Comunicación Cliente - Servidor

Comunicación Cliente - Servidor

Cliente

- Conexión WiFi en la misma red del servidor
- Recibe y envía mensajes MQTT

Servidor

- Servidor MQTT (Mosquitto)
- Interfaz de control (NodeRED)

WiFi - Modos de Operación

- Estación (STA_IF)
  - Se conecta a redes existentes
- AP (AP_IF)
  - Permite que otros dispositivos se conecten a ella

Consulta de redes disponibles

```
1  import network
2
3  wlan = network.WLAN(network.STA_IF)
4  if not wlan.active():
5      wlan.active(True)
6
7  redes = wlan.scan()
8
9  for red in redes:
10     print(f"SSID: {red[0].decode()}")
11     print(f"Canal: {red[2]}")
12     print(f"RSSI: {red[3]}dBm")
13     print("----")
```

Devuelve una lista en la que cada elemento contiene:

1. SSID
2. BSSID
3. Canal
4. RSSI
5. Modo de autenticación
6. Si está oculta

Conexión a una red conocida

```
1  import network
2  from time import sleep_ms
3
4  wlan = network.WLAN(network.STA_IF)
5  if not wlan.active():
6      wlan.active(True)
7
8  if not wlan.isconnected():
9      wlan.connect("ssid", "contraseña")
10
11     print("Conectando...")
12     while not wlan.isconnected():
13         sleep_ms(1000)
14
15 config = wlan.ifconfig()
16 print(f"Conectado con ip {config[0]}")
```

Inicializa la interfaz de red

Conecta a una red

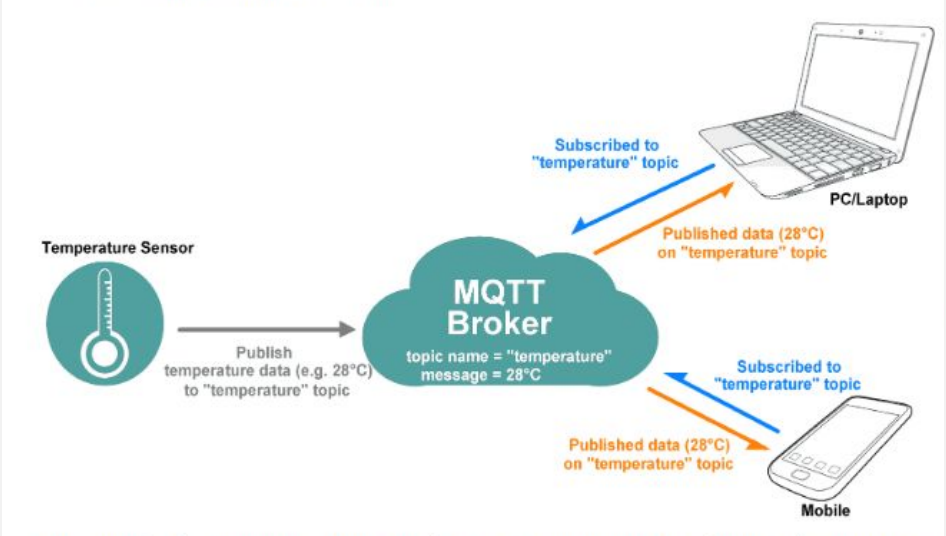
Muestra la IP actual

MQTT

- Protocolo de mensajería estandarizado que distribuye la información a través de publicación y suscripción (publish/subscribe) a determinado tema (topic)
- Es eficiente para situaciones que se transporta poca información ya que consume poco ancho de banda

MQTT - Funcionamiento

- La conexión entre el subscriber y el publisher se encuentran desacopladas.
- La comunicación es manejada por un tercer componente llamado broker (servidor)
- Un cliente puede suscribirse o publicar en diferentes topics en simultáneo
- Una red MQTT responde a una topología estrella



MQTT - Formato de Topics Sugerido

- Único dispositivo: **/<parámetro>**
- Múltiples dispositivos: **/<dispositivo>/<parámetro>**
- Ejemplos:
  - /humedad
  - /cocina/temperatura

MQTT - Instalación de Cliente MicroPython

En línea de comandos de MicroPython:

```
>>> import upip
>>> upip.install("micropython-umqtt.simple")
  Installing to: /lib/
  Warning: micropython.org SSL certificate is not validated
  Installing micropython-umqtt.simple 1.3.4 from https://micropython.org/
  pi/umqtt.simple/umqtt.simple-1.3.4.tar.gz
>>> upip.install("micropython-umqtt.robust")
  Installing to: /lib/
  Installing micropython-umqtt.robust 1.0.1 from https://micropython.org/
  pi/umqtt.robust/umqtt.robust-1.0.1.tar.gz
```

MQTT - Cliente

umqtt.simple

- Provee la funcionalidad básica del cliente de MQTT

umqtt.robust

- Agrega la capacidad de reconectar en caso pérdida de conexión

MQTT - Creación del Cliente

```
1 from umqtt.robust import MQTTClient
2 cliente = MQTTClient("nombre", "livra-mqtt.fi.mdp.edu.ar",
3                      port=1884, keepalive=30)
4 print("Conectando a MQTT...")
5 cliente.connect()
6 print("Conectado")
```

- Nombre: nombre arbitrario del cliente conectado. Tiene que ser único en la red
- Servidor: dirección del servidor
- Puerto: puerto donde escucha el servidor MQTT

MQTT - Suscripciones

- Escucha continuamente topics
- Ejecuta una función *callback* al recibir un mensaje con los parámetros:
  - Topic
  - Mensaje
- Los parámetros llegan como *bytes*: deben convertirse a texto con el método .decode()

MQTT - Suscripciones

```
4  def callback(topic, msg):
5      print(f"Llegó {msg.decode()} de {topic.decode()}")
6
7  cliente = MQTTClient("nombre", "livra-mqtt.fi.mdp.edu.ar",
8                       port=1884)
9  print("Conectando a MQTT...")
10 cliente.connect()
11 cliente.set_callback(callback)
12 cliente.subscribe("#")
13 print("Conectado")
14
15 while True:
16     cliente.check_msg()
17     sleep_ms(20)
```

Función *callback*

Configuración

Leo mensajes nuevos

MQTT - Publicaciones

- No lleva ninguna configuración adicional más que la creación del cliente
- No requiere *callback*
- Sintaxis:
  - cliente.publish(topic, mensaje)

MQTT - Ejemplo

Escribir un programa que se conecte a un servidor MQTT y emita un mensaje cada un segundo en un topic arbitrario en */alumnos/* y muestre los mensajes que escucha desde el topic */docente*.

Usar MQTT-Explorer para enviar y recibir mensajes.

MQTT - Ejercicio

Reutilizando el programa anterior, agregar el manejo de los siguientes mensajes:

- Suscripciones
  - **/<alumno>/salidas/led_<color>**: encender el LED del color correspondiente si se recibe *on*, sino apagar
  - **/<alumno>/salidas/servo:** leer el número recibido en el rango 0-180 y ubicar el servo en esa posición
  - **/<alumno>/salidas/neopixel/<num>**: se reciben tres valores separados por comas (R,G,B) y encender el NeoPixel como corresponda (ayuda: usar método .split(“,”) en el mensaje). Deben interpretarse qué píxel iluminar siendo el topic <num> el índice.

MQTT - Ejercicio

Reutilizando el programa anterior, agregar el manejo de los siguientes mensajes:

- Publicaciones
  - **/<alumno>/entradas/nivel**: al pulsar uno de los botones, enviar la posición del potenciómetro
  - **/<alumno>/entradas/alarma**: al pulsar el otro botón, enviar un mensaje de alarma
  - **/<alumno>/entradas/temperatura**: periódicamente enviar la temperatura actual
  - **/<alumno>/entradas/humedad**: periódicamente enviar la humedad actual