Escola Universitaria Politécnica

UNIVERSIDADE DA CORUÑA

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO DE FIN DE GRADO

TFG Nº: **770G01A193**

TÍTULO: **CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DE DISPOSITIVOS HARDWARE DE BAJO COSTE**

AUTOR: **YAGO GÓMEZ CASTRO**

TUTOR: **JOSÉ LUIS CASTELEIRO ROCA**
FRANCISCO ZAYAS GATO

FECHA: **DICIEMBRE DE 2020**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

TÍTULO: **CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DE DISPOSITIVOS HARDWARE DE BAJO COSTE**

ÍNDICE

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2020

AUTOR: EL ALUMNO

Fdo.: YAGO GÓMEZ CASTRO

Listado de figuras

Listado de tablas

Listado de códigos de programación

TÍTULO: **CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DE DISPOSITIVOS HARDWARE DE BAJO COSTE**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2020

AUTOR: EL ALUMNO

Fdo.: YAGO GÓMEZ CASTRO

Índice del documento MEMORIA

1 OBJETO

El objeto de este Trabajo Final de Grado es el de estudiar, primeramente, aquellos parámetros medidos en el agua procesada por las Estaciones de Depuración de Agua Residual. A continuación, se implementarán distintos tipos de sensores de bajo coste que permitan caracterizar aguas residuales. Los datos obtenidos a partir de los sensores implementados serán visualizados, procesados y almacenados en tiempo real. Por último, se estudiará la posibilidad de diseñar e implementar un sensor virtual capaz de determinar la naturaleza anómala de las aguas residuales.

2 ALCANCE

El presente proyecto consta de 6 etapas claramente diferenciadas, las cuales se enumeran a continuación:

- Estudiar los parámetros medidos sobre aguas residuales en Estaciones de Depuración de Agua Residual.
- Estudio de los distintos tipos de sensores de bajo coste que permitan caracterizar una muestra de agua.
- Conexionado físico de los elementos necesarios para el correcto funcionamiento de los sensores seleccionados.
- Desarrollo de un procedimiento para realizar las medidas.
- Análisis de la correlación entre los resultados de las medidas realizadas y los parámetros obtenidos en el laboratorio.
- Estudio de la posibilidad de desarrollar sensores virtuales.

3 ANTECEDENTES

En esta sección se definirá el punto de partida del proyecto. El objetivo será facilitar, mediante definiciones y explicaciones breves, la comprensión de algunos elementos básicos que lo conformarán.

3.1. Muestras.

El proyecto constará de varios sensores que medirán los parámetros expuestos en la sección 3.2, para comprobar que dichos sensores ofrecen medidas correctas se han utilizado una serie de disoluciones químicas. Las disoluciones de laboratorio facilitadas se pueden ver en la tabla 3.1.

Muestras		
Muestra	**Parámetro**	**Composición**
Agua desionizada o MQ (Milli Q)	Referencia para varias medidas	Iones H^+ y OH^-
Agua residual LAB	Turbidez	$0{,}27g\ peptona + 1g\ urea$ $+0{,}007g\ NaCl + 4g\ K_2HPO_4$ $+0{,}004g\ CaCl_2 2H_2O$ $+0{,}002g\ HSO_4 7H_2O$ $+Agua\ grifo\ (1L)$
Agua residual EDAR	Turbidez	Desconocida
Muestra HCl	pH	Agua MQ + HCl
Muestra $NaCl$	Conductividad eléctrica	$NaCl\ 4M$
Muestras incluídas con sensor de conductividad	Conductividad eléctrica	$12,88mS/cm$ y $1413uS/cm$
Muestra $AgNO_3$	Potencial reducción-oxidación	$AgNO_3\ 0{,}024M$
Muestra Ac_2Zn	Potencial reducción-oxidación	$Ac_2Zn\ 0{,}061M$
Muestra colorimetría	Espectro electromagnético y absorbancia	$CuSO_4\ 1M$

Tabla 3.1 – Tabla con las muestras facilitas para la realización del proyecto.

En las imágenes 3.1, 3.2 y 3.3 se pueden ver las muestras mencionadas.

(a) Agua MQ (b) Agua residual LAB (c) Agua residual EDAR

Figura 3.1 – Muestras de agua residual y MQ.

(a) Muestra de pH (b) Muestra $AgNO_3$ (c) Muestra Ac_2Zn

Figura 3.2 – Muestras de pH y potencial reducción-oxidación.

(a) Muestra de conductividad eléctrica (b) Muestras incluídas con el sensor de conductividad (c) Muestra de colorimetría

Figura 3.3 – Muestras de conductividad eléctrica y colorimetría.

3.2. Parámetros utilizados para la caracterización.

Cuando se busca analizar una muestra de agua existen multitud de parámetros que pueden aportar información acerca de los componentes que se hallan en ella. En este punto se exponen los parámetros que, durante el desarrollo del proyecto, se han tenido en consideración.

3.2.1. Temperatura.

La temperatura es una magnitud escalar e intensiva que hace referencia a la energía de un sistema termodinámico, concretamente a la energía media de las moléculas que componen dicho sistema. La relación entre la temperatura del sistema y el movimiento o vibración de sus partículas es directamente proporcional.

Hay multitud de parámetros que varían en función de la temperatura, como por ejemplo la conductividad eléctrica o el oxígeno disuelto en un líquido. Por tanto, es habitual realizar compensaciones o especificar a qué temperatura se han realizado las medidas de dichas propiedades.

3.2.2. Turbidez.

Se define la turbidez como una medida del grado en el cual el agua pierde su transparencia debido a la presencia de partículas en suspensión. La turbidez se mide en NTU (Unidad Nefelométrica de Turbidez, *Nephelometric Turbidity Unit*), y su obtención se realiza midiendo la intensidad de la luz dispersada a 90 grados cuando un rayo de luz pasa a través de una muestra del agua a analizar.

Valores bajos de NTU corresponden a turbidez baja, mientras que valores elevados resultan en alta turbidez. Este parámetro se utiliza habitualmente para estimar la concentración de TSS (Total de Sólidos en Suspensión, *Total Suspended Solids*).

3.2.3. pH.

El pH es la magnitud referida al grado de acidez o alcalinidad (basicidad) propio de una disolución, e indica la concentración de iones H^+ en esta. Se puede calcular siguiendo la relación mostrada en la ecuación 3.1.

$$pH = -log_{10}[H^+] \tag{3.1}$$

Se consideran ácidas las disoluciones con un pH inferior a 7 y básicas si su pH es superior a esta cifra. Si el pH es exacamente 7 la disolución se considera neutra. El pH es un parámetro que mide la actividad de los protones, pero no de los electrones. Para el estudio de estos últimos se utiliza el potencial de oxidación - reducción.

3.2.4. Potencial RedOx.

El potencial RedOx u ORP (Potencial de Oxidación - Reducción, *Oxidation Reduction Potencial*) es un parámetro que permite obtener información sobre la actividad de los electrones, para su medida se recurre a un electrodo de referencia que trabaja a un potencial constante.

El ORP es positivo cuando se produce una oxidación y negativo cuando se produce una reducción. Como se puede comprobar, este es un parámetro análogo al pH, ya que el pH mide la actividad de protones, mientras que el ORP mide la de los electrones.

3.2.5. Conductividad Eléctrica.

Se denomina conductividad eléctrica (σ) a la medida de la capacidad de un material o sustancia de permitir el paso de corriente eléctrica.

En medios líquidos, esta capacidad viene determinada por las sales en disolución, por lo cual la medida de este parámetro permite obtener una idea de la cantidad de sales disueltas.

3.2.6. Oxígeno Disuelto.

El oxígeno disuelto o DO (Oxígeno Disuelto, *Dissolved Oxygen*) hace referencia a la cantidad de oxígeno libre que se encuentra en disolución en un medio, y es un parámetro fundamental para la vida de seres acuáticos, lo cual lo hace un parámetro de gran importancia a la hora de realizar análisis de aguas.

3.2.7. Espectro electromagnético.

El espectro electromagnético clasifica los tipos de ondas existentes en función de sus propiedades fundamentales, estas son frecuencia, velocidad, amplitud y longitud de onda, siendo este último parámetro el más utilizado para la clasificación.

En la figura 3.4 se pueden observar las diferentes clasificaciones que existen para las ondas en función de su longitud, además de un plano en detalle del espectro visible acompañado de los colores que identifica el ojo humano para cada región del espectro.

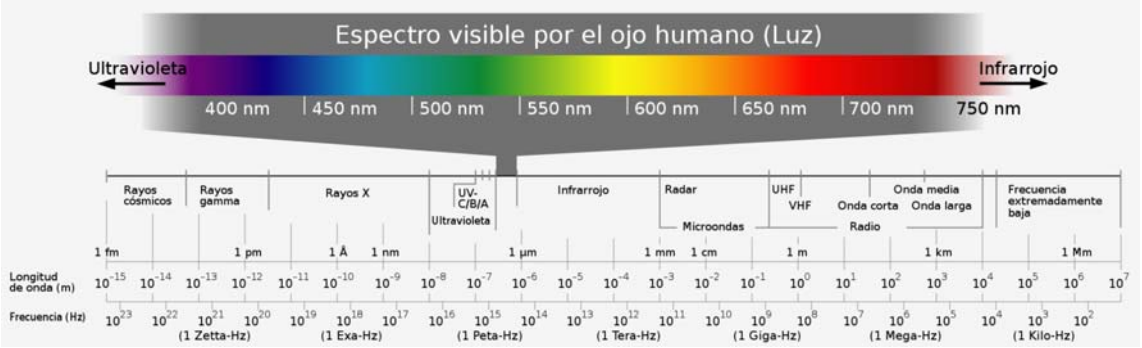


Figura 3.4 – Espectro electromagnético por Horst Frank [23].

Dependiendo del medio que se quiera analizar los rangos de longitud de onda variarán, en el estudio de aguas es habitual trabajar en el espectro UV cercano al visible, entre 200 y

350 nm, ya que esto permite detectar la presencia de sustancias orgánicas en disolución. Sin embargo, estudiar el espectro visible y parte del infrarrojo también puede aportar información útil.

3.2.7.1. Absorbancia.

La absorbancia, en espectrofotometría, define la capacidad de un medio de absorber luz, esto es, relaciona la intensidad de la luz incidente en el medio y la intensidad después de haberlo atravesado. En la imagen 3.5 se muestra esto gráficamente.

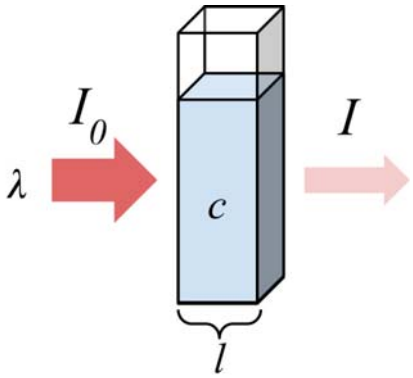


Figura 3.5 – Haz de luz atravesando una muestra [24].

Esta relación se puede definir matemáticamente mediante la ley de Beer-Lambert, mostrada en la ecuación 3.2.

$$A = -log_{10}(\frac{I}{I_0}) \tag{3.2}$$

Donde:

- *A*: la absorbancia en tanto por 1.
- I_0: intensidad de la luz que incide en la muestra.
- *I*: intensidad de la luz después de atravesar la muestra.

Debido a la relación que tiene la absorbancia con la luz hay que tener en cuenta que su valor varía en función de la longitud de onda que se esté considerando, por lo que la ecuación 3.2 es válida para una longitud de onda determinada. En la imagen 3.6 se muestra la variación de la absorbancia en una solución acuosa de permanganato de potasio a modo de ejemplo.

Conocer la absorbancia de una muestra puede ser de gran utilidad ya que hace posible detectar ciertos componentes químicos en una muestra además de su concentración en la misma.

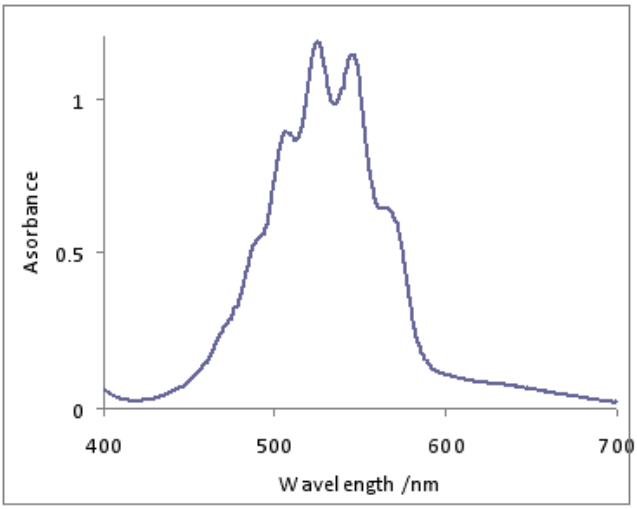


Figura 3.6 – Absorbancia de una solución acuosa de permanganato de potasio [25].

3.3. Aprendizaje automático.

El *Machine learning* o el aprendizaje automático es una rama de estudio dentro de la Inteligencia Artificial cuyo objetivo es el desarrollo de algoritmos, a partir de datos, con los que una máquina pueda aprender de manera automática. El principal atractivo de utilizar técnicas de aprendizaje automático es que con ellas se pueden representar complejos sistemas no lineales sin necesidad de definir matemáticamente el comportamiento de dichos sistemas.

El área del *Machine learning* se puede dividir en las 3 categorías mostradas a continuación:

- **Aprendizaje supervisado:** en este tipo de aprendizaje automático se requieren tanto los valores de entrada como los valores objetivo. El modelo, entrenado a partir de estos valores, se utilizará para predecir los valores objetivo a partir de valores de entrada nuevos. En este grupo se encuentran los modelos de clasificación y regresión.
- **Aprendizaje no supervisado:** este tipo de aprendizaje automático solo utiliza valores de entrada para el aprendizaje. Algunos ejemplos de este tipo de aprendizaje son el *clustering* o agrupamiento y la reducción de dimensionalidad.
- **Aprendizaje por refuerzo:** dentro de este grupo se encuentran los sistemas cuyo aprendizaje se lleva a cabo mediante recompensas y sanciones. En este caso no se trabaja con valores objetivo, sino que se aprende interactuando con el entorno. Cuando el algoritmo actúa correctamente recibe una recompensa, en caso contrario, es penalizado.

Para este proyecto se estudiará la posibilad de implementar técnicas de aprendizaje automático, concretamente de aprendizaje supervisado, con el fin de averiguar si es posible desarrollar un sensor virtual que sea capaz de reproducir un parámetro desconocido a partir de los datos obtenidos con los sensores que conformarán el proyecto.

El desarrollo de sensores virtuales ha ganado popularidad en los últimos años, ya que con ellos, a partir de parámetros de sencilla adquisición, se pueden producir otros de más

complicada obtención o cuya sensorización requiera de un desembolso económico mayor.

4 NORMAS Y REFERENCIAS

4.1. Disposiciones legales y normas aplicadas

Normativa establecida por la Escuela Universitaria Politécnica para la elaboración de los Trabajos de Fin de Grado (TFG) en las titulaciones de Grado en Ingeniería Electrónica Industrial y Automática y Grado en Ingeniería Eléctrica.

El estudio realizado en el presente proyecto tiene como objetivo el diseño de un prototipo, por lo que las normas seguidas para los controles en una EDAR no aplican en este caso.

4.2. Bibliografía

[1] MACKINNEY, W.; *Python for data analysis*, 1ª ed. California, O'Reilly Media, Inc, (2012).

[2] HAIMI, HENRI, ET AL. *Data-derived soft-sensors for biological wastewater treatment plants: An overview. Environmental Modelling Software*, 2013, vol. 47, p. 88-107.

[3] ABYANEH, HAMID ZARE. *Evaluation of multivariate linear regression and artificial neural networks in prediction of water quality parameters. Journal of Environmental Health Science and Engineering*, 2014, vol. 12, no 1, p. 40.

[4] CORNELISSEN, EMILE. *Prediction of wastewater treatment plants process performance parameters based on microbial communities using machine learning techniques*. 2018. Tesis Doctoral.

4.3. Software empleado

En esta sección se hace referencia a los programas utilizados en la realización del proyecto.

4.3.1. Software de diseño gráfico

- Autodesk Fusion 360 2019
- Autodesk AutoCAD 2020

4.3.2. Software de impresión 3D

- Ultimaker Cura 4.4

4.3.3. Software de programación

- Python 3
- Visual Studio Code 1.42

- Arduino IDE 1.8.10

4.3.4. Software de edición de texto

- Overleaf (Latex)

4.4. Otras referencias

[5] *Información del sensor de temperatura DS18B20.* [en línea], Datasheet, [Fecha de la consulta: 17 de Octubre de 2020]. Disponible en: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[6] *Información del fabricante sobre el sensor de turbidez.* [en línea], Web DFRobots, [Fecha de la consulta: 17 de Octubre de 2020]. Disponible en: https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189

[7] *Información del fabricante sobre el sensor de pH.* [en línea], Web DFRobots, [Fecha de la consulta: 17 de Octubre de 2020]. Disponible en: https://wiki.dfrobot.com/PH_meter_SKU__SEN0161_

[8] *Información del fabricante sobre el sensor de conductividad eléctrica.* [en línea], Web DFRobots, [Fecha de la consulta: 17 de Octubre de 2020]. Disponible en: https://wiki.dfrobot.com/Analog_EC_Meter_SKU_DFR0300

[9] *Información del fabricante sobre el sensor de potencial redox.* [en línea], Web DFRobots, [Fecha de la consulta: 17 de Octubre de 2020]. Disponible en: https://wiki.dfrobot.com/Analog_ORP_Meter_SKU_SEN0165_

[10] *Información del fabricante sobre el sensor de oxígeno disuelto.* [en línea], Web DFRobots, [Fecha de la consulta: 17 de Octubre de 2020]. Disponible en: https://wiki.dfrobot.com/Gravity__Analog_Dissolved_Oxygen_Sensor_SKU_SEN0237

[11] *Información del fabricante sobre la shield para Arduino.* [en línea], Web DFRobots, [Fecha de la consulta: 11 de Noviembre de 2020]. Disponible en: https://www.dfrobot.com/product-1009.html

[12] *Reducción del error al medir conductividad a diferentes temperaturas.* [en línea], Web Mettler Toledo, [Fecha de la consulta: 23 de Octubre de 2020]. Disponible en: https://www.mt.com/dam/MT-NA/pHCareCenter/Conductivity_Reduce_Common_Errors_WP.pdf

[13] *Guía técnica del dispositivo Smart Water.* [en línea], Web Libelium, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.libelium.com/downloads/documentation/smart_water_sensor_board.pdf

[14] *Guía técnica del dispositivo Smart Water Ions.* [en línea], Web Libelium, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.libelium.com/downloads/documentation/smart_water_ions_sensor_board.pdf

[15] *Información acerca de Waspmote Plug & Sense.* [en línea], Web Libelium, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.libelium.com/iot-products/plug-sense/

[16] *Datasheet del microespectrómetro C12880MA de Hamamatsu Photonics.* [en línea], Web de Hamamatsu, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.hamamatsu.com/resources/pdf/ssd/c12880ma_kacc1226e.pdf

[17] *Información sobre la breakout board del microespectrómetro para Arduino.* [en línea], Web de GroupGets, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://groupgets.com/manufacturers/hamamatsu-photonics/products/c12880ma-micro-spectrometer

[18] *Información sobre la placa Arduino UNO Rev 3.* [en línea], Web de Arduino, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://store.arduino.cc/arduino-uno-rev3

[19] *Información sobre la placa Tiva C de Texas Instruments.* [en línea], Web de Texas Instruments, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.ti.com/tool/EK-TM4C123GXL

[20] *Información sobre el módulo WiFi ESP-01.* [en línea], Datasheet, [Fecha de la consulta: 26 de Octubre de 2020]. Disponible en: https://saber.patagoniatec.com/wp-content/uploads/2019/09/esp8266-esp01-datasheet.pdf

[21] *Subir datos a un canal de ThingSpeak.* [en línea], Web hackster.io, [Fecha de la consulta: 28 de Octubre de 2020]. Disponible en: https://www.hackster.io/Dukee/thingspeak-get-post-c81d91

[22] *Coeficientes para conversión a longitud de onda del microespectrómetro.* [en línea], Web de GroupGets, [Fecha de la consulta: 30 de Octubre de 2020]. Disponible en: https://groupgets-files.s3.amazonaws.com/hamamatsu/uspectrometer/Cal-Data-42.pdf

4.5. Referencias gráficas

[23] *Espectro electromagnético.* [en línea], Wikipedia, [Fecha de la consulta: 30 de Agosto de 2020]. Disponible en: https://commons.wikimedia.org/wiki/File:Electromagnetic_spectrum-es.svg

[24] *Haz de luz atravesando una muestra.* [en línea], Lifeder, [Fecha de la consulta: 9 de Noviembre de 2020]. Disponible en: https://www.lifeder.com/ley-de-beer-lambert/

[25] *Absorbancia del permanganato de potasio.* [en línea], Wikiwand, [Fecha de la consulta: 9 de Noviembre de 2020]. Disponible en: https://www.wikiwand.com/en/Permanganate

[26] *Sensor Temperatura.* [en línea], Web de RS Components, [Fecha de la consulta: 4 de Noviembre de 2020]. Disponible en: https://es.rs-online.com/web/p/sensores-de-humedad-y-temperatura/1901709/

[27] *Sensor de Temperatura con encapsulado para líquidos.* [en línea], Web de DFRobots, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.dfrobot.com/product-689.html

[28] *Smart Water.* [en línea], Web de Libelium, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.libelium.com/wp-content/uploads/2020/09/SOLUTIONS-product-water.png

[29] *Smart Water Ions.* [en línea], Web de Libelium, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.libelium.com/wp-content/uploads/2020/09/SOLUTIONS-product-water-ions.png

[30] *C10082CA de Hamamatsu Photonics.* [en línea], Web de Hamamatsu Photonics, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.hamamatsu.com/eu/en/product/type/C10082CA/index.html

[31] *Ocean HDX de Ocean Optics.* [en línea], Web de Ocean Optics, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.oceaninsight.com/products/spectrometers/high-sensitivity/ocean-hdx/

[32] *C12880MA de Hamamatsu Photonics.* [en línea], Web de Hamamatsu Photonics, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.hamamatsu.com/eu/en/product/type/C12880MA/index.html

[33] *Breakout Board del microespectrómetro C12880MA para Arduino.* [en línea], Web de GroupGets, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://groupgets-files.s3.amazonaws.com/hamamatsu/uspectrometer/board-with-sensor-transparent-b%26f.png

[34] *Espectro característico de la luz de lámpara de mercurio* [en línea], Research Gate, [Fecha de la consulta: 8 de Noviembre de 2020]. Disponible en: https://www.researchgate.net/profile/Snehasish_Dutta_Gupta/publication/320644849/figure/fig14/AS:578664409636865@1514975754101/Spectral-outputs-of-the-various-light-sources.png

[35] *Espectro característico de la luz de xenón.* [en línea], Web Bioprocess Online, [Fecha de la consulta: 8 de Noviembre de 2020]. Disponible en: https://www.bioprocessonline.com/doc/light-sources-for-spectrophotometers-0001

[36] *Espectro electromagnético de diferentes fuentes de luz.* [en línea], Web de Sea Side Vision, [Fecha de la consulta: 8 de Noviembre de 2020]. Disponible en: https://seasidevision.com/2015/08/05/is-blue-light-harmful-to-our-eyes/

[37] *Arduino UNO Rev 3.* [en línea], Web de Arduino, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://store.arduino.cc/arduino-uno-rev3

[38] *Tiva C de Texas Instruments.* [en línea], Web de Texas Instruments, [Fecha de la consulta: 25 de Octubre de 2020]. Disponible en: https://www.ti.com/tool/EK-TM4C123GXL

[39] *Raspberry Pi 4 Model B.* [en línea], Web de Ebay, [Fecha de la consulta: 26 de Octubre de 2020]. Disponible en: https://www.ebay.es/itm/Raspberry-Pi-4-Model-B-with-4GB-RAM-2019-Model-/233276661035

[40] *Beaglebone black.* [en línea], Web de RS Components, [Fecha de la consulta: 26 de Octubre de 2020]. Disponible en: https://es.rs-online.com/web/p/kits-de-desarrollo-de-microcontroladores/1252411/

[41] *Módulo WiFi ESP-01.* [en línea], Wikipedia, [Fecha de la consulta: 26 de Octubre de 2020]. Disponible en: https://es.wikipedia.org/wiki/ESP8266#/media/Archivo:ESP-01.jpg

[42] *Plataforma de IoT ThingsBoard.* [en línea], Web de ThingsBoard, [Fecha de la consulta: 27 de Octubre de 2020]. Disponible en: https://thingsboard.io/smart-energy/

[43] *Plataforma IoT ThingSpeak de MathWorks.* [en línea], Web de MathWorks, [Fecha de la consulta: 27 de Octubre de 2020]. Disponible en: https://blogs.mathworks.com/loren/2016/03/23/explore-your-iot-data-with-thingspeak-and-matlab/

[44] *Pineado de la shield de expansión para Arduino.* [en línea], Web de DigiKey, [Fecha de la consulta: 11 de Noviembre de 2020]. Disponible en: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0265_Web.pdf

[45] *Vial de vidrio para microespectrómetro.* [en línea], Amazon, [Fecha de la consulta: 14 de Noviembre de 2020]. Disponible en: https://www.amazon.es/KAHEIGN-Transparente-Botellas-Contenedor-Laboratorio/dp/B088BMKSYZ/ref=sr_1_5?dchild=1&keywords=viales+cristal&qid=1603237647&sr=8-5

[46] *Vial de vidrio para microespectrómetro.* [en línea], Web de Tesla Electronic, [Fecha de la consulta: 14 de Noviembre de 2020]. Disponible en: https://www.teslaelectronic.com.pe/producto/diodo-led-blanco/

[47] *Relación voltaje - oxígeno disuelto.* [en línea], Web de DFRobot, [Fecha de la consulta: 31 de Octubre de 2020]. Disponible en: https://dfimg.dfrobot.com/nobody/wiki/c16250605ea3d282501f45e43f3cb7ad.png

[48] *Variación del voltaje de saturación del sensor de DO en función de la temperatura.* [en línea], Web de DFRobot, [Fecha de la consulta: 31 de Octubre de 2020]. Disponible en: https://dfimg.dfrobot.com/nobody/wiki/24b8745527bca423222e68fed18f28fe.png

[49] *Capuchón para el sensor de DO.* [en línea], Web de DFRobot, [Fecha de la consulta: 7 de Noviembre de 2020]. Disponible en: https://www.dfrobot.com/product-1739.html

5 DEFINICIONES Y ABREVIATURAS

- MQ: Milli Q, sistema de purificación de agua.
- DO: *Dissolved Oxygen* u Oxígeno Disuelto.
- NTU: *Nephelometric Turbidity Unit* o Unidades Nefelométricas de Turbidez.
- σ: conductividad eléctrica.
- ORP: *Oxidation Reduction Potencial* o Potencial RedOx.
- EC: *Electrical Conductivity* o Conductividad Eléctrica.
- TSS: *Total Suspended Solids* o Sólidos Totales en Suspensión.
- RedOx: Reducción - Oxidación.
- DAQ: Adquisición de datos.
- pip: Instalador de paquetes de python.
- PC: Ordenador.
- M: Molar.
- HCl: Ácido clorhídrico.
- IoT: *Internet of Things* o Internet de las cosas.
- UART: *Universal Asynchronous Receiver-Transmitter* o Transmisor-Receptor Asíncrono Universal.
- MQTT: *Message Queue Telemetry Transport* o Transporte de Telemetría de Cola de Mensajes.
- HTTP: *Hypertext Transfer Protocol* o Protocolo de Transferencia de Hipertexto.
- MLR: *Multiple Linear Regression* o Regresión lineal múltiple.
- ANN: *Artificial Neural Network* o Red Neuronal Artificial.
- MLP: *Multi-layer Perceptron* o Perceptrón multicapa.
- mA: mili Amperios.
- mS: mili Siemens.
- cm: centímetros.
- F.S: Full Scale.

- CSV: *Comma Separated Values* o Valores separados por comas.
- MSE: *Mean Squared Error* o Error cuadrático medio.
- MAE: *Mean Absolute Error* o Error absoluto medio
- RMSE: *Root Mean Square Error* o Raíz del error cuadrático medio

6 REQUISITOS DE DISEÑO

Durante el desarrollo del proyecto han de cumplirse ciertos requisitos, estos pueden clasificarse dentro de los siguientes puntos:

- Requisitos hardware.
- Requisitos software.

6.1. Requisitos hardware.

Los requisitos hardware a cumplir son los siguientes:

- El coste de los sensores utilizados deberá ser lo más reducido posible.
- El dispositivo de adquisición de los datos que envía los datos al PC deberá ser también de bajo coste.

6.2. Requisitos software.

- Para los códigos de programación se utilizarán preferiblemente plataformas software de código libre.
- Desarrollar un programa que permita muestrear lo más rápido posible dentro de las posibilidades hardware existentes.
- Todos los datos obtenidos durante el muestreo deben ser guardados en un histórico para su consulta en un futuro si fuese necesario.
- Los datos obtenidos se graficarán en tiempo real para facilitar al usuario la tarea de visualizarlos.
- Subir a la nube los resultados de tal forma que se puedan realizar consultas remotas y ver el estado de las diferentes variables en cada instante.

7 ANÁLISIS DE LAS SOLUCIONES

Para realizar el análisis de las diferentes soluciones que se pueden implementar es conveniente mencionar, de forma breve, los dispositivos que formarán parte del sistema a desarrollar. En la figura 7.1 se puede observar un esquema general de las partes que conformarán el proyecto.

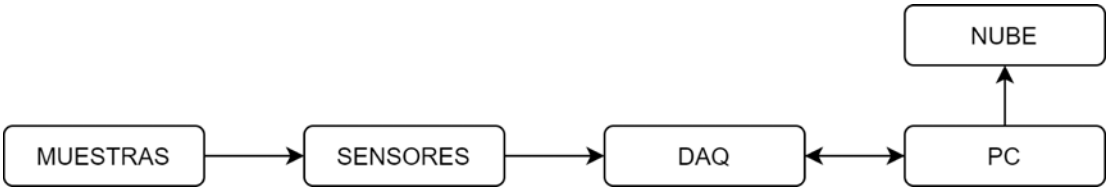


Figura 7.1 – Elementos físicos que constituyen el proyecto.

En este capítulo se estudiarán, siguiendo el esquema de la figura 7.1, las posibilidades que ofrece cada propuesta, además, para cada bloque se presentarán varias alternativas para posteriormente elegir la más conveniente.

Durante el desarrollo del capítulo se seguirá un orden lógico en lo que respecta a las fases del proyecto. Se comenzará por la selección de sensores que permitan cuantificar los parámetros deseados, siguiendo con la elección de un dispositivo que permita leer los valores que proporcionan dichos sensores. A continuación, se propondrán distintas herramientas software para la visualización, almacenado y estudio de los datos obtenidos. Finalmente los datos se subirán a la nube para permitir su consulta desde cualquier lugar con acceso a la red.

Tomando en cuenta esto, se ha decidido dividir el proyecto en las 7 etapas mostradas:

- Estudio del tipo de sistema para la realización del proyecto.
- Análisis de los diferentes sensores disponibles en el mercado.
- Sistemas de adquisición de datos.
- Comunicación.
- Selección del software de programación.
- Subida a la nube de los datos.
- Estudio de las diferentes técnicas que permitan desarrollar un sensor virtual.

7.1. Estudio del tipo de sistema para la realización del proyecto.

Para la realización de este proyecto se pueden dividir las diferentes soluciones en dos grupos, sistemas abiertos y sistemas cerrados. A continuación se explicará brevemente qué caracteriza de forma general a cada tipo de sistema.

7.1.1. Sistemas abiertos.

En los sistemas abiertos o dispositivos de hardware libre se conoce su funcionamiento y este se puede modificar fácilmente, además de realizar comunicación entre diferentes dispositivos de manera sencilla.

Para este tipo de dispositivos las especificaciones y esquemáticos han sido publicados, siendo posible acceder a ellos tanto gratuitamente como mediante diferentes métodos de pago.

Los costes suelen ser reducidos, sin embargo, en proyectos de alta complejidad que hagan uso de estos dispositivos los costes de desarrollo pueden llegar a ser elevados.

7.1.2. Sistemas propietarios.

Son aquellos sistemas cuyo código fuente, especificaciones o esquemáticos no son de acceso libre. Son diseñados y comercializados por terceros con el objetivo de ser utilizados de forma rápida y sencilla, evitando elevados costes de desarrollo.

El coste inicial suele ser superior a los sistemas abiertos, aunque se ahorra en costes de diseño y desarrollo.

7.2. Análisis de los diferentes sensores disponibles en el mercado.

Para medir los parámetros mencionados en la sección 3 se muestran algunas opciones de sistemas abiertos y cerrados.

7.2.1. Sensor de temperatura.

La temperatura es uno de los parámetros más comunmente medidos en el mundo, por lo que existen infinidad de opciones a la hora de seleccionar un sensor de este tipo.

Un ejemplo válido es el DS18B20 de *Maxim Integrated*, uno de los sensores de temperatura más populares en proyectos que hacen uso de dispositivos de hardware libre. En la imagen 7.2 se puede ver el aspecto del sensor en su versión con encapsulado TO-92.

Figura 7.2 – Sensor de temperatura DS18B20 con encapsulado TO-92 [26].

En este proyecto será necesario introducir el sensor en distintas disoluciones, para evitar que estos líquidos entren en contacto directo con el sensor existe una versión con encapsulado, pensado para realizar medidas de este tipo. En la imagen 7.3 se muestra el aspecto de dicho encapsulado.

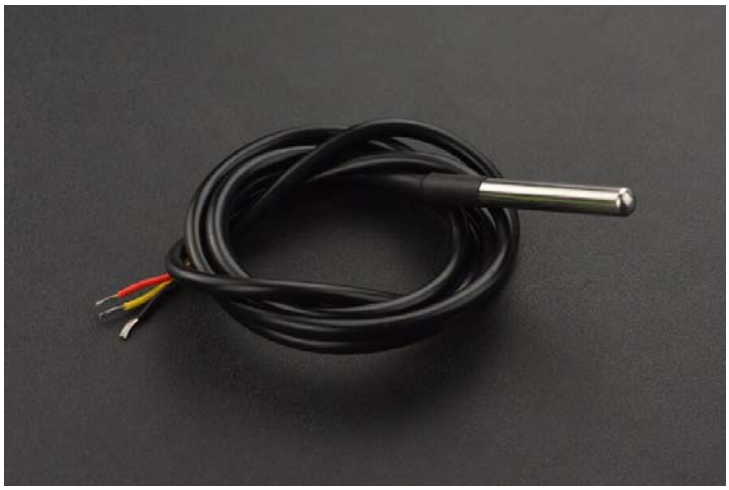


Figura 7.3 – Sensor DS18B20 en su versión con encapsulado para líquidos [27].

7.2.2. Sistema *Smart Water* [13].

Inicialmente se estudió la posibilidad de implementar un sistema propietario para realizar toda la adquisición de datos. El sistema estudiado fue *Smart Water* de Libelium, mostrado en la imagen 7.4.

Figura 7.4 – Dispositivo *Smart Water* de Libelium [28].

Este dispositivo, pensado para una configuración rápida y con protección IP65, ofrece la posibilidad de medir los siguientes parámetros:

- Temperatura.
- Turbidez.
- pH.
- Conductividad eléctrica.
- Potencial RedOx.

- Oxígeno Disuelto.

En lo referente a la comunicación el dispositivo soporta diferentes protocolos de comunicación industriales como ModBus, CAN bus, RS-232 o RS-485. Además de los anteriores, el dispositivo ofrece la posibilidad de realizar comunicaciones WiFi o 4G.

7.2.3. Sistema Smart Water Ions [14].

Otro sistema similar al anterior y de la misma compañía es el *Smart Water Ions*, su aspecto es el mismo que el sistema del apartado anterior, mostrado en la imagen 7.4.

Este dispositivo se divide en 3 familias: Single, Double y Pro. Para cada una de ellas las especificaciones son las mismas que las del sistema Smart Water ya que en ambos sistemas se parte de la plataforma *Waspmote Plug & Sense* [15].

La diferencia entre ambos reside en qué parámetros se miden. En el caso del sistema descrito en este apartado, el objetivo es detectar la presencia de iones en la muestra a analizar, además de medir la temperatura y el pH.

Los iones que se pueden detectar son diferentes en función de la familia seleccionada de este modelo, en la tabla 7.1 se pueden observar las posibilidad que ofrece cada versión.

Single	Double	Pro
Ca^{2+}	Br^{-}	NH_4^{+}
F^{-}	Cl^{-}	Br^{-}
BF_4^{-}	Cu^{2+}	Ca^{2+}
NO_3^{-}	I^{-}	Cl^{-}
	Ag^{+}	Cu^{2+}
		F^{-}
		I^{-}
		Li^{+}
		Mg^{2+}
		NO_3^{-}
		NO_2^{-}
		ClO_4^{-}
		K^{+}
		Ag^{+}
		Na^{+}

Tabla 7.1 – Listado de los iones que puede medir cada modelo.

Como se puede observar en la tabla 7.1 con la versión Pro se puede medir un número de iones mucho mayor en comparación con las otras dos opciones, además de que los sensores son más precisos (cada modelo incorpora sus propios sensores). Sin embargo, han de tenerse en cuenta los siguientes factores:

- El precio de la versión Pro es muy superior al resto de versiones.
- El máximo número de iones que se pueden medir simultáneamente es 4.

7.2.4. Sensores de DFRobots.

Dentro de las opciones de hardware libre presentes en el mercado para la caracterización de aguas destacan los sensores ofertados por DFRobots. De entre todos los dispositivos disponibles cabe destacar los siguientes:

- Sensor de turbidez.
- Sensor de pH.
- Sensor de conductividad eléctrica.
- Sensor de potencial RedOx.
- Sensor de oxígeno disuelto.

De entre las diversas ventajas que presentan esta serie de sensores cabe destacar su bajo coste si se comparan con los sistemas cerrados tratados en las secciones 7.2.2 y 7.2.3, su sencillez de conexionado y la facilidad de configuración mediante software.

Finalmente se seleccionaron estos sensores para el desarrollo del proyecto, principalmente debido a su bajo coste en comparación con los sistemas *Smart Water* estudiados.

7.2.5. Microespectrómetros.

Para el análisis de parámetros relacionados con la luz, como la absorbancia, los sensores empleados son fotodiodos, diodos que permiten o no el paso de corriente en función de la intensidad lumínica incidente. Sin embargo, estos diodos funcionan con una longitud de onda determinada y, por lo tanto, para medir un rango de longitudes de onda sería necesario un array de estos fotodiodos.

Los microespectrómetros se basan en este principio. Generalmente estos dispositivos son comercializados como sistemas propietarios listos para su uso sin necesidad de ser configurados, pero existen ciertas excepciones.

7.2.5.1. Microespectrómetros basados en sistemas propietarios.

Algunos productos pertenecientes a este grupo pueden ser el microespectrómetro C10082CA de *Hamamatsu Photonics* o el Ocean HDX de *Ocean Optics*, ambos mostrados en la imagen 7.5.

(a) C10082CA [30]

(b) Ocean HDX [31]

Figura 7.5 – Microespectrómetros basados en sistemas propietarios.

Estos modelos se adecúan a los requisitos del proyecto desde el punto de vista de rango espectral, ya que con ellos se puede analizar el espectro UV cercano al visible (entre 200 nm y 350 nm) además de todo el rango visible.

La principal desventaja de este tipo de dispositivos es su elevado coste, lo que hace necesario buscar otras alternativas.

7.2.5.2. Microespectrómetro basado en sistema abierto.

La alternativa a los microespectrómetros mencionados en el apartado anterior y que mejor se adapta a los requerimientos del proyecto es el modelo *C12880MA* de *Hamamatsu Photonics* mostrado en la imagen 7.6.

Figura 7.6 – Microespectrómetro C12880MA de Hamamatsu Photonics [32].

Entre las diferentes características que posee el dispositivo son especialmente destacables el rango espectral (340 nm - 850 nm) y su resolución de 15 nm.

Para hacer más sencillo su uso es conveniente adquirir una placa de adaptación o *breakout board* especialmente diseñada para el dispositivo. En la plataforma GroupGets se facilita una placa diseñada para la integración del dispositivo con Arduino UNO. En la figura 7.7 se muestra una imagen de dicha placa con el dispositivo ya incorporado.

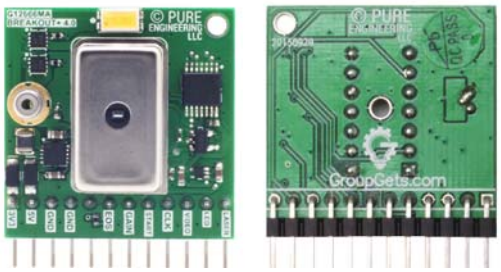


Figura 7.7 – *Breakout Board* del C12880MA para Arduino [33].

De entre las opciones estudiadas para la implementación de un microespectrómetro se decidió optar por el modelo C12880MA de *Hamamatsu Photonics*. De nuevo, el precio ha sido el parámetro decisivo a la hora de realizar la selección del modelo.

7.2.6. Fuentes de luz para el microespectrómetro.

Para poder realizar medidas con el microespectrómetro seleccionado será necesario diseñar un sistema que aísle al dispositivo de la luz ambiente, dicho sistema se desarrolla en la sección 8.1.7.3. El entorno constará de una fuente de luz cuyo espectro electromagnético será conocido. Es importante seleccionar una fuente de luz adecuada, de tal forma que se aproveche al máximo el rango espectral del microespectrómetro.

A continuación se muestran los diferentes tipos de fuentes de luz estudiadas junto con los espectros aproximados que caracterizan a cada tipo de luz.

- Lámpara halógena: muy presente en la parte alta del espectro visible pero sin presencia en las longitudes de onda bajas del mismo. En la imagen 7.8 se puede ver su espectro característico.

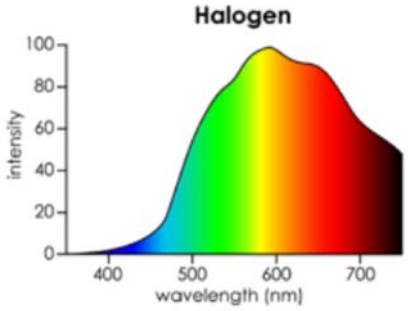


Figura 7.8 – Espectro característico de una lámpara halógena [36].

- Lámpara fluorescente: presenta picos de alta intensidad para ciertas longitudes de onda, puede resultar una opción interesante para comprobar si el sensor ofrece medidas correctas. En la imagen 7.9 se puede ver su espectro característico.

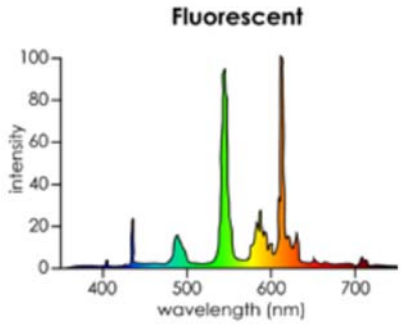


Figura 7.9 – Espectro característico de una lámpara fluorescente [36].

- Lámpara de xenón: la mejor opción para el espectro UV y visible, abarca dicho espectro por completo con una intensidad más o menos constante, exceptuando picos a 450 nm y 850 nm. En la imagen 7.10 se puede ver su espectro característico.

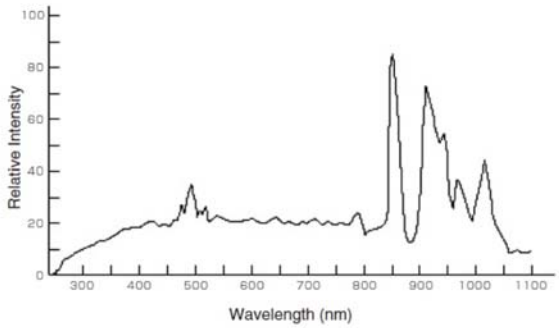


Figura 7.10 – Espectro característico de una lámpara xenón [35].

- Led blanco frío y led blanco cálido: ambos casos presentan un espectro similar pero el blanco frío ofrece un pico de mayor intensidad en longitudes de ondas bajas. En la imagen 7.11 se pueden ver sus espectros característicos.

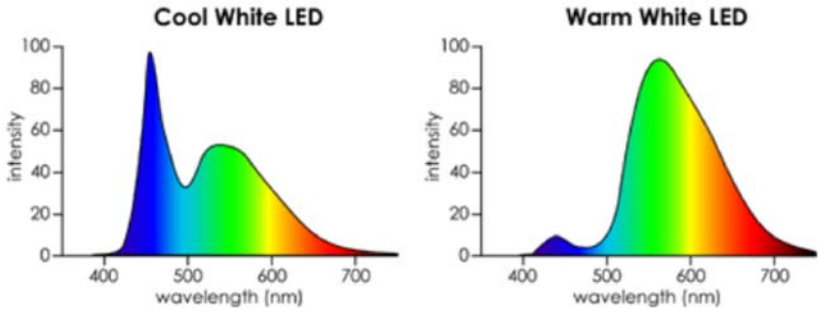


Figura 7.11 – Espectro característico de un led blanco frío (izquierda) y cálido (derecha) [36].

- Lámpara de mercurio: presenta un espectro similar a la lámpara de luz fluorescente, aunque con picos aún más marcados. En la imagen 7.12 se puede ver su espectro característico.

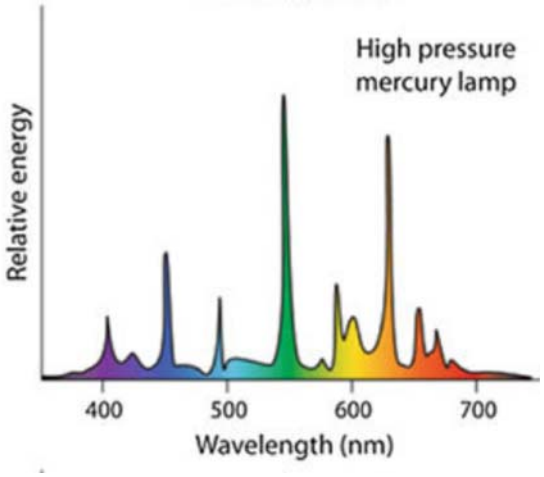


Figura 7.12 – Espectro característico de una lámpara de mercurio [34].

Después de observar las diferentes longitudes de onda que componen cada tipo de luz, se tuvieron en cuenta para una posible implementación la luz de xenón y el led de color blanco frío. Finalmente, se decidió utilizar como fuente de luz el led para obtener unos primeros resultados orientativos, ya que la otra opción requería de un sistema de alimentación externo. Esta es la solución que aporta un menor coste y además se puede alimentar de forma sencilla.

Si después del estudio se concluye que los resultados recogidos con el microespectrómetro son relevantes podría mejorarse la fuente de luz y utilizar una lámpara de xenón, que abarca todo el espectro electromagnético visible, tarea que el led blanco no llega a cumplir por completo.

7.3. Sistemas de adquisición de datos.

En esta sección se estudiarán diferentes plataformas de hardware libre que permitan la comunicación con los sensores seleccionados en la sección 7.2. El objetivo es seleccionar el dispositivo que mejor se adapte a las necesidades tanto de los sensores seleccionados como del proyecto en general.

7.3.1. Microcontroladores.

Los microcontroladores son circuitos integrados programables que, aunque contienen todos los elementos básicos de un ordenador, no están pensados para funcionar como uno (no utilizan un sistema operativo) además de que su capacidad de procesado es muy inferior si se compara con la de un PC.

7.3.1.1. Arduino.

Arduino es una plataforma de hardware libre que ofrece una amplia gama de placas de bajo coste basadas en diferentes microcontroladores. Así mismo, Arduino cuenta con un entorno

de desarrollo y lenguaje propios, siendo este bastante similar a lenguaje C. En la imagen 7.13 se puede ver el modelo seleccionado para esta categoría, un Arduino UNO.

Figura 7.13 – Arduino UNO Rev 3 [37].

Aunque existen muchos modelos, cada uno de ellos destinado para aplicaciones diferentes, en la tabla 7.2 se muestran algunas de las características del modelo de la imagen 7.13, el más comúnmente utilizado y de propósito general. Las características mostradas son las fundamentales a tener en cuenta de cara a la realización del proyecto.

Características del microcontrolador Arduino UNO R3	
Microcontrolador	ATmega328P
Tensión de alimentación	5/12 V
Nº entradas analógicas	6
Tensión de entradas analógicas	0-5 V
Resolucion del convertidor A/D	10 bits
Reloj	16 MHz

Tabla 7.2 – Características principales de la placa Arduino UNO.

En caso de que se necesitasen más entradas analógicas se podría optar por su modalidad MEGA, que cuenta con 16 entradas analógicas.

7.3.1.2. Tiva.

Una alternativa, sin salir de los microcontroladores, es la placa Tiva de *Texas Instruments*, mostrada en la imagen 7.14.

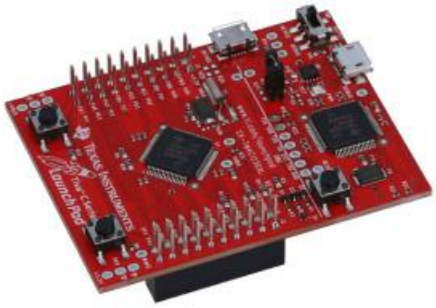


Figura 7.14 – Tiva C de *Texas Instruments* [38].

Si se compara esta placa con un Arduino UNO se puede ver que la Tiva es una placa bastante más potente. La Tiva, con el ARM Cortex-M4F presenta una arquitectura de 32 bits y un reloj de 80 MHz, mientras que Arduino UNO utiliza el ATmega328P con arquitectura de 8 bits y un reloj de 16 MHz.

Estas placas cuentan también con su propio entorno de programación, siendo este muy similar al de Arduino.

7.3.2. Microprocesadores.

Otra opción a tener en cuenta además de los microcontroladores son los microprocesador. Estos sistemas, a diferencia de los microcontroladores, necesitan que elementos como la memoria RAM o la ROM sean añadidos de forma externa pero ofrecen una capacidad de cómputo mucho mayor. Los microprocesadores soportan el empleo de un sistema operativo, con las ventajas que ello conlleva, como puede ser la ejecución de múltiples procesos simultáneamente.

7.3.2.1. Raspberry Pi.

Un ejemplo de dispositivo que incorpora un microprocesador es la Raspberberry Pi, mostrada en la imagen 7.15.

Figura 7.15 – Raspberry Pi 4 Model B [39].

Raspberry se inicia desde una tarjeta de memoria SD, en ella se puede alojar el sistema operativo, como puede ser Raspbian (una distribución Linux propia para Raspberry) o Windows. El dispositivo ofrece la posibilidad de conectarse de forma sencilla a otras placas, como puede ser Arduino, y realizar las tareas que desempeñaría un ordenador.

7.3.2.2. Beaglebone.

La placa Beaglebone, desarrollada por *Texas Instruments* en colaboración con otras empresas, es otra placa de hardware libre que hace uso de un microprocesador. En la imagen 7.16 se puede ver el modelo Beaglebone Black.

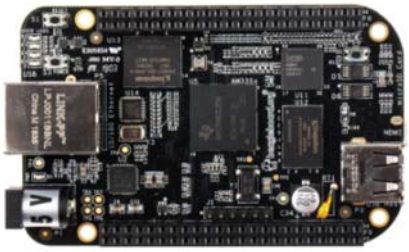


Figura 7.16 – Beaglebone Black [40].

Aunque esta placa es similar a la Raspberry Pi en algunos aspectos el campo de aplicación al que están orientados ambos dispositivos es muy diferente.

La Rasberry Pi es una buena elección para proyectos de tipo multimedia, sin embargo, la Beaglebone destaca en aplicaciones de IoT (*Internet of Things* o Internet de las cosas). Su gran número de pines permiten controlar diferentes sensores y actuadores además de realizar tareas de comunicación que impliquen el uso de Internet.

Para seleccionar un dispositivo de entre las opciones estudiadas se ha de tener en cuenta que de los 7 sensores seleccionados 6 son análogicos y 1 digital, por lo que el dispositivo debe soportar almenos 6 entradas analógicas. Las posibilidades que se barajaron principalmente son:

- Sistema de adquisición con Beaglebone.
- Sistema de adquisición con Arduino + módulo WiFi externo.
- Sistema de adquisición con Arduino + PC.

Finalmente se decidió utilizar una placa Arduino, concretamente el modelo Arduino UNO. En la decisión se ha tenido en cuenta tanto la compatibilidad del dispositivo con los sensores, como el efecto que tiene el uso de dicha placa sobre etapas posteriores.

En lo referente a la comunicación de los datos para su procesado la sección 7.4 trata diferentes posibilidades y estudia qué opción puede ofrecer mejores resultados.

7.4. Comunicación.

En esta sección se estudian (partiendo de las decisiones tomadas en el punto 7.3) diferentes posibilidades para la comunicación de los datos. A continuación se muestra una lista con las opciones estudiadas:

- Módulo WiFi compatible con Arduino.
- Comunicación con PC por puerto serie.

7.4.1. Módulo WiFi ESP-01 para Arduino [20]

Con el módulo ESP-01 de la imagen 7.17 se puede dotar de conectividad WiFi a la placa Arduino. El módulo hace uso de un microprocesador de bajo coste, el ESP8266.

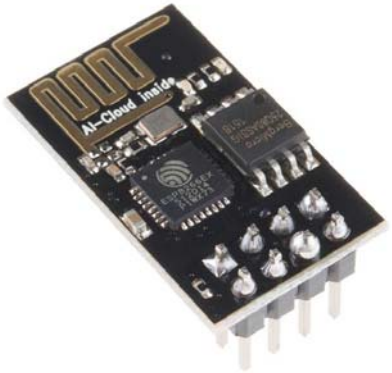


Figura 7.17 – Módulo WiFi ESP-01 compatible con Arduino [41].

Aunque el módulo ESP-01 está diseñado para ser utilizado con placas Arduino su alimentación resulta problemática si se realiza directamente con ellas. En la tabla 7.3 se muestran algunos parámetros a tener en cuenta para su alimentación.

Parámetros de alimentación del ESP-01	
Tensión de alimentación máxima	3,6 V
Tensión de alimentación recomendada	3,3 V
Tensión de alimentación mínima	3,0 V
Consumo medio de corriente	80 mA
Consumo máximo de corriente	170 mA

Tabla 7.3 – Tensión y corriente de alimentación del ESP-01.

Arduino UNO incorpora un regulador interno para generar tensiones de 3,3V, cuya corriente máxima de salida es 150 mA. Si nos fijamos en la tabla 7.3 el módulo ESP-01 puede llegar a requerir una corriente de 170 mA. Por tanto, aunque el módulo podría alimentarse directamente con Arduino, es recomendable la utilización de una fuente de alimentación externa.

Esto hace que la solución propuesta en el punto 7.4.2 cobre ventaja sobre la utilización de

este módulo WiFi.

7.4.2. Comunicación serial entre Arduino y PC.

La alternativa planteada a la utilización del módulo ESP-01 es la de comunicar la placa Arduino con un ordenador mediante puerto serie. Los datos captados por los sensores pasarían a Arduino y estos a su vez serían envíados al PC, donde se pueden almacenar en un fichero, graficar y subir a la nube.

Aunque no es necesario entender cómo funciona el protocolo de comunicación serie utilizado, sí es importante tener en cuenta que la comunicación por USB utiliza el protocolo UART (*Universal Asynchronous Receiver-Transmitter*). Esto quiere decir que los pines digitales 0 (RX) y 1 (TX), que están conectados al puerto USB mediante un convertidor RS-232/USB y que por tanto se utilizan en la comunicación, deben dejarse libres para realizar exclusivamente esta tarea y evitar conflictos en la transmisión.

Finalmente la solución adoptada fue esta, ya que el uso de un ordenador facilita la tarea de procesar los datos.

7.5. Selección del software de programación.

En esta sección se estudiarán diferentes entornos y lenguajes de programación que permitan realizar las siguientes tareas:

- Comunicación con la tarjeta Arduino.
- Guardado de los datos en un fichero para futuras consultas.
- Graficar los datos en tiempo real.
- Subir los datos a la nube para poder realizar consultas remotas.

A continuación se muestran las 3 opciones que se han tenido en cuenta.

7.5.1. Python.

Python es un lenguaje de programación interpretado que ha ido ganando popularidad en los últimos años. Principalmente debido a los siguientes factores:

- Es relativamente fácil aprender a usarlo, sobretodo si ya se conocen otros lenguajes de programación como C.
- Ofrece soluciones sencillas para tareas complejas gracias a que cuenta con un gran número de librerías específicas.
- Ha ido ganando relevancia en el campo de la inteligencia artificial gracias a librerías como *SKLearn* o *TensorFlow*.

- Su uso también está muy extendido en el ámbito del *Big Data* y se está comenzando a usar para el análisis de datos, sustituyendo a otros entornos como por ejemplo Matlab.

Además de que Python puede resultar útil en el desarrollo de este proyecto es un lenguaje que puede resultar útil en un futuro.

7.5.2. Matlab.

Matlab es un software de cálculo númerico que implementa un entorno de desarrollo integrado y un lenguaje propio de programación.

Al igual que Python, Matlab ofrece una serie de herramientas que facilitan el análisis y visualización de grandes volúmenes de datos.

7.5.3. Labview.

Labview es un entorno gráfico de programación desarrollado por *National Instruments* destinado al desarrollo de sistemas de control, visualización de datos o desarrollo de algoritmos para análisis de datos.

Este software, al igual que Matlab, se utiliza en el Grado de Ingeniería Electrónica Industrial y Automática.

Finalmente se decidió utilizar Python como lenguaje de programación, principalmente debido a que su uso es gratuito (a diferencia de Matlab y Labview que necesitan licencias) y a que, aunque no se utiliza en ninguna asignatura del Grado de Ingeniería Electrónica Industrial y Automática, su aprendizaje resulta relativamente sencillo si previamente se tienen nociones de programación en C.

7.6. Subida a la nube de los datos.

En este apartado se estudiarán diferentes plataformas de IoT que permitan subir datos a la nube y visualizarlos. Actualmente, en el mercado existen una gran variedad de servicios de este tipo, sin embargo, se estudiaron especialmente dos opciones:

- La plataforma *ThingsBoard*.
- La plataforma *ThingSpeak* desarrollada por *MathWorks*.

7.6.1. ThingsBoard.

Esta es una plataforma IoT de código abierto para recolección, procesado y visualización de datos con la que se pueden crear aplicaciones como la mostrada en la imagen 7.18.

Figura 7.18 – Ejemplo de un proyecto en ThingsBoard [42].

ThingsBoard soporta diferentes protocolos estándar de comunicación industrial para aplicaciones de este tipo, como pueden ser HTTP (*Hypertext Transfer Protocol* o Protocolo de Transferencia de Hipertexto) o MQTT (*Message Queue Telemetry Transport* o Transporte de Telemetría de Cola de Mensajes). Además, uno puede comunicarse fácilmente con la nube utilizando Python, gracias a que existen librerías específicas para ello.

7.6.2. ThingSpeak.

ThingSpeak es una plataforma de servicios IoT desarrollada por *MathWorks* que permite visualizar y analizar flujos de datos en tiempo real. Esta plataforma hace uso de algunas de las herramientas del software Matlab para el tratamiento de los datos y su interfaz, aunque sencilla, ofrece la información necesaria acerca de los datos. En la imagen 7.19 se muestra un ejemplo de una aplicación sencilla.

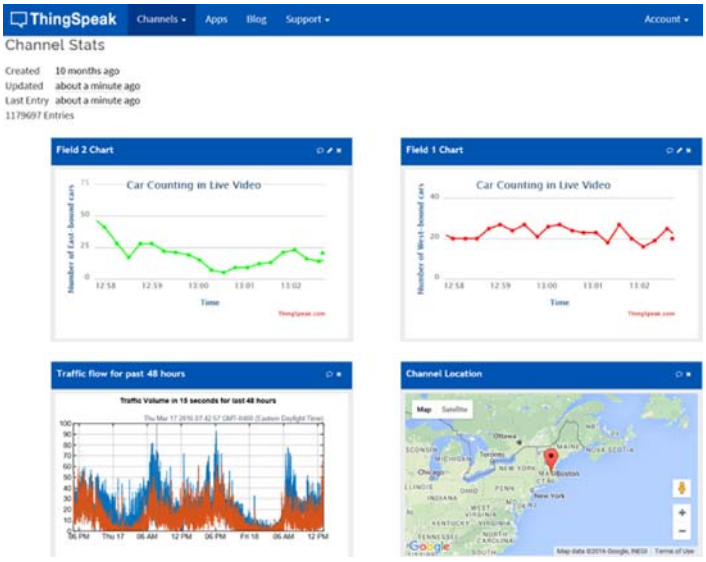


Figura 7.19 – Ejemplo de un proyecto en Thingspeak [43].

La API de *ThingSpeak* permite realizar comunicaciones de manera fácil entre diferentes plataformas. Si se decide usar Matlab o Simulink esta tarea se vuelve aún más sencilla, ya que existen Toolboxes que facilitan el trabajo, sin embargo, también es posible comunicarse con su nube utilizando otras vías, como por ejemplo, un script en Python.

La plataforma también permite su uso de forma gratuita, aunque con ciertas limitaciones, que se verán en la sección 8.4.2.5. Además, al igual que *ThingsBoard*, la comunicación puede realizarse mediante protocolos HTTP o MQTT (añadido recientemente).

Finalmente decidió utilizarse *ThingSpeak* como servicio IoT ya que la programación necesaria para realizar la comunicación de los datos con la nube resulta más sencilla que en el caso de utilizar *ThingsBoard.*

7.7. Estudio de las diferentes técnicas que permitan desarrollar un sensor virtual.

En la sección 3.3 de antecedentes se explicaron brevemente algunos conceptos generales sobre aprendizaje automático y cómo pueden resultar útiles en el diseño de sensores virtuales. En esta sección se estudirán algunas de las técnicas más utilizadas para problemas de regresión, que es el campo de interés de este proyecto.

Para comprender mejor el objetivo de este estudio se explica brevemente qué se pretende conseguir con el desarrollo de este tipo de sensores. En la imagen 7.20 se puede ver un diagrama que ilustra el concepto general.

Figura 7.20 – Diagrama explicativo del sensor virtual.

Como se puede ver, el sensor virtual actuaría como una caja negra o caja gris, cuyo funcionamiento interno se desconoce total o parcialmente. Las variables de entrada son los datos conocidos y que nos proporcionan los sensores estudiados en este proyecto, mientras que las variables de salida se corresponden con los parámetros de interés. El modelo por tanto, se compone de un conjunto de algoritmos o normas, que describen la relación entre los parámetros de entrada y de salida.

Actualmente existen multitud de técnicas con las que se pueden obtener modelos regresivos, por lo tanto, para reducir el rango de posibilidades y no extender excesivamente el proceso de selección de técnicas, se tomarán en consideración estudios anteriores sobre el desarrollo de sensores virtuales para caracterización de aguas. Las posibilidades aquí consideradas se basan, principalmente, en los estudios [2] y [3].

Teniendo esto en cuenta, se exponen los puntos más importantes a considerar a la hora de desarrollar el sensor virtual, además de las técnicas regresivas más comúnmente utilizadas:

- **Adquisición de datos:** constituye una parte fundamental del proceso de diseño, la aproximación del modelo será mejor si el volumen de datos es elevado.

- **Preprocesado de los datos de entrada:** este paso, aunque no es extrictamente necesario y depende de los datos con los que se trabaje, se recomienda aplicar siempre que se trabaje con este tipo de técnicas. Aquí se incluyen tareas como la normalización de las variables de entrada, la exclusión de datos redundantes o interpolación para estimar datos perdidos durante la adquisición.

- **Diseño del modelo:** después de procesar los datos de entrada es momento de seleccionar la técnica para obtener el modelo. No existe un curso de acción determinado que garantice el mejor resultado posible, por lo que generalmente se comprueba qué técnica es la más apropiada de manera experimental. De todos modos, inicialmente se recomienda implementar técnicas sencillas para obtener los primeros resultados. Siguiendo esto, las técnicas consideradas han sido:

  ◇ **Regresión lineal múltiple (MLR):** esta técnica es una de las más simples que se puede implementar, para construír el modelo asume que existe una relación lineal entre las entradas y las salidas. La técnica utiliza mínimos cuadrados ordinarios para realizar la aproximación del modelo lineal, siendo su mayor desventaja el hecho de que considere previamente que existe una relación lineal entre variables.

  ◇ **Redes neuronales artificiales (ANN):** las redes neuronales se componen de neuronas distribuidas en diferentes capas y conectadas entre sí. Los datos se transforman de forma no lineal en las neuronas mediante la función de activación asociada, pasando el resultado a otras neuronas. La relación entre entradas y salidas se ajusta otorgando pesos específicos a cada neurona, de tal forma que se minimice lo máximo posible el error entre las salidas de la red y las salidas objetivo o *target*.

  ◇ **Lógica difusa:** además de las redes neuronales existen otras técnicas utilizadas para problemas de regresión en el estudio de aguas, una de ellas es la lógica difusa. Este tipo técnicas siguen un razonamiento similar al de un humano, en el que para una entrada analógica cuyos valores varían entre 0 y 1 se asigna un porcentaje de pertenencia a diferentes grupos previamente definidos. Aunque existen menos estudios en los que se hayan implementado este tipo de técnicas se ha comprobado que pueden ofrecer resultados interesantes en ciertos casos.

En la mayoría de estudios de caracterización de aguas residuales se determina que no es habitual que exista una relación lineal entre los parámetros de entrada y salida, por lo que finalmente se decidió optar por la opción de las redes neuronales, concretamente una MLP (*Multi-layer Perceptron* o Perceptrón Multicapa), el tipo más simple de red. El motivo de descartar la lógica difusa se debe a que existen un mayor número de estudios que apoyan la utilización de redes neuronales en el campo de la caracterización de aguas.

Otra de las razones por las que se decidió seleccionar una red neuronal ha sido debido al lenguaje de programación utilizado en el proyecto (Python). Actualmente existen multitud de librerías específicas para el desarrollo de redes neuronales en Python, algunos ejemplos son *Tensorflow*, *Scikit-Learn*, *PyTorch*...

Finalmente, es importante mencionar que debido a problemas con la disposición de muestras cuyos parámetros fuesen conocidos mediante análisis en laboratorio, no se pudo realizar la implementación del sensor virtual. De todos modos, en la sección 8.7 se explicarán algunos conceptos y puntos a tener en cuenta para su posible implementación en un futuro.

8 RESULTADOS FINALES

Estudiadas las alternativas de diseño existentes, se procede a presentar la solución adoptada. Para su explicación, el apartado será dividido en varias etapas con el fin de establecer un orden de presentación para los elementos que conforman el sistema.

A continuación se presentan las etapas mencionadas:

- Sensores seleccionados y acondicionamiento.
- *Shield* de expansión para Arduino.
- Montaje del sistema de adquisición de datos.
- Programación software.
- Técnicas utilizadas para el desarrollo del sensor virtual.
- Resultados obtenidos y conclusiones.

8.1. Sensores seleccionados y acondicionamiento.

De los sensores presentados en la sección 7 de análisis de resultados se han seleccionado finalmente un total de 7 sensores, los cuales se muestran en la tabla 8.1, junto al parámetro que se medirá con cada uno de ellos.

Sensor	Parámetro
Sensor DS18B20 TO-92 con encapsulado para líquidos	Temperatura en ºC
Sensor de turbidez de *DFROBOTS*	Turbidez en NTU
Sensor de pH de *DFROBOTS*	pH
Sensor de EC de *DFROBOTS*	Conductividad eléctrica en mS/cm
Sensor de ORP de *DFROBOTS*	Potencial de reducción oxidación en mV
Sensor de DO de *DFROBOTS*	Oxígeno disuelto en mg/L
Microespectrómetro C12880MA de *Hamamatsu*	Absorbancia en el espectro visible

Tabla 8.1 – Tabla con los sensores seleccionados para el desarrollo del proyecto.

En los siguientes apartados se mostrarán las características y posibilidades que ofrecen cada uno de los sensores de la tabla 8.1. En el caso del microespectrómetro también se explicará cómo configurarlo para que funcione correctamente.

8.1.1. Sensor de temperatura.

El sensor de temperatura DS18B20 fabricado por *Maxim Integrated* que se puede ver en la imagen 7.3 es un sensor de temperatura acondicionado para realizar medidas en líquidos. En la tabla 8.2 se muestran las características de este sensor.

Sensor de temperatura DS18B20	
Tensión de alimentación	3,0V - 5,5V
Rango de temperatura efectivo	-55ºC - 125ºC
Precisión entre -10ºC y 85ºC	$\pm$ 0,5ºC
Bus de comunicación	1-Wire
Resolución configurable	9 - 10 - 11 - 12 bits

Tabla 8.2 – Tabla con las características del sensor de temperatura.

De las características mencionadas se realizará una explicación más detallada tanto del protocolo de comunicación como de los modos de resolución disponibles.

- **Protocolo 1-Wire.**

  Este protocolo serie, desarrollado por *Dallas Semiconductor*, hace uso de una única línea de datos para la comunicación. Por tanto, se utiliza una línea de alimentación, línea de GND y línea de datos. También existe una modalidad de conexión con 2 líneas denominada “parásito”, en la que el dispositivo se alimenta a partir de la línea de datos, pero esto se tratará más en profundidad en la sección 8.3.

  Mediante el uso de 1-Wire los valores analógicos recogidos por el sensor se transmiten de manera digital, de este modo, la temperatura puede leerse con un pin digital de Arduino.

  Este protocolo está basado en una serie de temporizaciones de la señal entre receptor y emisor. No se entrará en cómo funciona internamente este protocolo, ya que existen librerías que facilitan el uso del mismo sin necesidad de conocer a fondo su funcionamiento.

  Debido a este sistema de temporizaciones surge una desventaja, el tiempo de adquisición o conversión. Este tiempo es resultado de la suma de todas las temporizaciones internas que forman parte del protocolo y su valor depende de la resolución de medida seleccionada.

- **Resolución.**

  Como se puede ver la resolución del DS18B20 es configurable, siendo la resolución de 12 bits el modo por defecto. Aunque el modo de 12 bits es el que proporciona la mayor resolución hay que tener en cuenta que este modo implica un mayor tiempo de conversión.

Como ya se explicó, el bus 1-Wire se basa en una serie de temporizaciones que hacen posible la transmisión de un valor analógico mediante señales digitales. La suma de estas temporizaciones conforman el tiempo de conversión, esto es, el tiempo que se tarda en obtener una medida de temperatura. Los tiempos de conversión, junto con la variación de resolución en temperatura se muestran en la tabla 8.3.

Resolución (bit)	Resolución (ºC)	Tiempo de conversión
9-bit	0,5ºC	93,75 ms
10-bit	0,25ºC	187,5 ms
11-bit	0,125ºC	375 ms
12-bit	0,0625ºC	750 ms

Tabla 8.3 – Tabla con los tiempos de adquisición para cada resolución.

Para este proyecto se utiliza la resolución por defecto de 12 bits, ya que ese tiempo de conversión no supondrá un problema a la hora de realizar el muestreo. La resolución será por tanto de 0,0625ºC, sin embargo, esto no quiere decir que esa sea su precisión. En la hoja de características del fabricante se proporciona el error medio para distintas temperaturas aunque para este proyecto es suficiente con saber que la precisión es superior a $\pm$ 0,5ºC.

- El sensor también posee una EEPROM interna, sin embargo, en este proyecto no se hará uso de ella.

8.1.2. Sensor de turbidez.

El sensor de turbidez de *DFROBOTS* que se puede ver en la imagen 8.1 necesita una placa de acondicionamiento para su conexionado con Arduino. A la salida de dicha placa el sensor presenta las características mostradas en la tabla 8.4.

Figura 8.1 – Aspecto del sensor de turbidez junto con su placa de acondicionamiento [6].

Sensor de turbidez de DFROBOTS	
Tensión de alimentación	5V
Corriente máxima de operación	40mA
Tiempo de respuesta	$<$ 500ms
Modos de funcionamiento	Analógico y Digital
Tensión de salida en el modo analógico	0V - 4,5V
Rango operable de temperaturas	5ºC - 90ºC

Tabla 8.4 – Tabla con las características del sensor de turbidez.

Aunque la placa incorpora un interruptor para cambiar entre los modos digital y analógico para este proyecto se utiliza el modo analógico. El umbral del modo digital se modificaría ajustando el potenciómetro que incorpora la propia placa.

Además, aunque se menciona que la tensión máxima de salida debería ser 4,5V esto no es así, se comprobó experimentalmente que la tensión máxima ronda los 4,2V.

8.1.3. Sensor de pH.

El sensor de pH de *DFROBOTS* que se puede ver en la imagen 8.2 presenta las características mostradas en la tabla 8.5.

Figura 8.2 – Aspecto del sensor de pH [7].

Sensor de pH de DFROBOTS	
Tensión de alimentación	5V
Rango de medida	0 - 14 pH
Rango operable de temperatura	0ºC - 60ºC
Precisión	$\pm$ 0,1pH (25ºC)
Tiempo de respuesta	$<$ 1min
Voltaje de salida	Ajustable mediante potenciómetro

Tabla 8.5 – Tabla con las características del sensor de pH.

Como se puede ver el tiempo de respuesta es inferior a 1 minuto, esto hace referencia al tiempo que tarda en estabilizarse la medida de pH cuando se introduce en una muestra. Además de esto, un detalle a tener en cuenta es que la placa de acondicionamiento de este sensor incorpora un potenciómetro mediante el que se puede modificar el *offset* del dispositivo.

Esto se consigue ajustando la ganancia de una de las etapas de amplificacion, en el anexo 11.2 se puede ver el proceso más en detalle.

8.1.4. Sensor de conductividad eléctrica.

El sensor de conductividad eléctrica de *DFROBOTS* que se puede ver en la imagen 8.3 presenta las características mostradas en la tabla 8.6.

Figura 8.3 – Aspecto del sensor de conductividad eléctrica [8].

Sensor de EC de DFROBOTS	
Tensión de alimentación	5V
Rango de medida	1ms/cm - 20ms/cm
Rango operable de temperatura	5ºC - 40ºC
Precisión	$< \pm$ 10 % F.S
Constante del electrodo	0,995

Tabla 8.6 – Tabla con las características del sensor de conductividad eléctrica.

La constante del eléctrodo varía ligeramente debido al proceso de fabricación del mismo, sin embargo, el fabricante asegura que su valor será próximo a 1.

8.1.5. Sensor de potencial redox.

El sensor de potencial redox de *DFROBOTS* que se puede ver en la imagen 8.4 presenta las características mostradas en la tabla 8.7.

Figura 8.4 – Aspecto del sensor de potencial redox [9].

Sensor de ORP de DFROBOTS	
Tensión de alimentación	5V
Rango de medida	-2000mV - 2000mV

Rango operable de temperatura	5ºC - 70ºC
Precisión	$\pm$ 10mV (25ºC)
Tiempo de respuesta	$\leq$ 20s
Tensión de salida	0V - 4V

Tabla 8.7 – Tabla con las características del sensor de potencial redox.

Es importante mencionar que el electrodo que incorpora el sensor puede pasivizarse, si esto sucede deberá procederse a reactivarlo. En la sección 23.1.1 se detalla el proceso a seguir para la reactivación.

8.1.6. Sensor de oxígeno disuelto.

El sensor de oxígeno disuelto de *DFROBOTS* que se puede ver en la imagen 8.5 presenta las características mostradas en la tabla 8.8.

Figura 8.5 – Aspecto del sensor de oxígeno disuelto junto con la placa de acondicionamiento [10].

Sensor de DO de DFROBOTS	
Tensión de salida	0V - 3V
Rango de medida	0 - 20mg/L
Rango operable de temperatura	0ºC - 40ºC
Tiempo de respuesta	90s
Placa de acondicionamiento	
Tensión de alimentación	3,3V - 5,5V
Tensión de salida	0V - 3V

Tabla 8.8 – Tabla con las características del sensor de oxígeno disuelto.

Este sensor, al igual que el de ORP, necesita de mantenimiento además de una solución de NaOH para su correcto funcionamiento. En la sección 23.1.2 se detallan ambos aspectos.

8.1.7. Microespectrómetro.

De los microespectrómetros estudiados el que finalmente se seleccionó fue el C12880MA de *Hamamatsu*, como ya se comentó en 7.2.5.2.

8.1.7.1. Características y restricciones temporales.

Para la utilización del dispositivo se han tomado en consideración 3 factores:

- **Valores de longitud de onda que detecta**: el sensor posee 288 píxeles, cada uno de ellos detecta una longitud de onda. Para conocer estas longitudes de onda es necesario realizar una serie de cálculos. Estos se detallan en el anexo 11.6.
- **Frecuencia mínima de reloj**: el sensor necesita una señal de reloj para funcionar. Esta debe tener una frecuencia superior a 200 KHz, esto es, un período de menos de 5 microsegundos.
- **Tiempos a cumplir para su correcto funcionamiento**: para entender el funcionamiento del sensor es necesario conocer las señales que regulan el comportamiento del mismo, estas se muestran en la tabla 8.9.

Señales internas del microespectrómetro	
Entradas	
CLK	Señal de reloj
ST	Pulso de *Start* para comenzar adquisición
Salidas	
TRG	Señal de pulsos para la adquisición
EOS	Pulso de fin de adquisición
VIDEO	Salida del sensor

Tabla 8.9 – Tabla con las señales internas que controlan el microespectrómetro.

Además de las señales, en la hoja de características [16] también se proporcionan los tiempos utilizados para configurarlo. Estos tiempos se pueden ver en la tabla 8.10.

Tiempos a cumplir		
Parámetro	**Símbolo**	**Nº mínimo de ciclos de CLK**
Tiempo de ciclo	tpi	381
Tiempo ST nivel alto	thp	6
Tiempo ST nivel bajo	tlp	375

Tabla 8.10 – Tabla con los tiempos para la configuración del microespectrómetro.

El tiempo de ciclo (*tpi*) se obtiene a partir de la suma de los tiempos *thp* y *tlp*.

Además de estos, existe un 4º tiempo encargado de ajustar la tensión máxima de salida

del sensor, como se puede ver en la imagen 8.6.

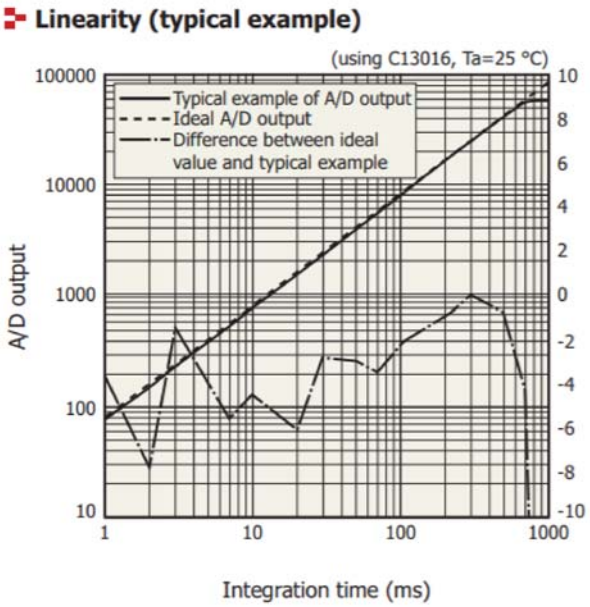
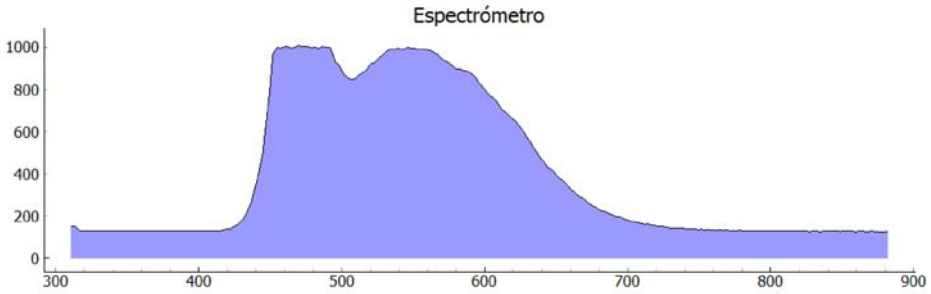
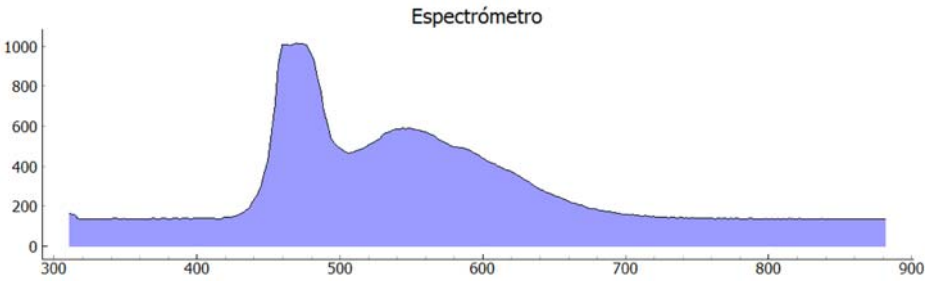


Figura 8.6 – Variación de la tensión de salida en función del tiempo de integración [16].

A este tiempo se le denomina tiempo de integración y resulta de gran utilidad para casos en los que el espectro electromagnético medido posee picos muy altos a determinadas longitudes de onda y muy bajos a otras. En la imagen 8.7 se muestra el caso mencionado para dos tiempos de integración distintos, 800 microsegundos en la imagen 8.7(a) y 300 microsegundos en la imagen 8.7(b). Además de esto, el ajuste del tiempo de integración se realiza en función de la intensidad lumínica de la fuente de luz.



(a) Tiempo de integración de 800 us.



(b) Tiempo de integración de 300 us.

Figura 8.7 – Efectos de la variación del tiempo de integración.

Como se puede comprobar, el ajuste del tiempo de integración resulta fundamental a la hora de obtener correctamente el espectro. Para el ajuste de este tiempo se utiliza la relación 8.1, en donde se puede comprobar que el número de ciclos a nivel alto de la señal ST define dicho tiempo.

$$Tiempo\ integración = T_{CLK} \times 48 + Nciclos\ ST\ nivel\ alto \times T_{CLK} \tag{8.1}$$

8.1.7.2. Etapas de funcionamiento.

Una vez comprendidas las características y restricciones temporales del sensor se proceden a explicar las diferentes etapas de funcionamiento en las que se puede encontrar. Los 3 estados de funcionamiento posible son:

- **Proceso de espera**: ST se encuentra a nivel bajo y el sensor no se encuentra ni en la etapa de adquisición ni en la de transmisión.
- **Proceso de adquisición de la señal VIDEO.**
  - ◇ La adquisición comienza ante un flanco de bajada en ST y durará 87 ciclos de reloj.
  - ◇ El estado de ST deberá controlarse durante la adquisición para configurar el tiempo de integración. Siempre cumpliendo las restricciones mencionadas en la tabla 8.10.
- **Proceso de transmisión de la señal obtenida.**
  - ◇ Una vez finalizada la adquisición y la configuración del tiempo de integración se procede a realizar la transmision de los datos obtenidos.
  - ◇ ST no se vuelve a poner a nivel alto hasta el siguiente proceso de muestreo.

En la sección 8.4.1.1 se mostrarán de manera más detallada los cambios a realizar en ST para realizar una adquisición y transmisión correctas.

Finalmente, un detalle a tener en cuenta es que la señal de reloj debe ser constante durante la adquisición, sin embargo, la transmisión de los datos a Arduino puede realizarse al ritmo que se desee.

8.1.7.3. Encapsulado para el microespectrómetro.

Para que el sensor no se vea afectado por la luz ambiente será necesario aislar el dispositivo para que solo la luz proveniente del led le afecte.

La solución adoptada ha sido diseñar un encapsulado que contenga los elementos necesarios para realizar las medidas con el sensor de manera adecuada, para posteriormente construirlo mediante impresión 3D. Los elementos a tener en cuenta para dicho diseño son:

- **Microespectrómetro.**

- **Vial que contiene la muestra**: El diseño se ha realizado para un vial cilíndrico de 22 mm de diámetro y 51 mm de altura. En la imagen 8.8 se puede ver el aspecto del vial utilizado en este proyecto.

Figura 8.8 – Vial de vidrio utilizado [45].

- **Fuente de luz**: en este caso un led como el de la imagen 8.9.

Figura 8.9 – Led blanco de 5 mm [46].

Además de lo anterior, para el diseño del encapsulado se ha utilizado como referencia la imagen 3.5, por lo tanto la muestra (vial) se encontrará entre la fuente de luz y el sensor.

Teniendo todo esto en cuenta el diseño realizado finalmente se muestra en la imagen 8.10, donde se pueden ver distintas vistas del encapsulado.

(a) Encapsulado despiezado. (b) Vista lateral.

Figura 8.10 – Encapsulado diseñado para el microespectrómetro.

Las medidas de las diferentes partes que componen el encapsulado se detallan en la sección de planos [IV].

En la imagen 8.11 se muestra la implementación final del encapsulado junto con el sensor, la fuente de luz y el vial.

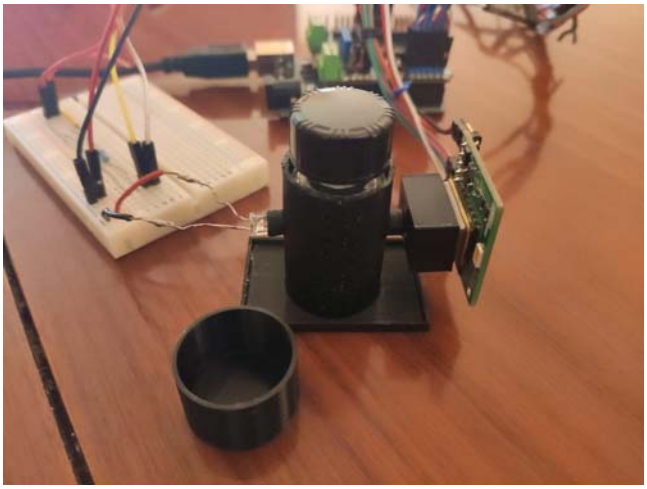


Figura 8.11 – Implementación física del encapsulado diseñado.

8.2. Shield de expansión para Arduino.

Como ya se comentó en el apartado 7.3 el sistema encargado de adquirir los datos y comunicarse con el PC será un Arduino UNO.

Para facilitar el conexionado de todos los dispositivos al Arduino el fabricante DFROBOT ofrece una *shield* de expansión que simplifica el conexionado de los sensores gracias a un formato de 3 pines que incorpora alimentación, masa y datos. En la imagen 8.12 se muestra el aspecto de la placa.

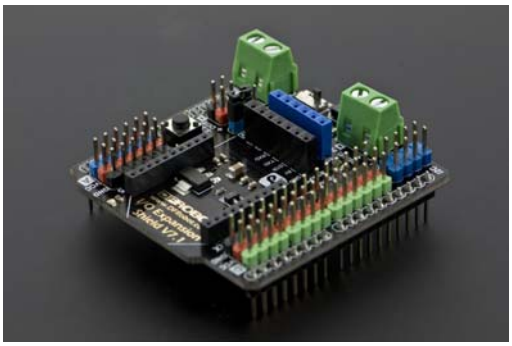


Figura 8.12 – Aspecto de la shield de expansión para Arduino [11].

Otras características particulares de esta *shield* que se tuvieron en cuenta para el proyecto son:

- **Modo de operación a 3,3V y 5V:** para seleccionar el modo de operación se utiliza un jumper que puentea 2 pines, en caso de no seleccionar ninguno el voltaje que ofrece la

placa es 0.

- **Switch RUN/PROG:** para este proyecto no debería ser necesario cambiar este switch, ya que este solo es necesario cuando se hace uso de módulos especiales que utilizan el puerto serie de la placa. Sin embargo, conviene mencionar que si se experimentan errores al intentar descargar el programa en Arduino el switch debe colocarse en PROG, una vez descargado, cambiarlo a RUN para correr el programa.
- **Salida constante de 3,3V:** se utiliza para alimentar directamente el led que actúa como fuente de luz para el microespectrómetro.

En este caso, el modo de alimentación seleccionado ha sido el de 5V. En la imagen 8.13 se muestran diferentes pines que posee la *shield* y que no han sido mencionados ya que no se hace uso de ellos en este proyecto.

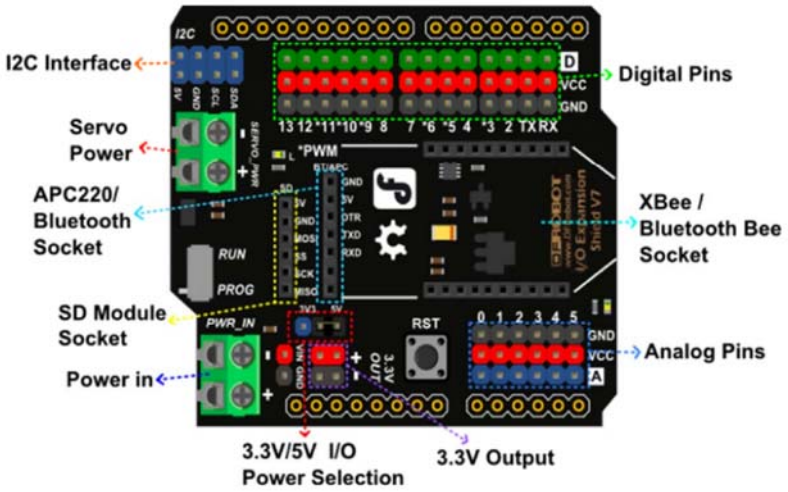


Figura 8.13 – Esquema de pines de la shield de expansión para Arduino [44].

8.3. Montaje del sistema de adquisición de datos.

En esta sección se muestran las conexiones realizadas y el montaje físico del proyecto, que se puede ver en la imagen 8.14.

Aunque el conexionado es sencillo se dividará la sección en 3 partes para facilitar la explicación:

- Conexionado del sensor de temperatura.
- Conexionado del microespectrómetro.
- Conexionado de los sensores de DFROBOT.

En cada uno de los apartados se mostrará qué pines de la placa se utilizan. Para el sensor de temperatura también se comentará el tipo de alimentación que se ha seleccionado, ya que como se mencionó en el apartado 8.1.1 hay dos formas de alimentar este sensor.

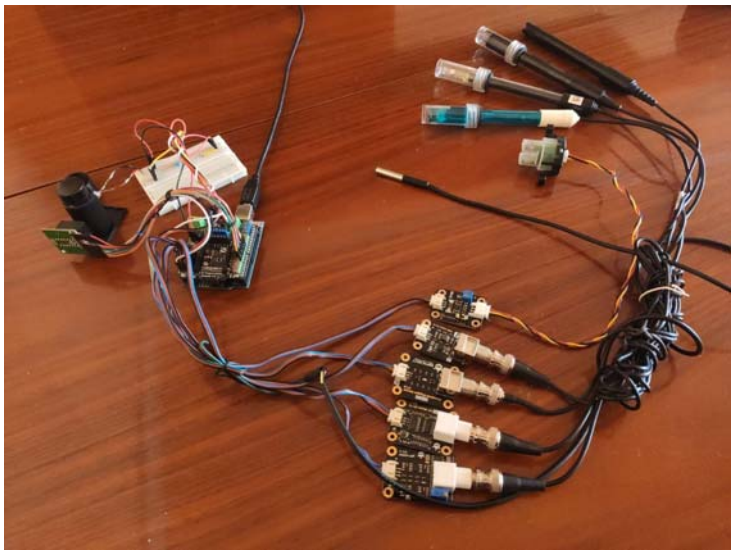


Figura 8.14 – Montaje físico realizado.

Todos los sensores se alimentan a 5V desde los pines destinados a esta tarea. Como ya se comentó, la *shield* proporciona pines de alimentación y GND para cada pin digital y analógico de Arduino.

Finalmente, en la imagen 8.15 se muestra, a modo de guía, un esquema ilustrativo del conexionado de los elementos que conforman el proyecto diseñado finalmente.

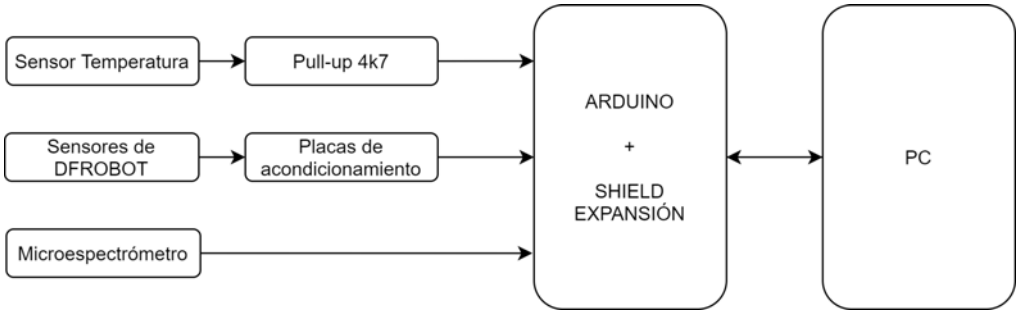


Figura 8.15 – Esquema ilustrativo del conexionado de los elementos que conforman el proyecto.

Destacar que la fuente de luz (led) se incluye en el bloque “Microespectrómetro”.

8.3.1. Conexionado del sensor de temperatura.

El DS18B20 consta de 3 pines, estos son:

- Alimentación (Cable rojo).
- GND (Cable negro).
- Datos (Cable amarillo).

Aunque existe un pin destinado para la alimentación del dispositivo este puede alimentarse a través del pin de datos, lo que reduciría el conexionado a dos cables: GND y Datos.

A este tipo de configuración se le denomina 'modo parásito' y es una características del bus 1-Wire. Mientras la línea de datos se encuentra en estado alto un condensador interno almacena energía, que se utilizará para la alimentación cuando el bus se encuentre en estado bajo. Si se decide implementar esta configuración el pin de alimentación ha de estar conectado a GND. En la imagen 8.16 se puede ver el conexionado ha realizar.

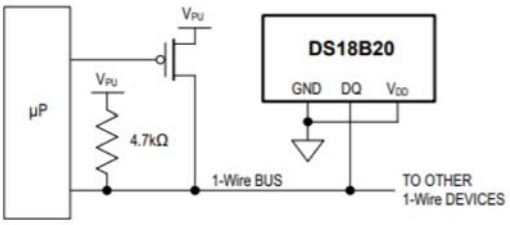


Figura 8.16 – Conexionado en modo parásito del sensor de temperatura [5].

Independientemente del modo seleccionado, el bus 1-Wire requiere requiere una resistencia de pull-up de 4.7 kΩ entre la línea destinada a la alimentación y la de datos para funcionar correctamente.

Mencionado lo anterior es momento de decidir cómo se alimentará el sensor. En este caso se ha decidido descartar el modo parásito y se realizó el conexionado con alimentación externa (desde el Arduino) como se muestra en la imagen 8.17.

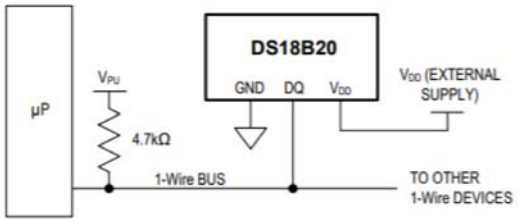


Figura 8.17 – Conexionado final del sensor de temperatura [5].

Para su conexión con Arduino los pines utilizados se muestran en la tabla 8.11.

Conexionado DS18B20	
Pin sensor	**Pin Arduino**
DQ (Datos)	Pin digital D9

Tabla 8.11 – Tabla con los pines utilizados para la conexión del sensor de temperatura.

8.3.2. Conexionado del microespectrómetro.

Para el conexionado del microespectrómetro se muestran sus pines con el correspondiente pin de Arduino en la tabla 8.12.

Conexionado Microespectrómetro	
Pin sensor	**Pin Arduino**
LASER	Pin digital D4
LED	Pin digital D5
VIDEO	Pin analógico A5
CLK	Pin digital D6
START	Pin digital D7
TRG	Pin digital D8

Tabla 8.12 – Tabla con los pines utilizados para la conexión del microespectrómetro.

Para la alimentación del led se hace uso de los pines de 3,3V que incorpora la *shield* del apartado 8.2 además de una resistencia de 10kΩ. Es importante destacar que este valor de resistencia es apropiado para el tiempo de integración seleccionado en la sección 8.4.1.1, sin embargo, si se decide modificar dicho tiempo de integración es posible que se necesite redimensionar el valor de dicha resistencia. Se ha comprobado experimentalmente que la luminosidad debe ser baja para que el sensor no proporcione medidas saturadas, teniendo esto en cuenta habrá que limitar la corriente con resistencias altas.

La conexión realizada finalmente se muestra en la imagen 8.18.

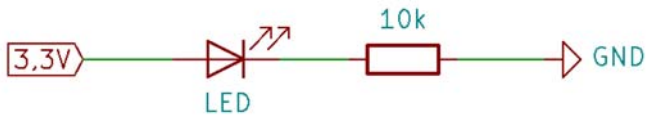


Figura 8.18 – Alimentación del led para el microespectrómetro.

8.3.3. Conexionado de los sensores de DFROBOT.

Para la conexión de los sensores con las placas de acondicionamiento se utilizan conectores tipo BNC como el de la imagen 8.19.

Figura 8.19 – Conector tipo BNC.

El único sensor que utiliza otro tipo de conector es el de turbidez. En este caso el conector es de tipo PH2.0 de 3 pines, como el que se muestra en la imagen 8.20.

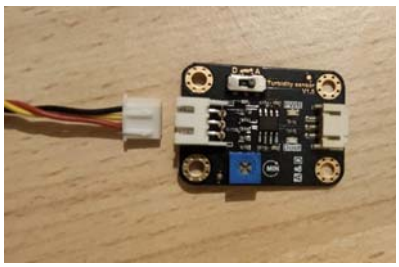


Figura 8.20 – Conector tipo PH2.0 de 3 pines.

Con el fin de facilitar la conexión de las placas de acondicionamiento con Arduino estas incorporan a su salida conectores PH2.0 de 3 pines similares al utilizado por el sensor de turbidez. Mostradas las anteriores conexiones, en la tabla 8.13 se detallan los pines de Arduino utilizados y a qué sensor están asociados.

Conexionado sensores DFROBOT	
Pin sensor	**Pin Arduino**
Turbidez	Pin analógico A0
pH	Pin analógico A1
Conductividad eléctrica	Pin analógico A2
Potencial redox	Pin analógico A3
Oxígeno disuelto	Pin analógico A4

Tabla 8.13 – Tabla con los pines utilizados para la conexión de los sensores de DFROBOT.

8.4. Programación software.

La programación realizada se divide en dos partes:

- Código en Arduino: se realizará la adquisición de los datos proporcionados por diferentes sensores. Una vez obtenidas las medidas se realizarán una serie de cálculos para convertir las medidas a magnitudes físicas entendibles para, por último, enviarlas al PC haciendo uso del puerto serie.
- Código en Python: será el encargado de controlar tanto el muestreo, como el guardado, la visualización y la subida de los datos a la nube.

En la imagen 8.21 se muestra un diagrama en el que se explica de manera general como se comunican el PC y Arduino.

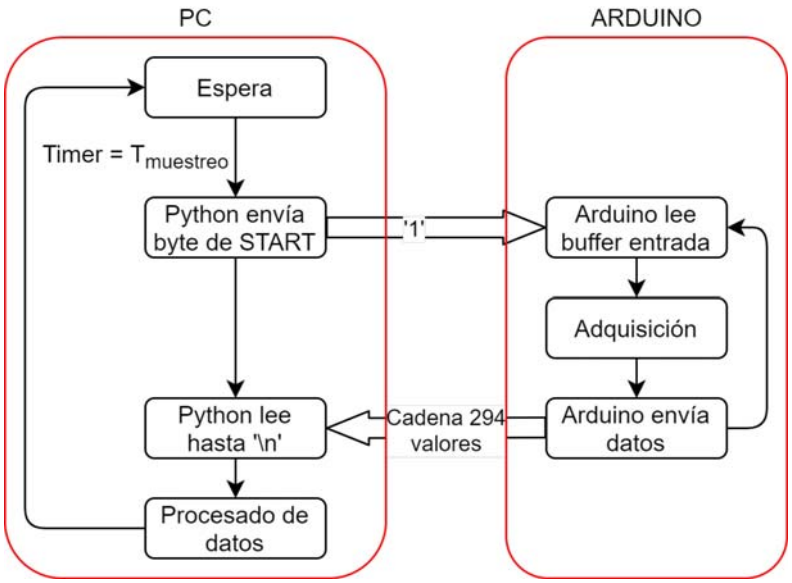


Figura 8.21 – Flujograma general de comunicación entre el PC y Arduino.

Como se puede ver, el PC será el que controle el muestreo enviando el caracter “1” en forma de byte, que llegará al buffer de entrada de Arduino, cuando este lee un “1” la adquisición comienza. Una vez finalizada la adquisición los datos son enviados al PC siguiendo el formato mostrado en la tabla 8.14. Los valores se separan con comas y se marca el fin de la cadena de datos con el marcador “$\backslash n$”, que indica el inicio de una nueva línea. Este proceso se repite cada vez que un timer alcanza el valor del tiempo de muestreo definido por el usuario, en este caso el tiempo de muestreo será de 15 segundos debido a una limitación que se trata en la sección 8.4.2.5.

Formato de la cadena de datos que se entrega desde Arduino						
Temperatura	Turbidez	pH	EC	ORP	DO	288 valores microespectrómetro

Tabla 8.14 – Formato de la cadena de datos que se entrega desde Arduino.

Para la explicación de las distintas funciones que componen el código realizado se seguirá un orden. Se comenzará con explicaciones generales y poco a poco se irá entrando en detalle.

8.4.1. Software desarrollado en Arduino.

En esta sección se explicará de manerá detallada el funcionamiento del código mostrado en el anexo 13. Como se puede observar en la imagen 8.22 los diferentes estados del programa son:

- **Proceso inicial:** en esta etapa se declaran las librerías utilizadas y las variables globales del programa además de realizar configuraciones como la declaración de los pines de entrada y salida o el inicio del bus 1-Wire para el sensor de temperatura.
- **Tmuestreo OK?:** En esta etapa se comprueba si desde la última medida realizada ya ha transcurrido un tiempo predefinido por el usuario (en este caso 25 ms). El objetivo

de esta etapa es controlar que se tomen medidas con intervalos de 25 ms de diferencia entre sí, para posteriormente, calcular una media y así obtener un valor más fiable, ya que es habitual que sensores de DFROBOT proporcionen medidas erróneas alguna que otra vez.

- **Muestra:** etapa en la que se obtiene una muestra para los sensores de turbidez, pH, EC, ORP y DO. Además de esto se va aumentando el parámetro ArrayIndex cada vez que se entra en esta etapa, de esta forma los datos se van guardando en vectores. Este proceso se repite un total de 10 veces antes de proceder con la conversión y transmisión de los datos.
- **Temperatura + Microespectrómetro:** en el caso del microespectrómetro y el sensor de temperatura se toma una única medida en lugar de 10. Esto se debe a que el tiempo de adquisición de una medida es bastante elevado (750 ms), por lo que para no ralentizar el proceso se realiza una única medida. Cabe mencionar que las medidas proporcionadas por el sensor de temperatura son bastante precisas, por lo que con obtener una es suficiente. En el caso del microespectrómetro el tiempo de adquisición de una medida no es muy grande, sin embargo, al igual que el sensor de temperaura este ofrece medidas bastante precisas por lo que con una es suficiente.
- **Conversiones:** en esta etapa se aplican las ecuaciones del anexo 11 para convertir las tensiones en las magnitudes físicas correspondientes en cada caso.

El diagrama 8.22 detalla el proceso completo para realizar 1 medida completa, esto es, una cadena de datos de 294 valores como la mostrada en 8.14. Este proceso requiere de un tiempo mínimo para ejecutarse de 900 ms, por tanto, para aplicar un tiempo de muestreo constante se tendrá esto en cuenta.

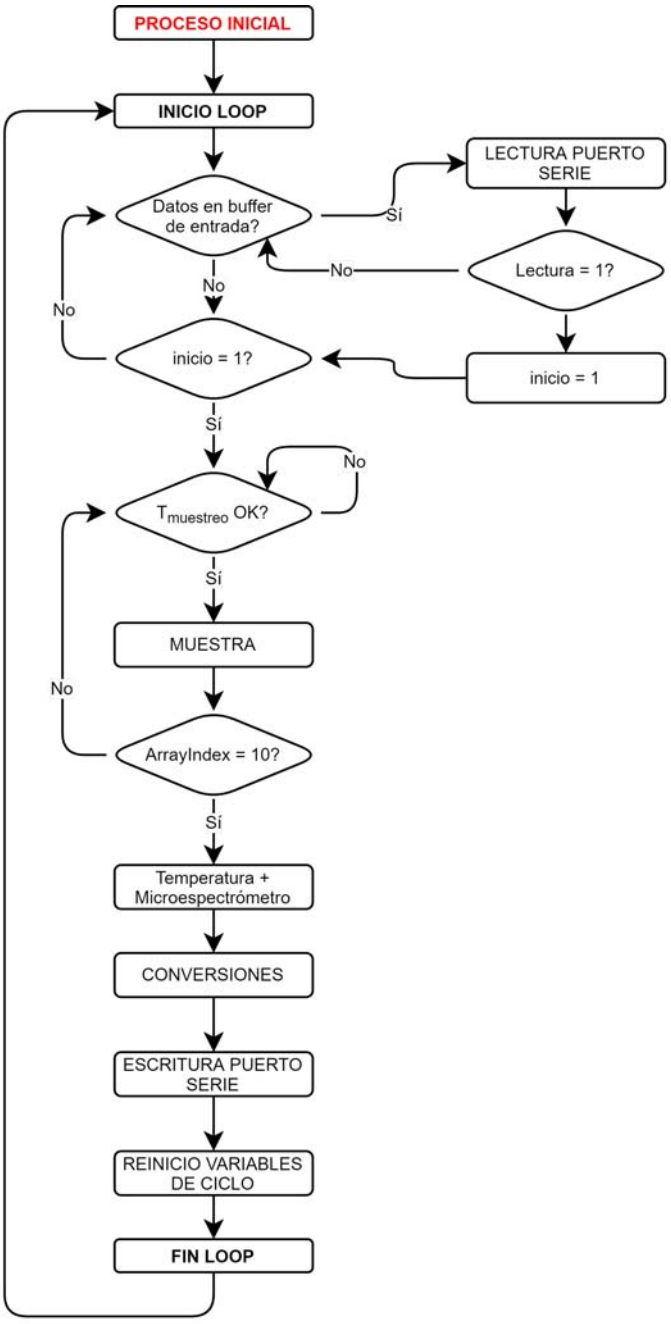


Figura 8.22 – Flujograma del código desarrollado para Arduino.

8.4.1.1. Función para el muestreo con el microespectrómetro.

Para realizar medidas con el microespectrómetro se ha desarrollado el código del anexo 13.1.1, en el que se controla el estado de ST siguiendo las explicaciones y restricciones mencionadas en el apartado 8.1.7.1.

En la imagen 8.23 se muestra el flujograma con los distintos estados de ST conforme transcurren los ciclos de reloj.

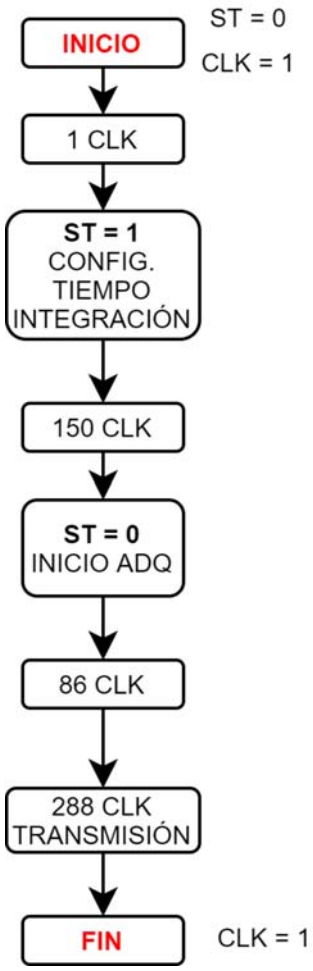


Figura 8.23 – Flujograma del código para el funcionamiento del microespectrómetro.

- **Restricciones temporales:** En la tabla 8.15 se realiza una comparativa entre los tiempos a cumplir y los tiempos obtenidos con el código realizado, comprobándose que se cumplen las restricciones temporales impuestas en la sección 8.1.7.

Tabla comparativa de restricciones temporales		
Parámetro	**Mínimo**	**Obtenido**
Frecuencia de reloj	200 KHz	500 KHz
Ciclos totales del proceso (*tpi*)	381	525
Ciclos ST alto (*thp*)	6	150
Ciclos ST bajo (*tlp*)	375	375

Tabla 8.15 – Tabla de comparación con los tiempos mínimos vs. tiempos generados.

- **ST = 1**: Con el flanco de subida en ST comienza el proceso de configuración del tiempo de integración, que durará en este caso 150 ciclos de reloj.
- **ST = 0:** Con el flanco de bajada en ST comienza el proceso de adquisición, durará 87 ciclos de reloj.

La forma de onda obtenida para la señal ST es similar a la que se puede ver en la imagen 8.24, en la que también se muestran los tiempo *thp*, *tlp* y *tpi*.

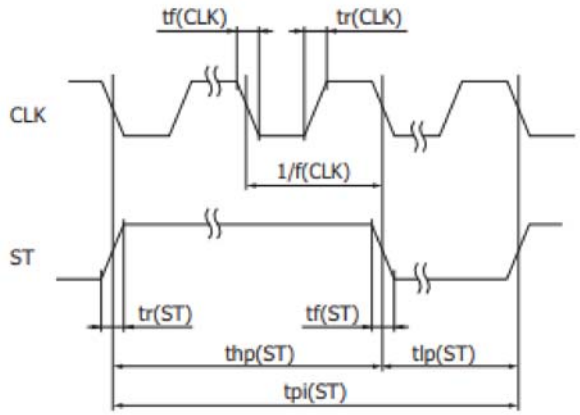


Figura 8.24 – Forma de la señal ST obtenida [16]

Cabe mencionar que la primera medida obtenida no será correcta, después de esta primera medida los datos se obtendrán correctamente.

8.4.1.2. Función de cálculo de la media.

El código del anexo 13.1.2 calcula la media de un vector de valores descartando el máximo y el mínimo para realizar dicho cálculo. En la imagen 8.25 se puede ver un diagrama de estados con las etapas que componen la función.

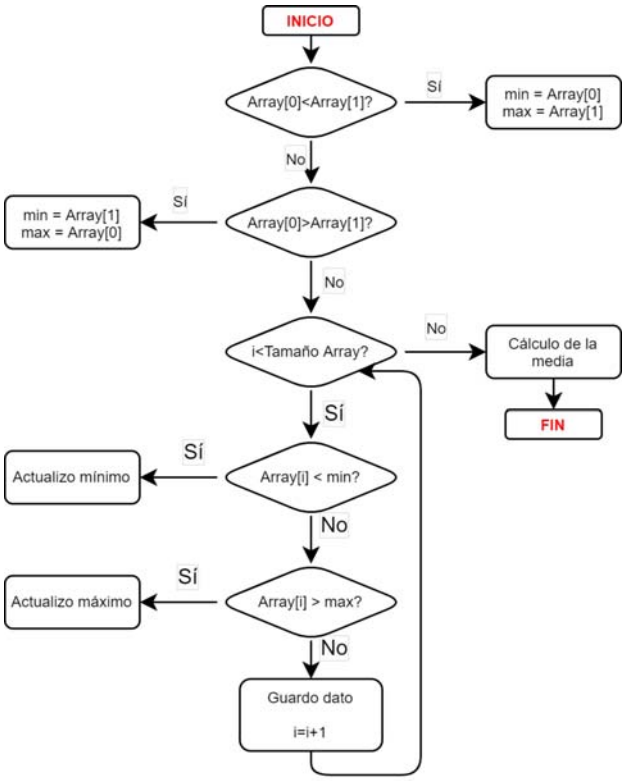


Figura 8.25 – Flujograma de la función del cálculo de la media.

8.4.2. Software desarrollado en Python.

El código mostrado en el anexo 14, como ya se comentó, realiza las siguientes tareas:

- Controlar el tiempo de muestreo.
- Comunicación con Arduino mediante puerto serie.
- Guardar los datos adquiridos por el puerto serie en un archivo CSV (*Comma Separated Values* o Valores separados por comas).
- Graficar los datos en tiempo real.
- Subir los datos a ThingSpeak.

Antes de comenzar con la explicación del código que realiza las anteriores tareas conviene entender como funciona el script de python de forma general. En el diagrama de la imagen 8.26, en el que se describe el funcionamiento de la función *main* del anexo 14.1.4, se pueden ver los estados en los que se puede encontrar el programa. Basicamente el script hace una serie de operaciones iniciales como la creación de la ventana en la que se graficarán los datos y la declaración de variables globales.

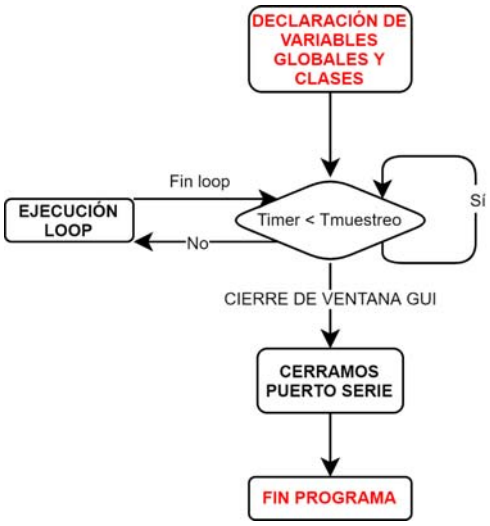


Figura 8.26 – Flujograma general del funcionamiento del script de python.

8.4.2.1. Función 'loop'.

Esta función, mostrada en el anexo 14.1.5, es llamada por un timer desde la función *main* cada 15 segundos, que es el tiempo de muestreo que se ha definido para este proyecto. La razón de fijar el muestreo en 15 segundos se debe a las limitaciones mencionadas en el apartado 8.4.2.5.

Figura 8.27 – Flujograma de la función *loop*.

El flujograma de la imagen 8.27 muestra las etapas que componen esta función. Cuando se entra en esta función lo primero que se hace es guardar el tiempo actual, para posteriormente concatenar este tiempo a la cadena de valores obtenida. El formato de la cadena pasa a ser el mostrado en la tabla 8.16.

Formato de la cadena de datos que se entrega desde Arduino							
HH:MM:SS	Temperatura	Turbidez	pH	EC	ORP	DO	microespectrometro x 288

Tabla 8.16 – Formato de la cadena de datos con la que se trabaja en python.

8.4.2.2. Clase para comunicación serie.

Para la comunicación serie con Arduino se ha creado la clase *SerialCommunication*, cuyo código se puede ver en el anexo 14.1.1. Como ya se mencionó, la comunicación se realiza utilizando el protocolo UART y los datos se transmiten en forma de bytes, esto debe tenerse en cuenta a la hora trabajar con los datos leídos.

La clase consta de 4 funciones:

- **Función *__init__*:** función de inicialización, se ejecuta automáticamente al crear la clase.

Su objetivo es establecer conexión con el puerto serie indicado y antes de finalizar se envía un aviso de si la conexión se ha realizado o no con éxito. La función se ejecuta al iniciar el script y para poder realizar correctamente la primera medida se hace una pausa de 2 segundos.

- **Función *SendData*:** función encargada de envíar la orden de inicio de adquisición a Arduino, para ello se envía un 1. Además de esto, se concatena a la cadena de datos la hora de realización de la media siguiendo el formato mostrado en 8.16.
- **Función *ReadData*:** esta función, cuyo flujograma se muestra en la imagen 8.28, lee el puerto serie hasta encontrarse con un carácter de salto de línea. Una vez finalizada la lectura se limpia el buffer de entrada, dejándolo vacío para la siguiente medida. Por último, el mensaje se decodifica para transformar la variable tipo *byte* a una variable de tipo *string* y se devuelve la cadena de datos con el tiempo concatenado.

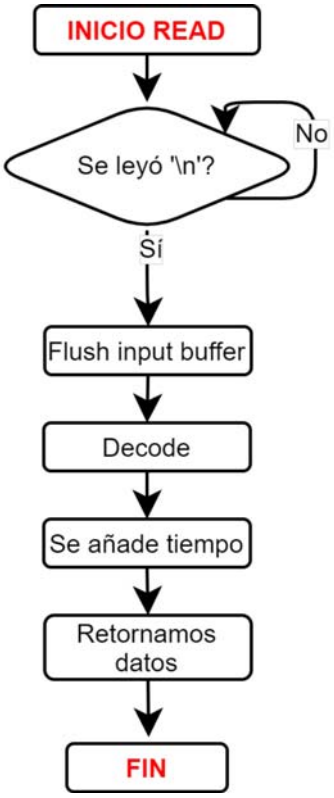


Figura 8.28 – Flujograma del código para la comunicación por puerto serie.

- **Función *Close*:** cierra el puerto serie antes de parar la ejecución del script.

8.4.2.3. Función de guardado de datos.

Para guardar los datos en un archivo CSV se utiliza la función *CSVfile*, cuyo código se muestra en el anexo 14.1.2. Las distintas etapas que componen la función se pueden ver en el flujograma de la imagen 8.29.

Figura 8.29 – Flujograma de la función para el guardado de los datos en CSV.

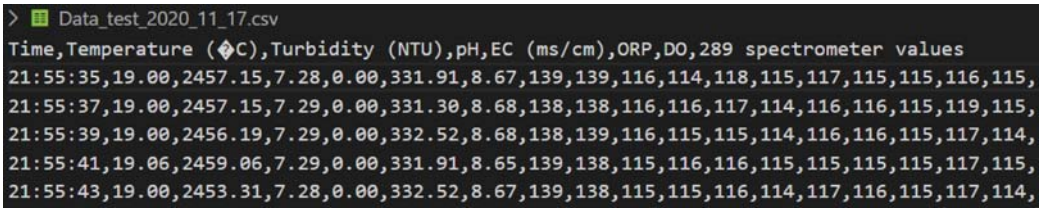
A continuación se muestran algunos detalles a tener en cuenta del código desarrollado:

- **Encabezado:** El encabezado del archivo se introduce en el momento de su creación, ocupa una línea y siempre es el mismo: *Time,Temperature (ºC),Turbidity (NTU),pH,EC (ms/cm),ORP,DO,288 spectrometer values*.

- **Cierre del archivo después de cada escritura:** como se puede comprobar en la imagen 8.29 el archivo se cierra después de cada escritura. Esto permite ver como se va actualizando el archivo conforme se adquieren los datos.

- **Creación de un archivo nuevo cada día:** Además de lo anterior, también se ha realizado un control del día, mediante el cual se consigue guardar los datos de días diferentes en archivos diferentes. Para la generación del nombre de los archivos se sigue el siguiente patrón: *Data_AÑO_MES_DIA.csv*, en donde los parámetros que varían son el año, el mes y el día. Si existe un archivo cuya fecha coincide con la de ese día se abre ese archivo, en caso de que no exista, se crea uno nuevo.

En la imagen 8.30 se puede ver un ejemplo de la apariencia del archivo, en la que cada línea representa una medida completa.


Data_test_2020_11_17.csv

```
Time,Temperature (ºC),Turbidity (NTU),pH,EC (ms/cm),ORP,DO,289 spectrometer values
21:55:35,19.00,2457.15,7.28,0.00,331.91,8.67,139,139,116,114,118,115,117,115,115,116,115,
21:55:37,19.00,2457.15,7.29,0.00,331.30,8.68,138,138,116,116,117,114,116,116,115,119,115,
21:55:39,19.00,2456.19,7.29,0.00,332.52,8.68,138,139,116,115,115,114,116,116,115,117,114,
21:55:41,19.06,2459.06,7.29,0.00,331.91,8.65,139,138,115,116,116,115,115,115,115,117,115,
21:55:43,19.00,2453.31,7.28,0.00,332.52,8.67,139,138,115,115,116,114,117,116,115,117,114,
```

Figura 8.30 – Disposición de los datos en el archivo CSV.

8.4.2.4. Clase para el visualizado de los datos en tiempo real.

El código del anexo 14.1.3 muestra la clase *GUI* que permite representar los datos en diferentes gráficas, facilitando su visualización. Todas las gráficas se crean en una ventana emergente, para detener la ejecución del script basta con cerrar esta ventana.

La clase consta de dos funciones, una que se ejecuta una vez durante su declaración y otra para actualizar las gráficas cada vez que llegan datos. Para la función *UpdateGUI*, encargada de actualizar las gráficas, no es necesario realizar un diagrama de estados ya que solo se realizan dos acciones, estas son:

- **Limitación del tamaño de los arrays de ploteo:** el tamaño de estos vectores define el número de datos que se visualizan simultáneamente en una gráfica. En este caso se ha configurado para mostrar los 50 últimos valores obtenidos, aunque este valor se puede cambiar modificando la variable *axisX* de la función *main* mostrada en el anexo 14.1.4.
- **Actualización de los valores de las gráficas.**

En la imagen 8.31 se muestra el flujograma que describe las acciones realizadas en la etapa de inicialización. Como se puede ver, en esta función solo se realizan tareas como la creación de las gráficas y ajustes de los ejes.

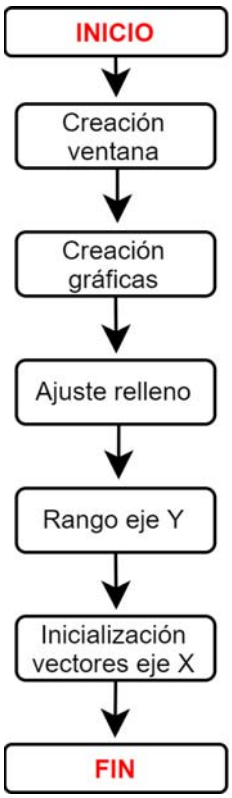


Figura 8.31 – Flujograma de la función de inicialización de las gráficas.

Por otro lado, en la imagen 8.32 se puede ver la ventana resultante con las distintas gráficas generadas.

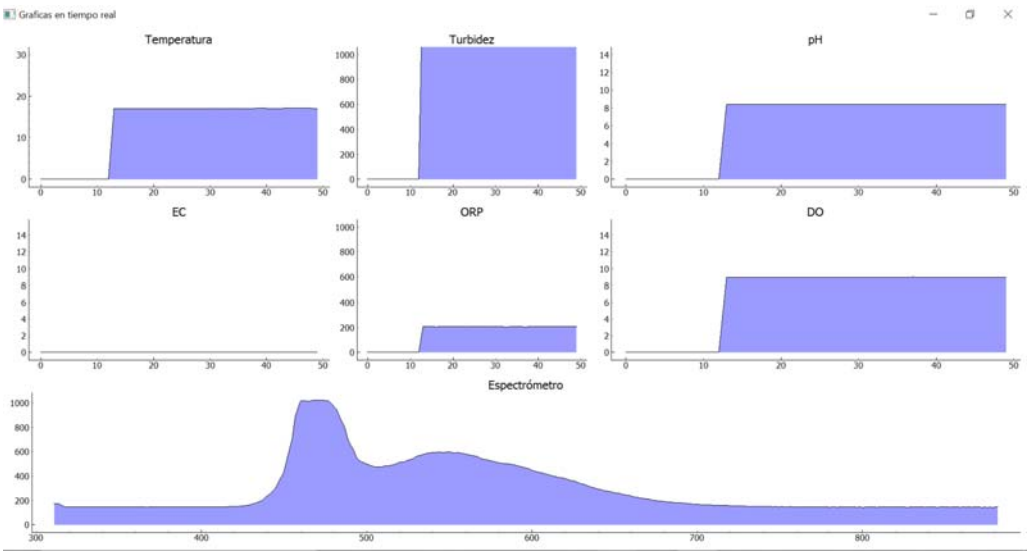


Figura 8.32 – Aspecto de las gráficas generadas.

8.4.2.5. Subida de los datos a la nube.

La subida de los datos a ThingSpeak se ha realizado utilizando la modalidad gratuita que el servicio ofrece. El uso de la versión gratuita de ThingSpeak conlleva una restricción en la tasa de refresco de 15 segundos, mientras que con la versión de pago se puede muestrear hasta a 1 segundo. Para este proyecto la licencia gratuita es suficiente, por lo que el tiempo de muestreo se ha fijado en 15 segundos, para tiempos de muestreo más bajos habría que recurrir a la versión de pago.

En lo referente al código que permite subir los datos a la nube, la librería *requests* simplifica enormemente esta tarea. Para realizar la comunicación y subir los datos basta con escribir la dirección web, la *API KEY* y un *field* por cada dato que se quiera subir. En la tabla 8.17 se muestran los parámetros utilizados además de la función que simplifica la comunicación mediante protocolo HTTP. El parámetro *Key* es único y particular de cada canal y es el elemento que concede los permisos para, en este caso, realizar escrituras en el mismo.

Partes que conforman la URL completa	
Url	https://api.thingspeak.com/update?api_key=
Key	XXXXXXXXXXXXXXXX
Header	&field1=Temperatura&field2=Turbidez&field3=pH &field4=EC&field5=ORP&field6=DO
Función	urllib.request.urlopen(URL+KEY+HEADER)

Tabla 8.17 – Componentes de la URL a utilizar para subir datos a ThingSpeak.

En la imagen 8.33 se puede ver un ejemplo de como se guardan los datos en la nube de ThingSpeak.

Figura 8.33 – Canal de ThingSpeak donde se almacenan los datos.

8.5. Procedimiento para la realización de las medidas.

En esta sección se presenta la metodología aplicada para realizar las medidas junto con las consideraciones realizadas para comprobar si los sensores proporcionan valores lógicos cuando son sumergidos en sus respectivas muestras. Cabe mencionar que, exceptuando las medidas de pH y de conductividad eléctrica, no se ha podido disponer de muestras cuyos parámetros sean conocidos de manera exacta. Por tanto, para cada caso, se han tomado las consideraciones expuestas a continuación:

- **Medidas de turbidez:** se utilizarán 5 muestras de diferente turbidez y se comprobará si el sensor entrega valores mayores conforme la turbidez aumenta.
- **Medidas de pH:** se utilizará la muestra de pH 3 mencionada en puntos anteriores, una muestra de agua de grifo, que debería presentar un pH aproximadamente neutro y por último, una muestra de NaOH que presentará un pH básico, aunque de valor desconocido.
- **Medidas de EC:** se realizarán 6 medidas con el objetivo de comprobar si el sensor es capaz de diferenciar distintos valores de conductividad.
- **Medidas de ORP:** se utilizarán las muestras de $AgNO_3$ y Ac_2Zn facilitadas, además de otras 4 para hacer más completo el experimento. El potencial redox de las dos primeras muestras es desconocido, aunque se sabe que los potenciales que presentan son diferentes. Una de las 4 muestras mencionadas anteriormente será agua de grifo, cuyo

potencial redox debería ser de entre +150mV y +350mV. Para el resto de muestras, los valores de potencial redox son desconocidos.

- **Medidas de DO:** para comprobar el correcto funcionamiento del sensor de oxígeno disuelto se utilizará una muestra de agua que se dejará reposar durante un tiempo. El nivel de oxígeno disuelto debería ser próximo a 4-5 mg/L, se comenzará a agitar levemente el agua, en este momento debería detectarse un ascenso gradual del oxígeno disuelto.
- **Medidas de absorbancia y espectro electromagnético:** en primer lugar se medirá el espectro electromagnético de leds de 4 colores diferentes (rojo, verde, azul y blanco) para comprobar si el sensor captura diferentes rangos del espectro. A continuación se medirá la absorbancia de la muestra de $CuSO_4$, además de una muestra de agua turbia y la muestra de agua residual elaborada en laboratorio. Se ha establecido como disolución blanca (absorbancia nula) una muestra de agua desionizada.

En lo que respecta a la medida del pH, el valor obtenido en laboratorio ha sido de 2,94, como se puede ver en la imagen 8.34. Para su obtención se ha acidificado agua MQ añadiendo poco a poco ácido clorhídrico hasta llegar al valor de pH mencionado.

Figura 8.34 – Medida del pH realizada en el laboratorio.

Una vez comprendido el proceso lógico aplicado para determinar si los sensores proporcionan medidas válidas se procede a mostrar la metodología seguida para la realización de las mismas. Para todas ellas la temperatura de las disoluciones ha sido de entorno a 18 ºC y el tiempo de muestreo se ha reducido a 1 segundo para ver de manera más clara la respuesta en el tiempo de los sensores.

Los pasos seguidos para cada medida han sido los siguientes:

1. Se introduce el sensor, previamente lavado y secado, en la muestra.
2. El sensor debe permanecer en la muestra durante un tiempo hasta que los valores lleguen a regimen permanente.
3. Una vez estabilizada la medida se puede retirar el sensor.

4. En el momento en el que se retira el dispositivo de la muestra es conveniente lavarlo y secarlo. Para el lavado es recomendable utilizar agua destilada o desionizada, de esta forma se evita la posible contaminación de otras muestras.

8.6. Resultados obtenidos y conclusiones.

En esta sección se proporcionan los resultados obtenidos para las muestras del apartado 3.1 de antecedentes además de alguna adicional utilizada para hacer los experimentos más completos, como se mencionó en 8.5. Realizados los experimentos se procederá a presentar una serie de conclusiones a partir de los datos obtenidos.

Antes de comenzar con las medidas es importante mencionar que, para hacer más completas las pruebas, se han utilizado algunas muestras no mencionadas en la sección 3.1 de antecedentes, debido a que no son muestras de laboratorio. Estas disoluciones son:

- NaOH + agua: de esta forma se puede obtener una solución de pH básico.
- Agua turbia: útil para llegar a valores de turbidez altos, ya que las muestras de laboratorio no son muy turbias.
- Agua de grifo: utilizada para la realización del experimento de oxígeno disuelto. Además, se ha medido su conductividad, ya que aunque se conoce que esta debe ser baja, debería ser mayor de 0.

Para la realización de los experimentos se ha utilizado un tiempo de muestreo de 1 segundo para ver las curvas de respuesta de los sensores de manera más clara. Destacar también que el objetivo de estos experimentos es comprobar si existe la posibilidad de obtener resultados que permitan diferenciar entre muestras diferentes.

8.6.1. Medidas de turbidez.

Para comprobar la respuesta del sensor antes distintos niveles de turbidez se han utilizado las muestras de la imagen 8.35.

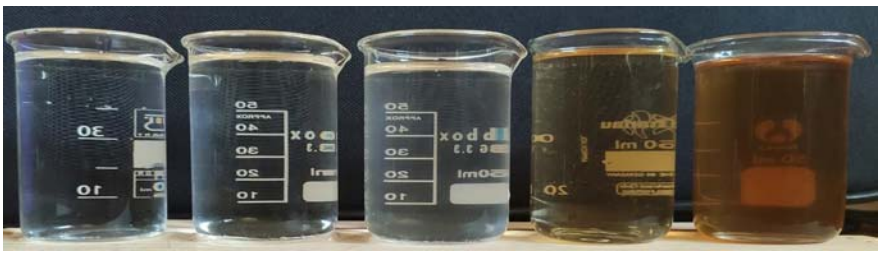


Figura 8.35 – Muestras utilizadas para el experimento de turbidez.

A continuación, de izquierda a derecha, se nombran estas muestras: Agua pura (MQ), agua residual EDAR, agua residual LAB, agua turbia 1 y agua turbia 2.

Es importante mencionar que antes del experimento se realizó una compensación de offset de +0.18 V para obtener valores de NTU cercanos a 0 en caso de que el sensor se encontrase en la muestra de agua destilada, que como se puede ver, es la menos turbia.

Con el sensor ya calibrado se procedió a tomar medidas para las muestras mencionadas, como se puede ver en la imagen 8.36. Las secciones en las que se dispara la turbidez se corresponden con los instantes de tiempo en los que el sensor se ha retirado de la muestra para ser introducido en la siguiente.

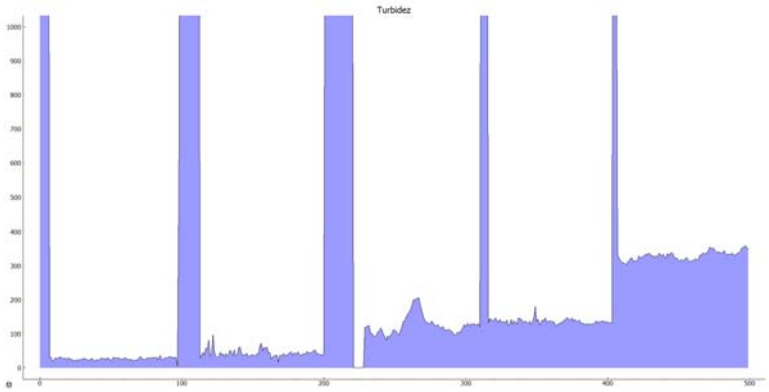


Figura 8.36 – Respuesta del sensor para las distintas muestras.

En la imagen 8.37 se muestra una gráfica comparativa con los resultados obtenidos para las distintas muestras. Cabe destacar que se ha realizado una distinción entre las muestras que presentan partículas en suspensión (línea discontinua) y las que no (línea continua).

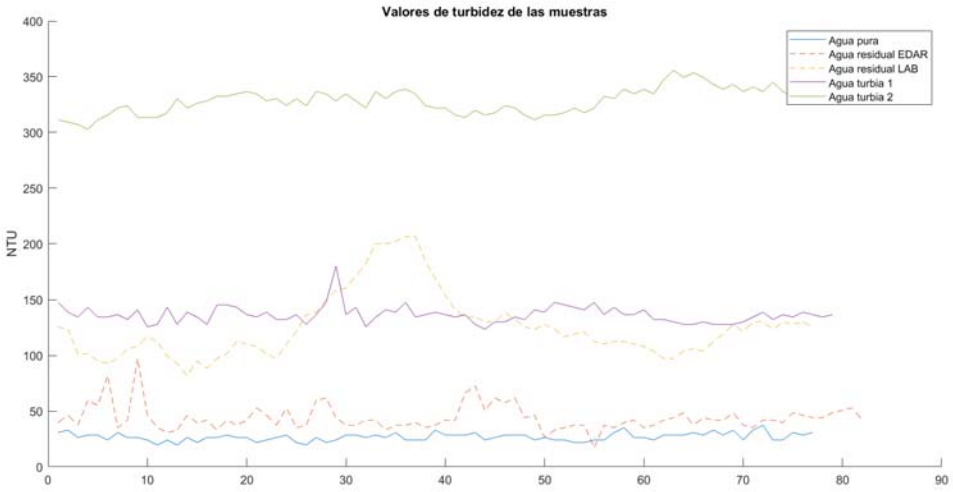


Figura 8.37 – Gráfica comparativa de los distintos valores de turbidez.

Analizando los resultados obtenidos se pueden extraer las siguientes conclusiones:

- La presencia de partículas en suspensión afecta considerablemente a la estabilidad de las medidas, como se puede ver en el caso de la muestra “agua residual LAB”. Esto se debe al principio de funcionamiento del dispositivo, que cuantifica la turbidez a partir de

la capacidad de transmitancia que tiene un haz de luz dentro de la muestra. Por tanto, los picos de turbidez obtenidos se producen cuando las partículas en suspensión se cruzan con este haz, disminuyendo la transmitancia y disparando consecuentemente los valores de turbidez.

Aunque esto se puede ver como un problema, al no poder obtener un valor estable de turbidez, lo cierto es que existe la alternativa de cuantificar los sólidos en suspensión de la muestra.

- Se comprueba que es posible diferenciar las muestras utilizadas en el experimento a partir de los niveles de turbidez obtenidos.

Finalmente se concluye que, aunque con ciertas limitaciones, el sensor proporciona resultados bastante buenos.

8.6.2. Medidas de pH.

Para la realización del experimento de pH se han utilizado las 3 muestras presentadas en la tabla 8.18.

Nombre de la muestra	pH
Muestra NaOH	Básico (pH $>$ 7)
Muestra agua grifo	Neutro (pH $\approx$ 7)
Muestra HCl	Ácido (pH = 2,94)

Tabla 8.18 – Muestras utilizadas para el experimento de pH.

En la imagen 8.38 se puede ver la gráfica de respuesta del sensor con las distintas transiciones entre muestras, mientras que en la imagen 8.39 se realiza una comparativa de los resultados obtenidos.

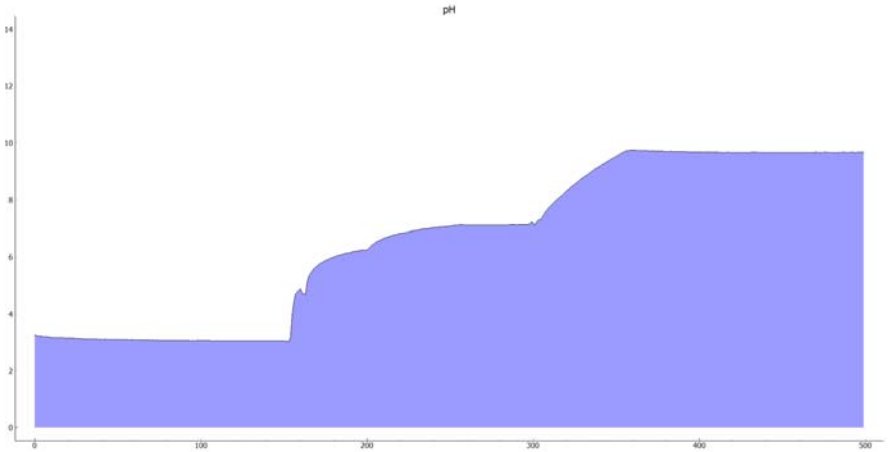


Figura 8.38 – Respuesta del sensor para las 3 muestras.

Analizando la gráfica comparativa 8.39 se concluye que el sensor proporciona valores de pH bastante cercanos a los reales, diferenciando claramente las 3 muestras. Concretamente,

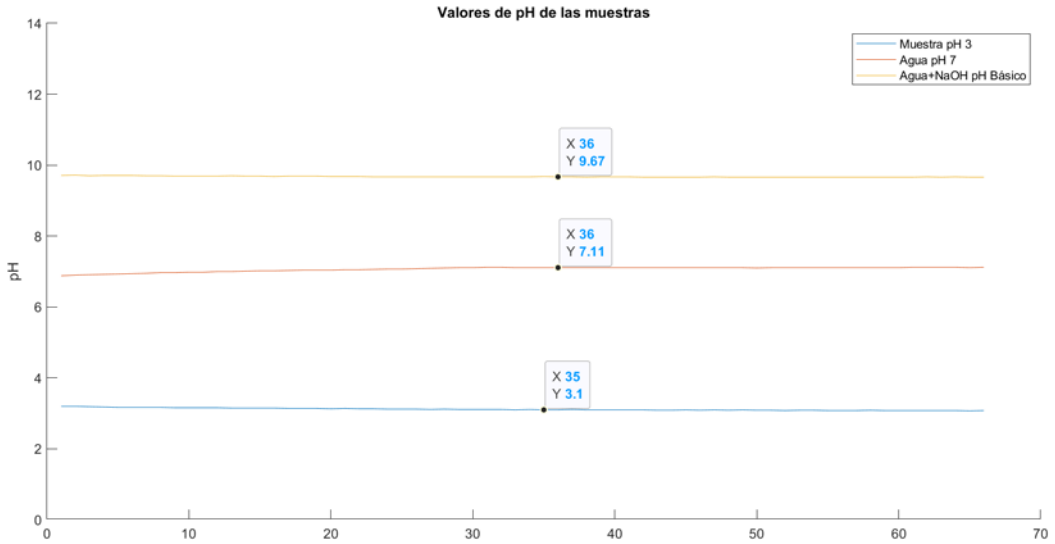


Figura 8.39 – Gráfica comparativa de los valores de pH.

para la muestra ácida de pH 2,94 se han registrado valores de pH de entorno a 3,1.

8.6.3. Medidas de conductividad eléctrica.

Para las medidas de conductividad eléctrica se han utilizado las 6 muestras recogidas en la tabla 8.19.

Nombre de la muestra	Conductividad estimada
Muestra agua grifo	Baja
Muestra agua residual EDAR	Baja o Media
Muestra agua residual LAB	Media
Muestra 1,4mS/cm	Baja
Muestra 12,88mS/cm	Alta
Muestra NaCl	Alta

Tabla 8.19 – Muestras utilizadas para el experimento de pH.

Un detalle a destacar es que, aunque las muestras incluídas con el sensor (“Muestra 1,4mS/cm” y “Muestra 12,88mS/cm”), indicaban valores de conductividad determinados, estos no han sido medidos en laboratorio por lo que es posible que los valores de conductividad reales no se correspondan con los valores indicados.

En la imagen 8.40 se muestra la respuesta del sensor para las muestras mencionadas. Cabe destacar que los intervalos de conductividad 0 se corresponden con los instantes de tiempo en los que el sensor se encontraba al aire.

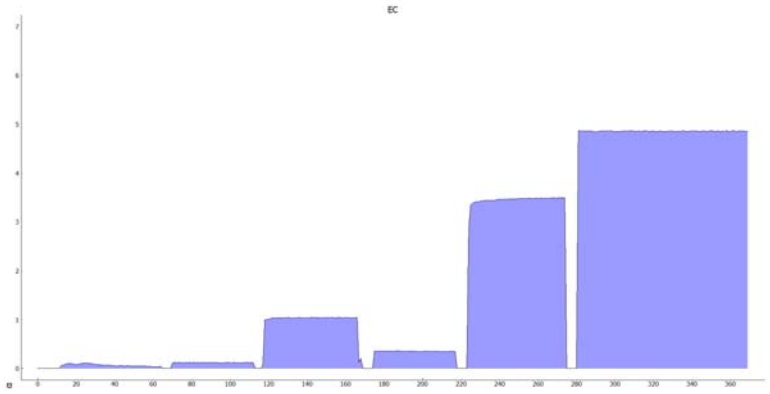


Figura 8.40 – Respuesta del sensor de conductividad para las diferentes muestras.

Los resultados obtenidos se comparan en la gráfica 8.41.

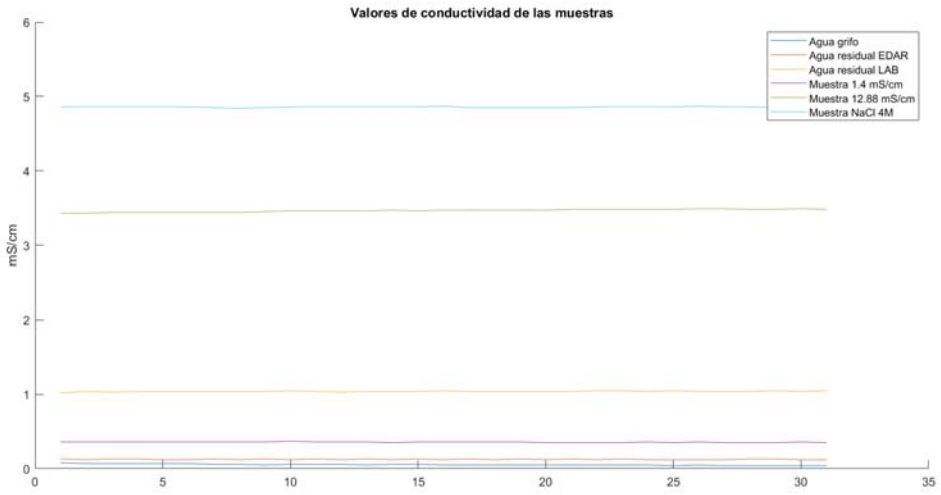


Figura 8.41 – Gráfica comparativa de los valores de conductividad.

Como se puede ver, además de obtener valores de conductividad diferentes para cada muestra los resultados obtenidos son lógicos. Las muestras de agua presentan una conductividad baja, mientras que la muestra de $NaCl$ presenta una conductividad alta.

8.6.4. Medidas de potencial redox.

Para el experimento de potencia redox se han utilizado las 6 muestras recogidas en la tabla 8.20.

Nombre de la muestra
Muestra agua destilada
Muestra agua MQ
Muestra agua residual LAB
Muestra Ac_2Zn
Muestra $AgNO_3$

Muestra NaCl

Tabla 8.20 – Muestras utilizadas para el experimento de potencial redox.

En este experimento se desconoce qué valores de potencial redox presentarán las muestras, por tanto, tan solo se comprobará si se pueden diferenciar dichas muestras a partir de los resultados obtenidos. En la imagen 8.42 se puede ver la respuesta del sensor a lo largo del experimento.

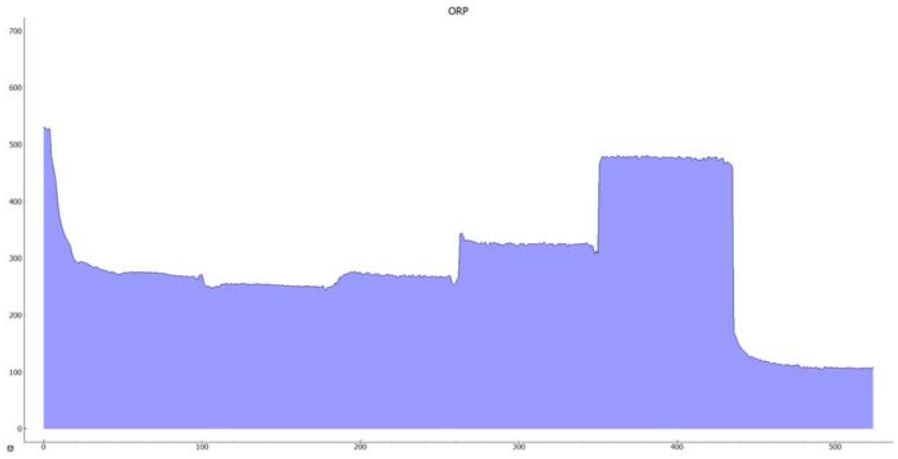


Figura 8.42 – Respuesta del sensor de potencial redox para las diferentes muestras.

Los resultados obtenidos se comparan en la gráfica 8.43.

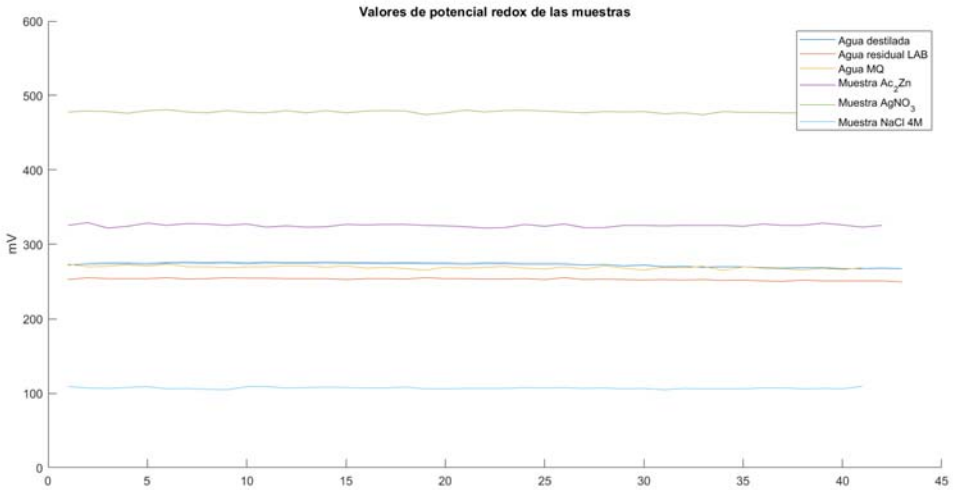


Figura 8.43 – Gráfica comparativa de los valores de potencial redox.

Observando la gráfica se comprueba que el potencial redox de las muestras de agua MQ y destilada es prácticamente el mismo, mientras que el agua residual de la EDAR es ligeramente menor. Todas ellas presentan un potencial redox de entre 250mV y 275mV.

En el caso de las muestras de Ac_2Zn y $AgNO_3$ se comprueba que existe una clara di-

ferencia entre ambas. Por último, la muestra de NaCl se diferencia claramente del resto de disoluciones, presentando el menor potencial redox.

En conclusión, el sensor es capaz de diferenciar diferentes agentes químicos, sin embargo, se desconoce si las magnitudes entregadas se corresponden con la realidad.

8.6.5. Medidas de oxígeno disuelto.

Para este experimento se ha utilizado únicamente una muestra de agua de grifo. El objetivo fue comprobar si se detectaban cambios en el oxígeno disuelto en la muestra si esta se oxigenaba mediante agitación.

En la gráfica 8.44 se puede ver la respuesta del sensor a lo largo del experimento.

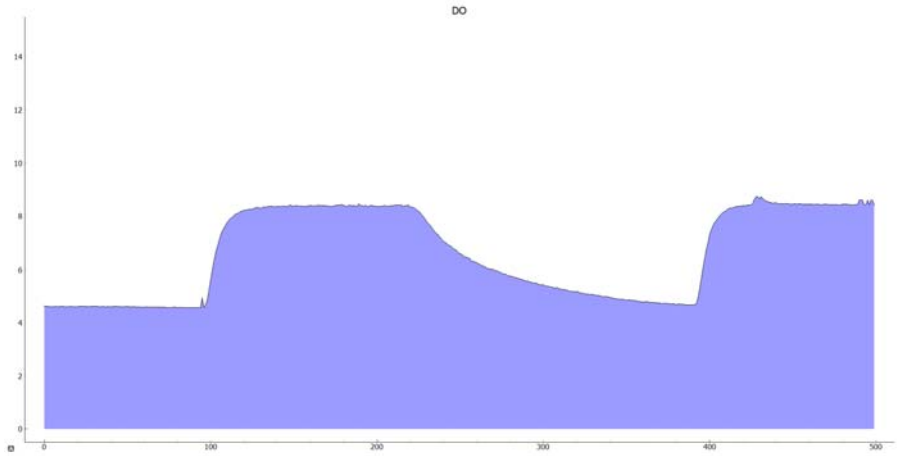


Figura 8.44 – Gráfica resultante de la medida de oxígeno disuelto.

Para comprender esta gráfica se detalla el proceso seguido durante la realización del muestreo:

1. El primer tramo de la gráfica se corresponde con el nivel de oxígeno disuelto cuando el agua ha estado en reposo durante un tiempo lo suficientemente largo como para considerar que el nivel de oxígeno disuelto se ha estabilizado.

2. Se comienza a oxigenar la muestra removiendo levemente hasta llegar a un valor en el que la medida se estabiliza.

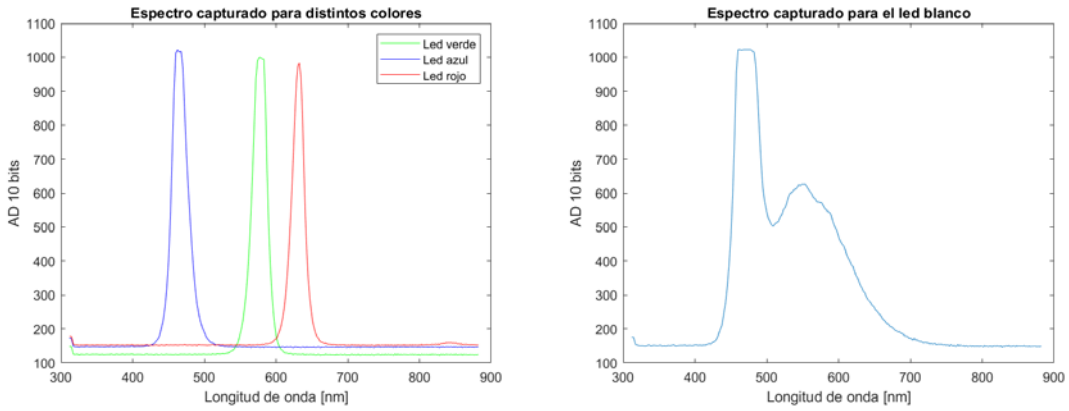
3. Al dejar de remover el agua el nivel de oxígeno comienza a descender poco a poco.

4. Se realiza de nuevo el proceso del punto 2. En este caso, una vez que se estabilizan las medidas, se realizan agitaciones rápidas y puntuales, comprobando que el sensor es capaz de registrarlas. Este último paso se corresponde con los picos finales que presenta la gráfica.

Analizando los resultados expuestos, se puede concluír que el sensor detecta diferentes niveles de oxígeno disuelto.

8.6.6. Medidas de absorbancia y espectro electromagnético.

Para comprobar el correcto funcionamiento del microespectrómetro se han llevado a cabo 2 pruebas. La primera de ellas ha consistido en analizar los espectros recogidos para leds de 4 colores diferentes. El resultado de esta prueba se puede ver en la imagen 8.45.



(a) Espectro de los leds R, G y B.

(b) Espectro del led blanco.

Figura 8.45 – Espectro obtenido para leds de diferentes colores.

Se ha comprobado que los espectros recogidos son correctos, ya que se detectaron picos aislados de alta intensidad a las longitudes de onda correspondientes para cada color. En el caso del color blanco la forma de onda obtenida también se corresponde con el espectro emitido por un led blanco.

Después de esta primera prueba, se han realizado medidas de absorbancia para distintas muestras, estableciendo como referencia la absorbancia de agua desionizada. Las muestras utilizadas se pueden ver en la imagen 8.46.

(a) Aspecto muestra $CuSO_4$.

(b) Aspecto muestra turbia.

(c) Aspecto muestra agua residual LAB.

(d) Aspecto muestra agua pura.

Figura 8.46 – Aspecto de las muestras utilizadas para el experimento de absorbancia.

El espectro capturado para cada una se muestra en la imagen 8.47.

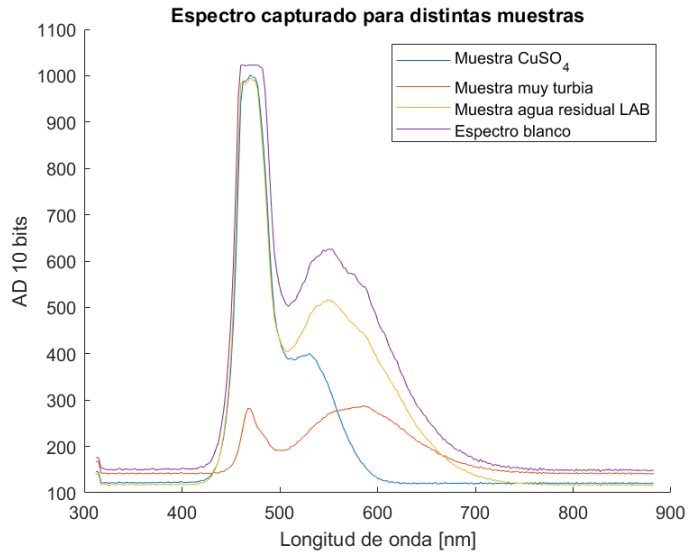


Figura 8.47 – Espectro resultante de las diferentes muestras.

Como se puede ver, el espectro recogido varía en función de la muestra, aunque esta diferencia se hace más notable si el espectro capturado se expresa en forma de absorbancia. La absorbancia de cada muestra se puede observar en la gráfica de la imagen 8.48.

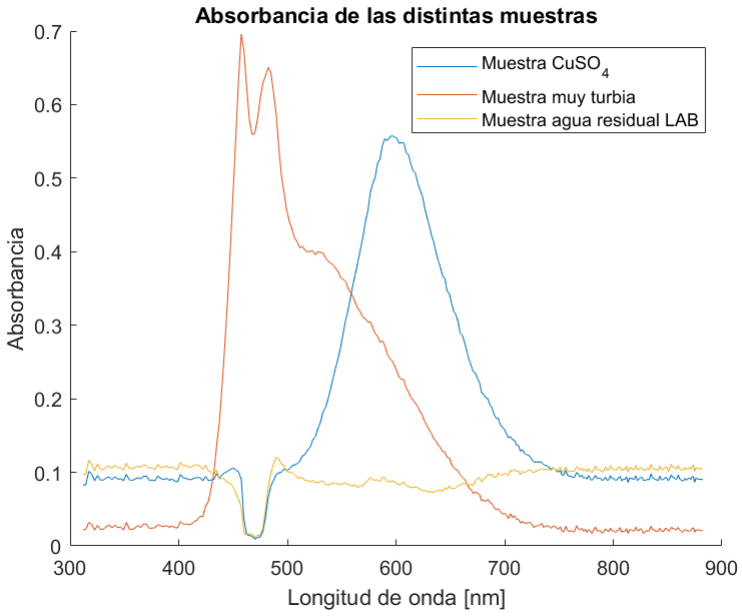


Figura 8.48 – Absorbancia de las diferentes muestras

Analizando la gráfica se pueden apreciar resultados lógicos:

- **Muestra de agua residual LAB:** es la muestra que presenta menor turbidez y al no poseer color no presenta una absorbancia relevante en ningún rango del espectro visible.
- **Muestra de agua turbia:** presenta una absorbancia elevada en el rango efectivo de la fuente de luz. El resultado es el esperado, debido al tono oscuro de la muestra.

- **Muestra $CuSO_4$:** presenta una baja absorbancia en longitudes de onda próximas a los 475 nm, sin embargo, para longitudes de onda superiores a los 550 nm se puede observar un gran ascenso de la absorbancia.

  El resultado obtenido es el esperado, ya que longitudes de onda próximas a los 475 nm se corresponden con el color azul. Esto quiere decir que la muestra no absorbe estas longitudes de onda, de ahí que la muestra presente un tono azulado.

Finalmente, se concluye que el sensor es capaz de diferenciar distintas muestras, resultando especialmente útil si el objetivo es detectar colores.

En esta sección se ha comprobado que todos los sensores utilizados en el proyecto pueden ofrecer datos relevantes para la caracterización de diferentes muestras químicas, abriendo la posibilidad de desarrollar sensores virtuales capaces de inferir parámetros desconocidos a partir de los estudiados en este proyecto.

Por último, destacar que a pesar de que todos los sensores ofrecen medidas útiles se pueden observar similitudes entre los resultados obtenidos con el microespectrómetro y el sensor de turbidez. Ambos permiten medir la turbidez de una muestra, adicionalmente, el microespectrómetro permite conocer el color de la muestra. Aunque es cierto que el microespectrómetro puede ofrecer información adicional si se compara con el sensor de turbidez, este último presenta un coste mucho menor.

No obstante, esto no significa que el microespectrómetro deba descartase completamente. Es posible que este pueda ofrecer información útil en función del parámetro objetivo que se busca predecir, por tanto, su ausencia en el diseño se recomienda únicamente para las primeras etapas de la implementación del sensor virtual. Esto, como se verá en la sección 8.7, permitirá reducir la dimensión de la capa de entrada de la red neuronal.

8.7. Desarrollo de un sensor virtual mediante una red perceptrón multicapa.

Como se detalló en la sección 8.5 no se pudo disponer de la suficiente cantidad de muestras con parámetros conocidos para el entrenamiento de la red neuronal, por lo que la implementación del sensor virtual no se pudo llevar a cabo. De todos modos, se explicarán algunos conceptos fundamentales acerca de la red MLP, además de realizar una serie de recomendaciones de cara a una posible implementación futura y realización de las primeras pruebas.

Como ya se mencionó la red que se propone utilizar es una de las más comunes, la MLP, que consta de 3 tipos de capas:

- **Capa de entrada:** esta capa contiene las neuronas que reciben los parámetros de entrada de la red, en este caso, los valores obtenidos de los sensores. Como ya se comentó en la sección 8.6 para las etapas iniciales de diseño e implementación de la red se recomienda considerar los siguientes parámetros de entrada:

- ◇ Temperatura.
- ◇ Turbidez.
- ◇ pH.
- ◇ Conductividad eléctrica.
- ◇ Potencial redox.
- ◇ Oxígeno disuelto.

En etapas más avanzadas podrían incorporarse los datos proporcionados por el micro-espectrómetro.

- **Capa oculta:** es la encargada de conectar las entradas con las salidas. La red puede presentar una o varias capas intermedias, lo mismo se aplica al número de neuronas. No existe un procedimiento claro que permita obtener la mejor configuración, por lo que el número de neuronas y capas ocultas deberá determinarse empíricamente.
- **Capa de salida:** esta capa contiene las neuronas que determinan la salida de la red. El número de neuronas de esta capa está determinado por el número de parámetros que se desea predecir, en este caso, se considerará una única neurona de salida.

En la imagen 8.49 se puede ver ilustrado un ejemplo de la estructura de la red mencionada, en la que se pueden observar 6 parámetros de entrada, una única capa oculta con 4 neuronas y una salida.

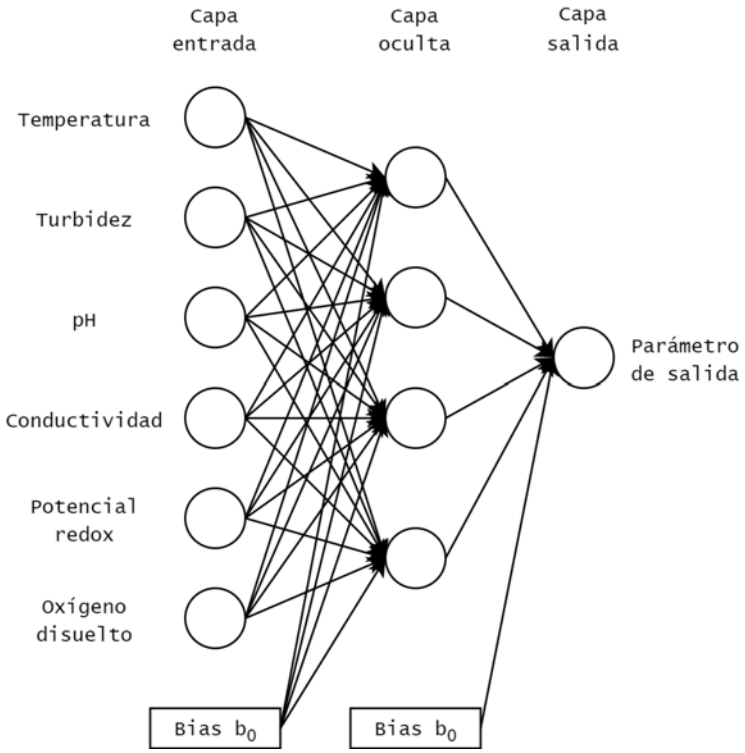


Figura 8.49 – Ejemplo del diseño de la red neuronal.

De nuevo, se insiste en que el diseño final de la red no tiene que ser exactamente el que se muestra en la imagen. Para obtener un diseño óptimo será necesario entrenar redes con diferentes configuraciones de capas y neuronas ocultas, seleccionando finalmente la red que mejor resultados proporcione.

En lo referente a la función de activación a asignar para las neuronas de la capa oculta y de salida se recomiendan las siguientes, teniendo en cuenta que la red se utilizará para regresión:

- Función lineal.

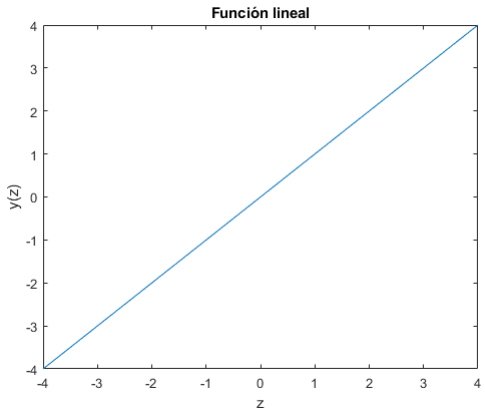


Figura 8.50 – Función de activación lineal.

$$y(z) = mz \tag{8.2}$$

- ReLU.

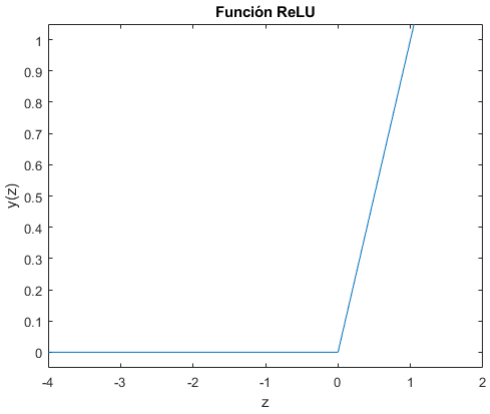


Figura 8.51 – Función de activación ReLU.

$$y(z) = max[0, z] \tag{8.3}$$

- Función tangente hiperbólica.

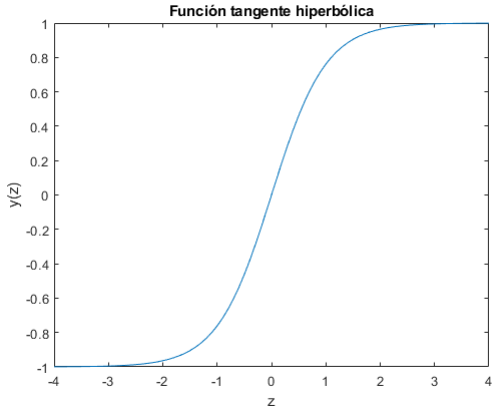


Figura 8.52 – Función de activación tangente hiperbólica.

$$y(z) = tanh(z) \tag{8.4}$$

- Función sigmoide.

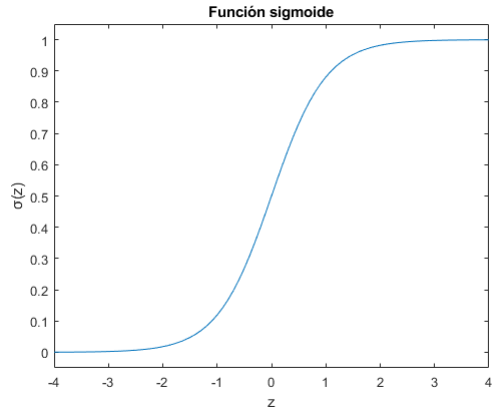


Figura 8.53 – Función de activación sigmoide.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{8.5}$$

Por último, en lo que respecta a la función de coste o *loss function* se recomienda utilizar MSE (*Mean Squared Error* o Error cuadrático medio), la más utilizada habitualmente en problemas de regresión, aunque también se pueden utilizar otras como MAE (*Mean Absolute Error* o Error absoluto medio) o RMSE (*Root Mean Square Error* o Raíz del error cuadrático medio). La función MSE calcula el promedio de los errores al cuadrado, siendo el error la diferencia entre el valor real y el predicho.

9 ORDEN DE PRIORIDAD ENTRE LOS DOCUMENTOS

1. Memoria
2. Anexos
3. Planos
4. Mediciones
5. Presupuesto
6. Pliego de Condiciones

TÍTULO: **CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DE DISPOSITIVOS HARDWARE DE BAJO COSTE**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2020

AUTOR: EL ALUMNO

Fdo.: YAGO GÓMEZ CASTRO

Índice del documento ANEXOS

10 DOCUMENTACIÓN DE PARTIDA

10.1. Asignación del Trabajo Fin de Grado.

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Gómez Castro,Yago*
APELIDOS E NOME:

Fecha de Solicitud: OCT2020
Fecha de Solicitude:

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:Caracterización de aguas residuales a través de dispositivos hardware de bajo coste

Número TFG: *770G01A193*

TUTOR:(Titor) *Casteleiro Roca,José Luis*

COTUTOR/CODIRECTOR: *Francisco Zayas Gato*

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento:

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Miercoles, 7 de Octubre del 2020

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Gómez Castro,Yago

DESCRIPCIÓN Y OBJETIVO:OBJETO:
El objeto de este Trabajo Final de Grado es el de estudiar, primeramente, aquellos parámetros medidos en el agua procesada por las Estaciones de Depuración de Agua Residual. A continuación, se implementarán distintos tipos de sensores de bajo coste para caracterizar aguas residuales. A partir de las medidas realizadas por estos sensores implementados, se estudiará la posibilidad diseñar e implementar un sensor virtual para determinar la naturaleza anómala de las aguas residuales.

ALCANCE:
Estudiar los parámetros medidos sobre aguas residuales en Estaciones de Depuración de Agua Residual.
Estudio de los distintos tipos de sensores de bajo coste que permitan caracterizar una muestra de agua.
Desarrollo de un procedimiento para realizar las medidas.
Análisis de la correlación entre las variables en las medidas realizadas.
Estudio de la posibilidad de desarrollar sensores virtuales.

11 CÁLCULOS

En este apartado se presentan los cálculos realizados para convertir los valores de tensión entregados por los sensores a las unidades físicas correspondientes.

Para todas las conversiones se utiliza el valor medio de un array de 10 medidas como se explica en la sección 8.4.1. Este valor será digital, por lo tanto se le aplica una constante que lo convierte a voltios. El valor de la constante será producto de dividir la tensión de referencia (5V) entre la resolución del convertidor A/D (10 bits).

Las dos operaciones se muestran en forma de ecuación en 11.1.

$$Lectura\ (V) = \frac{5V}{2^{10\ bits}} \times Valor\ medio(bits) \tag{11.1}$$

Cabe mencionar que para la conversión realizada en el sensor de conductividad eléctrica el valor utilizado no es el de tensión, sino el valor entregado por el convertidor A/D, esto es, un valor entre 0 y 1024.

11.1. Conversión a NTU.

Para obtener valores de NTU a partir del voltaje que proporciona el sensor se necesita saber cómo varía la tensión en función de la turbidez, dicha relación se muestra en la gráfica de la figura 11.1.

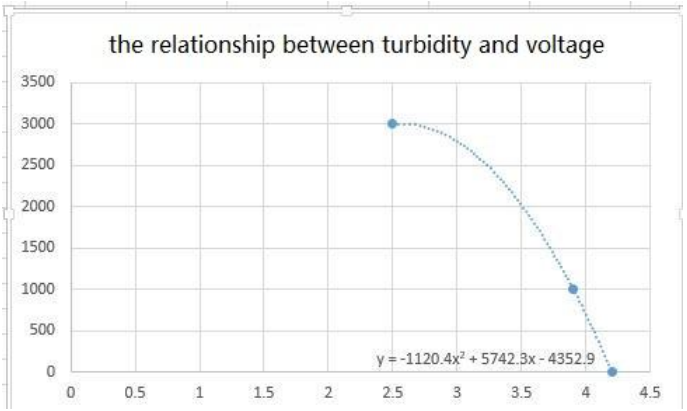


Figura 11.1 – Gráfica para conversión voltaje - NTU [6].

La ecuación de la curva mostrada en la gráfica 11.1 se puede ver más claramente en la ecuación 11.2

$$Turbidez\ [NTU] = -1120{,}4 \times {V_{sensor}}^{2} + 5742{,}3 \times V_{sensor} - 4352{,}9 \tag{11.2}$$

11.2. Conversión a pH.

Para la conversión de los valores de tensión entregados por el sensor a valores expresados en unidades de pH se utiliza la ecuación 11.3.

$$pH = 3{,}5 \times V_{out} + \textit{Offset} \tag{11.3}$$

El parámetro *"Vout"* representa la tensión entregada por el sensor, mientras que *"Offset"* se utiliza para el ajuste en caso de que el dispositivo necesite calibración.

La tensión de salida del sensor cumple con la relación mostrada en la ecuación 11.4, que a su vez se obtiene a partir de las ecuaciones 11.5 y 11.6.

$$V_{out} = 5V \times \frac{30K}{75K} - (1 + \frac{2{,}2K + 5K \times \alpha}{1k + 5k \times (1 - \alpha)}) \times V_{in} \tag{11.4}$$

$$V_{out} = 5V \times \frac{30K}{75K} - V_{out\ NO\ INVERSOR} \tag{11.5}$$

$$V_{out\ NO\ INVERSOR} = V_{in} \times (1 + \frac{2{,}2K + 5K \times \alpha}{1k + 5k \times (1 - \alpha)}) \tag{11.6}$$

Estas dos últimas relaciones representan las ganancias del circuito mostrado en la imagen 11.2. Este es una versión simplificada del esquemático facilitado por el fabricante en su web [7].

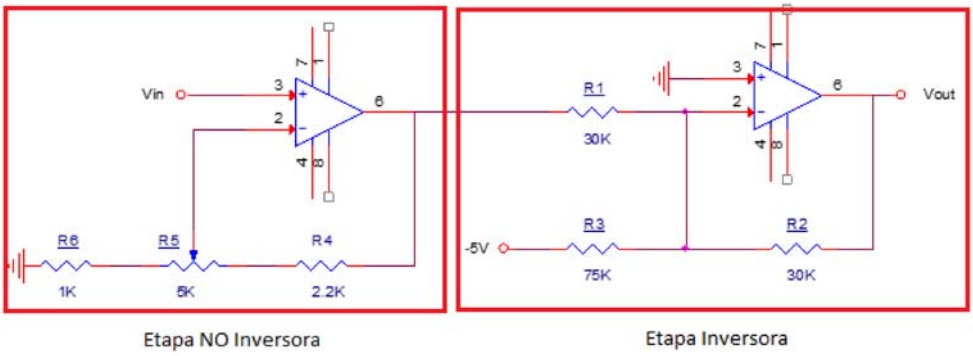


Figura 11.2 – Circuito de amplificación del sensor de pH.

El parámetro *"Vin"* que aparece en la imagen 11.2 y en la ecuación 11.4 representa la tensión que entrega el electrodo. Como se puede ver esta tensión se ve afectada por la ganancia de la etapa inversora y no inversora, siendo esta última variable en función de la posición del potenciómetro, representada por el parámetro α.

Aunque la posición del potenciómetro, y por tanto la ganancia de la etapa no inversora, se desconoce, esta se puede calcular utilizando la tabla 11.1, en la que se proporcionan las tensiones que entrega el electrodo para distintos valores de pH.

Voltaje (mV)	pH	Voltaje (mV)	pH
414,12	0,00	-414,12	14,00

354,96	1,00	-354,96	13,00
295,80	2,00	-295,80	12,00
236,64	3,00	-236,64	11,00
177,48	4,00	-177,48	10,00
118,32	5,00	-118,32	9,00
59,16	6,00	-59,16	8,00
0,00	7,00	0,00	7,00

Tabla 11.1 – Tabla con los valores de tensión del electrodo ante distintos valores de pH.

Ahora, tomando un valor de pH con su correspodiente valor de tensión y sustituyendo en las ecuaciones 11.3 y 11.4, se puede obtener la ganancia de la etapa no inversora.

$$14 = 3{,}5 \times (5V \times \frac{30K}{75K} - (1 + \frac{2{,}2K + 5K \times \alpha}{1k + 5k \times (1-\alpha)}) \times -0{,}414V) \xrightarrow{Despejando\ \alpha} \alpha = 0{,}86 \quad (11.7)$$

Finalmente, en la ecuación 11.8 se muestra una versión simplificada de la función de transferencia del circuito una vez hallada la ganancia del amplificador no inversor.

$$V_{out} = 2 - 4{,}82 \times V_{in} \quad (11.8)$$

Como se puede ver, los 5 voltios aportados en una de las etapas de amplificación sirven para elevar la curva y hacer que la salida del sensor sea siempre positiva.

11.3. Conversión a EC.

En la conversión de tensión a unidades de conductividad eléctrica (mS/cm en este caso) intervienen diferentes coeficientes y relaciones. En primer lugar es necesario conocer la función de transferencia del circuito que se muestra en la ecuación 11.9.

$$V_{outAD} = \frac{R10}{R} \times V_{ref} \quad (11.9)$$

Siendo *"Vref"* una tensión entorno a 200 mV, *"R10"* una resistencia de aproximadamente 820Ω, *"Vout"* la tensión de salida del sensor y *"R"* la resistencia del electrodo cuando se introduce en una solución. Este último parámetro varía en función de la solución en la que sea sumergido el electrodo, permitiendo medir la conductividad del líquido.

Para despejar el parámetro desconocido *"R"* se introducen las ecuaciones 11.10 y 11.11 en la función de transferencia del sensor.

$$R = Resistividad \times \frac{Longitud}{Sección} \quad (11.10)$$

$$Conductividad = \frac{1}{Resistividad} \tag{11.11}$$

La ecuación 11.12 es el resultado de introducir estas relaciones. Además de aplicar dichas relaciones, se ha de tener en cuenta que *"Vout"* representa el valor analógico (en bits) entregado por el convertidor A/D. La razón de que se utilice el valor analógico del convertidor en lugar de la tensión no se detalla en la hoja de características del fabricante, simplemente se estipula que la conversión debe realizarse de este modo. Además de esto, se añade un coeficiente de compensación de la temperatura, ya que esta influye cuando se trata de medir conductividad

El parámetro *"Kvalue"* que aparece al introducir las ecuaciones de resistividad y conductividad, es una constante cuyo valor depende del tipo de electrodo que se utilice. Para el sensor del proyecto su valor es 0,995. Por último, se multiplica por 1000 para obtener la conductividad en mS/cm.

$$EC\ [mS/cm] = 1000 \times \frac{V_{out} \times V_{ref}}{R10} \times \frac{K_{value}}{\textit{TempCoef}} \tag{11.12}$$

El coeficiente de compensación de temperatura *"TempCoef"* se calcula con la ecuación 11.13.

$$\textit{TempCoef} = 1 + \alpha \times (Temperatura - 25) \tag{11.13}$$

El valor del parámetro α de la ecuación 11.13 varía según el tipo de solución, tal y como se explica en [12]. Sin embargo, para facilitar el cálculo de la conductividad se le asignará un valor de 0,02. Si fuesen necesarios resultados más precisos se podría ajustar el valor de este coeficiente según el tipo de solución que se esté analizando en cada momento.

11.4. Conversión a ORP.

Para obtener el potencial RedOx de una solución a partir del voltaje de salida del sensor basta con analizar el esquemático facilitado por el fabricante en [9]. Si se tiene en cuenta la etapa de ganancia de dicho esquemático, representada en la imagen 11.3 y cuya función de transferencia se muestra en la ecuación 11.14, se obtiene la relación que convierte la tensión del sensor en potencial RedOx.

$$ORP\ [mV] = (5 \times \frac{30K}{75K} - V_{in} - \textit{ORP\_offset}) \times 1000 \tag{11.14}$$

El parámetro *"Vin"* representa la tensión entregada por el electrodo que incorpora el sensor y el parámetro *"ORP_offset"* se incorpora a la ecuación para facilitar posibles calibraciones.

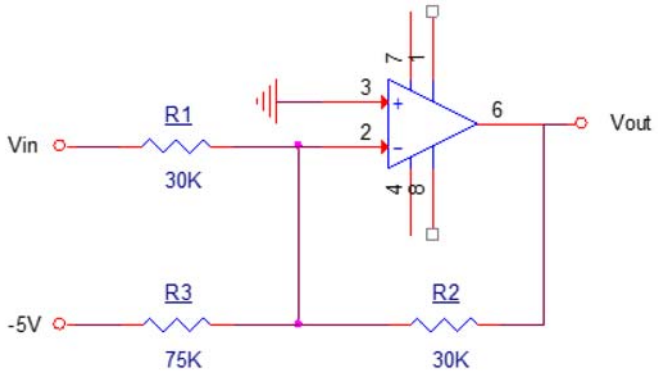


Figura 11.3 – Etapa de amplificación del sensor de ORP.

11.5. Conversión a DO.

Antes de realizar el cálculo del oxígeno disuelto es necesario entender dos conceptos:

- **Voltaje de saturación**: es el voltaje máximo entregado por el sensor a una temperatura determinada y se corresponde con el valor máximo de oxígeno disuelto. En la imagen 11.4 se puede ver una gráfica que representa lo explicado.

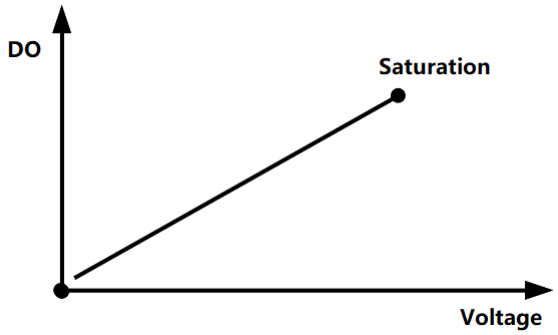


Figura 11.4 – Relación entre voltaje y oxígeno disuelto [47].

- **Variación del voltaje de saturación con la temperatura**: el voltaje de saturación explicado anteriormente varía en función de la temperatura siguiendo una relación similar a la que se muestra en la imagen 11.5.

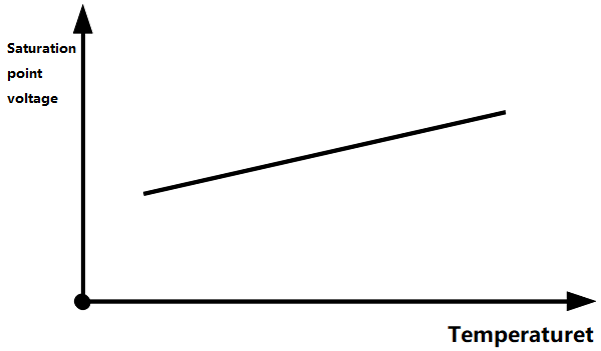


Figura 11.5 – Variación del voltaje de saturación en función de la temperatura [48].

Una vez comprendida la relación entre el voltaje entregado por el sensor y el oxígeno disuelto se establece la ecuación 11.15, que entrega el oxígeno disuelto de una solución en mg/L.

$$DO\ [mg/L] = SaturaciónDO\_Temperatura[mg/L] \times \frac{V_{out}[V]}{SaturaciónV\_Temperatura[V]} \quad (11.15)$$

En la tabla 11.2 se muestran valores de saturación de oxígeno disuelto para diferentes temperaturas. Al parámetro *"SaturaciónDO_Temperatura"* de la ecuación 11.15 se le asignará el coeficiente de saturación de DO que corresponda en función de la temperaura medida.

ºC	DO (mg/L)	ºC	DO (mg/L)	ºC	DO (mg/L)
0	14,60	16	9,86	32	7,30
1	14,22	17	9,64	33	7,17
2	13,80	18	9,47	34	7,06
3	13,44	19	9,27	35	6,94
4	13,08	20	9,09	36	6,84
5	12,76	21	8,91	37	6,72
6	12,44	22	8,74	38	6,60
7	12,11	23	8,57	39	6,52
8	11,83	24	8,41	40	6,40
9	11,56	25	8,25	41	6,33
10	11,29	26	8,11	42	6,23
11	11,04	27	7,96	43	6,13
12	10,76	28	7,83	44	6,06
13	10,54	29	7,68	45	5,97
14	10,31	30	7,56	46	5,88
15	10,06	31	7,43	47	5,79

Tabla 11.2 – Tabla con los valores de saturación de DO a diferentes temperaturas.

En el caso del voltaje de saturación, representado en la ecuación de conversión de DO como *"SaturaciónV_Temperatura"*, se han obtenido las tensiones entregadas por el sensor a diferentes temperaturas. El sensor no se ha introducido en ninguna muestra, sino que se ha dejado al aire, considerando que así se obtienen valores de saturación.

Los valores obtenidos del proceso anterior se muestran en la tabla 11.3. Esta tabla, junto con la 11.2, se incorporan en el código de Arduino del anexo 13.

ºC	Voltaje de saturación
8	968 mV
9,5	1005 mV
10	1026 mV

11	1044 mV
12	1072 mV
13	1090 mV
14	1110 mV
15	1150 mV
16	1171 mV
17,25	1225 mV
18,25	1276 mV
19	1335 mV
20,25	1363 mV
21,5	1417 mV

Tabla 11.3 – Tabla con los valores de saturación de voltaje a diferentes temperaturas.

Si se grafican las tensiones obtenidas frente a sus correspodientes temperaturas se obtiene la gráfica 11.6, en donde se puede comprobar que la relación mostrada en la imagen 11.5 se cumple.

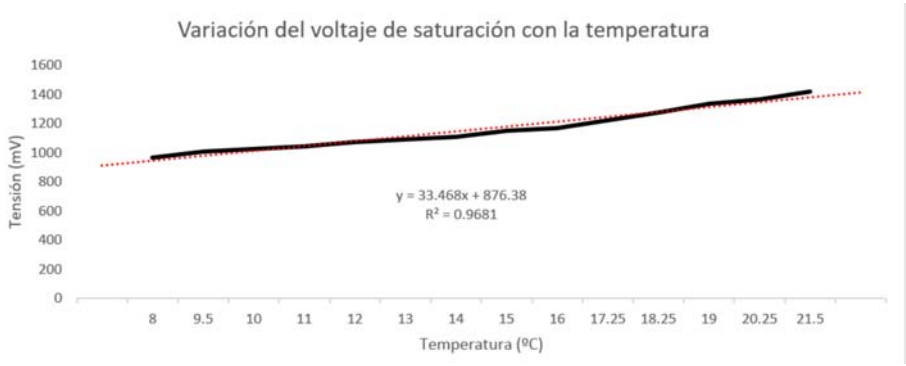


Figura 11.6 – Tensiones de saturación del sensor de DO a distintas temperaturas.

La curva negra se obtiene al unir los voltajes de la tabla 11.6, mientras que la línea roja discontinua es la recta que mejor se aproxima a dicha curva. La ecuación que define esta recta, representada en la relación 11.16, será la utilizada para obtener el voltaje de saturación aproximado para distintas temperaturas.

$$SaturaciónV\_Temperatura\ [mV] = 33{,}468 \times Temperatura + 876{,}38 \tag{11.16}$$

11.6. Conversión de los píxeles del microespectrómetro a longitudes de onda [22].

Para obtener la longitud de onda que capta cada uno de los 288 píxeles que componen el dispositivo se necesita aplicar un polinomio de 5º grado, cuyos coeficientes se facilitan en

función del número de serie del sensor. En la imagen 11.7 se muestra el número de serie asociado al microespectrómetro utilizado en el proyecto.

Figura 11.7 – Número de serie del microespectrómetro.

Los coeficientes que corresponden al número de serie **18A00161** se muestran en la tabla 11.4.

Coeficiente	Valor
A0	3,09127381 $\times 10^{2}$
B1	2,69111242 $\times 10^{0}$
B2	-1,156056159 $\times 10^{-3}$
B3	-6,599636705 $\times 10^{-6}$
B4	2,507372316 $\times 10^{-9}$
B5	1,735837823 $\times 10^{-11}$

Tabla 11.4 – Coeficientes del polinomio de conversión.

Aplicando los coeficientes al polinomio de la ecuación 11.17 se obtienen las diferentes longitudes de onda. El parámetro *"pix"* corresponde al número de píxel, por lo que se aplicará el polinomio para valores de *"pix"* entre 1 y 288.

$$Wavelength\ [nm] = A0 + B1 \times pix + B2 \times pix^2 + B3 \times pix^3 + B4 \times pix^4 + B5 \times pix^5 \quad (11.17)$$

12 INSTALACIÓN DE SOFTWARE Y LIBRERÍAS

Para la utilización del sistema de adquisición de los datos es necesario instalar el IDE de Arduino y Python 3, si es posible la versión 3.4 o cualquiera superior ya que de esta forma se instalará automaticamente *pip* (instalador de paquetes de python).

Esta herramienta facilita la instalación de librerías para python desde la ventana de coman-

dos del sistema.

12.1. Librerías para Arduino.

Para el código implementado en Arduino es necesario instalar 3 librerías, 2 de ellas son necesarias para el correcto funcionamiento del sensor de temperatura mientras que la otra se utiliza para el sensor de DO.

- Librería OneWire: como se explico en la seccion 8.1.1 el sensor de temperatura utiliza el bus OneWire. La instalación de esta librería permite utilizar más comodamente este sensor al no ser necesario que conozcamos cómo funciona este protocolo de comunicación.
- Librería DallasTemperature: las funciones que incorpora esta librería facilitan la comunicación con el bus OneWire y los sensores conectados a este.
- Librería EEPROM: esta librería permite acceder a la memoria EEPROM de Arduino. Su instalación es necesaria ya que en esta memoria se almacenan valores de calibración del sensor de DO.

Para instalar las anteriores librerías basta con seguir los pasos indicados a continuación.

- En el IDE de Arduino ir a **Herramientas** y *Administrar Bibliotecas...*
- Se abrirá el gestor de librerías mostrado en la figura 12.1. Buscar e instalar las 3.

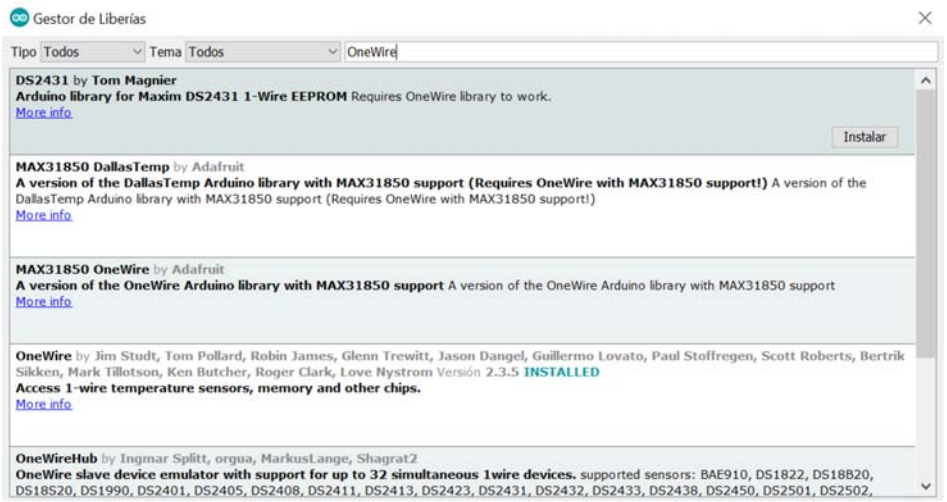


Figura 12.1 – Gestor de librerías de Arduino.

12.2. Librerías para Python.

El código desarrollado en python utiliza las librerías nombradas en la siguiente lista:

- *PySerial*: librería utilizada para la comunicación serie con el microcontrolador.
- *time* y *datetime*: librerías que permiten trabajar con la fecha actual y tiempos.

- *csv*: librería que permite tratar con archivos de extensión CSV de manera más sencilla.
- *numpy*: librería básica para operaciones y conversiones matemáticas, además otras librerías más complejas hacen uso de *Numpy*.
- *pyqtgraph*: librería gráfica utilizada para el desarrollo de la GUI en la que se muestran todos los datos. Se decidió utilizar esta librería en lugar de otras más comunes como puede ser *Matplotlib* debido a la alta tasa de refresco que permite *pyqtgraph*.

  Esta librería está basada en PyQt4, PySide y Numpy por lo que para su correcto funcionamiento, además de instalar la propia librería, es necesario instalar las citadas.
- *urllib y requests*: librerías utilizadas para la comunicación con ThingSpeak. Ambas sirven para simplicar la comunicación mediante protocolo HTTP.

La instalación de todas las librerías se puede realizar desde la ventana de comandos del sistema utilizando el comando mostrado en la figura 12.2.

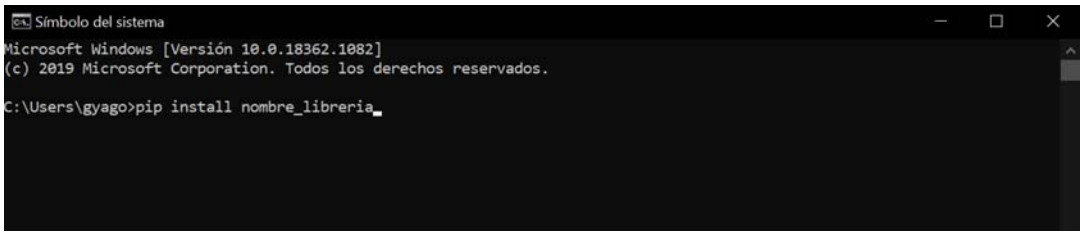


Figura 12.2 – Ventana de comandos del sistema con el comando instalador de librerías.

13 CÓDIGO DE PROGRAMACIÓN PARA ARDUINO

En esta sección se muestra el código desarrollado en Arduino para la lectura de los sensores y su comunicación con el PC a través del puerto serie.

13.1. Código completo.

```
   breakatwhitespace
// Librerias.
#include <OneWire.h>              // Protocolo de transmision del DS18B20.
#include <DallasTemperature.h>
//#include <avr/pgmspace.h>
//#include <EEPROM.h>
```

```
// ------------------------------------------------------------ //
// ------------------- VARIABLES GLOBALES --------------------- //
// ------------------------------------------------------------ //


bool inicio = 0; // Variable que controla si se muestrea o no
double baudRate = 115200;

// Parametros generales.
#define sample_time 25           // ms. Con 33 ms x 10 + 671 enviamos datos cada 1s.
#define ArrayLenth  10           // Numero de muestras tomadas en cada ronda.
int ArrayIndex = 0;              // Inicializamos a 0 para primer caso.
unsigned long last_sample;       // Guarda ultimo valor de millis() para controlar
    muestreo.

// Sensor de temperatura.
#define pinDatosDQ 9    // 1-Wire bus pin
float Temperatura;
// Instancia las clases OneWire y DallasTemperature.
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);

// Parametros para el sensor de turbidez.
#define turbidity A0
#define Turbidez_offset   0.18
int turbidity_array[ArrayLenth];
double avg_turbidity;
float NTU;

// Parametros para el sensor de pH
#define pH A1
#define pH_offset   0.00
int pH_array[ArrayLenth];
double avg_pH;
float pH_value;

// Parametros para el sensor de EC (+ temperatura)
#define EC A2
#define R10 820.0
#define ECREF 200.0
int EC_array[ArrayLenth];
double avg_EC;
float EC_value;
float TempCoef;
float kvalue = 0.995;

// Parametros para el sensor de ORP
```

```
#define ORP A3
#define ORP_offset   0
int ORP_array[ArrayLenth];
double avg_ORP;
float ORP_value;

// Parametros para el sensor de DO
#define DO A4
int DO_array[ArrayLenth];
double avg_DO;
float SaturationDOVoltage;
float DO_value;
// Concentraciones de saturacion de DO a varias temperaturas. Para calibraci n.
const float SaturationValueTab[41] PROGMEM = {
  14.46, 14.22, 13.82, 13.44, 13.09,
  12.74, 12.42, 12.11, 11.81, 11.53,
  11.26, 11.01, 10.77, 10.53, 10.30,
  10.08, 9.86,  9.66,  9.46,  9.27,
  9.08,  8.90,  8.73,  8.57,  8.41,
  8.25,  8.11,  7.96,  7.82,  7.69,
  7.56,  7.43,  7.30,  7.18,  7.07,
  6.95,  6.84,  6.73,  6.63,  6.53,
  6.41,
};

// Par metros para el microespectrometro
#define SPEC_TRG         8
#define SPEC_ST          7
#define SPEC_CLK         6
#define SPEC_VIDEO       A5 // Salida datos espectr.
//#define WHITE_LED      5  // LED microespectrometro.
#define LASER_404        4

#define SPEC_CHANNELS    288 // 288 canales
int data[SPEC_CHANNELS];     // Array datos espectr.



// ------------------------------------------------------------ //
// ------------------------- SETUP ---------------------------- //
// ------------------------------------------------------------ //



void setup()
{
  Serial.begin(baudRate);
  sensorDS18B20.begin(); // Iniciamos el bus 1-Wire

  pinMode(turbidity,  INPUT);
```

```
  pinMode(pH,  INPUT);
  pinMode(EC,  INPUT);
  pinMode(ORP,  INPUT);
  pinMode(DO,  INPUT);
  pinMode(SPEC_CLK, OUTPUT);
  pinMode(SPEC_ST, OUTPUT);
  pinMode(LASER_404, OUTPUT);
  //pinMode(WHITE_LED, OUTPUT);

  digitalWrite(SPEC_CLK, HIGH); // Set SPEC_CLK High
  digitalWrite(SPEC_ST, LOW); // Set SPEC_ST Low

  last_sample = millis();
}



// ------------------------------------------------- //
// --------------------- LOOP ---------------------- //
// ------------------------------------------------- //



void loop()
{

  // Lectura puerto serie para inicio del muestreo.
  if (inicio == 0 && Serial.available() > 0)
  {
    if (Serial.read() == '1')
    {
      inicio = 1;
    }
  }

  // Caso de desborde. Cuando millis() llega a su maximo reinicia a 0.
  if (millis() < last_sample){ last_sample = 0; }

  // Muestreo.
  if ((millis() - last_sample >= sample_time) && inicio == 1)
  {
    last_sample = millis(); // Actualizacion para siguiente ciclo.

    turbidity_array[ArrayIndex] = analogRead(turbidity);
    pH_array[ArrayIndex] = analogRead(pH);
    EC_array[ArrayIndex] = analogRead(EC);
    ORP_array[ArrayIndex] = analogRead(ORP);
    DO_array[ArrayIndex] = analogRead(DO);

    ArrayIndex++;
```

```
  }


  // C lculo de la media y envio de datos
  if (ArrayIndex == ArrayLenth)
  {
    // Se toma una sola medida de la temperatura para no ralentizar el bucle del
        resto de medidas.
    // Dentro de la libreria DallasTemperature hay delays en funcion de la
        resolucion, esto es necesario para que el integrado funcione correctamente.
    // La velocidad de lectura es mejorable si se reduce la resolucion (9 - 10 - 11
        - 12 bits)
    sensorDS18B20.requestTemperatures();
    Temperatura = sensorDS18B20.getTempCByIndex(0);

    readSpectrometer();

    avg_turbidity = avg_array(turbidity_array) * (5.0 / 1024.0) + Turbidez_offset;
    NTU = -1120.4 * avg_turbidity * avg_turbidity + 5742.3 * avg_turbidity - 4352.9;

    avg_pH = avg_array(pH_array) * (5.0 / 1024.0);
    pH_value = 3.5 * avg_pH + pH_offset;

    avg_EC = avg_array(EC_array); // OJO, para el calculo de la conductividad se
        utiliza el valor directo del AD sin convertir el valor a tensi n a
        diferencia del resto de sensores.
    avg_EC = 1000 * avg_EC / R10 / ECREF;
    TempCoef = 1.0 + 0.02 * (Temperatura - 25.0);          // Coeficiente de
        compensacion por temperatura. estaba 0.0185 pero para el agua es 0.2, probar
        , si no ajusta bien usar el recomendado por DFROBOT.
    EC_value = avg_EC * kvalue / TempCoef;

    avg_ORP = avg_array(ORP_array) * (5.0 / 1024.0);
    ORP_value = (2.0 - avg_ORP - ORP_offset) * 1000;

    avg_DO = avg_array(DO_array) * (5000.0 / 1024.0);
    SaturationDOVoltage = 33.468 * Temperatura + 876.38;
    DO_value =  pgm_read_float_near( &SaturationValueTab[0] + (int)(Temperatura +
        0.5) ) * avg_DO / SaturationDOVoltage;

    printData(); // Se envian datos por puerto serie

    // Actualizaci n de variables de ciclo.
    ArrayIndex = 0;
    inicio = 0;
  } // end if
} // loop
```

```
// ------------------------------------------------------------ //
// ------------------------ FUNCIONES ------------------------- //
// ------------------------------------------------------------ //



// FUNCION PARA LECTURA MICRO.
void readSpectrometer() {

  int delayTime = 1; // delay time 1 us

  // Flanco de subida en ST para comenzar muestreo
  digitalWrite(SPEC_CLK, LOW);
  delayMicroseconds(delayTime);
  digitalWrite(SPEC_CLK, HIGH);
  delayMicroseconds(delayTime);
  digitalWrite(SPEC_CLK, LOW);
  digitalWrite(SPEC_ST, HIGH);
  delayMicroseconds(delayTime);

  // Se configura el tiempo de integracion
  for (int i = 0; i < 150; i++) {
    digitalWrite(SPEC_CLK, HIGH);
    delayMicroseconds(delayTime);
    digitalWrite(SPEC_CLK, LOW);
    delayMicroseconds(delayTime);
  }

  //ST a estado bajo
  digitalWrite(SPEC_ST, LOW);

  //La adquisicion necesita 87 ciclos de reloj
  for (int i = 0; i < 86; i++) {
    digitalWrite(SPEC_CLK, HIGH);
    delayMicroseconds(delayTime);
    digitalWrite(SPEC_CLK, LOW);
    delayMicroseconds(delayTime);
  }

  //Read from SPEC_VIDEO
  for (int i = 0; i < SPEC_CHANNELS; i++) {
    data[i] = analogRead(SPEC_VIDEO);
    digitalWrite(SPEC_CLK, HIGH);
    delayMicroseconds(delayTime);
    digitalWrite(SPEC_CLK, LOW);
    delayMicroseconds(delayTime);
  }

  digitalWrite(SPEC_CLK, HIGH); // Set SPEC_CLK High
```

```
}

// PRINT DATA.
void printData() {
    // Escritura en puerto serie.
    Serial.print(Temperatura);
    Serial.print(",");
    Serial.print(NTU, 2);
    Serial.print(",");
    Serial.print(pH_value, 2);
    Serial.print(",");
    Serial.print(EC_value, 2);
    Serial.print(",");
    Serial.print(ORP_value, 2);
    Serial.print(",");
    Serial.print(DO_value, 2);
    Serial.print(",");

    for (int i = 0; i < SPEC_CHANNELS; i++) {
      Serial.print(data[i]);
      Serial.print(',');
    }

    Serial.println();
}

// FUNCION CALCULO DE LA MEDIA
double avg_array(int* arr)
{
  int i;
  int max, min;
  double avg;
  long amount = 0;

  // Media (sin maximo ni minimo)

  if (arr[0] < arr[1])
  {
    min = arr[0]; max = arr[1];
  }
  else
  {
    min = arr[1]; max = arr[0];
  }
  for (i = 2; i < ArrayLenth; i++)
  {
    if (arr[i] < min) {    // Nuevo minimo
      amount += min;       // A adir minimo anterior
      min = arr[i];        // Actualizar minimo
    }
```

```
    else
    {
      if (arr[i] > max) { // Nuevo maximo
        amount += max;    // A adir maximo anterior
        max = arr[i];     // Actualizar maximo
      }
      else
      {
        amount += arr[i]; // No es ni maximo ni minimo
      }
    } //else
  } //for
  avg = (double)amount / (ArrayLenth - 2);

  return avg;
}
```

Código 13.1: Código completo para el uso de Arduino como DAQ.

13.1.1. Función para lectura del microespectrómetro.

```
   breakatwhitespace
void readSpectrometer() {

  int delayTime = 1; // delay time 1 us

  // Flanco de subida en ST para comenzar muestreo
  digitalWrite(SPEC_CLK, LOW);
  delayMicroseconds(delayTime);
  digitalWrite(SPEC_CLK, HIGH);
  delayMicroseconds(delayTime);
  digitalWrite(SPEC_CLK, LOW);
  digitalWrite(SPEC_ST, HIGH);
  delayMicroseconds(delayTime);

  // Se configura el tiempo de integracion
  for (int i = 0; i < 150; i++) {
    digitalWrite(SPEC_CLK, HIGH);
    delayMicroseconds(delayTime);
    digitalWrite(SPEC_CLK, LOW);
    delayMicroseconds(delayTime);
  }

  //ST a estado bajo
  digitalWrite(SPEC_ST, LOW);

  //La adquisicion necesita 87 ciclos de reloj
```

```
  for (int i = 0; i < 86; i++) {
    digitalWrite(SPEC_CLK, HIGH);
    delayMicroseconds(delayTime);
    digitalWrite(SPEC_CLK, LOW);
    delayMicroseconds(delayTime);
  }

  //Read from SPEC_VIDEO
  for (int i = 0; i < SPEC_CHANNELS; i++) {
    data[i] = analogRead(SPEC_VIDEO);
    digitalWrite(SPEC_CLK, HIGH);
    delayMicroseconds(delayTime);
    digitalWrite(SPEC_CLK, LOW);
    delayMicroseconds(delayTime);
  }

  digitalWrite(SPEC_CLK, HIGH); // Set SPEC_CLK High

}
```

13.1.2. Cálculo de la media.

```
   breakatwhitespace
double avg_array(int* arr)
{
  int i;
  int max, min;
  double avg;
  long amount = 0;

  // Media (sin maximo ni minimo)

  if (arr[0] < arr[1])
  {
    min = arr[0]; max = arr[1];
  }
  else
  {
    min = arr[1]; max = arr[0];
  }
  for (i = 2; i < ArrayLenth; i++)
  {
    if (arr[i] < min) {    // Nuevo minimo
      amount += min;       // A adir minimo anterior
      min = arr[i];        // Actualizar minimo
```

```
    }
    else
    {
      if (arr[i] > max) { // Nuevo maximo
        amount += max;    // A adir maximo anterior
        max = arr[i];     // Actualizar maximo
      }
      else
      {
        amount += arr[i]; // No es ni maximo ni minimo
      }
    } //else
  } //for
  avg = (double)amount / (ArrayLenth - 2);

  return avg;
}
}
```

14 CÓDIGO DE PROGRAMACIÓN PARA PYTHON

En esta sección se muestra el código desarrollado en Python para la lectura de los datos recogidos por Arduino. Los datos se grafican en tiempo real, se guardan y se suben a la nube.

14.1. Código completo.

```
   breakatwhitespace
# El modulo lee datos de puerto serie separados por comas hasta \n
# (Temperatura, Turbidez, pH, EC, ORP, DO, 289 valores de espectrometro)
# Los datos son guardados en un archivo .CSV junto con el tiempo (h:m:s)
# Se grafica y se sube a la nube

import serial
import time
import datetime
import csv
import numpy as np
from pyqtgraph.Qt import QtGui, QtCore
import pyqtgraph as pg
import urllib.request
import requests


##############################################################################

############### Clase para la comunicaci n Arduino - PC ###############

##############################################################################

class SerialCommunication:
    def __init__(self, serialPort, serialBaud, timeout):
        self.port = serialPort
        self.baud = serialBaud
        self.timeout = timeout

        # Se muestra por consola si se ha podido realizar la conexi n con el puerto
        print('Trying to connect to: ' + str(serialPort) + ' at ' + str(serialBaud)
            + ' BAUD.')
        try:
            self.serialConnection = serial.Serial(serialPort,serialBaud,timeout=3)
            print('Connected to ' + str(serialPort) + ' at ' + str(serialBaud) + '
                BAUD.')
        except:
```

```
            print("Failed to connect with " + str(serialPort) + ' at ' + str(
                serialBaud) + ' BAUD.')
            exit()

        time.sleep(2) # Pausa de 2 segundos para poder leer el primer caso.

    def SendData(self, ts):
        self.serialConnection.write(b'1')  # Se al de control para solicitar array
            de datos
        # Guardado del tiempo actual
        self.time_data_now = (str(ts.hour)+':'+str(ts.minute)+':'+str(ts.second))

    def ReadData(self):
        # Leemos datos, no se pasa de aqui hasta el final de linea '\n' (una vez
            leido se elimina)
        serialdata = self.serialConnection.readline()[:-1]
        self.serialConnection.flushInput() # Limpiamos puerto
        # Decode() porque la comunicacion es en bytes. A adimos tiempo al array.
        self.fulldata = (self.time_data_now+','+str(serialdata.decode()))
        return self.fulldata

    def Close(self):
        print('Closing serial port...')
        self.serialConnection.close()
        print('Closed.')

###########################################################################

############## Funcion para crear y escribir archivo CSV ##############

###########################################################################

def CSVfile(fulldata, ts):
    # Nombre del fichero
    filename = ('Data_'+str(ts.year)+'_'+str(ts.month)+'_'+str(ts.day)+'.csv')

    # Si el archivo ya existe no se crea otro, se continuar  escribiendo donde se
        dej  en el ya creado
    try:
        try_open = open(filename,'r')
    except:
        create = open(filename,'w')
        # Encabezado
        create.write('Time,Temperature ( C ),Turbidity (NTU),pH,EC (ms/cm),ORP,DO
            ,288 spectrometer values'+'\n')
        create.close()

    with open(filename,'a') as add: # Se escribe en la fila siguiente a la ultima
        escrita
        add.write(fulldata)
```

```
        # Se cierra archivo con cada escritura para poder consultarlo durante la
            adquisicion
        add.close()

##########################################################################

########################### Interfaz grafica ############################

##########################################################################

class GUI:
    def __init__(self, axisX, spectrX):
        # Se crea ventana de la GUI
        self.app = QtGui.QApplication([])
        pg.setConfigOptions(background='w')
        pg.setConfigOptions(foreground='k')
        self.win = pg.GraphicsWindow(title="Graficas en tiempo real")
        self.win.resize(1000,1000)
        self.axisX = axisX
        self.spectrX = spectrX

        # Se crean las graficas
        self.p1 = self.win.addPlot(title="Temperatura", row=0, col=0)
        self.p2 = self.win.addPlot(title="Turbidez", row=0, col=1)
        self.p3 = self.win.addPlot(title="pH", row=0, col=2)
        self.p4 = self.win.addPlot(title="EC", row=1, col=0)
        self.p5 = self.win.addPlot(title="ORP", row=1, col=1)
        self.p6 = self.win.addPlot(title="DO", row=1, col=2)
        self.p7 = self.win.addPlot(title="Espectr metro", row=2, col=0, colspan=3)

        # Relleno
        self.curva1 = self.p1.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva2 = self.p2.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva3 = self.p3.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva4 = self.p4.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva5 = self.p5.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva6 = self.p6.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva7 = self.p7.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))

        # Rango eje y
        self.p1.setRange(yRange=[0, 30])
        self.p2.setRange(yRange=[0, 1000])
        self.p3.setRange(yRange=[0, 14])
```

```
        self.p4.setRange(yRange=[0, 15])
        self.p5.setRange(yRange=[0, 1000])
        self.p6.setRange(yRange=[0, 15])
        self.p7.setRange(yRange=[0, 1023])

        # Inicializacion de vectores para ploteo
        self.data1 = np.zeros(self.axisX)
        self.data2 = np.zeros(self.axisX)
        self.data3 = np.zeros(self.axisX)
        self.data4 = np.zeros(self.axisX)
        self.data5 = np.zeros(self.axisX)
        self.data6 = np.zeros(self.axisX)


    def UpdateGUI(self, temperatura, turbidez, ph, ec, orp, do, spectrY):
        # Se guardan valores acutales al final del array. Se limita el tama o del
            array.
        self.data1 = np.append(self.data1, temperatura)[-self.axisX:]
        self.data2 = np.append(self.data2, turbidez)[-self.axisX:]
        self.data3 = np.append(self.data3, ph)[-self.axisX:]
        self.data4 = np.append(self.data4, ec)[-self.axisX:]
        self.data5 = np.append(self.data5, orp)[-self.axisX:]
        self.data6 = np.append(self.data6, do)[-self.axisX:]

        #Actualizamos los datos y refrescamos la gr fica.
        self.curva1.setData(self.data1)
        self.curva2.setData(self.data2)
        self.curva3.setData(self.data3)
        self.curva4.setData(self.data4)
        self.curva5.setData(self.data5)
        self.curva6.setData(self.data6)
        self.curva7.setData(self.spectrX,spectrY)

        self.app.processEvents()



########################################################################

######################### Funcion principal ############################

########################################################################


def loop():

    ts = datetime.datetime.now() # Se guarda el tiempo en el que se hizo la medida

    s.SendData(ts) # Enviamos orden al Arduino para escribir datos en puerto serie
    fulldata = s.ReadData() # Leemos datos
```

```
        CSVfile(fulldata, ts) # Escritura csv

        # Array of floats despreciando primera medida(tiempo)
        fulldata = np.array(fulldata.split(',')[1:-1],dtype=float)
        # Graficamos
        g.UpdateGUI(fulldata[0], fulldata[1], fulldata[2], fulldata[3], fulldata[4],
            fulldata[5], fulldata[6:])

        # A la nube
        HEADER='&field1={}&field2={}&field3={}&field4={}&field5={}&field6={}'
        try:
            data=urllib.request.urlopen(NEW_URL+HEADER.format(fulldata[0],fulldata[1],
                fulldata[2],fulldata[3],fulldata[4],fulldata[5]))
        except:
            print('Failed to connect with ThingSpeak at time: ' + str(ts.hour) + ':' +
                str(ts.minute) + ':' + str(ts.second))



        # print(time.perf_counter() + 'segundos.')


if __name__ == '__main__':

        Tmuestreo = 15 # Segundos
        axisX = 50 # Numero de muestras en graficas
        # Vector valores eje X espectrometro
        # Constantes de calibraci n.
        A0=309.127381; B1=2.69111242; B2=-0.001156056159;
        B3=-0.000006599636705; B4=0.000000002507372316;
        B5=0.00000000001735837823
        channels=range(1,289)
        spectrX = []
        for pix in channels:    # Se calculan longitudes de onda para cada pixel
            wavelength = A0+B1*int(pix)+B2*pow(int(pix),2)+B3*pow(int(pix),3)+B4*pow(int
                (pix),4)+B5*pow(int(pix),5)
            spectrX.append(int(wavelength))

        g = GUI(axisX, spectrX) # Se crean graficas para ploteo

        # Configuraci n para subir a la nube
        URL='https://api.thingspeak.com/update?api_key='
        KEY= 'H831KUU19V47QCQC'
        NEW_URL = URL+KEY

        s = SerialCommunication('COM3', 115200, 3) # Inicio conexion puerto serie

        # Temporizador. Se llama a la funcion loop cada Tmuestreo segundos
        timer = QtCore.QTimer()
```

```
    timer.timeout.connect(loop)
    timer.start(Tmuestreo*1000)

    pg.QtGui.QApplication.exec_() # No se pasa hasta cerrar ventana gr fica
    s.Close() # Cerramos puerto
```

Código 14.1: Código completo en Python.

14.1.1. Comunicación con Arduino por puerto serie.

```
   breakatwhitespace
class SerialCommunication:
    def __init__(self, serialPort, serialBaud, timeout):
        self.port = serialPort
        self.baud = serialBaud
        self.timeout = timeout

        # Se muestra por consola si se ha podido realizar la conexi n con el puerto
        print('Trying to connect to: ' + str(serialPort) + ' at ' + str(serialBaud)
            + ' BAUD.')
        try:
            self.serialConnection = serial.Serial(serialPort,serialBaud,timeout=3)
            print('Connected to ' + str(serialPort) + ' at ' + str(serialBaud) + '
                BAUD.')
        except:
            print("Failed to connect with " + str(serialPort) + ' at ' + str(
                serialBaud) + ' BAUD.')

        time.sleep(2) # Pausa de 2 segundos para poder leer el primer caso.

    def SendData(self, ts):
        self.serialConnection.write(b'1')  # Se al de control para solicitar array
            de datos
        # Guardado del tiempo actual
        self.time_data_now = (str(ts.hour)+':'+str(ts.minute)+':'+str(ts.second))

    def ReadData(self):
        # Leemos datos, no se pasa de aqui hasta el final de linea '\n' (una vez
            leido se elimina)
        serialdata = self.serialConnection.readline()[:-1]
        self.serialConnection.flushInput() # Limpiamos puerto
        # Decode() porque la comunicacion es en bytes. A adimos tiempo al array.
        self.fulldata = (self.time_data_now+','+str(serialdata.decode()))
        return self.fulldata

    def Close(self):
        print('Closing serial port...')
```

```
        self.serialConnection.close()
        print('Closed.')
```

14.1.2. Función para crear y escribir archivo CSV.

```
   breakatwhitespace
def CSVfile(fulldata, ts):
    # Nombre del fichero
    filename = ('Data_'+str(ts.year)+'_'+str(ts.month)+'_'+str(ts.day)+'.csv')

    # Si el archivo ya existe no se crea otro, se continuar escribiendo donde se
        dej en el ya creado
    try:
        try_open = open(filename,'r')
    except:
        create = open(filename,'w')
        # Encabezado
        create.write('Time,Temperature ( C ),Turbidity (NTU),pH,EC (ms/cm),ORP,DO
            ,288 spectrometer values'+'\n')
        create.close()

    with open(filename,'a') as add: # Se escribe en la fila siguiente a la ultima
        escrita
        add.write(fulldata)
        # Se cierra archivo con cada escritura para poder consultarlo durante la
            adquisicion
        add.close()
```

14.1.3. Creación y actualización de la GUI.

```
   breakatwhitespace
class GUI:
    def __init__(self, axisX, spectrX):
        # Se crea ventana de la GUI
        self.app = QtGui.QApplication([])
        pg.setConfigOptions(background='w')
        pg.setConfigOptions(foreground='k')
        self.win = pg.GraphicsWindow(title="Graficas en tiempo real")
        self.win.resize(1000,1000)
        self.axisX = axisX
```

```
        self.spectrX = spectrX

        # Se crean las graficas
        self.p1 = self.win.addPlot(title="Temperatura", row=0, col=0)
        self.p2 = self.win.addPlot(title="Turbidez", row=0, col=1)
        self.p3 = self.win.addPlot(title="pH", row=0, col=2)
        self.p4 = self.win.addPlot(title="EC", row=1, col=0)
        self.p5 = self.win.addPlot(title="ORP", row=1, col=1)
        self.p6 = self.win.addPlot(title="DO", row=1, col=2)
        self.p7 = self.win.addPlot(title="Espectrometro", row=2, col=0, colspan=3)

        # Relleno
        self.curva1 = self.p1.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva2 = self.p2.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva3 = self.p3.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva4 = self.p4.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva5 = self.p5.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva6 = self.p6.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))
        self.curva7 = self.p7.plot(pen='k', width=5, fillLevel=0, brush
            =(0,0,255,100))

        # Rango eje y
        self.p1.setRange(yRange=[0, 30])
        self.p2.setRange(yRange=[0, 1000])
        self.p3.setRange(yRange=[0, 14])
        self.p4.setRange(yRange=[0, 15])
        self.p5.setRange(yRange=[0, 1000])
        self.p6.setRange(yRange=[0, 15])
        self.p7.setRange(yRange=[0, 1023])

        # Inicializacion de vectores para ploteo
        self.data1 = np.zeros(self.axisX)
        self.data2 = np.zeros(self.axisX)
        self.data3 = np.zeros(self.axisX)
        self.data4 = np.zeros(self.axisX)
        self.data5 = np.zeros(self.axisX)
        self.data6 = np.zeros(self.axisX)


    def UpdateGUI(self, temperatura, turbidez, ph, ec, orp, do, spectrY):
        # Se guardan valores acutales al final del array. Se limita el tama o del
            array.
        self.data1 = np.append(self.data1, temperatura)[-self.axisX:]
        self.data2 = np.append(self.data2, turbidez)[-self.axisX:]
```

```
        self.data3 = np.append(self.data3, ph)[-self.axisX:]
        self.data4 = np.append(self.data4, ec)[-self.axisX:]
        self.data5 = np.append(self.data5, orp)[-self.axisX:]
        self.data6 = np.append(self.data6, do)[-self.axisX:]

        #Actualizamos los datos y refrescamos la gr fica.
        self.curva1.setData(self.data1)
        self.curva2.setData(self.data2)
        self.curva3.setData(self.data3)
        self.curva4.setData(self.data4)
        self.curva5.setData(self.data5)
        self.curva6.setData(self.data6)
        self.curva7.setData(self.spectrX,spectrY)

        self.app.processEvents()
```

14.1.4. Función “main()”.

```
   breakatwhitespace
if __name__ == '__main__':

    Tmuestreo = 15 # Segundos
    axisX = 50 # Numero de muestras en graficas
    # Vector valores eje X espectrometro
    # Constantes de calibraci n.
    A0=309.127381; B1=2.69111242; B2=-0.001156056159;
    B3=-0.000006599636705; B4=0.000000002507372316;
    B5=0.0000000001735837823
    channels=range(1,289)
    spectrX = []
    for pix in channels:    # Se calculan longitudes de onda para cada pixel
        wavelength = A0+B1*int(pix)+B2*pow(int(pix),2)+B3*pow(int(pix),3)+B4*pow(int
            (pix),4)+B5*pow(int(pix),5)
        spectrX.append(int(wavelength))

    g = GUI(axisX, spectrX) # Se crean graficas para ploteo

    # Configuraci n para subir a la nube
    URL='https://api.thingspeak.com/update?api_key='
    KEY= 'H831KUU19V47QCQC'
    NEW_URL = URL+KEY

    s = SerialCommunication('COM3', 115200, 3) # Inicio conexion puerto serie

    # Temporizador. Se llama a la funcion loop cada Tmuestreo segundos
```

```
    timer = QtCore.QTimer()
    timer.timeout.connect(loop)
    timer.start(Tmuestreo*1000)

    pg.QtGui.QApplication.exec_() # No se pasa hasta cerrar ventana gr fica
    s.Close() # Cerramos puerto
```

14.1.5. Función “loop()”.

```
   breakatwhitespace
def loop():

    ts = datetime.datetime.now() # Se guarda el tiempo en el que se hizo la medida

    s.SendData(ts) # Enviamos orden al Arduino para escribir datos en puerto serie
    fulldata = s.ReadData() # Leemos datos

    CSVfile(fulldata, ts) # Escritura csv

    # Array of floats despreciando primera medida(tiempo)
    fulldata = np.array(fulldata.split(',')[1:-1],dtype=float)
    # Graficamos
    g.UpdateGUI(fulldata[0], fulldata[1], fulldata[2], fulldata[3], fulldata[4],
        fulldata[5], fulldata[6:])

    # A la nube
    HEADER='&field1={}&field2={}&field3={}&field4={}&field5={}&field6={}'
    data=urllib.request.urlopen(NEW_URL+HEADER.format(fulldata[0],fulldata[1],
        fulldata[2],fulldata[3],fulldata[4],fulldata[5]))

    print(time.perf_counter() + 'segundos.')
```

15 CONFIGURACIÓN DE THINGSPEAK. [21]

Para utilizar la plataforma *ThingSpeak* necesitaremos una cuenta de *MathWorks*, esta puede crearse de forma gratuita. Con ella podrá enviarse una petición a *ThingSpeak* una vez cada

15 segundos. En caso de que se necesitase trabajar con un tiempo inferior habría que disponer de una licencia asociada a la cuenta, sin embargo, para este proyecto una tasa de refresco de 15 segundos es suficiente.

A continuación se explica pormenorizadamente los pasos a seguir para configurar un canal al que subir los datos.

15.1. Creación del canal.

Una vez que se inicia sesión en la cuenta correspondiente nos dirigimos al apartado *"Channels"* y entramos en *"My Channels"* tal y como se muestras en la imagen 15.1.

Figura 15.1 – Acceso a los canales asociados a la cuenta de *MathWorks*.

Una vez dentro hacer click en *New Channel*. Se podrán observar distintos campos como el nombre del canal y una descripción (opcional), pero lo realmente importante son las casillas *Field*. Por cada parámetro que se quiera visualizar habrá que crear un *Field*. En la imagen 15.2 se muestra un ejemplo con 6 campos, cuando los campos esten creados hacer click en *Save Channel*.

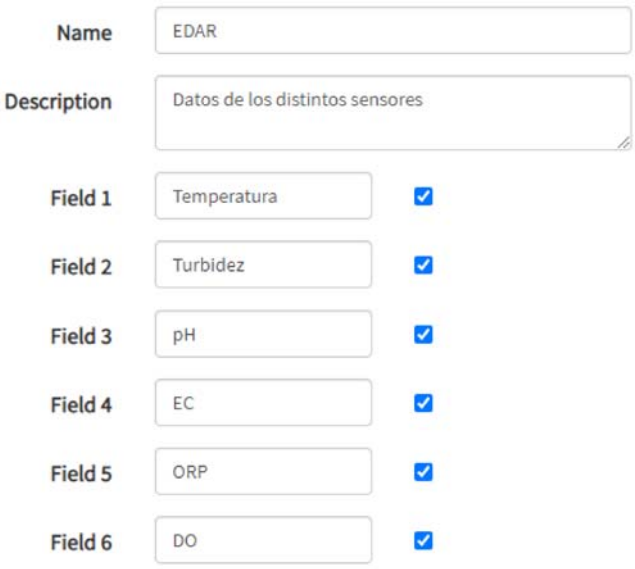


Figura 15.2 – Campos rellenados con nombres identificativos.

Una vez creado el canal se podrán hacer distintas gestiones, como hacerlo público o privado según se desee, o que sólo determinadas cuentas de *MathWorks* puedan visualizarlo. Incluso puede geolocalizarse introduciendo coordenadas en caso de utilizar distintos canales para distintos flujos de datos.

Otra opción interesante es la posibilidad de exportar los datos del canal a un archivo CSV. Se podrían descargar los datos obtenidos en un determinado día para posteriormente trabajar con ellos, como se puede ver la plataforma ofrece un gran número de posibilidades.

El canal ya estaría listo para su utilización, sólo queda implementar el software necesario para realizar la comunicación entre el canal y el equipo. En la imagen 15.3 se muestra el aspecto final de la aplicación.

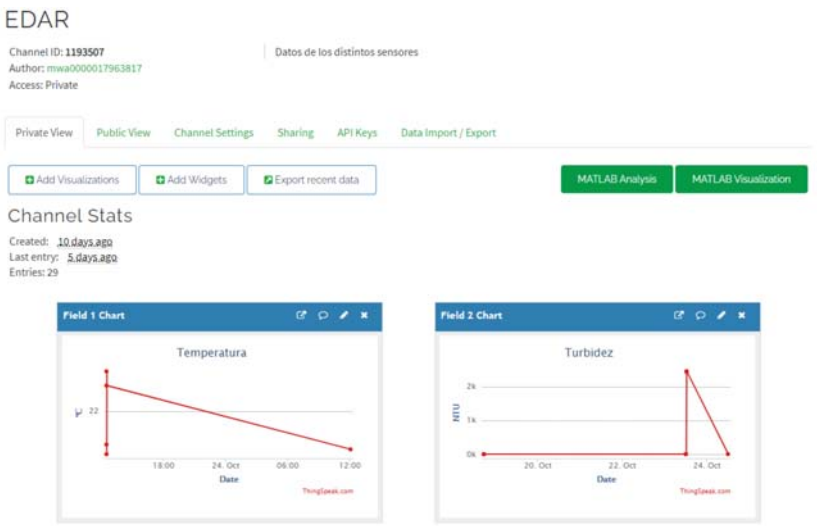


Figura 15.3 – Aspecto del canal configurado.

En caso de querer modificar algunos aspectos de las gráficas, como pueden ser el número de datos mostrados o las unidades de los ejes basta con dirigirse al icono con forma de lápiz. Al pulsarlo se abrirá la ventana mostrada en la imagen 15.4, en la que se podrán editar los campos deseados.

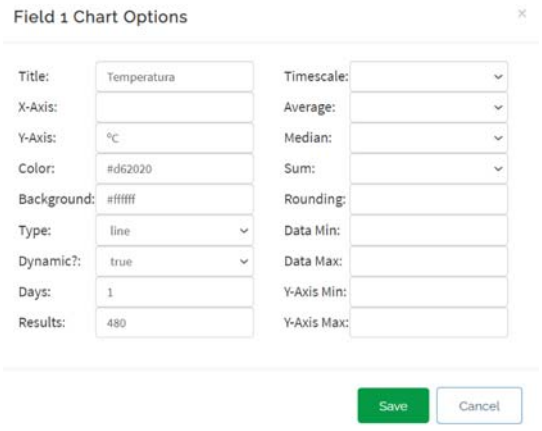


Figura 15.4 – Ventana de edición de las gráficas del canal de *ThingSpeak*.

15.2. Clave API para escritura del canal.

Para la comunicación con el canal será necesario poseer clave API o *API Key*, cada canal tiene unas claves propias, una para escritura y otra para lectura. En este caso se necesita la clave para escritura, para obtenerla basta con ir a la pestaña *API Keys* en el menú del canal. En la imagen 15.5 se muestran ambas claves, para el código del anexo 14 se ha utilizado la clave de escritura.

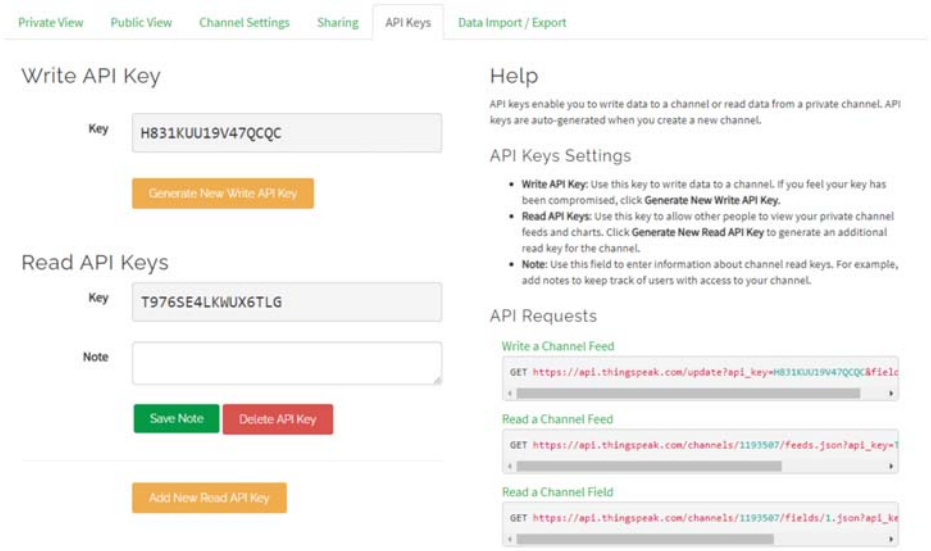
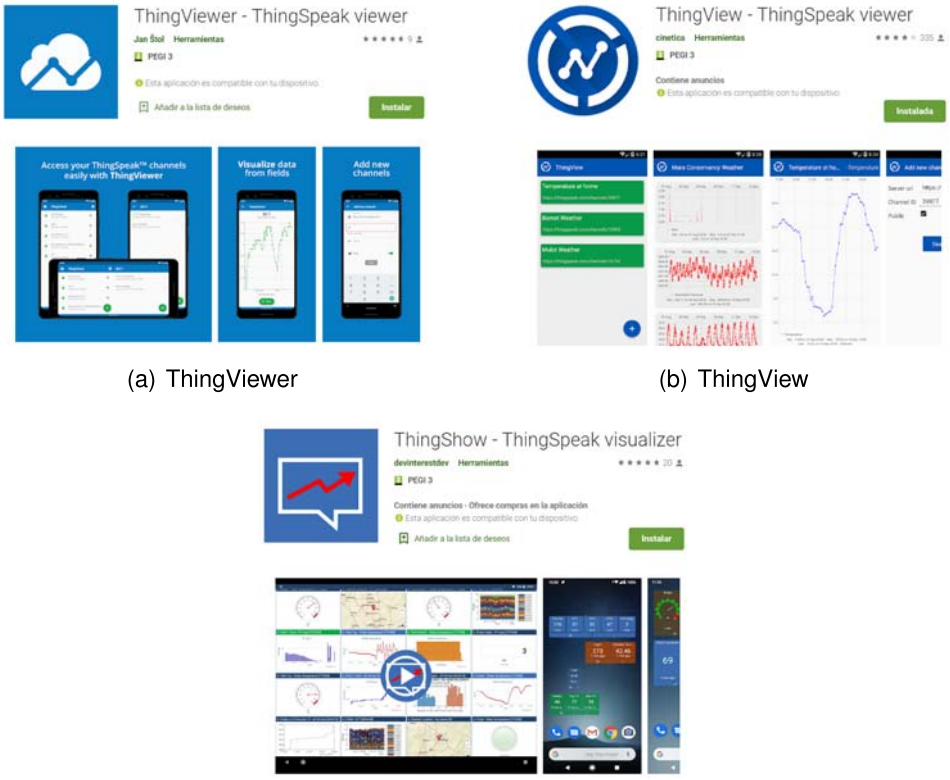


Figura 15.5 – Sección con las claves API del canal.

15.3. Aplicaciones para visualizar los datos desde un smartphone.

Si se desease consultar los datos desde un dispositivo móvil o tablet hay aplicaciones que facilitan el acceso a la plataforma tanto para dispositivos Android como para iOS. En las imágenes 15.7 y 15.6 se pueden ver algunos ejemplos de las aplicaciones mencionadas.

Figura 15.6 – Aplicación en iOS para acceder a *ThingSpeak*.



(a) ThingViewer (b) ThingView

(c) ThingShow

Figura 15.7 – Aplicaciones Android para acceder a *ThingSpeak*.

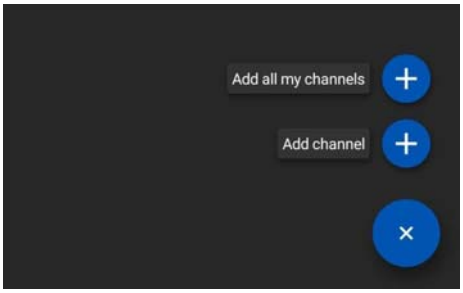
15.3.1. Ejemplo de conexión al canal con la aplicación ThingView.

A continuación se mostrarán los pasos a seguir para conectarse a un canal utilizando la aplicación ThingView. El proceso mostrado también es válido en caso de utilizar otra aplicación, ya que no hay grandes diferencias entre ellas.

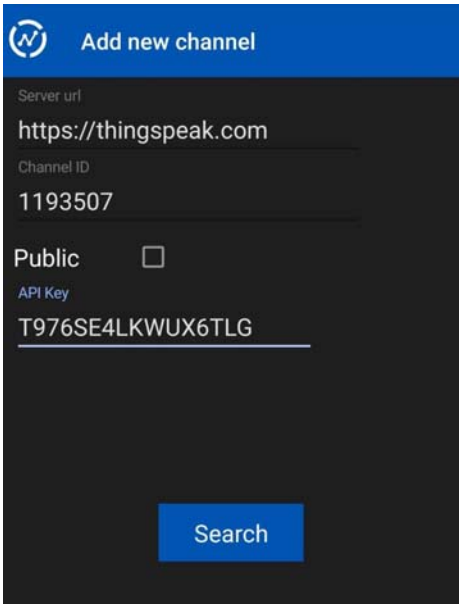
Hay que tener en cuenta que para ver un canal público bastará con introducir el *Channel ID*, mientras que si es privado, habrá que introducir también la clave API. En este caso solo interesa ver el canal, no escribir datos, por lo que se utiliza la clave de lectura.

En la imagen 15.8 se muestran de forma gráfica los pasos a seguir, estos son:

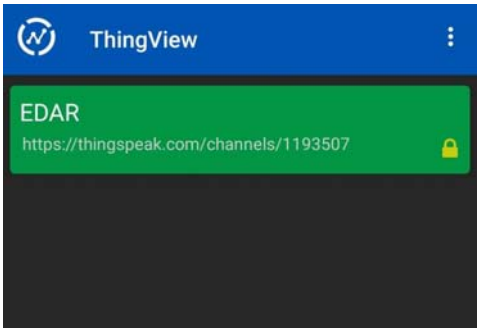
- Paso 1. Añadir un canal (15.8(a)).
- Paso 2. Introducir el ID de canal y la clave de lectura de la API (15.8(b)).
- Paso 4. El canal debe aparecer en el menú principal (15.8(c)).
- Paso 5. Ya se pueden ver los datos del canal (15.8(d)).



(a) Añadir un canal.



(b) Introducir ID del canal y clave de lectura.



(c) Canal en el menú principal.

(d) Datos del canal.

Figura 15.8 – Pasos a seguir para asociar un canal a la aplicación móvil.

TÍTULO: **CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DE DISPOSITIVOS HARDWARE DE BAJO COSTE**

PLANOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

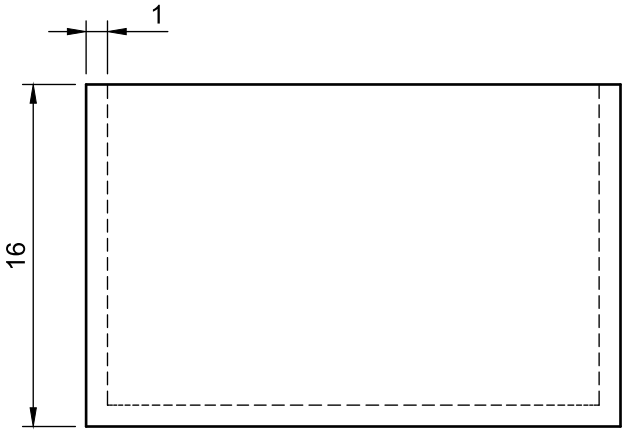
15405 - FERROL

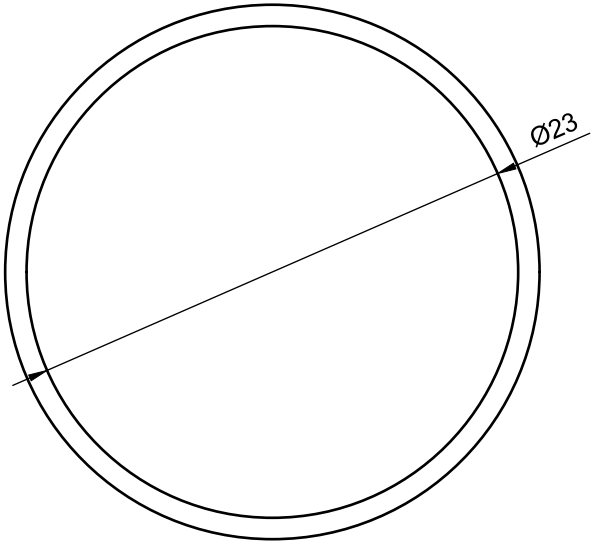
FECHA: DICIEMBRE DE 2020

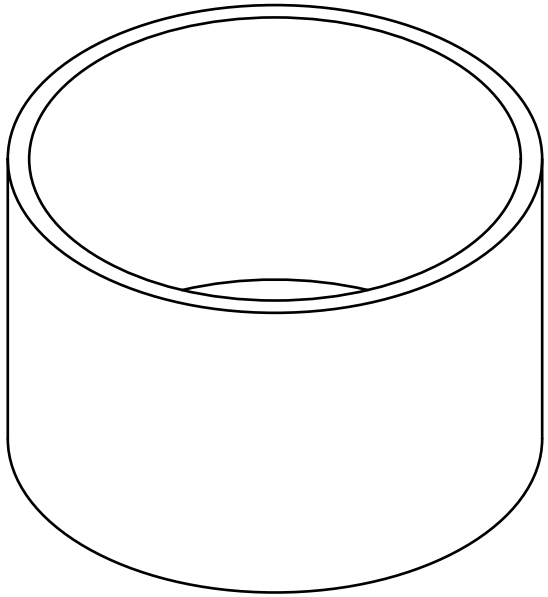
AUTOR: EL ALUMNO

Fdo.: YAGO GÓMEZ CASTRO

Índice de planos

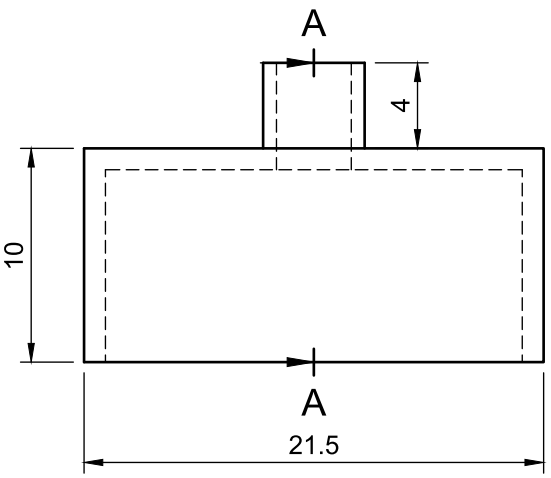


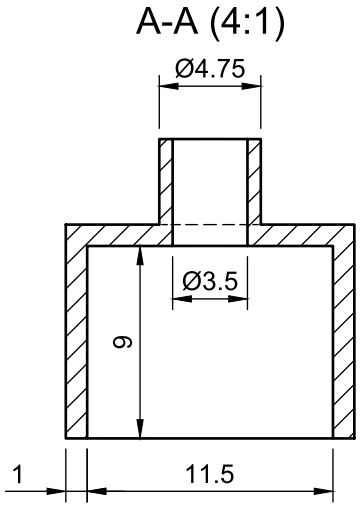


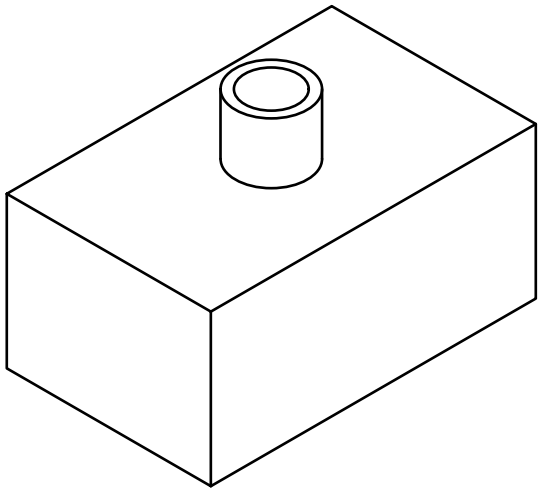


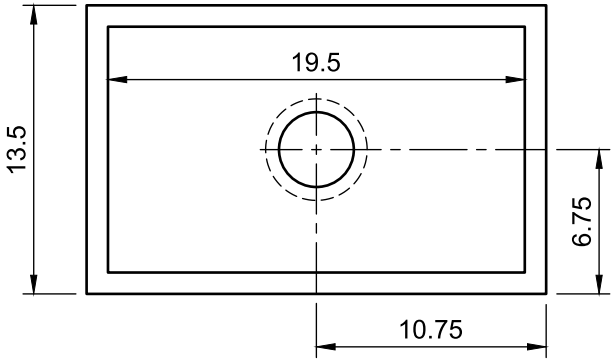
UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA		TFG Nº: 770G01A193
TÍTULO DEL TFG: CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DISPOSITIVOS HARDWARE DE BAJO COSTE		
TÍTULO DEL PLANO: Encapsulado espectrómetro. Tapa.		FECHA: DICIEMBRE-2020 ESCALA: 4 : 1
AUTOR: Yago Gómez Castro	FIRMA:	PLANO Nº: 01

A3





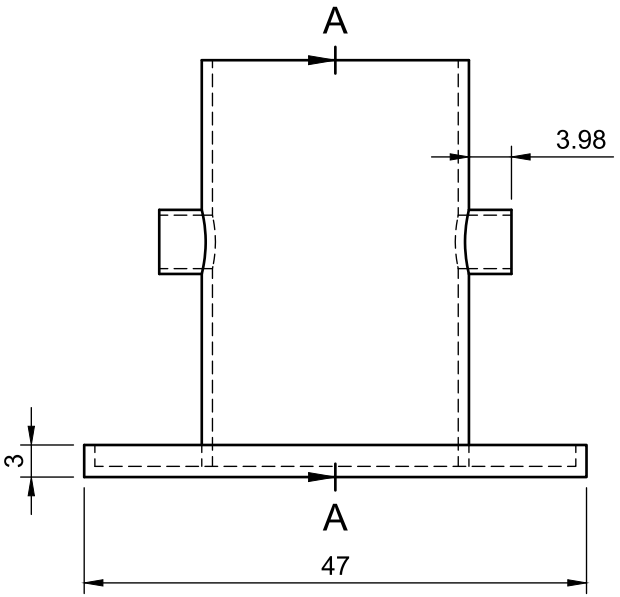


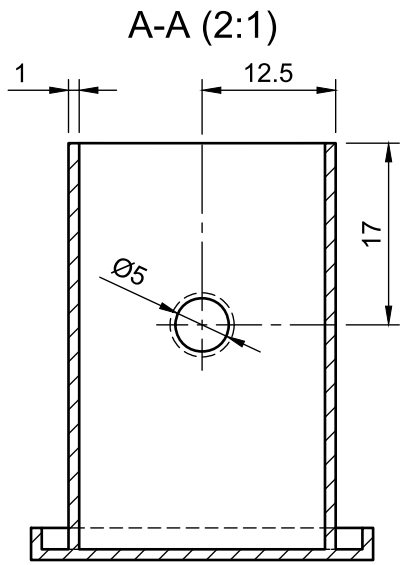


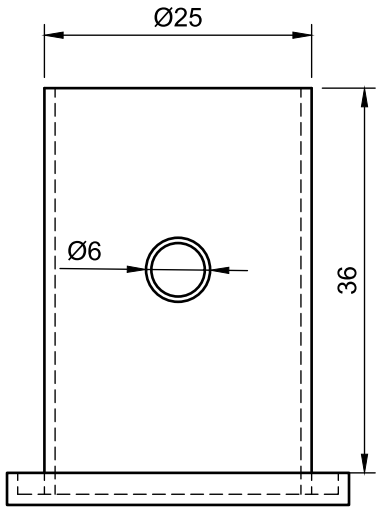
UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA		TFG Nº: 770G01A193
TÍTULO DEL TFG: CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DISPOSITIVOS HARDWARE DE BAJO COSTE		
TÍTULO DEL PLANO: Encapsulado espectrómetro. Carcasa sensor.		FECHA: DICIEMBRE-2020
AUTOR: Yago Gómez Castro	FIRMA:	ESCALA: 4 : 1 PLANO Nº: 02

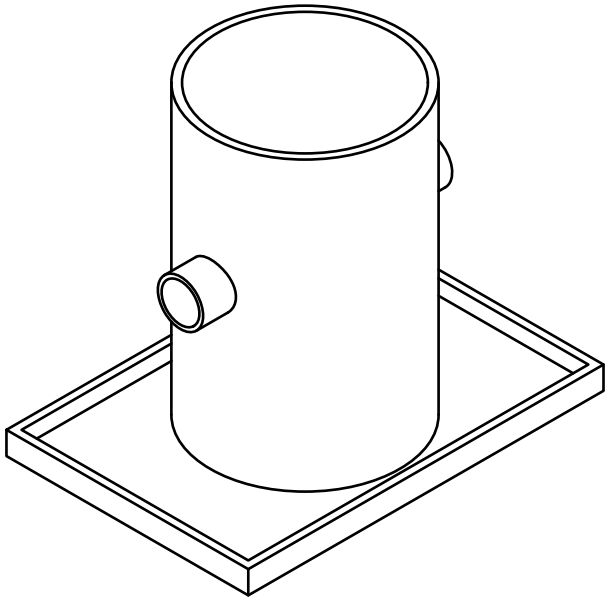
A3

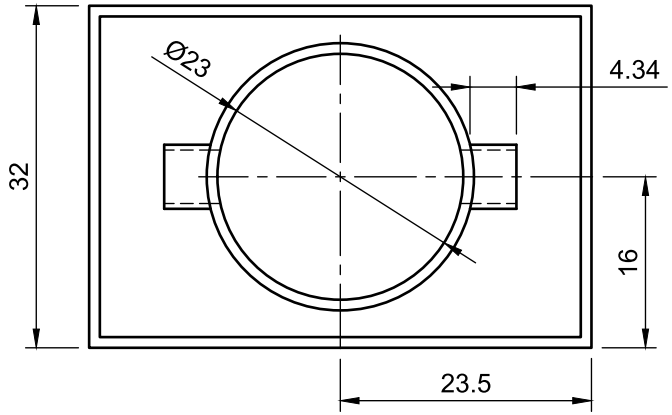
PRODUCED BY AN AUTODESK STUDENT VERSION











UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A193

TÍTULO DEL TFG:

CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DISPOSITIVOS HARDWARE DE BAJO COSTE

TÍTULO DEL PLANO:

Encapsulado espectrómetro. Cuerpo.

FECHA: DICIEMBRE-2020

ESCALA: 2 : 1

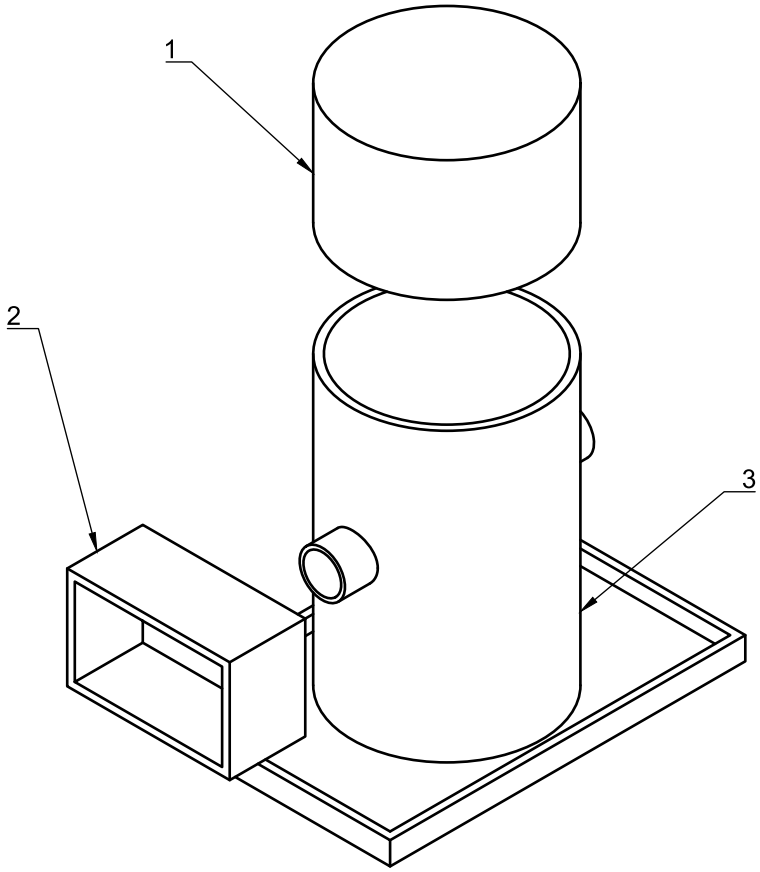
AUTOR:

Yago Gómez Castro

FIRMA:

PLANO Nº: 03

A3



Leyenda de montaje	
Número	Descripción
1	Tapa
2	Carcasa sensor
3	Cuerpo del encapsulado

ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A193

TÍTULO DEL TFG:

CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DISPOSITIVOS HARDWARE DE BAJO COSTE

TÍTULO DEL PLANO:

Vista explosionada del encapsulado.

FECHA: DICIEMBRE-2020

ESCALA: 2 : 1

AUTOR:

Yago Gómez Castro

FIRMA:

PLANO Nº: 04

A3

PRODUCED BY AN AUTODESK STUDENT VERSION

TÍTULO: **CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DE DISPOSITIVOS HARDWARE DE BAJO COSTE**

MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2020

AUTOR: EL ALUMNO

Fdo.: YAGO GÓMEZ CASTRO

Índice del documento MEDICIONES

16 ELEMENTOS FÍSICOS DEL SISTEMA DAQ

16.1. Sensores.

Imagen	Descripción	Unidades
	Sensor de temperatura DS18B20 con encapsulado TO92 y capuchón impermeable	1
	Sensor de turbidez SEN0189	1
	Sensor de pH SEN0161	1
	Sensor de conductividad eléctrica DFR0300	1
	Sensor de potencial RedOx SEN0165	1
	Sensor de oxígeno disuelto SEN0237	1
	Microespectrómetro Hamamatsu CM12880MA con *Breakout Board* incorporada	1

Tabla 16.1 – Lista de mediciones para los sensores.

16.2. Placas de acondicionamiento y tarjeta de adquisición.

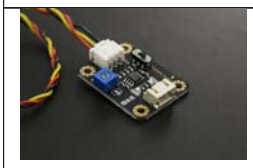
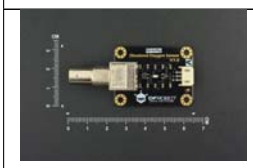
Imagen	Descripción	Unidades
	Arduino Uno Rev 3	1
	Shield de expansión IO V7.1 para Arduino Uno	1
	Placa de acondicionamiento para sensor de turbidez	1
	Placa de acondicionamiento para sensor de pH	1
	Placa de acondicionamiento para sensor de EC	1
	Placa de acondicionamiento para sensor de ORP	1
	Placa de acondicionamiento para sensor DO	1

Tabla 16.2 – Lista de mediciones para el acondicionamiento y obtención de las señales.

16.3. Encapsulado para el microespectrómetro y las muestras.

Imagen	Descripción	Unidades
	Bobina PLA Negro BQ (1.75mm)	1

	Cuerpo del encapsulado	1
	Tapa del encapsulado	1
	Capsula de contención del sensor	1
	Vial de vidrio de 22x51 mm	3
	Diodo led de 5 mm color blanco frío	1

Tabla 16.3 – Lista de mediciones para el encapsulado del microespectrómetro y las muestras.

16.4. Conexionado de los dispositivos que componen el sistema.

Imagen	Descripción	Unidades
	Cable USB tipo A a USB tipo B	1

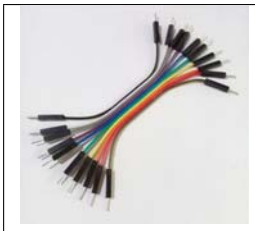
	Jumper de conexión macho - macho	-
	Jumper de conexión macho - hembra	-
	Jumper de conexión macho - hembra	-
	Cable analógico de conexión JSTx3 a tipo servo con PWR, GND y señal	5

Tabla 16.4 – Lista de mediciones con elementos necesarios para el conexionado del sistema DAQ.

16.5. Elementos necesarios para el mantenimiento de los sensores.

Imagen	Descripción	Unidades
	Capuchón de recambio para el sensor de oxígeno disuelto	1
	Solución de NaOH 0.5M para el sensor de DO	1

	Solución de HCl 0.1M para reactivación del electrodo del sensor de ORP	1

Tabla 16.5 – Lista de mediciones con elementos necesarios para el mantenimiento de los sensores.

17 LICENCIAS SOFTWARE

Software empleado	Horas de uso
Autodesk Fusion 360 2019	15 h
Autodesk AutoCAD 2020	5 h
MathWorks Thingspeak Cloud Standard License	5 h
Arduino IDE 1.8.11	40 h
Python 3.8	150 h

Tabla 17.1 – Lista de mediciones software

18 MANO DE OBRA

Proceso	Categoría profesional	Horas
Análisis de soluciones	Graduado en Ingeniería Técnica Industrial Especialidad en Electrónica Industrial y Automática	50 h
Selección de la solución adoptada	Graduado en Ingeniería Técnica Industrial Especialidad en Electrónica Industrial y Automática	10 h
Selección de los componentes electrónicos del sistema	Graduado en Ingeniería Técnica Industrial Especialidad en Electrónica Industrial y Automática	10 h
Programación de Arduino para la adquisición de datos	Graduado en Ingeniería Técnica Industrial Especialidad en Electrónica Industrial y Automática	40 h
Diseño y fabricación del encapsulado del microespectrómetro	Graduado en Ingeniería Técnica Industrial Especialidad en Electrónica Industrial y Automática	15 h
Programación en Python del software necesario para el tratamiento de los datos	Graduado en Ingeniería Técnica Industrial Especialidad en Electrónica Industrial y Automática	150 h
Redacción del proyecto	Graduado en Ingeniería Técnica Industrial Especialidad en Electrónica Industrial y Automática	200 h
	Total	470 h

Tabla 18.1 – Listado de mediciones de la mano de obra.

TÍTULO: **CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DE DISPOSITIVOS HARDWARE DE BAJO COSTE**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2020

AUTOR: EL ALUMNO

Fdo.: YAGO GÓMEZ CASTRO

Índice del documento PRESUPUESTO

19 ELEMENTOS FÍSICOS DEL SISTEMA DAQ

19.1. Sensores y circuitos de acondicionamiento.

Descripción	Cantidad	Precio uni. (€)	Precio tot. (€)
Sensor de temperatura DS18B20 conencapsulado TO92 y capuchón impermeable	1	2,25 €	2,25 €
Sensor de turbidez SEN0189 + placa de acondicionamiento	1	8,35 €	8,35 €
Sensor de pH SEN0161 + placa de acondicionamiento pH	1	24,88 €	24,88 €
Sensor de conductividad eléctrica DFR0300 + placa de acondicionamiento EC	1	58,96 €	58,96 €
Sensor de potencial RedOx SEN0165 + placa de acondicionamiento ORP	1	75,07 €	75,07 €
Sensor de oxígeno disuelto SEN0237 + placa de acondicionamiento DO	1	142,54 €	142,54 €
Microespectrómetro Hamamatsu CM12880MA con *Breakout Board* incorporada	1	336,61 €	336,61 €
		Importe total	648,66 €

Tabla 19.1 – Lista de precios para los sensores y circuitos de acondicionamiento.

19.2. Tarjetas de adquisición y cableado.

Descripción	Cantidad	Precio uni. (€)	Precio tot. (€)
Arduino Uno Rev 3	1	3,10 €	3,10 €
Shield de expansión IO V7.1 para Arduino Uno	1	7,50 €	7,50 €
Cable USB tipo A a USB tipo B	1	1,20 €	1,20 €
Jumper de conexión macho - macho 40 pcs	1	2,30 €	2,30 €

Jumper de conexión macho - hembra 40 pcs	1	1,84 €	1,84 €
Cables analógicos JSTx3 a tipo servo - 10 pcs	1	5,06 €	5,06 €
Resistencia de película de carbón 4,7 kΩ - 5 % - 1/4W	1	0,04 €	0,04 €
		Importe total	21,03 €

Tabla 19.2 – Lista de precios para las tarjetas de adquisición y el cableado.

19.3. Encapsulado para el microespectrómetro y las muestras.

Descripción	Cantidad	Peso	Precio uni. (€)	Precio tot. (€)
Bobina PLA Negro BQ (1.75mm)	1	1 Kg	16,74 €	0,17 €
Cuerpo del encapsulado	1	7 g	0,12 €	-
Tapa del encapsulado	1	2 g	0,03 €	-
Capsula de contención del sensor	1	1 g	0,02 €	-
Vial de vidrio de 22x51 mm	3	-	0,75 €	2,25 €
Diodo led de 5 mm color blanco frío	1	-	0,60 €	0,60 €
			Importe total	3,02 €

Tabla 19.3 – Lista de precios para el encapsulado del microespectrómetro y las muestras.

19.4. Mantenimiento de los sensores.

Descripción	Cantidad	Precio uni. (€)	Precio tot. (€)
Capuchón para el sensor de DO	1	12,55 €	12,55 €
Solución de 1L de NaOH 0.5M para sensor de DO	1	8,37 €	8,37 €
Solución de 1L de HCl 0.1M para reactivación de electrodo del sensor de ORP	1	10,33 €	10,33 €
		Importe total	31,25 €

Tabla 19.4 – Lista de precios para los elementos de mantenimiento de los sensores.

20 LICENCIAS SOFTWARE

Software	Horas de uso (h)	Precio uni. (€)	Precio tot. (€)
Autodesk Fusion 360 2019	15 h	61 €/mes	1,27 €
Autodesk AutoCAD 2020	5 h	279 €/mes	1,94 €
MathWorks Thingspeak Cloud Standard License	5 h	Gratis	Gratis
Arduino IDE 1.8.11	40 h	Gratis	Gratis
Python 3.8	150 h	Gratis	Gratis
		Importe total	3,21 €

Tabla 20.1 – Lista con los precios de las licencias software.

21 MANO DE OBRA

Proceso	Ct. prof	Horas	Precio uni. (€)	Precio tot. (€)
Análisis de las soluciones	Graduado en Ing. Téc. Ind. Esp. Electrónica Ind. y Automática	50 h	40 €/h	2000 €
Selección de la solución adoptada	Graduado en Ing. Téc. Ind. Esp. Electrónica Ind. y Automática	10 h	40 €/h	400 €
Selección de los componentes electrónicos del sistema	Graduado en Ing. Téc. Ind. Esp. Electrónica Ind. y Automática	10 h	40 €/h	400 €
Programación de Arduino para la adquisición de datos	Graduado en Ing. Téc. Ind. Esp. Electrónica Ind. y Automática	40 h	40 €/h	1600 €
Diseño y fabricación del encapsulado del microes-pectrómetro	Graduado en Ing. Téc. Ind. Esp. Electrónica Ind. y Automática	15 h	40 €/h	600 €
Programación en Python del software necesario para el tratamiento de los datos	Graduado en Ing. Téc. Ind. Esp. Electrónica Ind. y Automática	150 h	40 €/h	6000 €
Redacción del proyecto	Graduado en Ing. Téc. Ind. Esp. Electrónica Ind. y Automática	200 h	40 €/h	8000 €
			SUBTOTAL	19000 €
			IVA (21 %)	3990 €
			Importe total	22990 €

Tabla 21.1 – Lista de precios de la mano de obra.

22 PRESUPUESTO TOTAL

Descripción		Importe partida (€)	Importe total (€)
Elementos físicos del sistema DAQ	Sensores y circuitos de acondicionamiento	648,66 €	703,96 €
	Tarjeta de adquisición y cableado	21,03 €	
	Encapsulado para microespectrómetro y muestras	3,02 €	
	Mantenimiento de los sensores	31,25 €	
Licencias software	-	-	3,21 €
Mano de obra	Graduado en Ing. Téc. Ind. Especialidad en Electrónica Ind. y Automática	-	22990 €
		TOTAL	23697,17 €

Tabla 22.1 – Listado del precio total.

El coste total asciende a la cantidad de **VEINTITRÉS MIL SEISCIENTOS NOVENTA Y SIETE EUROS CON DIECISIETE CÉNTIMOS (23697,17 €).**

TÍTULO: **CARACTERIZACIÓN DE AGUAS RESIDUALES A TRAVÉS DE DISPOSITIVOS HARDWARE DE BAJO COSTE**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: DICIEMBRE DE 2020

AUTOR: EL ALUMNO

Fdo.: YAGO GÓMEZ CASTRO

Índice del documento PLIEGO DE CONDICIONES

23 MANTENIMIENTO Y CALIBRACIÓN DE LOS SENSORES

Algunos de los sensores utilizados en el proyecto necesitan de un mantenimiento cada cierto tiempo, la no realización de este se traduce en una disminución de la fiabilidad de las medidas que entrega el sensor. A continuación se explicará cómo realizar el mantenimiento y calibración de los sensores que lo necesiten.

23.1. Mantenimiento.

Dentro de los dispositivos que componen el proyecto hay algunos que necesitan mantenimiento cada cierto tiempo, estos son:

- Sensor de ORP.
- Sensor de DO.

23.1.1. Sensor de ORP.

Es posible que con el paso del tiempo el sensor comience a proporcionar medidas erróneas o a presentar un tiempo de respuesta más lento de lo habitual. Cuando esto ocurre es necesario reactivar el electrodo que incorpora, para ello se recomienda seguir los pasos mostrados a continuación:

- Sumergir la lamina de platino (electrodo) en una solución 0,1 M (molar) de HCl (ácido clorhídrico) durante 24 horas.
- Realizar la calibración del sensor siguiendo las explicaciones del apartado 23.2.3.

Si el uso del dispositivo ha sido de más de un año es posible que este proceso no produzca resultados debido a que se habría superado el tiempo de vida estimado del electrodo. En este caso sería necesario adquirir un sensor nuevo, ya que el electrodo va fijado al dispositivo impidiendo que se pueda cambiar por otro.

23.1.2. Sensor de DO.

El mantenimiento del sensor de oxígeno ha de realizarse para dos elementos:

- El capuchón que incorpora para contener la solución que hace posible su funcionamiento.
- La solución contenida en dicho capuchón.

La solución que se debe utilizar es NaOH (Hidróxido de sodio) 0,5 M, esta ha de introducirse en el interior del capuchón mostrado en la imagen 23.1.

Figura 23.1 – Capuchón del sensor de DO que contiene la solución de NaOH [49].

En la imagen 23.2 se muestra de forma gráfica cómo se debe introducir dicha solución.

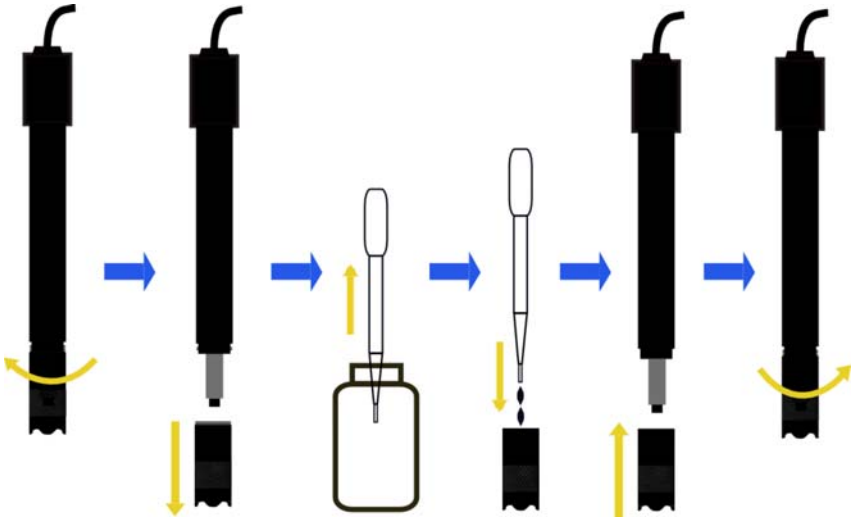


Figura 23.2 – Pasos a seguir para introducir la solución de NaOH en el capuchón [10].

Para garantizar que no se forman burbujas en el interior del capuchón y que este queda totalmente lleno se aconseja llenar entorno a 2/3 de este, de tal forma que al cerrarlo se desborde un poco de solución. Se recomienda renovar la solución del capuchón una vez al mes siguiendo este proceso.

En el caso del capuchón se recomienda ser sustituído a los 4 - 5 meses si las mediciones han sido en muestras de agua. Sin embargo, para aguas muy turbias puede llegar a necesitarse mantenimiento a los 2 meses.

23.2. Calibración.

Algunos de los sensores utilizados en este proyecto pueden ser calibrados, a continuación se explica cómo realizar dichas calibraciónes en caso de que sea necesario.

23.2.1. Sensor de pH.

Para la calibración de este sensor sólo se necesita una solución de pH conocido. El procedimiento a seguir es sencillo:

- Introducir el sensor en la solución de pH conocido y observar el pH medido por el sensor.
- Comparar el valor obtenido con el valor real de pH de la muestra y guardar la diferencia.
- En el código del anexo 13 asignar la diferencia obtenida en el parámetro **“pH_offset”**, teniendo en cuenta que este parámetro se suma en el cálculo del pH. Otra opción sería ajustar el offset mediante el potenciómetro incorporado en la PCB conectada al sensor.
- Si ahora se vuelven a realizar medidas de la misma solución el valor obtenido debería ser el correcto.

23.2.2. Sensor de EC.

Para la calibración del sensor de conductividad eléctrica se utilizan dos de las cuatro muestras que vienen incluidas en el kit del sensor, ambas de conductividad conocida para realizar una calibración a dos puntos.

En la tabla 23.1 se ven los valores de conductividad de las muestras utilizadas.

Muestra 1	Muestra 2
1413 uS/cm	12,88 mS/cm

Tabla 23.1 – Tabla con los valores de conductividad de las muestras.

Este proceso de calibración es necesario si la función de transferencia vista en el anexo 11.3 no produce buenos resultados.

En este caso habrá que obtener una función que relacione correctamente tensión y conductividad. El resultado de la calibración debería ser algo similar a lo mostrado en la imagen 23.3.

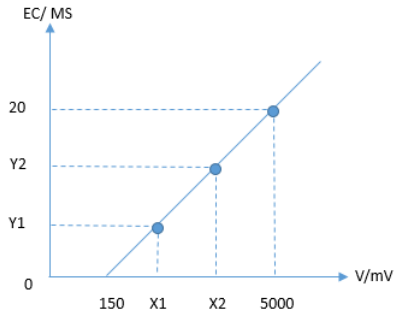


Figura 23.3 – Gráfica de calibrado del sensor de EC [8].

Los valores Y1 e Y2 representan los valores de conductividad de las muestras, mientras que X1 y X2 son los valores de tensión obtenidos para las respectivas muestras. La ecuación de la recta que pasa por ambos puntos es la nueva función de transferencia del sensor.

Hay que recordar que este proceso es realizable gracias a la placa de acondicionamiento conectada al sensor, que linealiza su salida.

23.2.3. Sensor de ORP.

Para la calibración de este sensor se puede utilizar una solución de KCl (Cloruro de potasio) de 3,5 M. Para esta solución el fabricante proporciona la tabla 23.2 de tensiones para diferentes temperaturas.

ºC	mV	ºC	mV
10	242	30	215
15	235	35	209
20	227	40	205
25	222	45	201

Tabla 23.2 – Tabla con los valores de tensión para una solución de KCl 3,5 M a diferentes temperaturas.

Si el valor de tensión entregado por el sensor no es el que debería, basta con modificar el parámetro **"ORP_offset"** del código mostrado en el anexo 13 de forma que la diferencia sea 0.

23.2.4. Sensor de DO.

Si es la primera vez que se utiliza este sensor es necesario calibrarlo. Para llevar a cabo la calibración existen dos posibilidades:

- Calibración de un punto: si la temperatura es constante se puede obtener el *voltaje de saturación* [11.4] para esa temperatura. Existen dos opciones a la hora de realizar este tipo de calibración.

  Utilizar una muestra de agua saturada en oxígeno, que ofrecerá un resultado preciso.

  Exponer el sensor al aire y realizar la medida, un proceso más sencillo.

- Calibración a dos puntos: para realizar una compensación de la temperatura en caso de que se vayan a tomar medidas en un entorno en el que esta varíe.

  Realizar el procedimiento explicado en la calibración de un punto, pero en este caso utilizar 2 muestras a distinta temperatura.

  Es recomendable no superar los 40 ºC en la muestra de temperatura elevada para evitar dañar el sensor.