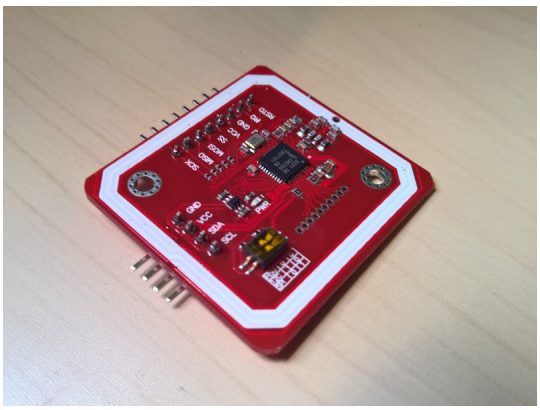
NFC-Reader am Raspberry Pi

Hier sind wir in der Welt der berührungslos funktionierenden Chipkarten, Tags oder Labels angekommen. Solcherlei RFID-Technik gibt es bei Zutrittssystemen oder Studierendenausweisen schon länger, hält aber erst in letzter Zeit richtig Einzug. Zum Beispiel im eTicketing oder beim Bezahlen mittels kontaktloser Kreditkartenfunktion im Markt oder Tankstelle. Auch viele Smartphones können mittlerweile „NFC“, dort eröffnen sich ganz neue Möglichkeiten.

NFC-Reader im Eigenbau [Bild nfc05.jpg]

Es gibt einige Unterschiede in der Ausprägung der Technik, der klassische Bereich der Chipkarten ist in Normen wie der ISO14443 spezifiziert. Bei den mobilen Geräten spielen nun auch Standards aus den NFC-Books eine Rolle, wo man heutzutage Bookmarks oder Visitenkarten untereinander austauschen kann. Auch andere Hardware nutzt mittlerweile diese Technik, wo sich zum Beispiel eine Digitalkamera gegenüber ihrer Dockingstation identifizieren kann.

Allen Anwendungen mit "Near-Field-Communication" ist aber gemein, dass über eine Kurzstrecke eine Funkverbindung zur Datenübertragung aufgebaut wird. Es hat sich das 13.56Mhz-Band im Proximity/Nahfeld-Bereich durchgesetzt, man kommuniziert über bis 10cm Abstand. Bei der notwendigen Technik in Karten, Schlüsselanhänger oder Labels wird gerne über "Transponder" gesprochen, welche dort verbaut sind. Damit ist die Grundeinheit von Chip und Antenne gemeint. Passive Transponder sind kompakt und brauchen keine weiteren Komponenten, denn sie beziehen ihre Stromversorgung aus dem anliegenden elektrischen Feld über ihre Antenne.



PN532 V3-Readermodul [Bild nfc01.jpg]

In unserem Projekt geht es nun also um einen Leser, um diverse Transponder auswerten und kodieren zu können. Da gibt es natürlich auch käufliche Reader zu erwerben. Solche Chipkartenleser kennt man im Allgemeinen als Geräte mit USB-Anschluss. Im Handel findet man sie selten, wenn dann oft auch als Klasse3-Leser mit PIN-Pad und Display, für das Homebanking oder den neuen Personalausweis. In der Industrie sind sie häufiger anzutreffen und in vielen Terminals im öffentlichen Nahverkehr verbaut.

Aber einen solchen RFID-Reader kann man sich auch selbst bauen und die Software auf die ganz eigenen Bedürfnisse abstimmen. Dazu braucht es einmal die Leseeinheit mit Antenne, bei uns ein Breakout-Board

mit dem PN532-Chipsatz. Der Steuerrechner soll ein Raspberry Pi sein. Das PN532-Board in der 3er-Version bietet für die Anbindung gleich drei Möglichkeiten, per I2C, SPI oder seriell.
Ist man sich über das zu verwendende Interface einig, müssen eventuell die Microschalter auf der Platine umgestellt werden, um wie in unserem Aufbau die I2C-Schnittstelle zu aktivieren. Die Verkabelung zum Pi braucht dann nur GND, VCC (5V), SDA und SCL, und anschließend kann man eine Probe mit

```
$ sudo i2cdetect -y 1
```

machen. Ergibt sich dieses Bild

```
     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: -- -- -- -- -- -- -- --
```

wurde das Lesermodul also gefunden. Manchmal bekommt man auch die Adresse 0x25 angezeigt.

Nun zuerst *libusb-dev* und *dh-autoreconf* und die PC/SC-Bibliotheken installieren:

```
$ sudo apt-get install i2c-tools libusb-dev dh-autoreconf
$ sudo apt-get install pcscd libpcsclite1 libpcsclite-dev
```

Danach die libnfc-Sourcen holen und das Paket vorbereiten:

```
$ cd ~
$ git clone https://github.com/nfc-tools/libnfc.git
$ cd libnfc
$ autoreconf -vis
$ ./configure --with-drivers=all --sysconfdir=/etc --prefix=/usr
$ sudo make clean
$ sudo make install all
```

Damit "nfc-list" funktioniert ("error while loading shared libraries: libnfc.so.5: cannot open shared object file: No such file or directory") muss mitunter noch Folgendes gefixt werden:

```
$ echo '/usr/local/lib' | sudo tee -a /etc/ld.so.conf.d/usr-local-lib.conf && sudo
ldconfig
```

Die Konfiguration von *libnfc* weiß auch noch nicht, welches Device angesprochen werden soll. Für unser PN532-Modul am I2C-Bus muss dazu im Config-File unter /etc/nfc/ noch der entsprechende *connstring* eingetragen werden:

```
connstring = pn532_i2c:/dev/i2c-1
```

Daneben können in der *libnfc.conf* noch Einstellungen zu Environment-Variablen, Device-Name, Scan und Loglevel vorgenommen werden. Oder man arbeitet mit den Defaults, welche für uns funktionieren. Ist die Konfigurationsdatei noch nicht vorhanden, kriegt man ein Beispiel aus dem „contrib“-Verzeichnis aus dem „libnfc“-Verzeichnisbaum oder man legt sie neu an.

Anschließend sollt man mittels

```
$ nfc-list
```

den NFC-Reader angezeigt bekommen:

```
nfc-list uses libnfc 1.7.1
NFC device: pn532_i2c:/dev/i2c-1 opened
```

Tag auslesen [Bild nfc02.jpg]

Jetzt wird es beim ersten Test spannend. Dazu einen Transponder auf das Readermodul auflegen und wieder

```
$ nfc-list
```

im Terminal eingeben. Ein NTAG213-Label mit 7 Byte UID ergibt dann eine solche Ausgabe:

```
nfc-list uses libnfc libnfc-1.7.1
NFC device: pn532_i2c:/dev/i2c-1 opened
1 ISO14443A passive target(s) found:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  44
       UID (NFCID1): 04  ed  0f  02  e5  3f  84
      SAK (SEL_RES): 00
```

Geschafft!

Recht hilfreich, dass man nun auch alle diverse Kommandozeilen-Tools von *libnfc* zur Verfügung hat, wie:

```
nfc-anticol                nfc-emulate-uid            nfc-read-forum-tag3
nfc-dep-initiator          nfc-list                   nfc-relay
nfc-dep-target             nfc-mfclassic              nfc-relay-picc
nfc-emulate-forum-tag2     nfc-mfsetuid               nfc-scan-device
nfc-emulate-forum-tag4     nfc-multralight
```

Hat man *Libfreefare* installiert, erweitert sich der Funktionsumfang noch. Der detaillierte Umgang mit kontaktlosen Chipkarten wie *Mifare Classic, Desfire, Ultralight* oder *NTAG* soll dann auch nicht Bestandteil hier sein, das ist ein weites Feld und soll euren eignen, speziellen Projekten überlassen bleiben.

Pi und Readermodul verbinden [Bild nfc03.jpg]

Nachdem nun alles funktioniert, kann unser Gehäuse nun auch geschlossen werden. Für unser Projekt haben wir ein fertiges Rundgehäuse für den Pi verwandt, für die Aufnahme eines „großen“ Pi vorbereitet und mit einer ebenen Fläche gut für das Auflegen der Transponder. Letztlich reicht auch ein *Pi Zero W*, die Rechenaufgaben sind nicht besonders anspruchsvoll.

Nur eine WiFi-Verbindung sollte es haben, damit man in einem Terminal an einem Rechner im heimischen Netzwerk arbeiten kann. Dann braucht unser Lesegerät auch nur ein einziges Verbindungskabel zum Steckernetzteil, was aus der seitlichen Aussparung herausgeführt ist.

Fertiges Lesegerät im Rundgehäuse [Bild nfc04.jpg]

PN532-Boards bekommt man im diversen Versand. Und es gibt verschiedene Ausführungen dieser Platinen. Die Version 3 hat mir gefallen, da sie kompakt ist und ein Antennendesign hat, was im Durchmesser einen guten Kompromiss zwischen größeren Transpondern, wie in Chipkarten verbaut, und kleineren Objekten macht. Damit ist auch eine gute Lesbarkeit von Schlüsselanhängern oder kleinen Labeln gegeben.

Auch die anderen Anbindungsmöglichkeiten an den Raspi wurden getestet. So bietet *SPI* die beste Übertragungsgeschwindigkeit. Jeder mag da selber auswählen. Auch die *Libnfc* ist sicher nicht die einzige Variante für die Programmierung, wer in Python firm ist, kann mittels *nfcpy* loslegen.

Nicht uninteressant auch, dass der PN532-Chipsatz die Möglichkeit bietet, NFC-Tags zu emulieren, das können derzeit die wenigsten USB-Geräte. Interessant ist diese Technik allemal, und wohl ein Baustein im „Internet der Dinge“, wo alles und überall miteinander kommunizieren soll...