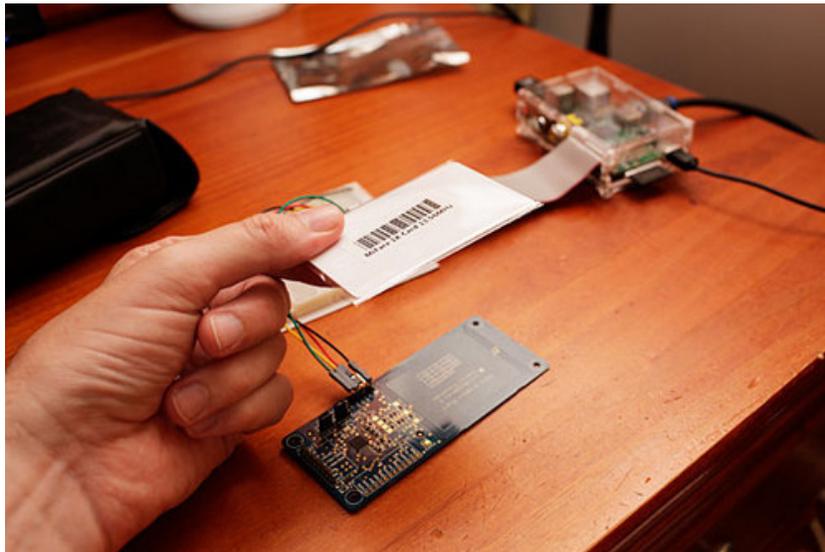


## Adafruit NFC/RFID on Raspberry Pi

Created by Kevin Townsend

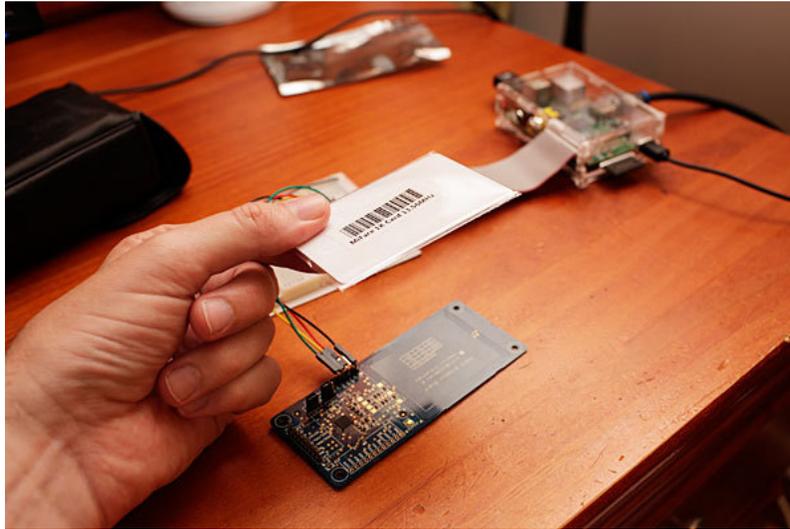


## Guide Contents

Guide Contents	2
Overview	3
Freeing UART on the Pi	4
Step One: Edit /boot/cmdline.txt	4
Step Two: Edit /etc/inittab	4
Step Three: Reboot your Pi	4
Building libnfc	6
Step One: Download libnfc	6
Step Two: Modify libnfc/buses/uart_posix.c	7
Step Three: Configure libnfc	7
Step Four: Build!	9
Testing it Out	11
Hooking Everything Up	11
Read an ISO14443-A (Mifare, etc.) Card with nfc-poll	12

## 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) (<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

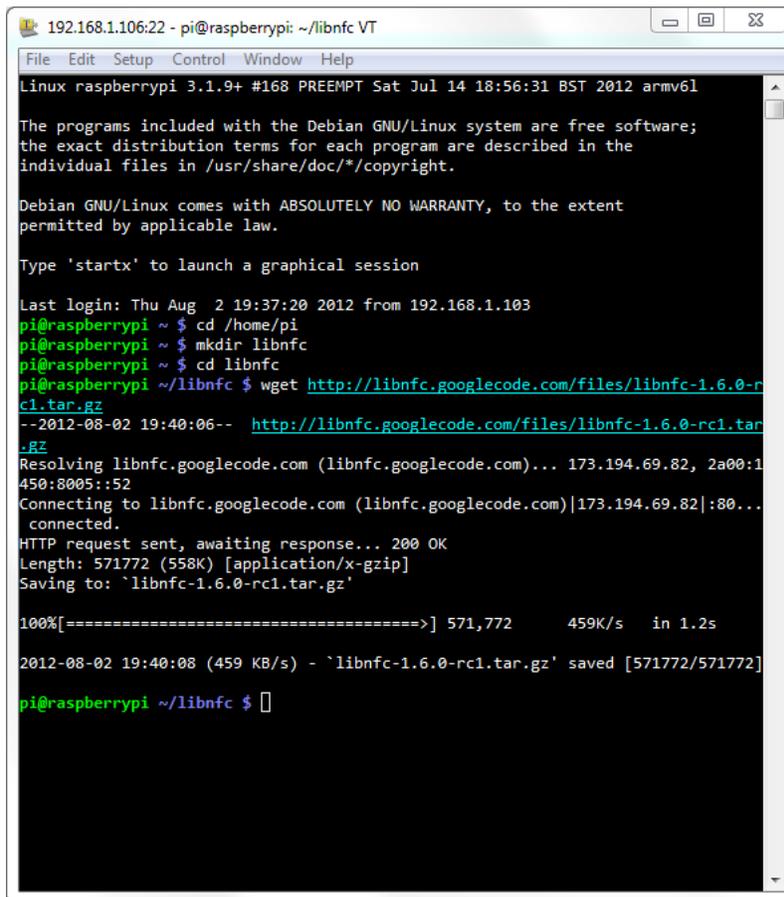
### Step One: Download libnfc

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:



```
192.168.1.106:22 - pi@raspberrypi: ~/libnfc VT
File Edit Setup Control Window Help
Linux raspberrypi 3.1.9+ #168 PREEMPT Sat Jul 14 18:56:31 BST 2012 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Type 'startx' to launch a graphical session

Last login: Thu Aug  2 19:37:20 2012 from 192.168.1.103
pi@raspberrypi ~ $ cd /home/pi
pi@raspberrypi ~ $ mkdir libnfc
pi@raspberrypi ~ $ cd libnfc
pi@raspberrypi ~/libnfc $ wget http://libnfc.googlecode.com/files/libnfc-1.6.0-rc1.tar.gz
--2012-08-02 19:40:06-- http://libnfc.googlecode.com/files/libnfc-1.6.0-rc1.tar.gz
Resolving libnfc.googlecode.com (libnfc.googlecode.com)... 173.194.69.82, 2a00:1450:8005::52
Connecting to libnfc.googlecode.com (libnfc.googlecode.com)|173.194.69.82|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 571772 (558K) [application/x-gzip]
Saving to: `libnfc-1.6.0-rc1.tar.gz'

100%[=====>] 571,772    459K/s  in 1.2s

2012-08-02 19:40:08 (459 KB/s) - `libnfc-1.6.0-rc1.tar.gz' saved [571772/571772]

pi@raspberrypi ~/libnfc $
```

## 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 configured 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... GNU
```

```
192.168.1.106:22 - pi@raspberrypi: ~/libnfc/libnfc-1.6.0-rc1 VT
File Edit Setup Control Window Help
checking for off_t... yes
checking return type of signal handlers... void
checking for debug flag... no
checking which drivers to build... pn532_uart
checking for serial autoprobe flag... yes
checking for documentation request... no
checking pkg-config is at least version 0.9.0... yes
checking for CUTTER... no
checking for readline.h... not found
configure: creating ./config.status
config.status: creating Doxyfile
config.status: creating Makefile
config.status: creating cmake/Makefile
config.status: creating cmake/modules/Makefile
config.status: creating contrib/Makefile
config.status: creating contrib/devd/Makefile
config.status: creating contrib/udev/Makefile
config.status: creating contrib/win32/Makefile
config.status: creating contrib/win32/sys/Makefile
config.status: creating examples/Makefile
config.status: creating examples/pn53x-tamashell-scripts/Makefile
config.status: creating include/Makefile
config.status: creating include/nfc/Makefile
config.status: creating libnfc.pc
config.status: creating libnfc/Makefile
config.status: creating libnfc/buses/Makefile
config.status: creating libnfc/chips/Makefile
config.status: creating libnfc/drivers/Makefile
config.status: creating test/Makefile
config.status: creating utils/Makefile
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing libtool commands

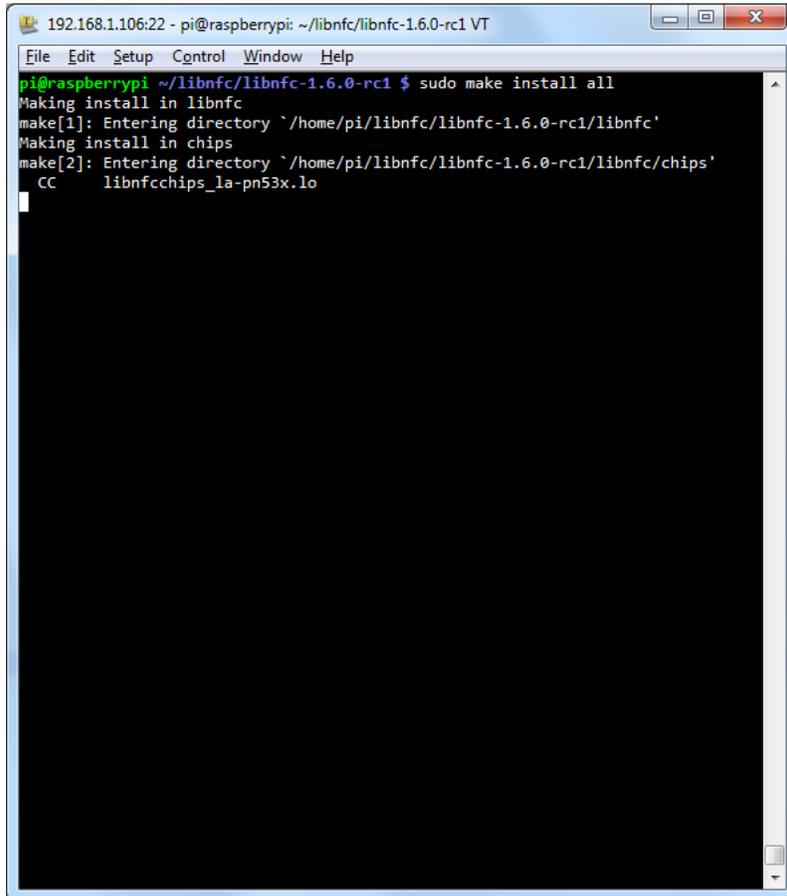
Selected drivers:
acr122..... no
acr122s..... no
arygon..... no
pn53x_usb..... no
pn532_uart..... yes
pi@raspberrypi ~/libnfc/libnfc-1.6.0-rc1 $
```

## 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:



```
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 $ sudo make install all
Making install in libnfc
make[1]: Entering directory `/home/pi/libnfc/libnfc-1.6.0-rc1/libnfc'
Making install in chips
make[2]: Entering directory `/home/pi/libnfc/libnfc-1.6.0-rc1/libnfc/chips'
CC      libnfcchips_la-pn53x.lo
```

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

## Testing it Out

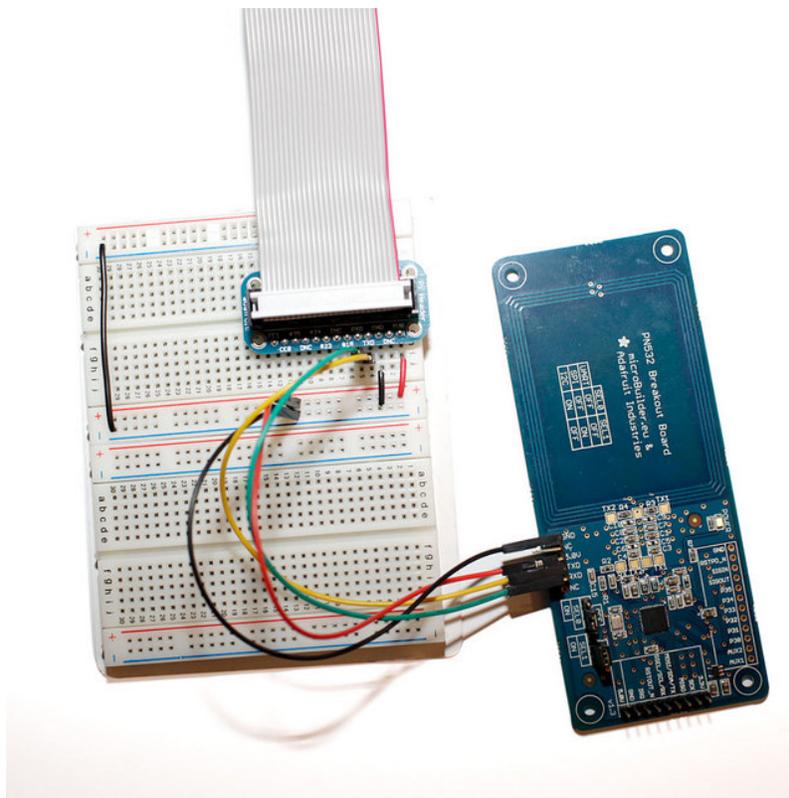
---

### Hooking Everything Up

---

The Adafruit [NFC Breakout](http://adafru.it/364) (<http://adafru.it/364>) board is much more appropriate with the Pi than the [NFC Shield](http://adafru.it/789) (<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) (<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.

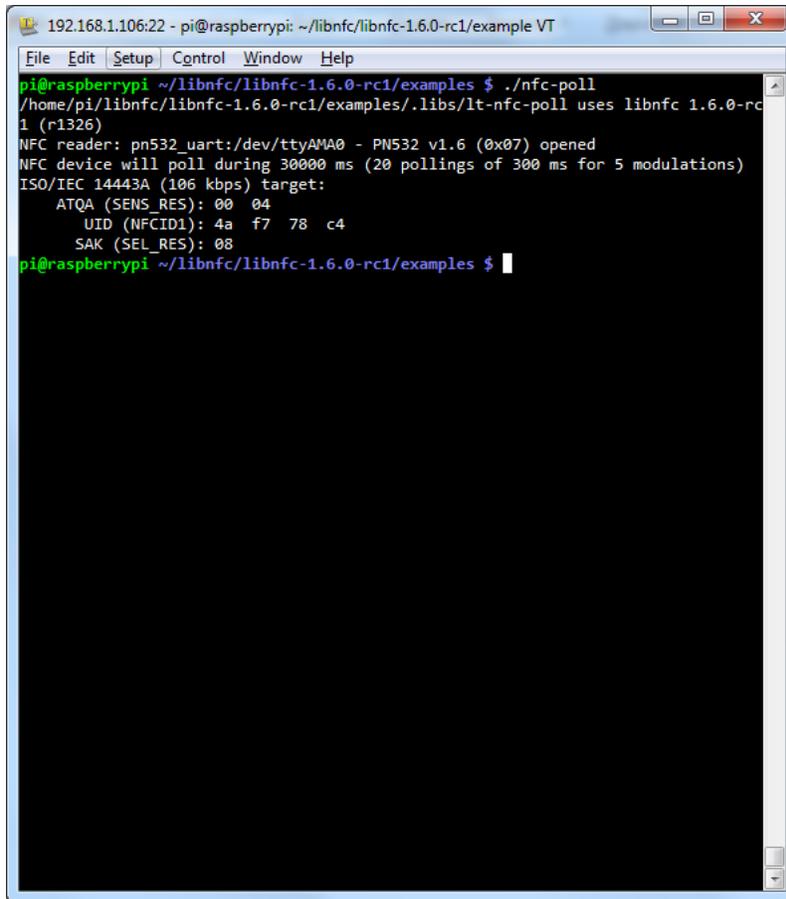
Use the 5V supply on the Pi Cobbler, and the 5V input on the FTDI header rather than the 3.3V supply, since the 3.3V supply is used by the core on the raspberry Pi and you don't want to pull sharp, heavy loads from it, like when you first enable and charge the near field.

## 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:

A terminal window titled "192.168.1.106:22 - pi@raspberrypi: ~/libnfc/libnfc-1.6.0-rc1/example VT" showing the execution of the nfc-poll command. The output displays the NFC reader details, the polling duration, and the unique ID (UID) of the card: 4a f7 78 c4.

```
pi@raspberrypi ~/libnfc/libnfc-1.6.0-rc1/examples $ ./nfc-poll
/home/pi/libnfc/libnfc-1.6.0-rc1/examples/.libs/lt-nfc-poll uses libnfc 1.6.0-rc1 (r1326)
NFC reader: pn532_uart:/dev/ttyAMA0 - PN532 v1.6 (0x07) opened
NFC device will poll during 30000 ms (20 pollings of 300 ms for 5 modulations)
ISO/IEC 14443A (106 kbps) target:
  ATQA (SENS_RES): 00 04
  UID (NFCID1): 4a f7 78 c4
  SAK (SEL_RES): 08
pi@raspberrypi ~/libnfc/libnfc-1.6.0-rc1/examples $
```

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 \(http://adafru.it/aN3\)](http://adafru.it/aN3) which also contains a useful and active forum.