Senior Design
ENG EC 463

Memo

To: Professor Pisano

From: Jaden Cho choj@bu.edu

Brendan Slabe bps123@bu.edu

Christopher Lemus lemusc@bu.edu

Henry Bojanowski henryboj@bu.edu

Stephanie Pellicane pellisdp@bu.edu

Team: Tethered Power Drone

Date: February 26, 2021 11:59PM

Subject: Test Report

1.0 Required Materials

- **Hardware**
  - Computer with Intel Processor Core i5 or higher
- **Software**
  - X-Plane 11
  - Mission Planner
  - Eclipse
  - Ardupilot code base
  - Windows built-in command prompt

2.0 Setup

The setup for the project is to first boot up X-Plane, Mission Planner, and Eclipse. Eclipse holds the code base and allows for editing and building of the executable. The executable is then run through the command prompt with the command “arduplane.exe -Mxplane -s1 --uartA tcp:0” after going into the correct directory. Once this command is run, Mission Planner has to be connected to X-Plane via data network ports. After the connection has been established, the drone must be armed. The simulation can then take a series of commands from a drop down menu in Mission Planner. The result of the command can be seen from both X-Plane and Mission Planner.

3.0 Pre-Testing Setup Procedure

1. Create executable from Eclipse (custom build of ardupilot)
2. Start a new X plane flight
3. Navigate to the bin folder within ardupilot on local machine
4. Run executable from Command Prompt with command ***arduplane.exe -Mxplane -s1 --uartA tcp:0***
5. Establish connection between X-Plane and Mission Planner
6. Arm drone simulation
7. Input command

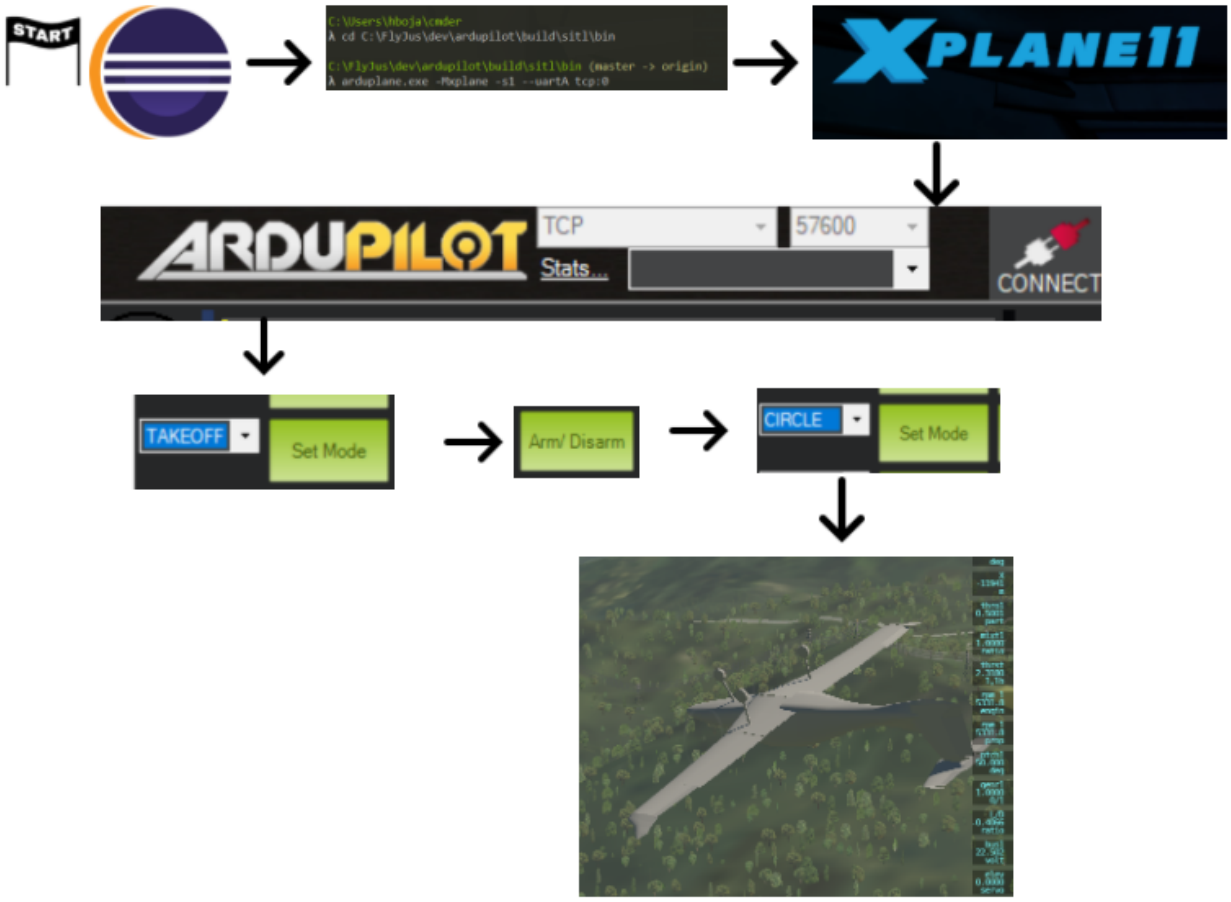

Figure 1: Starting with a custom build of ardupilot in Eclipse, ending with circle flight mode in X Plane/Mission Planner

4.0 Testing Procedure

1. Input Circle command into Mission Planner
2. Drone will begin to do spiral circling with varying pitch and roll
    a. Varying roll is to demonstrate the changing of the radius of the circle
    b. Varying pitch is to demonstrate the changing of the angle of attack
3. X-Plane will output the pitch and roll parameters of the maneuver

5.0 Measurable Criteria

1. Simulation should successfully begin takeoff and reach proper altitude and proper distance from runway without crashes (parameters set before takeoff)
2. Once command is set, drone should immediately begin maneuver
3. As roll increases, circle should get tighter and vice versa
4. As pitch increases, altitude should increase and vice versa

Measurable criteria include altitude changes corresponding to changes in combinations of pitch and roll, as well as aerodynamic drag forces that do occur when the aircraft is flying perpendicular to the wind and upside-down. Another significant feature of this project is to figure out a model to accurately log energy usage during the takeoff phase of the aircraft, since we want to be efficient in how much energy is consumed when traveling from the ground to the boundary layer where the airspeeds will be used to our advantage. All of these data files can be created when running the X-Plane simulation since it offers real time data. Another important aspect will be PID gain tunings and this is important so that our system has the responses we need when we need them and accurate steady state values with the least error and overshoot possible.

6.0 Conclusions

Our starting point at the beginning of this project was the standard circle mode which is a part of the Ardupilot/Arduplane codebase. So far the modifications we have made to the code has shown that we can oscillate the roll angle which tilts the aircraft left and right, as well as control the pitch angle to increase the altitude and even maintain it constant to obtain a vertical circle that spirals. The standard circle mode yields horizontal circles in the air by maintaining a constant roll angle, where the larger the roll angle is the smaller the circle radius becomes. These concepts will be applied - but in reverse - to the vertical circle where ideally we would like to figure out a way to maintain a constant pitch angle with no roll so we can have a stationary vertical circle that we could work from and modify. We will be looking into any exceptions that the EKF code might print just in case it does not like the aircraft exceeding safe, commercial flight parameters that were previously set in the Ardupilot/Arduplane codebase. Looking down the line, once we achieve a stationary vertical circle, we will begin figuring out how we can allow the vertical circle to move in either x-direction (or horizontally).