Open Source Rover

Software Install Instructions

Jet Propulsion Laboratory
California Institute of Technology

Contents

1 Setting up the Raspberry Pi 3 **3**

1.1 Getting the Pi online 3

1.1.1 Installing a new OS on the Pi 3

1.1.2 Installing packages 3

2 Control Method **5**

2.1 Android App Control 5

2.1.1 Naming the Bluetooth Device 6

2.1.2 Pairing Bluetooth from Commandline 6

2.1.3 Downloading the Android App 7

2.2 Init Script 9

2.3 Setting up serial communication 11

3 Test files **12**

3.1 Serial Test 12

3.2 Motor Test 13

1 Setting up the Raspberry Pi 3

In this section we'll go over setting up the Raspberry Pi and how to get all the code to run the rover going, as well as setting up the bluetooth pairing from the android device.

1.1 Getting the Pi online

1.1.1 Installing a new OS on the Pi

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.jpl.nasa.gov/ejunkins/osr_rover_code`

THIS LINK TO CHANGE UPON RELEASE

1.1.2 Installing packages

Below are the packages you will need to use Bluetooth communication through the raspberry pi. Open up a terminal on the Raspberry Pi and type the following commands:

`sudo apt-get update`

`sudo apt-get upgrade`

`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`

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

```
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 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 done this exit and save (ctrl + x, y, enter), and then reboot the Pi. Once 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 ways that are built into the code already to control the rover, one of which is a Bluetooth app run on Android devices, and the other is using a wireless Xbox controller.

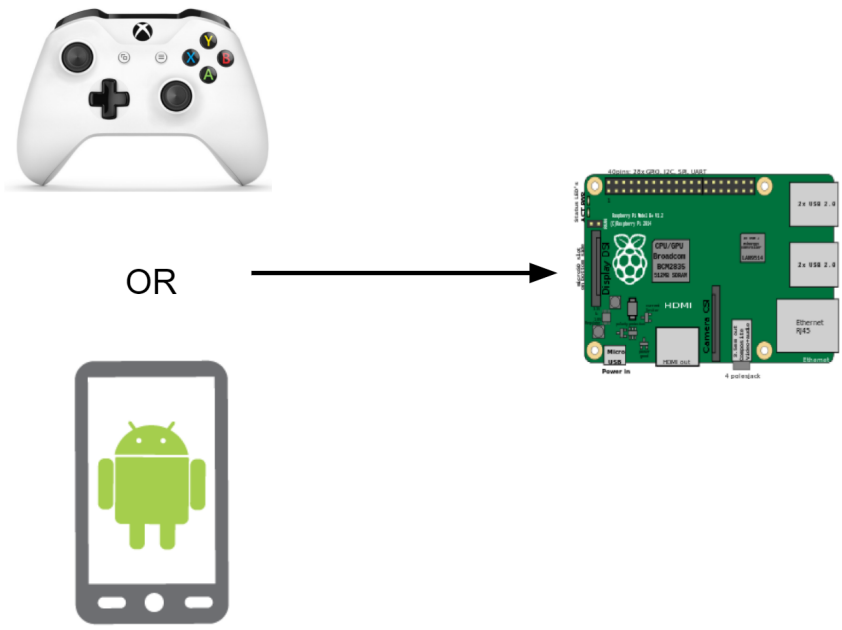

Figure 2: Control Methods

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

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 to do this in this tutorial.

2.1.1 Naming the Bluetooth Device

On the Raspberry Pi create a file that will name the Bluetooth device your desired name [2]. First on the RPi in a terminal type

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

Name your device by by replacing deviceName with want the name of your Raspberry Pi to be:

```
PRETTY_HOSTNAME=deviceName
```

Now edit the maain.conf file, Uncomment the "Name=" line and add your deviceName.

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

The first few lines should look like Figure 3.

```
1  [General]
2
3  # Default adapter name
4  # Defaults to 'BlueZ X.YZ'
5  Name = deviceName
6
7  # Default device class. Only the major and minor device class bits are
8  # considered. Defaults to '0x000000'.
9  #Class = 0x000100
10
11
```

Figure 3: bluetooth main.conf File

2.1.2 Pairing Bluetooth from Commandline

We're now ready to pair the device with your android device. We followed along with the following tutorial to learn how to do this:

- https://www.cnet.com/how-to/how-to-setup-bluetooth-on-a-raspberry-pi-3/

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

Have your android phone discover-able and searching for Bluetooth [3]. Begin by in a terminal starting the bluetoothctl

```
sudo bluetoothctl
```

Now in that bluetoothctl terminal do the following commands:

```
agent on
```

```
default-agent
```

```
scan on
```

Once your find your device you'll need to pair to it from its' Address.

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

```
yes
```

You'll be prompted on the phone asking to accept the pairing, select okay, and then on the Raspberry Pi now that you have paired the devices quit the bluetoothctl.

```
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 [4].

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

In order to allow the android device to be able to have source code loaded onto it you'll need to enable USB debugging, which is going to be different for a lot of different version

[3] For individual phones look up instructions on google if necessary to do this
[4] This is to be done on a laptop or desktop computer, not the Raspberry Pi

of phones. The best way to figure this out is to google "android USB debugging" for your specific device and follow the steps there. Android studio also provides lots of support information and forums to help getting through any problems and issues had on that end, consult those if there are issues.

To get the code for the app you can run the following command in a terminal at the location you desire it to be put:

```
git clone https://github.jpl.nasa.gov/ejunkins/osr_android_app
```

THIS LINK TO CHANGE UPON RELEASE TO FINAL LINK LOCATION

Or you can navigate to the URL at:

https://github.jpl.nasa.gov/ejunkins/osr_android_app

THIS LINK TO CHANGE UPON RELEASE TO FINAL LINK LOCATION

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

Now open up the source code 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 be equal to 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 device.

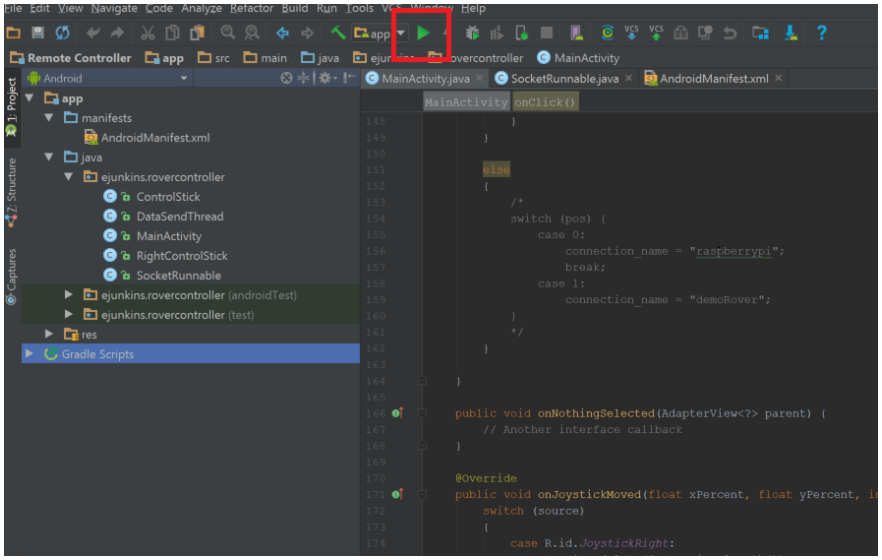

Figure 4

2.2 Init Script

We will want the code on the rover to run when the pi boots up, so we will add that to an init script, which on Raspberry Pi can be found at /etc/rc.local. In a terminal type

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

The following lines are added after the "if" block, as shown in Figure 5. This example is to run with the LED screen, and from an Xbox controller.

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

```
#!/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: Init Script

The command line args it 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 should be listened for

b : Bluetooth app (default)

x : Xbox controller

If you wish to run without the LED screen do not put the line starting the *screen.py* into the init script, and do not run with the *-s* flag.

2.3 Setting up serial communication

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

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

Do the following to setup/enable the serial communication for the RoboClaws:

```
sudo raspi-config
```

In this menu navigate and do the following commands:

- 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 rebooted open up a terminal again and look at the serial devices

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

Make sure that this says: serial0 -> ttyS0 . Now we have to edit the cmdline.txt file. In a terminal type

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

Edit **ONLY** the part with "console = ...", change it to be "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]:

[5] If it does not exactly match this it is fine, the import part is the "console=tty1" section

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

Figure 6: boot/cmdline.txt File

Here is a link to help with this process if necessary.

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

Once this is all done then 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. These working will not fully guarantee everything will work but is supposed to 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 between pin 14 and 15.

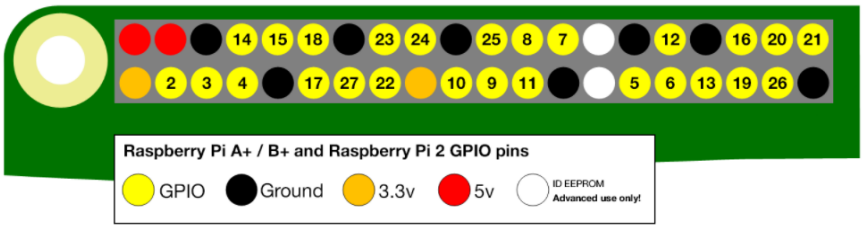

Figure 7: Raspberry Pi pinout

Then in a terminal run the script:

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

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

3.2 Motor Test

NEED INFORMATION ABOUT THE MOTOR TESTING SCRIP