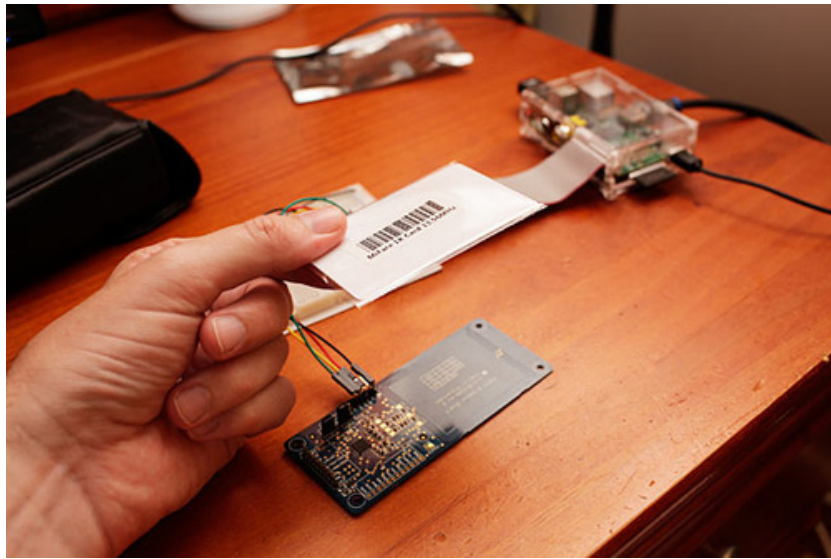




Adafruit NFC/RFID on Raspberry Pi

Created by Kevin Townsend

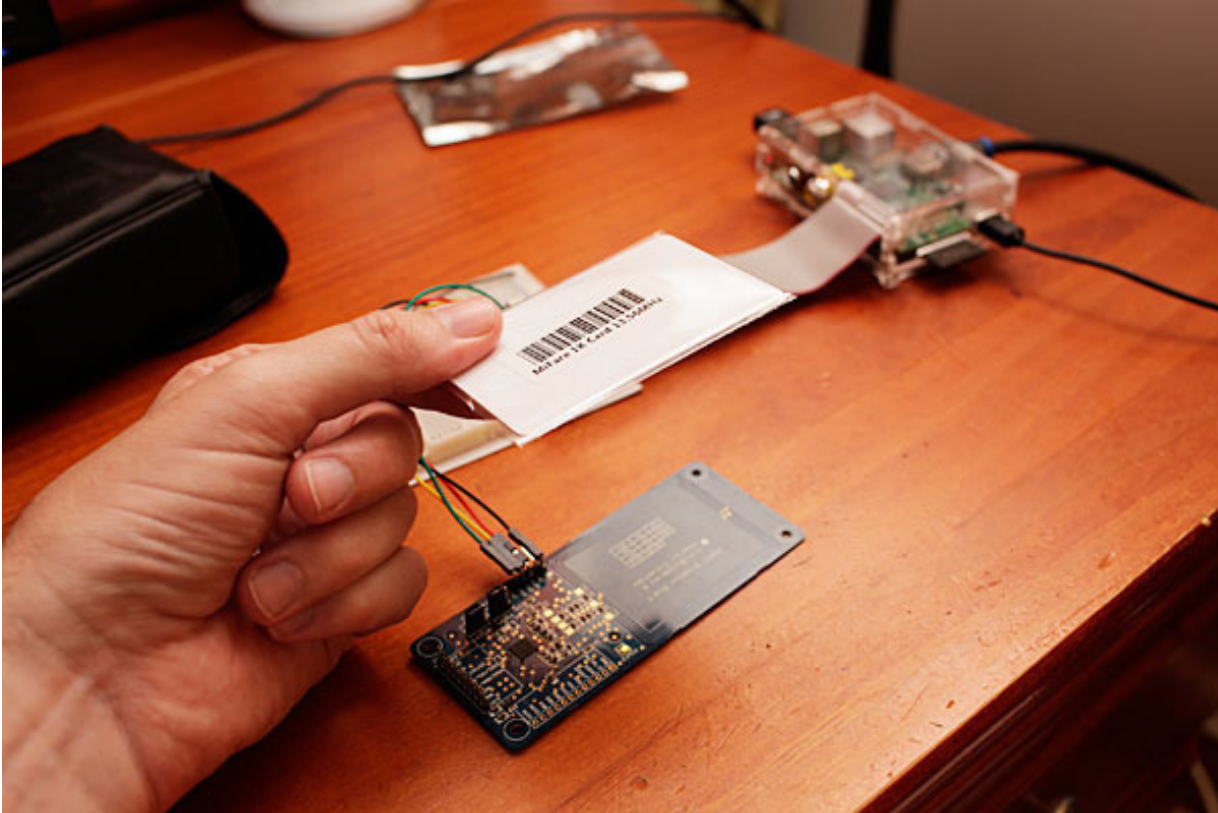


Last updated on 2014-04-25 04:45:19 PM EDT

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: Setup libnfc for the Pi	6
Step Three: Run Config	8
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://adafruit.it/aN2) (<http://adafruit.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 (1.7.0-rc7 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
```

(note that new # at the beginning of the second line!)

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 1.7.0

UPDATE 25 April 2014: Google has changed the way direct downloads work on Google Code. To download the .zip package, go to the following URL and select 'download .zip': <https://code.google.com/p/libnfc/source/browse/?name=libnfc-1.7.0>

```
$ cd /home/pi
$ mkdir libnfc
$ cd libnfc
$ wget http://libnfc.googlecode.com/files/libnfc-1.7.0.tar.gz
$ tar -xvzf libnfc-1.7.0.tar.gz
```

You should see something similar to the following:

```
pi@raspberrypi ~ $ mkdir libnfc
pi@raspberrypi ~ $ cd libnfc
pi@raspberrypi ~/libnfc $ wget http://libnfc.googlecode.com/files/libnfc-1.7.0-rc7.tar.gz
--2013-06-17 14:42:06-- http://libnfc.googlecode.com/files/libnfc-1.7.0-rc7.tar.gz
Resolving libnfc.googlecode.com (libnfc.googlecode.com)... 173.194.66.82, 2a00:1450:400c:c0
3::52
Connecting to libnfc.googlecode.com (libnfc.googlecode.com)|173.194.66.82|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 608357 (594K) [application/x-gzip]
Saving to: `libnfc-1.7.0-rc7.tar.gz'

100%[=====>] 608,357      595K/s   in 1.0s
2013-06-17 14:42:08 (595 KB/s) - `libnfc-1.7.0-rc7.tar.gz' saved [608357/608357]
```

Step Two: Setup libnfc for the Pi

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.

libnfc 1.7.0 and later use a new config file, which needs to be placed at a specific location. Thankfully, libnfc 1.7.0 includes a config file for the Raspberry Pi, which you can copy to the right destination with the following commands:

```
$ cd libnfc-1.7.0
$ sudo mkdir /etc/nfc
$ sudo mkdir /etc/nfc/devices.d
```

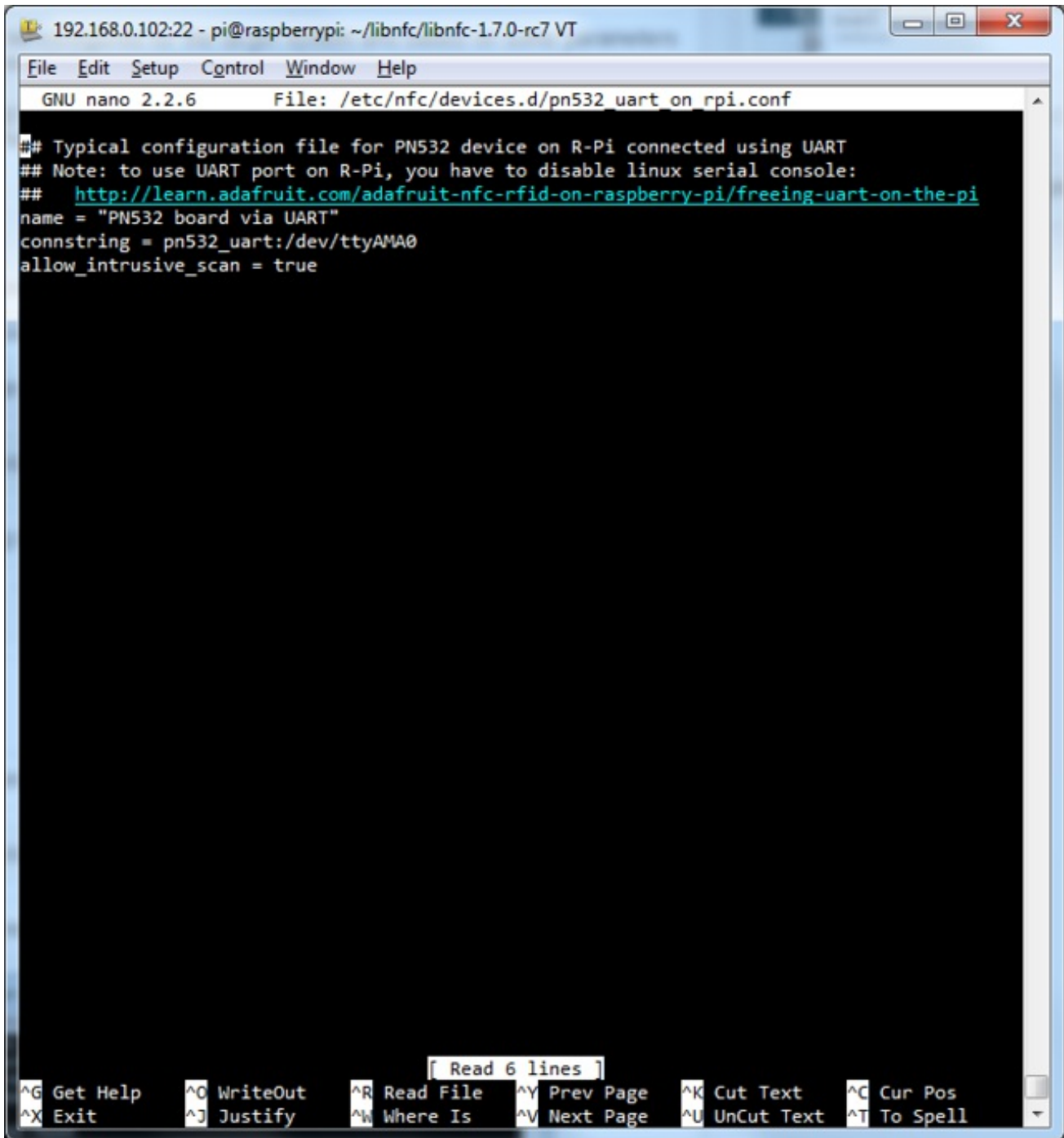
```
$ sudo cp contrib/libnfc/pn532_uart_on_rpi.conf.sample /etc/nfc/devices.d/pn532_uart_on_rpi.conf
```

Next, we need to make a small change to this file, which we can do by entering:

```
sudo nano /etc/nfc/devices.d/pn532_uart_on_rpi.conf
```

Update the file to include the following line at the bottom:

```
allow_intrusive_scan = true
```



The screenshot shows a terminal window with the title bar "192.168.0.102:22 - pi@raspberrypi: ~/libnfc/libnfc-1.7.0-rc7 VT". The terminal displays the GNU nano 2.2.6 text editor editing the file "/etc/nfc/devices.d/pn532_uart_on_rpi.conf". The file content is as follows:

```
# Typical configuration file for PN532 device on R-Pi connected using UART
## Note: to use UART port on R-Pi, you have to disable linux serial console:
## http://learn.adafruit.com/adafruit-nfc-rfid-on-raspberry-pi/freeing-uart-on-the-pi
name = "PN532 board via UART"
connstring = pn532_uart:/dev/ttyAMA0
allow_intrusive_scan = true
```

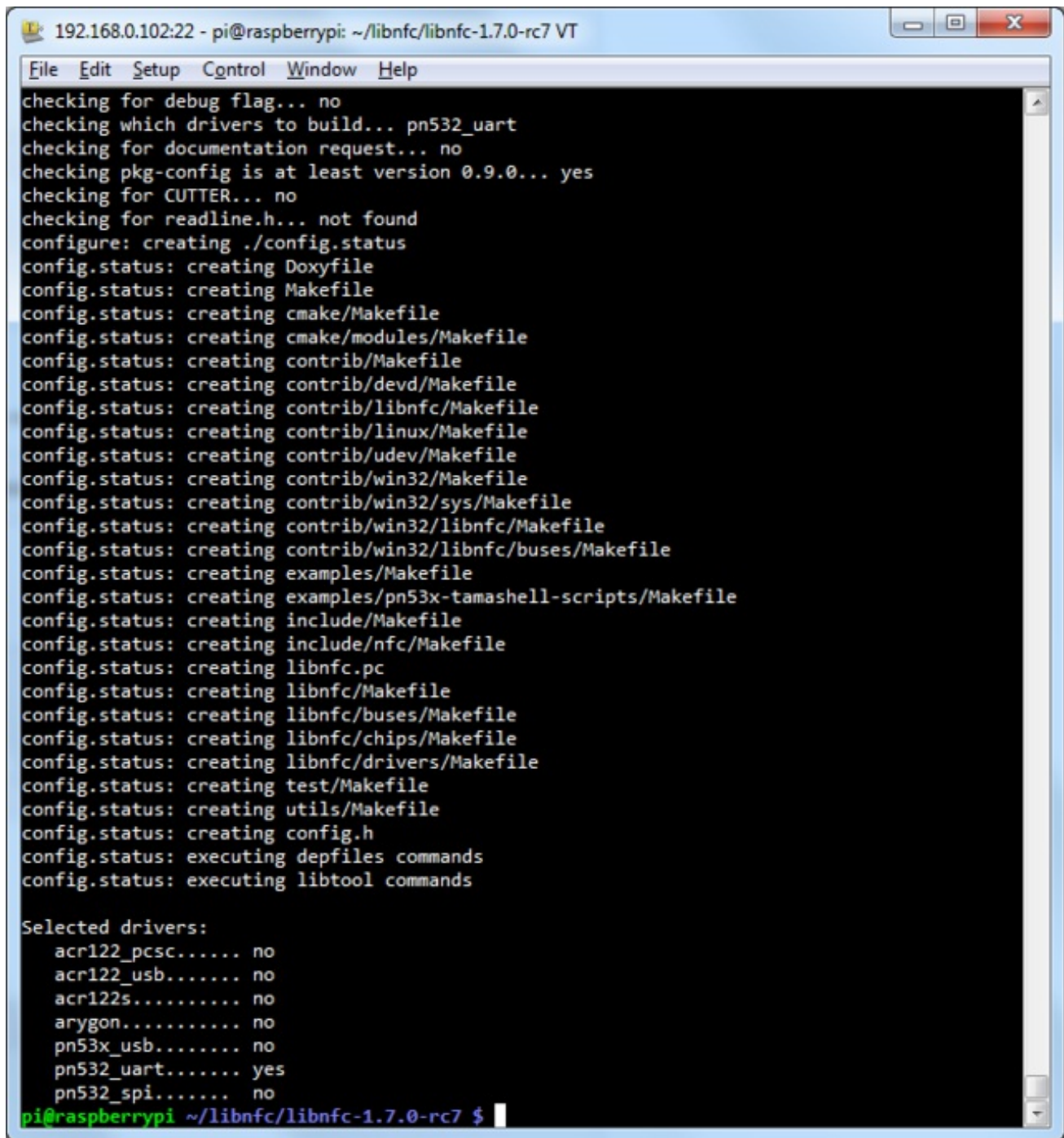
The bottom of the terminal shows the nano editor's help menu with various keyboard shortcuts for navigation and editing.

Step Three: Run Config

The next step is to configure the project itself using the 'configure' tool, as follows:

```
$ ./configure --with-drivers=pn532_uart --sysconfdir=/etc --prefix=/usr
```

This should give you the following screen once the configuration process is complete (it takes a few minute though):



```
192.168.0.102:22 - pi@raspberrypi: ~/libnfc/libnfc-1.7.0-rc7 VT
File Edit Setup Control Window Help
checking for debug flag... no
checking which drivers to build... pn532_uart
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/libnfc/Makefile
config.status: creating contrib/linux/Makefile
config.status: creating contrib/udev/Makefile
config.status: creating contrib/win32/Makefile
config.status: creating contrib/win32/sys/Makefile
config.status: creating contrib/win32/libnfc/Makefile
config.status: creating contrib/win32/libnfc/buses/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

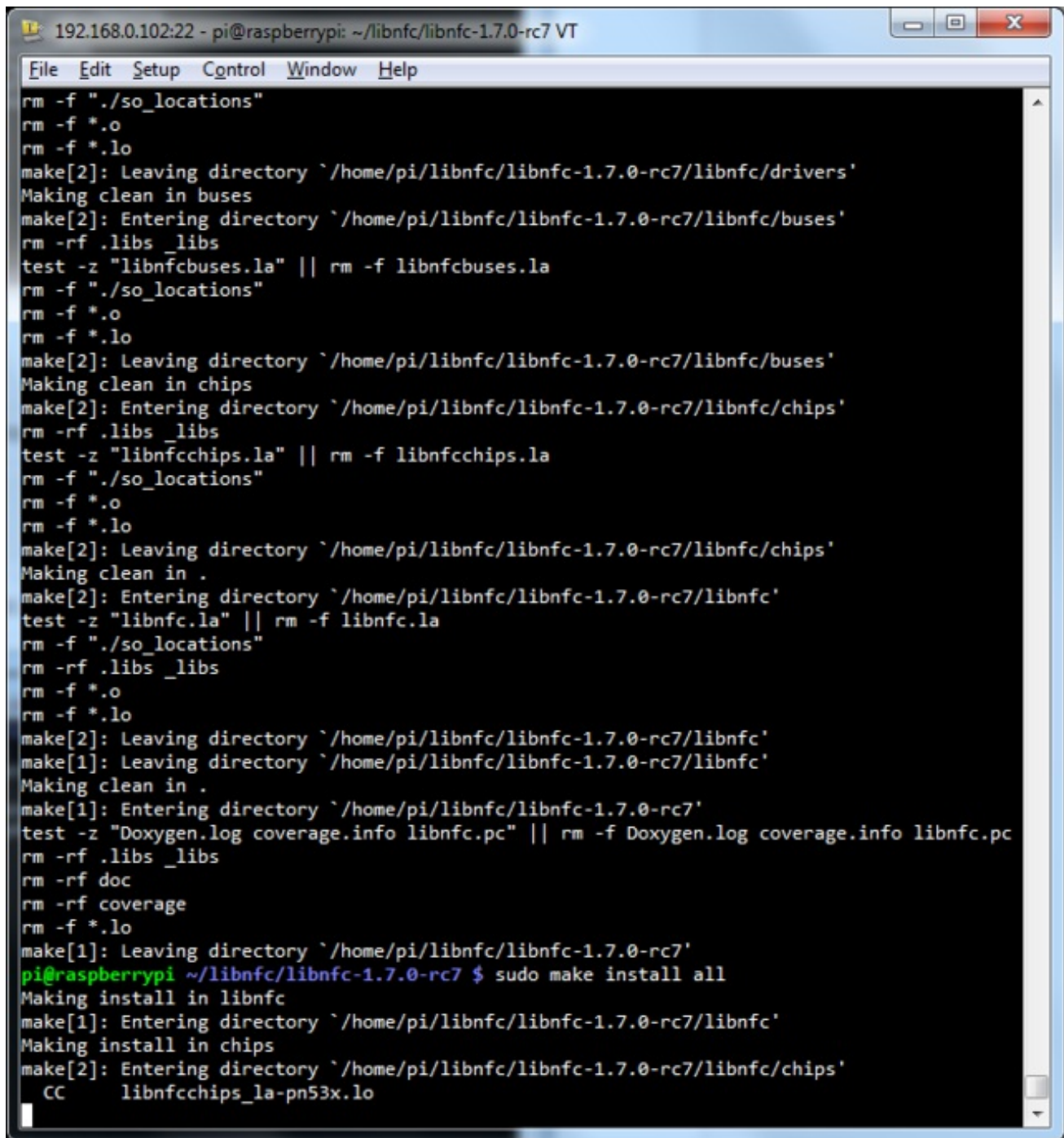
Selected drivers:
  acr122_pcsc..... no
  acr122_usb..... no
  acr122s..... no
  arygon..... no
  pn53x_usb..... no
  pn532_uart..... yes
  pn532_spi..... no
pi@raspberrypi ~/libnfc/libnfc-1.7.0-rc7 $
```

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:



```
192.168.0.102:22 - pi@raspberrypi: ~/libnfc/libnfc-1.7.0-rc7 VT
File Edit Setup Control Window Help
rm -f "./so_locations"
rm -f *.o
rm -f *.lo
make[2]: Leaving directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc/drivers'
Making clean in buses
make[2]: Entering directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc/buses'
rm -rf .libs _libs
test -z "libnfcbuses.la" || rm -f libnfcbuses.la
rm -f "./so_locations"
rm -f *.o
rm -f *.lo
make[2]: Leaving directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc/buses'
Making clean in chips
make[2]: Entering directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc/chips'
rm -rf .libs _libs
test -z "libnfcchips.la" || rm -f libnfcchips.la
rm -f "./so_locations"
rm -f *.o
rm -f *.lo
make[2]: Leaving directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc/chips'
Making clean in .
make[2]: Entering directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc'
test -z "libnfc.la" || rm -f libnfc.la
rm -f "./so_locations"
rm -rf .libs _libs
rm -f *.o
rm -f *.lo
make[2]: Leaving directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc'
make[1]: Leaving directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc'
Making clean in .
make[1]: Entering directory `/home/pi/libnfc/libnfc-1.7.0-rc7'
test -z "Doxygen.log coverage.info libnfc.pc" || rm -f Doxygen.log coverage.info libnfc.pc
rm -rf .libs _libs
rm -rf doc
rm -rf coverage
rm -f *.lo
make[1]: Leaving directory `/home/pi/libnfc/libnfc-1.7.0-rc7'
pi@raspberrypi ~/libnfc/libnfc-1.7.0-rc7 $ sudo make install all
Making install in libnfc
make[1]: Entering directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc'
Making install in chips
make[2]: Entering directory `/home/pi/libnfc/libnfc-1.7.0-rc7/libnfc/chips'
CC      libnfcchips_la-pn53x.lo
```

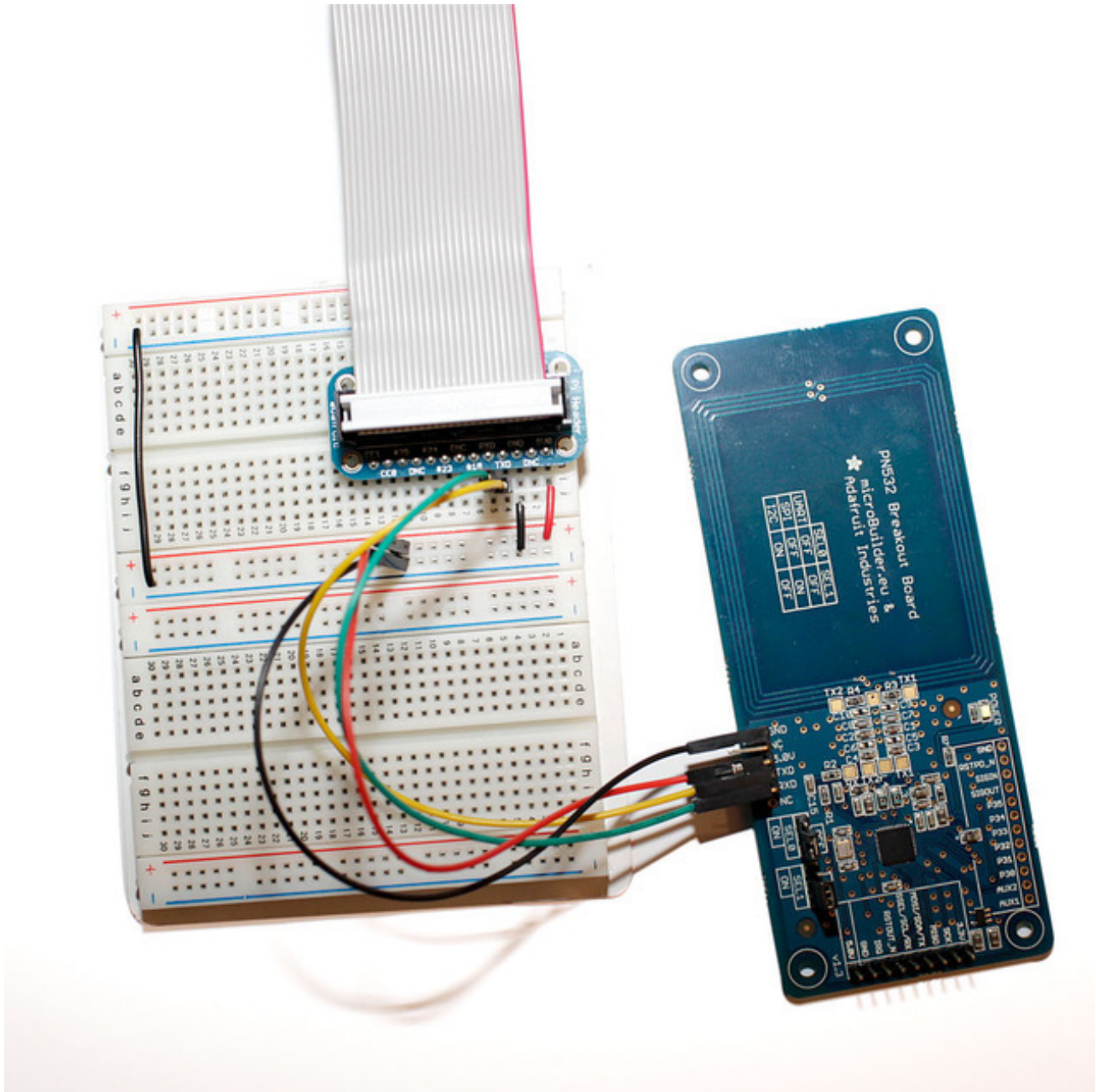
Once the build process is complete, you're ready to go on to testing the library on actual HW
...

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.

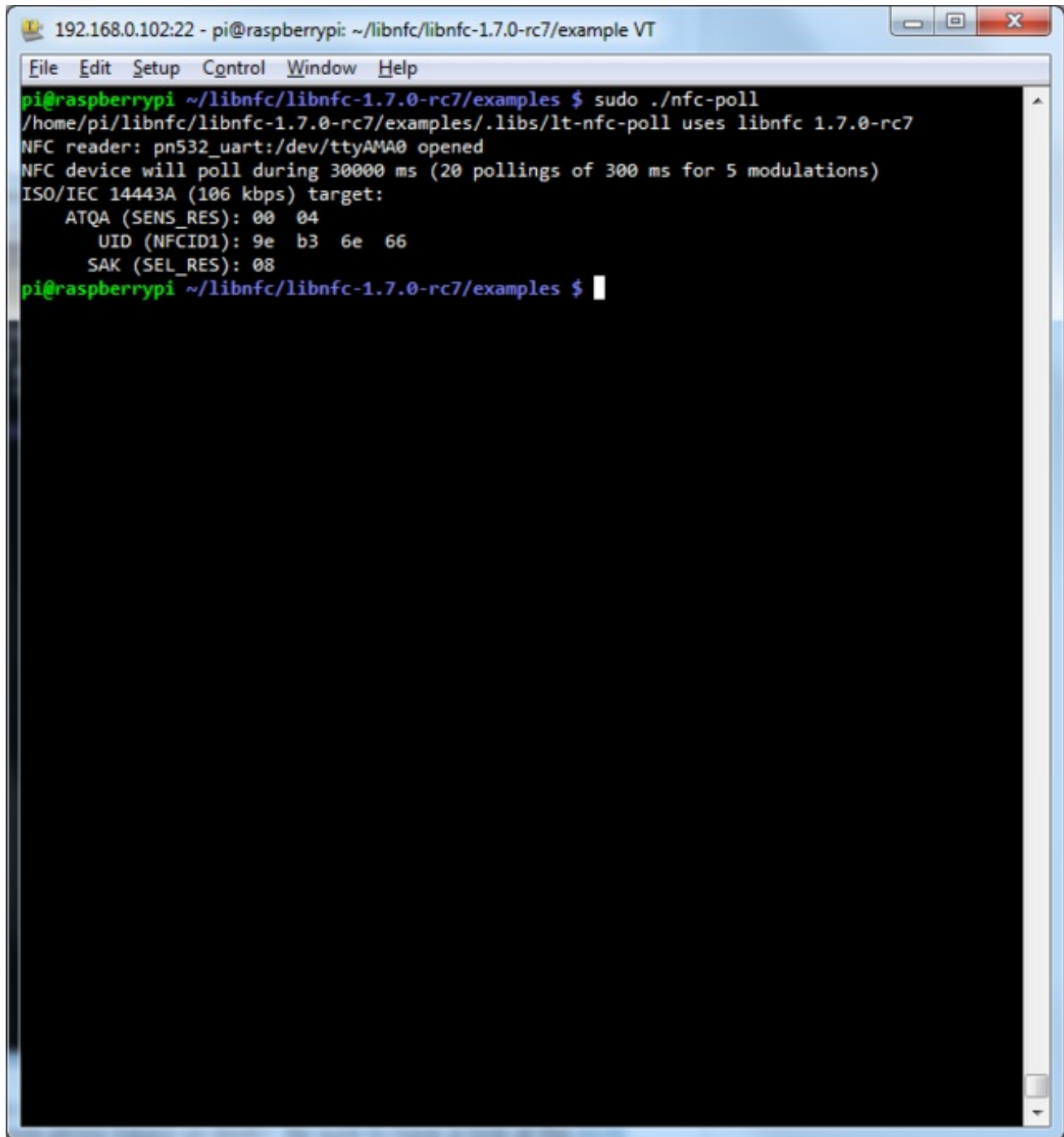
Use the 5V supply on the Pi Cobbler, and the 5V input on the FTDI header rather than the 3.3V supply, since the 3.3V supply is used by the core on the raspberry Pi and you don't want to pull sharp, heavy loads from it, like when you first enable and charge the near field.

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

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

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

Which should result in the following:



```
192.168.0.102:22 - pi@raspberrypi: ~/libnfc/libnfc-1.7.0-rc7/example VT
File Edit Setup Control Window Help
pi@raspberrypi ~/libnfc/libnfc-1.7.0-rc7/examples $ sudo ./nfc-poll
/home/pi/libnfc/libnfc-1.7.0-rc7/examples/.libs/lt-nfc-poll uses libnfc 1.7.0-rc7
NFC reader: pn532_uart:/dev/ttyAMA0 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): 9e b3 6e 66
  SAK (SEL_RES): 08
pi@raspberrypi ~/libnfc/libnfc-1.7.0-rc7/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.