Puzzle 2 - LCD en Ruby

L'objectiu d'aquest puzzle era crear una finestra que complís la mateixa funció que el primer puzzle. La finestra té un botó que quan es prem, dona a veure la frase, que s'ha de poder escriure a la finestra, a la LCD.

Per fer tot això, vaig pensar en construir una gema nova a partir del meu codi escrit al primer puzzle. Segons la meva investigació, no només havia de definir a un arxiu les funcions que voldré utilitzar, com la de display, sinó que ha d'haver un arxiu de text nomenat novagema.gemspec que defineix les especificacions de la nova gema.

Però al veure de què estaria formada la meva gema, vaig donar-me compte de que el que volia reduir a una sola línia de codi amb la nova biblioteca, serien només 2 línies utilitzant la gema tan completa que vaig trobar a github per per el puzzle 1.

Demostració del codi a la nova gema puzzle1.rb:

(Tot i posar Python als blocs de codi, son en Ruby, només he escollit aquest llenguatge per a que es vegi millor)

```
Python
require 'i2c/drivers/lcd'

class Puzzle1
	def lcd_display(lcd_address, data)
			display =
I2C::Drivers::LCD::Display.new('/dev/i2c-1',lcd_address, cols = 20, rows =
4)
			display.clear
			display.text(data,0)
	end
end
```

De totes maneres, per completar l'activitat i practicar més la programación en aquest llenguatge, vaig intentar utilitzar aquesta gema. Quan la intentava utilitzar a un arxiu diferent amb la funció construïda, a la terminal apareixen uns missatges:

- No trobava la funció lcd_display, és a dir, deia que no estava definida.
- Seguidament, després d'aquest problema, sortien unes frases fent referencia a una altra gema, no utilitzada al codi executat, era gobject-introspection, i mencionava una funció nomenada invoke. Vaig intentar estudiar la gema i vaig pensar que potser el problema era la mala construcció de la meva puzzle1.rb.

L'últim pas d'aquest repte és escriure el codi per fer la finestra.

Després d'investigar els propis exemples proposats a la gema, vaig trobar els diferents components que hauria de tenir la finestra en qüestió. I per formular el mòdul display amb la quantitat de columnes i files i l'adreça de la LCD a la Raspberry, tal i com vaig aprendre a fer al puzzle anterior.

Vaig haver de provar moltes vegades fins a adonar-me que la finestra només pot donar lloc a un widget, en aquest cas jo he escollit una "caixa", que englobarà els demés components. I així també vaig trobar el widget `entry` on s'escriu el text i el puc exportar fins a la LCD.

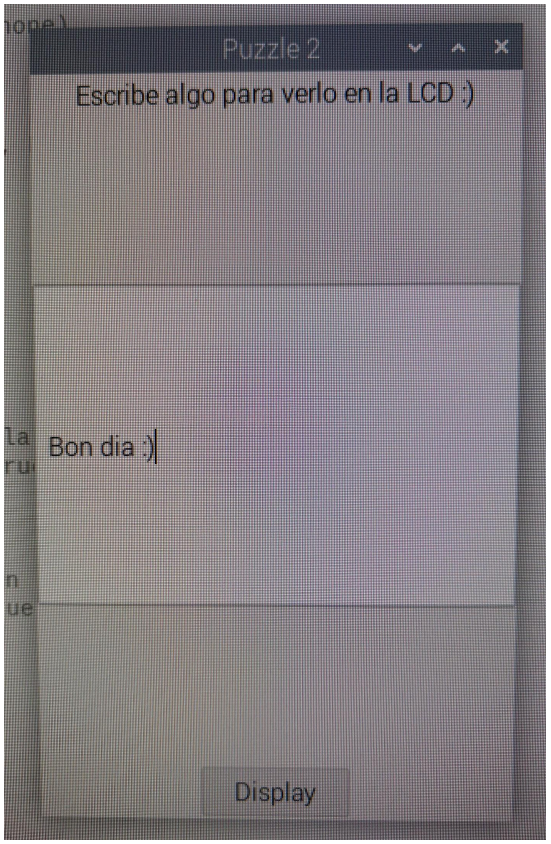
Python

```
require 'i2c/drivers/lcd'
require 'gtk3'

app = Gtk::Application.new("org.gtk.example", :flags_none)
display = I2C::Drivers::LCD::Display.new('/dev/i2c-1', 0x27, cols=20,
rows=4)

app.signal_connect "activate" do |application|
  window = Gtk::ApplicationWindow.new(application)
  window.set_title("Puzzle 2")
  window.set_default_size(300, 50)

  box = Gtk::Box.new(:vertical, 100)
  window.add(vbox)

  titol = Gtk::Label.new("Escribe algo para verlo en la LCD :)")
  box.pack_start(titol, :expand => false, :fill => true, :padding => 2)

  entry = Gtk::Entry.new
  entry.set_size_request(200, 200)
  entry.placeholder_text = "Escribe aquí"
  box.pack_start(entry, :expand => true, :fill => true, :padding => 2)

  button_box = Gtk::ButtonBox.new(:horizontal)
  button = Gtk::Button.new(label: "Display")
  button_box.add(button)
  button.signal_connect "clicked" do
       puts
           # Enviem el missatge a la LCD
              display.clear
              display.text(entry.text, 0)
           # Per estètica, quan es fa click al botó display, es borra el
           contingut escrit anteriorment
              entry.set_text("")
  end
  box.pack_start(button_box, :expand =>false, :fill => true, :padding => 2)
```

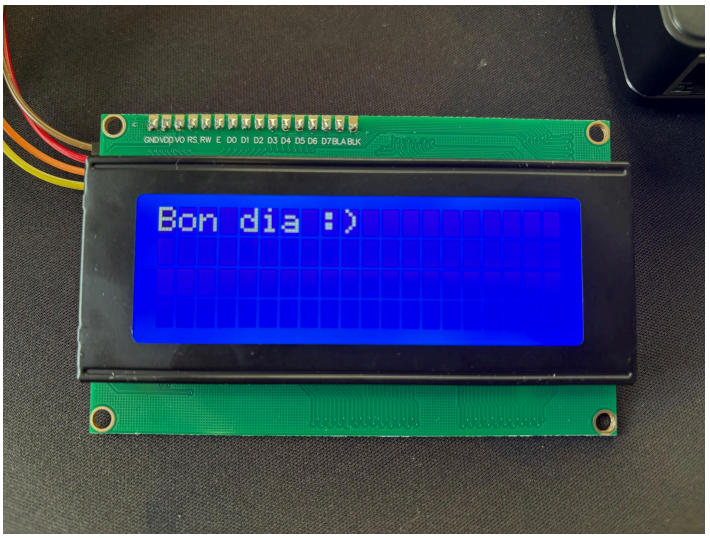
```
  window.show_all

  window.signal_connect("destroy") do
       window.destroy
  end

end

# Gtk::Application#run need C style argv ([prog, arg1, arg2, ...,argn]).
# The ARGV ruby variable only contains the arguments ([arg1, arg2,
...,argb])
# and not the program name. We have to add it explicitly.

puts app.run([$0] + ARGV)
```

Si el codi ha acabat sent tant llarg, és perquè volia fer la finestra suficientment funcionable i pensant en que fos comprensible i fàcil d'utilitzar per una persona externa al projecte o a la mateixa LCD.

La finestra es veu tal que:



I amb el missatge escollit i escrit en aquesta mateixa, el resultat després de fer clic al botó és el següent:



Cal afegir que cada vegada que es fa click al botó Display, es perd el missatge a la finestra, i fins que no s'ha escrit un altre missatge i es torna a fer click, no es borra el que hi ha escrit a la LCD.