

Laboratorio de Desarrollo de Hardware

# Memoria de prácticas

Raspberry PI

Ángel López Santiago

## Contenido

Objetivos .....	2
¿Qué es Raspberry Pi? .....	2
Ejercicio 1: Preparar Raspberry Pi .....	2
Ejercicio 2: Controlar GPIOs desde línea de comandos .....	2
Ejercicio 3: Accedes a raspberry desde SSH.....	3
Ejercicio 4: Controlar encendido de un LED a través de un servidor web .....	4

## Objetivos

Nuestro objetivo es preparar la plataforma Raspberry Pi para que puedan cargarse diferentes versiones de Sistema Operativo y comprobar su funcionamiento.

Desarrollaremos ejemplos de utilización de los pines de expansión GPIOs e instalaremos un Servidor WEB que pueda ejecutar código Python.

## ¿Qué es Raspberry Pi?

Raspberry Pi es un computador de placa reducida, computador de placa única o computador de placa simple (SBC) de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

## Ejercicio 1: Preparar Raspberry Pi

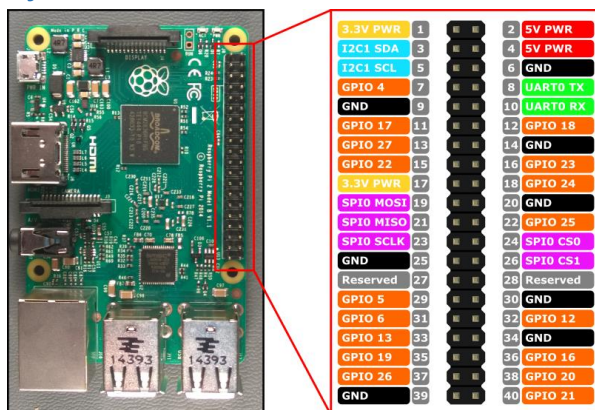
Para esta práctica utilizaremos la distribución Ubuntu Mate para nuestra placa Raspberry Pi.

Para ello descargaremos la versión más reciente de este SO desde la web de Raspberry.

Una vez descargada la imagen de Ubuntu Mate, lo instalaremos a bajo nivel en nuestra tarjeta SD.

Una vez instalada la imagen en nuestra SD, la insertaremos en nuestra placa y la encenderemos.

## Ejercicio 2: Controlar GPIOs desde línea de comandos



En este ejercicio controlaremos el encendido de un LED conectado al pin GPIO17 de nuestra raspberry mediante un programa en Python.

Disponemos de este código Python:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT) ## GPIO 17 como salida
GPIO.setup(27, GPIO.OUT) ## GPIO 27 como salida
def blink():
    print "Ejecucion iniciada..."
    iteracion = 0
    while iteracion < 30: ## Segundos que durara la funcion
        GPIO.output(17, True) ## Enciendo el 17
        GPIO.output(27, False) ## Apago el 27
        time.sleep(1) ## Esperamos 1 segundo
        GPIO.output(17, False) ## Apago el 17
        GPIO.output(27, True) ## Enciendo el 27
        time.sleep(1) ## Esperamos 1 segundo
        iteracion = iteracion + 2 ## Sumo 2 porque he hecho
dos parpadeos
    print "Ejecucion finalizada"
    GPIO.cleanup() ## Hago una limpieza de los GPIO
blink() ## Hago la llamada a la funcion blink
```

Una vez creado nuestro programa solo tenemos que ejecutarlo en nuestra raspberry.

### Ejercicio 3: Accedes a raspberry desde SSH.

Para este ejercicio solo necesitaremos que nuestra placa esté conectada mediante ethernet.

Desde nuestro PC accederemos al terminal y ejecutaremos el siguiente comando:

```
$ ssh -X <usuario>@<IPadressRasp>
```

## Ejercicio 4: Controlar encendido de un LED a través de un servidor web

En este ejercicio crearemos una web para controlar el encendido y apagado de un LED conectado a un pin del GPIO de nuestra raspberry.

Primero crearemos nuestro programa en Python:

```
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, and
pin state:

pins = {
    24 : {'name' : 'led1', 'state' : GPIO.LOW},
    25 : {'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
    dictionary:
    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 return it
    to the user
    return render_template('main.html', **templateData)
```

# The function below is executed when someone requests a URL with the pin number and action in it:

```
@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 indented below:
```

```
    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, toggle it):
```

```
        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 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)
```

Después, crearemos nuestro fichero main.html ubicado en la carpeta templates en el mismo directorio que nuestro programa Python:

```
<!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>
```

Una vez creado nuestros ficheros, ejecutaremos nuestro programa Python y abriremos nuestro navegador web insertando la IP de nuestra placa.