Raspberry Pi RFID RC522 Tags auslesen (NFC)

Felix Mai 06, 2016

RFID ist eine Technologie, wodurch Daten ohne Berührung übertragen werden, was in Chipkarten Anwendung findet. Mit einem Raspberry Pi RFID Modul (RC522) können Zugangskarten ausgelesen werden und somit z.B. Zugriff zu Türen oder Schlössern gegeben werden. Aktuelle Smartphones besitzen ähnliche

Wie man mit dem RC522 und dem Raspberry Pi RFID Tags ausließt und außerdem Chipkarten beschreibt, zeige ich in diesem Tutorial. Außerdem kann der vorgestellte Code für andere Projekte (Türöffner, Zugangskontrolle) verwendet werden.

NFC ist eine zugehörige Technologie, dessen Unterschiede hier nachgelesen werden können (Englisch). Sowohl RFID als auch NFC senden auf einer Frequenz von 13.56 MHz, weshalb die Module miteinander kompatibel sind.

Verwendetes Zubehör

Die folgenden (Bau-)Teile habe ich für dieses Tutorial verwendet:

- Raspberry Pi 3 (funktioniert auch mit allen Vorgängern)
- Mifare RC522 RFID Modul (inkl. KeyCard)
- Female – Female Jumper Kabel
- Lötutensilien

Es gibt auch RFID USB Lesegeräte, allerdings habe ich diese nicht ausprobiert.

Falls man das Kartenlesegerät als Eingangskontrolle, etc. verwenden will, macht es Sinn jedem Nutzer eine Karte zu geben. Man kann diese Chipkarten in kleineren und größeren Mengen auch für kleines Geld

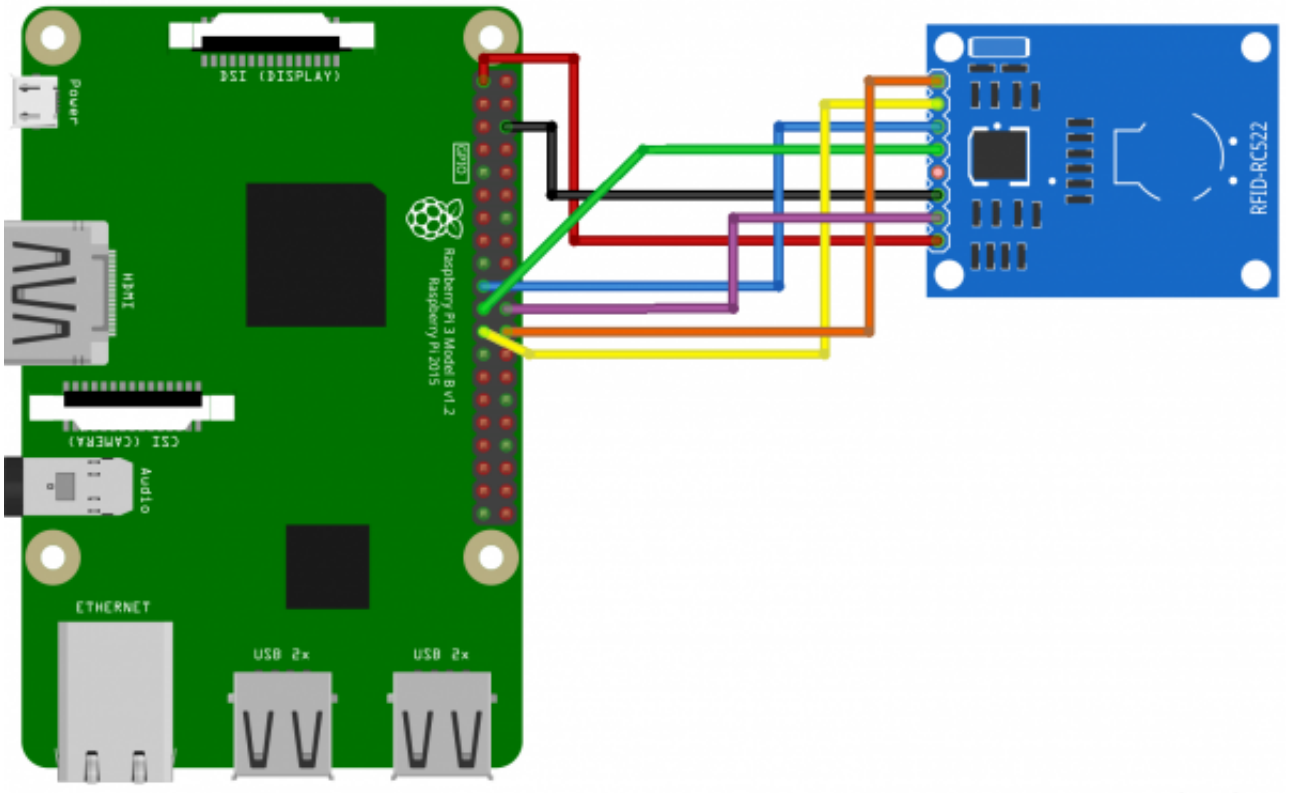
zusätzlich erwerben und dann mit dem RC522 jede Karte individuell beschreiben (Anleitung dazu weiter unten).

Aufbau

Bei meinem Modul war die Pin Leiste noch nicht angelötet, daher musste ich sie erst löten, bevor ich die Pins verbinden konnte.
Die Verkabelung ist dabei wie folgt:

RF522 Modul	Raspberry Pi
SDA	Pin 24 / GPIO8 (CE0)
SCK	Pin 23 / GPIO11 (SCKL)
MOSI	Pin 19 / GPIO10 (MOSI)
MISO	Pin 21 / GPIO9 (MISO)
IRQ	—
GND	Pin6 (GND)
RST	Pin22 / GPIO25
3.3V	Pin 1 (3V3)

Schematisch sieht das ganze dann so aus:



Raspberry Pi RC522 Verbindung

SPI aktivieren und Software installieren

Um das RFID RC522 Shield verwenden zu können brauchen wir den SPI Bus. Damit der Kernel beim Starten geladen wird, bearbeiten wir die config Datei:

```
sudo nano /boot/config.txt
```

Folgender Inhalt wird an das Ende der Datei hinzugefügt:

```
device_tree_param=spi=on
dtoverlay=spi-bcm2708
```

Gespeichert und beendet wird mit STRG+O, STRG+X. Danach aktivieren wir SPI noch:

```
sudo raspi-config
```

Unter „Advanced Options“ > „SPI“ aktivieren und den Neustart bestätigen (alternativ `sudo reboot now`).

Danach kann mittels `dmesg | grep spi` überprüft werden, ob das Modul geladen wurde. Es sollte dann so einen Output ergeben:

```
pi@raspberrypi:~ $ dmesg | grep spi
[    10.784368] bcm2708_spi 20204000.spi: master is unqueued, this is deprec
[    10.813403] bcm2708_spi 20204000.spi: SPI Controller at 0x20204000 (irq
```

Nun müssen noch die Pakete installiert werden, sodass wir auf den SPI Bus zugreifen können und eine entsprechende Bibliothek aus GitHub laden können.

```
sudo apt-get install git python-dev --yes
```

Zu erst installieren wir also das Python SPI Modul

```
git clone https://github.com/lthiery/SPI-Py.git
cd SPI-Py
sudo python setup.py install
cd ..
```

und danach die Raspberry Pi RFID RC522 Bibliothek:

```
git clone https://github.com/mxgxw/MFRC522-python.git && cd MFRC522-python
```

Raspberry Pi RFID Reader/Writer testen

Zusätzlich zum RC522 Modul werden meist eine weiße Karte sowie ein NFC fähiger Schlüsselanhänger geliefert. Diese Teile können als Authentifizierung benutzt werden, da sie beschreibbar und lesbar sind. Ebenfalls könnte ein NFC fähiges (Android/iOS) Smartphone verwendet werden (was die meisten neueren Handys ja sind).

Um einen ersten Test der Karte/Schlüsselanhängers durchzuführen, lassen wir das Skript laufen:

```
sudo python Read.py
```

Sobald die Chipkarte daran gehalten wird und erkannt wird, sieht man eine Ausgabe wie diese:

```
pi@raspberrypi:~/MFRC522-python $ sudo python Read.py
Welcome to the MFRC522 data read example
Press Ctrl-C to stop.
Card detected
Card read UID: 69,245,238,117
Size: 8
Sector 8 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Um nun die gespeicherten Daten (Zahlen) auf dem Chip zu ändern, bearbeiten wir die „Write.py“ Datei (`sudo nano Write.py`). Dazu

bearbeitest du den Code ab Zeile 55 folgendermaßen (die 16 Zahlen von `data` kannst du dabei frei zwischen 0 und 255 wählen. Ich habe dabei ein Wort mit ASCII Zeichen dargestellt):

```
55
56
57
58          # Variable for the data to write
59          data = [114, 97, 115, 112, 98, 101, 114, 114, 121, 45, 116, 117, 116, 111,
60  114, 0]
61          # Fill the data with 0xFF
62          #for x in range(0,16):
63          #    data.append(0xFF)
64          print "Sector 8 looked like this:"
65          # Read block 8
66          MIFAREReader.MFRC522_Read(8)
67          print "\n"
68          #print "Sector 8 will now be filled with 0xFF:"
69          # Write the data
70          MIFAREReader.MFRC522_Write(8, data)
71          #print "\n"
72          print "It now looks like this:"
73          # Check to see if it was written
74          MIFAREReader.MFRC522_Read(8)
75          print "\n"
76          """
77          data = []
```

```
78          # Fill the data with 0x00
79          for x in range(0,16):
80              data.append(0x00)
81          print "Now we fill it with 0x00:"
82          MIFAREReader.MFRC522_Write(8, data)
83          print "\n"
84          print "It is now empty:"
85          # Check to see if it was written
86          MIFAREReader.MFRC522_Read(8)
87          print "\n"
88          """
89
90
91
```

NFC / RFID Reader in Raspberry Pi Projekten verwenden (Türschloss, etc.)

Die beiden Python Dateien „Read.py“ und „Write.py“ enthalten einigen Beispielcode zum Lesen und beschreiben eines Chips, der in anderen Projekten zum Einsatz kommen kann. Die wichtigste Datei ist dabei „MFRC522.py“, welche in ein anderes Projekt mit kopiert werden kann.

Der folgende Ausschnitt kann in anderen Projekten verwendet werden, wie z.B. als Überprüfung eines Codeschloss bzw. Türschloss. Ich verwende eine Authentifizierungsstufe (man könnte auch mehrere einstellen) mit einem bestimmten Anfangs-Code. Die letzten Ziffern geben Auskunft über den Inhaber der Karte (sofern irgendwo dazu Daten gespeichert sind). Man könnte den User auch nur anhand der UID identifizieren, allerdings gehe ich vom Fall aus, dass auch mehrere Karten einem Nutzer gehören können.

Wem diese Lösung nicht gefällt, kann es natürlich umändern.

Ich habe in der Datei „MFRC522.py“ eine kleine Änderung vorgenommen, sodass die Funktion `MIFAREReader.MFRC522_Read` einen Rückgabewert hat:

```
331   def MFRC522_Read(self, blockAddr):
332     recvData = []
333     recvData.append(self.PICC_READ)
334     recvData.append(blockAddr)
335     pOut = self.CalulateCRC(recvData)
336     recvData.append(pOut[0])
337     recvData.append(pOut[1])
338     (status, backData, backLen) =
self.MFRC522_ToCard(self.PCD_TRANSCEIVE, recvData)
339     if not(status == self.MI_OK):
340       print "Error while reading!"
341     i = 0
342     #if len(backData) == 16:
343     #  print "Sector "+str(blockAddr)+" "+str(backData)
344     return backData
```

Der Beispielcode sieht dann folgendermaßen aus (die vorherigen Änderungen sind wichtig, da sonst kein Vergleich stattfinden kann):

```
1
2
3
4   #!/usr/bin/env python
5   # -*- coding: utf8 -*-
6   import RPi.GPIO as GPIO
```

```
import MFRC522
def sample_func(sample_var):
    # Beispiel Funktion
    # Skript starten, Daten loggen, etc.
    print("Test Funktion wurde aufgerufen")
# ...
MIFAREReader = MFRC522.MFRC522()
authcode = [114, 97, 115, 112, 98, 101, 114, 114, 121] # die ersten 9 Ziffern sind der Authentifizierungscode

try:

    while True:

        # Scan for cards

        (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)
        # If a card is found
        if status == MIFAREReader.MI_OK:
            # Get the UID of the card
            (status,uid) = MIFAREReader.MFRC522_Anticoll()
            # This is the default key for authentication
            key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
            # Select the scanned tag
            MIFAREReader.MFRC522_SelectTag(uid)
            # Authenticate
            status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
            # Check if authenticated
            if status == MIFAREReader.MI_OK:
```

```
            # Read block 8
            data = MIFAREReader.MFRC522_Read(8)
            if data[:9] == authcode:
                sample_func(data)
            #elif ...
except KeyboardInterrupt:
    print("Abbruch")
    GPIO.cleanup()


```

Wie man sieht ist der Code sehr simpel und soll, wie gesagt, nur als Anstoß dienen um eigene ausgefeiltere Anwendungen damit zu starten. Der Code in Zeile ist natürlich anzupassen

Welche Projekte setzt ihr damit um? Ich überlege ein Türschloss bzw. ein Schubladenschloss/kleinen Tresor mit dem Raspberry Pi RFID Reader zu bauen.