Haoyan Chen

MEMORIA PUZZLE 1

1-CONEXIONES A INTERNET

Para conectarnos a internet lo podemos hacer de varias maneras, la primera sería mediante un cable ethernet entre la raspberry y el ordenador.Después de saber nuestro IP del ordenador ,luego tendríamos que descargar el putty y vnc viewer e introducir nuestra ip. Pero yo no lo hice de esta manera porque encontré un método más rápido de conectarlo. Mediante un cable HDMI primero conecto la raspberry a un monitor y configurar la raspberry a los datos moviles de mi móvil, en interfaces habilitó VNC y i2C(esta parte tmb se podria hacer mediante comandos en la terminal). Ahora cada vez que abrimos la raspberry y tenemos los datos móviles activados se conectara automáticamente.
Ahora solo nos queda descargar VNC viewer para visualizar nuestra raspberry en nuestro ordenador, solo tenemos que introducir la misma ip de nuestros datos compartidos del móvil, de esta manera podremos visualizar el escritorio de la raspberry.

2-CONFIGURACIONES REALIZADAS EN LA RPI

Antes de configurar la raspberry pi tenemos que conectar mediante cables hembra-hembra con nuestro ITEAD PN532, mediante I2C. En un nuestra PN532 tiene un interruptor donde tenemos que tener SET0 en 'H' y SET1 en 'L'
En la siguiente página encontraremos todos los datos que necesitamos

https://wiki.iteadstudio.com/ITEAD_PN532_NFC_MODULE

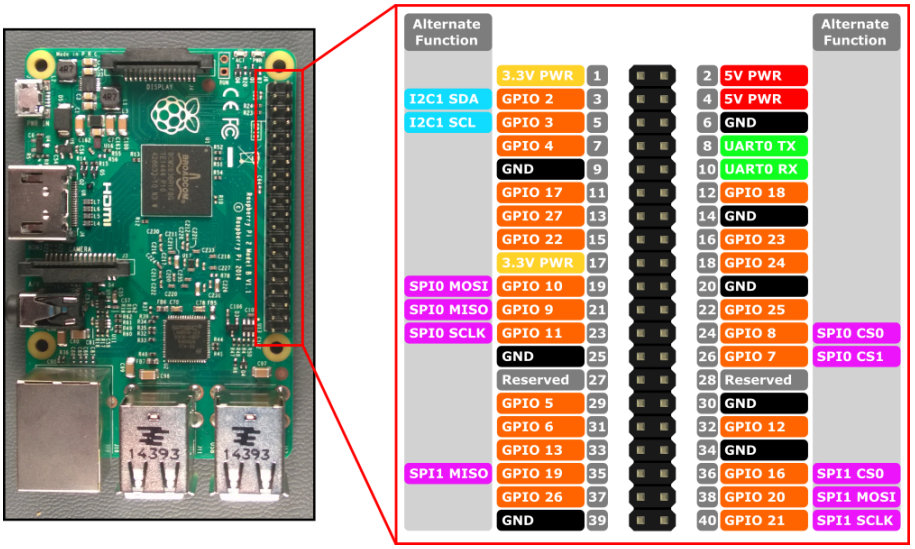
	SET0	SET1
UART	L	L
SPI	L	H
IIC	H	L

IC	NXP PN532
Operating Voltage	3.3V
Power Supply Voltage	3.3~5.5V
Max Supply Current	150mA
Working Current(Standby Mode)	100mA
Working Current(Write Mode)	120mA
Working Current(Read Mode)	120mA
Indicator	PWR
Interface	SPI Interface, Std Raspberry Pi 20pins Interface

Ahora tenemos que conectar el el módulo ITEAD PN532 con la raspberry

- VCC del PN532 al pin 2 de la raspberry (5V)
- GND del PN532 al pin 6 (GND)
- SDA del PN532 al pin 3 (GPIO2, SDA)
- SCL del PN 532 al pin 5 (GPIO4, SCL)

Aquí adjunto una imagen de los pins de la raspberry para saber donde van



Ahora podemos empezar con la configuración de la raspberry PI:

- sudo raspi-config

Primero entramos en interfacing options y de todas las option tendremos que habilitar el I2C a continuación reiniciamos la raspberry → “sudo reboot”
Ahora tendremos que descargar los tools de la I2C además de verificar si está bien configurado y conectado.

- sudo apt-get install -y i2c-tools
- sudo i2cdetect -y 1

```
haoyan@haorsb:~/Pbe $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- 24 -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Este comando escanea los dispositivos conectados al bus I2C en la Raspberry Pi y muestra sus direcciones.

Explicación de la tabla:

- La matriz de números representa las direcciones I2C en formato hexadecimal.
- Un guión (`--`) indica que no hay ningún dispositivo en esa dirección.
- El número `24` en la fila `20:` indica que hay un dispositivo I2C en la dirección **0x24**

3-INSTALACIONES DE LIBRERÍAS

Tenemos que instalar dos librerías libnfc y libfreefare, ambas son bibliotecas de código abierto diseñadas para interactuar con lectores NFC y tarjetas RFID.
Las páginas donde descargamos la librería son las siguientes.

https://github.com/nfc-tools/libnfc
https://github.com/nfc-tools/libfreefare

Descargar libnfc

```
Unset
# tar xjvf libnfc-1.7.1.tar.bz2
# cd libnfc-1.7.1/
# ./configure
# make && make install
```

```
Unset
sudo cp ./contrib/linux/blacklist-libnfc.conf /etc/modprobe.d/
sudo cp ./contrib/udev/42-pn53x.rules /etc/udev/rules.d/
```

Descargar libfreefire :

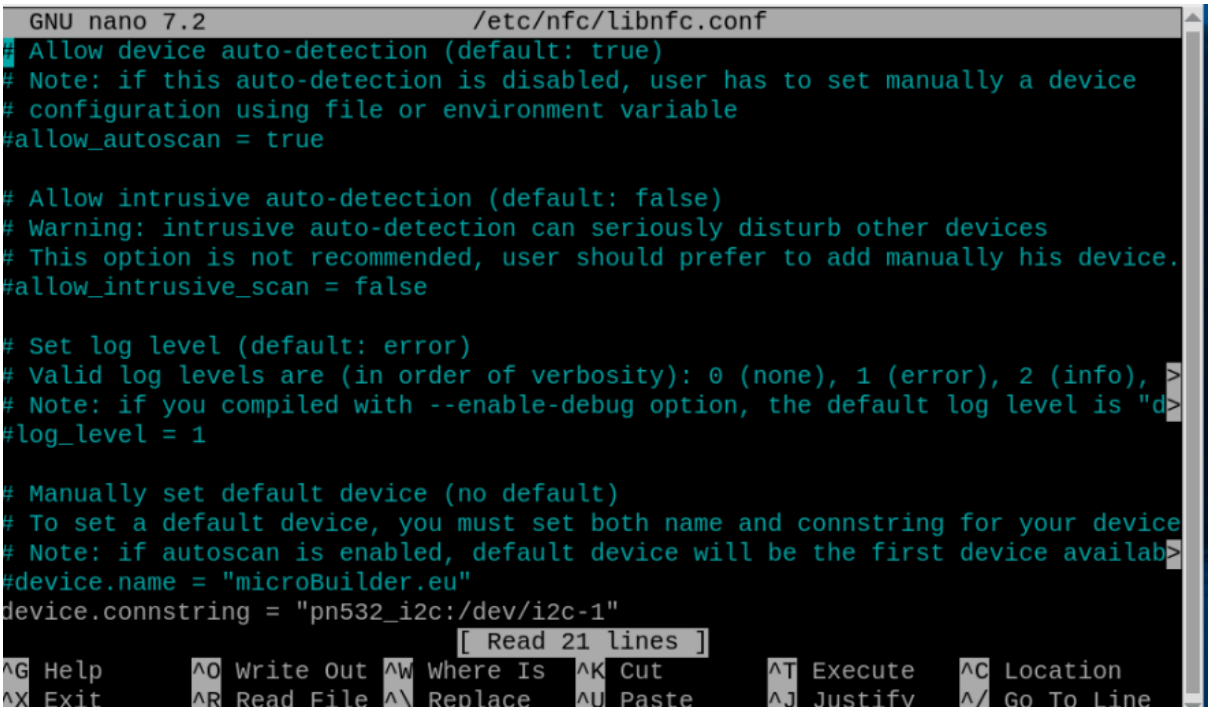
```
Unset
# git clone https://github.com/nfc-tools/libfreefare.git
# cd libfreefare
# autoreconf -vis
# ./configure && make && make install
```

Ahora tendremos que configurar libnfc :

- sudo mkdir -p /etc/nfc
- sudo nano /etc/nfc/libnfc.conf

Cambiamos la última line del fichero por :

- device.connstring = "pn532_i2c:/dev/i2c-1"



Ahora instalaremos el idioma ruby, mediante la siguiente línea

```
Unset
gem install ruby-nfc
```

si no es posible lo haremos con la siguiente línea :

- sudo apt-get install autoconf automake
- sudo apt-get install ruby
- sudo gem install ruby-nfc

- gem build ruby-nfc.gemspec
- gem install ruby-nfc-*.gem

4-PROBLEMAS ENCONTRADOS

Al inicio tuve problemas a la hora detectar los datos de la NFC mediante I2C, porque no tenía los puertos o SET0 i SET1 configurados correctamente en H i L, después de ponerlos bien ya me iba.

```
haoyan@haorsb:~/Pbe $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

El problema principal que tuve fue a la hora de descargar las librerías, no sabía qué librerías tenía que descargar. Al inicio descargaba cualquier cosa que me parecía que estaba bien, pero al final eso hizo que mis librerías se solapen.
Así que al final decidí volver a eliminar todas las librerías y volver a empezar de 0, que fue lo correcto. Y en la segunda vez descargue las librerías correctas y necessarias.

Eliminar el paquete de las herramientas NFC
sudo apt-get remove --purge libnfc-bin
Eliminar las bibliotecas de desarrollo
sudo apt-get remove --purge libnfc-dev
Eliminar dependencias no necesarias
sudo apt-get autoremove --purge
Limpiar paquetes descargados
sudo apt-get clean
Verificar si `nfc-list` sigue disponible
nfc-list

Después de estas líneas de código volvemos a descargar las librerías y todo fue bien. Haciendo comprobaciones con nfc-poll, vemos que recibe el código(UID) de la NFC en hexadecimal perfectamente.

```
haoyan@haorsb:~/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)
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
       UID (NFCID1): 9a  6a  c9  01
      SAK (SEL_RES): 08
waiting for card removing...nfc_initiator_target_is_present: Target Released
done.
```

5- CÓDIGO

Hemos utilizado el editor de texto nano, mediante la siguiente línea creamos un texto en ruby en linux :

- nano codigo.rb

Y podemos empezar a codificar, para comprobar si esta bien y ejecutar el código haremos lo siguiente :

- ruby codigo.rb

La siguiente imagen es el código que implementamos para ejecutar una función parecida a nfc-poll, que imprima la UID de la NFC en Hexadecimal.

```ruby
require 'ruby-nfc'

class Rfid
  def read_uid
    readers = NFC::Reader.all
    return nil if readers.empty?

    readers[0].poll(IsoDep::Tag, Mifare::Classic::Tag, Mifare::Ultralight::Tag) do |tag|
      begin
        case tag
        when Mifare::Classic::Tag, Mifare::Ultralight::Tag
          return tag.uid_hex.upcase
        when IsoDep::Tag
          return tag.uid.unpack1('H*').upcase
        end
      rescue StandardError => e
        puts "Error al leer la tarjeta: #{e.message}"
        return nil
      end
    end
    nil
  end
end

if __FILE__ == $0
  rf = Rfid.new
  puts "Esperando para leer la tarjeta NFC..."
  uid = rf.read_uid

  if uid
    puts "UID de la tarjeta: #{uid}"
  else
    puts "No se pudo leer la tarjeta."
  end
end
```

La siguiente imagen sería la ejecucion de codigo.rb, podemos ver que nos imprime correctamente el código de la NFC

```
haoyan@haorsb:~/Pbe $ ruby codigo.rb
Esperando para leer la tarjeta NFC...
UID de la tarjeta: 9A6AC901
```

Explicación del código :

```
#Añadimos la librería que utilizaremos en el código
Require ‘ruby-nfc’

#Una nueva clase Rfid
class Rfid

#Declaramos un método que se encarga de leer el UID de las tarjetas NFC
def read_uid

#Obtener los lectores NFC disponibles y devuelve nil si no hay lectores disponibles
readers = NFC::Reader.all
return nil if readers.empty?

#iniciar la detección de tarjetas de 3 diferentes tipos de NFC
readers[0].poll(IsoDep::Tag, Mifare::Classic::Tag, Mifare::Ultralight::Tag) do |tag|

#Determinar el tipo de tarjeta y extraer su UID
begin
  case tag
  when Mifare::Classic::Tag, Mifare::Ultralight::Tag
    return tag.uid_hex.upcase
  when IsoDep::Tag
    return tag.uid.unpack1('H*').upcase
  end

#Detectar si hay algún error
rescue StandardError => e
  puts "Error al leer la tarjeta: #{e.message}"
  return nil
end

#Ejecutar el código si es el archivo principal
if __FILE__ == $0

#crear un objeto Rfid
rf = Rfid.new

#Método para obtener el UID
uid = rf.read_uid

#Mostrar el resultado
if uid
  puts "UID de la tarjeta: #{uid}"
else
  puts "No se pudo leer la tarjeta."
end
```