only communicates with relevant video feeds. The security of this protocol can be enhanced by implementing an access token for the HTTPS server.

USER INTERFACE

Mobile Phone Application

Users can interact with the drones through the mobile phone application called PIGSDrone. The application is designed for Android devices running on Android API Level 26 (Android 8.0), which is compatible with 93.44% of Android devices as of January 2023 [3]. However, in theory, this application design can also be implemented on other platforms that support touchscreen inter-

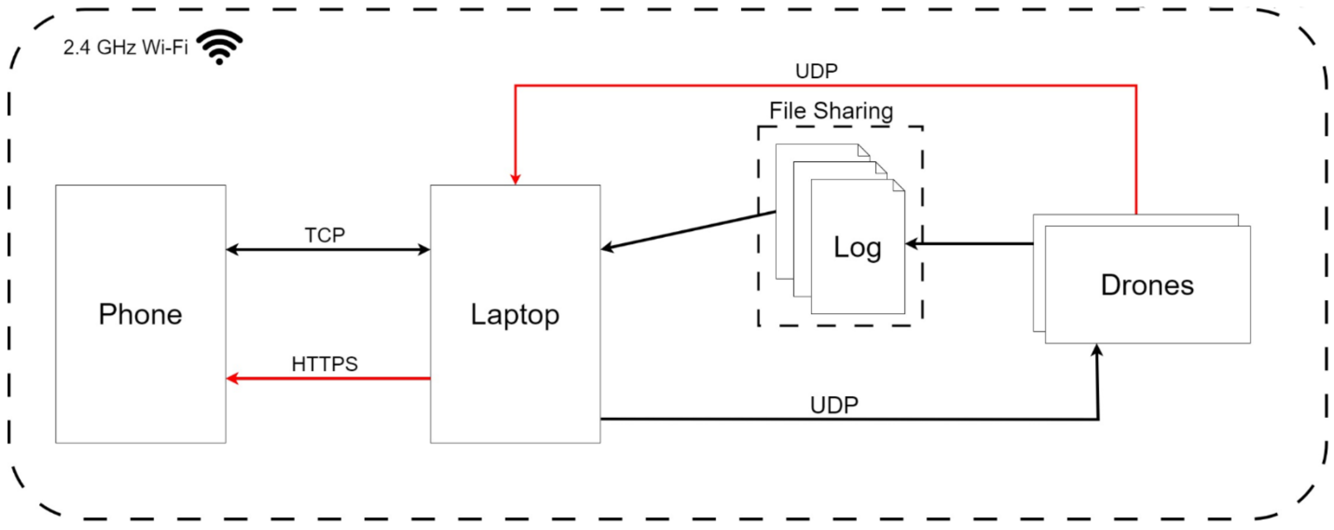

Figure 4: Block diagram for the communication and protocols we used in the network.

faces, such as iOS and Linux mobile.

PIGSDrone consists of a single screen with one Android Activity [4]5.a. This screen serves as a centralized hub for all information and commands, allowing for efficient interaction. The top half of the screen displays a map utilizing the Google Map API. It shows the locations of both the drones and the user's phone during the flight. Users can set a targeted location for the explore mode by pressing the desired location on the map. Additionally, buttons aligned to the left of the map enable users to focus on specific drones or their own location.

Below the map, videos from the drones' cameras are displayed. Each video is streamed to an HTTPS webpage by a laptop and accessed by the PIGSDrone application through a web view window. Only one video, corresponding to one drone's camera, can be accessed by the application at a time. Users have the ability to switch between the two cameras. If object detection is activated, the application overlays bounding boxes and identifies objects in the video stream, as shown in Fig.5.b. The location of the phone and the drones is shown beneath the video, providing supplementary information to the map display.

At the bottom of the screen, there are ten buttons for controlling the drones. These buttons include commands to switch the video stream, activate object detection, and control the movement of the drones. All commands can be executed with a single click, eliminating the need for manual controls, which are not included in the application.

Object Detection

The application of object detection through computer vision significantly enhances the functionality and operational efficiency of drone scouting, granting it the capability to execute tasks autonomously and safely. PIGSDrone employs a Tensorflow-based YOLOv5-Lite model [5], a

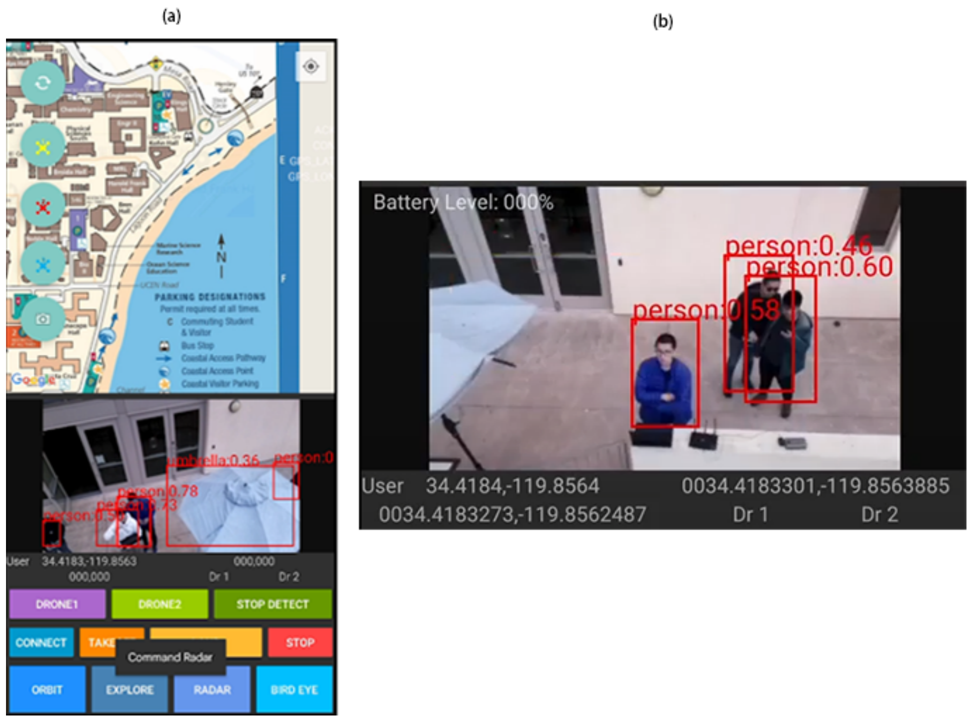

Figure 5: Screenshots of PIGSDrone Mobile Application on Pixel 6. We show the overall layout of the app, and the video dispalyed with object detection in the radar mode(b).

lean yet powerful deep learning construct capable of performing real-time object detection across 80 classes on a mobile device. Upon activation by a user, the model identifies the object of interest and annotates the video feed in real-time with bounding boxes.

Testing has shown that the model delivers satisfactory performance on 720p 30Hz video without substantial delay when deployed on Google's Pixel 6 with the Google Tensor platform. Even for drone footage of lower resolution at 360p 15Hz, devices with relatively lower computational power should be able to execute the model effectively.

Furthermore, the PIGS Drone system offers flexibility in terms of model selection. The weights of the YOLOv5-Lite model can be replaced with those of other models as per user requirements. Models such as the YOLOv5-small have also undergone testing. Users also have the opportunity to implement specially designed Convolutional Neural Network (CNN) layers and custom datasets into their models and integrate them with PIGSDrone, thereby enhancing the extensibility of the system. See [6] for a demo video of the system.

CONCLUSION

This paper discussed the design choices of PIGS with the intention of a simple to use, and robust system for information relay. In our approach to this problem, we utilize several different communication protocols at various different stages of our project, along with various different techniques for software flow control. Our project provides an open-source implementation of our solution to the original motivation, along with the reasoning for our design choices, both system and network. There is still room for improvement through potentially more processing power, to allow for more robust and efficient communication between components of the system, to allow for more fault tolerance and versatility.

Our solution delivers a network of drones controllable from a high level for the user, along with various different methods of retrieving the data for the user. This is a relatively inexpensive system, with the addition of being highly configurable and extensible for future work. This future work can entail new modes, adding new sensors for more generalized object avoidance, along with more telemetry measurements for task-specific implementation.

ACKNOWLEDGMENTS

We would like to thank AeroVironment for sponsoring this project and Tiziano Fiorenzani for his guidance.

REFERENCES

[1] J. W. Yanming Feng, "Gps rtk performance characteristics and analysis," *Journal of Global Positioning Systems*, vol. 7, no. 1, pp. 1–8, 2008.

[2] "Pigs laptop backend." `https://github.com/yifanpan2023/drone_command`.

[3] P. Taylor, "Mobile android operating system market share by version worldwide from january 2018 to january 2023," 2023.

[4] "Pigs drone android application." `https://github.com/Hudayday/PIGSDrone`.

[5] J. Wang, Y. Chen, Z. Dong, and M. Gao, "Improved yolov5 network for real-time multi-scale traffic sign detection," *Neural Computing and Applications*, vol. 35, no. 10, pp. 7853–7865, 2023.

[6] "Pigs project demo." `https://www.youtube.com/watch?v=dZNtxEBDu8k`.