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 puzle 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 > ...

```python
import board
import busio
from digitalio import DigitalInOut
from adafruit_pn532.i2c import PN532_I2C

def puzzle1(d=False):
    # Configurar I2C en la Raspberry Pi
    i2c = busio.I2C(board.SCL, board.SDA)

    # Inicializar el módulo PN532
    pn532 = PN532_I2C(i2c, debug=d)

    # Obtener el firmware para comprobar comunicación
    ic, ver, rev, support = pn532.firmware_version
    print(f"PN532 encontrado: chip {ic:#x}, versión {ver}.{rev}")

    # Configurar el PN532 para leer tarjetas NFC
    pn532.SAM_configuration()

    print("Acerque una tarjeta NFC...")
    while True:
        uid = pn532.read_passive_target(timeout=1.0)
        if uid:
            print(f"Tarjeta detectada. UID: {uid.hex().upper()}")
            return uid
            break

def main():
    puzzle1(False)

if __name__ == "__main__":
    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.

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 PyGObject 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:

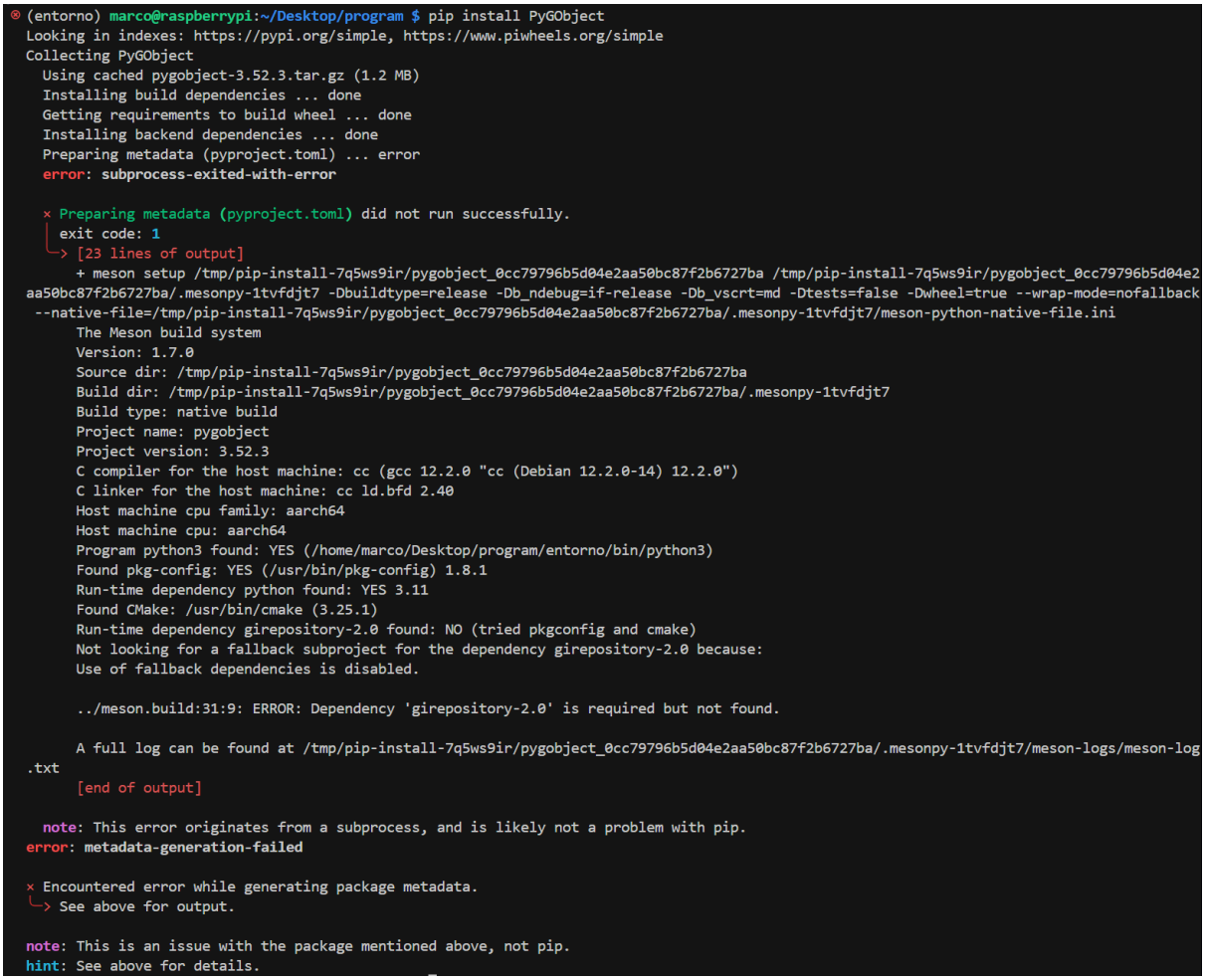
   export PKG_CONFIG_PATH=/usr/lib/aarch64-linux-gnu/pkgconfig:/usr/share/pkgconfig

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

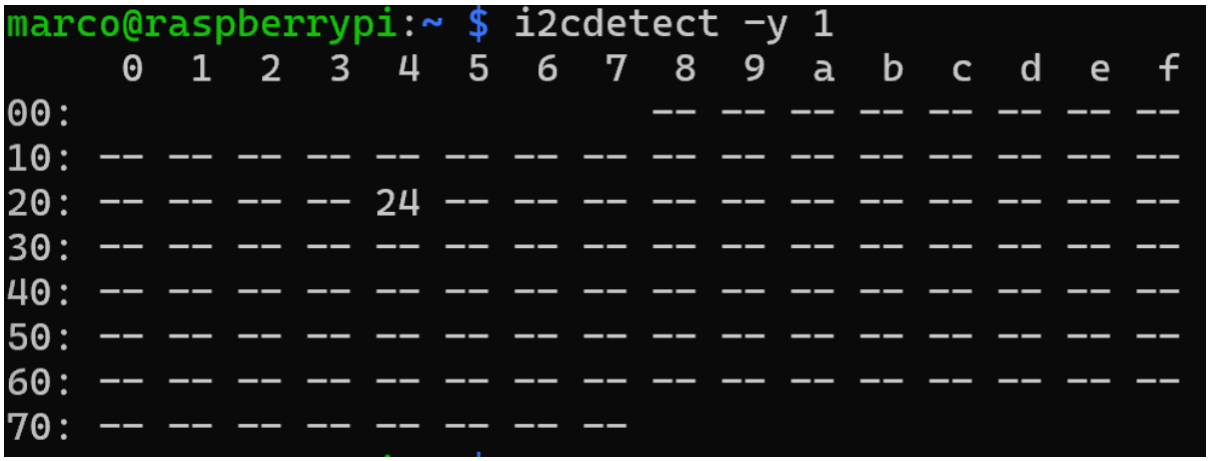
Imagen del error:



Problemas:

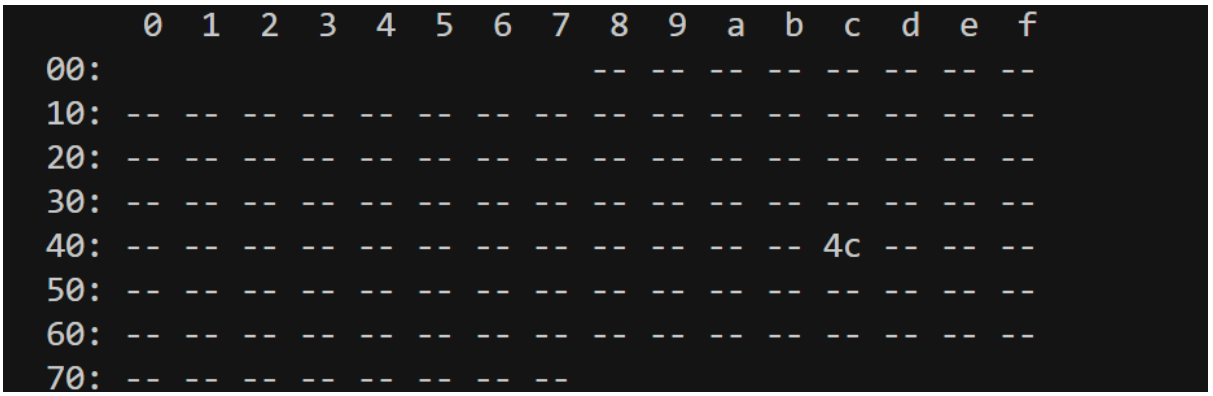
Aparte de todo lo comentado anteriormente, cuando volví a hacer la carpeta en el Desktop para volver a hacerlo todo desde 0 y tuve que comprobar otra vez el puzzle1 no me detectaba el NFC la Raspberry Pi.

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:



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:



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

```python
import gi
import threading
from puzzle1 import puzzle1
from gi.repository import Gtk, GLib


class NFCApp(Gtk.Window):
    def __init__(self):
        super().__init__(title="NFC Puzzle 2")
        self.set_default_size(400, 200)

        # Contenedor vertical
        vbox = Gtk.VBox(spacing=10)
        self.add(vbox)

        # Etiqueta de instrucciones
        self.label_info = Gtk.Label(label="Please, login with your NFC card")
        vbox.pack_start(self.label_info, True, True, 0)

        # Etiqueta para mostrar el UID
        self.label_uid = Gtk.Label(label="UID: ----")
        vbox.pack_start(self.label_uid, True, True, 0)

        # Botón para limpiar
        self.button_clear = Gtk.Button(label="Clear")
        self.button_clear.connect("clicked", self.clear_label)
        vbox.pack_start(self.button_clear, True, True, 0)

        # Hilo para leer NFC
        self.nfc_thread = threading.Thread(target=self.read_nfc, daemon=True)
        self.nfc_thread.start()

    def read_nfc(self):
        """Ejecuta la función puzzle1() y actualiza la interfaz cuando se detecta una tarjeta."""
        uid = puzzle1()
        if uid:
            GLib.idle_add(self.update_label, uid.hex().upper())  # Actualizar interfaz gráfica

    def update_label(self, uid_hex):
        """Actualizar la etiqueta del UID en la interfaz."""
        self.label_uid.set_text(f"UID: {uid_hex}")

    def clear_label(self, widget):
        """Limpiar el UID cuando se presiona el botón."""
        self.label_uid.set_text("UID: ----")


def main():
    app = NFCApp()
    app.connect("destroy", Gtk.main_quit)
    app.show_all()
    Gtk.main()


if __name__ == "__main__":
    main()
```