

PUZZLE 1 PBE

El primer *puzzle* de l'assignatura proposat consta de crear un programa que imprimeixi per consola el número d'identificació de la targeta de la UPC en hexadecimal i majúscules, escrit en llenguatge Ruby.

1. Configuració micro sd i connexió a internet

Primer de tot s'ha hagut de configurar la targeta *micro SD* per tal de poder-ne fer ús posteriorment. Tal com se'ns ha indicat, s'ha seguit el procediment de preparació explícit a la pàgina web de la placa, és a dir, la descàrrega de la *Raspberry Pi Imager*, especificar el seu nom d'usuari, contrasenya i l'enllaç a la xarxa entre d'altres. En el cas de la connexió a internet, es va decidir des d'un principi que seria a través de compartir les dades mòbils, i fins al moment, no ha suposat cap problema.

2. Configuració de la Raspberry i del portàtil utilitzat

El model de la placa utilitzada és la *Raspberry Pi 3B+*. Amb aquesta versió, m'hi he trobat amb un problema, ja que a diferència d'altres, necessita una actualització del *firmware* (programari bàsic que permet a un dispositiu funcionar i interactuar amb el maquinari), en concret dos fitxers que s'han hagut de descarregar a partir d'un ZIP que s'adjuntava en una solució proposada per altres usuaris amb el mateix inconvenient.

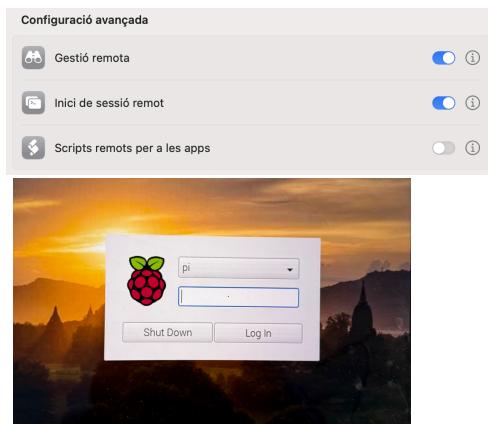
Per tant, un cop configurada la *micro SD*, he modificat el nom dels dos fitxers que no són vàlids per tal de no perdre'ls i he afegit els actualitzats a la carpeta *boot* de la targeta, el **fixup.dat** i l' **start.elf**, extrets del ZIP comentat anteriorment.

Un cop realitzat aquests canvis, ja he pogut inserir la targeta a la placa i inicialitzar-la. S'ha escollit finalment l'opció de fer-ho amb un monitor, ratolí i teclat, a més a més del cable *HDMI* i el carregador. El procediment ha sigut senzill, només ha calgut connectar la Raspberry amb la pantalla a través del cable *HDMI* i a la font d'alimentació, i tot seguit, ja ha aparegut per a poder introduir el nom d'usuari i la contrasenya.

Quan ja he estat a la pantalla de l'escriptori, a través de l'ordre `$ raspi-config` a la Terminal, he pogut activar el servidor VNC per permetre connexions remotes i també SSH per administrar la Raspberry des de la Terminal del meu portàtil de manera que pugui treballar sense la necessitat del monitor. A més a més de connectar-m'hi a la xarxa wifi compartida des de les meves dades mòbils per així utilitzar una adreça IP concreta, la qual he obtingut amb l'ordre introduïda a la Terminal `$ hostname -I`.

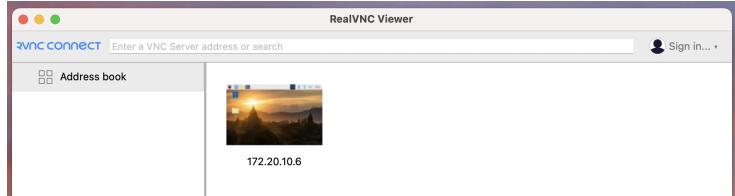
En el cas de la configuració del meu portàtil he hagut habilitar les dues opcions d'inici de sessió i gestió remota:

He decidit tenir les dues opcions de connexió, ja que únicament amb SSH només tinc accés des de la Terminal de la Raspberry, i, en canvi, amb la VNC Viewer, que l'he hagut de descarregar al meu portàtil prèviament, tinc la possibilitat de veure-ho tot per pantalla, així que, si només he d'executar el codi, ho puc fer ràpidament des de la Terminal, però si no, és més còmode veient tot l'escriptori.



Les ordres que s'utilitzen per establir la connexió són les següents:

- Per SSH, a la Terminal del portàtil : `$ ssh pi@direcció_IP_raspberry` o bé, `$ ssh pi@raspberrypi.local` (si s'ha configurat així), i la contrasenya creada.
- Per VNC Viewer : introduir l'adreça IP corresponent i iniciar sessió amb nom d'usuari i contrasenya.



En tots dos casos he de tenir el portàtil connectat a la mateixa xarxa de dades mòbils.

3. Instal·lació i configuració de Ruby, i llibreries utilitzades

Per a poder codificar amb Ruby i a més a més, habilitar que la Raspberry detecti el meu lector de targetes ACR122U s'ha hagut de descarregar un seguit de biblioteques i instal·lar diferents accessos. En el cas de Ruby, s'han escrit diferents ordres des de la Terminal de la Raspberry aconseguint doncs, la versió 3.3.5 que la podem consultar amb `$ ruby -v`.

He hagut d'actualitzar els paquets amb `$ sudo apt update` i `$ sudo apt upgrade`, instal·lar llibreries necessàries per a Ruby amb `$ sudo apt install gcc make nom_llibreria-dev` que han estat: **libssl** (necessària per a algunes funcionalitats de Ruby), **libreadline** (implementa funcions avançades d'edició de línia d'ordres), **zlib1g** (per a compressió de fitxers i dades), **libsdlite3** (útil per a aplicacions Ruby) i **libyaml** (necessària per RubyGems). A més a més, per instal·lar Ruby he utilitzat rbenv (eina d'administració de versions per Ruby), i, per obtenir totes dues, s'ha seguit els passos marcats per: <https://www.theodinproject.com/lessons/ruby-installing-ruby> la qual finalment s'obté la versió rbenv 1.3.2 i ruby 3.3.5.

Per la instal·lació de llibreries per al meu lector ACR122U de targetes, s'ha hagut de tenir en compte un inconvenient el qual aquest tipus de dispositiu no tenia una versió compatible, amb el **firmware** (microprogramari) altra vegada, amb la meva Raspberry, per tant, a més a més de configurar les llibreries necessàries per al meu codi, s'ha hagut de fer una recerca per solucionar aquest problema, però s'explicarà posteriorment a especificar quines llibreries he utilitzat per dur a terme el *puzzle* ja que han estat necessàries per evidenciar aquest imprevist.

He hagut d'instal·lar les llibreries **smartcard** i **ruby-nfc**, pel fet que són les que principalment en faig ús al meu programa, però a més a més, han estat necessaris els paquets **build-essential** (per compilar algunes gemmes), **libffi-dev** (biblioteca per a crides natives) i **libpcslite-dev** (necessari per als lectors). I finalment, instal·lar i activar **pcscd** que és l'encarregat de la comunicació amb el lector, i **pcsc-tools** que proporciona eines per comprovar el funcionament.

Amb aquests seguits d'ordres s'ha pogut comprovar el seu funcionament o bé inicialitzar-lo:

```
$ sudo systemctl start pcscd
$ sudo systemctl enable pcscd
$ sudo systemctl status pcscd
```

I a partir d'aquest procés, si s'escriu la següent línia a la Terminal de la Raspberry hauria d'aparèixer sense problemes el lector comentat anteriorment: `$ pcsc_scan`, i no va ser així, però, en canvi, amb `$ lsusb` que serveix per veure totes les connexions que té la Raspberry, sí que apareixia.

```
pi@raspberrypi:~ $ lsusb
→ Bus 001 Device 005: ID 072f:2200 Advanced Card Systems, Ltd ACR122U
Bus 001 Device 004: ID 0424:7800 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 002: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@raspberrypi:~ $
```

Per tant, per a resoldre-ho, he hagut de descarregar el fitxer ZIP per a Linux facilitat a la pàgina web del lector (*PC/SC Driver Installer*), descomprimir-lo i entrar a la carpeta corresponent, *Raspbian* i *buster* (es pot confirmar amb el command `$ cat /etc/os-release`) i instal·lar el *driver* adjunt a la carpeta. Ara, doncs, si es comprova que `pcscd` està actiu, i s'executa `$ pcsc_scan`, sí que apareix el lector:

```
pi@raspberrypi:~ $ pcsc_scan
Using reader plug'n play mechanism
Scanning present readers...
|: ACS ACR122U 00 00

Sun Mar 2 12:39:26 2025
Reader 0: ACS ACR122U 00 00
Event number: 0
Card state: Card removed,
\ - |
```

4. Codi i sortida per pantalla

Finalment, s'ha implementat un codi per a poder realitzar l'objectiu d'aquest primer *puzzle*:

El programa creat es podria fer amb menys línies de codi, tot i això, he preferit fer-ho amb claredat i separar cada mètode per tal d'entendre-ho millor.

A la captura de pantalla del codi hi ha comentada la funció de cada línia, però en resum, comença inicialitzant-se per a poder fer ús del perifèric i s'estableix com a “buit” de manera que fins que no es detecti un canvi d'estat, per tant, hi ha una targeta, m'espero. Un cop hi ha una targeta esperant per ser llegida, el lector és “connecta” i si no hi ha cap error, llegeix l'uid, el converteix en hexadecimal i tot en majúscules, desactiva la connexió amb la targeta i retorna la variable llegida `uid`.

La sortida per pantalla ha estat:

```
pi@raspberrypi:~/Desktop/pbe $ ruby puzzle1.rb
UID llegit: F6D678829000
pi@raspberrypi:~/Desktop/pbe $
```