Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MANTENIMIENTO DE UNA PISCINA BASADO EN ARDUINO

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Borja Sanz Gresa

Tutor: Jorge Más Estellés

Director experimental: Antonio Pinci Ferrer

2020-2021

Resumen

Actualmente, es muy común la disposición de sistemas de piscina y sistemas de riego que impliquen la realización de un conjunto de tareas repetitivas para su mantenimiento. La realización de estas tareas manual y repetidamente conlleva a una constante preocupación sobre la calidad del agua, seguridad del personal y usuarios y, costes de mantenimiento del sistema y del agua. Se propone la automatización de las tareas repetitivas mediante la implementación de un sistema electrónico basado en un conjunto de sensores y accionadores junto con la plataforma Arduino. Además de proporcionar un servidor web para el control y supervisión a distancia de las diferentes variables relacionadas con las tareas. El diseño e implementación del sistema que se propone facilitará a los usuarios con una comodidad, niveles de calidad y de seguridad, mayor eficiencia del sistema y, reducción de costes y tiempo empleado. Asimismo, se presenta como una solución competitiva de automatización de estos sistemas no considerada en el mercado actual.

Palabras clave: piscina, Arduino, control, riego.

Abstract

Nowadays, it is rather common to own a maintenance system for pools and irrigation systems that require to handle a set of repetitive maintenance tasks. Performing these tasks manually and repetitively may be a matter of concern about the quality of the water, staff's and user's safety, and costs of maintenance of the system and water. It is proposed to automate those repetitive tasks by means of assembling an electronic system based on a set of sensors and actuators alongside Arduino. Additionally, a web server is provided in order to control and monitor variables related to the tasks remotely. The proposed design and implementation of the system will result in a better wellness and safety for the user, a higher quality and efficiency for the system, and a reduction of the costs and time spent in maintenance. Furthermore, this project is a competitive solution to the automation of this type of systems that is not considered by the current market.

Keywords: pool, Arduino, control, irrigation.

Tabla de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En la costa española y en zonas de climas cálidos es cada vez más habitual que las viviendas unifamiliares y comunidades de vecinos dispongan de piscina propia. Esto conlleva realizar un conjunto de tareas repetitivas para el mantenimiento de la piscina. ¿Se imaginan lo tedioso que puede llegar a ser realizar manualmente algunas de estas tareas de manera rutinaria? ¿Y si me olvido de suplementar el desinfectante y aparecen algas difíciles de retirar? Podemos evitar tener estas preocupaciones y encontrar solución.

También, los espacios urbanizados suelen disponer de piscinas municipales o piscinas dentro de pabellones o polideportivos. Muchas de estas piscinas ya constan de sistemas automatizados profesionales muy caros. Además, es usual disponer de un sistema en un espacio ajardinado que necesite agua para su riego, muchas de las veces, procedente de la piscina.

Hoy en día, es muy importante la constante actualización a las nuevas tecnologías, puesto que mejoran y facilitan los procesos de nuestra vida diaria. Buscando y aplicando la automatización a la problemática del proyecto, se propone motorizar aquellas tareas, susceptibles de ser programadas, propias de un sistema de piscina mediante el sistema Arduino. De esta manera, se favorecerá la calidad y eficiencia del servicio y, el control y la seguridad de las tareas repetitivas que conllevan estos sistemas. También, se ahorrará tiempo del personal y costos operativos.

1.1. Motivación

Ante la presente situación de la pandemia, la cual ha cambiado nuestra perspectiva sobre la manera de afrontar muchos de los aspectos de la vida diaria, se ha visto cómo ha incrementado significativamente el uso de las nuevas tecnologías y ha facilitado el teletrabajo. Muchos se han visto obligados a adaptarse a los cambios y, se ha obtenido una nueva capacidad de adaptación y constante actualización en la sociedad. Con esto, se quiere llegar a que, ahora más que nunca, es posible conseguir un cambio que facilite nuestras necesidades y nos ahorre tiempo. Es aquí donde se debe aprovechar para actualizar y modernizar sistemas como puede ser uno tan presente en España: las piscinas y sistemas de riego.

1.1.3. Motivación personal

Siempre supe que quería realizar un TFG relacionado con la automatización, puesto que he sido fan de la domótica y el Internet de las Cosas desde que los descubrí. Dentro de este campo, se han propuesto diferentes proyectos de automatización interesantes, pero este despertó especialmente mi interés debido a que, además de haber sido propuesto por uno de los profesores con los que más he disfrutado aprendiendo, permite acercarme más a mi madre. Ella trabaja en Salud Pública y una de las muchas tareas que realiza consiste en el control de calidad higiénico-sanitaria del agua de las piscinas municipales y polideportivos. Además, mi

familia dispone de una piscina y jardín propios, por lo que la automatización del proceso de control de estos es una motivación para realizar el proyecto en casa de mis padres en un futuro.

1.2. Objetivos

La principal meta de este proyecto reside en diseñar, implementar y evaluar un sistema de control de una piscina que facilite la realización de aquellas tareas repetitivas susceptibles de ser programadas mediante la automatización. Teniendo en cuenta la frecuencia con la que un sistema de riego para jardines acompaña a las piscinas, se definen tareas para este también.

Al ser un proyecto limitado tanto de material como de presupuesto y, por lo tanto, de corto alcance, para conseguir este principal objetivo se ha tenido que definir un conjunto pequeño de tareas específicas a ser controladas mediante un servidor web simple. Sin embargo, algunos aspectos de las tareas se tendrán que simular.

- Programar el horario de filtración del agua.
- Conocer en todo momento la temperatura del agua y la temperatura ambiente.
- Controlar el nivel de llenado del agua.
- Crear un servidor web simple que permita la visualización de parámetros en tiempo real y control de variables.
- Medir la presión que proporciona la bomba de agua para satisfacer la seguridad del sistema.
- Programar el horario de riego de un terreno ajardinado.

1.3. Impacto esperado

Por lo general, la automatización de piscinas suele ser un proceso de altos costes en lo que refiere a la instalación y a la obtención de sistemas profesionales del mercado, como podría ser un dosificador automático de desinfectante (p.ej.: un clorador automático[1]). Por otra parte, si se define un proyecto de automatización mediante recursos domóticos y de Internet de las Cosas, tal y como se va a proponer en este proyecto, los costes se reducen notablemente.

Aunque en los últimos años cada vez se vea la domótica e Internet de las cosas más incrustados en la vida diaria, la causa de que estos no estén tan avanzados, normalizados y al alcance de la sociedad es el alto coste de las licencias de *software*

[1] Son componentes profesionales que buscan mantener un nivel adecuado de cloro dosificando más o menos cantidad en función del valor actual de cloro en el agua. https://www.quimipool.com/3969-control-automatico-de-cloro-redox-15-l-h.html (*Pool Basic Seko Cloro (Redox),* 543.00€. 30/08/2021)

comerciales y la necesidad de *hardware* específico (p.ej.: bombillas inteligentes). De hecho, este coste se mantiene alto debido a que no se invierte lo suficiente en este ámbito. Como alternativa para lidiar con este elevado coste, aparece la combinación de adquirir *hardware* libre (Arduino y similares), programar con *software* de código abierto y el concepto de *Do-It-Yourself* ("hazlo tú mismo") que lleva a usuarios a realizar proyectos económicos con ventajes tales como una flexibilidad adicional, y mejorar o modificar proyectos y diseños originales ya existentes. Los dos únicos inconvenientes que presenta esta solución son el requerimiento de conocimientos y experiencia previos y, un mayor tiempo a emplear en el desarrollo del sistema.

Según los diferentes tipos de usuarios y presupuestos, se tendrá en cuenta qué tipo de automatización es más interesante. Para una piscina municipal y abierta a un amplio público, probablemente sea más conveniente la obtención de sistemas profesionales que garanticen el correcto funcionamiento y la disminución de posibilidad de fallo. El coste de obtención y mantenimiento será drásticamente mayor a cambio de esa confianza que proporciona un sistema probado y estudiado a fondo. Además, estas piscinas suelen ser subvencionadas por los ayuntamientos, comunidades de vecinos o asociaciones con acceso a un mayor presupuesto. Sin embargo, en el caso de una piscina privada en el ámbito familiar puede que sea más interesante obtener recursos que se ajusten al bolsillo. Pues un buen sistema de automatización liderado por el *IoT* puede proporcionar buenísimos resultados. La mayor desventaja es no ser un sistema probado a fondo que te garantice el funcionamiento con precisión milimétrica y sin fallos. Aunque la programación puede ayudar a manejar estás disfunciones que se puedan encontrar.

Un usuario doméstico que lleva a cabo las tareas de desinfección y limpieza entre otras, manualmente, puede que no dosifique las cantidades de manera precisa, por lo que haya irregularidades en los gastos que refieren a la compra de desinfectantes o que refieren a problemas causados a raíz de este suministro manual. Por este motivo, la automatización proporciona un nivel adicional de calidad con un menor desperdicio de recursos y tiempo, y una mayor precisión en el sistema.

1.3.1. Relación con los objetivos de desarrollo sostenible (ODS)

"El 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible." [1]

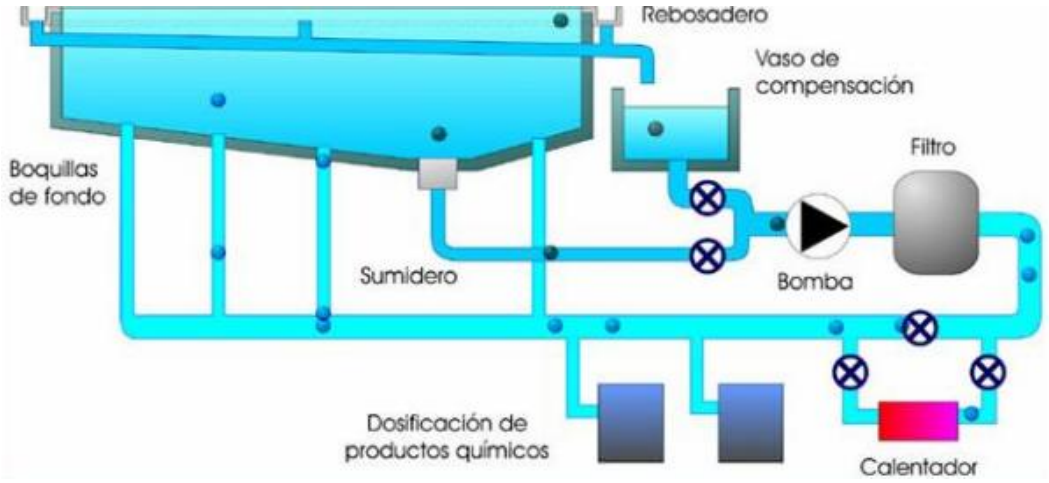
Así pues, Arduino y *hardware* similares, en general, contribuyen en varios aspectos a estos objetivos ODS. Relacionándolos con el título del TFG, podemos contemplar que existe un fuerte lazo según lo comentado en la memoria: salud y bienestar; educación de calidad; agua limpia y saneamiento; energía asequible y no contaminante; trabajo decente y crecimiento económico; industria, innovación e infraestructura; ciudades y comunidades sostenibles; producción y consumos responsables; y acción por el clima.

El diseño e implementación de un sistema de mantenimiento de una piscina basado en la tecnología Arduino facilita la automatización de tareas repetitivas y calidad de las aguas de la piscina, contribuyendo a la salud y bienestar de los usuarios. También, es un proyecto didáctico que te permite adquirir experiencia en un amplio abanico de tecnologías utilizadas por las *IoT*, la informática y la electrónica, entre otras; recurriendo a una educación de mayor calidad. Además, la inversión en estas tecnologías más eficientes dentro del campo del *IoT* es un buen comienzo para la obtención de sistemas más sostenibles y económicos que ahorran energía y aumentan la eficiencia (consumo responsable) mediante la optimización con un buen diseño. De esta manera, se da una calidad superior a la vida cotidiana (ciudades y comunidades sostenibles).

2. Estado del arte

En primer lugar, para poder comprender dónde se va a aplicar el contexto tecnológico, se introducen las partes básicas de las cual se compone una piscina:

Figura 1. Esquema de las partes básicas de una piscina



Fuente: higieneambiental.com

- Vaso: recipiente principal en el que reside el agua.
- Rebosadero: desagüe que rodea toda o parte de la piscina a la altura de la superficie con el fin de llevar el agua que desborda al vaso de compensación.
- Vaso de compensación: Se encuentra en las piscinas con rebosadero. Este recibe el agua que ha desbordado y ajusta el nivel del agua. Esta agua pasará a ser filtrada y, a continuación, devuelta a la piscina.
- *Skimmer*: suele ser una alternativa al rebosadero. Se trata de una apertura a la altura de la superficie de la piscina, por la cual el agua es aspirada para ser filtrada.
- Sumidero de fondo: desagüe que se encuentra en la parte de mayor profundidad de la piscina.
- Bomba de agua: impulsa el agua haciéndola circular desde las aperturas, por las canaletas y hasta el filtro y, posteriormente, la devuelve al vaso.
- Filtro: se encarga de recibir el agua que lo atraviesa para depositar sus impurezas y suciedad. Uno de los tipos de filtro más comunes es el filtro de arena.
- Desinfección: productos químicos que se suministran para el tratamiento del agua de forma que esta alcance una calidad óptima.
- Cuadro eléctrico: es un armario maniobra que conecta el circuito eléctrico al que se conectan todos los componentes eléctricos (filtro, bomba, etc.). Dispone

de interruptores para el control (apagado, encendido, programación de horarios…) de los componentes.

El sistema de riego se compone por un conjunto de válvulas, bocas, aspersores, filtros, bomba de agua, tuberías y un depósito (que será la propia piscina). Es cada vez más común obtener el agua proveniente de la piscina, pero cabe resaltar que se necesita una bomba de agua independiente al sistema de la piscina para satisfacer el sistema de riego. Por lo que se tendrían dos bombas de agua diferentes: una para el sistema de riego y otra para el sistema de mantenimiento de la piscina. De manera que ambas bombas se activan de forma independiente cuando sea necesario.

En segundo lugar, hay que comprender la evolución histórica del Internet de las cosas: Teniendo en cuenta que el Internet es la principal tecnología del *IoT*, se puede destacar que ARPANET (1969) fue el primer inicio de una amplia revolución tecnológica. Internet tiene el objetivo de interconectar ordenadores mediante un sistema de redes haciendo uso del protocolo Internet. Otra principal tecnología del *IoT* es los sistemas empotrados (1974), es decir, microcontroladores incrustados dentro de sistemas más grandes en los cuales la principal meta no es el procesamiento de instrucciones. Sobre 1990, apareció el concepto de la computación ubicua: la omnipresencia e invisibilidad de computadores en todo. A mediados de los 90, el concepto redes de sensores aparece: redes de objetos capaces de percibir datos del entorno que encuentran múltiples aplicaciones y convivencia con otras tecnologías como la comunicación inalámbrica y la electrónica. La primera vez que apareció el Internet de las cosas fue en 1999 a raíz de estas ciencias previas. Aunque, incluso hoy en día, todavía no se ha consensuado el alcance de las tecnologías y la definición que abarca el concepto [2]. A modo de conclusión, un sistema automatizado basado en el *IoT* se compone por una red de sensores que proporcionan información a través de Internet para ser procesada y actuar en su consecuencia.

Una vez comprendido el marco tecnológico y el escenario en cuestión al que se aplica, pasamos a analizar el mercado orientado a la domotización de una piscina, el cual se basa en un dispositivo central inteligente con conexión a internet (cuadro eléctrico) al que se conectan los demás componentes con la premisa de ser compatibles con este (bombas, filtros, dosificadores automáticos de productos desinfectantes…) y que se pueden controlar remotamente desde Internet. Al igual que en otras aplicaciones de la domótica, no se ha impulsado la comercialización de elementos centrales de control (puentes de conexión) universales, si no que cada fabricante produce el suyo y suelen ser compatibles con otros productos del mismo fabricante o un rango limitado de productos ofrecidos por otros fabricantes. Por consiguiente, nos encontramos que la compatibilidad se desenlaza como un gran problema que suele impulsar el precio del presupuesto. Si los componentes eléctricos de una piscina ya son costosos de por sí, su habilitación a la inteligencia domótica los hace prácticamente inasequibles para gran parte de la población. A causa de ello, el mercado no es muy amplio. Se

proporciona un ejemplo visual del funcionamiento de estos sistemas de mano de la empresa *Zodiac*[2].

Analizando algunos ejemplos en el mercado actual, la empresa *Zodiac* dispone de un cuadro central "*Aqualink Tri*"[3], valorado en 1400€, compatible con una gama de productos limitada de otros fabricantes o de la misma. Algunos de los automatismos[4] compatibles más populares son los robots limpia-fondos, con un precio que oscila entre 700 y 1500€; bombas de agua, desde 200€; o dosificadores de corrector de pH sobre los 350€. En comparación con componentes no domóticos, su precio suele ser bastante mayor.

Otro cuadro central que ofrece la compañía *AstraPool*, valorado en 449€[5], es una opción mucho más barata, aunque todavía costosa. Esta empresa tiene en venta un dispositivo llamado GPIO [6] (alrededor de 230€) que permite a hasta cuatro componentes no compatibles o no inteligentes conectarse a la caja central para ser automatizados mediante su tecnología basada en relés y protocolos de comunicación. Esta opción puede llegar a ser asequible si ya se tiene un sistema antiguo que no se quiere retirar. Sin embargo, puede que no sea de utilidad debido a que, si estos componentes no inteligentes no tienen sensores para obtener información del estado del entorno físico y realimentarse en función de ello, no se podrá automatizar la tarea deseada de manera adecuada.

Con el análisis anterior se quiere dar una idea de los elevados precios de cada componente. Merece la pena subrayar el coste del cuadro central, puesto que se quiere proponer el uso de un microcontrolador Arduino o similar (entre 5 y 40€) para sustituir dicho elemento central de control. La diferencia de coste es abismal: de 5-40€ a 450€ como mínimo. Además, como sustitución del elemento GPIO o para controlar componentes básicos no inteligentes de la piscina, se estudia hacer uso de un circuito eléctrico que interactúe con preaccionadores como relés. También, el uso de sensores independientes para la retroalimentación de los actuadores después del procesado de la información. Más adelante se incidirá en el estudio del coste de esta solución, el cual reduce drásticamente el gasto total.

Por último, aunque sí que existe algún proyecto similar[7], cabe resaltar que no existe ninguna solución comercializada como la que se quiere proponer.

---

[2] **Control y funcionamiento de una piscina domotizada:** Control a distancia de equipamientos de piscina (03/09/2021)
[3] **Aqualink Tri:** zodiac-poolcare.es (03/09/2021)
[4] **Dispositivos compatibles zodiac:** quimipool.com (03/09/2021)
[5] **Cuadro central de *AstraPool*:** Poolaria.com (03/09/2021)
[6] **GPIO:** Naturclara.com (03/09/2021)
[7] **Proyecto de automatización de una piscina basado en Arduino:** Automatización del sistema de control eléctrico, utilizando la tecnología arduino, aplicado a la bomba de la piscina de la Universidad Estatal del Sur de Manabí

2.1. Crítica al estado del arte

En la Escuela Técnica Superior de Informática se han presentado multitud de trabajos basados en Arduino para la automatización de tareas cotidianas. La mayoría de ellos incidían en el campo de la casa inteligente como, por ejemplo, el control de un sistema de iluminación o calefacción. Sin embargo, ninguno ha tratado sobre la automatización de las tareas que una piscina o un sistema de riego conllevan.

En algunos de estos trabajos presentados anteriormente, se muestra tan solo cómo se realiza la parte del sistema electrónico y luego proponen mejoras para futuros trabajos en las que solo ampliarían el circuito electrónico incluyendo mejoras o modificaciones con otros componentes. No obstante, una ampliación interesante a estos proyectos, que ya se ha implementado en TFGs más amplios, consiste en poder monitorear variables y controlar accionadores remotamente, ya pueda ser con una aplicación móvil, un servidor web, un mando a distancia, …

Asimismo, en este proyecto también se pretende abordar el control y monitoreo en tiempo real de variables que el sistema electrónico maneja, de manera remota mediante un servidor web con base de datos relacional.

2.2. Propuesta

Una vez estudiado qué nos ofrece el mercado actual y habiendo comparado fortalezas y debilidades de otros trabajos de la ETSINF que abarcan temas similares, podemos definir qué tecnologías se usarán para la realización del diseño e implementación de un sistema de mantenimiento de piscina. Se dividirá el proyecto en dos partes principales: un sistema electrónico, basado en un microcontrolador como Arduino simulando el cuadro eléctrico del sistema que conectará todo el sistema físico; y, el control y monitoreo de variables del sistema electrónico remotamente.

La primera parte del TFG, la cual comprende el sistema electrónico, constará de los componentes principales que buscan alcanzar los objetivos marcados: sensores de temperatura, de presión diferencial y de nivel de agua, un microcontrolador y relés para accionar los actuadores. Se simularán los actuadores con el encendido y apagado de unos LEDs.

Para la segunda parte, se propone crear un servidor web y base de datos que residan localmente en el ordenador personal, con acceso no restringido desde otros dispositivos desde la misma red local.

3. Análisis del problema

En este nuevo capítulo, nos dedicaremos a analizar cómo afrontaremos el camino elegido para llevar a cabo el proyecto teniendo en cuenta la infinidad de alternativas disponibles y justificaremos cómo se ha tomado la decisión.

3.1. Identificación y análisis de soluciones posibles

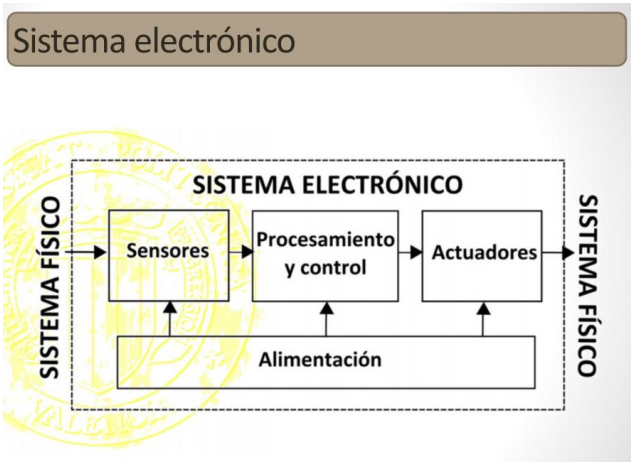
Actualmente, existe infinidad de tecnologías que se usan para este tipo de proyectos y, por consiguiente, infinitos caminos por los que se podría afrontar la problemática del TFG. Principalmente, vamos a centrarnos en comparar algunas de las alternativas más populares para el camino elegido. Cabe resaltar que solo se contemplarán soluciones de automatización mediante sistemas formados por componentes no profesionales, puesto que no existe ninguna aplicación ni sistema en el mercado que se dedique a la automatización de sistemas de piscina o riego que no sea mediante componentes específicos diseñados especialmente para piscinas.

El camino elegido para afrontar el diseño del proyecto se divide en dos partes principales: un sistema electrónico y su control a distancia.

3.1.1. El sistema electrónico:

Un sistema electrónico se compone por un conjunto de sensores, que perciben información del entorno físico en el que se encuentran; un microcontrolador, que procesa las señales que recibe por los pines de entrada y ejecuta instrucciones en base a ellas; actuadores, que transforman la señal recibida por el microcontrolador a través de los pines de salida en energía que realiza alguna función sobre un sistema físico. Por último, añadir la necesidad de alimentar cada componente con el fin de que estos puedan llevar a cabo sus funciones. Se detalla el esquema de interacción en la figura 2.

Figura 2. Esquema de un sistema electrónico



Fuente: (DSIIC, ETSID, UPV)

- **Sensores:**

  - Medición de la temperatura:

    En la tabla 1, se detallan los sensores de temperatura más populares del mercado y se comparan sus prestaciones. Podemos apreciar que el sensor DS18B20 presenta características mejores (p.ej.: una mayor resolución para una mayor precisión de medidas), pero tan solo nos interesa el aspecto de la resistencia al agua que determinará si el sensor es útil para el proyecto, puesto que ambos abarcan rangos de temperatura apropiados para el sistema en cuestión. Vemos que el DHT11 no cuenta con resistencia al agua, por lo que se tendría la necesidad de añadir una carcasa que proteja el sensor del agua.

Tabla 1. Comparativa de modelos de sensor de temperatura

SENSORES DE TEMPERATURA	DS18B20	DHT11
Pin	digital	digital
Resolución	9-12 bits	8 bits
Voltaje de alimentación	3 - 5.5 V	3 - 5.5 V
Rango de temperatura medible	[-55, 125]°C	[0, 50]°C
Tasa de error	±0.5°C	±2°C
Resistencia al agua	sí	no
Precio	7.48€[8]	5.55€[9]

Fuente: elaboración propia

[8] RS-components (31/08/2021)
[9] RS-components (31/08/2021)

Figura 3. Sensor de temperatura: DS18B20

Fuente: cloudinary.com

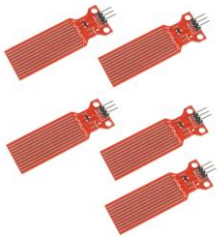
Figura 4. Sensor de temperatura: DHT11

Fuente: cloudinary.com

Por tanto, sería una mejor opción decantarse por el DS18B2o a raíz de ser sumergible en líquidos. Además, vemos que su diseño facilita la instalación al disponer de un cable que permite alcanzar longitudes más largas.

- Medición del nivel de llenado de agua:

Figura 5. Módulo sensor detector de nivel de agua (5 unidades)



Fuente: amazon.com

Con un sensor de detector de nivel de agua podemos determinar cuándo el agua hace contacto con este. Se recibe un '1' en contacto con el agua o un '0' sin contacto. Nos vemos limitados a colocar el sensor cerca de la superficie de la piscina y solo podemos conocer cuándo el agua llega a cierto a nivel. Pero no sabremos cuál es el nivel del agua en todo momento.

Como alternativa, para conocer el nivel del agua en todo momento, se considera un sensor de presión diferencial que nos permitiría conocer el nivel del agua a cualquier profundidad:

Figura 6. Sensor de presión diferencial: MPX5500DP

Fuente: cloudinary.com

La presión en la superficie del agua (al nivel del mar: cero metros) es de 1 atmósfera o 1 bar[10] y, cada vez que descendemos 10 metros de profundidad dentro del agua, la presión aumenta en 1 atmósfera o bar aproximadamente. Así pues, la presión no depende de la forma del recipiente, si no de la profundidad. La relación es de f(x)=0.1x+1, siendo 'x' la profundidad en metros y, 'f(x)', la presión en bares o atmósferas.
Teniendo en cuenta que una piscina tendrá en torno a 2 metros de profundidad, se tendrá que considerar la obtención de un sensor que sea capaz de medir presiones en un rango de 0 a 1.2 bares (120 kilopascales), como mínimo. En este caso, debido a la facilidad de obtención de estos modelos de sensores, se ha considerado la familia MPX de sensores de presión diferencial, teniendo en cuenta su rango de medición. Estos sensores se comprenden por dos tubos, uno que mide la presión en el vacío y, otro, que mide presiones más fuertes (dispone de un gel de fluorosilicona protector). De manera que se calcula la diferencia de presión en función de variables como el voltaje y la altura. Gracias a disponer de dos tubos, se dejará un tubo fuera del agua y otro dentro, haciendo posible la medición de la diferencia de presión.

- Medición de la presión de filtros:

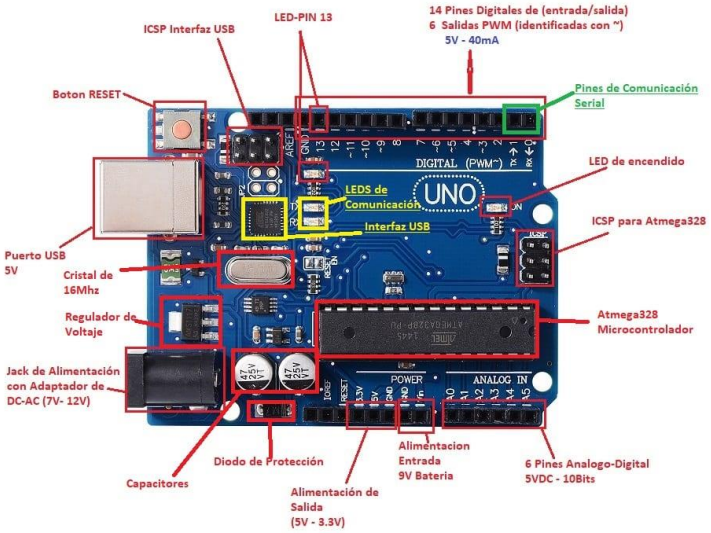
  Se puede utilizar también el sensor de presión diferencial MPX5500DP o sensores de flujo de líquido.

- **Microcontroladores**

Los microcontroladores son circuitos integrados programables que ejecutan instrucciones pregrabadas en su memoria e interactúan con pines de entrada y salida a los que se pueden conectar diferentes tipos de componentes electrónicos. En la siguiente imagen podemos apreciar las diferentes partes de una placa con microcontrolador.

[10] **Presión:** 1 bar = 0.9869 atmósferas = 100 kilopascales

Figura 7. Microcontrolador Arduino Uno



Fuente: controlautomaticoeducacion.com

Es necesario incidir en la compatibilidad de las placas originales Arduino con las demás placas que se comentarán a continuación. De manera que se podría ampliar el número de pines o combinar funcionalidades de Arduino con la comunicación e interacción con otra familia de microcontroladores. Así pues, sería necesario determinar una configuración maestro-esclavo y tenerlo en cuenta para su programación. Se detallan algunas familias de microcontroladores:

- Familia Arduino [3]:

  Es un proyecto que surgió en 2005 y ha ido evolucionando hasta ser el microcontrolador más popular con infinidad de documentación y proyectos en la web. Los modelos más populares son:

  - Arduino Uno Rev3 (por su relación número de pines y prestaciones con el precio, 20€). Existe la variante Uno WiFi Rev2 con módulo WiFi empotrado (38.90€).
  - Arduino Mega 2560 (por un mayor número de pines, 35€)
  - Placas enfocadas hacia el *IoT*, que son compatibles con Arduino IoT Cloud[11] y que se pueden conectar por WiFi a Internet, han sido lanzadas. Aunque si no se va a hacer uso del servicio de Arduino IoT Cloud no son placas que supongan ventajas significativas en comparación con las anteriores.

  Como solución al mantenimiento de la piscina, se necesitará una gran cantidad de pines, por lo que Arduino Mega es la mejor opción para

[11] **Arduino IoT Cloud:** Plataforma de Arduino que permite controlar dispositivos *IoT* mediante la vinculación de nuestros microcontroladores con el fin de desarrollar y manejar aplicaciones *IoT*.

considerar. Aunque existen protocolos de comunicación que permiten conectar múltiples sensores a un mismo pin. Un gran punto a favor para Arduino es el *software* desarrollado especialmente para microcontroladores (admite gran cantidad de placas externas a la familia). Debido al elevado precio de la familia Arduino y su carencia en la mayoría de los modelos de un módulo *WiFi* ya empotrado, se podrían considerar otras alternativas según necesidades del usuario antes de decantarse por dicha familia.

- Raspberry Pi:

Cabe destacar de que no se trata de un microcontrolador, sino de un ordenador de bajo coste con su sistema operativo y mayor capacidad de cómputo, por lo que su abanico de aplicaciones es mucho más amplio que el de Arduino. Sigue siendo compatible con Arduino al ser un ordenador. Sin embargo, no se trata de *hardware* libre y, por lo tanto, el precio de estas placas aumenta. En la tabla 2, se comparan las principales características de los modelos Arduino AVR (todos los modelos Arduino presentan características similares) y Rapberry Pi:

Tabla 2. Tabla comparativa: Arduino y Raspberry Pi

Raspberry Pi	Arduino
Es un mini PC que puede ejecutar múltiples programas al mismo tiempo	Es un micro controlador, parte de un ordenador, que ejecuta un único programa una y otra vez.
Es complicado hacerlo funcionar con batería.	Está pensado para funcionar con batería.
Requiere tareas complejas como instalar librerías y software para interactuar con sensores y otros componentes.	Sus componentes y sensores funcionan de manera integrada.
Es caro en relación a Arduino.	Es barato.
Se conecta fácilmente a Internet con su puerto RJ-45 o con WiFi por USB.	Requiere hardware externo para conectarse a Internet y hay que programarlo utilizando código para que funcione. No está pensado para conectar a Internet.
No tiene almacenamiento, pero puede usar su ranura micro SD para ello.	Puede venir con almacenamiento integrado.
Tiene 4 puertos USB para conectar distintos dispositivos.	Solo tiene un puerto USB Type-B hembra para conectarlo a un PC.
Utiliza procesadores ARM.	Utiliza un procesador de familia AVR.
Debemos apagarlo correctamente para que no haya riesgo de corrupción de archivos.	Es un dispositivo plug and play.
El lenguaje de programación recomendado es Python, pero puede usar C, C++ y Ruby también.	Solo utiliza Arduino y C/C++.

Fuente: hardzone.es

En un proyecto como el de este TFG, lo que queremos es que, nada más encender el sistema electrónico, todo se ejecute al instante (ya sea encender un LED, por ejemplo) y no tener que esperar. Con el uso de una Raspberry Pi, tendríamos que esperar a que el SO iniciara. Por otra parte,

sí que es verdad que se necesita ampliar con módulos externos las funcionalidades de Arduino, como puede ser la conexión a Internet y esto es un punto a favor para Raspberry. No obstante, cada vez está apareciendo más variedad de módulos asequibles e incluso placas Arduino que ya implementan WiFi.

Por lo tanto, teniendo en cuenta el calibre del proyecto y que no se requiere gran potencia de cómputo, Arduino es más que suficiente e incluso mejor opción al tener tal multitud de módulos de ampliación compatibles.

- Familia ESP8266 [4]:

Esta familia de microprocesadores, con la principal característica de conexión WiFi, presentan una capacidad de cómputo bastante mayor que los Arduino y una diferencia en precio sorprendentemente significativa (en torno a 5€).

Existen actualmente 16 variantes del sistema en chip *(SoC)* ESP-XX que se integra en las placas ESP8266 y ejecuta las operaciones. Todas muestran características comunes como un microprocesador de 32 bits (mayor que el de la mayoría de Arduinos) y conectividad WiFi. Entre ellos difieren en el número de pines de entrada y salida, y su implementación en una placa microcontroladora. Algunos de los modelos más populares son: Wemos D1 y Wemos D1 Mini, o NodeMCU.

La familia opera a un voltaje de 3.3 voltios, mientras que los Arduinos más populares operan a 5 voltios. Algunos modelos de la ESP no son capaces de suministrar 5 voltios por lo que puede limitarnos y requerir de una fuente de alimentación externa con algunos sensores que necesiten una alimentación superior. Aunque, por lo general, cada vez hay más modelos que solventan este aspecto. Sin embargo, en algunos modelos puede que nos sintamos limitados por el número de pines analógicos o digitales y es un aspecto para tener muy en cuenta.

En conclusión, esta familia de microcontroladores presenta grandes ventajas como conexión *WiFi*, mayor potencia de cómputo y precio drásticamente inferior. Además, la programación de los ESP se realiza mediante Arduino IDE, por lo que ya son una opción superior para considerar frente a Arduino.

- Familia ESP32 [5]:

Se trata del sucesor de ESP8266 que cuenta con una capacidad de cómputo muy superior (consta de doble núcleo) y le dobla en cualquier otra característica. No presenta casi ninguna funcionalidad adicional a su predecesor. Es un microcontrolador que encontramos por tan solo

alrededor de 10€. De todas maneras, para la automatización de las tareas de la piscina sobraría con las prestaciones de la familia predecesora, además de que la programación de ambos núcleos no resulta de gran interés en nuestro proyecto.

Por todo esto, ESP32 es una familia incluso más interesante que la original Arduino, pero para nuestro proyecto no resulta de gran interés en comparación a ESP8266.

- Actuadores

Debido al corto alcance y presupuesto del trabajo, se va a considerar la simulación de los accionadores como podrían ser llaves de paso compatibles con el sistema electrónico que accionarían actuadores como la bomba de agua.

Relés serán utilizados como preaccionadores y LEDs como los componentes. De tal manera que el relé encienda o apague el LED simulando cuando una llave de paso abriría el sistema de filtrado del agua o como cuando se activa el riego.

Para la elección de estos componentes se ha de tener en cuenta que sus características de operación sean compatibles con el circuito electrónico que se quiere montar.

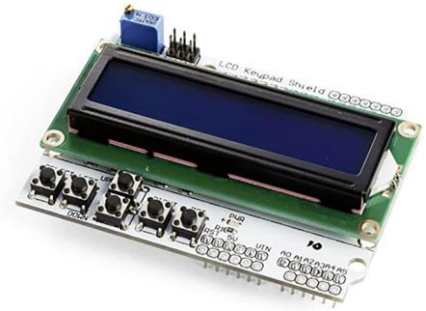
3.1.2. Plataforma de monitoreo y control de variables

En esta parte, se busca monitorear y controlar los diferentes rangos, límites y valores relacionados con las variables del entorno físico, además de la visualización en tiempo real de estos. Para ello, se proponen diferentes alternativas, siendo ninguna de ellas más válidas que otras para este proyecto, pero sí más o menos apropiadas. La elección tomada ha tenido en cuenta una mayor complejidad y relación a los aspectos abordados en las asignaturas a lo largo del grado universitario. Se proponen a continuación algunas soluciones populares:

- *Display* y teclado electrónico [6]:

Como sabemos, los microcontroladores disponen de pines de entrada y salida que permiten la conexión con módulos externos. Una de las opciones más interesantes es la interacción con una pantalla de cristal líquido y teclado. La programación del módulo no es complicada ya que existen librerías que proporcionan funciones para su fácil uso.

Figura 8. LCD con teclado



Fuente: media-amazon.com

La visualización de múltiples variables y su control se limita a la adaptación de estas a la pantalla. Admite cadenas de caracteres no muy largas que deslizan por pantalla (*scroll*) y se adaptan a una matriz de 2x16. Por lo tanto, el manejo y la programación se complica conforme aumenta el número de variables a controlar. El teclado permite interactuar con la pantalla. Sin embargo, el módulo tiene que estar cerca del circuito, por lo que se suprimiría la idea de manejo remoto del sistema. Este módulo se puede encontrar alrededor de 16€. En definitiva, no es la mejor opción a causa de la cantidad de variables a manejar en el proyecto.

- Aplicación móvil:

Esta opción puede que sea la más acertada y cómoda para los usuarios, puesto que todas las funcionalidades se recogen en una aplicación. Para desarrollar una aplicación móvil hay una gran variedad de *frameworks*, tecnologías y documentación en línea.

- Plataforma de computación en la nube dedicada al Internet de las Cosas:

La computación en la nube busca ofrecer recursos y servicios de forma escalable a través de Internet, persiguiendo un uso eficiente y económico (pago por uso). Nos provee de cómputo y almacenamiento. En nuestro proyecto tan solo nos centramos en el almacenamiento de datos, puesto que la ejecución de instrucciones las realiza el microcontrolador.

Algunas de las opciones más recomendables son Amazon Web Services (AWS), Microsoft Azure IoT o, incluso, Arduino IoT Cloud (si se dispone de una placa de Arduino original). Estas plataformas nos proporcionan servicios que se ejecutan en la nube y permiten la fácil comunicación y almacenamiento de datos entre nuestros dispositivos *IoT* (microprocesador y sensores) con el *back-end*[12] . La programación que permite esta comunicación es fácil e intuitiva con las librerías que se nos

[12] ***Back-end*:** se trata de la capa de acceso a datos.

facilita. Además, nos proporcionan otros servicios a través del navegador para poder controlar, monitorear y visualizar nuestros dispositivos o variables, mediante gráficas e interfaces ya proporcionadas.

Como gran ventaja tenemos la fácil programación para controlar nuestros proyectos de Internet de las Cosas. Por el contrario, dichas plataformas nos limitan el número de interacciones o dispositivos para cuentas gratuitas. Como se ha dicho anteriormente, los servicios en la nube buscan la escalabilidad y el pago por uso. A modo de conclusión, en nuestro proyecto esta sería una opción simple de realizar y no tendríamos problema con una cuenta gratuita para llevar a cabo pruebas. Pero si se quiere realizar la instalación definitiva del sistema de mantenimiento de la piscina habría que realizar un estudio a fondo de características y planes económicos.

- Servidor web y base de datos locales:

Se trata de la creación de un servidor web y base de datos (*full-stack*[13]) de manera local en tu ordenador. Existen herramientas gratuitas que proveen de una estructura de directorios de trabajo en la que asociar tus ficheros para la web que el navegador o el intérprete correspondiente recibirán y actuarán en consecuencia. Una web sí que nos da la flexibilidad de hacerla tan grande como se desee, por tanto, un gran conjunto de tareas a monitorizar no supone ningún problema al contrario que con la pantalla LCD. También permiten la configuración de restricciones, por lo que se podría configurar las limitaciones de acceso para poder acceder al servidor desde otros dispositivos o, incluso, desde fuera de la red local.

La funcionalidad puede llegar a ser la misma o muy similar a plataformas de computación en la nube anteriormente comentadas, aunque no se nos proporciona ninguna facilidad (ni *front-end*[14] o *back-end*, ni comunicación con la base de datos o dispositivos físicos) teniendo que crear todo desde cero. Para llevar a cabo esta solución de mayor complejidad, se requiere de mucho más tiempo, aprendizaje y conocimiento sobre tecnologías de web y protocolos.

Existen servidores web locales de código abierto que nos permiten realizar esto de manera totalmente gratuita como el servidor Apache. También, existen sistemas de gestión de bases de datos relacionales como MariaDB de código abierto. Así pues, la plataforma gratuita XAMPP es una distribución multiplataforma proporcionada para desarrolladores que nos proporciona, entre otros, la instalación y servicios de servidor Apache y base de datos mySQL que necesitamos.

[13] ***Full-stack*:** se trata de ambas capas de desarrollo (*back-end* y *front-end*).
[14] ***Front-end*:** se trata de la capa de presentación.

Figura 9. Logo de XAMPP

Fuente: desarrolloweb.com

Finalmente, XAMPP nos daría toda la flexibilidad que necesitamos para crear esta segunda parte del proyecto a nuestro gusto y de forma totalmente gratuita con capacidad de interactuar con ilimitados dispositivos.

Existen otros ecosistemas de datos similares a XAMPP, como por ejemplo WAMP para Windows y LAMP para Linux.

3.2. Solución propuesta

El análisis exhaustivo de las diferentes tecnologías nos ha llevado a elegir los siguientes componentes para el circuito electrónico:

El sensor de temperatura DS18B20 para controlar tanto la temperatura del agua al gusto del usuario como la temperatura ambiente en piscinas climatizadas. Aunque ambos sensores de temperatura comentados sean igual de válidos en cuanto a características (rangos de medición, tensión, etc.), este se ha elegido porque dispone de resistencia al agua, mientras que el sensor DHT11, no, lo cual conllevaría a buscar una carcasa de protección de líquidos. En función de la información proporcionada por el sensor:

- Se accionará el calentador del agua para mantener una temperatura deseada.
- Se ajustará la dosificación de productos de mantenimiento. La temperatura juega un papel importante en la aparición de organismos (a mayor temperatura, mayor favorecimiento de su aparición) y la solubilidad de los productos.

El sensor de nivel de agua elegido es el de detección de nivel, puesto que no necesitamos conocer el nivel del agua en todo momento para llegar al objetivo que determina si una piscina está suficientemente llena o no.

- Se accionará la llave de paso que permite el llenado de la piscina.

El sensor de presión diferencial MPX5500DP nos permitirá saber la presión ejercida por la bomba de agua.

- Se establecerá un rango de presión normal (entre 0.5 y 1 bares), otro rango de presión cuando el filtro está sucio (entre 1.5 y 2 bares) y un límite superior que alertará de una sobrepresión en el sistema. En este último caso, sería

conveniente detener el funcionamiento de la bomba y proceder a la limpieza de filtros para evitar que el sistema se dañe o reviente.

Como microcontrolador, se ha considerado la placa Wemos D1 R1 (ESP8266), la cual tiene un número de pines de entrada y salida tanto analógicos como digitales suficientes para conectar los elementos necesarios. Esta dispone de conexión WiFi sin la necesidad de otros módulos externos, se programa en el mismo entorno de Arduino y, como se ha comentado anteriormente, su capacidad de cómputo es superior a Arduino. Se podría reconocer como el corazón del proyecto. Su precio también ha sido un punto muy grande a favor.

- Se encargará de procesar las instrucciones e interactuar con todo el sistema electrónico y el servidor web.

Por último, se usarán relés para accionar los componentes que actúan sobre la piscina. Pero la simulación de estos componentes se hará con LEDs como ya se ha comentado en otros puntos de la memoria. Sobre un escenario real, habría que adaptar la conexión de los relés a estos componentes reales.

Para la parte del control y monitoreo del sistema a distancia, nos centramos en el uso de tecnologías de código abierto. Se hará uso de la plataforma XAMPP que incluye servidor web Apache y base de datos relacional mySQL. Se desarrollará una página web que permita manejar las funcionalidades mínimas establecidas.

Una vez establecida la solución, se pretende, por una parte, construir el circuito en una placa de pruebas y desarrollar el *software* que lo controlará y, por otra parte, el diseño del servidor que interactuará con el circuito. Primeramente, se desarrollará y probará individualmente cada parte independiente del circuito y su código: parte del sensor de temperatura, parte del sensor de nivel de agua, parte de la programación de horarios del filtro o del sistema de riego, etc. y su respectiva interacción con su actuador. Cuando se haya comprobado el correcto funcionamiento de cada parte, se unificará todo en un mismo programa de Arduino implementando funciones para cada parte que se llaman desde el código principal. A su vez, se desarrollará la estructura de la base de datos, el servidor web y la interacción entre ambos. Y como segundo paso, se hará una fase de pruebas que corrobore la interacción y funcionamiento adecuado de ambas partes como un todo.

3.3. Presupuesto

Llegamos a una parte interesante del proyecto donde vamos a estimar el coste del material y esfuerzo del ingeniero. Nos permitirá apreciar la diferencia en precio de esta solución a la problemática en comparación con las alternativas en el mercado actual. Es necesario incidir en que la mayor parte del coste pertenece al diseño y desarrollo

del prototipo, por lo que el precio en el mercado sería muy competitivo. Pese al corto alcance y las limitaciones de este trabajo, vamos a poder ejecutar con los medios especificados anteriormente gran parte de lo que sería el proyecto sobre un escenario real. Pues la parte orientada a los actuadores reales de la piscina no se incluirá en el presupuesto, sino que se tendrá en cuenta el importe del material para su simulación.

Por una parte, la tabla 3 identifica los recursos materiales necesarios y estima el número de unidades, su valor económico (IVA incluido) y la fuente de referencia del producto, dando paso a un importe subtotal del presupuesto.

Por otra parte, en la tabla 3 se muestra la estimación del coste proveniente de los recursos humanos. Se ha necesitado un ingeniero informático para llevar a cabo toda la investigación y el desarrollo del sistema electrónico (trabajo correspondiente al alumno). También, un desarrollador web para la creación del servidor web que maneja la parte de control del sistema físico (trabajo correspondiente al alumno). Y, por último, se incluye la supervisión del proyecto (por parte del tutor de este TFG). Para estimar el salario de cada puesto de trabajo se ha consultado la página web *Indeed*[15].

Tabla 3. Presupuesto: recursos materiales y humanos

RECURSOS MATERIALES			
COMPONENTE	**UNIDADES**	**VALORACIÓN ECONÓMICA (€)**	**REFERENCIA**
Placa de pruebas Protoboard MB-102	1	3.95	Electrón Perdido
Alimentación protoboard negra MB-102	1	1.99	Electrón Perdido
Reloj de tiempo real RTC DS3231 (3 uds.)	1	10.79	Amazon España
Sensor de temperatura DS18B20	1	7.53	RS Components
Sensor de detección de nivel (5 uds.)	1	4.5	Amazon España
Sensor de presión diferencial MPX5500DP	1	16.29	RS Components
Microcontrolador Wemos D1 R1 (ESP8266)	1	6.5	Electrón Perdido
Módulo Relé 5V	2	1.5	Electrón Perdido
Resistor 220Ω 1/2W	2	0.05	cespedes
Resistor 4700Ω 2W	1	0.19	cespedes
Diodo LED 3mm verde	1	0.21	cespedes
Diodo LED 3mm azul	1	0.75	cespedes

[15] **Web de estimación de salaries base:** Salarios | Indeed. Calcula el sueldo base de cada puesto de trabajo en un lugar determinado en base a una recopilación de salarios en publicaciones de empleo en la misma web y enviadas por trabajadores anónimos en los últimos 36 meses.

Cables puente macho-macho	1	4.33	cespedes
Cables puente macho-hembra	1	3.9	cespedes
SUBTOTAL (€):		**64.03**	
RECURSOS HUMANOS			
OCUPACIÓN	**UNIDADES**	**HORAS**	**COSTE (€/HORA)**
Ingeniero informático: diseño de sistemas informáticos	1	100	12.95
Programador Web	1	140	12.02
Supervisor	1	30	7.34
SUBTOTAL (€):		**3198**	
TOTAL (€):		**3262.03**	

Fuente: Elaboración propia

4. Diseño de la solución

En este capítulo, se especificarán las tecnologías utilizadas y se detallará el diseño elegido para el proyecto.

4.1. Tecnología Utilizada

El conjunto de tecnologías empleadas es bastante numeroso, pero cada una de ellas tiene un rol importante en su determinada parte del proyecto. Hay que destacar que todo el trabajo ha sido realizado en Windows 10, pero todas las tecnologías usadas tienen su adaptación en los sistemas operativos principales. En primer lugar, se definirá lo que es el pilar en el que se apoya toda la programación del sistema electrónico: Arduino; y, posteriormente, se describen aquellos lenguajes y entornos que componen el servidor web.

4.1.1. ARDUINO

"Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en *hardware* y *software* libre, flexible y fácil de utilizar para los creadores y desarrolladores" [7].

En realidad, Arduino son tres cosas a la vez (DSIIC, ETSID, UPV. 2020):

Hardware

Arduino proporciona diferentes tipos de placas con microcontrolador reprogramable y diferentes pines de entrada/salida según las necesidades de cada usuario. Cada una presenta sus propias características y aplicaciones destinadas a diferentes campos como puede ser el *IoT* o la impresión 3D. Además, Arduino también cuenta con los denominados *shields* que son componentes que se conectan a las placas extendiendo la funcionalidad. Pueden ser desde módulos *WiFi* o *Bluetooth* hasta sensores, pantallas o relojes de tiempo real. En resumen, son placas que permiten ejecutar programas que interactúan con los pines de entrada y salida y, por lo tanto, con sensores y actuadores conectados a ellos.

Al ser un *hardware* libre, Arduino proporciona un diseño base de placas en los que otros fabricantes se pueden basar para comercializar sus propias placas electrónicas.

Lenguaje de programación:

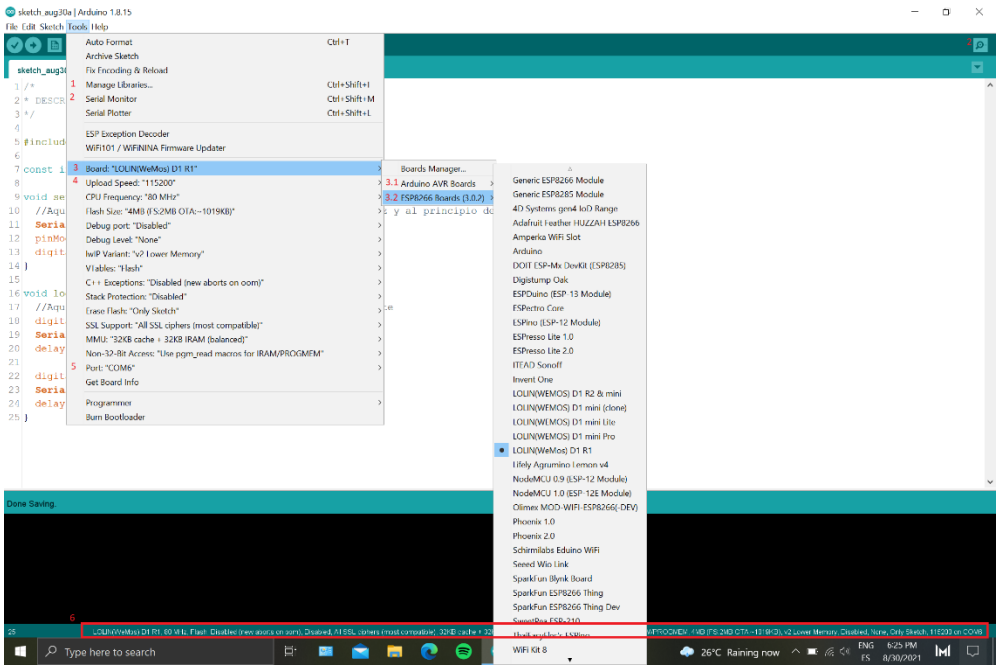
Arduino es un lenguaje de programación muy similar a "*Processing*", aunque este se basa en Java, y Arduino se basa en C/C++.

***Software*:**

Las placas Arduino se programan en el propio entorno de desarrollo proporcionado en la página oficial de Arduino llamado Arduino *Software* IDE o, coloquialmente, *sketch* de Arduino. Este es gratuito, libre y multiplataforma (compatible con sistemas operativos Linux, Windows y Mac OS).

La instalación y configuración del IDE de Arduino[16] y estructura básica de programa se describe a continuación: los archivos tienen la extensión ".ino" y se deben encontrar dentro de una carpeta con el mismo nombre para poder ser leídos por el IDE. Para instalar el IDE, simplemente hay que descargar el instalador de la página oficial y, a continuación, configurar el entorno según la placa de la que dispongamos:

Figura 10. IDE de Arduino: configuración del entorno



Fuente: Elaboración propia

1. Se nos permite descargar o importar librerías desde el manejador de librerías.
2. Atajo para abrir la impresión en pantalla del monitor en serie.
3. Se nos permite seleccionar la placa adecuada y descargar la familia de placas adecuadas desde el manejador de placas (recordemos que Arduino es un *hardware* libre)

   3.1. Aquí encontramos la amplia gama de placas originales Arduino.

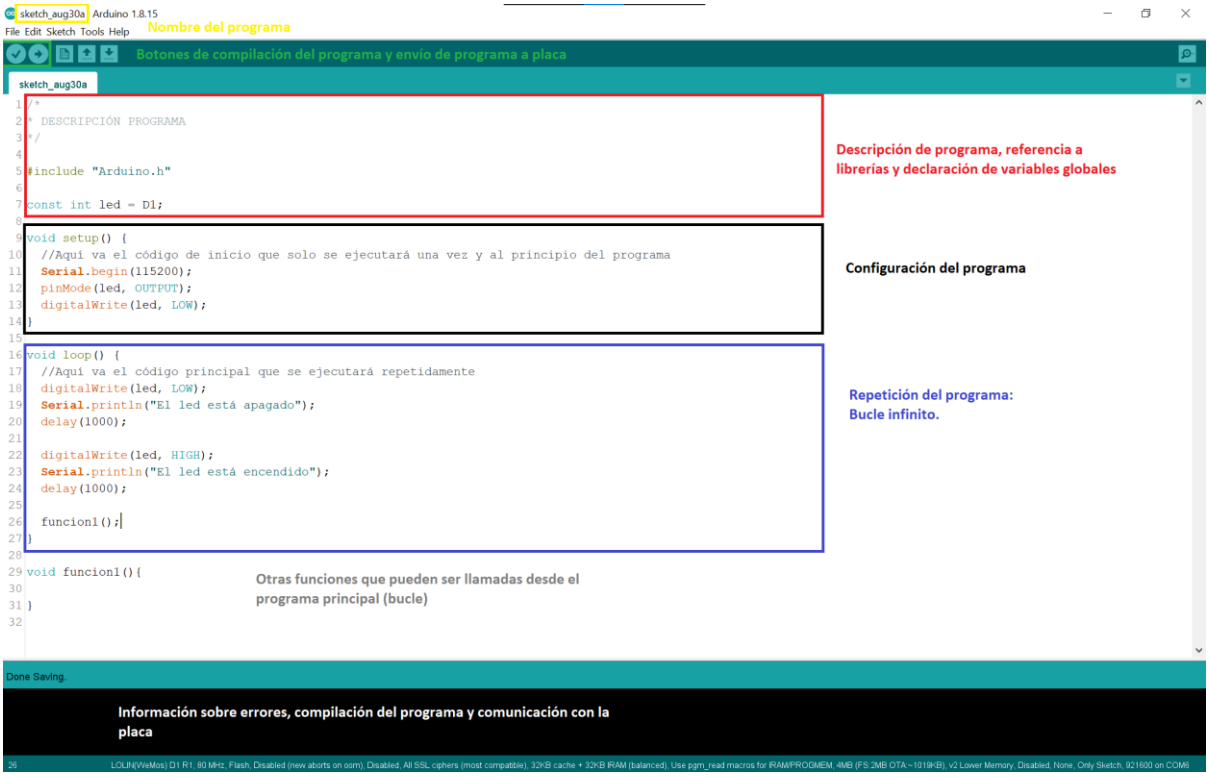
   3.2. Aquí encontramos la familia de placas basadas en el microcontrolador ESP8266 con WiFi.

[16] https://www.arduino.cc/en/software

4. Configuración de la velocidad de transmisión de datos con la que se comunica el monitor en serie en función de las prestaciones de cada microcontrolador. Para Arduino Mega 2560 es de 9600 baud/segundo (número de unidades de señal) por segundo y para ESP8266 será de 115200.
5. Es el puerto virtual en el que la placa se conecta al ordenador que corresponde a un puerto USB.
6. Información sobre la configuración actual de la placa en el IDE.

Cada programa o *sketch* de Arduino consta de un único archivo formado por tres partes básicas que se pueden ver marcadas por tres colores (rojo, negro y azul) en la siguiente figura:

Figura 11. IDE de Arduino: estructura básica de programa



Fuente: Elaboración propia

El cuadro rojo muestra la parte declarativa del programa donde se importan las librerías externas y se declaran las variables globales. El cuadro negro consta de la función *setup* que se ejecuta al iniciar el programa y se incluirán configuraciones iniciales en ella que solo se han de ejecutar una sola vez tales como iniciar el monitor en serie. El cuadro azul consta de la función *loop* que ejecuta el programa principal repetidamente hasta que se desconecta el microcontrolador o se sale anormalmente del programa. En esta se pueden invocar funciones externas (p.ej.: la función "funcion1" que se muestra en la imagen). Fuera de la estructura básica de un sketch

de Arduino encontramos el nombre del sketch (cuadro amarillo en la ilustración), botones de compilación y envío de datos a placa (cuadro verde) e información relevante de interacción programa-placa en la parte inferior del entorno.

Por otra parte, se especifican las tecnologías para la programación del servidor web y la base de datos:

4.1.2. Servidor Web Apache

Se trata del servidor HTTP de código abierto más popular para plataformas UNIX y Windows desarrollado por una comunidad pública donde los derechos de autor residen en el autor individual. Apache permitirá la comunicación entre el servidor web y el cliente mediante el protocolo HTTP a través del navegador. Su estructura se basa en módulos. Al contar con una comunidad tan grande, existe mucha documentación, un buen soporte y alta seguridad con actualizaciones constantes [8].

4.1.3. Base de datos relacional MySQL

MySQL es el sistema de gestión de bases de datos relacional de código abierto más popular para el desarrollo web. Funciona bajo la Licencia Pública General de GNU y la licencia comercial por Oracle Corporation. A diferencia de Apache, MySQL es patrocinado por una empresa privada que posee el *copyright* de la mayoría del *software* [9].

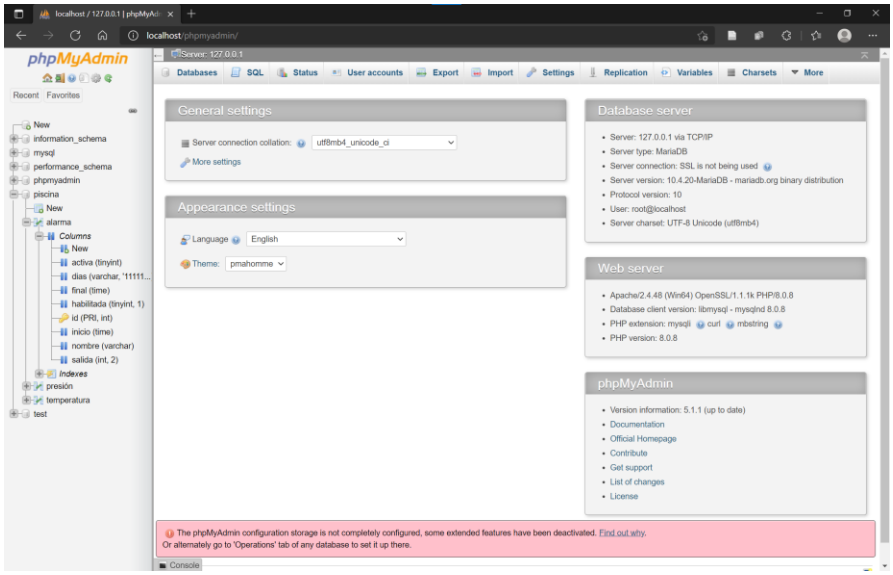
Un importante aspecto es su administración protagonizada por el *software* phpMyAdmin.

4.1.4. phpMyAdmin

Se trata de una herramienta de administración de MySQL de código abierto escrita en el propio lenguaje de programación web PHP y que se accede mediante el navegador web. Permite la gestión (creación, modificación y eliminación) de bases de datos, tablas, usuarios y permisos. También, la ejecución de *queries* SQL o la visualización de la relación entre tablas, entre muchas otras opciones. En el menú lateral izquierdo de la imagen 12, se aprecian las diferentes bases de datos y su estructura desplegable que muestra las tablas y sus columnas. Nuestra base de datos “piscina” guarda los elementos necesarios para el proyecto. Para acceder al gestor, hay que ingresar en el navegador “phpMyAdmin” después de la dirección IP asignada para Internet (si se accede desde Internet) o para la red local (si se accede desde la misma red local) del equipo en el que reside el servidor cuando se accede desde un equipo diferente o, la dirección *loopback* (localhost, 127.0.0.1) si se trata del mismo equipo [10].

Con el fin de brindar el acceso del servidor a dispositivos del área local, se ha modificado el archivo de configuración XAMPP\Apache\conf*httpd.conf* [11]. La estructura de directorios del servidor Apache se verá más adelante.

Figura 12. phpMyAdmin



Fuente: Elaboración propia

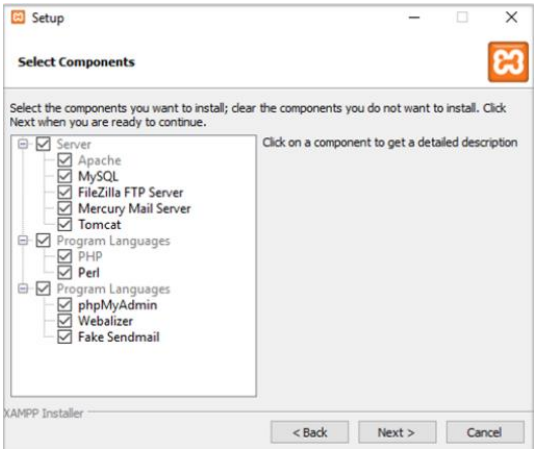
4.1.5. XAMPP

Se trata de un sistema de gestión de *software* libre multiplataforma. De acuerdo con la Licencia Pública General de GNU, XAMPP puede ser copiada con libertad. Este se compone de diferentes componentes: el servidor Web Apache, la base de datos MySQL, PHP y Perl, que cada uno funciona bajo su propia licencia [12].

El instalador se descarga desde la web oficial[17] y nos proporciona un conjunto de opciones de instalación. Generalmente, es una buena idea instalarlo directamente en la raíz de nuestro disco duro y, no en una subcarpeta. Como opciones de servidor encontramos, por ejemplo, Apache o Tomcat (contenedor de servlets); como lenguajes de programación web PHP o Perl; y, como otro tipo de gestores, phpMyAdmin. Nosotros estamos interesados en Apache y MySQL, PHP y phpMyAdmin. (*Web Design, University College Dublin, 2021. Unit: Server Side Scripting*)

[17] https://www.apachefriends.org/es/index.html

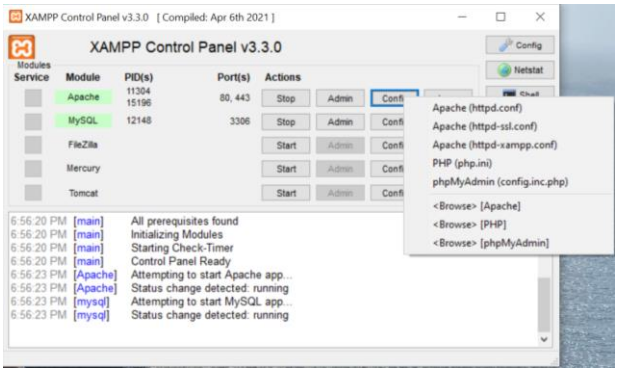
Figura 13. XAMPP: instalación



Fuente: Elaboración propia

XAMPP nos permitirá, mediante su panel de control, para manejar todas las herramientas de *software* para la web. Podremos arrancar las herramientas deseadas y establecer configuraciones mediante los archivos con la extensión ".conf".

Figura 14. XAMPP: Panel de control



Fuente: Elaboración propia

En cuanto a la estructura de directorios en Windows, encontramos que cada herramienta se encuentra en la raíz de la carpeta XAMPP. Por ejemplo, los archivos de configuración, instalación y ejecución de Apache se encontrarán en C:\XAMPP\Apache y los del lenguaje PHP en C:\XAMPP\PHP. Nuestro directorio de trabajo será C:\XAMPP\htdocs, donde residirán todos archivos de nuestra web. Cuando el cliente ejecute una petición HTTP al servidor, el servidor empezará a buscar los archivos necesarios en la raíz especificada, siendo esta "htdocs". Por defecto, XAMPP no permite el acceso al servidor desde otros dispositivos, pero esto es configurable y se pueden eliminar restricciones para permitir el acceso desde dispositivos de la misma red local o, incluso, desde Internet.

4.1.6. HTTP

Protocolo de transferencia de hipertexto es un protocolo que permite la comunicación entre los navegadores y servidores web mediante un modelo cliente-servidor sin estado (no se guardan datos entre peticiones) en el cual se envían peticiones por parte del cliente y se espera una respuesta por el servidor. Las peticiones HTTP constan de un método (GET, POST, DELETE, …) que se realizan a un determinado *host*. (*Web Design, University College Dublin, 2021. Unit: Server Side Scripting*)

4.1.7. PHP

"*Hypertext Preprocessor* es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y puede ser incrustado en HTML. Las páginas PHP (extensión ".php") permiten la ejecución de HTML y, a su vez de PHP haciendo uso de las etiquetas "<?php" de inicio y "?>" de final, incluyendo el código PHP entre ellas. Pues un navegador interpretaría el HTML y el PHP para mostrar la página web. Principalmente, PHP se basa en la programación de *scripts* del lado del servidor. Algunos ejemplos pueden ser la creación de webs con contenido dinámico o la transmisión de datos de un formulario entre páginas. Dispone de una gran documentación en línea tanto en su propia web oficial como en multitud de webs [13].

4.1.8. HTML (*Hypertext Markup Language)*

Se trata del lenguaje básico para la elaboración de páginas web que se ejecuta en el lado del cliente. Hipertexto es un concepto abstracto que se refiere a texto plano con referencias a otros documentos mediante hiperenlaces. Anotaciones (*markup*) se refiere al conjunto de datos incrustados mediante etiquetas "<etiqueta>" que definen la estructura del documento. Estos documentos (".html") escritos en lenguaje HTML son analizados sintácticamente por el navegador y traducidos visualmente.

(*Web Design, University College Dublin, 2021. Unit: HTML*)

4.1.9. Css (*Cascading Style Sheets*)

Es un lenguaje web que compenetra con el HTML con el fin de dar al desarrollador el control absoluto de la representación del contenido de páginas web. Se ejecuta en el lado del cliente. Así pues, método en cascada (*cascading*) se refiere a la manera en que las opciones de estilo pueden ser aplicadas y combinadas estableciendo un orden de precedencia entre ellas, el estilo se refiere a la presentación visual y hojas (*sheets*) al almacenamiento del código CSS en documentos (".css") o incrustado dentro del archivo html.

(*Web Design, University College Dublin, 2021. Unit: CSS*)

4.1.10. Javascript

Es un lenguaje de programación multipropósito que se ejecuta en el lado del cliente. Principalmente se usa para la visualización de datos, el desarrollo web y el desarrollo *full stack*. Con él se puede realizar desde partes dinámicas de una web hasta aspectos visuales. Se puede combinar con otros lenguajes.

(*Web Design, University College Dublin, 2021. Unit: Javascript*)

4.1.11. AJAX

Significa Javascript y XML Asíncrono por lo que es una técnica para programar webs con funcionalidades asíncronas, permitiendo la interacción "con estado" del cliente con el servidor. Esto es la capacidad de transmisión de datos entre ellos sin recargar la página web, lo cual facilita muchos aspectos de fluidez y comunicación entre códigos ejecutados en los dos lados. XML es un lenguaje para almacenar datos comprensibles por el humano [14].

4.1.12. Visual studio code (VSC)

Es un editor de código fuente de Microsoft y disponible en los sistemas operativos principales. Incluye servicios como mercado de extensiones, control de Git integrado o depurador de código. Es un entorno de desarrollo bastante ligero e intuitivo. El entorno de desarrollo elegido ha sido Visual Studio Code por su ligereza y por haber trabajado anteriormente con él [15].

4.2. Arquitectura del Sistema

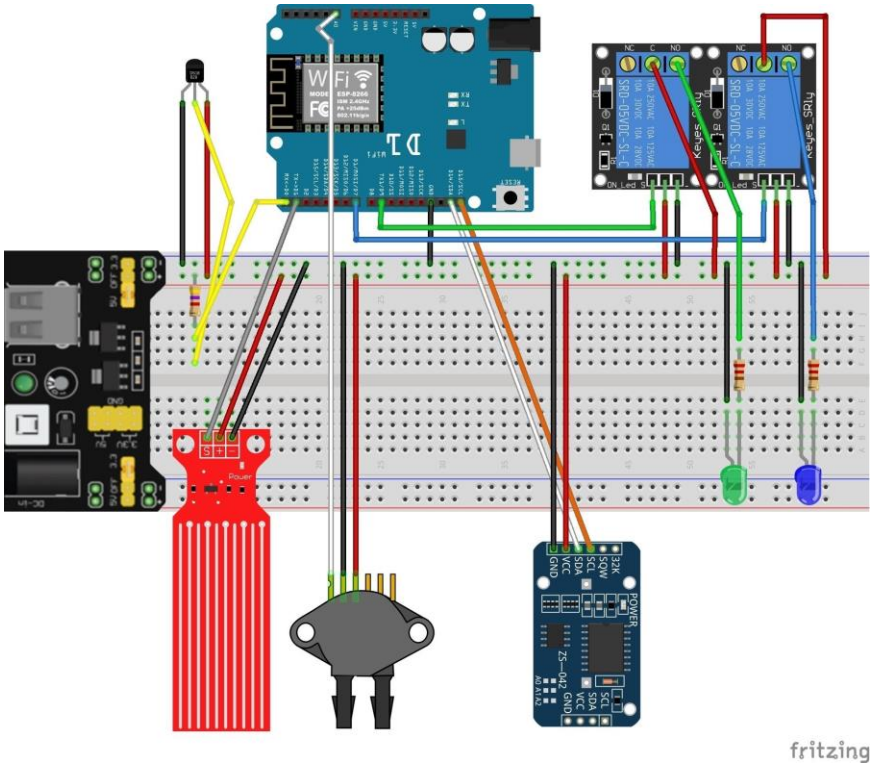
Se pretende describir minuciosamente la arquitectura del sistema y la interacción entre los componentes. Se tiene de corazón del sistema a la placa Wemos D1 que mediante los pines interactúa con circuitos independientes. En la siguiente ilustración vemos la interconexión de los sensores y accionadores. Se ha utilizado la herramienta de software *Fritzing*[18] para su elaboración. Pues podemos subdividir el esquema en los siguientes circuitos: circuito del sensor de temperatura, se sitúa en el extremo superior izquierdo (cableado amarillo); circuito de detección de nivel de agua (cableado gris), situado debajo del circuito de temperatura; circuito del sensor de

[18] https://fritzing.org/

presión (cableado blanco), situado a la derecha del circuito de sensor de detección de nivel; circuito del reloj de tiempo real (cableado naranja y blanco), se sitúa en la parte inferior derecha; y, en último lugar, el circuito actuador (cableado verde y azul), se localiza en la esquina superior derecha de la imagen.

Es importante resaltar la necesidad de tantos relés como elementos se deseen actuar. Del mismo modo, se tiene la necesidad de disponer de tantas electroválvulas como conductos se quieran controlar (abrir o cerrar), incluyendo las bombas de agua. Aunque en la figura 15 se muestre tan solo el uso de dos relés, habrá tantos como sean necesarios. Pues el circuito probado ha sido con hasta tres relés y LEDs simultáneos.

Figura 15. Prototipo del sistema electrónico



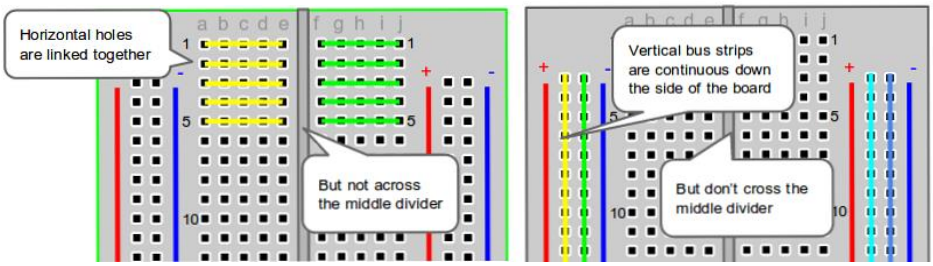
Fuente: Elaboración propia

Se detallan los componentes individualmente a continuación:

Placa de prototipado (*breadboard*): protoboard MB-102

Figura 16. Funcionamiento de la placa de prototipado (1)

Figura 17. Funcionamiento de la placa de prototipado (2)



Fuente: jueaccentsconagua.com

Una placa de pruebas sirve para montar y probar circuitos electrónicos antes de llegar a una versión definitiva. En la imagen superior izquierda se puede ver cómo funcionan los agujeros centrales de la placa: las columnas 'a', 'b', 'c', 'd' y 'e' se conectan entre sí horizontalmente, pero no verticalmente. Al igual que, en la otra parte del separador, 'f', 'g', 'h', 'i' y 'j'. Esta parte de la placa se utiliza para conectar los componentes del circuito, interactuando entre sí y con la alimentación. En la imagen superior derecha, apreciamos cómo es la conexión en ambas partes exteriores de la placa de pruebas: la conductividad es vertical en toda la línea, pero no horizontal. Estas líneas laterales se suelen utilizar para conectar el voltaje y la tierra (alimentación del circuito).
El modelo elegido es "protoboard MB-102" que consta con 830 agujeros.

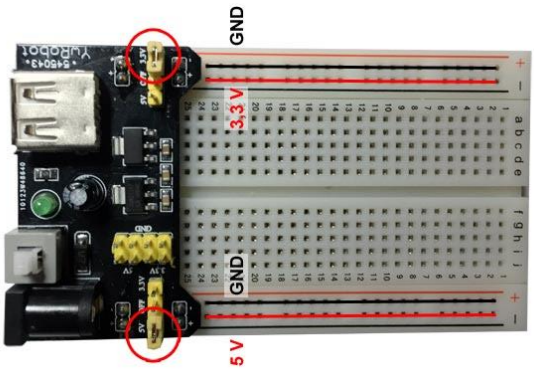
Cable puente

Son cables conductores de electricidad que se usan para la interconexión de componentes eléctricos. Existen dos tipos de conector: macho (terminación en punta y se conecta a un pin hembra) y hembra (recibe la conexión de un conector macho).
En este proyecto se han necesitado tanto cables puente macho-macho como macho-hembra.

Fuente de alimentación

Permite alimentar el sistema electrónico con una tensión y una corriente máxima determinados.
El microcontrolador elegido es capaz de suministrar tensiones de 3.3 o 5 voltios. Aunque la corriente pico puede que no sea suficiente para el correcto funcionamiento del circuito en algunos casos como cuando se conectan múltiples dispositivos que consumen más de lo que es capaz de suministrar. Debido a ello, se ha optado por un módulo alimentador de la *protoboard* MB-102, capaz de suministrar dos canales simultáneamente de 3.3 o 5 voltios según se configure (figura 18) y una corriente máxima de salida de 700mA. Esta funciona con una fuente externa de entrada de entre 7 y 12 V [16].

Figura 18. Fuente de alimentación



Fuente: programarfacil.com

Reloj de tiempo real RTC DS3231

Este módulo dispone de una pila que le permite mantener la hora actual, aunque deje de ser alimentado por una fuente externa. Es el elemento que se muestra en la parte inferior derecha de la ilustración 10. Consta de 6 pines, de los cuales usaremos los de alimentación (VCC y tierra) y los de conexión I2C[19] (SDA y SCL). En él se pueden programar hasta dos alarmas con ayuda de la librería específica "RTClib". Sin embargo, tan solo usaremos el RTC para saber la hora en todo momento [17].

Sensor de temperatura DS18B20

Este sensor de temperatura sumergible nos permite la medición en un rango de temperaturas de -55 a 125 grados centígrados con una resolución configurable[20]. Las características de dicho sensor se han definido previamente en el apartado 3.1. Dispone de dos pines para la alimentación (VCC y GND) y el bus digital de datos D_Q. El conexionado del sensor dentro del circuito (se puede ver en la figura 10) necesita una resistencia de *pull up* de 4.7kΩ para mantener un nivel de tensión alto en la entrada digital cuando haya interferencias o perturbaciones en la entrada.

Al igual que el rtc, DS18B20 funciona con un protocolo de comunicación en serie muy similar que permite la conexión de múltiples sensores al mismo pin digital de datos. Dicho protocolo se denomina *One-wire* y también se necesita una librería específica que facilita su uso [18][19].

[19] **Protocolo de comunicación I2C:** protocolo de comunicación serial para transmitir datos entre múltiples dispositivos conectados al mismo pin. Se distinguen los dispositivos mediante sus direcciones físicas.

[20] **Resolución:** indica la precisión de medida de una variable. Este sensor es configurable a 9, 10, 11 o 12 bits, lo que corresponde a una precisión de 0.5, 0.25, 0.125 y 0.0625°C, respectivamente.

Sensor de detección de nivel de agua

Se trata de un sensor digital que tiene tres pines: los dos de alimentación y ‘s’ que es la señal. Se recibe un ‘0’ cuando no hay contacto con el líquido y un ‘1’ cuando se detecta que el agua toca el sensor. Por ello, lo situaremos en la superficie de la piscina y se activará el suministro de agua nueva para llenar la piscina cuando se detecte una señal baja y se cerrará la llave de paso del agua cuando se reciba una señal alta.

Sensor de presión diferencial MPX5500DP

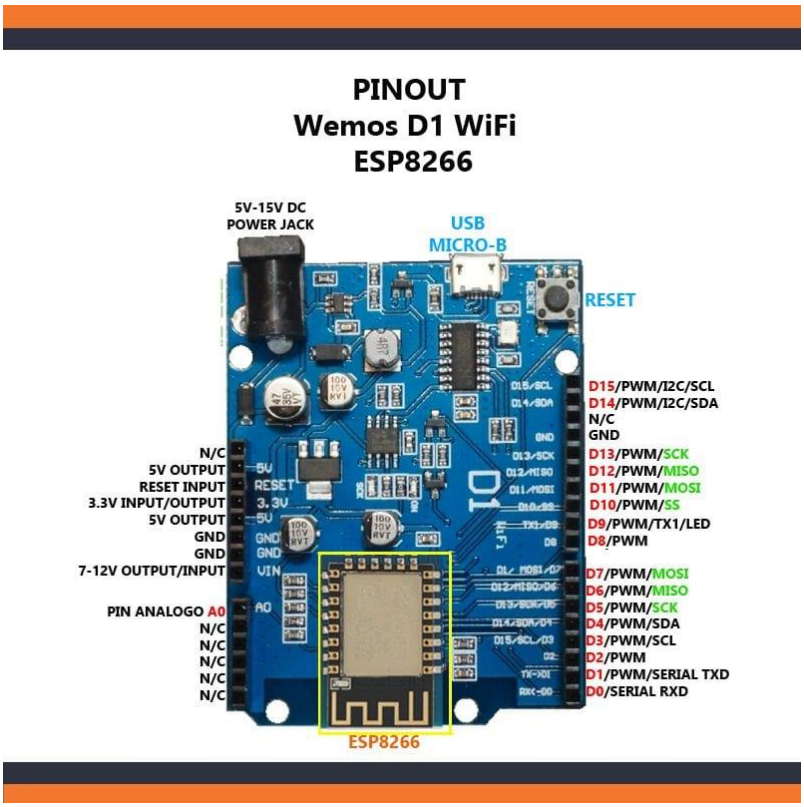
Se pretende medir el flujo del caudal que pasa por la tubería posterior al filtro del agua. Dará la seguridad del sistema de conocer cuándo hay una sobrepresión para desactivar la bomba de agua. Es un sensor analógico que consta de 6 pines, de los cuales usaremos los pines 1, 2 y 3 correspondientes a VCC, GND y datos, respectivamente. Es capaz de medir un rango de GND a VCC voltios con una resolución de 10 bits (valores de 0 a 1023). Por lo que si lo suministramos con una tensión de 3.3V, podemos hacer una conversión por unidad: 3.3 /1024 = 0.003222 voltios por unidad. En la hoja técnica, se nos proporciona una fórmula de relación presión-voltaje para el sensor. Despejando la incógnita de la presión, obtendremos el valor deseado a conocer:

$P = \frac{\mathrm{Vout} - 0.004 * \mathrm{Vs} \pm \mathrm{Error}}{0.0018 * Vs}$, siendo P, la presión; Vout, el voltaje proporcionado por el sensor; Vs, la tensión que alimenta al sensor; y, Error, un 2.5% del voltaje a escala completa [20].

Hay que aclarar que la placa Wemos D1 solo es capaz de obtener voltajes de GND a 3.3V a través del pin analógico, por lo que sería recomendable suministrar el sensor con 3.3V para limitar y adecuar su rango al microcontrolador, o adaptar el resultado obtenido mediante la programación.

Microcontrolador Wemos D1 R1 (ESP8266)

Figura 19. Wemos D1 WiFi ESP8266



Fuente: uelectronics.com

Tras haber estudiado exhaustivamente qué placa con microcontrolador se adecúa más a las necesidades del proyecto, se ha tomado la decisión de adquirir la placa Wemos D1. La arquitectura del microprocesador ESP8266EX consta de una arquitectura de 32 bits, su propia RAM, memoria Flash de 4MB, antena WiFi en PCB con banda 2.4 GHz 802.11 b/g/n (velocidades de hasta 11, 54 y 600 Mbps según la distancia al punto de conexión), alimentación de 3.3V y un consumo típico de 70mA y pico de 400mA, frecuencia de reloj de 80 o 160 MHz, 11 entradas/salidas digitales y 1 analógica capaz de medir desde 0 a 3.3V. Cuenta, además, con interfaces I2C, SPI y AURT (protocolos de comunicación). Su programación es en Arduino, por lo que el sistema se sigue basando en Arduino. En la imagen anterior, se detalla su esquema de pines.

Módulo Relé (1 canal) 5V

Es un dispositivo preaccionador electromagnético que permite controlar un circuito eléctrico con su juego de contactos (un contacto está normalmente cerrado o en contacto con el común y, el otro, abierto, sin contacto con el común). Consta de una bobina que al alimentarse de carga eléctrica se crea un campo electromagnético que atrae y gira el electroimán, de manera que se cambia el estado de los contactos

(funcionando como un interruptor cerrado) permitiendo el paso de corriente por el contacto conectado al circuito de salida del relé.
Este componente es necesario para decidir cuándo activar de manera automatizada con la programación cada uno de los actuadores de la piscina o el riego. En este proyecto se simulan los actuadores con el encendido de un LED, tal y como se muestran en la parte superior derecha de la figura 10.
Disponemos de relés que funcionan a 5 voltios. La descripción de su conexión es la siguiente:
Este se forma por tres patillas nombradas como '+', '-' y 's' conectadas a VCC, tierra y pin de salida digital del microcontrolador. Luego dispone de tres contactos: normalmente abierto (NO), común (C) y normalmente cerrado (NC) que se conectan a la alimentación positiva del actuador o led en nuestro caso, a VCC y sin conectar, respectivamente.

Resistencias
Es un componente que añade resistencia eléctrica entre dos puntos de un circuito, haciendo posible la obtención de una corriente y voltaje diferente en ciertos puntos del circuito. Aquí entra en juego la ley de Ohm ($V = I \cdot R$).
Un LED opera con unas características eléctricas determinadas. Si se superan hay riesgo de daño. Se recomienda que no se supere una intensidad de 23 mA. Por ejemplo, si el voltaje suministrado es de 5 voltios, aplicando la ley de Ohm obtenemos que necesitamos una $R = 5 / 0.023 = 220$ ohmios para limitar dicha intensidad.
Se adquieren dos resistencias de 220 ohmios que se conectarán en serie con el cátodo de cada LED.

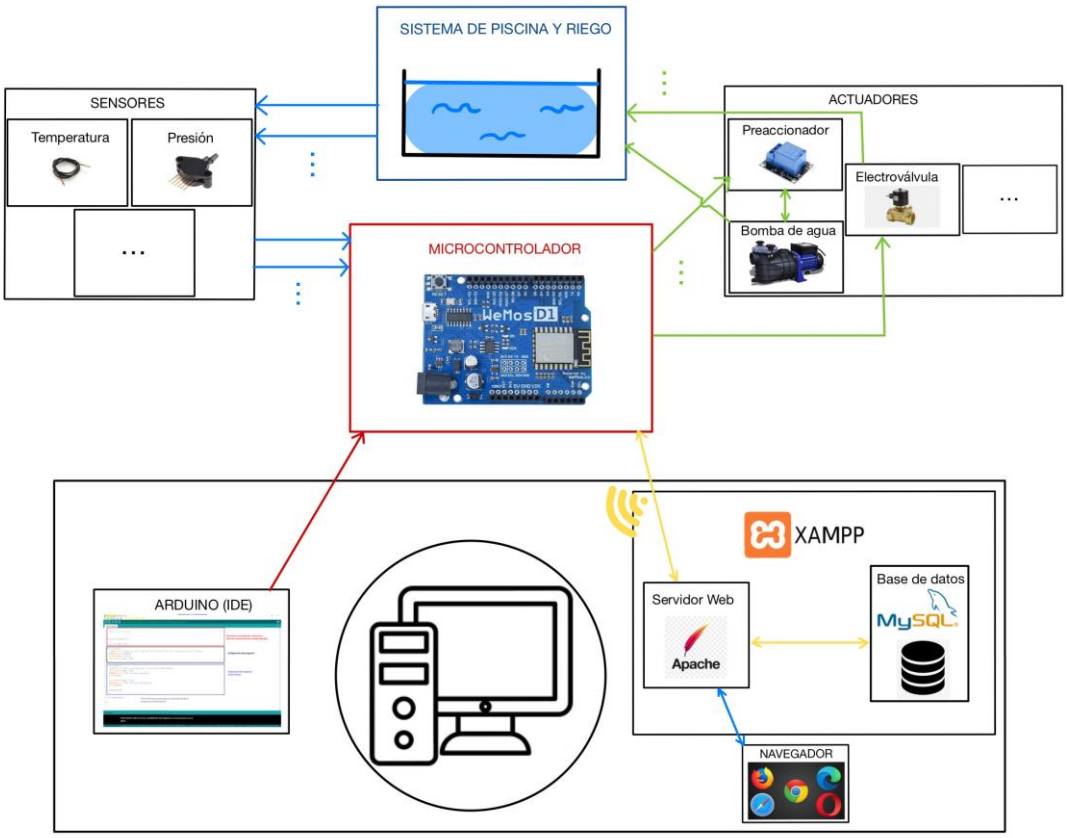
Diodo LED
Un diodo es un componente eléctrico semiconductor unidireccional: permite solo que la corriente eléctrica circule de su electrodo positivo al negativo, es decir, de su cátodo a su ánodo, pero no al contrario. Y en el caso del LED (*Light Emitting Diode*) transforma corriente en luz [21].
Disponemos de un LED verde y otro azul.

Finalmente, se programará con Arduino la interacción de los componentes anteriores, además de con el servidor.

Por otra parte, nuestro servidor y base de datos se basan en el conjunto de tecnologías explicadas en el apartado anterior. El servidor web estará compuesto de páginas web HTML y PHP. Con ayuda del resto de lenguajes refinaremos la interacción entre las múltiples páginas del servidor: Javascript para la creación de elementos dinámicos en la web, AJAX para la comunicación de códigos ejecutados en los dos lados (servidor y cliente) y, CSS para la presentación. En la configuración de nuestro proyecto, Arduino mandará peticiones HTTP al servidor para obtener información de la base de datos y actuar en consecuencia. Visual Studio Code será nuestro entorno de desarrollo en el que estableceremos nuestro directorio de trabajo con su estructura de archivos, programados en los diferentes lenguajes, que compondrán el servidor.

Una vez conocidos todos los componentes, podemos ver cómo el sistema electrónico interactúa con la base de datos y el servidor como un todo:

Figura 20. Diagrama de interacción del sistema



Fuente: Elaboración propia

4.3. Diseño Detallado

4.3.1. Sketch de Arduino

La estructura implementada contiene las partes básicas de un programa de Arduino especificadas en el apartado 4.1.1. y, en cada una, añadimos cuatro apartados correspondientes al código de cada circuito: reloj de tiempo real y rutinas, sensor de temperatura, sensor de presión y sensor de detección de nivel de líquido. También se añade el código correspondiente a la conexión WiFi y para los accionadores (LEDs) cuando procede. Se añade el código en el anexo correspondiente.

En primer lugar, se importan las librerías correspondientes. Además, se crea una librería en C para manejar objetos rutina. Estos objetos contienen un atributo por cada columna que nos interese de la tabla correspondiente a las rutinas. Y, posteriormente, se declaran las variables globales que son aquellas que deseemos que tengan visibilidad desde cualquier parte del programa. Se ha requerido declarar variables globales para el manejo del WiFi, las rutinas y reloj de tiempo real, la temperatura, la presión y, los accionadores (LEDs).

En segundo lugar, tenemos la función *setup()* en la que inicializamos aquellos aspectos que solo se vayan a ejecutar una vez al principio del programa. Podemos encontrar la inicialización de la conexión a la red local, del rtc y del monitor serial, y la asociación del pin analógico como entrada.

En tercer lugar, encontramos el código principal. La función *loop()* se ejecuta repetidamente hasta que se salga del programa anormalmente. Aquí vamos a leer los valores que nos envían los sensores de temperatura, detección de nivel de líquido y presión y a manejar las rutinas. Para ello se hace la llamada a cuatro funciones que corresponden a cada uno de los objetivos.

Por último, se escriben cinco secciones de código. Hay que resaltar que se busca la creación de funciones con un solo objetivo. Por lo que se crean funciones diferentes para cada propósito dentro de cada sección:

1. WiFi: contiene una función que envía peticiones HTTP al servidor. Se hace uso de las funciones proporcionadas por las librerías de la conexión WiFi. La función obtiene dos parámetros String: la ruta del servidor que se quiere ejecutar y los parámetros para el método HTTP GET. De esta forma, la Wemos D1 puede enviar peticiones HTTP e intercambiar datos con el servidor y base de datos. En el siguiente punto de la memoria, se explica la estructura del servidor. Pues se especificará una carpeta includes\Arduino donde tenemos estas clases para ser llamadas desde el microcontrolador y que atienden a las peticiones HTTP.

2. Rutinas[21]: se compone de la función inicializarAlarmas() que comprueba si hay actualizaciones (algún cambio en la tabla de la base de datos), obtiene los objetos rutina correspondientes a cada fila de la tabla y los almacena en una lista de rutinas. Una segunda función comprobarAlarmas() compara el inicio de cada objeto rutina con la hora actual recorriendo la lista de rutinas. Si la rutina está activa, se activa el accionador correspondiente (simulado con un LED).
3. Temperatura: se tienen tres funciones obtenerTemperatura(), obtenerRangoDeseado() y accionarCalentador(). La primera obtiene la temperatura que proporciona el sensor, la segunda obtiene el rango de temperatura óptima que el cliente ha especificado en el servidor y la tercera enciende el calentador si la temperatura actual está por debajo de un mínimo o avisa al servidor si supera un máximo.
4. Nivel de agua: contiene una función que abre la llave de paso del agua si la piscina no alcanza un nivel de agua determinado y la cierra si se encuentra en el nivel deseado.
5. Nivel de presión: obtiene la presión que causa el flujo del agua sobre los filtros y valora si se encuentra por encima de un límite. Si es así, se apaga la bomba de agua para evitar daños.

Otro aspecto general que considerar: los LEDs (accionadores) se asocian a un pin digital y se encienden o apagan simulando los accionadores a lo largo del código. Cada rutina se asociará a una salida digital de la placa que acciona un LED. Como se pueden crear varias rutinas diferentes, se pueden tener muchos LEDs, sin embargo, el código es probado con solo tres LEDs en total.

4.3.2. Estructura del servidor web y su interacción con la base de datos

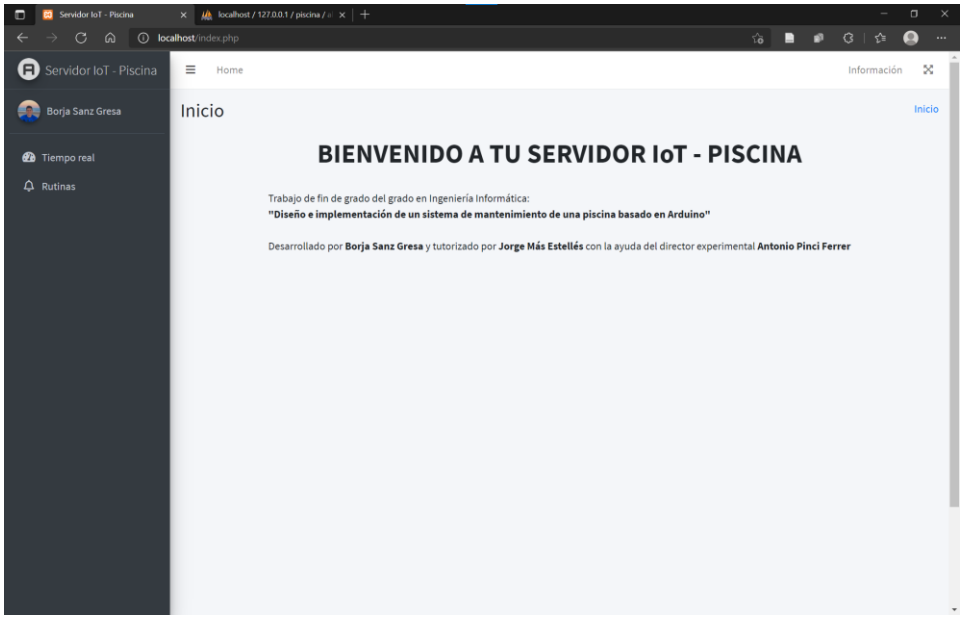
La estructura de la base de datos se compone por tres tablas correspondientes al manejo de las rutinas y la temperatura. Las rutinas son objetos con los siguientes atributos (columnas): identificador, inicio de la rutina, fin de la rutina, los días de la semana en qué se programa, si está activa, si está habilitada, en número de la salida asociada a Arduino, y un nombre. Estos objetos rutina interactúan con los objetos rutina del sketch de Arduino. La tabla de la temperatura contiene tres columnas: temperatura actual, temperatura máxima y mínima. Estos dos últimos determinan el rango deseado para la temperatura del agua.

Pasamos a detallar la estructura del servidor. Antes que nada, hay que destacar que para el diseño de la web se ha hecho uso de una plantilla adquirida de la mano de *AdminLTE*[22] para un uso no comercial. Esta plantilla es de código abierto y consta de un diseño que se adapta a cualquier tamaño de pantalla y ventana (*responsive design*). Se proporciona el código HTML, las hojas de estilo CSS y el código Javascript

[21] Véase que rutina y alarma se usan como sinónimos en el código. Pues su significado es el de la programación de un horario que tiene un inicio, un periodo activo y un fin.
[22] Free Bootstrap Admin Template | AdminLTE.IO

para la interacción dinámica. De esta manera, tan solo se ha tenido que buscar el código fuente de cada elemento deseado, copiarlo y adaptarlo al gusto. Se proporciona una vista previa de las páginas del servidor en las siguientes imágenes:

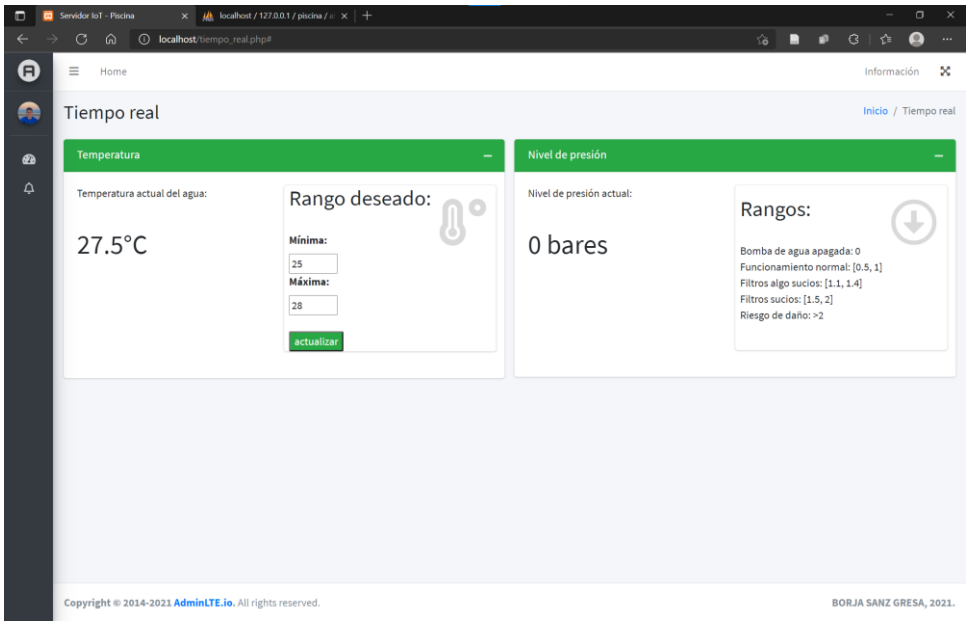
Figura 21. Servidor web: página de inicio



Fuente: Elaboración propia

La figura 21 muestra la página de inicio por defecto del servidor. El archivo que se cargará por defecto al acceder al servidor se puede configurar dentro de XAMPP.

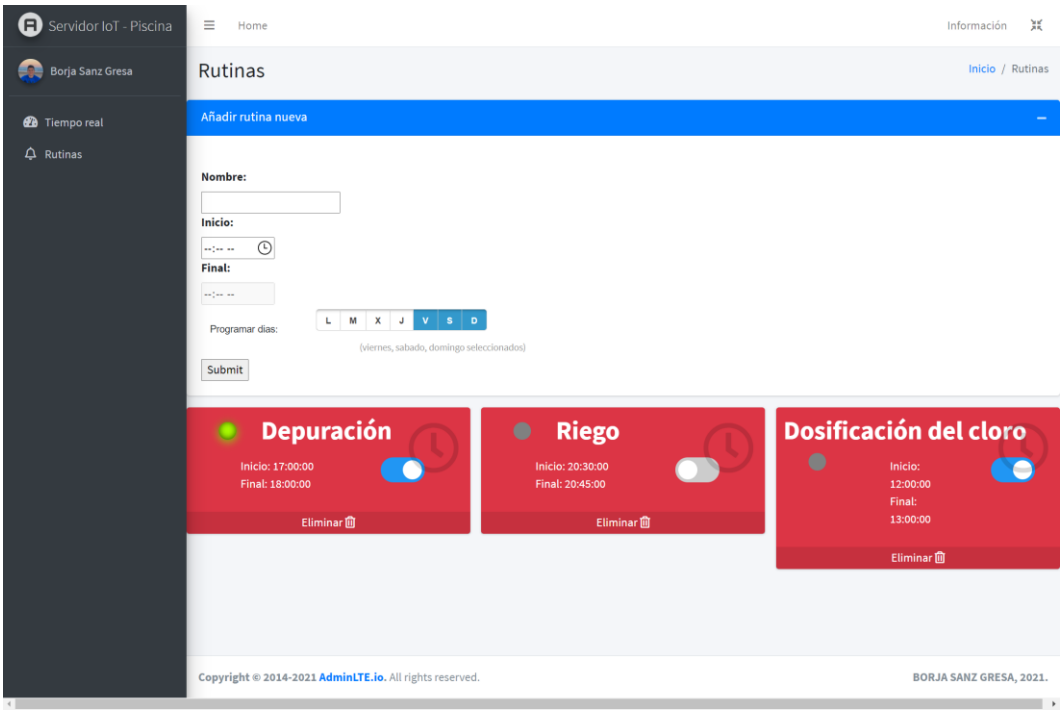
Figura 222. Servidor web: vista de tiempo real



Fuente: Elaboración propia

Esta página (figura 22) muestra los datos en tiempo real: se actualizan cada vez que se recarga la página. Tenemos el sensor de temperatura que nos muestra mediante un *widget* la temperatura actual y nos deja configurar el rango deseado al que mantener nuestra agua. En el caso de salirse de los límites, se acciona el calentador o salta una alerta. Se proporciona también información sobre la presión obtenida por el sensor correspondiente. Véase que el usuario no puede modificar los rangos de presión óptimos por seguridad del sistema.

Figura 23. Servidor web: vista de las rutinas

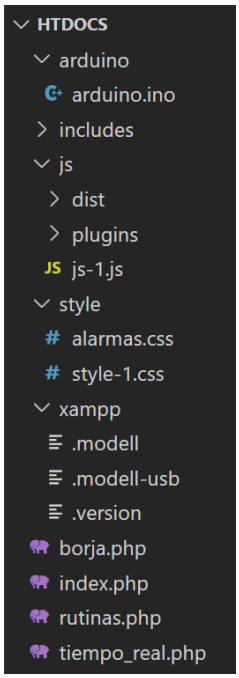


Fuente: Elaboración propia

En esta página (figura 23) se muestra un menú desplegable para añadir una rutina nueva, especificando para ella un nombre, el inicio, el final y qué días de la semana se desea programar. Más abajo, se proporcionan *widgets* rojos que enseñan cada rutina ya creada y guardada en la base de datos. Dentro de estas tarjetas rojas, se puede eliminar, habilitar o deshabilitar la rutina, así como ver su inicio y fin y si está activa en estos momentos (LED: verde para encendida y gris para apagada).

Como se ha comentado previamente, nuestro directorio de trabajo se encuentra en C:\XAMPP\htdocs que contiene la siguiente estructura de proyecto:

Figura 244. Servidor web: estructura de htdocs



Fuente: Elaboración propia

Arduino\arduino.ino: corresponde al sketch de Arduino. Aunque sea una parte totalmente independiente del servidor, se ha situado aquí para tener todo el diseño del producto en el directorio de trabajo.

El contenido de **htdocs\includes** se explicará detalladamente más adelante.

js\js-1.js: se trata del archivo javascript creado para el proyecto.

js\dist y js\plugins: se trata de los archivos CSS y Jasvascript proporcionados por la plantilla *AdminLTE*.

style: contiene las hojas de personalización css creadas para el proyecto.

xampp: contiene información sobre la herramienta XAMPP. No relevante para la estructura del proyecto.

Archivos php en la raíz de XAMPP\htdocs: son las diferentes páginas del servidor. Se acceden mediante la dirección URL "http://localhost/" seguida del nombre del archivo .php. Todos estos archivos tienen la misma estructura que se explica a continuación:

Figura 25. Servidor web: Estructura de index.php

index.php

```
1  <?php
2  include('./includes/head.php');
3  include('./includes/header.php');
4  include('./includes/sidebar.php');
5  ?>
6
7  <!-- Content Wrapper. Contains page content -->
8  <div class="content-wrapper">
9    <!-- Content Header (Page header) -->
10     <div class="content-header">
11       <div class="container-fluid">
12         <div class="row mb-2">
13           <div class="col-sm-6">
14             <h1 class="m-0">Inicio</h1>
15           </div><!-- /.col -->
16           <div class="col-sm-6">
17             <ol class="breadcrumb float-sm-right">
18               <li class="breadcrumb-item"><a href="./index.php">Inicio</a></li>
19             </ol>
20           </div><!-- /.col -->
21         </div><!-- /.row -->
22       </div><!-- /.container-fluid -->
23     </div>
24     <!-- /.content-header -->
25
26   <div class="div-centered">
27     <h1><b>BIENVENIDO A TU SERVIDOR IoT - PISCINA</b></h1><br>
28     <p>Trabajo de fin de grado del grado en Ingeniería Informática:<br>
29       <b>"Diseño e implementación de un sistema de mantenimiento de una piscina basado en Arduino"</b><br><br>
30       Desarrollado por <b>Borja Sanz Gresa</b> y tutorizado por <b>Jorge Más Estellés</b>
31       con la ayuda del director experimental <b>Antonio Pinci Ferrer</b>
32     </p>
33   </div>
34 </div>
35
36 <?php
37 include('./includes/footer.php');
38 include('./includes/scripts.php');
39 ?>
```

Fuente: Elaboración propia

Vemos que al inicio y al final del archivo (fig. 25: líneas 1-5 y 36-39) se incluye la referencia a otros documentos php mediante la sentencia “include(‘ruta relativa’)”. De esta manera estamos incrustando el código de esos documentos en el documento actual, evitándonos repetir dicho código. Esta es una técnica útil para cuando haya que reutilizar el código múltiples veces. Pues en este caso, a cada página principal php se le incrustarán los siguientes elementos que puede ser apreciados en la figura 25:

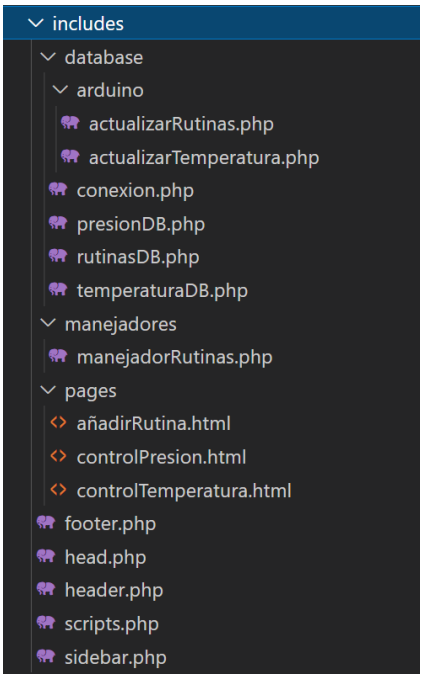
- *head.php*: corresponde a la cabecera de la página HTML, siendo esta una parte de la estructura básica de las páginas HTML. Esta incluye las etiquetas “html”, “head”, “meta”, “*title*” y “*link*”. Mediante la etiqueta *link* se establece la referencia a los archivos CSS.
- *header.php*: contiene el inicio del cuerpo (etiqueta “*body*”) y la barra de navegación superior.
- *sidebar.php*: código para la barra lateral desplegable con los enlaces a cada elemento que esta contiene.

- *footer.php*: contiene la barra inferior con los datos de los creadores.
- *scripts.php*: contiene las referencias a los scripts de javascript externos. Se incrustan al final de las páginas web para que se ejecuten una vez la página haya sido cargada del todo. Si se hace al contrario, puede que no funcionen porque los elementos a los que el script llama no se han creado todavía.

El código HTML presentado en medio de los códigos php (líneas 7 a 34, figura 25) es el contenido principal específico de cada página y el cual no se repite en ninguna otra.

Nos queda detallar la estructura de la carpeta *includes* dentro de *htdocs*. Aquí se sitúa casi todo el código del servidor, ya que las clases principales incrustan el código repetitivo desde estas. Podemos apreciar la presencia de los archivos head.php, footer.php, etc.

Figura 26. Servidor web: estructura de htdocs\includes



Fuente: Elaboración propia

pages: directorio que incluye las páginas html que se llamarán desde otros archivos para ser incrustadas en estos. Por ejemplo, dentro de la tarjeta de la temperatura en la figura 2, se incrusta el código de "controlTemperatura.html" al llamar a este archivo. Al igual, dentro de la tarjeta de la presión se incrusta el contenido de "controlPresion.html" y, "añadirRutina.html" dentro de la tarjeta azul de la figura 23.

database: maneja todos los archivos que proporcionan métodos para la interacción con la base de datos de la piscina.

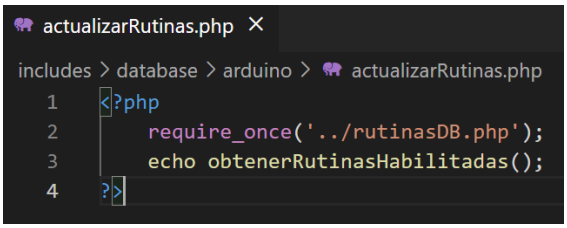
- \conexion.php: código para iniciar una conexión con la base de datos de la piscina. Es llamado desde los archivos presionDB.php, rutinasDB.php y temperaturaDB.php para manejar las tablas de datos correspondientes.

- \presionDB.php, \rutinasDB.php y \temperaturaDB.php: proporcionan métodos para la gestión de sus tablas correspondientes en la base de datos mediante *queries* SQL. Se proporcionan métodos *get* y *set* para cada columna de las tablas. Por ejemplo, la tabla de la presión se forma por las columnas que almacenan los diferentes rangos deseados identificados por "id", y especificando los límites "min" y "max". Se incluyen, pues, los métodos getId(), getMin(), setMin($min), getMax() y setMax($max).

- \arduino: contiene los archivos *wrappers* que se invocan desde Arduino. Estos contienen la llamada a una función que devuelve un String de alguno de los

archivos temperaturaDB o rutinasDB como en la imagen 27. De esta manera, el microcontrolador recibe datos de la base de datos. Es importante destacar que no se incluye una clase para actualizar la presión puesto que en la tabla correspondiente a la presión se almacenan rangos de presión que no queremos que sean modificados por el usuario. La razón de que se utilice una clase *wrapper* será explicada en el apartado 4.3.3.

Figura 27. Servidor web: ejemplo de clase wrapper para Arduino



Fuente: Elaboración propia

\manejadores\manejadorRutinas.php: pues en esta carpeta, se añaden las clases que contienen código para la creación de elementos dinámicamente en función de la base de datos. Como su objetivo principal es este y no gestionar la base de datos (creación o modificación de tablas), se tiene una clase independiente que interactúa con la clase dedicada a gestionar la base de datos (rutinaDB.php).

Así pues, vemos que, en la imagen 23, hay tres tarjetas rojas, las cuales se crean dinámicamente al cargar la página. Se crea una tarjeta por cada fila de la tabla de las rutinas. De modo que, si hubiera 8 entradas, se crearían 8 tarjetas.

4.3.3. Comunicación del microcontrolador y el servidor

Como hemos visto en el apartado 4.3.1., el programa de Arduino ejecuta peticiones HTTP especificando la ruta del servidor y pasando unos parámetros para el método HTTP. Así pues, las respuestas HTTP del servidor devolverán un String que Arduino procesará.

Otro aspecto importante es que no se pueden invocar funciones php desde las peticiones HTTP o desde las URL a través del navegador. Por ejemplo, el archivo rutinasDB.php contiene la función obternerRutinasHabilitadas(), pero no puede ser llamada de esta manera: http://localhost/includes/database/rutinasDB.php/obtenerRutinasHabilitadas, si no que ha de invocarse como en la fig. 27 con un wrapper, devolviendo un String mediante el método “echo”.

No es la manera más segura ni óptima de obtener datos de la base de datos, pero se considerarán mejoras para futuros trabajos.

5. Conclusiones

Los objetivos principales de este proyecto se basaban en la automatización de algunas de las tareas repetitivas susceptibles de ser programadas de un sistema de mantenimiento de piscina y un sistema de riego. Merece la pena subrayar que los objetivos especificados en la introducción se ordenan según la relevancia para el proyecto.
Entre estos objetivos encontrábamos la programación del horario de la filtración del agua y del horario de riego, pues han sido alcanzados con éxito. Como solución, se ha proporcionado un diseño de administrador de rutinas en el servidor que permite añadir y gestionar la programación de horarios. Al haberse implementado un diseño general de este administrador, se posibilita añadir la programación del horario no solo de estas dos tareas, sino de cualquier tarea que sea compatible con la programación de una rutina.
En cuanto a los objetivos que involucran sensores, se ha logrado la implementación de la obtención de la temperatura y del nivel de llenado del agua en tiempo real, así como su gestión. No obstante, la medición de la presión no ha sido posible debido a la imposibilidad de obtener un sensor de presión diferencial. De todas maneras, se ha proporcionado el código correspondiente a esta parte con el fin de su implementación en otros trabajos futuros que consulten esta memoria.
En relación con la creación de un servidor web simple para el monitoreo de parámetros en tiempo real y control de variables, se ha conseguido superando las expectativas del autor. Este proporciona un apartado para gestionar cada una de las otras metas y tiene potencial para ser ampliado con futuras mejoras, puesto que suministra una estructura base y conjunto de clases que facilita la implementación de extensiones.

Por otra parte, la realización del proyecto ha causado un gran interés por llevar a cabo el proyecto en un escenario real. Nos hemos visto limitados a la hora de realizar la fase de pruebas con simuladores, debido a que la interacción con los componentes no es la misma y se deja una gran parte del sistema por probar. De cara a futuros estudios, sería conveniente analizar este fenómeno y aplicar el prototipo sobre un diseño de componentes reales, concediendo un alcance del proyecto más extenso con resultados experimentales auténticos.

En el desarrollo del proyecto nos hemos visto limitados por un problema bastante común a la hora de descubrir el funcionamiento de los sensores y la placa microcontroladora. Muchas tiendas online no contemplan añadir la ficha técnica de especificaciones del producto. Si esto fuera poco, de cada modelo de sensor o placa hay diversos fabricantes con sus variantes del producto, por lo que cada variante puede que cambie la forma en que se conectan los pines, se lleven a cabo tareas relacionadas con el producto o de su funcionamiento general. Asimismo, configurar correctamente la placa Wemos D1 en el entorno de programación de Arduino nos llevó bastante tiempo, pues hay varias versiones de esta placa y no obtuvimos nunca una ficha técnica que coincidiera con el esquema de pines. A causa de esto, se recomienda a futuros profesionales o estudiosos del campo en cuestión la adquisición de material que asocie la ficha técnica correspondiente para su comprensión más fácil y rápidamente.

Todo aquello relacionado con Arduino como el montaje del sistema electrónico o la programación del microcontrolador, se ha tenido que aprender desde cero y ha llevado

muchas horas de aprendizaje, pero la disposición de una gran comunidad y cantidad de documentación online sobre Arduino ha sido un punto a favor que ha facilitado el aprendizaje. Así pues, nos encontramos con algunas dificultades como el deterioro de material debido a diseños de conexionado erróneos. Pero al final se ha podido construir el prototipo con triunfo y ampliado el conocimiento sobre electrónica aplicada. Por otra parte, se tenía una base sobre el uso de las tecnologías empleadas para el servidor, pero la implementación del servidor extendía notablemente dicha base que desembocó en continuar aprendiendo el cien por cien del tiempo sobre estas herramientas.

Un aspecto que destacar es la sustitución del microcontrolador Arduino Mega 2560 planteado en un principio, del que se disponía y presentaba características óptimas para la realización del proyecto, por el ESP8266 que presenta mejores prestaciones por un precio seis veces menor y ha posibilitado más facilidades para la ejecución del diseño.

Este trabajo contribuye con una serie de aportaciones y utilidades al diseño de sistemas de piscina y riego en la medida en que apuesta por incluir mejoras que apoyan los objetivos de desarrollo sostenible (ODS) impulsados por las Naciones Unidas. El modelo planteado en este trabajo supone la obtención de un diseño competitivo y asequible para el bolsillo de un público mucho más amplio que el de los proyectos comercializados ya existentes en el mercado actual. Además de la reducción y ahorro del uso de productos químicos gracias al diseño de un sistema eficiente que contribuye a disminuir los contaminantes en piscinas.

A modo de conclusión, se ha satisfecho el conjunto de objetivos. Y la realización del proyecto se ha considerado una experiencia muy enriquecedora a nivel personal que ha permitido el crecimiento de los conocimientos sobre una gran gama de tecnologías nuevas para el autor. También, se ha profundizado en muchas herramientas de las que tan solo se tenía una idea básica, es decir, se han puesto en práctica y llevado más allá aquellos conocimientos adquiridos a lo largo del grado universitario. Igualmente, se ha aprendido a relacionarse con los buscadores de información académica y se ha adquirido la capacidad de crítica y selección de la información.

5.1. Relación del trabajo desarrollado con los estudios cursados

En este trabajo de fin de grado, se han puesto en práctica gran cantidad de conocimientos adquiridos en multitud de asignaturas a lo largo del grado universitario, además de haberse introducido en aspectos externos a cualquier conocimiento anteriormente adquirido.

La parte que compone el circuito electrónico, compuesto por procesador, entradas y salidas de datos, electrónica lógica, etc. se puede relacionar con muchas de las asignaturas introductorias del grado. Nótese su relación entre ellas:

- Fundamentos de Computadores
- Fundamentos físicos de la Informática
- Tecnología de Computadores

También relacionamos esta parte con la asignatura de Estructura de Computadores, ya que esta aborda la unidad de entrada/salida y los dispositivos periféricos.
La rama de Ingeniería de computadores cursada por el autor presenta un gran vínculo a esta parte del sistema electrónico con las siguientes asignaturas:

- Control por Computador: se divide en control (señales y sistemas discretos) y tecnología de automatización.
- Sistemas Empotrados y de Tiempo Real: dentro de sus aspectos abordados, las correlaciones más útiles han sido la programación secuencial, la programación de tiempo real y los sistemas empotrados.
- Diseño de Sistemas Digitales: plantea aspectos sobre los lenguajes de descripción *hardware* permitiendo la programación de sistemas digitales. Además, en la parte práctica se reprogramaban dispositivos lógicos programables (circuitos ya diseñados) denominados FPGAs.

En cuanto a la programación del microcontrolador: se ha estado programando en multitud de asignaturas en diversos lenguajes. La capacidad que se adquiere para programar con un lenguaje de programación específico ayuda al aprendizaje más rápido de otros lenguajes. Además, Arduino es un lenguaje basado en C/C++, el cual hemos estado constantemente utilizándolo en muchas asignaturas de programación.

Por otra parte, la asignatura de Bases de Datos y Sistemas de Información nos ha facilitado conocimientos para el desarrollo de la base de datos y su gestión mediante sentencias (*queries*). Por último, el desarrollo web ha tenido una gran influencia por las asignaturas *Web Design* y *Web Development* cursadas en 4ºB en *University College Dublin*. Estas involucraban tanto XAMPP como el *software* que contiene, que nos ha llevado a elegir esta herramienta frente a otras alternativas. Además, en estas asignaturas también se aprendió sobre lenguajes de programación web.
Las asignaturas con una parte de computación en la nube han ayudado a abordar aspectos relacionados con el Internet de las cosas y el diseño del servidor: Lenguajes y Entornos de Programación Paralela y, Tecnologías de Sistemas de Información en la Red. Las asignaturas de redes y algunas de sistemas distribuidos que tocaban aspectos sobre HTTP, servidores y modelos cliente-servidor han sido de gran ayuda (Configuración, Administración y Gestión de Redes; Tecnología de Redes; y, Redes de Computadores). Y, por último, Interfaces Persona Computador proporciona conocimientos a cómo abordar la presentación visual e interacción con el cliente de la página web.

6. Trabajos futuros

De cara a futuros trabajos, se sugiere automatizar más tareas susceptibles de ser programadas. Se proponen algunas a continuación:

- Medición de la acidez del agua con una probeta sensor de pH. A raíz de un rango determinado, accionar el dosificador regulador de pH.
- Medición de la humedad del terreno ajardinado con un sensor de humedad. Activar el riego fuera del horario programado si el terreno se encuentra muy seco o desactivar la rutina del riego si el terreno es demasiado húmedo.
- Medición de las presiones de la instalación de riego.
- Medir la concentración de desinfectante en el agua. Desactivar la dosificación si la concentración supera unos valores.
- Para las piscinas con s*kimmer*, habilitar un sistema de alertas ante necesidad de limpieza.
- Automatización domótica de la iluminación artificial de la piscina.
- Sistema de videovigilancia para piscinas públicas.
- Automatización del sistema de ventilación y climatización de las piscinas cubiertas.

La ampliación y mejora de las funcionalidades y la presentación del servidor web daría calidad al proyecto. Entre ellas, sería conveniente mejorar la manera en qué el servidor y Arduino se comunican: el protocolo HTTP es adecuado, pero, en nuestro caso, Arduino funciona como cliente y pide o modifica información de la base de datos mediante las clases *wrappers* explicadas en la memoria. De manera que otro cliente podría hacer estas peticiones HTTP al servidor y gestionar la información también. Una forma más segura sería que el servidor enviara actualizaciones a Arduino automáticamente sin la necesidad de estas clases.

Por último, se propone realizar una fase experimental y de pruebas del prototipo en un escenario real con los accionadores de una piscina y sistema de riego para obtener resultados más auténticos.

7. Referencias

[1] GAMEZ, M.J., [sin fecha]. Objetivos y metas de desarrollo sostenible. *Desarrollo Sostenible* [en línea]. [Consulta: 30 agosto 2021]. Disponible en: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[2] IBARRA-ESQUER, J.E., GONZÁLEZ-NAVARRO, F.F., FLORES-RIOS, B.L., BURTSEVA, L. y ASTORGA-VARGAS, M.A., 2017. Tracking the Evolution of the Internet of Things Concept Across Different Application Domains. *Sensors*, vol. 17, no. 6, pp. 1379. DOI 10.3390/s17061379.

[3] What is Arduino? [en línea], [sin fecha]. [Consulta: 30 agosto 2021]. Disponible en: https://www.arduino.cc/en/Guide/Introduction.

[4] Boards — ESP8266 Arduino Core 3.0.2-10-g211606fe documentation. [en línea], [sin fecha]. [Consulta: 31 agosto 2021]. Disponible en: https://arduino-esp8266.readthedocs.io/en/latest/boards.html#.

[5] Comparativa ESP8266 frente a ESP32, los SoC de Espressif para IoT. *Luis Llamas* [en línea], [sin fecha]. [Consulta: 31 agosto 2021]. Disponible en: https://www.luisllamas.es/comparativa-esp8266-esp32/.

[6] BITWISE AR, 2017. *Arduino desde cero en Español - Capítulo 10 - Módulo LCD 1602A con librería LiquidCrystal* [en línea]. [Consulta: 31 agosto 2021]. Disponible en: https://www.youtube.com/watch?v=JEZiHQY-JPI.

[7] FERNÁNDEZ, Y., 2020. Qué es Arduino, cómo funciona y qué puedes hacer con uno. *Xataka* [en línea]. [Consulta: 1 septiembre 2021]. Disponible en: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno.

[8] Welcome! - The Apache HTTP Server Project. [en línea], [sin fecha]. [Consulta: 5 septiembre 2021]. Disponible en: http://httpd.apache.org/.

[9] MySQL. En: Page Version ID: 138001774, *Wikipedia, la enciclopedia libre* [en línea], 2021. [Consulta: 5 septiembre 2021]. Disponible en: https://es.wikipedia.org/w/index.php?title=MySQL&oldid=138001774.

[10] CONTRIBUTORS, phpMyAdmin, [sin fecha]. phpMyAdmin. *phpMyAdmin* [en línea]. [Consulta: 6 septiembre 2021]. Disponible en: https://www.phpmyadmin.net/.

[11]VERY ACADEMY, 2020. *XAMPP Web Server Network Access - Connect to Apache Server From the Network* [en línea]. [Consulta: 6 septiembre 2021]. Disponible en: https://www.youtube.com/watch?v=I_cVBp3gX14.

[12] About the XAMPP project. [en línea], [sin fecha]. [Consulta: 6 septiembre 2021]. Disponible en: https://www.apachefriends.org/es/about.html.

[13] PHP: ¿Qué es PHP? - Manual. [en línea], [sin fecha]. [Consulta: 6 septiembre 2021]. Disponible en: https://www.php.net/manual/es/intro-whatis.php.

[14] AJAX. En: Page Version ID: 137525491, *Wikipedia, la enciclopedia libre* [en línea], 2021. [Consulta: 6 septiembre 2021]. Disponible en: https://es.wikipedia.org/w/index.php?title=AJAX&oldid=137525491.

[15] Get Started with Visual Studio Code. [en línea], [sin fecha]. [Consulta: 6 septiembre 2021]. Disponible en: https://code.visualstudio.com/learn/overview.

[16] Guía para configurar un ESP-01, el módulo WiFi basado en ESP8266. [en línea], 2017. [Consulta: 6 septiembre 2021]. Disponible en: https://programarfacil.com/podcast/como-configurar-esp01-wifi-esp8266/.

[17] BITWISE AR, 2018. *Arduino desde cero en Español - Capítulo 38 - Reloj de Tiempo Real (RTC) DS3231 bus I2C* [en línea]. [Consulta: 6 septiembre 2021]. Disponible en: https://www.youtube.com/watch?v=ZOMXEYuQwwY.

[18] PROGRAMAR FACIL, 2017. *DS18B20 y Arduino cómo medir la temperatura del agua y ambientes húmedos* [en línea]. [Consulta: 6 septiembre 2021]. Disponible en: https://www.youtube.com/watch?v=Yq1J-exPp1I.

[19] *A700000007238410.pdf* [en línea], [sin fecha]. S.l.: s.n. [Consulta: 6 septiembre 2021]. Disponible en: https://docs.rs-online.com/8ff0/A700000007238410.pdf.

[20] *0900766b80ef4270.pdf* [en línea], [sin fecha]. S.l.: s.n. [Consulta: 6 septiembre 2021]. Disponible en: https://docs.rs-online.com/c464/0900766b80ef4270.pdf.

[21] Diodo - Wikipedia, la enciclopedia libre. [en línea], [sin fecha]. [Consulta: 6 septiembre 2021]. Disponible en: https://es.wikipedia.org/wiki/Diodo.

[22] Domótica. En: Page Version ID: 138101638, *Wikipedia, la enciclopedia libre* [en línea], 2021. [Consulta: 6 septiembre 2021]. Disponible en: https://es.wikipedia.org/w/index.php?title=Dom%C3%B3tica&oldid=138101638.

[23] Hardware libre. En: Page Version ID: 135768249, *Wikipedia, la enciclopedia libre* [en línea], 2021. [Consulta: 6 septiembre 2021]. Disponible en: https://es.wikipedia.org/w/index.php?title=Hardware_libre&oldid=135768249.

[24] Software libre. En: Page Version ID: 137213362, *Wikipedia, la enciclopedia libre* [en línea], 2021. [Consulta: 6 septiembre 2021]. Disponible en: https://es.wikipedia.org/w/index.php?title=Software_libre&oldid=137213362.

[25] Zona Maker - Resistencias de Pull-Up y Pull-Down. [en línea], [sin fecha]. [Consulta: 6 septiembre 2021]. Disponible en: https://www.zonamaker.com/electronica/intro-electronica/teoria/resistencias-de-pull-up-y-pull-down.

Anexos

En este capítulo, se proporciona el código que se cree más relevante para el entendimiento de las clases.

ANEXO 1: Arduino.ino

Se proporciona el archivo .ino con el contenido del programa Arduino que el microcontrolador ESP8266 ejecuta:

```
#include <Arduino.h>            //Librería para utilizar funciones
básicas de Arduino

                                /* Librerías para la conexión WiFi del
ESP8266 */
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

#include <RTClib.h>             //Librería para el Reloj de Tiempo
Real
#include <Wire.h>               //Librería para la comunicación I2C
del sensore de presión diferencial

#include <Alarma.h>             //Librería creada para los objetos
rutina (alarma)
#include <ListLib.h>            //Manejador de listas

#include <OneWire.h>            //Conexión 1-Wire (múltiples sensores
en mismo pin digital)
#include <DallasTemperature.h>  //Conversión de señal a temperatura
Celsius


/*-----------------------------------------------------------------------
------------------------------*/
/* DECLARACIÓN DE VARIABLES
/*-----------------------------------------------------------------------
------------------------------*/

/* WiFi */
ESP8266WiFiMulti WiFiMulti;
String serverURI = "http://192.168.1.54:80/";             //dirección
del servidor (mi IP local)

/* Accionadores (simulación con LEDs)s */
bool ledAzul = false, ledVerde = false, ledRojo = false;

/* Rutinas y rtc */
RTC_DS3231 rtc;                                           //Reloj de
tiempo real (para manejar las alarmas)
List <Alarma> alarmas;                                    //Conjunto
de rutinas (alarmas)
```

```
DateTime ahora;                                                    //DateTime
con los valores de fecha y tiempo de este instante
int hoy;                                                           //Índice con
el día 0:L a 6:D
String consultaAnterior, consulta;                                 //Almacena
la última consulta para ver si hay actualizaciones en las rutinas

/* Temperatura */
#define ONE_WIRE_BUS D12                                           //El sensor
transmitirá la señal por el pin D12
OneWire oneWire(ONE_WIRE_BUS);                                     //Habilita
la conexión 1wire por pin D12
DallasTemperature sensors(&oneWire);                               //Referencia
la conexión 1wire a la librería de temperatura
float temperatura;
int maxT;
int minT;

/* Presión */
double P;
double Vs=3.3;
//Alimentación
double Vout;                                                       //Tensión
recibida por el sensor
double unidad = Vs / 1024;                                         //Voltaje
por unidad recibida por el sensor (debido a la resolución)
double aux;                                                        //Se usará
para tomar las muestras y obtener la presión de ellas
double error = 0.0;                                                //Ajustar la
medida de presión


/*-------------------------------------------------------------------------
-----------------------------*/
/* INICIALIZACIÓN DEL PROGRAMA
/*-------------------------------------------------------------------------
-----------------------------*/
void setup() {

    Serial.begin(115200);

    /* WiFi */
    WiFi.mode(WIFI_STA);
    WiFiMulti.addAP("MASFIBRA-SG", "Max11Sanz15Gresa20*");
    delay(10000);

    /* Rutinas y Reloj de tiempo real */
    if(!rtc.begin()) {
        Serial.println("Fallo en la conexión con el RTC");
        Serial.flush();
        abort(); //Llama a _exit(1). Como no hay un SO al que volver,
esto es lo que realmente exit realiza: cli(); (disable interrupts)
while(1); (forever loop)
    }
    if(rtc.lostPower()) {
        //Ajustar el rtc a la fecha y hora actual
        rtc.adjust(DateTime(__DATE__, __TIME__));
    }
    rtc.disable32K(); //No se necesita el pin
```

```
    /* Temperatura */
    sensors.begin(); //Inicializa la librería

    /* Presión */
    pinMode(A0, INPUT);
}


/*--------------------------------------------------------------------
-------------------------------*/
/* PROGRAMA
/*--------------------------------------------------------------------
-------------------------------*/
void loop() {
  /* Actualización, obtención y comprobación de las rutinas*/
  comprobarAlarmas();

  /* Actualización de rango deseado y obtención de la temperatura */
  obtenerTemperatura();

  /* Comprobación del nivel de agua */
  detectarNivelAgua();

  /* Comprobación de la presión sobre los filtros */
  obtenerNivelPresion();
}



/*--------------------------------------------------------------------
-------------------------------*/
/* WIFI: FUNCIONES RELACIONADAS CON LA COMUNICACIÓN BASE DE DATOS -
ARDUINO. PETICIONES HTTP
/*--------------------------------------------------------------------
-------------------------------*/

/*
 * formato ruta: [a]/[b]/[c]/[...] (ejemplo:
includes/database/obtenerAlarmas.php)
 * formato parámetros: variable=valor[&variable2=valor2&[...]=[...]]
(ejemplo: color=rojo&forma=circulo
 */
String conectar(String ruta, String parametros) {
  String payload;
  if ((WiFiMulti.run() == WL_CONNECTED)) {
    WiFiClient client;
    HTTPClient http;

    if (http.begin(client, serverURI + ruta + "?" + parametros)) {
      int httpCode = http.GET();

      if (httpCode > 0) {
        Serial.printf("[HTTP] GET... code: %d\n", httpCode);

        if (httpCode == HTTP_CODE_OK || httpCode ==
HTTP_CODE_MOVED_PERMANENTLY) {
          payload = http.getString();
          Serial.println(payload);
```

```
        }
      } else {
        Serial.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
      }

      http.end();
    } else {
      Serial.printf("[HTTP} Unable to connect\n");
    }
  }
  return payload;
}



/*-----------------------------------------------------------------------
------------------------------*/
/* RUTINAS
/*-----------------------------------------------------------------------
------------------------------*/

/*
* Inicializa las alarmas habilitadas: comprueba que cada alarma en la
base de datos esté habilitada o no (variable booleana atributo de cada
alarma)
* Si está habilitada, añade la alarma a la lista de alarmas que se
comprueban mediante polling si han de sonar
* Si se añade una alarma nueva o se modifica la variable "habilitada"
a través del servidor. Se vuelve a llamar a inizializar alarmas
*/
void inicializarAlarmas() {
  consulta =
conectar("includes/database/arduino/actualizarRutinas.php", "");
//Obtengo las alarmas habilitadas en la BD
  //comparar si hay actualizaciones
  if(consulta == consultaAnterior) return;
  else {
    consultaAnterior = consulta;
    alarmas.Clear(); //Vaciamos el conjunto de alarmas
  }

  int num_alarmas = (consulta.length()/27), id, salida;
  DateTime inicio, final;
  String dias;
  Alarma actual;

  /* Añadir las rutinas procesando el String devuelto por la respuesta
HTTP del servidor */
  for (int i = 0; i < num_alarmas; i++){
    int i_id = i*27, i_inicio = i*27 + 2, i_final = i*27 + 10, i_dias
= i*27 + 18, i_salida = i*27+25; //Posición inicial de cada atributo
dentro de la consulta

    id = consulta.substring(i_id, i_id+2).toInt();
    inicio = DateTime(__DATE__, consulta.substring(i_inicio,
i_inicio+8).c_str());
    final = DateTime(__DATE__, consulta.substring(i_final,
i_final+8).c_str());
```

```
    dias = consulta.substring(i_dias, i_dias+7);
    salida = consulta.substring(i_salida, i_salida+2).toInt();
    char buf[9];

    actual = Alarma(id, inicio, final, dias, salida);
    alarmas.Add(actual);
  }
}


void comprobarAlarmas() {
  inicializarAlarmas(); //Comprobamos actualizaciones del servidor

  for (int i = 0; i < alarmas.Count(); i++)     {

    if(alarmas[i].getDiasSemana().charAt(hoy) == '1'){ //Si hoy (día
de la semana) ha de saltar la alarma actual
      bool alarmaEncendida = alarmas[i].getActiva(); //True->Relé ya
accionado; false-> Relé no accionado

      DateTime inicio = alarmas[i].getInicio(), final =
alarmas[i].getFinal();

      bool activa = ahora >= inicio && ahora <= final; //Alarma
sonando

      int salida = alarmas[i].getSalida();

      if(activa) {
          if(!alarmaEncendida) {
              digitalWrite(salida, LOW);
          }
      } else {
        if(alarmaEncendida) {
            digitalWrite(salida, HIGH);
        }
      }
      alarmas[i].setActiva(activa); //TODO: en la BD
    }
  }
}

/* Mantiene el rtc con el instante actual */
void reloj() {
    ahora = rtc.now();
}



/*-----------------------------------------------------------------------
------------------------------*/
/* TEMPERATURA
/*-----------------------------------------------------------------------
------------------------------*/
/* Obtener la temperatura del sensor y enviarla a la base de datos */
void obtenerTemperatura() {
  obtenerRangoDeseado();
  sensors.requestTemperatures(); //Recibir valores de todos los
sensores conectados al pin 1wire
```

```
  temperatura = sensors.getTempCByIndex(0);
  conectar("includes/database/arduino/actualizarTemperatura.php",
"temperatura=" + String(temperatura)); //actualiza el valor de la
temperatura en la base de datos
}

/* Actualiza el rango deseado obtenido desde la base de datos */
void obtenerRangoDeseado() {
  String rango =
conectar("includes/database/arduino/actualizarTemperatura.php", "");
  maxT = rango.substring(0, 2).toInt();
  minT = rango.substring(2, 4).toInt();
}

/* Control del accionador */
void accionarCalentador(){
  if(temperatura < minT) {
    //Encender calentador
    if(ledRojo == false){
      digitalWrite(D8, HIGH);
      ledRojo = true;
    }
  } else {
    //Apagar calentador
    if(ledRojo == true) {
      digitalWrite(D8, LOW);

      //Avisar de que se supera el máximo llamando a la clase
correspondiente del servidor
      if(temperatura > maxT){

conectar("includes/database/arduino/actualizarTemperatura.php",
"maximoSuperado=true");
      }

      ledRojo = false;
    }
  }
}

/*-------------------------------------------------------------------
-------------------------------*/
/* DETECCIÓN DE NIVEL DE AGUA
/*-------------------------------------------------------------------
-------------------------------*/

/*
 * Si se recibe un '0' del sensor el agua no llega al nivel deseado:
encender la bomba de agua para llenar
 * Si se recibe un '1' del sensor el agua está tocando el sensor:
mantener la bomba de agua apagada
 */
void detectarNivelAgua(){
  if(digitalRead(D1) == 0) {
    //Accionar bomba de agua
    if(ledAzul == false) {
      digitalWrite(D7, HIGH);
      ledAzul = true;
    }
```

```
  } else {
    //Apagar bomba de agua
    if(ledAzul) {
      digitalWrite(D7, LOW);
      ledAzul = false;
    }
  }
}


/*-----------------------------------------------------------------------
------------------------------*/
/* NIVEL DE PRESIÓN
/*-----------------------------------------------------------------------
------------------------------*/
void obtenerNivelPresion(){
  //Leer el voltaje del Sensor 10 veces (10 muestras) y sacar el
promedio
   aux=0;
   for(int i=0;i<10;i++){
    aux = aux + (float(analogRead(A0))*unidad); //el pin análogo
proporciona la medida en bytes. Convertimos a voltios al multiplicar
por unidad
    delay(10);
   }
   Vout=aux/10.0;

  //Presión en Kpa según la fórmula proporcionada en la ficha técnica
  P = ( Vout - 0.04*Vs - error ) / (0.0018 * Vs); //kPa

  //Ver si la presión está en un rango óptimo y actuar en consecuencia
  valorarPresion();
}

void valorarPresion(){
  //si el valor es superior a 2 bares, apagar la bomba de agua para
evitar daños
  if(P > 2) {
    //bomba todavía encendida
    if(ledVerde == true) {
      digitalWrite(D9, LOW);
      ledVerde = false;
    }
  }
}
```

ANEXO 2: XAMPP\rutinas.php

Contiene el código de la página web principal “rutinas” correspondiente a la vista en la figura 23.

```
<?php
error_reporting(E_ERROR);
include('./includes/head.php');
include('./includes/header.php');
include('./includes/sidebar.php');
include('./includes/manejadores/manejadorRutinas.php');
?>

<!-- Content Wrapper. Contains page content -->
<div class="content-wrapper">
    <!-- Content Header (Page header) -->
    <div class="content-header">
      <div class="container-fluid">
        <div class="row mb-2">
          <div class="col-sm-6">
            <h1 class="m-0">Rutinas</h1>
          </div><!-- /.col -->
          <div class="col-sm-6">
            <ol class="breadcrumb float-sm-right">
              <li class="breadcrumb-item"><a
href="./index.php">Inicio</a></li>
              <li class="breadcrumb-item active">Rutinas</li>
            </ol>
          </div><!-- /.col -->
        </div><!-- /.row -->
      </div><!-- /.container-fluid -->
    </div>
    <!-- /.content-header -->

    <!-- Añadir alarma widget -->
    <div class="row">
      <div class="col-md-12">
        <div class="card card-primary collapsed-card">
          <div class="card-header">
            <h3 class="card-title" id="tituloAñadirModificar">Añadir
rutina nueva</h3>

            <div class="card-tools">
              <button type="button" class="btn btn-tool" data-card-
widget="collapse"><i class="fas fa-plus"></i>
              </button>
            </div>
            <!-- /.card-tools -->
          </div>
          <!-- /.card-header -->
          <div class="card-body">
            <?php include('./includes/pages/añadirRutina.html') ?>
          </div>
          <!-- /.card-body -->
        </div>
        <!-- /.card -->
      </div>
      <!-- /.col -->
```

```
    </div>
    <!-- /.row -->
    <!-- /.Añadir alarma widget -->

    <!-- Mostrar todas las alarmas widgets -->
    <?php echo mostrarTarjetasAlarmas(); ?>
    <!-- /.Mostrar todas las alarmas widgets -->

  </div>
  <!-- /.content-wrapper -->


<?php
include('./includes/footer.php');
include('./includes/scripts.php');
?>
```

ANEXO 3: XAMPP\includes\database\rutinasDB.php

Se pretende proporcionar un mayor entendimiento del código que hace interactuar el servidor con la base de datos. Nótese que temperaturaDB.php y presionDB.php contienen código con el mismo propósito.

```
<?php
/* @Author Borja Sanz Gresa
*  @Date 09/08/2021
*  @Description Manejador de las alarmas: Funciones que conectan la
base de datos con el servidor
*/

error_reporting(E_ERROR);

/* function conectar() {
    $mysqli = new mysqli("127.0.0.1", "root", "", "piscina");
    if ($mysqli->connect_errno) {
        echo "Fallo al conectar a MySQL: (" . $mysqli->connect_errno .
") " . $mysqli->connect_error;
    }
    mysqli_query($mysqli, "SET NAMES 'utf8'");
    return $mysqli;
} */

include('./conexion.php');

/* Ejecutar al recibir el formulario */
if (isset($_POST['añadirAlarma']) || isset($_POST['modificarAlarma']))
{
    manejarAlarma();
    header("Location: http://192.168.1.54:80/alarmas.php");
    exit();
}


/*
* Añade una alarma a la base de datos o la modifica si ya existe
*/
function manejarAlarma() {
    $conexion = conectar();

    if(!empty($_POST['nombre']) && !empty($_POST['inicio']) &&
!empty($_POST['final']) && !empty($_POST['salida'])) {
        $nombre = $_POST['nombre'];
        $inicio = $_POST['inicio'];
        $final =  $_POST['final'];
        $dias = "";
        $salida =  $_POST['salida'];

        //Obtener valores de los días de la semana en qué la alarma
está programada
        $claves =
["lunes","martes","miercoles","jueves","viernes","sabado","domingo"];
        foreach ($claves as $clave => $value) {
            if($_POST[$value] == "on") $dias .= "1";
            else $dias .= "0";
        }
```

```
        //Añadir alarma nueva
        if(isset($_POST['añadirAlarma'])) {
            //id = NULL, porque la BD está configurada con auto
incrementar el valor
            $consulta = "INSERT INTO alarma
(id,nombre,inicio,final,activa,habilitada,dias) VALUES (NULL,
'$nombre', '$inicio', '$final', '0', '1', '$dias', '$salida');";
            $run = mysqli_query($conexion,$consulta);
        }


        //Modificar alarma nueva
        $id_alarma = $_POST['id_alarma'];
        if(isset($_POST['modificarAlarma']) && isset($id_alarma)) {
            setNombre($id_alarma, $nombre);
            setInicio($id_alarma, $inicio);
            setFinal($id_alarma, $final);
            setDias($id_alarma, $dias);
            setSalida($id_alarma, $salida);
        }

    } else echo "TODOS LOS CAMPOS SON NECESARIOS";


    mysqli_close($conexion);
}



/*
* Devuelve el conjunto de alarmas habilitadas (atributo "habilitada" =
1)
*/
function obtenerRutinasHabilitadas() {
    $conexion = conectar();
    $resultado = "";
    /* Obtener alarmas habilitadas mediante consulta */
    if ($alarmas = mysqli_query($conexion, "SELECT `id`, `inicio`,
`final`, `dias`, `salida` FROM `alarma` WHERE `habilitada` = 1;")) {
        /* obtener el array asociativo */
        while ($columna = mysqli_fetch_assoc($alarmas)) {
            $id_columna = $columna['id'];
            if (strlen($id_columna) == 1) $id_columna =
'0'.$id_columna;
            $salida_columna = $columna['salida'];
            if (strlen($salida_columna) == 1) $salida_columna =
'0'.$salida_columna;

            $resultado .=
$id_columna.$columna['inicio'].$columna['final'].$columna['dias'].$sal
ida_columna;
        }
        /* liberar el conjunto de resultados */
        mysqli_free_result($alarmas);
    }
    mysqli_close($conexion);

    return $resultado;
```

```
}


/*
* Devuelve todo el conjunto de alarmas
*/
function obternerAlarmas() {
    $conexion = conectar();
    $alarmas = array(); //Array multidimensional

    if ($consulta = mysqli_query($conexion, "SELECT `id`, `nombre`,
`inicio`, `final`, `activa`, `habilitada`, `dias`, `salida` FROM
`alarma` WHERE 1;")) {
        /* obtener el array asociativo */
        while ($columna = mysqli_fetch_assoc($consulta)) {
            $actual = array(
                "id" => $columna['id'],
                "nombre" => $columna['nombre'],
                "inicio" =>  $columna['inicio'],
                "final" => $columna['final'],
                "activa" => $columna['activa'],
                "habilitada" => $columna['habilitada'],
                "dias" => $columna['dias'],
                "salida" => $columna['salida'],
            );
            array_push($alarmas, $actual);
        }
        /* liberar el conjunto de resultados */
        mysqli_free_result($consulta);
    }
    mysqli_close($conexion);

    return $alarmas;
}


/*
* Devuelve el nombre de la alarma con @param $id
*/
function getNombre($id) {
    if(!isset($id)) return;

    $conexion = conectar();

    $consulta = "SELECT `nombre` FROM `alarma` WHERE `id` = $id;";
    if ($resultado = mysqli_query($conexion, $consulta)) {
        $res = mysqli_num_rows($resultado);

        /* liberar el conjunto de resultados */
        mysqli_free_result($resultado);
    }

    mysqli_close($conexion);

    return $res;
}
/*
* Actualiza el nombre de la alarma con @param $id
*/
```

```php
function setNombre($id, $nombre) {
    if(!isset($nombre) || !is_string($nombre) || strlen($nombre) >
64){
        echo "<script type=text/javascript>alert('El nombre ha de ser
más corto que 64 caracteres');</script>";
        return;
    }

    $conexion = conectar();

    $consulta = "UPDATE `alarma` SET `nombre`='$nombre' WHERE
`id`=$id;";
    mysqli_query($conexion, $consulta);

    mysqli_close($conexion);
}


/*
* Devuelve el inicio de la alarma con @param $id
*/
function getInicio($id) {
    if(!isset($id)) return;

    $conexion = conectar();

    $consulta = "SELECT `inicio` FROM `alarma` WHERE `id` = $id;";
    if ($resultado = mysqli_query($conexion, $consulta)) {
        $res = mysqli_num_rows($resultado);

        /* liberar el conjunto de resultados */
        mysqli_free_result($resultado);
    }

    mysqli_close($conexion);

    return $res;
}
/*
* Actualiza cuándo empieza a sonar la alarma con @param $id
*/
function setInicio($id, $inicio) {
    if(!isset($inicio) || !preg_match("/^(?:2[0-3]|[01][0-9]):[0-5][0-
9]$/", $inicio)){
        echo "<script type=text/javascript>alert('No ha introducido un
formato de hora correcto');</script>";
        return;
    }

    $conexion = conectar();

    $consulta = "UPDATE `alarma` SET `inicio`='$inicio' WHERE
`id`=$id;";
    mysqli_query($conexion, $consulta);

    mysqli_close($conexion);
}
```

```
/*
* Devuelve el final de la alarma con @param $id
*/
function getFinal($id) {
    if(!isset($id)) return;

    $conexion = conectar();

    $consulta = "SELECT `final` FROM `alarma` WHERE `id` = $id;";
    if ($resultado = mysqli_query($conexion, $consulta)) {
        $res = mysqli_num_rows($resultado);

        /* liberar el conjunto de resultados */
        mysqli_free_result($resultado);
    }

    mysqli_close($conexion);

    return $res;
}
/*
* Actualiza cuándo para de sonar la alarma con @param $id
*/
function setFinal($id, $final) {
    if(!isset($final) || !preg_match("/^(?:2[0-3]|[01][0-9]):[0-5][0-
9]$/", $final)){
        echo "<script type=text/javascript>alert('No ha introducido un
formato de hora correcto');</script>";
        return;
    }

    $conexion = conectar();

    $consulta = "UPDATE `alarma` SET `final`='$final' WHERE
`id`=$id;";
    mysqli_query($conexion, $consulta);

    mysqli_close($conexion);
}


/*
* Devuelve el estado de la alarma con @param $id: si está sonando o no
*/
function getActiva($id) {
    if(!isset($id)) return;

    $conexion = conectar();

    $consulta = "SELECT `activa` FROM `alarma` WHERE `id` = $id;";
    if ($resultado = mysqli_query($conexion, $consulta)) {
        $res = mysqli_num_rows($resultado);

        /* liberar el conjunto de resultados */
        mysqli_free_result($resultado);
    }

    mysqli_close($conexion);
```

```
    return $res;
}
/*
* Actualiza el estado de la alarma con @param $id: si está sonando o
no
*/
function setActiva($id, $activa) {
    if (!isset($activa)) return;

    $conexion = conectar();

    $consulta = "UPDATE `alarma` SET `activa`='$activa' WHERE
`id`=$id;";
    mysqli_query($conexion, $consulta);

    mysqli_close($conexion);
}


/*
* Devuelve si la alarma con @param $id está habilitada o deshabilitada
*/
function getHabilitada($id) {
    if(!isset($id)) return;

    $conexion = conectar();

    $consulta = "SELECT `habilitada` FROM `alarma` WHERE `id` = $id;";
    if ($resultado = mysqli_query($conexion, $consulta)) {
        $res = mysqli_num_rows($resultado);

        /* liberar el conjunto de resultados */
        mysqli_free_result($resultado);
    }

    mysqli_close($conexion);

    return $res;
}
/*
* Habilita o deshabilita la alarma con @param $id
*/
function setHabilitada($id, $habilitada) {
    if (!isset($habilitada) || !isset($id)) {
        echo "<script type=text/javascript>alert('No se pudo cambiar
el estado de la alarma');</script>";
        return;
    }

    //Actualizar estado del atributo "activa"
    if ($habilitada == 0) {
        setActiva($id, 0);
    }

    $conexion = conectar();

    $consulta = "UPDATE `alarma` SET `habilitada`='$habilitada' WHERE
`id`=$id;";
    mysqli_query($conexion, $consulta);
```

```
    mysqli_close($conexion);

}
/* Ejecutar POST request */
if(isset($_REQUEST['idAlarma'])){
    //setHabilitada($_POST['idAlarma'], $_POST['setHabilitada']);
    echo "<script type=text/javascript>alert('FUNCIONA');</script>";
}

/*
* Devuelve los días en qué la alarma con @param $id sonará
*/
function getDias($id) {
    if(!isset($id)) return;

    $conexion = conectar();

    $consulta = "SELECT `dias` FROM `alarma` WHERE `id` = $id;";
    if ($resultado = mysqli_query($conexion, $consulta)) {
        $res = mysqli_num_rows($resultado);

        /* liberar el conjunto de resultados */
        mysqli_free_result($resultado);
    }

    mysqli_close($conexion);

    return $res;
}
/*
* Actualiza los dias en que sonará la alarma con @param $id
*/
function setDias($id, $dias) {
    if(!isset($dias) || !is_string($dias) || !preg_match("/[0-1][0-
1][0-1][0-1][0-1][0-1][0-1]/", $dias)){
        return;
    }

    $conexion = conectar();

    $consulta = "UPDATE `alarma` SET `dias`='$dias' WHERE `id`=$id;";
    mysqli_query($conexion, $consulta);

    mysqli_close($conexion);
}

/*
* Devuelve la salida que la alarma con @param $id activará
*/
function getSalida($id) {
    if(!isset($id)) return;

    $conexion = conectar();

    $consulta = "SELECT `salida` FROM `alarma` WHERE `id` = $id;";
    if ($resultado = mysqli_query($conexion, $consulta)) {
        $res = mysqli_num_rows($resultado);
```

```
        /* liberar el conjunto de resultados */
        mysqli_free_result($resultado);
    }

    mysqli_close($conexion);

    return $res;
}
/*
* Actualiza la salida que la alarma con @param $id activará
*/
function setSalida($id, $salida) {
    if(!isset($salida)){
        return;
    }

    $conexion = conectar();

    $consulta = "UPDATE `alarma` SET `salida`='$salida' WHERE
`id`=$id;";
    mysqli_query($conexion, $consulta);

    mysqli_close($conexion);
}
?>
```

Glosarios

Domótica: “Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado” [22].

***Hardware* libre o de código abierto:** “son aquellos dispositivos de *hardware* cuyas especificaciones y diagramas esquemáticos son de acceso público, ya sea bajo algún tipo de pago, o de forma gratuita.” [23]

***Software* libre o de código abierto: “**es un software cuyo código fuente puede ser estudiado, modificado, y utilizado libremente con cualquier finalidad y redistribuido con cambios o mejoras sobre ellas.” [24]

VCC, VDD, 5V, 3.3V, +: son formas de especificar el voltaje de alimentación digital de un circuito, tensión positiva o nivel alto.

Tierra, *ground,* GND, 0V, -: son formas de especificar un nivel bajo o 0 voltios.

Resistencia de *pull-up* / *pull-down*: “Las resistencias de *Pull-Up y Pull-Down* tienen la función de definir un nivel de tensión concreto en cada instante.” [25]. Independientemente de las perturbaciones o interferencias en la salida del circuito obtendremos el nivel de tensión deseado. Las resistencias de *Pull-Up* definirán un nivel alto y las resistencias de *Pull-Down* un nivel bajo.