

LHD: Memoria Práctica

RASPBERRY

Alejandro González

INDICE

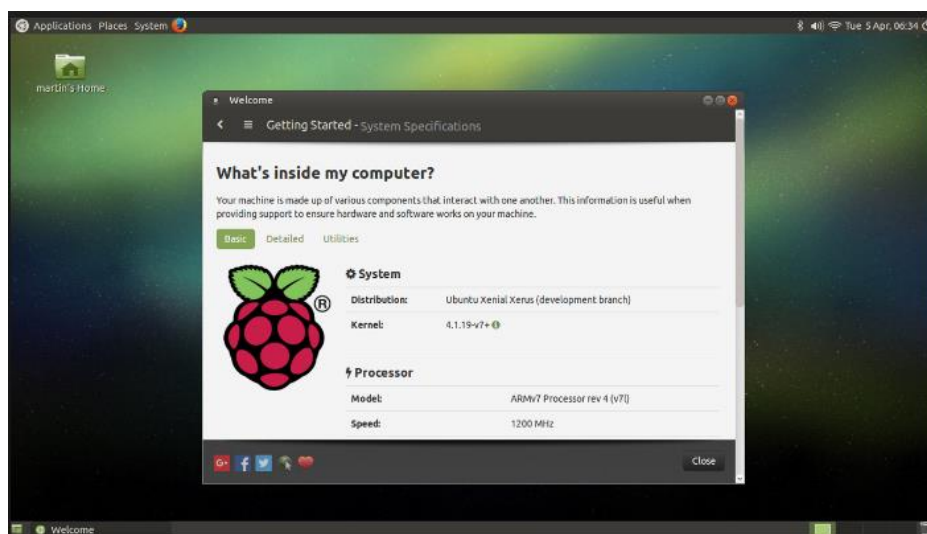
- Objetivos pág. 3
- Actividades Realizadas pág. 7
- Opinión personal pág. 18
- Conclusiones pág. 19

OBJETIVOS

Preparar la plataforma Raspberry Pi para que puedan cargarse diferentes versiones de Sistema Operativo.

- Arrancar y comprobar el funcionamiento de la placa Raspberry Pi
- Desarrollar ejemplos de utilización de los pines de expansión GPIOs
- Instalar un Servidor WEB que pueda ejecutar código PYTHON

En este tercer bloque de práctica se busca aprender a utilizar una de las placas más conocida del mercado, como el pequeño pc o el pc de 40\$, funciona como un mini pc, a este placa podemos cargarle un sistema operativo en su tarjeta SD (Raspbian, Windows 10,...), una vez que el sistema operativo está instalado, depositamos la tarjeta en su ranura y arrancamos la placa con la tarjeta introducida, en algunos sistemas operativos cabe la posibilidad de instalar un entorno grafico lo que hace más fácil su utilización. (MATE es el que usaremos).



IMPORTANTE: Descargar la imagen de Coria:

→ <http://coria.dte.us.es/~bellido/> <https://10.1.15.78/~bellido>

→ Imagen a descargar:

- ubuntu-mate-16.04-desktop-armhf-raspberry-pi-resize.img

Para obtener mas información de Raspberry Pi:

→ <https://ubuntu-mate.org/raspberry-pi/>

- **Una vez preparada la tarjeta:**

- Conectar la SD a RaspberryPi (RPI)

- Desconectar Teclado, ratón y cable Ethernet del PC y conectar a RPI

- Conectar cable HDMI-DVI a RPI y Monitor.

- Conectar cable USB-Micro usb a RPI (microusb) y PC para alimentar la RPI

Configurar Ubuntu-Mate:

- Seguir los pasos en el arranque configurando idioma, teclado, localización, fecha y hora

- **IMPORTANTE: Guardar** información sobre usuario y password.

- Configuraremos la red manualmente:

→ Desactivar network-manager

→ Editar /etc/network/interfaces:

```
iface <nombre_InterfazDeRed> inet static
address 10.1.15.123 (Mi puesto)
netmask 255.255.252.0
gateway 10.1.15.78
dns-nameservers 8.8.8.8
```

→ Ejecutar sucesivamente: \$ sudo ifdown <nombre_InterfazDeRed>

- \$ sudo ifup <nombre_InterfazDeRed>

→ Probar conexión: \$ ping www.google.es



```
Aplicaciones Lugares Sistema
• alegonsan@alegonsanrasp: ~
Archivo Editar Ver Buscar Terminal Ayuda
alegonsan@alegonsanrasp:~$ ifconfig
enxb827eb00ebb8 Link encap:Ethernet HWaddr b8:27:eb:00:eb:b8
inet6 addr: fe80::cd68:3fb2:94c0:ed8d/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2190 errors:0 dropped:0 overruns:0 frame:0
TX packets:58 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:183523 (183.5 KB) TX bytes:10392 (10.3 KB)

lo        Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:104 errors:0 dropped:0 overruns:0 frame:0
TX packets:104 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:8132 (8.1 KB) TX bytes:8132 (8.1 KB)

wlan0     Link encap:Ethernet HWaddr b8:27:eb:55:be:ed
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:135 errors:0 dropped:135 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:46974 (46.0 KB) TX bytes:0
```

PASOS A SEGUIR IMPORTANTES

/dev/ son los discos que están montados en el equipo

/dev/sd+tab Para ver las unidades de almacenamiento

df para ver todas las unidades

umount para desmontar las dos imagenes

pskill -USR1 -n -x dd para ver el % que va la copia

dd bs=4M if=2016-09-23-raspbian-jessie.img of=/dev/sdd

para copiar a nivel de byte la imagen

Configurar ethernet

network connection - wired connection - ipv6 ignore - ipv4 manual

10.1.15.123 Ip - 255.255.252.0 Mask - 10.1.15.78 Gateway

Usuario y contraseña

Alegonsan- alexxit0

por ssh

ssh -X alegonsan@10.1.15.123

y python blink.py e inicia la ejecución

sudo shutdown -h now

vamos a desactivar el network manager para hacerlo manual

con ifconfig lo copiamos . el interfaz es enx827eb00ebb8 este cambio lo hace ubuntu mate

//comando dmesg |grep eth0 saca los mensajes internos y busca con eth0

como gedit no lo tiene..

sudo pluma /etc/network/interfaces

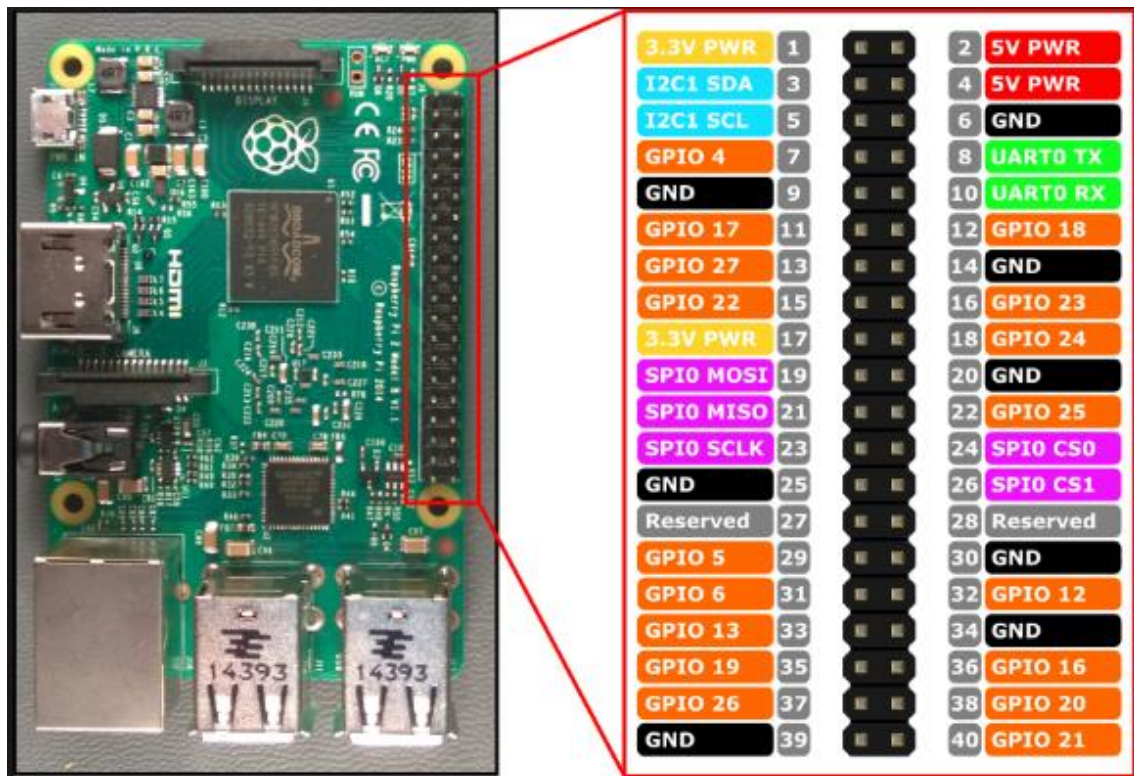
aqui configuramos a mano la red y luego ping

hay que instalar el flask → sudo pip install flask

Guardamos y reiniciamos la interface, ejecutamos sudo ifdown eth0 y sudo ifup eth0, una vez realizado eso podemos navegar por nuestra placa porque ya tendrá conexión con internet.

Una vez configurado todo empezamos nuestros ejercicios.

ACTIVIDAD 1.- En esta actividad vamos a ver el manejo de GPIOs desde línea de comandos: cogemos un led y lo conectamos al puerto 17 de nuestra placa.



Ahora la manera que vamos acceder a los GPIO es como si fuesen directorios. En principio no tenemos ningún pin accesible, ni de entrada ni de salida. Tenemos que crearlo nosotros mismo. Queremos tener acceso al GPIO 17, así que introducimos el siguiente comando.

```
echo 17 > /sys/class/gpio/export
```

Tras esto el sistema ha creado un archivo con una estructura GPIO que corresponde al número 17. A continuación, tenemos que informar a la Raspberry Pi de si el pin va a ser de salida o, de entrada. Como lo que queremos es encender un LED, GPIO será de salida, para ello ejecutamos el siguiente comando

```
echo out > /sys/class/gpio/gpio17/direction
```

Con esto el sistema sabe que es de salida. Ahora tenemos que darle valores 0 o 1 para que muestre el LED encendido o apagado.

Para encender: `echo 1 > /sys/class/gpio/gpio17/value`

Para apagar: `echo 0 > /sys/class/gpio/gpio17/value`

Una vez acabada con las pruebas borramos

```
echo 17 > /sys/class/gpio/unexport
```

ACTIVIDAD 2

En esta segunda actividad vamos a ver ejemplos sencillos realizados con PYTHON, el primer ejemplo es controlar un led. Usaremos el código .py de las anteriores que usábamos para los leds, lo primero que tenemos es importar las librerías GPIO, en MATE. Tenemos que declarar los pines GPIO17 y GPIO27 como salida en nuestro código, e implementamos el programa BLINK que es el que se encarga de encender y apagar periódicamente

Lo creamos y lo ejecutamos con el siguiente comando

```
sudo nano blink.py
```

```
1 def blink():
2     print "Ejecucion iniciada..."
3     iteracion = 0
4     while iteracion < 30: ## Segundos que durara la funcion
5         GPIO.output(17, True) ## Enciendo el 17
6         GPIO.output(27, False) ## Apago el 27
7         time.sleep(1) ## Esperamos 1 segundo
8         GPIO.output(17, False) ## Apago el 17
9         GPIO.output(27, True) ## Enciendo el 27
10        time.sleep(1) ## Esperamos 1 segundo
11        iteracion = iteracion + 2 ## Sumo 2 porque he hecho dos parpadeos
12    print "Ejecucion finalizada"
13    GPIO.cleanup() ## Hago una limpieza de los GPIO
```


ACTIVIDAD 3

Vamos a realizar el segundo método. SSH

Lo primero que hacemos en esta práctica es conectar solo la RaspBerri al cable de red, y como ya está todo configurado para que tenga red no hace falta conectar la pantalla. Accederemos mediante SSH a nuestra placa. Para ello nos vamos a otro pc que este en la misma red y en Python utilizamos este código

```
ssh -X pi@10.1.15.123 (Mi pc era 123)  
(Es posible que tengamos que descargar el ssh)
```

Ahora al conectarnos la placa nos pide la clave que teníamos configurada al principio y estamos dentro.

```
Sudo apt-get install Python-pip
```

Y a continuación instalamos FLASK

```
Sudo pip install flask
```

Ahora nos creamos un archivo que lo llamaremos hello.py y dentro le introducimos el siguiente código

```
from flask import Flask  
app = Flask(__name__)  
@app.route("/")  
def hello():  
    return "Bienvenido Alejandro!"  
if __name__ == "__main__":  
    app.run(host='0.0.0.0', port=80, debug=True)
```

Ahora nos vamos y ejecutamos el hello.py, a continuación, nos vamos al menú y buscamos una aplicación navegadora y escribimos la ip que tiene asignada nuestra placa 10.1.15.123 , y nos sale la web mostrando **Bienvenido Alejandro.**

ACTIVIDAD 4:

En el proyecto webLamp emplearemos LEDS. Cambiaremos los nombres:

- “coffee maker” por LED1
- “Lamp” por LED2

```
import RPi.GPIO as GPIO
from flask import Flask, render_template, request
app = Flask(__name__)

GPIO.setmode(GPIO.BCM)

# Create a dictionary called pins to store the pin number, name
pins = {
    17 : {'name' : 'led1', 'state' : GPIO.LOW},
    22 : {'name' : 'led2', 'state' : GPIO.LOW}
}

# Set each pin as an output and make it low:
for pin in pins:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, GPIO.LOW)

@app.route("/")
def main():
    # For each pin, read the pin state and store it in the pins
    for pin in pins:
        pins[pin]['state'] = GPIO.input(pin)
    # Put the pin dictionary into the template data dictionary:
    templateData = {
        'pins' : pins
    }
    # Pass the template data into the template main.html and re
    return render_template('main.html', **templateData)

# The function below is executed when someone requests a URL w
@app.route("/<changePin>/<action>")
def action(changePin, action):
    # Convert the pin from the URL into an integer:
    changePin = int(changePin)
    # Get the device name for the pin being changed:
    deviceName = pins[changePin]['name']
    # If the action part of the URL is "on," execute the code i
    if action == "on":
```

Esta es la parte ½ del weblamp.py

```

if action == "on":
    # Set the pin high:
    GPIO.output(changePin, GPIO.HIGH)
    # Save the status message to be passed into the template:
    message = "Turned " + deviceName + " on."
if action == "off":
    GPIO.output(changePin, GPIO.LOW)
    message = "Turned " + deviceName + " off."
if action == "toggle":
    # Read the pin and set it to whatever it isn't (that is,
    GPIO.output(changePin, not GPIO.input(changePin))
    message = "Toggled " + deviceName + "."

# For each pin, read the pin state and store it in the pins dictionary
for pin in pins:
    pins[pin]['state'] = GPIO.input(pin)

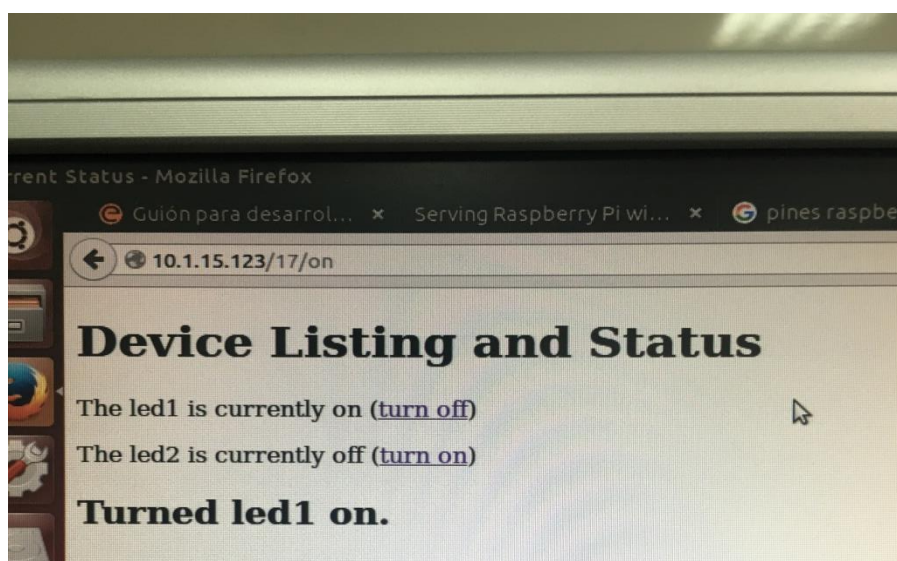
# Along with the pin dictionary, put the message into the template data
templateData = {
    'message' : message,
    'pins' : pins
}

return render_template('main.html', **templateData)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)

```

Parte 2/2 del código weblamp



Muestra de la pagina HTML

Código página HTML

```
<!DOCTYPE html>
<head>
  <title>Current Status</title>
</head>

<body>
  <h1>Device Listing and Status</h1>

  {% for pin in pins %}
  <p>The {{ pins[pin].name }}
  {% if pins[pin].state == true %}
    is currently on (<a href="/{{pin}}/off">turn off</a>)
  {% else %}
    is currently off (<a href="/{{pin}}/on">turn on</a>)
  {% endif %}
</p>
  {% endfor %}

  {% if message %}
  <h2>{{ message }}</h2>
  {% endif %}

</body>
</html>
```

En esta práctica se trata de una vez montado el servidor web, poder aprovecharlo y mediante el código de una web sencilla poder aprovechar y mezclar la potencia de las 3 partes de manera que mediante Raspberry nos conectamos a arduino, este manda a Raspberry por puerto serie en python la información que deseamos y nosotros la actualizamos mediante el servidor web.

ACTIVIDAD 5:

Sensor de temperatura tmp36GZ:

- Calcularemos la temperatura y la transmitiremos vía serie: `Serial.println(temp)`

- Desde el servidor debo ser capaz de leer la Temperatura

Código .py para recoger la temperatura de Arduino

```
1  import RPi.GPIO as GPIO
2  import serial
3  from flask import Flask, render_template, request
4  app = Flask(__name__)
5
6  GPIO.setmode(GPIO.BCM)
7
8  # Create a dictionary called pins to store the pin number, name
9  pins = {
10     24 : {'name' : 'coffee maker', 'state' : GPIO.LOW},
11     25 : {'name' : 'lamp', 'state' : GPIO.LOW}
12 }
13 temperatura={
14     1 :{'temps':''}
15 }
16
17 # Set each pin as an output and make it low:
18 ser=serial.Serial('/dev/ttyUSB0',9600)
19 for pin in pins:
20     GPIO.setup(pin, GPIO.OUT)
21     GPIO.output(pin, GPIO.LOW)
22
23 @app.route("/")
24 def main():
25
26     # For each pin, read the pin state and store it in the pins di
27     temp=ser.readline()
28     print("temps=", temp)
29     for pin in pins:
30         pins[pin]['state'] = GPIO.input(pin)
31     # Put the pin dictionary into the template data dictionary:
32     templateData = {
33         'pins' :pins,
34         'temps':temp,
35     }
36     # Pass the template data into the template main.html and ret
37     return render_template('main.html', **templateData)
```

```

39 @app.route("/update")
40 def temp():
41     temp=ser.readline()
42     temperatura[1]['temps'] +=temp
43     print("temps=", temp)
44     for pin in pins:
45         pins[pin]['state'] = GPIO.input(pin)
46         # Put the pin dictionary into the template data dictionary:
47         templateData = {
48             'pins':pins,
49             'temps':temp,
50         }
51         return render_template('main.html',**templateData)
52
53 # The function below is executed when someone requests a URL with th
54 @app.route("/<changePin>/<action>")
55 def action(changePin, action):
56     # Convert the pin from the URL into an integer:
57     changePin = int(changePin)
58     # Get the device name for the pin being changed:
59     deviceName = pins[changePin]['name']
60     # If the action part of the URL is "on," execute the code indente
61     if action == "on":
62         # Set the pin high:
63         GPIO.output(changePin, GPIO.HIGH)
64         # Save the status message to be passed into the template:
65         message = "Turned " + deviceName + " on."
66     if action == "off":
67         GPIO.output(changePin, GPIO.LOW)
68         message = "Turned " + deviceName + " off."
69     if action == "toggle":
70         # Read the pin and set it to whatever it isn't (that is, togg
71         GPIO.output(changePin, not GPIO.input(changePin))
72         message = "Toggled " + deviceName + "."
73

```

```

75     for pin in pins:
76         pins[pin]['state'] = GPIO.input(pin)
77
78     # Along with the pin dictionary, put the message into the tem
79     templateData = {
80
81         'message' : message,
82         'temps' : temperatura,
83         'pins' : pins,
84     }
85
86     return render_template('main.html', **templateData)
87
88 if __name__ == "__main__":
89     app.run(host='0.0.0.0', port=80, debug=True)
90
91

```

CODIGO EN ARDUINO PARA LEER TEMPERATURA

```
sketch_nov19a $  
void setup() {  
    Serial.begin(9600);  
}  
void loop() {  
    int sensorValue=analogRead(A0);  
    float volt= (sensorValue/1020.0)*4.9;  
    float tempC= (volt-0.5)*100;  
    Serial.println(tempC);  
    delay(3000);  
}
```

CODIGO DE HTML

```
1  <!DOCTYPE html>  
2  <head>  
3    <title>Current Status</title>  
4  </head>  
5  
6  <body>  
7    <h1>Device Listing and Status</h1>  
8  
9    {% for pin in pins %}  
10   <p>The {{ pins[pin].name }}  
11   {% if pins[pin].state == true %}  
12     is currently on (<a href="/{{pin}}/off">turn off</a>)  
13   {% else %}  
14     is currently off (<a href="/{{pin}}/on">turn on</a>)  
15   {% endif %}  
16   </p>  
17   {% endfor %}  
18   <p> temperatura es: {{temps}} (<a href="/update">actualizar</a>)</p>  
19   <p> LED control(<a href="/led">toggle</a>)</p>  
20   {% if message %}  
21     <h2>{{ message }}</h2>  
22   {% endif %}  
23  
24 </body>  
25 </html>
```

En esta práctica además de usar los led y enviar esa información de una placa a otra, vamos a hacer un paso más. Mediante la comunicación de puerto serie vamos a transmitir de Arduino a RaspBerry la información que estamos generando a través del sensor de temperatura. Y esta a su vez mostrada en pantalla tal y como vemos.

ACTIVIDAD 6.-

Añadir al sensor de temperatura un Relé con bombilla:

- Estaremos esperando datos vía serie para encender o apagar LUZ

CODIGO ARDUINO PARA RELÉ

```
sketch_nov19a$  
int pinRele = 13;  
int sensorPin = 0;  
char val;  
  
void setup(){  
  pinMode(pinRele, OUTPUT);  
  Serial.begin(9600);  
}  
  
void loop(){  
  float sensorVal;  
  float temperatura;  
  if(Serial.available() > 0){  
    char c = Serial.read();  
    if(c == 'H'){  
      digitalWrite(pinRele, HIGH);  
    }else if(c == 'L'){  
      digitalWrite(pinRele, LOW);  
    }  
  }  
  sensorVal = analogRead(sensorPin);  
  temperatura = ((sensorVal/1024)*5 - .5)*100;  
  Serial.println(temperatura);  
  delay(3000);  
}
```

VISTA HTML CON RELÉ

Listado de dispositivos y sus estados

The LED1 is currently off ([Encender](#))

The RELE is currently off ([Encender](#))

The LED2 is currently off ([Encender](#))

La temperatura actual es: 29.59 °C ([Actualizar por favor](#))

CODIGO PYTHON CON RELÉ

```
import RPi.GPIO as GPIO
import serial

from flask import Flask, render_template, request

app = Flask(__name__)

GPIO.setmode(GPIO.BCM)
#Create a dictionary called pins to store the pin number, name, and pin state
pins = {
    17: {'name' : 'RELE', 'state' : GPIO.LOW},
    24: {'name' : 'LED1', 'state' : GPIO.LOW},
    25: {'name' : 'LED2', 'state' : GPIO.LOW}
}
ser = serial.Serial('/dev/ttyACM0', 9600)
# Set each pin as an output and make it low:
for pin in pins:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, GPIO.LOW)

@app.route("/")
def main():
    temp = ser.readline()
    # For each pin, read the pin state and store it in the pins dictionary:
    templateData = {
        'pins' : pins,
        'temps' : temp,
    }

    #Pass the template data into the template main.html and return it to the user
    return render_template('main.html', **templateData)

@app.route("/update")
def temp():
    temp = ser.readline()
    for pin in pins:
        pins[pin]['state'] = GPIO.input(pin)

    templateData = {
        'pins' : pins,
        'temps' : temp,
    }
    return render_template('main.html', **templateData)

# The function below is executed when someone requests a URL with the pin numbend action in it:
@app.route("/<changePin>/<action>")
```

```

deviceName = pins[changePin]['name']
if action == "on":
    if changePin == 17:
        ser.write('L')
        GPIO.output(changePin, GPIO.HIGH)
        message = "Cambiado" + deviceName + "Ya ha encendido"
    else:
        pass
if action == "off":
    if changePin == 17:
        ser.write('H')
        GPIO.output(changePin, GPIO.LOW)
        message = "Cambiado" + deviceName + "Ya ha sido apagada"
    else:
        pass
if action == "toogle":
    GPIO.output(changePin, not GPIO.input(changePin))
    message = "Toogle " + deviceName + "."
for pin in pins:
    pins[pin]['state'] = GPIO.input(pin)
#Along with the pin dictionary, put the message into the template data dictionary:
templateData = {
    'message': message,
    'pins': pins,
}
return render_template('main.html', **templateData)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)

```

En esta parte ultima de la práctica correspondiente a Raspberry, además de mostrar el estado de los LEDs y de la información del sensor de temperatura que tenemos en Arduino, además de eso, tenemos que poder encender/apagar una bombilla de 220v. Para ello usaremos un rele, para poder activarla y desactivarla con una señal de 5v. Para ello tenemos que incluir esas variables en nuestro código HTML y para ello previamente hemos tenido que crear la función en python, dándole su @approute correspondiente para que calcule esos valores y a través de template, se los entrega a HTML.

OPINION PERSONAL

Estas prácticas con RaspBerry me han gustado mucho más, es una placa que nunca había utilizado tampoco pero que me ha gustado muchísimo, a partir de conocerla me he decantado por ella para utilizarla para el proyecto fin de carrera, pienso que tiene muchísimo potencial y se le puede sacar mucho rendimiento de esta placa con tantas prestaciones como esta. Sobre todo tal y como hemos hecho en las practicas, componiendo un sistema junto a Arduino.

En general ha sido unas prácticas muy buenas porque hemos aprendido tanto a usarla como un servidor web tanto como un mini pc con Linux, y la potencia que tiene en si misma con las memorias y procesadores. Por ello es la practica que más me ha gustado de las tres.

CONCLUSIONES

La práctica ha quedado concluida, no con tantas dificultades como las de ZPUino por cuestiones técnicas si no por cuestiones de como explique en la anterior parte. Pienso que ha sido muy positivo e ilustrativo haber trabajado con estos tres sistemas, ya no solo por conocer lo que hay en el mundo exterior, saber usarlos y aprobar la asignatura, si no que estas prácticas nos pueden servir perfectamente para el fin de carrera o para futuros proyectos personales que queramos desarrollar tanto en casa como en el ámbito laboral.