Entregable 1

Sistemas Industriales

Francisco Javier Pérez Martínez

19 de diciembre de 2021

Índice

1. Descripción **3**

2. Captura de datos del API OpenWeather **3**
2.1. Panel de monitorización (Dashboard) 4

3. Captura de datos del API Esios **5**
3.1. Panel de monitorización (Dashboard) 5

4. Ampliación 1: Captura de datos del sensor BLE **6**
4.1. Panel de monitorización (Dashboard) 6
4.2. Publicación/Suscripción de los datos mediante MQTT 7

5. Ampliación 2: Captura del nivel de CO2 del aula **8**
5.1. Indicación del nivel con LED RGB 10
5.2. Panel de monitorización (Dashboard) 12

Referencias **12**

1. Descripción

Para el presente entregable, correspondiente a la primera parte de las prácticas de la asignatura se ha utilizado una **Raspberry Pi** para el desarrollo de los siguientes aspectos y herramientas: protocolos existentes de IoT, lectura de datos de APIs o la utilización de sensores físicos.

Además, destacamos la herramienta de node-red la cual ha sido utilizada para poder conectar ciertas APIs, nuestra Raspberry y servicios en internet para así facilitar la integración de estos.

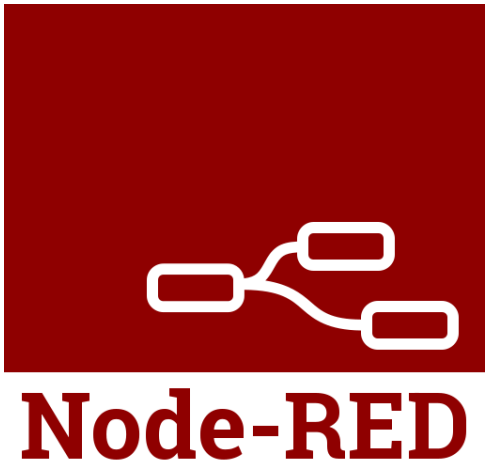

Figura 1: Herramienta Node-RED

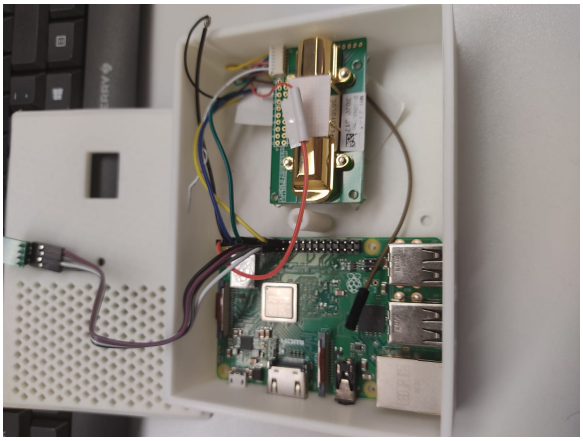

Figura 2: Raspberry Pi utilizada

2. Captura de datos del API OpenWeather

Para capturar datos de las API que proporciona OpenWeather es necesario registrarse en su web https://openweathermap.org/ y utilizar una API Key personal para poder obtener los datos y así analizar las diferentes prestaciones y servicios.

En esta sección, se ha obtenido el API que captura datos meteorológicos de cualquier coordenada geográfica. Se debe realizar una llamada a la API utilizando algunos de estos parámetros:

- Parámetros obligatorios:
  - **lat, long** : Coordenadas geográficas (latitud, longitud).
  - **appid** : Aquí introduciremos nuestra API Key.
- Parámetros opcionales:
  - **exclude** : Para excluir algunas partes de los datos meteorológicos de la API.
  - **units** : Unidades de medida (standard, metric e imperial).
  - **lang** : Para obtener los datos en nuestro idioma.

En nuestro caso, se han obtenido los datos meteorológicos de Alicante, concretamente la temperatura y también la previsión meteorológica de esta.

2.1. Panel de monitorización (Dashboard)

A continuación, se muestran los nodos de node-red utilizados para conectarse y realizar los filtros anteriormente mencionados:

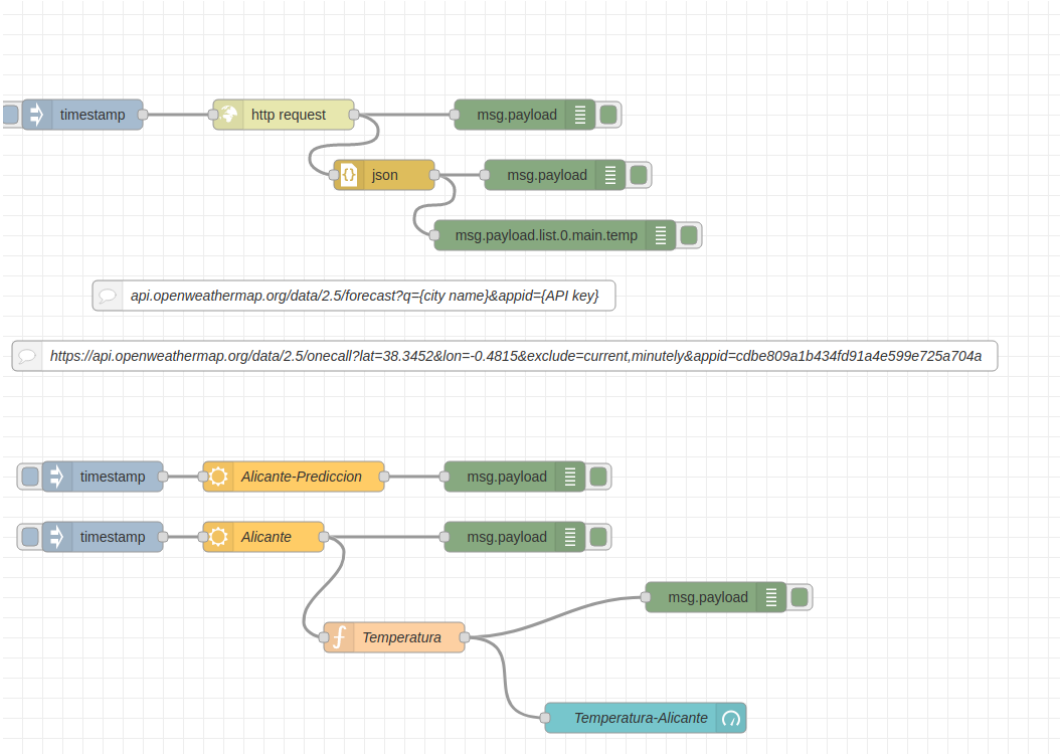

Figura 3: Nodos OpenWeather

Una vez finalizado la unión de los nodos, procedemos a desplegarlos e introducir después del puerto de node-RED **/ui** obteniendo así el dashboard en una interfaz de usuario.

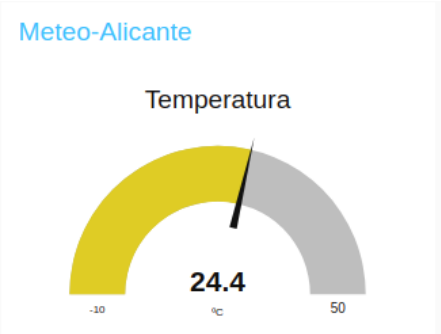

Figura 4: Dashboard Temperatura Alicante

3. Captura de datos del API Esios

Para capturar datos del API de ESIOS necesitaremos enviar un e-mail al siguiente correo consultasios@ree.es solicitando un token.

En esta sección, se ha obtenido el API que captura datos de la Red Eléctrica, concretamente el precio de electricidad actual de nuestro país.

3.1. Panel de monitorización (Dashboard)

A continuación, se muestran los nodos de node-red utilizados para conectarse y los tres filtros realizados: precio eléctrico actual, máximo y mínimo.

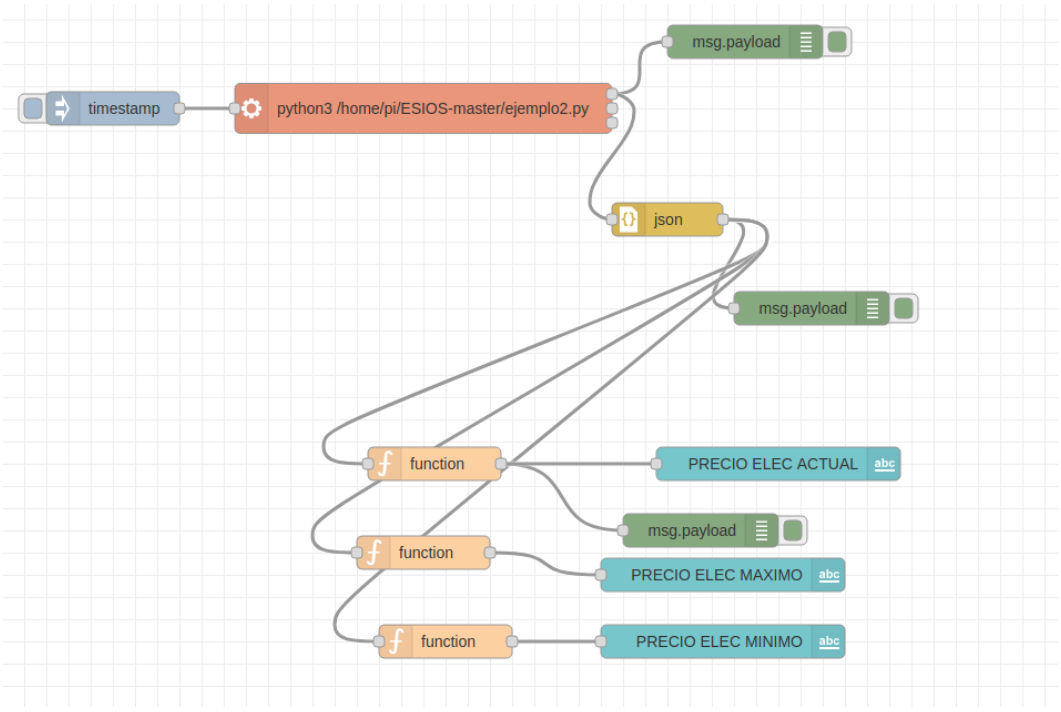

Figura 5: Nodos ESIOS

Una vez finalizado la unión de los nodos, realizando lo mismo que en la API anterior, obtendremos el siguiente dashboard:

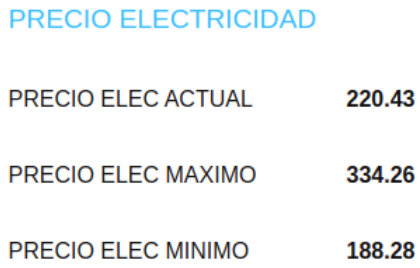

Figura 6: Dashboard Precio Electricidad

4. Ampliación 1: Captura de datos del sensor BLE

Para capturar datos del sensor BLE proporcionado en clase se ha usado bluepy, un módulo de Python que permite la comunicación con dispositivos Bluetooth Low Energy, para iniciar notificaciones BLE y posteriormente sondear los sensores.

4.1. Panel de monitorización (Dashboard)

A continuación, se muestran los nodos de node-red utilizados y los dos filtros realizados: Temperatura y Humedad del sensor BLE.

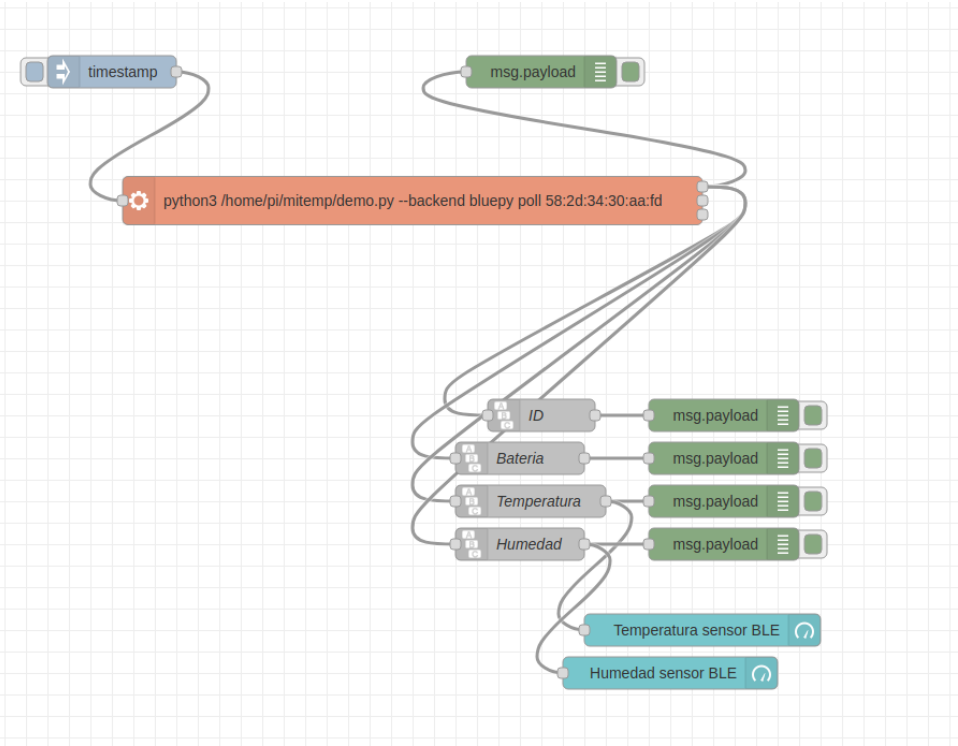

Figura 7: Nodos Sensor BLE

Una vez desplegado los nodos, nos dirigimos a la interfaz de usuario y obtendríamos el siguiente dashboard:

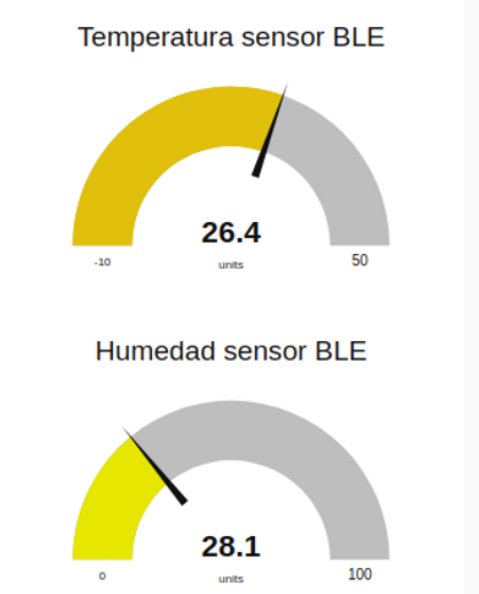

Figura 8: Dashboard Sensor BLE

4.2. Publicación/Suscripción de los datos mediante MQTT

El protocolo MQTT es un servicio de mensajería push con patrón publicador/suscriptor (pub-sub) entre máquinas (M2M). En primer lugar, nosotros como clientes nos conectaremos al servidor central, es decir, al broker. Una vez conectado, para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes suscritos.

En este apartado, hemos realizado una suscripción obteniendo la temperatura y la humedad del aula capturada por el sensor BLE que ha publicado otro grupo de compañeros.

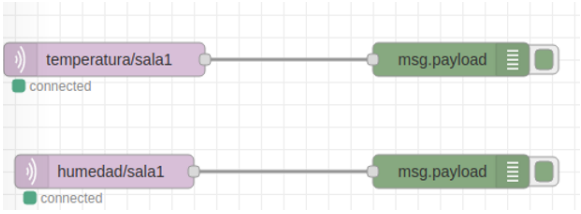

Figura 9: Topics MQTT BLE

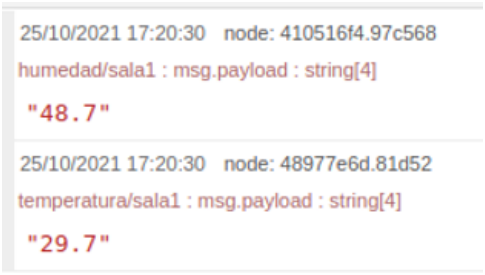

Figura 10: Valores obtenidos

5. Ampliación 2: Captura del nivel de CO2 del aula

Para ampliar lo visto en clase de prácticas, la Raspberry Pi utilizada cuenta con un sensor para medir los niveles de CO2 de un cierto espacio, en nuestro caso el aula de prácticas.

Figura 11: Sensor de CO2 conectado a la Raspberry Pi

Para ello, utilizando el siguiente código en Python podremos trabajar con el sensor creando así nuestro medidor de CO2:

```python
import serial
import time
import json

class CO2Sensor():
  request = [0xff, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79]

  def __init__(self, port='/dev/ttyS0'):
    self.serial = serial.Serial(
        port = port,
        timeout = 1
    )

  def get(self):
    self.serial.write(bytearray(self.request))
    response = self.serial.read(9)
    if len(response) == 9:
      current_time = time.strftime('%H:%M:%S', time.localtime())
      s = {"time": current_time, "ppa": (response[2] << 8) | response[3], "temp": response[4]}
      d = json.dumps(s)
      return d
    return -1

def main():
  # other Pi versions might need CO2Sensor('/dev/ttyAMA0')
  sensor = CO2Sensor()
  print(sensor.get())

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

Ejecutando el código obtendríamos el siguiente JSON:

```
Python 3.7.3 (/usr/bin/python3)
>>> %Run co2.py
  {'time': '16:55:05', 'ppa': 1091, 'temp': 71}
>>> %Run co2.py
  {'time': '16:56:09', 'ppa': 1356, 'temp': 71}
>>> %Run co2.py
```

Figura 12: Ejecución CO2 Sensor

Destacamos los parámetros:

- **time**: hora actual.
- **ppa**: nivel de CO2 en partes por millón.

5.1. Indicación del nivel con LED RGB

Además de capturar el nivel de CO2 del aula, se ha ampliado el código anterior para que dependiendo de ciertos niveles de CO2 mediante un LED instalado en la Raspberry se encienda un color u otro.

- Entre 0 y 400 = Verde
- Entre 400 y 800 = Azul
- Más de 800 = Rojo

El código ampliado es el siguiente:

```
1  import serial
2  import time
3  import json
4  import RPi.GPIO as GPIO
5  import time
6
7  # Define los colores en modo hexadecimal
8  COLORS = [0xFF0000, 0x00FF00, 0x0000FF, 0x00FFFF, 0xFF00FF, 0xFFFF00, 0xFFFFFF,
9            0xB695C0, 0xFFFF00]
10 # Establece los pines conforme a GPIO
11 pins = {'Red': 22, 'Green': 17, 'Blue': 27}
12
13 def mapea(x, in_min, in_max, out_min, out_max):
14     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
15
16 def set_color(color):
17     # calcula el valor para cada canal R, G, B
18     R_val = (color & 0xFF0000) >> 16
19     G_val = (color & 0x00FF00) >> 8
20     B_val = (color & 0x0000FF) >> 0
21
22     # Convierte el color de 0~255 a 0 ó 1 (entero)
23     R_val = int(mapea(R_val, 0, 255, 0, 1))
24     G_val = int(mapea(G_val, 0, 255, 0, 1))
25     B_val = int(mapea(B_val, 0, 255, 0, 1))
```

```
26
27      # asigna a cada pin el valor calculado
28      GPIO.output(pins['Red'], R_val)
29      GPIO.output(pins['Green'], G_val)
30      GPIO.output(pins['Blue'], B_val)
31
32  # quita la tensión en cada uno de los pines
33  def reset():
34      for i in pins:
35          GPIO.output(pins[i], 0)
36
37  class CO2Sensor():
38    request = [0xff, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79]
39
40    def __init__(self, port='/dev/ttyS0'):
41      self.serial = serial.Serial(
42          port = port,
43          timeout = 1
44      )
45
46    def get(self):
47      self.serial.write(bytearray(self.request))
48      response = self.serial.read(9)
49      if len(response) == 9:
50        current_time = time.strftime('%H:%M:%S', time.localtime())
51        s = {"time": current_time, "ppa": (response[2] << 8) | response[3], "temp": response[4]}
52        d = json.dumps(s)
53        return d
54      return -1
55
56  def main():
57    sensor = CO2Sensor()
58
59      # Establece el modo, en este caso a los valores GPIO
60    GPIO.setmode(GPIO.BCM)
61    for i in pins:
62          # configura los pines como de salida
63      GPIO.setup(pins[i], GPIO.OUT, initial=GPIO.HIGH)
64    reset()
65
66    s = sensor.get()
67    decoded = json.loads(s)
68    ppa = decoded['ppa']
69
70    if ppa >= 800:
71        set_color(0x00FF00) #rojo
72    elif ppa < 400:
73        set_color(0x0000FF) #verde
74    else:
75        set_color(0xFF0000) #azul
76    print(sensor.get())
77
78  if __name__ == '__main__': main()
```

Una vez implementado el código con el LED, podemos probarlo en node-red introduciendo ciertas inyecciones de prueba y comprobar si funcionan los 3 colores con los filtros mencionados:

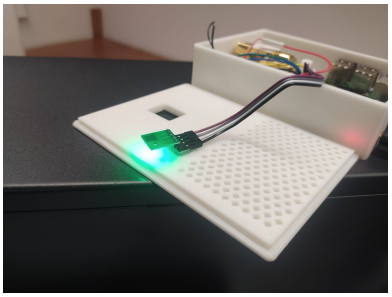

Figura 13: Prueba LED verde

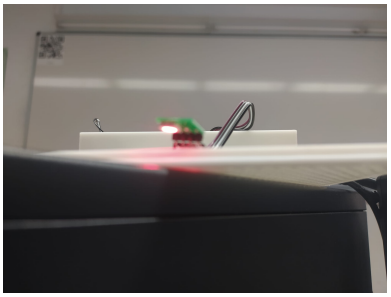

Figura 14: Prueba LED rojo

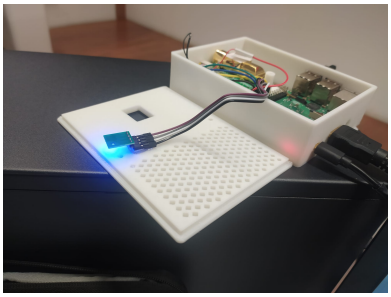

Figura 15: Prueba LED azul

5.2. Panel de monitorización (Dashboard)

A continuación, se muestran los nodos de node-red utilizados. Como podemos observar, al igual que en el apartado del sensor BLE se ha utilizado un nodo para el protocolo MQTT, pero en este caso nosotros somos el publicador, estamos publicando el nivel de CO2 capturado en el aula.

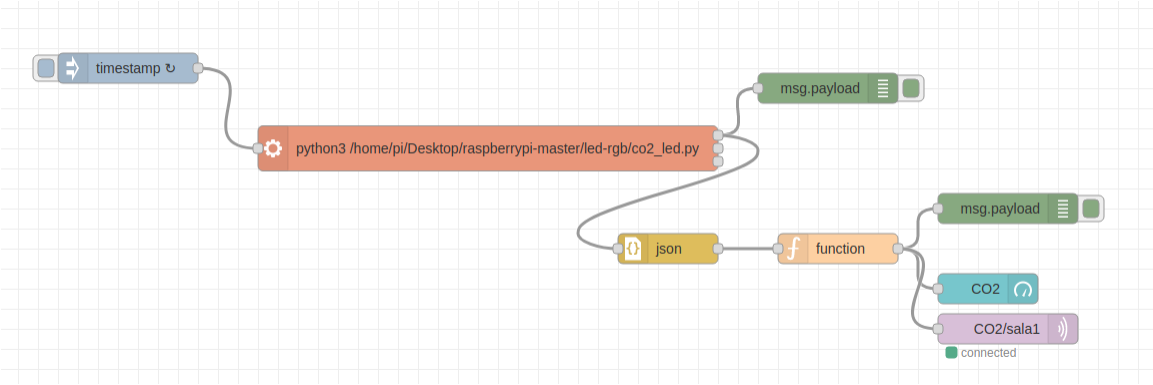

Figura 16: Nodos Sensor CO2

Una vez desplegado los nodos, nos dirigimos a la interfaz de usuario y obtendríamos el siguiente dashboard:

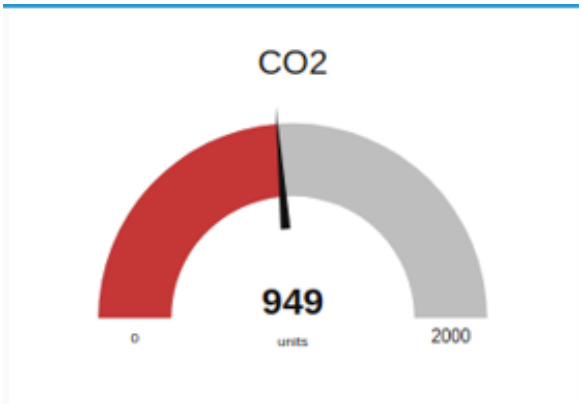

Figura 17: Dashboard Sensor CO2

Referencias

[1] Documentation. *Nodered.Org*. 2021. URL: https://nodered.org/docs/.

[2] Weather API. *Openweathermap.org*. 2021. URL: https://openweathermap.org/api.

[3] PVPC. *Ree.es*. 2021. URL: https://www.esios.ree.es/es/pvpc.

[4] L. Carbonell. *Jugando con la Raspberry Pi y un LED RGB. Atareao*. 2017. URL: https://atareao.es/raspberry/raspberry-pi-y-un-led-rgb/.

[5] Luis Llamas. *¿Qué es MQTT? Su importancia como protocolo IoT*. 2019. URL: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/.