# TRAJECTORY OPTIMIZATION OF

# METEOROLOGICAL SAMPLING

By

KAEVON ALEXANDER AZARTASH-NAMIN

Bachelor of Science in Mechanical & Aerospace
Engineering
Oklahoma State University
Stillwater, Oklahoma
2016

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July 2019

TRAJECTORY OPTIMIZATION OF

METEORLOGICAL SAMPLING

Thesis Approved:

DR. JAMEY JACOB

Thesis adviser

DR. HE BAI

DR. CHRISTOPHER CRICK

Name: Kaevon Alexander Azartash-Namin

Date of Degree: JULY 2019

Title of Study TRAJECTORY OPTIMIZATION OF METEORLOGICAL SAMPLING

Major Field: MECHANICAL & AEROSPACE ENGINEERING

ABSTRACT:

Swarming involves controlling multiple unmanned aerial systems or UAS in formation through the use of controls and algorithms. Swarm systems may be distributed and not rely on a central controller. As a result, this gives the system the potential to be robust and scalable, allowing for flexibility for the engineers to approach problems differently. Based on a variety of a few models and algorithms, such as artificial potential fields (APFs), agent-based modeling, dynamic data driven application systems (DDDAS), and virtual structures, it may be determined that using a variation of one of these would be the best course of action for formation flight for a swarm of UASs. Choosing the right controller is dependent on what works best for acquiring atmospheric data in a coordinated formation. Current atmospheric data is commonly taken using a weather tower or mesonet. A mesonet is typically a 10m high tower with a pressure, temperature, humidity sensor placed at the top. Deciding which controller can be used to not only take useful atmospheric data, but in many cases replace a mesonet due to mobility and customization is the goal. A wind profile is a transient matter, so using a swarm vs using one drone or a mesonet helps to solve the issues that the latter two run into due to time and space. A swarm can record multiple points at one time due to each agent being a data point representation, whereas a single drone can only account for a single location in time. A swarm using a virtual structure (VS) can cover a variety of amounts of space in a coordinated shape. A meosnet is stationary and only oriented vertically and an uncoordinated group of UAS does not have the capability to operate together. This leaves the capability that a VS swarm has to fill in the gaps or even replace the traditional approaches. An array of sensor packages with mobility, coordinated movement, and endless data points could give the VS swarm the advantage in atmospheric data sampling.

# ACKNOWLEDGEMENTS

I would like to acknowledge all of my research colleagues, peers, and professors for the support, advice and guidance throughout my undergraduate and graduate studies at Oklahoma State University. A personal acknowledgment to Dr. Jamey Jacob for his mentorship and guidance throughout the entirety of my education and this thesis research. I respect and look up to you more than you'll know. A big thank you to Rakshit Dayal for your help, guidance, and friendship. This project flourished due to your mentorship and I can truly call you a brother. I have to give a big "Thank You" to my sister, fellow engineer, and friend. Though you're in Aerospace because of me, thank you for being older so I can learn from your experiences. Thank you to my best friend and twin Bianca for your wisdom in not seeing the glass half full or empty, but where it really is. I want to acknowledge my parents, Nancy and Hossein, for them instilling a drive to be ambitious, take education serious, and reach for the stars. Lastly, I would like to thank the efforts on part of CLOUD-MAP and the National Science Foundation for their part in bringing together this research and funding in furthering the sciences associated with this project.

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University

# TABLE OF CONTENTS

Chapter Page

Chapter Page

# LIST OF FIGURES

Figure Page

Figure Page

Figure Page

# NOMENCLATURE

| | |
|---|---|
| $a$, $b$, $c$ | formation control constants |
| $C_e$, $L_e$ | exponential potential amplitude and length scale |
| $C_h$ | hyperbolic amplitude |
| $C_r$, $L_r$ | repulsive potential amplitude and length scale |
| $\mathbf{e}$ | error in system |
| $\mathbf{K}_1$, $\mathbf{K}_2$ | controller feedback gains |
| $N$ | number of unmanned aerial vehicles |
| $p$, $q$, $r$ | roll, pitch, and yaw rates, rad / $s^{-1}$ |
| $r$ | scalar constant |
| $u$, $v$, $w$ | body axis speed in $x$, $y$, and $z$ directions, $ms^{-1}$ |
| $u_d$ | desired unmanned-aerial-vehicle speed, m / s |
| $\mathbf{u}_{\mathrm{lat}}$ | lateral inputs |
| $\mathbf{u}_{\mathrm{long}}$ | longitudinal inputs |
| $u_x$ | desired unmanned-aerial-vehicle forward speed, m / s |
| $U_i^{S,e}$ | exponential steering potential of the *i*th unmanned aerial vehicle |
| $U_i^{S,h}$ | hyperbolic steering potential of the *i*th unmanned aerial vehicle |
| $U_i^{S,he}$ | combined hyperbolic-exponential steering potential of the *i*th unmanned aerial vehicle |
| $U^R$ | repulsive potential field |
| $U^S$ | steering potential field |
| $V_{\mathrm{max}}$ | maximum speed of the unmanned aerial vehicle |
| $V_{\mathrm{trim}}$ | trim speed of the unmanned aerial vehicle |
| $\mathbf{v}_i$ | velocity vector of *i*th unmanned aerial vehicle |

| | |
|---|---|
| $\mathbf{v}_j$ | velocity vector of *j*th unmanned aerial vehicle |
| $\mathbf{v}_\mathrm{i}^\mathrm{R}$ | repulsive velocity vector of the *i*th unmanned aerial vehicle |
| $\mathbf{v}_\mathrm{i}^\mathrm{S}$ | steering velocity vector of the *i*th unmanned aerial vehicle |
| $\mathbf{v}_\mathrm{i}^\mathrm{S,e}$ | exponential steering velocity vector of the *i*th unmanned aerial vehicle |
| $\mathbf{v}_\mathrm{i}^\mathrm{S,h}$ | hyperbolic steering velocity vector of the *i*th unmanned aerial vehicle |
| $\mathbf{x}_i$ | position vector of the *i*th unmanned aerial vehicle |
| $\mathbf{x}_j$ | position vector of the *j*th unmanned aerial vehicle |
| $\mathbf{x}_\mathrm{lat}$ | lateral state variables |
| $\mathbf{x}_\mathrm{long}$ | longitudinal state variable |
| $\mathbf{x}_o$ | equilibrium position vector |
| $\mathbf{y}$ | output of control system |
| $\mathbf{y}_d$ | desired input to the system |
| $\delta_e$, $\delta_a$, $\delta_r$ | input to elevator, aileron, and rudder, rad |
| $\delta_{e,d}$, $\delta_{a,d}$, $\delta_{r,d}$ | desired input to elevator, aileron, and rudder, rad |
| $\delta_t$ | thrust input, N |
| $\delta_{t,d}$ | desired thrust input, N |
| $\Theta_d$ | desired pitch angle, rad |
| $\mu$ | bifurcation parameter |
| $\sigma_i$ | formation control scalar |
| $\phi$, $\Theta$, $\psi$ | roll, pitch, and yaw angles, rad |
| $\psi_d$ | desired heading angle, rad |
| (^) | unit vector |

# CHAPTER I

## 1. INTRODUCTION

### 1.1 Motivation

Unmanned aerial vehicle or commonly known as UAS are being used across a variety of industries and research fields. There are many UASs that are being used in civil applications such as lightweight orders delivery, natural disaster inspection, weather surveillance, film industry, agricultural aid, and police surveillance, just to name a few. In addition, there is the military side of the UAS world where UASs are used in any way possible to gain a strategic edge. In fact this is where the concept of unmanned aerial vehicles was first used. The American Civil War when the North and South tried to launch balloons with explosive devices that would fall into the other side's ammunition depot and explode [2]. The difference between then and now is in the technology of controls. Now there is sufficient technology to launch and control aircraft, and with much better precision. It is in this continued growth in technology that has really given rise to the idea of swarming, and the use of multiple UASs in formation to complete a task more efficiently.

The motivation of this paper is to provide a way for a swarm system to effectively take atmospheric measurements by utilizing a virtual structure swarm formation to sample a wind profile. The reason for this is because using one UAV would result in one data point being taken at a given time, and using a weather tower would result in multiple data points at a given time but in fixed locations. However, if multiple UAVs were used in a swarm then each UAV could act as a data point but with the flexibility to move and change altitude to result in a better wind profile curve. Such curves can be seen below in (Figure 1) and show how much the profiles can change given certain characteristics to the day. Being able to place the UAVs in tighter clusters at the curves would allow for better wind profile results, thus is why using a UAV swarm would be more beneficial to using a single UAV or weather tower.

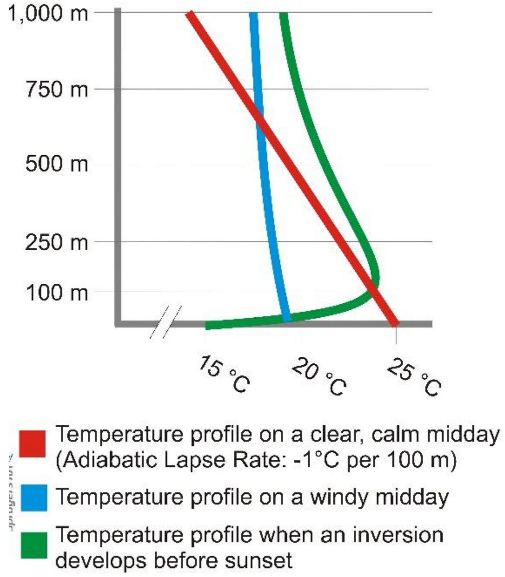


**Figure 1: Wind profile based off of temperature [8]**

This concept is an integral part of CLOUD-MAP, Collaboration Leading Operational UAS Development for Meteorology and Atmospheric Physics, a National Science Foundation (NSF) funded grant led by the Oklahoma State University (OSU), the University of Oklahoma, the University of Kentucky, and the University of Nebraska Lincoln. CLOUD-MAP is focused on the development and implementation of unmanned aircraft systems and their integration with sensors for atmospheric measurements on Earth with the emphasis on Meteorology and Atmospheric Physics (MAP). CLOUD-MAP has objectives to create and demonstrate UAS capabilities needed to support UAS operating in the extreme conditions typical in atmospheric observations, including the sensors, navigation planning, learning, control, and communications technologies as well as develop and demonstrate coordinated control and collaboration between autonomous air vehicles during MAP missions. The motivation for this work is rooted within the advancement of three-dimensional forecasting, but its impact will contribute to a much larger UAS movement.

## 1.2 Goals & Objectives

The purpose of this research is to evaluate the performance of sUAS using an adaptive controller for optimized trajectory, the systems can be used in many other areas of interest. Determining if using multiple sUAS using an adaptive controller so that they operate in a coordinated manner is the goal. Atmospheric boundary layer research and wind profile modeling is a driving force behind this project in application.

- Develop autonomous adaptive controller
- Test controller on 3DR Solos
- Test coordinated swarm functions on Solos in simulation
- Test coordinated swarm on Solos at airfield
- Sample atmospheric data for modeling the wind profile

## 1.3 Outline of Thesis

The layout of this paper proceeds with Chapters II through VI followed by an Appendix. Chapter II: Review of Literature, covers all background and previous works studied and referenced in this body of work. Followed by, Chapter III: Theory, which covers a discussion on adaptive control theory that leads further into a discussion on data driven adaptive control theory. Adaptive controller application and governing equations used in this work are introduced in this chapter as well. Chapter IV: Methodology & Experimental Arrangement, discusses the tools and methods used to evaluate experimental procedures and the setup. Validation methods in simulation is discussed here. The last two chapters are Chapter V: Results and Chapter VI: Conclusions. Chapter V: Results, presents and discusses the experimental results derived from Chapter IV. Simulation results are presented in detail here. The last chapter, Chapter VI: Conclusions, follows giving a brief summary of the results and recommendations for this body of work. Future work is presented in this chapter for furthering this research area and ensuring that the research ahead is maintained with quality and a clear understanding of what still needs to be done going forward. An appendix is given at the end with results, figures, and data that could not be presented in previous chapters for reference.

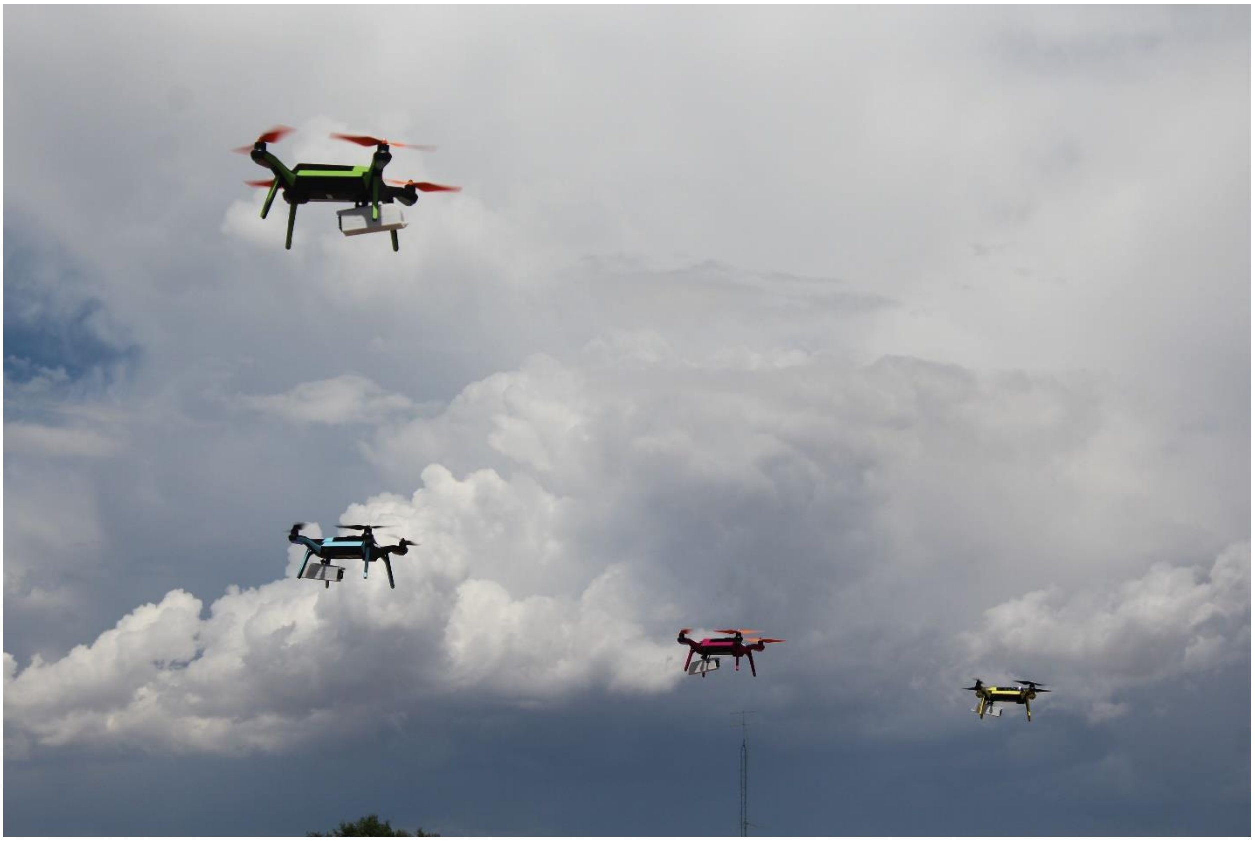


**Figure 2: Solo swarm during flight campaign**

# CHAPTER II

## 2. REVIEW OF LITERATURE

### 2.1 Background

As stated before, one of the prime purposes of an unmanned aerial vehicle, or any unmanned, is to take a task that was performed by someone and have an unmanned vehicle do it, while still being effective. In many cases, such as in military application, removing the pilot from the scenario eliminates risk to the pilot. In other cases the convenience and reduction of cost by using a UAS is what pushes the need for them. Either way these tasks not only need to be effective, but also cost efficient to stay relevant. One approach to this solution is to allow the unmanned aerial vehicles to function as a swarm which can reduce the complexity of motion control by reducing code size and communication requirements. Swarming relies on local sensing and reactive behaviors of autonomous and homogeneous individuals from which an emergent global behavior arises [2].

The term swarm depending on how applied can mean a slight variation of the same thing. For instance a classic example of swarming in a specific biological sense, a swarm of bees or colony of ants could be discussed. Furthermore, looking to other systems that swarm like, a flock of birds, a crowd of people, or even cars in traffic; all of these examples show fundamentals of swarming. There are many instances where a single UASs role it can play becomes limited due to operating range and payload. However, with the use of multiple UASs or swarms, those same tasks that a single UAS could not do not only becomes possible, but in many cases more efficient. Swarm intelligent systems are not only efficient at solving group-level problems, but also decentralized, controllable by few simple parameters, making possible the command and control of UAS swarms by a single operator [6].

There are multiple different thought process that have been considered for achieving the most useful and efficient swarm. There is the artificial potential field method that is based on classical bifurcation theory; agent-based modeling that requires careful model calibration, and DDDAS that when coupled with agent-based modeling it minimizes model inaccuracy.

## 2.2 Artificial Potential Field

The artificial potential field or APF method is a fusion behavior based architecture that combines several behaviors together, resulting in a superimposed behavior [1]. APF was first introduced by Khatib for obstacle avoidance for manipulators and mobile robots. It originally was studied for the purpose of path planning for autonomous single mobile robots; however, it now includes the study of path planning for swarming autonomous systems. APF has been able to replace traditional algorithm validation by generating a first or second order dynamical system which is often used to mathematically prove the stability of the emergent behaviors. This is beneficial due to the array of theorems in dynamical systems theory that can be used to develop new ways of controlling a swarm. In addition, by using a steering and repulsive APF, a swarm of unmanned aerial vehicles can be successfully controlled so that desired formations are formed, with the new approach of bifurcating potential fields allowing for a transition between different patterns via a parameter switch [1]. To get the desired swarm velocity field, a first-order dynamical system is used to transform the velocity field into guidance commands for forward control speed and heading angle. To demonstrate this model the guidance algorithm is applied to a formation of UASs, while considering a linearized six-degree-of-freedom (6-DOF) UAS model, with a robust controller design for the linear time-invariant multivariable systems used [1].

## 2.3 Agent-Based Modeling

Agent-based modeling and simulation is an approach to representing a system as autonomous agents that interact amongst one another as well as with the environment [6]. An agent specifically has a behavior that is designed to capture local interactions, which over time the characteristics of the system form. Agent-based modeling is an intuitive paradigm for representing swarms [6]. Unlike equations, which apply to top-down models, agent-based modeling is looked at as a bottom-up approach to modeling a system. Interestingly, the bottom-up approach can show the generative nature of system properties, thus leading to agent-based simulation. It should be noted that though agent-based models offer a great deal of explanatory capability, there is a cost. Rapid interactions between agents can allow for unpredictable, non-linear results, which require careful model calibration. In addition, verification and validation of the model challenge the paradigm.

## 2.4 Dynamic Data Driven Application Systems (DDDAS)

DDDAS or Dynamic Data Driven Application Systems entails the ability of an executing application to incorporate simulated data into the decision process, while conversely being able to dynamically manage sensors to refine measurements [6]. As new sensor data is taken into the systems simulation, a feed-back and control-loop is formed between the real-world application and simulation model; via the simulation modeling complex non-linear dynamics in quick time. The sensor controls constantly drive the measurement process for recalibrating the simulation thus resulting in precise results. Because agent-based simulations frequently require careful model calibration to prevent unpredictable dynamics, using an agent-based model within a DDDAS framework minimizes model inaccuracy by repeatedly recalibrating with new data [6]. As a result an accurate model is produced, supporting application optimization using simulation.

## 2.5 Boundary Layer Meteorology

The thick layer of gases commonly known as air that surrounds earth is the atmosphere. It is divided into five layers with most of the weather and clouds being in the first layer known as the troposphere. The troposphere is itself loosely made up of two additional portions, a boundary layer and free atmosphere (Figure 3). For this research this troposphere is where the focus will be, with the primary focus being on the lowest portion of the troposphere, the atmospheric boundary layer (ABL) or planetary boundary layer (PBL). This 100 to 3000 m of the atmosphere is what plays a major role in different weather phenomena. This is due to the contact the atmosphere makes with the surface resulting in an energy exchange. The presence of the earth's surface on the atmosphere allows for a response from local weather with a timescale of around an hour or less. This response is due to forcings that include frictional drag, evaporation and transpiration, heat transfer, pollutant emission and terrain induced flow modification [10]. The thickness of the boundary layer varies in time and space with a range of hundreds of meters to a few kilometers.

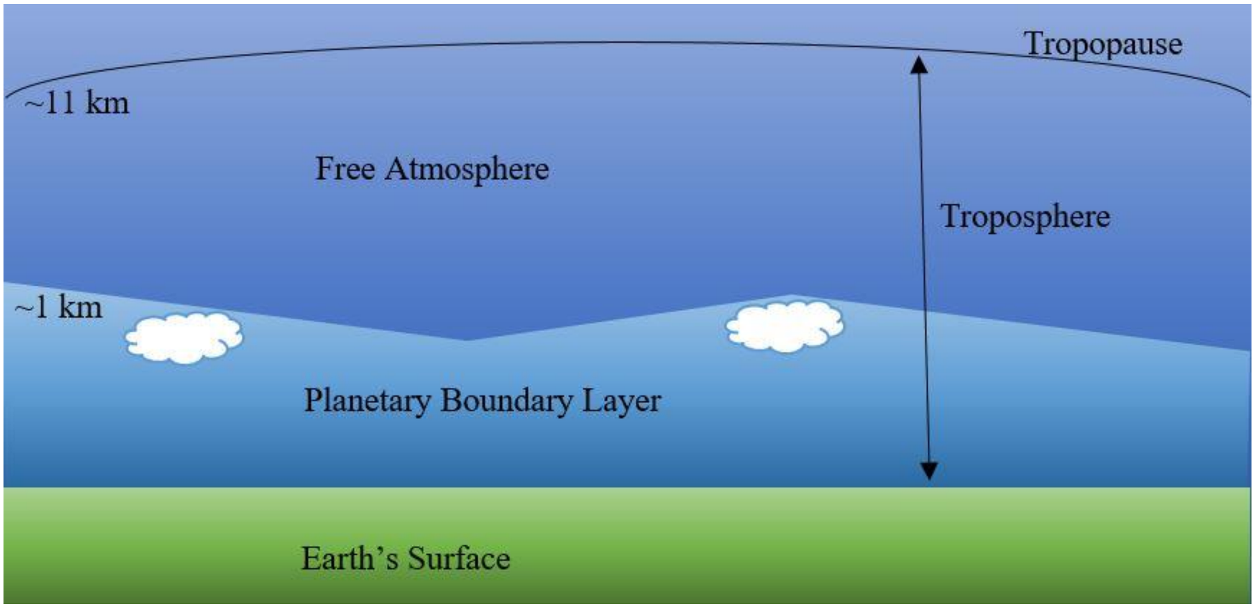


**Figure 3: Troposphere divided into two parts: boundary layer and free atmosphere**

To better understand the energy exchange in the ABL there are meteorological measurements that can be collected and looked at. First there are the thermodynamic variables of temperature, pressure, and humidity. Next there are the kinematic variables of wind velocity. All of these are relatively simple measurements, but essential in understanding the formation of severe weather like thunderstorms and tornadoes [11].

## 2.6 Small UAS Atmospheric Boundary Layer Profiling

Low-altitude sampling would allow for measurements of surface-based convergence and the intersection of air mass boundaries, both of which would aid in the understanding of tornado genesis. With the possibility of rotation occurring in as few as 20 minutes from the first sign of possible tornadic activity, rapidly deployable, low altitude platforms that can collect measurements at fine spatial and temporal scales can lead to more timely and more precise tornado warnings [12,14,15].

Looking at the limits that ground based systems like weather stations and radar have, and the limits that satellite sensors and weather balloons have, there is a need for additional technology that can capture ABL data in ways that these cannot. The network of ground weather stations, or mesonets, are usually 10 m high towers that record temperature, pressure, humidity, wind velocity, and other environmental data [16]. The spacing between these towers are anywhere from 2 km to 40 km apart, and the measurements get interpolated for regional sections [13]. The drawback is that sampling taken below 10 m and above that cannot be recorded by the mesonets so the full dynamics of the ABL cannot be seen [12]. Ground based weather radar send directional pulses of microwave radiation and measure the reflectivity of the radiation scattered by water droplets or ice particles back to the sensor [17]. The drawbacks of radar however include issues in sensing temperature and humidity, difficulty sensing the ABL because of the Earth curvature and physical obstructions, and interference from birds, insects, and ground clutter [18-20]. Weather sensing satellites such as the Geostationary Operational Environmental Satellite (GOES) system has been a centerpiece for weather forecasting in the U.S. [21]. The drawback for weather satellites are their inability to provide spatial precision, temporal resolution, and capturing certain types of data for observing the ABL [22]. In addition, though weather balloons allow for sensing the entire vertical profile of variables in the ABL, they are either limited by their tether height or ascend in an uncontrollable manner when not tethered [12]. It is because of all of these reasons and drawbacks that small unmanned aircraft systems (sUAS) have the potential to fill the spatio-temporal gaps in ABL sampling. Using sUAS whether they be fixed-wing or rotor platforms allow for an array of sensor layouts and customization depending of the need.

In addition, to using sUAS for ABL data measurements, using common geostatistical techniques to determine vertical spatial sampling like variogram modeling can be looked at. This geostatistical technique can quantify the spatial autocorrelation of a given signal, and be used to capture the spatial structure of atmospheric phenomena at different times of the day [12]. Therefore, the optimal spatial separation that should be allowed between measurements recorded by the sensors on the sUAS can be determined by the variogram [12]. Below in figure 4 an example variogram is provided, and seen from that, the distance which spatial dependence for the regionalized variable is not present anymore is determined by analyzing three properties known as the range, sill, and nugget [12]. The upper boundary of values is the sill and occurs when measured values between samples are invariant at larger lag distances and the curve levels out. The lag distance at the sill is the range and is where the measurements have spatial dependency [12]. In some cases the variogram model will not pass through the origin and instead intersect the ordinate at $\hat{y}(h)$greater than zero, and when there is uncertainty in the data this is referred to as the nugget effect [12].

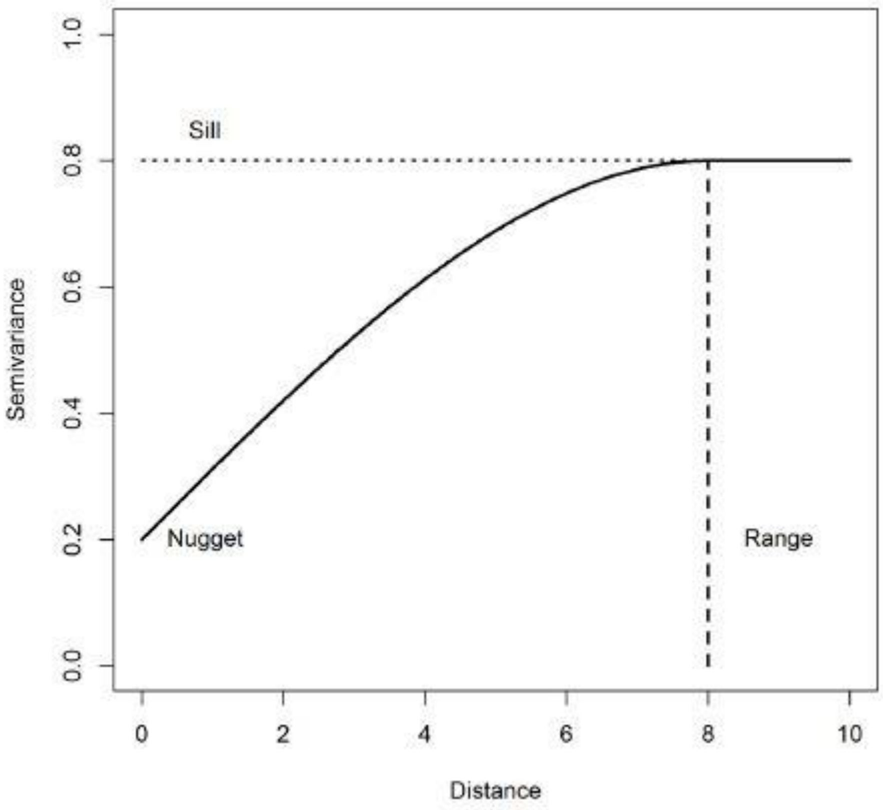


**Figure 4: Example of typical variogram produced from plotting semi variance versus lag distance. Locations of nugget, range, and sill are shown [12].**

# CHAPTER III

## 3. THEORY

This chapter discusses the theory behind controls, specifically swarms, as well as the applicability and science that drives it. The first section will describe the theory of autonomous controls followed by a discussion of the governing equations used for developing the guidance and controls that are used in various applications. Section 3.2 will discuss theory in how it is applied and the science motivations.

## 3.1 Control Theory

### 3.1.1 Autonomous Controls

#### A. Behavior Based Control

There are many different types of behavior based controls that a UAS or swarm could be asked to perform and with that many different terminology. For the purpose of simplicity in explaining the terms, let's consider the UASs to be point-masses and in a 2D environment. For low-level maneuvers we will look at approach and avoid. Approach is when a UAS moves toward a target by either banking right or left to turn in its direction and increase or decrease thrust in order to catch the target. Avoid is when a UAS banks right or left to move away from a target and either increases or decreases thrust to miss the target. Now let's look at a number of basic behaviors such as collision avoidance, cluster forming, area spreading, target tracking, path following, leader following, and obstacle avoidance. Collision avoidance for a UAS is a function that calculates the position of the closest object, and if that position relative to each other is smaller than the collision avoidance distance, then the UAS and object avoids each other. Cluster forming uses a function that calculates all of the centroids of the UASs, and it approaches the centroid if the UAS's distance relative to the centroid is greater than aggregation distance. Area spreading is when the UAS avoids the centroid because the UAS's distance to the centroid is smaller than the dispersion distance. Target tracking occurs when the UAS's distance to a target is greater than the allowed distance, and as a result the UAS approaches the target. In path following the UAS approaches the path when the distance relative to the path is greater than the distance that is allowed. Leader following assumes the UASs know the leader's location, and if their distance to the leader is greater than the following distance, they approach the leader. Finally, obstacle avoidance is when a UAS avoids an obstacle when its distance to the obstacle is less than the avoidance distance. All the previously stated terms are important to know when trying to understand UASs and swarms, and their movements and behaviors with one another.

## B. Swarm UAS Model

Let's start off with setting up the definition of UAS position and velocity vectors for a swarm (Figure5). Here there shows two UAS that are treated as a particle, but it really could be looked at as having N homogeneous UASs, where ($x_i$, $v_i$) and ($x_j$, $v_j$) represent the position and velocity vectors of the i$^{th}$ and j$^{th}$ UASs. In addition, $x_{ij}$ is the separation distance between the UASs.

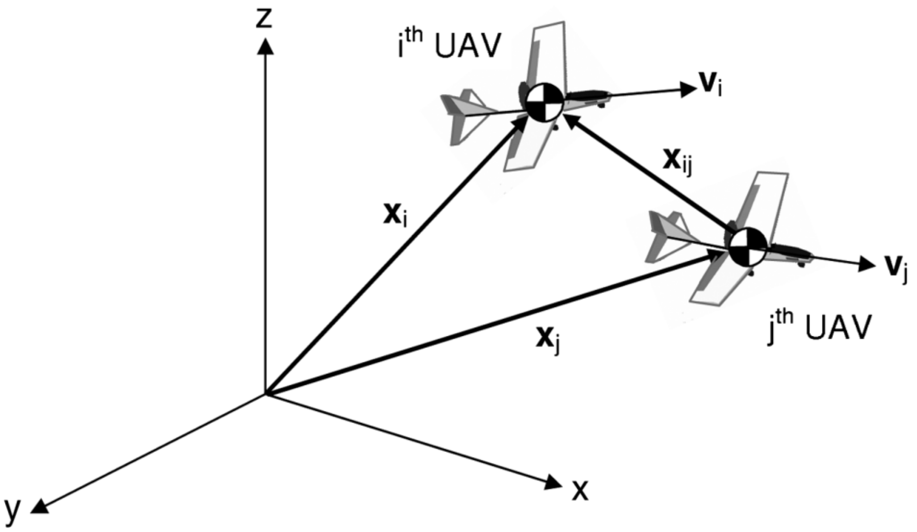


**Figure 5: Definition of UAS position and velocity vectors [1]**

Since each UAS is seen as a particle with a velocity field being applied to it, we get Eq. (1):

$$\mathbf{v}_i = -\nabla_i U^S(\mathbf{x}_i) - \nabla_i U^R(\mathbf{x}_{ij}) \tag{1}$$

Here $U^S$ represents the steering artificial potential field and $U^R$ represents the repulsive artificial potential field. In addition to the equation, each UAVs velocity field is defined by the gradient of the steering APF and the gradient of the repulsive APF. This is important because the steering APF is used to control the formation of the swarm, and the repulsive APF is used for collision avoidance within the swarm and equal spacing between the UAVs.

In regards to artificial potential fields, previous work has shown that by using a guidance algorithm based on classical bifurcation theory, a formation or swarm of UAVs can create autonomous desired patterns by switching between patterns via simple parameter change [1]. For APFs using the Lyapunov stability methods is chosen because the autonomous patterns can be proven, unlike traditional means where the algorithm validation methods cannot be proven analytically. A bounded bifurcating APF is developed for the purpose of saturation being an issue to the stability of the system. Looking back at Eq. (1) the ith UAV has a maximum control velocity represented by Eq. (2):

$$|\mathbf{v}_i| \leq |\nabla_i U^S(\mathbf{x}_i)| + \left|\nabla_i U^R(\mathbf{x}_{ij})\right| \tag{2}$$

As a result, this means each UAV in the swarm will have a maximum control velocity that is made up of the maximum gradient of the steering APF and the maximum gradient of the repulsive APF.

Bifurcating potential fields are useful because they allow for easy shape change of the potential just from changing the parameters. As a result, the stability properties for the potential changes as well as the patterns that the swarm uses. Here Eq. (3) shows the bifurcating steering potential based on the pitchfork bifurcation equation:

$$U_i^S(\mathbf{x}_i;\ \mu) = -\tfrac{1}{2}\mu(|\mathbf{x}_i| - r)^2 + \tfrac{1}{4}(|\mathbf{x}_i| - r)^4 \tag{3}$$

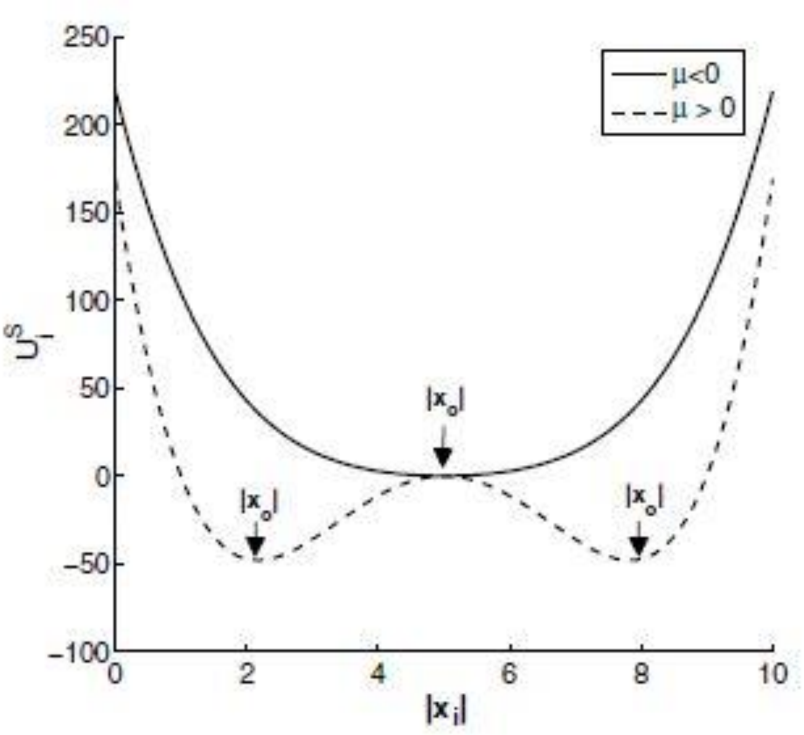


**Figure 6: Pitchfork potential, r = 5 [1]**

Figures 6 and 7 show how the potential and number of equilibrium positions alter as the bifurcation parameter is changed [1].

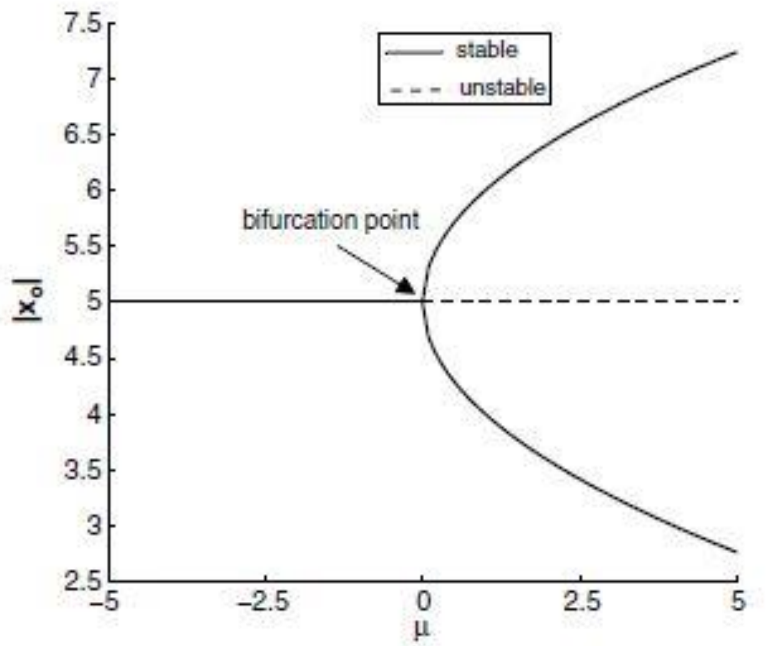


**Figure 7: Pitchfork bifurcation diagram, r = 5 [1]**

### 3.1.2 Governing Equations

UAS Guidance and Control

**Guidance Law Equations:**

Below are a set of real commands for each UAS taken from the velocity field, and they are forward speed, heading, and pitch:

$$u_{d,i} = \sqrt{v_{x,i}^2 + v_{y,i}^2 + v_{z,i}^2} \tag{4}$$

$$\psi_{d,i} = \arctan\left(\frac{v_{y,i}}{v_{x,i}}\right) \tag{5}$$

$$\theta_{d,i} = \arctan\left(\frac{v_{z,i}}{v_{x,i}}\right) \tag{6}$$

Verified in a 6-DOF linear kinematic model, this particular model is for a low-speed fixed-wing UAV that is linearized about straight and level conditions. However, a similar model could be made for other types of UAV such as quadrotor UAVs. Next up there are the uncoupled longitudinal equations of motion (Figure 8) and lateral equations of motion (Figure 9) for this model.

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{\delta}_e \\ \dot{\delta}_t \end{bmatrix} = \begin{bmatrix} X_u & X_w & 0 & -g\cos\theta_0 & 0 & 0 \\ Z_u & Z_w & u_0 + z_q & -g\sin\theta_0 & 0 & 0 \\ M_u & M_w & M_q & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{T_e} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{T_t} \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ \delta_e \\ \delta_t \end{bmatrix} + \begin{bmatrix} X_{\delta e} & X_{\delta th} \\ Z_{\delta e} & Z_{\delta th} \\ M_{\delta e} & M_{\delta th} \\ 0 & 0 \\ -\frac{1}{T_e} & 0 \\ 0 & -\frac{1}{T_t} \end{bmatrix} \begin{bmatrix} \delta_{e,d} \\ \delta_{t,d} \end{bmatrix}$$

**Figure 8: Uncoupled longitudinal equations of motion [1]**

$$\begin{bmatrix} \dot{v} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \\ \dot{\psi} \\ \dot{\delta}_a \\ \dot{\delta}_r \end{bmatrix} = \begin{bmatrix} Y_v & Y_p & Y_r - u_0 & g\cos\theta_0 & 0 & 0 & 0 \\ L_v & L_p & L_r & 0 & 0 & 0 & 0 \\ N_v & N_p & N_r & 0 & 0 & 0 & 0 \\ 0 & 1 & \tan\theta_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{T_a} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{T_r} \end{bmatrix} \begin{bmatrix} v \\ p \\ r \\ \phi \\ \psi \\ \delta_a \\ \delta_r \end{bmatrix} + \begin{bmatrix} Y_{\delta a} & Y_{\delta r} \\ L_{\delta a} & L_{\delta r} \\ N_{\delta a} & N_{\delta r} \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{T_a} & 0 \\ 0 & -\frac{1}{T_r} \end{bmatrix} \begin{bmatrix} \delta_{a,d} \\ \delta_{r,d} \end{bmatrix}$$

**Figure 9: Uncoupled lateral equations of motion [1]**

It should be noted, to achieve steady-state flight, consider the use of a robust controller of a linear time-invariant multivariable system [1]. This can be seen below in Figure 6:

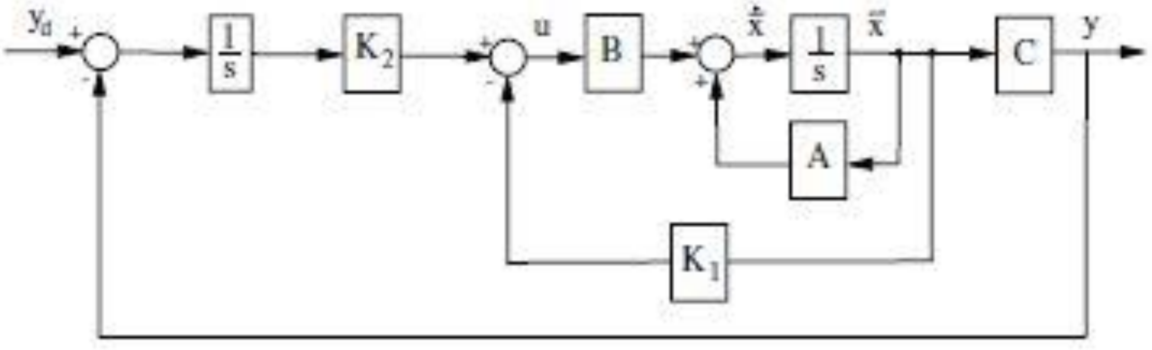


**Figure 10: Robust multivariable linear time-invariant control system [1]**

**Control Law Equations:**

Using the state-space form below, the longitudinal equation of motion and lateral equation of motion can be expressed.

$$\dot{\bar{\mathbf{x}}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\mathbf{u} = \mathbf{A}\begin{bmatrix}\bar{\mathbf{x}}_{\text{long}}\\ \bar{\mathbf{x}}_{\text{lat}}\end{bmatrix} + \mathbf{B}\begin{bmatrix}\mathbf{u}_{\text{long}}\\ \mathbf{u}_{\text{lat}}\end{bmatrix} \tag{7}$$

$$\mathbf{y} = \mathbf{C}\bar{\mathbf{x}} = \mathbf{C}\begin{bmatrix}\bar{\mathbf{x}}_{\text{long}}\\ \bar{\mathbf{x}}_{\text{lat}}\end{bmatrix} \tag{8}$$

The state variables of the system are $\bar{\mathbf{x}}_{\text{long}} = [u, w, q, \theta, \delta_e, \delta_t]^T$and $\bar{\mathbf{x}}_{\text{lat}} = [v, p, r, \emptyset, \psi, \delta_a, \delta_r]^T$**.** The inputs are $\mathbf{u}_{\text{long}} = [\delta_{e,d}, \delta_{t,d}]^T$ and $\mathbf{u}_{\text{lat}} = [\delta_{a,d}, \delta_{r,d}]^T$**.** The output of the system is **y**.

Below is the error in the system where $\mathbf{y}_d$ is the input, which can be seen in Figure 10.

$$\mathbf{e}(t) = y - y_d \tag{9}$$

Differentiating Eqs. (7) and (8) brings Eqs. (10) and (11) if steady state is assumed, i.e. $\dot{\boldsymbol{y}}_d = 0$.

$$\frac{d}{dt}\dot{\mathbf{x}} = \mathbf{A}\dot{\mathbf{x}} + \mathbf{B}\dot{\mathbf{u}} \tag{10}$$

$$\frac{d}{dt}\mathbf{e} = \mathbf{C}\dot{\mathbf{x}} \tag{11}$$

Combining Eqs. (10) and (11) results in Eq. (12).

$$\frac{d}{dt}\begin{bmatrix}\dot{\mathbf{x}}(t)\\ \mathbf{e}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{A} & 0\\ \mathbf{C} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\dot{\mathbf{x}}(t)\\ \mathbf{e}(t)\end{bmatrix} + \begin{bmatrix}\mathbf{B}\\ 0\end{bmatrix}\dot{\mathbf{u}}(t) \tag{12}$$

Consider the rank for controllability in Eq. (13) to successfully control the system. The order of the **A** matrix is *n* and the order of the **C** matrix is *p*.

$$\text{rank}\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{0}\end{bmatrix} = n + p \tag{13}$$

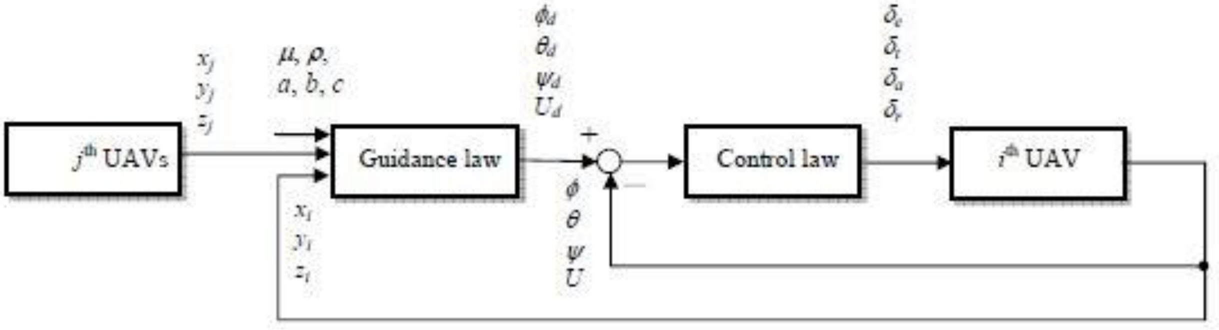


**Figure 11: Guidance and control block diagram [1]**

The equation below is the input for $\mathbf{u}$ for the longitudinal motions of the controller and the lateral motions of the controller. In addition, $\mathbf{K}_1$ and $\mathbf{K}_2$ are the feedback gains for the controller.

$$\mathbf{u}(t) = -\mathbf{K}_1\bar{\mathbf{x}}(t) - \mathbf{K}_2 \int_0^t \mathbf{e}(t)dt \tag{14}$$

Lastly, $\mathbf{y}_\mathrm{d}$ is not constant if the system is in transition. The system can still be controlled toward $\mathbf{y}_\mathrm{d}$, since the poles of the system never change [1].

## Virtual Structure

Here preliminaries and model formulation for consensus of general linear multi-agent systems with intermittent measurements are introduced. We will have $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$ as a directed graph and a set of vertices $\mathcal{V} = \{v_1, v_2, \dots, v_{\mathcal{N}})$. The set of directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and weighted adjacency matrix $\mathcal{A} = [a_{ij}\ ]_{N\times N}$ with elements $a_{ij}$. Edge $e_{ij}$ in $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$ is denoted by ordered pair of vertices $(v_j, v_i)$, $v_j$ being the parent vertices and $v_i$ being the child vertices with $e_{ij} \in \mathcal{E}$ if and only if $a_{ij} > 0$. There is a directed path from node $v_i$ to $v_j$ as a sequence of edges with distinct vertices, which if there is a directed path between any pair of distinct vertices then a strongly connected directed graph can be made. A directed tree is a graph where every vertex $v$, except the root vertex r to $v$ [9]. Network $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$ is a directed tree that contains all of its vertices.

Now looking at how the model is formed, consider a network of N agents that have a general linear dynamics. We can represent the dynamics of agent *i* as,

$$\dot{x}_i(t) = Ax_i(t) + Bu_i(t), \tag{15}$$

where $x_i(t) \in \mathbb{R}^n$ is the state of agent *i*, $u_i(t) \in \mathbb{R}^m$ is the control input acting of agent *i*, $(1 \leq i \leq N)$, and A, B are constant real matrices with compatible dimensions [9]. Communication between agents is represented by the fixed directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$. The vertices are the agents and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the communication channels between the agents. It should be stated that many currently existing protocols implemented assume that all the information is transmitted continuously among agents in their communication network. It can be seen though in real world scenarios that agents will only communicate with neighbor agents over some disconnected time intervals due to how unreliable communication channels can be and the failure of hardware. The following is a distributed consensus protocol with intermittent measurements:

$$u_i(t) = \begin{cases} cF \sum_{j=1}^{N} a_{ij} \left(x_i(t) - x_j(t)\right), & t \in [k_\rho, k_\rho + \delta), \\ 0, & t \in [\, k_\rho + \delta, (k+1)\rho), k \in \mathbb{N}, \end{cases} \tag{16}$$

$c > 0$ is the coupling strength, $F \in \mathbb{R}^{m \times n}$ is the feedback gain matrix to be determined, $\mathcal{A} = [a_{ij}\,]_{N \times N}$ is the adjacency matrix of graph $\mathcal{G}(\mathcal{A})$, and scalars $\rho > \delta > 0$ [9].

$$\begin{array}{ll} \dot{x}_i(t) = Ax_i(t) + cBF \sum_{j=1}^{N} a_{ij}\left(x_i(t) - x_j(t)\right), & t \in \left[k_\rho, k_\rho + \delta\right), \\ \dot{x}_i(t) = Ax_i(t), & t \in \left[\, k_\rho + \delta, (k+1)\rho\right), k \in \mathbb{N}, \end{array} \tag{17}$$

Where i = 1, 2, …, N.

### Graph Theory

Graph theory is most commonly known as the study of graphs. In mathematical structures it is used to model pairwise relations between objects. It is made up of vertices, also known as nodes or points, and those are connected by edges also known as links or lines. There are two common types of graphs, undirected and directed. Undirected are made up of edges that link two vertices symmetrically. Directed are made up of edges that link two vertices asymmetrically. Below Figure 12 shows the difference between an undirected and directed graph. For the purpose of communication and UAS, a line with no arrow denoting direction means communication is bilateral and can be done either direction. A line denoting direction with an arrow means that communication can only be done in the direction and to the node that the arrow is pointing.

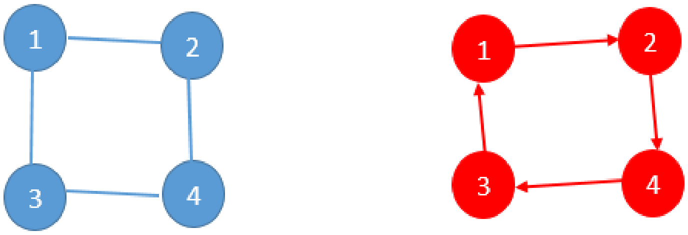


**Figure 12: Undirected graph (left) directed graph (right)**

## 3.2 Application

### 3.2.1 Atmospheric Boundary Layer Profile

ABL measurements can already be done in multiple ways. These ways being ground stations like mesonets, weather balloons, satellite systems, weather surveillance radar, and sUAS. Knowing the limitations that most all of these have, there is a reason why sUAS are rapidly emerging and being used to take atmospheric measurements around the world. While sUAS are increasingly being employed in ABL sampling in recent years, they have first been used many years back to record temperature, pressure, humidity, and aircraft velocity at altitudes of around 3000 m [12]. In addition, recently ABL research using sUAS has been used in Antarctica like with the Small Unmanned Meteorological Observer (SUMO) UAV seen in figure 13. In Iceland fine-scale atmospheric models have been validated using sUAS [11], and in New Zealand relative humidity and temperature was recorded and compared to radiosondes using sUAS [24]. Furthermore, sUAS have been used to gather data from super cells and air masses due to their versatility and ability to keep operators out of harm's way [12]. These are all proven ways that UAS have been used in taking measurements from the ABL. Applying a swarm controller would open up their versatility even more in allowing for more data points over a given space at a single time.

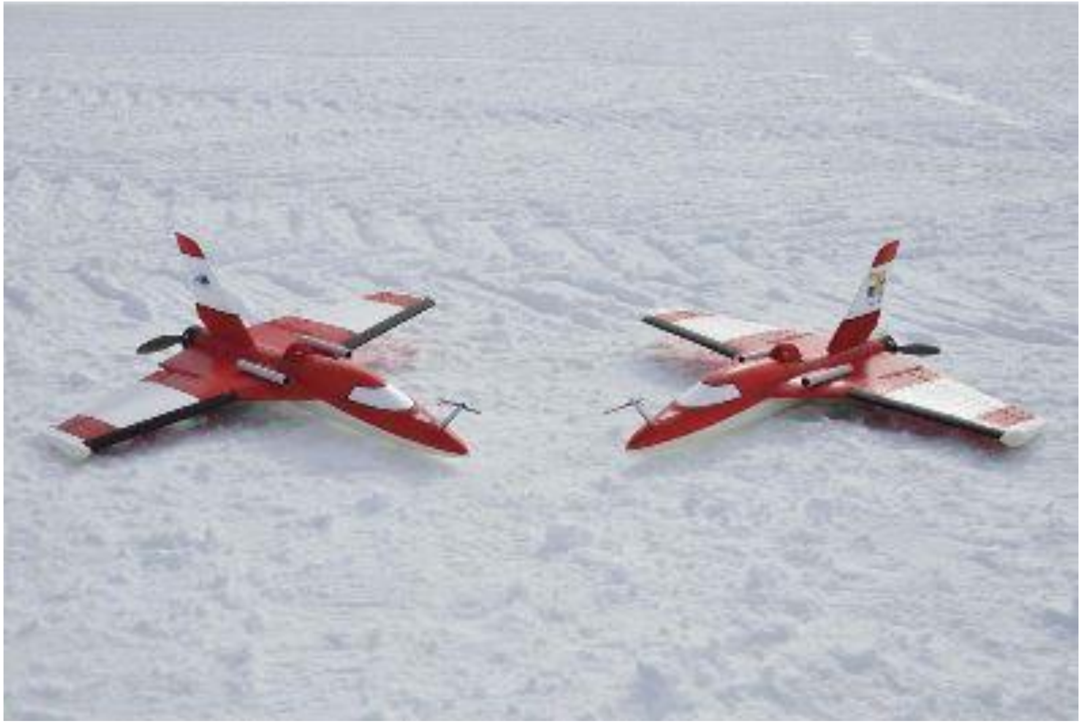


**Figure 13: Small Unmanned Meteorological Observer (SUMO) unmanned aerial vehicles (UAVs) at Williams Field Antarctica [23]**

CHAPTER IV

# 4. METHODOLOGY & EXPERIMENTAL ARRANGEMENT

This chapter discusses the tools, methods, and experimental setups that were used to develop the swarm. The first section will go into detail on controller design and selection processes. The second and third sections discuss the simulation and data reduction methods, respectively. The benefits of those specific approaches over other methods are discussed. Flight test setup at the flight field is presented in the next section and in subsequent subsections a discussion on the sensor placement are discussed. Followed by an uncertainty analysis evaluation on the experimental procedure in the last section of this chapter.

## 4.1 Controller Design

### 4.1.1 Controller Design Considerations

Generally there are three approaches to multi-vehicle coordination that is seen in controls literature. These are leader-follower, behavioral, and virtual structure, with leader-follower and behavioral being probably most commonly known. Neither one is necessarily better than the other, they just each perform better doing different tasks than their counterpart.

In leader-follower one of the agents is designated as a leader, with the rest of the agents designated as followers. The leader tracks a pre-defined trajectory, and the followers track a transformed version of the leader's states. The advantage of leader following is that group behavior is directed by specifying the behavior of a single quantity: the leader. The disadvantage is that there is no explicit feedback to the formation. Another disadvantage is that the leader is a single point of failure for the formation.

In behavioral, several desired behaviors are prescribed for each agent. The basic idea is to make the control action of each agent a weighted average of the control for each behavior. Possible behaviors include collision avoidance, obstacle avoidance, goal seeking, and formation keeping. The advantage of the behavioral approach is that it is natural to derive control strategies when agents have multiple competing objectives. In addition, there is explicit feedback to the formation since each agent reacts according to the position of its neighbors. Another advantage is that the behavioral approach lends itself naturally to a decentralized implementation. The disadvantage is that the group behavior is said to “emerge”. In addition, it is difficult to analyze the behavioral approach mathematically and guarantee its group stability.

In a virtual structure the entire formation is treated as a single structure. The virtual structure can evolve as a rigid body in a given direction with some given orientation and maintain a rigid geometric relationship among multiple vehicles. The advantage of the virtual structure approach is that it is fairly easy to prescribe a coordinated behavior of the group. The disadvantage is that requiring the formation to act as a virtual structure limits the class of potential applications of this approach. Other disadvantages are that its current development lends itself to a centralized control implementation and re-configurability for time varying formation

### 4.1.2 Controller Design

For this swarm, the controller decided on is a virtual structure. The entire formation is treated as single rigid body, while the desired state for each agent is specified by assigning a corner in the virtual structure. As a result the agents maintain a rigid geometric relations. Graph theory was implemented by representing each Solo and the ground station as a node and the communication path between them as an edge. The ground station is the central node and all Solo communication to one another is routed through the ground station. This mapping is due to the Solos not having the ability on board to communicate directly. Looking back at equation 17, the centralized controller is made up of multiple critical parts. First there is c the coupling strength and F the feedback gain matrix. If these two values are properly set, then the system won’t have unbounded solutions. The adjacency matrix, $a_{ij}$ tells the controller which two nodes are communicating and it incorporates $e_{ij}$ which is an edge and is denoted by an ordered pair of vertices $(v_j, v_i)$. Whether the nodes are stationary or moving in time, the states of those nodes are communicated to the other nodes and controller, and all of the communication links are a value of $a_{ij}$. This entire action is part of what keeps the Solos in a coordinated formation and from colliding into one another. The figure below illustrates the trajectory tracking of three UAV in a virtual structure.

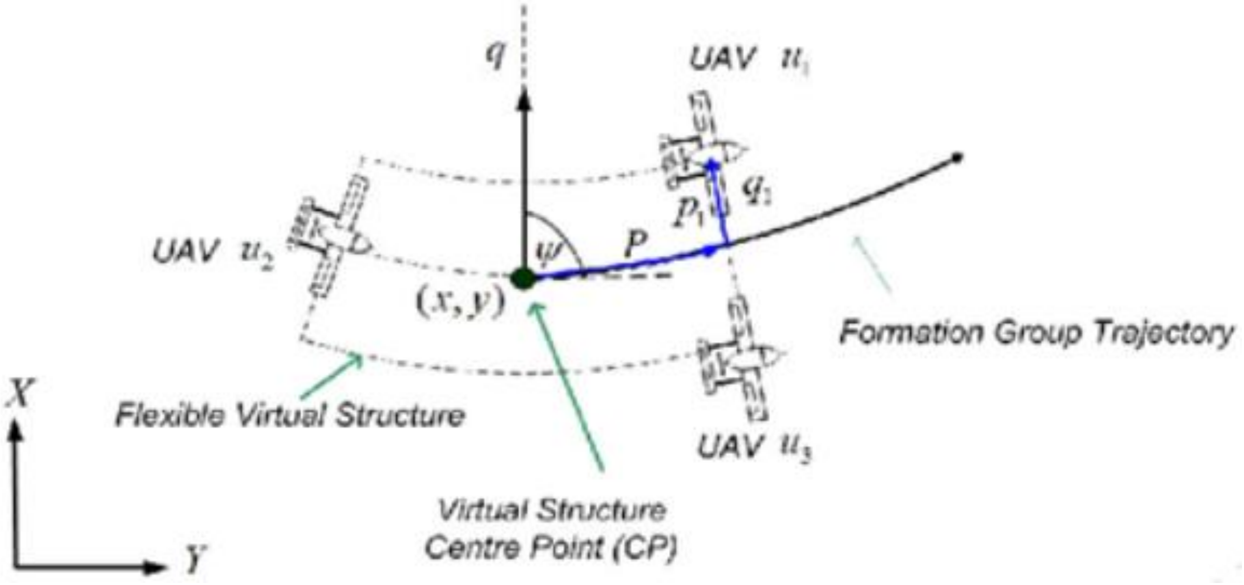


**Figure 14: Virtual Structure Trajectory Tracking**

A virtual structure was chosen due to its advantages meeting the necessary requirements for taking the atmospheric readings. These advantages being easy to execute coordinated behavior for various formations and maintaining a stable formation during maneuvers.

## 4.2 Simulation

The simulations were performed using ROS and the program Gazebo. ROS which stands for robot operating system is robotics middleware. Although it is not actually an operating system, it does provide many services like message passing and package management. ROS is geared toward Unix systems and utilizes C++, Python, and Lisp. All the code made for this swarm is written in Python. Gazebo is the robot simulation that was used for testing the controller and code before implementing it in the real world tests. Gazebo makes it possible to rapidly test algorithms in realistic scenarios. Due to the robust physics engine, complex indoor and outdoor environments can be simulated. Below in Figures 15-18 there are two different simulations being performed. In Figures 15 and 16 there are twenty simulated 3DR Solos starting off in a straight line. Once the swarm controller is activated to begin, the Solos begin forming a square formation two at a time as seen below.

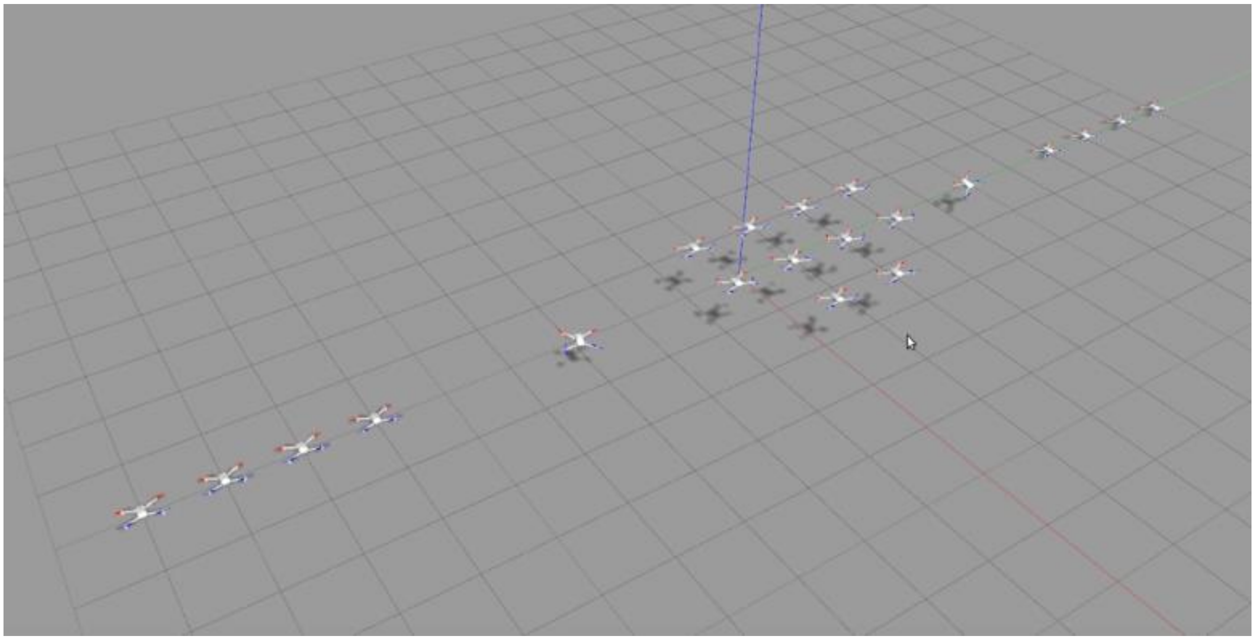


**Figure 15: Swarm moving from straight line formation to a square formation**

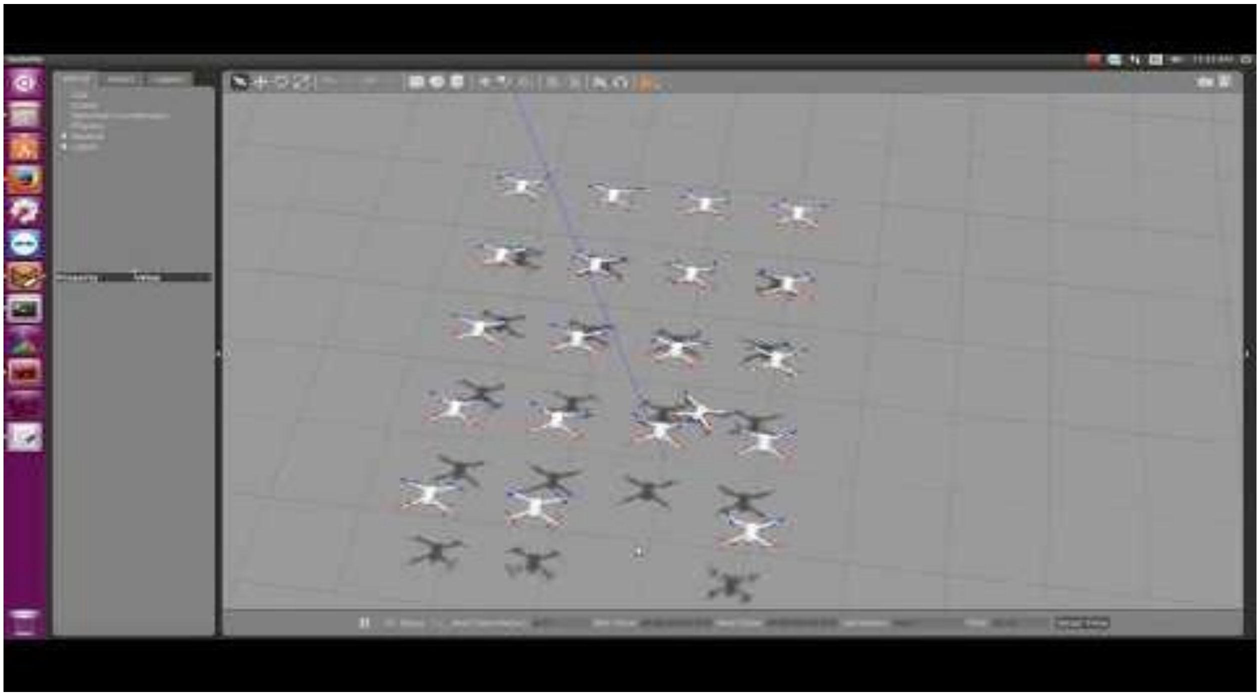


**Figure 16: Movement into square formation almost complete**

In Figures 17 and 18 the V formation is being tested. The Solos start off in a straight line and once the controller is activated to begin the flight, the Solos move into a V formation and begin moving forward while keeping a rigid formation.

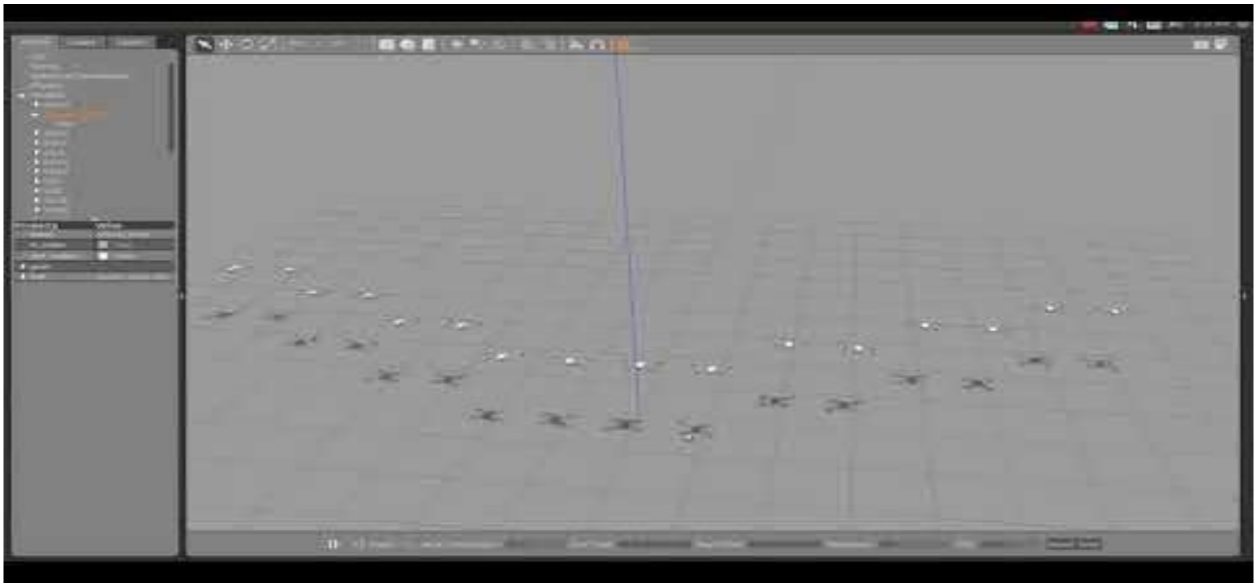


**Figure 17: V formation swarm**

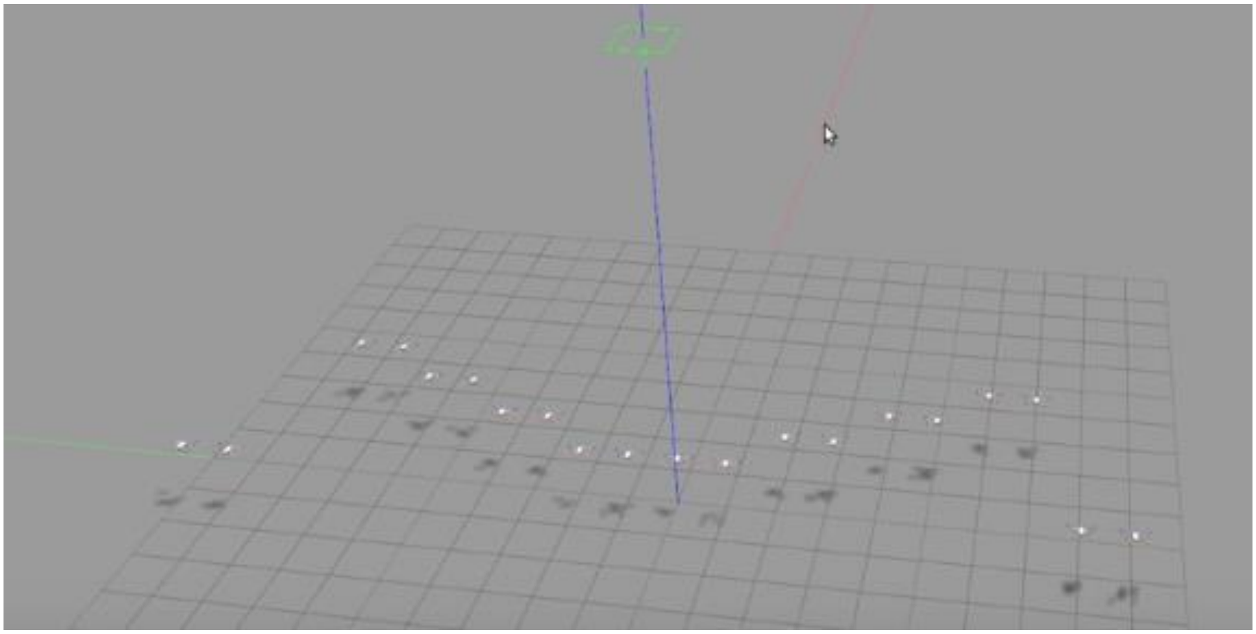


**Figure 18: V formation swarm in coordinated movement**

## 4.4 Flight Testing

Flight testing was performed at the Unmanned Aircraft Flight Station at Oklahoma State University is a dedicated UAS flight development and test facility 12 miles East of Stillwater that includes 2 runways (600 and 400 feet), an aircraft hangar and a state-of-the-art control room with monitoring capabilities. The UAFS includes a 1 mile by 1 mile flight area of unpopulated land to use for research, education and outreach in UAS.

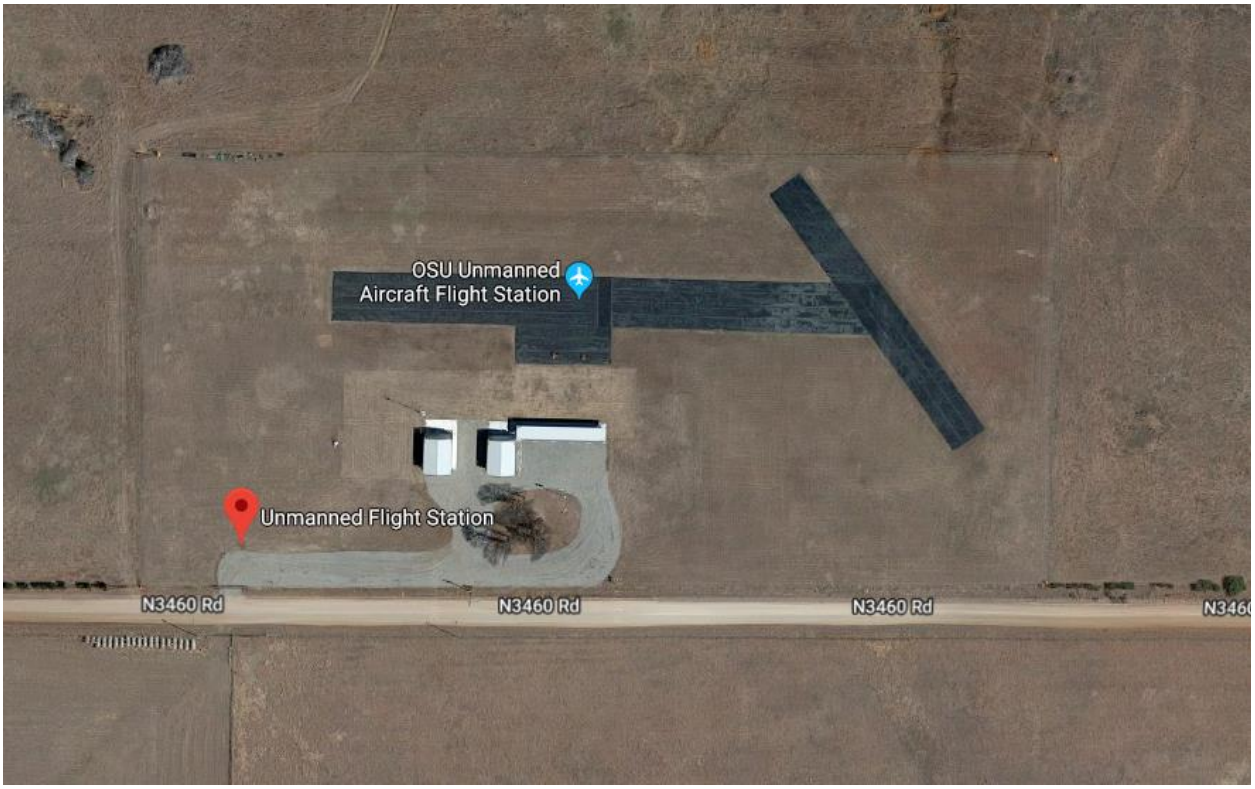
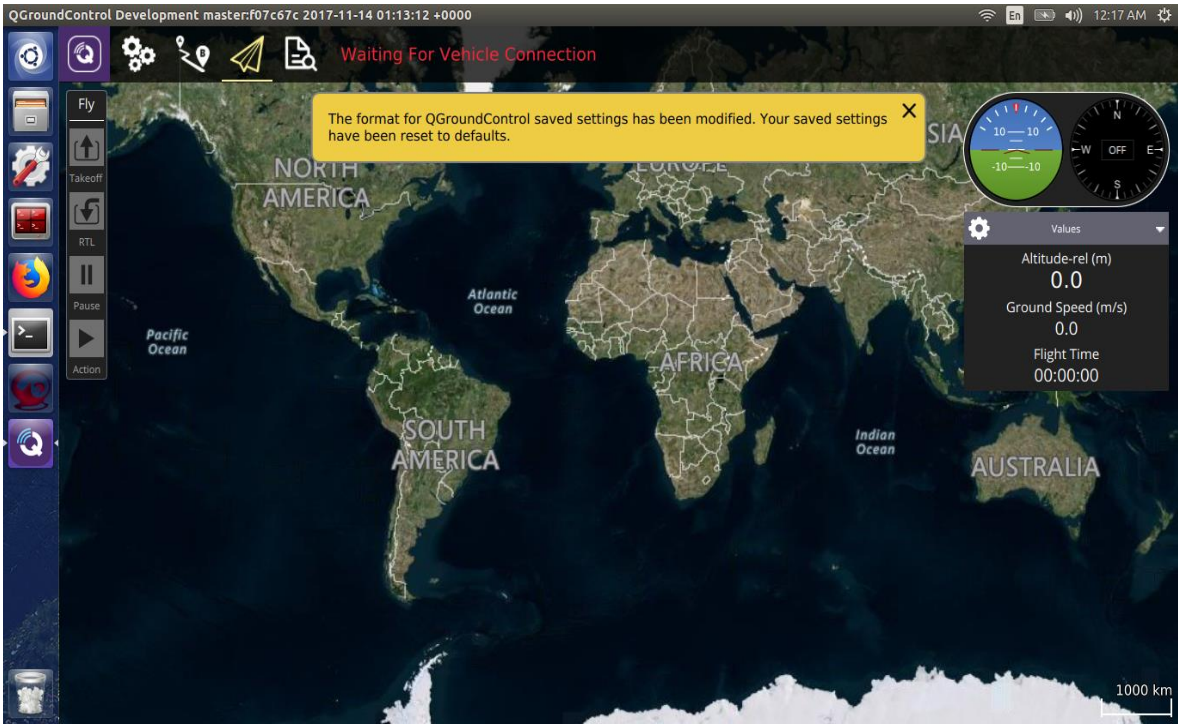


**Figure 19: OSU UAS Flight Station**



QGroundControl Development master:f07c67c 2017-11-14 01:13:12 +0000 12:17 AM

Waiting For Vehicle Connection

The format for QGroundControl saved settings has been modified. Your saved settings have been reset to defaults.

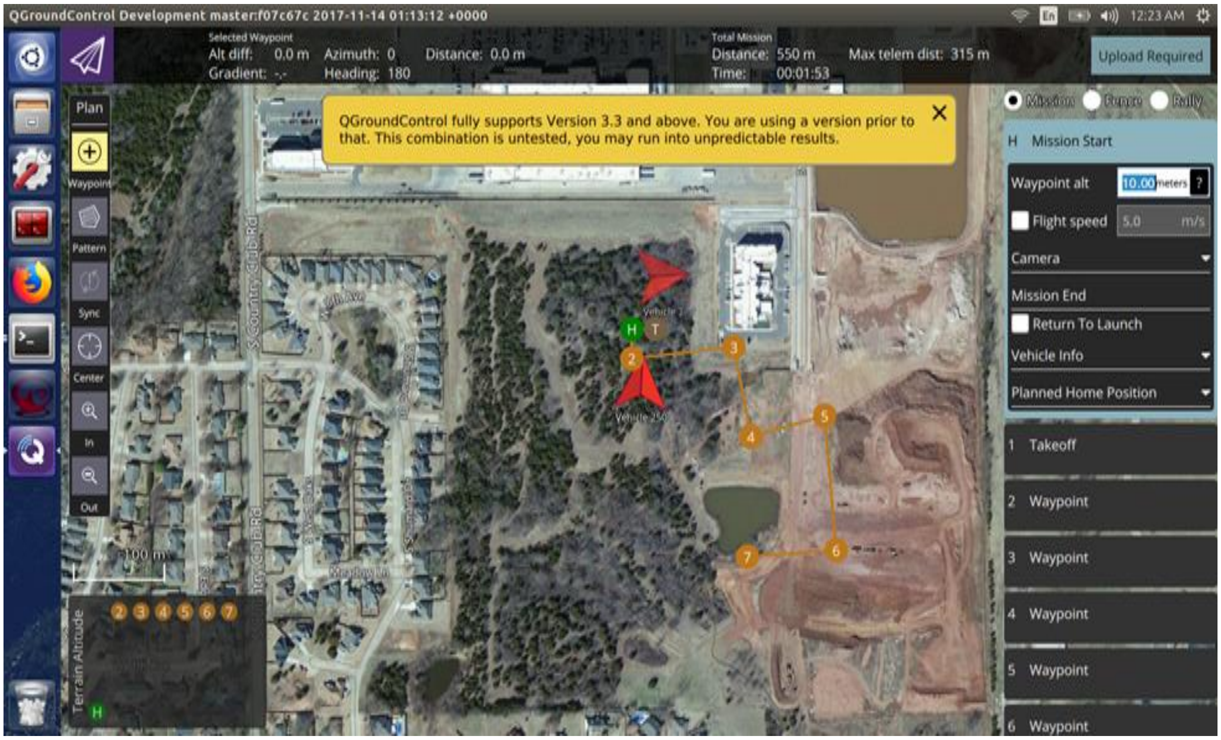
Values

Altitude-rel (m) 0.0

Ground Speed (m/s) 0.0

Flight Time 00:00:00

**Figure 20: Ground Station user interface**



QGroundControl Development master:f07c67c 2017-11-14 01:13:12 +0000 12:23 AM

Selected Waypoint

Alt diff: 0.0 m Azimuth: 0 Distance: 0.0 m

Gradient: -.- Heading: 180

Total Mission

Distance: 550 m Max telem dist: 315 m

Time: 00:01:53

Upload Required

QGroundControl fully supports Version 3.3 and above. You are using a version prior to that. This combination is untested, you may run into unpredictable results.

● Mission ○ Fence ○ Rally

H Mission Start

Waypoint alt 10.00 meters

□ Flight speed 5.0 m/s

Camera

Mission End

□ Return To Launch

Vehicle Info

Planned Home Position

1 Takeoff

2 Waypoint

3 Waypoint

4 Waypoint

5 Waypoint

6 Waypoint

**Figure 21: Ground Station waypoint selection**

### 4.4.1 Sensors and System Layout

Computers used were two Lenovos with i5 2.4GHz processors running 4GB of ram. Connected to that is an Adrupilot open source autopilot hardware with a 3DR GPS attached, and all stored in a 3D printed housing. A TP-Link dual band wireless router was used to supply a communication link between the command station and the Solos.

For the dropsondes a custom board was designed. Attached to that is a Teensy controller with SD card and Maestro GPS Receiver A2235-H. A BME 280 Environmental Sensor is used to measure pressure, temperature, and humidity. A 900 MHz Xbee radio module is attached to the custom board and used for wireless communication. The entire dropsonde is powered by a single 3.7V battery and packed inside a custom 3D printed case, with a fan attached to the back to pull air through for aspiration. The dropsonde is connected to the Solo's camera attachment and located on the bottom of the Solo.

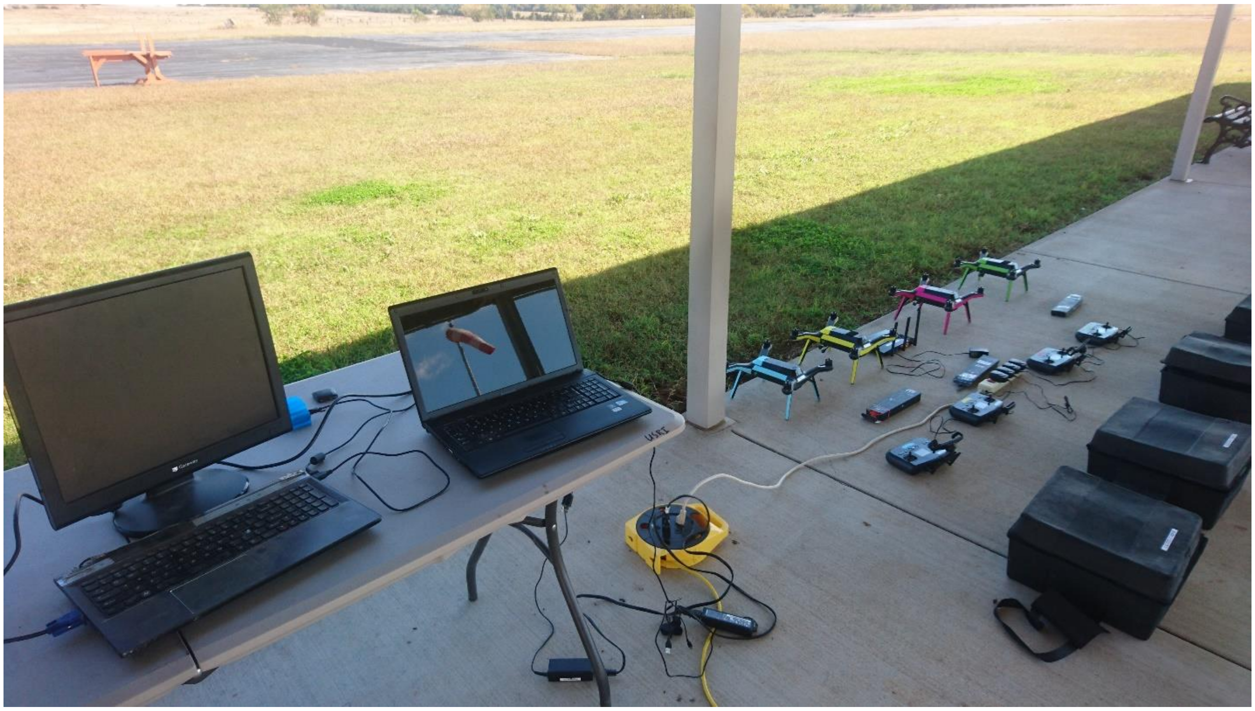


**Figure 22: Setup**

Figures 22 thru 27 show the swarm setup and components needed to successfully run the swarm. All is included minus the dropsonde and its components, shown in Figures 29 thru 34. Figure 22 shows a complete setup of what is needed. Figure 23 shows the ground station portion of the setup, with Figure 24 showing the four UAVs used and their communication links. Figure 25 gives a closer look at the two computers used to command the operation. One computer running the ground station user interface software, QGroundControl, which allows for the user to setup the swarm path using waypoints and swarm altitude; and the other running the Python scripts that makes up the architecture of the swarm. Some of the functions that can be adjusted by the user such as the virtual structure formation, distance between UAV, and UAV collision control. Figures 26 and 27 shows the Ardupilot and GPS for the ground station, this in a way allows for the UAV to see the ground station as a piece in the framework of the swarm and gives the ground station the necessary hardware to pilot the UAV with the autopilot and a sense of direction with the GPS. Figure 28 is the router that creates the necessary networks for the UAV hand held controllers and the ground station. Note the most important communication distance in the swarm is the 300m distance between the Solo controllers and the Solos themselves.

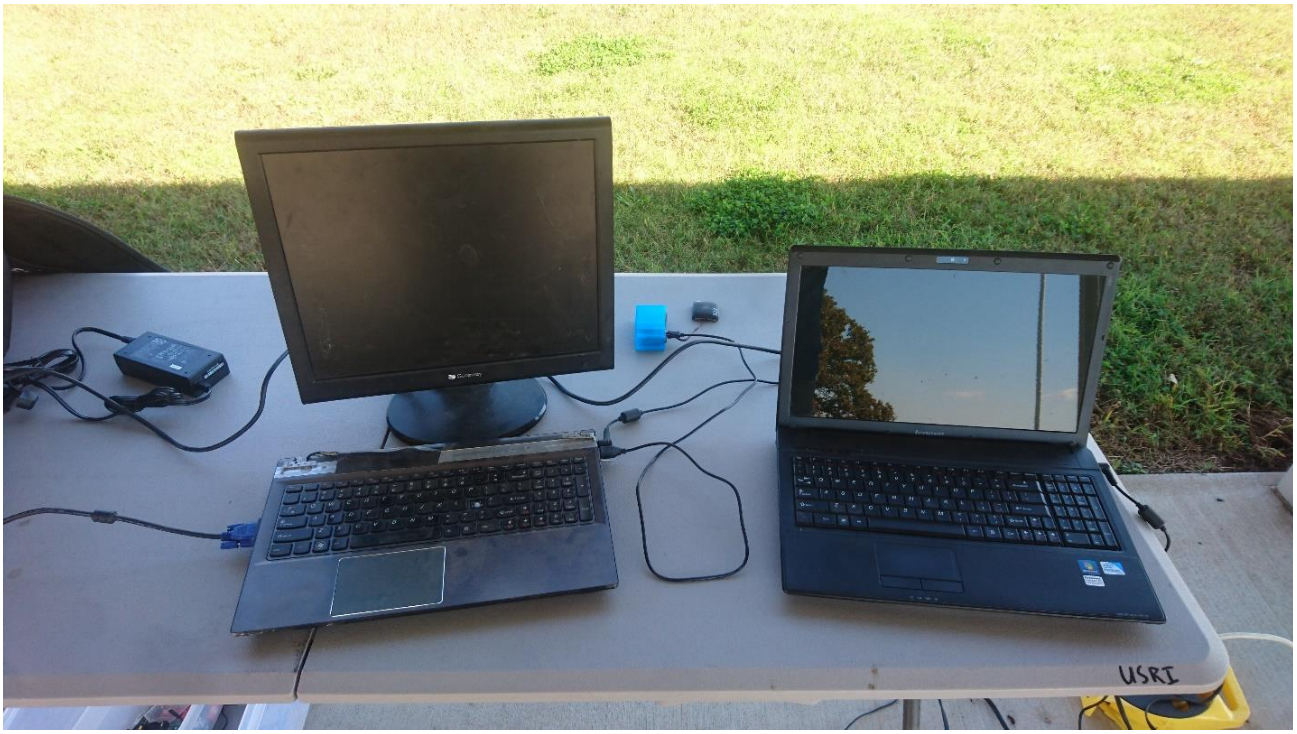


**Figure 23: Setup 2**

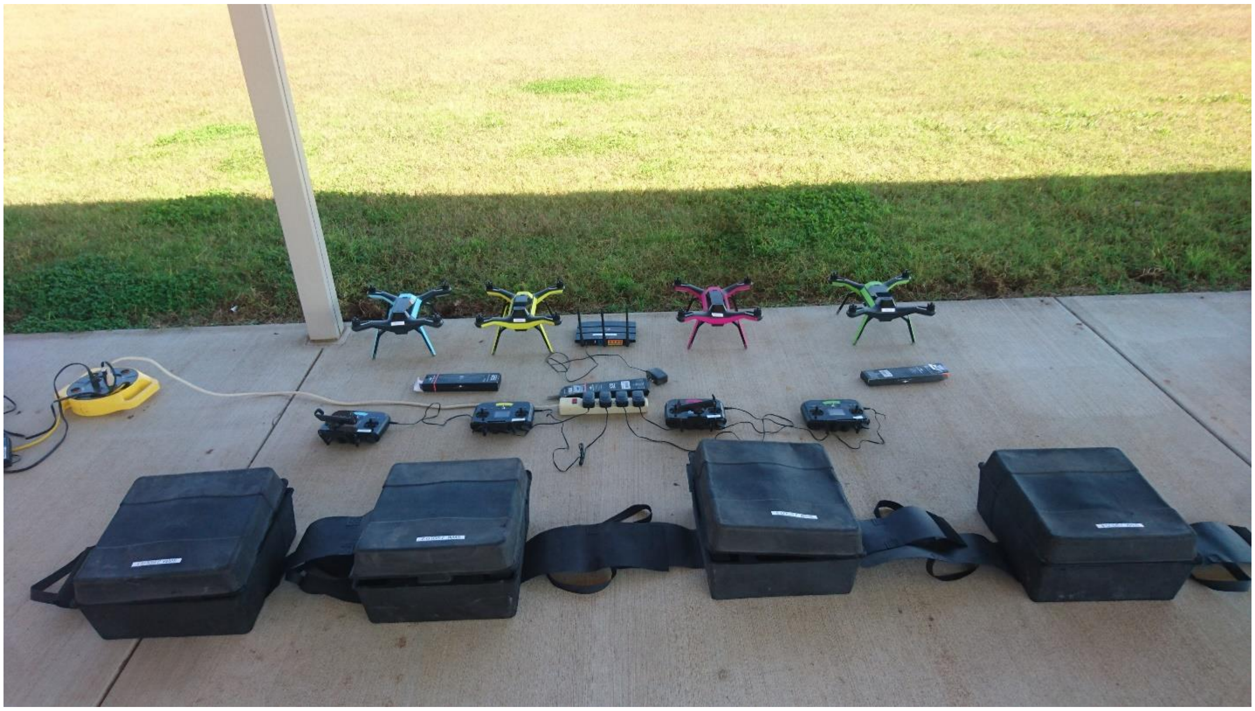


**Figure 24: Setup 3**

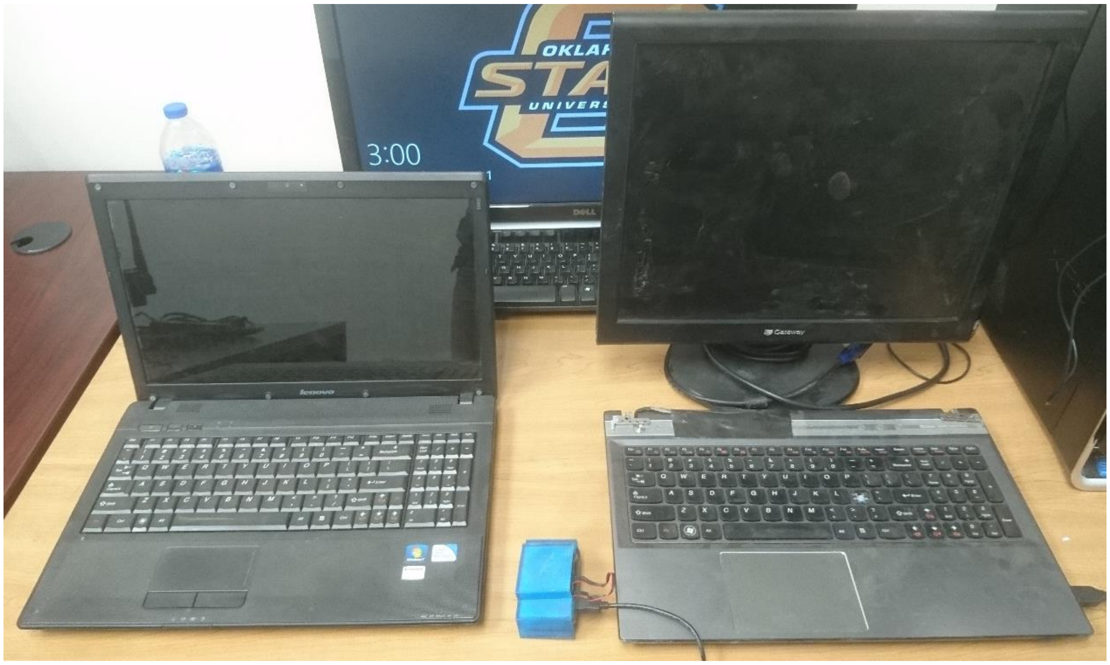


**Figure 25: The two swarm computers**

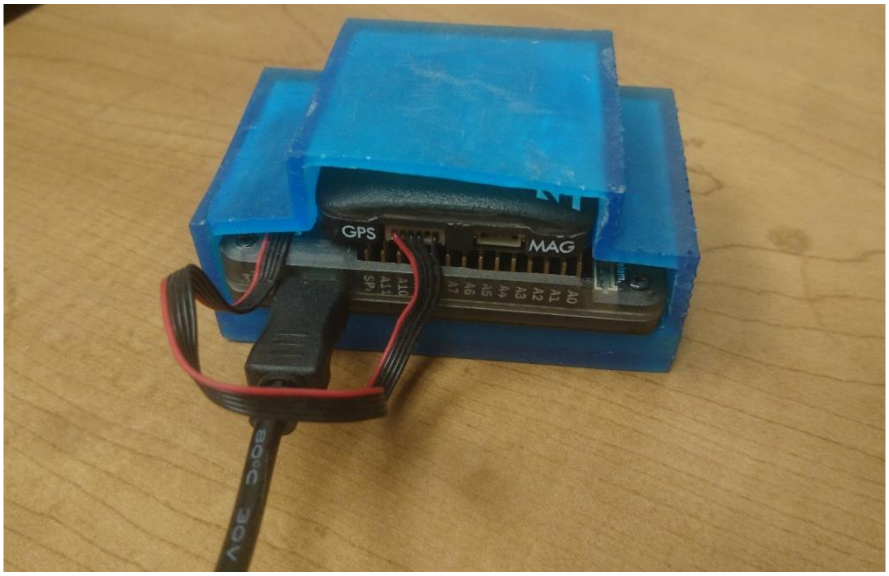


**Figure 26: Ardupilot and GPS in housing**

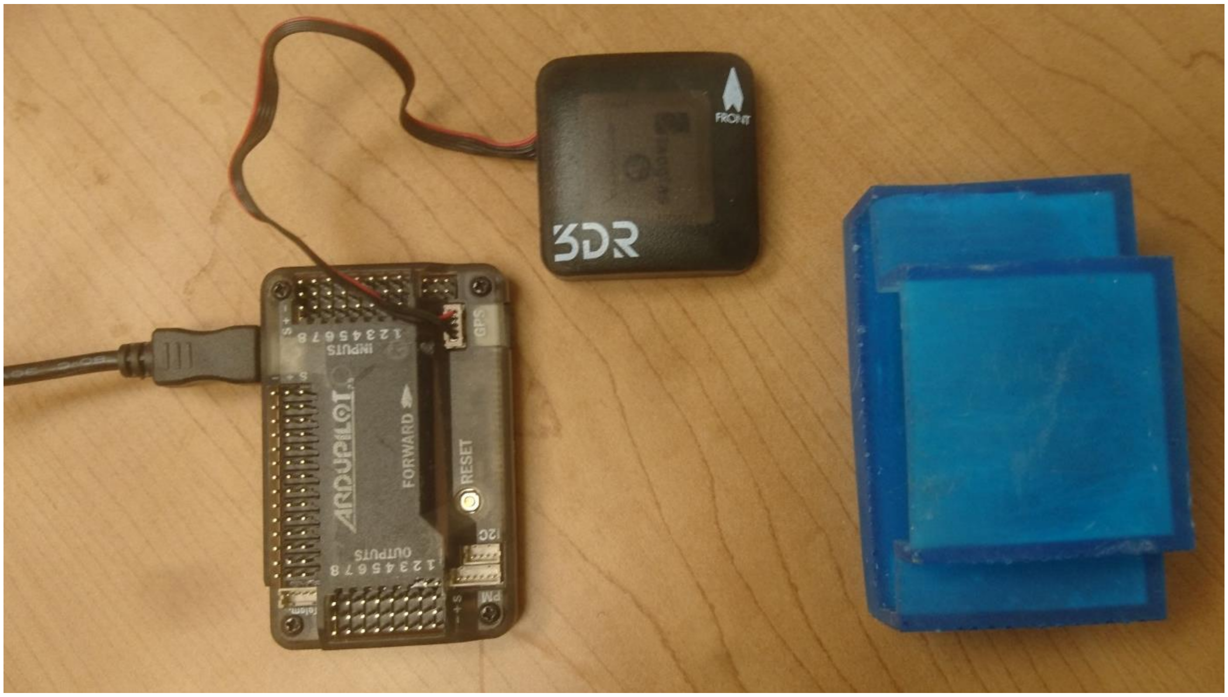


**Figure 27: Ardupilot and GPS**

**Figure 28: Solo swarm router**

Figures 29 thru 34 make up the dropsonde hardware and assembly. The dropsondes were not a major focus of the thesis but really just used as an example of possible sensor application for this swarm. It and the data obtained from them were only used and presented in this work as an example.

**Figure 29: BME 280 Environmental Sensor**

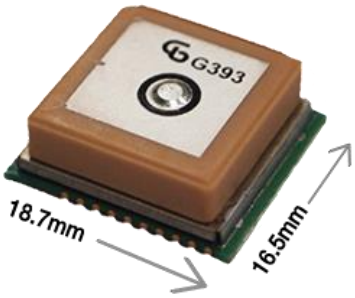


**Figure 30: Maestro GPS Receiver A2235-H**

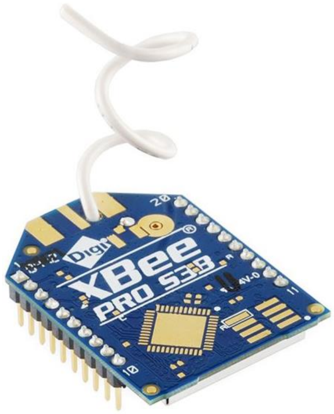


**Figure 31: 900 MHz Xbee radio module**

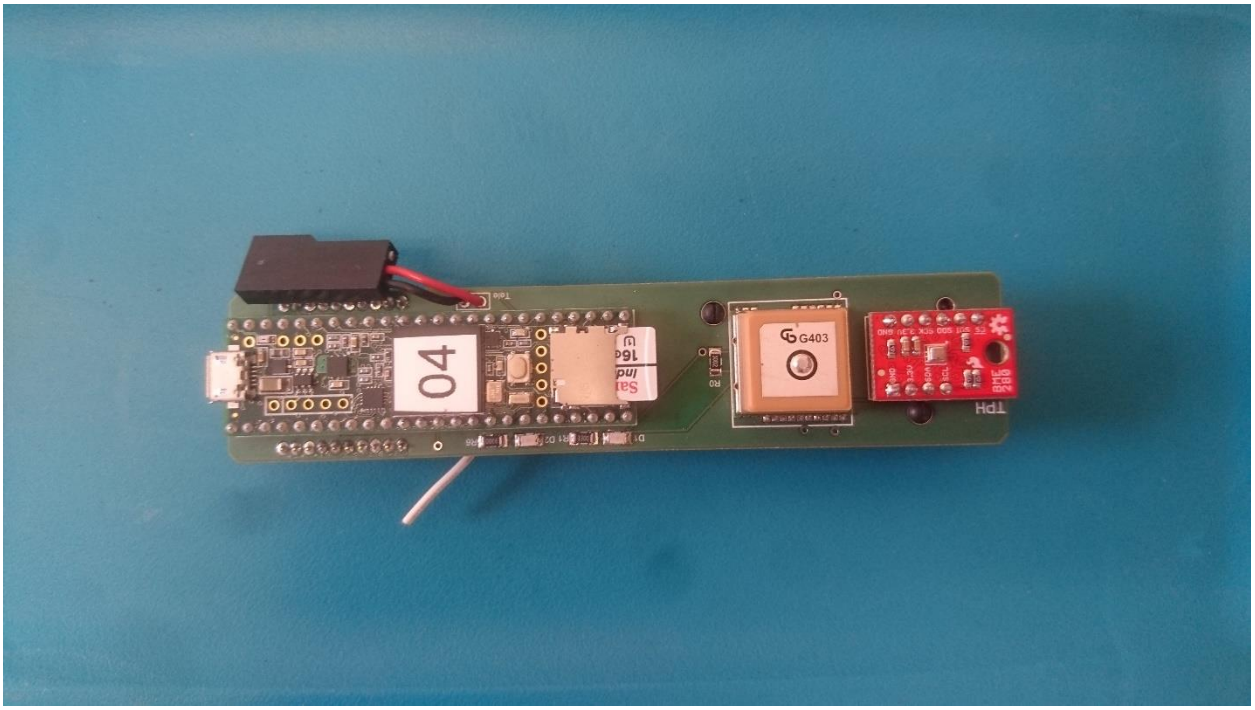


**Figure 32: Dropsonde Top View**

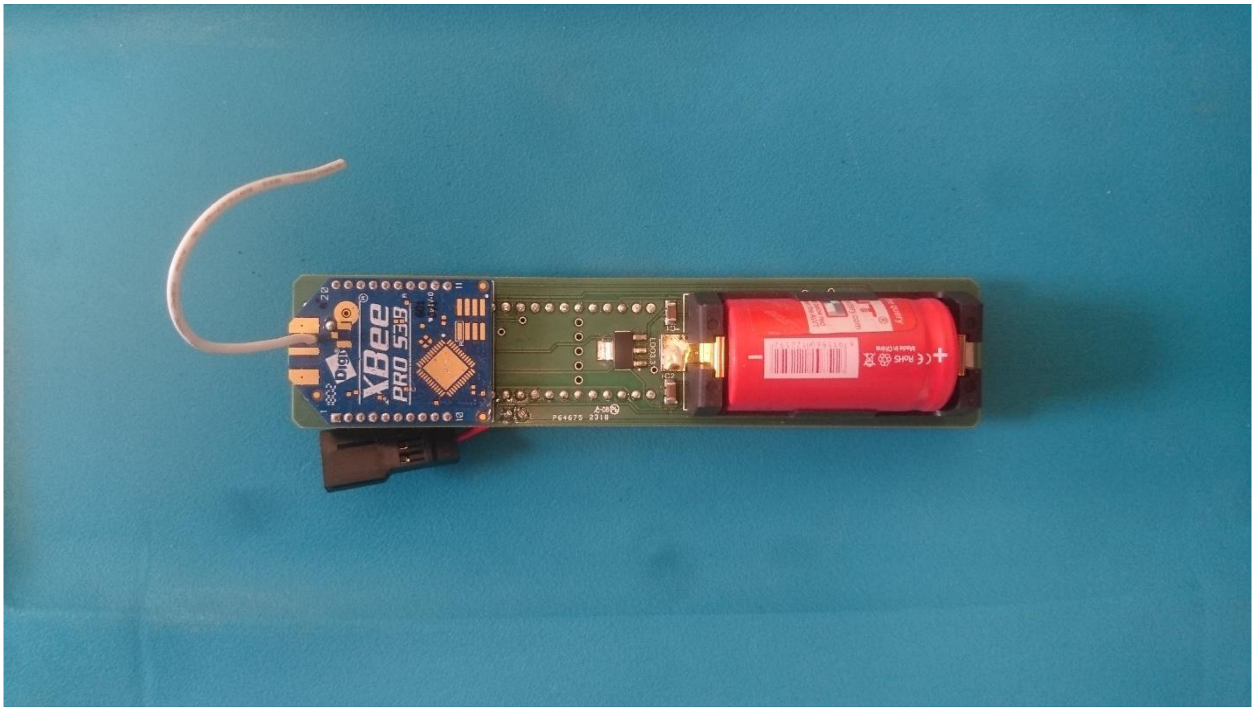


**Figure 33: Dropsonde Bottom View**

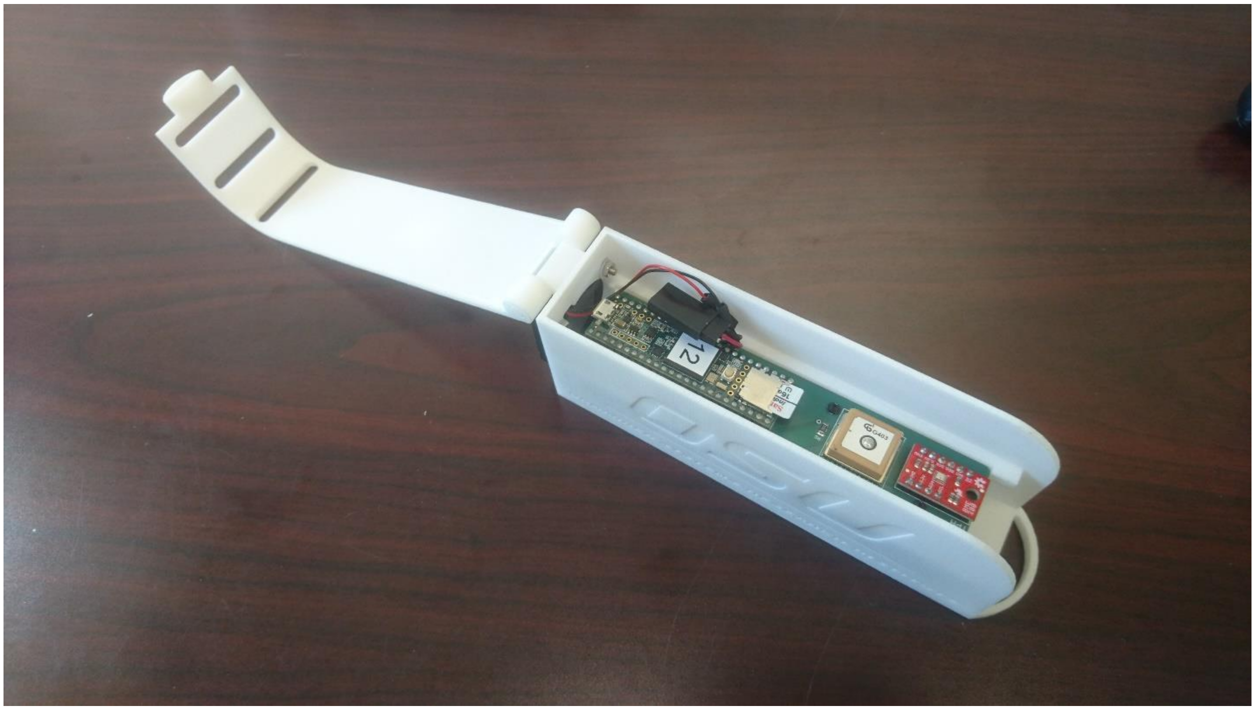


**Figure 34: Dropsonde housing open**

# CHAPTER V

## 5. RESULTS

The results presented in this section are derived from the methodology and experimental setup previously discussed. The results are presented in correlation to their relevance to the objectives and goals of this research. Since the main focus was swarming the following chapter will firstly discuss those results. Then followed by the results from supplemental tests and a discussion on flight testing.

### 5.1 Swarm Results

As for results, it has been proven that the coordinated swarm is possible, as up to five 3DR Solo quadrotors have been flown in a coordinated swarm (Figure 35). The UAV follow a path set by the command station along with a fixed shape that can be modified from the command station. Distances between the UAV can be adjusted as well as altitude and X, Y positions resulting in essentially whatever 3-Dimensional shape that is desired.

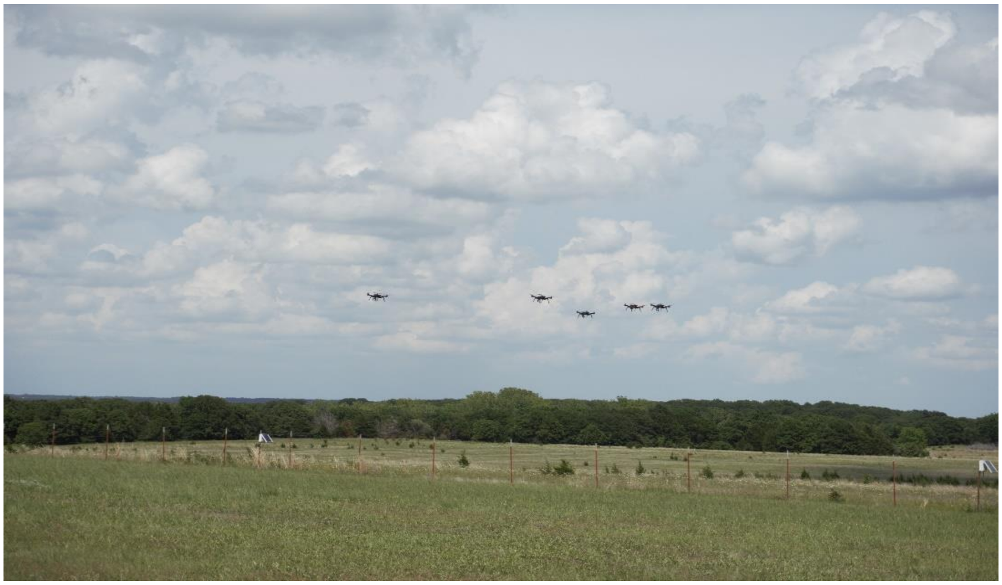


**Figure 35: Swarm of five 3DR Solos**

Figure 36 shows a swarm of four Solos in an X, Y square shape. One shape that is desired would be a tower formation that results in the UAV being stacked one over the other. This would simulate what a weather tower does, but because the UAV can move, the swarm can position itself in the optimal position for the best data results.

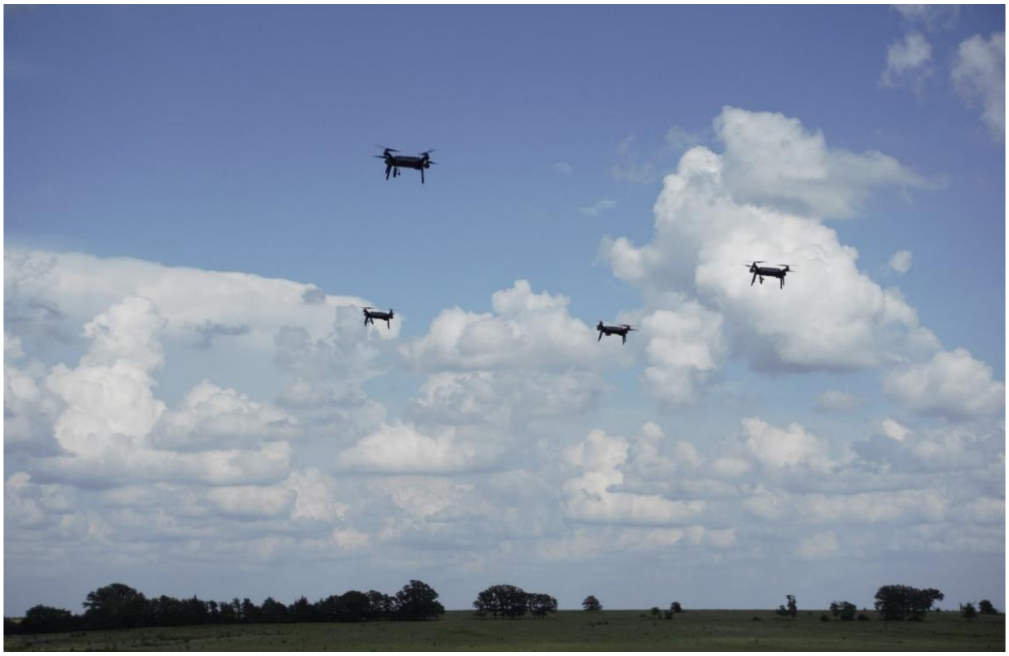


**Figure 36: Swarm of four 3DR Solos in square formation**

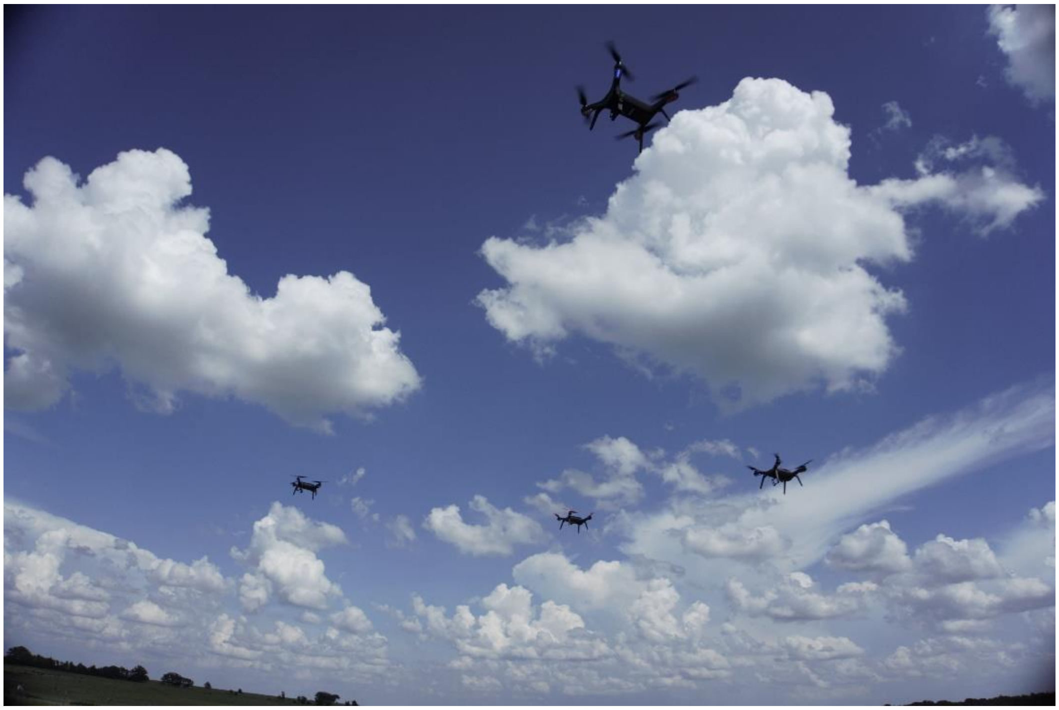


**Figure 37: Swarm of four 3DR Solos**

Though pictures give a nice visual representation of what the swarm structure looks like. Another way to see what the structure is doing is with telemetry data. This can be done plotting different things like latitude, longitude, altitude, time, and other characteristics if desired. The following figures are of three different flight profiles: tower formation, level box formation, and box formation with altitude changes. The altitude, latitude and longitude is plotted to show the position for each Solo and the time is plotted for the course of the flight profile. Using these and plotting them in different arrangements gives each Solos' position over time. The weather during these flights were sunny with a temperature around 74 degrees Fahrenheit, and a wind speed between 5 and 10 knots in the East/Northeast direction. The four Solos were spaced out at least 10 yards away from each other on the flight field runway at the start, then when the command was executed the Solos began their arranged formation and flight sequence.

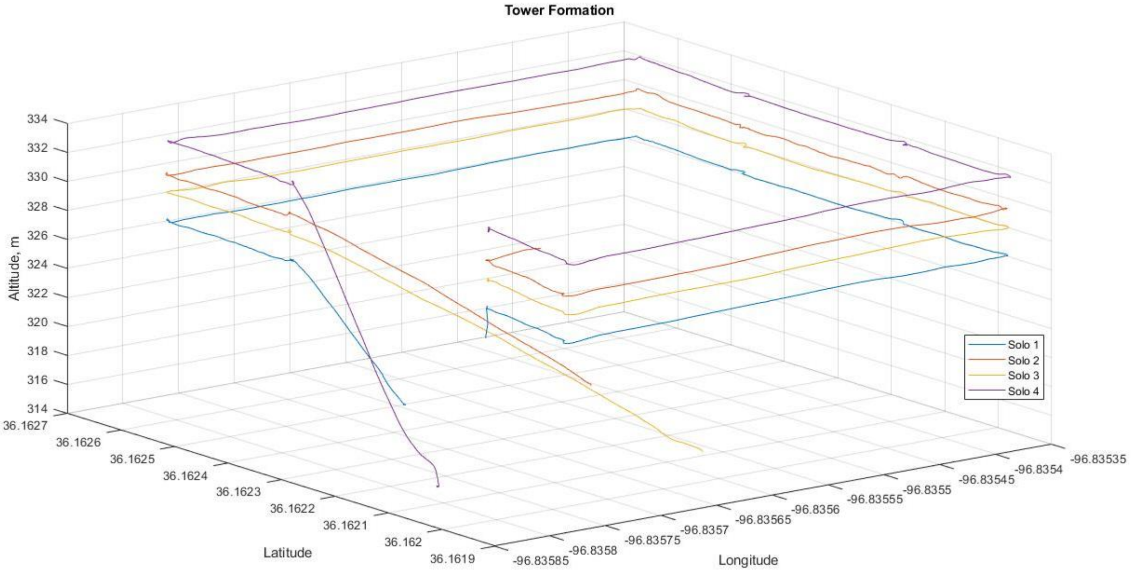


**Figure 38: Tower Formation Flight Profile**

The first formation, and arguably the most interesting for this research due to the similarities of it with a weather tower, is the tower formation. Here in Figure 38 it shows a 3D representation of the flight plotting latitude vs longitude vs altitude, with altitude being in meters. In Figure 39 latitude vs longitude is plotted for each Solo so it can be seen if the Solos stray from the formation or path. Overall based on the plot, the Solos keep a fairly rigid formation and even when there is a slight offset in the line path, likely due to cross wind, the Solos still move as one structure. The virtual structure center for this formation and in reference to Figure 39 would be in the middle of the group of lines, and seeing these lines stacked gives a good indication that the Solos are not straying much or often from their path and structure. Next Figure 40 is plotted with altitude vs time, and this gives a view of the flight path from a side 2D approach. Solo 1, 2, 3, and 4 is plotted and shown keeping an altitude of 327m, 329m, 330m, and 332m respectively. There is a slight increase and decrease in the lines due to the environment acting on the UAV. With these altitudes this would put the virtual center at about 329.5m in altitude. For this flight a spacing of 2m was used for the space between Solos 1 and 2 and Solos 3 and 4. The space between Solo 2 and 3 was set to 1 meter. Based on these inputs and the shown outputs, the Solos are not straying from the path or structure.

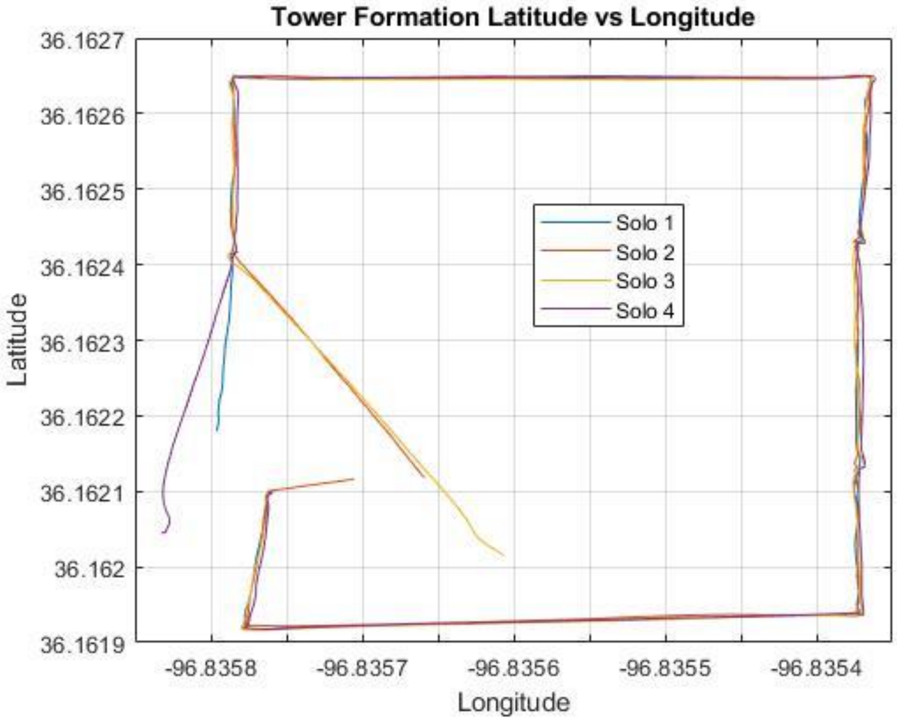


**Figure 39: Tower Formation Latitude vs Longitude**

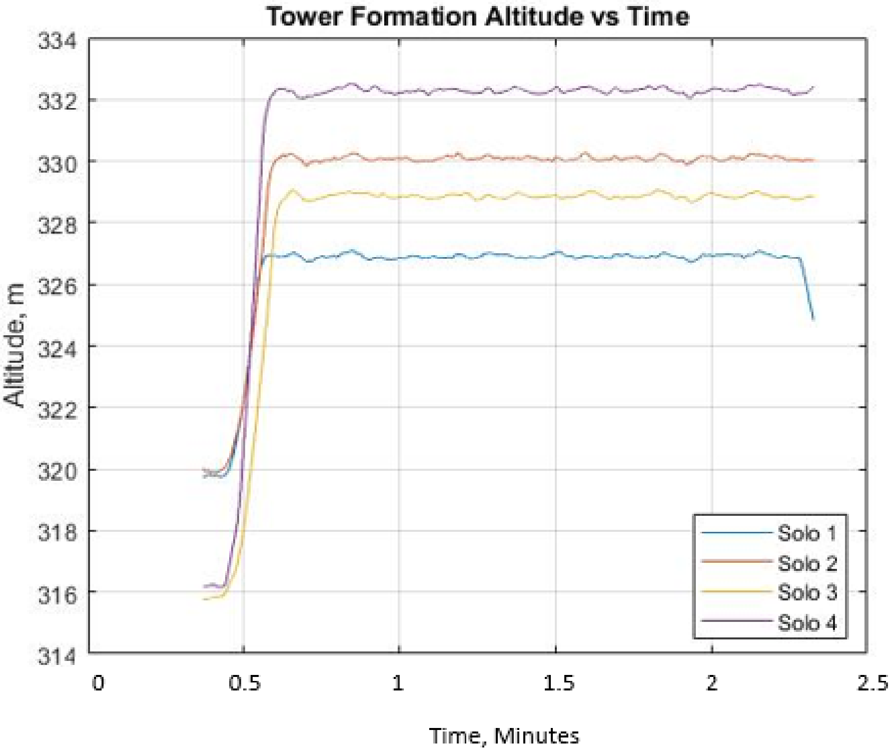


**Figure 40: Tower Formation Altitude vs Time**

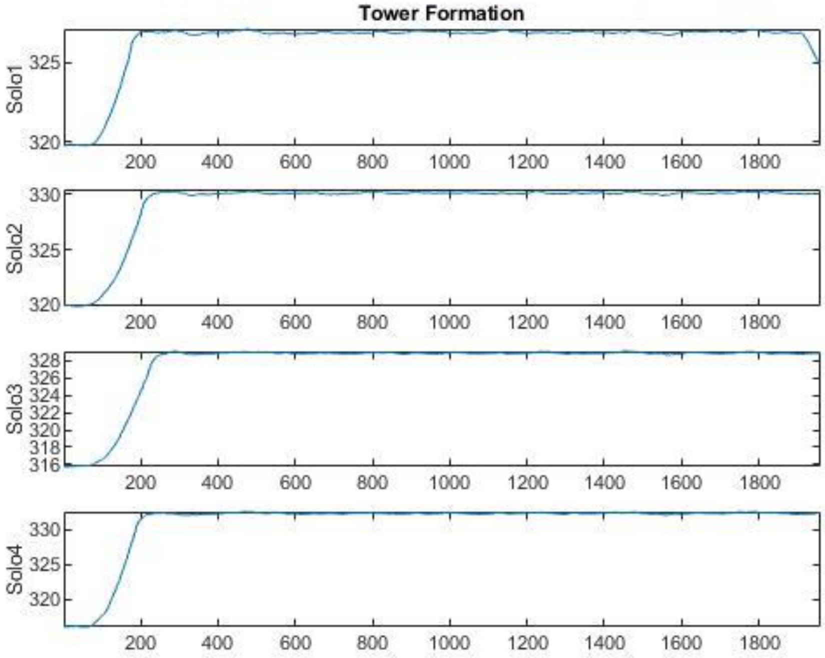


**Figure 41: Time stamped altitudes for each Solo in tower formation**

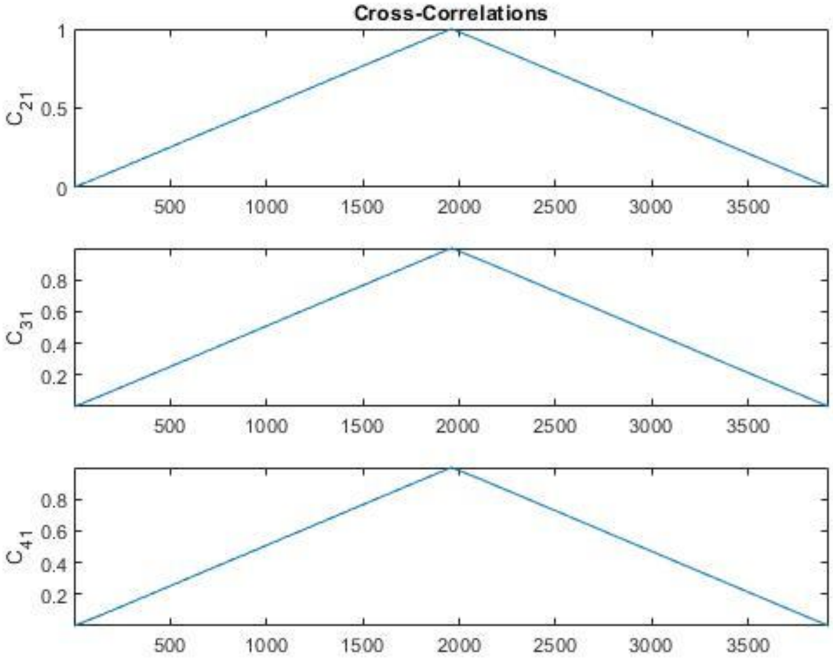


**Figure 42: Cross-Correlated Altitudes for Tower Formation**

Figures 41 and 42 show the time stamped altitudes for the tower formation and those correlated altitudes with respect to Solo 1. The cross correlation was done correlating Solo 2 to Solo1, Solo 3 to Solo 1, and Solo 4 to Solo 1. Each cross correlation peaks to 1 meaning that there is high correlation between all the sets of data.

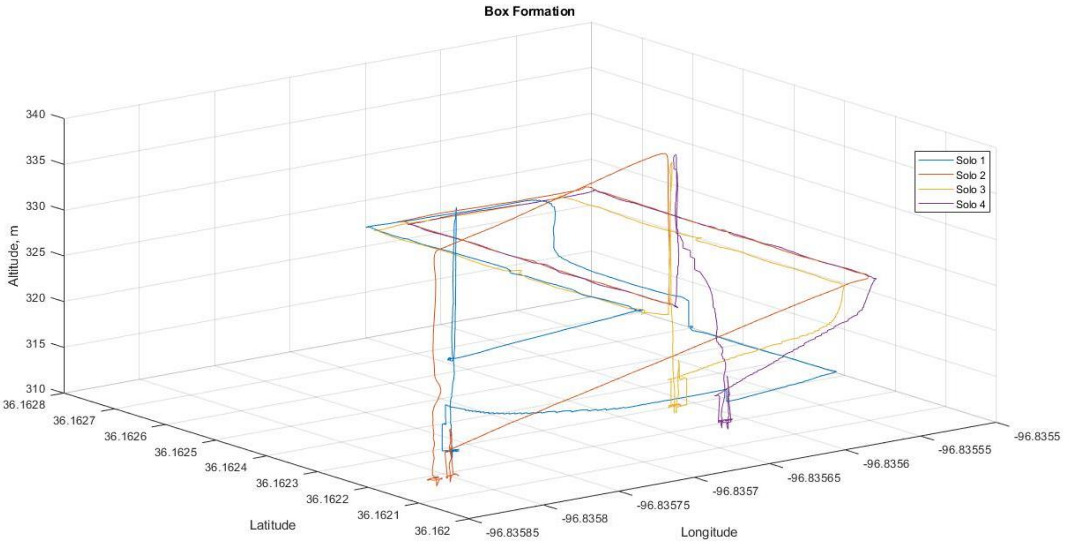


**Figure 43: Box Formation Flight Profile**

The next formation is the box formation which consist of the four Solos making up the four corners of a level horizontal box shape. Here in Figure 43 it shows a 3D representation of the flight plotting latitude vs longitude vs altitude, with altitude being in meters. In Figure 44 latitude vs longitude is plotted for each Solo so it can be seen if the Solos stray from the formation or path. Overall based on the plot, the Solos keep a fairly rigid formation in regards to latitude and longitude. The virtual structure center for this formation and in reference to Figure 44 would run down the middle of the two sets of parallel lines. Next Figure 45 is plotted with altitude vs time, and this gives a view of the flight path from a side 2D approach. Solo 1, 2, 3, and 4 is plotted and for the box portion of the flight they keep an altitude of about 323m. However, Solo 1 shows a time delay in the flight likely due to a software glitch, possibly a bad connection with Solo 1. There is a slight increase and decrease in the lines due to the environment acting on the UAV. Based on these inputs and the shown outputs, the Solos are not straying from the path or structure.

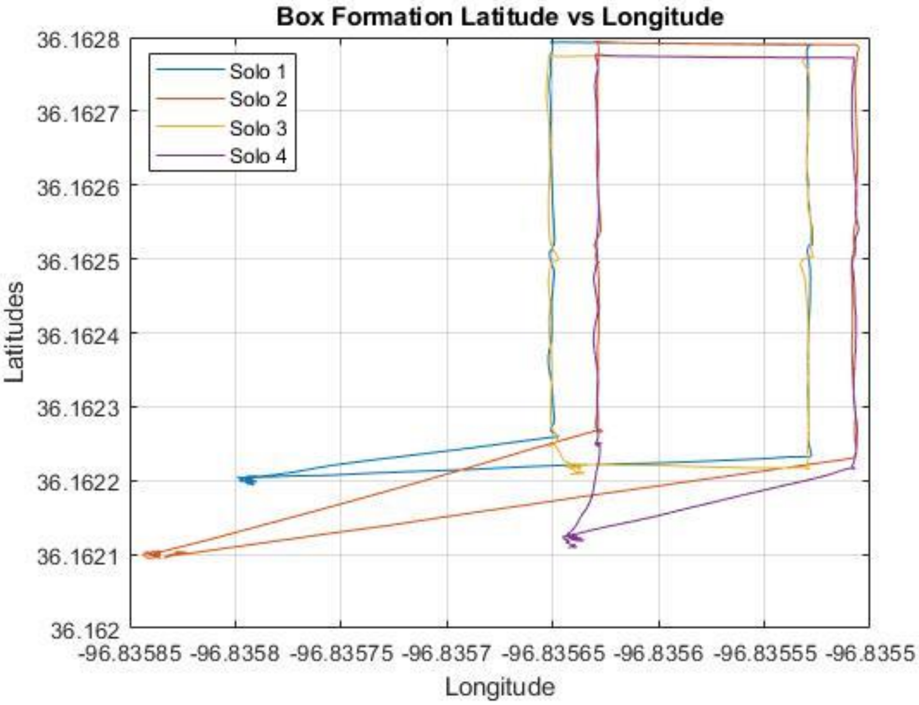


**Figure 44: Box Formation Latitude vs Longitude**

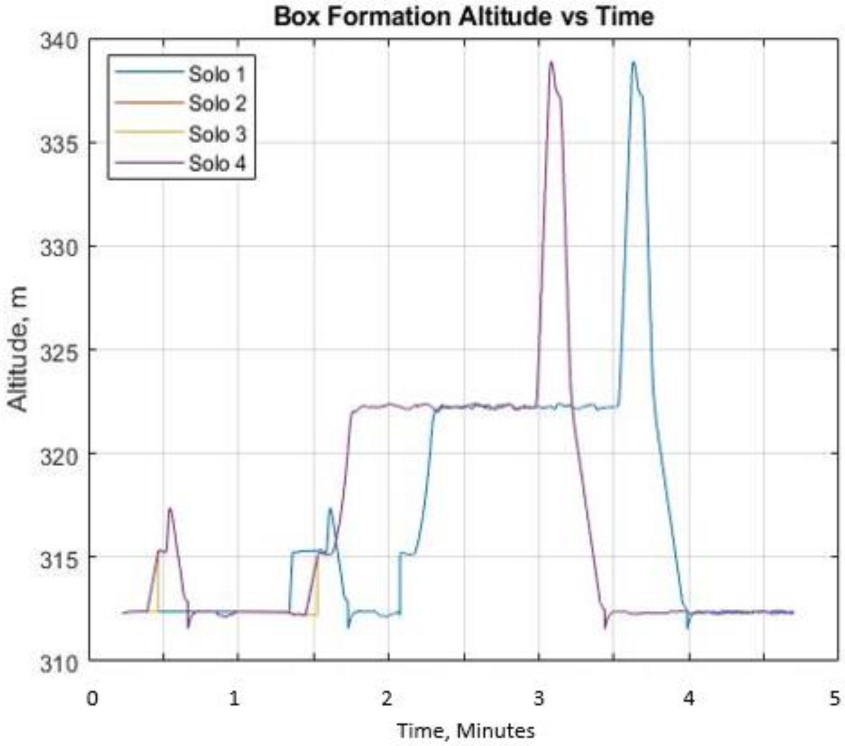


**Figure 45: Box Formation Altitude vs Time**

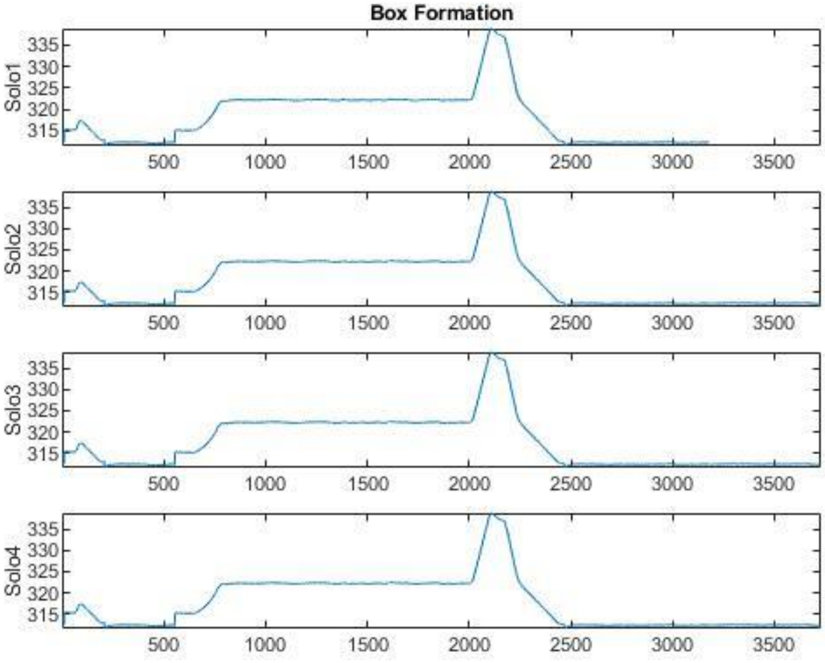


**Figure 46: Time stamped altitudes for each Solo in box formation**

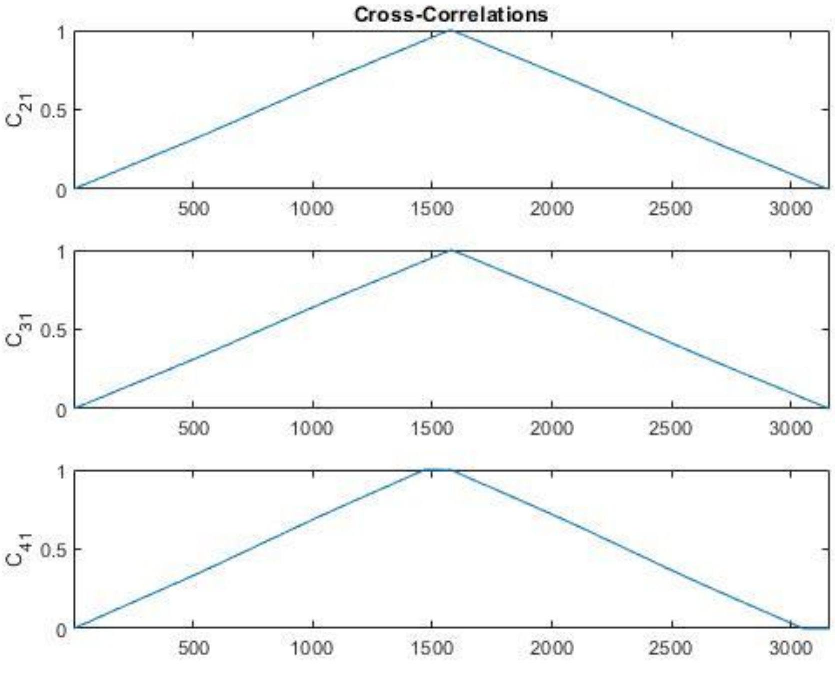


**Figure 47: Cross-Correlated Altitudes for Box Formation**

Figures 46 and 47 show the time stamped altitudes for the box formation and those correlated altitudes with respect to Solo 1. The cross correlation was done correlating Solo 2 to Solo1, Solo 3 to Solo 1, and Solo 4 to Solo 1. Each cross correlation peaks to 1 meaning that there is high correlation between all the sets of data.

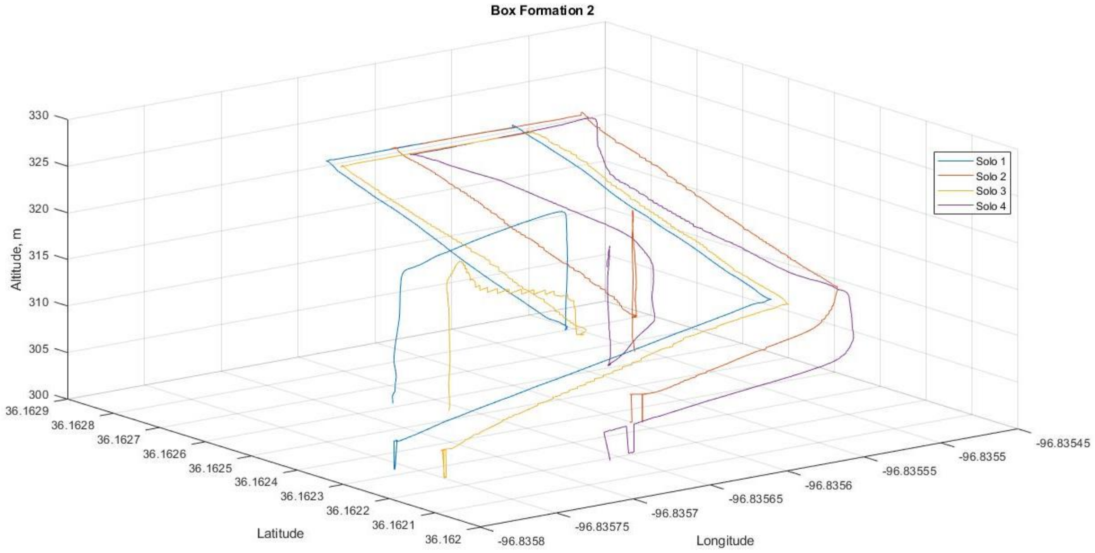


**Figure 48: Box Formation 2 Flight Profile**

The final formation is another variation of the box formation. It consists of the Solos still being in a horizontal box shape, but changing in altitude together. Here in Figure 48 it shows a 3D representation of the flight plotting latitude vs longitude vs altitude, with altitude being in meters. In Figure 49 latitude vs longitude is plotted for each Solo so it can be seen if the Solos stray from the formation or path. Overall based on the plot, the Solos keep a fairly rigid formation and even when there is a slight offset in the line path, likely due to cross wind, the Solos still move as one structure. The virtual structure center for this formation and in reference to Figure 49 would run down the middle of the two sets of parallel lines which means the Solos are not straying much or often from their path and structure. Next Figure 50 is plotted with altitude vs time, and this gives a view of the flight path from a side 2D approach. Solo 1, 2, 3, and 4 is plotted and shown rising to an altitude together, from the virtual center, to an altitude of about 323m and dropping back down to about 313m. For this flight Solo 4 has a slight time delay, again that's likely due to a software glitch, possibly a bad connection with Solo 4. A few ways to fix this would be to either start the flight over or increase the wait time at waypoint 1 so the UAV can reach their positions. There is a slight increase and decrease in the lines due to the environment acting on the UAV. Based on these inputs and the shown outputs, the Solos are not straying from the path or structure.

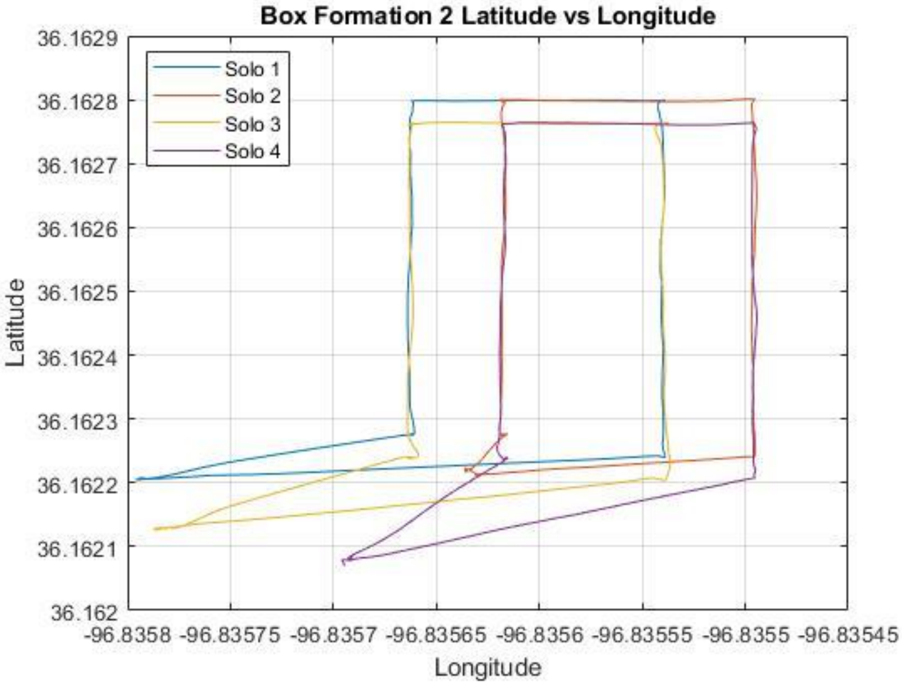


**Figure 49: Box Formation 2 Latitude vs Longitude**

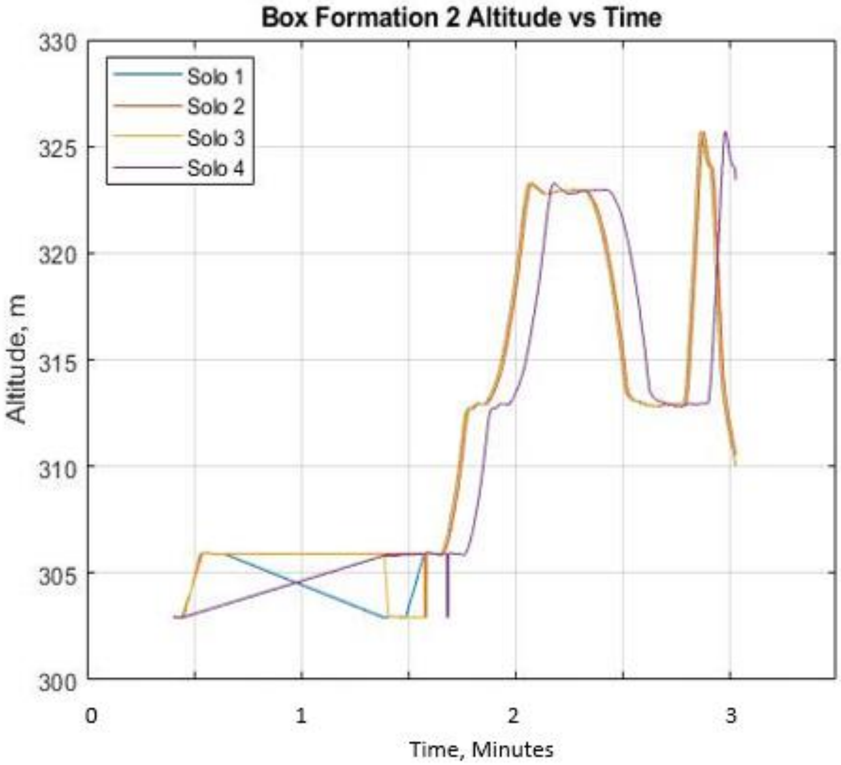


**Figure 50: Box Formation 2 Altitude vs Time**

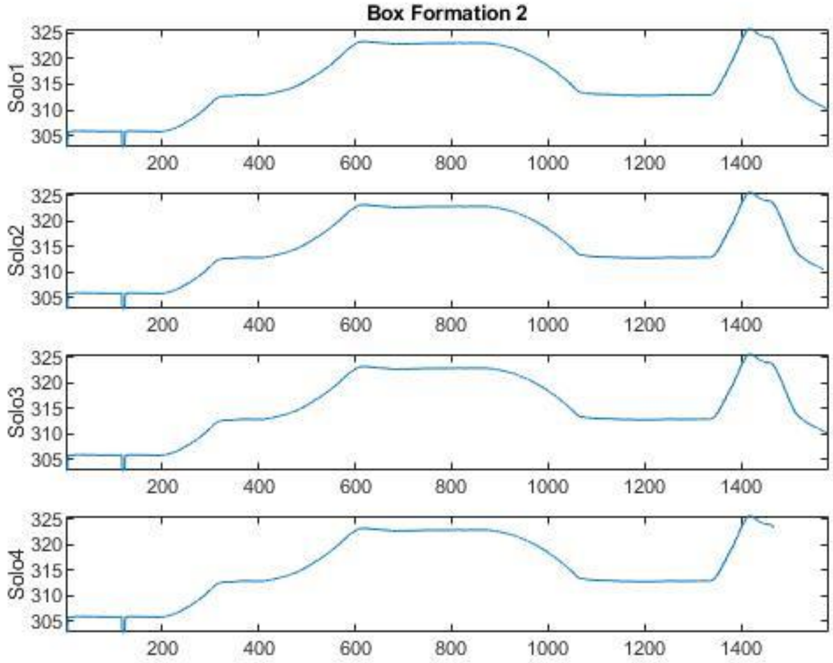


**Figure 51: Time stamped altitudes for each Solo in box formation 2**

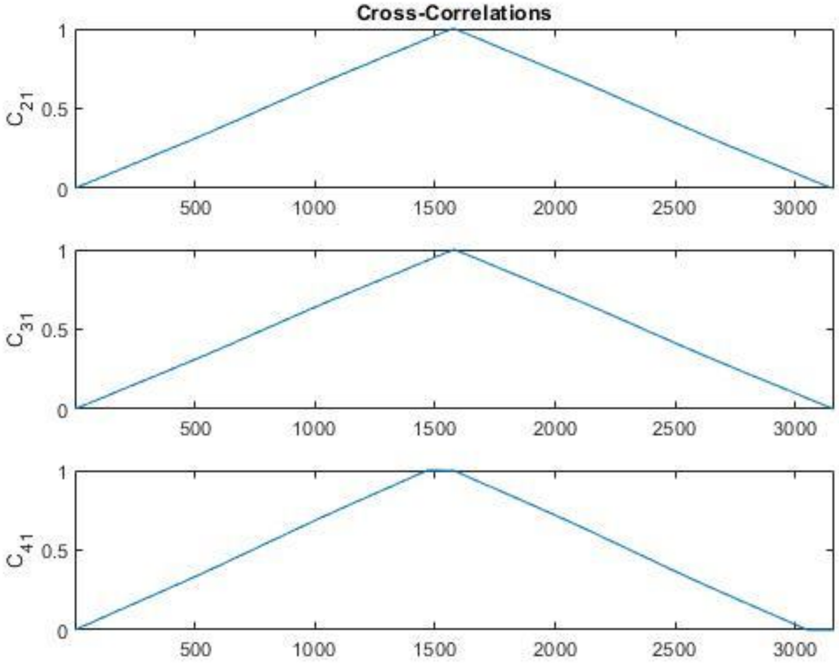


**Figure 52: Cross-Correlated Altitudes for Box Formation 2**

Figures 51 and 52 show the time stamped altitudes for the box formation 2 and those correlated altitudes with respect to Solo 1. The cross correlation was done correlating Solo 2 to Solo1, Solo 3 to Solo 1, and Solo 4 to Solo 1. Each cross correlation peaks to 1 meaning that there is high correlation between all the sets of data.

The next steps include attaching the dropsonde sensors to the Solos, which will record temperature, pressure, humidity and GPS.

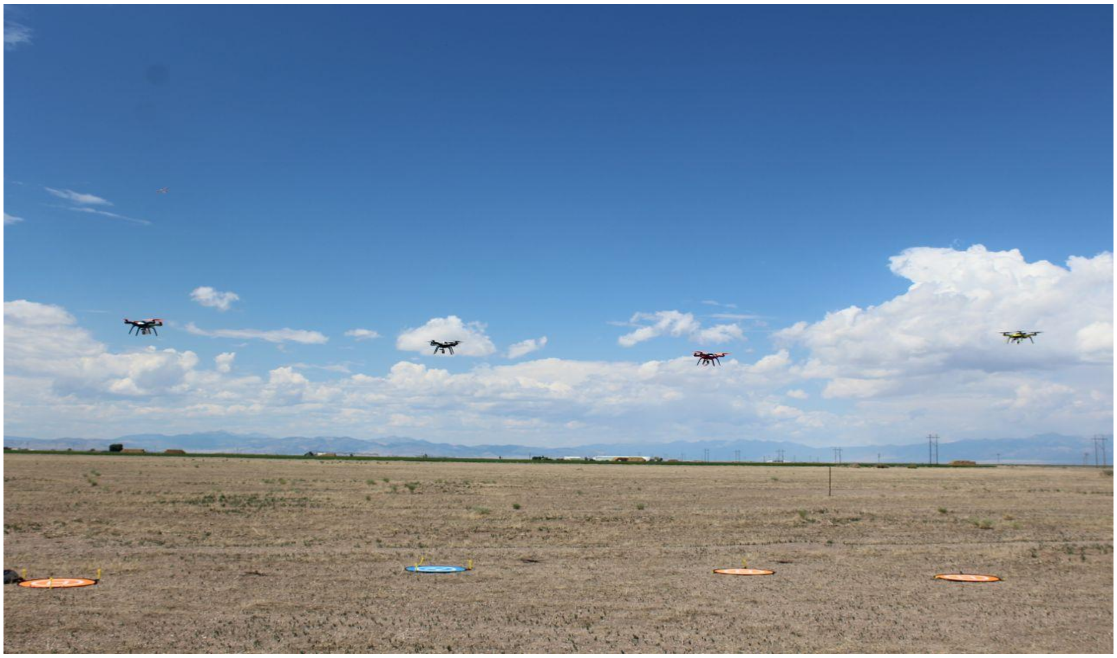


**Figure 53: Swarm takeoff**

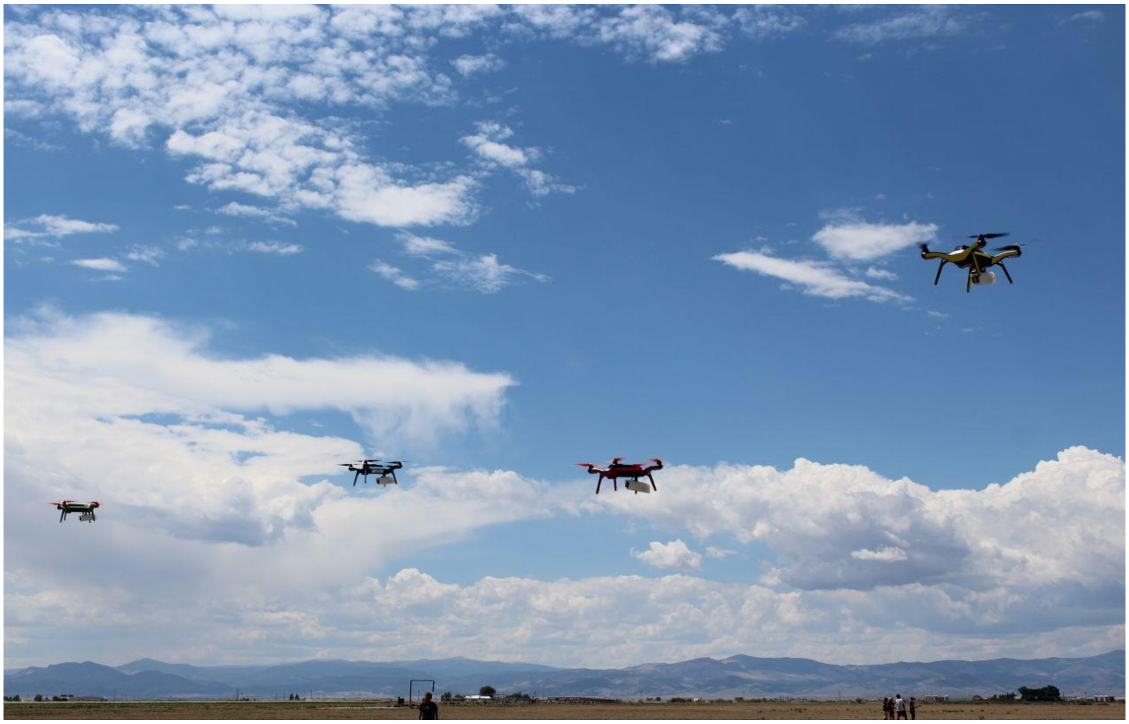


**Figure 54: Swarm facing desired direction**

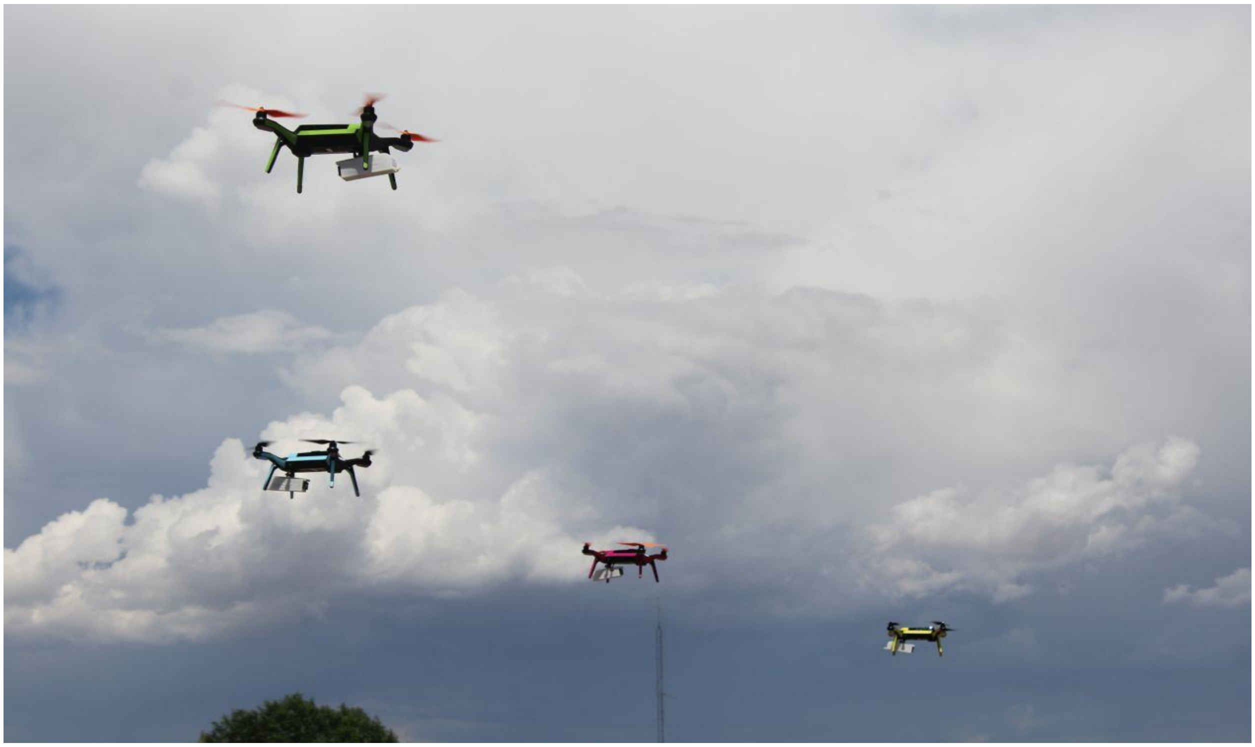


**Figure 55: Solo swarm moving into formation and toward first waypoint**

**Figure 56: Moving into high altitude tower formation**

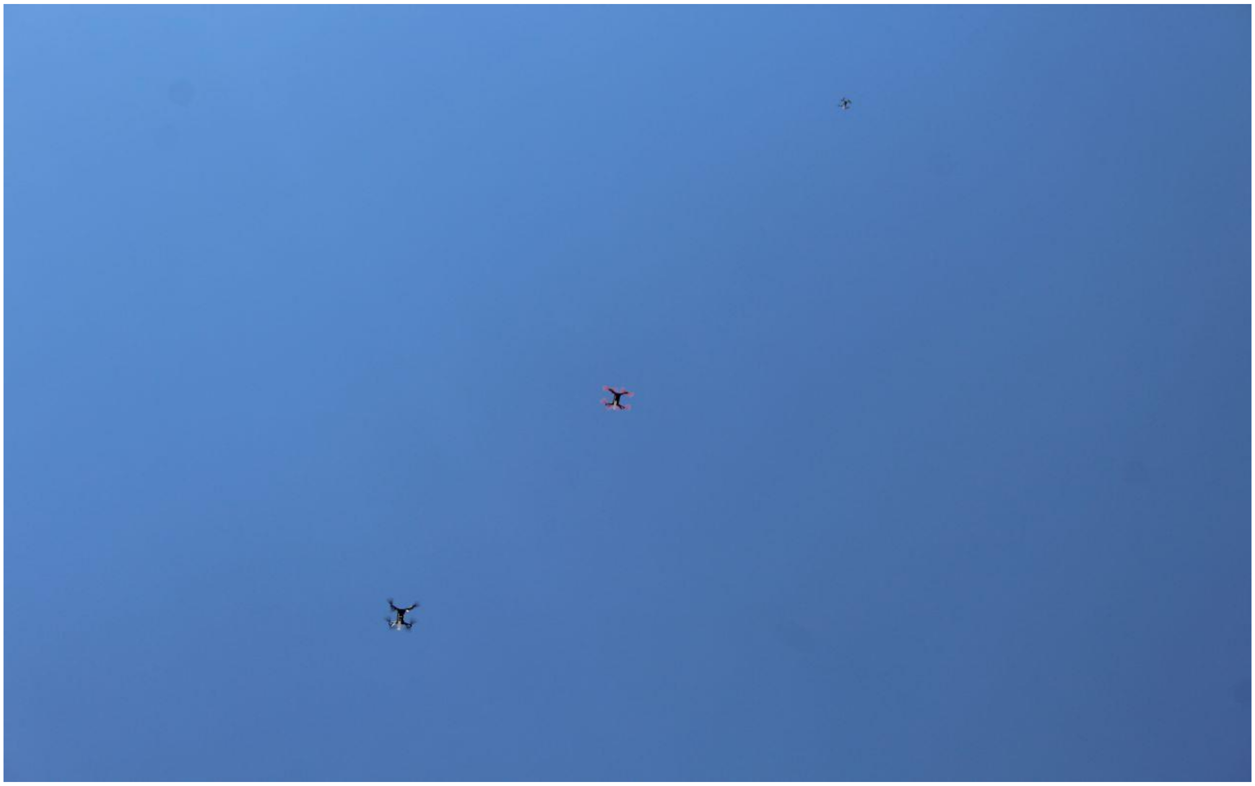


Figure 57: Tower formation

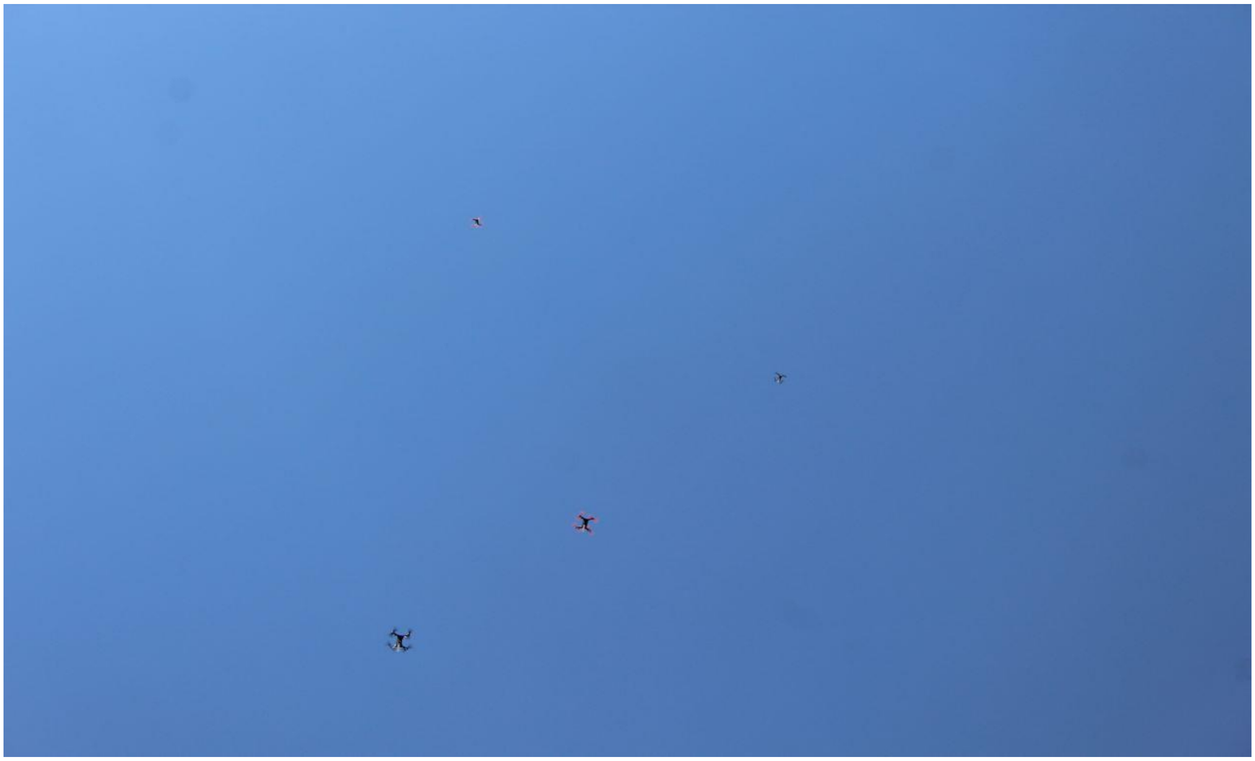


Figure 58: Exiting tower formation for landing sequence

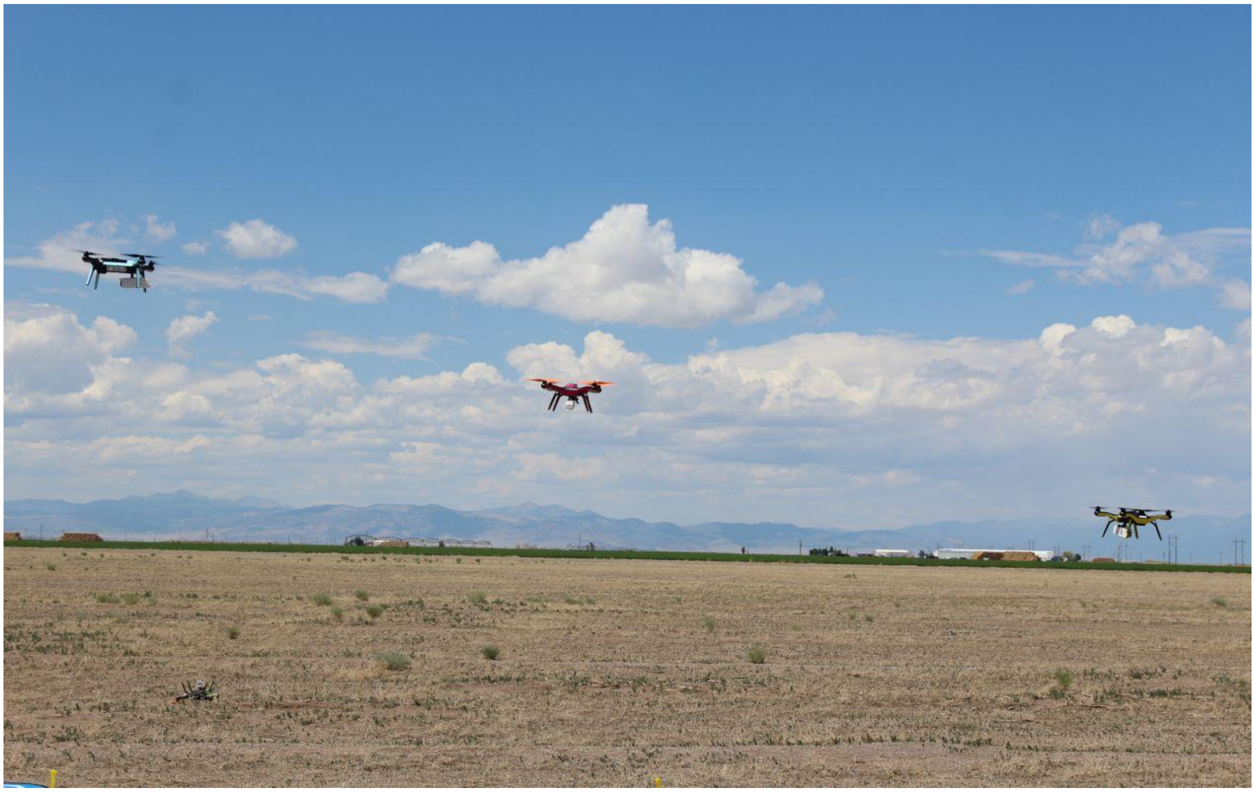


**Figure 59: Swarm landing**

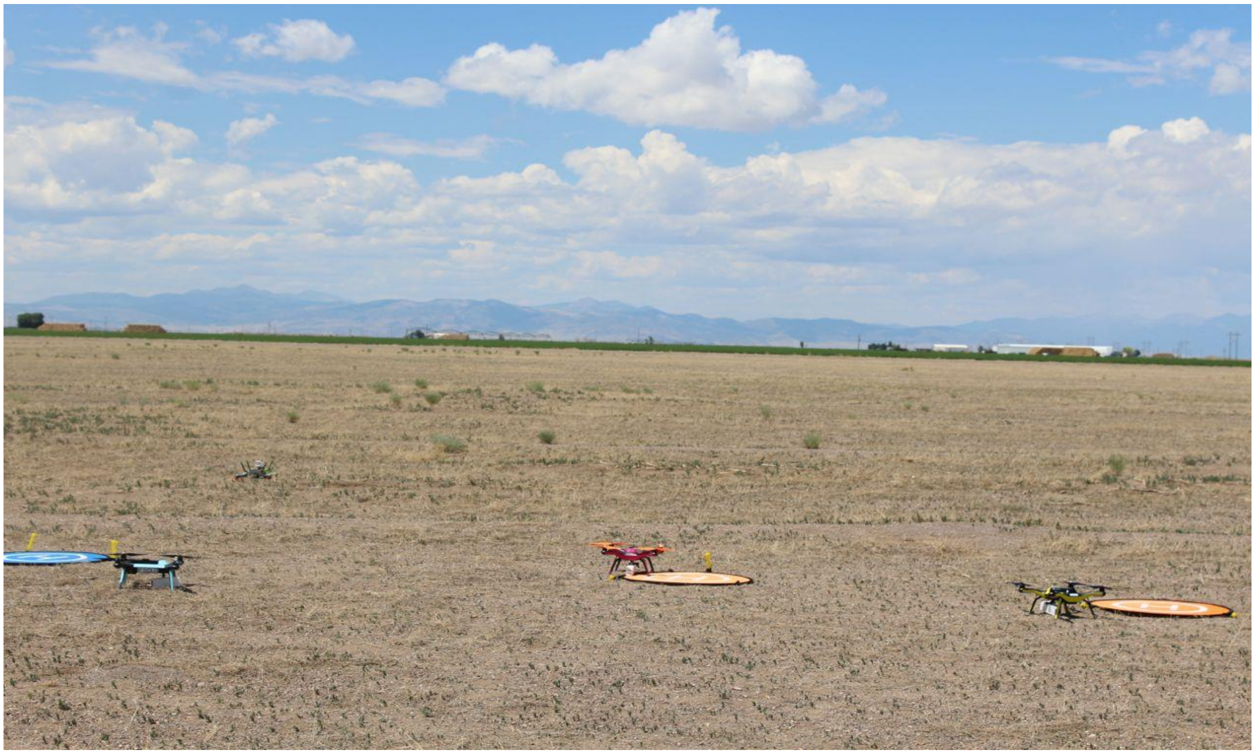


**Figure 60: Swarm operation landed and complete**

## 5.2 Atmospheric Boundary Layer Profile Results

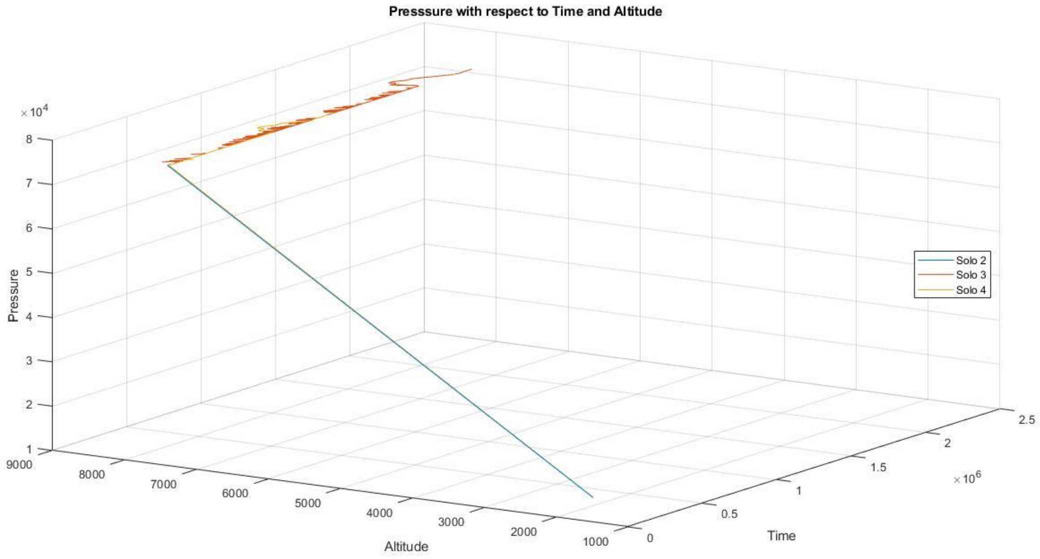


**Figure 61: Solo Swarm Pressure readings with respect to Time and Altitude**

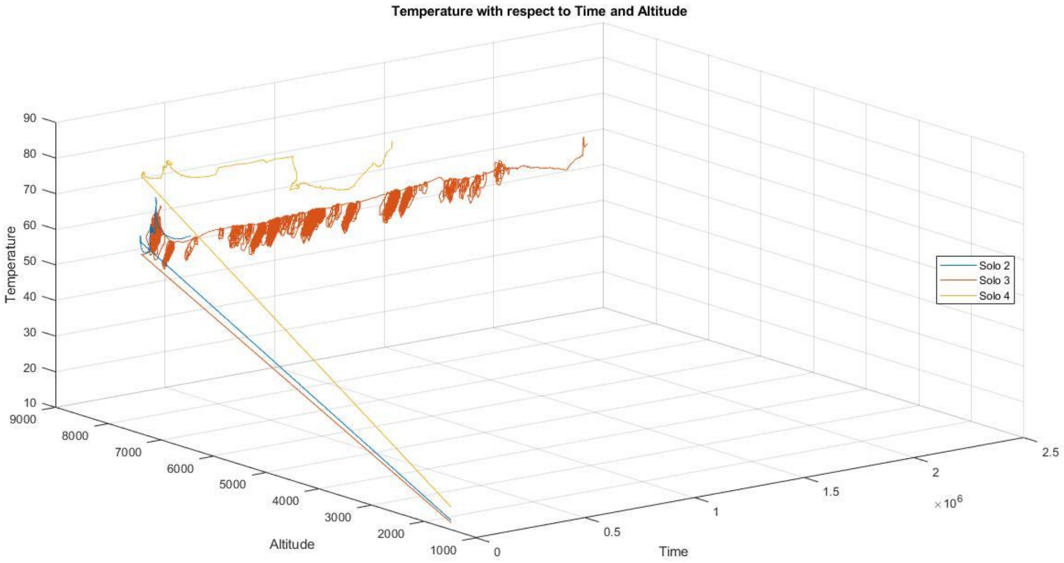


**Figure 62: Solo Swarm Temperature readings with respect to Time and Altitude**

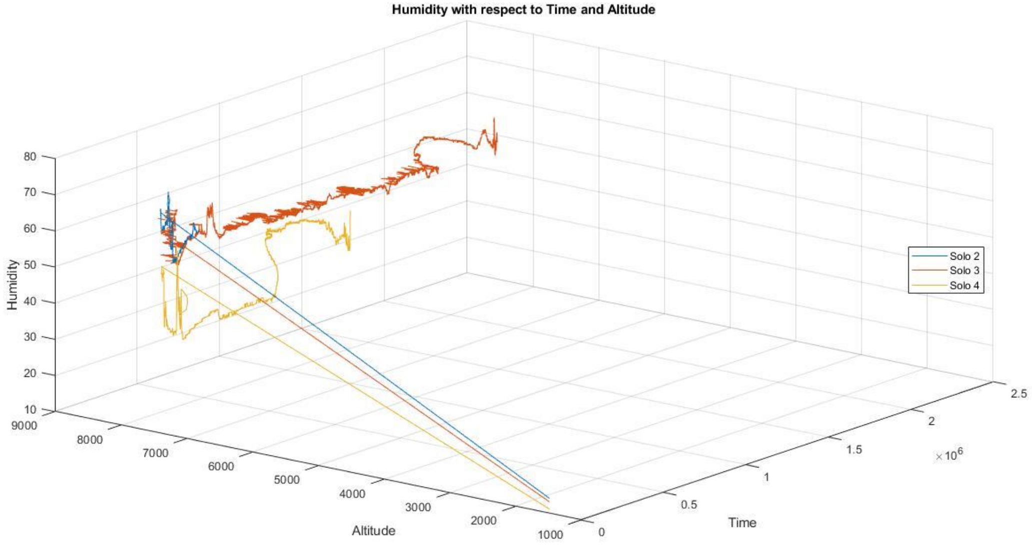


**Figure 63: Solo Swarm Humidity readings with respect to Time and Altitude**

The dropsonde data shown in Figures 61, 62 and 63 is supposed to show pressure, temperature, and humidity vs time and altitude respectively. Doing this represents the different atmospheric properties during the Solos flight profile. Due to the GPS in the dropsondes not keeping a lock, the data in these figures are not of importance. This data is only shown to illustrate what is possible in using a set of UAV in a virtual structure. Taking technologies that are already used in weather recording and mobilizing it while also allowing for relatively easy customization.

# CHAPTER VI

# 6. CONCLUSIONS

## 6.1 Summary

Overall the swarm operating as a virtual structure has performed as needed. There are improvements that can be made to better it for a more desired outcome. This would primarily be to make the controller data driven, which would allow for the swarm to better place its agents for better wind profile results. For this research however, the virtual structure allowed the Solos to take a desired formation and maintain this rigid formation during a flight. Coupling this with the onboard sensors, and the swarm was able to take atmospheric readings and save that for processing after flight.

## 6.2 Recommendations

Should have the atmospheric data sent to the controller to allow for it to reposition the Solos to a better position to record data. Adding the capability for the live data being taken in to be piped to the ground station as well. Including other types of UAS such as larger and smaller rotor UAS and fixed-wing, seen in Figure 64, would increase the versatility of the swarm overall.

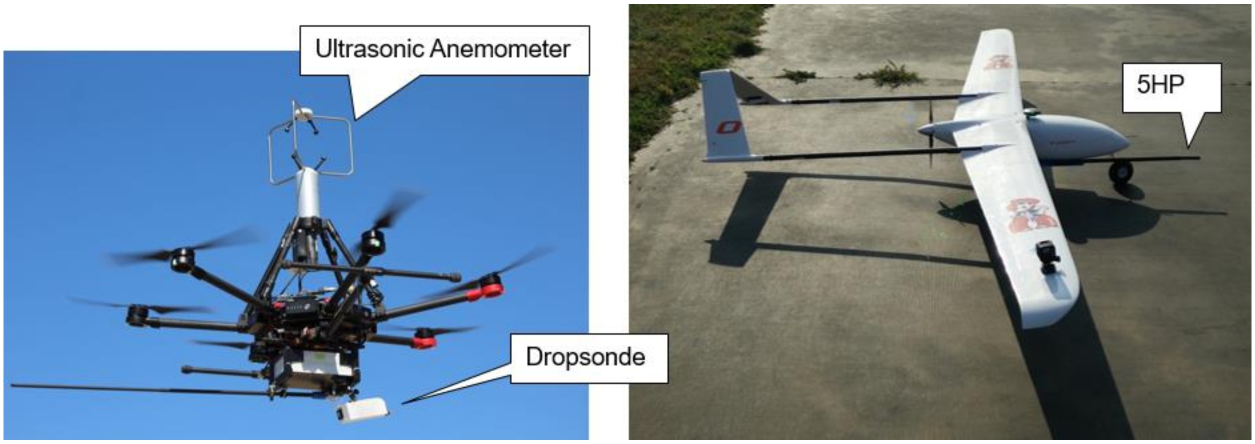


**Figure 64: Larger rotor UAS (left) and fixed wing UAS (right)**

Including the larger rotor UAS would allow for the use of larger sensor packages such as an ultrasonic anemometer. The fixed-wing UAS could introduce the use of a multi-hole probe while providing better endurance to the swarm. Incorporating these additional UAS would only broaden the capabilities of the swarm when taking atmospheric measurements.

## 6.3 Future Work

For this work, improving controller functionality and then seeing further testing in the field are both wanted and needed. Adding the capability for the live data being taken in to be piped to the ground station would be useful as well.

### 6.3.1 Controller design & optimization

Applying a DDDAS or Dynamic Data Driven Application Systems controller to the swarm would be ideal. This would entail the ability of an executing application to incorporate simulated data into the decision process, while conversely being able to dynamically manage sensors to refine measurements [6]. In addition, as new sensor data is taken into the systems simulation, a feed-back and control-loop is formed between the real-world application and simulation model, via the simulation modeling complex non-linear dynamics in quick time. This would allow the swarm to reposition itself in order to record a more desired data set, because the sensor controls constantly drive the measurement process for recalibrating the simulation. As a result of making the swarm controller data driven an accurate model is produced.

### 6.3.2 Flight testing

Further flight testing in multiple weather scenarios, and testing formations in XY coordinate formation as well as a tower formation with the included data driven controller would be the bulk of the future work flight testing. Future test flights with the larger rotor UAV carrying additional sensor packages and a fix-wing UAV with a multi-hole probe would expand swarm capabilities.

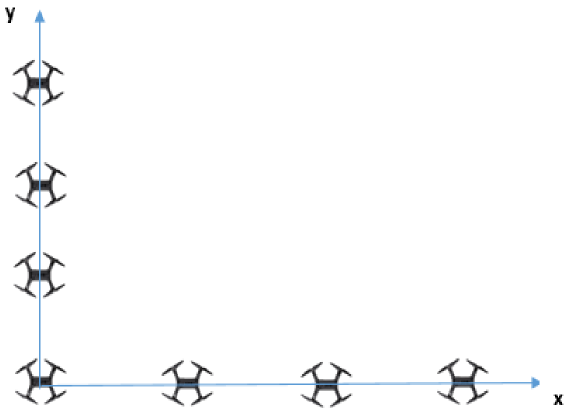


**Figure 65: Virtual Structure swarm in XY coordinate formation**

Figure 65 illustrated how the XY coordinate formation would be set up. The idea behind the formation is to expand on the tower formation by adding UAV or in this case data points in the x axis direction or horizontal direction. This will help in modeling the ABL in 3-dimensions.

APPENDIX

# Steps to run the Solo Swarm Formation with the ROS based Virtual Structure Controller

Components Required

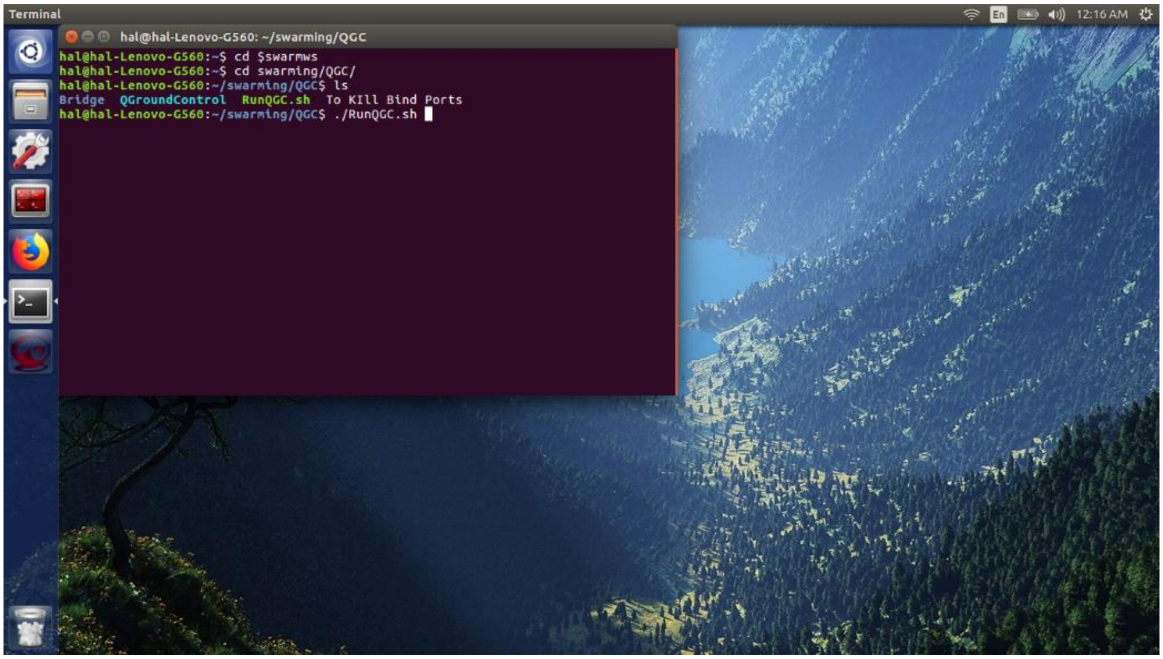
1. Controller Laptop with the ROS controller (one without the display). We will call this laptop the control laptop
2. Laptop for the Ground Station. We will call this the QGC laptop
3. Ardupilot with GPS
4. Router

Hardware setup

1. Power on the router, both the laptops and connect the Ardupilot with the GPS to a USB port on the control laptop.
2. Ensure that both the laptops are connected to SoloNetwork wifi.
3. Power on all the solos and make sure the solos are connected to their respective controllers and have a GPS lock
4. Set the Solos in the field.

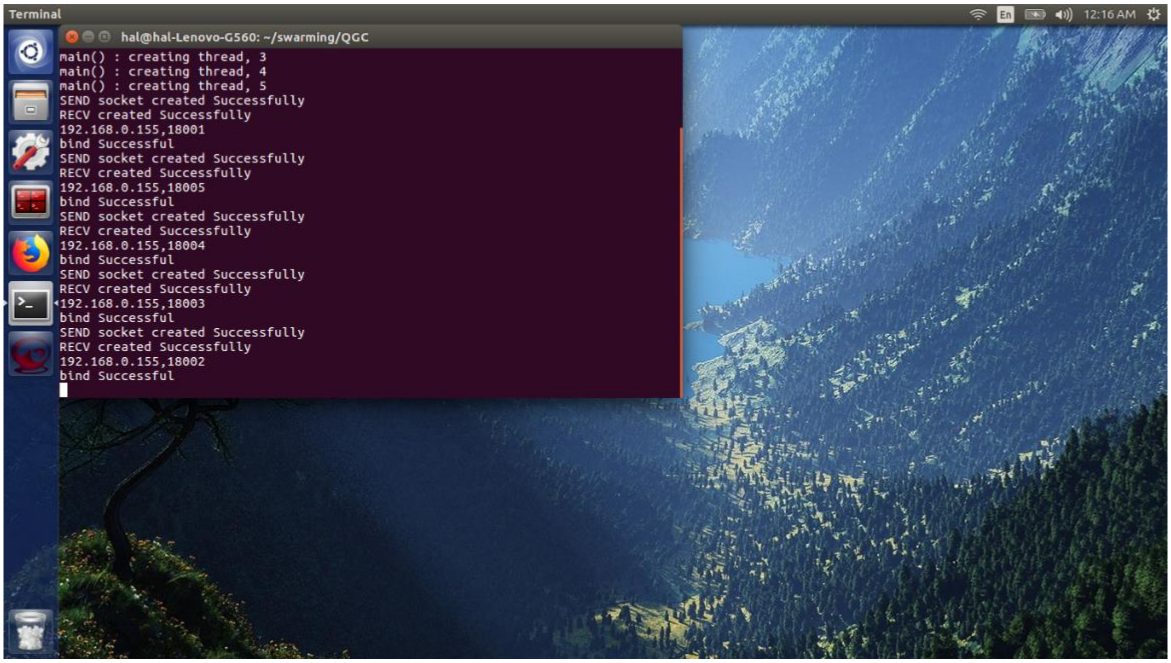
Command line steps

1. In the QGC laptop open a terminal, change directory to swarming/QGC and execute ‘RunQGC.sh’ as shown below

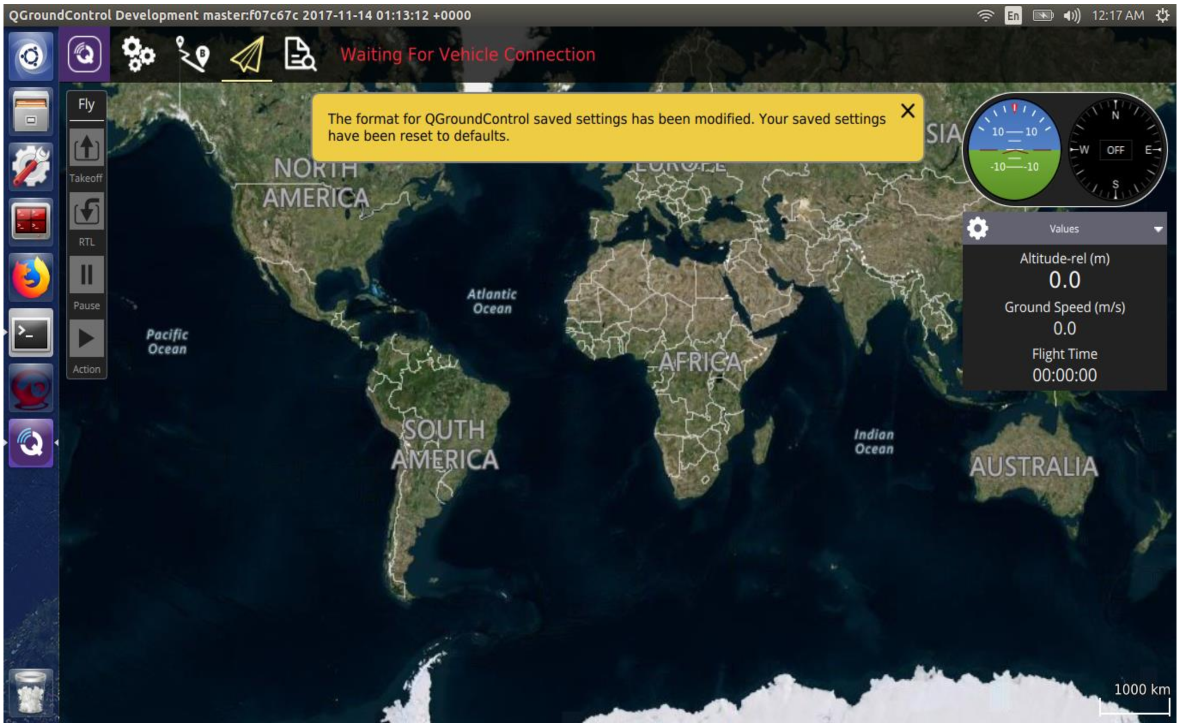


```
hal@hal-Lenovo-G560:~$ cd $swarmws
hal@hal-Lenovo-G560:~$ cd swarming/QGC/
hal@hal-Lenovo-G560:~/swarming/QGC$ ls
Bridge  QGroundControl  RunQGC.sh  To Kill Bind Ports
hal@hal-Lenovo-G560:~/swarming/QGC$ ./RunQGC.sh
```

2. The bash script executed in the previous step opens QGroundControl and starts a program that connects the ROS based controller on the control Laptop to the QGC running on the QGC laptop. You should see the following lines printed on the console and QGC open up as shown in the next image. If you see any line ‘binding failed’, then the connection didn’t start correctly and you need to kill the open ports by following the steps in a text file named ‘To Kill Bind Ports’



```
main() : creating thread, 3
main() : creating thread, 4
main() : creating thread, 5
SEND socket created Successfully
RECV created Successfully
192.168.0.155,18001
bind Successful
SEND socket created Successfully
RECV created Successfully
192.168.0.155,18005
bind Successful
SEND socket created Successfully
RECV created Successfully
192.168.0.155,18004
bind Successful
SEND socket created Successfully
RECV created Successfully
192.168.0.155,18003
bind Successful
SEND socket created Successfully
RECV created Successfully
192.168.0.155,18002
bind Successful
```



QGroundControl Development master:f07c67c 2017-11-14 01:13:12 +0000

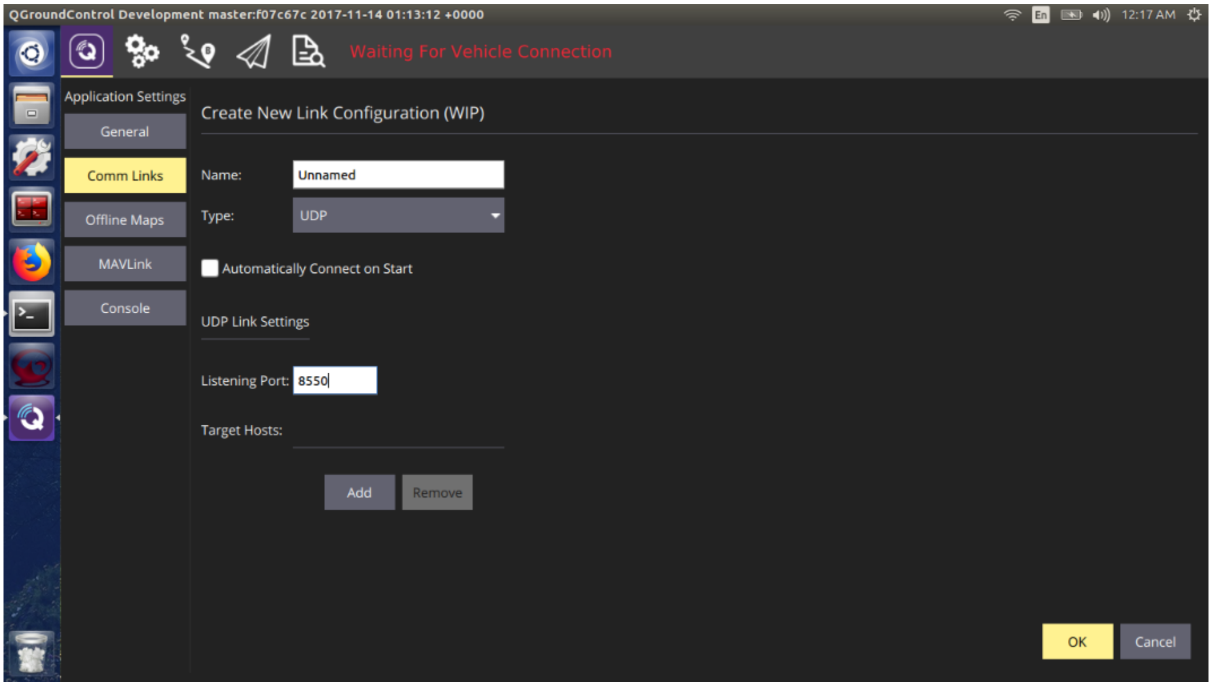
Waiting For Vehicle Connection

The format for QGroundControl saved settings has been modified. Your saved settings have been reset to defaults.

Altitude-rel (m) 0.0

Ground Speed (m/s) 0.0

Flight Time 00:00:00



QGroundControl Development master:f07c67c 2017-11-14 01:13:12 +0000

Waiting For Vehicle Connection

Create New Link Configuration (WIP)

Name: Unnamed

Type: UDP

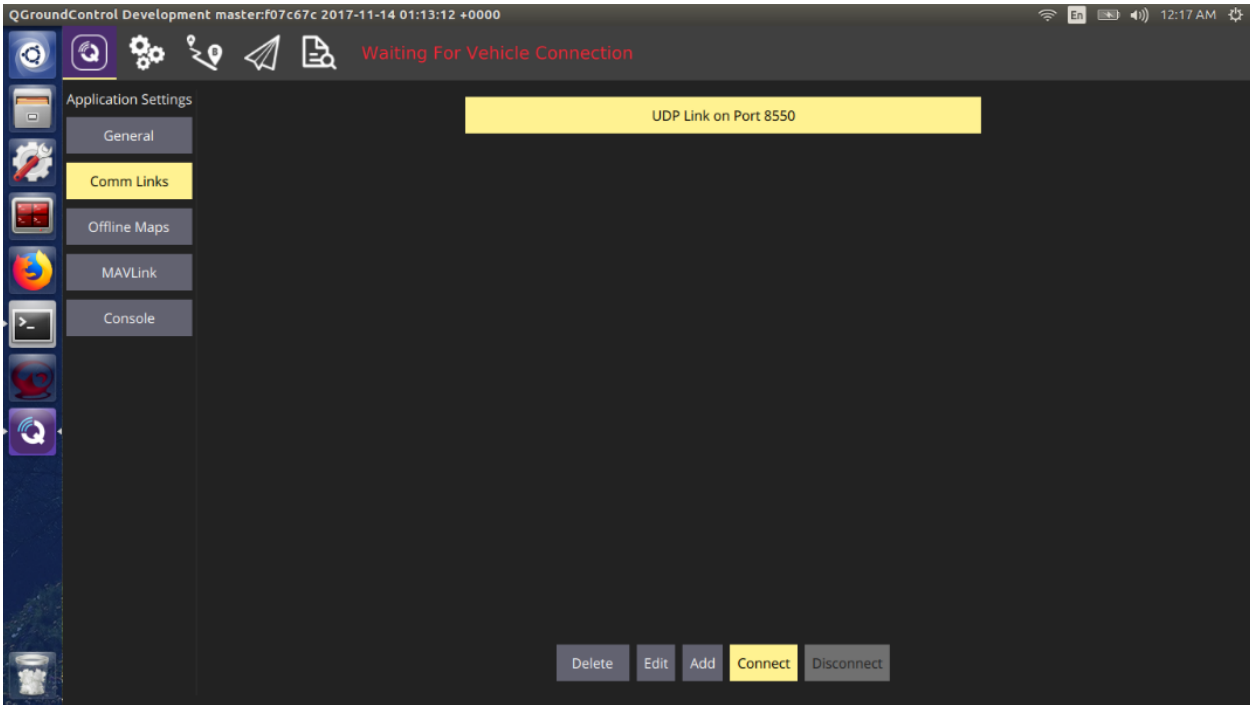
☐ Automatically Connect on Start

UDP Link Settings

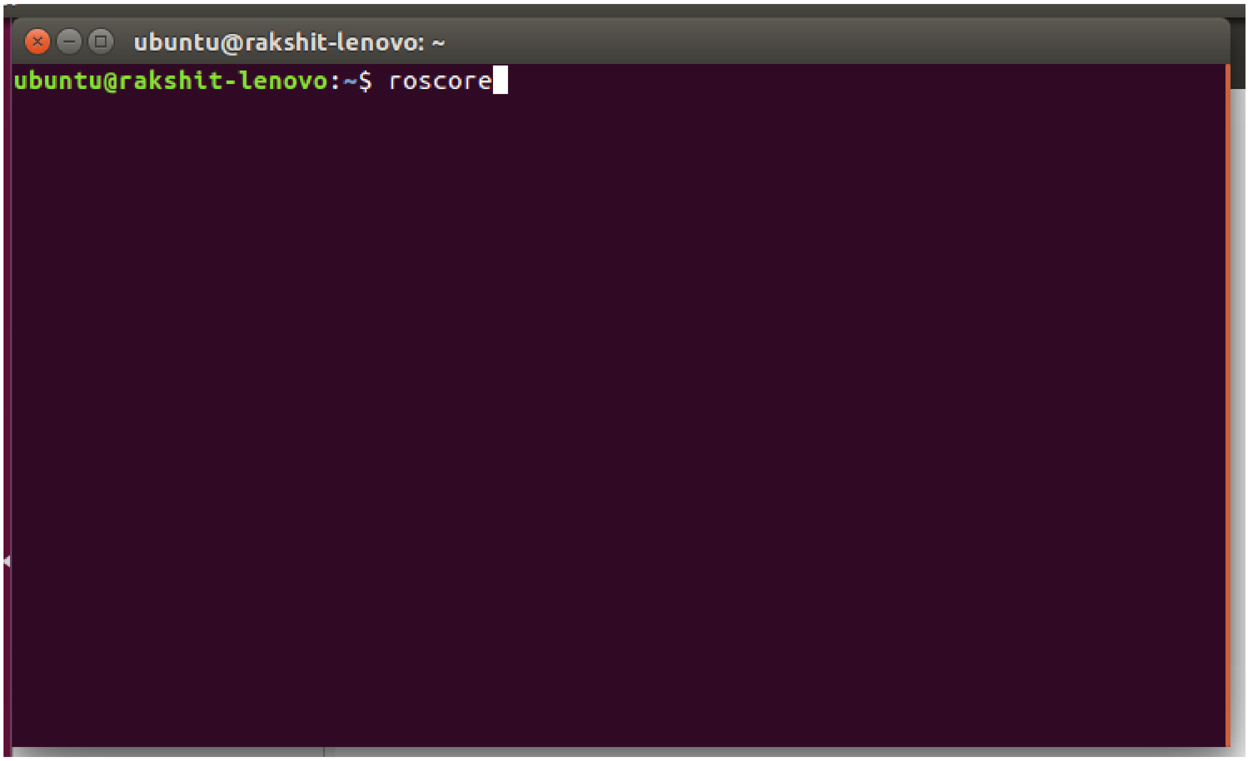
Listening Port: 8550

Target Hosts:

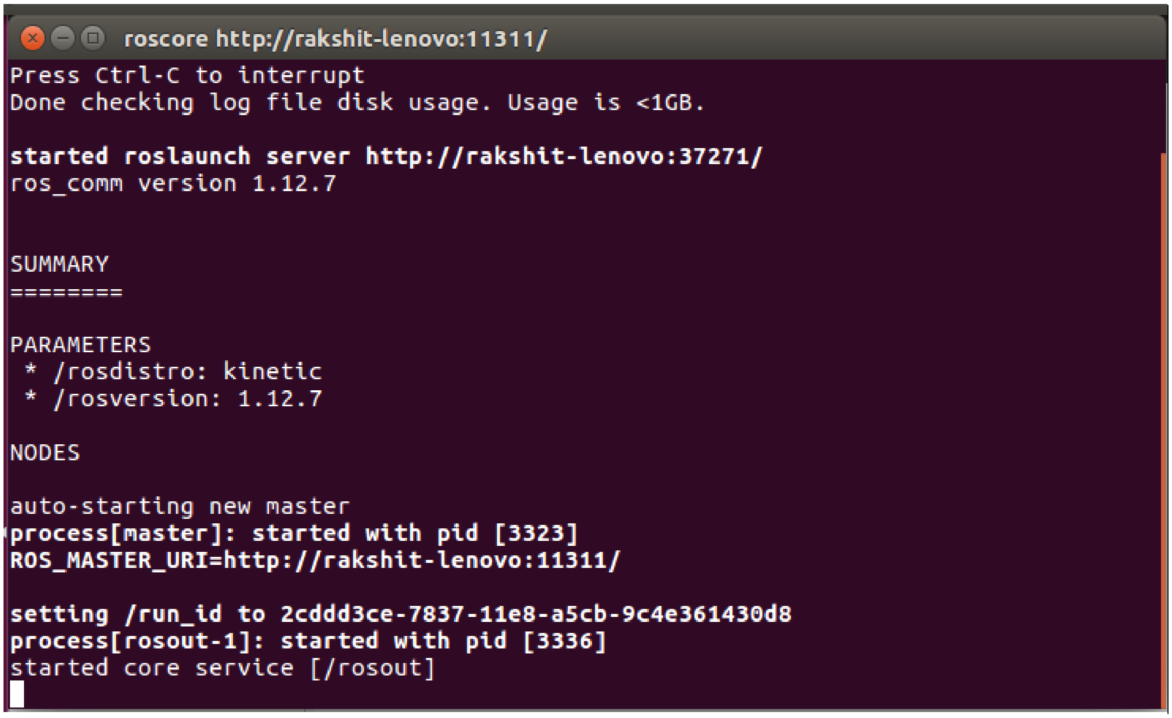
3. In QGroundControl open up Comm Links and create a new UDP connection with 8550 as the listening port.
4. Connect to it.



5. In the control laptop open up a terminal start ros master using the ‘roscore’ command as shown below.

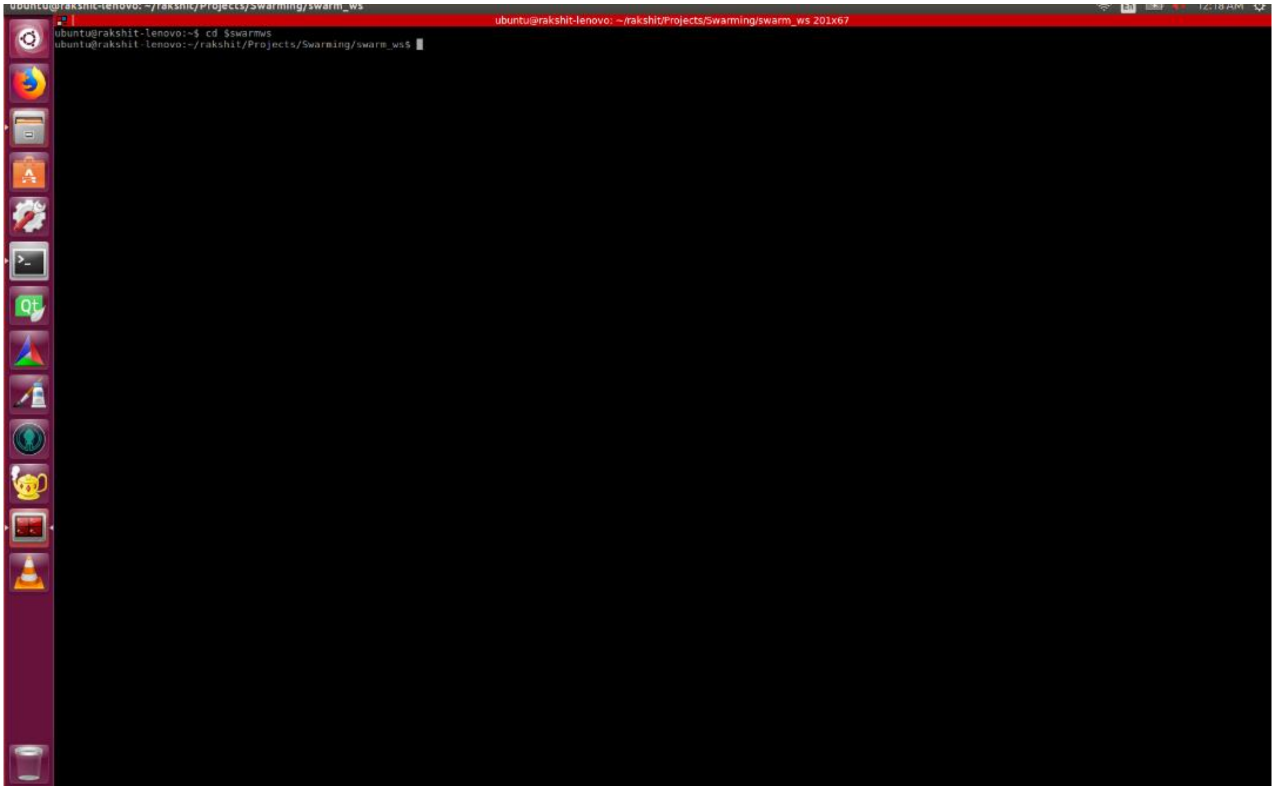


```
ubuntu@rakshit-lenovo:~$ roscore
```



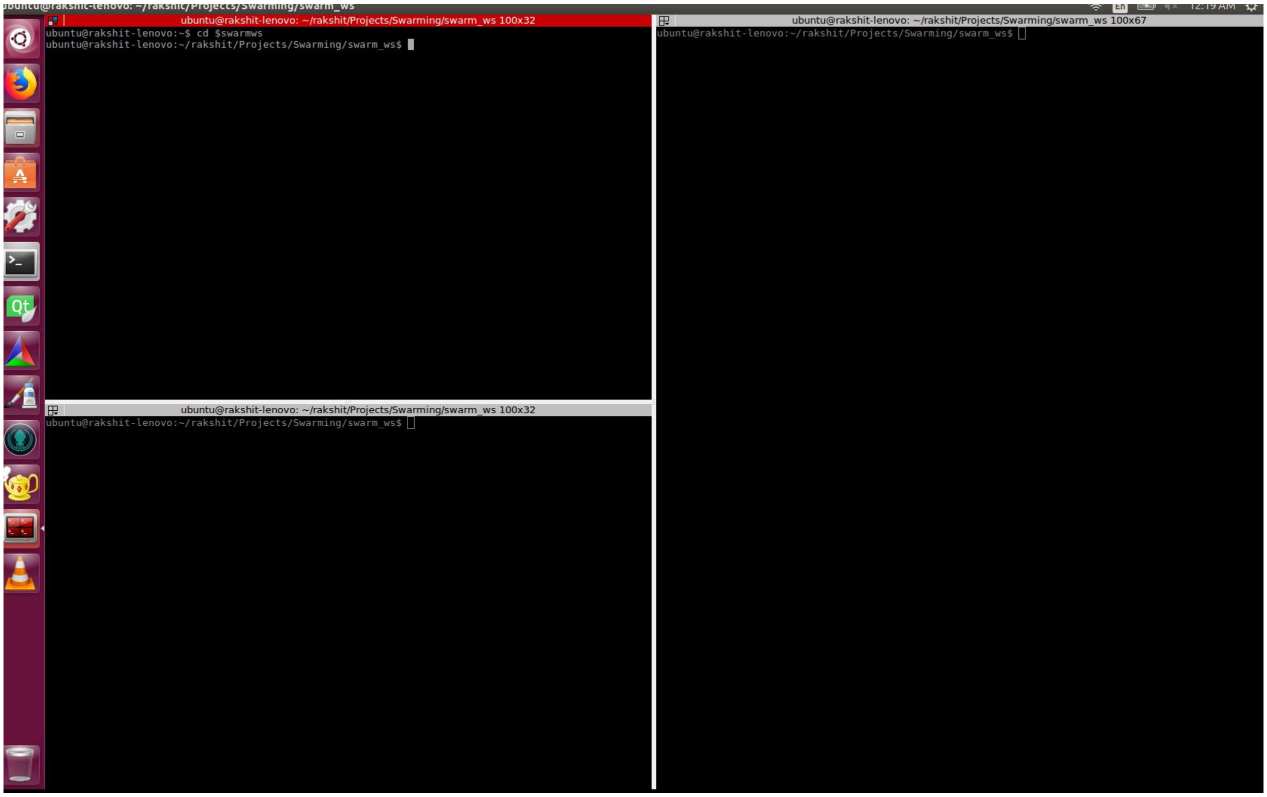
```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://rakshit-lenovo:37271/
ros_comm version 1.12.7


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES

auto-starting new master
process[master]: started with pid [3323]
ROS_MASTER_URI=http://rakshit-lenovo:11311/

setting /run_id to 2cddd3ce-7837-11e8-a5cb-9c4e361430d8
process[rosout-1]: started with pid [3336]
started core service [/rosout]
```

6. Open terminator and cd into $swarmws using the command as shown in the image below. Once the path is changed split the terminal into 3 different terminals



ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws
12:18 AM
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 201x67

```
ubuntu@rakshit-lenovo:~$ cd $swarmws
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ 
```



ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws
12:19 AM
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32

```
ubuntu@rakshit-lenovo:~$ cd $swarmws
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ 
```

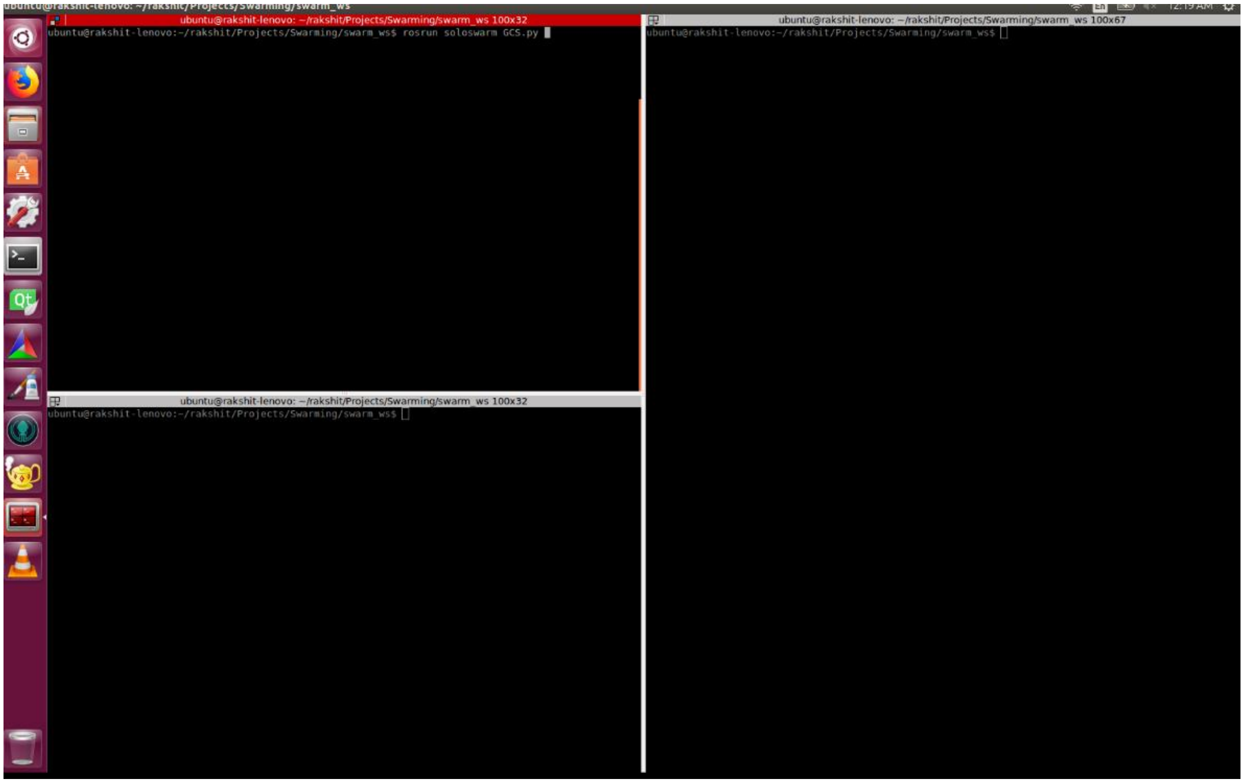
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ 
```

ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x67

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ 
```

7. In one terminal rosrun the GCS.py from the soloswarm package with the following command ‘rosrun soloswarm GCS.py’



ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws

ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ rosrun soloswarm GCS.py
```

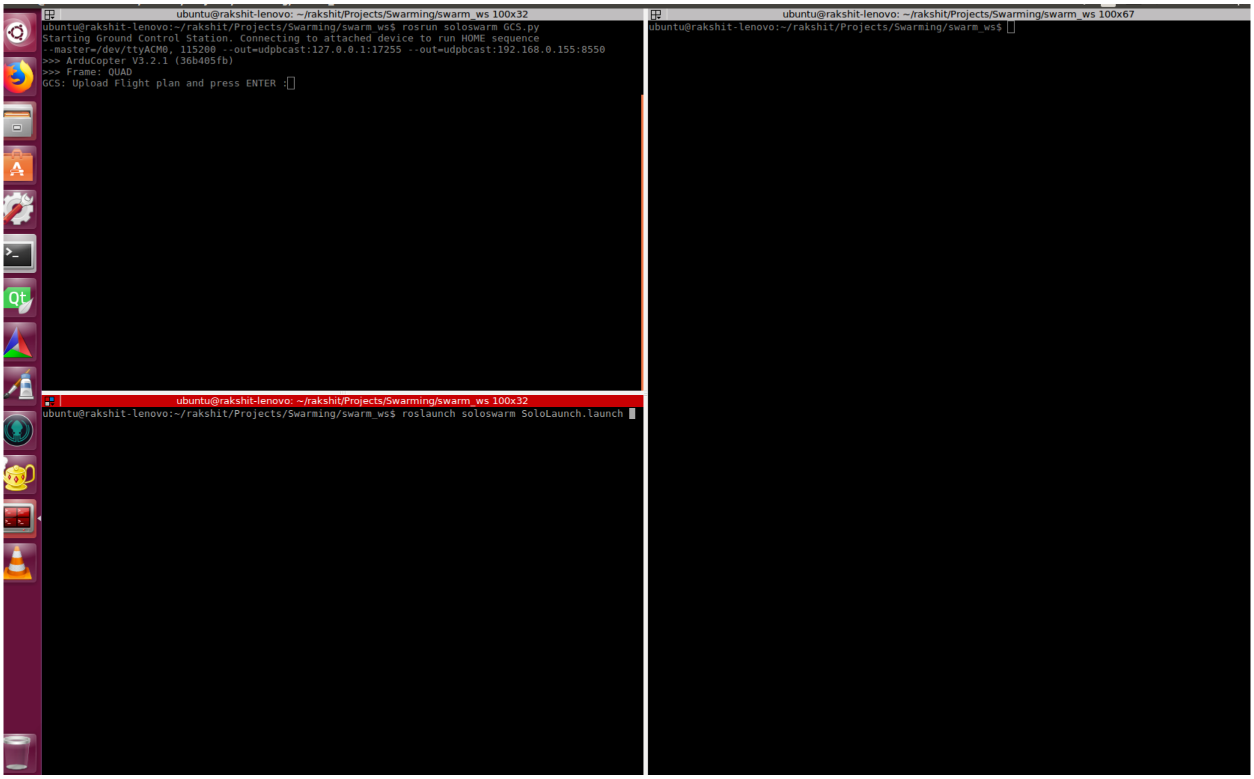
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x67

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$
```

ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$
```

8. The GCS.py will wait for the user to upload a flight plan.
9. Open a second terminal and roslaunch SoloLaunch with the following command ‘roslaunch soloswarm SoloLaunch.launch’

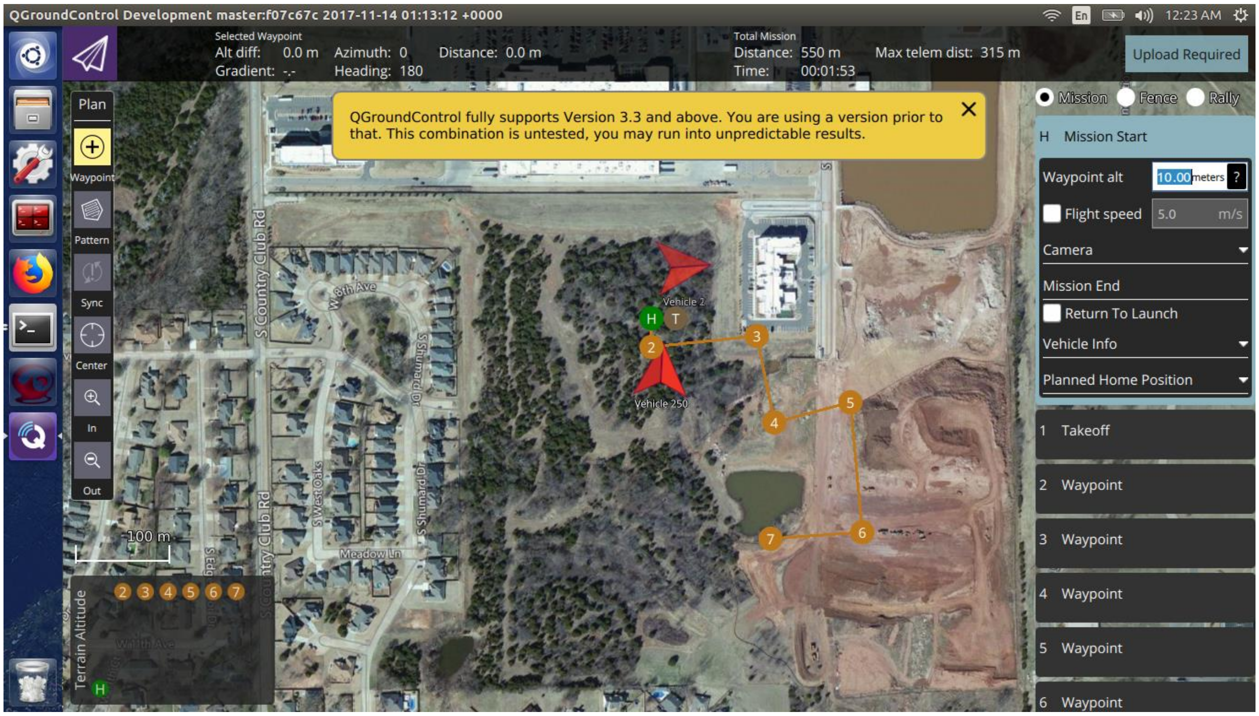


```
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ rosrun soloswarm GCS.py
Starting Ground Control Station. Connecting to attached device to run HOME sequence
--master=/dev/ttyACM0, 115200 --out=udpbcast:127.0.0.1:17255 --out=udpbcast:192.168.0.155:8550
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
GCS: Upload Flight plan and press ENTER :
```

```
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ roslaunch soloswarm SoloLaunch.launch
```

```
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x67
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$
```

10. When steps 7 to 9 are executed, QGC should start updating with the current positions of the Groundstation and the Solos



QGroundControl Development master:f07c67c 2017-11-14 01:13:12 +0000

Selected Waypoint
Alt diff: 0.0 m  Azimuth: 0  Distance: 0.0 m
Gradient: -.-  Heading: 180

Total Mission
Distance: 550 m  Max telem dist: 315 m
Time: 00:01:53

QGroundControl fully supports Version 3.3 and above. You are using a version prior to that. This combination is untested, you may run into unpredictable results.

● Mission ○ Fence ○ Rally

H Mission Start

Waypoint alt: 10.00 meters

□ Flight speed: 5.0 m/s

Camera

Mission End

□ Return To Launch

Vehicle Info

Planned Home Position

1 Takeoff

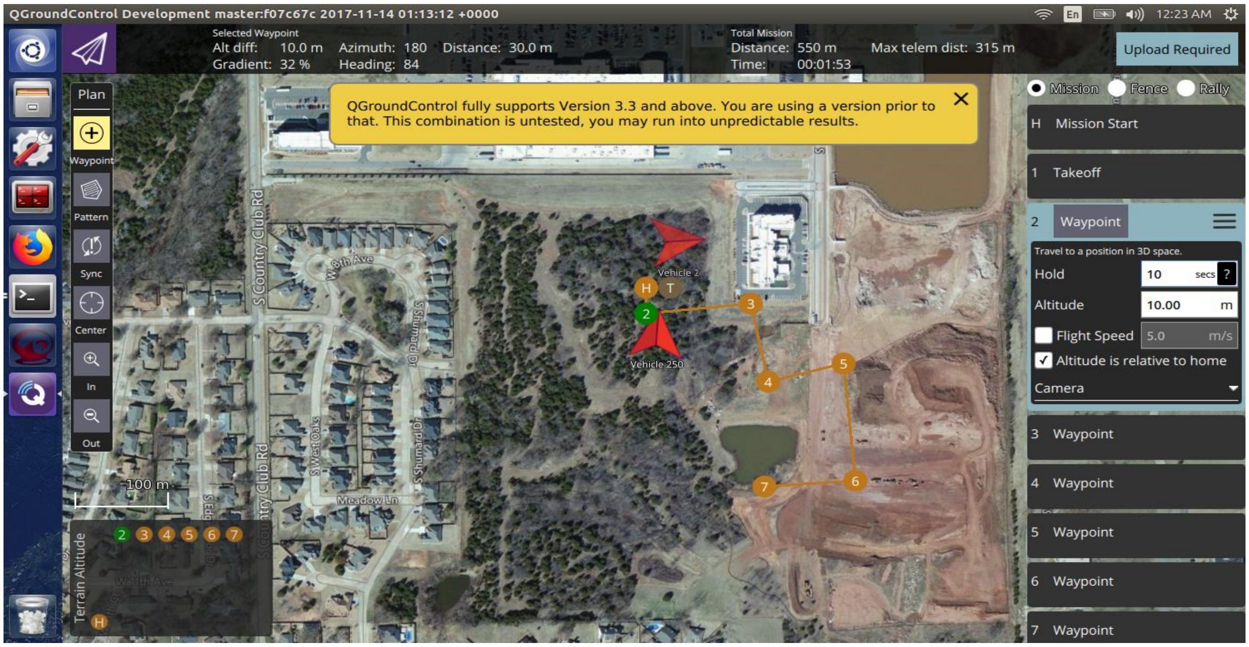
2 Waypoint

3 Waypoint

4 Waypoint

5 Waypoint

6 Waypoint


QGroundControl Development master:f07c67c 2017-11-14 01:13:12 +0000

Selected Waypoint
Alt diff: 10.0 m Azimuth: 180 Distance: 30.0 m
Gradient: 32 % Heading: 84

Total Mission
Distance: 550 m Max telem dist: 315 m
Time: 00:01:53

QGroundControl fully supports Version 3.3 and above. You are using a version prior to that. This combination is untested, you may run into unpredictable results.

○ Mission ○ Fence ○ Rally

H Mission Start

1 Takeoff

2 Waypoint

Travel to a position in 3D space.

Hold 10 secs

Altitude 10.00 m

□ Flight Speed 5.0 m/s

☑ Altitude is relative to home

Camera

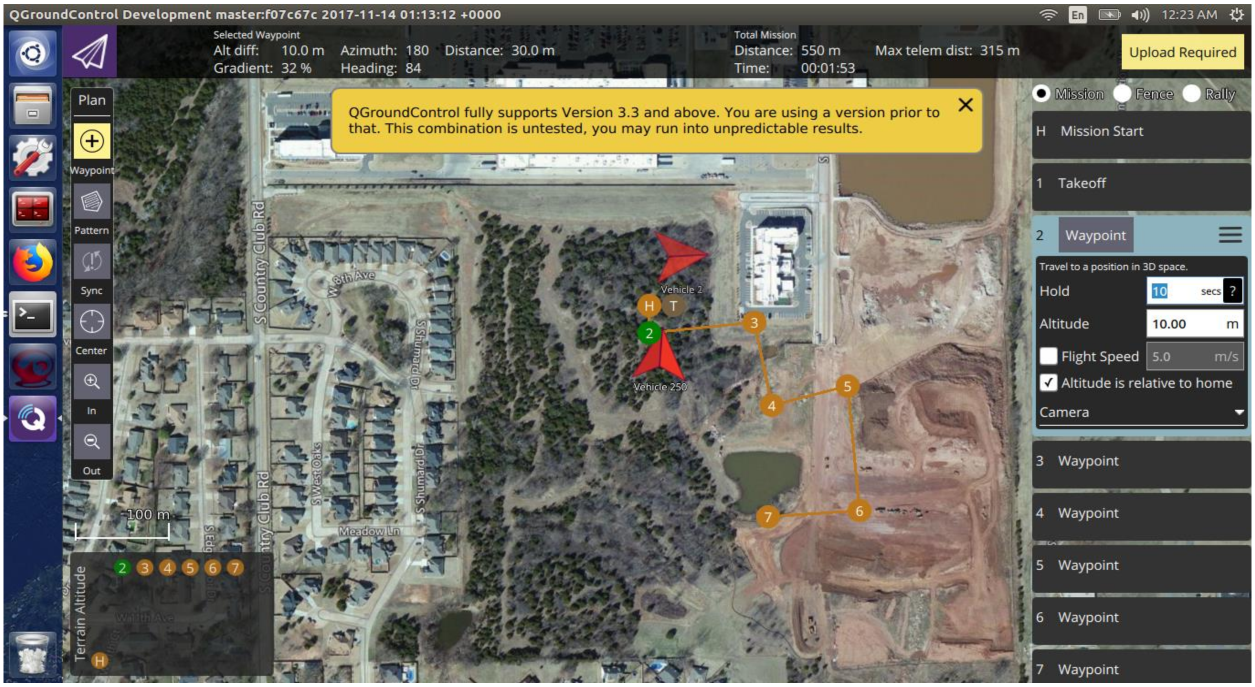
3 Waypoint

4 Waypoint

5 Waypoint

6 Waypoint

7 Waypoint


QGroundControl Development master:f07c67c 2017-11-14 01:13:12 +0000

Selected Waypoint
Alt diff: 10.0 m Azimuth: 180 Distance: 30.0 m
Gradient: 32 % Heading: 84

Total Mission
Distance: 550 m Max telem dist: 315 m
Time: 00:01:53

QGroundControl fully supports Version 3.3 and above. You are using a version prior to that. This combination is untested, you may run into unpredictable results.

○ Mission ○ Fence ○ Rally

H Mission Start

1 Takeoff

2 Waypoint

Travel to a position in 3D space.

Hold 10 secs

Altitude 10.00 m

□ Flight Speed 5.0 m/s

☑ Altitude is relative to home

Camera

3 Waypoint

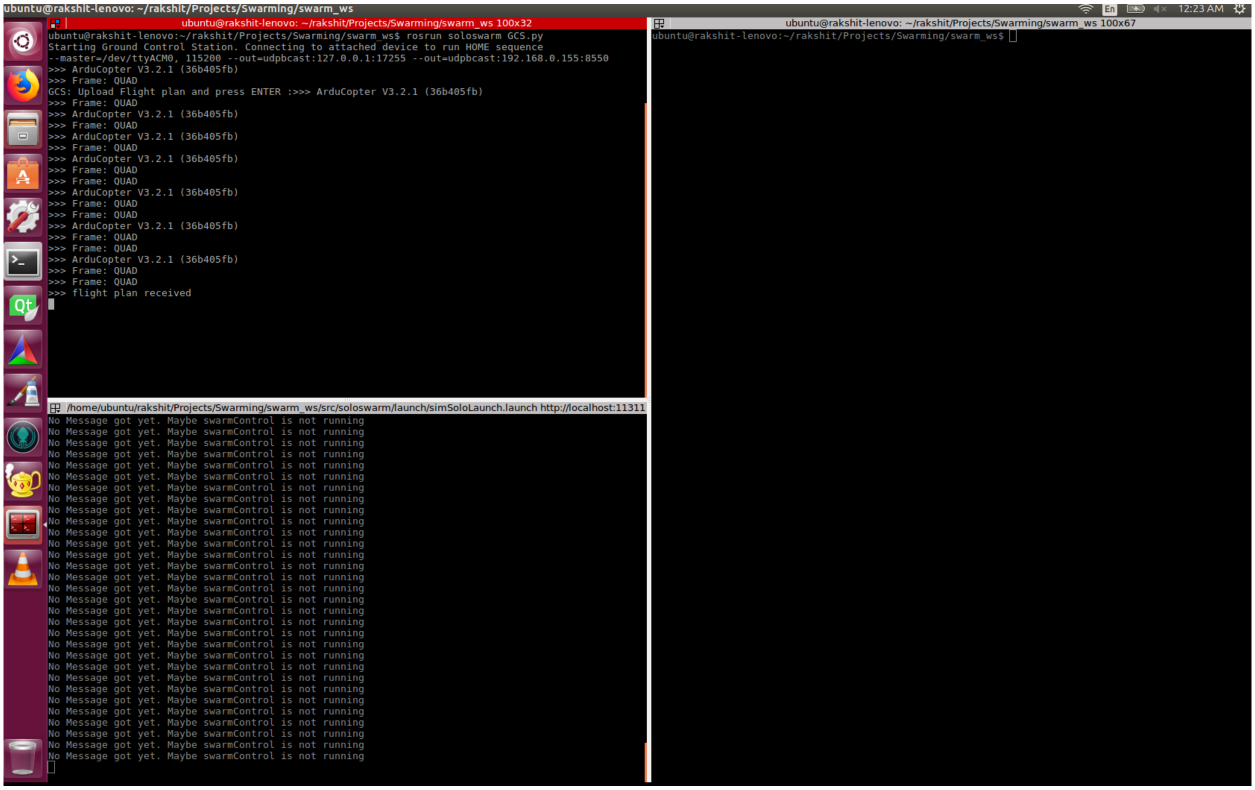
4 Waypoint

5 Waypoint

6 Waypoint

7 Waypoint

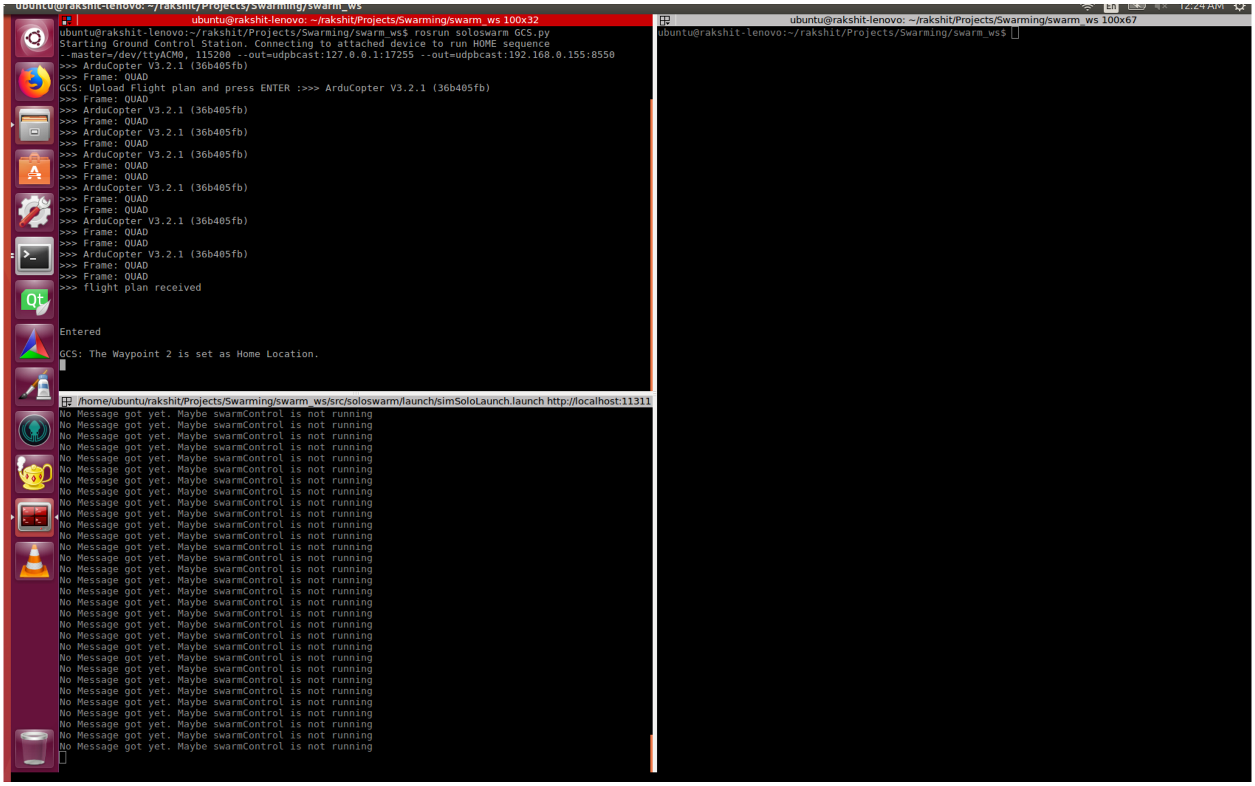
11. Create a flight plan by adding waypoints. Make sure put in a correct altitude value and hold in seconds for each waypoint. The hold time is the time in secs the swarm loiters around the waypoint location.
12. Once the flight plan is ready click upload
13. The terminal on which GCS.py is launch should now print a line saying ‘flight plan received’. Once this is printed, press Enter a couple of time on the GCS.py terminal


ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws
12:23 AM
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32
/home/ubuntu/rakshit/Projects/Swarming/swarm_ws/src/soloswarm/launch/simSoloLaunch.launch http://localhost:11311
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x67

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ rosrun soloswarm GCS.py
Starting Ground Control Station. Connecting to attached device to run HOME sequence
--master=/dev/ttyACM0, 115200 --out=udpbcast:127.0.0.1:17255 --out=udpbcast:192.168.0.155:8550
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
GCS: Upload Flight plan and press ENTER :>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> Frame: QUAD
>>> flight plan received
```

```
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
```

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$
```

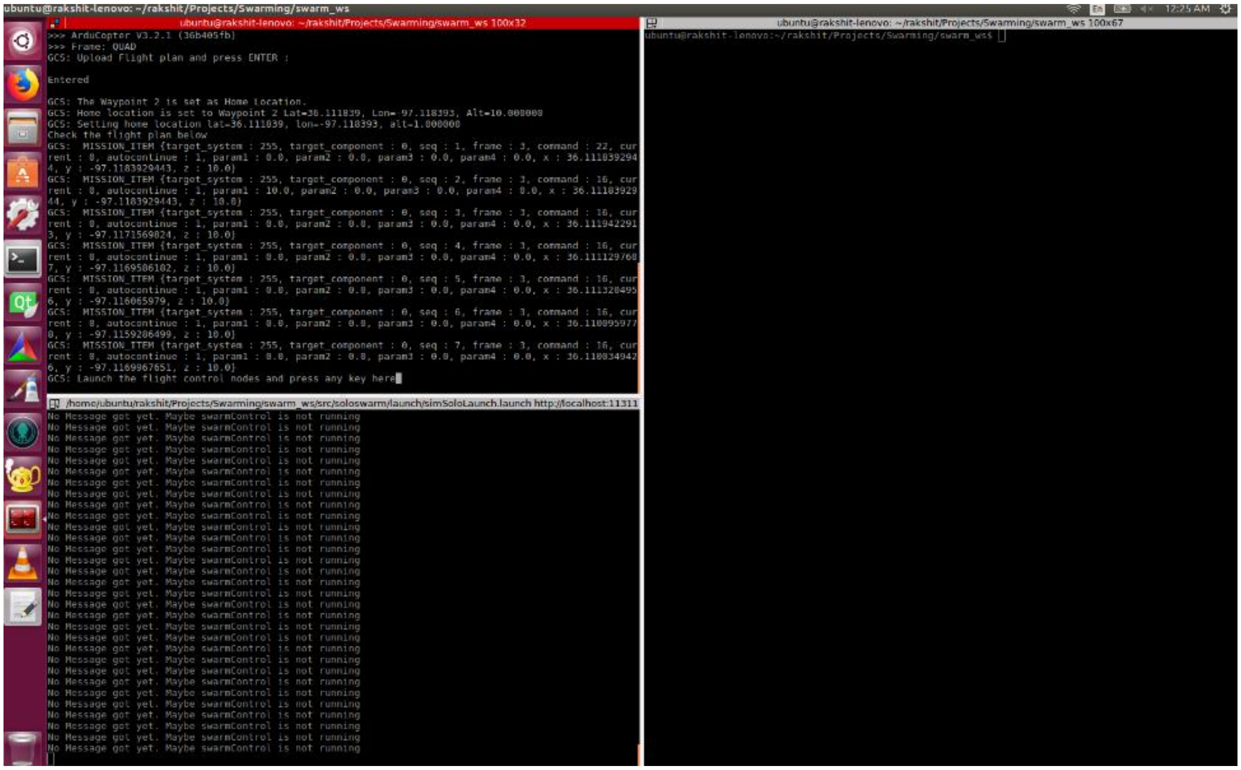

ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws
12:24 AM
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32
/home/ubuntu/rakshit/Projects/Swarming/swarm_ws/src/soloswarm/launch/simSoloLaunch.launch http://localhost:11311
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x67

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ rosrun soloswarm GCS.py
Starting Ground Control Station. Connecting to attached device to run HOME sequence
--master=/dev/ttyACM0, 115200 --out=udpbcast:127.0.0.1:17255 --out=udpbcast:192.168.0.155:8550
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
GCS: Upload Flight plan and press ENTER :>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> Frame: QUAD
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
>>> Frame: QUAD
>>> flight plan received



Entered

GCS: The Waypoint 2 is set as Home Location.
```

```
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
```

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$
```

14. Wait until the mission plan is printed on the terminal as shown below. The terminal now waits for the swarm controller to launch.

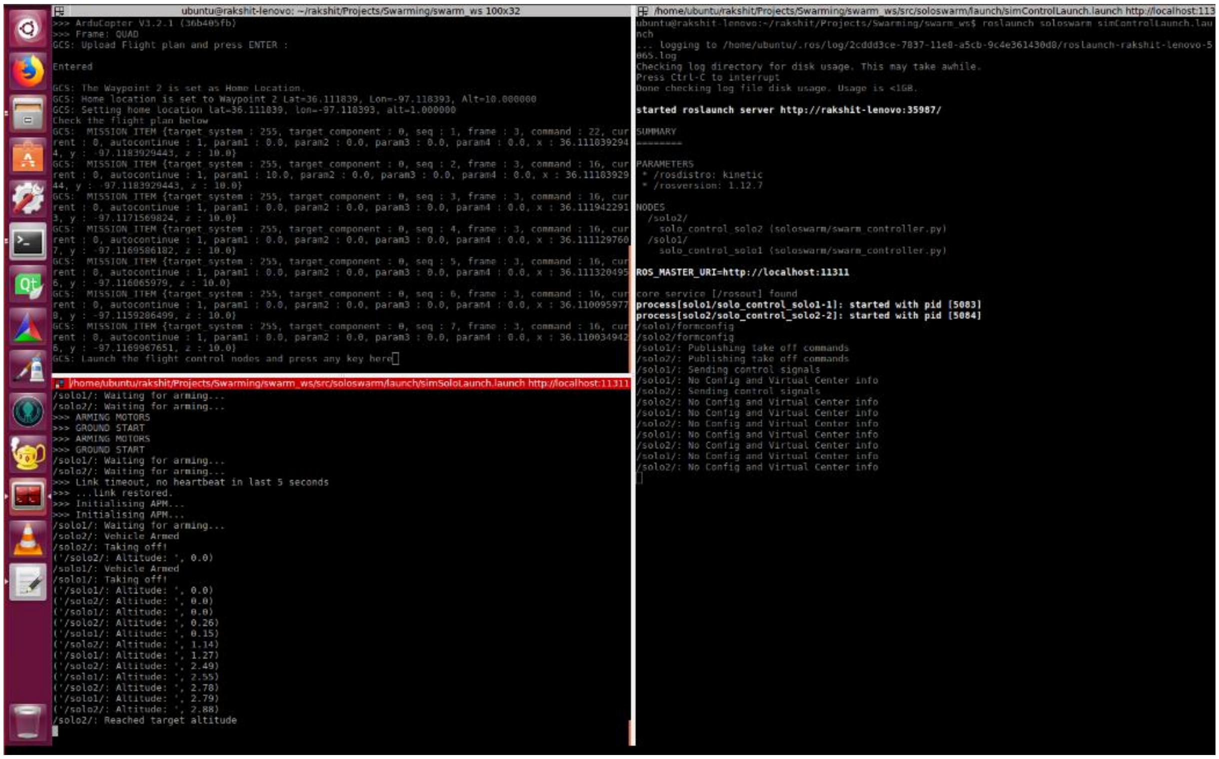

ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws
12:25 AM
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32
/home/ubuntu/rakshit/Projects/Swarming/swarm_ws/src/soloswarm/launch/simSoloLaunch.launch http://localhost:11311
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x67

```
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
GCS: Upload Flight plan and press ENTER :

Entered

GCS: The Waypoint 2 is set as Home Location.
GCS: Home location is set to Waypoint 2 Lat=36.111839, Lon=-97.118393, Alt=10.000000
GCS: Setting home location lat=36.111839, lon=-97.118393, alt=1.000000
Check the flight plan below
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 1, frame : 3, command : 22, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111839294
4, y : -97.1183929443, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 2, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 10.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.11183929
44, y : -97.1183929443, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 3, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111942291
3, y : -97.1171569824, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 4, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111129760
7, y : -97.1169586182, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 5, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111328495
6, y : -97.116065979, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 6, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.110895977
8, y : -97.1159286499, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 7, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.110834942
6, y : -97.1169967651, z : 10.0}
GCS: Launch the flight control nodes and press any key here
```

```
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
No Message got yet. Maybe swarmControl is not running
```

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$
```

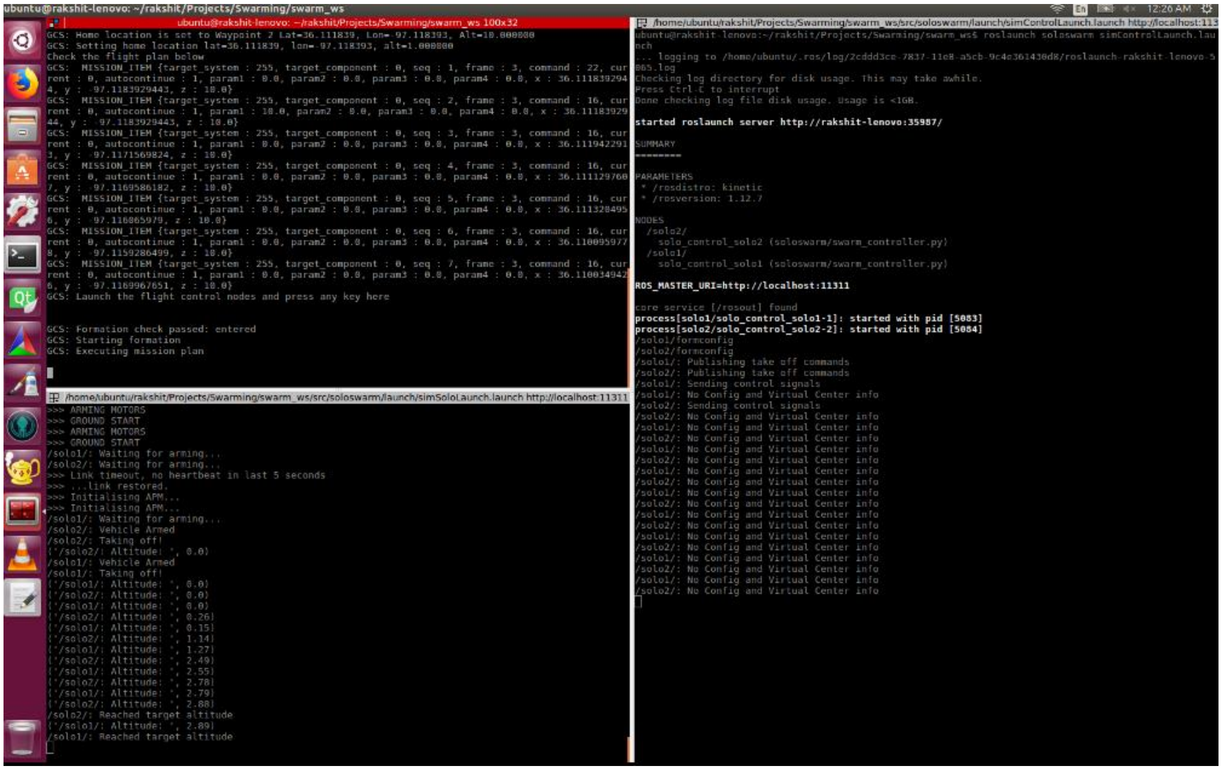
15. In the third terminal roslaunch the swarm controller with the following command ‘roslaunch soloswarm controlLaunch.launch’
16. With the above command the solos should arm and takeoff to an altitude of 3 meters from the ground. When all solos reach the desired altitude, the second terminal on which SoloInterface is running should print statements from all solos saying ‘Reached target altitude’.
17. Once step 16 is completed hit Enter a couple of times on the GCS.py terminal. It should print ’starting formation’ and the solo should start moving in formation.


ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32
/home/ubuntu/rakshit/Projects/Swarming/swarm_ws/src/soloswarm/launch/simSoloLaunch.launch http://localhost:11311
/home/ubuntu/rakshit/Projects/Swarming/swarm_ws/src/soloswarm/launch/simControlLaunch.launch http://localhost:113

```
>>> ArduCopter V3.2.1 (36b405fb)
>>> Frame: QUAD
GCS: Upload Flight plan and press ENTER :

Entered

GCS: The Waypoint 2 is set as Home Location.
GCS: Home location is set to Waypoint 2 Lat=36.111839, Lon=-97.118393, Alt=10.000000
GCS: Setting home location lat=36.111839, lon=-97.118393, alt=1.000000
Check the flight plan below
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 1, frame : 3, command : 22, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111839294
4, y : -97.1183929443, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 2, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 10.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.11183929
44, y : -97.1183929443, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 3, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111942291
3, y : -97.1171569824, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 4, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111129760
7, y : -97.1169586182, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 5, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111320495
6, y : -97.116065979, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 6, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.110095977
8, y : -97.1159286499, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 7, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.110034942
6, y : -97.1169967651, z : 10.0}
GCS: Launch the flight control nodes and press any key here
```

```
/solo1/: Waiting for arming...
/solo2/: Waiting for arming...
>>> ARMING MOTORS
>>> GROUND START
>>> ARMING MOTORS
>>> GROUND START
/solo1/: Waiting for arming...
/solo2/: Waiting for arming...
>>> Link timeout, no heartbeat in last 5 seconds
>>> ...link restored.
>>> Initialising APM...
>>> Initialising APM...
/solo1/: Waiting for arming...
/solo2/: Vehicle Armed
/solo2/: Taking off!
('/solo2/: Altitude: ', 0.0)
/solo1/: Vehicle Armed
/solo1/: Taking off!
('/solo1/: Altitude: ', 0.0)
('/solo2/: Altitude: ', 0.0)
('/solo1/: Altitude: ', 0.0)
('/solo2/: Altitude: ', 0.26)
('/solo1/: Altitude: ', 0.15)
('/solo2/: Altitude: ', 1.14)
('/solo1/: Altitude: ', 1.27)
('/solo2/: Altitude: ', 2.49)
('/solo1/: Altitude: ', 2.55)
('/solo2/: Altitude: ', 2.78)
('/solo1/: Altitude: ', 2.79)
('/solo2/: Altitude: ', 2.88)
/solo2/: Reached target altitude
```

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ roslaunch soloswarm simControlLaunch.lau
nch
... logging to /home/ubuntu/.ros/log/2cddd3ce-7837-11e8-a5cb-9c4e361430d8/roslaunch-rakshit-lenovo-5
065.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://rakshit-lenovo:35987/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES
  /solo2/
    solo_control_solo2 (soloswarm/swarm_controller.py)
  /solo1/
    solo_control_solo1 (soloswarm/swarm_controller.py)

ROS_MASTER_URI=http://localhost:11311

core service [/rosout] found
process[solo1/solo_control_solo1-1]: started with pid [5083]
process[solo2/solo_control_solo2-2]: started with pid [5084]
/solo1/formconfig
/solo2/formconfig
/solo1/: Publishing take off commands
/solo2/: Publishing take off commands
/solo1/: Sending control signals
/solo1/: No Config and Virtual Center info
/solo2/: Sending control signals
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
```


ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws
ubuntu@rakshit-lenovo: ~/rakshit/Projects/Swarming/swarm_ws 100x32
/home/ubuntu/rakshit/Projects/Swarming/swarm_ws/src/soloswarm/launch/simSoloLaunch.launch http://localhost:11311
/home/ubuntu/rakshit/Projects/Swarming/swarm_ws/src/soloswarm/launch/simControlLaunch.launch http://localhost:113
En
12:26 AM

```
GCS: Home location is set to Waypoint 2 Lat=36.111839, Lon=-97.118393, Alt=10.000000
GCS: Setting home location lat=36.111839, lon=-97.118393, alt=1.000000
Check the flight plan below
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 1, frame : 3, command : 22, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111839294
4, y : -97.1183929443, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 2, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 10.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.11183929
44, y : -97.1183929443, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 3, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111942291
3, y : -97.1171569824, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 4, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111129760
7, y : -97.1169586182, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 5, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.111320495
6, y : -97.116065979, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 6, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.110095977
8, y : -97.1159286499, z : 10.0}
GCS:  MISSION_ITEM {target_system : 255, target_component : 0, seq : 7, frame : 3, command : 16, cur
rent : 0, autocontinue : 1, param1 : 0.0, param2 : 0.0, param3 : 0.0, param4 : 0.0, x : 36.110034942
6, y : -97.1169967651, z : 10.0}
GCS: Launch the flight control nodes and press any key here


GCS: Formation check passed: entered
GCS: Starting formation
GCS: Executing mission plan
```

```
>>> ARMING MOTORS
>>> GROUND START
>>> ARMING MOTORS
>>> GROUND START
/solo1/: Waiting for arming...
/solo2/: Waiting for arming...
>>> Link timeout, no heartbeat in last 5 seconds
>>> ...link restored.
>>> Initialising APM...
>>> Initialising APM...
/solo1/: Waiting for arming...
/solo2/: Vehicle Armed
/solo2/: Taking off!
('/solo2/: Altitude: ', 0.0)
/solo1/: Vehicle Armed
/solo1/: Taking off!
('/solo1/: Altitude: ', 0.0)
('/solo2/: Altitude: ', 0.0)
('/solo1/: Altitude: ', 0.0)
('/solo2/: Altitude: ', 0.26)
('/solo1/: Altitude: ', 0.15)
('/solo2/: Altitude: ', 1.14)
('/solo1/: Altitude: ', 1.27)
('/solo2/: Altitude: ', 2.49)
('/solo1/: Altitude: ', 2.55)
('/solo2/: Altitude: ', 2.78)
('/solo1/: Altitude: ', 2.79)
('/solo2/: Altitude: ', 2.88)
/solo2/: Reached target altitude
('/solo1/: Altitude: ', 2.89)
/solo1/: Reached target altitude
```

```
ubuntu@rakshit-lenovo:~/rakshit/Projects/Swarming/swarm_ws$ roslaunch soloswarm simControlLaunch.lau
nch
... logging to /home/ubuntu/.ros/log/2cddd3ce-7837-11e8-a5cb-9c4e361430d8/roslaunch-rakshit-lenovo-5
065.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://rakshit-lenovo:35987/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES
  /solo2/
    solo_control_solo2 (soloswarm/swarm_controller.py)
  /solo1/
    solo_control_solo1 (soloswarm/swarm_controller.py)

ROS_MASTER_URI=http://localhost:11311

core service [/rosout] found
process[solo1/solo_control_solo1-1]: started with pid [5083]
process[solo2/solo_control_solo2-2]: started with pid [5084]
/solo1/formconfig
/solo2/formconfig
/solo1/: Publishing take off commands
/solo2/: Publishing take off commands
/solo1/: Sending control signals
/solo1/: No Config and Virtual Center info
/solo2/: Sending control signals
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
/solo1/: No Config and Virtual Center info
/solo2/: No Config and Virtual Center info
```

18. When the final waypoint is reached the solos should reconfigure into a safe formation and the landing sequence should execute.

# REFERENCES

[1] "Autonomous Three-Dimensional Formation Flight for a Swarm of Unmanned Aerial Vehicles" Bennet, McInnes, et al., AIAA Journal of Guidance Control and Dynamics, 2011

[2] "Behavior-Based Control Hierarchy of Unmanned Aerial Vehicle Swarming" Pamphile, Lin, IEEE, 2006

[3] "Autonomous Formation Flight Using Bifurcating Potential Fields" Suzuki, Uchiyama, ICAS, 2010

[4] "Towards a Fully Autonomous Swarm of Unmanned Aerial Vehicles" Leonard, Savvaris, et al., UKACC International Conference on Control, 2012

[5] "Towards Autonomous Micro UAV Swarms" Burkle, Segor, et al., J Intell Robot Syst, 2011

[6] "Investigations of DDDAS for Command and Control of UAV Swarms with Agent-Based Modeling" McCune, Purta, et al., Proceedings of the 2013 Winter Simulation Conference, 2013

[7] "The Robust Control of the Servomechanism Problem for Linear Time-Invariant Multivariable Systems" Davison, IEEE, 1976

[8] "http://sprayers101.com/surface-inversions/" Jason, 2018

[9] Wen, Guanghui & Li, Zhongkui & Duan, Zhisheng & Chen, Guanrong. (2012). Distributed Consensus Control for Linear Multi-agent Systems with Discontinuous Observations. International Journal of Control. 86. 95–106. 10.1080/00207179.2012.719637.

[10] Stull, R.B. An Introduction to Boundary Layer Meteorology, 1st ed.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1988.

[11] Mayer, S.; Sandvik, A.; Jonassen, M. O.; Reuder, J. Atmospheric profiling with the UAS SUMO: A new perspective for the evaluation of fine-scale atmospheric models. Meteorol. Atmos. Phys. 2012, 116, 15–26.

[12] Hemingway, B.L.; Frazier, A.E.; Elbing, B.R.; Jacob, J.D. Vertical Sampling Scales for Atmospheric Boundary Layer Measurements from Small Unmanned Aircraft Systems (sUAS). Atmosphere 2017, 8, 176.

[13] Fujita, T.T. A Review of Researches on Analytical Mesometeorology; Mesometeorology Project: Department of the Geophysical Sciences, University of Chicago, Chicago, IL, USA, 1962.

[14] LaDue, D.S.; Heinselman, P.L.; Newman, J.F. Strengths and limitations of current radar systems for two stakeholder groups in the southern plains. Bull. Am. Meteorol. Soc. 2010, 91, 899–910.

[15] Frew, E.W.; Elston, J.; Argrow, B.; Houston, A.; Rasmussen, E. Sampling severe local storms and related phenomena: Using unmanned aircraft systems. IEEE Robot. Autom. Mag. 2012, 19, 85–95.

[16] McPherson, R.A.; Fiebrich, C.A.; Crawford, K. C.; Kilby, J.R.; Grimsley, D.L.; Martinez, J.E.; Basara, J.B.; Illston, B.G.; Morris, D.A.; Kloesel, K.A. Statewide monitoring of the mesoscale environment: A technical update on the Oklahoma Mesonet. J. Atmos. Ocean. Tech. 2007, 24, 301–321.

[17] Doviak, R.J.; Zrnic, D.S. Doppler Radar & Weather Observations; Academic Press: Cambridge, MA, USA, 2014.

[18] RoyChowdhury, A.; Sheldon, D.; Maji, S.; Learned-Miller, E. Distinguishing Weather Phenomena from Bird Migration Patterns in Radar Imagery. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 10–17.

[19] Farnsworth, A.; Van Doren, B.M.; Hochachka, W.M.; Sheldon, D.; Winner, K.; Irvine, J.; Geevarghese, J.; Kelling, S. A characterization of autumn nocturnal migration detected by weather surveillance radars in the northeastern USA. Ecol. Appl. 2016, 26, 752–770.

[20] Golbon-Haghighi, M.-H.; Zhang, G.; Li, Y.; Doviak, R.J. Detection of Ground Clutter fromWeather Radar Using a Dual-Polarization and Dual-Scan Method. Atmosphere 2016, 7, 83.

[21] Jensen, J.R. Introductory Digital Image Processing: A Remote Sensing Perspective; Prentice Hall, Inc.: Old Tappan, NJ, USA, 1986.

[22] Frew, E.W.; Elston, J.; Argrow, B.; Houston, A.; Rasmussen, E. Sampling severe local storms and related phenomena: Using unmanned aircraft systems. IEEE Robot. Autom. Mag. 2012, 19, 85–95.

[23] Cassano, J.J. Observations of atmospheric boundary layer temperature profiles with a small unmanned aerial vehicle. Antarct. Sci. 2014, 26, 205–213.

[24] Cook, D.; Strong, P.; Garrett, S.; Marshall, R. A small unmanned aerial system (UAS) for coastal atmospheric research: Preliminary results from New Zealand. J. R. Soc. N. Z. 2013, 43, 108–115.

# VITA

Kaevon Alexander Azartash-Namin

Candidate for the Degree of

Master of Science

Thesis: TRAJECTORY OPTIMIZATION OF METEORLOGICAL SAMPLING

Major Field: Mechanical and Aerospace Engineering

Biographical:

Education:

Completed the requirements for the Master of Science in Mechanical and Aerospace Engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2019.

Completed the requirements for the Bachelor of Science in Mechanical and Aerospace Engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2016.

Experience:

Research Assistant – Oklahoma State University, Stillwater, OK<br>
Teaching Assistant – Oklahoma State University, Stillwater, OK

Professional Memberships:

AIAA