Western Norway
University of
Applied Sciences

MYBOT
SHOP

Husky Development Packages

MYBOTSHOP uG

Author

Tahir Mehmood

January, 2022

MYBOTSHOP

Contents

MYBOTSHOP

1 Overview

Clearpath Husky is a medium sized research robotic platform. it features rugged and all terrain, precise control, easy to use and customized capabilities. The customization for the current Husky include a a precise GPS receiver and a 3D lidar. Both of them are used in for obstacle avoidance and autonomous navigation in outdoor as well as indoor environments.

2 Getting Started

Husky is shipped in big wooden box. A very detailed guideline to unbox Husky can be found here From MYBOTSHOP customization all the software packages are already built in Husky's *catkin_ws*.

For developer's convinience and to setup a software in future a bash script(*husky_installation.bash*) is provided along this guide, follow the procedure below to built everything.

Change your directory to the one containing *husky_installation.bash* and run the following command:

```
sudo ./husky_installation.bash
```

After the command is executed, it will ask you to provide the workspace name and path for the workspace. After providing path and workspace name everything will be installed automatically.

Note: Provide full path like */home/YOUR_USERNAME/Desktop* when prompted for the path.

The workspace is organized initially into ROS specific and ROS independent packages. *YOUR_WORKSPACE/utils* have all the packages which are ROS independent and *src* by default have ROS dependent packages, this is furthermore divided into *mbs* and *third_party* directories. As the names suggests all the packages developed at MYBOTSHOP are in *mbs* directory and all other packages are in *third_party* with their folder names exhibiting their use cases like 3D slam packages are again inside in *src/third_party/3d_slam*. This structure for files and folders is followed for everything. For example all the packages by MYBOTSHOP starts

MYBOTSHOP

with a name *mbs_husky_*.

3 Network

A WLAN router is also installed on Husky and has a network address in *192.168.131.** range. For this reason router is also set on the same network. SSID of the Wifi network is *MYBOTSHOP - HVL - Husky* and the password is *mybotshop2021*.

The ip address of the router is *192.168.131.100* (password for the router network is also *mybotshop2021*). By connecting to the Wifi network one can access Husky on ip address *192.168.131.1*. Moreover if a static connection is desired than setting for the host PC in the same network will also work. For example by setting *192.168.131.51* in the ip and *255.255.255.0* in netmask for the static connection will connect to the Husky network as well. To SSH into Husky run the following command:

```
ssh administrator@192.168.131.1
```

The password for the robot is *clearpath*.

Ouster lidar is also set to work with a static ip which is *192.168.131.4*. If you are connected to the robot either via static connection or Wifi by typing the ip *192.168.131.4* in a browser one can access the sensors homepage with all information.

Nvidia board on the robot is also statically connected to the robot. Its hostname and is *administrator* and the ip address is *192.168.131.2*. The password for Nvidia board is *mybotshop2021*.

The last device which is statically connected to the robot is the gps sensor *Emlid Reach RS2*. The ip of the sensor is 192.168.131.4. Once the sensor is turned on you can also see its status by just typing its ip in a browser, ofcourse you should be in the same network i.e. connected to the Wifi of the robot.

Note: GPS sensor needs to be powered on and off separately from the robot. Also make sure its turned on and the battery has enough power. Also make sure that it is sending the gps coordinates to the robot when working outdoors.

MYBOTSHOP

4 ROS-Packages

As stated above all the ROS packages for this cusotmization of the robot starts with *mbs_husky_* and their built directory is *src/mbs/*. Below comprehensive descriptions for the packages are outlined.

4.1 mbs_husky_description

The customizations done to the factory standard Clearpath Husky also needs to be updated for the *robot_descrption*. This purpose is served by *mbs_husky_description*. All the necessary changes should be updated in *mbs_husky-description/urdf/custom_description.urdf.xacro*. Apart from that nothing else is needed to be done as at the startup of the robot this file is uploaded along with standard Husky *robot_description*. An environment variable *HUSKY_URDF_EXTRAS* should always be pointing to this file location and should never be changed.

Note: The launch file description.launch is for debug purpose only!

4.2 mbs_husky_ouster

mbs_husky_ouster is responsible for running the ROS driver for the lidar. The arguments for ip and port in launch file *ouster.launch* should never be changed in order for sensor to work fine. Sensor metadata could be found in *mbs_husky_ouster/etc/sensor_metadata.json* file.

4.3 mbs_husky_control

This package contains the configuration and the launch file for the localization using gps sensor. The launch file for gps based localization can be run by the following command:

```
roslaunch mbs_husky_control ekf_localization.launch
```

Note: This launch will not interfere with the default *ekf_localization* of the robot. However will start a new node alongside the default one with wheel odometry and imu whereas the new node will have wheel odometry, imu and gps readings.

MYBOTSHOP

4.4 mbs_husky_navigation

For 2d navigation using ROS navigation for Husky this package can be used. This package is built on launch files from *husky_navigation* launch with some modifications in MYBOTSHOP customized Husky.

To create a 2d map:

```
roslaunch mbs_husky_navigation gmapping_demo.launch
```

and after creating a map save the map being built by:

```
rosrun map_server map_saver -f your_map_name
```

Mapless Navigation in outdoor and indoor envrionments can be run by the following launch file:

```
roslaunch mbs_husky_navigation move_base_mapless_demo.launch
<!-- or -->
roslaunch mbs_husky_navigation gps_move_base_mapless_demo.launch
```

Lastly a map based navigation can be launched by:

```
roslaunch mbs_husky_navigation amcl_demo.launch
```

Note: Never forget to copy the map to *mbs_husky_navigation/maps/*

4.5 mbs_husky_slam

3D localization and mapping is achieved by the package *mbs_husky_slam*. To create a 3d point cloud map of an environment run the following launch file:

```
roslaunch mbs_husky_slam indoor_slam.launch
```

In case of outdoor environment run the following command to create a 3d map:

```
roslaunch mbs_husky_slam outdoor_slam.launch
```

To view the map being built run the following commands:

```
roscd hdl_graph_slam/rviz
rviz -d hdl_graph_slam.rviz
```

At the end to save a generated map call a ros service:

MYBOTSHOP

```
rosservice call /hdl_graph_slam/save_map "resolution:
0.05 destination: '/full_path_directory/map.pcd'"
```

4.6 mbs_husky_utilities

This pacakge have several utilities which are useful for this project development. For example *bag_record.launch* to record *rosbag* file with desired topics in case needed for debug purposes, *laserscan.launch* to convert 3d pointcloud data to 2d laserscan messages, *map_upload.launch* read a pcd file created using 3d slam package and publishing it as a ros message. The most important launch file is *pcl_filter.launch* this launch runs a ROS node which can filter pointclouds given radius around the sensor or angle range for which to filter and publish point cloud messages. In case of current cusotmization the lidar sensor detects the router and the onboard gps topwer which later on is dealt as an obstacle while autonomous navigation which is not desirable. So this launch file can be used to filter out any objects around the sensor by appropriately fine tuning the parameters.

4.7 mbs_husky_waypoint_navigation

Outdoor GPS based autonoumous navigation can be achived by *mbs_husky_waypoint_navigation*. To laucnh the process run the following command:

```
roslaunch mbs_husky_waypoint_navigation waypoint_navigation
.launch
```

This launch will run a utility node which specifies what actions needs to be taken. First it is reccomended to calibrate the robot's heading atleast twice. After that collect the gps waypoints while navigating the robot. Exit the point collection which will save the points automatically. After point collection you can use autonomously navigate the robot using these saved waypoints by pressing the desired key which is also printed on the console.

MYBOTSHOP

4.8 mbs_husky_people_tracking

This package demonstrate a small use case of 3d lidar by people tracking via using *hdl_people_tracking*. To track without localization launch *people_tracking_static.launch* or *people_tracking.launch* with localization.

4.9 mbs_husky_startup

When ROS is running at startup this indicated by a green light on Husky comm led. In case of a red line there is some problem and Husky won't work as ROS is not running due to some error.

Every launch file which needs to run at the startup can be created as OS services. A ROS packages *robot_upstart* can be used to create startup jobs from launch files. By default the service responsible for startup jobs on Husky is *mbs_husky*. You can check its status by:

```
sudo service mbs_husky status
```

The launch file *mbs_husky.launch* includes everything which is being launched at the startup. This file can be updated to include all the nodes needed to run at the startup. Startup job and its content can be verified in the directory */etc/ros/melodic* here the original launch file will be copied.

If the systems breaks at startup you can start debugging by:

```
roslaunch mbs_husky_startup mbs_husky.launch
```

This will show which part of the launch file is having errors or you can start runing everything in the launch file one by one in seperate terminals.

Note: Do not create multiple startup jobs with multiple launch files this will break the system. Only one launch file should be created and everything else should be included in this.

MYBOTSHOP