Iria Ceballos

Memòria Puzzle 1

La Raspberry que he utilitzat és la *Raspberry Pi 3B+* i, primer de tot, per a configurar-la, he descarregat la imatge Raspbian Stretch per a instal·lar el sistema operatiu de la placa. Un cop descarregada, he configurat el sistema operatiu, mantenint el hostname com a “raspberrypi”, afegint nom d’usuari i contrasenya, seleccionat com a ‘wireless network’ el punt d'accés al meu telèfon, i activat el servei SSH. Un cop configurat el sistema operatiu i gravat a la microSD, ja es pot inserir la targeta de memòria a la placa, i començar a fer-ne ús.
Per a fer aquesta configuració, he seguit les instruccions de la web https://www.raspberrypi.com/documentation/computers/getting-started.html que se’ns ha proporcionat. Però amb aquesta imatge, la placa no funciona correctament, i no es pot connectar a l’ordinador, aleshores, seguint els passos del pdf https://pip.raspberrypi.com/categories/685-whitepapers-app-notes/documents/RP-003476-WP/Updating-Pi-firmware.pdf , he eliminat dos arxius de la carpeta ‘boot’ de la imatge targeta, i els he substituït per uns de nous proporcionats al pdf. D’aquesta manera, ja s’ha pogut inicialitzar i configurar correctament la raspberry.

Aleshores per a connectar la placa a l’ordinador, he connectat, mitjançant el cable HDMI, la placa a un monitor; i els cables del teclat i el ratolí els he connectat a dues entrades USB qualsevols de la placa. Ja connectada, he instal·lat totes les llibreries necessàries per a programar en ruby, i per a llegir el meu perifèric Itead PN532 NFC.
Primer, m’he instal·lat Ruby amb el comandament

```
sudo apt update
sudo apt install ruby-full
```

Com s’ha instalat una versió bastant antiga, l’he actualitzat fent servir ‘rbenv’, amb el comandament: `rbenv install <versió>`. Jo he escollit la versió 3.4.0. En intentar actualitzar-ho, m’he trobat que primer havia d’instal·lar ‘ruby-build’, les llibreries ‘libffi’ i ‘libyaml’.

```
git clone https://github.com/rbenv/ruby-build.git $(rbenv
root)/plugins/ruby-build
sudo apt-get install -y libffi-dev
sudo apt-get install -y libyaml-dev
```

Em trobo que, tot i que al fer `rbenv install 3.4.0`, em diu que s'ha instalat correctament, al comprovar la versió amb `ruby -v`, em continua dient la versió inicial (2.5.0), per tant, he hagut de fer algunes modificacions per a que es guardés bé l'actualització.

```
echo $SHELL
nano ~/.bashrc
```

Dins aquest arxiu, he afegit:

```
export PATH="$HOME/.rbenv/bin:$PATH"
eval "$(rbenv init -)"
```

Ara, al posar `ruby -v`, ja em diu que la versió és 3.4.0

Tot seguit, intento instal·lar la llibreria 'ruby-nfc' que he trobat a github (https://github.com/hexdigest/ruby-nfc) per a fer servir el meu perifèric (ITEAD PN532 NFC). Ho faig amb el comandament `gem install ruby-nfc` (Havent instalat abans el que em demana com a prerrequisits a la pàgina web). Però em diu que hi ha un error, i que no es pot instal·lar, ja que em cal la versió 1.17.1 de ffi, i per a aquesta, em cal actualitzar la versió de RubyGems(actualment tinc la versió 2.7.6.2 i em cal la 3.3.22).
Per tant, executo: `gem update –system` per a que s'actualitzi RubyGems.
Ara, ja em deixa instal·lar la llibreria 'ruby-nfc'. Comprovo que està, posant `gem list`, i buscant a la llista el nom 'ruby-nfc', i efectivament, apareix.

Per a poder executar el codi, també em cal instal·lar la llibreria 'libusb', 'libfreefare':

```
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libfreefare-dev
```

Aleshores, un cop tot instal·lat, connecto el lector ITEAD a la Raspberry, fent ús de cables dupont femella-femella. Faig les connexions de la següent manera:

- El pin SDA del PN532 al pin SDA de la Raspberry Pi (GPIO 2).
- El pin SCL del PN532 al pin SCL de la Raspberry Pi (GPIO 3).
- El pin 5V del PN532 al pin 3.3V de la Raspberry Pi.

- El pin GND del PN532 al pin GND de la Raspberry Pi.

Per comprovar que el lector està ben connectat i funciona correctament, poso a la terminal nfc-list. Efectivament, apareix el lector a la consola, i si està la targeta/clauer a sobre, apareix a la consola el lector i la targeta/clauer detectat.

Seguidament, redacto el codi necessari per a aconseguir el que demana el puzzle 1, que és un programa que imprimeixi per consola el uid (user identifier) de la targeta o clauer *Mifare S50* o de la targeta UPC (Infineon Mifare classic 1K)
Per a l'elaboració del codi, em baso en l'exemple d'aplicació proporcionat per la llibreria 'ruby-nfc' de github, i l'adapto als meus requisits.

A continuació, la descripció detallada del codi.

```
require 'ruby-nfc'


class Rfid

        def read_uid

                reader = NFC::Reader.all.first

                if reader.nil?
                        return nil
                end

                reader.poll(IsoDep::Tag, Mifare::Classic::Tag, Mifare::Ultralight::Tag) do |tag|
                        begin
                                uid = tag.uid.unpack1('H*')
                                return uid
                        end
                rescue Exception => e
                        return e
                end
        end
end

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

Primerament, s'importa la biblioteca ruby-nfc, que proporciona les funcionalitats necessàries per interactuar amb dispositius NFC (Comunicació de Camp Proper) i llegir etiquetes RFID.

Aleshores, es defineix una classe anomenada Rfid, la qual tindrà les funcionalitats que calen per a llegir les etiquetes.

I dins d'aquesta, declarem el mètode read_uid, que tal com indica el nom, serà un mètode capaç de detectar mitjançant el lector, la targeta o el clauer, i guardarà el seu número identificador a la variable 'uid'.
Per a fer això, en primer lloc, s'obté el primer lector NFC disponible al sistema (com sempre connectarem com a màxim un, si aquest està ben connectat, serà el que obtenim) i s'identifica amb la variable 'reader'. Si no hi ha lectors disponibles, la variable 'reader' serà nil.
Es verifica que s'hagi detectat el lector, de no ser així, el mètode retorna el nil, indicant que no hi ha un lector disponible.
Aleshores, comença un cicle de polling (com un sondeig) per detectar etiquetes RFID. El mètode poll (de la llibreria 'ruby-nfc') cerca etiquetes de tipus específics: IsoDep, Mifare::Classic, i Mifare::Ultralight. Quan es detecta una targeta o clauer, s'executa el bloc de codi següent.
En aquest bloc, s'intenta llegir l'UID de l'etiqueta detectada. L'UID s'obté i es converteix a una representació hexadecimal fent servir *unpack1('H*')*. Després es retorna l'UID.
Si es produeix qualsevol excepció durant el procés de lectura, es captura i es retorna l'objecte d'excepció. Això permet que el mètode no falli silenciosament, sinó que informi sobre l'error.
Finalment, es crea una instància de la classe Rfid, s'anomena el mètode read_uid per llegir l'UID de l'etiqueta i s'imprimeix el resultat a la consola.
Finalment, podem executar el codi a la terminal mitjançant el comandament: ruby puzzle1.rb i comprovem que ens apareix per consola el uid en hexadecimal.