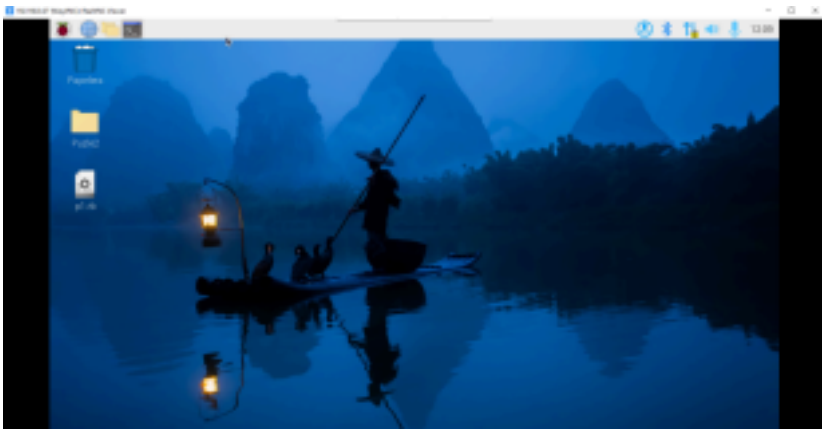
Segundo Puzle en Ruby

Lucas Mira

Configuración del entorno gráfico Raspberry Pi4

Como este puzle requería entorno gráfico primero había que configurarlo correctamente, lo primero que hice fue descargarme el programa RealVNC viewer, con el cual puedes acceder a la RB Pi4 mediante la ip que se le asigna cuando esta se conecta a internet, siempre que el dispositivo donde tengas instalado el programa esté en la misma LAN que la RB Pi4, la ip ya la sabía de cuando hice el puzle 1.



Creación y ejecución del Puzzle 2

Lo primero que hice fue instalar la biblioteca GTK para crear una interfaz gráfica que interactúa con el lector RFID para leer el UID de una tarjeta, con el siguiente comando por consola:

$sudo apt-get install libgtk-3-dev.

La librería ruby-nfc pero esta ya la tenemos instalada del Puzle 1.

Y por último necesitamos la librería Logger que se utiliza para registrar mensajes durante la ejecución de un programa, y la librería thread que ambas ya vienen preinstaladas una vez descargas ruby.

Creamos el documento llamado pn532.rb:

```ruby
require 'ruby-nfc'


class Rfid

        def read_uid
                reader = NFC::Reader.all
                reader[0].poll(Mifare::Classic::Tag) do |tag|

                        uid = tag.uid_hex.upcase
                        return uid
                end
        end
end
```

Este código define una clase llamada Rfid, que se utiliza para interactuar con un lector de tarjetas RFID

utilizando la gema Ruby ruby-nfc. El método read_uid dentro de esta clase realiza la lectura del UID de una tarjeta RFID.

Dentro de este método, se obtiene una lista de todos los lectores NFC disponibles utilizando NFC::Reader.all. Luego, se realiza una llamada al método poll en el primer lector de la lista. Cuando se detecta una etiqueta Mifare Classic, se extrae su UID en formato hexadecimal.

Finalmente creamos el Main del programa.

```ruby
require "gtk3"
require_relative 'pn532'
require "thread"
require "ruby-nfc"

rf = Rfid.new
win = Gtk::Window.new("Gtk::Label sample")
box = Gtk::Box.new(:vertical)

label = Gtk::Label.new("Please, login with your university card",:expand => true)

label.override_background_color(0,Gdk::RGBA::new(0,0,1,1))
button = Gtk::Button.new(:label => "Clear")
label.set_size_request(100,100)
button.set_size_request(100, 200)
win.set_title("rfid_gdk.rb")
win.set_size_request(450,200)
win.set_window_position(:center)
win.signal_connect("destroy"){Gtk.main_quit}
box = Gtk::Box.new(:vertical,7)
box.add(label)


box.add(button)
win.add(box)
win.show_all

button.signal_connect("clicked") do
    label.set_markup("Please, login with your university card")
    label.override_background_color(0,Gdk::RGBA::new(0,0,1,1))
end

thread = Thread.new{
  while true do
  uid = rf.read_uid
  label.set_markup("uid "+ uid)
  label.override_background_color(0,Gdk::RGBA::new(1,0,0,1))

end
}

win.signal_connect("delete_event"){thread.kill;Gtk.main_quit}
Gtk.main
thread.join
```

Primeramente definimos las librerías que vamos a utilizar e importamos el archivo pn532.rb utilizando la ruta relativa al archivo actual.

Creamos la ventana de GTK con un mensaje de bienvenida y un botón "Clear" para restablecer el mensaje. El tamaño y la posición de la ventana se configuran, y se conecta la señal de destrucción de la ventana para salir del programa correctamente cuando se cierre la ventana.

Definimos un hilo en segundo plano que se ejecutará continuamente, leyendo el UID de la tarjeta RFID utilizando el objeto 'rf' de la clase 'Rfid'. Cuando se detecta una tarjeta RFID, se actualiza dinámicamente el mensaje de la etiqueta con el UID leído y se cambia el color de fondo de la etiqueta. Este proceso se repite indefinidamente mientras el programa esté en ejecución.

Finalmente, conectamos la señal de evento de eliminación de la ventana para detener el hilo en segundo plano y salir del programa correctamente cuando se cierre la ventana. El bucle principal de eventos GTK se inicia utilizando Gtk.main, lo que permite que la interfaz de usuario responda a las interacciones del usuario y las señales. Una vez que el bucle principal de eventos GTK termina, el hilo en segundo plano se detiene usando thread.kill, y el programa sale limpiamente.

La salida del programa al ejecutar el Main es la siguiente:

Y al introducir la tarjeta: