



Sistema de mantenimiento preventivo y control de fallos en electroválvulas

Manuel Patiño Villalón

Grado en Ingeniería Informática

TFG – Arduino

Antoni Morell Pérez

Pere Tuset Peiró

07/01/2020



Esta obra está sujeta a una licencia de [Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Sistema de mantenimiento preventivo y control de fallos en electroválvulas</i>
Nombre del autor:	<i>Manuel Patiño Villalón</i>
Nombre del consultor/a:	<i>Antoni Morell Pérez</i>
Nombre del PRA:	<i>Pere Tuset Peiró</i>
Fecha de entrega (mm/aaaa):	<i>01/2020</i>
Titulación o programa:	<i>Grado en Ingeniería Informática</i>
Área del Trabajo Final:	<i>TFG – Arduino</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>Arduino, electroválvula, mantenimiento</i>
Resumen del Trabajo:	
<p>Este proyecto consiste en un sistema de ayuda al mantenimiento de electroválvulas de forma preventiva y predictiva. Se creará mediante <i>open hardware</i> basado en la plataforma Arduino junto con un servidor Web implementado en código abierto, con Apache y MySQL.</p> <p>En un entorno industrial, donde habitualmente se realizan programas de mantenimiento de los equipos en base a métricas de tiempo desde el momento de la instalación de cada componente, se hace necesario un sistema que pudiera medir el tiempo de uso real de una electroválvula y una detección anticipada de problemas que podrían dar lugar a grandes pérdidas para la empresa.</p> <p>Este sistema recoge el número de ciclos que hace una electroválvula a lo largo de su vida y lo envía a un servidor web a través de una red inalámbrica donde se almacena y compara con los datos recomendados por el fabricante.</p> <p>Además, junto con el número de ciclos, se envía también información del caudal que existe en la electroválvula mediante el uso de un caudalímetro, monitorizando de esta manera si existe un problema de fugas o de ausencia de caudal.</p> <p>El resultado es un sistema escalable y económico, que facilita realizar un mantenimiento de las electroválvulas programado y no de forma reactiva, anticipándose a los posibles problemas mediante la detección de fallos cuando aún no son graves.</p> <p>El proyecto tiene posibilidades de ampliación en un futuro para monitorizar más elementos, siempre que sea posible medir el número de pulsos o maniobras.</p>	

Abstract:

This project consists on a helping system for solenoid valves maintenance in a preventive and predictive way. It will be created using open hardware based on the Arduino platform and a Web server implemented in open source software, with Apache and MySQL.

In an industrial environment, where maintenance programs for the equipment are usually carried out based on time metrics from the moment of the installation of each component, it becomes necessary a system that could measure the real time that a solenoid valve is used and an early detection of problems that could lead to large losses for the company.

This system collects the number of cycles that a solenoid valve does throughout its life and sends it to a web server through a wireless network where it is stored and compared with the data recommended by the manufacturer.

In addition, together with the number of cycles, information on the flow that exists in the solenoid valve is also sent by means of the use of a flowmeter, monitoring in this way whether there is a problem of leaks or lack of flow.

The result is a scalable and economical system, which makes it easier to perform scheduled maintenance of the solenoid valves and not in a reactive way, anticipating possible problems by detecting failures when they are not yet serious.

The project has expansion possibilities in the future to monitor more elements, whenever it is possible to measure the number of pulses or cycles.

INDICE

1.	INTRODUCCIÓN	5
1.1.	Mantenimiento industrial: Correctivo, preventivo y predictivo	5
1.2.	Arduino: ¿Por qué?	6
2.	DESCRIPCION	7
3.	OBJETIVOS	8
4.	VIABILIDAD	10
5.	PLANIFICACION	12
6.	HARDWARE.....	14
6.1.	Componentes electrónicos	14
6.2.	Esquema de conexión	20
6.3.	Detalle de conexión.....	21
6.4.	Firmware ESP8266	22
7.	SOFTWARE.....	24
7.1.	Entorno de desarrollo	24
7.2.	Estructura	26
7.3.	Arduino.....	28
7.3.1.	Configuración de red.....	28
7.3.2.	Lectura de valores.....	29
7.3.3.	Envío y recepción de lecturas al servidor WEB.....	30
7.4.	Base de datos MYSQL.....	33
7.5.	Comunicación entre Arduino y MYSQL	33
7.6.	Servidor WEB.....	35
7.6.1.	Dashboard.....	35
7.6.2.	Electrovalves	39
7.6.3.	Datalog.....	41
7.6.4.	Settings.....	43
8.	RESULTADOS.....	45
9.	CONCLUSIONES.....	46
9.1.	Trabajos futuros	46
10.	BIBLIOGRAFIA.....	47

INDICE DE FIGURAS

Figura 1 - Diagrama de Gantt	13
Figura 2 - Arduino UNO R3	14
Figura 3 - Módulo WIFI ESP8266 ESP-01S	15
Figura 4 - Pin-out ESP-01s.....	16
Figura 5 - Regulador de tensión.....	17
Figura 6 - Alimentador 9V.....	17
Figura 7 - Fuente de alimentación.....	18
Figura 8 - Electroválvula 12V DC.....	18
Figura 9 - Relé 1 canal 5V.....	19
Figura 10 - Caudalímetro	19
Figura 11 - Esquema de conexión.....	20
Figura 12 - Esquema de conexión ESP8266 modo UART.....	22
Figura 13 - ESP Flash download tool.....	23
Figura 14 - IDE Arduino.....	24
Figura 15 - Diagrama de flujo Arduino	26
Figura 16 - Diagrama intercambio de información	27
Figura 17 - Tabla valves	33
Figura 18 - Tabla state	33
Figura 19 – Dashboard.....	35
Figura 20 –Panel de mando de electroválvulas.....	39
Figura 21 - Datalog.....	41
Figura 22 - Panel de ajustes.....	43

1. INTRODUCCIÓN

1.1. Mantenimiento industrial: Correctivo, preventivo y predictivo

En una empresa de carácter industrial, tener un programa de mantenimiento es imprescindible si se quiere tener confiabilidad en el proceso productivo y calidad en el acabado final. Una avería imprevista puede causar paradas en producción, con la correspondiente pérdida que eso provoca, de tiempo y de dinero.

Para llevar a cabo este mantenimiento existen diferentes clasificaciones, siendo la más extendida la que se refiere a la naturaleza de las tareas. De esta manera pueden distinguirse las siguientes estrategias: mantenimiento correctivo, mantenimiento preventivo y mantenimiento predictivo.

- **Mantenimiento correctivo**
Es un tipo de mantenimiento que, como su nombre indica, corrige las deficiencias que se van presentando en los diferentes equipos. No requiere de ningún tipo de planificación, suele ser de tipo reactivo y se basa en arreglar las averías conforme van surgiendo.
Este tipo de mantenimiento supone que la empresa ha de tener efectivos de personal y recambios estocados para poder hacer frente a las averías, además, normalmente requiere paralizar la producción, con las pérdidas que ello genera.
- **Mantenimiento preventivo**
Es un mantenimiento enfocado a la prevención de fallos en los equipos, anticipándose a las averías. Se suele realizar en función de las horas de funcionamiento, por periodos de tiempos o por número de maniobras.
Este sistema permite planificar la intervención y se puede planificar una parada preventiva que afecte lo menos posible a la producción.
- **Mantenimiento predictivo**
Este tipo de mantenimiento consiste en analizar y medir el desgaste de los elementos para reemplazarlos en cuanto muestran síntomas que predicen el fallo, antes de que se llegue a materializar la avería.
Se trata de predecir cuándo empieza a fallar una determinada máquina o equipo mediante el análisis de diferentes variables.

[1]

1.2. [Arduino: ¿Por qué?](#)

Arduino es una plataforma electrónica de código abierto que está basada en una placa con un microcontrolador y un entorno de desarrollo.

Para la parte de hardware, éste dispone de una amplia variedad de placas y shields, éstas son placas compatibles que se pueden colocar en la parte superior permitiendo extender sus capacidades. Además, incorpora los elementos necesarios para conectar periféricos a las entradas y salidas del microcontrolador.

Al ser un hardware libre, sus especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos, es decir, que es posible crear una placa propia con base Arduino adaptada a unas necesidades concretas sin tener que pagar ningún tipo de licencia.

En lo que a software se refiere, consiste en un entorno de desarrollo (IDE) compuesto de un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Desde este IDE se pasa el programa ya compilado a la memoria flash del hardware mediante comunicación con el puerto serie.

Aunque existen muchos tipos de placas con microcontrolador incorporado, los principales motivos para la elección de Arduino son:

- Es de libre distribución, lo que implica que no hay que pagar licencia por uso.
- Es multiplataforma, se puede ejecutar en cualquier entorno (Windows, Mac, ...)
- Tiene una gran comunidad internacional que colabora de forma activa.
- El entorno de desarrollo es muy simple, permitiendo el acceso a principiantes.
- El bajo precio del hardware, tanto de Arduino original como la infinidad de clones que hay en el mercado.

[2]

2. DESCRIPCION

Este proyecto consiste en un sistema de ayuda al mantenimiento de electroválvulas.

Se divide en dos partes, la primera hace referencia al mantenimiento preventivo y la segunda al mantenimiento predictivo.

En la primera parte, el sistema monitorizará los ciclos que va haciendo una electroválvula y los irá enviando en tiempo real y de forma inalámbrica (WIFI) a un servidor Web donde estará registrado el numero de maniobras máximo que da el fabricante como vida útil. En el momento que se acerque a un porcentaje preestablecido, el sistema dará un aviso de mantenimiento.

Desde el mismo servidor Web, se podrá controlar la electroválvula remotamente y enviar una señal de encendido o apagado que activará el relé asociado a la misma.

La segunda parte, hace referencia al control de fallos. Se comprobará que no existan fugas en la electroválvula y que la bobina funciona correctamente. Para ello se hará servir un caudalímetro situado en la entrada de la misma. La lectura de caudal se enviará igualmente al servidor Web de forma remota y en tiempo real de manera que cuando la electroválvula esté cerrada, no debería haber caudal, si lo hubiese significaría que existe una fuga. De la misma manera, si la electroválvula esta dando señal de activación, pero no hay caudal, significaría que posiblemente la bobina esté dañada. En ambos casos el servidor Web dará un aviso de mantenimiento, esta vez urgente.

En todo momento se podrán visualizar los datos de caudal y el tiempo de vida de la electroválvula.

Los componentes que se utilizaran son:

- Placa Arduino UNO R3
- Alimentador 9V
- Módulo WIFI ESP8266 ESP-01S
- Fuente de alimentación de 12V para la electroválvula
- Relé
- Electroválvula 12VDC
- Caudalímetro

3. OBJETIVOS

La calidad y producción de una planta industrial depende en gran medida del buen funcionamiento de sus máquinas, la avería en alguno de sus elementos puede resultar en reparaciones largas y costosas, así como en paradas de producción. Es por ello que se necesita un mantenimiento adecuado de los equipos, así como la mayor anticipación posible de problemas mediante control de fallos.

Uno de los elementos sensibles en una planta de producción es la electroválvula, aunque se trata de un dispositivo relativamente económico en comparación con el resto de elementos, un fallo en la misma puede resultar en tiempo de inactividad, que en términos de producción tendría grandes inconvenientes y costes.

La solución actual, se base en un plan de mantenimiento preventivo, que dependiendo del objetivo que se pretenda conseguir (disponibilidad, reducción de fallos, ...), suele hacerse en base a periodos de tiempo o a diversas métricas como podría ser las horas de funcionamiento de un equipo.

El proyecto que se propone, tratará de dar una alternativa económica a los sistemas de mantenimiento existentes en el mercado. La orden de intervención, no vendría dada por el tiempo desde la primera instalación, sino que se van a ir almacenando el número de maniobras y la orden de intervención vendría dada por el uso real del elemento, ahorrando así de manera significativa tanto en el recambio en sí, como en el número de intervenciones.

El sistema trabajará también en modo predictivo. Mediante la medición continua del caudal, se detectará cualquier posible anomalía y se generará una orden de intervención, anticipando así una posible avería y de nuevo ahorrando los costes que ésta habría producido.

Pongamos como ejemplo y sin entrar en detalle, una refinería, donde existen multitud de tuberías con distintos tamaños y fluidos, y, que para interrumpir o desviar esos fluidos, se utilizan válvulas de gran tamaño. Estas válvulas a su vez, están pilotadas por electroválvulas.

Con un sistema de mantenimiento tradicional, un operario debería desplazarse cada x tiempo a hacer un control visual. De la misma manera, cada x tiempo se debería realizar un mantenimiento del producto, ya sea cambiando elastómeros, bobinas, o simplemente sustituyendo la electroválvula por una nueva. Ahora multipliquemos esto por los cientos de electroválvulas que existen en una refinería.

Mediante la adaptación de este proyecto, no sería necesaria esa inspección visual ya que cualquier anomalía sería rápidamente informada. Además, se conseguiría alargar la vida de esas electroválvulas mas tiempo, ya que solo se cambiarían cuando el número de maniobras estuviese próximo al máximo de su vida útil. Existiría un histórico completo de toda la vida del elemento en cuanto a tres parámetros: tiempo, número de ciclos y caudal.

El objetivo de este proyecto es, por tanto, monitorizar tanto el número de ciclos que trabaja una electroválvula como el caudal del fluido que pasa por ella y transmitir toda esta información por WIFI a un servidor Web.

Para poder cumplir con las expectativas del proyecto, se marcan los siguientes objetivos específicos en los que el sistema debe:

- Almacenar el número de maniobras de una electroválvula.
- Dar un aviso de mantenimiento al llegar a un umbral de maniobras
- Permitir accionar una electroválvula remotamente a través de un web Server.
- Detectar si existe una fuga en el sistema
- Detectar si pudiera haber un fallo en la bobina de la electroválvula
- Guardar la información en caso de corte de suministro eléctrico.
- Visualizar e interactuar con el sistema desde cualquier dispositivo conectado a la red WIFI mediante el protocolo HTTP.

4. VIABILIDAD

Los costes de fabricación de este sistema, variarían en función del número de elementos que se quiere monitorizar, de la misma manera, los sensores adecuados para cada elemento pueden tener grandes diferencias de precio.

El paquete básico, a falta de concretar modelos exactos (previsto en la etapa de planificación), tendría el siguiente coste aproximado:

Componente	Precio (octubre 2019)
Arduino UNO Rev3	19,90€
Modulo WIFI ESP8266 ESP-01S	2,67€
Regulador de tensión	1,14€
Alimentador 9V para módulo WIFI	8,99€
Total:	32,70€

Estos costes son con elementos comprados en Amazon a fecha octubre 2019. De cara a una versión comercial, serian bastante inferiores ya que las compras se harían en otros proveedores y comprando por lotes, con lo que el coste se reduciría sensiblemente.

Para esta aplicación en la que se controlará y monitorizará una electroválvula, el coste de los componentes necesarios sería:

Componente	Precio (octubre 2019)
Fuente alimentación 64W	12,50€
Electroválvula 12VDC	8,60€
Relé	1,60€
Caudalímetro	8€
Total:	30,70€

Estos componentes, en principio los tendría el cliente, exceptuando los sensores que ejecutan la parte preventiva, en este caso el caudalímetro.

Ya que el coste dependerá del número de elementos a monitorizar, en un supuesto de una máquina donde se quieran monitorizar 10 electroválvulas, bastaría un sensor para las 10.

Se estima el tiempo de montaje del paquete básico en 1 hora y la adaptación del sensor a la instalación del cliente otra hora más.

Comparando en diferentes empresas como Randstad o Indeed [3], el salario medio de un técnico electrónico estaría sobre los 20.000€. Sumando el resto de cotizaciones que habría que pagar: contingencias comunes, cotización por desempleo, accidentes de trabajo y enfermedades, formación profesional y cotización a Fogasa, supondría un aumento del 30,9% sobre el sueldo bruto. El total sería 26.180€. Contando 1944 horas anuales, quedaría un coste por hora de 13,46€.

Resumiendo, los costos de desarrollo en el supuesto anterior serían:

Componente	Precio (octubre 2019)
Paquete básico	32,70€
Caudalímetro	8€
Mano de obra: 2 horas	26,92€
Total:	67,62€

A este coste, habría que añadir los costos de comercialización. Para ello y en un inicio, se empezaría con un sistema de promoción online. Una aproximación a estos costos sería:

Concepto	Estimación por mes
Página web + hosting + dominio	10€
Publicidad online, precio medio 0,20€ por clic, límite en 200 clics diarios.	1200€
Total:	1210€

Estimando una venta de 300 unidades el primer año, tendríamos:

Concepto	Estimación anual
Fabricación 300 unidades	20.286€
Costos comercialización	14.520€
Total:	34.806€

Esta estimación, nos daría un coste por aparato de 116,02€, aplicando un margen comercial de un 50%, tendríamos un precio de venta de 232,04€

El mercado objetivo sería muy amplio. En cualquier instalación industrial, especialmente donde existan zonas en las que el fallo de un componente sea crítico cualquier ayuda a la monitorización tendría mucho valor.

Existen en el mercado muchas herramientas de software de apoyo al mantenimiento, en plantas de proceso, por ejemplo, existen sistemas similares, aunque mucho más complejos integrados en sus sistemas de control distribuido. La diferencia principal es que en esos sistemas no existe un control real de lo que está pasando en la instalación, esa diferencia sería la que permitiría a este producto encajar en el mercado.

5. PLANIFICACION

Para la planificación, no se ha tenido en cuenta ningún día festivo puesto que puedo dedicar unas horas diariamente, independientemente de si es festivo o no.

Respecto al tiempo empleado en algunas tareas (las de programación mayormente), tengo poca experiencia y por eso el espaciado temporal es similar.

La lista de tareas definidas sería:

Nombre	Fecha de in...	Fecha de fin
Decisión del proyecto	19/09/19	25/09/19
PAC1: Planificación del trabajo	26/09/19	2/10/19
• Descripción y objetivos	26/09/19	27/09/19
• Viabilidad	28/09/19	29/09/19
• Índice provisional	30/09/19	30/09/19
• Temporización: Diagrama de Gantt	1/10/19	2/10/19
PAC2: Primera entrega del proyecto	3/10/19	13/11/19
• Elección y compra del hardware	3/10/19	6/10/19
• Diseño esquemas electrónicos	7/10/19	16/10/19
• Montaje del hardware	17/10/19	26/10/19
• Instalación de firmwares	27/10/19	5/11/19
• Testeo individual simple de elementos	6/11/19	13/11/19
PAC3: Segunda entrega del proyecto	14/11/19	18/12/19
• Programación código consulta	14/11/19	23/11/19
• Accionamiento y registro electroválvula	14/11/19	18/11/19
• Lectura caudalímetro	19/11/19	23/11/19
• Programación alarmas	24/11/19	7/12/19
• Alarma porcentaje alcanzado	24/11/19	30/11/19
• Alarma caudalímetro	1/12/19	7/12/19
• Configuración de Web Server	8/12/19	14/12/19
• Tests	15/12/19	18/12/19
Memoria final	19/12/19	5/01/20
• Definición estructura	19/12/19	20/12/19
• Redacción	21/12/19	1/01/20
• Revisión y corrección	2/01/20	5/01/20
Presentación y código	6/01/20	12/01/20
• Guión presentación	6/01/20	8/01/20
• Grabación presentación en video	9/01/20	10/01/20
• Depuración final código	11/01/20	12/01/20

Y el diagrama de Gantt correspondiente:

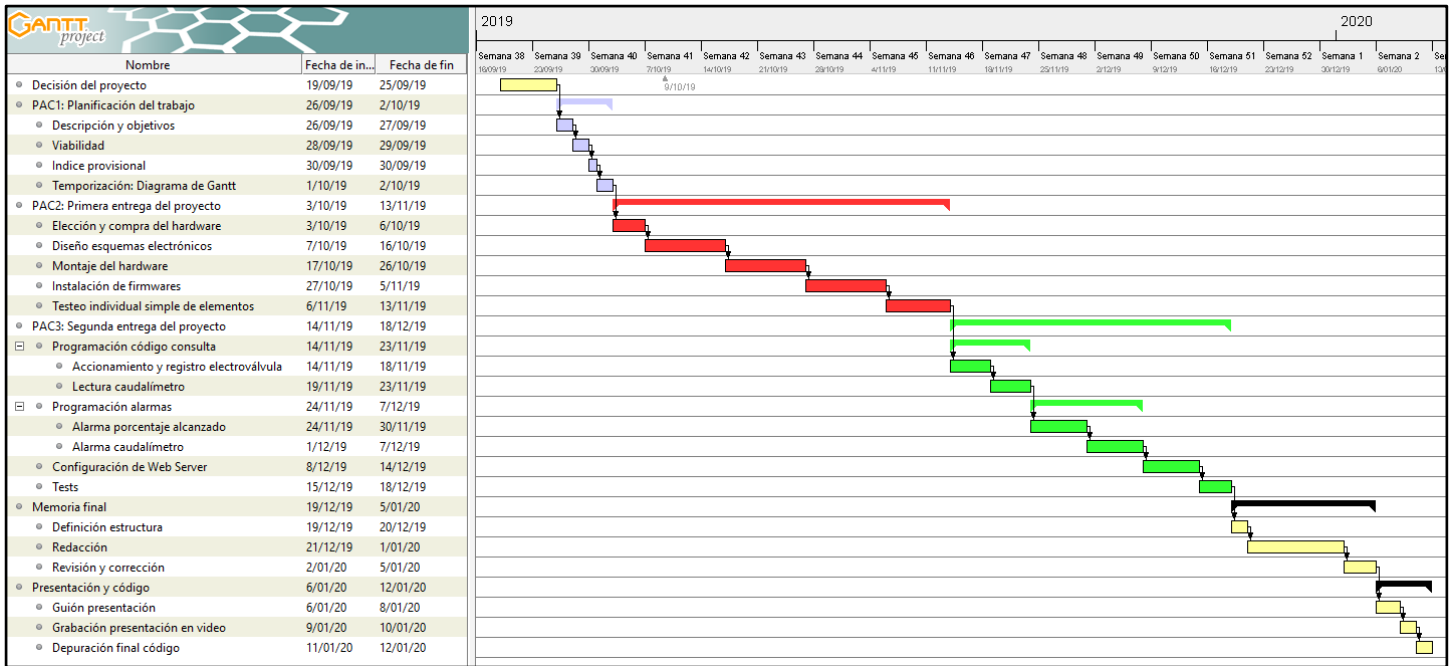


Figura 1 - Diagrama de Gantt

6. HARDWARE

6.1. [Componentes electrónicos](#)

A continuación, se muestran los componentes utilizados y sus características

- **Arduino Uno R3**

Se trata de una placa electrónica de muy bajo coste basada en el chip de Atmel ATmega328P, tiene todos los elementos necesarios para conectar periféricos a sus entradas y salidas y puede ser programada tanto en Windows como macOS y GNU/Linux.



Figura 2 - Arduino UNO R3

Características técnicas principales [4]:

Microcontrolador	ATmega328P
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Pines de E/S digitales	14 (6 con salida PWM)
Pines de entrada analógica	6
Corriente DC por pin de E/S	40 mA
Corriente DC para 3.3V Pin	50 Ma
Memoria Flash	32 Kbyte (0,5 KB utilizados por bootloader)
SRAM	2 Kbyte (ATmega328)
EEPROM	1 Kbyte (ATmega328)
Velocidad de reloj	16 Hz

- Módulo WIFI ESP8266 ESP-01S
Chip de bajo coste con conexión WiFi y compatible con protocolo TCP/IP. Incluye además un microcontrolador y 2 GPIO. Este módulo no permite encajarse fácilmente en una protoboard, para solucionarlo se puede fabricar o comprar un sencillo adaptador que lo facilita. Su mayor ventaja, aparte de su precio, es su bajo consumo y su tamaño.
Hay que tener presente que el voltaje de operación es de 3 a 3,6V.

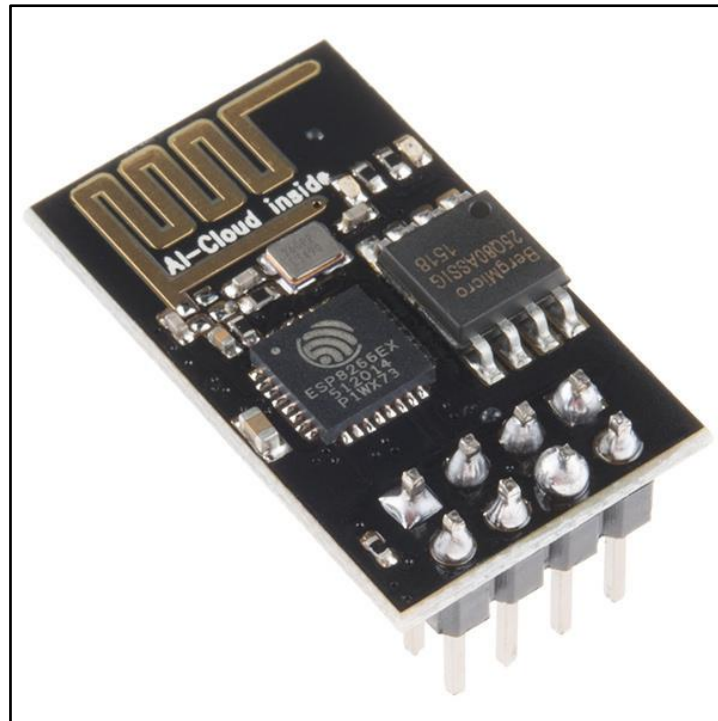


Figura 3 - Módulo WIFI ESP8266 ESP-01S

Características técnicas principales:

Microcontrolador	32-bit MCU @ 80MHz
Voltaje de operación	3,3V
Consumo	100mA
Memoria Flash	1Mhz (ESP01 tenía 512Kb)
Pines de E/S digitales	2
Soporta comunicación en serie, por lo tanto, es compatible con muchas plataformas como Arduino	
Puede programarse usando el IDE de Arduino o comandos AT	
Soporta "Deep sleep" (<10uA)	

Configuración de los pines [5]:

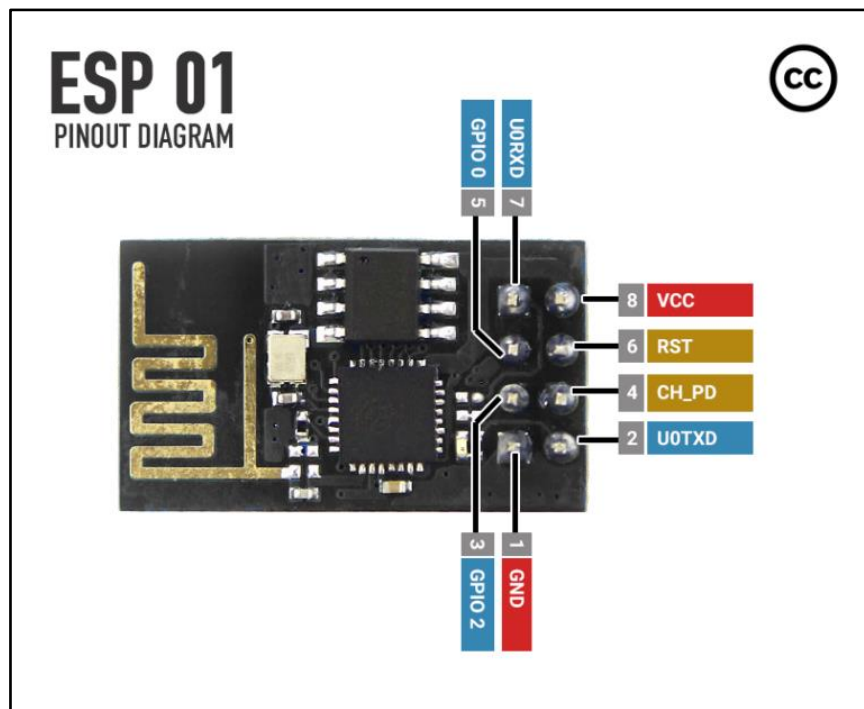


Figura 4 - Pin-out ESP-01s

PIN	NOMBRE	DESCRIPCION
1	GROUND	Conexión a tierra
2	TX	Se conecta al pin RX de Arduino
3	GPIO 2	Pin I/O
4	CH_EN	“Chip Enable” ha de estar siempre activo
5	GPIO 0	Pin I/O, se utiliza también para activar el modo de flash del firmware
6	RESET	Resetea el módulo
7	RX	Se conecta al pin TX de Arduino
8	VCC	Conexión a 3,3V

Los pines TX y RX se podrían usar como I/O cuando no se usasen para transmisión.

- **Regulador de tensión**
Regulador de voltaje con entrada de 4.5 a 28V y salida de 0.8 a 20V
Permitirá conectar la fuente con salida de 12V y regularla a 3,3V para alimentar el módulo WIFI.

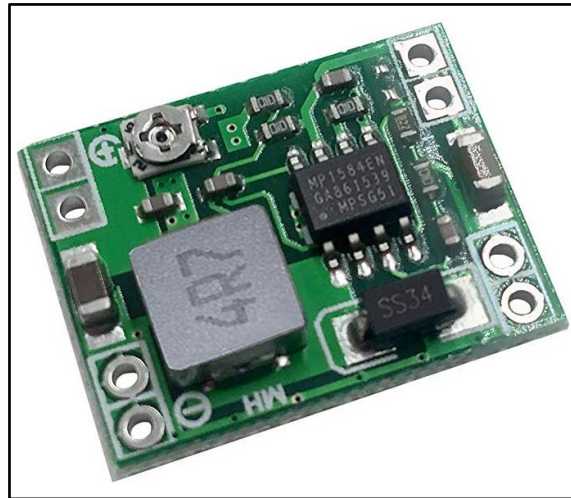


Figura 5 - Regulador de tensión

- **Alimentador 9V**
Alimentador con salida de 9V DC 1000mA y con conector de 5,5mm x 2,1mm para poder conectar Arduino directamente a 230V



Figura 6 - Alimentador 9V

- Fuente de alimentación
Fuente de alimentación de 64W, permite conectar a 230V 50Hz y da una tensión de salida de 12V DC. Necesaria para alimentar elementos como las electroválvulas.



Figura 7 - Fuente de alimentación

- Electroválvula
Electroválvula de 2 vías, normalmente cerrada, 1/2", alimentada a 12V DC y con un consumo de 300mA, la presión máxima admisible es de 0.8Mpa (8bar)

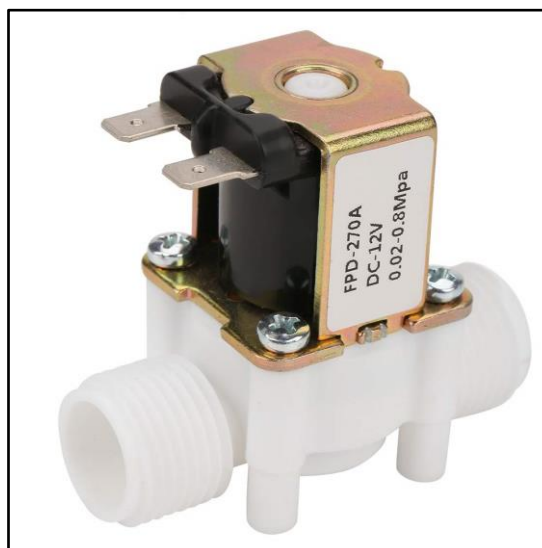


Figura 8 - Electroválvula 12V DC

- Relés
Módulo Relé de 1 canal a 5V, adecuado para controlar otro dispositivo con corrientes más altas. Admite hasta 250V AC o 30V DC / 10A
El relé permitirá trabajar a 12V DC con la electroválvula.

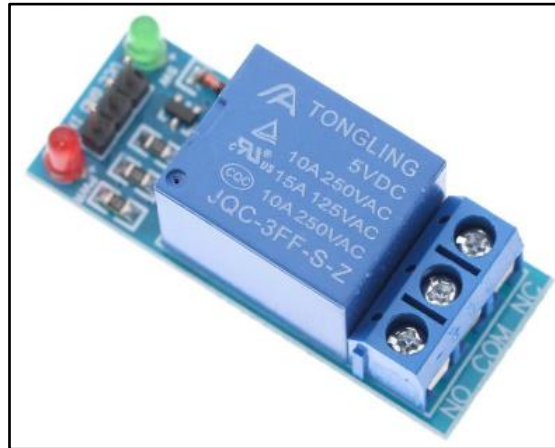


Figura 9 - Relé 1 canal 5V

- Caudalímetro
Sensor que permite medir la cantidad de agua que atraviesa una tubería. Consta de una turbina que gira al pasar el fluido a través de ella, un imán situado en la misma, genera un pulso positivo cada vez que pasa por un sensor de efecto Hall. De esta forma podemos conocer las RPM generadas y calcular el caudal de agua con una sencilla ecuación.



Figura 10 - Caudalímetro

6.2. Esquema de conexión

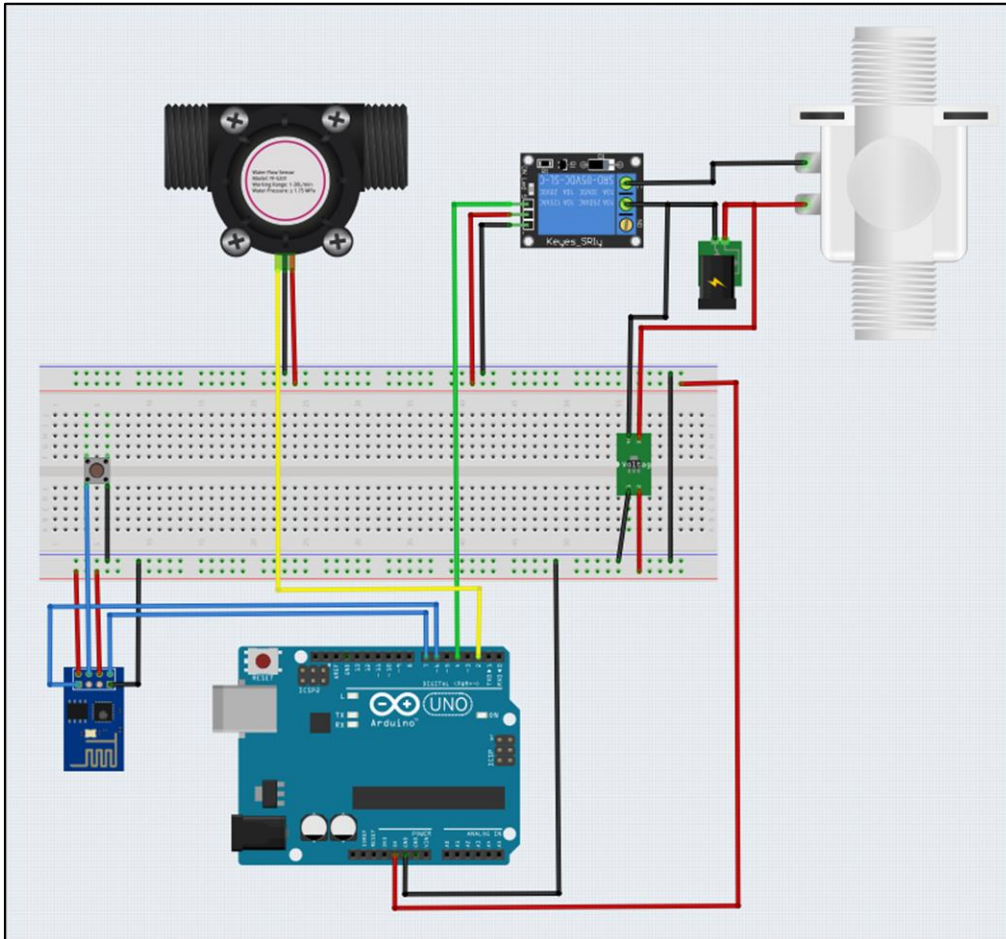


Figura 11 - Esquema de conexión

Para dibujar el esquema se ha utilizado Fritzing [6], herramienta de código abierto que facilita el prototipado de componentes electrónicos.

Fritzing dispone de bibliotecas con la mayoría de componentes además de una comunidad de usuarios muy activa que proporciona nuevas partes, permite dibujar tanto esquemas eléctricos como el diagrama para fabricar una PCB a medida y se pueden listar los componentes utilizados.

6.3. [Detalle de conexión](#)

Regulador de voltaje

Para poder alimentar el módulo WIFI, se necesitan 3,3V. La salida de 3,3V de Arduino, proporciona una intensidad máxima de 50mA, por lo que no sería apta para poder usarla ya que en según qué momento, el módulo ESP8266 puede requerir hasta 200mA, en caso contrario existe el riesgo de cortes intermitentes en la comunicación.

Mediante un regulador de voltaje, se pueden proporcionar 3,3V de una manera muy sencilla. Se alimenta desde la fuente de alimentación a 12V y mediante un tornillo se regula a la salida deseada.

Electroválvula + Relé

Para poder accionar la electroválvula, se necesita una corriente de 300mA y una tensión de 12V DC, para ello es imprescindible el uso de un relé y una fuente de alimentación adicional.

La fuente transforma el voltaje de 230V 50Hz a 12V DC, además proporciona hasta 10A. El relé funciona como un interruptor el cual se acciona enviándole 5V desde Arduino, en el momento que cierra el contacto, se alimenta la electroválvula desde la fuente.

Caudalímetro

El conexionado del caudalímetro es muy sencillo, necesitando únicamente conectarse a 5V, tierra y a una de las E/S de Arduino. Hay que tener en cuenta que, para el cálculo de caudal, será necesario utilizar interrupciones por hardware que en Arduino UNO están limitadas a los pines 2 y 3.

Módulo WIFI ESP8266

Este módulo requiere conectar siempre positivo (pin 8), tierra (pin 1), los 2 pines de transmisión (2 y 7) y el chip activo (pin 4). Es conveniente, como se ve en el esquema, conectar el pin de RESET (pin 6) a un botón, de manera que al pulsarlo lo conecta a tierra y resetea el módulo.

Los pines de transmisión se conectan a los pines 6 y 7 ya que se emplearán como puertos serie por software, el motivo es que, si se utilizan los pines de transmisión serie estándar RX y TX, no se puede utilizar el puerto serie para la comunicación con el PC. Esto implicaría desconectar el módulo para programar con Arduino y trabajar sin poder ver la salida del puerto serie.

6.4. [Firmware ESP8266](#)

El módulo ESP8266 integra un SOC (System on chip) que permite proporcionar WIFI a cualquier micro controlador/procesador. Se puede usar tanto con Arduino como de forma independiente.

Existen diferentes modelos de este módulo, actualizar el firmware a la última versión evitará no tener errores, asegurará que el módulo funciona correctamente y en muchos casos añade mejoras al mismo.

Este es el proceso seguido para actualizar el Firmware:

En cuanto al hardware, la manera más sencilla es usando un adaptador FTDI a USB. Para este proyecto y al no disponer de uno, se ha usado el propio Arduino, quitando temporalmente el microcontrolador ATmega y usando el puerto USB con interface FTDI incorporada en la placa.

La conexión es muy similar a la utilizada en la figura 10, solo que esta vez sí que se conectan los pines de comunicación a RX y TX y el pin GPIO 0 (pin 5) a tierra para poner el módulo en modo UART.

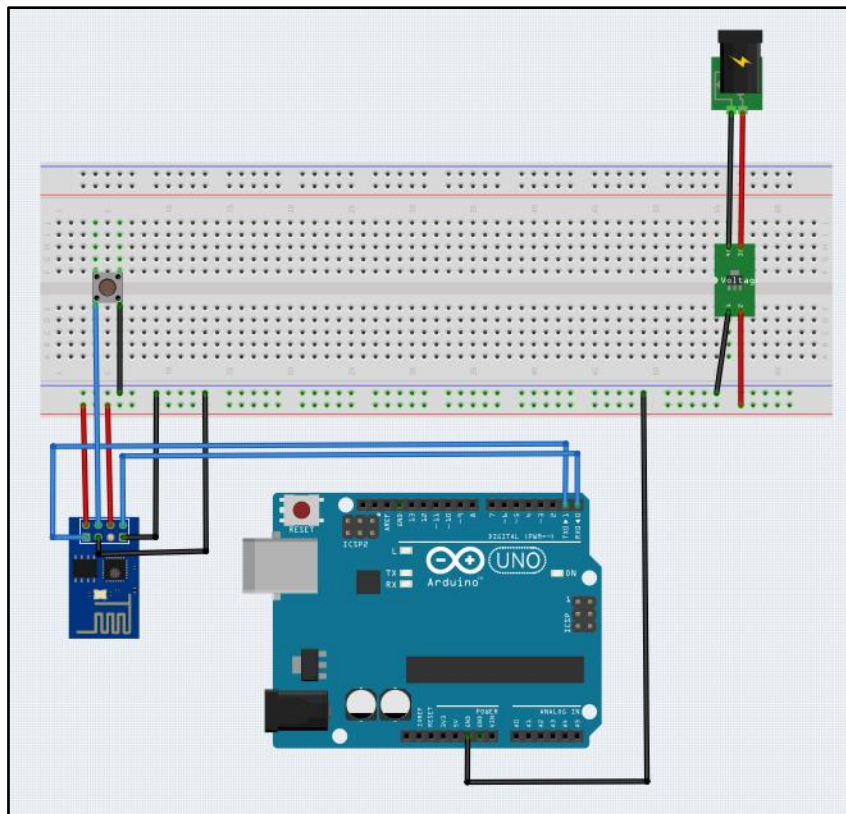


Figura 12 - Esquema de conexión ESP8266 modo UART

Para el software, se ha de descargar tanto la herramienta “ESP Flash Download Tool” como el SDK perteneciente al módulo, en nuestro caso, la versión que más nos convenga de “Non-OS ESP8266 SDK firmware” desde la página oficial de Espressif [7].

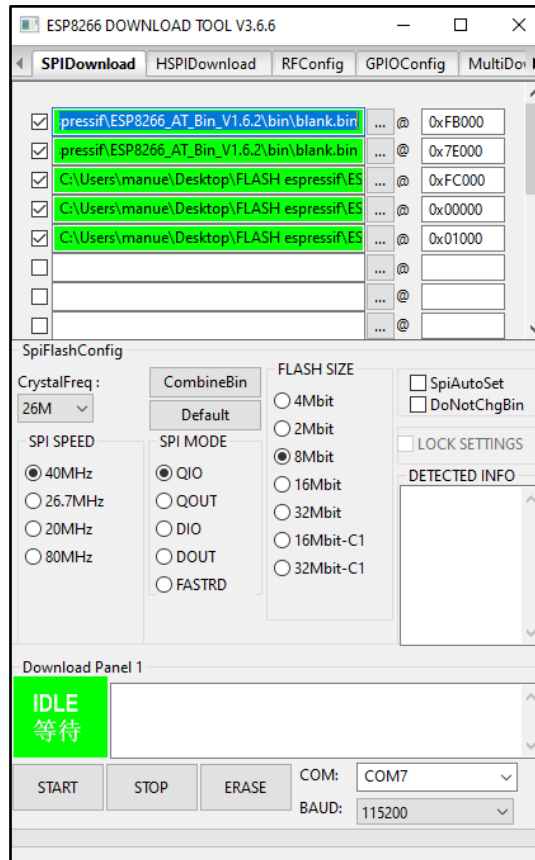


Figura 13 - ESP Flash download tool

Se han de introducir las direcciones de memoria donde colocar cada uno de los archivos que se van a subir. Para ello, en el directorio del SDK descargado bin\at se encuentra un archivo README donde se encuentran esas direcciones dependiendo del tamaño de memoria flash de nuestro módulo.

Una vez rellenado y pulsando START, cuando acabe el proceso, que puede durar unos minutos, el módulo tendrá el último firmware instalado.

También es necesario cambiar la velocidad de transmisión del modulo para su compatibilidad total con Software Serial, mediante el comando: AT+UART_DEF=9600,8,1,0,0 se consigue cambiar la velocidad a 9600 baudios de forma permanente.

7. SOFTWARE

7.1. [Entorno de desarrollo](#)

Para la programación de la parte del microcontrolador, se ha utilizado exclusivamente el IDE de Arduino. Este entorno se compone de:

- Editor de código
- Compilador
- Depurador
- Interfaz gráfica (GUI)
- Herramientas para la carga del programa compilado en la memoria flash del hardware
- Otras funciones: Como gestión de librerías, gestión de placas, ...

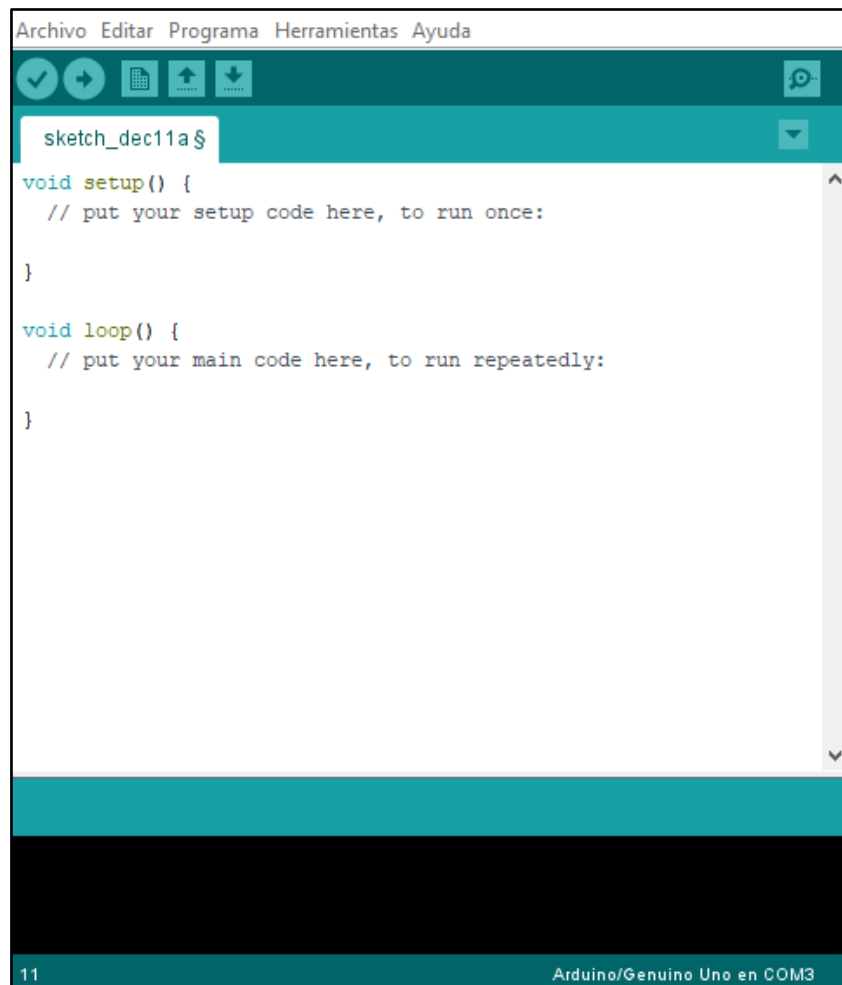


Figura 14 - IDE Arduino

El lenguaje, es una versión simplificada de C++.

Las partes principales de un programa en Arduino se localizan en 2 funciones: *setup* y *loop*. La primera se ejecuta una vez al iniciarse el programa y la segunda es un bucle infinito que repite las instrucciones que se encuentren en su interior.

Se podrían haber utilizado otras estrategias, como *multitask*, pero para este proyecto no ha sido necesario ya que, en principio, no se requiere una acción instantánea, incluso en el caso de una micro fuga de fluido que sería la parte más crítica, el operario recibiría el aviso en pocos segundos. Por tanto, se ha utilizado la programación secuencial clásica consistente en las dos funciones antes mencionadas. En cada ciclo del bucle *loop* se hace un barrido de todas las E/S y se envían al servidor web.

En el caso de la programación del servidor Web, se ha utilizado Notepad++ para programar en PHP, HTML y CSS.

Notepad++ es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación.

Además, se ha utilizado un servidor APACHE y una base de datos MYSQL local. Para ello se ha instalado un paquete de XAMPP, acrónimo de **X** (para los diferentes sistemas operativos), **A**pache, **M**ariaDB (fork de MySQL), **P**HP, **P**erl. De esta manera se ha podido programar y testear todo en un equipo local de manera muy sencilla.

7.2. Estructura

El programa debe leer unos valores que luego enviará mediante una petición GET de HTTP a la base de datos MYSQL, estos datos se tratarán mediante PHP y se aprovechará el retorno de la petición para interactuar con el relé que activa las electroválvulas

En el programa se pueden distinguir las siguientes partes:

- Declaración de variables y constantes: Para un mejor seguimiento, se comentarán en cada una de las partes siguientes.
- Configuración de red: Se debe configurar el módulo en modo estación, para conectarse a una red WIFI existente. Destacar que no se ha utilizado ninguna librería, todo se hace mediante comandos AT.
- Lectura de valores: Estado del pin perteneciente a la electroválvula y cálculo de caudal entregado por el caudalímetro.
- Envío y recepción de las lecturas al servidor WEB.

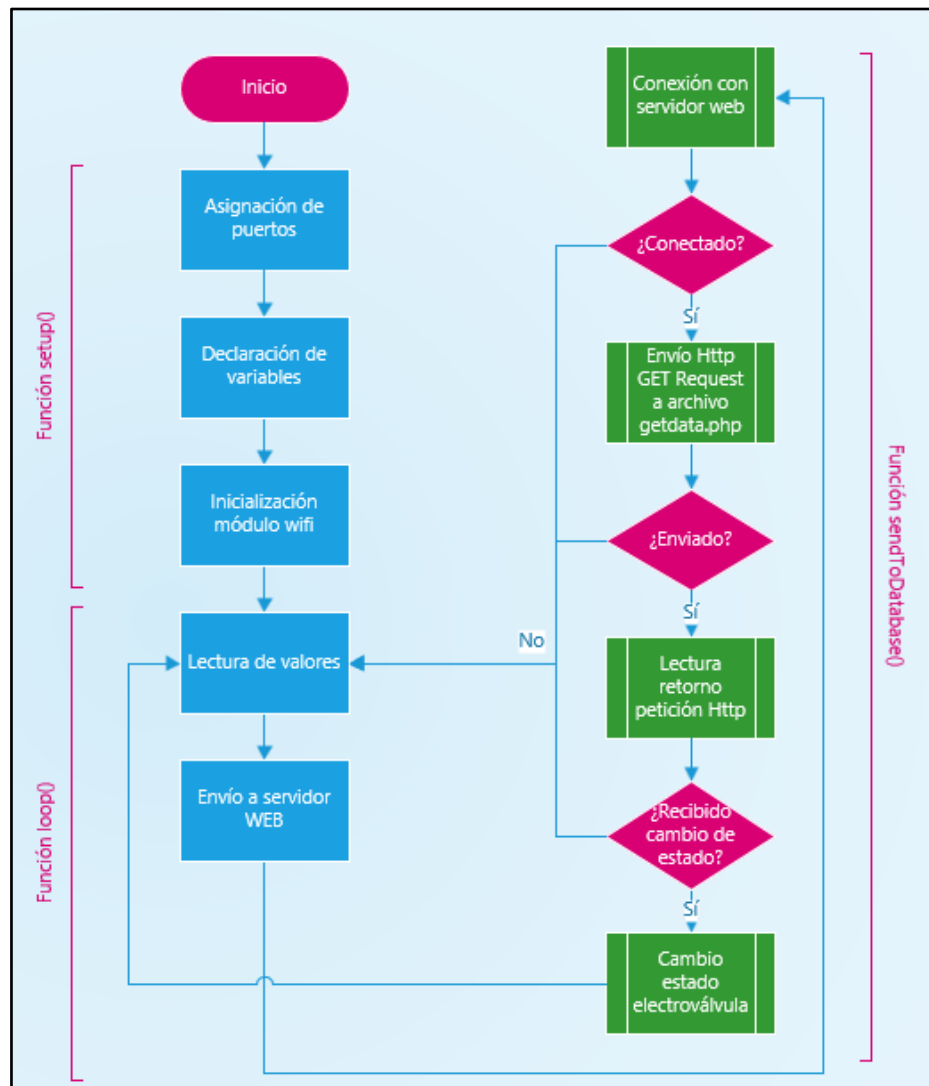


Figura 15 - Diagrama de flujo Arduino

Respecto al servidor WEB, tenemos:

- Base de datos MYSQL
 - Comunicación entre Arduino a MYSQL: Mediante un programa en PHP
 - Servidor WEB: En PHP, HTML y CSS
- Dashboard: Se visualizan todos los datos, así como los mensajes y alarmas que puedan surgir.
 - Electrovalves: Para activar o desactivar las electroválvulas, este módulo no sería necesario en una instalación ya existente.
 - Datalog: Registro con todas las lecturas, para poder realizar todo tipo de análisis.
 - Settings: Para ajustar algunos valores como el número máximo de maniobras.

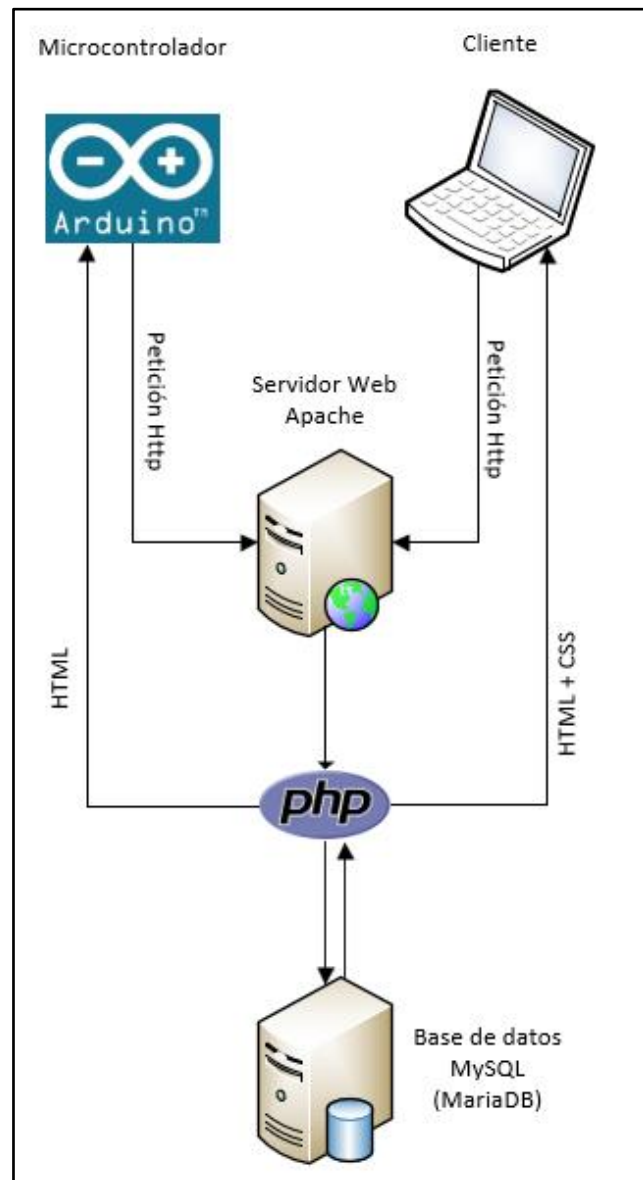


Figura 16 - Diagrama intercambio de información

7.3. [Arduino](#)

7.3.1. *Configuración de red*

Esta parte de código se encarga de preparar el módulo WIFI para enviar y recibir datos.

Para empezar, se resetea el módulo mediante el comando *AT+RST* y se comprueba si está operativo, para ello se usa el comando *AT* a modo de pregunta y si la respuesta es *OK*, significa que sí lo está.

```
//Reset the module
SerialESP8266.println("AT+RST")

//Check if ESP8266 responds
SerialESP8266.println("AT");
if(SerialESP8266.find("OK"))
    Serial.println("AT answer OK");
else
    Serial.println("Error in ESP8266");
```

A continuación, se configura en modo cliente ya que el sistema se conectará a una red WIFI existente, para ello se usa el comando *AT+CWMODE=1*. Para conectar con la red, se utiliza el comando *AT+CWJAP* pasándole como parámetros el usuario y contraseña de acceso a la red WIFI. También se deshabilitan las conexiones múltiples que, al no estar en modo servidor, no son necesarias (*AT+CIPMUX=0*)

```
//-----Net configuration-----//

//ESP8266 in station mode (client)
SerialESP8266.println("AT+CWMODE=1");
if(SerialESP8266.find("OK"))
    Serial.println("ESP8266 en modo Estacion");

//Connecting to a WIFI network
SerialESP8266.println("AT+CWJAP=\"vodafoneDD70\", \"UFUKZTGUVNM8NE\"");
Serial.println("Connecting to the WIFI ...");
SerialESP8266.setTimeout(10000); //Increase if necessary
if(SerialESP8266.find("OK"))
    Serial.println("WIFI connected");
else
    Serial.println("Error trying to connect to the WIFI");
SerialESP8266.setTimeout(2000);
//Disable multiplex mode
SerialESP8266.println("AT+CIPMUX=0");
if(SerialESP8266.find("OK"))
    Serial.println("Multiplex disabled");

//-----End configuration-----//
```

Electroválvula

En primer lugar, se asigna el pin correspondiente y se da un valor de apagado.

```
//Valve is configured as OUTPUT because it will be activated through the
webservice
pinMode(valve, OUTPUT);
digitalWrite(valve, LOW);
```

A continuación, ya en la fase de *loop*, se lee el estado en el que está la electroválvula y se asigna ese valor a una variable para luego poder enviarlo al servidor Web, también se imprime por la consola.

```
valv_state=digitalRead(valve);
Serial.print("Electrovalve: ");
Serial.println(valv_state);
```

Caudalímetro

El sensor de flujo o caudalímetro integra un rotor que tiene un pequeño imán adherido, un sensor magnético de efecto Hall detecta el imán al girar y, de esta manera, se generan pulsos de salida a una velocidad proporcional a la del caudal.

La salida de este sensor es una onda cuadrada cuya frecuencia es proporcional al caudal atravesado, así tenemos que:

$$Q(l/min) = \frac{f(Hz)}{K}$$

Donde K es una constante que depende de cada caudalímetro, el fabricante nos da un valor de referencia de 7,5 que es el que se va a tomar. Con este valor, la precisión es de +- 10%. [8]

En primer lugar, se declara 1 constante para almacenar el pin donde se encuentra el caudalímetro y 2 variables donde se almacenan el factor k y el número de pulsos:

```
Const int flowsensor = 2;    // Flow sensor in pin 2
Float kfactor = 7.5;        // K Factor: To pass from frequency to flow
volatile int pulseCounter;   // Save number of pulses in flowsensor
```

Para medir el número de pulsos, se hace servir una interrupción por hardware. La función de interrupción y el código ejecutado es el siguiente:

```
// Interrupt service routine (ISR)
void ISRPulses()
{
  pulseCounter++; // Number of pulses +1
}
attachInterrupt(digitalPinToInterrupt(flowsensor), ISRPulses, RISING);
```

Por último y dentro del *loop*, las llamadas a la función de interrupción y la impresión por la consola, sería:

```
noInterrupts(); //disable interrupts
pulseCounter = 0; //Set pulseCounter to 0 ready for calculations
interrupts(); // re-enable interrupts

delay (1000); //Wait 1 second

noInterrupts(); //disable interrupts so calculation sees an atomic value
flow = (pulseCounter / kfactor); //Pulse frequency / K factor
interrupts(); // re-enable interrupts.

Serial.print ("Flow: ");
Serial.print (flow, DEC); //Prints the number calculated above
Serial.print (" l/min\r\n"); //Prints "l/min" and returns a new line
```

7.3.3. Envío y recepción de lecturas al servidor WEB

Este apartado se encarga de enviar la información del estado de las electroválvulas y del caudal calculado al instante al archivo *getdata.php*. Se engloba dentro de una función, llamada *sendToDatabase()*, a la que se invoca dentro del *loop* y luego se aplica un retardo de 5 segundos.

La primera parte consiste en conectarse al servidor, como se está trabajando con un servidor local, que sería *localhost*, debemos indicar aquí la dirección IP en su lugar. Para averiguar la dirección IP, bastaría con ejecutar IPConfig desde Windows.

Se ejecuta, por tanto, el comando AT para iniciar una comunicación TCP con el servidor:

```
void sendToDatabase() // Connection with MySQL
{
  //Connection to the server

  SerialESP8266.println("AT+CIPSTART=\"TCP\", \"192.168.0.22\", 80")
```


A continuación, siempre que se tenga una respuesta positiva del módulo (OK), se monta la cabecera de la petición HTTP GET y se envía mediante el comando AT correspondiente, indicándole su longitud.

```
if( SerialESP8266.find("OK"))
{
    Serial.println();
    Serial.println();
    Serial.println();
    Serial.println("ESP8266 connecting to the server...");

    //Mounting the header for the http request
    String httpRequest= "GET
/maintenance/getdata.php?valv_state=";
    httpRequest=httpRequest+String(valv_state)+"&flow="+
String(flow)+" HTTP/1.1\r\n";
    httpRequest=httpRequest+"Host: localhost\r\n";

    //Send the size of the http request
    SerialESP8266.print("AT+CIPSEND=");
    SerialESP8266.println(httpRequest.length()+2);
    //Wait for ">" to send http request
    if(SerialESP8266.find(">"))
    {
        Serial.println("Sending HTTP request ...");
        SerialESP8266.println(httpRequest);
        if( SerialESP8266.find("SEND OK"))
        {
            Serial.println("HTTP request sent:");
            Serial.println();
            Serial.println(httpRequest);
        }
    }
}
```

Una vez enviado, el siguiente paso es tratar la respuesta. Para ello se crea una variable, *endResponse*, donde se almacena toda la cadena que viene de vuelta.

Mediante condicionales *If / else* y buscando dentro del string, se localizan las variables necesarias, en este caso al haber solo una electroválvula, será b1. Una vez obtenida, se activa o desactiva.

Si se recibe un Warning, quiere decir que la base de datos no está conectada y si el tiempo de espera es de más de 10 segundos, se finaliza la conexión.

Al finalizar la transmisión, se cierra la conexión.

```

boolean endResponse=false;
long initialTime=millis();
espString="";

while(endResponse==false)
{
    while(SerialESP8266.available(>0)
    {
        char c=SerialESP8266.read();
        //Serial.write(c);
        espString.concat(c); //Save the response in the string
        "espString"
    }
    if(espString.indexOf("Warning")>0) //If a Warning returns, means
    database is not connected
    {
        Serial.println();
        Serial.println("Cannot connect to MySQL database");
        endResponse=true;
    }
    if((millis()-initialTime)>10000) //Finishing if 10 secs have
    passed
    {
        Serial.println("Timeout");
        SerialESP8266.println("AT+CIPCLOSE");
        if( SerialESP8266.find("OK"))
            Serial.println("Connection finished");
            endResponse=true;
        }
        if(espString.indexOf("_b1")>0) //Locate b1, if we
        receive it, response has finished.
        {
            String state_text = "";
            int strposition = espString.indexOf("_b1");
            int chg_state =
            espString.substring(strposition+3,strposition+4).toI
            nt();
            if (chg_state == 0){
                state_text = "LOW";
                digitalWrite (valve, LOW);
            }else if (chg_state == 1){
                state_text = "HIGH";
                digitalWrite (valve, HIGH);
            }
            Serial.println();
            Serial.print("Changed electrovalve state to " +
            state_text);
            endResponse=true;
        }
    }
}

```

7.4. [Base de datos MYSQL](#)

Como se ha mencionado anteriormente, se ha utilizado una base de datos MYSQL para almacenar los valores registrados.

El nombre de la base de datos es “maintenance” y tiene dos tablas: valves y state.

La primera, *valves*, es básicamente un “datalogger”. En ella se almacenan fecha y hora, estado de la válvula (activada o no) y el caudal en ese instante.

Esta tabla puede contener miles de registros que corresponden a una sola electroválvula.


Nombre	Tipo
ID 	bigint(20)
valv_state	int(11)
flow	float
date	timestamp

Figura 17 - Tabla valves

La segunda tabla, *state*, contendría únicamente un registro por electroválvula. En ella se verá el estado actual de la electroválvula (*state_EV1*), el número de pulsos que lleva cada una de ellas (*counter*), los valores máximos que aconseja el fabricante (*max_pulses*) y el umbral a partir del que se recibirá un aviso (*threshold*).

Por ultimo se ha añadido un booleano para indicar si la electroválvula está conectada o no (*active*).


#	Nombre	Tipo
1	id 	bigint(11)
2	state_EV1	int(11)
3	counter	int(11)
4	threshold	int(11)
5	max_pulses	int(11)
6	active	tinyint(1)

Figura 18 - Tabla state

7.5. [Comunicación entre Arduino y MYSQL](#)

La petición GET HTTP que se envía desde Arduino, se recoge a través del archivo *getdata.php*. Este archivo se puede dividir en dos partes, donde el primero sería la introducción de los valores de estado y caudal en la tabla *valves*.

En esta parte podemos ver tres funciones, la primera es un constructor que llama a las otras dos. La segunda es la que se ocupa de la conexión con la base de datos, al usar un servidor local, indicamos localhost y para mayor facilidad en esta fase, se utiliza usuario *root* y no se usa contraseña. La última función es la que crea la consulta de inserción en SQL.

El último paso es añadir el registro a la tabla con los datos recibidos por la petición GET.

```

class valves{
    public $link='';
    function __construct($valv_state, $flow){
        $this->connect();
        $this->storeInDB($valv_state, $flow);
    }

    function connect(){
        $this->link = mysqli_connect('localhost','root','') or
        die('Cannot connect to the DB');
        mysqli_select_db($this->link, 'maintenance') or
        die('Cannot select the DB');
    }

    function storeInDB($valv_state, $flow){
        $query = "insert into valves set
        valv_state='".$valv_state."', flow='".$flow.'";
        $result = mysqli_query($this->link,$query) or
        die('Errant query: '.$query);
    }
}
if($_GET['valv_state'] != '' and $_GET['flow'] != ''){
    $valves=new valves($_GET['valv_state'],$_GET['flow']);
}

```

En la segunda parte del mismo archivo, se ejecuta una nueva consulta en SQL, pero esta vez es de lectura. En ella se retorna como respuesta de la petición GET HTTP a Arduino el estado de la electroválvula que previamente pudiera haber sido modificado mediante el servidor WEB, de manera que se pueda tratar al llegar a Arduino y activarla o no.

```

//Connect to database
$link = mysqli_connect('localhost','root','');
mysqli_select_db($link, 'maintenance') or die('Cannot select the
DB');

$result = mysqli_query($link,"SELECT state_EV1 FROM state");
//Select evt_status to change

while($row = mysqli_fetch_array($result)) {

    //We update the values for the boolean we receive from
the Arduino, then we echo the boolean
//from the database back to the Arduino
    $b1 = $row['state_EV1'];

    //Next line will echo the data back to the Arduino
    echo " _b1$b1##";
}
echo "CLOSED";

```

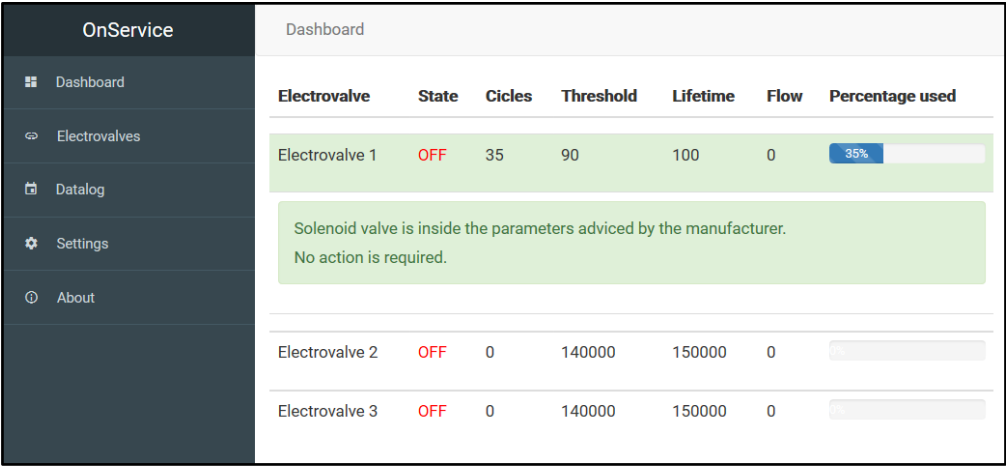
7.6. [Servidor WEB](#)

El servidor WEB se ha hecho en HTML, CSS y PHP. Se ha utilizado parte de CSS de Bootstrap para tener un resultado más visual y además para que sea “responsive” o “adaptativo”, esto es, que busca la correcta visualización de una misma página en distintos dispositivos.

El servidor se ha dividido en 4 apartados accesibles desde un menú lateral. Estos apartados, que se desarrollan más adelante, son: Dashboard, Electrovalves, Datalog y Settings.

7.6.1. *Dashboard*

Desde la página de *Dashboard* se puede tener una visión global del estado de todo el sistema. En él, se pueden ver las electroválvulas conectadas (en el caso que nos ocupa, únicamente una es funcional), su estado (activada o no), el número de ciclos que lleva desde que se instaló, el umbral desde el que se recibirá el primer aviso de mantenimiento, el tiempo de vida que recomienda el fabricante, el caudal en ese instante (tiene un retardo de unos segundos) y el porcentaje de vida.



The screenshot shows a web dashboard titled "OnService" with a sidebar menu containing "Dashboard", "Electrovalves", "Datalog", "Settings", and "About". The main content area is titled "Dashboard" and contains a table with the following data:

Electrovalve	State	Cicles	Threshold	Lifetime	Flow	Percentage used
Electrovalve 1	OFF	35	90	100	0	35%
Solenoid valve is inside the parameters advised by the manufacturer. No action is required.						
Electrovalve 2	OFF	0	140000	150000	0	
Electrovalve 3	OFF	0	140000	150000	0	

Figura 19 – Dashboard

Las partes principales del código son las siguientes:

En primer lugar, hay que volver a conectarse a la base de datos y crear una consulta en SQL que devolverá todos los registros de la tabla *state*.

```
$link = mysqli_connect('localhost','root','');  
mysqli_select_db($link, 'maintenance') or die('Cannot select the  
DB');  
$result = mysqli_query($link, "SELECT * FROM state");
```

A continuación, se crea una tabla HTML, se indican los nombres de campo y se listan todos los registros de la tabla *state* en la tabla HTML mediante un bucle *while*. Se crea una segunda consulta SQL para recuperar el estado de la electroválvula, obtenido del último registro de la tabla *valves*.

```

echo "
<table class='table'>
<thead>
  <tr>
    <th>Electrovalve</th>
    <th>State</th>
    <th>Cicles</th>
    <th>Threshold</th>
    <th>Lifetime</th>
    <th>Flow</th>
    <th>Percentage used</th>
  </tr>
</thead>
<tbody>
  <tr><td></td></tr>
";

while($row = mysqli_fetch_array($result)) {

  $result2 = mysqli_query($link, "SELECT valv_state,flow FROM
valves ORDER BY ID DESC LIMIT 1");
  $row2 = mysqli_fetch_array($result2);
  $flow = $row2['flow'];
  $state = $row2['valv_state'];
  if ($row['active'] == true) {
    echo "<tr class='success'>";
  } else {
    echo "<tr>";
  }
  $unit_id = $row['id'];
  $cicles = $row['counter'];
  $threshold = $row['threshold'];
  $max_pulses = $row['max_pulses'];
  $bar = ($cicles * 100)/$max_pulses;
  echo "<td> Electrovalve " . $row['id'] . "</td>";
  if ($state == 0){
    echo "<td style='color:red'>OFF</td>";
  } elseif ($state == 1){
    echo "<td style='color:green'>ON</td>";
  }
  echo "<td>" . $row['counter'] . "</td>";
  echo "<td>" . $row['threshold'] . "</td>";
  echo "<td>" . $row['max_pulses'] . "</td>";
  echo "<td>" . $flow . "</td>";
  echo "<td>
<div class='progress progress-striped active'>
  <div class='progress-bar' role='progressbar'
    aria-valuenow='".$bar."' aria-valuemin='0' aria-
    valuemax='100'
    style='width: ".$bar."%'>".$bar."%
  </div>
</div>
</td></tr>";
}

```

Por último, se crean los diferentes sistemas de alerta con todos los datos obtenidos a través de condicionales *if/else*. El color y tipo de mensaje cambia, en función de la gravedad.

El mantenimiento preventivo se realizaría de manera que se compara el número de ciclos que lleva la electroválvula, con dos variables, el umbral y el límite máximo.

Si la electroválvula lleva un número de ciclos inferior al umbral, se mostrará un mensaje conforme todo es correcto y con formato de Bootstrap *success*.

Si el número de ciclos es superior al umbral e inferior al máximo recomendado, se muestra un mensaje de alerta conforme se debería ir programando un mantenimiento, se muestra con formato de Bootstrap *warning*.

Por último, si el número de ciclos es superior al máximo recomendado, se muestra un mensaje de alarma con formato de Bootstrap *danger*, indicando que, si no se hace algo, la válvula puede fallar en cualquier momento.

```
if ($row['active'] == true) {
    echo "<tr>";
    echo "<td colspan='7'>";
    if ($cicles < $threshold){
        echo "<div class='alert alert-success'>
            <p>Solenoid valve is inside the parameters
            advised by the manufacturer.</p>
            <p>No action is required.</p>
        </div>";

        } elseif ($cicles >= $threshold and $cicles <
$max_pulses){
        echo "<div class='alert alert-warning'>
            <h1 align='center'>Caution!</h1>
            <br>
            <p>Solenoid valve is close to meeting the
            maximum number of cycles recommended by the
            manufacturer. Consider scheduling
            maintenance.</p>
        </div>";

        } elseif ($cicles >= $max_pulses){
        echo "<div class='alert alert-danger'
align='center'>
            <h1 align='center'>Warning!</h1>
            <br>
            <p>Solenoid valve is over the maximum number
            of cycles recommended by the manufacturer.
            If not action is taken, it can fail at any
            time.</p>
        </div>";
    }
}
```

El mantenimiento predictivo, compararía si la electroválvula esta activada con el caudal que proporciona, de esta manera y una vez más, a través de condicionales *if/else*, tendríamos que:

- Si está desactivada y da caudal, la electroválvula tendría una fuga, por tanto, se enviaría una señal de alarma a través de un nuevo *<div>* con formato de Bootstrap *danger*.
- Si está activada y no da caudal, posiblemente la bobina esté dañada, al no ser algo tan crítico como el caso anterior, se enviaría también una señal de alarma con un *<div>*, pero esta vez con formato de Bootstrap *warning*.

```
echo "<tr>";
echo "<td colspan='7'>";
if ($state == 0 and $flow <> 0){
    echo "<div class='alert alert-danger'>
        <h1 align='center'>Warning!</h1>
        <br>
        <p>Solenoid valve 1 has a leakage.
        Intervention required</p>
    </div>";
} elseif ($state == 1 and $flow == 0){
    echo "<div class='alert alert-warning'>
        <h1 align='center'>Caution!</h1>
        <br>
        <p>Solenoid valve 1 is open and there is no
        flow. The coil could be damaged.</p>
    </div>";
}
echo "</td></tr>";
}
echo "</tbody>";
}
echo "</table>
<br>
";
```

La página se refresca cada 5 segundos o cada vez que el usuario pulsa en una de las opciones del menú izquierdo.

```
header ("Refresh:5");
```


7.6.2. Electrovalves

Esta página es únicamente para accionar manualmente las electroválvulas.

En teoría, no sería necesaria una vez instalado este sistema en el usuario, ya que las electroválvulas se accionarían a través del sistema automatizado que tengan en la instalación y únicamente se recogerían las señales de activación. Para este proyecto, no obstante, se hace necesario poderlas accionar manualmente y por eso se ha creado este apartado.

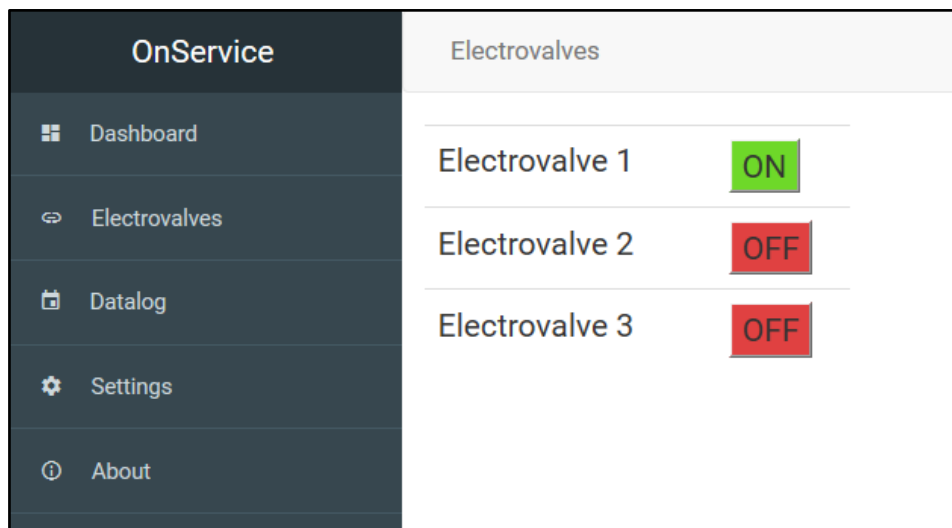


Figura 20 –Panel de mando de electroválvulas

Pulsando sobre el botón, se enviaría la señal a Arduino de abrir o cerrar la electroválvula. Una vez más, se conecta a la base de datos, se crea una consulta SQL que devuelve todos los registros de la tabla *state* (es decir, todas las electroválvulas) y se crea una tabla que contiene todos los registros con un botón de formulario (on/off) de tipo *post* que recoge el archivo *update_values.php*.

```
//Connect to database
$link = mysqli_connect('localhost','root','');
mysqli_select_db($link, 'maintenance') or die('Cannot select the
DB');

//We grab the table state out of the database
$result = mysqli_query($link,"SELECT * FROM state");//table select

//Now we create the table with all the values from the database
echo "<table class='table' style='font-size: 20px;'>
<tbody>
";
```

```

while($row = mysqli_fetch_array($result)) {
    echo "<tr>";
    $unit_id = $row['id'];
    echo "<td> Electrovalve " . $row['id'] . "</td>";
    $column1 = "state_EV1";
    $current_state = $row['state_EV1'];
    if($current_state == 1){
        $inv_current_state = 0;
        $text_current_state = "ON";
        $color_current_state = "#6ed829";
    } else {
        $inv_current_state = 1;
        $text_current_state = "OFF";
        $color_current_state = "#e04141";
    }
    echo "<td><form action= update_values.php method= 'post'>
    <input type='hidden' name='value2' value=$current_state
    size='10' >
    <input type='hidden' name='value'
    value=$inv_current_state size='10' >
    <input type='hidden' name='unit' value=$unit_id >
    <input type='hidden' name='column' value=$column1 >
    <input type= 'submit' name= 'change_but' style='
    margin-left: 5%; font-size: 20px; text-align:center;
    background-color: $color_current_state'
    value=$text_current_state></form></td></td>";
    echo "</tr>";
}
echo "</tbody></table>
<br>
";

```

El archivo *update_values.php* es el encargado de cambiar el estado en la tabla *state* para más tarde ser enviado de vuelta a Arduino, además incrementa el contador de maniobras de la electroválvula cuando el valor sea de activación.

El código es el siguiente:

```

<?php
//This file will get the values when you click any of the ON/OFF
buttons or change buttons on the electrovalves.php file
//We get that value and send it to the database table and by that
update the values
$value = $_POST['value']; //Get the value
$unit = $_POST['unit']; //Get the id if the unit where
we want to update the value
$column = $_POST['column']; //Which column of the database

//Connect to database
$link = mysqli_connect('localhost','root','');
mysqli_select_db($link, 'maintenance') or die('Cannot select the
DB');

$counter = $link->query("SELECT counter FROM state WHERE
id=$unit")->fetch_object()->counter;

```

```

//Now update the value sent from the post (ON/OFF, change or send
button)
mysqli_query($link,"UPDATE state SET $column = '{$value}' WHERE
id=$unit");

if ($value == 1) {
    $counter++;
    mysqli_query($link,"UPDATE state SET counter = '{$counter}'
WHERE id=$unit");
}

//go back to the interface
header("location: electrovalves.php");
?>

```

7.6.3. Datalog

Esta parte del servidor Web, permite analizar los datos obtenidos. Para mayor facilidad, en el caso de haber habido alguna incidencia, se marca en diferentes colores con las alertas que proporciona Bootstrap para *warning* (en el caso de válvula abierta y sin caudal) y *danger* (en el caso de válvula cerrada y con caudal.)

Electrovalve 1	Flow	Date / Time
OFF	0	2019-12-09 19:17:04
OFF	0	2019-12-09 19:16:48
OFF	0	2019-12-09 19:16:31
ON	0	2019-12-09 19:16:14
ON	0	2019-12-09 19:15:58
ON	0	2019-12-09 19:15:41
ON	0	2019-12-09 19:15:25
OFF	0	2019-12-09 19:15:08
OFF	0	2019-12-09 19:14:52
OFF	0	2019-12-09 19:14:35
OFF	16.53	2019-12-09 19:14:19
OFF	0	2019-12-09 19:14:02
OFF	0	2019-12-09 19:13:45
OFF	15.47	2019-12-09 19:13:29
OFF	0	2019-12-09 19:13:12

Figura 21 - Datalog

En este caso, se conecta primero la base de datos y se crea una consulta SQL que devuelve todos los registros de la tabla *valves* ordenados de forma descendente, de manera que el primer registro que se visualiza corresponde siempre a la última lectura.

Esta consulta se refresca cada 5 segundos si no se ha pulsado ninguna otra opción.

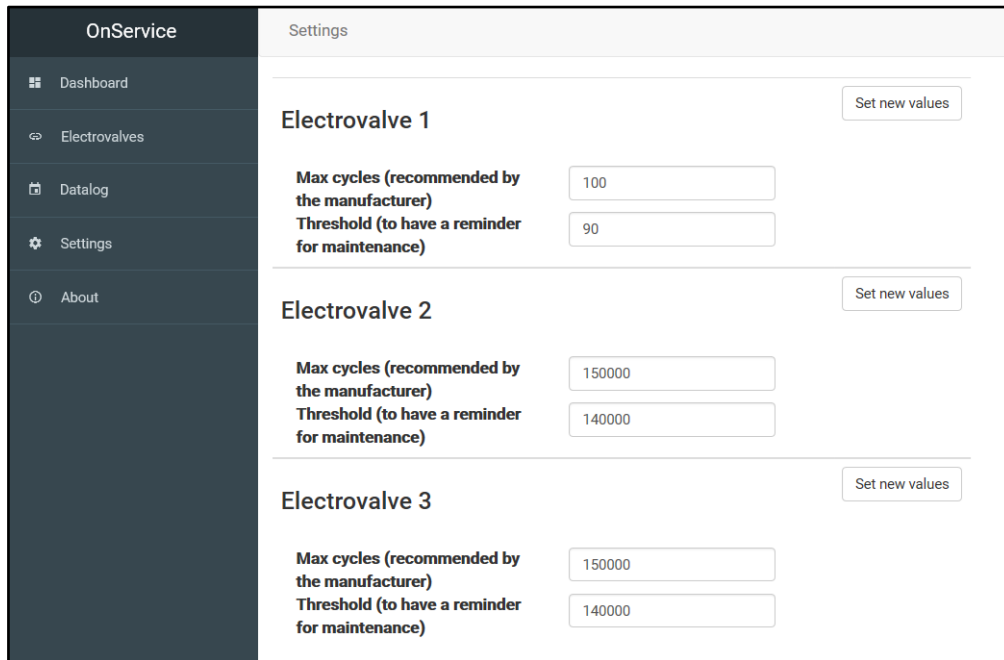
```
//Connect to database
$link = mysqli_connect('localhost','root','');
mysqli_select_db($link, 'maintenance') or die('Cannot select the
DB');

$result = mysqli_query($link, "SELECT valv_state, flow, date FROM
valves ORDER BY ID DESC");

if ($row = mysqli_fetch_array($result)){
    echo "<table class='table'> \n";
    echo "<thead><tr><th class='col-xs-2'><span
class='text'>Electrovalve 1</span></th>
<th class='col-xs-2'><span class='text'>Flow</span></th>
<th class='col-xs-2'><span class='text'>Date /
Time</span></th></tr></thead><tbody>\n";
    do {
        if ($row["valv_state"] == 0 and $row["flow"] <> 0){
            echo "<tr class='danger'>";
        } elseif ($row["valv_state"] == 1 and $row["flow"] ==
0){
            echo "<tr class='warning'>";
        }
        if ($row["valv_state"] == 0){
            echo "<td class='col-xs-2'
style='color:red'>OFF</td>";
        } elseif ($row["valv_state"] == 1){
            echo "<td class='col-xs-2'>ON</td>";
        }
        echo "<td class='col-xs-2'>".$row["flow"]."</td><td
class='col-xs-2'>".$row["date"]."</td></tr> \n";
    } while ($row = mysqli_fetch_array($result));
    echo "</tbody></table> \n";
} else {
    echo "No record found!";
}
header("Refresh:5");
```

7.6.4. Settings

Este apartado, permite introducir o modificar el número máximo de ciclos que establece el fabricante para cada modelo de electroválvula. Del mismo modo, es posible introducir un umbral a partir del que el sistema avisará de la necesidad de un mantenimiento.



The screenshot shows a web interface for 'OnService' with a sidebar menu containing 'Dashboard', 'Electrovalves', 'Datalog', 'Settings', and 'About'. The main content area is titled 'Settings' and displays configuration for three electrovalves. Each electrovalve has two input fields: 'Max cycles (recommended by the manufacturer)' and 'Threshold (to have a reminder for maintenance)', followed by a 'Set new values' button.

Electrovalve	Max cycles (recommended by the manufacturer)	Threshold (to have a reminder for maintenance)
Electrovalve 1	100	90
Electrovalve 2	150000	140000
Electrovalve 3	150000	140000

Figura 22 - Panel de ajustes

Una vez más, el primer paso es conectarse a la base de datos y crear una consulta que seleccione todos los registros de la tabla *state*.

Dentro del bucle *while*, se va creando una tabla, con un formulario en cada fila.

El botón es el encargado de hacer la actualización en la tabla mediante el uso de otra consulta SQL, esta vez de actualización.

```

//Connect to database
$link = mysqli_connect('localhost','root','');
mysqli_select_db($link, 'maintenance') or die('Cannot select the
DB');

//We grab the table state out of the database
$result = mysqli_query($link,"SELECT * FROM state");//table select

echo "<form class='form-inline' method='post' action=''>";
//Now we create the table with all the values from the database
echo "<table class='table'>
<tbody>
";
while($row = mysqli_fetch_array($result)) {
    $current_maxpulses = $row['max_pulses'];
    $threshold = $row['threshold'];
    echo "<tr>";
    $unit_id = $row['id'];
    echo "<td><h3>Electrovalve " . $row['id'] . "</h3>
<br>
<div>
        <label class='control-label col-sm-6'
        for='maxCycles'>Max cycles (recommended by the
        manufacturer) </label>
        <div class='col-sm-3'>
            <input class='form-control' id='maxCycles'
            type='text' name='max_cycles'
            value='". $current_maxpulses. "'>
        </div>
    </div>
<br><br>
<div>
        <label class='control-label col-sm-6'
        for='threshold'>Threshold (to have a reminder for
        maintenance)</label>
        <div class='col-sm-3'>
            <input class='form-control' id='threshold'
            type='text' name='threshold'
            value='". $threshold. "'>
        </div>
    </div>
</td>";
    echo "<td><input type='hidden' name='unit' value=$unit_id
><input class='btn btn-default' type='Submit' value='Set new
values' name='update'></td>";
    echo "</tr>
</tbody>";
}
echo "</table></form><br>";

if(isset($_POST['update'])){ //if the Update button is clicked
    $unit = $_POST['unit']; //Get the id of the unit
    where we want to update the value
    $max_cycles = $_POST['max_cycles'];
    $threshold = $_POST['threshold'];
    mysqli_query($link,"UPDATE state SET max_pulses =
    $max_cycles, threshold = $threshold WHERE id=$unit");
    header("Refresh:0");
}

```

8. RESULTADOS

Para las pruebas de funcionamiento se ha utilizado un compresor de aire conectado a la electroválvula + caudalímetro, la presión de entrada se ha tarado a 2 bar mediante un regulador ya que las pruebas se han hecho a escape.

El accionamiento de la electroválvula se ha hecho desde el servidor web, funcionando correctamente, aunque con un retardo de unos 7 segundos. Esto se debe a que el sistema debe esperar a que llegue la petición http y lea el estado de la base de datos, esa petición se genera cada 5 segundos.

Para emular una fuga, se ha utilizado un destornillador para abrir manualmente la electroválvula cuando está cerrada y se comprueba que efectivamente da una alarma a los pocos segundos.

En cuanto a la detección de fallo de bobina, se ha provocado el fallo desconectando una de las bornas, lo que produce que la electroválvula se cierre por acción de su resorte interno, al hacerlo, se acciona una alarma de posible daño en la bobina.

El 100% de las pruebas realizadas ha funcionado correctamente por lo que se puede afirmar que los objetivos se han cumplido.

9. CONCLUSIONES

En un inicio, el proyecto iba a consistir en un sistema de ayuda al mantenimiento para equipos industriales, mas tarde se decidió acotar únicamente al entorno de las electroválvulas debido principalmente al tiempo disponible.

Tras la finalización de este proyecto, se puede constatar que los principales objetivos tales como almacenar número de maniobras, detectar fallos o dar un aviso de mantenimiento en cada caso, se han cumplido.

El resultado final, ha sido un sistema que facilita en gran medida el mantenimiento de las electroválvulas en un equipo o planta industrial, que detecta fallos y se anticipa a las posibles averías.

Los conocimientos adquiridos a lo largo del grado y el minor realizado en programación avanzada web, me han sido de gran utilidad en todo el proyecto, aplicando muchas de las técnicas aprendidas en diferentes partes del trabajo. La investigación, montaje y programación de los diferentes componentes del entorno Arduino, han supuesto un amplio aprendizaje personal que seguro podré aplicar en nuevos proyectos.

9.1. Trabajos futuros

Al realizar el proyecto, han surgido nuevas ideas, unas veces debido a las búsquedas realizadas, otras por solucionar inconvenientes que han ido surgiendo. Muchas de ellas no se han podido aplicar en el proyecto por falta de tiempo. Algunas de las mejoras que se podrían incluir, serian:

- En lugar de usar un relé conectado por cable, usar uno con el módulo ESP8266 integrado para conectarse a la red de forma autónoma. Seria lógico intentar eliminar el cableado entre cada electroválvula y Arduino.
- Considerar sustituir módulos autónomos ESP8266 en lugar de Arduino como placa (no como entorno), puesto que la comunicación podría ser a más velocidad y disminuiría el costo sensiblemente.
- Comprobar el comportamiento del sistema con un número significativo de electroválvulas conectadas y enviando información simultáneamente.
- Ampliar la aplicación a otro tipo de componentes, por ejemplo, motores, donde se podría medir el tiempo de uso y su temperatura para predecir posibles fallos.

10. BIBLIOGRAFIA

- [1] <https://leanmanufacturing10.com/mantenimiento-correctivo-preventivo-y-predictivo-definiciones-y-diferencias>
- [2] <https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno>
- [3] <https://www.indeed.es/cmp/Randstad/salaries/técnico-electronico/Barcelona-CT>
- [4] <https://store.arduino.cc/arduino-uno-rev3>
- [5] <https://designedbyashw.in/blog/blog/2018/07/11/programming-the-esp-01/>
- [6] <https://fritzing.org/home/>
- [7] https://www.espressif.com/en/support/download/overview?keys=&field_type_tid%5B%5D=14
- [8] <https://www.luisllamas.es/caudal-consumo-de-agua-con-arduino-y-caudalimetro/>