Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

"Aplicación de la tecnología WEB en el desarrollo de un sistema SCADA"

Autor: Guarirai Tomé Noda

Tutor: Alberto Francisco Hamilton Castro

La Laguna, a 13 de Julio de 2023

Resumen

Actualmente asistimos a un proceso de transformación en el que todas las tendencias apuntan hacia la digitalización. Los entornos web han experimentado un notable desarrollo y se han vuelto más atractivos para los usuarios en comparación con años anteriores. Las páginas web nos permiten tener el mundo al alcance de la mano. Por ello, hemos llevado esta nueva forma de vida al entorno industrial, trabajando con todas estas tecnologías con el objetivo de crear un sistema SCADA en formato de página web.

La creación de este sistema no sería posible si no pudiéramos interactuar con el mundo físico y real, es decir, si no pudiéramos realizar cambios en los componentes que queremos controlar. Para ello, hemos seleccionado la placa de desarrollo BeagleBone Black, que actúa como servidor de páginas web y se comunica con los elementos cuyo estado deseamos modificar.

A lo largo de este proyecto se ha ido evolucionando desde prototipos simples, elevando poco a poco su dificultad, hasta lograr el objetivo de comprobar que se pueden usar páginas web como sistemas SCADA.

Palabras clave: “SCADA”, “Tecnologías WEB”, “BeagleBone Black”

Abstract

We are currently witnessing a transformation process in which all trends point towards digitalization. Web environments have undergone a remarkable development and have become more attractive to users compared to previous years. Websites allow us to have the world at our fingertips. Because of this, we have brought this new way of life to the industrial environment, working with all these technologies with the aim of creating a SCADA system in web page format.

The creation of this system would not be possible if we could not interact with the physical and real world, that is, if we could not make changes in the components we want to control. To achieve this, we have selected the BeagleBone Black development board, which acts as the web page server and communicates with the elements whose status we wish to change.

Throughout this project we have evolved from simple prototypes, gradually increasing their difficulty, until we have achieved the goal of proving that web pages can be used as SCADA systems.

Keywords: "SCADA", "WEB Technologies", "BeagleBone Black".

Índice

Índice de figuras

Índice de códigos

Índice de tablas

Capítulo 1: Introducción

1.1. Motivación y antecedentes

En la era digital en la que nos encontramos, la tecnología web ha revolucionado la forma en que interactuamos con el mundo que nos rodea. En este contexto, el desarrollo de un sistema SCADA (Supervisory Control and Data Acquisition, por sus siglas en inglés) ha encontrado en la aplicación de la tecnología web un aliado clave para su eficiente funcionamiento.

El sistema SCADA es ampliamente utilizado en diversos sectores industriales para recopilar y supervisar datos en tiempo real, permitiendo una gestión eficiente de procesos y facilitando la toma de decisiones. La utilización de la tecnología web en el desarrollo de este sistema, ha permitido la integración de diversos dispositivos y la accesibilidad remota a la información, transformando la forma en que se monitorean y controlan los procesos industriales. En este sentido, resulta de vital importancia explorar y comprender el papel que desempeña la tecnología web en este tipo de sistema, así como sus ventajas y desafíos, con el objetivo de maximizar su potencial y aprovechar al máximo sus beneficios en la gestión de procesos industriales.

Actualmente existen diversos sistemas SCADA en nuestro día a día utilizados tradicionalmente en diversos sectores industriales para supervisar y controlar procesos. Estos sistemas consisten en una red de dispositivos conectados mediante cables, sensores, controladores y software que permiten recopilar datos de los equipos y enviar comandos de control desde una ubicación central.

En cambio, los sistemas SACADA web son más adaptativos, permitiendo el uso de las comunicaciones por métodos más avanzados como MQTT y no necesitan de un equipo fijo en un lugar, se puede utilizar cualquiera mientras esté dentro de la red local para conectarse al servidor.

1.2. Objetivos

El objetivo final de este trabajo es integrar las tecnologías utilizadas en el desarrollo web en el ámbito industrial. En la actualidad, existen páginas web con usos muy diversos, desde informativas hasta interactivas, de ocio o divulgativas. Por esta razón, se considera que también se puede incluir a las páginas web de uso industrial.

El propósito de este proyecto no es controlar una planta, sino comprobar la viabilidad de utilizar páginas web como sistemas SCADA. Se comenzará con maquetas simples para el diseño, realizando pruebas en circuitos básicos que incluyen LEDs, un potenciómetro y un botón. A medida que los resultados de las pruebas sean

satisfactorios, se irá incrementando gradualmente la complejidad hasta llegar a crear un SCADA más completo, enfocado en un ejemplo de una planta industrial.

Para llevar a cabo estas pruebas, se ha seleccionado un módulo educativo llamado "38-001 SYSTEM Level & Flow control", que permite aprender sobre diferentes sensores y aplicar distintos tipos de control.

En este sentido, se desarrollará un sistema SCADA para esta planta industrial, utilizando como hardware una microcomputadora llamada BeagleBone Black. Para mejorar y darle funcionalidad a este sistema, se implementará varios controles On-Off y un control PID.

Capítulo 2: Conocimientos Previos

Para facilitar el entendimiento de este proyecto presentaremos los conocimientos y herramientas básicas que han servido de base para su elaboración.

2.1. SCADA

2.1.1. Interfaz Humano-Máquina

Una interfaz Humano-Máquina o también conocido por sus siglas en inglés HMI ("Human Machine Interface") es parte de un software operativo encargado de presentar los datos a un operador humano para que éste supervise y controle un proceso de forma remota y en tiempo real [1].

Los sistemas HMI son encargados de representar la información necesaria para que el operador pueda realizar correctamente su función. Dicha función es de extremada delicadeza, pues no conviene escasear en la información para no perder detalle de lo que sucede, ni saturar de esta, para que su manejo no resulte agobiante.

Para que la información recogida sea de utilidad se necesita de su aportación en tiempo real. Es de utilidad contar con elementos visuales, donde poder saber la evolución temporal del proceso. Ejemplos prácticos son gráficos que se actualicen según reciben la información, pequeñas animaciones del sistema o incluso señales lumínicas.

Figura 1. Interfaz Humano-Máquina [2].

2.1.2. Sistema SCADA

En pleno siglo XXI a causa de la gran evolución tecnológica y la escasez de recursos para crear nuevos aparatos electrónicos la informática a desarrollado una nueva forma versátil de poder manipular elementos físicos dejando obsoletas a las botoneras. Esta nueva forma de manipular el entorno se le conoce como sistema SCADA *Supervisory Control And Data Acquisition*, que en español sería Supervisión, Control y Adquisición de Datos. Este sistema se caracteriza por una mayor flexibilidad a la hora de ser implementada y ahorra dinero reduciendo los gastos materiales.

Trata de un software capaz de ser adaptado a cualquier proceso industrial, permite la visualización y control adquiriendo y transmitiendo información en tiempo real desde un ordenador. Dotando al sistema de ciertas capacidades telemáticas a su usuario.

Este sistema tiene como principales componentes [3].

- **Interfaz HMI:** Previamente comentado, es un espacio visual que ofrece el software para que el operario actúe con la planta.
- **Unidades terminales remotas (RTU – remote Terminal Unit):** Componente que se encarga de la adquisición y envio de datos.
- **Unidades terminales maestras (MTU – Master Terminal Unit):** Es el subsistema que centralizado los datos y ejecuta el envio de instrucciones.
- **Red de comunicaciones:** Estructura necesaria para mantener todos los dispositivos conectados entre sí.
- **Sensores:** Encargados de obtener la información para su posterior procesado.
- **Actuadores:** Encargados de manipular el proceso tras recibir órdenes.
- **Base de Datos:** Para almacenar datos y consultarlos.

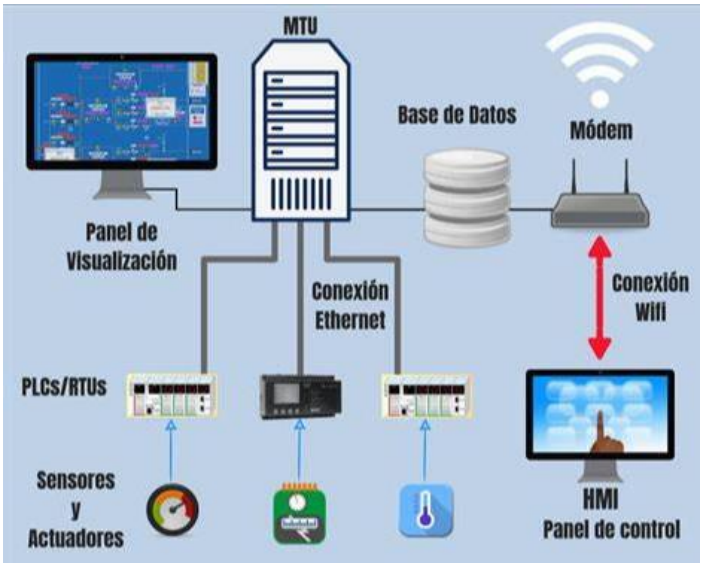


Figura 2. Componentes de un sistema SCADA [4].

2.2. Página Web

El presente trabajo se ha centrado en la utilización de los lenguajes web para el desarrollo del sistema SCADA. Para diseñarlo se contará con herramientas que sean fáciles de manejar, con un coste reducido y modificable en el futuro.

En el desarrollo web distinguir dos términos [5].

- **Frontend:** Es Interfaz con el cliente, elementos visibles para el usuario desarrollados con HTML, CSS y JavaScript. Y ejecuta en navegador web del usuario en cualquier dispositivo que lo soporte (ordenador, móvil, tablet, televisor inteligente, ...).

- **Backend:** Encargado de procesar la información desde el lado del servidor para que el frontend opere correctamente.

Figura 3. Frontend y Backend [6].

2.2.1. Estructura básica

A la hora de crear una página web es de buena práctica contar con la siguiente estructura para asegurar un orden dentro de esta.

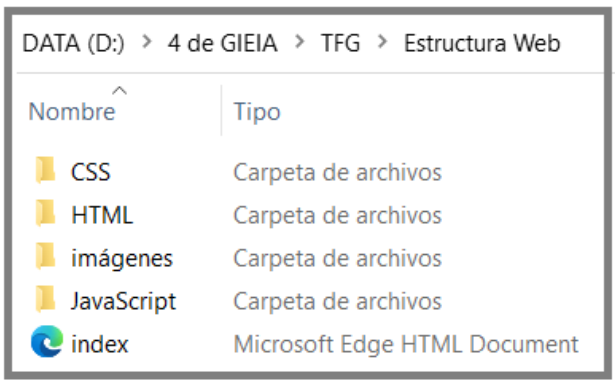


Figura 4. Estructura de una página web.

En esta organización se necesita de varias carpetas para diferenciar entre archivos CSS, JavaScript, HTML y las imágenes que se fueran a usar. También es importante establecer un archivo "index.html" fuera de las carpetas, el cual se encargará de conectar toda la configuración tomando el papel tanto de índice como de página principal.

2.2.2. Notepad ++

Para desarrollar en una primera instancia los lenguajes que serán nombrados a continuación será utilizado el programa Notepad ++, un editor de texto y código capaz de soportar varios lenguajes y de fuente libre.

Este programa ha sido elegido por su edición sencilla, fácil entendimiento y comodidad de edición.

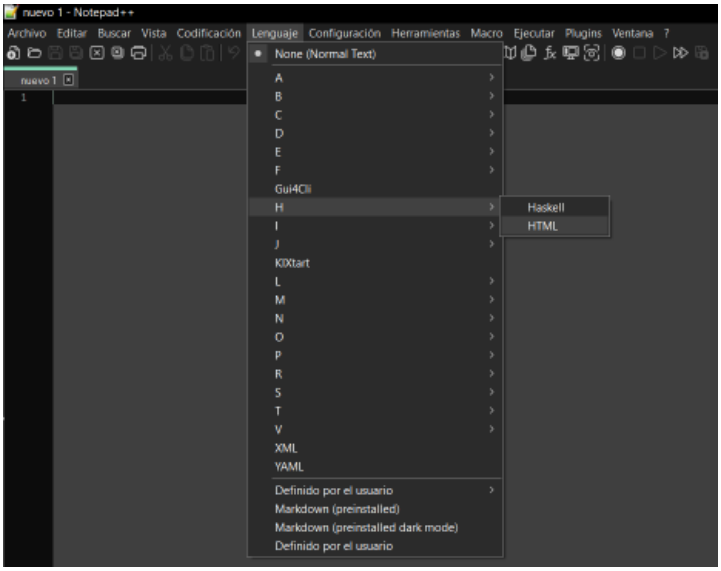


Figura 5. Entorno de notepad++.

2.2.3. HTML

El lenguaje de marcado Hyper Text Markup Language o más conocido como HTML, forma el esqueleto de cualquier página web. Nació de la necesidad de añadir formato de texto escrito. No se le puede confundir como un lenguaje de programación ya que carece de estructuras lógicas.

El HTML consiste en un conjunto de etiquetas que sirven para definir el texto y otros elementos que componen la página web. Dichas etiquetas tienen la siguiente forma *"<A>"*. Cada etiqueta tiene su significado desde *"<text>"* que se utiliza para insertar texto hasta *" "*, utilizada para dividir la página en porciones ajustables útiles para organizar el contenido. Casi todas las etiquetas tienen su correspondiente etiqueta de cierre. Por ejemplo *"</text>"*, cerraría la etiqueta *"<text>"* y delimitaría su influencia en el entorno. Definiendo de esta forma su escritura: *"<text> Hola mundo </text>" [7].*

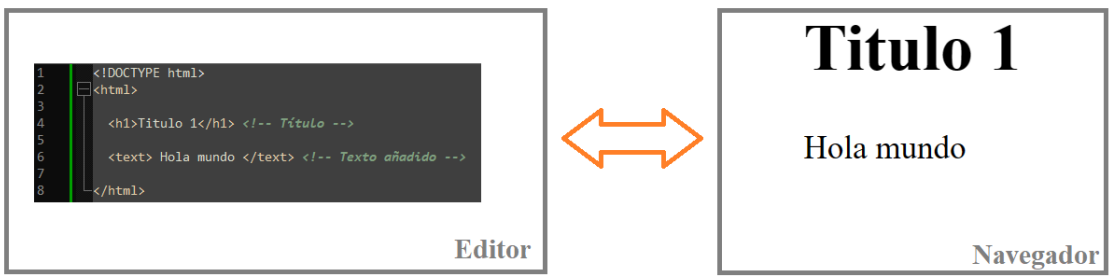


Figura 6. Comparación Editor-Navegador.

En el HTML podemos distinguir 2 etiquetas básicas [8].

- **<head> </head>:** Es una etiqueta que hace de cabecera donde podemos poner información no visible para el usuario como scripts, enlaces o librerías. También se puede usar para editar la pestaña de la página.
- **<body> </body>:** Esta etiqueta como su nombre indica es el cuerpo de la estructura en el que situaremos y organizaremos los elementos que se mostrarán en la página. Dentro del body podemos distinguir las etiquetas esenciales para dar una estructura simple a nuestra web.
    - *<header> </header>:* Para crear una cabecera donde ilustrar el título y algún logotipo.
    - *<nav> </nav>:* Para definir un menú de navegación.
    - *<section> </section>:* Donde colocaremos el contenido principal ya sea texto o imágenes. También la podemos sustituir por una etiqueta div para mejorar la distribución.
    - *<footer> </footer>:* Para hacer un pie de página.

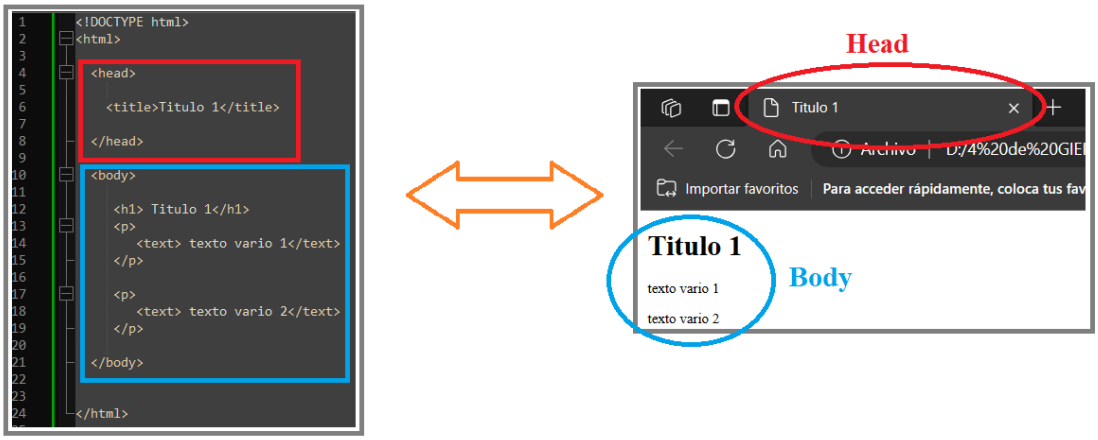


Figura 7. Head y Body.

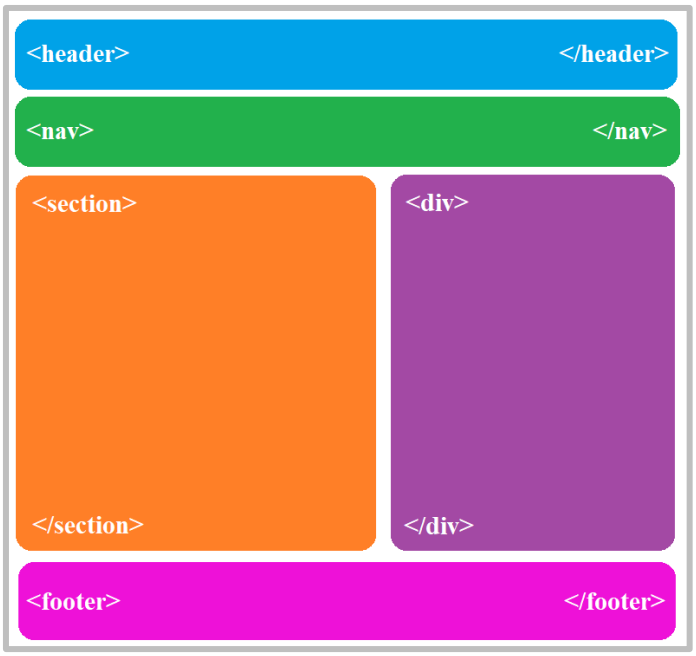


Figura 8. Cuerpo básico de una página web.

2.2.4. CSS

Existen varias formas de dar estilos a las páginas web. Una es mediante el uso de la etiqueta "*<style> </style>*". Dentro de esta etiqueta se puede acceder a los distintos elementos para darles color, un fondo, cambiar el tamaño o añadir márgenes.

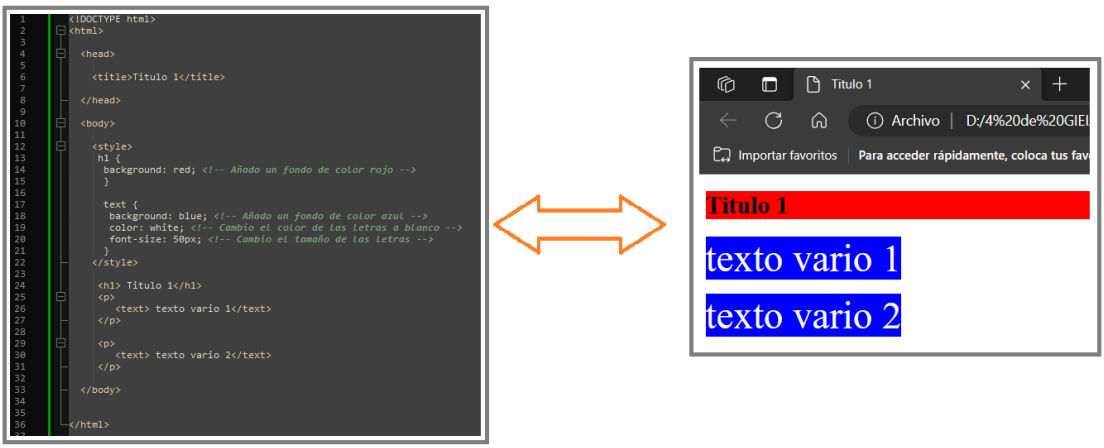


Figura 9. Uso de la etiqueta <style>.

Esta forma de cambiar el aspecto es útil para páginas muy simples porque como se puede ver, cuantos más cambios se quieran realizar el archivo, este va quedando muy engorroso y dificulta su entendimiento.

La mejor forma de aplicar estilos a una página es mediante el uso de un fichero aparte de CSS. El CSS o Cascading Style Sheets permite añadir vida a nuestro esqueleto HTML. Es decir, se puede modificar la página web para un aspecto visual más

agradable para el usuario incorporando colores o animaciones. Para ello se creará el archivo y añadirá de la siguiente forma como se muestra en la Figura 10. De esta manera obtenemos un código más limpio y ordenado.

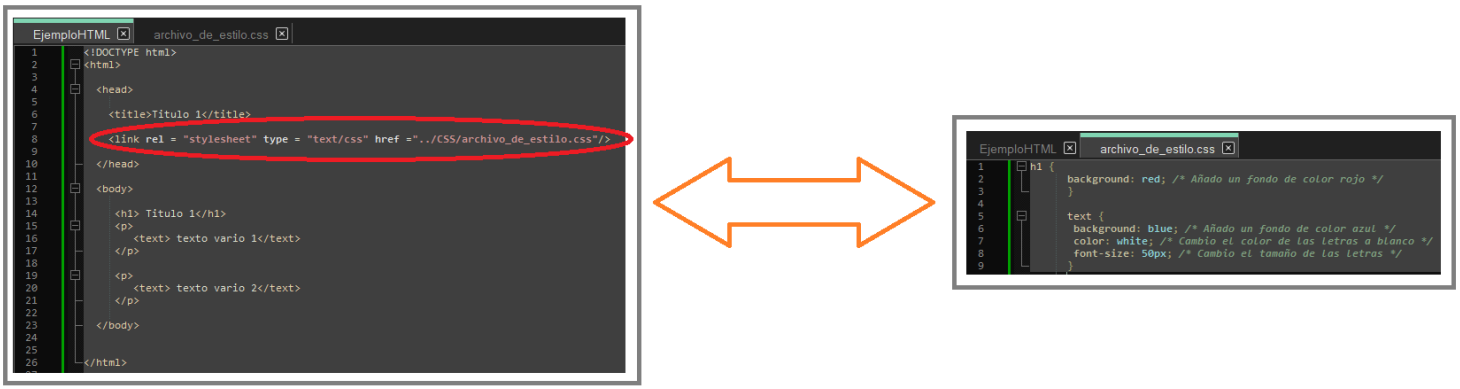


Figura 10. Crear archivo CSS.

Una vez creado e incorporado el archivo lo siguiente es saber como diferenciar entre elementos del mismo tipo, ya que está la posibilidad de querer dar formato distinto a varios títulos o querer posicionar en distintos lugares ciertos bloques que comparten la misma etiqueta. Con este fin se le pueden añadir "clases" a las etiquetas.

Hay que tener en cuenta que si varias elementos comparten la misma clase todos se verán afectados por los mismo cambios.

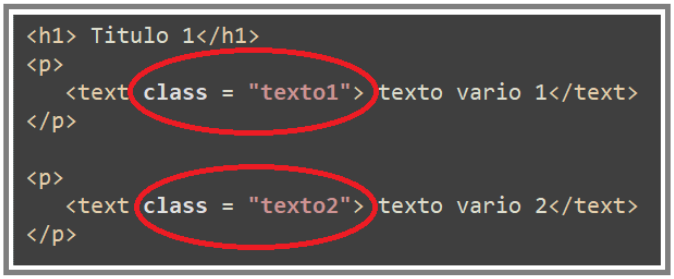


Figura 11. Como añadir clases.

Para decir que vamos a modificar una etiqueta mediante su clase solo hay que añadir un "." al nombre que se le ha asignado. Como por ejemplo ".texto1".

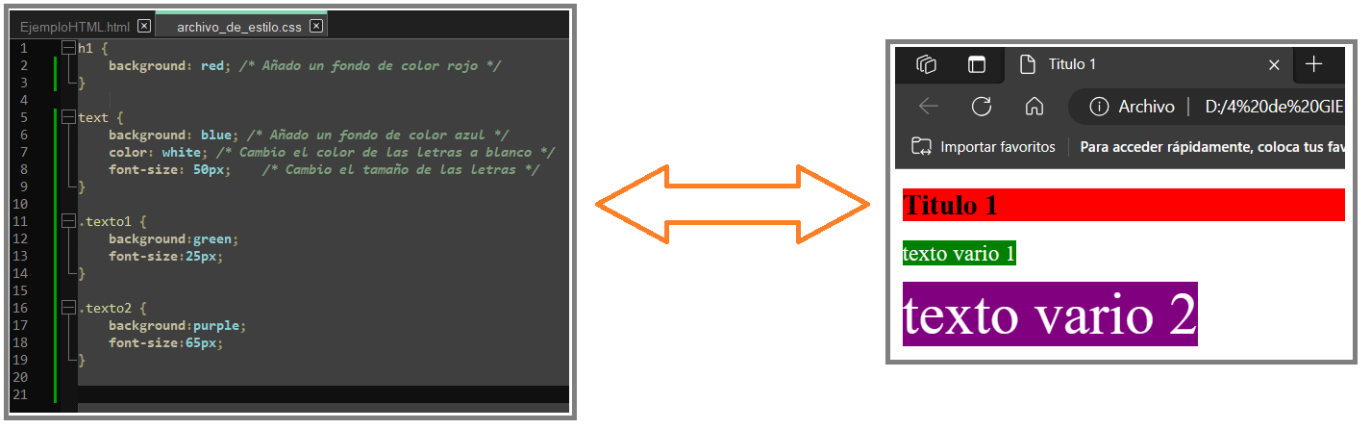


Figura 12. Uso de clases.

Por otro lado, si solo queremos modificar un elemento en concreto se puede añadir un identificador "id" con el fin de diferenciar un elemento del resto.

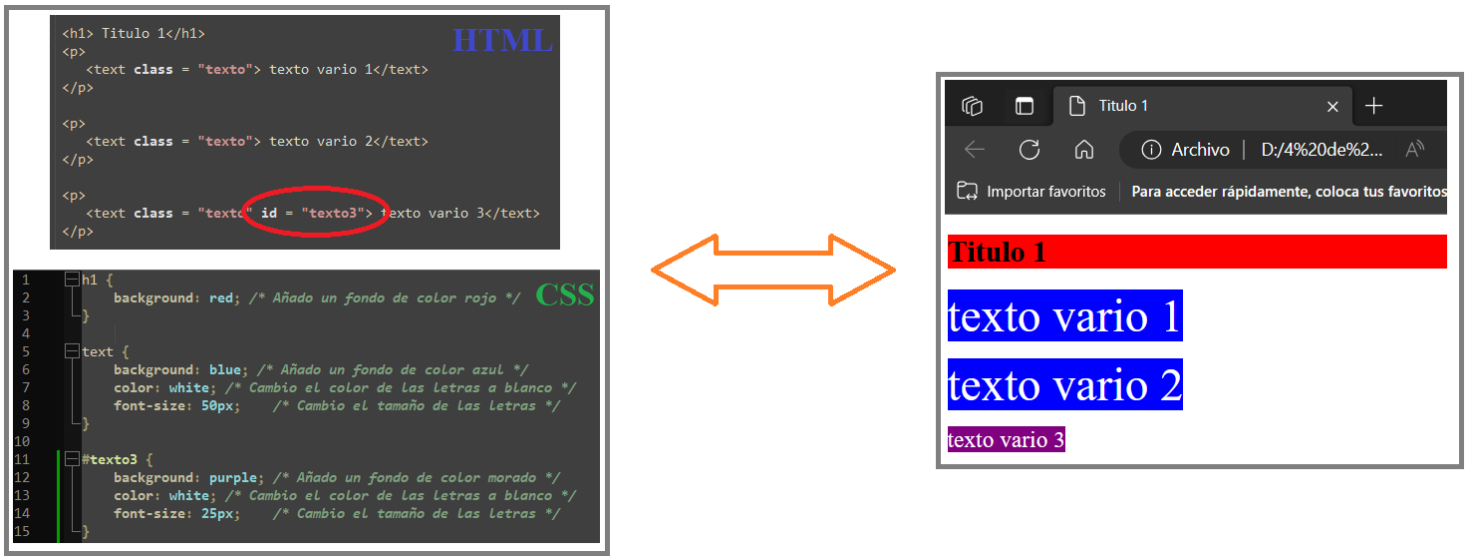


Figura 13. Uso del identificador.

Como podemos observar a pesar de que todos los elementos tienen la clase "texto" se ha modificado el último añadiendo un id "texto3" para diferenciarlo del resto.

2.2.5. JavaScript

El JavaScript es un lenguaje de programación orientada a objetos y multiplataforma que permite implementar funciones complejas en las páginas web, con la finalidad de añadir o modificar los elementos de esta para incorporar dinámicas o interacciones con el usuario [9]. De tal forma que se puede hacer un programa que espere a que el usuario haga clic en algún elemento para ejecutarse.

Dichas interacciones se pueden añadir al archivo HTML mediante la etiqueta "<script></script>", una vez escrita se podrá programar este lenguaje dentro de ella. Pero la mejor forma para la utilización de esta etiqueta es mediante la incorporación del atributo "src" [10], el cual nos permite añadir una dirección de un archivo externo.

```
<!DOCTYPE html>
<html>

  <head>

    <title>Titulo 1</title>

    <link rel = "stylesheet" type = "text/css" href ="../CSS/archivo_de_estilo.css"/>

  </head>

  <body>

     <h1> Titulo 1</h1>
     <p>
        <text class = "texto1"> texto vario 1</text>
     </p>

     <p>
        <text class = "texto2"> texto vario 2</text>
     </p>

     <script src = "../JavaScript/myScript.js"></script>
  </body>

</html>
```

Figura 14. Como incluir scripts.

Es recomendable incluirlos al final del fichero, de este modo se puede asegurar que se ha cargado todo el contenido antes de empezar a manipularlo.

El JavaScript además de ejecutarse en los navegadores, se puede ejecutar en el backend gracias al NodeJS lo que aporta que se pueda usar el mismo lenguaje de programación en el frontend como en el backen

2.2.6. Árbol DOM

Las siglas DOM significan *Document Object Model*, haciendo referencia a la propia estructura del documento HTML. En la creación de una página web es inevitable que se termine anidando una etiqueta dentro de otra, formando un árbol donde están todas relacionadas entre sí [11].

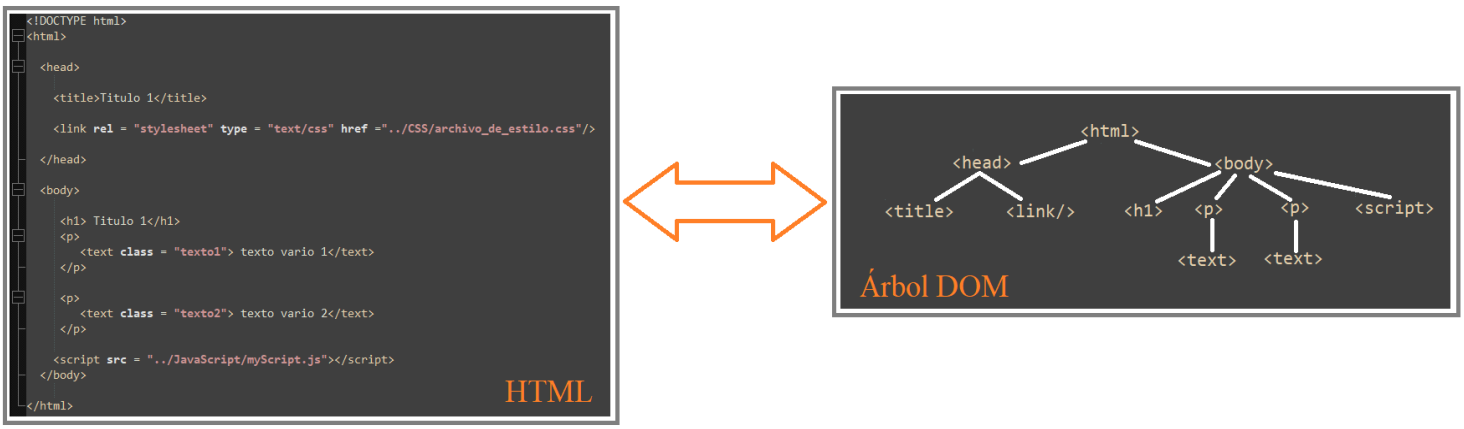


Figura 15. Árbol DOM.

En JavaScript se puede hablar de Árbol DOM cuando se accede a ella para manipular las diversas etiquetas, añadir nuevas o eliminarlas.

2.3. SVG

En las páginas web se suelen incorporar archivos visuales tales como gráficos o imágenes tanto *".png"* como ".jpg". Pero para este proyecto se ha decidido utilizar el formato SVG (*Scalable Vector Graphics*). Este formato basado en XML (usa la misma idea que el HTML: etiquetas, elementos, atributos) sirve para crear gráficos vectoriales en 2D que incluyen las siguientes ventajas [12].

- **Escalabilidad:** A diferencia de las imágenes compuestas por pixeles los SVG mantienen siempre su resolución.
- **Tamaño pequeño:** Ocupan poco espacio de almacenamiento.
- **Flexibilidad:** Permiten modificaciones en sus colores, posición y tamaño.
- **Compatibilidad:** Todos los exploradores actuales son capaces de reconocerlos.

En el desarrollo de este proyecto se ha necesitado de la creación de elementos para favorecer la interactividad entre el usuario y la web. Y para diseñar esos elementos se ha recurrido al editor Inkscape [Anexo II]. El cual nos permite definir identificadores en las formas que componen la imagen para facilitar su futura manipulación.

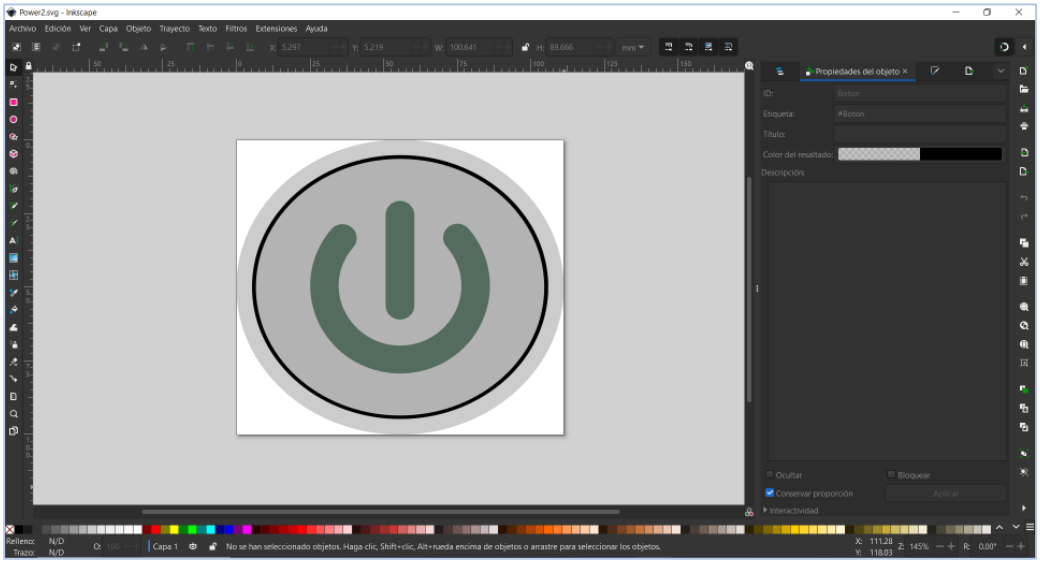


Figura 16. Entorno de Inkscape.

2.4. BeagleBone Black (BBB)

El elemento fundamental de este proyecto es el miniordenador BeagleBone Black, el cual es una placa de desarrollo basada en el procesador AM335x Arm Cortex-A8. Esta placa BeagleBone es un ordenador de una sola placa (SBC) que se puede utilizar como ordenador autónomo o integrado en un sistema.

Esta placa de ordenador de bajo coste y de código abierto basada en Linux es ideal para la interacción con dispositivos físicos.

En la Tabla 1 se muestran las principales características.

Tabla 1. Características BegaleBone Black.

Características	
Procesador	AM3358BZCZ100 ARM Cortex-A8 de 1 GHz
Memoria	Flash eMMC de 4 GB, DDR3 de 512 MB, EEPROM de 4 k
Gráficos	SGX530 3D, 20M Polygons/S
Almacenamiento	Tarjeta microSD, MMC de 3,3 V
PMIC	TPS65217C
USB	1 host USB A 2.0, 1 cliente miniUSB 2.0
Ethernet	Conector RJ45 10/100
Audio	Mini HDMI
GPIO	2 conectores de 46 contactos
Depuración	Puerto de depuración serie CTI JTAG de 20 contactos
Indicadores LED	1 de encendido, 2 de Ethernet, 4 controlados por el usuario
Botones	Reinicio, arranque, encendido
Alimentación	Mini USB o conector jack de 5 V dc

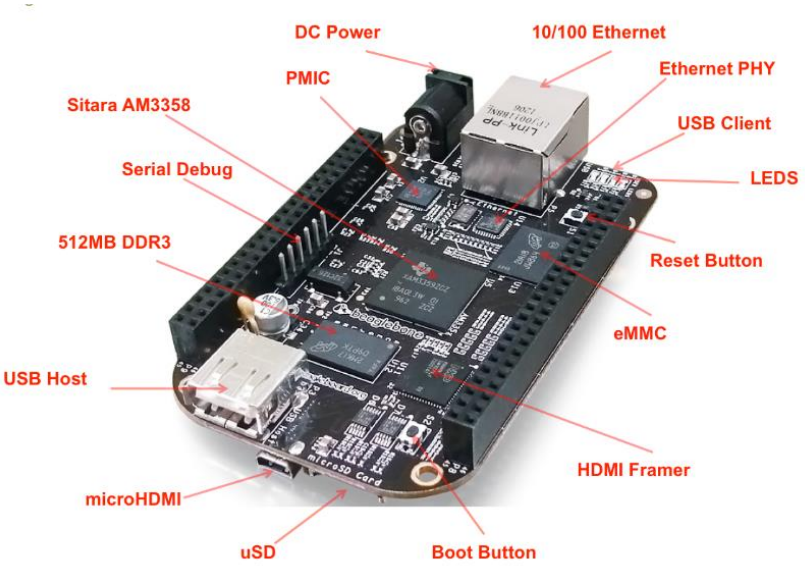


Figura 17. BeagleBone Black [14].

2.4.1. Pines BeagleBone Black

La gestión de entradas y salidas de la BBB se debe tener claro que los pines se pueden diferenciar entre aquellos que son de uso analógico o digital divididas en 2 columnas, P9 y P8. Estos vienen agrupados como GPIO, General Purpose Input Otput, que hacen referencia a las entradas y salidas digitales, entradas analógicas y salidas PWM.

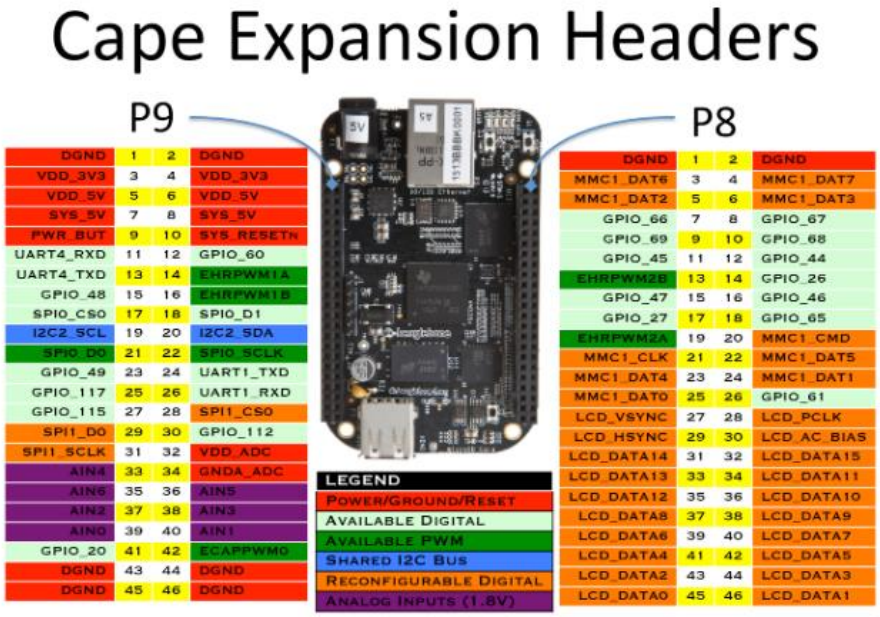


Figura 18. Pines BBB [15].

2.4.2. BoneScript

BoneScript es una librería desarrollada en Node.js que permiten la interacción con los pines de la placa. Las funciones más usadas son las siguientes.

- **pinMode():** Configura el modo de un pin, es decir, si se va a usar como entrada o salida digital.
- **digiatalWrite():** Activa o desactiva un pin de salida digital.
- **digitalRead():** Lee el estado de un pin de entrada digital.
- **analogRead():** Lee el voltaje de un pin de entrada analógica.
- **analogWrite():** Muestra una señal modulada por ancho de pulso en un pin de salida PWM.

2.4.3. Cloud 9 IDE

La BeagleBone Black su propio servidor web privado desarrollado en Node.js [13]. Es decir, podemos acceder a un entorno controlado multiplataforma para desarrollar, ejecutar y depurar programas más fácilmente, llamado Cloud 9.

Este entorno proporciona la posibilidad de aplicar multitud de lenguajes de programación donde están incluidos C, C ++, PHP, Python, HTML, JavaScript con Node.js, etc.

Para acceder a Clou9 desde la placa de desarrollo existen varias posibilidades, en este proyecto se han usado las dos siguientes.

- Conectar al ordenador a través del cable Mini USB, una vez inicializada puede accederse a la dirección IP de Beaglebone Black.
- Conectar mediante un cable de ethernet a la red y buscar la IP mediante comandos en la consola o algún programa externo. En este caso debemos conectar la placa a una fuente de alimentación.

Además de Cloud9 se puede acceder a varias páginas que residen en el servidor y que ofrecen ejemplos de códigos simples.

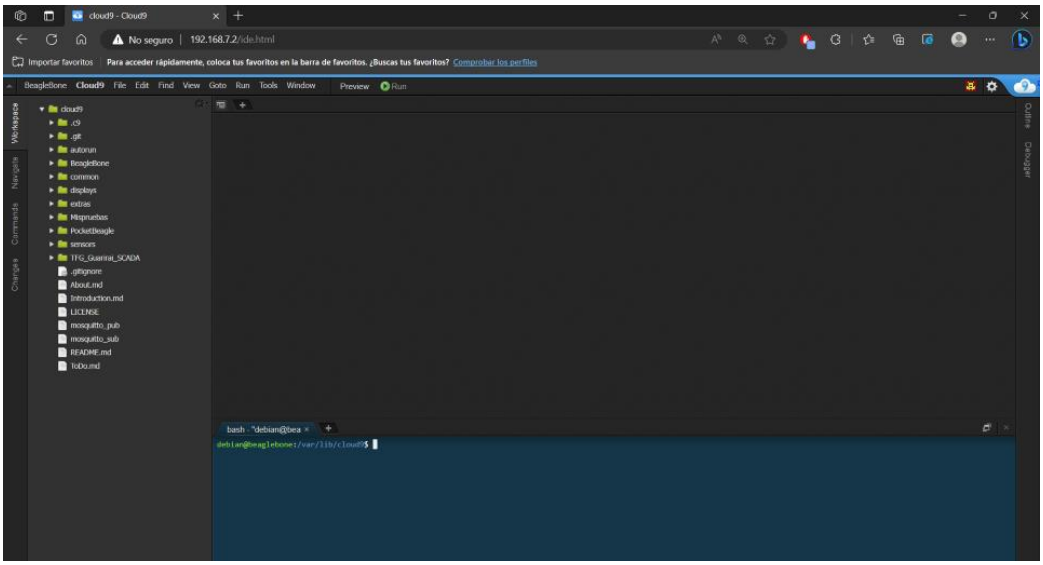


Figura 19. Entorno de trabajo Cloud 9 IDE.

Un servidor web almacena los archivos que forman una página web, el index, el archivo CSS, el archivo JavaScript y las posibles imágenes que fuera a contener, también transmite los datos según solicite el navegador.

Los servidores son el motor de Internet, son los encargados de generar el tráfico de información. Para establecer correctamente la transferencia de información, todo dispositivo conectado a la red debe de estar identificado mediante una IP única, con el fin de hacer distinción entre todos los dispositivos que se encuentren conectados.

Una vez se conecta la BBB al ordenador este asigna una de las siguientes IPs en función del como se conecte.

IP Address	Connection Type	Operating System(s)
192.168.7.2	USB	Windows
192.168.6.2	USB	Mac OS X, Linux
192.168.8.1	WiFi	all
beaglebone.local	all	mDNS enabled
beaglebone-2.local	all	mDNS enabled

Figura 20. IPs según el tipo de conexión.

2.5. Comunicaciones mediante MQTT

Las comunicaciones son una de las piezas clave para este proyecto, sin ellas no se podría establecer la conexión entre el backend y el frontend.

A fin de efectuar dicha conexión se ha decidido hacer uso de MQTT (*Transporte de telemetría de Message Queue Server*). El MQTT es un novedoso sistema que trata de un protocolo de mensajería ligero basado en la programación orientada a eventos. Se utiliza principalmente para el trámite de información de máquina a máquina [16] [17].

En la arquitectura MQTT existen dos tipos de sistemas [16].

- **Brokers:** El servidor que se comunica con los clientes.
- **Clientes:** Ente que envía o recibe información siempre a través de un broker.

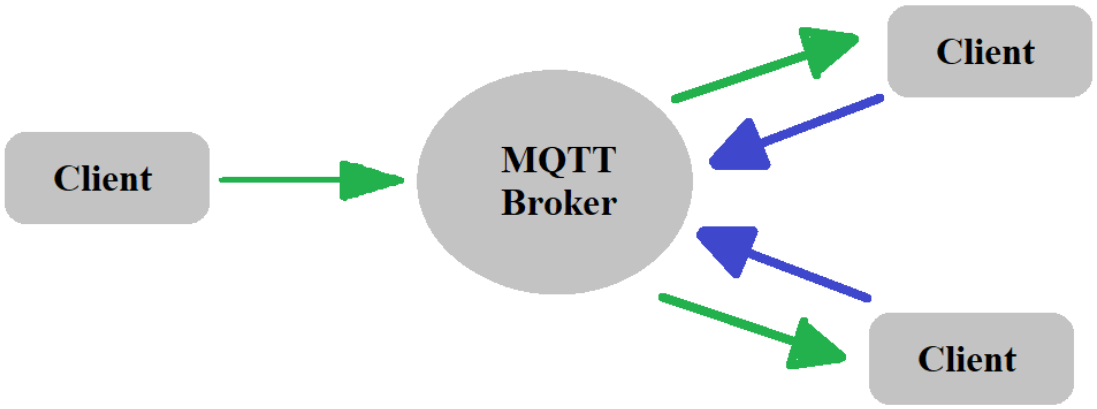


Figura 21. MQTT

Para que el protocolo sea lo más óptimo posible, solo se pueden realizar 4 acciones: publicar, suscribirse, cancelar la suscripción y hacer ping (asegurarse de que la conexión sigue funcionando).

El centro de comunicaciones transcurre en los topics o temas en los que los clientes pueden suscribirse, son algo así como los chats a los que uno puede unirse. Los Topics están formados por uno o más "niveles" separados entre sí por una barra inclinada '/'. Cada nivel debe está formado por uno o más caracteres distinguiente entre mayúsculas y minúsculas [18].

Figura 22. Niveles de topics [18].

Es importante destacar que los navegadores no tienen soporte MQTT incorporado, por lo que se deberá usar un cliente JavaScript MQTT con Websockets. Añadiendo así canales de comunicación dúplex completo a través de una única conexión TCP/IP [19].

2.5.1. Librerías

2.5.1.1. Eclipse Mosquitto

Para completar las comunicaciones se han usado varias librerías MQTT en función de las necesidades.

En primera instancia se ha recurrido al broker de Eclipse Mosquitto, un agente de mensajes de código abierto famoso por sus dos funciones simples para comprobar su funcionamiento.

- **mosquitto_sub:** Crea un cliente que se suscribirá a los topics (temas) y los imprimirá por pantalla. Dicha función tiene dos parámetros principales, “-h” para indicar el servidor al que se va a conectar y “-t”, para indicar el tema del que se quiere recibir los mensajes. Una vez activada, la función se queda a la espera hasta que llega el mensaje.

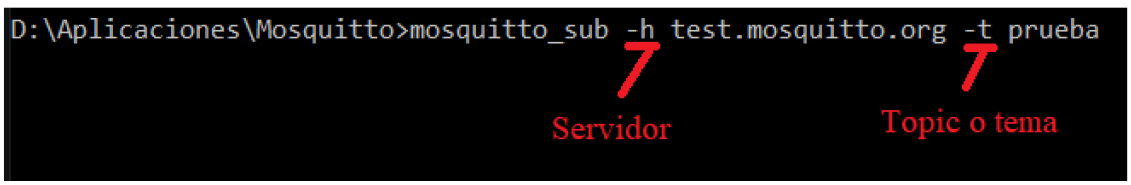


Figura 23. Función mosquitto_sub.

Para este ejemplo se está usando "test.mosquitto.org" como servidor disponible públicamente proporcionado por Eclipse Fundation.

- **mosquitto_pub:** Crea un cliente que publica un solo mensaje sobre un topic o tema. Esta función tiene tres parámetros principales "-h", "-t", explicados anteriormente y "-m" para escribir el mensaje deseado.

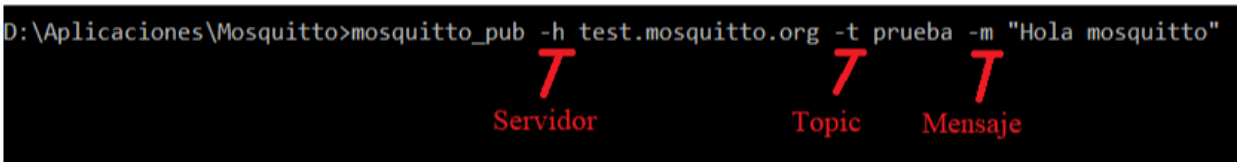


Figura 24. Función mosquitto_sub.

2.5.1.2. MQTT.js

Para el desarrollo del backend y del frontend se utilizó MQTT.js, una versión de MQTT implementada en JavaScript.

Esta librería de acceso público en Github, una vez importada permite crear un cliente, conectarnos al servidor, suscribirnos, publicar y capturar los mensajes escuchados. Esta última función da la capacidad de guardar los mensajes en variables según su tópico o contenido, con ello las posibilidades a la hora de desarrollar código son infinitas.

```
const mqtt = require('mqtt')
const client  = mqtt.connect('mqtt://test.mosquitto.org')

client.on('connect', function () {
  client.subscribe('presence', function (err) {
    if (!err) {
      client.publish('presence', 'Hello mqtt')
    }
  })
})

client.on('message', function (topic, message) {
  // message is Buffer
  console.log(message.toString())
  client.end()
})
```

Figura 25. Ejemplo MQTT.js [20].

2.5.2. Clientes de depuración

Con el objetivo de asegurar el buen funcionamiento de las comunicaciones se utilizaron clientes de depuración. Esto se llevó a cabo usando la consola de comandos tras haber instalado el Eclipse Mosquitto en el ordenador y el MQTT Explorer.

El cliente creado en la consola ya se explicó con el uso de mosquitto_pub y mosquitto_sub.

Por otra parte, el MQTT explorer es un cliente que proporciona una visión general estructurada por temas y recopila la información recibida [21].

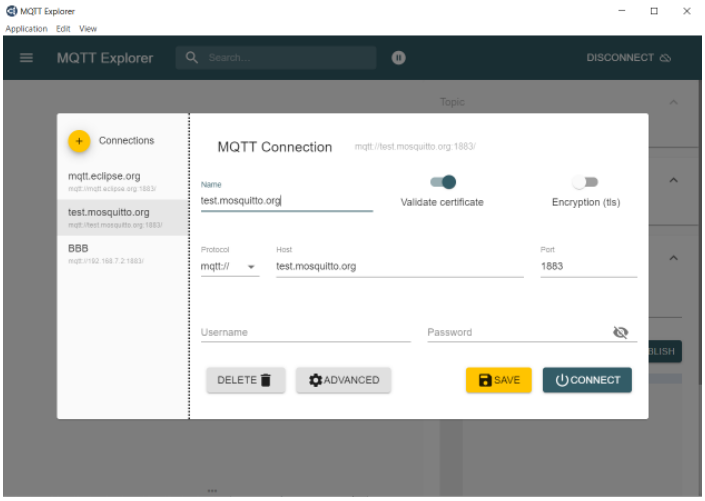


Figura 26. Inicio MQTT Explorer.

Una vez inicializado se puede decidir a que servidor conectarse y mediante que puerto o si necesita de usuario y contraseña.

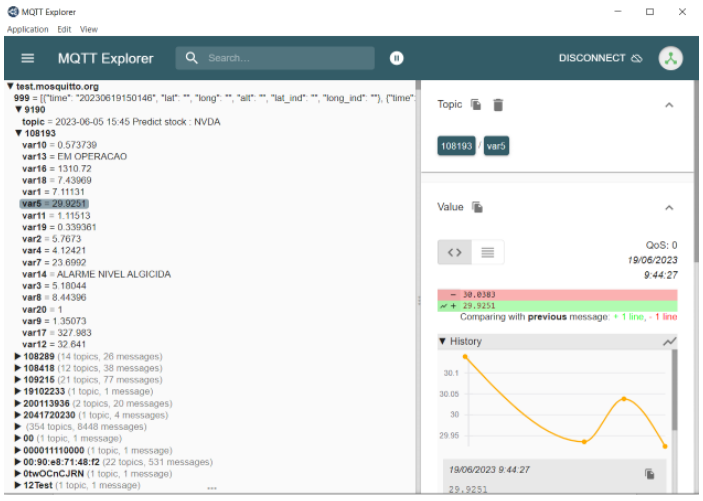


Figura 27. Zona de trabajo de MQTT Explorer.

Ya dentro, se puede seleccionar el tópico al que se desea suscribirse y visualizar la comparativa de un mensaje nuevo con el anterior o su evolución en el tiempo. También permite publicar mensajes.

2.6. Planta de tanques

La planta seleccionada para efectuar un control y monitorización aplicando la tecnología web, es “38-001 System Level & Flow control”, de la empresa Feedback. Es un panel de aprendizaje educativo de un sistema en lazo cerrado, cuyas variables de proceso vienen caracterizadas por el nivel del líquido y las tasas de flujo.

Este panel está compuesto por un tanque de proceso de doble compartimiento, conectado a un tanque sumidero mediante válvulas manuales y solenoides. Se emplea una bomba (situada en el tanque inferior) para hacer fluir el agua por el circuito, se hace pasar por un medidor de flujo de área variable y una válvula de control motorizada, el nivel del tanque se mide a través de un sensor de nivel con resistencia variable (potenciómetro) [21].

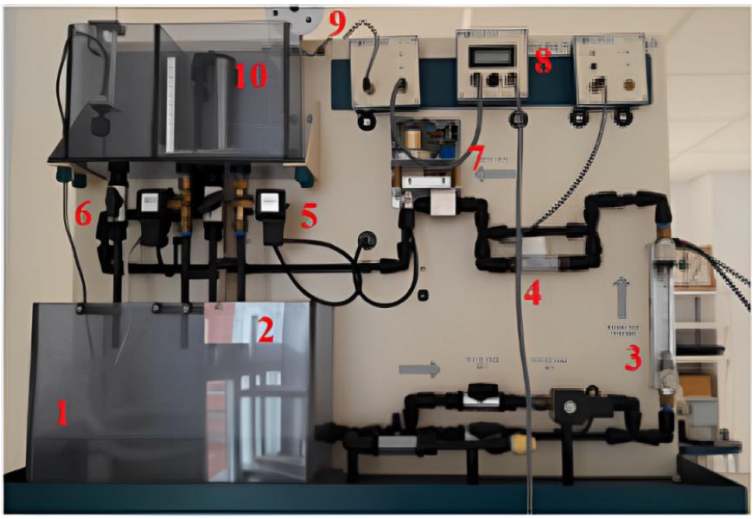


Figura 28. Elementos de la planta parte 1.

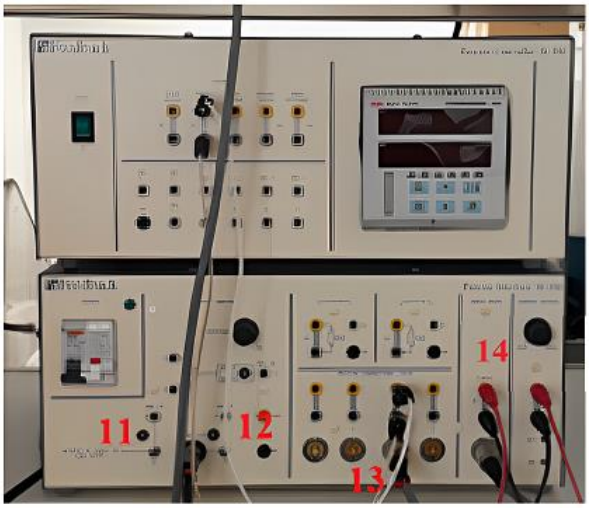


Figura 29. Elementos de la planta parte 2.

Tabla 2. Elementos del estanque.

Nº	Elementos
1	Bomba
2	Tanque sumidero
3	Caudalímetro
4	Sensor de flujo
5	Válvula solenoide
6	Válvula manual
7	Servo-Válvula
8	Visualizador de transmisión
9	Sensor de nivel (Flotador)
10	Tanque de proceso con doble compartimento
11	Interruptor bomba
12	Interruptor válvula solenoide
13	Salida de transmisores
14	Regulador de Servo-Válvula

Su funcionamiento consiste activar en una bomba (1) de agua que llena un estanque más rápido de lo que se vacía con el fin de subir el nivel de agua o apagar la bomba para que solo salga el agua. También se cuenta con la posibilidad de abrir una válvula de desagüe (6), para reducir el nivel, y una servo-válvula (7) para manipular el caudal de entrada (14).

Se emplea el estándar de 4 a 20 miliamperios tanto como para el control de elementos (14), como en el transductor de 4 a 20 miliamperios (13) y el visualizador de transmisión (8). Además todo el panel está protegido mediante un interruptor automático de corriente residual [22].

2.7. Controladores

Con el objetivo de interaccionar con la planta se han implementado una serie de controladores.

2.7.1. Control ON/OFF

El Control ON/OFF es un tipo de control que permite encender o apagar un dispositivo o sistema. Su funcionamiento es bastante simple: cuando está en posición "On" (encendido), permite el paso de corriente eléctrica o activa el funcionamiento del dispositivo/sistema; mientras que al colocarse en posición "OFF" (apagado), bloquea el paso de corriente o detiene el funcionamiento. Esto provoca un control muy impreciso ya que se va a los extremos provocando cambios muy rápidos en la salida del controlador [23].

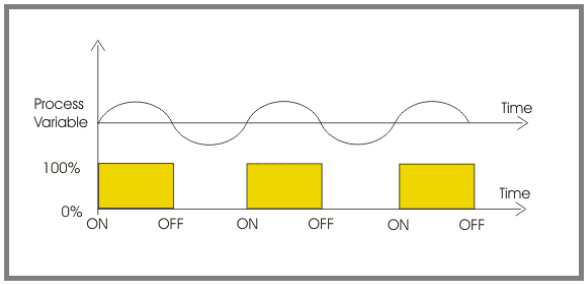


Figura 30. Control ON / OFF [23].

2.7.1.1. Control ON/OFF con histéresis

Para mejorar el controlador se le puede añadir una histéresis, es decir, un tiempo de espera antes de que se produzca el cambio entre los dos estados creando así una pequeña zona muerta antes de subir y otra antes de bajar.

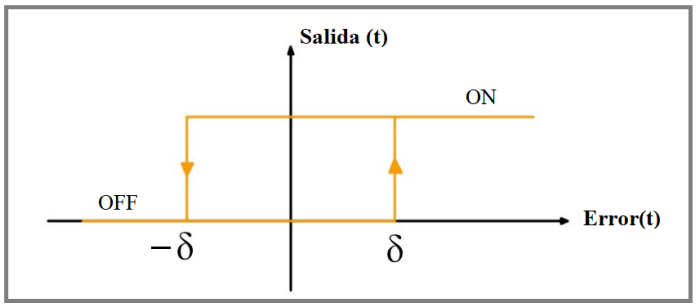


Figura 31. Control ON / OFF con histéresis.

Ahora la variable solo cambiará de estado cuando llegue al umbral delta o menos delta dando lugar a que no se produzcan cambios de forma tan instantánea. Pero se manifiesta una oscilación en la salida que aumenta directamente con un aumento en el valor de los límites y es inversamente proporcional a la tasa de cambio en la variable.

2.7.2. Control PID

El otro controlador a implementar será el PID, que a diferencia con el controlador ON / OFF es un control por realimentación continuo que usa todo el rango disponible de la variable de actuación y está compuesto por tres partes fundamentales [25].

- **Acción proporcional (P):** Generando una salida proporcional al error ayuda a alcanzar el valor deseado.

$$\boldsymbol{u(t) = K_p * e(t)}$$

- **Acción integral (I):** Corrige el error en el estacionario.

$$u(t) = K_i * \int_0^t e(\tau)\, d\tau$$

- **Acción derivativa (D):** De cierta manera vigila la velocidad de cambio del error, evitando cambios bruscos en la consigna.

$$u(t) = K_d * \frac{de(t)}{dt}$$

Combinando estas tres acciones se puede definir la salida del controlador de la siguiente forma.

$$u(t) = K_p * e(t) + K_i * \int_0^t e(\tau)\, d\tau + K_d * \frac{de(t)}{dt}$$

Capítulo 3: Desarrollo del proyecto

El desarrollo de este proyecto se presentará de forma cronológica con la finalidad de hacer partícipe al lector tanto de la adquisición de conocimientos como su implementación. Y mostrar así la evolución de este hasta su producto final.

3.1. Alcance del proyecto

En este apartado se detallan los procesos y decisiones tomadas durante el desarrollo del sistema SCADA utilizando tecnología web. Se presentarán diseños para sistemas más simples, así como para una maqueta de una planta industrial y se demostrará que su aplicación es fácil en cualquier tipo de sistema.

Aunque la planta ya cuenta con su propio sistema de control, se ha decidido implementar la utilización del sistema SCADA mediante la plataforma BeagleBone para llevar a cabo un control externo tanto ON / OFF como PID. A pesar de que el fabricante también provee un sistema de visualización del estado de la planta, se ha optado por desarrollar uno nuevo utilizando tecnología web para poder demostrar su amplio potencial.

3.2. Aprendizaje de HTML

El proyecto comenzó por aprender sobre el lenguaje HTML de cara a familiarizarse con los diseños web. Para conocer este lenguaje se recurrió a una serie de tutoriales de Youtube [26] y a páginas como W3Schools [27]. Siguiendo estas referencias se consiguió una estructura básica como se muestra en la Figura 31. También se diseñó un logo mediante el uso de FreeLodoDesign [28] y se descargó una imagen SVG de Pixabay [29], ambas páginas permitieron descargar los archivos de forma gratuita y libre de derechos de autor.

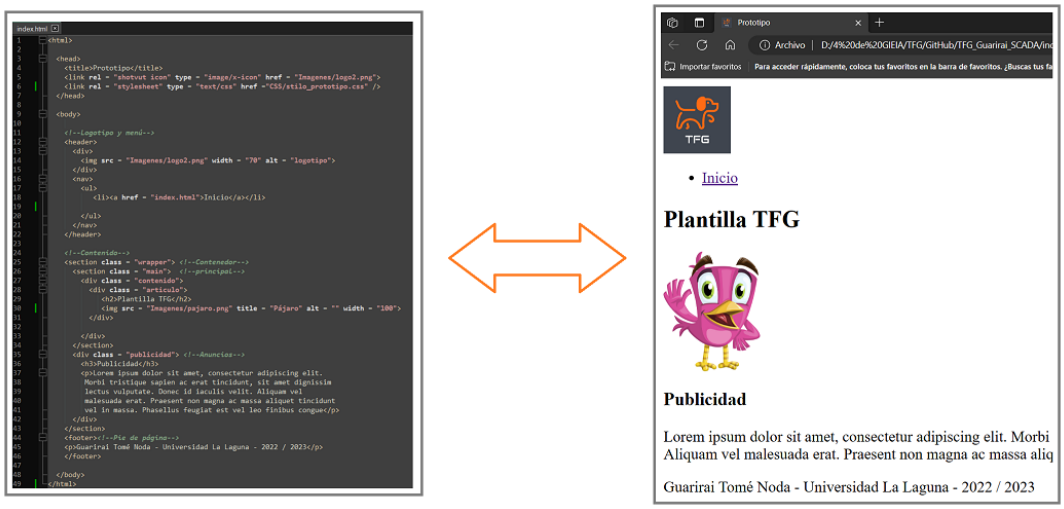


Figura 32. Primera estructura HTML.

El archivo HTML constaba de la siguiente distribución, aunque a primera vista no se detecte.

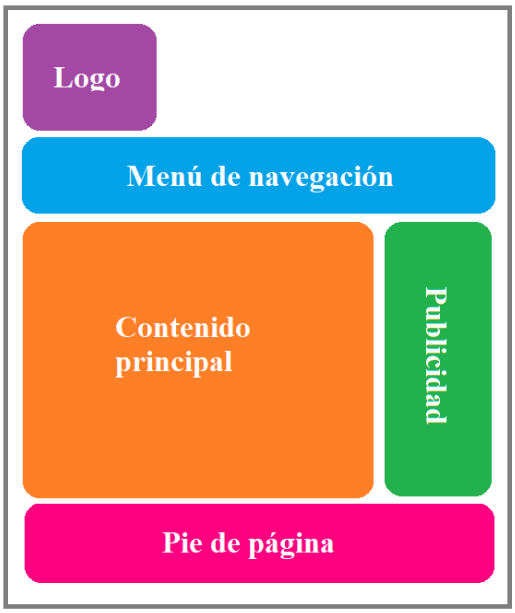


Figura 33. Distribución primera página.

3.3. Aprendizaje de CSS

El siguiente paso se centró en como hacer que los elementos de la página fueran agradables a la vista del usuario. Por ello se recurrió a video tutoriales [30], W3School [27] y páginas varias según las dudas que surgían o para conocer nuevas formas de distribuir los elementos.

Un atributo importante en el CSS son los colores, los cuales se pueden escribir de forma hexadecimal o en formato rgb. Habiendo tanta variedad se hace difícil saber cual es el mejor, pero se encontró una página que de forma didáctica facilitaba la elección de estos, HTMLlcolorcodes [31].

Aplicando los conocimientos adquiridos se consiguió trazar una plantilla donde trabajar los futuros diseños.

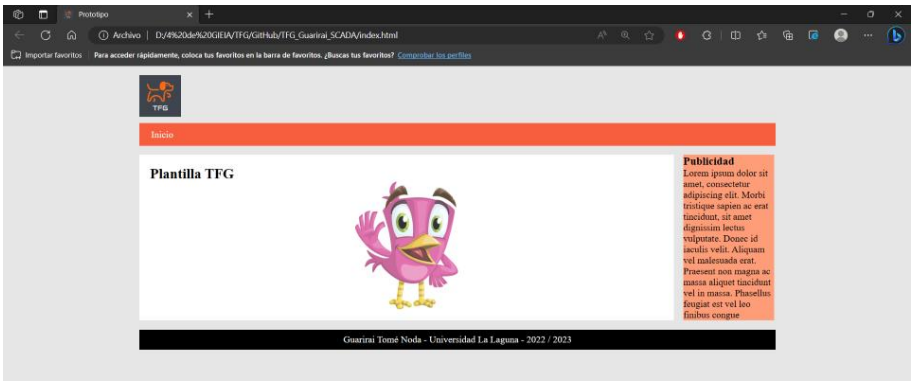


Figura 34. Diseño primera página web.

3.4. Aprendizaje de JavaScript

Para aprender la manipulación de objetos mediante el uso de JavaScript se propuso un pequeño ejercicio que consistía en crear una bombilla y hacer que se encendiera y apagara. Para dicho ejercicio se creó otra página web y se enlazó al menú de navegación.

La nueva página comparte exactamente todos los elementos excepto la imagen central que se cambió por la bombilla.

```
<!--Logotipo y menú-->
<header>
  <div>
    <img src = "Imagenes/logo2.png" width = "70" alt = "logotipo">
  </div>
  <nav>
    <ul>
      <li><a href = "index.html">Inicio</a></li>
      <li><a href = "HTML/Prtotipo_complementario.html">Bombilla</a></li>
    </ul>
  </nav>
</header>
```

Figura 35. Segunda página web.

La bombilla se descargó de Pixabay [29] y se editó con el fin de tener tres bombillas de distintos colores.

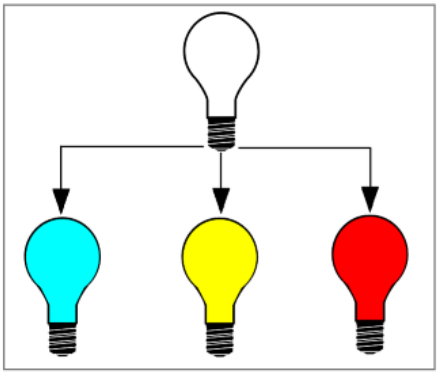


Figura 36. Bombilla y sus estados.

Una vez terminado el proceso de edición se comenzó con el de interactividad. Este punto se hizo mediante la integración de un nuevo elemento al menú de navegación, “Colores”. El cual consiste en un menú desplegable solo accesible desde del apartado “Bombilla”.

El menú desplegable se hizo mediante la inserción de una “div” que contuviera los distintos los distintos colores de la bombilla. Y se programó en CSS para que los elementos solo fueran visibles una vez el ratón estuviera encima del apartado “Colores”.

```html
<!--Logotipo y menú-->
<header>
  <div>
    <img src = "../Imagenes/logo2.png" width = "70" alt = "logotipo">
  </div>
  <nav>
    <ul>
      <li><a href = "../index.html">Inicio</a></li>
      <li><a href = "Prototipo_complementario.html">Bombilla</a></li>
      <li class = "desplegable">
      <a href = "#" class = "desp">Colores</a>
        <div class = "contenido_desplegable">
          <a id = "red" href = "#">Rojo</a>
          <a id = "blue" href = "#">Azul</a>
          <a id = "yellow" href = "#">Amarillo</a>
          <a id = "off" href = "#">Apagar</a>
        </div>
      </li>
```

+

```css
/* Menú desplegable*/
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden; /* Permite una nueva reorganización*/
}

li .desplegable {
  display: inline-block; /* Para que aparezcan en columna */
}
.contenido_desplegable {
    display: none; /* Mantiene el contenido oculto */
    position: absolute;
}
.contenido_desplegable a { /* Personalización */
  color: black;
  background:#f1f1f1;
  border: 1px solid black;
  padding: 1px 15px;
  text-decoration: none;
  display: block;
  text-align: left;
  cursor: pointer;
}

/* Cuando se hace un Hover se muestran*/
.contenido_desplegable a:hover {background-color: white;}

.desplegable:hover .contenido_desplegable {
  display: block;
}
```

Código 1. Menú desplegable.

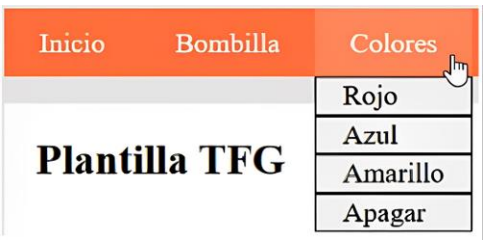


Figura 37. Menú desplegable.

Ya terminado la presentación solo resta añadir el script que cumplimentará la interactividad. El código es bastante sencillo, consta de guardar en variables las etiquetas de los colores y una vez se haga clic sobre alguna de ellas la imagen será sustituida por la del color correspondiente.

```
var redBton = document.getElementById("red");
var blueBton = document.getElementById("blue");
var yellowBton = document.getElementById("yellow");
var offBton = document.getElementById("off");
var color = document.getElementById("color");

redBton.onclick = function() {
    color.src = "../Imagenes/bombilla_vectorial_roja.svg";
}

blueBton.onclick = function() {
    color.src = "../Imagenes/bombilla_vectorial_azul.svg";
}

yellowBton.onclick = function() {
    color.src = "../Imagenes/bombilla_vectorial_amarilla.svg";
}

offBton.onclick = function() {
    color.src = "../Imagenes/bombilla_vectorial.svg";
}
```

Código 2. Cambio de color.

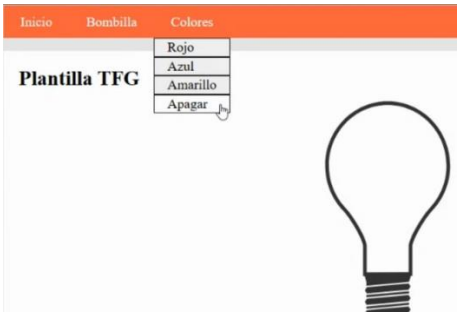


Figura 38. Bombilla apagada.

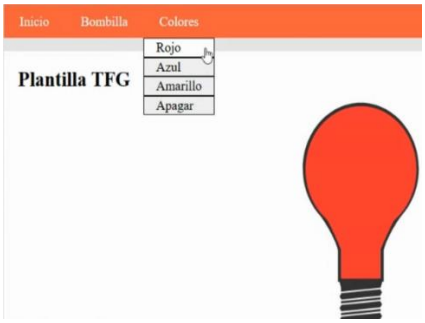


Figura 39. Bombilla roja.

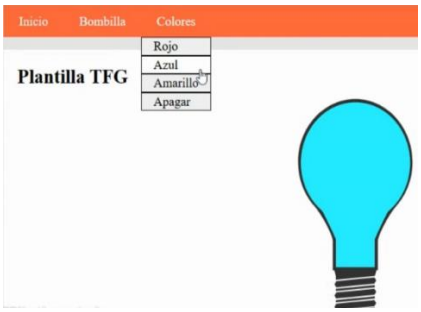


Figura 40. Bombilla azul.

Figura 41. Bombilla amarilla.

3.5. Primeros pasos con la BBB

A continuación, para familiarizarse con la librería BoneScript y el entorno Cloud9 se diseñaron e implementaron una serie de pequeños circuitos con las funciones básicas que más adelante cobrarán más importancia.

3.5.1. Circuito salida digital

En las entradas y salidas digitales hay que tomar ciertas precauciones y es que estos pines trabajan a 3.3 voltios cuando se configuran como entrada, si se utilizan con una tensión mayor se puede dañar la placa. Asimismo, cuando están configurados como salida, la máxima corriente que se debe solicitar es de 6 miliamperios [32].

El primer circuito consiste en utilizar la función `setInterval()` combinada con `digitalWrite()` para apagar y encender un led externo.

Las conexiones se hicieron de la siguiente forma. Se seleccionó el pin de propósito general 47, correspondiente a la columna P8 posición 15, para apagar o encender el led y se cerró el circuito usando un pin de tierra.

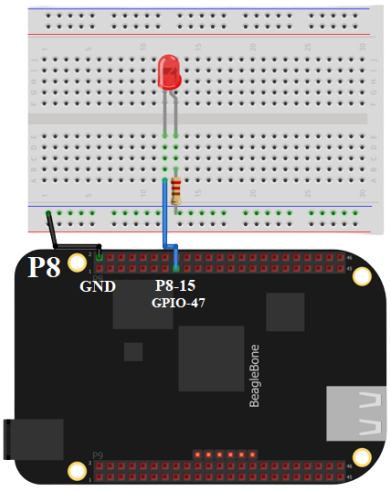


Figura 42. Circuito 1 - Parpadeo de Led.

El código correspondiente es muestra en Código 3.

```
// JavaScript File
var b = require('bonescript'); // LLamar a la librería
var LED = "P8_15"; // Elegir PIN
var state = 0; // Inicializar estado del Led
b.pinMode(LED, 'out'); // Declarar como salidad el pin

setInterval(blink, 1000); // Cada segundo el setInterval()
                          // activa la función blink
function blink() {
  state = !state; // Cambia el estado
  b.digitalWrite(LED, state); // Escribe el nuevo estado en el led
}

```

Código 3. Parpadeo led.

3.5.2. Circuito lectura Analógica

El segundo circuito consiste en leer el voltaje en una de las entradas analógicas mediante el uso de un potenciómetro. De este modo se comprueba que la entrada analógica trabaja en el rango de 0 a 1.8 voltios [32].

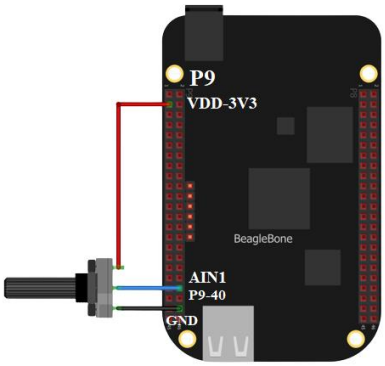


Figura 43. Circuito 2 - Potenciómetro.

El programa encargado de leer la entrada es el siguiente.

```
// JavaScript File
var b = require('bonescript'); // Llamda de librería

function Potenciometro() { // Función encargada de leer el pin
b.analogRead('P9_40', ValorPot);
}

function ValorPot(lectura) { // Mostar por consolo los valores leídos
    console.log('Valor medido = ' + lectura.value.toFixed(2) + "\n");
    var volts = lectura.value * 1.8;
    console.log("V = " + volts.toFixed(2) + " volts" + "\n");
}

setInterval(Potenciometro, 2000); // Activar función
```

Código 4 Lectura potenciómetro.

El valor de lectura va de 0 a 1, por lo que se multiplica por 1.8 para tener el valor real.

3.5.3. Circuito Entrada y Salida digital

El tercer circuito se destina a completar la interacción de entradas y salidas de la BeagleBone. De este modo se diseña se utiliza un pulsador para detectar cambios en una entrada digital y en función de su estado este se verá reflejado en un led como salida digital.

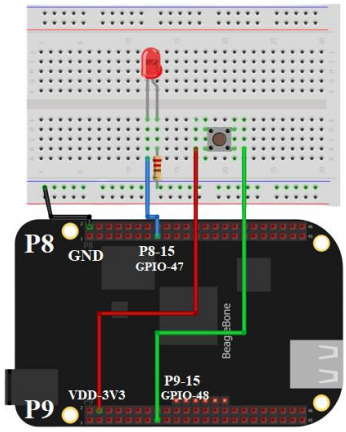


Figura 44. Circuito 3 - Entrada analógica y Salida digital

El programa ejecutado será el siguiente.

```
// JavaScript File

// Setup
var b = require('bonescript'); //Read library
var LED = "P8_11"; //Pin where LED is connected
var BUTTON = "P9_15";

b.pinMode(LED, b.OUTPUT); // declara el pin del led como salida
b.pinMode(BUTTON, b.INPUT); // declara el pin del botón como entrada

setInterval(readBUTTON, 1000); // Cada segundo lee el pin del botón

function readBUTTON() {  // Leed el pin y activa la función writeLed
    b.digitalRead(BUTTON, writeLED);
}

function writeLED(x) { // En función del valor leido apaga o enciende el led
    b.digitalWrite(LED, !x.value);
}
```

Código 5. Encender led.

3.6. Comunicaciones

Las comunicaciones son el engranaje central de este proyecto, sin ellas el sistema SCADA no podría llevarse a cabo. En los siguientes apartados se describirán los métodos utilizados para su desarrollo.

3.6.1. Función setTargetAddress()

La primera idea para implementar las comunicaciones fue estudiar el uso de la función `setTargetAddress()` proporcionada por la propia librería Bonescript. Al incorporar la mencionada línea de código en los archivos de JavaScript del frontend, esta buscará el archivo deseado en el backend y lo ejecutará.

Desgraciadamente no se pudo aplicar. Esto debió a una actualización de la BeagleBone Black donde se dejó de hacer distinción entre el servidor de la placa, es decir, la IP “192.168.7.2” que aloja las instrucciones para iniciar la BeagleBone y su puerto “192.167.7.2:3000”, destinado al entorno Cloud9. Por consecuencia, se impedía la conexión de la placa con su página interna.

3.6.2. Aprendizaje de MQTT

La segunda opción fue el uso del MQTT por su fácil implementación, su poco peso y la gran cantidad de posibilidades que ofrece.

Tras su instalación de forma global en la BBB se procedió a hacer un código de prueba.

```
// JavaScript File

const mqtt = require('mqtt'); // LLamada a la librería

const host = 'localhost';    // 192.168.7.2 por USB
const port = '1883';         // puerto de actuación
const clientId = `mqtt_BBB`; // nombre del cliente que creamos

console.log(clientId);

const connectUrl = `mqtt://${host}:${port}`; // Dirección a la que queremos conectar
const client = mqtt.connect(connectUrl, {    // Crear al cliente
  clientId,                                  // Especificamos la URL, el nombre,
  clean: true                                // y decimos que no queremos mensajes offline
});

const topic = 'test'; // Creamos un tópico
client.on('connect', () => { // Utilizamos la función conectar
  console.log('Connected');

  client.subscribe([topic],() => {  // Una vez conectados nos suscribimos
    console.log(`Subscribe to topic '${topic}'`);
  });

  client.publish(topic, 'nodejs mqtt test', { qos: 0, retain: false }, (error) => {
    if (error) {
      console.error(error);  // Después mandamos un mensaje de prueba
    }
  client.on('message', (topic, payload) => { // Recibimos el mensaje y imprimimos
  console.log('Received Message:', topic, payload.toString());
    });

  });
});
```

Código 6. Prueba comunicaciones.

En las líneas mostradas se observa como se invoca la librería, se crea un cliente, se forma la IP de este, se establece un tópico y mientras se está conectado al servidor nos suscribimos y se lanza un mensaje de prueba para verificar que se recibe bien.

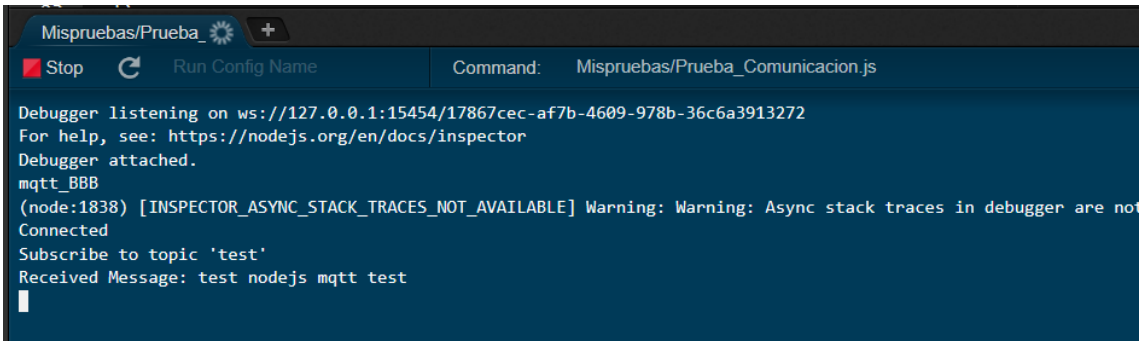


Figura 45. Resultado de la prueba de comunicación.

Una vez comprobado que el programa y las funciones operan de forma correcta vemos que se queda a la espera de nuevos mensajes. Por lo que se aprovecha para usar los clientes de depuración y comprobar su alcance para simular el frontend.

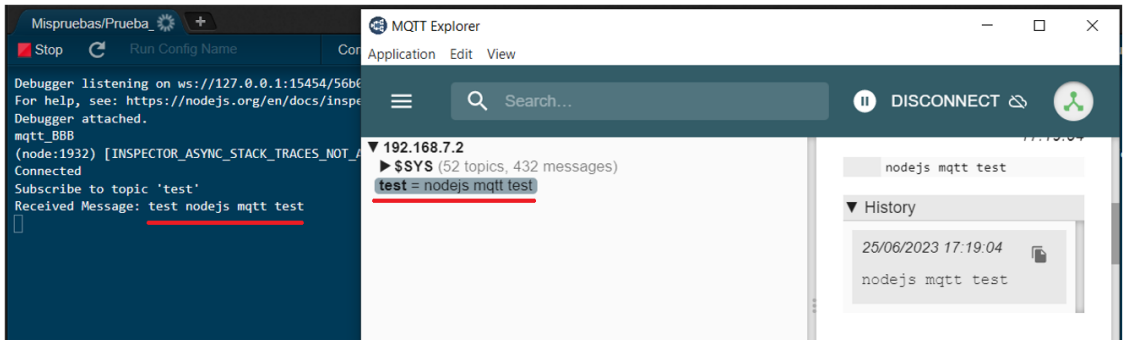


Figura 46. Recibiendo mensaje.

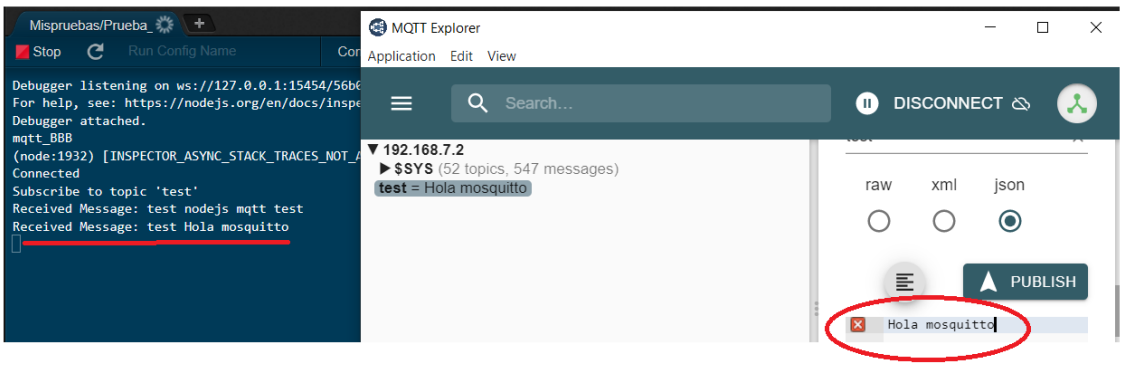


Figura 47. Enviando mensaje.

Mediante el Cliente de depuración MQTT Explorer se comprueba que el envío y recepción de mensajes se hace correctamente.

3.7. Primer Backend

Una vez controladas las comunicaciones se decidió aplicarlas para prototipar el backend. Por ende, se diseñaron tres códigos que interactúen con la BeagleBone mediante estos mensajes.

Primero se reutilizó el código que encendía y apagaba un led de forma intermitente y se le añadieron las comunicaciones de tal forma que cuando cambia de estado este envía y recibe un mensaje dando a conocer su situación.

```
// JavaScript File

// Estructura de control
var b = require('bonescript'); // Llammada de libreria bonescript
var LED = "P8_15"; // Selección de Pin
var state = 0; // Inicializacion del estado del led
b.pinMode(LED, 'out'); // Declaración del pin como salida

// Estructura de comunicación
const mqtt = require('mqtt'); // llamada librería comuicaciones

const host = 'localhost';
const port = '1883';
const clientId = `ParpadeoLed`;

console.log(clientId);

const connectUrl = `mqtt://${host}:${port}`;
const client = mqtt.connect(connectUrl, {  // Creamos el cliente
  clientId,
  clean: true,
});

// Funcionamineto
const topic = 'ParpadeoLed';
client.on('connect', () => { // Nos conectamos
  console.log('Connected');
  client.subscribe([topic], () => {  // Nos suscrbibimos
    console.log(`Subscribe to topic '${topic}'`);
  });

  setInterval(blink,2000); // Cada 2 segundos activamos la función blink

  function blink() {
    state = !state;
    b.digitalWrite(LED, state);
    console.log('Nuevo estado = ' + state);

    client.publish(topic, state.toString(), { qos: 0, retain: false }, (error) => {
      if (error) {
        console.error(error);
      } else {
        console.log('Mensaje enviado a', topic, state.toString())}
    });
  }

  client.on('message', (topic, payload) => {
  console.log('Mensaje recibido:', topic, payload.toString());
    });
});
```

Código 7. parpadeo led con comunicaciones.

En el siguiente enlace se muestra un vídeo del funcionamiento:

https://drive.google.com/file/d/1BQohMjsL_cgYelUiK9ZSPicsFkoB741N/view?usp=sharing

Luego, se realizó el mismo procedimiento para el código del potenciómetro. Dicho código sigue la misma estructura que el anterior, pero cambiando la parte de la publicación de mensajes.

```
function Potenciometro() {
  b.analogRead('P9_40', ValorPot);

  client.publish(topic, volts.toFixed(4).toString(), { qos: 0, retain: false }, (error) => {
  if (error) {
    console.error(error);
  } else {
    console.log('Mensaje enviado a', topic, volts.toFixed(2).toString())}
  });

}

function ValorPot(lectura) { // Convertimos el valor medido a voltaje
  console.log('Valor medido = ' + lectura.value + "\n");
  volts = lectura.value * 1.8;
  console.log("V = " + volts.toFixed(4) + " volts" + "\n");
}
```

Código 8. Potenciómetro con comunicaciones.

En el siguiente enlace se muestra el funcionamiento:

https://drive.google.com/file/d/1jxp8FX9XXRzrbVx64Es0y-lsV8edpP9s/view?usp=sharing

Por último, se diseñó un script que permitiera manipular un led en función de los mensajes recibidos. De este modo se le puede ordenar al led encenderse con "ON", apagarse con "OFF" y parpadear con "Parpadear".

```
// JavaScript File

// Estructura de control
var b = require('bonescript'); // Llamada de librería bonescript
var LED = "P8_15"; // Selecionar pin
var  state = 0; // Inicialización del led
b.pinMode(LED, 'out'); // Pin como salida

// Estructura de comunicación
const mqtt = require('mqtt'); // Llamada de librería de comunicaciones

const host = 'localhost';
const port = '1883';
const clientId = `ManipularLed`;

console.log(clientId);

const connectUrl = `mqtt://${host}:${port}`;
const client = mqtt.connect(connectUrl, {  // Crear cliente
  clientId,
  clean: true,
});

// Funcionamiento
const topic = "Manipular_Led";
client.on("connect", () => { // Nos conectamos
    console.log("Conectado");
    client.subscribe([topic], () => { // Nos suscribimos
        console.log("Subscrito al tópico: " + topic);
    });

    var mensaje2 = 0; // Variable que permite activar o desactivar el
                      // setInterval() del parpadeo

    client.on('message', (topic, mensaje) => {
    console.log('Mensaje recibido:', topic, mensaje.toString());

     switch(mensaje.toString()) {  // Según el mensaje que entre hará una cosa u otra

        case "ON":
            state = 1;
            b.digitalWrite(LED, state);
            console.log("Led encendido");

            mensaje2 = "ON"; // marcamos la variable en encendido

            client.publish(topic, state.toString(), { qos: 0, retain: false }, (error) => {
              if (error) {
                 console.error(error);
              } else {
                 console.log('Mensaje enviado: ', topic, state)}
             });

        break;
```

Código 9. Manipular un led con mensajes, primera parte.

```
55
56          case "OFF":
57              state = 0;
58              b.digitalWrite(LED, state);
59              console.log("Led apagado");
60
61              mensaje2 = "OFF"; // marcamos la variable en eapagado
62
63              client.publish(topic, state.toString(), { qos: 0, retain: false }, (error) => {
64                if (error) {
65                   console.error(error);
66                } else {
67                   console.log('Mensaje enviado: ', topic, state)}
68               });
69
70          break;
71
72          case "Parpadear":
73              console.log("Party");
74
75                   mensaje2 = "Party";
76                   var interval = setInterval(blink, 1000);
77
78                   function blink() { // Si el mensaje es uno de los siguientes deja de parpadear
79                       if ((mensaje2 == "OFF") || (mensaje2 == "ON") ) {
80                           clearInterval(interval); // función que permite desactivar un setInterval
81
82                       } else { // En caso contrario parpadea
83                          state = !state;
84                          b.digitalWrite(LED, state);
85
86                          client.publish(topic, state.toString(), { qos: 0, retain: false }, (error) => {
87                          if (error) {
88                            console.error(error);
89                          } else {
90                            console.log('Mensaje enviado: ', topic, state) }
91                        });
92                       }
93                   }
94          break;
95      }
96     });
97 });
```

Código 10. Manipular un led con mensajes, segunda parte

En el siguiente enlace se muestra el funcionamiento:

https://drive.google.com/file/d/1mu3KuIeG9qA_d_QDoq_zdPC_xf32jank/view?usp=sharing

3.8. Primer Frontend

Con la finalidad de incorporar la página web en la BeagleBone y aprovechar su servidor, se configuró la placa habilitando el puerto “:8080” de manera que ejecute el archivo “index.html” y así poner a funcionar el sistema SCADA.

3.8.1. Diseño Web

Para mejorar la presentación se hicieron cambios basándose en el estilo de la plantilla añadiendo nuevos elementos y cambiando la distribución.

La sección de contenido principal se dejó igual, pero cambiando la imagen que mostraba y situándola ala derecha de la pantalla.

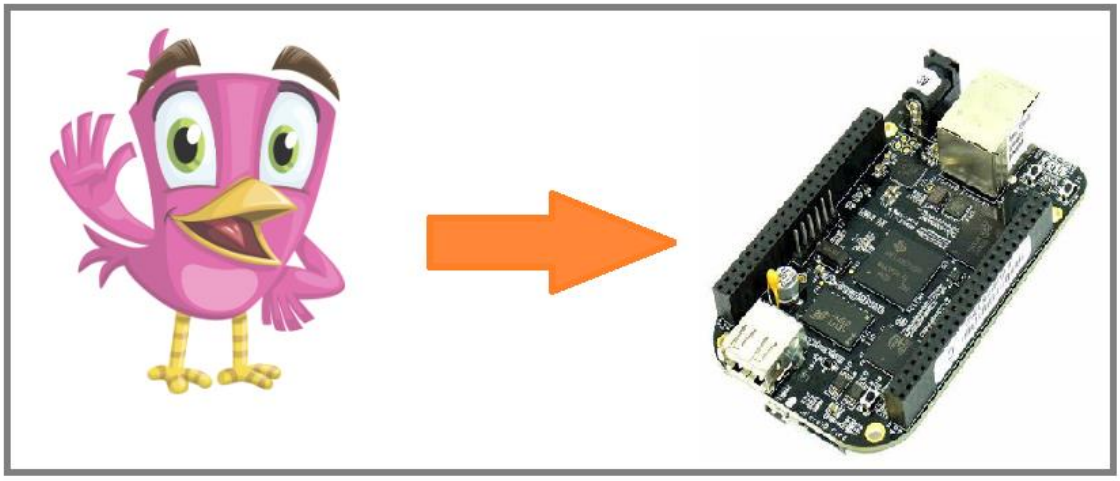


Figura 48. Cambio de imagen.

Por otra parte, el cambio más significativo se produjo en la zona destinada a "publicidad" dándole utilidad agregando los siguientes elementos.

```
<!--Coneccion-->
<section class = "wrapper">
  <div class = "conect">
    <section class="Cont">
    <h3><u>Conéctate a BeagleBone</u></h3>
  <br>
  <div id="status" class="status"><strong>Connection Status:</strong> <text id = "textoEstado"></text></div>
  <br>
  <form  method="get"> <!-- Formulario -->
        <label for="server">Server:</label> <!-- Etiqueta -->
        <text id = "host" class = "host">location.hostname</text> <!-- Texto que se modificará -->
        <br><br> <!-- Doble salto de línea -->
    <input id="btc" type="button" value="Conectar"><br><br> <!-- Botón -->

     <label for="Topic">Topic:</label>
        <select type="text" id="topic" name="topic"> <!-- Selector -->
         <option>----</option> <!-- opciones del selector -->
    </select><br><br>

    <input id="bts" type="button" value="Suscribir"> <!-- Botón para suscribirse -->
    <input id="btds" type="button" value="Desuscribir"> <!-- Botón para desuscribirse -->
    <br><br>

     <label for="messege">Mensaje:</label>
     <select type="text" id="mensaje" name="mensaje"> <!-- Un segundo select -->
         <!-- Las opciones se añadirán mediante JavaScript -->
     </select><br><br>
        <input id="btm" type="button" value="Enviar">
      </form>

  <!--Mensajes por pantalla-->
  <h3>Mensajes de depuración:</h2>  <!-- Zona para mostrar mensajes -->
  <textarea id = "console output" cols="42" rows="13" wrap="hard" readonly="yes"></textarea>
  </div>
</section>
```

Código 11. Zona de interacción.

Una vez definido el orden de los nuevos elementos y darles formato en CSS se obtuvo el siguiente resultado.

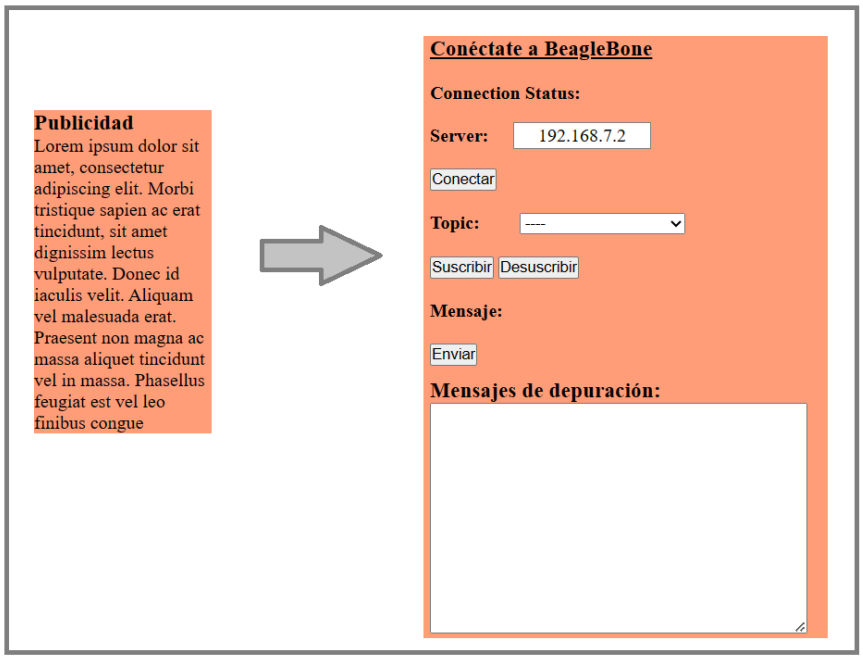


Figura 49. Panel de control.

De este modo se obtuvo un pequeño panel de control mediante selección capaz de mostrar un registro de las acciones tomadas, dejando el diseño de la página como se muestra a continuación.

Figura 50. Prototipo TFG.

3.8.2. Comunicación con mosquitto websocket

Ya concluido el diseñó se procedió a asegurar que la página web pudiera comunicarse con la BeagleBone.

Inicialmente se introdujo la estructura actual en la placa para aprovechar el acceso a la librería Node,js instalada globalmente. Pero no resultó como se esperaba, como ya se explicó, los navegadores no tienen soporte MQTT por lo que se necesitaba añadírselo.

Para ello se probaron varias librerías encargadas de esto como.

- CDN [33].
- Paho [34].
- Reboltz [35].

Las dos primeras fueron descartadas debido a que no terminaban de descargar completamente el contenido y no estaban bien estructuradas. En cambio, la segunda cumplía todos los requisitos pero no lograba conectarse al servidor.

Tras una investigación se descubrió la necesidad de incorporar el protocolo Websocket, una extensión de MQTT para páginas web. La librería selecciona ya contaba con esa variante por lo que solo se requiere de un pequeño cambio en la URL del cliente. Basta con sustituir “mqtt” por “ws”.

Figura 51. cambio para usar Webseocket.

Un problema añadido fue que el servidor de la placa no tiene habilitado la extensión websocket, por lo que se le añadió un archivo para permitirlo y otro para que no se perdiera la capacidad de usar el mosquitto de forma normal. Dejando así el puerto predeterminada “:1883” para el mosquitto normal y el puerto “:9001” para el uso del mosquitto websocket.

Ya terminadas todas las configuraciones y unas pequeñas pruebas se verificó el correcto funcionamiento de las comunicaciones, el frontend y el backend ya se envían y reciben información una de la otra.

3.8.3. Interacción de la página web

La interacción de la página se centra en el panel de control situado a la izquierda de la página, en ella podemos encontrar los siguientes elementos funcionales.

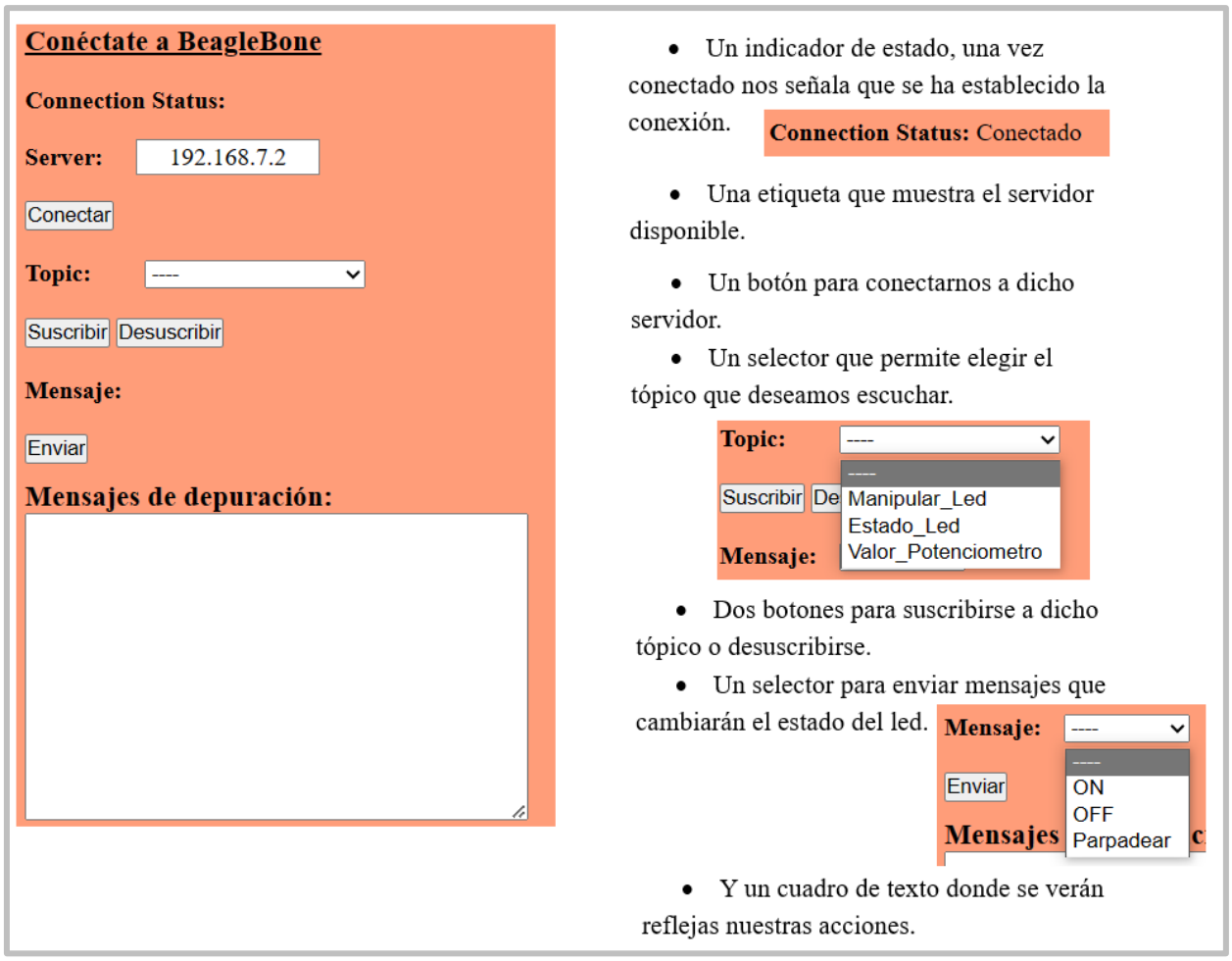


Figura 52. Elementos interactivos.

La ya mencionada interacción sigue la lógica. Primero se necesita del arranque mediante el botón "Conectar" para establecer la conexión con el servidor y habilitar el resto de elementos. Después se selecciona el tópico, solo se puede suscribir a una a la vez, por lo que si se desea cambiar se tendrá que desuscribir antes de seleccionar el nuevo. Dependiendo del tópico escogido ocurrirán los siguientes eventos.

- Tanto para "Estado_Led" como para "Manipular_Led" la imagen será cambiada por una bombilla que representa el estado físico del led, si está encendida aparece una bombilla roja y si está apagada una bombilla en blanco. La diferencia está en que para el tema "Manipular_Led" se vuelve visible un segundo selector que permite enviar mensajes para controlar el led.

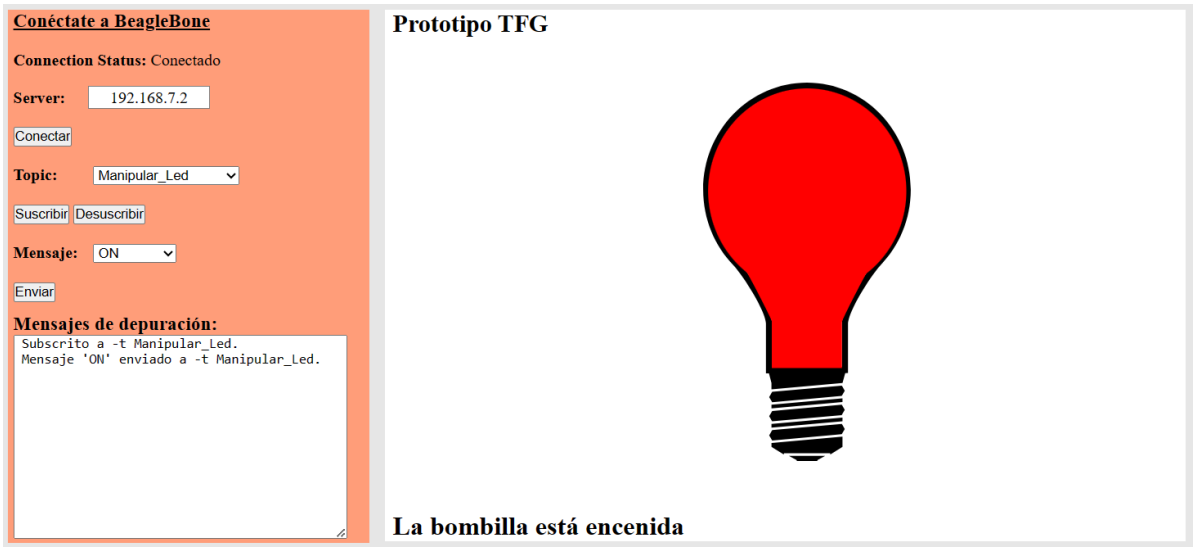


Figura 53. Ejemplo 1 de funcionamiento.

- Para el tópico "Valor_Potenciómetro" la imagen será sustituida por un potenciómetro y en la parte baja se muestra el voltaje que se lee del pin.

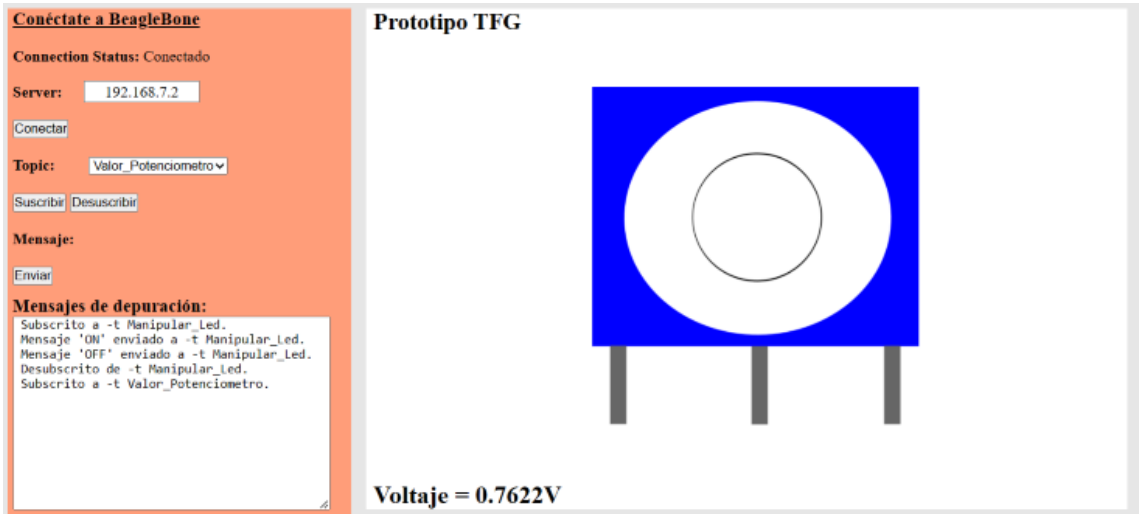


Figura 54. Ejemplo 2 del funcionamiento.

La interactividad de la página sigue la lógica de la figura 61.

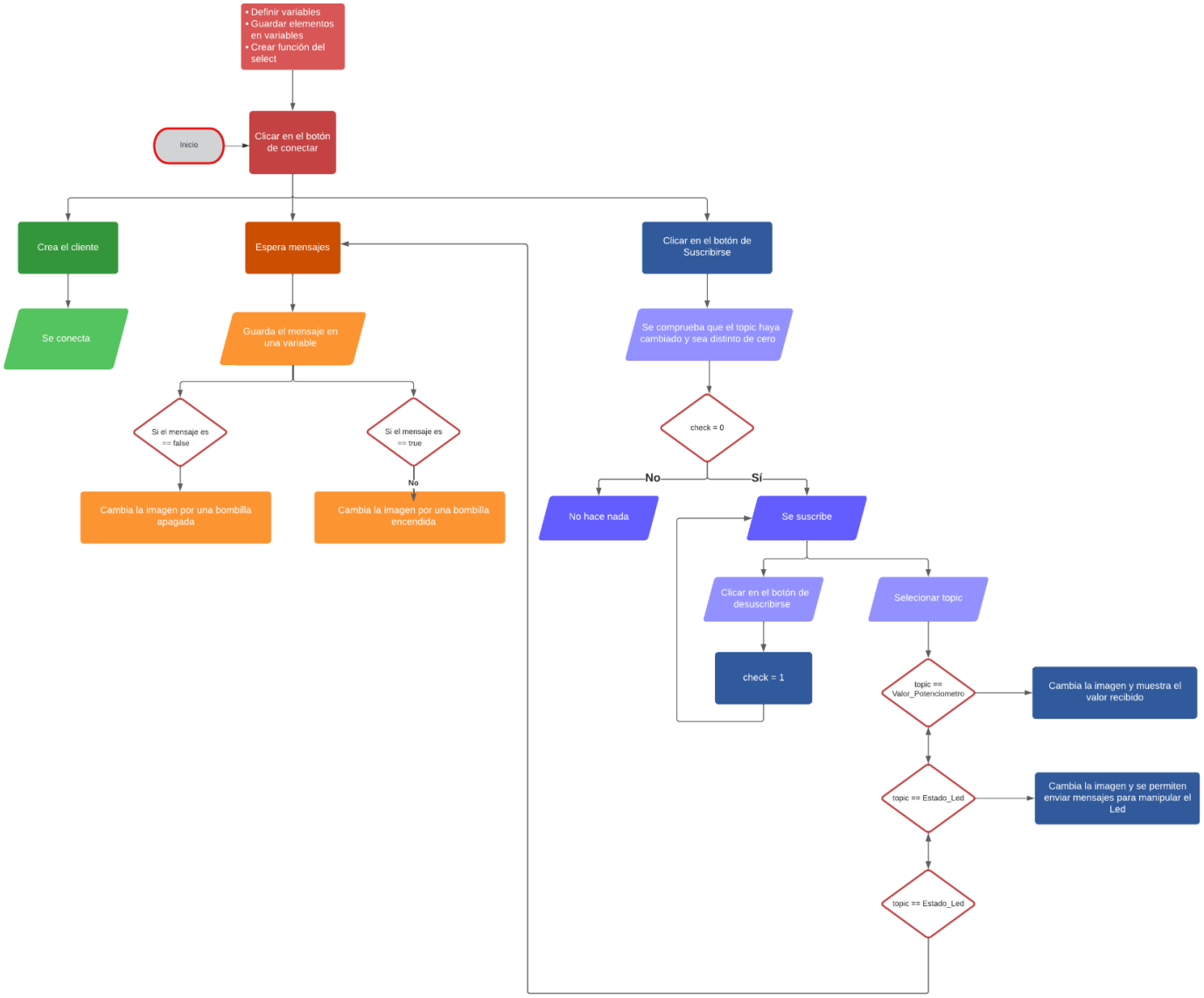


Figura 55. Diagrama de flujo.

En el siguiente enlace se muestra el funcionamiento de la página web:

https://drive.google.com/file/d/1NdxMSNScz6w9aM8y8l_dSx_YLzv5W1YN/view?usp=sharing

3.9. SCADA Estanque

Una vez acabado el prototipo se decidió llevar el proyecto un paso más allá con la propuesta de diseñar e implementar un sistema SCADA para la planta tipo estanque que se mencionó en el apartado 2.7.

3.9.1. Frontend

Se comenzó por el frontend con la idea de hacer unos ajustes al prototipo y añadir un gráfico, un esquema del estanque y una botonera para simular el control manual de esta.

3.9.1.1. Elementos SVG

Los elementos SVG que se diseñaron no se pueden abrir con los navegadores, puesto que estos por seguridad los bloquean. Por lo que esta parte se trabajó dentro del servidor de la BBB creando una página donde solo se mostrarán estos archivos.

Recordemos que los diseños se hacen en el entorno Inkscape el cual nos permite enfocar los archivos SVG a animaciones y dentro de un dibujo se puede diferenciar entre sus componentes añadiendo identificadores.

- Esquema del estanque:
  Para representar de forma visual el nivel de agua en el estanque se hizo el siguiente dibujo.

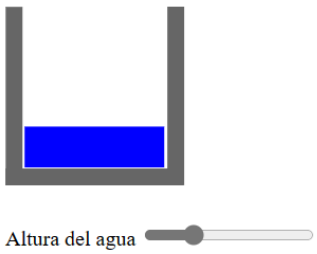


Figura 56. Diagrama estanque.

El esquema funciona de la siguiente forma. Mediante el input de tipo rango se determina el nivel deseado del líquido y a su vez se recoge dicho nuevo valor para escalar el elemento correspondiente al agua.

El código empleado es el siguiente.

```
document.addEventListener("DOMContentLoaded", () => {

//Estanque
const svg1 = document.getElementById("svgTanque"); // se accede al elemento
svg1.style.transform = "rotate(180deg)"; // se debe rotar porque los ejes de los
                                         // SVG están dispuestos de forma opuesta
console.log('svg:', svg1);

svg1.addEventListener('load', () => { // Se espera a que se carge el archivo
  console.log("Se cargó el SVG");
  console.log("Documento", svg1.contentDocument);

  // Del elemento se selecciona el agua
  const aguaEle = svg1.contentDocument.getElementById("agua");

  // Y también el atributo correspondiente a la altura
  const aguaMaxHeight = svg1.getAttribute('height');
  console.log("La altura de elemento agua es: ", aguaMaxHeight);

  // Se accede al input de tipo rango
  const eleAltoAgua = document.getElementById('altoAgua');

  fijaAltura(); // Se crea una función para manipular la altura del agua

  // cada vez que el input se modifique actuva la función
  eleAltoAgua.addEventListener('change',fijaAltura);

  function fijaAltura() {
    const valor = eleAltoAgua.value; // recoge el valor del input
    console.log("Valor es " + valor);
    // Lo escala y añade al agua
    aguaEle.setAttribute('height', valor * aguaMaxHeight / 100.0);
  }
});
});
```

Código 12. Altura del agua.

- Botón de encendido:
  Para activar o desactivar el sistema de forma remota se creó un botón de power, el cual al hacer clic encima de él, este simula su encendido o apagado cambiando un color oscuro por uno más intenso.

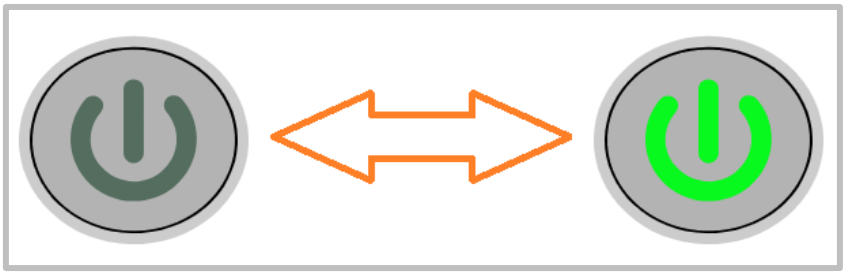


Figura 57. Botón de power encendido y apagado.

El código empleado es el siguiente.

```
// Power
const svg2 = document.getElementById("svgPower"); // se accdede al elemento
console.log('svg:', svg2);
svg2.addEventListener('load', () => { // se espera a que se carge
  console.log('Se cargó el SVG2');
  console.log("Documento", svg2.contentDocument);

  // Se accede al botón dentro del elemento
  const Power = svg2.contentDocument.getElementById("Boton");
  console.log("BotonPower es, ", Power);

  // Se accede al icono dentro del elemento
  const Icono = svg2.contentDocument.getElementById("Icono_Encendido");
  console.log("Icon es ", Icono);

  // Se guarda el atributo correspondiente al color
  const Color_Power = svg2.getAttribute("fill");
  console.log("El atributo fill es: " + Color_Power);

  Power.addEventListener("click", Cambio_Color); // Cuando se clique en el
                                                 // botón se cambia el color
  function Cambio_Color() {
    if (Icono.style.fill == "rgb(5, 249, 27)") { // Si está encendido lo apaga
      Icono.style.fill = "#536c5d";

    } else { // Si está apagado lo enciende
      Icono.style.fill = "#05F91B";
    }
  }
});
```

Código 13. Encendido y apagado del power.

- Palanquita:

En la planta podemos encontrar dos interruptores tipo palanca que hacen de actuadores para encender y apagar la bomba o la válvula. El siguiente elemento se diseñó de forma lo más fiel posible a la realidad tanto para una mejor sensación de control al usuario.

Dicho componente rota sobre sí mismo para dar el efecto óptico de un movimiento vertical simulando la apertura o cierre del circuito.

El código empleado es el siguiente:

```
// Palanquita
const svg3 = document.getElementById("svgPalanquita"); // se accede al elemento
svg3.addEventListener("load", () => { // se espera a que carge
  console.log('Se cargó el SVG3');
  console.log("Documento", svg3.contentDocument);

  // Se accede al elemento palanca del svg
  const Palanquita = svg3.contentDocument.getElementById("Palanquita1");

  // Cuando se clica sobre él gira
  Palanquita.addEventListener("click", Girar);

  function Girar() {
    let rotada = 0;
    if (svg3.style.transform == "rotate(180deg)") {
      svg3.style.transform = "rotate(0deg)";
      rotada = 0;
      console.log(rotada);
    } else {
      svg3.style.transform = "rotate(180deg)";
      rotada = 1;
      console.log(rotada);
    }
  }
});
```

Código 14. Rotar palanca.

3.9.1.2. Gráfico

Otro elemento importante es el gráfico, el cual nos mostrará la evolución temporal del nivel del agua. Para crearlo y mostrarlo se ha recurrido a W3School ya que cuenta con un apartado dedicado a Chart.js [36].

Para utilizar el gráfico solo se necesita incorporar la librería que se muestra a continuación.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"></script>
```

Figura 58. Librería Chart.js.

Una vez cargada la librería añadimos la etiqueta *"<canvas>"*, de uso específico para gráficos.

```
<div>
  <canvas id="myChart" style="width:100%;max-width:300px"></canvas>
</div>
```

Figura 59. Etiqueta canvas.

Seguidamente se define el objeto con las características deseadas en un script aparte. Para este proyecto se ha escogido el siguiente diseño.

```
// Gráfico
window.onload = (event) => {
creaChart();
};
function creaChart() {
const xValues = []; // eje X
const yValues = []; // eje Y

// Definimos que el gráfico va a ser 2d
var grafico = document.getElementById("myChart").getContext("2d");

// creamos el gráfico
var lineChart = new Chart("myChart", {
  type: "line",
  data: {
    labels: xValues,
    datasets: [{
      fill: true,
      lineTension: 0,
      backgroundColor: "rgba(81,213,209)",
      borderColor: "rgba(13,126,246)",
      data: yValues
    }]
  },
  options: {
    legend: {display: false},
    scales: {
      xAxes: [{display : false}],
    }
  }
});

lineChart.update(); // Importante actualizar después de crear
```

Figura 60. Diseño del gráfico.

Y para que el gráfico se actualice se necesita usar el método ".update()", con ello se consigue que los cambios realizados se vean reflejados. En este caso cada vez que entra un dato nuevo se elimina el último, así se consigue un efecto de ventana corrediza.

El efecto comentado se diseña en el siguiente fragmento de código.

```
setInterval(AñadirPuntos, 1000); // cada segundo añade un punto
 var t = 0;
 function AñadirPuntos() {
  if (t < 10) { // primero nos aseguramos de que
  t = t + 1;    // en el gráfico aparecen 10 puntos

  xValues.push(t); // .push sirve para incorporar datos a un array
  console.log("valor de x: " + xValues);

  yValues.push(getRandomInt(10));
  console.log("valor de y: " + yValues);

  } else {  // una vez ya existan los 10 puntos se elimina el
    t = t + 1; // último y se añade uno nuevo para mantener el mismo número

    xValues.push(t);
    xValues.shift(); // elimina el primer elemento

    yValues.push(getRandomInt(10));
    yValues.shift();
  }

  // Finalizada la condiación actualizamos para ver los cambios
  lineChart.update();
 }
```

Código 15. Actualización del gráfico.

3.9.1.3. Diseño

Con la finalidad de adaptar el prototipo a la planta se planteó un diseño inicial donde aparecería el gráfico en la zona de control y en la sección de contenido se montaría una botonera junto al diagrama del estanque.

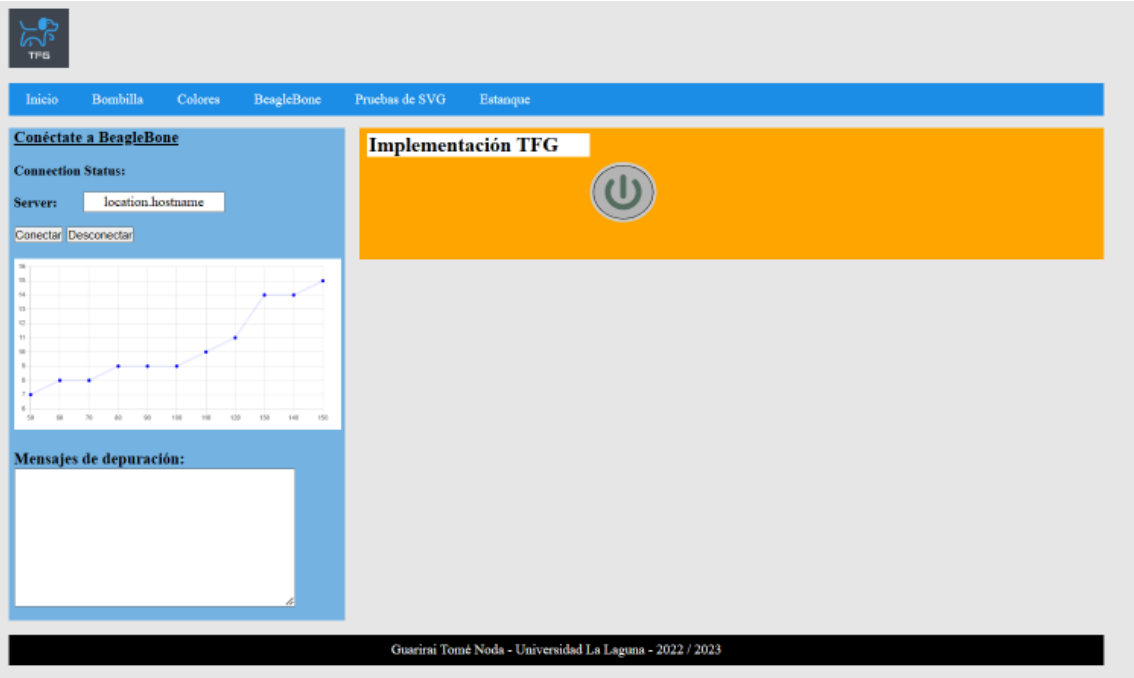


Figura 61. Primera página del estanque.

El diseño quedó a medias debido a que la sección del contenido no terminaba de agradar y no era muy intuitiva. Por lo que después de unos días y varios bocetos se llegó a la propuesta de la figura 61.

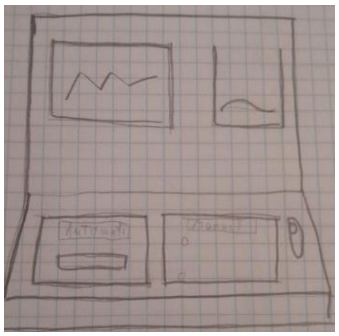


Figura 62. Boceto 1.

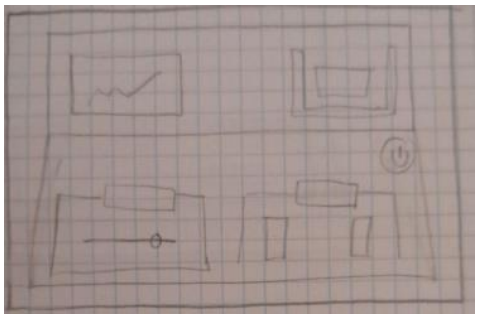


Figura 63. Boceto 2.

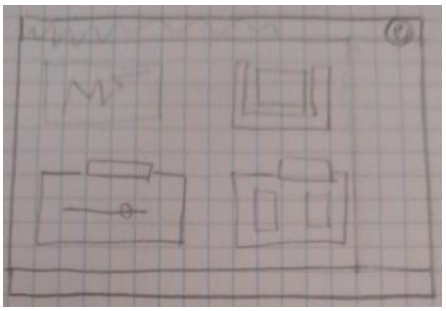


Figura 64. Boceto 3.

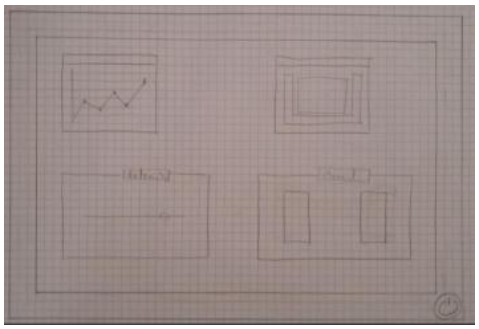


Figura 65. Boceto final.

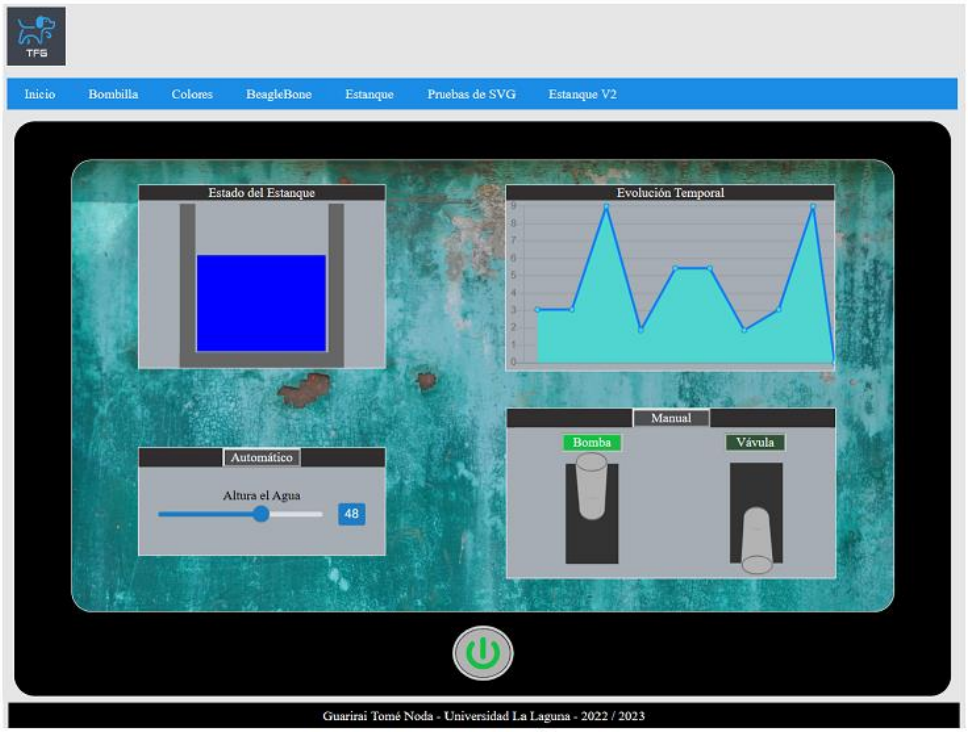


Figura 66. Nuevo diseño del SCADA.

El nuevo diseño inspirado en una Tablet presenta los elementos de una forma más agradable visualmente y de forma más intuitiva.

3.9.1.4. Interactividad

En esta página web la interactividad se produce de la siguiente forma.

- El botón de power muestra los contenidos del sistema SCADA mientras se conecta al servidor o los oculta mientras en una pantalla oscura mientras se desconecta.
- El esquema del estanque sube o baja según recibe los mensajes con la altura del nivel del agua en tiempo real.
- El gráfico muestra la evolución temporal de la cantidad de agua.
- El modo automático solo tiene implementado un controlador. Este de forma automática activa o desactiva los actuadores del estanque para llegar a la altura deseada enviando el valor seleccionado como mensaje al backend.
- El modo manual permite controlar a voluntad la bomba o la válvula según desee el usuario. La palanca activada se resalta con un indicador luminoso. Las acciones de subida o bajada de las palancas son transmitas por mensajes para que el backend las transmita a la panta de forma física,
- Encima de las ventanas de control automático o manual se encuentran dos botones, estos efectúan el cambio de un modo de control al otro mostrando los mandos del seleccionado y ocultando los del otro para evitar posibles errores.

Figura 67. Estado inicial de los modos de control.

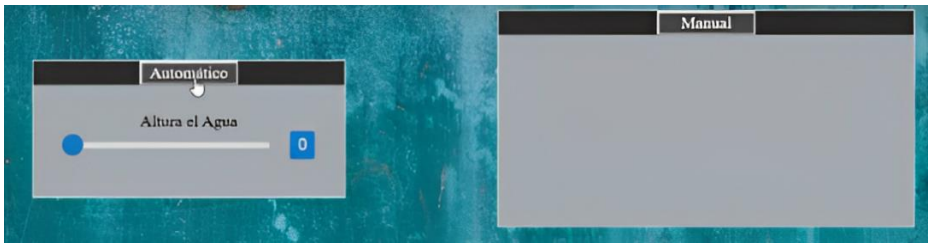


Figura 68. Selección de modo automático.

Figura 69. Selección de modo manual.

3.9.2. Conexiones entre la planta y la BeagleBone Black

Para conectar la BeagleBone a la planta de tanques se recurrió al siguiente modelo.

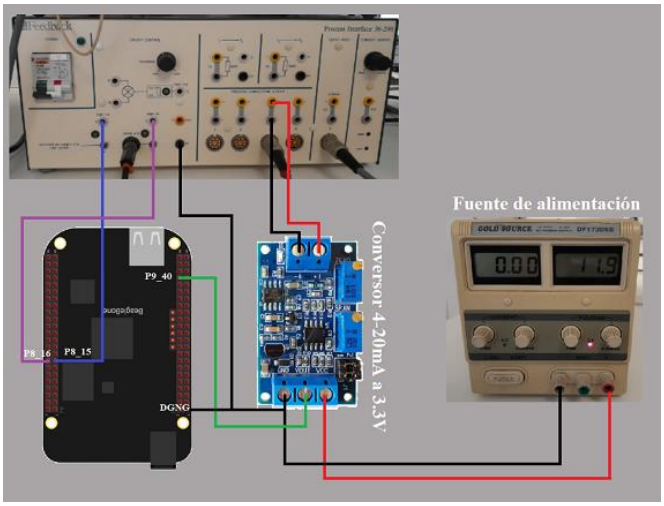


Figura 70. Conexión BBB con estanque.

Para poder manipular los interruptores de la bomba y la válvula se utilizaron los pines de propósito general P8_15 y P8_16 con el fin de enviar un cero o un uno a la entrada.

Se necesitaba leer la cantidad de corriente que sale de la salida de transmisiones, el cual está directamente relacionado con el nivel del agua en el estanque, partiendo de 0% representado con 4 mA hasta el 100% correspondiente a 20 mA.

Con esto surge un problema, no puede leer tanto amperaje. Por lo que se recurre a un conversor de amperios – voltios [Anexo I] para aprovechar la entrada modular de la placa y convertir el rango de amperios a 0 - 1.8V. Una vez calibrado y conectado se comprobó que el conversor satura a 1.8V a partir del 75% de la cantidad de agua en el estanque limitando el volumen de agua a controlar.

Otro punto importante es la puesta conjunta de las tierras para asegurar la misma referencia en todos los dispositivos implicados.

3.9.3. Comunicaciones Sistema SCADA

Las comunicaciones para esta versión se establecen de manera que cada parte implicada escucha y recibe información en un único sentido.

El frontend escucha a través del tópico “AutomaticoWeb” recibiendo asi los distintos niveles de agua que se van leyendo con la finalidad de mostrar en tiempo real lo que sucede. Por otra parte, envía los mensajes de las acciones realizadas en la pantalla al tópico “ControlBBB” para que la placa los gestione.

Mientras el backend funciona de forma inversa, se suscribe al tópico “ControlBBB” para gestionar las acciones realizadas por el usuario y envía los datos recogidos del estanque al tema “AutomaticoWeb” en tiempo real para conocer tanto la evolución temporal del nivel del agua como su posición actual.

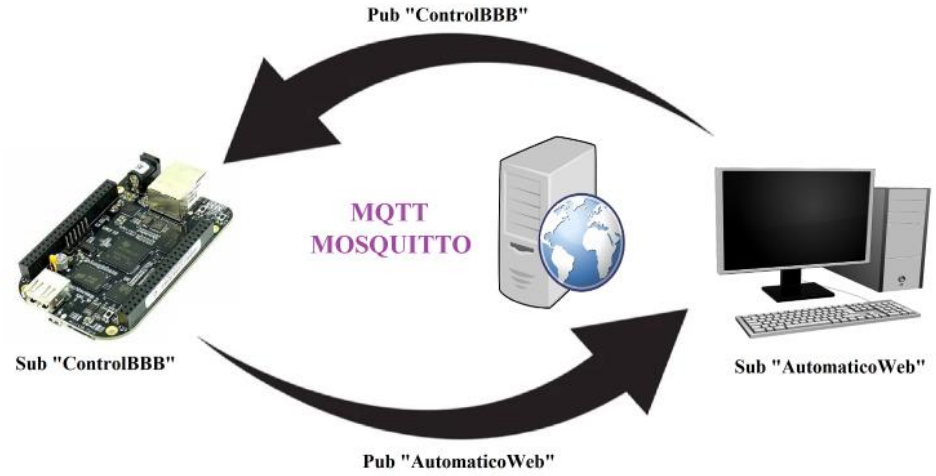


Figura 71. Esquema comunicaciones.

En la Tabla 3 se muestran las variables de la comunicación.

Tabla 3. Variables de la comunicación.

Tópicos	Descripción	Mensajes que recibe
ControlBBB	Destinado a recibir las acciones que toma el usuario o las modificaciones de las variables. Usado para representar la evolución temporal del estanque y el nivel de agua en tiempo real.	Automático: Para desactivar el modo manual, activar el modo automático y así comparar la consigna con la altura mediante un control On – OFF. Una vez activado este modo es capaz de recibir el valor del input como mensaje y trabajarlo como una variable.
		Manual: Para desactivar el modo automático y activar el modo manual, donde el usuario decide que actuador manipular. Una vez activado este modo distingue entre los siguientes mensajes. • B1 → Para encender la bomba. • B0 → Para apagar la bomba. • V1 → Para encender la válvula. • V0 → Para apagar la válvula.
AutomáticoWeb	Usado para representar la evolución temporal del estanque y el nivel de agua en tiempo real.	Mediante la lectura del voltaje de la planta la BBB transforma esa señal en altura y la envía a la página web.

3.9.4. Backend

El backend de la versión actual del sistema SCADA funciona según la siguiente lógica mostrada en la Figura 70.

Primero se inicia el backend cargando las librerías y conectándose al servidor. Una vez establecida la conexión efectúa en paralelo tres acciones.

La primera es suscribirse al tópico “ControlBBB” para poder recibir los mensajes enviados por el usuario.

La segunda consiste en leer el voltaje del conversor Am – V conectado a la planta para transformar los valores en altura mediante una regla de tres y enviarlos al frontend.

Por último, se activa el modo manual o automático en función del mensaje que se recibe desactivando el otro. Si se activa el modo manual este pasa a recibir los valores del input como consigna del sistema y compara con la altura actual usando un control On-Off. Por otro lado, si el usuario elije el modo manual la consigna pasa a valer cero y el llenado o vaciado del estanque pasa ser manipulado directamente mediante el encendido o apagado de la bomba o la válvula.

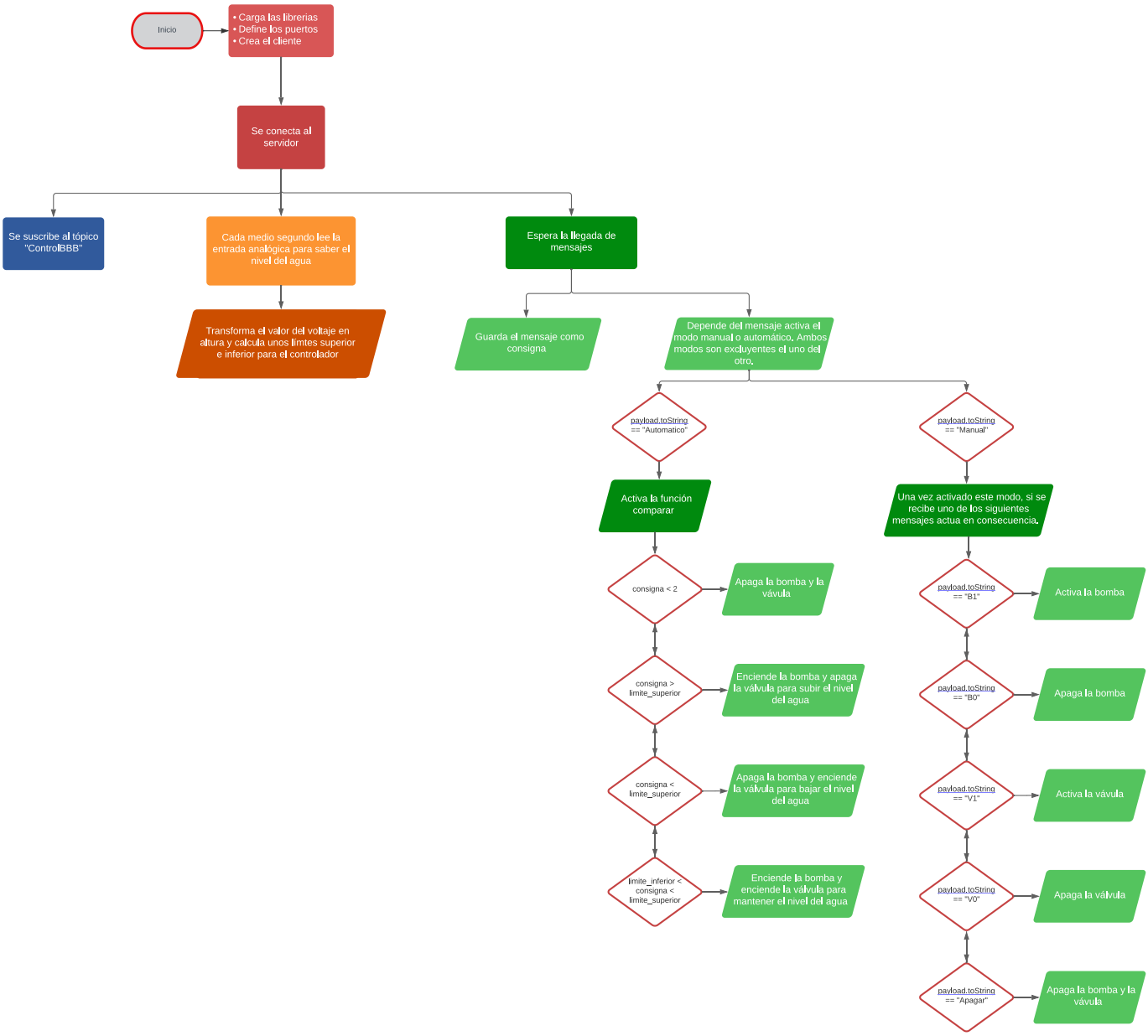


Figura 72. Diagrama de flujo del estanque.

Las partes más importantes de este código se muestran en Código 16.

```
setInterval(Estanque, 500); // Cada medio segundo lee el pin de entrada analógico

function Estanque() {
  b.analogRead('P9_40', ValorEst); // lee el pin y activa la función ValorEst

  // Publica un mensaje con la altura
  client.publish("AutomaticoWeb", altura.toFixed(4).toString(), { qos: 0, retain: false }, (error) => {
    if (error) {
      console.error(error);
    } else {
      console.log('Mensaje enviado a', topic, altura + "\n");
    }
  });
}

// Inicializamos unos limites
var limite_inferior = 0;
var limite_superior = 0;

function ValorEst(lectura) {
  console.log("\n" + 'Valor medido = ' + lectura.value.toFixed(4) + "\n");
  volts = lectura.value * 1.8; // convertimos la lectura en voltaje
  // y convertimos el voltaje en altura
  altura = (volts * 50)/ 1.1758;  // uso este valor para hacer la regla de 3 poque es lineal y el cáculo sale bastante bien

  // Con los límtes se genera un margen de actuación para el controlador
  limite_inferior = altura - 0.5;
  limite_superior = altura + 0.5;
  console.log("Limite inferior: " + limite_inferior);
  console.log("Limite superior: " + limite_superior);
  console.log("V = " + volts.toFixed(4) + " volts" + "\n");
  console.log("H = " + altura.toFixed(0) + " unidades" + "\n");
}
```

Código 16. Lectura y envio de altura más el cálculo de los límites.

```
client.on('message', (topic, payload) => {
  console.log('Mensaje recibido:', topic, payload.toString() + "\n");
  consigna = payload.toString();
  parseInt(consigna, 10); // convierte un string en entero con base decimal


  console.log("La altura es " + consigna + " unidades");

  // Para activar o desactivar el modo manual
  if (payload.toString() == "Automatico") {
    modo = payload.toString();
  } else if (payload.toString() == "Manual") {
    modo = payload.toString();
  }

    switch (modo) {
      case "Manual":
        mensaje2 = "Manual";
         b.digitalWrite(Bomba, 0); //Para evitar que siga funcionando en el cambio
         b.digitalWrite(Val_Esc, 0);
        if (payload.toString() == "B1") { // Enciende la  bomba
          b.digitalWrite(Bomba, 1);
        } else if (payload.toString() == "B0") { // Apaga la bomba
          b.digitalWrite(Bomba, 0);
        } else if (payload.toString() == "V1") { // Enciende la válvula
          b.digitalWrite(Val_Esc, 1);
        } else if (payload.toString() == "V0") { // Apaga la válvula
          b.digitalWrite(Val_Esc, 0);
        } else if (payload.toString() == "Apagar") { // Apaga todo
          b.digitalWrite(Bomba, 0);
          b.digitalWrite(Val_Esc, 0);
        }

      break;
```

Código 17. Guardar consigna y modo manual.

```
    case "Automatico":
      mensaje2 = "Automatico";
       b.digitalWrite(Bomba, 0); //Para evitar que siga funcionando en el cambio
       b.digitalWrite(Val_Esc, 0);

       // Cada medio segundo compara la altura con la consigna
       var interval = setInterval(Comparar, 500);
    break;
  }

function Comparar() {
 if (mensaje2 == "Manual") {
   clearInterval(interval); // Cuando el mensaje es "Manual" desactiva el intervalo
   b.digitalWrite(Bomba, 0); //Para evitar que siga funcionando en el cambio
   b.digitalWrite(Val_Esc, 0);
 } else {
   //console.log("estoy funcionando");
      if (consigna < 2) {
        b.digitalWrite(Val_Esc, 0);
        b.digitalWrite(Bomba, 0);
        console.log("apagando");

      } else if (consigna > limite_superior) {
          b.digitalWrite(Val_Esc, 0);
          b.digitalWrite(Bomba, 1);
          console.log("subiendo");

      } else if (consigna < limite_inferior) {
          b.digitalWrite(Val_Esc, 1);
          b.digitalWrite(Bomba, 0);
          console.log("bajando");

      } else if (limite_inferior < consigna < limite_superior) {
          b.digitalWrite(Val_Esc, 1);
          b.digitalWrite(Bomba, 1);
          console.log("me mantengo");
      }
 }
}
```

Código 18. Modo automático.

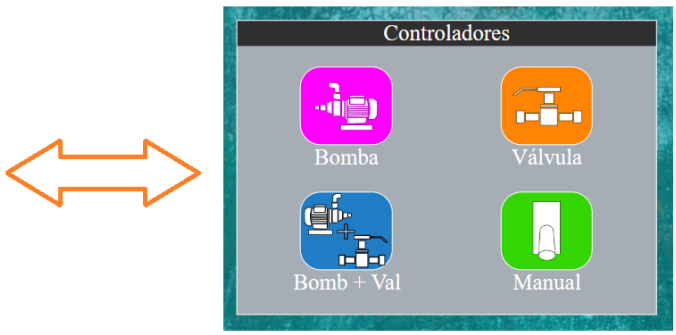
3.10. Actualización del Sistema SCADA

El proyecto ha alcanzado una etapa en la que es totalmente operativo, pero aún puede ser mucho mejor. Por ello se trabajó en una mejora significativa donde la presentación muestra más información al usuario, se incluyen varias formas de manejar la planta de forma automática y se optimizan las comunicaciones.

3.10.1. Mejoras en la página web

Los cambios se empezaron por el diseño de la página web. Se realizó una ventana donde se muestran “iconos de aplicaciones” para representar los tipos de control a implementar.

```
<!--Aplicaciones-->
    <div class = "Apps" id = "apps">
        <div class = "titulo5">Controladores</div>
      <div class = "elementos3" id = "elementos3">
        <div class = "Ap_bomba" id = "Ap_bomba">   <!-- ON / OFF Bomba -->

          <div class = "ap_b"><img src="../Imagenes/Icono bomba.png" width="45px"></img></div>
          <div class ="Nombre_ApB">Bomba</div>
        </div>
        <div class = "Ap_valvula" id = "Ap_valvula">  <!-- ON / OFF Válvula -->
          <div class = "ap_v"><img src="../Imagenes/Icono valvula.png" width="50px"></img></div>
          <div class ="Nombre_ApV">Válvula</div>

        </div>

        <div class = "Ap_Mix" id = "Ap_mix"> <!-- ON / OFF Propio -->
            <div class = "ap_byv"><img src="../Imagenes/Icono_byv.png" width="60px"></img></div>
            <div class ="Nombre_ApMix">Bomb + Val</div>

        </div>
        <div class = "Ap_Manual" id = "Ap_manual"> <!-- Modo manual -->
          <div class = "ap_m"><img src="../Imagenes/Icono palanquita.png" width="20px"></img></div>
          <div class ="Nombre_ApM">Manual</div>

        </div>
      </div>

    </div>
```



Código 19. Aplicaciones.

Según la aplicación elegida esta se resalta haciéndose un poco más grande y abre una ventana a su derecha. Cada ventana cuenta con una cruz en la esquina para poder cerrarla a voluntad. También las pestañas de tipo automático cuentan con dos inputs, uno para indicar el nivel de agua que se desea alcanzar y otro para indicar la cantidad de histéresis que se le aplica al controlador ON / OFF.

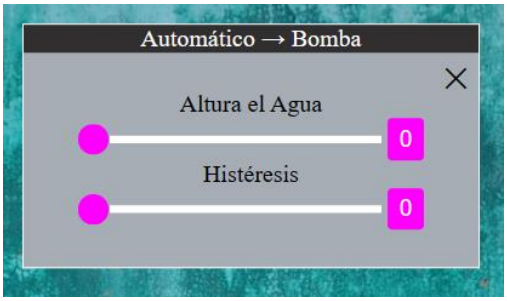


Figura 73. Ventana de Bomba.

Figura 74. Ventana de Válvula.

Figura 75. Ventana de Mixto

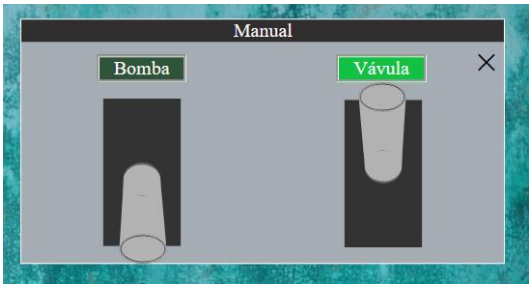


Figura 76. Ventana manual.

Otra mejora añadida son el indicador del nivel que se desea alcanzar en el esquema del estanque y una según recta en el gráfico que recoge las distintas alturas del agua que se marcaron.

Figura 77. Indicador de nivel en el estanque.

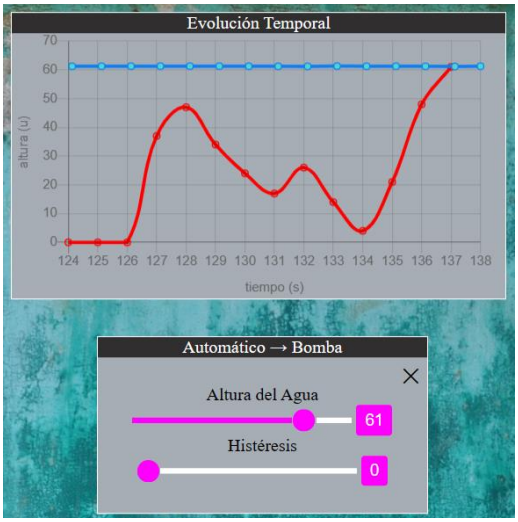


Figura 78. Evolución de las alturas marcadas.

3.10.2. Tipos de control implementados

Para manipular la planta se aplicó el controlador ON / OFF de tres formas distintas y una forma de activar manualmente las palancas de forma remota.

- **ON / OFF para la bomba:** En este modo maneja únicamente el encendido y apagado de la bomba para alcanzar la altura de agua marcada. De esta forma se obtiene un control preciso, pero poco estable.

```
function Comparar2() {
  e = consigna - altura;
  Deltasup = margen;
  Deltainf = - margen;
  console.log ("El error es: ", e);

  console.log("Comparar2 activado");

    if (e < Deltainf) {
      estado = 0;
      b.digitalWrite(Bomba, 0);
      console.log("Estado ", estado);
    } else if (e > Deltasup) {
      estado = 1;
      b.digitalWrite(Bomba, 1);
      console.log("Estado ", estado);
   } else if ((estado == 1) && (e <= Deltainf)) {
     b.digitalWrite(Bomba, 0);
     estado = 0;

   } else if ((estado == 0) && (e >= Deltasup)) {
     b.digitalWrite(Bomba, 1);
     estado = 1;

   }

}
```

Código 20. Control ON / OFF Bomba.

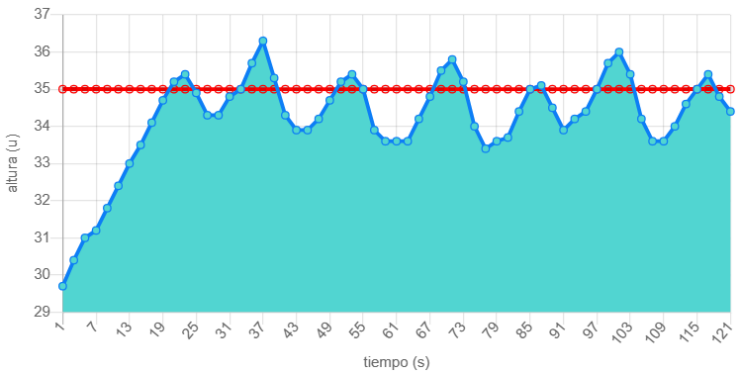


Figura 79. Estabilidad del controlador Bomba

- **ON / OFF para la válvula:** Activando este modo se deja encendida la bomba y se regula la cantidad de agua en el estanque activando o desactivando la palanca de la válvula. Con lo cual se consigue un control estable en el tiempo pero muy lento y poco preciso.

```
function Comparar3() {
  e = consigna - altura;
  Deltasup = margen;
  Deltainf = - margen;
  console.log ("El error es: ", e);

  console.log("Comparar3 activado");

    b.digitalWrite(Bomba, 1);

    if (e < Deltainf) {
      estado = 0;
      b.digitalWrite(Val_Esc, 1);

    } else if (e > Deltasup) {
      estado = 1;
      b.digitalWrite(Val_Esc, 0);

   } else if ((estado == 1) && (e <= Deltainf)) {
     b.digitalWrite(Val_Esc, 1);
     estado = 0;

   } else if ((estado == 0) && (e >= Deltasup)) {
     b.digitalWrite(Val_Esc, 0);
     estado = 1;

   }

}
```

Código 21.Control ON / OFF Válvula.

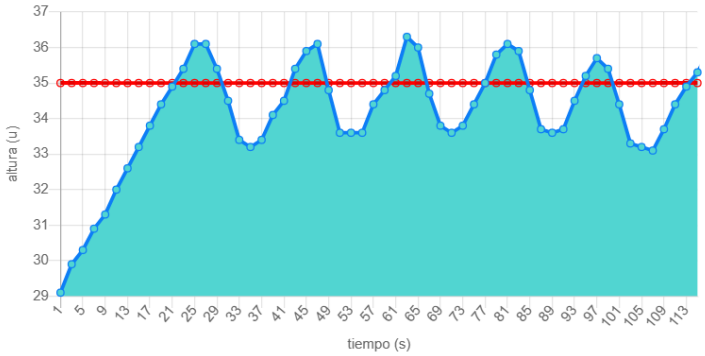


Figura 80. Estabilidad del controlador Válvula.

- **ON / OFF mixto:** Este modo se desarrolló mediante la investigación del funcionamiento de la planta. Se ha comprobado que la mayor estabilidad se consigue cuando la bomba y la válvula están activos al mismo tiempo, causando que el caudal de agua que sale es levemente mayor que el caudal de entrada.

```
function Comparar() {
 console.log("Comparar activado");

     if (consigna > limite_superior) {
         b.digitalWrite(Val_Esc, 0);
         b.digitalWrite(Bomba, 1);
         console.log("subiendo");
         console.log("La consigna es: ", consigna);
         console.log("El limite superior es: ", limite_superior);
     } else if (consigna < limite_inferior) {
         b.digitalWrite(Val_Esc, 1);
         b.digitalWrite(Bomba, 0);
         console.log("bajando");
         console.log("La consigna es: ", consigna);
         console.log("El limite inferior es: ", limite_inferior);
     } else if (limite_inferior < consigna < limite_superior) {
         b.digitalWrite(Val_Esc, 1);
         b.digitalWrite(Bomba, 1);
         console.log("me mantengo");
         console.log("La consigna es: ", consigna);
         console.log("Los límites son: ", limite_inferior, " y " , limite_superior);
     }


}
```

Código 22. Control ON /OFF Mixto.

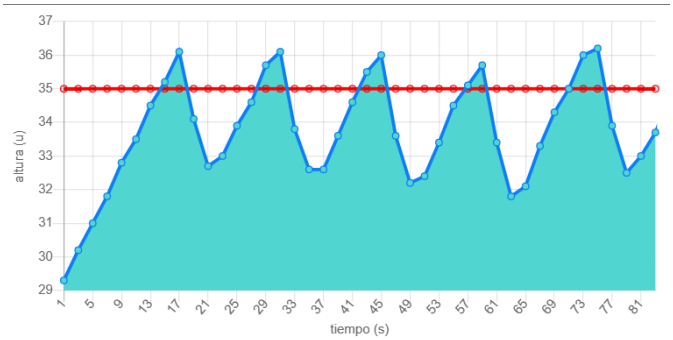


Figura 81. Estabilidad aproximada del controlador Mixto.

- **Control manual:** Se manipula los pines conectados al estanque para mandar señales de encendido o apagado diferenciados por mensajes.

```
} else if (topic == "Manual") {
  mensaje2 = "Manual";
  console.log("Modo manual, activado");
  consigna = 0;
  stopAll();

  if (payload.toString() == "B1") {
        b.digitalWrite(Bomba, 1);
  } else if (payload.toString() == "B0") {
        b.digitalWrite(Bomba, 0);
  } else if (payload.toString() == "V1") {
        b.digitalWrite(Val_Esc, 1);
  } else if (payload.toString() == "V0") {
        b.digitalWrite(Val_Esc, 0);
  }
```

Código 23. Control manual.

- **Control PID:** Este controlador fue diseñado pero no se pudo llegar a implementar por falta de tiempo. La finalidad de este consiste en, una vez alcanzado el nivel de agua deseada, aplicarse para mantener de forma más constante dicho nivel.

```
var Kp = 0;
var Ki = 0;
var Kd = 0;
var ek = 0;
var en = 0; // en = ek-1
var tk = 0;
var tn = 0; // tn = tk-1
var Sumerror = 0;

function RegularCaudal() {
  // cálculo del error
  ek = consigna - altura;
  Sumerror = Sumerror + ek;

  // Cálulo parte proporcional
  const Proporcional = Kp * ek;

  // Cálculo parte integral
  const Integral = Ki * Sumerror * 1; // tiempo de muestreo = 1 segundo

  // acumular errores

  // Cálculo parte derivativa
  tk = tk + 1;
  const T = tk - tn;
  const Derivativa = Kd *(ek - en)/ T;

  // Salida PID
  const ut = Proporcional + Integral + Derivativa;

  // Añadir saturación
  if (ut <= 0) {
    b.digitalWrite(Bomba, 1);
    b.analogWrite(outputPin, ut);

  } else if (ut >= 1) {
    b.digitalWrite(Bomba, 1);
    b.analogWrite(outputPin, ut);

  } else if ((ut > 0) && (ut < 1)) {
    b.digitalWrite(Bomba, 1);
    b.analogWrite(outputPin, ut);

  }

  en = ek;
  tn = tk;
}
```

Código 24. Control PID.

3.10.3. Optimización de comunicaciones

En la nueva actualización se mejora la velocidad de las comunicaciones enviando los mensajes a distintos tópicos según la acción que se realiza en vez de distinguir por el contenido de los propios mensajes. Logrando así una disminución de líneas de comandos en el código y una mejora en el entendimiento de este.

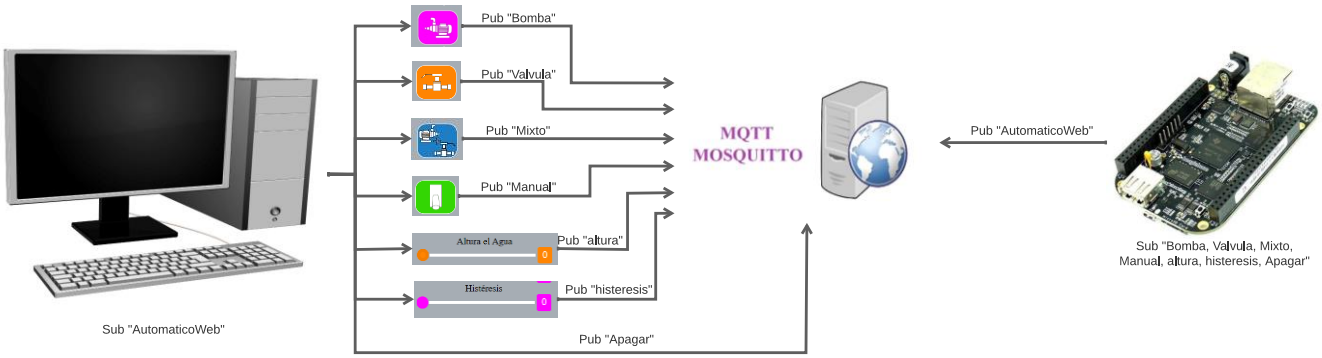


Figura 82. Nuevas comunicaciones.

Tabla 4. Variables de la nueva comunicación.

Tópicos	Descripción	Mensaje que recibe
Bomba	Para activar el control automático On-off aplicado a la bomba.	Recibe el mensaje de "Automático2" para activar este modo y desactivar los otros.
Valvula	Para activar el control automático On-off aplicado a la vávula.	Recibe el mensaje de "Automático3" para activar este modo y desactivar los otros.
Mix	Para activar el control automático On-off inventado.	Recibe el mensaje de "Automático" para activar este modo y desactivar los otros.
Manual	Para activar el control manual	Recibe el mensaje de "Manual" para activar este modo y desactivar los otros. Una vez en este modo distingue los siguientes mensajes. • B1 → Para encender la bomba. • B0 → Para apagar la bomba. • V1 → Para encender la válvula.

		• V0 → Para apagar la válvula.
histeresis	Para manipular la histéresis de los controladores On-off	Recibe el valor del input asignado para manipular esta variable.
altura	Para designar la altira del nivel del agua en los modos automáticos.	Recibe el valor del input asignado para manipular esta variable.
AutomáticoWeb	Para representar gráficamente el valor actual del nivel del agua y su evolución temporal.	Recibe el valor de la altura leído por la BBB para representarlo.
Apagar	Para poner el sistema a cero y desactivar todo	Recibe el mensaje de "Apagar" para desactivar la planta y sus controladores.

3.10.4. Cambios en el Backend

Para finalizar el proyecto se incorpora la nueva modalidad de comunicaciones al backend y se implementan los controladores anteriormente explicados.

Para evitar fallos a la hora de cambiar el modo de control se procede a pasar por cero, cambiando la consigna por este valor y apagando la bomba y válvula. Así se asegura la planta contra posibles desbordamientos.

La lógica implementada es la siguiente que se muestra en la Figura 83.

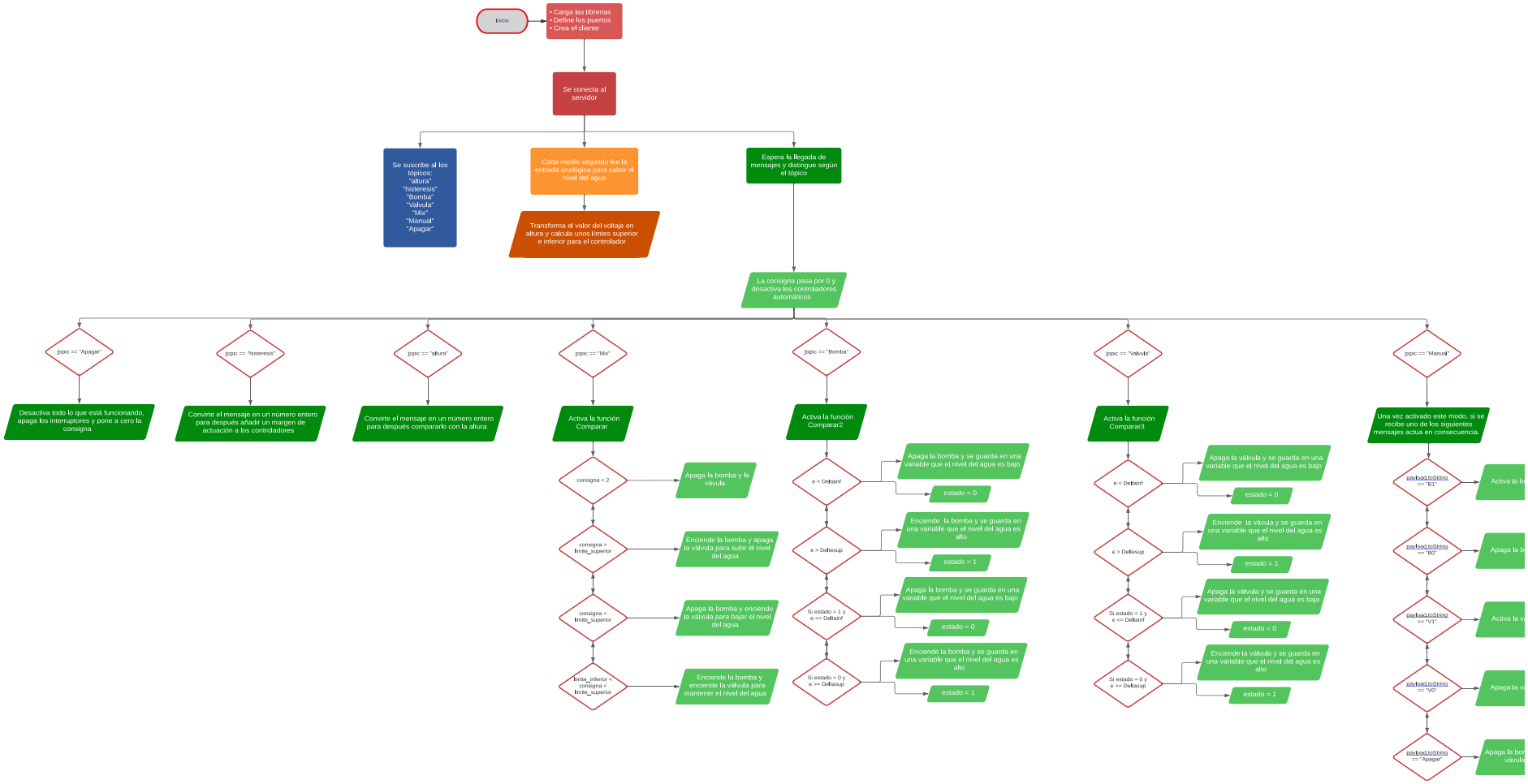


Figura 83. Diagrama de flujo EstanqueV2

Capítulo 4: Conclusiones

4.1. Conclusiones

A través de este estudio, hemos concluido que la aplicación de la tecnología web en el desarrollo de un sistema SCADA (Supervisory Control and Data Acquisition) presenta diversas ventajas y beneficios.

En primer lugar, la tecnología web permite una mayor flexibilidad y accesibilidad en comparación con los sistemas tradicionales SCADA. Gracias a la interfaz web, los usuarios pueden acceder al sistema desde cualquier ubicación a través de internet, lo que facilita el monitoreo y el control en tiempo real.

Además, el uso de la tecnología web en un sistema SCADA simplifica la implementación y reduce los costos asociados. Al estar basado en estándares web abiertos, como HTML, CSS y JavaScript, es más fácil desarrollar, mantener y actualizar el sistema. Además, la tecnología web es compatible con una amplia gama de dispositivos, lo que significa que no se requiere hardware especializado ni software propietario costoso.

Otra conclusión importante es que la aplicación de la tecnología web en un sistema SCADA permite una mayor colaboración entre diferentes equipos y departamentos dentro de una organización. La información recopilada y presentada a través del sistema SCADA puede ser compartida fácilmente en tiempo real, lo que mejora la toma de decisiones y la eficiencia operativa.

Sin embargo, también hemos identificado algunos desafíos y consideraciones importantes. La seguridad es un factor crítico en cualquier sistema SCADA, y la implementación de la tecnología web debe tener en cuenta los protocolos de seguridad adecuados para proteger la información y evitar ataques cibernéticos.

El objetivo final de este proyecto era la implementación de un sistema SCADA en un entorno web, para ello se han dado los siguientes pasos:

Se han adquirido conocimientos de las tecnologías WEB (HTML, CSS, JavaScript) mediante cursos y tutoriales disponibles en internet.

Para familiarizarse con el uso de la BeagleBone Black se aprendió sobre la librería bonescript y el uso de sus diferentes pines.

Con el fin de establecer una comunicación máquina-máquina se aprendió sobre MQTT y su gran versatilidad.

Se diseñaron e implementaron varios controladores ON / OFF para manipular remotamente la planta e incluso se creó uno propio.

Se desarrollaron varias páginas web con sistemas SCADA cada vez más complejos con la finalidad de demostrar la adaptabilidad de los sistemas SCADA WEB

Por último, se diseñó un control PID que no se llegó a implementar debido a la falta de tiempo.

4.2. In conclusión

Through this study, we have concluded that the application of web technology in the development of a SCADA (Supervisory Control and Data Acquisition) system presents several advantages and benefits.

First, web technology allows greater flexibility and accessibility compared to traditional SCADA systems. Thanks to the web interface, users can access the system from any location via the Internet, which facilitates real-time monitoring and control.

In addition, the use of web technology in a SCADA system simplifies implementation and reduces associated costs. Being based on open web standards, such as HTML, CSS and JavaScript, it is easier to develop, maintain and update the system. In addition, web technology is compatible with a wide range of devices, which means that specialized hardware and expensive proprietary software are not required.

Another important conclusion is that the application of web technology in a SCADA system enables greater collaboration between different teams and departments within an organization. Information collected and presented through the SCADA system can be easily shared in real time, which improves decision making and operational efficiency.

However, we have also identified some important challenges and considerations. Security is a critical factor in any SCADA system, and the implementation of web technology must take into account proper security protocols to protect information and prevent cyber attacks.

The final objective of this project was the implementation of a SCADA system in a web environment, for this purpose the following steps have been taken:

Knowledge of WEB technologies (HTML, CSS, JavaScript) has been acquired through courses and tutorials available on the internet.

In order to become familiar with the use of the BeagleBone Black we learned about the bonescript library and the use of its different pins.

In order to establish machine-to-machine communication we learned about MQTT and its great versatility.

Several ON/OFF controllers were designed and implemented to remotely manipulate the plant and even created one of their own.

Several web pages with increasingly complex SCADA systems were developed in order to demonstrate the adaptability of SCADA WEB systems.

Finally, a PID control was designed but not implemented due to lack of time.

Capítulo 5: Líneas abiertas

Comenzando por el propio sistema y su desarrollo, se plantea implementar un sistema de seguridad que restrinja ciertas funciones a los usuarios del SCADA según su rol en la planta industrial. Por ejemplo, un supervisor debería tener acceso a todos los parámetros de control, mientras que a un operario o a cualquier persona sin los conocimientos necesarios no se le debería permitir modificar los parámetros relacionados con el control de la planta, ya que esto podría ocasionar errores en la línea de producción. De esta manera, se garantiza un buen funcionamiento de la actividad industrial.

Además, sería conveniente contar con una base de datos donde se almacenen todos los datos recopilados, ya que actualmente se guardan en el navegador y se eliminan al cerrar la pantalla. Si logramos esto, sería posible crear gráficas que muestren el historial de la planta y se podría acceder a ellos mediante botones de acceso directo.

Asimismo, se debería incluir un panel de configuración más completo en el sistema. En este proyecto, no se permitió al usuario elegir el intervalo de muestreo de la señal, pero esto podría ser un parámetro flexible en función de la planta que se desee monitorear en el futuro.

También se podría trabajar en un diseño adaptativo (responsive design) para poder acceder al sistema desde cualquier tamaño de pantalla y que los elementos se adapten y recoloquen adecuadamente.

En cuanto a posibles mejoras basadas en este mismo proyecto, sería importante implementar un sistema de alarmas para evitar accidentes o fallos.

Capítulo 6: Presupuesto

Para la realización del proyecto se han utilizado los siguientes medios materiales y humanos detallados en el presupuesto que se muestra a continuación cuyo valor total asciende a 8.868,55 €.

Tabla 5. Presupuesto.

Cantidad	Material	Coste
1	BeagleBone Black	48,57 €
1	Protoboard	11,77 €
1	Conversor 4-20 mA a 3,3 V	1,02 €
1	Conversor 3,3 V a 4-20 mA	0,22 €
1	Ordenador portátil	400 €
3	Led	4,77 €
3	Resistencia de 220 ohms	4 €
1	Pulsador	0,50 €

Cantidad	Personal	Saldo por hora	Tiempo dedicado	Coste
1	Ingeniero	28 €/h	300 h	8.400,00 €
Total:				8.870,85 €

Capítulo 7: Bibliografía

[1] "SCADA", *Wikipedia* ,21 de Octubre de 2022. Disponible en: SCADA - Wikipedia, la enciclopedia libre. (Accedido: 08 de Junio de 2023).

[2] [Interfaz Humano-Máquina]. (2014). Disponible en: TICS- INDUSTRIA ELÉCTRICA: SISTEMA HIM (Interfaz Hombre-Máquina) (kevinarmastics.blogspot.com). (Accedido: 08 de Junio de 2023).

[3] "Sistema SCADA: en qué consisten, funciones e importancia en la Industria 4.0", *UNIR La Universidad de Internet*. Disponible en: ¿Qué es un sistema SCADA?: su importancia en la industria 4.0 (unir.net). (Accedido: 08 de Junio de 2023).

[4] [Componentes de un sistema SCADA]. (2019). Disponible en: Qué es un sistema SCADA, para qué sirve y cómo funciona (cursosaula21.com). (Accedido: 08 de Junio de 2023).

[5] Frontend y backend: qué son, en qué se diferencian y ejemplos, "*HubSpot*". Disponible en: Frontend y backend: qué son, en qué se diferencian y ejemplos (hubspot.es). (Accedido: 09 de Junio de 2023).

[6] [Frontend y Backend]. (24 de Julio de 2019). Disponible en: Unidad 1: Definición del FRONTEND & BACKEND (groupceneco.blogspot.com). (Accedido: 11 de Julio de 2023).

[7] "Qué es HTML", *desarrolloweb.com,* 01 de Enero de 2001. Disponible en: Qué es HTML (desarrolloweb.com). (Accedido: 09 de Junio de 2023).

[8] "Estructura de una página HTML5", *desarrolladoresweb.org*. Disponible en: ▷Estructura de una página HTML5 - Desarrolladores web. (Accedido: 09 de Junio de 2023).

[9] "¿Qué es el JavaScript?", *MDM Web Docs*. Disponible en: ¿Qué es JavaScript? - Aprende desarrollo web | MDN (mozilla.org). (Accedido: 14 de Junio de 2023).

[10] "Atributo HTML <script> src", *w3schools. Disponible en:* Script HTML src Atributo (w3schools.com). (Accedido: 14 de Junio de 2023).

[11] "¿Qué es el DOM?", *lenguajejs*. Disponible en: ¿Qué es el DOM? - Javascript en español - Lenguaje JS. (Accedido: 15 de Junio de 2023).

[12] "Qué es SVG", *desarrolloweb,* 29 de Enero de 2020. Disponible en: Qué es SVG (desarrolloweb.com). (Accedido: 15 de Junio de 2023).

[13] "¿Qué es Node.js, y para qué sirve?", *itdo.* 27 de Julio de 2021. Disponible en: ¿Qué es Node.js, y para qué sirve? (itdo.com). (Accedido: 16 de Junio de 2023).

[14] [BeagleBone Black hardware details]. Disponible en: BeagleBoard.org - bone101. (Accedido: 17 de Junio de 2023).

[15] [The expansion headers provide extensive I/O capabitilities]. Disponible en: BeagleBoard.org - bone101. (Accedido: 17 de Junio de 2023).

[16] "IT Explained MQTT", *Paessler.* Disponible en: ¿Qué es MQTT? Definición y detalles (paessler.com). (Accedido: 17 de Junio de 2023).

[17] "Introducción a MQTT y configuración de un Broker Mosquitto", *Ichi.pro*, 2020-2023. Disponible en: Introducción a MQTT y configuración de un Broker Mosquitto (ichi.pro). (Accedido: 17 de Junio de 2023).

[18] "¿Qué son los topics en mqtt?", *Murky Robot*. Disponible en: MQTT - Como utilizar bien los topics - Murky Robot. (Accedido: 11 de Julio de 2023).

[19] "Uso de MQTT sobre WebSockets con Mosquitto", *Steve´s Internet Guide*, 08 de Abril de 2023. Disponible en: Uso de MQTT sobre WebSockets con Mosquitto (steves-internet-guide.com). (Accedido: 17 de Junio de 2023).

[20] [Example]. (06 de Diciembre de 2022). Disponible en: GitHub - mqttjs/MQTT.js: The MQTT client for Node.js and the browser. (Accedido: 19 de Junio de 2023).

[21] "MQTT Explorer", *MQTT Explorer.* Disponible en: MQTT Explorer | Un cliente MQTT completo que proporciona una visión general estructurada del tema (mqtt-explorer.com). (Accedido: 19 de Junio de 2023).

[22] "Level/Flow Process Control", *feedback-instruments*. Disponible en: Microsoft Word - 38-001_datasheet_level_flow_control_ESPIAL_10_2013.doc (feedback-instruments.com). (Accedido: 19 de Junio de 2023).

[23] "Control ON/OFF o Todo/Nada", *InstrumentacionyControl.net*. Disponible en: Control ON/OFF o Todo/Nada - Instrumentacion y Automatizacion Industrial (instrumentacionycontrol.net). (Accedido: 19 de Junio de 2023).

[24] [Ideal on off control system], *sosteneslekule.blogspot.com.* Disponible en: On Off Control Theory,Controller - LEKULE (sosteneslekule.blogspot.com). (Accedido: 19 de Junio de 2023).

[25] "Controlador PID", *Wikipedia*, 03 de Mayo de 2023. Disponible en: Controlador PID - Wikipedia, la enciclopedia libre. (Accedido: 21 de Junio de 2023).

[26] FalconMasters. (30 de Mayo de 2014). *Curso Básico de HTML desde 0*. [Archivo de Vídeo]. Youtube. Curso Básico de HTML desde 0 - Introducción - YouTube (Accedido: 22 de Junio de 2023).

[27] W3School. Disponible en: W3Schools Online Web Tutorials.

[28] FreeLogoDesign. Disponible en: Manage | FreeLogoDesign.

[29] Pixabay. Disponible en: Más de 1 millón de Imágenes Gratis para Descargar - Pixabay - Pixabay.

[30] FalconMasters. (17 de Junio de 2014). *Curso Básico de CSS desde 0*. [Archivo de Vídeo]. Youtube. Curso Básico de CSS desde 0 - Introducción - YouTube (Accedido: 22 de Junio de 2023).

[31] HTMLcolorscode. Disponible en: Códigos de Colores HTML (htmlcolorcodes.com).

[32] Molloy, Derek (2019). Interfaz con las entradas/salidas del tablero Beagle. Wiley. *Explorando BeagleBone, 2ª Edición*. Disponible en: CAPÍTULO 6: Interfaz con las entradas/salidas del tablero Beagle | Explorando BeagleBone, 2ª Edición (oreilly.com).

[33] mqtt - Libraries - cdnjs - The #1 free and open source CDN built to make life easier for developers.

[34] Using The JavaScript MQTT Client With Websockets (steves-internet-guide.com).

[35] GitHub - mqttjs/MQTT.js: The MQTT client for Node.js and the browser.

[36] Chart.js, *W3School*. Disponible en: Chart.js (w3schools.com). (Accedido: 27 de Junio de 2023).

Capítulo 8: Anexos

Anexo I: Datasheet

Para no incrementar el número de páginas de este trabajo se incorporan los datasheet de los convertidores con un enlace web.

- Conversor 4-20mA a 3.3V: Módulo de corriente a voltaje, transmisor de señal, placa convertidora, 0 20Ma, 4 20 mA a 0 3,3 V, 0 5V, 0 10V|Circuitos integrados| - AliExpress
- Conversor 3.3V a 4-20mA: Módulo de voltaje a corriente 0 2,5 V 0 3,3 V 0 5V 0 10V 0 0 15V 24V a 0 20mA/4 20mA, transmisor de corriente, convertidor de señal módulo|Circuitos integrados| - AliExpress

Anexo II: Código del proyecto

A lo largo del proyecto se han mostrado fragmentos de código, en el siguiente enlace de github se encuentra todo el contenido y versiones de lo programado.

Universidad-de-La-Laguna/TFG_Guarirai_SCADA: Código del TFG de Guarirai Tomé Noda (github.com)

Anexo III: Manual de usuario

Manual de usuario.

Con el fin de facilitar el uso del proyecto y su correcto funcionamiento se ha realizado el siguiente manual de instrucciones.

1. Conectar la BeagleBone Black por USB a un ordenador
2. Esperar a que la placa se inicie y su muestre la siguiente ventana en pantalla.

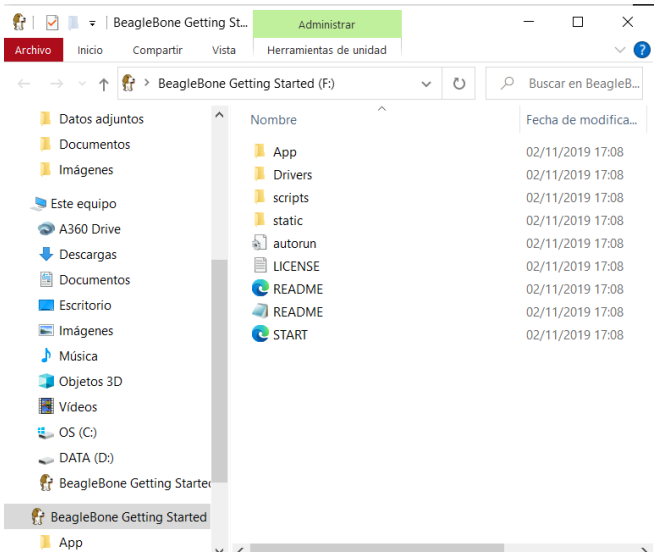


Figura 84. Inicio BBB

3. Para acceder al entorno Cloud9, ir al navegador y escribir la siguiente dirección IP → 192.168.7.2

4. Buscar la carpeta "Mispruebas" y ejecutar el archivo "ControlEstanqueV2_topics.js".

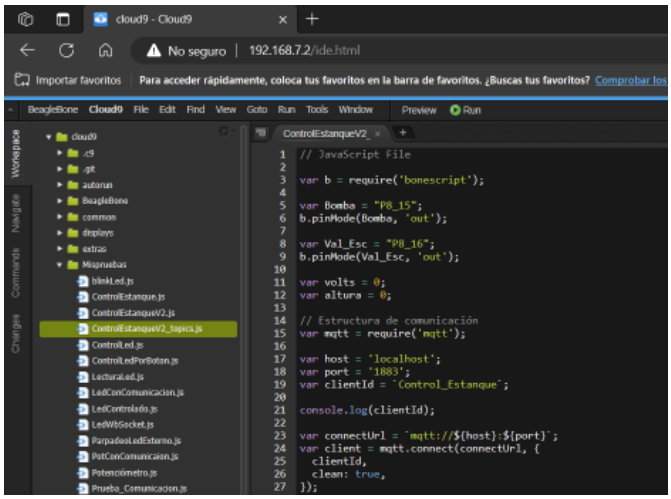


Figura 85. Archivo a ejecutar.

5. Para acceder a la página web abrir una pestaña nueva y escribir la dirección IP anterior con el siguiente puerto → 192.168.7.2:8080

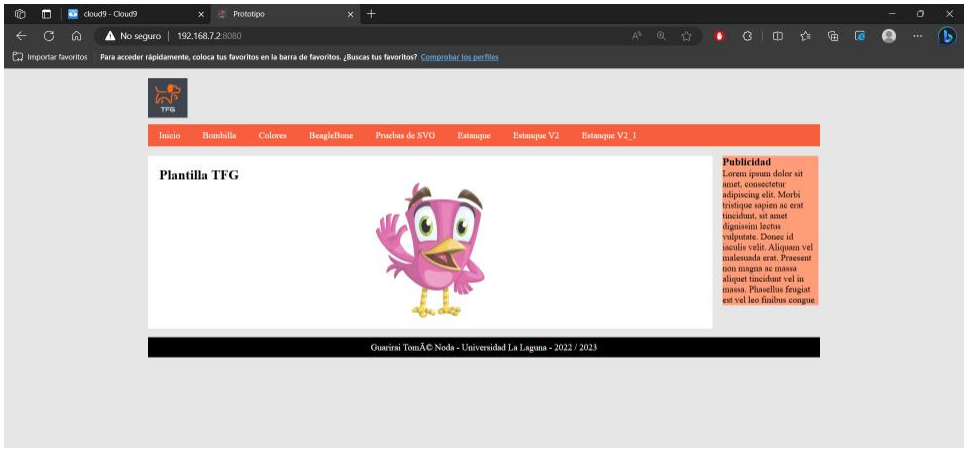


Figura 86. Página de inicio.

6. Una vez en la página de inicio clique en “Estanque V2_1” situado a la derecha del menú de navegación para acceder a los controles de la panta.

Figura 87. Seleccionar.

7. Espere 2 segundos a que se carguen los elementos de la página antes de encender la tablet.
8. Una vez encendida tiene acceso a los siguientes modos de control.
   - Tres de tipo automático.

Figura 88. Controladores automáticos.

Estos controladores destacan por tener los mismos controles, pero es distintos colores según el seleccionado.

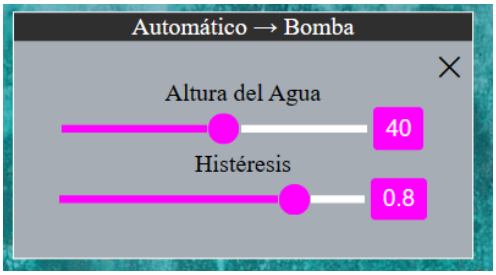


Figura 89. Controles automáticos.

Desde estos modos usted puede indicar el nivel de agua que desea en el estanque y la velocidad de acción del controlador seleccionado (histéresis).

También cuenta con una cruz en la esquina superior derecha para cerrar la ventana si lo desea. Tenga en cuenta que al cerrar la ventana el sistema pasará por un cero para evitar desbordamientos.

- Uno de tipo manual.

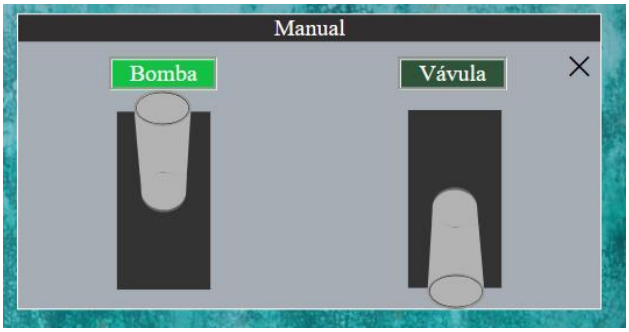


Figura 90. Controles manuales.

En este modo usted el libre de encender/ apagar la bomba o abrir/cerrar la válvula de escape.

9. Antes de salir recuerde apagar la tablet para deshabilitar los controladores y dejar apagado los interruptores.