# Puzle 1 - LCD

## 1. Pas previ: habilitació del protocol I2C a la Raspberry

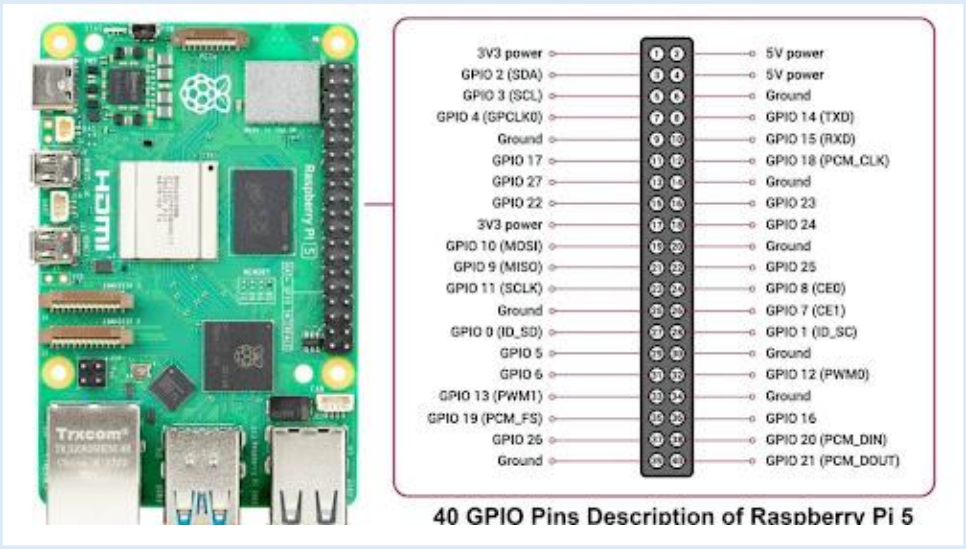
Per poder fer servir una pantalla LCD amb interfície I2C com la que disposem, és necessari i imprescindible habilitar el protocol I2C. Des de l'entorn gràfic de Raspberry Pi OS es pot fer de manera senzilla des del menú d'inici, entrant a preferències, després a la configuració de la Raspberry Pi i a la pestanya d'interfícies i activant la opció i2C.

```
Start → Preferences → Raspberry Pi Configuration → Interfaces → I2C
```

Després de fer això caldrà reiniciar el sistema per aplicar els canvis. Alternativament, es pot activar també des del terminal amb el comandament "sudo raspi-config", que ens obriria una finestra des d'on també es por activar.

## 2. Connectar el display LCD a la Raspberry Pi

Amb el protocol I2C activat, ara cal connectar el display a la nostra Raspberry Pi. El display va muntat i connectat a un mòdul PCF8574, que simplifica la connexió i permet la comunicació mitjançant I2C amb només quatre cables: $V_{cc}$, GND, SDA i SCL. Buscant el pinout de la Raspberry Pi que estem fent servir (en aquest cas la Pi 5, que és idèntic al de models anteriors), trobem el següent esquema:




Ens indica que les connexions que hem de realitzar son les següents:

| Número de pin de Raspberry Pi | Connexió pertinent |
|---|---|
| 3 (fila 2, columna 1) | SDA |
| 4 (fila 2, columna 2) | $V_{CC}$ |
| 5 (fila 3, columna 1) | SLC |
| 6 (fila 3, columna 2) | GND |

Una vegada s'ha activat el protocol I2C i s'ha connectat el display a la Raspberry, podem comprovar la direcció del dispositiu instal·lant les eines I2C amb els següents dos comandaments al terminal:

```
sudo apt install -y i2c-tools
i2cdetect -y 1
```

Amb això, trobem que la direcció del display LCD és 0x27:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Aquesta informació serà important més endavant.

## 3. Creació d'un entorn virtual

Per evitar conflictes amb paquets del sistema, és necessari crear un entorn virtual en Python, des del qual farem tota la programació requerida. Per fer-ho, primer ens assegurem de que Virtual Environment ("venv") estigui instal·lat amb el següent comandament al terminal:

```
sudo apt install -y python3-venv
```

Ara, creem l'entorn virtual al directori a on es treballarà. Ho he fet a la carpeta principal del sistema (/home):

```
python3 -m venv lcd_env
```

Una vegada fet això, ja podem activar l'entorn que acabem de crear, i que s'haurà d'activar cada vegada que vulguem fer ús del perifèric:

```
source lcd_env/bin/activate
```

Amb això, ja tenim llest l'entorn virtual des del qual treballarem.

## 4. Instal·lació de dependències i programació en Python

Una vegada ja tenim el display LCD connectat i detectat a la Raspberry i un entorn virtual preparat, ja podem començar la part de programació.

Primer, hem d'instal·lar una llibreria que ens permeti interactuar amb el display. He escollit la llibreria RPLCD ja que permet controlar displays LCD basats en el controlador PCF8574 fàcilment. Per instal·lar-la, escrivim al terminal:

```
pip install RPLCD
```

Després, ja podem començar amb el programa en Python. Primer, hem d'importar els mòduls `sys`, per llegir l'entrada de l'usuari, i `CharLCD`, de RPLCD-i2c i que ens permetrà controlar el display LCD.

El codi inicial que he fet és el següent:

```
import sys
from RPLCD.i2c import CharLCD

lcd = CharLCD('PCF8574', 0x27, cols=20, rows=4) # Inicialitza el display
user_text = sys.stdin.read().strip() # Llegeix l'string de l'usuari

lcd.write_string(user_text) # Escriu directament l'string al display LCD
```

Amb `CharLCD`, inicialitzem el display amb tota la informació del nostre model (controlador, direcció, columnes i files). Amb la variable `user_text`, llegim el text que passa per teclat l'usuari (`.strip()` elimina possibles espais a l'inici o al final del text). Amb la funció `lcd.write_string`, directament mostrem el text que ha escrit l'usuari.

No obstant, trobem una sèrie d'inconvenients amb aquesta implementació:

- No es poden mostrar alguns caràcters especials, com ara "ç", "ñ" o qualsevol lletra amb accent (encara que alguns signes de puntuació, com punts, comes, parèntesis o guions sí es mostren correctament).
- Els "intro" no es mostren correctament. En comptes de passar a la següent línia, quan s'escriu un, es mostren 20 caràcters "en blanc", de manera que si s'escriu a la meitat d'una línia, la següent continuarà a la columna següent de la següent fila (i no a la columna 1 de la següent fila, com s'esperaria).
- Els caràcters que superen el límit (80 en total), es mostren des del principi (fila 1, columna 1), esborrant els que en teoria haurien d'anar en aquelles posicions.

Mentre que el primer és bastant complex de solucionar (ja que requeriria alguna funció que ens permetés escriure caràcters personalitzats), els dos últims es troben solucionats en la versió que es mostra a continuació:

```
import sys
from RPLCD.i2c import CharLCD

lcd = CharLCD('PCF8574', 0x27, cols=20, rows=4)

user_text = sys.stdin.read().strip().split("\n")[:4] # Llegir i dividir el text
for i, line in enumerate(user_text): # Escriure les linies al LCD
    lcd.cursor_pos = (i, 0)
    lcd.write_string(line[:20]) # Escriu cada linia truncada a 20 caracters
```

Això és degut a que el text que es guarda a `user_text` es separa per canvis de línies "\n" (màxim 4) i el bucle `for` mostra al display els 20 primers caràcters de cada línia. No obstant, amb aquesta versió del codi, cal introduir canvis de línia per passar a la següent (no com abans, que era suficient amb superar els 20 caràcters per línia).

Tots dos codis es troben a la al meu repositori de Github, el primer amb el nom “puzle1.py” i el segon amb el nom “puzle1millorat.py”: https://github.com/davmarin/lab_pbe/