

PySimpleGUI con Raspberry PI

Ramiro González / Leonel Mandarino / Sofía Martin Facultad de
Informática - UNLP

Junio 2019



Contenido

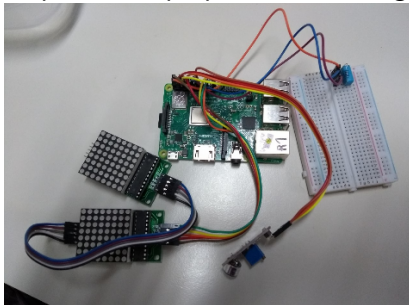
1 Hardware

- Raspberry
- Componentes a utilizar

2 El trabajo propuesto

- La matriz de LED
- Sensor Temperatura y Humedad DHT12
- El micrófono

Arquitectura propuesta a investigar y programar



Repositorio del trabajo realizado:

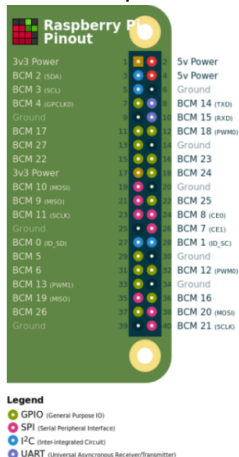
<https://github.com/Skydler/sensores-RPI>

Características

- Hardware libre
- Instalación de Debian
- Múltiple usos por su tamaño

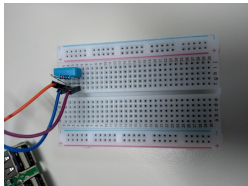
¿Cómo conecto los restantes dispositivos?

Se debe investigar los puertos disponibles: <https://pinout.xyz/>

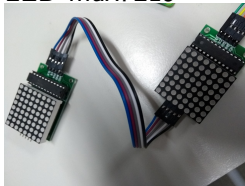


Recordemos: ¿qué componentes vamos a usar?

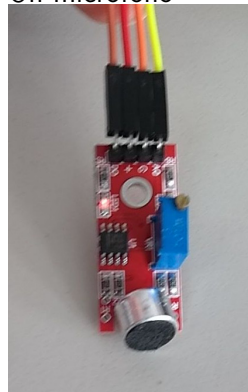
Un sensor de
temperatura y
humedad



Dos matrices de
LED max7219



Un micrófono



¿Qué vamos a hacer con estos componentes?

- Vamos a generar una aplicación que registre en archivos los datos de temperatura y humedad en distintas oficinas de la facultad.
- Los datos se mostrarán en las matrices de LEDs al producir un sonido tal como un aplauso.
- Los datos se almacenarán en archivos con formato JSON y serán utilizados en la configuración de la aplicación central a realizar. En este caso, una aplicación educativa para armar y resolver sopas de letras.
 - De acuerdo a los valores obtenidos se configura el *look and feel* de la aplicación.

¿Cómo los utilizamos?

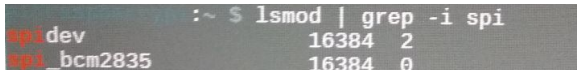
- Para conectar estos componentes no nos alcanza con conocer el esquema de la Raspberry, también tenemos que tener información técnica sobre cada uno de ellos.
- La información se encuentra en documentos que servirán de guía para conocer la conexión entre los pines del componente y los de la placa.
- Algunos componentes necesitan conectarse a través de una *protoboard* utilizando cables *dupont*.

PySimpleGUI con RaspBerry PI

Comandos útiles

Una vez que conectados los componentes necesitamos identificar el puerto en que está conectado cada uno, el driver relacionado.

```
lsmod | grep -i modulo_dispositivo
```



```
:~ $ lsmod | grep -i spi  
spidev                16384  2  
spi_bcm2835           16384  0
```

Comandos útiles

Para ver los dispositivos conectados

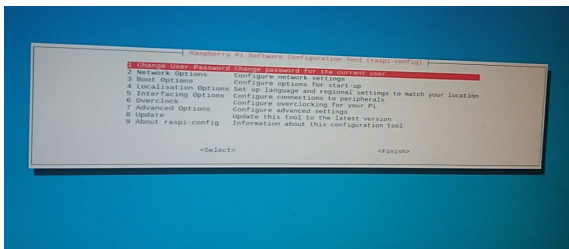
```
ls -l /dev/spi*
```

```
~ $ ls -l /dev/spi*  
crw-rw---- 1 root spi 153, 0 jun 11 15:15 /dev/spidev0.0  
crw-rw---- 1 root spi 153, 1 jun 11 15:15 /dev/spidev0.1
```

Comandos útiles

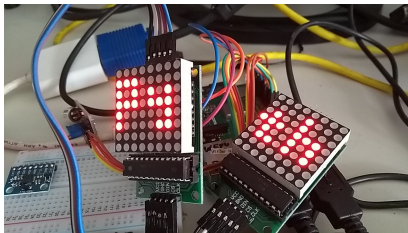
Habilitar componentes

```
# raspi-config
```

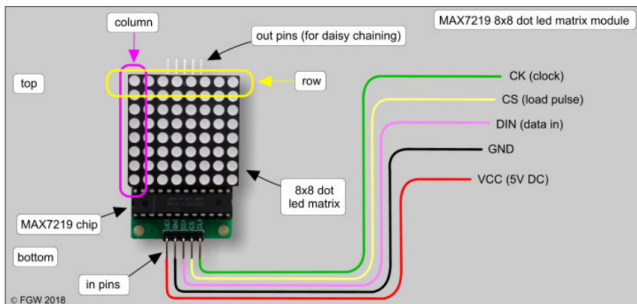


Matriz max7219

- Este componente es el más complejo de analizar.
- Hay que explorar el datasheet.
- En nuestro caso, se utilizarán dos matrices conectadas para mostrar contenido sincronizado en ambas.



Esquema



<https://luma-led-matrix.readthedocs.io/en/latest/>

Configuración en el entorno

- Antes de comenzar se debe agregar grupos *spi* y *gpio*, poner el usuario *pi* a ambos grupos

```
usermod -a -G spi,gpio pi
```

¿Cómo programamos en Python?

Vamos a utilizar dos librerías:

- Luma core: permite dibujar/escribir en pantallas pequeñas.
Provee un conjunto de módulos con funcionalidades específicas
- Luma Led Matrix: permite conectar con matrices

```
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.virtual import viewport
from luma.core.legacy import text, show_message
from luma.core.legacy.font import proportional, CP437_FONT,
    TINY_FONT, SINCLAIR_FONT, LCD_FONT
```


¿Cómo accedemos a la matriz de LED?

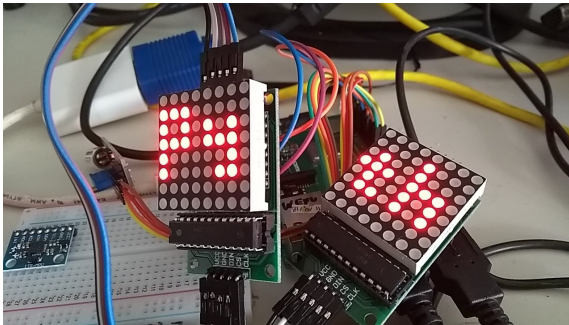
Nosotros definimos una clase Matriz:

```
class Matriz:
    def __init__(self, numero_matrices=1, orientacion=0,
                  rotacion=0, ancho=8, alto=8):
        self.font = [CP437_FONT, TINY_FONT, SINCLAIR_FONT,
                     LCD_FONT]
        self.serial = spi(port=0, device=0, gpio=noop())
        self.device = max7219(self.serial, width=ancho, height=
                               alto, cascaded=numero_matrices, rotate=rotacion)

    def mostrar_mensaje(self, msg, delay=0.1, font=1):
        show_message(self.device, msg, fill="white",
                     font=proportional(self.font[font]),
                     scroll_delay=delay)

matriz = Matriz(numero_matrices=2, ancho=16)
matriz.mostrar_mensaje("Python", delay=0.3)
```

Así se ve



Pasos para utilizar

- Inicializar la matriz: identificar el puerto

```
serial = spi(port=0, device=0, gpio=noop())
```

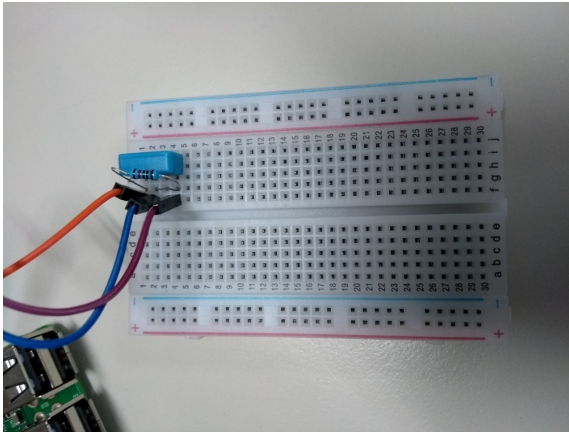
- Crear un objeto matriz

```
device = max7219(serial, width, height, cascaded, rotate)
```

- Mostrar un mensaje

```
show_message(device, msg, font,)max7219(serial, width,  
height, cascaded, rotate)
```

Sensor Temperatura y Humedad DHT12



Información técnica

- El uso de este sensor es más simple.
- Debemos explorar el datasheet.
- Para programar en Python vamos a utilizar la librería: `Adafruit_DHT`
- ¿Cómo tomamos los datos?
 - A través de `Adafruit_DHT.read_retry(self._sensor, self._data_pin)`.

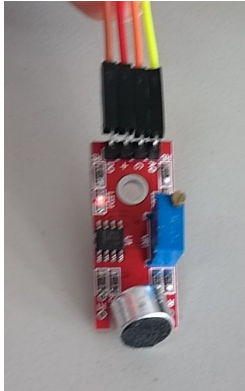
Nosotros definimos la clase Temperatura:

```
import Adafruit_DHT
class Temperatura:
    __def __init__(self, pin=17, sensor=Adafruit_DHT.DHT11):
    __    # Usamos el DHT11 que es compatible con el DHT12
    __self._sensor = sensor
    __self._data_pin = pin

    __def datos_sensor(self):
    __    humedad, temperatura = Adafruit_DHT.read_retry(self.
    __        _sensor, self._data_pin)
    __    return {'temperatura': temperatura, 'humedad': humedad}
```

```
temp = Temperatura()
datos = temp.datos_sensor()
print('Temperatura = {0:0.1f°}C  Humedad = {1:0.1f} %'.format
      (datos['temperatura'], datos['humedad']))
```

El micrófono



Información técnica

- Datasheet
- Para programar en Python vamos a utilizar la librería:
RPi.GPIO

Nosotros definimos la clase Sonido:

```
import RPi.GPIO as GPIO

class Sonido:

    def __init__(self, canal=22):
        self._canal = canal
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self._canal, GPIO.IN)
        # Desactivo las warnings por tener más de un circuito en
        la GPIO
        GPIO.setwarnings(False)
        GPIO.add_event_detect(self._canal, GPIO.RISING)

    def evento_detectado(self, funcion):
        if GPIO.event_detected(self._canal):
            funcion()
```

Lo usamos así:

```
def test():  
    print('Sonido detectado!')  
  
sonido = Sonido()  
while True:  
    time.sleep(0.0001)  
    sonido.evento_detectado(test)
```