

Primer Puzzle

Itead PN532 NFC amb Ruby i protocol I2C

1. Introducció

La primera tasca del projecte ha estat el desenvolupament del Puzzle 1. En aquest informe detallo tots els passos seguits per completar aquesta primera fase del projecte, incloent-hi totes les configuracions aplicades, les llibreries utilitzades, els problemes que han sorgit i, finalment, el codi realitzat.

L'objectiu del Puzzle 1 del projecte és imprimir per consola el UID (User Identifier) de la tarjeta o clauer Mifare S50, o de la targeta UPC—Infineon Mifare classic 1K— en format hexadecimal i en majúscules.

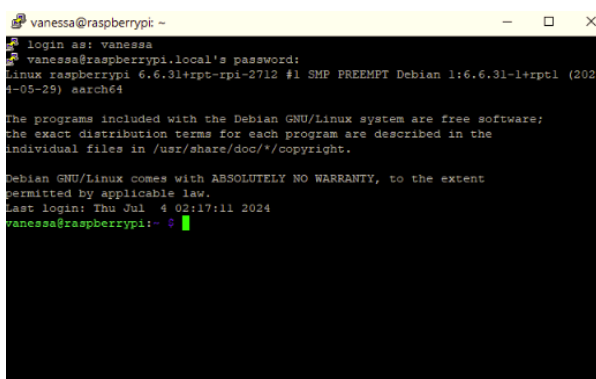
He desenvolupat el projecte amb la Raspberry Pi 5 i el mòdul Itead PN532 NFC, utilitzant el llenguatge de programació Ruby i mitjançant el protocol de comunicació I2C.

2. Connexió

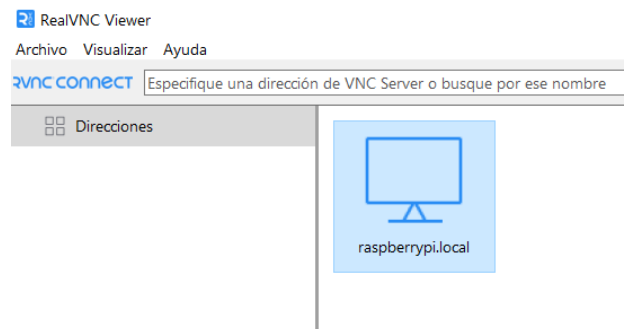
El primer pas del projecte va ser connectar la Raspberry Pi a Internet, per poder crear una connexió des de l'ordinador i accedir-hi. Des del programa Raspberry Pi Imager, a l'hora d'instal·lar el sistema operatiu a la tarjeta microSD, vaig configurar-lo per a que es connectés automàticament al punt de xarxa del meu telèfon mòbil. Així, podia treballar inalàmbricament des del meu ordinador connectat a aquesta mateixa xarxa, sense necessitat de cable Ethernet.

El següent pas va ser accedir al terminal de la Raspberry Pi. Mitjançant el programa [PuTTY](#), vaig crear una connexió remota SSH des del meu ordinador introduint la ip *raspberrypi.local*. Finalment, vaig instal·lar el servidor gràfic [RealVNC](#) i a través de la ip, vaig poder accedir a l'escriptori remot.

Una altra opció que vaig provar per accedir a l'escriptori de la Raspberry Pi va ser mitjançant un cable HDMI i connectant-ho a un monitor amb teclat i ratolí.



Terminal de la Raspberry Pi mitjançant la connexió remota

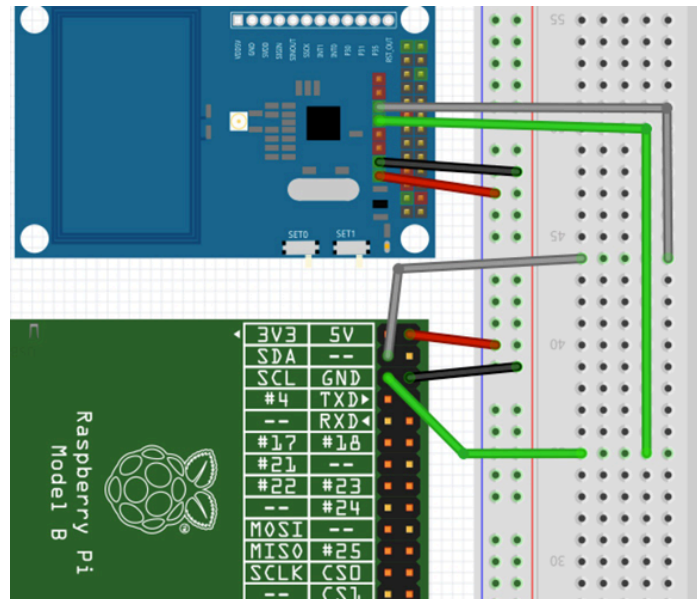


RealVNC amb connexió a raspberrypi.local

3. Comunicació

La comunicació de la Raspberry Pi amb el mòdul NFC la vaig fer mitjançant el protocol I2C. Primer de tot, vaig assegurar-me que els interruptors del mòdul NFC estiguessin en la posició correcta, seleccionant la opció de I2C.

A continuació, vaig connectar físicament el mòdul NFC amb la Raspberry Pi, interconnectant els pins SDA (dades), SCL (rellotge), 5V i Ground.



Imatge extreta de la web oficial de Itead

El següent pas va ser configurar la Raspberry Pi per a poder dur a terme aquest protocol de comunicació amb les següents ordres:

```
→ sudo apt-get update  
sudo apt-get install i2c-tools libi2c-dev
```

Un cop instal·lades les eines del protocol de comunicació I2C, vaig provar una ordre per a comprovar si el mòdul NFC es detectava i efectivament, va aparèixer en pantalla una taula que indicava que el dispositiu s'havia connectat correctament al bus I2C.

```
→ sudo i2cdetect -y 1
```

```
vanessa@raspberrypi: ~  
File Edit Tabs Help  
vanessa@raspberrypi:~$ sudo i2cdetect -y 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: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

El número 24 de la taula indica l'existència d'un dispositiu I2C connectat a la direcció 0x24

4. Lliberies

A continuació, era necessari instal·lar les lliberies relacionades amb el lector NFC. Vaig fer una cerca a Internet i vaig trobar una gema de Ruby que em permetia treballar amb el mòdul, anomenada [*ruby-nfc*](#). Dins del seu repositori de github, vaig trobar tota la informació relacionada amb la seva instal·lació i utilització.

El paquet *ruby-nfc* depèn d'algunes lliberies de *nfc-tools* per funcionar. Per a instal·lar-les vaig introduir les següents ordres al terminal:

```
→ apt-get install libusb-dev

→ git clone https://github.com/nfc-tools/libnfc.git
   cd libnfc
   autoreconf -vis
   ./configure
   make
   sudo make install

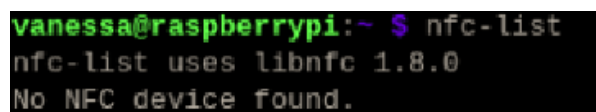
→ git clone https://github.com/nfc-tools/libfreefare.git
   cd libfreefare
   autoreconf -vis
   ./configure
   make
   sudo make install
```

Finalment, vaig instal·lar la gema de Ruby *ruby-nfc*:

```
→ gem install ruby-nfc
```

Un cop vaig tenir instal·lat tot el necessari per a treballar amb el mòdul Itead PN532 NFC, en principi, ja podia començar a detectar i llegir les targetes. Vaig utilitzar les ordres *nfc-list* per a poder analitzar la connexió amb el lector NFC i *nfc-poll*, per llegir les targetes i clauers.

No obstant, en fer aquest pas, em va sorgir un problema: no vaig poder trobar cap dispositiu NFC connectat. Al següent apartat, exposaré la causa d'això i la solució que vaig aplicar.



```
vanessa@raspberrypi:~ $ nfc-list
nfc-list uses libnfc 1.8.0
No NFC device found.
```

Resultat d'aplicar l'ordre *nfc-list*

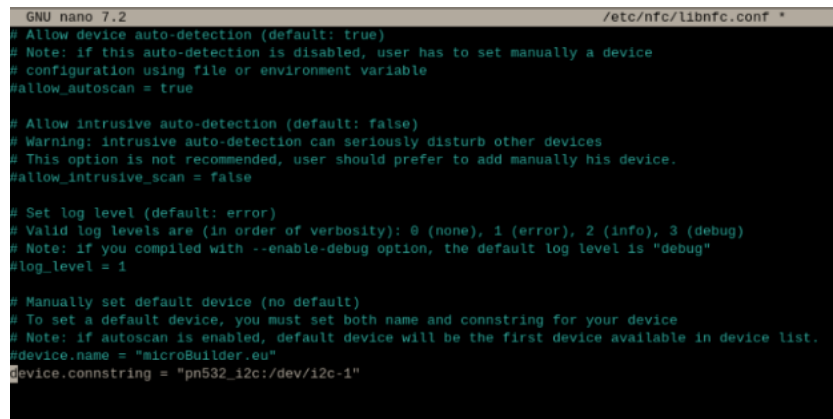
5. Problemes

Durant la realització del Puzzle 1, m'han sorgit tres problemes principals. El primer d'ells, esmentat a l'apartat anterior, va ser el fet de no poder detectar el mòdul NFC mitjançant l'ordre *nfc-list*. Després de cercar a Internet, vaig descobrir que es tractava d'un problema força comú relacionat amb el protocol de comunicació i que tenia una solució senzilla. Executant la instrucció *sudo nano /etc/nfc/libnfc.conf* al terminal de la Raspberry Pi, es va obrir un editor de text amb un fitxer que contenia tota la configuració de la comunicació que feia servir el lector NFC. A una de les línies, apareixia el següent:

→ `device.connstring = "pn532_uart:/dev/ttyUSB0"`

Això indicava una comunicació mitjançant el protocol uart, en lloc de I2C. La solució va ser canviar la configuració per utilitzar una comunicació I2C, de la següent manera:

→ `device.connstring = "pn532_i2c:/dev/i2c-1"`



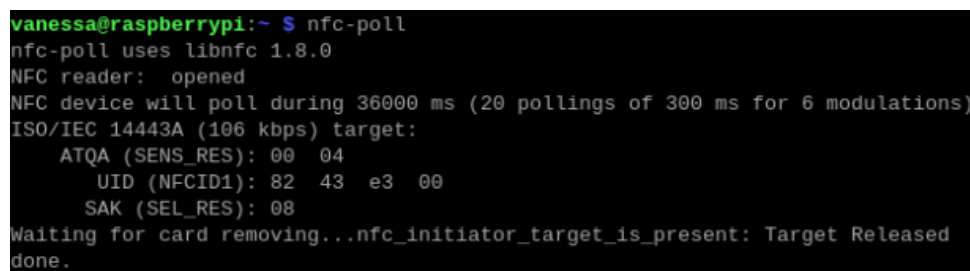
```
GNU nano 7.2 /etc/nfc/libnfc.conf *
# Allow device auto-detection (default: true)
# Note: if this auto-detection is disabled, user has to set manually a device
# configuration using file or environment variable
#allow_autoscan = true

# Allow intrusive auto-detection (default: false)
# Warning: intrusive auto-detection can seriously disturb other devices
# This option is not recommended, user should prefer to add manually his device.
#allow_intrusive_scan = false

# Set log level (default: error)
# Valid log levels are (in order of verbosity): 0 (none), 1 (error), 2 (info), 3 (debug)
# Note: if you compiled with --enable-debug option, the default log level is "debug"
#log_level = 1

# Manually set default device (no default)
# To set a default device, you must set both name and connstring for your device
# Note: if autoscan is enabled, default device will be the first device available in device list.
#device.name = "microBuilder.eu"
device.connstring = "pn532_i2c:/dev/i2c-1"
```

Ara, el *nfc-list* ja permetia detectar el mòdul connectat i a més, utilitzant *nfc-poll*, es podia llegir el contingut de la targeta.



```
vanessa@raspberrypi:~$ nfc-poll
nfc-poll uses libnfc 1.8.0
NFC reader: opened
NFC device will poll during 36000 ms (20 pollings of 300 ms for 6 modulations)
ISO/IEC 14443A (106 kbps) target:
  ATQA (SENS_RES): 00 04
  UID (NFCID1): 82 43 e3 00
  SAK (SEL_RES): 08
Waiting for card removing...nfc_initiator_target_is_present: Target Released
done.
```

Informació de la targeta utilitzant l'ordre *nfc-poll*

No obstant, en el moment de llegir el clauer, el lector no va detectar res. Aquest va ser el segon problema que vaig tenir. Vaig comprovar que el clauer funcionés correctament, mitjançant un altre model diferent de mòdul NFC. També, vaig buscar a Internet si això era resultat d'una mala configuració però no vaig trobar res relacionat. Finalment, vaig atribuir el problema al mòdul utilitzat, el Itead PN532, ja que era possible que no funcionés bé amb el clauer.

Finalment, el tercer problema estava relacionat amb el codi. Vaig fer una primera versió del codi (Veure [apartat 6.1](#)), on aparentment tot era correcte i havia de funcionar. Tot i així, a l'hora de rebre l'output del codi, només apareixien 6 dígit hexadecimals, en comptes dels 8 necessaris. Això era perquè el UID de la tarjeta que jo estava llegint tenia dos zeros que, amb els mètodes de *ruby-nfc*, s'estaven guardant com a *nil* (equivalent de null en Ruby) i no s'imprimien en pantalla com el caràcter correcte, és a dir, '00'. Per a solucionar-ho, vaig haver de fer una nova versió (Veure [apartat 6.2](#)), on a partir d'un bucle, es comprova si algun dels caràcters del UID es guarda com a *nil* i es canvia per un '0'.

6. Codi

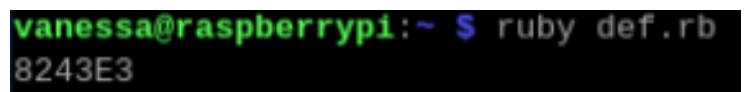
6.1. Versió original

```
require 'ruby-nfc'

class Rfid
  def read_uid
    NFC::Reader.all[0].poll(Mifare::Classic::Tag) do |tag|
      return tag.uid.unpack("H*")[0].upcase
    end
  end
end

if __FILE__ == $0
  rf = Rfid.new
  uid = rf.read_uid
  puts uid
end
```

Output:



```
vanessa@raspberrypi1:~ $ ruby def.rb
8243E3
```

UID de la targeta en format hexadecimal sense
tenir en compte els caràcters '00'

6.2. Versió millorada

```
require 'ruby-nfc'

class Rfid
  def read_uid
    NFC::Reader.all[0].poll(Mifare::Classic::Tag) do |tag|
      uid=""
      8.times do |i|
        if(tag.uid_hex[i]==nil)
          uid+='0'
        else
          uid+=tag.uid_hex[i]
        end
      end
      return uid.upcase
    end
  end
end
```

```
if __FILE__ == $0
  rf = Rfid.new
  uid = rf.read_uid
  puts uid
end
```

Output:

```
vanessa@raspberrypi:~ $ ruby puzzle.rb
8243E300
```

UID de la tarjeta en format hexadecimal