# Rovio Guide

## General:

- Rovios are available in Upson 317. Ours are the ones that have RO# written on a piece of tape on them.
- Basic support for the Rovio can be found on its official website http://www.wowwee.com/en/support/rovio. This includes the setup software and a description of the API if you want to write your own drivers.
- To connect to the Rovios you will need the IP address, the username, and the password. To see what these are go to https://spreadsheets.google.com/ccc?key=0AvxLSfS7ghEedHA3Rl9LeUh4LWlzTmRsT283MFhFVFE&hl=en. If the IP address gets changed by Red Rover, use the setup software from Wowwee to reset the bot and get the new IP. Please update the above doc to reflect the changes. You can type the IP address into your browsers URL bar to access the robot's web interface.
- There can only be one base station projecting on the ceiling at a time. More than one can cause the localization data to jump between the axis of the two stations. You can either unplug the extra base stations, or to all the robots to charge, just cover the projectors with something.
- The Rovios can start behaving oddly and have trouble connecting when they are low on batteries. You can try to improve its performance by moving the projector on the base and making sure the two shapes are visible on the ceiling. Keep in mind moving this projection will move the localization axis.

## ROS:

### Setup:

Download the zip file from the course website onto the computer running ROS you plan to use, and extract the files. Open a terminal and use "cd" to navigate to the extracted directory. Enter "chmod 777 install.sh" to make the installer executable and then run it by entering "./install.sh". You must run it twice to install fully.

This install builds the dependencies and creates the package “rovio”.

You can navigate to this package by entering “roscd rovio”. In the “nodes” directory there are two files, Rovio.py which is a python driver for the Rovio, and Interface.py which connects this driver to ROS. In the launch directory there is a file test.launch which configures the software for your particular Rovio.

# Usage:

Once the rovio package is installed you can run it by entering “roslaunch rovio test.launch”. The test.launch looks like the following:

```
<launch>
    <node pkg="rovio" name="rovio" type="Interface.py">
        <param name="rovioIP" value="192.168.1.222" />
        <param name="rovioUser" value="admin" />
        <param name="rovioPass" value="roviorobo" />
        <param name="getPos" value="1" />
        <param name="getImage" value="1" />
        <param name="getVel" value="1" />
    </node>
</launch>
```

Roslaunch starts roscore as well as the nodes specified in the launch file. The test file runs the rovio interface and configures it for the IP address and login info. The last three parameters can be used to enable or disable certain data collection to improve update speed for the remaining data. If getPos is 0 the interface will not publish the rovios position from the beacon on the ceiling. getImage turns on/off grabbing frames from the camera. getVel tries to calculate velocity from wheel encoder data and gets obstacle detection info.

- Interface.py publishes the following topics if the data collection for them is enabled:

- /images which has the type sensor_msgs/Image and is the frame grabbed from the Rovio's camera
- /base_pose_ground_truth which has the type nav_msgs/Odometry and has the position and/or the velocity depending on what sampling is enabled. The orientations z value is the yaw instead of the Quaternion value that is normally there.
- /infrared which has the type std_msgs/Bool it is true if an obstacle is detected. getVel must be enabled for this to be published.

- Interface.py subscribes to the following topic, use it to drive the Rovio:
  - `/cmd_vel` which has type geometry_msgs/Twist
  - `/cmd_head` which has type std_msgs/Int64 this allows you to move the rovio's camera. Sending 0 puts it in the lowered position, 1 the middle position, and 2 the upper position.

To see how to use topics look at the HW problems and the documentation on ROS.

When publishing to cmd_vel there are certain limitations. You can only move straight, to the sides, at a 45 degree angle, or turn in place. A current limitation that will probably be removed later is that you can only move at one speed. The controller will just look at what directions you're trying to move. Due to the relatively long time a HTTP request can take you make find that the controller is sending commands faster than they can be sent to the Rovio. If this is a problem decrease your drivers update rate.

Another issue is the latency. We are working to make the communication faster, but you find that you only have a 5Hz or so update rate. Disabling the other sensor information may speed this up, but your rate may still be very low.

# Scripts:

The scripts folder has some tools for viewing the data coming from the Rovio.

- plot.sh – lets you see the position data the Rovio is producing.
- view.sh – creates a window to show the image data being produced by the Rovio. Requires that you install nav_view by typing "rosmake --rosdep-yes nav_view"

# Example:

There is an example controller called rovio_ex.py. This uses the localization data to drive to a point determined by its parameters. You can either run it by starting the driver and then running “python nodes/rovio_ex.py” from within rovio directory, or just running “roslaunch rovio example.launch”

By reading the example and running the scripts while it’s operating you should be able to get the basics of how the driver works. Also look at the launch files for a starting point on making your own.