Water Quality Sensor Module

1. Descripción y objeto

Se ha creado, con el fin de tener una capa de abstracción que simplifique la programación de las misiones de más alto nivel, un módulo que envuelva la adquisición, interpretación y guardado en la base de datos de las muestras provenientes del Smart Water. El módulo está escrito en el archivo **SensorModule.py** y, de momento, contiene una única clase denominada **WaterQualityModule**.

Esta clase es una abstracción de los códigos previamente desarrollados por Jose Miguel, Alicia y Samuel y constituye el bloque final de implementación de los sensores en el código de los drones. Dentro de este módulo y clase, se han de implementar las mejoras que se consideren. Todo lo relacionado con los sensores, debe estar contenido en dicha clase para evitar que, durante el diseño de las misiones y del código de alto nivel, se tenga que llamar a rutinas u otros

```
WaterQualityModule()
|
|------ __init__(database_name,
|                USB_string,
|                    timeout,
|                  baudrate)
|
|------ take_a_sample(position,
|                     num_of_samples)
|
|------ close()
```

scripts paralelos.

La arquitectura del módulo es simple:

2. Funcionamiento

El funcionamiento del módulo es muy sencillo:

- Para importar el módulo, el archivo .py del mismo tiene que estar en la misma carpeta que el script que va a usar sus objetos:

```
from SensorModule import WaterQualityModule
```

- A continuación, podemos crear un objeto de tipo **WaterQualityModule**. En la creación del objeto, se dispara la función constructora (__init__). Se le han de pasar 4 argumentos: **database_name** (string: **“LOCAL_DATABASE.db”**, por ejemplo), **USB_string** (string: **“USB1”**, por ejemplo), **timeout** (int: 6, por ejemplo) y **baudrate** (int: 115200, por ejemplo).

```
modulo_de_sensores = WaterQualityModule(database_name = 'LOCAL_DATABASE.db',
                                        USB_string = 'USB1',
                                        timeout = 6,
                                        baudrate = 115200)
```

Esto crea el objeto, se conecta a la base de datos (que ha de estar en el mismo directorio) y crea la conexión serial con el USB que se le indique. Previamente habrá que comprobar si, efectivamente, el USB1 (o el que fuera) es en el que hay que conectarse o no. En futuras modificaciones la detección automática se incluirá. De momento, se le indica directamente puesto que primero se conecta el TELEM y luego el Zigbee. El timeout y el baudrate son predeterminados pero pueden indicarse como parámetros por si hiciera falta.

- Una vez creado el objeto, simplemente puede llamarse al método **take_a_sample**, que recibe una lista de dos componentes con la posición actual del dron (que se lee en el programa de más alto nivel con el DroneKit) y el número de muestras que se quiere que se tome:

```
check = modulo_de_sensores.take_a_sample(position = [24.34, 35.45],
                                         num_of_samples = 4)
```

La posición es una lista ordenada: position[0] es la LATITUD y position[1] es la LONGITUD.
El número de muestras indica cuantas muestras seguidas se toman y se guardan para esa posición concreta. El método devuelve un check True cuando termina de tomar todas las muestras.

- Para cerrar las conexiones, se utiliza el método **close()**:

```
modulo_de_sensores.close()
```

Esto cierra la conexión seria y se desconecta de la base de datos.

---

3. Test

Se ha escrito un test de verificación llamado test_SensorModule.py. Asegurarse que se ejecuta el script **create_database.py** primero.

NOTAS IMPORTANTES:

- Se ha eliminado de la implementación de Jose Miguel lo relacionado con la instalación de librerías. Un script de Python jamás debería instalar ninguna librería. Se presuponen instaladas en cada dron anteriormente.
- Como tenemos dos tipos de Smart Water y uno de ellos no tiene sensor de Oxígeno Disuelto, el valor predeterminado en el diccionario **self.sensor_data['DO']** se pone a -1. Si existe sensor, en la trama se hace override de ese valor con el valor real. Si no existe el sensor, no se sobrescribe.
- El módulo recibe el USB al que se ha de conectar. Debería ser un programa de más alto nivel que determine cuál es, nunca comprobarlo más a abajo o la arquitectura resulta demasiado liosa.

- La base de datos ha de crearse de antes. Es como con la librería. Se presupone creada y nunca se ha de crear desde el mismo programa que maneja los sensores.