Weather Bot

En este proyecto crearemos un programa que nos mandará notificaciones a nuestro teléfono móvil con información sobre la previsión del tiempo a lo largo del día. Al tratarse de un proyecto algo más avanzado, necesitaremos ciertos conocimientos mínimos sobre programación, uso de APIs y el manejo de un entorno Linux.

Preparando el proyecto

Herramientas a utilizar

Para realizar este proyecto emplearemos las APIs de PushOver (para mandar notificaciones a nuestro móvil mediante su aplicación) y OpenWeather (para obtener información sobre el tiempo).

Obteniendo las claves

En primer lugar, necesitaremos obtener acceso a las APIs mencionadas en el apartado anterior:

- Para obtener las claves de PushOver simplemente deberemos acceder a https://pushover.net/, crearnos una cuenta personal y registrar una aplicación desde https://pushover.net/apps/build. Una vez hecho esto podremos ver tanto nuestra USER KEY como nuestro API TOKEN.
- Para las de OpenWeather deberemos acudir a https://openweathermap.org/, crearnos una cuenta de usuario y, a través de https://home.openweathermap.org/api_keys, registrar una API KEY.

Es muy importante destacar que estas claves son privadas y su uso está sujeto a las condiciones de cada servicio.

Instalando el Sistema Operativo en la Raspberry

Como en este proyecto sólo queremos emplear la placa para ejecutar el programa que desarrollemos, emplearemos una imagen lo más ligera posible y que apenas consuma recursos: la versión Lite de Raspberry OS.
Para instalarla abriremos Raspberry Pi Imager, seleccionaremos Raspberry Pi OS (other) > Raspberry Pi OS Lite (32-bit), elegimos como medio de almacenamiento la SD de nuestra Raspberry y clicamos sobre WRITE:

Una vez terminada la escritura, extraemos la tarjeta SD y la insertamos en la placa. Tras conectar el resto de periféricos conectamos la alimentación y esperamos a que se encienda. Como podremos observar, se trata de una interfaz basada en linea de comandos, por lo que su uso puede hacerse algo más pesado.

Mandando notificaciones

Para empezar, probaremos a enviar desde la Raspberry a nuestro teléfono una notificación con un mensaje que nosotros especifiquemos. Para ello deberemos crear una carpeta para nuestro proyecto, a la que llamaremos “weatherbot”:

```
cd weatherbot
```

Y nos movemos a dicha carpeta:

```
cd weatherbot
```

Una vez ahí crearemos el archivo sobre el que escribiremos nuestro programa en Python:

```
sudo nano weather_bot.py
```

Se abrirá una nueva vista en la cual podremos escribir el código de nuestro programa:

```
#!/usr/bin/env python3
import requests

def send_notification(message_string):
    user = 'XXXXXXX'    # clave de user de PushOver
    token = 'XXXXXXX'   # clave de aplicación de PushOver
    json_header = {'token': token, 'user': user, 'message':
message_string, 'title': 'Título'}
    requests.post('https://api.pushover.net/1/messages.json',
json=json_header)


def main():
    message_string = "Hello world"
    send_notification(message_string)


if __name__ == '__main__':
    main()
```

Para comprobar que funciona correctamente, guardamos el archivo y lo ejecutamos mediante:

```
python3 weather_bot.py
```

Hecho esto deberíamos haber recibido una notificación en nuestro teléfono con el texto especificado.

Completando el programa

Una vez hemos comprobado que las notificaciones funcionan correctamente, volveremos a abrir el archivo mediante

```
sudo nano weather_bot.py
```

Y lo editamos de forma que quede como lo siguiente:

```
#!/usr/bin/env python3
import requests
import datetime

def get_weather_info():
    rain = False
    avg_temperature = 0
    rain_string = ''
    ow_key = 'XXXXXXX'      # clave de la api de OpenWeather
    lat = '41.683712'   # latitud
    lon = '-0.888141'   # longitud
    ow_request =
'https://api.openweathermap.org/data/2.5/forecast?units=metric&cnt=8&lat=' + \
        lat + '&lon=' + lon + '&appid=' + ow_key
    try:
        r = requests.get(ow_request)
        weather_list = r.json()['list']
        for weather in weather_list:
            if weather['weather'][0]['id'] in range(200, 232) and not rain:
                rain_string = '\n¡Sal con paraguas! Empezará a llover a las ' +
\
                    datetime.utcfromtimestamp(weather['dt']).strftime('%H:%M')
            avg_temperature += int(weather['main']['temp'])
    except Exception:
        print('Ha ocurrido un error al solicitar la previsión')
    return 'La temperatura media de hoy será de ' +
str(round(avg_temperature/8,2)) + 'ºC.' + rain_string

def send_notification(notification_string):
    user = 'XXXXXXX'    # clave de user de PushOver
    token = 'XXXXXXX'  # clave de aplicación de PushOver
    json_header = {'token': token, 'user': user, 'message': notification_string,
'title': 'Weather'}
    requests.post('https://api.pushover.net/1/messages.json', json=json_header)

def main():
    weather_string = get_weather_info()
    send_notification(weather_string)

if __name__ == '__main__':
    main()
```

De nuevo, podemos comprobar que el programa funciona si guardamos el archivo y ejecutamos

```
python3 weather_bot.py
```

Y debería llegarnos una notificación con la previsión del tiempo.

Automatizando la ejecución del programa

Ahora que hemos escrito nuestro programa, el siguiente y último paso es que se ejecute cíclicamente sin necesidad de que nosotros toquemos la Raspberry, para ello emplearemos un programador de tareas en Linux conocido como “Cron Job”. Para programarlo primero deberemos hacer nuestro archivo ejecutable mediante:

```
sudo chmod +x ~/weatherbot/weather_bot.py
```

Lo cual podremos comprobar si ejecutamos:

```
./weathe_bot.py
```

Y lo copiamos a la siguiente carpeta para poder ejecutarlo desde cualquier directorio como un comando:

```
sudo cp weather_bot.py /usr/local/bin/
```

Para automatizarlo simplemente tendremos que editar /etc/crontab mediante:

```
sudo nano /etc/crontab
```

Y añadimos la siguiente línea

```
0 8 * * * root weather_bot.py
```

Que se traducirá en “Correr el comando ~/weatherbot/weather_bot.py a las 8:00 todos los días, todos los meses, todos los días de la semana”, puesto que la sintaxis de un Cron Job es tal que

```
"minute" "hour" "day" "month" "day of the week" "task"
```