Informe: Uso de Ruby en Raspberry Pi para Interacción con Dispositivos

Introducción

En este informe se describe cómo usar el lenguaje de programación Ruby en una Raspberry Pi para interactuar con dispositivos USB, como lectores de tarjetas, y cómo leer y procesar datos de estos dispositivos.

La Raspberry Pi es una plataforma ideal para proyectos de bajo costo y fácil acceso, y Ruby es un lenguaje muy accesible que permite escribir código eficiente y claro para interactuar con hardware.

Objetivo

El objetivo de este proyecto es mostrar cómo usar Ruby para leer información de un lector de tarjetas Mifare USB M301 conectado a una Raspberry Pi, cómo convertir los datos obtenidos en un formato hexadecimal y cómo gestionar la entrada y salida de información desde el terminal.

Instalación de Ruby en Raspberry Pi

Necesitamos el cable Ethernet y un adaptador.

Debemos habilitar el acceso Ethernet desde la red inalámbrica, lo cual se puede hacer desde los ajustes. Esta es la parte más complicada.

En mi caso, he usado la IP**: 192.168.137.227**.

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

Interfaz: 192.168.137.1 --- 0x8
  Dirección de Internet          Dirección física      Tipo
  192.168.137.227       2c-cf-67-77-91-65     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
```

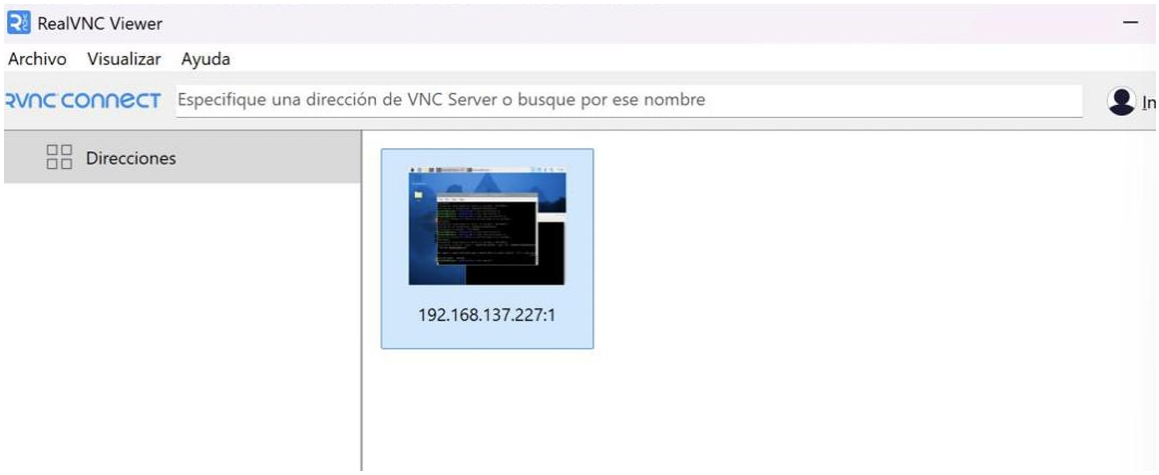
A continuación, debéis descargar PuTTY y RealVNC Viewer. En PuTTY, tendréis que introducir vuestro usuario y contraseña para acceder a la Raspberry Pi.

Para iniciar el servidor VNC, escribid el siguiente comando en la terminal:

---

vncserver-virtual

Si todo está configurado correctamente, en RealVNC Viewer debería aparecer la interfaz gráfica de la Raspberry Pi:



Instalación de Ruby en Raspberry Pi:

Para comenzar, es necesario tener Ruby instalado en la Raspberry Pi. Ruby generalmente está preinstalado en la mayoría de las distribuciones de Linux, pero si no lo tienes, puedes instalarlo con el siguiente comando:

---

sudo apt-get update

sudo apt-get install ruby-full

Interacción con el Lector de Tarjetas Mifare

El lector de tarjetas Mifare USB M301 se comunica con la Raspberry Pi a través del puerto USB. Cuando una tarjeta se acerca al lector, se emite un conjunto de datos que representa el UID (Identificador único) de la tarjeta.

A continuación, se muestra un ejemplo de cómo leer los datos de la tarjeta y convertirlos a formato hexadecimal en Ruby.

Código en Ruby para leer el UID de la tarjeta y convertirlo a hexadecimal

```ruby
puts "Acerca una tarjeta al lector (o escribe algo en el teclado)..."

input = gets.chomp  # Lee una línea de entrada

# Convierte la entrada (que se asume como número decimal) a hexadecimal
hex_input = input.to_i.to_s(16).upcase.rjust(8, '0').scan(/.{2}/).reverse.join

# Imprime la información leída y la representación hexadecimal
puts "Información leída desde el lector (o teclado): #{input}"
puts "Información en hexadecimal: #{hex_input}"

```

Explicación del Código

- **gets**: Lee una línea de entrada desde el teclado o desde un lector de tarjetas que emule entrada de teclado.
- **input.to_i**: Convierte la entrada de texto en un número entero.
- **to_s(16)**: Convierte el número decimal a su representación en hexadecimal.
- **rjust(8, '0')**: Asegura que el valor hexadecimal tenga 8 caracteres, rellenando con ceros a la izquierda si es necesario.
- **scan(/.{2}/).reverse.join**: Reorganiza el orden de los bytes para que se muestre correctamente en formato hexadecimal.

Ejemplo de uso:

Cuando se ejecuta el código, el sistema pedirá que acerques una tarjeta al lector o que ingreses una cadena manualmente. Si introduces un valor como `0067385523`, el programa imprimirá lo siguiente:

Acerca una tarjeta al lector (o escribe algo en el teclado)...

0067385523

Información leída desde el lector (o teclado): 0067385523

Información en hexadecimal: B3380404

Little Endian y Big Endian

En nuestro caso, con nuestra tarjeta, tuvimos que convertir el código porque estaba en **formato Little Endian**, lo que significa que los bytes estaban almacenados en orden inverso al que esperábamos.

¿Cuál es la diferencia entre Little Endian y Big Endian?

- **Big Endian**: Almacena el byte más significativo (MSB - Most Significant Byte) primero. Es decir, el número se guarda en el mismo orden en que lo leemos.
    - Ejemplo:
        - Número en hexadecimal: 0x12345678
        - Almacenado en memoria: 12 34 56 78
- **Little Endian**: Almacena el byte menos significativo (LSB - Least Significant Byte) primero, es decir, en orden inverso.
    - Ejemplo:
        - Número en hexadecimal: 0x12345678
        - Almacenado en memoria: 78 56 34 12

La parte avanzada:

He creado un archivo datos.txt para almacenar el UID de cada tarjeta junto con el nombre correspondiente.

Al ejecutar el código, este compara el UID de la tarjeta con el contenido del archivo. Si encuentra una coincidencia, muestra el nombre en la pantalla.

```
1    0067385523, Anna
2    3255837600, BOB
3    3503970084, Dilyara
4    2319798526, Alice
```

```ruby
# Función para leer los datos y almacenarlos en un hash
def leer_identificadores_y_nombres(archivo)
  datos = {}
  File.open(archivo, 'r') do |file|
    file.each_line do |line|
      next if line.strip.empty?  # Ignorar líneas vacías

      # Dividimos la línea por la coma y comprobamos si se generaron exactamente 2 elementos
      partes = line.strip.split(',')
      if partes.length == 2
        identificador, nombre = partes
        datos[identificador.strip] = nombre.strip  # Almacenamos el identificador y el nombre
      else
        puts "Formato incorrecto en la línea: '#{line.strip}'"
      end
    end
  end
  datos
end
```

```ruby
# Función para comparar el UID con los datos
def comparar_identificadores(input, datos)
  # Compara el identificador directamente

  if datos.key?(input)
    hex_input = input.to_i.to_s(16).upcase.rjust(8, '0').scan(/.{2}/).reverse.join
    hex_input1 = input.to_i.to_s(16).upcase.rjust(8, '0');

    puts "Información en hexadecimal(Little Endian): #{hex_input} pertenece a #{datos[input]}"
    puts "Información en hexadecimal sin girar(Big Endian): #{hex_input1} pertenece a #{datos[input]}"
  else
    puts "No se encontró ninguna coincidencia para el identificador #{input}."
  end
end
```

```ruby
# Main

puts "Acerca una tarjeta al lector (o escribe algo en el teclado)..."

# Leer los identificadores y nombres del archivo
archivo = 'datos.txt'  # Nombre del archivo donde están los identificadores y nombres
datos = leer_identificadores_y_nombres(archivo)

input = gets.strip

# Comparar la entrada con los identificadores en el archivo
comparar_identificadores(input, datos)
```

Conclusión

El uso de Ruby en Raspberry Pi proporciona una manera sencilla y poderosa de interactuar con dispositivos de hardware, como lectores de tarjetas USB. En este informe, hemos aprendido cómo leer datos de un lector de tarjetas y convertirlos en un formato hexadecimal adecuado para procesarlos de manera eficiente.

Dificultades:

La parte más complicada para mí fue configurar la Raspberry Pi y conectarla a eduroam. Tuve que ir al Panel de Control y acceder a la red inalámbrica, luego seleccionar Ethernet6. Por probar, intenté cambiar la IP de la conexión Ethernet, pero creo que sin querer cambié la IP de mi portátil y ya no pude acceder a Internet.

Además, tuve dificultades con el código porque estaba usando un lenguaje nuevo y no sabía que la tarjeta nos da la información en formato decimal. También me resultaron complicados los métodos que utilizamos para convertir de decimal a hexadecimal.