Dyno Documentation

Release 1

Samuel Lindgren

Sep 19, 2018

Contents

1 General **1**
1.1 Setup environment 1

2 Dyno ROS system **5**
2.1 Simulation 5

3 Other **13**
3.1 Repositories 13

CHAPTER 1

General

1.1 Setup environment

This chapter will cover how to setup and configure your development environment.

The first step is to setup your ssh-key if not done already.

1.1.1 Installing ROS

The Dyno robotics stack is supported on ROS Kinetic and Ubuntu 16.04 (Xenial). The following commands will install ROS on a fresh Ubuntu 16.04 system.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /
↪etc/apt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key␣
↪421C365BD9FF1F717815A3895523BAEEB01FA116
sudo apt-get update
sudo apt-get install ros-kinetic-desktop-full
apt-cache search ros-kinetic
sudo rosdep init
rosdep update
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool␣
↪build-essential ros-kinetic-rosbridge-server
```

The official installation guide can be found here.

1.1.2 Raspberry Pi

The easiest way to setup ROS on Raspberry Pi is to install Ubuntu Mate. Although it's possible to flash your SD using the terminal I would recommend to use a graphical tool to assist, because it really sucks to destroy your main drive.

Etcher is available on Linux, MacOS and Windows to write the image to the SD card.

Note: Ubuntu Mate is not currently supported on Pi B3+, don't use the B3+ model for now!

The next step is to enable SSH. Hookup the Pi to a keyboard and monitor and execute the following.

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

Then copy your ssh-key from your workstation to the Pi.

```
ssh-copy-id -i ~/.ssh/id_rsa.pub pi@x.x.x.x
```

Note: The latest Firefox (version 55) is not supported on the Raspberry kernel, use `sudo apt-get install chromium-browser` instead.

You might want add the Pi to your *~/.ssh/config* file to make your life easier.

Follow the same steps on *Installing ROS* for Xenial to complete the installation on Mate.

1.1.3 Setup workspace

Catkin Tools

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu `lsb_release -sc` main" > /
↪etc/apt/sources.list.d/ros-latest.list'
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
sudo apt-get update
sudo apt-get install python-catkin-tools
```

Dyno shell tools

```
mkdir ~/Code && cd ~/Code
git clone git@github.com:ErikOrjehag/dyno-shell.git
cd dyno-shell
cp rosenv-templ.sh ~/.rosenv
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin build
echo "source ~/.rosenv" >> ~/.bashrc
source ~/.bashrc
```

ROS repositories

```
git clone git@github.com:samiamlabs/dyno.git
git clone git@github.com:samiamlabs/omnibot_driver.git
git clone git@github.com:samiamlabs/mecanum_controller.git
type ddep
ddep
dmake
```

Documentation

```
cd ~/Code
git clone git@github.com:samiamlabs/dyno_docs.git
sudo pip install sphinx sphinx-autobuild sphinx-rtd-theme testresources
cd dyno_docs
make html
```

1.1.4 Docker

Follow instructions here to install Docker.

Follow instructions here to add your user to docker user group to not need sudo, log out and in again for changes to take effect.

Follow instructions here to create the dyno container.

CHAPTER 2

Dyno ROS system

2.1 Simulation

2.1.1 Getting Started

Look at instructions for installing ROS here: *Installing ROS*.

Go the dyno repository and follow instructions to build dyno's ros repository.

Start simulation by running:

```
roslaunch dyno_gazebo bringup.launch
```

2.1.2 Robots

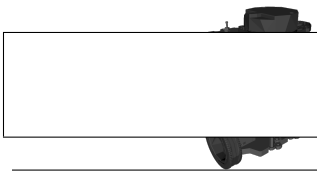
Omnibot

```
export DYNO_BASE=omnibot
export DYNO_USE_ROS_CONTROL_FOR_BASE="false"
roslaunch dyno_gazebo bringup.launch
```

Turtlebot3

```
export DYNO_BASE=turtlebot3
export DYNO_USE_ROS_CONTROL_FOR_BASE="true"
roslaunch dyno_gazebo bringup.launch
```

Forklift



```
export DYNO_BASE=forklift
export DYNO_USE_ROS_CONTROL_FOR_BASE="true"
roslaunch dyno_gazebo bringup.launch
```

Magni

```
export DYNO_BASE=magni
export DYNO_USE_ROS_CONTROL_FOR_BASE="true"
roslaunch dyno_gazebo bringup.launch
```

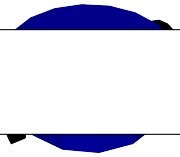
Holonomic

```
export DYNO_BASE=holonomic
export DYNO_USE_ROS_CONTROL_FOR_BASE="false"
roslaunch dyno_gazebo bringup.launch
```

Diff Drive

```
export DYNO_BASE=diff_drive
export DYNO_USE_ROS_CONTROL_FOR_BASE="true"
roslaunch dyno_gazebo bringup.launch
```

Quadrotor



```
export DYNO_BASE=quadrotor
export DYNO_USE_ROS_CONTROL_FOR_BASE="false"
roslaunch dyno_gazebo bringup.launch
```

The quadrotor platform is not fully implemented. Mostly because of a conflict in *Protocol Buffers* (*protobuf*) between Cartographer 1.0 and hector_quadrotor.

2.1.3 Environment Variables

Descriptions

DYNO_BASE

What robot base shoud be used. This applies both to simulation and real robots.

DYNO_DEFAULT_RAPP

What Robot App (Rapp) shoud start automaticly when you bring up a Concert Client.

Simulation: `roslaunch dyno_gazebo bringup.launch`
Real Robot: `roslaunch dyno_bringup bringup.launch`

DYNO_USE_ROS_CONTROL_FOR_BASE

Changes whether the robot base subscribes to */mobile_base_controller/cmd_vel* or uses ros_control to control the joints directly.

If set to true, a controller that translates */mobile_base_controller/cmd_vel* to joint velocities is started.

Diffrent controllers are started depending on what base is used. The controllers currently used are: mecanum_controller, diff_drive_controller, four_wheel_steering_controller and tricycle_controller.

Defalut Gazebo physics with ros_control doesn't currently work for the holonomic/omni plattform, so this variable sould be set to false when simulating theese platforms.

DYNO_JOY_TYPE

Type of joystick.

JOY_SERIAL_PORT

Serial port to use for joystick.

Values

Name	Defalut Value	Optional Values
DYNO_BASE	turtlebot3	omnibot, magni, holonomic, diff-drive, quadrotor
DYNO_DEFAULT_RAPP	dyno_common_rapps/exploration	dyno_common_rapps/waypoint_navigation
DYNO_USE_ROS_CONTROL_FOR_BASE	false	true
DYNO_JOY_TYPE	gamepad	xbox360, playstation
JOY_SERIAL_PORT	/dev/input/js0	–

2.1.4 Teleop

GamePad

Button #	Function (details below)
1	Control robot movement, based on 2 and 3
2	Deadman, forward and sideways
3	Deadman, forward and rotation
4	Not used
5	Not used
6	Joystick/mouse-keyboard
7	Power, bluetooth sync


3
0
4
12
7
5
15
13
6
14
1
16
2


8
9
10
11
SHARE
OPTIONS

Playsation

Button #	Function (details below)
0	Not used
1	Control robot movement, forward to the side
2	Control robot movement, rotate
3	Not used
4	Not used
5	Not used
6	Not used
7	Not used
8	Not used
9	Not used
10	Primary deadman
11	Disable safety (not implemented)
12	Not used
13	Not used
14	Not used
15	Not used
16	Pair/unpair with robot

To pair the controller with the robot, press the middle button (16) once the robot has powered on. The big LED on the controller will turn a solid blue when successful. To unpair, hold the button for 10 s. The blue LED indicator on the back will turn off. Alternativly, plug in the usb connector.

To drive the robot base, hold the primary deadman button (button 10 above) and use the left joystick.

2.1.5 Guide: Exploration

Open a terinal and run *roslaunch dyno_gazebo bringup.launch*

Open web_interface in a browser. Make sure that hostname set to your computer.

2.1.6 Guide: Loading and saving maps

Open a terinal and run *roslaunch dyno_gazebo bringup.launch*

Open web_interface in a browser. Make sure that hostname set to your computer.

Start making a new map toggling the *cartographer new 2d* switch in the *Capabilies* section of the web interface.

Enable navigation with move_base by toggling navigation.

Move around in the environment by sending navigation goals in rviz.

You also drive around manually by using a joystick or keyboard. See *Teleop* for more details.

Alternativly you can use the *Exploration* Rapp to make the robot explore the environment autonomously. See *Guide: Exploration*.

When you have finished mapping, first click the *Finish Trajectory 0* button and then the *Save State* button in the *Cartographer*.

The *Finish Trajectory 0* button freezes the trajectory and runs some optimizations, so you will not be able to coninue mapping after pressing is. (Saving the sate without finishing trajectory first also works)

Turn off mapping.

Toggle the *cartographer load state 2d* switch to load the saved map.

The robot will start on a new trajectory (green) and continue mapping from where it left off.

2.1.7 Guide: Route Scheduler

Open a terinal and run:

```
roslaunch dyno_gazebo bringup.launch
```

Open web_interface in a browser. Make sure that hostname set to your computer.

Start the Route scheduler app by clicking on *Route Scheduler* in the *Robot Apps* section.

Previously saved locations will be automaticly loaded when you start the app. To add additional locations, move the robot to where you want it and press the *Add Current Location* button in the *World State* section of the web interface.

Add a waypoint/location to the queue by selecting it in the dropdown list and clicking the + button.

Click the *Start* button to start moving the robot.

To get an idea of what is happening behind the scenes, run this in a new terminal:

```
rosrun rqt_py_trees rqt_py_trees
```

2.1.8 Unity

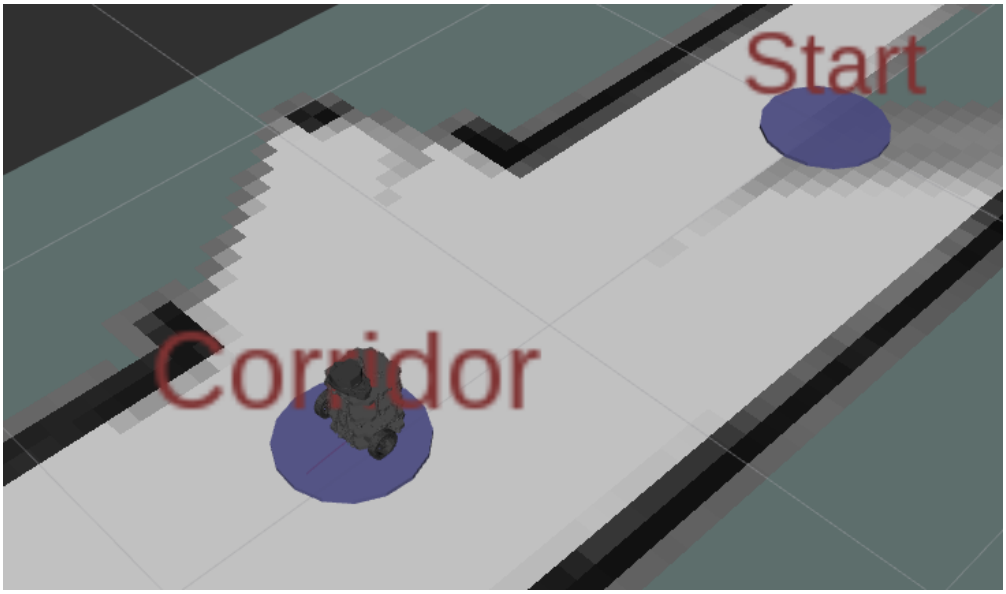
You can use ros-sharp to import dyno robots into unity. You need to run rosbridge and a file server that trasfers the urdf and the mesh files to the windows computer running unity:

```
roslaunch dyno_unity bringup.launch
```

Importing URDF to Unity form Ros

Moving pallet in Unity

The dyno repository allow you to simulate a few different robot platforms.


Start
Corridor

CHAPTER 3

Other

3.1 Repositories

This is an index of all relevant git repositories.

3.1.1 dyno

Short description of this repo...

More...

3.1.2 dyno_hsr

Short description of this repo...

More...

3.1.3 omnibot_driver

Short description of this repo...

More...

3.1.4 mecanum_controller

Short description of this repo...

More...

3.1.5 dyno-shell

Short description of this repo...

More...

3.1.6 dyno_docs

Short description of this repo...

More...