Sílvia Ruano Ferrer
PBE – Puzzle 1

Memòria Puzzle 1

Elechouse PN532 NFC module

1. Objectiu

L'objectiu d'aquesta primera part del projecte era primer connectar i configurar la raspberry tal que es pugui utilitzar des d'un ordinador portàtil sense necessitat d'un monitor extern, la connexió del perifèric escollir en aquest cas el PN532 NFC module de Elechouse i, finalment, programar en ruby un programa que imprimeixi per consola el uid, user identifier, de la targeta o clauer Mifare S50 o la targeta UPC. Aquest uid s'ha d'imprimir en hexadecimal i en majúscules.

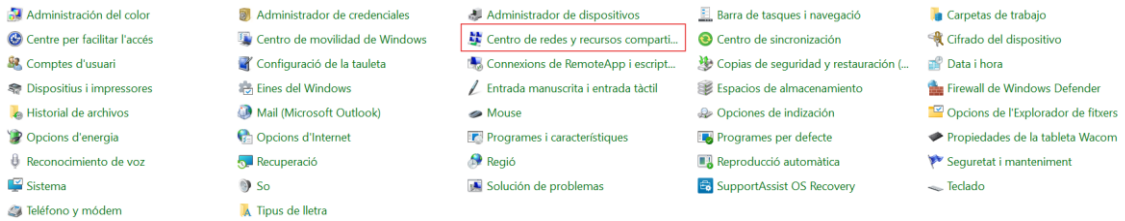
2. Connexió a Internet

La primera configuració d'internet que vaig provar va ser directament per wifi, des del Raspberry Pi Imager es podia configurar ja una connexió a internet a través de wifi, i en connectar l'ordinador portàtil a la mateixa xarxa ja em va deixar entrar a consola i poder configurar-la.

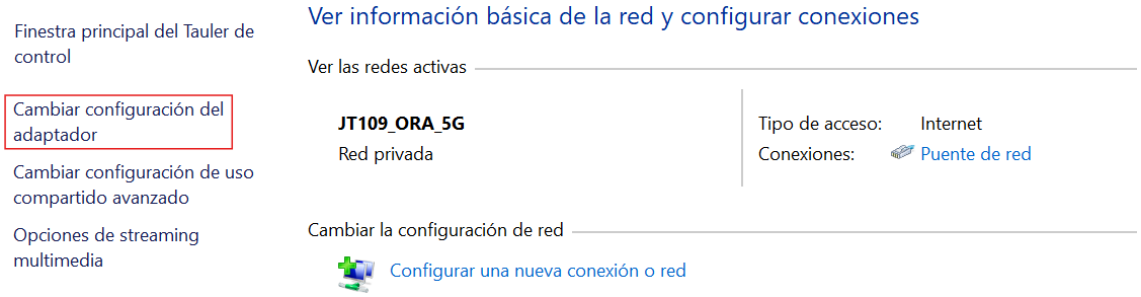
El problema va estar que des de la universitat aquest mètode no és viable pel tipus de seguretat que té la xarxa, tret que fos a través de la xarxa compartida del telèfon mòbil propi. Llavors vaig haver d'implementar un mètode a través d'un cable Ethernet i un adaptador a port USB per la falta de port Ethernet en el meu ordinador.

En tenir l'ordinador i la Raspberry connectats per Ethernet, vaig aplicar el mètode de crear un pont entre connexions, mètode bridge, ja que des del Windows 11 del meu portàtil la compartició de connexió pot ser molt inestable. Llavors, per crear aquest pont vaig haver d'anar al Tauler de Control del meu dispositiu i seguir els següents passos:

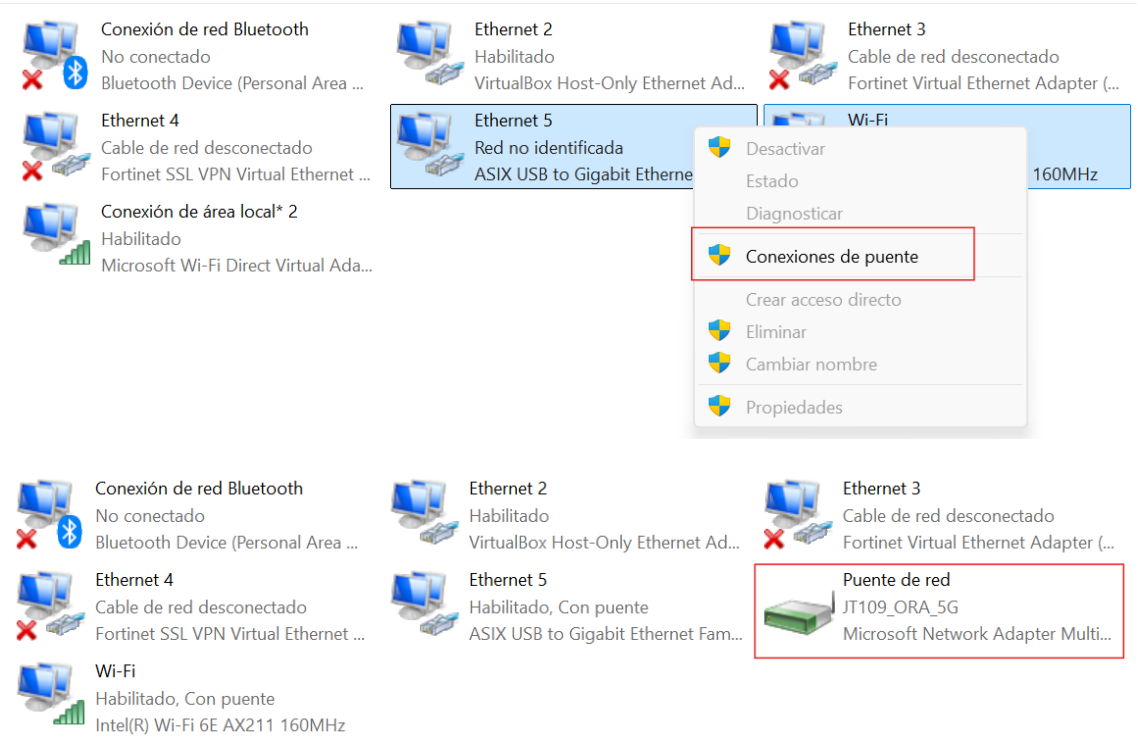
1. Anar al *Centro de Redes y Recursos Compartidos*



2. Clicar a *Cambiar la Configuración del Adaptador*



3. S'obrirà una nova finestra, *Conexiones de Red*, i allà vaig haver d'identificar quin dels 5 Ethernets que em sortien era el que havia de fer-li un pont. Desconnectant i connectant el cable d'Ethernet i veient com canviaven els estats de les connexions vaig veure que el que havia d'utilitzar era l'Ethernet 5. Sabent això, vaig haver de seleccionar tant l'Ethernet com la wifi i clicant botó dret et surt l'opció de *Conexiones de Puente*. En clicar-ho es crearà un *Puente de Red*.



Amb tot això ja es pot desactivar la connexió wireless de la Raspberry Pi, ja que es connecta a l'ordinador i a internet perfectament a través del cable Ethernet i el pont de xarxa.

3. Configuració de la Raspberry Pi

Per poder fer tota la configuració sense haver de connectar la Raspberry a un monitor extern amb HDMI, en el Raspberry Pi Imager em vaig assegurar que tingués activada l'opció que habilita el SSH, per poder després accedir a la terminal sense problemes.

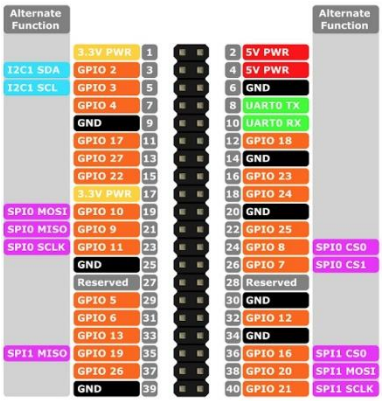
Un cop connectada la Raspberry Pi amb l'ordinador, ja podia començar la configuració d'aquesta. Primer vaig provar d'entrar a la consola de la Raspberry a través del terminal del meu ordinador amb l'ordre: ssh pi@raspberrypi.local. Però no m'acceptava la contrasenya de l'usuari, al posar la correcta. Per això vaig recorre a PuTTy, un emulador de terminal per connectar-se a través de SSH. En aquest programa li poses el nom del host, en aquest cas Raspberrypi.local i t'obre la terminal de la Raspberry Pi on vaig poder iniciar sessió sense cap problema.

Amb la terminal oberta, havia d'habilitar el VNC i així poder accedir a l'escriptori de forma remota. Això ho vaig fer amb l'ordre: sudo raspi-config , anant a *Interface Options*, activar el VNC. Amb això activat ja podíem tancar la pantalla que s'havia obert, i instal·lar en el meu ordinador un VNC *viewer*, en el meu cas vaig optar per *RealVNC viewer.* Des d'aquest tornes a posar el nom del host iniciant sessió amb l'usuari adient i ja vaig poder obrir l'escriptori de la meva Raspberry Pi.

Per poder després programar amb ruby ens hem d'assegurar que estigui instal·lat, per això utilitzem l'ordre: ruby -v , amb això vaig saber que no el tenia instal·lat llavors amb sudo apt-get install ruby , ja el vaig poder posar. I en tornar a posar ruby -v , ara ja veig que tinc la versió 3.1.2p20 .

Ara tocava connectar i configurar la Raspberry amb el perifèric escollit amb connexió UART. Primer de tot em vaig haver d'assegurar que el jumper del perifèric estigues en 00 perquè estigués en UART, si estigués en 10 seria I2C i en 01 seria SPI. La connexió dels pins que he utilitzat per

la connexió UART és la següent, tenint en compte que jo ja tenia connectat en el pin 4 i 6 el ventilador:



- GND → GND pin 9
- VCC → 5 V PWR pin 2
- SDA → UART TX pin 10
- SCL → UART RX pin 8

Per iniciar la comunicació UART vaig haver de tornar a obrir sudo raspi-config , i allà des de *Interface Options* anar a *serial port*, i et salten 2 pantalles, a la primera et surt el següent missatge: *Would you like a login shell to be accessible over serial?* , s'ha de dir que no, i a la segona amb el següent missatge: *Would you like the serial port hardware to be enabled?* , aquesta s'ha de dir que si. I ja podem sortir de la configuració de la raspberry. Per acabar, s'ha de reiniciar la Raspberry perquè siguin afectius els canvis de configuració realitzats.

Per poder comprovar que s'ha activat la comunicació per UART es pot veure de la següent forma: sudo nano /boot/firmware/config.txt , si esta habilitat ha de sortir enable_uart = 1 .

4. Biblioteques

Per poder instal·lar les gemmes de ruby, necessitava un gestor d'aquestes. Per instal·lar RubyGems vaig fer el següent: sudo apt install rubygems. Després vaig saber que podria haver instal·lat ruby i RubyGems tot junt utilitzant sudo apt install ruby-full.

La gemma que vaig trobar que em podria anar bé va ser ruby-nfc, i aquesta utilitza les llibreries libnfc i libfreefare. Seguint les indicacions del github les vaig instal·lar de la següent manera:

- sudo apt-get install libusb-1.0-0-dev
- git clone https://github.com/nfc-tools/libnfc.git
    1. cd libnfc
    2. autoreconf -vis
    3. ./configure , aquí em van saltar molts erros sobre libusb i no se'm generaven alguns fitxers, com vaig solucionar està en el següent apartat.
    4. make
    5. sudo make install
- git clone https://github.com/nfc-tools/libfreefare.git
    1. cd libfreefare
    2. autoreconf -vfi
    3. ./configure
    4. Make
    5. Sudo make install

Quan ja m'havia d'instal·lar les dues llibreries llavors ja em podia posar la gema, sudo gem install ruby-nfc. Per comprovar que funcionava puc utilitzar l'ordre nfc-list i et diu quins perifèrics de NFC tens connectats. Finalment, per comprovar que funcionar que el NFC i les llibreries funcionaven, amb l'ordre nfc-poll pots provar el lector amb qualsevol targeta amb NFC i et dona la seva informació, a partir d'això basarem el nostre codi.

4.1. Problemes Trobats

En fer el ./configure de la primera llibreria em saltaven diversos errors sobre que a la llibreria libusb, necessària per instal·lar libnfc, li faltaven les capçaleres, i no generava els fitxers de Makefile. Investigant vaig veure que em faltava instal·lar-me diversos paquets i llibreries:

Sudo apt-get install buid-essential pkg-config libudev-dev autoconf automake libssl-dev libtool libpcsclite-dev git

Aquest és el comando per instal·lar-ho tot de cop, però es podria haver fet per separat perfectament. Després de tenir això instal·lat i torna a provar la instal·lació de la llibreria libnfc, ja no vaig tenir cap problema i se'm van instal·lar perfectament les dues llibreries.

Amb les llibreries instal·lades vaig provar l'ordre nfc-list on m'hauria de sortir el perifèric PN532 però posava que no hi havia cap dispositiu de NFC connectat. Per arreglar-ho, obrint sudo nano /etc/nfc/libnfc.conf, he hagut d'afegir les següents línies:

- device.name = "PN532 over UART"
- device.connstring = "pn532_uart:/dev/serial0"

Ja que aquestes estaven com a exemples en format comentari i llavors no s'executaven. En tornar a cridar l'ordre de nfc-list ja em va sortir el meu dispositiu amb el missatge següent: *NFC device: PN532 over UART opened.*

5. Codi Imprès

El codi implementat tal que el PN532 llegeixi una targeta o clauer Mifare S50 o la targeta UPC, i imprimeixi per pantalla el uid en hexadecimal i en majúscules és el següent:

```
require 'ruby-nfc'
class Rfid

  def read_uid
        nfc = NFC::Reader.all
        nfc[0].poll(Mifare::Classic::Tag) do |tag|
                return uid_hex.upcase
        end
  end
end

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

Que per poder-lo editar des de la consola he utilitzat nano puzzle1.rb , i per executar-lo ruby puzle1.rb .