nfc-tools Reference Manual

How to use nfc-tools

nfc-tools is a free open-sources Linux project which provides functions for using PN532. Herein we will install and use nfc-tools in Raspberry Pi OS.

Install libraries

Before we install nfc-tools, several libraries are required, otherwise, the nfc-tools cannot be compiled successfully.

```
sudo apt-get install libusb-dev libpcsclite-dev libtool
sudo apt-get install automake autoconf
```

Clone libnfc and mfoc

Download the source codes of libnfc and mfoc.

```
git clone https://github.com/nfc-tools/libnfc.git
git clone https://github.com/nfc-tools/mfoc.git
```

Compile libnfc

Compile libnfc by the following commands:

```
cd libnfc
autoreconf –vis
./configure --prefix=/usr --sysconfdir=/etc
make
sudo make install
```

Note：sudo should be added in front of the make install

Compile mfoc

Compile mfoc by the following commands:

```
cd mfoc
autoreconf -vis
./configure --prefix=/usr --sysconfdir=/etc
make
sudo make install
```

Note: sudo should be added in the front of make install

nfc-tools Configure serial Port

1. Configure the jumpers on board, you should Set I0 to L and I1 to L.
2. Configure the DIP switch into:

SCK	MISO	MOSI	NSS	SCL	SDA	RX	TX
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

That is ：

- NSS/SCL/RX → TX of Raspberry Pi
- MOSI/SDA/TX → RX of Raspberry Pi

3. Create `/etc/nfc/libnfc.conf`

   ```
   sudo mkdir /etc/nfc/
   sudo cp libnfc/libnfc.conf.sample /etc/nfc/libnfc.conf
   ```

4. Modify `/etc/nfc/libnfc.conf`

   ```
   sudo nano /etc/nfc/libnfc.conf
   ```

   Delete character "#" from the line `device.connstring = "pn532_uart:/dev/ttyUSB0"` ,and change it to `device.connstring = "pn532_uart:/dev/ttyS0"`

   Note:

   a) You need to enable the serial port of Raspberry Pi by command `sudo raspi-config` , then disable login shell function and enable the hardware serial by the following steps:

   - Interfacing Options → Serial
   - Would you like a login shell to be accessible over serial? → No
   - Would you like the serial port hardware to be enabled? → Yes

   b) If you want to use the module with other Linux PCs, you may requires a TTL to USB Converter and connect the PN532 NFC HAT to Linux PC by the TTL to USB Converter.

5. If you can get the response `NFC device: opened` after running the command: `nfc-list`, and it means the PN532 NFC HAT is connected successfully.

nfc-tools Configure I2C

1. You need to configure the jumpers on board, set I0 to H and I1 to L.通
2. Configure the DIP switch into:

SCK	MISO	MOSI	NSS	SCL	SDA	RX	TX
OFF	OFF	OFF	OFF	ON	ON	OFF	OFF

That is:

- NSS/SCL/RX → SCL of Raspberry Pi
- MOSI/SDA/TX → SDA of Raspberry Pi

3. Run command `sudo raspi-config` to enable I2C interface
   - Interfacing Options → I2C
   - Would you like the ARM I2C interface to be enabled? → Yes
4. Create `/etc/nfc/libnfc.conf`

```
sudo mkdir /etc/nfc/
sudo cp libnfc/libnfc.conf.sample /etc/nfc/libnfc.conf
```

5. Modify `/etc/nfc/libnfc.conf`

   ```
   sudo nano /etc/nfc/libnfc.conf
   ```

   Delete the character "#" from the line: `device.connstring = "pn532_uart:/dev/ttyUSB0"` , and change it to `device.connstring = "pn532_i2c:/dev/i2c-1"`

6. If you can get the response `NFC device:  opened` after running the command: `nfc-list`, and it means the PN532 NFC HAT is connected successfully.

nfc-tools Configure SPI

To leave pins for other SPI devices, PN532 NFC HATG use D4 (BCM) pins instead of the CE0 as chip selection pin. However, nfc-tools use CE0 pins as chip selection of SPI interface, in this case, you need to connect D4 and CE0 when you use SPI interface. Before you connect the pins, please follow the steps below to configure the pins for protect pins from damage.

1. Check status of D4 pin by command: `gpio readall`

```
+-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
|     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
|   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5v      |     |     |
|   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
|   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT5 | TxD     | 15  | 14  |
|     |     |      0v |      |   |  9 || 10 | 1 | ALT5 | RxD     | 16  | 15  |
|  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
|  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
|  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
|     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
|  10 |  12 |    MOSI | ALT0 | 1 | 19 || 20 |   |      | 0v      |     |     |
|   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
|  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
|     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
|   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
|   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
|   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
|  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
|  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 1 | IN   | GPIO.27 | 27  | 16  |
|  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 1 | IN   | GPIO.28 | 28  | 20  |
|     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

2. **Mode of D4 should be IN instead of OUT!必须为 IN，please take care of it. Only the board can work normally when the Mode of D4 is IN, otherwise the Raspberry Pi may be damaged. If the Mode of D4 is OUT, please configure it by the following command:**

   ```
   python3
   import RPi.GPIO as GPIO
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(4, GPIO.IN)
   exit()
   ```

3. Connect pin D4 and CE0.
4. Configure jumpers on board, set I0 to L and I1 to H.

5. Connect RSTPDN → D20 by jumper
6. Set DIP switch into:

SCK	MISO	MOSI	NSS	SCL	SDA	RX	TX
ON	ON	ON	ON	OFF	OFF	OFF	OFF

That is：

- SCK → SCK of Raspberry Pi
- MISO → MISO of Raspberry Pi
- MOSI/SDA/TX → MOSI of Raspberry Pi
- NSS/SCL/RX → D4 of Raspberry Pi

7. Use `sudo raspi-config` command to enable SPI interface:
   - Interfacing Options → SPI
   - Would you like the SPI interface to be enabled? → Yes
8. Create `/etc/nfc/libnfc.conf`

   ```
   sudo mkdir /etc/nfc/
   sudo cp libnfc/libnfc.conf.sample /etc/nfc/libnfc.conf
   ```
9. Modify `/etc/nfc/libnfc.conf`

   ```
   sudo nano /etc/nfc/libnfc.conf
   ```

   Delete the character "#" from the line: `device.connstring = "pn532_uart:/dev/ttyUSB0"` and change it to `device.connstring = "pn532_spi:/dev/spidev0.0"`
10. If you can get the response `NFC device: opened` after running the command: `nfc-list`, and it means the PN532 NFC HAT is connected successfully.

nfc-tools Examples

1. List the NFC card:

   ```
   nfc-list
   ```

   Near the NFC card to PN532 NFC HAT and run the command, it will read the ID and type of the NFC card.
2. Export NFC card:

   ```
   mfoc -O output.mfd
   ```

   The command is used to read the data of NFC card and save in output.mfd file. You can also change the file name as you like. Even the command will exhaustively try all common password if the NFC card is encrypted, it may failed in this case.
3. Write the whole card

   ```
   nfc-mfclassic w a f input.mfd
   ```

   This command is used to write all the data to a card, which are dump from other card. input.mfd are the data input, you can modify it according the actual file name.
4. Configure UID。

   ```
   nfc-mfsetuid 01234567
   ```

   This command is used to modify the UID (Block 0) of MIFARE card. 01234567 is the four bytes UID and it is configurable.

   Note：

   a) This card can only be used for MIFARE card and the card should support backdoor command: HALT, 0x40, 0x43。

   修改成功的话，终端提示：

```
Sent bits:     50  00  57  cd
Sent bits:     40 (7 bits)
Received bits: a (4 bits)
Sent bits:     43
Received bits: 0a
Card unlocked
Sent bits:     a0  00  5f  b1
Received bits: 0a
Sent bits:     01  23  45  67  00  08  04  00  46  59  25  58  49  10  23  02  23  eb
Received bits: 0a
```

否则终端提示

```
Sent bits:     50  00  57  cd
Sent bits:     40 (7 bits)
Warning: Unlock command [1/2]: failed / not acknowledged.
Sent bits:     a0  00  5f  b1
Sent bits:     01  23  45  67  00  08  04  00  46  59  25  58  49  10  23  02  23  eb
```

b) MIFARE card may be recognized by some NFC reader without any response, it does not mean that the MIFARE card is broken.

5. Format card：

   `nfc-mfclassic f A u dummy.mfd dummy.mfd f`

   dummy.mfd is the data file of empty card.

6. Unlock. Sometime, users may lock their card by mistake (for example, write wrong bits), in this case, users can try this command to force format the card:

   `nfc-mfclassic W A u dummy.mfd dummy.mfd f`

   This command can only work for MIFARE card.

For more information about the tools, you can refer to this website: http://nfc-tools.org/index.php?title=Libnfc

The name of tools can be found under libnfc/utils directory, for example: nfc-barcode、nfc-emulate-forum-tag4、nfc-list、nfc-mfclassic　and so on.

BTW, there are also examples of nfc-tools, which can be found in libnfc/examples directory. Please refer to the website for more information:：http://nfc-tools.org/index.php/Category:Examples