**Graduate Projects**

University of Colorado at Boulder

Aerospace Engineering Sciences

ASEN 5018/6028 –Fall 2015

|  |
| --- |
| **FlyNet**  **Flight Controls Subsystem Summary/Continuity Document** |

|  |  |  |
| --- | --- | --- |
| **Approvers List** | | |
|  | Title | Name |
| Prepared By | Flight Controls Team | Drew Ellison, Austin Anderson |
|  |  |  |
| Approved By | Flight Controls Lead | Austin Anderson |
| Approved By | Systems Engineer | Austin Anderson |
| Approved By | Project Manager | Drew Ellison |

**1: Introduction & Summary**

The purpose of the flight controls subsystem is to provide a reliable platform for indoor flight, given a desired position or trajectory and an estimate of the current position.

The flight control system’s current concept of operations is to leverage position controllers on the Pixhawk autopilot consumer product. The majority of work for this team consists of developing the interfaces with this autopilot and with the planning and perception subsystems, which will provide desired positions and position estimates, respectively.

# **2: Semester Report**

## 2.1: Objectives and Tasks List

**Completed**:

1. Interfacing with VICON for position feedback
2. Autonomous take off
3. Station keeping with VICON
4. Waypoint tracking
5. Yaw set point commands

**Incomplete**:

1. Tune controller gains
2. Interface with RTAB SLAM solution for position feedback
3. Receive waypoints autonomously generated from planning subsystem
4. Obstacle avoidance implementation and testing

## 2.2: Issues

What problems prevented you from being able to complete the tasks above?

1. The SLAM (simultaneous localization and mapping) aspect of the Perception subsystem is not developed enough and does not provide a stable enough position estimate in order to reliably use for feedback in the position controllers. This was expected since both subsystems were initialized during this semester.
2. A predefined list of waypoints was used to evaluate the performance of waypoint tracking, however, we have yet to close the control loop with the planning subsystem, which shall provide waypoint lists autonomously.
3. The sensor suite was used exclusively by the perception team during this semester, so no time with the sensors has been available in order to develop obstacle avoidance algorithms.

## 2.3: Lessons Learned

The following lessons learned have been mainly derived from the tasks there *were* completed:

1. Extensive research and reading of autopilot documentation for interfaces with onboard controllers is highly recommended before coding. The team lost about a month and a half trying to develop in-house interfaces when the interfaces that were needed already existed.
2. Contact with the manufacturer of autopilot is extremely useful. Again, the interfaces with autopilot controllers moved much smoother once the manufacturer was contacted and provided information on how the autopilot accepts position commands and estimates.
3. Communication between subsystem leads and management is key. The original leads of the flight controls team did not express the difficulty the subteam was facing with the flight control problem, which led to weeks of wasted engineering effort on designing interfaces that already existed.

## 2.4: Procedures

Table 2.: Software list

|  |  |  |
| --- | --- | --- |
| Program Name | Version | Purpose |
| PX4 Firmware | Hash: 2a1b1fe11db3406ff9e6027f858c7a994adee15d | Run autopilot firmware on Pixhawk. Includes estimators and all controllers |
| MAVROS | Github Commit : bf04343 (Nov 2nd 2015) | Main communication interface with PX4 Firmware from supervisory computer |

Table .2 : Other Useful Links

|  |  |
| --- | --- |
| Page Description | Link |
| Pixhawk Developer page | [https://pixhawk.org/dev/start](https://pixhawk.org/dev/start" \t "_blank) |
| MC Flight Controls | [https://pixhawk.org/dev/multirotor/start](https://pixhawk.org/dev/multirotor/start" \t "_blank) |
| PX4 ROS Links | [https://pixhawk.org/dev/ros/start](https://pixhawk.org/dev/ros/start" \t "_blank) |
| PX4 Github | [https://github.com/PX4/Firmware](https://github.com/PX4/Firmware" \t "_blank) |
| PX4 Source Code | [https://github.com/PX4/Firmware/tree/master/src/modules](https://github.com/PX4/Firmware/tree/master/src/modules" \t "_blank) |
| PX4 Extra Developer Page | [http://dev.px4.iol](http://dev.px4.iol/" \t "_blank) |
| MAVROS Wiki | [http://wiki.ros.org/mavros](http://wiki.ros.org/mavros" \t "_blank) |
| MAVROS\_extras | [http://wiki.ros.org/mavros\_extras](http://wiki.ros.org/mavros_extras" \t "_blank) |
| MAVROS Github | [https://github.com/mavlink/mavros](https://github.com/mavlink/mavros" \t "_blank) |
| MAVROS Issues | [https://github.com/mavlink/mavros/issues](https://github.com/mavlink/mavros/issues" \t "_blank) |

# **3: Nsext Semester/Future Expectations**

## 3.1: Prioritized List of Tasks and Objectives

1. Close the loop with planning subsystem
2. Implement obstacle avoidance algorithms
3. Evaluate performance of planning subsystem with obstacle avoidance using VICON
4. Implement search pattern criteria (movements and camera scanning maneuvers)
5. Interface with SLAM algorithm for position estimate feedback

## 3.2: Starting Points

All technical documents are located in the following GitHub repository: [www.github.com/dme722/FlyNet](http://www.github.com/dme722/FlyNet)

All flight code can be found in the following BitBucket repository:

<https://bitbucket.org/cuflynet/>

1. The planning subsystem is currently under construction, and is implemented in MATLAB files in the GitHub repository in the following directory: “/FlyNet/Techincal/pathPlanner”. The planning subsystem is still in the process of implementing these algorithms in flight compatible version (in C/C++ or Python). Please coordinate with the planning team on how to pass waypoint information from the planning subsystem to autopilot controllers.
2. The obstacle avoidance algorithms are not yet implemented in any form, however, they will be implemented using obstacle information from the DJI Guidance system. The documentation for this sensor can be found here: <http://www.sekidorc.com/pdf/Guidance_UserManual_en_v1.2_150803.pdf>  
   In order to pull data from the Guidance sensor, a Guidance ROS node will be used, the documentation and code for which can be found in the following repository:  
   <https://bitbucket.org/cuflynet/guidance-sdk-ros>  
   Once data is being pulled into your script using this node, obstacle avoidance algorithms can be implemented by examining the distances to closest obstacles. If an obstacle is within a certain distance threshold, a maneuver can be performed to avoid collision.
3. Once the planning and obstacle avoidance algorithms have been implemented, the flight control code can be found in the BitBucket repository listed above (all nodes that don’t have “Guidance” in the name”). The instructions for running flight code, VICON position publishing, etc. can be found at: <https://github.com/dme722/FlyNet/blob/master/Technical/documentation/howto/flight_test_instructions.txt>
4. Similar to 3, use the same flight instructions. The flight search sequences are defined to be stop every 5 meters and perform a scan, the commands for yaw scans have yet to be implemented, so these scripts should be written, and then flight tested as above. The yaw control gains may need to be tuned.
5. For now, we need to wait for the Perception team to become more mature on it’s SLAM position estimate reliability. However, work does need to be done on creating a ROS node where the position estimate from RTAB Map can be published. See the BitBucket for RTAB Map code.

## 3.3: Improvement, Updates, Verification

1. As the path planning algorithms and SLAM position feedback are incorporated, it is likely that control gains will need to be adjusted, as our current models for SLAM position estimate performance are low fidelity. Our current gain scheme performs well with VICON as the position estimate source, but since VICON has relatively low noise and error, these gains will most likely need to be relaxed once the Perception subsystem is in place for position feedback.
2. Similarly, the thresholds for maneuvering away from obstacles will need to be adjusted as the algorithms are incorporated.
3. An autonomous landing feature still needs to be implemented, as this could likely save the flight team time and effort in creating landing gear, as the landing gear frequently breaks during manual takeover due to hard landings.