Sebastián Román

Puzle 1: MEMORIA

Fase Inicial: Setup de la RaspberryPi y la implementació de les seves llibrerires:

Una vegada configurat el raspberrypi OS:

1) L'objectiu ha sigut provar de connectar la RPI al ordinador laptop per tal de poder manipular-la.

La opció per la que he optat ha sigut mitjançant un cable de ethernet utilitzant un protocol ssh

Prèviament amb una pantalla i un ratolí externs els he connectat a la RPI i he posat des de el terminal els següents comandaments que he trobat a internet:

sudo raspi-config →Habilitar el protocol SSH

2) A continuació he intentat esbrinar la direcció IP :

1er cercant la IP de manera estàtica, amb els següents comandaments: Però no me'n sortit

sudo nano /etc/dhcpcd.conf

interface eth0

static ip_address=192.168.1.10/24

static routers=192.168.1.1

static domain_name_servers=192.168.1.1

2n He utilitzat el protocol DHCP: (és un protocol de xarxa que permet als nodes d'una xarxa IP obtenir els seus paràmetres de configuració automàticament. Es tracta d'un protocol de tipus client/servidor en què generalment un servidor posseeix una llista d'adreces IP dinàmiques i les va assignant als clients a mesura que aquestes van estant lliures, sabent en tot moment qui ha estat en possessió d'aquella IP, quant temps l'ha tinguda o a qui se li ha assignat després.)

Simplement amb el comandament :

hostname -I

Això hem dona facilitat per quan canvií de xarxa , ja sigui en casa o a la universitat. Evitant-me que la hagi de configurar manualment cada cop amb una configuració de IP estàtica.

3) Posteriorment he configurat la xarxa del meu ordinador des de

Panel de control > **Centro de redes y recursos compartidos** > **Cambiar configuración del adaptador**.

He seleccionat **Protocolo de Internet versión 4 (TCP/IPv4)** y he seleccionat **Propiedades**.

Dirección IP: 192.168.1.1
Máscara de subred: 255.255.255.0
Puerta de enlace:

4) Connexió SHH a la RPI:
   El protocol SSH utilitza una **arquitectura client-servidor** per establir connexions segures.
   Investigant he trobat que el programa PUTTY em simplificava el procés de connexió SSH, ja que és l'aplicació que utilitzes per connectar-te a un servidor remot a mode de **client** y el RPI és el servidor remot, al qual es vol accedir.
   Així que obro Putty , i poso en el àrea Host Name (or IP address), la direcció IP del meu RPI.
   I en aquest punt s'inicia un procés de autenticació , on introdueixo la clau SSH, que garanteix la seguretat de la comunicació.
   A partir d'aquí s'estableix la connexió via cable ethernet.
5) Instal·lar Ruby :

   sudo apt-get install ruby-full

6) Per trovar les llibreries que utilitza el meu perifèric simplement he anat a la pagina web de RubyGems y buscat el nom del meu perifèric RFID_RC522 i les he instal·lat.
   L **'RC522** sol treballar amb el protocol SPI, per la qual cosa he necessitat una gem que permeti la comunicació amb aquest protocol.

   gem install mfrc522

7) Per utilitzar el lector RC522 he hagut de instal·lar unes dependències addicionals (SPI y GPIO), de tal manera que pugui acceder als pins , per tal de que es doni una comunicació SPI (Serial Peripheal Interface);

   gem install pi_piper

8) Habilitem el SPI al RPI amb els seguents comandaments:

   sudo raspi-config

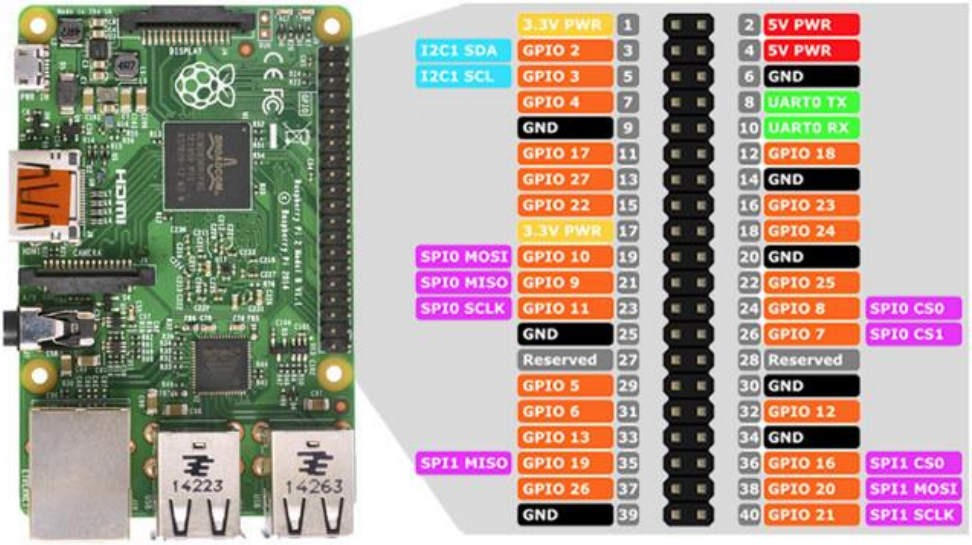
   Interfacing Options > **SPI** L'habilito

   sudo reboot

9) Connectem en SPI per mitjà de cables amb el perifèric:

Pin del RC522	Función	Pin GPIO de la Raspberry Pi	Número de pin físico
VCC	Alimentación	3.3V	1
GND	Tierra	GND	6
MISO	Salida SPI	GPIO 9 (MISO)	21
MOSI	Entrada SPI	GPIO 10 (MOSI)	19
SCK	Reloj SPI	GPIO 11 (SCLK)	23
SS (SDA)	Selección de esclavo	GPIO 8 (CE0)	24
RST	Reinicio del módulo	GPIO 25	22

He utilitzat la següent guia de com connectar els cables amb els pins adients.



PUZZLE1 Guia del codi:

```
require 'mfrc522' # Requereix la llibreria per interactuar amb el lector RFID-RC522
class Rfid
  def read_uid
    puts "Por favor, acerque su tarjeta al lector."
# Es mostra un missatge per pantalla, indicant a l'usuari que acosti la seva targeta al lector. Aquesta és una simple instrucció per facilitar la interacció amb el sistema.

    begin
```

```
        r = MFRC522.new #S'inicia una nova instància del lector RFID utilitzant MFRC522.new. Això permet establir la connexió amb el lector de targetes per començar la lectura.

        r.picc_request(MFRC522::PICC_REQA)
#Es fa una petició al lector RFID perquè busqui una targeta a prop. PICC_REQA és el codi que indica que el lector ha de començar a buscar una targeta activa (ja que les targetes RFID responen a aquestes sol·licituds).

        uid_dec, _ = r.picc_select

#Si es troba una targeta, picc_select llegeix el seu UID (Identificador Únic de la Targeta) i l'emmagatzema en la variable uid_dec. Aquesta variable és un array que conté els bytes del UID en decimal

    rescue CommunicationError => e
        puts "Error de comunicación: #{e.message}"
        retry # Reintentamos en caso de error
    end
#Si hi ha un error de comunicació amb el lector (per exemple, si la targeta no es detecta correctament), es captura l'excepció CommunicationError. Es mostra un missatge d'error i es reintenta la lectura automàticament.

    uid = uid_dec.map { |byte| byte.to_s(16).rjust(2, '0') }.join('').upcase
El mètode **map** en Ruby se aplica a arrays, i itera sobre cada element del array, aplicant el bloc de codi  proporcionat, i després retorna un nou array amb els resultats de l'aplicació d'aquest bloc.

# byte.to_s(16):Converteix el valor decimal del byte en la seva representació hexadecimal. El paràmetre 16 especifica que la conversió es fa a base 16 (hexadecimal).

# rjust(2, '0'): Assegura que cada número hexadecimal tingui exactament dos dígits. Si el número té menys de dos dígits, s'afegeixen zeros a l'esquerra ('0'). Això és important perquè els valors hexadecimal d'un byte han de tenir sempre dos dígits (exemple: 5 es converteix en 05).

    return uid # Retornem el UID
  end
end

if __FILE__ == $0
# Condició que es fa servir a Ruby per controlar sun script está sent executat directament o is està sent important una llibreria d'un alter programa.
```

- `__FILE__ == $0` és **cert** només quan el fitxer que estàs executant és el mateix fitxer que conté el codi (és a dir, que s'està executant directament i no com a part d'un altre codi). Això vol dir que el bloc de codi dins d'aquest `if` només s'executarà si el programa s'està llançant de forma directa, no quan s'importi en un altre fitxer.
- **`rf = Rfid.new`**
- # Aquesta línia es crea un objecte anomenat `rf` que tindrà accés a tots els mètodes definits a la classe `Rfid`, incloent-hi el mètode **`read_uid`**
- `uid = rf.read_uid`
- #Aquest mètode s'encarrega de llegir el UID d'una targeta RFID utilitzant el lector RC522.
- **`puts "UID: #{uid}"`**
- #Aquesta línia utilitza `puts` per imprimir el UID que s'ha llegit de la targeta. El **`#{uid}`** és una manera en Ruby d'inserir el valor de la variable `uid` dins del string.
-
- **`End`**
- `#Aquesta part del codi és important perquè gestiona com i quan s'executa el programa.`