EtherCAT Dextrous Hand

- User Manual -

Shadow Software team - software@shadowrobot.com

June 22, 2012

Contents

1 Overview

This user manual will get you started with using our EtherCAT hand or simulated robots. We'll guide you through installing the code in the **section 2**, show you some basic commands and code samples in **section 4**, and explain where to find things in **section 3**. This guide ends with some pointers to essential documentation you should read (**section 5**), and some troubleshooting tips (**section 6**).

If you have any questions, don't hesitate to contact us at: `software@shadowrobot.com`.

2 Installing our Software

For a typical user, we really recommend installing our software directly from the apt repository (see section 2.1). This will ensure that you keep the latest released software on your computer.

We assume you've already installed ROS electric, as detailed on the ROS Wiki[1].

However if you plan to modify some parts of our code, you can use an overlay to install some of the sources from our publicly available Launchpad repository on top of your existing installation.

2.1 From our apt repository (recommended)

TODO: Mark fill this in!

2.1.1 Install a sr_config overlay for the EtherCAT Hand

If you have an EtherCAT Hand, you should create an overlay for the `sr_config` stack: this stack contains all the different configuration files which can be modified to reflect your specific system. You shouldn't use the one installed by default as they are defined for the simulated hand.

To create this overlay, just follow the **section 2.2**.

2.2 From sources

If you need to modify one of the packages for your own use, you'll want to first install the released version from the apt repository as explained in the **section 2.1**. This will ensure that you have all the code you need. Once you've identified in which stack the package you want to modify is, you can create an overlay[2] to use the source version.

To create an overlay, we'll use the rosinstall[3] utility. In this example, we'll overlay the sr_config[4] stack, using trunk version of the code hosted on launchpad using bzr[5]: `lp:sr-config`.

- First we need to install the rosinstall utility if you haven't done so already:

```
> sudo apt-get install python-setuptools bzr
> sudo easy_install -U rosinstall vcstools
```

- We can now create a rosinstall file that will contain the information about the stacks we want to overlay (in our case we're overlaying the sr_config stack only):

[1] url: http://www.ros.org/wiki/electric/Installation/Ubuntu
[2] **Overlay:** use another version of a stack than the default installed one.
[3] url: http://ros.org/wiki/rosinstall
[4] url: http://launchpad.net/sr-config
[5] url: http://bazaar.canonical.com

shadow_robot.rosinstall

```
- bzr:                          # VCS used (bzr in our case)
  uri: 'lp:sr-config'           # public uri for the code
  local-name: sr_config         # name under which we install the stack
```

- We're now going to create a workspace (in the ~/`workspace` directory), based on your currently installed ROS system (we're assuming you're using ROS electric), and use the rosinstall file we created to download the code into it.

```
> rosinstall ~/workspace /opt/ros/electric/setup.bash
> rosinstall ~/workspace shadow_robot.rosinstall
```

- To use the newly created environment, simply source the setup.bash created in the previous step (to have it sourced automatically when you open a new terminal, simply add it to your ~/`.bashrc` as show on the second line).

```
> source ~/workspace/setup.bash
> echo source ~/workspace/setup.bash >> ~/.bashrc
```

- Finally, you can build the whole stack in one go using the `rosmake` command:

```
> rosmake --rosdep-install --rosdep-yes sr_config
```

2.2.1 Available rosinstall files

If you want to install all our stacks in one go, you can use the `rosinstall` files available here[6].

TODO: Mark fill this in! May be we should have a simpler shadow_robot.rosinstall which doesn't contain the unstable stacks (sr-demo / denso / object-manip / sr-manip)?

Once you've downloaded the code using the `rosinstall` commands described previously, you can build all our stacks:

```
> rosmake --rosdep-install --rosdep-yes shadow_robot \
  shadow_robot_ethercat sr_contrib sr_visualization \
  sr_teleop
```

2.3 Import our custom Environment Variables

As described in **section 3.2.1**, we're using a file to define some useful environment variables. To make sure the variables are imported in your setup, you need to source it in your ~/`.bashrc`, after sourcing the ros setup.bash. It should look like this:

~/.bashrc

```
#source the ros setup.bash
/opt/ros/electric/setup.bash
```

[6] url: http://bazaar.launchpad.net/~shadowrobot/sr-build-tools/trunk/view/head:/data/shadow_robot.rosinstall

```
#or the one created in your workspace
# if you used an overlay
#source ~/workspace/setup.bash

#now source our additional environment variables
source `rosstack find sr_config`/bashrc/env_variables.bashrc
```

3 Navigating our code

TODO: Mark fill this in! Just describe quickly the way the stacks are organised (for shadow robot), name the main packages and give quick description + explain how the settings are set.

3.1 Important Packages

- **sr_hand:** Contains the code and the different launch files for the simulated Hand.
- **sr_edc_launch:** Contains the main launch file for the EtherCAT Hand.

3.2 Where to find the settings

We moved all the configuration to the `sr_config` stack. The idea is to isolate the data that is specific to the user in a given stack, this way you can easily create an overlay, as described in **section 2**, to make sure your specific configuration is not overwritten by an update.

3.2.1 Specifying system parameters

3.2.2 Specifying default controllers settings

4 Running our code

4.1 Starting the real hand

4.1.1 Setting up the Ethernet port

The first time you plug an EtherCAT Hand, you probably need to check on which Ethernet port the hand is connected (you probably have 2 network ports in your computer, one of them would typically be connected to the network, the other one to the hand).

An easy way to see which port the hand is using is to unplug the hand network cable from your computer and then plug it back in. You can now type `dmesg` and you should see a line saying which port has just been plugged in, for example, in the lines below, you see that `eth1` is used by the hand:

```
[16661.901920] e1000e: eth1 NIC Link is Down
[16666.460518] e1000e: eth1 NIC Link is Up 1000 Mbps Full Duplex
```

Now that we know which Ethernet port is used by the hand (`eth1` in our example), we can edit the interface in the system parameters, as shown in **section 3.2.1**:

- Make sure you've created your overlay for the `sr_config` stack as described in the **section 2.1.1**.
- Let's open the configuration file:

```
> roscd sr_config
> cd bashrc
> gedit env_variables.bashrc
```

- Set the `ETHERCAT_PORT` to the correct value (it is `eth1` by default)
- Save and restart your terminal to make sure the new options have been taken into account.

4.1.2 Starting the driver

You can now start the driver. To be able to run the driver, you need to elevate the permissions to root access:

```
#elevate permissions
> sudo -s
> roslaunch sr_edc_launch sr_edc.launch
```

4.2 Starting the simulated robot

To start the simulated robot you can simply run:

```
> roslaunch sr_hand gazebo_hand_motor.launch
```

Or if you want a simulated Shadow Arm as well as our hand:

```
> roslaunch sr_hand gazebo_arm_and_hand_motor.launch
```

4.3 Checking it works

4.3.1 Reading data out of the Hand

You can first have a look at the data coming out of the hand. The data is published on the `/joint_states` topic (or `/gazebo/joint_states` for the simulated hand). You'll see the incoming data streaming in your console. You have access to the joint names as well as the current positions, velocity and effort for each joint as shown below (stop with `ctrl+c`):

```
> rostopic echo /joint_states
---
header:
  seq: 7235
  stamp:
    secs: 73
    nsecs: 785000000
  frame_id:
name: [WRJ2, WRJ1, FFJ4, FFJ3, FFJ1, FFJ2, ...]
position: [-0.007345193851005405, 0.0007282010249589632, ...]
velocity: [0.0002164649710794793, 0.0015999764385997265, ...]
effort: [0.0, 0.0, 0.0, 0.0, ...]
```

You should also be able to visualise the current state of the Hand using rviz.

TODO: Mark fill this in! May be check if we can have a simple rviz config that shows the robot and explain how to load it (it's easier than explaining to add the robot model etc...) + add screenshot?

4.3.2 Sending Commands to the Hand

You can easily control the different joints of the Hand: each of the joint is controlled by a controller. Each of those controllers have a `/command` and a `/state` topic. The `/state` topic is printing useful debug information regarding the controllers, while the `/command` topic is used to send new targets to the controllers.

To identify the topic on which you want to send the target, you can use the `rostopic list` command. Let's say we want to send a target to `FFJ3`:

```
> rostopic list | grep ffj3
/sh_ffj3_mixed_position_velocity_controller/command
/sh_ffj3_mixed_position_velocity_controller/state
```

This means that we're going to publish our target on the topic:
`/sh_ffj3_mixed_position_velocity_controller/command`.
To do this, we can simply use the `rostopic pub` command. For example, to send a target of 1 rad to `FFJ3` at 10Hz:

```
> rostopic pub /sh_ffj3_mixed_position_velocity_controller/command \
  std_msgs/Float64 -r 10 1.0
```

To see a code example you can have a look at the package `sr_ethercat_example`.

5 Where to go next

The ROS wiki is probably the best place to start learning about ROS. We strongly recommend going through the excellent ROS Start Guide, especially the set of tutorials.

You should also look at some of our tutorials for the EtherCAT Hand.

We'd be delighted to get your contributions (patches, nice demos, etc...). Don't forget our software is open-source, so if you want to contribute, you should probably check this wiki page out.

6 What to do if it doesn't work

6.1 Check your ROS setup

You can use very simple examples to check different things:

- Check you can access our stacks (if this command doesn't work, make sure you've installed the code properly and sourced the generated setup.bash as described in **section 2**):

```
> roscd shadow_robot
```

- Check you can publish and subscribe to a simple topic. You'll need to start each of the following commands in a separate terminal. The first command is used to start the rosmaster. The second one will publish some data on the `test` topic while the third command subscribes to the topic and print the output on the screen. If this test works, you should see the incoming data streaming in your third terminal.

```
#in terminal 1, start the rosmaster
> roscore
#in terminal 2 publish data at 5Hz on /test
> rostopic pub /test std_msgs/Float64 -r 5 1.0
#in terminal 3, subscribe to the /test topic
# you should see the streaming data
# stop it with Ctrl+c
> rostopic echo /test
data: 1.0
---
data: 1.0
---
data: 1.0
```

- If all hope is lost, run the roswtf utility.

6.2 Use the ROS community

A good thing to remember is that you're probably not the first one having run into this problem. You should check the ROS community website: answers.ros.org and post your question if you can't find a relevant answer.

6.3 Contact us

Don't hesitate to contact us: `software@shadowrobot.com`.

7 Changelog

TODO: Mark fill this in! Just add a changelog.tex to each of our stack containing a quick description?

7.1 Shadow Robot Stack

This is the log for the Shadow Robot stack.