# Individual Lab Report

Erik Sjoberg

Team C – Column Robotics

*Rohan Thakker, Job Bedford, Cole Gulino*

ILR 3

October 30, 2015

## Individual Progress

Since the last ILR I've been focused on bringing up the core of the ROS navigation stack and integrating it with the AR.Drone2 interface. Primarily, this has entailed understanding the launch files and configuration options for the move_base node in addition to the odometry and coordinate transform infrastructure used by ROS.

The following nested tree structure represents the nav_msgs/Odometry message type, which is one of the more complex data structures encapsulating both pose and velocity in two separate reference frames, with their corresponding covariances:

```
# nav_msgs/Odometry represents an estimate of a position and velocity in free space.
std_msgs/Header header                (Header.frame_id = odom frame)
        → uint32 seq   (increasing ID)
          time stamp
          string frame_id
string child_frame_id                 (ID of base_link = frame on quadcopter)
geometry_msgs/PoseWithCovariance pose    (USING odom GLOBAL FRAME)
        → geometry_msgs/Pose pose
                → geometry_msgs/Point position
                  geometry_msgs/Quaternion orientation
            float64[36] covariance          (x, y, z, rot about X, rot about Y, rot about Z) X 6
geometry_msgs/TwistWithCovariance twist   (USING base_link QUADCOPTER FRAME)
        → geometry_msgs/Twist twist
                → geometry_msgs/Vector3 linear (x, y, z)
                  geometry_msgs/Vector3 angular (x, y, z)
            float64[36] covariance          (x, y, z, rot about X, rot about Y, rot about Z) X 6
```

Since the navigation stack requires both the tf coordinate transformation package and the above odometry message types in order to function, it was critical to invest time in understanding the detailed functioning and configuration of these components.

To date, I have succeeded in launching a complete set of nodes for our controller, with data supplied by the AR.Drone2, interpreted by the move_base node, and target velocities fed back to the AR.Drone2 node with all transforms and data messages hooked up appropriately. These velocities have not yet been acted upon or tested.

In addition, I completed a simple visualization demo with rviz, showing our ability to visualize the state of our system using the same transforms and message types that will be used in our final system.

## Challenges

The most significant challenge I've run into was in acquiring the camera feed from the AR.Drone2. It appears that ROS requires a camera configuration file in order to properly display the captured images, however it's not yet clear to me what I need to do in order to get it working properly. I intend to research the appropriate set-up procedures for using the

AR.Drone2 with ROS, since I imagine this problem is faced by most people who attempt to use the system. I don't expect it to be too difficult to solve the issue.

## Teamwork

As can be seen in Figure 1 below, our team is slightly ahead of target when it comes to delivering on our planned tasks, with a nice 40hr chunk of work being completed since Monday. This can be seen by the position (and slope) of the red "total work remaining" line being below that of the blue "planned work" line.

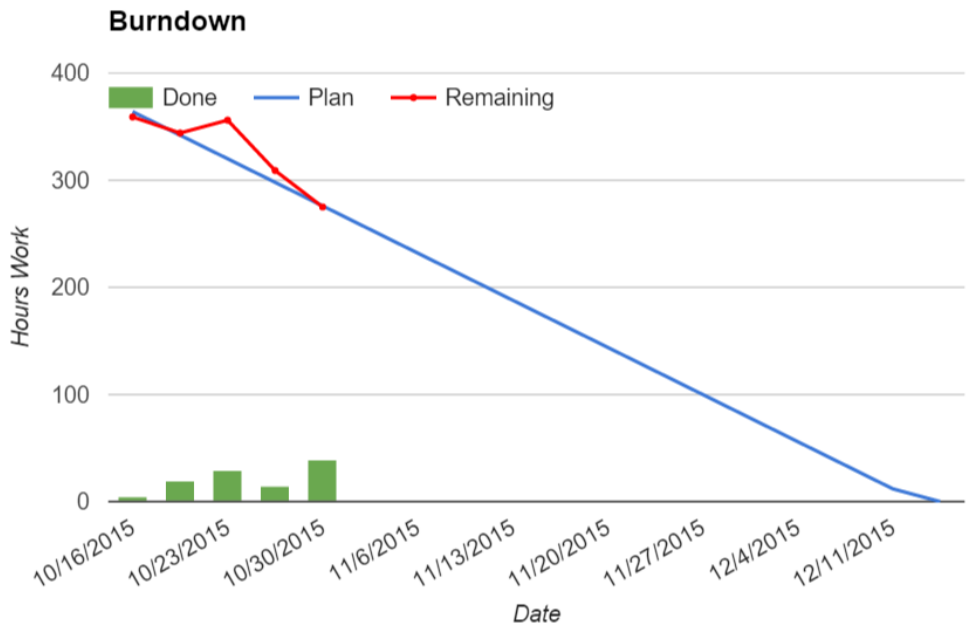


Figure 1: Burndown Chart

Note that the above graph is updated twice weekly which has proven sufficient.

Furthermore, we are ahead of schedule on our current sprint which ends on Monday; all of the most significant tasks are already complete going into the weekend.

As for individual teammates, Cole has taken the lead on identifying the power requirements of our system and planning our power distribution hardware. Cole and Rohan are together researching the details of the Pixhawk flight controller, and in response to the delay in our Iris+ drone delivery they have begun the setup of a complete Gazebo simulator which is provided for this purpose. Job has completed our first pass of the MOVER node (giving us programmatic control over the AR.Drone), and has taken charge of our first prototype dock designs.

Our team's work is synergizing nicely and progress is tangible.

## Plans for Upcoming Work

In the coming week I intend to:

- Test the level of drift in the built-in odometry readings from the AR.Drone2 using Job's MOVER node
- Acquire the camera feed from the AR.Drone2
- Begin evaluating open-source implementations of visual odometry