

Puzzle 2

Introducción:

Para empezar con la práctica, lo primero que hice fue corregir los errores que tenía en el puzzle 1.

En este caso, cuando hice el primer ejercicio no entendí que se tenía que reutilizar el código en el siguiente puzzle como una biblioteca. Por tanto, hice el código como un script para simplificarlo y ahorrarme trabajo.

En conclusión, edité el archivo para convertirlo en una función y le añadí un return para poder usar el valor del uid:

```
puzzle1.py > ...
1  import board
2  import busio
3  from digitalio import DigitalInOut
4  from adafruit_pn532.i2c import PN532_I2C
5
6  def puzzle1(d=False):
7      # Configurar I2C en la Raspberry Pi
8      i2c = busio.I2C(board.SCL, board.SDA)
9
10     # Inicializar el módulo PN532
11     pn532 = PN532_I2C(i2c, debug=d)
12
13     # Obtener el firmware para comprobar comunicación
14     ic, ver, rev, support = pn532.firmware_version
15     print(f"PN532 encontrado: chip {ic:#x}, versión {ver}.{rev}")
16
17     # Configurar el PN532 para leer tarjetas NFC
18     pn532.SAM_configuration()
19
20     print("Acerque una tarjeta NFC...")
21     while True:
22         uid = pn532.read_passive_target(timeout=1.0)
23         if uid:
24             print(f"Tarjeta detectada. UID: {uid.hex().upper()}")
25             return uid
26         break
27
28 def main():
29     puzzle1(False)
30
31 if __name__ == "__main__":
32     main()
```

Después, me di cuenta de que trabajar desde la terminal en remoto con la Raspberry Pi era bastante engorroso y nada práctico, sobre todo para moverme dentro del archivo .py y decidí trabajar con un editor de código en remoto siguiendo teniendo la conexión inalámbrica de la Raspberry Pi con los datos compartidos de mi móvil, pero en este

caso, también decidí fijar la IP así no tener que buscarla cada vez que quisiera conectarme.

Usé el editor de código de Microsoft, Visual Studio Code.



Visual Studio Code

Configuración:

En este caso, para desarrollar la versión gráfica tenía que usar las bibliotecas PyGObject y GTK3.

Por tanto, instalé las como administrador:

```
sudo apt update
```

```
sudo apt install -y python3-gi python3-gi-cairo gir1.2-gtk-3.0
```

```
pip install PyGObject (porque trabajo con un entorno virtual)
```

Cuando instalé lo necesario y cree mi primer código de prueba para ver como trabajaba las nuevas librerías que me había instalado y antes de hacer el CSS me salió este error en la terminal cuando ejecutaba el código

```
⊗ (entorno) marco@raspberrypi:~/Desktop/program $ python3 puzzle2.py
Traceback (most recent call last):
  File "/home/marco/Desktop/program/puzzle2.py", line 1, in <module>
    import gi
ModuleNotFoundError: No module named 'gi'
```

El error `ModuleNotFoundError: No module named 'gi'` indica que la biblioteca **PyGObject** (que contiene gi) no está instalada en tu sistema.

En este caso, volví a repetir los pasos anteriores de instalación de las bibliotecas y ahora me salía un fallo en la instalación porque no se pudo encontrar la dependencia **'cairo'**. Así que instalé las dependencias necesarias:

```
sudo apt update
```

```
sudo apt install -y python3-gi python3-gi-cairo gir1.2-gtk-3.0 libcairo2-dev  
libgirepository1.0-dev build-essential pkg-config
```

Una vez solucionado la dependencia **cairo** ahora me salía ERROR: Dependency **'girepository-2.0'** is required but not found.

Otra dependencia de GObject y que indica que falta la biblioteca **GObject Introspection** en mi sistema.

Volví a asegurarme que los paquetes necesarios estaban instalados correctamente:

```
sudo apt update
```

```
sudo apt install -y libgirepository1.0-dev gir1.2-gtk-3.0
```

y volví a ejecutar el programa, pero me decía lo mismo y me di cuenta de que se volvía en un bucle todo el rato del cual no sabía salir.

A continuación, voy a dejar constancia de todo lo que intenté para conseguir solucionar el error sin resultado:

1. Verifiqué si **'girepository-2.0'** estaba instalado

```
dpkg -l | grep girepository
```

El paquete 'libgirepository1.0-dev' no me aparecía e intenté volver a reinstalarlo con:

```
sudo apt install --reinstall libgirepository1.0-dev
```

Comprobé otra vez por si era posible que la instalación de GObject Introspection estuviera rota.

```
pkg-config --modversion gobject-introspection-1.0
```

2. Intenté instalar paquetes adicionales:

```
sudo apt install -y \ libgirepository1.0-dev \ gir1.2-glib-2.0 \  
gobject-introspection \ libgtk-3-dev
```

3. Verifiqué que '**PKG_CONFIG_PATH**' estaba bien configurado y configuré la variable manualmente:

```
export
PKG_CONFIG_PATH=/usr/lib/aarch64-linux-gnu/pkgconfig:/usr/share/pkgconfig
```

4. Borré el caché de instalación y reinstalé PyGObject:

```
pip install --no-cache-dir --force-reinstall PyGObject
```

5. Intenté crear una nueva carpeta en el Desktop y volver a hacer todo desde el principio añadiendo el puzzle1.py, las bibliotecas correspondientes de este archivo y de nuevo todo el código y bibliotecas del puzzle2

Imagen del error:

```
(entorno) marco@raspberrypi:~/Desktop/program $ pip install PyGObject
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting PyGObject
  Using cached pygobject-3.52.3.tar.gz (1.2 MB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... error
error: subprocess-exited-with-error

× Preparing metadata (pyproject.toml) did not run successfully.
  exit code: 1
  [23 lines of output]
+ meson setup /tmp/pip-install-7q5ws9ir/pygobject_0cc79796b5d04e2aa50bc87f2b6727ba /tmp/pip-install-7q5ws9ir/pygobject_0cc79796b5d04e2aa50bc87f2b6727ba/.mesonpy-1tvfdjt7 -Dbuildtype=release -Db_ndebug=if-release -Db_vscrt=md -Dtests=false -Dwheel=true --wrap-mode=nofallback --native-file=/tmp/pip-install-7q5ws9ir/pygobject_0cc79796b5d04e2aa50bc87f2b6727ba/.mesonpy-1tvfdjt7/meson-python-native-file.ini
The Meson build system
Version: 1.7.0
Source dir: /tmp/pip-install-7q5ws9ir/pygobject_0cc79796b5d04e2aa50bc87f2b6727ba
Build dir: /tmp/pip-install-7q5ws9ir/pygobject_0cc79796b5d04e2aa50bc87f2b6727ba/.mesonpy-1tvfdjt7
Build type: native build
Project name: pygobject
Project version: 3.52.3
C compiler for the host machine: cc (gcc 12.2.0 "cc (Debian 12.2.0-14) 12.2.0")
C linker for the host machine: cc ld.bfd 2.40
Host machine cpu family: aarch64
Host machine cpu: aarch64
Program python3 found: YES (/home/marco/Desktop/program/entorno/bin/python3)
Found pkg-config: YES (/usr/bin/pkg-config) 1.8.1
Run-time dependency python found: YES 3.11
Found CMake: /usr/bin/cmake (3.25.1)
Run-time dependency girepository-2.0 found: NO (tried pkgconfig and cmake)
Not looking for a fallback subproject for the dependency girepository-2.0 because:
Use of fallback dependencies is disabled.

../meson.build:31:9: ERROR: Dependency 'girepository-2.0' is required but not found.

A full log can be found at /tmp/pip-install-7q5ws9ir/pygobject_0cc79796b5d04e2aa50bc87f2b6727ba/.mesonpy-1tvfdjt7/meson-logs/meson-log.txt
[end of output]

note: This error originates from a subprocess, and is likely not a problem with pip.
error: metadata-generation-failed

× Encountered error while generating package metadata.
  See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

Problemas:

Aparte de todo lo comentado anteriormente, cuando volví a hacer la carpeta en el Desktop para volver a hacerlo todo desde 0 y tuve que comprobar otra vez el puzzle1 no me detectaba el NFC la Raspberry Pi.

Usé el comando **'i2cdetect -y 1'** que escanea los dispositivos conectados al bus I2C en la Raspberry Pi y esperando que me saliera lo correcto que es lo siguiente:

```
marco@raspberrypi:~ $ 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: -- -- -- -- -- -- -- -- -- -- -- --
```

para mi sorpresa, no me detectaba nada.

Ya había tenido este problema en la práctica anterior, por tanto, con lo que aprendí, intenté jugar con los interruptores y reiniciando el NFC e incluso alimentándolo con la entrada de 5 V por si el voltaje no era suficiente, pero a veces me detectaba el pin, pero se iba, otras no los detectaba e incluso una vez me detectó un pin que no tenía nada que ver:

```
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- 4c -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- --
```

Dentro de mi poca experiencia, llegué a la conclusión de que había problemas de conexión y quizás no estaba bien conectado o se cortocircuitaban los pines o se me había roto el aparato o la Raspberry Pi.

Programa Final:

Con todo lo comentado anteriormente, no tuve la suerte de poder probar mi código final ni siquiera probar el CSS.

Entonces no sé que tengo incorrecto o que podría mejorar por temas de optimización.

Aun así, aquí está mi idea de código final:

```
puzzle2.py > NFCApp > clear_label
1  import gi
2  import threading
3  from puzzle1 import puzzle1
4  from gi.repository import Gtk, GLib
5
6
7  class NFCApp(Gtk.Window):
8      def __init__(self):
9          super().__init__(title="NFC Puzzle 2")
10         self.set_default_size(400, 200)
11
12         # Contenedor vertical
13         vbox = Gtk.VBox(spacing=10)
14         self.add(vbox)
15
16         # Etiqueta de instrucciones
17         self.label_info = Gtk.Label(label="Please, login with your NFC card")
18         vbox.pack_start(self.label_info, True, True, 0)
19
20         # Etiqueta para mostrar el UID
21         self.label_uid = Gtk.Label(label="UID: ----")
22         vbox.pack_start(self.label_uid, True, True, 0)
23
24         # Botón para limpiar
25         self.button_clear = Gtk.Button(label="Clear")
26         self.button_clear.connect("clicked", self.clear_label)
27         vbox.pack_start(self.button_clear, True, True, 0)
28
29         # Hilo para leer NFC
30         self.nfc_thread = threading.Thread(target=self.read_nfc, daemon=True)
31         self.nfc_thread.start()
32
33     def read_nfc(self):
34         """Ejecuta la función puzzle1() y actualiza la interfaz cuando se detecta una tarjeta."""
35         uid = puzzle1()
36         if uid:
37             GLib.idle_add(self.update_label, uid.hex().upper()) # Actualizar interfaz gráfica
38
39     def update_label(self, uid_hex):
40         """Actualizar la etiqueta del UID en la interfaz."""
41         self.label_uid.set_text(f"UID: {uid_hex}")
42
43     def clear_label(self, widget):
44         """Limpiar el UID cuando se presiona el botón."""
45         self.label_uid.set_text("UID: ----")
46
47
48     def main():
49         app = NFCApp()
50         app.connect("destroy", Gtk.main_quit)
51         app.show_all()
52         Gtk.main()
53
54
55 if __name__ == "__main__":
56     main()
```