RED Delivery System

Software Subsystem Requirements

Andrea Swanson

1. Requirements
   1. **Subsystem**
      1. **Mission Phases:**

* Initialization
* SynchronizedTargetPursuit
* DeployedTargetPursuit
* LoadDelivery
* DeployedDockingPursuit
* ApproximateDocking
* PreciseDocking
* Docked
* DockingErrorHandling
* SynchronizedHoming

**1.1.2 Mothership Manual Takeover**: The system shall have the ability for a manual takeover, where someone overseeing the ground station can send a command to the subsystem in the air. This way the Raspberry Pis are no longer in control and the mission planner controls on the ground are the only commands being sent to the Pixhawk flight controller that is controlling the mothership.

**1.1.3 Drone Manual Takeover**: The system shall have the ability for a manual takeover, where someone overseeing the ground station can send a command to the subsystem in the air. This way the Raspberry Pis are no longer in control and the mission planner controls on the ground are the only commands being sent to the Pixhawk flight controller that is controlling the drone.

**1.1.4 Mothership Contingency Landing**: The system shall have the ability for someone on the ground to call for a contingency landing for if that if safety or FAA compliance is at risk, the mothership is instructed to return home without a successful docking of the drone to the mothership.

**1.1.5 Drone Contingency Landing**: The system shall have the ability for someone on the ground to call for a contingency landing for if that if safety or FAA compliance is at risk, the drone is instructed to descend to the ground whether the payload was delivered or not.

**1.1.6 Version** **Control**: RED custom processing and flight commands shall be controlled over in a Github Repository for collaboration, revision tracking, and feature implementation.

**2**.**2** **Flight** **Commands**

**2.2.1 Synchronized Deployment**: The system shall have the ability to direct the mothership to carry the drone over an approximate area of the target location and start the drone in order for it to be deployed in a state of descent to target.

**2.2.2 Drone Deployment**: The system shall have the ability to direct the deployed drone to the target location of delivery and direct the mothership to circle above the drone until precise docking phase is initiated.

**2.2.3 Load Delivery**: The system shall have the ability to tell the drone to release the payload at the target location.

**2.2.4 Approximate** **Docking**: The system shall have the ability to direct the drone back to the location it was deployed from at a lower altitude.

**2.2.5 Precise Docking**: The system shall have the ability to tell the mothership to begin flying in a straight direction toward the location that the system was originally deployed from. The drone shall have the ability to use camera imaging of LEDs to recognize the precise location that it must move upward into for mechanical docking.

**2.2.5 Docking Success:** The system shall have the ability to let both aerial vehicles that the drone has successfully docked and turn off the drone rotors.

**2.2.6 Docking Error**: The system shall have the ability to recognize an error state and circle back to an appropriate position to reattempt docking in the mechanical docking mechanism.

**2.2.7 Homing**: The system shall have the ability to direct the mothership back to the original takeoff location for safe landing of both aerial vehicles.

**3.3 Communications**

**3.3.1 GPS Data Exchange:** The system shall have the ability to exchange GPS data between the Raspberry Pi’s on each UAV.

**3.3.2 Unified Mission State:** The system shall have the ability to have both UAV’s in the same state based on the GPS data and sensor data.