

# PUZZLE 2

## Llibreries

Per començar a programar, ens cal instalar la llibreria gtk 4. Això ho he fet amb aquests comandos.

```
(myenv) pi@raspberrypi:~/Proyectos $ sudo apt-get install libgtk-4-dev  
(myenv) pi@raspberrypi:~/Proyectos $ pip install pygobject
```

Pero al intentar instalar el pygobject, em surt el següent error:

```
ERROR: Dependency 'girepository-2.0' is required but not found.
```

He intentat solucionar-ho varios cops, pero no he pogut. Llavors he optat per utilitzar un altre llibreria.

Aquesta llibreria es diu **tkinter**, la qual he descarregat amb el següent comando:

```
(myenv) pi@raspberrypi:~/Proyectos $ sudo apt-get install python3-tk
```

Aquesta llibreria em permet crear un label, que es el que se'ns demana en el puzzle 2.

## Codi

Primer de tot, per poder programar en base al programa del puzzle 1, he fet uns canvis en aquest.

```
Python  
import time  
import serial  
from adafruit_pn532.uart import PN532_UART  
  
class RFID:  
    def __init__(self, port='/dev/serial0', baudrate=115200, timeout=1,  
    terminal=False):  
        # Configuración del UART usando pyserial  
        self.uart = serial.Serial(port, baudrate=baudrate, timeout=timeout)
```

```

# Configuración del módulo PN532
self.pn532 = PN532_UART(self.uart, debug=False) # Deshabilitar
depuración
self.pn532.SAM_configuration()

# Almacenar si se deben imprimir mensajes en la terminal
self.terminal = terminal

def read_card(self):
    waiting_message_printed = False

    while True:
        uid = self.pn532.read_passive_target(timeout=1)
        if uid is None:
            if self.terminal and not waiting_message_printed:
                print("Esperando una tarjeta...")
                waiting_message_printed = True
            time.sleep(1)
            continue
        # Convertir el UID a una cadena hexadecimal concatenada
        uid_str = ''.join([f'{i:02X}' for i in uid])

        # Imprimir el UID solo si terminal está habilitado
        if self.terminal:
            print(f'UID de la tarjeta encontrada: {uid_str}')

    return uid_str # Devolver el UID cuando se encuentre una tarjeta

# Ejecutar el código de prueba si este archivo es ejecutado directamente
if __name__ == "__main__":
    rfid = RFID(terminal=True) # Activamos los mensajes de la terminal solo
    cuando se ejecuta directamente
    rfid.read_card() # Llamamos al método que leerá la tarjeta

```

He convertit tot el que tenia en forma de classe, per així poder utilitzar la funció de `read_card()`. També he fet que el programa pugui escollir si mostrar el contingut a la terminal a través d'una variable “terminal” (si volem que es mostri, haurem de posar `terminal = True`, sino `terminal = False`). Ja que en el puzzle 1 si que ens demanen que es mostri el contingut de la tarjeta a la terminal, pero en el puzzle 2 només ha d'aparèixer en el label.

Dit això, explicaré com he muntat el meu programa del puzzle 2 per parts.

## Import de llibreries

Python

```
import threading
import tkinter as tk
from tarjetas import RFID
```

He utilitzat 3 llibreries. Aquí explico perquè d'aquestes 3:

- **threading**: Aquesta llibreria permet que crear un fil separat que executa la funció `read_rfid()`. Això permet que la interfície gràfica d'usuari estigui receptiva mentres es realitza la lectura continua de la tarjeta RFID en segon pla.
- **tkinter**: Aquesta llibreria permet crear la interfície gràfica d'usuari.
- **tarjetas**: “Tarjetas” és el nom del fitxer del programa del puzzle 1, la qual importo la classe `RFID` per poder utilitzar la funció `read_card()`.

## Inicialització del programa

Python

```
class RFIDApp:
    def __init__(self, root):
        self.root = root
        self.root.title("interfaz.py")

        # Variable booleana para controlar la lectura
        self.read = False

        # Label para mostrar el estado o el UID
        self.label = tk.Label(root, text="Please, login with your university
card", width=40, height=4, font=("Verdana", 16), bg="lightblue", fg="black",
bd=2, relief="solid", padx=10, pady=10)
        self.label.pack(pady=20)

        # Botón para limpiar
        self.clear_button = tk.Button(root, text="Clear",
command=self.clear_label, width=70)
        self.clear_button.pack()

        # Instancia del lector RFID sin mensajes en la terminal
        self.rfid = RFID(terminal=False) # No imprimir mensajes en la terminal
```

```
# Iniciar el hilo para leer la tarjeta
self.thread = threading.Thread(target=self.read_rfid, daemon=True)
self.thread.start()
```

Aquí és on creo la interfície gràfica. On primer creo un títol, que li dic com **interfaz.py**, que es el nom del meu programa del puzzle 2. Després creo el label on es mostrerà la uid de la tarjeta RFID. Seguit de el botó per borrar la uid del label per a que torni a sortir el missatge de “Please, login with your university card”. Després defineixo la variable “**terminal**” del puzzle 1 com a fals, ja que no vull que es mostri el contingut a la terminal. Per últim defineixo una variable booleana “**read**”, que controla si s’ha llegit una tarjeta per tal de que no es pugui torna a llegir un altre fins que s’hagi utilitzat el botó **clear**.

Cal comentar que l’ordre en el que es crea el contingut de la interfície gràfica és important, ja que el programa construeix aquesta en l’ordre en el que el programador el posi.

### **read\_rfid(self)**

Python

```
def read_rfid(self):
    """Hilo que lee la tarjeta y actualiza la UI"""
    while True:
        if not self.read:
            uid_str = self.rfid.read_card()
            if uid_str:
                self.update_label(f"UID: {uid_str}")
```

Aquesta funció permet llegir la tarjeta constantment la tarjeta RFID a través de la funció `read_card()` del puzzle 1.

## update\_label(self,info)

Python

```
def update_label(self, info):
    """Actualizar el Label en el hilo principal"""
    self.label.config(text=info, bg="red")
    self.read = True # Establecer la variable booleana en True
```

Aquesta funció permet actualitzar el contingut del label, inserint el text del paràmetre “**info**”, que correspon a la uid de la tarjeta RFID. També posa la variable “**read**” en true, per indicar que s’ha llegit una tarjeta, lo que farà que no es pugui tornar a llegir un altre fins que es pulsi el botó clear.

## clear\_label(self)

Python

```
def clear_label(self):
    """Limpiar el Label al presionar el botón"""
    self.label.config(text="Please, login with your university card",
bg="lightblue")
    self.read = False # Restablecer la variable booleana en False
```

Aquesta funció també actualitza el label, inserint el missatge “Please, login with your university card”. A part també canvia el valor de “**read**” a false, per indicar que es pot llegir un altre tarjeta.

## main

Python

```
if __name__ == "__main__":
    root = tk.Tk()
    app = RFIDApp(root)
    root.mainloop()
```

Per últim tenim el “**main**”. On primer creem una instància de la finestra principal. Seguit de la creació de la instància de la classe RFIDApp. I per últim, iniciem el bucle principal del programa, que espera events com la lectura de la tarjeta i l’utilització dels botons.

L'intefície gràfica queda així:

