# Development of Unmanned Aerial Vehicle (UAV) Swarm Testbed for Cooperative Applications

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
IN

**ELECTRONICS AND COMMUNICATION ENGINEERING**


Submitted by:

**Akshath Singhal**
**(2K15/EC/020)**

Under the supervision of

Dr. N S RAGHAVA
Professor, Department of Electronics and Communication Engineering




**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042


MAY, 2019

# CERTIFICATE

This is to certify that the project entitled **"Development of Unmanned Aerial Vehicle (UAV) Swarm Testbed for Cooperative Applications"** is being submitted at Delhi Technological University, Delhi for the award of **Bachelor of Technology** in **Electronics and Communication Engineering** degree. It contains the record of bonafide work carried out by **Akshath Singhal** under my supervision and guidance. It is further certified that the work presented here has reached the standard of B.Tech and to the best of my knowledge has not been submitted anywhere else for the award of any degree or diploma.

Date:

Place:


**Dr S Indu**
Head of the Department
Department of ECE
Delhi Technological University

**Dr N S Raghava**
Professor
Department of ECE
Delhi Technological University

# ABSTRACT

The past decade has witnessed an exponential growth in the number of robots and unmanned vehicles in use by defence organisations, industries and general public for a wide variety of uses from surveillance and manufacturing to learning and entertainment. However, the effect is highly significant in the field of unmanned aerial vehicles (UAVs). Owing to the low-cost and ease of availability, the use of multi-rotor UAVs has become a common sight. With roboticists drawing inspiration for ornithopter (flapping wing) UAV designs and increasing presence of these vehicles in airspace, the idea of devising strategies for these vehicles to collaborate like a flock of birds seems both intriguing and promising.

Swarm robotics, though studied in literature for decades, is still a new field of research with implementation and establishment of reliable communication links being major challenges. The following projects provides an overview of current state of aerial robotic swarms and entails the development of a low-cost UAV swarming test-bed which can be used for testing of swarm specific algorithms. The system proposed is decentralised and scalable in nature and allows for dynamic handling of new agents in the environment.

# ACKNOWLEDGEMENT

Setting an endeavor may not always be an easy task; obstacles are bound to come in its way and when this happens, help is welcome and needless to say without help of those people whom I am mentioning here, this endeavor would not have been successful.

The satisfaction that accompanies the successful ongoing of this project would be incomplete without the mention of the people who made it possible, without whose guidance and encouragement would have made efforts go in vain. I consider myself privileged to express gratitude and respect towards all those who guided us.

I convey my sincere thanks to my project guide **Dr N S Raghava** (Professor, Department of Electronics and Communication Engineering) for providing constant support, advice and suggestions which were of a great help.

I am extremely happy to acknowledge and express my sincere gratitude to my parents for their constant suggestions and encouragement.

Akshath Singhal

Roll No: 2K15/EC/020

Date :

Place :

# TABLE OF CONTENTS

# List of Tables

# List of Figures

# Chapter 1

# Introduction

With the constant growth of science and technology in our society, the world has witnessed an overall better life expectancy and reduced mortality rate: owing to improved living standards, better medical facilities and safer man-made environments. This ever-growing population has led to high demands of increased productivity of uniform quality end products for consumption. The industry has thus started shifting towards computer based automation and use of machinery to reduce manufacturing time and eliminate potential human error. These purpose specific machines, *hard automation systems*, are a common practise in-spite of their inflexible and expensive nature. Thus, creating the need for reliable, flexible and low-cost robotic solutions.

The term *Robot* was originally coined in *1920* by Czech playwright Karol Capek in his play *Rossum's Universal Robots*. Robot has been derived from Czech word *robota* meaning **work**. This was used to portray robots as mechanical slaves which were developed to reduce human efforts by replacing humans where similar tasks were to be performed repeatedly.

In 1979, the **Robot Institute of America** (RIA) provided an official definition for a *Robot* as follows:

*"A robot is a reprogrammable, multifunctional manipulator designed to move ma-*

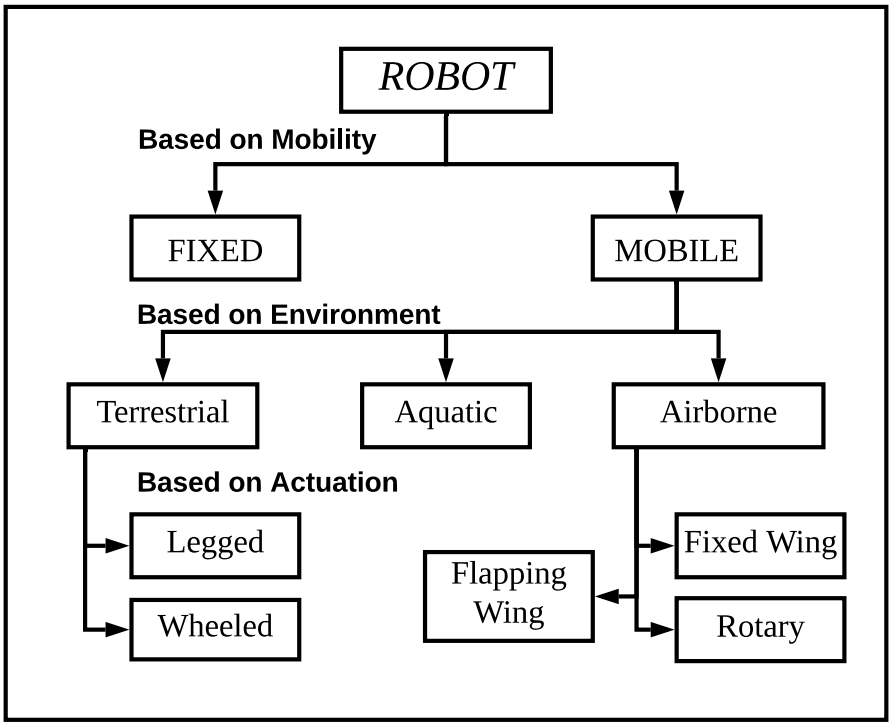



Figure 1.1: Robot Classification

*terial, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks."*

Robots can be broadly categories into *Fixed* or *Mobile* robots based on mobility as depicted in Fig 1.1.

**Fixed Robots**

The definition given above typically refers to a robotic manipulator which is commonly used in the manufacturing industry for the purpose of pick and place operations, welding, assembly and spraying. These robots are usually attached to an immobile base, thus classified as *Fixed robots*.

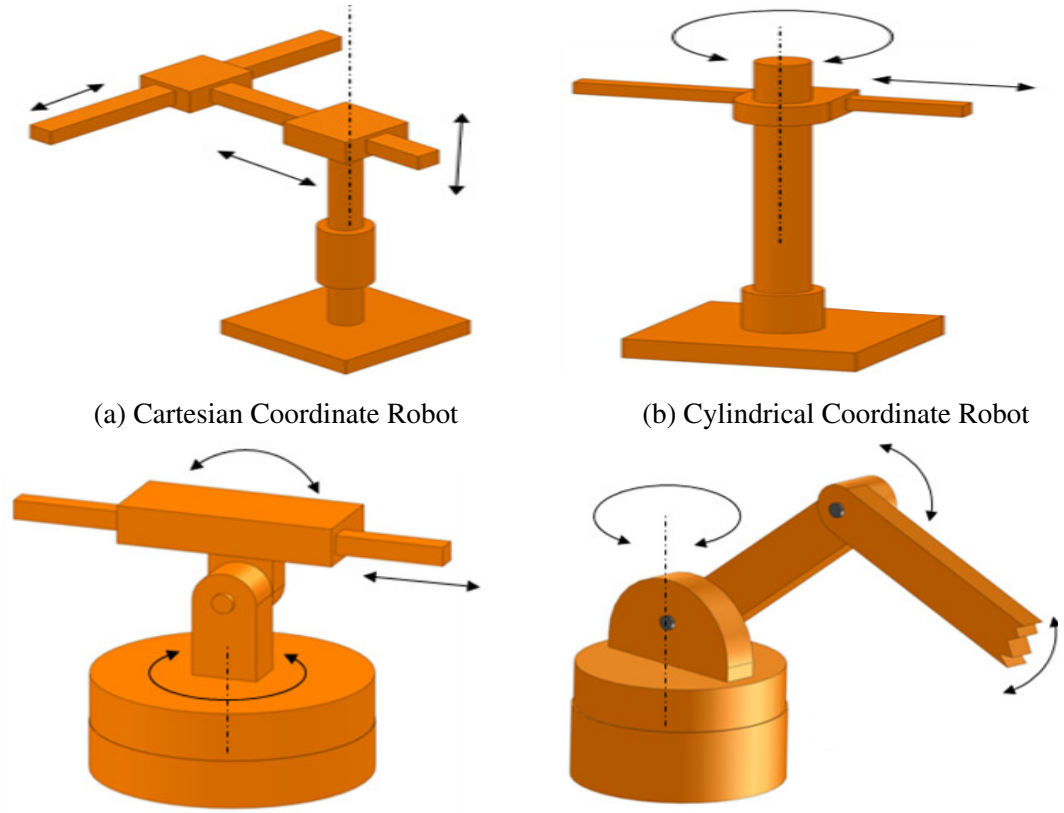
Fixed robots, usually robotic manipulators, consists of a series of links and joints. These joints can be either **Prismatic** (linear) or **Revolute** (rotary). They are broadly divided into the following 4 categories based on their physical configuration as depicted in Fig 1.2:

1. Cartesian Coordinate Robot

2. Cylindrical Coordinate Robot

3. Polar Coordinate Robot

4. Articulated Arm Robot



(a) Cartesian Coordinate Robot (b) Cylindrical Coordinate Robot

(c) Polar Coordinate Robot (d) Articulated Arm Robot

Figure 1.2: Physical Configurations of Fixed Robotic Manipulator[1]

### Mobile Robots

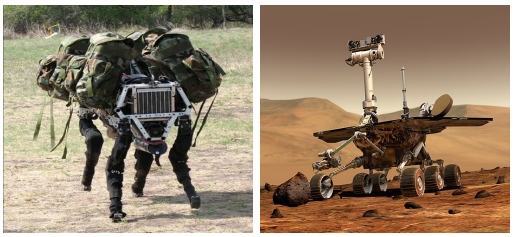
Mobile robots are a class of robots which possess locomotion capabilities. While fixed robotics manipulators have witnessed large growth due to their massive use in the industries, mobile robotics is a relatively new field. The field of mobile robotics has been

[1]Courtesy of https://nptel.ac.in/courses/112103174/module7/lec5/3.html

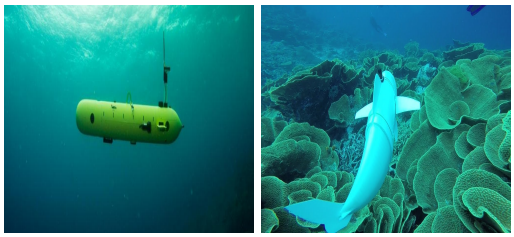
found to gain interest due to open-source projects and independent research work performed by academic researchers. These systems are being tested extensively for use in potential applications such as warehouse automation, surveillance, disaster relief and military applications.

Mobile robots are further classified on the basis of the environment in which they operate into three broad categories as depicted in Fig 1.1 as follows:

1. Terrestrial Robots
2. Aquatic Robots
3. Airborne / Aerial Robots



(a) Terrestrial Robot [Legged, Wheeled]



(b) Aquatic Robot

(c) Aerial Robot [Multirotor, Plane, Helicopter, Flapping Wing]

Figure 1.3: Mobile Robotic Platforms

Fig 1.3 shows some of the common mobile robots from different categories. There also exist some robotic system which can operate in more than one kind of environment, these systems however are out of the scope of this report. The usage of each of these robotic systems vary significantly based on the environment and user. The rest of the report emphasises majorly on the *Aerial Robots*.

## 1.1 Aerial Robots

Aerial robots are a subclass of mobile robotic systems that possess the ability to aviate and navigate in aerial environment, or simply put, these are the robots which can fly. The ability to fly leads to an exponential rise in the applications at the cost of limitations like reduced endurance, higher cost, difficult operation and maintenance. Aerial robots are broadly classified into copters, planes and flapping wing vehicles based on the mechanism used to produce lift for the purpose of flying. Other classifications can be done based on autonomy, presence of pilot and passengers, payload capacity and gross take-off weight. We consider the use of unmanned vehicles only for this report.

An unmanned aerial vehicle (UAV), is an aircraft without a human pilot onboard. Also known as drones, UAVs form a crucial part of an unmanned aerial system (UAS); which includes a UAV, remote operator, ground control station and communication between these. The degree of autonomy of a UAV may range from remote-piloting to autonomous on-board control.

#### Plane

An airplane (informally plane) is a fixed-wing aircraft, powered and propelled forward using the thrust generated from a engine or propeller. They rely on the relative motion of wings with respect to air to generate lift. Airplanes can have a large variety of sizes, shapes, and wing configurations (like high wing, low wing, biplanes, canards, blended wing bodies) which can be observed being used in RC hobby platforms. They are more efficient and thus used for lifting heavy payloads over large distances. The broad spectrum of applications for planes includes recreation, transportation of goods

and people, academic research and military use.

### Flapping Wing

Flapping wing aerial robots, also known as ornithopters, are robots which generate lift by flapping their wings, similar to birds. Flapping wing aerial robots have been developed by taking inspiration from nature. The bio-inspired design provide them vertical takeoff and landing and hover capabilities unlike planes. The relatively new platforms are under research and not commonly used due to high cost. The added advantage of stealth makes them highly suitable for military applications.

### Copter

Copters are the most commonly used aerial robotic platform due to their low cost, high agility, vertical takeoff and landing capabilities and nearly holonomic motion as compared to planes. These comprises of heli-copter and multi-rotor systems which use thrust generated by rotating propellers for lift. The most commonly used copter configuration is quadrotor (4 rotors). While other configurations like bi-copter, tri-copter, hexacopter etc. also exist; a quadrotor provides maximum stability with minimum redundancy. Multi-rotors have become a common platform among RC hobbyists and researchers around the globe. Many open-source projects like PX4 and Ardupilot have made significant contribution in making them popular. Companies like DJI and 3DR have provided a wide range of low cost reliable multi-rotor platforms.


Figure 1.4: RICE R-One platform for multi-robot cooperative missions [2]

## 1.2 Swarm Robotics

Swarm robotics is the study of cooperative behaviour and coordination of multiple simple robots to work as one sophisticated or enhanced system. Emergence of a desired collective behaviour based upon the interactions of multiple robots with each other and with the environment is the goal of swarm robotics. The field has been inspired from nature, studying swarms of insects, flocks of birds and schools of fishes.

The research of swarm robotics is to study the design, physical body, controlling behaviour and cooperative goal of individual robots which is inspired by the emergent behaviour observed in social insects, called swarm intelligence but not limited to it. A large set of complex swarm behaviours can be produced by relatively simple individual rules. Real time communication between the members of the group is a key component for building a system of constant feedback. The swarm behaviour involves constant change of individuals in cooperation with others, as well as the behaviour of the whole group.

Most of the systems developed for the application of swarm algorithms are compact size indoor ground robots localised using ceiling cameras. Some of the basic compo-

[2]Courtesy of http://mrsl.rice.edu/projects/r-one

nents needed for such systems include:

1. Motors and motor drivers
2. Wireless connectivity
3. Indoor localisation unit

Other components which are not necessary but usually found are:

1. Compass
2. Accelerometers and gyroscopes
3. Barometer
4. IR skirt

One such system is R-One platform demostrated in Fig. 1.4

Some of the commonly used robots for swarm implementation have been studied in [1] and given in I.

Table I: Summary of the commonly used robotic platforms for swarms

| Name | Size(mm) | Actuators | Sensors | Communications | Relative Positioning Systems | Development |
|---|---|---|---|---|---|---|
| Khepera | 55(dia) | Wheeled | 8 IR | RS232 Wired Link | - | Research commercial |
| Khepera III | 120(dia) | Wheeled | 11 IR, 5 Ultrasound | Wifi and Bluetooth | IR based | Research commercial |
| ePuck | 75(dia) | Wheeler | 11 IR, contact ring, colour camera | Bluetooth | IR based | OpenSource Research commercial |
| Alice | 20x20 | Wheeled | IR proximity and linear camera | Radio | - | Research Non-commercial |
| Jasmine | 23x23 | Wheeled | 8 IR | Infrared | IR based | Opensource Research |
| S-bot | 120(dia) | Wheeled | 15 proximity omnicamera, microphone, temperature | WiFi | Camera Based | Research Non-commercial |
| Kobot | 120(dia) | Wheeled | 8 IR, Colour camera | Zigbee | IR based | Research Non-commercial |
| Swarmbot | 127x127 | Wheeled | IR, light, contact, camera | IR based | IR based | Research Non-commercial |

# Chapter 2

# Aerial Robotic Swarms

Aerial Robotic Swarms refer to a group of autonomous unmanned aerial vehicles capable of coordinating and interacting with each other in a shared environment for performing tasks to achieve a common goal.

The idea of UAV swarms has been directly inspired from the flocks of birds travelling together in nature. Birds travelling long distances for migration have been observed to maintain formations which reduce the overall drag on the flock thereby increasing efficiency and saving energy. Travelling in groups also provide benefits like increased navigational accuracy and protection from predators as observed in groups of other animals and insects. Use of multiple coordinated and cooperative vehicles helps increase reliability, redundancy, area coverage and save time.

(a) Flock of cranes (b) Planes in formation

Figure 2.1: Formation Flying Analogy

## Applications

The consumer base for UAVs comprises of entertainment industry (aerial photography), mapping industry (mosaicing of aerial images to create maps), mining industry (for creating 3D maps of the mines), precision agriculture and inspection of difficult to reach areas. However, the most important yet generally unknown users of UAVs are law enforcement and defense forces. The UAVs are very commonly used by law enforcement authorities for tasks like surveillance, crowd control and encroachment detection. In the recent years, use of UAVs by first responders going for aid in natural disaster scenario has increased considerably, highlighting their capabilities like never before.

The large and varied requirements of a disaster scenario makes it very difficult to design and develop an optimal and efficient UAV and hence each disaster scenario requires a different UAV. Also, the UAV developed to cover the whole affected area while providing ample situation's awareness tends be expensive, large and complex to operate and maintain. Failure of one such UAV may compromise the whole mission. As widely believed, multiple entities are always better than one. Thus, the author propose development of a testbed to help in implementing the popular algorithms developed for utilizing swarm behaviour in such life-altering situations. The ability to use many small vehicles instead of one large vehicle will increase efficiency, area coverage and situation's awareness while minimizing the risk of mission compromise.

Potential applications for a swarm of aerial robot are:

1. **Search and Rescue Operations** around the globe have witnessed increasing use of UAVs as first responders acting as eyes and ears for the rescue teams scanning a wide area for possible survivors and delivering immediate medical supplies

wherever required. Using multiple coordinated vehicles helps in sensor sharing and covering larger areas. Recent natural disasters like Cyclone Fani in Orissa and Chennai floods of 2018 were major grounds for use of UAVs for search and rescue missions.

2. **Surveillance and Military Use** of UAV swarms includes aerial photography and videography and delivery of necessary payload or explosives. The presence of multiple vehicles with different capabilities allows for use of more sensors and varied payloads.

3. **Entertainment** industry has witnessed the maximum usage of aerial swarms specially as a way to replace fireworks. The largest multi-rotor swarm demonstrations have been in the form of light shows by companies like Intel and Ehang.

4. **Payload Transfer and Construction** applications of aerial swarms are under research at University of Pennsylvania and ETH Zurich.

## Features

Various features commonly witnessed in robotic swarms can be classified as follows:

1. **Aggregation** deals with spatially grouping all robots together in a region of the environment. It is used to get robots in a swarm sufficiently close together and can be used as a starting point for performing some additional tasks, such as communication with limited range. Aggregation near points of interest can be viewed as the first step of more complex tasks, such as collective transportation where objects of interest need to be transported by several robots.

2. **Pattern formation** is the deployment of robots into the environment forming

a predefined geometric shape or pattern like a square, a circle, a line, a lattice etc. Pattern formation enables preserving communication range and overcoming environment limitations (e.g., passing a narrow passage by forming a chain).

3. **Self-assembly** refers to the phenomenon of physical connection of robots to each other resulting in the formation of a particular structure. Self assembly is used to increase the pulling power of the robots, provide stability to the robot swarm while moving on rough terrains, form a connected structure to guide other swarm robots, assemble structures used to overcome holes that a single robot would fall into and to combine capabilities of heterogeneous robots. Self-assembly is studied in several large-scale research projects such as SWARM-BOTS, Symbrion, Swarmanoid , and Replicator.

4. In **swarm-guided** navigation robots of the swarm are navigated by other members of the swarm. Robots are not aware of their actual location or the location of the target. Instead, the swarm is guided by directions supplied by previously deployed robots forming a communication relay. Examples include robots forming a chain from a prey to the nest and indicating directions to other robots in a foraging task, navigation via exchanging navigation messages and flying robots navigating wheeled robots.

## Advantages

As stated earlier, the use of multiple robots in a coordinated manner has many advantages over use of a single, larger, bulkier and more expensive system capable of performing the same mission. Some of these advantages are listed below:

1. Maximize Area Coverage

   Use of multiple vehicles increases the area coverage by dividing overall area into smaller cells to be covered by individual vehicles.

2. Increased Reliability and Redundancy

   The loss of one vehicle doesn't sabotage the whole mission.

3. Sensor and Payload Sharing

   The vehicles in an aerial swarm need not be identical to each other. This allows for use of a rich sensor suite and payload distributed among various vehicles.

4. Reduction in size and Loss

   The size of an aerial robot grows rapidly with increase in payload weight. Thus, splitting of payload leads to smaller vehicles. The loss caused by failure of a smaller vehicle is much lesser.

5. Parallel Execution of task saving time

## Disadvantages

While robotic swarms increase the potential applications and use cases, they possess some limitations when compared to a single robot with similar capabilities. These limitations are as follows:

1. Real time high bandwidth communication is required for swarm operations to exchange information between vehicles in the swarm and the ground control station.

2. Increased operational complexity due to launching and planning multiple vehicles.

3. Requirement of more human operators to deal with multiple vehicles.

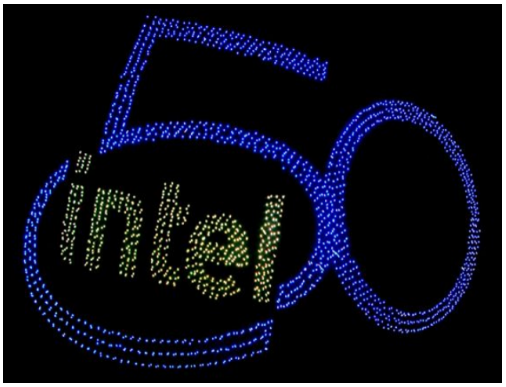
## 2.1 Current State of the Art

The state of the art UAV swarms implemented so far have been observed for academic research, entertainment or military applications only.

The United States of America and Chinese defence forces have successfully tested swarms of more than 50 fixed wing aircrafts in year 2015 and 2018 respectively. The details of these systems (centralized or decentralised, cooperative or time-varying trajectories etc) has not been provided in details.

Companies like Intel and Ehang have demonstrated swarms of more than 1000 micro multirotors for the purpose of entertainment using time-varying trajectories and a centralised system.

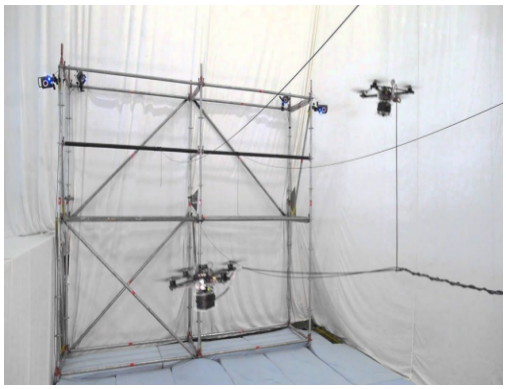
Scholars from various esteemed universities and institutes like Georgia Institute of Technology, University of Pennsylvania, ETH Zurich, National University of Singapore, Czech Technical University and École polytechnique fédérale de Lausanne (EPFL) are working in the field of swarm algorithm implementation. The GRASP lab headed by Dr. Vijay Kumar at University of Pennsylvania and the flying robotics club at ETH Zurich have set new benchmarks by creating algorithms that utilise the potential of swarms like never before and implementing them on small groups of quadcopters. Some of these systems are depicted in Fig. 2.2




(a) Intel Drone Show

(b) Naval Postgraduate School, US

(c) University of Pennsylvania



(d) ETH Zurich

Figure 2.2: Current state of UAV Swarms

## 2.2 Challenges

Development and implementation of aerial swarms is still under research and faces many challenges, some of these are:

1. Lack of reliable Communication system
2. Precise Localisation
3. High Cost

# Chapter 3

# System Design

The system design of a testbed for aerial robotic swarms in the given timeframe involved many independent sub-systems to be developed. An incremental yet rapid prototyping approach, as depicted in Fig. 3.1, was followed for development of a Multi-rotor unmanned aerial vehicle which could be used in both indoor and outdoor case scenarios with intra-swarm communication capabilities. The high level system requirements were analysed and sub-systems identified, and developed in parallel. The subsystems so developed were individually tested and integrated into the final design.

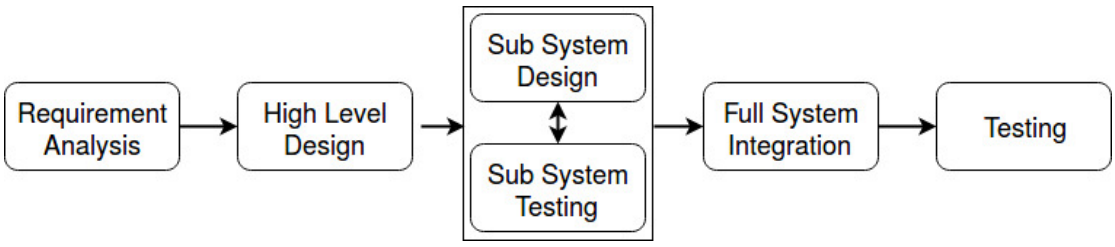



Figure 3.1: System Design Approach

## 3.1 System Architecture

The overall system architecture required for a aerial swarm can be divided into two parts:

**On-board system**

The on-board system comprises of the following components present on the aerial vehicle:

1. A fully autonomous unmanned aerial vehicle capable of autonomous flight and navigation using flight controller, localisation unit (GPS), propulsion system, telemetry and RC receiver.

2. Intraswarm communication module capable of handling Ad-hoc network and exchanging current state of the vehicles.

3. An on-board computer capable of communicating with the flight controller and the intraswarm communication module.

The on-board system architecture is depicted in Fig 3.2

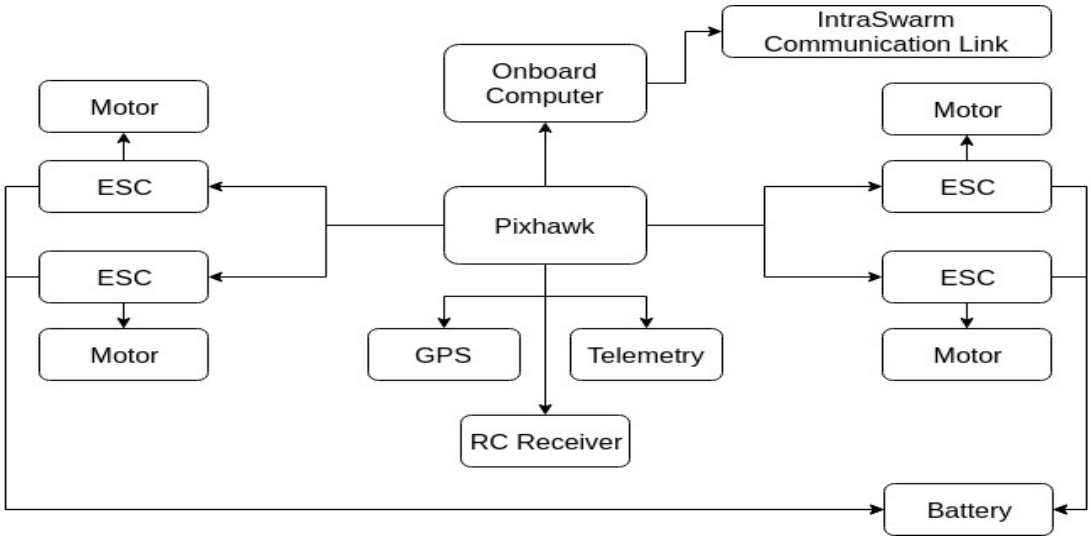



Figure 3.2: On-board System Architecture

**Ground Control System**

The ground control system comprises of a computer with telemetry communication units and a ground station software capable of handling and commanding multiple ve-

hicles. Another computer is used to remote-logging into the on-board computer for running the predefined scripts for communication and running different missions.

## 3.2 Autopilot

An autopilot is a software stack that uses the information from the sensors connected to a flight controller and the desired mission information to guide the vehicle from one point to the other using the actuators connected to the flight controller. A flight controller is a micro computer hardware with various sensors, actuators and motor drivers connected to it for detecting the current state of the vehicle and controlling it. There exist a number of hobby-grade autopilot stacks and most of the flight controllers come with software already loaded that has an autopilot function.

### 3.2.1 ArduPilot

*ArduPilot is an open source, unmanned vehicle Autopilot Software Suite, capable of controlling autonomous*:

- Multirotor drones
- Boats
- Fixed-wing and VTOL aircraft
- Submarines
- Helicopters
- Antenna trackers
- Ground rovers

ArduPilot which was initially developed as a hobby grade autopilot has evolved to a full-featured autopilot and is being used by professionals in the industry, researchers, as well as amateurs. It's extensive usage can be owed to the growing community of developers that have ensured that ArduPilot consists of all the features and libraries required by any roboticist working in the corresponding field. Ardupilot's source code is stored and managed on GitHub. The navigation software present in the ArduPilot is known as firmware when it is compiled to binary form for micro-controller hardware targets, it runs on vehicles such as copter, plane, rover, antenna tracker and subs. The ground controlling station software includes Mission Planer, APM Planner, QGroundControl, MavProxy.

### 3.2.2 Off-board Control

Off-board control of an autonomous vehicle involves controlling the flight behaviour by providing actuation commands or lowlevel targets in **Guided** mode using a computer (on-board or ground based) in the form of mavlink messages over telemetry or serial connection. The system suggested in this report uses Ardupilot software stack for flight controller which allows the multi-rotor to receive commanded linear and angular velocities in all three axis from an on-board computer over serial connection.

## 3.3 Communication System

The proposed system requires communication between multiple vehicles and between each vehicle and ground station. The required communication can be divided into 3 categories as follows:

- **Telemetry** connection between vehicle and ground control station.

- **Remote control** communication to ensure safety in case of unexpected behaviour.
- **Intra-swarm** communication to transfer information between vehicles for coordination.

Due to the presence of plenty of custom off the shelf (COTS) telemetry and remote control modules in the market, we only cover the **intra-swarm** communication module in detail.

### 3.3.1 Intra-Swarm Ad-Hoc Network

The intra swarm communication module requires efficient and real-time transfer of state of a UAV to all other vehicles in the swarm in order to maintain coordination and avoid collisions. Since each vehicle requires information of every vehicle in its neighbourhood, we need a mesh topology. Also, the author chose to use Ad-hoc network instead of a centralised station and access point approach to ensure the crash of one vehicle doesnt render the rest of the swarm useless.

A decentralised network that does not rely on pre-built infrastructure or devices such as routers and access points is known as an ad-hoc network. WANET i.e. wireless ad-hoc network and MANET i.e. mobile ad-hoc network are commonly used ad-hoc networks. Ad-hoc networks do not require a connection with a ground control station and the messages are transferred between individual nodes directly. It relies on a set of routing algorithms that dynamically determine the flow of messages between the nodes depending on the connectivity, traffic and cost of transfer of messages. However, essential information is shared with the ground control station with the help of ”star” approach where in the individual nodes that is the UAVs in our case take turns to transfer information to the ground control station. An ad-hoc network is an essential

requirement of swarm of robots to ensure that the whole swarm system is not affected by the failure, removal or addition of nodes. This feature will make the swarm system more flexible and reliable. Ad-hoc networking allows local information sharing and distributive decision making which makes the swarm capable of resolving a problem in the most efficient and effective way.

Ad-Hoc networks, being self-configuring in nature, allow devices to create and join networks ”on the fly”. With wireless ad-hoc networks embedded in UAVs, multiple UAVs can work collaboratively like a team to complete a task.

## 3.4 ROS Architecture

Robotics Operating System is an open-source, meta-operating system for robotics applications. Despite not being a complete operating system, it provides services like hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management which are usually found in a completely developed operating system. ROS also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. Despite the importance of reactivity and low latency in robot control, ROS itself is not a real-time OS (RTOS). It is possible, however, to integrate ROS with real-time code. Use of ROS architecture allows easy scaling up of the swarm and provides an easy to use and highly accessible interface to users for testing new algorithms rapidly.

### 3.4.1 MAVLink ROS Bridge

MAVlink is the communication protocol developed for sending and receiving messages from micro autonomous vehicle to ground control station (laptop). The library has been

around since 2009 and is supported and used by more than 95% of open-source flight firmwares and ground control station softwares. Since the autopilot can only respond to mavlink commands, it is necessary to create a mavlink to ros bridge to account for the high level of computational requirements for running multiple algorithms. The MAVLink ROS interface is provided by the MAVROS package developed by Vladimir Ermakov for ROS Indigo. [2]

The open-source ROS package however has support for connecting only 1 vehicle at a time. The package provides interface to current state of the system in the form of topics which can be subscribed to using other nodes to work upon this data. Other topics like setpoint wait for ROS nodes to publish data to act upon. Other features like mode change and arming/disarming are provided by the use of services. The position of vehicle in the local frame can be accessed using /mavros/local_position/pose topic or /mavros/global_position/global to access GPS coordinates.

The MAVROS package was changed in order to allow multiple vehicles to connect to itself by making the use of namespaces. Namespaces act like a mask in order to change the name of any ROS node, topic or service. Namespaces were utilised to avoid the problem of overriding names when running multiple MAVROS instances. All the topics, nodes and services related to mavros have the common '/mavros/..' keyword at the beginning of their names. Thus, launching multiple MAVROS instances forced the names of two or more entities to be same causing the ROS master to crash. The problem was solved by renaming the nodes, topics and services to "/id/mavros/.." which not only helped solve the override problem but also allowed easier identification and differentiation of vehicles. The topics used for the purpose of localisation and control of multi-rotor are as follows:

- /id/mavros/local_position/pose
- /id/mavros/global_position/global
- id/mavros/VFR_Hud
- id/mavros/setpoint_velocity/cmd_vel

Various other services have also been used to change mode of operation of vehicle and to arm and disarm the vehicles. Fig. 3.3 shows the graph of major topics and nodes being used by three vehicles sharing a common swarm architecture.

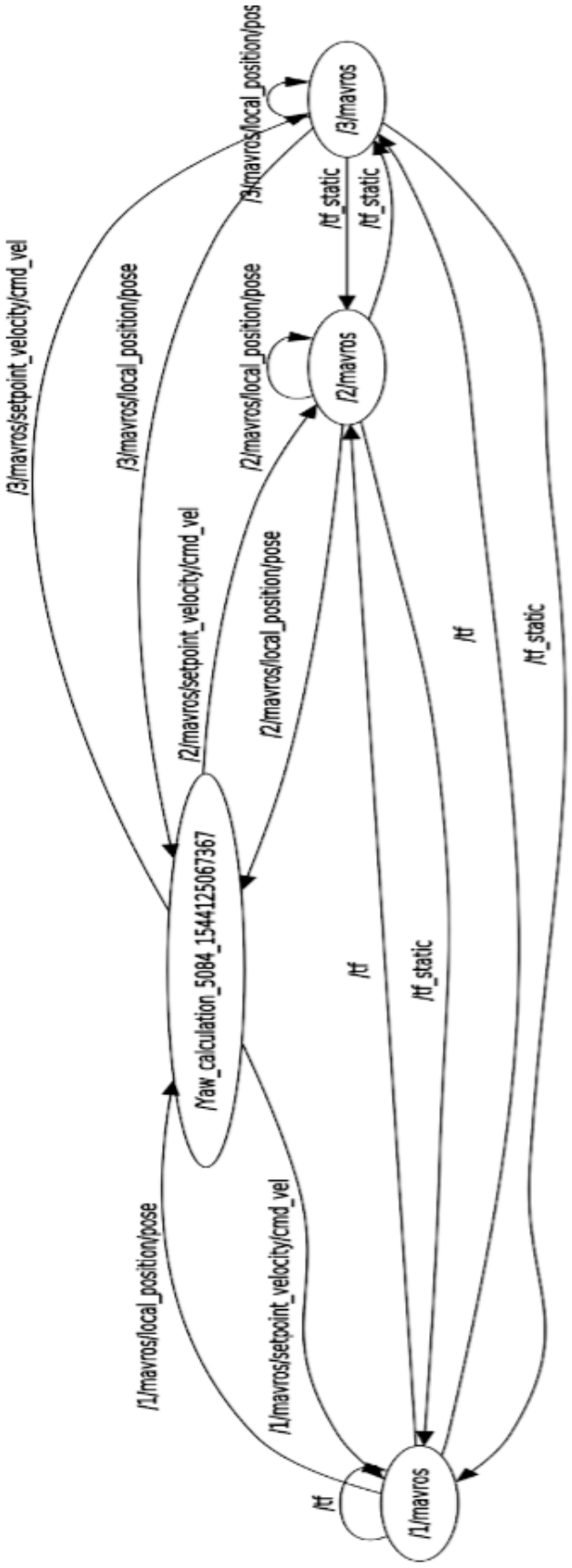



Figure 3.3: Communication Node graph for three vehicles

# Chapter 4

# Hardware Design

The hardware design chapter deals with the designing of the quadrotor vehicle to be used for testing and the component selection process. The design is based on optimising the trade-off between the following constraints:

- Size of the vehicle
- Endurance
- Cost
- Payload capacity (weight and volume)

The multirotor was designed such as to maximize the endurance while minimizing the footprint to ensure longer testing as well as operational time.

## 4.1 Size and Payload Estimation

The multirotor was designed such as to maximize the endurance while minimizing the footprint to ensure longer testing as well as operational time.

### 4.1.1 Size Estimation

To minimize the cost of development and increase the number of units in a confined environment, constrained by the size of the avionics subsystems required, it was decided to go for the easily available low-cost 450mm frame. In addition to the above mentioned benefits, the small size of the multi-rotor is also less prone to damage in the event of a crash.

### 4.1.2 Payload Estimation

A brief estimate of all the requisite payload is made for size and weight required for full system mission and given in II.

Table II: Payload Estimation

| **Payload** | **Estimated weight** |
|---|---|
| Autopilot System(Autopilot, GPS, Wires) | 90-110 g |
| Communication System(RC Link, Telemetry) | 30-40 g |
| Propulsion system (Motor, ESCs, Battery) | 550-600 g |
| Mechanical Frame | 120-140 g |
| Onboard Computer | 40-50 g |
| Miscellaneous | 70-110 g |
| **Total** | 900-1050 g |

## 4.2 Avionics Selection

Avionics components form an integral part of an unmanned aerial vehicle. These comprise of all the major electronic components on-board including flight controllers, sensors, motor drivers, communication systems and computational units.

### 4.2.1 Flight Controller and Peripherals

The opensource **Pixhawk** flight controller is chosen for controlling the multirotor due to its reliability, low cost and excellent support for ardupilot software stack. The flight controller is widely used among hobbyists and researchers around the globe.

### 4.2.2 On-board Computer

The on-board computer selected is **Raspberry Pi 3B+** owing to its easy availability, presence of stable linux image, large user community, low weight and cost.

### 4.2.3 Communication System

The communication system selected for the project is **Sik 433MHz Telemetry Radio**, **Flysky 2.4Ghz RC receiver and TPLink WIFI Dongles.** The TPLink Wifi dongles provide added advantage of scale size as compared to routers like **Ubiquiti Bullet**.

## 4.3 Propulsion Selection

With the weight and size estimation completed, it was necessary to design a propulsion system which was optimised for maximum endurance under the given constraints. The availability of products in the market was also considered before making the final selection. The task was thus sub-divided as follows:

### 4.3.1 Motor and Propeller Selection

The things to be considered while selecting the motor and propeller combination are endurance required, average operating temperature and Gross Take-Off Weight (GTOW)

of the system.

The different motor parameters considered were:

1. **Motor Size:** Size of the motor determines the propeller size it can rotate and also the torque it can produce. So, larger motors can move larger propellers as they produce greater torque and thus greater thrust.

2. **kV Rating:** kV rating decides the RPM/volt of the motor. This governs thrust and size of the motor.

3. **Motor Weight:** Emphasis was kept on reducing the weight of the motor to make a lighter multirotor without compromising endurance.

4. **Current Rating:** Higher the current rating of the motor, greater is the power it can deliver for the same battery voltage.

5. **Efficiency:** The motors to be used need to be efficient so that more flight time can be achieved.

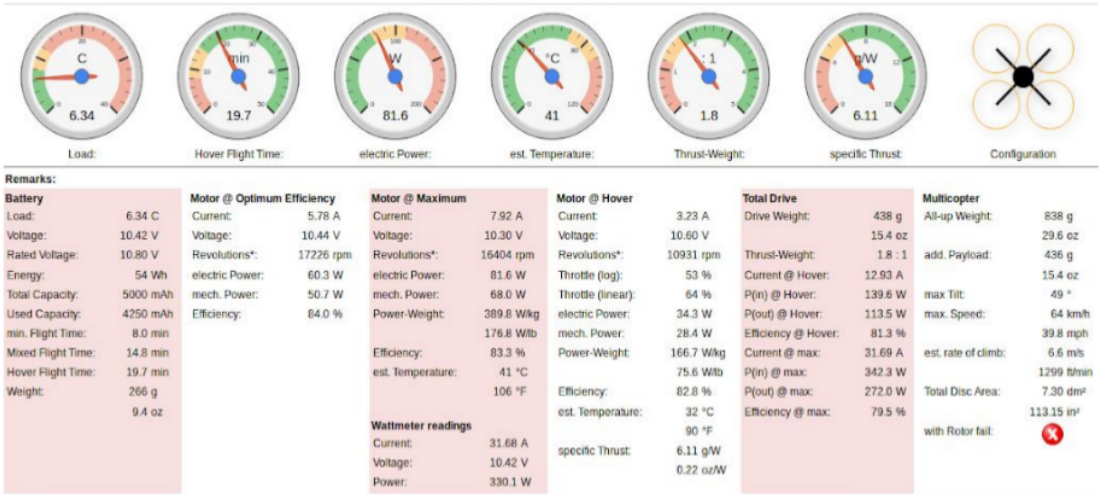



Figure 4.1: ECalc results for selected configuration

Based on these parameters, a market research was carried out to find out the best possible configuration for the multirotor, which was the followed by testing on the online

RC calculator ECalc as depicted in Fig 4.1.

### 4.3.2 Battery Selection

As the battery capacity increases so does its weight. Thus, flight time is not directly proportional to the battery capacity as depicted in 4.2. Maximum current that can be drawn from any battery is equal to the battery capacity times the C- rating.

$$Max\ Current = Battery\ Capacity * C\ rating \tag{4.1}$$

Therefore, for the projected flight time of 18-25 minutes, a 5000mAh LiPo battery was selected.

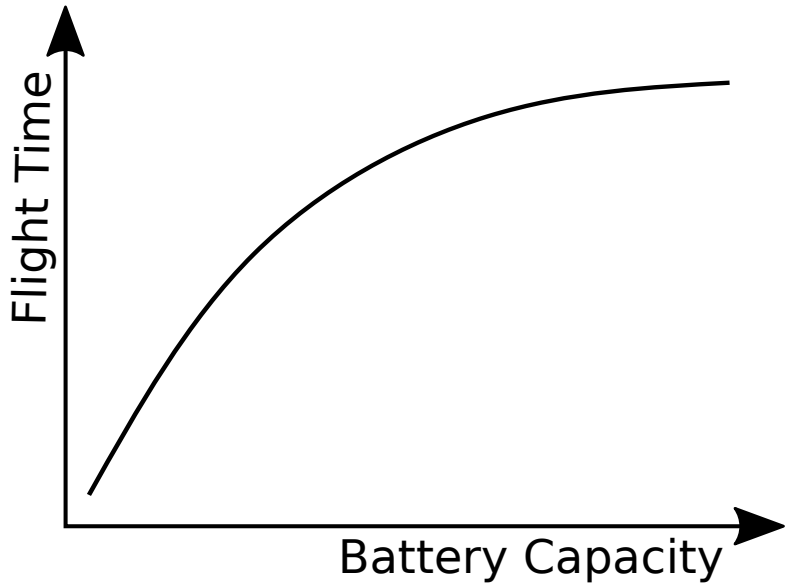



Figure 4.2: Flight Time vs Battery Capacity(mAh)

The final selected components for the quadrotor are tabulated in III and the assembled prototype demonstrated in 4.3.

Table III: Selected Components

| Component | Product |
|---|---|
| Frame | ZMR 450 mm |
| Flight Controller | 3DR Pixhawk 2.4.8 |
| Telemetry | Sik 433MHz Radio |
| Motor | DJI 2212 920kv |
| ESC | Afro 30Amp |
| Radio Control | FlySky 2.4GHz RC |
| Onboard Computer | Raspberry Pi 3B+ |
| Intra-swarm | TPLink Wifi dongle |
| Battery | Tattu 3300mAh 3S Lipo |

Figure 4.3: Prototype developed for testing

# Chapter 5

# Implementation

Implementation of aerial swarm requires development of autonomous unmanned aerial vehicles, intra-swarm communication system and swarming algorithms.

## 5.1 Aerial Vehicle

The unmanned aerial vehicle developed for the aerial swarm was PID tuned for autonomous flights in the presence of a GPS and navigation unit. The system was made capable of off-board control using mavros and python scripts running on the on-board computer which communicates with the flight controller over the serial interface using GPIO pins on raspberry pi.

## 5.2 Intra-swarm Communication

Intra-swarm communication is implemented by configuring the wireless interface on raspberry pi as ad-hoc network and using python scripts to create UDP sockets for data transfer. This configuration enables vehicles to connect to a common network which is self organising in nature. The use of WiFi ad-hoc network for data transfer enables transferring the information throughout the swarm in real time thus allowing

for cooperation in a decentralised fashion. Refer Appendix B for detailed configuration files.

## 5.3 Algorithms

There exist numerous algorithms for swarming applications which ensure formation flight and swarm behaviour while avoiding collisions between vehicles and with the environment. The authors of [3] have studied the use of artificial potential fields for swarm behaviour and collision avoidance by using distance dependent attracting and repulsive fields. The use of optimal control strategy for flocking (aggregation) has been studied in [4].

### 5.3.1 Formation Flight

The formation control refers to the problem of swarm being able to follow a given path and avoid collision while maintaining its pattern formation. There exists numerous ways in which this can be achieved.

The most commonly used approach is leader-follower approach. In leader-follower approach, the pattern formation takes place by maintaining a specified distance and orientation from the leader. This ensures that the vehicles maintain a formation throughout the mission. However, loss of the leader can lead to failure of the whole swarm.

This led to formulation of virtual leader approach in which a virtual leader is simulated and all the vehicles maintain a particular relative distance and orientation.

The use of decentralised approaches depending on the immediate neighbourhood instead of a leader are a little computationally expensive but more robust, scalable and less prone to errors.

The basic equation for artificial potential field methods is given as:

$$u_i = \sum_{j \in \Lambda_i} a(x_i - x_j) \tag{5.1}$$

where $u_i$ is the velocity of $i^{th}$ vehicle, $x_i$ is the position of $i^{th}$ vehicle and $a$ is a function which defines the behaviour of field.

### 5.3.2 Aggregation

The use of a attractive potential field from each neighbouring vehicle leads to aggregation or flocking of the vehicles, attracting them towards each other. Aggregation can be achieved using a negative value of $a$ in Eq 5.1.

### 5.3.3 Dispersion

Presence of repelling potential field from each vehicle to every other vehicle pushes them away from each other leading to dispersion. Dispersion can be achieved using a positive value of $a$ in Eq 5.1.

### 5.3.4 Collision Avoidance

Collision Avoidance is ensured by using a short-ranged but high amplitude exponential repelling field. The use of short ranged field ensures that it does not lead to dispersion or affect the swarm behaviour. A high amplitude exponential field helps in ensuring that the probability of collision is minimized. Collision Avoidance can be achieved using a distance dependent value of $a$ in Eq 5.1.

# Chapter 6

# Testing Results

## 6.1 Software in the loop Simulation

The term 'software-in-the-loop testing' is a test methodology for testing executable codes, algorithms and controller strategies written for a robotic system within a modelling or simulated environment to test the software. Ardupilot, the flight firmware chosen possesses an additional benefit of SITL simulation support which means that actual flight tests are not required in the beginning stages of ROS architecture development and initial testing.

SITL allows to run ArduPilot on PC directly, without any special hardware like flight controller. ArduPilot being a portable autopilot can be run on a very wide variety of platforms allowing SITL testing. A personal computer is just another platform on which ArduPilot can be built and run. When running in SITL the sensor data comes from a flight dynamics model in a flight simulator (generally JSBSim). ArduPilot provides a wide range of built in vehicle simulators, and can interface to several external simulators like flight-gear and Xplane. This allows ArduPilot to be tested on a very wide variety of vehicle types. For example, SITL can simulate:

- multi-rotor aircraft

- fixed wing aircraft
- ground vehicles
- underwater vehicles
- camera gimbals
- antenna trackers
- optional sensors, such as Lidars and optical flow sensors

Its easy to add a new simulated vehicle type or sensor. A big advantage of ArduPilot on SITL is it gives access to the full range of development tools available to desktop C++ development, such as interactive debuggers, static analyzers and dynamic analysis tools making development and testing of new features in ArduPilot much simpler.

### 6.1.1 SITL Architecture

The architecture of the SITL simulation is illustrated in the Fig. 6.1. The arduplane desktop executable file is used to mimic the controls generated by the firmware while Physics of Flight Dynamics are handled by JSBSim and FlightGear. The Ground Control Software used during the SITL simulations is MAVProxy, however, all telemetry data can be forwarded for interfacing to other GCS.

SITL support for Ardupilot was built by the developers for simulating only one vehicle and hence the architecture needs to be modified in order to run different instances of SITL on a common computer and make them compatible for swarm implementation. Some of these changes made to the SITL framework of Ardupilot to support swarming include:

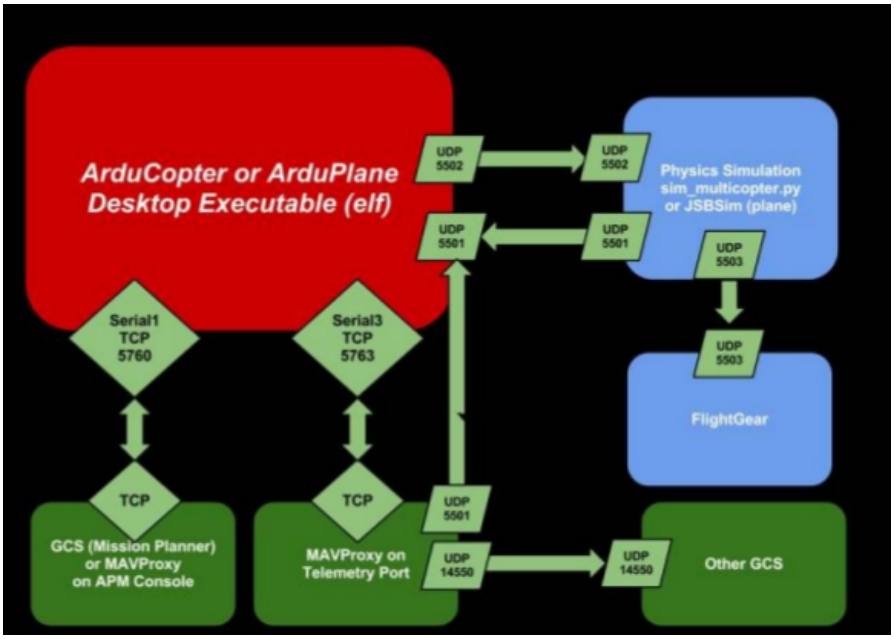



Figure 6.1: Software in the Loop architecture for Ardupilot

- Shifting of vehicle by a certain offset for each new instance
- Assigning a unique MAV_ID to each vehicle
- Change of destination IPs and ports

The changes in the SITL framework was followed by creation of a bash script which takes number of vehicles N and place of origin P as arguments and create N instances of SITL each shifted by safe distance x around the spawn location P. Additional telemetry output ports were added in order to visualise all the vehicles on a common Ground Control Software. MAVLink outputs were also added for MAVROS interfacing. The home location of all the vehicles was kept same using a pymavlink based script which ensured a common local coordinate system for all the vehicles, which further reduces the effort of converting the local coordinate system based codes to global positioning system based coordinate system. Fig. 6.2 and Fig. 6.3 shows a running SITL instance and a group of 3 vehicles connected to a common ground station respectively.

Table IV: SITL Testing Results

| Parameter | Value |
|---|---|
| Maximum Number of vehicles | 10 |
| Total Flight Hours | 58 |
| Number of Formations | 6 |
| Total Missions Performed | 152 |

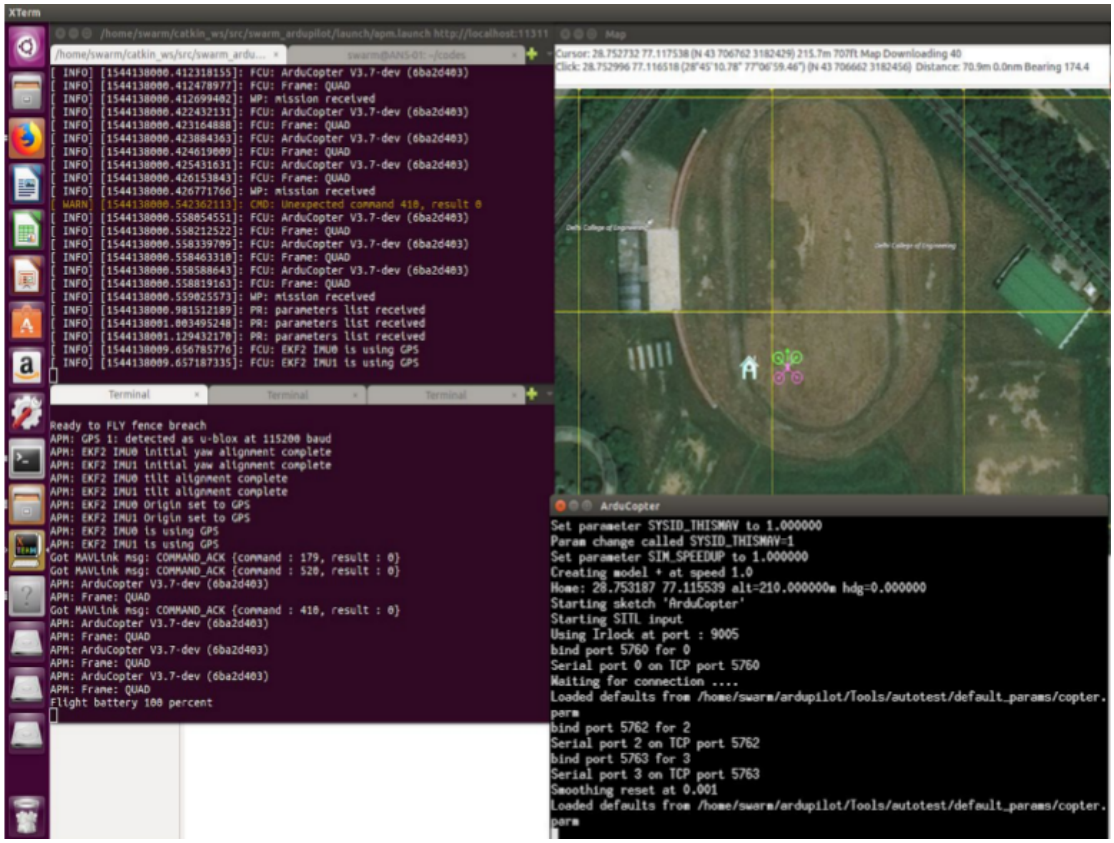



Figure 6.2: Ardupilot SITL instance


Figure 6.3: Multi-UAV SITL simulation

## 6.2 Hardware Testing

Hardware Testing of aerial swarms was done by using 4 vehicles in both centralized and decentralised manner using different communication systems for data transfer.

The centralized approach involved transfer of current state of all vehicles to the ground control station where it was analysed and used for generating output for all the vehicles in the swarm.

The decentralised approach, on the other hand, used on-board computers and wifi ad-hoc intra-swarm network for exchanging current state information and each vehicle's on-board computer computed the desired output for itself. Fig. 6.4 depicts flight testing of 4 vehicles.

Table V: Hardware Testing Results

| Parameter | Value |
|---|---|
| Maximum Number of vehicles | 4 |
| Total Flight Time | 48 mins |
| Number of Formations | 3 |
| Total Missions Performed | 7 |

### 6.2.1 Communication Link

The communication system was tested on ground to ensure reasonable latency at safe distances. A bash script was written to log the latency and the on-board computers tried to ping each other over the ad-hoc network. While addition of new nodes increased the latency initially, the results were negligible after 7 vehicles and the overall system proved to have a latency of less than 150 ms with 9 nodes separated by 10 feet each, providing sufficient margin to avoid collisions despite minor GPS inaccuracies.

Figure 6.4: Flight Testing of 4 UAVs implementing aggregation

### 6.2.2 Flight Testing

Various flight tests were conducted with 3 and 4 multi-rotors to achieve flight formations (grid, line and V) using both centralised and decentralised approaches. A total of 3 crashes were observed due to less safety margin and latency observed in centralised communication approach. The data for the conducted hardware testing is given in V.

# Chapter 7

# Conclusion

The system developed during the project provides a low-cost testbed for aerial robotic swarms for testing of algorithms. The multi-rotors chosen provide a decent payload capacity for additional hardware like a lost cost camera or payload drop modules increasing the applications and usability of the said system.

The presence of a SITL Mavros swarm simulation environment provides a user-friendly rapid prototyping environment for researchers to ensure the reliability of the code before hardware testing. Also, the project helps to bridge the gap between user community and academic research by providing guidelines for easily reproducing the work for the community.

# Chapter 8

# Future Prospects

## 8.1 Indoor Localisation

While outdoor flying and testing of aerial swarms has been ignored majorly and needs to be at greater priority, the increased efficiency and faster testing due to indoor localisation and flying of aerial swarms should not be ignored. Indoor swarms open new opportunities in terms of heterogeneous swarms and their aiding capabilities in disaster scenarios. The major challenges faced in indoor swarm operations is due to unavailability of Global Positioning System. In order to facilitate indoor localisation, ceiling mounted cameras or wifi localisation based POZYX systems are used generally. One such technique is used in the WHYCON image based localisation technique based on detection and tracking. It involves detection of white and black roundel. An initial detection step is used to identify the position of the roundel. After that using camera re-projection techniques and known dimensions of the inner and outer roundels, the three dimensional position of the roundel with respect to the camera is computed. Once detected, the roundel is relocated by template matching and particle tracking in the subsequent frames of video in close proximity to prior position. [5] Localisation can also be done by using multiple wifi beacons and time difference of arrival from various sources for localisation. [6] However, imagery based localisation systems are more accurate and faster in comparison.

## 8.2 Fixed Wing Aircraft

Multi-rotors have gained tremendous popularity amongst drones due to easier flights, less space requirements and less crashes observed in the learning phase, however, fixed wing aircrafts are still preferred owing to their greater efficiency, range and endurance. The swarming of fixed wings, though achieved by Naval Postgraduate School, [7] is very difficult to perform for larger swarms. Thus, shifting to a fixed wing-multirotor hybrid can help in achieving swarms having larger range and endurance by exploiting the capabilities of both.

# Appendix A

# Scripts Used

## A.1 ROS Code Structure

A ROS package was developed in order to perform the following functions using a launch file, thereby, reducing the effort significantly and providing a consolidated environment to the users.

- Launch File for multiple MAVROS subscribers with different namespaces and parametric System Ids
- RosNode to subscribe to multiple topics of multiple vehicles and synchronize the data based on timestamp
- RosNode to handle the subscribed data, convert to a formatted array and publish at 1 single topic
- Ros Nodes to process the data received and generate cmd_vel for each vehicle
- Custom array type messages

An array of custom message vehicle_state is formed to represent the current state of the swarm given as swarm_state.
The swarm_state message is given as:

swarm_ardupilot/vehicle_state[] vehicles

uint8 particles

The vehicle_state messgae comprises of:

float32 x

float32 y

float32 pitch

float32 si

int16 heading

float32 groundspeed

float32 airspeed

int32 id

float64 lat

float64 lon

int32 alt

The following script subscribes to various nodes from different vehicles and provides their current state on a common topic "/publishing". Other nodes can subscribe to this topic and publish command velocities for each vehicle on their respective topics ($/id/mavros/setpoint\_velocity/cmd\_vel$).

```
arr = rospy.Publisher('/publishing', swarm_state, queue_size =
    10)
def callback(*args):
  x=np.zeros(shape=(0))
  y=np.zeros(shape=(0))
  si=np.zeros(shape=(0))
  vel=np.zeros(shape=(0))
  pit=np.zeros(shape=(0))
  al=np.zeros(shape=(0))
  status=swarm_state()
  n=num[0]
  for i in range(0,n):
    x1=args[i].pose.position.x
```

```
    y1=args[i].pose.position.y
    si1=((-1*(args[i+n].heading-90))%360)*3.1416/180
    v1=args[i+n].groundspeed
    x=np.append(x,x1)
    y=np.append(y,y1)
    si=np.append(si,si1)
    vel=np.append(vel,v1)
    al=np.append(al,args[i+n].altitude)
    v=vehicle_state()
    v.x=x1
    v.y=y1
    v.si=si1
    v.heading=args[i+n].heading
    v.groundspeed=v1
    v.id=i
    v.lat=args[i+2*n].latitude
    v.lon=args[i+2*n].longitude
    v.alt=args[i+n].altitude
    v.pitch=euler_from_quaternion(args[i].pose.orientation).pitch
    status.vehicles.append(v)
    status.particles+=1
  arr.publish(status)

  for i in range(0,n):
    [[x1,y1],[x2,y2]]=waypt[i]
    u,a=lqr_coop.lqr_coop(x,y,si,vel,x1,y1,x2,y2,i)
    message = TwistStamped()
    v=vel[i]
    v=v+a*0.25
    if v<3:
      v=3
```

```
    elif v>8:
      v=8
    vx=v*math.cos(si[i])
    message.twist.linear.x = vx
    vy=v*math.sin(si[i])
    message.twist.linear.y = vy
    message.twist.linear.z = 0.0
    message.twist.angular.x = 0.0
    message.twist.angular.y = 0.0
    message.twist.angular.z = 1.0*u#/(2*3.1416)
    pub[i].publish(message)

def Yaw_calculation():
      rospy.init_node('Yaw_calculation', anonymous = True)
  pos=position_arr()
  sta=state_arr()
  nav=navsat_arr()
  for i in range(1,num[0]+1):
    pos.position.append(message_filters.Subscriber
       ('/'+str(i)+'/mavros/local_position/pose',
       PoseStamped))
       sta.state.append(message_filters.Subscriber
          ('/'+str(i)+'/mavros/vfr_hud', VFR_HUD))
    nav.navsat.append(message_filters.Subscriber
       ('/'+str(i)+'/mavros/global_position/global',
       NavSatFix))
      ts = message_filters.ApproximateTimeSynchronizer
         (pos.position+sta.state+nav.navsat, 10, 0.25,
         allow_headerless = True)
      ts.registerCallback(callback)
      rospy.spin()
```

## A.2 Swarming Scripts

```
class swarm_bot:
    def __init__(self,n,s):
        self.id=n
        self.string=str(s)
        self.velocity = [0,0]
        print("Connecting to Vehicle id:",n)
        self.vehicle = connect(self.string)#, wait_ready=True)
        print("Connected")


    def get_pos(self):
        self.pos= [self.vehicle.location.global_frame.lat,
            self.vehicle.location.global_frame.lon]
        return self.pos


    def update_vel(self,v):
        self.velocity=v
        velocity_x=v[0]
        velocity_y=v[1]
        velocity_z=0
        """
        Move vehicle in direction based on specified velocity
            vectors.
        """
        msg = self.vehicle.message_factory.
             set_position_target_global_int_encode(
            0,      # time_boot_ms (not used)
            0, 0,  # target system, target component
```

```
        mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT_INT, #
            frame
        0b0000111111000111, # type_mask (only speeds enabled)
        0, # lat_int - X Position in WGS84 frame in 1e7 *
            meters
        0, # lon_int - Y Position in WGS84 frame in 1e7 *
            meters
        0, # alt - Altitude in meters in AMSL altitude(not
            WGS84 if absolute or relative)
        # altitude above terrain if GLOBAL_TERRAIN_ALT_INT
        velocity_x, # X velocity in NED frame in m/s
        velocity_y, # Y velocity in NED frame in m/s
        velocity_z, # Z velocity in NED frame in m/s
        0, 0, 0, # afx, afy, afz acceleration (not supported
            yet, ignored in GCS_Mavlink)
        0, 0)  # yaw, yaw_rate (not supported yet, ignored in
            GCS_Mavlink)

    self.vehicle.send_mavlink(msg)


def arm_and_takeoff(self, aTargetAltitude):

    print "Basic pre-arm checks",self.id

    while not self.vehicle.is_armable:
        print " Waiting for vehicle to initialise...",self.id
        time.sleep(1)


    print "Arming motors",self.id
```

```
        # Copter should arm in GUIDED mode
        self.vehicle.mode = VehicleMode("GUIDED")
        self.vehicle.armed = True

        while not self.vehicle.armed:
            print " Waiting for arming...",self.id
            time.sleep(1)

        print "Taking off!",self.id
        self.vehicle.simple_takeoff(aTargetAltitude)

        while True:
            print " Altitude: ",
                self.vehicle.location.global_relative_frame.alt
            if self.vehicle.location.global_relative_frame.alt
                >=aTargetAltitude*0.90:
                print "Reached target altitude",self.id
                break
            time.sleep(1)

    def land(self):
        self.vehicle.mode = VehicleMode("LAND")
```

# Appendix B

# Configuration Files

The configuration of Ad-Hoc wifi network is done by using the given commands in $/etc/network/interfaces$ file.

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
auto wlan0
iface wlan0 inet static
address 192.168.1.101
netmask 255.255.255.0
wireless-channel 1
wireless-essid SwarmPiNet
wireless-mode ad-hoc
```

Figure B.1: Interfaces Configuration file

# Appendix C

# MAVLink Protocol

MAVLink is a telemetry protocol designed and developed for unmanned vehicles and other low-bandwidth systems. The binary protocol is well suited for resource-constrained systems and deployed in two versions: v1.0 and v2.0. MAVLink v2.0 is backwards-compatible i.e. it can parse and send v1.0 data packets. Telemetry data which is less critical is sent in multicast streams while critical protocols that effect the system configuration or require guaranteed delivery like the mission or parameter protocol are sent point-to-point with re-transmission. The format of MAVLink protocol has been illustated in Fig C.1.

| | |
|---|---|
| uint8_t magic; | protocol magic marker |
| uint8_t len; | Length of payload |
| uint8_t incompat_flags; | flags that must be understood |
| uint8_t compat_flags; | flags that can be ignored if not understood |
| uint8_t seq; | Sequence of packet |
| uint8_t sysid; | ID of message sender system/aircraft |
| uint8_t compid; | ID of the message sender component |
| uint8_t msgid 0:7; | first 8 bits of the ID of the message |
| uint8_t msgid 8:15; | middle 8 bits of the ID of the message |
| uint8_t msgid 16:23; | last 8 bits of the ID of the message |
| uint8_t payload[max 255]; | A maximum of 255 payload bytes |
| uint16_t checksum; | X.25 CRC |

Figure C.1: MAVLink Protocol Format

# References


[1] Iñaki Navarro and Fernando Matía. An introduction to swarm robotics. *Isrn robotics*, 2013, 2012.

[2] Mavros - mavlink extendable communication node for ros. http://wiki.ros.org/mavros. Accessed: 2019-05-26.

[3] Dong Hun Kim, Hua Wang, and Seiichi Shin. Decentralized control of autonomous swarm systems using artificial potential functions: Analytical design guidelines. *Journal of Intelligent and Robotic Systems*, 45(4):369–394, 2006.

[4] Osamah Saif, Isabelle Fantoni, and Arturo Zavala-Río. Flocking of multiple unmanned aerial vehicles by lqr control. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 222–228. IEEE, 2014.

[5] Matias Nitsche, Tomas Krajnik, Petr Cizek, Marta Mejail, Tom Duckett, et al. Whycon: an efficient, marker-based localization system. 2015.

[6] Chouchang Yang and Huai-Rong Shao. Wifi-based indoor positioning. *IEEE Communications Magazine*, 53(3):150–157, 2015.

[7] Victoria Ochoa. Nps arsenl advances team vs. team swarm research.