



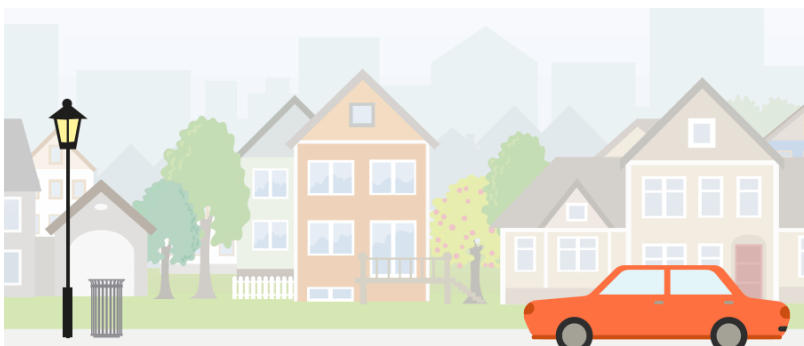
Jul. 2022

Unidad 6: Internet de las Cosas

Internet of Things (IoT)



A central illustration of a man in a light blue shirt and dark pants is surrounded by twelve circular icons, each connected to a label for a smart clothing item. The items are: Smart Ring (finger), Smart Glasses (eyes), Smart Shirt (with heart and respiration sensors inside), Smart Watch (wrist), Bluetooth Key Tracker (keychain), Smart Shoes (feet), Smart Socks (feet), Smart Pants (thighs), Smart Belt (waist), Smart Bracelet (wrist), SGPS/GPRS Baby Control (child icon), and Smart Finger (finger).



¿Qué es el IoT?

- Se refiere a cosas físicas que se comunican a través de una red digital
- IoT ofrece la posibilidad de recoger datos
 - sobre cómo usamos las cosas
 - para que funcionen mejor o de forma más inteligente



Las posibilidades

- Depende de los datos y el valor que pueden ofrecernos
- Los cambios que se ha producido en los últimos años han marcado la diferencia desde la vida cotidiana hasta la forma en que nos mantenemos sanos
- Pulsera fitness que monitoriza
 - frecuencia cardíaca
 - pasos dados al día
 - patrones de sueño
 - etc

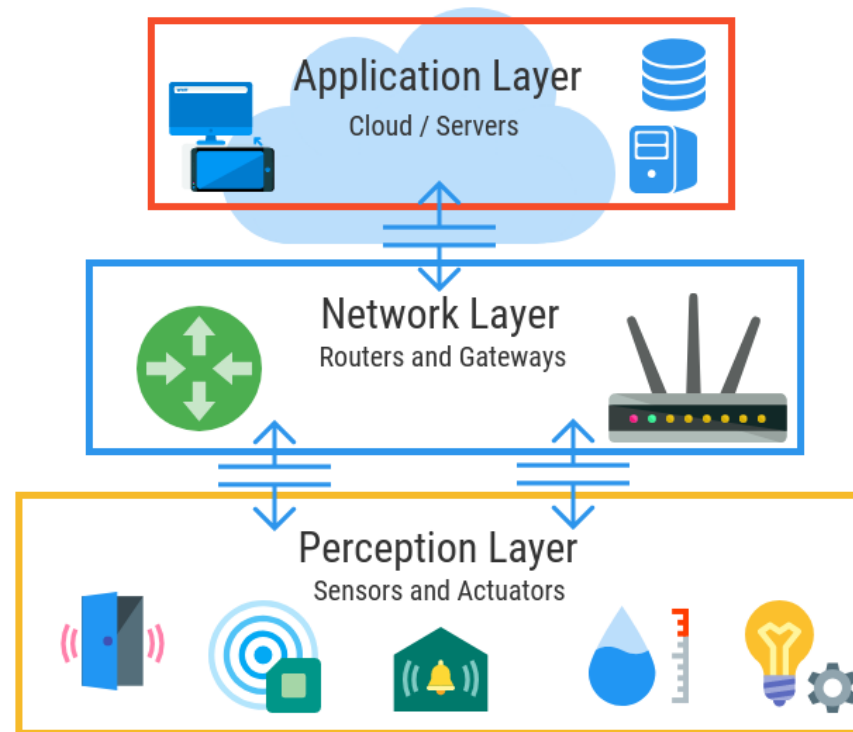


¿Cómo funciona?

- Transformar los **datos en bruto** en algo **útil y utilizable**
- Hay **pasos críticos** que todos los datos tienen que pasar

La arquitectura básica del IoT

- Arquitectura de tres capas interconectadas



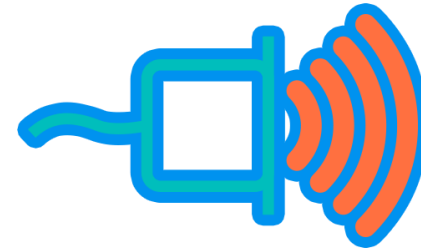
Objetivos de aprendizaje

- Explorar las características de la tecnología IoT
- Descubrir qué es el protocolo MQTT por medio del software [mosquitto](#)
- Usar la librería [Paho](#) para construir aplicaciones basadas en MQTT
- Usando código Python
 - Simular datos que provienen de dispositivos IoT
 - Publicar datos recuperados de los sensores en servidores MQTT locales y basados en la nube
- Construir un tablero de mando basado en la web

Tipos de dispositivos IoT

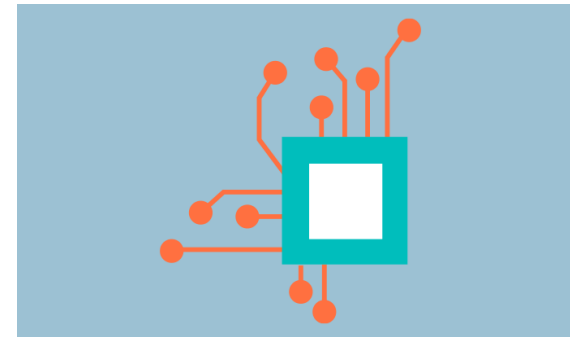
- Sensores
- Actuadores
- Puerta de enlace o

Procesamiento en el borde



Chipset / Sensor

- Dentro del dispositivo están incrustados un conjunto de chips
- Que pueden medir variables como
 - temperatura
 - aceleración
 - luminosidad
 - etc...
- El chipset obtendrá los datos



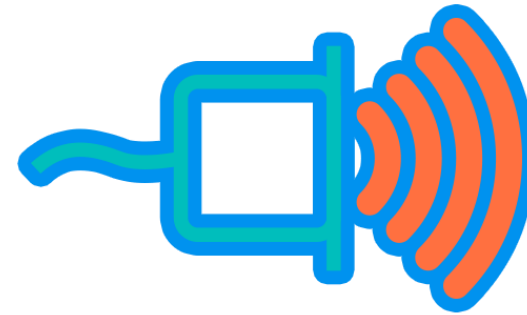
Sensores



- Recogen datos y los comunican a la siguiente capa
luego se transfieren a la nube
- Su capacidad de procesamiento
 - puede ser baja
(si se necesita mantener un tamaño reducido)
- Se incluyen en
 - Cosas como teléfonos, relojes y pequeños electrodomésticos
 - Artículos vestibles o fácilmente transportables

Actuadores

- Pueden convertir una señal eléctrica en una acción física
- Por ejemplo
 - cortar una fuente de alimentación
 - ajustar la presión de una válvula
 - mover un brazo robótico
 - activar una parada de emergencia
 - etc.



Puerta de enlace (*Gateway*)

- Reenvía (y encripta) los datos desde el dispositivo a Internet



La nube (*cloud*)

- Utiliza recursos de Internet para almacenar/procesar la información en lugar de hacerlo en los ordenadores locales
 - usuarios pueden acceder y colaborar en la información en tiempo real
- Manejar enormes cantidades de datos en bruto
- Crucial para la gestión de eventos concretos/la analítica de datos



Procesamiento en el borde

- El preprocesamiento de los datos tiene lugar cerca del sensor
- Cuando la solución requiere
 - una respuesta rápida / una acción inmediata
- Ayuda a reducir la cantidad de datos a transferir
- El sensor puede tener un procesador incorporado
 - aumenta el tamaño
 - la potencia de procesamiento
- No es adecuado para todos los casos de uso
 - Pero si para aplicaciones como
 - cirugías
 - vehículos autónomos



Interfaz de usuario

- Los datos se pueden pasar a los usuarios finales a través de aplicaciones/software (dashboards/cuadros de mando)
- Pueden presentarse como
 - información
 - recomendaciones
 - requisitos
 - automatización de tareas
 - Basadas en tendencias o en usos anteriores de los mismos datos



Análisis

- Extraer el valor (información) de grandes conjuntos de datos y explotar esa información
- Hay varias tecnologías emergentes disponibles para el análisis de datos
 - Inteligencia artificial
 - Aprendizaje automático
 - Aprendizaje profundo



El valor de los datos

- La industria en la que operamos está evolucionando
- Los datos son el activo más valioso que tenemos
- Con las nuevas soluciones tecnológicas de IoT podemos
 - ofrecer mejores servicios a los clientes
 - ahorrar tiempo y dinero



Arquitectura IoT (PubSub)



Patrón común en la programación embebida *ad infinitum*

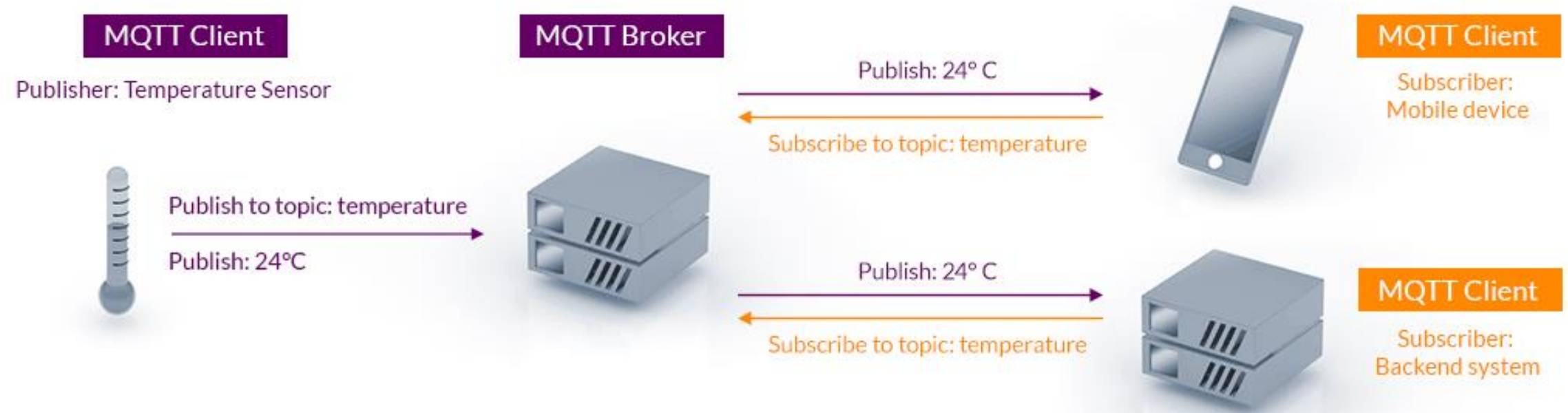
```
if __name__ == "__main__":  
  
    setup()  
    while True:  
        leer_sensores()  
        refrescar_UI()  
        # etc  
  
    # la ejecución del programa nunca llegará aquí
```

El protocolo MQTT

- MQTT es un protocolo de mensajería estándar para el Internet de las cosas (IoT)
- Ideal para conectar dispositivos remotos
 - con poco código
 - con poco ancho de banda
- MQTT se utiliza en una gran variedad de sectores:
 - Automoción
 - Fabricación
 - Telecomunicaciones
 - Petróleo y el gas



Arquitectura del protocolo



¿Por qué MQTT?

- **Ligero y eficiente:** Pueden utilizarse en microcontroladores. Las cabeceras de los mensajes MQTT son pequeñas para optimizar el ancho de banda de la red.
- **Comunicaciones bidireccionales:** Permite la mensajería entre el dispositivo y la nube y viceversa
- **Escalable:** Puede escalar a millones de dispositivos IoT
- **Entrega fiable:** 3 niveles de calidad de servicio
 - 0 - como máximo una vez
 - 1- como mínimo una vez
 - 2 - exactamente una vez
- **Soporte para redes poco fiables:** Sesiones persistentes reduce el tiempo de reconexión del cliente con el broker
- **Seguridad habilitada:** Cifrado de mensajes mediante TLS y la autenticación de clientes mediante modernos protocolos de autenticación, como OAuth.

Software

- Mosquitto
- Eclipse Paho

mqtt.org/software/

MQTT - Cliente

1. Importarnos `paho.mqtt.client`
2. Crear una instancia de la clase
3. Conectarse al broker utilizando una de las funciones `connect()`
4. Llamar a una de las funciones `loop()`
para mantener el flujo de tráfico de red con el broker
5. Utilizar `subscribe()` para suscribirse a un tema y recibir mensajes
Implementar las callbacks (`on_connect`, `on_message`)
6. Utilizar `publish()` para publicar mensajes en el broker
7. Utilizar `disconnect()` para desconectarse del broker

```
cliente = mqtt.Client()
```

MQTT - Publicar

- Publicar un único/múltiple mensaje en un bróker y desconectarse
- Sencillo

```
single(topic, payload=None, qos=0, retain=False,  
hostname="localhost", port=1883, client_id="", keepalive=60,  
will=None, auth=None, tls=None, protocol=mqtt.MQTTv311,  
transport="tcp")
```

- Múltiples

```
multiple(msgs, hostname="localhost", port=1883, client_id="",  
keepalive=60, will=None, auth=None, tls=None,  
protocol=mqtt.MQTTv311, transport="tcp")
```

MQTT - Suscribirse

- Permitir la suscripción y el procesamiento directo de los mensajes
- Simple

```
simple(topics, qos=0, msg_count=1, retained=False,  
hostname="localhost", port=1883, client_id="", keepalive=60,  
will=None, auth=None, tls=None, protocol=mqtt.MQTTv311)
```

- Callback

```
callback(callback, topics, qos=0, userdata=None,  
hostname="localhost", port=1883, client_id="", keepalive=60,  
will=None, auth=None, tls=None, protocol=mqtt.MQTTv311)
```



Callback (*Llamada de retorno*)

- Es una función que se pasa como argumento a otra función.
- Se espera que esta otra función llame a esta función callback en su definición.
- Las funciones callback se utilizan generalmente con funciones asíncronas.
- El código declara tres funciones

Función	Respuesta	cuando se haya
<code>on_connect</code>	CONNACK	establecido con éxito una conexión
<code>on_subscribe</code>	SUBACK	completado con éxito una suscripción
<code>on_message</code>	PUBLISH	publicado un mensaje basado en la suscripciones

Actividad 1: Monitorización desde la nube de una competición de surf en tiempo real

1. Entender los requisitos
2. Definir los temas y las cargas útiles
3. Codificación de un emulador de sensores de una tabla de surf
4. Configuración de la interfaz MQTT de PubNub
5. Publicar los datos de los sensores en el servidor MQTT de la nube
6. Trabajar con múltiples servidores MQTT
7. Construir un dashboard web con freeboard

Actividad 1: Monitorización desde la nube de una competición de surf en tiempo real

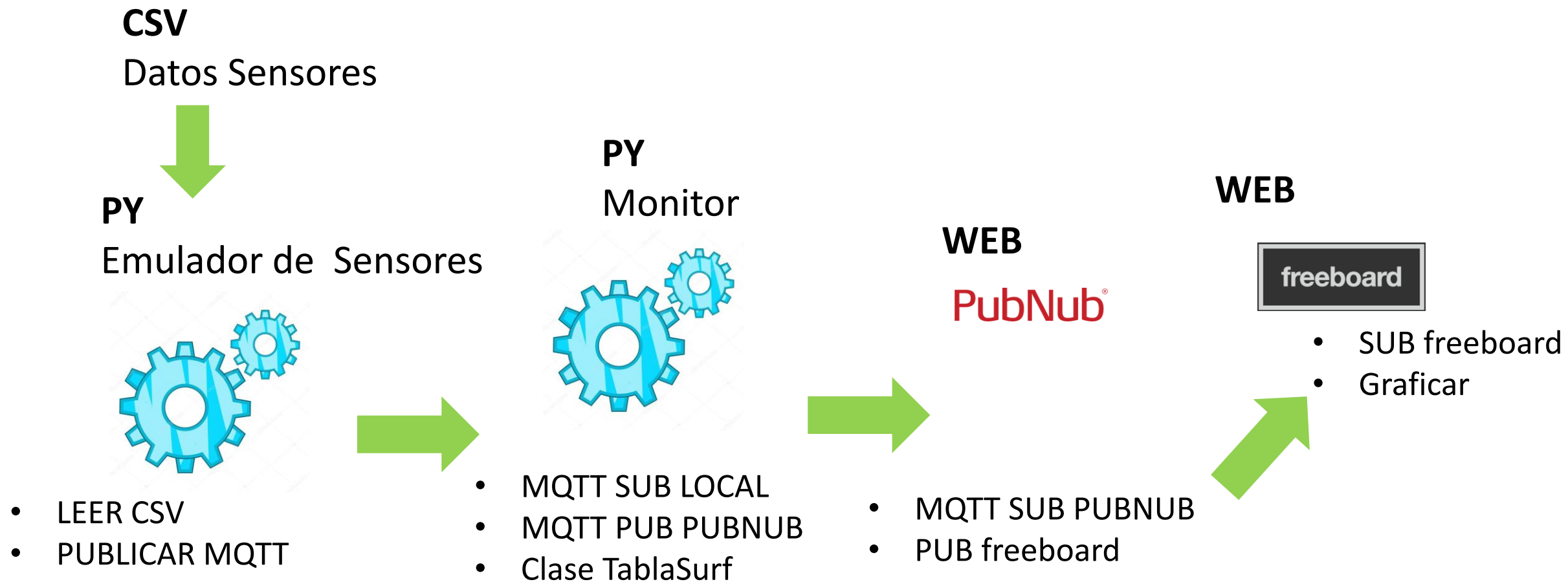
1. Requisitos:



Cada tabla proporciona los siguientes datos:

- **Estado:** Un sensor incorporado en el traje para indicar el estado del surfista
- **Velocidad:** Un sensor medirá la velocidad de la tabla de surf
- **Altitud:** Un sensor medirá la altitud de la tabla de surf en pies
- **Temperatura del agua:** Un sensor situado en una de las aletas de la tabla

Flujo de datos - Monitorización desde la nube de una competición de surf en tiempo real



Emulador de sensores de una tabla de surf

- El programa funcionará como si tuviéramos un surfista real con su traje de neopreno y los sensores de la tabla de surf montando olas y publicando datos.
- Este programa establecerá una conexión con el servidor Mosquitto MQTT y publicará cada segundo los valores de estado, velocidad, altitud y temperatura del agua leídos desde un archivo CSV en los temas correspondientes.

Emulador de sensores de una tabla de surf (cont.)

- El programa publicará cada segundo un único mensaje con los pares clave-valor que determinan el estado de un surfista y su tabla de surf a la interfaz PubNub MQTT.
- Establecerá una conexión con el servidor Mosquitto MQTT y se suscribirá a los temas en los que el emulador de sensores publique.
- Establecerá una conexión con la interfaz MQTT de PubNub.

Emulador de sensores de una tabla de surf (cont.)

Archivo CSV

- Emular escenarios reales de un surfista y su tabla de surf
- Datos recuperados de una breve sesión
- No es conveniente trabajar con valores aleatorios
- Cada línea del archivo representa un conjunto de valores que el emulador de sensores de la tabla de surf publicará en los temas correspondientes

emulador_sensores.py

C:\WINDOWS\system32\cmd.exe

```
Resultado de la conexión: Connection Accepted.  
tabla_surf/tabla_surf01/estado: 0  
tabla_surf/tabla_surf01/velocidad(kph): 1.6  
tabla_surf/tabla_surf01/altitud(metros): 0.61  
tabla_surf/tabla_surf01/temperatura(c): 40.0  
tabla_surf/tabla_surf01/estado: 0  
tabla_surf/tabla_surf01/velocidad(kph): 1.8  
tabla_surf/tabla_surf01/altitud(metros): 0.61  
tabla_surf/tabla_surf01/temperatura(c): 40.0  
tabla_surf/tabla_surf01/estado: 1  
tabla_surf/tabla_surf01/velocidad(kph): 3.2  
tabla_surf/tabla_surf01/altitud(metros): 0.91  
tabla_surf/tabla_surf01/temperatura(c): 39.0  
tabla_surf/tabla_surf01/estado: 1  
tabla_surf/tabla_surf01/velocidad(kph): 4.8  
tabla_surf/tabla_surf01/altitud(metros): 0.91  
tabla_surf/tabla_surf01/temperatura(c): 39.0  
tabla_surf/tabla_surf01/estado: 1  
tabla_surf/tabla_surf01/velocidad(kph): 4.8  
tabla_surf/tabla_surf01/altitud(metros): 0.91  
tabla_surf/tabla_surf01/temperatura(c): 39.0  
tabla_surf/tabla_surf01/estado: 1  
tabla_surf/tabla_surf01/velocidad(kph): 4.8  
tabla_surf/tabla_surf01/altitud(metros): 0.91  
tabla_surf/tabla_surf01/temperatura(c): 39.0  
tabla_surf/tabla_surf01/estado: 1
```

monitor.py

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

```
91, "Temperatura agua(c)": 38.0}
Mensaje recibido. Tema: tabla_surf/tabla_surf01/estado. Carga util: b'2'
Mensaje recibido. Tema: tabla_surf/tabla_surf01/velocidad(kph). Carga util: b'19.3'
Mensaje recibido. Tema: tabla_surf/tabla_surf01/altitud(metros). Carga util: b'0.91'
Mensaje recibido. Tema: tabla_surf/tabla_surf01/temperatura(c). Carga util: b'38.0'
Publicando: {"Estado": "Navegando de pie", "Velocidad(kph)": 19.3, "Altitud(metros)": 0.
91, "Temperatura agua(c)": 38.0}
█
```


Mensajería Pub/Sub de *PubNub.com*

- Permite crear experiencias conectadas con potentes API de mensajería publicación/suscripción
- Proporciona
 - transmisión de datos en tiempo real
 - señalización de dispositivos
 - cifrado AES incorporado
 - cifrado TLS/SSL opcional
- Los componentes atómicos que conforman un flujo de datos son:
 - claves API
 - mensajes
 - canales.
- Nos permite
 - crear un número ilimitado de canales
 - para cualquier conjunto de claves API
 - sin tener que declarar primero el canal.

The PubNub logo, consisting of the word "PubNub" in a red, sans-serif font.

Configuración de la interfaz MQTT de PubNub

- Reemplazar las claves
 - Publish key empieza con el prefijo "pub-c-"
`keypubnub_publish_key = "pub-c-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx"`
 - Subscribe key empieza con el prefijo "sub-c"
`keypubnub_subscribe_key = "sub-c-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx"`
- Datos de configuración
 - `pubnub_mqtt_server_host = "mqtt.pndsn.com"`
 - `pubnub_mqtt_server_port = 1883`
 - `pubnub_mqtt_keepalive = 60`
 - `device_id = tablasurf_nombre`
 - `pubnub_topic = tablasurf_nombre`

The PubNub logo, consisting of the word "PubNub" in a red, sans-serif font.

Configuración de la interfaz MQTT de PubNub

- PubNub requiere que nos registremos y creemos una cuenta

```
surfboard = TablaSurf("tabla_id",1,0,0,0)
pubnub_client_id = f"{pubnub_publish_key}/{pubnub_subscribe_key}/{tabla_id}"
pubnub_client = mqtt.Client(client_id=pubnub_client_id)
pubnub_client.on_connect = on_connect_pubnub
pubnub_client.on_disconnect = on_disconnect_pubnub
pubnub_client.connect(
    host=pubnub_mqtt_server,
    port=pubnub_mqtt_server_port,
    keepalive=pubnub_mqtt_keepalive)
pubnub_client.loop_start()
```

The PubNub logo, consisting of the word "PubNub" in a red, sans-serif font.



Desplegar Dashboards (*freeboard.io*)

- Es una página web que permite crear y desplegar dashboards basados en casi cualquier tipo de conjunto de datos generados por nuestros dispositivos IoT y publicados en la nube.
- Destaca por su
 - sencillez,
 - rapidez
 - asequibilidad
- Prueba gratuita de **30 días**



Configuración del dashboard en .io

- <http://freeboard.io>
- Crear nuevo: “Tabla_surf_01”
- Añadir una fuente de datos (*datasources*)
 - Escoger de tipo **PubNub**
 - Introducir la **clave de suscripción** copiada de la configuración en PubNub.
 - Introducir “**tabla_surf01**” en Channel.
- Comprobar que aparecen nuestros datos
 - Ej. `datasources ["surfboard01"] ["Status"]`

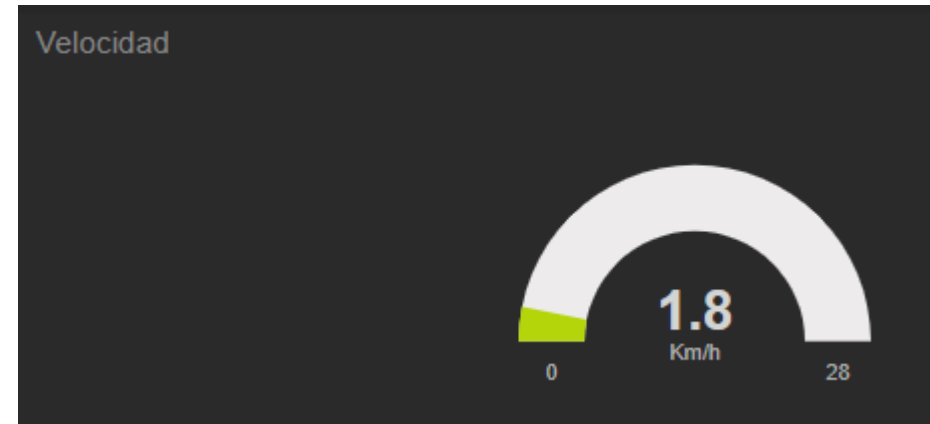
DATASOURCES		
Name	Last Updated	
tablaSurf01	11:36:28	 

Configuración del dashboard en .io

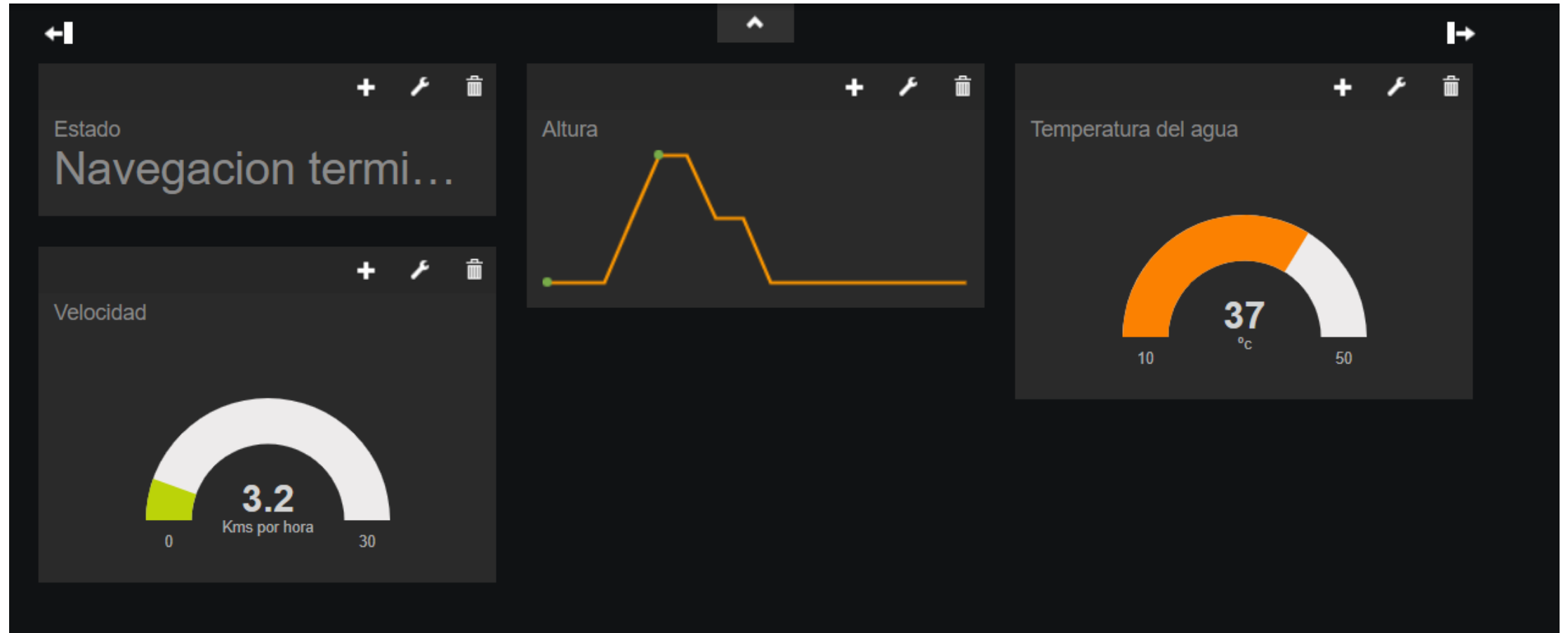
- Añadir paneles al gusto con los tipos de datos recibidos
 - Ej: Velocidad

WIDGET

TYPE	Gauge	+	DATASOURCE	JS EDITOR
TITLE	Velocidad			
VALUE	datasources["tablaSurf01"]["Velocidad(kph)"]			
UNITS	Km/h			
MINIMUM	0			
MAXIMUM	28			



Dashboard en freeboard.io



Resumen

Hemos:

- Explorado las características de la tecnología IoT
- Descrito qué es el protocolo MQTT por medio del software mosquitto
- Usado la librería Paho para construir aplicaciones basadas en MQTT
- Simulado datos que provienen de dispositivos IoT usando código Python
- Publicado datos recuperados de los sensores en el servidor MQTT basado en la nube
- Construido un tablero de mando basado en la web con freeboard