

# MEMÒRIA PUZZLE 1

Mòdul: RFID-RC522



Hugo Roldan Lopez

## CONFIGURACIONS PRÈVIES

La RPi utilitzada es la Raspberry Pi 5. Utilitzo una màquina virtual per utilitzar el S.O Linux Ubuntu per treballar amb la RPi.

El primer que he realitzat és descarregar el programa **rpi-imager** per grabar el Sistema Operatiu a la tarjeta microSD de la Raspberry. El que vaig fer inicialment és descarregar el software desde la terminal fent:

**sudo apt install rpi-imager**

Però aquesta comanda te instala una versió que no es la més actual, ja que no et permet definir l'usuari i la contrasenya per accedir a la Raspberry desde el propi programa. Només et permet seleccionar la tarjeta microSD i el S.O. Això em va portar problemes alhora d'accedir mitjançant ssh a la Raspberry ja que la contrasenya per defecte "raspberrypi" no era vàlida. Necessitaria un adaptador micro HDMI - HDMI per entrar a la Raspberry desde un monitor i configurar la contrasenya desde allà.

Finalment, vaig decidir anar a la pròpia pàgina web i descarregar la versió més recent, la qual si et permet configurar l'usuari i la contrasenya a la interfaç del programa. Vaig gravar el S.O a la microSD utilitzant un lector USB 2.0 de targetes.

Després, havia d'accedir a la partició boot de la targeta per generar un fitxer amb títol ssh. D'aquesta manera, quan la Raspberry s'encengues, actives el protocol ssh i ens permetes accedir remotament a la seva terminal. Per fer això, primer vaig haver de trobar com accedir a la partició *boot*. Executo la comanda **lsblk** que et mostra una llista dels dispositius d'emmagatzematge connectats.

```
loop11  7:11  0 460,6M  1 loop  /snap/gnome-42-2204/102
loop12  7:12  0 505,1M  1 loop  /snap/gnome-42-2204/176
loop13  7:13  0  91,7M  1 loop  /snap/gtk-common-themes/1535
loop14  7:14  0  37,1M  1 loop  /snap/hunspell-dictionaries-1-7-2004/2
loop15  7:15  0 286,5M  1 loop  /snap/octave/306
loop16  7:16  0  12,2M  1 loop  /snap/snap-store/1216
loop17  7:17  0  12,3M  1 loop  /snap/snap-store/959
loop18  7:18  0  44,3M  1 loop  /snap/snapd/23258
loop19  7:19  0  44,4M  1 loop  /snap/snapd/23545
loop20  7:20  0   568K  1 loop  /snap/snapd-desktop-integration/253
loop21  7:21  0   452K  1 loop  /snap/snapd-desktop-integration/83
sda      8:0    0 30,7G   0 disk
├─sda1   8:1    0    1M   0 part
├─sda2   8:2    0   513M  0 part  /boot/efi
└─sda3   8:3    0 30,2G   0 part  /var/snap/firefox/common/host-hunspell
sdb      8:16   1    0B   0 disk
sdc      8:32   1 59,5G   0 disk
├─sdc1   8:33   1   512M  0 part
└─sdc2   8:34   1    5,1G  0 part
sr0     11:0    1 1024M   0 rom
```

Identifico que s'han generat dos particions de la microSD, sdc1 i sdc2. La partició de *boot*, que està a sdc1 no s'ha montat automàticament i per tant no es visible, la vaig haver de montar manualment amb la comanda:

**sudo mount /dev/sdc1 /mnt**

Aquesta comanda monta el contingut de `sdcl` al directori `/mnt` per poder accedir al contingut de *boot*. La partició `sdcl` es troba al directori `/dev`. Ara faig `cd /mnt` i creo l'archiu buit `ssh` fent **touch ssh**. Per seguretat, executo **sudo umount /mnt** un cop fetes les configuracions.

Ara ja es pot introduir la targeta a la Raspberry.

## CONNEXIÓ AMB LA RASPBERRY

Per connectarme a la Raspberry he utilitzat un cable Ethernet. Conecto el portàtil mitjançant un adaptador USB-Ethernet a la Raspberry. Seguidament, a l'apartat de **Red** del sistema, selecciono la red **USB-Ethernet** y la configuro com **Compartida con otros Usuarios**. D'aquesta manera, el portàtil actuarà com un servidor DHCP i assignarà automàticament una direcció IP a la Raspberry.

Per trobar la IP assignada i comprovar que la connexió és correcta, executo la comanda **ping raspberrypi.local**. Vaig obtenir el següent:

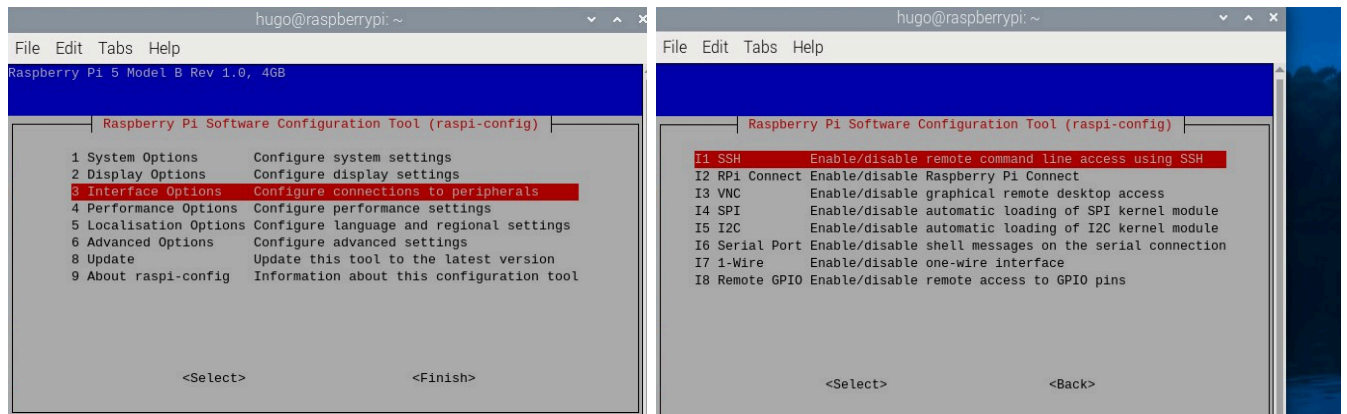
```
hugo@hugo-VirtualBox:~$ ping raspberrypi.local
PING raspberrypi.local (10.42.0.138) 56(84) bytes of data.
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=1 ttl=64 time=3.50 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=3 ttl=64 time=1.68 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=4 ttl=64 time=1.49 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=8 ttl=64 time=2.43 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=10 ttl=64 time=2.87 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=11 ttl=64 time=2.46 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=15 ttl=64 time=1.32 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=17 ttl=64 time=2.97 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=18 ttl=64 time=2.12 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=21 ttl=64 time=0.519 ms
64 bytes from 10.42.0.138 (10.42.0.138): icmp_seq=23 ttl=64 time=3.24 ms
^C
--- raspberrypi.local ping statistics ---
24 packets transmitted, 11 received, 54.1667% packet loss, time 23349ms
rtt min/avg/max/mdev = 0.519/2.236/3.499/0.869 ms
```

Per tant la IP de la Raspberry és **10.42.0.138**. També vaig adonar-me que el cable Ethernet no estava ben connectat ja que hi havia un 54 % de paquets perduts.

Seguidament, executo la comanda **ssh hugo@10.42.0.138** per accedir a la Raspberry amb el usuari configurat (hugo).

Ara per poder accedir a l'escriptori virtual habilito el vnc.

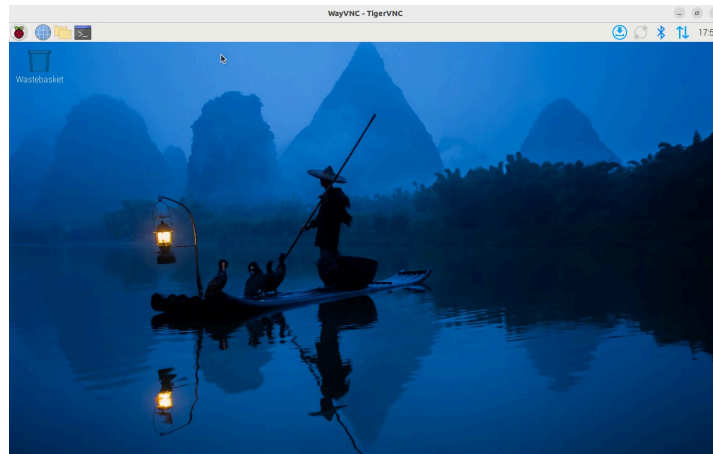
Executo la comando **sudo raspi-config** i se m'obre una interfaz on fàcilment es pot activar el vnc.



Després reinicio la Raspberry amb **sudo reboot**.

Desde la terminal del portàtil executo **sudo apt install tigervnc-viewer** per instal·lar el paquet necessari per visualitzar l'escriptori de la Raspberry remotament.

Finalment executo **vncviewer 10.42.0.138** i ja s'obrirà l'escriptori virtual.



## PUZZLE 1

Un cop fetes les configuracions, podem començar a fer el puzzle1. El S.O de la Raspberry ja inclou Python3 instal·lat a la versió 3.11 per defecte així que directament podem desde la terminal crear un fitxer de tipus Python amb **vim puzzle1.py** descarregant prèviament vim amb **sudo apt install vim**. Descarrego el paquet de llibreries mfrfc522 desde el repositori en línia de Python PyPI amb **pip3 install mfrfc522**. Però obtinc la següent resposta:

```

hugo@raspberrypi:~/PBE/Puzzles $ sudo pip3 install mfrc522
error: externally-managed-environment

× This environment is externally managed
╰─> To install Python packages system-wide, try apt install
    python3-xyz, where xyz is the package you are trying to
    install.

    If you wish to install a non-Debian-packaged Python package,
    create a virtual environment using python3 -m venv path/to/venv.
    Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make
    sure you have python3-full installed.

    For more information visit http://rptl.io/venv

note: If you believe this is a mistake, please contact your Python installation
or OS distribution provider. You can override this, at the risk of breaking your
Python installation or OS, by passing --break-system-packages.
hint: See PEP 668 for the detailed specification.

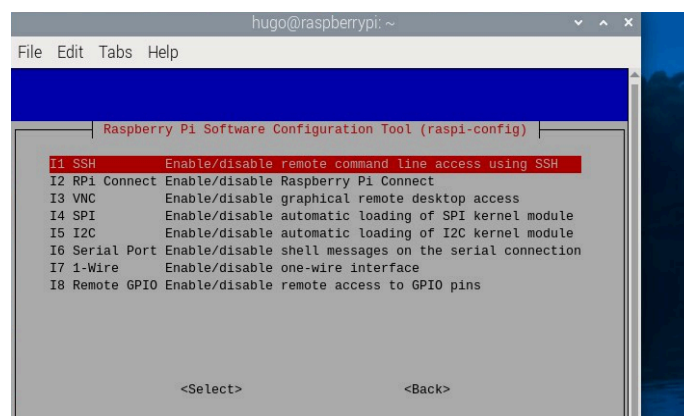
```

Vaig llegir la part de **note** i descobrir que si afegia el flag **--break-system-packages**, podia descarregar el paquet. L'únic problema es que això instala el paquet a un entorn global del sistema, el que podria generar problemes de compatibilitat amb el gestor de paquets apt o sobrescriure paquets essencials del sistema. Com sóc conscient que això no succeirà al instal·lar aquestes llibreries, ho faig.

D'aquest conjunt de llibreries, faré servir la **SimpleMFRC522** ja que es la que simplifica més el procés de lectura.

També instalo el paquet **spidev** per tractar amb el protocol SPI: **sudo pip3 install spidev --break-system-packages**.

Finalment, habilito la interfaz SPI a la RPi fent **sudo raspi-config** de la mateixa manera que per activar el VNC, pero ara seleccionant SPI.

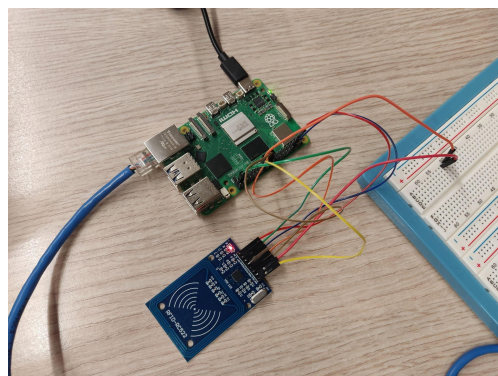


## CONNEXIONS FÍSQUES

Per comunicar la Raspberry amb el mòdul mfr522, s'utilitza el protocol SPI. La configuració de pins es pot veure a la següent taula:

RF522 Module	Raspberry Pi
SDA	Pin 24 / GPIO8 (CE0)
SCK	Pin 23 / GPIO11 (SCKL)
MOSI	Pin 19 / GPIO10 (MOSI)
MISO	Pin 21 / GPIO9 (MISO)
IRQ	-
GND	GND
RST	Pin 22 / GPIO25
3.3V	3.3V

La següent imatge mostra les connexions físiques. He fet servir la protoboard ja que necessitava 7 cables femella-femella però en aquell moment només tenia 6.



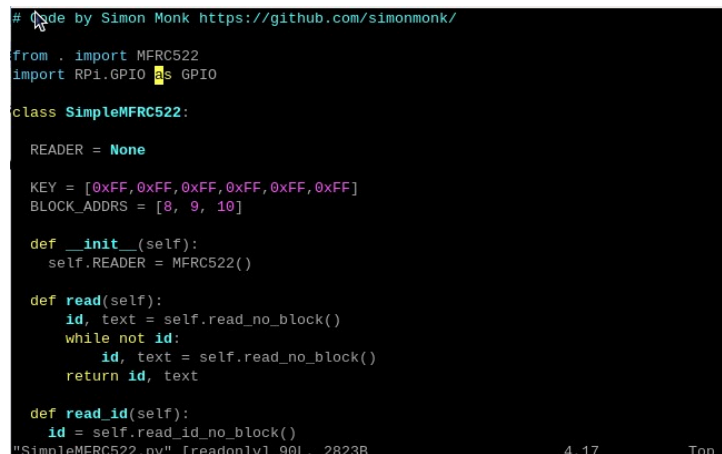
## CODI

Malauradament, la funcionalitat GPIO de la Raspberry Pi 5 a diferència de la RPi 4, no pot tractar-se amb la llibreria RPi.GPIO, cal utilitzar una altre llibreria anomenada *lgpio*. El problema que he tingut es que la llibreria mfr522 està feta a partir de la llibreria RPi.GPIO. Vaig buscar llibreries del mfr522 adaptades a la Raspberry Pi 5 sense èxit. Així que vaig decidir modificar-la manualment. Per fer això vaig accedir al directori on es troba la llibreria. El procés es mostra a la imatge següent:

```
hugo@raspberrypi:~ $ cd /usr
hugo@raspberrypi:/usr $ ls
bin  games  include  lib  libexec  local  sbin  share  src
hugo@raspberrypi:/usr $ cd local
hugo@raspberrypi:/usr/local $ ls
bin  etc  games  include  lib  man  sbin  share  src
hugo@raspberrypi:/usr/local $ cd lib
hugo@raspberrypi:/usr/local/lib $ ls
python3.11
hugo@raspberrypi:/usr/local/lib $ cd python3.11
hugo@raspberrypi:/usr/local/lib/python3.11 $ ls
dist-packages
hugo@raspberrypi:/usr/local/lib/python3.11 $ cd dist-packages/
hugo@raspberrypi:/usr/local/lib/python3.11/dist-packages $ ls
mfr522  mfr522-0.0.7.dist-info  RPi  rpi_gpio-0.7.1.dist-info
hugo@raspberrypi:/usr/local/lib/python3.11/dist-packages $ cd mfr522
hugo@raspberrypi:/usr/local/lib/python3.11/dist-packages/mfr522 $ ls
__init__.py  MFRC522.py  __pycache__  SimpleMFRC522.py
hugo@raspberrypi:/usr/local/lib/python3.11/dist-packages/mfr522 $
```



Ara, vaig entrar al fitxer SimpleMFRC522.py fent **sudo vim SimpleMFRC522.py** (sudo perquè si no, no et permet editar el fitxer).

A screenshot of a terminal window showing the code of the SimpleMFRC522.py file. The code is written in Python and includes imports for MFRC522 and RPi.GPIO. It defines a class SimpleMFRC522 with attributes like READER, KEY, and BLOCK\_ADDRS. The class has methods for \_\_init\_\_, read, and read\_id. The terminal shows the file is being edited in vim, with the cursor at the end of the line 'import RPi.GPIO as GPIO'.

```
# Code by Simon Monk https://github.com/simonmonk/

from . import MFRC522
import RPi.GPIO as GPIO

class SimpleMFRC522:

    READER = None

    KEY = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
    BLOCK_ADDRS = [8, 9, 10]

    def __init__(self):
        self.READER = MFRC522()

    def read(self):
        id, text = self.read_no_block()
        while not id:
            id, text = self.read_no_block()
        return id, text

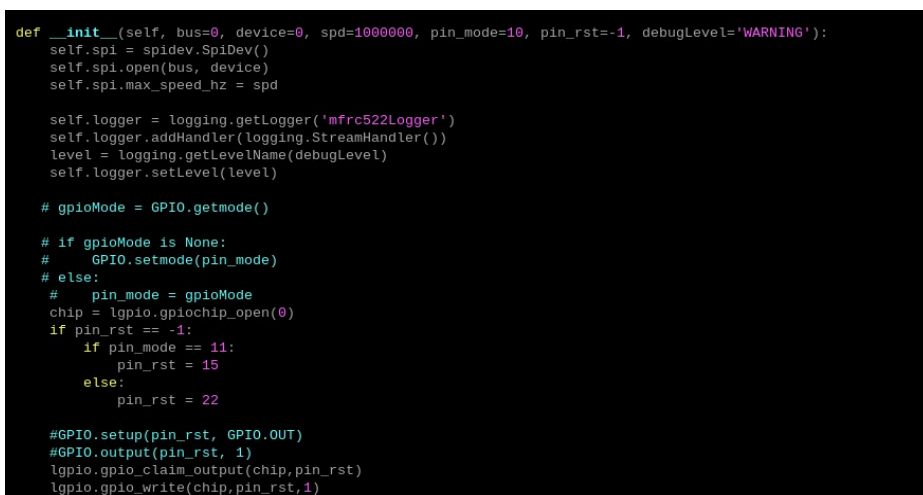
    def read_id(self):
        id = self.read_id_no_block()
```

*Fitxer SimpleMFRC522.py de la llibreria mfrc522*

Aquí modifico el “**import RPi.GPIO as GPIO**” per “**import lgpio as GPIO**”.

Però també cal modificar el fitxer **MFRC522.py** ja que es aquí on es realitzen totes les inicialitzacions per fer funcionar el mòdul.

Afortunadament, només vaig haver de modificar la primera funció **\_\_init\_\_** del fitxer. Vaig fer les modificacions necessàries per adaptar aquesta funció a l'ús de la llibreria *lgpio*:

A screenshot of a terminal window showing the \_\_init\_\_ function of the MFRC522.py file. The function is decorated with a docstring and includes several initialization steps. It sets up the SPI interface, configures logging, and initializes the GPIO pin. The code uses the lgpio library for GPIO operations. The terminal shows the function definition with its parameters and the initialization logic.

```
def __init__(self, bus=0, device=0, spd=1000000, pin_mode=10, pin_rst=-1, debugLevel='WARNING'):
    self.spi = spidev.SpiDev()
    self.spi.open(bus, device)
    self.spi.max_speed_hz = spd

    self.logger = logging.getLogger('mfrc522Logger')
    self.logger.addHandler(logging.StreamHandler())
    level = logging.getLevelName(debugLevel)
    self.logger.setLevel(level)

    # gpioMode = GPIO.getmode()

    # if gpioMode is None:
    #     GPIO.setmode(pin_mode)
    # else:
    #     pin_mode = gpioMode
    chip = lgpio.gpiochip_open(0)
    if pin_rst == -1:
        if pin_mode == 11:
            pin_rst = 15
        else:
            pin_rst = 22

    #GPIO.setup(pin_rst, GPIO.OUT)
    #GPIO.output(pin_rst, 1)
    lgpio.gpio_claim_output(chip, pin_rst)
    lgpio.gpio_write(chip, pin_rst, 1)
```

*Funció \_\_init\_\_ dintre del fitxer MFRC522.py*

Les línies corresponents al codi corresponent a RPi.GPIO estàn comentades. Vaig afegir les línies amb la sintaxis de *lgpio*. Les línies afegides són:

```
chip = lgpio.gpiochip_open(0)
lgpio.gpio_claim_output(chip, pin_rst)
lgpio.gpio_write(chip, pin_rst, 1)
```

No entraré en detall de la funció de cada línia, simplement comentar que substitueixen les línies corresponents a RPi.GPIO.

Ara la llibreria mfrc522 ja està adaptada perquè sigui compatible amb la Raspberry Pi 5.

Finalment, executo **sudo python3 puzzle1.py**, apropo el clauer i obtinc la lectura del seu uid en format hexadecimal i majúscules.

```
hugo@raspberrypi:~ $ cd PBE/  
hugo@raspberrypi:~/PBE $ cd Puzzles/  
hugo@raspberrypi:~/PBE/Puzzles $ sudo python3 puzzle1.py  
uid: 135B8414  
hugo@raspberrypi:~/PBE/Puzzles $
```

El codi utilitzat es mostra a continuació:

Python

```
from mfrc522 import SimpleMFRC522  
class Rfid_522:  
    def __init__(self):  
        self.uid = None  
        self.reader = SimpleMFRC522()  
    def read_uid(self):  
        self.uid = hex(self.reader.read_id())[2:][:8].upper()  
if __name__ == "__main__":  
    rf = Rfid_522()  
    rf.read_uid()  
    print(f"uid:{rf.uid}")
```

La versió comentada es troba al GitHub.



