Raspberry Pi Setup

These directions will lead you through setting up the Raspberry Pi as a node and running the rtl dongle on it.

NOTE: You probably already know this, but **$** before a command indicates running as a normal user, while **#** indicates running as root

Installing Raspbian

You will need:

- A Linux Computer
- A Raspberry Pi
- An SD card (at least 4 GB, Class 4)
- An SD card reader
- A Type A to Micro B USB cable
- Either:
    - An HDMI cable and a suitable display
    - **OR** an RCA cable and a suitable display
- A USB keyboard
- An Ethernet cable or Wifi dongle

SD Card Instructions:

1. Download the Raspbian zip file from http://www.raspberrypi.org/downloads
2. Change to the download directory and unzip the downloaded file. This should produce a single *.img file
3. If your SD card is plugged in, remove it. Run **$ df -h** to list the connected devices.
4. Insert the SD card and run **$ df -h** again to identify your SD card device (e.g. /dev/sdb1)
5. Run **$ umount /your/device** for each partition on the SD card (e.g. **$ umount /dev/sdb1; $ umount /dev/sdb2** if you have 2 partitions)
6. Run **# dd bs=4M if=/your/downloaded/*.img of=/your/device**. This will take a while. NOTE: Make sure that you have the right device! Otherwise, bad things will happen. ALSO, don't specify the partition number in **of** option. We want to write to the whole device. FINALLY, if bs=4M doesn't work, try bs=1M, which will take longer
7. Run **# sync**, which will allow you to safely remove the SD card from your computer

Raspbian Setup:

1. If the Raspberry Pi already has power, remove it. Insert the SD card, the keyboard, the display cable, the ethernet cable or Wifi dongle, and finally, the power cable.
2. The raspi-config screen should come up. Execute the following:
    a. Expand Filesystem
    b. Change User Password (optional, but recommended…)
    c. Advanced Options -> A4 SSH (If you would like to log in remotely)
    d. Any other options that you would like

    e. Select <Finish> and reboot
3. Once the reboot completes, login (username is pi, default password is raspberry, unless you changed it)
4. If you would like to login remotely, run **$ ifconfig** and note the ip address of the main internet interface. From the remote machine, execute **$ ssh pi@your.rpi.ip.address**

Installing REDHAWK

Raspbian is based on Debian (like Ubuntu, which we used on the BeagleBone Black). Axios Engineering has an Ubuntu port of the REDHAWK source code available which we will use to build and install the framework.

You will need:

- Your Raspberry Pi running Raspbian
- A Linux Computer (if you're logging in remotely, go ahead and do that now)

Installation Instructions:

1. First we need to install the dependencies of the framework. Begin by running **#apt-get update**. Next, run **# apt-get install libboost-all-dev**. Now for the rest of them: **# apt-get install vim build-essential openmpi-bin libopenmpi-dev python2.7 python2.7-dev uuid uuid-dev openjdk-7-jdk libtool autotools-dev autoconf automake python-omniorb omnievents omniidl omniidl-python omniorb omniorb-idl omniorb-nameserver libcos4-dev libomnievents-dev libomniorb4-dev python-numpy liblog4cxx10-dev xsdcxx**
2. Edit your /etc/hosts file to include the following (you will need to be root):
    a. 127.0.0.1   localhost   localhost.localdomain
3. Edit your /etc/omniORB.cfg file in the InitRef section (you will need to be root):
    a. Remove "DefaultInitRef = corbaloc::"
    b. Add "InitRef = NameService=corbaname::127.0.0.1"
    c. Add "InitRef = EventService=corbaloc::127.0.0.1:11169/omniEvents"
4. Restart omniNames and omniEvents:
    a. # /etc/init.d/omniorb4-nameserver stop
    b. # /etc/init.d/omniorb-eventservice stop
    c. # /etc/init.d/omniorb4-nameserver start
    d. # /etc/init.d/omniorb-eventservice start
5. Create a symbolic link for omniidl:
    a. # ln -s /usr/lib/omniidl/omniidl /usr/lib/python2.7/dist-packages/omniidl
6. From your home directory, run **$ mkdir redhawk** then **$ cd redhawk**
7. Get, build, and install the core framework (this will take a LONG time, ~ 4.5 h):
    a. $ git clone git://github.com/Axios-Engineering/framework-core.git
    b. $ cd framework-core
    c. $ git checkout -b ubuntu origin/ubuntu
    d. $ cd src

    e. **$ ./reconf; ./configure --disable-java --with-ossie=/usr/local/redhawk/core --with-sdr=/var/redhawk/sdr; make -j 2; sudo make install**
    f. **$ cd etc/profile.d/** (Note that this is in your current directory, not /etc/profile.d)
    g. **$ chmod +x ***
    h. **# cp * /etc/profile.d/**
    i. **$ cd ~**
8. Add the following to your .bashrc file:
    a. export OSSIEHOME=/usr/local/redhawk/core
    b. export SDRROOT=/var/redhawk/sdr
    c. export PYTHONPATH=${OSSIEHOME}/lib/python
    d. export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-armhf
    e. /etc/profile.d/redhawk.sh
    f. /etc/profile.d/redhawk-sdrroot.sh
9. Run **$ source .bashrc** to update your environment variables
10. Get, build, and install Bulkio Interfaces:
    a. **$ cd redhawk**
    b. **$ git clone git://github.com/Axios-Engineering/framework-bulkioInterfaces.git**
    c. **$ cd framework-bulkioInterfaces**
    d. **$ git checkout -b ubuntu origin/ubuntu**
    e. **$ ./reconf; ./configure --disable-java; make -j 2; sudo make install; cd ../**
11. Get, build, and install the GPP:
    a. **$ git clone git://github.com/Axios-Engineering/framework-GPP.git**
    b. **$ cd framework-GPP**
    c. **$ git checkout -b ubuntu origin**
    d. **$ cd python**
    e. **$ ./reconf; ./configure; sudo make install**
12. Take ownership of SDRROOT with **# sudo chown -R <yourusername>.<yourusername> $SDRROOT**

Setting Up Your Domain and Device Managers

Before we can set the Raspberry Pi up as a node, it would be nice to run it as a Domain Manager to make sure things have gone well so far. Then you can connect the Device Manager to your local Domain Manager as an intermediate test. Finally, you can connect your Device Manager to a remote Domain Manager.

You will need:

- Your Raspberry Pi running Raspbian
- A Linux Computer (if you're logging in remotely, go ahead and do that now)
- A Remote Domain Manager (it's fine if it is the same as the computer above)

Domain Manager Instructions:

1. **$ cd $SDRROOT/dom/domain**
2. Make a copy of the template file with **$ cp DomainManager.dmd.xml.template DomainManager.dmd.xml**
3. Run **$ uuid** and copy the results to use in the next step
4. Edit the DomainManager.dmd.xml file so that:
    a. @UUID@ is replaced with the uuid you generated in the last step
    b. @NAME@ is replaced with REDHAWK_DEV (or another name of your choosing. Note that if another running Domain Manager has the same name, you will not be able to start this domain)
    c. @DESCRIPTION@ is replaced with some information about your Domain Manager (it doesn't really matter what you put here)
5. **$ cd $SDRROOT/dom/mgr**
6. Edit the DomainManager.spd.xml file:
    a. Remove the two processor tags
    b. Insert a new processor tag with the name armv6l (<processor name="armv6l"/>)
7. **$ nodeBooter -D** (At this point, the domain manager should start without any problems)
8. **$ <Ctrl-C>**

Device Manager Instructions:

1. **$ cd $SDRROOT/dev/devices/GPP**
2. Edit the GPP.spd.xml file:
    a. Remove the two processor tags
    b. Insert a new processor tag with the name armv6l (<processor name="armv6l"/>)
3. **$ cd python**
4. Edit the nodeconfig.py file:
    a. Go to line 89
    b. Replace x86 with armv6l
    c. Replace any instances of uuidgen with uuid (There should be eight instances)
5. Edit the GPP.py file:
    a. Go to line 509
    b. Make the p in processor uppercase
6. **$ python nodeconfig.py --domainname <the name you gave to the domain>**
7. **$ cd $SDRROOT/dev/mgr**
8. Run **$ uuid** and copy the results to use in the next step
9. Edit the DeviceManager.spd.xml file:
    a. Copy the x86_64 implementation (from <implementation…> to </implementation>)
    b. Paste this after the x86_64 implementation
    c. Replace your copy's id with the uuid you generated in Step 8
    d. Replace any instances of x86_64 with armv6l (Only under your implementation, there should be three)
10. **$ cp DeviceManager.Linux.x86_64.prf.xml DeviceManager.Linux.armv6l.prf.xml**
11. Edit the DeviceManager.Linux.armv6l.prf.xml file:

    a. Replace x86_64 with armv6l (There should be only one instance)
12. **$ nodeBooter -D &**
13. **$ nodeBooter -d /nodes/DevMgr_raspberrypi/DeviceManager.dcd.xml**
14. **$ <Ctrl-C>**
15. **$ jobs -l**
16. **$ kill -9 <pid corresponding to domain manager from Step 15>**

Remote Domain Manager Instructions:

1. Edit the /etc/omniORB.cfg file (you will need to be root):
    a. On the InitRef lines you added before building the framework, replace the localhost ip (127.0.0.1) with that of the machine running the remote domain manager
2. Edit the /etc/hosts file (you will need to be root):
    a. Add a line with your network ip address (e.g. 192.168.1.x) and the hostname of your device (probably raspberrypi), separated by a tab
    b. Add a line with your remote domain manager's ip address and the hostname of your remote domain manager, separated by a tab
3. *On the device running the domain manager,* run **# iptables -I INPUT -s <your raspberry pi ip> -j ACCEPT**
4. *Still on the device running the domain manager,* run **$ nodeBooter -D**
5. *Back on the raspberry pi,* restart omniNames and omniEvents:
    i. **# /etc/init.d/omniorb-eventservice stop**
    ii. **# /etc/init.d/omniorb4-nameserver stop**
    iii. **# /etc/init.d/omniorb4-nameserver start**
    iv. **# /etc/init.d/omniorb-eventservice start**
6. *Still on the raspberry pi, run* **$ nodeBooter -d /nodes/DevMgr_raspberrypi/DeviceManager.dcd.xml**

Running a Waveform

Now that your raspberry pi is set up as a node and connected to your domain manager, you should be able to run a waveform and execute components on the raspberry pi's GPP.

You will need:

- Your Raspberry Pi running Raspbian and REDHAWK
- A Linux Computer (if you're logging in remotely, go ahead and do that now)
- A Remote Domain Manager (it's fine if it is the same as the computer above)

Create a simple component:

1. In your IDE, create a new SCA Component Project. Call it MultiplyByConst (the name may be different, simply replace it with a different one in later steps)
2. Add a provides port called dataIn of type dataDouble and a uses port called dataOut of type DataDouble

3. Add a Simple property called multiple of type double with a default value of 1.0
4. Generate the project
5. Replace the existing code in the serviceFunction with the following:

```
BULKIO_dataDouble_In_i::dataTransfer * input = dataIn->getPacket(-1);

if (not input) {
        return NOOP;
}

if (input->sriChanged) {
        dataOut->pushSRI(input->SRI);
}

std::vector<double> output(input->dataBuffer.size());

for (unsigned int i = 0; i < output.size(); i++) {
        output[i] = multiple * input->dataBuffer[i];
}

dataOut->pushPacket(output, input->T, input->EOS, input->streamID);

return NORMAL;
```

6. Save the file, right click the project in the Project Explorer, and select Build Project. Once the build finishes, drag the MultiplyByConst project to "Target SDR"

Create a simple waveform:

1. Create a new SCA Waveform Project. Call it RaspberryPiTest (the name may be different , simply replace it with a different one in later steps)
2. Drop a SigGen onto the waveform. Leave the default properties.
3. Drop a MultiplyByConst onto the waveform. Set the multiple property to 10.
4. Save the file and drag the RaspberryPiTest project to "Target SDR"

Launch the waveform on your local machine:

1. If a Device Manager is not already running, launch it with **$ nodeBooter -d /nodes/<Your Local Node Name>/DeviceManager.dcd.xml**
2. If not already connected to the Domain Manager in the IDE, select the green plus in the SCA Explorer View. When the Dialog Box opens, enter the name of your Domain Manager in the Domain Name textbox (Probably REDHAWK_DEV)
3. Right click the Domain Manager in the SCA Explorer View and select "Launch Waveform…"
4. Select RaspberryPiTest from the list and click Finish

5. Once the waveform has launched, verify that the multiple property of MultiplyByConst is still 10. Run the waveform.
6. Right click the output port of SigGen and select Plot Port Data. Note the magnitude of the wave.
7. Right click the output port of MultiplyByConst and select Plot Port Data. The magnitude of the wave should be 10 times that of the magnitude for the wave coming from SigGen.
8. Stop and Release the waveform.

Launch the waveform on two devices:

1. In order to execute the MultiplyByConst component on the Raspberry Pi Device Manager, we need an armv6l executable. The easiest way to do this is to generate a second implementation of the MultiplyByConst.
2. Open the MultiplyByConst.spd.xml file and select the Implementation tab. Click the Add… button and select C++ as your programming language. Leave the other options as their default values. Click the Finish button.
3. Make sure that the cpp_impl2 Implementation is selected and navigate to the Dependencies section on the right. Remove the x86 and x86_64 Processors and Add an armv6l Processor. Click the Generate button. When the dialog displays, select Generate All and click Ok.
4. At this point, the IDE probably changed this new Implementation to be your automatic build directory. To fix this, right click the MultiplyByConst, select Properties, then C/C++ Build. Under the Builder Settings tab, replace any instance of cpp2 with cpp and click OK.
5. Copy the contents of cpp/MultiplyByConst.cpp to cpp2/MultiplyByConst.cpp
6. Open a terminal in your workspace and execute the command **$ scp -r MultiplyByConst pi@<Your.Rpi.ip>:~**
7. In the home directory of your Raspberry Pi, run **$ cd MultiplyByConst/cpp2.**
8. Run **$ ./reconf; ./configure; make**
9. Once the build has finished, back in your workspace, run **$ cd MultiplyByConst/cpp2**. Now copy the generated executable over with **$ scp pi@<Your.Rpi.pi>:~/MultiplyByConst/cpp2/MultiplyByConst .**
10. Drag the MultiplyByConst project to “Target SDR”
11. If your Raspberry Pi Device Manager is not already connected to your Domain Manager, do so now.
12. Right click the Domain Manager in the SCA Explorer View and select Launch Waveform…
13. Select the RaspberryPiTest waveform and click Next twice
14. In the table, select your Raspberry Pi Device Manager as the Device for your MultiplyByConst component. Click Finish.
15. When asked if you would like to save this configuration, choose No (In 1.8.4, the IDE didn’t save the configuration regardless of your answer. Not sure about 1.9)
16. Click the run button and verify that the MultiplyByConst component is indeed running on your Raspberry Pi. This can be done by opening a terminal on the Raspberry Pi and

running top.

17. Stop and Release the waveform.