LOW-COST UAV SWARM FOR REAL-TIME OBJECT DETECTION APPLICATIONS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Joel Valdovinos Miranda

June 2022

COMMITTEE MEMBERSHIP

TITLE: Low-Cost UAV Swarm for Real-Time Object Detection Applications

AUTHOR: Joel Valdovinos Miranda

DATE SUBMITTED: June 2022

COMMITTEE CHAIR: Franz Kurfess, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Maria Pantoja, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Lynne Slivovsky, Ph.D.
Professor of Computer Engineering

ABSTRACT

Low-Cost UAV Swarm for Real-Time Object Detection Applications

Joel Valdovinos Miranda

With unmanned aerial vehicles (UAVs), also known as drones, becoming readily available and affordable, applications for these devices have grown immensely. One type of application is the use of drones to fly over large areas and detect desired entities. For example, a swarm of drones could detect marine creatures near the surface of the ocean and provide users the location and type of animal found. However, even with the reduction in cost of drone technology, such applications result costly due to the use of custom hardware with built-in advanced capabilities. Therefore, the focus of this thesis is to compile an easily customizable, low-cost drone design with the necessary hardware for autonomous behavior, swarm coordination, and on-board object detection capabilities. Additionally, this thesis outlines the necessary network architecture to handle the interconnection and bandwidth requirements of the drone swarm.

The drone on-board system uses a PixHawk 4 flight controller to handle flight mechanics, a Raspberry Pi 4 as a companion computer for general-purpose computing power, and a NVIDIA Jetson Nano Developer Kit to perform object detection in real-time. The implemented network follows the 802.11s standard for multi-hop communications with the HWMP routing protocol. This topology allows drones to forward packets through the network, significantly extending the flight range of the swarm. Our experiments show that the selected hardware and implemented network can provide direct point-to-point communications at a range of up to 1000 feet, with extended range possible through message forwarding. The network also provides sufficient bandwidth for bandwidth intensive data such as live video streams. With an

expected flight time of about 17 minutes, the proposed design offers a low-cost drone swarm solution for mid-range aerial surveillance applications.

ACKNOWLEDGMENTS

Thanks to:

- Dr. Kurfess for his guidance throughout this project.
- My family for their support. Their sacrifices are what allowed me to pursue a higher education and gave me the motivation to try my best.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The cost of drone technology and powerful embedded computers has continued decreasing over the years, enabling the exploration of innovative applications for UAVs, specifically multirotor drones. Besides being great for recreational purposes, drone swarms have multiple professional applications such as aerial surveillance, package delivery, search and rescue, etc [8]. One of these applications was demonstrated in Australia where swarms of drones with object detection capabilities are being used to monitor beaches and detect sharks near the shore [9]. While this is a very helpful application that increases public safety, the professional level drones utilized cost thousands of dollars. The high cost of advanced drones imposes a barrier of entry for university research groups to continue exploring drone swarm technology. Therefore, an affordable drone swarm solution is necessary.

Affordable drones have many limitations compared to professional drones such as reduced payload weight, flight time, communication range, and on-board processing power. These limitations make it necessary to find innovative solutions to achieve proper drone swarm behavior, and demand research to be done on drone hardware and communication networks. The goal of this thesis is to compile the specifications for a low-cost drone capable of performing autonomous flight, swarm coordination, and real-time object detection using readily available components. Also, to implement a reliable communication network with sufficient range and bandwidth to achieve swarm coordination, collection of live video feeds, and collection of real-time object detection data.

The design of this paper's drone swarm platform was inspired by the needs of Cal Poly's collaboration with CSU Long Beach's Shark Lab to track sharks and other marine creatures and study their behavior. While the target application is shark spotting and the drone hardware was selected to fulfill the requirements of this application, significant attention was given to making the design affordable and highly customizable. This allows for student research groups to build on top of the suggested platform and continue developing novel functionality for the drone swarm. Additionally, the design considerations of the proposed solution are thoroughly described, providing enough information for researchers to understand and customize the platform for their specific applications. This paper may also serve as an introduction to custom drone builds and drone swarm operation for researchers interested in beginning work in the field of intelligent UAVs or in need of a low-cost drone swarm.

Chapter 2

BACKGROUND

2.1 Ground Control Station

In UAV applications, the Ground Control Station (GCS) is the computer used by the drone operators on the ground to interact with the drone's on-board system. The GCS software can establish preplanned flight paths, enable failsafe measures, receive status data from the drones, send flight commands, etc. Some popular GCS softwares are Mission Planner [10], QGroundControl [11], and MAVProxy [12]. While all three GCS softwares support simultaneous communication with multiple drones, QGroundControl arguably provides the most user friendly interface. Therefore, QGroundControl was selected for initial testing of our drone swarm.

2.2 Swarm Network

The communication network is an essential aspect of a drone swarm. The network allows the GCS to communicate with the drones and exchange different types of data such as telemetry data (location, speed, battery status, etc), commands to alter flight paths, video streams, and any other application specific data. In drone swarms, the network is also responsible for transmitting messages between drones to achieve swarm behavior. Drone-to-drone messages may include status information to keep track of each other, updates on shared data, alert messages broadcasted from one drone to the rest of the swarm, or any other information meant to affect the behavior

of the swarm. Different network topologies with different communication technologies are possible, which will be discussed in later sections.

2.3 Real-Time Object Detection

Object detection is an application of machine learning in which a machine learning model is used to identify and locate objects of interest within an image. The model is trained to identify objects from a specified set of classes (such as person, dog, car, chair, etc.) and locate the objects within the image using bounding boxes. This is shown below in Figure 2.1.

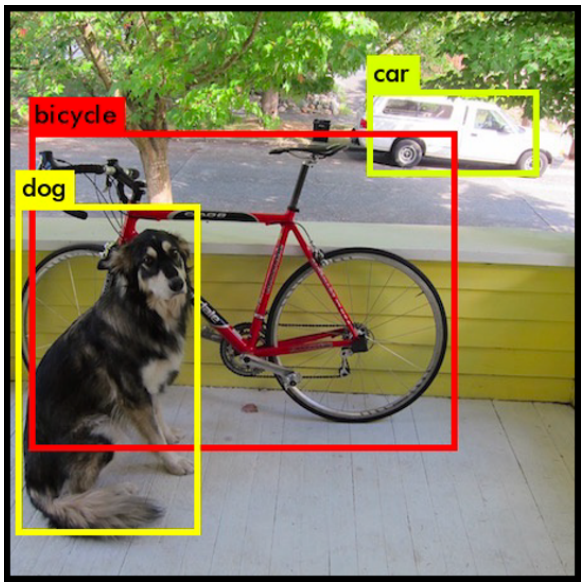

Figure 2.1: Output of an object detection model. The image shows bounding boxes enclosing the detected objects: a dog, a bicycle and a car [4]

There are two common categories for object detection models: two-shot and single-shot detectors. At a high-level, two-shot detectors require two stages to achieve objection detection. The first stage performs "region proposal", in which areas of interest with high potential for finding objects are selected. These proposed regions are then passed to the second stage of the detector, where it is determined whether

or not an object is found in the selected region. Single shot detectors skip region proposal. Instead, the input image is processed as a grid and each cell is processed to determine if an object of interest is found within the cell [13].

Choosing between single-shot and two-shot detectors leads to a trade-off between speed and accuracy. Single-shot detectors require less computational power than two-shot detectors, providing better inference speed at the expense of detection accuracy. On the other hand, the two stage architecture of two-shot detectors provides better detection accuracy with slower inference speeds. The added complexity and processing needs of some two-shot detectors render them unusable in systems with limited resources, making single-shot detectors a popular approach for real-time applications with embedded devices. This trade-off between speed and accuracy must be given sufficient consideration when deciding which model to use for a specific application. While selection and implementation of a machine learning model is out of the scope of this thesis, the proposed hardware aims to provide sufficient flexibility to explore different options.

The term "real-time" may have different meanings in different applications. In this paper, we describe object detection as "real-time" to denote that processing of images or video is handled by the drone's on-board hardware with reasonable inference speed. Performing object detection on recorded video post-flight is not real-time.

2.4 Video Streaming

To stream video from the drones to the ground control station over the network, two transport layer protocols can be used: the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP). TCP is a more reliable protocol as it can detect when a transmitted packet has been lost and attempt to recover it. However, with

this packet recovery feature, TCP is susceptible to video delays. If the network is unstable, many packets may be lost and TCP will not send the newest video frames to the GCS until the lost video frames are recovered. TCP may be a good approach for other types of data that require reliable delivery such as flight commands. UDP is a protocol better suited for low-latency applications like video streaming because it does not attempt to recover lost packets. If a packet is lost, UDP simply ignores the loss and continues sending the newest video frames. The receiver of the video stream might receive corrupted frames or completely miss frames occasionally, but overall the video will have less delay.

On top of the TCP or UDP protocols, the Real-Time Streaming Protocol (RTSP) can be used to control a swarm's video streams. This protocol allows for the use of a server-client model, where the server offers a video stream and the client may pause or resume the video stream. In our drone fleet application, each drone may be configured as a server that is controlled by the GCS. This way, the only video streams taking up bandwidth in the network are the ones activated by the GCS. Without this protocol, all drones would continuously stream their video and saturate the network.

Chapter 3

RELATED WORK

3.1 Previous Work in Drone Networks

The paper *Communication Architectures and Protocols for Networking Unmanned Aerial Vehicles* provides an overview of four types of drone swarm network architectures: Centralized Communications, UAV Ad Hoc Network, Multi-Group UAV Network, and Multi-Layer UAV network [5]. The Centralized Communications architecture relies on a central node that handles all communications through the network. Nodes are not allowed to send messages directly to each other without routing their communications through the central node. The other three network architectures are considered "decentralized architectures," in which a central node is not required as nodes are allowed to communicate directly to each other and forward messages through the network. Each of the three decentralized architectures are differentiated by their use of "backbone UAVs," which typically carry higher-quality transceivers to communicate with the ground control station at longer distances and serve as a gateway for the UAV swarm. These architectures will be further discussed later in this paper.

On top of the underlying network architecture, a routing protocol is necessary to decide how messages travel through the network. The paper *Brief Analysis of Drone Swarms Communication* addresses different routing techniques that can be used on drone ad hoc mesh networks [14]. The paper offers two approaches for guiding messages through mesh networks: routing and flooding. Each approach has pros and cons that will be explored later in this paper.

There are a variety of routing-based protocols that have been considered for Flying Ad-Hoc Network (FANET) purposes. The paper *A Comparative Performance Evaluation of Routing Protocols for Flying Ad-Hoc Networks in Real Conditions* provides a comprehensive overview of different routing protocols suitable for UAV swarm applications and compares the performance of three different proactive routing protocols in real-world conditions, as opposed to providing results from simulated network tests [15]. The paper outlines five categories for routing protocol: Topology-Based Routing, Position-Based Routing, Clustering/Hierarchical Routing, Swarm-Based Routing, and Delay-Tolerant Network Routing. Topology-Based Routing protocols rely on information about links between nodes to choose the best path to reach another node. Position-Based Routing protocols use the geographical position of nodes to choose the best path. Clustering/Hierarchical Routing protocols rely on UAVs moving as a cluster and establish a hierarchy for message routing, with a cluster head that serves as a gateway for each cluster. To improve network reliability, these routing protocols attempt to predict swarm formation through UAV mobility patterns. Swarm-Based Routing protocols attempt to imitate the behavior of animals such as bees and ants. Typically, the imitated behavior consists of sending scouting messages through the network that eventually return to the source and provide an optimal path for the rest of the data [16]. Lastly, Delay-Tolerant Network Routing protocols in which nodes that become disconnected from the network store outgoing packets in a buffer until they regain connection to another node, at which point the buffered packets are transmitted. Topology-Based routing protocols are the most popular as there is a plethora of possible implementations and plenty of papers comparing their performance for varying applications. This category will be further explored later in this paper.

3.2 Previous Work in Drone Coordination

The paper *Extending QGroundControl for Automated Mission Planning of UAVs* addresses mission planning for drone swarms by extending QGroundControl to accept swarm configuration and target goals as inputs to automatically generate possible flight plans that achieve the target goals [17]. Additionally, the extension described in the paper enables mid-flight mission re-planning to adapt to new conditions, such as when a sensor fails or an area of interest is detected. This allows pilots to efficiently coordinate drone actions without the need for intelligent agents with decision-making capabilities mounted on the UAVs. The paper *Surveying Areas in Developing Regions Through Context Aware Drone Mobility* implements an object detection application [18]. In this paper, the researchers use a fixed-wing drone equipped with object detection hardware to fly over developing regions to detect people's settlements and livestock. The UAV starts with a preplanned flight path generated to cover a selected area. This path is marked by waypoints that the UAV has to follow. If in-between waypoints the drone detects an object of interest, such as a settlement or livestock, the drone will autonomously alter its path to circle around the object of interest and gather data. Once the UAV has recorded enough data, it will continue following the waypoints in the original flight plan. The approach followed in this paper could be transferred to our target applications. We could assign preplanned flight paths to each of the drones in our fleet, hover shortly over detected objects of interest, and then return to their original flight paths. The paper *Fleets of drones could aid searches for lost hikers* describes an application where quadcopter drones were used to autonomously search an area and report back to the base station when the object is found [19]. The target objects were lost hikers. In this application the drones do not begin with a preplanned flight path; instead, the drones generate their flight decisions as they search the area. An interesting strategy described in this paper is

the use of the drone's current direction to decide where to go next. By generating paths that avoid abrupt changes in direction, drones can sweep an area much faster. This paper provides a perfect example of an object detection application with drone swarms, and our low-cost swarm platform should be capable of following the paper's coordination strategies.

An important area in drone swarm research is the effect of different mobility models on network reliability. Two popular mobility models for area surveying applications are the Pheromone Repel Model and the Semi-Random Circular Movement Model.

In [20], the authors describe the Distributed Pheromone Repel Mobility Model, which can be used for reconnaissance missions to cover an area as quickly as possible. This is achieved by representing the area to be covered as a grid where each cell contains a timestamp of the last time it was visited by any drone. This grid is referred to as a "pheromone map". Each drone stores a local version of the pheromone map in memory and regularly broadcasts the locally stored map to drones that are within communication range, which then merge the incoming and the local map together. Individual drones make decisions as to what direction to fly every few seconds based on their current version of the pheromone map. The direction with the "weakest smell" (has not been visited recently) has the highest probability to be chosen by the drone. The paper notes that the pheromone model logic tends to push away UAVs from each other to maximize area coverage, which results in low network reliability. This leads the authors to the conclusion that "coverage and connectivity of communication are two conflicting objectives."

In [21], the Semi-Random Circular Movement (SRCM) model is outlined. The basic idea behind SRCM is that drones in a swarm can randomly select a circular path from a set of concentric circles centered at some target location, typically the GCS. The drone will then follow the path along the selected circle, with brief pauses at areas

of interest, until the entire circle is covered, at which point a new circle is selected and the process restarts. This approach maximizes area coverage, keeps drones out of each other's flight paths, and provides steady network connectivity based on swarm network simulations. Additionally, SRCM minimizes network bandwidth usage compared to other mobility models as location information is only shared between drones when a new circular path is being selected, as there is no risk of collision while drones are within their circular paths.

The two mobility models described above both require inter-drone communication and on-board processing for making autonomous decisions based on shared data. This illustrates the networking and processing requirements our swarm platform should be capable of handling.

3.3 Previous Work in Real-Time Object Detection on UAVs

In the paper *Deep Learning-Based Real-Time Multiple-Object Detection and Tracking from Aerial Imagery via a Flying Robot with GPU-Based Embedded Devices*, the authors provide an overview of different approaches to detect and track objects using drones with different hardware capabilities and different object detection algorithms [22]. The authors consider the cases of having an on-board GPU system, an off-board GPU-based ground station, and an onboard GPU-constrained device with an Intel Movidius Neural Compute Stick (NCS) attached. The authors also provide an algorithm for tracking objects using the onboard object detection capabilities. The experiments conducted demonstrate that a NCS is a viable solution if the drone design needs to minimize total cost and has limited space. An on-board GPU system, such as a Jetson Nano, can offer much better accuracy and lower latency. The off-board GPU-based ground station solution offered the best results in terms of accuracy and

latency. However, this solution is only viable for single drone flights, as processing video streams from a fleet would require a very large network bandwidth to transfer all video streams simultaneously. Also, one GPU may not be enough to handle all the streams simultaneously.

Chapter 4

SYSTEM DESIGN

Given this thesis focuses on the implementation of a low-cost drone fleet, significant attention was given to designing and implementing the drone hardware and the fleet network. The design of these two aspects will serve as the foundation for future drone swarm application research. The current target application is real-time shark detection, which means that the hardware should be capable of autonomously controlling the flight of a quadcopter drone swarm, performing real-time object detection, and transmitting all necessary data from the drones to the GCS and vice versa. While these capabilities are vital for our shark spotting application, they are also useful for other aerial surveillance applications.

4.1 Swarm Network

4.1.1 Bandwidth Requirements

One of the goals of the network in object detection applications is to transmit videos from cameras mounted on the drones back to the GCS for researchers to analyze. This way, if an object is detected, the researchers can immediately observe the object and modify the swarm behavior if necessary. Therefore, the swarm network should be capable of handling the transmission of video streams. Based on online video stream bandwidth calculators, a 720p 10FPS video stream compressed with medium h.264 compression requires 0.6Mbps of network bandwidth to be transferred reliably [23].

We will use this number as our baseline for the minimum bandwidth required between any drone and the GCS at any time to transmit video streams.

At bandwidths below 0.6Mbps, a video stream might not be reliable but flight data can still be delivered through the network. The flight controller used in this project uses a default baud rate of 56700 for transmitting telemetry data, meaning that a network bandwidth of 56Kbps is enough for controlling one drone. In fact, the baud rate of the flight controller for telemetry data may be reduced as low as 9600; however, we will consider 56Kbps our minimum network bandwidth per drone to ensure reliable communications.

For each additional drone in the network, an additional 56Kbps of network bandwidth at the GCS is required, since the new drone will require its own stream of data. We can estimate the required bandwidth at the GCS as *n x 56Kbps*, where n is the number of drones in the swarm. Any given drone, D, should maintain a throughput to the GCS of *56Kbps + (n x 56Kbps)*, where the initial 56Kbps is the bandwidth needed for drone D's flight data and the additional *n x 56Kbps* is the bandwidth required to forward messages to n other drones whose only link to the GCS is through drone D. This formula only takes into account the minimum bandwidth for flight control data. If a video stream needs to be transmitted, then the minimum throughput to the GCS of all drones that are used as hops by the video stream to reach the GCS increases by 0.6Mbps, which we established as the required throughput for 720p 10FPS video streams.

Taking these network bandwidth requirements into account, a wireless communication technology that supports the necessary bandwidth at reasonable range must be selected.

4.1.2 Wireless Communication Technologies

To meet the networking requirements of a drone swarm, an appropriate wireless communication technology is necessary. The wireless technology must support sufficient bandwidth to handle multiple data streams from the entire swarm and handle live video streams. At the same time, the protocol should provide reasonable outdoor range to maximize area coverage by the swarm. Table 4.1 below compares the capabilities of different popular wireless communication technologies. Licensed spectrum technologies are out of the reach of this project as the goal is to minimize swarm cost, and gaining access to these licensed spectrum technologies requires purchasing the access from service providers. This leaves ZigBee, Bluetooth, and Wi-Fi to be considered. ZigBee has a supported data rate of 250Kbps, which is less than the 0.6Mbps required to transmit a live video stream from the swarm to the GCS. Bluetooth has a supported data rate of up to 2Mbps, which could be sufficient for a video stream and the flight data of a drone swarm. However, 2Mbps is the maximum data rate possible under ideal conditions. This leaves us with a very small margin between required throughput and available bandwidth. In real world applications such as drone swarms, it is unlikely that the entire 2Mbps of bandwidth will be available. Therefore, we are left with Wi-Fi as a viable wireless communication technology for drone swarms. The 802.11n IEEE standard defines a Wi-Fi technology capable of data rates of up to 600Mbps and an outdoor range of up to 250 meters. Additionally, Wi-Fi supports mesh network topologies, which can extend the range of the network even more by forwarding messages through intermediate nodes to reach their destination. This shows Wi-Fi technology to be a good solution to the networking needs of a low-cost drone swarm. The ample available bandwidth of Wi-Fi can handle video streams and flight data transmission even in very large swarms, while the point-to-

point communication range paired with a mesh network topology can provide enough range for short to mid-range missions.

Table 4.1: Comparison between different wireless communication technologies [1]

Communication Technology	IEEE Standard	Frequency/Medium	Spectrum Type	Device Mobility	Theoretical Data Rate	Range Indoor-Outdoor	Network Typology	Latency	Advantages	Limitations
Wi-Fi [7,8]	802.11	2.4 GHz IR	Unlicensed	Yes	Up to 2 Mbps	20 m–100 m	Ad-hoc, star, mesh, hybrid	<5 ms	High speed and cheap	Limited range
	802.11a	5 GHz	Unlicensed	Yes	Up to 54 Mbps	35 m–120 m	Ad-hoc, star, mesh, hybrid			
	802.11b	2.4 GHz	Unlicensed	Yes	Up to 11 Mbps	35 m–140 m	Ad-hoc, star, mesh, hybrid			
	802.11n	2.4/5 GHz	Unlicensed	Yes	Up to 600 Mbps	70 m–250 m	Ad-hoc, star, mesh, hybrid			
	802.11g	2.4 GHz	Unlicensed	Yes	Up to 54 Mbps	38 m–140 m	Ad-hoc, star, mesh, hybrid			
	802.11ac	5 GHz	Unlicensed	Yes	Up to 3466 Mbps	35 m–120 m	Ad-hoc, star, mesh, hybrid			
Bluetooth 5 [9–12]	802.15.1	2.4 GHz	Unlicensed	Yes	Up to 2 Mbps	40 m–200 m	Ad-hoc, piconet	3 ms	Energy-efficient	Low data rate
ZigBee [13–15]	802.15.4	2.4 GHz	Unlicensed	Yes	250 Kbps	10 m–100 m	Ad-hoc, star, mesh, tree, cluster	15 ms	Low cost	Low data rate
WiMAX [16–18]	802.16a	2 to 11 GHz	Licensed	Yes	Up to 75 Mbps	Up to 48 km	Wide-area wireless backhaul	30 ms	High throughput	Interference issues
LTE [19–22]	LTE	Up to 20 MHz	Licensed	Yes	Up to 300 Mbps	Up to 100 km	Flat, IP based	5 ms	High bandwidth	Expensive
5G [23–29]	5G (eMBB)	28 GHz	Licensed	Yes	Up to 20 Gbps	Wide Area	IP based	1 ms	High data rate	Expensive
Satellite [30,31]	Satellite	Up to 40 GHz	Licensed	Yes	Up to 1 Gbps	World Wide	-	550 ms	Wide coverage	High delay and high cost

4.1.3 Network Topology

With Wi-Fi selected as our desired wireless communication technology, it is worth exploring the possible network topologies for drone swarms. Flying Ad Hoc Networks (FANETs) are networks intended for UAV swarms and can be divided into four architectures belonging to two categories [5]. The two categories are centralized architectures, or infrastructure-based architectures, and decentralized architectures. Decentralized architectures are divided into three architectures we will discuss later in this section.

The first architecture to discuss is the "centralized communications" architecture shown in Figure 4.1. This architecture consists of a central node (the ground control station) that is connected to all drones in the swarm and handles all communications through the network. When messages are sent from one drone to another, the messages must be routed through the GCS to reach the destination drone. This approach offers the advantage of having short delays between the ground control station and

any drone as they are directly connected. However, any drone-to-drone communication will suffer a longer delay as messages must be routed through the ground control station. Additionally, the centralized communications architecture may suffer from limited flight range, as any given drone in the swarm may only travel as far as its radio transmission system can maintain connection with the ground control station.

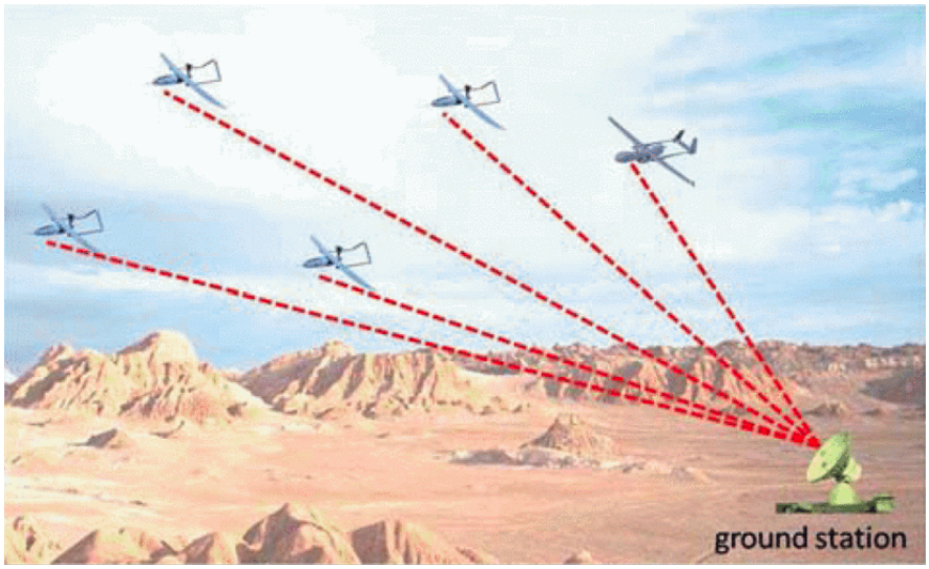

Figure 4.1: Centralized Communications Network Architecture [5]

The next three architectures are part of the "decentralized architectures" category. These architectures do not rely on a central node and messages between drones may be routed directly through the drones without using the ground control station. These architectures are also known as "mesh network" architectures.

The first type of decentralized architecture is the "UAV Ad Hoc Network" architecture, shown in Figure 4.2. In this architecture, all drones are responsible for forwarding messages through the network. [5] suggests using one drone as a "backbone UAV," which serves as a gateway between the ground control station and the rest of the drones in the swarm. The backbone UAV carries two types of transceivers, one for long-range communications with the ground control station and the other for short-range communications with the rest of the swarm. This allows for wide network

coverage as only the backbone UAV must remain connected to the ground control station at all times, while the rest of the swarm only needs to remain connected to the backbone UAV. This architecture works best for swarms of similar drones with similar speed and mobility patterns, as the fleet must remain relatively close to each other at all times.

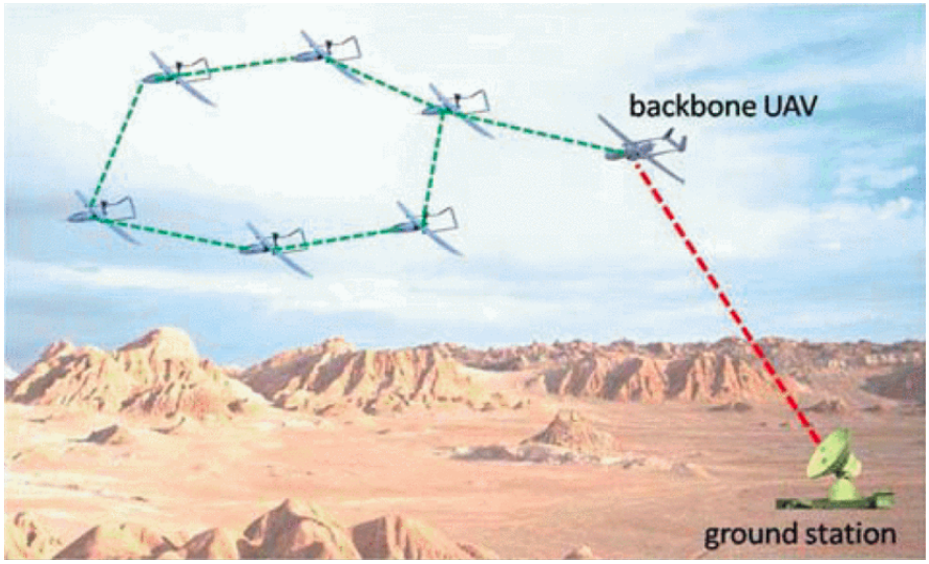

Figure 4.2: UAV Ad Hoc Network Architecture [5]

To solve the UAV Ad Hoc Architecture's limitation of requiring similar drone types for optimal network performance, the Multi-Group UAV Network and the Multi-Layer UAV Ad Hoc Network are suggested by [5]. In the Multi-Group UAV Network architecture, shown in Figure 4.3, the network is divided into separate groups by drone type and each group contains a backbone UAV directly connected to the ground control station. This way, a group of fixed-wing drones can fly independently from another group of slower multi-rotor drones. This architecture is a combination of centralized and decentralized architectures as intra-group communications can happen without the ground control station, while inter-group communications must be routed through the ground control station.

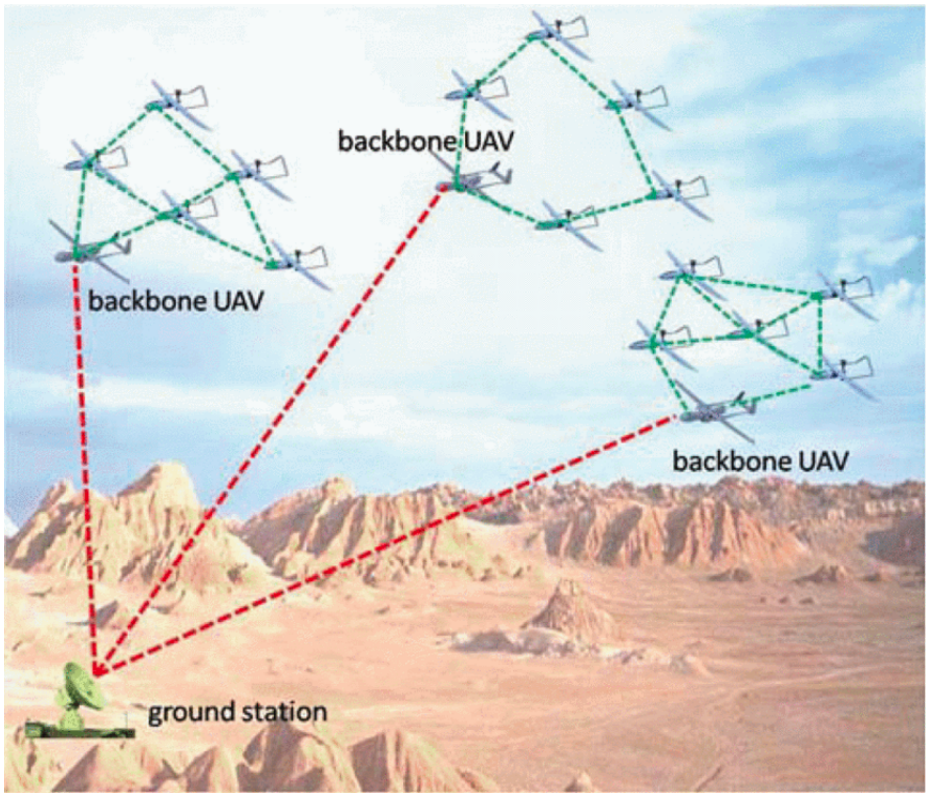

Figure 4.3: Multi-Group UAV Network Architecture [5]

To remove the single point of failure introduced by the ground control station in the Multi-Group UAV Network approach, the Multi-Layer UAV Ad Hoc Network (shown in Figure 4.4) allows for direct communications between the backbone UAVs. In this approach, only one backbone UAV must remain within range of the ground control station at all times, significantly increasing network area coverage as the backbone UAVs can forward messages through the network from the farthest UAV group.

The drone swarm used for this thesis is a homogeneous group of quadcopter drones that will be used mainly for aerial surveillance applications. Therefore, the speed and flight patterns of the drones will be similar, making the UAV Ad Hoc Network architecture ideal.

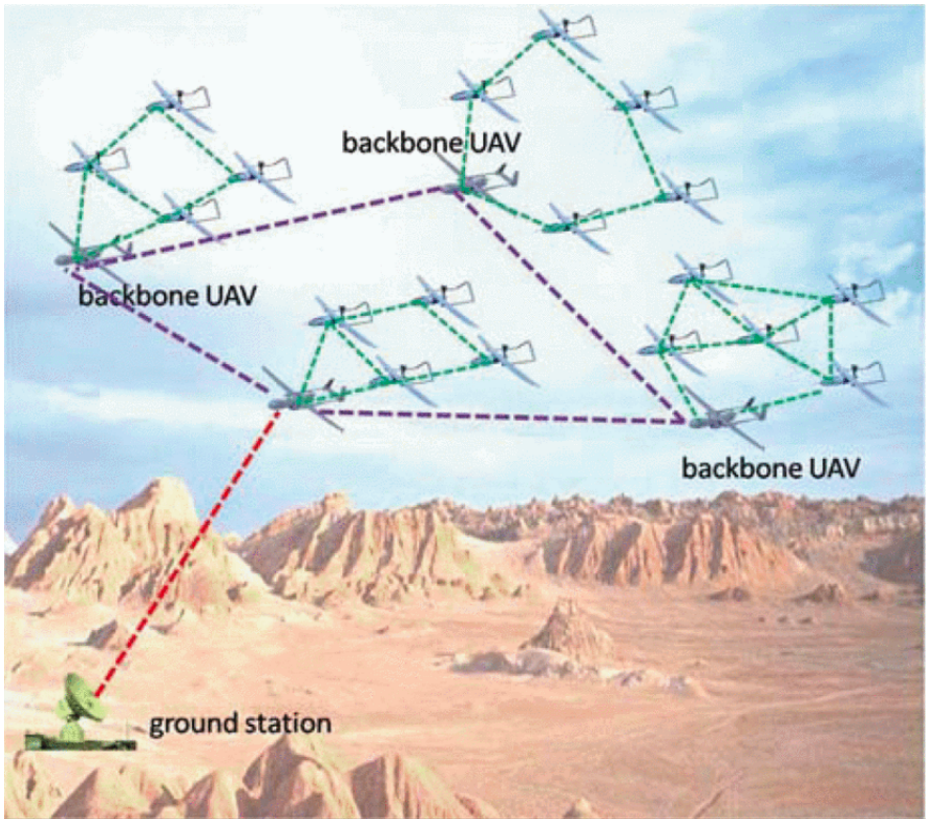

Figure 4.4: Multi-Layer UAV Ad Hoc Network Architecture [5]

4.1.4 WLAN Mesh Standard

An implementation of the UAV Ad Hoc Network architecture using Wi-Fi can be achieved using the 802.11s standard, also known as the WLAN Mesh Standard [24, 25]. The 802.11s standard was created by the IEEE 802.11 Task Group S and aims to introduce the underlying mechanism to enable multi-hop communications in Wi-Fi networks.

802.11s extends the capabilities of wireless devices to perform routing at the MAC layer, allowing each of the devices to forward messages through the network. Each device maintains a MAC layer routing table with paths to reach every other device in the network. Besides allowing for multi-hop routing, working at the MAC layer provides other benefits such as better Quality of Service (QoS) metrics and easy

integration with popular network protocols like ARP, DHCP, Spanning Tree and other higher level protocols [25]. In a drone swarm, reliable QoS metrics are essential as the drones continuously move and change the parameters of the network. This makes routing protocols at the MAC layer more efficient than routing at the IP layer. Therefore, the 802.11s standard is an efficient solution for swarm network implementation.

To take advantage of the benefits provided by 802.11s, an implementation of the standard and appropriate hardware are necessary. open80211s [26] is an implementation of the 802.11s standard that is integrated into the Linux operating system, making mesh networks possible on devices running a Linux distribution.

4.1.5 Message Routing

The 802.11s standard described in the previous section enables devices to forward messages through the network by using MAC layer routing tables stored at each device. However, the values stored in the routing tables are up to the chosen routing mechanism to decide. The routing mechanism determines which path is the optimal path to move a packet from a source node to a destination node. Two possible approaches to guide message through the network are flooding and routing [14].

The flooding approach broadcasts messages from the source node to all reachable nodes, which then rebroadcast the received message to all of their reachable nodes and so on until the messages reach their destination. The constant broadcasting of messages by multiple drones introduces the issue of "broadcast storms," where the network becomes saturated with messages as multiple drones broadcast their messages simultaneously. This can be avoided through techniques that use time-division access, which consists of synchronizing the internal clocks of each drone and allocating time

slots during which each drone is allowed to broadcast messages [27]. This eliminates packet collisions as multiple drones are not allowed to transmit messages during the same time slot. Flooding has the benefit of being very reliable in drone swarms since there are always multiple active paths between any pair of drones. If one specific path experiences high levels of interference at any given moment, the messages will also be transferred through paths with less interference, improving data integrity. On the other hand, flooding suffers the problem of taking up network bandwidth by broadcasting messages through paths that may be unnecessary and slowing down transmission rates through time-division access.

On the other hand, the routing approach stores routing tables at each drone to send messages through specific paths to reach their destination. If a path breaks, which could easily happen in a swarm network where drones are constantly moving, a self-healing mechanism is initiated to find an alternative path and the routing tables in the affected drones are updated. This self-healing mechanism introduces overhead into the network as drone swarms are very dynamic and the network topology constantly changes. However, routing has the advantage of being much more efficient in terms of bandwidth usage as messages are only routed through specific paths instead of the entire network.

As we can see, the flooding approach offers reliable packet delivery at the expense of increased bandwidth usage. With high-quality radio transceivers that enable large network bandwidths, the flooding approach is a solid option for drone swarms. However, high-quality radios are expensive and can easily turn a low-cost fleet into a high-cost project. Therefore, in the interest of keeping our fleet design low-cost, we will assume the use of low-cost radio transceivers with insufficient bandwidth to transmit messages using the flooding technique. Thankfully, the routing approach is common in FANETs, and a variety of routing protocols exist.

4.1.6 Routing Protocols Overview

Routing protocols can be divided into five categories: topology-based, position-based, clustering/hierarchical, swarm-based, and delay-tolerant [15]. From the five categories, topology-based is the most popular in FANETs as there is a variety of topology-based protocols that have been tested for drone swarm applications and show satisfactory performance. Topology-based protocols rely on information about the links between nodes to select the best path. This information is usually some type of link quality metric. Topology-based protocols can be further divided into three subcategories: proactive, reactive, and hybrid.

4.1.6.1 Proactive Protocols

In proactive routing protocols, all nodes in the network store a routing table that must be periodically updated and shared with other nodes in the network. This approach tends to provide low latency as all nodes know the path to reach each other at all times. However, proactive protocols are inefficient in terms of network bandwidth as nodes must constantly share their routing tables with each other, which consumes network resources. Also, as the network topology changes due to the constant movement of the swarm, all of the routing tables in the network must be updated. This makes proactive protocols inefficient in FANET applications as the routing tables may not be updated fast enough to keep up with the constantly changing network topology. Some popular proactive routing protocols for mesh networks are Optimized Link State Routing (OLSR) and Destination-Sequenced Distance Vector (DSDV).

4.1.6.2 Reactive Protocols

Reactive protocols attempt to eliminate the large bandwidth usage experienced in proactive protocols due to constantly updating routing tables. Instead of periodically updating all routing tables, reactive protocols follow an "on-demand" approach. Nodes in reactive protocols also maintain a routing table, but the table is only updated when an attempt to contact a node fails. This can happen if the destination node has never been contacted before or the previous known path to the destination breaks. To discover the new path to the destination, a route lookup process is initiated in which a packet is broadcasted on the network searching for the destination node. Once a path to the destination is found, both the destination and the source node update their routing table with the new path to reach each other. By using the on-demand approach, reactive protocols consume less bandwidth and tend to keep-up with topology changes at the expense of experiencing higher latency as a result of the route lookup process. Popular reactive protocols are Ad-Hoc On-Demand Distance Vector (AODV) and Dynamic Source Routing (DSR).

4.1.6.3 Hybrid Protocols

Hybrid protocols are a combination of the proactive and the reactive protocols that attempt to eliminate the bandwidth inefficiency of proactive protocols and the high latency of reactive protocols. Hybrid protocols are mainly used in Multi-Layer UAV Ad Hoc Networks where drones within groups tend to maintain a formation while different groups move independently of each other. Within each group, a proactive protocol can be used as the group's network topology remains constant and the routing tables remain valid most of the time. A reactive protocol is used to route messages between groups, as the groups tend to follow independent flight patterns that cause

frequent changes to the intra-group network topology. This way, the low-latency of proactive protocols is seen within groups and the bandwidth efficiency of reactive protocols is used between groups. Two popular hybrid protocols are the Zone Routing Protocol (ZRP) and the Hybrid Wireless Mesh Protocol (HWMP).

4.1.7 Routing Protocols Comparison

As mentioned before, topology-based protocols are popular in FANET applications and their performance has been tested in many research papers. In [6], the AODV, DSDV, DSR, OLSR, AOMDV, and HWMP routing protocols were tested using a drone swarm simulation. The metrics used for comparison of the different protocols were Packet Delivery Ratio (PDR), End-to-End Delay, and Throughput. The PDR is calculated as the number of packets successfully delivered divided by the number of packets transmitted. The End-to-End Delay is the total time required for a packet to travel from the source to the destination. The Throughput is calculated as the number of packets successfully delivered divided by the time it takes to deliver the packets. The protocols are tested with 20 drones in the simulation at varying flight speeds to test how the mobility of the swarm affects the routing protocol. Figure 4.5 below shows the results of the PDR test. The graph shows that at all speeds, the HWMP provides the best PDR with AOMDV in second place. This means that HWMP can reliably transmit packages through the network even as the network topology changes.

In terms of End-to-End Delay, HWMP also performed exceptionally well, maintaining some of the lowest end-to-end delays on par with DSDV. At a speed of 40 m/s, the OLSR showed a much lower end-to-end delay than all other protocols, but this datapoint seems irregular. End-To-End Delay can be seen below in Figure 4.6.

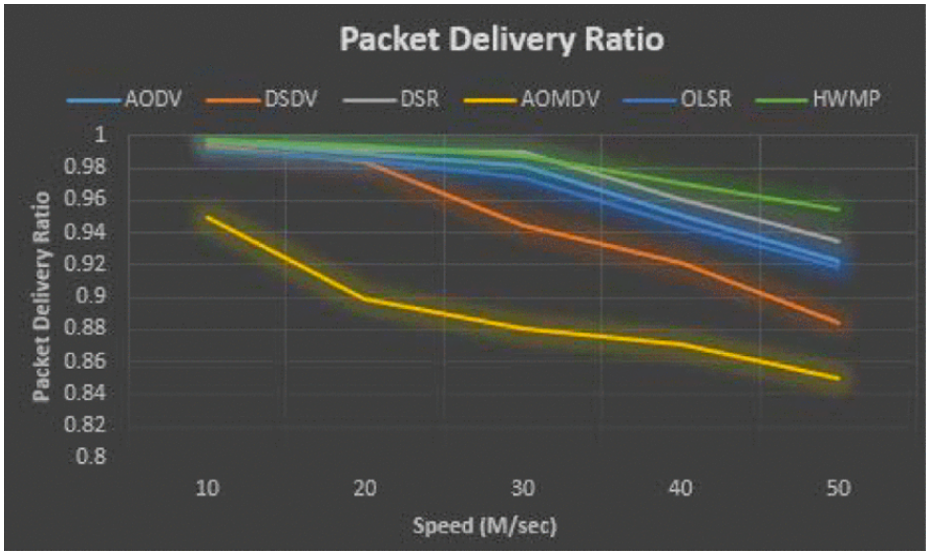

Figure 4.5: Comparison of Packet Delivery Ratio among Topology-Based Protocols [6]

The Throughput test also shows HWMP to perform better than the other protocols on average. Figure 4.7 below shows that HWMP maintained a higher Throughput at all speeds except 20 m/s, where AOMDV performed slightly better.

Based on the tests performed in [6], we can see that HWMP is the top performing topology-based protocol in swarm applications, capable of providing good PDR, low latency, and high throughput even in highly dynamic swarms with frequent network topology changes. Additionally, HWMP is the default routing protocol used in 802.11s mesh networks, making its integration into our drone swarm simpler than other protocols. For these reasons, HWMP will be used as the routing protocol for our low-cost drone swarm.

4.1.8 Hybrid Wireless Mesh Protocol

The Hybrid Wireless Mesh Protocol (HWMP) falls into the hybrid category of routing protocols as it can show both reactive and proactive behavior [28].

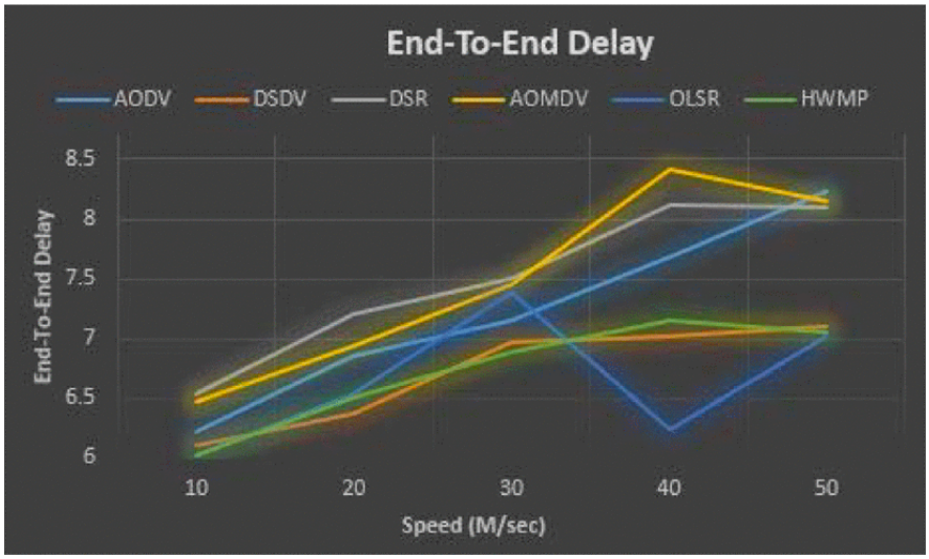

Figure 4.6: Comparison of End-To-End Delay among Topology-Based Protocols [6]

The reactive aspect of HWMP is an adaptation of AODV [29] that takes advantage of the improved QoS metrics provided by the MAC layer in which HWMP operates, as opposed to AODV which operates on the IP layer. AODV aims to find an optimal path between source and destination nodes by minimizing hop-count, which means using the path with the least amount of intermediate nodes. However, the hop-count metric does not take into account underlying link qualities such as congestion or signal strength that would make a longer, less saturated path better. By operating in the MAC layer, HWMP can calculate an "airtime metric," which takes into account a link's data rate, overhead, and frame error [25]. Using the airtime metric leads to a more robust network as the load can be properly balanced among many nodes to avoid saturating a small set of nodes.

The reactive behavior of HWMP follows the general behavior of reactive protocols. When a source node needs a path to a destination node, the source node broadcasts a path request message (PREQ) to all nodes within direct range. The neighboring nodes will then rebroadcast the message and so on until the PREQ message reaches

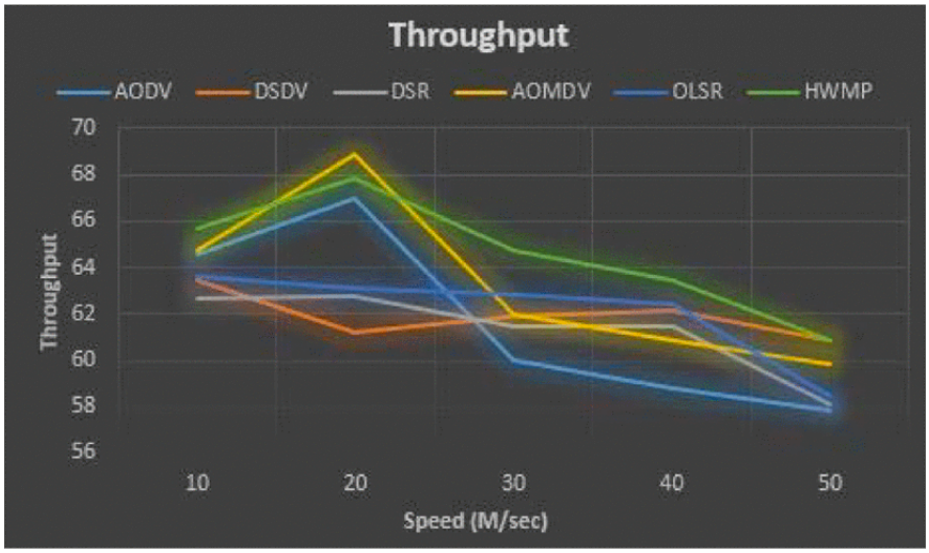

Figure 4.7: Comparison of Throughput Performance among Topology-Based Protocols [6]

the destination node. It is possible for the PREQ to be received again by a node that has already broadcasted the message. In this case, the node will identify the PREQ as an old message based on its sequence number and will not rebroadcast the PREQ. The destination node will receive multiple copies of the PREQ message corresponding to the multiple paths the message could take to reach the destination. The destination node will reply with a path reply message (PREP) to the best path found so far to the source. The established path will then be used for future communications between the two nodes until the path is lost or a better path is found.

A proactive component can also be configured in HWMP along with the reactive component. Proactive routing is used when a node is established as a root Mesh Point (MP), which is a type of node typically used for bridging the mesh network to other networks. A root MP attempts to maintain active paths to all nodes in the network by periodically broadcasting either proactive PREQ messages or root announcement messages (RANNs). While the inner workings of proactive PREQ messages and RANNs are slightly different, the end result is the same: the paths

from the root MP to all nodes and vice versa are periodically updated. This helps to reduce the latency of communications with the root MP, but also introduces the known drawback of proactive protocols which is an increase in bandwidth usage. Communications among other nodes in the network continue to use the reactive aspect of HWMP.

In our drone swarm application, we aim to minimize the overhead of HWMP to dedicate more network bandwidth for flight control data, video streams, and other custom application data. For this reason, we will not use the proactive aspect of HWMP. While this may result in higher latencies to reach the ground control station, the extra bandwidth will provide more reliable transmission of swarm data.

4.2 Drone Hardware

4.2.1 Flight Controller

The flight controller is an essential component on a drone mainly responsible for controlling the drone's movement by coordinating the motors to perform flight maneuvers. Thanks to the work of open-source projects on flight controller design, the PixHawk[30] flight controller is readily available to use and is a popular choice for custom low-budget drones. The PixHawk flight controller has gone through multiple design iterations and there are many different versions available in the market.

The firmware necessary to control a drone using a PixHawk is available in two libraries: ArduPilot[31] and PX4[32]. Functionality-wise, both libraries offer similar capabilities, and both are compatible with QGroundControl. One big difference between the two libraries is their licensing. PX4 has BSD licensing, while ArduPilot has GPL licensing. This means that anyone using ArduPilot on their product must

release their code modifications to the public. In contrast, anyone using PX4 on their product can keep their work private. For this reason, ArduPilot is arguably a more stable product. All modifications done by hobbyists and researchers to ArduPilot are available to other ArduPilot users. For most applications, both libraries are likely suitable for the job. For this project, ArduPilot was selected.

4.2.2 Companion Computer

A companion computer expands the capabilities of a drone by adding on board processing power for tasks like autonomous decision making. The chosen companion computer for this project is the Raspberry Pi 4 (RPi). The RPi is a popular small computer capable of running Linux-based operating systems, such as Raspbian, that contains sufficient processing power and memory for the complex tasks demanded from a drone companion computer. Additionally, there are a plethora of libraries and open-source projects for the RPi that, along with the flexibility of the Linux environment, make the RPi an outstanding board for drone research. In our design, a Raspberry Pi 4 will be responsible for advanced capabilities such as swarm networking, swarm coordination, and on-board decision-making.

4.2.3 Wi-Fi Adapter

The mesh network used in this project relies on the 802.11s standard. For a Wi-Fi adapter to support this protocol, it must be SoftMac compatible, meaning that frame management can be handled by the computer's kernel software instead of the adapter's hardware [33]. To handle frame management, the Linux module mac80211 is used. Therefore, any Wi-Fi chipset using a wireless driver compatible with mac80211 should be compatible with the 802.11s standard for mesh networking. A list of 802.11s

compatible drivers can be found at [34]. For this project, the rt2800usb chipset was selected, as it is 802.11s compatible, its drivers come pre-installed with Linux distributions, and Wi-Fi USB adapters with this chipset are readily available. For this project, the PAU06 USB Wi-Fi Adapter was selected due to its availability, low-cost, and flexibility to replace the antenna.

4.2.4 Wi-Fi Adapter Antenna

The type of antenna used at the ground control station and on each drone can affect both the range of the drone fleet and the reliability of the connection. While antenna design and RF transmission patterns are broad fields that would require their own thesis, we can extract high-level knowledge from these fields to choose antennas for our drones. We mainly need to be concerned with the type of antenna and the gain.

4.2.4.1 Omnidirectional vs Directional Antennas

A common analogy to explain signal radiation patterns from antennas is to think of the radio signals as light and the antenna as the light source. Omnidirectional antennas are similar to a light bulb that radiates light equally in all directions at a shorter range, while directional antennas are similar to flashlights that focus their light in a specific direction but reach longer ranges [35]. For applications where all-around area coverage is needed, omnidirectional antennas are better as the antenna does not need to aim in a specific direction. However, omnidirectional antennas tend to reach shorter ranges. If the application requires transmission of a signal over long ranges in a specific direction, directional antennas are ideal.

In a drone fleet, where drones are continuously moving and must maintain a connection with each other, an omnidirectional antenna mounted on the drones is a suitable

solution. At the ground control station, both types of antennas are viable depending on the expected flight patterns. If the entire fleet is expected to fly together as a unit in a specific direction, a directional antenna can increase the fleet's range by pointing in the direction of the fleet. A drawback to this approach is that a person must be designated to keep pointing the antenna towards the fleet or some type of signal tracking technology must be added to the antenna, which could significantly increase the cost of the project. On the other hand, if the fleet is expected to remain relatively close to the ground control station or fly around it, an omnidirectional antenna can still provide good range and eliminate the need of a tracking system.

4.2.4.2 Antenna Gain

While the radiation pattern of an antenna is mostly determined by the type of antenna, the radiation pattern can be further manipulated with the antenna gain. Antenna gain is measured in units of *decibels relative to isotropic* (dBi). As a rule of thumb, higher gain antennas transmit their signal farther, but narrow down the direction in which the signal is transmitted. The effect of increasing the gain of a directional antenna is to narrow down the breadth of the transmitted signal in the specific direction the antenna is pointing. On the other hand, increasing the gain of an omnidirectional antenna increases the range of the signal along the antenna's horizontal plane, but reduces the range of the signal in the vertical plane. An illustration of this is provided in Figure 4.8.

Again, the best antenna gain is determined by the specific application and the expected flight patterns. If all drones are expected to fly at the same altitude throughout the duration of a mission, having a high gain omnidirectional antenna could extend the range of the fleet as the signal would be focused along the horizontal plane, where the drones are located. If the drones are expected to fly at different altitudes

throughout the mission, then a lower gain antenna would be more suitable. A lower gain antenna would reduce the range of the fleet, but the drone-to-drone connection would be more stable as the altitude of individual drones changes.

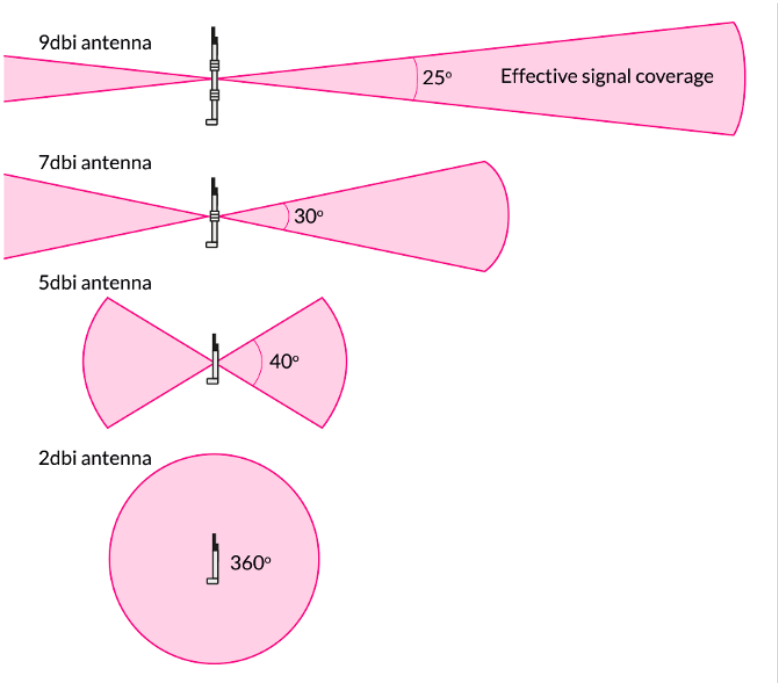

Figure 4.8: Effect of gain on omnidirectional antenna signal [7]

4.2.5 Neural Accelerator

A goal of this project is to perform real-time object detection. This can be achieved by mounting devices known as neural accelerators on the drone. These devices have architectures optimized for machine learning applications that require parallel computations. Some neural accelerators contain discrete Graphic Processing Units (GPUs) or Tensor Processing Units (TPUs) to accelerate the calculations in neural networks. A few popular neural accelerators were considered for this project, with key requirements being cost, flexibility, and performance. Specifically, the Jetson Nano Dev Board, Google Coral Edge TPU Dev Board 4GB, and Intel Movidius were considered, as they are all small and light enough to mount on a drone.

4.2.5.1 Cost

The prices of the three main neural accelerators at the time of writing are shown below in Table 4.2. The cheapest of the three is the Intel Movidius Stick (which requires a Raspberry Pi or some other computer to function), with the Jetson Nano having a slightly higher cost. The Google Coral Dev Board is significantly more expensive than the other two options.

Table 4.2: Price comparison between the three main neural accelerator platforms.

Device	Cost (US Dollars)
Jetson Nano 4GB	$99
Google Coral 4GB Dev Board	$170
Intel Movidius Computer Stick 2	$79

4.2.5.2 Performance

Performance for neural accelerators is typically measured in terms of time per inference (latency) or in terms of frames-per-second in object detection applications. NVIDIA's benchmark results, shown in Table 4.3 below, provide a performance comparison in terms of FPS for the main neural accelerator platforms.

The results show the Intel Movidius to have much lower performance than the Jetson Nano and Google Coral Dev Board. On most models supported by the Google Coral Dev Board, the Coral Dev Board outperformed the Jetson Nano thanks to its TPU and use of highly optimized versions of the machine learning models. The Jetson Nano also performed very well on all models and even outperformed the Google Coral Dev Board when using Inception V4.

Table 4.3: Performance comparison between popular neural accelerator platforms in terms of FPS [2]

Model	Application	Framework	NVIDIA Jetson Nano	Raspberry Pi 3	Raspberry Pi 3 + Intel Neural Compute Stick 2	Google Edge TPU Dev Board
ResNet-50 (224×224)	Classification	TensorFlow	36 FPS	1.4 FPS	16 FPS	DNR
MobileNet-v2 (300×300)	Classification	TensorFlow	64 FPS	2.5 FPS	30 FPS	130 FPS
SSD ResNet-18 (960×544)	Object Detection	TensorFlow	5 FPS	DNR	DNR	DNR
SSD ResNet-18 (480×272)	Object Detection	TensorFlow	16 FPS	DNR	DNR	DNR
SSD ResNet-18 (300×300)	Object Detection	TensorFlow	18 FPS	DNR	DNR	DNR
SSD Mobilenet-V2 (960×544)	Object Detection	TensorFlow	8 FPS	DNR	1.8 FPS	DNR
SSD Mobilenet-V2 (480×272)	Object Detection	TensorFlow	27 FPS	DNR	7 FPS	DNR
SSD Mobilenet-V2 (300×300)	Object Detection	TensorFlow	39 FPS	1 FPS	11 FPS	48 FPS
Inception V4 (299×299)	Classification	PyTorch	11 FPS	DNR	DNR	9 FPS
Tiny YOLO V3 (416×416)	Object Detection	Darknet	25 FPS	0.5 FPS	DNR	DNR
OpenPose (256×256)	Pose Estimation	Caffe	14 FPS	DNR	5 FPS	DNR
VGG-19 (224×224)	Classification	MXNet	10 FPS	0.5 FPS	5 FPS	DNR
Super Resolution (481×321)	Image Processing	PyTorch	15 FPS	DNR	0.6 FPS	DNR
Unet (1x512x512)	Segmentation	Caffe	18 FPS	DNR	5 FPS	DNR

4.2.5.3 Flexibility

Due to memory limitations, the Intel Neural Computer Stick is limited to lightweight models only. The Google Coral Dev Board with its 4GB configuration is capable of running complicated models. However, the Google Coral is limited to TensorflowLite models that have been optimized for the TPU [36]. This greatly reduces the flexibility of the Google Coral, as it limits the possibilities for using novel models with custom layers. On the other hand, the Jetson Nano is extremely flexible and can run models from any framework such as DarkNet, Pytorch, and Tensorflow. Therefore, the Jetson Nano is the most flexible device.

4.2.5.4 Neural Accelerator Selection

The comparisons above led to the following conclusions. If high FPS performance is not critical and the application does not require the use of memory-intensive models, the Intel Movidius is a good cost-saving solution. Otherwise, if performance is the most important factor for the target application, the selected object detection model is compatible with the Google Coral, and the extra cost can be spared, then the Google Coral Dev Board will offer the best results. For most applications that require an affordable solution, flexibility to explore different models, and satisfactory performance, the Jetson Nano is a solid neural accelerator that can satisfy most applications. Since the focus of this thesis is to compile a drone kit that is highly customizable, the Jetson Nano strikes the best balance between cost, flexibility, and performance.

An aspect that could steer researchers away from using a Jetson Nano is power consumption. The Jetson Nano weighs more and consumes more power than the other two neural accelerators, which decreases the total flight time of the drone. If flight

time is a priority, users of the proposed drone design could gain a couple extra minutes of flight time by using lighter and less power intensive neural accelerators. For our purposes of developing a highly customizable drone, the slight increase in flight time provided by other devices does not compensate for the loss in performance and/or flexibility.

Chapter 5

IMPLEMENTATION

5.1 On-Board Hardware Components

This section provides brief descriptions of some common drone components. This will make the upcoming discussion of our drone design easier to comprehend.

• Flight Controller: The flight controller is responsible for controlling the movement of the drone, interfacing with other peripherals necessary for flight (GPS, RC receiver, etc.), and transmitting flight data to the ground control station. In our design, the flight controller is the PixHawk 4. Flight controllers like the PixHawk 4 contain a variety of internal sensors such as accelerometer, gyroscope, magnetometer, and barometer.

• GPS Module: As the name suggests, the GPS module provides the GPS location of the drone to the flight controller. Additionally, GPS modules contain a magnetometer that is used along with the flight controller's magnetometer to increase accuracy in determining the orientation of the drone. This component is essential for autonomous flights.

• Power Module: The power module serves the purpose of stepping down the high voltage from the LiPo battery to the appropriate voltage to power the flight controller. Additionally, the power module provides current consumption and voltage measurements to the flight controller. This is essential for estimating the remaining charge of the battery during flight. In our drone, a PM02 power module is used.

• Power Distribution Board: The power distribution board is typically used to distribute power from the battery to the various motors on the drone. In some drone designs, the power distribution board is a discrete board that serves as both a way to distribute power to the motors and a power module to measure voltage and current.

• Electronic Speed Controller (ESC): An ESC is a component directly connected to a motor to control the speed of the motor based on the input signals received from the flight controller. ESCs are typically connected to the power distribution board for direct access to power from the battery. This way, an ESC can provide sufficient current to the motor to reach the desired RPM. Each motor on the drone is controlled by its own ESC.

• Universal Battery Eliminator Circuit (UBEC): The purpose of a UBEC is to step-down the high voltage from the battery to an appropriate voltage for on-board components. While the power module already steps down the voltage to power the flight controller, the supported current from the power module is usually only enough for the flight controller. If an additional component needs to be powered from the battery, such as a companion computer, then a UBEC is used to power such a component.

• RC Receiver: The RC receiver is the radio that directly communicates with the RC controller to enable manual control of the drone by a pilot. The RC receiver is a requirement for drone flights as the pilot should be able to take over control of the drone at any time for safety purposes.

• SiK Radio: These radios operate in the 915MHz range in the United States and serve the purpose of providing a communication channel for sending flight commands and retrieving telemetry data through a ground control station. SiK radios provide good flight range out of the box; however, they do not support packet forwarding

and their available bandwidth tends to be low. Therefore, SiK radios serve only as a control channel.

5.2 On-Board System

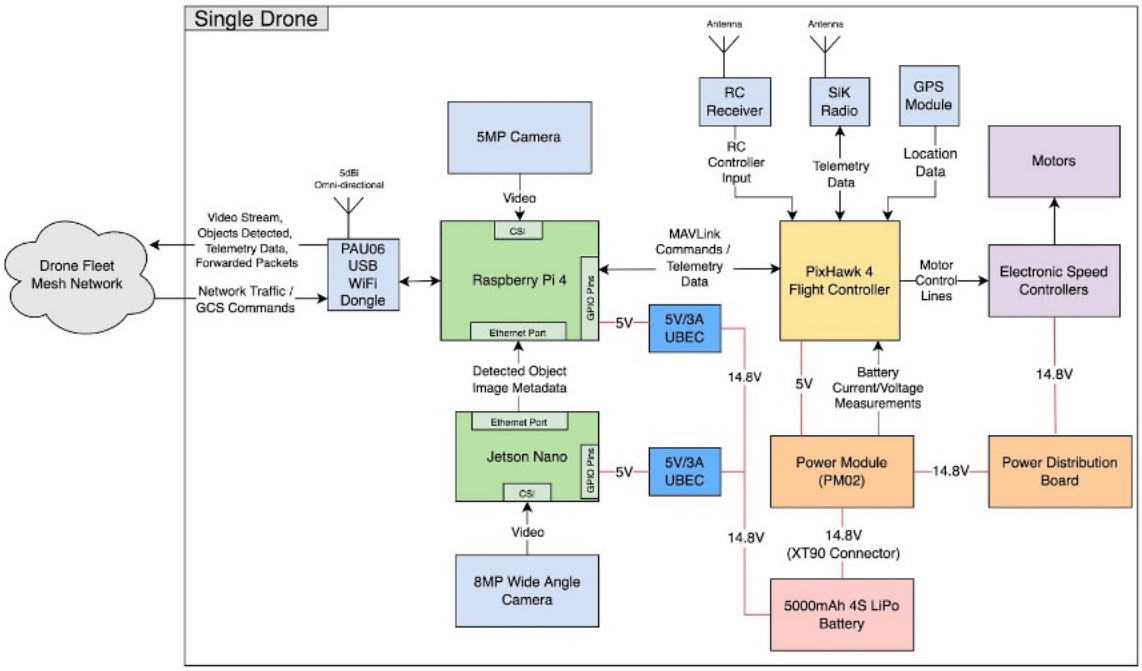

Figure 5.1: System Design Block Diagram

Figure 5.1 above provides a diagram of our proposed drone design for real-time object detection applications. In this design, a Raspberry Pi 4 (RPi) is used as a companion computer and a Jetson Nano is used to perform object detection on-board.

5.2.1 Raspberry Pi 4 Functionality

The Raspberry Pi 4 was selected for its processing power, ease of use, and networking capabilities. The RPi must perform multiple tasks simultaneously, which include streaming video to the ground control station, preparing custom "object detected" messages to send to the GCS, forwarding messages through the mesh network and

relaying telemetry data between the flight controller and the GCS. If advanced autonomous behavior and swarm coordination is implemented, this logic will run on the companion computer. Therefore, the RPi must have sufficient processing power to handle multiple processes in parallel. While swarm coordination and fully autonomous behavior are not implemented in this paper, the drone design aims to provide the necessary platform to continue research in these areas by providing sufficient processing power. The Raspberry Pi 4 offers ease of use as there are plenty of resources for its use as a companion computer. For example, the DrokeKit platform is an open-source project with resources for developing custom programs for drone applications [37]. These custom programs can be executed on the RPi to access the flight controller's data and even gain full control over the drone. Lastly, the RPi runs Raspbian, a Linux distribution that supports the 802.11s standard to participate in a mesh network. These qualities make the RPi an effective companion computer.

5.2.2 Jetson Nano Functionality

While the RPi is a powerful small computer, its architecture is not ideal for object detection purposes. To process the neural networks necessary for object detection, the Jetson Nano is used. As seen in Section 4.2.5.2, the Jetson Nano performs well on popular object detection models but suffers from slow inference times on some of the more resource intensive neural network architectures. As there exists a tradeoff between model accuracy and inference time, there is not one best model for drone swarm object detection applications. The best model depends on the specific parameters of the target application. In our intended target application of shark spotting, we would prioritize model accuracy over inference time as the slow flight pattern of the drones provides plenty of time for slow inferences.

5.2.3 Camera Usage

Consider the case where there is only one camera on-board. There are two purposes for recording video: (1) streaming video to the GCS for researchers to observe and (2) performing object detection. These applications have competing requirements, as video streaming aims to maximize video quality for the available network bandwidth, while object detection aims to maintain a steady video resolution for the object detection model. If the object detection model relies on a high-resolution video but the network bandwidth is low, then the high-resolution video will not be streamed to the GCS when a lower resolution video could have been streamed without any issues. On the other hand, if the object detection model requires low-resolution images to perform fast inferences, the video stream to the GCS will be the same low-resolution even when the network bandwidth allows for higher-resolution video. While it is possible to record video at the highest-resolution needed and then scale it down for either fast inferences or streaming on a low-bandwidth network, this approach increases system complexity and unnecessarily increases utilization of processing power by doing video rescaling on-the-fly. Additionally, sharing a video stream requires coordination between the Jetson Nano and the Raspberry Pi so that each device can use the video at their desired frames-per-second (FPS). For example, the RPi may want to stream video at 20FPS while the Jetson Nano only needs 5FPS due to slow inference time of heavy neural network models. We propose to mount two cameras on the drone. One camera is attached to the Raspberry Pi and is used only for video streaming to the GCS. This allows the RPi to modify the resolution and FPS of the video stream as necessary to adapt to the network bandwidth without affecting the video stream seen by the Jetson Nano. The other camera is attached to the Jetson Nano, allowing it to control the video stream to match the requirements of the object-detection model. RPi and Jetson Nano cameras are light, relatively inexpensive (under $30 USD), and

have low power-consumption (1W). Therefore, their usage does not add significant cost to the design while providing each board full autonomy over how they use their individual video streams.

5.2.4 Object Detection Notifications

One way to notify the ground control station when an object of interest is detected is to send an image of the detected object through the network to the GCS. This provides researchers on the ground an immediate image of the found object. However, using this approach introduces the risk of saturating the network with images if multiple drones are detecting objects over multiple frames simultaneously. This can then lead to unreliable communications or even loss of connection as drones cannot transmit their telemetry data over the saturated network. An alternative approach for notifying the GCS when objects are detected that reduces the risk of network congestion is to send metadata about the event. This reduces congestion as metadata is much smaller than images. The metadata may include information such as number and types of objects detected, GPS location of drone at time of detection, exact time of object detection with respect to the current video recording, and the name of the file in which the video is locally being recorded. This approach provides enough information for immediate and future action. As soon as the notification is received, the GCS can decode the message and notify the user. The GPS location in the incoming notification can be used to mark the location where the object of interest was detected on the GCS map. After the flight mission is finished, researchers may use the file name and exact time included in the notification data to locate the video frames marked as interesting by the object detection model. If researchers want to observe what a drone is detecting in real time, they may enable the video stream from the desired drone. This video stream will be provided by the on-board Raspberry Pi and will not

include the bounding boxes from the object detection model, but the video stream itself may be enough to identify what the object-detection model found.

5.2.5 Extra Communication Channels

While the mesh network aims to provide the necessary link from the GCS to all drones in the swarm, it is possible to lose connection to a drone mid-flight if the swarm formation happens to isolate the drone from the rest of the swarm. When the drone is isolated, it loses connection to all intermediate drones that were providing a relay point to reach the GCS through the mesh network. Without any additional communication channels, the flight controller will determine that communication was completely lost and will "return to launch" if such safety measure is enabled on the flight controller. This is undesirable as the loss of communication may only be temporary and the drone could have continued its mission and regained connection to the network after a short period of time. To reduce the frequency of this scenario, a SiK telemetry radio may be connected to the flight controller as a secondary telemetry channel. These radios only offer data rates of up to 250Kbps, but they offer long-range communication for telemetry data [38]. This can assist in maintaining connection to a drone that has been disconnected from the mesh network and avoiding early mission termination.

Besides using a SiK telemetry radio as a backup channel for telemetry data, an RC receiver must also be connected to the flight controller. The RC receiver enables manual control of the drone through an RC controller, allowing the pilot to gain control over the drone at any time. This serves as a safety measure and could also assist in recovering a disconnected drone as RC receivers tend to provide long-range communications to the RC controller. The extra communication channels discussed in this section do not provide the necessary bandwidth and infrastructure for a drone

swarm with object-detection capabilities but are great mechanisms for drone recovery purposes.

5.3 Drone Body Selection

There are multiple types of drone bodies with varying sizes and number of motors. For our object detection application, we want the drone to be as light as possible to minimize power consumption while at the same time spacious and powerful enough to carry all the components needed in our design. The S500 quadcopter frame is a great option as it is light, provides plenty of real estate for addition of custom components, and is relatively inexpensive. Larger drones and designs with more motors could provide even more real estate and power for carrying custom loads; however, the space and lift power necessary for our specific application is already provided by the cheaper S500 design. As a result, the S500 frame was chosen as the ideal drone body for our low-cost swarm implementation. Besides the drone body, there are multiple components necessary for a drone to be functional, such as a flight controller, a power module, GPS module, motors, propellers, electronic speed controllers, etc. Selection of these components can be done manually to build a fully customized drone or a standardized "almost ready-to-fly" drone kit may be used. For this project, the Holybro S500 V2 Kit [3], shown in Figure 5.2, was selected. The kit was chosen for its low-cost and completeness, as it includes nearly all the components of a functional drone. Additionally, the S500 V2 Kit is a popular choice and there is plenty of community support.

Figure 5.2: Holybro S500 V2 Kit [3]

5.4 Drone Weight Estimate

Estimating the total weight of the drone is an important part of drone design as it determines the type of motors necessary to lift the drone and significantly affects total flight time. The total weight of the drone consists of the cumulative weight of the body itself and all the components mounted on it. Table 5.1 below shows our drone weight estimate.

Table 5.1: Total drone weight estimate. Only the main on-board components were taken into account as smaller components have negligible weight. The total weight of our drone design is approximately 1611g.

Component	Weight (g)
Drone kit (Body, Motors, PixHawk, GPS, PM02)	935
5000mAh LiPo Battery	492
Raspberry Pi 4	46
Jetson Nano Dev Board	138
Total	1611

5.5 Weight vs Thrust Analysis

With an approximation of the total drone weight, a set of motors must be chosen that provide sufficient thrust to lift the weight of the drone and accelerate in different directions. A general rule of thumb is to have a 2:1 thrust to weight ratio. This means that the motors on the drones should be capable of cumulatively producing thrust equal to twice the weight of the drone. With this ratio, the drone should hover in place at 50% throttle. A 2:1 thrust to weight ratio is likely not enough for performing acrobatic maneuvers, but it is sufficient for aerial surveillance applications where the goal is to fly steadily. Table 5.2 below shows some of the properties of the motors included with the S500 V2 Kit when using the included propellers. For reference, the motors are AIR2216 KV880 with T1045 propellers.

Table 5.2: S500 V2 Kit motor properties. Values shown are per motor [3].

Throttle	Thrust (g)	Current (A)	Input Power (W)
50.00%	435	3.5	56
55.00%	527	4.6	73.6
60.00%	608	5.6	89.6
65.00%	702	6.8	108.8
75.00%	888	9.5	152
85.00%	1076	12.3	196.8
100.00%	1293	16.2	259.2

In the previous section, we estimated the total weight of the assembled drone to be about 1.6kg. Therefore, to hover in place, all four motors must cumulatively produce about 1.6kg of thrust, or about 400g each. The motors that come in the kit produce this amount of thrust at slightly under 50% throttle, as shown in Table 5.2 above. This shows that the included motors have plenty of power for our specific design. If the total weight of the drone were to increase significantly due to additional on-board components, it would be worthwhile to find an appropriate set of motors that produce the necessary thrust.

5.6 Motor Current Consumption

In aerial surveillance applications, the typical flight pattern consists of mostly straight paths at constant velocity. Since there are only occasional spikes in acceleration to change the direction of flight, the average thrust produced by the motors over an entire mission will remain fairly close to the thrust necessary for hovering. If flying in strong wind conditions, the average thrust produced by the motors is expected to increase as the motors must compensate for the force of the wind. In the following calculations, we assume flights are performed in low wind conditions and ignore the effect of wind on average motor thrust. In our drone design, hovering requires about 1.6kg of thrust, which we will consider to be the average motor thrust over an entire mission. At 1.6kg of thrust, each motor consumes about 3.15A or about 12.6A for all four motors. The XT60 connector in the PM02 power module is rated for 30A continuous current, so performing surveillance missions should pose no problems in terms of power consumption.

However, with only 30A of continuous current, we are limited to a maximum continuous throttle of about 65-70%, which yields about 2800g of thrust. If the intended application requires heavier payloads mounted on the drone or fast flight maneuvers, the average current consumption could reach or surpass the continuous current supported by the connectors and wires. This becomes a safety hazard as the connectors and wires could overheat and result in drone failure mid-flight. A simple solution is to replace the XT60 connector and 12 AWG wires that come with the PM02 power module for an XT90 connector and 10AWG wires. This replacement would allow for 45A of continuous current and up to 90A of burst current [39], which translates to about 80% continuous throttle (over 3600g of thrust) and even 100% throttle for

shorter periods of time. If the connector is modified, the selected LiPo battery should have an XT90 connector.

5.7 Flight Time Estimate

Lithium Polymer (LiPo) batteries are the most common power source on drones due to their high discharge rate and relatively low weight and cost [40]. While performing fast maneuvers, the motors of a drone may draw large amounts of current (up to about 64A in our design) and the power source must be able to handle such discharge rates. Common LiPo batteries used in drones have a capacity of 5000mAh with discharge rate of 50C. The C-rate specifies the maximum discharge rate of the battery with respect to its capacity and the max discharge current is calculated as *capacity x C-rate*. In the case of the common 5000mAh 50C LiPo battery, a maximum discharge current of 250A may be drawn. Therefore, LiPo batteries are ideal for the needs of drone hardware.

With the capabilities of a LiPo battery in mind, we can estimate the total power draw of the system and the average flight time of our drone. Table 5.3 below shows the power needs for each of the on-board components of our design.

Table 5.3: Approximate power draw from all on-board components

Device	Power Draw (Watts)
Raspberry Pi (all cores active)	6
PAU06 Wi-Fi Adapter	1.5
5MP Camera	1
8MP Camera	1
Jetson Nano Dev Board	10
PixHawk (with peripherals)	2.5
Motors producing 1611g of thrust	204
Total	226

As discussed in the previous section, we can estimate the average thrust produced by the motors to remain close to the drone weight in aerial surveillance applications. Therefore, the power draw of the motors was calculated at 1611g of thrust, which matches the weight of our drone.

With an estimate for total power consumption, the average flight time of our drone can be calculated. Assuming we use a 5000mAh LiPo battery, the capacity rating suggests the battery can sustain 5000mA of current, or 5 Amps, for an hour. However, the cells of a LiPo battery may be damaged if they are completely discharged. Therefore, we should only use about 80% of the battery's capacity to avoid damaging the cells. This leaves us with 4 Amps of current for an hour (4Ah). Table 5.3 above shows that about 226W of power will be consumed on average. At 16V (the voltage of a 4S LiPo battery), this translates to a current draw of 226W/16V = 14.125A. With this information, the flight time is calculated as 4Ah / 14.125A = 0.283 hours = 17 minutes.

The flight time estimate calculated may vary based on the configuration of the Raspberry Pi and Jetson Nano. The power draw of the Raspberry Pi was estimated with the assumption that the RPi is used to its full capacity with all cores active. The Jetson Nano has two configurable power modes: 5W and 10W. Our calculations assume 10W power mode is used. Flight time may be increased by decreasing the power draw of these devices.

5.8 Network Implementation

In this section we will discuss the steps to configure our devices to join a mesh network using the open80211s implementation of 802.11s that is integrated into the

Linux kernel [33]. This section assumes the use of a mesh compatible Wi-Fi adapter. A PAU06 USB Wi-Fi Adapter was used for this project.

5.8.1 Connecting Raspberry Pi to Mesh Network

To configure the connection to the mesh, the *ifconfig* and *iw* command line tools are necessary, which are installed by default on the Raspbian Buster distribution used in this project. Figure 5.3 below shows the script executed to join the mesh network.

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          setMeshPointStartupScript.sh
# Required-Start:    $remote_fs $sysylog
# Required-Stop:     $remote_fs $sysylog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Connect to mesh network at startup
### END INIT INFO

sudo ifconfig wlan0 down
sudo ifconfig wlan1 down
sudo iw dev wlan1 interface add mesh0 type mp mesh_id PIMESH
sudo ifconfig mesh0 down
sudo iw dev mesh0 set channel 1 HT20
sudo ifconfig mesh0 up
sudo ip addr add 192.168.10.3/24 dev mesh0
```

Figure 5.3: Script to join a mesh network from Raspberry Pi using a static IP.

The first two commands disable the wireless interfaces, which correspond to the RPi's internal Wi-Fi chip (wlan0) and the USB Wi-Fi adapter (wlan1). This step is necessary to enable modifications to those interfaces. The third line adds an interface called "mesh0" to the USB Wi-Fi adapter, establishes the device type as "mp" (Mesh Point), and establishes the ID of the desired mesh network to be "PIMESH." The ID of the network can be changed to any string with a max length of 32 bytes. All devices must use the same mesh ID to communicate with each other. With the mesh0 interface established, it is then disabled to set the channel number and channel type

used by mesh0. The available channels range from 1 to 11, and the channel type may be set to NOHT, HT20, or HT40+. Changing the channel number is useful to avoid interference from other devices operating in the 2.4GHz range used by the USB Wi-Fi adapter. All devices in the mesh network must operate in the same channel. The available channel types are dependent on the USB Wi-Fi adapter used. The PAU06 adapter supports 802.11 b/g/n standards. As shown in Table 4.1, the 802.11b and 802.11g standards offer much lower data rates than 802.11n. If the NOHT (No High-Throughput) channel type is used, the adapter will not utilize the 802.11n standard and will be limited to lower data rates. The HT20 and HT40+ channel types use the 802.11n standard to provide higher data rates. HT20 uses a channel width of 20MHz, while HT40 uses a channel width of 40MHz. Using a wider channel should provide better bandwidth; however, using wide channels in the 2.4GHz range increases the likelihood of experiencing interference from overlapping channels [41]. Therefore, we chose to use a channel width of 20MHz by setting the channel type to HT20. With the mesh0 interface parameters configured, the interface is reenabled and a static IP address is established. The static IP can be set to any address as long as the subnet is the same as the other devices in the mesh network and the subnet does not interfere with any nearby networks. Also, the IP address should not be shared with any other device in the mesh network. In our project, we had a limited number of devices, which made it easy to keep track of their IP addresses and avoid duplicates. If using a large drone swarm, manually keeping track of IP addresses may become cumbersome. In this case, the last line of the script may be omitted and a DHCP server may be used in the network to dynamically provide IP addresses as new devices join the network. The device running the DHCP server would still be given a static IP.

On a Raspberry Pi, the script in Figure 5.3 can be configured to execute on system startup to automatically join the mesh network with the given static IP or a dynamic

IP if a DHCP server is enabled. This can be done using the script shown in Figure 5.4 below, where "setMeshPointStartupScript.sh" contains the code in Figure 5.3.

```
#!/bin/bash
sudo cp setMeshPointStartupScript.sh /etc/init.d/
cd /etc/init.d
sudo chmod +x setMeshPointStartupScript.sh
sudo update-rc.d setMeshPointStartupScript.sh defaults
echo "Please run sudo reboot"
```

Figure 5.4: Script to enable Raspberry Pi to automatically join mesh network on system startup

5.8.2 Connecting PC to Mesh Network

A PC is necessary to function as the Ground Control Station that monitors the drone swarm and provides a user interface to researchers. There are two ways to connect a PC to the mesh network: (1) establishing the PC itself as a mesh point in the network, or (2) setting up an intermediate device as a mesh gate that provides the PC access to the mesh network.

The first option is only available on PC's running a Linux distribution as the Windows and MacOS operating systems do not support the 802.11s standard. Computers with a Linux distribution installed, such as Ubuntu, can be directly connected to the mesh network through a mesh compatible Wi-Fi adapter because open80211s is integrated into the Linux kernel. This is the cleaner solution as it reduces the hardware necessary to operate the drone swarm.

If a machine with a Linux installation is not available, an intermediate Raspberry Pi or some other computer that is Linux compatible is necessary to act as a gate to the mesh network. A possible setup consists of a Raspberry Pi connected to the mesh network through a mesh compatible USB Wi-Fi adapter and connected to the PC

over Ethernet. An Ethernet link is desirable as it provides fast transmission speeds and a reliable connection to the PC. This prevents the gate to the mesh network from becoming a bottleneck. Besides following the procedure outlined in the previous section to make the Raspberry Pi a mesh point, additional configuration is necessary to bridge the Ethernet and mesh interfaces at the Raspberry Pi [33]. This GCS setup was not implemented in our project and will therefore not be further explored.

If a Linux computer is used as the GCS, connecting to the mesh network is fairly straightforward. Figure 5.5 below shows the script executed to configure an Ubuntu machine as a mesh point.

```
#!/bin/bash
interfaceName=`ls /sys/class/net/ | grep wlx`

sudo ifconfig $interfaceName down
sudo iw $interfaceName set type mp
sudo ifconfig $interfaceName up
sudo iw $interfaceName mesh join PIMESH freq 2412 HT20
sudo ip addr add 192.168.10.1/24 dev $interfaceName
```

Figure 5.5: Script to join mesh network from Ubuntu machine

There are slight differences from the script used on the Raspberry Pi. On the Raspberry Pi, the interface name given to the USB Wi-Fi adapter is always wlan1, even if different adapters are used. On Ubuntu, the interface name may change with different Wi-Fi adapters. The first line of the script in Figure 5.5 tries to eliminate the need to manually input the interface name by searching for an available interface with the pattern "wlx". This pattern was used because all the interface names given to the different Wi-Fi adapters used in our project contained "wlx" as the first three characters. Once the WiFi adapter interface is identified, the interface is then set as a mesh point and given a mesh ID (PIMESH), channel frequency (2412), and channel type (HT20). The mesh ID and channel parameters must match the parameters of the other devices in the network. The channel frequency is provided in MHz instead of

a channel index. A frequency of 2412MHz corresponds to channel 1, which was used in the Raspberry Pi setup. Lastly, a static IP and subnet is given to the interface. Again, the subnet must match the subnet of the other devices in the network and the IP address must be unique.

5.8.3 Testing mesh network connections

After configuring all companion computers and the GCS as mesh points in the same mesh network, the connection between them should be verified. One way to do this is with the *nmap* command-line tool [42]. This tool provides advanced functionality that is useful for network security testing. However, we are only interested in its device discovery capabilities. Figure 5.6 below shows an example of nmap discovering the IP addresses of the two devices connected to our mesh network.

```
pi@raspberrypi:~ $ sudo nmap -sn 192.168.10.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2022-05-17 23:29 PDT
Nmap scan report for 192.168.10.1
Host is up (0.0044s latency).
MAC Address: 06:DA:35:E1:DF:96 (Unknown)
Nmap scan report for 192.168.10.3
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 5.26 seconds
```

Figure 5.6: Nmap discovering devices in mesh network

The -sn flag tells nmap to ping all possible IP addresses in the provided subnet and return the IP addresses of the devices that responded to the ping. In a mesh network where all devices were given a static IP, this tool provides a way to verify that all devices joined the mesh network with the expected IP addresses. If a DHCP server is used to dynamically assign IP addresses in a large swarm, using nmap provides both a way to verify all the devices joined the network and a way to find the IP addresses assigned to each device.

5.9 Swarm Control

In the previous sections we discussed the necessary hardware and network infrastructure to support our real-time object detection drone swarm. In this section, we will discuss how to monitor and control the swarm.

5.9.1 MAVLink Protocol

The MAVLink protocol [43] defines the structure and the type of messages required to communicate with the drone's flight controller from a GCS to retrieve drone status information and provide flight commands. This protocol is optimized for UAV communications as it is lightweight and can be transmitted over low bandwidth networks. MAVLink is natively supported by QGroundControl and the ArduPilot autopilot software used in our PixHawk 4 flight controllers.

5.9.2 Companion Computer Configuration

With MAVLink handling the content and structure of the messages used to communicate with the drones, communication channels must be established between the GCS and every flight controller in the swarm to transmit such messages. The GCS has direct access to the Raspberry Pis mounted on the drones through the mesh network, but not to the flight controller. Therefore, the Raspberry Pis must be configured to route incoming MAVLink messages from the GCS to their respective flight controller. This can be achieved with MAVLink Router [44].

MAVLink Router is a program that can be executed on Linux computers to route incoming MAVLink messages to other UART, TCP, or UDP endpoints available to the computer. In our case, we are interested in creating a bi-directional channel

between the Raspberry Pi's UART connection to the flight controller and the TCP connection established with the GCS through the mesh network. To achieve this, the configuration file shown in Figure 5.7 was created. This file was then used to create a *systemd* service that is executed on system startup. This way, the RPi is automatically configured to forward MAVLink messages from the GCS to the flight controller and vice-versa.

Figure 5.7: Configuration file for MAVLink Router

The text under the "General" section is used to configure a TCP server with port number 5760 on the RPi that listens for incoming connections. When QGroundControl requests a connection to port 5760 on the RPi's IP address, the GCS becomes an endpoint to MAVLink Router. The "UartEnpoint" section establishes the UART pins of the RPi (which should be connected to the telemetry port of the Pixhawk 4) as another endpoint with baud rate of 56700. This effectively bridges the TCP and UART endpoints. A baud rate of 56700 is used for the UART endpoint as this is the default telemetry port baud rate on the Pixhawk 4. Port 5760 is the default port number used by QGroundControl for TCP telemetry links.

5.9.3 Using QGroundControl

5.9.3.1 Connecting to Multiple Drones

As part of the MAVLink protocol, MAVLink compatible devices in the network are given a MAVLink System ID to enable unicast messages. By default, the GCS running QGroundControl has a system ID of 255 and flight controllers have a system ID of 1. To avoid ID collisions in a multi-drone swarm, each drone should be given a unique system ID. If multiple GCSs are used to monitor the swarm, each GCS should also be given a unique system ID. The MAVLink system IDs of the GCS and flight controllers are configurable through QGroundControl.

To connect to the drones over the mesh network from QGroundControl, we must provide the IP addresses of the RPis to QGroundControl. This can be done in the *Application Settings: Comm Links* tab of QGroundControl. Here, we can add a communication link for each drone, as shown in Figure 5.8.

When adding a communication link to reach a drone's companion computer, a link name, link type, and additional link parameters must be provided. From the previous section, we know that the MAVLink Router instances running on the RPis were configured with a TCP server listening on port 5760. Therefore, the communication links on QGroundControl must be configured as TCP links to port 5760 with a server address corresponding to the IP address of the RPi we wish to connect to. When a communication link is enabled in QGroundControl, the GCS will attempt to reach the drone's flight controller by sending MAVLink messages through the mesh network to the TCP server established by MAVLink Router on the Raspberry Pi. MAVLink Router will accept the new connection to the TCP server, route the received messages to the UART endpoint connected to the flight controller, and deliver the messages

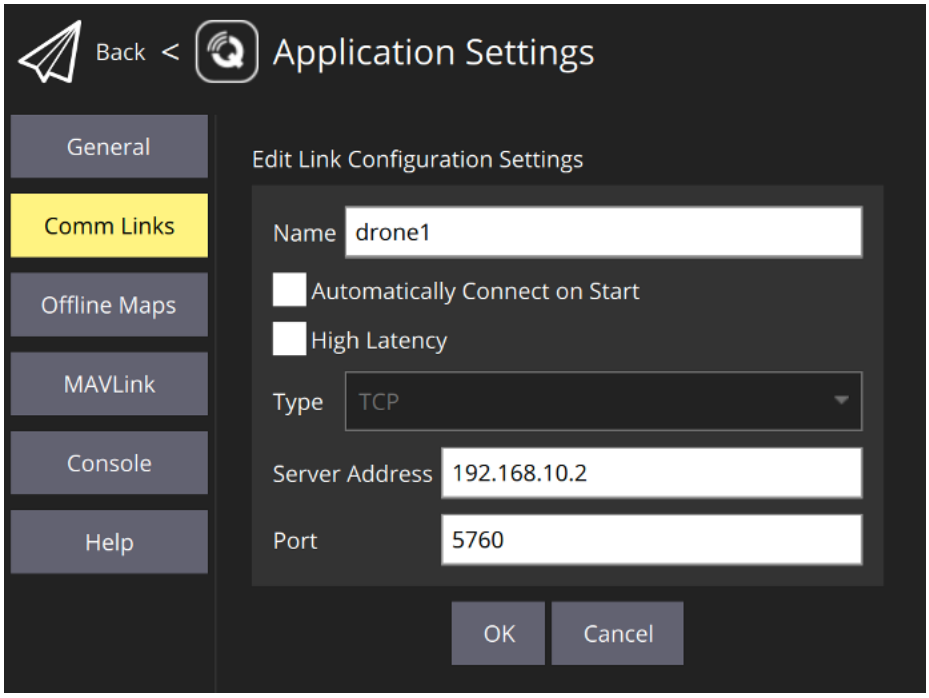

Figure 5.8: TCP link configuration in QGroundControl

coming from the GCS. When the flight controller responds to the GCS, MAVLink Router will send the flight controller's messages to the IP address of the GCS through the mesh network.

5.9.3.2 Mission Planning

A common feature for GCS software is the ability to upload preplanned flight paths to the flight controller for autonomous flights. After a mission is uploaded to the flight controller, the user can command the drone to start the mission and the drone will fly to a set of predetermined waypoints. A useful feature for aerial surveillance applications in QGroundControl is the "Survey" flight pattern, shown in Figure 5.9. In Survey planning mode, the user can provide a circular or polygonal shape outlining the target area to cover, and QGroundControl will automatically determine the waypoints the drone should follow to sweep the target area.

Figure 5.9: Mission planning with QGroundControl's Survey mode

In a multi-drone swarm, each drone may be given an independent preplanned flight path, as shown below in Figure 5.10.

Figure 5.10: Mission planning for multiple drones with QGroundControl

While planning missions for single drones and drone swarms, there are few considerations to consider.

• Mission Time: A mission cannot exceed the estimated flight time supported by the battery. QGroundControl assists with this aspect by providing an estimate of the time required to cover all the waypoints in a mission.

• Communication Range: In a single drone application, the range of the mission is limited by the direct communication range supported by the radio mounted on the drone. In our drone swarm implementation, the mesh network allows for multi-hop communication. However, to take advantage of message forwarding through the mesh network, drones must be positioned in a way that allows for intermediate drones to reach the source and the destination of the message. Therefore, when planning a long-range mission, attention must be paid to the position of the drones with respect to each other to ensure there is always a path between the GCS and every drone in the swarm.

• Overlapping Flight Paths: As the current drone design does not include collision avoidance hardware, the user should avoid crossing the flight paths of different drones. This should be simple as QGroundControl provides visualizations of the paths to be followed by all the drones in the swarm, as seen in Figure 5.10. If different drones must fly over the same area, each drone's mission should be configured at a different altitude to minimize the risk of collisions.

Chapter 6

TESTING/VALIDATION

In this section, we will show the results of testing our selected hardware and network implementation. Specifically, we tested the network range and throughput with different configurations, as well as the power consumption of a drone over a typical aerial surveillance flight pattern.

6.1 Omnidirectional Antenna Gain

The first experiment conducted was to test the differences in network range and throughput between two devices by using 10dBi and 5dBi gain omnidirectional antennas. As mentioned in Section 4.2.4.2, higher gain antennas can offer longer range communications by narrowing down the direction in which the signal is transmitted. Typically, as the gain of omnidirectional antennas increases, the size of the antenna also increases. Therefore, the goal of this experiment was to determine if the increased range offered by a 10dBi antenna was enough to compensate for the added weight and inconvenient size of the antenna.

The experiment was conducted in an open space with direct line of sight between the devices. Both antennas were positioned at the same altitude and oriented to transmit their signal along the same plane. To obtain the bandwidth available between the two antennas, the iPerf tool was used [45]. iPerf is a command-line tool that allows for testing the maximum bandwidth available between devices on a Local-Area Network (LAN), such as our mesh network. Bandwidth measurements were taken at various distances in 50ft increments until iPerf failed to estimate the bandwidth due to weak

signal strength. At each distance, iPerf was run for 50 seconds and the average throughput over the 50 seconds was recorded. Figure 6.1 below shows the results of the experiment.

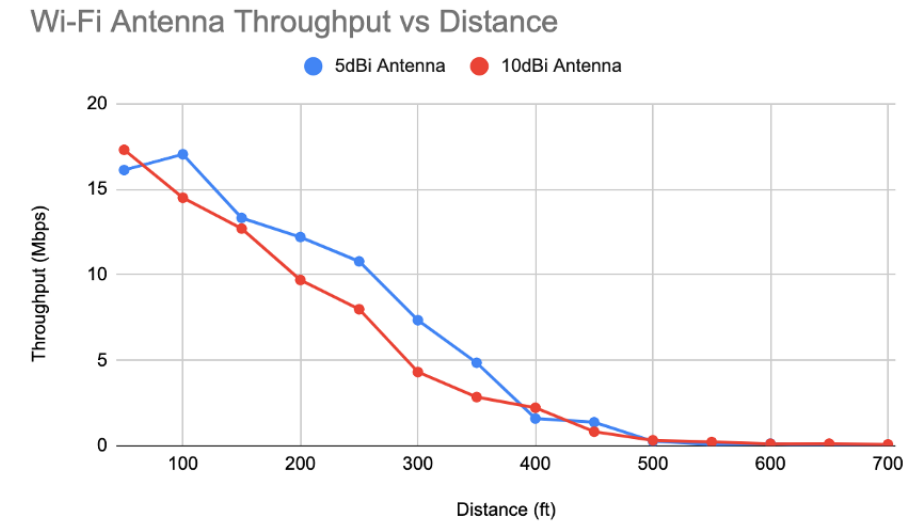

Figure 6.1: Recorded network throughput at varying distances for 5dBi and 10dBi omnidirectional antennas.

Figure 6.1 shows that both antennas had similar performance, with the lower gain antenna offering slightly better throughput at distances under 500ft. In terms of max range, the 10dBi antennas only offered an additional 100ft of range over the 5dBi antennas.

While the higher gain antenna did provide a longer communication range as expected, the additional range obtained was not significant. Also, the throughput offered by the lower gain antenna was higher on average. For these reasons, we considered the advantages of a 10dBi antenna over a 5dBi antenna to be insufficient to offset the cost of mounting a larger and heavier 10dBi antenna on our quadcopters. Therefore, 5dBi omnidirectional antennas were mounted on our drones.

6.2 Network Performance During Flight

To verify the performance of our network implementation with our selected hardware, we measured the network throughput - again using iPerf - while flying our drones at an altitude of 150ft, which is a suitable altitude for object detection applications. All distance values provided in the upcoming test results represent horizontal distance from the GCS. For all tests, the drones used a Raspberry Pi 4 with a PAU06 USB Wi-Fi adapter and 5dBi omnidirectional antenna to connect to the network.

6.2.1 GCS Omnidirectional Antenna

The first set of tests were executed by flying a single drone at a time. These tests aimed to measure the throughput and max range of a direct link from the GCS to a drone. To prevent telemetry data from affecting throughput measurements, the drone was controlled using SiK telemetry radios instead of using the mesh network. Table 6.1 and Figure 6.2 below show the test results when using a PAU06 USB Wi-Fi adapter with a 5dBi omnidirectional antenna at the GCS.

Table 6.1: Throughput measurements during single drone flight with omnidirectional antenna at GCS.

Distance (ft)	Throughput (Mbps)
200	5.75
400	3.84
600	0.0692
800	0.0749
1000	0
1200	0
1400	0

The results show that omnidirectional antennas perform well up to about 400ft, at which point the network throughput significantly drops. The last successful throughput measurement was at 800ft with a reading of 74.9 Kbits/sec. Connection to

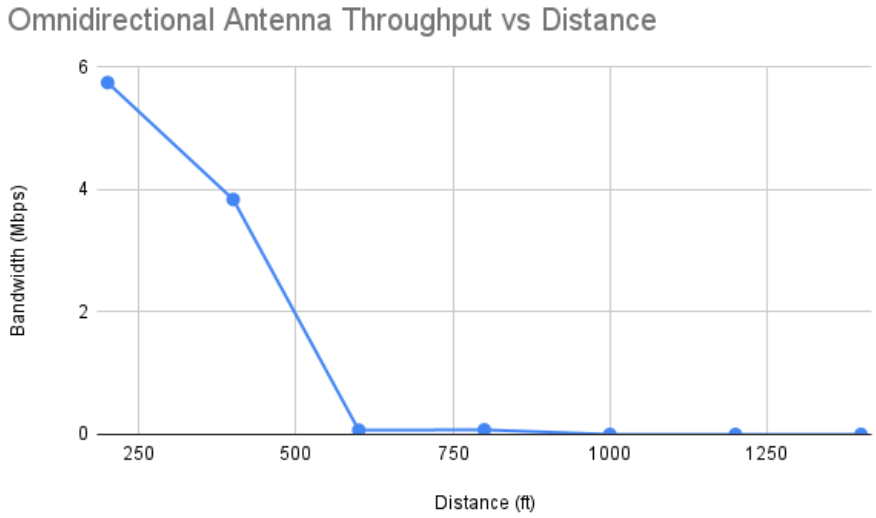

Figure 6.2: Graph representation of throughput measurements during single drone flight with omnidirectional antenna at GCS.

QGroundControl was lost at a distance of about 900ft when using the mesh network for telemetry data. This can be considered the maximum range for vehicle control.

6.2.2 GCS Directional Antenna

The same test was repeated after replacing the 5dBi omnidirectional antenna for a 13dBi directional panel antenna on the PAU06 adapter at the GCS. Results are shown below in Table 6.2 and Figure 6.3.

Table 6.2: Throughput measurements during single drone flight with directional panel antenna at GCS.

Distance (ft)	Throughput (Mbps)
200	6.92
400	3.92
600	1.15
800	0.105
1000	0.0912
1200	0.0511
1400	0.1295

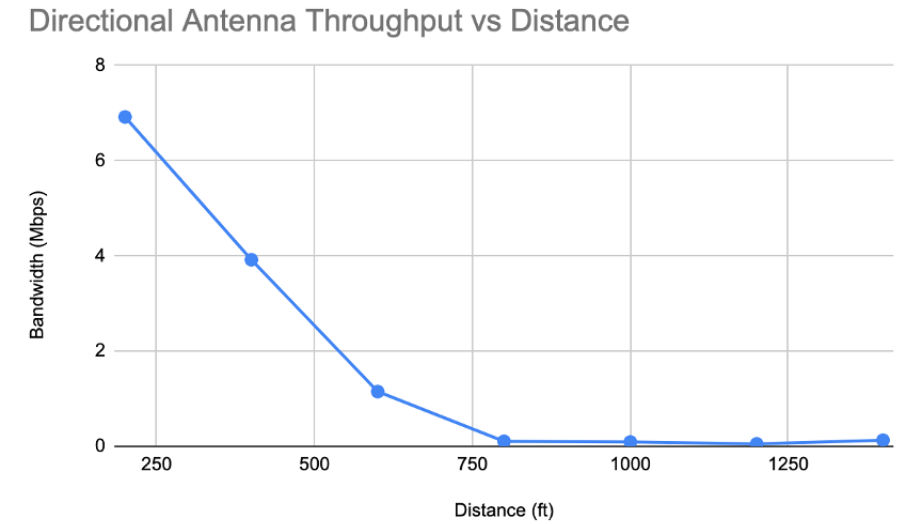

Figure 6.3: Graph representation of throughput measurements during single drone flight with directional panel antenna at GCS.

The results show that directional antennas are much better suited for longer range flights and exhibit a less dramatic decrease in throughput with distance. At every measured distance, the throughput for the directional antenna was higher than the throughput for the omnidirectional antenna. With the directional antenna, iPerf successfully measured the throughput at distances of up to 1400ft, as opposed to 800ft for the omnidirectional antenna. In terms of maximum range for vehicle control, the drone lost connection to QGroundControl at about 1000ft when using the mesh network for telemetry data. This is an increase in max range of only 100ft compared to the omnidirectional antenna. It is important to note that connection to directional antennas is highly affected by the orientation of the antenna, as explained in Section 4.2.4.1. At long distances, a drone can easily lose connection to the GCS if the antenna is not aimed directly at the drone. This is a possible explanation for the disparity in max range between telemetry data and iPerf measurements.

6.2.3 SiK Radio

To compare the range of Wi-Fi radios to the range of low-bandwidth SiK telemetry radios, an additional flight test was performed. In this test, a single drone was flown at an altitude of 150ft and was commanded to fly in a straight line with stops every 200ft. In our experiment, the SiK telemetry radio maintained a connection to the GCS at a maximum distance of 3400ft. This shows that although the throughput of SiK radios is insufficient for anything other than telemetry data, they serve as a great backup communication channel to maintain control of drones that become disconnected from the mesh network.

6.2.4 Multi-Hop Communication

Lastly, the mesh network's multi-hop communication was tested using a two-drone swarm and an omnidirectional 5dBi antenna at the GCS. For this experiment, the throughput to the farthest drone was recorded while the other drone was positioned at the halfway point between the GCS and the farthest drone at the same altitude. The SiK radios were not utilized as having two SiK radios at the GCS caused issues while trying to connect to two drones simultaneously. Instead, the mesh network was used to transfer the telemetry data. This approach is not ideal as transmitting telemetry data and measuring throughput both require network bandwidth, which leads to decreased performance for both operations if performed simultaneously. Table 6.3 below shows the results of this experiment.

At distances of 400ft and 500ft, we can see the throughput is lower than in previous tests, likely as a result of transmitting telemetry data over the mesh network while running iPerf. At 600ft, Table 6.3 includes the throughput measured over multiple executions of iPerf. In the first four throughput measurements at 600ft, the values

Table 6.3: Throughput measurements to farthest drone in two-drone swarm. Multiple throughput measurements are provided at 600ft to showcase the change in throughput when packets are routed through an intermediate node to reach the GCS.

Distance (ft)	Throughput (Mbps)
400	1.19
500	0.0873
600	0.241
	0.469
	0.0982
	0.0948
	1.24
	2.78
	1.97
	0.994

recorded show a low throughput (226 Kbits/sec on average), which is expected based on our previous single drone test results. However, the last four measurements show a significant increase in throughput at the same distance of 600ft, with the highest reading being 2.78 Mbits/sec. This illustrates the potential of multi-hop communications through our mesh network. On the first four readings at 600ft, the farthest drone was still within direct range of the GCS, so the intermediate drone was likely not being used to relay messages, which explains the results similar to the single drone tests. The last four results were recorded after connection to the GCS was lost momentarily. The HWMP protocol initiates a route lookup process when a path to the destination is not found, in this case a path to the GCS. The readings suggest that the better path through the intermediate drone was found as a result of the route lookup process, explaining the noticeable increase in throughput.

Tests at farther distances were not recorded as running iPerf affects telemetry data and causes the drone to disconnect from QGroundControl. Without telemetry data, we could not accurately estimate the distance of the drone from the GCS.

6.3 Power Consumption

To verify the accuracy of our power consumption and flight time estimate calculations, we commanded a single drone to follow a preplanned flight path with a flight pattern typical of aerial surveillance applications, shown in Figure 6.4. We then analyzed the logs recorded by the flight controller to extract power consumption data using UAV Log Viewer [46].

Figure 6.4: Flight pattern executed to measure power consumption and flight time of drone.

For this experiment, the Raspberry Pi was mounted on the drone, but not the Jetson Nano. Throughout the flight, the Raspberry Pi transmitted a TCP video stream down to the GCS using the Wi-Fi mesh network. Based on this configuration, we estimate the power draw from on-board peripherals to be about 5W. Also, since the Jetson Nano was not mounted, the estimated weight of the entire drone is reduced from 1611g (calculated in Table 5.1) to about 1473g. The flight mission lasted 18 minutes and 16 seconds with an average current draw of 10.97A and total discharge

of 3430mAh. Power consumption during flight is illustrated below in Figures 6.5 and 6.6.

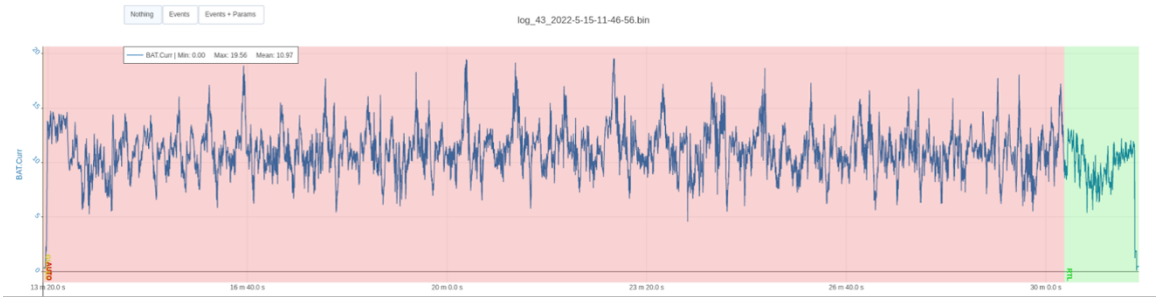

Figure 6.5: Plot of current consumption during experimental aerial surveillance flight mission.

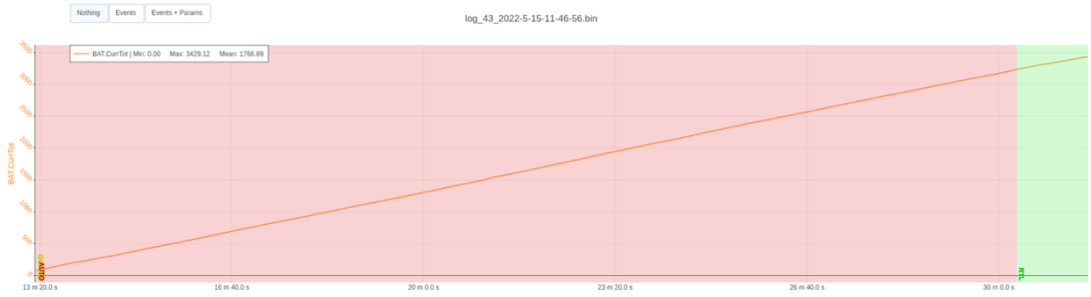

Figure 6.6: Plot of total battery discharge over time for experimental aerial surveillance flight mission.

Based on our assumption that 5W were consumed by the on-board peripherals, at an average battery voltage of 15.8V, 5W translates to 5W / 15.8V = 0.316A. This means that the four motors were cumulatively drawing 10.97A - 0.316A = 10.654A on average, or 2.664A per motor. By extrapolating the thrust to current conversions provided in Table 5.2 for our specific motors, 2.664A corresponds to approximately 355g of thrust per motor, or a total thrust of 1420g for all four motors.

The total thrust estimate of 1420g is slightly below our estimated drone weight of 1473g. However, both the drone weight and the total thrust are estimates calculated based on the extrapolation of known data, therefore a slight margin of error is expected. Even then, the estimates demonstrate that using an average thrust equal to the weight of the drone is a good measure for calculating power consumption and

flight time estimates for the flight patterns of aerial surveillance applications. Therefore, we can conclude that our flight time estimate of about 17 minutes when using all hardware and software components of our drone design is relatively accurate.

Chapter 7

CONCLUSION AND FUTURE WORK

This paper has outlined the design considerations and implementation of a drone swarm for real-time object detection applications. The proposed design features low-cost components only to make it accessible for other researchers interested in continuing work in the field of intelligent drone swarms. At the time of writing, the total cost of all components mounted on the drone along with the drone body and 5000mAh LiPo battery remained under $800 USD. This cost is comparable to that of consumer level drones in the market, but the capabilities of our drone design are extremely customizable and can match or exceed those of consumer level drones. For example, consumer level drones within this price range do not allow for customization of the object detection model, limiting the accuracy of detection and the types of objects supported. The on-board Raspberry Pi provides sufficient processing power for handling autonomous behavior, swarm coordination, and networking, while the Jetson Nano can handle a variety of machine learning models for real-time object detection. Our test results demonstrated that the suggested network implementation is sufficient for mid-range flights and supports bandwidth intensive data such as live video streams. Additionally, the mesh network architecture provides the necessary drone-to-drone communication channels for advanced swarm coordination behavior. While the current design is capable of flying multiple drones over desired areas using preplanned flight paths to detect objects using the Jetson Nano, there is still plenty of work to be done for both functionality and user friendliness purposes.

7.1 QGroundControl Limitations

The GCS software, QGroundControl, already offers the necessary functionality to control multiple drones simultaneously through the mesh network. However, there are limitations that could be worked on to improve the user interface and the performance of the drone swarm. QGroundControl supports receiving video streams from the drones over TCP, UDP and RTSP. However, it does not support multiple simultaneous video streams. A user interface that allows for enabling/disabling video streams over RTSP from all drones in the swarm would be beneficial.

Another possible improvement consists of adding support for multi-drone mission planning that takes into account mesh network limitations. This could be done by providing the mission planning interface the direct point-to-point communication range of the Wi-Fi radios and programming the GCS software to warn the user if a flight mission could cause a loss of communication. This feature would have to take into account the flight speed of all drones and simulate the mission to determine which drones could fly out of communication range. Furthermore, this feature could be expanded to support automatic generation of flight paths for the entire swarm that guarantee reliable connectivity and efficient surveillance of a shared area.

Lastly, QGroundControl does not offer any user interface features for object detection applications. A possible feature is GPS tagging of detected objects of interest. For example, in our target shark spotting application, it would be beneficial to mark the GPS location of detected sharks on the map shown by the GCS. For other applications, different objects of interest supported by the object detection model could be selected to be shown on the map. QGroundControl is an open-source software that could be modified to provide these features. Otherwise, a separate application could be developed to run at the GCS in parallel with QGroundControl.

7.2 Object Detection Optimization

The Jetson Nano is a powerful neural accelerator, but it still suffers from limitations typical of embedded devices, such as limited processing power and limited available memory. As discussed earlier in this paper, the lack of resources leads to a trade-off between accuracy and inference time in object detection models. Since drones are highly mobile, objects of interest captured by a drone's on-board camera may only remain within the field of view of the camera for short periods of time. Increasing the altitude of the drone can increase the amount of time an object at ground level remains within field of view, but increasing the altitude also decreases the size of the object within the frame and makes detection more difficult. This means that the inference time offered by the object detection model running on the Jetson Nano cannot be too slow, otherwise the object might be completely missed during flight. Decreasing the inference time typically involves using a simpler object detection model or decreasing the resolution of the input images, both of which affect accuracy. Therefore, future research could focus on achieving an optimal balance between flight altitude, flight speed, video resolution, and object detection model complexity to maximize detection accuracy of the on-board Jetson Nano while still surveying the target area in a timely manner.

7.3 Mesh Network on Large Swarms

For this project, a topology-based routing protocol was used to route messages through our swarm network. The extensive body of research on topology-based protocols led us to select HWMP as our preferred protocol. However, the field of routing protocols is broad and there is still a need to test alternative approaches outside of the topology-based category in swarm applications. The proposed drone design with the

802.11s compatible hardware already provides the underlying mechanism to enable multi-hop communications, making our design an ideal platform for testing different routing protocols in real-world applications. At the time of writing, the chip shortage caused by the COVID-19 pandemic made acquisition of Raspberry Pis and Jetson Nanos difficult, limiting the number of drones in our swarm. While we were able to test the network performance in a two-hop network, this swarm size does not properly test the routing protocol's performance for networks with many nodes and many possible paths. Therefore, it would be beneficial to test our proposed network implementation, as well as alternative routing protocols, on a larger drone swarm.

7.4 Swarm Coordination Implementation

The proposed design provides the necessary communication infrastructure and onboard processing power to handle intelligent cooperative behavior. While this paper does not cover implementation of autonomous behavior, the provided platform may be used by other researchers to develop and test their cooperative behavior algorithms. To begin with, an intelligent drone swarm would require a membership protocol that allows all drones to keep track of each other's status, location, and other information shared by the swarm. This shared information could then be used for applications such as autonomously monitoring a shared area, autonomously creating swarm formations that optimize network performance for a specific node or set of nodes in the network, tracking a set of objects with multiple drones simultaneously, etc. An initial cooperative behavior that would greatly benefit the swarm would be collision avoidance. Specialized drone collision avoidance hardware is expensive, but an alternative approach is possible using only the GPS modules on the drones. Through the membership protocol, members of the swarm may share their GPS location and heading with the rest of the swarm. If two or more drones detect that their paths will

intersect, they should agree on different flying altitudes that allow them to continue moving forward while avoiding a drone-to-drone collision. Colliding with other types of objects during aerial surveillance missions is unlikely as the drones typically fly at high altitudes. As we can see, there are a plethora of possible behaviors and applications for intelligent drone swarms that are still under development and our swarm platform provides a low-budget system that makes swarm technology accessible to future researchers.

BIBLIOGRAPHY

[1] Muhammad Asghar Khan, Ijaz Mansoor Qureshi, and Fahimullah Khanzada. A Hybrid Communication Scheme for Efficient and Low-Cost Deployment of Future Flying Ad-Hoc Network (FANET). *Drones*, 3(1):16, March 2019. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/2504-446X/3/1/16, doi:10.3390/drones3010016.

[2] Jetson Nano Brings AI Computing to Everyone, March 2019. Accessed on 2022-05-25. URL: https://developer.nvidia.com/blog/jetson-nano-ai-computing/.

[3] S500 V2 Kit. Accessed on 2022-05-17. URL: https://shop.holybro.com/s500-v2-kit_p1153.html.

[4] Real-time object detection with YOLO. Accessed on 2022-05-26. URL: https://machinethink.net/blog/object-detection-with-yolo/.

[5] Jun Li, Yifeng Zhou, and Louise Lamont. Communication architectures and protocols for networking unmanned aerial vehicles. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 1415–1420, December 2013. ISSN: 2166-0077. doi:10.1109/GLOCOMW.2013.6825193.

[6] Anand Nayyar. Flying Adhoc Network (FANETs): Simulation Based Performance Comparison of Routing Protocols: AODV, DSDV, DSR, OLSR, AOMDV and HWMP. In *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, pages 1–9, August 2018. doi:10.1109/ICABCD.2018.8465130.

[7] What Is Antenna Gain | NetXL. Accessed on 2022-03-15. URL: https://www.netxl.com/blog/networking/antenna-gain.

[8] Mohamed Abdelkader, Samet Güler, Hassan Jaleel, and Jeff S. Shamma. Aerial Swarms: Recent Applications and Challenges. *Current Robotics Reports*, 2(3):309–320, September 2021. doi:10.1007/s43154-021-00063-4.

[9] Little Ripper: The Aussie AI-powered drones providing an ethical shark deterrent, February 2022. Accessed on 2022-05-25. URL: https://www.smartcompany.com.au/startupsmart/startupsmart-technology/ai-drones-ethical-shark-deterrent/.

[10] Mission Planner Home — Mission Planner documentation. Accessed on 2022-05-25. URL: https://ardupilot.org/planner/.

[11] QGC. Accessed on 2022-05-25. URL: http://qgroundcontrol.com/.

[12] MAVProxy — MAVProxy documentation. Accessed on 2022-05-25. URL: https://ardupilot.org/mavproxy/#.

[13] The Battle of Speed vs. Accuracy: Single-Shot vs Two-Shot Detection Meta-Architecture, March 2020. Accessed on 2022-05-26. URL: https://clear.ml/blog/the-battle-of-speed-accuracy-single-shot-vs-two-shot-detection/.

[14] Qiannan Cui, Peizhi Liu, Jinhua Wang, and Jing Yu. Brief analysis of drone swarms communication. In *2017 IEEE International Conference on Unmanned Systems (ICUS)*, pages 463–466, October 2017. doi:10.1109/ICUS.2017.8278390.

[15] A Comparative Performance Evaluation of Routing - ProQuest. Accessed on 2022-01-25. URL: https://www.proquest.com/docview/2532423345.

[16] Alexey V. Leonov. Modeling of bio-inspired algorithms AntHocNet and BeeAdHoc for Flying Ad Hoc Networks (FANETs). In *2016 13th International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, volume 02, pages 90–99, October 2016. doi:10.1109/APEIE.2016.7806419.

[17] Cristian Ramirez-Atencia and David Camacho. Extending QGroundControl for Automated Mission Planning of UAVs. *Sensors (Basel, Switzerland)*, 18(7):2339, July 2018. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6068744/, doi:10.3390/s18072339.

[18] Alessandro Montanari, Fredrika Kringberg, Alice Valentini, Cecilia Mascolo, and Amanda Prorok. Surveying areas in developing regions through context aware drone mobility. In *Proceedings of the 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, DroNet'18, page 27–32, New York, NY, USA, 2018. Association for Computing Machinery. URL: https://doi-org.ezproxy.lib.calpoly.edu/10.1145/3213526.3213532, doi:10.1145/3213526.3213532.

[19] Rob Matheson. Fleets of drones could aid searches for lost hikers. *MIT News*, Nov 2018. URL: https://news.mit.edu/2018/fleets-drones-help-searches-lost-hikers-1102.

[20] Erik Kuiper and Simin Nadjm-Tehrani. Mobility Models for UAV Group Reconnaissance Applications. In *2006 International Conference on Wireless and Mobile Communications (ICWMC'06)*, pages 33–33, July 2006. doi:10.1109/ICWMC.2006.63.

[21] Wei Wang, Xiaohong Guan, Beizhan Wang, and Yaping Wang. A novel mobility model based on semi-random circular movement in mobile ad hoc networks. *Information Sciences*, 180(3):399–413, February 2010. URL: https://www.sciencedirect.com/science/article/pii/S0020025509004265, doi:10.1016/j.ins.2009.10.001.

[22] Sabir Hossain and Deok-jin Lee. Deep Learning-Based Real-Time Multiple-Object Detection and Tracking from Aerial Imagery via a Flying Robot with GPU-Based Embedded Devices. *Sensors*, 19(15):3371, January 2019. Number: 15 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/1424-8220/19/15/3371, doi:10.3390/s19153371.

[23] Bandwidth calculator | CCTV Calculator. Accessed on 2022-04-30. URL: https://www.cctvcalculator.net/en/calculations/bandwidth-calculator/.

[24] S.M. Faccin, C. Wijting, J. Kenckt, and A. Damle. Mesh WLAN networks: concept and system design. *IEEE Wireless Communications*, 13(2):10–17, April 2006. Conference Name: IEEE Wireless Communications. doi:10.1109/MWC.2006.1632476.

[25] Guido R. Hiertz, Dee Denteneer, Sebastian Max, Rakesh Taori, Javier Cardona, Lars Berlemann, and Bernhard Walke. IEEE 802.11s: The WLAN Mesh Standard. *IEEE Wireless Communications*, 17(1):104–111, February 2010. Conference Name: IEEE Wireless Communications. doi:10.1109/MWC.2010.5416357.

[26] o11s/open80211s, April 2022. original-date: 2011-07-28T21:36:15Z. URL: https://github.com/o11s/open80211s.

[27] Taemin Ahn, Jihoon Seok, Inbok Lee, and Junghee Han. Reliable Flying IoT Networks for UAV Disaster Rescue Operations. *Mobile Information Systems*, 2018:e2572460, August 2018. Publisher: Hindawi. URL: https://www.hindawi.com/journals/misy/2018/2572460/, doi:10.1155/2018/2572460.

[28] Michael Bahr. Update on the Hybrid Wireless Mesh Protocol of IEEE 802.11s. In *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 1–6, October 2007. ISSN: 2155-6814. doi:10.1109/MOBHOC.2007.4428721.

[29] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999. doi:10.1109/MCSA.1999.749281.

[30] The open standards for drone hardware. Accessed on 2022-05-26. URL: https://pixhawk.org/.

[31] ArduPilot. Ardupilot/ardupilot: Arduplane, arducopter, ardurover, ardusub source. Accessed on 2022-05-26. URL: https://github.com/ArduPilot/ardupilot.

[32] PX4. Px4/px4-autopilot: Px4 autopilot software. Accessed on 2022-05-26. URL: https://github.com/PX4/PX4-Autopilot.

[33] HOWTO · o11s/open80211s Wiki. Accessed on 2022-05-16. URL: https://github.com/o11s/open80211s.

[34] en:users:drivers [Linux Wireless]. URL: https://wireless.wiki.kernel.org/en/users/Drivers.

[35] Blog: How to Choose an Antenna. Accessed on 2022-03-15. URL: https://doodlelabs.com/about-us/blog/how-to-choose-an-antenna/.

[36] TensorFlow models on the Edge TPU. Accessed on 2022-05-25. URL: https://coral.ai/docs/edgetpu/models-intro/#compatibility-overview.

[37] DroneKit. Accessed on 2022-05-17. URL: http://dronekit.io.

[38] SiK Telemetry Radio — Copter documentation. Accessed on 2022-05-17. URL: https://ardupilot.org/copter/docs/common-sik-telemetry-radio.html.

[39] Power Module(PM02 V3). Accessed on 2022-05-17. URL: http://www.holybro.com/product/power-modulepm02-v3/.

[40] A Guide to Understanding LiPo Batteries. Accessed on 2022-05-17. URL: https://rogershobbycenter.com/lipoguide.

[41] q-a-basic-wireless-concepts. Accessed on 2022-05-18. URL: https://www.tp-link.com/us/configuration-guides/q_a_basic_wireless_concepts/?configurationId=2958.

[42] Nmap: the Network Mapper - Free Security Scanner. Accessed on 2022-05-18. URL: https://nmap.org/.

[43] Anis Koubaa, Azza Allouch, Maram Alajlan, Yasir Javed, Abdelfettah Belghith, and Mohamed Khalgui. Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey. Technical Report arXiv:1906.10641, arXiv, June 2019. arXiv:1906.10641 [cs] type: article. URL: http://arxiv.org/abs/1906.10641.

[44] MAVLink Router, May 2022. original-date: 2016-12-01T20:32:45Z. URL: https://github.com/mavlink-router/mavlink-router.

[45] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. Accessed on 2022-05-20. URL: https://iperf.fr/.

[46] UAV Log Viewer. Accessed on 2022-05-25. URL: https://plot.ardupilot.org/#/.