Open Source Rover: Software Instructions

Authors: Michael Cox, Eric Junkins, Olivia Lofaro

Jet Propulsion Laboratory
California Institute of Technology

Contents

1 Setting up the Raspberry Pi 3

In this section, we'll go over setting up the Raspberry Pi and setting up all the code that will run the rover. Our rover uses ROS (Robotic Operating System), which will be installed on the Raspbian operating system; we will set up both of these below. We will also set up the bluetooth pairing from the android device.

1.1 Getting the Raspberry Pi running

The first step is to install Raspbian on your Raspberry Pi. To install Raspbian, we recommend following the "Getting started with Raspberry Pi" tutorial at:

https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started

Once you finish the above tutorial, you should be able to boot your Raspberry Pi and see a desktop!

1.2 Installing ROS

Now that we have a clean installation of Raspbian, we need to install ROS.

First, make sure you update all the packages on your pi:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

(The above commands may take a few minutes to run.)

Next, we will install ROS (and specifically, the version called 'Kinetic'). To install ROS, follow the installation instructions at:

http://wiki.ros.org/ROSberryPi/Installing%20ROS%20Kinetic%20on%20the%20Raspberry%20Pi

1.3 Setting up serial communication

In this project we will be using serial communication to talk to the motor controllers. Serial communication is a communication protocol that describes how bits of information are transferred between one device and another. You can find more information on serial communication at:

- https://learn.sparkfun.com/tutorials/serial-communication

Run the following commands on the Pi to setup/enable the serial communication for the RoboClaws:

```
sudo raspi-config
```

In the raspi-config menu, set the following options:

- Interface Options $->$ Serial
- Would you like a login shell to be accessible over serial? $->$ No
- Would you like the serial port hardware to be enabled? $->$ Yes
- Would you like to reboot now? $->$ Yes

Once the Pi reboots, open up a terminal again and look at the serial devices:

```
ls -l /dev/serial*
```

Make sure that this shows serial0 -> ttyS0 . If it does not, ensure that you have followed every step in this tutorial in order. Next, edit the /boot/cmdline.txt file:

```
sudo nano /boot/cmdline.txt
```

Change **ONLY** the part with "console = ..." to read "console=tty1" and remove any other instance where it references console. The first bit of that line should look similar to the Figure 1: [1]

[1] It is okay if it does not exactly match what we show here; the important part is that the "console=tty1" flag matches.

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p7
```

Figure 1: boot/cmdline.txt File

Below is a link to help with the above steps if you need additional help:

- https://spellfoundry.com/2016/05/29/configuring-gpio-serial-port-raspbian-jessie-including-pi-3/.

Once you've completed the above steps, reboot the Pi.

`sudo reboot`

1.4 Downloading the Rover Code

On the Raspberry Pi, open up a terminal **(ctl + alt + t)** and then type the following commands[2]:

Move to your home directory: `cd /home/pi`

Clone the rover code repository: `git clone https://github.com/nasa-jpl/osr-rover-code osr`

Move into the directory you just cloned: `cd osr`

Check out the osr-ROS branch: `git checkout osr-ROS`

Pull to make sure you have the latest code: `git pull`

1.5 Building the Rover Code to work with your ROS installation

Now, you must build the rover code packages so that ROS can run them on your Raspberry Pi. To do this, run the following commands in a terminal:

Create a catkin workspace directory and move into it:

[2]In this document, terminal commands will be `highlighted`.

```
mkdir -p /home/pi/osr_ws/src && cd /home/pi/osr_ws
```

Build a basic, empty catkin project: `catkin_make`

Move to the directory containing the main OSR logic: `cd /home/pi/osr/ROS`

Copy the OSR source files to the catkin workspace:

```
mv led_screen/ osr/ osr_bringup/ osr_msgs/ /home/pi/osr_ws/src
```

Move back to your catkin workspace: `cd /home/pi/osr_ws`

Build the sensor_msgs ROS package:

```
rosinstall_generator sensor_msgs –rosdistro kinetic –deps –wet-only –tar >kinetic-sensor_msgs-wet.rosinstall
```

```
wstool init src kinetic-sensor_msgs-wet.rosinstall
```

Run a "catkin make" of the OSR workspace (this will take some time): `catkin_make`

If your catkin make command succeeded, you should now see new build/ and devel/ directories in /home/pi/osr_ws.

Source your newly created ROS environment:

```
source /opt/ros/kinetic/setup.bash
```

```
source /home/pi/osr_ws/devel/setup.bash
```

```
source /home/pi/osr_ws/devel/setup.sh
```

Use nano (or your editor of choice) to add the above three 'source' commands to the end of your /.bashrc file, each on its own line. This will allow you to re-launch the OSR code without needing to source the OSR ROS environment each time:

```
sudo nano /.bashrc
```

1.6 Running the Rover Code Manually

At this point, all the code should be ready to run on your rover! You can start the rover code by running the following commands in a terminal:

Move to your built catkin workspace: `cd /home/pi/osr_ws/`

Run the roslaunch command (read more about roslaunch at http://wiki.ros.org/roslaunch): `roslaunch osr_bringup osr.launch`

You should now see the rover trying to launch all its ros "nodes", which are the various parts of the robot that all communicate to each other. Figure 2 more information about the ROS ecosystem on this rover:

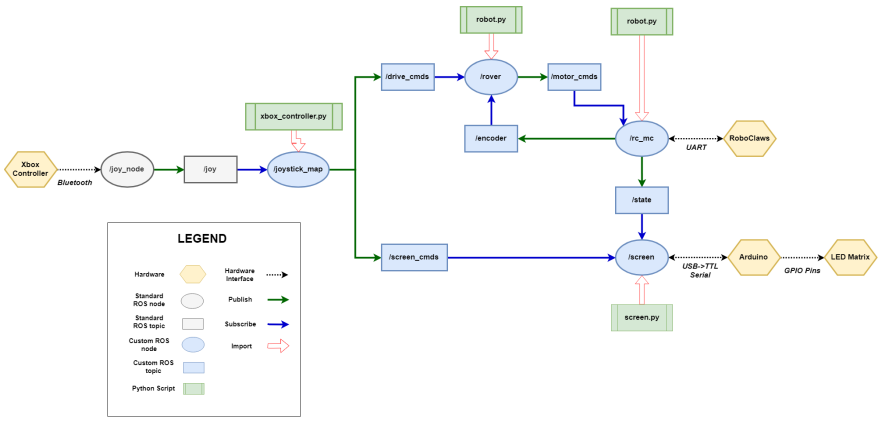

Figure 2: OSR ROS Architecture

You can read more details about the software / ROS architecture in the README document in the osr-rover-code repository at https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS.