Huankang Xu
PBE Telematica

Memoria 2n Puzzle

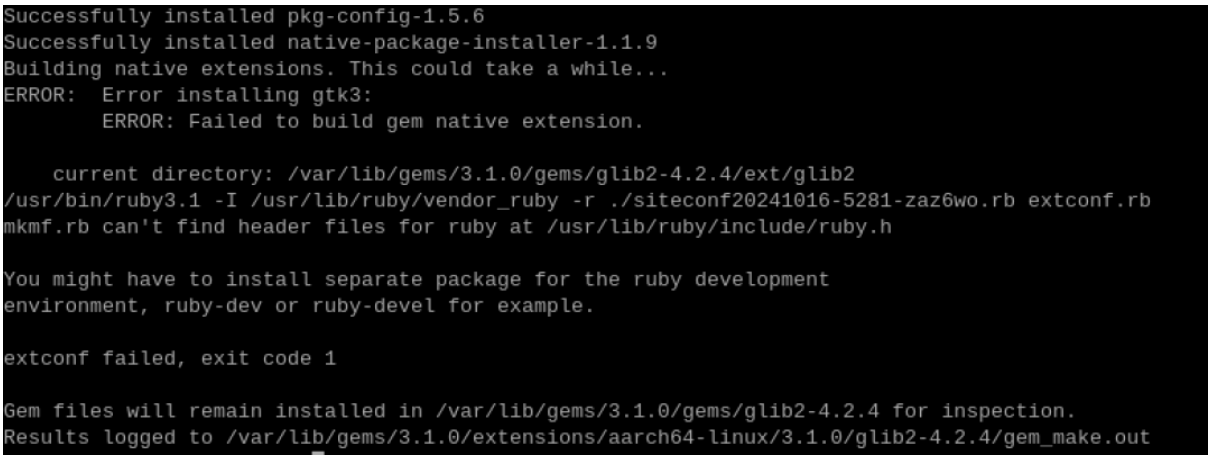
Para poder crear una ventana de texto para la RP, necesitamos instalar una biblioteca que nos permita mostrar de forma gráfica, en nuestro caso la GTK3.

1-Instalación de biblioteca grafica GTK3

- Para Ruby los pasos para la instalación de la biblioteca es muy sencilla, solo tenemos que ir al terminar de nuestro sistema OS y introducir el siguiente comando:

```
Unset
sudo apt install build-essential libgtk3
gem install gtk3
```

Que en mi caso me dio un error por falta instalar el paquete para compilar extensiones nativos:



Y claramente la solución es instalar dicho paquete, con el siguiente comando;
sudo apt-get install ruby-dev

```
Unset
sudo apt-get install ruby-dev
```

Huankang Xu
PBE Telematica

2- Librería del puzzle 1

- En esta librería contiene las funciones necesarias para el correcto funcionamiento de la lcd en el puzzle 2. Principalmente solo necesitamos 5 líneas de código que son las principales del puzzle 1:

    - La connexió entre la placa lcd amb el Raspberry
    - Hacer un clear
    - Que la línea de texto termine con “/n”
    - Para que lea el texto multiline
    - Haga el print correcto y que muestre por pantalla

- He agregado un controlador ya que al principio tenia un error al inicializar la variable display de la placa lcd.
- Volia posar aquesta llibreria com a funció a la llibreria propia de lcd en el meu cas “i2c/drivers/lcd” , pero sigo teniendo el problema que no me reconoce la función cuando la pongo dentro de la librería. A si que de momento he creado una libreria a parte del puzzle 1 para que pueda avanzar.

Python

```
# biblioteca de lcd_2004
require 'i2c/drivers/lcd'

class LCDController
  def initialize
    @display = I2C::Drivers::LCD::Display.new('/dev/i2c-1', 0x27, rows = 20,
cols = 4)
  end

  def printLCD(texto)
    if @display.nil?
      puts "El objeto display no está inicializado"
      return
    end

    @display.clear
    lineas = texto.split("\n")
    lineas.each_with_index do |linea, index|
      @display.text(linea[0, 20], index) if index < 4  # Asegúrate de no
exceder el número de líneas
    end
  end
end
```

Huankang Xu
PBE Telematica

3. Código principal del puzzle 2

- En el código principal del puzzle 2 he seguido los siguientes pasos para realizarlo:
    - Agregar las librerías gtk3 y el Controlador_LCD, inicializando el primero para que esté preparado para crear elementos visuales.
    - Creación de la ventana : A parte de crear la ventana puse un título y el tamaño que se tendría que ajustar con el tamaño de la pantalla 20x4 de la lcd.
    - Creación del texto: Crea un espacio donde el usuario pueda editar, en nuestro caso escribir en la lcd. Para que los textos se muestren con el mismo tamaño (cantidad de pixeles) puse como fuente “Monospace 10” y que sobreescriba.
    - Creación y menstruación del botón: Es un botón que tiene como función hacer que pase el texto escrito en la ventana a la placa lcd.
    - Creación de un contenedor: Su función es organizar la ventana de widgets y configurarlos de la forma que deseamos, en este caso pusimos lo siguiente: widget para mostrar el texto **textview** y el **button**, **expand** permite expandir y que ocupe espacio extra, **fill** hace expandir el widget hasta **padding,** en este caso 5.
    - Agregamos el contenedor a la ventana **window** y muestreamos.
    - **Gtk.main** inicia el bucle de funcionamiento de la interfaz gráfica hasta el usuario cierre el interfaz.

Python

```
require 'gtk3'
require_relative 'Controlador_LCD'
Gtk.init


#crear la ventana
window = Gtk::Window.new
window.set_title('Display LCD')
window.set_default_size(235, 120)  #falta per determinar

#area de texto multilinea
textview = Gtk::TextView.new
font_desc = Pango::FontDescription.new("Monospace 10")
textview.override_font(font_desc)

#es crea un boton
button = Gtk::Button.new(label: 'Display')
lcd = LCDController.new

#muestra si el boton es presionado
button.signal_connect('clicked') do
    texto = textview.buffer.text.chomp
```

```
    lcd.printLCD(texto)
end

#contenedor que organiza widgets en una disposición vertical o horizontal
box = Gtk::Box.new(:vertical, 5)
box.pack_start(textview, expand: true, fill:true, padding: 5)
box.pack_start(button, expand: false, fill: true, padding: 5)

window.add(box)
window.show_all

Gtk.main
```