Progress Review 9

Cole Gulino

Team C / Column Robotics

Teammates: Job Bedford, Erik Sjoberg, Rohan Thakker

ILR # 8

February 25, 2016

Individual Progress

During the last two weeks, I along with the rest of the team was focused on getting the Iris+ to dock.

I started by running commands in order to gather data as shown in Figure 1:

```
$ roslaunch mavros px4.launch # Mavlink topics from pixhawk
$ roslaunch mavros_extras px4flow.launch # Mavlink topics from PX4FLOW
$ rosbag record -a # Record the topics
$ rosrun april_tag april_tag # Run the april tag reading and publishing node
$ rosrun column offb_node # Run the custom offboard control node
```

Figure 1) Commands to Run OFFBOARD Control Mode

The `offb_node` is a custom node written so that we can send commands through mavlink to the Pixhawk flight controller. The code is shown in Figure 2:

```
#include <ros/ros.h>
#include <geometry_msgs/PoseStamped.h>
#include <geometry_msgs/Pose.h>
#include <mavros_msgs/CommandBool.h>
#include <mavros_msgs/SetMode.h>
#include <mavros_msgs/State.h>

bool flag = false;

mavros_msgs::PoseStamped current_pose; // Global variable for current position
void position_cb(const gemoetry_msgs::PoseStamped::ConstPtr& pose){
    current_pose = * pose
}

int main(int argc, char **argv){
    ros::init(argc, argv, "offb_node");
    ros::NodeHandle nh;
    ros::Subscriber local_pos_sub = nh.subscribe<geometry_msgs::PoseStamped>
        ("mavros/local_position/local", 10, position_cb);  // Callback function to get local
pos
    ros::Publisher local_pos_pub = nh.advertise<geometry_msgs::PoseStamped>
        ("mavros/setpoint_position/local", 10);

    geometry_msgs::PoseStamped ref_pose;
    while(ros::ok()){
        if(!flag){ // Set to current position once
            ref_pose = current_pose;
            flag = true;
        }
        local_pos_pub.publish(ref_pose); // Try to maintain original pose
    }
}
```

Figure 2) Code to Maintain Pose

The code that we ended up using was different than this, and we changed the code many times as a group. This was the original code that I wrote to see how well the position control and local position estimates worked.

Figure 3 shows a plot of the position setpoints and position estimates from the Pixhawk flight controller.

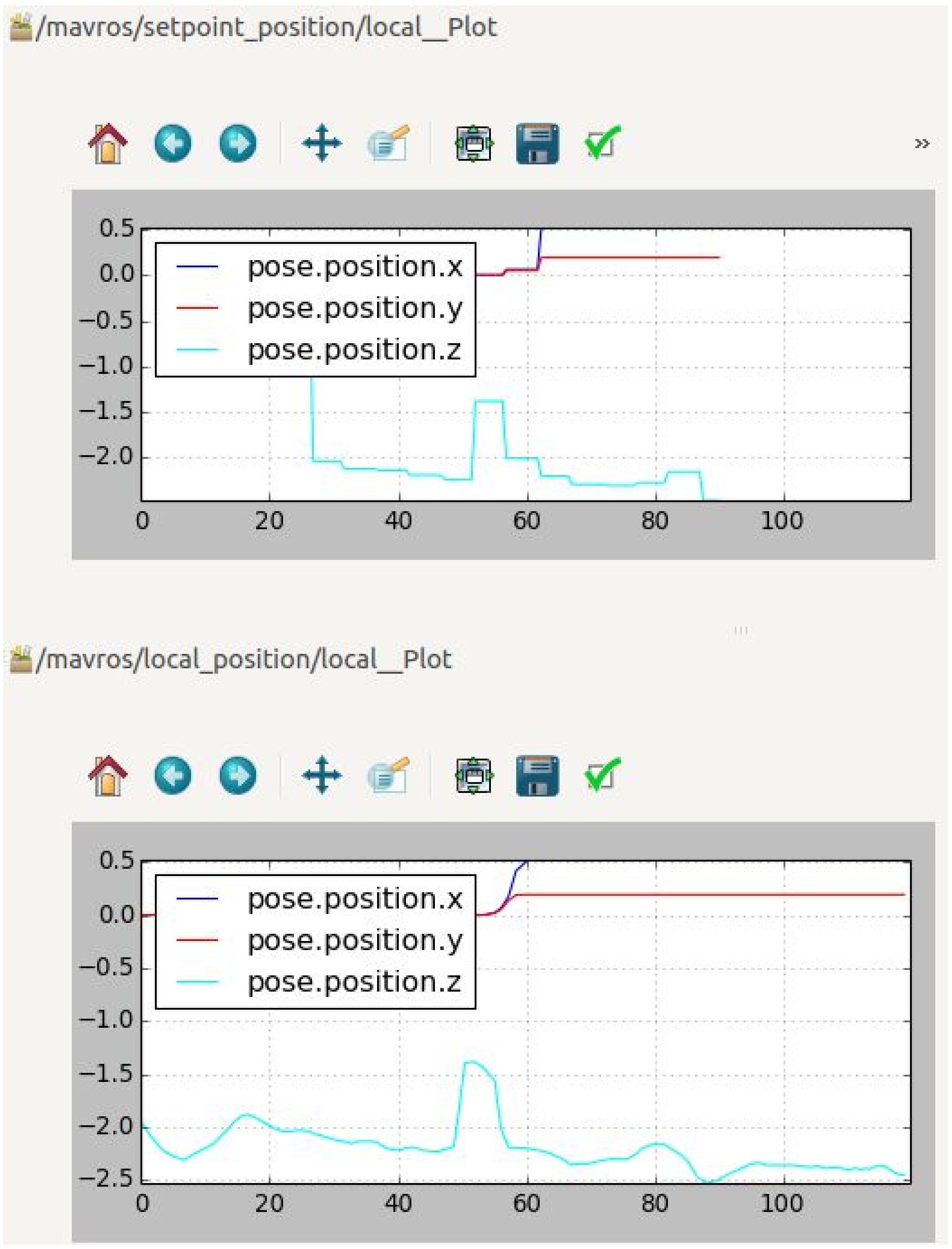


Figure 3) Position Setpoints and Local Position Estimates

The top figure shows the setpoints that we send to the Pixhawk. These plots were a test to determine if we were sending accurate position setpoints. In this plot, the position setpoints were given based on the previous position estimates.

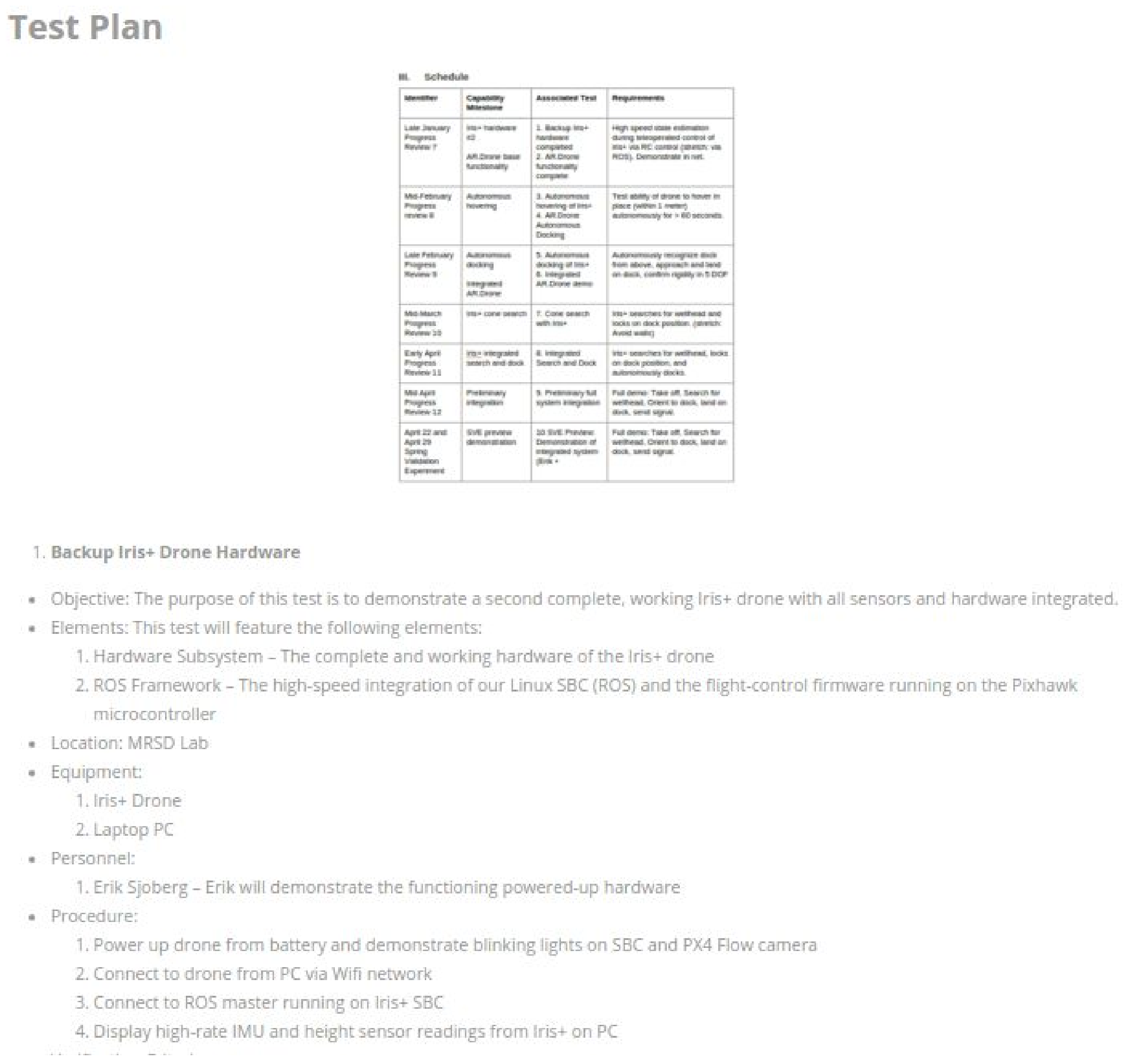
Figure 3 shows the Pixhawk can be given local setpoints correctly.

Doing these tests allowed us to figure out how to send setpoint positions to the Pixhawk in the local frame.

It also gave us an idea on how accurate the Pixhawk's position estimates were. We found that even whenever the quadcopter drifted while holding position, the position estimates were accurate.

I also worked on the website during this PR.

For the website, I put the test plan information in the appropriate section. Figure 4 shows a screenshot of the page.



Test Plan

III. Schedule

Identifier	Capability Milestone	Associated Test	Requirements
Late January Progress Review 7	Iris+ hardware x2 AR.Drone base functionality	1. Backup Iris+ hardware completed 2. AR.Drone functionality complete	High speed state estimation during teleoperated control of Iris+ via RC control (stretch: via ROS). Demonstrate in net.
Mid-February Progress review 8	Autonomous hovering	3. Autonomous hovering of Iris+ 4. AR.Drone Autonomous Docking	Test ability of drone to hover in place (within 1 meter) autonomously for > 60 seconds.
Late February Progress Review 9	Autonomous docking Integrated AR.Drone	5. Autonomous docking of Iris+ 6. Integrated AR.Drone demo	Autonomously recognize dock from above, approach and land on dock, confirm rigidity in 5 DOF
Mid-March Progress Review 10	Iris+ cone search	7. Cone search with Iris+	Iris+ searches for wellhead and locks on dock position. (stretch: Avoid walls)
Early April Progress Review 11	Iris+ integrated search and dock	8. Integrated Search and Dock	Iris+ searches for wellhead, locks on dock position, and autonomously docks.
Mid April Progress Review 12	Preliminary integration	9. Preliminary full system integration	Full demo: Take off, Search for wellhead, Orient to dock, land on dock, send signal.
April 22 and April 29 Spring Validation Experiment	SVE preview demonstration	10 SVE Preview: Demonstration of integrated system (Iris +	Full demo: Take off, Search for wellhead, Orient to dock, land on dock, send signal.

1. **Backup Iris+ Drone Hardware**

- Objective: The purpose of this test is to demonstrate a second complete, working Iris+ drone with all sensors and hardware integrated.
- Elements: This test will feature the following elements:
    1. Hardware Subsystem – The complete and working hardware of the Iris+ drone
    2. ROS Framework – The high-speed integration of our Linux SBC (ROS) and the flight-control firmware running on the Pixhawk microcontroller
- Location: MRSD Lab
- Equipment:
    1. Iris+ Drone
    2. Laptop PC
- Personnel:
    1. Erik Sjoberg – Erik will demonstrate the functioning powered-up hardware
- Procedure:
    1. Power up drone from battery and demonstrate blinking lights on SBC and PX4 Flow camera
    2. Connect to drone from PC via Wifi network
    3. Connect to ROS master running on Iris+ SBC
    4. Display high-rate IMU and height sensor readings from Iris+ on PC

Figure 4) Screenshot of the Test Plan on the Website

Challenges

A major challenge this sprint was understanding the coordinate frames of quadcopter, odometry frame, and the camera.

Figure 5 shows an image of the different coordinate frames we are considering.

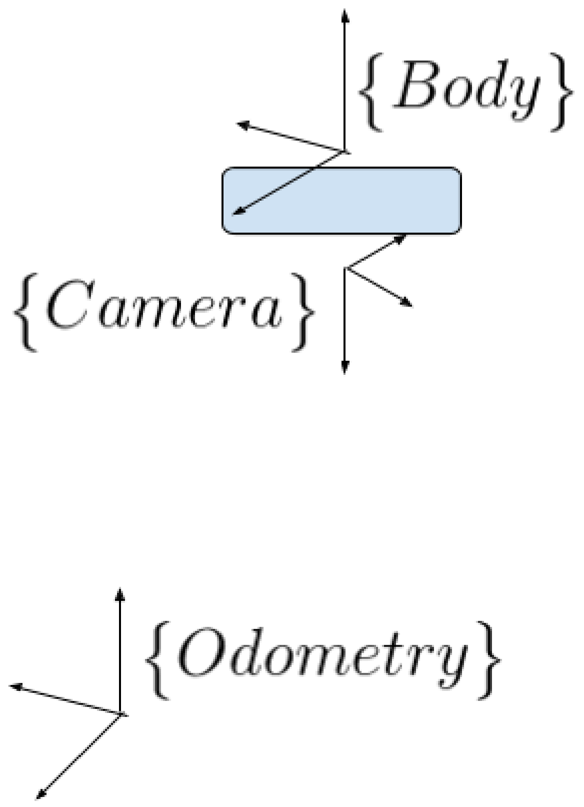


Figure 5) Coordinate Frames of the System

Figure 5 shows the coordinate frames that we are considering. A major challenge that we faced was determining how each of the frames were oriented. The odometry frame is based on where the quadcopter is at the start of its mission. The frame starts off in the same place as the world frame. errors in the state estimation are carried out in the odometry frame. This frame will drift.

We were able to generalize some of the coordinate frames through experiment. We are in the process of carrying out more rigorous tests in order to prove our understanding.

Figure 6 shows the orientations of the odometry and the camera frames.

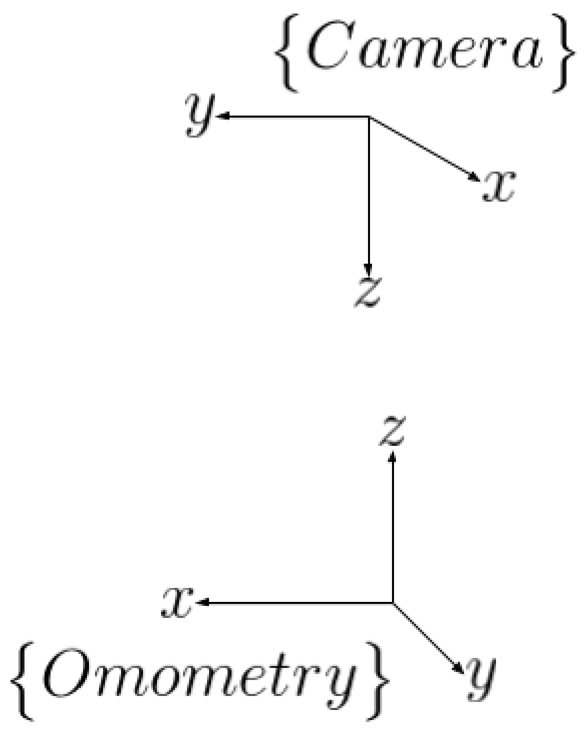


Figure 6) Frame Orientation

Another problem that we had was that it was difficult to figure out how to enable OFFBOARD mode so that we could send velocity and position setpoints.

We could not find any documentation on the problem, but we eventually found the parameters that we needed to change. From this, we were able to get the drone into OFFBOARD mode and we were able to set position and velocity setpoints.

Teamwork

Most of the work that we did this sprint was in a team setting. We broke up into two different teams, Me and Rohan: Job and Erik. Me and Rohan spent a lot of time focusing on getting OFFBOARD control to work. We also used pair programming to iterate over the node I initially wrote shown in Figure 2.

Erik and Job focussed on shifting from local position setpoints to velocity setpoints. These proved more fruitful as of now.

Once we had worked separately, we got back together in one group to finish the demo that we showed in Progress Review 9. We again used pair programming to get multiple eyes on the code.

Working on two separate threads for the same topic was very successful for us. By doing this, we had some work that was lost, but we increased our chances of finding a successful solution by a lot. We expect to use this tactic in the next PR.

Future Work

For the next PR we are going to get autonomous docking with our hardware completed. We will again be working in groups of two on separate threads for the same topic. We are doing this, because we want to try two different methods. By trying two different methods, we can increase our chances of finding the right answer within the next two weeks.

Erik and I will be working on getting autonomous position control using local position setpoints.

Job and Rohan will be working on getting autonomous velocity control using velocity setpoints.

The whole team will shift to the solution that works best once one team has completed their work. This will ensure that we find the right solution as soon as possible. We can then move on to working on separate tasks.