Segundo puzzle
Itead PN532 NFC module

Internet, configuraciones y librerías:

Durante la realización del primer puzzle, tuve algunos problemas de conexión relacionados con la compartición de internet que solo me permitía utilizar la Raspberry con una única red wifi, ya que al principio no utilizaba el cable ethernet para compartir internet con la Raspberry.

Con el cable ethernet ya conectado pude empezar a trabajar en el puzzle 2 con las mismas conexiones entre la Raspberry y el módulo PN532 sin más problemas, empezando por la instalación de la librería PyGObject:

sudo apt install python3-gi python3-gi-cairo gir1.2-gtk-3.0
sudo apt install libgirepository1.0-dev gcc libcairo2-dev pkg-config python3-dev gir1.2-gtk-3.0

Código:

Con el método Gtk.CssProvider() en el archivo del puzzle 2 me es posible cambiar el estilo de la ventana que queremos crear. Por lo tanto, he realizado un archivo estilos.css que me proporcionará los dos estilos que necesito:

```
.start{
    background: blue;
    color: white;
    font: 18px Calibri;
}
.uid_screen{
    background: red;
    color: white;
    font: 18px Calibri;
}
```

Utilizando este archivo y el del puzzle 1, importado en el del puzzle 2, podemos empezar a construir el programa.

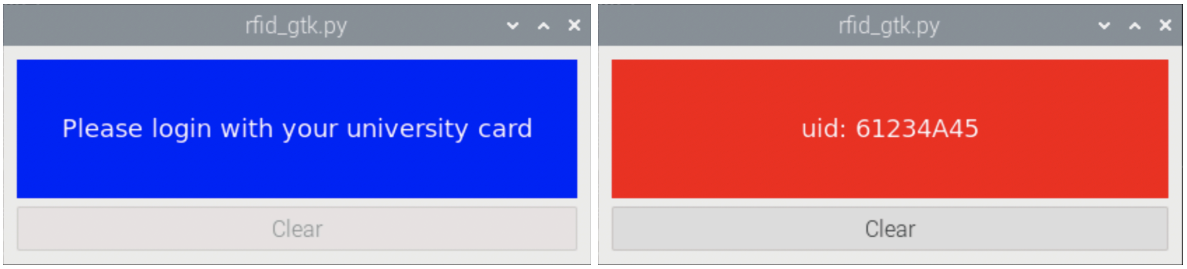
El código del puzzle 2, con el funcionamiento comentado en el mismo, es el siguiente:

```python
import threading
import gi
import puzzle1

gi.require_version("Gtk", "3.0")
from gi.repository import GLib, Gtk, Gdk #Importa GTK, GDK y GLib para estilo y ejecuciones en el hilo principal

#Define la clase principal para la ventana GTK
class Puzzle2(Gtk.Window):
    def __init__(self):
        super().__init__(title="rfid_gtk.py") #Inicializa la ventana
        self.set_border_width(10)  #Configura margen de pixeles

        #Crea ventana organizada verticalmente y los añade a la ventana principal
        box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        self.add(box)

        #Etiqueta del mensaje inicial (posicion, tamaño y agregado a la ventana)
        self.label = Gtk.Label(label="Please login with your university card")
        self.label.set_justify(Gtk.Justification.CENTER)
        self.label.set_size_request(400, 100)
        box.pack_start(self.label, True, True, 0)

        #Boton para "limpiar" la interfaz con metodo clicked inicialmente desactivado
        self.button = Gtk.Button(label="Clear")
        self.button.connect("clicked", self.clicked)
        self.button.set_sensitive(False)
        box.pack_start(self.button, True, True, 0)

        self.styles = Gtk.CssProvider() #Carga y aplica estilos CSS para la interfaz
        self.styles.load_from_path("estilos.css") #Lee el archivo CSS con los estilos escogidos
        Gtk.StyleContext.add_provider_for_screen(
            Gdk.Screen.get_default(), self.styles, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION
        )
        self.label.get_style_context().add_class("start")  #Aplica la clase CSS "start" a la etiqueta

        self.thread = None
        self.running = True

        self.connect("destroy", self.destroy)

        #Inicia el hilo de lectura
        self.start_reading_thread()

        #Muestra todos los widgets en la ventana
        self.show_all()

    #Metodo para iniciar el hilo de lectura NFC
    def start_reading_thread(self):
        if self.thread is None or not self.thread.is_alive():
            #Crea y empieza un nuevo hilo para leer el 'uid'
            self.thread = threading.Thread(target=self.read_uid)
            self.thread.daemon = True
            self.thread.start()

    #Lee el UID de la tarjeta mediante el puzzle1 y actualiza la interfaz para mostrar el UID
    def read_uid(self):
        while self.running:
            p1 = puzzle1.Rfid()
            uid = p1.read_uid()
            GLib.idle_add(self.update, uid)

    #Metodo para actualizar la interfaz cuando se detecta un UID
    def update(self, uid):
        self.label.get_style_context().remove_class("start")
        self.label.get_style_context().add_class("uid_screen")
        self.label.set_text(f"uid: {uid}")
        self.button.set_sensitive(True) #Activa el boton "Clear"
        self.running = False #Detiene el hilo de lectura
        return False

    #Metodo al hacer click al boton "Clear"
    def clicked(self, widget):
        #Restablece el mensaje inicial y el estilo de la etiqueta
        self.label.set_text("Please login with your university card")
        self.label.get_style_context().remove_class("uid_screen")
        self.label.get_style_context().add_class("start")
        #Desactiva el boton y reinicia la lectura
        self.button.set_sensitive(False)
        self.running = True
        self.start_reading_thread()

    #Metodo llamado al cerrar la ventana
    def destroy(self, widget):
        self.running = False  #Detiene el hilo de lectura
        self.thread.join()
        Gtk.main_quit()  #Sale de la aplicacion GTK

#Codigo main que inicia la aplicacion
if __name__ == "__main__":
    w = Puzzle2()
    Gtk.main()
```

El código crea una interfaz gráfica con GTK para detectar el identificador de una tarjeta. La clase 'Puzzle2', configura una ventana con una etiqueta inicial y un botón. Al iniciarse, se crea la interfaz, establece un estilo de CSS y lanza un hilo que ejecuta el método 'read_uid', encargado de detectar continuamente tarjetas mediante el 'puzzle1' hasta que encuentra una.

Se lee el UID de una tarjeta, al detectarlo, se cambia el texto de la etiqueta para mostrar el UID y cambia de color para reflejar el estado de detección de la tarjeta. También habilita el botón, que al hacer clic restaura la interfaz a su estado inicial, permitiendo nuevas lecturas. Cuando el usuario cierra la ventana, el método 'destroy' detiene el hilo de lectura y cierra la aplicación.

El resultado de la ejecución se puede ver en las imágenes:



Problemas:

El único problema que tuve y que sigo teniendo es a la hora de cerrar la aplicación GTK, que solo podía cerrarse una vez leída la tarjeta y finalizado el hilo. Porque cuando se intentaba cerrar la pestaña en el estado de espera se bloqueaba y no se podía cerrar hasta que el hilo se terminase, es decir, al acercar una tarjeta cuando se había intentado cerrar no se llegaba a imprimir por pantalla el UID, pero sí que se cerraba de golpe.

Intenté hacer que se pudiera cerrar igualmente con el estado de espera, pero no funcionó y me llevó a más errores, de modo que decidí dejarlo así.