Memoria Puzzle 1

Bruno Rivera Montaño Damian

1. Configuración de la Raspberrypi 3

Instalación del sistema operativo directamente desde la página oficial y si instalador Uso remoto a partir de los programas MobaXTerm (por ordenador) por terminal , VNC Viewer (por ordenador y móvil con interfaz gráfica).

2. Configuración de la librería para el lcd

comando para instalar la librería :
sudo gem install i2c_lcd
Inicialización por código del display :
display = I2C::Drivers::LCD::Display.new('/dev/i2c-1', 0x27, rows=20, cols=4)
Funciones proporcionadas por la librería :
display.text(texto para enviar , final donde escribir)
display.clear

3. Pines

GND-> PIN 6 Ground.
Vcc -> PIN 4 5V power.
SDA -> PIN 3 GPIO 2 (SDA).
SCL -> PIN 5 GPIO 3 (SCL).

4 Problemas al usar la librería

La librería permite el uso de enviar un string completo de manera instantánea pero no la posibilidad de escribir letra por letra aquello que queramos visualizar por el display por lo cual la posibilidad de manipular cada píxel por separado no estaba predeterminada.

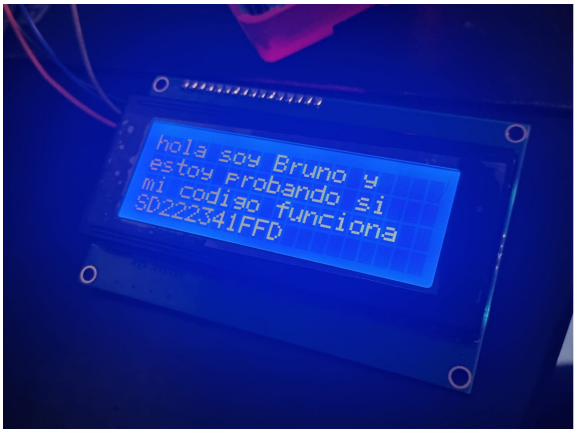
5 Soluciones y partes extra

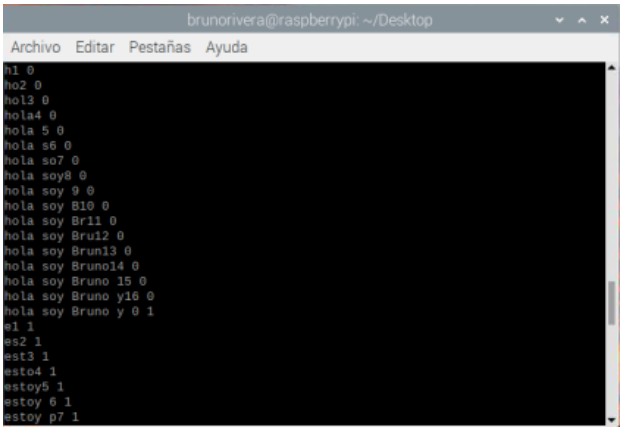
A partir de estas limitaciones. He creado un algoritmo que filtra el texto de manera que se pueda visualizar letra por letra y manipular individualmente cada píxel a partir de unos pequeños ajustes. En primer lugar convierto el texto adquirido en un vector el cual iré enviando al lcd de manera fraccionada a partir de strings que irán completando la palabra para poder obtener el efecto visual deseado además de su manipulación de tiempo de aparición y un pequeño filtro para tener menos posibilidad de que las palabras se corten.

6. Codigo

```ruby
require 'i2c/drivers/lcd'

class Rfid

  def initialize(display)
    # atributos
    @display = display

  end

  def read_uid(texto) # el texto es aquel que se adquiere por terminal
  end

end

#inicia el main

if __FILE__ == $0

  display = I2C::Drivers::LCD::Display.new('/dev/i2c-1', 0x27, rows=20, cols=4)
  rf = Rfid.new(display)
  rf.read_uid("Introduce un texto")
  text = gets.chomp
  rf.read_uid(text)

end
```

```ruby
def read_uid(texto) # el texto es aquel que se adquiere por terminal

  def interval seconds #Esta funcion se encarga de el delay de cada letra
    loop do
      yield
      sleep seconds
      break
    end
  end

  @display.clear
  vector = texto.chars # convierto el texto en un vector que contenga cada caracter
  puts texto.length.to_s
  i=0 # posicion de la letra en cada fila (20 por cada)
  j=0 # columna de el lcd

    v = ""
  vector.each do |element|  # itero cada letra para poder crear strings parciales de la palabra
    i+=1
    interval 0.1 do
    v += element # añado cada letra del vector texto
    print v
    @display.text(v,j) # envio cada string parcial
    #
    if i>17 && element == " " # filtro para no cortar las palabras
      i == 20                 # detecta si hay un espacio en los ultimos posibles 3 digitos
    end
    #
  end
      if i==20                # si he llenado una fila, paso a la siguiente
        i=0
        j+=1
        v = ""
      end
      if j==4                 # si estoy en la ultima fila, elimino todo y
        j=0                   # vuelvo a empezar en la primera fila
        @display.clear
      end
      puts i.to_s + " " + j.to_s
  end

end
```





Como se puede observar por terminal, lo que realmente ocurre es que se está enviando strings parciales con el texto, pero a vista del usuario solo se puede percibir como aparece cada letra de manera independiente a las otras.