2020 Field Testing Manual

Rover Setup, Connection, and ROS Configuration

Authors: *Braden S. & Pierre-Lucas A.F.*
Date: *August 25, 2020*

1. Introduction

This document is designed to be a self-contained directive on rover operation for future researchers and students at the Aerospace Robotics Laboratory (ARL). Some content is reiterated from the document *ConnectingControllingHuskyAndArgo* by J.S Fiset (see resources at `https://github.com/brahste/data_collection`). However, substantial changes to the field testing procedures have been implemented, warranting a new manual to explain the setup, both experimental and configurational.

2. Sensors & Hardware

An overview of the sensor and hardware suite is shown in Fig. 1. Three primary sensors are equipped on the rover setup:

1. a ZED stereo-camera (ZED cam). This dual-camera device acts as the eyes of the experimental setup and collects visual data in a variety of forms. For more information on this device see Appendix B.

2. a VectorNav-100 Inertial Measurement Unit (IMU). This device gathers information about linear accelerations and angular velocities induced in the rover chassis. It also provides integrals of these dynamic values (linear velocities, linear positions, angular positions). See Appendix B for more information.

3. Motor current sensors (MC sensors). These sensors detect the current being drawn by the front right and left wheels. See Appendix B for more information.

These sensors are connected as shown in Fig. 1 to two different computers, a third computer is used to remotely access and command the experiments; information on these computers follows:

1. *arl-thinkpad*: a Lenovo P53 laptop. This laptop is bound to the top of the Husky and is equipped with a GTX 1080 Ti Graphical Processing Unit (GPU) which allows it to operate the ZED cam. The *arl-thinkpad* is responsible for hosting the data collection experiments, for initializing the sensor suite, and for commanding data logging procedures and rover motions.

2. *cpr-conu1*: a rover dedicated computer. This computer is located in the bed of the Husky and serves as the ROS interface to Husky's drivetrain, as well as to the IMU and MC sensors. Although this computer plays an essential role in the field testing sessions, it is never directly accessed, instead commands to this computer are implicitly defined within the ROS configuration on *arl-thinkpad*.

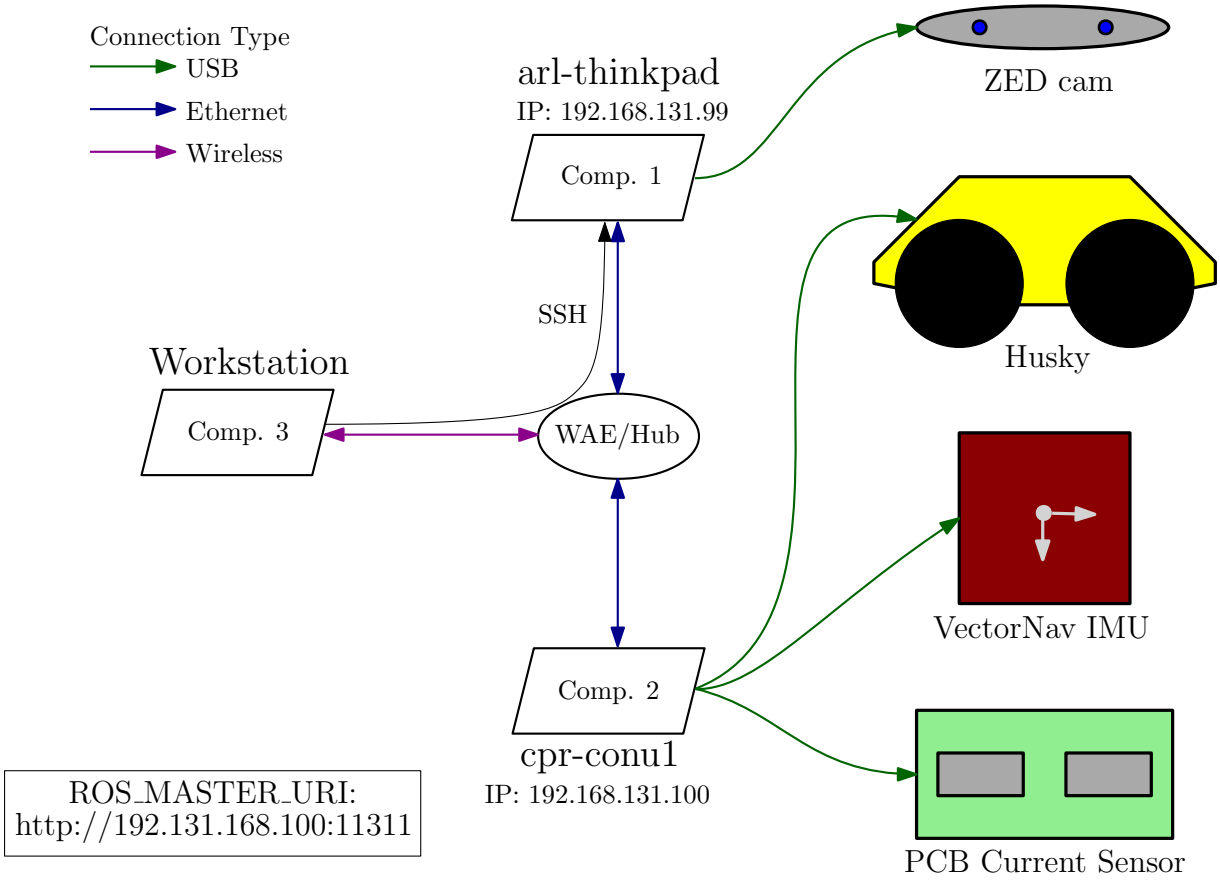

Figure 1: Overview of hardware Setup for 2020 field testing session. The workstation uses SSH to access *arl-thinkpad*, where the `data_collection` package is located. Sensor nodes and motion commands are launched from *arl-thinkpad*, sending information over the network to *cpr-conu1* when needed. Notice that although commands are sent from *arl-thinkpad*, the Master URI is configured to be on *cpr-conu1*; the reason for this is related to launch-on-boot features of the Husky and is discussed further in Section 4.

3. Workstation: any other computer suitable for the needs of the experiment. This computer is used to SSH into *arl-thinkpad*, from which the commands are launched. Ideally, this computer is a small, portable laptop with a strong battery life.

To incorporate the ZED cam into the sensor suite a laptop with a CUDA capable GPU is required. For this purpose, a Lenovo P53 with a GTX 1080 Ti is used.

3. Network Configuration

To command the rover from the Main Computer, ensure that it is connected through a hub. For our experiments a Cisco Wireless Access Point/Hub was used.

3.1. Usage.

4. Software Configuration

4.1. Overview. The field testing experiments are implemented using an array of different ROS packages, shell scripts, and devices; at the highest level these verious software items are localized and governed by the ROS package `data_collection`. Code for the package is hosted at `https://github.com/brahste/data_collection`. The main access point to the hardware suite is through the file `husky_sensors.launch`. Since Husky launches its nodes on boot-up (from the files in `/etc/ros/kinetic/`), `husky_sensors.launch` is responsible for initializing the nodes for peripheral sensors only. A secondary consequence of the Husky nodes intializing on boot is that it is simplest to use *cpr-conu1* as the Master URI (as shown in Fig. 1).

Although *arl-thinkpad* hosts the main data collection package and the ZED node, packages for the IMU and motor current sensors had been configured for past field tests on *cpr-conu1*; we have left this configuration intact. Since the Husky node initializes on boot, for this field testing session it was chosen to use *cpr-conu1* as the Master URI. Future researchers may find it desirable to connect and configure all sensors to *arl-thinkpad* and disable the launch-on-boot feature of Husky, this way the Master URI could be assigned to *arl-thinkpad* and complications regarding network latency would be less pronounced.

Note that `husky_sensors.launch` initializes nodes that exist on both *arl-thinkpad* and *cpr-conu1*. To enable this feature an environment configuration file `husky_env.sh`, located in the main catkin workspace of *cpr-conu1*, must be called when launching nodes over the network.

4.2. Cautionary statements. Because of a ROS limitation the IMU launch file located on *cpr-conu1* (which is called by *arl-thinkpad*) has to be located at the same path on both machines. However, the file on *arl-thinkpad* is merely symbolic as the rest of the `imu_vn_100` package is empty.

The IMU and terrain class publisher are linked at the same publication rate. This does not mean that their publications are synchronized; however, as the publication rate approaches zero, the time difference between publications will become negligible (like in calculus). The default rate is set to 200 Hz.

The bag files are recorded for the time is *should* take for the motion to execute. That is, the total distance commanded is calculated, and divided by the commanded velocity. It may be a good idea to add a few extra seconds on the end of this restriction such that some slip can be account for. In post processing a small threshold on the end of the IMU measurements can be used to truncate the data.

4.3. Usage.

1. Before moving forward with this section, ensure the steps from Section 3.1 are complete. Most importantly the `ROS_MASTER_URI`, hostnames, and IPs must be set correctly. At this point the Husky node should be running, this can be confirmed by calling `rosnode list`. If the nodes [...enter the names of a few nodes...] are up, then indeed the Husky node is running.

2. Initialize the peripheral sensors with:
   `roslaunch data_collection husky_sensors.launch`

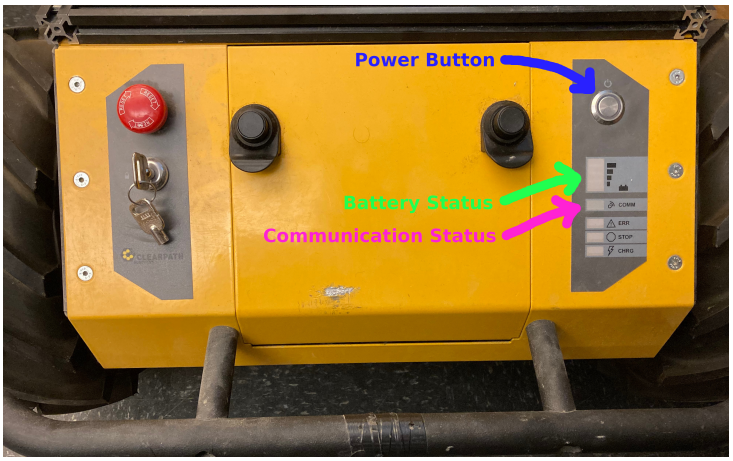

Figure 2: View of Husky status indicator lights on back panel.

Running `rostopic list` you should now be able to see that `imu/rpy`, `imu/imu`, and `zed/*` are being published. As a sanity check (or for debugging) you can run `rostopic echo <topic_name>` and see the data streams from each sensor in your terminal.

It is important that the bagging be called and logged on *arl-thinkpad* because

Upon `roslaunch`-ing, the master node is initialized, along with the following nodes:

5. Running the Experiment

Each experimental trial consists of a CLC-type path and a data logging procedure. Trials are initiated with `husky_experiment.launch`; inside this file are relavent path specifiers, such as turning radius and linear segment travel distance. These path specifiers can be manipulated within the launch file by changing the default value, or by passing command line arguments. The former case is prudent when many of the specifiers are to be altered, while the latter is prudent for rapid experimental iteration over variations in a single (or few) specifiers. The data collected over the duration of each trial is saved in the `bags/` directory of *arl-thinkpad*.

5.1. Pre-experimental setup.

1. **Ensure the ZED cam is connected to *arl-thinkpad*.**
2. **Ensure the IMU is connected to Husky's dedicated computer (*cpr-conu1*).**
3. **Ensure both *arl-thinkpad* and *cpr-conu1* are connected via ethernet to the WAE/Hub and that the WAE/Hub is connected to power.** The ports that the computers are connected to are irrelevant, at the time of writing *arl-thinkpad* was connected to Port 1 and *cpr-conu1* was connected to Port 3.
4. **Ensure the Husky is either elevated with its wheels free (for in-lab testing), or you are in a large unobstructed area (for experimental testing).** Most importantly, do not launch `husky_experiment.launch` without precaution.

5. (recommended) **Have a third *workstation* computer ready.** This computer will be used to SSH into *arl-thinkpad* to send commands remotely, it can also optionally be used to dynamically log the terrain class during experimental trials.

5.2. Starting the Experiment.

1. **Power on the Husky, *cpr-conu1*, and *arl-thinkpad*.** After a few seconds the *communication status* light should turn green (marked in pink in Fig. 2) Because of Husky's launch-on-boot feature, `roscore` will be initialized in the background. You can run `$ rosnode list` to see the nodes that are already up and running. Note that the `/imu/data` node is a Husky internal node and is not the node used for reading VectorNav data.

2. **On *arl-thinkpad* open a terminal**[1]. The environment has been automatically configured in the `.bashrc` file; you should see the ROS hostname, IP, and Master URI printed just above the command line.

   In `.bashrc` on *arl-thinkpad*...

   ```
   # Set Husky computer as master node
   export ROS_MASTER_URI=http://192.168.131.100:11311
   # Configure IP and hostname
   export ROS_IP=192.168.131.99
   export ROS_HOSTNAME=192.168.131.99
   ```

3. (optional) **To verify connectivity you can access *cpr-conu1* with,** `$ ssh administrator@cpr-conu1`. The SSH keys have already been configured so no password should be required; the ROS hostname, IP, and Master URI should also be shown just above the command line. Since the launch files depend on known SSH keys, if a password is prompted you may run into difficulties when running launch files in later steps. Either way, the password is: clearpath.

   In `.bashrc` on *cpr-conu1*...

   ```
   # Set this computer as master node
   export ROS_MASTER_URI=http://192.168.131.100:11311
   # Configure IP and hostname
   export ROS_IP=192.168.131.100
   export ROS_HOSTNAME=192.168.131.100
   ```

4. **In the terminal you just opened on *arl-thinkpad* run,** `arl@arl-thinkpad:$ roslaunch data_collection husky_sensors.launch`. This command starts the various sensor nodes. It is recommended that new users analyze the SUMMARY output to understand the values set on the parameter server and which nodes were initialized.

5. (optional) **Verify topic publications with,** `$ rostopic echo <topic_name>`. You can receive a list of topics with `$ rostopic list`, but if all went correctly then the list will be saturated with `/zed/*` related topics. The topics that are of specific interest are

[1] This computer has been set up with terminator, a terminal emulator. Use Ctrl(Fn)+Shift+E and Ctrl(Fn)+Shift+O to split the terminal vertically and horizonatally.

(a) `/imu/rpy` and `/imu/imu`. These are the topics that are published by the VectorNav IMU.

(b) `/zed/zed_node/rgb/image_rect_color`. This is one of many topics available for collection with the ZED cam. The user may wish to parse the available topics and select the one that fits their data product requirements.

(c) `/terrain_class_integer` This topic publishes an integer at the same rate as IMU publication, it is used to classify the terrain as the experiment proceeds.

6. **On the *workstation* computer, connect to the wireless network CONU01-LSYS2G.** The password is: husky_wireless. This will link the *workstation* to the same network as the other devices; you may wish to manually change the IP of your workstation device to ensure that it does not conflict with any other devices on the network (I changed mine to 192.168.131.101). It is also convenient to set up the `.bashrc` file similar to the other computers such that the Master URI is automatically linked to *cpr-conu1*. To do so, copy and paste the following code into your *workstation*'s `.bashrc`,

```
# Set Husky as master node
export ROS_MASTER_URI=http://192.168.131.100:11311
# Configure IP and hostname
export ROS_IP=192.168.131.101
export ROS_HOSTNAME=192.168.131.101
```

7. (recommended) **Setup *workstation* to remotely reconfigure experimental parameters.** To do this, open a terminal and conduct the following steps,

(a) Set the Master URI with,
`$ export ROS_MASTER_URI=http://192.168.131.100:11311`.
(This step is unnecessary if you set up the `.bashrc` in Step 6 above)

(b) Open the RQT reconfigure GUI with,
`$ rosrun rqt_reconfigure rqt_reconfigure`.

An `rqt_reconfigure` GUI will appear, here you can interact with the `/terrain_class_publisher` value and various parameters for the ZED cam. It is recommended that parameters for the `/husky_velocity_controller` are left untouched.

8. **Access *arl-thinkpad* from your remote *workstation* with** `$ ssh arl@192.168.131.99`. The password is: ARL2020\\.

5.3. Running Experimental Trials.

1. **As a first test, in your SSH session run,**
`arl@arl-thinkpad:$ roslaunch data_collection husky_experiment.launch`. This will start the Husky moving along the default path. You should be able to see the port-side wheels turning faster than the starboard-side (right-hand turn). A `.bag` file will be saved in the `bags/` directory with all trial parameters. Since

the algorithm depends on IMU inputs to know when it is has reached the selected yaw, if this command is run when the Husky is elevated, you will have to Ctrl+C out of the program. With default parameters it will take approximately 30 seconds for the published topics to be bagged.

2. **Continue to launch experiments with the same command.** To manage experimental parameters you can either alter the (default) arguments directly in the launch file, or you can use command line arguments to set values. For example, if you would like to change the turning radius followed by the rover you can use either of the following approaches,

   (a) Open `husky_experiment.launch` with you preferred editor. Find the "turning_radius" tag, `<arg name="turning_radius..._/>`. Change the default property to the desired turning radius, `...default="<float>"...`. Save the file and relaunch.

   (b) Run the command,
   ```
   $ roslaunch data_collection husky_experiment.launch turning_radius:=<float>
   ```

   Where the placeholder `<float>` is replaced by a floating point value of your choosing. In either case a bag will be saved in `bags/` with the newly specified turning radius.

A. Operating System Configuration

B. Node Information