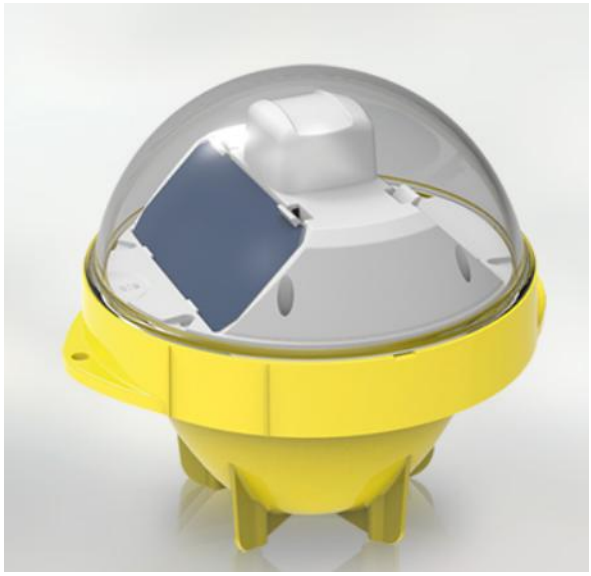
# Tecnologías IIoT, hands-on y MVPs
## Práctica: MVP para depuradora

## Presentación

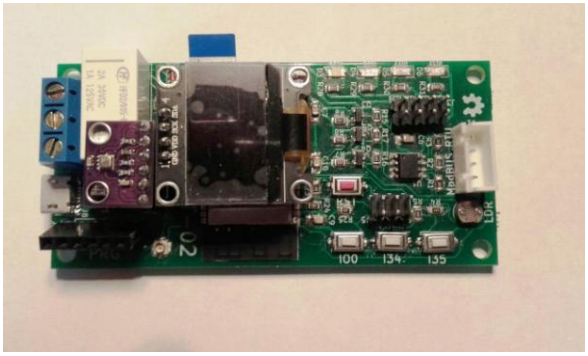
El objetivo principal de esta práctica es desarrollar un mínimo producto viable (MVP) de un sistema de IoT flotante, alimentado por batería de larga autonomía, que permita obtener lecturas del nivel del líquido en tanques, presión, temperatura, humedad relativa, nivel de luz, y en una evolución futura 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.



En la figura puede observarse una boya del fabricante Satlink (https://satlink.es/boyas-inteligentes/satlink-boya-palangre).

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 bases de datos orientadas a series temporales como InfluxDB.
3) Utilizar sistemas de software para IoT como NodeRED y Grafana.
4) Implementar servicios basados en HTTP y MQTT para monitorizar y visualizar los datos de las lecturas mediante dispositivos con navegador.
5) Conocer y experimentar con el concepto de Virtual Commissioning

Se recomienda haber realizado el ejercicio con enunciado "**P.04 - Enunciado Ejercicio Hands-on IoT**", y tener conocimientos de la placa IoT-02.

# Ejercicio 1

Se dispone de un firmware (ver punto de Recursos disponibles[1]) para la placa IoT-02 que permite solicitar vía MQTTs lecturas de de temperatura (T), humedad relativa (RH), presión (P), altura (A), y nivel de luz (L) 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:

```
{"T":20.0,"RH":45.5,"P":1001.2,"A":-8.3,"L":3040}
```

[1]Recursos disponibles:
http://binefa.com/index.php?title=Algunas_pistas_para_la_pr%C3%A1ctica_MVP_para_depuradora_(curso_2019-2020)

Programar 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 seguro | Broker no seguro |
|---|---|
| Datos del *broker*:<br>Servidor: iot.siarq.net<br>Puerto MQTTS: 8883<br>Puerto Websockets: 9001<br>Usuario: upc<br>Contraseña: school | Datos del *broker*:<br>Servidor: vps656540.ovh.net<br>Puerto MQTT: 1883<br>Puerto Websockets (no seguro): 8081<br>Usuario: user<br>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 se obtiene mediante la siguiente fórmula:

**`PuertoUsuario_masterxy = 1000 + xy * 880`**

Por ejemplo, el usuario master11 utilizará el puerto 11880.

Para gestionar las instancias de los Node-RED se dispone de la herramienta pm2, que está instalada en el VPS del máster, la cual permite ver la ejecución de las instancias:

**`sudo pm2 status`**

```
$ sudo pm2 status
```

| App name | id | version | mode | pid | status | restart | uptime | cpu | mem | user | watching |
|---|---|---|---|---|---|---|---|---|---|---|---|
| node-red00 | 2 | N/A | fork | 27988 | online | 0 | 23h | 0% | 2.9 MB | root | disabled |
| node-red10 | 1 | N/A | fork | 8391 | online | 0 | 20m | 0% | 2.8 MB | root | disabled |
| node-red11 | 3 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red12 | 4 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red13 | 5 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red14 | 6 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red15 | 7 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red16 | 8 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red17 | 9 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red18 | 10 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red19 | 11 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red20 | 12 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red21 | 13 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red22 | 14 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red23 | 15 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red24 | 16 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red25 | 17 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red26 | 18 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red27 | 19 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red28 | 20 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |
| node-red29 | 21 | N/A | fork | 0 | stopped | 0 | 0 | 0% | 0 B | root | disabled |

Así como arrancarlas y pararlas:

```
sudo pm2 start node-redxy
sudo pm2 stop node-redxy
```

*(pm2 es una utilidad que os podéis instalar en vuestro ordenador personal para vuestros proyectos: npm install pm2).*

# 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 InfluxDB. Para ello puede utilizarse como lenguaje de programación Node-RED [2].

En cuanto al InfluxDB, deberéis anteponer el prefijo "m" + xy + "_" al nombre del *measurement* (equivalente a una tabla) de la base de datos.

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

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

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

# 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 utilizarla para el cálculo de la diferencia cuando se esté colocando la placa a diferentes alturas y ver cómo va

evolucionando la lectura de ésta 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.

Nota sobre la seguridad de Node-RED:
Cada uno de los usuarios masterxy dispone de una carpeta propia **`node_modules/node-red`**, con el correspondiente fichero de configuración settings.js. Como tarea opcional, se pide activéis un acceso con usuario/contraseña para vuestra instancia de Node-RED (https://nodered.org/docs/user-guide/runtime/securing-node-red).

## Anexo: Virtual Commissioning

Uno de los conceptos emergentes en la Industria 4.0 es el Virtual Comissioning (entregas con puesta en marcha virtual). Para esta desarrollo se ha preparado un Digital Twin mínimo del sistema consistente en dos partes:

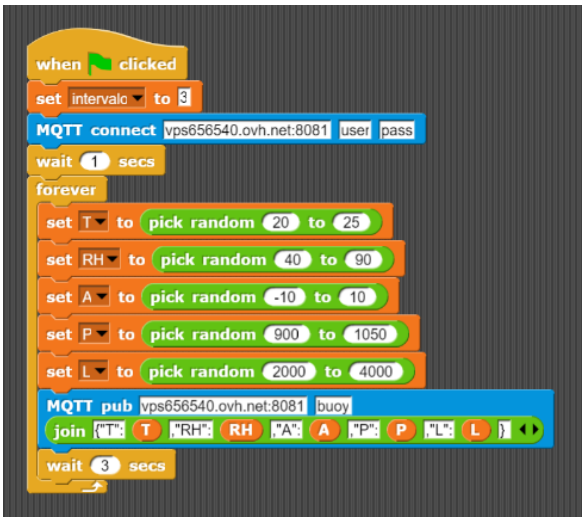
- Un simulador-1 mínimo de la IoT-02 en Snap! (Correspondiente al ejercicio 1)
- Un simulador-2 mínimo del sistema de recepción de mensajes en Node-RED (Correspondiente al ejercicio 2)

Si suponemos que los ejercicios 1 y 2 corresponden a los subsistema-1 y subsistema-2 (respectivamente), tal que juntos constituyen un sistema, la idea es que la entrega del subsistema-1 se valide contra el simulador-2, y que el subsistema-2 se valide contra el simulador-1

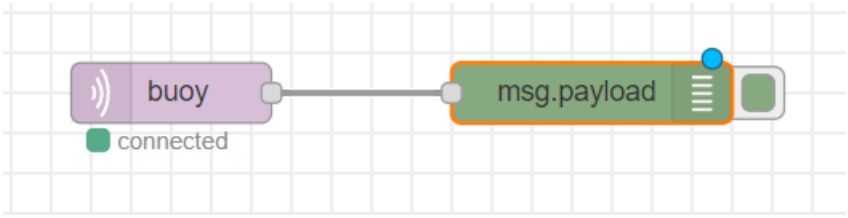
El simulador-1 genera y publica mensajes en JSON según el formato:

```
{"T":20.0,"RH":45.5,"P":1001.2,"A":-8.3,"L":3040}
```

El simulador-1, al estar implementado en Snap!, es fácilmente parametrizable (tiempo entre envío de mensajes, naturaleza de la aleatoriedad, etc).




En cuanto al simulador-2, la idea es poder validar que los datos se reciben correctamente, observando el comportamiento de la ventana de depuración.




Como muestra, se pueden ver los simuladores del Digital Twin mínimo en acción en las siguientes direcciones:

Simulador-1:

http://vps656540.ovh.net:88/snap.html#open:projs/buoy.xml

Simulador-2:

http://vps656540.ovh.net:1880/#flow/e603b6df.48d738