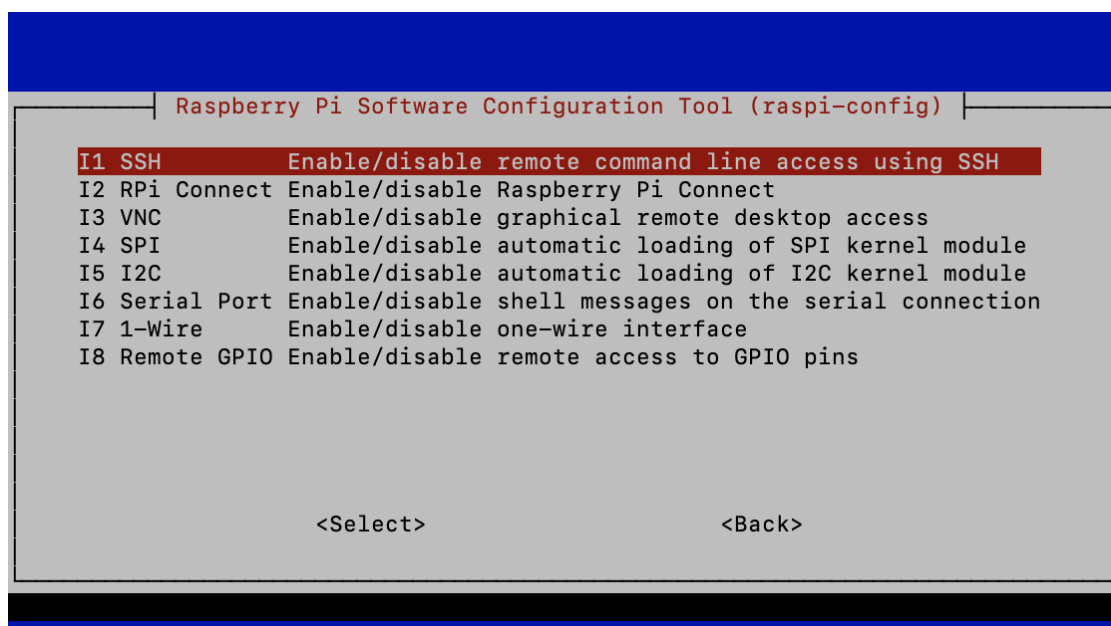


Memòria Puzzle 2

RC522

Configuracions realitzades.

En quant a configuracions per a la realització d'aquest puzzle, no hi ha varietat respecte a les fetes per al primer exercici, per tant no entrarem massa en detall. Recordem però: la connexió placa-portàtil la fem via cable *ethernet*; per a poder treballar des del programa VNC Viewer, activem l'opció des del menú de la raspberry i fem el mateix amb el SPI.



Finalment, connectem el perifèric RC522 a la placa segons les connexions que es mostren a la taula següent:

RC522 Pin	GPIO Pin Raspberry Pi
SDA	Pin 24 (GPIO 8)
SCK	Pin 23 (GPIO 11)
MOSI	Pin 19 (GPIO 10)
MISO	Pin 21 (GPIO 9)
IRQ	No conectado
GND	Pin 6 (Ground)
RST	Pin 22 (GPIO 25)
3.3V	Pin 1 (3.3V)

Biblioteques i paquets instal·lats.

Per a aquest exercici, afegim la biblioteca Gtk3 i fem import de *threading* i *time* per a poder crear fils i temporitzadors, dels quals veurem la seva utilitat més endavant en la memòria.

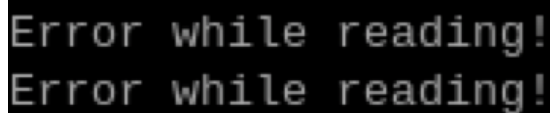
Problemes Trobats.

La principal dificultat ha estat saber com programar la interfície gràfica demanada. El fet de no haver fet mai un exercici similar suposava un repte. Per sort, existeixen moltes explicacions i exemples de com fer-ho, motiu pel qual he pogut superar aquesta dificultat.

La següent dificultat ha estat el tema de posar colors a la interfície. Inicialment només vaig trobar la manera de posar un fons de color al text que hi posés, en comptés de a tot el display que és el que volia aconseguir. Novament, gràcies a fer recerca a internet he pogut trobar la solució a aquest problema.

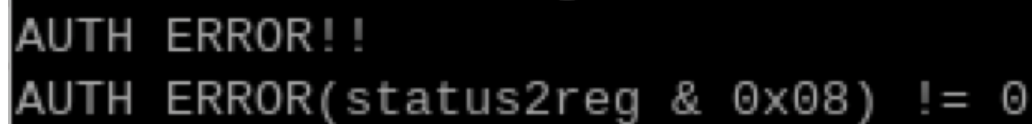
Ja per últim, anem amb els errors trobats al executar el codi, que no han estat massa nombrosos:

El primer es el *Error while reading!* que surt al intentar fer la lectura del UID de la targeta. Com bé indica, aquest error es deu a que no s'ha pogut fer la lectura correctament per qualsevol circumstància i només ens queda torna a provar a fer-la de nou.



```
Error while reading!  
Error while reading!
```

Lligat amb això, també em surt aquest error:



```
AUTH ERROR!!  
AUTH ERROR(status2reg & 0x08) != 0
```

Inicialment pensava que era alguna cosa del codi que estava errònia, pero resulta que és un altre error de lectura que en la majoria de casos es dona quan acostes l'identificador en un angle "extrany" i que no permet una correcta interpretació del UID. La solució és simplement tornar a provar mirant de posar la targeta el més recta possible.

Finalment, un altre problema trobat ha estat el fet de que quan acostava l'identificador al lector, feia moltes lectures seguides, la qual cosa he solucionat posant un temporitzador i creant una variable de control per evitar que es llegeixin 2 o més identificadors a la vegada.

Codi.

Aquest és el codi proposat:

```
import threading
from Puzzle1 import Rfid
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk, GLib, Gdk
import time

class RC522App:
    def __init__(self):
        # Configuracio de la interficie grafica
        self.window = Gtk.Window(title="Lector RC522")
        self.label = Gtk.Label(label="")
        self.clear_button = Gtk.Button(label="Esborrar")
        self.display_label = Gtk.Label(label="")

        self.clear_button.connect("clicked", self.clear_display)

        # Layout
        vbox = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=0)
        vbox.pack_start(self.label, False, False, 0)
        vbox.pack_start(self.display_label, False, False, 10)
        vbox.pack_start(self.clear_button, False, False, 0)

        self.window.add(vbox)
        self.window.connect("destroy", Gtk.main_quit)
        self.window.show_all()

    # Inicialitzem el lector RC522
    self.rfid = Rfid()
    self.card_read = False # Variable per controlar si hi ha una targeta llegada

    # Iniciem el fil per llegir des del lector
    self.run_rfid_thread()

    def clear_display(self, button):
        self.display_label.set_text("Display esborrat!")
        self.display_label.override_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(0, 0, 0, 1)) # Text negre en reiniciar
        self.window.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(255, 0, 0, 1)) # Fons vermell
        self.card_read = False # Reinicialitzar l'estat per poder llegir una nova targeta

        # Utilitzar GLib.timeout_add per esperar 3 segons (3000 mililisecons) abans de restaurar el format original
        GLib.timeout_add(3000, self.restore_default_format)

    def restore_default_format(self):
        # Aquesta funcio restaura el text original despres de 3 segons
        self.display_label.set_text("Aproximi la targeta...")
        self.display_label.override_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(1, 1, 1, 1)) # Text blanc
        self.window.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(0, 0, 1, 1)) # Fons blau
        return False # Retornar False per indicar que el temporitzador nomes ha de correr una vegada

    def run_rfid_thread(self):
        # Iniciem el fil per llegir en segon pla
        threading.Thread(target=self.read_rfid, daemon=True).start()

    def read_rfid(self):
        while True:
            if not self.card_read: # Nomes llegir si no hi ha una targeta ja llegada
                GLib.idle_add(self.update_display_blue) # Mostrem el fons blau quan esperem la targeta
                uid = self.rfid.read_uid()
                if uid:
                    self.card_read = True # Marquem que hi ha una targeta present
                    GLib.idle_add(self.update_display_green, uid) # Actualitzem a verd
                    # Esperem que la targeta es tregui del lector abans de llegir de nou
                    self.wait_until_card_removed()

    def wait_until_card_removed(self):
        """Aquesta funcio espera fins que es retiri la targeta abans de permetre una nova lectura"""
        while self.card_read:
            try:
                id, text = self.rfid.reader.read_no_block() # No bloqueja el programa si no hi ha targeta
                if not id: # Si no hi ha targeta, permet una nova lectura
                    print("Targeta retirada, preparat per llegir una nova targeta.")
                    time.sleep(3)
                    self.card_read = False
            except Exception as e:
                print(f"Error: {e}")
```

```
def update_display_blue(self):
    """Actualitza el display a blau quan estem esperant la targeta"""
    self.display_label.set_text("Aproximi la targeta...")
    self.display_label.override_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(1, 1, 1, 1)) # Text blanc
    self.window.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(0, 0, 1, 1)) # Fons blau

    return False

def update_display_green(self, uid_hex):
    """Actualitza el display a verd amb el UID llegit"""
    self.display_label.set_text(f"UID: {uid_hex}")
    self.display_label.override_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(0, 0, 0, 1)) # Text negre
    self.window.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(0, 1, 0, 1)) # Fons verd

    return False

if __name__ == "__main__":
    app = RC522App()
    Gtk.main()
```

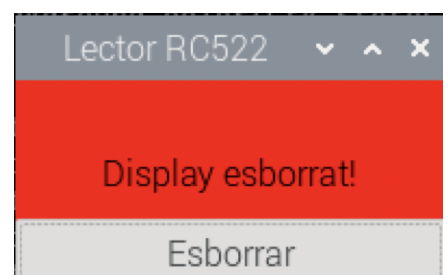
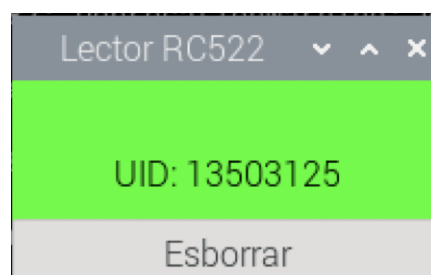
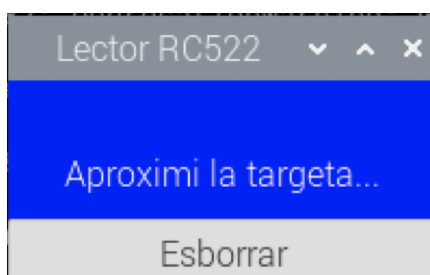
A mode de breu explicació del codi:

Primerament fem els imports necessaris, tant de les biblioteques com del puzzle 1. Seguidament, inicialitzem i configurem la interfície gràfica, de manera que quedin els títols que volem i l'espaiat que creiem necessari. Aleshores ja creem un nou fil que farà la tasca de lectura.

Com ja he mencionat anteriorment, he creat una variable de control (`card_read`) que evita llegir més d'un UID simultàneament. Aquesta variable prendrà el valor *False* si no s'ha fet cap lectura i *True* si sí que se n'ha fet, de manera que únicament permetrà actuar al lector si encara no ha llegit cap targeta.

També, mitjançant un temporitzador, controlo que, al cap d'un temps determinat a la meua elecció, un cop s'ha fet una lectura, l'estat de la variable de control passi a *False* i així poder repetir el cicle, sempre llegint un sol UID a la vegada.

Finalment, he fet que el fons del display canviï de color segons la situació en la que ens trobem: Si estem esperant a fer la lectura, es mantindrà en blau. Un cop feta, es mostrarà l'identificador per pantalla amb el fons verd (indicatiu de que la lectura ha estat correcta). I, si per algun motiu, volem esborrar el display, sortirà un text indicant-ho i el fons en vermell (Notar que, en cada cas, també he ajustat el color del text per a que es pugui llegir de la manera més nítida possible)



A continuació, mostro el que es veu al terminal quan s'executa el codi

```
Aproximi la targeta al lector...  
Targeta retirada, preparat per llegir una nova targeta.  
Aproximi la targeta al lector...  
□
```