ENSC 351 Spot-A-Bone

NFC Reader How-To Guide [2022-12-06]

Members

Justin Mateo	Raymond Cao	Avash Singh Thapa	Sina Haghighi

Table of Contents

1. PN532 NFC/RFID Controller

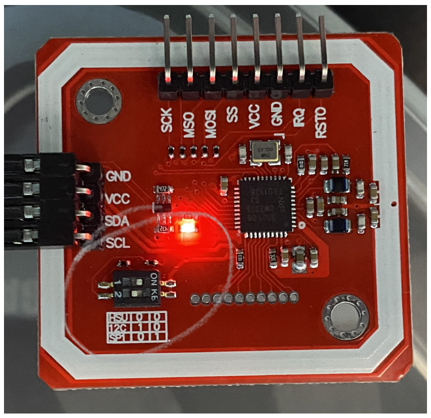
The PN532 is a transceiver module that uses Near Field Communication (NFC) or Radio-frequency Identification (RFID). There are six distinct operating modes that the PN532 module provides. The operating mode that is of interest is: *ISO/IEC 14443A/MIFARE Reader/Writer*. The NFC controller also supports SPI, I^2C and UART as host interfaces. This guide will go into how to set up the controller for reading (and some writing) using *I^2C* as its communication protocol between itself and the Beaglebone Green (BBG).

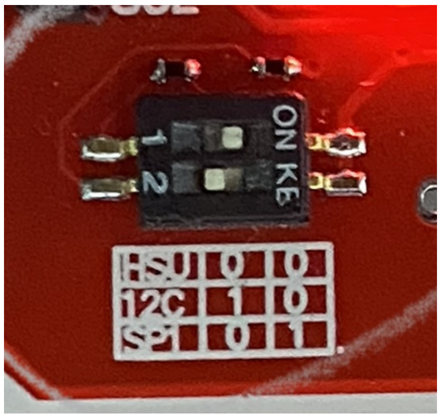
2. GPIO and Hardware Set Up

2.1 Set Communication Protocol as I^2C

The dip switches on the PN532 board give the user the opportunity to easily change the host interface between SPI, I^2C and UART.

1. Set the dip switches as in the following image to utilize I^2C:





2.2 Wiring

Connect the set of four header pins on the PN532 to the following GPIO pins on the BBG:

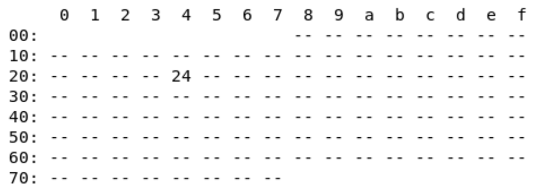
PN532	BeagleBone Green
GND	P9.01 (*Ground*)
VCC	P9.07 (*SYS 5V*)
SDA	P9.18 (*I2C1_SDA*)
SCL	P9.17 (*I2C1_SCL*)

3. I^2C Enabling and Testing

Note that this section is more deeply explained in Dr. Brian Fraser's *I2CGuide.pdf* which can be happily provided by Dr. Brian Fraser.

2.1 Enable the Bus

1. Install the I2C tools:
```
(bbg)$ sudo apt-get install i2c-tools
```
2. Configure the required pins for I2C:
```
(bbg)$ config-pin P9_17 i2c
(bbg)$ config-pin P9_18 i2c
```
3. Display that the PN532 module is on the **I2C bus at 0x24**:
```
(bbg)$ i2cdetect -y -r 1
```



```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- 24 -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

4. Remember that the steps above **must** be repeated every time the Beaglebone Green is rebooted.
5. Troubleshooting:
   If the above is not reflected, the most probable cause is that the pins were not configured.
   Check if the following returns `i2c`:
   ```
   (bbg)$ config-pin -q P9_17
   (bbg)$ config-pin -q P9_18
   ```

4. Library Setup of libnfc

3.1 Library Acquiry

1. Debian should provide the dpkg command. Ensure that the version is above 1.18 (amd64):
   ```
   (host)$ dpkg --version
   ```
2. Since the BBG's architecture is in ARM, it is crucial to have shared library files that can be translated into ARM machine code for the BBG with cross-compilation. The following ensures compatibility:
   ```
   (host)$ sudo dpkg --print-foreign-architectures
   armhf
   ```
   To declare that the desired library can be translated into arm hard float: concatenate "`:armhf`" to the end of the library name when installing using `apt install`. Examples are shown below.
3. Update all dependencies:
   ```
   (host)$ sudo apt update
   ```
4. Install all of the following libraries:
   ```
   (host)$ sudo apt install libnfc-bin:armhf
   (host)$ sudo apt install libnfc-dev:armhf
   (host)$ sudo apt install libnfc-examples:armhf
   (host)$ sudo apt install libnfc-libnfc-pn53x-examples:armhf
   (host)$ sudo apt install libnfc-libnfc6:armhf
   ```

3.2 Target Setup

Tools and git repositories will need to be installed, so Ethernet over USB must be set up and initiated. This is gone over in Dr. Brian Fraser's *Networking.pdf.*
The set up below takes inspiration from both Derek Molloy's "Exploring BeagleBone 2nd Ed (2019)" and the GitHub Repository for libnfc.

1. The following tools must be installed for target setup:
   ```
   (bbg)$ sudo apt-get install autoreconf
   (bbg)$ sudo apt-get install cmake
   (bbg)$ sudo apt-get install libusb-dev
   ```
2. Clone the libnfc git repository at root (or your desired directory), navigate to it and create a directory for *nfc* in the system configuration files directory:
   ```
   (bbg):~$ git clone https://github.com/nfc-tools/libnfc
   (bbg):~$ cd libnfc/
   (bbg):~/libnfc$ sudo mkdir /etc/nfc
   ```

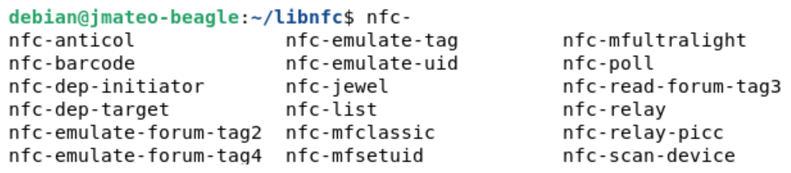
3. Copy and rename the libnfc configuration file sample into its system configuration file folder:
   ```
   (bbg):~/libnfc$ sudo cp libnfc.conf.sample /etc/nfc/libnfc.conf
   ```
4. Edit the configuration file such that it uses the I2C bus 1:
   ```
   (bbg):~/libnfc$ sudo nano /etc/nfc/libnfc.conf
   ```
   Paste the following at the bottom of the file:
   ```
   device.name = "PN532 over I2C"
   device.connstring = "pn532_i2c:/dev/i2c-1"
   ```
5. Do the following to further configure the cloned git repo folder and system configuration files:
   ```
   (bbg):~/libnfc$ autoreconf -vis
   (bbg):~/libnfc$ ./configure --prefix=/usr --sysconfdir=/etc
   (bbg):~/libnfc$ cmake .
   (bbg):~/libnfc$ make -f "Makefile.md"
   (bbg):~/libnfc$ sudo ldconfig -v
   (bbg):~/libnfc$ sudo cp contrib/udev/93-pn53x.rules /lib/udev/rules.d/
   ```

5. PN532 NFC/RFID Controller Troubleshooting

Even though all of the above has been executed, some problems may arise. Frequently ran into problems and their solution or procedure for diagnosing are discussed below.

5.1 Quick Hardware Check

It is safe to assume that a quick check is desired, such that running the whole program successfully is more predictable. Section 3.2 and its configurations gives the ability to use some nfc tools:



```
debian@jmateo-beagle:~/libnfc$ nfc-
nfc-anticol               nfc-emulate-tag        nfc-mfultralight
nfc-barcode               nfc-emulate-uid        nfc-poll
nfc-dep-initiator         nfc-jewel              nfc-read-forum-tag3
nfc-dep-target            nfc-list               nfc-relay
nfc-emulate-forum-tag2    nfc-mfclassic          nfc-relay-picc
nfc-emulate-forum-tag4    nfc-mfsetuid           nfc-scan-device
```

The more useful tools are `nfc-list` and `nfc-poll`. These will quickly determine whether there is a problem or not. A typical problem is shown in Section 5.2 below.

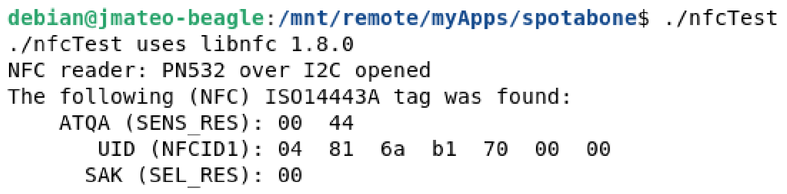
5.2 More In-depth Hardware Check

Attached support files are in the nfcTesting folder, bring them into VM:

nfc-utils.h, nfcTest.h, nfcTest.c, Makefile

1. Make, and run the NFC tester:
   ```
   (host):~/…/nfcTesting$ make
   ```
   Move to the BBG
   ```
   (bbg):~$ ./mount-nfs
   (bbg):~$ cd /mnt/remote/myApps/
   (bbg):~/mnt/remote/myApps$ ./nfcTest
   ```

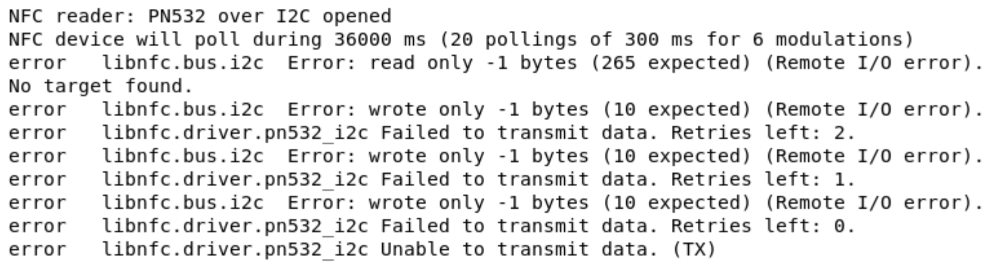
If working perfectly, the following will output:



```
debian@jmateo-beagle:/mnt/remote/myApps/spotabone$ ./nfcTest
./nfcTest uses libnfc 1.8.0
NFC reader: PN532 over I2C opened
The following (NFC) ISO14443A tag was found:
    ATQA (SENS_RES): 00  44
       UID (NFCID1): 04  81  6a  b1  70  00  00
      SAK (SEL_RES): 00
```

5.3 Failure to Transmit Data Failure

The following errors can show up from time to time. This is most likely due to a **loose connection**, **low power input**, or **unfortunate initial boot up of the PN532**.



```
NFC reader: PN532 over I2C opened
NFC device will poll during 36000 ms (20 pollings of 300 ms for 6 modulations)
error   libnfc.bus.i2c  Error: read only -1 bytes (265 expected) (Remote I/O error).
No target found.
error   libnfc.bus.i2c  Error: wrote only -1 bytes (10 expected) (Remote I/O error).
error   libnfc.driver.pn532_i2c Failed to transmit data. Retries left: 2.
error   libnfc.bus.i2c  Error: wrote only -1 bytes (10 expected) (Remote I/O error).
error   libnfc.driver.pn532_i2c Failed to transmit data. Retries left: 1.
error   libnfc.bus.i2c  Error: wrote only -1 bytes (10 expected) (Remote I/O error).
error   libnfc.driver.pn532_i2c Failed to transmit data. Retries left: 0.
error   libnfc.driver.pn532_i2c Unable to transmit data. (TX)
```

Take the following precautions to fix this:

1. Make sure all mechanical connections are well created.
2. The VDD input of the NFC module must be SYS 5V and not VDD_3_3. Though the module will still turn on, and function at times, there is not enough power to ensure a stable connection. Providing the SYS 5V is the most reliable, and provides the most stable connection.
3. The initial boot up of the PN532 is controlled by the module itself, and the libnfc library. With these constraints, it is very difficult to ensure a consistent successful boot up. The only solution found is to continue power cycling the module until desired outputs are returned.