Target Tracking using Drone Swarms

Project Members

Benjamin Ireland – Drone Control System & Network Protocol Integration

Finlay Cross – Drone Swarm Lifecycle & Software Porting

Se Hyun Kim – Target & Performance Estimation

Ronniel Padua – Drone Management/Visualisation Application

Raith Fullam - Network Communication Protocol (SENG, Assessed Separately)

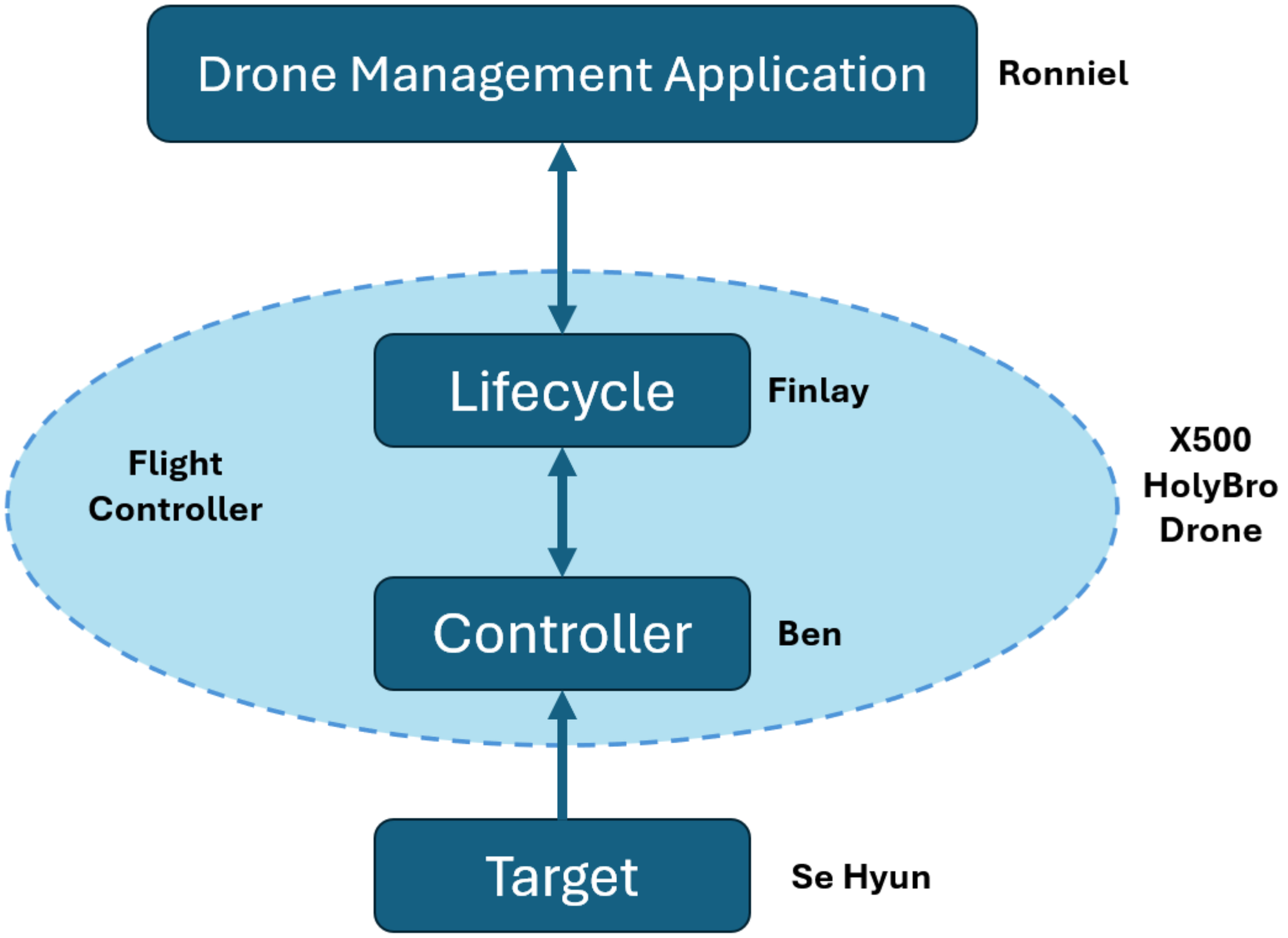
Project Code: E24

Sponsor: Wireless Research Centre

Project Overview

- WRC's Application:
  - Reliable tracking of endangered insects.
- Continued project.
- Subprojects:
  - Flight Controller
  - Control System
  - Drone Swarm Lifecycle
- Target Implementation and Performance Estimation
- Drone Management/Visualisation Application
- *Drone Communication Protocol*

System Hierarchy



Control System & Network Protocol Integration

Sub-project Details

- Design of the drones PID controller.
- How the drone swarms are configured.
- Network Protocol Integration.

```cpp
/*
 *  Publishes the GPS position of the reference point.
 */
void ReferencePoint::publish_pos()
{
    px4_msgs::msg::VehicleGlobalPosition msg{};

    msg.timestamp = this->get_clock()->now().nanoseconds() / 1000;
    msg.lat = global_rp_pos_.get_latitude_deg();
    msg.lon = global_rp_pos_.get_longitude_deg();
    msg.alt = global_rp_pos_.get_altitude();
    global_ref_point_pos_pub_->publish(msg);
}

/*
 *  Initialises the start position of the reference point ffrom the first active drone.
 *
 *  drone_pos:          The drones GPS position.
 *  drone_constraint:   The drones desired position from the reference point.
 */
GPSCoord ReferencePoint::set_rp(GPSCoord drone_pos, Constraints drone_constraint)
{
```

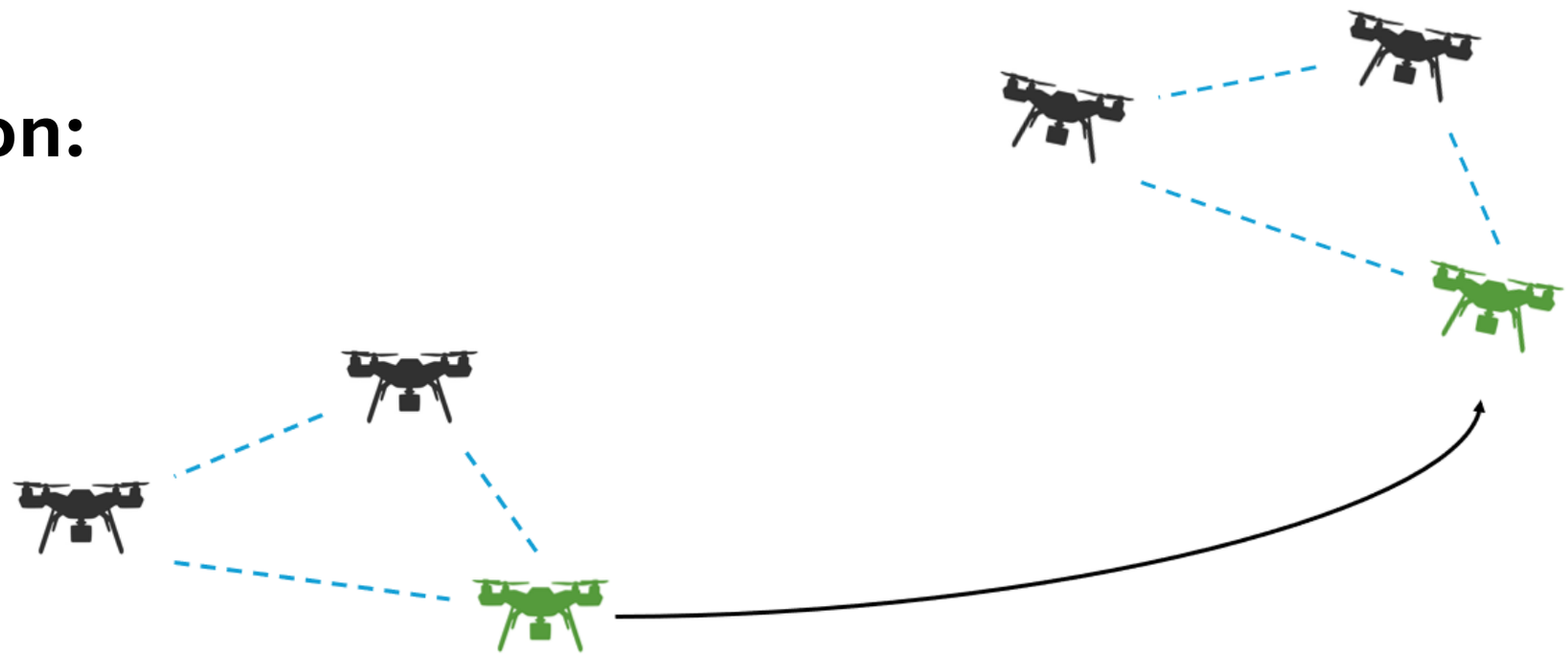
Design Goals

- Provide reliable target tracking capability.
- Address performance and scalability issues.

Swarm Design

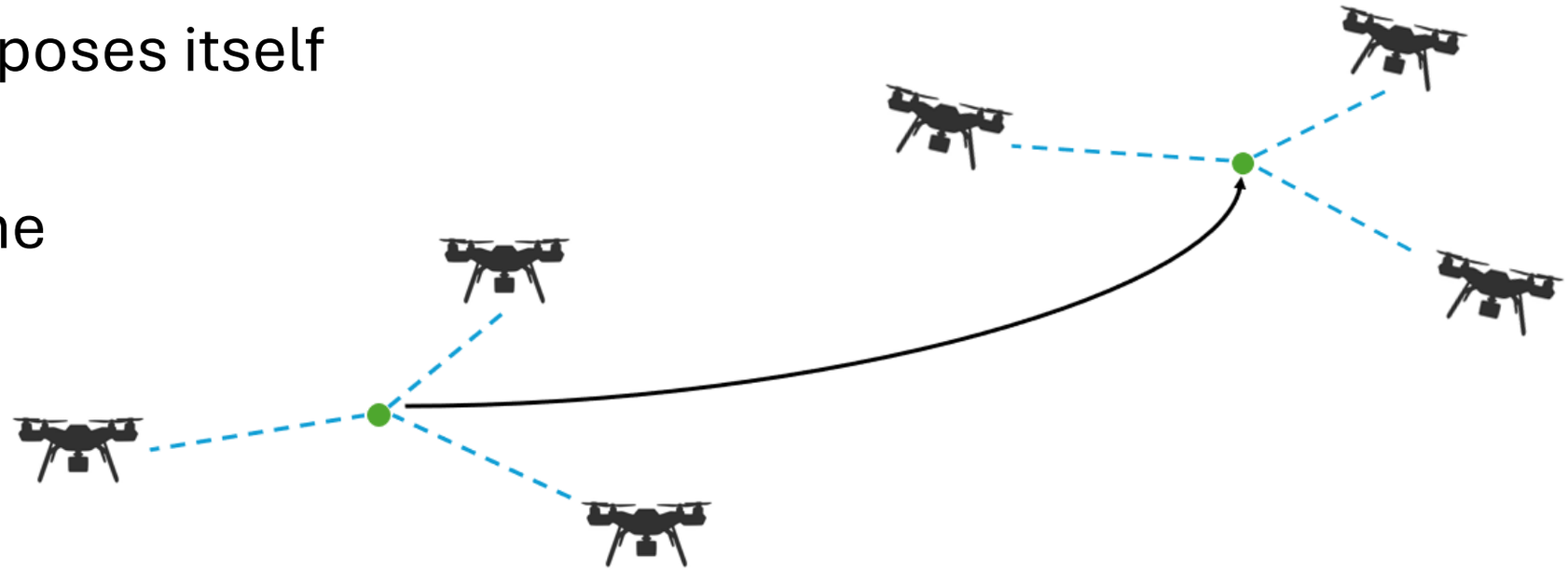
Previous Implementation:

- Leader follower structure.
- Predefined paths.
- Limited scalability

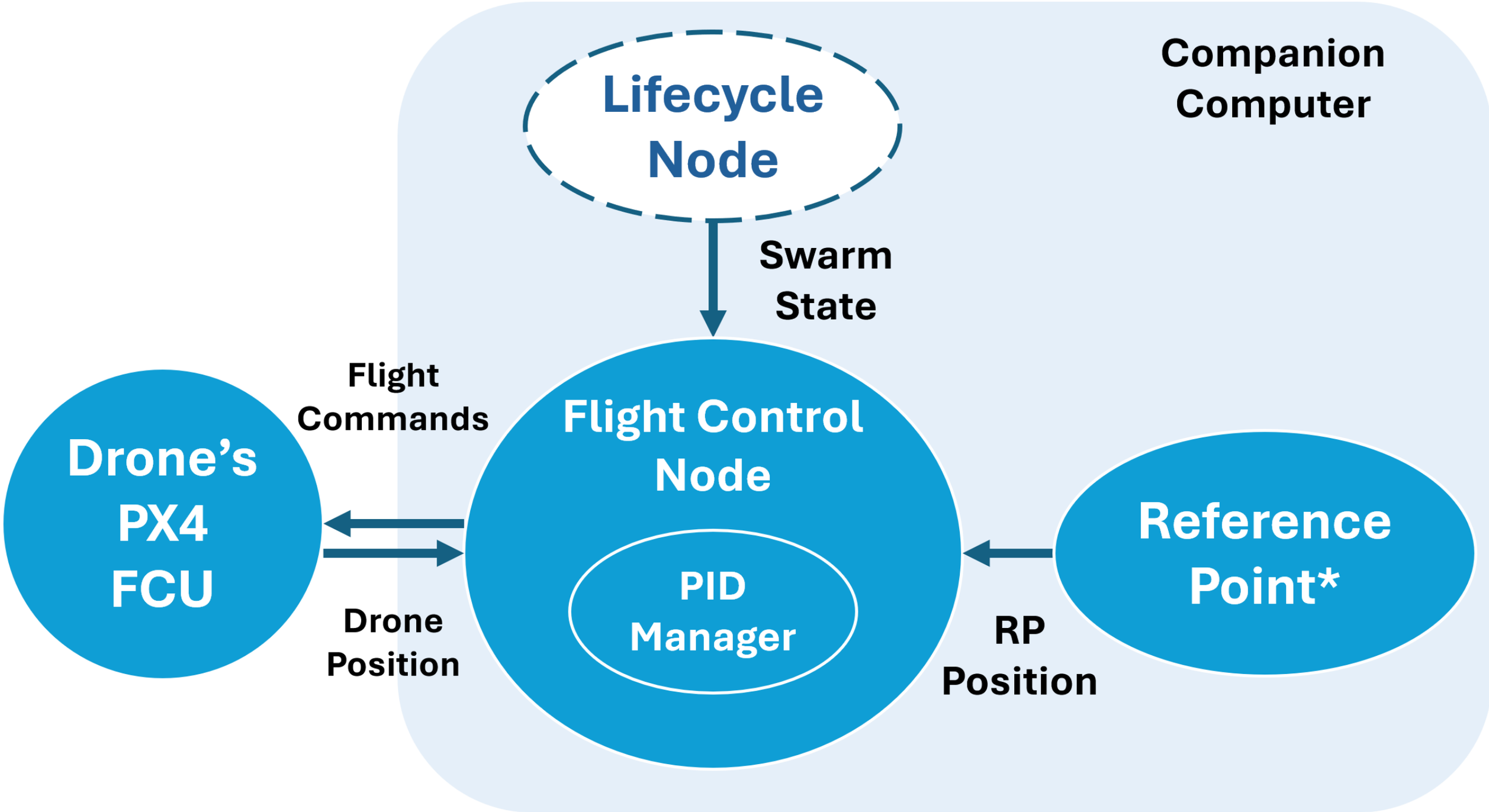


Proposed Solution:

- Maintain position to reference point.
- Reference point superimposes itself over target.
- Simplifies individual drone computation.
- Target can vary.



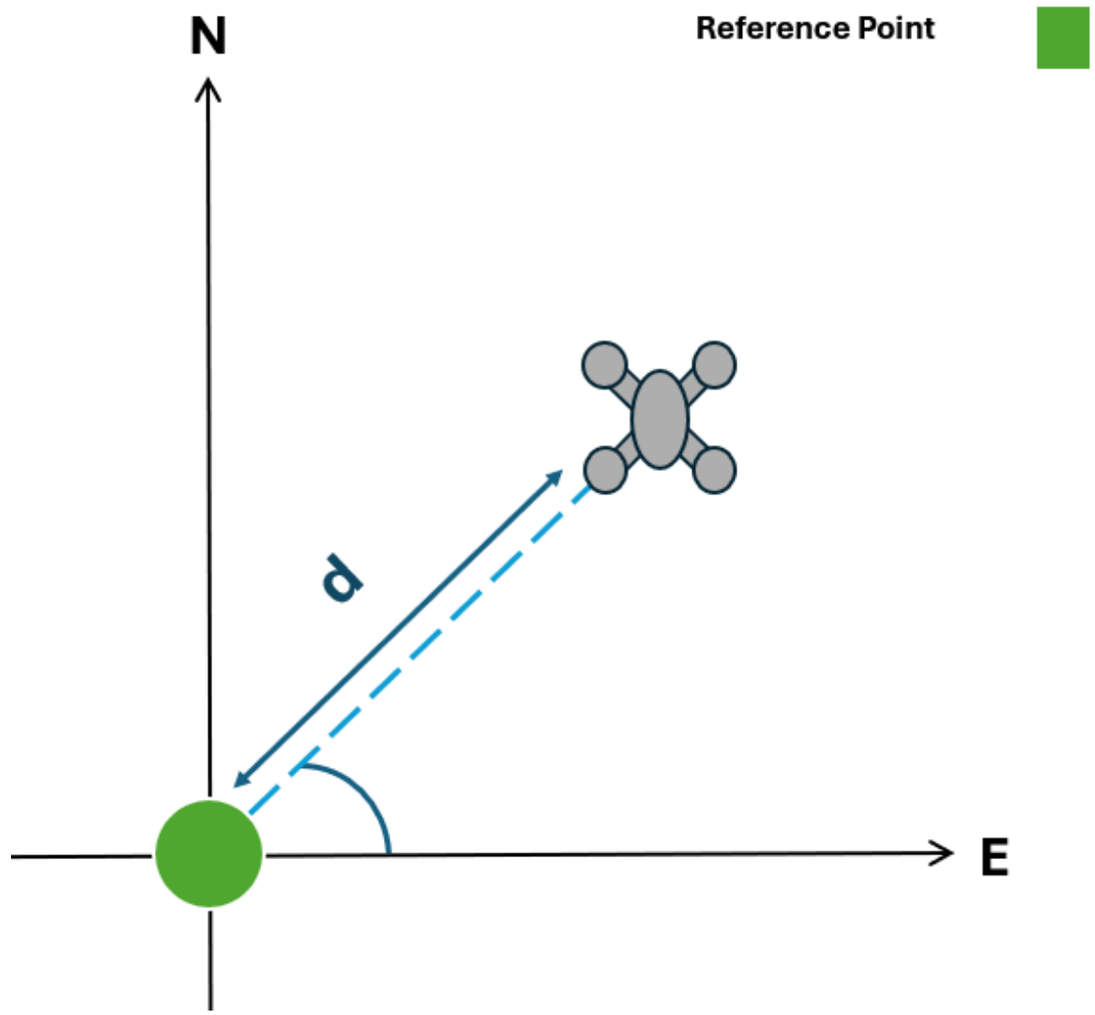
Software Implementation



Swarm Configuration & PID Manager

Individual Configuration

- Position, angle, and height displacement constraints are given with respect to the reference point.
- Drones are unaware of the configurations of their neighbour's.
- Benefits and Risks



PID Management

- Three PID control loops manage the drone's position, height, and yaw.
- PX4 flight control unit expects flight commands as velocities in x, y, z and yaw speed.
- PID manager supplies the FCU with an overall correction vector, based on the output of each control loop.

Progress & Goals

Control System	
Software Design	✓
Software Development	✓
Simulation & Field Testing	☜
Network Integration	
Protocol Integration	–
Testing	–

Software Implementation:

- At present the swarm is being tested in the Gazebo Simulation software to identify software issues and tune the PID controller.

Next Steps:

- Aiming to begin field testing within the coming weeks.
- Development has just begun on integrating the controller with the drone communication protocol.
- Aim to switch to an ad-hoc network between ground stations and drones.
- On target?

Software Porting

Previous

Current

Python

C++

Language

ROS1

Planned (2025) Deprecation

Long Term Support (2029)

ROS2

Robot Software

Primary support dropped

MavROS

µXRCE -DDS

Primary support

PX4 FCU Link

Implemented Software Structure

Drone Internal

Lifecycle

ROS2 Subscriber callback events

Responsible for:

- FSM state
- PX4 FCU state
- PX4 and drone Status

Listens to:

- PX4
- User Input
- Controller

Controller

Finite State Machine

External

User Input

User Interaction

Responsible for:

- Interpreting User commands
- Publishing command message

Talks to:

- Lifecycle

Target Node

Single Thread Target Interaction

Lifecycle Implementation

Sub-Classes: Dependencies of Lifecycle

State
- Name
- Valid Transitions
- Enter & Exit Functions

Co-op
- Boolean value
- All drone values

FCU
- PX4 Information
- PX4 Publishers
- PX4 Subscribers

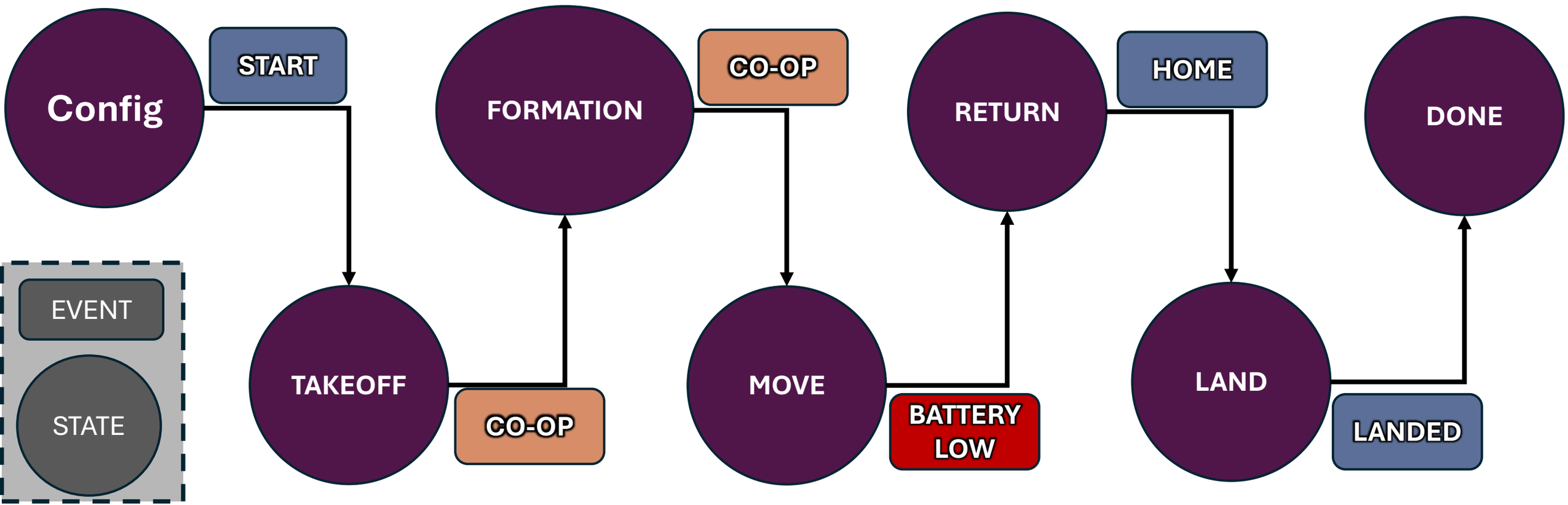
Dropout
- Other drone alive signals

LIFECYCLE

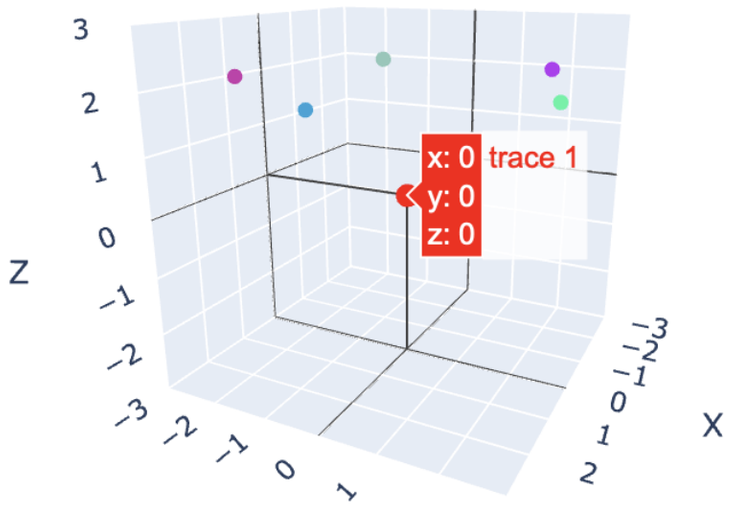
Friend-Classes: Self contained

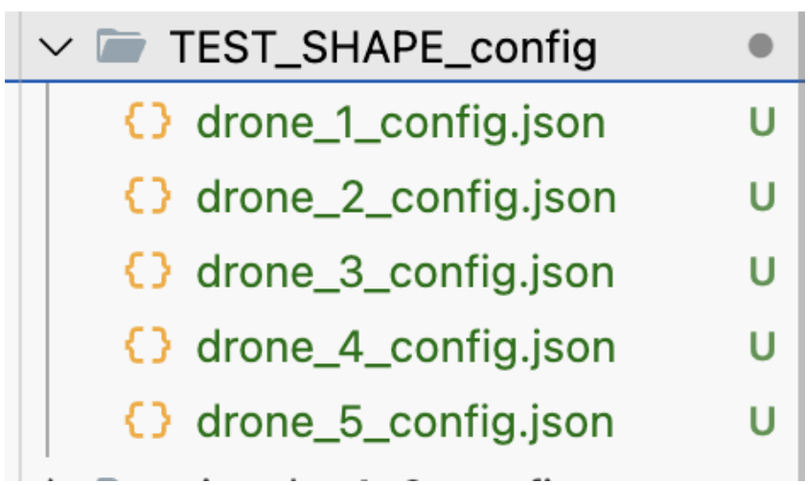
Input
Defines user input subscribers and associated actions

Action-Map
Reads states from config file. Defines enter and exit functions



Configuration App





Dash App

Proof of concept to

Quickly create new formation

Reference Point
- Origin {0,0,0}

Add Node
- x, y, z control

Create polygon
- Number of points
- Radius
- Height

Save
- Save Name

Progress & Goals

GOALS

Finlay

Ben

Review and port existing trajectory-following algorithms

Strong separation of trajectory and formation activities

Develop formation algorithms

Testing in Simulation

Implement state estimation for target

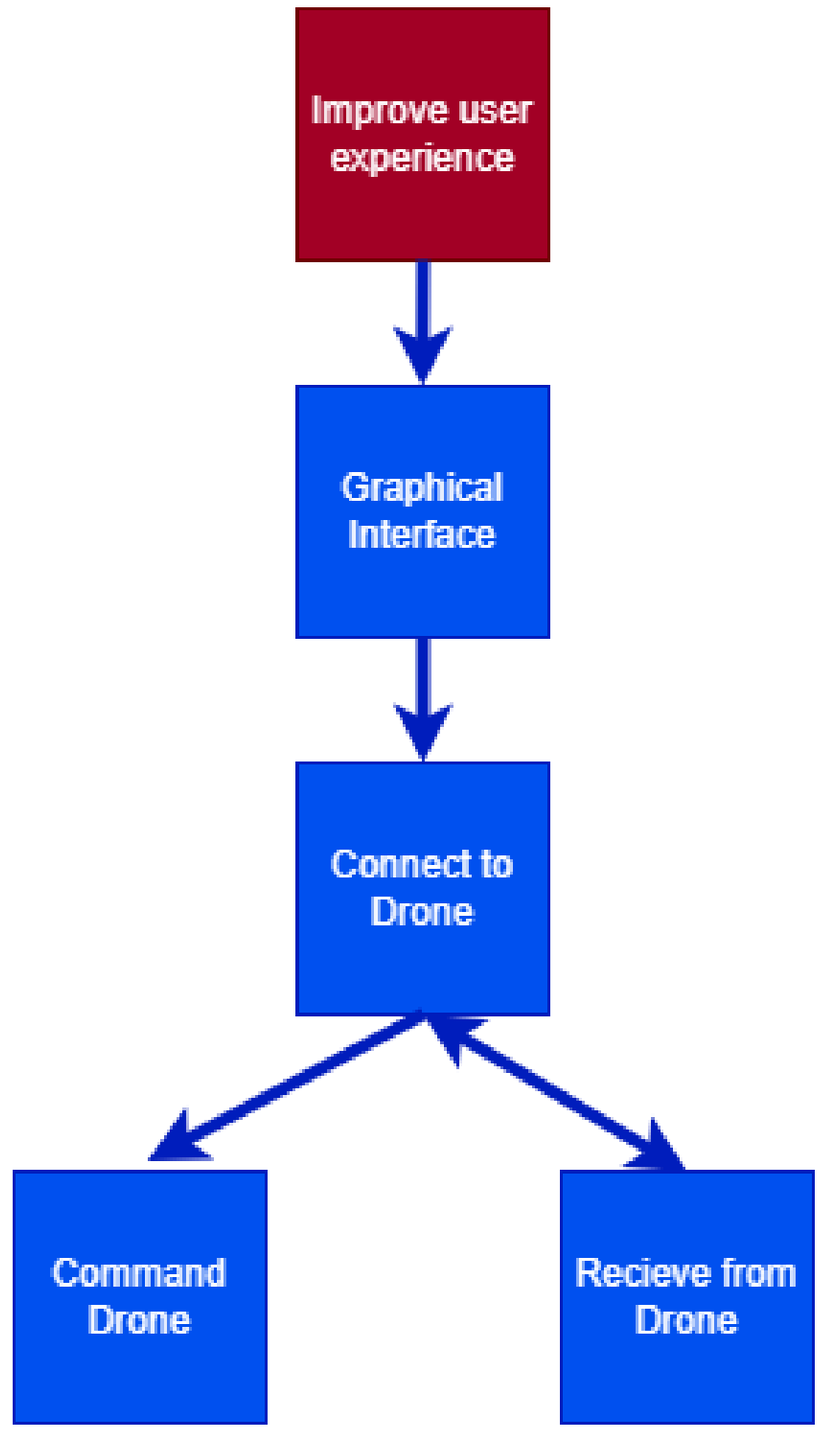
Testing in the field

On Track?

Today

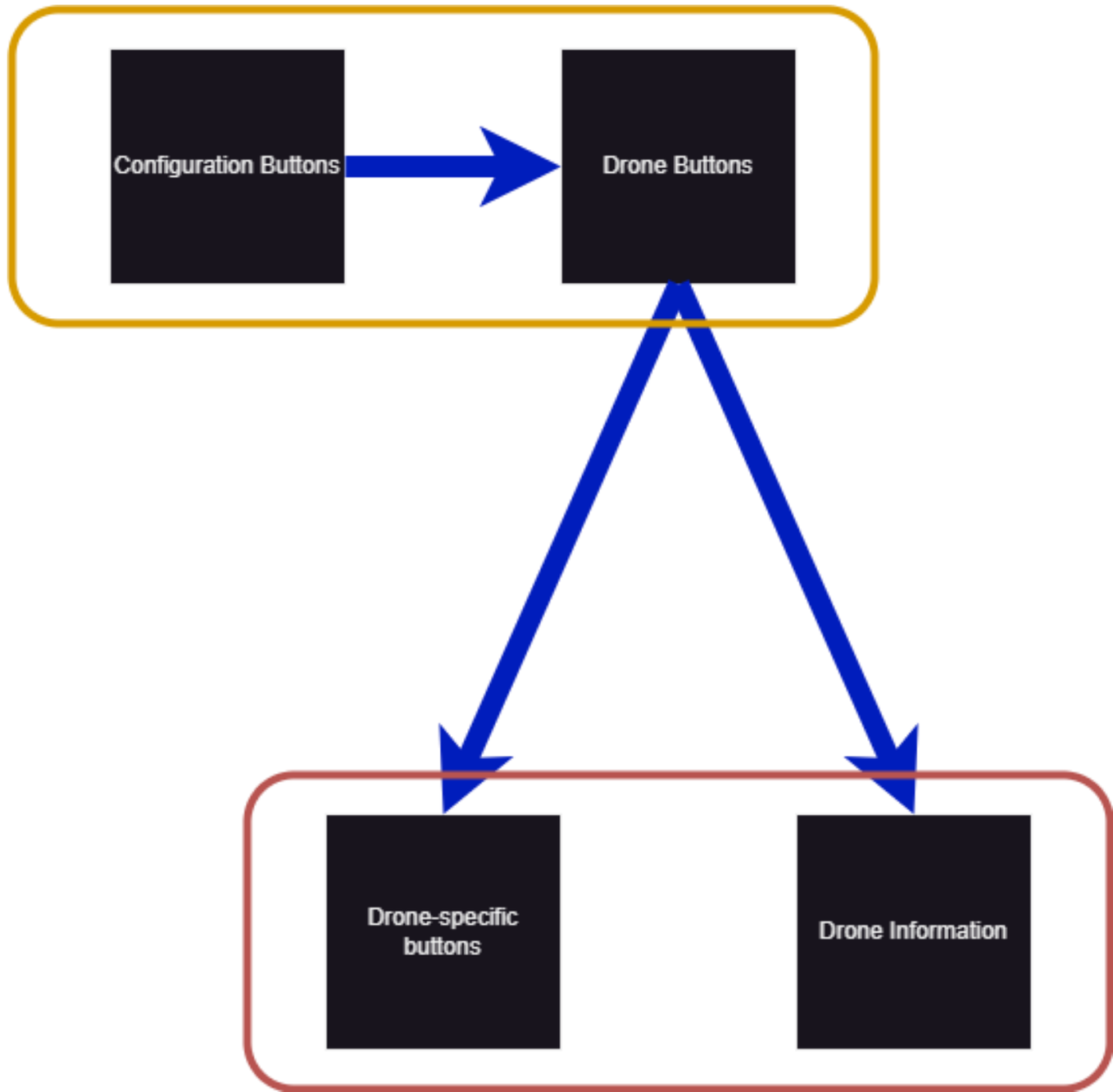
Purpose of the GUI

- Currently:
  - All text-based
  - Slow and error prone
  - Hard to visualize
- Want:
  - A better user experience

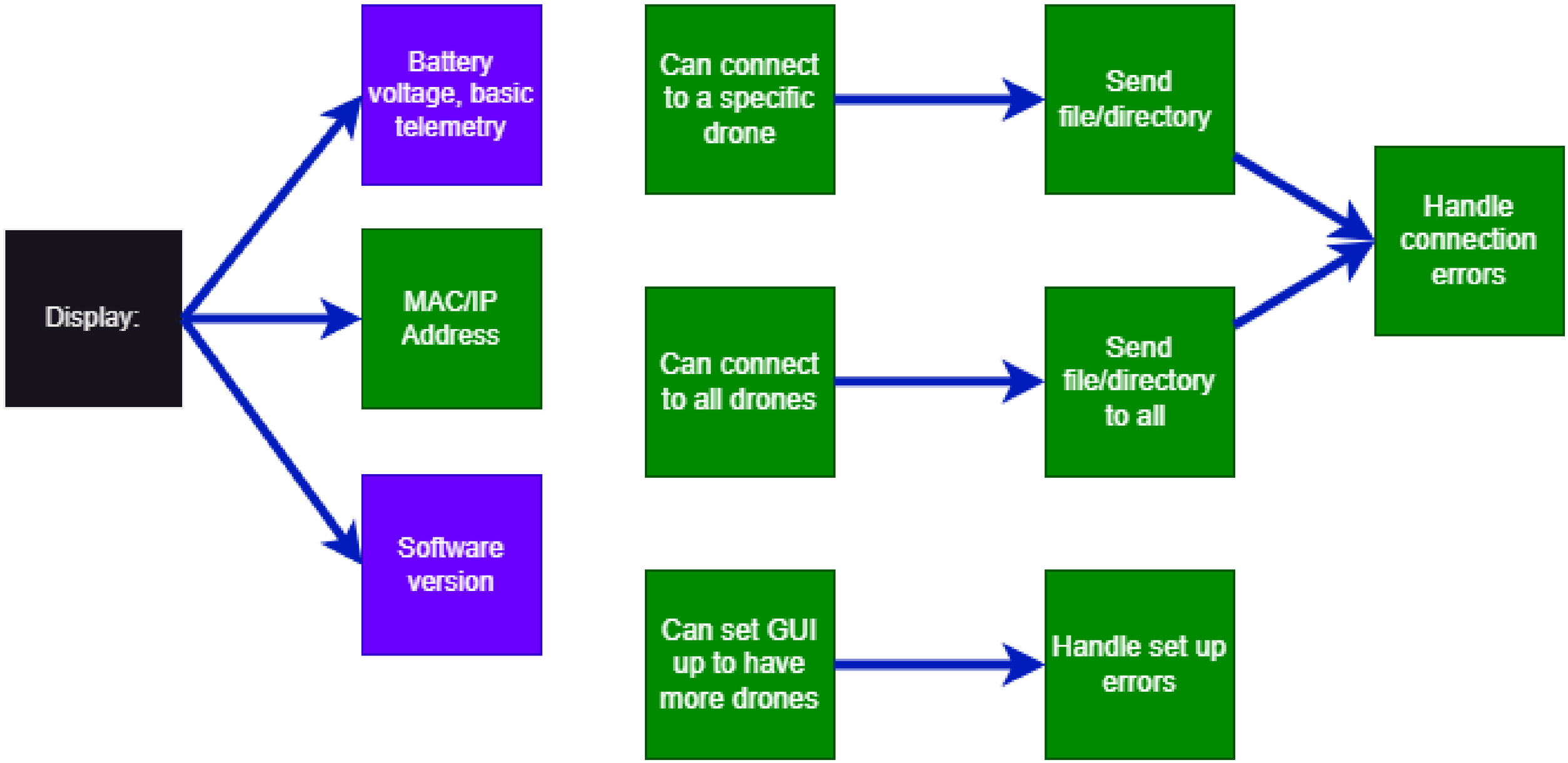


GUI Overview

- Configuration/Drone Buttons
- Command/Control all drones
- Or check a specific drone and control it

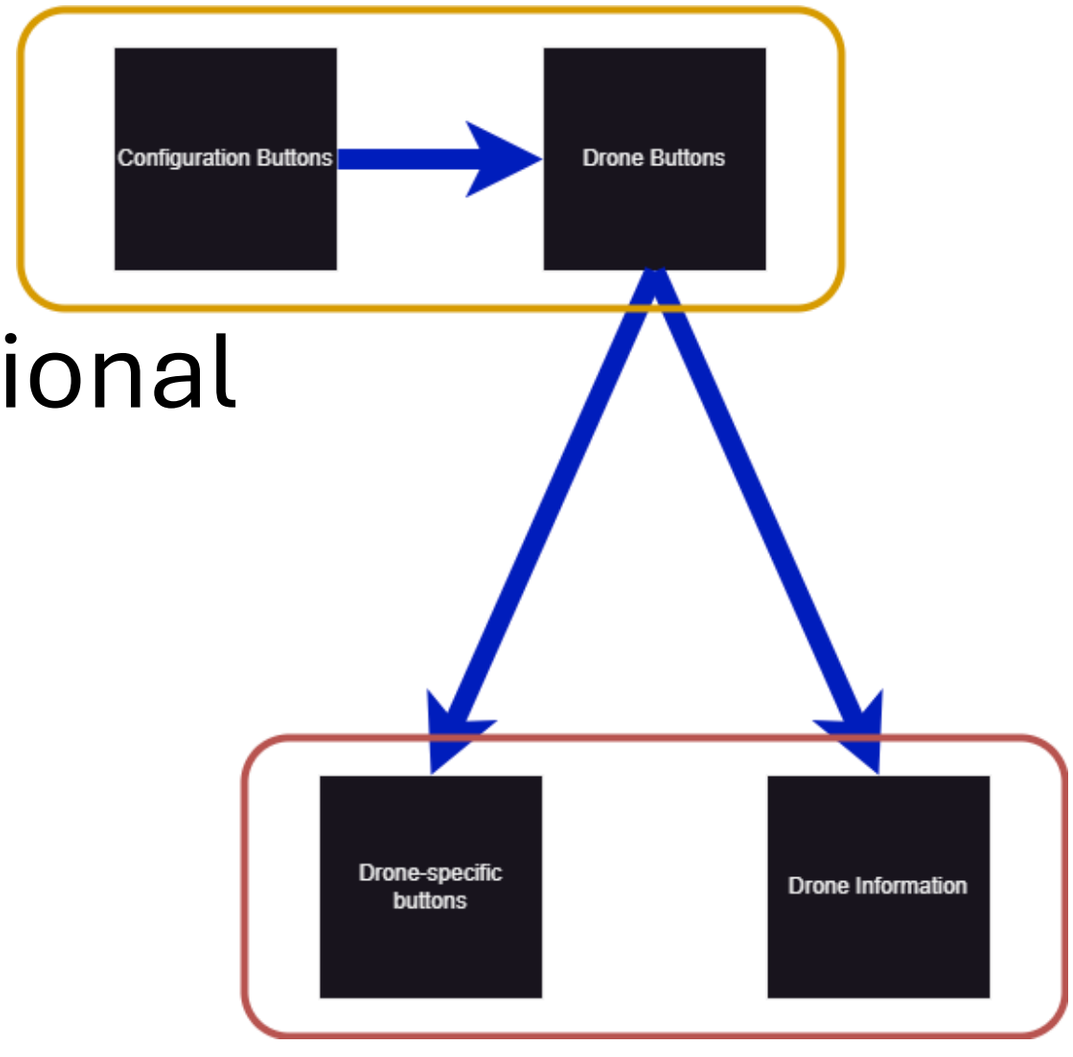


Progress so far



Key Design Choices

- Qt GUI vs Tkinter
  - C++ vs Python
  - Tkinter is simpler
- Design Architecture
  - Data flow is one-directional



Conclusion and Goals

Goals	Current Achievements
Communicate with control system to receive telemetry	**Currently displays important information**
Integrate Finlay's GUI to improve set up	**Can connect to drones securely and handle connection issues**
Test on actual drones and debug issues	**GUI can be set up to include more drones**

Expectation: **Project will be completed on time**