Memòria puzzle 2

Procediment i biblioteques instal·lades

Aquest projecte és la continuació del puzzle 1. Consisteix en fer una interfície gràfica que demanés llegir i escriure el número de la tarjeta universitària que inclou un botó per netejar el número llegit.
Per fer aquesta feina he utilitzat el paquet de python PyGObject que inclou biblioteques basades en GObject. La informació necessària per començar el projecte al PyGObjecta la vaig trobar a: https://pygobject.gnome.org/index.html.
per instalar el sistema vaig seguir els següents passos del apartat getting started de la web anterior:

Instalación desde PyPI con pip:

1. Abre una terminal y entra en tu entorno virtual
2. Ejecutar para instalar las dependencias de compilación y GTK `sudo apt install libgirepository1.0-dev gcc libcairo2-dev pkg-config python3-dev gir1.2-gtk-4.0`
3. Ejecutar para compilar e instalar Pycairo `pip3 install pycairo`
4. Ejecutar para compilar e instalar PyGObject `pip3 install PyGObject`
5. Cambie el directorio de trabajo a donde `hello.py` se puede encontrar su script
6. Correr `python3 hello.py`

Després de seguir els passos se'm van descarregar aquests paquets:

```
(entorno) blanca@raspberrypi:~/entorno/lib/python3.11/site-packages $ ls
cairo                       pycairo-1.27.0.dist-info
_distutils_hack             pygobject-3.50.0.dist-info
distutils-precedence.pth    pygtkcompat
gi                          RPi
pip                         RPi.GPIO-0.7.1.dist-info
pip-23.0.1.dist-info        setuptools
pirc522                     setuptools-66.1.1.dist-info
pi_rc522-2.3.0.dist-info    spidev-3.6.dist-info
pkg_resources               spidev.cpython-311-arm-linux-gnueabihf.so
```

A més a més del PyGobject em vaig haber de descarregar la biblioteca glibc:

Una vegada tenia tot el necessari vaig fer el codi del puzzle 2 en el VisualStudio utilitzant el codi del puzzle 1 i totes les llibreries necessàries.

Problemes trobats

El primer problema va ser a l'hora de entendre el funcionament de les llibreries. Vaig haver de buscar informació a les pagines oficials i a altres fonts per poder fer el codi necessari.

L'altre problema amb el que em vaig trobar es que el codi del puzzle 1 el tenia malament plantejat i vaig haver de canviar-lo creant una classe i de manera que només fes una lectura de tarjeta ja que el bucle de llegir continuament el fa el codi del puzzle 2 (que demana el uid en bucle).

Codi ben format

puzle2:

```
#importacio llibreries necessaries i del puzzle 1
import gi
import threading
import time
from Puzzle1corregido import RFIDReader

#especificacio de la versio del gtk i importacio de les llibreries necessaries d'aquesta
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk, GLib, Gdk

class RFIDReaderApp(Gtk.Window):
    #constructor
    def __init__(self):
        #creo la window especificant tamany de la bora i la mida
        Gtk.Window.__init__(self, title="RFID Reader")
        self.set_border_width(10)
        self.set_default_size(600, 300)

        #creo un box en la window per afegir altres eines
        box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        self.add(box)

        #creo el label i el poso de color blau
        self.label = Gtk.Label(label="Please, login with your university card")
        self.label.override_background_color(Gtk.StateType.NORMAL, Gdk.RGBA(0,0,1,1))
        box.pack_start(self.label, True, True, 0)

        #creo el boto per netejar
        self.button= Gtk.Button(label="Clear")
        self.button.connect("clicked", self.button_on)
        box.pack_start(self.button,True, True, 0)

        #creo objecte de la clase rfidreader (puzzle1)
        self.rfid_reader=RFIDReader()
```

```
        #creo fil per llegir uid i l'executo
        self.thread = threading.Thread(target=self.read_rfid_thread, daemon=True)
        self.thread.start()

    #funcio per quan cliqui el buto es reinicii la finestra
    def button_on(self, widget):
        self.label.set_text("Please, login with your university card")
        self.label.override_background_color(Gtk.StateType.NORMAL, Gdk.RGBA(0,0,1,1))

    #funcio per llegir uid en bucle revisant que no sigui None
    def read_rfid_thread(self):
        while True:
            uid_hex= self.rfid_reader.lectura()
            if uid_hex is not None:
                self.label.set_text("UID: "+uid_hex)
                self.label.override_background_color(Gtk.StateType.NORMAL,
Gdk.RGBA(1,0,0,1))
            time.sleep(0.2)

def main():
    app = RFIDReaderApp()               #creo objecte clase puzle 2
    app.connect("destroy", Gtk.main_quit)#faig que es connecti de manera que ho pugui
tancar
    app.show_all()                  #faig que es mostri la finestra i el que passa
    Gtk.main()                      #executo el gtk

if __name__ == "__main__":
    main() #executo el main
```

Puzle1 corregit:

```
#Importo las librerias necesarias
import MFRC522
import signal
import time

class RFIDReader:
    #constructor de la clase
    def __init__(self):
        # Inicializamos el lector RFID y la variable para continuar la lectura
        self.reader = MFRC522.MFRC522()
        self.continue_reading = True
        print("Lector preparado, presiona Ctrl-C para detener la ejecucion.")

        # Configuramos la seÃ±al de interrupcion para terminar la lectura
        signal.signal(signal.SIGINT, self.end_read)
```

```
    # Metodo para convertir el UID en un string hexadecimal en el orden correcto
    def uid_to_string(self, uid):
        mystring = ""
        for i in uid:
            mystring += format(i, '02X')
        return mystring

    # Metodo que se ejecuta al recibir una seÃ±al de interrupcion (Ctrl+C)
    def end_read(self, signal, frame):
        self.continue_reading = False
        print("Ctrl+C capturado, terminando lectura.")

    # Metodo principal para realizara la lectura de las tarjetas
    def lectura(self):
        uid_hex=None #variable que devuelve el uid en hexadecimal
        if self.continue_reading:
            # Busca tarjetas cercanas
            (status, TagType) = self.reader.MFRC522_Request(self.reader.PICC_REQIDL)

            # Si se encuentra una tarjeta, intenta leer el UID
            if status == self.reader.MI_OK:
                (status, uid) = self.reader.MFRC522_SelectTagSN()

                # Si el UID se obtiene correctamente, lo guardo en variable
                if status == self.reader.MI_OK:
                    uid_hex=self.uid_to_string(uid)

            time.sleep(0.2)  # Pausa para dar tiempo a retirar la tarjeta
        return uid_hex #retorno el uid

# Bloque principal de ejecucion
if __name__ == "__main__":
    rfid_reader = RFIDReader() #Creo objeto de la clase
    rfid_reader.lectura() #Ejecuta el metodo lectura
```

Exemple execució