

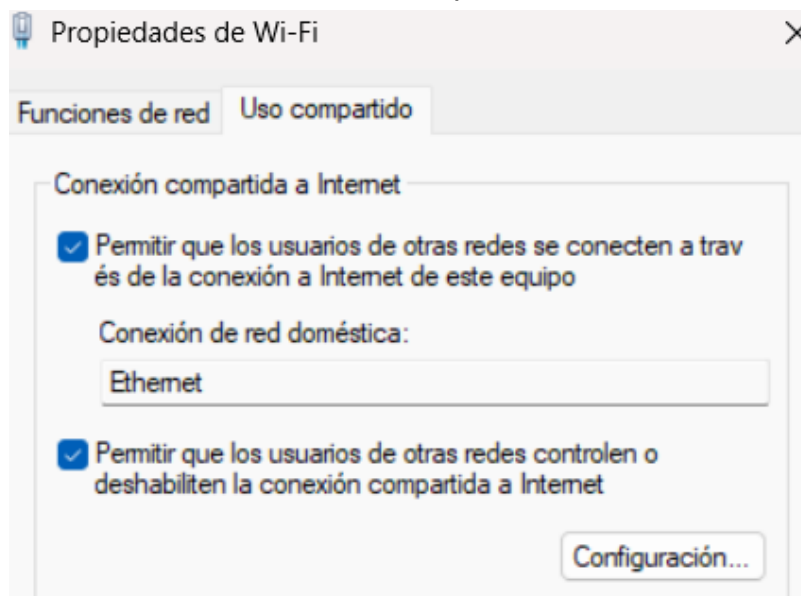
Memoria Puzzle 1

1. Conexiones a internet probados

Para que nuestra RP tenga conexión a internet podríamos conectar directamente mediante un cable ethernet al router. Luego enchufamos los periféricos como el ratón, el teclado y el monitor. Así ya podríamos trabajar con la RP. Esta sería una solución cuando trabajo en casa, pero si estuviera en la uni lo mejor sería activar el SSH de la RP y trabajar sin tantos cables y periféricos.

Mi opción ha sido conectar mediante un cable ethernet mi portátil y la RP. De esta manera mi portátil se encargaría de funcionar como si fuera un router.

Primero he activado el uso compartido de conexión a internet.



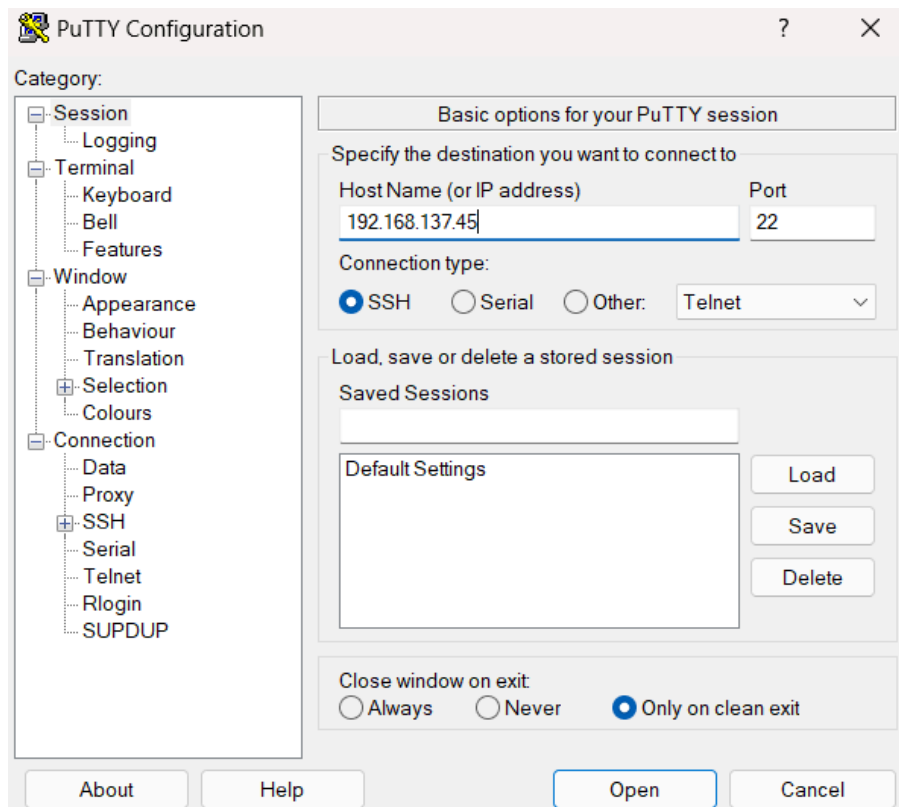
Luego mediante el comando “arp -a” en la cmd de windows puedo obtener la dirección IP que automáticamente ha asignado mediante DHCP nuestro sistema. Observo que en mi caso ha sido “192.168.137.45”.

```
C:\Users\guang>arp -a

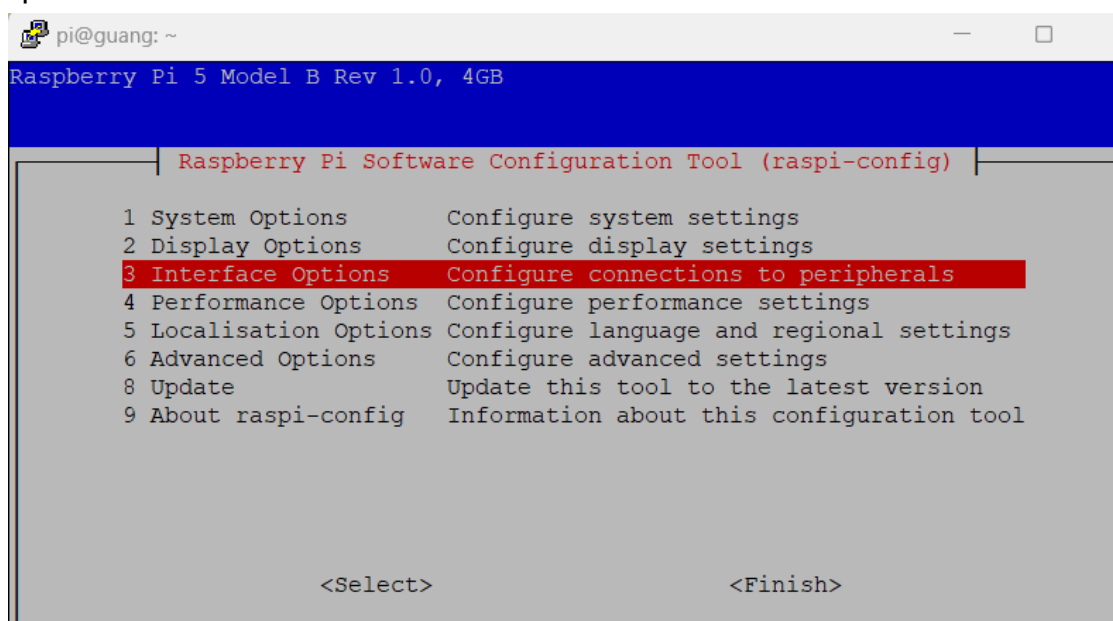
Interfaz: 192.168.137.1 --- 0x2
Dirección de Internet    Dirección física    Tipo
192.168.137.45          d8-3a-dd-aa-1a-c0  estático
192.168.137.255         ff-ff-ff-ff-ff-ff  estático
224.0.0.22              01-00-5e-00-00-16  estático
224.0.0.251             01-00-5e-00-00-fb  estático
224.0.0.252             01-00-5e-00-00-fc  estático
239.255.255.250         01-00-5e-7f-ff-fa  estático
255.255.255.255         ff-ff-ff-ff-ff-ff  estático

Interfaz: 10.192.180.150 --- 0x14
Dirección de Internet    Dirección física    Tipo
10.192.1.1              00-00-0c-07-ac-b2  dinámico
224.0.0.22              01-00-5e-00-00-16  estático
224.0.0.251             01-00-5e-00-00-fb  estático
224.0.0.252             01-00-5e-00-00-fc  estático
239.255.255.250         01-00-5e-7f-ff-fa  estático
255.255.255.255         ff-ff-ff-ff-ff-ff  estático
```

Ahora con un software llamado putty puedo acceder a la terminal de la RP.

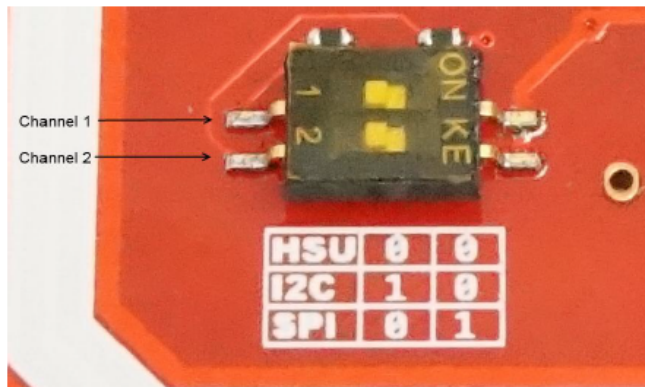


Pero es muy incómodo trabajar sólo con la terminal. Por eso, necesitamos un entorno gráfico, que en nuestro caso vamos a utilizar la RealVNCViewer. La RP ya viene con la VNC instalada, pero por defecto no está activado. Así que lo vamos a activar desde la terminal poniendo el comando “sudo raspi-config”, en interface options activamos VNC.



Ahora con que pongamos el comando “vncserver-virtual” ya podríamos visualizar la RP desde RealVNCViewer.

2. Configuraciones realizadas en la RPi



<i>Working Interface</i>	<i>Channel 1</i>	<i>Channel 2</i>
HSU	OFF	OFF
I2C	ON	OFF
SPI	OFF	ON

Como que el periférico que me ha tocado es el Elechouse y lo necesito conectar mediante UART. Observamos en la imagen de arriba que necesitamos configurar los dos canales a OFF para usar UART, pero en realidad por defecto ya nos viene así. Ahora tenemos que activar con el comando “sudo raspi-config” en la interface options la conexión serial port.

Una vez hecho esto vamos a conectar físicamente la RP con elechouse.

3v3 Power	1	•	•	2	5v Power
GPIO 2 (CTS / Clear to Send)	3	•	•	4	5v Power
GPIO 3 (RTS / Request to Send)	5	•	•	6	Ground
GPIO 4 (TXD / Transmit)	7	•	•	8	GPIO 14 (TXD / Transmit)
Ground	9	•	•	10	GPIO 15 (RXD / Receive)
GPIO 17 (RTS / Request to Send)	11	•	•	12	GPIO 18 (PCM CLK)
GPIO 27	13	•	•	14	Ground
GPIO 22	15	•	•	16	GPIO 23
3v3 Power	17	•	•	18	GPIO 24
GPIO 10 (CTS / Clear to Send)	19	•	•	20	Ground
GPIO 9 (RXD / Receive)	21	•	•	22	GPIO 25
GPIO 11 (RTS / Request to Send)	23	•	•	24	GPIO 8 (TXD / Transmit)
Ground	25	•	•	26	GPIO 7 (RTS / Request to Send)
GPIO 0 (TXD / Transmit)	27	•	•	28	GPIO 1 (RXD / Receive)
GPIO 5 (RXD / Receive)	29	•	•	30	Ground
GPIO 6 (CTS / Clear to Send)	31	•	•	32	GPIO 12 (TXD / Transmit)
GPIO 13 (RXD / Receive)	33	•	•	34	Ground
GPIO 19 (PCM FS)	35	•	•	36	GPIO 16 (CTS / Clear to Send)
GPIO 26	37	•	•	38	GPIO 20 (PCM DIN)
Ground	39	•	•	40	GPIO 21 (PCM DOUT)

Elechouse	Raspberry Pi
Gnd	6 (Ground)
Vcc	4 (5V power)
SDA / Tx	10 (GPIO 15, RX)
SCL / Rx	8 (GPIO 14, TX)

Podemos ver con la imagen y la tabla de arriba las conexiones que tenemos que realizar.

3. Bibliotecas probadas

```

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 = true

# 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
#device.name = "microBuilder.eu"
device.connstring = "pn532_uart:/dev/ttyAMA0"
[ Read 20 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line

```

Para realizar este paso he seguido los pasos que he encontrado en “Nfc-tools Reference Manual”. Por lo que he descargado las librerías “libnfc”, que lo vamos a usar para la lectura de las tarjetas mediante tecnología NFC y “mfoc” que es una implementación de código abierto del ataque “anidado sin conexión” de Nethemba.

Lo que nos permite recuperar claves de autenticación de la tarjeta MIFARE Classic. Con el comando “sudo nano /etc/nfc/libnfc.conf” cambiamos la última línea por “device.connstring = “pn532_uart:/dev/ttyAMA0”.

```

pi@guang:~/PBE $ sudo gem install ruby-nfc
    Don't forget to install libnfc and libfreefare
    see installation instructions here:
    https://github.com/hexdigest/ruby-nfc
Successfully installed ruby-nfc-1.6
Parsing documentation for ruby-nfc-1.6
Done installing documentation for ruby-nfc after 0 seconds
1 gem installed

```

Como que mi grupo nos ha tocado el lenguaje ruby, vamos a instalar la librería ruby-nfc con este comando “sudo gem install ruby-nfc”. Al poner este comando nos sugiere la instalación de la librería “libfreefare” que nos dice las instrucciones a seguir con la página de enlace “<https://github.com/hexdigest/ruby-nfc>”.

4. Problemas encontrados

```

pi@guang:~/PBE $ sudo nano /etc/nfc/libnfc.conf
pi@guang:~/PBE $ nfc-scan-device
nfc-scan-device uses libnfc 1.8.0
1 NFC device(s) found:
- :
    pn532_uart:/dev/ttyAMA0
pi@guang:~/PBE $ nfc-list
nfc-list uses libnfc 1.8.0
NFC device: opened
pi@guang:~/PBE $ 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)
No target found.
pi@guang:~/PBE $

```

Como la imagen mostrada, al utilizar los comandos “nfc-scan-device” y “nfc-list” podemos observar que la RP sí que detecta el dispositivo NFC. Pero al comprobar con el comando “nfc-poll” y pasar la tarjeta por elechouse la RP no es capaz de leer el uid de la tarjeta. Me aparece el mensaje “No target found”.

He probado muchas posibles soluciones que se me ha ocurrido:

- Reinstalar la librería nfc (que a mi compañero que hacía ITEAD con I2C le ha funcionado).
- Cambiar los cables dupont hembra por unos otros nuevos.
- Cambiar el sistema de comunicación de UART por I2C(me detecta al hacer i2cdetect -y 1 pero con nfc-poll tampoco funciona)
- He medido con un multímetro la tensión y la resistencia entre vcc y gnd del elechouse, y puedo observar la tensión de 5V y una resistencia de unos 4k Ohmios.
- He reinstalado el sistema operativo del raspberry y hacer todo de nuevo
- He probado en la configuración añadir "allow_intrusive_scan = true, allow_autoscan = true", para forzar la detección. Pero sigue igual con el mismo problema.

En varias ocasiones al probar nfc-poll y acercar la tarjeta directamente se me apagaba la RP. Entonces recordé que la primera vez que uní los pines GPIO de la RP con elechouse puse los cables de vcc y gnd al revés. Después de ver que salía humo y olor a quemado, desenchufé de seguida todo. La duración de este proceso ha sido de unos 3 segundos. Por lo que sospecho que el problema que tenía era que se había quemado mi elechouse.

5. Código

```
leer.rb
1  require 'ruby-nfc'
2
3  class Rfid
4    def initialize
5      @readers = NFC::Reader.all
6      if @readers.empty?
7        puts "No se encontraron lectores NFC."
8        exit
9      end
10   end
11
12   def read_uid
13     @readers[0].poll(Mifare::Classic::Tag) do |tag|
14       begin
15         return tag.uid_hex.upcase
16       rescue StandardError => e
17         puts "Error al leer la tarjeta: #{e.message}"
18         return nil
19       end
20     end
21     nil
22   end
23 end
24
25 if __FILE__ == $0
26   rf = Rfid.new
27   puts "Acerque la tarjeta NFC..."
28   uid = rf.read_uid
29
30   if uid
31     puts "UID de la tarjeta: #{uid}"
32   else
33     puts "Error al leer la tarjeta."
34   end
35 end
```

He instalado en la RP el software “Visual Studio Code” para programar un pequeño script en ruby que imprima la UID por la pantalla en hexadecimal.

Aquí adjunto una imagen del código ruby que utilizaré. Pero no puedo probar su correcto funcionamiento ya que mi elechouse se ha quemado.