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 how to set up all the code that will run the rover. We will also set up the bluetooth pairing from the android device.

1.1 Getting the Raspberry Pi online

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 Downloading the Rover Code

On the Raspberry Pi, open up a terminal **(ctl + alt + t)** and then type the following commands[1]:

```
cd /home/pi
```

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

1.2.1 Installing packages

Run each of the commands below to install the packages you will need to use Bluetooth communication on the Raspberry Pi. Open up a terminal on the Raspberry Pi and type the following commands:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

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

```
sudo apt-get install python-bluez
```

```
sudo apt-get install python-pip python-dev ipython
```

```
sudo apt-get install bluetooth libbluetooth-dev
```

```
sudo pip install pybluez
```

```
sudo systemctl start bluetooth
```

Now you need modify a line of code in the Bluetooth service file. Start by typing:

```
sudo nano /lib/systemd/system/bluetooth.service
```

Add "-C" after 'bluetoothd' on line 9. The file should now look like the Figure 1:

```
1  [Unit]
2  Description=Bluetooth service
3  Documentation=man:bluetoothd(8)
4  ConditionPathIsDirectory=/sys/class/bluetooth
5
6  [Service]
7  Type=dbus
8  BusName=org.bluez
9  ExecStart=/usr/lib/bluetooth/bluetoothd -C
10 NotifyAccess=main
11 #WatchdogSec=10
12 #Restart=on-failure
13 CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
14 LimitNPROC=1
15 ProtectHome=true
16 ProtectSystem=full
17
18 [Install]
19 WantedBy=bluetooth.target
20 Alias=dbus-org.bluez.service
```

Figure 1: bluetooth.service File

Once you have made this change, exit and save (ctrl + x, y, enter), and then reboot the Pi:

```
sudo reboot
```

Once the Pi has rebooted, open up a terminal and type:

```
sudo sdptool add SP
```

2 Control Method

The first thing to do is select which control method you want to use. There are two methods that are already built into the code to control the rover; one is a smartphone app run on Android devices, the other is using a wireless Xbox controller.

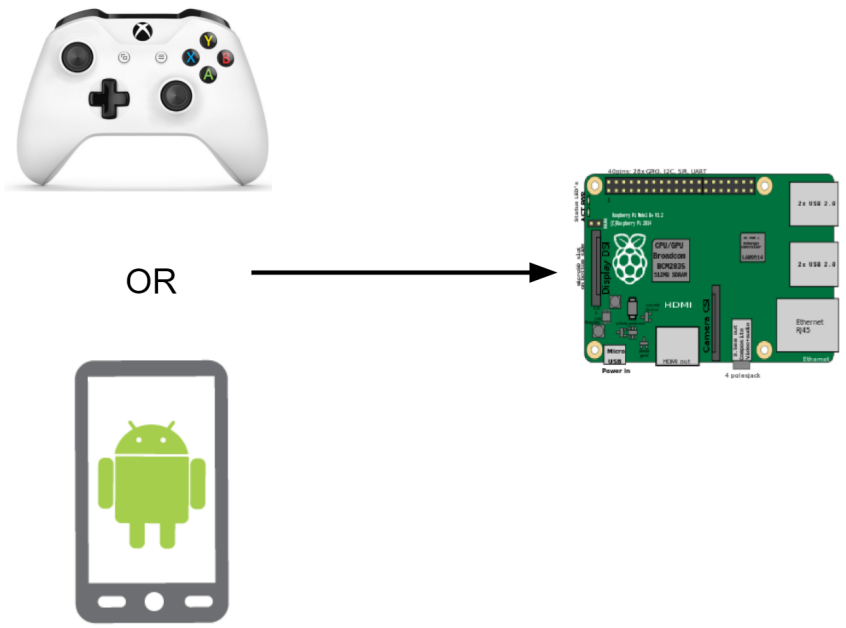

Figure 2: Control Methods

Any way that you can communicate to a Raspberry Pi is a way that you can control the rover, and we fully encourage you to find and develop other methods of remote control. For the methods above, if you decide to use an Android device you will need to follow some steps to get the app loaded on your phone. We provide the source code for the app so you can edit and change things on your own; the next section will walk you through getting the source

code for the Android app. If you wish to use the Xbox controller to control your rover, skip to step 2.2.

2.1 Android App Control

If you choose to control the rover from an Android App you'll need to setup Bluetooth communication between the Raspberry Pi and Android device. There are a few different methods of doing this; we'll be using a terminal interface.

2.1.1 Naming the Bluetooth Device

On the Raspberry Pi, create a file that will name the Bluetooth device your desired name[2]. You can do this by running the following in a terminal:

```
sudo nano /etc/machine-info
```

Name your device by replacing deviceName with whatever you want the name of your Raspberry Pi to be (we suggest something like "openSourceRover"):

```
PRETTY_HOSTNAME=deviceName
```

Now edit the /etc/bluetooth/main.conf file: Uncomment the "Name=" line and add your deviceName:

```
sudo nano /etc/bluetooth/main.conf
```

The first few lines of this file should look like Figure 3.

[2]This device name must also be updated in the Android App so that the App finds the correct device. Make sure to keep track of this name

```
[General]

# Default adapter name
# Defaults to 'BlueZ X.YZ'
Name = deviceName

# Default device class. Only the major and minor device class bits are
# considered. Defaults to '0x000000'.
#Class = 0x000100


```

Figure 3: bluetooth main.conf File

2.1.2 Pairing Bluetooth from Command Line

We're now ready to pair the Raspberry Pi with your Android device[3].

Have your Android device discoverable and searching for Bluetooth[4]. Begin by in a terminal starting bluetoothctl:

```
sudo bluetoothctl
```

You are now in the bluetoothctl interface. Run the following commands:

```
agent on
```

```
default-agent
```

```
scan on
```

Once your find your Android device, you'll need to pair to it using its address:

```
pair XX:XX:XX:XX:XX:XX
```

```
yes
```

[3]We followed the tutorial at https://www.cnet.com/how-to/how-to-setup-bluetooth-on-a-raspberry-pi-3/ to learn how to pair the Pi with an Android device.

[4]The process for making your phone or tablet discoverable and searching for bluetooth is different for different devices. Feel free to search the web for instructions on how to do this on your specific device.

You'll be prompted on the Android device asking to accept the pairing. Select Okay. Now that your Android is paired with the Pi, exit bluetoothctl by running the following on the Pi:

```
quit
```

2.1.3 Downloading the Android App

To begin, you need to download Android Studio which will allow you to put the source code onto the Android device as an app. You need to install Android Studio on a laptop or desktop computer, NOT the Raspberry Pi. Download and install Android Studio from

- https://developer.android.com/studio/index.html

Android Studio provides lots of support information and forums to help you install the software and then install apps on your Android. Consult the Android Studio documentation if you have any issues installing Android Studio or loading the app onto your Android device. In order to allow your Android device to accept the source code as an app, you'll need to enable USB debugging on the Android. This process varies across different version of phones and tablets. The best way to figure out how to turn on USB debugging for your device is to search the web for "android USB debugging" for your specific device and follow the steps that you find.

To get the source code for the Android app, you can navigate to the following URL on your laptop/desktop:

https://github.com/nasa-jpl/osr-android-app

and under the Clone or Download button click Download ZIP and extract the files in that manner.

If you wish, you can instead run the following command FROM YOUR LAPTOP/DESKTOP in a terminal from the location on your laptop/desktop where you want to have the source code:

```
git clone https://github.com/nasa-jpl/osr-android-app
```

Now, open up the source code that you just downloaded in Android Studio. In order for the App to communicate to the Raspberry Pi, you'll need to make sure it selects the correct Bluetooth device. On line 198 in the MainActivity.java file, change the variable *connection_name* to match whatever you named the Raspberry Pi in the previous section. Now hit run in the upper panel, and the Android app should build and load onto your Android device!

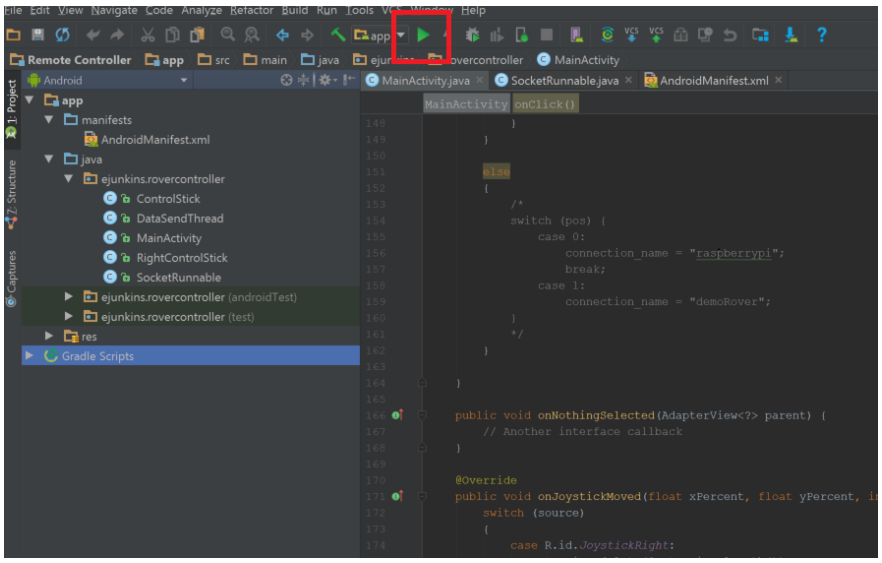

Figure 4

2.2 Init Script

We want the rover's code to automatically run whenever the pi boots up. We'll accomplish this by adding it to an init script, /etc/rc.local which is automatically run every time the Pi boots. In a terminal (now back on your Raspberry Pi), run:

```
sudo nano /etc/rc.local
```

The lines we need to add will be added after the "if" block, as shown in Figure 5. This example is to run with the LED screen and be controlled from the Android app:

```
sudo python /home/pi/osr/led/screen.py &

sleep 10

sudo python /home/pi/osr/rover/main.py -s &
```

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

sudo python /home/pi/os/led/screen.py &
sleep 10
sudo python /home/pi/os/rover/main.py -s -c x &

exit 0
```

Figure 5: /etc/rc.local script. Note that in this image, we set the `-c x` flag as an example of how to control the rover with the Xbox controller instead of from the Android app (see below).

The command line arguments that the main.py script accepts are the following:

-t : Testing mode - allows you to emulate a controller from the terminal for testing and control of the robot.

-s : Attempts to connect to the Unix Socket running in the LED screen script to send commands between the two processes

-c : Controller flag, tells which controller the rover should be listening for

b : Bluetooth app (default)

x : Xbox controller

If you wish to run your rover without the LED screen, delete line 20 and 21 from /etc/rc.local (those lines start the screen.py script) and remove the -s flag from the line that starts the main.py script.

2.3 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 6: [5]

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

Figure 6: 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
```

3 Test files

There are a few files that will help you test the robot before trying to fully drive it. Having all these tests working does not fully guarantee everything on the rover will work, but they should help with the testing and debugging process.

3.1 Serial Test

Under the 'rover' folder there is a script called *serialTest.py*. This will help you test the serial communication of the Raspberry Pi. To begin, use a jumper wire to connect the GPIO14 and GPIO15 pins.

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

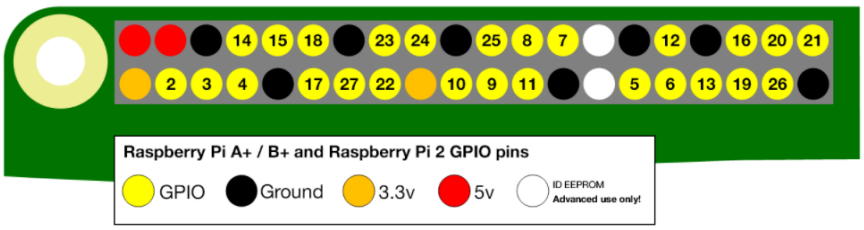

Figure 7: Raspberry Pi pinout

Then,run the script from a terminal:

```
python /home/pi/osr/rover/serialTest.py
```

If the serial communication is working properly, this script will let you know that it successfully read data over the serial pins. If it does not, verify that you have followed all the steps in this document to get serial communication working, and that you are connecting GPIO pins 14 and 15 together.