Project Title: Target Tracking using Drones

Project Code:

E24

Academic Supervisor:

Le Yang

Student Name:

Benjamin Ireland

Student ID Number:

93524067

Student Email:

bir16@uclive.ac.nz

Table of Contents

Executive Summary

The applications of target finding and tracking using drone swarms are numerous. Search and rescue, infrastructure monitoring, surveillance, and wireless networks are just a few of the applications of this technology. This project aims to develop a drone swarm capable of tracking endangered NZ insects for environmental preservation.

The Wireless Research Centre (WRC) is the primary stakeholder in this project's development. It is hoped that the target tracking technology developed throughout this final year project can be applied to larger drone swarms for insect tracking applications. As this technology can be used in a variety of different applications, a versatile approach must be taken throughout development.

The WRC requires a reliable, scalable, and dynamic drone swarm to complete insect tracking missions. Drones within the swarm must be able to fly in formation with other drones while communicating important flight data to their neighbours.

This project is a continuation of a previous final year projects work. Where a rigid drone formation that followed a predefined path was achieved. At this stage in the project's development a new design for the drone swarm that removes the limitations of the previous design has been developed. Using C++ and the Robot Operating System framework the new design implementation has been completed, now only requiring testing and review. Alongside this, the implementation of a network protocol for drone swarming applications is currently underway. Two applications, a management interface for the drone swarm, and a performance measurement application, are being developed for the drone swarms' pilot.

My contribution to the project lies with the improvement and redesign of the flight control node, as well as the implementation of the network protocols application programming interface (API). To complete the project, the flight control system needs to be simulated and testing in a real environment needs to be conducted. Improvements and rework to the design must then be made based on the data received from testing. The network API still requires implementation, and the network protocol must be integrated with the rest of the drone swarm. Each of these tasks must be completed to achieve the success criteria of having the drone swarm tracking a slow-moving target in the field.

1. Project overview

1.1 Purpose Statement

The purpose of this project is to further develop the Wireless Research Centre's (WRC) target tracking drone swarm technology. Potential applications of this technology are extensive and will impact key industries. This technology is being developed for tracking endangered NZ insects. Other applications include scanning for disease within cultivated plants and animals, providing support in natural disasters, and military/security surveillance. The stakeholders for this Final Year Project are the WRC sponsor (Graeme Woodward), the head of Computer Science and Software Engineering (Andreas Willig), and the Department of Conservation. The 2024 FYP team hopes to be able to coordinate at least four drones in formation tracking a slow-moving target in a field test.

1.2 Project Introduction

The WRC is one branch of the University of Canterbury's research sector that focuses on wireless communication and signal applications. They are responsible for sponsoring the drone swarm FYP and are the acting clients for this project. This FYP project is a continuation which builds upon previous final year project with a goal to create an autonomous drone swarm capable of finding, tracking, and following a target. The drones being used for this project are displayed in figure 1.

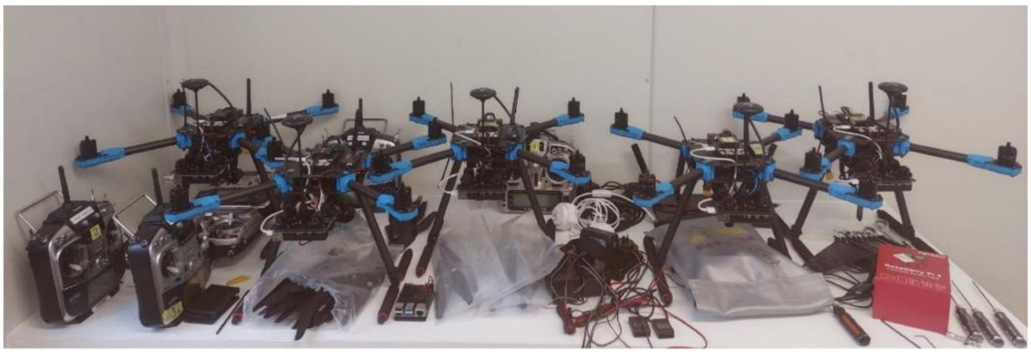

Figure 1 Fleet of HolyBro X500 Drones to be Used for the Drone Swarm

The primary application the WRC intends to use this technology for is to reliably track endangered insects. There is currently no known commercially operative drone swarm deployments in NZ as the development of this technology is relatively new. The most relatable technology which uses drone swarm communications are drone light shows. These differ from the drone swarm being developed. In these applications each drone has their flight path pre-orchestrated, whereas the drone swarm is dynamic. The previous iteration of the drone swarm depends on a leader/follower structure between drones, in which followers follow the leader along a predefined path.

The WRC requires a reliable and scalable drone swarm that accurately tracks a target and removes the rigidity of the previous implementation. This means that the drone swarm must be able to be deployed repeatedly without failure and additionally, increasing the number of drones should not require any significant changes to system software.

1.3 Project Breakdown

The overall project was partitioned into four components outlined as follows:

- **Network and Data Dissemination** – The implementation of a network protocol for ad-hoc drone communication and data dissemination throughout the drone swarm.
- **Path Planning and Trajectory Following** – Redesigning the current flight control system to improve the efficiency and scalability of system software while decoupling the swarm formation from target tracking.
- **Visualization and Diagnostics** – Development of an application that oversees the management of drone software, pre-flight diagnostics, and flight data.

- **Application and Integration** – Develop an application to track and record the movement of a target and the swarm's formation. Measuring the swarm's performance in completing tasks such as target tracking and formation changes.

These projects were divvied amongst the FYP team, resulting in the project delegation outlined in table 1. The first two projects were split into sub-projects due to the volume of work required to complete them. I was delegated to two sub-projects which have been highlighted in table 1.

Table 1 Project Delegation

Projects	Sub-project	Team Member Responsible
Network Protocol	VarDis Protocol Implementation	Raith Fullam
	Drone Coordination Protocol API	**Benjamin Ireland**
Path Planning & Trajectory Following	Drone Lifecycle Manager	Finlay Cross
	Control System	**Benjamin Ireland**
Data Visualisation & Flight Diagnostics	-	Ronniel Padua
Application & Integration	-	Se Hyun Kim

1.4 Project Implementation

The proposed solution to the current issues that are being faced is to implement a network protocol to reliably disseminate important flight data throughout the swarm. Porting the python code, the previous iteration is based on, to C++ and redesigning the flight control system to reduce coupling between swarm formation and target tracking. This is to be done by removing identified redundancies in the previous system that are preventing scalable and reliable drone swarms, and simplifying the way the swarm formation is managed. The development of diagnostic and performance-based applications will help streamline the set-up of drone swarms and the optimisation of system software. The end goal of this FYP will be accomplished when the drones can fly in a tight formation around a central moving target, communicating with other drones via an ad-hoc network.

2. Progress to Date

Due to my involvement in both the network protocol and the flight control system, each of the following sections have been partitioned into both network protocol and the flight control progress. The progress made encompasses both networking within drone swarms and the control methods they use.

2.1 Background Research

Research into other drone swarms, commercial and non-commercial, was conducted to gain an understanding of what systems already exist. Vehicular Ad-hoc Networks (VANETs) and network protocols used for vehicular applications were investigated. And for the flight control system different UAV control methods were investigated, and the previous drone swarm implementation was analysed.

2.1.1 Network Protocol Research

VANETs provide inter-vehicle communications using WLAN and cellular technologies. Some examples of VANETS using drones include [1], [2]. [1] proposes the use of a drone based VANET for propagating VANET communication along highways. In [2] a review of wireless communications amongst multi-UAV systems was conducted concluding that a drone-based communication systems are becoming increasingly popular.

The network protocol being implemented was predefined by project sponsors before the project began. The Drone Coordination Protocol (DCP), outlined in [3], explains how the drone swarm will eventually communicate. The functionality of the protocol was set; however, the protocol does not specify the exact implementation of the protocol itself. The protocol is also still in its infancy which allows for future improvements.

DCP consists of three sub-protocols:

- The Beaconing Protocol (BP).
- The State Reporting Protocol (SRP).
- The Variable Dissemination Protocol (VarDis).

BP is the overarching transmission protocol; it is used to encapsulate data from SRP and VarDis for reliable transmission. SRP manages the drone's state. Sharing information like, velocity, position, trajectory, etc. Whereas VarDis disseminates any arbitrary variable throughout drone swarm. This could be information such as drone IDs and software versions.

BP and SRP have already been implemented by previous FYP groups, but VarDis has not. The implementation of VarDis is to be conducted by Raith Fullam in this FYP. My contribution to the network protocol resides in creating an interface for DCP. This API must be able to obtain information from both the SRP and VarDis.

An investigation into what is good practice in API design was conducted, so that a robust API would be developed for DCP. In [4], maxims for good practice in API development were defined. The most important maxims from were itemised to be:

- APIs should be easy to use and hard to misuse.
- APIs should be self-documenting.
- API design is not a solitary activity.
- Documentation matters.

These claims are backed up in [5]. Both [4] and [5], discuss the importance of documentation and developing robust software if it is to be used by many different applications. These maxims need to be considered when API development takes place.

Currently the drone swarms do not communicate using DCP but a commercially available protocol, MAVLINK, described in [6]. This is done so that the drone swarm can be developed and tested independently of the DCP protocol.

2.1.2 Control Research

Before any progress could commence, the previous solution to the problem was analysed. The existing drone swarm consisted of a simple leader and follower structure as depicted in figure 2. The leader drone follows a predefined path, such as a figure of eight, communicating its trajectory to the follower drones, which correct their position based off the leader's trajectory. Each drone in the swarm establishes PID control between itself and every other drone in the swarm to maintain the overall formation. This is represented by the dashed lines in figure 2.

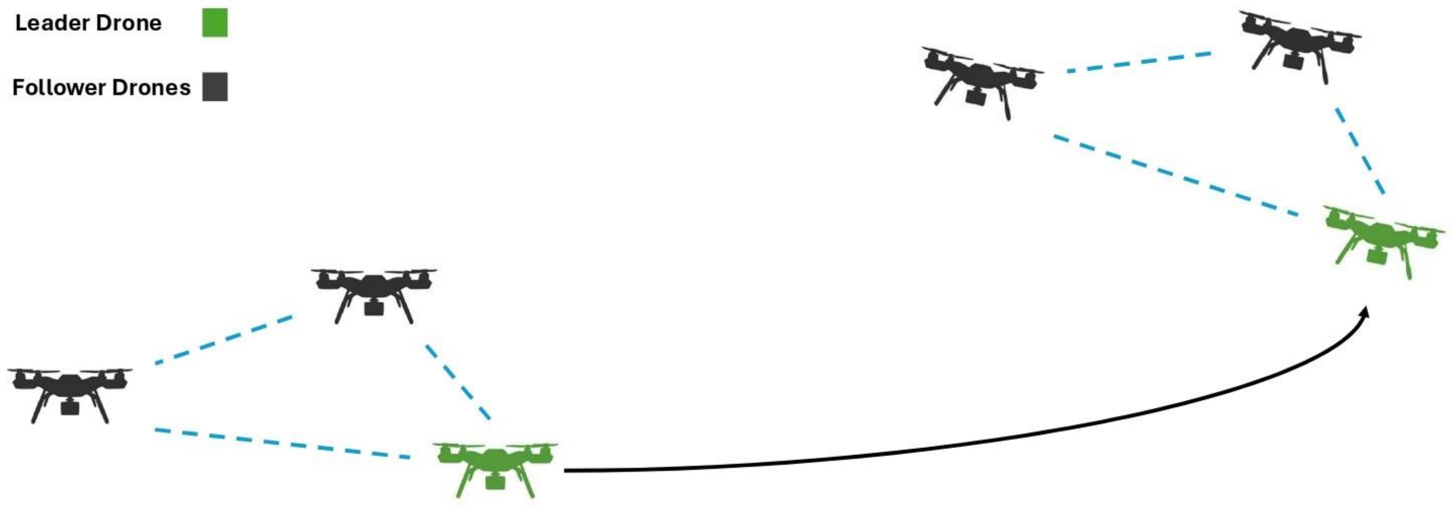

Figure 2 Depiction of Previous Drone Swarm Implementation.

Multiple flight tests of this system were conducted highlighting many problems with the current design. Drones would begin a rendezvous sequence upon launch to allow each drone to find its place within the formation, however using the previous control system drones would not be able to converge on their desired locations resulting in multiple unsuccessful flights. Figure 3 exhibits the drones stuck within the rendezvous state.

Figure 3 Drones Failing to Rendezvous

Through analysis of this design a few drawbacks were identified:

- The leader drone controls the whole swarm, representing a single point of failure.

- The leader is required to do significantly more work than the followers.
- Drones must maintain multiple control constraints to maintain swarm formation, effecting the overall scalability of the drone swarm.
- The drone swarms responsivity to changes is poor.

An investigation into different controllers was also conducted to find other methods that may improve the control of the drone swarm. In [7] a type-2 fuzzy neural network is used for trajectory control in UAVs. This was found to be more effective than traditional PID control. However, it was decided that before further investigation was to be made into this method, redefining the swarms design, and refining the PID controller of the previous implementation would be conducted.

2.2 Theoretical designs

2.2.1 API Design

The API will be used to access information from VarDis and SRP. Both protocols establish a database within the drone's memory containing the information they rely upon. The API implantation will be able to request data from both databases. In the case of VarDis the API must also be able to add new variables to the database to be disseminated throughout the drone network.

2.2.2 Control System Design

The reliance on a leader drone was undesirable and through correspondence with the project sponsors a solution that matched their desired implementation for the swarm was found. Figure 4 depicts the proposed drone swarm design.

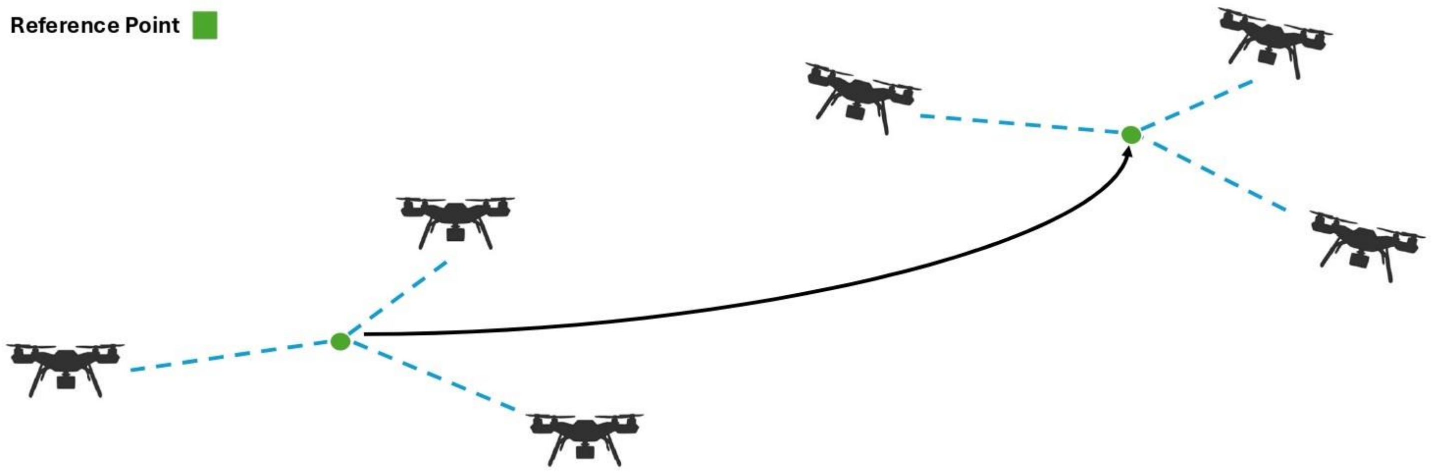

Figure 4 Proposed Drone Swarm Design.

This design removes the concept of the leader/follower structure. Instead, it allows each drone to have more independence. Each drone maintains their position in the swarm with respect to a reference point (RP). The reference point is defined as any node that communicates its position to the drone swarm. The RP is likely to be a target. In practice however, this may be a computational node that transmits its GPS location to the drones using DCP or, in following with the proposed application of the drone swarm, a harmonic signal from a tagged insect. If something were to happen to the reference point, the drone swarm would simply maintain its current flight position with the last known location of the reference point until it receives new information.

Because each drone is only focused on maintaining its position to the reference point, this simplifies the overall computation and allows for a more scalable drone swarm. Minimising the workload each drone must do, by reducing the number of control constraints it needs to worry about, will help to improve the responsivity of the drone swarm. To further improve system responsivity the software is to be ported from python to C++. As demonstrated in [8] the speed of C++ is 5-10 times faster than Python depending on the application.

An issue identified with the new system, is that each drone is essentially blind to the position of other drones within the swarm. This could result in collisions in unstable conditions. To mitigate this drones will check that they are not within a minimum distance from their neighbours, if they are then the swarm must stop or corrections to the offending drones flight paths must be made.

2.3 Software development

The previous implementation of the drone swarm utilises the Robot Operating System (ROS) framework to communicate between drones within the swarm. In the initial stages of designing the new drone swarm, it was found that the version of ROS being used was to become unsupported in early 2025. As this project is intended to be continued after this year, the decision was made to migrate the software to a supported version, from ROS 1 to ROS 2. ROS 2 also provides many additional benefits outlined in [9], such as the use of UDP over TCP in wireless communication. This change was a necessity that has delayed the progress of the network protocol but has unaffected the timeline of the control system.

2.3.1 Network API Implementation

The first task was to migrate previous implementation of the drone coordination protocol to ROS2. The previous implementation was written in C++, and because of this the migration process was relatively simple and followed a recursive process. This was my primary task for the network protocol, as Raith worked on getting a working implementation of the VarDis sub-protocol. At this time DCP has been completely migrated to ROS 2 and is currently in a buildable state, although some minor refinements need to be made to make sure that the intent of the previous implementation was carried through the translation process.

The API implementation has not yet begun due to a focus on getting the previous contributions to DCP working again using ROS 2.

2.3.2 Control System Software

The initial plan for the control system was to port the python implementation of the previous system to C++ while simultaneously migrating from ROS1 to ROS2, before implementing the new design. It quickly became apparent to the team that this was not progressing as quickly as had been hoped. The differences between the two programming languages as well as the migration to a different ROS framework made direct translation unrealistic and time consuming. Instead of this it was decided that some of the previous drone swarm's functionality would be scavenged for use in the new system, but the new software would be built from the ground up.

The ROS framework is built upon a publisher/subscriber pattern. Different processes (Nodes), communicate by publishing and subscribing to different topics. The new system establishes two primary nodes, a flight control node, and a drone lifecycle node. The control node calculates corrections to the drone's position with respect to the moving reference point, and the lifecycle node operates a simple state machine. Figure 5 shows how the two nodes communicate with each other and the rest of the system.

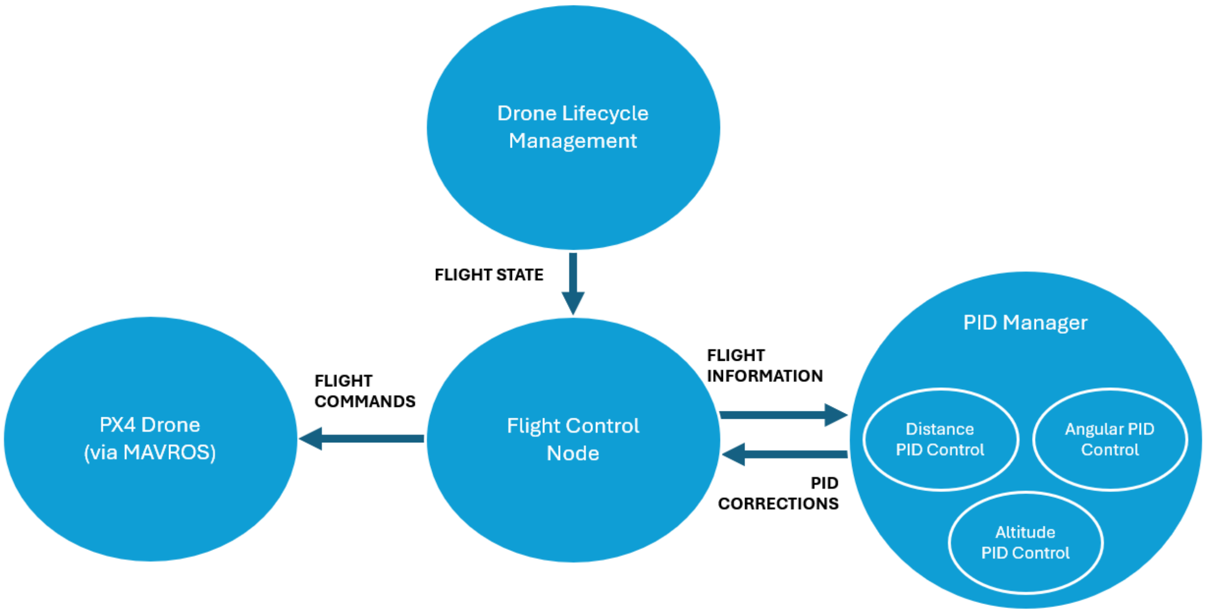

Figure 5 Flight Control System Diagram.

Communication with the drones PX4 Pixhawk flight controller is done through a series of MAVROS nodes. MAVROS is also responsible for communication between the drones using the MAVLINK protocol, although this functionality is to be replaced by DCP once it is completed. The PID manager, seen in figure 5, is a class that manages PID constraints for each drone and computes their flight corrections. The drone lifecycle node will publish the state of the drone and the flight control node will then act accordingly. Figure 6 depicts the state diagram that is used by the drone lifecycle node. State changes are currently caused by user inputted commands such as, *LAUNCH*, *PAUSE*, *RESUME* and *LAND*.

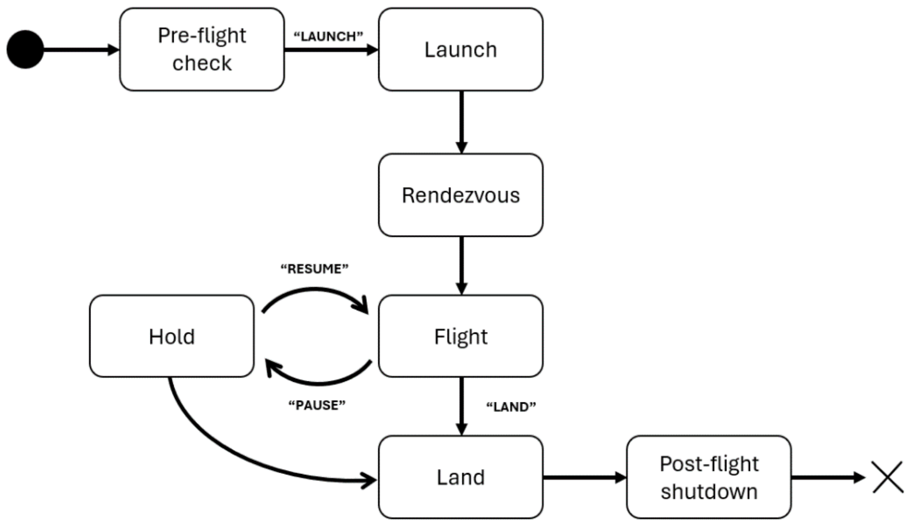

Figure 6 Drone Lifecycle State Machine

As identified in table 1 the lifecycle node was implemented by Finlay, and the control node was implemented by me.

The control node on each drone publishes the drones angular and directional velocities based on error corrections to MAVROS. From their MAVROS Informs the onboard flight controller about the adjustments that need to be made to remain in formation with respect to the reference point. There are three PID controllers established by each drone, these are identified in figure 5. The distance PID corrects the position from the drone to the RP, likewise for the altitude and angular PID controllers. The overall error vector is calculated and, using the ROS 2 control toolbox, PID corrections are calculated and passed through the control node to the PX4 flight controller.

At this stage the control system has met the end of its initial development stage and is in a buildable state, ready to be tested and simulated.

2.4 Simulations

The simulation software *Gazebo* will be used to simulate the drone swarms before in person flight tests are conducted. This software is well integrated with the ROS framework and allows for seamless simulation of the drone swarm.

2.5 Progress Summary

The development of the drone swarms control system is on schedule as the initial implementation has been completed and testing of the system can begin. The control system has been redesigned to better suit the requirements of the clients and remove the limitations of the previous drone swarm. The drone software has been rebuilt in C++ and ROS2 and has been implemented to match the design of the new drone swarm. Currently, the drone coordination protocol has been migrated from ROS1 to ROS2 and is now in a state where API development can begin. The overall progress made on both the network protocol and the aligns with the expected progress timeline. Upon completion, the drone swarm will be able to communicate using DCP, and with the aid of management and diagnostic applications complete flight missions.

3. Remaining Tasks

3.1 Timeline

The final year project is currently progressing smoothly, to be certain the project remains on track an updated timeline has been completed highlighting project deadlines. The components of the project timeline that relate to my contribution have been provided in Appendix A.

3.2 Milestones

Individual milestones for the remainder of the project are provided in table 2. These outline what tasks need to be achieved personally each fortnight to ensure that the timeline components relating to my contribution are achieved by their estimated deadline. The sub-project each milestone relates too has been stated alongside the estimated date.

Table 2 Personal Project Milestones

Personal Milestones		**Completion Date**
DCP API	**Control System**	
Complete Basic Outline of API	Complete Target Node	7th June 24
Achieve Full Functionality	Unit Test and Simulate Swarm Software in Gazebo	21st June 24
Integrate with BP and SRP	Improve Software until Confidence in Gazebo is attained	5th July 24
Integrate with VarDis	Flight Test Minimal Formation	19th July 24
Progress Inspection		**22nd - 26th July 24**
Finalise and Test DCP API	Flight Test Complex Formation	9th Aug 24
Integrate DCP with Control System		23rd Aug 24
Flight Test Completed Drone Swarm		6th Sept 24
Final Inspection		**20th Sept 24**
Prepare Final Report		4th Oct 24
Final Report Hand In		**18th Oct 24**

3.3 Budget Summary

As the drone swarm is primarily software based, and with all drones, raspberry pi's, batteries, and other necessary equipment already provided to us, the budget is minimal. At the current stage of the project nothing has been purchased. The updated project budget is provided in table 3.

Table 3 Budget for the Remainder of the Project

Item	Units	Cost/Unit	Shipping	Cost/Item
RC Truck	1	$40	$40	$80
Misc. Costs	-	$0 - $220	$20	$0 - $240
TOTAL COST	$80 - $320 (NZD)			

The RC Truck itemised in table 3, is to be used for demonstrating the drone swarms target tracking capabilities. This item is estimated to be purchased within the final weeks of the project. The miscellaneous costs are precautionary. They are to cover the cost of any potential replacement components needed for the drones in the event they are damaged. The HolyBro X500 drones are made from replaceable carbon fibre and plastic components. If a drone was to crash replacing every component of the drone's frame is estimated to cost ~$220 NZD, a figure taken from the HolyBro website [10]. Replacements are unlikely to be needed but have been added to the budget for completeness.

4. Sustainability Analysis

Understanding the impacts the drone swarm may have on the environment and society, is important for its development. Being informed of the negative impacts of the swarm give the team the opportunity to mitigate and minimise these aspects during the development process. In the sections below, both the environmental and societal impacts the drone swarm technology has at the current stage of development and the potential impacts it may have once commercialised have been considered.

4.1 Environmental Impact

Assessing the lifecycle of the drone swarm requires analysis of the inputs and outputs to the system. Looking at the environmental effects of the drones before, after, and during active duty will help us gain an understanding of their environmental impact. Climate change potential and the depletion of minerals are the primary environmental indicators for the drone swarm.

Investigating the climate change potential, we can analyse how much energy it takes to operate the drone swarm and convert this it to its CO_2 equivalent emissions. At this stage, flight time of each drone is short. It takes ~18 minutes using 5000mAh LiPo batteries, for a drone's battery to deplete. This is fine while the drone swarms are still being developed and are in the prototyping phase. However, once the drone swarms are used in the field, frequent recharging throughout missions will be inconvenient and require more energy consumption.

CO_2 emissions will vary depending on the energy mix of the electricity grid. Using New Zealand's emission factor of 0.101 kg CO_2-e / kWh from [11], one year of operation using a ten-drone swarm, completing one hour-long daily flights, would result in ~90 kg of CO_2-e emissions (the calculation of this can be found in Appendix B). This is only an estimation as losses in charging have not been accounted for. But, to put this in perspective this is less CO_2 than would be produced driving from Wellington to Auckland in a conventional petrol car [12]. Although this does not include CO_2 emitted from the production and end of life of the drone swarm, it shows that the total energy consumption for a single swarm is minimal. Regardless of this, efforts must be made during the project to ensure that software developed does not use excessive amounts of energy, draining the batteries faster and increasing the energy required to operate.

Turning to resource depletion, the HolyBro X500 drones consist of carbon fibre tubes and plates fastened using plastic connectors. The drones are not particularly robust. However, they are completely modular. All components can be replaced from the flight controller to the rotors. The plastic casings can be recycled, however, there are currently no commercially available methods for carbon fibre recycling. The batteries themselves are lithium-ion polymer batteries, these batteries are recyclable but if damaged or overcharged can ignite posing risks to public safety. It is worth investigating potential alternatives.

4.2 Social Impact

These drone swarms ultimately will be used to track endangered native insects for environmental preservation. However, adaptive drone swarms have many other applications from search and rescue, military use, and surveillance. If this drone swarm technology was to be used in these sectors, they would have a major impact on society. Three social impacts surrounding the use of commercial drones were identified in [13] to be:

- Safety and Security – Personal and of property.
- Rights to Civilian Airspace – How drones interfere with civilian aviation.
- Privacy and Ownership – In regard to data collection by drones

Relating these back to the drone swarm in development, the safety of the public when the swarm is active will be a concern. Currently, each operating drone in the swarm must have a pilot ready to take manual control, however, when these restrictions are removed in the future a crash could damage people and property. This ties into the rights to civilian airspace. Currently, all drone operators must sign a form saying they understand the Federal Aviation Associations regulations for recreational flying, before being allowed to operate the drones. These regulations can be found in [14].

In regard to privacy concerns, at this stage the drones within the swarm do not share any sensitive data. Although, once the variable dissemination protocol is integrated with the drone swarm, each drone will have the capability to share a wide range of data throughout the swarm, which will pose security threats. All these social concerns must be taken in consideration throughout development of the drone swarm to ensure public wellbeing is maintained.

5. Conclusions

The progress covered in this report encapsulates the design and development phase of the project. The background research conducted on drone based VANETs, API design, and control methods, has helped shape the drone swarm's new design. The switch from the previous leader/follower structure to the reference point focused design outlined in this report, will allow for a scalable and responsive drone swarm. The drone swarms flight control system, following the newly proposed design, has been fully implemented. The move from python to C++ during the development will decrease the response time of the swarm, improving the overall speed of the system. Using the ROS 2 framework, communication between nodes within the drone swarm has been vastly simplified reducing the overall development time.

Project progression has been steady. The drone swarms flight software will require a substantial amount of time in simulation and field testing before confidence in the system can be attained. The progress made has allowed for ample time to complete the testing phase of the project. However, minimal progress was made on the API development to date due to setbacks in the migration process from ROS 1 to ROS 2. Now that the development phase of the drone swarms flight system has been completed more resources can be put into the API development and the integration of DCP into the drone's flight system.

Investigating the social and environmental aspects of the project has highlighted that a commercialised version of this drone swarm will impact the two sectors. The environmental impacts of a small drone swarm, in terms of CO_2 emissions, were found to be minimal. However, to minimise resource depletion, steps need to be taken to find reusable, long-lasting, and recyclable materials for use in drones. The social issues for this project relating to safety, privacy, and legal have been taken in consideration throughout development to prevent endangerment to any individuals during flight tests.

All progress contributions stated in this progress report show that the project is on schedule to meet its success criteria of having the drone swarm tracking a slow-moving target in a field test.

References

[1] H. Seliem, R. Shahidi, M. H. Ahmed and M. S. Shehata, "Drone-Based Highway-VANET and DAS Service," in *IEEE Access*, vol. 6, pp. 20125-20137, 2018, doi: 10.1109/ACCESS.2018.2824839.

[2] L. Shi, N. J. H. Marcano, R. H. Jacobsen, “A review on communication protocols for autonomous unmanned aerial vehicles for inspection application”, Microprocessors and Microsystems, vol. 86, 2021, doi: 10.1016/j.micpro.2021.104340.

[3] A. Willig, “Drone Coordination Protocol (DCP) and VarDis Protocol Specification”, UC, Christchurch, Mar. 1, 2024.

[4] J. Bloch, “How to design a good API and why it matters”, In Companion to the 21st ACM SIGPLAN OOPSLA, Association for Computing Machinery, New York, NY, USA, pp. 506–507, 2006, doi:10.1145/1176617.1176622.

[5] M. Reddy, “Introduction”, in *API Design for C++*, 1st ed., USA, Elsevier, 2011, ch.1, sec.1.3, pp. 6-11.

[6] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," in *IEEE Access*, vol. 7, pp. 87658-87680, 2019, doi: 10.1109/ACCESS.2019.2924410.

[7] E. Kayacan and R. Maslim, "Type-2 Fuzzy Logic Trajectory Tracking Control of Quadrotor VTOL Aircraft With Elliptic Membership Functions," in *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 339-348, Feb. 2017, doi: 10.1109/TMECH.2016.2614672.

[8] L. Prechelt, "An empirical comparison of seven programming languages," in *Computer*, vol. 33, no. 10, pp. 23-29, Oct. 2000, doi: 10.1109/2.876288.

[9] S. Macenski *et al.*, *“*Robot Operating System 2: Design, architecture, and uses in the wild”, *.Sci. Robot*, vol.**7**, no.66, May. 2022, doi:10.1126/scirobotics.abm6074.

[10] “X-500 V2 Kits - HolyBro”, holybro.com. https://holybro.com/products/x500-v2-kits?variant=42541212008637 (accessed May. 20, 2024).

[11] *Measuring Emissions: A Guide for Organisations*, ME 1527, 2020. [Online]. Available: https://environment.govt.nz/assets/Publications/Files/Measuring-Emissions-Detailed-Guide-2020.pdf

[12] "Government moves on climate promises: Supporting Documents", beehive.govt.nz. https://www.beehive.govt.nz/release/government-moves-climate-promises (accessed May. 21, 2024)

[13] B. Rao, A. G. Gopi, R. Maione, "The societal impact of commercial drones", in *Technology in Society*, vol. 45, pp. 83-90, 2016, doi:10.1016/j.techsoc.2016.02.009.

[14] *Exception for limited recreational operations of unmanned aircraft*, 49 USC 44809, 2024. [Online]. Available: https://uscode.house.gov/view.xhtml?req=granuleid:USC-prelim-title49-section44809&num=0&edition=prelim#

Appendix A – Updated Timeline

		Remaining Weeks of Project Work																				
		13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Flight Control System																						
1.0	**Software Development**																					
1.1	Finalise Software	🟧																				
1.2	Implement Target Node		🟧																			
1.3	Build & Debug Software			🟧	🟧																	
1.4	Finalise Documentation					🟧																
2.0	**Simulation**																					
2.1	Simulate Control in Gazebo						🟧															
2.2	Flight Test							🟧	🟧													
A-1	**Progress Inspection**									🟧 ***												
3.0	**Integration**																					
3.1	Replace Comms with DCP										🟧	🟧	🟧									
3.2	Build & Debug Software												🟧	🟧								
4.0	**Review & Testing**																					
4.1	Flight Test & Optimise													🟧	🟧	🟧						
4.2	Review Swarm Performance																🟧					
Drone Coordination Protocol																						
1.0	**Software Development**																					
1.1	Implement BP API	🟦	🟦																			
1.2	Implement SRP API			🟦	🟦																	
1.3	Implement VarDis API				🟦	🟦	🟦															
1.4	Finalise API Documentation							🟦														
2.0	**Review & Testing**																					
2.1	Unit Test API							🟦														
2.2	Test API with Applications								🟦	🟦												
A-1	**Progress Inspection**									🟦 ***												
3.0	**Integration**																					

3.1	Integrate with Control System	🟦
Project End		
1.0	**Evaluation**	▨
A-2	**Final Inspection**	🟩 ***
5.1	Project Handover	🟩
5.2	Prepare Final Report	🟩
A-3	**Final Report Hand-in**	🟩 ***

KEY		
A-1	Major Assessments	***
1.0	Topic	▨
1.1	Control System Tasks	🟧
1.1	DCP Tasks	🟦
1.1	Collective Tasks	🟩

Appendix B – CO_2-e Calculation

Note: This calculation does not consider losses.

$$P = IV = 5A \times 16V = 80\,W$$

Using ten drones gives 800W per charge that occurs 3 times an hour,

$$800W \times 3 = 2.4\,kWh$$

Multiplying by an emission factor of 0.101 gives,

$$2.4kWh \times 0.101\ \mathrm{kg}CO_2 - e/kWh = 0.2424\ \mathrm{kg}CO_2 - e$$

Repeating this every day for one year,

$$0.2424\,kgCO_2\text{-}e \times 365 = 88.5 \approx 90\,kgCO_2\text{-}e$$