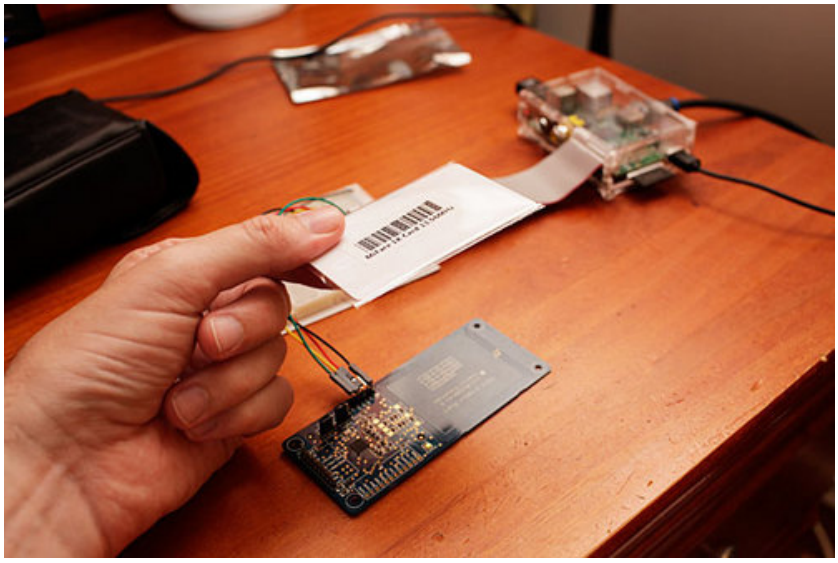
adafruit learning system

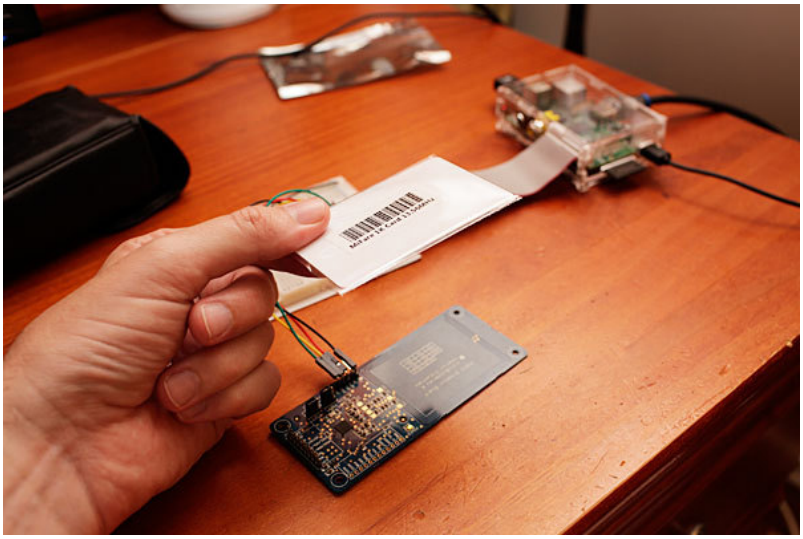
Adafruit NFC/RFID on Raspberry Pi

Created by Kevin Townsend



Guide Contents

Overview



Interested in adding some NFC fun and excitement to your Raspberry Pi? You're in luck!

One of the big advantages of Linux is that it includes a large number of stacks that have been developed by the open source community, and you get to take advantage of all that hard work simply by using or installing the right library.

NFC is no exception here, with libnfc (http://adafru.it/aN2) having been around for a quite some time -- in fact, it's the original reason the NFC Breakout was developed!

To get libnfc playing well with your Pi and your Adafruit NFC breakout you'll need to make some minor modification to your vanilla Wheezy distribution, and one small change to the latest NFC code (1.6.0-rc1 as of this writing), but it's pretty painless, and this tutorial will show you everything you need to do to start writing your own NFC-enabled apps on the Pi!

Freeing UART on the Pi

The easiest way to use libnfc with the Adafruit NFC Breakout is via UART, since it's well-supported by libnfc out of the box. Unfortunately the UART port on the Pi is already dedicated to other purposes, and needs to be freed up for libnfc.

The following steps (based on a clean 2012-07-15-wheezy-raspbian install but should also work with Adafruit's Occidentalis) should free UART up for us:

Step One: Edit /boot/cmdline.txt

From the command prompt enter the following command:

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

And change:

dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 $

to:

dwc_otg.lpm_enable=0 console=tty1 $

Step Two: Edit /etc/inittab

From the command prompt enter the following command:

```
$ sudo nano /etc/inittab
```

And change:

#Spawn a getty on Raspberry Pi serial line
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

to:

#Spawn a getty on Raspberry Pi serial line
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

Step Three: Reboot your Pi

After rebooting the Pi for the above changes to take effect, you can proceed with building and testing libnfc ...

Building libnfc

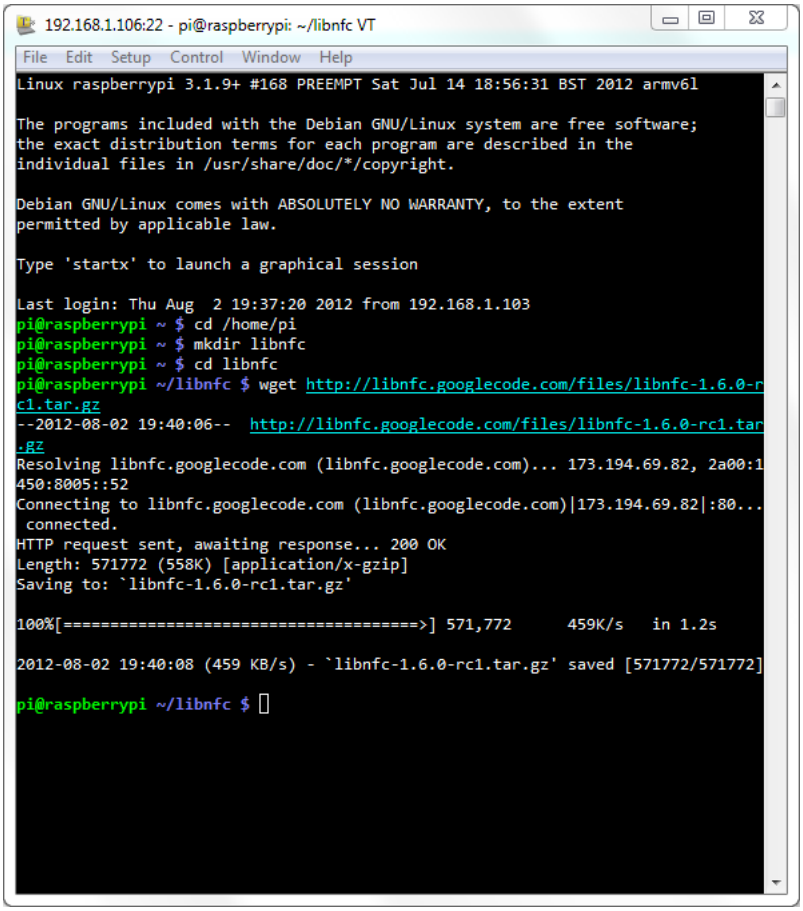
Step One: Download libnfc

Before you can do anything, you will need to get a copy of libnfc. Make sure you have an Ethernet cable connected to your Pi, and run the following commands to get libnfc 1.6.0-rc1

For reasons we don't understand it does not work with libnfc-1.7.0-rc4 - but we have tested it with 1.6.0-rc1 as above!

```
$ cd /home/pi
$ mkdir libnfc
$ cd libnfc
$ wget http://libnfc.googlecode.com/files/libnfc-1.6.0-rc1.tar.gz
$ tar -xvzf libnfc-1.6.0-rc1.tar.gz
$ cd libnfc-1.6.0-rc1
```

You should see something similar to the following:



Step Two: Modify libnfc/buses/uart_posix.c

uart_posix.c needs to be modified to make libnfc test for devices named ttyAMA*, the name of the default UART device on the Pi. To include ttyAMA in the serial autoprobe function, enter the following command:

```
$ nano libnfc/buses/uart_posix.c
```

And change:

```
#  elif defined (__linux__)
char *serial_ports_device_radix[] = { "ttyUSB", "ttyS", NULL };
#  else
```

to:

```
#  elif defined (__linux__)
char *serial_ports_device_radix[] = { "ttyUSB", "ttyS", "ttyAMA", NULL };
#  else
```

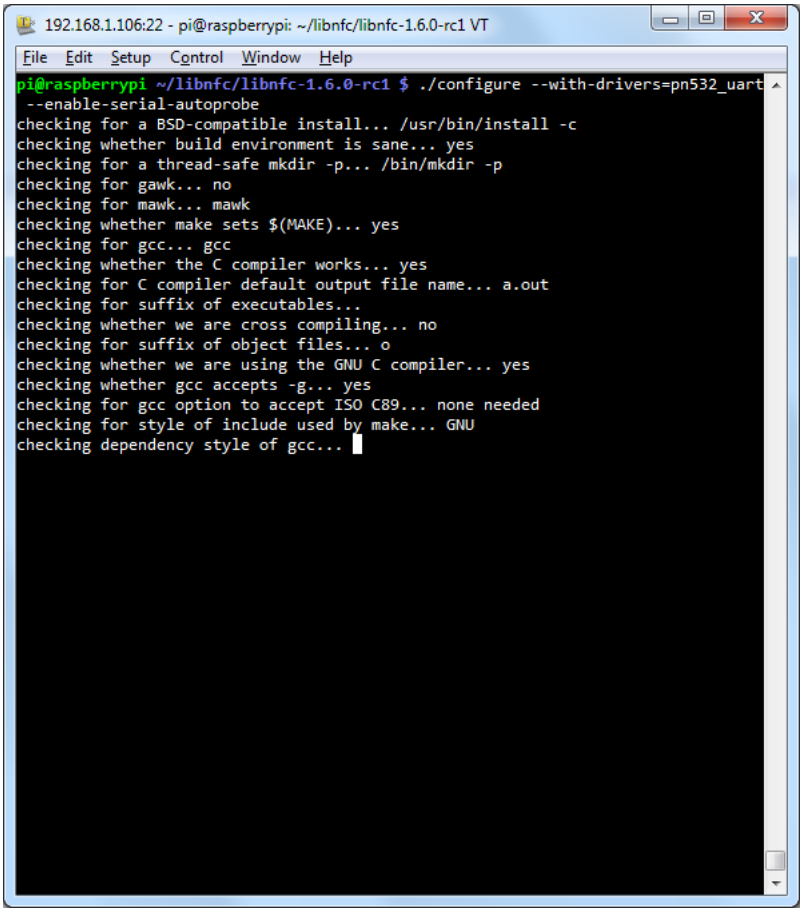
Step Three: Configure libnfc

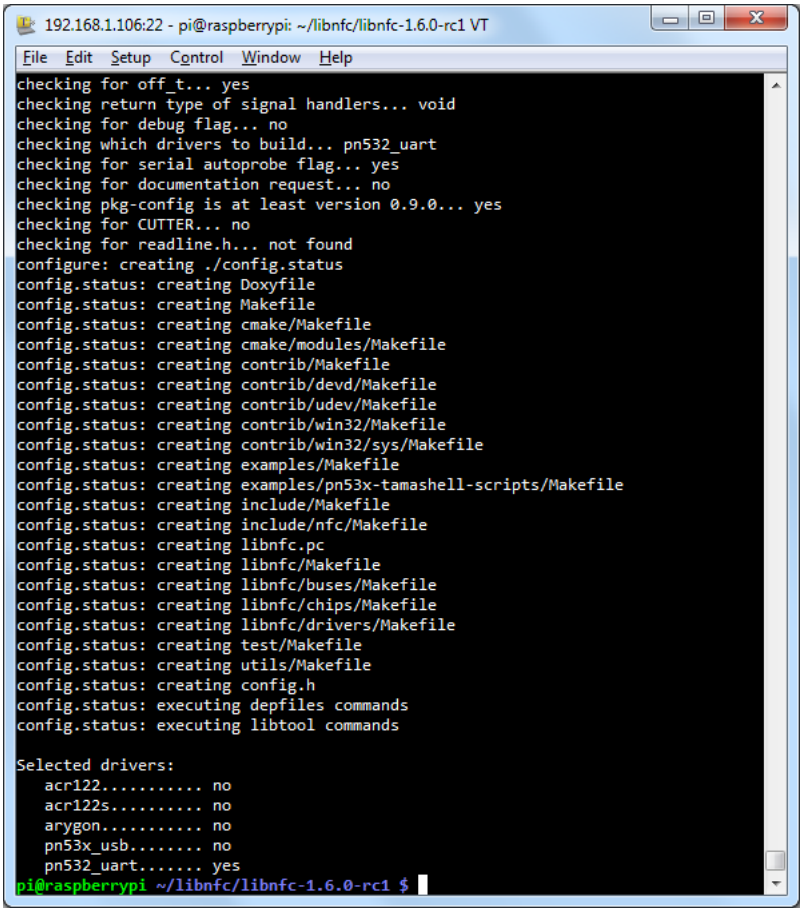
Before libnfc can be built, it needs to be cofigured for the target system and based on some parameters specific the NFC device you have connected.

Run the following command to configure libnfc to use UART and the PN532:

```
$ ./configure --with-drivers=pn532_uart --enable-serial-autoprobe
```

This should give you the following screens, during and after the configuration process:


192.168.1.106:22 - pi@raspberrypi: ~/libnfc/libnfc-1.6.0-rc1 VT
File Edit Setup Control Window Help
pi@raspberrypi ~/libnfc/libnfc-1.6.0-rc1 $ ./configure --with-drivers=pn532_uart
--enable-serial-autoprobe
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for style of include used by make... GNU
checking dependency style of gcc...

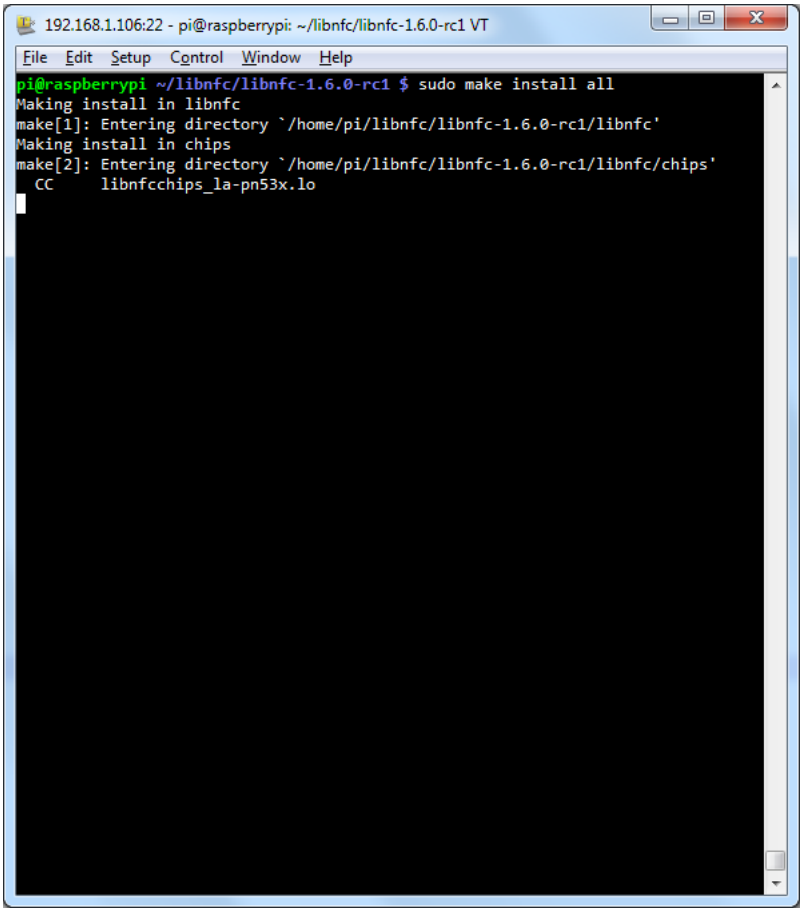


Step Four: Build!

To build libnfc, you simply need to enter the following commands:

```
$ sudo make clean
$ sudo make install all
```

Which should start the (slowish!) build process as follows:



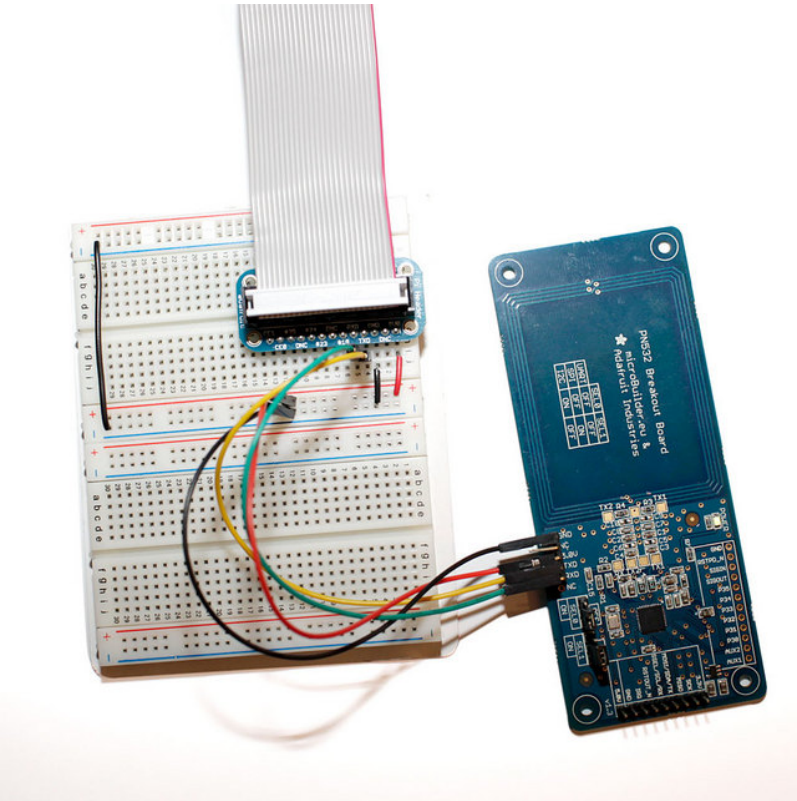
Once the build process is complete, you're ready to go on to testing the library on actual HW ...

Testing it Out

Hooking Everything Up

The Adafruit NFC Breakout (http://adafru.it/364) board is much more appropriate with the Pi than the NFC Shield (http://adafru.it/789), since the breakout doesn't have 5V level shifting (which means you won't accidentally damage your Pi!), and you have easier access to the bus select pins, etc.

If it isn't already hooked up, you can connect your breakout now using a convenient Pi Cobbler (http://adafru.it/914), following the image below:



Note: Make sure that the **SEL0** and **SEL1** jumpers on the NFC breakout are set to **OFF**, which will cause the PN532 to boot into UART mode (rather than SPI and I2C, which aren't currently supported by libnfc). You will need to reset the breakout after changing these pins, which you can do by cycling the power pin.

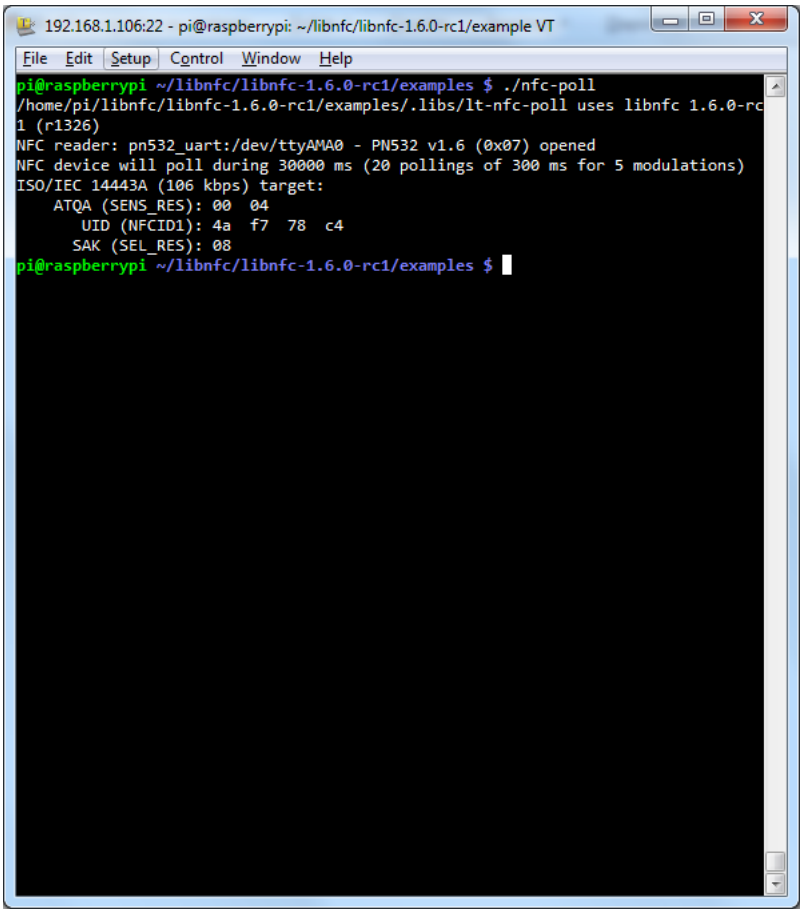
Use the 5V supply on the Pi Cobbler, and the 5V input on the FTDI header rather than the 3.3V supply, since the 3.3V supply is used by the core on the rasberry Pi and you don't want to pull sharp, heavy loads from it, like when you first enable and charge the near field.

Read an ISO14443-A (Mifare, etc.) Card with nfc-poll

With libnfc built and properly configure, you can go back to the command-line, place a card on the reader, and run the following command to get the tags unique ID:

```
$ cd ../examples
$ ./nfc-poll
```

Which should results in the following:



That's it! From here, you can explore some of the other examples in the 'examples' folder, and figure out how to get started writing your own applications based on libnfc! Be sure to have a look at the libnfc project page (http://adafru.it/aN3) which also contains a useful and active forum.