PBE

PUZZLE 1

1. Conexiones Raspberry - RC522:

Pines Específicos en la Raspberry Pi

Función	Pin GPIO	Número de Pin	Descripción
SDA (CS)	GPIO 24	Pin 18	Selección del dispositivo (CS)
SCK	GPIO 23	Pin 16	Reloj de la comunicación SPI
MOSI	GPIO 19	Pin 24	Datos de salida del maestro
MISO	GPIO 21	Pin 22	Datos de entrada del esclavo
GND	GND	Pin 6	Tierra
RST	GPIO 22	Pin 15	Reinicio del módulo
VCC	3.3V	Pin 1	Alimentación (3.3V)

La propia Raspberry tiene asignados pines específicos para su uso. En la imagen podemos observar los que usaremos y a que pin están asignados.

- **GPIO 24 (SDA):** Se utiliza para la selección del chip (CS). Cuando se baja, se habilita el módulo RC522 para recibir datos. Este pin se elige para que coincida con el estándar SPI.

- **GPIO 23 (SCK):** Este pin se usa como la línea de reloj en SPI (Interfaz de Periféricos en Serie), un protocolo de comunicación de datos. En la Raspberry Pi, este es el pin predefinido para SCK en la interfaz SPI, lo que permite que los datos se sincronicen correctamente.

- **GPIO 19 (MOSI):** Aquí se envían los datos desde la Raspberry Pi al RC522. Este pin también es predefinido para la interfaz SPI, lo que facilita la conexión.

- **GPIO 21 (MISO):** Este pin es utilizado por el RC522 para enviar datos de vuelta a la Raspberry Pi. Al igual que MOSI, está predefinido para SPI en la Raspberry Pi.

- **GPIO 22 (RST):** Este pin se utiliza para reiniciar el módulo RC522. Se conecta a un GPIO que puede ser controlado por software para permitir reinicios programáticos del módulo.

- **GND y VCC:** Estos pines son necesarios para proporcionar energía y una referencia de tierra al módulo.

2. Conexiones Raspberry-PC

Al principio hubo complicaciones con este tema, al no poder conectar la Raspberry a un monitor, teclado y ratón independientes para operar directamente en ella.

La primera opción fue optar por **conexión vía SSH** y aunque también surgió algún que otro problema, finalmente fue posible conectarla.

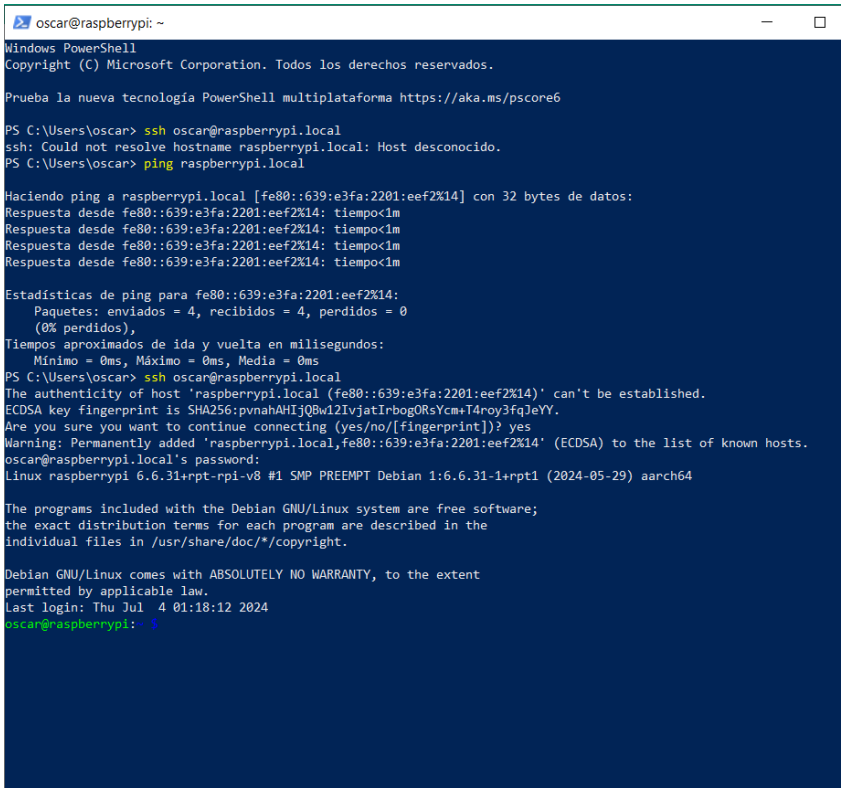
Al no disponer de la dirección IP de la Raspberry, decidí usar el hostname de la Raspberry: **raspberry.local,** que también nos permití la conexión.

Probé primero en mi máquina virtual de Ubuntu, sin embargo, al parecer era imposible activar la conexión. Nunca encontraba al host, aun estando las configuraciones correctas. Probando en otra máquina virtual, con la misma configuración, de algún compañero sí se conseguía exitosamente la conexión.

Ante este problema, decidí acceder directamente desde Windows usando PowerShell y finalmente pude realizar la conexión.

Nota: En cualquiera de estas opciones, la Raspberry tiene que estar alimentada y conectada al dispositivo a través de un cable Ethernet para darle internet.

Aquí observamos la ejecución de la conexión:



Explicación de los comandos

- La línea de comando **ssh oscar@raspberrypi.local** se utiliza para establecer una conexión SSH con un dispositivo que tiene la dirección raspberrypi.local, donde oscar es el nombre de usuario al que nos conectamos

  El hostname raspberrypi.local es una forma común de referirse a la Raspberry Pi en la red local cuando el dispositivo tiene habilitado el servicio mDNS (Multicast DNS). Esto permite que los dispositivos de la red se identifiquen por nombre en lugar de tener que recordar direcciones IP.

  Si la Raspberry Pi está en la misma red local y está configurada para aceptar conexiones SSH, deberías poder acceder a su terminal de línea de comandos.

  Finalmente, pide la contraseña asociada con el usuario oscar en la Raspberry Pi. Una vez ingresada correctamente, tenemos acceso a la terminal de la Raspberry Pi, lo que te permitirá ejecutar comandos, administrar archivos y realizar configuraciones.

  Nota: Como podemos observar al principio no encuentra el hostname. Es normal porque, a veces falla, por lo que ejecutamos el siguiente comando como vemos en la imagen.

- El comando ping raspberrypi.local verificar la conectividad de red y asegura que la Raspberry Pi esté accesible desde tu computadora. Si recibimos respuesta, significa que la Raspberry Pi está encendida y conectada a la red, lo que te permitirá continuar con otras interacciones, como establecer una conexión SSH

  Comprobamos que sí que está conectada, repetimos el primer comando y observamos que nos hemos conectado correctamente

3. Instalación Ruby y bibliotecas probadas RFID RC522:

Primero, actualizamos nuestra RaspberryPi: **sudo apt-get update**

Para poder comunicar la Raspberry con el RFID-RC522 necesitamos activar el puerto SPI usando: **sudo rasp-config**

Iremos a “Interfacing Options” y seleccionaremos SPI para habilitar la interfaz SPI. Hay que reiniciar la RaspberryPi: **sudo reboot**

Instalamos Ruby y algunas librerías que te permitirán interactuar con el RC522:

- Instalar Ruby: **sudo apt-get install ruby-full**

Para trabajar con un lector RFID MFRC522 en una Raspberry Pi utilizando Ruby, necesitamos instalar varias librerías:

- **rpi_gpio:** Esta librería permite controlar los pines GPIO de la Raspberry Pi desde Ruby. Necesaria para gestionar la comunicación con el lector RFID.

- **colorize**: Se utiliza para agregar color al texto en la terminal, lo que mejora la legibilidad de las salidas del script.

- **mfrc522:** Esta librería proporciona una interfaz para interactuar con el lector RFID MFRC522. Es fundamental para leer el UID de las tarjetas RFID.

Motivo de estas librerías

No he tenido que investigar mucho puesto que son las que mas recomiendan para trabajar con este periférico. Las dos tres librerías son gemas de Ruby, que son paquetes de software que se pueden instalar fácilmente desde el repositorio oficial de gemas de Ruby, conocido como RubyGems: **sudo gem install pi_piper, sudo gem install** y **sudo gem install colorize**

```
oscar@raspberrypi:~ $ sudo gem install rpi_gpio
Fetching rpi_gpio-0.5.0.gem
Fetching epoll-0.3.0.gem
Building native extensions. This could take a while...
Successfully installed epoll-0.3.0
Building native extensions. This could take a while...
Successfully installed rpi_gpio-0.5.0
Parsing documentation for epoll-0.3.0
Installing ri documentation for epoll-0.3.0
Parsing documentation for rpi_gpio-0.5.0
Installing ri documentation for rpi_gpio-0.5.0
Done installing documentation for epoll, rpi_gpio after 0 seconds
2 gems installed
```

```
oscar@raspberrypi:~ $ sudo gem install colorize
Fetching colorize-1.1.0.gem
Successfully installed colorize-1.1.0
Parsing documentation for colorize-1.1.0
Installing ri documentation for colorize-1.1.0
Done installing documentation for colorize after 0 seconds
1 gem installed
```

```
oscar@raspberrypi:~ $ sudo gem install mfrc522
Fetching mfrc522-3.0.0.gem
Fetching fubuki-1.0.1.gem
Building native extensions. This could take a while...
Successfully installed fubuki-1.0.1
Successfully installed mfrc522-3.0.0
Parsing documentation for fubuki-1.0.1
Installing ri documentation for fubuki-1.0.1
Parsing documentation for mfrc522-3.0.0
Installing ri documentation for mfrc522-3.0.0
Done installing documentation for fubuki, mfrc522 after 1 seconds
2 gems installed
```

4. Script Ruby para implementación Puzzle1:

Para ejecutar el código, clone el repositorio de GitHub con el código. Para ello tuve que instalar Github en la Raspberry con el comando: **sudo apt install git -y**

Clonamos el repositorio: git clone https://github.com/oscarparada08/PBE.git

Ahora tmb lo tenemos en nuestra rasberry. Navegamos en el con el comando cd PBE

Y ejecutamos el código: ruby PUZZLE1.rb

```ruby
# Este script utiliza un lector RFID MFRC522 para leer el UID de tarjetas RFID
# Requiere una Raspberry Pi y módulos adicionales para GPIO y control de colores en terminal.

require 'rpi_gpio'  # Biblioteca para controlar los pines GPIO en la Raspberry Pi
require 'colorize'  # Biblioteca para agregar color al texto en la terminal
require 'mfrc522'   # Biblioteca para interactuar con el lector RFID MFRC522

# Definimos una clase que se encargará de manejar el lector RFID
class RfidRc522
  # Método que inicializa el lector, lee el UID de la tarjeta y lo devuelve en formato hexadecimal
  def scan_uid
    # Creamos una nueva instancia del lector RFID
    reader = MFRC522::Reader.new

    # Leemos el UID de la tarjeta RFID
    uid = reader.read_uid

    # Convertimos el UID a formato hexadecimal en mayúsculas
    uid_hex = uid.map { |byte| byte.to_s(16).upcase }.join

    # Retornamos el UID en formato hexadecimal
    return uid_hex
  end
end

# Método que limpia la pantalla del terminal
def clear_screen
  system('clear')
end

# Variable para controlar el ciclo de escaneo
opc = ""

# Bucle que sigue ejecutándose hasta que el usuario elija no escanear más
```

```ruby
while opc != "n"
  # Limpiamos la pantalla antes de cada escaneo
  clear_screen

  # Mostramos un mensaje con instrucciones para el usuario, con colores
  puts "\t" + "<<<<<<<<<<<".red
  puts "\t" + "    SCAN   ".yellow
  puts "\t" + "    YOUR   ".yellow
  puts "\t" + "    PASS   ".yellow
  puts "\t" + "<<<<<<<<<<<".red

  begin
    # Inicializamos el objeto para manejar el lector RFID
    rf = RfidRc522.new

    # Escaneamos y obtenemos el UID de la tarjeta
    uid = rf.scan_uid

    # Mostramos el UID obtenido en la terminal
    puts "\t YOUR UID IS:"
    puts "\t" + ">>>>>>>>>>".green
    puts "\t" + uid.strip.sub(/^0x/i, "").green  # Eliminamos el prefijo '0x' y mostramos en verde
    puts "\t" + ">>>>>>>>>>".green
  ensure
    # Preguntamos al usuario si quiere escanear otra tarjeta
    opc = "n"  # Inicializamos la opción como 'n' (para evitar que se quede en ciclo infinito)
    print "\t SCAN AGAIN? (y/n): "

    # Leemos la entrada del usuario y la convertimos a minúsculas
    opc = gets.chomp.downcase

    # Limpiamos los pines GPIO después de cada escaneo

     RPi::GPIO.clean_up
   end
end
```

Este script en Ruby permite usar un lector **RFID MFRC522** conectado a una RaspberryPi para leer el UID (identificador único) de las tarjetas RFID y mostrarlo en la terminal de manera amigable. Se utilizan tres bibliotecas clave: **rpi_gpio**, **colorize** y **mfrc522**. A continuación, se describe cómo funciona el código y los comandos utilizados.

El script define una clase llamada **RfidRc522** que encapsula la lógica para manejar el lector RFID. Dentro de esta clase, el método **scan_uid**:

- Inicializa el lector RFID creando una instancia de **MFRC522::Reader**.
- Lee el UID de la tarjeta a través del método **read_uid**.
- Convierte este UID (que está en formato binario) a formato **hexadecimal** y en **mayúsculas** para una mejor visualización, y lo retorna.

El método para limpiar la pantalla de la terminal, **clear_screen**, utiliza el comando system('clear'), que borra todo el contenido de la terminal para que cada escaneo comience sudocon un entorno limpio.

El programa utiliza un bucle principal que sigue ejecutándose hasta que el usuario decide salir. Cada ciclo del bucle sigue este proceso:

- Limpia la pantalla.
- Muestra un mensaje en la terminal pidiendo al usuario que acerque su tarjeta al lector, con las palabras "SCAN YOUR PASS" en color amarillo.
- Llama al método **scan_uid** para leer el UID de la tarjeta RFID.
- Muestra el UID obtenido en la terminal en color verde, eliminando cualquier prefijo hexadecimal (0x).

- Pregunta al usuario si quiere escanear otra tarjeta o salir. Si el usuario responde con "n", el programa termina.

Entre cada lectura, el script llama a **RPi::GPIO.clean_up** para liberar los pines GPIO y prepararlos para el próximo uso.

Problemas encontrados

Aun conectando la Raspberry por SSH, no recibía conexión a internet. Este problema fue difícil de solucionar, puesto que a priori, se estaban realizando los comandos correctos. Probé con otros tipos de conexión y persistía el error.

Al final el error se encontraba en el propio portátil. La configuración de serie no permitía compartir datos de conexión y tenia inhabilitado el uso compartido en el puerto Ethernet. Una vez visto el problema fue fácil ponerle solución

Al principio quise usar la librería rpi_gpio.so para controlar los pines pero no pude ya que está compilado para una arquitectura de 32 bits (ELFCLASS32), mientras que el sistema es de 64 bits. Así que pase a usar la librería **pi_piper**