Project 1

Due: Project 1 will be demonstrated in lab on Thursday October 24. The report for Phase 1 of the project will be due at 5 pm on Friday October 25.

The goal of this project is for students to implement a new capability on the Pixhawk autopilot using the ArduPlane firmware. In particular, students must implement a guidance-layer path-following algorithm. Details of the specific algorithm to be implemented and details for how it is implemented are up to the students. RECUV has implemented a version of the assignment as described in the FixedWing.pdf document on the course web site.

Teams

The project will be carried out in teams. There can be three total teams for this assignment. With 16 registered students, that means two teams of 5 students and one team of 6. Each team submits one version of the report assignment (detailed below). Teams are encouraged to collaborate with one another and to share insights.

Task

Student teams are to implement a **path-following** capability. Recall, path following means the aircraft must be on a specified path and follow it in a given direction. There is no explicit requirement to be at a given point at a given time (that is the definition of a *trajectory* or *4-D trajectory* tracking).

A desired path will be given to teams as a sequential list of points as a .csv file.

- Each point will be described by its inertial position in a local N-E-D frame.
- Units of the positions are in meters relative to some origin (not necessarily the center of the desired pattern).
- The points will define a closed star pattern (unique mapping to unit circle at unit height, no self-crossing) that is not easily described by an analytical function.
- The desired path is the concatenation of the straight line segments connecting each point.
- The final point in the list connects to the first point in the list.
- The ordering of the points determines the direction the aircraft is to travel around the pattern.
- Points will be spaced relatively close together, i.e. much closer than typical distances between “waypoints”. Thus, good performance is not expected by using the path list as a waypoint plan for the autopilot.

Hardware

RECUV will provide one or two Pixhawk autopilots for use for this project. The main effort for the project will be implemented using the software in the loop (SITL) capability of the ArduPlane firmware. RECUV uses the Odroid single board computer as a supervisory computer for our aircraft. For the purposes of this assignment you may use any computer as your supervisory computer during the SITL phase of the project. RECUV does not have spare Odroid computers to provide to teams.

Software Architecture

Students will develop code to implement their approach on an aircraft using the ArduPlane firmware with the Pixhawk autopilot. Students may make changes to the ArduPlane firmware and/or use communication APIs (mavproxy, droneapi, mavlink) to communication with the autopilot from an onboard supervisory computer. The decision on what to modify in firmware and what to modify on the supervisory computer is a design decision of the team.

Safety Features

The ArduPlane system has several safety features that must be maintained with your modifications to the system. First, there must be a way to activate your control algorithm. The desired plan may (should) be hard-coded into your implementation. However, a command must be sent from the ground to the aircraft to initiate your path following control. This command can be given directly from the onboard supervisory computer via a telnet session with the supervisory computer.

All RECUV flight operations include a "lost-comm" behavior. This behavior is initiated in the ArduPlane code but is easily overridden by new capabilities. Thus, your code must be able to recognize when communication is lost with the ground and cause the aircraft to enter into lost-comm behavior. The obvious way this is broken is if your algorithm continues to send control commands from a supervisory computer even after that computer has lost link.

Assistance

RECUV student Kevin Rauhauser completed the project this summer in order to work through the process. He has agreed to hold office hours on Wed from 12-1:30 in the RECUV Fab Lab (ECAE 1B??). Kevin prefers that students email him to let him know they would ike to meet then: Kevin.Rauhauser@colorado.edu.

Assignment

This assignment has two parts. Phase 1 of the assignment is a report due for each team. Phase 2 is flight implementation of one solution. For each phase a report is due.

Phase 1

The report for Phase 1 should describe the following: the specific control algorithm designed by the team, simulation results in Matlab that show expected performance, block diagrams of the software implementation of your algorithms, plots and discussion from software-in-the-loop simulations, final recommendations (successes and failures) based on your experiences. A single report is due for each team at 5 pm on Friday October 25.

Phase 2

The Phase 2 report should discuss flight implementation of the class' chosen code. Describe and discuss the performance of the aircraft. Does the behavior match expectations from the SITL simulations? How could the implementation be improved?

Due date for the Phase 2 report will be given once flight data is provided to the teams.

Additional Notes

Here is information for downloading the code:
http://dev.ardupilot.com/wiki/where-to-get-the-code/

Here is information for editing the code in eclipse:
http://dev.ardupilot.com/wiki/editing-the-code-with-eclipse/

Here is information or compiling it for the Pixhawk:
http://dev.ardupilot.com/wiki/building-px4-with-make/

Here is the official documentation for SITL:
http://dev.ardupilot.com/wiki/setting-up-sitl-on-linux/
http://copter.ardupilot.com/wiki/common-sitl-software-in-the-loop-simulator/

To start SITL:

'cd ~/ardrupilot/Tools/autotest/'
'./fg_plane_view.sh'
Wait until Flight Gear is fully loaded and the aircraft is sitting on the ground
'sim_vehicle.sh -v ArduPlane'
The '-w' option resets the parameters if needed
With the '-L' flag you can set a location from the "locations.txt" file located in the autotest directory
Run with '-h' to see all of the options