

Tiger Rescue Senior Project

Design Documentation

Revision 2 - 5/5/2021

Tyler Baldwin

Dominic Calabria

Jacob Wood

**Table of Contents**

Table of Contents

[Problem Overview 3](#_Toc104985483)

[Domain Model 3](#_Toc104985484)

[High-Level Architecture 4](#_Toc104985485)

[Subsystems 5](#_Toc104985486)

[Flight Controller 5](#_Toc104985487)

[Path Interrupt Controller 5](#_Toc104985488)

[External Communication 6](#_Toc104985489)

[Hardware Interface 8](#_Toc104985490)

[Swarm Coordinator 10](#_Toc104985491)

[Algorithms 10](#_Toc104985492)

[Guaranteed Delivery 10](#_Toc104985493)

[Area Division 10](#_Toc104985494)

[Path Planning 11](#_Toc104985495)

[Target Detection 11](#_Toc104985496)

[References 11](#_Toc104985497)

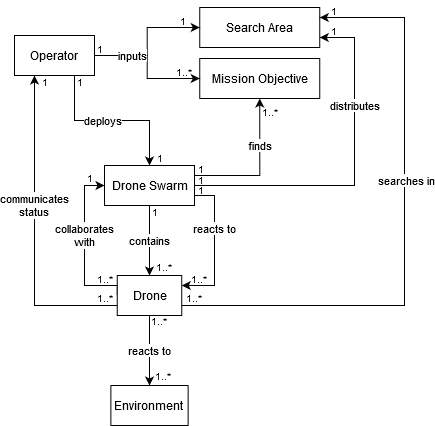
## Problem Overview

We are developing a distributed drone search and rescue system. The primary goal is to provide an area defined by an operator and to use any number of drones to efficiently divide and search the area for a target to rescue. A practical situation would be to use the system to look for a child that wandered away from home and is lost in a large cornfield. The drones are not aware of the topography of the area or what obstacles they may encounter but must navigate the field as quickly as possible and react in real time to ensure the child is returned safely. The system must be able to support any amount of concurrent drones that all operate under their own navigation systems. The operator of the swarm should be able to see the status of all of the drones at any time.

## Domain Model

A domain model was derived from the project requirements to provide a high-level conceptual overview of the system. This domain includes:

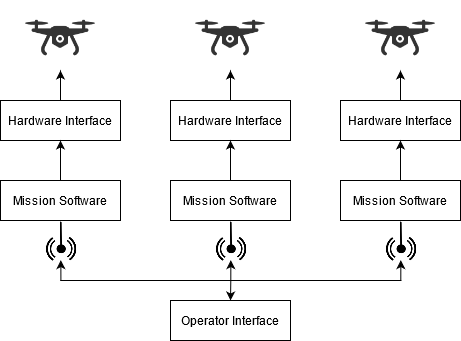
* Operator: system actor
  + Defines search area and mission objective
  + Deploys drone swarm
* Search Area: area to search for the mission objective, described by coordinates
* Mission Objective: search target (e.g. human)
* Drone Swarm: 1 to N drones
* Drone
  + Collaborates with rest of drone swarm to find mission objective in search area
  + Reacts to environment
* Environment: obstacles, positive signs of mission objective



## High-Level Architecture

The core component of the drone search and navigation system is highly modular. The design allows for one instance of the core system to run per drone hardware - a key aspect of mocked onboard compute. A number of subsystems support the system in communicating with the swarm and operator, executing the search algorithm, and monitoring sensor data. Two separate components exist within this architecture: the core mission software, and the drone SDK wrapper. These components are executed as separate processes, bridged by the Hardware Interface subsystem. However, if an SDK implementation permitted direct usage of Java, these components would be merged into a single process.

High Level Interaction Diagram



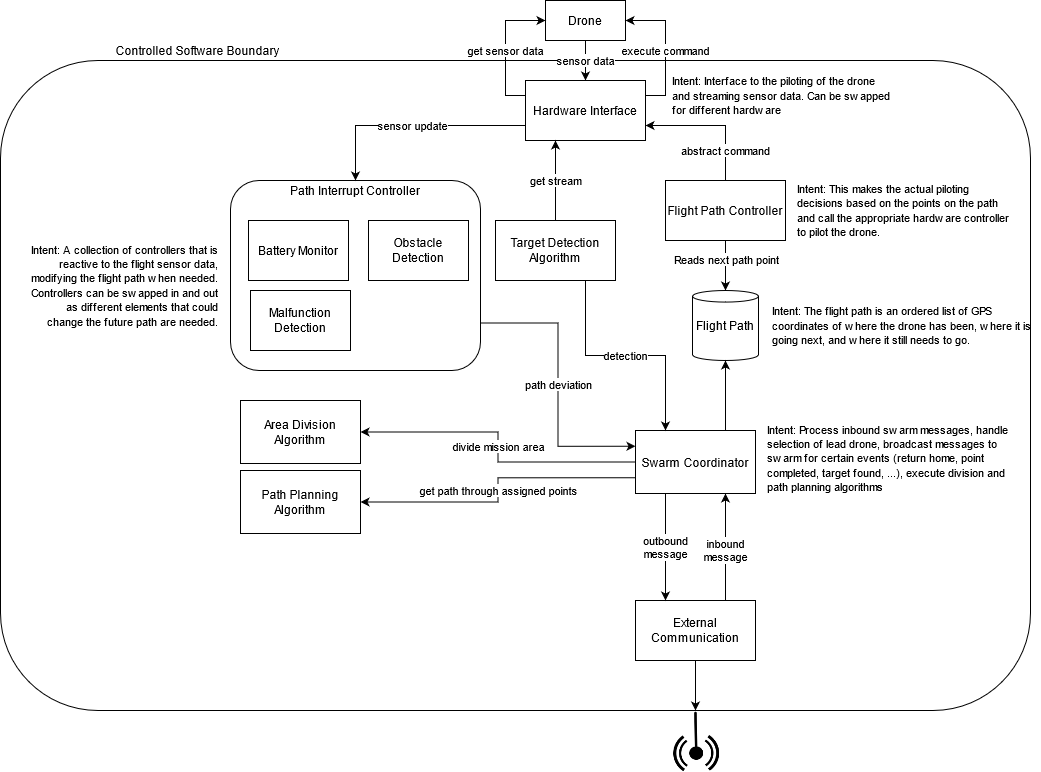
The core mission component is a reactive system that is responsible for area division consensus, swarm status monitoring, path planning, target detection, and path execution.

At the beginning of the mission, the swarm elects a lead drone that is responsible for decomposing and distributing the mission area. Each drone will then plan and execute a route through its assigned area points. During execution of this path, a stream of the drone’s downward-facing camera is fed to the target identification algorithm. If a target is detected, the drone’s current state is recorded and sent to the operator console along with the associated full video frame.

During the mission, the swarm continuously monitors the status of each drone. If a drone is detected to be down, its remaining points are re-distributed by the leader to the remaining drones, which will then update their flight paths accordingly. If the unreachable drone was the current leader, a new leader is selected.

The operator may request a detection to be focused on, in which case a new task will be created and distributed by the lead drone - once again resulting in the drone’s updating their flight paths as needed.

Mission System High Level Architecture



## Subsystems

### Flight Controller

The Flight Controller handles flight plan management and execution. At startup, a waypoint-based flight plan is established for the drone’s assigned points using the search algorithm. During flight, this path can be modified if the drone is assigned new points, such as in the case of another drone becoming unreachable. The flight plan is executed through interaction with the Hardware Interface. The Flight Controller does not directly ingest any sensor data - the determination of when a flight command has been fully executed is handled by the Hardware Interface.

### Path Interrupt Controller

The Path Interrupt Controller is responsible for monitoring sensor data and changing the flight plan based on this data when needed. The subsystem is composed of a set of Path Interrupt Modules that operate independently for extensibility. Currently, the only implemented module is the Low Battery Monitor. Future modules could include Obstacle Avoidance and Malfunction Detection. These modules are intended to be *reactive* - it is expected that the Hardware Interface will stream sensor data as opposed to the modules needing to continuously query sensors.

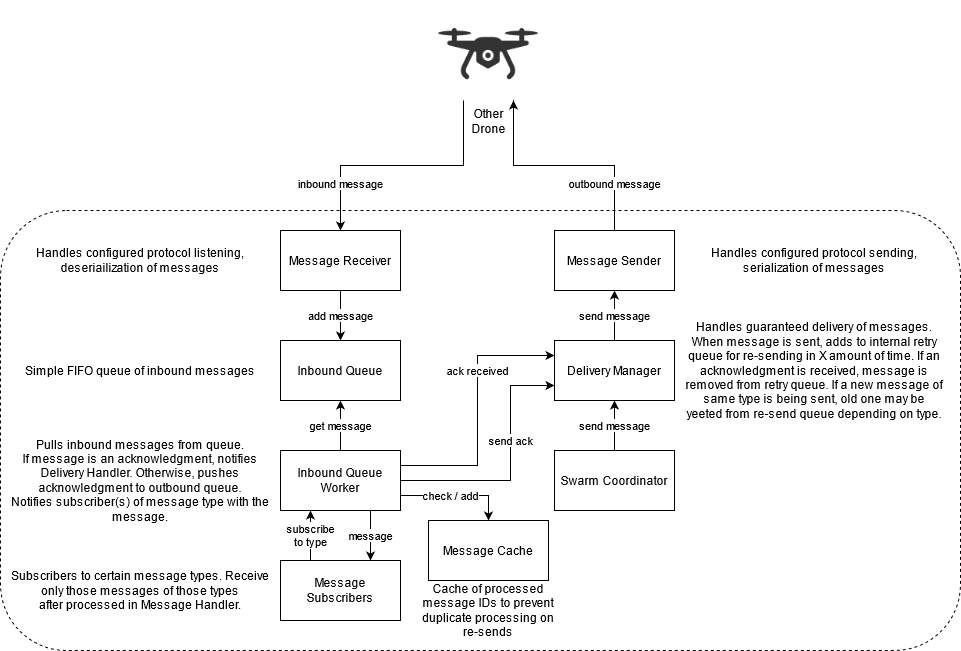
**Low Battery Monitor**

To ensure a drone will not die mid-flight, the Low Battery Monitor checks its battery percentage. Should the percentage drop below a configured abort amount, the Flight Path Controller is instructed to immediately fly the drone back to its home position. This has a side-effect of the drone informing the swarm that its remaining points need to be re-allocated.

### External Communication

The External Communication subsystem is responsible for drone-drone and drone-operator messaging. This is the only major subsystem that is leveraged both in the mission and operator software components.

Custom queueing and guaranteed delivery mechanisms were chosen for extensibility. We assessed that our use case of an IP network may be unlikely in non-mocked onboard compute scenarios, so we avoided usage of traditional IP-based solutions like ZeroMQ. With this design, different protocols can be easily supported by adding implementations for the Message Receiver and Message Sender components. Elsewhere, the underlying communication protocol is completely abstracted away.



**Messages**

Many message types have been defined to support the functionality of the system. These messages are serialized using protocol buffers and currently are not transmitted with any encryption.

Table 1: Current external communication messages.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Source** | **Destination** |
| Acknowledgement | Message acknowledgement used in the Guaranteed Delivery subsystem. | Any | Reply |
| Mission Definition | Mission parameters including swarm/operator network information, search area bounds, operating altitude and altitude separation (vertical deconfliction), and target type. | Operator | Swarm |
| Coverage State | Remaining point and altitude assignments. Sent at beginning of mission and when points need to be re-distributed or a new task is created. | Lead Drone | Swarm |
| Location Query | Request drone to reply with its current location. | Any | Drone |
| Location Status | Current location (latitude, longitude, altitude) in response to Location Query. | Drone | Reply |
| Point Completed | Assigned point (from Coverage State) completed by drone. Swarm updates Coverage State to reflect the point as completed. | Drone | Swarm |
| Target Detection | Target detection information including location, confidence, and an image of the detection (image sent in chunks). | Drone | Operator |
| Focus Detection Candidate | A candidate task for focusing on a target detection (positioning drone and camera to center and zoom on target) sent when that detection is made. The task is not added to the Coverage State unless requested by the operator (via Focus Detection message). | Drone | Swarm |
| Focus Detection | Request the swarm to execute a focusing task. The task will be distributed by the lead drone via an updated Coverage State. | Operator | Swarm |
| Return Home | Instruct the receiving drone to return home (ending its role in the mission). Has a side-effect of the drone removing itself from the swarm via a Remove Drone message. | Operator | Drone / Swarm |
| Remove Drone | Remove drone from swarm. Its points will be redistributed by the lead drone through an updated Coverage State message. | Drone | Swarm |

**Implementation: Parrot ANAFI**

Onboard compute is mocked for our Parrot ANAFI implementation, so communication occurs between computers on an IP network. As such, UDP was chosen as our underlying protocol implementation. This choice was made based on the unwanted network overhead of TCP.

### Hardware Interface

The Hardware Interface provides a common bridge between our mission software and drone-specific control software. This interface includes both the ability to send commands and queries to the drone, as well as listen for drone events (i.e. sensor data). A key principle of this system is that it abstracts away drone-specific knowledge to provide extensibility. Currently, the only implementation of this interface is for the Parrot ANAFI, but implementations for other drones would be easily supported without any modification to the rest of the system.

**Commands**

Commands are executed synchronously. For IPC-based implementations, a “Command Completed” event is provided to permit blocking until command execution has completed.

Table 2: Commands supported by Hardware Interface.

|  |  |
| --- | --- |
| **Command / Query** | **Description** |
| Takeoff | Hover drone above ground (low altitude) at current position. Should only be executed from a landed state. |
| Land | Land the drone at its current position. Drone should remain on after landing. |
| Move To | Move to a specified GPS coordinate. Implementation must permit this command to be cancelled mid-execution. |
| Move Gimbal | Move camera gimbal to specified orientation. |
| Change Zoom Level | Change the camera’s zoom level. |
| Get Position | Query the drone’s current location. |
| Get Orientation | Query the drone’s current orientation. |
| Get Connection State | Query the SDK’s connection state with the drone. |

**Drone Events**

It is expected that each drone event will be automatically generated by the Hardware Interface at a reasonable interval. This enables the system to be reactive as sensor data does not have to be continuously queried. However, these events can also be triggered by one of the above queries (e.g. Get Orientation results in an Orientation Status event).

Table 3: Events generated by Hardware Interface implementations.

|  |  |
| --- | --- |
| **Event** | **Description** |
| Battery Status | Current battery percentage remaining. |
| Position Status | Current GPS position (latitude, longitude, relative altitude). Should not be sent when GPS is in an uncalibrated state. |
| Orientation Status | Drone’s current orientation (pitch, yaw, and roll). |
| Connection State | SDK connection state with drone (connected or not connected). |
| Command Completed | Fired when a command is completed (e.g. Move To is completed). |

**Implementation: Olympe**

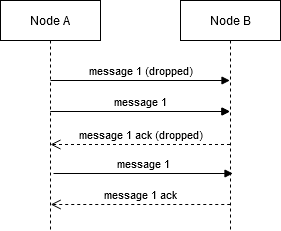
The [Olympe SDK](https://developer.parrot.com/docs/olympe/) is used to interface with Parrot Anafi drones. Because Olympe is strictly a Python SDK, a cross-language bridge was needed to support it. This bridge is implemented using [ZeroMQ](https://zeromq.org/) [PAIR](https://learning-0mq-with-pyzmq.readthedocs.io/en/latest/pyzmq/patterns/pair.html) sockets with Protobuf message serialization. Events listed in Table 2 are automatically generated by Olympe and streamed back to the Java-based Hardware Interface client. Similarly, the Python server uses the Olympe SDK to execute the commands listed in Table 1.

### Swarm Coordinator

The Swarm Coordinator is responsible for selecting and maintaining a lead drone, monitoring the status of each drone in the swarm, maintaining the swarm coverage state, and executing the area division and path planning algorithms. This subsystem handles most inbound messages and generates most outbound messages. The responsibilities of the subsystem make it a key component to the distributed and autonomous aspects of the system, as it enables the swarm to respond to changes without the need for operator interference. For example, if a drone needs to return home due to low battery, the Swarm Coordinator will inform the other drones of this, resulting in the lead drone re-allocating the returning drone’s remaining points.

## Algorithms

### Guaranteed Delivery

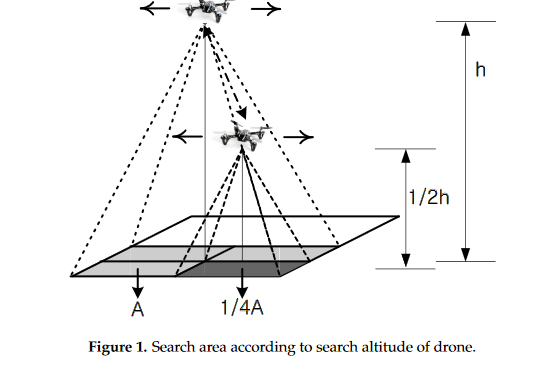
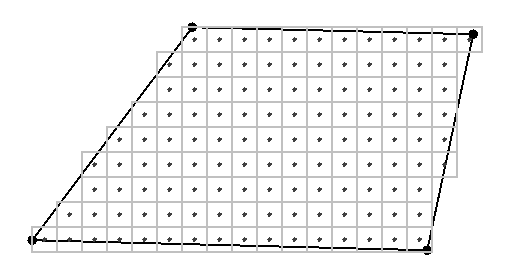


The guaranteed delivery algorithm ensures that external messages are delivered. This is necessary due to the inherently unreliable communication between drones.

The flow of this algorithm is rather simple: when a message is received, an acknowledgement message is returned. Each message will be continuously re-sent after some delay until an acknowledgment message is received. However, acknowledgement messages are not re-sent unless the same message is received again.

### Area Division

The search area polygon is first approximated by grid cells. The side of these grid cells is determined by the horizontal field of view of the lead drone’s camera and the minimum operating altitude for the mission. The center of these grid cells are used as the distributable search points.

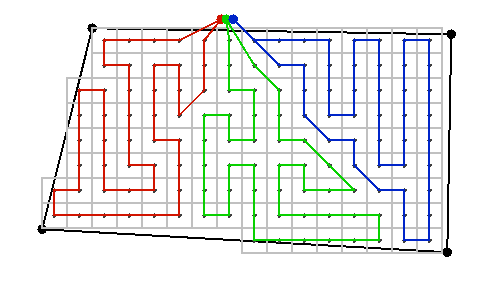
  
 Example of grid cell approximation Demonstration of altitude effect on cell size

These points are then divided using each drone’s current position as inputs to the MILP optimization load balancing algorithm defined in [1]. The battery status of each drone is *not* taken into account by this algorithm.

This division algorithm is repeated in the event that points need to be re-distributed or added to the mission.

### Path Planning

The modified Lin-Kernighan heuristic defined in [1] is used for path planning through a drone’s assigned points. This algorithm takes into account the cost of turns to minimize flight time and energy cost.

  
Example of planned paths in divided area

### Target Detection

The [Okutama-Action](http://okutama-action.org/) Pedestrian model is used for target detection. Streamed video frames are fed to this model through OpenCV and the output detection bounding boxes above a configured confidence threshold are processed by the system.

## References

[1] Modares, J., Ghanei, F., Mastronarde, N., & Dantu, K. (2017). UB-ANC planner: Energy efficient coverage path planning with multiple drones. 2017 IEEE International Conference on Robotics and Automation (ICRA). doi:10.1109/icra.2017.7989732

Tiger Salvage Senior Project

Design Documentation

Revision 1 - 6/1/2022

Michael Padovani

Marcus Chung

Aaron Segal

Bikram Subedi

Bailey Giordano

## Problem Overview

We are building upon the tiger rescue project developed by Tyler Baldwin, Dominic Calabria, and Jacob Wood. The primary goal of this project iteration is to adapt the software to find a submarine in a body of water instead of a human being on land. The software should be able to switch between the two distinct mission types seamlessly. The system must now also be able to support DJI Mavic enterprise 2 drones in addition to the Parrot Anafi drones which were utilized in the first iteration. Lastly, the system must be able to land drones on a boat in a body of water.

## Domain Model

A domain model was derived from the project requirements to provide a high-level conceptual overview of the system. This domain includes:

* Operator: system actor
  + Defines search area and mission objective
  + Deploys drone swarm
* Operators’ terminal: system hardware
  + Device by which operator interacts with the mission software
  + Connects to drone controllers to send drone events over the network
* Target: search target (i.e., human or submarine)
* Drone
  + Collaborates with rest of drone swarm to find mission objective in search area
  + Drone is in a Swarm: 1 to N drones
* Formation: how the drone’s space themselves physically when searching
* Search Algo: determines how to command the drones to most efficient find the target
* Boat: base of operations for the Operator
  + Drones must be able to land here for battery swap outs
  + Drones take off from here to conduct the mission

Diagram

Description automatically generated

## High-Level Architecture

## Subsystems

### Mobile Device Controller

The Mobile device controller handles coordination between the other subsystems and those drones which require a mobile device to coordinate between their hardware.

## Algorithms

## References