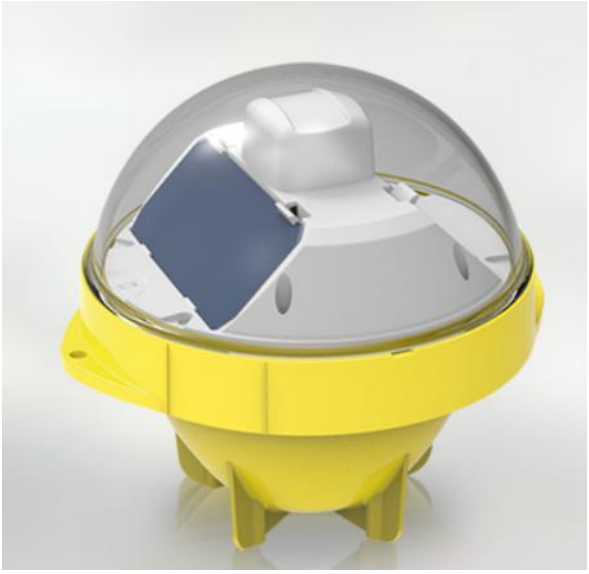
Tecnologías IIoT, hands-on y MVPs
Práctica: MVP para depuradora

Presentación

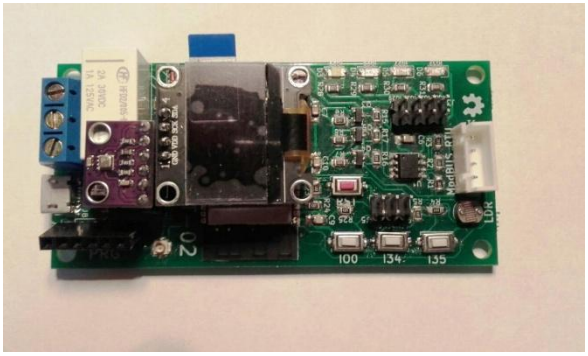
El objetivo principal de esta práctica es desarrollar un mínimo producto viable (MVP) de un sistema de IoT flotante, alimentado por batería de larga autonomía, que permita obtener lecturas del nivel del líquido en tanques, presión, temperatura, humedad relativa, nivel de luz, y en una evolución futura podría incorporar lecturas de sensores en contacto con el líquido como PH y otras informaciones provenientes del líquido de los tanques. El trabajo debe realizarse preferentemente en equipo, pudiendo también ser realizado de forma individual.

En el sector de la pesca hay disponibles diversos productos consistentes en boyas con conexión vía satélite para el seguimiento de bancos de peces y aplicaciones similares. Supondremos que la parte mecánica será resuelta por un equipo ajeno, así como un mecanismo de igualamiento de la presurización del interior con el exterior mediante una válvula de apertura puntual antes de realizar las lecturas de presión, temperatura u otras.



En la figura puede observarse una boya del fabricante Satlink (https://industriaspesqueras.com/noticia-66393-sec-Ferias%20y%20Exposiciones).

En nuestro caso la tecnología de comunicaciones inalámbricas a utilizar será a utilizar será Wi-Fi estándar, y utilizaremos la placa de prototipaje IoT-02, de hardware abierto, basada en el procesador ESP32:



Objetivos y competencias

Los **objetivos** de esta actividad son los siguientes:

1) Conocer la conexión de sensores a microcontroladores de escala mediana con capacidades de conexión inalámbrica.
2) Conectar sistemas embebidos a PCs industriales o al *cloud*, almacenando datos de las lecturas en base de datos como series temporales.
3) Utilizar sistemas de software para IoT como Node-RED y Grafana para monitorizar y visualizar los datos de las lecturas mediante dispositivos con navegador.
4) Conocer y experimentar con el concepto gemelo digital para prototipaje.

Requisito previo: Haber realizado algún ejercicio previo con la placa IoT-02.

Ejercicio 1

Se dispone de un firmware (ver punto de Recursos disponibles) para la placa IoT-02 que permite solicitar vía MQTTs lecturas de de temperatura (Tx100, que es la temperatura multiplicada por 100), humedad relativa (RHx100), presión (Px100), altura (Ax100), y nivel de luz de la LDR (LDR) provenientes del sensor BME-280 incorporado y la LDR integrada en la placa. El archivo de **`cabecera IoT-02_mqttTopics.h`** define los temas MQTT disponibles en la placa, los cuales permiten solicitar lecturas individuales de cada lectura, así como una lectura conjunta de los sensores en el siguiente formato JSON:

```
{"Tx100":2768,"RHx100":4550,"Px100":100123,"Ax100":-830,"LDR":3040}
```

Programad la placa de electrónica embebida IoT-02, indicando en caso los datos de conexión de vuestro punto de acceso Wi-Fi, y verificad los datos relativos al broker MQTT que vais a utilizar (podéis optar entre broker seguro o broker no seguro):

Broker no seguro
Datos del *broker*: Servidor: vps656540.ovh.net Puerto MQTT: 1883 Puerto Websockets (no seguro): 8081 Usuario: user Contraseña: pass

Implementad un programa en Node-RED que realice una petición MQTT a la placa IoT-02 de las lecturas en el formato JSON indicado, y las represente en un *dashboard* de Node-RED que tenéis que crear.

Cada uno de vosotros dispone de una instancia de Node-RED que utiliza el puerto cuyo número tenéis asignado.

Ejercicio 2

Implementar un proceso que se suscriba al *topic* de publicación de la IoT-02 de forma que las lecturas se almacenen en una base de datos MySQL[1]. Para ello puede utilizarse como lenguaje de programación Node-RED [2].

En cuanto al MySQL, deberéis anteponer el prefijo "m" + puerto + "_" al nombre de la tabla de la base de datos.

Representar en el sistema Grafana las lecturas de presión, temperatura, humedad relativa y nivel de luz almacenadas en la base de datos.

Utilizar los usuarios de Grafana disponibles en el VPS del máster, y nombrar los paneles de control (*dashboards*) según la regla del prefijo "m" + puerto + "_".

[2] Node-RED es la opción por defecto, pero quien tenga interés puede utilizar adicionalmente Python, Node.js o una combinación de éstos. Asimismo, quien tenga interés, puede probar algún otro tipo de base de datos como Microsoft Azure SQL, MongoDB u otro sistema.

[1] En nuestro caso utilizaremos MariaDB, que es un fork open source del proyecto original de MySQL (MariaDB está impulsado por Google y MySQL está en la órbita de Oracle)

Ejercicio 3

El sensor de presión se puede utilizar para medir la altura en la que está la placa IoT-02, la cual se puede medir calculando la diferencia entre las lecturas entre dos puntos. En un caso real, dispondríamos de dos placas IoT-02, una colocada a la altura del suelo del depósito y la otra colocada en la boya flotante, de manera que la altura se calcularía por la diferencia de las dos lecturas. Como sólo disponemos de una placa, para calcular la altura, deberemos realizar una primera lectura con la placa en el suelo, y utiliza dicha lectura para el cálculo de la diferencia con la IoT-02 y ver cómo va evolucionando en tiempo real. Alimentar la placa con un *powerbank* de teléfono móvil puede ayudar a realizar el experimento.

La presión de la placa a la altura del suelo va variando con el tiempo, pero supondremos que la variación de ésta mientras dure el experimento, no es significativa.

Se pide que incorporéis en el *dashboard* del ejercicio 1 la visualización de la altura de la placa IoT-02 respecto al suelo en tiempo real. Valorad la precisión del sistema y proponed un sistema de redondeo para aplicar en depósitos de depuradoras.

Ejercicio 4

En la dirección https://wokwi.com/projects/328227183923298899 hay un gemelo digital de un módulo de prototipaje popular basado en ESP32, que se acompaña del código fuente que permite la consulta síncrona (request/response) con MQTT de la lectura de una LDR[2].

Por otra parte, en http://vps656540.ovh.net/snap/snap.html#open:test_dt_esp32_ldr.xml se muestra cómo realizar las consultas con un bloque del tipo “MQTT Request”, que implementa la llamada request/response síncrono con MQTT[3].

Para ver el funcionamiento de llamadas request/response síncrono se pide:

1) Arrancar el simulador del gemelo digital[4]

[2] Se basa en el mecanismo explicado en el apartado request/response blocks en el que el símbolo @ actúa de marca a partir del cual está el topic de callback (https://github.com/pixavier/mqtt4snap)

[3] Al igual que el punto anterior (2)

[4] Dependiendo de la hora del día, la conexión del simulador Wokwi puede fallar de forma intermitente, puesto que está en modo demostración sin coste

2) Arrancar la página low-code (Snap*!*) del link anterior, ejecutar los bloques de conexión y luego el bloque de consulta, y comprobar que se realiza la lectura de forma síncrona, como se muestra en la Fig 1.
3) Ejecutar el bloque tipo *reporter* denominado “getLux” i analizar cómo está implementado.

.

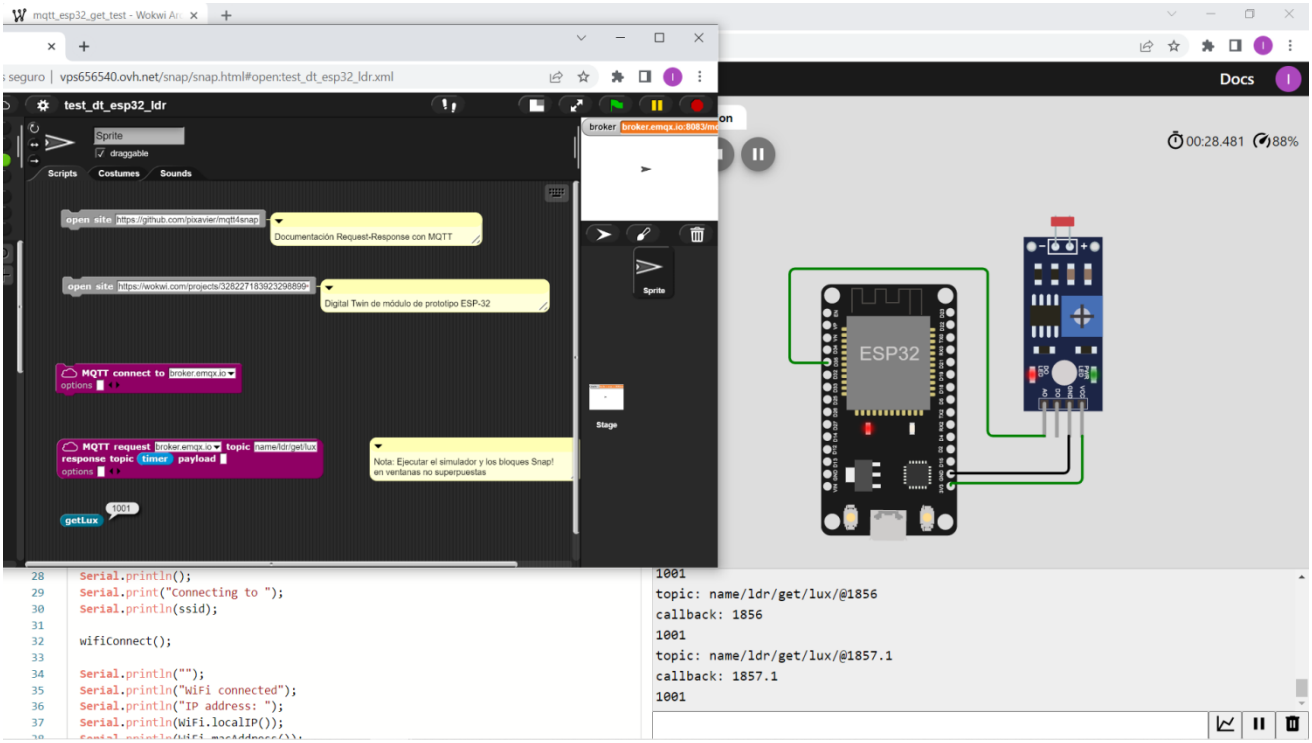


Fig 1

Se pide que implementéis un bloque (o *subflow*) denominado getLux en Node-RED con la funcionalidad equivalente al bloque con el mismo nombre de Snap*!*, es decir, que permita la consulta síncrona del nivel de luz vía MQTT.

Modificad el firmware de la IoT-02 para que soporte consultas síncrona vía MQTT de los valores de las magnitudes indicadas en el JSON del enunciado del ejercicio 1. Podéis implementarlo para consultas individuales (como getLux) o bien para consultar el conjunto en un JSON.

Nota: Podéis realizar el desarrollo del bloque de Node-RED getFlux contra el gemelo digital (modalidad *Virtual Commissioning*), o bien empezar por el firmware y desarrollarlo contra la IoT-02.